Bipartite Edge Prediction via Transductive Learning over Product Graphs

Hanxiao Liu HANXIAOL@CS.CMU.EDU
Carnegie Mellon University, Pittsburgh, PA 15213 USA

Yiming Yang YIMING@CS.CMU.EDU
Carnegie Mellon University, Pittsburgh, PA 15213 USA

Abstract

This paper addresses the problem of predicting the missing edges of a bipartite graph where each side of the vertices has its own intrinsic structure. We propose a new optimization framework to map the two sides of the intrinsic structures onto the manifold structure of the edges via a graph product, and to reduce the original problem to vertex label propagation over the product graph. This framework enjoys flexible choices in the formulation of graph products, and supports a rich family of graph transduction schemes with scalable inference. Experiments on benchmark datasets for collaborative filtering, citation network analysis and prerequisite prediction of online courses show advantageous performance of the proposed approach over other state-of-the-art methods.

1. Introduction

Machine learning applications to many important problems involve predicting the missing edges in a bipartite graph based on heterogeneous sources of information about both the vertices and the edges. In recommendation systems, for example, observed user-item interactions can be represented as the (weighted) edges in a bipartite graph where the users are the vertices on the left and the items are vertices on the right. In order to predict the unobserved user-item interactions successfully, inference needs to be made not only based on the observed edges, but also based on additional information about the vertices, such as demographic data of users and textual descriptions about items. The induced intrinsic structures within those vertices would also be informative for inference, such as the graph of user-user similarities and the graph of item-item similarities. The challenging question in context-aware collaborative filtering (Melville et al., 2002; Basilico & Hofmann, 2004; Gu et al., 2010) therefore is how to jointly leverage the rich and heterogeneous information about both the observed edges and the vertices in the bipartite graph.

Other examples include citation network analysis (with publications as the vertices on both sides of the bipartite graph), multi-label text classification (with documents on the left sides and categories on the right side of the bipartite graph), question-answer mapping, host-pathogen interaction modeling, prerequisite linkage within online courses, and more. All of those problems can be viewed as *Bipartite Edge Prediction* (BEP), whose success crucially depends on how to jointly leverage the observed edges and the intrinsic structures within vertices.

A representative approach to BEP is *matrix completion*, which has been intensively studied in recent machine learning (Mnih & Salakhutdinov, 2007; Candès & Recht, 2009). Using a sparse matrix to record the observed edges in a bipartite graph, the prediction of the missing entries in this matrix (i.e., the missing edges in the graph) is accomplished via dimensionality reduction. That is, by finding a lower-dimensional vector space for the observed data, the missing entries can be estimated by approximation. A major weakness of such a matrix completion approach is that the inference is based on observed edges only, ignoring other information about vertices or the intrinsic manifolds among them. As a consequence, such methods cannot effectively handle the cold-start problems in collaborative filtering, for example, where new users or new items do not have enough observed interactions for reliable inference.

Other representative works in BEP include a family of *tensor-kernel* based approaches (Basilico & Hofmann, 2004; Yu et al., 2006; Brunner et al., 2012), which makes a combined use of observed edges and additional information about vertices. E.g., the tensor kernel can be used to combine a matrix of user-user similarities based on demographic data of users and a matrix of item-item similarities based text descriptions of items, and to obtain the induced

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

kernel matrix of edge-edge similarities. Then a kernelized supervised learning algorithm (such as Support Vector Machine or perceptron) can be used to obtain a statistical mapping from any missing edge to the class labels or graded relevance with estimated confidence scores based on a training set of labeled edges. Yet another representative approach is referred to as *Graph Regularized Matix Complection* (Cai et al., 2011; Gu et al., 2010), which extends conventional matrix completion with additional graph regularization terms in the objective for optimization, and the regularization terms are defined based on the manifold structures among the vertices on each side of the bipartite graph.

Although the tensor kernels and graph-regularized matrix completion methods are more powerful than matrix completion as they jointly exploit both the observed edges and the intrinsic structures within vertices, they still have a fundamental limitation. That is, none of those methods explicitly model the intrinsic manifold among *edges* (observed and unobserved) for transductive semi-supervised learning in the prediction of missing edges. Recall that transductive graph learning has been intensively studied for solving vertex classification or vertex label propagation problems (Zhu et al., 2003; Zhu, 2005; Agarwal, 2006), where the intrinsic manifold among unlabeled vertices is proven to be useful for improving the prediction accuracy based on some smoothness or manifold assumption within the homogeneous graph. Transductive learning should also be useful for missing edge prediction in bipartite graphs, we believe; however, such a potential has not been studied for BEP so far.

Improving the current state of the art by proposing a new transductive learning approach to BEP is our goal in this paper. Specifically, we accomplish this goal with the following technical contributions:

(1) A unified optimization framework to establish a principled mapping from the original BEP problem to a vertex label propagation problem over an induced product graph, and to maximally leverage both the observed (labeled) edges and unobserved (unlabeled) edges via transductive semi-supervised learning (Sections 2);

(2) The principled solutions for constructing graph products (via a family of graph product operations), where each edge in the original bipartite graph is mapped onto a vertex in the product graph, and the intrinsic structures within the original vertices are used to define the structure of vertices in the product graph (Section 3);

(3) A rich family of kernel mapping schemes which allow the graph transduction to be carried out in various forms over different product graphs (Section 4);

(4) The scalable algorithms for transductive learning over product graphs (Section 5);

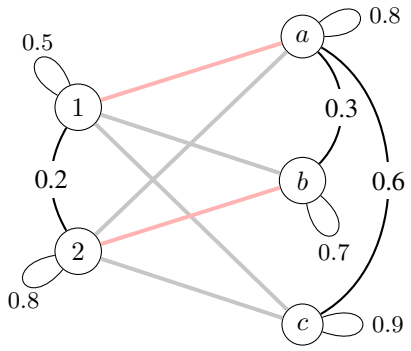

Figure 1. Two interactive graphs $\mathcal{G}$ (left) and $\mathcal{H}$ (right) with vertex sets $V_{\mathcal{G}} = \{1, 2\}$ and $V_{\mathcal{H}} = \{a, b, c\}$. Each graph is equipped with its own intrinsic structure denoted by the dark curved lines. Their interactions are modeled by the straight lines in the middle, i.e. the edge set $E_{\mathcal{B}}$ of the complete bipartite graph $\mathcal{B}$. Given two labeled (red) edges $(1, a)$ and $(2, b)$, our goal is to make predictions on the unlabeled (gray) edges $(1, b)$, $(1, c)$, $(2, a)$, $(2, c)$.

(5) Thorough experiments with our approach and other representative BEP methods on benchmark data sets in collaborative filtering, citation network analysis, and prerequisite prediction over online courses (Section 6).

