LATENT FACTOR MODELS FOR STATISTICAL RELATIONAL LEARNING

by

WU-JUN LI

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in Computer Science and Engineering

July 2010, Hong Kong

Authorization

WU-JUN LI

LATENT FACTOR MODELS FOR STATISTICAL RELATIONAL LEARNING

by

WU-JUN LI

This is to certify that I have examined the above Ph.D. thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

PROF. DIT-YAN YEUNG, THESIS SUPERVISOR

PROF. MOUNIR HAMDI, HEAD OF DEPARTMENT

Department of Computer Science and Engineering
30 July 2010

ACKNOWLEDGMENTS

First and foremost, I would like to thank my supervisor, Prof. Dit-Yan Yeung, for his support throughout my years at HKUST. Without his guidance, encouragement, and kindness, this thesis would never have been possible.

I would like to thank my thesis committee members, Prof. Huan Liu (from Department of Computer Science and Engineering, Arizona State University), Prof. Weichuan Yu (from Department of Electronic and Computer Engineering, HKUST), Prof. Nevin L. Zhang, Prof. James Kwok, and Prof. Dit-Yan Yeung, for their insightful comments and suggestions.

I would also like to thank Prof. Zhihua Zhang (from College of Computer Science and Technology, Zhejiang University) for his constructive discussions and help during my PhD study.

I would also like to thank the members of DY's group for their friendships, encouragements, support and collaboration. They are: Hong Chang, Gang Wang, Guang Dai, Wei Fan, Chris Kui Jia, Yang Ruan, Jingni Chen, Yu Zhang, Yi Zhen, Craig Yu, Tony Ho.

Finally, I am deeply indebted to my family whose love and support are the source of my courage.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LATENT FACTOR MODELS FOR STATISTICAL RELATIONAL LEARNING

by

WU-JUN LI

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

To simplify modeling procedures, traditional statistical machine learning methods always assume that the instances are independent and identically distributed (i.i.d.). However, it is not uncommon for some real-world data, such as web pages and research papers, to contain relationships (links) between the instances. Different instances in such data are *correlated* (linked) with each other, which implies that the common i.i.d. assumption is unreasonable for such *relational data*. Hence, naively applying traditional statistical learning methods to relational data may lead to misleading conclusions about the data.

Statistical relational learning (SRL), which attempts to perform learning and inference in domains with complex relational structures, has become an emerging research area because relational data widely exist in a large variety of application areas, such as web mining, social network analysis, bioinformatics, economics and marketing. The existing mainstream SRL models extend traditional graphical models, such as Bayesian networks and Markov networks, by eliminating their underlying i.i.d. assumption. Some typical examples of such SRL models include *relational Bayesian networks*, *relational Markov networks*, and *Markov logic networks*. Because the dependency structure in relational data is typically very complex, structure learning for these relational graphical models is often very time-consuming. Hence, it might be impractical to apply these models to large-scale relational data sets.

In this thesis, we propose a series of novel SRL models, called *relational factor models* (RFMs), by extending traditional *latent factor models* from i.i.d. domains to relational domains. These proposed RFMs provide a toolbox for different learning settings: some of them are well suited for transductive inference while others can be used for inductive inference; some of them are parametric while others are nonparametric; some of them can be used to model data with undirected relationships while others can be used for data with directed relationships. One promising advantage of our RFMs is that there is no need for time-consuming structure learning and the time complexity of most of them is linear to the number of observed links in the data. This implies that our RFMs can be used to model large-scale data sets. Experimental results show that our models can achieve state-of-the-art performance in many real-world applications such as linked-document classification and social network analysis.

CHAPTER 1

INTRODUCTION

To simplify modeling procedures, traditional statistical machine learning (ML) methods are always based on two assumptions about the data. One assumption is that all the instances have the same set of attributes and can be represented as feature vectors all of the same dimensionality. Data represented in this form are referred to as *flat data* or *propositional data* (Getoor & Taskar, 2007). The other assumption is the so-called i.i.d. assumption, which means that the instances are assumed to be independent and identically distributed (i.i.d.). [1]

However, these two assumptions in traditional ML are unreasonable for *relational data* (Getoor & Taskar, 2007) in which the instances are linked (i.e., related) to each other. *Statistical relational learning* (SRL) (Getoor & Taskar, 2007), which tries to perform learning and inference from relational data, has attracted many researchers' interests recently due to its wide applications. This thesis focuses on SRL problems.

In the following content of this chapter, we first introduce some basic concepts about relational data and SRL. Then, we present the motivation and summarize the main contributions of this thesis. Finally, we outline the organization of the whole thesis.

1.1 Relational Data

Relational data refer to those kinds of data in which the instances are linked (i.e., related) to each other. The instances in a relational data set can either be homogeneous or heterogeneous. For example, a data set collected from a university may contain professors, students, courses, and their corresponding attribute information. Obviously, the instances in this data set are heterogeneous. There exist many different kinds of relationships between the instances. For example, there are *supervising* relationships between professors and students, and there are *taking* relationships between students and courses. If we consider a subset which only consists of the student instances, we can say that the instances in this subset are homogeneous. In this subset of students, there also exist many relationships, such as the *friendship* relationships between the students.

[1]We use the terms 'instances', 'entities' and 'objects' interchangeably throughout this thesis.

1.1.1 Representation

As in (Singh, 2009), we use similar terminologies as those in entity-relationship models (Chen, 1976) to describe a relational data set. More specifically, a relational data set contains a set of entities (i.e., instances) which can be divided into T different types $(\mathcal{E}_1, \mathcal{E}_2, \cdots, \mathcal{E}_T)$, and a set of relations[2] each of which can be represented by a matrix. A relation can exist between either the entities of the same type or the entities of two different types. For example, in the above data set of a university, there are at least three entity types: professors $(\mathcal{E}_1)$, students $(\mathcal{E}_2)$, and courses $(\mathcal{E}_3)$. Assume there are n_i entities of type $\mathcal{E}_i$. The *friendship* relation between students can be represented by an $n_2 \times n_2$ matrix whose (i, j) entry represents the friendship between students i and j. The *supervising* relation between professors and students can be represented by an $n_1 \times n_2$ matrix whose (i, j) entry represents whether professor i is a supervisor of student j. There may exist multiple relations over the same types. For example, besides the *friendship* relation, there also exists a *roommate* relation between students. Like the *friendship* relation, the *roommate* relation can also be represented by an $n_2 \times n_2$ matrix but these two matrices have different entry values. More formally, the kth relation between $\mathcal{E}_i$ and $\mathcal{E}_j$ can be defined as $R_{ij}^k : \mathcal{E}_i \times \mathcal{E}_j \to \mathcal{U}_k$, where $\mathcal{U}_k$ can be $\mathbb{R}$ or other sets like $\{0, 1\}$. Note that here we overload $\mathcal{E}_i$ to refer to the set of entities of type $\mathcal{E}_i$, and $\mathcal{E}_i \times \mathcal{E}_j$ denotes the Cartesian product between two sets $\mathcal{E}_i$ and $\mathcal{E}_j$. Although in real applications, there may exist relations with order higher than two, such as the three-order relation $R_{ijp}^k : \mathcal{E}_i \times \mathcal{E}_j \times \mathcal{E}_p \to \mathcal{U}_k$, in this thesis we only focus on binary relations like R_{ij}^k between entities.

From the above definition of a relation, we can divide relations into the following three categories:

- Undirected relation: An undirected relation can only be defined over the same type of entities. An undirected relation is also called a symmetric relation because $\mathbf{A}(i, j) = \mathbf{A}(j, i)$ if $\mathbf{A}$ is the matrix representing an undirected relation. For example, the *roommate* relation is an undirected relation because if i is a roommate of j, j is also a roommate of i.

- Directed relation: A directed relation can only be defined over the same type of entities. A directed relation is also called an asymmetric relation because typically $\mathbf{A}(i, j) \neq \mathbf{A}(j, i)$ if $\mathbf{A}$ is the matrix representing a directed relation. For example,

[2]Note that a relationship refers to a link or edge between two instances, while "a relation" refers to a set of relationships between either the entities of the same type or the entities of two different types, which is similar to "a table" in database terminology. This definition of a relation is also similar to that in (Kemp et al., 2006; Sutskever et al., 2009)

a paper citation relation is a directed relation because a paper can be cited by but cannot cite papers which are published later than it.

- Bipartite relation: In a bipartite relation, there are two different types of entities. Relationships can only exist from one type to the other. For example, the *supervising* and *taking* relations in the above university case are bipartite relations. Another example is from collaborative filtering (CF) (Adomavicius & Tuzhilin, 2005; Zhen et al., 2009). In CF, there are two types of entities, one being users and the other being items. The entries in the relation matrix record the rating values on the items by the users.

In flat data, the instances are always represented as feature vectors all of the same dimensionality. A set of these instances can be represented by an $n \times d$ matrix where n is the number of instances and d is the number of attributes. Hence, a flat data set can be seen as a bipartite relation if we treat the attributes as another type of entities. Similarly, the attribute information in relational data can also be treated as bipartite relations. Hence, a relational data set can be fully represented by a set of matrices each of which describes a relation.

1.1.2 Homophily and Stochastic Equivalence

Homophily and *stochastic equivalence* are two primary features of interest in social networks (or called graphs) which contain only one type of entities (Wasserman & Katherine, 1994; Hoff, 2007). A graph containing only one type of entities can be either directed or undirected. If it is directed, it is actually a directed relation. Otherwise, it is an undirected relation. In a graph, if a link is more likely to exist between two nodes with similar characteristics than between those nodes having different characteristics, we say the graph exhibits homophily. For example, two individuals are more likely to be friends if they share common interests. Hence, a friendship graph has the feature of homophily. On the other hand, if the nodes of a graph can be divided into groups where members within a group have similar patterns of links, we say this graph exhibits stochastic equivalence. The web graph has such a feature because some nodes can be described as hubs which are connected to many other nodes called authorities but the hubs or authorities are seldom connected among themselves. For stochastic equivalence, the property that members within a group have similar patterns of links also implies that if two nodes link to or are linked by one common node, the two nodes are most likely to belong to the same group.

Examples of homophily and stochastic equivalence in directed graphs are illustrated in Figure 1.1, where the locations in the 2-dimensional space denote the characteristics of the points (nodes). From Figure 1.1(a), we can see that a link is more likely to exist between two points close to each other, which is the property of homophily. In Figure 1.1(b), the points form three groups associated with different colors, and the nodes in each group share similar patterns of links to nodes in other groups, but the nodes in the same group are not necessarily connected to each other. This is the property of stochastic equivalence. Note that in a graph exhibiting stochastic equivalence, two points close to each other are not necessarily connected to each other and connected points are not necessarily close to each other, which is different from the property of homophily.

In this thesis, we call links in a graph exhibiting homophily *homophily links*, and call links in a graph exhibiting stochastic equivalence *SE links*.

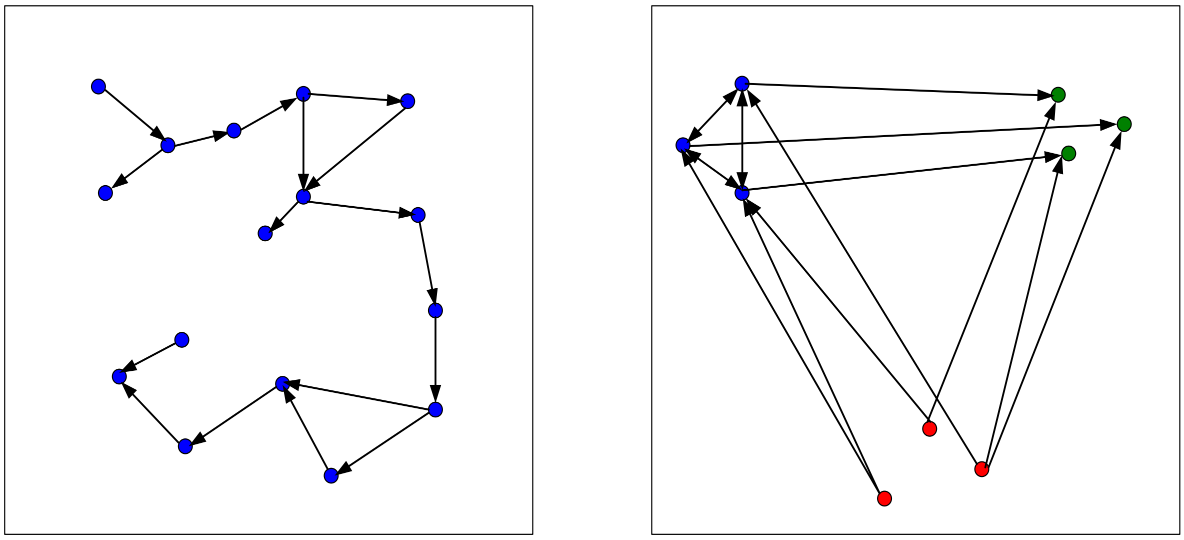

(a) homophily (b) stochastic equivalence

Figure 1.1: Homophily and stochastic equivalence in networks.

1.2 Statistical Relational Learning

Relational data can be found in such diverse application areas as web mining (Chakrabarti et al., 1998; Castillo et al., 2007; Nallapati et al., 2008; Chang & Blei, 2009), bioinformatics (Vert, 2009), social network analysis (SNA) (Wasserman & Katherine, 1994; Chang et al., 2009; Yang et al., 2009a, 2009b), and marketing (Hill et al., 2006, 2007). Both assumptions of traditional statistical ML methods may be violated in relational data. In relational data, there may be *heterogeneous* instances, which means that instances have different types and different types of instances have different kinds of attributes. For example, in the above university data set, the attributes for professors and those for courses are typically different. Furthermore, some instances in such data are *correlated* (i.e., linked) with each other, which implies that the common i.i.d. assumption is unreasonable for such relational data. For example, if two students are friends, it is highly likely that they like to attend similar school activities. Another example is about the paper citation graph, where a citation/reference relationship between two papers provides a very strong evidence for them to belong to the same topic. Hence, naively applying traditional statistical learning methods to relational data may lead to misleading conclusions about the data.

Recently, SRL (Getoor & Taskar, 2007), which tries to perform learning and inference in domains with complex relational structures, has attracted many researchers' interests. Besides the statistical ML community, SRL has also been pursued by many other communities, such as inductive logic programming (ILP) (Muggleton, 1991) community and data mining community. In ILP community, SRL is called *probabilistic logic learning* (Raedt & Kersting, 2003). In data mining community, SRL is always called *multi-relational data mining* (Dzeroski, 2003; Domingos, 2003). Some typical SRL tasks (Getoor & Diehl, 2005) include: collective classification (Macskassy & Provost, 2007; Sen et al., 2008), link prediction (Taskar et al., 2003), link-based clustering (i.e., social community detection) (Wasserman & Katherine, 1994), and collective entity resolution (Bhattacharya & Getoor, 2007) and so on.

Most existing SRL models (Neville, 2006; Getoor & Taskar, 2007) assume that in a relational data set there exists at least one relation, either directed or undirected, defined over the same type of entities. Actually, it is the relations defined over the same type of entities that make the linked instances correlated rather than i.i.d. Hence, in this thesis, unless otherwise stated, we assume there exists at least one relation defined over the same type of entities in a relational data set.

1.3 Motivation

According to the representation formalisms, the existing mainstream SRL models can be divided into two categories (Getoor & Taskar, 2007): logic-based (i.e., rule-based) models and frame-based (i.e., object-oriented) models. The logic-based models are also called *probabilistic logic models* (PLMs), and the frame-based models are also called *probabilistic relational models* (PRMs) (Neville & Jensen, 2007). [3]

PLMs extend traditional ILP models to support probabilistic reasoning in the first-order logic environment. Typical PLMs include *Bayesian logic programs* (BLPs) (Kersting & Raedt, 2001) and *Markov logic networks* (MLNs) (Richardson & Domingos, 2006). BLPs can be viewed as a combination of ILP and Bayesian networks, and MLNs can be

[3]In many papers, such as (Friedman et al., 1999; Getoor et al., 2002), the term *PRMs* refers to a specific kind of models called *relational Bayesian network* in recent literatures (Neville & Jensen, 2007). In this thesis, we use PRMs to refer to the general frame-based relational models, just the same as the convention in (Neville & Jensen, 2007).

seen as a combination of ILP and Markov networks.

PRMs extend traditional graphical models, such as Bayesian networks and Markov networks, by eliminating their underlying i.i.d. assumption. For example, *relational Bayesian networks* (RBNs) (Getoor et al., 2002) adapt Bayesian networks to relational domains, and *relational Markov networks* (RMNs) (Taskar et al., 2002) are relational extensions of Markov networks. Other PRMs include *relational dependency networks* (RDNs) (Neville & Jensen, 2007), and *probabilistic entity-relationship models* (PERs) (Heckerman et al., 2004).

As stated by Cussens James in Chapter 9 of (Getoor & Taskar, 2007), it is very hard (or time-consuming) to perform *structure learning* for these relational graphical models like PLMs and PRMs. Even in traditional graphical models for flat data, structure learning is also very hard. Since the dependency structure in relational data is far more complex than that in flat data, structure learning for PLMs and PRMs will be much harder than that for traditional graphical models. This main drawback of relational graphical models has also been found by (Xu et al., 2006; Xu, 2007) and (Sutskever et al., 2009). Hence, it might be impractical to apply these relational graphical models to large-scale relational data sets.

1.4 Main Contributions

In this thesis, we propose a series of novel SRL models, called *relational factor models* (RFMs), by extending traditional *latent factor models* (LFMs) (Bartholomew & Knott, 1999; Memisevic, 2008) [4] from i.i.d. domains to relational domains.[5] These RFMs are briefly introduced as follows:

- Relation regularized matrix factorization (RRMF): Matrix factorization (MF), such as latent semantic indexing (LSI) (Deerwester et al., 1990), is a very hot topic in machine learning and other related areas due to its wide applications in document analysis, image analysis, and CF. Some relational data, such as web pages and research papers, contain relational (i.e., link) structure among instances in addition to textual content information [6]. Traditional MF methods have been successfully used

[4]In (Bartholomew & Knott, 1999), LFMs are actually called *latent variable models* (LVMs) or *factor analysis*. In this thesis, we use LFMs and LVMs interchangeably.

[5]As stated above, a relational data set can be represented as a set of matrices. In this thesis, we focus on some simple data sets with only one or two matrices. More specifically, if the instances in a data set have attributes, this data set will contain two matrices, one about the attribute information and the other about the relationships between instances. Otherwise, if the instances have no attributes, the data set will contain only one matrix which is about the relationships between instances. Based on our models developed for one or two matrices, it is easy to extend them to model complex relational data sets with more than two matrices, which will be discussed in Chapter 8.

[6]In some chapters of this thesis, we use document classification as a running example for relational

to map either content information or relational information into a lower-dimensional latent space for subsequent processing. However, how to simultaneously model both the relational information and the content information effectively with an MF framework is still an open research problem. We propose a novel MF method, called RRMF, for relational data analysis. By using relational information to regularize the content MF procedure, RRMF seamlessly integrates both the relational information and the content information into a principled framework. We propose a linear-time learning algorithm with convergence guarantee to learn the parameters of RRMF.

- Probabilistic relational PCA (PRPCA): Projection methods, such as principal component analysis (PCA) (Jolliffe, 2002) and probabilistic PCA (PPCA) (Tipping & Bishop, 1999), have been widely used to explore the structure of a high-dimensional data set by mapping the data set into a low-dimensional space via a projection (or called transformation) matrix. One crucial assumption made by traditional projection methods is that the instances are i.i.d. However, this common i.i.d. assumption is unreasonable for relational data. By explicitly modeling covariance between instances as derived from the relational information, we propose a novel probabilistic projection method, called PRPCA, for relational data analysis. Although the i.i.d. assumption is no longer adopted in PRPCA, the learning algorithms for PRPCA can still be devised easily like those for PPCA which makes explicit use of the i.i.d. assumption. Experiments on real-world data sets show that PRPCA can effectively utilize relational information to dramatically outperform PCA.

- Sparse probabilistic relational projection (SPRP): The results learned by PRPCA lack interpretability because each principal component is a linear combination of all the original variables. We propose a sparse version of PRPCA, called SPRP, to learn a sparse projection matrix for relational dimensionality reduction. The sparsity in SPRP is achieved by imposing on the projection matrix a sparsity-inducing prior such as the Laplace prior or Jeffreys prior. An expectation-maximization (EM) (Dempster et al., 1977) algorithm is derived to learn the parameters of SPRP. Compared with PRPCA, the sparsity in SPRP not only makes the results more interpretable but also makes the projection operation much more efficient without compromising its accuracy. Furthermore, compared with traditional sparse projection methods based on the i.i.d. assumption, SPRP can learn a more discriminative projection by explicitly modeling the covariance between instances.

data analysis. Hence, for convenience of illustration, the specific term ‘content information’ is used to refer to the feature vectors describing the instances. However, the algorithms devised in this thesis can be applied to any relational data in which the instance feature vectors can represent any attribute information.

- Latent Wishart processes (LWP): Kernel methods (Schölkopf & Smola, 2002), such as support vector machine (SVM) (Schölkopf & Smola, 2002) and kernel PCA (KPCA) (Schölkopf et al., 1998), are widely used in many applications. One main concern towards kernel methods is on their sensitivity to the choice of kernel function or kernel matrix which characterizes the similarity between instances. Wishart processes (Zhang et al., 2006) have been successfully used to learn kernels for flat data. In relational data, the relational information often provides strong hints on the correlation (i.e., similarity) between instances. We propose a novel relational kernel learning model, called LWP, to learn the kernel function for relational data. This is done by seamlessly integrating the relational information and input attributes into the kernel learning process.

- Generalized latent factor model (GLFM): All the above models, including RRMF, PRPCA, SPRP and LWP, can only model data with undirected relationships. To model data with directed relationships using these models, we have to transform directed relationships into undirected ones with some preprocessing procedures. Recently, the *multiplicative latent factor model* (MLFM) (Hoff, 2009), has been proposed to model social networks with directed relationships. In social networks, *homophily* and *stochastic equivalence* (Wasserman & Katherine, 1994; Hoff, 2007) are two primary features of interest. Although MLFM can capture stochastic equivalence, it cannot model well homophily in networks. However, many real-world networks exhibit homophily or both homophily and stochastic equivalence, and hence the network structure of these networks cannot be modeled well by MLFM. We propose a novel model, called GLFM, for social network analysis by enhancing homophily modeling in MLFM. We devise a *minorization-maximization* (MM) algorithm (Lang et al., 2000) with linear-time complexity and convergence guarantee to learn the model parameters.

All the above RFMs (RRMF, PRPCA, SPRP, LWP, GLFM) are based on the methodology of latent factor modeling (Bartholomew & Knott, 1999; Memisevic, 2008), which is totally different from the methodology for existing mainstream SRL models like PLMs and PRMs. Hence, our RFMs provide an alternative way to model relational data. Compared with PLMs and PRMs, one promising advantage of our RFMs is that there is no need to perform time-consuming structure learning at all and the time complexity of most of them (except LWP) is linear to the number of observed links in the data [7]. This implies that our RFMs can be used to model large-scale data sets.

[7]In the complexity analysis of our RFMs, sometimes we say that the time complexity is linear to the number of instances rather than observed links. This is also true because typically the number of observed links is a constant multiple of the number of instances due to the sparsity of the networks.

The connections and differences between our RFMs are illustrated in Figure 1.2. More specifically, our RFMs can be categorized into two main classes according to whether they can model directed relationships or not. All the models in the class *undirected relation modeling* cannot be straightforwardly used to model directed relationships. We have to resort to preprocessing procedures to transform directed relationships into undirected ones if we want to use them for data with directed relationships. On the contrary, GLFM provides a way to directly model data with directed relationships. The models in the class *undirected relation modeling* can be further categorized according to two dimensions. One dimension is about whether the model is parametric or nonparametric, and the other dimension is about whether the model can be used to perform transductive or inductive inference. Note that typically most parametric models can be used for inductive inference. Hence, in this thesis, we do not design parametric models specially for transductive inference. The advantages and disadvantages of all the RFMs will be compared in Section 8 after we have introduced all the models, and some guidelines to choose between them for different learning problems will also be provided there. Therefore, the proposed RFMs in this thesis provide a toolbox for different learning settings in SRL.

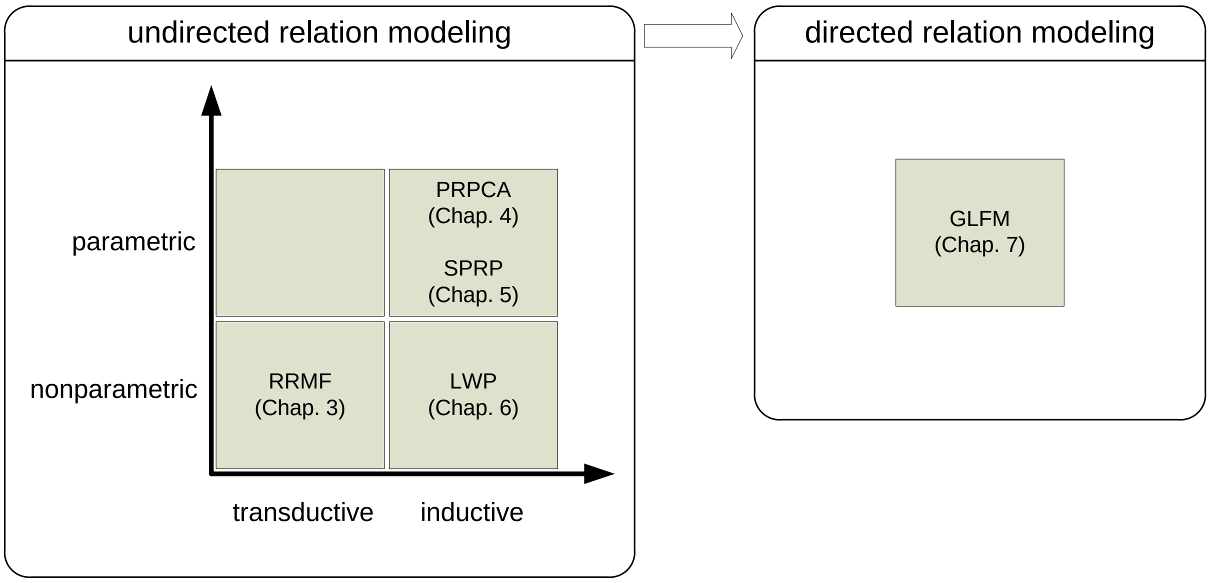


Figure 1.2: The connections and differences between our RFMs.

1.5 Thesis Outline

The rest of this thesis is organized as follows:

Chapter 2 introduces some background knowledge, including a brief literature review of SRL, traditional latent factor models (Bartholomew & Knott, 1999), and matrix variate distributions (Gupta & Nagar, 2000).

Chapter 3 presents the RRMF model, including its model formulation, parameter learning and experimental results on real applications.

Chapter 4 presents the PRPCA model, including its model formulation, parameter learning, and experimental performance on linked-document classification.

Chapter 5 introduces the sparse version of PRPCA, called SPRP, which can learn interpretable results for relational dimensionality reduction.

Chapter 6 introduces the LWP model, including its model formulation, parameter learning and experimental results.

Chapter 7 presents the GLFM model to enhance homophily modeling in MLFM. Application to social community detection is also demonstrated.

Chapter 8 concludes the whole thesis and proposes several potential directions for future pursuit.

CHAPTER 2

BACKGROUND

In this chapter, we briefly introduce some background knowledge related to the whole thesis, including a brief survey of SRL, traditional latent factor models (Bartholomew & Knott, 1999), and matrix variate distributions (Gupta & Nagar, 2000).

Besides, some common notations and abbreviations used in this thesis are summarized in Appendix A.

2.1 A Brief Survey of SRL

SRL tries to perform learning and inference in domains with complex relational structures (Getoor & Taskar, 2007). One important characteristic of relational data is *autocorrelation* (Neville, 2006), which refers to a statistical dependency between values of the same variable on related instances. To model the *autocorrelation*, most SRL methods adopt *collective inference* (Jensen et al., 2004) to infer various interrelated values simultaneously. Collective inference has been shown to achieve better results than inference for each variable independently (Chakrabarti et al., 1998; Neville & Jensen, 2000; Macskassy & Provost, 2007). Since most traditional statistical learning methods can only perform inference for each variable independently, SRL can achieve better performance than traditional methods if autocorrelation exists in the data. Hence, SRL has been successfully used in a large variety of applications with relational data. Here, we briefly introduce some representative SRL tasks, models and applications.

2.1.1 SRL Tasks

Some representative SRL tasks (Getoor & Diehl, 2005) include: collective classification (Macskassy & Provost, 2007; Sen et al., 2008), link prediction (Taskar et al., 2003; Hoff, 2007; Miller et al., 2009; Sutskever et al., 2009), link-based clustering (i.e., social community detection) (Wasserman & Katherine, 1994; Kemp et al., 2006), link-based ranking (i.e., social *centrality* estimation) (Wasserman & Katherine, 1994; Page et al., 1999), and collective entity resolution (Bhattacharya & Getoor, 2007).

In relational data, the class labels of linked instances are often correlated. *Collective classification* tries to jointly infer the class labels of a set of instances in a graph by exploiting the correlation between them. Some representative collective classification methods are introduced in (Macskassy & Provost, 2007; Sen et al., 2008).

Some links may not be observed in a graph. *Link prediction* (Taskar et al., 2003; Hoff, 2007; Miller et al., 2009; Sutskever et al., 2009) tries to predict those unobserved links based on some observed links or the attribute information of the instances. For example, the interactions between some proteins can be specified by chemical experiments. But it is unreasonable to test all pairs of proteins by chemical experiments due to the high cost of chemical experiments. Hence, we have to perform link prediction to find some potential interactions between the proteins. CF (Adomavicius & Tuzhilin, 2005) is another well-known link prediction problem where we have to predict the users' ratings on new items based on their rating history.

Community detection tries to group the nodes in a graph into several clusters so that the nodes in the same cluster have similar characteristics while those separated into different clusters have different characteristics. Community detection is a very hot topic in SNA (Wasserman & Katherine, 1994; Newman, 2006; Yang et al., 2009a).

Link-based ranking tries to give an order to the set of instances in a graph based on some criteria. The most well-known ranking algorithms are PageRank (Page et al., 1999) and HITS (Kleinberg, 1999). In SNA, there is also a ranking task which tries to rank instances according to their importance in the network. This importance is often called *centrality* (Wasserman & Katherine, 1994).

Entity resolution tries to determine which references in a data set refer to the same real-world entity. For example, in some data, "SRL" and "Statistical Relational Learning" appear simultaneously and entity resolution tries to determine whether they refer to the same thing or not. *Collective entity resolution* tries to perform entity resolution jointly for a set of entities by utilizing the links among them. Some recent development of collective entity resolution can be found in (Bhattacharya & Getoor, 2007).

Here, we only briefly introduce some representative SRL tasks. More detailed information about them can be found in (Getoor & Diehl, 2005).

2.1.2 SRL Models

Recently, many novel models, most of which try to integrate autocorrelation into the learning and inference process, have been proposed for SRL. These models can be mainly classified into four categories: individual inference models, heuristic collective inference (HCI) models, probabilistic relational models (PRMs), and probabilistic logic models (PLMs).

Individual inference models (Neville, 2006) transform relational data into flat data and then adopt conventional non-relational ML models to perform learning and inference. Hence, these models do not adopt collective inference techniques at all. Some representative individual inference models for relational data include relational Bayesian classifier (RBC) (Neville et al., 2003b) and relational probability trees (RPT) (Neville et al., 2003a). The RBC is a modification of the traditional naive Bayesian classifier for relational data, and the RPT is a modified version of the conventional decision tree algorithm for relational data. The individual inference models discard the autocorrelation information among the instances. The advantage of this kind of models is that they are very simple and many sophisticated traditional learning techniques can be directly adapted for relational data. However, naively discarding the autocorrelation information will deteriorate the prediction accuracy.

HCI models, such as those in (Neville & Jensen, 2000; Lu & Getoor, 2003; Macskassy & Provost, 2007), apply some iterative procedures to perform collective inference (Jensen et al., 2004) in an ad hoc way. A HCI method always contains three components (Macskassy & Provost, 2007): a *local model*, a *relational model*, and a *collective inferencing* component. A local model is a traditional method trained on local attribute information of the instances, which is always used for initialization. A relational model can employ both the relational structure and the attributes of the related instances for training. A collective inferencing component can use some collective inference procedures, such as Gibbs sampling (Geman & Geman, 1984) and relaxation labeling (Chakrabarti et al., 1998), to perform collective inference. Some representative HCI methods can be found in (Macskassy & Provost, 2007; Sen et al., 2008). Although HCI methods perform training and inference in an ad hoc way, many experimental results (Lu & Getoor, 2003; Jensen et al., 2004; Neville & Jensen, 2007) have shown that they can outperform individual inference models in most cases. However, the HCI methods are mainly designed for classification.

PRMs extend traditional graphical models to relational domains by eliminating their underlying i.i.d. assumption. Representative PRMs include *relational Bayesian networks* (RBNs) (Getoor et al., 2002), *relational Markov networks* (RMNs) (Taskar et al., 2002), and *relational dependency networks* (RDNs) (Neville & Jensen, 2007). RBNs, RMNs, RDNs are extensions of Bayesian networks, Markov networks, dependency networks (Heckerman et al., 2000), respectively. Learning a graphical model comprises two components: *structure learning* and *parameter learning*. Structure learning tries to specify the graph structure which indicates the dependencies between the variables (nodes). Given a graph structure, parameter learning tries to specify the parameters of the distributions for the nodes in the graph. RBNs have efficient parameter learning techniques, which will make structure learning relatively easier than other PRMs. However, the acyclicity constraint for RBNs excludes many important relational dependencies. Compared with RBNs, RMNs have two advantages: first, they can represent more complex relational dependencies without the acyclicity constrains; second, they are well suited for discriminative training. However, the learning process for RMNs is very complicated and has a very high computational cost. RDNs integrate some advantages from both RBNs and RMNs. Firstly, the relationships in RDNs are not constrained to be acyclic. Secondly, by using the pseudo-likelihood (Heckerman et al., 2000), the parameter learning process for RDNs is relatively efficient. However, the pseudo-likelihood technique makes the learned network not necessarily specify a consistent joint distribution. Hence, Gibbs sampling should be used to extract a unique joint distribution. Therefore, the performance of RDNs is not good when the number of labeled data is small. Overall, PRMs provide a principled way to model the relational data. However, the complicated models for relational autocorrelation make exact learning and inference intractable. Hence, only approximate learning and inference techniques are used by all existing PRMs.

