Deep Learning on Recommender Systems

BY

LEI ZHENG
M.E., Harbin Institute of Technology, 2013
B.S., Jilin University, 2010

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2019

Chicago, Illinois

Defense Committee:
Philip S. Yu, Chair and Advisor
Bing Liu
Xinhua Zhang
Bhaskar DasGupta
Jiawei Zhang, Florida State University

This dissertation is dedicated to my parents, my wife and my daughter

for their encouragement, support and unconditional love.

ACKNOWLEDGMENTS

First, I would like to thank my Ph.D. advisor, Prof. Philip S. Yu, for his guidance and precious patience along the way. Without his suggestions and support, this dissertation would not have been possible.

Besides, I would like to thank the rest of my dissertation committee: Prof. Bing Liu, Prof. Bhaskar DasGupta, Prof. Xinhua Zhang and Prof. Jiawei Zhang for being as my dissertation committee members. I am grateful for their valuable remarks and insightful advises for this dissertation.

I am also grateful to all the friends I met at the University of Illinois at Chicago. We together have countless fun and happy moments in all these years. I also want to thank people I met at Pinterest Research Lab. Especially, my sincere thanks also go to my mentor, Dan Xie, for guiding me through during my internship.

At last, I would like to thank my parents, who offered me support and love since I am a child. Especially, I would like to thank my wife, Yahan Hsu, for her love and support. Without her, I would not be able to finish this dissertation.

GM

CONTRIBUTION OF AUTHORS

Chapter 1 is an introduction of this dissertation. Chapter 2 presents a published manuscript (Zheng et al., 2017) for which I was the primary author. Vahid Noroozi drafted a part of the manuscript. Prof. Philip S. Yu contributed to discussion regarding to the work and suggestions for revising the manuscript.

Chapter 3 presents a published manuscript (Zheng et al., 2018b), for which I was the primary author. Dr. Chun-Ta Lu and Fei Jiang contributed to the draft of the manuscript. Prof. Jiawei Zhang and Prof. Philip S. Yu provided valuable advises regarding to the work and suggestions for revising the manuscript.

Chapter 4 presents a published manuscript (Zheng et al., 2019) for which I was the primary author. Ziwei Fan, Dr. Chun-Ta Lu, Prof. Jiawei Zhang, and Prof. Philip S. Yu contributed advises regarding to the work and suggestions for revising the manuscript.

Chapter 5 presents a published manuscript (Zheng et al., 2019) for which I was the primary author. Chaozhuo Li, Dr. Chun-Ta Lu, Prof. Jiawei Zhang, and Prof. Philip S. Yu gave valuable advises with respect to the work and suggestions for revising the manuscript.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BPR	Bayesian Personalized Ranking
CDL	Collaborative Deep Learning
CF	Collaborative Filtering
CML	Collaborative Metric Learning
CNN	Convolutional Neural Network
CTR	Collaborative Topic Regression
DBN	Deep Belief Network
DDN	Deep Distribution Network
DeepCoNN	Deep Cooperative Neural Networks
FM	Factorization Machine
GCMC	Graph Convolutional Matrix Completion
GNMF	Graph regularized Non-negative Matrix Factorization
GSUs	Gated Spectral Units
HFT	Hidden Factors as Topic
LDA	Latent Dirichlet Allocation
MAP	Mean Average Precision

LIST OF ABBREVIATIONS (Continued)

MC	Markov Chain
MF	Matrix Factorization
MLP	Multi-Layer Perceptron
MSE	Mean Square Error
NCF	Neural Collaborative Filtering
NLP	Natrual Language Processing
NN	Neural Network
PMF	Probabilistic Matrix Factorization
RBM	Restricted Boltzmann Machines
ReLUs	Rectified Linear Units
RS	Recommender Systems
RNN	Recurrent Neural Network
SEGs	Sequential Evolving Graphs
SpectralCF	Spectral Collaborative Filtering

SUMMARY

The so-called cold-start problem has haunted the recommender systems community for years. The problem happens when a user rated/clicked/liked a few number of items. Classic approaches, such as collaborative filtering, assume that a user has a fair amount of actions so that the preferences of the user can be inferred. As a result, traditional methods cannot effectively model the interests of cold users due to the scarcity of data.

In this dissertation, I will introduce our recent works on employing deep learning methods for alleviating the cold-start problem for recommendation. In the first part, we focus on utilizing review data to build deep learning models to ease the cold start problem. In the second part, I present an spectral approach to discover users' interests from the spectral domain of the user-item bipartite graph. In the third part, I introduce a recurrent method designed to capture users' evolving interests from dynamic graphs. In the fourth part, we propose to model users and items with probability distributions, rather than the popular vectors. With distribution-based representations, the proposed model is able to alleviate the cold-start problem and therefore, delivers the start-of-the-art performances in three real-world datasets.

CHAPTER 1

INTRODUCTION

1.1 Dissertation Outline

This dissertation focuses on alleviating the cold-start problem in multiple perspectives. Specifically, four different tasks are covered to easy the cold-start problem:

- To ease the cold-start problem, we propose Deep Cooperative Neural Networks (DeepCoNN). DeepCoNN leverages review data to characterize user preferences and item properties to alleviate the cold-start problem for recommendation.
- We propose a spectral framework for recommendation to analyze users' preferences from the spectral domain, where not only the proximity information but also the connectivity information can be revealed.
- In order to leverage co-evolving patterns from users' actions for sequential recommendation, we introduce Gated Spectral Units (GSUs) to discover users' evolving interests from spectral domains of dynamic graphs.
- To combat the cold-start problem, we propose to model users and items with probability distributions, instead of vectors. Since distributions are proved to be effective to handle sparse data, the proposed model, Deep Distribution Network (DDN), diminish the negative impact of the cold start problem and delivers the start-of-the-art performances in three real-world datasets.

1.2 Review-based Deep Recommender Systems

(Part of this chapter was previously published in (Zheng et al., 2017).)

On one hand, in many recommender systems, other than the numeric ratings, users are allowed to write reviews for products. Users explain the reasons behind their ratings in text reviews. The reviews contain information which can be used to alleviate sparsity problem. One of the drawbacks of most current collaborative filtering (CF) techniques is that they model users and items just based on the numeric ratings provided by users and ignore the abundant information existed in the review text. Recently, some studies (McAuley and Leskovec, 2013) (Ling et al., 2014) have shown that using review text can improve the prediction accuracy of recommender systems, in particular for the items and users with few ratings (Wang et al., 2010). On the other hand, deep learning methods have shown a promising performance in analyzing and modeling textual data in a variety of tasks (Collobert et al., 2011; Zheng et al., 2017; Zheng and Han, 2013; Zheng, 2016). Thus, this calls for a review-based deep recommender system leveraging review data for alleviating the cold-start problem.

In Chapter 2, we present a deep model to learn item properties and user behaviors jointly from review text. The proposed model, named Deep Cooperative Neural Networks (DeepCoNN), consists of two parallel neural networks coupled in the last layers. One of the networks focuses on learning user behaviors exploiting reviews written by the user, and the other one learns item properties from the reviews written for the item. A shared layer is introduced on the top to couple these two networks together. The shared layer enables latent factors learned for users and items to interact with each other in a manner similar to factorization machine

techniques. Experimental results demonstrate that DeepCoNN significantly outperforms all baseline recommender systems on a variety of datasets.

1.3 Spectral Collaborative Filtering

(Part of this chapter was previously published in (Zheng et al., 2018b))

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. In Chapter 3, 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.

1.4 Modeling Co-evolving Patterns for Sequential Recommendation

(Part of this chapter was previously published in (Zheng et al., 2019).)

Exploiting historical data of users to make future predictions lives at the heart of building effective recommender systems. Recent approaches for sequential recommendations often render past actions of a user into a sequence, seeking to capture the temporal dynamics in the sequence to predict the next item. However, the interests of users evolve over time together due to their mutual influence, and most of existing methods lack the ability to utilize the rich coevolutionary patterns available in underlying data represented by sequential graphs.

In order to capture the co-evolving knowledge for sequential recommendations, in Chapter 4, we start from introducing an efficient spectral convolution operation to discover complex relationships between users and items from the spectral domain of a graph, where the hidden connectivity information of the graph can be revealed. Then, the spectral convolution is generalized into an recurrent method by utilizing gated mechanisms to model sequential graphs. Experimentally, we demonstrate the advantages of modeling *co-evovling patterns*, and Gated Spectral Units (GSUs) achieve state-of-the-art performance on several benchmark datasets.

1.5 Distribution-based Representations for Top-N Recommendation

(Part of this chapter was previously published in (Zheng et al., 2019).)

Existing recommendation methods mostly learn fixed vectors for users and items in a low-dimensional continuous space, and then calculate the popular dot-product to derive user-item distances. However, these methods suffer from two drawbacks: (1) the data sparsity issue prevents from learning high-quality representations; and (2) the dot-product violates the crucial triangular inequality and therefore, results in a sub-optimal performance.

In Chapter 5, in order to overcome the two aforementioned drawbacks, we propose Deep Distribution Network (DDN) to model users and items via Gaussian distributions. We argue that, compared to fixed vectors, distribution-based representations are more powerful to characterize users' uncertain interests and items' distinct properties. In addition, we propose a Wasserstein-based loss, in which the critical triangular inequality can be satisfied. In experiments, we evaluate DDN and comparative models on standard datasets. It is shown that DDN significantly outperforms state-of-the-art models, demonstrating the advantages of the proposed distribution-based representations and wassertein loss.

CHAPTER 2

REVIEW-BASED DEEP RECOMMENDER SYSTEMS

(This chapter was previously published as "Joint deep modeling of users and items using reviews for recommendation", in the Tenth International Conference on Web Search and Data Mining (WSDM'17) (Zheng et al., 2017). DOI: https://doi.org/10.1145/3018661.3018665.)

2.1 Introduction

Many of the prominent approaches employed in recommender systems (Koren et al., 2009) are based on Collaborative Filtering (CF) techniques. Many of the most successful CF techniques are based on matrix factorization (Koren et al., 2009). Although CF techniques have shown good performance for many applications, the sparsity problem is considered as one of their significant challenges (Koren et al., 2009). The sparsity problem arises when the number of items rated by users is insignificant to the total number of items. It happens in many real applications. It is not easy for CF techniques to recommend items with few ratings or to give recommendations to the users with few ratings.

One of the approaches employed to address this lack of data is using the information in review text (Ling et al., 2014; McAuley and Leskovec, 2013). In many recommender systems, users explain the reasons behind their ratings in text reviews. The reviews contain information which can be used to alleviate sparsity problem. One of the drawbacks of most current CF

techniques is that they model users and items just based on the numeric ratings provided by users and ignore the abundant information existed in the review text. Recently, some studies (McAuley and Leskovec, 2013) (Ling et al., 2014) have shown that using review text can improve the prediction accuracy of recommender systems, in particular for the items and users with few ratings (Wang et al., 2010).

In this paper, we propose a neural network (NN) based model, named Deep **Co**operative **N**eural **N**etworks (DeepCoNN), to model users and items jointly using review text for rating prediction problems. One of the networks models user behavior using the reviews written by the user, and the other network models item properties using the written reviews for the item. The learned latent features for user and item are used to predict the corresponding rating in a layer introduced on the top of both networks. This interaction layer is motivated by matrix factorization techniques (Koren et al., 2009) to let latent factors of users and items interact with each other.

To the best of our knowledge, DeepCoNN is the first deep model that represents both users and items in a joint manner using reviews. It makes the model scalable and also suitable for online learning scenarios where the model needs to get updated continuously with new data. Another key contribution is that DeepCoNN represents review text using pre-trained word-embedding technique (Mikolov et al., 2013), (Mikolov et al., 2010) to extract semantic information from the reviews. Recently, this representation has shown excellent results in many Natural Language Processing (NLP) tasks (Collobert et al., 2011; Bengio et al., 2006). Moreover, a significant advantage of DeepCoNN compared to most other approaches (McAuley

and Leskovec, 2013; Ling et al., 2014) which benefit from reviews is that it models users and items in a joint manner with respect to prediction accuracy. Most of the similar algorithms perform the modeling independently of the ratings. Therefore, there is no guarantee that the learned factors can be beneficial to the rating prediction.

The experiments on real-world datasets including *Yelp*, *Amazon* (McAuley et al., 2015a), and *Beer* (McAuley et al., 2012) show that DeepCoNN outperforms all the compared baselines in prediction accuracy. Also, the proposed algorithm increases the performance for users and items with fewer ratings more than the ones with a higher number of ratings. It shows that DeepCoNN alleviates the sparsity problem by leveraging review text.

Our contributions and also advantages of DeepCoNN can be summarized as follows:

- The proposed Deep Cooperative Neural Networks (DeepCoNN) jointly model user behaviors and item properties using text reviews. The extra shared layer at the top of two neural networks connects the two parallel networks such that user and item representations can interact with each other to predict ratings. To the best of our knowledge, DeepCoNN is the first one that jointly models both user and item from reviews using neural networks.
- It represents review text as word-embeddings using pre-trained deep models. The experimental results demonstrate that the semantic meaning and sentimental attitudes of reviews in this representation can increase the accuracy of rating prediction. All competing techniques which are based on topic modeling (Wu and Ester, 2015; Bao et al., 2014; Diao et al., 2014) use the traditional *bag of words* techniques.

- It does not only alleviate the problem of sparsity by leveraging reviews, but also improves the overall performance of the system significantly. It outperforms state-of-the-art techniques (McAuley and Leskovec, 2013; Salakhutdinov and Mnih, 2007; Wang et al., 2015) in terms of prediction accuracy on all of the evaluated datasets including Yelp, 21 categories of Amazon, and Beer (see Section 3.4).

In Section 2.2, we describe DeepCoNN in detail. Experiments are presented in Section 3.4 to analyze DeepCoNN and demonstrate its effectiveness compared to the state-of-the-art techniques for recommendation systems. In Section 2.4, we give a short review of the works related to our study.

2.2 Methodology

The proposed model, DeepCoNN, is described in detail in this section. DeepCoNN models user behaviors and item properties using reviews. It learns hidden latent factors for users and items by exploiting review text such that the learned factors can estimate the ratings given by users. It is done with a CNN based model consisting of two parallel neural networks, coupled to each other with a shared layer at the top. The networks are trained in a joint manner to predict the ratings with minimum prediction error. We first describe notations used throughout this paper and formulate the definition of our problem. Then, the architecture of DeepCoNN and the objective function to get optimized is explained. Finally, we describe how to train this model.

2.2.1 Definition and Notation

A set of training set $\mathcal{T}$ consists of N tuples. Each tuple (u, i, r_{ui}, w_{ui}) denotes a review written by user u for item i with rating r_{ui} and text review of w_{ui}. The mathematical notations used in this paper are summarized in Table I.

TABLE I: Notations

Symbols	Definitions and Descriptions
$d^u_{1:n}$	user or item u's review text consisting of n words
$V^u_{1:n}$	word vectors of user or item u
w_{ui}	a review text written by user u for item i
o_j	the output of j_{th} neuron in the convolutional layer
n_i	the number of neurons in the layer i
K_j	the j_{th} kernel in the convolutional layer
b_j	the bias of j_{th} convolutional kernel
g	the bias of the fully connected layer
z_j	the j_{th} feature map in the convolutional layer
W	the weight matrix of the fully connected layer
t	the window size of convolutional kernel
c	the dimension of word embedding
$\mathbf{x_u}$	the output of Net_u
$\mathbf{y_i}$	the output of Net_i
λ	the learning rate

2.2.2 Architecture

The architecture of the proposed model for rating prediction is shown in Figure 1. The model consists of two parallel neural networks coupled in the last layer, one network for users (Net_u) and one network for items (Net_i). User reviews and item reviews are given to Net_u and

Net_i respectively as inputs, and corresponding rating is produced as the output. In the first layer, denoted as look-up layer, review text are utilized to capture the semantic information in the review text. Next layers are CNN based models, including convolution layer, max pooling layer, and fully connected layer. Also, a top layer is added on the top of the two networks to let the hidden latent factors of user and item interact with each other. This layer calculates an objective function that measures the rating prediction error using the latent factors produced by Net_u and Net_i. In the following subsections, since Net_u and Net_i only differ in their inputs, we focus on illustrating the process for Net_u in detail. The same process is applied for Net_i with similar layers.

2.2.3 Word Representation

A word embedding $f : M \rightarrow \Re^n$, where M represents the dictionary of words, is a parameterized function mapping words to n-dimensional distributed vectors. Recently, this approach has boosted the performance in many NLP applications (Kim, 2014), (Collobert et al., 2011). DeepCoNN uses this representation technique to exploit the semantics of reviews. In the look-up layer, reviews are represented as a matrix of word embeddings to extract their semantic information. To achieve it, all the reviews written by user u, denoted as user reviews, are merged into a single document $d^u_{1:n}$, consisting of n words in total. Then, a matrix of word vectors, denoted as $V^u_{1:n}$, is built for user u as follows:

$$V^u_{1:n} = \phi(d^u_1) \oplus \phi(d^u_2) \oplus \phi(d^u_3) \oplus \quad ... \quad \oplus \phi(d^u_n), \tag{2.1}$$

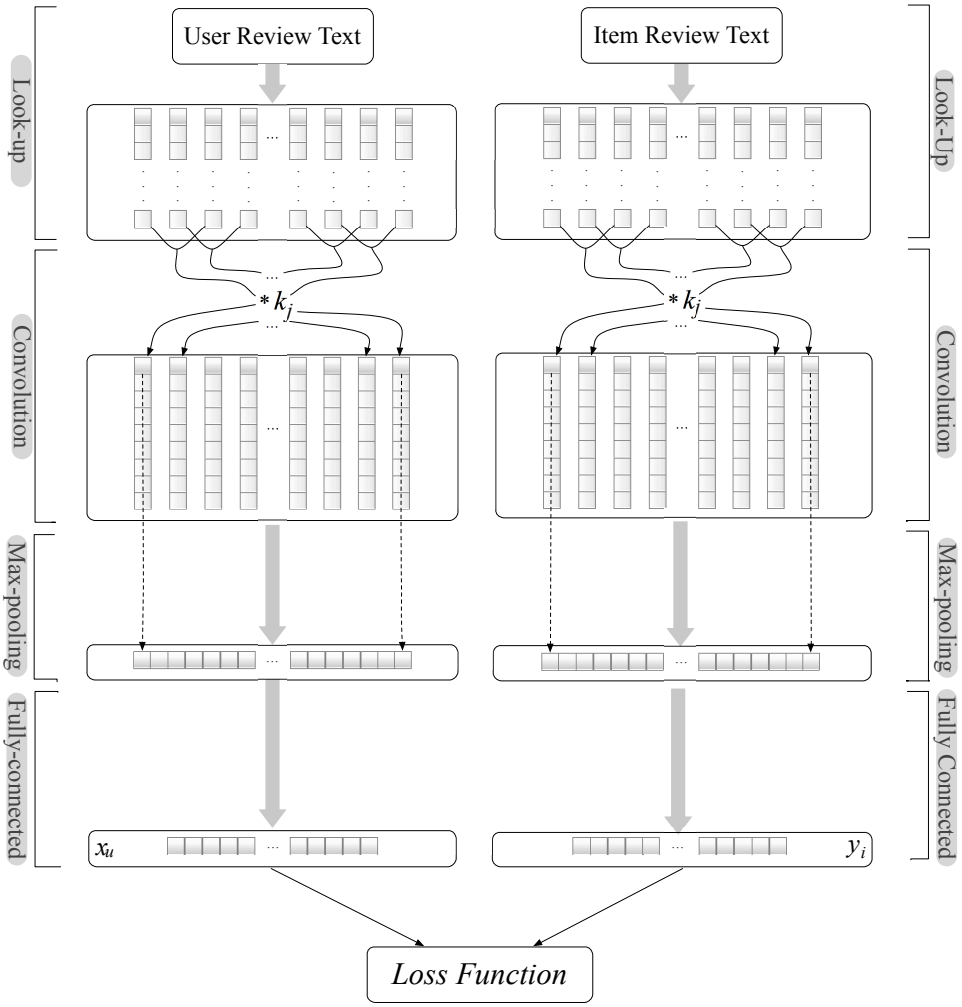

Figure 1: The architecture of the proposed model

where d_k^u indicates the kth word of document $d_{1:n}^u$, look-up function $\phi(d_k^u)$ returns the corresponding c-dimensional word vector for the word d_k^u, and $\oplus$ is the concatenation operator. It should be considered that the order of words is preserved in matrix $V_{1:n}^u$ that is another advantage of this representation comparing to *bag-of-words* techniques.

