Dynamic Hypergraph Learning for Collaborative Filtering

Chunyu Wei*
Alibaba Group
Bejing, China
weichunyu.wcy@alibaba-inc.com

Jian Liang*
Alibaba Group
Bejing, China
xuelang.lj@alibaba-inc.com

Bing Bai
Tencent Security Big Data Lab, Tencent Inc.
Bejing, China
icebai@tencent.com

Di Liu
Alibaba Group
Bejing, China
wendi.ld@alibaba-inc.com

ABSTRACT

Hypergraph-based collaborative filtering for recommendations has emerged as an important research topic due to its ability to model complex relations among users and items. However, most existing methods typically construct the hypergraph structures using heuristics (e.g., motifs and jump connections) based on existing graphs (e.g., user-item bipartite graphs and social networks). From a learning perspective, we argue that the fixed heuristic topology of hypergraph may become a limitation and thus potentially compromise the recommendation performance. To tackle this issue, we propose a novel dynamic hypergraph learning framework for collaborative filtering (DHLCF), which learns hypergraph structures and makes recommendations collectively in a unified framework. In the hypergraph learning process, we solve two main challenges, i.e., 1) optimization issue and 2) regularization issue. Firstly, we propose a differentiable hypergraph learner to adaptively learn the optimized hypergraph structures dynamically for the hypergraph convolutions during the training process. Secondly, to better regularize dynamic hypergraph learning, we introduce a novel hypergraph learning objective, which forces the learned hypergraphs to retain the original graph topology. Extensive experiments on public datasets from different domains are provided to show that our proposed model significantly outperforms strong baselines.

CCS CONCEPTS

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

KEYWORDS

collaborative filtering, hypergraph learning, recommender systems

∗ Equal contributions from both authors. This work is done when Chunyu Wei works as an intern at Alibaba.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557301

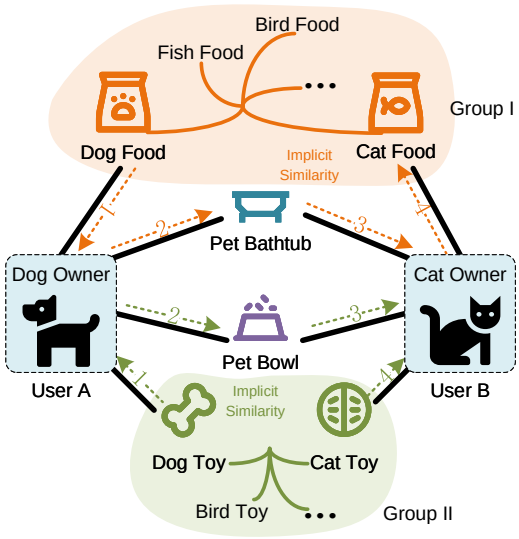

Figure 1: A possible illustration of high-order relations.

ACM Reference Format:
Chunyu Wei, Jian Liang, Bing Bai, and Di Liu. 2022. Dynamic Hypergraph Learning for Collaborative Filtering. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557301

1 INTRODUCTION

By predicting users' future interactions (e.g., clicks, ratings, purchases) and displaying the personalized items of interest in the most conspicuous positions, recommender systems can encourage more transactions and improve user satisfaction when they are browsing overwhelming information. Among all recommendation algorithms, collaborative filtering (CF) is one of the most popular and widely adopted methods in both industry and research communities [4, 8, 15, 22]. Recently, thanks to the powerful capability in modeling relational data, graph neural networks (GNNs) [24] have led to great success in CF. These GNNs-based methods formulate both users and items as entities on graphs, e.g., user-item bipartite graphs [7, 20, 22] and social relation graphs [17, 23], which exploit the explicit high-order relations defined by the initial pairwise links through multi-hop convolutions and show prominent performance.

However, besides these explicit high-order relations, there exist implicit high-order relations among entities, which these GNNs-based methods are hard to model constrained by the fixed graph structure. In this paper, the explicit high-order relation denotes the similarity between two entities shown by multi-hop connections within a certain number of steps on existing links. And by implicit high-order relation, we represent the similarity between two entities with similar collaborative information and local topology, regardless of whether they have connections within a certain number of steps or not. In Figure 1, we use an example to illustrate these two kinds of high-order relations. On a user-item bipartite graph, user A is a dog owner, and user B is a cat owner. Since they are both pet owners, they share many characteristics and have many items purchased in common, e.g., pet bathtubs and pet bowls, which connect them in 2-hop. And this means they have an explicit high-order relation. Meanwhile, they have also bought something special for their pets, like dog foods and dog toys for A's dog, as well as cat foods and cat toys for B's cat. Intuitively, these paired products (i.e., [dog food, cat food] and [dog toy, cat toy]) will have implicit high-order relations, since they have similar characteristics and local topology. However, for the traditional GNNs-based methods, it takes at least 4 steps of propagation for these paired products to pass their message to each other, which is computationally expensive and will bring too much noise. Besides the pairwise relation, different groups of products (e.g., Group I and Group II) may also share similar characteristics or local topological structure, which can hardly be well-modeled by vanilla GNNs. Similarly, users also have this kind of implicit high-order relations.

To provide a natural way to model complex high-order relationships, hypergraphs are proposed by generalizing the concept of edges to hyperedges, which are able to connect a flexible number of nodes [5]. This inspires researchers to combine GNNs and hypergraphs to enhance representation learning for recommendations [1, 3, 12, 21]. Nevertheless, existing research leveraging hypergraphs may have two limitations. Firstly, most existing hypergraph-based neural networks initially construct a hypergraph with hand-crafted rules based on the existing topology which may be sub-optimal, e.g., motifs presented in subgraphs and jump connections for K-hop nodes. Secondly, the hypergraph structure remains static throughout the training procedure. Thus in this sense, the construction of a hypergraph is only a pre-processing step, independent of the representation learning task. From a learning standpoint, we argue that the static hand-crafted hypergraph structure may hinder the performance of recommendation, since it cannot guarantee optimized higher-order relations for the ultimate downstream task and dynamic learning process.

To this end, we set out to learn the dynamic hypergraph structures (i.e., hyperedges) as well as the representations of users and items collectively in a unified framework. To achieve this goal, we need to address two notable challenges.

- **Optimization issue**. Although several approaches have been proposed to construct hypergraphs beyond the original topological constraints (e.g., DHGNN [13] utilizes k-means clustering to build the hypergraph), we argue that they are still model-free methods relying on heuristics, which always require strong assumptions of the entity distribution. These heuristic model-free methods may focus on similarities among entities that are suboptimal to the downstream tasks, thus failing to build optimal hypergraphs.
- **Regularization issue**. Hypergraphs provide a flexible way to model complex-high order relationships, however, it also brings a significant challenge for learning the hypergraph structures solely with the recommendation objective in a fully end-to-end manner, as this will make the learned hypergraph obscure the complex relations among entities and overfit the noise in the downstream recommendation task, especially when we update the hypergraph many times during the training. So how to regularize the learning process to capture effective signals and suppress noise becomes a key point for successful hypergraph structure learning.

In this paper, we propose a novel Dynamic Hypergraph Learning framework for Collaborative Filtering (DHLCF) to address the aforementioned challenges synergistically.

To address the optimization issue, we propose a differentiable lightweight multi-layer hypergraph learner, which can efficiently learn hypergraph structures dynamically at different layers alongside the training process. The differentiable learner can contrapuntally capture the complex entity correlations coupled with our downstream recommendation task in a learnable manner, which may outperform suboptimal heuristic similarities. By learning the dynamic hypergraphs at different layers, the following hypergraph convolutions are capable of exploiting implicit high-order relations of different semantic hierarchies.

