Multi-Order Hypergraph Convolutional Neural Network for Dynamic Social Recommendation System

YU WANG 1 AND QILONG ZHAO 2

1 Key Laboratory of High Confidence Software Technologies of Ministry of Education, School of Computer Science, Institute of Software, Peking University, Beijing 100871, China
2 Tencent, Beijing 100193, China

Corresponding author: Yu Wang (wangyu18@pku.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFA0706401; and in part by the National Natural Science Foundation of China under Grant 62172014, Grant 62172015, Grant 61872166, and Grant 62002002.

ABSTRACT Recently, online social networks have enriched the users’ lives greatly and social recommendation systems make it easier for users to discover more information that they are interested in. The most advanced graph neural network based social recommendation methods start to utilize the higher-order social relations, e.g. the friends of friends, to reveal users’ preferences. However, existing high-order methods ignore the implicit social relations among users and the users’ interests changing dynamically over time. In this paper, we propose a Multi-Order Hypergraph Convolutional Neural Network (MOHCN) for dynamic social recommendation system to improve the recommendation task, which models the users’ dynamic interest evolution at the session level. To compensate for the lack of social information of some users, we combine the implicit social relations obtained from user-item interaction graph with the explicit social relations from user-user social graph through hypergraph modeling. Extensive experimental results on three real-world datasets demonstrate the effectiveness of our proposed MOHCN compared with the state-of-the-art methods.

INDEX TERMS Recommendation system, hypergraph convolutional neural network, multi-order social influence, dynamic interests.

I. INTRODUCTION

With the explosion of information, personalized recommendation is becoming more and more important. The key to successful recommendation lies in the “Collective Intelligence”, that is, the behavior of one user is similar to her friends. Generally, when a user considers buying some products, her friends may influence her choice. In addition, the popularity of online social networks facilitates the communication among users, which further amplifies the influence of social factors. By using trust relations among users as auxiliary information, social recommendation system can alleviate data sparsity and cold-start problem in collaborative filtering.

The data in the social recommendation has two kinds of relationship, including a user-user relationship and a user-item relationship. We can use graph to model these relationships. Users with friend relationships can be seen as node and be linked to build a user-user social graph, as shown in Figure 1. Using graph neural networks which imitate biological neural networks can well simulate the social relations between users and the interactions between users and items. The core idea of Graph Neural Networks (GNN) [1], [2] is how to iteratively aggregate feature information from local graph neighborhoods using neural networks to represent the node information, and then propagate the node information through the graph. Therefore, it is suitable to use GNN for social recommendation.

User-user social graph provides the explicit social relations clearly which can be used for cold-start problem for user who
has more friends data and less items data. But it is not suitable for the user with few friends which cannot build user-user relations in the graph. In addition to the explicit social relations, there are implicit social relations among users. Each user has user-item relations, then a hypergraph [3], [4] can be build as shown in Figure 2. Hypergraph is composed of hyperedge set and vertex set, and the hyperedge can connect any numbers of vertices. In this way, hypergraph can be used to encode high-order data correlations and efficiently extract higher-order relations on hypergraph via node-edge-node transformations. Hypergraph has hyperedges connecting more than two nodes which is suitable for the scene where a single user corresponds to multiple items. In Figure 2, the nodes are items, and the edges connected to the items are the users who purchased these items. Users who purchased the same items may not be directly related in social relations, but have a certain similarity in behavior, so they have implicit social relations. Using hypergraph to model implicit social relations is more beneficial to the cold start of new users with fewer friends. For users with more friends, it can also alleviate the recommendation accuracy problem caused by data sparse. HyperRec [5] uses hypergraph to model the short-term user preference for next-item recommendation. DHCN [6] captures the beyond-pairwise relations among items and the cross-session information through hypergraph modeling. However, they do not utilize user-item to build hypergraph, therefore they do not take full advantage of implicit social relations.

Existing social recommendation systems have already performed a good modeling of 1st-order social relations [7], [8], [9], [10]. In the highly informative social relations, the products purchased by the friends of friends will also affect the user’s preference when purchasing. Therefore, it is natural to extend them by adding higher-order edges to social networks to increase the influence of higher-order friends on social recommendations, especially for users with fewer 1st-order friends. We add higher-order edges to explicit social relation graph and implicit social relation hypergraph to improve recommendation effect.