2. The Unified Framework for BEP

Let us formally define the Bipartite Edge Prediction problem (BEP) first, and then show how to reduce BEP to a vertex label propagation problem over an induced product graph, and to optimize the transductive learning over the product graph.

2.1. Bipartite Edge Prediction

For any graph $\mathcal{G}$, we denote by $V_{\mathcal{G}}$, $E_{\mathcal{G}}$ and $\boldsymbol{G}$ its vertex set, edge set and adjacency matrix. Let $\boldsymbol{U}_{\mathcal{G}}$ and $\{[\lambda_{\mathcal{G}}]_i\}_{i=1}^{|V_{\mathcal{G}}|}$ be the eigenvectors and eigenvalues of $\boldsymbol{G}$, respectively.

Given two graphs $\mathcal{G}$ and $\mathcal{H}$, let $\mathcal{B}$ be a complete bipartite graph with $V_{\mathcal{B}} = \{V_{\mathcal{G}}, V_{\mathcal{H}}\}$ and $E_{\mathcal{B}} = V_{\mathcal{G}} \times V_{\mathcal{H}}$. Suppose $E_{\mathcal{B}}$ can be partitioned into $E^l_{\mathcal{B}}$ and $E^u_{\mathcal{B}}$ where only edges in $E^l_{\mathcal{B}}$ are labeled with $\mathcal{T} = \{y_{ij} \in \mathcal{Y} \mid (i, j) \in E^l_{\mathcal{B}}\}$.

The bipartite edge prediction problem is defined as

Problem 1 (Bipartite Edge Prediction)**.** *Given $\mathcal{G}$, $\mathcal{H}$ and $\mathcal{T}$, learn $f : E_{\mathcal{B}} \mapsto \mathcal{Y}$ such that f accurately predicts the labels over $E^u_{\mathcal{B}}$.*

This is illustrated in Figure 1.

2.2. Vertex Label Propagation over Product Graphs

Since the edges to label $E^u_{\mathcal{B}}$ are given, we consider a transductive learning strategy that propagates the labels $\mathcal{T}$ over

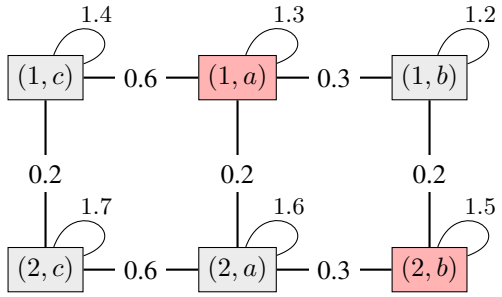

Figure 2. The (Cartesian) product graph of $\mathcal{G}$ and $\mathcal{H}$. Each vertex corresponds to an edge of the complete bipartite graph $\mathcal{B}$ in Figure 1, and each edge encodes the similarity between two edges in $\mathcal{B}$. Now our task becomes: given two labeled (red) vertices $(1, a)$ and $(2, b)$, to predict the remaining unlabeled (gray) vertices.

$E^l_{\mathcal{B}}$ to $E^u_{\mathcal{B}}$. To enable such graph propagation, it would be desirable to have the graph structure of $E_{\mathcal{B}}$, i.e. some graph whose vertices are $E_{\mathcal{B}}$, and whose edge strengths code the similarities among the elements in $E_{\mathcal{B}}$. Though such edge-level graph manifold structure is not directly provided, it can be induced by taking the graph product of $\mathcal{G}$ and $\mathcal{H}$.

Definition 1 (Graph Product)**.** *Given some graph product operator "$\circ$", the graph product of $\mathcal{G}$ and $\mathcal{H}$, denoted by $\mathcal{G} \circ \mathcal{H}$, is a graph with the vertex set $V_{\mathcal{G}\circ\mathcal{H}} = V_{\mathcal{G}} \times V_{\mathcal{H}}$ and adjacency matrix $\boldsymbol{G} \circ \boldsymbol{H}$.*

Note we have assumed that the graph product operator "$\circ$" also defines a matrix operator. Different realizations of "$\circ$" (i.e. different ways of computing the matrix $\boldsymbol{G} \circ \boldsymbol{H}$) will be discussed in detail in Section 3.

Once the product graph $\mathcal{G} \circ \mathcal{H}$ is constructed, $V_{\mathcal{G}\circ\mathcal{H}} \equiv E_{\mathcal{B}}$, and the affinity between any two edges (i, j) and (i', j') of $\mathcal{B}$ is quantified by $[\boldsymbol{G} \circ \boldsymbol{H}]_{(i,j),(i',j')}$. The labeled and unlabeled edges in $\mathcal{B}$, i.e. $E^l_{\mathcal{B}}$ and $E^u_{\mathcal{B}}$ are mapped onto $V^l_{\mathcal{G}\circ\mathcal{H}}$ and $V^u_{\mathcal{G}\circ\mathcal{H}}$, respectively. Only vertices in $V^l_{\mathcal{G}\circ\mathcal{H}}$ are labeled with $\mathcal{T} = \{y_{ij} \in \mathcal{Y} \mid (i, j) \in V^l_{\mathcal{G}\circ\mathcal{H}}\}$.

This suggests that BEP over $\mathcal{B}$ can be reduced to the following vertex label propagation problem over $\mathcal{G} \circ \mathcal{H}$:

Problem 2 (Vertex Label Propagation)**.** *Given $\mathcal{G} \circ \mathcal{H}$ and $\mathcal{T}$, learn $f : V_{\mathcal{G}\circ\mathcal{H}} \mapsto \mathcal{Y}$ such that f accurately predicts the labels over $V^u_{\mathcal{G}\circ\mathcal{H}}$.*

This is illustrated in Figure 2.

2.3. Optimization Objective

For brevity we let $m = |V_{\mathcal{G}}|$ and $n = |V_{\mathcal{H}}|$.

Denote by $\boldsymbol{F} \in \mathbb{R}^{m\times n}$ our estimation matrix where f_{ij} is the function value of f evaluated on vertex (i, j) in $V_{\mathcal{G}\circ\mathcal{H}}$. Given Problem 2, we consider the following graph regularization framework over the product graph $\mathcal{G} \circ \mathcal{H}$

$$\min_{\boldsymbol{F}} \; L_{\mathcal{T}}(\boldsymbol{F}) + \frac{C}{2}\mathrm{vec}(\boldsymbol{F})^{\top} [\kappa(\boldsymbol{G} \circ \boldsymbol{H})]^{-1} \mathrm{vec}(\boldsymbol{F}) \quad (1)$$

where $\mathrm{vec} : \mathbb{R}^{m\times n} \mapsto \mathbb{R}^{mn}$ concatenates the rows of $\boldsymbol{F}$ into a single vector, $L_{\mathcal{T}} : \mathbb{R}^{m\times n} \mapsto \mathbb{R}$ denotes some loss function measuring the discrepancy between our estimation matrix $\boldsymbol{F}$ and the ground truth $\mathcal{T}$, $\kappa : \mathbb{R}^{mn\times mn} \mapsto \mathbb{R}^{mn\times mn}$ maps the adjacency matrix of the product graph, i.e. $\boldsymbol{G} \circ \boldsymbol{H}$, to a kernel matrix related to graph transduction.