To address the regularization issue, we integrate a hypergraph learning loss into the training process of the hypergraph learners, which encourages strongly connected nodes to be clustered together in the same hyperedge, with the intuition that correlated users or items tend to be in the same community on the original graph. The hypergraph learning loss acts as a regularization term and can be unified with the recommendation objective to be jointly optimized, which will significantly benefit the overall learning process.

The contributions of this paper are summarized as follows:

- We propose the DHLCF to learn hypergraph structures dynamically as well as the representations of users and items collectively for collaborative filtering. To the best of our knowledge, this is the first study on learning the hypergraph structure of both users and items for recommendations.
- We propose a novel hypergraph learning loss to regularize the hypergraph structure learning and show that the learned hypergraph structures for representation learning can significantly improve the recommendation performance.
- Experimental results show that our method outperforms the state-of-the-art methods on three benchmark datasets from different domains and demonstrate the effectiveness of dynamic hypergraph learning.

2 RELATED WORK

2.1 Hypergraph in Recommender Systems

Hypergraph learning is first introduced in [30] as a propagation process on hypergraph structure to tackle various problems. HGNN [5] is the first work proposing to perform a spectral hypergraph convolution to learn the hidden layer representation considering the

high-order data structure. To further improve the representation learning, research attention has been attracted for introducing hypergraphs to the recommendation area, which has greatly improved the modeling of high-order correlations among users and items. Bu et al. [3] use hypergraph to model various objects and relations in music social communities, and consider music recommendation as a ranking problem on this hypergraph, which is considered as the earliest attempt. More recent works concentrate on initially and heuristically constructing the user or item hypergraphs based on existing interaction data, and performing hypergraph convolution to capture the high-order correlations in the representations of users and items. Ji et al. [12] initially construct the user and item hypergraph with jump connection and propose a jump hypergraph convolution method to support the explicit propagation of high-order correlations. Bai et al. [1] adopt hypergraph to represent the short-term item correlations and applies multiple hypergraph convolution layers to capture multi-order connections in the hypergraph. Social networks bring more complex user correlations, which is proved to be helpful to alleviate the data sparsity issue and improve the recommendation performance. Methods like HMF [29], LBSN2Vec [26] and MHCN [27] borrow the strengths of hypergraph neural networks to incorporate the user-item interactions and social relations simultaneously, with the main difference being the way to construct the hypergraph. LBSN2Vec [26] builds hyperedges by jointly sampling friendships and check-ins with random walks. MHCN [27] utilizes a set of motifs to depict triangular structures in social networks, which guides the hypergraph construction. However, existing hypergraph-based methods for recommendation usually require initial heuristically-constructed hypergraphs and they remain fixed throughout the training procedure, while in this paper, we explore how to learn better hypergraph structures dynamically during the training.

2.2 Hypergraph Structure Learning

Researchers recently investigate the potential of learning hypergraph structure and making inferences on the hypergraph, which can be divided into feature-based methods and representation-based methods. The feature-based methods explore the nearest neighbors in the feature space for constructing hyperedges, based on k-NN or a search radius. For example, Huang et al. [9] select a set number of nearest neighbors for each vertex to generate a hyperedge. However, it is hard to determine an optimal number of nearest neighbors or the search radius, which may be sensitive to noise and limit the representation learning. Different from the feature-based methods, representation-based methods [16, 31] aim to exploit the relations among vertices through feature reconstruction. More recent works focus on improving the hypergraph structure by learning the weights of hyperedges, where the hyperedges could have different importance on representation learning. For example, an l_2 regularizer on the weights is introduced in [6] to learn optimal hyperedge weights. And Hwang et al. [10] propose to model the correlations among hyperedges and assign similar weights to highly correlated hyperedges. However, they cannot repair the connections which are inappropriate or even contain errors because the hypergraph structure is predefined and static in these works.

To tackle this problem, the concept of dynamic hypergraph structure learning was proposed. Zhang et al. [28] use the raw input data to optimize the hypergraph structure iteratively. Furthermore, Jiang et al. [13] utilize the k-NN method and k-means clustering method to update hypergraph structure based on both local and global features. Though these attempts sought to adjust the hypergraph structure under traditional learning settings, such procedures cannot be readily integrated into end-to-end deep frameworks due to the undifferentiable nature, not to mention being generalized to the recommender systems. In this paper, we overcome the above issues by proposing an end-to-end DHLCF framework to jointly learn the hypergraph structure and entity representation for recommendation. To the best of our knowledge, this is the first attempt to learn the dynamic hypergraph structure for recommendation.

3 PRELIMINARIES

Let $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ denotes the set of users, and let $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ denotes the set of items. In the recommendation scenario, we typically use a binary matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ to store user-item interactions (e.g., purchases and clicks), where $r_{ui} = 1$ indicates that user u consumed item i while $r_{ui} = 0$ means that item i is unexposed to user u or user u is not interested in item i. Here we represent interaction data as a user-item bipartite graph $\mathcal{G}$ and the adjacency matrix $\mathbf{A}_{\mathcal{G}}$ can be formulated as follows:

$$\mathbf{A}_{\mathcal{G}} = \begin{bmatrix} 0 & \mathbf{R} \\ \mathbf{R}^T & 0 \end{bmatrix}. \tag{1}$$

With respect to the adjacency matrix $\mathbf{A}_{\mathcal{G}}$, the degree matrix $\mathbf{D}_{\mathcal{G}} \in \mathbb{N}^{(m+n)\times(m+n)}$ is a diagonal matrix, in which each entry $\mathbf{D}_{\mathcal{G}}[i,i]$ denotes the number of nonzero entries in the i-th row of $\mathbf{A}_{\mathcal{G}}$.

For a graph structure, an edge only connects two vertices, while in a hypergraph, a hyperedge connects two or more vertices. Let $\mathcal{HG} = (\mathcal{V}, \mathcal{E})$ denote a hypergraph, where $\mathcal{V}$ is the vertex set containing M vertices and $\mathcal{E}$ is the hyperedge set containing K hyperedges. The hypergraph can be represented by an incidence matrix $\mathbf{H} \in \{0,1\}^{M \times K}$, where $h_{ve} = 1$ if the hyperedge $e \in \mathcal{E}$ connects a vertex $v \in \mathcal{V}$, otherwise $h_{ve} = 0$. For a vertex $v \in \mathcal{V}$ and a hyperedge $e \in \mathcal{E}$, their degrees can be defined as: $d(v) = \sum_{e \in \mathcal{E}} h_{ve}$, and $d(e) = \sum_{v \in \mathcal{V}} h_{ve}$, respectively. And then we use two diagonal matrices $\mathbf{D}_{\mathcal{HG}} \in \mathbb{R}^{M \times M}$ and $\mathbf{L} \in \mathbb{R}^{K \times K}$ to represent the vertex degree matrix and the hyperedge degree matrix, respectively, where the diagonal elements $\mathbf{D}_{\mathcal{HG}}[v,v] = d(v)$ and $\mathbf{L}[e,e] = d(e)$.

In this paper, we use bold capital letters to denote matrices and bold lowercase letters to denote vectors.