Recent GNN-based social recommendation methods start to utilize the higher-order social relations to reveal users’ preferences. The model HOSR [11] first models high-order social relations with Graph Convolutional Network. Light-GCN [12] also utilizes the higher-order social relations and combines the embeddings learned at different propagation layers to obtain the final embedding for prediction. HAEGNN [13] is a higher-order attribute-enhancing framework for heterogeneous network representation learning. However, these models both suffer from the complex data correlation without exploiting implicit social relations, and do not consider the users’ session-based dynamic interests changes.

In this paper, we propose a Multi-order Hypergraph Convolutional Neural Network (MOHCN) for dynamic social recommendation. We present the multi-order social influence which is composed of these high-order social influence with different orders. Moreover, user-item interaction graph can be represented as a implicit social relation hypergraph, which obtains the implicit friends of the user with similar purchase preferences. User-user social graph provides the explicit social relations clearly. To get the user’s interests influenced by her multi-order friends better, combining the above two graphs is necessary. Furthermore, the users’ interests change over time dynamically. We utilize GRU [14] to model users’ dynamic interest evolution at the session level. The GRU model is simple and has few parameters, so it is more suitable for building larger networks and can reduce the risk of overfitting. Our approach first models the influence of multi-order friends in social networks using Hypergraph Convolutional Neural Networks. Extensive experimental results on real-world datasets show that our proposed MOHCN outperforms the state-of-the-art models and justify the effectiveness of MOHCN.

The main contributions of this paper are summarized as follows:

- We highlight multi-order social relations in social networks to balance the high-order social with different orders, and enhance the influence of high-order explicit and implicit friends through hypergraph modeling.
- We propose a Multi-order Hypergraph Convolutional Neural Network for dynamic social recommendation system, taking into account the users’ dynamic interest evolution at the session level and multi-order social relations.
• We conduct experiments on three real-world datasets. The experimental results show that our MOHCN achieves state-of-the-art results for social recommendation.

II. RELATED WORK
In this section, we briefly review some works that are relevant to our work, including social recommendation, graph neural network-based learning, and session-based recommendation.

A. SOCIAL RECOMMENDATION
With the rapid development of online social, using social relations for recommendation has attracted widespread attention in recent years [15]. A common assumption is that users’ preferences can be influenced by their friends [16]. SocialMF [17] constrains users’ feature vectors to be close to those of their friends to learn users’ latent features for users with little or no ratings. TrustMF [18] models the interaction between users, and maps users to the two low-dimensional spaces, i.e. the trustee space and the truster space. SoDimRec [19] first utilizes a community detection algorithm to divide users into several clusters. TranSIV [20] adopts a transfer learning strategy to coordinate social and rating information. IF-BPR [21] proposes to use social relations and purchase relations to build a heterogeneous information network. From the perspective of multi-task learning, TrustEV [9] unites collaborative filtering for recommendation and network embedding for user trust. ConsisRec [22] samples consistent neighbors by relating sampling probability with consistency scores between neighbors. Sun et al. present an attentive recurrent network based approach for temporal social recommendation [23]. DiffNet [24] propose a deep influence propagation model to stimulate how users are influenced by the recursive social diffusion process for social recommendation. DiffNet++ [25] is an improved algorithm of DiffNet that models the neural influence diffusion and interest diffusion in a unified framework.

Taking into account the influence of higher-order friends is more accurate than only considering 1st-order friends. The most relevant work to ours is HOSR [11], which first models high-order social relations with Graph Convolutional Network. However, it does not exploit hypergraph information and does not consider the users’ dynamic interest changes. Currently, there is no research bridging multi-order social influence and session-based dynamic interest changes, and we are the first to fill this gap.

B. GRAPH NEURAL NETWORK-BASED LEARNING
Graph Neural Network (GNN) [1], [2] is an extension of neural network to process graph structure data. Due to its good performance and interpretability, GNN has become a widely used graph analysis method. In the recommendation tasks, the social relations between users and the user-item interactions including users’ ratings of items are both typical graphical data.

GCMC [26] generates implicit features between users and items based on differentiable message passing on the bipartite interaction graph. PinSage [27] combines random walk and GraphSAGE, and proposes a random walk graph neural network to learn the embedding vector of nodes in the graph. K-GNNs [28] borrows the extension from 1-WL to k-WL, and extends GNN to k-GNN, where one node is regarded as one node in 1-GNN, two nodes are regarded as one node in 2-GNN, and so on. HAEGNN [13] is a higher-order attribute-enhancing framework for heterogeneous network representation learning, which simultaneously incorporates meta-paths and meta-graphs for semantics. GraphRec [8] integrates the user-item interaction information and the user’s direct friends social information to model the graph data in social recommendation. LightGCN [12] removes feature transformation and nonlinear activation from GCN, and proposes to construct the final embedding of a node as the weighted sum of its embeddings on all layers.

