Spectral Collaborative Filtering

Lei Zheng
Department of Computer Science
University of Illinois at Chicago
IL, US
lzheng21@uic.edu

Chun-Ta Lu
Department of Computer Science
University of Illinois at Chicago
IL, US
clu29@uic.edu

Fei Jiang
Department of Computer Science
Peking University
Beijing, China
fei.jiang1989@pku.edu.cn

Jiawei Zhang
IFM Lab, Department of Computer Science
Florida State University
FL, US
jiawei@ifmlab.org

Philip S. Yu
Department of Computer Science
University of Illinois at Chicago
IL, US
Institute for Data Science
Tsinghua University
Beijing, China
psyu@uic.edu

ABSTRACT

Despite the popularity of Collaborative Filtering (CF), CF-based methods are haunted by the *cold-start* problem, which has a significantly negative impact on users' experiences with Recommender Systems (RS). In this paper, to overcome the aforementioned drawback, we first formulate the relationships between users and items as a bipartite graph. Then, we propose a new spectral convolution operation directly performing in the *spectral domain*, where not only the proximity information of a graph but also the connectivity information hidden in the graph are revealed. With the proposed spectral convolution operation, we build a deep recommendation model called Spectral Collaborative Filtering (SpectralCF). Benefiting from the rich information of connectivity existing in the *spectral domain*, SpectralCF is capable of discovering deep connections between users and items and therefore, alleviates the *cold-start* problem for CF. To the best of our knowledge, SpectralCF is the first CF-based method directly learning from the *spectral domains* of user-item bipartite graphs. We apply our method on several standard datasets. It is shown that SpectralCF significantly outperforms state-of-the-art models. Code and data are available at https://github.com/lzheng21/SpectralCF.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Neural networks*;

KEYWORDS

Recommender Systems, Spectrum, Collaborative Filtering

RecSys '18, October 2–7, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5901-6/18/10…$15.00
https://doi.org/10.1145/3240323.3240343

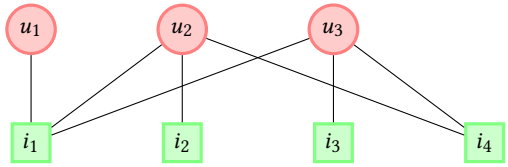


Figure 1: A toy example of a user-item bipartite graph $\mathcal{B}$ with edges representing observed user-item interactions. Red circles and green rectangles denote users and items, respectively.

ACM Reference Format:
Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S. Yu. 2018. Spectral Collaborative Filtering. In *Twelfth ACM Conference on Recommender Systems (RecSys '18), October 2–7, 2018, Vancouver, BC, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3240323.3240343

1 INTRODUCTION

The effectiveness of recommender systems (RS) often relies on how well users' interests or preferences can be understood and interactions between users and items can be modeled. Collaborative Filtering (CF) [19] is one of the widely used and prominent techniques for RS. The underlying assumption of the CF approach is that if a user u_1 shares a common item with another user u_2, u_1 is also likely to be interested in other items liked by u_2. Although CF has been successfully applied to many recommendation applications, the *cold-start* problem is considered as one of its major challenges [19]. The problem arises when a user interacted with a very small number of items. Consequently, the user shares few items with other users, and effectively recommending for the user becomes a challenging task for RS.

If we formulate the relationships between users and items as a bipartite graph[1], we argue that the connectivity information of the graph can play an important role for tackling the *cold-start* problem. For example, let us see a bipartite graph $\mathcal{B}$ in Figure 1. A *cold-start* user u_1 only interacts with item i_1. Since u_1 shares i_1 with

[1]In this paper, we use the terminology "graph" to refer to the graph/network structure of data and "network" for the architecture of machine learning models.

user u_2 and user u_3, as a result, three items (i_2, i_3 and i_4) connected with u_2 or u_3 can all be recommended to u_1 by a CF-based model. However, a natural and important question arises: which one in the three items is the most reliable recommendation for u_1? The key to answer the question lies in the user-item connectivity information. In fact, if we take a look at the connections of the graph, it is clear that there is only one path existing between u_1 and i_2 (or i_3), while two paths connect u_1 to i_4. Thus, compared with i_2 and i_3, obviously, i_4 is a more reliable recommendation for u_1.

However, existing CF-based methods, including model-based and memory-based approaches, often suffer from the difficulty of modeling the connectivity information. Previous model-based approaches, such as Matrix Factorization (MF) [19], are usually designed to approximate the direct connections (or proximities). However, indirect connectivity information hidden in the graph structures is rarely captured by traditional model-based approaches. For instance, it is formidable for them to model the number of paths between u_1 and i_4 in Figure 1. Whereas a number of memory-based approaches [14, 25] is introduced to model the connectivity information, these methods often rely on pre-defined similarity functions. However, in the real world, defining an appropriate similarity function suitable for diverse application cases is never an easy task.

Spectral graph theory [27] studies connections between combinatorial properties of a graph and the eigenvalues of matrices associated to the graph, such as the laplacian matrix (see Definition 2.4 in Section 2). In general, the spectrum of a graph focuses on the connectivity of the graph, instead of the geometrical proximity. To see how does the *spectral domain* come to help for recommendations and better understand the advantages of viewing a user-item bipartite graph in the spectral perspective, let us revisit the toy example shown in Figure 1. For the bipartite graph $\mathcal{B}$, we visually plot its vertices in one specific frequency domain. Although vertices do not come with coordinates, a popular way to draw them in a space is to use eigenvectors of a laplacian matrix associated with the graph to supply coordinates [28]. Figure 2 shows that, compared with i_2 and i_3, i_4 becomes closer to u_1 in the space[2]. Thus, when transformed into the frequency domain, i_4 is revealed to be a more suitable choice than i_2 or i_3 for u_1. The underlying reason is that the connectivity information of the graph has been uncovered in the frequency domain, where the relationships between vertices depend on not only their proximity but also connectivity. Thus, exploiting the spectrum of a graph can help better explore and identify the items to be recommended.

Inspired by the recent progress [6, 17] in node/graph classification methods, we propose a *spectral graph theory* based method to leverage the broad information existing in the *spectral domain* to overcome the aforementioned drawbacks and challenges. Specifically, to conquer the difficulties (see Section 3.3) of directly learning from the *spectral domain* for recommendations, we first present a new spectral convolution operation (see Eq. (10)), which is approximated by a polynomial to dynamically amplify or attenuate each frequency domain. Then, we introduce a deep recommendation model, named Spectral Collaborative Filtering (SpectralCF), built by

[2]In *spectral graph theory*, smaller (or larger) eigenvalues of the associated laplacian matrix corresponds to lower- (or higher-) frequency domains. In Figure 1, we plot each vertex j at the point $(\boldsymbol{\mu}_1(j), \boldsymbol{\mu}_2(j))$, where $\boldsymbol{\mu}_l(j)$ indicates the j_{th} value of the l_{th} eigenvector of the laplacian matrix $\boldsymbol{L}$.