4 METHODOLOGY

4.1 Model Architecture

In Figure 2, the schematic overview of DHLCF is illustrated. At a high level, we introduce its architecture from left to right. Each layer of DHLCF consists of three core processes. (1) DHLCF first performs graph convolution directly on the user-item bipartite graph, which models the explicit high-order relations and effectively captures the collaborative signals and the local topology into the embeddings of users and items. (2) Based on these embeddings, DHLCF adaptively discovers groups of implicitly correlated users and items separately, and constructs dynamic user hypergraph and item hypergraph. (3)

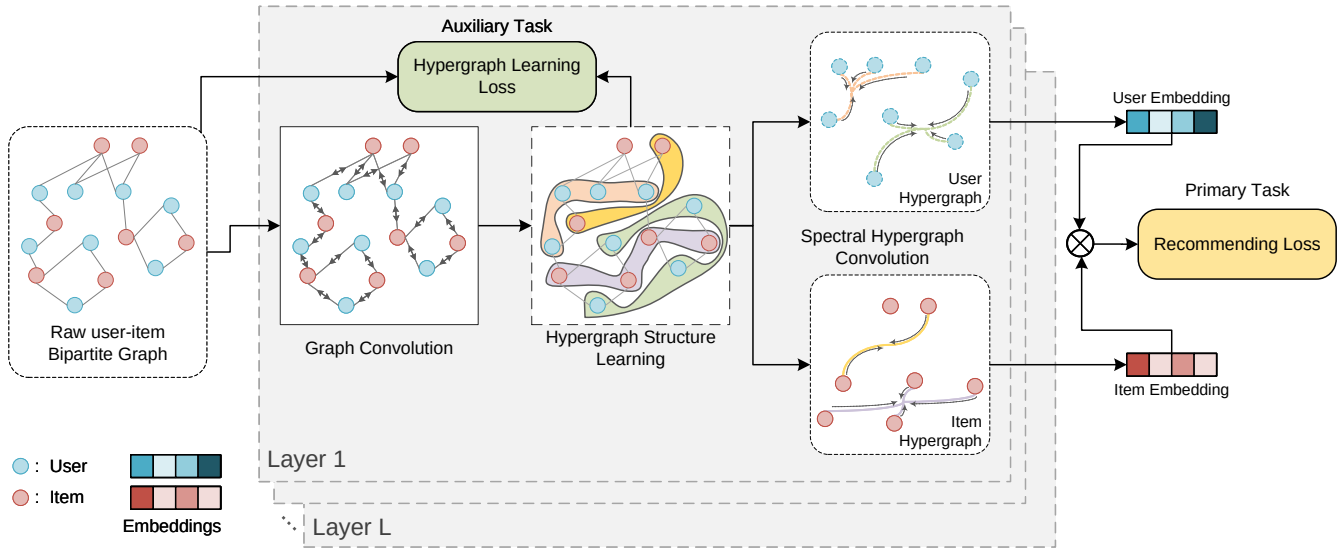

Figure 2: The overview of the model structure of DHLCF. Best view in color.

Given the embeddings in (1) and the learned dynamic hypergraphs in (2), we refine the embeddings of users and items separately with hypergraph convolution, which enables the information to interact among users (or items) in the same hyperedge. These three processes are alternatively and sequentially performed, and support each other to simultaneously improve the hypergraph structure learning and recommendation performance.

4.2 Graph Convolution on User-item Bipartite Graph

The raw user-item bipartite graph explicitly represents the user-item interaction information as its links, and a node's local structure (i.e., the topology of its multi-hop neighbors) is shown to encode a user's preference or an item's characteristic [22]. To capture the collaborative signal alongside the local topology, we exploit the high-order connectivity following the recent advance in LightGCN [7]. To initialize, each user and item in the raw bipartite graph is associated with an ID embedding, which can be denoted with two learnable parameter matrices $\mathbf{E}_u \in \mathbb{R}^{m\times d}$ and $\mathbf{E}_i \in \mathbb{R}^{n\times d}$, respectively, where d is the embedding size. Different from the traditional GCN propagation rule, we remove the learnable matrix for linear transformation and the nonlinear activation function (e.g.leaky ReLU) following the suggestion in [7]. The explicit propagation process can be formulated as follows:

$$\mathbf{E}^{(0)} = [\mathbf{E}_u, \mathbf{E}_i] = [\mathbf{e}_{u_1}, \ldots, \mathbf{e}_{u_m}, \mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_n}], \tag{2}$$

$$\hat{\mathbf{E}}^{(k)} = (\mathbf{D}_{\mathcal{G}}^{-\frac{1}{2}}\mathbf{A}_{\mathcal{G}}\mathbf{D}_{\mathcal{G}}^{-\frac{1}{2}})\mathbf{E}^{(k-1)}, k \in \mathbb{N}^{+}, \tag{3}$$

where $\mathbf{E}^{(k-1)} = [\mathbf{E}_u^{(k-1)}, \mathbf{E}_i^{(k-1)}]$ is the output of the previous DHLCF layer or the initial $\mathbf{E}^{(0)}$. Each round of the propagation explicitly updates the node's representation in the bipartite graph by its higher-order relations. It's worth noting that we do not directly stack multiple graph convolution layers together, instead, the graph convolution provides updated representations for the dynamic hypergraph learning, which will be elaborated below.

4.3 Dynamic Hypergraph Structure Learning

Besides the collaborative signal, graph convolution in Eq 3 is capable of capturing local structural information of graph according to GIN [25], and the topological similarity of the nodes (i.e., users and items) is always accompanied by the characteristic similarity. To break the message passing limit of the fixed and heuristic graph structure (i.e., the user-item bipartite graph), we propose to learn implicit high-order relations (i.e., hyperedges) for users and items separately based on the updated $\hat{\mathbf{E}}^{(k)}$ in Eq 3. For simplicity, we omit the layer number (k) in the symbols below, since we mainly discuss the subprocesses in the same DHLCF layer.

Following the recent advance in spectral clustering [2], we believe the collaborative signal and the topological information are adequate for computing the hyperedge assignments, which should contain the implicit similarity of user preference or item characteristic. So we compute the hyperedge assignment of users (or items) using a multi-layer perceptron (MLP) with softmax on the output layer, which maps each user (or item) representation $\mathbf{e}_{u_k}$ (or $\mathbf{e}_{i_k}$) into the k-th row of a soft hypergraph incidence matrix $\hat{\mathbf{H}}_u$ (or $\hat{\mathbf{H}}_i$):

$$\hat{\mathbf{H}}_u = Softmax(ReLU(\hat{\mathbf{E}}_u\mathbf{W}_{u,1})\mathbf{W}_{u,2}), \tag{4}$$

$$\hat{\mathbf{H}}_i = Softmax(ReLU(\hat{\mathbf{E}}_i\mathbf{W}_{i,1})\mathbf{W}_{i,2}), \tag{5}$$

where $\mathbf{W}_{u,1} \in \mathbb{R}^{d\times d}$, $\mathbf{W}_{u,2} \in \mathbb{R}^{d\times K}$, $\mathbf{W}_{i,1} \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_{i,2} \in \mathbb{R}^{d\times K}$ are trainable weight matrices, in which K is the number of hyperedges to learn. By using $Softmax$, we softly assign a vertex to multiple hyperedges with different probabilities. It's also possible to apply the Gumbel-softmax trick [11] to make it a differentiable sampling procedure, but we found $Softmax$ performs better.

