An improved recommendation based on graph convolutional network

Yichen He1 · Yijun Mao1 · Xianfen Xie2 · Wanrong Gu1

Received: 8 June 2022 / Revised: 28 June 2022 / Accepted: 1 July 2022 / Published online: 11 July 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
Graph convolutional network is a recently developed artificial neural network method commonly used in recommendation system research. This paper points out three shortcomings of existing recommendation systems based on the graph convolutional network. 1. Existing models that take the one-hot encoding based on node ordinal numbers in the graph or encoding based on original entity attributes as input may not fully utilize the information carried by the attribute interactions. 2. Previous models update the node embeddings only by the first-order neighbors in the graph convolution layer, which is easily affected by noise. 3. Existing models do not take into account differences in user opinions. We propose an improved graph convolutional network-based collaborative filtering model to address these drawbacks. We identify inner and cross interaction between user attributes and item attributes, and then we take the vector representations of aggregated attributes graph as input. In the convolutional layer, we aggregate the second-order collaborative signals and incorporate the different user opinions. The experiments on three public datasets show that our model outperforms state-of-the-art models.

Keywords Graph convolutional network · High-order collaborative signal · Collaborative filtering · Recommendation system

Wanrong Gu
guwanrong@scau.edu.cn

Yichen He
heyichen666@hotmail.com

Yijun Mao
yijunmao@163.com

Xianfen Xie
txiexianfen2009@jnu.edu.cn

1 School of Mathematics and Information, South China Agricultural University, Tianhe, Guangzhou 510642, Guangdong, China

2 School of Economy, Jinan University, Tianhe, Guangzhou 510632, Guangdong, China
1 Introduction

Collaborative Filtering (CF) is one of the most important algorithms in recommendation systems. The recommendation algorithm based on collaborative filtering has been scholars’ most extensive and profound research (Kluver et al., 2018; Jalili et al., 2018). Collaborative filtering can be divided into memory-based and model-based collaborative filtering, depending on whether a machine learning model is used or not. Compared with the former, which relies on the similarity function, has high memory occupation, and is hard to discover nonlinear and implicit connections between entities, the model-based collaborative filtering uses matrix factorization (MF) (Mehta & Rana, 2017), Markov Chain (MC) (Quadrana et al., 2018), deep neural network (Liu & Wu, 2017) and other models to learn user and item embeddings from user-item interactions, and becomes the most widely studied recommendation algorithm.

In recent years, Graph Convolutional Networks (GCN) have emerged as the hottest algorithms in current recommendation systems due to their powerful ability to learn about entities and their relations on graph structures. GCN uses spectral graph theory to define graph convolution operations, which propagate and aggregate messages from neighboring nodes on the topological graph. GCN has the advantage of learning entity embedding by using both entity features and graph structure features. It extends the excellent feature learning and representation capabilities of traditional Convolutional Neural Networks (CNN) to non-regular graph data. The entities and their interactions in the recommendation scenario can be naturally represented by a topological graph, which makes it very suitable for implementing graph-based recommendation tasks based on the GCN model.

Due to the powerful feature representation learning capability, GCN-based recommendation models can better exploit the implicit and nonlinear collaborative signals on entity interaction graphs and usually obtain better accuracy than traditional collaborative recommendation models. Although graph convolutional networks have been recognized to have significant advantages in recommendation systems, the proposed GCN-based recommendation models usually neglect three aspects that need further improvement.

- A key point in designing a graph convolutional network is what information is used as the input to the network. Current graph convolutional networks take one or more of the following four as inputs: 1. The attribute information of the nodes in the graph. 2. Degrees of the nodes in the graph. 3. One-hot encodings of the node ordinal numbers in the graph. 4. Constant vectors of all 1. However, we argue that they may not be sufficient to use the important semantic features and the attribute interactions of entities, which are not conducive to accurately learning the embedding of the node.
- The graph convolution layers of most models do not take full advantage of the collaborative signals of higher-order neighbors. In contrast, Xu et al. (Xu et al., 2018) show that higher-order neighbours’ collaborative signals are also important for learning node embeddings. Several models stack multiple graph convolutional layers in the GCN framework to implicitly pass higher-order collaborative signals. However, they still update node embeddings with only first-order neighbors inside their convolutional layers. These approaches are not only difficult to control the signal strength when aggregating higher-order signals but also easily affected by noise. (see the analysis in Section 3.3 for details).
- The convolutional layers of existing models pay little attention to the influence of different user opinions on the recommendation results when propagating and aggregating
collaborative signals. The ratings represent the user opinions in recommendation systems. For example, if a user gives ratings of 2 and 5 to items $i_1$ and $i_2$ respectively, he obviously expresses a more “like” opinion on $i_2$. However, most existing models ignore the difference in user opinions when aggregating the first-order collaborative signals. We note that the GCMC model (Berg et al., 2017) attempts to separately aggregate collaborative signals at different rating levels. However, it still does not consider the differences in user opinions when fusing signals from multiple levels.

We design an Improved Graph Convolutional Network based Collaborative Filtering (IGCN-CF) recommendation model for rating prediction to address the above shortcomings. In this paper, we use the interaction of entity attributes, the collaborative signals of second-order neighbors, and the user opinions to improve the accuracy of the graph convolutional network in learning node embedding.

We summarize the main contributions of our work as follows:

• We emphasize the importance of considering the entity’s attributes on the graph convolution network, and categorize the attribute interactions into inner and cross interactions. Finally, we take the vector representations of aggregated attribute vectors as input in the IGCN-CF model.
• We propose a method to compute the second-order collaborative signals and their strength from the adjacency matrix, which uses to update the node embeddings in the graph convolution layer.
• We give a simple method to integrate user opinions into the first-order collaborative signals aggregation process, which facilitates learning of the node embeddings with user preferences.
• We propose an improved collaborative recommendation model based on the graph convolutional network, and its performance advantages are verified on several public datasets.

2 Related works

2.1 Implicit feedback-based recommendation methods

The Latent Factor Model (LFM) is the most widely used algorithm in collaborative filtering. The method represents the users’ ratings of items as a matrix, then mines the low-dimensional latent feature space by factorizing the matrix, and finally re-represents the users and items on the low-dimensional space. The model finally uses the inner product between the embedding vectors of users and items to characterize the correlation between them. Most efforts to improve the embedding function focus on incorporating side information. Such as Li et al. (Li et al., 2014) introduced users’ purchase, search, and browsing records based on the traditional one-class collaborative filtering model. They used the added user’s features to improve the accuracy of recommendations. Xin et al. (Dong et al., 2017) proposed a hybrid model, which is a variant of the stacked denoising auto-encoder. The model can efficiently incorporate users’ and items’ side information into latent factors. Guo et al. (Guo et al., 2019) proposed a collaborative filtering model that considers emotion and trust. The model excludes the malicious users from neighbors according to the degree of emotional consistency and obtains the recommendation list based on the trust
relationships of target users. Similarly, some scholars incorporated social relations (Wang et al., 2017) and external knowledge graphs (Wang et al., 2019; Ren et al., 2021) into the collaborative model. While LFM uses the inner product between the embedding vectors of users and items to fit the user-item interactions, its linear characteristic makes it not enough to reveal the complicated non-linear relationships between users and items (Hsieh et al., 2017).