In this paper, we restrict our attention to a representative family of κ's called the Spectral Transformation (ST).

Definition 2 (Spectral Transformation)**.** *Given some adjacency matrix $\boldsymbol{A}$ with eigendecomposition $\sum_i \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^{\top}$ and a scalar-valued function κ. The ST of $\boldsymbol{A}$ w.r.t. κ, denote by $\kappa(\boldsymbol{A})$, is defined as $\kappa(\boldsymbol{A}) = \sum_i \kappa(\lambda_i)\, \boldsymbol{u}_i \boldsymbol{u}_i^{\top}$.*

That is, the kernel mapping κ is a spectral transformation if applying κ to an adjacency matrix amounts to applying κ to each of its eigenvalues.

Graph transduction is crucial for leveraging unlabeled vertices in vertex label propagation, and many famous graph transduction schemes can be encoded with ST (Smola & Kondor, 2003; Kunegis & Lommatzsch, 2009). As will be discussed in more detail in Section 4, specifying different κ in optimization (1) is equivalent to carrying out different forms of graph transduction over the vertices of the product graph $\mathcal{G} \circ \mathcal{H}$, i.e. the edges of the bipartite graph $\mathcal{B}$.

3. Constructing the Product Graph

As we can see from the previous section, the graph structure among the edges of $\mathcal{B}$ is coded in the adjacency matrix $\boldsymbol{G} \circ \boldsymbol{H}$, which is a function of the graph product operator "$\circ$". Here we are going to discuss about different kinds of graph products and their intuitions.

We are going to start from two basic graph products: the Tensor graph product (TGP) and the Cartesian graph product (CGP). Then, we generalize TGP and CGP to a family of graph products called the spectral graph product (SGP).

3.1. Tensor Graph Product

Definition 3 (TGP)**.** *The Tensor Graph Product of $\mathcal{G}$ and $\mathcal{H}$, denoted by $\mathcal{G} \otimes \mathcal{H}$, has the adjacency matrix $\boldsymbol{G} \otimes \boldsymbol{H}$, where "$\otimes$" is the Kronecker (Tensor) product.*

Namely, for all (i, j) and (i', j') in $V_{\mathcal{G}\otimes\mathcal{H}}$:

$$[\boldsymbol{G} \otimes \boldsymbol{H}]_{(i,j),(i',j')} := \boldsymbol{G}_{i,i'} \boldsymbol{H}_{j,j'} \quad (2)$$

Therefore, TGP defines the edge-level similarity in $\mathcal{B}$ (left-hand side) as the product of two vertex-level similarities in

$\mathcal{G}$ and $\mathcal{H}$ (right-hand side). In other words, TGP takes the similarity between two edges (i, j) and (i', j') in $\mathcal{B}$ as the similarity between two vertices i and i' in $\mathcal{G}$ multiplied by the similarity between two vertices j and j' in $\mathcal{H}$.

3.2. Cartesian Graph Product

Definition 4 (CGP). *The Cartesian Graph Product of $\mathcal{G}$ and $\mathcal{H}$, denoted by $\mathcal{G} \oplus \mathcal{H}$, has the adjacency matrix $\boldsymbol{G} \oplus \boldsymbol{H}$, where "$\oplus$" is the Kronecker sum.*

Namely, for all (i, j) and (i', j') in $V_{\mathcal{G}\oplus\mathcal{H}}$:

$$[\boldsymbol{G} \oplus \boldsymbol{H}]_{(i,j),(i',j')} := \boldsymbol{G}_{i,i'} 1_{\{j=j'\}} + 1_{\{i=i'\}} \boldsymbol{H}_{j,j'} \quad (3)$$

where the indicator function $1_{\{\cdot\}}$ equals to one if the condition inside the brackets is satisfied and equals to zero otherwise. Hence CGP considers two edges in $\mathcal{B}$ to be similar only when they share at least one mutual vertex in $\mathcal{G}$ or $\mathcal{H}$.

3.3. Random Walk Interpretations

For any graph $\mathcal{G}$, let us assume its adjacency matrix $\boldsymbol{G}$ has been normalized as the transition matrix of a random walk over $\mathcal{G}$, i.e. $\boldsymbol{G}$ is doubly stochastic. In this case $\boldsymbol{G}_{i,i'}$ is the probability for the random walker on $\mathcal{G}$ to travel from a state (vertex) i to another state i'.

According to (2), it is not hard to see that the random walk over $\mathcal{G} \otimes \mathcal{H}$ amounts to two "synchronized" random walks on $\mathcal{G}$ and $\mathcal{H}$ (Vishwanathan et al., 2010). In each move, walkers on $\mathcal{G}$ and $\mathcal{H}$ simultaneously and independently travel to their next state. Each joint state of the two walkers corresponds to a vertex in the product graph $\mathcal{G} \otimes \mathcal{H}$, i.e. an edge in the bipartite graph $\mathcal{B}$ we concern about.

When self-loops are added to every vertex in $\mathcal{G}$ and $\mathcal{H}$ before computing their TGP, the two random walks on both graphs become "lazy" (Zhou & Schölkopf, 2004). This additional self-reinforcement can be crucial—otherwise according to (2), and since $\boldsymbol{G}_{i,i} \equiv 0$ and $\boldsymbol{H}_{j,j} \equiv 0$, any two edges in $\mathcal{B}$ with a mutual vertex will have zero similarity.

By similar analysis, the random walk over $\mathcal{G} \oplus \mathcal{H}$ amounts to two "asynchronized" random walks over $\mathcal{G}$ and $\mathcal{H}$. In each move, one of $\mathcal{G}$ and $\mathcal{H}$ is chosen with equal probability, and only the random walker on the chosen graph is allowed to travel. If $\mathcal{G}$ is chosen, only $\mathcal{G}$'s random walker will travel from i to i' with probability $\boldsymbol{G}_{i,i'}$; otherwise, only $\mathcal{H}$'s random walker will travel from j to j' with probability $\boldsymbol{H}_{j,j'}$.

3.4. The Generalization: Spectral Graph Products

Now let us generalize TGP and CGP to a broader family of graph products. The family gives a unified characterization of the graph products which we are interested in, and can lead to efficient optimization (Section 5).

