Node-Polysemy Aware Recommendation by Matrix Completion with Side Information

Bo Hui
Auburn University
bohui@auburn.edu

Da Yan
University of Alabama at Birmingham
yanda@uab.edu

Wei-Shinn Ku
Auburn University
weishinn@auburn.edu

2021 IEEE International Conference on Big Data (Big Data) | 978-1-6654-3902-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigData52589.2021.9672000

***Abstract*—Matrix completion is a well-known approach for recommender systems. It predicts the values of the missing entries in a sparse user-item interaction matrix, based on the low-rank structure of the rating matrix. However, existing matrix completion methods do not take node polysemy and side information of social relationships into consideration, which can otherwise further improve the performance. In this paper, we propose a novel matrix completion method that employs both users' friendships and rating entries to predict the missing values in a user-item matrix. Our approach adopts a graph-based modeling where nodes are users and items, and two types of edges are considered: user friendships and user-item interactions. Polysemy-aware node features are extracted from this heterogeneous graph through a graph convolution network by considering the multifaceted factors for edge formation, which are then connected to a hybrid loss function with two heads: (1) a social-homophily head to address node polysemy, and (2) an error head for user-item rating regression. The latter is formulated on all matrix entries to combat the sensitivity of negative sampling of the vast majority of missing entries during training, with a smart technique to reduce the time complexity. Extensive experiments over real datasets verify that our model outperforms the state-of-the-art matrix completion methods by a significant margin.**

***Index Terms*—Matrix Completion; Node Polysemy; Social Networks; Recommender Systems; PU Learning**

I. Introduction

With the increasing popularity of e-commerce and social media platforms, recommender systems have drawn much attention. At the core of recommendation is the user-item rating matrix, where the items can be POIs, products, movies, etc. As each user can only interact with a small subset of items, the user-item matrix is generally sparse. Moreover, since users expressing similar ratings on multiple items tend to have similar interests for the new product, and items associated with users sharing similar interests also tend to be similar, the rating matrix exhibits a low-rank. Based on this idea, matrix completion algorithms have achieved great success [6].

However, existing matrix completion methods have not considered the node polysemy, which is an analogy to the property possessed by words in natural language. Intuitively, a user or item may belong to different factors representing different communities or interest groups at the same time. Figure 1(a) illustrates the user polysemy, where Bob and Tom are football fans and are interested in products like football and jersey, while Tom and Alice are both employees of a company in need of buying suits and ties. Note that action (Tom, Suit) is generated by factor *Job*, while (Tom, football) is generated by factor *Football*, so we can give Tom a polysemy vector

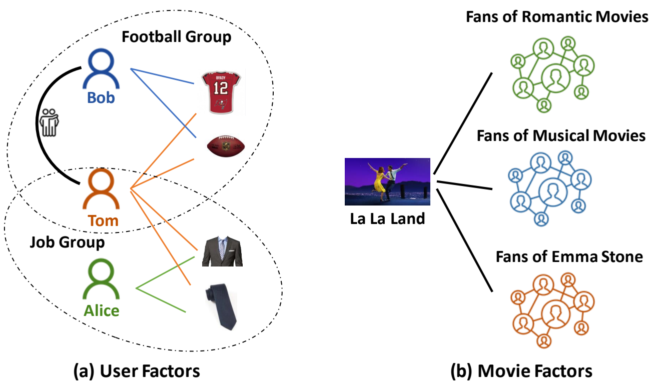


Fig. 1. Polysemy of Users and Items

(e.g., Job = 60%, $Football$ = 40%). Likewise, Figure 1(b) illustrates the item polysemy, where "La La Land" is a musical movie that tells a romantic love story starring Emma Stone, so three communities including fans of romantic movies, fans of musical movies, and fans of Emma Stone will watch "La La Land." If the three communities are all the factors, then "La La Land" has a polysemy vector (e.g., $1/3, 1/3, 1/3$). Intuitively, whether a user-item interaction happens or not can be regarded as a mixture model of K factors, where one factor will generate the interaction but with a certain probability. In this paper, we design *a social-homophily loss head* to elicit these factors and address the node-polysemy.

Modern content sharing services are often accompanied with a social network where users can become friends/followers and create/subscribe groups, giving rise to the new term "social recommender systems" [36]. Examples include YouTube [3] (video sharing), Gowalla [1] and Foursquare (check-ins), LibraryThing (book reviews) and Epinions (consumer reviews) [2]. Compared with traditional recommender systems, social relations provide an independent source to improve recommendation beyond the ratings, since a user's preference is similar to or influenced by their socially connected friends. The rationale behind this assumption can be explained by social correlation theories such as homophily [28] and social influence [27]. Existing works on social recommender systems extend low-rank factorization with social co-factorization [25], [34], social ensemble [24], [35], or social regularization [19], [22], [26]. Following the success of deep learning in CV and NLP, research on neural recommender systems aim to apply neural building blocks to model user-item interactions [43]. These works [37], [40] aim at an end-to-end training to avoid user feature engineering, by directly learning the black-box user and item embeddings for link prediction. However, they do not utilize the social network as side information for recommendation.

In this paper, we propose a new model called NP-MC (Node-Polysemy Aware Matrix Completion) which models both users' friendships and rating entries as edge links in a heterogeneous graph, to learn for each user or item, a K-dimensional polysemy vector of "soft" involvement probabilities to be in K latent social community factors. Specifically, user and item embeddings are translated by a graph convolution network (GCN) into their polysemy vectors. NP-MC integrates *the novel social-homophily loss head* with *a rating regression loss head* to train the model. The optimized polysemy vectors are used to predict unseen user-item ratings.

Another challenge lies in the sparsity of the rating matrix, where only a small fraction of user-item entries are filled with observed positive data of interactions, while most entries are unlabeled. Previous works rely on negative sampling over the unlabeled entries to compute the rating regression loss for training, but sampling is biased and previous studies [7], [38] found that the performance of negative sampling is not robust, and may not converge to the optimal ranking performance regardless of how many update steps have been taken [39]. We, therefore, formulate another regression error head over all the positive and unlabeled user-item entries, with a smart technique to combat the high time complexity.

