Neural Computing and Applications (2023) 35:2633–2646
https://doi.org/10.1007/s00521-022-07735-y

ORIGINAL ARTICLE

Effective hybrid graph and hypergraph convolution network for collaborative filtering

Xunkai Li[1] · Ronghui Guo[1] · Jianwen Chen[2] · Youpeng Hu[1] · Meixia Qu[1] · Bin Jiang[1]

Received: 22 February 2022 / Accepted: 17 August 2022 / Published online: 3 September 2022

Abstract

In recent years, graph convolution networks and hypergraph convolution networks have become a research hotspot in collaborative filtering (CF) because of their information extraction ability in dealing with the user-item interaction information. In particular, hypergraph can model high-order correlation of users and items to achieve better performance. However, the existing graph-based CF methods for mining interactive information remain incomplete and limit the expressiveness of the model. Moreover, they directly use low-order Chebyshev polynomials to fit the convolution kernel of graph and hypergraph without experimental proof or analysis, lacking interpretability. We propose an effective hybrid graph and hypergraph convolutional network (EHGCN) for CF to obtain a capable and interpretable framework. In EHGCN, the graph and the hypergraph are used to model the correlation among nodes in the interaction graph for multilevel learning. EHGCN also optimizes the information flow framework to match the improved convolution strategy of the graph and hypergraph we proposed. Extensive experiments on four real-world datasets show the considerable improvements of EHGCN over other state-of-the-art methods. Moreover, we analyze the graph and hypergraph convolution kernel in terms of the spectral domain to reveal the core of the graph-based CF, which has a heuristic effect on future work.

Keywords Collaborative filtering · Graph neural network · Hypergraph learning · Recommender systems

1 Introduction

Collaborative filtering (CF) is a key method for constructing personalized recommender systems, which exploits the experience of user groups with similar preferences to recommend items in which users are interested. Specifically, CF uses the corresponding collaborative filtering algorithm to filter the user groups based on the user-item interaction history and generates the recommendation sequences according to their interaction items. As an early model, Matrix Factorization (MF) [1] directly uses the ID of users and items to perform feature mapping. Then, it generates a rating matrix by inner product for recommendations. Given the deep learning development, Neural Collaborative Filtering (NCF) [2] uses multi-layer perceptron (MLP) to obtain the final embeddings, then generates the rating matrix through interaction function. It achieves significant improvement. Therefore, we summarize that the core of CF is to generate effective user and item embeddings that contain the interaction behavior patterns between them.

Xunkai Li and Ronghui Guo have contributed equally and should be regarded as co-first authors.

✉ Bin Jiang
jiangbin@sdu.edu.cn

Xunkai Li
lxk_yb@163.com

Ronghui Guo
guoronghui.grh@gmail.com

Jianwen Chen
cjianwen@126.com

Youpeng Hu
yoooooohu@gmail.com

Meixia Qu
mxqu@sdu.edu.cn

[1] School of Mechanical, Electrical and Information Engineering, Shandong University, No.180, Wenhua West Road, Weihai 264209, Shandong, China

[2] Wendeng Big Data Bureau, No.288, Tianfu Road, Weihai 264499, Shandong, China

Inspired by the Convolutional Neural Network (CNN), Graph Convolution Network (GCN) [3–5] adopts spectral theory to perform the graph convolution operation. The theoretical basis of graph convolution is that convolution in the time domain is equivalent to the dot product in the spectral domain. First, the Fourier Transform is used to transform the graph signals in the time domain and the spectral domain, and then the convolution operation is conducted by parameterizing the convolution kernel. The widely used GCN [5] utilizes the first-order Chebyshev polynomials as the graph convolution kernel to simplify the calculation. In CF, the interactive information between users and items is a bipartite graph, which can be extracted by GCN. Based on this, the graph-based CF methods, such as Neural Graph Collaborative Filtering (NGCF) [6], LightGCN [7], and Linear Residual Graph Convolutional Collaborative Filtering (LR-GCCF) [8], possess remarkable advantages. NGCF applies GCN to the CF task, which is divided into message construction and message aggregation. LightGCN improves the GCN processes to achieve better performance in the recommendation. LR-GCCF uses a new information propagation model Simple Graph Convolution (SGC) [9] and proposes residual links to conduct feature embeddings. However, the existing graph-based CF methods use the first-order Chebyshev polynomials as the filter to conduct the convolution operation by default without experimental proof or analysis. Therefore, there is a lack of interpretability.

As a generalization of the graph, the hypergraph has also attracted extensive attention in recent years. In the view of mathematics, the hyperedges of hypergraphs connect any of the vertices, which can model high-order correlation among nodes. Inspired by the GCN method [5], the hypergraph convolutional network (HGCN) [10] utilizes the unique structured information of the hypergraph to perform the hypergraph convolution in the spectral domain. Compared with the graph, the hypergraph successfully models high-order data correlation (except pair connection) for downstream tasks. Dual channel hypergraph collaborative filtering (DHCF) [11] first applies the hypergraph convolution method to CF, which generates the hypergraph over the user-item interaction matrix. However, it is limited by the complex information propagation framework and the intuitive judgment that defines the hypergraph convolution kernel.

Although the above graph-based CF methods have achieved great performance improvement, the following limitations still exist: (1) insufficient mining of interactive information and (2) intuitively fitting convolution kernel with first-order Chebyshev polynomials. The above problems are attributed to the fact that the existing methods do not analyze the core of the graph-based CF.

In order to solve the above problems, we propose an effective hybrid graph and hypergraph convolution network (EHGCN), focusing on the embedding process. First, we further mine the interaction information by generating the hypergraphs of users (items) to model the high-order correlation among users (items) for multilevel learning. On this basis, the convolution strategy and information flow framework of graph and hypergraph are optimized to improve the model performance. Compared with the baselines, our method shows a considerable improvement in four benchmark datasets. Meanwhile, the core of the graph-based CF is analyzed for the first time from the perspective of signal processing.

In summary, the main contributions of our work are as follows:

- We propose an effective hybrid graph and hypergraph convolution network for CF, which is the first method to mine interactive information from different levels by graph and hypergraph with significant performance improvement.
- We optimize the convolution strategy from the graph signal processing point of view, and the corresponding improved information flow framework called DenseGCN is proposed.
- This is the first time to analyze the core of the graph-based CF in the process of graph and hypergraph convolution. Meanwhile, we draw an interpretable conclusion through large quantities of experiments.

The remainder of our paper is organized as follows. The preliminaries are introduced in Sect. 2. Then, the architecture and details of the proposed model are illustrated in Sect. 3. Moreover, Sect. 4 details the experimental results, and EHGCN is analyzed based on the results. Then, Sect. 5 summarizes the existing literature on recommender systems. Finally, Sect. 6 highlights the conclusion of the article with future guidelines.

2 Preliminaries

2.1 Graph and hypergraph convolution

The theoretical basis of graph convolution is that the product operation in the spectral domain is equal to the convolution operation in the time domain. Based on this, the graph convolution process is divided into graph filtering and feature transformation. Hence, we need to define an appropriate graph filter and a trainable weight matrix $\mathbf{W}$ to support it. Laplacian smoothing filter depicts the trend of signal change in the graph [12], which is often used because of its high computational efficiency and superior performance. It is defined as $\mathbf{H}_f = \mathbf{I} - k_g \mathbf{L}^g_{sym}$, where $\mathbf{I}$

represents identity matrix and $\mathbf{L}^{g}_{sym}$ represents symmetric normalized Laplacian matrix. $\mathbf{X}$ is usually the feature matrix of graph nodes. Adaptive Graph Encoder (AGE) [13] illustrates that k_g is real-valued and can be selected according to some rules. $\mathbf{H}_f$ can be employed as the filter matrix. Then, $\widetilde{\mathbf{X}}$ denotes the signal processed by the filter:

$$\widetilde{\mathbf{X}} = \mathbf{H}_f\mathbf{X} = (\mathbf{I} - k_g\mathbf{L}^{g}_{sym})\mathbf{X} = \mathbf{U}(\mathbf{I} - k_g\Lambda)\mathbf{U}^{-1}\mathbf{X}, \tag{1}$$

where Λ and $\mathbf{U} \in \mathbb{R}^{N\times N}$ represent the diagonal matrix and the unitary matrix comprising the eigenvectors of $\mathbf{L}^{g}_{sym}$, respectively. $\mathbf{I} - k_g\Lambda$ represents a decaying non-negative frequency response function, which is essentially a low-pass filter. AGE denotes the maximum eigenvalue in Λ as λ_{max}. In theory, if $k_g > 1/\lambda_{max}$, the filter is not low-pass in the $(1/k_g, \lambda_{max}]$ interval. Otherwise, if $k_g < 1/\lambda_{max}$, the filter cannot denoise all the high-frequency components. Moreover, $k_g = 1/\lambda_{max}$ represents a low-pass filter. Graph signal processing related works [14] and [15] prove that the signals of input data contain low-frequency and high-frequency signals, which, respectively, reflect the correlation and difference among signals and correspond to the assortative networks and disassortative networks. The symmetric normalized graph Laplacian is defined as:

$$\widetilde{\mathbf{L}}^{g}_{sym} = \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{L}}^{g}\widetilde{\mathbf{D}}^{-\frac{1}{2}} = \mathbf{I} - \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}_g\widetilde{\mathbf{D}}^{-\frac{1}{2}}, \tag{2}$$

where $\widetilde{\mathbf{A}}_g$ represents the adjacency matrix with self-connection, and $\widetilde{\mathbf{D}}$ represents the diagonal matrix of $\widetilde{\mathbf{A}}_g$. SGC [9] uses stacked t filtering operations to construct a low-pass filter with stronger effect. Based on the above derivation process, we illustrate the final generalized graph Laplacian convolutional representation:

$$\begin{aligned}\widetilde{\mathbf{X}} &= (\mathbf{I} - k_g\widetilde{\mathbf{L}}^{g}_{sym})^t\mathbf{X} \\ &= (\mathbf{I} - k_g(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{L}}^{g}\widetilde{\mathbf{D}}^{-\frac{1}{2}}))^t\mathbf{X} \\ &= (\mathbf{I} - k_g(\mathbf{I} - \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}_g\widetilde{\mathbf{D}}^{-\frac{1}{2}}))^t\mathbf{X}.\end{aligned} \tag{3}$$

It should be noted that the feature transformation process and activation function are ignored in the above derivation.

GCN [5] reduces computation complexity by using the first-order Chebyshev polynomials to fit the convolution kernels where $k_g = 1$ and $t = 1$. After the feature transformation process and activation function are added, the widely used graph convolution formula is expressed as:

$$\begin{aligned}\mathbf{X}^{(l+1)} &= \sigma(\widetilde{\mathbf{X}}\mathbf{W}^{(l)}) \\ &= \sigma((\mathbf{I} - \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{L}}^{g}\widetilde{\mathbf{D}}^{-\frac{1}{2}})\mathbf{X}^{(l)}\mathbf{W}^{(l)}) \\ &= \sigma(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}_g\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}^{(l)}\mathbf{W}^{(l)}),\end{aligned} \tag{4}$$

where $\mathbf{W}^{(l)}$ represents the weight matrix of the l-th layer and $\sigma(\cdot)$ is a nonlinear activation function, such as ReLU. Eq. 4 is the convolution strategy of GCN [5].

On the other hand, the hypergraph structure models the high-order correlation of data and expresses complex relationships through hyperedges. The adjacency matrix of a graph corresponds to the incidence matrix of a hypergraph. The incidence matrix of a hypergraph is defined as $\mathbf{H} \in \mathbb{R}^{|V|\times|E|}$, where $|V|$ and $|E|$ represent the number of nodes and the number of hyperedges, respectively. $\mathbf{W}_h$ is the weight matrix of the hyperedges. The degree of nodes and hyperedges are $d(v) = \sum_{e\in E}\mathbf{W}_h(e)\mathbf{H}(v,e)$ and $d(e) = \sum_{v\in V}\mathbf{H}(v,e)$, respectively. $\mathbf{D}_v$ is the diagonal nodes degree matrix, and $\mathbf{D}_e$ is the diagonal hyperedges degree matrix, comprising $d(v)$ and $d(e)$, respectively. The related work of hypergraph [16] derives the symmetric normalized Laplacian matrix of hypergraph $\mathbf{L}^{hg}_{sym} = \mathbf{I} - \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}_h\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-\frac{1}{2}}$ from the spectral domain. Similar to the derivation of generalized graph Laplacian convolution operation, the generalized hypergraph Laplacian convolution operation is defined as:

$$\begin{aligned}\widetilde{\mathbf{X}}_h &= (\mathbf{I} - k_{hg}\mathbf{L}^{hg}_{sym})^t\mathbf{X}_h \\ &= (\mathbf{I} - k_{hg}(\mathbf{I} - \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}_h\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-\frac{1}{2}}))^t\mathbf{X}_h,\end{aligned} \tag{5}$$

where $\mathbf{X}_h$ denotes hypergraph signals, $\widetilde{\mathbf{X}}_h$ represents the hypergraph signals processed by the filters, and k_{hg} represents the weight coefficient of the frequency response function of the hypergraph Laplacian matrix. Note that the feature transformation process and activation function are ignored in the above derivation.

The hypergraph convolution proposed by HGCN [10] is a spectral-domain convolution operation applied to hypergraph structures, where $k_{hg} = 1$ and $t = 1$. After the feature transformation process and activation function are added, the widely used hypergraph convolution formula can be expressed as:

$$\mathbf{X}_h^{(l+1)} = \sigma(\mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}_h\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-\frac{1}{2}}\mathbf{X}_h^{(l)}\mathbf{W}^{(l)}), \tag{6}$$

where $\mathbf{X}_h^{(l+1)}$ and $\mathbf{W}_h^{(l)}$ denote the model output and weight matrix at l-th layer, respectively. Essentially, hypergraph convolution is also an information propagation model. The graph signals referred to below contain the simple graph signals and the hypergraph signals.

2.2 Graph-based CF method

Graph-based CF methods combine the users and items in a shared space. $\mathbf{R} \in \{0,1\}^{|M|\times|N|}$ is a user-item interaction matrix, where $|M|$ is the number of users, and $|N|$ is the number of items. The matrix element value is 1 if an interaction exists between user i and item j; otherwise, it is 0. Then, the adjacency matrix representing the interaction history of users and items is defined as a symmetric matrix:

$$\mathbf{A} = \begin{bmatrix} 0 & \mathbf{R} \\ \mathbf{R}^{\mathbf{T}} & 0 \end{bmatrix}, \mathbf{A} \in \mathbb{R}^{|M+N| \times |M+N|}. \tag{7}$$

$\mathbf{A}$ is used to perform the matrix calculation for user and item embeddings. In implicit feedback-based CF, user and item IDs only have weak semantic information. We obtain embeddings by mining the topological information in the user-item interaction matrix and weak semantic information, which is described as:

$$\begin{aligned} e_u^{(l+1)} &= AGG(e_u^{(l)}, \{e_i^{(l)} : i \in \mathcal{N}_u\}), \\ e_i^{(l+1)} &= AGG(e_i^{(l)}, \{e_u^{(l)} : u \in \mathcal{N}_i\}), \end{aligned} \tag{8}$$

where $AGG(\cdot)$ represents the graph information propagation model, $e_u^{(l)}$ is the user embeddings at the l-th layer, and $e_i^{(l)}$ is the item embeddings at the l-th layer. $\mathcal{N}_u$ and $\mathcal{N}_i$ represent the item i with which user u has interacted and the user u with which item i has interacted, respectively.

The hypergraph uses degree-free hyperedges to encode high-order data correlation and strengthen the topological information in the interaction matrix. We present the definition of hypergraph generation from the perspective of items, and the definition of hypergraph generation based on the perspective of users are similar.

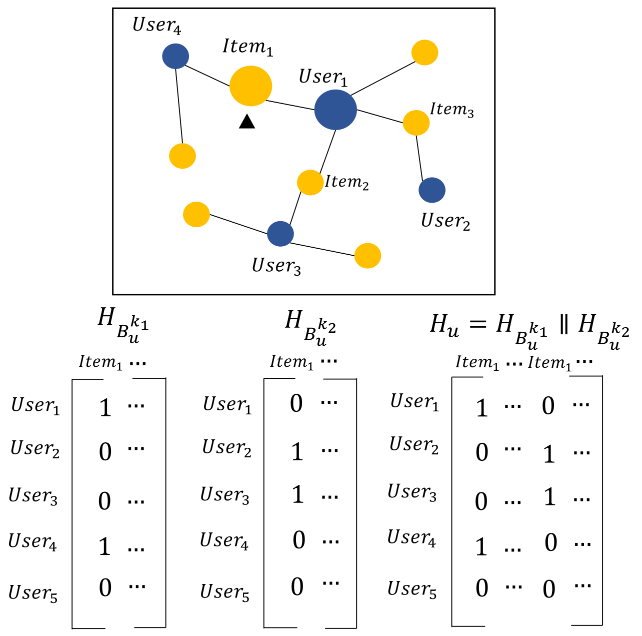
Definition 1 (Item's k-order reachable users.) If a path exists between $user_k$ and $item_j$, and the number of users on that path is equal to k, then $user_k$ is the k-order reachable user of $item_j$.

For example, Fig. 1 shows that the first-order reachable users of $Item_1$ are $User_1$ and $User_4$ because paths "$Item_1$ - $User_1$" and "$Item_1$ - $User_4$", respectively, exist and the number of users on these paths is equal to 1. Similarly, the second-order reachable users of $Item_1$ are $User_3$ and $User_2$ because paths "$item_1$ - $User_1$ - $Item_2$ - $User_3$" and "$Item_1$ - $User_1$ - $Item_3$ - $User_2$", respectively, exists and the number of users on these paths is equal to 2.

On the basis of the above definition, we generate the hypergraph structure data over the interaction matrix. The information propagation model of the hypergraph is defined as:

$$\begin{aligned} e_u^{(l+1)} &= HAGG(e_u^{(l)}, \{e_u^{(l)} : u \in \mathcal{N}_u^k\}), \\ e_i^{(l+1)} &= HAGG(e_i^{(l)}, \{e_i^{(l)} : i \in \mathcal{N}_i^k\}), \end{aligned} \tag{9}$$

where $HAGG(\cdot)$ represents the hypergraph information propagation model, k is the k-order correlation information


$User_4$
$Item_1$
$User_1$
$Item_3$
$Item_2$
$User_2$
$User_3$
▲
$H_{B_u^{k_1}}$
$H_{B_u^{k_2}}$
$H_u = H_{B_u^{k_1}} \parallel H_{B_u^{k_2}}$

$$\begin{array}{c|cc} & Item_1 & \cdots \\ \hline User_1 & 1 & \cdots \\ User_2 & 0 & \cdots \\ User_3 & 0 & \cdots \\ User_4 & 1 & \cdots \\ User_5 & 0 & \cdots \end{array} \quad \begin{array}{c|cc} & Item_1 & \cdots \\ \hline User_1 & 0 & \cdots \\ User_2 & 1 & \cdots \\ User_3 & 1 & \cdots \\ User_4 & 0 & \cdots \\ User_5 & 0 & \cdots \end{array} \quad \begin{array}{c|cccc} & Item_1 & \cdots & Item_1 & \cdots \\ \hline User_1 & 1 & \cdots & 0 & \cdots \\ User_2 & 0 & \cdots & 1 & \cdots \\ User_3 & 0 & \cdots & 1 & \cdots \\ User_4 & 1 & \cdots & 0 & \cdots \\ User_5 & 0 & \cdots & 0 & \cdots \end{array}$$

Fig. 1 Illustration for constructing the user hypergraph incidence matrix with $Item_1$ marked by ▲. User-item interaction matrix is used to encode the high-order relevance of users

from the same type of nodes, and other notations are the same as those in Eq. 8.

3 Method

In this section, we introduce hypergraph generation methods, the EHGCN with generalized graph and hypergraph Laplacian convolution kernel, and the improved information flow framework.

3.1 Hypergraph generation

In this section, the method of generating a hypergraph structure with high-order data correlation based on the user-item interaction matrix is introduced. The above-mentioned user-item interaction matrix is defined as $\mathbf{R} \in \{0,1\}^{|M| \times |N|}$. In the hypergraph, the hyperedges are defined to connect two or more nodes. Therefore, the high-order correlation information between the nodes is extracted by modeling the topology with a hypergraph. For example, we construct the user hypergraph incidence matrix based on the interaction graph and the hyperedge over the k-order reachable user sets of certain items. Then, the high-order hyperedge group $\mathcal{E}_{B_u^k}$ is constructed according to the k-order reachable rule among users. The k-order reachable matrix for the items is described as $\mathbf{A}_i^k \in \{0,1\}^{|N| \times |N|}$, and the hyperedge group of incidence matrix is $\mathbf{H}_{B_u^k} \in \{0,1\}^{|M| \times |N|}$, which is generated by the user's k-order rules. As a hyperedge group of incidence matrix, fusion function $f(\cdot)$ is a connection operation $\cdot||\cdot$. Finally, we generate the user hypergraph correlation matrix $\mathbf{H}_u$. The above calculations are defined as:

$$\mathcal{E}_{B_u^k} = \{B_u^k(i), i \in I\}, \tag{10}$$

$$\mathbf{A_i}^k = min(1, power(\mathbf{H}^T\mathbf{H}, k)), \tag{11}$$

$$\mathbf{H}_{B_u^k} = \mathbf{H}\mathbf{A}_i^{(k-1)}, \tag{12}$$

$$\begin{aligned}\mathbf{H}_u &= f(\mathcal{E}_{B_u^{k_1}}, \mathcal{E}_{B_u^{k_2}}, ..., \mathcal{E}_{B_u^{k_a}}) \\ &= \mathbf{H}_{B_u^{k_1}}||\mathbf{H}_{B_u^{k_2}}||...||\mathbf{H}_{B_u^{k_a}},\end{aligned} \tag{13}$$

where I represents the entity sets, indicating that each user node u must be added to the high-order hyperedge group, $power(\mathbf{M}, k)$ is a function for calculating the k power of the given matrix $\mathbf{M}$, and $\mathbf{H} \in \{0,1\}^{|M|\times|N|}$ denotes the user-item incidence matrix. In the same way, we also perform the corresponding operations on the items. Finally, $\mathbf{H}_u$ and $\mathbf{H}_i$ are obtained to represent the hypergraph correlation matrices of users and items, respectively. We present the construction of the second-order hyperedge incidence matrix as an example. As shown in Fig. 1, this matrix is represented as $\mathbf{H}_u = \mathbf{H}_{B_u^{k_1}}||\mathbf{H}_{B_u^{k_2}}$. In short, the interaction matrix of users and items is illustrated as a special bipartite graph with two types of nodes. Taking the first-order hypergraph of the users as an example, we consider the item nodes as the hyperedge to connect the user nodes in the broad sense. Thus, the item nodes simply constitute the first-order hypergraph of the users based on the items. The hypergraph convolution uses the hyperedge of the item nodes as the medium to construct the high-order relevance among users for model learning.

3.2 Hybrid convolution of graph and hypergraph

In CF, the user and item IDs only own weak semantic information. This scenario indicates that using the initial embedding layer to map the feature space of the IDs is sufficient for the learning of the initial features. The core of the convolution calculation is using graph and hypergraph filters in processing graph signals to obtain meaningful embeddings, which is considered complex interaction pattern between users and items in CF. Therefore, we optimize the information flow framework to match the filter operation, which can be expressed as:

$$\begin{aligned}\mathcal{G}_{l+1} &= \mathcal{D}(\mathcal{G}_l, \mathcal{L}^g, \mathcal{L}^{hg}) \\ &= \mathcal{T}(\mathcal{F}(\mathcal{G}_l, \mathcal{L}^g, \mathcal{L}^{hg}), \mathcal{G}_{l-1}) \\ &= \mathcal{T}(\mathcal{F}(\mathcal{G}_l, \mathcal{L}^g, \mathcal{L}^{hg}), ..., \mathcal{F}(\mathcal{G}_0, \mathcal{L}^g, \mathcal{L}^{hg}), \mathcal{G}_0),\end{aligned} \tag{14}$$

where $\mathcal{D}(\cdot)$ is the DenseGCN convolution operation and is divided into $\mathcal{F}(\cdot)$ and $\mathcal{T}(\cdot)$. $\mathcal{F}(\cdot)$ is the information propagation function, which represents the use of generalized graph and hypergraph Laplacian filters $\mathcal{L}^g$ and $\mathcal{L}^{hg}$ to process the graph signals. The generalized graph and hypergraph Laplacian convolution kernel is used to achieve robust learning and filter the invalid signals. $\mathcal{G}_l$ denotes the graph information at l-th layer. $\mathcal{T}(\cdot)$ is a vertex cascading function, which intensively fuses the input graph $\mathcal{G}_0$ with all intermediate GCN layers. Thus, $\mathcal{G}_{l+1}$ contains the results of all GCN outputs from the previous layer.

According to LightGCN and SGC, we conclude that the self-loop connection is essentially equivalent to a weighted sum of the embeddings propagated at every EHGCN layer, which is also a part of DenseGCN. Inspired by this finding, we propose EHGCN, which contains the optimized convolution strategy and information flow framework. The final representation is as follows:

$$\begin{aligned}e_u^{(l+1)} &= \mathcal{D}(\sum_{i\in\mathcal{N}_u}\sum_{u\in\mathcal{N}_u^k}((\mathbf{I} - k_g\mathbf{L}_{sym}^g)e_i^{(l)} + (\mathbf{I} - k_{hg}\mathbf{L}_{sym}^{hg})e_u^{(l)})), \\ e_i^{(l+1)} &= \mathcal{D}(\sum_{u\in\mathcal{N}_i}\sum_{i\in\mathcal{N}_i^k}((\mathbf{I} - k_g\mathbf{L}_{sym}^g)e_u^{(l)} + (\mathbf{I} - k_{hg}\mathbf{L}_{sym}^{hg})e_i^{(l)})),\end{aligned} \tag{15}$$

where $e_u^{(l+1)}$ and $e_i^{(l+1)}$ represent the user and item embeddings at the l-th layer, and $\mathcal{N}_u(\mathcal{N}_i)$ denotes the sets of items (users) that interact with user u (item i). $\mathcal{N}_u^k$ and $\mathcal{N}_i^k$ represent the same type neighbor nodes of the k-order of the user nodes and the item nodes, respectively. $\mathbf{L}_{sym}^h$ and $\mathbf{L}_{sym}^{hg}$ represent the normalized graph and hypergraph Laplacian filter, respectively. In the definition of the propagation model, we eliminate the meaningless parameter matrix $\mathbf{W}$ and the redundant nonlinear activation function $\sigma(\cdot)$. The learning strategy of our model is to learn the features of weak semantic information through the initial embedding layer. We hope to use the filters to mine the implicit association in the data based on the graph topology, that is, the interaction mode between users and items.

In summary, our model uses an optimized hybrid graph and hypergraph information propagation framework to conduct multilevel learning from the perspective of signal processing. The model framework can be referred to Fig. 2. With one user as an example, the graph is used to model the topology information of the adjacent items, whereas the hypergraph is used to model the topology information of the high-order adjacent users. High-order connections are established for users and items by constructing hypergraph connections, respectively. Thus, each node feature update can obtain information not only from adjacent nodes but

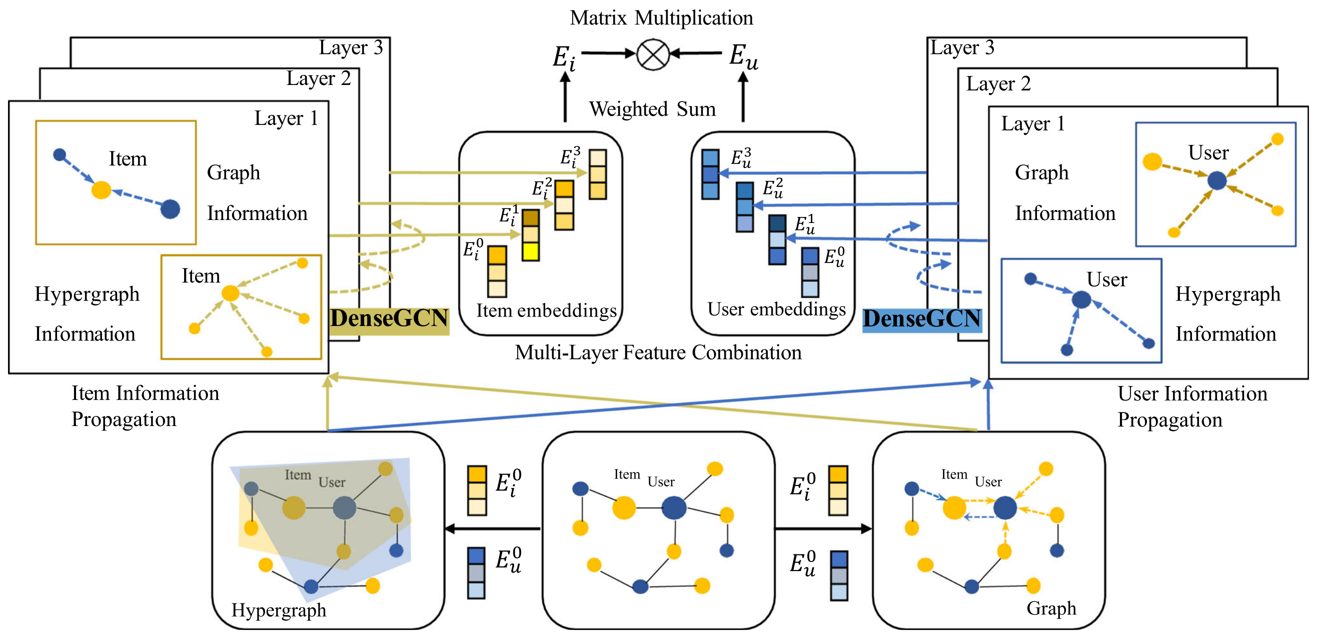


Fig. 2 Framework overview of EHGCN, where the weighted sum module representing the output of every layer is concatenated to obtain its average value as the final user and item embeddings. E_u^n and E_i^n represent the embedding result of the n-th layer. E_u and E_i represent the final embeddings output of the model

also from super-edge nodes, allowing the full mining of interactive information from different levels by the graph and hypergraph. Moreover, we optimize convolution theory and framework to process graph signals, which can also be considered as a signal enhancement approach. Finally, we sum up the output results of every layer and average the final user and item embeddings. The embeddings of users and items are learned by EHGCN and can be expressed as:

$$E_u = \frac{1}{L+1}\sum_{l=0}^{L} e_u^{(l)}, \quad E_i = \frac{1}{L+1}\sum_{l=0}^{L} e_i^{(l)}, \tag{16}$$

where L is the number of convolution layers. We finally implement the recommendations according to the score matrix $\hat{y}$ to verify the ability of EHGCN through the simple inner product of user embeddings E_u and item embeddings E_i:

$$\hat{y} = E_u^T E_i. \tag{17}$$

3.3 Optimization

In our proposed model, we only require to train the weight matrix of the initial embedding layer, which transforms the initial feature into the specified dimension. We choose Bayesian Personalized Ranking (BPR) [17] as the loss function. BPR mainly adopts the implicit feedback of users to rank items by the maximum posterior probability and then generates a recommendation. The $\hat{y}$ from the inner product is used as implicit user feedback. The positive and negative sample pairs are obtained through the negative sampling strategy to accelerate the model training. Therefore, our loss function is described as:

$$\mathcal{L}_{BPR} = \sum_{(u,i,j)\in\mathcal{O}} (-ln\sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda||\Theta^{(0)}||_2^2), \tag{18}$$

where $\mathcal{O} = \{(u,i,j)|(u,i) \in \mathcal{R}^+, (u,j) \in \mathcal{R}^-\}$, and $\mathcal{R}^+$ and $\mathcal{R}^-$ represent the positive and negative samples obtained by negative sampling, respectively. λ represents the regularization parameter, and $||\Theta^{(0)}||_2^2$ is a regularization term. The only trainable parameter of our model is the initial feature mapping; thus, our regularization term only applies to the first layer to prevent overfitting. In the process of model training, we use the Adam [18] optimizer for gradient descent and the appropriate learning rate for different datasets.

4 Experiments

We perform our model on four benchmark datasets and compare it with the latest models. The experimental data are the best result of the methods in the test process. Moreover, we further design the following questions to investigate the effectiveness of EHGCN, explain the core

Table 1 Details of the datasets used in the experiment

Dataset	User	Item	Interactions	Denisty
AToy	3122	33950	81,785	0.00077
AMusic	1733	12926	44,802	0.00200
LastFM	1892	4489	52,658	0.00620
ML-1M	6040	3706	994,169	0.04441

of the graph-based CF task, and explain the principle of the method:

- **RQ1**: What is the core of the graph-based CF from the perspective of signal processing?
- **RQ2**: How can EHGCN build an efficient and sufficient information propagation model?
- **RQ3**: What does the number of EHGCN layers mean in the generalized convolution process of graph and hypergraph?
- **RQ4**: How does hypergraph work efficiently in graph signal processing?

4.1 Experiments settings

4.1.1 Datasets

The benchmark datasets AToy, AMusic, LastFM, and ML-1M are utilized as our experiment datasets; LastFM is the same as the LightGCN used in the present study, whereas the AMusic, AToy, and ML-1M datasets we use are provided by DeepCF [19]. On the basis of these datasets, we perform random extraction operations to obtain sparse interaction matrices and verify the validity of our model. The details of the datasets are shown in Table 1.

4.1.2 Baselines and evaluation metrics

The models we compared, including MultiVAE [20][1], NGCF [6][2], LR-GCCF [8][3], LightGCN [7][4] and Low-pass Collaborative Filter Network (LCFN) [21][5], are the methods with the best performance in recent years. DHCF [11] is one of the hypergraph works for CF. However, the implementation method codes and data are not open sources. The details of other compared models are as follows:

[1] https://github.com/makgyver/rectorch.

[2] https://github.com/huangtinglin/NGCF-PyTorch.

[3] https://github.com/newlei/LR-GCCF.

[4] https://github.com/gusye1234/LightGCN-PyTorch.

[5] https://github.com/Wenhui-Yu/LCFN.

- **MultiVAE**: MultiVAE applies the variational autoencoder (VAE) to CF. It is expected to use the generative model in simulating the interaction mode of users and items full of various possibilities. However, the performance of the model is limited.
- **NGCF**: The information propagation in NGCF requires message construction and message propagation to conduct embeddings. The model is limited by the complex information propagation process and high computational complexity.
- **LR-GCCF**: LR-GCCF applies SGC [9] and residual links for CF. However, it does not mine the topological relationship based on the interaction matrix of users and items.
- **LightGCN**: LightGCN constructs a light GCN framework for CF. However, the sparsity of the interaction matrix confines the first-order associated entities and leads to ineffective data utilization.
- **LCFN**: LCFN analyzes the frequency information in the user-item interaction matrix under the condition of adding the artificial noise and obtains the trainable convolution kernel through multiple matrix decomposition. However, the model is limited by the high computational complexity.

We use two evaluation metrics commonly adopted in Top-N recommendation issues: recall@N and ndcg@N. Both are generated by the rating matrix, and the principles are as follows: (1) The results of a high correlation degree have more influence on the final index score than those of a general correlation degree. (2) The index is high because the results with a high correlation degree appear at a higher position. In our experiment, N is set as 20 to evaluate the comprehensive ability of the models.

4.1.3 Parameters settings

We perform our method EHGCN in the PyTorch framework. In the experiments, our hyperparameters include L_2 regularization parameter weight λ, the number of model layers N, the retention probability α for trimming edges when generating hypergraphs, and the parameters of the frequency response function of graph and hypergraph filters k_g and k_{hg}. To ensure the fairness of model performance comparison, the user and item embeddings dimensions are set to 64. The L_2 regularization hyperparameter settings are as follows: $[1e-1, 1e-2, 1e-3, 1e-4, 1e-5]$. Our experimental results of the regularization parameter are shown in Fig. 3, and the best choice of the regularization parameter is $\lambda = 1e-4$. We use the Xavier [22] method to initialize the embedded parameters, whereas Adam [18] is used to optimize the

EHGCN. The size of the mini-batch for all datasets is 1024, and the experimental result is shown in Table 2.

4.2 Model analysis

According to Table 2, our method shows obvious advantages in most cases. The existing graph-based CF methods are confined to one single graph modeling approach, which leads to incomplete information mining and thus impacts the expressiveness of the model. We use hybrid graph and hypergraph convolution to learn node features, and mine the interactive information from different levels by the graph and hypergraph. The graph models the pairs of nodes, whereas the hypergraph establishes the high-order connectivity to mine additional information. Therefore, better node representations are learned than baselines. It should be emphasized that the ML-1M dataset is an extreme case with a dense user-item interaction matrix, and LCFN adopts matrix decomposition to generate the embeddings of users and items. Thus, the initial embeddings with richer semantics are obtained from the dense interaction matrix. However, this phenomenon is not general, as shown by the performance of LCFN on other datasets. LCFN has limited expression ability in the case of minimal interactive information. Moreover, the computational speed of the model decreases while dealing with dense interactive data. Compared with other baseline methods, the performance of EHGCN on the ML-1M dataset is significantly improved, and our method is applicable to both sparse and dense datasets. All of these demonstrate that EHGCN has good scalability. Meanwhile, due to the extreme distribution of data in AToy, the existing baseline methods do not learn it well, but our method still shows superior performance. The model is further analyzed in detail below.

4.3 Discussion of graph and hypergraph Laplacian filters (RQ1)

In the graph-based CF, there is no analysis work from the perspective of graph and hypergraph convolution kernel. Most of the existing methods use graph and hypergraph Laplacian convolution kernel by default which is essentially a low-pass filter for processing graph signals and then generate the embeddings of users and items to obtain the final recommendations. In this section, we aim to explore how to perform efficient graph and hypergraph convolution in CF from the perspective of signal processing. In Fig. 4, K-G and K-G-Enhanced stands for $\mathbf{I} - k_g \mathbf{L}^g_{sym}$ and $(\mathbf{I} - k_g \mathbf{L}^g_{sym})^2$, respectively. K-HG and K-HG-Enhanced are similar. We use four generalized graph Laplacian filters to analyze the properties of the graph signals and then draw the conclusion of scalability about the graph-based CF.

Analyzing the properties of the graph signals is particularly important. According to Frequency Adaptation Graph Convolutional Networks (FAGCN) [15], the low-frequency and the high-frequency information reflect the similarity and difference, respectively, among the graph nodes signals, which correspond to the assortative and disassortative networks. Moreover, both are important to the embedding process. Note that LCFN [21] proposes a trainable convolution kernel, which is also a low-pass filter in essence. In the information propagation framework of LCFN, the interference signal is added artificially, and the results of real-time matrix decomposition are embedded as users and items. We find that LCFN is fundamentally different from our work because its framework is not illustrated by the general graph-based CF. From the previous works, the learnable filters need to be performed by the attention mechanism or matrix factorization. In summary, both of the above methods require a large number of parameters and extra computational costs. The core remains to be discussed. Our main focus is on exploring the core of graph-based CF through the filters and leaving the way to obtain adaptive filters for future work.

Before analyzing the experimental results, we first explain the reasons for using the four filters and the intuitive analysis of graph and hypergraph signals. In the graph signal process, the amplitude may be negative as the value of the frequency response function parameter changes, resulting in a negative effect. As enhanced graph and hypergraph filters, K-G-Enhanced and K-HG-Enhanced solve the above problems by keeping the amplitude positive, thereby providing a greater value for the low-frequency signals and high-frequency signals, respectively. Meanwhile, they provide more alternatives in the signal process. In the user-item interaction matrix, we consider it as a simple heterogeneous graph with two types of nodes. Therefore, the graph signals are often complex because they contain not only high-frequency information with differences between two types of nodes but also low-frequency information with a high correlation that is directly connected with the nodes. Conversely, for the hypergraph signals, the difference information between nodes has been taken into account in the construction of the hypergraph, so it only contains the low-frequency information of the direct correlation between the hyperedge vertices.

Figure 4 shows our experimental results and verifies our conclusion. According to the experimental results, the enhanced graph and hypergraph Laplacian filters alleviate the impact of negative amplitude, and their relatively stable property can be used as the basis for the follow-up work. In the experiments, it is a low-pass filter when $k = 1$, a bandpass filter when $k \in [-1, 1)$, and a high-pass filter

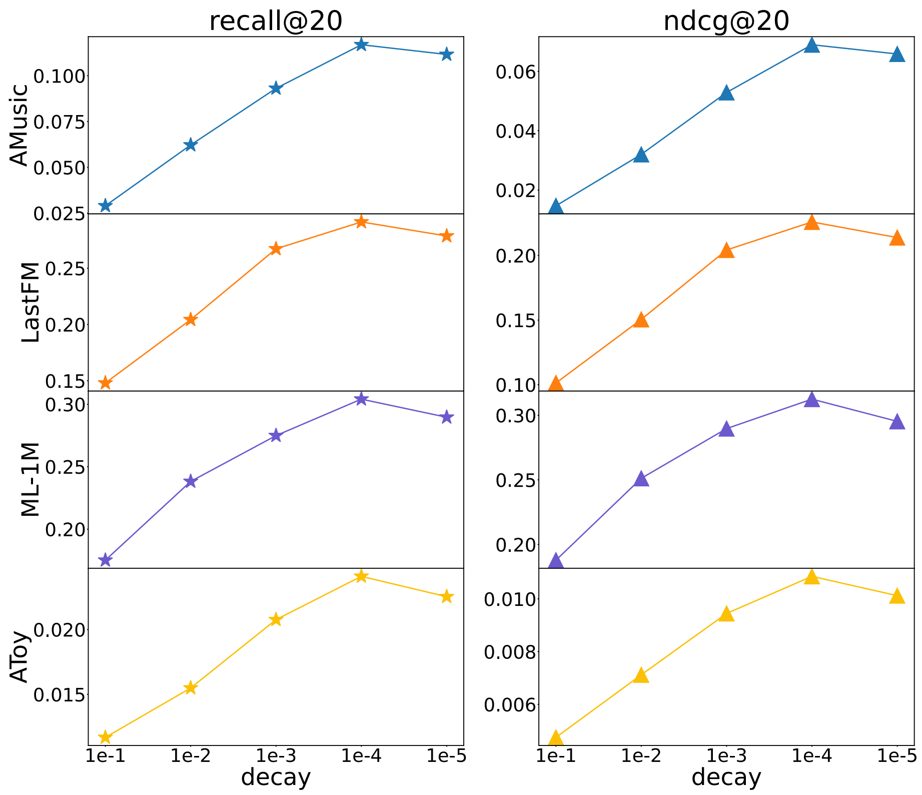


Fig. 3 The L_2 regularization parameter weight λ of EHGCN on AMusic, LastFM, AToy and ML-1M datasets

Table 2 Experimental results of different models on AMusic, LastFM, AToy, and ML-1M

Dataset	LastFM		AMusic		AToy		ML-1M	
Model	recall@20	ndcg@20	recall@20	ndcg@20	recall@20	ndcg@20	recall@20	ndcg@20
MultiVAE	0.2384	0.1772	0.0853	0.0483	0.0142	0.0068	0.2713	0.2617
NGCF	0.2329	0.1823	0.0649	0.0335	0.0170	0.0075	0.2750	0.2806
LightGCN	<u>0.2749</u>	<u>0.2138</u>	<u>0.1006</u>	<u>0.0607</u>	0.0189	0.0080	0.2868	0.2921
LR-GCCF	0.1519	0.0961	0.0989	0.0596	<u>0.0227</u>	<u>0.0098</u>	0.2793	0.2651
LCFN	0.1868	0.1162	0.0919	0.0539	0.0145	0.0075	**0.3096**	**0.3184**
EHGCN	**0.2911**	**0.2265**	**0.1180**	**0.0682**	**0.0243**	**0.0109**	<u>0.3041</u>	<u>0.3125</u>

The best results of the model are shown in bold, and the second-best result is underlined. The evaluation metrics are recall@20 and ndcg@20 by default

when $k \in (1, 3]$. We find that the model shows the best performance when $k_g = 1$ and $k_{hg} = 1$ in most cases, and the two kinds of filters are low-pass filters. Therefore, our conclusion is as follows. For the hypergraph signals, low-pass filters have obvious advantages because the nodes that connect in the hypergraph belong to the same type and have a high association for the existence of hyperedges. For the graph signals, the low-frequency information is as important as the high-frequency information. The low-frequency signals reflect the direct interaction information between the users and the items. The high-frequency signals reflect the interactive correlation between the users and the items, which have a high value in the following situation: if the user u interacts with the item i and the final rating matrix y_{ui} is a large value. On the contrary, if user u and item i interact but y_{ui} in the final rating matrix is small, then the value of the high-frequency signal is small. Hence, for some special datasets or specific task requirements, it is necessary to optimize the filter by adjusting the parameters of the frequency response function to achieve the best performance, rather than directly using a low-pass filter.

4.4 Ablation experiments for EHGCN (RQ2)

We design the ablation experiments on EHGCN to analyze each module and answer **RQ2**. The experimental results

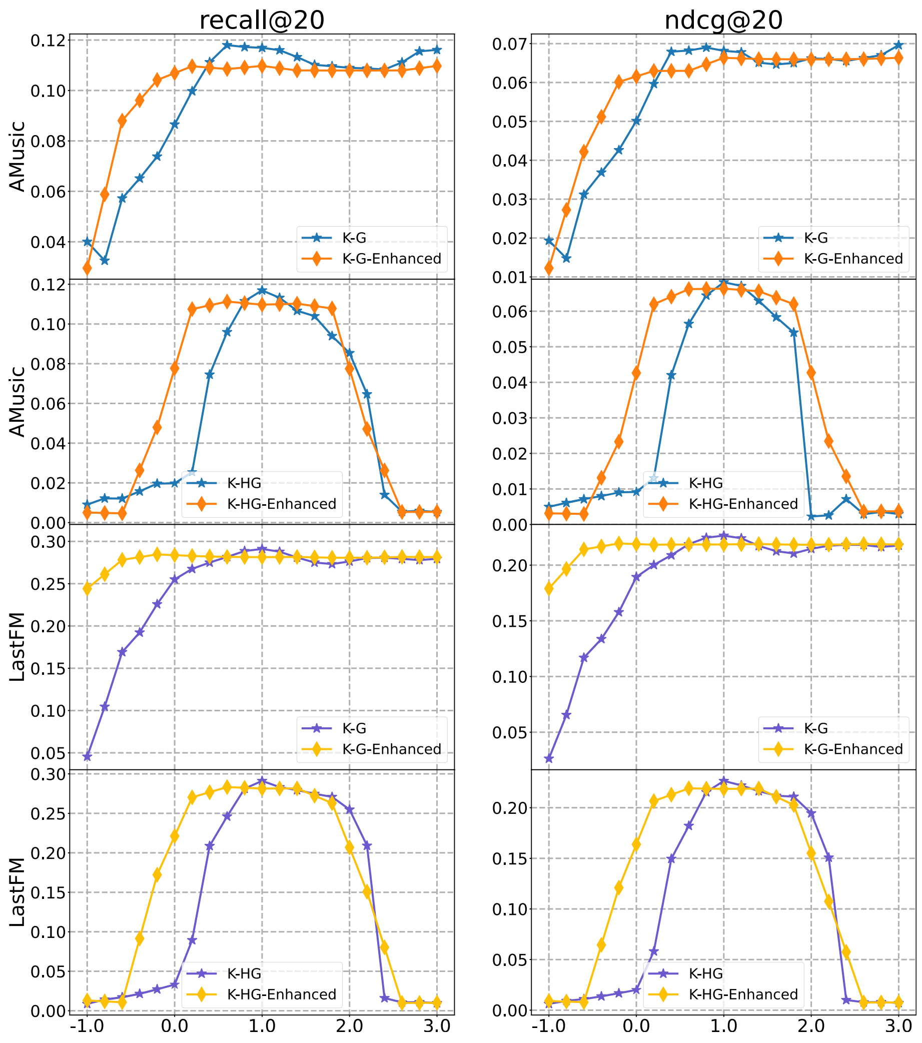


Fig. 4 Experimental results of Laplacian filters and enhanced filters for the graph and the hypergraph on AMusic and LastFM datasets. (The results on AToy and ML-1M, which are omitted for space reasons, show the same trend)

are shown in Table 3. The baseline represents a graph convolution paradigm composed of feature transformation and information aggregation without any optimization methods. The representative methods are GCN [5] and HGCN [10].

EHGCN* indicates that we use the generalized Laplacian convolutional kernel and an optimized information propagation framework to perform learning. However, due to the application of hybrid graph and hypergraph convolution, its computational complexity increases. EHGCN*(Hypergraph, W, A) uses the generalized information propagation model after using the hypergraph structure. We implement multilevel learning by the graph and hypergraph, containing generalized Laplacian filters and an improved information flow framework. Moreover, EHGCN*(Hypergraph, A) and EHGCN*(Hypergarph) represent the efficient convolution approach of removing redundant feature transformation and simultaneously removing feature transformation and nonlinear activation function during the convolution of graphs and hypergraphs, respectively. From EHGCN*(Hypergraph, W, A) to EHGCN*(Hypergraph, A), the number of parameters and the amount of computation are reduced; from EHGCN*(Hypergraph, A) to EHGCN*(Hypergarph), the amount of computation is reduced again. In this scenario, the computational burden is reduced and the efficiency of information propagation is increased. Ultimately, there are no extra parameters in the convolution of graph and hypergraph, so the increase of the model computation is limited. The data distribution of the AToy dataset is extreme in Table 1, so we choose the result of loss convergence to express. Different from the baselines, our model constructs hypergraph structure to establish high-order connectivity in users and items respectively, enabling the direct feature transformation and information aggregation between nodes of the same type. Finally, improved node representations can be obtained through hybrid convolution of the graph and hypergraph. As can be seen from Table 3, with the introduction of hypergraph structure, the performance of the model is significantly improved, undoubtedly verifying the effectiveness of EHGCN. The generalized Laplacian smoothing filter is used to process more complex graph signals, and the improved information flow framework matches our information propagation model to improve the performance. Meanwhile, we eliminate the feature transformation process and nonlinear activation function of learning weak semantic information to focus on mining the topological relationship between data. The results show that our methods further improve the performance of the model. They also prove that the weak semantic information is learned sufficiently by the trainable parameters of the initial embedding layer to conduct the transformation of feature space.

4.5 EHGCN layers analysis (RQ3)

EHGCN layers represent the number of calculations that the filters act on the graph signals. Fitting the convolution kernel with first-order Chebyshev polynomials is essentially a low-pass filter, which inevitably leads to the over-smoothing problem with the increase of model layers. However, we choose the generalized Laplacian graph and hypergraph filters in EHGCN and use the DenseGCN framework as a signal enhancement method to optimize the information flow. All of these methods effectively improve the performance of the model. Meanwhile, no extra parameters exist in the convolution process of graph and hypergraph, so the increase in the computational cost of the model is limited. The experimental results are represented in Table 4.

The generalized graph and hypergraph filters are presented in the functional normal forms Eqs. 3 and 5. The effect of order t is the same as that of the layers; thus, $t = 1$ is adopted by default. From the experimental results, we draw a conclusion that the generalized Laplacian filters and DenseGCN are effective for signal processing to improve the model effect and alleviate the signal failure phenomenon, such as the over-smoothing problem.

4.6 Efficient application of hypergraph (RQ4)

We use users (items) as hyperedges to generate the hypergraph based on items (users) to model the high-order relationship of data. Although there are no extra parameters in the convolution of graphs and hypergraphs, this method undoubtedly increases the computational and spatial complexity of the model. We eliminate the values of the hypergraph incidence matrix in the process of generating hypergraph to alleviate this problem and improve the robustness of the model. Moreover, the high-order incidence matrix constructed from the interaction data is incomplete and noisy, so the value of the incidence matrix of the hypergraph is eliminated in the process of generating the hypergraph to reduce the influence of noise information. The experimental results show that this method effectively improves the performance of the model while reducing computational complexity and spatial complexity. The experimental results are shown in Table 5.

Our conclusion is that if the interaction matrix is dense, the edge retention probabilities need to be set to a large value when the hypergraph is generated. The reason is that the topology information of the data is abundant, and hypergraph provides valuable information, which is conducive to the model mining information from the interaction matrix. On the contrary, if the interactive information is scarce, the construction of a complete hypergraph may generate a large amount of noise information. This scenario produces a negative impact on the interactive information data and affects the performance of the model.

5 Related work

5.1 Collaborative filtering

MF [1] has been widely applied to recommender systems because it was proposed to solve the problem of predicting unknown ratings by an incomplete rating matrix, which evolves from the singular value decomposition (SVD) [23] algorithm. Consequently, the subsequent variants are proposed in [24, 25]. These methods optimize the matrix factorization algorithm by introducing additional information or adapting the learning rate. However, the expression ability of the above methods is limited.

Table 3 Ablation experiments results for EHGCN

	LastFM		AMusic		AToy		ML-1M	
	recall@20	ndcg@20	recall@20	ndcg@20	recall@20	ndcg@20	recall@20	ndcg@20
Baseline	0.1793	0.1203	0.0546	0.0268	0.0109	0.0043	0.2008	0.2098
EHGCN*(Hypergraph, W, A)	0.2103	0.1505	0.0613	0.0318	0.0167	0.0069	0.2393	0.2475
EHGCN*(Hypergraph, A)	0.2771	0.2109	0.1048	0.0594	0.0211	0.0091	0.3029	0.3115
EHGCN*(Hypergarph)	0.2911	0.2265	0.1180	0.0682	0.0243	0.0109	0.3041	0.3126

Table 4 Experimental results of different EHGCN layers in ML-1M and AToy

Dataset	ML-1M		AToy	
Layers	recall@20	ndcg@20	recall@20	ndcg@20
EHGCN-2	0.2903	0.2985	0.0227	0.0104
EHGCN-4	0.2998	0.3083	0.0234	0.0102
EHGCN-6	**0.3041**	**0.3125**	0.0235	0.0104
EHGCN-8	0.2995	0.3088	**0.0243**	**0.0109**
EHGCN-10	0.2964	0.3052	0.0240	0.0107

The best performance is shown in bold. (The results on AMusic and LastFM, which are omitted for space reasons, show the same trend)

Table 5 Experimental results of different edges retention probabilities α on LastFM, AMusic, AToy, and ML-1M

Dataset	LastFM	AMusic	AToy	ML-1M
HyperGraph	recall@20	recall@20	recall@20	recall@20
EHGCN(0.1)	0.2885	0.1164	0.0233	0.3029
EHGCN(0.3)	**0.2911**	0.1159	**0.0243**	0.3032
EHGCN(0.5)	0.2890	**0.1180**	0.0239	0.3030
EHGCN(0.7)	0.2850	0.1158	0.0236	**0.3041**
EHGCN(0.9)	0.2797	0.1160	0.0232	0.3030

The evaluation metrics are recall@20. The best performance is shown in bold

In recent years, deep learning has achieved the best performance in various fields. Thus, some CF methods based on it have been proposed. AutoRec [26] uses an auto-encoder to learn user and item embeddings, and the training purpose is to minimize the difference between the constructed rating matrix and the original rating matrix. The Collaborative Denoising Auto-Encoder (CDAE) [27] proposes the ID numbers as an additional feature to train a denoising auto-encoder based on AutoRec. MultiVAE [20] models the probability distribution of interaction mode through the variational auto-encoder to predict the scoring matrix. NCF [2] is a general framework expressing and generalizing matrix factorization which uses MLP to learn user-item interaction functions. DeepCF [19] trains an extensive network comprising stacked MLP to conduct CF. The advantage of DeepCF is the construction of an MLP as a complex matching function that calculates feature embeddings to generate the final scoring matrix. Outer Product-based Neural Collaborative Filtering (ONCF) [28] uses the CNN to model the interaction relationship and learn the score prediction function considered as another matching function.

5.2 Graph in recommendation

GCN and HGCN have been widely studied in recent years due to their strong ability to express non-Euclidean data. Some optimization methods related to GCN [9, 29–31] are proposed to perform graph convolution effectively, and the hypergraph has recently attracted extensive attention. For example, SGC [9] eliminates the nonlinear calculation among GCN layers and folds the resulting function into a linear transformation to reduce the additional complexity brought by GCN. The results show that SGC is equivalent to a fixed low-pass filter and a linear classifier. Based on these models, graph structures have been widely used in various studies. In particular, GCN and HGCN are widely used in clustering tasks [32–35], text classification [36–38], and computer vision [39–41].

In the recommender systems, we abstract the interactions among users and items into a bipartite graph. Currently, graph and hypergraph are two widely graph structures. GCN [5] and HGCN [10] define the widely recognized information propagation models of the graph and hypergraph, respectively. Graph Convolutional Matrix Completion (GC-MC) [42] and STAcked and Reconstructed Graph Convolutional Networks (STAR-GCN) [43] use the graph auto-encoder to transform the rating prediction into a link prediction problem and aim to restore the original rating matrix as much as possible. PinSage [44], NGCF [6], LR-GCCF [8], and LightGCN [7] use GCN to learn the topological relations from neighbors and generate the user and item embeddings. Finally, matrix multiplication computes the prediction of the rating matrix. DHCF [11] conducts embeddings by modeling the high-order correlation of the interaction matrix through the hypergraph and generates a prediction score matrix.

6 Conclusion and future work

In this work, we propose a hybrid graph and hypergraph convolution framework for the graph-based CF. EHGCN performs data mining and multilevel learning from the hypergraph, optimizes the convolution kernel and information flow framework from the perspective of graph signal processing, and redefines the graph-based CF information propagation framework. The experiments show that our method achieves state-of-the-art performance. Moreover, we use the generalized graph and hypergraph Laplacian filters for the first time to analyze the core of the graph-based CF and draw reliable conclusions on large quantities of experiments, thereby contributing to the follow-up research works.

In the future, we are committed to finding other efficient hypergraph structure generation methods to reduce the computational cost further and improve the scalability of the model. We also hope to find an adaptive convolution kernel optimization method to optimize the performance of the model further while reducing the computational cost.

Acknowledgements This paper was supported by the Shandong Provincial Natural Science Foundation (ZR2020MA064).

Data availability LastFM dataset is available at https://github.com/gusye1234/LightGCN-PyTorch. AMusic, AToy, and ML-1M datasets are available at https://github.com/familyld/DeepCF.

Code availability The source code of EHGCN will be published in https://github.com/RonghuiGuo/EHGCN after paper acceptance for publication.

Declarations

Conflict of interest The authors declare that we have no conflict of interest.

References

1. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. IEEE Comput 42(8):30–37
2. He X, Liao L, Zhang H et al (2017) Neural collaborative filtering. In: WWW '17 proceedings of the 26th international conference on world wide web, pp 173–182
3. Bruna J, Zaremba W, Szlam A et al (2014) Spectral networks and locally connected networks on graphs. In: ICLR 2014: international conference on learning representations (ICLR) 2014
4. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: NIPS'16 proceedings of the 30th international conference on neural information processing systems, pp 3844–3852
5. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. In: ICLR (Poster)
6. Wang X, He X, Wang M et al (2019) Neural graph collaborative filtering. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, pp 165–174
7. He X, Deng K, Wang X et al (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, pp 639–648
8. Chen L, Wu L, Hong R et al (2020) Revisiting graph based collaborative filtering: a linear residual graph convolutional network approach. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 27–34
9. Wu F, Souza AH, Zhang T et al (2019) Simplifying graph convolutional networks. In: International conference on machine learning, pp 6861–6871
10. Feng Y, You H, Zhang Z et al (2019) Hypergraph neural networks. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 3558–3565
11. Ji S, Feng Y, Ji R et al (2020) Dual channel hypergraph collaborative filtering. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, pp 2020–2029
12. Taubin G (1999) A signal processing approach to fair surface design. ACM
13. Cui G, Zhou J, Yang C et al (2020) Adaptive graph encoder for attributed graph embedding. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, pp 976–985
14. Nt H, Maehara T (2019) Revisiting graph neural networks: all we have is low-pass filters. arXiv preprint arXiv:1905.09550
15. Bo D, Wang X, Shi C et al (2021) Beyond low-frequency information in graph convolutional networks. In: AAAI. AAAI Press
16. Zhou D, Huang J, Schölkopf B (2006) Learning with hypergraphs: clustering, classification, and embedding. Adv Neural Inf Process Syst 19:1601–1608
17. Rendle S, Freudenthaler C, Gantner Z et al (2009) Bpr: Bayesian personalized ranking from implicit feedback. In: UAI '09 proceedings of the twenty-fifth conference on uncertainty in artificial intelligence, pp 452–461
18. Kingma DP, Ba JL (2015) Adam: a method for stochastic optimization. In: ICLR 2015: international conference on learning representations 2015
19. Deng ZH, Huang L, Wang CD et al (2019) Deepcf: a unified framework of representation learning and matching function learning in recommender system. In: Proceedings of the AAAI Conference on artificial intelligence, vol 33, pp 61–68
20. Liang D, Krishnan RG, Hoffman MD et al (2018) Variational autoencoders for collaborative filtering. In: Proceedings of the 2018 world wide web conference, pp 689–698
21. Yu W, Qin Z (2020) Graph convolutional network for recommendation with low-pass collaborative filters. In: ICML 2020: 37th international conference on machine learning, pp 10,936–10,945
22. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. J Mach Learn Res 9:249–256
23. Golub GH (1970) Singular value decomposition and least squares solutions. Numer Math 14(5):403–420
24. Luo X, Xia Y, Zhu Q (2013) Applying the learning rate adaptation to the matrix factorization based collaborative filtering. Knowl Based Syst 37:154–164
25. Liu J, Wu C, Liu W (2013) Bayesian probabilistic matrix factorization with social relations and item contents for recommendation. Decis Support Syst 55(3):838–850
26. Sedhain S, Menon AK, Sanner S et al (2015) Autorec: autoencoders meet collaborative filtering. In: International conference on world wide web
27. Wu Y, Dubois C, Zheng AX et al (2016) Collaborative denoising auto-encoders for top-n recommender systems. In: The ninth ACM international conference
28. He X, Du X, Wang X et al (2018) Outer product-based neural collaborative filtering. In: Twenty-seventh international joint conference on artificial intelligence IJCAI-18
29. Chen M, Wei Z, Huang Z et al (2020) Simple and deep graph convolutional networks. In: ICML 2020: 37th international conference on machine learning, pp 1725–1735
30. Li G, Muller M, Thabet A et al (2019) Deepgcns: can gcns go as deep as cnns?. In: 2019 IEEE/CVF international conference on computer vision (ICCV), pp 9267–9276
31. Rong Y, Huang W, Xu T et al (2020) Dropedge: towards deep graph convolutional networks on node classification. In: ICLR 2020: eighth international conference on learning representations
32. Zhang X, Liu H, Li Q et al (2019) Attributed graph clustering via adaptive graph convolution. In: Twenty-eighth international joint conference on artificial intelligence IJCAI-19

33. Li X, Hu Y, Sun Y et al (2020) A deep graph structured clustering network. IEEE Access. https://doi.org/10.1109/ACCESS.2020.3020192
34. Wang C, Pan S, Long G et al (2017) Mgae: marginalized graph autoencoder for graph clustering. In: Proceedings of the 2017 ACM on conference on information and knowledge management, pp 889–898
35. Wang C, Pan S, Hu R et al (2019) Attributed graph clustering: a deep attentional embedding approach. In: Proceedings of the twenty-eighth international joint conference on artificial intelligence, pp 3670–3676
36. Tong G, Zhu M (2020) Text classification based on graph convolutional network with attention. In: Journal of physics conference, vol 1693, no 1, p 012038
37. Yao L, Mao C, Luo Y (2019) Graph convolutional networks for text classification. In: 33rd AAAI conference on artificial intelligence, AAAI 2019, 31st annual conference on innovative applications of artificial intelligence, IAAI 2019 and the 9th AAAI symposium on educational advances in artificial intelligence, EAAI 2019, vol 33, pp 7370–7377
38. Lu Z, Du P, Nie JY (2020) Vgcn-bert: augmenting bert with graph embedding for text classification. In: European conference on information retrieval, pp 369–382
39. Zhang Z, Shi Y, Yuan C et al (2020) Object relational graph with teacher-recommended learning for video captioning. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 13278–13288
40. Mohamed A, Qian K, Elhoseiny M et al (2020) Social-stgcnn: a social spatio-temporal graph convolutional neural network for human trajectory prediction. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 14424–14432
41. Chen ZM, Wei XS, Wang P et al (2019) Multi-label image recognition with graph convolutional networks. In: 2019 IEEE/CVF Conference on computer vision and pattern recognition (CVPR), pp 5177–5186
42. Berg Rvd, Kipf TN, Welling M (2017) Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263
43. Zhang J, Shi X, Zhao S et al (2019) Star-gcn: stacked and reconstructed graph convolutional networks for recommender systems. arXiv preprint arXiv:1905.13129
44. Ying R, He R, Chen K et al (2018) Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 974–983