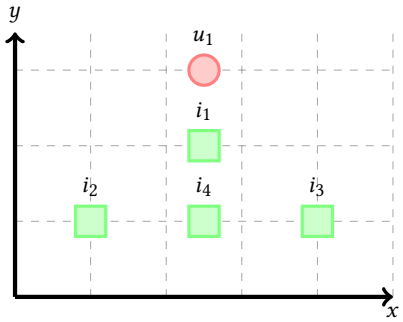


Figure 2: Vertices of the bipartite graph in Figure 1 are plotted in a frequency domain. Note that the vertices not shown above are omitted for simplicity.

multiple proposed spectral convolution layers. SpectralCF directly performs collaborative filtering in the *spectral domain*.

The key contributions of this work can be summarized as follows:

- **Novelty**: To the best of our knowledge, it is the first CF-based method directly learning from the *spectral domains* of user-item bipartite graphs.
- **A deep recommendation model**: We propose a new spectral convolution operation performing in the *spectral domain*. Stacked by multiple layers of the proposed spectral convolution operation, a deep recommendation model, named Spectral Collaborative Filtering (SpectralCF), is introduced.
- **Strong Performance**: In the experiments, SpectralCF outperforms state-of-the-art comparative models. It is shown that SpectralCF effectively utilizes the rich information of connectivity existing in the *spectral* domain to ease the *cold-start* problem.

The rest of the paper is organized as follows. In Section 2, we provide preliminary concepts. Section 3 describes SpectralCF in detail. Experiments are presented in Section 4 to analyze SpectralCF and demonstrate its effectiveness compared with state-of-the-art techniques for RS. In Section 5, we give a short review of the works related to our study. Finally, conclusions are presented in Section 6.

2 DEFINITIONS AND PRELIMINARIES

In this section, we present the background and preliminaries of this study. Throughout the paper, we denote scalars by either lowercase or uppercase letters, vectors by boldfaced lowercase letters, and matrices by boldfaced uppercase letters. Unless otherwise specified, all vectors are considered to be column vectors. Let $\boldsymbol{I}$ denote an identity matrix, and $\mathbf{1}$ and $\mathbf{0}$ denote matrices of ones and zeros, respectively. In addition, we define the following definitions in this paper as:

Definition 2.1. *(Bipartite Graph). A bipartite user-item graph with N vertices and E edges for recommendations is defined as $\mathcal{B} = \{\mathcal{U}, \mathcal{I}, \mathcal{E}\}$, where $\mathcal{U}$ and $\mathcal{I}$ are two disjoint vertex sets of users and items. Every edge $e \in \mathcal{E}$ has the form $e = (u, i)$ where $u \in \mathcal{U}$ and $i \in \mathcal{I}$ and denotes that user u has interacted with item i in the training set.*

Definition 2.2. *(Implicit Feedback Matrix). An implicit feedback matrix $\boldsymbol{R}$ is a $|\mathcal{U}| \times |\mathcal{I}|$ matrix defined as following:*

$$R_{r,j} = \begin{cases} 1 & \text{if } (u_r, i_j) \text{ interaction is observed,} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Definition 2.3. *(Adjacent Matrix). For the bipartite graph $\mathcal{B}$, its corresponding adjacent matrix $\boldsymbol{A}$ can be defined as:*

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{R} \\ \boldsymbol{R}^\top & \boldsymbol{0} \end{bmatrix}, \quad (2)$$

where $\boldsymbol{A}$ is an $N \times N$ matrix.

Definition 2.4. *(Laplacian Matrix). The random walk laplacian matrix $\mathbf{L}$ is defined as $\mathbf{L} = \mathbf{I} - \boldsymbol{D}^{-1}\boldsymbol{A}$, where $\mathbf{I}$ is the $N \times N$ identity matrix and $\boldsymbol{D}$ is the $N \times N$ diagonal degree matrix defined as $D_{nn} = \sum_j A_{n,j}$.*

This paper focuses on the recommendation problem with implicit feedbacks, where we only observe whether a person has viewed/liked/clicked an item and do not observe explicit ratings. Let $\mathcal{I}_i^+$ denote the set of all items liked by user i and $\mathcal{I}_i^-$ denote the remaining items. We define the recommendation problem which we study in this paper as the following:

Definition 2.5. *(Problem Definition). Given a user set $\mathcal{U}$ and an item set $\mathcal{I}$, for each user $u \in \mathcal{U}$ who has liked/clicked/viewed an item set $\mathcal{I}_u^+ \subseteq \mathcal{I}$, we aim to recommend a ranked list of items from $\mathcal{I}_u^-$ that are of interests to the user.*

3 PROPOSED MODEL

In this section, we first describe the process of performing a *graph fourier transform* on a bipartite graph $\mathcal{B}$ for recommendations. Then we propose to place a novel spectral convolution filter on vertices (users and items) of the bipartite graph to dynamically filter the contributions of each frequency component in the *spectral domain*. Later, a polynomial approximation is employed to overcome the shortcomings of the proposed convolution operation. Finally, with the approximate convolution operation, we introduce our final recommender system, named Spectral Collaborative Filtering, stacked by multiple spectral convolution layers.

3.1 Graph Fourier Transform

Definition 3.1. *(Graph Signal). Given any graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ and $\mathcal{E}$ are a vertex and an edge set, respectively, a graph signal is defined as a state vector $\boldsymbol{x} \in \mathcal{R}^{|\mathcal{V}|\times 1}$ over all vertices in the graph, where x_j is the j_{th} value of $\boldsymbol{x}$ observed at the j_{th} vertex of $\mathcal{G}$.*

The classical *fourier transform* is defined as an expansion of a function f in terms of the complex exponentials as:

$$\hat{f}(\xi) = \int_{-\infty}^{+\infty} f(x) e^{-2\pi i \xi} dx, \quad (3)$$

where i is an imaginary number, and the complex exponentials $(e^{-2\pi i \xi})$ form an orthonormal basis.

Analogously, the *graph fourier transform* is defined as an expansion of an observed *graph signal* in terms of the eigenvectors of the graph laplacian $\boldsymbol{L}$, and the eigenvectors serve as a basis in the *spectral domain*. Let us assume that a *graph signal* $(\boldsymbol{x} \in \mathcal{R}^{|\mathcal{V}|\times 1})$ is observed on a graph $\mathcal{G}$, we define the *graph fourier transform* and its inverse on $\mathcal{G}$ as:

$$\hat{x}(l) = \sum_{j=0}^{N-1} x(j)\mu_l(j) \quad \text{and} \quad x(j) = \sum_{l=0}^{N-1} \hat{x}(l)\mu_l(j), \quad (4)$$

where $x(j)$, $\hat{x}(l)$ and $\mu_l(j)$ denote the j_{th}, l_{th} and j_{th} value of $\boldsymbol{x}$, $\hat{\boldsymbol{x}}$ and $\boldsymbol{\mu}_l$, respectively; $\boldsymbol{\mu}_l$ denotes the l_{th} eigenvector of $\boldsymbol{L}$; $\hat{\boldsymbol{x}}$ represents a *graph signal* which has been transformed into the *spectral domain*. For simplicity, we rewrite Eq. (4) in the matrix form as $\hat{\boldsymbol{x}} = \boldsymbol{U}^\top \boldsymbol{x}$ and $\boldsymbol{x} = \boldsymbol{U}\hat{\boldsymbol{x}}$, respectively, where $\boldsymbol{U} = \{\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_l, ..., \boldsymbol{\mu}_{N-1}\}$ are eigenvectors of $\boldsymbol{L}$.