Definition 5 (SGP). *"$\circ$" is called the spectral graph product if for any $\mathcal{G}$ and $\mathcal{H}$, the adjacency matrix of the product graph $\mathcal{G} \circ \mathcal{H}$ has the eigendecomposition*

$$\boldsymbol{G} \circ \boldsymbol{H} := (\boldsymbol{U}_{\mathcal{G}} \otimes \boldsymbol{U}_{\mathcal{H}}) \boldsymbol{\Lambda}_{\mathcal{G}\circ\mathcal{H}} (\boldsymbol{U}_{\mathcal{G}} \otimes \boldsymbol{U}_{\mathcal{H}})^{\top} \quad (4)$$

where $\boldsymbol{\Lambda}_{\mathcal{G}\circ\mathcal{H}}$ is a diagonal matrix in $\mathbb{R}^{mn\times mn}$ with $[\boldsymbol{\Lambda}_{\mathcal{G}\circ\mathcal{H}}]_{(i,j),(i,j)} = [\lambda_{\mathcal{G}}]_i \circ [\lambda_{\mathcal{H}}]_j$; "$\circ$" is overloaded to be a scalar-valued binary operator $\circ : \mathbb{R}_+ \times \mathbb{R}_+ \mapsto \mathbb{R}_+$ such that i) $x \circ y \equiv y \circ x$, ii) $x \circ y$ is nondecreasing in both x, y.

Note that both TGG and CGP are special cases of SGP—it is not hard to verify that (4) leads to TGP when $x \circ y = xy$, and CGP when $x \circ y = x + y$. This also suggests that TGP and CGP are fundamental in that they can be viewed as the arithmetic multiplication and addition in the SGP family.

If both $\boldsymbol{G}$ and $\boldsymbol{H}$ are positive semidefinite, by Definition 5 we conclude that $\boldsymbol{G} \circ \boldsymbol{H}$ must also be positive semidefinite. In this case, (1) is always a desirable convex optimization problem, provided that $L_{\mathcal{T}}$ is a convex loss function and κ preserves positive semidefiniteness.

4. Transduction over Product Graphs

Recall that our optimization framework (Section 2) has two key ingredients: the graph product operator $\circ$ and the kernel mapping κ. The former specifies how a product graph (edge manifold) should be induced, and the later specifies how the graph transduction should be carried out over such a product graph. In this section, we show how to take different combinations of graph product operations and κ's to define a rich family of transductive learning models for BEP, starting from the kernel mapping schemes (restricted to the spectral transformation family).

4.1. Kernel Mapping Schemes

The regularization term in the optimization objective of formula (1) can be viewed as a Gaussian prior over the estimation matrix $\boldsymbol{F}$, i.e.,

$$\mathrm{vec}(\boldsymbol{F}) \sim \mathcal{N}(\boldsymbol{0}_{mn}, \kappa(\boldsymbol{G} \circ \boldsymbol{H})) \quad (5)$$

where $\boldsymbol{0}_{mn}$ is an all-zero vector in $\mathbb{R}^{mn}$. Given $\circ$, κ specifies the covariance matrix in the Gaussian prior. Choosing different κ allows us to inject our beliefs into the formulation of transduction over a product graph.

Let us use the exponential kernel $\kappa(z) := \exp(z)$ as a concrete example. In this case, $\kappa(\boldsymbol{G} \circ \boldsymbol{H}) = \exp(\boldsymbol{G} \circ \boldsymbol{H}) = \sum_{t=0}^{\infty} \frac{1}{t!} (\boldsymbol{G} \circ \boldsymbol{H})^t$. If $\boldsymbol{G} \circ \boldsymbol{H}$ encodes the transition probabilities across the edges in bipartite graph $\mathcal{B}$, this essentially entails the infinite random walk (Kondor & Lafferty, 2002) over the intrinsic manifold within the edges. Other representative kernel mapping schemes we study in this paper are listed in Table 1 (column 3), including:

- *fixed-step random walk*, where α specifies the number of steps and $(\boldsymbol{G} \circ \boldsymbol{H})^{\alpha}$ are the transition probabilities;
- *von-Neumann kernel* for graph-Laplacian based manifold regularization (to be discussed in Section 4.3);
- *sigmoid kernel*, which can be viewed as a composition of the exponential and the von-Neumann kernel.

4.2. Graph-product based Transduction Models

In Table 1 we list some examples of the transduction models with different combinations of graph products (columns 1&2) and kernel mapping schemes (columns 3&4), and important factors (columns 5&6) for the algorithm design of each model (Section 5). In the next two sections, we pick two Cartesian-product based models to illustrate their interpretations, including the connection to representative works in vertex label propagation [1].

4.3. Cartesian product with von-Neumann Kernel

This corresponds to the transduction model in the fourth row of Table 1. Now we show that this combination is related to Laplacian-based manifold regularization, and is the generalization (for BEP problems) of typical approaches to vertex label propagation (Zhu, 2005; Zhu et al., 2003).

Given any graph $\mathcal{G}$ with adjacency matrix $\boldsymbol{G}$, let $\boldsymbol{D}$ be a diagonal degree matrix with $d_{ii} = \sum_k \boldsymbol{G}_{i,k}$. The normalized graph Laplacian of $\mathcal{G}$ is defined as $\mathcal{L}_{\mathcal{G}} := \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{G} \boldsymbol{D}^{-\frac{1}{2}}$. For simplicity we will assume $\boldsymbol{G}$ has already been symmetrically normalized during preprocessing, thus $\mathcal{L}_{\mathcal{G}} = \boldsymbol{I} - \boldsymbol{G}$.

Recall that $\boldsymbol{F}$ is an estimation matrix where f_{ij} corresponds to the function value of f evaluated on edge (i, j) in $E_{\mathcal{B}}$. Reinforcing a smooth transition of f across the vertices in $\mathcal{G}$ requires the following quantity to be set small:

$$\sum_{i \in V_{\mathcal{G}}} \sum_{i' \in V_{\mathcal{G}}} \boldsymbol{G}_{i,i'} \|\boldsymbol{F}_{i,:} - \boldsymbol{F}_{i',:}\|_2^2 \equiv \operatorname{tr}\left(\boldsymbol{F}^{\top} \mathcal{L}_{\mathcal{G}} \boldsymbol{F}\right) \tag{6}$$

where $\boldsymbol{F}_{i,:}$ stands for the i-th row of our estimation matrix $\boldsymbol{F}$ about edges (i.e. strengths for the out-links of vertex i in $\mathcal{G}$). The similar analysis applies to graph $\mathcal{H}$.