Although we couple the dynamic hypergraph learning process and the recommendation process together, we find relying solely on the recommendation objective can not well guide the hypergraph

structure learning and may lead to overfitting the downstream noise, especially when we learn multiple hypergraphs throughout the training, since the hypergraph learning is not intrinsically adaptive to the recommendation task. We argue that the learned hypergraph should somehow be constrained by the initial interaction history. Following [2], we relax the minCUT problem [19] and propose the following hypergraph learning loss for the dynamic hypergraph learning process:

$$\mathcal{L}_{\hat{\mathbf{H}}_u} = -Tr(\hat{\mathbf{H}}_u^T \mathbf{R}\mathbf{R}^T \hat{\mathbf{H}}_u) + \|\hat{\mathbf{H}}_u^T \hat{\mathbf{H}}_u\|_F, \tag{6}$$

$$\mathcal{L}_{\hat{\mathbf{H}}_i} = -Tr(\hat{\mathbf{H}}_i^T \mathbf{R}^T \mathbf{R} \hat{\mathbf{H}}_i) + \|\hat{\mathbf{H}}_i^T \hat{\mathbf{H}}_i\|_F, \tag{7}$$

where $Tr(\cdot)$ denotes the trace of a matrix, and $\|\cdot\|_F$ indicates the Frobenius norm. For the loss $\mathcal{L}_{\hat{\mathbf{H}}_u}$, the multiplication operation $\mathbf{R}\mathbf{R}^T$ defines an $m \times m$ 1-order reachable matrix of users, of which each entry represents the number of the corresponding user pair's commonly consumed items. So $Tr(\hat{\mathbf{H}}_u^T \mathbf{R}\mathbf{R}^T \hat{\mathbf{H}}_u)$ is to sum all commonly consumed items of every user pair in the same hyperedge, which can be used to represent the pairwise similarity in a hyperedge. The first term can encourage those users sharing similar interaction history to be clustered in the same hyperedge. However, this term is a non-convex function, which may lead to a local minimum. For example, when we put all the users in the same hyperedge, the first term reaches its minimum. So we add the second term to penalize such situations, which encourages all learned hyperedges to contain different vertices and be of a similar size. The loss $\mathcal{L}_{\hat{\mathbf{H}}_i}$ is symmetrically similar to $\mathcal{L}_{\hat{\mathbf{H}}_u}$, so we omit the analysis here.

4.4 Hypergraph Convolution on Learned Hypergraph

After obtaining the learned hypergraphs from the learning process, we describe how to incorporate the implicit high-order relations among users and items, and update the users' and items' representations. Inspired by HGNN [5], we follow the conventional schema of the spectral hypergraph convolution, which is composed of vertex convolution and hyperedge convolution. The former aggregates vertex features to the hyperedge and then the latter aggregates adjacent hyperedge features to the centroid vertex.

We define our hypergraph convolution on the learned hypergraph $\hat{\mathbf{H}}_u$ and $\hat{\mathbf{H}}_i$ as follows:

$$\mathbf{E}_u = \hat{\mathbf{H}}_u \mathbf{L}_u^{-1} \hat{\mathbf{H}}_u^T \hat{\mathbf{E}}_u + \hat{\mathbf{E}}_u \tag{8}$$

$$\mathbf{E}_i = \hat{\mathbf{H}}_i \mathbf{L}_i^{-1} \hat{\mathbf{H}}_i^T \hat{\mathbf{E}}_i + \hat{\mathbf{E}}_i \tag{9}$$

where $\mathbf{L}_u$ and $\mathbf{L}_i$ denote the hyperedge degree matrices of the learned user hypergraph $\hat{\mathbf{H}}_u$ and item hypergraph $\hat{\mathbf{H}}_i$, respectively, which help re-scale the output embeddings. The matrix multiplication $\hat{\mathbf{H}}_u^T \hat{\mathbf{E}}_u$ and $\hat{\mathbf{H}}_i^T \hat{\mathbf{E}}_i$ perform the vertex convolution, propagating the vertex messages to the hyperedges. And the following $\hat{\mathbf{H}}_u$ and $\hat{\mathbf{H}}_i$ aggregate the hyperedge information to update the vertices.

Compared with traditional hypergraph convolution [5, 12, 27], there are three main differences. First, we remove the learnable matrix for linear transformation and the nonlinear activation function consistent with the previous graph convolution as suggested in [7], since it's proved to be of little help in the scenario of collaborative filtering with only user and item IDs. Second, we remove the vertex degree matrix to re-scale the propagation process. This is because in Eq 4 and 5, unlike the usual hypergraph, we already utilized the $Softmax$ operation to interpret the hyperedge assignment as a probability distribution and it is equivalent to the normalization with the degree matrix. Third, we take skip-connection into consideration to retain the original collaborative signal and local topology information from the graph convolution, which can also help avoid the problem of gradient vanishing.

4.5 Recommending

The above three subprocesses are sequentially performed in a single integrated DHLCF layer, and we apply a hierarchical learning strategy to stack multiple layers, in which we learn the hypergraph structure dynamically in order to capture different hierarchies of high-order information (i.e., different patterns to cluster users or items in a hyperedge). We illustrate the detailed procedure of one single layer in Algorithm 1:

Algorithm 1 DHLCF Layer k

Input: User-item interaction matrix $\mathbf{R}$;
User-item bipartite graph $\{\mathbf{A}_{\mathcal{G}}, \mathbf{D}_{\mathcal{G}}\}$;
Last layer's output embedding $\mathbf{E}^{(k-1)}$ or the initial $\mathbf{E}^{(0)}$.
Output: Embedding $\mathbf{E}^{(k)}$;
Hypergraph learning loss $\mathcal{L}_{\hat{\mathbf{H}}_u^{(k)}}, \mathcal{L}_{\hat{\mathbf{H}}_i^{(k)}}$.

1: Compute the updated $\hat{\mathbf{E}}^{(k)}$ by Graph Convolution. (Eq 3)
2: Learn the k-th hypergraph structure $\hat{\mathbf{H}}_u^{(k)}, \hat{\mathbf{H}}_i^{(k)}$. (Eq 4, 5)
3: Compute hypergraph learning losses $\mathcal{L}_{\hat{\mathbf{H}}_u^{(k)}}, \mathcal{L}_{\hat{\mathbf{H}}_i^{(k)}}$. (Eq 6, 7)
4: Compute $\mathbf{E}_u^{(k)}, \mathbf{E}_i^{(k)}$ by Hypergraph Convolution on $\hat{\mathbf{H}}_u^{(k)}, \hat{\mathbf{H}}_i^{(k)}$. (Eq 8, 9)
5: Return $\mathbf{E}^{(k)} = [\mathbf{E}_u^{(k)}, \mathbf{E}_i^{(k)}]$, $\mathcal{L}_{\mathbf{H}_u^{(k)}}, \mathcal{L}_{\mathbf{H}_i^{(k)}}$.

After stacking L DHLCF layers, we obtain multiple representations of users and items, $\{\mathbf{E}_u^{(0)}, \ldots, \mathbf{E}_u^{(L)}\}$ and $\{\mathbf{E}_i^{(0)}, \ldots, \mathbf{E}_i^{(L)}\}$. Since the representations at different layers contain different semantic hierarchies, we simply concatenate them to construct the final representations, which can be formulated as follows:

$$\bar{\mathbf{E}}_u = \mathbf{E}_u^{(0)} || \ldots || \mathbf{E}_u^{(L)}, \ \bar{\mathbf{E}}_i = \mathbf{E}_i^{(0)} || \ldots || \mathbf{E}_i^{(L)}, \tag{10}$$

where $||$ denotes the concatenation operation. Note that other function like mean-pooling, max-pooling and attention network can be adopted, but we found that concatenation performs better in our scenario. Finally, we perform an inner product to estimate the user u_j's preference for the target item i_k:

$$\hat{r}_{jk} = \bar{\mathbf{e}}_{u_j}^T \bar{\mathbf{e}}_{i_k}, \tag{11}$$

where $\bar{\mathbf{e}}_{u_j}$ is u_j's corresponding row in $\bar{\mathbf{E}}_u$, and similarly $\bar{\mathbf{e}}_{i_k}$ is i_k's corresponding row in $\bar{\mathbf{E}}_i$.