PLMs extend traditional ILP models to support probabilistic reasoning in the first-order logic environment. Representative PLMs include *probabilistic logic programming* (PLP) (Ng & Subrahmanian, 1992), *probabilistic Horn abduction* (PHA) (Poole, 1993), *independent choice logic* (ICL) (Poole, 1997), probabilistic knowledge bases (PKBs) (Ngo & Haddawy, 1997), *Bayesian logic programs* (BLPs) (Kersting & Raedt, 2001), *stochastic logic programs* (SLPs) (Muggleton, 2000), the PRISM system (Sato & Kameya, 2001), and *Markov logic networks* (MLNs) (Richardson & Domingos, 2006). According to Chapter 9 in (Getoor & Taskar, 2007), only MLNs are *undirected* which can be seen as a combination of ILP and Markov networks, and all the other PLMs are *directed* which can be viewed as a combination of ILP and Bayesian networks.

In the four categories of SRL models discussed above, individual inference models can not effectively exploit the autocorrelation in the relational data, and HCI models are mainly used for collective classification. Hence, PRMs and PLMs are much more popular than the other two categories. For example, in (Getoor & Taskar, 2007), the authors mainly focus on PRMs and PLMs. As stated by Cussens James in Chapter 9 of (Getoor & Taskar, 2007), it is very hard (time-consuming) to perform structure learning for PRMs and PLMs. This main drawback of PRMs and PLMs has also been found by (Xu et al., 2006; Xu, 2007) and (Sutskever et al., 2009). Hence, it might be impractical to apply these models to large-scale relational data sets, which motivates the work in this thesis.

2.1.3 Application Areas

SRL has been widely used in a large variety of application areas, a portion of which are outlined as follows:

- *Web mining*: The hyper-links between the web pages provide very strong hint to improve classification, which has been adopted for web page classification (Chakrabarti et al., 1998; Lu & Getoor, 2003) and web spam detection (Castillo et al., 2007). Furthermore, the links (citations) between documents are used to help classify or cluster scientific research papers (Taskar et al., 2001; Nallapati et al., 2008).
- *Social network analysis*: In (Fawcett & Provost, 1997), fraud detection is completed by looking at indirect (two-hop) connections in the call network to known fraudulent accounts. SRL can also be used for social behavior modeling (Tang & Liu, 2009) and social community detection (Yang et al., 2009a, 2009b).
- *Marketing*: Network-based marketing techniques have been shown to achieve far better performance than traditional targeted marketing based on prior purchase information and demographics (Hill et al., 2006, 2007).
- *Bioinformatics*: Many entities, such as proteins, in molecules interact with each other. RMNs (Taskar et al., 2002) have been used to discover the molecular pathways (Segal et al., 2003a, 2003b). Protein-protein interaction (PPI) prediction is also a very interesting topic in bioinformatics which can be solved by SRL (Vert, 2009).

2.2 Latent Factor Models

Latent factor models (LFMs) (Teh et al., 2005; Memisevic, 2008; Hoff, 2009; Agarwal & Chen, 2009), which are also called latent variable models (LVMs) or factor analysis (Bartholomew & Knott, 1999), are statistical techniques widely used in many different disciplines, such as psychology, social science and economics. In an LFM, there are two kinds of variables. One is called *manifest* variables (or observed variables) which can be directly observed from data, and the other is called *latent* variables (or unobserved variables) which cannot be directly observed from data. For example, the *intelligence* cannot be directly observed from students. If we want to measure students' intelligence, we need to design a set of specific tests such as English test and Maths test. Then based on the observed *test scores*, we can infer the students' intelligence. Hence, in the LFM of students' intelligence, intelligence is a latent variable and the scores are manifest variables. Conventionally, *latent variables* are also called *factors*, and the modeling procedure with latent variables is called *latent variable modeling* or *latent factor modeling* (Bartholomew & Knott, 1999).

An LFM specifies the joint distribution for a set of manifest and latent variables. Let $\mathbf{t} = (t_1, t_2, \ldots, t_d)^T$ denote the d manifest variables of an instance, and $\mathbf{x} = (x_1, x_2, \ldots, x_q)^T$

denote the q latent variables corresponding to $\mathbf{t}$. Since LFMs are always used for dimensionality reduction, we assume that $q < d$. An LFM defines a joint distribution on $\mathbf{t}$ and $\mathbf{x}$ as follows:

$$p(\mathbf{t}, \mathbf{x}) = p(\mathbf{x})p(\mathbf{t} \,|\, \mathbf{x}), \tag{2.1}$$

where $p(\mathbf{x})$ is the prior distribution on $\mathbf{x}$, and $p(\mathbf{t} \,|\, \mathbf{x})$ is the conditional distribution of $\mathbf{t}$ given $\mathbf{x}$.

Any inference in LFMs must be based on the joint distribution $p(\mathbf{t}, \mathbf{x})$. For example, if we want to perform dimensionality reduction, we need to compute $p(\mathbf{x} \,|\, \mathbf{t})$ as follows:

$$p(\mathbf{x} \,|\, \mathbf{t}) = \frac{p(\mathbf{t}, \mathbf{x})}{p(\mathbf{t})} = \frac{p(\mathbf{x})p(\mathbf{t} \,|\, \mathbf{x})}{p(\mathbf{t})}, \tag{2.2}$$

where $p(\mathbf{t})$ is computed as follows:

$$p(\mathbf{t}) = \int p(\mathbf{t}, \mathbf{x}) d\mathbf{x} = \int p(\mathbf{x})p(\mathbf{t} \,|\, \mathbf{x}) d\mathbf{x}. \tag{2.3}$$

This dimensionality reduction procedure is adopted by probabilistic principal component analysis (PPCA) (Tipping & Bishop, 1999).

Both manifest and latent variables can be either *metrical* or *categorical* (Bartholomew & Knott, 1999). Metrical variables will take real numbers which can be discrete or continuous. Categorical variables assign each instance to be in one of a set of categories. The term *factor analysis* refers to a special case of LFMs in which both manifest and latent variables are metrical (Bartholomew & Knott, 1999).

Typically, both $p(\mathbf{x})$ and $p(\mathbf{t} \,|\, \mathbf{x})$ have some parameters to specify. These parameters are always learned from a set of training data $\{\mathbf{t}_n\}_{n=1}^N$ where $\mathbf{t}_n \in \mathbb{R}^d$. One representative learning method is maximum likelihood estimation (MLE), which is shown as follows:

$$\widehat{\Theta} = \underset{\Theta}{\operatorname{argmax}}\, p(\{\mathbf{t}_n\}_{n=1}^N \,|\, \Theta) = \underset{\Theta}{\operatorname{argmax}} \prod_{n=1}^{N} p(\mathbf{t}_n \,|\, \Theta), \tag{2.4}$$

where Θ denotes all the parameters in $p(\mathbf{x})$ and $p(\mathbf{t} \,|\, \mathbf{x})$, and $p(\mathbf{t}_n \,|\, \Theta)$ is computed according to (2.3). Note that, $p(\mathbf{t}_n \,|\, \Theta)$ is the same as $p(\mathbf{t}_n)$ except that $p(\mathbf{t}_n \,|\, \Theta)$ explicitly indicates its dependence on Θ.

As shown in (2.4), the assumption that $p(\{\mathbf{t}_n\}_{n=1}^N \,|\, \Theta) = \prod_{n=1}^N p(\mathbf{t}_n \,|\, \Theta)$ is a key characteristic of traditional LFMs, which explicitly assumes that the instances are i.i.d. From this assumption, we can see that each instance separately contributes to the objective function, such as the likelihood function. However, in relational data, there is correlation between the linked instances. To model the correlation (or covariance) between the instances, we need some advanced tools, such as the matrix variate distributions (Gupta & Nagar, 2000).

2.3 Matrix Variate Distributions

Traditional statistical learning methods separately model each instance which is represented as a vector of attributes. Hence, most of them are based on vector variate distributions. If we want to jointly model a set of instances, like those in relational data, we need to resort to matrix variate distributions (Gupta & Nagar, 2000). Here, we briefly review some basics about matrix variate distributions on which our RFMs are based. More detailed information can be found in (Gupta & Nagar, 2000).

Definition 2.1 (Gupta & Nagar, 2000) *The Kronecker product of two matrices* $\mathbf{P} \in \mathbb{R}^{m\times n}$ *and* $\mathbf{Q} \in \mathbb{R}^{p\times q}$*, denoted by* $\mathbf{P}\otimes\mathbf{Q}$*, is the* $mp \times nq$ *matrix defined by* $\mathbf{P}\otimes\mathbf{Q} = (P_{ij}\mathbf{Q})$.

Definition 2.2 (Gupta & Nagar, 2000) *For a matrix* $\mathbf{K} \in \mathbb{R}^{m\times n}$, $\text{vec}(\mathbf{K})$ *is the* $mn \times 1$ *vector defined as* $\text{vec}(\mathbf{K}) = (\mathbf{K}_{*1}^T, \mathbf{K}_{*2}^T, \ldots, \mathbf{K}_{*n}^T)^T$.

Definition 2.3 (Gupta & Nagar, 2000) *The random matrix* $\mathbf{K} \in \mathbb{R}^{m\times n}$ *is said to follow a* ***matrix variate normal distribution*** *with mean matrix* $\mathbf{E} \in \mathbb{R}^{m\times n}$ *and covariance matrix* $\boldsymbol{\Sigma}\otimes\boldsymbol{\Psi}$*, where* $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times m}, \boldsymbol{\Sigma} \succ 0$ *and* $\boldsymbol{\Psi} \in \mathbb{R}^{n\times n}, \boldsymbol{\Psi} \succ 0$*, if* $\text{vec}(\mathbf{K}^T) \sim \mathcal{N}_{mn}(\text{vec}(\mathbf{E}^T), \boldsymbol{\Sigma}\otimes\boldsymbol{\Psi})$*. This is denoted as* $\mathbf{K} \sim \mathcal{N}_{m,n}(\mathbf{E}, \boldsymbol{\Sigma}\otimes\boldsymbol{\Psi})$.

Theorem 2.1 (Gupta & Nagar, 2000) *If* $\mathbf{K} \sim \mathcal{N}_{m,n}(\mathbf{E}, \boldsymbol{\Sigma}\otimes\boldsymbol{\Psi})$*, then the p.d.f. of* $\mathbf{K}$ *is given by*

$$p(\mathbf{K}) = \frac{\text{etr}\left\{-\frac{1}{2}\boldsymbol{\Sigma}^{-1}(\mathbf{K}-\mathbf{E})\boldsymbol{\Psi}^{-1}(\mathbf{K}-\mathbf{E})^T\right\}}{(2\pi)^{mn/2}\,|\boldsymbol{\Sigma}\,|^{n/2}\,|\boldsymbol{\Psi}\,|^{m/2}}.$$

As said in (Gupta & Nagar, 2000), a matrix variate normal distribution arises when sampling from a multivariate normal population. For example, if $\mathbf{K}_{*1}, \ldots, \mathbf{K}_{*n}$ are *independently* sampled from $\mathcal{N}_m(\mathbf{m}, \boldsymbol{\Sigma})$, the observation random matrix $\mathbf{K}$ will follow the following distribution: $\mathbf{K} \sim \mathcal{N}_{m,n}(\mathbf{m}\mathbf{e}^T, \boldsymbol{\Sigma}\otimes\mathbf{I}_n)$. As for relational data, we can use a non-identity covariance matrix $\boldsymbol{\Psi}$ to model the correlation between the instances.

Theorem 2.2 (Gupta & Nagar, 2000) *Let* $\mathbf{K} \sim \mathcal{N}_{m,n}(\mathbf{0}, \boldsymbol{\Sigma}\otimes\boldsymbol{\Psi})$ *where* $\boldsymbol{\Sigma} = (\sigma_{ij})$ *and* $\boldsymbol{\Psi} = (\psi_{ij})$*, then* $\langle K_{i_1j_1}K_{i_2j_2}\rangle = \sigma_{i_1i_2}\psi_{j_1j_2}$.

Theorem 2.3 (Gupta & Nagar, 2000) *Let* $\mathbf{K} \sim \mathcal{N}_{m,n}(\mathbf{E}, \boldsymbol{\Sigma}\otimes\boldsymbol{\Psi})$ *and* $\mathbf{F}$ *is of size* $n \times n$*, then* $\langle \mathbf{K}\mathbf{F}\mathbf{K}^T\rangle = \text{tr}(\mathbf{F}^T\boldsymbol{\Psi})\boldsymbol{\Sigma} + \mathbf{E}\mathbf{F}\mathbf{E}^T$.

Definition 2.4 (Gupta & Nagar, 2000) *An $n \times n$ random symmetric positive definite matrix $\mathbf{K}$ is said to have a **Wishart distribution** with parameters $n, q,$ and $n \times n$ scale matrix $\mathbf{\Sigma} \succ 0$, written as $\mathbf{K} \sim \mathcal{W}_n(q, \mathbf{\Sigma})$, if its p.d.f. is given by*

$$\frac{|\mathbf{K}|^{(q-n-1)/2}}{2^{qn/2}\Gamma_n(q/2)|\mathbf{\Sigma}|^{q/2}} \exp\Big(-\frac{1}{2}\mathrm{tr}(\mathbf{\Sigma}^{-1}\mathbf{K})\Big), \; q \geq n. \tag{2.5}$$

Here $\mathbf{\Sigma} \succ 0$ means that $\mathbf{\Sigma}$ is positive definite (pd).

Other matrix variate distributions, such as matrix variate t-distributions, matrix variate Beta distributions and matrix variate Dirichlet distributions, and their properties can be found in (Gupta & Nagar, 2000).

CHAPTER 3

RELATION REGULARIZED MATRIX FACTORIZATION

3.1 Introduction

Matrix factorization (MF) methods (Singh & Gordon, 2008b), which try to learn a lower-dimensional latent representation for instances, have been widely used for various data analysis applications. One of the most popular MF methods is latent semantic indexing (LSI) (Deerwester et al., 1990), which uses SVD to map the *content* of documents into a lower-dimensional latent semantic space. The objective is to retain as much information of the documents as possible while simultaneously removing the noise. Subsequent analysis, such as clustering or classification, can be performed based on this latent space representation. Maximum margin matrix factorization (MMMF) (Srebro et al., 2004; Rennie & Srebro, 2005), which can be seen as a regularized version of SVD by controlling the complexity of the factors,[1] has been successfully used for many applications, such as CF (Rennie & Srebro, 2005). Many other variants of MF methods and their applications can be found in (Singh & Gordon, 2008b).

Although MF methods have achieved very promising performance in many applications, most of them are designed to handle only one matrix at a time. The matrix can be either the feature representation (*content information*) of a set of instances, or the link structure (*relational information*) among a set of instances. For example, LSI was originally proposed for content-based information analysis by performing SVD on the bag-of-words representation of a set of documents, and MMMF has been successfully applied to collaborative filtering which employs only the relationships among a set of entities (Rennie & Srebro, 2005).

However, some relational data, such as web pages and research papers, contain both textual content information and relational structure.[2] These two kinds of information often complement each other because they are typically collected at different semantic

[1]MMMF can make use of different loss functions. The original MMMF (Srebro et al., 2004) only used the hinge loss, but many subsequent works also used other loss functions such as smooth hinge (Rennie & Srebro, 2005) and squared loss. Here we refer in particular to the squared loss function.

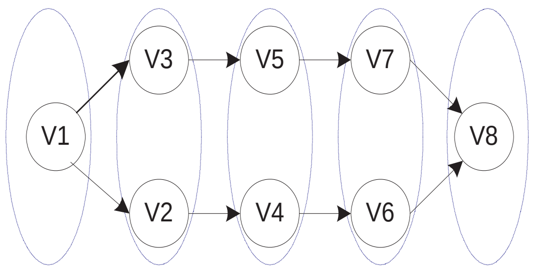
[2]As stated in Section 1.1, a relational data set can be represented as a set of matrices. Here, we use linked-document classification as a running example for relational data analysis. Hence, there are two matrices in our data set, one being the content (attribute) matrix and the other being the adjacency (link) matrix.

levels. For example, a citation/reference relationship between two papers provides a very strong evidence for them to belong to the same topic, although sometimes they bear low similarity in their content due to the sparse nature of the bag-of-words representation. Similarly, the content information can also provide additional insights about the relationships among instances. Hence, naively discarding any one kind of information would not allow us to fully utilize all the relevant information available. Ideally, we should strive for integrating both kinds of information seamlessly into a common framework for data analysis.

In (Zhu et al., 2007), a joint link-content MF method, abbreviated as LCMF here, was proposed to seamlessly integrate content and relational information into an MF framework through a set of shared factors for the factorization of both content and link structures. The authors of LCMF argue that their method is more reasonable than some other simple methods for combining content and link information, such as that in (Kurland & Lee, 2005), which seek to convert one type of information into the other. Experimental results in (Zhu et al., 2007) show that LCMF can outperform other state-of-the-art methods.

However, as said in Section 1.1.2, the links in different applications might capture different semantics. As a result, the model assumption adopted by LCMF might not be satisfied for some applications, which would cause LCMF to fail in such cases. Let us take a synthetic link structure from (Zhu et al., 2007) as an example to illustrate this point. This example is shown in Figure 3.1(a), in which there are eight nodes, corresponding to eight entities, and eight directed links. After performing link MF based on Zhu *et al.*'s method, the entities will be automatically grouped into five clusters, each corresponding to one ellipse in Figure 3.1(a). The learned latent factors of the entities (Zhu et al., 2007) are shown in Figure 3.1(b), in which the latent factors for V1 to V8 are listed from the first row to the last row in order. We can see that the entities in each cluster, say V2 and V3, have the same latent factor representation. From this example, it is not difficult to reveal the model assumption behind LCMF: if two entities link to or are linked by one common entity, the two entities will have similar latent factor representations. That is to say, LCMF assumes that the links between the entities exhibit *stochastic equivalence*. This is a very promising property if the application scenario indeed satisfies this assumption. One such application is the web page classification task on WebKB (Craven et al., 1998). For example, the homepages of faculties are always linked by or link to the homepage of the department, but the homepages of two faculties seldom link to each other. The advantage of this property has been verified by the promising accuracy of LCMF on the WebKB data set (Zhu et al., 2007).

However, one problem with LCMF is that the factor representations of two linked entities are not guaranteed to be similar, i.e., LCMF cannot effectively model links ex-



(a) Synthetic link structure

$$\begin{matrix}
-.8 & -.5 & .3 & -.1 & -.0 \\
-.0 & .4 & .6 & -.1 & -.4 \\
-.0 & .4 & .6 & -.1 & -.4 \\
.3 & -.2 & .3 & -.4 & .3 \\
.3 & -.2 & .3 & -.4 & .3 \\
-.4 & .5 & .0 & -.2 & .6 \\
-.4 & .5 & .0 & -.2 & .6 \\
-.1 & .1 & -.4 & -.8 & -.4
\end{matrix}$$

(b) Result of link MF

Figure 3.1: A synthetic example of link structure from [Zhu *et al.*, 2007] for illustration.

hibiting *homophily*. For example, the latent factor representation of V8 is relatively far away from that of V6 or V7 after LCMF learning. This property is undesirable for many other applications with homophily links. For example, to classify research papers according to their topic, the citation (link) information is usually very important. More specifically, two papers with a citation relationship between them are most likely about the same topic, and consequently, the learned latent factors of two linked papers should be similar. Unfortunately, this cannot be handled well by LCMF, which is also verified by the relatively poor performance of LCMF on the Cora data set (McCallum et al., 2000) for paper classification.

From the analysis above, we can see that there exist at least two types of links with different semantics: links exhibiting stochastic equivalence (*SE links*), such as those in the WebKB data set; links exhibiting homophily (*homophily links*), such as those in the Cora data set. As discussed above, LCMF can handle well applications with SE links but not those with homophily links. It is this limitation that has motivated us to pursue research reported in this chapter.

We propose in this chapter a novel MF method. The basic idea is to make the latent factors of two entities as close as possible if there exists a link between them. More specifically, we utilize relational information to regularize the content MF procedure, resulting in a principled framework which seamlessly integrates content and relational information. We refer to our method as *relation regularized matrix factorization*, which will be abbreviated as RRMF in the sequel for convenience. To learn the parameters of RRMF, we propose a linear-time algorithm, which makes RRMF suitable for large-scale problems. Furthermore, the learning procedure of RRMF can be proved to be convergent. Experimental results on a data set with homophily links demonstrate that RRMF can dramatically outperform LCMF and other state-of-the-art methods.

Although RRMF is motivated to model homophily links, it can also be used for applications with SE links by adding a simple step to preprocess the link structure of the data. This will be discussed in detail in the following sections.

3.2 Relation Regularized Matrix Factorization

We use a $d \times N$ matrix $\mathbf{T}$ to denote the content matrix, with N being the number of entities and d the number of features. The $N \times N$ matrix $\mathbf{A}$ denotes the adjacency (link) matrix of the N instances. In RRMF, we assume that the links are undirected. For those data with directed links, we will convert the directed links into undirected links which can keep the original physical meaning of the links. This will be described in detail in Section 3.3. Hence, $A_{ij} = 1$ if there exists a link between instances i and j, and otherwise $A_{ij} = 0$. Moreover, we always assume that there exist no self-links, i.e., $A_{ii} = 0$. q denotes the number of latent factors for each instance.

3.2.1 Model Formulation

Like in LSI (Deerwester et al., 1990), we adopt a similar MF method to approximate the content matrix:

$$\min_{\mathbf{U},\mathbf{V}} \frac{1}{2}\|\mathbf{T} - \mathbf{V}\mathbf{U}^T\|^2 + \frac{\alpha}{2}(\|\mathbf{U}\|^2 + \|\mathbf{V}\|^2), \tag{3.1}$$

where we use an $N \times q$ matrix $\mathbf{U}$ and a $d \times q$ matrix $\mathbf{V}$ to denote the latent q-dimensional representations of all the documents (or entities in a more general case) and words (or features in a more general case), respectively, with $\mathbf{U}_{i*}$ for document i and $\mathbf{V}_{j*}$ for word j. α is a hyperparameter.

To integrate the relational information into the MF procedure, we use the relationships among entities to regularize the latent factors. The basic idea of our method is to make the latent representations of two entities as close as possible if there exists a link between them. We can achieve this goal by minimizing the following objective function:

$$\begin{aligned} l &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} A_{ij}\|\mathbf{U}_{i*} - \mathbf{U}_{j*}\|^2 \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left[A_{ij}\sum_{k=1}^{q}(U_{ik} - U_{jk})^2\right] \\ &= \frac{1}{2}\sum_{k=1}^{q}\left[\sum_{i=1}^{N}\sum_{j=1}^{N} A_{ij}(U_{ik} - U_{jk})^2\right] \\ &= \sum_{k=1}^{q}\mathbf{U}_{*k}^T\boldsymbol{\mathcal{L}}\mathbf{U}_{*k} = \mathrm{tr}(\mathbf{U}^T\boldsymbol{\mathcal{L}}\mathbf{U}), \end{aligned} \tag{3.2}$$

where $\boldsymbol{\mathcal{L}} = \mathbf{D} - \mathbf{A}$ is known as the Laplacian matrix (Chung, 1997) with $\mathbf{D}$ being a diagonal matrix whose diagonal elements $D_{ii} = \sum_j A_{ij}$, and $\mathrm{tr}(\cdot)$ denotes the trace of a matrix.

Combining (3.1) and (3.2), we obtain the following objective function which we seek to minimize during learning:

$$\begin{aligned} f &= \frac{1}{2}\|\mathbf{T} - \mathbf{V}\mathbf{U}^T\|^2 + \frac{\alpha}{2}(\|\mathbf{U}\|^2 + \|\mathbf{V}\|^2) + \frac{\beta}{2}\mathrm{tr}(\mathbf{U}^T\boldsymbol{\mathcal{L}}\mathbf{U}) \\ &= \frac{1}{2}\|\mathbf{T} - \mathbf{V}\mathbf{U}^T\|^2 + \frac{1}{2}\mathrm{tr}[\mathbf{U}^T(\alpha\mathbf{I}_N + \beta\boldsymbol{\mathcal{L}})\mathbf{U}] + \frac{\alpha}{2}\mathrm{tr}(\mathbf{V}\mathbf{V}^T), \end{aligned} \tag{3.3}$$

which seamlessly integrates the content information and the relational information into a principled framework. Here, β is a hyperparameter controlling the contribution of the relational information. Because we use relational information to regularize the content MF procedure, we call the model in (3.3) <u>r</u>elation <u>r</u>egularized <u>m</u>atrix <u>f</u>actorization, abbreviated as RRMF.

Remark 3.1 *The normalized Laplacian matrix (Chung, 1997), given by* $\tilde{\boldsymbol{\mathcal{L}}} = \mathbf{D}^{-1/2}\boldsymbol{\mathcal{L}}\mathbf{D}^{-1/2} = \mathbf{I}_N - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$*, can be used to substitute* $\boldsymbol{\mathcal{L}}$ *in (3.2) and (3.3) for regularization. If* $\tilde{\boldsymbol{\mathcal{L}}}$ *is adopted, the objective function in (3.2) will be* $\tilde{l} = \frac{1}{2}\sum_{k=1}^{q}\left[\sum_{i=1}^{N}\sum_{j=1}^{N}A_{ij}\left(\frac{U_{ik}}{\sqrt{D_{ii}}} - \frac{U_{jk}}{\sqrt{D_{jj}}}\right)^2\right]$. *The choice between* $\boldsymbol{\mathcal{L}}$ *and* $\tilde{\boldsymbol{\mathcal{L}}}$ *depends on applications. In RRMF, the lemmas and theorems are derived based on* $\boldsymbol{\mathcal{L}}$*, but they also hold for* $\tilde{\boldsymbol{\mathcal{L}}}$ *with slight changes in the derivations.*

From (3.3), it is easy to see that the model assumption behind RRMF is in line with the semantics of homophily links. Hence, RRMF can be expected to achieve good performance for applications with homophily links, such as research paper classification, which will be verified by experiment in Section 3.3.

On the other hand, RRMF can also be used for applications with SE links. All we need to do is just to preprocess the link structure in the data with a very simple strategy, but the model and learning algorithms need not be changed. According to the semantics of SE links, two entities linking to or linked by one common entity will likely be from the same class. One simple way to preprocess the link structure is to artificially add a link between two entities if they link to or are linked by a common entity. The added links will then satisfy the semantics of homophily links. For example, in Figure 3.1(a), after preprocessing, V2 and V3 will be connected and V6 and V7 will also be connected. Learning RRMF based on these added links is now in line with the underlying semantics. From our experiments on the WebKB data set, we find that RRMF can still achieve performance comparable with LCMF just by adopting the simple link preprocessing strategy as described above.

3.2.2 Probabilistic Reformulation

We can associate RRMF with a probabilistic formulation, which is shown as follows:

$$\begin{aligned} p(\mathbf{U}) &= \mathcal{N}_{N,q}\Big(\mathbf{0}, (\mathbf{I}_N + \frac{\beta}{\alpha}\boldsymbol{\mathcal{L}})^{-1} \otimes \mathbf{I}_q\Big), \\ p(\mathbf{V}) &= \mathcal{N}_{d,q}(\mathbf{0}, \mathbf{I}_d \otimes \mathbf{I}_q), \\ p(\mathbf{T} \,|\, \mathbf{U}, \mathbf{V}) &= \mathcal{N}_{d,N}(\mathbf{V}\mathbf{U}^T, \alpha\mathbf{I}_d \otimes \mathbf{I}_N). \end{aligned}$$

More specifically, we use the relational information incorporated in $\boldsymbol{\mathcal{L}}$ to define the prior distribution for $\mathbf{U}$, and use the content information ($\mathbf{T}$) to define the likelihood given the latent factors $\mathbf{U}$ and $\mathbf{V}$. We can see that the parameters to minimize (3.3) are just a maximum a posteriori (MAP) estimation of the parameters for the above probabilistic formulation. From this probabilistic reformulation, it is easy to see that the learned latent factors of the N entities are correlated with covariance matrix $(\mathbf{I}_N + \frac{\beta}{\alpha}\boldsymbol{\mathcal{L}})^{-1}$ rather than i.i.d.

3.2.3 Learning

In this subsection, we first prove that the objective function in (3.3) is convex with respect to (w.r.t.) any one of its parameters, $\mathbf{U}$ and $\mathbf{V}$. Then, we propose an alternating projection algorithm to learn the parameters in linear time and show that it is guaranteed to converge to a local optimum.

Convexity of the Objective Function

Lemma 3.1 *The Laplacian matrix $\boldsymbol{\mathcal{L}}$ is psd, i.e., $\boldsymbol{\mathcal{L}} \succeq 0$.*

Proof: For any $N \times 1$ vector $\mathbf{x} \neq 0$,

$$\mathbf{x}^T\boldsymbol{\mathcal{L}}\mathbf{x} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} A_{ij}(x_i - x_j)^2 \geq 0.$$

■

Lemma 3.2 $\sum_{i=1}^{d} \mathbf{V}_{i*}^T\mathbf{V}_{i*} \succeq 0$.

Proof: For any $q \times 1$ vector $\mathbf{z} \neq 0$,

$$\begin{aligned} \mathbf{z}^T\left(\sum_{i=1}^{d}\mathbf{V}_{i*}^T\mathbf{V}_{i*}\right)\mathbf{z} &= \sum_{i=1}^{d}\mathbf{z}^T\mathbf{V}_{i*}^T\mathbf{V}_{i*}\mathbf{z} \\ &= \sum_{i=1}^{d}(\mathbf{V}_{i*}\mathbf{z})^2 \geq 0. \end{aligned}$$

■

Theorem 3.1 *f is convex w.r.t.* $\mathbf{U}$.

Proof: We first rewrite f as follows:

$$f = \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{N}(T_{ij} - \mathbf{V}_{i*}\mathbf{U}_{j*}^{T})^2 + \frac{1}{2}\sum_{k=1}^{q}\mathbf{U}_{*k}^{T}(\alpha\mathbf{I}_N + \beta\boldsymbol{\mathcal{L}})\mathbf{U}_{*k} + C_1,$$

where C_1 is a constant independent of $\mathbf{U}$.

Let

$$\begin{aligned} g &= \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{N}(T_{ij} - \mathbf{V}_{i*}\mathbf{U}_{j*}^{T})^2 \\ &= C_2 + \frac{1}{2}\sum_{j=1}^{N}[\mathbf{U}_{j*}(\sum_{i=1}^{d}\mathbf{V}_{i*}^{T}\mathbf{V}_{i*})\mathbf{U}_{j*}^{T} - 2(\sum_{i=1}^{d}T_{ij}\mathbf{V}_{i*})\mathbf{U}_{j*}^{T}], \end{aligned}$$

where C_2 is a constant independent of $\mathbf{U}$. It is easy to see that

$$\begin{aligned} \mathbf{G}_j \triangleq \frac{\partial^2 g}{\partial\mathbf{U}_{j*}^{T}\partial\mathbf{U}_{j*}} &= \sum_{i=1}^{d}\mathbf{V}_{i*}^{T}\mathbf{V}_{i*}, \\ \frac{\partial^2 g}{\partial\mathbf{U}_{j*}^{T}\partial\mathbf{U}_{k*}} &= \mathbf{0}, \quad \text{if} \quad j \neq k. \end{aligned}$$

If we use $\mathbf{u} = (\mathbf{U}_{1*}, \mathbf{U}_{2*}, \ldots, \mathbf{U}_{N*})^T$ to denote the vector representation of $\mathbf{U}$, the second-order derivative (Hessian) of g w.r.t. $\mathbf{u}$ will be a block-diagonal matrix:

$$\mathbf{G}_{\mathbf{u}} \triangleq \frac{\partial^2 g}{\partial\mathbf{u}\partial\mathbf{u}^T} = \mathrm{diag}[\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_N].$$

According to Lemma 3.2, $|\mathbf{G}_j| \geq 0$, where $|\cdot|$ denotes the determinant of a matrix. Because $|\mathbf{G}_{\mathbf{u}}| = \prod_{j=1}^{N}|\mathbf{G}_j| \geq 0$, we can conclude that g is convex w.r.t. $\mathbf{U}$.

Let

$$h = \frac{1}{2}\sum_{k=1}^{q}\mathbf{U}_{*k}^{T}(\alpha\mathbf{I}_N + \beta\boldsymbol{\mathcal{L}})\mathbf{U}_{*k}.$$

If we use $\mathbf{a} = (\mathbf{U}_{*1}^{T}, \mathbf{U}_{*2}^{T}, \ldots, \mathbf{U}_{*q}^{T})^T$ to denote another vector representation of $\mathbf{U}$, the Hessian of h w.r.t. $\mathbf{a}$ will also be a block-diagonal matrix:

$$\mathbf{H}_{\mathbf{a}} \triangleq \frac{\partial^2 h}{\partial\mathbf{a}\partial\mathbf{a}^T} = \mathrm{diag}[\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_q],$$

where $\mathbf{H}_k$ is defined as follows:

$$\mathbf{H}_k \triangleq \frac{\partial^2 h}{\partial \mathbf{U}_{*k} \partial \mathbf{U}_{*k}^T} = \alpha \mathbf{I}_N + \beta \boldsymbol{\mathcal{L}}.$$

According to Lemma 3.1, we can conclude that $|\mathbf{H_a}| > 0$, and hence h is convex w.r.t. $\mathbf{U}$.

Hence, $f = g + h + C_1$ is convex w.r.t. $\mathbf{U}$. ■

Theorem 3.2 *f is convex w.r.t.* $\mathbf{V}$.