To reinforce smooth transductions over both $\mathcal{G}$ and $\mathcal{H}$, consider the following manifold regularization objective

$$\begin{aligned} \min_{\boldsymbol{F}} \quad & L_{\mathcal{T}}(\boldsymbol{F}) + \frac{C_1}{2} \|\boldsymbol{F}\|_{\mathrm{F}}^2 \\ & + \frac{C_2}{2} \operatorname{tr}\left(\boldsymbol{F}^{\top} \mathcal{L}_{\mathcal{G}} \boldsymbol{F}\right) + \frac{C_2}{2} \operatorname{tr}\left(\boldsymbol{F} \mathcal{L}_{\mathcal{H}} \boldsymbol{F}^{\top}\right) \end{aligned} \tag{7}$$

[1] Although we choose to only focus on a few interesting combinations of graph products and κ's in this paper, our framework is more general for many other combinations. We leave those opportunities for future research

Comparing the objective in (7) to those in typical manifold-based vertex label propagation problems, the only differences are that each vertex is associated with a vector value (the column or row of $\boldsymbol{F}$) instead of a scalar, and we have the regularization terms for two graphs ($\mathcal{G}$ and $\mathcal{H}$) instead of a single graph ($\mathcal{G}$).

The following indicates optimization (7) can be equivalent to (1) with κ specified to be the von-Neumann kernel.

Proposition 1. *Optimization (7) is equivalent to optimization (1) if $\circ$ is the Cartesian graph product (i.e. $x \circ y \equiv x + y$), $C = C_1 + 2C_2$ and $\kappa(z) = \left(1 - \frac{C_2}{C} z\right)^{-1}$.*

4.4. Cartesian product with Exponential Kernel

This corresponds to the transduction model in the second row of Table 1. The following shows that this model connects together the Cartesian and the Tensor graph product.

Applying the exponential kernel to the Cartesian product graph amounts to setting the covariance matrix of the Gaussian prior to $\exp(\boldsymbol{G} \oplus \boldsymbol{H})$. Notice that the following nice equivalence holds (Neudecker, 1969)

$$\exp(\boldsymbol{G} \oplus \boldsymbol{H}) = \exp(\boldsymbol{G}) \otimes \exp(\boldsymbol{H}) \tag{8}$$

In other words, an infinite random walk over the Cartesian product graph is equivalent to the Tensor graph product of the two infinite random walks over $\boldsymbol{G}$ and $\boldsymbol{H}$, respectively.

5. Optimization Algorithms

Although our proposed framework enjoys the nice property of allowing various forms of edge-level graph transduction, the induced product graph is typically extremely large thus leads to nontrivial optimization.

In this section, we focus on speeding up the gradient computation of the criterion function in optimization (1), which is the building block for many optimization routines. After examining the bottleneck of gradient computation, we show that the gradient can be computed much more efficiently when $\circ$ belongs to the SGP family. Then, we give a generic characterization about in what cases (i.e. for which combinations of $\circ$ and κ) the gradient can be concisely expressed, and in what cases the optimization efficiency can be further boosted by restricting the rank of $\boldsymbol{F}$.

5.1. Bottleneck of Gradient Computation

The gradient of (1) can be expressed as

$$\nabla \boldsymbol{F} = \nabla L_{\mathcal{T}}(\boldsymbol{F}) + C \varphi_{\circ,\kappa}(\boldsymbol{F}) \tag{9}$$

where we have used $\varphi_{\circ,\kappa}(\boldsymbol{F})$ as a shorthand for the gradient of the graph regularization term

$$\varphi_{\circ,\kappa}(\boldsymbol{F}) := \operatorname{vec}^{-1}\left(\left[\kappa(\boldsymbol{G} \circ \boldsymbol{H})\right]^{-1} \operatorname{vec}(\boldsymbol{F})\right) \tag{10}$$

Table 1. Graph-product based Transduction Models

GRAPH PRODUCT	$x \circ y := z$	TRANSDUCTION	$\kappa(z)$	$\operatorname{rank}(\boldsymbol{\Sigma})$	$\varphi_{\circ,\kappa}(\boldsymbol{F})$
TENSOR	xy	RANDOM WALK	z^{α}	1	$\boldsymbol{G}^{-\alpha}\boldsymbol{F}\boldsymbol{H}^{-\alpha}$
CARTESIAN	$x+y$	EXPONENTIAL	$\exp(\alpha z)$	1	$\exp(-\alpha\boldsymbol{G})\,\boldsymbol{F}\exp(-\alpha\boldsymbol{H})$
TENSOR	xy	VON-NEUMANN	$(1-\alpha z)^{-1}$	2	$\boldsymbol{F}-\alpha\boldsymbol{G}\boldsymbol{F}\boldsymbol{H}$
CARTESIAN	$x+y$	VON-NEUMANN	$(1-\alpha z)^{-1}$	3	$\boldsymbol{F}-\alpha\boldsymbol{G}\boldsymbol{F}-\alpha\boldsymbol{F}\boldsymbol{H}$
CARTESIAN	$x+y$	SIGMOID	$(1+\exp(-\alpha z))^{-1}$	2	$\boldsymbol{F}+\exp(-\alpha\boldsymbol{G})\,\boldsymbol{F}\exp(-\alpha\boldsymbol{H})$

One has to compute $\varphi_{\circ,\kappa}(\boldsymbol{F})$ in order to compute the gradient $\nabla\boldsymbol{F}$. Unfortunately, computing $\varphi_{\circ,\kappa}(\boldsymbol{F})$ according to (10) is prohibitively expensive in both space and time, due to the huge $mn \times mn$ size of $\boldsymbol{G}\circ\boldsymbol{H}$, the presence of κ (which can lead to a fully dense $mn \times mn$ kernel matrix $\kappa(\boldsymbol{G}\circ\boldsymbol{H})$ even when $\boldsymbol{G}\circ\boldsymbol{H}$ is sparse), and the presence of matrix inverse operation. More specifically, each gradient step will consume $O\left(m^2n^2\right)$ in both time and space, assuming $[\kappa(\boldsymbol{G}\circ\boldsymbol{H})]^{-1}$ is already somehow precomputed.

5.2. Efficient Gradient Computation with SGP

The following lemma indicates that $\varphi_{\circ,\kappa}(\boldsymbol{F})$ can be computed much more efficiently if $\circ$ is a spectral graph product.

Lemma 1. *If $\circ$ defines a SGP, then*

$$\varphi_{\circ,\kappa}(\boldsymbol{F}) = \boldsymbol{U}_{\mathcal{G}}\left[\boldsymbol{\Sigma}_{\circ,\kappa} * \left(\boldsymbol{U}_{\mathcal{G}}^{\top}\boldsymbol{F}\boldsymbol{U}_{\mathcal{H}}\right)\right]\boldsymbol{U}_{\mathcal{H}}^{\top} \tag{11}$$

where $*$ is the matrix Hadamard (a.k.a. element-wise) product, $\boldsymbol{\Sigma}_{\circ,\kappa}$ is a $m\times n$ matrix with each of its elements defined as $[\boldsymbol{\Sigma}_{\circ,\kappa}]_{ij} = 1\Big/\kappa\big([\lambda_{\mathcal{G}}]_i \circ [\lambda_{\mathcal{H}}]_j\big)$.

