# Graph Convolutional Matrix Completion for Bipartite Edge Prediction

Yuexin Wu, Hanxiao Liu and Yiming Yang
*Carnegie Mellon University, 5000 Forbes Ave, 15213, Pittsburgh, PA, U.S.A.*

Keywords: Matrix Completion, Graph Convolution, Deep Learning.

Abstract: Leveraging intrinsic graph structures in data to improve bipartite edge prediction has become an increasingly important topic in the recent machine learning area. Existing methods, however, are facing open challenges in how to enrich model expressiveness and reduce computational complexity for scalability. This paper addresses both challenges with a novel approach that uses a multi-layer/hop neural network to model a hidden space, and the first-order Chebyshev approximation to reduce training time complexity. Our experiments on benchmark datasets for collaborative filtering, citation network analysis, course prerequisite prediction and drug-target interaction prediction show the advantageous performance of the proposed approach over several state-of-the-art methods.

## 1 INTRODUCTION

Many machine learning applications can be formulated as the problem of predicting the missing edges in a bipartite graph (Kunegis et al., 2010). For example, in collaborative filtering users and items in a training set form a bipartite graph with partially observed (rated) edges, and the task is to predict the unobserved ones in the graph. Another example is in the biomedical domain, where we want to find out unknown pathogen candidates for a new drug given the known effective pathogens in the drug history (Yamanishi et al., 2008; Yamanishi et al., 2010). We call those the bipartite edge prediction (BEP) problems.

A large body of work has been devoted to solving the BEP challenge. Typical approaches treat it as a *matrix completion* problem (Candes and Recht, 2012; Salakhutdinov and Mnih, 2007; Lee and Seung, 2001). By representing observed edges using an adjacency matrix, the unobserved entries in the matrix can be discovered by imposing a low-rank constraint on the underlying model of the data. In other words, by learning a low-dimensional vector representation for each vertex, the missing entries (edges) in the adjacency matrix can be approximated using the linear combination of the observed edges. A primary drawback, or limitation, of such methods is that the prediction is solely based on the observed edges, not leveraging additional information about the vertices. For example, users' demographical information could be useful for inference about the similarity among users in collaborative filtering, and such similarity would be informative for propagating our beliefs from observed edges to unobserved edges. However, standard matrix completion methods do not leverage such side information.

Recent efforts for addressing the above limitation of BEP methods include the *Graph Regularzied Matrix Factorization* (GRMF) method (Cai et al., 2011; Gu et al., 2010) which adds a graph Laplacian regularizer in its objective function for utilizing the vertex similarities on both sides of a bipartite graph. *Transductive Learning over Product Graphs* (TOP) (Liu and Yang, 2015; Liu and Yang, 2016) is another remarkable approach, which projects the vertex-similarity graphs on both sides of the bipartite graph onto a unified product graph for semi-supervised belief propagation. Namely, by representing bipartite edges as the vertices in the product graph and by establishing the similarity among edges based on the side information, TOP propagates its beliefs from observed edges to unobserved edges over the product graph under smooth assumptions. Both GRMF and TOP improve the performance of standard matrix completion methods, according to the published evaluations; however, both have their own limitations or weaknesses. For instance, the Laplacian regularizer in GRMF is equivalent to one-hop belief propagation from each vertex to its neighbor vertices, which ignores the effects of multi-hop belief propagation as a natural phenomenon. As for TOP, although it models multi-hop propagation explicitly in various ways (using Kronecker product and spectral transformation for example), the hidden vector space of the graph product is re-

Wu, Y., Liu, H. and Yang, Y.
Graph Convolutional Matrix Completion for Bipartite Edge Prediction.
DOI: 10.5220/0006900000510060
In *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2018) - Volume 1: KDIR*, pages 51-60
ISBN: 978-989-758-330-8

stricted within the linear combination of the eigenvectors of the input similarity graphs, which could be too restrictive for some applications. Moreover, TOP requires the eigendecomposition of each input graph and the multiplication of the eigenvectors as a necessary step, which could be a computation bottleneck for scalability (see Section 3.2).

In order to address the aforementioned limitations of existing BEP methods, we propose a novel approach which employs a neural network architecture to model graph convolutions in a dimension-reduced hidden space. Specifically, by allowing the hidden representations of vertices to be located in the nonlinear space that combines the original bases of the input similarity graphs, our method captures multihop effects with a more flexible/expressive model, and at the same time enjoys its simplicity and computational efficiency by avoiding the eigendecomposition step in TOP. We also demonstrate a simple low-rank prior as the input of convolution for robust prediction. In our experiments on BEP benchmark datasets in the application domains of collaborative filtering, citation network analysis, course prerequisite prediction and drug-target interaction prediction, the proposed approach showed advantageous performance over GRMF, TOP and other representative baseline methods in most cases.

## 2 RELATED BACKGROUND

Let us introduce a generic formulation for graph-convolution based matrix completions, based on that of (Liu and Yang, 2015). We use bold upper cases for matrices, and bold lower cases for vectors.

For graphs $\mathcal{G}$ and $\mathcal{H}$, denote by $\mathcal{B}$ the bipartite graph over the node sets of $\mathcal{G}$ and $\mathcal{H}$ with $V_{\mathcal{B}} = \{V_{\mathcal{G}}, V_{\mathcal{H}}\}$ and $E_{\mathcal{B}} = V_{\mathcal{G}} \times V_{\mathcal{H}}$, and by $m = |V_{\mathcal{G}}|$ and $n = |V_{\mathcal{H}}|$ the sizes of $V_{\mathcal{G}}$ and $V_{\mathcal{H}}$, respectively. Assume $\mathcal{Y}$ is the set for edge values and edges $E_{\mathcal{B}} \in \mathcal{Y}^{m\times n}$ can be divided into the labeled part $E^l_{\mathcal{B}}$ and the unlabeled one $E^u_{\mathcal{B}}$. Then, the BEP problem is defined as:

**Problem 1.** *given $\mathcal{G}$, $\mathcal{H}$ and $E^l_{\mathcal{B}}$, find $f : E_{\mathcal{B}} \to \mathcal{Y}$ such that $f$ predicts $E^u_{\mathcal{B}}$ as accurately as possible.*

An illustration can be found in Figure 1.