In particular, for a bipartite graph $\mathcal{B}$, assume that there are two types of *graph signals*: $\boldsymbol{x}^u \in \mathcal{R}^{|\mathcal{U}|\times 1}$ and $\boldsymbol{x}^i \in \mathcal{R}^{|\mathcal{I}|\times 1}$, associated with user and item vertices, respectively. We transform them into the *spectral domain* and vice versa as :

$$\begin{bmatrix} \hat{\boldsymbol{x}}^u \\ \hat{\boldsymbol{x}}^i \end{bmatrix} = \boldsymbol{U}^\top \begin{bmatrix} \boldsymbol{x}^u \\ \boldsymbol{x}^i \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \boldsymbol{x}^u \\ \boldsymbol{x}^i \end{bmatrix} = \boldsymbol{U} \begin{bmatrix} \hat{\boldsymbol{x}}^u \\ \hat{\boldsymbol{x}}^i \end{bmatrix}. \quad (5)$$

3.2 Spectral Convolution Filtering

The broad information of graph structures exists in the *spectral domain*, and different types of connectivity information between users and items can be uncovered in different frequency domains. It is desirable to dynamically adjust the importance of each frequency domain for RS.

To this end, we propose a convolution filter, parameterized by $\boldsymbol{\theta} \in \mathcal{R}^N$, as $g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}) = diag([\theta_0\lambda_0, \theta_1\lambda_1, ..., \theta_{N-1}\lambda_{N-1}])$ into the *spectral domain* as:

$$\begin{bmatrix} \boldsymbol{x}^u_{new} \\ \boldsymbol{x}^i_{new} \end{bmatrix} = \boldsymbol{U} g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}) \begin{bmatrix} \hat{\boldsymbol{x}}^u \\ \hat{\boldsymbol{x}}^i \end{bmatrix} = \boldsymbol{U} g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}) \boldsymbol{U}^\top \begin{bmatrix} \boldsymbol{x}^u \\ \boldsymbol{x}^i \end{bmatrix}, \quad (6)$$

where $\boldsymbol{x}^u_{new}$ and $\boldsymbol{x}^i_{new}$ are new *graph signals* on $\mathcal{B}$ learned by the filter $g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})$, and $\boldsymbol{\Lambda} = \{\lambda_0, \lambda_1, ..., \lambda_{N-1}\}$ denotes eigenvalues of the graph laplacian matrix $\mathbf{L}$.

In Eq. (6), a convolution filter $g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})$ is placed on a spectral *graph signal* $\begin{bmatrix} \hat{\boldsymbol{x}}^u \\ \hat{\boldsymbol{x}}^i \end{bmatrix}$, and each value of $\boldsymbol{\theta}$ is responsible for boosting or diminishing each corresponding frequency component. The eigenvector matrix $\boldsymbol{U}$ in Eq. (6) is used to perform an inverse *graph fourier transform*.

3.3 Polynomial Approximation

Recall that we proposed a convolution operation, as shown in Eq. (6), to directly perform in the *spectral domain*. Although the filter is able to dynamically measure contributions of each frequency component for the purpose of recommendations, there are two limitations. First, as shown in Eq. (6), the learning complexity of the filter is $O(N)$, where N is the number of vertices. That is, unlike classical Convolutional Neural Networks (CNNs), the number of parameters of the filter is linear to the dimensionality of data. It constrains the scalability of the proposed filter. Second, the learned *graph signals* $(\boldsymbol{x}^u_{new} \in \mathcal{R}^{|\mathcal{U}|\times 1}$ and $\boldsymbol{x}^i_{new} \in \mathcal{R}^{|\mathcal{I}|\times 1})$ are vectors. It means that each vertex of users or items is represented by a scalar feature. However, a vector for every user and item is necessary to model the deep and complex connections between users and items.

The first limitation can be overcome by using a polynomial approximation. We first demonstrate that the set of all convolution

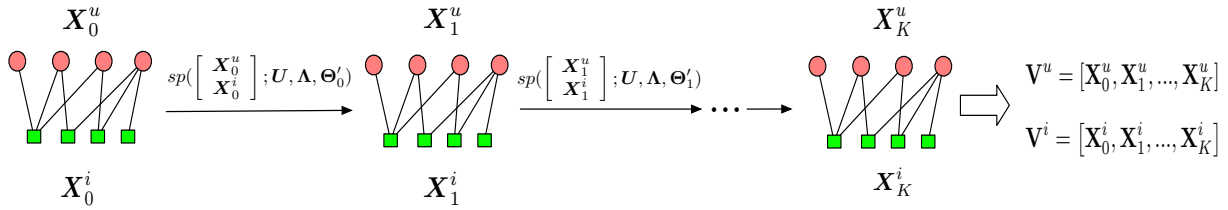


Figure 3: The feed-forward procedure of SpectralCF. The function $sp(:; U, \Lambda, \Theta)$ denotes the spectral convolution operation shown in Eq. (10).

filters $\mathcal{S}_g = \{g_\theta(\Lambda) = diag([\theta_0\lambda_0, \theta_1\lambda_1, ..., \theta_{N-1}\lambda_{N-1}]), \theta \in \mathcal{R}^N\}$ is equal to the set of finite-order polynomials $\mathcal{S}_h = \{h_{\theta'}(\Lambda) = \sum_{p=0}^{N-1} \theta'_p \Lambda^p, \theta' \in \mathcal{R}^N\}$.

Proposition 3.1. *$\mathcal{S}_h$ is equal to $\mathcal{S}_g$.*

Proof. Let us consider an instance $h_{\theta'}(\Lambda) \in \mathcal{S}_h$. Then, $h_{\theta'}(\Lambda) = \sum_{p=0}^{N-1} \theta'_p \Lambda^p = diag([\sum_{p=0}^{N-1} \theta'_p \lambda_0^{p-1} \cdot \lambda_0, \sum_{p=0}^{N-1} \theta'_p \lambda_1^{p-1} \cdot \lambda_1, ..., \sum_{p=0}^{N-1} \theta'_p \lambda_{N-1}^{p-1} \cdot \lambda_{N-1}])$. So, $h_{\theta'}(\Lambda) \in \mathcal{S}_g$. Now, consider a convolution filter $g_\theta(\Lambda) \in \mathcal{S}_g$. Then, there must exist a polynomial function $\phi(\lambda) = \sum_{p=0}^{N-1} a_p \lambda^p$ that interpolates through all pairs $(\lambda_i, \theta_i\lambda_i)$ for $i \in \{0, 1, ..., N-1\}$. The maximum degree of such a polynomial is at most $N-1$ as there are maximum N points to interpolate. Therefore, $g_\theta(\Lambda) = \sum_{p=0}^{N-1} a_p \Lambda^p = h_a(\Lambda) \in \mathcal{S}_h$. □