2.2.4 CNN Layers

Next layers including convolution layer, max pooling, and fully connected layer follow the CNN model introduced in (Collobert et al., 2011). Convolution layer consists of m neurons which produce new features by applying convolution operator on word vectors $V^u_{1:n}$ of user u. Each neuron j in the convolutional layer uses filter $K_j \in \Re^{c \times t}$ on a window of words with size t. For $V^u_{1:n}$, we perform a convolution operation regarding each kernel K_j in the convolutional layer.

$$z_j = f(V^u_{1:n} * K_j + b_j) \tag{2.2}$$

Here symbol $*$ is convolution operator, b_j is a bias term and f is an activation function. In the proposed model, we use Rectified Linear Units (ReLUs) (Nair and Hinton, 2010). It is defined as Equation 2.3. Deep convolutional neural networks with ReLUs train several times faster than their equivalents with tanh units (Krizhevsky et al., 2012).

$$f(x) = max\{0, x\} \tag{2.3}$$

Following the work of (Collobert et al., 2011), we then apply Equation 2.4. The most important feature of each feature map, which has the highest value, has been captured. This pooling scheme can naturally deal with the varied length of the text. After the max pooling operation, convolutional results are reduced to a fixed size vector.

$$o_j = max\{z_1, z_2, ..., z_{(n-t+1)}\} \tag{2.4}$$

We have described the process by which one feature is extracted from one kernel. The model uses multiple filters to obtain various features and the output vector of the convolutional layer is as Equation 2.5.

$$O = \{o_1, o_2, o_3, ..., o_{n_1}\}, \tag{2.5}$$

where n_1 denotes the number of kernel in the convolutional layer.

$$\mathbf{x_u} = f(W \times O + g) \tag{2.6}$$

The results from the max-pooling layer are passed to a fully connected layer with weight matrix W. As shown in Equation 2.6, the output of the fully connected layer $\mathbf{x_u} \in \Re^{n_2 \times 1}$ is considered as features for user u. Finally, the outputs of both user and item CNN $\mathbf{x_u}$ and $\mathbf{y_i}$ can be obtained.

2.2.5 The Shared Layer

Although these outputs can be viewed as features of users and items, they can be in different feature space and not comparable. Thus, to map them into the same feature space, we introduce a shared layer on the top to couple Net_u and Net_i. First, let us concatenate $\mathbf{x_u}$ and $\mathbf{y_i}$ into a single vector $\hat{\mathbf{z}} = (\mathbf{x_u}, \mathbf{y_i})$. To model all nested variable interactions in $\hat{\mathbf{z}}$, we introduce Factorization Machine (FM) (Rendle, 2012) as the estimator of the corresponding

rating. Therefore, given a batch of N training examples $\mathcal{T}$, we can write down its cost as Eq. Equation 2.7.

$$J = \hat{w}_0 + \sum_{i=1}^{|\hat{z}|} \hat{w}_i \hat{z}_i + \sum_{i=1}^{|\hat{z}|} \sum_{j=i+1}^{|\hat{z}|} \langle \hat{\mathbf{v}}_i, \hat{\mathbf{v}}_j \rangle \hat{z}_i \hat{z}_j, \tag{2.7}$$

where $\hat{w}_0$ is the global bias, $\hat{w}_i$ models the strength of the i_{th} variable in $\hat{z}$ and $\langle \hat{\mathbf{v}}_i, \hat{\mathbf{v}}_j \rangle = \sum_{f=1}^{|\hat{z}|} \hat{\mathbf{v}}_{i,f} \hat{\mathbf{v}}_{j,f}$. $\langle \hat{\mathbf{v}}_i, \hat{\mathbf{v}}_j \rangle$ models the second order interactions.

2.2.6 Network Training

Our network is trained by minimizing Equation 2.7. We take derivatives of J with respect to z, as shown in Equation 2.8.

$$\frac{\partial J}{\partial \hat{z}_i} = \hat{w}_i + \sum_{j=i+1}^{|\hat{z}|} \langle \hat{\mathbf{v}}_i, \hat{\mathbf{v}}_j \rangle \hat{z}_j \tag{2.8}$$

The derivatives of other parameters in different layers can be computed by applying differentiation chain rule.

Given a set of training set $\mathcal{T}$ consisting of N tuples, we optimize the model through RMSprop (Tieleman and Hinton, 2012) over shuffled mini-batches. RMSprop is an adaptive version of gradient descent which adaptively controls the step size with respect to the absolute value of the gradient. It does it by scaling the update value of each weight by a running average of its gradient norm. The updating rules for parameter set θ of the networks are as the following:

$$r_t \leftarrow 0.9(\frac{\partial J}{\partial \theta})^2 + 0.1 r_{t-1} \tag{2.9}$$

$$\theta \leftarrow \theta - (\frac{\lambda}{\sqrt{r_t} + \epsilon})\frac{\partial J}{\partial \theta}, \tag{2.10}$$

where λ is the learning rate, ϵ is a small value added for numerical stability. Additionally, to prevent overfitting, the dropout (Srivastava et al., 2014) strategy has also been applied to the fully connected layers of the two networks.

2.2.7 Some Analysis on DeepCoNN

2.2.7.1 Word Order Preservation

Most of the recommender systems which use reviews in the modeling process employ topic modeling techniques to model users or items (Chen et al., 2015). However, in many text modeling applications, word order is crucial (Wallach, 2006). DeepCoNN is not based on topic modeling and uses word embeddings to create a matrix of word vectors where the order of words are preserved. In this way, convolution operations make use of the internal structure of data and provide a mechanism for efficient use of words' order in text modeling (Johnson and Zhang, 2015).

2.2.7.2 Online Learning

Scalability and handling dynamic pools of items and users are considered as critical needs of many recommender systems. The time sensitivity of recommender systems poses a challenge in learning latent factors in an online fashion. DeepCoNN is scalable to the size of the training data, and also it can easily get trained and updated with new data because it is based on NN. Updating latent factors of items or users can get performed independently from historical

TABLE II: The Statistics of the datasets

Class	#users	#items	#review	#words	#reviews per user	#words per review
Yelp	366,715	60,785	1,569,264	198M	4.3	126.41
Amazon	6,643,669	2,441,053	34,686,770	4.053B	5.2	116.67
Beer	40,213	110,419	2,924,127	154M	72.7	52.67

data. All the approaches which employ topic modeling techniques do not benefit from these advantages to this extent.

2.3 Experiments

We have performed extensive experiments on a variety of datasets to demonstrate the effectiveness of DeepCoNN compared to other state-of-the-art recommender systems. We first present the datasets and the evaluation metric used in our experiments in Section 2.3.1. The baseline algorithms selected for comparisons are explained in Section 2.3.2. Experimental settings are given in Section 2.3.3.

2.3.1 Datasets and Evaluation Metric

In our experiments, we have selected the following three datasets to evaluate our model.

- **Yelp**: It is introduced in the 6th round of *Yelp Challenge* and a large-scale dataset consisting of restaurant reviews containing more than 1M reviews and ratings.

- **Amazon**: It (McAuley et al., 2015a) contains product reviews and metadata from Amazon website[1]. It includes more than 143.7 million reviews spanning from May 1996 to July 2014. It has 21 categories of items, and as far as we know, this is the largest public available rating dataset with text reviews.
- **Beer**: It is a beer review dataset extracted from *ratebeer.com*.

As we can see in Table II, all datasets contain more than half a million of reviews. However, in *Yelp* and *Amazon*, customers provide less than six pair of reviews and ratings on average which shows these two datasets are extremely sparse. This sparsity can largely deteriorate the performance of recommender systems. Besides, in all datasets, each review consists of less than 150 words on average.

In our experiments, we adopt the well-known Mean Square Error (MSE) to evaluate the performance of the algorithms. It is selected because most of the related works have used the same evaluation metric(McAuley and Leskovec, 2013; Ling et al., 2014; Almahairi et al., 2015). MSE can be defined as follows:

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(r_n - \hat{r}_n)^2, \tag{2.11}$$

where r_n is the nth observed value, $\hat{r}_n$ is the nth predicted value and N is the total number of observations.

---

[1]https://snap.stanford.edu/data/web-Amazon.html

2.3.2 Baselines

To validate the effectiveness of DeepCoNN, we have selected three categories of algorithms for evaluations: (i) purely rating based models. We chose Matrix Factorization (MF) and Probabilistic Matrix Factorization (PMF) to validate that review information is helpful for recommender systems, (ii) topic modeling based models which use review information. Most of the recommender systems which take reviews into consideration are based on topic modeling techniques, and (iii) deep recommender systems. In (Wang et al., 2015), authors have proposed a state-of-the-art deep recommender system named Collaborative Deep Learning (CDL). Note that all the baselines except MF and PMF have incorporated review information into their models to improve prediction.

- **MF**: **M**atrix **F**actorization (Koren et al., 2009) is the most popular CF-based recommendation method. It only uses rating matrix as input and estimates two low-rank matrices to predict ratings. In our implementation, Alternating Least Squares (ALS) technique is adopted to minimize its objective function.
- **PMF**: **P**robabilistic **M**atrix **F**actorization is introduced in (Salakhutdinov and Mnih, 2007). It models latent factors of users and items by Gaussian distributions.
- **LDA**: **L**atent **D**irichlet **A**llocation is a well-known topic modeling algorithm presented in (Blei et al., 2003).

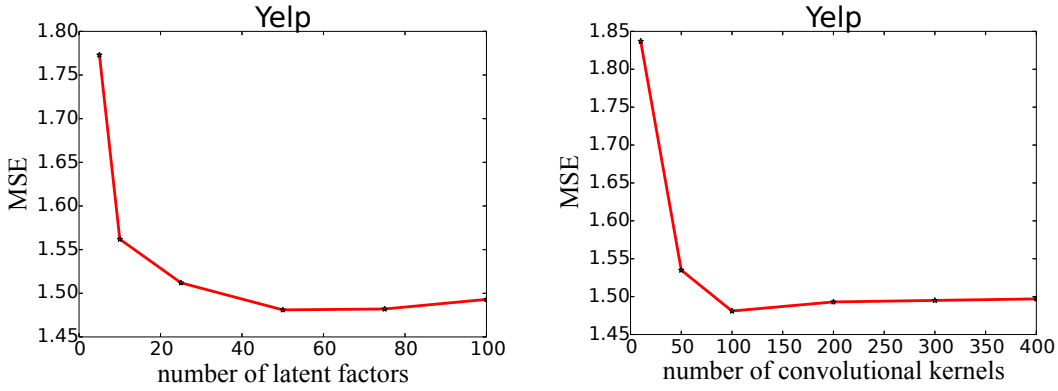

Figure 2: The impact of the number of latent factors and convolutional kernels on the performance of DeepCoNN in terms of MSE (*Yelp* Dataset).

- **CTR**: **C**ollaborative **T**opic **R**egression has been proposed by (Wang and Blei, 2011). It showed very good performance on recommending articles in a one-class collaborative filtering problem where a user is either interested or not.
- **HFT**: **H**idden **F**actor as **T**opic proposed in (McAuley and Leskovec, 2013) employs topic distributions to learn latent factors from user or item reviews. The authors have shown that item specific topic distributions produce more accurate predictions than user specific ones. Thus, we report the results of HFT learning from item reviews.
- **CDL**: **C**ollaborative **D**eep **L**earning combines a stacked denoising auto-encoders with a probabilistic matric factorization (PMF).

2.3.3 Experimental Settings

We divided each dataset shown in Table II into three sets of training set, validation set, and test set. We use 80% of each dataset as the training set, 10% is treated as the validation set

to tune the hyper-parameters, and the rest is used as the test set. All the hyper-parameters of the baselines and DeepCoNN are selected based on the performance on the validation set.

For MF and PMF, we used grid search to find the best values for the number of latent factors from $\{25, 50, 100, 150, 200\}$, and regularization parameter from $\{0.001, 0.01, 0.1, 1.0\}$.

For LDA, CTR and HFT, we searched the number of topics K from {5, 10, 20, 50, 100} using the validation set. We set $K = 10$ for LDA and CTR. The CTR model solves the *one-class collaborative filtering problem* (Pan et al., 2008) by using two different values for the precision parameter c of a Gaussian distribution. Following the work of (Ling et al., 2014), in our experiments, we set precision c as the same for all the observed ratings for rating prediction. HFT-k ($k = 10, 50$) are included to show the impact of the number of latent factors for HFT. By performing a grid search on the validation set, we set hyper-parameters $\alpha = 0.1$, $\lambda_u = 0.02$ and $\lambda_v = 10$ for CTR and HFT. To optimize the performance of CDL, we performed a grid search on the hyper-parameters λ_u, λ_v, λ_n, λ_w and L. Similar with CTR, the confidence parameter c_{ij} of CDL is set as the same for all observed ratings.

We empirically studied the effects of two important parameters of DeepCoNN: the number of latent factors($|\mathbf{x_u}|$ and $|\mathbf{y_i}|$) and the number of convolutional kernels: n_1. In Figure 2, we show the performance of DeepCoNN on the validation set of *Yelp* with varying $|\mathbf{x_u}|$ and $|\mathbf{y_i}|$ from 5 to 100 and n_1 from 10 to 400 to investigate its sensitivity. As it can be seen, it does not improve the performance when the number of latent factors and number of kernels is greater than 50 and 100 respectively. Thus, we set $|\mathbf{x_u}| = |\mathbf{y_i}| = 50$ and $n_1 = 100$. Other hyper-parameters: t, c, λ and batch size are set as 3, 300, 0.002 and 100, respectively. These values were chosen

TABLE III: MSE Comparison with baselines. Best results are indicated in bold.

Dataset	MF	PMF	LDA	CTR	HFT-10	HFT-50	CDL	Deep-CoNN	IMPV of DeepCoNN (%)
Yelp	1.792	1.783	1.788	1.612	1.583	1.587	1.574	**1.441**	8.5%
Amazon	1.471	1.460	1.459	1.418	1.378	1.383	1.372	**1.268**	7.6%
Beer	0.612	0.527	0.306	0.305	0.303	0.302	0.299	**0.273**	8.7%
Average on all datasets	1.292	1.256	1.184	1.112	1.088	1.09	1.081	**0.994**	8.3%

through a grid search on the validation sets. We used a pre-trained word embeddings which are trained on more than 100 billion words from Google News (Mikolov et al., 2013) [1].

Our models are implemented in *Theano* (Theano Development Team, 2016), a well-known Python library for machine learning and deep learning. The NVIDIA CUDA Deep Neural Network4 (cuDNN v4) accelerated our training process. All models are trained and tested on an NVIDIA Tesla K40 GPU.

2.3.4 Performance Evaluation

The performance of DeepCoNN and the baselines (see Section 2.3.2) are reported in terms of MSE in Table III. Table III shows the results on the three datasets including the performance averaged on all 21 categories of *Amazon*. The last column indicates the percentage of improvements gained by DeepCoNN compared to the best baseline in the corresponding category.

---

[1] https://code.google.com/archive/p/word2vec/

In Table III, all models perform better on *Beer* dataset than on *Yelp* and *Amazon*. It is mainly related to the sparsity of *Yelp* and *Amazon*. Although PMF performs better than MF on *Yelp*, *Beer*, and most categories of *Amazon*, both techniques do not show good performance compared to the ones which use reviews. It validates our hypothesis that review text provides additional information, and considering reviews in models can improve rating prediction.

Although simply employing LDA to learn features from item reviews can help the model to achieve improvements, LDA models reviews independent of ratings. Therefore, there is no guarantee that the learned features can be beneficial to rating prediction. Therefore, by modeling ratings and reviews together, CTR and HFT attain additional improvements. Among those topic modeling based models (LDA, CTR and HFT), both HFT-10 and HFT-50 perform better in all three datasets.

With the capability of extracting deep effective features from item review text, as we can see in Table III, CDL outperforms all topic modeling based recommender systems and advances the state-of-the-art. However, in benefiting from joint modeling capacity and semantic meaning existing from review text, DeepCoNN beats the best baseline in *Yelp*, *Beer* and *Amazon* and gains **8.3%** improvement on average.

2.3.5 Model Analysis

Are the two parallel networks really cooperate to learn effective features from reviews? Does the proposed model benefit from the use of word embedding to exploit the semantic information in the review text? How much does the shared layer help in improving the predcition accuracy comparing to a simpler coupling approach? To answer these questions, we compare

TABLE IV: Comparing variants of the proposed model. Best results are indicated in bold.

Model	Yelp	Amazon Music Instruments	Beer
DeepCoNN-User	1.577	1.373	0.292
DeepCoNN-Item	1.578	1.372	0.296
DeepCoNN-TFIDF	1.713	1.469	0.589
DeepCoNN-Random	1.799	1.517	0.627
DeepCoNN-DP	1.491	1.253	0.278
DeepCoNN	**1.441**	**1.233**	**0.273**

the DeepCoNN with its five variants: DeepCoNN-User, DeepCoNN-Item, DeepCoNN-TFIDF, DeepCoNN-Random and DeepCoNN-DP. These five variants are summarized in the following:

- **DeepCoNN-User**: The Net_i of DeepCoNN is substituted with a matrix. Each row of the matrix is the latent factors of one item. This matrix is randomly initialized and optimized during the training.
- **DeepCoNN-Item**: Similar with DeepCoNN-User, the Net_u of DeepCoNN is replaced with a matrix. Each row of the matrix is the latent factors of one user. This matrix is randomly initialized and optimized during the training.
- **DeepCoNN-TFIDF**: Instead of using word embedding, the TFIDF scheme is employed to represent review text as input to DeepCoNN.
- **DeepCoNN-Random**: Our baseline model where all word representations are randomly initialized as fixed-length vectors.
- **DeepCoNN-DP**: The factorization machine in the objective function is substitued with a simple dot product of $\mathbf{x_u}$ and $\mathbf{y_i}$.

The performance of DeepCoNN and its variants on *Yelp*, *Beer* and one category of the *Amazon* dataset: *Music Instruments* are given in Table IV.

To demonstrate that the two deep CNNs can cooperate with each other to learn effective latent factors from user and item reviews, DeepCoNN-User and DeepCoNN-Item are trained with only one CNN with review text as input and the other CNN is substituted with a list of latent variables as the parameters to get learned. In this manner, latent factors of users or items are learned without considering their corresponding review text. As it can be seen in Table IV, while DeepCoNN-User and DeepCoNN-Item achieve similar results, DeepCoNN delivers the best performance by modeling both users and items. It verifies that review text is necessary for modeling latent factors of both users and items. Also, it shows that review text has informative information that can help to improve the performance of recommendation.

Furthermore, to validate the effectiveness of word representation, we compare DeepCoNN with DeepCoNN-TFIDF and DeepCoNN-Random. The DeepCoNN-TFIDF and DeepCoNN-Random are trained to show that word embedding is helpful to capture semantic meaning existed in the review text. While the performance of DeepCoNN-TFIDF is slightly better than DeepCoNN-Random, they both perform considerably weaker than DeepCoNN. It shows the effectiveness of representing review text in semantic space for modeling the latent factors of items or users.