Hypergraph [3], [4], [29] provides a powerful way to model high-order relations. HyperRec [5] uses hypergraph to model the short-term user preference for next-item recommendation. DHCN [6] captures the beyond-pairwise relations among items and the cross-session information through hypergraph modeling. Although previous works have achieved excellent results, using hypergraph structure to model dynamic multi-order social relations and implicit social relations has not been considered.

C. SESSION-BASED RECOMMENDATION
Since the user’s entire historical sequence may be very long, the user’s behavior sequence can be divided into different sessions, and recommendations can be provided dynamically at the session level.

SASRec [30] models the entire user sequence, and adaptively considers consumed items for prediction. ACKRec [31] is an end-to-end graph neural network based approach that incorporates rich heterogeneous context side information into knowledge concept recommendation in MOOC. STOSA [32] is a stochastic self-attention sequential model for modeling dynamic uncertainty and capturing collaborative transitivity. However, the above sequence recommendation models are based on long-term user profiles which are less flexible and less timely than session-based recommendation, and don’t consider the influence of social relations on users.

GRU4Rec [33] uses a recurrent neural network method to process session-based recommendations. Later, STAMP [34] utilizes a simple MLP network and attention network to effectively capture the user’s overall interests and current interests. SR-GNN [35] models the session sequence as graph structure data, and uses the attention mechanism to combine overall preferences with current preferences. And DGRec [7] combines the users’ dynamic interests with their social influence based on the graph attention mechanism, but it does not consider the multi-order social relations which can affect users’ interests greatly. CD-GCN [36] combines LSTM networks and GCNs to learn long short-term dependencies together with graph structure. However, the computational cost of LSTM is relatively high than GRU and this model is not suitable for high-order social recommendation.
Compared to previous work, we propose a Multi-order Hypergraph Convolutional Neural Network for dynamic social recommendation system, which focuses on multi-order social relations using graph structures, combined with modeling the users’ dynamic interests evolution at the session level.

III. THE PROPOSED METHOD

In this section, we introduce the proposed MOHCN, a Multi-order Hypergraph Convolutional Neural Network for dynamic social recommendation system. The model includes four parts: dynamic individual interests modeling, multi-order social influence modeling, item feature modeling, and rating prediction. Figure 3 illustrates the overview of our framework. Before introducing the details of MOHCN, we introduce the definitions and notations used in this paper.

A. NOTATIONS AND DEFINITIONS

In our model, let $U = \{u_1, u_2, \ldots, u_N\}$ denote the set of users and $V = \{v_1, v_2, \ldots, v_M\}$ denote the set of all unique items, where $N$ is the number of users and $M$ is the number of items. $K(i)$ is the session sequence of items which user $u_i$ have interacted with, and $H(j)$ is the set of users interacting with item $v_j$. Each session sequence $s$ can be represented as a vector $s = [v_{s,1}, v_{s,2}, \ldots, v_{s,K(i)}]$ ordered by timestamps, where $v_{s,j} \in V$. The user-item graph is represented as $\mathbf{R} \in \mathbb{R}^{N \times M}$, which is a user-item rating matrix. If user $u_i$ gives a rating to $v_j$, then $r_{ij}$ is the rating score, otherwise we represent the unknown $r_{ij}$ as 0. The rating score $r_{ij}$ can be regarded as the opinion of user $u_i$ on item $v_j$.

The user-user social graph is represented as $\mathbf{S} \in \mathbb{R}^{N \times N}$. If the user $u_i$ and $u_j$ are directly connected, then $S_{ij} = 1$, otherwise it is 0. The degree matrix is diagonal matrix denoted as $\mathbf{D_S}$, where $D_{S_{ij}} = \sum_{j=1}^{N} S_{ij}$. We use the embedding vector $\mathbf{p}_i \in \mathbb{R}^d$ to represent the user $u_i$ and the embedding vector $\mathbf{q}_j \in \mathbb{R}^d$ to represent the item $v_j$. And we use the embedding vector $\mathbf{e}_i \in \mathbb{R}^d$ to represent the opinion of the user $u_i$ on the item $v_j$.