We also carefully initialize the initial user and item embeddings using a spectral method to achieve fast convergence and avoid training from getting trapped in undesirable stationary points. Our main contributions are summarized as follows:

- We extract polysemy vectors of users and items from a heterogeneous graph considering both user-item interactions and the side information of user friendships.
- We design a node-polysemy aware social-homophily loss head by minimizing the number of cross-factor friendship/interaction edges, to learn polysemy vectors.
- We bypass the weaknesses of negative sampling in a sparse user-item matrix, by formulating the rating regression head over all user-item entries.
- We carefully initialize user and item embeddings using a spectral method to improve training performance.
- We combine the two loss heads to train the model and extensive experiments verified that NP-MC outperforms existing methods by a significant margin.

II. Model Design with Node Polysemy

A. Problem Formulation

We consider a recommender system with n users and m items, and we denote the set of users (resp. items) by U (resp. V). Since each user can only afford to interact with a small subset of all items, rating matrix $\mathbf{R}$ is highly sparse with many missing values, as illustrated by the matrix shown at the lower left corner of Figure 2. We also denote the adjacency matrix of the social network by $\mathbf{A}$ where element $\mathbf{A}_{u_i,u_j} = 1$ if friendship link (u_i, u_j) exists, and 0 otherwise.

Given the observed rating matrix $\mathbf{R}$ and the accompanied social network with adjacency matrix $\mathbf{A}$, our goal is to learn a user (resp. item) embedding matrix $\mathbf{U} \in \mathbb{R}^{n\times d}$ (resp. $\mathbf{V} \in \mathbb{R}^{m\times d}$) where each row $\mathbf{u}_i$ (resp. $\mathbf{v}_i$) corresponds to the d-dimensional embedding vector of User $u_i \in U$ (resp. Item $v_i \in V$). Once $\mathbf{U}$ and $\mathbf{V}$ are learned, for an unobserved entry $\mathbf{R}_{u_i,v_j}$ (e.g., $\mathbf{R}_{u_1,v_4}$ in Figure 2), we can estimate the rating by User u_i over Item v_j as the inner product $\hat{\mathbf{R}}_{u_i,v_j} = \langle \dot{\mathbf{u}}_i, \dot{\mathbf{v}}_j \rangle$ of the embedding vectors $\mathbf{u}_i$ and $\mathbf{v}_j$.

B. Model Overview

Step 1 is polysemy vector extraction, which learns the polysemy vectors for all users and items:

$$\mathbf{P} = [\dot{\mathbf{u}}_1, \ldots, \dot{\mathbf{u}}_n, \dot{\mathbf{v}}_1, \ldots, \dot{\mathbf{v}}_m]^T \in \mathbb{R}^{(n+m)\times K} \tag{1}$$

with $\dot{\mathbf{u}}_i \in \mathbb{R}^K$ (resp. $\dot{\mathbf{v}}_j \in \mathbb{R}^K$) being the K-dimensional polysemy vector of User u_i (resp. Item v_j). We learn $\mathbf{P}$ as a probability matrix. Formally, the k-th value of user polysemy vector $\dot{\mathbf{u}}_i$, denoted by $\dot{\mathbf{u}}_i[k]$, is defined as the probability that User u_i interacts with an item because of Factor g_k (e.g., a user community in Figure 1(a)). It is common to assume independence of users and items in the user-item interactions in the literature [8], [21], so

$$\begin{aligned} Pr(\mathbf{R}_{u_i,v_j} \,|\, g_k) &\triangleq Pr\{\mathbf{R}_{u_i,v_j} \,|\, g_k\} \\ &= Pr(\mathbf{R}_{u_i,v} \,|\, g_k) \times Pr(\mathbf{R}_{u,v_j} \,|\, g_k) \\ &= \dot{\mathbf{u}}_i[k] \times \dot{\mathbf{v}}_j[k], \end{aligned}$$

and by marginalizing out the variable g_k assuming a uniform prior probability over factors (i.e., $Pr(g_k) = \text{const.}$), we have

$$\begin{aligned} Pr(\mathbf{R}_{u_i,v_j}) &= \sum_{k=1}^{K} Pr(\mathbf{R}_{u_i,v_j} \,|\, g_k) \times Pr(g_k) \\ &= \text{const.} \times \sum_{k=1}^{K} \Big(\dot{\mathbf{u}}_i[k] \times \dot{\mathbf{v}}_j[k] \Big) \\ &\propto \dot{\mathbf{u}}_i^T \cdot \dot{\mathbf{v}}_j \triangleq \langle \dot{\mathbf{u}}_i, \dot{\mathbf{v}}_j \rangle. \end{aligned} \tag{2}$$

Assume that $\mathbf{R}_{u,v}$ is an indicator variable that equals 1 if u interacts with v, and 0 otherwise. Then, we have $E(\mathbf{R}_{u,v}) = Pr(\mathbf{R}_{u,v}) \propto \langle \dot{\mathbf{u}}, \dot{\mathbf{v}} \rangle$ according to Eq (2), justifying our design of predicting the rating estimate $\hat{\mathbf{R}}_{u,v}$ as the inner product of the polysemy vectors of User u and Item v, to find top-t item.

Step 2 is optimization, which takes the extracted polysemy matrix $\mathbf{P}$ and computes a social-homophily loss $\mathcal{L}_{homo}$ (to be detailed in Section III) and a rating error loss $\mathcal{L}_{rate} = \sum_{u\in U} \sum_{v\in V} (\mathbf{R}_{u,v} - \hat{\mathbf{R}}_{u,v})^2$ (the actual $\mathcal{L}_{rate}$ is weighted but we omit the weights in this section for simplicity). The final loss objective is computed as $\mathcal{L} = \mathcal{L}_{rate} + \lambda \cdot \mathcal{L}_{homo}$ to minimize both loss heads, where λ is a hyperparameter to adjust the importance of $\mathcal{L}_{homo}$ w.r.t. $\mathcal{L}_{rate}$.

