Stacked Mixed-Order Graph Convolutional Networks for Collaborative Filtering

Hengrui Zhang* Julian McAuley†

Abstract

Graph-based recommendation algorithms treat user-item interactions as bipartite graphs, based on which low-dimensional vector representations of users and items seek to preserve the relationships among them. Previous methods usually capture users' preferences by directly learning first-order neighborhood patterns for each node, which limits their ability to exploit the similarity between two distant users/items as well as a user's preferences toward distant items. To address this potential weakness, in this paper, we propose *SMOG-CF* (Stacked Mixed-Order Graph Convolutional Networks for Collaborative Filtering), a GCN-based framework that can directly capture high-order connectivity among nodes. Instead of implicitly capturing high-order connectivity through embedding propagation, SMOG-CF facilitates 'path-level' information propagation between neighboring nodes at any order. The matrix form of our embedding propagation formulas yields a model that is easy to deploy and can be extended to a general framework by adopting various information construction and aggregation equations. Experiments on several datasets of varying scale demonstrate the efficacy of our model.

Keywords: Recommendation Systems, Network Embedding, Graph Convolutional Networks, High-order Connectivity.

1 Introduction

Recommender Systems play an important role in many services including advertising, e-commerce, and social media. Collaborative Filtering (CF) addresses recommendation problems by assuming that similar users typically share similar interests over items. Users and items can be embedded as low dimensional vectors according to historical user-item interaction data, such that the trained embeddings will be useful for predicting whether a user will interact with an item.

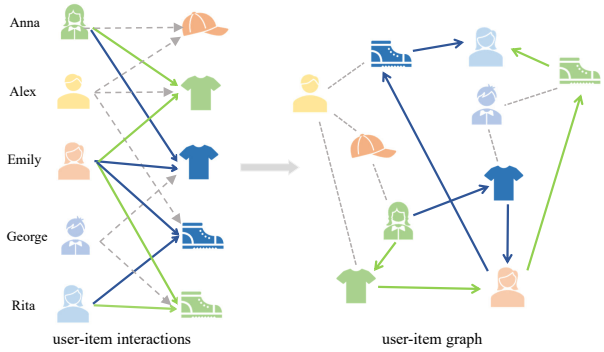

Figure 1: An example of high-order relationships in a user-item recommendation scenario. Left is the user-item interactions and right is the user-item graph. Blue and green lines are two 4-hop paths connecting Anna and Rita.

Following the development of Network Representation Learning, graph-based models have gained increasing attention. For recommendation, interactions can be modeled as a bipartite graph, where users and items are two isolated node sets and edges denote interactions between users and items. Graph-based models explore the local and global topological structure of the user-item graph combined with other attributes, and aim to learn efficient low-dimensional representations for each user and item. Graph Convolutional Networks (GCNs) [7] and their variants [4, 19–21] extend deep learning algorithms to graph-structured data by defining convolution operators on graphs, and have proven powerful when dealing with various downstream tasks [3, 13, 17, 22], including learning low-dimensional embeddings of users and items in a recommender system [19, 21, 26]. However, such models struggle to capture higher-order connectivity patterns among nodes, as they only aggregate information from directly neighboring nodes (or first-order neighbors), though it could be beneficial to take high-order connectivity into account [8, 15].

Fig. 1 illustrates the importance of high-order connections when inferring users' preferences toward items. Consider the following scenario, where customers are purchasing clothes. According to the user-item bipartite

*Shanghai Jiao Tong Univerisity, China; University of California, San Diego, the United States. (sqstardust@sjtu.edu.cn)

†University of California, San Diego, the United States. (jmcauley@eng.ucsd.edu)

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

graph presented, Anna and Emily share similar preferences, as they both bought blue and green shoes; similarly Rita shares preferences with Emily. Graph Convolutional Networks that exploit information from first-order neighbors can easily capture this type of information. However, consider the case where two users share no items in common. In Fig. 1, Anna and Rita have no common purchases but might still common share preferences. As highlighted in Fig. 1, we could find two paths between Anna and Rita: Anna → blue T-shirt → Emily → blue shoes → Rita (blue line), and Anna → green T-shirt → Emily → green shoes → Rita (green line). These two paths indicate that although Anna and Rita have no common items, there is a strong connection between these two users and they are likely to have similar preference to these items. This example suggests that multi-hop paths are vital for exploiting the high-order connectivity between two nodes in a graph and thus potentially helpful for inferring users' high-order preferences.

In this paper, we propose SMOG-CF, a framework for directly aggregating information from high-order neighbors of a target node in interaction graphs. We start with pair-wise message propagation from a p-order ($p > 1$) neighboring node to a target node, and demonstrate how to transform the pair-wise message propagation formula into a matrix form that can be deployed on large graphs, based on which SMOG-CF could be extended to other basic GCN models. Experiments on both large and small recommendation datasets with implicit feedback verify the effectiveness of our model.

Our contributions can be summarized as follows:

1. We propose SMOG-CF, a new CF model that seeks to learn the message propagation from a node's any-order neighbors in a user-item graph by defining a pair-wise message construction formula between two nodes that are not directly connected.

2. We present a derivation of the any-order embedding propagation formulas in matrix form, which makes our model straightforward and efficient to implement, based on which we could extend SMOG-CF to other GCN-based models.

