Relation Regularized Matrix Factorization

Wu-Jun Li and Dit-Yan Yeung
Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong, China
{liwujun, dyyeung}@cse.ust.hk

Abstract

In many applications, the data, such as web pages and research papers, contain relation (link) structure among entities in addition to textual content information. Matrix factorization (MF) methods, such as latent semantic indexing (LSI), have been successfully used to map either content information or relation information into a lower-dimensional latent space for subsequent processing. However, how to simultaneously model both the relation information and the content information effectively with an MF framework is still an open research problem. In this paper, we propose a novel MF method called *relation regularized matrix factorization* (RRMF) for relational data analysis. By using relation information to regularize the content MF procedure, RRMF seamlessly integrates both the relation 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 real data sets show that RRMF can achieve state-of-the-art performance.

1 Introduction

Matrix factorization (MF) methods [Singh and Gordon, 2008], which try to project objects (entities) into a lower-dimensional latent space, have been widely used for various data analysis applications.[1] One of the most popular MF methods is latent semantic indexing (LSI) [Deerwester *et al.*, 1990], which uses singular value decomposition (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 and Srebro, 2005], which can be seen as a regularized version of SVD by controlling the complexity of the factors,[2] has been successfully used for many applications, such as collaborative filtering [Rennie and Srebro, 2005]. Many other variants of MF methods and their applications can be found in [Singh and Gordon, 2008].

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 objects (entities), or the link structure (*relation information*) among a set of objects. 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 relationship among a set of entities [Rennie and Srebro, 2005].

However, the data in many applications, such as web pages and research papers, contain both textual content information and relation structure. These two kinds of information often complement each other because they are typically collected at different semantic levels. For example, a citation/reference relation 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 relation among entities. 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.

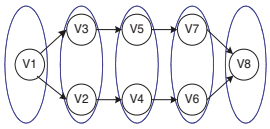
In [Zhu *et al.*, 2007], a joint link-content MF method, abbreviated as LCMF here, was proposed to seamlessly integrate content and relation information into an MF framework through a set of shared factors for the factorization of both content and link structures. The authors of LCMF argue that their method is more reasonable than some other simple methods for combining content and link information, such as that in [Kurland and Lee, 2005], which seek to convert

[1] Due to the page limit constraint, many related MF references are not cited in this paper. We refer the readers to [Singh and Gordon, 2008] for many such references.

[2] 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 and Srebro, 2005] and squared loss. Here we refer in particular to the squared loss function.

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, 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 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 1(a). The learned latent factors of the entities [Zhu *et al.*, 2007] are shown in Figure 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. 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].



(a) Synthetic link structure

-.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

(b) Result of link MF

Figure 1: A synthetic example of link structure from [Zhu *et al.*, 2007] for illustration.

However, one problem with LCMF is that the factor representations of two linked entities are not guaranteed to be similar. 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. 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 relation 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. In this paper, we call the first type of links, such as those in the WebKB data set, *Type I links*, and the other type of links, such as those in the Cora data set, *Type II links*. As discussed above, LCMF can handle well applications with Type I links but not those with Type II links. It is this limitation that has motivated us to pursue research reported in this paper.

We propose in this paper 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 relation information to regularize the content MF procedure, resulting in a principled framework which seamlessly integrates content and relation 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 Type II links demonstrate that RRMF can dramatically outperform LCMF and other state-of-the-art methods.

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

2 Relation Regularized Matrix Factorization

2.1 Notations

We use boldface uppercase letters, such as $\mathbf{K}$, to denote matrices, and boldface lowercase letters, such as $\mathbf{z}$, to denote vectors. The ith row and the jth column of a matrix $\mathbf{K}$ are denoted as $\mathbf{K}_{i*}$ and $\mathbf{K}_{*j}$, respectively. K_{ij} denotes the element at the ith row and jth column in $\mathbf{K}$. z_i denotes the ith element in $\mathbf{z}$. $\mathbf{X}$ denotes the content matrix of size $n \times m$, with n being the number of entities and m the number of features. $\mathbf{A}$ is the adjacency matrix of the n entities. D denotes the number of latent factors. $\mathbf{I}$ denotes the identity matrix whose dimensionality depends on the context. For a matrix $\mathbf{K}$, $\mathbf{K} \succeq 0$ means that $\mathbf{K}$ is positive semi-definite (psd) and $\mathbf{K} \succ 0$ means that $\mathbf{K}$ is positive definite (pd).