Lemma 1 indicates as long as the eigensystems of $\boldsymbol{G}$ and $\boldsymbol{H}$ are precomputed and $\circ$ defines a SGP, the computation of $\varphi_{\circ,\kappa}(\boldsymbol{F})$ does not require the explicit construction of $\boldsymbol{G}\circ\boldsymbol{H}$. It allows us to compute $\varphi_{\circ,\kappa}(\boldsymbol{F})$ in $O(mn(m+n))$ time and $O\left((m+n)^2\right)$ space, far more efficient than the $O\left(m^2n^2\right)$ complexity using the naive approach.

In (11) we see $\boldsymbol{\Sigma}_{\circ,\kappa} := \boldsymbol{\Sigma}$ play as the key quantity, since it summarizes all the information about our choices of the graph product $\circ$ and the kernel mapping κ. In the extreme case where $\boldsymbol{\Sigma}$ is a constant matrix, $\varphi_{\circ,\kappa}(\boldsymbol{F})$ will degenerate to the gradient of the squared Frobenius norm of $\boldsymbol{F}$.

Theorem 1. *If $\circ$ is a SGP, let $\sum_k^{\operatorname{rank}(\boldsymbol{\Sigma})} c_k\boldsymbol{u}_k\boldsymbol{v}_k^{\top}$ be the eigendecomposition of $\boldsymbol{\Sigma}$, and $r(\boldsymbol{G},\boldsymbol{u})$ be matrix $\boldsymbol{G}$ with its eigenvalues replaced by some other vector $\boldsymbol{u}$. We have*

$$\varphi_{\circ,\kappa}(\boldsymbol{F}) = \sum_{k=1}^{\operatorname{rank}(\boldsymbol{\Sigma})} c_k r(\boldsymbol{G},\boldsymbol{u}_k)\,\boldsymbol{F}\,r(\boldsymbol{H},\boldsymbol{v}_k) \tag{12}$$

Theorem 1 provides an alternative approach to compute $\varphi_{\circ,\kappa}(\boldsymbol{F})$. The summation might appear to be cumbersome at the first glance. However, interestingly, we observe that the rank of $\boldsymbol{\Sigma}$, i.e. $\operatorname{rank}(\boldsymbol{\Sigma})$, is bounded by a very small integer (typically ≤ 3) for many different combinations of $\circ$ and κ. For those listed in Table 1, we put their corresponding $\operatorname{rank}(\boldsymbol{\Sigma})$ and the expression of $\varphi_{\circ,\kappa}(\boldsymbol{F})$ derived from Theorem 1 in the last two columns.

Corollary 1. *If there exists σ_1,σ_2 such that for all $x,y\in\mathbb{R}$, $\frac{1}{\kappa(x\circ y)} \equiv \sigma_1(x)\,\sigma_2(y)$. Then*

$$\varphi_{\circ,\kappa}(\boldsymbol{F}) = \sigma_1(\boldsymbol{G})\,\boldsymbol{F}\sigma_2(\boldsymbol{H}) \tag{13}$$

Similarly, if there exits σ_1 and σ_2 such that for all $x,y\in\mathbb{R}$, $\frac{1}{\kappa(x\circ y)} \equiv \sigma_1(x)+\sigma_2(y)$. Then

$$\varphi_{\circ,\kappa}(\boldsymbol{F}) = \sigma_1(\boldsymbol{G})\,\boldsymbol{F} + \boldsymbol{F}\sigma_2(\boldsymbol{H}) \tag{14}$$

Corollary 1 provides us some useful insights about under what cases $\varphi_{\circ,\kappa}(\boldsymbol{F})$, and therefore $\nabla\boldsymbol{F}$, can be expressed concisely. As an example, consider the SGP $x\circ y = x^p+y^q$ and the kernel mapping $\kappa(z)=\exp(z)$, we have $\frac{1}{\kappa(x\circ y)} \equiv (e^x)^{-p}(e^y)^{-q}$. According to the corollary

$$\varphi_{\circ,\kappa}(\boldsymbol{F}) = [\exp(\boldsymbol{G})]^{-p}\,\boldsymbol{F}\,[\exp(\boldsymbol{H})]^{-q} \tag{15}$$

5.3. Optimization with Rank Constraint

So far we have been discussing about how to jointly exploit the structures in both $\mathcal{G}$ and $\mathcal{H}$. Sometimes, we believe that the true labels over the edges of $\mathcal{B}$ are also structured. For example, a popular assumption is that $\boldsymbol{F}$ should have a low-rank nature, i.e. $\operatorname{rank}(\boldsymbol{F}) \leq d \ll \min(m,n)$. This additional constraint has the following two advantages:

First, in the extreme case where neither $\mathcal{G}$ nor $\mathcal{H}$ is informative (e.g. $\boldsymbol{G}$ and $\boldsymbol{H}$ are identity matrices), we still have the hope to recover the missing edges in $\mathcal{B}$ according to the theory of low-rank matrix recovery (Candès & Recht, 2009). This suggests that the low-rank structural information in $\boldsymbol{F}$, to some extent, is orthogonal to the graph structural information in $\mathcal{G}$ and $\mathcal{H}$. Thus this additional low-rank assumption can be used to enhance our BEP framework.

Second, by assuming $\boldsymbol{F} = \boldsymbol{U}\boldsymbol{V}^{\top}$ where $\boldsymbol{U}\in\mathbb{R}^{m\times d}$ and $\boldsymbol{V}\in\mathbb{R}^{n\times d}$, the number of free variables to be optimized

in problem (1) is substantially reduced. As in existing fast matrix factorization approaches (Rennie & Srebro, 2005), alternatively optimizing w.r.t. thin matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ may lead to improved computational efficiency.

Ideally, the cost of gradient computation should decrease as d decreases, namely as $\boldsymbol{U}$ and $\boldsymbol{V}$ become thinner. However, this is not true if we compute $\varphi_{\circ,\kappa}(\boldsymbol{F})$ according to Lemma (1), due to the presence of the Hadamard product.

Fortunately, Theorem 1 allows the gradient computation to benefit from the rank constraint over $\boldsymbol{F}$, by which the complexity for $\varphi_{\circ,\kappa}(\boldsymbol{F})$ is $O\left(d\left(m^2+n^2\right)\operatorname{rank}(\boldsymbol{\Sigma})\right)$. Notice $\operatorname{rank}(\boldsymbol{\Sigma})$ is typically a very small constant (Table 1), we have $O\left(d\left(m^2+n^2\right)\operatorname{rank}(\boldsymbol{\Sigma})\right) \approx O\left(d\left(m^2+n^2\right)\right) \ll O\left(mn\left(m+n\right)\right)$, hence we can get huge computation savings via (12) by restricting d to be small.