3. We deploy SMOG-CF on four benchmark datasets. Our experiments verify the superiority of SMOG-CF over state-of-the-art techniques and the effectiveness of the proposed components.

2 Related Work

2.1 Network Representation Learning. Network representation learning techniques aim to learn low-dimensional latent representations of nodes in a network. Low-dimensional representations can effectively preserve local and global topological structures of a graph as well as node features, and can be used on downstream tasks, such as node classification and link prediction. Random-walk based models [2, 12, 18] were first proposed to learn such embeddings. Graph Neural Networks [4, 10, 20, 23], which try to adopt neural network methods on graph-structured data, have developed rapidly in recent years. Graph Convolutional Networks (GCNs) [7], which attempt to learn latent node representations by defining convolutional operation on graphs, were first proposed to solve semi-supervised classifications, and soon other GCN-based models were proposed for other tasks like social influence analysis [13], text classification [25] and recommender systems [19, 26]. Following previous studies [21, 24], in this paper we focus on learning user and item embeddings in a recommender (interaction) graph with implicit feedback.

2.2 High-order Proximity Learning on Graphs. Higher-order structure in graphs has proven beneficial in many cases such as hierarchical object representations, scene understanding, link prediction and recommender systems [1, 16, 21]. Network Motifs [15] were first proposed to learn such higher-order embeddings through Random-Walk based models, and were extended to Graph Neural Network structures by designing a convolutional layer with Motif attention that could aggregate first-order neighborhood information as well as high-order Motif information [8]. Based on GCNs, The high-order normalized Laplacian matrix is leveraged to aggregate information passed from any-order neighboring nodes [1, 9], though they fail to give a reasonable explanation as to why the high-order normalized Laplacian matrix should work in capturing connections across remote nodes. SMOG-CF directly captures the information passed from any of a node's p-order neighbors through a specific directed path, and demonstrates that the high-order normalized Laplacian matrix can be derived through message aggregation.

2.3 Graph-based Recommendation. As user-item interactions can be viewed as a bipartite graph, and a variety of studies have focused on using network embedding methods to learn user and item representations in recommender systems. [19] proposes a GCN-based auto-encoder framework for completing the user-item interaction matrix, though it only exploits first-order connections between nodes. PinSage [26] develops a data-efficient GCN structure which combines efficient random walks and graph convolutions to gener-

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

ate node embeddings. HOP-Rec [24] combines graph-based models with factorization models by performing random walks to enrich the interactions between high-order neighboring nodes and considering different orders of items simultaneously when decomposing the latent factors of user preferences. The recently proposed NGCF [21] stacks multiple Graph Convolution Layers to perform high-order embedding propagation and concatenates the output of each layer as the final representation of users and items, as an indirect way to leverage high-order connections, because this method cannot control the propagation strength between two nodes that are not directly connected. SMOG-CF is based on NGCF, though we devise a multi-hop information propagation path, based on which we can control the strength of message propagation between two high-order neighbors at any distance in an explicit way.

3 Preliminaries

3.1 Notations. In this paper, we focus on recommendation systems with implicit feedback, and we treat the user-item interaction as a bipartite graph. Following traditional recommendation models, we project each user or item to a low-dimensional embedding vector. Our task is to learn the best embedding vectors for each user and item and use these embeddings to make predictions. Table 1 introduces some of the most important symbols in this paper, while other notations will be introduced when mentioned.

Table 1: Notations

Symbol	Definition
$\mathcal{G}$	user-item bipartite graph
$\mathcal{U}$	user set
$\mathcal{V}$	item set
y_{ui}	interaction record of u and i
$\mathcal{N}_u$	user u's neighboring items
$\mathcal{N}_i$	item i's neighboring users
$\mathbf{A} = \{y_{ij}\}$	adjacency matrix of graph $\mathcal{G}$
$\mathbf{D} = \text{diag}(d_{ii})$	degree matrix of graph $\mathcal{G}$
$\mathbf{X} = \mathbf{E}^{(0)} = \{x_i\}$	initialized embedding matrix
$\mathbf{E}^{(l)} = \{e_i^{(l)}\}$	embedding matrix at layer l
$m_{i \leftarrow j}$	message propagated from j to i

3.2 General Graph Convolutional Networks. Here we introduce previous Graph Convolutional Network structures.

Graph Convolutional Networks try to define convolutional neural network structures on graph data, aiming to aggregate information from the neighborhoods of each node and propagate it to update the embedding of itself. This operation usually contains two parts: message construction and message aggregation. Message construction aims at calculating the information that each node receives from its neighboring nodes while message aggregation is aimed at delivering the integrated information as efficiently as possible so that the newly learned representation can better preserve topological information as well as the node's inherent attributes.

Graph Convolutional Network is an effective framework for representation learning on graphs, and many variants of GCNs have been proposed for different tasks. As introduced above, the representation of each node is updated using a pre-defined construction and aggregation operator to recursively aggregate the representation of its neighboring nodes. Formally, the l-th layer of a graph neural network can be generally defined as:

$$I_i^{(l)} = \text{AGGREGATE}\left(\left\{ m_{i \leftarrow j}^{(l)} : j \in \mathcal{N}_i \cup \{i\} \right\}\right), \tag{3.1}$$

where $m_{i \leftarrow j}^{(l)}$ is the message constructed that node j passed to node j at layer l, and can be formulated as:

$$m_{i \leftarrow j}^{(l)} = \text{CONSTRUCT}\left(e_j^{(l-1)}, e_i^{(l-1)}, c_{ij}\right), \tag{3.2}$$

where $e_i^{(l-1)}$ and $e_j^{(l-1)}$ represent the embeddings of node i and node j respectively. c_{ij} is a coefficient that reflects the connection strength between i and j. The choice of the function CONSTRUCT(*) and AGGREGATE(*) is very important in the design of Graph Convolutional Network models, which usually focus on designing efficient functions for these two operators.