2.2 Model Formulation

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

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

where we use an $n \times D$ matrix $\mathbf{U}$ and an $m \times D$ matrix $\mathbf{V}$ to denote the latent D-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 relation (link) information into the MF procedure, we use the relations 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 relation between them. We can achieve this goal by

minimizing the following objective function:

$$\begin{aligned} l &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}\|\mathbf{U}_{i*}-\mathbf{U}_{j*}\|^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left[A_{ij}\sum_{d=1}^{D}(U_{id}-U_{jd})^2\right] \\ &= \frac{1}{2}\sum_{d=1}^{D}\left[\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}(U_{id}-U_{jd})^2\right] \\ &= \sum_{d=1}^{D}\mathbf{U}_{*d}^T\boldsymbol{\mathcal{L}}\mathbf{U}_{*d} = \mathrm{tr}(\mathbf{U}^T\boldsymbol{\mathcal{L}}\mathbf{U}), \end{aligned} \quad (2)$$

where $A_{ij} = 1$ if there is a relation between entities i and j, and otherwise $A_{ij} = 0$, and $\boldsymbol{\mathcal{L}} = \mathbf{D} - \mathbf{A}$ 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 (1) and (2), we obtain the following objective function which we seek to minimize during learning:

$$\begin{aligned} f &= \frac{1}{2}\|\mathbf{X}-\mathbf{U}\mathbf{V}^T\|^2 + \frac{\alpha}{2}(\|\mathbf{U}\|^2+\|\mathbf{V}\|^2) + \frac{\beta}{2}\mathrm{tr}(\mathbf{U}^T\boldsymbol{\mathcal{L}}\mathbf{U}) \\ &= \frac{1}{2}\|\mathbf{X}-\mathbf{U}\mathbf{V}^T\|^2 + \frac{1}{2}\mathrm{tr}[\mathbf{U}^T(\alpha\mathbf{I}+\beta\boldsymbol{\mathcal{L}})\mathbf{U}] + \frac{\alpha}{2}\mathrm{tr}(\mathbf{V}\mathbf{V}^T), \end{aligned} \quad (3)$$

which seamlessly integrates the content information and the relation information into a principled framework. Because we use relation information to regularize the content MF procedure, we call the model in (3) relation regularized matrix factorization, abbreviated as RRMF.

Remark 1 *The normalized Laplacian matrix [Chung, 1997], given by* $\tilde{\boldsymbol{\mathcal{L}}} = \mathbf{D}^{-1/2}\boldsymbol{\mathcal{L}}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, *can be used to substitute* $\boldsymbol{\mathcal{L}}$ *in (2) and (3) for regularization. If* $\tilde{\boldsymbol{\mathcal{L}}}$ *is adopted, the objective function in (2) will be* $\tilde{l} = \frac{1}{2}\sum_{d=1}^{D}\left[\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}\left(\frac{U_{id}}{\sqrt{D_{ii}}} - \frac{U_{jd}}{\sqrt{D_{jj}}}\right)^2\right]$. *The choice between* $\boldsymbol{\mathcal{L}}$ *and* $\tilde{\boldsymbol{\mathcal{L}}}$ *depends on applications. In this paper, 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), it is easy to see that the model assumption behind RRMF is in line with the semantics of Type II links. Hence, RRMF can be expected to achieve good performance for applications with Type II links, such as research paper classification, which will be verified by experiment in Section 3.

On the other hand, RRMF can also be used for applications with Type I 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 Type I 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 Type II links. For example, in Figure 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.

2.3 Learning

In this subsection, we first prove that the objective function in (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 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 2 $\sum_{j=1}^{m}\mathbf{V}_{j*}^T\mathbf{V}_{j*} \succeq 0$.

Proof: For any $D\times 1$ vector $\mathbf{z} \neq 0$, $\mathbf{z}^T\left(\sum_{j=1}^{m}\mathbf{V}_{j*}^T\mathbf{V}_{j*}\right)\mathbf{z} = \sum_{j=1}^{m}\mathbf{z}^T\mathbf{V}_{j*}^T\mathbf{V}_{j*}\mathbf{z} = \sum_{j=1}^{m}(\mathbf{V}_{j*}\mathbf{z})^2 \geq 0$. ■

Theorem 1 *f is convex w.r.t.* $\mathbf{U}$.

Proof: We first rewrite f as follows: $f = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}(X_{ij} - \mathbf{U}_{i*}\mathbf{V}_{j*}^T)^2 + \frac{1}{2}\sum_{d=1}^{D}\mathbf{U}_{*d}^T(\alpha\mathbf{I} + \beta\boldsymbol{\mathcal{L}})\mathbf{U}_{*d} + C_1$, where C_1 is a constant independent of $\mathbf{U}$.