Definition 1 (Hypergraph): Let $G = (V, E)$ denote a hypergraph, where $V$ is a set containing $M$ vertices and $E$ is a set containing $N$ hyperedges, and each hyperedge $e \in E$ contains two or more vertices. The hypergraph is assigned a weighted matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$, which is a diagonal matrix with $W_{ee}$ representing the weight of the hyperedge $e$. The hypergraph can be represented by an incidence matrix $\mathbf{H} \in \mathbb{R}^{M \times N}$, which is the transpose matrix of the user-item rating matrix, where $H_{ie} = 1$ if the hyperedge $e \in E$ contains a vertex $v_i \in V$, otherwise 0. The vertex and hyperedge degree matrices are diagonal matrices denoted by $\mathbf{D}$ and $\mathbf{B}$ respectively, where $D_{ii} = \sum_{j=1}^{N} W_{ee}H_{ie}$ and $B_{ee} = \sum_{j=1}^{M} H_{ee}$. In this paper, $W_{ee}$ is uniformly assigned as 1 and hence $\mathbf{W}$ is an identity matrix.

Definition 2 (Multi-Order Social Influence): Let $N(i)$ be the set of friends of user $u_i$, where $N_1(i)$ is the set of 1st-order friends, that is, the direct friends of users, $N_2(i)$ is the set of 2nd-order friends, that is, the friends of friends, and so on, as shown in Figure 4. Notably, $L$ is the order size of the multi-order friends of user $u_i$ we consider. The multi-order social influence is composed of high-order social influence with different orders, defined as $\sum_{l=1}^{L} \alpha_l f(N_l(i))$, where $f(N_l(i))$ is the aggregator of the influence of $l$th-order friends and $\alpha_l$ is the weight of $l$th-order friends. For example, the $L$-order social influence aggregates all social influence whose order is less than or equal to $L$ with different weights.

B. DYNAMIC INDIVIDUAL INTERESTS MODELING

The ratings for items given by users can capture users’ preferences for items, so we jointly utilize the interaction information and opinion information in the user-item graph to learn the individual interests $\mathbf{h}_i^f \in \mathbb{R}^d$ of user $i$.

To obtain the user’s individual interests accurately, we use $\mathbf{x}_{ij}$ to represent the opinion-interaction representation vector between user $u_i$ and item $v_j$. The opinion-interaction vector $\mathbf{x}_{ij}$ is modeled by combining interaction embedding $\mathbf{q}_j$ and opinion embedding $\mathbf{e}_i$ via a multi-layer perceptron (MLP), i.e. $\mathbf{x}_{ij} = f_{\alpha}(\mathbf{q}_j \oplus \mathbf{e}_i)$, where $\oplus$ represents the concatenation operation between two vectors.

Considering that the users’ interests evolution over time can be seen as a session, Gated Recurrent Unit (GRU) is suitable for extracting users’ dynamic interests at the session level. We denote $\mathbf{x}_{ij}$ processed by GRU as $\hat{\mathbf{x}}_{ij} = \sigma(\mathbf{W}_o \cdot \mathbf{h}_i^f)$.

The detailed is as follows:

$$r' = \sigma(\mathbf{W}_r \cdot [\mathbf{h}_i^{l-1} \oplus \hat{\mathbf{x}}_{ij}]),$$

$$\hat{z}' = \sigma(\mathbf{W}_z \cdot [\mathbf{h}_i^{l-1} \oplus \hat{\mathbf{x}}_{ij}]),$$

$$\hat{\mathbf{h}}' = \tanh(\mathbf{W}_h \cdot ([r' \cdot \mathbf{h}_i^{l-1}] \oplus \hat{\mathbf{x}}_{ij})), $$

$$\mathbf{h}_i^l = (1 - \hat{z}') \cdot \mathbf{h}_i^{l-1} + \hat{z}' \cdot \hat{\mathbf{h}}'. $$

To assign individualized weights for each user-item pair $(u_i, v_j)$, inspired by Graph Attention Network [37], we associate the opinion-interaction vector $\mathbf{x}_{ij}$ with the user embedding $\mathbf{p}_i$ of user $u_i$, and adjust the attention weight $\alpha_{ij}$ to adapt to the target user $u_i$.

$$\alpha_{ij} = \mathbf{w}_{a2} \cdot \text{LeakyReLU}(\mathbf{W}_{a1} \cdot [\hat{\mathbf{x}}_{ij} \oplus \mathbf{p}_i] + \mathbf{b}_{a1}) + \mathbf{b}_{a2},$$

$$\alpha_{ij} = \text{softmax}(\alpha_{ij}),$$

where $j \in K(i)$ denotes the items which user $u_i$ have interacted with. The user’s individual interests $\mathbf{h}_i^l$ can be obtained as follows:

$$\mathbf{h}_i^l = \sigma(\mathbf{W}_{h_l} \cdot \sum_{j \in K(i)} \alpha_{ij} \hat{\mathbf{x}}_{ij} + \mathbf{b}_{h_l}).$$

