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
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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

RecSys ’18, October 2–7, 2018, Vancouver, BC, Canada
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5901-6/18/10...
$15.00
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 $u_2$, $u_4$ 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 \( B \), 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 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 \( I \) denote an identity matrix, and \( I \) and \( 0 \) 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 \( B = \{U, I, E\} \), where \( U \) and \( I \) are two disjoint vertex sets of users and items. Every edge \( e \in E \) has the form \( e = (u, i) \) where \( u \in U \) and \( i \in I \) and denotes that user \( u \) has interacted with item \( i \) in the training set.
we study in this paper as the following:

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

(1)

**Definition 2.3.** (Adjacency Matrix). For the bipartite graph \( B \), its corresponding adjacent matrix \( A \) can be defined as:

\[
A = \begin{bmatrix} 0 & R \\ R^\top & 0 \end{bmatrix},
\]

(2)

where \( A \) is an \( N \times N \) matrix.

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

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 \( I^+ \) denote the set of all items liked by user \( i \) and \( I^- \) 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 \( U \) and an item set \( I \), for each user \( u \in U \) who has liked/clicked/viewed an item \( i \in I \), we aim to recommend a ranked list of items from \( I^+ \) 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 \( B \) 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 \( G = (V,E) \), where \( V \) and \( E \) are a vertex and an edge set, respectively, a graph signal is defined as a state vector \( x \in \mathbb{R}^{|V|} \) over all vertices in the graph, where \( x_j \) is the \( j \)-th value of \( x \) observed at the \( j \)-th vertex of \( G \).

The classical fourier transform is defined as an expansion of a complex exponential as:

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

where \( \xi \) is an imaginary number, and the complex exponentials \( e^{-2\pi i \xi x} \) 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 \( L \), and the eigenvectors serve as a basis in the spectral domain. Let us assume that a graph signal \( x \in \mathbb{R}^{\mid V \mid} \) is observed on a graph \( G \), we define the graph fourier transform and its inverse on \( G \) as:

\[
\hat{x}(l) = \sum_{j=0}^{N-1} x(j) \mu_l(j) \quad \text{and} \quad x(j) = \sum_{l=0}^{N-1} \hat{x}(l) \mu_l(j).
\]

(4)

where \( x(j) , \hat{x}(l) \) and \( \mu_l(j) \) denote the \( j \)-th, \( l \)-th and \( j \)-th value of \( x, \hat{x} \) and \( \mu_l \), respectively. \( \mu_l \) denotes the \( l \)-th eigenvector of \( L \); \( \hat{x} \) represents a graph signal which has been transformed into the spectral domain. For simplicity, we rewrite Eq. (4) in the matrix form as \( \hat{x} = U^\top x \) and \( x = U \hat{x} \), respectively, where \( U = \{ \mu_0, \mu_1, ..., \mu_{N-1} \} \) are eigenvectors of \( L \).

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

\[
\begin{bmatrix}
\hat{x}^u \\
\hat{x}^i
\end{bmatrix} = U^\top
\begin{bmatrix}
x^u \\
x^i
\end{bmatrix}
\quad \text{and} \quad
\begin{bmatrix}
x^u \\
x^i
\end{bmatrix} = U
\begin{bmatrix}
\hat{x}^u \\
\hat{x}^i
\end{bmatrix}.
\]

(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 \( \theta \in \mathbb{R}^N \), as \( g_\theta(x) = diag(\{\theta_0, \theta_1, ..., \theta_{N-1}\}) \) into the spectral domain as:

\[
\begin{bmatrix}
x^u_{\text{new}} \\
x^i_{\text{new}}
\end{bmatrix} = U g_\theta(A) U^\top
\begin{bmatrix}
x^u \\
x^i
\end{bmatrix}.
\]

(6)

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

In Eq. (6), a convolution filter \( g_\theta(A) \) is placed on a spectral graph signal \( \begin{bmatrix} \hat{x}^u \\ \hat{x}^i \end{bmatrix} \), and each value of \( \theta \) is responsible for boosting or diminishing each corresponding frequency component. The eigenvector matrix \( U \) 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 \( x_{\text{new}}^u \in \mathbb{R}^{\mid U \mid} \) and \( x_{\text{new}}^i \in \mathbb{R}^{\mid I \mid} \) 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
where we generalize the filter parameter $\theta$ as follows:

$$h_{\theta}(\lambda) = \sum_{p=0}^{N-1} \theta_p^0 \lambda^p, \theta' \in \mathbb{R}^N.$$ 