C. Initialization of Embedding Matrices.

Carefully-designed initialization of the user and item embedding vectors is important for achieving fast convergence and avoiding getting trapped in undesirable stationary points (e.g., saddle points). Motivated by prior approaches developed for covariance estimation with missing data [23], [29], we

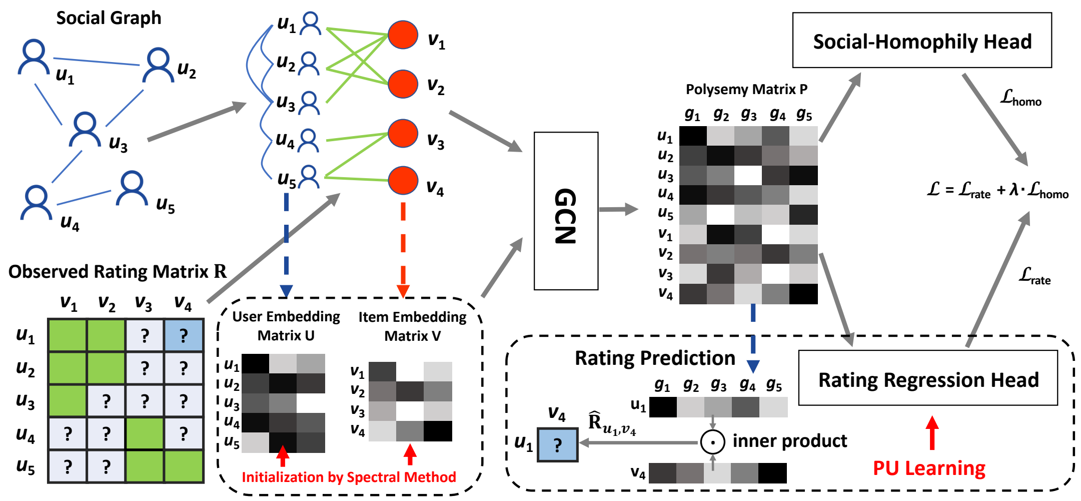


Fig. 2. NP-MC Model Overview

propose to explore the principal subspace of $\mathbf{R}\mathbf{R}^T$ and $\mathbf{R}^T\mathbf{R}$, which can be considered as the covariance metrics of users and items, respectively. However, the diagonal entries of $\mathbf{R}\mathbf{R}^T$ and $\mathbf{R}^T\mathbf{R}$ bear too much influence on the principal directions, so we follow [4] to zero out all diagonal components to operate on two new matrices denoted by $\mathcal{P}_{\text{off-diag}}(\mathbf{R}\mathbf{R}^T)$ and $\mathcal{P}_{\text{off-diag}}(\mathbf{R}^T\mathbf{R})$. Specifically, we initialize $\mathbf{U} \in \mathbb{R}^{n\times d}$ as an orthonormal matrix whose columns are the top-d eigenvectors of $\mathcal{P}_{\text{off-diag}}(\mathbf{R}\mathbf{R}^T) \in \mathbb{R}^{n\times n}$, and we initialize $\mathbf{V} \in \mathbb{R}^{m\times d}$ as an orthonormal matrix whose columns are the top-d eigenvectors of $\mathcal{P}_{\text{off-diag}}(\mathbf{R}^T\mathbf{R}) \in \mathbb{R}^{m\times m}$.

D. *Polysemy Vector Extraction by Heterogeneous GCN*

We fuse the social graph and the user-item matrix into a heterogenous graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = U \cup V$ is the set of nodes representing users and items, and $\mathcal{E}$ consists of all user friendship links (u_i, u_j) and observed user-item interactions (u_i, v_j) (regarded as a bipartite graph). The initial node features are the rows of $\mathbf{U}$ and $\mathbf{V}$, i.e., the initial user and item embeddings obtained by our spectral-based initialization.

We leverage the GCN model of [20] to extract node polysemy features (i.e., rows of $\mathbf{P}$) from the above-mentioned heterogenous graph $\mathcal{G}$, where the input node features are node embeddings given by the rows of $\mathbf{U}$ and $\mathbf{V}$. Let us denote the adjacency matrix of $\mathcal{G}$ by $\mathbf{B}$, and denote the input feature matrix $\mathbf{H}^{(0)} = [\mathbf{U}, \mathbf{V}]^T$, i.e., the vertical stacking of $\mathbf{U}$ and $\mathbf{V}$. Then, we compute $\mathbf{P}$ as:

$$\mathbf{H}^{(\ell+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{B}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}), \tag{3}$$

$$\mathbf{P} = softmax(FC(\mathbf{H}^{(L)})) \in \mathbb{R}^{(n+m)\times K}, \tag{4}$$

where $FC(.)$ is a fully-connected layer. Recall that each row of $\mathbf{P}$, is essentially a probability distribution of interaction attribution over the K user communities, so $softmax(.)$ ensures that the probabilities in every row sum to 1.

III. The Social-Homophily Loss Head

Formally, given the heterogenous graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, let us denote each node in $\mathcal{N}$ by n_i, rather than by u_i or v_i since the node can either be a user or an item. We also use $n_i \in g_k$ to denote the event that n_i is in community g_k.

We use $I(e)$ to denote an indicator variable: $I(e) = 1$ if event e occurs, and 0 otherwise. Let N_k be the number of cross-factor edges that are adjacent to g_k, we have:

$$\begin{aligned} E(N_k) &= E\left(\sum_{(n_i,n_j)\in\mathcal{E}} I(n_i \in g_k, n_j \notin g_k)\right) \\ &= \sum_{(n_i,n_j)\in\mathcal{E}} E(I(n_i \in g_k, n_j \notin g_k)) \\ &= \sum_{(n_i,n_j)\in\mathcal{E}} Pr(n_i \in g_k, n_j \notin g_k) \\ &= \sum_{(n_i,n_j)\in\mathcal{E}} \mathbf{P}_{i,k} \cdot (1 - \mathbf{P}_{j,k}) \qquad (5) \\ &= reduce\_sum\big((\mathbf{P}_{:,k}(\mathbf{1} - \mathbf{P}_{:,k})^T) \odot \mathbf{B}\big), \qquad (6) \end{aligned}$$

where (1) the outer product $\mathbf{P}_{:,k}(\mathbf{1}-\mathbf{P}_{:,k})^T \in \mathbb{R}^{(n+m)\times(n+m)}$ gives a matrix with entries $\mathbf{P}_{i,k} \cdot (1 - \mathbf{P}_{j,k})$, (2) $\odot\mathbf{B}$ is an element-wise product with the adjacency matrix $\mathbf{B}$ of the heterogeneous graph $\mathcal{G}$ to zero out elements in the outer product where $(n_i, n_j) \notin \mathcal{E}$, and (3) $reduce\_sum(.)$ computes the sum of all matrix elements, so it is not difficult to see that Eq (6) equals Eq (5) but in the form of matrix algebra.