2.2 Deep learning-based recommendation methods

Scholars improve the interaction function through deep learning (Liu & He, 2022; Fu et al., 2019) to capture the non-linear relationships between users and items. For instance, Guo et al. (Guo et al., 2017) and He et al. (He & Chua, 2017) used a Multilayer Perceptron (MLP) to model the interactions. Tay et al. (Tay et al., 2018) used Euclidean distance to represent the strength of the interaction. Although deep learning has significant advantages in recommendation systems, we argue that embedding functions of existing works are insufficient to yield optimal embeddings for collaborative filtering. Summarizing existing deep learning methods, the embedding function transforms the entities’ features (e.g. ID and attributes) into vectors. Therefore, they ignore the attribute interactions, which are not conducive to learning the node embeddings accurately.

2.3 Graph convolutional networks

Graph convolutional network is one of the deep learning methods, with rich results in theory and application. With the development of GCN, several recommendation models based on graph convolutional networks have been proposed successively. Berg et al. (Berg et al., 2017) implemented a GCMC (Graph Convolutional Matrix Completion) recommendation model using GCN as an encoder in a graph self-coding framework, which uses a single graph convolutional layer to aggregate the collaborative signals of first-order neighbors on the user-item bipartite graph. Wang et al. (Wang et al., 2019) proposed an NGCF (Neural Graph Collaborative Filtering) model, which stacks multiple graph convolutional layers to achieve the implicit transmission of collaborative signals of higher-order neighbors on the bipartite graph, and finally aggregates the node embeddings in all layers for the recommendation. NGCF model compensates for the weakness of the GCMC model, which can only utilize the first-order collaborative signals. He et al. (He et al., 2020) argued that the NGCF model takes One-hot encodings with no concrete semantics besides being an identifier as input, which will increase the difficulty of model training, resulting in reduced prediction performance. They proposed a light GCN model, which removes feature transformation, neighborhood aggregation, and nonlinear activation in the embedding propagation layers. Much effort has been devoted to improving the efficiency of graph convolution operations (Chen et al., 2020), using attention mechanisms (Song et al., 2019; Feng et al., 2019), incorporating side information (Fan et al., 2019; Wang et al., 2019; Hui et al., 2022), improving model scalability (Ying et al., 2018; Wang et al., 2018), and overcoming the cold-start problem (Zhang et al., 2019) to improve the GCN-based models. We argue that existing works do not take full advantage of the collaborative signals of higher-order neighbors. For example, GCMC model only utilizes the signals of first-order neighbors to update the node embeddings. The GraphRec (Fan et al., 2019) model aggregates the collaborative signals of friends directly using social relationships, which does not exploit the higher-order signals of items and requires additional side information. Compared to other
models that implicitly pass higher-order collaborative signals by stacking multiple graph convolutions, our model uses second-order collaborative signals to update the node embedding inside the graph convolution layer.

3 Problem statement & background

3.1 Problem statement

In the recommendation systems, denote by $R \in \mathbb{R}^{m \times n}$ the sparse rating matrix, which contains few non-zero elements $r_{u,i} \in \{1, 2, \ldots, T\}$. $r_{u,i}$ represent user $u$ rating of item $i$, with $T$ levels. Denote by $X_u \in \mathbb{R}^{m \times d_u}$ the entity characteristic of users, and by $X_v \in \mathbb{R}^{m \times d_v}$ the entity characteristic of items, where $m$ and $n$ is the number of user set $\mathcal{U}$ and item set $\mathcal{V}$; $d_u$ and $d_v$ is the dimension of the characteristic vector of users and items. The goal of the recommendation problem is to design a model to predict ratings of items that have not been rated and then realize the recommendation.

In graph neural network-based models, users, items and their rating relationships are usually considered as undirected bipartite graphs $G = \langle E, A \rangle$, where $E = \mathcal{U} \cup \mathcal{V}$ is the node set in the graph, and adjacent matrix $A \in \mathbb{R}^{(m+n)\times(m+n)}$ represents whether there is an interaction between nodes, which can be obtained directly from the rating matrix $R$. At this point, the rating prediction problem can be considered as a prediction of the unknown links on the bipartite graph $G$.

3.2 GCN-based recommendation model

Recommendation models based on graph convolutional networks use the idea of message passing to propagate and aggregate collaborative signals over the bipartite graph, and learn the node embeddings for recommendation prediction. The model structure usually consists of the input layer, graph convolution layer, and prediction layer.

3.2.1 Input layer

The entity nodes on the graph $G$ are encoded to obtain their initial low-dimensional embedding vector $e^0$, such as:

$$e^0 = Enc(x)$$

(1)

where $x$ is the information of entity nodes, which can be represented by entity original characteristics or one-hot encoding vectors based on node ordinal numbers; $e^0$ is the initial embedding vector after encoding, which can be subdivided into user embedding $e^0_u$ and item embedding $e^0_v$. $Enc(\cdot)$ is the encoding function.

3.2.2 Convolution layer

The convolution layer is also known as the “embedding propagation layer”. It uses the structural features of the bipartite graph to propagate and aggregate the collaborative signals of neighboring nodes on the graph to achieve the node embedding update. Its main operations include collaborative signals construction and node embeddings update.
take a user $u$ rating of an item $v$ as an example, the rating behavior reflects a certain degree of user preference for the item. Therefore, the first-order collaborative signal propagated by this behavior can be constructed as:

$$s_{u\rightarrow v} = f(e_u, e_v, p_{u,v}, \Theta^f)$$ (2)

where $p_{u,v} = 1/\sqrt{|N_u| \cdot |N_v|}$ is the signal strength coefficient. Denote by $N_u$ and $N_v$ the set of directly adjacent first-order neighbors of user $u$ and item $v$, where $|N_u|$ and $|N_v|$ is the node degree. $f(\cdot)$ is the encoding function of collaborative signals, where $\Theta^f$ are all parameters to be learned in the function.

The embedding update operation aggregates all collaborative signals of first-order neighbors on the bipartite graph. The update of the user embeddings can be expressed as:

$$e^\text{update}_u = g(e_u, \{ s_{u\rightarrow v} | v \in N_u \})$$ (3)

where aggregation function $g(\cdot)$ can be implemented by weighted average, max pooling, LSTM network, etc.

Similarly, when item $v$ receives a rating from user $u$, it indicates that the item’s quality matches the user’s preference to some extent. Therefore, the embedding of the item can be updated using the user collaborative signals as:

$$s_{v\rightarrow u} = f(e_v, e_u, p_{u,v}, \Theta^f)$$ (4)

$$e^\text{update}_v = g(e_v, \{ s_{v\rightarrow u} | u \in N_v \})$$ (5)

In this way, the graph convolution layer uses the first-order collaborative signals to achieve a single update of the entity node embeddings. In addition, many GCN-based recommendation models stack $L$ graph convolution layers to iteratively refine the node embedding, which get $e_u^1, e_u^2, \ldots, e_u^L$ and $e_v^1, e_v^2, \ldots, e_v^L$ in order. Our method also implicitly realizes the layer-by-layer propagation of higher-order collaborative signals.

