Cross-Graph Learning of Multi-Relational Associations

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

Abstract

Cross-graph Relational Learning (CGRL) refers to the problem of predicting the strengths or labels of multi-relational tuples of heterogeneous object types, through the joint inference over multiple graphs which specify the internal connections among each type of objects. CGRL is an open challenge in machine learning due to the daunting number of all possible tuples to deal with when the numbers of nodes in multiple graphs are large, and because the labeled training instances are extremely sparse as typical. Existing methods such as tensor factorization or tensor-kernel machines do not work well because of the lack of convex formulation for the optimization of CGRL models, the poor scalability of the algorithms in handling combinatorial numbers of tuples, and/or the non-transductive nature of the learning methods which limits their ability to leverage unlabeled data in training. This paper proposes a novel framework which formulates CGRL as a convex optimization problem, enables transductive learning using both labeled and unlabeled tuples, and offers a scalable algorithm that guarantees the optimal solution and enjoys a linear time complexity with respect to the sizes of input graphs. In our experiments with a subset of DBLP publication records and an Enzyme multi-source dataset, the proposed method successfully scaled to the large cross-graph inference problem, and outperformed other representative approaches significantly.

1. Introduction

Many important problems in multi-source relational learning could be cast as joint learning over multiple graphs about how heterogeneous types of objects interact with each other. In literature data analysis, for example, publication records provide rich information about how authors collaborate with each other in a co-authoring graph, how papers are linked in citation networks, how keywords are related via ontology, and so on. The challenging question is about how to combine such heterogeneous information in individual graphs for the labeling or scoring of the multi-relational associations in tuples like `(author,paper,keyword)`, given some observed instances of such tuples as the labeled training set. Automated labeling or scoring of unobserved tuples allows us to discover who have been active in the literature on what areas of research, and to predict who would become influential in which areas in the future. In protein data analysis, as another example, a graph of proteins with pairwise sequence similarities is often jointly studied with a graph of chemical compounds with their structural similarities for the discovery of interesting patterns in `(compound,protein)` pairs. We call the prediction problem in both examples *cross-graph learning of multi-relational associations*, or simply *cross-graph relational learning* (CGRL), where the multi-relational associations are defined by the tuples of heterogeneous types of objects, and each object type has its own graph with type-specific relational structure as a part of the provided data. The task is to predict the labels or the scores of unobserved multi-relational tuples, conditioned on a relatively small set of labeled instances.

CGRL is an open challenge in machine learning for several reasons. Firstly, the number of multi-relational tuples grows combinatorially in the numbers of individual graphs and the number of nodes in each graph. How to make cross-graph inference computationally tractable for large graphs is a tough challenge. Secondly, how to combine the internal structures or relations in individual graphs for joint inference in a principled manner is an open question. Thirdly, supervised information (labeled instances) is typically extremely sparse in CGRL due to the very large number of all possible combinations of heterogeneous objects in individual graphs. Consequently, the success of cross-graph learning crucially depends on effectively leveraging the massively available unlabeled tuples (and the latent relations

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

among them) in addition to the labeled training data. In other words, how to make the learning transductive is crucial for the true success of CGRL. Research on transdcutive CGRL has been quite limited, to our knowledge.

Existing approaches in CGRL or CGRL-related areas can be outlined as those using tensors or graph-regularized tensors, and kernel machines that combine multiple kernels.

Tensor methods have been commonly used for combining multi-source evidence of the interactions among multiple types of objects (Nickel et al., 2011; Rendle et al., 2009; Kolda & Bader, 2009) as the combined evidence can be naturally represented as tuples. However, most of the tensor methods do not explicitly model the internal graph structure for each type of objects, although some of those methods implicitly leverage such information via graph-based regularization terms in their objective function that encourage similar objects within each graph to share similar latent factors (Narita et al., 2012; Cai et al., 2011). A major weakness in such tensor methods is the lack of convexity in their models, which leads to ill-posed optimization problems particularly in high-order scenarios. It has also been observed that tensor factorization models suffer from label-sparsity issue, which is typically severe in CGRL.

Kernel machines have been widely studied for supervised classifiers, where a kernel matrix corresponds to a similarity graph among a single type of objects. Multiple kernels can be combined, for example, by taking the tensor product of each individual kernel matrix, which results in a desired kernel matrix among cross-graph multi-relational tuples. The idea has been explored in relational learning combined with SVMs (Ben-Hur & Noble, 2005), perceptions (Basilico & Hofmann, 2004) or Gaussian process (Yu & Chu, 2008) for two types of objects and is generalizable to the multi-type scenario of CGRL. Although being generic, the complexity of such kernel-based methods grows exponentially in the number of individual kernels (graphs) and the size of each individual graph. As a result, kernel machines suffer from poor scalability in general. In addition, kernel machines are purely supervised (not for transductive learning), i.e., they cannot leverage the massive number of available non-observed tuples induced from individual graphs and the latent connections among them. Those limitations make existing kernel methods less powerful for solving the CGRL problem in large scale and under severely data-sparse conditions.

In this paper, we propose a novel framework for CGRL which can be characterized as follows: (i) It uses graph products to map heterogeneous sources of information and the link structures in individual graphs onto a single *homogeneous* graph; (ii) It provides a convex formulation and approximation of the CGRL problem that ensure robust optimization and efficient computation; and (iii) It enables transductive learning in the form of label propagation over the induced homogeneous graph so that the massively available non-observed tuples and the latent connections among them can play an important role in effectively addressing the label-sparsity issue.

The proposed framework is most related to (Liu & Yang, 2015), where the authors formulated graph products for learning the edges of a bipartite graph. Our new framework is fundamentally different in two aspects. First, our new formulation and algorithms allow the number of individual graphs to be greater than two, while method in (Liu & Yang, 2015) is only applicable to two graphs. Secondly, the algorithms in (Liu & Yang, 2015) suffer from cubic complexity over the graphs sizes (quadratic by using a non-convex approximation), while our new algorithm enjoys both the convexity of the formulation and the low time complexity which is linear over the graph sizes.

Our method also shares the high-level goal with Statistical Relational Learning (SRL) (Getoor, 2007) and Inductive Logic Programming (ILP) (Lavrac & Dzeroski, 1994) in terms of multirelational learning. However, both of our problem setting and formulation differ substantially from existing SRL/ILP approaches focusing on first-order logic and/or probabilistic reasoning over graphical models.