At last, to investigate the efficiency of the shared layer, DeepCoNN-DP is introduced that couples the two networks with a simpler objective function. The comparison shows the superi-

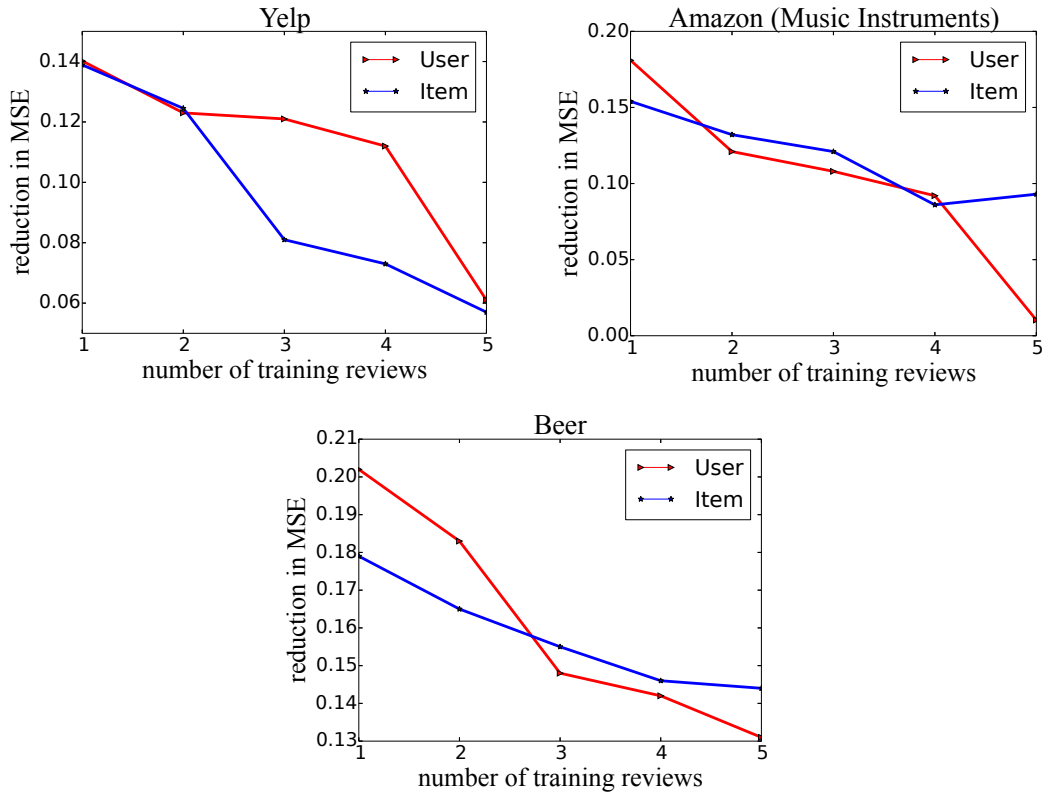

Figure 3: MSE improvement achieved by DeepCoNN compared to MF. For users and items with different number of training reviews, DeepCoNN gains different MSE reductions.

ority of the factorization machine coupling. It can be the result of not only modeling the first order interactions but also the second order interactions between $\mathbf{x_u}$ and $\mathbf{y_i}$.

2.3.6 The Impact of the Number of Reviews

The cold start problem (Schein et al., 2002) is prevalent in recommender systems. In particular, when a new user joins or a new item is added to the system, their available ratings are limited. It would not be easy for the system to learn preferences of such users just from their ratings. It has been shown in some of the previous works that exploiting review text can help

to alleviate this problem especially for users or items with few ratings (McAuley and Leskovec, 2013). In this section, we conduct a set of experiments to answer the following questions. Can DeepCoNN help to tackle the cold start problem? What is the impact of the number of reviews on the effectiveness of the proposed algorithm?

In Figure 3, we have illustrated the reductions in MSE resulted from DeepCoNN compared to MF technique on three datasets of Yelp, Beer, and a group of Amazon (Music Instruments). By reduction in MSE, we mean the difference between the MSE of MF and the MSE of DeepCoNN. Users and items are categorized based on the number of their reviews, and reductions are plotted for both users and items groups. It can be seen that in all three datasets, reductions are positive, and DeepCoNN can achieve RMS reduction on all groups of users and items with few number of ratings. A more important advantage of DeepCoNN is that higher reductions are gained for groups with fewer ratings. It shows that DeepCoNN can alleviate the sparsity problem and help on the cold start problem.

It can also be seen that there exists a relation between the effectiveness of DeepCoNN and the number of ratings for a user or item. For users or items with a lower number of ratings, DeepCoNN reduction in MSE is higher. It shows that review text can be valuable information especially when we have limited information on the users or items.

2.4 Related Works

There are two categories of studies related to our work: techniques that model users and/or items by exploiting the information in online review text, and deep learning techniques employed

for recommender systems. In this section, we give a short review of these two research areas and distinguish our work from the existing approaches.

The first studies that used online review text in rating prediction tasks were mostly focused on predicting ratings for an existing review (Baccianella et al., 2009; Wu and Ester, 2015), while in our paper, we predict the ratings from the history of review text written by a user to recommend desirable products to that user.

One of the pioneer works that explored using reviews to improve the rating prediction is presented in (Jakob et al., 2009). In (McAuley and Leskovec, 2013), the authors proposed Hidden Factors as Topics (HFT) to employ topic modeling techniques to discover latent aspects from either item or user reviews. This method achieves significant improvement compared to models which only use ratings or reviews. A similar approach is followed in (Bao et al., 2014) with the main difference that it models user's and items' reviews simultaneously. Ratings Meet Reviews (RMR) (Ling et al., 2014) also tries to harness the information of both ratings and reviews.

Overall, one limitation of the above studies is that their textual similarity is solely based on lexical similarity. The semantic meaning is of particular importance and has been ignored in these works. Additionally, reviews are represented by using *bag-of-words*, and words' order exists in reviews has not been preserved. At last, the approaches which employ topic modeling techniques suffer from a scalability problem and also cannot deal with new coming users and items.

Recently, several studies have been done to use neural network based models including deep learning techniques for recommendation tasks (Zheng et al., 2018; Zheng et al., 2017; Wang et al., 2017). Several works (Salakhutdinov et al., 2007; Wu et al., ; Li et al., 2015) model users and/or items from the rating matrix using neural networks like denoising auto-encoders or Restricted Boltzmann Machines (RBM). They are considered as collaborative based techniques because they just utilize the rating matrix and ignore review text unlike our approach.

In (Van den Oord et al., 2013) and (Wang and Wang, 2014), deep models of CNN and Deep Belief Network (DBN) are introduced to learn latent factors from music data for music recommendation. In both models, initially, they find user and item latent factors using matrix factorization techniques. Then, they train a deep model such that it can reconstruct these latent factors for the items from the music content. A similar approach is followed in (Wang et al., 2015) for movie recommendation by using a generalized Stacked Auto Encoder (SAE) model. In all these works (Van den Oord et al., 2013), (Wang and Wang, 2014), (Wang et al., 2015), an item's latent factors are learned from item's content and review text is ignored.

In (Elkahky et al., 2015; Zheng et al., 2018a), a multi-view deep model is built to learn the user and item latent factors in a joint manner and map them to a common space. The general architecture of the model seems to have some similarities to our proposed model, but it differs from ours in some aspects. Their model is a content-based recommender system and does not use review text. Moreover, their outputs are coupled with a cosine similarity objective function to produce latent factors with high similarity. In this way, user and item factors are

not learned explicitly in relation to the rating information, and there is no guarantee that the learned factors can help the recommendation task.

All the above NN based approaches differ from DeepCoNN because they ignore review text. To the best of our knowledge, the only work which has utilized deep learning techniques to use review text to improve recommendation is presented in (Almahairi et al., 2015). To use the information exists in reviews, they proposed a model consisting of a matrix factorization technique and a Recurrent Neural Network (RNN). The matrix factorization is responsible for learning the latent factors of users and items, and the RNN models the likelihood of a review using the item's latent factors. The RNN model is combined with the MF simply via a trade-off term as some sort of a regularization term to tame the curse of data sparsity. Due to the matrix factorization technique, handling new users and items is not trivial in this model unlike DeepCoNN that handles them easily. Their proposed algorithm does not model users and items explicitly in a joint manner from their reviews, and it just uses reviews to regularize their model. In addition, since item text is represented by using *bag-of-words*, semantic meaning existing in words has not been explored.

2.5 Conclusion

In comparison with state-of-the-art baselines, DeepCoNN achieved **8.5%** and **7.6%** improvements on datasets of *Yelp* and *Beer*, respectively. On *Amazon*, it outperformed all the baselines and gained **8.7%** improvement on average. Overall, **8.3%** improvement is attained by the proposed model on all three datasets.

Additionally, in the experiments by limiting modeling to just one of the users and items, we demonstrated that the two networks could not only separately learn user and item latent factors from review text but also cooperate with each other to boost the performance of rating prediction. Furthermore, we showed that word embedding could be helpful to capture semantic meaning of review text by comparing it with a variant of DeepCoNN which uses random or TF-IDF representations for reviews.

At last, we conducted experiments to investigate the impact of the number of reviews. Experimental results showed that for the users and items with few reviews or ratings, DeepCoNN obtains more reduction in MSE than MF. Especially, when only one review is available, DeepCoNN gains the greatest MSE reduction. Thus, it validates that DeepCoNN can effectively alleviate the sparsity problem.

CHAPTER 3

SPECTRAL COLLABORATIVE FILTERING

(This chapter was previously published as "Spectral Collaborative Filtering", in the 12th ACM Conference on Recommender Systems (RecSys'18) (Zheng et al., 2018b). DOI: `https://doi.org/10.1145/3240323.3240343`.)

3.1 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) (Koren et al., 2009) 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 (Koren et al., 2009). 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.

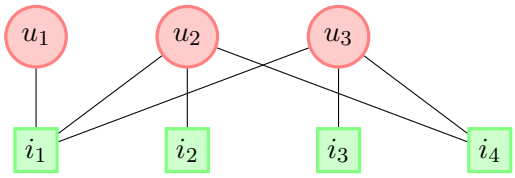

Figure 4: A toy example of a user-item bipartite graph $\mathcal{B}$ with edges representing observed user-item interactions. Red circles and green rectangles denote users and items, respectively.

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 4. A *cold-start* user u_1 only interacts with item i_1. Since u_1 shares i_1 with 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) (Koren et al., 2009), are usually designed to ap-

[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.

proximate 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 4. Whereas a number of memory-based approaches (Sarwar et al., 2001; Jamali and Ester, 2009) 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 (Shuman et al., 2013) studies connections between combinatorial properties of a graph and the eigenvalues of matrices associated to the graph, such as the laplacian matrix (see Definition 3.2.4 in Section 3.2). 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 4. 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 (Spielman, 2007). Figure 5 shows that, compared with i_2 and i_3, i_4 becomes closer to u_1 in the space[1]. Thus, when transformed into the frequency domain,

---

[1]In *spectral graph theory*, smaller (or larger) eigenvalues of the associated laplacian matrix corresponds to lower- (or higher-) frequency domains. In Figure 4, 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 $\boldsymbol{L}$.

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 (Kipf and Welling, 2016; Defferrard et al., 2016) 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 3.3.3) of directly learning from the *spectral domain* for recommendations, we first present a new spectral convolution operation (see Equation 3.10), 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.

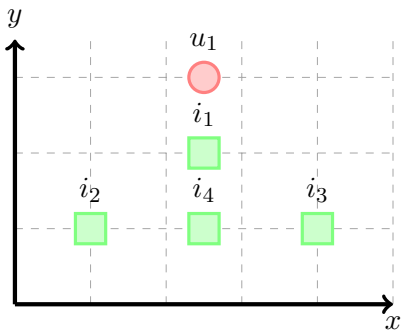

Figure 5: Vertices of the bipartite graph in Figure 4 are plotted in a frequency domain. Note that the vertices not shown above are omitted for simplicity.

- **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 3.2, we provide preliminary concepts. Section 3.3 describes SpectralCF in detail. Experiments are presented in Section 3.4 to analyze SpectralCF and demonstrate its effectiveness compared with state-of-the-art techniques for RS. In Section 3.5, we give a short review of the works related to our study. Finally, conclusions are presented in Section 3.6.

3.2 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 3.2.1. *(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 3.2.2. *(Implicit Feedback Matrix). An implicit feedback matrix $\boldsymbol{R}$ is a $|\mathcal{U}| \times |\mathcal{I}|$ matrix defined as following:*

$$\boldsymbol{R}_{r,j} = \begin{cases} 1 & \text{if } (u_r, i_j) \text{ interaction is observed,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

Definition 3.2.3. *(Adjacent Matrix). For the bipartite graph $\mathcal{B}$, its corresponding adjacent matrix $\boldsymbol{A}$ can be defined as:*

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{0} & \boldsymbol{R} \\ \boldsymbol{R}^\intercal & \mathbf{0} \end{bmatrix}, \tag{3.2}$$

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

Definition 3.2.4. *(Laplacian Matrix). The random walk laplacian matrix $\boldsymbol{L}$ is defined as $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A}$, where $\boldsymbol{I}$ 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 3.2.5. *(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.*

3.3 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.

3.3.1 Graph Fourier Transform

Definition 3.3.1. *(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:

$$\hat{f}(\xi) = \int_{-\infty}^{+\infty} f(x) e^{-2\pi i \xi} dx, \tag{3.3}$$

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 $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A}$, 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:

$$\hat{x}(l) = \sum_{j=0}^{N-1} x(j)\mu_l(j) \quad \text{and} \quad x(j) = \sum_{l=0}^{N-1} \hat{x}(l)\mu_l(j), \tag{3.4}$$

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 Equation 3.4 in the matrix form as $\hat{\boldsymbol{x}} = \boldsymbol{U}^\intercal\boldsymbol{x}$ 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 :

$$\begin{bmatrix} \hat{\boldsymbol{x}}^u \\ \\ \hat{\boldsymbol{x}}^i \end{bmatrix} = \boldsymbol{U}^\intercal \begin{bmatrix} \boldsymbol{x}^u \\ \\ \boldsymbol{x}^i \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \boldsymbol{x}^u \\ \\ \boldsymbol{x}^i \end{bmatrix} = \boldsymbol{U} \begin{bmatrix} \hat{\boldsymbol{x}}^u \\ \\ \hat{\boldsymbol{x}}^i \end{bmatrix}. \tag{3.5}$$

3.3.2 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:

$$\begin{bmatrix} \boldsymbol{x}^u_{new} \\ \\ \boldsymbol{x}^i_{new} \end{bmatrix} = \boldsymbol{U} g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}) \begin{bmatrix} \hat{\boldsymbol{x}}^u \\ \\ \hat{\boldsymbol{x}}^i \end{bmatrix} = \boldsymbol{U} g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}) \boldsymbol{U}^\intercal \begin{bmatrix} \boldsymbol{x}^u \\ \\ \boldsymbol{x}^i \end{bmatrix}, \tag{3.6}$$

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 Equation 3.6, 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. (Equation 3.6) is used to perform an inverse *graph fourier transform*.

3.3.3 Polynomial Approximation

Recall that we proposed a convolution operation, as shown in Equation 3.6, 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.

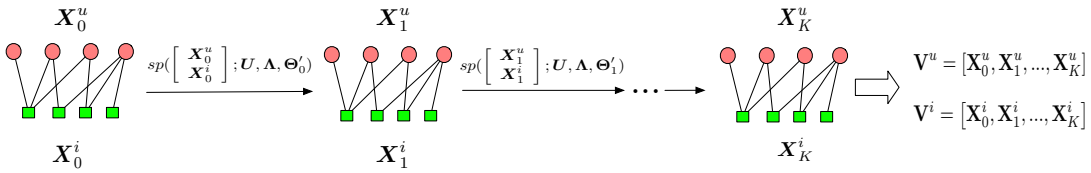

Figure 6: The feed-forward procedure of SpectralCF. The function $sp(:; \boldsymbol{U}, \boldsymbol{\Lambda}, \boldsymbol{\Theta})$ denotes the spectral convolution operation shown in Equation 3.10.

First, as shown in Equation 3.6, the learning complexity of the filter is $\mathcal{O}(N)$, 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* ($\boldsymbol{x}_{new}^u \in \mathcal{R}^{|\mathcal{U}|\times 1}$ and $\boldsymbol{x}_{new}^i \in \mathcal{R}^{|\mathcal{I}|\times 1}$) 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 filters $\mathcal{S}_g = \{g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}) = diag([\theta_0\lambda_0, \theta_1\lambda_1, ..., \theta_{N-1}\lambda_{N-1}]), \boldsymbol{\theta} \in \mathcal{R}^N\}$ is equal to the set of finite-order polynomials $\mathcal{S}_h = \{h_{\boldsymbol{\theta'}}(\boldsymbol{\Lambda}) = \sum_{p=0}^{N-1} \theta_p' \boldsymbol{\Lambda}^p, \boldsymbol{\theta'} \in \mathcal{R}^N\}$.

Proposition 3.3.1. *$\mathcal{S}_h$ is equal to $\mathcal{S}_g$.*

Proof. Let us consider an instance $h_{\boldsymbol{\theta'}}(\boldsymbol{\Lambda}) \in \mathcal{S}_h$. Then, $h_{\boldsymbol{\theta'}}(\boldsymbol{\Lambda}) = \sum_{p=0}^{N-1} \theta_p' \boldsymbol{\Lambda}^p = diag([\sum_{p=0}^{N-1} \theta_p' \lambda_0^{p-1} \cdot \lambda_0, \sum_{p=0}^{N-1} \theta_p' \lambda_1^{p-1} \cdot \lambda_1, ..., \sum_{p=0}^{N-1} \theta_p' \lambda_{N-1}^{p-1} \cdot \lambda_{N-1}])$. So, $h_{\boldsymbol{\theta'}}(\boldsymbol{\Lambda}) \in \mathcal{S}_g$. Now, consider a convolution filter $g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}) \in \mathcal{S}_g$. 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, $g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}) = \sum_{p=0}^{N-1} a_p \boldsymbol{\Lambda}^p = h_{\boldsymbol{a}}(\boldsymbol{\Lambda}) \in \mathcal{S}_h$. □

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

$$g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}) \approx \sum_{p=0}^{P} \theta'_p \boldsymbol{\Lambda}^p. \tag{3.7}$$

In this way, the learning complexity of the filter becomes $\mathcal{O}(P)$, 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 Equation 3.7 into Equation 3.6, we have:

$$\begin{bmatrix} \boldsymbol{x}^u_{new} \\ \\ \boldsymbol{x}^i_{new} \end{bmatrix} = (\theta'_0 \boldsymbol{U}\boldsymbol{U}^\intercal + \theta'_1 \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\intercal) \begin{bmatrix} \boldsymbol{x}^u \\ \\ \boldsymbol{x}^i \end{bmatrix}. \tag{3.8}$$

Furthermore, it is beneficial to further decrease the number of parameters by setting $\theta' = \theta'_0 = \theta'_1$. As a result, Equation 3.8 becomes:

$$\begin{bmatrix} \boldsymbol{x}^u_{new} \\ \\ \boldsymbol{x}^i_{new} \end{bmatrix} = \theta' (\boldsymbol{U}\boldsymbol{U}^\intercal + \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\intercal) \begin{bmatrix} \boldsymbol{x}^u \\ \\ \boldsymbol{x}^i \end{bmatrix}, \tag{3.9}$$

where θ' is a scalar.

For the second limitation, one can generalize the *graph signals* ($\boldsymbol{x}^u \in \mathcal{R}^{|\mathcal{U}|\times 1}$ and $\boldsymbol{x}^i \in \mathcal{R}^{|\mathcal{I}|\times 1}$) to C-dimensional *graph signals*: $\boldsymbol{X}^u \in \mathcal{R}^{|\mathcal{U}|\times C}$ and $\boldsymbol{X}^i \in \mathcal{R}^{|\mathcal{I}|\times C}$. Hence, Equation 3.9 becomes

$\begin{bmatrix} \boldsymbol{X}^u_{new} \\ \\ \boldsymbol{X}^i_{new} \end{bmatrix} = (\boldsymbol{U}\boldsymbol{U}^\intercal + \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\intercal) \begin{bmatrix} \boldsymbol{X}^u \\ \\ \boldsymbol{X}^i \end{bmatrix} \theta'$. To take one step further, we generalize the filter parameter θ' to a matrix of filter parameters $\boldsymbol{\Theta}' \in \mathcal{R}^{C \times F}$ with C input channels and F filters. As a result, our final spectral convolution operation is shown as the following:

$$\begin{bmatrix} \boldsymbol{X}^u_{new} \\ \\ \boldsymbol{X}^i_{new} \end{bmatrix} = \sigma \left((\boldsymbol{U}\boldsymbol{U}^\intercal + \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\intercal) \begin{bmatrix} \boldsymbol{X}^u \\ \\ \boldsymbol{X}^i \end{bmatrix} \boldsymbol{\Theta}' \right), \tag{3.10}$$

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

In fact, Equation 3.10 is a general version of Equation 3.9 as it is equivalent to perform Equation 3.9 in C input channels with F filters. Hereafter, the proposed convolution operation as shown in Equation 3.10 is denoted as a function $sp(:; \boldsymbol{U}, \boldsymbol{\Lambda}, \boldsymbol{\Theta}')$, which is parameterized by $\boldsymbol{U}, \boldsymbol{\Lambda}$ and $\boldsymbol{\Theta}'$.