### 3.2.3 Prediction layer

The prediction layer uses the final node embedding $e_u^L$ and $e_v^L$ to predict the unknown ratings between any user $u$ and item $v$, as:

$$\hat{r}_{u,v} = h(e_u^L, e_v^L, \Theta^h)$$ (6)

where prediction function $h(\cdot)$ is usually implemented by a bilinear decoding function or a multilayer perceptron network; $\Theta^h$ are all parameters in the function.

### 3.3 Problem analysis

As described in Section 1, a key point in designing a graph convolutional network is what information is used as the input to the network. The existing encoding functions do not utilize entity attributes or analyze all attribute interactions equally, which encode all user and item attributes as vectors without distinguishing their interactions. For example, suppose there is a user with attributes <Female,20-24> and an item with attributes <Romance,Comedy>. Existing encoding functions ignore the interaction, such as "whether
female users like romance movies” (<Female,Romance>). We argue that they do not take full advantage of the entity’s attribute information.

On the one hand, as described in Section 3.2, the collaborative signals aggregation (3) and (5) only use collaborative signals from directly interacting first-order neighbors and themselves, so they ignore the higher-order signals inside the graph convolution layer. However, according to the basic idea of collaborative filtering, user behaviors or preferences are also influenced by other similar users who can be considered as higher-order neighbors on the bipartite graph. Therefore, it is beneficial to aggregate higher-order signals in the graph convolution layer. Recently, Xu et al. (Xu et al., 2018) have pointed out the value of higher-order signals for optimizing node embeddings.

Several GCN-based recommendation models stack multiple graph convolutional layers to pass higher-order collaborative signals implicitly. However, they are not only difficult to control the strength when aggregating higher-order signals but also easily affected by noise. For example, suppose there is a connected path \( u_0 \rightarrow v_1 \rightarrow u_2 \rightarrow v_3 \rightarrow u_4 \) on the bipartite graph, where user \( u_4 \) rate item \( v_3 \) for accidental reasons. Although it is not similar to the preference of the target user \( u_0 \), this rating behaviour will still result in the collaborative signal of \( u_4 \) being implicitly passed backwards along with the stacked four convolutional layers to \( u_0 \), rather than be considered as a noisy signal being suppressed or eliminated.

On the other hand, in the aggregation (3) and (5), all the first-order neighbor signals are treated equally without considering the difference in the user’s opinions (i.e. rating value). However, recent researches (Fan et al., 2019; Xiang et al., 2010) point out that high-rated items are more reflective of user preferences than low-rated items and should play a more significant role in signal aggregation operations.

4 Our approach

In this section, we describe an end-to-end IGCN-CF model to address the shortcomings of the GCN-based recommendation model as described in Section 3.3. First, we improve the coding function of the input layer, fuse the inner interaction and cross interaction of the entities, and finally take the aggregated vectors with the characteristic information as input. Then, we introduce second-order collaborative signals and user opinions inside the graph convolution layer, which learn the node embeddings more effectively. Finally, we use MLP to achieve predictions of unknown ratings. The basic structure is shown in Fig. 1, including the improved input layer, the improved graph convolution layer and the prediction layer.

4.1 Input layer

Inspired by (Su et al., 2021), we fuse the inner interaction and cross interaction of the entities and take the aggregated vectors as input. Figure. 2 shows an overview of the input layer. First, we construct the user attribute graphs and item attribute graphs to represent the users and items. The attributes are treated as nodes in the attribute graphs, and each pair of nodes are connected as edges. Therefore, the attribute graphs are complete graphs. For example, if a user is male and 20-24 years old, in the attribute graph of this user, we take “male” and “20-24” as nodes, and two nodes are connected with an edge. Donate by \( u_i^U \) the node \( i \) embedding in a user attribute graph and by \( u_i^I \) the node \( i \) embedding in an item attribute graph. Due to the symmetric modelling of user attribute graphs and item attribute graphs, we omit the superscript \( U \) and \( I \) in the following section.
Fig. 1 The basic structure of IGCN-CF

Fig. 2 An overview of the input layer
4.1.1 Inner interaction

Inner interactions are used for user (item) characteristics learning. We use the $MLP \in \mathbb{R}^{2 \times d} \rightarrow \mathbb{R}^d$ to model the inner interactions:

$$z_{i,j} = MLP(u_i, u_j)$$ (7)

where $z_{i,j}$ are the inner interaction results of node pair $i$ and $j$, $u_i$ and $u_j$ is the embedding of node $i$ and $j$. The same modelling is performed for each node pair in the attribute graph, and the final results of the modelling are aggregated with the element-wise sum:

$$z_i = \sum_{j \in N_i} z_{ij}$$ (8)

where $z_i$ is the aggregated inner interaction results of node $i$, and $N_i$ is a set of neighbors of node $i$.

Moreover, inner interactions are employed to capture user (item) characteristics and are intrinsically complicated. Note that in contrast to cross interaction model results that reveal similarity (will be discussed in Section 4.1.2), a high inner interaction result does not imply that the two attributes are similar when learning the characteristics of a user (an item). As a result, we use a neural approach that non-linearly to model the two attributes.

4.1.2 Cross interaction

Cross interactions describe the dependency between the user attributes and the item attributes. We expect a high interaction result between an attribute $u_i^U$ and an attribute $u_j^I$ if $u_i^U$ has a strong preference for $u_j^I$. For example, if female users prefer romance movies, the interaction result between the attributes “Female” and “Romance” should be high. Moreover, if a user attribute has a strong preference for an item attribute, their embeddings should be similar after training in collaborative filtering. Inspired by (He & Chua, 2017), we use Bi-interaction to model the cross interaction:

$$s_{ij} = u_i \odot \hat{u}_j$$ (9)

where $s_{ij}$ are the cross interaction results of node pair $i$ and $j$, $u_i$ is the embedding of node $i$ in one graph, and $\hat{u}_j$ is the embedding of node $j$ in the other graph. The element-wise product $\odot$ makes sure $s_{ij}$ is high if node $i$ has a high preference of node $j$. Similar to inner interaction, we aggregate the results with the element-wise sum:

$$s_i = \sum_{j \in \hat{V}} s_{ij}$$ (10)

where $s_i$ is the aggregated cross interaction results of node $i$, and $\hat{V}$ is a set of nodes in the other attribute graph.

4.1.3 Information fusing

We can get the final embeddings by fusing the initial node embeddings $u_j$, the inner interaction results $z_i$, and the cross interaction results $s_i$. The fusing function reference document (Su et al., 2021), we use Gated Recurrent Unit ($GRU \in \mathbb{R}^{3 \times d} \rightarrow \mathbb{R}^d$) to get the fused node representation:
Then we aggregate the node representations with the element-wise sum:

\[ e^0 = \sum_{i \in V} u'_i \]

where \( e^0 \) are the initial input of the graph convolution network in this paper.