The paper is organized as follows: Section 2 shows how cross-graph multi-relations can be embedded into the vertex space of a homogeneous graph. Section 3 describes how efficient label propagation among multi-relations can be carried out in such space with approximation. We discuss our optimization algorithm in Section 4 and provide empirical evaluations over real-world datasets in Section 5.

2. The Proposed Method

We introduce our notation in 2.1 and the notion of graph product (GP) in 2.2. We then narrow down to a specific GP family with desirable computational properties in 2.2, and finally propose our GP-based optimization objective in 2.4.

2.1. Notations

We are given J heterogeneous graphs where the j-th graph contains n_j vertices and is associated with an adjacency matrix $G^{(j)} \in \mathbb{R}^{n_j \times n_j}$. We use i_j to index the i_j-th vertex of graph j, and use a tuple $(i_1, \ldots, i_J)$ to index each multi-relation across the J graphs. The system predictions over all possible $\prod_{j=1}^{J} n_j$ multi-relations is summarized in an order-J tensor $f \in \mathbb{R}^{n_1 \times \cdots \times n_J}$, where $f_{i_1, i_2, \ldots, i_J}$ corresponds to the prediction about tuple $(i_1, \ldots, i_J)$.

Denote by $\otimes$ the Kronecker (Tensor) product. We use $\bigotimes_{j=1}^{J} x_j$ (or simply $\bigotimes_j x_j$) as the shorthand for $x_1 \otimes \cdots \otimes x_J$. Denote by $\times_j$ the j-mode product between tensors. We

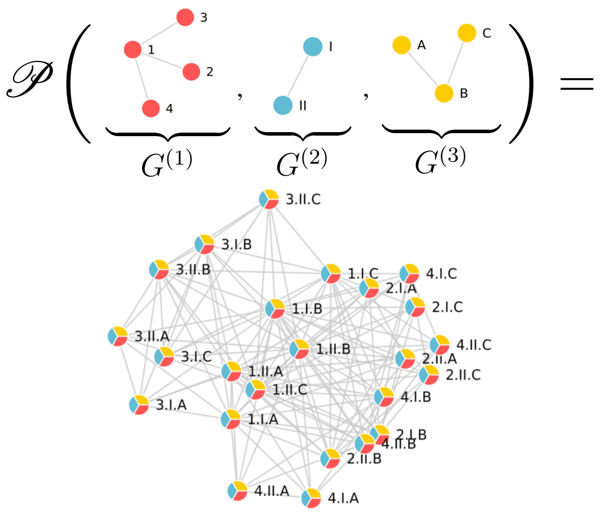

Figure 1. Product of $G^{(1)}$, $G^{(2)}$ and $G^{(3)}$. Vertices in the product graph $\mathscr{P}\left(G^{(1)}, G^{(2)}, G^{(3)}\right)$ corresponds to multi-relations across the original graphs. For instance, vertex `3.II.B` in $\mathscr{P}$ corresponds to multi-relation `(3,II,B)` across $G^{(1)}$, $G^{(2)}$ and $G^{(3)}$.

refer the readers to (Kolda & Bader, 2009) for a thorough introduction about tensor mode product.

2.2. Graph Product

In a nutshell, graph product (GP) [1] is a mapping from each cross-graph multi-relation to each vertex in a new graph $\mathscr{P}$, whose edges encode similarities among the multi-relations (illustrated in Fig. 1). A desirable property of GP is it provides a natural reduction from the original multi-relational learning problem over *heterogeneous* information sources (Task 1) to an equivalent graph-based learning problem over a *homogeneous* graph (Task 2).

Task 1. *Given J graphs $G^{(1)}, \dots, G^{(J)}$ with a small set of labeled multi-relations $\mathcal{O} = \{(i_1, \dots, i_J)\}$, predict labels of the unlabeled multi-relations.*

Task 2. *Given the product graph $\mathscr{P}\left(G^{(1)}, \dots, G^{(J)}\right)$ with a small set of labeled vertices $\mathcal{O} = \{(i_1, \dots, i_J)\}$, predict labels of its unlabeled vertices.*

2.3. Spectral Graph Product

We define a parametric family of GP operators named the spectral graph product (SGP), which is of particular interest as it subsumes the well-known Tensor GP and Cartesian GP (Table 1), is well behaved (Theorem 1) and allows efficient optimization routines (Section 3).

Let $\lambda_{i_j}^{(j)}$ and $v_{i_j}^{(j)}$ be the i_j-th eigenvalue and eigenvector for the graph j, respectively. We construct SGP by defining the eigensystem of its adjacency matrix based on the provided J heterogeneous eigensystems of $G^{(1)}, \dots, G^{(J)}$.

[1] While traditional GP only applies to two graphs, we generalize it to the case of multiple graphs (Section 2.3).

Definition 1. *The SGP of $G^{(1)}, \dots, G^{(J)}$ is a graph consisting of $\prod_j n_j$ vertices, with its adjacency matrix $\mathscr{P}_\kappa := \mathscr{P}_\kappa\left(G^{(1)}, \dots, G^{(J)}\right)$ defined by the following eigensystem*

$$\left\{\kappa\left(\lambda_{i_1}^{(1)}, \dots, \lambda_{i_J}^{(J)}\right), \bigotimes_j v_{i_j}^{(j)}\right\}_{i_1, \dots, i_J} \qquad (1)$$

where κ is a pre-specified nonnegative nondecreasing function over $\lambda_{i_j}^{(j)}, \forall j = 1, 2, \dots, J$.

In other words, the $(i_1, \dots, i_J)$-th eigenvalue of $\mathscr{P}_\kappa$ is defined by coupling the $\lambda_{i_1}^{(1)}, \dots, \lambda_{i_J}^{(J)}$ with function κ, and the $(i_1, \dots, i_J)$-th eigenvector of $\mathscr{P}_\kappa$ is defined by coupling $v_{i_1}^{(1)}, \dots, v_{i_J}^{(J)}$ via tensor (outer) product.