4 Methodology

We first introduce the framework of SMOG-CF, which can capture high-order connection information in a direct way. Then, we extend our framework to other Graph Convolutional Network based recommender systems, and provide the embedding update formula in matrix form, so that it can be quickly adapted to other basic models. Finally, we present various techniques to improve the model performance.

4.1 Model Framework. Fig.4.1 provides an overview of SMOG-CF, which we describe in detail below.

4.1.1 Raw Input and Embedding Initialization. Our model requires a user-item bipartite interaction graph $\mathcal{G}$ as input. As the raw input of each user (resp. item) is a one-hot vector with a high-dimensional representation, each user and item are embedded into low-dimensional vectors x_u and x_i.

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

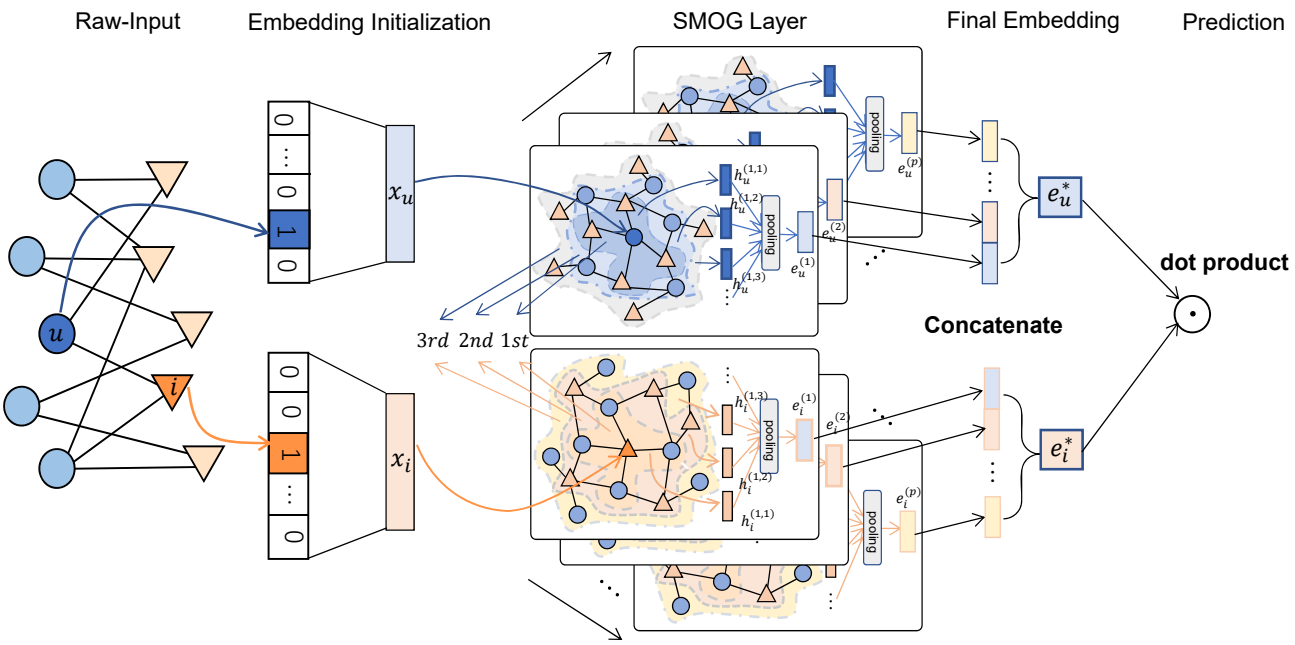

Figure 2: Framework of SMOG-CF.

4.1.2 Stacked Mixed-Order GCN layer. The Stacked Mixed-Order GCNs (SMOG) layer takes the initialized user/item embeddings as input and outputs the final embeddings that take advantage of the information propagated from the neighboring nodes at any order.

In order to better illustrate our method, we first introduce the **path** used for information propagation. Then we present the equations used to calculate the message propagated from a node to its neighboring nodes at any order. Finally, we sum up all the messages passed from a node's neighbors from any distance and present the embedding update formula for each user and item.

Message Construction. Given a node i, node $j \in \mathcal{N}_i$, for any node $k \in \mathcal{N}_j$ (including node i), we define k as the second-order neighbors of i. Similarly, we can define any p-order neighborhood connection: if there exists a connection path from node i to node j through p hops $i \to n_1 \to \cdots \to n_{p-1} \to j$, we call node j a p-order neighboring node of node j, and the path is a p-hop path connecting i and j.