Note that the total number of cross-factor edges in $\mathcal{G}$ can be computed as $N = \frac{1}{2}\sum_{k=1}^{K} N_k$ where factor $\frac{1}{2}$ is because each edge (n_i, n_j) with $n_i \in g_{k_1}$ and $n_i \in g_{k_2}$ is counted twice, once in N_{k_1} and once in N_{k_2}. Our objective is to minimize $E(N) = \frac{1}{2}\sum_{k=1}^{K} E(N_k)$, and by using Eq (6):

$$E(N) = \frac{1}{2} \cdot reduce\_sum((\mathbf{P}(\mathbf{1} - \mathbf{P})^T) \odot \mathbf{B}), \tag{7}$$

which be derived by observing that

$$\begin{aligned} & \mathbf{P} \cdot (\mathbf{1} - \mathbf{P})^T \\ = \; & [\mathbf{P}_{:,1}, \ldots, \mathbf{P}_{:,K}] \cdot [(\mathbf{1} - \mathbf{P}_{:,1}), \ldots, (\mathbf{1} - \mathbf{P}_{:,K})]^T \\ = \; & \sum_{k=1}^{K} \mathbf{P}_{:,k} \cdot (\mathbf{1} - \mathbf{P}_{:,k})^T. \end{aligned}$$

Directly minimizing Eq (7) is problematic, since this factor attribution scheme tends to generate a small factor g_k that is disconnected from the remaining part of $\mathcal{G}$ through a few low-degree boundary nodes n_i. Specifically, such a factor g_k only contains a small number of nodes, with a few low-degree boundary nodes $n_i \in g_k$ so that the number of edges (n_i, n_j) $(n_j \notin g_k)$ would be small. Borrowing ideas from the concept of normalized cut which is based on the graph conductance [33], [44], we normalize N_k (i.e., the expected number of nodes in g_k) by the sum of the degrees of all nodes in g_k, denoted by:

$$D_k = \sum_{n_i \in g_k} \deg(n_i), \tag{8}$$

where we use $\deg(n_i)$ to denote the degree of node n_i (i.e., the number of edges adjacent to n_i). In our problem setting, due to the node polysemy, node involvement in the factors are soft (or, probabilistic), so we minimize the objective below:

$$\mathcal{L}_{norm} = \sum_{k=1}^{K} \frac{E[N_k]}{E[D_k]}. \tag{9}$$

$$\begin{aligned} E(D_k) &= E\left(\sum_{n_i \in \mathcal{N}} \Big(I_{i,k} \cdot \deg(n_i) \Big) \right) \\ &= \sum_{n_i \in \mathcal{N}} \Big(\mathbf{P}_{i,k} \cdot \deg(n_i) \Big) \\ &= \mathbf{P}_{:,k}^T \cdot \mathbf{\Delta} \;=\; \langle \mathbf{P}_{:,k}, \mathbf{\Delta} \rangle, \end{aligned} \tag{10}$$

where $\mathbf{\Delta} = [\deg(n_1), \ldots, \deg(n_{|\mathcal{N}|})]^T$ is a column vector of node degrees in $\mathcal{G}$. Let us define:

$$\mathbf{\Gamma} \;=\; \mathbf{P}^T \mathbf{\Delta} \;\in\; \mathbb{R}^K,$$

where element $\mathbf{\Gamma}_k = \mathbf{P}_{:,k}^T \cdot \mathbf{\Delta} = E(D_k)$. Therefore, $\mathbf{\Gamma} = [E(D_1), \ldots, E(D_K)]^T$, so using Eq (6), we have:

$$\mathcal{L}_{norm} = reduce\_sum\Big(\big((\mathbf{P} \oslash \mathbf{\Gamma})(\mathbf{1} - \mathbf{P})^T \big) \odot \mathbf{B} \Big). \tag{11}$$

Since there are $|\mathcal{N}|$ nodes in $\mathcal{G}$, and there are K factors, we would like each factor to impact around $|\mathcal{N}|/K$ nodes. Note that the expected number of nodes in factor g_k is given by $\sum_{i=1}^{|\mathcal{N}|} \mathbf{P}_{i,k}$, so we add a factor-balancing regularization term

$$\sum_{k=1}^{K} \left(\sum_{i=1}^{|\mathcal{N}|} \mathbf{P}_{i,k} - \frac{|\mathcal{N}|}{K} \right)^2 = reduce\_sum \left(\mathbf{1}^T \mathbf{P} \ominus \frac{|\mathcal{N}|}{K} \right),$$

where $\ominus$ is element-wise subtraction, to $\mathcal{L}_{norm}$ to obtain our final loss function for the social-homophily head:

$$\begin{aligned} \mathcal{L}_{homo} \;=\; & reduce\_sum\Big(\big((\mathbf{P} \oslash \mathbf{\Gamma})(\mathbf{1} - \mathbf{P})^T \big) \odot \mathbf{B} \Big) \\ & + \; reduce\_sum \left(\mathbf{1}^T \mathbf{P} \ominus \frac{|\mathcal{N}|}{K} \right). \end{aligned} \tag{12}$$