4.6 Optimization

For the recommendation part, we employ the Bayesian Personalized Ranking (BPR) loss [18], which is a pairwise loss that assumes the observed interactions should be assigned higher prediction values than unobserved ones. The objective function is as follows:

$$\mathcal{L}_{rec} = \sum_{(u,i,j) \in O} -ln\sigma(\hat{r}_{ui} - \hat{r}_{uj}), \tag{12}$$

where $O = \{(u, i, j)|(u, i) \in \mathcal{R}^+, (u, j) \in \mathcal{R}^-\}$ is the pairwise training data, in which $\mathcal{R}^+$ denotes the observed interactions, and $\mathcal{R}^-$ denotes the unobserved interactions.

Finally, we unify the objectives of the recommendation task and the dynamic hypergraph learning task for joint learning. The overall objective is defined as:

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda \sum_{k=1}^{L} (\mathcal{L}_{\hat{\mathbf{H}}_u^{(k)}} + \mathcal{L}_{\hat{\mathbf{H}}_i^{(k)}}) + \beta \|\Theta\|_2^2, \tag{13}$$

where $\mathcal{L}_{\hat{\mathbf{H}}_u^{(k)}}$ and $\mathcal{L}_{\hat{\mathbf{H}}_i^{(k)}}$ denote the hypergraph learning losses at layer k. The last term is an L_2 regularization to prevent overfitting. λ and β are the hyper-parameters controlling the effect strength of the hypergraph learning task and L_2 regularization, respectively.

4.7 Time Complexity

As illustrated in Algorithm 1, the computation of each DHLCF layer consists of four parts. (1) In the graph convolution process, the matrix multiplication has computational complexity less than $O(|\mathbf{A}^+_{\mathcal{G}}|d) = O(|\mathbf{R}^+|d)$, where $|\mathbf{A}^+_{\mathcal{G}}|$ denotes the number of nonzero elements in $\mathbf{A}_{\mathcal{G}}$ and d is the embedding size. (2) The hypergraph structure learning computes two separate soft incidence matrices for users and items respectively with two shallow networks. The complexity for this part is $O((md^2 + mdK) + (nd^2 + ndK)) = O((m + n)d(d + K)) < O(K^2(m + n))$, since we usually set $K > d$. (3) As for the hypergraph learning loss for users, the first term is to compute the trace product, of which only the diagonal elements of $(\hat{\mathbf{H}}_u^T\mathbf{R})(\hat{\mathbf{H}}_u^T\mathbf{R})^T$ are involved. The time complexity of the first term will be $O(|\mathbf{R}^+|K) + O(Kn))$, and $O(Kn))$ can be omitted due to $n \ll |\mathbf{R}^+|$. The second term is a self-multiplication and its time complexity is $O(mK^2)$. Hence, the time complexity for computing the loss of user hypergraph learning is $O(|\mathbf{R}^+|K) + O(mK^2)$, and symmetrically it is $O(|\mathbf{R}^+|K) + O(nK^2)$ for the item hypergraph learning. (4) For hypergraph convolution, the complexity is $O(mKd)$ and $O(nKd)$ for the user $\mathcal{HG}$ and the item $\mathcal{HG}$, respectively. Empirically, if we stack l DHLCF layers, the overall theoretical time complexity for all user and items in one full step of model training is less than $O(l(m + n)K^2) + O(l|\mathbf{R}^+|K)$, since we usually set $K > d$. So the time complexity of our model is at the same level as that of previous GNNs-based recommendation models. It's worth noting that for inference, we don't need to compute the hypergraph learning loss in part (3), which further reduces the overall time complexity.

5 EXPERIMENTS

To evaluate the effectiveness of the proposed method, we have conducted experiments on three public datasets. Ablation studies are also presented for better investigation of our proposed model.

5.1 Dataset Description

Three public available datasets are employed in our experiments, i.e., *Yelp*, *Gowalla* and *LastFM-2K*. The statistical details of these datasets are presented in Table 1.

Yelp [27]. This is an online location-based review system, on which users can express their experience (i.e., local businesses) through the form of reviews and ratings. We construct our dataset by using each review as evidence that a user's consumption. We filtered out users who post less than 5 reviews and businesses which receive less than 20 reviews.

Table 1: Descriptive statistics of the datasets.

Dataset	#Users	#Items	#Interactions	Density
Yelp	19,539	21,266	450,884	0.00109
Gowalla	4,473	3,564	156,322	0.00981
LastFM-2K	1,892	1,865	65,789	0.01864

Gowalla. [1] This is the check-in dataset obtained from Gowalla, where users share their locations by checking-in. To ensure the quality, we collected a set of check-in logs in a temporal interval between 2010-05-01 and 2010-10-01 and treat each check-in as a consumption. We also filter out those users with less then 5 check-ins and places with less than 10 reviews.

LastFM-2K. [2] This dataset contains music streaming history from a set of 2K users on Last.fm. Similarly, we retain users and items with at least ten interactions.

For each dataset, we randomly select 80% of the historical interactions of each user as the training set, 10% of those as the validation set, and the remaining 10% as the test set.

5.2 Experimental Setup

Evaluation metrics. To evaluate the performance of all methods, we adopt a ranking-based metric namely Normalized Discounted Cumulative Gain@k (NDCG@k) and a relevancy-based metric Hit Ratio@k (RECALL@k). The NDCG@k metric accounts for the position of the hits by assigning higher scores to hits at top ranks. The RECALL@k metric measures the percentage of relevant items selected out of all the relevant items for the user. Both of the adopted metrics can be formulated as follows:

$$\text{NDCG@}k = \frac{1}{R_N} \sum_{i=1}^{N} \frac{2^{rel_i - 1}}{\log_2(1 + i)}, \tag{14}$$

$$\text{RECALL@}k = \frac{\sum_{i=1}^{k} rel_i}{|\mathcal{I}^u_{test}|}, \tag{15}$$

where k is the size of the recommendation list, $rel_i = 0$ or 1 denotes whether the item at the rank i is in the test set or not, and the R_N term indicates the maximum possible cumulative component through ideal ranking. $|\mathcal{I}^u_{test}|$ is the number of relevant items in the testing set for user u. Especially, as suggested by Krichene and Rendle [14], we perform item ranking on all the candidate items instead of the sampled item sets to calculate the values of these two metrics, which guarantees that the evaluation process is unbiased.

Baselines. We compare our DHLCF with the following six baseline methods to evaluate the performance, which we can categorize into three classes: (1) MF-based methods; (2) GNNs-based methods; and (3) hypergraph-based methods.

[1] Data set available from https://snap.stanford.edu/data/loc-gowalla.html
[2] Data set available from http://files.grouplens.org/datasets/hetrec2011/

Table 2: Comparison among models. Boldface denotes the highest score and underline indicates the best result of the baselines.