Let us write $\boldsymbol{G}$ and $\boldsymbol{H}$ for the adjacency matrices of $\mathcal{G}$ and $\mathcal{H}$, respectively, and $\boldsymbol{Y} \in \mathcal{R}^{m\times n}$ as the complete adjacency matrix of $E_{\mathcal{B}}$. Also we use indicator matrix $\boldsymbol{I} \in \{0,1\}^{m\times n}$ for the observed edges in $E^l_{\mathcal{B}}$, and $\boldsymbol{F} \in \mathcal{R}^{m\times n}$ for the predicted bipartite edges by the system. Then we can formulate our objective in

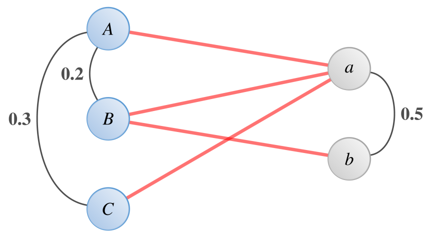

Figure 1: Illustration of the BEP problem. Given the intrinsic structures of $\mathcal{G}$ (left) and $\mathcal{H}$ (right) about the similarity information and partially observed edges (labeled in red), we want to predict whether the missing edges are valid.

matrix completion as the following:

$$\min_{\boldsymbol{F}} \quad L_{\boldsymbol{I}}(\boldsymbol{F}, \boldsymbol{Y}) \tag{1}$$

$$\text{such that } \boldsymbol{F} = \boldsymbol{U}\boldsymbol{V}^\top \tag{2}$$

$$\boldsymbol{U}, \boldsymbol{V} \in \Omega_{\boldsymbol{G},\boldsymbol{H}} \tag{3}$$

The objective denotes the empirical loss between prediction $\boldsymbol{F}$ and ground truth $\boldsymbol{Y}$ based on observed data $\boldsymbol{I}$. The first constraint assumes that $\boldsymbol{F}$ should have a low-rank representation with $\boldsymbol{U} \in \mathcal{R}^{m\times d}$, $\boldsymbol{V} \in \mathcal{R}^{n\times d}$ and $d < \min\{m,n\}$. This dimension constraint which pushes the hidden space to focus only on the principal components allows the possibility of projecting two vertices into similar embeddings even if they have minor disagreed linkages. The second constraint requires the embeddings of vertices $\boldsymbol{U}, \boldsymbol{V}$ to lie within the subspace which is induced by the manifold structures of $\boldsymbol{G}$ and $\boldsymbol{H}$ (i.e. $\Omega_{\boldsymbol{G},\boldsymbol{H}}$). Here we slightly abuse the notation of $\boldsymbol{G}$ to refer to the structure of $\mathcal{G}$. This constraint basically influences the final prediction through injecting similarity information into the hidden representations, such that vertices have to be close in the latent space if the vertices themselves are similar based on the information from $\boldsymbol{G}, \boldsymbol{H}$.

It is important to notice that how to further specify Equation 3 would make fundamental differences among methods and their performance in BEP tasks. In TOP (Liu and Yang, 2016), for example, $\boldsymbol{U}, \boldsymbol{V} \in \Omega_{\boldsymbol{G},\boldsymbol{H}}$ lies in the linear span of the (top-ranking) eigenvectors of $\boldsymbol{G}$ and $\boldsymbol{H}$. In our approach in this paper, $\boldsymbol{U}, \boldsymbol{V} \in \Omega_{\boldsymbol{G},\boldsymbol{H}}$ lies in an enriched space which is both more expressive in terms of modeling and more efficient with respect to the training-time complexity (see more discussions in Section 4).

## 3 RELATED MATRIX COMPLETION METHODS

We outline several competing methods in matrix completion, focusing on their constraints in Equation 3.

## 3.1 Hyper-ball Constraint

One common type of constraints is the ball-shape constraint. In Probabilistic Matrix Factorization (Salakhutdinov and Mnih, 2007) (PMF), for instance, the authors specify the priors of $\boldsymbol{U}$ and $\boldsymbol{V}$ to be Gaussian:

$$\|\boldsymbol{U}\|_F^2 \leq c_1, \|\boldsymbol{V}\|_F^2 \leq c_2 \tag{4}$$

where $\|\cdot\|_F$ is the Frobenius norm and $c_1, c_2$ are manually set hyperparameters. This form suggests that the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ reside within hyper-balls of radii $\sqrt{c_1}$ and $\sqrt{c_2}$ respectively. Though effectively limiting the size of the factorized representations, such constraints do not utilize any prior information from the intrinsic structures of $\boldsymbol{G}$ and $\boldsymbol{H}$, which makes it unable to produce valid embeddings for vertices that have cold-start problems (vertices having few or no linkages).

One fix for this limitation is to employ a different graph-related ball-shape constraint. Graph Regularized Matrix Factorization (Cai et al., 2011; Gu et al., 2010) (GRMF), therefore, defines the constraint as follows:

$$\operatorname{Tr}\left(\boldsymbol{U}^\top \boldsymbol{L_G} \boldsymbol{U}\right) \leq c_1, \operatorname{Tr}\left(\boldsymbol{V}^\top \boldsymbol{L_H} \boldsymbol{V}\right) \leq c_2 \tag{5}$$

where $\operatorname{Tr}(\cdot)$ denotes the trace, $\boldsymbol{L_G}$ is the unnormalized graph Laplacian defined as $\boldsymbol{L_G} = \boldsymbol{D_G} - \boldsymbol{G}$, where $(D_{\boldsymbol{G}})_{ii} = \sum_j G_{ij}$ and $\boldsymbol{L_H}$ has similar meanings. Since $\operatorname{Tr}\left(\boldsymbol{U}^\top \boldsymbol{L_G} \boldsymbol{U}\right) = \sum_i \boldsymbol{u}_i^\top \boldsymbol{L_G} \boldsymbol{u}_i$, it is meaningful to consider the effect of each individual term in the summation. Actually, we can show that:

$$\boldsymbol{u}_i^\top \boldsymbol{L_G} \boldsymbol{u}_i = \frac{1}{2} \sum_{jj'} (U_{ji} - U_{j'i})^2 G_{jj'} \tag{6}$$

This indicates that vertices $j$ and $j'$ should have close embeddings with respect to the $i$-th dimension. And this graph Laplacian plays as a smooth measure along such dimension.

Meanwhile, we can also relate this constraint to Equation 4. It is easy to prove that $\boldsymbol{L_G}$ and $\boldsymbol{L_H}$ are positive semi-definite, which means we can write the constraint similarly as:

$$\|\sqrt{\boldsymbol{L_G}}\boldsymbol{U}\|_F^2 \leq c_1, \|\sqrt{\boldsymbol{L_H}}\boldsymbol{V}\|_F^2 \leq c_2 \tag{7}$$

This suggests that the constraint is still of a hyper-ball shape after a linear transformation of columns in $\boldsymbol{U}$ and $\boldsymbol{V}$.

One drawback of this constraint is that the penalty is based on the one-hop similarity alone (i.e. $j$'s coordinates are influenced only by its direct neighbors as indicated by $G_{jj'}$). The information may not be enough to make effective regularization if the number of a vertex's neighbors is limited. e.g. in collaborative filtering, a person is hard to get enough useful recommendation if he/she has a short item-review history *and* few similar friends.