For a p-hop path l_k form node i to node j, following the message construction formula presented by [21], we define the message constructed from j to i through this path as:

$$
\begin{aligned}
m_{ij,k}^{(p)} =& \text{CONSTRUCT}(e_i, e_j, c_{ij}^{(p)}) \\
=& c_{ij,k}^{(p)} \left(W_1^{(p)} e_j + W_2^{(p)} (e_i \odot e_j) \right),
\end{aligned} \tag{4.3}
$$

where W_j^p is the weight vector for distilling the p-order neighboring node j's information and e_i and e_j are the initialized vectors of node i and j respectively. $\odot$ denotes the element-wise product and $c_{ij,k}^{(p)}$ is the connection strength between node i and j through path l_k. Here $c_{ij,k}^{(p)}$ is set as:

$$
c_{ij,k}^{(p)} = \frac{1}{\sqrt{|\mathcal{N}_i|} \ldots |\mathcal{N}_{n_t}| \ldots \sqrt{|\mathcal{N}_j|}}. \tag{4.4}
$$

where $|\mathcal{N}_i|$ denotes the number of first-order neighboring nodes of i. This definition is similar to that in vanilla GCNs [7], but we consider all nodes on the path instead of the end nodes only, so that the connection strength will decay as the length of the path increases.

Message Aggregation. If we use $n_{i \to j}^{(p)}$ to represent the number of paths from node i to its p-order neighboring node j, and $I_{i \leftarrow j}^{(p)}$ to represent the sum of the messages passed from node j to i through all of the p-hop paths, we can conclude that:

$$
I_{i \leftarrow j}^{(p)} = \sum_{k=1}^{n_{i \to j}^{(p)}} m_{ij,k}^{(p)}, \tag{4.5}
$$

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Then we aggregate the messages propagated from all of node i's p-order neighboring nodes:

(4.6)
$$
\begin{aligned}
I_i^{(p)} &= \sum_{j \in \mathcal{N}_i^{(p)}} \sum_{k=1}^{n_{i\to j}^{(p)}} m_{ij,k}^{(p)} + m_{ii} \\
&= \sum_{j \in \mathcal{N}_i^{(p)} \cup \{i\}} \sum_{k=1}^{n_{i\to j}^{(p)}} c_{ij,k}^{(p)} \left(W_1^{(p)} e_j + W_2^{(p)} (e_i \odot e_j) \right) \\
&= \sum_{j=1}^{N} \sum_{k=1}^{n_{i\to j}^{(p)}} \frac{\hat{A}_{in_{k1}} \dots \hat{A}_{n_{kt}n_{k(t+1)}} \dots \hat{A}_{n_{kp}j}}{\sqrt{D_{ii}} \dots D_{n_{kt}n_{kt}} \dots \sqrt{D_{jj}}} \\
&\quad \left(W_1^{(p)} e_j + W_2^{(p)} e_i \odot e_j \right).
\end{aligned}
$$

where $N = |\mathcal{U}| + |\mathcal{V}|$ represents the number of nodes in the graph. In the derivation process, we use the equation $|\mathcal{N}_i| = D_{ii}$, where D is the degree matrix of the graph $\mathcal{G}$. Then we transform the formula above into a matrix form

(4.7)
$$
\begin{aligned}
I_i^{(p)} &= \sum_{j=1}^{N} (\mathbf{D}^{-1/2} \hat{\mathbf{A}} \mathbf{D}^{-1/2})_{i,j}^{p} \left(W_1^{(p)} e_j + W_2^{p} (e_i \odot e_j) \right) \\
&= \left[\left(\mathbf{D}^{-1/2} \hat{\mathbf{A}} \mathbf{D}^{-1/2} \right)^p \left(\mathbf{E} \mathbf{W}_1^{(p)} + (\mathbf{E} \odot \mathbf{E}) \mathbf{W}_2^{(p)} \right) \right]_i .
\end{aligned}
$$

where the subscript i, j represents the element at the i-th row and j-th column of the matrix, and the subscript i represents the vector at the i-th row. Now we can conclude the matrix form embedding formula by concatenating all nodes' embeddings as a matrix and adopt the ReLU activation:

(4.8)
$$
\begin{aligned}
\mathbf{H}^{(p)} &= \mathrm{ReLU} \left(\left[I_1^{(p)}, \dots I_t^{(p)} \dots I_1^{(p)} \right] \right) \\
&= \mathrm{ReLU} \left(\hat{\mathcal{L}}^p \left(\mathbf{E} \mathbf{W}_1^{(p)} + (\mathbf{E} \odot \mathbf{E}) \mathbf{W}_2^{(p)} \right) \right),
\end{aligned}
$$

where $\hat{\mathcal{L}} = \mathbf{D}^{-1/2} \hat{\mathbf{A}} \mathbf{D}^{-1/2}$ is the symmetric normalized Laplacian matrix of graph $\mathcal{G}$, $\mathbf{H}^{(p)}$ is the hidden embedding matrix that focuses on the p-order connections between nodes. If we take up to p-order connection into consideration, there would be p different hidden embedding matrixes representing different levels of information. In order to fuse this information together, we devise a pooling operation.