Let $g = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}(X_{ij} - \mathbf{U}_{i*}\mathbf{V}_{j*}^T)^2 = C_2 + \frac{1}{2}\sum_{i=1}^{n}\left[\mathbf{U}_{i*}\left(\sum_{j=1}^{m}\mathbf{V}_{j*}^T\mathbf{V}_{j*}\right)\mathbf{U}_{i*}^T - 2\left(\sum_{j=1}^{m}X_{ij}\mathbf{V}_{j*}\right)\mathbf{U}_{i*}^T\right]$, where C_2 is a constant independent of $\mathbf{U}$. It is easy to see that $\mathbf{G}_i \triangleq \frac{\partial^2 g}{\partial\mathbf{U}_{i*}^T\partial\mathbf{U}_{i*}} = \sum_{j=1}^{m}\mathbf{V}_{j*}^T\mathbf{V}_{j*}$, and $\frac{\partial^2 g}{\partial\mathbf{U}_{i*}^T\partial\mathbf{U}_{k*}} = \mathbf{0}$ (if $i \neq k$). 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 2, $\det(\mathbf{G}_i) \geq 0$, where $\det(\cdot)$ denotes the determinant of a matrix. Because $\det(\mathbf{G}_{\mathbf{u}}) = \prod_{i=1}^{n}\det(\mathbf{G}_i) \geq 0$, we can conclude that g is convex w.r.t. $\mathbf{U}$.

Let $h = \frac{1}{2}\sum_{d=1}^{D}\mathbf{U}_{*d}(\alpha\mathbf{I} + \beta\boldsymbol{\mathcal{L}})\mathbf{U}_{*d}^T$. If we use $\mathbf{a} = (\mathbf{U}_{*1}^T, \mathbf{U}_{*2}^T, \ldots, \mathbf{U}_{*D}^T)^T$ to denote another vector representation of $\mathbf{U}$, the Hessian of h w.r.t. $\mathbf{a}$ will also be a block-diagonal matrix: $\mathbf{H}_{\mathbf{a}} \triangleq \frac{\partial^2 h}{\partial\mathbf{a}\partial\mathbf{a}^T} = \mathrm{diag}[\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_D]$, where $\mathbf{H}_d$ is defined as follows: $\mathbf{H}_d \triangleq \frac{\partial^2 h}{\partial\mathbf{U}_{*d}\partial\mathbf{U}_{*d}^T} = \alpha\mathbf{I} + \beta\boldsymbol{\mathcal{L}}$. According to Lemma 1, we can conclude that $\det(\mathbf{H}_{\mathbf{a}}) > 0$, and hence h is convex w.r.t. $\mathbf{U}$.

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

Theorem 2 *f is convex w.r.t.* $\mathbf{V}$.

Proof: We first rewrite f as follows: $f = C_3 + \frac{1}{2}\sum_{j=1}^{m}[\mathbf{V}_{j*}\left(\sum_{i=1}^{n}\mathbf{U}_{i*}^T\mathbf{U}_{i*} + \alpha\mathbf{I}\right)\mathbf{V}_{j*}^T - 2\left(\sum_{i=1}^{n}X_{ij}\mathbf{U}_{i*}\right)\mathbf{V}_{j*}^T]$, where C_3 is a constant independent of $\mathbf{V}$. If we use $\mathbf{v} = (\mathbf{V}_{1*}, \mathbf{V}_{2*}, \ldots, \mathbf{V}_{m*})^T$

to denote the vector representation of $\mathbf{V}$, the Hessian of f w.r.t. $\mathbf{v}$ will be a block-diagonal matrix: $\mathbf{K}_{\mathbf{v}} \triangleq \frac{\partial^2 f}{\partial \mathbf{v} \partial \mathbf{v}^T} = \text{diag}[\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_m]$, where $\mathbf{K}_j = \sum_{i=1}^{n} \mathbf{U}_{i*}^T \mathbf{U}_{i*} + \alpha \mathbf{I}$. It is easy to verify that $\mathbf{K}_j \succ 0$ 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 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 $nD \times nD$. In this paper, we adopt an alternative strategy to perform optimization on $\mathbf{U}$, which is to optimize one column $\mathbf{U}_{*d}$ 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. $\mathbf{U}_{*d}$ with all other variables fixed. Hence, computing the gradient of f w.r.t. $\mathbf{U}_{*d}$ and setting it to 0, we can optimize $\mathbf{U}_{*d}$ by solving the following linear system:

$$\mathbf{F}^{(d)} \mathbf{U}_{*d} = \mathbf{e}^{(d)}, \tag{4}$$

where $\mathbf{F}^{(d)} = \mathbf{E}^{(d)} + \alpha \mathbf{I} + \beta \boldsymbol{\mathcal{L}}$, and $\mathbf{E}^{(d)} = \text{diag}(\frac{\partial^2 g}{\partial U_{1d} \partial U_{1d}}, \frac{\partial^2 g}{\partial U_{2d} \partial U_{2d}}, \ldots, \frac{\partial^2 g}{\partial U_{nd} \partial U_{nd}})$ with $\frac{\partial^2 g}{\partial U_{id} \partial U_{id}} = \sum_{j=1}^{m} V_{jd}^2$, and $\mathbf{e}^{(d)} = (e_1^{(d)}, e_2^{(d)}, \ldots, e_n^{(d)})^T$ with $e_i^{(d)} = \sum_{j=1}^{m} V_{jd}(X_{ij} - \mathbf{U}_{i*} \mathbf{V}_{j*}^T + U_{id} V_{jd})$.

One direct way to solve the linear system in (4) is to set $\mathbf{U}_{*d} = [\mathbf{F}^{(d)}]^{-1} \mathbf{e}^{(d)}$. 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 $\mathbf{U}_{*d}$: $r(t) = \mathbf{e}^{(d)} - \mathbf{F}^{(d)} \mathbf{U}_{*d}(t)$, $\delta(t) = \frac{r(t)^T r(t)}{r(t)^T \mathbf{F}^{(d)} r(t)}$, $\mathbf{U}_{*d}(t+1) = \mathbf{U}_{*d}(t) + \delta(t) r(t)$.

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 $\mathbf{V}_{j*}$.

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

2.4 Convergence and Complexity Analysis

Theorem 3 *The learning algorithm will converge.*

Proof: (Sketch) In each iteration, the learning algorithm ensures that the objective function value in (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 Experiments

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 1.

Table 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

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 each paper into one of the subfields of data structure (DS), hardware and architecture (HA), machine learning

Table 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

(ML), and programming language (PL). The characteristics of the Cora data set are summarized in Table 2.

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 D is set to 50 for RRMF. The α in (3) is fixed to 1, and β is specified by cross-validation on the training data. After obtaining the factors, a linear support vector machine (SVM) 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.

3.2 Baselines

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

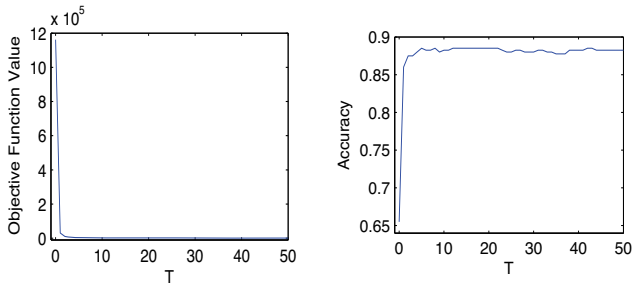
- SVM on content: This method ignores the link structure in the data, and applies SVM only on the content information in the original bag-of-words representation.
- SVM on links: This method ignores the content information, and treats links as the features, i.e, the ith feature is *link-to-page*$_i$.
- 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 and 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 principal component analysis (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) by setting $\beta = 0$. It is used to show that the relation 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 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 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 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 2: Convergence properties of RRMF.

3.4 Performance

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, 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 4. It should be noted that the WebKB data set does not satisfy the model assumption of RRMF. However, with simple preprocessing, RRMF can still achieve performance comparable to the link-content MF method and its supervised counterpart, and outperform other methods.

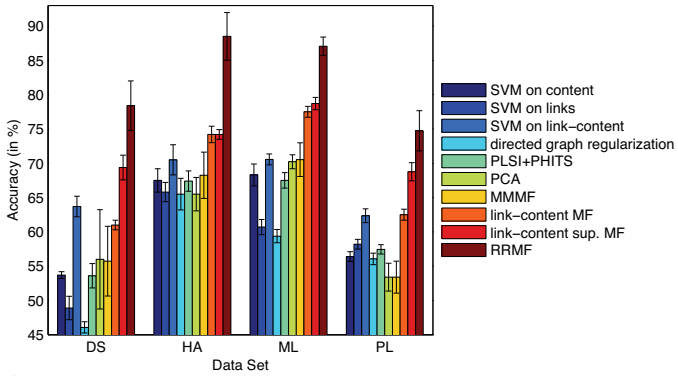


Figure 3: Average classification accuracies with standard deviations on the Cora data set.