IV. The Rating Error Loss Head for PU Learning

One key feature of the user-item matrix $\mathbf{R}$ is its sparsity, where the number of unobserved entries is much larger than that of the observed entry. Existing works mainly use negative sampling to sample random unlabeled entries as negative data [9], [15]. To avoid the bias caused by sampling, we formulate our rating regression head over all user-item entries. Specifically, to account for class imbalance, we define an entry (aka. sample) weight as follows:

$$w_{u,v} = \begin{cases} w^+, & \text{if entry } (u,v) \in \mathbf{R} \text{ is positive} \\ w^-, & \text{if entry } (u,v) \in \mathbf{R} \text{ is unlabeled} \end{cases}$$

Since there are many more unlabeled samples than positive ones, we usually set w^+ to be much larger than w^- so that a positive sample is as important as many unlabeled samples. Formally, we adopt the weighted SSE loss function below:

$$\mathcal{L}_{rate} = \sum_{u \in U} \sum_{v \in V} \Big(w_{u,v} \cdot (\mathbf{R}_{u,v} - \hat{\mathbf{R}}_{u,v})^2 \Big). \tag{13}$$

Directly computing this loss function suffers from a high time complexity of $O(n \times m \times K)$. To reduce the cost, we factorize $(\mathbf{R}_{u,v} - \hat{\mathbf{R}}_{u,v})^2$ and drop the constant term $w_{u,v} \cdot \mathbf{R}_{u,v}^2$, to rewrite the loss function as:

$$\begin{aligned} \mathcal{L}_{rate} &= \sum_{u \in U} \sum_{v \in V} \Big(w_{u,v} \cdot \hat{\mathbf{R}}_{u,v}^2 - 2 \cdot w_{u,v} \cdot \hat{\mathbf{R}}_{u,v} \cdot \mathbf{R}_{u,v} \Big) \\ &= \sum_{(u,v) \in \Omega^+} \Big(w^+ \cdot \hat{\mathbf{R}}_{u,v}^2 - 2 \cdot w^+ \cdot \hat{\mathbf{R}}_{u,v} \cdot \mathbf{R}_{u,v} \Big) \\ &\quad + \sum_{(u,v) \in \Omega^-} \Big(w^- \cdot \hat{\mathbf{R}}_{u,v}^2 - 2 \cdot w^- \cdot \hat{\mathbf{R}}_{u,v} \cdot 0 \Big) \\ &= \sum_{(u,v) \in \Omega^+} \Big((w^+ - w^-) \hat{\mathbf{R}}_{u,v}^2 - 2 w^+ \hat{\mathbf{R}}_{u,v} \mathbf{R}_{u,v} \Big) \\ &\quad + \; w^- \cdot \sum_{u \in U} \sum_{v \in V} \hat{\mathbf{R}}_{u,v}^2, \end{aligned} \tag{14}$$

where Ω^+ (resp. Ω^-) is the set of all positive (resp. unlabeled) entries in $\mathbf{R}$. In Eq (14), the first term is computed over the small number of positive entries and is thus efficient to compute. The bottleneck lies in computing the second term which is over all entries of $\mathbf{R}$. By replacing $\hat{\mathbf{R}}_{u,v}$, $\sum_{u \in U} \sum_{v \in V} \hat{\mathbf{R}}_{u,v}^2$ in the second term of Eq (14) can be rewritten as follows:

$$\begin{aligned} \sum_{u \in U} \sum_{v \in V} \hat{\mathbf{R}}_{u,v}^2 \;&=\; \sum_{u \in U} \sum_{v \in V} \Big(\hat{\mathbf{R}}_{u,v} \times \hat{\mathbf{R}}_{u,v} \Big) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{m} \left(\sum_{k_1=1}^{K} \Big(\dot{\mathbf{u}}_{i,k_1} \times \dot{\mathbf{v}}_{j,k_1} \Big) \times \sum_{k_2=1}^{K} \Big(\dot{\mathbf{u}}_{i,k_2} \times \dot{\mathbf{v}}_{j,k_2} \Big) \right) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{m} \left(\sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \Big(\dot{\mathbf{u}}_{i,k_1} \times \dot{\mathbf{u}}_{i,k_2} \Big) \times \Big(\dot{\mathbf{v}}_{j,k_1} \times \dot{\mathbf{v}}_{j,k_2} \Big) \right) \\ &= \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \left(\sum_{i=1}^{n} \Big(\dot{\mathbf{u}}_{i,k_1} \times \dot{\mathbf{u}}_{i,k_2} \Big) \times \sum_{j=1}^{m} \Big(\dot{\mathbf{v}}_{j,k_1} \times \dot{\mathbf{v}}_{j,k_2} \Big) \right), \end{aligned} \tag{15}$$

where $\dot{\mathbf{u}}_{ik}$ and $\dot{\mathbf{v}}_{jk}$ are entries of $\mathbf{P}$.

This reduces the time cost of computing the second term of Eq (14) from $O(n \times m \times K)$ to $O((n+m) \times K^2)$, since computing $\sum_{i=1}^{n} (\dot{\mathbf{u}}_{ik_1} \times \dot{\mathbf{u}}_{ik_2})$ takes $O(n)$ time and computing $\sum_{j=1}^{m} (\dot{\mathbf{v}}_{jk_1} \times \dot{\mathbf{v}}_{jk_2})$ takes $O(m)$ time. Since $n, m \gg K$, the time cost of computing $\mathcal{L}_{rate}$ now becomes more tractable.