## 3.2 Subspace Constraints

Other constraints can also be used to limit the subspace, instead of that on the hyper-balls. Transductive Learning over Product Graphs (Liu and Yang, 2016) (TOP), for example, imposes eigen-space related constraints:

$$\boldsymbol{u}_i \in \operatorname{img}(\operatorname{eigen}(\boldsymbol{G})), \boldsymbol{v}_j \in \operatorname{img}(\operatorname{eigen}(\boldsymbol{H})) \tag{8}$$

where $\boldsymbol{u}_i$ is any column of $\boldsymbol{U}$ and $\boldsymbol{v}_j$ is any column of $\boldsymbol{V}$. Here $\operatorname{img}(\operatorname{eigen}(\boldsymbol{G}))$ denotes the image or equivalently the subspace spanned by the columns of $\operatorname{eigen}(\boldsymbol{G})$. These constraints are imposed as a result of the eigen-transformation in TOP. Different from the one-hop penalty in Equation 7, TOP enables multi-hop information propagation through spectral transforming which generally amounts to exponential transformation of the eigen-space $\operatorname{eigen}(\boldsymbol{G})$ and $\operatorname{eigen}(\boldsymbol{H})$. Eigen-vectors $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ typically have small coefficients within the coordinate system, similar to that with the hyper-ball constraint.

One drawback of TOP is the running time. That is, it requires computation for eigen-decompositions of $\boldsymbol{G}$ and $\boldsymbol{H}$ which are very costly when the graphs are large. Evaluating $\boldsymbol{u}_i$ in Equation 8 is also expensive, i.e. given a set of coordinates in the space of $\operatorname{eigen}(\boldsymbol{G})$, the evaluation for $\boldsymbol{u}_i$ will take $O(m^2)$ since the eigen-matrix is always dense even if $\boldsymbol{G}$ is sparse. Another limitation of TOP is that it only takes a linear transformation of the eigensystems, which may not be sufficiently expressive. Our proposed approach (next section) addresses both of the issues in TOP.

## 4 OUR APPROACH

In this section, we propose a new model which is more flexible and expressive for multi-hop nonlinear modeling of graph convolution. We start with a rather simple linear constraint on the sub-spaces (Section 4.1), and then show how to generalize it to a richer nonlinear multi-hop framework via graph convolution (Section 4.2) and nonlinear transformation (Section 4.4), with the first order Chebyshev approximation for efficient computation (Section 4.3). In addition, we show how to construct the initial input signals for the convolution in Section 4.5, and the over-all algorithm in Section 4.6.

## 4.1 Simple Subspace Constraints

Our method resides in the category of subspace constraints. Instead of requiring the embedding $U$ and $V$ to lie within the eigen-spaces of $\boldsymbol{G}$ or $\boldsymbol{H}$, we only require it to be in the span of the column space. Formally, we define the constraint to be: for any column $\boldsymbol{u}_i$ of $\boldsymbol{U}$ and column $\boldsymbol{v}_j$ of $\boldsymbol{V}$

$$\boldsymbol{u}_i \in \text{img}(\boldsymbol{G}), \boldsymbol{v}_j \in \text{img}(\boldsymbol{H}) \tag{9}$$

Alternatively, we can use a normalized version:

$$\boldsymbol{u}_i \in \text{img}(\boldsymbol{D}_{\boldsymbol{G}}^{-1/2}\boldsymbol{G}\boldsymbol{D}_{\boldsymbol{G}}^{-1/2}), \boldsymbol{v}_j \in \text{img}(\boldsymbol{D}_{\boldsymbol{H}}^{-1/2}\boldsymbol{H}\boldsymbol{D}_{\boldsymbol{H}}^{-1/2}) \tag{10}$$

We could observe that our method is computationally less costly than TOP. That is, given the coordinates (e.g. $\boldsymbol{x}$) for $\text{img}(\boldsymbol{G})$, the multiplication of $\boldsymbol{G}$ and $\boldsymbol{x}$ (i.e. $\boldsymbol{u}_i = \boldsymbol{G}\boldsymbol{x}$) only costs $O(\text{nnz}(\boldsymbol{G}))$ where nnz is the number of non-zero entries. Besides, when $\boldsymbol{G}$ is nearly fully-rank, the column space would have close similarity as the eigen-space, which means our method could enjoy similar spectral transformation power at less cost.

## 4.2 Generalization via Graph Convolution

For more expressive embeddings, we can extend the constraint in Equation 10 through graph convolution.

First, note that Equation 10 could be viewed as a transformation over a given signal $\boldsymbol{x} \in \mathcal{R}^m$ (scaler features on each node of of graph $\mathcal{G}$):

$$\boldsymbol{u}_i = \boldsymbol{D}_{\boldsymbol{G}}^{-1/2}\boldsymbol{G}\boldsymbol{D}_{\boldsymbol{G}}^{-1/2}\boldsymbol{x} \tag{11}$$

$$= \boldsymbol{U}_{\boldsymbol{G}}\Lambda_{\boldsymbol{G}}\boldsymbol{U}_{\boldsymbol{G}}^T\boldsymbol{x} \tag{12}$$

where $\boldsymbol{U}_{\boldsymbol{G}}$ and $\Lambda_{\boldsymbol{G}}$ are the corresponding eigen-vector matrix and eigen-value matrix for normalized $\boldsymbol{G}$. The $\Lambda_{\boldsymbol{G}}$ part specifies the scaling factor, which is fixed for a given $\boldsymbol{G}$. It is thus beneficial to replace this factor with a parameter to be learnt for more varied expressiveness:

$$\boldsymbol{u}_i = \boldsymbol{g}_\theta * \boldsymbol{x} = \boldsymbol{U}_{\boldsymbol{G}}\text{diag}(\boldsymbol{g}_\theta)\boldsymbol{U}_{\boldsymbol{G}}^T\boldsymbol{x} \tag{13}$$

where $\text{diag}(\boldsymbol{g}_\theta)$ denotes the diagonal matrix parameterized by vector $\boldsymbol{g}_\theta \in \mathcal{R}^m$ (called filter in the later literature).

This is called a one-hop generalized graph convolution over $\boldsymbol{G}$ (Hammond et al., 2011).

The benefits for this replacement are two-fold. First, by using a parameter $\text{diag}(\boldsymbol{g}_\theta)$, we could go beyond for a richer representation and even stack the expressions for multi-hop convolutions (Section 4.4). Second, the introduction of new parameter $\text{diag}(\boldsymbol{g}_\theta)$ separates the coordinate $\boldsymbol{x}$ and the learning parameters, which enables our model to use $\boldsymbol{x}$ to represent auxiliary information (Section 4.5).

However, before we continue with the benefits of generalized graph convolution, we first do an approximation in order to drop the cost in decomposing graph $\boldsymbol{G}$, which keeps the time complexity to be $O(\text{nnz}(\boldsymbol{G}) + m)$.