6. Experiments

We conducted experiments with various kinds of BEP (Bipartite Edge Prediction) problems, including those in collaborative filtering, citation network analysis, and prerequisite prediction for online courses.

6.1. Tasks and Datasets

- **Collaborative Filtering**: We used *MovieLens-100K*, a benchmark data set in collaborative filtering where the task is to predict the unknown ratings for new user-movie pairs. The bipartite graph $\mathcal{B}$ in this case has the vertex sets $V_{\mathcal{G}}$, $V_{\mathcal{H}}$ and the edge set $E_{\mathcal{B}}$ corresponding to 943 users, 1682 movies and 10^5 ratings, respectively. Each user is provided with a binary vector indicating his/her gender and occupation, and each movie is provided with a binary vector indicating its genre.
- **Citation Network Analysis**: We also used *Cora* (Sen et al., 2008), including 2708 publication records and 5429 citation links, which has been commonly used in citation network analysis where the task is to predict the relevance of unknown citations for each "query" publication. The bipartite graph in this case has identical $V_{\mathcal{G}}$ and $V_{\mathcal{H}}$, i.e., both correspond to the publication (document) set, and the edge set $E_{\mathcal{B}}$ corresponds to the citation links. Each document is also provided with a binary vector, indicating the within-document presence or absence of each word in the vocabulary.
- **Prerequisite Prediction**: *Courses*[2] (Yang et al., 2015) is a new set of course descriptions and prerequisite links we collected from the web sites of Massachusetts Institute of Technology (2322 courses, 1173 links), California Institute of Technology (1048 courses, 761 links), Princeton University (56 courses, 90 links) and Carnegie Mellon University (83 courses and 150 links). For each institution, the bipartite graph $\mathcal{B}$ has identical vertex sets $V_{\mathcal{G}}$ and $V_{\mathcal{H}}$, corresponding to the courses, and the edges in $E_{\mathcal{B}}$ indicate the perquisite relations among courses. Each course is provided with a bag-of-words representation based on the course description. The task is to predict the strength for each unknown edge in $E_{\mathcal{B}}$.

[2] http://nyc.lti.cs.cmu.edu/teacher/dataset/

All the above data were used in 5-fold cross validation settings: we used 60% of the data for training, 20% for parameter tuning, and 20% for testing. By rotating the 5-fold training/validating/test subsets we measure the performance of each method on average.

Based on the features of the vertices, we construct $\mathcal{G}$ and $\mathcal{H}$ as sparse, symmetrized kNN graphs under cosine similarity. The value of k is tuned during cross validation.

6.2. Evaluation Metrics

For evaluating BEP methods in citation network analysis and prerequisite prediction, each vertex on the left side of the bipartite graph is treated as a query, and the system-produced ranked list of unknown edges in $E_{\mathcal{B}}$ is evaluated using the standard metrics of the Mean Average Precision (MAP), the Area Under the Curve (AUC) of ROC, and the Normalized Discounted Cumulative Gain (NDCG). All those metrics have been commonly used in the benchmark evaluations of ranked lists.

For evaluating BEP methods in collaborative filtering, MAP and AUC do not apply because they are defined for binary relevance judgments but MovieLens has multi-scale ratings (1-5). On the other hand, NDCG is well-defined for multi-scale relevance judgments which has been commonly used in collaborative filtering evaluations (Weimer et al., 2007; Rendle et al., 2009; Balakrishnan & Chopra, 2012), and therefore is our choice of metric for this task.

6.3. Results of Proposed Methods

We conducted controlled experiments with all the proposed models in Table 1. For collaborative filtering we use mean squared error (MSE) as the loss function, and for other two tasks we use the pairwise ranking loss.

In our result summary (Table 2), we name these methods as BEP with a subscript (for the type of graph product) and superscript (for the formulation of the kernel mapping κ). For example, $\text{BEP}_{\oplus}^{exp}$ means BEP with an exponential kernel over the Cartesian product graph.

From Table 2 we see that $\text{BEP}_{\oplus}^{exp}$, i.e. exponential kernel over the Cartesian product graph, yields the best overall performance. This empirically justifies the effectiveness of the infinite random walk we discussed in Section 4.

Table 2. Results of our methods

Dataset	Method	MAP	AUC	ndcg@3
Courses	$\text{BEP}^{rw}_{\otimes}$	0.488	0.827	0.461
	$\text{BEP}^{exp}_{\oplus}$	**0.518**	**0.872**	**0.500**
	$\text{BEP}^{von}_{\otimes}$	0.472	0.861	0.449
	$\text{BEP}^{von}_{\oplus}$	0.366	0.531	0.359
	$\text{BEP}^{sig}_{\oplus}$	0.443	0.617	0.431
Cora	$\text{BEP}^{rw}_{\otimes}$	0.222	0.764	0.205
	$\text{BEP}^{exp}_{\oplus}$	**0.256**	**0.884**	**0.232**
	$\text{BEP}^{von}_{\otimes}$	0.230	0.853	0.211
	$\text{BEP}^{von}_{\oplus}$	0.218	0.633	0.212
	$\text{BEP}^{sig}_{\oplus}$	0.192	0.443	0.188
MovieLens	$\text{BEP}^{rw}_{\otimes}$	-	-	0.7695
	$\text{BEP}^{exp}_{\oplus}$	-	-	0.7702
	$\text{BEP}^{von}_{\otimes}$	-	-	**0.7720**
	$\text{BEP}^{von}_{\oplus}$	-	-	0.7624
	$\text{BEP}^{sig}_{\oplus}$	-	-	0.7650

6.4. Comparison with Baseline Methods

We further conducted experiments with other representative methods in the literature as strong baselines, including:

- **Matrix Completion** (MC) has been a common approach to the prediction of the missing entries in a sparse input matrix via matrix factorization (Mnih & Salakhutdinov, 2007; Kapicioglu et al., 2014). Those methods do not exploit any intrinsic structure within the vertex sets of $\mathcal{G}$ or $\mathcal{H}$ as a common limitation.
- **Graph Regularized Matrix Completion** (GRMC) extends conventional matrix completion with additional graph regularization based on the manifold structures of $\mathcal{G}$ and $\mathcal{H}$ (Cai et al., 2011; Gu et al., 2010). The GRMC approach is similar to ours in the sense of simultaneously leveraging the manifold information on each side of the bipartite graph. The main difference is that we combine the manifolds of $\mathcal{G}$ and $\mathcal{H}$ into a single product graph, and leverage the induced manifold structure of *edges* in $\mathcal{B}$ to enable edge-based graph transduction across edges, while GRMC does not explicitly model the manifold of edges or the transduction over the edges.
- **Tensor Kernel** (TK) constructs the kernel matrix for the edges in $\mathcal{B}$ by taking the tensor product of the kernel matrices of $\mathcal{G}$'s and $\mathcal{H}$'s (Basilico & Hofmann, 2004; Yu et al., 2006; Brunner et al., 2012). The tensor kernel is then used in supervised learning of edge weight prediction (e.g., using a perceptron algorithm) or edge classification (e.g., using SVM). Although this approach explicitly constructs the similarity measure among $\mathcal{B}$'s edges, it does not leverage transductive learning among those edges.