V. Experiments

Experiment Setup. Three real-world recommendation system datasets are used in our experiments: LibraryThing [2], Epinions [2], Gowalla [1]. We compare NP-MC with the following state-of-the-art: SVT [5], PureSVD [10], NCF [15],

TABLE I
PERFORMANCE COMPARISON OF THE VARIOUS MODELS ON THE THREE DATASETS

Model	**LibraryThing**			**Epinions**			**Gowalla**		
	Hit@10	**MRR**	**NDCG@10**	**Hit@10**	**MRR**	**NDCG@10**	**Hit@10**	**MRR**	**NDCG@10**
SVT	0.2254	0.1100	0.1223	0.1875	0.0982	0.1050	0.3455	0.1961	0.2173
PureSVD	0.3495	0.1839	0.2066	0.1731	0.1023	0.1036	0.7655	0.4705	0.5354
NCF	0.2699	0.1272	0.1558	0.1576	0.0953	0.0874	0.7756	0.4003	0.4461
PinSage	0.3176	0.1329	0.1566	0.1762	0.0899	0.0949	0.5751	0.2272	0.2890
sRGCNN	0.2963	0.1559	0.1721	0.2044	0.1109	0.1162	0.7306	0.4644	0.4627
GC-MC	0.4402	0.2227	0.2576	0.2254	0.1249	0.1334	0.7717	0.4728	0.4981
Random-I	0.4742	0.2367	0.2777	0.1995	0.1056	0.1123	0.8851	0.4709	0.5647
One-Hot	0.4466	0.2354	0.2710	0.1509	0.0844	0.0849	0.8414	0.4153	0.5102
NS	0.4409	0.2195	0.2553	0.2231	0.1183	0.1292	0.8078	0.4129	0.5052
NP-MC	**0.5245**	**0.2722**	**0.3174**	**0.2872**	**0.1495**	**0.1683**	**0.9086**	**0.4818**	**0.5801**

PinSage [40], sRGCNN [30], GC-MC [37]. Note that graph-based models including Pinsage, sRGCNN and GC-MC are adopted to take the heterogeneous graph with social network as input for fair comparison. Three commonly used metrics for recommender systems are adopted: (1) hit ratio (HR), (2) mean reciprocal rank (MRR) and (3) normalized discounted cumulative gain (NDCG). In the experiments, we use the default value $\lambda = 0.1$ in our loss $\mathcal{L} = \mathcal{L}_{rate} + \lambda \cdot \mathcal{L}_{homo}$. We use $d = 200$ dimensions for user and item embeddings $\mathbf{u}$ and $\mathbf{v}$, which are initialized with the top-200 eigenvectors of $\mathcal{P}_{\text{off-diag}}(\mathbf{R}\mathbf{R}^T)$ and $\mathcal{P}_{\text{off-diag}}(\mathbf{R}^T\mathbf{R})$, respectively. Our GCN for computing the polysemy matrix $\mathbf{P}$ has two graph convolution layers, with the number of neural units in hidden layer and output layer being 100 and 50, respectively. The default number of factors (i.e., polysemy vector length) is $K = 30$. We set the weight of positive (resp. negative) samples as $w^+ = 0.9$ (resp. $w^- = 0.1$), respectively.

Model Performance Comparison. Table I compares the performance of our NP-MC model with the selected representative recommendation models. It is clear from Table I that NP-MC achieves the best performance and outperforms the second best (which is GC-MC) by a large margin in terms of all performance metrics. This is because NP-MC is the only method that considers node polysemy while utilizing the social network information. It demonstrates the effectiveness of introducing node polysemy to leverage the side information of social relationships to solve the matrix completion problem for recommendation. We also observe that the performance tends to be better when the sparsity of user-item matrix is low. For example, the sparsity of the matrix in Gowalla is 99.790%, which is much lower than that in Librarything and Epinions (99.996%), so we have higher Hit@10, MRR and NDCG@10 on Gowalla. It implies that with more entries being observed, the recommendation performance can be further improved.

Ablation Study. (1) Initialization. To verify the effectiveness of our spectral method for initializing $\mathbf{U}$ and $\mathbf{V}$, we replace it with (i) naïve Random-I and (ii) One-Hot. In particular, the naïve Random-I associates each node with a random vector; while the one-hot encoding indexes each node with its corresponding position being 1 and others being 0, and this high-dimensional encoding is then converted to 200-dimensions using an embedding layer. As Table I shows, replacing our spectral-based initialization with either of the two methods degrades the performance of recommendation.

(4) Learning on Whole Data. Instead of learning on the whole user-item matrix (including all unlabeled entries), we follow the strategy of negative sampling in [15] to randomly sample some negative entries, where the number of negative samples equals to that of the observed positive entries. We then redefine the rating error loss function over all the observed entries and sampled negative entries. This variant is marked as "NS". Note that our PU-learning strategy on the whole data clearly outperforms negative sampling.

We also compare the training time of our model to two variants: (i) negative sampling and (ii) computing $\mathcal{L}_2$ directly by Eq (13), the time complexity of which is $O(n \times m \times K)$ which is expensive. Note that our rearrangement method reduces the complexity to $O((n + m) \times K^2)$. Table II shows a comparison of training time. We observe many orders of magnitude speedup in the training of our method over the baseline that directly computes Eq (13). Table II also shows that our training time is only around 65% of that of negative sampling, while being more accurate as Table I has indicated.

Hyperparameter Sensitivity. In this set of experiments, we tune the value of λ (in $\mathcal{L} = \mathcal{L}_{rate} + \lambda \cdot \mathcal{L}_{homo}$) and the number of polysemy factors K to investigate their performance effects.

(1) Varying λ. Figure 3 shows the effect of the weight λ for social-homophily loss head on the overall recommendation performance. We can see that as λ increases till 0.1, the model performance improves in all metrics; but the performance degrades as λ increases further to 1. This shows that there exists a tradeoff in weighing the social-homophily regularization loss head over the regular rating error loss.

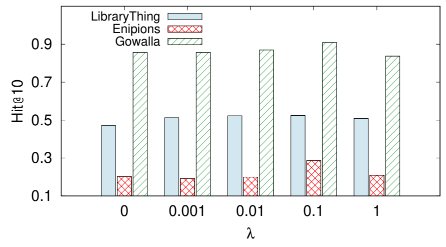
(2) Varying K. Lastly, we report the effect of factor number parameter K. Figure 4 shows the performance of our model as K varies. For all our three datasets, we see that the optimal K is around 30 to 40. We find that when $K > 40$, the performance of our model starts to degrade. This shows that the number of latent polysemy factors is around 30 to 40 in real applications, where our model has the least inductive bias.