Note that the user or item attributes may not be available in some cases. In this case, our model can also be used. When the user and item attributes are unavailable, the attribute graph becomes a single node (user/item ID). Then there is no inner interaction modeling, simply cross interaction modeling between the user ID and the item ID, i.e. \( u'_i = GRU(u_i, s_i) \), where \( u_i \) is the node representation of entity ID.

### 4.2 Graph convolution layer

Similar to the NGCF model, the model in this paper achieves layer-by-layer refinement of the node embedding representation by stacking \( L \) graph convolution layers. We improve the graph convolution layer by aggregating the second-order collaborative signals and user opinions. This section takes the \( l \in \{1, 2, \ldots, L\} \)-th graph convolution layer as an example to introduce the main idea of our models. Figure 3 shows an overview of the graph convolution layer.

#### 4.2.1 Second-order collaborative signals construction

Let \( u_i \rightarrow v_k \rightarrow u_j \rightarrow \ldots \) be any connectivity path from user \( u_i \) on the bipartite graph \( G \), where \( i, k, j, \ldots \) represent the node number. We call \( v_k \) a first-order neighbor of \( u_i \), \( u_j \) a second-order neighbor of \( u_i \), and so on, according to the number of hop on the path from user \( u_i \).

As discussed in Section 3.3, aggregating higher-order collaborative signals in the graph convolution layer is beneficial for learning node embedding representations. However, our model only uses the collaborative signals of the second-order neighbors.
among all higher-order neighbors for two following reasons: 1. According to Xu et al. (Xu et al., 2018), the strength of the collaborative signals of higher-order neighbors on the connected path decreases rapidly with increasing order, and the noise signals tend to accumulate on the path. This assertion also explains why many GCN-based models stack only three or four convolutional layers. 2. Second-order neighbors are more similar in their preferences to the target entity than higher-order neighbors, and the more common interaction entities that second-order neighbors have with the target entity, the more similar they are. Moreover, the experimental results in Section 5 verify that adding fourth-order collaborative signals cannot significantly improve the performance of the recommended model.

Inspired by the document (Wang et al., 2006) in terms of recommended entity similarity: the more similar neighbors transmit more signals, and the neighbors with more common interaction entities transmit more signals. We propose a method to compute the second-order collaborative signals and their strength. Let \( u_j \) be the second-order neighbor of target user \( u_i \) on the graph \( G \), \( u_i \rightarrow v_k \rightarrow u_j \) be any connected path of length two between them. We define the second-order signal propagating from \( u_j \) to \( u_i \) as:

\[
s^{l=2}_{i \rightarrow j} = \sum_{k \in N_i \cap N_j} p_{i,k,j} (e^{l=1}_{u,j} + e^{l=1}_{u,i} \odot e^{l=1}_{u,j})
\]  

(13)

where \( e^{l=1}_{u,j} \) denotes the embedding of user \( u_j \) in the \( l-1 \)-th layer, \( \odot \) denotes the element-wise product, \( N_i \cap N_j \) denotes the set of common interaction items between \( u_j \) and \( u_i \), and \( p_{i,k,j} \) denotes the signal strength coefficient of the second-order collaborative signals in the network, which is related to the degree of nodes on the path. In this paper, we set \( p_{i,k,j} = 1/\sqrt{|N_i| \cdot |N_j| \cdot |N_k|} \). Note that \( e^{l=1}_{u,i} \odot e^{l=1}_{u,j} \) also plays a role in regulating the strength of collaborative signals of neighboring nodes, so that similar neighbors propagate more second-order collaborative signals. For example, if a neighbor \( u_i \) and a target user \( u_j \) have the similar embedding, since the vectors have the same direction, the product term will enhance their collaborative signal, otherwise, the effect will be weak.

The analysis shows that (13) has a good suppression effect on the noise signal. Suppose user \( u_i \) does not have similar preferences to target user \( u_j \), but becomes a second-order neighbor of \( u_i \) only through a chance rating operation. However, since their node embeddings are not similar, resulting in weak signal enhancement of the element-wise product term, and considering that their common interaction items \( N_i \cap N_j \) are very few, the noise collaborative signals propagated from \( u_j \) to \( u_i \) are also very limited.

Denote by \( N^{(2)}_i \) all the second-order neighbors of user \( u_i \), the aggregated second-order collaborative signal of \( u_i \) is:

\[
S^{(2)}_{u,i} = \sum_{j \in N^{(2)}_i} s^{(2)}_{i \rightarrow j} = \sum_{j \in N^{(2)}_i} \sum_{k \in N_i \cap N_j} \frac{e^{l=1}_{u,j} + e^{l=1}_{u,i} \odot e^{l=1}_{u,j}}{\sqrt{|N_i| \cdot |N_j| \cdot \sqrt{|N_k|}}}
\]  

(14)

Symmetrically, it is easy to get the aggregated second-order collaborative signal \( s^{(2)}_{v,j} \) of item \( v_j \) in the \( l \)-th layer. Here we will not go into details.

Denote by \( E^{l=1} \in \mathbb{R}^{d \times (m+n)} \) the matrix of all embedding vectors in the \( (l-1) \)-th layer, and by \( S^{(2)} \in \mathbb{R}^{d \times (m+n)} \) the signal matrix aggregated by second-order collaborative signals of all nodes in the \( l \)-th layer, where row vectors consist of \( s^{(2)}_{u,i} \) or \( s^{(2)}_{v,j} \). Based on the
spectral graph theory (Bruna et al., 2013), the concise form of the second-order collaborative signal matrix \( S^{(2)} \) is:

\[
S^{(2)} = \mathcal{L}^{(2)} E^{l-1} + (\mathcal{L}^{(2)} E^{l-1}) \odot E^{l-1}
\]

where \( \mathcal{L}^{(2)} \) is the de-diagonalization second-order Laplacian matrix, \( \mathcal{L}^{(2)} = \mathcal{L}^{(2)} - \text{diag}(\mathcal{L}^{(2)}) \). \( \mathcal{L}^{(2)} \) is the second-order Laplacian matrix which represents the second-order signal strength based on the node degree and is the matrix form of the signal strength coefficient \( p_{i,k,j} \):

\[
\mathcal{L}^{(2)} = (D^{-\frac{1}{2}} \cdot A \cdot D^{-\frac{1}{2}})^2
\]

where \( D \) is the diagonal matrix consists of node degrees; \( A \) is the adjacency matrix of graph \( G \). In this paper, \( A = \begin{pmatrix} 0 & z(R) \\ z(R^T) & 0 \end{pmatrix} \), where \( z(\cdot) \) is a function that resets the non-zero elements of the matrix to one, \( R \) is the rating matrix.

### 4.2.2 First-order collaborative signals with user opinions construction

Users’ ratings of items indicate the existence of an interactive relationship between them, and specifically express the users’ opinions. For example, in the movie recommendation, a rating of 5, 3, and 1 expresses three different opinions of users: “like it very much”, “like it a little”, and “do not like it”. According to the document (Xiang et al., 2010), high ratings with positive views are more reflective of users’ true preferences. Therefore, incorporating user opinions when aggregating the first-order collaborative signals can more accurately learn user embeddings.

