Latent Wishart Processes for Relational Kernel Learning

Wu-Jun Li
Dept. of Comp. Sci. and Eng.
Hong Kong Univ. of Sci. and Tech.
Hong Kong, China
liwujun@cse.ust.hk

Zhihua Zhang
College of Comp. Sci. and Tech.
Zhejiang University
Zhejiang 310027, China
zhzhang@cs.zju.edu.cn

Dit-Yan Yeung
Dept. of Comp. Sci. and Eng.
Hong Kong Univ. of Sci. and Tech.
Hong Kong, China
dyyeung@cse.ust.hk

Abstract

One main concern towards kernel classifiers is on their sensitivity to the choice of kernel function or kernel matrix which characterizes the similarity between instances. Many real-world data, such as web pages and protein-protein interaction data, are relational in nature in the sense that different instances are correlated (linked) with each other. The relational information available in such data often provides strong hints on the correlation (or similarity) between instances. In this paper, we propose a novel relational kernel learning model based on *latent Wishart processes* (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.

1 Introduction

Kernel methods, such as support vector machines (SVM) and Gaussian processes (GP) (Rasmussen and 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 real-world applications, relationships or "links" between (some) instances may also be available in the data in addition to the input attributes. Data of this sort, referred to as *relational data* (Getoor and Taskar, 2007), can be found in such diverse application areas as web mining, social network analysis, bioinformatics, marketing, and so on. In relational data, the attributes of connected (linked) instances are often correlated and the class label of one instance may have an influence on that of a linked instance. This means that the relationships (or links) between instances are very informative for instance classification (Getoor and Taskar, 2007), sometimes even much more informative than input attributes. For example, two hyperlinked web pages are very likely to be related to the same topic, even when their attributes may look quite different when represented as bags of words. In biology, interacting proteins are more likely to have the same biological function than those without interaction. In marketing, knowing one's shopping habit will provide useful information about his/her close friends' shopping inclinations. Hence, in such data, 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.

Thanks to its promising potential in many application areas, *relational learning* (Getoor and Taskar, 2007), which tries to model relational data, has become a hot topic in the machine learning community. Many methods have been proposed over the past few years. For example, probabilistic relational models

Appearing in Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS) 2009, Clearwater Beach, Florida, USA. Volume 5 of JMLR: W&CP 5. Copyright 2009 by the authors.

(PRM) (Getoor and Taskar, 2007), such as relational Bayesian networks (RBN) (Getoor et al., 2002), relational Markov networks (RMN) (Taskar et al., 2002) and relational dependency networks (RDN) (Neville and Jensen, 2007), try to model the relational data by adapting conventional graphical models for the relational scenario. There are also some methods that handle relational data in heuristic ways (Macskassy and Provost, 2007). These methods 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 paper.

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 (Silva et al., 2008), 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.

The rest of this paper is organized as follows. Section 2 introduces some basic background for Wishart processes. Section 3 presents our LWP model and Section 4 compares it with some related works. Experimental results are presented in Section 5 and then finally Section 6 concludes the paper.

2 Wishart Processes

Definition 1 (Gupta and Nagar, 2000) *An $n \times n$ random symmetric positive definite matrix $\mathbf{A}$ is said to have a* ***Wishart distribution*** *with parameters n, q, and $n \times n$ scale matrix $\mathbf{\Sigma} \succ 0$, written as $\mathbf{A} \sim W_n(q, \mathbf{\Sigma})$, if its p.d.f. is given by*

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

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

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

Definition 2 (Zhang et al., 2006) *The kernel function $\{A(\mathbf{x}_i, \mathbf{x}_j); \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}\}$ is said to be a* ***Wishart process*** *(WP) if for any $n \in \mathbb{N}$ and $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq \mathcal{X}$, the $n \times n$ random matrix $\mathbf{A} = [A(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^n$ follows a Wishart distribution.*

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

Although q is possibly infinite, we assume that it is finite in this paper for simplicity. Denote $\mathbf{A} = [A(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^n$ $(n \times n)$ and $\mathbf{B} = [B(\mathbf{x}_1), \ldots, B(\mathbf{x}_n)]' = [\mathbf{b}_1, \ldots, \mathbf{b}_n]'$ $(n \times q)$. Then the $\mathbf{b}_i$ are the latent vectors, and $\mathbf{A} = \mathbf{B}\mathbf{B}'$ is the linear kernel in the latent space but is a nonlinear kernel w.r.t. the input space.