3.3.4 Multi-layer Model

Given user vectors $\boldsymbol{X}^u$ and item vectors $\boldsymbol{X}^i$, new *graph singals* ($\boldsymbol{X}^u_{new}$ and $\boldsymbol{X}^i_{new}$) in Equation 3.10 are convolution results learned from the *spectral domain* with a parameter matrix $\boldsymbol{\Theta}' \in \mathcal{R}^{C \times F}$. As in classical CNNs, one can regard Equation 3.10 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 $\boldsymbol{X}_0^u$ and item vectors $\boldsymbol{X}_0^i$. Taking $\boldsymbol{X}_0^u$ and $\boldsymbol{X}_0^i$ as inputs, a K layered deep spectralCF can be formulated as:

$$\begin{bmatrix} \boldsymbol{X}_K^u \\ \boldsymbol{X}_K^i \end{bmatrix} = \underbrace{sp\Big(...sp(}_{K} \begin{bmatrix} \boldsymbol{X}_0^u \\ \boldsymbol{X}_0^i \end{bmatrix} ; \boldsymbol{U}, \boldsymbol{\Lambda}, \boldsymbol{\Theta}'_0)...; \boldsymbol{U}, \boldsymbol{\Lambda}, \boldsymbol{\Theta}'_{K-1}\Big), \tag{3.11}$$

where $\boldsymbol{\Theta}'_{K-1} \in \mathcal{R}^{F \times F}$ is a matrix of filter parameters for the k_{th} layer; $\boldsymbol{X}_k^u$ and $\boldsymbol{X}_k^i$ 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:

$$\mathbf{V}^u = [\mathbf{X}_0^u, \mathbf{X}_1^u, ..., \mathbf{X}_K^u] \quad \text{and} \quad \mathbf{V}^i = \left[\mathbf{X}_0^i, \mathbf{X}_1^i, ..., \mathbf{X}_K^i\right], \tag{3.12}$$

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 (Rendle et al., 2009) is employed. BPR is a pair-wise loss to address the implicit data for recommendations. Unlike point-wise based methods (Koren, 2008), 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 Equation 3.12, the loss function of SpectralCF is given as:

$$\begin{aligned}\mathcal{L} = \arg\min_{\boldsymbol{V}^u, \boldsymbol{V}^i} \sum_{(r,j,j') \in \mathcal{D}} & -\ln \sigma(\boldsymbol{v}_r^{u\intercal} \boldsymbol{v}_j^i - \boldsymbol{v}_r^{u\intercal} \boldsymbol{v}_{j'}^i) \\ & + \lambda_{reg}(||\boldsymbol{V}^u||_2^2 + ||\boldsymbol{V}^i||_2^2),\end{aligned} \tag{3.13}$$

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} stands for the weight on the regularization terms, and the training data $\mathcal{D}$ is formulated as:

$$\mathcal{D} = \{(r, j, j') | r \in \mathcal{U} \wedge j \in \mathcal{I}_i^+ \wedge j' \in \mathcal{I}_i^-\}. \tag{3.14}$$

3.3.5 Optimization and Prediction

At last, RMSprop (Tieleman and Hinton, 2012) 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 $\boldsymbol{\Theta}$, $\boldsymbol{X}_0^u$ and $\boldsymbol{X}_0^i$, we derive the user r's preference over item j as $\boldsymbol{v}_r^{u\intercal} \boldsymbol{v}_j^i$. The final item recommendation for a user r is given according to the ranking criterion as Equation 3.15.

$$r : j_1 \succcurlyeq j_2 \succcurlyeq ... \succcurlyeq j_n \Rightarrow \boldsymbol{v}_r^{u\intercal} \boldsymbol{v}_{j_1}^i > \boldsymbol{v}_r^{u\intercal} \boldsymbol{v}_{j_2}^i > ... > \boldsymbol{v}_r^{u\intercal} \boldsymbol{v}_{j_n}^i. \tag{3.15}$$

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: $\boldsymbol{\Psi} = \{\boldsymbol{\Theta}'_0, \boldsymbol{\Theta}'_1, ..., \boldsymbol{\Theta}'_{K-1}, \boldsymbol{X}_0^u, \boldsymbol{X}_0^i\}$.
Randomly initialize $\mathbf{X}_0^u$ and $\mathbf{X}_0^i$ 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 $\boldsymbol{X}_{k+1}^u$ and $\boldsymbol{X}_{k+1}^i$ by using Equation 3.10;
  end for
  Concatenate $[\boldsymbol{X}_0^u, \boldsymbol{X}_1^u, ..., \boldsymbol{X}_K^u]$ into $\boldsymbol{V}^u$ and $[\boldsymbol{X}_0^i, \boldsymbol{X}_1^i, ..., \boldsymbol{X}_K^i]$ into $\boldsymbol{V}^i$;
  Estimate gradients $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Psi}_e}$ by back propagation;
  Update $\boldsymbol{\Psi}_{e+1}$ according to the procedure of RMSprop optimization (Tieleman and Hinton, 2012);
end for
Return $\boldsymbol{\Psi}_E$.

3.4 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?

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

3.4.1 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** (Sarwar et al., 2001): ItemKNN is a standard neighbor-based collaborative filtering method. The model finds similar items for a user based on their similarities.
- **BPR** (Rendle et al., 2009): 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 (Gantner et al., 2011).
- **eALS** (He et al., 2016): 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.

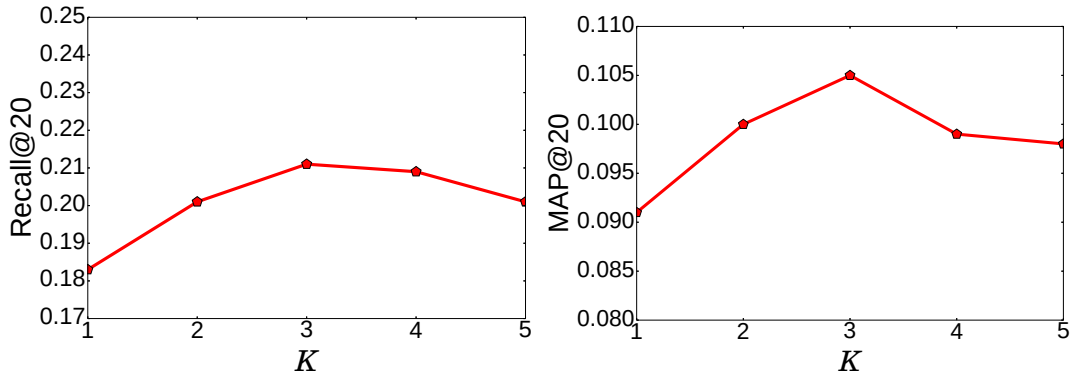

Figure 7: Effects of hyper-parameter K in terms of Recall@20 and MAP@20 in the dataset of *MovieLens-1M*.

- **NCF** (He et al., 2017b): **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** (Cai et al., 2008): **G**raph regularized **N**on-negative **M**atrix **F**acto-rization considers the graph structures by seeking a matrix factorization with a graph-based regularization.
- **GCMC** (Berg et al., 2017): **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 (Hu et al., 2008) to adapt them for implicit data.

TABLE V: 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

3.4.2 Datasets

We test our method as well as comparative models on three publicly available datasets[1]:

- ***MovieLens-1M*** (Harper and Konstan, 2016): 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 (He et al., 2017b) 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*** (Cantador et al., 2011): This dataset has been released by the Second International Workshop on Information Heterogeneity and Fusion in Recommender Systems[2]. 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.

---

[1] *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/`

[2] http://ir.ii.uam.es/hetrec2011/

- ***Amazon Instant Video*** (McAuley et al., 2015b): 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.

3.4.3 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:

$$\text{Recall@M} = \frac{\#\text{items the user likes among the top M}}{\text{total number of items the user likes}}. \tag{3.16}$$

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 3.4.1. 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 (He et al., 2017b) 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 7 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 V, where λ denotes the learning rate of RMSprop. Our models are implemented in *TensorFlow* (Abadi et al., 2016).

3.4.4 Experimental Results (RQ1 and RQ2)

In Figure 8, 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

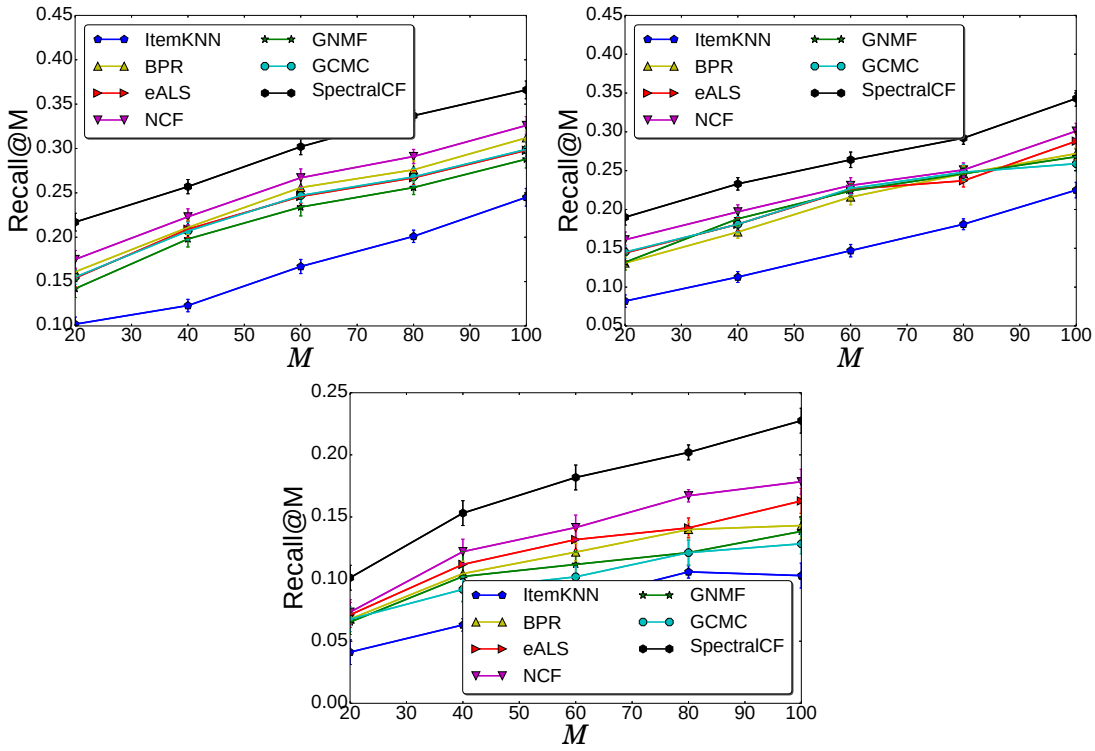

Figure 8: Performance comparison in the dataset of *MovieLens-1M*, *HetRec*, and *Amazon Instant Video*, respectively, in terms of recall@M with M varied from 20 to 100. Errors bars are 1-standard deviation.

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.

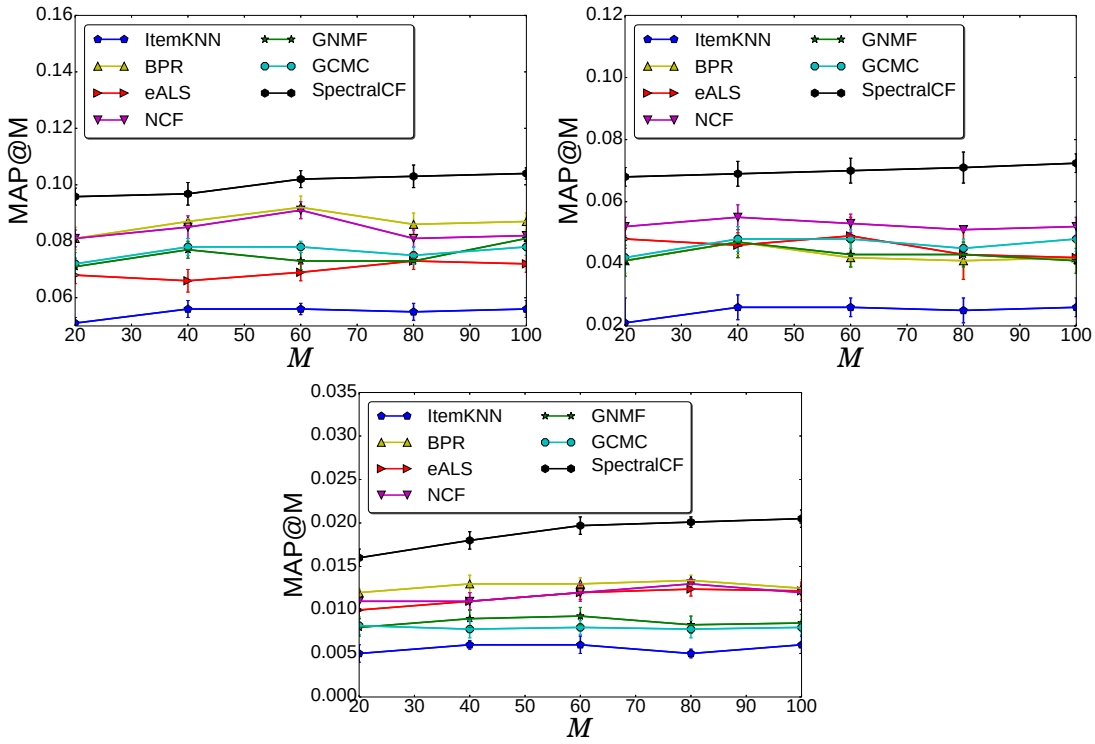

Figure 9: Performance comparison in the dataset of *MovieLens-1M*, *HetRec*, and *Amazon Instant Video*, respectively, in terms of MAP@M with M varied from 20 to 100. Errors bars are 1-standard deviation.

In Figure 9, 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 8 and Figure 9, 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 comparing SpectralCF with graph-based models, we show that SpectralCF can effectively learn from the *spectral domain*.

3.4.5 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 9. The test results are reported in the Table VI.

In Table VI, 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,

TABLE VI: 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)
	Improvement	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)
	Improvement	35.7%	41.2%	33.3%	29.2%	29.6%

respectively. Hence, it is demonstrated that compared with BPR, spectralCF can better handle *cold-start* users and provide more reliable recommendations.

3.5 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.

3.5.1 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 (Salakhutdinov et al.,

2007). 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 (Wang et al., 2017a), 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 (Salakhutdinov et al., 2007), (Zheng et al., 2016) proposed a CF Neural Autoregressive Distribution Estimator (CF-NADE) model for collaborative filtering tasks. CF-NADE shares parameters between different ratings. (He et al., 2017b) 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. (Wang and Wang, 2014) 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 (Elkahky et al., 2015) and (Wang et al., 2017), multi-view deep models are built to utilize item information from more than one domain. (Kim et al., 2016) integrates a CNN with PMF to analyze documents associated with items to predict users' future explicit ratings. (Zheng et al., 2017) leverage two parallel neural networks to jointly model latent factors of users and items. To incorporate visual signals into RS, (Wang et al., 2017b) 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. (Zhang et al., 2016) investigates how to leverage the multi-view information to improve the quality of recommender systems. (Cheng et al., 2016) jointly trains wide linear models and deep neural networks for video recommendations. (Wang et al., 2016) and (Zheng et al., 2017) utilize RNN to consider word orders and extract complex semantics for recommendations. (Wang et al., 2017) applies an attention mechanism on a sequence of models to adaptively capture the change of criteria of editors. (Zheng et al., 2018a) 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 (Zhang et al., 2017).

3.5.2 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. (Zhou et al., 2008) develops a semi-supervised learning model on graphs for document recommendation. The model combines multiple graphs in order to measure item similarities. In (Yuan et al., 2014), 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. (Guan et al., 2009) 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, (Berg et al., 2017) proposes a graph convolution network based model for recommendations. In (Berg et al., 2017), a graph auto-encoder learns the structural information of a graph for latent factors of users and items. (Cai et al., 2008) 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 (Rao et al., 2015). (Monti et al., 2017) 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. (Jamali and Lakshmanan, 2013) suggests a general latent factor model for entities in a graph. (Yu et al., 2013) introduces a recommendation model for implicit data by taking advantage of different item similarity semantics in the graph. (Shi et al., 2015) 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.

3.6 Conclusions

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.

CHAPTER 4

MODELING CO-EVOLVING PATTERNS FOR SEQUENTIAL RECOMMENDATION

(This chapter was previously published as "Gated Spectral Units: Modeling Co-evolving Patterns for Sequential Recommendation", in the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19) (Zheng et al., 2019). DOI: https://doi.org/10.1145/3331184.3331329.)

4.1 Introduction

What will a customer buy next? The importance of this question cannot be overstated for building effective recommender systems (RS). RS intersect multiple products and customers, where characteristics of users and perceptions of items not only shift over time but also influence each other. This complex temporal information raises unique challenges.

In order to build a predictive model for users' future purchases, we observe that a user's actions are correlated to not only his or her past activities but also other users' behaviors. Interests of users co-evolve over time and their preferences influence each other dynamically. For example, as shown in Figure 10, if user u_1 is related to user u_2, when u_1 purchases a pair of shoes at time t_3, u_2 may buy a pair of *socks* for u_1 at a later time, say t_5. Some time later (t_7), when u_2 shops a bottle of *wine*, it is reasonable to expect u_1 to be interested in a *bottle opener*. We term this phenomenon of evolving actions of users and their mutual influence over

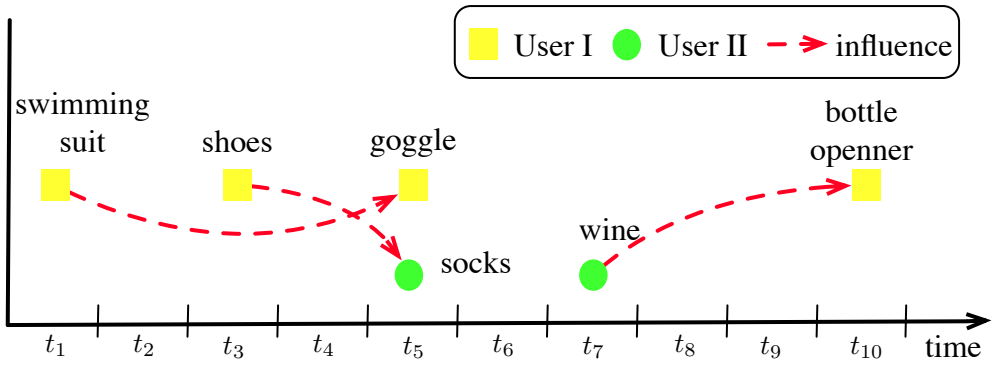

Figure 10: An example illustrates how activities of user co-evovles over time. Yellow and green circles denote purchases of user I and user II, respectively. (Best viewed in color)

time as *co-evovling patterns*. It is no doubt that effectively capturing such rich patterns can help reason over complex non-linear user-item interactions.

Existing approaches often lack the ability of learning the co-evovling knowledge, resulting in a limited understanding on behaviors of users and how they influence each other over time. Early works, such as TimeSVD++ (Koren, 2009), focus on modeling the shifting patterns of a user's preferences and the popularity of an item by introducing additional variables changing over time. Recent models first regard activities of a user as a sequence, and then propose Markov Chain (MC) based methods to capture the item dependencies or correlations within the sequence. For instance, in Figure 10, user u_1 shops a goggle at t_5 time because of the purchase of a *swimming suit* at the time of t_1. Another line of work adopts Recurrent Neural Networks (RNNs) to model the sequence. However, almost all of them fail to capture the rich *co-evovling patterns*.

In this paper, in order to utilize the *co-evovling patterns* and capture dependencies between actions across different users, we first formulate timestamped user-item interactions into Sequential Evolving Graphs (SEGs) (see Definition 4.2.1), where co-evolutionary knowledge can be revealed. Then, we generalize a spectral unit into a recurrent model by introducing gated mechanisms (Chung et al., 2014) to model the *co-evovling patterns* from spectral domains. The proposed model, Gated Spectral Units (GSUs), recurrently takes a sequence of graphs as input, and learns state vectors of users and items to summarize *co-evovling patterns* within the sequence. Our work makes the following contributions:

- **Novelty**: To our knowledge, it is the very first recommendation method to model sequential graphs from spectral domains.
- **Demonstrated Effectiveness**: It is demonstrated that the *co-evovling patterns* can be effectively captured from spectral domains of temporal graphs.
- **High Performance**: Benefiting from the *co-evovling* knowledge being effectively captured, GSU significantly outperforms state-of-the-art methods on three real-world datasets.