Proof: We first rewrite f as follows:

$$f = C_3 + \frac{1}{2}\sum_{i=1}^{d}[\mathbf{V}_{i*}\left(\sum_{j=1}^{N}\mathbf{U}_{j*}^T\mathbf{U}_{j*} + \alpha\mathbf{I}_q\right)\mathbf{V}_{i*}^T - 2\left(\sum_{j=1}^{N}T_{ij}\mathbf{U}_{j*}^T\right)\mathbf{V}_{i*}],$$

where C_3 is a constant independent of $\mathbf{V}$. If we use $\mathbf{v} = (\mathbf{V}_{1*}, \mathbf{V}_{2*}, \ldots, \mathbf{V}_{d*})^T$ to denote the vector representation of $\mathbf{V}$, the Hessian of f w.r.t. $\mathbf{v}$ will be a block-diagonal matrix:

$$\mathbf{K}_\mathbf{v} \triangleq \frac{\partial^2 f}{\partial \mathbf{v} \partial \mathbf{v}^T} = \mathrm{diag}[\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_d],$$

where $\mathbf{K}_i = \sum_{j=1}^{N}\mathbf{U}_{j*}^T\mathbf{U}_{j*} + \alpha\mathbf{I}_q$. It is easy to verify that $\mathbf{K}_i \succ 0$ and $\mathbf{K}_\mathbf{v} \succ 0$. Hence, f is convex w.r.t. $\mathbf{V}$. ■

Learning Strategy

We adopt an alternating projection method to learn the parameters $\mathbf{U}$ and $\mathbf{V}$. More specifically, each time we fix one parameter and then update the other one. This procedure will be repeated for several iterations until some termination condition is satisfied. We will prove that the learning algorithm is convergent.

Learning U According to Theorem 3.1, one straightforward way to learn $\mathbf{U}$ is to set the gradient of f w.r.t. $\mathbf{U}$ to 0 and solve the corresponding linear system. However, this is computationally demanding if N is large because we have to invert a Hessian matrix of size as large as $Nq \times Nq$. In RRMF, we adopt an alternative strategy to perform optimization on $\mathbf{U}$, which is to optimize one column $\mathbf{U}_{*k}$ at a time with the other columns fixed. Because f is convex w.r.t. $\mathbf{U}$, this alternative strategy will be guaranteed to converge to the optimal solution.

Because f is convex w.r.t. $\mathbf{U}$, f is also convex w.r.t. $\mathbf{U}_{*k}$ with all other variables fixed. Hence, computing the gradient of f w.r.t. $\mathbf{U}_{*k}$ and setting it to 0, we can optimize $\mathbf{U}_{*k}$ by solving the following linear system:

$$\mathbf{F}^{(k)}\mathbf{U}_{*k} = \mathbf{e}^{(k)}, \tag{3.4}$$

where

$$\begin{aligned}
\mathbf{F}^{(k)} &= \mathbf{E}^{(k)} + \alpha \mathbf{I}_N + \beta \boldsymbol{\mathcal{L}}, \\
\mathbf{E}^{(k)} &= \mathrm{diag}(\frac{\partial^2 g}{\partial U_{1k} \partial U_{1k}}, \frac{\partial^2 g}{\partial U_{2k} \partial U_{2k}}, \ldots, \frac{\partial^2 g}{\partial U_{Nk} \partial U_{Nk}}),
\end{aligned}$$

with

$$\begin{aligned}
\frac{\partial^2 g}{\partial U_{jk} \partial U_{jk}} &= \sum_{i=1}^{d} V_{ik}^2, \\
\mathbf{e}^{(k)} &= (e_1^{(k)}, e_2^{(k)}, \ldots, e_N^{(k)})^T, \\
e_j^{(k)} &= \sum_{i=1}^{d} V_{ik}(T_{ij} - \mathbf{V}_{i*}\mathbf{U}_{j*}^T + V_{ik}U_{jk}).
\end{aligned}$$

One direct way to solve the linear system in (3.4) is to set $\mathbf{U}_{*k} = [\mathbf{F}^{(k)}]^{-1}\mathbf{e}^{(k)}$. However, the computation cost is $O(N^3)$, which is computationally prohibitive for general text classification applications because N is always very large. Here, we propose to use the *steepest descent* method (Shewchuk, 1994) to iteratively update $\mathbf{U}_{*k}$:

$$\begin{aligned}
r(t) &= \mathbf{e}^{(k)} - \mathbf{F}^{(k)}\mathbf{U}_{*k}(t), \\
\delta(t) &= \frac{r(t)^T r(t)}{r(t)^T \mathbf{F}^{(k)} r(t)}, \\
\mathbf{U}_{*k}(t+1) &= \mathbf{U}_{*k}(t) + \delta(t) r(t).
\end{aligned}$$

Steepest descent will guarantee the algorithm to converge to the global minimum with the objective function value decreased in each iteration. Suppose the number of nonzero elements in $\boldsymbol{\mathcal{L}}$ is M. We can see that the computation cost in each iteration is $O(N+M)$. If the number of iterations is K, the time complexity will be $O(K(N+M))$. From our experiments, good performance can be achieved with a small value of K, such as $K = 10$ in our following experiments. Furthermore, M is typically a constant multiple of N. Hence, the overall complexity is roughly $O(N)$, which is dramatically less than $O(N^3)$.

Learning V One very nice property of $\mathbf{V}$ is that the Hessian $\mathbf{K}_\mathbf{V}$ is block-diagonal, where each nonzero block corresponds to a row of $\mathbf{V}$. Since the inverse of a block-diagonal matrix can be expressed as the inverse of each block, the update of the whole matrix $\mathbf{V}$ can naturally be decomposed into the update of each row $\mathbf{V}_{i*}$.

Unlike the learning of $\mathbf{U}$, all the Hessian matrices for different rows of $\mathbf{V}$ are equal, i.e., $\mathbf{K}_i = \mathbf{K}_k = \mathbf{K} = \sum_{j=1}^{N} \mathbf{U}_{j*}^T \mathbf{U}_{j*} + \alpha \mathbf{I}_q$. Furthermore, $\mathbf{K}$ is a small matrix of size $q \times q$, where q is typically a small number and is less than 50 in our experiments. Hence, we can directly update $\mathbf{V}_{i*}$ as follows:

$$\mathbf{V}_{i*} = \left(\sum_{j=1}^{N} T_{ij} \mathbf{U}_{j*} \right) \mathbf{K}^{-1}.$$

3.2.4 Convergence and Complexity Analysis

Theorem 3.3 *The learning algorithm will converge.*

Proof: In each iteration, the learning algorithm ensures that the objective function value in (3.3) always decreases. Furthermore, the objective function is bounded below by 0. Hence, the learning algorithm will converge. Because f is not jointly convex w.r.t. $\mathbf{U}$ and $\mathbf{V}$, the solution is a local optimum. ∎

We have seen that in each iteration, the time required for updating $\mathbf{U}$ is $O(N)$, and it is easy to see that the time complexity for updating $\mathbf{V}$ is also $O(N)$. In our experiments, typically the algorithm can converge in less than 50 iterations. In fact, we find that 5 iterations are sufficient to achieve good performance. Hence, the overall time complexity of the learning algorithm is $O(N)$.

3.3 Experiments

3.3.1 Data Sets and Evaluation Scheme

We use the same data sets, WebKB (Craven et al., 1998) and Cora (McCallum et al., 2000), and the same bag-of-words representation with the same original link structure as those in (Zhu et al., 2007) to evaluate our method. The WebKB data set contains about 6,000 web pages collected from the web sites of computer science departments of four universities (Cornell, Texas, Washington, and Wisconsin). Each web page is labeled with one out of seven categories: student, professor, course, project, staff, department, and "other". It should be noted that to train our RRMF method we adopt the simple preprocessing strategy for the link structure in this data set. That is, if two web pages are co-linked by another common web page, we add a link between these two pages. After preprocessing, all the directed links are converted into undirected links. The characteristics about the WebKB data set are briefly summarized in Table 3.1.

The Cora data set contains the abstracts and references of about 34,000 research papers from the computer science community. For fair comparison, we adopt the same subset of the data as that in (Zhu et al., 2007) to test our method. The task is to classify

Table 3.1: Characteristics of the WebKB data set.

	#classes	#entities	#terms
Cornell	7	827	4,134
Texas	7	814	4,029
Washington	7	1,166	4,165
Wisconsin	6	1,210	4,189

each paper into one of the subfields of data structure (DS), hardware and architecture (HA), machine learning (ML), and programming language (PL). The characteristics of the Cora data set are summarized in Table 3.2.

Table 3.2: Characteristics of the Cora data set.

	#classes	#entities	#terms
DS	9	751	6,234
HA	7	400	3,989
ML	7	1,617	8,329
PL	9	1,575	7,949

As in (Zhu et al., 2007), we adopt 5-fold cross validation to evaluate our method. More specifically, we randomly split the data into five folds (subsets), and then repeat the test five times, in each one of which we use one fold for testing and the other four folds for training. As in (Zhu et al., 2007), the number of factors q is set to 50 for RRMF. The α in (3.3) is fixed to 1, and β is specified by cross-validation on the training data. After obtaining the factors, a linear SVM[3] is trained for classification based on the low-dimensional representation, which is the same as the test procedure for the methods in (Zhu et al., 2007). The average classification accuracies and the standard deviations over the five repeats are adopted as the performance metric.

3.3.2 Baselines

The methods adopted for our comparative study belong to two classes. The first class contains the baselines used in (Zhu et al., 2007):

- SVM on content: This method ignores the link structure in the data, and applies SVM only on the content information in the original bag-of-words representation.

[3]We use LIBSVM (Chang & Lin, 2001) to train all the SVM classifiers.

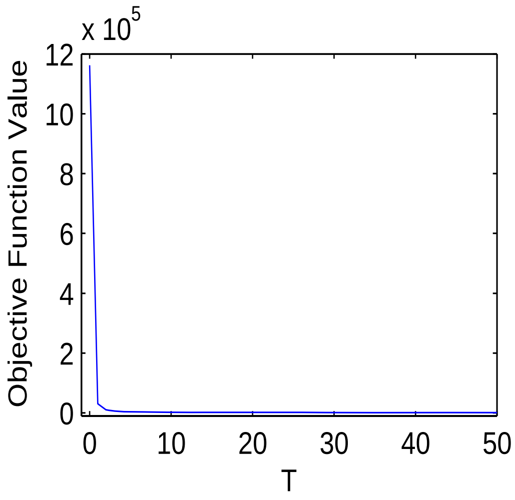
- SVM on links: This method ignores the content information, and treats links as the features, i.e, the ith feature is *link-to-page*$_i$.
- SVM on link-content: The content features and link features of the two methods above are combined to give the feature representation.
- Directed graph regularization: This is the method introduced in (Zhou et al., 2005), which is only based on the link structure.
- PLSI+PHITS: This method, described in (Cohn & Hofmann, 2000), combines text content information and link structure for analysis.

The second class contains some methods that are most related to RRMF:

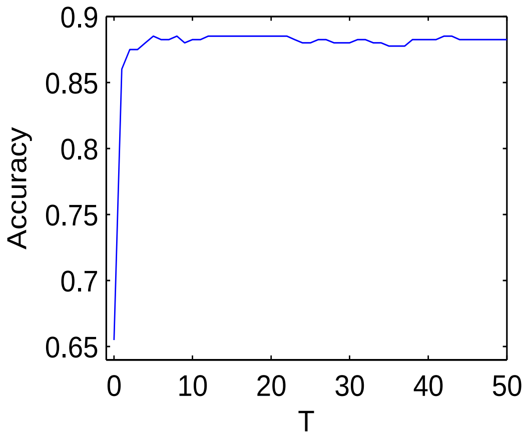
- PCA: This method first applies PCA on the content information to get a low-dimensional representation, based on which SVM is trained for classification. Because we use PCA to initialize $\mathbf{U}$ and $\mathbf{V}$ in RRMF, this method serves to demonstrate whether the good performance of RRMF comes from a good initialization value or from the learning procedure of RRMF.
- MMMF: This method applies MF only on the content information, which is a special case of (3.3) by setting $\beta = 0$. It is used to show that the relational information does help a lot in relational data analysis.
- Link-content MF: This is the joint link-content MF method in (Zhu et al., 2007).
- Link-content sup. MF: This is the supervised counterpart of link-content MF by using the data labels to guide the MF procedure, which is introduced in (Zhu et al., 2007).

3.3.3 Convergence Speed

We use the HA data set to illustrate the convergence speed of RRMF. The objective function values against the iteration number T are plotted in Figure 3.2(a), from which we can see that RRMF converges very fast. The average classification accuracy of the 5-fold cross validation against T is shown in Figure 3.2(b). We can see that even though the initial value (corresponding to $T = 0$) is not satisfactory (accuracy = 65.5%), RRMF can still achieve very promising and stable performance without requiring many iterations. Because we can achieve promising performance when $T \geq 5$, we set $T = 5$ in all our following experiments.



(a) Objective function



(b) Accuracy

Figure 3.2: Convergence properties of RRMF.

3.3.4 Performance

According to our prior knowledge, the links in Cora data set are homophily links. Hence, the Cora data set satisfies the model assumption of RRMF but it does not satisfy the model assumption of link-content MF. We first test RRMF on Cora to verify that when the model assumption is satisfied, RRMF can dramatically outperform link-content MF and other methods. The average classification accuracies with standard deviations are shown in Figure 3.3, from which we can see that RRMF does outperform other methods dramatically. Even though the link-content sup. MF method uses label information for MF, RRMF can still give much better result, showing that it is indeed very effective. Comparing RRMF to PCA, we can see that the good performance of RRMF does not come from a good initialization but from the learning algorithm itself. Comparing RRMF to MMMF, we can see that the relational information is very useful. Comparing RRMF to directed graph regularization, we can see that the content information also does great help for classification.

We also test RRMF on the WebKB data set. Here, we adopt the normalized Laplacian. The performance is shown in Figure 3.4. It should be noted that the WebKB data set does not satisfy the model assumption of RRMF because it contains SE links. However, with simple preprocessing, RRMF can still achieve performance comparable to the link-content MF method and its supervised counterpart, and outperform other methods.

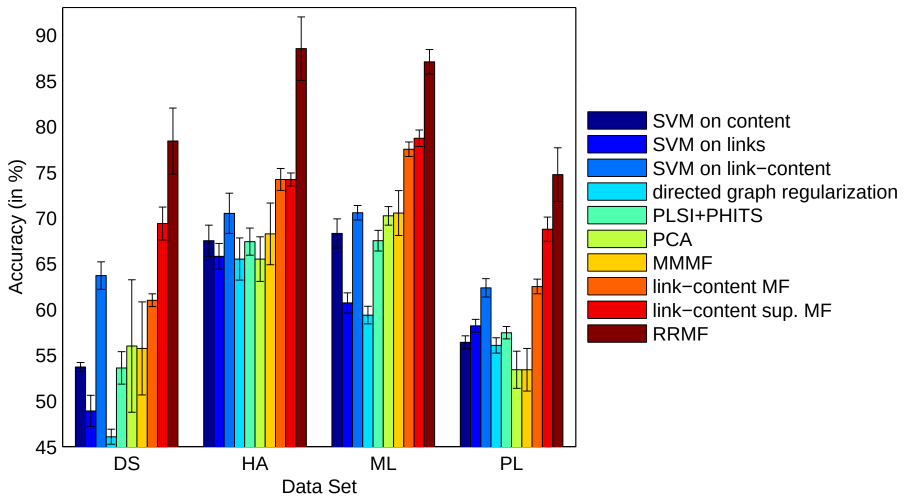


Figure 3.3: Average classification accuracies with standard deviations on the Cora data set.

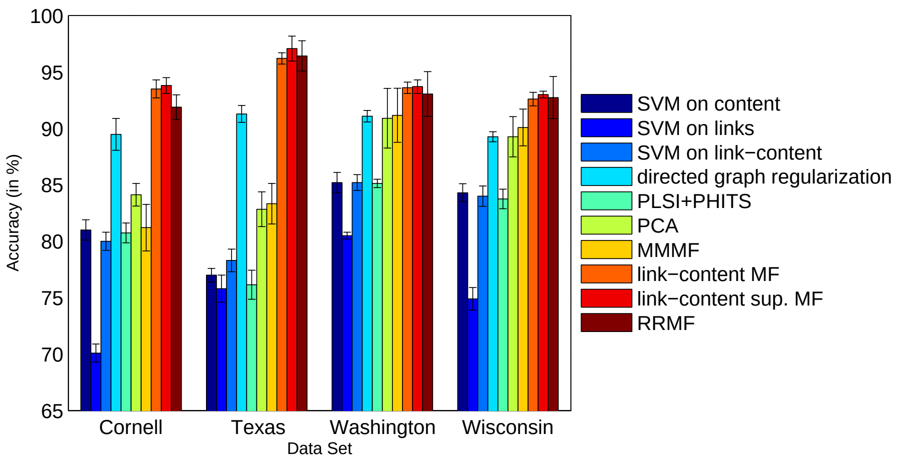
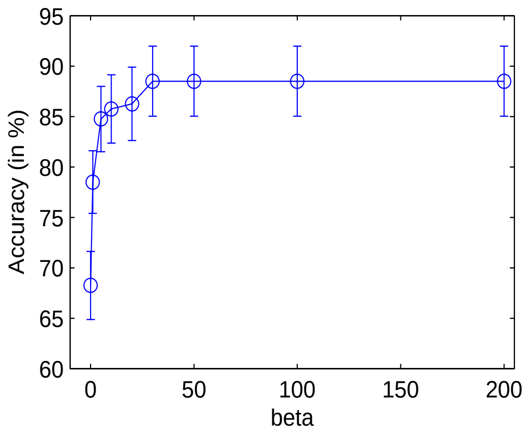


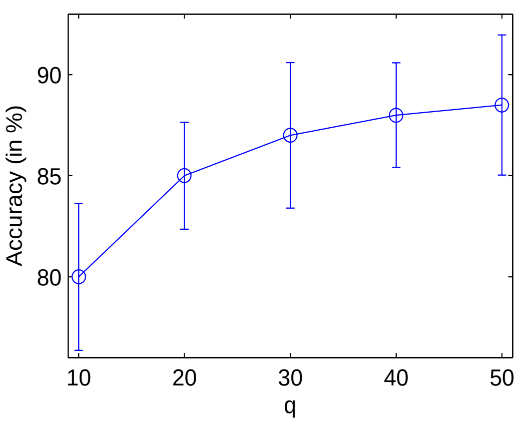
Figure 3.4: Average classification accuracies with standard deviations on the WebKB data set.

3.3.5 Sensitivity to Parameters

We have illustrated the effect of the number of iterations in Section 3.3.3. Here, we examine the sensitivity of RRMF to β and the number of factors q. We again use the HA data set for illustration. Figure 3.5 illustrates the accuracy of RRMF when β and q take different values. From Figure 3.5(a), we can see that the smaller the β, the worse the performance will be. Larger β means that the relational information plays a more significant role in the learning procedure. Hence, we can conclude that the relational information is very important. When β exceeds some value (around 30), the performance becomes very stable, which means RRMF is not sensitive to β. From 3.5(b), we can see that as q increases, the accuracy also increases. But larger q will incur higher computation cost. Hence, in real applications, we need to consider the tradeoff between computation cost and accuracy.



(a) Effect of β



(b) Effect of q

Figure 3.5: Sensitivity to parameters of RRMF.

3.4 Conclusion

In this chapter, we propose a novel MF framework, called RRMF, to model relational data of homophily links. One promising property of RRMF is that it can also be used to model data of SE links just by the inclusion of a very simple preprocessing step. Experimental results verify that RRMF can dramatically outperform other methods on data of homophily links, and can achieve performance comparable with state-of-the-art methods on data of SE links. Another attractive property of RRMF is that its training time is linear in the number of training entities, which makes it scalable to large-scale problems. Moreover, the learning algorithm of RRMF is guaranteed to be convergent and is very stable.

CHAPTER 4

PROBABILISTIC RELATIONAL PCA

4.1 Introduction

Using a low-dimensional embedding to summarize a high-dimensional data set has been widely used for exploring the structure in the data. The methods for discovering such low-dimensional embedding are often referred to as dimensionality reduction (DR) methods. Principal component analysis (PCA) (Jolliffe, 2002) is one of the most popular DR methods with great success in many applications. As a more recent development, probabilistic PCA (PPCA) (Tipping & Bishop, 1999) provides a probabilistic formulation of PCA (Jolliffe, 2002) based on a Gaussian latent variable model (Bartholomew & Knott, 1999). Compared with the original non-probabilistic derivation of PCA in (Howard, 1933), PPCA possesses a number of practical advantages. For example, PPCA can naturally deal with missing values in the data; the expectation-maximization (EM) algorithm (Dempster et al., 1977) used to learn the parameters in PPCA may be more efficient for high-dimensional data; it is easy to generalize the single model in PPCA to the mixture model case; furthermore, PPCA as a probabilistic model can naturally exploit Bayesian methods (Bishop, 1998).

Like many existing DR methods, both PCA and PPCA assume that the data are i.i.d. flat data. However, as we have said, many relational data contain relationships or links between (some) instances in the data in addition to the textual content information which is represented in the form of feature vectors.[1] *On one hand*, the link structure among instances cannot be exploited easily when traditional DR methods such as PCA are applied to relational data. Very often, the useful relational information is simply discarded. One possible use of the relational information in PCA or PPCA is to first convert the link structure into the format of flat data by extracting some additional features from the links. However, as argued in (Getoor & Taskar, 2007), this approach fails to capture some important structural information in the data. *On the other hand*, the i.i.d. assumption underlying PCA and PPCA is unreasonable for relational data. Therefore, PCA and PPCA, or more generally most existing DR methods based on the i.i.d. assumption, are not suitable for relational data analysis.

[1]As in RRMF, we use linked document classification as a running example for relational data analysis. Hence, there are two matrices in our data set, one being the content (attribute) matrix and the other being the adjacency (link) matrix.

In this chapter, a novel DR method called *probabilistic relational PCA* (PRPCA) is proposed for relational data analysis. By explicitly modeling the covariance between instances as derived from the relational information, PRPCA seamlessly integrates relational information and textual content information into a unified probabilistic framework. Two learning algorithms, one based on a closed-form solution and the other based on an EM algorithm (Dempster et al., 1977), are proposed to learn the parameters of PRPCA. Although the i.i.d. assumption is no longer adopted in PRPCA, the learning algorithms for PRPCA can still be devised easily like those for PPCA which makes explicit use of the i.i.d. assumption. Extensive experiments on some real-world data sets show that PRPCA can effectively utilize the relational information to dramatically outperform PCA.

Notations for this chapter As in (Tipping & Bishop, 1999), $\{\mathbf{t}_n\}_{n=1}^N$ denotes a set of observed d-dimensional data (content) vectors, the $d \times q$ matrix $\mathbf{W}$ denotes the q principal axes (or called factor loadings), $\boldsymbol{\mu}$ denotes the data sample mean, and $\mathbf{x}_n = \mathbf{W}^T(\mathbf{t}_n - \boldsymbol{\mu})$ denotes the corresponding q principal components (or called latent variables) of $\mathbf{t}_n$. We further use the $d \times N$ matrix $\mathbf{T}$ to denote the content matrix with $\mathbf{T}_{*n} = \mathbf{t}_n$, and the $q \times N$ matrix $\mathbf{X}$ to denote the latent variables of $\mathbf{T}$ with $\mathbf{X}_{*n} = \mathbf{W}^T(\mathbf{t}_n - \boldsymbol{\mu})$. For relational data, the $N \times N$ matrix $\mathbf{A}$ denotes the adjacency (link) matrix of the N instances. In PRPCA, we assume that the links are undirected. For those data with directed links, we will convert the directed links into undirected links which can keep the original physical meaning of the links. This will be described in detail in Section 4.3.1, and an example will be given in Section 4.4. Hence, $A_{ij} = 1$ if there exists a relationship between instances i and j, and otherwise $A_{ij} = 0$. Moreover, we always assume that there exist no self-links, i.e., $A_{ii} = 0$.

4.2 Probabilistic PCA

To set the stage for the next section which introduces our PRPCA model, we first briefly present the derivation for PPCA (Tipping & Bishop, 1999), which was originally based on (vector-based) multivariate normal distributions, from the perspective of matrix variate normal distributions (Gupta & Nagar, 2000).

If we use $\boldsymbol{\Upsilon}$ to denote the Gaussian noise process and assume that $\boldsymbol{\Upsilon}$ and the latent variable matrix $\mathbf{X}$ follow these distributions:

$$\boldsymbol{\Upsilon} \sim \mathcal{N}_{d,N}(\mathbf{0}, \sigma^2 \mathbf{I}_d \otimes \mathbf{I}_N), \tag{4.1}$$

$$\mathbf{X} \sim \mathcal{N}_{q,N}(\mathbf{0}, \mathbf{I}_q \otimes \mathbf{I}_N), \tag{4.2}$$

we can express a generative model as follows:

$$\mathbf{T} = \mathbf{W}\mathbf{X} + \boldsymbol{\mu}\mathbf{e}^T + \boldsymbol{\Upsilon}.$$

Based on some properties of matrix variate normal distributions in (Gupta & Nagar, 2000), we get the following results:

$$\mathbf{T} \mid \mathbf{X} \sim \mathcal{N}_{d,N}(\mathbf{W}\mathbf{X} + \boldsymbol{\mu}\mathbf{e}^T, \sigma^2\mathbf{I}_d \otimes \mathbf{I}_N),$$
$$\mathbf{T} \sim \mathcal{N}_{d,N}\left(\boldsymbol{\mu}\mathbf{e}^T, (\mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}_d) \otimes \mathbf{I}_N\right). \tag{4.3}$$

Let $\mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}_d$. The corresponding log-likelihood of the observation matrix $\mathbf{T}$ is then

$$\mathcal{L} = \ln p(\mathbf{T}) = -\frac{N}{2}\Big[d\ln(2\pi) + \ln|\mathbf{C}| + \mathrm{tr}(\mathbf{C}^{-1}\mathbf{S})\Big], \tag{4.4}$$

where

$$\mathbf{S} = \frac{(\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T)(\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T)^T}{N} = \frac{\sum_{n=1}^{N}(\mathbf{T}_{*n} - \boldsymbol{\mu})(\mathbf{T}_{*n} - \boldsymbol{\mu})^T}{N}. \tag{4.5}$$

We can see that $\mathbf{S}$ is just the sample covariance matrix of the content observations. It is easy to see that this log-likelihood form is the same as that in (Tipping & Bishop, 1999). Using matrix notations, the graphical model of PPCA based on matrix variate normal distributions is shown in Figure 4.1 below.

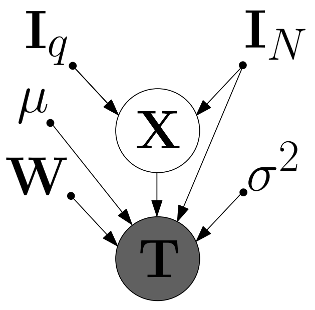


Figure 4.1: The graphical model of PPCA, in which $\mathbf{T}$ is the observation matrix, $\mathbf{X}$ is the latent variable matrix, $\boldsymbol{\mu}$, $\mathbf{W}$ and σ^2 are the parameters to learn, and the other quantities are kept constant.

4.3 Probabilistic Relational PCA

PPCA assumes that all the observations are independent and identically distributed. Although this i.i.d. assumption can make the modeling process much simpler and has achieved great success in many traditional applications, this assumption is however very unreasonable for relational data (Getoor & Taskar, 2007). In relational data, the attributes of connected (linked) instances are often *correlated*.

In this section, a probabilistic relational PCA model, called PRPCA, is proposed to integrate both the relational information and the content information seamlessly into a unified framework by eliminating the i.i.d. assumption. Based on our reformulation of

PPCA using matrix variate notations as presented in the previous section, we can obtain PRPCA just by introducing some relatively simple (yet very effective) modifications. A promising property is that the computation needed for PRPCA is as simple as that for PPCA even though we have eliminated the restrictive i.i.d. assumption.

4.3.1 Model Formulation

Assume that the latent variable matrix $\mathbf{X}$ has the following distribution:

$$\mathbf{X} \sim \mathcal{N}_{q,N}(\mathbf{0}, \mathbf{I}_q \otimes \mathbf{\Phi}). \tag{4.6}$$

According to Theorem 2.2, we can get $\text{cov}(\mathbf{X}_{i*}) = \mathbf{\Phi}$ $(i \in \{1, \ldots, q\})$, which means that $\mathbf{\Phi}$ actually reflects the covariance between the instances. From (4.2), we can see that $\text{cov}(\mathbf{X}_{i*}) = \mathbf{I}_N$ for PPCA, which also coincides with the i.i.d. assumption of PPCA.

Hence, to eliminate the i.i.d. assumption for relational data, one direct way is to use a non-identity covariance matrix $\mathbf{\Phi}$ for the distribution of $\mathbf{X}$ in (4.6). This $\mathbf{\Phi}$ should reflect the physical meaning (semantics) of the relationships between entities, which will be discussed in detail later. Similarly, we can also change the $\mathbf{I}_N$ in (4.1) to $\mathbf{\Phi}$ to eliminate the i.i.d. assumption for the noise process.

Relational Covariance Construction

Because the covariance matrix $\mathbf{\Phi}$ in PRPCA is constructed from the relational information in the data, we refer to it as *relational covariance* here.

The goal of PCA and PPCA is to find those principal axes onto which the retained variance under projection is maximal (Tipping & Bishop, 1999; Jolliffe, 2002). For one specific $\mathbf{X}$, the retained variance is $\text{tr}[\mathbf{X}\mathbf{X}^T]$. If we rewrite (4.2) as

$$p(\mathbf{X}) = \frac{\exp\left\{\text{tr}[-\frac{1}{2}\mathbf{X}\mathbf{X}^T]\right\}}{(2\pi)^{qN/2}} = \frac{\exp\left\{-\frac{1}{2}\text{tr}[\mathbf{X}\mathbf{X}^T]\right\}}{(2\pi)^{qN/2}}, \tag{4.7}$$

we have the following observation:

Observation 4.1 *For PPCA, the larger the retained variance of* $\mathbf{X}$*, i.e., the more* $\mathbf{X}$ *approaches the destination point, the lower is the probability density at* $\mathbf{X}$ *given by the prior.*

Here, the *destination point* refers to the point where the goal of PPCA is achieved, i.e., the retained variance is maximal. Moreover, we use the retained variance as a *measure* to define the gap between two different points. The smaller is the gap between the retained variance of two points, the more they approach each other.

Because the design principle of PRPCA is similar to that of PPCA, our working hypothesis here is that Observation 4.1 can also guide us to design the relational covariance of PRPCA. Its effectiveness will be empirically verified in Section 4.4.

In PRPCA, we assume that the attributes of two linked instances are positively correlated.[2] Under this assumption, the *ideal goal* of PRPCA should be to make the latent representations of two instances as close as possible if there exists a link between them. Hence, the *measure* to define the gap between two points refers to the closeness of the linked instances, i.e., the summation of the Euclidean distances between the linked instances. Based on Observation 4.1, the more $\mathbf{X}$ approaches the destination point, the lower should be the probability density at $\mathbf{X}$ given by the prior. Hence, under the latent space representation $\mathbf{X}$, the closer the linked instances are, the lower should be the probability density at $\mathbf{X}$ given by the prior. We will prove that if we set $\mathbf{\Phi} = \mathbf{\Delta}^{-1}$ where $\mathbf{\Delta} \triangleq \gamma \mathbf{I}_N + (\mathbf{I}_N + \mathbf{A})^T(\mathbf{I}_N + \mathbf{A})$ with γ being typically a very small positive number to make $\mathbf{\Delta} \succ 0$, we can get an appropriate prior for PRPCA. Note that $A_{ij} = 1$ if there exists a link between instances i and j, and otherwise $A_{ij} = 0$. Because $\mathbf{A}^T = \mathbf{A}$, we can also express $\mathbf{\Delta}$ as $\mathbf{\Delta} = \gamma \mathbf{I}_N + (\mathbf{I}_N + \mathbf{A})(\mathbf{I}_N + \mathbf{A})$.

Let $\tilde{\mathbf{D}}$ denote a diagonal matrix whose diagonal elements $\tilde{D}_{ii} = \sum_j A_{ij}$. It is easy to prove that $(\mathbf{AA})_{ii} = \tilde{D}_{ii}$. Let $\mathbf{B} = \mathbf{AA} - \tilde{\mathbf{D}}$, which means that $B_{ij} = (\mathbf{AA})_{ij}$ if $i \neq j$ and $B_{ii} = 0$. We can get $\mathbf{\Delta} = (1+\gamma)\mathbf{I}_N + 2\mathbf{A} + \mathbf{AA} = (1+\gamma)\mathbf{I}_N + \tilde{\mathbf{D}} + (2\mathbf{A} + \mathbf{B})$. Because $B_{ij} = \sum_{k=1}^N A_{ik}A_{kj}$ for $i \neq j$, we can see that B_{ij} is the number of paths, each with path length 2, from instance i to instance j in the original adjacency graph $\mathbf{A}$. Because the attributes of two linked instances are positively correlated, B_{ij} actually reflects the degree of correlation between instance i and instance j. Let us take the paper citation graph as an example to illustrate this. The existence of a citation relationship between two papers often implies that they are about the same topic. If paper i cites paper k and paper k cites paper j, it is highly likely that paper i and paper j are about the same topic. If there exists another paper $a \neq k$ linking both paper i and paper j as well, the confidence that paper i and paper j are about the same topic will increase. Hence, the larger B_{ij} is, the stronger is the correlation between instance i and instance j. Because $B_{ij} = \sum_{k=1}^N A_{ik}A_{kj} = \mathbf{A}_{*i}^T\mathbf{A}_{*j}$, B_{ij} can also be seen as the similarity between the link vectors of instance i and instance j. Therefore, $\mathbf{B}$ can be seen as a weight matrix (corresponding to a weight graph) derived from the original adjacency matrix $\mathbf{A}$, and $\mathbf{B}$ is also consistent with the physical meaning underlying $\mathbf{A}$.

[2]Links with other physical meanings, such as SE links in web graphs (Zhou et al., 2004), can be transformed into links satisfying the assumption in PRPCA via some preprocessing strategies. One such strategy to preprocess the WebKB data set (Craven et al., 1998) will be given as an example in Section 4.4.