We draw on the message passing mechanism of GCN and propose a first-order collaborative signal with the user opinions construction method. It denotes the aggregated first-order collaborative signal converged of user \( u_i \) as:

\[
S^{(1)}_{u,i} = \sum_{j \in \mathcal{N}_i} c_{ij} p_{ij} W^l \epsilon^{l-1}_{v,j}
\]

where opinion coefficient \( c_{ij} \) denotes user \( u_i \)’s opinion of item \( v_j \), the values in this paper are taken as normalized ratings. Specifically, \( c_{ij} = ru_i,v_j/\max(R) \); \( p_{ij} \) is the first-order collaborative signal strength, which is the same as \( p_{u,v} \) in (2); \( W^l_1 \in \mathbb{R}^{h \times h} \) is the weight matrix to be learned in the \( l \)-th layer.

Obviously, the opinion coefficients of the higher rating items take larger values, and they transmit more collaborative signals. Therefore, this approach with opinions aggregates the first-order collaborative signals more reasonably.

Symmetrically, it is easy to get the aggregated first-order collaborative signal \( S^{(1)}_{v,j} \) of item \( v_j \) in the \( l \)-th layer. We will not go into details.

Similarly, We denote the concise form of the first-order collaborative signal matrix based on the spectral graph theory as:

\[
S^{(1)} = (C \odot \mathcal{L}^{(1)}) E^{l-1} W^l_1
\]

where \( C \) is the normalized opinion coefficient matrix consist of \( c_{i,j} \), specificaly, \( C = \begin{pmatrix} \frac{R}{\max(R)} & 0 \\ 0 & \frac{\max(R)}{\max(R)} \end{pmatrix} \); \( \mathcal{L}^{(1)} \) is the first-order Laplacian matrix, and \( \mathcal{L}^{(1)} = (D^{-\frac{1}{2}} \cdot A \cdot D^{-\frac{1}{2}}) \).
4.2.3 Node embeddings update

We aggregate the different order collaborative signals and node embeddings in the previous layer, and update the node embedding vectors in the $l$-th layer as:

$$E^l = \text{ReLU}((E^{l-1} \oplus S^{(1)} \oplus S^{(2)})W^l_2)$$ (19)

where $\text{ReLU}()$ is the activation function; $\oplus$ denotes the concatenate operation; $W^l_2 \in \mathbb{R}^{3h \times h}$ is the weight matrix to be learned. Note that (19) fuses the embedding of the previous layer $E^{l-1}$ to the collaborative signals as well, which is equivalent to the self-connected signals on the bipartite graph (Chen et al., 2020), and helps to avoid the “Over-Smoothing” problem in the embedding learning process.

The IGCN-CF model can stack $L$ graph convolution layers to achieve layer-by-layer refinement of the embedding representation according to (19). Then we can get $E^1, E^2, ..., E^L$.

4.3 Nonlinear prediction layer

Figure 4 shows an overview of the input layer. The final learned node embeddings $e^L_u$ and $e^L_v$ of the graph convolution layer are connected and input to a 2-layers MLP network to predict the unknown rating $\hat{r}_{u,v}$ between user $u$ and item $v$:

$$\hat{r}_{u,v} = W_4(\text{ReLU}(W_3(e^L_u \oplus e^L_v)))$$ (20)

![Fig. 4 An overview of the prediction layer](image)
where \( W_3 \in \mathbb{R}^{2h \times d'} \) and \( W_4 \in \mathbb{R}^{d' \times 1} \) is the weight matrix of MLP to be learned; \( d' \) is the number of neurons in the first layer (hidden layer) of the MLP.

Noted that compared to some linear prediction models used in the documents (Berg et al., 2017; Wang et al., 2019; Chen et al., 2020), the nonlinear MLP model used in this paper has a stronger ability to capture the complex relationships between node embedding vectors and is more suitable for the rating prediction.

4.4 Model training

The IGCN-CF model is an end-to-end recommendation model. We give an optimized objective function based on the mean square error to train the parameters:

\[
\text{obj} = \frac{1}{2|O|} \sum_{(u,v) \in O} (r_{u,v} - \hat{r}_{u,v})^2 + \lambda \|\Theta\|_2^2
\]  

(21)

where \((u,v)\) is the edges of ratings \( r_{u,v} \) on the bipartite graph \( G \), and \( O \) is the set of them; \( \Theta = \{ W_1, W_2 \}_{i=1}^L, W_3, W_4 \) is all parameters to be trained in the model; \( \|\Theta\|_2^2 \) represents the \( L_2 \) norm to regularize all parameters, where \( \lambda \) is the regularization coefficient.

In addition, the node dropout method used in the document (Wang et al., 2019) is also borrowed in the model training to prevent overfitting, and the dropout rate is \( \eta \). Finally, we use the Adam gradient descent optimizer (Kingma & Ba, 2014) to optimize (21), and use the minibatch method to improve the model training efficiency.

4.5 Discussions

We aggregate the collaborative signals of homogeneous second-order neighbors explicitly inside the graph convolution layer of the IGCN-CF model. Theoretically, it is possible to aggregate higher-order collaborative signals. For example, when aggregating both second and fourth order collaborative signals in the bipartite graph, it is sufficient to simply replace (16) as:

\[
L^{(2,4)} = (D^{-\frac{1}{2}} \cdot A \cdot D^{-\frac{1}{2}})^2 + (D^{-\frac{1}{2}} \cdot A \cdot D^{-\frac{1}{2}})^4
\]  

(22)

In addition, in the recommendation model proposed by Fan et al. (Fan et al., 2019), the collaborative signals of friends in social networks are aggregated into the graph convolution layer. Social friends are equivalent to homogeneous higher-order neighbors with high similarity in the bipartite graph, and aggregating their collaborative signals is obviously beneficial for learning the embedding representation of user nodes. By contrast, the model in this paper reasonably computes the second-order neighbors and their signal strengths by matrix operations without additional side information. Moreover, when there is side information about homogeneous node relationships in the dataset, such as social relationship matrix \( Q_u \in \mathbb{R}^{m \times m} \) and item similarity matrix \( Q_v \in \mathbb{R}^{n \times n} \), it is easy to incorporate them into the model by modifying (16) as:

\[
L^{(2,4)} = (D^{-\frac{1}{2}} \cdot A \cdot D^{-\frac{1}{2}})^2 + \alpha \left( \begin{array}{c} Q_u \ 0 \\ 0 \ Q_v \end{array} \right)
\]  

(23)

where \( \alpha \) is the incorporation parameter.
Finally, the 3-layer structure of IGCN-CF has similarities with the NGCF and LightGCN model, but they have significant differences in the input layer, collaborative signals utilization, user opinions combination, and prediction layer design. Taking Fig. 5 as an example, the red arrows show the propagation of the signals. NGCF and LightGCN stack 3 convolution layers to propagate and aggregate the higher-order collaborative signals like $u_1 \leftarrow v_2 \leftarrow u_3 \leftarrow v_4$. As discussed in Section 3.3, we argue that this approach is easily affected by noise. Thus our model propagates and aggregates the higher-order collaborative signals inside the graph convolution layer rather than stacking multiple layers. Compared with NGCF and LightGCN, our model can better control the strength of the higher-order collaborative signals and reduce the effect of the noisy signals.