4.2 Background and Preliminaries

This paper focuses on the recommendation problem with implicit feedbacks (e.g. clicks, purchases, likes), where we only observe whether a person has interacted with an item and do not observe explicit ratings. Let us denote a set of users as $\mathcal{U}$ and an item set $\mathcal{I}$. $\mathcal{I}_i^+$ represents the set of all items liked by user i and $\mathcal{I}_i^-$ stands for the remaining items. Each user-item

TABLE VII: Notations

Notation	Description
$\mathcal{U}$, $\mathcal{I}$	user and item set
$\mathcal{I}_i^+$, $\mathcal{I}_i^-$	a set of items liked by user i, and all other items without interactions with user i
n_u, n_i	number of users and items
$\mathcal{G}_t = \{\mathcal{U}, \mathcal{I}, \mathcal{E}_t\}$	the t_{th} graph consisting of user set $\mathcal{U}$, item set $\mathcal{I}$ and edge set $\mathcal{E}_t$
$\mathbf{L}_t$	the laplacian matrix of the t_{th} graph
$\mathbf{U}_t^{(l)}$, $\mathbf{\Lambda}_t^{(l)}$	the top-l eigenvectors and eigenvalues of $\mathbf{L}_t$
$\mathbf{H}_t^u$, $\mathbf{H}_t^i$	hidden state matrices of users and items at timestamp t
$\mathbf{W}_z$, $\mathbf{W}_h$, $\mathbf{W}_r$	convolutional filters of update, candidate and reset gate
$\mathbf{b}_z$, $\mathbf{b}_h$, $\mathbf{b}_r$	bias vectors of update, candidate and reset gate
Δ	time interval

interaction is represented as a tuple $(i, j, \hat{t})$, denoting that user i has interacted with item j at timestamp $\hat{t}$. We define *Sequential Evolving Graphs (SEGs)* as:

Definition 4.2.1. *(Sequential Evolving Graphs). Sequential Evovling Graphs (SEGs) are represented as a sequence of bipartite graphs* $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_t, ...\}$. *The* t_{th} *bipartite graph* $\mathcal{G}_t$ *is defined as* $\mathcal{G}_t = \{\mathcal{U}, \mathcal{I}, \mathcal{E}_t\}$, *where* $\mathcal{U}$ *and* $\mathcal{I}$ *are the user set and item set, respectively and* $\mathcal{E}_t$ *denotes an edge set connecting users in* $\mathcal{U}$ *and items in* $\mathcal{I}$. *For an edge* $(i, j, \hat{t}) \in \mathcal{E}_t$, *it denotes user* i *has interacted with item* j *at timestamp* $\hat{t}$ *when* $\Delta(t-1) < \hat{t} \leq \Delta t$.[1]

Given SEGs of length T, we aim to predict edges (user-item interactions) to be formed in $\mathcal{G}_{T+1}$. Throughout the paper, we denote scalars by either lowercase or uppercase letters,

[1]In the case that $\mathcal{E}_t$ is empty, we remove $\mathcal{G}_t$ from $\mathcal{G}$.

vectors by boldfaced lowercase letters, and matrices by boldfaced uppercase letters. Important notations are summarized in Table IX.

4.3 Proposed Model

4.3.1 Spectral Convolution

Inspired by the recent success of graph convolution methods (Kipf and Welling, 2016), a recently proposed method (Zheng et al., 2018b), SpectralCF, extends the idea of spectral convolutions to the task of collaborative filtering. SpectralCF shows a great ability to capture preference patterns of users from the spectral domain of a user-item bipartite graph, and therefore achieves state-of-the-art performance.

Specifically, given a user-item bipartite graph $\mathcal{G}$ and its graph laplacian $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$, where $\mathbf{D}$ and $\mathbf{A}$ denote the degree matrix and adjacent matrix of $\mathcal{G}$, respectively, the spectral convolution operation is defined as: $\begin{bmatrix} \tilde{\mathbf{h}}^u \\ \\ \tilde{\mathbf{h}}^i \end{bmatrix} = \mathbf{U} f_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}) \mathbf{U}^T \begin{bmatrix} \mathbf{h}^u \\ \\ \mathbf{h}^i \end{bmatrix}$, where $\mathbf{U} \in \mathbb{R}^{(n_u+n_i)\times(n_u+n_i)}$ and $\boldsymbol{\Lambda} \in \mathbb{R}^{(n_u+n_i)\times(n_u+n_i)}$ are eigenvectors and eigenvalues of $\mathbf{L}$, respectively; $f_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})$ is a convolutional filtering function placed on eigenvalues; $\mathbf{h}^u \in \mathbb{R}^{(n_u\times 1)}$ and $\mathbf{h}^i \in \mathbb{R}^{(n_i\times 1)}$ respectively denote state vectors of users and items, and $\tilde{\mathbf{h}}^u \in \mathbb{R}^{(n_u\times 1)}$ and $\tilde{\mathbf{h}}^i \in \mathbb{R}^{(n_i\times 1)}$ represent new state vectors of users and items, respectively, learned from the spectral domain.

Nonetheless, the number of parameters in $f_\theta(\boldsymbol{\Lambda})$ is linear to the dimensionality of data, resulting in an unscalable model. To circumvent this issue, (Zheng et al., 2018b) utilizes a polynomial approximation to approximate $g_\Theta(\Lambda)$ as $g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}) \approx \sum_{p=0}^{P} \theta'_p \boldsymbol{\Lambda}^p$. As a result, the spectral

convolution is reformulated as $\begin{bmatrix} \tilde{\mathbf{H}}^u \\ \\ \tilde{\mathbf{H}}^i \end{bmatrix} = (\mathbf{U}\mathbf{U}^\intercal + \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\intercal) \begin{bmatrix} \mathbf{H}^u \\ \\ \mathbf{H}^i \end{bmatrix} \mathbf{\Theta}$, where $\mathbf{H}^u \in \mathbb{R}^{(n_u+n_i)\times F}$ and $\mathbf{H}^u \in \mathbb{R}^{(n_u+n_i)\times F}$ are F-dimensional row-vectors for users and items, respectively; and $\Theta \in \mathbb{R}^{(F\times F)}$ is a generalized convolutional filter with F channels and F filters.

However, as we aim to model *co-evovling patterns* from sequential graphs other than one static graph, computing the eigendecomposition of laplacians of multiple graphs would be prohibitively expensive. In order to adopt the aforementioned spectral convolution operation for modeling sequential graphs, we notice that, rather than the full eigen-decomposition, top-l eigenvectors and eigenvalues are sufficient to approximate $\mathbf{L}$ (Chen and Cai, 2011). Thus, we adopt ARPACK (Lehoucq et al., 1998), a most popular iterative eigensolver. Its complexity is $\mathcal{O}((n_u + n_i)l^2 + el)$, where e stands for the number of edges, and linear *w.r.t.* the graph size $(n_u + n_i)$. Due to the sparsity of our graphs, we have $e \ll n_u + n_i$. Given the top-l eigenvectors $\mathbf{U}_t^{(l)}$ and eigenvalues $\mathbf{\Lambda}_t^{(l)}$ of the t_{th} graph, we have:

$$\begin{bmatrix} \tilde{\mathbf{H}}_t^u \\ \\ \tilde{\mathbf{H}}_t^i \end{bmatrix} = (\mathbf{U}_t^{(l)}\mathbf{U}_t^{(l)\intercal} + \mathbf{U}_t^{(l)}\mathbf{\Lambda}_t^{(l)}\mathbf{U}_t^{(l)\intercal}) \begin{bmatrix} \mathbf{H}_{t-1}^u \\ \\ \mathbf{H}_{t-1}^i \end{bmatrix} \mathbf{\Theta}, \tag{4.1}$$

where $\mathbf{H}_{t-1}^u \in \mathbb{R}^{(n_u+n_i)\times F}$ and $\mathbf{H}_{t-1}^i \in \mathbb{R}^{(n_u+n_i)\times F}$ are state matrices of users and items from the previous time step $t-1$; $\tilde{\mathbf{H}}_t^u$ and $\tilde{\mathbf{H}}_t^i$ are learned by convolving $\begin{bmatrix} \mathbf{H}_{t-1}^u \\ \\ \mathbf{H}_{t-1}^i \end{bmatrix}$ on the current graph $\mathcal{G}_t$. As such, $\tilde{\mathbf{H}}_t^u$ and $\tilde{\mathbf{H}}_t^i$ captures the evolving patterns by integrating information from the previous step $(t-1)$ with newly formed connections of the current step t. Hereafter, we denote

the spectral convolution operation in Equation 4.1 as a function: $Conv(\begin{bmatrix} \mathbf{H}^u_{t-1} \\ \\ \mathbf{H}^i_{t-1} \end{bmatrix}, \mathcal{G}_t; \mathbf{\Theta})$, parameterized by a convolutional filter Θ.

4.3.2 Gated Spectral Units

Recall that our spectral convolution $Conv(\begin{bmatrix} \mathbf{H}^u_{t-1} \\ \\ \mathbf{H}^i_{t-1} \end{bmatrix}, \mathcal{G}_t; \mathbf{\Theta})$ is capable of capturing the patterns co-evolving from the previous graph to the current graph. It is an natural idea to introduce gated mechanisms (Chung et al., 2014) into our spectral convolution to capture co-evolving patterns from a sequence of graphs. Therefore, we present Gated Spectral Units (GSUs), which are capable of learning the *co-evovling patterns* from sequential graphs.

In GSUs, the update gate $\mathbf{Z}_t \in \mathbb{R}^{(n_u+n_i)\times F}$ convolves the historical state matrices on the current graph to decide how much the unit updates its state matrices. It is computed by:

$$\mathbf{Z}_t = \sigma(Conv(\begin{bmatrix} \mathbf{H}^u_{t-1} \\ \\ \mathbf{H}^i_{t-1} \end{bmatrix}, \mathcal{G}_t; \mathbf{W}_z) + \mathbf{b}_z), \tag{4.2}$$

where $\mathbf{W}_z \in \mathbb{R}^{(n_u+n_i)\times F}$, $\mathbf{b}_z \in \mathbb{R}^{(n_u+n_i)\times 1}$, and σ denotes the sigmoid function. A candidate gate generates a candidate state matrices by resetting the previous $\mathbf{H}^u_{t-1}$ and $\mathbf{H}^i_{t-1}$, and convolving them on $\mathcal{G}_t$ as:

$$\begin{bmatrix} \hat{\mathbf{H}}^u_t \\ \\ \hat{\mathbf{H}}^i_t \end{bmatrix} = \tanh(Conv(\mathbf{R}_t \odot \begin{bmatrix} \mathbf{H}^u_{t-1} \\ \\ \mathbf{H}^i_{t-1} \end{bmatrix}, \mathcal{G}_t; \mathbf{W}_h) + \mathbf{b}_h), \tag{4.3}$$

where $\mathbf{W}_h \in \mathbb{R}^{(n_u+n_i)\times F}$, $\mathbf{b}_h \in \mathbb{R}^{(n_u+n_i)\times 1}$, and the reset gate $\mathbf{R}_t \in \mathbb{R}^{(n_u+n_i)\times F}$ is similar to the update gate as below:

$$\mathbf{R}_t = \sigma(Conv(\mathbf{H}^u_{t-1}, \mathbf{H}^i_{t-1}, \mathcal{G}_t; \mathbf{W}_r) + \mathbf{b}_r), \tag{4.4}$$

where $\mathbf{W}_r \in \mathbb{R}^{(n_u+n_i)\times F}$ and $\mathbf{b}_r \in \mathbb{R}^{(n_u+n_i)\times 1}$. Finally, the output of GSUs at time t is a linear interpolation between the previous state matrices $\begin{bmatrix} \mathbf{H}^u_{t-1} \\ \mathbf{H}^i_{t-1} \end{bmatrix} \in \mathbb{R}^{(n_u+n_i)\times F}$ and the candidate $\begin{bmatrix} \hat{\mathbf{H}}^u_t \\ \hat{\mathbf{H}}^i_t \end{bmatrix} \in \mathbb{R}^{(n_u+n_i)\times F}$ as below:

$$\begin{bmatrix} \mathbf{H}^u_t \\ \mathbf{H}^i_t \end{bmatrix} = \mathbf{Z}_t \odot \begin{bmatrix} \mathbf{H}^u_{t-1} \\ \mathbf{H}^i_{t-1} \end{bmatrix} + (1 - \mathbf{Z}_t) \odot \begin{bmatrix} \hat{\mathbf{H}}^u_t \\ \hat{\mathbf{H}}^i_t \end{bmatrix}), \tag{4.5}$$

where $\odot$ denotes the element-wise multiplication.

Overall, GSUs take the current graph $\mathcal{G}_t$ and previous $\mathbf{H}^u_{t-1}$ and $\mathbf{H}^i_{t-1}$ as inputs, and output $\mathbf{H}^u_t$ and $\mathbf{H}^i_t$ for the current time step t. Thus, given the initial state matrices, $\mathbf{H}^u_0$ and $\mathbf{H}^i_0$, which are randomly initialized as trainable parameters, GSUs are able to recurrently process a sequence of graphs, and output state matrices of users and items of the last step, which summarize the *co-evovling patterns* within the sequence.

4.3.3 Optimization and Prediction

Given SEGs of length K generated from the training data, we randomly sample a batch of SEGs of length $T+1$ ($T+1 \ll K$) for training. For each SEGs of length $T+1$, we feed the first T graphs into GSUs to obtain $\mathbf{H}_T^u$ and $\mathbf{H}_T^i$. And, the score of an edge $(i,j) \in \mathcal{E}_{T+1}$ at step $T+1$ can be calculated as $\mathbf{H}_T^u(i,:)^\intercal \mathbf{H}_T^i(j,:)$, where $\mathbf{H}_T^u(i,:)$ and $\mathbf{H}_T^i(j,:)$ denote the i_{th} and j_{th} row of $\mathbf{H}_T^u$ and $\mathbf{H}_T^i$, respectively. We optimize the parameters of GSUs by minimizing the loss as:

$$\begin{aligned}\mathcal{L} = \quad & -\sum_{\substack{(i,j)\in\mathcal{E}_{T+1} \\ j'\in\mathcal{I}_i^-}} \ln \sigma(\mathbf{H}_T^u(i,:)^\intercal \mathbf{H}_T^i(j,:) - \mathbf{H}_T^u(i,:)^\intercal \mathbf{H}_T^i(j',:)) \\ & +\lambda(||\mathbf{H}_T^u||_F^2 + ||\mathbf{H}_T^i||_F^2), \end{aligned} \tag{4.6}$$

where λ is an regularization term. Equation 5.5 seeks to maximize the difference between the scores of an existing edge $(i,j) \in \mathcal{E}_{T+1}$ and a non-existing edge (i,j'), where j' is sampled from $\mathcal{I}_i^-$.

For evaluation, the last T graphs of SEGs of length K are taken into GSUs to attain $\mathbf{H}_T^u$ and $\mathbf{H}_T^i$. The final item recommendation for a user i is given by ranking the score $\mathbf{H}_T^u(i,:)^\intercal \mathbf{H}_T^i(j,:)$ in a descending order.

4.4 Experiments

In this section we conduct experiments to answer the following research questions:

- **RQ1:** Are the *co-evolving patterns* being effectively captured?
- **RQ2:** How do the *co-evovling patterns* work for handling the *cold-start* problem?

4.4.1 Datasets

In our experiments, we use three publicly available timestamped datasets: (1) **ML-1M**: (Harper and Konstan, 2016) MovieLens-1M contains $1,000,209$ ratings, $6,014$ users and $3,706$ movies; (2) **ADM** (McAuley et al., 2015b): Amazon Digital Music 5-core includes $4,731$ users, $2,420$ video games; (3) **AIV**: Amazon Instant Videos 5-core is collected by (McAuley et al., 2015b) and consists of $4,818$ users and $1,685$ items.

As in (He et al., 2017b), we transform datasets with explicit ratings into implicit data by regarding rating of 5 as positive feedback and all others as negative ones. For each dataset, we select the most recent item of each user for testing and the second most recent one for validation. All remaining items will be used for training. To create SEGs to capture *co-evovling patterns*, we set the time interval Δ as 1 day for *ML-1M* and *AIV*, and 7 for *ADM* to reduce the numder of graphs. As a result, we attain the SEGs of length (K) 977, 754, and $1,472$ for the dataset of *ML-1M*, *ADM*, and *AIV*, respectively.

4.4.2 Experimental Settings

Evaluation Protocols. We evaluate all models in two metrics: Hit Ratio@10 (HR@10) and NDCG@10. HR@10 measures the fraction of relevant items at top-10 recommendations out of all relevant items, while NDCG@10 evaluates their ranking performance. We follow a similar strategy as in (He et al., 2017b) to avoid heavy computation on evaluating all user-item pairs. For each user i, we randomly sample 999 negative items, and rank these items with the ground-truth item. Based on the rankings of these $1,000$ items, HR@10 and NDCG@10 can be evaluated.

Comparative Models. We compare GSUs with six state-of-the-art algorithms. They can be categorized into two groups: (1) Non-sequential Models: **BPR** (Rendle et al., 2009) and **SpectralCF** (Zheng et al., 2018b)[1]; (2) Sequential Models: **FPMC** (Rendle et al., 2010), **TransRec** (He et al., 2017a), **GRU4Rec** (Hidasi et al., 2015)[2] and **Caser** (Tang and Wang, 2018)[3]. The first group is added to validate the usefulness of sequential recommendation models, and the second group is for demonstrating the advantage of modeling *co-evovling patterns*.

Parameter Settings. For all methods, we search the latent dimensions from $\{8, 16, 32, 64\}$. The $\mathcal{L}_2$ regularization term is selected from $\{0.0001, 0.001, 0.01, 0.1\}$ for BPR, SpectralCF, FPMC, TransRec and GSUs. We tune all hyper-parameters using the validation set. For GRU4Rec and Caser, we use the parameter settings as suggested in the original papers. The *Adam* optimizer (Kingma and Ba, 2014) with the learning rate of 0.001 is adopted, and l in Equation 4.1 and T are empirically set to 6 and 10, respectively, for GSUs.

4.4.3 Performance Comparison (RQ1)

In this section we compare GSUs with six state-of-the art methods to answer **RQ1**. Table XI shows the performance comparison in terms of HR@10 and NDCG@10. Overall, GSUs improves the best comparative method by **27.9**% and **53.4**% in terms of HR@10 and NDCG@10, respectively, averaging on all three datasets. This experiment reveals two interesting observations:

---

[1] `https://github.com/lzheng21/SpectralCF`

[2] https://github.com/hidasib/GRU4Rec

[3] `https://github.com/graytowne/caser_pytorch`

TABLE VIII: Performance comparison in HR@10 and NDCG@10. The best and second best method are boldfaced and underlined, respectively. * and ** denote the statistical significance for $p < 0.05$ and $p < 0.01$, respectively, compared to the best competitor.

Data set	**Metric**	**BPR**	**Spectral CF**	**FPMC**	**Trans Rec**	**GRU-4Rec**	**Caser**	**GSUs**	***GSUs vs. best***
ML-1M	HR@10	0.061	0.081	0.092	0.099	0.102	<u>0.103</u>	**0.131****	27.2%
	NDCG@10	0.025	0.031	0.039	0.041	0.046	<u>0.045</u>	**0.061****	35.6%
ADM	HR@10	0.022	0.031	0.041	0.043	<u>0.051</u>	0.048	**0.065****	27.4%
	NDCG@10	0.011	0.018	0.019	0.021	<u>0.024</u>	0.022	**0.034***	41.7%
AIV	HR@10	0.072	0.088	0.096	0.010	0.111	<u>0.117</u>	**0.151***	29.1%
	NDCG@10	0.022	0.031	0.034	0.037	0.039	<u>0.041</u>	**0.075***	82.9%

- Non-sequential methods underperform sequential methods, indicating the benefits of modeling the short- and long- term dynamics in users' actions.
- Regardless of the data sets and the evaluation metrics, the proposed GSUs always achieve the best performance. This shows that by leveraging the power of *co-evovling patterns*, GSUs can better predict users' future actions.