Remark 1. *If each individual $\left\{v_{i_j}^{(j)}\right\}_{i_j=1}^{n_j}$ forms an orthogonal basis in $\mathbb{R}^{n_j}$, $\forall j \in 1, \dots, J$, then $\left\{\bigotimes_j v_{i_j}^{(j)}\right\}_{i_1, \dots, i_J}$ forms an orthogonal basis in $\mathbb{R}^{\prod_{j=1}^J n_j}$.*

In the following example we introduce two special kinds of SGPs, assuming $J = 2$ for brevity. Higher-order cases are later summarized in Table 1.

Example 1. *Tensor GP defines $\kappa(\lambda_{i_1}, \lambda_{i_2}) = \lambda_{i_1}\lambda_{i_2}$, and is equivalent to Kronecker product: $\mathscr{P}_{\text{Tensor}}\left(G^{(1)}, G^{(2)}\right) = \sum_{i_1, i_2}(\lambda_{i_1}\lambda_{i_2})\left(v_{i_1}^{(1)} \otimes v_{i_2}^{(2)}\right)\left(v_{i_1}^{(1)} \otimes v_{i_2}^{(2)}\right)^\top \equiv G^{(1)} \otimes G^{(2)}$.*

Cartesian GP defines $\kappa(\lambda_{i_1}, \lambda_{i_2}) = \lambda_{i_1} + \lambda_{i_2}$, and is equivalent to the Kronecker sum: $\mathscr{P}_{\text{Cartesian}}\left(G^{(1)}, G^{(2)}\right) = \sum_{i_1, i_2}(\lambda_{i_1} + \lambda_{i_2})\left(v_{i_1}^{(1)} \otimes v_{i_2}^{(2)}\right)\left(v_{i_1}^{(1)} \otimes v_{i_2}^{(2)}\right)^\top \equiv G^{(1)} \oplus G^{(2)}$.

SGP Type	$\kappa\left(\lambda_{i_1}^{(1)}, \cdots, \lambda_{i_J}^{(J)}\right)$	$[\mathscr{P}_\kappa]_{(i_1, \cdots i_J), (i'_1, \cdots i'_J)}$
Tensor	$\prod_j \lambda_{i_j}^{(j)}$	$\prod_j G_{i_j, i'_j}^{(j)}$
Cartesian	$\sum_j \lambda_{i_j}^{(j)}$	$\sum_j G_{i_j, i'_j}^{(j)} \prod_{j' \neq j} \delta_{i_{j'} = i'_{j'}}$

Table 1. Tensor GP and Cartesian GP in higher-orders.

While Tensor GP and Cartesian GP provide mechanisms to associate multiple graphs in a multiplicative/additive manner, more complex cross-graph association patterns can be modeled by specifying κ. E.g., $\kappa(\lambda_{i_1}, \lambda_{i_2}, \lambda_{i_3}) = \lambda_{i_1}\lambda_{i_2} + \lambda_{i_2}\lambda_{i_3} + \lambda_{i_3}\lambda_{i_1}$ indicates pairwise associations are allowed among three graphs, but no triple-wise association is allowed as term $\lambda_{i_1}\lambda_{i_2}\lambda_{i_3}$ is not involved. Including higher order polynomials in κ amounts to incorporating higher-order associations among the graphs, which can be achieved by simply exponentiating κ.

Since what the product graph $\mathscr{P}$ offers is essentially a similarity measure among multi-relations, shuffling the order of input graphs $G^{(1)}, \dots, G^{(J)}$ should not affect $\mathscr{P}$'s topological structure. For SGP, this property is guaranteed by the following theorem:

Theorem 1 (The Commutative Property). *SGP is commutative (up to graph isomorphism) if κ is commutative.*

We omit the proof. The theorem suggests the SGP family is well-behaved as long as κ is commutative, which is true for both Tensor and Cartesian GPs as both multiplication and addition operations are order-insensitive.

2.4. Optimization Objective

It is often more convenient to equivalently write tensor f as a multi-linear map. E.g., when $J = 2$, tensor (matrix) $f \in \mathbb{R}^{n_1 \times n_2}$ defines a bilinear map from $\mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$ to $\mathbb{R}$ via $f(x_1, x_2) := x_1^\top f x_2$ and we have $f_{i_1,i_2} = f(e_{i_1}, e_{i_2})$. Such equivalence is analogous to high-order cases where f defines a multi-linear map from $\mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_J}$ to $\mathbb{R}$.

To carry out transductive learning over $\mathscr{P}_\kappa$ (Task 2), we inject the structure of the product graph into f via a Gaussian random fields prior (Zhu et al., 2003). The negative log-likelihood of the prior $-\log p\,(f \mid \mathscr{P}_\kappa)$ is the same (up to constant) as the following squared semi-norm

$$\|f\|^2_{\mathscr{P}_\kappa} = vec(f)^\top \mathscr{P}_\kappa^{-1} vec(f) \tag{2}$$

$$= \sum_{i_1, i_2, \dots, i_J} \frac{f\big(v^{(1)}_{i_1}, \dots, v^{(J)}_{i_J}\big)^2}{\kappa\big(\lambda^{(1)}_{i_1}, \dots, \lambda^{(J)}_{i_J}\big)} \tag{3}$$

Our optimization objective is therefore defined as

$$\min_{f \in \mathbb{R}^{n_1 \times \cdots \times n_J}} \ \ell_{\mathcal{O}}(f) + \frac{\gamma}{2}\|f\|^2_{\mathscr{P}_\kappa} \tag{4}$$

where $\ell_{\mathcal{O}}(\cdot)$ is a loss function to be defined later (Section 4), $\mathcal{O}$ is the set of training tuples, and γ is a tuning parameter controlling the strength of graph regularization.

3. Convex Approximation

The computational bottleneck for optimization (4) lies in evaluating $\|f\|^2_{\mathscr{P}_\kappa}$ and its first-order derivative, due to the extremely large size of $\mathscr{P}_\kappa$. In section 3.1, we first identify the computation bottleneck of using the exact formulation, based on which we propose our convex approximation scheme in 3.2 that reduces the time complexity of evaluating the semi-norm $\|f\|^2_{\mathscr{P}_\kappa}$ from $O\left(\big(\sum_j n_j\big)\big(\prod_j n_j\big)\right)$ to $O\big(\prod_j d_j\big)$, where $d_j \ll n_j$ for $j = 1, \dots, J$.