C. MULTI-ORDER SOCIAL INFLUENCE MODELING

A user’s interests are not only influenced by her directly connected friends, but also by her higher-order friends. We use $\mathbf{h}_i^f \in \mathbb{R}^d$ to represent the interests of $u_i$ influenced by multi-order friends, where the interests influenced by $l$th-order friends are marked as $\mathbf{h}_i^{(l)} \in \mathbb{R}^d$. In the user-item relation graph, each user has interacted with many items and each
item has interacted with many users. User-item graph can be represented as a implicit social relation hypergraph, which obtains the implicit friends of the user with similar purchase preferences. Therefore, it is reasonable to use hypergraph convolutional network (HGCN) to obtain the user’s interest similar to the implicit friends. User-user social graph provides the explicit social relations clearly, so graph convolutional network (GCN) can learn the user’s interest influenced by the explicit friends. To get the interests of $u_i$ influenced by multi-order friends, combining these two graphs is necessary.

In our user-item hypergraph modeling, vertices and hyperedges represent items and users respectively, and users’ interests are aggregated by vertices connected to the hyperedges. Referring to the spectral hypergraph convolution [3], we define our hypergraph convolution as:

$$x_j^{(l)} = \sum_{i=1}^{M} \sum_{\epsilon=1}^{N} H_{ji} H_{\epsilon j} x_j^{(l-1)}.$$  \hspace{1cm} (8)

Following the suggestions in [38], we remove the learnable matrix for linear transformation and the nonlinear activation function. Hypergraph convolution can be regarded as a two-stage refinement of the feature transformation of “node-hyperedge-node” on the hypergraph structure. $H_{ji}$ means the information aggregation from nodes to hyperedges, and premultiplying $H_{\epsilon j}$ means the information aggregation from hyperedges to nodes. We formulate this convolution process into a matrix form as:

$$X^{(l)} = D^{-1} HB^{-1}H^T X^{(l-1)},$$  \hspace{1cm} (9)
To prevent numerical instabilities caused by stacking multiple convolutional layers, we add in symmetric normalization which can improve numerical stability:

\[
X^{(l)} = D^{-\frac{1}{2}}HB^{-1}H'D^{-\frac{1}{2}}X^{(l-1)},
\]

(10)

where \(X^{(0)} \in \mathbb{R}^{M \times d}\) is the base item embeddings. Users’ interests are aggregated by vertices connected to the hyper-edges, and then we can get the interests of \(u_i\) influenced by \(l\)-th order implicit friends through the attention mechanism:

\[
\beta_{il} = w_{p2}^T \cdot \text{LeakyReLU}(W_{p1} \cdot [x_i^{(l)} \oplus p_i] + b_{p1}) + b_{p2},
\]

(11)

\[
\beta_{il} = \text{softmax}(\beta_{il}^{(l)} = \frac{\exp(\beta_{il}^{(l)})}{\sum_{o \in N(i)} \exp(\beta_{io}^{(l)})}),
\]

(12)

\[
h^{(l)}_{\text{imp}} = \sigma(W_{h3} \cdot \sum_{o \in N(i)} \beta_{io}h_o^{(l)} + b_{h3}),
\]

(13)

where \(x_i^{(l)} \in X^{(l)}\) is the item embedding that the user \(i\) has interacted with and \(p_i\) is the user embedding of user \(i\). By concatenating \(x_i^{(l)}\) and \(p_i\) in Equation (11), the output can be seen as the implicit social-aware attention representation of the interaction between \(x_i^{(l)}\) and \(p_i\). Through stacking more hypergraph convolutional layers, we can assemble the interests of higher-order implicit friends. By stacking \(l\) HGNC layers in the implicit relation hypergraph, we can obtain the user’s interests influenced by her \(l\)-th order implicit friends \(h^{(l)}_{\text{imp}} \in \mathbb{R}^d\).