Letting $\mathbf{G} = 2\mathbf{A} + \mathbf{B}$,[3] we can find that $\mathbf{G}$ actually combines the original graph reflected by $\mathbf{A}$ and the derived graph reflected by $\mathbf{B}$ to get a new graph, and puts a weight $2A_{ij}+B_{ij}$ on the edge between instance i and instance j in the new graph. The new weight graph reflected by $\mathbf{G}$ is also consistent with the physical meaning underlying $\mathbf{A}$. Letting $\mathbf{L} \triangleq \mathbf{D}-\mathbf{G}$, where $\mathbf{D}$ is a diagonal matrix whose diagonal elements $D_{ii} = \sum_j G_{ij}$ and $\mathbf{L}$ is called the Laplacian matrix (Chung, 1997) of $\mathbf{G}$, we can get $\mathbf{\Delta} = (1+\gamma)\mathbf{I}_N + \tilde{\mathbf{D}} + \mathbf{D} - \mathbf{L}$. If we define another diagonal matrix $\hat{\mathbf{D}} \triangleq (1+\gamma)\mathbf{I}_N + \tilde{\mathbf{D}} + \mathbf{D}$, we can get $\mathbf{\Delta} = \hat{\mathbf{D}} - \mathbf{L}$.

Then we have

$$\begin{aligned}
\mathrm{tr}[\mathbf{X}\mathbf{\Delta}\mathbf{X}^T] &= \mathrm{tr}[\mathbf{X}\hat{\mathbf{D}}\mathbf{X}^T] - \mathrm{tr}[\mathbf{X}\mathbf{L}\mathbf{X}^T] = \mathrm{tr}[\mathbf{X}\hat{\mathbf{D}}\mathbf{X}^T] - \sum_{k=1}^{q} \mathbf{X}_{k*}\mathbf{L}\mathbf{X}_{k*}^T \\
&= \mathrm{tr}[\mathbf{X}\hat{\mathbf{D}}\mathbf{X}^T] - \frac{1}{2}\sum_{k=1}^{q}\left[\sum_{i=1}^{N}\sum_{j=1}^{N} G_{ij}(X_{ki}-X_{kj})^2\right] \\
&= \mathrm{tr}[\mathbf{X}\hat{\mathbf{D}}\mathbf{X}^T] - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left[G_{ij}\sum_{k=1}^{q}(X_{ki}-X_{kj})^2\right] \\
&= \sum_{i=1}^{N}\hat{D}_{ii}\|\mathbf{X}_{*i}\|^2 - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} G_{ij}\|\mathbf{X}_{*i}-\mathbf{X}_{*j}\|^2. \qquad (4.8)
\end{aligned}$$

Letting $\mathbf{\Phi} = \mathbf{\Delta}^{-1}$, we can get

$$p(\mathbf{X}) = \frac{\exp\left\{\mathrm{tr}[-\frac{1}{2}\mathbf{X}\mathbf{\Delta}\mathbf{X}^T]\right\}}{(2\pi)^{qN/2}\,|\mathbf{\Delta}\,|^{-q/2}} = \frac{\exp\left\{-\frac{1}{2}\mathrm{tr}[\mathbf{X}\mathbf{\Delta}\mathbf{X}^T]\right\}}{(2\pi)^{qN/2}\,|\mathbf{\Delta}\,|^{-q/2}}. \qquad (4.9)$$

The first term $\sum_{i=1}^{N}\hat{D}_{ii}\|\mathbf{X}_{*i}\|^2$ in (4.8) can be treated as a measure of weighted variance of all the instances in the latent space. We can see that the larger $\hat{D}_{ii}$ is, the more weight will be put on instance i, which is reasonable because $\hat{D}_{ii}$ mainly reflects the degree of instance i in the graph. It is easy to see that, for those latent representations having a fixed value of weighted variance $\sum_{i=1}^{N}\hat{D}_{ii}\|\mathbf{X}_{*i}\|^2$, the closer the latent representations of two linked entities are, the larger is their contribution to $\mathrm{tr}[\mathbf{X}\mathbf{\Delta}\mathbf{X}^T]$, and subsequently the less is their contribution to $p(\mathbf{X})$. This means that under the latent space representation $\mathbf{X}$, the closer the linked instances are, the lower is the probability density at $\mathbf{X}$ given by the prior. Hence, we can get an appropriate prior for $\mathbf{X}$ by setting $\mathbf{\Phi} = \mathbf{\Delta}^{-1}$ in (4.6).

[3]This means that we put a 2:1 ratio between $\mathbf{A}$ and $\mathbf{B}$. Other ratios can be obtained by setting $\mathbf{\Delta} = \gamma\mathbf{I}_N + (\alpha\mathbf{I}_N + \mathbf{A})(\alpha\mathbf{I}_N + \mathbf{A}) = \gamma\mathbf{I}_N + \alpha^2\mathbf{I}_N + 2\alpha\mathbf{A} + \mathbf{B}$. Preliminary results show that PRPCA is not sensitive to α as long as α is not too large, but we omit the detailed results here because they are out of the scope of this work.

Model

With the constructed relational covariance $\mathbf{\Phi}$, the generative model of PRPCA is defined as follows:

$$\mathbf{\Upsilon} \sim \mathcal{N}_{d,N}(\mathbf{0}, \sigma^2 \mathbf{I}_d \otimes \mathbf{\Phi}) \tag{4.10}$$

$$\mathbf{X} \sim \mathcal{N}_{q,N}(\mathbf{0}, \mathbf{I}_q \otimes \mathbf{\Phi}), \tag{4.11}$$

$$\mathbf{T} = \mathbf{W}\mathbf{X} + \boldsymbol{\mu}\mathbf{e}^T + \mathbf{\Upsilon},$$

where $\mathbf{\Phi} = \mathbf{\Delta}^{-1}$.

We can further obtain the following results:

$$\mathbf{T} \mid \mathbf{X} \sim \mathcal{N}_{d,N}(\mathbf{W}\mathbf{X} + \boldsymbol{\mu}\mathbf{e}^T, \sigma^2 \mathbf{I}_d \otimes \mathbf{\Phi}), \tag{4.12}$$

$$\mathbf{T} \sim \mathcal{N}_{d,N}\left(\boldsymbol{\mu}\mathbf{e}^T, (\mathbf{W}\mathbf{W}^T + \sigma^2 \mathbf{I}_d) \otimes \mathbf{\Phi}\right). \tag{4.13}$$

The graphical model of PRPCA is illustrated in Figure 4.2(b). For convenience of comparison, we also show the graphical model of PPCA in 4.2(a). We can see that the difference between PRPCA and PPCA lies solely in the difference between $\mathbf{\Phi}$ and $\mathbf{I}_N$. Comparing (4.13) to (4.3), we can find that the observations of PPCA are sampled independently while those of PRPCA are sampled with correlation. In fact, PPCA may be seen as a degenerate case of PRPCA as detailed below in Remark 4.1:

Remark 4.1 *When the i.i.d. assumption holds, i.e., all $A_{ij} = 0$, PRPCA degenerates to PPCA by setting $\gamma = 0$. Note that the only role that γ plays is to make $\mathbf{\Delta} \succ 0$. Hence, in our implementation, we always set γ to a very small positive value, such as 10^{-6}. Actually, we may even set γ to 0, because $\mathbf{\Delta}$ does not have to be pd.*[4] *In Definition 2.3, $\mathbf{\Sigma} \succ 0$ and $\mathbf{\Psi} \succ 0$ can be relaxed to $\mathbf{\Sigma} \succeq 0$ and $\mathbf{\Psi} \succeq 0$. When $\mathbf{\Sigma} \succeq 0$ and $\mathbf{\Psi} \succeq 0$, we say $\mathbf{K}$ follows a singular matrix variate normal distribution (Gupta & Nagar, 2000), and all the derivations for PRPCA are still correct. In our experiment, we find that the performance under $\gamma = 0$ is almost the same as that under $\gamma = 10^{-6}$.*

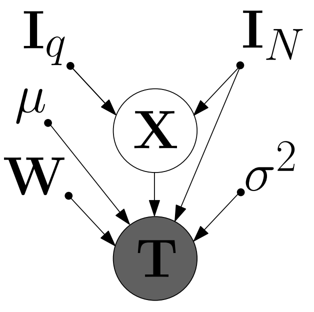
As in PPCA, we set $\mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2 \mathbf{I}_d$. Then the log-likelihood of the observation matrix $\mathbf{T}$ in PRPCA is

$$\mathcal{L}_1 = \ln p(\mathbf{T}) = -\frac{N}{2}\Big[d \ln(2\pi) + \ln |\mathbf{C}| + \operatorname{tr}(\mathbf{C}^{-1}\mathbf{H})\Big] + c, \tag{4.14}$$

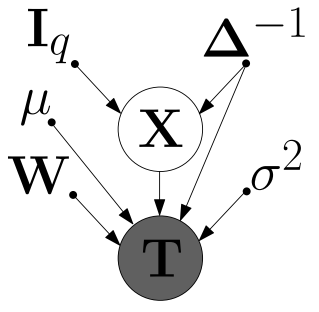
where $c = -\frac{d}{2} \ln |\mathbf{\Phi}|$ can be seen as a constant independent of the parameters $\boldsymbol{\mu}$, $\mathbf{W}$ and σ^2, and $\mathbf{H}$ is defined as:

$$\mathbf{H} = \frac{(\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T)\mathbf{\Delta}(\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T)^T}{N}. \tag{4.15}$$

[4]If $\mathbf{\Delta} \succeq 0$, $\mathbf{\Phi} = \mathbf{\Delta}^{-1}$ may not be computed. But fortunately, in the following learning procedure of PRPCA, we only need $\mathbf{\Delta}$ rather than $\mathbf{\Delta}^{-1}$ for computation.



(a) Model of PPCA



(b) Model of PRPCA

Figure 4.2: Graphical models of PPCA and PRPCA, in which $\mathbf{T}$ is the observation matrix, $\mathbf{X}$ is the latent variable matrix, $\boldsymbol{\mu}$, $\mathbf{W}$ and σ^2 are the parameters to learn, and the other quantities are kept constant.

It is interesting to compare (4.14) with (4.4). We can find that to learn the parameters $\mathbf{W}$ and σ^2, the only difference between PRPCA and PPCA lies in the difference between $\mathbf{H}$ and $\mathbf{S}$. Hence, all the learning techniques derived previously for PPCA are also potentially applicable to PRPCA simply by substituting $\mathbf{S}$ with $\mathbf{H}$.

4.3.2 Learning

By setting the gradient of $\mathcal{L}_1$ with respect to $\boldsymbol{\mu}$ to 0, we can get the maximum-likelihood estimator (MLE) for $\boldsymbol{\mu}$ as follows:

$$\widehat{\boldsymbol{\mu}} = \frac{\mathbf{T\Delta e}}{\mathbf{e}^T\mathbf{\Delta e}}. \tag{4.16}$$

As in PPCA (Tipping & Bishop, 1999), we devise two methods to learn $\mathbf{W}$ and σ^2 in PRPCA, one based on a closed-form solution and the other based on EM.

Closed-Form Solution

Theorem 4.1 *The log-likelihood in (4.14) is maximized when*

$$\mathbf{W}_{ML} = \mathbf{U}_q(\mathbf{\Lambda}_q - \sigma^2_{ML}\mathbf{I}_q)^{1/2}\mathbf{R}, \tag{4.17}$$

$$\sigma^2_{ML} = \frac{\sum_{i=q+1}^{d}\lambda_i}{d-q}, \tag{4.18}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$ are the eigenvalues of $\mathbf{H}$, $\mathbf{\Lambda}_q$ is a $q \times q$ diagonal matrix containing the first q largest eigenvalues, $\mathbf{U}_q$ is a $d \times q$ matrix in which the q column vectors are the principal eigenvectors of $\mathbf{H}$ corresponding to $\mathbf{\Lambda}_q$, and $\mathbf{R}$ is an arbitrary $q \times q$ orthogonal rotation matrix.

The proof of Theorem 4.1 makes use of techniques similar to those in Appendix A of (Tipping & Bishop, 1999) and is omitted here.

EM Algorithm

During the EM learning process, we treat $\{\mathbf{W}, \sigma^2\}$ as parameters, $\mathbf{X}$ as missing data and $\{\mathbf{T}, \mathbf{X}\}$ as complete data. The EM algorithm operates by alternating between the E-step and M-step. Here we only briefly describe the update rules and their derivation is left to Appendix B.1.

In the E-step, the expectation of the complete-data log-likelihood with respect to the distribution of the missing data $\mathbf{X}$ is computed. To compute the expectation of the complete-data log-likelihood, we only need to compute the following *sufficient statistics*:

$$\langle \mathbf{X} \rangle = \mathbf{M}^{-1}\mathbf{W}^T(\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T), \tag{4.19}$$

$$\langle \mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T \rangle = N\sigma^2\mathbf{M}^{-1} + \langle \mathbf{X} \rangle \boldsymbol{\Delta} \langle \mathbf{X} \rangle^T, \tag{4.20}$$

where $\mathbf{M} = \mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I}_q$. Note that all these statistics are computed based on the parameter values obtained from the previous iteration.

In the M-step, to maximize the expectation of the complete-data log-likelihood, the parameters $\{\mathbf{W}, \sigma^2\}$ are updated as follows:

$$\widetilde{\mathbf{W}} = \mathbf{H}\mathbf{W}(\sigma^2\mathbf{I}_q + \mathbf{M}^{-1}\mathbf{W}^T\mathbf{H}\mathbf{W})^{-1}, \tag{4.21}$$

$$\widetilde{\sigma}^2 = \frac{\mathrm{tr}(\mathbf{H} - \mathbf{H}\mathbf{W}\mathbf{M}^{-1}\widetilde{\mathbf{W}}^T)}{d}, \tag{4.22}$$

where $\mathbf{H}$ is defined in (4.15). Note that we use $\mathbf{W}$ here to denote the old value and $\widetilde{\mathbf{W}}$ for the updated new value.

4.3.3 Complexity Analysis

Suppose there are δ nonzero elements in $\boldsymbol{\Delta}$. We can see that the computation cost for $\mathbf{H}$ is $O(dN + d\delta)$. In many applications δ is typically a constant multiple of N. Hence, we can say that the time complexity for computing $\mathbf{H}$ is $O(dN)$.

For the closed-form solution, we have to invert a $d \times d$ matrix. Hence, the computation cost is $O(dN + d^3)$.

For EM, because d is typically larger than q, we can see that the computation cost is $O(dN + d^2qT)$, where T is the number of EM iterations. If the data are of very high dimensionality, EM will be more efficient than the closed-form solution.

4.4 Experiments

Although PPCA possesses additional advantages when compared with the original non-probabilistic formulation of PCA, they will get similar DR results when there exist no missing values in the data. If the task is to classify instances in the low-dimensional embedding, the classifiers based on the embedding results of PCA and PPCA are expected to achieve comparable results. Hence, we adopt PCA as the baseline to study the performance of PRPCA. For the EM algorithm of PRPCA, we use PCA to initialize $\mathbf{W}$, σ^2 is initialized to 10^{-6}, and $\gamma = 10^{-6}$. Because the EM algorithm and the closed-form solution achieve similar results, we only report the results of the EM algorithm of PRPCA in the following experiments.

4.4.1 Data Sets and Evaluation Scheme

We use three data sets to evaluate PRPCA. The first two data sets are Cora (McCallum et al., 2000) and WebKB (Craven et al., 1998). We adopt the same strategy as that in RRMF and link-content MF (Zhu et al., 2007) to preprocess these two data sets, the detailed information about which can be found in Section 3.3.1.

The third data set is the PoliticalBook data set used in (Silva et al., 2008). This data set contains 105 books, 43 of which are labeled as liberal ones. Pairs of books that are frequently bought together by the same customer are used to represent the relationships between them. The task is to decide whether a specific book is of liberal political inclination or not. The words in the Amazon.com front page for a book are used as features to represent the book. Each book is represented as bag-of-words. After preprocessing, each book is represented by a feature vector of length 13,178. The original data is available at http://www-personal.unich.edu/mejn/netdata, and the preprocessed data can be downloaded from http://www.statslab.cam.ac.uk/∼silva. This data set is used to demonstrate that PRPCA can also work well on data of very high dimensionality.

We adopt the same strategy as that in RRMF to evaluate PRPCA on the Cora and WebKB data sets. More specifically, we adopt 5-fold cross validation to evaluate our method. After the learning process of PRPCA, we adopt (4.19) to perform DR. Then a linear SVM[5] is trained for classification based on the low-dimensional representation. The average classification accuracy and standard deviation over five repeats are used to report the performance.

For the PoliticalBook data set, we first use PRPCA to perform DR. Then, a Gaussian process classifier (Rasmussen & Williams, 2006) is trained based on the low-dimensional representation, which is the same as the testing procedure of mixed graph Gaussian process (XGP) (Silva et al., 2008).

[5]We use LIBSVM (Chang & Lin, 2001) to train all the SVM classifiers.

4.4.2 Convergence Speed of EM

We use the DS and Cornell data sets to illustrate the convergence speed of the EM learning procedure of PRPCA. The performance on other data sets has similar characteristics, which is omitted here. With $q = 50$, the average classification accuracy based on 5-fold cross validation against the number of EM iterations T is shown in Figure 4.3. We can see that PRPCA achieves very promising and stable performance after a very small number of iterations. We set $T = 5$ in all our following experiments.

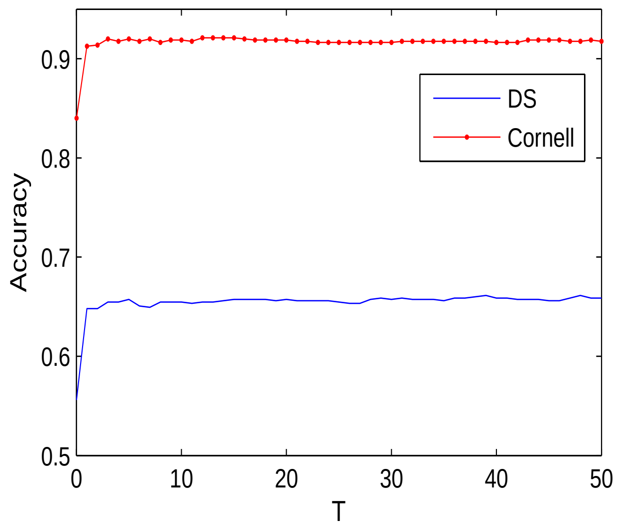


Figure 4.3: Convergence speed of the EM learning procedure of PRPCA.

4.4.3 Visualization

We use the PoliticalBook data set to visualize the DR results of PCA and PRPCA. For the sake of visualization, q is set to 2. The results are depicted in Figure 4.4. We can see that it is not easy to separate the two classes in the latent space of PCA. However, the two classes are better separated from each other in the latent space of PRPCA. Hence, good classification performance can be expected when the examples are classified in the latent space of PRPCA, which will be verified by our subsequent experiment.

4.4.4 Performance

The dimensionality of Cora and WebKB is moderately high, but the dimensionality of PoliticalBook is very high. We evaluate PRPCA on these two different kinds of data to verify its effectiveness in general settings.

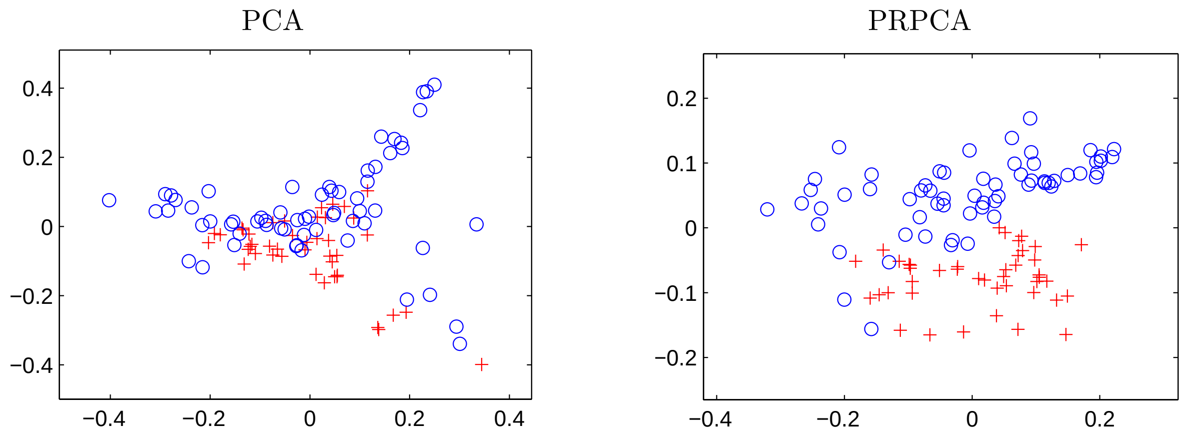


Figure 4.4: Visualization of data points in the latent spaces of PCA and PRPCA for the PoliticalBook data set. The positive and negative examples are shown as red crosses and blue circles, respectively.

Performance on Cora and WebKB

We perform comparison between PRPCA and PCA to demonstrate that PRPCA can exploit relational information successfully to outperform PCA which discards the relational information. The average classification accuracy with its standard deviation based on 5-fold cross validation against the dimensionality of the latent space q is shown in Figure 4.5. We can find that PRPCA can dramatically outperform PCA on all the data sets under any dimensionality, which confirms that the relational information is very informative and PRPCA can utilize it very effectively.

We also perform comparison between PRPCA and those methods evaluated in (Zhu et al., 2007) and RRMF. The methods include: *SVM on content*, *SVM on links*, *SVM on link-content*, *directed graph regularization (DGR)* (Zhou et al., 2004), *PLSI+PHITS* (Cohn & Hofmann, 2000), and *link-content MF* (Zhu et al., 2007)[6]. As in the RRMF and link-content MF method, we set $q = 50$ for PRPCA. The results are shown in Figure 4.6. We can see that PRPCA and link-content MF achieve the best performance among all the evaluated methods. Compared with link-content MF, PRPCA performs slightly better on DS and HA while performing slightly worse on ML and Texas, and achieves comparable performance on the other data sets. We can conclude that the overall performance of PRPCA is comparable with that of link-content MF. Unlike link-content MF which is transductive in nature, PRPCA naturally supports inductive inference. More specifically, we can apply the learned transformation matrix of PRPCA to perform DR for unseen test data, while link-content MF can only perform DR for those data available during the

[6] *Link-content sup. MF* in (Zhu et al., 2007) is not adopted here for comparison. Because during the DR procedure link-content sup. MF employs additional label information which is not employed by other DR methods, it is unfair to directly compare it with other methods.

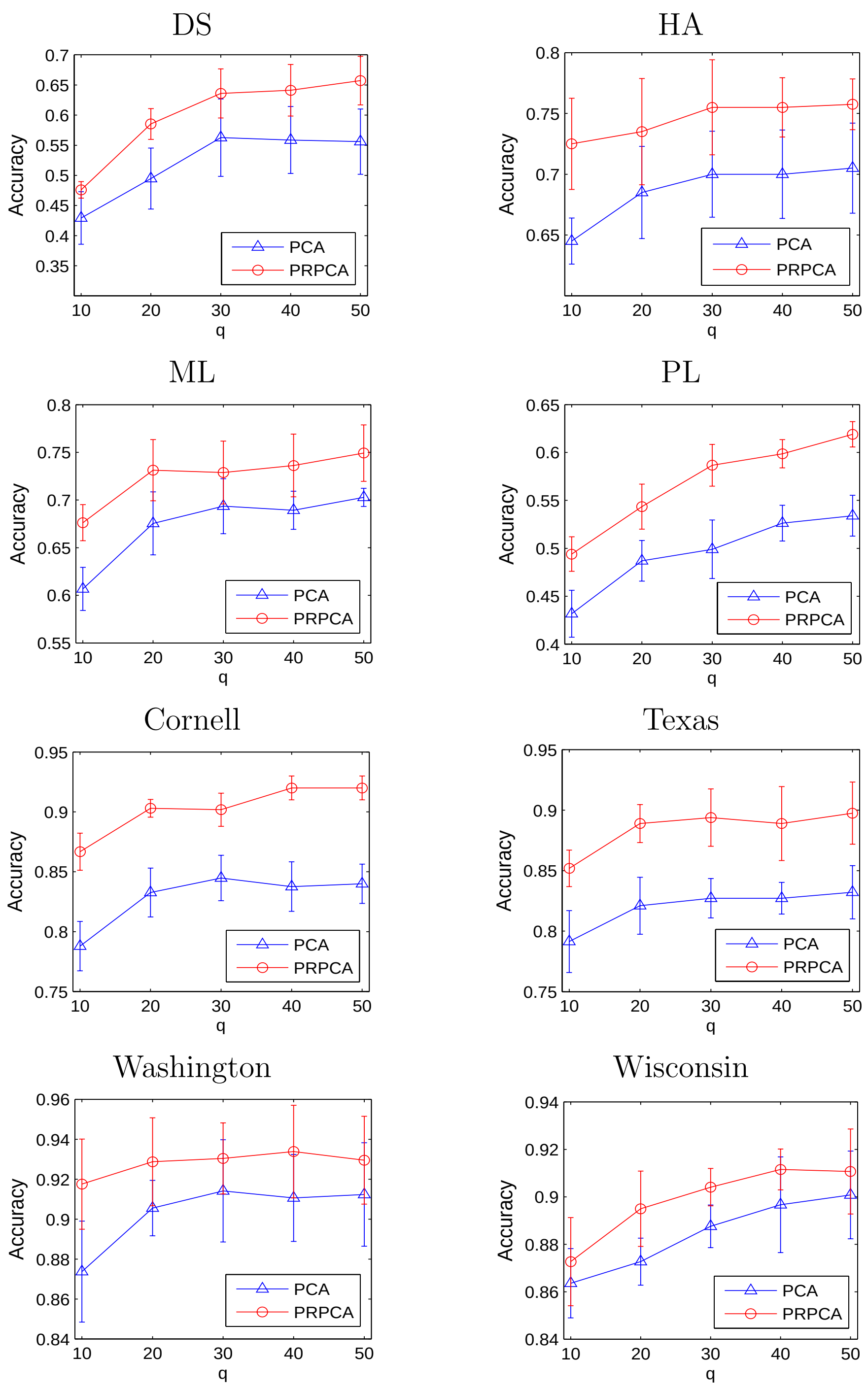


Figure 4.5: Comparison between PRPCA and PCA on Cora and WebKB.

training phase. From Chapter 3, we find that RRMF (Li & Yeung, 2009) achieves better performance than PRPCA on the Cora data set. However, similar to link-content MF, RRMF cannot be used for inductive inference either.

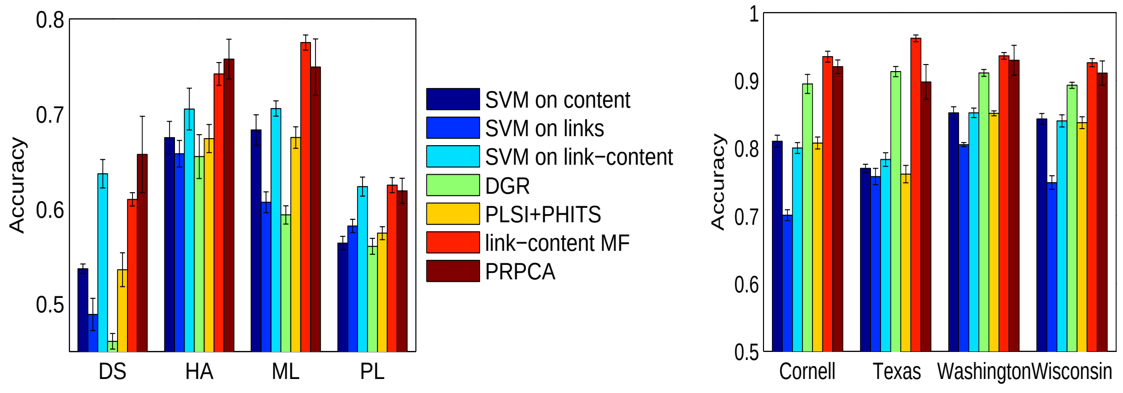


Figure 4.6: Comparison between PRPCA and other methods on Cora and WebKB.

Performance on PoliticalBook

As in XGP (Silva et al., 2008), we randomly choose half of the whole data for training and the rest for testing. This subsampling process is repeated for 100 rounds and the average *area under the ROC curve* (AUC) with its standard deviation is reported in Table 4.1, where GPC is a Gaussian process classifier trained on the original feature representation, relational Gaussian process (RGP) is the method in (Chu et al., 2007), and latent Wishart process (LWP) (Li et al., 2009b) will be introduced in Chapter 6. For PCA and PRPCA, we first use them to perform DR, and then a Gaussian process classifier is trained based on the low-dimensional representation. Here, we set $q = 5$ for both PCA and PRPCA. We can see that on this data set, PRPCA also dramatically outperforms PCA and achieves performance comparable with the state of the art. Note that RGP and XGP cannot learn a low-dimensional embedding for the instances. Although LWP can also learn a low-dimensional embedding for the instances, the computation cost to obtain a low-dimensional embedding for a test instance is $O(N^3)$ because it has to invert the kernel matrix defined on the training data.

Table 4.1: Performance on the PoliticalBook data set.

GPC	RGP	XGP	LWP	PCA	PRPCA
0.92	0.98	0.98	0.98 ± 0.02	0.92 ± 0.03	0.98 ± 0.02

4.5 Conclusion

The i.i.d. assumption adopted by most traditional DR methods is unreasonable for relational data. In this chapter, we propose a novel DR method, called PRPCA, for relational data analysis. Although we no longer make the i.i.d. assumption in PRPCA, the learning algorithms for PRPCA can still be devised easily like those for PPCA. Extensive experiments on some real-world data sets show that PRPCA can effectively utilize the relational information to dramatically outperform PCA.

CHAPTER 5

SPARSE PROBABILISTIC RELATIONAL PROJECTION

5.1 Introduction

PCA (Jolliffe, 2002) and PPCA (Tipping & Bishop, 1999) are very popular dimensionality reduction methods which have been widely used to explore the structure of a high-dimensional data set by mapping the data set into a low-dimensional space via a projection (or called transformation) matrix. However, it is difficult to interpret the results of PCA and PPCA because each principal component is a linear combination of all the original variables. To achieve interpretability, some sparse versions of PCA or PPCA have been proposed by enforcing many entries of the projection matrix to go to zero. Sparse PCA (SPCA) (Zou et al., 2006) first reformulates PCA as a regression-type optimization problem and then applies the *elastic net* (Zou & Hastie, 2005) constraint on the regression coefficients to get a sparse projection matrix. In (Sigg & Buhmann, 2008), sparsity is achieved by putting a 1-norm (L_1) constraint on the projection matrix during the EM (Dempster et al., 1977) learning procedure of PPCA. In (Archambeau & Bach, 2008) and (Guan & Dy, 2009), sparse versions of PPCA are proposed by putting some sparsity-inducing priors such as the Jeffreys prior on the projection matrix.

All the variants of PCA and sparse PCA mentioned above assume that the instances are i.i.d. Hence, they are not suitable for modeling relational data in which the instances are not i.i.d. PRPCA, which extends PPCA by eliminating the i.i.d. assumption, can perform dimensionality reduction for relational data. By explicitly modeling the covariance between instances, PRPCA dramatically outperforms PCA and PPCA. However, as in PCA and PPCA, the results learned by PRPCA also lack interpretability.

In this chapter, we propose a sparse version of PRPCA, called *sparse probabilistic relational projection* (SPRP), to learn a sparse projection matrix for relational dimensionality reduction. The sparsity in SPRP is achieved by imposing on the projection matrix a sparsity-inducing prior such as the Laplace prior or Jeffreys prior. We propose an EM algorithm to learn the parameters of SPRP. Compared with PRPCA, the sparsity in SPRP not only makes the results more interpretable but also makes the projection operation much more efficient without compromising its accuracy. Furthermore, compared with traditional sparse projection methods based on the i.i.d. assumption, SPRP can learn a more discriminative projection by explicitly modeling the covariance between instances.

5.2 Sparse Probabilistic Relational Projection

We propose to put a sparsity-inducing prior on $\mathbf{W}$ to encourage many of its entries to go to zero. The resulting model is called *sparse probabilistic relational projection* (SPRP) due to its sparsity property. Although there exist many sparsity-inducing priors in the literature, e.g., (Figueiredo, 2003; Caron & Doucet, 2008; Archambeau & Bach, 2008; Guan & Dy, 2009), here we consider only two of them, the Laplace prior and Jeffreys prior. Using other sparsity-inducing priors in SPRP is expected to follow the same principle and will be left to our future pursuit.

5.2.1 SPRP with Laplace Prior

In SPCA (Zou et al., 2006), sparsity is achieved by putting an L_1 regularization term on the projection matrix. Here we learn a sparse $\mathbf{W}$ for SPRP in a similar way. However, unlike SPCA which is formulated from a non-probabilistic view, SRPR is based on a probabilistic formulation which can automatically learn the hyperparameters while the non-probabilistic formulation cannot.

We adopt the Laplace (i.e., double-exponential) prior (Park & Casella, 2008; Guan & Dy, 2009) for $\mathbf{W}$:

$$\begin{aligned} p(W_{ij} \mid \lambda) &= \frac{\sqrt{\lambda}}{2} \exp\left\{-\sqrt{\lambda}\|W_{ij}\|_1\right\}, \\ p(\mathbf{W} \mid \lambda) &= \prod_{i=1}^{d}\prod_{j=1}^{q} p(W_{ij} \mid \lambda) \\ &= \left(\frac{\sqrt{\lambda}}{2}\right)^{dq} \exp\left\{-\sqrt{\lambda}\|\mathbf{W}\|_1\right\}, \end{aligned}$$

where $\|\cdot\|_1$ denotes the absolute value for a scalar and the 1-norm for a matrix.

If we use $\Theta = \{\boldsymbol{\mu}, \mathbf{W}, \sigma^2\}$ to denote the set of parameters in PRPCA and SPRP, the log-posterior of Θ can be computed as follows:

$$\begin{aligned} \ln p(\Theta \mid \mathbf{T}) &= \ln p(\mathbf{T} \mid \Theta) + \ln p(\Theta) + c_0 \\ &= -\frac{N}{2}\Big[\ln|\mathbf{C}| + \mathrm{tr}(\mathbf{C}^{-1}\mathbf{H})\Big] - \sqrt{\lambda}\|\mathbf{W}\|_1 + \ln p(\boldsymbol{\mu}) + \ln p(\sigma^2) + c_1, \end{aligned} \tag{5.1}$$

where $\mathbf{C}$ and $\mathbf{H}$ are the same as those in PRPCA, c_0 and c_1 are constants independent of Θ [1].

For simplicity, here we adopt the *maximum a posteriori* (MAP) strategy to estimate the parameters. Due to the L_1 constraint on $\mathbf{W}$, the MAP estimation of $\mathbf{W}$ will naturally induce sparsity, which means that many entries of $\mathbf{W}$ will be automatically driven to zero during the learning procedure. Here, we assume that $p(\boldsymbol{\mu})$ and $p(\sigma^2)$ are uniform.