4.4.4 Recommending for Cold-start Users (RQ2)

The *cold-start* problem is one of the most challenging issues for RS. It happens when a user interacted with very few number of items, causing a difficulty to understand the user's preferences. We are interested in if *co-evovling patterns* are helpful for alleviating the *cold-start* problem. As such, we conduct experiments under an extremely sparse setting, where we only use the first interaction of each user for training, and the second and third one for validation and test, respectively. All others are discarded. Consequently, we obtain SEGs of

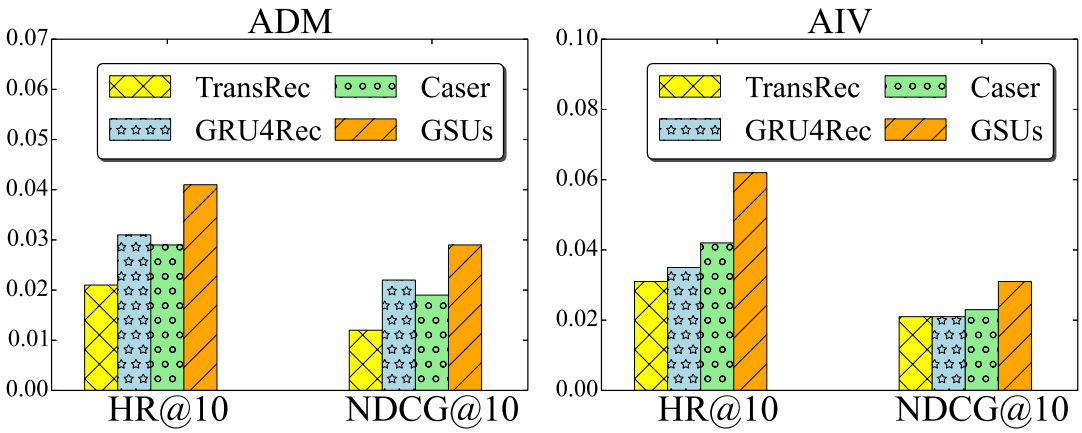

Figure 11: Performance comparison in HR@10 and NDCG@10 under a sparse setting, where each user is associated with only one user-item interaction for training.

length 458 and 1,158 for the datasets of *ADM* and *AIV*, respectively. Figure 14 illustrates the performance comparison under the sparse setting. In *ADM*, GSUs outperform the best comparative method, GRU4Rec, by **32.3**% and **31.8**% in HR@10 and NDCG@10, respectively. In *AIV*, GSUs beat the best performing competitor, Caser, by **47.6**% and **34.8**% in HR@10 and NDCG@10, respectively. It is validated that, benefiting from the ability of capturing *co-evolving patterns*, GSUs can better handle cold-start users than state-of-the-art comparative methods.

CHAPTER 5

DISTRIBUTION-BASED REPRESENTATIONS FOR TOP-N RECOMMENDATION

(This chapter was previously published as "Deep Distribution Network: Addressing the Data Sparsity Issue for Top-N Recommendation", in the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19) (Zheng et al., 2019). DOI: https://doi.org/10.1145/3331184.3331330.)

5.1 Introduction

The effectiveness of recommender systems (RS) often relies on how well users' interests or preferences can be understood and user-item interactions can be modeled. However, the data sparsity issue arises when users interacted with a limited number of items, hindering RS from understanding users' intentions. The problem is considered as one of major challenges for RS. Nonetheless, tackling the sparsity issue raises great challenges. Users' interests are diverse, and perceptions of items differ from user to user. This intricate information requires models of high-complexity while training such models needs a large amount of data, which contradicts to the reality of data scarcity.

Recent studies (Koren et al., 2009; He et al., 2017b) have suggested the importance of learning embeddings, or vectors, for users and items. Although embedding-based models have been proven useful in capturing typical interests of users and general concepts of items, most of

existing approaches learn fixed vectors to represent users and items. Arguably, users' behaviors are uncertain, and can be seen as stochastic events sampled from underlying distributions. When a user is modeled with a fixed vector, all actions of the user are considered to be certain and the uncertainty is hardly captured. Moreover, existing well-known Collaborative Filtering (CF) methods, such as matrix factorization (Koren et al., 2009), mostly use the popular dot-product as a metric, which violates the triangular inequality[1], to calculate user-item similarities. Nevertheless, according to (Hsieh et al., 2017; Tay et al., 2018), the triangular inequality is a prerequisite for fine-grained setting of users and items in a vector space.

Probabilistic distributions are classic and fundamental tools for tackling uncertainty and dealing with limited data. As users' actions are uncertain, we can consider them as observed stochastic events governed by underlying distributions of user interests. These distributions are able to describe how interests of users distribute in the space. As such, in order to power RS with the ability of combating the data sparsity issue with limited data, we propose Deep Distribution Network (DDN) to learn distributions for users and items. Specifically, we associate each user and item with a Gaussian distribution, whose mean and covariance matrix are estimated by deep neural networks, to characterize their interests and properties. Then, instead of calculating the popular dot-product, the Wasserstein distance is utilized to measure the difference between two Gaussian distributions, and the triangular inequality can therefore be satisfied. Finally, a

---

[1]The triangular inequality states that, given any three objects o_1, o_2, and o_3, the distance between any two objects, say $d(o_1, o_2)$, should satisfy the constraint $d(o_1, o_2) \leq d(o_1, o_3) + d(o_2, o_3)$)

pair-wise loss is proposed to minimize the Wasserstein distance of positive user-item pairs and maximize negative pairs.

Our work makes the following contributions:

- **Novelty:** To the best of our knowledge, it is the first work proposing to model users and items by Gaussian distributions via deep architectures for recommendation. We demonstrate that, distributions of users and items can be well modeled to alleviate the data sparsity issue.
- **A Wasserstein Loss:** We propose a Wasserstein loss for recommendation tasks. In the proposed loss, the crucial triangular inequality can be satisfied and therefore, leads to better performances, compared to conventional methods.
- **High Performance:** In the experiments, it is shown that DDN achieves state-of-the-art performances on three benchmark datasets. Specifically, compared to the best performing comparative method, DDN gains **42.4**% and **47.3**% improvements in Hit Ratio@10 and NDCG@10, respectively, averaging on all datasets.

5.2 Background and Preliminaries

In this section we present the background and preliminaries. We consider the most common scenario of RS with implicit feedbacks (e.g. clicks, purchases, likes). We follow the convention that the observed user-item interactions, such as clicks/purchase/likes, are treated as positive, and the non-observed ones are regarded as negative observations. Let us assume that a user u and an item i are associated with a feature vector $\mathbf{x}_u \in \mathcal{R}^{n_{(0)} \times 1}$ and $\mathbf{x}_i \in \mathcal{R}^{n_{(0)} \times 1}$, respectively (notation $n_{(0)}$ is described in Table IX). A user set and an item set are denoted as $\mathcal{U}$ and $\mathcal{I}$,

TABLE IX: Notations

Notation	Description
$\mathcal{U}$, $\mathcal{I}$	user and item set
$\mathcal{I}_u^+$, $\mathcal{I}_u^-$	a set of items liked by user u, and all the remaining items without interactions with u
$\mathbf{x}_u, \mathbf{x}_i$	feature vectors for user u and item i
$f_u(:;\mathbf{\Omega}^u)$,$f_i(:;\mathbf{\Omega}^i)$	two mean networks of user u and item i
$g_u(:;\mathbf{\Pi}^u)$, $g_i(:;\mathbf{\Pi}^i)$	two covariance networks of user u and item i
$\mathbf{W}_{\text{mean}}^{u,l}$,$\mathbf{W}_{\text{mean}}^{i,l}$	projection matrices of the l_{th} layer of the mean network of user u and item i
$\mathbf{W}_{\text{cov}}^{u,l}$,$\mathbf{W}_{\text{cov}}^{i,l}$	projection matrices of the l_{th} layer of the covariance network of user u and item i
$n_{(l)}$	number of neurons of the l_{th} layer

respectively. For a user $u \in \mathcal{U}$, let $\mathcal{I}_u^+$ denote a set of items liked by user u and $\mathcal{I}_u^-$ denote the remaining items. Important notations are summarized in Table IX.

5.3 Proposed Model

Instead of deriving vectors of users and items based on their interactions, we aim to learn Gaussian distributions to characterize interests of users and perceptions of items. To do so, as illustrated in Figure 12, we introduce a mean and a covariance network to learn these two parameters for the users' distribution. And, since users and items are two different types of entities, another two deep models will be built to estimate mean vectors and covariance matrices of items. Please bear in mind that, although these mean vectors and covariance matrices are also fixed after training, they together describe a probability density function. And, this function describes the sampling probability of each point in a space. This is a key point to distinguish DDN from existing embedding-based methods .

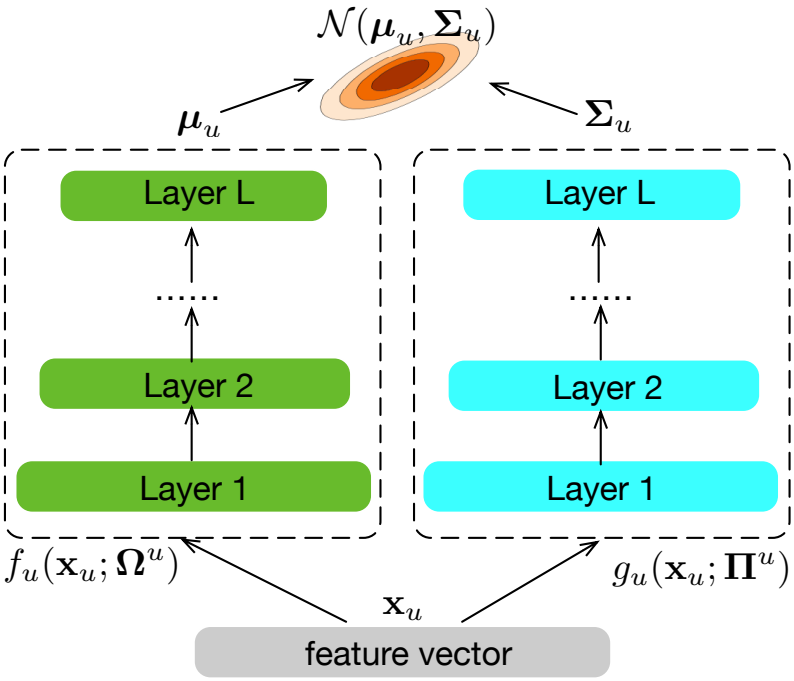

Figure 12: The mean and covariance networks of users. A feature vector $\mathbf{x}_u$ of user u is taken into $f_u(\mathbf{x}_u; \boldsymbol{\Omega}^u)$ and $g_u(\mathbf{x}_u; \boldsymbol{\Pi}^u)$ to learn the mean $\boldsymbol{\mu}_u$ and covariance $\boldsymbol{\Sigma}_u$, respectively.

5.3.1 Mean Networks

To learn a mean vector for user u, we build a mean network to take the user feature $\mathbf{x}_u \in \mathcal{R}^{n_{(0)} \times 1}$ into account, and output a mean vector $\boldsymbol{\mu}_u$ as:

$$\boldsymbol{\mu_u} = \underbrace{\text{elu}\Big(...\text{elu}\big(\mathbf{W}^{u,2}_{\text{mean}}(\text{elu}(}_{L}\ \mathbf{W}^{u,1}_{\text{mean}}\mathbf{x}_u + \mathbf{b}^{u,1}_{\text{mean}})) + \mathbf{b}^{u,2}_{\text{mean}}\big)...\Big), \tag{5.1}$$

where $\mathbf{W}^{u,l}_{\text{mean}} \in \mathcal{R}^{n_{(l-1)} \times n_{(l)}}$ and $\mathbf{b}^{u,l}_{\text{mean}} \in \mathcal{R}^{n_{(l)} \times 1}$ are projection matrix and bias vector of the l_{th} layer, respectively; elu is an activation function (Clevert et al., 2015). We denote

the mean network of users as $f_u(:;\mathbf{\Omega}^u)$, where $\mathbf{\Omega}^u = \{\mathbf{W}^{u,1}_{\text{mean}}, ..., \mathbf{W}^{u,L}_{\text{mean}}, \mathbf{b}^{u,1}_{\text{mean}}, ..., \mathbf{b}^{u,L}_{\text{mean}}\}$ is a parameter set. Likewise, another mean network, denoted as $f_i(:;\mathbf{\Omega}^i)$ parameterized by $\mathbf{\Omega}^i = \{\mathbf{W}^{i,1}_{\text{mean}}, ..., \mathbf{W}^{i,L}_{\text{mean}}, \mathbf{b}^{i,1}_{\text{mean}}, ..., \mathbf{b}^{i,L}_{\text{mean}}\}$, is utilized to derive mean vectors of items.

5.3.2 Covariance Networks

To learn covariance matrices of users, we establish a L-layer covariance network for estimating the covariance matrix of user i. The diagonal elements of $\mathbf{\Sigma}_u$ is computed as:

$$\boldsymbol{\sigma}_u = \underbrace{\text{elu}\Big(...\text{elu}\big(\mathbf{W}^{u,2}_{\text{cov}}(\text{elu}(}_{L}\ \mathbf{W}^{u,1}_{\text{cov}}\mathbf{x}_i + \mathbf{b}^{u,1}_{\text{cov}}) + \mathbf{1}\big) + \mathbf{b}^{u,2}_{\text{cov}}\big) + \mathbf{1}...\Big), \tag{5.2}$$

where $\mathbf{W}^{u,l}_{\text{cov}} \in \mathcal{R}^{n_{(l-1)} \times n_{(l)}}$ and $\mathbf{b}^{u,l}_{\text{cov}} \in \mathcal{R}^{n_{(l)} \times 1}$ are projection matrix and bias vector of the l_{th} layer, respectively; $\mathbf{1}$ denotes an vector of all ones. Finally, the covariance matrix of user u is given by:

$$\mathbf{\Sigma}_u = \text{diag}(\boldsymbol{\sigma}_u) + \mathbf{I}, \tag{5.3}$$

where $\mathbf{I} \in \mathcal{R}^{n_{(L)} \times n_{(L)}}$ is an identity matrix ensuring $\mathbf{\Sigma}_u$ to be positive semi-definite. The covariance network of users is denoted as $g_u(:;\mathbf{\Pi}^u)$, where $\mathbf{\Pi}^u = \{\mathbf{W}^{u,1}_{\text{cov}}, ..., \mathbf{W}^{u,L}_{\text{cov}}, \mathbf{b}^{u,1}_{\text{cov}}, ..., \mathbf{b}^{u,L}_{\text{cov}}\}$ is a parameter set. Analogously, another covariance network for items is denoted as $g_i(:;\mathbf{\Pi}^i)$, where $\mathbf{\Pi}^i = \{\mathbf{W}^{i,1}_{\text{cov}}, ..., \mathbf{W}^{i,L}_{\text{cov}}, \mathbf{b}^{i,1}_{\text{cov}}, ..., \mathbf{b}^{i,L}_{\text{cov}}\}$ includes all parameters of the network.

Our focus is to model the sparse user-item interaction data with the proposed distribution-based representations, we therefore avoid using additional information, such as user demographics or item textual descriptions, for feature vectors of users and items, even though these information is shown to be helpful for easing the sparsity issue. Instead, $\mathbf{x}_u$ and $\mathbf{x}_i$ are randomly

initialized, and then optimized during the training. Overall, given a user feature vector $\mathbf{x}_u$, we derive its mean vector and covariance matrix as $\boldsymbol{\mu}_u = f_u(\mathbf{x}_u; \boldsymbol{\Omega}^u)$ and $\boldsymbol{\Sigma}_u = g_u(\mathbf{x}_u; \boldsymbol{\Pi}^u)$, respectively. With an item feature vector $\mathbf{x}_i$, its mean vector and covariance matrix are calculated as $\boldsymbol{\mu}_i = f_i(\mathbf{x}_i; \boldsymbol{\Omega}^i)$ and $\boldsymbol{\Sigma}_i = g_i(\mathbf{x}_i; \boldsymbol{\Pi}^i)$, respectively.

5.3.3 A Wasserstein Loss

Recall that two Gaussian distributions of user u and item i, $\mathcal{N}(\boldsymbol{\mu}_u, \boldsymbol{\Sigma}_u)$ and $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, are estimated by mean and covariance networks. Instead of using the dot-product, one can utilize statistical distances to measure the distance between $\mathcal{N}(\boldsymbol{\mu}_u, \boldsymbol{\Sigma}_u)$ and $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. In this section we compare two popular distribution distances: Kullback-Leibler (KL) divergence and the p_{th} Wasserstein distance, and propose a Wasserstein based loss for recommendation.

The p_{th} Wasserstein distance (W_p) between two probability measures, $\mathbf{x}_1 \sim \mathbb{P}_1$ and $\mathbf{x}_1 \sim \mathbb{P}_2$, is defined as:

$$W_p(\mathbb{P}_1, \mathbb{P}_2) := \left(\inf_{\boldsymbol{\gamma} \in \Gamma(\mathbf{x}_1, \mathbf{x}_2)} \int d(\mathbf{x}_1, \mathbf{x}_2)^p d\boldsymbol{\gamma}(\mathbf{x}_1, \mathbf{x}_2) \right)^{1/p}, \tag{5.4}$$

where $\Gamma(\mathbb{P}_1, \mathbb{P}_2)$ denotes the joint distribution of $\mathbb{P}_1$ and $\mathbb{P}_2$, and $d(:,:)$ can be any distance, such as $\mathcal{L}_2$ distance. It is easy to verify that W_p distance satisfies the triangular inequality (Clement and Desch, 2008), while KL-divergence violates the inequality. As discussed in (Hsieh et al., 2017),(Tay et al., 2018), the satisfaction of the inequality benefits RS for reasoning over intricate user-item relationships, while the violation results in problematic representations of users and items. Moreover, if two distributions are non-overlapping, the Wasserstein distance can still

TABLE X: Statistics of Datasets

Dataset	**#users**	**#items**	**density**
MovieLens-1M	6,014	3,706	1.0%
LastFM	1,892	17,632	0.28%
Amazon Video Games	22,996	10,672	0.049%

measure the distance between them, while KL-divergence fails and leads to vanishing gradients. Hence, a Wasserstein based loss is proposed as:

$$\begin{aligned}\mathcal{L} = \quad & - \sum_{(u,i,i')\in\mathcal{D}} \ln \sigma\{W_2(\mathcal{N}(\boldsymbol{\mu_u}, \boldsymbol{\Sigma_u}), \mathcal{N}(\boldsymbol{\mu}_{i'}, \boldsymbol{\Sigma}_{i'})) \\ & -W_2(\mathcal{N}(\boldsymbol{\mu_u}, \boldsymbol{\Sigma_u}), \mathcal{N}(\boldsymbol{\mu_i}, \boldsymbol{\Sigma}_i))\} + \lambda(||\boldsymbol{\Omega}^u||_2^2 + \\ & ||\boldsymbol{\Omega}^i||_2^2 + ||\boldsymbol{\Pi}^u||_2^2 + ||\boldsymbol{\Pi}^i||_2^2),\end{aligned} \tag{5.5}$$

where σ denotes a sigmoid function; the training data $\mathcal{D}$ is created by $\{(u, i, i')|u \in \mathcal{U} \wedge i \in \mathcal{I}_u^+ \wedge i' \in \mathcal{I}_u^-\}$; and λ represents the weight on the regularization terms. Fortunately, the W_2 distance between two Gaussian distributions has an analytical solution as $W_2(\mathcal{N}(\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1}), \mathcal{N}(\boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})) = ||\boldsymbol{\mu_1} - \boldsymbol{\mu_2}||_2^2 + Tr(\boldsymbol{\Sigma_1} + \boldsymbol{\Sigma_2} - 2 * (\boldsymbol{\Sigma_1}^{1/2}\boldsymbol{\Sigma_2}\boldsymbol{\Sigma_1}^{1/2})^{1/2})$. Equation 5.5 seeks to maximize the Wasserstein distance of a negative pair (u, i') and minimize the distance of a positive pair (u, i). For evaluation, the final recommendation list of items for a user u is given by ranking $W_2(\mathcal{N}(\boldsymbol{\mu_u}, \boldsymbol{\Sigma_u}), \mathcal{N}(\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}))$ in an ascending order.

5.4 Experiments

In this section we conduct experiments to anwser the following research questions:

TABLE XI: Performance comparison in HR@10 and NDCG@10. The best and second best method are boldfaced and underlined, respectively. * and ** denote the statistical significance for $p < 0.05$ and $p < 0.01$, respectively, compared to the best baseline.

Dataset	Metric	Item KNN	eALS	BPR	NCF	CML	DDN	*DDN vs. best*
MovieLens-1M	HR@10	0.038	0.049	0.061	0.081	<u>0.092</u>	**0.128***	39.1%
	NDCG@10	0.021	0.024	0.025	0.039	<u>0.041</u>	**0.058****	41.4%
LastFM	HR@10	0.063	0.101	0.121	<u>0.103</u>	0.101	**0.147***	42.7%
	NDCG@10	0.031	0.035	0.039	<u>0.052</u>	0.050	**0.076****	46.1%
Amazon Video Games	HR@10	0.032	0.041	0.046	0.052	<u>0.055</u>	**0.080****	45.4%
	NDCG@10	0.018	0.019	0.021	0.022	<u>0.034</u>	**0.042***	54.5%

RQ1: Does DDN outperform state-of-the-art methods?