3.1. Complexity of the Exact Formulation

The brute-force evaluation of $\|f\|^2_{\mathscr{P}_\kappa}$ according to (3) costs $O\big(\big(\prod_j n_j\big)^2\big)$, as one has to evaluate $O\big(\prod_j n_j\big)$ terms inside the summation where each term costs $O\big(\prod_j n_j\big)$. However, redundancies exist and the minimum complexity for the exact evaluation is given as follows

Proposition 1. *The exact evaluation of semi-norm $\|f\|_{\mathscr{P}_\kappa}$ takes $O\big(\big(\sum_j n_j\big)\big(\prod_j n_j\big)\big)$ flops.*

Proof. Notice that the collection of all numerators in (3), namely $\big[f\big(v^{(1)}_{i_1}, \dots, v^{(J)}_{i_J}\big)\big]_{i_1,\cdots,i_J}$, is a tensor in $\mathbb{R}^{n_1 \times \cdots \times n_J}$ that can be precomputed via

$$\big(\big(f \times_1 V^{(1)}\big) \times_2 V^{(2)}\big) \cdots \times_J V^{(J)} \tag{5}$$

where $\times_j$ stands for the j-mode product between a tensor in $\mathbb{R}^{n_1 \times \cdots \times n_j \times \cdots \times n_J}$ and $V^{(j)} \in \mathbb{R}^{n_j \times n_j}$. The conclusion follows as the j-th mode product in (5) takes $O\big(n_j \prod_j n_j\big)$ flops, and one has to do this for each $j = 1, \dots, J$. When $J = 2$, (5) reduces to the multiplication of three matrices ${V^{(1)}}^\top f V^{(2)}$ at the complexity of $O\left((n_1 + n_2) n_1 n_2\right)$. □

3.2. Approximation via Tucker Form

Equation (5) implies the key for complexity reduction is to reduce the cost of the j-mode multiplications $\cdot \times_j V^{(j)}$. Such multiplication costs $O\big(n_j \prod_j n_j\big)$ in general, but can be carried out more efficiently if f is structured.

Our solution is twofold: First, we include only the top-d_j eigenvectors in $V^{(j)}$ for each graph $G^{(i)}$, where $d_j \ll n_j$. Hence each $V^{(j)}$ becomes a thin matrix in $\mathbb{R}^{n_j \times d_j}$. Second, we restrict tensor f to be within the linear span of the top $\prod_{j=1}^J d_j$ eigenvectors of the product graph $\mathscr{P}_\kappa$

$$f = \sum_{k_1,\cdots,k_J=1}^{d_1,\cdots,d_J} \alpha_{k_1,\cdots,k_J} \bigotimes_j v^{(j)}_{k_j} \tag{6}$$

$$= \alpha \times_1 V^{(1)} \times_2 V^{(2)} \times_3 \cdots \times_J V^{(J)} \tag{7}$$

The combination coefficients $\alpha \in \mathbb{R}^{d_1 \times \cdots \times d_J}$ is known as the core tensor of Tucker decomposition. In the case where $J = 2$, the above is equivalent to saying $f \in \mathbb{R}^{n_1 \times n_2}$ is a low-rank matrix parametrized by $\alpha \in \mathbb{R}^{d_1 \times d_2}$ such that $f = \sum_{k_1,k_2} \alpha_{k_1,k_2} v^{(1)}_{k_1} {v^{(2)}_{k_2}}^\top = V^{(1)} \alpha {V^{(2)}}^\top$.

Combining (6) with the orthogonality property of eigenvectors leads to the fact that $f\big(v^{(1)}_{k_1}, \dots, v^{(J)}_{k_J}\big) = \alpha_{k_1,\cdots,k_J}$. To see this for $J = 2$, notice $f\big(v^{(1)}_{k_1}, v^{(2)}_{k_2}\big) = {v^{(1)}_{k_1}}^\top f v^{(2)}_{k_1} = {v^{(1)}_{k_1}}^\top V^{(1)} \alpha {V^{(2)}}^\top v^{(2)}_{k_1} = e_{k_1}^\top \alpha e_{k_2} = \alpha_{k_1,k_2}$. Therefore the semi-norm in (2) can be simplified as

$$\|f\|^2_{\mathscr{P}_\kappa} = \|\alpha\|^2_{\mathscr{P}_\kappa} = \sum_{k_1,\dots,k_J=1}^{d_1,\cdots,d_J} \frac{\alpha^2_{k_1,\cdots,k_J}}{\kappa\big(\lambda^{(1)}_{k_1}, \dots, \lambda^{(J)}_{k_J}\big)} \tag{8}$$

Comparing (8) with (3), the number of inside-summation terms is reduced from $O\big(\prod_j n_j\big)$ to $O\big(\prod_j d_j\big)$ where