5 Experiments

We conduct experiments on three real-world datasets to evaluate the proposed IGCN-CF recommendation model and compare its performance with state-of-the-art models.

5.1 Datasets

To evaluate the effectiveness of IGCN-CF, we conduct experiments on three datasets: MovieLens 1M, Book-crossing, and Yelp2018. Table 1 shows their statistic information. Below are the descriptions of the datasets:

**MovieLens-1M** is a dataset collected by the GroupLens Research on movie ratings, which is the most famous dataset in the recommendation system. We use the pre-processed
dataset provided in document (Su et al., 2021), which enriches the dataset with movies’ other attributes, such as directors and casts from IMDB. In the dataset, the ratings greater than 3 are regarded as positive ratings, and we retain users with more than 10 positive ratings.

**Book-crossing** is a dataset with users’ implicit and explicit ratings of books. We also use the pre-processed dataset provided in document (Su et al., 2021). In the dataset, all rated explicit ratings are regarded as positive ratings due to its sparsity, and we retain users with more than 20 positive ratings.

**Yelp2018** is a dataset adopted from the 2018 edition of the Yelp challenge. We use the pre-processed dataset provided in document (He et al., 2020). In the dataset, all ratings are regarded as positive ratings, and we retain users with more than 10 positive ratings. Note that this dataset does not contain any user or item attribute information.

### 5.2 Experimental set-up

In the experiments, we selected four state-of-the-art recommendation models as baselines: NFM (He & Chua, 2017)\(^1\), GCMC (Berg et al., 2017)\(^2\), GraphRec (Fan et al., 2019)\(^3\), NGCF (Wang et al., 2019)\(^4\), and LightGCN (He et al., 2020)\(^5\). We run the codes released by the authors. NFM uses a deep neural network to replace the dot product operation of embedding vectors. In our experiments, we use Equation (21) to replace its original optimization objective so that it can be applied to rating prediction. The other four models are the prevailing GCN-based collaborative filtering recommendation models. For the NGCF and LightGCN model, we also use Equation (21) instead of their original paired BPR (Bayesian personalized ranking) loss objective function for rating prediction.

Unless otherwise specified, we use the following hyper-parameter settings: The embedding size \(h\) is fixed to 64; the number of graph convolutional layers \(L\) is fixed to 3; the regularization coefficient \(\lambda\) is searched in \(\{0.01, 0.05, 0.1, 0.5, 1, 5, 10\}\) by cross validation; the hidden layer parameter \(d\) of MLP is 16; we use Adam as the optimization algorithm, where the learning rate is \(10^{-3}\) and the batch size is fixed at 1024. The maximum number of epochs is 1000; the size of each batch of training data is set to 1024; the node dropout ratio \(\eta\) is 0.3. For all the four baselines, we use the author-provided implementations.

We randomly split each dataset into the training set, validation set and test set in the ratio of 8:1:1. We use the NDCG@K and recall@K to evaluate the effectiveness of top-K recommendation and preference ranking of our model and baseline models. By default, we set \(K = 10\). All results are the average of three replicate experiments.

---

1. [https://github.com/hexiangnan/neural_factorization_machine](https://github.com/hexiangnan/neural_factorization_machine)
2. [https://github.com/riannevdberg/gc-mc](https://github.com/riannevdberg/gc-mc)
3. [https://github.com/wenqifan03/GraphRec-WWW19](https://github.com/wenqifan03/GraphRec-WWW19)
4. [https://github.com/xiangwang1223/neural_graph_collaborative_ltering](https://github.com/xiangwang1223/neural_graph_collaborative_ltering)
5. [https://github.com/kuandeng/LightGCN](https://github.com/kuandeng/LightGCN)
Performance comparison

We compare the performance of the IGCN-CF model with the comparison methods on each dataset, and their prediction performances on the test set are shown in Table 2. The numbers in bold represent the best results for each dataset, and the numbers in underline represent the best baseline results. We further compare the training loss between our model and the optimal baseline LightGCN on each dataset. The results are shown in Fig. 6.

Observing the experimental results in Table 2, four conclusions can be drawn:

- The prediction performances of the four recommendation models based on graph convolutional networks are significantly better than those of the NFM model based on matrix factorization. This result indicates that the GCN-based models can learn the embedding representations of nodes more accurately and get better prediction results due to the fusion of entity attributes of nodes and graph structure features on the topological graph.
- The GCMC model has the worst performance among the four GCN-based recommendation models, especially on Yelp2018, which is a very sparse dataset, because it uses only one graph convolutional layer and only aggregates the collaborative signals of first-order neighbors. Therefore, it has insufficient learning ability for node embedding.
- GraphRec, NGCF and LightGCN leverage higher-order collaborative signals, achieving better prediction results than the GCMC model. The difference is that the GraphRec model directly aggregates the collaborative signals of social friends, who are equivalent to similar higher-order neighbors in a bipartite graph, while the NGCF and LightGCN model implicitly propagates and aggregates higher-order collaborative signals by stack-

### Table 2 Summary of the performance in comparison with baselines

| Dataset     | NFM      | GCMC     | GraphRec | NGCF     | LightGCN | IGCN-CF  |
|-------------|----------|----------|----------|----------|----------|----------|
|             | recall   | ndcg     | recall   | ndcg     | recall   | ndcg     |
| Moveilens-1M| 0.1731   | 0.4953   | 0.2108   | 0.5743   | 0.2251   | 0.6408   |
| Book-crossing| 0.1025   | 0.1768   | 0.1198   | 0.1908   | 0.1275   | 0.2079   |
| Yelp2018    | 0.1208   | 0.1973   | 0.1393   | 0.2079   | 0.1533   | 0.2284   |

|             | 0.2288   | 0.6375   | 0.1312   | 0.2121   | 0.1547   | 0.2329   |
|             | 0.2304   | 0.6477   | 0.1339   | 0.2159   | 0.1642   | 0.2445   |
|             | 0.2336   | 0.6557   | 0.1359   | 0.2204   | 0.1612   | 0.2424   |

| %Improvement| 1.39%    | 1.24%    | 1.49%    | 2.08%    | -1.86%   | -0.87%   |

![Fig. 6 Training Loss curves of IGCN-CF and LightGCN on each dataset](image_url)

### 5.3 Performance comparison

We compare the performance of the IGCN-CF model with the comparison methods on each dataset, and their prediction performances on the test set are shown in Table 2. The numbers in bold represent the best results for each dataset, and the numbers in underline represent the best baseline results. We further compare the training loss between our model and the optimal baseline LightGCN on each dataset. The results are shown in Fig. 6.