Model	Yelp				Gowalla				LastFM-2K			
	NDCG@10	RECALL@10	NDCG@20	RECALL@20	NDCG@10	RECALL@10	NDCG@20	RECALL@20	NDCG@10	RECALL@10	NDCG@20	RECALL@20
BPRMF	4.33%	9.45%	6.51%	18.18%	4.39%	9.73%	6.64%	18.56%	1.44%	3.06%	2.35%	6.48%
NCF	3.66%	7.74%	5.31%	14.34%	3.41%	7.16%	4.75%	12.55%	1.11%	2.40%	1.80%	5.14%
NGCF	4.00%	8.63%	5.82%	15.87%	4.80%	10.59%	6.75%	18.27%	2.21%	4.65%	3.25%	8.57%
DHCF	3.48%	7.45%	5.23%	14.47%	3.88%	8.50%	5.52%	14.94%	1.54%	3.36%	2.36%	6.46%
LightGCN	4.71%	10.30%	7.11%	19.77%	4.98%	11.15%	7.31%	20.21%	2.24%	4.89%	3.38%	9.39%
MHCN	– –	– –	– –	– –	– –	– –	– –	– –	2.23%	5.02%	3.47%	9.55%
DHLCF	**5.41%**	**11.66%**	**7.92%**	**21.77%**	**5.71%**	**12.44%**	**8.20%**	**22.32%**	**2.82%**	**6.10%**	**4.03%**	**10.99%**
Improv.	+14.91%	+13.20%	+11.47%	+10.15%	+14.67%	+11.59%	+12.09%	+10.48%	+25.67%	+21.52%	+16.06%	+15.13%
p-value	1.16e-4	3.41e-4	4.07e-5	3.78e-5	4.82e-3	2.46e-3	2.12e-4	4.11e-4	3.70e-3	2.46e-3	1.58e-3	2.98e-3

† MHCN requires extra social information, so we use *LastFM-2K* to justify that our DHLCF can better extract high-order relations among users without social information.

- **BPRMF** [18]. This method improves matrix factorization (MF) with the BPR objective function. It is a highly competitive method for implicit feedback based recommendation.
- **NCF** [8]. This is a deep learning based framework that combines matrix factorization (MF) with a multilayer perceptron model (MLP) for item ranking.
- **NGCF** [22]. This is a graph-based model, which first encodes the collaborative signal into the user-item interaction graph structure and adopts multiple graph convolution to explore high-order connectivity.
- **LightGCN** [7]. This is the state-of-the-art GCN-based general recommendation model that leverages the user-item proximity to learn node representations and generate recommendations.
- **DHCF** [12]. This is a recent hypergraph convolutional method that models the high-order correlations among users and items for general recommendation.
- **MHCN** [27]. This is a social-based hypergraph convolutional network for recommendation by exploiting multiple types of high-order user relations under a multi-channel setting.

Note that **MHCN** is a social-based recommendation framework, thus it's not applicable to some datasets without social information.

Hyper-parameter settings. We initialize the latent vectors of both users and items with small random values for all models. The parameters for baseline methods are initialized as in the original papers, and are then carefully tuned to achieve optimal performances. For a fair comparison, the dimensions of both the user and item latent factors are all fixed to 64. We use Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e^{-8}$ to optimize all these methods. The batch size is set to 200. We set $\lambda = 0.1$ and $\beta = 0.001$ for the coefficients in Eq 13. The learning rate is set as 0.005 and decayed at the rate of 0.9 every five epochs. Sec 5.5 reports detailed analyses of different parameters and we adopt the best settings for experiments in Sec 5.3 and Sec 5.4.

5.3 Performance Comparisons

We summarize the performance of different algorithms in terms of NDCG@k and RECALL@k ($k = 10, 20$) over three datasets in Table 2. The experimental results demonstrate that DHLCF outperforms other methods on all evaluation metrics. Besides, we can draw the following conclusions:

- DHLCF consistently yields the best performance on all datasets. In particular, DHLCF improves over the strongest baseline with respect to NDCG@10 by 14.91%, 14.67%, and 25.67% in Yelp, Gowalla, and LastFM-2K, respectively. These improvements demonstrate that DHLCF is capable of modeling the comprehensive high-order relations among users and items through its dynamic hypergraph learning. We conduct two-sample t-tests and p-value < 0.05 indicates that the improvements of our DHLCF are statistically significant.
- Specifically, when both stacking 3 layers as in the original papers [7, 22], DHLCF outperforms both NGCF and LightGCN by a large margin. These results verify that only performing message passing on the given initial topological structure is suboptimal for learning the representations of users and items. By contrast, our model can better capture the implicit high-order relations by learning hyperedges to "skip-connect" similar nodes and share their features.
- Without incorporating the social information, our DHLCF still performs better than the social-based hypergraph convolution method MHCN. MHCN models the complex user high-order relations by distinguishing nodes that have specific triangular relations in social networks, while DHLCF does not rely on any hand-crafted rules and learns these implicit high-order relations. The results demonstrate that a learned hypergraph structure has more vital expressiveness for capturing the flexible dependencies among users.
- The GNNs-based methods consistently outperform the MF-based models. These results verify that incorporating the explicit high-order relations on the user-item bipartite graph is essential to capture the collaborative signals, which also justify the skip-connection in Eq. 8 and Eq. 9.
- Although DHCF is also a hypergraph-based method, consistent with the results in [27], we are also unable to reproduce the superiority reported in its original paper [12] on these datasets. According to Yu et al. [27], the k-order reachable rule to construct hypergraph in DHCF will create a very dense incidence matrix, which brings heavy computation and over-smoothing problems, especially on large datasets.

5.4 Ablation Studies

We conduct ablation studies to validate the effectiveness of our proposed hypergraph learning and the hypergraph learning loss.

Investigation of Hypergraph Learning. To investigate the effect of dynamic hypergraph learning, we consider the following variants of DHLCF with different hypergraph constructing strategies:

Table 3: Effect of dynamic hypergraph Learning. The results of NDCG@10 and RECALL@10 are similar hence omitted.

Variation	Gowalla		Yelp	
	NDCG@20	RECALL@20	NDCG@20	RECALL@20
DHLCF$_{knn}$	7.61%	20.92%	7.17%	19.72%
DHLCF$_{2-order}$	7.09%	19.28%	3.14%	8.82%
DHLCF	**8.20%**	**22.32%**	**7.92%**	**21.77%**

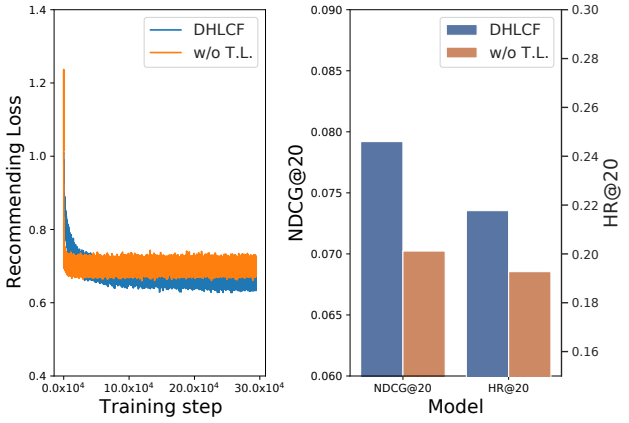

Figure 3: Effect of hypergraph learning loss on *Yelp*

- **DHLCF**$_{knn}$: This is a variant model in which the learned hypergraph is replaced by the product of k-NN clustering method as proposed in DHGNN [13] based on the graph convolution output in Sec 4.2.
- **DHLCF**$_{2-order}$: This is a variant model in which the learned hypergraph is replaced by the fixed 2-order reachable matrix proposed in [12].