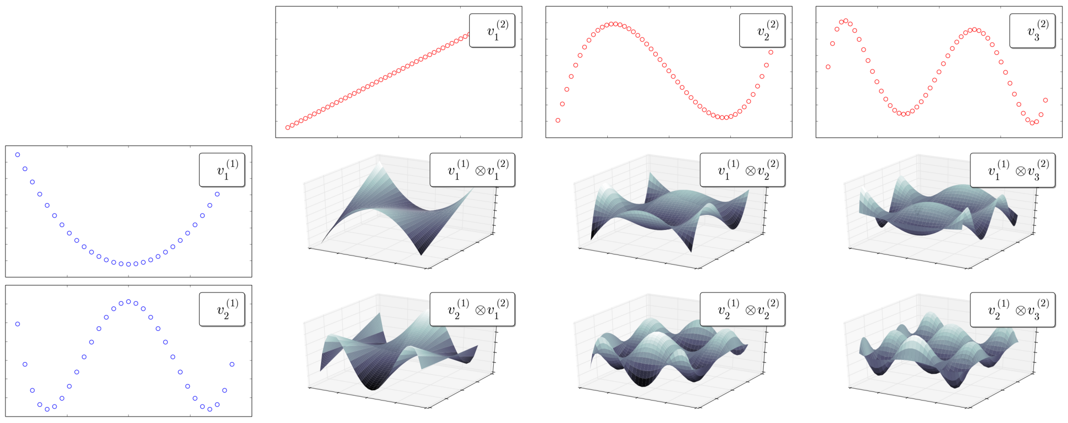

Figure 2. An illustration of the eigenvectors of $G^{(1)}$, $G^{(2)}$ and $\mathscr{P}\big(G^{(1)}, G^{(2)}\big)$. The leading nontrivial eigenvectors of $G^{(1)}$ and $G^{(2)}$ are denoted by blue and red curves, respectively. The induced leading nontrivial eigenvectors of $\mathscr{P}\big(G^{(1)}, G^{(2)}\big)$ are illustrated in 3D. If $G^{(1)}$ and $G^{(2)}$ are symmetrically normalized, their eigenvectors (corresponding to eigenvectors of the graph Laplacian) will be ordered by smoothness w.r.t. the graph structures. As a result, eigenvectors of $\mathscr{P}\big(G^{(1)}, G^{(2)}\big)$ will also be ordered by smoothness.

$d_j \ll n_j$. In addition, the cost for evaluating each term inside summation is reduced from $O\big(\prod_j n_j\big)$ to $O(1)$.

Denote by $V_{i_j}^{(j)} \in \mathbb{R}^{d_j}$ the i_j-th row of $V^{(j)}$, we obtain the following optimization by replacing f with α in (4)

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^{d_1 \times \cdots \times d_J}} \quad & \ell_{\mathcal{O}}(f) + \frac{\gamma}{2}\|\alpha\|^2_{\mathscr{P}_\kappa} \\ \text{s.t.} \quad & f = \alpha \times_1 V^{(1)} \times_2 \cdots \times_J V^{(J)} \end{aligned} \tag{9}$$

Optimization above has intuitive interpretations. In principle, it is natural to emphasis bases in f that are "smooth" w.r.t. the manifold structure of $\mathscr{P}_\kappa$, and de-emphasis those that are "nonsmooth" in order to obtain a parsimonious hypothesis with strong generalization ability. We claim this is exactly the role of regularizer (8). To see this, note any non-smooth basis $\bigotimes_j v_{k_j}^{(j)}$ of $\mathscr{P}_\kappa$ is likely to be associated with small a eigenvalue $\kappa\big(\lambda_{k_1}^{(1)}, \dots, \lambda_{k_J}^{(J)}\big)$ (illustrated in Fig. 2). The conclusion follows by noticing that $\alpha_{k_1,\dots,k_J}$ is essentially the activation strength of $\bigotimes_j v_{k_j}^{(j)}$ in f (implied by (6)), and that (8) is going to give any $\alpha_{k_1,\dots,k_J}$ associated with a small $\kappa\big(\lambda_{k_1}^{(1)}, \dots, \lambda_{k_J}^{(J)}\big)$ a stronger penalty.

(9) is a convex optimization problem over α with any convex $\ell_{\mathcal{O}}(\cdot)$. Spectral approximation techniques for graph-based learning has been found successful in standard classification tasks (Fergus et al., 2009), which are special cases under our framework when $J = 1$. We introduce this technique for multi-relational learning, which is particularly desirable as the complexity reduction will be much more significant for high-order cases ($J >= 2$).

While f in (6) is assumed to be in the Tucker form, other low-rank tensor representation schemes are potentially applicable. E.g., the Candecomp/Parafac (CP) form that further restricts α to be diagonal, which is more aggressive but substantially less expressive. The Tensor-Train decomposition (Oseledets, 2011) offers an alternative representation scheme in the middle of Tucker and CP, but the resulting optimization problem will suffer from non-convexity.

4. Optimization

Let $(x)_+ = \max(0, 1 - x)$ be the shorthand for hinge loss. We define $\ell_{\mathcal{O}}(f)$ to be the ranking ℓ_2-hinge loss

$$\ell_{\mathcal{O}}(f) = \frac{\sum_{\substack{(i_1,\dots,i_J) \in \mathcal{O} \\ (i'_1,\dots,i'_J) \in \bar{\mathcal{O}}}} \Big(f_{i_1\dots i_J} - f_{i'_1\dots i'_J}\Big)_+^2}{|\mathcal{O} \times \bar{\mathcal{O}}|} \tag{10}$$

where $\bar{\mathcal{O}}$ is the complement of $\mathcal{O}$ w.r.t. all possible multi-relations. Eq. (10) encourages the valid tuples in our training set $\mathcal{O}$ to be ranked higher than those corrupted ones in $\bar{\mathcal{O}}$, and is known to be a surrogate of AUC.

We use stochastic gradient descent for optimization as $|\mathcal{O}|$ is usually large. In each iteration, a random valid multirelation $(i_1, \dots, i_J)$ is uniformly drawn from $\mathcal{O}$, a random corrupted multirelation $(i'_1, \dots, i'_J)$ is uniformly drawn from $\bar{\mathcal{O}}$. The associated noisy gradient is computed as

$$\nabla_\alpha = \frac{\partial \ell_{\mathcal{O}}}{\partial f}\left(\frac{\partial f_{i_1,\dots,i_J}}{\partial \alpha} - \frac{\partial f_{i'_1,\dots,i'_J}}{\partial \alpha}\right) + \gamma \alpha \oslash \kappa \tag{11}$$

where we abuse the notation by defining $\kappa \in \mathbb{R}^{d_1 \times \cdots \times d_J}$, $\kappa_{k_1,\dots,k_J} := \kappa\big(\lambda_{k_1}^{(1)}, \dots, \lambda_{k_J}^{(J)}\big)$; $\oslash$ is the element-wise di-