**Proposition 3.1.** $S_h$ is equal to $S_y$.

**Proof.** Let us consider an instance $h_{\theta}(\lambda) \in S_h$. Then, $h_{\theta}(\Lambda) = \sum_{p=0}^{N-1} \theta_p^0 \Lambda^p = \text{diag}(\sum_{p=0}^{N-1} \theta_p^0 \lambda^p, \lambda_0, \sum_{p=0}^{N-1} \theta_p^0 \lambda^p, \lambda_1, \ldots, \sum_{p=0}^{N-1} \theta_p^0 \lambda^p, \lambda_{N-1})$. So, $h_{\theta}(\Lambda) \in S_y$. Now, consider a convolution filter $g_{\theta}(\Lambda) \in S_y$. Then, there must exist a polynomial function $\phi(\lambda) = \sum_{p=0}^{P-1} a_p \lambda^p$ that interpolates through all pairs $(\lambda_i, \theta_i)$ for $i \in \{0, 1, \ldots, N-1\}$. The maximum degree of such a polynomial is at most $N-1$ as there are maximum $N$ points to interpolate. Therefore, $g_{\theta}(\Lambda) = \sum_{p=0}^{P-1} a_p \Lambda^p = h_{\theta}(\Lambda) \in S_h$. \hfill \Box

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

$$g_{\theta}(\Lambda) \approx \sum_{p=0}^{P-1} \theta_p^0 \Lambda^p. \quad (7)$$

In this way, the learning complexity of the filter becomes $O(P)$, where $P$ is a hyper-parameter, and independent from the number of vertices. 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:

$$\begin{bmatrix} x_u^{\text{new}} \\ x_i^{\text{new}} \end{bmatrix} = \begin{bmatrix} \theta_0^0 U^T + \theta_1^0 U^T \end{bmatrix} \begin{bmatrix} x_u^0 \\ x_i^0 \end{bmatrix}. \quad (8)$$

Furthermore, it is beneficial to further decrease the number of parameters by setting $\theta' = \theta_0^0 = \theta_1^0$. As a result, Eq. (8) becomes:

$$\begin{bmatrix} x_u^{\text{new}} \\ x_i^{\text{new}} \end{bmatrix} = \theta'(U^T + U^T) \begin{bmatrix} x_u^0 \\ x_i^0 \end{bmatrix}. \quad (9)$$

where $\theta'$ is a scalar.

For the second limitation, one can generalize the graph signals $(x_u^0 \in \mathbb{R}^{1 \times |U|}$ and $x_i^0 \in \mathbb{R}^{1 \times |X_i|}$) to $C$-dimensional graph signals: $X_u^0 \in \mathbb{R}^{1 \times |U| C}$ and $X_i^0 \in \mathbb{R}^{1 \times |X_i| C}$. Hence, Eq. (9) becomes:

$$\begin{bmatrix} X_u^{\text{new}} \\ X_i^{\text{new}} \end{bmatrix} = (U^T + U^T) \begin{bmatrix} X_u^0 \\ X_i^0 \end{bmatrix} \theta'. \quad (10)$$

To take one step further, we generalize the filter parameter $\theta'$ to a matrix of filter parameters $\Theta'$ that is $\mathbb{R}^{C \times F}$ with $C$ input channels and $F$ filters. As a result, our final spectral convolution operation is shown as the following:

$$\begin{bmatrix} X_u^{\text{new}} \\ X_i^{\text{new}} \end{bmatrix} = \sigma \begin{bmatrix} U^T + U^T \end{bmatrix} \begin{bmatrix} X_u^0 \\ X_i^0 \end{bmatrix} \Theta', \quad (11)$$

where $X_u^{\text{new}} \in \mathbb{R}^{1 \times |U| F}$ and $X_i^{\text{new}} \in \mathbb{R}^{1 \times |X_i| F}$ denote convolution results learned with $F$ filters from the spectral domain for users and items, respectively; $\sigma$ 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 $sp(\cdot; U, \Lambda, \Theta')$, which is parameterized by $U$, $\Lambda$, and $\Theta'$.