Now, we can approximate the convolution filters by using first P polynomials as the following:

$$g_\theta(\Lambda) \approx \sum_{p=0}^{P} \theta'_p \Lambda^p. \tag{7}$$

In this way, the learning complexity of the filter becomes $O(P)$, where P is a hyper-parameter, and independent from the number vertices. Specially, we limit the order of the polynomial, P, to 1 in order to avoid over-fitting. By substituting Eq. (7) into Eq. (6), we have:

$$\begin{bmatrix} x^u_{new} \\ x^i_{new} \end{bmatrix} = (\theta'_0 UU^\top + \theta'_1 U\Lambda U^\top) \begin{bmatrix} x^u \\ x^i \end{bmatrix}. \tag{8}$$

Furthermore, it is beneficial to further decrease the number of parameters by setting $\theta' = \theta'_0 = \theta'_1$. As a result, Eq. (8) becomes:

$$\begin{bmatrix} x^u_{new} \\ x^i_{new} \end{bmatrix} = \theta'(UU^\top + U\Lambda U^\top) \begin{bmatrix} x^u \\ x^i \end{bmatrix}, \tag{9}$$

where θ' is a scalar.

For the second limitation, one can generalize the *graph signals* ($x^u \in \mathcal{R}^{|\mathcal{U}|\times 1}$ and $x^i \in \mathcal{R}^{|\mathcal{I}|\times 1}$) to C-dimensional *graph signals*: $X^u \in \mathcal{R}^{|\mathcal{U}|\times C}$ and $X^i \in \mathcal{R}^{|\mathcal{I}|\times C}$. Hence, Eq. (9) becomes $\begin{bmatrix} X^u_{new} \\ X^i_{new} \end{bmatrix} = (UU^\top + U\Lambda U^\top) \begin{bmatrix} X^u \\ X^i \end{bmatrix} \theta'$. To take one step further, we generalize the filter parameter θ' to a matrix of filter parameters $\Theta' \in \mathcal{R}^{C\times F}$ with C input channels and F filters. As a result, our final spectral convolution operation is shown as the following:

$$\begin{bmatrix} X^u_{new} \\ X^i_{new} \end{bmatrix} = \sigma\left((UU^\top + U\Lambda U^\top) \begin{bmatrix} X^u \\ X^i \end{bmatrix} \Theta'\right), \tag{10}$$

where $X^u_{new} \in \mathcal{R}^{|\mathcal{U}|\times F}$ and $X^i_{new} \in \mathcal{R}^{|\mathcal{I}|\times F}$ denote convolution results learned with F filters from the *spectral domain* for users and items, respectively; σ denotes the logistic sigmoid function.

In fact, Eq. (10) is a general version of Eq. (9) as it is equivalent to perform Eq. (9) in C input channels with F filters. Hereafter, the proposed convolution operation as shown in Eq. (10) is denoted as a function $sp(:; U, \Lambda, \Theta')$, which is parameterized by U, Λ and Θ'.

3.4 Multi-layer Model

Given user vectors X^u and item vectors X^i, new *graph singals* (X^u_{new} and X^i_{new}) in Eq. (10) are convolution results learned from the *spectral domain* with a parameter matrix $\Theta' \in \mathcal{R}^{C\times F}$. As in classical CNNs, one can regard Eq. (10) as a propagation rule to build a deep neural feed-forward network based model, which we refer as Spectral Collaborative Filtering (SpectralCF).

Similar to word embedding techniques, we first randomly initialize user vectors X^u_0 and item vectors X^i_0. Taking X^u_0 and X^i_0 as inputs, a K layered deep spectralCF can be formulated as:

$$\begin{bmatrix} X^u_K \\ X^i_K \end{bmatrix} = \underbrace{sp\Big(...sp\Big(}_{K}\begin{bmatrix} X^u_0 \\ X^i_0 \end{bmatrix}; U, \Lambda, \Theta'_0\Big)...; U, \Lambda, \Theta'_{K-1}\Big), \tag{11}$$

where $\Theta'_{K-1} \in \mathcal{R}^{F\times F}$ is a matrix of filter parameters for the k_{th} layer; X^u_k and X^i_k denote the convolution filtering results of the k_{th} layer.

In order to utilize features from all layers of SpectralCF, we further concatenate them into our final latent factors of users and items as:

$$\mathbf{V}^u = [X^u_0, X^u_1, ..., X^u_K] \quad \text{and} \quad \mathbf{V}^i = [X^i_0, X^i_1, ..., X^i_K], \tag{12}$$

where $\mathbf{V}^u \in \mathcal{R}^{|\mathcal{U}|\times(C+KF)}$ and $\mathbf{V}^i \in \mathcal{R}^{|\mathcal{I}|\times(C+KF)}$.

In terms of the loss function, the conventional BPR loss suggested in [23] is employed. BPR is a pair-wise loss to address the implicit data for recommendations. Unlike point-wise based methods [18], BPR learns a triple (r, j, j'), where item j is liked/clicked/viewed by user r and item j' is not. By maximizing the preference difference between j and j', BPR assumes that the user i prefers item j over the unobserved item j'. In particular, given a user matrix $\boldsymbol{V}^u$ and an item matrix $\boldsymbol{V}^i$ as shown in Eq. (12), the loss function of SpectralCF is given as:

$$\mathcal{L} = \arg\min_{\boldsymbol{V}^u, \boldsymbol{V}^i} \sum_{(r,j,j')\in\mathcal{D}} -\ln \sigma(\boldsymbol{v}_r^{u\top} \boldsymbol{v}_j^i - \boldsymbol{v}_r^{u\top} \boldsymbol{v}_{j'}^i) + \lambda_{reg}(||\boldsymbol{V}^u||_2^2 + ||\boldsymbol{V}^i||_2^2), \tag{13}$$

where $\boldsymbol{v}_r^u$ and $\boldsymbol{v}_j^i$ denote r_{th} and j_{th} column of $\boldsymbol{V}^u$ and $\boldsymbol{V}^i$, respectively; λ_{reg} represents the weight on the regularization terms. The

Algorithm 1: SpectralCF

Input: Training set: $\mathcal{D} := \{(r, j, j') | r \in \mathcal{U} \wedge j \in \mathcal{I}_i^+ \wedge j' \subseteq \mathcal{I}_i^-\}$, number of epochs E, batch size B, number of layers K, dimension of latent factors C, number of filters F, regularization term λ_{reg}, learning rate λ, laplacian matrix $\boldsymbol{L}$ and its corresponding eigenvectors $\boldsymbol{U}$ and eigenvalues $\boldsymbol{\Lambda}$.