Algorithm 1: Transductive Learning over Product Graph (TOP)

```
foreach $j \in 1, \dots, J$ do
    $\{v_k^{(j)}, \lambda_k^{(j)}\}_{k=1}^{d_j} \leftarrow$ APPROX_EIGEN$(G^{(j)})$;
foreach $(k_1, \dots, k_J) \in [d_1] \times \dots [d_J]$ do
    $\kappa_{k_1,\dots,k_J} \leftarrow \kappa(\lambda_{k_1}^{(1)}, \dots, \lambda_{k_J}^{(J)})$;
$\alpha \leftarrow 0$, $Z \leftarrow 0$;
while not converge do
    $(i_1, \dots, i_J) \overset{uni}{\sim} \mathcal{O}, \ (i'_1, \dots, i'_J) \overset{uni}{\sim} \bar{\mathcal{O}}$;
    $f_{i_1,\dots,i_J} \leftarrow \alpha \times_1 V_{i_1}^{(1)} \times_2 \cdots \times_J V_{i_J}^{(J)}$;
    $f_{i'_1,\dots,i'_J} \leftarrow \alpha \times_1 V_{i'_1}^{(1)} \times_2 \cdots \times_J V_{i'_J}^{(J)}$;
    $\delta = f_{i_1,\dots,i_J} - f_{i'_1,\dots,i'_J}$;
    if $\delta < 1$ then
        $\nabla_\alpha \leftarrow 2(\delta - 1)\Big(\bigotimes_j V_{i_j}^{(j)} - \bigotimes_j V_{i'_j}^{(j)}\Big) + \gamma\alpha \oslash \kappa$;
    else
        $\nabla_\alpha \leftarrow \gamma\alpha \oslash \kappa$;
    $Z \leftarrow Z + \nabla_\alpha^{\odot 2}$;
    $\alpha \leftarrow \alpha - \eta_0 Z^{\odot -\frac{1}{2}} \odot \nabla_\alpha$;
return $\alpha$
```

vision between tensors. The gradient w.r.t. α in (11) is

$$\frac{\partial f_{i_1,\dots,i_J}}{\partial \alpha} = \frac{\partial\big(\alpha \times_1 V_{i_1}^{(1)} \times_2 \cdots \times_J V_{i_J}^{(J)}\big)}{\partial \alpha} \tag{12}$$

$$= \bigotimes_j V_{i_j}^{(j)} \quad \in \mathbb{R}^{d_1 \times \dots d_J} \tag{13}$$

Each SGD iteration costs $O\big(\prod_j d_j\big)$ flops, which is independent from $n_1, n_2, \dots, n_J$. After obtaining the solution $\hat{\alpha}(\kappa)$ of optimization (9) for any given SGP $\mathscr{P}_\kappa$, our final predictions in $\hat{f}(\kappa)$ can be recovered via (6).

Following (Duchi et al., 2011), we allow adaptive step sizes for each element in α. That is, in the t-th iteration we use $\eta_{k_1,\dots,k_J}^{(t)} = \eta_0 \Big/ \Big[\sum_{\tau=1}^t \nabla\alpha_{k_1,\dots,k_J}^{(\tau)}{}^2\Big]^{\frac{1}{2}}$ as the step size for $\alpha_{k_1,\dots,k_J}$, where $\big\{\nabla\alpha_{k_1,\dots,k_J}^{(\tau)}\big\}_{\tau=0}^t$ are historical gradients associated with $\alpha_{k_1,\dots,k_J}$ and η_0 is the initial step size (set to be 1). The strategy is particularly efficient with highly redundant gradients, which is our case where the gradient is a regularized rank-2 tensor, according to (11) and (13).

In practice (especially for large J), the computation cost of tensor operations involving $\bigotimes_{j=1}^J V_{i_j}^{(j)} \in \mathbb{R}^{d_1,\dots,d_J}$ is not ignorable even if $d_1, d_2, \dots, d_J$ are small. Fortunately, such medium-sized tensor operations in our algorithm are highly parallelable over GPU. The pseudocode for our optimization algorithm is summarized in Alg. 1.

5. Experiments

5.1. Datasets

We evaluate our method on real-world data in two different domains: the Enzyme dataset (Yamanishi et al., 2008) for compound-protein interaction and the DBLP dataset of scientific publication records. Fig. 3 illustrates their heterogeneous objects and relational structures.

The Enzyme dataset has been used for modeling and predicting drug-target interactions, which contains a graph of 445 chemical compounds (drugs) and a graph of 664 proteins (targets). The prediction task is to label the unknown compound-protein interactions based on both the graph structures and a small set of 2,926 known interactions. The graph of compounds is constructed based on the SIMCOMP score (Hattori et al., 2003), and the graph of proteins is constructed based on the normalized Smith-Waterman score (Smith & Waterman, 1981). While both graphs are provided in the dense form, we converted them into sparse kNN graphs where each vertex is connected with its top 1% neighbors.

As for the DBLP dataset, we use a subset of 34,340 DBLP publication records in the domain of Artificial Intelligence (Tang et al., 2008), from which 3 graphs are constructed as:

- For the author graph ($G^{(1)}$) we draw an edge between two authors if they have coauthored an overlapping set of papers, and remove the isolated authors using a DFS algorithm. We then obtain a symmetric kNN graph by connecting each author with her top 0.5% nearest neighbors using the count of co-authored papers as the proximity measure. The resulting graph has 5,517 vertices with 17 links per vertex on average.
- For the paper graph ($G^{(2)}$) we connect two papers if both of them cite another paper, or are cited by another paper. Like $G^{(1)}$, we remove isolated papers using DFS and construct a symmetric 0.5%-NN graph. To measure the similarity of any given pair of papers, we represent each paper as a bag-of-citations and compute their cosine similarity. The resulted graph has 11,879 vertices and has an average degree of 50.
- For the venue graph ($G^{(3)}$) we connect two venues if they share similar research focus. The venue-venue similarity is measured by the total number of cross-citations in between, normalized by the size of the two venues involved. The symmetric venue graph has 22 vertices and an average degree of 7.

Tuples in the form of `(Author,Paper,Venue)` are extracted from the publication records, and there are 15,514 tuples (cross-graph interactions) after preprocessing.

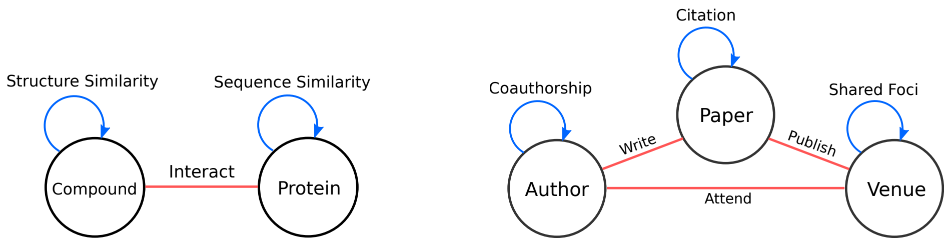

Figure 3. The heterogeneous types of objects (the circles) and the relational structures in the Enzyme (left) and DBLP (right) data sets. The blue edges represent the within-graph relations and the red edges represent the cross-graph interactions. The corresponding tuples in Enzyme is in the form of `(Compound,Protein)`, and in DBLP is in the form of `(Author,Paper,Venue)`.