Remark 5.1 *Of course, we may also put non-uniform priors, such as conjugate priors, on $\boldsymbol{\mu}$ and σ^2. Here, uniform priors for $\boldsymbol{\mu}$ and σ^2 are adopted mainly for fair comparison because they are also adopted in PRPCA. Alternatively, we may also treat all the parameters as random variables and resort to fully Bayesian methods, such as variational methods (Jordan et al., 1999), for learning and inference. However, since our focus is on demonstrating the promising advantages of sparsity under the PRPCA framework, all these possible variants are left to future extensions.*

It is not easy to directly optimize the objective function in (5.1) though. As in (Park & Casella, 2008; Guan & Dy, 2009), we adopt a hierarchical interpretation of the Laplace prior:

$$p(Z_{ij} \mid \lambda) = \frac{\lambda}{2} \exp\left\{-\frac{\lambda}{2} Z_{ij}\right\}, \text{ for } Z_{ij} \geq 0, \tag{5.2}$$

$$W_{ij} \mid Z_{ij} \sim \mathcal{N}(0, Z_{ij}). \tag{5.3}$$

It is easy to show that this hierarchical reformulation is equivalent to the original Laplace prior, because

$$\begin{aligned} p(W_{ij} \mid \lambda) &= \int p(W_{ij} \mid Z_{ij}) p(Z_{ij} \mid \lambda) dZ_{ij} \\ &= \frac{\sqrt{\lambda}}{2} \exp\left\{-\sqrt{\lambda} \|W_{ij}\|_1\right\}. \end{aligned} \tag{5.4}$$

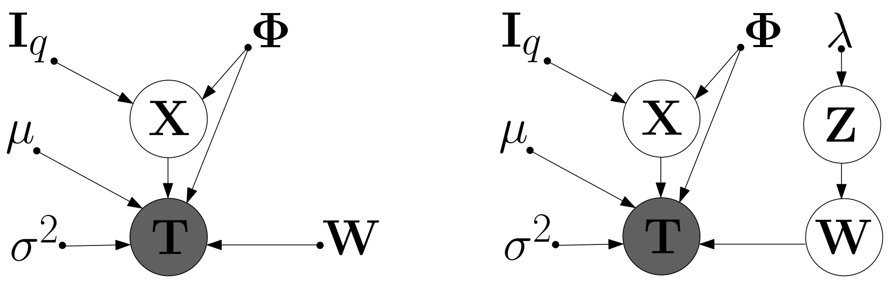
Figure 5.1(b) depicts the graphical model of SPRP as compared with that of PRPCA in Figure 5.1(a).

Learning

By setting the gradient of $\ln p(\Theta \mid \mathbf{T})$ with respect to $\boldsymbol{\mu}$ to zero, we get the (closed-form) MAP estimate for $\boldsymbol{\mu}$ as follows:

$$\widehat{\boldsymbol{\mu}} = \frac{\mathbf{T}\boldsymbol{\Delta}\mathbf{e}}{\mathbf{e}^T \boldsymbol{\Delta}\mathbf{e}}. \tag{5.5}$$

[1]Here, $p(\mathbf{T} \mid \Theta)$ is the same as $p(\mathbf{T})$ in PRPCA because $p(\mathbf{T})$ in PRPCA is actually dependent on Θ. In SPRP, we explicitly show that $\mathbf{T}$ is dependent on Θ for convenience of discussion.



(a) PRPCA (b) SPRP

Figure 5.1: Graphical models of PRPCA and SPRP, in which $\mathbf{T}$ is the observation matrix, $\mathbf{X}$ and $\mathbf{Z}$ are the latent variable matrices, $\boldsymbol{\mu}$, $\mathbf{W}$ and σ^2 are the parameters to learn, λ is the hyperparameter, and the other quantities are kept constant.

For the other parameters ($\mathbf{W}$ and σ^2) of SPRP, we derive an (iterative) EM (Dempster et al., 1977) algorithm to learn them. For the rest of this chapter, we still use Θ to refer to the parameters but they only contain $\mathbf{W}$ and σ^2 because $\boldsymbol{\mu}$ can be directly computed by (5.5). During the EM learning procedure, we treat $\{\mathbf{Z}, \mathbf{X}\}$ as missing data and $\{\mathbf{T}, \mathbf{Z}, \mathbf{X}\}$ as complete data. The EM algorithm for MAP estimation operates by alternating between the following two steps:

- **E-step:** The expectation of the complete-data log-posterior with respect to the distribution of the missing variables $\{\mathbf{Z}, \mathbf{X}\}$ is computed. This expected value is often called the Q-function which is defined as follows:

$$Q\left(\Theta \mid \Theta(t)\right) = \int d\mathbf{Z}\, d\mathbf{X}\, p(\mathbf{Z}, \mathbf{X} \mid \Theta(t), \mathbf{T}) \ln p(\Theta \mid \mathbf{T}, \mathbf{Z}, \mathbf{X}).$$

- **M-step:** The Q-function is maximized to update the parameters:

$$\Theta(t+1) = \operatorname*{argmax}_{\Theta} Q\left(\Theta \mid \Theta(t)\right).$$

The whole EM learning procedure is summarized in Algorithm 5.1 and the detailed derivation is left to Appendix B.2. Note that as in PRPCA, we use $\mathbf{W}$ and σ^2 to denote the old values and $\widetilde{\mathbf{W}}$ and $\widetilde{\sigma}^2$ for the updated ones.

To learn the hyperparameter λ, we may resort to the cross-validation strategy. Alternatively, we may treat λ as one of the parameters, just like $\mathbf{W}$, to get the following EM updating equation:

$$\lambda = \frac{2dq}{\sum_{i=1}^{d}\sum_{j=1}^{q}\langle Z_{ij}\rangle}.$$

Algorithm 5.1 EM algorithm for SPRP

```
Initialize $\mathbf{W}$ and $\sigma^2$.
for $t = 1$ to $T$
    E-step: Compute the sufficient statistics
      $\mathbf{M} = \mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I}_q$,
      $\langle\mathbf{X}\rangle = \mathbf{M}^{-1}\mathbf{W}^T(\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T)$,
      $\langle\mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T\rangle = N\sigma^2\mathbf{M}^{-1} + \langle\mathbf{X}\rangle\boldsymbol{\Delta}\langle\mathbf{X}\rangle^T$,
      $\langle\frac{1}{Z_{ij}}\rangle = \frac{\sqrt{\lambda}}{\|W_{ij}\|_1}$.

    M-step: Update the parameters
      for $i = 1$ to $d$
        $\boldsymbol{\Sigma}_i = \mathrm{diag}\big(\frac{\|W_{i1}\|_1}{\sqrt{\lambda}}, \cdots, \frac{\|W_{iq}\|_1}{\sqrt{\lambda}}\big)$,
        $\widetilde{\mathbf{W}}_{i*} = \mathbf{H}_{i*}\mathbf{W}\mathbf{M}^{-1}\boldsymbol{\Sigma}_i\big[(\sigma^2\mathbf{I}_q + \mathbf{M}^{-1}\mathbf{W}^T\mathbf{H}\mathbf{W})\mathbf{M}^{-1}\boldsymbol{\Sigma}_i + \frac{\sigma^2}{N}\mathbf{I}_q\big]^{-1}$.
      end for
      $\widetilde{\sigma}^2 = \frac{\mathrm{tr}[\mathbf{H} - \mathbf{H}\mathbf{W}\mathbf{M}^{-1}\widetilde{\mathbf{W}}^T]}{d}$.
end for
```

5.2.2 SPRP with Jeffreys Prior

The Jeffreys prior (Figueiredo, 2003; Guan & Dy, 2009) is a noninformative prior. To use the Jeffreys prior, we only need to change the density function of Z_{ij} in (5.2) to:

$$p(Z_{ij}) \propto \frac{1}{Z_{ij}}.$$

We can see that there exist no hyperparameters in the Jeffreys prior but the Laplace prior does have the hyperparameter λ which needs to be learned.

With the Jeffreys prior, the log-posterior can be computed as follows:

$$\begin{aligned} \ln p(\Theta \mid \mathbf{T}) &= \ln p(\mathbf{T} \mid \Theta) + \ln p(\Theta) + c_2 \\ &= -\frac{N}{2}\Big[\ln|\mathbf{C}| + \mathrm{tr}(\mathbf{C}^{-1}\mathbf{H})\Big] - \ln\|\mathbf{W}\|_1 + \ln p(\boldsymbol{\mu}) + \ln p(\sigma^2) + c_3, \end{aligned} \quad (5.6)$$

where c_2 and c_3 are constants independent of the parameters. We can see that the only difference between (5.1) and (5.6) lies in the difference between the regularization terms $\sqrt{\lambda}\|\mathbf{W}\|_1$ and $\ln\|\mathbf{W}\|_1$.

To learn the parameters for SPRP with the Jeffreys prior, we only need to change $\langle\frac{1}{Z_{ij}}\rangle$ and $\boldsymbol{\Sigma}_i$ in Algorithm 5.1 as follows:

$$\begin{aligned} \langle\frac{1}{Z_{ij}}\rangle &= \frac{1}{W_{ij}^2} \\ \boldsymbol{\Sigma}_i &= \mathrm{diag}\big(W_{i1}^2, \ldots, W_{iq}^2\big). \end{aligned}$$

5.2.3 Complexity Analysis

To train the model, we need $O(dN)$ time to compute $\mathbf{H}$ and $O(Tqd^2 + Tdq^3)$ for the T EM iterations. Hence, the total time complexity is $O(dN + Tqd^2 + Tdq^3)$. If $d > q^2$, Tqd^2 will be larger than Tdq^3 and so the time complexity will become $O(dN + Tqd^2)$ which is equal to that of PRPCA.

If we want to use the learned $\mathbf{W}$ to perform projection, the time complexity will depend on the number of nonzero entries in $\mathbf{W}$. Generally speaking, SPRP has lower projection cost than PRPCA because the $\mathbf{W}$ in SPRP is more sparse than that in PRPCA.

5.3 Experiments

As in PRPCA, we adopt PCA to initialize $\mathbf{W}$, initialize σ^2 to 10^{-6}, and set γ to 10^{-6}. We set the number of EM iterations T to 30 because 30 iterations are sufficient for both PRPCA and SPRP to achieve good performance. The baseline methods for comparison include PCA, sparse probabilistic projection (SPP) (Archambeau & Bach, 2008) and PRPCA. Through the experiments we want to verify the following claims: (1) PCA cannot effectively exploit the relational information in relational data. Furthermore, it cannot learn interpretable results. (2) Due to its i.i.d. assumption, SPP cannot achieve satisfactory performance even though it can learn interpretable results. (3) PRPCA can effectively exploit the relational information, but it cannot learn interpretable results. (4) SPRP not only can effectively exploit the relational information, but it can also learn interpretable results.

5.3.1 Data Sets

Three data sets are used for our experimental evaluation. The first two are the *preprocessed* WebKB (Craven et al., 1998) and Cora (McCallum et al., 2000) data sets used for RRMF and PRPCA. The third data set is called Cora-IR, which contains the information retrieval papers from the original Cora data set (McCallum et al., 2000). All these data sets use the bag-of-words representation for the content information.

The detailed information about the preprocessed WebKB and Cora data sets can be found in Section 3.3.1. Because we do not have the dictionary for generating the bag-of-words representation in the preprocessed WebKB and Cora data sets, we collect another data set, called Cora-IR. The Cora-IR data set contains 350 information retrieval papers from the original Cora set (McCallum et al., 2000). There are four subfields (classes) in Cora-IR: *retrieval*, *filtering*, *extraction*, and *digital_library*. We use the title of each paper for the content information. After preprocessing, we get a dictionary of 787 words. For each word, there is at least one instance (paper) containing it. We will use this dictionary to demonstrate the interpretability of SPRP.

In RRMF and PRPCA, only information about the words (bag-of-words) is used to represent the content information. We expand the original content features by adding some extra features extracted from the original *directed* links. The ith link feature is referred to as *link-to-instance$_i$*. For example, if instance k links to instance i, the ith link feature of instance k will be 1. Otherwise, it will be 0. In fact, this kind of link features can also be treated as content information. For example, given a paper, the *link-to-instance$_i$* feature actually reflects whether the reference part of that paper contains paper i. For a web page, the *link-to-instance$_i$* feature can also be directly extracted from the HTML file (content) of that page. Note that it is somewhat impractical to treat the *linked-by-instance$_i$* features as content features because they cannot be directly extracted from the content of the instances. For example, the papers citing a specific paper i are not included in the content of paper i. After extracting the link features, we combine the original bag-of-words with the link features to obtain the *expanded content features*. We can see that the way to get the expanded content features also assumes that the instances are i.i.d. We will show that this way of using link information is not good enough to capture the structure information in relational data. On the contrary, PRPCA and SPRP, which are not based on the i.i.d. assumption, can provide more effective ways to model relational data. In what follows, we will refer to the original bag-of-words representation as *original content features*.

5.3.2 Laplace Prior vs. Jeffreys Prior

We define the *degree of sparsity* (*DoS*) of $\mathbf{W}$ as follows:

$$DoS = \frac{\text{number of zero entries in } \mathbf{W}}{dq} \times 100\%.$$

From (5.1), we can see that λ in the Laplace prior controls the *DoS* of the learned $\mathbf{W}$. The larger λ is, the larger will the *DoS* of $\mathbf{W}$ be. Here, we vary λ to get different *DoS* and then evaluate the corresponding accuracy.[2] We only report here results on the DS data set because other data sets exhibit similar properties. The accuracy of PRPCA on the DS data set is 68.1%. The corresponding results of SPRP with the Laplace prior are shown in Table 5.1. From the results, we can discover some interesting properties of SPRP:

- In general, the larger the *DoS* is, the lower will the accuracy be. This is reasonable because less information about the features will be used to construct the principal components with larger *DoS*.

[2]The definition of accuracy will be given later in Section 5.3.4.

- Compared with PRPCA, SPRP can achieve a *DoS* as large as 60% without compromising its accuracy. Even when $DoS = 70\%$, the accuracy of SPRP is still comparable with that of PRPCA. This shows that the sparsity pursuit in SPRP is very meaningful because it can obtain interpretable results without compromising its accuracy.

Table 5.1: Accuracy (Acc) against *DoS* for SPRP with the Laplace prior.

DoS (%)	30	50	60	70	76	80	90	96
Acc (%)	68.8	68.7	68.1	67.5	66.9	66.7	65.9	63.2

For the Jeffreys prior, there are no hyperparameters to tune. After learning, we get an accuracy of 68.1% with $DoS = 76\%$. Hence, with similar *DoS*, the Jeffreys prior can achieve slightly higher accuracy than the Laplace prior. From Table 5.1, we also find that a relatively good tradeoff between the *DoS* and accuracy can be obtained if $70\% < DoS < 80\%$. Hence, we can say that the Jeffreys prior can *adaptively* learn a good *DoS*. Due to this nice property, we only report the results of SPRP with the Jeffreys prior in the rest of this chapter. For fair comparison, we also use the Jeffreys prior for SPP (Archambeau & Bach, 2008).

5.3.3 Interpretability

For all the projection methods, we set the dimensionality of the latent space to 50. For Cora-IR, we adopt the original content features because we need to use the selected words for illustration. For all the other data sets, we use the expanded content features.

The *DoS* comparison of PCA, SPP, PRPCA and SPRP is shown in Table 5.2. We can see that the *DoS* of both PCA and PRPCA on WebKB and Cora-IR is either 0 or close to 0, which means that all the original variables (i.e., words) will be used to compute the principal components for PCA and PRPCA. For Cora, there exist some features (words) that no instances (papers) contain them. That is to say, all entries in the corresponding rows of the content matrix $\mathbf{T}$ will be zero. We also find that the zeroes in $\mathbf{W}$ are from those rows corresponding to the all-zero rows in $\mathbf{T}$. Hence, we can say that on Cora, PCA and PRPCA cannot learn sparse projection matrices either. Due to this non-sparse property, the results of PCA and PRPCA lack interpretability. On the contrary, both SPP and SPRP can learn sparse projection matrices. Compared with SPP, SPRP achieves lower *DoS*. However, the discrimination ability of SPRP is much higher than SPP, as to be shown later in Section 5.3.4.

Table 5.2: *DoS* (in %) comparison of PCA, SPP, PRPCA and SPRP.

	PCA	SPP	PRPCA	SPRP
Cora-IR	0	90	0	72
DS	18	88	20	76
HA	16	86	18	72
ML	10	90	12	76
PL	11	90	13	76
Cornell	0	74	0	48
Texas	0	77	0	42
Washington	1	74	1	47
Wisconsin	0	75	0	47

To further compare the results of SPP and SPRP in terms of interpretability, we show some details of the first six columns of **W** in Table 5.3. In the table, the 'Selected Words', arranged in descending order in terms of their **W** values, correspond to the top 10 nonzero entries in **W**. It is easy to see that the learned projection matrix of either SPRP or SPP does show some discrimination ability. More specifically, W_{*1} mainly corresponds to the class *retrieval*, W_{*2} to *filtering*, W_{*3} and W_{*4} to *extraction*, and W_{*5} and W_{*6} to *digital_library*. This is very promising because we can use the magnitude of the corresponding principal components to measure the class proportions of each instance. Detailed comparison between the words selected by SPP and SPRP shows that the words selected by SPRP is more discriminative than those selected by SPP. For example, 'dictionary' is more related to *retrieval* than 'agents', and 'symbolic' and 'wrapper' are more related to *extraction* than 'multi' and 'empirical'.

Table 5.3: Some details of the projection matrices learned by SPRP and SPP.

		Selected Words (arranged in descending order in terms of their **W** values)
SPRP	W_{*1}	information; retrieval; cross; language; extraction; system; evaluation; techniques; dictionary; incremental
	W_{*2}	text; categorization; learning; classification; information; feature; retrieval; selection; classifiers; algorithm
	W_{*3}	web; wide; learning; world; information; extract; symbolic; aid; formatting; extraction
	W_{*4}	extraction; information; learning; structured; rules; wrapper; documents; induction; grammatical; machine
	W_{*5}	text; language; digital; wide; world; high; processing; structured; sources; information
	W_{*6}	digital; library; learning; libraries; image; market; services; decoding; stanford; metadata
SPP	W_{*1}	information; retrieval; agents; evaluation; system; cross; language; intelligent; dissemination; distributed
	W_{*2}	text; categorization; learning; classification; information; feature; selection; extraction; study; case
	W_{*3}	web; wide; world; learning; information; search; multi; patterns; server; performance
	W_{*4}	extraction; information; learning; rules; disclosure; automatically; structured; basis; dictionary; empirical
	W_{*5}	text; digital; world; wide; information; system; library; categorization; processing; high
	W_{*6}	digital; library; learning; services; libraries; market; video; access; navigating; agents

For SPRP, 116 out of 787 words are not used to construct any principal component, which means that the entire rows of **W** corresponding to those words are zero. Hence, SPRP can also be used to perform feature elimination, which will speed up the collection process for new data. For example, some eliminated words include ‘wwww’, ‘aboutness’, ‘uu’, ‘erol’, ‘stylistic’, ‘encounter’, ‘classificatin’, ‘hypercode’, ‘broswer’, ‘lacking’, ‘multi-spectral’, and ‘exchanges’. It is interesting to note that most of them are typos or not related to information retrieval at all.

5.3.4 Accuracy

As in PRPCA, we adopt 5-fold cross validation to evaluate the accuracy. The dimensionality of the latent space is set to 50 for all the dimensionality reduction methods. After dimensionality reduction, a linear SVM is trained for classification based on the low-dimensional representation. The average classification accuracy for 5-fold cross validation, together with the standard deviation, is used as the performance metric.

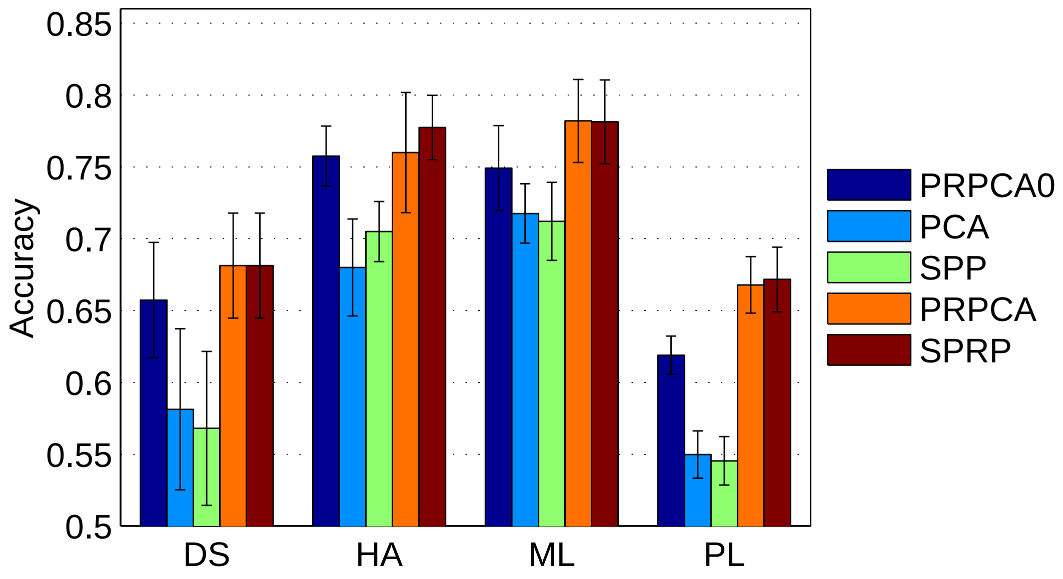


Figure 5.2: Results in average classification accuracy with standard deviation on the Cora data set.

The results on Cora and WebKB are shown in Figure 5.2 and Figure 5.3, respectively. PRPCA based on the original content features is denoted as PRPCA0, which achieves performance comparable with the state-of-the-art methods (Li et al., 2009a). All the other methods are based on the expanded content features. Compared with PCA, the higher accuracy of PRPCA0 shows that it is not good enough to just extract the extra information from the links and still assume the instances to be i.i.d. Comparison between PRPCA and PRPCA0 shows that slightly better performance can be achieved with the expanded content features, particularly for the Cora data set. Comparison between PRPCA and PCA verifies the claim in Chapter 4 that PRPCA dramatically outperforms PCA by eliminating the i.i.d. assumption. Comparison between SPP and PCA shows that the

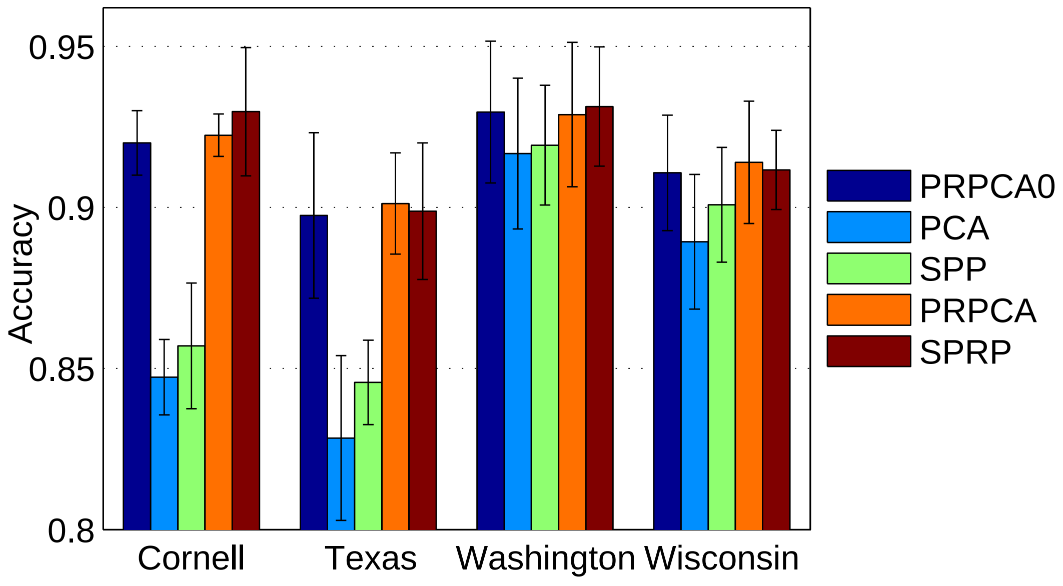


Figure 5.3: Results in average classification accuracy with standard deviation on the WebKB data set.

sparsity pursuit does not necessarily deteriorate the accuracy for the case with the i.i.d. assumption. Comparison between SPRP and SPP shows that under the sparsity pursuit case, dramatic accuracy improvement can also be achieved by explicitly modeling the covariance between instances, which once again verifies that the i.i.d. assumption is unreasonable for relational data. Finally, comparison between SPRP and PRPCA shows that under the PRPCA framework, we can also achieve sparsity without compromising accuracy.

5.3.5 Projection Cost

When the projection matrix learned is used to perform projection, the sparsity of SPRP will make its projection cost much lower than that of PRPCA. Table 5.4 reports the projection time needed to perform projection on the Cora and WebKB data sets. The test is performed with MATLAB implementation on a 2.33GHz personal computer. We can see that SPRP is much faster than PRPCA for the projection operation.

5.4 Conclusion

We have proposed a sparse version of PRPCA, called SPRP, to learn a sparse projection matrix for relational dimensionality reduction. Compared with PRPCA, the sparsity in SPRP not only makes its results interpretable, but it also makes the projection operation much more efficient without compromising its accuracy. Furthermore, compared with

Table 5.4: Projection time (in seconds) comparison of PRPCA and SPRP.

	PRPCA	SPRP
DS	2.431	0.749
HA	0.834	0.284
ML	8.225	2.272
PL	7.507	2.123
Cornell	2.362	1.241
Texas	2.273	1.323
Washington	3.588	1.946
Wisconsin	3.778	2.029

traditional sparse projection methods based on the i.i.d. assumption, SPRP can learn a more discriminative projection by explicitly modeling the covariance between instances.

CHAPTER 6

LATENT WISHART PROCESSES

6.1 Introduction

Kernel methods, such as SVM and Gaussian process (GP) (Rasmussen & Williams, 2006), have been widely used in many applications giving very promising performance. In kernel methods, the similarity between instances is represented by a kernel function defined over the input attributes. In general, the choice of an appropriate kernel function and its corresponding parameters is difficult in practice. Poorly chosen kernel functions can impair the performance significantly. Hence, kernel learning (Lanckriet et al., 2004; Zhang et al., 2006), which tries to find a good kernel matrix for the training data, is very important for kernel-based classifier design.

In many relational data, relationships (i.e., links) between (some) instances may also be available in the data in addition to the input attributes. The relational information often provides very strong hints to refine the correlation (or similarity) between instances. This motivates our work on *relational kernel learning* (RKL), which refers to learning a kernel matrix (or a kernel function) for relational data by incorporating relational information into the learning process.

Most existing SRL methods, such as HCI methods, PRMs and PLMs, are not based on the kernel approach. More recently, relational Gaussian process (RGP) (Chu et al., 2007) and mixed graph Gaussian process (XGP) (Silva et al., 2008) were proposed to model relational data from a kernel point of view. Both of them utilize relational information to learn the covariance (or kernel) matrix for a Gaussian process. Hence, RGP and XGP are relational kernel learning methods and we will follow their path in this chapter.

We propose a novel model, called LWP hereafter, based on *latent Wishart processes* to learn the kernel for relational data by seamlessly integrating relational information with input attributes. Several promising properties of LWP are highlighted here:

- LWP adopts the kernel for input attributes to define the prior for the target kernel, and use the link information to define the likelihood. Finally, MAP estimation is applied to learn the target kernel. Hence, LWP seamlessly integrates the two views into a principled framework.

- To the best of our knowledge, LWP is the first model that employs Wishart processes for relational learning.
- Unlike many other existing models, such as XGP, which are only suitable for the transductive setting, LWP is naturally applicable for inductive inference over unseen test data.
- During the kernel learning phase, *no label information* for the instances is needed, which makes it easy to extend LWP for visualization and clustering of relational data.

6.2 Wishart Processes

Definition 6.1 (Gupta & Nagar, 2000) *An $N \times N$ random symmetric positive definite matrix $\mathbf{C}$ is said to have a* ***Wishart distribution*** *with parameters N, q, and $N \times N$ scale matrix $\mathbf{\Sigma} \succ 0$, written as $\mathbf{C} \sim \mathcal{W}_N(q, \mathbf{\Sigma})$, if its p.d.f. is given by*

$$\frac{|\mathbf{C}|^{(q-N-1)/2}}{2^{qN/2}\Gamma_N(q/2)|\mathbf{\Sigma}|^{q/2}} \exp\Big(-\frac{1}{2}\mathrm{tr}(\mathbf{\Sigma}^{-1}\mathbf{C})\Big), \; q \geq N. \tag{6.1}$$

Here $\mathbf{\Sigma} \succ 0$ means that $\mathbf{\Sigma}$ is positive definite (pd).

Assume that we are given an input space $\mathcal{T} = \{\mathbf{t}_1, \mathbf{t}_2, \ldots\}$.

Definition 6.2 (Zhang et al., 2006) *The kernel function $\{C(\mathbf{t}_i, \mathbf{t}_j); \mathbf{t}_i, \mathbf{t}_j \in \mathcal{T}\}$ is said to be a* ***Wishart process*** *(WP) if for any $N \in \mathbb{N}$ and $\{\mathbf{t}_1, \ldots, \mathbf{t}_N\} \subseteq \mathcal{T}$, the $N \times N$ random matrix $\mathbf{C} = [C(\mathbf{t}_i, \mathbf{t}_j)]_{i,j=1}^N$ follows a Wishart distribution.*

For $C : \mathcal{T} \times \mathcal{T} \to \mathbb{R}$, there exists a corresponding mapping (say B) from the input space $\mathcal{T}$ to a latent (feature) space (say $\mathcal{F} \subset \mathbb{R}^q$), i.e., a vector-valued function $B(\mathbf{t}) = (B_1(\mathbf{t}), \ldots, B_q(\mathbf{t}))^T$ such that $C(\mathbf{t}_i, \mathbf{t}_j) = B(\mathbf{t}_i)^T B(\mathbf{t}_j)$. (Zhang et al., 2006) proved that $C(\mathbf{t}_i, \mathbf{t}_j)$ is a *Wishart process* if and only if the $B_j(\mathbf{t})$ $(j = 1, \ldots, q)$ are q mutually independent *Gaussian processes*.

Although q is possibly infinite, we assume that it is finite here for simplicity. Denote $\mathbf{C} = [C(\mathbf{t}_i, \mathbf{t}_j)]_{i,j=1}^N \in \mathbb{R}^{N \times N}$ and $\mathbf{B} = [B(\mathbf{t}_1), \ldots, B(\mathbf{t}_N)]^T = [\mathbf{b}_1, \ldots, \mathbf{b}_N]^T \in \mathbb{R}^{N \times q}$. Then the $\mathbf{b}_i$ are the latent vectors, and $\mathbf{C} = \mathbf{B}\mathbf{B}^T$ is the linear kernel in the latent space but is a nonlinear kernel w.r.t. the input space.

Theorem 6.1 *Let $\mathbf{\Sigma}$ be an $N \times N$ positive definite matrix. Then $\mathbf{C}$ is distributed according to the Wishart distribution $\mathcal{W}_N(q, \mathbf{\Sigma})$ if and only if $\mathbf{B}$ is distributed according to the matrix-variate normal distribution $\mathcal{N}_{N,q}(\mathbf{0}, \mathbf{\Sigma} \otimes \mathbf{I}_q)$.*

Proof: If $q \geq N$, the theorem can be proven by combination of Theorem 3.2.2 and Theorem 3.3.3 in (Gupta & Nagar, 2000).

If $q < N$, $\mathbf{C}$ follows a singular Wishart distribution (Srivastava, 2003), and the corresponding $\mathbf{B}$ is distributed according to the singular matrix-variate normal distribution (Definition 2.4.1 in (Gupta & Nagar, 2000)). The proof is similar to the case of $q \geq N$. ■

Theorem 6.1 shows an interesting connection; namely, the degree of freedom q in the Wishart process is the dimensionality of the latent space $\mathcal{F}$.

6.3 Methodology

Although directed links are common in some data sets, they can be converted into undirected links in many cases. In this chapter, we only focus on modeling undirected links which represent symmetric (or reciprocal) relationships. Furthermore, in the real world the relationship between two nodes can be either "positive" or "negative", which means that the attributes of the connected nodes have positive or negative correlation, respectively. Here we only consider positive relationships, although it is straightforward to extend our proposed model to negative relationships. The extension of our model for directed links with negative relationships will be pursued in our future work.

As in RRMF and PRPCA, we use a $d \times N$ matrix $\mathbf{T}$ to denote the content matrix with $\mathbf{T}_{*n} = \mathbf{t}_n \in \mathbb{R}^d$. The $N \times N$ matrix $\mathbf{A}$ denotes the adjacency (link) matrix of the N instances. A_{ik} is a binary variable indicating the existence of a relationship (link) between instances i and k, namely,

$$A_{ik} = \begin{cases} 1 & \text{if there exists a link between } \mathbf{t}_i \text{ and } \mathbf{t}_k \\ 0 & \text{otherwise.} \end{cases}$$

Rather than considering the design of a kernel classifier, we focus our attention on the learning of the kernel function for any kernel classifier by utilizing the relational information, i.e., relational kernel learning.

Unlike conventional kernel learning methods (Lanckriet et al., 2004; Zhang et al., 2006) which use the class labels to learn the kernel matrix, the setting for the relational kernel learning in LWP does not use any instance label. LWP only uses the input feature vectors and the relational information $\mathbf{A}$ to learn the kernel. Thus, LWP is *unsupervised* in nature.

6.3.1 Model

The available information for RKL includes both the input attributes and the binary variables $\{A_{ik}\}$. The goal of RKL is to learn a target kernel function $C(\mathbf{t}_i, \mathbf{t}_k)$ which takes both the input attributes and the relational information into consideration. Let $C_{ik} = C(\mathbf{t}_i, \mathbf{t}_k)$. Then $\mathbf{C} = [C_{ik}]_{i,k=1}^N$ will be a positive semidefinite matrix. We now model $\mathbf{C}$ by a (singular) Wishart distribution $\mathcal{W}_N(q, \mathbf{\Sigma})$. This implies that $C(\mathbf{t}_i, \mathbf{t}_k)$ follows a Wishart process.