## 4.3 First Order Chebyshev Approximation

The original definition of graph convolution (Equation 13) requires to solve the eigen-decomposition for $\boldsymbol{G}$ in the first place (e.g. in TOP), which can be expensive for large graphs. Even if $\boldsymbol{G}$ is sparse, its eigenmatrix will not be sparse. Hence the evaluation of Equation 13 will take time $O(m^2)$ for the dense matrix multiplication. This will lead to a computation bottleneck, as for optimizing $\boldsymbol{g}_\theta$ through gradient descent, we need to perform this evaluation in every iteration.

In order to fix this problem, (Kipf and Welling, 2016) proposed a first order Chebyshev approximation for this calculation. Denoting by $\tilde{\boldsymbol{G}} = \boldsymbol{G} + \boldsymbol{I}$ the adjacency matrix of the graph with self-loops added, and by $\tilde{\boldsymbol{D}}_{\boldsymbol{G}}$ the diagonal matrix with $(\tilde{D}_{\boldsymbol{G}})_{ii} = \sum_j \tilde{G}_{ij}$, the approximated convolution operation can be written as:

$$\boldsymbol{u}_i = \boldsymbol{g}_\theta * \boldsymbol{x} \approx g_{\theta 1}\tilde{\boldsymbol{D}}_{\boldsymbol{G}}^{-1/2}\tilde{\boldsymbol{G}}\tilde{\boldsymbol{D}}_{\boldsymbol{G}}^{-1/2}\boldsymbol{x} \tag{14}$$

where $g_{\theta 1}$ is the first component of $\boldsymbol{g}_\theta$. We see that this approximation no longer requires the eigen-decomposition and the computation time is reduced from $O(m^2)$ to $O(\text{nnz}(\tilde{\boldsymbol{G}})) \leq O(\text{nnz}(\boldsymbol{G}) + m)$.

Moreover, we can naturally write a compact matrix representation for $\boldsymbol{U}$ convoluting over multiple filters as:

$$\boldsymbol{U} = \tilde{\boldsymbol{D}}_{\boldsymbol{G}}^{-1/2}\tilde{\boldsymbol{G}}\tilde{\boldsymbol{D}}_{\boldsymbol{G}}^{-1/2}\boldsymbol{X}\boldsymbol{\Theta} \tag{15}$$

where $\boldsymbol{X} \in \mathcal{R}^{m\times c}$ is a input signal matrix and $\boldsymbol{\Theta} \in \mathcal{R}^{c\times d}$ is the concatenated filter parameter matrix.

Note this expression enjoys the advantage of fast computation without doing decomposition on $\boldsymbol{G}$ while approximately reserving the flexible representation power as in Equation 13. We can further enhance such representation power through the following nonlinear multi-hop mechanisms.

## 4.4 Nonlinear Multi-hop Convolution

The formulation of Equation 15, can be regarded as one linear layer of feed-forward neural networks if we view $\boldsymbol{X}$ as the input and $\boldsymbol{U}$ as the feature map convoluted through graph $\boldsymbol{G}$. Therefore, natural extensions will be to add in nonlinear activation functions and stack multiple layers (LeCun et al., 2015), which will enhance the expressiveness of the model. And each layer can be regarded as an intermediate representation. For simplicity, denote $\hat{\boldsymbol{G}} = \tilde{\boldsymbol{D}}_{\boldsymbol{G}}^{-1/2}\tilde{\boldsymbol{G}}\tilde{\boldsymbol{D}}_{\boldsymbol{G}}^{-1/2}$. Then, an example of the 2-layer model for the embedding of $\boldsymbol{U}$ can take the form of:

$$\boldsymbol{U} = \tanh\left(\hat{\boldsymbol{G}}\tanh\left(\hat{\boldsymbol{G}}\boldsymbol{X}\boldsymbol{\Theta}_1\right)\boldsymbol{\Theta}_2\right) \tag{16}$$

where $\boldsymbol{\Theta}_1$ and $\boldsymbol{\Theta}_2$ are the filter parameters on each layer. Note the final layer has to be tanh instead of ReLU or sigmoid, as we need to allow the coordinates of the hidden embeddings to have negative values in order for the final product of $\boldsymbol{U}\boldsymbol{V}^\top$ to predict negative entries (or zero entries). We denote the whole structure of Equation 16 as $\text{Net}_{\boldsymbol{G}}$, that is,

$$\boldsymbol{U} = \text{Net}_{\boldsymbol{G}}(\boldsymbol{X}). \tag{17}$$

The network $\text{Net}_{\boldsymbol{H}}$ for $\boldsymbol{V}$ can be defined similarly. We use such 2-layer structure in our experiments (Table 3).

In order to distinguish between the signals on $\boldsymbol{G}$ and $\boldsymbol{H}$, we use notations $\boldsymbol{X}_{\boldsymbol{G}}$ and $\boldsymbol{X}_{\boldsymbol{H}}$ in the following literature. Similarly, $\boldsymbol{\Theta}_{\boldsymbol{G}}$ and $\boldsymbol{\Theta}_{\boldsymbol{H}}$ are used to represent parameters (concatenation of $\boldsymbol{\Theta}_1$ and $\boldsymbol{\Theta}_2$) in the corresponding multi-layer networks.

### 4.5 Construction of Input Matrices

As shown in equation 16, graph convolution starts with input signals. This means that for all the nodes in $\mathcal{G}$ and $\mathcal{H}$, we need to have input matrices $\boldsymbol{X}_{\boldsymbol{G}}$ and $\boldsymbol{X}_{\boldsymbol{H}}$ whose rows are the feature vectors of the corresponding nodes. We construct these matrices using the observed bipartite matrix $\boldsymbol{Y}_{\boldsymbol{I}} = \boldsymbol{Y} \otimes \boldsymbol{I}$, where $\otimes$ is the Hadamard product [1].

One natural way is to use the rows (or columns) of $\boldsymbol{Y}_{\boldsymbol{I}}$ for $\boldsymbol{X}_{\boldsymbol{G}}$ (or $\boldsymbol{X}_{\boldsymbol{H}}$), which essentially views the edge profiles as node features. Such an approach leverages all given information in $\boldsymbol{Y}_{\boldsymbol{I}}$. However, the computation would be too expensive when $\boldsymbol{G}$ and $\boldsymbol{H}$ are very large (e.g. with thousands of node). Moreover, this would lead to over-fitting of our model as the observed edges are typically highly sparse in the bipartite graph, not sufficient for robust estimation of models with too many free parameters.

[1]Our framework allow the flexibility of using $\boldsymbol{X}_{\boldsymbol{G}}$ and $\boldsymbol{X}_{\boldsymbol{H}}$ to represent other types of node features as well, such as vectorized demographical information about users or meta features about movies. However, in this paper we only focus on the node features induced based on the observed bipartite matrix.