4.1.3 Embedding Pooling. We stack multiple SMOG layers to better exploiting the rich information in the graph. The initialized embedding matrix X could be regarded as the embedding of 0-th layer $E^{(0)}$, then we use the output of the l-the layer as the input of layer $l+1$. Given p hidden embedding matrices from layer l: $\mathbf{H}^{(l,1)}, \cdots, \mathbf{H}^{(l,p)}$, we use Average Pooling, which means using the average of the hidden embedding aggregated from different order nodes as the output embedding of layer l as well as the input of layer $l+1$:

(4.9)
$$
\begin{aligned}
\mathbf{E}^{(l+1)} &= \mathrm{MEAN}(\mathbf{H}^{(l,1)}, \cdots, \mathbf{H}^{(l,p)}) \\
&= \sum_{k=1}^{p} \mathbf{H}^{(l,k)} / p.
\end{aligned}
$$

Other pooling methods like Max Pooling [9], might also be used. However, in our model, we adopt average pooling because max pooling might focus on low-order hidden embeddings.

4.2 Model Prediction. Following [21], we concatenate the output of each layer in order to enhance the representation capability. For a user u, the representations from l layers are denoted as $\{e_u^{(1)}, \cdots, e_u^{(l)}\}$. As such, we concatenate them to constitute the final embedding for a user; we do the same operation on items, concatenating the item representations learned by different layers to get the final item embedding:

$$
e_u^* = \Big\|_{j=1,2,\cdots,l} e_u^{(j)}, \quad e_i^* = \Big\|_{j=1,2,\cdots,l} e_i^{(j)}. \tag{4.10}
$$

e_u^* and e_i^* are the final representation of user u and item i respectively. Finally, we take the inner product to estimate the preference of user u toward item i:

$$
\hat{y}(u,i) = {e_u^*}^T e_i^*. \tag{4.11}
$$

A higher $\hat{y}(u,i)$ indicates that user u is more likely to have an interaction with item i. Other more complex interaction functions such as multi-layer neural networks could also be leveraged to make predictions.

4.3 SMOG-CF as a Framework. As mentioned above, different GCN-based algorithms vary in their design of message construction and aggregation functions. As the proposed SMOG-CF does not depend on a specific basic model, we could extend it to a generalized framework by adopting other message construction and aggregation functions. Here we present the p-order hidden embedding formulas of theses variants in Table 2.

The 'vanilla' column of Table 2 represents the embedding propagation equation from layer l to layer $(l+1)$ while the p-order column represents the p-order hidden embedding $\mathbf{H}^{(l,p)}$ of layer l. Node embeddings of layer $(l+1)$ $\mathbf{E}^{(l)}$ can be calculated using Eq. 4.9. Derivations for other models follow similarly. Noting that the symmetric normalized Laplacian matrix plays

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Table 2: High-Order Embedding propagation formulas for different GCNs

Model	Vanilla formula: $\mathbf{E}^{(l+1)} =$	p-order formula: $\mathbf{H}^{(l+1,p)} =$
GCN [7]	$\text{ReLU}(\hat{\mathcal{L}}\mathbf{E}^{(l)}\mathbf{W}^{(l)})$	$\text{ReLU}(\hat{\mathcal{L}}^p\mathbf{E}^{(l)}\mathbf{W}^{(l,p)})$
GC-MC [19]	$\text{ReLU}\left(\text{ReLU}(\hat{\mathcal{L}}\mathbf{E}^{(l)}\mathbf{W}_1^{(l)})\mathbf{W}_2^{(l)}\right)$	$\text{ReLU}\left(\text{ReLU}(\hat{\mathcal{L}}^p\mathbf{E}^{(l)}\mathbf{W}_1^{(l,p)})\mathbf{W}_2^{(l,p)}\right)$
PinSage [26]	$\text{ReLU}\left(\text{CONCAT}\left(\text{ReLU}(\hat{\mathcal{L}}\mathbf{E}^{(l)}\mathbf{W}_1^{(l)}), \mathbf{E}^{(l)}\right)\mathbf{W}_2^{(l)}\right)$	$\text{ReLU}\left(\text{CONCAT}\left(\text{ReLU}(\hat{\mathcal{L}}^p\mathbf{E}^{(l)}\mathbf{W}_1^{(l,p)}), \mathbf{E}^{(l)}\right)\mathbf{W}_2^{(l,p)}\right)$
NGCF[1][21]	$\text{ReLU}\left(\hat{\mathcal{L}}\left(\mathbf{E}^{(l)}\mathbf{W}_1^{(l)} + (\mathbf{H}^{(l)} \odot \mathbf{H}^{(l)})\mathbf{W}_2^{l}\right)\right)$	$\text{ReLU}\left(\hat{\mathcal{L}}^p\left(\mathbf{E}^{(l)}\mathbf{W}_1^{(l,p)} + (\mathbf{E}^{(l)} \odot \mathbf{E}^{(l)})\mathbf{W}_2^{(l,p)}\right)\right)$

[1] NGCF is the basic model of SMOG-CF.

an important role in the propagation formula (in matrix form), the p-order connection can be easily acquired by $\hat{\mathcal{L}}$ multiplied by itself $p-1$ times, which can be computed efficiently.

4.4 Training Strategies Here we introduce the strategies adopted when training our model.

Loss Function. The loss function used in SMOG-CF is expressed as

$$\mathcal{L} = \mathcal{L}_1 + \lambda\mathcal{L}_2 \tag{4.12}$$

where $\mathcal{L}_1$ is the pair-wise BPR loss function widely used in binary classification problems, and $\mathcal{L}_2$ is the mini-batch aware regularization loss [28]. λ is the trade-off parameter between accuracy and complexity.