Output: Model's parameter set: $\Psi = \{\Theta'_0, \Theta'_1, \ldots, \Theta'_{K-1}, \boldsymbol{X}^u_0, \boldsymbol{X}^i_0\}$.

Randomly initialize $\mathbf{X}^u_0$ and $\mathbf{X}^i_0$ from a Gaussian distribution $\mathcal{N}(0.01, 0.02)$;
for $e = 1, 2, \cdots, E$ **do**
  Generate the e_{th} batch of size B by uniformly sampling from $\mathcal{U}$, $\mathcal{I}_i^+$ and $\mathcal{I}_i^-$;
  for $k = 0, 1, \cdots, K-1$ **do**
    Calculate $\boldsymbol{X}^u_{k+1}$ and $\boldsymbol{X}^i_{k+1}$ by using Eq. (10);
  end
  Concatenate $[\boldsymbol{X}^u_0, \boldsymbol{X}^u_1, \ldots, \boldsymbol{X}^u_K]$ into $\boldsymbol{V}^u$ and $[\boldsymbol{X}^i_0, \boldsymbol{X}^i_1, \ldots, \boldsymbol{X}^i_K]$ into $\boldsymbol{V}^i$;
  Estimate gradients $\frac{\partial \mathcal{L}}{\partial \Psi_e}$ by back propagation;
  Update Ψ_{e+1} according to the procedure of RMSprop optimization [29];
end
return Ψ_E.

training data $\mathcal{D}$ is generated as:

$$\mathcal{D} = \{(r, j, j') | r \in \mathcal{U} \wedge j \in \mathcal{I}_i^+ \wedge j' \in \mathcal{I}_i^-\}. \quad (14)$$

3.5 Optimization and Prediction

At last, RMSprop [29] is used to minimize the loss function. The RMSprop is an adaptive version of gradient descent which adaptively controls the step size with respect to the absolute value of the gradient. It is done by scaling the updated value of each weight by a running average of its gradient norm.

As shown in Algorithm 1, for a batch of randomly sampled triple (r, j, j'), we update parameters in each epoch using the gradients of the loss function. After the training process, with optimized $\boldsymbol{\Theta}$, $\boldsymbol{X}^u_0$ and $\boldsymbol{X}^i_0$, we derive the user r's preference over item j as $\boldsymbol{v}_r^{u\top} \boldsymbol{v}_j^i$. The final item recommendation for a user r is given according to the ranking criterion as Eq. (15).

$$r : j_1 \succcurlyeq j_2 \succcurlyeq \ldots \succcurlyeq j_n \Rightarrow \boldsymbol{v}_r^{u\top} \boldsymbol{v}_{j_1}^i > \boldsymbol{v}_r^{u\top} \boldsymbol{v}_{j_2}^i > \ldots > \boldsymbol{v}_r^{u\top} \boldsymbol{v}_{j_n}^i. \quad (15)$$

4 EXPERIMENTS

As discussed in the introduction section, leveraging the connectivity information in a user-item bipartite graph is essentially important for an effective recommendation model. In this section, we argue that, directly learning from the *spectral domain*, the proposed SpectraCF can reveal the rich information of graph structures existing in the *spectral domain* for making better recommendations. One may ask the following research questions:

RQ1: How much does SpectralCF benefit from the connectivity information learned from the *spectral domain*?
RQ2: Does SpectralCF learn from the *spectral domain* in an effective way?
RQ3: Compared with traditional methods, can SpectralCF better counter the *cold-start* problem?

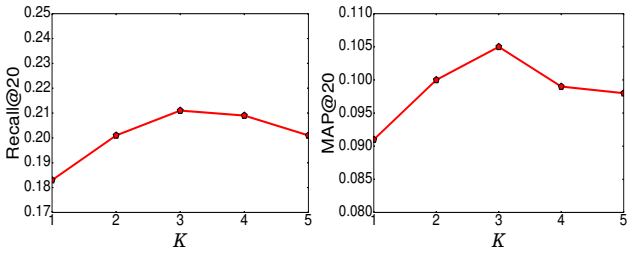


Figure 4: Effects of hyper-parameter K in terms of Recall@20 and MAP@20 in the dataset of *MovieLens-1M*.

In this section, in order to answer the questions above, we conduct experiments to compare SpectralCF with state-of-the-art models.

4.1 Comparative Methods

To validate the effectiveness of SpectralCF, we compare it with six state-of-the-art models. The comparative models can be categorized into two groups: (1) **CF-based Models:** To answer **RQ1**, we compare SpectralCF with four state-of-the-art CF-based methods (ItemKNN, BPR, eALS and NCF) which ignore the information in the *spectral domain*; (2) **Graph-based Models**: For **RQ2**, we are interested in how effectively does SpetralCF learn the connectivity information from the *spectral domain*. We therefore compare SpectralCF with two graph-based models: GNMF and GCMC. Although the two models are also CF-based, we term them as graph-based models since they learn the structural information from a bipartite graph. These two groups of comparative models are summarized below:

- **ItemKNN** [25]: ItemKNN is a standard neighbor-based collaborative filtering method. The model finds similar items for a user based on their similarities.
- **BPR** [23]: We use **B**ayesian **P**ersonalized **R**anking based Matrix Factorization. BPR introduces a pair-wise loss into the Matrix Factorization to be optimized for ranking [8].
- **eALS** [12]: This is a state-of-the-art matrix factorization based method for item recommendation. This model takes all unobserved interactions as negative instances and weighting them non-uniformly by the item popularity.
- **NCF** [11]: **N**eural **C**ollaborative **F**iltering fuses matrix factorization and Multi-Layer Perceptron (MLP) to learn from user-item interactions. The MLP endows NCF with the ability of modelling non-linearities between users and items.
- **GNMF** [3]: **G**raph regularized **N**on-negative **M**atrix **F**actorization considers the graph structures by seeking a matrix factorization with a graph-based regularization.
- **GCMC** [2]: **G**raph **C**onvolutional **M**atrix **C**ompletion utilizes a graph auto-encoder to learn the connectivity information of a bipartite interaction graph for latent factors of users and items.

Please note that, GNMF and GCMC are originally designed for explicit datasets. For a fair comparison, we follow the setting of [13] to adapt them for implicit data.

Table 1: The hyper-parameter setting of SpectralCF.

Hyper-parameters	K	C	F	λ_{reg}	B	E	λ
Values	3	16	16	0.001	1,024	200	0.001

4.2 Datasets

We test our method as well as comparative models on three publicly available datasets[3]:

- ***MovieLens-1M*** [10]: This movie rating dataset has been widely used to evaluate collaborative filtering algorithms. We used the version containing 1,000,209 ratings from 6,040 users for 3,900 movies. While it is a dataset with explicit feedbacks, we follow the convention [11] that transforms it into implicit data, where each entry is marked as 0 or 1 indicating whether the user has rated the item. After transforming, we retain a dataset of **1.0**% density.
- ***HetRec*** [4]: This dataset has been released by the Second International Workshop on Information Heterogeneity and Fusion in Recommender Systems[4]. It is an extension of *MovieLens-10M* dataset and contains 855,598 ratings, 2,113 users and 10,197 movies. After converting it into implicit data as *MovieLens-1M*, we obtain a dataset of **0.3**% density.
- ***Amazon Instant Video*** [20]: The dataset consists of 426,922 users, 23,965 videos and 583,933 ratings from *Amazon.com*. Similarly, we transformed it into implicit data and removed users with less than 5 interactions. As a result, a dataset of **0.12%** density is obtained.