Therefore, for robust induction of node features and for efficient computation, we take a simple strategy: use the top left/right singular vectors (those corresponding to the largest singular values) of $\boldsymbol{Y}_{\boldsymbol{I}}$ to construct $\boldsymbol{X}_{\boldsymbol{G}}$ and $\boldsymbol{X}_{\boldsymbol{H}}$, respectively. Since we only need to compute the top few eigenvectors of a sparse matrix instead of its full spectrum, the time/space complexities are linear in the non-zero elements in matrix $\boldsymbol{Y}_{\boldsymbol{I}}$. The dimension-reduced representation of node features should also effectively avoid overfitting.

### 4.6 Overall Algorithm

We summarize the overall training procedure and architecture in Algorithm 1 and Figure 2. Note that we do not use any regularization tricks (dropout/L1 regularization). The result shows that graph convolution and constructed low-rank input matrices have such regularization ability and performs robustly with regard to the hidden dimension (Figure 3).

Algorithm 1 : Training procedure for Graph Convolution Matrix Completion (GCMC) network.

```
Input: Bipartite Matrix $\boldsymbol{Y}$, Indicator Matrix $\boldsymbol{I}$ for training
set, Adjacency Matrices $\boldsymbol{G}$ and $\boldsymbol{H}$
$\boldsymbol{Y}_{\boldsymbol{I}} = \boldsymbol{Y} \otimes \boldsymbol{I}$
Perform low-rank SVD on $\boldsymbol{Y}_{\boldsymbol{I}}$ s.t. $\boldsymbol{X}_{\boldsymbol{G}}\boldsymbol{X}_{\boldsymbol{H}}^\top \approx \boldsymbol{Y}_{\boldsymbol{I}}$
Initialize parameters $\Theta_{\boldsymbol{G}}$, $\Theta_{\boldsymbol{H}}$ for $\text{Net}_{\boldsymbol{G}}$, $\text{Net}_{\boldsymbol{H}}$ defined in
Eq. 16
Set learning rate $\alpha$
while not converging do
    $\boldsymbol{U} = \text{Net}_{\boldsymbol{G}}(\boldsymbol{X}_{\boldsymbol{G}})$
    $\boldsymbol{V} = \text{Net}_{\boldsymbol{H}}(\boldsymbol{X}_{\boldsymbol{H}})$
    $\boldsymbol{F} = \boldsymbol{U}\boldsymbol{V}^\top$
    $\boldsymbol{\Theta}_{\boldsymbol{G}} = \boldsymbol{\Theta}_{\boldsymbol{G}} - \alpha\nabla_{\boldsymbol{\Theta}_{\boldsymbol{G}}} L_{\boldsymbol{I}}(\boldsymbol{F}, \boldsymbol{Y})$
    $\boldsymbol{\Theta}_{\boldsymbol{H}} = \boldsymbol{\Theta}_{\boldsymbol{H}} - \alpha\nabla_{\boldsymbol{\Theta}_{\boldsymbol{H}}} L_{\boldsymbol{I}}(\boldsymbol{F}, \boldsymbol{Y})$
end while
```

## 5 EXPERIMENTS

We used four benchmark datasets for evaluations in bipartite edge detection, including the applications to collaborative filtering, citation network analysis, course prerequisite prediction and drug-target interaction prediction.

### 5.1 Datasets

- **Collaborative Filtering:** *MovieLens-100K*[2] (Harper and Konstan, 2016) is a collaborative filtering benchmark where the intrinsic graphs are

[2]https://grouplens.org/datasets/movielens/100k/. We do not use any larger MovieLens dataset since no user demographic features are provided.

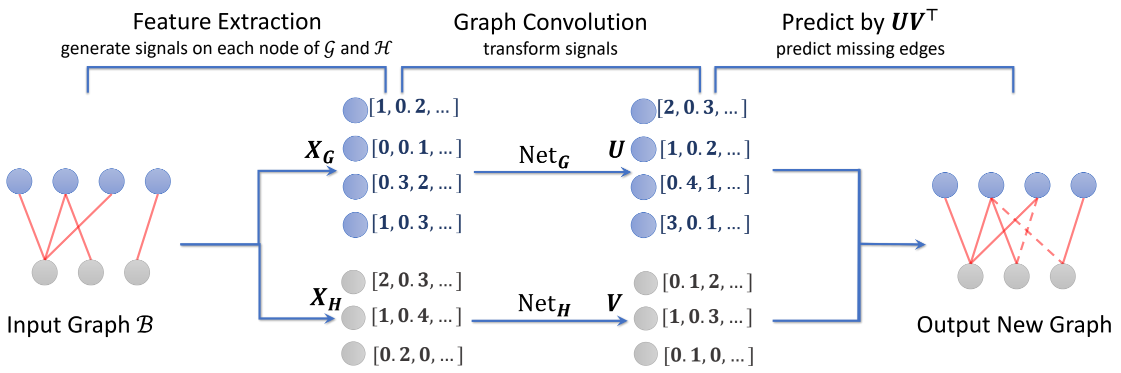

Figure 2: Architecture of the Graph Convolutional Matrix Completion (GCMC) network. The input bipartite graph $\mathcal{B}$ (equivalently observed bipartite matrix $\boldsymbol{Y_I}$) is used to extract features/signals $\boldsymbol{X_G}$ and $\boldsymbol{X_H}$ on $\mathcal{G}$ and $\mathcal{H}$. Graph convolutions are performed to transform the signals into hidden representations $\boldsymbol{U}$ and $\boldsymbol{V}$ on $\mathcal{G}$ and $\mathcal{H}$ respectively. The prediction is made by doing product between $\boldsymbol{U}$ and $\boldsymbol{V}$: $\boldsymbol{F} = \boldsymbol{UV}^\top$. $\boldsymbol{U} = \text{Net}_{\boldsymbol{G}}(\boldsymbol{X_G})$ is defined in Equation 16. Net$_{\boldsymbol{H}}$ can be defined similarly.

within users and movies. Specifically, we have $V_\mathcal{G}$ with 943 users and $V_\mathcal{H}$ with 1682 movies. The task is to predict user-movie ratings ranging from 1 to 5. Each user is provided with a binary vector indicating the gender, occupation and zip code; for each movie, the corresponding genres are provided.