Similarly, through stacking \(l\) GCN layers in the explicit social graph where vertices represent users, the user \(i\) can aggregate her interests from her \(l\)-th order explicit friends \(h^{(l)}_{\text{exp}} \in \mathbb{R}^d\). The simple graph convolution is defined as:

\[
h^{(l)}_{\text{exp}} = (D_{S}^{-\frac{1}{2}}S_{D}^{-\frac{1}{2}})h^{(l-1)}_{\text{exp}},
\]

(14)

where \(D_{S}\) and \(S\) are degree matrix and adjacency matrix of explicit social graph, and \(h^{(0)}_{\text{exp}} \in \mathbb{R}^{N \times d}\) is the base user embeddings and \(h^{(l)}_{\text{exp}} \in \mathbb{R}^{N \times d}\). Then we use MLP to combine the \(l\)-th order implicit social influence and the \(l\)-th order explicit social influence on user \(i\), i.e. \(h^{(l)} = f_{\gamma}(h^{(l)}_{\text{imp}} \oplus h^{(l)}_{\text{exp}})\), to represent the user’s interests influenced by \(l\)-th order friends.

In order to capture the different influence of friends with different orders and solve the problem of over-changing importance and over-smoothing, we employ the attention mechanism to adaptively aggregate the influence of high-order friends with different orders on users’ interests. The interests of \(u_i\) influenced by multi-order friends \(h^S\) can be obtained as follows:

\[
\gamma_{il}^u = w_{y2}^T \cdot \text{LeakyReLU}(W_{y1} \cdot [h^{(l)} \oplus p_i] + b_{y1}) + b_{y2},
\]

(15)

\[
\gamma_{il} = \text{softmax}(\gamma_{il}^u = \frac{\exp(\gamma_{il}^u)}{\sum_{l \in L} \exp(\gamma_{il}^u)}),
\]

(16)

\[
h^S = \sigma(W_{h5} \cdot \sum_{l \in L} \gamma_{il}h^{(l)} + b_{h5}),
\]

(17)

where \(l \in L\) denotes the order of multi-order friends of user \(u_i\), \(h^{(l)}\) represents the interests of \(u_i\) influenced by \(l\)-th order friends, and the attention weight \(\gamma_{il}^u\) represents the degree of influence of \(l\)-th order friends on the user’s interests.

In order to learn the user’s interests better, we utilize MLP to combine the user’s individual interests \(h^i\) and the interests influenced by multi-order friends \(h^S\) as the final user’s interests latent factor \(h_i \in \mathbb{R}^d\):

\[
h_i = f_\delta(h^i \oplus h^S).
\]

(18)

\[\text{D. ITEM FEATURE MODELING}\]

Since the item embedding obtained through HGNC can be updated as the user’s implicit friends change, the item embedding learned from the user-item relation graph through the L-layer HGNC can be regard as the dynamic item embedding. Combined with the dynamic item embedding \(x_{j}^{(L)}\) obtained in Section III.C, in this section, we get the static item embedding \(\tilde{y}_j\) by the interactions in the user-item graph, to obtain the item \(v_j\)’s final feature latent factor \(y_j \in \mathbb{R}^d\).

Even for the same item, different users may have different opinions about it. These opinions from different users can capture the feature of the same item from a different perspective. To obtain the static item embedding, we utilize the opinion-interaction vector \(g_{ji}\) to represent the interaction and opinion between item \(v_j\) and user \(u_i\), which is obtained from the user embedding \(p_i\) and the opinion embedding \(e_j\) through an MLP function \(f_\gamma\), i.e. \(g_{ji} = f_\gamma(p_i \oplus e_j)\), combining the interaction information and the opinion information.

What’s more, in order to learn the static item embedding \(\tilde{y}_j\) accurately, we introduce an attention mechanism to distinguish the weight of different opinion-interaction representation vector \(g_{ji}\) about the target item \(v_j\):

\[
\mu_{ji}^* = w_{\mu2}^T \cdot \text{LeakyReLU}(W_{\mu1} \cdot [g_{ji} \oplus q_j] + b_{\mu1}) + b_{\mu2},
\]

(19)

\[
\mu_{ji} = \text{softmax}(\mu_{ji}^* = \frac{\exp(\mu_{ji}^*)}{\sum_{i \in H(j)} \exp(\mu_{ji}^*)}),
\]

(20)

\[
\tilde{y}_j = \sigma(W_{\gamma} \cdot \sum_{i \in H(j)} \mu_{ji}g_{ji} + b_{\gamma}),
\]