Observing the experimental results in Table 2, four conclusions can be drawn:

- The prediction performances of the four recommendation models based on graph convolutional networks are significantly better than those of the NFM model based on matrix factorization. This result indicates that the GCN-based models can learn the embedding representations of nodes more accurately and get better prediction results due to the fusion of entity attributes of nodes and graph structure features on the topological graph.
- The GCMC model has the worst performance among the four GCN-based recommendation models, especially on Yelp2018, which is a very sparse dataset, because it uses only one graph convolutional layer and only aggregates the collaborative signals of first-order neighbors. Therefore, it has insufficient learning ability for node embedding.
- GraphRec, NGCF and LightGCN leverage higher-order collaborative signals, achieving better prediction results than the GCMC model. The difference is that the GraphRec model directly aggregates the collaborative signals of social friends, who are equivalent to similar higher-order neighbors in a bipartite graph, while the NGCF and LightGCN model implicitly propagates and aggregates higher-order collaborative signals by stack-
ing multiple graph convolution layers. This illustrates that aggregating higher-order collaborative signals helps to improve the performance of the recommendation model.

- The IGCN-CF model in this paper achieves the best prediction performance on most of the datasets, only slightly worse than the LightGCN model on the Yelp2018. Because the Yelp2018 contains only the users’ and item’s IDs without any attribute information, we cannot fully utilize the attribute interactions. Thus the prediction performance of our model is not optimal in this dataset. However, as shown in Fig. 6, compared with the LightGCN model, our model consistently obtains lower training loss, and the training loss of our model tends to be smooth in lower epochs, which means that our model fits the training data better than LightGCN. Our model outperforms the LightGCN model on Movielens-1M and Book-crossing but is slightly worse on Yelp2018 also indicates the significance of the attribute interaction. Our model performs better when there is sufficient attribute information about the users and the items. Moreover, we use “A/B testing” for the significance test and find that the experimental results of the IGCN-CF model are significantly better than other models with 95% confidence in most cases. The superior performance of our model can be attributed to the fact that the model leverages the attributes of entities and integrates them as inputs. In contrast, the other three GCN-based model only use one-hot encoding vectors based on node ordinal numbers as input. Furthermore, our model further exploits higher-order collaborative signals inside the graph convolution layer and introduces user opinions into the signal aggregation. In contrast, the GraphRec model only obtains the higher-order collaborative signals of users directly from social relationships without considering the higher-order signals of items; the NGCF and LightGCN model does not explicitly exploit the higher-order collaborative signals inside the graph convolutional layer and does not consider the influence of users’ opinions.

5.4 Study of IGCN-CF

To further analyze the reasons for the excellent performance achieved by the IGCN-CF model, we give five variants of its model and compare their performance on the datasets, including: IGCN-CF-oh, IGCN-CF-s4, IGCN-CF-ro, IGCN-CF-s1, IGCN-CF-l. Compared with the basic model, the IGCN-CF-oh model takes one-hot encoding vectors based on node ordinal numbers as input; the IGCN-CF-s4 model employs the fourth-order collaborative signals; the IGCN-ro model removes the user opinions in (18); the IGCN-CF-s1 model removes the second-order collaborative signals in (19); the IGCN-CF-l directly uses the inner product \((e_u^L)^T \cdot e_v^L\) instead of the nonlinear neural network in (20).
The experimental parameter settings are consistent with those described in Section 5.2. The average prediction performances of three replicate experiments are summarized in Table 3. According to Table 3, we find that:

- The prediction performance of the IGCN-CF-oh model is inferior to the basic model in all data sets. This result means that distinguishing the entity attribute interactions into inner interactions and cross interactions, taking the aggregated vectors as input helps to improve the prediction performance.

- Although the IGCN-CF-s4 model adds fourth-order collaborative signals, its prediction performance is no better than that of the basic model. This result indicates that the higher-order collaborative signals are weaker and may have noise, which is not helpful in significantly improving the performance of the recommended model. This is also the reason why the model in this paper only leverages the homogeneous second-order collaborative signals in the graph convolution layer.

- The prediction performance of the IGCN-CF-ro model without the user’s opinions is lower in all datasets than in the basic model. This result shows that incorporating the user’s opinions is beneficial to improving the performance of the recommendation model.

- The IGCN-CF-s1 model without second-order collaborative signals performs the worst on each dataset. This result indicates that leveraging the second-order collaborative signals in the graph convolution layer is beneficial to improve the rating prediction accuracy.

- The IGCN-CF-1 model with a linear prediction layer also achieves good prediction performance, which is slightly worse than the basic model. This result indicates that: on the one hand, the graph convolution layer has been able to learn about node embedding representations accurately, and even using simple linear prediction methods can achieve a good prediction performance; on the other hand, using a nonlinear prediction layer can further improve the prediction accuracy of the model.

### 5.5 Effect of graph convolution layer numbers

Note that NGCF, LightGCN and IGCN-CF stack multiple graph convolutional layers to achieve layer-by-layer refinement of node embeddings. We experimentally examine the

![Fig. 7 Results of three models at different layer numbers on Movielens-1M](image-url)
prediction performances of the three models when taking different numbers of graph convolutional layers. Fig. 7, Fig. 8, and Fig. 9 shows the results.

According to the figures, We have the following observations:

- The prediction performances of the three models are obviously affected by the graph convolution layer numbers. They show a similar trend on three datasets: the prediction performances first increase significantly with the increasing of layer numbers, but after 3 layers, they no longer increase significantly and show a slight decrease. This phenomenon indicates that stacking too many graph convolution layers is prone to overfitting problems. Therefore, the maximum number of layers of the model in this paper is set to 3.

- In most cases (except on the Yelp2018 dataset), the prediction performances of our model are significantly better than those of the NGCF and LightGCN model, especially when the number of graph convolutional layers is smaller. This phenomenon is attributed to the advantage of the enhanced graph convolutional layers used in the IGCN-CF model, which improves the accuracy of node embeddings learning by aggregating the second-order collaborative signals inside the layers and incorporating the user’s opinions. Thus our model achieves excellent prediction accuracy even with fewer graph convolutional layers.
6 Conclusion

In this paper, we propose an end-to-end collaborative recommendation model based on an improved graph convolutional network for the rating prediction in recommender systems. The model leverages the attribute interactions of entities and uses the aggregated attribute vectors as input, which is more conducive to learning node embeddings. Moreover, it obtains an improved graph convolution layer by adding second-order collaborative signals and incorporating user opinions. The model stacks multiple layers to capture the collaborative signals on the bipartite graph, improving the accuracy of node embeddings learning. Finally, the model uses a nonlinear MLP network to achieve rating prediction. Experimental results on several recommendation datasets show that the prediction accuracy of this model outperforms state-of-the-art recommendation models.

The entity attribute graphs in our model are complete graphs with high memory overhead. The model needs to store the adjacency matrix of the bipartite graph at runtime, which may encounter a storage bottleneck on the vast graph. In future work, we will study the node sampling technique based on the random walk (He & Chua, 2017) to improve the model in this paper so that it can be adapted to very large scale recommendation applications.