Let $\mathbf{\Sigma} \otimes \mathbf{I}_q$ denote the Kronecker product of $\mathbf{\Sigma}$ and $\mathbf{I}_q$ (the $q \times q$ identity matrix). Using the notation $N_{n,q}(\mathbf{0}, \mathbf{\Sigma} \otimes \mathbf{I}_q)$ in (Gupta and Nagar, 2000, page 55) for matrix-variate normal distributions, we have

Theorem 1 *Let $\mathbf{\Sigma}$ be an $n \times n$ positive definite matrix. Then $\mathbf{A}$ is distributed according to the Wishart distribution $W_n(q, \mathbf{\Sigma})$ if and only if $\mathbf{B}$ is distributed according to the matrix-variate normal distribution $N_{n,q}(\mathbf{0}, \mathbf{\Sigma} \otimes \mathbf{I}_q)$.*

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

If $q < n$, $\mathbf{A}$ 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 and Nagar, 2000)). The proof is similar to the case of

$q \geq n$. We omit the details here due to the page limit constraint. □

Theorem 1 shows an interesting connection; namely, the degree of freedom q in the Wishart process is the dimensionality of the latent space $\mathcal{F}$. Theorem 1 extends the results given in (Gupta and Nagar, 2000) and (Zhang et al., 2006) where the condition $q \geq n$ is required. Moreover, the asymptotic distribution of $q^{1/2}(\mathbf{A} - \mathbf{\Sigma})$ as $q \to \infty$ is $N_{n,n}(\mathbf{0}, (\mathbf{I}_{n^2} + \mathbf{C})(\mathbf{\Sigma} \otimes \mathbf{\Sigma}))$, where $\mathbf{C}$ is the commutation matrix (Muirhead, 1982).

3 Methodology

One common representation format for relational data is a graph, in which a node corresponds to an instance and an edge corresponds to a pairwise relationship between the two connected nodes. Although directed edges are common in some data sets, they can be converted into undirected edges in many cases. In this paper, we only focus on modelling undirected edges 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. Due to the page limit constraint, 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 graphs or hypergraphs with negative relationships will be pursued in our future work.

Suppose we are given a set of data $\{(\mathbf{x}_i, y_i, z_{ik}) : i, k = 1, \ldots, n\}$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})'$ and y_i are respectively the input feature vector and the label for instance i, and z_{ik} is a binary variable indicating the existence of a relationship (link) between instances i and k, namely,

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

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 binary variables $\{z_{ik}\}$ to learn the kernel. Thus, essence, LWP is *unsupervised* in nature.

3.1 Model

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

Let $K(\mathbf{x}_i, \mathbf{x}_k)$ be a kernel function just defined on the input attributes. For example, the linear kernel $K(\mathbf{x}_i, \mathbf{x}_k) = \mathbf{x}_i'\mathbf{x}_k$ is such a kernel function. Similarly, $\mathbf{K} = [K(\mathbf{x}_i, \mathbf{x}_k)]_{i,k=1}^n$ 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

$$\mathbf{A} \sim W_n(q, \beta(\mathbf{K} + \lambda \mathbf{I})). \tag{2}$$

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

To incorporate the relational information for the learning of $A(\cdot, \cdot)$, we regard each z_{ik} as a Bernoulli variable, which is determined by the latent variable a_{ik} via a logistic link function s_{ik}. Given the a_{ik}, we further assume that the z_{ik} are independent. Moreover, we assume that the links are symmetric with no self loops, i.e., $z_{ik} = z_{ki}$ and $z_{ii} = 0$. Letting $\mathbf{Z} = [z_{ik}]_{i,k=1}^n$, we have

$$p(\mathbf{Z}|\mathbf{A}) = \prod_{i=1}^{n} \prod_{k=i+1}^{n} s_{ik}^{z_{ik}} (1 - s_{ik})^{1 - z_{ik}}$$
$$\text{with} \quad s_{ik} = \frac{\exp(a_{ik}/2)}{1 + \exp(a_{ik}/2)}. \tag{3}$$

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 *maximum a posteriori* (MAP) estimation can be used to learn the latent variables a_{ik}. After learning the model, the target kernel function $A(\mathbf{x}_i, \mathbf{x}_k)$ 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 $A(\mathbf{x}_i, \mathbf{x}_k)$ as a kernel function for a GP-based classifier.