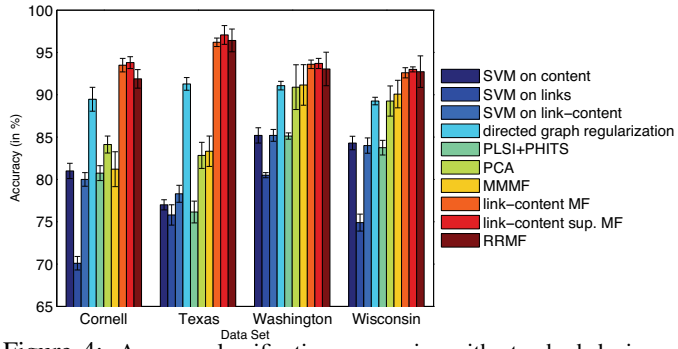
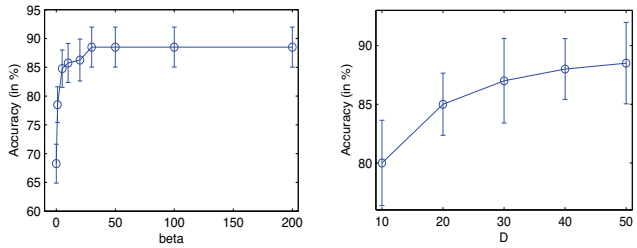


Figure 4: Average classification accuracies with standard deviations on the WebKB data set.

3.5 Sensitivity to Parameters

We have illustrated the effect of the number of iterations in Section 3.3. Here, we examine the sensitivity of RRMF to β and the number of factors D. We again use the HA data set for illustration. Figure 5 illustrates the accuracy of RRMF when β and D take different values. From Figure 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 5(b), we can see that as D increases, the accuracy also increases. But larger D 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 D

Figure 5: Sensitivity to parameters of RRMF.

4 Conclusion and Future Work

In this paper, we propose a novel MF framework, called RRMF, to model relational data of Type II links. One promising property of RRMF is that it can also be used to model data of Type I links just by the inclusion of a very simple preprocessing step. Experimental results verify that RRMF can dramatically outperform other methods on data of Type II links, and can achieve performance comparable with state-of-the-art methods on data of Type I 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.

Incorporating label information into RRMF will be pursued in our future work. Furthermore, although the collective classification methods [Sen *et al.*, 2008], latent Wishart processes (LWP) [Li *et al.*, 2009] and relational topic model (RTM) [Chang and Blei, 2009] are not closely related to RRMF, performing experimental comparison between them will be very interesting and will help to reveal the relationships between several approaches of relational learning work which are currently disparate.

Acknowledgements

We thank Shenghuo Zhu for the preprocessed data sets. This research has been supported by General Research Fund 621407 from the Research Grants Council of Hong Kong.

References

[Chang and Blei, 2009] Jonathan Chang and David M. Blei. Relational topic models for document networks. In *AISTATS*, 2009.

[Chung, 1997] Fan Chung. *Spectral Graph Theory*. Number 92 in Regional Conference Series in Mathematics. American Mathematical Society, 1997.

[Cohn and Hofmann, 2000] David A. Cohn and Thomas Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In *NIPS*, 2000.

[Craven *et al.*, 1998] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to extract symbolic knowledge from the world wide web. In *AAAI/IAAI*, pages 509–516, 1998.

[Deerwester *et al.*, 1990] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6), 1990.

[Kurland and Lee, 2005] Oren Kurland and Lillian Lee. Pagerank without hyperlinks: structural re-ranking using links induced by language models. In *SIGIR*, 2005.

[Li *et al.*, 2009] Wu-Jun Li, Zhihua Zhang, and Dit-Yan Yeung. Latent Wishart processes for relational kernel learning. In *AISTATS*, 2009.

[McCallum *et al.*, 2000] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Inf. Retr.*, 3(2):127–163, 2000.

[Rennie and Srebro, 2005] Jason D. M. Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML*, 2005.

[Sen *et al.*, 2008] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3), 2008.

[Shewchuk, 1994] Jonathan R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Pittsburgh, PA, USA, 1994.

[Singh and Gordon, 2008] Ajit Paul Singh and Geoffrey J. Gordon. A unified view of matrix factorization models. In *ECML/PKDD (2)*, 2008.

[Srebro *et al.*, 2004] Nathan Srebro, Jason D. M. Rennie, and Tommi Jaakkola. Maximum-margin matrix factorization. In *NIPS*, 2004.

[Zhou *et al.*, 2005] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *ICML*, 2005.

[Zhu *et al.*, 2007] Shenghuo Zhu, Kai Yu, Yun Chi, and Yihong Gong. Combining content and link for classification using matrix factorization. In *SIGIR*, 2007.