### 3.4 Multi-layer Model

Given user vectors $X_u^0$ and item vectors $X_i^0$, new graph signals $(X_u^{\text{new}}, \text{and } X_i^{\text{new}})$ in Eq. (10) are convolution results learned from the spectral domain with a parameter matrix $\Theta' \in \mathbb{R}^{C \times F}$. 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 to 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:

$$\begin{bmatrix} X_u^k \\ X_i^k \end{bmatrix} = sp(...sp(\begin{bmatrix} X_u^0 \\ X_i^0 \end{bmatrix}; U, \Lambda, \Theta_0'; U, \Lambda, \Theta_1'; \ldots; U, \Lambda, \Theta_{K-1}'), \quad (11)$$

where $\Theta_{K-1} \in \mathbb{R}^{F \times F}$ 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:

$$V^u = [X_u^0, X_u^1, \ldots, X_u^K] \quad \text{and} \quad V^i = [X_i^0, X_i^1, \ldots, X_i^K], \quad (12)$$

where $V^u \in \mathbb{R}^{(|U|C + KF)}$ and $V^i \in \mathbb{R}^{(|X_i|C + KF)}$.

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 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 $V^u$ and an item matrix $V^i$ as shown in Eq. (12), the loss function of SpectralCF is given as:

$$L = \arg \min_{V^u, V^i} \sum_{(r, j, j') \in \mathcal{D}} -\ln \sigma(v^u_r^T v^i_j - v^u_r^T v^i_{j'}) + \lambda_{reg} \left( ||V^u||_2^2 + ||V^i||_2^2 \right), \quad (13)$$

where $v^u_r$ and $v^i_j$ denote $r_{th}$ and $j_{th}$ column of $V^u$ and $V^i$, respectively; $\lambda_{reg}$ represents the weight on the regularization terms.
Recall@20

As discussed in the introduction section, leveraging the connectivity information learned from the spectral domain, the proposed SpectralCF 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?

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 SpectralCF 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 Bayesian Personalized Ranking 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]: Neural Collaborative Filtering 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]: Graph regularized Non-negative Matrix Factorization considers the graph structures by seeking a matrix factorization with a graph-based regularization.
- **GCMC** [2]: Graph Convolutional Matrix Completion 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 GCMM are originally designed for explicit datasets. For a fair comparison, we follow the setting of [13] to adapt them for implicit data.
We test our method as well as comparative models on three publicly available datasets:

- **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. 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:

\[
\text{Recall@M} = \frac{\# \text{items the user likes among the top M}}{\text{total number of items the user likes}}
\]  

(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 \(\lambda\) 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 \(\lambda\) 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 MovieLens-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 MovieLens-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.
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 $\frac{1}{P}$, from 1 to 5. The average results are reported and the best results are in bold. The standard deviation is shown in parentheses.

<table>
<thead>
<tr>
<th></th>
<th>$P$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Recall@20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BPR</td>
<td>SpectralCF</td>
<td>BPR</td>
<td>SpectralCF</td>
<td>BPR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.003)</td>
<td>(0.003)</td>
<td>(0.002)</td>
<td>(0.000)</td>
<td>(0.003)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.021</td>
<td>0.031</td>
<td>0.024</td>
<td>0.031</td>
<td>0.034</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.039</td>
<td>0.042</td>
<td>0.028</td>
<td>0.031</td>
<td>0.035</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.045</td>
<td>0.051</td>
<td>0.030</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>47.6%</td>
<td>34.5%</td>
<td>35.5%</td>
<td>32.4%</td>
<td>34.2%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAP@20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BPR</td>
<td>SpectralCF</td>
<td>BPR</td>
<td>SpectralCF</td>
<td>BPR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.002)</td>
<td>(0.000)</td>
<td>(0.002)</td>
<td>(0.000)</td>
<td>(0.003)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.014</td>
<td>0.021</td>
<td>0.024</td>
<td>0.028</td>
<td>0.031</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.024</td>
<td>0.028</td>
<td>0.031</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.031</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>35.7%</td>
<td>41.2%</td>
<td>33.3%</td>
<td>29.2%</td>
<td>29.6%</td>
</tr>
</tbody>
</table>

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

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

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 problem 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