For fair comparisons, we adapted all the baseline methods to use the same loss function as our proposed method. Table 3 summarizes the results of the baselines and our best method, i.e., $\text{BEP}^{exp}_{\oplus}$ [3].

Table 3. Results of the baseline methods and our method

Dataset	Method	MAP	AUC	ndcg@3
Courses	MC	0.319	0.758	0.294
	GRMC	0.366	0.777	0.343
	TK	0.449	0.810	0.446
	$\text{BEP}^{exp}_{\oplus}$	**0.490**	**0.838**	**0.473**
Cora	MC	0.101	0.697	0.086
	GRMC	0.115	0.702	0.101
	TK	0.248	0.872	0.231
	$\text{BEP}^{exp}_{\oplus}$	**0.268**	**0.894**	**0.243**
MovieLens	MC	-	-	0.748
	GRMC	-	-	0.752
	TK	-	-	0.718
	$\text{BEP}^{exp}_{\oplus}$	-	-	**0.765**

According to Table 3, TK outperforms GRMC on Courses (link sparsity: 0.33%) and Cora (link sparsity: 0.074%), but not on MovieLens (link sparsity: 6.3%). This suggests that MC-based approaches tend to work better when the observed edges in the bipartite graph $\mathcal{B}$ is relatively dense.

In Table 3 we also see our proposed method $\text{BEP}^{exp}_{\oplus}$ consistently outperforms other baselines in all the tasks under all three metrics. This justifies our intuition that transductive learning over the manifold (induced via graph product) of edges in the bipartite graph is useful for BEP problems.

7. Conclusion

We presented a novel approach to bipartite edge prediction by reducing the original problem to a vertex label propagation problem over product graphs. It enables us to simultaneously exploit both the partially labeled edges and the intrinsic structures within the vertices on both sides of the bipartite graph in a principled manner, and to effectively leverage both labeled edges and unlabeled edges via transductive learning over the product graphs. We showed that the optimization can be efficiently implemented for rich combinations of graph products and graph transduction schemes. Our experiments demonstrated the advantageous performance of our proposed approach over strong baselines in real-world BEP tasks.

Acknowledgements

We thank the reviewers for their helpful comments. This work is support in part by the National Science Foundation (NSF) under grants IIS-1216282 and IIS-1350364.

[3] Table 2 and Table 3 are using different 5-fold data splits.

References

Agarwal, Shivani. Ranking on graph data. In *Proceedings of the 23rd international conference on Machine learning*, pp. 25–32. ACM, 2006.

Balakrishnan, Suhrid and Chopra, Sumit. Collaborative ranking. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pp. 143–152. ACM, 2012.

Basilico, Justin and Hofmann, Thomas. Unifying collaborative and content-based filtering. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 9. ACM, 2004.

Brunner, Carl, Fischer, Andreas, Luig, Klaus, and Thies, Thorsten. Pairwise support vector machines and their application to large scale problems. *The Journal of Machine Learning Research*, 13(1):2279–2292, 2012.

Cai, Deng, He, Xiaofei, Han, Jiawei, and Huang, Thomas S. Graph regularized nonnegative matrix factorization for data representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(8):1548–1560, 2011.

Candès, Emmanuel J and Recht, Benjamin. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

Gu, Quanquan, Zhou, Jie, and Ding, Chris HQ. Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs. In *SDM*, pp. 199–210. SIAM, 2010.

Kapicioglu, Berk, Rosenberg, David S, Schapire, Robert E, and Jebara, Tony. Collaborative ranking for local preferences. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pp. 466–474, 2014.

Kondor, Risi Imre and Lafferty, John. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, volume 2, pp. 315–322, 2002.

Kunegis, Jérôme and Lommatzsch, Andreas. Learning spectral graph transformations for link prediction. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 561–568. ACM, 2009.

Melville, Prem, Mooney, Raymond J, and Nagarajan, Ramadass. Content-boosted collaborative filtering for improved recommendations. In *AAAI/IAAI*, pp. 187–192, 2002.

Mnih, Andriy and Salakhutdinov, Ruslan. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pp. 1257–1264, 2007.

Neudecker, H. A note on kronecker matrix products and matrix equation systems. *SIAM Journal on Applied Mathematics*, 17(3):603–606, 1969.

Rendle, Steffen, Freudenthaler, Christoph, Gantner, Zeno, and Schmidt-Thieme, Lars. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 452–461. AUAI Press, 2009.

Rennie, Jasson DM and Srebro, Nathan. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pp. 713–719. ACM, 2005.

Sen, Prithviraj, Namata, Galileo Mark, Bilgic, Mustafa, Getoor, Lise, Gallagher, Brian, and Eliassi-Rad, Tina. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

Smola, Alexander J and Kondor, Risi. Kernels and regularization on graphs. In *Learning theory and kernel machines*, pp. 144–158. Springer, 2003.

Vishwanathan, S Vichy N, Schraudolph, Nicol N, Kondor, Risi, and Borgwardt, Karsten M. Graph kernels. *The Journal of Machine Learning Research*, 11:1201–1242, 2010.

Weimer, Markus, Karatzoglou, Alexandros, Le, Quoc Viet, and Smola, Alex. Maximum margin matrix factorization for collaborative ranking. *Advances in neural information processing systems*, 2007.

Yang, Yiming, Liu, Hanxiao, Carbonell, Jaime G., and Ma, Wanli. Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015, Shanghai, China, February 2-6, 2015*, pp. 159–168, 2015. doi: 10.1145/2684822.2685292. URL http://doi.acm.org/10.1145/2684822.2685292.

Yu, Kai, Chu, Wei, Yu, Shipeng, Tresp, Volker, and Xu, Zhao. Stochastic relational models for discriminative link prediction. In *Advances in neural information processing systems*, pp. 1553–1560, 2006.

Zhou, Dengyong and Schölkopf, Bernhard. A regularization framework for learning from graph data. 2004.

Zhu, Xiaojin. Semi-supervised learning literature survey. 2005.

Zhu, Xiaojin, Ghahramani, Zoubin, Lafferty, John, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pp. 912–919, 2003.