RQ2: Are the distribution-based representations helpful for tackling the data sparsity issue?

RQ3: How does the proposed Wasserstein loss work?

RQ4: Can DDN handle cold-start users in an effective way?

5.4.1 Experimental Settings

Comparative Models. We compare DDN with five state-of-the-art methods: **ItemKNN** (Sarwar et al., 2001), **eALS** (He et al., 2016), **BPR** (Rendle et al., 2009), **NCF** (He et al., 2017b) and **CML** (Hsieh et al., 2017) [1]. Among them, only ItemKNN does not learn user vectors or item vector, while all others represent users and items with vectors. NCF optimizes vectors

[1] `https://github.com/changun/CollMetric`

of users and items via deep architectures, and CML proposes a metric obeying the triangular inequlity.

Datasets. We test all methods on three standard datasets: ***MovieLens-1M***, ***LastFM***, and ***Amazon Video Games*** (Lakkaraju et al., 2013). As in (He et al., 2017b), we transform datasets with explicit ratings into implicit data by regarding rating of 5 as positive feedbacks and all others as negative. For each dataset, we select the latest item of each user for testing and the second latest one for validation. All remaining items are for training. The statistics of datasets are shown in Table X.

Evaluation Protocols. We evaluate all models in two metrics: Hit Ratio@N (HR@N) and NDCG@N. We follow a common strategy as in (He et al., 2017b) to avoid heavy computation on evaluating all user-item pairs. For each user i, we randomly sample 999 negative items, and rank them with the single ground-truth item. Based on the rankings of these $1,000$ items, HR@N and NDCG@N can be evaluated.

Paramter Settings. For ItemKNN, we employ the cosine distance to measure item similarities. For eALS and BPR, we search the latent dimensions from $\{8, 16, 32, 64\}$ and $\mathcal{L}_2$ regularization term from $\{0.0001, 0.001, 0.01, 0.1\}$. The network shape of NCF is set as $(32, 16, 8)$, as suggested in the original paper (He et al., 2017b). Since we avoid using the item content, the $\mathcal{L}_f$ loss of CML is excluded for a fair comparison. λ_c is chosen from $\{0.001, 0.01, 0.1, 1, 10\}$. All hyper-parameters are tuned using the validation set. For DDN, the *Adam* optimizer (Kingma and Ba, 2014) with the learning rate of 0.001 is adopted.

5.4.2 Performance Comparison (RQ1 and RQ2)

To anwser RQ1 and RQ2, DDN is compared with five state-of-the-art models on three datasets with different densities. Table XI shows the performance comparison. Overall, benefiting from the proposed distribution-based representations and Wasserstein loss, DDN beats all comparative methods, and achieves **42.4**% and **47.3**% improvements over the best comparative model in HR@10 and NDCG@10, respectively, averaging on all three datasets. These experiments reveal a number of interesting discoveries: (1) CML yields the second best performances in *MovieLens-1M* and *Amazon Video Games*, demonstrating the importance of the satisfaction of the triangular inequality; (2) Owing to the capability of capturing non-linearities via deep models, NCF defeats other comparative methods in *LastFM*; (3) It is shown that DDN achieves more improvements in a sparser dataset than in a denser one. It is validated that, compared to comparative approaches, DDN can better diminish the negative impacts of the data sparsity issue.

5.4.3 Effectiveness of the Wasserstein Loss (RQ3)

In order to anwser RQ3, we conduct experiments to compare DDN with DDN-KL, which is a variant of DDN employing the KL-divergence to measure the distances between users and items. Figure 13 shows the performance comparison between DDN and DDN-KL in *MovieLens-1M*. Overall, when N is vared from 3 to 10, DDN consistently outperforms DDN-KL in HR@N and NDCG@N. Specifically, DDN improves DDN-KL by **21.0**% and **31.0**% in HR@N and

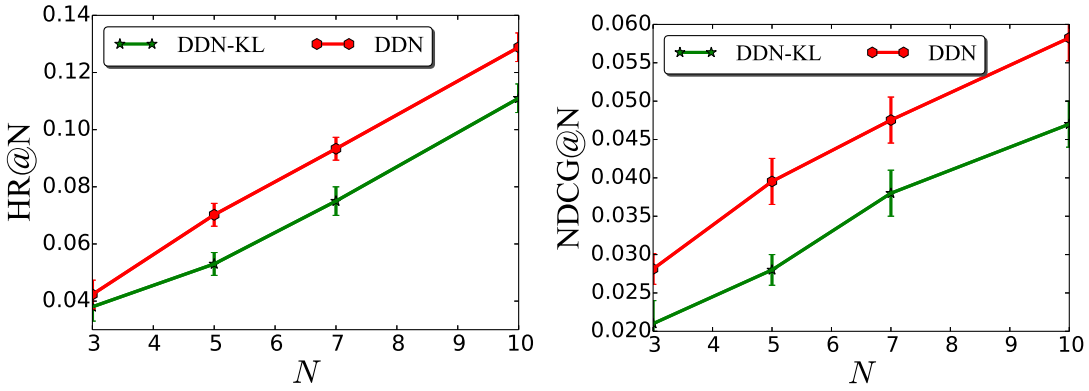

Figure 13: In *MovieLens-1M*, DDN is compared with DDN-KL in terms of HR@N and NDCG@N with N varied from 3 to 10. Errors bars are 1-standard deviation.

NDCG@N, respectively, averaging on N [1]. This experiment shows that, benefiting from the satisfaction of the triangular inequality, the proposed Wasserstein loss assists DDN with reasoning over complex user-item relations with limited data.

5.4.4 Recommending for Cold-start Users (RQ4)

The *cold-start* problem is one of the major challenges for RS. In this section we are curious if DDN can handle cold-start users in an effective way. Therefore, we compare DDN with two strong competitors, NCF and CML, in an extremely sparse setting, where each user is only associated with one item for training, one for validation and one for testing. Figure 14 shows that, suffering from the *cold-start* problem, the performances of NCF and CML inevitably degrade. However, DDN outperforms NCF and CML in terms of HR@10 and NCDG@10.

[1] Although similar results are observed in other two datasets, *LastFM* and *Amazon Video Games*, we omit the results due to limited space.

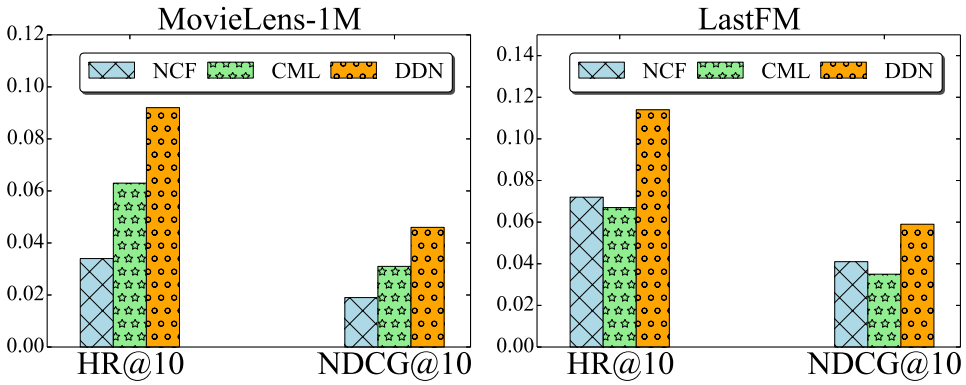

Figure 14: Performance comparison in HR@10 and NDCG@10 under a sparse setting, where each user is associated with only one user-item interaction for training.

Specifically, in *MovieLens-1M*, DDN improves CML by **46.0%** and **48.4%**, in HR@10 and NCDG@10, respectively. In *LastFM*, DDN beats NCF by **58.3%** and **43.9%**, in HR@10 and NCDG@10, respectively. Hence, it is demonstrated that, compared with two best performing state-of-the-art baselines, DDN can better handle *cold-start* users.

CHAPTER 6

CONCLUSION

(Part of the chapter was previously published in (Zheng et al., 2017; Zheng et al., 2018b; Zheng et al., 2019; Zheng et al., 2019).)

In this dissertation, we have explored the problem of deep learning methods for recommender systems. Towards this direction, we thoroughly studied four different research problems: review-based deep recommender systems, spectral collaborative filtering, modeling co-evolving patterns for sequential recommendation and distribution-based representation for Top-N recommendation. We evaluate our proposed methods by conducting intensive experiments on a variety of real-world datasets. The main contributions of our works are summarized as follows:

- It is shown that reviews written by users can reveal some info on the customer buying and rating behavior, and also reviews written for items may contain info on their features and properties. In this paper, we presented Deep Cooperative Neural Networks (DeepCoNN) which exploits the information exists in the reviews for recommender systems. DeepCoNN consists of two deep neural networks coupled together by a shared common layer to model users and items from the reviews. It makes the user and item representations mapped into a common feature space. Similar to MF techniques, user and item latent factors can effectively interact with each other to predict the corresponding rating.

- We show 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. 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.

- Despite the promising results achieved by recent sequential methods, most of them fail to leverage the *co-evovling patterns*, and such patterns are affluent in users actions and beneficial for reasoning over intricate user-item relationships. In order to power RS with the ability to capture *co-evolving patterns*, we first formulate the dynamic user-item bipartite graph into *Sequential Evovling Graphs*. Then, in order to utilize co-evolutionary patterns from SEGs, we propose Gated Spectral Units (GSUs). GSUs incorporate gated mechanisms into a spectral convolution. In this way, GSUs are able to learn from sequential graphs and capture the *co-evovling patterns* from spectral domains. In experiments, we demonstrate the usefulness of leveraging *co-evolving patterns* by comparing GSUs with

six state-of-the-art comparative methods. Overall, averaging on all three datasets, GSUs achieve **27.9**% and **53.4**% improvements over the best performing competitor in terms of HR@10 and NDCG@10, respectively. Additionally, we evaluate GSUs and three comparative methods in an extremely sparse setting, where each user is associated with only one user-item interaction. In the sparse setting, GSUs show its superior ability for handling cold-start users.

- We present Deep Distribution Network (DDN) to model users and items with Gaussian distributions for Top-N recommendation. Compared to existing approaches learning fixed vectors of users and items, DDN addresses the uncertainty inherent from the data sparsity issue by distribution-based representations. In DDN, each user and item is associated with a Gaussian distribution, whose mean and covariance are estimated by deep neural networks. Experimentally, we show that, compared to fixed vectors, the proposed distribution-based representations can better ease the sparsity issue and handle cold-start users. Additionally, we propose a Wasserstein distance based loss satisfying the triangular inequality, which is crucial for the performances of RS. By comparing DDN with one of its variants, DDN-KL, it is demonstrated that the proposed Wasserstein loss leads to a better performance.

APPENDICES

.1 ACM Copyright Letter

"Authors can reuse any portion of their own work in a new work of their own (and no fee is expected) as long as a citation and DOI pointer to the Version of Record in the ACM Digital Library are included.

Contributing complete papers to any edited collection of reprints for which the author is not the editor, requires permission and usually a republication fee.

Authors can include partial or complete papers of their own (and no fee is expected) in a dissertation as long as citations and DOI pointers to the Versions of Record in the ACM Digital Library are included. Authors can use any portion of their own work in presentations and in the classroom (and no fee is expected)." [1]

---

[1] `http://authors.acm.org/main.html`

.2 IEEE Copyright Letter

Title: PER: A Probabilistic Attentional Model for Personalized Text Recommendations

Conference Proceedings: 2018 IEEE International Conference on Big Data (Big Data)

Author: Lei Zheng

Publisher: IEEE

Date: Dec. 2018

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

CITED LITERATURE

[Abadi et al. , 2016]Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: 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, 2016.

[Almahairi et al. , 2015]Almahairi, A., Kastner, K., Cho, K., and Courville, A.: Learning distributed representations from reviews for collaborative filtering. In Proceedings of the 9th ACM Conference on Recommender Systems, pages 147–154. ACM, 2015.

[Apté et al. , 1994]Apté, C., Damerau, F., and Weiss, S. M.: Towards language independent automated learning of text categorization models. In Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, pages 23–30. Springer-Verlag New York, Inc., 1994.

[Baccianella et al. , 2009]Baccianella, S., Esuli, A., and Sebastiani, F.: Multi-facet rating of product reviews. In Advances in Information Retrieval, pages 461–472. Springer, 2009.

[Bao et al. , 2014]Bao, Y., Fang, H., and Zhang, J.: Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In AAAI, pages 2–8. AAAI Press, 2014.

[Bengio, 2009]Bengio, Y.: Learning deep architectures for ai. Machine Learning, 2(1):1–127, 2009.

[Bengio and LeCun, 2007]Bengio, Y. and LeCun, Y.: Scaling learning algorithms towards ai. Large-scale kernel machines, 34:1–41, 2007.

[Bengio et al. , 2006]Bengio, Y., Schwenk, H., Senécal, J.-S., Morin, F., and Gauvain, J.-L.: Neural probabilistic language models. In Innovations in Machine Learning, pages 137–186. Springer, 2006.

[Berg et al. , 2017]Berg, R. v. d., Kipf, T. N., and Welling, M.: Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263, 2017.

[Blei et al. , 2003]Blei, D. M., Ng, A. Y., and Jordan, M. I.: Latent dirichlet allocation. the Journal of machine Learning research, 3:993–1022, 2003.

[Bottou, 2004]Bottou, L.: Stochastic learning. In Advanced lectures on machine learning, pages 146–168. Springer, 2004.

[Bottou, 2012]Bottou, L.: Stochastic gradient descent tricks. In Neural Networks: Tricks of the Trade, pages 421–436. Springer, 2012.

[Brody and Elhadad, 2010]Brody, S. and Elhadad, N.: An unsupervised aspect-sentiment model for online reviews. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 804–812. Association for Computational Linguistics, 2010.

[Cai et al. , 2008]Cai, D., He, X., Wu, X., and Han, J.: Non-negative matrix factorization on manifold. In Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on, pages 63–72. IEEE, 2008.

[Cantador et al. , 2011]Cantador, I., Brusilovsky, P., and Kuflik, T.: 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In Proceedings of the 5th ACM conference on Recommender systems, RecSys 2011, New York, NY, USA, 2011. ACM.

[Chauvin and Rumelhart, 1995]Chauvin, Y. and Rumelhart, D. E.: Backpropagation: theory, architectures, and applications. Psychology Press, 1995.

[Chen et al. , 2015]Chen, L., Chen, G., and Wang, F.: Recommender systems based on user reviews: the state of the art. User Modeling and User-Adapted Interaction, 25(2):99–154, 2015.

[Chen and Cai, 2011]Chen, X. and Cai, D.: Large scale spectral clustering with landmark-based representation. In AAAI, volume 5, page 14, 2011.

[Chen et al. ,]Chen, Y., Xu, L., Liu, K., Zeng, D., and Zhao, J.: Event extraction via dynamic multi-pooling convolutional neural networks.

[Cheng et al. , 2016]Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al.: Wide & deep learning for recommender systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pages 7–10. ACM, 2016.

[Chung et al. , 2014]Chung, J., Gulcehre, C., Cho, K., and Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, 2014.

[Clement and Desch, 2008]Clement, P. and Desch, W.: An elementary proof of the triangle inequality for the wasserstein metric. Proceedings of the American Mathematical Society, 136(1):333–339, 2008.

[Clevert et al. , 2015]Clevert, D.-A., Unterthiner, T., and Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289, 2015.

[Collobert and Weston, 2008]Collobert, R. and Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning, pages 160–167. ACM, 2008.

[Collobert et al. , 2011]Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P.: Natural language processing (almost) from scratch. The Journal of Machine Learning Research, 12:2493–2537, 2011.

[Defferrard et al. , 2016]Defferrard, M., Bresson, X., and Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In Advances in Neural Information Processing Systems, pages 3844–3852, 2016.

[Diao et al. , 2014]Diao, Q., Qiu, M., Wu, C., Smola, A. J., Jiang, J., and Wang, C.: Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In KDD, pages 193–202. ACM, 2014.

[dos Santos and Gatti, 2014]dos Santos, C. N. and Gatti, M.: Deep convolutional neural networks for sentiment analysis of short texts. In Proceedings of the 25th International Conference on Computational Linguistics (COLING), Dublin, Ireland, 2014.

[Elkahky et al. , 2015]Elkahky, A. M., Song, Y., and He, X.: 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, pages 278–288. International World Wide Web Conferences Steering Committee, 2015.

[Gamon, 2004]Gamon, M.: Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In Proceedings of the 20th international conference on Computational Linguistics, page 841. Association for Computational Linguistics, 2004.

[Gantner et al. , 2011]Gantner, Z., Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L.: MyMediaLite: A free recommender system library. In 5th ACM International Conference on Recommender Systems (RecSys 2011), 2011.

[Guan et al. , 2009]Guan, Z., Bu, J., Mei, Q., Chen, C., and Wang, C.: 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, pages 540–547. ACM, 2009.

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

[He et al. , 2017a]He, R., Kang, W.-C., and McAuley, J.: Translation-based recommendation. In Proceedings of the Eleventh ACM Conference on Recommender Systems, pages 161–169. ACM, 2017.

[He et al. , 2017b]He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.: Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017, pages 173–182, 2017.

[He et al. , 2016]He, X., Zhang, H., Kan, M.-Y., and Chua, T.-S.: 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, pages 549–558. ACM, 2016.

[Heise et al. , 1994]Heise, L. L., Pitanguy, J., and Germain, A.: Violence against women. the hidden health burden. 1994.

[Hidasi et al. , 2015]Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D.: Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939, 2015.

[Hinton et al. , 2006]Hinton, G., Osindero, S., and Teh, Y.-W.: A fast learning algorithm for deep belief nets. Neural computation, 18(7):1527–1554, 2006.

[Hinton and Salakhutdinov, 2006]Hinton, G. E. and Salakhutdinov, R. R.: Reducing the dimensionality of data with neural networks. Science, 313(5786):504–507, 2006.

[Hinton et al. , 2012]Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012.

[Hong and Davison, 2010]Hong, L. and Davison, B. D.: Empirical study of topic modeling in twitter. In Proceedings of the First Workshop on Social Media Analytics, pages 80–88. ACM, 2010.

[Hsieh et al. , 2017]Hsieh, C.-K., Yang, L., Cui, Y., Lin, T.-Y., Belongie, S., and Estrin, D.: Collaborative metric learning. In Proceedings of the 26th International Conference on World Wide Web, pages 193–201. International World Wide Web Conferences Steering Committee, 2017.

[Hu and Liu, 2004]Hu, M. and Liu, B.: Mining and summarizing customer reviews. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 168–177. ACM, 2004.

[Hu et al. , 2008]Hu, Y., Koren, Y., and Volinsky, C.: Collaborative filtering for implicit feedback datasets. In 2008 Eighth IEEE International Conference on Data Mining, pages 263–272. Ieee, 2008.

[Huang et al. , 2007]Huang, F. J., Boureau, Y.-L., LeCun, Y., et al.: Unsupervised learning of invariant feature hierarchies with applications to object recognition. In 2013 IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8. IEEE, 2007.

[Jakob et al. , 2009]Jakob, N., Weber, S. H., Müller, M. C., and Gurevych, I.: Beyond the stars: exploiting free-text user reviews to improve the accuracy of movie recommendations. In Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion, pages 57–64. ACM, 2009.

[Jamali and Ester, 2009]Jamali, M. and Ester, M.: 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, pages 397–406. ACM, 2009.

[Jamali and Lakshmanan, 2013]Jamali, M. and Lakshmanan, L.: Heteromf: recommendation in heterogeneous information networks using context dependent factor models. In Proceedings of the 22nd international conference on World Wide Web, pages 643–654. ACM, 2013.

[Jo and Oh, 2011]Jo, Y. and Oh, A. H.: Aspect and sentiment unification model for online review analysis. In Proceedings of the fourth ACM international conference on Web search and data mining, pages 815–824. ACM, 2011.

[Johnson and Zhang, 2015]Johnson, R. and Zhang, T.: Effective use of word order for text categorization with convolutional neural networks. In HLT-NAACL, pages 103–112. The Association for Computational Linguistics, 2015.

[Kalchbrenner et al. , 2014]Kalchbrenner, N., Grefenstette, E., and Blunsom, P.: A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188, 2014.