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

3.2 Learning

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

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

Theorem 1 shows that finding the MAP estimate of $\mathbf{A}$ is equivalent to finding the MAP estimate of $\mathbf{B}$. In particular, we note that for the applications presented in Section 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:

$$\begin{aligned} L(\mathbf{B}) &= \log\{p(\mathbf{Z}|\mathbf{B})p(\mathbf{B})\} \\ &= \sum_{i\neq k} \log p(z_{ik}|\mathbf{b}_i, \mathbf{b}_k) + \log p(\mathbf{B}) \\ &= \sum_{i\neq k} \Big[z_{ik}\mathbf{b}_i'\mathbf{b}_k/2 - \log(1+\exp(\mathbf{b}_i'\mathbf{b}_k/2))\Big] \\ &\quad -\frac{1}{2}\mathrm{tr}\Big[\frac{(\mathbf{K}+\lambda\mathbf{I})^{-1}}{\beta}\mathbf{B}\mathbf{B}'\Big] + C \\ &= \sum_{i\neq k} \Big[z_{ik}\mathbf{b}_i'\mathbf{b}_k/2 - \log(1+\exp(\mathbf{b}_i'\mathbf{b}_k/2))\Big] \\ &\quad -\frac{1}{2}\sum_{i,k}\sigma_{ik}\mathbf{b}_i'\mathbf{b}_k + C, \end{aligned}$$

where $[\sigma_{ik}]_{i,k=1}^n = \frac{(\mathbf{K}+\lambda\mathbf{I})^{-1}}{\beta}$ 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 $\frac{\partial^2 L}{\partial\mathbf{B}\partial\mathbf{B}'}$ by using the block diagonal matrix Block diag$\Big(\frac{\partial^2 L}{\partial\mathbf{b}_1\partial\mathbf{b}_1'}, \ldots, \frac{\partial^2 L}{\partial\mathbf{b}_n\partial\mathbf{b}_n'}\Big)$.

The Fisher score vector and Hessian matrix of L w.r.t. $\mathbf{b}_i$ are given by

$$\frac{\partial L}{\partial \mathbf{b}_i} = \sum_{j\neq i}(z_{ij} - s_{ij} - \sigma_{ij})\mathbf{b}_j - \sigma_{ii}\mathbf{b}_i$$

$$\frac{\partial^2 L}{\partial\mathbf{b}_i\partial\mathbf{b}_i'} = -\frac{1}{2}\sum_{j\neq i} s_{ij}(1-s_{ij})\mathbf{b}_j\mathbf{b}_j' - \sigma_{ii}\mathbf{I}_q \;\triangleq\; -\mathbf{H}_i.$$

Given the initial values $\mathbf{b}_i(0)$, the update equations for the $\mathbf{b}_i$ are

$$\mathbf{b}_i(t{+}1){=}\mathbf{b}_i(t){+}\gamma\cdot\mathbf{H}_i(t)^{-1}\frac{\partial L}{\partial\mathbf{b}_i}\Big|_{\mathbf{B}=\mathbf{B}(t)}, \quad i = 1,\ldots,n, \tag{4}$$

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. In this paper, we adopt KPCA (Schölkopf et al., 1998) to initialize the values $\mathbf{b}_i(0)$.

3.3 Out-of-Sample Extension

It is easy for LWP to perform out-of-sample extension (or induction), which means that we can use the learned LWP to predict the $\mathbf{b}_i$ for new test data.

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

$$\mathbf{B}_2\,|\,\mathbf{B}_1 \sim N_{n_2,q}\big(\mathbf{\Sigma}_{21}\mathbf{\Sigma}_{11}^{-1}\mathbf{B}_1,\ \mathbf{\Sigma}_{22\cdot1}\otimes\mathbf{I}_q\big), \tag{5}$$

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

For inductive inference, we first find the MAP estimate of $\mathbf{B}_1$ based on $\operatorname{argmax}_{\mathbf{B}_1}\log p(\mathbf{Z}_{11}|\mathbf{B}_1)p(\mathbf{B}_1)$ 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 $\mathbf{\Sigma}_{21}\mathbf{\Sigma}_{11}^{-1}\mathbf{B}_1$ based on (5).