- **Citation Networks:** in *Cora* (Sen et al., 2008) and *Citeseer* (Lawrence et al., 1999): Each dataset uses the publications as its $V_\mathcal{G}$ and $V_\mathcal{H}$, which are identical. The task is to predict the missing citations for a given publication. Each publication has a sparse binary feature vector, indicating whether or not a specific word is present within the publication. *Cora* contains 2708 publication records and 5429 citations and *Citeseer* contains a slightly larger publication set with 3312 documents and sparser citation records with 4715 links.
- **Course Prerequisite Prediction:** with the *Course* dataset (Yang et al., 2015). This dataset is comprised of course prerequisite data from the course 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). The task is to predict missing prerequisite dependencies among courses. Similar to citation network, $V_\mathcal{G}$ and $V_\mathcal{H}$ are identical, and for each course, a bag-of-words vector from the course description is provided.
- **Drug-target Interaction Prediction:** We use *Drug* dataset (Yamanishi et al., 2008). This dataset contains drug-target interaction data, which can be divided into 4 categories based on the corresponding target protein types: Enzymes (664 target proteins, 445 drugs), Ion Channels (204 target proteins, 210 drugs), GPCRs (95 target proteins, 223 drugs) and Nuclear Receptors (26 target proteins and 54 drugs). In this dataset, the task is to predict missing interaction pairs between target proteins and drugs. Specifically, we use $V_\mathcal{G}$ to denote the target protein node set and $V_\mathcal{H}$ for the drug node set. The similarity measures between target proteins/drugs are calculated by SIMCOMP (Hattori et al., 2003) directly on their chemical structures.

Table 1 summarizes the detailed dataset statistics.

Table 1: Dataset Statistics. $V_\mathcal{G}$ and $V_\mathcal{H}$ are the vertex sets and $E_\mathcal{B}$ denotes the edge set of the bipartite graph.

| Datasets | $\lvert V_\mathcal{G}\rvert$ | $\lvert V_\mathcal{H}\rvert$ | $\lvert E_\mathcal{B}\rvert$ | Edge Value |
|---|---|---|---|---|
| MovieLens-100K | 943 | 1,682 | 100,000 | {1...5} |
| Cora | 2,708 | 2,708 | 5,429 | {0,1} |
| Citeseer | 3,312 | 3,312 | 4,715 | {0,1} |
| Course-MIT | 2,322 | 2,322 | 1,173 | {0,1} |
| Course-CalTech | 1,048 | 1,048 | 761 | {0,1} |
| Course-CMU | 83 | 83 | 150 | {0,1} |
| Course-Princeton | 56 | 56 | 90 | {0,1} |
| Drug-Enzyme | 664 | 445 | 2926 | {0,1} |
| Drug-Ion_Channel | 204 | 210 | 1476 | {0,1} |
| Drug-GPCR | 95 | 223 | 635 | {0,1} |
| Drug-Nuclear_Receptor | 26 | 54 | 90 | {0,1} |

## 5.2 Methods to Compare

We compare our method with other major matrix completion methods from the categories of both hyper-ball constraints and subspace constraints. Also, two state-of-the-art neural network methods tailored for collaborative filtering are added as baselines.

- GCMC: graph convolutional matrix completion. This is our method[3].

[3]Code available at https://github.com/CrickWu/GCMC

- TOP (Liu and Yang, 2015; Liu and Yang, 2016): transductive learning over product-graph. This method adopts a subspace constraint in the span of eigen-matrices of $\boldsymbol{G}$ and $\boldsymbol{H}$. It utilizes spectral transformation for multi-hop feature representations.
- GRMF (Cai et al., 2011): graph regularized matrix factorization. This method employs a hyperball constraint. The ball shape is induced by the intrinsic structures of $\mathcal{G}$ and $\mathcal{H}$, which projects similar vertices into close proximity. This method considers only one-hop relations.
- PMF (Salakhutdinov and Mnih, 2007): probabilistic matrix factorization. PMF uses a hyperball constraint which is equivalent to imposing uniform Gaussian priors to the final embeddings. PMF does not utilize the information within $\mathcal{G}$ and $\mathcal{H}$.
- CF-NADE (Zheng et al., 2016): a state-of-the-art neural network based method. This framework uses a feed-forward, multilayer autoregressive architecture for collaborative filtering with an ordinal cost, which also uses nonlinear mechanism. It embeds the vertices on $\mathcal{G}$ into a lower-dimension representation. The embedding is then multiplied with a weight matrix as the prediction scores for new linkages. This method does not utilize the information of $\mathcal{G}$ and $\mathcal{H}$ either.
- RGCNN (Monti et al., 2017): another state-of-the-art neural network based method for collaborative filtering. This framework only uses side information as initialization embeddings through graph convolution, which are then finalized by a recurrent network. Side information is not utilized during the updating process of the recurrent network.

We summarize the detailed properties of comparing methods in Table 2.

Table 2: Method comparison. Side-info denotes the approach uses side information (intrinsic information from $\mathcal{G}$ and $\mathcal{H}$). Multi-hop/nonlinear denotes the approach supports multi-hop/nonlinear mechanisms. No-eigen denotes the approach does *not* need to perform expensive eigen-decomposition on $\boldsymbol{G}$ and $\boldsymbol{H}$. * using side-info partially through graph convolution as initialization.

| Methods | Side-info | Multi-hop | Nonlinear | No-eigen |
|---|---|---|---|---|
| GCMC | ✓ | ✓ | ✓ | ✓ |
| TOP | ✓ | ✓ | ✗ | ✗ |
| GRMF | ✓ | ✗ | ✗ | ✓ |
| PMF | ✗ | ✗ | ✗ | ✓ |
| CF-NADE | ✗ | ✗ | ✓ | ✓ |
| RGCNN | ✓* | ✓ | ✓ | ✓ |

## 5.3 Evaluation Metrics

In the all datasets except MovieLens-100K, the edges to be predicted have a binary value. Therefore, by treating each vertex in $\mathcal{G}$ as a query, we used the standard metric of Mean Average Precision (MAP) to measure the returned ranked list by the algorithm. On the other hand, for the collaborative filtering task on the MovieLens-100K dataset, MAP is no longer appropriate for prediction with multiple values. Instead, we used the Root Mean Square Error (RMSE) to measure the performance, which has been widely used in collaborative filtering evaluations (Zheng et al., 2016; Bennett et al., 2007; Sedhain et al., 2015). We also reported NDCG@3 on all datasets in order to evaluate the ranking properties of high-scored prediction for all methods.

We run a 5-fold cross validation for each method on each dataset. Each time 20% of the data is used for testing, 20% is used for hyper-parameter tuning, and the remaining is used for training. The results on the test sets are then averaged.

## 5.4 Empirical Settings and Parameter Tunning

Using the features of vertices in each graph, we construct sparse $k$NN graphs for both $\mathcal{G}$ and $\mathcal{H}$, and fix them for all methods to ensure they all utilize the same information.