(21)

where \(i \in H(j)\) denotes the users interacting with item \(v_j\), \(q_j\) is the item embedding of the target item \(v_j\), and the attention weight \(\mu_{ji}\) can be seen as the contribution of the opinion-interaction vector \(g_{ji}\) to the static item embedding of item \(v_j\). Then we use MLP to combine the dynamic item embedding \(x_{j}^{(L)}\) and the static item embedding \(\tilde{y}_j\) as the final item feature latent factor \(y_j \in \mathbb{R}^d\):

\[
y_j = f_\epsilon(x_{j}^{(L)} \oplus \tilde{y}_j).
\]

(22)

\[\text{E. RATING PREDICTION AND MODEL TRAINING}\]

In this paper, our MOHCN is used for the recommendation task of rating prediction. We connect the user’s interests latent factor \(h_i\) and item feature latent factor \(y_j\), and then feed them into an MLP function \(f_\zeta\) for rating prediction. The final
predicted rating from user $u_i$ to item $v_j$ is $r'_{ij}$:

$$r'_{ij} = f_t(h_i + y_j).$$  (23)

And the loss function for training is calculated as:

$$L = \frac{1}{2|T|} \sum_{(i,j) \in T} (r'_{ij} - r_{ij})^2 + \lambda \|\theta\|^2,$$  (24)

where $T$ is the set of ratings given by users to items, $|T|$ is the number of known ratings, $r_{ij}$ is the ground truth rating given by user $u_i$ to item $v_j$, and $\|\theta\|^2$ denotes the L2 regularization and $\lambda$ is used to control its weight.

IV. EXPERIMENTS

A. EXPERIMENTAL SETTINGS

1) DATASETS

Ciao, Epinions\textsuperscript{1} [39], [40], [41], [42] and Yelp\textsuperscript{2} [43] are three representative datasets for studying social recommendation problem. We conduct social recommendation experiments on Ciao, Epinions and Yelp to verify the effectiveness of the proposed MOHCN. The three datasets we use are taken from three popular product review social websites Ciao,\textsuperscript{3} Epinions\textsuperscript{4} and Yelp.\textsuperscript{5} These datasets are time-stamped and contain users’ ratings for items and their friends, that is, they contain rating information and social information. For fair comparison, we filter out all sessions with less than 5 items and items appearing less than 5 times in these datasets. We reserve the sessions of the last $d$ days (timestamps) in these datasets where we choose $d = 25$. Table 1 lists the descriptive statistics of all datasets.

2) BASELINES

We compare the proposed MOHCN with the following types of benchmark models: (A) Classic recommendation methods without social information; (B) Classic social recommendation methods taking into account social influence; (C) Session-based or sequential recommendation methods; (D) Social recommendation methods based on deep learning.

- **PMF** [44] (A): only using user-item rating matrix and Gaussian distribution to model latent factors of users and items.
- **SocialMF** [17] (B): considering social trust information and propagating trust information to the matrix factorization model.
- **SoRec** [45] (B): performing co-factorization on user-item rating matrix and user-user social relation matrix.
- **SocialLMF** [18] (B): using the user’s social relation information to design a social regularization term to constrain the matrix factorization framework.
- **TrustMF** [18] (B): mapping users to the two low-dimensional spaces of the trustee space and the truster space by factorizing the social trust network.
- **NeuMF** [47] (A): the lastest matrix factorization model with neural network architecture.
- **SASRec** [30] (C): modeling the entire user sequence, and adaptively considering consumed items for prediction.
- **DeepSoR** [10] (D): using a deep neural network to learn the representation of each user from social relations and integrates it into the PMF.
- **SR-GNN** [35] (C): modeling session sequences as graph-structured data.
- **DGRec** [7] (D): combining the users’ dynamic interests with their social influence based on the graph attention mechanism.
- **GraphRec** [8] (D): integrating user-item interaction information and user’s 1-order social information to model graph data in social recommendation.
- **TrustEV** [9] (D): taking the view of multi-task learning to unite collaborative filtering for recommendation and network embedding for user trust.
- **CD-GCN** [36] (C): combining LSTM networks and GCNs to learn long short-term dependencies together with graph structure.
- **HOSR** [11] (D): modeling high-order social relations with Graph Convolutional Network.
- **CosisRec** [22] (D): sampling consistent neighbors by relating sampling probability with consistency scores between neighbors.