Let $K(\mathbf{t}_i, \mathbf{t}_k)$ be a kernel function just defined on the input attributes. For example, the linear kernel $K(\mathbf{t}_i, \mathbf{t}_k) = \mathbf{t}_i^T \mathbf{t}_k$ is such a kernel function. Similarly, $\mathbf{K} = [K(\mathbf{t}_i, \mathbf{t}_k)]_{i,k=1}^N$ is a positive semidefinite matrix. If we set $\mathbf{\Sigma} = \beta(\mathbf{K} + \lambda \mathbf{I}_N)$ with $\beta > 0$ and λ being typically a very small number to make $\mathbf{\Sigma} \succ 0$, we have

$$\mathbf{C} \sim \mathcal{W}_N(q, \beta(\mathbf{K} + \lambda \mathbf{I}_N)). \tag{6.2}$$

Consequently, the input attributes are successfully integrated into the target kernel function.

To incorporate the relational information for the learning of $C(\cdot, \cdot)$, we regard each A_{ik} as a Bernoulli variable, which is determined by the latent variable C_{ik} via a logistic link function S_{ik}. Given the C_{ik}, we further assume that the A_{ik} are independent. Moreover, we assume that the links are symmetric with no self loops, i.e., $A_{ik} = A_{ki}$ and $A_{ii} = 0$. Then, we have

$$\begin{aligned} p(\mathbf{A}|\mathbf{C}) &= \prod_{i \neq k} S_{ik}^{A_{ik}} (1 - S_{ik})^{1 - A_{ik}} \\ \text{with} \quad S_{ik} &= \frac{\exp(C_{ik}/2)}{1 + \exp(C_{ik}/2)}. \end{aligned} \tag{6.3}$$

To this end, both the input attributes and the relational information are seamlessly integrated into the same framework, in which the input attributes define the *scale matrix* of the prior for the distribution of the target kernel function and the relational information defines the likelihood computed based on the target kernel function. Then, learning algorithms such as MAP estimation can be used to learn the latent variables C_{ik}. After learning the model, the target kernel function $C(\mathbf{t}_i, \mathbf{t}_k)$ will take both the input attributes and the relational information into consideration. We can directly use this new kernel function to implement kernel classification methods such as SVM and GP-based classifiers. In our experiments, we apply $C(\mathbf{t}_i, \mathbf{t}_k)$ as a kernel function for a GP-based classifier.

In this model, the Wishart process $C(\mathbf{t}_i, \mathbf{t}_k)$ is used to define latent variables $\{C_{ik}\}_{i,k=1}^N$. We thus call this model latent Wishart process (LWP) model.

6.3.2 Learning

It is natural to consider the MAP estimate of $\mathbf{C}$, namely,

$$\underset{\mathbf{C}}{\operatorname{argmax}} \ \log \big[p(\mathbf{A}|\mathbf{C})p(\mathbf{C})\big].$$

Theorem 6.1 shows that finding the MAP estimate of $\mathbf{C}$ is equivalent to finding the MAP estimate of $\mathbf{B}$. In particular, we note that for the applications presented in Section 6.5, small values of q, typically no larger than 50, are sufficient for delivering good performance. This motivates us to alternatively find the MAP estimate of $\mathbf{B}$, which will dramatically reduce the computation cost. Consequently, we attempt to maximize the following log posterior probability:

$$\begin{aligned}
L(\mathbf{B}) &= \log\{p(\mathbf{A}|\mathbf{B})p(\mathbf{B})\} \\
&= \sum_{i\neq k} \log p(A_{ik}|\mathbf{B}_{i*},\mathbf{B}_{k*}) + \log p(\mathbf{B}) \\
&= \sum_{i\neq k} \Big[A_{ik}\mathbf{B}_{i*}\mathbf{B}_{k}^{T}/2 - \log(1+\exp(\mathbf{B}_{i*}\mathbf{B}_{k*}^{T}/2))\Big] - \frac{1}{2}\mathrm{tr}\big[\frac{(\mathbf{K}+\lambda\mathbf{I}_N)^{-1}}{\beta}\mathbf{B}\mathbf{B}^{T}\big] + C \\
&= \sum_{i\neq k} \Big[A_{ik}\mathbf{B}_{i*}\mathbf{B}_{k*}^{T}/2 - \log(1+\exp(\mathbf{B}_{i*}\mathbf{B}_{k*}^{T}/2))\Big] - \frac{1}{2}\sum_{i,k}\sigma_{ik}\mathbf{B}_{i*}\mathbf{B}_{k*}^{T} + C,
\end{aligned}$$

where $[\sigma_{ik}]_{i,k=1}^{N} = \frac{(\mathbf{K}+\lambda\mathbf{I}_N)^{-1}}{\beta}$ and C is a constant independent of $\mathbf{B}$. We employ a block quasi-Newton method to solve the maximization of $L(\mathbf{B})$ w.r.t. $\mathbf{B}$. The basic idea is to approximate the Hessian matrix $\frac{\partial^2 L}{\partial\mathbf{B}\partial\mathbf{B}^T}$ by using the block diagonal matrix:

$$\frac{\partial^2 L}{\partial\mathbf{B}\partial\mathbf{B}^T} = \text{Block diag}\Big(\frac{\partial^2 L}{\partial\mathbf{B}_{1*}^T\partial\mathbf{B}_{1*}},\ldots,\frac{\partial^2 L}{\partial\mathbf{B}_{N*}^T\partial\mathbf{B}_{N*}}\Big).$$

The Fisher score vector and Hessian matrix of L w.r.t. $\mathbf{B}_{i*}^T$ are given by

$$\begin{aligned}
\frac{\partial L}{\partial\mathbf{B}_{i*}^T} &= \sum_{j\neq i}(A_{ij}-S_{ij}-\sigma_{ij})\mathbf{B}_{j*}^T - \sigma_{ii}\mathbf{B}_{i*}^T \\
\frac{\partial^2 L}{\partial\mathbf{B}_{i*}^T\partial\mathbf{B}_{i*}} &= -\frac{1}{2}\sum_{j\neq i} S_{ij}(1-S_{ij})\mathbf{B}_{j*}^T\mathbf{B}_{j*} - \sigma_{ii}\mathbf{I}_q \ \triangleq\ -\mathbf{H}_i.
\end{aligned}$$

Given the initial values $\mathbf{B}_{i*}(0)$, the update equations for the $\mathbf{B}_{i*}$ are

$$\mathbf{B}_{i*}^T(t{+}1){=}\mathbf{B}_{i*}^T(t) + \gamma\cdot\mathbf{H}_i(t)^{-1}\frac{\partial L}{\partial\mathbf{B}_{i*}^T}\bigg|_{\mathbf{B}=\mathbf{B}(t)}, \quad i=1,\ldots,N, \tag{6.4}$$

where γ is the step size. A strategy to make the objective function monotonically increase is to learn γ in each update step, but to search for a suitable γ in each update step might incur high computation cost. In our experiment, we find that if γ is simply set to a value less than 0.1, our algorithm works very well for all the experiments. Although in this case the objective function does not necessarily increase monotonically, the long-term trend of it is increasing. This makes our algorithm not only very fast but also very accurate.

Note that this iterative procedure works in a parallel manner. It may also work sequentially. However, our experiments show that the parallel scheme is more stable than the sequential scheme. As we have mentioned, the size of $\mathbf{H}_i$ is $q \times q$ with q being always a small number in our experiments, so the iterative procedure is very efficient. We adopt KPCA (Schölkopf et al., 1998) to initialize the values $\mathbf{B}_{i*}(0)$.

6.3.3 Out-of-Sample Extension

It is easy for LWP to perform out-of-sample extension (or called inductive inference), which means that we can use the learned LWP to predict the $\mathbf{B}_{i*}$ for new test data.

Let $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$ and $\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}$, where $\mathbf{A}_{11}(\boldsymbol{\Sigma}_{11})$ is an $n_1 \times n_1$ matrix and $\mathbf{A}_{22}(\boldsymbol{\Sigma}_{22})$ is an $n_2 \times n_2$ matrix. Suppose the n_1 instances corresponding to $\mathbf{A}_{11}$ and $\boldsymbol{\Sigma}_{11}$ are training data, and $\mathbf{A}_{22}$ and $\boldsymbol{\Sigma}_{22}$ correspond to new test data. Similarly, we partition $\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{bmatrix}$, $\mathbf{C} = \begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{B}_1\mathbf{B}_1^T & \mathbf{B}_1\mathbf{B}_2^T \\ \mathbf{B}_2\mathbf{B}_1^T & \mathbf{B}_2\mathbf{B}_2^T \end{bmatrix}$. Because $\mathbf{B} \sim \mathcal{N}_{N,q}(\mathbf{0},\ \boldsymbol{\Sigma} \otimes \mathbf{I}_q)$, we have $\mathbf{B}_1 \sim \mathcal{N}_{n_1,q}(\mathbf{0},\ \boldsymbol{\Sigma}_{11} \otimes \mathbf{I}_q)$ and

$$\mathbf{B}_2 \,|\, \mathbf{B}_1 \sim \mathcal{N}_{n_2,q}\big(\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\mathbf{B}_1,\ \boldsymbol{\Sigma}_{22\cdot1} \otimes \mathbf{I}_q\big), \tag{6.5}$$

where $\boldsymbol{\Sigma}_{22\cdot1} = \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}$.

For inductive inference, we first find the MAP estimate of $\mathbf{B}_1$ based on

$$\underset{\mathbf{B}_1}{\operatorname{argmax}} \log p(\mathbf{A}_{11}|\mathbf{B}_1)p(\mathbf{B}_1),$$

and then adopt the conditional expectation (or conditional mean) of $\mathbf{B}_2$ given $\mathbf{B}_1$ to estimate $\mathbf{B}_2$, which is given by $\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\mathbf{B}_1$ based on (6.5).

6.3.4 Complexity Analysis

Since $\mathbf{H}_i$ is of size $q \times q$, the computational complexity to invert it is $O(q^3)$. Typically, q is a very small number. Hence, to update the whole $\mathbf{B}$, only $O(N)$ time is needed.

The most costly operation is to compute $(\mathbf{K} + \lambda\mathbf{I}_N)^{-1}$, which typically has complexity of $O(N^3)$. However, if a linear kernel is adopted, i.e., $\mathbf{K} = \mathbf{T}^T\mathbf{T}$, we can use Woodbury identity to speed up the computation. That is, $(\mathbf{T}^T\mathbf{T} + \lambda\mathbf{I}_N)^{-1} = \frac{1}{\lambda}\mathbf{I}_N - \frac{1}{\lambda}\mathbf{T}^T(\lambda\mathbf{I}_d + \mathbf{T}\mathbf{T}^T)^{-1}\mathbf{T}$. Then the time complexity will be $O(N^2)$.

Hence, if $\mathbf{K}$ is nonlinear, the overall complexity is $O(N^3)$. Otherwise, it is $O(N^2)$.

6.4 Related Work

The methods that are most related to our LWP model are RGP (Chu et al., 2007) and XGP (Silva et al., 2008). Both RGP and XGP also try to learn the covariance (kernel) matrix for Gaussian processes by exploiting the relationships between instances. Unlike LWP which is to learn multiple (q) GPs, RGP and XGP only learn one GP. In fact, when $q = 1$, $\mathbf{B}$ ($N\times 1$) in LWP degenerates to a single Gaussian process. Hence, our model can be regarded as a generalization of RGP and XGP. The *key difference* between them is that LWP treats $\mathbf{C} = \mathbf{B}\mathbf{B}^T$ as the learned kernel matrix, which can be further used to train all kinds of kernel classifiers. In RGP and XGP, however, $p(\mathbf{B}|\mathbf{A})$ is itself a prediction function with $\mathbf{B}$ being a vector of function values for all input points. The learned kernel, which is the *covariance matrix* of the posterior distribution $p(\mathbf{B}|\mathbf{A})$, is $(\mathbf{K}^{-1} + \mathbf{\Pi}^{-1})^{-1}$ in RGP and $(\mathbf{K} + \mathbf{\Pi})$ in XGP, where $\mathbf{\Pi}$ is a kernel matrix capturing the link information. Since there is no closed-form solution for $p(\mathbf{B}|\mathbf{A})$, RGP and XGP adopt different approximation strategies to compute the posterior covariance matrix.

Another related work is the stochastic relational model in (Yu et al., 2006; Yu & Chu, 2007), where $\mathbf{C}$ is modeled as a tensor GP rather than a WP. Since $\mathbf{C}$ in (Yu & Chu, 2007; Yu et al., 2006) is not guaranteed to be positive semi-definite, it cannot be regarded as a kernel matrix. Furthermore, the focus of (Yu et al., 2006; Yu & Chu, 2007) is on linkage prediction rather than instance classification as in our LWP model.

Our work has also been motivated by the latent space approaches in social network analysis (Hoff et al., 2002). Let $d_{ij}^2 = \|\mathbf{B}_{i*}^T - \mathbf{B}_{j*}^T\|^2$ be the squared distance between $\mathbf{B}_{i*}^T$ and $\mathbf{B}_{j*}^T$, and $\mathbf{D} = [d_{ij}^2]$ be the $N\times N$ distance matrix. Then $-\frac{1}{2}\mathbf{E}\mathbf{D}\mathbf{E} = \mathbf{E}\mathbf{C}\mathbf{E}$ where $\mathbf{E}$ is an $N\times N$ centering matrix. This reveals a connection between our model and the latent distance model in (Hoff, 2007). In addition, we also note that in the latent eigenmodel (Hoff, 2007) for symmetric relational data, (Hoff, 2007) defined $\mathbf{C} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ where $\mathbf{U}$ is an orthogonal matrix and $\mathbf{\Lambda}$ is a diagonal matrix but its diagonal elements can be negative. Thus, $\mathbf{C}$ does not play the role of a kernel matrix.

6.5 Experiments

We compare our LWP method with several related methods, such as the standard GP classifier (GPC) (Rasmussen & Williams, 2006), RGP and XGP, on three real-world data sets. The first two are subsets of WebKB (Craven et al., 1998) and Cora (McCallum et al., 2000) used in XGP. Note that these two subsets are different from those used for RRMF and PRPCA. The third data set is the PoliticalBook data set used in PRPCA and XGP. The centralized linear kernel $K(\mathbf{t}_i, \mathbf{t}_j) = \mathbf{t}_i^T\mathbf{t}_j$ (Chu et al., 2007) is used to define the covariance matrix $\mathbf{K}$ in the Wishart distribution defined in (6.2) for all these data sets. The λ in (6.2) is set to a small number 10^{-4}. All the $\mathbf{B}_{i*}$ are *initialized* to the feature vectors obtained by KPCA [1] (Schölkopf et al., 1998) based on $\mathbf{K} + \lambda\mathbf{I}_N$.

Although our method can be applied under the inductive setting, for fair comparison we run our method under the transductive setting because both RGP and XGP were only tested under this setting (Silva et al., 2008).[2] More specifically, for our LWP method, we first perform kernel learning based on all the points, including training and test points, and their links, but *without using any label information.* Then, based on the learned kernel, we learn a GP with the training points only and evaluate the learned model on the test points. Hence, the main difference between our method and other methods is just in the kernel learning part.

6.5.1 Sensitivity to Parameters

There are four parameters in total which will affect the training of LWP. They are the dimensionality of the latent space q, the β in (6.2), the γ in (6.4), and the number of iterations (T) to optimize $L(\mathbf{B})$. We find that the performance is very stable by setting $1000 \leq \beta \leq 10000$. Hence, in all the following experiments, $\beta = 1000$.

We first study the effect of γ and T. Here we use the Texas subset of the WebKB data set to illustrate this. The description of this subset is given in Section 6.5.3. We find that when $\gamma \leq 0.01$, our algorithm is very stable. The performance, measured in area under the ROC curve (AUC), and the objective function against the change in T are illustrated in Figure 6.1, in which the X-axis denotes T, "AUC01" is the performance with $\gamma = 0.01$, "AUC001" is the performance with $\gamma = 0.001$, "obj01" is the objective function values with $\gamma = 0.01$ and "obj001" is the objective function values with $\gamma = 0.001$. Note that the objective function values in the figure are transformed to $L(\mathbf{B})/10^5 + 4$ for the convenience of demonstration. We can see that the long-term trend of the objective function is increasing, and the performance of our algorithm is very stable. For $\gamma = 0.01$, 10 iterations are enough to give good performance.

We also test the sensitivity of our method to the change in q on the Texas subset and

[1]The KPCA here is actually PCA, because for text processing the linear kernel always outperforms other kernels. Calling it KPCA is just for the consistency of our algorithm, because during the derivation of our algorithm K can be any kind of kernel.

[2]In fact, XGP can only work under the transductive setting.

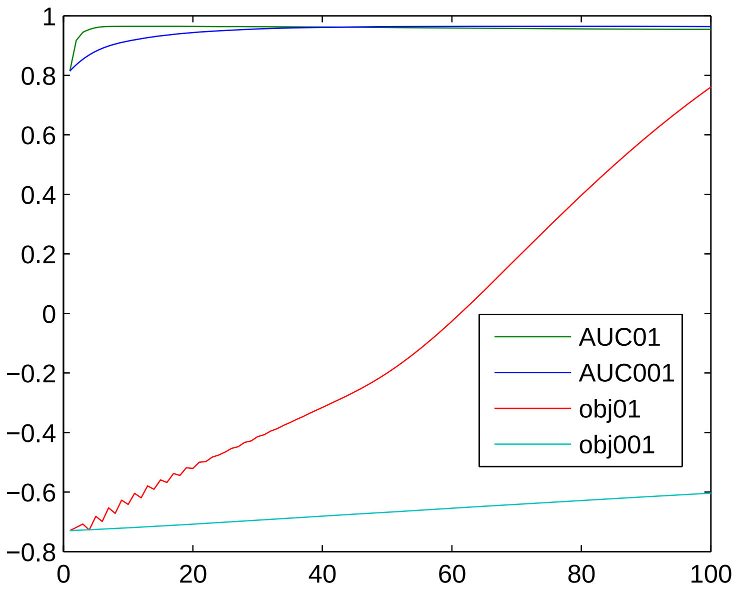


Figure 6.1: The performance (AUC) and the objective function against the change in the number of iterations.

the political books data set. Figure 6.2 shows the average AUC with standard deviation over 100 trials of our method when q is set to different values. Note that we use KPCA to initialize the $\mathbf{B}_{i*}$s. Hence, KPCA actually refers to a GPC with the kernel matrix computed based on the *initial values* of $\mathbf{B}_{i*}$, i.e., $\mathbf{B}_{i*}(0)$, in LWP. This means KPCA corresponds to the case that the iteration number in LWP is set to 0. From Figure 6.2, we can see that LWP is very robust to the change in q. Furthermore, by comparing LWP with KPCA [3], it is not difficult to see that the learning method in LWP is very effective. Note that the performance of GPC on the Texas subset is 0.799 ± 0.021, and that on the political books data set is 0.92.

We can see that LWP is robust to parameter changes. In all our following experiments, unless otherwise stated, we just set $\beta = 1000, \gamma = 0.01, T = 10, q = 20$.

6.5.2 Visualization

The learned $\mathbf{B}_{i*}$ can be treated as the feature representation in a latent space, which can be utilized for data visualization. Here we use the Texas subset of the WebKB data set to illustrate this. We use the whole data set (827 examples with their links) without any label information to perform kernel learning with our LWP model. For the sake of

[3]The comparison between LWP and KPCA is just used to demonstrate that the good performance of LWP is not from a good initial value but from the learning process. We denote the initial values as KPCA just because we use KPCA for initialization.

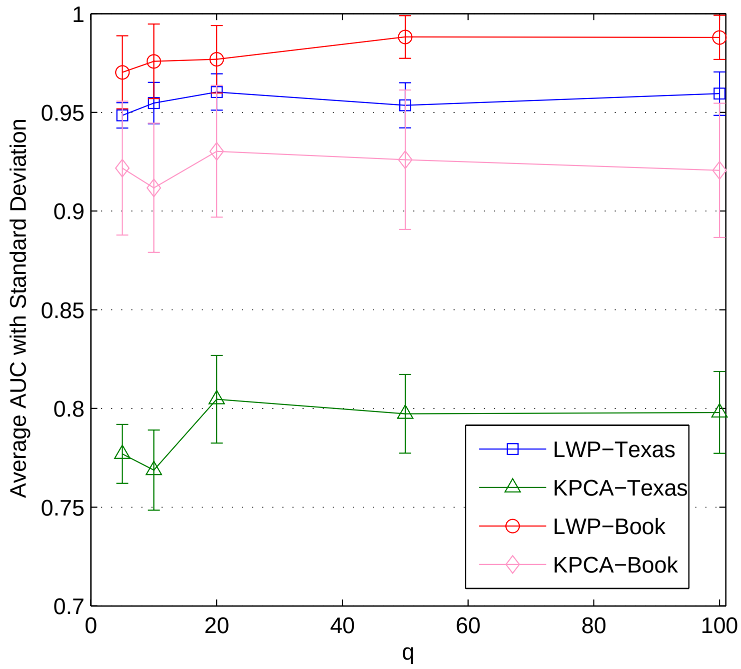
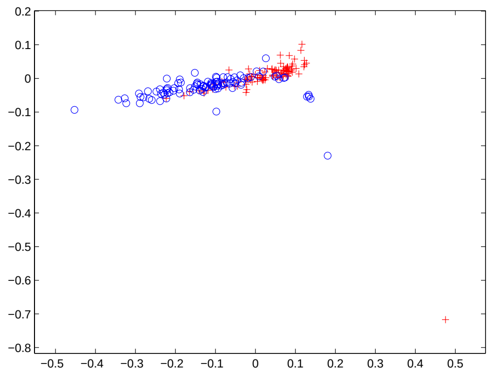


Figure 6.2: Performance of LWP against the change in degree of freedom q in the Wishart process (i.e., dimensionality of the latent space).

visualization, q is set to 2. After learning, 100 positive and 100 negative examples are randomly chosen for visualization. The results are demonstrated in Figure 6.3, where KPCA refers to the initial values of $\mathbf{B}_{i*}$, i.e., $\mathbf{B}_{i*}(0)$. We can see that different classes are well separated in the learned latent space by LWP, although the initialization by KPCA is very poor. Good classification performance can be expected when the examples are classified in this latent space, which will be verified by our subsequent experiment. Furthermore, because no label information is used to learn this feature representation, good clustering performance can also be expected in this learned space.

6.5.3 Performance on WebKB Data Set

The original data set contains 4,160 pages and 9,998 hyperlinks. We adopt the same strategy as that in (Chu et al., 2007; Silva et al., 2008) to translate the hyperlinks into 66,249 undirected linkages over the pages by assuming that two pages are likely to be positively correlated if they are hyperlinked by the same hub page. Each webpage is represented as bag-of-words, a vector of "term frequency" components scaled by the "inverse document frequency", and then normalized to unit length. This preprocessing step is the same as that in (Chu et al., 2007). The task is to classify each webpage into two classes: "other" and "non-other". The same performance measure (AUC) and the same evaluation strategy as those in (Silva et al., 2008) are adopted for all the methods, i.e., for a specific


(a) KPCA

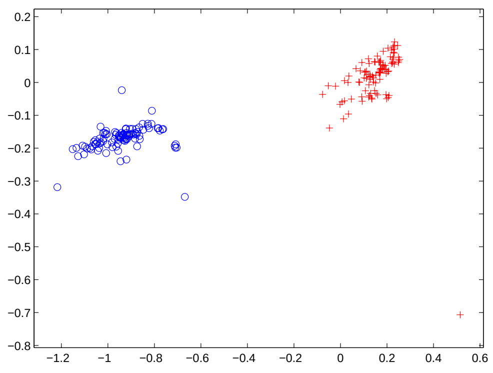

(b) LWP

Figure 6.3: Visualization of data points in the transformed space by KPCA and in the latent space learned by LWP. 100 positive (red cross) and 100 negative (blue circle) examples are shown.

university, the same 100 subsamples as those in (Chu et al., 2007; Silva et al., 2008) are used, in each of which 10% of the data points are randomly chosen for training and the rest for testing.

The average AUC with standard deviation over 100 trials is reported in Table 6.1, from which we can find that LWP achieves performance at least comparable with the state of the art for all four universities. In particular, compared with GPC and RGP, LWP achieves far better results.

Table 6.1: Mean and standard deviation of AUC over 100 rounds of test on the WebKB data set. All the methods are based on the same data partitions for both training and testing. #Other and #All refer to the numbers of positive examples and all examples, respectively. #Links is the number of links in the corresponding data set.

University	#Other/#All/#Links	GPC	RGP	XGP	LWP
Cornell	617 / 865 / 13177	0.708 ± 0.021	0.884 ± 0.025	0.917 ± 0.022	**0.932 ± 0.019**
Texas	571 / 827 / 16090	0.799 ± 0.021	0.906 ± 0.026	0.949 ± 0.015	**0.960 ± 0.009**
Washington	939 / 1205 / 15388	0.782 ± 0.023	0.877 ± 0.024	0.923 ± 0.016	**0.935 ± 0.010**
Wisconsin	942 / 1263 / 21594	0.839 ± 0.014	0.899 ± 0.015	**0.941 ± 0.018**	0.940 ± 0.012

6.5.4 Performance on Cora Data Set

The information about the subset from Cora is shown in the second column of Table 6.2. We adopt the same feature representation scheme as that in (Silva et al., 2008), where each instance is represented by a feature vector of dimensionality 20,082. For this data set, very good performance can be achieved even if q is set to as small as 1. Hence, we just set q to 1 for this data set. For each class, 1% of the whole set is randomly selected for training and the rest for testing. One hundred rounds of such partitioning are repeated.

The average AUC with standard deviation is reported in Table 6.2, in which "GPC with Citation" is the method proposed in (Silva et al., 2008) by adding the citation adjacency matrix as a binary input feature for each paper. From the table, we can see that LWP is far better than related methods. In (Silva et al., 2008), the authors said that the AUC of RGP on this data set is close to 1. So it means that on this data set, LWP is also comparable with the state of the art.

Table 6.2: Mean and standard deviation of AUC over 100 rounds of test on the Cora data set. All the methods are based on the same data partitions for both training and testing. #Pos and #Neg refer to the numbers of positive examples and negative examples, respectively. #Citations is the number of links in the corresponding data set.

Group	#Pos/#Neg/#Citations	GPC	GPC with Citation	XGP	LWP
5vs1	346 / 488 / 2466	0.905 ± 0.031	0.891 ± 0.022	0.945 ± 0.053	**0.990 ± 0.000**
5vs2	346 / 619 / 3417	0.900 ± 0.032	0.905 ± 0.044	0.933 ± 0.059	**0.991 ± 0.001**
5vs3	346 / 1376 / 3905	0.863 ± 0.040	0.893 ± 0.017	0.883 ± 0.013	**0.986 ± 0.001**
5vs4	346 / 646 / 2858	0.916 ± 0.030	0.887 ± 0.018	0.951 ± 0.042	**0.997 ± 0.000**
5vs6	346 / 281 / 1968	0.887 ± 0.054	0.843 ± 0.076	0.955 ± 0.041	**0.998 ± 0.000**
5vs7	346 / 529 / 2948	0.869 ± 0.045	0.867 ± 0.041	0.926 ± 0.076	**0.992 ± 0.002**

6.5.5 Performance on PoliticalBook Data Set

The detailed information about PoliticalBook data set can be found in Section 4.4. We randomly choose half of the whole data for training and the rest for testing. This subsampling process is repeated for 100 rounds and the average AUC with its standard deviation is reported in Table 6.3 [4], from which we can see that LWP is comparable with the state of the art. Here, KPCA also refers to the method using the initial values of $\mathbf{B}_{i*}$s in LWP to perform classification.

Table 6.3: Experiment on the data set of political books.

GPC	RGP	XGP	KPCA	LWP
0.92	**0.98**	**0.98**	0.93 ± 0.03	**0.98 ± 0.02**

6.5.6 Discussions

From the above experiments, we can see that LWP achieves very promising performance on all the data sets tested, while RGP and XGP can only perform well on some of the data sets. More specifically, RGP performs quite well on the Cora data set but it

[4]The results for GPC, RGP and XGP are taken from (Silva et al., 2008), in which the standard deviation is not reported.

performs relatively badly on the WebKB data set. On the other hand, XGP achieves unsatisfactory performance on the Cora data set. In particular, compared with GPC which naively discards the relational information in the data, our method achieves far better results, implying that the relational information is indeed very informative and our LWP can exploit the information very effectively.

6.6 Conclusion

Relational information is very useful for specifying the similarity between different instances. We have presented a very effective LWP model for performing kernel learning by seamlessly integrating relational information with the input attributes. Besides the promising performance for instance classification, LWP can also be used for data visualization and clustering.

CHAPTER 7

GENERALIZED LATENT FACTOR MODELS

7.1 Introduction

A social network [1] (Wasserman & Katherine, 1994) is often represented as a graph in which the nodes represent the entities and the edges (or called links) represent the binary relations between entities. In this chapter, we assume that a graph contains only one type of entities, which is a typical assumption for SNA (Wasserman & Katherine, 1994). The edges in a graph can be directed or undirected. If the edges are directed, we call the graph a directed graph (or directed relation). Otherwise, the graph is an undirected graph (or undirected relation). Unless otherwise stated, we focus on directed graphs in this chapter because an undirected edge can be represented by two directed edges with opposite directions. Some typical networks include friendship networks among people, web graphs, and paper citation networks. As stated in Section 1.1.2, *homophily* and *stochastic equivalence* are two primary features of interest in social networks. The difference between homophily and stochastic equivalence is demonstrated in Figure 1.1.

As SNA is becoming more and more important in a wide range of applications, many SNA models have been proposed (Goldenberg et al., 2009). In this chapter, we focus on *latent variable models* (Bartholomew & Knott, 1999) which have been successfully applied to model social networks (Nowicki & Snijders, 2001; Hoff et al., 2002; Kemp et al., 2006; Hoff, 2007; Airoldi et al., 2008; Hoff, 2009). These models include: the latent class model (Nowicki & Snijders, 2001) and its extensions (Kemp et al., 2006; Airoldi et al., 2008), the latent distance model (Hoff et al., 2002), the latent eigenmodel (Hoff, 2007), and the multiplicative latent factor model (MLFM) (Hoff, 2009). Among all these models, the recently proposed latent eigenmodel, which includes both the latent class model and the latent distance model as special cases, can capture both homophily and stochastic equivalence in networks. However, it can only model *undirected* graphs. MLFM (Hoff, 2009) adapts the latent eigenmodel for *directed* graphs. However, as to be shown in our experiments, in fact it cannot model well homophily.

In this chapter, we propose a novel model, called *generalized latent factor model* (GLFM), for social network analysis by enhancing homophily modeling in MLFM. The learning algorithm of GLFM is guaranteed to converge to a local optimum and has linear-time

[1]In this chapter, we use the terms 'network', 'social network' and 'graph' interchangeably.

complexity. Hence, GLFM can be used to model large-scale graphs. Extensive experiments on community detection in some real-world networks show that GLFM dramatically outperforms existing methods.

Notations for this chapter GLFM is designed to model a directed relation (i.e., graph) defined on a set of instances of the same type. Let N denote the number of nodes in the graph. $\mathbf{A}$ is the adjacency (link) matrix for the N nodes. $A_{ij} = 1$ if there exists a link from node i to node j. Otherwise, $A_{ij} = 0$. q denotes the number of latent factors. In real-world networks, if $A_{ij} = 1$, we can say that there is a link from i to j. However, $A_{ij} = 0$ does not necessarily mean that there is no link from i to j. In most cases, $A_{ij} = 0$ means that the link from i to j is missing. Hence, we use an indicator matrix $\mathbf{Z}$ to indicate whether or not an element is missing. More specifically, $Z_{ij} = 1$ means that A_{ij} is observed while $Z_{ij} = 0$ means that A_{ij} is missing.

7.2 Multiplicative Latent Factor Models

The latent eigenmodel is formulated as follows [2]:

$$\Theta_{ik} = \log \text{odds}(A_{ik} = 1 \,|\, \mathbf{X}_{i*}, \mathbf{X}_{k*}, \mu) = \mu + \mathbf{X}_{i*}\mathbf{\Lambda}\mathbf{X}_{k*}^T,$$

where $\mathbf{X}$ is an $N \times q$ matrix with $\mathbf{X}_{i*}$ denoting the latent representation of node i and μ is a parameter reflecting the overall density of the links in the network, $\mathbf{\Lambda}$ is a $q \times q$ diagonal matrix with the diagonal entries being either positive or negative. The latent eigenmodel generalizes both latent class models and latent distance models. It can model both homophily and stochastic equivalence in *undirected* graphs.

To adapt the latent eigenmodel for *directed* graphs, MLFM defines

$$\Theta_{ik} = \mu + \mathbf{X}_{i*}\mathbf{\Lambda}\mathbf{W}_{k*}^T, \tag{7.1}$$

where $\mathbf{X}$ and $\mathbf{W}$ have orthonormal columns. Note that the key difference between the latent eigenmodel and MLFM lies in the fact that MLFM adopts a different *receiver* factor matrix $\mathbf{W}$ which enables MLFM to model directed (asymmetric) graphs. As we will show in our experiments, this modification in MLFM makes it fail to model homophily in networks.

Letting $\mathbf{\Theta} = [\Theta_{ik}]_{i,k=1}^N$, we can rewrite MLFM as follows:

$$\mathbf{\Theta} = \mu\mathbf{E} + \mathbf{X}\mathbf{\Lambda}\mathbf{W}^T, \tag{7.2}$$

[2]Note that in this chapter, we assume for simplicity that there is no attribute information for the links. It is straightforward to integrate attribute information into the existing latent variable models as well as our proposed model.

where $\mathbf{E}$ is an $N \times N$ matrix with all entries being 1.

We find that MLFM is a special case of the following model:

$$\mathbf{\Theta} = \mu\mathbf{E} + \mathbf{U}\mathbf{V}^T. \tag{7.3}$$

For example, we get MLFM by setting $\mathbf{U} = \mathbf{X}$ and $\mathbf{V} = \mathbf{W}\mathbf{\Lambda}$. Furthermore, it is easy to compute the $\mathbf{X}$, $\mathbf{W}$ and $\mathbf{\Lambda}$ in (7.2) based on the learned $\mathbf{U}$ and $\mathbf{V}$ in (7.3). Hence, in the sequel, MLFM refers to the model in (7.3).