We use the regularized versions of TOP, GRMF and PMFin our experiments. That is, instead of specifying the size of the hyper-balls, we add the Lagrangian multiplier to the original loss function. The parameter for the ratio between the original loss (Equation 1) and the constraint (Equation 3) is tunned on the validation set. Squared loss is adopted as the objective function for these 3 methods. We report the best performance from the set of $\{1e-3, 1e-2, 1e-1, 1e0, 1e1\}$. For CF-NADE, we use the implementation from the authors[4]. The learning rate and weight decay are set by cross-validation among $\{0.001, 0.0005, 0.0002\}$ and $\{0.015, 0.02\}$ as suggested in the paper (Zheng et al., 2016). For RGCNN[5], we use default parameters for MovieLens-100K, and tune the feature numbers on validation set in other datasets.

For our proposed method (GCMC), we used the major singular-vectors (singular vectors corresponding to the largest singular values) from random SVD as the input signals for $\boldsymbol{X_G}$ and $\boldsymbol{X_H}$. We use squared loss as the objective function and Adam (Kingma

[4]https://github.com/Ian09/CF-NADE

[5]https://github.com/fmonti/mgcnn

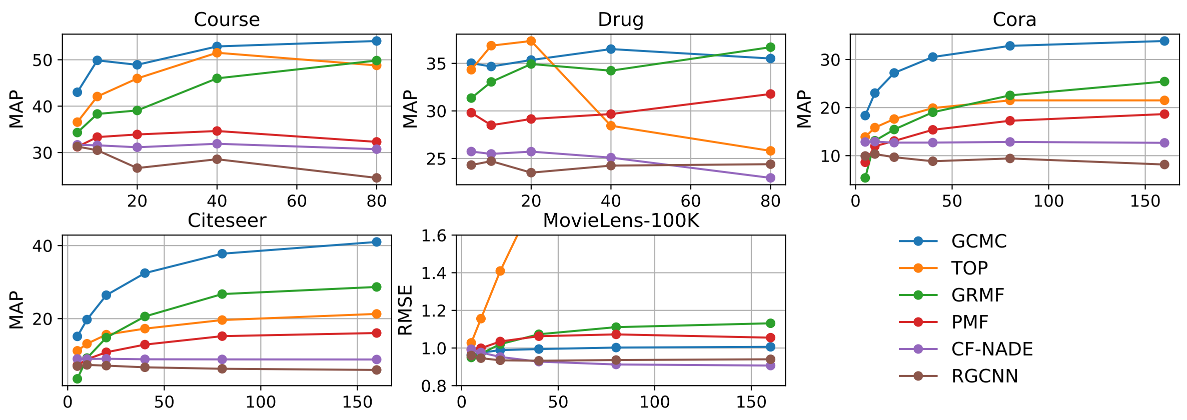

Figure 3: Performance of methods vs. the model dimensions (matrix column sizes for $\boldsymbol{U}$ and $\boldsymbol{V}$). For MAP, higher scores indicate better performances. For RMSE, lower scores indicate better performances.

and Ba, 2014) with default parameters ($b_1 = 0.1, b_2 = 0.001$ and $\varepsilon = 10^{-8}$) for optimization. For binary edge prediction tasks, (citation networks, course prerequisite prediction and drug-target interaction prediction), we multiply the loss of positive edges by a factor of 10, which has a similar effect as negative sampling strategy in most neural network algorithms and encourages more accurate prediction on the positive instances.

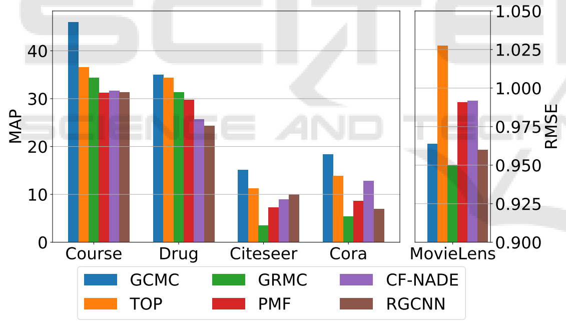

Figure 4: Result summary on all datasets when the hidden dimension is set to 5. For MAP, higher scores indicate better performances. For RMSE, lower scores indicate better performances. GCMC outperforms all the other methods in all binary prediction tasks (left sub-figure).

## 5.5 Main Results

The main results of our experiments are summarized in Table 3 and Figure 4. Clearly, our approach strictly outperforms all the other methods in the binary prediction tasks on Cora, Citeseer, Course and Drug both on MAP and NDCG@3. And the advantage is pronounced when the dataset is large enough (Cora and Citeseer), which justifies the expressiveness of our framework in data-sufficient settings. On MovieLens-100K, we see that our algorithm achieves comparable second best result in RMSE (as collaborate filtering tailored RGCNN), which is the metric all algorithms choose to optimize except CF-NADE. Since the similarity information between users and movies is highly limited, it is reasonable that simpler method GRMF which concentrates on combining major similarity features performs better. And not surprisingly, methods that utilize the intrinsic structures of graphs (GCMC and TOP) dominate the performance of the methods that do not use such information (PMF and CF-NADE) in most cases.

## 5.6 Effect of Latent Dimensions

Figure 3 shows how the performance (in MAP or RMSE) of all methods change when the hidden dimensions (i.e., the ranks of the matrices) vary. It can be seen that GCMC consistently outperforms the other methods on Cora, Citeseer and Course data, and performs as the second best method on Drug. Recall that Cora and Citeseer have highly sparse networks, i.e., with many unknown links. The excellent performance of GCMC on these datasets suggests that our approach successfully addresses the data sparse issue by effectively leveraging graph-structure based knowledge and regulating the latent representations in the model.

We also find that comparing with the other complex multi-hop algorithm, TOP, our method is more robust when the hidden dimension size changes. For instance, in datasets Course and Drug, GCMC can almost get better performance when the hidden dimension increase, while TOP easily achieves the highest score at a small dimension (40 for Course and 20 for Drug) and drops quickly. Note this phenomenon is justified since we introduce the low-rank prior in the input signals, which is effective in preventing overfitting (Section 4.5).