4.3 Experimental Setting

Ideally, a recommendation model should not only be able to retrieve all relevant items out of all items but also provide a rank for each user where relevant items are expected to be ranked in the top. Therefore, in our experiments, we use Recall@M and MAP@M to evaluate the performance of the top-M recommendations. Recall@M is employed to measure the fraction of relevant items retrieved out of all relevant items. MAP@M is used for evaluating the ranking performance of RS. The Recall@M for each user is then defined as:

$$\text{Recall@M} = \frac{\#\text{items the user likes among the top M}}{\text{total number of items the user likes}}. \quad (16)$$

The final results reported are average recall over all users.

For each dataset, we randomly select 80% items associated with each user to constitute the training set and use all the remaining as the test set. For each evaluation scenario, we repeat the evaluation five times with different randomly selected training sets and the average performance is reported in the following sections.

We use a validation set from the training set of each dataset to find the optimal hyper-parameters of comparative methods introduced in the Section 4.1. For ItemKNN, we employ the cosine distance to measure item similarities. The dimensions of latent factors for BPR, eALS and GNMF are searched from {8,16,32,64,128} via the validation set. The hyperparameter λ of eALS is selected from 0.001 to 0.04. Since the architecture of a multi-layer perceptron (MLP) is difficult to optimize, we follow the suggestion from the original paper [11] to employ a three-layer MLP with the shape of (32, 16, 8) for NCF. The dropout rate of nodes for GCMC is searched from {0.3,0.4,0.5,0.6,0.7,0.8}. Our SpectralCF has one essential hyper-parameter: K. Figure 4 shows how the performances of SpectralCF vary as K is set from 1 to 5 on the validation set of *MovieLens-1M*. As we can see, in terms of Recall@20 and MAP@20, SpectralCF reaches its best performances when K is fixed as 3. Other hyper-parameters of SpectralCF are empirically set and summarized in Table 1, where λ denotes the learning rate of RMSprop. Our models are implemented in *TensorFlow* [1].

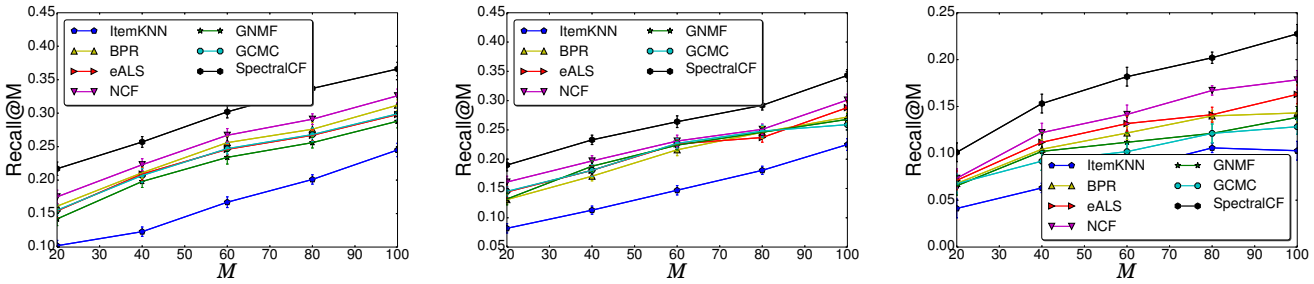
4.4 Experimental Results (RQ1 and RQ2)

In Figure 5, we compare SpectralCF with four CF-based models and two graph-based models in terms of Recall@M on all three datasets. Overall, when M is varied from 20 to 100, SpectralCF consistently yields the best performance across all cases. Among CF-based comparative models, ItemKNN gives the worst performances in all three datasets, indicating the necessity of modeling users' personalized preferences rather than just recommending similar items to users. For graph-based models (GNMF and GCMC), they generally underperform CF-based models such as BPR and NCF. The unsatisfying performance of GNMF shows that adding a graph-based regularization is not sufficient to capture complex structures of graphs. Though GCMC directly performs on a user-item bipartite graph, each vertex in the graph is only allowed to learn from its neighbors. This constrains its ability of capturing global structures in the graph. Among all comparative models, benefiting from its capability of modeling non-linear relationships between users and items, NCF beats all other models and becomes the strongest one. However, none of models above are able to directly perform in the *spectral domain*. They lose the rich information in the domain and as a result, SpectralCF greatly outperforms NCF by **16.1%**, **16.2%** and **28.0%** in the dataset of *MovieLen-1M*, *HetRec* and *Amazon Instant Video*, respectively.

In Figure 6, we compare SpectralCF with all comparative models in terms of MAP@M. Again, when M is in a range from 20 to 100, SpectralCF always yields the best performance. Neighbor-based ItemKNN performs the worst among all models. It further shows the advantages of modeling users' personalized preferences. Compared with NCF and BPR, graph-based models (GNMF and GCMC) again fail to show convincing ranking performances measured by MAP@M. For CF-based models, while NCF beats other CF-based models in the dataset of *HetRec*, BPR shows itself as a strong model for ranking, owing to its pairwise ranking loss. It slightly outperforms NCF on average in the datasets of *MovieLens-1M* and *Amazon Instant Video*. However, SpectralCF improves BPR by **15.9%**, **64.9%** and **47.5%** in the dataset of *MovieLen-1M*, *HetRec* and *Amazon Instant Video*, respectively.

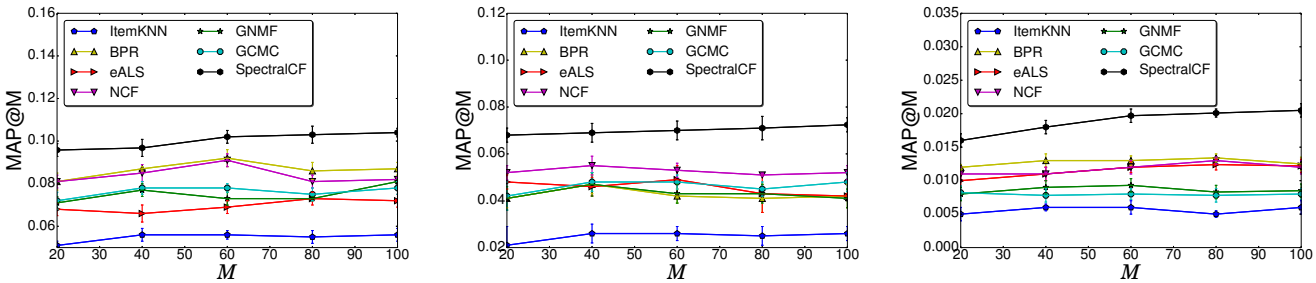
Overall, as shown in Figure 5 and 6, not surprisingly, the performances of all models decline as the dataset becomes sparse. However, SpectralCF always outperforms all comparative models regardless of the sparsities of the datasets. By comparing spectralCF with traditional CF-based models, we demonstrate that the rich information of connectivity existing in the *spectral domain* assists SpectralCF in learning better latent factors of users and items. By

[3] *MovieLens-1M* and *HetRec* are available at https://grouplens.org/datasets/; and *Amazon Instant Video* can be found at http://jmcauley.ucsd.edu/data/amazon/
[4] http://ir.ii.uam.es/hetrec2011/



(a) MovieLens-1M (b) HetRec (c) Amazon Instant Video

Figure 5: Performance comparison in terms of recall@M with M varied from 20 to 100. Errors bars are 1-standard deviation.