Owing to the page limit, out-of-sample extension will not be evaluated via experiments in this paper. We will report the experiments in our future work.

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 $\mathbf{A} = \mathbf{B}\mathbf{B}'$ as the learned kernel matrix, which can be further used to train all kinds of kernel classifiers. In RGP and XGP, however, $p(\mathbf{B}|\mathbf{Z})$ 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 $p(\mathbf{B}|\mathbf{Z})$, 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 $p(\mathbf{B}|\mathbf{Z})$, RGP and XGP adopt different approximation strategies to compute the posterior covariance matrix.

Another related work is the stochastic relational model in (Yu and Chu, 2008; Yu et al., 2006), where $\mathbf{A}$ is modeled as a tensor GP rather than a WP. Since $\mathbf{A}$ in (Yu and Chu, 2008; 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 and Chu, 2008; Yu et al., 2006) 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 $d_{ij}^2 = \|\mathbf{b}_i - \mathbf{b}_j\|^2$ be the squared distance between $\mathbf{b}_i$ and $\mathbf{b}_j$, and $\mathbf{D} = [d_{ij}^2]$ be the $n\times n$ distance matrix. Then $-\frac{1}{2}\mathbf{EDE} = \mathbf{EAE}$ where $\mathbf{E}$ is an $n\times n$ centering matrix. This reveals a connection between our model and the latent distance model in (Hoff, 2008). In addition, we also note that in the latent eigenmodel (Hoff, 2008) for symmetric relational data, Hoff (2008) defined $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}'$ where $\mathbf{U}$ is an orthogonal matrix and $\mathbf{\Lambda}$ is a diagonal matrix but its diagonal elements can be negative. Thus, $\mathbf{A}$ does not play the role of a kernel matrix.

5 Experiments

We compare our LWP method with several related methods, such as the standard Gaussian process classifier (GPC) (Rasmussen and Williams, 2006), RGP and XGP, on three real-world data sets, WebKB (Craven et al., 1998), Cora (McCallum et al., 2000) and political books data set, which are also used in (Silva et al., 2008). The centralized linear kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i'\mathbf{x}_j$ (Chu et al., 2007) is used to define the covariance matrix $\mathbf{K}$ in the Wishart distribution defined in (2) for all these data sets. The λ in (2) is set to a small number 10^{-4}. All the $\mathbf{b}_i$ are *initialized* to the feature vectors obtained by kernel principal component analysis (KPCA) [1] (Schölkopf et al., 1998) based on $\mathbf{K} + \lambda\mathbf{I}$.

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.

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 (2), the γ in (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 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 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 the political books data set. Figure 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

[1]The KPCA in our paper 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.

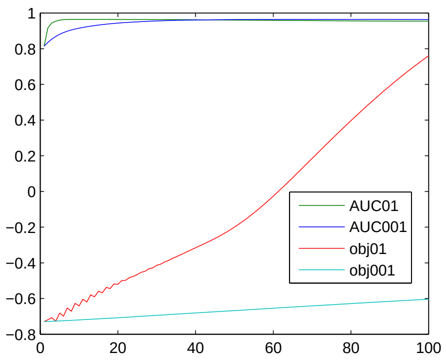

Figure 1: The performance (AUC) and the objective function against the change in the number of iterations.

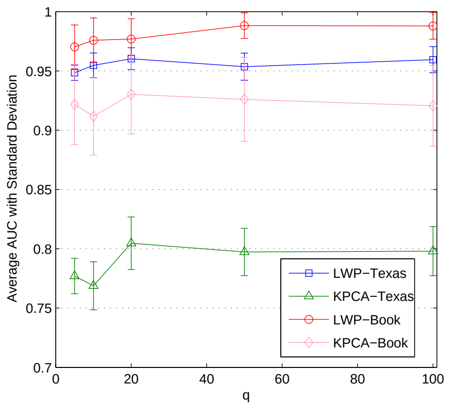

Figure 2: Performance of LWP against the change in degree of freedom q in the Wishart process (i.e., dimensionality of the latent space).

LWP is set to 0. From Figure 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 $\beta = 1000, \gamma = 0.01, T = 10, q = 20$.

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 visualization, q is set to 2.

[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.