$\mathcal{L}_1$ and $\mathcal{L}_2$ can be defined as follows:

$$\mathcal{L}_1 = -\sum_{(u,i,j)\in\mathcal{O}} \log\sigma(\hat{y}_{ui} - \hat{y}_{uj}) \tag{4.13}$$

$$\mathcal{L}_2 = \sum_{(u,i,j)\in\mathcal{O}} \left[\|e_u\| + \|e_i\| + \|e_j\|\right]. \tag{4.14}$$

where $\mathcal{O} = \{(u,i,j)|(u,i) \in \mathcal{R}^+, (u,j) \in \mathcal{R}^-\}$ represents the pairwise training data, $\mathcal{R}^+$ denotes the positive samples while $\mathcal{R}^-$ denotes the negative samples.

Node Dropout. In our model we also adopt dropout to alleviate overfitting. Following prior work on GCNs, we propose to adopt node dropout, which is an extension of dropout in MLP. Specifically, we randomly block a particular node with probability p and discard its outgoing messages. In practice, we randomly drop $(|\mathcal{U}| + |\mathcal{V}|)p$ nodes of the Laplacian matrix.

5 Experiments

We conduct experiments on multiple real-world datasets to evaluate the proposed model. We aim to answer the following research questions:

- **RQ1** How does SMOG-CF perform compared with state-of-the-art collaborative filtering models?
- **RQ2** Is the higher-order connection learned directly in our model necessary for improving performance?
- **RQ3** How do different hyper-parameter settings affect model performance?

5.1 Experiment Setup

5.1.1 Data sets. We apply our model to four datasets: *Gowalla*, *Amazon-Books*, *MovieLens-latest*, and *MovieLens-1m*. All datasets are treated as implicit feedback (i.e., we only consider the presence of an interaction). We provide statistic of the datasets in Table 3. These datasets are summarized as follows:

Table 3: Dataset statistics

Datasets	#Users	#Items	#Clicks	Density
Gowalla	29,858	40,981	1,027,370	0.00084
Amazon-Books	18,343	30,001	805,885	0.00146
Movielens-1m	6,040	3,706	1,000,209	0.04468
MovieLens-latest	610	9,724	100,836	0.01700

Gowalla: This dataset is derived from gowalla[1], a location-based social network where users share their locations by checking-in. We use the cleaned version provided by [21], where only users and items with over 10 interactions are preserved.

Amazon-books: This is a subset of the Amazon-books dataset derived from the Amazon-Review dataset [11].

Movielens-latest & Movielens-1m [5]: These two datasets contain explicit movie rating data collected by GroupLens Research from the Movielens website[2]. The Moivelens-1m dataset is a classical recommendation dataset, while Moivelens-latest a regularly updated version. The version of the Movielens-latest dataset used in our experiments is from September 2018.

5.1.2 Implementation Details We use Tensorflow to implement our model and deploy it on an Nvidia

[1] www.gowalla.com

[2] https://grouplens.org/datasets/movielens/

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Table 4: Experimental Results

	Gowalla		Amazon-books		Movielens-latest		Movielens-1m	
Name	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
BPR	0.1291	0.1878	0.0261	0.0533	0.2128	0.3384	0.1972	0.3687
NeuMF [6]	0.1326	0.1985	0.0273	0.0551	0.2163	0.3443	0.2009	0.3703
GCN	0.1443	0.2125	0.0301	0.0596	0.2314	0.3607	0.2154	0.3897
GC-MC [19]	0.1480	0.2147	0.0317	0.0606	0.2325	0.3663	0.2187	0.3934
PinSage [26]	0.1491	0.2160	0.0320	0.0615	0.2338	0.3694	0.2195	0.3958
HOP-Rec [24]	0.1499	0.2188	0.0335	0.0642	0.2371	0.3771	0.2235	0.4018
NGCF [21]	0.1547	0.2237	0.0364	0.0672	0.2418	0.3822	0.2277	0.4082
SMOG-CF	**0.1574**	**0.2268**	**0.0383**	**0.0691**	**0.2453**	**0.3874**	**0.2301**	**0.4133**

TiTan X GPU with 11G memory. The batch size B is fixed to 64 for all datasets. The embedding dimension D is set to 64 for Gowalla and Amazon-Books, and 22 for Movielens-latest and Movielens-1m. The learning rate is set to 10^{-4} for Gowalla, $5 * 10^{-4}$ for Amazon-Books and $5 * 10^{-3}$ for Movielens-latest and Movielens-1m. The number of layers and hops is fixed at three if not specified, and the default dropout ratio is 0.2 for all datasets.

5.1.3 Baselines. We choose several comparative methods, including some state-of-the-art models for recommendation, to evaluate the performance of our proposed model. The optimal hyper-parameter settings for each method are determined either by our experiments or as suggested in the original papers.

- BPR [14]: This is a basic model-based recommendation method which uses both user-specific and item-based user embeddings to represent users' preferences.
- NeuMF [6]: This is one of the first models proposed to use deep neural networks to model the interactions between user embeddings and item embeddings. User embeddings and item embeddings are directly learned via back propagation.
- GC-MC [19]: This model considers recommendation as a matrix completion problem, and adopt GCN as a graph auto-encoder framework to generate the representations for users and items.
- PinSage [26]: This model is designed to employ GraphSAGE [4] on item-item graph of pictures on Pinterest[1]. We adopt this model on user-item graph in this paper.
- HOP-Rec [24]: This is a unified method that incorporates factorization with graph-based models. The high-order information is harvest from random surfing among neighborhood items for each user.
- NGCF [21]: This is a recent model that proposed to use Graph Convolution to capture user-item collaborative filtering signals. One contribution of NGCF is that it proposes to stack all layers of GCNs to take high-order user-item connections into consideration.