VI. RELATED WORK

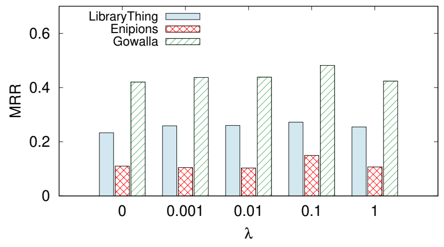
Matrix completion has been extensively studied in the context of recommender systems. With partial observations, the desired low-rank matrix can be recovered by solving a rank minimization problem. Since the rank minimization problem

TABLE II
TRAINING TIME (FOR ONE EPOCH)

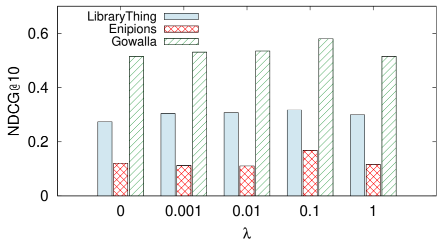
Methods	LibraryThing	Epinions	Gowalla
Computing $\mathcal{L}_2$ Directly by Eq (13)	41,902.10 ms	30,095.61 ms	37,625.27 ms
Negative Sampling	40.18 ms	11.76 ms	255.79 ms
Our Method by Eq (14) and Eq (15)	23.68 ms	7.84 ms	172.84 ms



(a) Hit@10

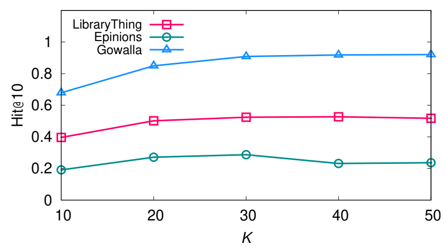


(b) MRR

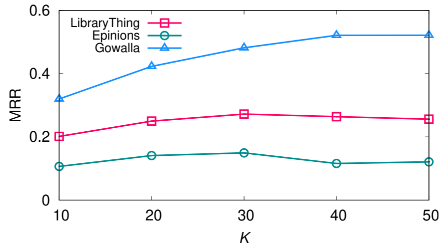


(c) NDCG@10

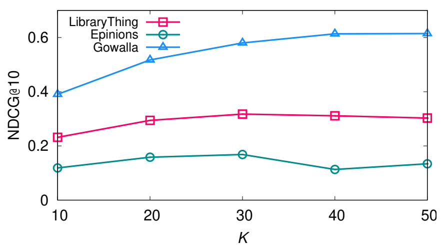
Fig. 3. Varying λ



(a) Hit@10



(b) MRR



(c) NDCG@10

Fig. 4. Varying K

is NP-hard [32] and hence intractable when the dimensions are large, a common trick is to replace the non-convex objective function with its convex surrogate, allowing efficient convex optimization. For example, the nuclear norm is a convex surrogate widely used in various methods such as SVT [5]. See [31] for a comprehension survey of these methods.

In recommender systems, a common approach to model the interaction between a user-item pair (u, v) is to compute the inner product of their latent features vectors [16]. To capture the complicated correlations between user and item that can be non-linear, recent works learn such interactions with a neural network such as in NCF [15]. The recent popularity of Graph neural networks [20], [41] has been widely used in various tasks [17], [18] It has also been considered to operate on the user-item interaction graph. Examples include PinSage [40], sRGCNN [30] and GC-MC [37] that we have reviewed and compared in Section V, among a few others [14], [42], [46].

Social network information is proven to benefit the performance of item recommendation [26], especially the social media is widely used in people's daily life. Conventional matrix completion methods either co-factorize the user-item matrix and user-user social relation matrix or add regularization terms to force a user's preference to be closer to his/her friends. See [36] for a comprehensive survey of these methods. GraphRec [12] encodes both user-item interactions and social relations with a graph neural network framework for social recommendations. However, it fails to integrate user-item interactions and social relations as a heterogeneous graph and ignores the polysemy nature of nodes. A following work [13] further considers an item-item graph and proposes an attention mechanism to enhance the recommendation. Among other works, TASRec predicts the next item that is most likely to be clicked by an anonymous user, based on the historical clicking sequence. STGN [45] uses a long-short term memory network to capture the spatio-temporal relationships between successive checkins for POI recommendation.

Besides collaborative filtering by matrix factorization, content-based filtering is another common approach for recommendation. Content-based recommender systems associate each user or item with real-valued attributes, such as text and image features, and user profiles. Item-based k-nearest-neighbor method [11] is the most representative approach to measure the similarity of two items. However, the performance of content-based methods relies on the quality of user and item features, and the content information is not always available.

VII. CONCLUSION

We proposed a neural recommendation model called NP-MC to explore social homophily with node polysemy. NP-MC utilizes social graph as side information and uses graph convolutional network to generate multi-faceted polysemy feature vectors for users and items, which are used to compute two loss heads: a social-homephily loss head and a rating regression loss head. Extensive experiments verified that NP-MC outperforms existing matrix completion methods by a large margin on various datasets.

References

[1] Location-Based Social Network Data. http://www.yongliu.org/datasets/.

[2] Social Recommendation Data. https://cseweb.ucsd.edu/~jmcauley/datasets.html#social_data.

[3] YouTube Social Network. https://snap.stanford.edu/data/com-Youtube.html.

[4] C. Cai, G. Li, H. V. Poor, and Y. Chen. Nonconvex low-rank symmetric tensor completion from noisy data. *CoRR*, abs/1911.04436, 2019.

[5] J. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.*, 20(4):1956–1982, 2010.

[6] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Commun. ACM*, 55(6):111–119, 2012.

[7] C. Chen, M. Zhang, C. Wang, W. Ma, M. Li, Y. Liu, and S. Ma. An efficient adaptive transfer neural network for social-aware recommendation. In *SIGIR*, pages 225–234. ACM, 2019.

[8] C. Chen, M. Zhang, Y. Zhang, W. Ma, Y. Liu, and S. Ma. Efficient heterogeneous collaborative filtering without negative sampling for recommendation. In *AAAI*, pages 19–26. AAAI Press, 2020.