(a) MovieLens-1M (b) HetRec (c) Amazon Instant Video

Figure 6: Performance comparison in terms of MAP@M with M varied from 20 to 100. Errors bars are 1-standard deviation.

comparing SpectralCF with graph-based models, we show that SpectralCF can effectively learn from the *spectral domain*.

4.5 Quality of Recommendations for Cold-start Users (RQ3)

To answer **RQ3**, in this section, we conduct an experiment to investigate the quality of recommendations made by SpectralCF for *cold-start* users. To this end, in the dataset of *MovieLens-1M*, we build training sets with different degrees of sparsity by varying the number of items associated with each user, denoted as P, from one to five. All the remaining items associated with users are used as the test set. We compare SpectralCF with BPR, which is widely known and also shown as a strong ranking performer in Figure 6. The test results are reported in the Table 2.

In Table 2, it is shown that, suffering from the *cold-start* problem, the performances of BPR and SpectralCF inevitably degrade. However, regardless of the number of items associated with users, SpectralCF consistently outperforms BPR in terms of Recall@20 and MAP@20. On average, SpectralCF improves BPR by **36.8%** and **33.8%** in Recall@20 and MAP@20, respectively. Hence, it is demonstrated that compared with BPR, spectralCF can better handle *cold-start* users and provide more reliable recommendations.

Table 2: Performance Comparison in terms of Recall@20 and MAP@20 in the sparse training sets. In the dataset of *MovieLens-1M*, we vary the number of items associated with each users, denoted as P, from 1 to 5. The average results are reported and the best results are in bold. The standard deviation is shown in parentheses.

	P	1	2	3	4	5
Recall @20	BPR	0.021 (0.003)	0.029 (0.004)	0.031 (0.003)	0.034 (0.004)	0.038 (0.003)
	SpectralCF	**0.031** (0.003)	**0.039** (0.003)	**0.042** (0.002)	**0.045** (0.003)	**0.051** (0.003)
	Improve-ment	47.6%	34.5%	35.5%	32.4%	34.2%
MAP @20	BPR	0.014 (0.002)	0.017 (0.002)	0.021 (0.002)	0.024 (0.003)	0.027 (0.003)
	SpectralCF	**0.019** (0.002)	**0.024** (0.002)	**0.028** (0.003)	**0.031** (0.003)	**0.035** (0.002)
	Improve-ment	35.7%	41.2%	33.3%	29.2%	29.6%

5 RELATED WORKS

There are two categories of studies related to our work: deep learning based RS and graph-based RS. In this section, we will first briefly review existing works in the area of deep RS. Then, we focus on presenting recent works on graph-based RS. Despite all these approaches, SpectralCF is the first model to directly learn latent factors of users and items from the *spectral domains* of user-item bipartite graphs.

5.1 Deep Recommender Systems

One of the early works utilizing deep learning for RS builds a Restricted Boltzmann Machines (RBM) based method to model users

using their rating preferences [24]. Although the method is still a relatively shallow model, it slightly outperforms Matrix Factorization technique and shows the promising future for deep recommender systems. In [32], a generative model and a discriminative model are employed to play a minimax game. The two models are iteratively optimized and achieve promising results for the item recommendation problem. Inspired by [24], [43] proposed a CF Neural Autoregressive Distribution Estimator (CF-NADE) model for collaborative filtering tasks. CF-NADE shares parameters between different ratings. [11] presents to utilize a Multilayer Perceptron (MLP) to model user-item interactions.

A number of researchers proposed to build a hybrid recommender systems to counter the sparsity problem. [34] introduce Convolutional Neural Networks (CNN) and Deep Belief Network (DBN) to assist representation learning for music data. As such, their model is able to extract latent factors of songs without ratings while CF based techniques like MF are unable to handle these songs. These approaches above pre-train embeddings of users and items with matrix factorization and utilize deep models to fine-tune the learned item features based on item content. In [7] and [30], multi-view deep models are built to utilize item information from more than one domain. [16] integrates a CNN with PMF to analyze documents associated with items to predict users' future explicit ratings. [42] leverage two parallel neural networks to jointly model latent factors of users and items. To incorporate visual signals into RS, [33] propose CNN-based models to incorporate visual signals into RS. They make use of visual features extracted from product images using deep networks to enhance the performance of RS. [38] investigates how to leverage the multi-view information to improve the quality of recommender systems. [5] jointly trains wide linear models and deep neural networks for video recommendations. [31] and [40] utilize RNN to consider word orders and extract complex semantics for recommendations. [35] applies an attention mechanism on a sequence of models to adaptively capture the change of criteria of editors. [41] leverages an attentional model to learn adaptive user embeddings. A survey on the deep learning based RS with more works on this topic can be found in [39].

5.2 Graph-based Recommender Systems

In order to learn latent factors of users and items from graphs, a number of researchers have proposed graph-based RS. [44] develops a semi-supervised learning model on graphs for document recommendation. The model combines multiple graphs in order to measure item similarities. In [37], they propose to model the check-in behaviors of users and a graph-based preference propagation algorithm for point of interest recommendation. The proposed solution exploits both the geographical and temporal influences in an integrated manner. [9] addresses the problem of personalized tag recommendation by modeling it as a "query and ranking" problem. Inspired by the recent success of graph/node embedding methods, [2] proposes a graph convolution network based model for recommendations. In [2], a graph auto-encoder learns the structural information of a graph for latent factors of users and items. [3] adds graph-based regularizations into the matrix factorization model to learn graph structures. Graph-regularized methods are developed for the problemm of matrix completion in [22]. [21] combines a convolutional neural network and a recurrent neural network to model the dynamic rating generation process. Although this work also considers the *spectral domain*, they learn from a graph constructed from side information, such as genres or actors for movies. In contrast, our method learns directly from user-item bipartite graphs and does not require the side information. Thus, this work is not comparable to our method.

Additionally, some scholars have proposed to incorporate the heterogeneous information on a graph for recommendations. [15] suggests a general latent factor model for entities in a graph. [36] introduces a recommendation model for implicit data by taking advantage of different item similarity semantics in the graph. [26] introduces a semantic path based personalized recommendation method to predict the rating scores of users on items.