Table 3: Result summary on benchmark datasets. The hidden dimension was set to 5 for all the methods. The bold faces indicate the approach with the best score on each dataset. We include the detailed statistics of Drug and Course in the appendix.

| Datasets | Metric | GCMC | TOP | GRMF | PMF | CF-NADE | RGCNN |
|---|---|---|---|---|---|---|---|
| MovieLens-100K | RMSE | 0.9641 | 1.0276 | **0.9498** | 0.9907 | 0.9917 | 0.9600 |
| | NDCG@3 | 73.52 | **75.00** | 74.88 | 74.73 | 66.84 | 74.45 |
| Cora | MAP | **18.34** | 13.89 | 5.38 | 8.62 | 12.85 | 9.94 |
| | NDCG@3 | **16.80** | 12.17 | 4.27 | 7.67 | 10.99 | 9.45 |
| Citeseer | MAP | **15.14** | 11.20 | 3.53 | 7.22 | 8.97 | 7.00 |
| | NDCG@3 | **13.82** | 9.84 | 2.49 | 6.24 | 7.19 | 6.22 |
| Course | MAP | **46.02** | 36.56 | 34.32 | 31.23 | 31.62 | 31.30 |
| | NDCG@3 | **44.62** | 34.09 | 31.00 | 27.82 | 26.77 | 26.67 |
| Drug | MAP | **35.02** | 34.33 | 31.35 | 29.81 | 25.73 | 24.31 |
| | NDCG@3 | **30.96** | 30.03 | 26.77 | 24.78 | 20.97 | 19.27 |

## 6 CONCLUSION

In this paper we presented a new approach to the bipartite edge prediction problem, which uses a multi-hop neural network structure to effectively enrich the model expressiveness, and the first-order Chebyshev approximation to substantially reduce the complexity of training time. We also employ a low-rank prior in the input signals so as to make robust prediction. Our approach consistently outperformed several state-of-the-art methods in our experiments on the benchmark datasets for collaborative filtering, citation network analysis, course prerequisite prediction and drug-target interaction prediction in most cases.

## ACKNOWLEDGEMENTS

This work is supported in part by the National Science Foundation (NSF) under grant IIS-1546329.

## REFERENCES

Bennett, J., Lanning, S., et al. (2007). The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA.

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

Candes, E. and Recht, B. (2012). Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119.

Gu, Q., Zhou, J., and Ding, C. (2010). Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 199–210. SIAM.

Hammond, D. K., Vandergheynst, P., and Gribonval, R. (2011). Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150.

Harper, F. M. and Konstan, J. A. (2016). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19.

Hattori, M., Okuno, Y., Goto, S., and Kanehisa, M. (2003). Development of a chemical structure comparison method for integrated analysis of chemical and genomic information in the metabolic pathways. *Journal of the American Chemical Society*, 125(39):11853–11865.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907.*

Kunegis, J., De Luca, E. W., and Albayrak, S. (2010). The link prediction problem in bipartite networks. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems*, pages 380–389. Springer.

Lawrence, S., Giles, C. L., and Bollacker, K. D. (1999). Autonomous citation matching. In *Proceedings of the third annual conference on Autonomous Agents*, pages 392–393. ACM.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562.

Liu, H. and Yang, Y. (2015). Bipartite edge prediction via transductive learning over product graphs. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1880–1888.

Liu, H. and Yang, Y. (2016). Cross-graph learning of multi-relational associations. In *Proceedings of the 33nd International Conference on Machine Learning, ICML*

*2016, New York City, NY, USA, June 19-24, 2016*, pages 2235–2243.

Monti, F., Bronstein, M., and Bresson, X. (2017). Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*, pages 3700–3710.

Salakhutdinov, R. and Mnih, A. (2007). Probabilistic matrix factorization. In *Nips*, volume 1, pages 2–1.

Sedhain, S., Menon, A. K., Sanner, S., and Xie, L. (2015). Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, pages 111–112. ACM.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, 29(3):93.

Yamanishi, Y., Araki, M., Gutteridge, A., Honda, W., and Kanehisa, M. (2008). Prediction of drug–target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics*, 24(13):i232–i240.

Yamanishi, Y., Kotera, M., Kanehisa, M., and Goto, S. (2010). Drug-target interaction prediction from chemical, genomic and pharmacological data in an integrated framework. *Bioinformatics*, 26(12):i246–i254.

Yang, Y., Liu, H., Carbonell, J., and Ma, W. (2015). Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 159–168. ACM.

Zheng, Y., Tang, B., Ding, W., and Zhou, H. (2016). A neural autoregressive approach to collaborative filtering. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 764–773.

## APPENDIX

Statistics for the sub-tasks in Course and Drug datasets. Our method achieves the best performance on all sub-tasks except Drug-GPCR.

Table 4: Results in MAP and NDCG@3 on the subsets of the Course data: the hidden dimension was set to be 5 for all the methods. The bold faces indicate the approach with the best score on each dataset.

| Datasets | Metric | GCMC | TOP | GRMF | PMF | CF-NADE | RGCNN |
|---|---|---|---|---|---|---|---|
| Course-MIT | MAP | **35.39** | 33.64 | 31.12 | 26.10 | 30.76 | 24.16 |
| | NDCG@3 | **34.51** | 32.25 | 30.47 | 25.37 | 27.50 | 22.86 |
| Course-CalTech | MAP | **45.17** | 31.70 | 33.48 | 29.12 | 32.79 | 23.66 |
| | NDCG@3 | **43.62** | 30.45 | 32.47 | 27.79 | 29.58 | 33 |
| Course-CMU | MAP | **53.24** | 49.68 | 34.11 | 41.25 | 49.84 | 40.46 |
| | NDCG@3 | **51.26** | 46.65 | 26.92 | 37.43 | 48.29 | 33.68 |
| Course-Princeton | MAP | **50.29** | 31.21 | 38.55 | 28.46 | 13.06 | 36.9 |
| | NDCG@3 | **49.08** | 27.01 | 34.16 | 20.68 | 1.71 | 27.11 |
| Average | MAP | **46.02** | 36.56 | 34.32 | 31.23 | 31.62 | 31.30 |
| | NDCG@3 | **44.62** | 34.09 | 31.00 | 27.82 | 26.77 | 26.67 |

Table 5: Results in MAP and NDCG@3 on the subsets of the Drug data: the hidden dimension was set to be 5 for all the methods. The bold faces indicate the approach with the best score on each dataset.

| Datasets | Metric | GCMC | TOP | GRMF | PMF | CF-NADE | RGCNN |
|---|---|---|---|---|---|---|---|
| Drug-Enzyme | MAP | **12.71** | 8.81 | 6.46 | 7.60 | 6.29 | 9.94 |
| | NDCG@3 | **6.72** | 5.79 | 2.23 | 4.61 | 2.98 | 4.35 |
| Drug-Ion_Channel | MAP | **24.90** | 21.34 | 14.86 | 13.53 | 13.36 | 12.26 |
| | NDCG@3 | **19.96** | 13.87 | 7.21 | 7.42 | 6.17 | 6.32 |
| Drug-GPCR | MAP | 38.16 | 45.54 | 44.96 | **45.59** | 31.60 | 23.98 |
| | NDCG@3 | 33.95 | **44.78** | 44.58 | 44.72 | 28.97 | 21.18 |
| Drug-Nuclear_Receptor | MAP | **64.30** | 61.64 | 59.13 | 52.53 | 51.67 | 51.04 |
| | NDCG@3 | **63.21** | 55.68 | 53.07 | 42.36 | 45.74 | 45.24 |
| Average | MAP | **35.02** | 34.33 | 31.35 | 29.81 | 25.73 | 24.31 |
| | NDCG@3 | **30.96** | 30.03 | 26.77 | 24.78 | 20.97 | 19.27 |