Table 3 summarizes the performance of different variants. The hypergraphs, constructed by the heuristic k-NN clustering and the hand-crafted 2-order reachable rule, both compromise the capability of extracting the implicit high-order relations. According to the results, DHLCF$_{knn}$ also yields comparable performance compared with other baselines. This may be because the k-nn operation on the graph convolution outputs can also cluster users (or items) with similar collaborative signals and local typologies together in the same hyperedge. However, it's not learnable and thus cannot act as an auxiliary task to improve the primary recommending task, which limits its performance. Also, k-nn requires to compute an expensive global search on all the users (or items), which also limits its scalability to update the hypergraph structure at each layer. By contrast, DHLCF$_{2-order}$ has poor performance, especially on the large dataset. A possible reason is that it only exploits explicit 2-order relations which have been considered in the earlier graph convolution, and this duplication causes over-smoothing problem.

Investigation of the hypergraph learning loss. In DHLCF, we adopt a hypergraph learning loss to help guide the learning of hypergraph structure. To investigate the effectiveness of this loss, we introduce a variant of DHLCF by simply removing the hypergraph learning loss and training the model in a fully end-to-end manner. For convenience, we term the variant as **w/o T.L.**. We then run **DHLCF** and **w/o T.L.** on *Yelp* to observe the performance changes. Figure 3 shows the recommending loss w.r.t. the number of the training steps, from which we can observe that the loss of the framework trained in a fully end-to-end manner has a faster descent at the very beginning and turns to a steady-state very soon. However, with the hypergraph learning loss, the recommending loss of our DHLCF appears to have a declining trend after an initial sharp drop, instead of getting an early-stop, which proves that the hypergraph learning loss can help our framework converge to a better local optimum compared with the fully end-to-end framework. This might also be the reason why **DHLCF** has better performance than **w/o T.L.**, as illustrated by the right part of Figure 3.

5.5 Parameter Sensitivity Analysis

Effect of Layer number. To investigate whether DHLCF can benefit from multiple stacked layers and find the optimal depth L, we vary the model depth L from 0 to 6 on different datasets *Yelp* and *Gowalla*. Experimental results are shown in Figure 4 and we use DHLCF-l to indicate the model with l layers. We have the following observations:

- DHLCF-1 has a significant improvement compared to DHLCF-0. Note that when the layer number is 0, the model degenerates to **BPRMF** [18] and the experimental results are consistent with Table 2. We attribute the improvement to the fact that, compared with DHLCF-0 which only encodes the implicit collaborative signals, the DHLCF-1 explicitly models the connectivity of user-item (from user-item bipartite graph), user-user, and item-item (from learned hypergraphs).
- We can observe that increasing the depth of DHLCF substantially enhances the recommendation performance. The best performance is achieved when the layer number is 4 for *Yelp* and 3 for *Gowalla*. This demonstrates that when increasing DHLCF layers, the model not only explicitly captures higher-order collaborative signals from multi-hop connectives, but also can exploit implicit high-order information in different aspects by learning different hypergraph structures hierarchically. A visualization example of the learned hyperedges in different layers is shown in Section 6.
- Our model can apply a much deeper structure, compared with other hypergraph-based models like **DHCF** (1 layer deep) [12] and **MHCN** (2 layers deep) [27]. A possible reason is that we learn and update the hypergraph structure at each layer instead of using a fixed structure, which brings new information to capture for each node with the increase of depth. Thus the common over-smoothing of stacking multiple layers can be significantly alleviated in our model.

Effect of Hyperedge number. This paragraph investigates the influence of parameter K which controls the number of hyperedges to learn. we vary K from 10 to 160 while keeping other parameters fixed. We present the experimental results of NDCG@20 and RECALL@20 on *Yelp* and *Gowalla* in Figure 5 and results of other metrics are omitted due to the space limitation. From Figure 5, we can observe that for *Yelp*, the optimal hyperedge number is 100, while for *Gowalla*, the performance reaches its peak when $K = 20$. When we further increase the hyperedge number, the recommendation performances drop to varying degrees, probably because too

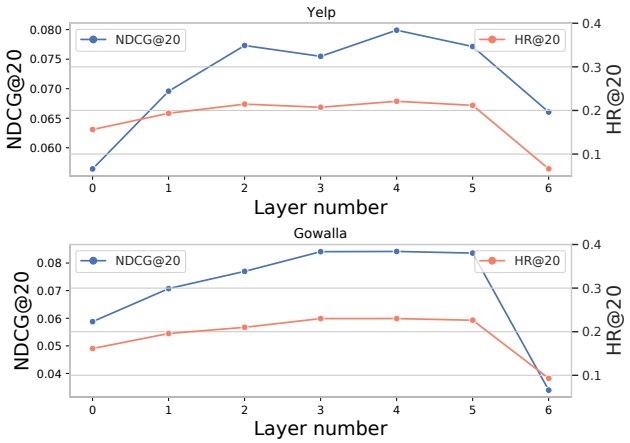

Figure 4: Influence of the depth of DHLCF

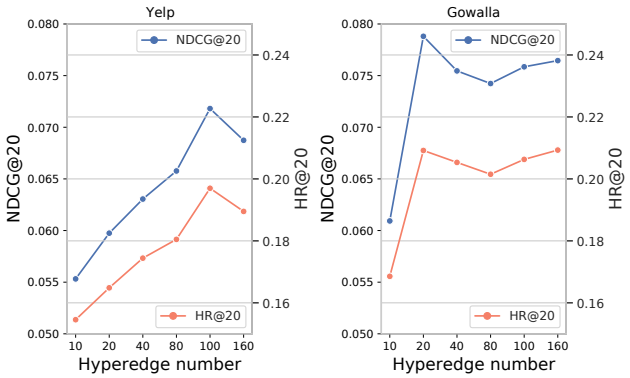

Figure 5: Influence of the learned hyperedge number

many meaningless hyperedges will propagate too much noise to the entity representations. Also, we found that in different recommendation scenarios, the optimal hyperedge numbers are varied, which should be influenced by many factors including the scale and diversity of users and items, e.g., compared with *Gowalla*, *Yelp* has more users and items, as well as more complex types of users and items, so it needs more hyperedges to depict those relations.

6 VISUALIZATION OF LEARNED HYPEREDGES

In different DHLCF layers, we exploit implicit high-order information in different hierarchical aspects by dynamically updating the hypergraph structure. In this section, we visually analyze the learned hyperedges in our model. We use a 3-layer DHLCF as suggested in Section 5.5 to train on the *Gowalla* dataset. Since the learning processes of the user and item hypergraph are symmetrical, we focus on the user hypergraph and randomly choose a user to visualize his assigned hyperedges.

Figure 6 shows the user nodes connected with the chosen user by the learned hyperedges in different DHLCF layers. As the connected user nodes become less, we can observe that our proposed hypergraph learner captures more and more sophisticated hierarchical

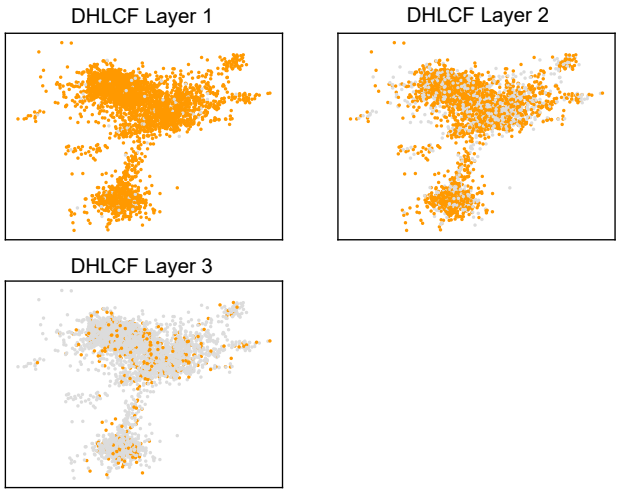

Figure 6: Visualization of learned hyperedges on *Gowalla*. Each of the sub-figure plots all the users and the orange nodes denote users connected by a hyperedge.

structure. The intuition behind this is also the same as our analysis in Section 5.5. At the first layer, we only exploit the first-order neighbors and thus the hyperedge learner can find massive users with similar collaborative signals to connect in the same hyperedge. In this sense, the learned hyperedges in the early layers help the chosen user encode the global features of the user set. When stacking more layers, the node representations contain collaborative signal and local topology from higher-order connectivities. So the hyperedges learned based on these representations tend to depict more complex local features of its connected users.