However, all works above are different from ours because they fail to consider the rich information in the *spectral domains* of user-item bipartite graphs. Also, our study focuses on learning from the implicit feedbacks, and leaves incorporating the heterogeneous information in a graph and the item content for future works.

6 CONCLUSIONS

It is shown that the rich information of connectivity existing in the *spectral domain* of a bipartite graph is helpful for discovering deep connections between users and items. In this paper, we introduce a new spectral convolution operation to directly learn latent factors of users and items from the *spectral domain*. Furthermore, with the proposed operation, we build a deep feed-forward neural network based recommendation model, named Spectral Collaborative Filtering (SpectralCF). Due to the rich information of connectivity existing in the *spectral domain*, compared with previous works, SpectralCF is capable of discovering deep connections between users and items and therefore, alleviates the *cold-start* problem for CF. To the best of our knowledge, SpectralCF is the first CF-based method directly learning from the *spectral domains* of user-item bipartite graphs. We believe that it shows the potential of conducting CF in the *spectral domain*, and will encourage future works in this direction.

In comparison with four state-of-the-art CF-based and two graph-based models, SpectralCF achieved **20.1%** and **42.6%** improvements averaging on three standard datasets in terms of Recall@M and MAP@M, respectively.

Additionally, in the experiments, by varying the number of items associated with each user from 1 to 5, we build training sets with different degrees of sparsity to investigate the quality of recommendations made by SpectralCF for *cold-start* users. By comparing SpectralCF with BPR, on average, SpectralCF improves BPR by **36.8%** and **33.8%** in Recall@20 and MAP@20, respectively. It is validated that SpectralCF can effectively ameliorate the *cold-start* problem.

ACKNOWLEDGMENTS

This work is supported in part by NSF through grants IIS-1526499, IIS-1763325, and CNS-1626432, and NSFC 61672313. This work is also partially supported by NSF through grant IIS-1763365 and by FSU through the startup package and FYAP award.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA.*

[2] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. *arXiv preprint arXiv:1706.02263* (2017).

[3] Deng Cai, Xiaofei He, Xiaoyun Wu, and Jiawei Han. 2008. Non-negative matrix factorization on manifold. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on.* IEEE, 63–72.

[4] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2011. 2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011). In *Proceedings of the 5th ACM conference on Recommender systems (RecSys 2011).* ACM, New York, NY, USA.

[5] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems.* ACM, 7–10.

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems.* 3844–3852.

[7] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 278–288.

[8] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. MyMediaLite: A Free Recommender System Library. In *5th ACM International Conference on Recommender Systems (RecSys 2011).*

[9] Ziyu Guan, Jiajun Bu, Qiaozhu Mei, Chun Chen, and Can Wang. 2009. Personalized tag recommendation using graph-based ranking on multi-type interrelated objects. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval.* ACM, 540–547.

[10] F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5, 4 (2016), 19.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017.* 173–182. https://doi.org/10.1145/3038912.3052569

[12] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval.* ACM, 549–558.

[13] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining.* Ieee, 263–272.

[14] Mohsen Jamali and Martin Ester. 2009. Trustwalker: a random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 397–406.

[15] Mohsen Jamali and Laks Lakshmanan. 2013. HeteroMF: recommendation in heterogeneous information networks using context dependent factor models. In *Proceedings of the 22nd international conference on World Wide Web.* ACM, 643–654.

[16] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems.* ACM, 233–240.

[17] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[18] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 426–434.

[19] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[20] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based Recommendations on Styles and Substitutes. *arXiv preprint arXiv:1506.04757* (2015).

[21] Federico Monti, Michael Bronstein, and Xavier Bresson. 2017. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems.* 3700–3710.

[22] Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon. 2015. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in neural information processing systems.* 2107–2115.

[23] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence.* AUAI Press, 452–461.

[24] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning.* ACM, 791–798.

[25] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web.* ACM, 285–295.

[26] Chuan Shi, Zhiqiang Zhang, Ping Luo, Philip S Yu, Yading Yue, and Bin Wu. 2015. Semantic path based personalized recommendation on weighted heterogeneous information networks. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management.* ACM, 453–462.

[27] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine* 30, 3 (2013), 83–98.

[28] Daniel A Spielman. 2007. Spectral graph theory and its applications. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on.* IEEE, 29–38.

[29] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning* 4 (2012), 2.

[30] Fengjiao Wang, Yongzhi Qu, Lei Zheng, Chun-Ta Lu, and S Yu Philip. 2017. Deep and Broad Learning on Content-Aware POI Recommendation. In *Collaboration and Internet Computing (CIC), 2017 IEEE 3rd International Conference on.* IEEE, 369–378.

[31] Hao Wang, SHI Xingjian, and Dit-Yan Yeung. 2016. Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks. In *Advances in Neural Information Processing Systems.* 415–423.

[32] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval.* ACM, 515–524.

[33] Suhang Wang, Yilin Wang, Jiliang Tang, Kai Shu, Suhas Ranganath, and Huan Liu. 2017. What your images reveal: Exploiting visual contents for point-of-interest recommendation. In *Proceedings of the 26th International Conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 391–400.

[34] Xinxi Wang and Ye Wang. 2014. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the ACM International Conference on Multimedia.* ACM, 627–636.

[35] Xuejian Wang, Lantao Yu, Kan Ren, Guanyu Tao, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Dynamic attention deep model for article recommendation by learning human editors' demonstration. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 2051–2059.

[36] Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Brandon Norick, and Jiawei Han. 2013. Recommendation in heterogeneous information networks with implicit user feedback. In *Proceedings of the 7th ACM conference on Recommender systems.* ACM, 347–350.

[37] Quan Yuan, Gao Cong, and Aixin Sun. 2014. Graph-based point-of-interest recommendation with geographical and temporal influences. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management.* ACM, 659–668.

[38] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining.* ACM, 353–362.

[39] Shuai Zhang, Lina Yao, and Aixin Sun. 2017. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435* (2017).

[40] Lei Zheng, Bokai Cao, Vahid Noroozi, S Yu Philip, and Nianzu Ma. 2017. Hierarchical collaborative embedding for context-aware recommendations. In *Big Data (Big Data), 2017 IEEE International Conference on.* IEEE, 867–876.

[41] Lei Zheng, Chun-Ta Lu, Lifang He, Sihong Xie, Vahid Noroozi, He Huang, and Philip S Yu. 2018. MARS: Memory Attention-Aware Recommender System. *arXiv preprint arXiv:1805.07037* (2018).

[42] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining.* ACM, 425–434.

[43] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. 2016. A neural autoregressive approach to collaborative filtering. *arXiv preprint arXiv:1605.09477* (2016).

[44] Ding Zhou, Shenghuo Zhu, Kai Yu, Xiaodan Song, Belle L Tseng, Hongyuan Zha, and C Lee Giles. 2008. Learning multiple graphs for document recommendations. In *Proceedings of the 17th international conference on World Wide Web.* ACM, 141–150.