7.3 Generalized Latent Factor Models

As discussed above, MLFM can capture stochastic equivalence but cannot model well homophily in directed graphs. Here, we propose our GLFM to enhance homophily modeling in MLFM.

7.3.1 Model

In GLFM, Θ_{ik} is defined as follows:

$$\Theta_{ik} = \mu + \frac{1}{2}\mathbf{U}_{i*}\mathbf{U}_{k*}^T + \frac{1}{2}\mathbf{U}_{i*}\mathbf{V}_{k*}^T. \tag{7.4}$$

Comparing (7.4) to (7.3), we can find that GLFM generalizes MLFM by adding an extra term $\mathbf{U}_{i*}\mathbf{U}_{k*}^T$.[3] It is this extra term that enables GLFM to model homophily in networks, which will be detailed in Section 7.3.2 when we analyze the objective function in (7.9). This will also be demonstrated empirically later in our experiments.

Based on (7.4), the likelihood of the observations can be defined as follows:

$$p(\mathbf{A} \mid \mathbf{U}, \mathbf{V}, \mu) = \prod_{i \neq k} [S_{ik}^{A_{ik}}(1 - S_{ik})^{1-A_{ik}}]^{Z_{ik}}, \tag{7.5}$$

where

$$S_{ik} = \frac{\exp(\Theta_{ik})}{1 + \exp(\Theta_{ik})}. \tag{7.6}$$

Note that as in the conventional SNA model, we ignore the diagonal elements of $\mathbf{A}$. That is, we set $A_{ii} = Z_{ii} = 0$ by default.

[3]Note that the coefficient $\frac{1}{2}$ in (7.4) makes no essential difference between (7.4) and (7.3). It is only for convenience of computation.

Furthermore, we put normal priors on the parameters μ, $\mathbf{U}$ and $\mathbf{V}$:

$$p(\mu) = \mathcal{N}(\mu \,|\, 0, \tau^{-1}),$$

$$p(\mathbf{U}) = \prod_{j=1}^{q} \mathcal{N}(\mathbf{U}_{*j} \,|\, \mathbf{0}, \beta \mathbf{I}_N), \tag{7.7}$$

$$p(\mathbf{V}) = \prod_{j=1}^{q} \mathcal{N}(\mathbf{V}_{*j} \,|\, \mathbf{0}, \gamma \mathbf{I}_N). \tag{7.8}$$

7.3.2 Learning

Although the Markov chain Monte Carlo (MCMC) algorithms designed for other latent variable models can easily be adapted for GLFM, we do not adopt MCMC here for GLFM because MCMC methods typically incur very high computational cost. We adopt the MAP estimation strategy to learn the parameters. The log posterior probability can be computed as follows:

$$\begin{aligned} L = & \sum_{i \neq k} \left\{ \frac{1}{2} A_{ik} \mathbf{U}_{i*} \mathbf{U}_{k*}^T + \frac{1}{2} A_{ik} \mathbf{U}_{i*} \mathbf{V}_{k*}^T + A_{ik}\mu - Z_{ik} \log \left[1 + \exp(\Theta_{ik}) \right] \right\} \\ & - \frac{1}{2\beta} \text{tr}(\mathbf{U}\mathbf{U}^T) - \frac{1}{2\gamma} \text{tr}(\mathbf{V}\mathbf{V}^T) - \frac{\tau}{2}\mu^2 + c, \end{aligned} \tag{7.9}$$

where c is a constant independent of the parameters. Note that in (7.9) we assume that all existing links should be observed. That is to say, if $A_{ik} = 1$, then $Z_{ik} = 1$.

The term $A_{ik}\mathbf{U}_{i*}\mathbf{U}_{k*}^T$ in (7.9) results from the extra term $\mathbf{U}_{i*}\mathbf{U}_{k*}^T$ in (7.4). In (7.9), to maximize the objective function L, we have to make $\mathbf{U}_{i*}\mathbf{U}_{k*}^T$ as large as possible if there exists a link between nodes i and k (i.e., $A_{ik} = 1$). This conforms to the property of homophily, i.e., a link is more likely to exist between two nodes with similar characteristics than between those nodes having different characteristics. Note that here the latent factor $\mathbf{U}_{i*}$ reflects the characteristics of node i. Therefore, the extra term $\mathbf{U}_{i*}\mathbf{U}_{k*}^T$ in (7.4) enables GLFM to model homophily in networks.

If we directly optimize all the parameters $\mathbf{U}$, $\mathbf{V}$ and μ jointly, the computational cost will be very high. For example, if we want to use second-order information, we generally need to invert the Hessian matrix where the time complexity is cubic in the number of parameters.

Here, we adopt an alternating projection strategy to maximize L. More specifically, each time we optimize one parameter, such as $\mathbf{U}$, with the other parameters fixed.

Learning U

To learn $\mathbf{U}$, we optimize each row of it with all other rows fixed. The gradient vector and Hessian matrix can be computed as follows:

$$\frac{\partial L}{\partial \mathbf{U}_{i*}^T} = -\frac{1}{\beta}\mathbf{U}_{i*}^T + \frac{1}{2}\mathbf{V}^T\Big[\mathbf{A}_{i*}^T - (\mathbf{Z}_{i*} \circ \mathbf{S}_{i*})^T\Big] + \frac{1}{2}\mathbf{U}^T\Big[\mathbf{A}_{i*}^T + \mathbf{A}_{*i} - (\mathbf{Z}_{i*} \circ \mathbf{S}_{i*})^T - \mathbf{Z}_{*i} \circ \mathbf{S}_{*i}\Big],$$

$$\begin{aligned}\frac{\partial^2 L}{\partial \mathbf{U}_{i*}^T \partial \mathbf{U}_{i*}} =& -\frac{1}{\beta}\mathbf{I}_q - \frac{1}{4}\sum_{k,k\neq i}\Big\{Z_{ik}S_{ik}(1-S_{ik})[\mathbf{U}_{k*}+\mathbf{V}_{k*}]^T[\mathbf{U}_{k*}+\mathbf{V}_{k*}]\Big\} \\ & -\frac{1}{4}\sum_{k,k\neq i}\Big\{Z_{ki}S_{ki}(1-S_{ki})\mathbf{U}_{k*}^T\mathbf{U}_{k*}\Big\}.\end{aligned}$$

Since both the gradient vector and Hessian matrix depend on $\mathbf{S}_{i*}$ which is a function of $\mathbf{U}_{i*}$, we have to resort to iterative methods to find the optimal values. Here, we devise a *minorization-maximization* (MM) algorithm (Lang et al., 2000) to learn it. MM is a so-called EM algorithm (Dempster et al., 1977) without missing data, alternating between constructing a concave lower bound of the objective function and maximizing that bound.

Since $0 < S_{ik} < \frac{1}{2}$, we can get $S_{ik}(1-S_{ik}) < \frac{1}{4}$.

Let us define:

$$\mathbf{H}_i = -\frac{1}{\beta}\mathbf{I}_q - \frac{1}{16}\sum_{k,k\neq i}\Big\{Z_{ik}[\mathbf{U}_{k*}+\mathbf{V}_{k*}]^T[\mathbf{U}_{k*}+\mathbf{V}_{k*}]\Big\} - \frac{1}{16}\sum_{k,k\neq i}\Big\{Z_{ki}\mathbf{U}_{k*}^T\mathbf{U}_{k*}\Big\}.$$

It is easy to prove that $\frac{\partial^2 L}{\partial \mathbf{U}_{i*}^T \partial \mathbf{U}_{i*}} \succeq \mathbf{H}_i$.

Let

$$f(\mathbf{U}_{i*}) = L(\mathbf{U}_{i*}(t)) + [\mathbf{U}_{i*} - \mathbf{U}_{i*}(t)] \times \frac{\partial L}{\partial \mathbf{U}_{i*}^T}(t) + \frac{1}{2}[\mathbf{U}_{i*} - \mathbf{U}_{i*}(t)]\mathbf{H}_i(t)[\mathbf{U}_{i*} - \mathbf{U}_{i*}(t)]^T,$$

where $\mathbf{U}_{i*}(t)$ denotes the value of the former iteration and $\mathbf{H}_i(t)$ is computed with the updated $\mathbf{U}$ except for $\mathbf{U}_{i*}$.

Then we have the following theorem:

Theorem 7.1 *$L(\mathbf{U}_{i*}) \geq f(\mathbf{U}_{i*})$, which means that $f(\mathbf{U}_{i*})$ is a lower bound of $L(\mathbf{U}_{i*})$.*

The proof of Theorem 7.1 is simple and we omit it here.

We can see that $f(\mathbf{U}_{i*})$ has a quadratic form of $\mathbf{U}_{i*}$. By setting the gradient of $f(\mathbf{U}_{i*})$ with respect to $\mathbf{U}_{i*}$ to 0, we have the update rule for $\mathbf{U}_{i*}$:

$$\mathbf{U}_{i*}(t+1) = \mathbf{U}_{i*}(t) - \Big[\frac{\partial L}{\partial \mathbf{U}_{i*}^T}(t)\Big]^T \times \mathbf{H}_i(t)^{-1}. \tag{7.10}$$

Learning $\mathbf{V}$

The gradient vector and Hessian matrix of $\mathbf{V}_{i*}$ can be computed as follows:

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{V}_{i*}^T} &= -\frac{1}{\gamma}\mathbf{V}_{i*}^T + \frac{1}{2}\mathbf{U}^T\Big[\mathbf{A}_{*i} - (\mathbf{Z}_{*i} \circ \mathbf{S}_{*i})\Big] \\ \frac{\partial^2 L}{\partial \mathbf{V}_{i*}^T \partial \mathbf{V}_{i*}} &= -\frac{1}{\gamma}\mathbf{I}_q - \frac{1}{4}\sum_{k,k\neq i}\Big\{Z_{ki}S_{ki}(1-S_{ki})\mathbf{U}_{k*}^T\mathbf{U}_{k*}\Big\}. \end{aligned} \tag{7.11}$$

Let

$$\mathbf{G}_i = -\frac{1}{\gamma}\mathbf{I}_q - \frac{1}{16}\sum_{k,k\neq i}\Big\{Z_{ki}\mathbf{U}_{k*}^T\mathbf{U}_{k*}\Big\}, \tag{7.12}$$

we can prove that $\frac{\partial^2 L}{\partial \mathbf{V}_{i*}^T \partial \mathbf{V}_{i*}} \succeq \mathbf{G}_i$.

Similar to the update rule for $\mathbf{U}_{i*}$, we can obtain the update rule for $\mathbf{V}_{i*}$ as follows:

$$\mathbf{V}_{i*}(t+1) = \mathbf{V}_{i*}(t) - \Big[\frac{\partial L}{\partial \mathbf{V}_{i*}^T}(t)\Big]^T \times \mathbf{G}_i(t)^{-1}, \tag{7.13}$$

where $\mathbf{V}_{i*}(t)$ denotes the value of the former iteration and $\mathbf{G}_i(t)$ is computed with the updated parameters except for $\mathbf{V}_{i*}$.

Learning μ

Using similar learning techniques as those for $\mathbf{U}$ and $\mathbf{V}$, we can get the update rule for μ:

$$\mu(t+1) = \mu(t) + \frac{4[\sum_{k\neq i}(A_{ik} - Z_{ik}S_{ik}) - \tau\mu(t)]}{4\tau + \sum_{k\neq i} Z_{ik}}.$$

7.3.3 Convergence and Complexity Analysis

With the MM algorithm, the learning procedure of GLFM is guaranteed to converge to a local maximum.

The time complexity to compute the gradient and Hessian for node i is linear to the total number of ones in both $\mathbf{Z}_{*i}$ and $\mathbf{Z}_{i*}$. In general, this number is $O(1)$ because the observations in real networks are always very sparse. Furthermore, since $\mathbf{H}_i$ and $\mathbf{G}_i$ are $q\times q$, the computational complexity to invert the Hessian matrices is $O(q^3)$. Typically, q is a very small number. Hence, to update the whole $\mathbf{U}$ and $\mathbf{V}$, only $O(N)$ time is needed.

7.4 Experiments

There exist many different SNA tasks such as social position and role estimation (Wasserman & Katherine, 1994), link prediction (Hoff, 2009), node classification (Taskar et al., 2002; Silva et al., 2008), community detection (Yang et al., 2009a), and so on. Here, we adopt the same evaluation strategy as that in (Yang et al., 2009a, 2009b) for social community detection. The main reason for choosing this task is that from our model formulation we can clearly see the difference between GLFM and other latent factor models. However, many other models from different research communities have also been proposed for SNA. It is difficult to figure out the connection and difference between those models and GLFM from the formulation perspective. Hence, we use empirical evaluation to compare them. Most mainstream models have been compared in (Yang et al., 2009a, 2009b) for community detection, which provides a good platform for our empirical comparison.

For MLFM and GLFM, we adopt k-means to perform clustering based on the normalized latent representation $\mathbf{U}$. Here normalization means that the latent representation of each node is divided by its length. Since the magnitude of $\mathbf{U}_{i*}$ reflects the activity of i, we select the most active user as the first seed of the k-means, and then choose a point as the seed of the next community if summation of the distances between this point and all the existing seeds is the largest one. Hence, the initialization of k-means is fixed. The hyperparameters τ, β and γ are tuned to make GLFM achieve the best performance. More specifically, τ is fixed to 10^6, β and γ are set to 2. We set $\mathbf{Z} = \mathbf{A}$, which means we only model the observed links.

7.4.1 Data Sets

As in (Yang et al., 2009a), we use two paper citation networks, Cora and Citeseer data sets [4], for evaluation. Both data sets contain content information in addition to the directed links.

The Cora data set used for evaluation here is a subset of the larger Cora set (Craven et al., 1998). It contains 2708 research papers from the seven subfields of machine learning: case-based reasoning, genetic algorithms, neural networks, probabilistic methods, reinforcement learning, rule learning, and theory. Each paper is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from a dictionary of 1433 unique words. There are overall 5429 citations (links) between the papers.

The Citeseer data set contains 3312 papers which can be classified into six categories. Each paper is described by a 0/1-valued word vector indicating the absence/presence of

[4]The two data sets can be downloaded from `http://www.cs.umd.edu/projects/linqs/projects/lbc/index.html`.

the corresponding word from a dictionary of 3703 unique words. There are overall 4732 citations (links) between the papers. After deleting the self-links, we obtain 4715 links for our evaluation.

7.4.2 Evaluation Metric

As in (Yang et al., 2009a), we use *Normalized Mutual Information* (*NMI*), *Pairwise F-Measure* (*PWF*) and *Modularity* (*Modu*) as metrics to measure the clustering accuracy of our model. For all the algorithms, we set the number of communities to the ground-truth number of class labels in the data.

Assume that the ground-truth community structure is $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_K\}$, where K is the number of communities and C_k denotes the set of nodes belonging to the kth community. If the community assignment by an algorithm is $\mathcal{C}' = \{\mathcal{C}'_1, \mathcal{C}'_2, \cdots, \mathcal{C}'_K\}$, we can compute the *NMI* as follows:

$$NMI(\mathcal{C}, \mathcal{C}') = \frac{\sum_{k,j}\left\{p(\mathcal{C}_k, \mathcal{C}'_j)\log\frac{p(\mathcal{C}_k,\mathcal{C}'_j)}{p(\mathcal{C}_k)p(\mathcal{C}'_j)}\right\}}{\max\big(H(\mathcal{C}), H(\mathcal{C}')\big)},$$

where $H(\mathcal{C})$ and $H(\mathcal{C}')$ denote the entropies of the partitions $\mathcal{C}$ and $\mathcal{C}'$. The larger the *NMI*, the better the algorithm.

The *PWF* is defined as follows:

$$PWF = \frac{2 \times P \times R}{P + R},$$

where $P = \frac{|\mathcal{Y}\bigcap\mathcal{Y}'|}{|\mathcal{Y}'|}$, $R = \frac{|\mathcal{Y}\bigcap\mathcal{Y}'|}{|\mathcal{Y}|}$, $\mathcal{Y}$ is the set of node pairs having the same ground-truth label, and $\mathcal{Y}'$ is the set of node pairs assigned to the same community by the algorithm. We can see that the larger the *PWF*, the better the algorithm.

Given a community assignment by an algorithm $\mathcal{C}' = \{\mathcal{C}'_1, \mathcal{C}'_2, \cdots, \mathcal{C}'_K\}$, the modularity *Modu* (Newman, 2006) is computed as follows:

$$Modu = \sum_{k=1}^{K}\left[\frac{Cut(\mathcal{C}'_k, \mathcal{C}'_k)}{Cut(\mathcal{C}', \mathcal{C}')} - \left(\frac{Cut(\mathcal{C}'_k, \mathcal{C}')}{Cut(\mathcal{C}', \mathcal{C}')}\right)^2\right],$$

where $Cut(\mathcal{C}'_k, \mathcal{C}'_j) = \sum_{p\in\mathcal{C}'_k, q\in\mathcal{C}'_j} A_{pq}$.

7.4.3 Baselines

We compare GLFM with the closely related method MLFM (Hoff, 2009). The **U** and **V** in both MLFM and GLFM are initialized by PCA on the content information. In addition, we also adopt the methods introduced in (Yang et al., 2009a) and (Yang et al., 2009b) for comparison. Those methods can be divided into three groups: link-based methods, content-based methods, and link+content based methods.

The link-based methods include: PHITS (Cohn & Chang, 2000), LDA-Link (Erosheva et al., 2004)–an extension of latent Dirichlet allocation (LDA) for link analysis, the popularity-based conditional link model (PCL) (Yang et al., 2009b), and the normalized cut (NCUT) for spectral clustering (Shi & Malik, 2000).

The content-based methods include: the probabilistic latent semantic analysis (PLSA) (Hofmann, 1999), LDA-Word, and NCUT respectively with the Gaussian RBF kernel and the probabilistic product (PP) kernel (Jebara et al., 2004).

The link+content based methods include: PHITS-PLSA (Cohn & Hofmann, 2000), LDA-Link-Word (Erosheva et al., 2004), link-content matrix factorization (LCMF) (Zhu et al., 2007), NCUT, PCL-PLSA, PHITS-DC, PCL-DC and C-PLDC. Here PCL-PLSA represents the combination of PCL and PLSA, PHITS-DC represents the PHITS model combined with the discriminative content model (DC), PCL-DC represents the PCL model combined with DC, and C-PLDC refers to the combined popularity-driven link model and DC model (Yang et al., 2009a). Moreover, the setting for t in C-PLDC follows that in (Yang et al., 2009a). More specifically, C-PLDC($t = 1$) denotes a special case of C-PLDC without popularity modeling (Yang et al., 2009a).

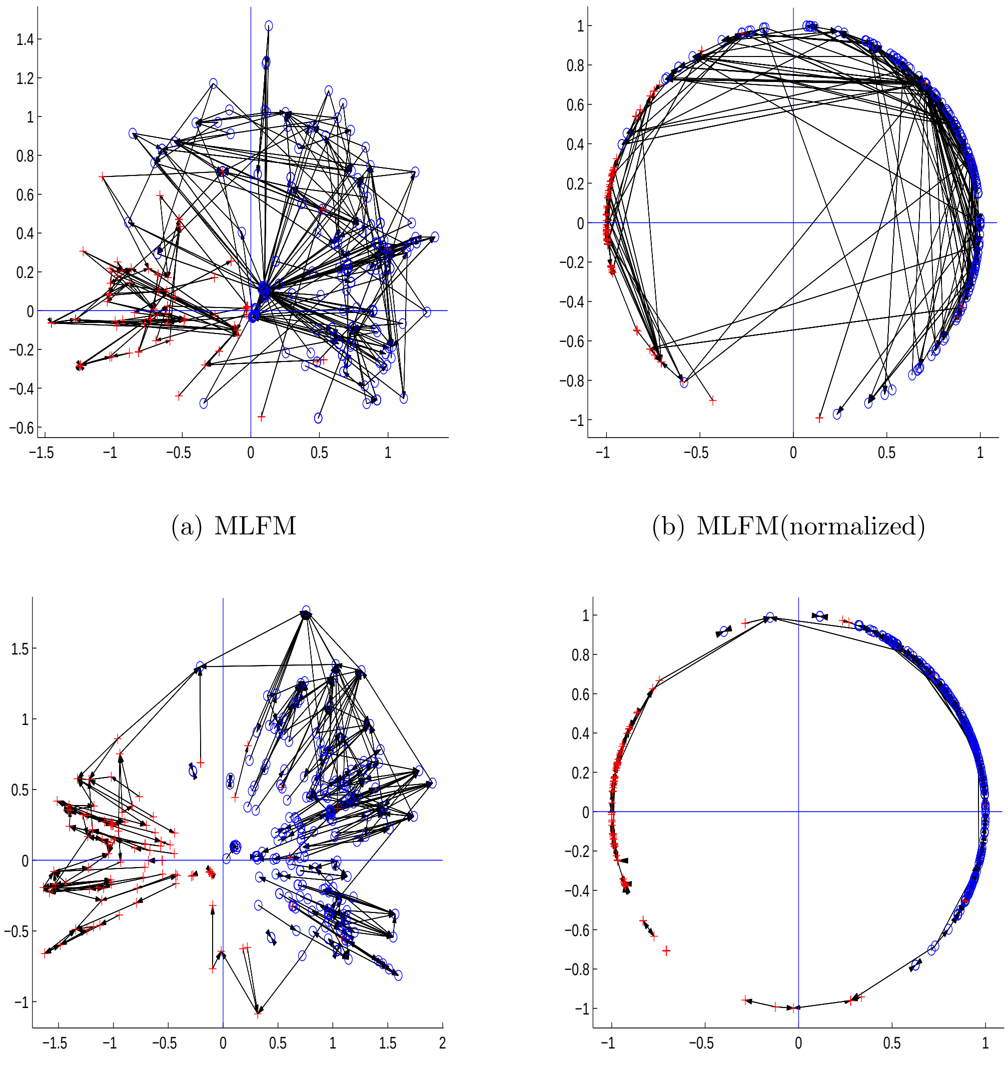
7.4.4 Illustration

Here, we use real data to verify our claim that MLFM cannot model homophily effectively. On the contrary, GLFM provides a potential approach for modeling homophily in networks.

We sample a subset from Cora for illustration. The sampled data set contains two classes. The learned latent representations $\mathbf{U}$ for the data instances are illustrated in Figure 7.1, where the blue circle and red cross are used to denote the data instances from two different classes respectively, and the (directed) black edges are the citation relationships between the data points. In Figure 7.1, (a) and (c) show the original learned latent factors of MLFM and GLFM, respectively, (b) and (d) show the corresponding normalized latent factors of MLFM and GLFM, respectively. Here normalization means that we divide the latent factor of each node by its length. Hence, it is clear to see that all the points in (b) and (d) have unit length. Note that for fair comparison all the different subfigures from (a) to (d) are generated automatically by our program with the same parameter settings and initial values.

In (a) and (b) of Figure 7.1, two instances are more likely to be close if they are connected by or connect to the same instance, which is just the feature of stochastic equivalence. However, there exist many links across the inner part of the circle in (b), which means that two instances linked with each other are not necessarily close in the latent space. This just violates the feature of homophily. Hence, we can conclude that MLFM cannot effectively model homophily in networks.

In (c) and (d) of Figure 7.1, homophily is obvious since two nodes are close to each other in general if there exists a link between them.



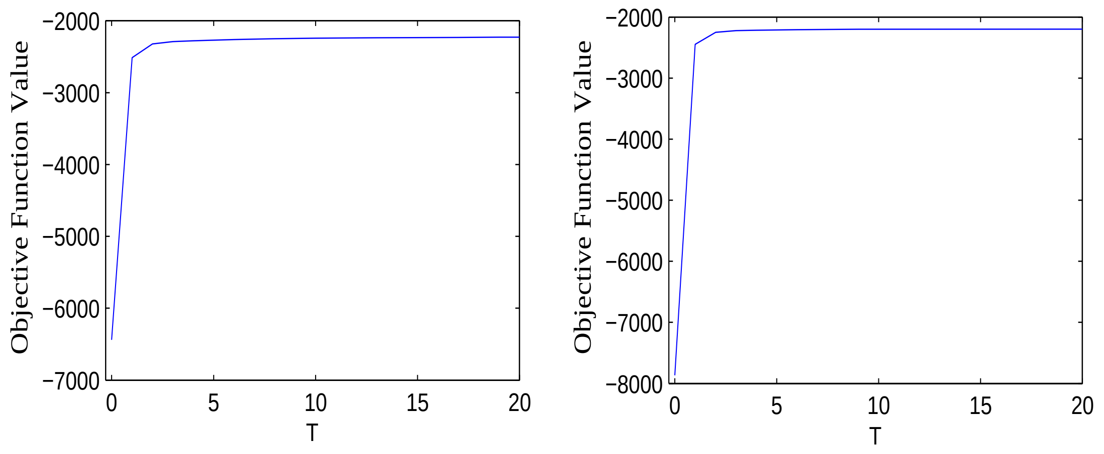
(a) MLFM (b) MLFM(normalized)

(c) GLFM (d) GLFM(normalized)

Figure 7.1: Illustration of the homophily and stochastic equivalence modeling in networks.

7.4.5 Convergence Speed

When $q = 20$, the objective function values of GLFM against the iteration number T are plotted in Figure 7.2, from which we can see that our learning procedure with the MM method for GLFM converges very fast. We set the maximum iterative number T as $T = 5$ in all our following experiments.

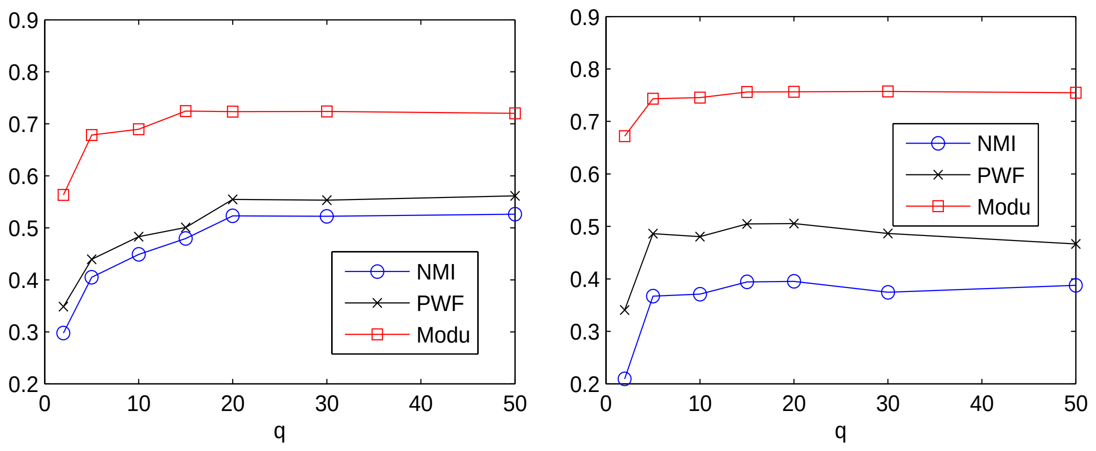


(a) Cora (b) Citeseer

Figure 7.2: Convergence speed of GLFM.

7.4.6 Sensitivity to Parameters

In GLFM, q (the dimension of the latent space) is the main parameter to affect its performance. Figure 7.3 shows the performance of GLFM when q takes different values. We see that GLFM is not sensitive to q as long as q is not too small. Unless otherwise stated, we set $q = 20$.



(a) Cora (b) Citeseer

Figure 7.3: Sensitivity to the parameter q of GLFM.

7.4.7 Accuracy Comparison

We compare GLFM with all the baselines introduced in Section 7.4.3 in terms of NMI, PWF and $Modu$. The results are reported in Table 7.1, from which we can see that GLFM achieves the best performance on all the data sets for the three criteria. Especially for the Citeseer data set, GLFM dramatically outperforms the second best model. According to the prior knowledge, the paper citation networks are more likely to exhibit homophily because the citations often exist among papers from the same community. This can explain why GLFM can achieve such good performance on these data sets. Hence, GLFM provides a way to model networks which cannot be modeled well by MLFM.

Table 7.1: Community detection performance on Cora and Citeseer data sets (the best performance is shown in bold face).

		Cora			Citeseer		
	Algorithm	NMI	PWF	$Modu$	NMI	PWF	$Modu$
Link	PHITS	0.0570	0.1894	0.3929	0.0101	0.1773	0.4588
	LDA-Link	0.0762	0.2278	0.2189	0.0356	0.2363	0.2211
	PCL	0.0884	0.2055	0.5903	0.0315	0.1927	0.6436
	NCUT	0.1715	0.2864	0.2701	0.1833	0.3252	0.6577
Content	PLSA	0.2107	0.2864	0.2682	0.0965	0.2298	0.2885
	LDA-Word	0.2310	0.2774	0.2970	0.1342	0.2880	0.3022
	NCUT(RBF kernel)	0.1317	0.2457	0.1839	0.0976	0.2386	0.2133
	NCUT(pp kernel)	0.1804	0.2912	0.2487	0.1986	0.3282	0.4802
Link + Content	PHITS-PLSA	0.3140	0.3526	0.3956	0.1188	0.2596	0.3863
	LDA-Link-Word	0.3587	0.3969	0.4576	0.1920	0.3045	0.5058
	LCMF	0.1227	0.2456	0.1664	0.0934	0.2361	0.2011
	NCUT(RBF kernel)	0.2444	0.3062	0.3703	0.1592	0.2957	0.4280
	NCUT(pp kernel)	0.3866	0.4214	0.5158	0.1986	0.3282	0.4802
	PCL-PLSA	0.3900	0.4233	0.5503	0.2207	0.3334	0.5505
	PHITS-DC	0.4359	0.4526	0.6384	0.2062	0.3295	0.6117
	PCL-DC	0.5123	0.5450	0.6976	0.2921	0.3876	0.6857
	C-PLDC(t=1)	0.4294	0.4264	0.5877	0.2303	0.3340	0.5530
	C-PLDC	0.4887	0.4638	0.6160	0.2756	0.3611	0.5582
	MLFM	0.3640	0.3874	0.2325	0.2558	0.3356	0.0089
	GLFM	**0.5229**	**0.5545**	**0.7234**	**0.3951**	**0.5053**	**0.7563**

7.5 Conclusion

In this chapter, a generalized latent factor model is proposed to model homophily in social networks with directed links. A linear-time learning algorithm with convergence guarantee is proposed to learn the parameters. Experimental results on community detection in real-world networks show that our model can effectively model homophily to outperform state-of-the-art methods.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

In this chapter, we conclude the whole thesis and propose several possible directions for future pursuit.

8.1 Conclusions

In this thesis, we have proposed a series of novel SRL models, called *relational factor models* (RFMs), by extending traditional *latent factor models* from i.i.d. domains to relational domains. Unlike the existing mainstream SRL models such as PLMs and PRMs which need to perform time-consuming structure learning, our RFMs provide an alternative way for SRL which is efficient enough for large-scale SRL problems. We briefly summarize our RFMs here:

- Relation regularized matrix factorization (RRMF): We propose a novel MF method, called RRMF, for relational data analysis. By using relational information to regularize the content MF procedure, RRMF seamlessly integrates both the relational information and the content information into a principled framework. We propose a linear-time learning algorithm with convergence guarantee to learn the parameters of RRMF. Extensive experiments on linked-document classification show that RRMF can outperform other MF variants.

- Probabilistic relational PCA (PRPCA): By explicitly modeling covariance between instances as derived from the relational information, we propose a novel probabilistic projection method, called PRPCA, for relational data analysis. Although the i.i.d. assumption is no longer adopted in PRPCA, the learning algorithms for PRPCA can still be devised easily like those for PPCA which makes explicit use of the i.i.d. assumption. Experiments on real-world data sets show that PRPCA can effectively utilize the relational information to dramatically outperform PCA.

- Sparse probabilistic relational projection (SPRP): We propose a sparse version of PRPCA, called SPRP, to learn a sparse projection matrix for relational dimensionality reduction. The sparsity in SPRP is achieved by imposing on the projection matrix a sparsity-inducing prior such as the Laplace prior or Jeffreys prior. An EM algorithm is derived to learn the parameters of SPRP. Compared with PRPCA, the sparsity in SPRP not only makes the results more interpretable but also makes the projection operation much more efficient without compromising its accuracy. Furthermore, compared with traditional sparse projection methods based on the i.i.d. assumption, SPRP can learn a more discriminative projection by explicitly modeling the covariance between instances.

- Latent Wishart processes (LWP): We propose a novel relational kernel learning model, called LWP, to learn the kernel function for relational data. This is done by seamlessly integrating the relational information and the input attributes into the kernel learning process. Through extensive experiments on real-world applications, we demonstrate that our LWP model can give very promising performance in practice.

- Generalized latent factor model (GLFM): We propose a novel model, called GLFM, to model social networks with directed links by enhancing homophily modeling in MLFM. We devise a learning algorithm with linear-time complexity and convergence guarantee to learn the model parameters. Extensive experiments on some real-world networks show that GLFM can effectively model homophily to dramatically outperform state-of-the-art methods.

The connections and differences between our RFMs are illustrated in Figure 1.2, from which we can see that our RFMs range from transductive to inductive, from parametric to nonparametric, and from undirected relation modeling to directed relation modeling. More specifically, GLFM can be used to model directed relations, while all the other models can only model undirected relations. PRPCA and SPRP can learn a transformation matrix ($\mathbf{W}$) to perform dimensionality reduction, and subsequently, they are parametric. However, RRMF and LWP do not have a parametric form, and hence they are nonparametric. PRPCA, SPRP and LWP can be used to perform out-of-sample inference, which means that they have inductive inference ability. However, RRMF can not be used for out-of-sample inference, and hence can only be used for transductive inference.

In general, nonparametric methods have stronger representation ability than parametric methods. For example, RRMF has achieved better performance than PRPCA and SPRP on Cora and WebKB data sets. However, it is not easy to perform out-of-sample (i.e., inductive) inference for nonparametric methods. For example, RRMF cannot be used for out-of-sample inference at all and LWP has time complexity of $O(N^3)$ for out-of-sample inference. Therefore, if out-of-sample inference is unnecessary, we can choose both RRMF and LWP, but RRMF is faster than LWP. Otherwise, if we need to perform out-of-sample inference, we can choose PRPCA (or SPRP) and LWP, but PRPCA is faster than LWP. If we want to perform feature selection or to get interpretable results, SPRP is a good choice. In sum, our proposed RFMs in this thesis provide a toolbox for different learning settings in SRL.