5.1.4 Evaluation Protocol. We adopt different metrics to evaluate recommendation performance: Recall@k and NDCG@k, two widely used metrics for top-k ranking. By default, we set $k = 20$. The detailed introduction of these metrics are presented in the supplementary material.

5.2 Comparative Results: RQ1. The experimental results of the proposed model and other comparative models are shown in Table 4. Results show that SMOG-CF outperforms other comparative models on these four datasets for top-k recommendation. This is because the proposed SMOG-CF can better exploit high-order connections in user-item graphs through multi-hop paths. Experiments also verify the effectiveness of the proposed model on both medium-scale and small-scale datasets.

5.3 Study of SMOG-CF: RQ2. In order to verify the effectiveness of various components in our model, we conduct an ablation study. As the order of neighbors (or the number of hops taken into consideration) plays an important role, we first study how the performance changes as a function of hop number. In Section 4, we showed that the SMOG-CF framework could be generalized to other GCN models by redefining the message reconstruction and aggregation function, so we extend our SMOG-CF framework to other GCN models

[1] www.pinterest.com

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

and would like to study their performance.

5.3.1 Effect of Hop Numbers. To investigate how the order of neighbors affects model performance, we change the depth of neighbors considered in the message propagation step. In particular, we consider hop numbers from 1 to 4. Table 5 summarizes experimental results, where SMOG-CF-p indicates the model considering up to p-order neighbors. In each experiment, we set the number of embedding propagation layers to 3. Note that SMOG-CF-1 is the same as NGCF, as we do not consider high-order connectivity information at each layer, but directly stack all layers together.

Table 5: Effect of hop number.

	Gowalla		Movielens-latest	
Name	Recall	NDCG	Recall	NDCG
SMOG-CF-1	0.1547	0.2237	0.2418	0.3822
SMOG-CF-2	0.1563	0.2256	0.2435	0.3845
SMOG-CF-3	0.1574	0.2268	0.2453	0.3874
SMOG-CF-4	**0.1580**	**0.2276**	**0.2462**	**0.3883**

We find that when we increase the number of hops, SMOG-CF achieves better performance on both datasets. This is because we can directly uncover the information propagated from high-order neighboring nodes at every layer. Another finding is that the marginal improvements decrease as the number of hops increases. This is due to: 1) the farther the distance between two nodes in a graph, the weaker the connection between them. 2) applying a too deep architecture will cause overfitting. Considering the balance between performance and model complexity, we set the hop number as 3 by default.

5.3.2 SMOG-CF as a framework: extension to other GCN models. Here, we apply SMOG-CF to other embedding propagation layers as variants of our models. In particular, we replace the message construction and aggregation functions used in SMOG-CF (the same as that of NGCF, termed SMOG-CF$_{\text{NGCF}}$) with that of GCNs (termed SMOG-CF$_{\text{GCN}}$), PinSage (SMOG-CF$_{\text{PinSage}}$) and GC-MC (SMOG-CF$_{\text{GC-MC}}$). In all experiments, the number of hops and layers is set to three. Results are shown in Table 6.

Experiments show that when extending our architecture to other GCN models, they achieve better performances on both medium and small datasets, compared with the original model performance presented in Table 4. This is because the proposed path-level information propagation for exploiting high-order connections works in different basic GCN models by redefining the information construction and aggregation equations. Also, SMOG-CF$_{\text{NGCF}}$ outperforms other variants as we leverage the direct user-item information interaction as is shown in Eq. 4.3.

Table 6: Performance of different embedding propagation techniques.

	Gowalla		Movielens-latest	
Name	Recall	NDCG	Recall	NDCG
SMOG-CF$_{\text{GCN}}$	0.1493	0.2197	0.2391	0.3798
SMOG-CF$_{\text{GC-MC}}$	0.1533	0.2229	0.2418	0.3831
SMOG-CF$_{\text{PinSag}}$	0.1537	0.2232	0.2427	0.3842
SMOG-CF$_{\text{NGCF}}$	**0.1574**	**0.2268**	**0.2453**	**0.3874**

5.4 Parameter Sensitivity: RQ3. We study the performance variation for our model on the MovieLens-latest dataset with respect to the embedding dimension D. Results are shown in Fig. 3. Our findings are as follows: The embedding dimension plays an important role. When D is too small, the embedding vectors' representation capacity is restricted and cannot fully exploit the abundant information in the user-item graph. When the embedding dimension increases, the performance improves rapidly at the very beginning, but soon the growth slows down and even declines after reaching the best setting due to overfitting.