7 CONCLUSIONS

Hypergraph provides a natural way to model high-order relations among users and items, and has been introduced to improve recommendations. However, existing methods construct the hypergraph structures using heuristics based on existing topology, which restricts the model's capacity of capturing the implicit high-order relations. In this paper, we propose a novel dynamic hypergraph learning framework to produce a better hypergraph structure and make recommendations collectively. We overcome the optimization issue by proposing a fully differentiable hypergraph learner to update the hypergraph structure at each layer. To address the regularization issue brought by the auxiliary hypergraph learning task, we innovatively propose a novel hypergraph learning loss to guide the hypergraph learning. The extensive experiments conducted on three public datasets verify the effectiveness of DHLCF.

ACKNOWLEDGMENTS

This work was supported by Alibaba Group through Alibaba Research Intern Program.

REFERENCES

[1] Song Bai, Feihu Zhang, and Philip H. S. Torr. 2021. Hypergraph convolution and hypergraph attention. *Pattern Recognit.* 110 (2021), 107637. https://doi.org/10.1016/j.patcog.2020.107637

[2] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. 2020. Spectral Clustering with Graph Neural Networks for Graph Pooling. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 874–883. http://proceedings.mlr.press/v119/bianchi20a.html

[3] Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. 2010. Music recommendation by unified hypergraph: combining social media information and music content. In *Proceedings of the 18th International Conference on Multimedia 2010, Firenze, Italy, October 25-29, 2010*, Alberto Del Bimbo, Shih-Fu Chang, and Arnold W. M. Smeulders (Eds.). ACM, 391–400. https://doi.org/10.1145/1873951.1874005

[4] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz (Eds.). ACM, 515–524. https://doi.org/10.1145/3209978.3209991

[5] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph Neural Networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 3558–3565. https://doi.org/10.1609/aaai.v33i01.33013558

[6] Yue Gao, Meng Wang, Zheng-Jun Zha, Jialie Shen, Xuelong Li, and Xindong Wu. 2013. Visual-Textual Joint Relevance Learning for Tag-Based Social Image Search. *IEEE Trans. Image Process.* 22, 1 (2013), 363–376. https://doi.org/10.1109/TIP.2012.2202676

[7] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 639–648. https://doi.org/10.1145/3397271.3401063

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich (Eds.). ACM, 173–182. https://doi.org/10.1145/3038912.3052569

[9] Yuchi Huang, Qingshan Liu, and Dimitris N. Metaxas. 2009. Video object segmentation by hypergraph cut. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, 1738–1745. https://doi.org/10.1109/CVPR.2009.5206795

[10] TaeHyun Hwang, Ze Tian, Rui Kuang, and Jean-Pierre A. Kocher. 2008. Learning on Weighted Hypergraphs to Integrate Protein Interactions and Gene Expressions for Cancer Outcome Prediction. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*. IEEE Computer Society, 293–302. https://doi.org/10.1109/ICDM.2008.37

[11] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=rkE3y85ee

[12] Shuyi Ji, Yifan Feng, Rongrong Ji, Xibin Zhao, Wanwan Tang, and Yue Gao. 2020. Dual Channel Hypergraph Collaborative Filtering. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 2020–2029. https://doi.org/10.1145/3394486.3403253

[13] Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. 2019. Dynamic Hypergraph Neural Networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, Sarit Kraus (Ed.). ijcai.org, 2635–2641. https://doi.org/10.24963/ijcai.2019/366

[14] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 1748–1757. https://doi.org/10.1145/3394486.3403226

[15] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis (Eds.). ACM, 689–698. https://doi.org/10.1145/3178876.3186150

[16] Qingshan Liu, Yubao Sun, Cantian Wang, Tongliang Liu, and Dacheng Tao. 2017. Elastic Net Hypergraph Learning for Image Clustering and Semi-Supervised Classification. *IEEE Trans. Image Process.* 26, 1 (2017), 452–463. https://doi.org/10.1109/TIP.2016.2621671

[17] Siwei Liu, Iadh Ounis, Craig Macdonald, and Zaiqiao Meng. 2020. A Heterogeneous Graph Neural Model for Cold-start Recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 2029–2032. https://doi.org/10.1145/3397271.3401252

[18] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, Jeff A. Bilmes and Andrew Y. Ng (Eds.). AUAI Press, 452–461. https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=1630&proceeding_id=25

[19] Mechthild Stoer and Frank Wagner. 1994. A Simple Min Cut Algorithm. In *Algorithms - ESA '94, Second Annual European Symposium, Utrecht, The Netherlands, September 26-28, 1994, Proceedings (Lecture Notes in Computer Science, Vol. 855)*, Jan van Leeuwen (Ed.). Springer, 141–147. https://doi.org/10.1007/BFb0049404

[20] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. *CoRR* abs/1706.02263 (2017). arXiv:1706.02263 http://arxiv.org/abs/1706.02263

[21] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. 2020. Next-item Recommendation with Sequential Hypergraphs. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 1101–1110. https://doi.org/10.1145/3397271.3401133

[22] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer (Eds.). ACM, 165–174. https://doi.org/10.1145/3331184.3331267

[23] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A Neural Influence Diffusion Model for Social Recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer (Eds.). ACM, 235–244. https://doi.org/10.1145/3331184.3331214

[24] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* 32, 1 (2021), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

[25] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=ryGs6iA5Km

[26] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudré-Mauroux. 2019. Revisiting User Mobility and Social Relationships in LBSNs: A Hypergraph Embedding Approach. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (Eds.). ACM, 2147–2157. https://doi.org/10.1145/3308558.3313635

[27] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. 2021. Self-Supervised Multi-Channel Hypergraph Convolutional Network for Social Recommendation. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM / IW3C2, 413–424. https://doi.org/10.1145/3442381.3449844

[28] Zizhao Zhang, Haojie Lin, and Yue Gao. 2018. Dynamic Hypergraph Structure Learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, Jérôme Lang (Ed.). ijcai.org, 3162–3169. https://doi.org/10.24963/ijcai.2018/439

[29] Xiaoyao Zheng, Yonglong Luo, Liping Sun, Xintao Ding, and Ji Zhang. 2018. A novel social network hybrid recommender system based on hypergraph topologic structure. *World Wide Web* 21, 4 (2018), 985–1013. https://doi.org/10.1007/s11280-017-0494-5

[30] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2006. Learning with Hypergraphs: Clustering, Classification, and Embedding. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, Bernhard Schölkopf, John C. Platt, and Thomas Hofmann (Eds.). MIT Press, 1601–1608. https://proceedings.neurips.cc/paper/2006/hash/dff8e9c2ac33381546d96deea9922999-Abstract.html

[31] Lei Zhu, Jialie Shen, Hai Jin, Ran Zheng, and Liang Xie. 2015. Content-Based Visual Landmark Search via Multimodal Hypergraph Learning. *IEEE Trans. Cybern.* 45, 12 (2015), 2756–2769. https://doi.org/10.1109/TCYB.2014.2383389