As stated in Section 1.1, a relational data set can be represented by a set of matrices. Currently, the proposed RFMs in this thesis can only model relational data sets containing one or two matrices. However, it is easy to extend our RFMs to model data sets with more than two matrices by adopting the same techniques as those in *collective matrix factorization* (CMF) (Singh & Gordon, 2008a, 2008b; Singh, 2009) [1]. The basic idea is to share latent factors for different relations if those relations share the same entities. More specifically, let R_{ij}^k denote the kth relation between entity type $\mathcal{E}_i$ and $\mathcal{E}_j$, and $\mathbf{U}_i$ are the latent factors for $\mathcal{E}_i$. Since R_{ij}^k and R_{im}^p share the same entity type $\mathcal{E}_i$, the same $\mathbf{U}_i$ should be used for $\mathcal{E}_i$ in these two relations. More details and examples can be found in (Singh, 2009; Sutskever et al., 2009).

As stated in Section 2.1.1, there exist many different SRL tasks, such as link-based clustering (i.e., social community detection) and link prediction. Although in this thesis we mainly perform experiments for linked-document classification and social community detection, our RFMs can also be used for other tasks. For example, we can use the learned latent factors $\mathbf{B}$ in LWP for link prediction. If the distance between $\mathbf{B}_{i*}$ and $\mathbf{B}_{j*}$ is less than some threshold, we can say that there exists a link between instances i and j. This prediction strategy can be expected to achieve good performance, based on the illustration in Figure 6.3. Similarly, other RFMs can also be used for link prediction. From Figure 7.1(c), we can see that the magnitude of the latent factor $\mathbf{U}_{i*}$ in GLFM can reflect the activity of instance i. Hence, we can adapt the learned results of GLFM for ranking problems. In sum, our RFMs are flexible enough to provide potential solutions for different SRL tasks. All these possible extensions with empirical verifications are left for our future pursuit.

8.2 Future Work

In future work, we will pursue the following potential directions:

- Improving the current RFMs: As the *link-content sup. MF* in (Zhu et al., 2007), we can incorporate label information into RRMF to get a supervised version of RRMF. As for PRPCA, we can extend it to handle data with missing values. In addition, a mixture of PRPCA is also very interesting. As for SPRP, we will consider other sparsity-inducing priors and empirically compare them. As for LWP, the current

[1]CMF can also be treated as a latent factor model for SRL. The underlying principle of CMF is similar to that of LCMF (Zhu et al., 2007). Hence, as stated in Chapter 3, CMF cannot model well some relations exhibiting homophily.

learning algorithm does not necessarily converge. We will design more sophisticated algorithms to learn the parameters in LWP. As for GLFM, it can only perform transductive inference. It is very meaningful to extend it for out-of-sample inference.

- Designing dynamic RFMs: Most traditional relational models and our RFMs introduced above can only model *static* network which is either derived from an aggregation of data over all time or taken as a snapshot of data at a specific point in time. However, many real-world networks are *dynamic* which means that the relationships between the instances will evolve over time. For example, in a network of friendships, A and B will not necessarily be friends at time $t+1$ even if they are friends at time t. How to extend our RFMs to model dynamic relational data will be very interesting but challenging.

- Applying RFMs for other SRL tasks and new applications: As said above, our RFMs can easily be adapted for other SRL tasks such as link prediction and ranking. Extensive empirical evaluation of our RFMs on other SRL tasks will be very interesting. Furthermore, SRL can be applied to a large variety of application areas, such as web mining, social network analysis, bioinformatics, economics and marketing. In our future work, besides linked-document classification and social community detection, we will apply our models to other application domains, which is practically important and will also give insights for us to develop novel SRL models.

Bibliography

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734–749.

Agarwal, D., & Chen, B.-C. (2009). Regression-based latent factor models. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Airoldi, E. M., Blei, D. M., Fienberg, S. E., & Xing, E. P. (2008). Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, *9*, 1981–2014.

Archambeau, C., & Bach, F. (2008). Sparse probabilistic projections. In *Advances in Neural Information Processing Systems (NIPS).*

Bartholomew, D. J., & Knott, M. (1999). *Latent Variable Models and Factor Analysis* (Second edition). Kendall's Library of Statistics,7.

Bhattacharya, I., & Getoor, L. (2007). Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *1*(1).

Bishop, C. M. (1998). Bayesian PCA. In *Advances in Neural Information Processing Systems (NIPS).*

Caron, F., & Doucet, A. (2008). Sparse Bayesian nonparametric regression. In *Proceedings of the International Conference on Machine Learning (ICML).*

Castillo, C., Donato, D., Gionis, A., Murdock, V., & Silvestri, F. (2007). Know your neighbors: web spam detection using the web topology. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Chakrabarti, S., Dom, B., & Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. In *Proceedings ACM SIGMOD International Conference on Management of Data.*

Chang, C.-C., & Lin, C.-J. (2001). *LIBSVM: a Library for Support Vector Machines.* Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chang, J., & Blei, D. M. (2009). Relational topic models for document networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS).*

Chang, J., Boyd-Graber, J. L., & Blei, D. M. (2009). Connections between the lines: augmenting social networks with text. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Chen, P. P. (1976). The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*, *1*(1), 9–36.

Chu, W., Sindhwani, V., Ghahramani, Z., & Keerthi, S. S. (2007). Relational learning with Gaussian processes. In *Advances in Neural Information Processing Systems (NIPS).*

Chung, F. (1997). *Spectral Graph Theory.* Number 92 in Regional Conference Series in Mathematics. American Mathematical Society.

Cohn, D., & Chang, H. (2000). Learning to probabilistically identify authoritative documents. In *Proceedings of the International Conference on Machine Learning (ICML).*

Cohn, D. A., & Hofmann, T. (2000). The missing link - a probabilistic model of document content and hypertext connectivity. In *Advances in Neural Information Processing Systems (NIPS).*

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T. M., Nigam, K., & Slattery, S. (1998). Learning to extract symbolic knowledge from the world wide web. In *AAAI/IAAI*, pp. 509–516.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science and Technology (JASIST)*, *41*(6), 391–407.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, *39*(1), 1–38.

Domingos, P. (2003). Prospects and challenges for multi-relational data mining. *SIGKDD Explorations*, *5*(1), 80–83.

Dzeroski, S. (2003). Multi-relational data mining: an introduction. *SIGKDD Explorations*, *5*(1), 1–16.

Erosheva, E., Fienberg, S., & Lafferty, J. (2004). Mixed membership models of scientific publications. In *Proceedings of the National Academy of Sciences*, Vol. 101, pp. 5220–5227.

Fawcett, T., & Provost, F. J. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery*, *1*(3), 291–316.

Figueiredo, M. A. T. (2003). Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(9), 1150–1159.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Geman, S., & Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *6*(6).

Getoor, L., & Diehl, C. P. (2005). Link mining: a survey. *SIGKDD Explorations*, *7*(2), 3–12.

Getoor, L., Friedman, N., Koller, D., & Taskar, B. (2002). Learning probabilistic models of link structure. *Journal of Machine Learning Research*, *3*, 679–707.

Getoor, L., & Taskar, B. (2007). *Introduction to Statistical Relational Learning*. The MIT Press.

Goldenberg, A., Zheng, A. X., Fienberg, S. E., & Airoldi, E. M. (2009). A survey of statistical network models. *Foundations and Trends in Machine Learning*, *2*, 129–233.

Guan, Y., & Dy, J. G. (2009). Sparse probabilistic principal component analysis. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Gupta, A. K., & Nagar, D. K. (2000). *Matrix Variate Distributions*. Chapman & Hall/CRC.

Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., & Kadie, C. M. (2000). Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, *1*, 49–75.

Heckerman, D., Meek, C., & Koller, D. (2004). Probabilistic models for relational data. Technical report MSR-TR-2004-30, Microsoft Research and Stanford University.

Hill, S., Provost, F., & Volinsky, C. (2006). Network-based marketing: Identifying likely adopters via consumer networks. *Statistical Science*, *21*(2), 256–276.

Hill, S., Provost, F., & Volinsky, C. (2007). Learning and inference in massive social networks. In *The 5th International Workshop on Mining and Learning with Graphs.*

Hoff, P. D. (2007). Modeling homophily and stochastic equivalence in symmetric relational data. In *Advances in Neural Information Processing Systems (NIPS).*

Hoff, P. D. (2009). Multiplicative latent factor models for description and prediction of social networks. *Computational and Mathematical Organization Theory*, *15*, 261–272.

Hoff, P. D., Raftery, A. E., & Handcock, M. S. (2002). Latent space approaches to social network analysis. *Journal of the American Statistical Association*, *97*(460), 1090–1098.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Howard, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, *27*, 417–441.

Jebara, T., Kondor, R. I., & Howard, A. (2004). Probability product kernels. *Journal of Machine Learning Research*, *5*, 819–844.

Jensen, D., Neville, J., & Gallagher, B. (2004). Why collective inference improves relational classification. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Jolliffe, I. T. (2002). *Principal Component Analysis* (Second edition). Springer.

Jordan, M. I., Ghahramani, Z., Jaakkola, T., & Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, *37*(2), 183–233.

Kemp, C., Tenenbaum, J. B., Griffiths, T. L., Yamada, T., & Ueda, N. (2006). Learning systems of concepts with an infinite relational model. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI).*

Kersting, K., & Raedt, L. D. (2001). Adaptive bayesian logic programs. In *Proceedings of the International Conference on Inductive Logic Programming (ILP).*

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

Kurland, O., & Lee, L. (2005). Pagerank without hyperlinks: structural re-ranking using links induced by language models. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Lanckriet, G. R. G., Cristianini, N., Bartlett, P. L., Ghaoui, L. E., & Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, *5*, 27–72.

Lang, K., Hunter, D. R., & Yang, I. (2000). Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, *9*.

Li, W.-J., & Yeung, D.-Y. (2009). Relation regularized matrix factorization. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).*

Li, W.-J., Yeung, D.-Y., & Zhang, Z. (2009a). Probabilistic relational PCA. In *Advances in Neural Information Processing Systems (NIPS).*

Li, W.-J., Zhang, Z., & Yeung, D.-Y. (2009b). Latent Wishart processes for relational kernel learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS).*

Lu, Q., & Getoor, L. (2003). Link-based classification. In *Proceedings of the International Conference on Machine Learning (ICML).*

Macskassy, S. A., & Provost, F. J. (2007). Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, *8*, 935–983.

McCallum, A., Nigam, K., Rennie, J., & Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval*, *3*(2), 127–163.

Memisevic, R. (2008). *Non-Linear Latent Factor Models for Revealing Structure in High-Dimensional Data*. Ph.D. thesis, University of Toronto.

Miller, K. T., Griffiths, T. L., & Jordan, M. I. (2009). Nonparametric latent feature models for link prediction. In *Advances in Neural Information Processing Systems (NIPS).*

Muggleton, S. (1991). Inductive logic programming. *New Generation Computing*, *8*(4), 295–318.

Muggleton, S. (2000). Learning stochastic logic programs. *Electronic Transactions on Artificial Intelligence*, *4*(B), 141–153.

Nallapati, R., Ahmed, A., Xing, E. P., & Cohen, W. W. (2008). Joint latent topic models for text and citations. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Neville, J. (2006). *Statistical Models and Analysis Techniques for Learning in Relational Data*. Phd thesis, University of Massachusetts Amherst.

Neville, J., & Jensen, D. (2000). Iterative classification in relational data. In *AAAI Workshop Learning Statistical Models From Relational Data*.

Neville, J., & Jensen, D. (2007). Relational dependency networks. *Journal of Machine Learning Research*, *8*, 653–692.

Neville, J., Jensen, D., Friedland, L., & Hay, M. (2003a). Learning relational probability trees. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Neville, J., Jensen, D., & Gallagher, B. (2003b). Simple estimators for relational bayesian classifiers. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*.

Newman, M. E. J. (2006). Modularity and community structure in networks. In *Proceedings of the National Academy of Sciences*, Vol. 103, pp. 8577–8582.

Ng, R. T., & Subrahmanian, V. S. (1992). Probabilistic logic programming. *Information and Computation*, *101*(2), 150–201.

Ngo, L., & Haddawy, P. (1997). Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, *171*(1-2), 147–177.

Nowicki, K., & Snijders, T. A. B. (2001). Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, *96*, 1077–1087.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report 1999-66, Stanford InfoLab.

Park, T., & Casella, G. (2008). The Bayesian lasso. *Journal of the American Statistical Association*, *103*(482), 681–686.

Poole, D. (1993). Probabilistic horn abduction and bayesian networks. *Artificial Intelligence*, *64*(1), 81–129.

Poole, D. (1997). The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, *94*(1-2), 7–56.

Raedt, L. D., & Kersting, K. (2003). Probabilistic logic learning. *SIGKDD Explorations*, *5*(1), 31–48.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Rennie, J. D. M., & Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*(1-2), 107–136.

Sato, T., & Kameya, Y. (2001). Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, *15*, 391–454.

Schölkopf, B., Smola, A. J., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*(5), 1299–1319.

Schölkopf, B., & Smola, A. J. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Segal, E., Wang, H., & Koller, D. (2003a). Discovering molecular pathways from protein interaction and gene expression data. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology (ISMB)*.

Segal, E., Yelensky, R., & Koller, D. (2003b). Genome-wide discovery of transcriptional modules from dna sequence and gene expression. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology (ISMB)*.

Sen, P., Namata, G. M., Bilgic, M., Getoor, L., Gallagher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29*(3).

Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain. Tech. rep., Pittsburgh, PA, USA.

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

Sigg, C. D., & Buhmann, J. M. (2008). Expectation-maximization for sparse and non-negative PCA. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Silva, R., Chu, W., & Ghahramani, Z. (2008). Hidden common cause relations in relational learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Singh, A. P. (2009). *Efficient Models for Relational Learning*. Ph.D. thesis, Carnegie Mellon University.

Singh, A. P., & Gordon, G. J. (2008a). Relational learning via collective matrix factorization. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Singh, A. P., & Gordon, G. J. (2008b). A unified view of matrix factorization models. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*.

Srebro, N., Rennie, J. D. M., & Jaakkola, T. (2004). Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*.

Srivastava, M. S. (2003). Singular Wishart and multivariate Beta distributions. *The Annals of Statistics*, *31*(5), 1537–1560.

Sutskever, I., Salakhutdinov, R., & Tenenbaum, J. B. (2009). Modelling relational data using bayesian clustered tensor factorization. In *Advances in Neural Information Processing Systems (NIPS)*.

Tang, L., & Liu, H. (2009). Relational learning via latent social dimensions. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. In *Proceedings of the Conference on Uncertainty in Aritifical Intelligene (UAI)*.

Taskar, B., Segal, E., & Koller, D. (2001). Probabilistic classification and clustering in relational data. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Taskar, B., Wong, M. F., Abbeel, P., & Koller, D. (2003). Link prediction in relational data. In *Advances in Neural Information Processing Systems (NIPS)*.

Teh, Y. W., Seeger, M., & Jordan, M. I. (2005). Semiparametric latent factor models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Tipping, M. E., & Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, *61*(3), 611–622.

Vert, J.-P. (2009). Reconstruction of biological networks by supervised machine learning approaches. In *Elements of Computational Systems Biology*.

Wasserman, S., & Katherine, F. (1994). *Social Network Analysis: Methods and Applications*. Cambridge University Press.

Xu, Z. (2007). *Statistical Relational Learning with Nonparametric Bayesian Models*. Ph.D. thesis, Ludwig-Maximilians-University of Munich.

Xu, Z., Tresp, V., Yu, K., & Kriegel, H.-P. (2006). Infinite hidden relational models. In *Proceedings of the Conference on Uncertainty in Aritifical Intelligene (UAI)*.

Yang, T., Jin, R., Chi, Y., & Zhu, S. (2009a). A Bayesian framework for community detection integrating content and link. In *Proceedings of the Conference on Uncertainty in Aritifical Intelligene (UAI)*.

Yang, T., Jin, R., Chi, Y., & Zhu, S. (2009b). Combining link and content for community detection: a discriminative approach. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Yu, K., & Chu, W. (2007). Gaussian process models for link analysis and transfer learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Yu, K., Chu, W., Yu, S., Tresp, V., & Xu, Z. (2006). Stochastic relational models for discriminative link prediction. In *Advances in Neural Information Processing Systems (NIPS)*.

Zhang, Z., Kwok, J. T., & Yeung, D.-Y. (2006). Model-based transductive learning of the kernel matrix. *Machine Learning*, *63*(1), 69–101.

Zhen, Y., Li, W.-J., & Yeung, D.-Y. (2009). TagiCoFi: tag informed collaborative filtering. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*.

Zhou, D., Huang, J., & Schölkopf, B. (2005). Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Zhou, D., Schölkopf, B., & Hofmann, T. (2004). Semi-supervised learning on directed graphs. In *Advances in Neural Information Processing Systems (NIPS)*.

Zhu, S., Yu, K., Chi, Y., & Gong, Y. (2007). Combining content and link for classification using matrix factorization. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, *67*(2), 301–320.

Zou, H., Hastie, T., & Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, *15*(2), 265–286.

APPENDIX A

NOTATIONS AND ABBREVIATIONS

A.1 Notations

Some commonly used notations in this thesis are listed in Table A.1.

Table A.1: Notations

$\mathbb{R}$	real number space
$\mathbb{R}^d$	d-dimensional Euclidean space
$\mathbf{K}$	boldface uppercase letters denote matrices
$\mathbf{K} \in \mathbb{R}^{m \times n}$	$\mathbf{K}$ is of size $m \times n$
$\mathbf{K}_{i*}$	the ith row of matrix $\mathbf{K}$
$\mathbf{K}_{*j}$	the jth column of matrix $\mathbf{K}$
K_{ij}	the (i, j)th entry of matrix $\mathbf{K}$
$\mathbf{z}$	boldface lowercase letters denote (column) vectors
z_i	the ith element of vector $\mathbf{z}$
$\mathbf{z}_i$	vector indexed for some purpose
$\mathbf{K}^T$	transpose of matrix $\mathbf{K}$
$\mathbf{K}^{-1}$	inverse of matrix $\mathbf{K}$
$\mathrm{tr}(\mathbf{K})$	trace of matrix $\mathbf{K}$
$\mathrm{etr}(\mathbf{K})$	$\exp(\mathrm{tr}(\mathbf{K}))$
$\mathbf{K} \succeq 0$	$\mathbf{K}$ is positive semidefinite (psd)
$\mathbf{K} \succ 0$	$\mathbf{K}$ is positive definite (pd)
$\mathbf{K} \succeq \mathbf{M}$	$\mathbf{K} - \mathbf{M} \succeq 0$
$\|\mathbf{K}\|_1$	the L_1 norm of matrix $\mathbf{K}$
$\|\mathbf{K}\|^2$	equal to $\mathrm{tr}(\mathbf{K}^T\mathbf{K})$
$\mathbf{I}_n$	identity matrix of size $n \times n$
$\mathbf{0}$	a matrix with zeroes in all entries
$\mathbf{e}$	a vector of 1s
$\mathcal{N}(\cdot)$	normal distributions or multivariate normal distributions
$\mathcal{N}_{m,n}(\cdot)$	for matrix variate normal distributions
$\mathbf{K} \circ \mathbf{L}$	Hadamard (elementwise) product
$\mathbf{K} \otimes \mathbf{L}$	Kronecker product
$\langle \cdot \rangle$	expectation operation
$\mathrm{cov}(\cdot)$	covariance operation
$\mathrm{diag}(\mathbf{v})$	convert the vector $\mathbf{v}$ into a diagonal matrix
$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$	a set of n points
$|\mathcal{X}|$	cardinality of set $\mathcal{X}$
$|\mathbf{K}|$	determinant if $\mathbf{K}$ is a matrix

A.2 Abbreviations

Table A.2 lists some abbreviations and their corresponding full names in this thesis.

Table A.2: Abbreviations and their corresponding full names

EM	expectation-maximization
GLFM	generalized latent factor model
GP	Gaussian process
HCI	heuristic collective inference
ILP	inductive logic programming
LCMF	link-content matrix factorization
LFM	latent factor model
LSI	latent semantic indexing
LVM	latent variable model
LWP	latent Wishart process
MF	matrix factorization
MLFM	multiplicative latent factor model
MLN	Markov logic network
MM	minorization-maximization
MMMF	maximum margin matrix factorization
PCA	principal component analysis
PLM	probabilistic logic model
PPCA	probabilistic PCA
PRM	probabilistic relational model
PRPCA	probabilistic relational PCA
RBN	relational Bayesian network
RDN	relational dependency network
RFM	relational factor model
RGP	relational Gaussian process
RKL	relational kernel learning
RMN	relational Markov network
RRMF	relation regularized matrix factorization
SNA	social network analysis
SPCA	sparse PCA
SPRP	sparse probabilistic relational projection
SRL	statistical relational learning
SVD	singular value decomposition
SVM	support vector machine
XGP	mixed graph Gaussian process

APPENDIX B

DERIVATION

B.1 Derivation of the EM Algorithm for PRPCA

The complete-data log-likelihood is

$$\begin{aligned} L_c &= \ln p(\mathbf{T}, \mathbf{X}) \\ &= \ln p(\mathbf{T} \mid \mathbf{X}) + \ln p(\mathbf{X}) \\ &= -\frac{Nd}{2}\ln\sigma^2 - \frac{1}{2\sigma^2}\Big[\mathrm{tr}\big[(\mathbf{T}-\boldsymbol{\mu}\mathbf{e}^T)\boldsymbol{\Delta}(\mathbf{T}-\boldsymbol{\mu}\mathbf{e}^T)^T\big] \\ &\qquad - 2\mathrm{tr}\big[(\mathbf{T}-\boldsymbol{\mu}\mathbf{e}^T)\boldsymbol{\Delta}\mathbf{X}^T\mathbf{W}^T\big] + \mathrm{tr}(\mathbf{W}^T\mathbf{W}\mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T)\Big], \end{aligned} \tag{B.1}$$

where we have omitted the terms independent of the parameters $\mathbf{W}$ and σ^2. It is easy to find that $\mathbf{X}$ and $\mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T$ are the complete-data sufficient statistics for $\mathbf{W}$ and σ^2.

Based on the definition of $p(\mathbf{X})$ in (4.11) and $p(\mathbf{T} \mid \mathbf{X})$ in (4.12), and using some properties of matrix variate normal distributions (Gupta & Nagar, 2000), we can get

$$p(\mathbf{X} \mid \mathbf{T}) = \mathcal{N}_{q,N}(\mathbf{M}^{-1}\mathbf{W}^T(\mathbf{T}-\boldsymbol{\mu}\mathbf{e}^T), \sigma^2\mathbf{M}^{-1} \otimes \boldsymbol{\Phi}), \tag{B.2}$$

where $\mathbf{M} = \mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I}_q$.

Then, we can get

$$\langle\mathbf{X}\rangle = \mathbf{M}^{-1}\mathbf{W}^T(\mathbf{T}-\boldsymbol{\mu}\mathbf{e}^T), \tag{B.3}$$

$$\langle\mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T\rangle = N\sigma^2\mathbf{M}^{-1} + \langle\mathbf{X}\rangle\boldsymbol{\Delta}\langle\mathbf{X}\rangle^T, \tag{B.4}$$

where the result of $\langle\mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T\rangle$ is derived based on Theorem 2.3 and the fact that $\mathrm{tr}(\boldsymbol{\Delta}\boldsymbol{\Phi}) = \mathrm{tr}(\mathbf{I}_N) = N$.

The E-step computes the expectation of L_c with respect to $p(\mathbf{X} \mid \mathbf{T}, \mathbf{W}(t), \sigma^2(t))$, giving

$$\begin{aligned} Q(\mathbf{W}, \sigma^2 \mid \mathbf{W}(t), \sigma^2(t)) &= -\frac{Nd}{2}\ln\sigma^2 - \frac{1}{2\sigma^2}\Big[\mathrm{tr}\big[(\mathbf{T}-\boldsymbol{\mu}\mathbf{e}^T)\boldsymbol{\Delta}(\mathbf{T}-\boldsymbol{\mu}\mathbf{e}^T)^T\big] \\ &\qquad - 2\mathrm{tr}\big[(\mathbf{T}-\boldsymbol{\mu}\mathbf{e}^T)\boldsymbol{\Delta}\langle\mathbf{X}\rangle^T\mathbf{W}^T\big] + \mathrm{tr}(\mathbf{W}^T\mathbf{W}\langle\mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T\rangle)\Big], \end{aligned} \tag{B.5}$$

where $\mathbf{W}(t)$ and $\sigma^2(t)$ denote the parameter values of the tth iteration, and $\langle\mathbf{X}\rangle$ and $\langle\mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T\rangle$ are computed based on $\mathbf{W}(t)$ and $\sigma^2(t)$.

In the M-step, $Q(\mathbf{W}, \sigma^2 \mid \mathbf{W}(t), \sigma^2(t))$ is maximized with respect to $\mathbf{W}$ and σ^2, giving their new values:

$$\mathbf{W}(t+1) = (\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T)\boldsymbol{\Delta}\langle\mathbf{X}\rangle^T\langle\mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T\rangle^{-1} = \mathbf{H}\mathbf{W}(t)\Big(\sigma^2(t)\mathbf{I}_q + \mathbf{M}^{-1}(t)\mathbf{W}^T(t)\mathbf{H}\mathbf{W}(t)\Big)^{-1},$$

$$\sigma^2(t+1) = \frac{\mathrm{tr}(\mathbf{H} - \mathbf{H}\mathbf{W}(t)\mathbf{M}^{-1}(t)\mathbf{W}^T(t+1))}{d},$$

where $\mathbf{H}$ is defined in (4.15). Note that in (4.21) and (4.22), for notational convenience, $\mathbf{W}$ and σ^2 are used to denote the old values of the tth iteration, and $\widetilde{\mathbf{W}}$ and $\widetilde{\sigma}^2$ are for the updated new values of the $(t+1)$th iteration.

B.2 Derivation of the EM Algorithm for SPRP

According to the graphical model of SPRP, we can get

$$\begin{aligned}\ln p(\Theta \mid \mathbf{T}, \mathbf{Z}, \mathbf{X}) &= \ln p(\mathbf{T}, \mathbf{Z}, \mathbf{X}, \Theta) + c_2 \\ &= \ln p(\mathbf{T} \mid \mathbf{X}, \Theta) + \ln p(\mathbf{X}) + \ln p(\mathbf{W} \mid \mathbf{Z}) + \ln p(\mathbf{Z}) + c_2 \\ &= -\frac{1}{2\sigma^2}\Big\{\mathrm{tr}\big[(\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T)\boldsymbol{\Delta}(\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T)^T\big] - 2\mathrm{tr}\big[(\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T)\boldsymbol{\Delta}\mathbf{X}^T\mathbf{W}^T\big] \\ &\quad + \mathrm{tr}(\mathbf{W}^T\mathbf{W}\mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T)\Big\} - \frac{Nd}{2}\ln\sigma^2 - \sum_{i=1}^{d}\sum_{j=1}^{q}\frac{W_{ij}^2}{2Z_{ij}} + c_3, \end{aligned} \tag{B.6}$$

where c_2 and c_3 are constants independent of the parameters.

From (B.6), we can find that $\langle\mathbf{X}\rangle$, $\langle\mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T\rangle$ and $\langle\frac{1}{Z_{ij}}\rangle$ are the *sufficient statistics* for $\mathbf{W}$ and σ^2.

From the graphical model of SPRP, we can get

$$p(\mathbf{Z}, \mathbf{X} \mid \Theta, \mathbf{T}) = p(\mathbf{X} \mid \mathbf{T}, \Theta)p(\mathbf{Z} \mid \mathbf{W}).$$

From Bayes' rule, we get

$$\begin{aligned}p(\mathbf{X} \mid \mathbf{T}, \Theta) &= \frac{p(\mathbf{T} \mid \mathbf{X}, \Theta)p(\mathbf{X} \mid \Theta)}{p(\mathbf{T} \mid \Theta)} \\ &= \mathcal{N}_{q,N}(\mathbf{M}^{-1}\mathbf{W}^T(\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T), \sigma^2\mathbf{M}^{-1} \otimes \boldsymbol{\Phi}),\end{aligned}$$

where $\mathbf{M} = \mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I}_q$.

We can also get

$$\begin{aligned} p(Z_{ij} \,|\, W_{ij}) &= \frac{p(W_{ij} \,|\, Z_{ij}) p(Z_{ij})}{p(W_{ij})} \\ &= \sqrt{\frac{\lambda}{2\pi Z_{ij}}} \exp\left\{ -\frac{W_{ij}^2}{2Z_{ij}} - \frac{\lambda}{2} Z_{ij} + \sqrt{\lambda}\|W_{ij}\|_1 \right\}, \\ p(\mathbf{Z} \,|\, \mathbf{W}) &= \prod_{i=1}^{d}\prod_{j=1}^{q} p(Z_{ij} \,|\, W_{ij}). \end{aligned}$$

Then, the *sufficient statistics* can be computed as follows:

$$\begin{aligned} \langle \mathbf{X} \rangle &= \mathbf{M}^{-1}\mathbf{W}^T(\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T), \\ \langle \mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T \rangle &= N\sigma^2\mathbf{M}^{-1} + \langle \mathbf{X} \rangle \boldsymbol{\Delta} \langle \mathbf{X} \rangle^T, \\ \langle \frac{1}{Z_{ij}} \rangle &= \frac{\sqrt{\lambda}}{\|W_{ij}\|_1}. \end{aligned}$$

Hence,

$$\begin{aligned} Q(\Theta \,|\, \Theta(t)) = &-\frac{1}{2\sigma^2}\Big\{ \mathrm{tr}[(\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T)\boldsymbol{\Delta}(\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T)^T] - 2\mathrm{tr}[(\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T)\boldsymbol{\Delta}\langle \mathbf{X} \rangle^T \mathbf{W}^T] \\ &+ \mathrm{tr}[\mathbf{W}^T\mathbf{W}\langle \mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T \rangle] \Big\} - \frac{Nd}{2}\ln\sigma^2 - \sum_{i=1}^{d}\sum_{j=1}^{q} \langle \frac{1}{Z_{ij}} \rangle \frac{W_{ij}^2}{2}, \end{aligned}$$

where $\langle \mathbf{X} \rangle$, $\langle \mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T \rangle$ and $\langle \frac{1}{Z_{ij}} \rangle$ are computed based on $\Theta(t)$.

To compute $\mathbf{W}$, we can get $\mathrm{argmax}_{\mathbf{W}}\, Q\,(\Theta \,|\, \Theta(t)) = \mathrm{argmin}_{\mathbf{W}}\, f(\mathbf{W})$ where

$$\begin{aligned} f(\mathbf{W}) = &\mathrm{tr}\big[\mathbf{W}^T\mathbf{W}\langle \mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T \rangle\big] - 2\mathrm{tr}\big[(\mathbf{T} - \boldsymbol{\mu}\mathbf{e}^T)\boldsymbol{\Delta}\langle \mathbf{X} \rangle^T\mathbf{W}^T\big] + \sigma^2\sum_{i=1}^{d}\sum_{j=1}^{q}\langle \frac{1}{Z_{ij}} \rangle W_{ij}^2 \\ = &\sum_{i=1}^{d}\big\{ \mathbf{W}_{i*}[\langle \mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T \rangle + \sigma^2\boldsymbol{\Lambda}_i]\mathbf{W}_{i*}^T - 2(\mathbf{T}_{i*} - \mu_i\mathbf{e}^T)\boldsymbol{\Delta}\langle \mathbf{X} \rangle^T \mathbf{W}_{i*}^T \big\}, \end{aligned}$$

with

$$\boldsymbol{\Lambda}_i = \mathrm{diag}\big(\langle \frac{1}{Z_{i1}} \rangle, \cdots, \langle \frac{1}{Z_{iq}} \rangle\big).$$

Hence, in the M-step, we can update $\mathbf{W}$ as follows:

$$\widetilde{\mathbf{W}}_{i*} = (\mathbf{T}_{i*} - \mu_i\mathbf{e}^T)\boldsymbol{\Delta}\langle \mathbf{X} \rangle^T[\langle \mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T \rangle + \sigma^2\boldsymbol{\Lambda}_i]^{-1}.$$

Note that when W_{ij} goes to zero, the term $\frac{\sqrt{\lambda}}{\|W_{ij}\|_1}$ in $\boldsymbol{\Lambda}_i$ will become arbitrarily large and hence will make the computation unstable. To overcome this computational problem, we rewrite the updating of $\mathbf{W}$ as follows:

$$\widetilde{\mathbf{W}}_{i*} = (\mathbf{T}_{i*} - \mu_i\mathbf{e}^T)\boldsymbol{\Delta}\langle \mathbf{X} \rangle^T\boldsymbol{\Sigma}_i[\langle \mathbf{X}\boldsymbol{\Delta}\mathbf{X}^T \rangle\boldsymbol{\Sigma}_i + \sigma^2\mathbf{I}_q]^{-1},$$

where $\mathbf{\Sigma}_i = \mathbf{\Lambda}_i^{-1} = \mathrm{diag}\big(\frac{\|W_{i1}\|_1}{\sqrt{\lambda}}, \cdots, \frac{\|W_{iq}\|_1}{\sqrt{\lambda}}\big)$.

We can further rewrite $\widetilde{\mathbf{W}}$ as

$$\widetilde{\mathbf{W}}_{i*} = \mathbf{H}_{i*}\mathbf{W}\mathbf{M}^{-1}\mathbf{\Sigma}_i\big[(\sigma^2\mathbf{I}_q + \mathbf{M}^{-1}\mathbf{W}^T\mathbf{H}\mathbf{W})\mathbf{M}^{-1}\mathbf{\Sigma}_i + \frac{\sigma^2}{N}\mathbf{I}_q\big]^{-1}.$$

By setting the gradient of $Q(\Theta \,|\, \Theta(t))$ with respect to σ^2 to zero, σ^2 can be updated as follows:

$$\widetilde{\sigma}^2 = \frac{\mathrm{tr}[\mathbf{H} - \mathbf{H}\mathbf{W}\mathbf{M}^{-1}\widetilde{\mathbf{W}}^T]}{d}.$$