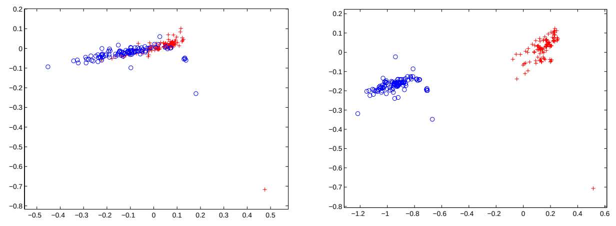

Figure 3: Visualization of data points in the transformed space by KPCA and in the latent space learned by LWP. 100 positive (red cross) and 100 negative (blue circle) examples are shown.

After learning, 100 positive and 100 negative examples are randomly chosen for visualization. The results are demonstrated in Figure 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.

5.3 Performance on WebKB Data Set

A subset of the WebKB data set (Craven et al., 1998), which was collected from the web sites of the computer science departments of four universities, is used to test our method. Each webpage is labeled with one out of seven categories: student, professor, course, project, staff, department, and “other”. 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, area under the ROC curve (AUC), and the same evaluation strategy as those in (Silva et al., 2008) are adopted for all the methods, i.e., for a specific 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 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.

5.4 Performance on Cora Data Set

A subset of the Cora corpus (McCallum et al., 2000) including 4,285 machine learning papers with their bibliographic citations is used for this experiment. All the papers in this subset are partitioned into 7 classes, the information about which is shown in the second column of Table 2. We adopt the same feature representation scheme as that in (Silva et al., 2008), where each paper 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 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.

5.5 Performance on Political Books Data Set

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 and is preprocessed by the same techniques as for the WebKB data set. 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. 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 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 $\mathbf{b}_i$ in LWP to perform classification.

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

Table 3: Experiment on the data set of political books. Results for GPC, RGP and XGP are taken from (Silva et al., 2008).

GPC	RGP	XGP	KPCA	LWP
0.92	**0.98**	**0.98**	0.93 ± 0.03	$\mathbf{0.98} \pm \mathbf{0.02}$

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 performs relatively badly on the WebKB data set. On the other hand, XGP achieves unsatisfactory performance on the Cora data set. This implies that LWP is much more robust than RGP and XGP. 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 Concluding Remarks

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.

In our future work, we will focus on extensive empirical comparison of LWP with other related methods on the aspect of inductive inference. Moreover, it is also desirable to apply our model to social network analysis.

Acknowledgements

We thank Ricardo Silva for sharing the data and code of XGP and providing useful help for our experiments. We also thank Wei Chu and Vikas Sindhvani for the preprocessed Cora data set. This research has been supported by General Research Fund 621407 from the Research Grants Council of the Hong Kong Special Administrative Region, China.

Table 1: Mean and standard deviation of AUC over 100 rounds of test on the WebKB data set. Results for other related methods are taken from (Silva et al., 2008). 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 linkages 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

Table 2: Mean and standard deviation of AUC over 100 rounds of test on the Cora data set. Results for other related methods are taken from (Silva et al., 2008). 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 linkages 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**

References

Chu, W., V. Sindhwani, Z. Ghahramani, and S. S. Keerthi (2007). Relational learning with gaussian processes. In *Advances in Neural Information Processing Systems 19*, pp. 289–296.

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

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

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

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

Hoff, P. D. (2008). Modeling homophily and stochastic equivalence in symmetric relational data. In J. Platt, D. Koller, Y. Singer, and S. Roweis (Eds.), *Advances in Neural Information Processing Systems 20.* Cambridge, MA: MIT Press.

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

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

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

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

Muirhead, R. J. (1982). *Aspects of Multivariate Statistical Theory.* New York: John Wiley and Sons.

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

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

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

Silva, R., W. Chu, and Z. Ghahramani (2008). Hidden common cause relations in relational learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis (Eds.), *Advances in Neural Information Processing Systems 20*, pp. 1345–1352. Cambridge, MA: MIT Press.

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

Taskar, B., P. Abbeel, and D. Koller (2002). Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence*, pp. 485–492.

Yu, K. and W. Chu (2008). Gaussian process models for link analysis and transfer learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis (Eds.), *Advances in Neural Information Processing Systems 20*, pp. 1657–1664. Cambridge, MA: MIT Press.

Yu, K., W. Chu, S. Yu, V. Tresp, and Z. Xu (2006). Stochastic relational models for discriminative link prediction. In *Advances in Neural Information Processing Systems 19*, pp. 1553–1560.

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