[Kim et al. , 2016]Kim, D., Park, C., Oh, J., Lee, S., and Yu, H.: Convolutional matrix factorization for document context-aware recommendation. In Proceedings of the 10th ACM Conference on Recommender Systems, pages 233–240. ACM, 2016.

[Kim, 2014]Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882, 2014.

[Kingma and Ba, 2014]Kingma, D. and Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

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

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

[Koren, 2009]Koren, Y.: Collaborative filtering with temporal dynamics. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 447–456. ACM, 2009.

[Koren et al. , 2009]Koren, Y., Bell, R., and Volinsky, C.: Matrix factorization techniques for recommender systems. Computer, (8):30–37, 2009.

[Krizhevsky et al. , 2012]Krizhevsky, A., Sutskever, I., and Hinton, G. E.: Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[Lakkaraju et al. , 2013]Lakkaraju, H., McAuley, J. J., and Leskovec, J.: What's in a name? understanding the interplay between titles, content, and communities in social media. ICWSM, 1(2):3, 2013.

[Lapedes and Farber, 1988]Lapedes, A. S. and Farber, R. M.: How neural nets work. In Neural information processing systems, pages 442–456, 1988.

[LeCun et al. , 1998]LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[Lee et al. , 2009]Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In Proceedings of the 26th Annual International Conference on Machine Learning, pages 609–616. ACM, 2009.

[Lehoucq et al. , 1998]Lehoucq, R. B., Sorensen, D. C., and Yang, C.: ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods, volume 6. Siam, 1998.

[Li et al. , 2015]Li, S., Kawale, J., and Fu, Y.: Deep collaborative filtering via marginalized denoising auto-encoder. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pages 811–820. ACM, 2015.

[Ling et al. , 2014]Ling, G., Lyu, M. R., and King, I.: Ratings meet reviews, a combined approach to recommend. In Proceedings of the 8th ACM Conference on Recommender systems, pages 105–112. ACM, 2014.

[McAuley and Leskovec, 2013]McAuley, J. and Leskovec, J.: Hidden factors and hidden topics: understanding rating dimensions with review text. In Proceedings of the 7th ACM conference on Recommender systems, pages 165–172. ACM, 2013.

[McAuley et al. , 2012]McAuley, J., Leskovec, J., and Jurafsky, D.: Learning attitudes and attributes from multi-aspect reviews. In Data Mining (ICDM), 2012 IEEE 12th International Conference on, pages 1020–1025. IEEE, 2012.

[McAuley et al. , 2015a]McAuley, J., Pandey, R., and Leskovec, J.: Inferring networks of substitutable and complementary products. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 785–794. ACM, 2015.

[McAuley et al. , 2015b]McAuley, J., Targett, C., Shi, Q., and Hengel, A. v. d.: Image-based recommendations on styles and substitutes. arXiv preprint arXiv:1506.04757, 2015.

[Melville et al. , 2002]Melville, P., Mooney, R. J., and Nagarajan, R.: Content-boosted collaborative filtering for improved recommendations. In AAAI/IAAI, pages 187–192, 2002.

[Meng et al. , 2015]Meng, F., Lu, Z., Wang, M., Li, H., Jiang, W., and Liu, Q.: Encoding source language with convolutional neural network for machine translation. arXiv preprint arXiv:1503.01838, 2015.

[Mikolov et al. , 2010]Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S.: Recurrent neural network based language model. In INTERSPEECH, pages 1045–1048, 2010.

[Mikolov et al. , 2013]Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J.: Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111–3119, 2013.

[Monti et al. , 2017]Monti, F., Bronstein, M., and Bresson, X.: Geometric matrix completion with recurrent multi-graph neural networks. In Advances in Neural Information Processing Systems, pages 3700–3710, 2017.

[Nair and Hinton, 2010]Nair, V. and Hinton, G. E.: Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), pages 807–814, 2010.

[Ngiam et al. , 2011]Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. Y.: Multimodal deep learning. In Proceedings of the 28th international conference on machine learning (ICML-11), pages 689–696, 2011.

[Pan et al. , 2008]Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., and Yang, Q.: One-class collaborative filtering. In Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on, pages 502–511. IEEE, 2008.

[Phan et al. , 2008]Phan, X.-H., Nguyen, L.-M., and Horiguchi, S.: Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In Proceedings of the 17th international conference on World Wide Web, pages 91–100. ACM, 2008.

[Popescu and Etzioni, 2007]Popescu, A.-M. and Etzioni, O.: Extracting product features and opinions from reviews. In Natural language processing and text mining, pages 9–28. Springer, 2007.

[Ram and Gray, 2012]Ram, P. and Gray, A. G.: Maximum inner-product search using cone trees. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 931–939. ACM, 2012.

[Rao et al. , 2015]Rao, N., Yu, H.-F., Ravikumar, P. K., and Dhillon, I. S.: Collaborative filtering with graph information: Consistency and scalable methods. In Advances in neural information processing systems, pages 2107–2115, 2015.

[Rendle, 2012]Rendle, S.: Factorization machines with libfm. ACM Transactions on Intelligent Systems and Technology (TIST), 3(3):57, 2012.

[Rendle et al. , 2009]Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence, pages 452–461. AUAI Press, 2009.

[Rendle et al. , 2010]Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L.: Factorizing personalized markov chains for next-basket recommendation. In Proceedings of the 19th international conference on World wide web, pages 811–820. ACM, 2010.

[Salakhutdinov and Mnih, 2007]Salakhutdinov, R. and Mnih, A.: Probabilistic matrix factorization. In NIPS, pages 1257–1264. Curran Associates, Inc., 2007.

[Salakhutdinov and Mnih, 2008]Salakhutdinov, R. and Mnih, A.: Bayesian probabilistic matrix factorization using markov chain monte carlo. In Proceedings of the 25th international conference on Machine learning, pages 880–887. ACM, 2008.

[Salakhutdinov et al. , 2007]Salakhutdinov, R., Mnih, A., and Hinton, G.: Restricted boltzmann machines for collaborative filtering. In Proceedings of the 24th international conference on Machine learning, pages 791–798. ACM, 2007.

[Santos et al. , 2015]Santos, C. N. d., Xiang, B., and Zhou, B.: Classifying relations by ranking with convolutional neural networks. arXiv preprint arXiv:1504.06580, 2015.

[Sarwar et al. , 2001]Sarwar, B., Karypis, G., Konstan, J., and Riedl, J.: Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web, pages 285–295. ACM, 2001.

[Schein et al. , 2002]Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M.: Methods and metrics for cold-start recommendations. In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, pages 253–260. ACM, 2002.

[Schistad Solberg and Solberg, 1996]Schistad Solberg, A. and Solberg, R.: A large-scale evaluation of features for automatic detection of oil spills in ers sar images. In Geoscience and Remote Sensing Symposium, 1996. IGARSS'96.'Remote Sensing for a Sustainable Future.', International, volume 3, pages 1484–1486. IEEE, 1996.

[Shi et al. , 2015]Shi, C., Zhang, Z., Luo, P., Yu, P. S., Yue, Y., and Wu, B.: Semantic path based personalized recommendation on weighted heterogeneous information networks. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pages 453–462. ACM, 2015.

[Shuman et al. , 2013]Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P.: 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):83–98, 2013.

[Socher et al. , 2012]Socher, R., Huval, B., Manning, C. D., and Ng, A. Y.: Semantic compositionality through recursive matrix-vector spaces. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1201–1211. Association for Computational Linguistics, 2012.

[Soucy and Mineau, 2005]Soucy, P. and Mineau, G. W.: Beyond tfidf weighting for text categorization in the vector space model. In IJCAI, volume 5, pages 1130–1135, 2005.

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

[Srebro et al. , 2003]Srebro, N., Jaakkola, T., et al.: Weighted low-rank approximations. In ICML, volume 3, pages 720–727, 2003.

[Srivastava et al. , 2014]Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[Srivastava and Salakhutdinov, 2012]Srivastava, N. and Salakhutdinov, R. R.: Multimodal learning with deep boltzmann machines. In Advances in neural information processing systems, pages 2222–2230, 2012.

[Su and Khoshgoftaar, 2009]Su, X. and Khoshgoftaar, T. M.: A survey of collaborative filtering techniques. Advances in artificial intelligence, 2009:4, 2009.

[Tang and Wang, 2018]Tang, J. and Wang, K.: Personalized top-n sequential recommendation via convolutional sequence embedding. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pages 565–573. ACM, 2018.

[Tay et al. , 2018]Tay, Y., Anh Tuan, L., and Hui, S. C.: Latent relational metric learning via memory-based attention for collaborative ranking. In Proceedings of the 2018 World Wide Web Conference on World Wide Web, pages 729–739. International World Wide Web Conferences Steering Committee, 2018.

[Theano Development Team, 2016]Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints, abs/1605.02688, May 2016.

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

[Titov and McDonald, 2008]Titov, I. and McDonald, R.: Modeling online reviews with multi-grain topic models. In Proceedings of the 17th international conference on World Wide Web, pages 111–120. ACM, 2008.

[Turian et al. , 2010]Turian, J., Ratinov, L., and Bengio, Y.: Word representations: a simple and general method for semi-supervised learning. In Proceedings of the 48th annual meeting of the association for computational linguistics, pages 384–394. Association for Computational Linguistics, 2010.

[Turian et al. , 2009]Turian, J., Ratinov, L., Bengio, Y., and Roth, D.: A preliminary evaluation of word representations for named-entity recognition. In NIPS Workshop on Grammar Induction, Representation of Language and Language Learning, pages 1–8, 2009.

[Van den Oord et al. , 2013]Van den Oord, A., Dieleman, S., and Schrauwen, B.: Deep content-based music recommendation. In Advances in Neural Information Processing Systems, pages 2643–2651, 2013.

[Wallach, 2006]Wallach, H. M.: Topic modeling: beyond bag-of-words. In Proceedings of the 23rd international conference on Machine learning, pages 977–984. ACM, 2006.

[Wang and Blei, 2011]Wang, C. and Blei, D. M.: Collaborative topic modeling for recommending scientific articles. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 448–456. ACM, 2011.

[Wang et al. , 2017]Wang, F., Qu, Y., Zheng, L., Lu, C.-T., and Philip, S. Y.: Deep and broad learning on content-aware poi recommendation. In 2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC), pages 369–378. IEEE, 2017.

[Wang et al. , 2015]Wang, H., Wang, N., and Yeung, D.-Y.: Collaborative deep learning for recommender systems. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1235–1244. ACM, 2015.

[Wang et al. , 2016]Wang, H., Xingjian, S., and Yeung, D.-Y.: Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks. In Advances in Neural Information Processing Systems, pages 415–423, 2016.

[Wang et al. , 2010]Wang, H., Lu, Y., and Zhai, C.: Latent aspect rating analysis on review text data: a rating regression approach. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 783–792. ACM, 2010.

[Wang et al. , 2017a]Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., Zhang, P., and Zhang, D.: 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, pages 515–524. ACM, 2017.

[Wang et al. , 2017b]Wang, S., Wang, Y., Tang, J., Shu, K., Ranganath, S., and Liu, H.: What your images reveal: Exploiting visual contents for point-of-interest recommendation. In Proceedings of the 26th International Conference on World Wide Web, pages 391–400. International World Wide Web Conferences Steering Committee, 2017.

[Wang and Wang, 2014]Wang, X. and Wang, Y.: Improving content-based and hybrid music recommendation using deep learning. In Proceedings of the ACM International Conference on Multimedia, pages 627–636. ACM, 2014.

[Wang et al. , 2017]Wang, X., Yu, L., Ren, K., Tao, G., Zhang, W., Yu, Y., and Wang, J.: 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, pages 2051–2059. ACM, 2017.

[Wu et al. ,]Wu, Y., DuBois, C., Zheng, A. X., and Ester, M.: Collaborative denoising auto-encoders for top-n recommender systems.

[Wu and Ester, 2015]Wu, Y. and Ester, M.: FLAME: A probabilistic model combining aspect based opinion mining and collaborative filtering. In WSDM, pages 199–208. ACM, 2015.

[Xu and Rudnicky, 2000]Xu, W. and Rudnicky, A.: Can artificial neural networks learn language models? In Sixth International Conference on Spoken Language Processing, 2000.

[Yang and Liu, 1999]Yang, Y. and Liu, X.: A re-examination of text categorization methods. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pages 42–49. ACM, 1999.

[Yih et al. , 2014]Yih, W.-t., He, X., and Meek, C.: Semantic parsing for single-relation question answering. In Proceedings of ACL, 2014.

[Yu et al. , 2013]Yu, X., Ren, X., Sun, Y., Sturt, B., Khandelwal, U., Gu, Q., Norick, B., and Han, J.: Recommendation in heterogeneous information networks with implicit user feedback. In Proceedings of the 7th ACM conference on Recommender systems, pages 347–350. ACM, 2013.

[Yuan et al. , 2014]Yuan, Q., Cong, G., and Sun, A.: 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, pages 659–668. ACM, 2014.

[Zeng et al. , 2014]Zeng, D., Liu, K., Lai, S., Zhou, G., Zhao, J., et al.: Relation classification via convolutional deep neural network. In COLING, pages 2335–2344, 2014.

[Zhang et al. , 2016]Zhang, F., Yuan, N. J., Lian, D., Xie, X., and Ma, W.-Y.: Collaborative knowledge base embedding for recommender systems. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 353–362. ACM, 2016.

[Zhang et al. , 2017]Zhang, S., Yao, L., and Sun, A.: Deep learning based recommender system: A survey and new perspectives. arXiv preprint arXiv:1707.07435, 2017.

[Zheng, 2016]Zheng, L.: A survey and critique of deep learning on recommender systems. 2016.

[Zheng et al. , 2017]Zheng, L., Cao, B., Noroozi, V., Philip, S. Y., and Ma, N.: Hierarchical collaborative embedding for context-aware recommendations. In 2017 IEEE International Conference on Big Data (Big Data), pages 867–876. IEEE, 2017.

[Zheng et al. , 2019]Zheng, L., Fan, Z., Lu, C.-T., Zhang, J., and Yu, P. S.: Gated spectral units: Modeling co-evolving patterns for sequential recommendation. In The 42st International ACM SIGIR Conference on Research & Development in Information Retrieval. ACM, 2019.

[Zheng and Han, 2013]Zheng, L. and Han, K.: Multi topic distribution model for topic discovery in twitter. In 2013 IEEE Seventh International Conference on Semantic Computing, pages 420–425. IEEE, 2013.

[Zheng et al. , 2019]Zheng, L., Li, C., Lu, C.-T., Zhang, J., and Yu, P. S.: Deep distribution network: Addressing the data sparsity issue for top-n recommendation. In The 42st International ACM SIGIR Conference on Research & Development in Information Retrieval. ACM, 2019.

[Zheng et al. , 2018a]Zheng, L., Lu, C.-T., He, L., Xie, S., Noroozi, V., Huang, H., and Yu, P. S.: Mars: Memory attention-aware recommender system. arXiv preprint arXiv:1805.07037, 2018.

[Zheng et al. , 2018b]Zheng, L., Lu, C.-T., Jiang, F., Zhang, J., and Yu, P. S.: Spectral collaborative filtering. In Proceedings of the 12th ACM Conference on Recommender Systems, pages 311–319. ACM, 2018.

[Zheng et al. , 2017]Zheng, L., Noroozi, V., and Yu, P. S.: 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, pages 425–434. ACM, 2017.

[Zheng et al. , 2018]Zheng, L., Wang, Y., He, L., Xie, S., Wang, F., and Philip, S. Y.: Per: A probabilistic attentional model for personalized text recommendations. In 2018 IEEE International Conference on Big Data (Big Data), pages 911–920. IEEE, 2018.

[Zheng et al. , 2017]Zheng, L., Zhang, J., Cao, B., Yu, P. S., and Ragin, A.: A novel ensemble approach on regionalized neural networks for brain disorder prediction. In Proceedings of the Symposium on Applied Computing, pages 817–823. ACM, 2017.

[Zheng et al. , 2016]Zheng, Y., Tang, B., Ding, W., and Zhou, H.: A neural autoregressive approach to collaborative filtering. arXiv preprint arXiv:1605.09477, 2016.

[Zhou et al. , 2008]Zhou, D., Zhu, S., Yu, K., Song, X., Tseng, B. L., Zha, H., and Giles, C. L.: Learning multiple graphs for document recommendations. In Proceedings of the 17th international conference on World Wide Web, pages 141–150. ACM, 2008.

VITA

Name: Lei Zheng

EDUCATION:

M.E. in Computer Science, Harbin Institute of Technology, 2013.

B.S. in Computer Science, Jilin University, 2010.

PUBLICATIONS:

- Lei Zheng, Chaozhuo Li, Chun-Ta Lu, Jiawei Zhang, and Philip S. Yu. *Deep Distribution Network: Addressing the Data Sparsity Issue for Top-N Recommendation.* In the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval **(SIGIR)**, 2019.
- Lei Zheng, Ziwei Fan, Chun-Ta Lu, Jiawei Zhang, and Philip S. Yu. *Gated Spectral Units: Modeling Co-evolving Patterns for Sequential Recommendation.* In the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval **(SIGIR)**, 2019.
- Lei Zheng, Yixue Wang, Lifang He, Sihong Xie, Fengjiao Wang, and Philip S. Yu. *PER: A Probabilistic Attentional Model for Personalized Text Recommendations* . In IEEE International Conference on Big Data **(IEEE BigData)**, 2018.
- Vahid Noroozi, Sara Bahaadini, Lei Zheng, Sihong Xie, Weixiang Shao, and Philip S. Yu. *Semi-supervised Deep Representation Learning for Multi-View Problems.* In IEEE International Conference on Big Data **(IEEE BigData)**, 2018.

- Chaozhuo Li, Senzhang Wang, Philip S. Yu, Lei Zheng, Xiaoming Zhang, Zhoujun Li, and Yanbo Liang. *Distribution Distance Minimization for Unsupervised User Identity Linkage*. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management **(CIKM)**, 2018

- Fei Jiang, Lei Zheng, Jin Xu, and Philip S. Yu. *FI-GRL: Fast Inductive Graph Representation Learning via Projection-Cost Preservation*. In IEEE International Conference on Data Mining **(ICDM)**, 2018

- Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S. Yu. *Spectral Collaborative Filtering*. In the 12th ACM Conference on Recommender Systems **(RecSys)**, 2018

- Lei Zheng, Chun-Ta Lu, Lifang He, Sihong Xie, Vahid Noroozi, He Huang, and Philip S Yu. *MARS: Memory attention-aware recommender system.* arXiv: 1805.07037, 2018.

- Xiaotian Han, Chuan Shi, Lei Zheng, Philip S. Yu, Jianxin Li, and Yuanfu Lu. *Representation Learning with Depth and Breadth for Recommendation Using Multi-view Data* In the Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data **APWeb-WAIM**, 2018

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

- Bokai Cao, Lei Zheng, Chenwei Zhang, Philip S. Yu, Andrea Piscitello, John Zulueta, Olu Ajilore, Kelly Ryan, Alex Leow. *DeepMood: Modeling Mobile Phone Typing Dynamics*

for Mood Detection. In the 23rd ACM SIGKDD Conference of Knowledge Discovery and Data Mining. **(KDD)**, 2017.

- Vahid Noroozi, Lei Zheng, Sara Bahaadini, Sihong Xie, Philip S. Yu. *SEVEN: Deep Semi-supervised Verification Networks.* In the 26th International Joint Conference on Artificial Intelligence. **(IJCAI)**, 2017.

- Lei Zheng, Vahid Noroozi, and Philip S. Yu. *Joint deep modeling of users and items using reviews for recommendation.* In the Tenth ACM International Conference on Web Search and Data Mining. **(WSDM)**, 2017.

- Lei Zheng, Jingyuan Zhang, Bokai Cao, Philip S. Yu, and Ann Ragin. *A novel ensemble approach on regionalized neural networks for brain disorder prediction.* In the 32nd ACM SIGAPP Symposium On Applied Computing. **(SAC)**, 2017.

- Fengjiao Wang, Yongzhi Qu, Lei Zheng, Chun-Ta Lu, Philip S. Yu. *Deep and broad learning on content-aware POI recommendation.* In the IEEE 3rd International Conference on Collaboration and Internet Computing. **(CIC)**, 2017.

- Lei Zheng and Kai Han. *Extracting Categorical Topics from Tweets Using Topic Model.* In the Ninth Asia Information Retrieval Societies Conference. **(AIRS)**, 2013.

- Lei Zheng and Kai Han. *Multi topic distribution model for topic discovery in twitter.* In the IEEE Seventh International Conference on Semantic Computing. **(ICSC)**, 2013.