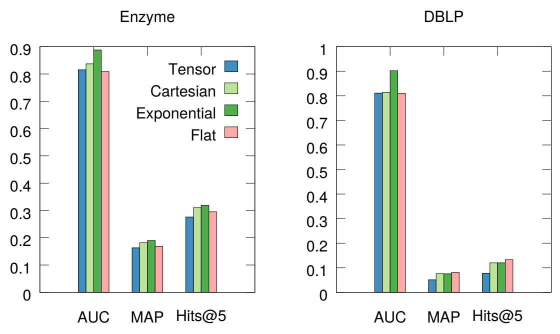

Figure 4. Performance of TOP with different SGPs.

5.2. Methods for Comparison

- Transductive Learning over Product Graph (**TOP**). The proposed method. We explore the following κ's for parametrizing the spectral graph product.

Name	$\kappa(x,y)$ $(J=2)$	$\kappa(x,y,z)$ $(J=3)$
Tensor	xy	xyz
Cartesian	$x+y$	$x+y+z$
Exponential	e^{x+y}	$e^{xy+yz+xz}$
Flat	1	1

- Tensor Factorization (**TF**) and Graph-regularized TF (**GRTF**). In TF we factorize $f \in \mathbb{R}^{n_1 \times \cdots \times n_J}$ as a set of dimensionality-reduced latent factors $C^{d_1, \times \cdots \times d_J}$, $U_1^{n_1 \times d_1}, \ldots, U_J \in \mathbb{R}^{n_J \times d_J}$. In GRTF, we further enhanced the traditional TF by adding graph regularizations to the objective function, which enforce the model to be aware of the context information in $G^{(j)}$'s (Narita et al., 2012; Cai et al., 2011);

- One-class Nearest Neighbor (**NN**). We score each tuple $(i_1, \ldots, i_J)$ in the test set with $\hat{f}(i_1, \ldots, i_J) = \max_{(i'_1, \ldots, i'_J) \in \mathcal{O}} \prod_{j=1}^{J} G_{i_j i'_j}$. That is, we assume the tuple-tuple similarity can be factorized as the product of vertex-level similarities across different graphs. We experimented with several other similarity measures and empirically found the multiplicative similarity leads to the best overall performance. Note it does not rely on the presence of any negative examples.

- Ranking Support Vector Machines (Joachims, 2002) (**RSVM**). For the task of completing the missing paper in `(Author,?,Venue)`, we use a Learning-to-Rank strategy by treating `(Author,Venue)` as the query and `Paper` as the document to be retrieved. The query feature is constructed by concatenating the eigen-features of `Author` and `Venue`, where we define the eigen-feature of vertex i_j in graph j as $V_{i_j}^{(j)} \in \mathbb{R}^{d_j}$. The feature for each query-document pair is obtained by taking the tensor product of the query feature and document eigen-feature.

- Low-rank Tensor Kernel Machines (**LTKM**). While traditional tensor-based kernel construction methods for tuples suffer from poor scalability. We propose to speedup by replacing each individual kernel with its low-rank approximation before tensor product, leading to a low-rank kernel of tuples which allows more efficient optimization routines.

For fair comparison, loss functions for TF, GRTF, RSVM and LTKM are set to be exactly the same as that for TOP, i.e. E.q. (10). All algorithms are trained using a mini-batched stochastic gradient descent.

We use the same eigensystems (eigenvectors and eigenvalues) of the $G^{(j)}$'s as the input for TOP, RSVM and LTKM. The number of top-eigenvalues/eigenvectors d_j for graph j is chosen such that $\lambda_1^{(j)}, \ldots, \lambda_{d_j}^{(j)}$ approximately cover 80% of the total spectral energy of $G^{(j)}$. With respect to this criterion, we choose $d_1 = 1,281$, $d_2 = 2,170$, $d_3 = 6$ for DBLP, and $d_1 = 150$, $d_2 = 159$ for Enzyme.

5.3. Experiment Setups

For both datasets, we randomly sample one third of known interactions for training (denoted by $\mathcal{O}$), one third for validation and use the remaining ones for testing. Known interactions in the test set, denoted by $\mathcal{T}$, are treated as positive

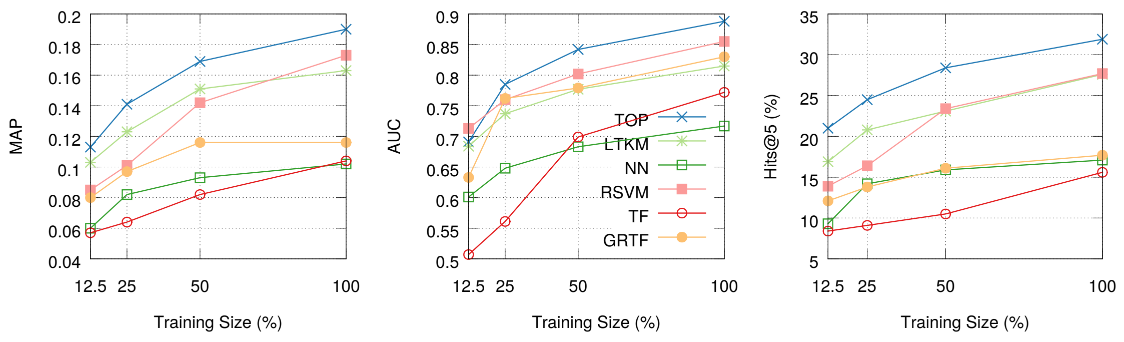

Figure 5. Test-set performance of different methods on Enzyme.