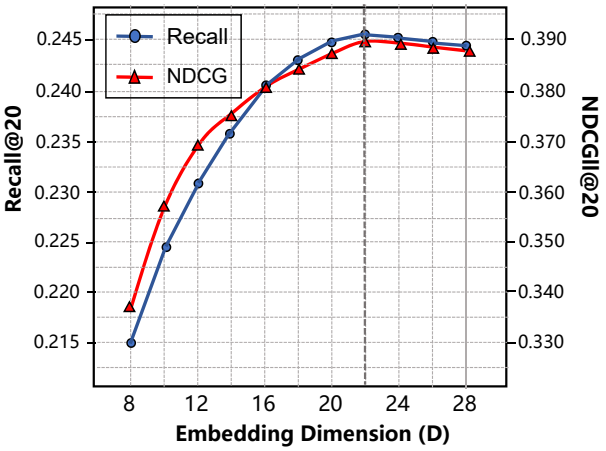

Figure 3: Recall@20 and NDCG@20 of SMOG-CF on MovieLens-latest under different embedding dimension settings. The vertical dotted lines mark the values set in this paper.

6 Conclusion

In this paper, we propose SMOG-CF, a GCN-based model for learning low-dimensional representations of users and items in a recommender system which can

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

directly capture higher-order connections in the user-item graph. The crucial point of our method is that we consider information propagation between neighboring nodes at any order by designing path-level information construction and aggregation formulas, and the concise form of the corresponding embedding update formula in matrix form ensures the simplicity and feasibility of our method. We also extended our method to various basic GCN models to make SMOG-CF a general framework. Our comparative experiments and ablation studies showed that our method can learn effective representations of mixed-order user-item collaborative information and thus improve recommendation performance.

As future work, we would like to extend our method to recommender systems with explicit feedback, meaning that the user-item graph would be weighted. We are also interested in applying our method to other graph-structured datasets such as knowledge graph learning.

References

[1] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan, *Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing*, in ICML, 2019, pp. 21–29.

[2] A. Grover and J. Leskovec, *node2vec: Scalable feature learning for networks*, in KDD, 2016, pp. 855–864.

[3] Z. Guo, Y. Zhang, and W. Lu, *Attention guided graph convolutional networks for relation extraction*, in ACL, 2019, pp. 241–251.

[4] W. L. Hamilton, Z. Ying, and J. Leskovec, *Inductive representation learning on large graphs*, in NIPS, 2017, pp. 1024–1034.

[5] F. M. Harper and J. A. Konstan, *The movielens datasets: History and context*, TiiS, 5 (2016), pp. 19:1–19:19.

[6] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, *Neural collaborative filtering*, in WWW, 2017, pp. 173–182.

[7] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, in ICLR, 2017.

[8] J. B. Lee, R. A. Rossi, X. Kong, S. Kim, E. Koh, and A. Rao, *Higher-order graph convolutional networks*, CoRR, abs/1809.07697 (2018).

[9] F. Lei, X. Liu, Q. Dai, B. W. Ling, H. Zhao, and Y. Liu, *Hybrid low-order and higher-order graph convolutional networks*, CoRR, abs/1908.00673 (2019).

[10] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel, *Gated graph sequence neural networks*, in ICLR, 2016.

[11] J. J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, *Image-based recommendations on styles and substitutes*, in SIGIR, 2015, pp. 43–52.

[12] B. Perozzi, R. Al-Rfou, and S. Skiena, *Deepwalk: online learning of social representations*, in KDD, 2014, pp. 701–710.

[13] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, *Deepinf: Social influence prediction with deep learning*, in KDD, 2018, pp. 2110–2119.

[14] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, *BPR: bayesian personalized ranking from implicit feedback*, in UAI, 2009, pp. 452–461.

[15] R. A. Rossi, N. K. Ahmed, and E. Koh, *Higher-order network representation learning*, in WWW Companion, 2018, pp. 3–4.

[16] R. A. Rossi, R. Zhou, and N. K. Ahmed, *Estimation of graphlet counts in massive networks*, IEEE Trans. Neural Netw. Learning Syst., 30 (2019), pp. 44–57.

[17] C. Sun, Y. Gong, Y. Wu, M. Gong, D. Jiang, M. Lan, S. Sun, and N. Duan, *Joint type inference on entities and relations via graph convolutional networks*, in ACL, 2019, pp. 1361–1370.

[18] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, *LINE: large-scale information network embedding*, in WWW, 2015, pp. 1067–1077.

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

[20] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, CoRR, abs/1710.10903 (2017).

[21] X. Wang, X. He, M. Wang, F. Feng, and T. Chua, *Neural graph collaborative filtering*, in SIGIR, 2019, pp. 165–174.

[22] Q. Wu, H. Zhang, X. Gao, P. He, P. Weng, H. Gao, and G. Chen, *Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems*, in WWW, 2019, pp. 2091–2102.

[23] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, *Representation learning on graphs with jumping knowledge networks*, in ICML, 2018, pp. 5449–5458.

[24] J. Yang, C. Chen, C. Wang, and M. Tsai, *Hop-rec: high-order proximity for implicit recommendation*, in RecSys, 2018, pp. 140–144.

[25] L. Yao, C. Mao, and Y. Luo, *Graph convolutional networks for text classification*, in AAAI, 2019, pp. 7370–7377.

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

[27] T. Zhang, B. Liu, D. Niu, K. Lai, and Y. Xu, *Multiresolution graph attention networks for relevance matching*, in CIKM, 2018, pp. 933–942.

[28] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, *Deep interest network for click-through rate prediction*, in KDD, 2018, pp. 1059–1068.

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited