Dual Channel Hypergraph Collaborative Filtering

Shuyi Ji
Tsinghua University
jisy19@mails.tsinghua.edu.cn

Yifan Feng
Xiamen University
evanfeng97@gmail.com

Rongrong Ji
Xiamen University
rrji@xmu.edu.cn

Xibin Zhao
Tsinghua University
zxb@tsinghua.edu.cn

Wanwan Tang*
Baidu, Inc.
tangwanwan01@baidu.com

Yue Gao*
Tsinghua University
gaoyue@tsinghua.edu.cn

ABSTRACT

Collaborative filtering (CF) is one of the most popular and important recommendation methodologies in the heart of numerous recommender systems today. Although widely adopted, existing CF-based methods, ranging from matrix factorization to the emerging graph-based methods, suffer inferior performance especially when the data for training are very limited. In this paper, we first pinpoint the root causes of such deficiency and observe two main disadvantages that stem from the inherent designs of existing CF-based methods, *i.e.,* 1) *inflexible modeling of users and items* and 2) *insufficient modeling of high-order correlations among the subjects*. Under such circumstances, we propose a dual channel hypergraph collaborative filtering (DHCF) framework to tackle the above issues. *First*, a dual channel learning strategy, which holistically leverages the divide-and-conquer strategy, is introduced to learn the representation of users and items so that these two types of data can be elegantly interconnected while still maintaining their specific properties. *Second*, the hypergraph structure is employed for modeling users and items with explicit hybrid high-order correlations. The jump hypergraph convolution (JHConv) method is proposed to support the explicit and efficient embedding propagation of high-order correlations. Comprehensive experiments on two public benchmarks and two new real-world datasets demonstrate that DHCF can achieve significant and consistent improvements against other state-of-the-art methods.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; *Retrieval models and ranking*.

KEYWORDS

Collaborative Filtering; Dual Channel; Hypergraph

ACM Reference Format:
Shuyi Ji, Yifan Feng, Rongrong Ji, Xibin Zhao, Wanwan Tang, and Yue Gao. 2020. Dual Channel Hypergraph Collaborative Filtering. In *26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394486.3403253

*Corresponding author

KDD '20, August 23–27, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403253

1 INTRODUCTION

The ever-growing demand for valuable, attractive and personalized information has driven the development and deployment of various recommender systems in diverse fields [6, 16]. The core of the recommender systems is a series of recommendation algorithms that efficiently sift the exploding information for users based on their personal characteristics. Collaborative filtering (CF) [22] is one of the most popular and widely-adopted methods in both industry and research communities.

In a nutshell, CF holds a basic assumption that when providing recommendations to *users*: those who behave similarly (*e.g.,* visiting the same website frequently) are likely to share similar preferences on *items* (*e.g.,* music, videos, websites). To fulfill this, a typical CF-based method performs in a two-step strategy: it first distinguishes similar users and items by leveraging the historical interactions; then based on the information gathered above, it generates recommendations to specific users.

Specially, existing CF methods can be classified into three categories. The first kind of CF methods [29] are *user-based methods* that yield recommendations merely based on similarities between users, *i.e.,* user-user correlations that describe the relationship between different users' interactions with the same item. Similarly, the *item-based methods*, as the second kind of methods [20], only employ the item-item correlations for recommendation. Both *user-based* and *item-based* methods only adopt part of the historical information when predicting the items attractive to the users, thus inevitably bearing inferior performance. The third kind of CF methods, including matrix factorization (MF) [14] and graph-based methods [2, 25], have made efforts to indiscriminately integrate the users and the items together for recommendation. The matrix factorization method models both users and items in a shared space. The graph-based methods [2, 25] formulate both users and items in a graph, which can jointly investigate the correlations among these users and items and further lead to performance improvement.

Although CF methods have been investigated for years, limitations still exist, especially when the prior knowledge for training is very limited. To understand such deficiencies, we dig deep into the inherent mechanisms of existing CF methods and obtain the following observations of limitations:

• *Inflexible modeling of users and items.* Although the graph-based CF methods model the users and items into indistinguishable nodes, there do not exist necessary distinctions between the users and

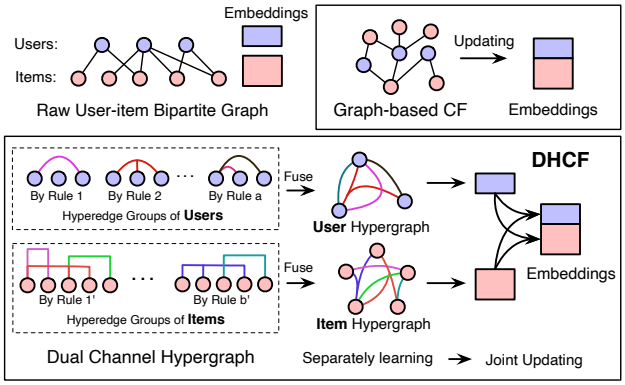


Figure 1: Comparison between graph-based CF and dual channel hypergraph CF.

the items. When an item is connected with plenty of users, such item can be rather popular. Contrarily, when a user is linked with diverse items, it does not indicate the user is popular. Under such circumstances, more flexible modeling of users and items is needed.

• *Insufficient modeling of high-order correlations.* High-order correlations between users and items are essential for data modeling. Existing methods attempt to incorporate with high-order correlations, while the employed graph structure has constraints on high-order correlation modeling and processing, as only pairwise connections can be represented in a graph.

These two issues may become more severe in the case where the training data is very limited. To tackle the above issues, we propose a dual channel hypergraph collaborative filtering (DHCF) framework, which can learn better high-order representation for users and items. To handle the distinct representation issues for users and items, we employ the divide-and-conquer strategy into the modeling process so as to integrate users and items together while still maintaining their individual properties. Specifically, as depicted in Figure 1, first we construct multiple connection groups based on the given data for both users and items. Here the connection generation rules can be regarded as a new perspective to describe the raw data, which can be defined flexibly. For instance, it can associate users who have similar behaviors but without direct connections, and thus the relation constructed based on such an association rule in a connection group can represent high-order correlation, leading to a hyperedge accordingly. Based on these generated connection groups, *i.e.,* hyperedge, we can construct two hypergraphs for users and items, *i.e.,* the two channels' representation, respectively. Here, a novel jump hypergraph convolution (JHConv) is introduced by aggregating the embeddings of neighbors' and additionally introducing prior information to efficiently conduct information propagation on hypergraph. The learned representation can be further integrated to generate recommendations.

Figure 1 provides a comparison between graph-based CF and the proposed dual channel hypergraph CF. As shown in the figure, given the raw user-item connections, graph-based methods generate a graph structure to learn the representation and the recommendation results. Different from these methods, the proposed DHCF framework can learn the representation of users and items using the high-order information in two hypergraphs respectively. The two hypergraphs, *i.e.,* the user hypergraph and the item hypergraph, can be more flexible on complex data correlation modeling and incorporation with different types of data. We have conducted extensive experiments on two public benchmarks and two new real-world datasets to evaluate the performance of the proposed DHCF framework.

The main contributions are summarized as follows:

- To learn distinct representations for users and items, the divide-and-conquer methodology is introduced to CF, which can integrate users and items together for recommendation while still maintaining their specific properties, leading to a dual channel collaborative filtering framework.
- We propose to employ the hypergraph for explicitly modeling the high-order correlations among users and items. We also propose a novel jump hypergraph convolution (JHConv) method to efficiently propagate the embedding by additionally introducing prior information.

2 PRELIMINARY OF HYPERGRAPH

We first briefly introduce the preliminaries of hypergraph.

2.1 Hypergraph Definition

For a graph structure, an edge only connects two vertices, while in a hypergraph, a hyperedge connects two or more vertices [1]. A hypergraph is usually defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the vertex set, and $\mathcal{E}$ denotes the hyperedge set. An incidence matrix $\mathbf{H} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{E}|}$ is used to represent connections among vertices on the hypergraph, with each entry $h(v,e)$ indicating whether a vertex v is connected by a hyperedge e:

$$h(v,e) = \begin{cases} 1 & if\, v \in e \\ 0 & if\, v \notin e \end{cases} \tag{1}$$

For a vertex $v \in \mathcal{V}$ and a hyperedge $e \in \mathcal{E}$, their degree can be defined, respectively, as $d(v) = \sum_{e\in\mathcal{E}} h(v,e)$, and $\delta(e) = \sum_{v\in\mathcal{V}} h(v,e)$. Further, two diagonal matrices $\mathbf{D}_v \in \mathbb{N}^{|\mathcal{V}|\times|\mathcal{V}|}$ and $\mathbf{D}_e \in \mathbb{N}^{|\mathcal{E}|\times|\mathcal{E}|}$ represent the vertex degrees and hyperedge degrees respectively.

2.2 Hypergraph Convolution

Given a hypergraph constructed from datasets, the incidence matrix $\mathbf{H}$ and vertex feature $\mathbf{X}^{(l)}$ are fed into hypergraph neural networks (HGNN) [3] with a hypergraph convolutional layer (HGNNConv) defined as:

$$\mathbf{X}^{(l+1)} = \sigma\left(\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{X}^{(l)}\Theta^{(l)}\right), \tag{2}$$

in which the $\Theta^{(l)} \in \mathbb{R}^{C^{(l)}\times C^{(l+1)}}$ is a trainable parameter and $C^{(l)}/C^{(l+1)}$ is the input/output vertex feature dimension at layer l. $\sigma(\cdot)$ denotes an arbitrary nonlinear activation function (*e.g., ReLU*$(\cdot)$). $\mathbf{X}^{(l+1)}$ is the output of layer l that can be further used for deeper vertex feature extraction or final vertex classification.

It can also be viewed as a two-stage refinement performing vertex-hyperedge-vertex feature transform upon hypergraph structure. The hypergraph incidence matrix $\mathbf{H}$ defines the message passing paths from hyperedges (column) to vertices (row). Similarly, $\mathbf{H}^T$ defines the paths from vertices (column) to hyperedges (row).

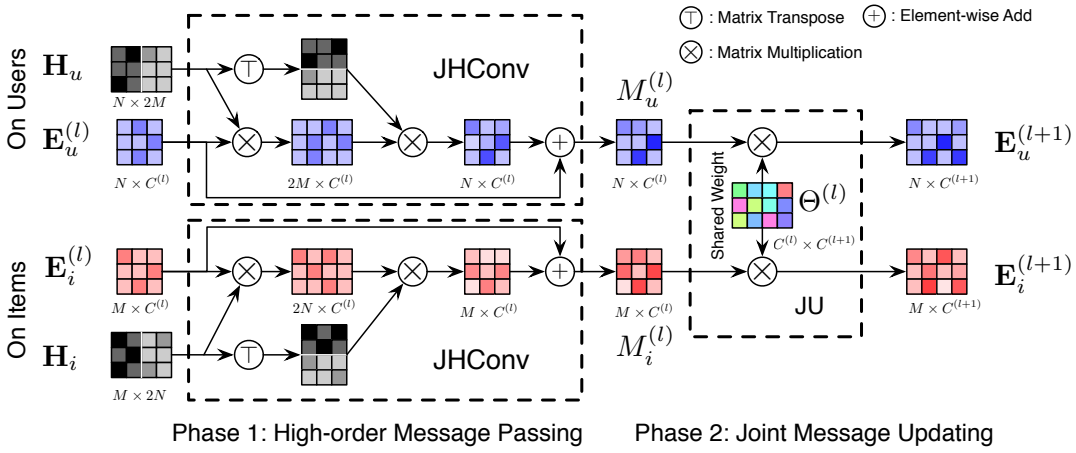


Figure 2: The framework of DHCF.

Then, we describe hypergraph convolution through the information propagation process on hypergraph. Two diagonal matrices $\mathbf{D}_v$ and $\mathbf{D}_e$ are used for normalization in Eq. 2 and they have no impact on the path of message passing on hypergraph. Thus we do not discuss them here. First, with message passing guided by H^T, the vertex feature is gathered according to the hyperedge to form the hyperedge features. Then, by aggregating their related hyperedge features through H, refined vertex features are generated. Finally, the trainable Θ and nonlinear activation function $\sigma(\cdot)$ is applied.

To sum up, compared with graph, hypergraph naturally possesses the ability to model higher-order connections. Besides, the hypergraph convolution can handle high-order correlations structures. As an effective and deep operation, hypergraph convolutions enable the high-level information interaction among vertices by leveraging the vertex-hyperedge-vertex transform.

3 METHODOLOGY

In this section, the proposed DHCF framework is introduced. We first briefly introduce the general framework of DHCF. Then we discuss a series of individual components of DHCF in detail. Afterwards, we show the specific configurations and optimizations that we adopt in this method.

3.1 A General DHCF Framework

Figure 2 sketches the overall framework of DHCF. At a high level, DHCF first learns two groups of embeddings for users and items via a dual channel hypergraph framework, upon which DHCF further figures out the user-item preference matrix by calculating the inner product of users' and items' embedding look-up table. Based on such preference matrix, DHCF estimates how likely a user is interested in an item.

In order to obtain accurate modeling and representation for both users and items, DHCF employs a dual channel hypergraph embedding framework. Specially, given the initial embedding vector $e_u \in \mathbb{R}^d / (e_i \in \mathbb{R}^d)$ with regard to a single user/item (d denotes the dimension of the embedding vector), we refine the embeddings of the user and item separately with hypergraph convolution by propagating them on a hypergraph in two phases: *high-order message passing* and *joint message updating*.

3.1.1 Initialization. Given user-item interactions with N users and M items, we first construct initial representations and hypergraph structures for users and items separately as inputs of DHCF framework. Specifically, we create two embedding look-up tables for both users and items as $\mathbf{E}_u = [\mathbf{e}_{u_1}, ..., \mathbf{e}_{u_N}]$ and $\mathbf{E}_i = [\mathbf{e}_{i_1}, ..., \mathbf{e}_{i_M}]$.

The k hyperedge groups $\{\mathcal{E}_{r_1}, ..., \mathcal{E}_{r_k}\}$ can be generated respectively based on a self-defined association rule list $\{r_1, ..., r_k\}$ for users and items. Beyond the interactions directly presented by the observed instances, such association rules can be regarded as a new perspective to describe the raw data. For example, it can associate users who have similar behaviors but without direct connections, and thus the constructed relation in a hyperedge group based on such association rule is able to capture high-order information rather than pair-wise relationships.

Then we can construct a hypergraph incidence matrix $\mathbf{H}$ with hybrid high-order correlations via fusing different hyperedge groups:

$$\mathbf{H} = f(\mathcal{E}_{r_1}, ..., \mathcal{E}_{r_k}), \tag{3}$$

in which $f(\cdot)$ indicates hyperedge fusion operation (for more details refer to Section 3.4.2).

Now, the user's/item's initial embeddings and hypergraphs are prepared for later propagation.

3.1.2 Phase 1: High-order Message Passing. To aggregate the neighboring message upon pre-defined hybrid high-order relations, we perform high-order message passing as follows:

$$\begin{cases} M_u = \text{HConv}(E_u, H_u) \\ M_i = \text{HConv}(E_i, H_i) \end{cases}, \tag{4}$$

where $\text{HConv}(\cdot, \cdot)$ indicates arbitrary hypergraph convolution operations like HGNNConv. Specifically, the $\text{HConv}(\cdot, \cdot)$ here is purely an information propagation on hypergraph without any trainable parameters ($\Theta^{(l)}$ in Equation 2). The output M_u and M_i have learned the complex correlations from its high-order neighbors, respectively. Note that the neighbor here is an abstract description beyond the direct connections in historical interactions, which can reveal the similarity in the latent behavior space. Then we will consider jointly update E_u and E_i using M_u and M_i.

3.1.3 Phase 2: Joint Message Updating. To distill discriminative information, we conduct joint message updating for users and items defined as:

$$\begin{cases} E'_u = \mathrm{JU}(M_u, M_i) \\ E'_i = \mathrm{JU}(M_i, M_u) \end{cases}, \tag{5}$$

where $\mathrm{JU}(\cdot,\cdot)$ can be arbitrary learnable feed-forward neural networks designed to update the first parameter with the second one.

To sum up, the two above-mentioned processes constitute an integrated DHCF layer, which allows explicit modeling and encoding high-order correlations on both users and items, and further update and generate more accurate embeddings through powerful hypergraph structure. Such refined embeddings can be further applied in various downstream tasks in recommender systems.

3.2 Jump Hypergraph Convolution

Inspired by some instructive methods like Weisfeiler-Lehman algorithm [21] and GCN [12], especially GraphSAGE [7], which concatenates a node's current representation with its aggregated neighborhood vectors to generate the learned representations, we introduce a novel JHConv operator:

$$\mathbf{X}^{(l+1)} = \sigma\left(\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{X}^{(l)}\Theta^{(l)} + \mathbf{X}^{(l)}\right), \tag{6}$$

in which notations have been introduced in Section 2.2.

Comparison with HGNN [3]. Compared with traditional HGNNConv in [3], JHConv allows the model to simultaneously consider both its original features and aggregated related representations, as we can see from Equation 2 and 6. On the other side, such resnet-like skip connection enables the model to avoid information dilution due to the integration of many additional connections.

3.3 High-order Connectivity Definition

In this subsection, we introduce a way of constructing high-order connectivity on user-item bipartite graph for users and items, respectively. A user-item bipartite graph can be indicated by an incidence matrix $\mathbf{H} \in \{0,1\}^{N\times M}$.

3.3.1 On Users. **Definition 1: Item's *k*-order reachable neighbors.** In a user-item interaction graph, more specifically the bipartite graph, $item_i$ ($item_j$) is $item_j$ ($item_i$)'s k-order reachable neighbor if there exists a sequence of adjacent vertices (*i.e.,* a path) between $item_i$ and $item_j$, and the number of user in this path is smaller than k.

Definition 2: Item's *k*-order reachable users. In an item-user bipartite graph, $user_j$ is k-order reachable from $item_i$ if there exists direct interaction between $user_j$ and $item_k$, and $item_k$ is $item_i$'s k-order reachable neighbor.

For $item_i$, its k-order reachable users set is termed as $B_u^k(i)$. Mathematically speaking, a hypergraph can be defined upon a set family, in which each set represents a hyperedge. Thus, here a hyperedge can be built by an item's k-order reachable users set. Then we can construct high-order hyperedge group upon the ***k*-order reachable rule among users**, which can be formulated as:

$$\mathcal{E}_{B_u^k} = \{B_u^k(i) \mid i \in I\} \tag{7}$$

The k-order reachable matrix of items can be denoted as $A_i^k \in \{0,1\}^{M\times M}$, which takes the form as:

$$\mathbf{A}_i^k = \min(1, \mathrm{power}(\mathbf{H}^\top \cdot \mathbf{H}, k)), \tag{8}$$

where $\mathrm{pow}(M,k)$ is the function that calculates the k power of a given matrix M. $\mathbf{H} \in \{0,1\}^{N\times M}$ denotes the incidence matrix of the user-item bipartite graph. Then the hyperedge group incidence matrix $\mathbf{H}_{B_u^k} \in \{0,1\}^{N\times M}$ that constructed by ***k*-order reachable rule among users** can be formulated as:

$$\mathbf{H}_{B_u^k} = \mathbf{H} \cdot \mathbf{A}_i^{k-1}. \tag{9}$$

Supposing we have a hyperedge groups built upon users via the ***k*-order reachable rule**, then the final hybrid high-order connections among users can be represented by the hypergraph $\mathcal{G}_u$ fusing a hyperedge groups. Due to the advantage of hypergraph in multi-modality fusion [3], a simple concatenation operation $\cdot||\cdot$ for hyperedge groups' incidence matrix can be applied for hyperedge groups fusion $f(\cdot)$. Finally, the hypergraph incidence matrix $\mathbf{H}_u$ for users can be formulated as:

$$\mathbf{H}_u = f\left(\mathcal{E}_{B_u^{k_1}}, \mathcal{E}_{B_u^{k_2}}, \ldots, \mathcal{E}_{B_u^{k_a}}\right) = \underbrace{\mathbf{H}_{B_u^{k_1}} || \mathbf{H}_{B_u^{k_2}} || \ldots || \mathbf{H}_{B_u^{k_a}}}_{a} \tag{10}$$

3.3.2 On Items. **User's *k*-order reachable neighbors** and **User's *k*-order reachable items** can be symmetrically defined in the similar way. Specifically, the k-order reachable matrix of user can be denoted as $A_u^k \in \{0,1\}^{N\times N}$, which can be written as:

$$\mathbf{A}_u^k = \min(1, \mathrm{power}(\mathbf{H} \cdot \mathbf{H}^\top, k)), \tag{11}$$

The hyperedge group incidence matrix $\mathbf{H}_{B_i^k} \in \{0,1\}^{M\times N}$ constructed by ***k*-order reachable rule among items** can be formulated as:

$$\mathbf{H}_{B_i^k} = \mathbf{H}^\top \cdot \mathbf{A}_u^{k-1}. \tag{12}$$

Then, the hypergraph incidence matrix $\mathbf{H}_i$ for items can be formulated as:

$$\mathbf{H}_i = f\left(\mathcal{E}_{B_i^{k_1}}, \mathcal{E}_{B_i^{k_2}}, \ldots, \mathcal{E}_{B_i^{k_b}}\right) = \underbrace{\mathbf{H}_{B_i^{k_1}} || \mathbf{H}_{B_i^{k_2}} || \ldots || \mathbf{H}_{B_i^{k_b}}}_{b} \tag{13}$$

In this way, we can define the high-order connectivity for both users and items. Figure 3 illustrates one example of user's high-order connectivity when $k = 1$ and 2, respectively.

3.4 DHCF Layer Configurations

Here we introduce a specific configuration of the DHCF framework.

3.4.1 Construction of Hybrid High-order Connections. For each user/item, we set k as 1 and 2 respectively to capture the k-order neighbors of it, namely B^1 and B^2, and then construct two kinds of high-order correlations for each user/item. Then we incorporate these two high-order correlations respectively to construct the hypergraph of user/item and the incidence matrix of it can thereby be denoted as $\mathbf{H}_u$ and $\mathbf{H}_i$ respectively:

$$\begin{cases} \mathbf{H}_u = f(\mathcal{E}_{B_u^1}, \mathcal{E}_{B_u^2}) & = \mathbf{H}_{B_u^1} || \mathbf{H}_{B_u^2} \\ \mathbf{H}_i = f(\mathcal{E}_{B_i^1}, \mathcal{E}_{B_i^2}) & = \mathbf{H}_{B_i^1} || \mathbf{H}_{B_i^2} \end{cases} \tag{14}$$

It is worth noting that the proposed DHCF framework is generally applicable and extensible. In other words, although we adopt B^1 and B^2 to explore high-order connectivity information here, our model has no specific requirements for that, which means other flexible high-order correlation definitions are also plausible to be incorporated into the proposed framework.

3.4.2 Configuration of an Integrated DHCF *Layer.* Given user embeddings E_u, item embeddings E_i, hypergraph incidence matrix on user H_u and item H_i, we now provide a detailed definition of high-order message passing and joint message updating.

Phase 1: For high-order message passing, we adopt JHConv introduced in Section 3.2 to propagate user/item embeddings on the hypergraph while retaining original information. Here the user representation and the item representation can be learned through this dual-channel way.

Phase 2: For joint message updating, we apply a shared fully connected layer on the output of the previous phase. The detailed configurations are formulated as:

$$\begin{cases} f(\dots) = & \cdot||\cdot \\ \text{HConv}(\cdot,\cdot) = & \text{JHConv}(\cdot,\cdot) \\ \text{JU}(\cdot,\cdot) = & \text{MLP}_1(\cdot) \end{cases}, \tag{15}$$

where $\cdot||\cdot$ is the concatenation operation. $\text{MLP}_1(\cdot)$ denotes a fully connected layer with trainable Θ, which only adopts the first parameter of $\text{JU}(\cdot,\cdot)$.

Matrix Form of Propagation Rule To provide a general understanding of embedding propagation rule on hypergraph, here we give the matrix form of it:

$$\begin{cases} \left.\begin{aligned} \mathbf{H}_u &= \mathbf{H}||\left(\mathbf{H}(\mathbf{H}^\top\mathbf{H})\right) \\ \mathbf{H}_i &= \mathbf{H}^\top||\left(\mathbf{H}^\top(\mathbf{H}\mathbf{H}^\top)\right) \end{aligned}\right\} \text{hypergraph setup} \\ \left.\begin{aligned} \mathbf{M}_u^{(l)} &= \mathbf{D}_{u_v}^{-1/2}\mathbf{H}_u\mathbf{D}_{u_e}^{-1}\mathbf{H}_u^\top\mathbf{D}_{u_v}^{-1/2}\mathbf{E}_u^{(l)} + \mathbf{E}_u^{(l)}, \\ \mathbf{M}_i^{(l)} &= \mathbf{D}_{i_v}^{-1/2}\mathbf{H}_i\mathbf{D}_{i_e}^{-1}\mathbf{H}_i^\top\mathbf{D}_{i_v}^{-1/2}\mathbf{E}_i^{(l)} + \mathbf{E}_i^{(l)}, \end{aligned}\right\} \text{Phase 1} \\ \left.\begin{aligned} \mathbf{E}_u^{(l+1)} &= \sigma(\mathbf{M}_u^{(l)}\Theta^{(l)}) \\ \mathbf{E}_i^{(l+1)} &= \sigma(\mathbf{M}_i^{(l)}\Theta^{(l)}) \end{aligned}\right\} \text{Phase 2,} \end{cases} \tag{16}$$

where $H \in \{0,1\}^{N\times M}$ is the initial incidence matrix of a user-item bipartite graph. $\mathbf{D}_{u_v}$, $\mathbf{D}_{u_e}$ and $\mathbf{D}_{i_v}$, $\mathbf{D}_{i_e}$ are diagonal matrices representing the vertex degrees and hyperedge degrees of user and item hypergraph respectively. $\mathbf{E}_u^{(l)}$, $\mathbf{E}_i^{(l)}$ and $\mathbf{E}_u^{(l+1)}$, $\mathbf{E}_i^{(l+1)}$ are the input/ouput user embeddings and item embeddings in layer l, respectively. Given raw incidence matrix of a user-item bipartite graph $\mathbf{H}$, hypergraph $\mathbf{H}_u$ and $\mathbf{H}_i$ with hybrid high-order correlations are firstly constructed on users and items, respectively. Then $\mathbf{E}_u^{(l)}$, $\mathbf{H}_u$, $\mathbf{E}_i^{(l)}$ and $\mathbf{H}_u$ are fed into **phase 1** and **phase 2** to generate updated user embeddings $\mathbf{E}_u^{(l+1)}$ and item embeddings $\mathbf{E}_i^{(l+1)}$ for next propagation or final link prediction. More detailed descriptions of generating user/item embedding with DHCF layer can be found at Appendix A.

3.5 Optimization

In this paper we focus on limited implicit feedback, which is more pervasive in practice but also more deficient and inherently noisy, compared with explicit feedback like ratings and reviews. Specifically, in implicit feedback, only positive feedback is visible and the rest of the data are all treated as missing data, as there is no way to tell whether a user dislikes or just neglects the item. Therefore, we opt for logistic optimization, *i.e.,* pairwise optimization instead of point wise approaches. It holds the pairwise assumption that a given user u prefers the observed item i^+ to the unobserved item i^- and therefore the observed ones should be ranked higher. Despite still inherently noisy, the sheer volume of unobserved items compensates for this drawback and conduces to build upon robust recommender systems. We further leverage the pairwise Bayesian Personalized Ranking (BPR) [18] optimization criterion as the loss function:

$$\mathcal{L} = -\sum_{(u,i^+,i^-)\in\mathcal{T}} -\ln\sigma(\hat{r}_{ui^+} - \hat{r}_{ui^-}) + \lambda||\Theta||_2^2, \tag{17}$$

where $\mathcal{T}$ indicates the pairwise training data, in which it is assumed that user u prefers item i^+ to item i^-; $\sigma(\cdot)$ is the logistic sigmoid function; Θ denotes all model parameters and λ is model specific regularization parameter to avoid overfitting.

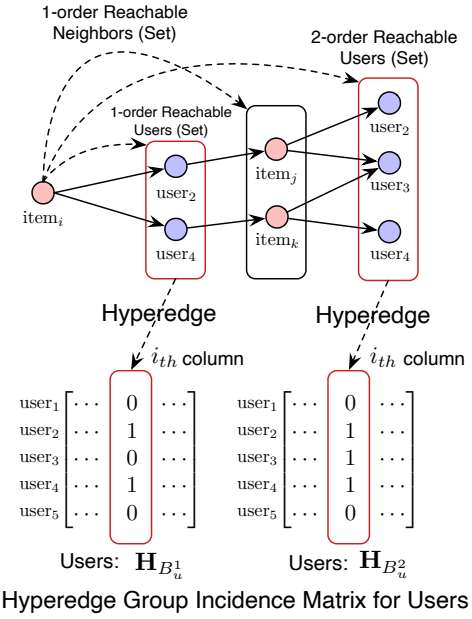


Figure 3: Illustration of defined high-order connectivity for users.

4 EXPERIMENTS

To evaluate the effectiveness of the proposed method, we have conducted experiments on two public benchmarks and two newly-collected real-world datasets. Ablation studies are also presented for better investigation of our proposed model.

4.1 Datasets and Evaluation Metrics

Four datasets are employed in our experiments, including MovieLens-100K, CiteUlike-A, Baidu-Feed, and Baidu-news. The former two

Table 1: Characteristics of datasets.

Datasets	#Users	#Items	#Interactions	Density
MovieLens-100K	943	1682	100000	0.06305
CiteUlike-A	3277	16807	178062	0.00323
Baidu-feed	2000	20062	107778	0.00027
Baidu-news	2000	15595	74268	0.00238

datasets are publicly used benchmarks, and the latter two are newly-collected ones from Baidu app. For MovieLens-100K and CiteUlike-A datasets, we transform the original explicit feedback into implicit data, where each entry is marked as 0 or 1 indicating whether the user has rated or cited. The characteristics of the four datasets are summarized in Table 1.

MovieLens-100K. The MovieLens dataset [8] has been widely used in different recommendation tasks for evaluation. We select the version of implicit feedback of MovieLens-100K, including 1,682 ratings of 943 users on the Movielens website.

CiteUlike-A. The CiteUlike dataset [24] is another commonly-used benchmark in recommender systems. It has two versions collected from CiteULike and Google Scholar independently, *i.e.,* CiteUlike-A and CiteUlike-T. In our experiment, the CiteUlike-A data is used.

Baidu-feed. Baidu app is a one-stop mobile application with more than 700 million active users, comprising a series of vertical services like web searching, social networking, and entertainment services. Here we first present a formal definition to the feed-stream recommendation. The feed-stream recommendation is a kind of service that continuously integrates the contents (*e.g.,* news and stream media) that the users may be interested in together, and further pushes to the client side. We collected one-day feed-stream click traces of 2000 anonymous users on Baidu app in December 2019, *i.e., Baidu-feed dataset*. Different from other public benchmark datasets, user-item interactions in this dataset are not gathered from one specific vertical service, yet instead, from a one-stop mobile application (Baidu app). Therefore, the items in *Baidu-feed dataset* possess high heterogeneity in terms of data type (including videos, news, ads, and *etc.*), which is consistent with the real data that users can acquire in the wild.

Baidu-news dataset is also collected from the mobile Baidu app but different from *Baidu-feed dataset*, where the Baidu-news item uniquely refers to the news. Note that Baidu-news dataset is not a subset of the Baidu-feed dataset.

In experiments, 10% of all observed interactions of each user are randomly selected for training and the remaining data are used for testing. Such a setting increases the difficulty of the CF task, as the model can only fetch very limited observed interactions. In addition, due to the high sparsity of data, it can well evaluate models' ability to dig out useful information from the limited implicit datasets. For all four datasets, we use the 20-core setting to ensure data quality, *i.e.,* each user has at least two interactions for training. During the training process, each positive instance (observed interactions) is paired with a negative item sampled from unobserved interactions.

For evaluation, four widely-used metrics, including *precision@K*, *recall@K*, normalized discounted cumulative gain (*ndcg@K*), and *hit@K*, are calculated to comprehensively compare the performance of different methods when acting on the top-K recommendation and the ranking task. In our experiments, K is set as 20.

4.2 Compared Methods

Four recent competitive methods are selected for comparison.

• **BPR-MF** [18] is the classic and popular matrix factorization using the BPR as loss function, which optimizes the pairwise ranking between the positive instances and sampled negative items.

• **GC-MC** [2] employs a graph auto-encoder framework for the matrix completion task in recommendation systems, in which a graph convolution layer is introduced into the encoder to generate user and item embeddings through message passing.

• **PinSage** [28] learns node embeddings through random walk GCN on pins-boards graph. Here we apply it on the user-item interaction graph for comparison.

• **NGCF** [25] is a state-of-the-art graph-based model, which encodes the collaborative signal into the user-item interaction graph structure. It adopts multiple graph convolution layers and performs embedding propagation to explore high-order connectivity.

For a fair comparison, we adopt the multi-grained strategy to generate the final embeddings as suggested in [25] for all methods, which can be formulated as $\mathbf{E}^r = \mathbf{E}||\mathbf{E}^{(1)}||\mathbf{E}^{(2)}||\dots||\mathbf{E}^{(L)}$, where $\cdot||\cdot$ denotes concatenation operation. $\mathbf{E}$ and $\mathbf{E}^{(l)}$ denote the initial embeddings and the output embeddings of layer l, respectively. Through concatenating the initial user/item embeddings and the output embeddings of each layer, the final user/item embeddings $\mathbf{E}^r$ is generated. Besides, we adopt the BPR optimization criterion shown in Equation (17) as loss function for all methods.

4.3 Parameter Settings

The proposed DHCF framework is implemented in PyTorch. The hidden dimension, *i.e.,* the embedding size for users and items, are fixed as 64 for all methods. The batch size is set as 128. For all compared methods, we opt for the Adam optimizer and other optimizers are also plausible. We perform a grid search for hyper-parameters: the dropout ration in {0.0, 0.1, 0.2}, the learning rate is searched in {0.0001, 0.0005, 0.001} and the regularization terms $\{10^{-5}, 10^{-4}, 10^{-3}\}$. In terms of initialization, for user and item embeddings, we adopt Xavier initializer [4] to initialize, while for model learnable parameters, we use Xavier initializer to initialize weight and the bias is initialized as 0. For DHCF, without specification, the network depth L is set as 1 and JHConv is adopted. The hyperedge groups are set as the combination of B^1 and B^2 for both users and items. For more details of parameter settings, please refer to Appendix B.

4.4 Experimental Results and Discussions

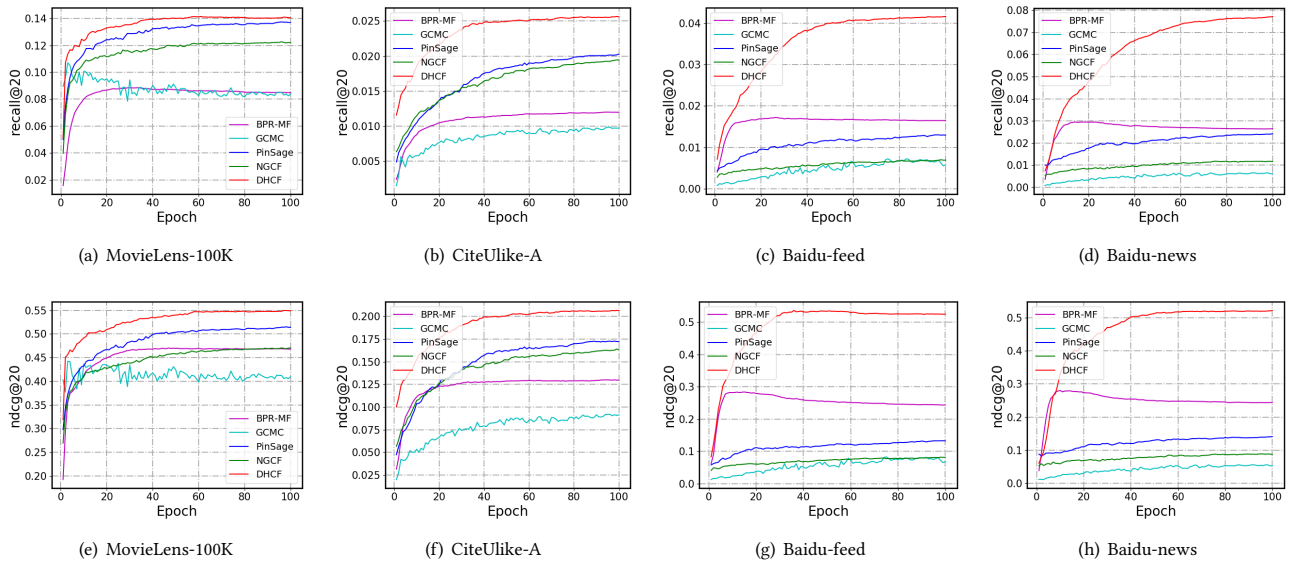
We first discuss the overall experimental results of all methods on four datasets and then the comparisons of convergence performance are presented.

4.4.1 Overall Comparison. The experimental results and comparisons are shown in Table 2. From these results we can have the following observations:

• As a simple but effective method, BPR-MF shows robust performance when transforming to nosier and more complex real-world datasets from cleaned public benchmarks. It performs better on

Table 2: Experimental results for different methods on MovieLens-100K [8], CiteUlike-A [24], Baidu-feed, and Baidu-news datasets. The best results are marked in red, and the second-best results are marked in blue. We set K=20 by default.

	MovieLens-100K				CiteUlike-A				Baidu-feed				Baidu-news			
	prec.	recall	ndcg	hit	prec.	recall	ndcg	hit	prec.	recall	ndcg	hit	prec.	recall	ndcg	hit
BPR-MF [18]	0.3410	0.0869	0.4670	0.9512	0.0330	0.0124	0.1302	0.2970	**0.0362**	**0.0158**	**0.2451**	**0.5140**	**0.0416**	**0.0266**	**0.2562**	**0.5550**
GC-MC [2]	0.3171	0.0891	0.4171	0.9417	0.0317	0.0103	0.0953	0.2402	0.0162	0.0065	0.0763	0.2381	0.0098	0.0060	0.0498	0.1595
PinSage [28]	**0.4295**	**0.1342**	**0.5082**	**0.9830**	0.0508	**0.0194**	**0.1712**	**0.4034**	0.0300	0.0136	0.1380	0.4025	0.0376	0.0253	0.1451	0.4480
NGCF [25]	0.3889	0.1233	0.4716	0.9799	**0.0517**	0.0193	0.1606	0.3824	0.0162	0.0072	0.0826	0.2515	0.0170	0.0112	0.0836	0.2685
DHCF	**0.4375**	**0.1424**	**0.5572**	**0.9862**	**0.0635**	**0.0249**	**0.2015**	**0.4858**	**0.0937**	**0.0430**	**0.5067**	**0.8540**	**0.1136**	**0.0763**	**0.5198**	**0.8970**
% Improv.	1.8%	6.1%	9.6%	0.3%	22.8%	28.4%	17.7%	20.4%	158.8%	172.2%	106.7%	66.1%	173.1%	186.8%	102.8%	61.6%
p-value	6.40e-5	6.72e-4	1.97e-6	3.01e-3	3.41e-7	1.03e-7	5.10e-9	3.73e-9	3.70e-12	1.87e-12	5.37e-12	4.09e-11	4.05e-11	1.62e-11	1.43e-9	2.25e-10



(a) MovieLens-100K (b) CiteUlike-A (c) Baidu-feed (d) Baidu-news

(e) MovieLens-100K (f) CiteUlike-A (g) Baidu-feed (h) Baidu-news

Figure 4: Test performance *w.r.t.* epoch of all baselines and DHCF.

Baidu-feed and Baidu-news datasets compared with GC-MC, PINSage and NGCF. BPR-MF has no dependency on information flowed in graph structure, which may be partly disturbed by noisy data.

• GC-MC bears an unsatisfactory performance across all datasets, probably due to the fact that GC-MC only utilizes one layer of GCN, which means it can only adopt 1-hop neighbors for learning. When the scale of the training data is small, GC-MC fails to fetch enough prior information for the learning process. In addition, another factor that attributes to GC-MC's poor performance lies in that it drops the current node's information, which incurs poor robustness to the noises in the aggregated neighbors' information.

• PinSage shows better performance in two public benchmarks compared with BPR-MF, GC-MC and NGCF. PinSage introduces high-order connectivity as well as retains the information of both the current node and aggregated neighbors during message passing and updating process, thus becoming the strongest baseline in MovieLens-100K and CiteUlike-A datasets.

• NGCF employs more complex node update strategy and has double learnable parameters compared with Pinsage. Nonetheless, because of the small scale of the dataset coupled with the low proportion of the training data, NGCF easily suffers from the overfitting problem and yields inferior performance.

• DHCF yields consistent best performance on all datasets. In particular, DHCF outperforms the strongest baseline by 6.1%, 28.4%, 172.2%, and 186.8% in terms of recall@20 on MovieLens-100K, CiteUlike-A, Baidu-feed, and Baidu-news datasets, respectively. We attribute the improvement mainly to the flexible and explicit modeling of hybrid high-order connections for both users and items within the extensible DHCF framework. Meanwhile, it cut down the volume of learnable parameters in model to a large extent (one sixth of NGCF for comparison), which can effectively prevent overfitting on small datasets. Specifically, NGCF has three embedding propagation layers where each layer has two trainable matrices. By contrast, DHCF only adopts one propagation layer with one learnable matrix, and thus is more lightweight and efficient. We can also observe that when the data density becomes lower, the gains of the proposed method are higher. The lower data density indicates that the useful connection is less and the correlation exploration becomes even more difficult. This result justifies the effectiveness of the proposed dual channel hypergraph method on high-order correlation modeling.

4.4.2 Performance Comparison on Convergence. Figure 4 shows the convergence curve of DHCF and all baselines, taking recall@20 and ndcg@20 as examples. From results we can observe that DHCF

Table 3: Experimental comparison between JHConv and HGNNConv.

	MovieLens-100K			
	prec.@20	recall@20	ndcg@20	hit@20
DHCF-JHConv	**0.4439**	**0.1417**	**0.5511**	**0.9926**
DHCF-HGNNConv	0.4286	0.1319	0.5407	0.9894
	CiteUlike-A			
	prec.@20	recall@20	ndcg@20	hit@20
DHCF-JHConv	**0.0635**	**0.0249**	**0.2015**	**0.4858**
DHCF-HGNNConv	0.0615	0.0238	0.2002	0.4769

Table 4: Experimental comparison with respect to different numbers of layers.

	MovieLens-100K			
	prec.@20	recall@20	ndcg@20	hit@20
DHCF-1	0.4439	0.1417	**0.5511**	**0.9926**
DHCF-2	**0.4455**	**0.1428**	0.5507	0.9841
DHCF-3	0.4273	0.1328	0.5266	0.9873
	CiteUlike-A			
	prec.@20	recall@20	ndcg@20	hit@20
DHCF-1	**0.0635**	**0.0249**	**0.2015**	**0.4858**
DHCF-2	0.0603	0.0230	0.1938	0.4684
DHCF-3	0.0573	0.0217	0.1732	0.4229

yields superior performance at each epoch with rapid increasing. The proposed DHCF framework requires about 60 epochs to achieve stable performance.

4.5 Ablation Studies

4.5.1 On the Effect of JHConv. To investigate the effectiveness of JHConv, we consider the variants of DHCF with different hypergraph convolution schemes, *i.e.,* JHConv and HGNNConv [3]. The experimental results are demonstrated in Table 3. From these results we can observe that DHCF-JHConv consistently outperforms DHCF-HGNNConv, while DHCF-HGNNConv also yields comparable performance compared with other baselines. We attribute the improvement to the resnet-like skip connection during convolution, which HGNNConv does not consider. Such skip connection can prevent the current node's information from being diluted when message passing process, thus achieving effective learning.

4.5.2 On the Number of Layers. To explore how the number of layers affects the performance, we vary the model depth in the range of {1, 2, 3}. Experimental results are shown in Table 4, wherein DHCF-1 indicates the proposed model with one embedding propagation layers, and notations are similar for -2 and -3. We observe that DHCF-1 outperforms DHCF-2 on CiteUlike-A dataset while on MovieLens-100K DHCF-2 shows a slight improvement over DHCF-1. Such performance fluctuations are normal and depend on the intrinsic structure of the dataset. Usually, DHCF can yield satisfactory performances with only one hidden layer. Stacking multiple layers such as DHCF-3 in MovieLens-100K may introduce noise and further lead to overfitting. However, such DHCF framework has the potential to be applied in a large-scale sparse real-world scenario without the necessity of stacking a very deep GCN-based architecture to capture high-order connectivity.

Table 5: Experimental comparison with respect to different high-order Connectivity.

User		Item		MovieLens-100K		Baidu-feed	
B^1	B^2	B^1	B^2	recall@20	ndcg@20	recall@20	ndcg@20
✓		✓		0.1419	0.5332	0.0411	0.5452
✓			✓	0.1403	0.5408	0.0419	0.5532
✓		✓	✓	0.1422	0.5487	0.0424	0.5645
	✓	✓		0.1324	0.5453	0.0410	0.5386
	✓		✓	0.1351	0.5387	0.0411	0.5080
	✓	✓	✓	0.1346	0.5285	0.0423	0.5218
✓	✓	✓		0.1377	0.5524	0.0423	0.5439
✓	✓		✓	0.1399	0.5455	0.0421	0.5093
✓	✓	✓	✓	**0.1424**	**0.5572**	**0.0430**	**0.5667**

4.5.3 On the Effect of High-order Connectivity. We establish different high-order associations on users and items, and conduct experiments to explore the effect of different combinations of high-order correlations on performance. Experimental results are shown in Table 5. From these results we can observe that the configuration of simultaneously considering B^1 and B^2 for both users and items can yield the best performance, which verifies the effectiveness of modeling hybrid high-order connections.

5 RELATED WORK

5.1 Model-based CF.

A typical model-based CF framework aims to develop a model through partially observed user-item interactions, enabling the recommender system to identify more complex patterns and further generate recommendations accordingly. There are several model-based CF algorithms, including association algorithms [15], clustering approaches [23], Bayesian networks [27], and latent factor models like singular value decomposition (SVD) [13, 17]. Among these methods, Matrix Factorization (MF) [14, 19] is one of the most popular and effective CF methods due to its stable performance and high scalability. It projects user and item to a shared latent factor space, and then estimates possible interactions through inner products of their vector representations. Despite being widely-adopted, MF still bears the shortcoming where it fails to tell the unobserved positive pairs apart from the unobserved negative ones. Further, Rendle *et al.* devised BPR [18] which uses pairwise log-sigmoid function to directly optimize the vanilla AUC based on Bayesian AUC optimization. In addition, considering the MF's linear inner product cannot comprehensively reflect the intricate inter-relationships between users and items, several works have made efforts to investigate the non-linear interaction function [10] with the support of deep neural networks. Typically, He *et al.* [10] presents Neural MF, which leverages multi-layer perception to learn a user-item

interaction function, which serves as an alternative to the linear inner product.

Although some of the aforementioned approaches achieve decent performance, they still fall short of generating the representative embeddings between users and items. Such deficiency lies in that they do not directly encode the collaborative signals, and thus cannot ensure users and items which are potentially connected own similar properties in the embedding space.

5.2 Graph-based CF.

Recently many researchers explore graph structure to model user-item interactions, with each user or item indicating a vertex in the graph and the interactions denote an edge between them. One typical prior work is SimRank [11], whose basic idea lies in that two objects referred by similar objects tend to be similar too. In addition, some other works borrow the idea of label propagation, for example, ItemRank [5] and BiRank [9]. Recently, the emerging GCN [12] has injected new vitality into CF. GCMC [2] applies the GCN to user-item interactions. PinSage [28] embeds the GCN to commercial recommender systems through adopting the idea of GraphSAGE [7], coupled with employing multiple GCN layers in pins-boards graph. HOP-Rec [26] is proposed based on the original BPR-MF. It explicitly defines the high-order connectivities on the CF model, which, however, is only adopted for enriching the dataset. In order to endow GNN with the high-order connectivities, NGCF [25] empowers the collaborative signal with high-order connectivities during the embedding propagation process.

6 CONCLUSIONS AND FUTURE WORK

In this work, we propose a DHCF framework , which can model latent collaborative signals in a high-order and divide-and-conquer way to address CF with implicit feedback. Specifically, it allows explicitly modeling hybrid high-order connections for users and items respectively in hypergraphs and thus can yield more accurate embeddings using the proposed JHConv operator, which can conduct the efficient information propagation on hypergraphs. Extensive experiments on two public benchmarks and two new real-world datasets demonstrate significant improvements over competitive baselines. As shown in the evaluations, we can conclude that the high-order information is useful for data representation and the proposed dual channel hypergraph model is effective.

Future work will focus on the learnable parameters that can balance the weight of hyperedge groups defined by different association rules to make DHCF more flexible.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Funds of China (U1701262).

REFERENCES

[1] Austin R Benson, David F Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (2016), 163–166.
[2] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2018. Graph Convolutional Matrix Completion. In *Proc. of the 24th ACM International Conference on Knowledge Discovery and Data mining (SIGKDD)*.
[3] Yifan Feng, Haoxuan You, Zizhao Zhang, et al. 2019. Hypergraph neural networks. In *Proc. of the 33rd Conference on Artificial Intelligence (AAAI)*. 3558–3565.
[4] Xavier Glorot and Yoshua Bengio. 2010. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proc. of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 249–256.
[5] Marco Gori, Augusto Pucci, V Roma, et al. 2007. Itemrank: A Random-walk Based Scoring Algorithm for Recommender Engines. In *Proc. of the 20th International Joint Conference on Artifical Intelligence (IJCAI)*. 2766–2771.
[6] Mihajlo Grbovic and Haibin Cheng. 2018. Real-time Personalization using Embeddings for Search Ranking at Airbnb. In *Proc. of the 24th ACM International Conference on Knowledge Discovery and Data mining (SIGKDD)*. 311–320.
[7] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proc. of the 31st Advances in Neural Information Processing Systems (NeurIPS)*. 1024–1034.
[8] F Maxwell Harper and Joseph A Konstan. 2015. The Movielens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2015), 1–19.
[9] Xiangnan He, Ming Gao, Min-Yen Kan, et al. 2016. Birank: Towards Ranking on Bipartite Graphs. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2016), 57–71.
[10] Xiangnan He, Lizi Liao, Hanwang Zhang, et al. 2017. Neural Collaborative Filtering. In *Proc. of International Conference on World Wide Web*. 173–182.
[11] Glen Jeh and Jennifer Widom. 2002. SimRank: a Measure of Structural-context Similarity. In *Proc. of the 8th ACM International Conference on Knowledge Discovery and Data mining (SIGKDD)*. 538–543.
[12] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. of International Conference on Learning Representations (ICLR)*.
[13] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proc. of the 14th ACM International Conference on Knowledge Discovery and Data mining (SIGKDD)*. 426–434.
[14] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.
[15] C-H Lee, Y-H Kim, and P-K Rhee. 2001. Web Personalization Expert with Combining Collaborative Filtering and Association Rule Mining Technique. *Expert Systems with Applications* 21, 3 (2001), 131–137.
[16] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM international conference on Knowledge discovery and data mining (SIGKDD)*. 1222–1230.
[17] Arkadiusz Paterek. 2007. Improving regularized singular value decomposition for collaborative filtering. In *Proc. of the ACM SIGKDD cup and workshop*, Vol. 2007. 5–8.
[18] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, et al. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proc. of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*.
[19] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *Proc. of the 34th International ACM Conference on Research and Development in Information Retrieval (SIGIR)*. 635–644.
[20] Badrul Sarwar, George Karypis, Joseph Konstan, et al. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proc. of the 10th international conference on World Wide Web*. 285–295.
[21] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, et al. 2011. Weisfeiler-lehman Graph Kernels. *Journal of Machine Learning Research* 12 (2011), 2539–2561.
[22] Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence* (2009).
[23] Lyle H Ungar and Dean P Foster. 1998. Clustering Methods for Collaborative Filtering. In *AAAI Workshop on Recommendation Systems*. 114–129.
[24] Hao Wang, Binyi Chen, and Wu-Jun Li. 2013. Collaborative Topic Regression with Social Regularization for Tag Recommendation. In *Proc. of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*.
[25] Xiang Wang, Xiangnan He, Meng Wang, et al. 2019. Neural Graph Collaborative Filtering. In *Proc. of the 42nd International ACM Conference on Research and Development in Information Retrieval (SIGIR)*. 165–174.
[26] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, et al. 2018. HOP-rec: High-order Proximity for Implicit Recommendation. In *Proc. of the 12th ACM Conference on Recommender Systems (RecSys)*. 140–144.
[27] Xiwang Yang, Yang Guo, and Yong Liu. 2012. Bayesian-inference-based Recommendation in Online Social Networks. *IEEE Transactions on Parallel and Distributed Systems* 24, 4 (2012), 642–651.
[28] Rex Ying, Ruining He, Kaifeng Chen, et al. 2018. Graph Convolutional Neural Networks for Web-scale Recommender Systems. In *Proc. of the 24th ACM International Conference on Knowledge Discovery and Data mining (SIGKDD)*. 974–983.
[29] Zhi-Dan Zhao and Ming-Sheng Shang. 2010. User-based Collaborative-filtering Recommendation Algorithms on Hadoop. In *Proc. of the third International Conference on Knowledge Discovery and Data mining (SIGKDD)*. IEEE, 478–481.

APPENDIX

A ALGORITHM OF DHCF FRAMEWORK

Algorithm 1 details how stacked DHCF layers generate user/item embeddings. We first construct hybrid high-order structures with specified rules (*e.g.,* k-order reachable rule) for user and item, respectively. Then their initial embeddings $\mathbf{E}_u^{(0)}, \mathbf{E}_i^{(0)}$ and hypergraph incidence matrices $\mathbf{H}_u, \mathbf{H}_i$ are generated. After that, L DHCF layers are adopted to explore high-order correlation among users/items. Each layer includes two phases. **Phase 1: High-order Message Passing.** Here JHConv is applied for incorporating high-order structure information into user/item embeddings, respectively. **Phase 2: Joint Message Updating.** Shared learnable weight $W^{(l)}$ and bias $b^{(l)}$ are employed to jointly generate expressive embeddings for users and items. After L layer propagation, the initial embedding and the output embedding of each layer are combined to generate the final representation for the user/item.

Algorithm 1 DHCF embeddings generation algorithm

Input: Initial user embeddings $\mathbf{E}_u \in \mathbb{R}^{N \times C}$; initial item embeddings $\mathbf{E}_i \in \mathbb{R}^{M \times C}$; user-item bipartite graph incidence matrix $\mathbf{H} \in \{0,1\}^{N \times M}$; depth parameter L; trainable weight and bias $W^{(l)}, b^{(l)}, \forall l \in \{1, \ldots, L\}$; $\cdot||\cdot$ concatenation operation.

Output: User embeddings $\mathbf{E}_u^r$; Item embeddings $\mathbf{E}_i^r$.

// Initialization
$\mathbf{E}_u^{(0)} \leftarrow \mathbf{E}_u$
$\mathbf{H}_u = \mathbf{H}||\left(\mathbf{H}(\mathbf{H}^\top\mathbf{H})\right)$ ▷ users' hypergraph incidence matrix
$\mathbf{E}_i^{(0)} \leftarrow \mathbf{E}_i$
$\mathbf{H}_i = \mathbf{H}^\top||\left(\mathbf{H}^\top(\mathbf{H}\mathbf{H}^\top)\right)$ ▷ items' hypergraph incidence matrix
// DHCF layer propagation
for $l = 1 \ldots L$ **do**
  // High-order Message Passing
  $M_u^{(l)} \leftarrow \text{JHConv}(\mathbf{E}_u^{(l-1)}, \mathbf{H}_u)$ ▷ for users
  $M_i^{(l)} \leftarrow \text{JHConv}(\mathbf{E}_i^{(l-1)}, \mathbf{H}_i)$ ▷ for items
  // Joint Message Updating
  $\mathbf{E}_u^{(l)} \leftarrow W^{(l)} \cdot M_u^{(l)} + b^{(l)}$
  $\mathbf{E}_i^{(l)} \leftarrow W^{(l)} \cdot M_i^{(l)} + b^{(l)}$
end for
// Generate final user/item embeddings
$\mathbf{E}_u^r \leftarrow \mathbf{E}_u||\mathbf{E}_u^{(1)}||\mathbf{E}_u^{(2)}||\ldots||\mathbf{E}_u^{(L)}$
$\mathbf{E}_i^r \leftarrow \mathbf{E}_i||\mathbf{E}_i^{(1)}||\mathbf{E}_i^{(2)}||\ldots||\mathbf{E}_i^{(L)}$
return $\mathbf{E}_u^r, \mathbf{E}_i^r$;

B HYPERPARAMETER SETTINGS

In our experiments, we adopt the same train settings for all methods as shown in Table 1. The initial learning rate is 0.001, which will be decayed 0.5 in epoch 10, 40, 60, and 80.

Table 2 illustrates detailed hyperparameter settings of each model such as layer and embedding dimension. Note that in Table 2 **dim.** denotes the abbreviation of dimension. For all methods, we concatenate the output of each layer with the initial embeddings to generate the final representations of both users and items.

Table 1: Hyperparamter settings in the training stage.

Parameter	Value
Epochs	100
Initial learning rate	0.001
Learning rate milestone	[10, 40, 60, 80]
Learning rate decay rate	0.5
Batch size	128
Dropout	0.1
l_2 weight regularization	0.00001

Table 2: Hyperparamter settings for all models.

Model	Layer	Embedding dim.	Hidden dim.
BPR-MF	-	64	-
PinSage	2	64	[64, 64]
GCMC	3	64	[64, 64, 64]
NGCF	3	64	[64, 64, 64]
DHCF	1	64	[64]