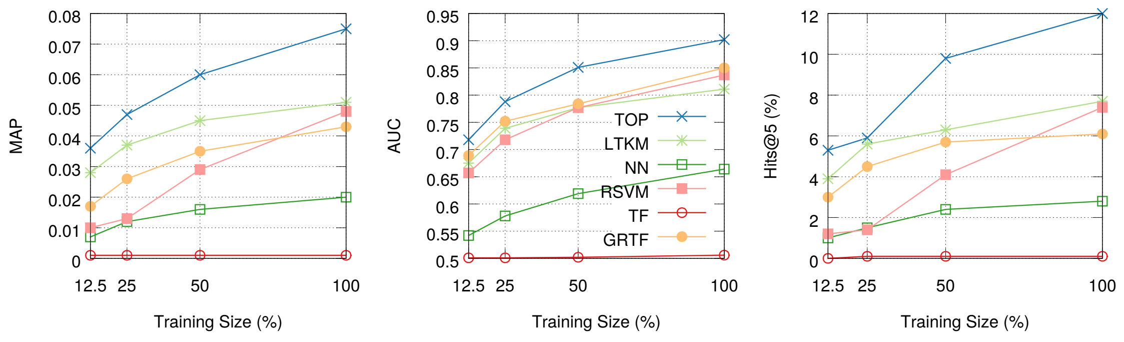

Figure 6. Test-set performance of different methods on DBLP.

examples. All tuples not in $\mathcal{T}$, denoted by $\bar{\mathcal{T}}$, are treated as negative. Tuples that are already in $\mathcal{O}$ are removed from $\bar{\mathcal{T}}$ to avoid misleading results (Bordes et al., 2013).

We measure algorithm performance on Enzyme based on the quality of inferred target proteins given each compound, namely by the ability of completing `(Compound,?)`. For DBLP, the performance is measured by the quality of inferred papers given author and venue, namely by the ability of completing `(Author,?,Venue)`. We use Mean Average Prevision (MAP), Area Under the Curve (AUC) and Hits at Top 5 (Hits@5) as our evaluation metrics.

5.4. Results

Fig. 4 compares the results of TOP with various parameterizations of the spectral graph product (SGP). Among those, Exponential κ works better on average.

Figs. 5 and 6 show the main results, comparing TOP (with Exponential κ) with other representative baselines. Clearly, TOP outperforms all the other methods on both datasets in all the evaluation metrics of MAP [2], AUC and Hit@5.

Fig. 7 shows the performance curves of TOP on Enzyme

[2]MAP scores for random guessing are 0.014 on Enzyme and 0.00072 on DBLP, respectively.

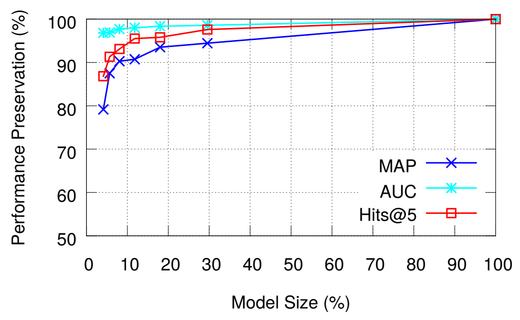

Figure 7. Performance of TOP v.s. model size on Enzyme.

over different model sizes (by varying the d_j's). With a relatively small model size compared with using the full spectrum, TOP's performance converges to the optimal point.

6. Concluding Remarks

The paper presents a novel convex optimization framework for transductive CGRL and a scalable algorithmic solution with guaranteed global optimum and a time complexity that does not depend on the sizes of input graphs. Our experiments on multi-graph data sets provide strong evidence for the superior power of the proposed approach in modeling cross-graph inference and large-scale optimization.

Acknowledgements

We thank the reviewers for their helpful comments. This work is supported in part by the National Science Foundation (NSF) under grants IIS-1216282, IIS-1350364, and IIS-1546329.

References

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.

Ben-Hur, Asa and Noble, William Stafford. Kernel methods for predicting protein–protein interactions. *Bioinformatics*, 21(suppl 1):i38–i46, 2005.

Bordes, Antoine, Usunier, Nicolas, Garcia-Duran, Alberto, Weston, Jason, and Yakhnenko, Oksana. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pp. 2787–2795, 2013.

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.

Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

Fergus, Rob, Weiss, Yair, and Torralba, Antonio. Semi-supervised learning in gigantic image collections. In *Advances in neural information processing systems*, pp. 522–530, 2009.

Getoor, Lise. *Introduction to statistical relational learning*. MIT press, 2007.

Hattori, Masahiro, Okuno, Yasushi, Goto, Susumu, and Kanehisa, Minoru. Heuristics for chemical compound matching. *Genome Informatics*, 14:144–153, 2003.

Joachims, Thorsten. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133–142. ACM, 2002.

Kolda, Tamara G and Bader, Brett W. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

Lavrac, Nada and Dzeroski, Saso. Inductive logic programming. In *WLP*, pp. 146–160. Springer, 1994.

Liu, Hanxiao and Yang, Yiming. Bipartite edge prediction via transductive learning over product graphs. In *Proceedings of The 32nd International Conference on Machine Learning*, pp. 1880–1888, 2015.

Narita, Atsuhiro, Hayashi, Kohei, Tomioka, Ryota, and Kashima, Hisashi. Tensor factorization using auxiliary information. *Data Mining and Knowledge Discovery*, 25(2):298–324, 2012.

Nickel, Maximilian, Tresp, Volker, and Kriegel, Hans-Peter. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 809–816, 2011.

Oseledets, Ivan V. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

Rendle, Steffen, Balby Marinho, Leandro, Nanopoulos, Alexandros, and Schmidt-Thieme, Lars. Learning optimal ranking with tensor factorization for tag recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 727–736. ACM, 2009.

Smith, Temple F and Waterman, Michael S. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

Tang, Jie, Zhang, Jing, Yao, Limin, Li, Juanzi, Zhang, Li, and Su, Zhong. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 990–998. ACM, 2008.

Yamanishi, Yoshihiro, Araki, Michihiro, Gutteridge, Alex, Honda, Wataru, and Kanehisa, Minoru. Prediction of drug–target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics*, 24(13): i232–i240, 2008.

Yu, Kai and Chu, Wei. Gaussian process models for link analysis and transfer learning. In *Advances in Neural Information Processing Systems*, pp. 1657–1664, 2008.

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.