[9] H. Chen and J. Li. Neural tensor model for learning multi-aspect factors in recommender systems. In *IJCAI*, pages 2449–2455. ijcai.org, 2020.

[10] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In X. Amatriain, M. Torrens, P. Resnick, and M. Zanker, editors, *RecSys*, pages 39–46. ACM, 2010.

[11] M. Deshpande and G. Karypis. Item-based top-*N* recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[12] W. Fan, Y. Ma, Q. Li, Y. He, Y. E. Zhao, J. Tang, and D. Yin. Graph neural networks for social recommendation. In L. Liu, R. W. White, A. Mantrach, F. Silvestri, J. J. McAuley, R. Baeza-Yates, and L. Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 417–426. ACM, 2019.

[13] W. Fan, Y. Ma, Q. Li, J. Wang, G. Cai, J. Tang, and D. Yin. A graph neural network framework for social recommendations. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.

[14] J. S. Hartford, D. R. Graham, K. Leyton-Brown, and S. Ravanbakhsh. Deep models of interactions across sets. In J. G. Dy and A. Krause, editors, *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 1914–1923. PMLR, 2018.

[15] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua. Neural collaborative filtering. In *WWW*, pages 173–182. ACM, 2017.

[16] X. He, H. Zhang, M. Kan, and T. Chua. Fast matrix factorization for online recommendation with implicit feedback. In R. Perego, F. Sebastiani, J. A. Aslam, I. Ruthven, and J. Zobel, editors, *SIGIR*, pages 549–558. ACM, 2016.

[17] B. Hui, H. Chen, D. Yan, and W. Ku. EDGE: entity-diffusion gaussian ensemble for interpretable tweet geolocation prediction. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*, pages 1092–1103. IEEE, 2021.

[18] B. Hui, D. Yan, W. Ku, and W. Wang. Predicting economic growth by region embedding: A multigraph convolutional network approach. In M. d'Aquin, S. Dietze, C. Hauff, E. Curry, and P. Cudré-Mauroux, editors, *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pages 555–564. ACM, 2020.

[19] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *RecSys*, pages 135–142. ACM, 2010.

[20] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR (Poster)*. OpenReview.net, 2017.

[21] D. Liang, L. Charlin, J. McInerney, and D. M. Blei. Modeling user exposure in recommendation. In *WWW*, pages 951–961. ACM, 2016.

[22] T. Lin, C. Gao, and Y. Li. Recommender systems with characterized social regularization. In *CIKM*, pages 1767–1770. ACM, 2018.

[23] K. Lounici et al. High-dimensional covariance matrix estimation with missing observations. *Bernoulli*, 20(3):1029–1058, 2014.

[24] H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. In *SIGIR*, pages 203–210. ACM, 2009.

[25] H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM*, pages 931–940. ACM, 2008.

[26] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *WSDM*, pages 287–296. ACM, 2011.

[27] P. V. Marsden and N. E. Friedkin. Network studies of social influence. *Sociological Methods & Research*, 22(1):127–151, 1993.

[28] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[29] A. Montanari and N. Sun. Spectral algorithms for tensor completion. *Communications on Pure and Applied Mathematics*, 71(11):2381–2425, 2018.

[30] F. Monti, M. M. Bronstein, and X. Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *NIPS*, pages 3697–3707, 2017.

[31] L. T. Nguyen, J. Kim, and B. Shim. Low-rank matrix completion: A contemporary survey. *IEEE Access*, 7:94215–94237, 2019.

[32] B. Recht, W. Xu, and B. Hassibi. Necessary and sufficient conditions for success of the nuclear norm heuristic for rank minimization. In *CDC*, pages 3065–3070. IEEE, 2008.

[33] J. Shi and J. Malik. Normalized cuts and image segmentation. In *CVPR*, pages 731–737. IEEE Computer Society, 1997.

[34] J. Tang, H. Gao, X. Hu, and H. Liu. Exploiting homophily effect for trust prediction. In *WSDM*, pages 53–62. ACM, 2013.

[35] J. Tang, H. Gao, and H. Liu. mtrust: discerning multi-faceted trust in a connected world. In *WSDM*, pages 93–102. ACM, 2012.

[36] J. Tang, X. Hu, and H. Liu. Social recommendation: a review. *Soc. Netw. Anal. Min.*, 3(4):1113–1133, 2013.

[37] R. van den Berg, T. N. Kipf, and M. Welling. Graph convolutional matrix completion. *CoRR*, abs/1706.02263, 2017.

[38] M. Wang, M. Gong, X. Zheng, and K. Zhang. Modeling dynamic missingness of implicit feedback for recommendation. In *NeurIPS*, pages 6670–6679, 2018.

[39] X. Xin, F. Yuan, X. He, and J. M. Jose. Batch IS NOT heavy: Learning word representations from all samples. In *ACL (1)*, pages 1853–1862. Association for Computational Linguistics, 2018.

[40] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, pages 974–983. ACM, 2018.

[41] J. Zhang, B. Hui, P. Harn, M. Sun, and W. Ku. smgc: A complex-valued graph convolutional network via magnetic laplacian for directed graphs. *CoRR*, abs/2110.07570, 2021.

[42] M. Zhang and Y. Chen. Inductive matrix completion based on graph neural networks. In *ICLR*. OpenReview.net, 2020.

[43] S. Zhang, L. Yao, A. Sun, and Y. Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, 52(1):5:1–5:38, 2019.

[44] Y. Zhang and K. Rohe. Understanding regularized spectral clustering via graph conductance. In *NeurIPS*, pages 10654–10663, 2018.

[45] P. Zhao, H. Zhu, Y. Liu, J. Xu, Z. Li, F. Zhuang, V. S. Sheng, and X. Zhou. Where to go next: A spatio-temporal gated network for next POI recommendation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 5877–5884. AAAI Press, 2019.

[46] L. Zheng, C. Lu, F. Jiang, J. Zhang, and P. S. Yu. Spectral collaborative filtering. In S. Pera, M. D. Ekstrand, X. Amatriain, and J. O'Donovan, editors, *RecSys*, pages 311–319. ACM, 2018.