Author Contributions Yichen He and Yijun Mao wrote the main manuscript text. Yichen He and Wanrong Gu wrote the key code and conducted the experiments. Xianfen Xie conceived and designed the analysis. All authors reviewed the manuscript.

Funding This work was financially supported by the Philosophy and Social Science Planning Project of Guangdong (GD19CGL34), the Guangdong Computational Science Key Laboratory Development Fund Project (2021010), the National Statistical Science Research Project (2020LY018), the National Social Science Foundation of China (18ZDA093), and Natural Science Foundation of Guangdong (2022A1515011489).

Data Availability Data and Codes are available upon request.

Declarations

Ethics approval and consent to participate All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

Consent for Publication Authors give full consent for publication.

Competing interests The authors declare that they have no conflict of interest.

References

Kluver, D., Ekstrand, M.D., & Konstan, J.A. (2018). Rating-based collaborative filtering: Algorithms and evaluation. Social Information Access, pp 344–390, https://doi.org/10.1007/978-3-319-90092-6_10.
Jalili, M., Ahmadian, S., Izadi, M., Moradi, P., & Salehi, M. (2018). Evaluating collaborative filtering recommender algorithms: A survey. IEEE Access, 6, 74003–74024. https://doi.org/10.1109/ACCESS.2018.2883742.
Mehta, R., & Rana, K. (2017). A review on matrix factorization techniques in recommender systems. In 2017 2nd International conference on communication systems, computing and it applications (CSCITA), IEEE, pp 269–274. https://doi.org/10.1109/CSCITA.2017.8066567.
Quadrana, M., Cremonesi, P., & Iannach, D. (2018). Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)*, 51(4), 1–36. https://doi.org/10.1145/3190616.

Liu, J., & Wu, C. (2017). Deep learning based recommendation: A survey. In K. Kim N. Joukov (Eds.) (eds.) *Information Science and Applications 2017 pp 451–458*. Springer, Singapore. https://doi.org/10.1007/978-981-10-4154-9_52.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-I., & Jegelka, S. (2018). Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pp 5453–5462, PMLR. https://proceedings.mlr.press/v80/xu18c.html.

Berg, R.V.D., Kipf, T.N., & Welling, M. (2017). Graph convolutional matrix completion. arXiv:1706.02263.

Li, Y., Zhai, C., & Chen, Y. (2014). Exploiting rich user information for one-class collaborative filtering. *Knowledge and Information Systems*, 38(2), 277–301. https://doi.org/10.1007/s10115-012-0583-9.

Dong, X., Yu, L., Wu, Z., Sun, Y., Yuan, L., & Zhang, F. (2017). A hybrid collaborative filtering model with deep structure for recommender systems. Proceedings of the AAAI Conference on Artificial Intelligence, vol 31(1). https://doi.org/10.1609/aaai.v31i1.10747.

Guo, L., Liang, J., Zhu, Y., Luo, Y., Sun, L., & Zheng, X. (2019). Collaborative filtering recommendation based on trust and emotion. *Journal of Intelligent Information Systems*, 53 (1), 113–135. https://doi.org/10.1007/s10844-018-0517-4.

Wang, X., He, X., Nie, L., & Chua, T.-S. (2017). Item silk road: Recommending items from information domains to social users. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. SIGIR ’17, pp 185–194. Association for Computing Machinery. https://doi.org/10.1145/3077136.3080771.

Wang, X., Wang, D., Xu, C., He, X., Cao, Y., & Chua, T.-S (2019). Explainable reasoning over knowledge graphs for recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01), 5329–5336. https://doi.org/10.1609/aaai.v33i01.33015329.

Ren, X., Yin, H., Chen, T., Wang, H., Huang, Z., & Zheng, K. (2021). Learning to ask appropriate questions in conversational recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. SIGIR ’21, pp 808–817. Association for Computing Machinery. https://doi.org/10.1145/3404835.3462839.

Hsieh, C.-K., Yang, L., Cui, Y., Lin, T.-Y., Belongie, S., & Estrin, D. (2017). Collaborative metric learning. In *Proceedings of the 26th international conference on world wide Web*. WWW ’17, pp. 193–201. International World Wide Web Conferences Steering Committee. https://doi.org/10.1145/3038912.3052639.

Liu, T., & He, Z. (2022). Dlir: a deep learning-based initialization recommendation algorithm for trust-aware recommendation. Applied Intelligence, pp 1–12. https://doi.org/10.1007/s10489-021-03039-1.

Fu, M., Qu, H., Yi, Z., Lu, L., & Liu, Y. (2019). A novel deep learning-based collaborative filtering model for recommendation system. *IEEE Transactions on Cybernetics*, 49(3), 1084–1096. https://doi.org/10.1109/TCYB.2018.2795041.

Guo, H., Tang, R., Ye, Y., Li, Z., & He, X. (2017). DeepFM: A factorization-machine based neural network for ctr prediction. https://doi.org/10.48550/ARXIV.1703.04247.

He, X., & Chua, T.-S. (2017). Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. SIGIR ’17 pp 355–364. Association for Computing Machinery. https://doi.org/10.1145/3077136.3080777.

Tay, Y. Anh Tuan, L., & Hui, S.C. (2018). Latent relational metric learning via memory-based attention for collaborative ranking. *WWW ’18*, pp. 729–739. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE. https://doi.org/10.1145/3178876.3186154.

Wang, X., He, X., Wang, M., Feng, F., & Chua, T.-S (2019). Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. SIGIR ’19, pp 165–174. Association for Computing Machinery. https://doi.org/10.1145/3331184.3331267.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020). Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. SIGIR ’20, pp 639–648. Association for Computing Machinery. https://doi.org/10.1145/3397271.3401063.

Chen, L., Wu, L., Hong, R., Zhang, K., & Wang, M. (2020). Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01), 27–34. https://doi.org/10.1609/aaai.v34i01.5330.

Song, W., Xiao, Z., Wang, Y., Charlin, L., Zhang, M., & Tang, J. (2019). Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the twelfth ACM international
Feng, C., Liu, Z., Lin, S., & Quek, T.Q.S. (2019). Attention-based graph convolutional network for recommendation system. In ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE. pp 7560–7564. https://doi.org/10.1109/ICASSP.2019.8683050.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., & Yin, D. (2019). Graph neural networks for social recommendation. In The World wide web conference. WWW ’19, pp 417–426. Association for Computing Machinery. https://doi.org/10.1145/3308558.3313488.

Wang, X., He, X., Cao, Y., Liu, M., & Chua, T.-S. (2019). Kgat: Knowledge graph attention network for recommendation. In Proceedings of the 25th ACM SIGKDD International conference on knowledge discovery & data mining. KDD ’19, pp 950–958. Association for Computing Machinery. https://doi.org/10.1145/3292500.3330989.

Hui, B., Zhang, L., Zhou, X., Wen, X., & Nian, Y. (2022). Personalized recommendation system based on knowledge embedding and historical behavior. Applied Intelligence, 52(1), 954–966. https://doi.org/10.1007/s10489-021-02363-w.

Publisher’s note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.