3) EVALUATION METRICS

We use Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) which are widely used as measures of rating prediction, to evaluate the performance of the above models. MAE is the average value of absolute error between the predicted value and the actual value, and RMSE is the square root of average value of the squared difference between the predicted value and the actual value.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |\hat{y}_i - y_i|$$  (25)

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2}$$  (26)

The smaller values of MAE and RMSE indicate the higher prediction accuracy and the better performance for recommendation. We randomly divide the whole dataset into three
TABLE 2. Comparison of different models.

| Model Class | Models       | Ciao MAE | Ciao RMSE | Epinions MAE | Epinions RMSE | Yelp MAE | Yelp RMSE |
|-------------|--------------|----------|-----------|--------------|---------------|----------|-----------|
| Classic     | PMF (A)      | 0.9021   | 1.1234    | 0.9952       | 1.2128        | 1.1740   | 1.5323    |
|             | NeuMF (A)    | 0.8062   | 1.0617    | 0.9072       | 1.1476        | 1.1368   | 1.3927    |
| Social      | SoRec (B)    | 0.8410   | 1.0652    | 0.8961       | 1.1437        | 0.9571   | 1.2485    |
|             | SoReg (B)    | 0.8611   | 1.0848    | 0.9119       | 1.1703        | 0.9487   | 1.2512    |
|             | SocialMF (B) | 0.8270   | 1.0501    | 0.8837       | 1.1328        | 0.9365   | 1.2061    |
|             | TrustMF (B)  | 0.7690   | 1.0479    | 0.8410       | 1.1395        | 0.9270   | 1.1899    |
| Dynamic     | SASRec (C)   | 0.7791   | 1.0393    | 0.8414       | 1.1126        | 0.9177   | 1.1602    |
|             | SR-GNN (C)   | 0.7763   | 1.0375    | 0.8391       | 1.1027        | 0.9124   | 1.1589    |
|             | CD-GCN (C)   | 0.7706   | 1.0281    | 0.8310       | 1.0904        | 0.9092   | 1.1534    |
| Social+Deep Learning | DeepSoR (D) | 0.7739   | 1.0316    | 0.8383       | 1.0972        | 0.9033   | 1.1452    |
|             | DGRRec (D)   | 0.7531   | 1.0127    | 0.8263       | 1.0846        | 0.8975   | 1.1301    |
|             | TrustEV (D)  | 0.9388   | 1.2491    | 0.8151       | 1.0755        | 0.8755   | 1.1316    |
|             | GraphRec (D) | 0.7387   | 0.9794    | 0.8168       | 1.0631        | 0.8621   | 1.1288    |
|             | HOSR (D)     | 0.7372†  | 0.9740    | 0.8177       | 1.0643        | 0.8412†  | 1.1127†   |
|             | ConsisRec (D)| 0.7394   | 0.9722†   | 0.8046†      | 1.0495†       | 0.8437† | 1.1204†   |

Our Proposal | MOHCN | 0.6988 †±0.0031 | 0.9362 †±0.0044 | 0.7908 †±0.0060 | 1.0327 †±0.0065 | 0.8272 †±0.0042 | 1.0920 †±0.0057 |

Our Proposal | Impr. | 5.21% | 3.70% | 1.75% | 1.60% | 1.77% | 1.86% |

parts: 80% for training, 10% for validation and 10% for testing.

4) HYPER-PARAMETER SETTINGS

We implement our proposed model using Pytorch. In order to optimize the objective function, we use Adam [48] as the optimizer, which does exponentially weighted average and regularization of the gradient term, and uses momentum and adaptive learning rate to accelerate the convergence, with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e^{-4}$. When optimizing deep neural network models, overfitting is an eternal problem. In order to alleviate this problem, we adopt a dropout strategy in the model [49], randomly dropping some neurons during the training process, and the dropout rate is set to 0.5. The embedding size is set to 64, and the initial learning rate is set to 0.001. And we use small batch for model training, each batch size is set to 64.

B. PERFORMANCE EVALUATION

We conduct experiments on the three datasets Ciao, Epinions and Yelp to compare our MOHCN with state-of-the-art methods, and the MOHCN here considers the 2-order social as default. The results of the above different models are summarized in Table 2. The best performance is highlighted in bold while the suboptimal performance is marked by ‘†’. It shows that only considering the user-item rating matrix, such as PMF and NeuMF, is not enough to capture user preferences for items. The user’s social relation has a great influence on the user’s interests, and the dynamic interest is also important for discovering the user’s interest. What’s more, digging deeper into social networks like these algorithms (D) can get better recommendation effect.

In all models, our proposed MOHCN achieved the best recommendation accuracy on all datasets. HOSR and ConsisRec achieve the best baseline results on Ciao, Yelp and Epinions respectively. On Ciao, MOHCN outperforms the best baseline over 3.70% for MAE and 5.21% for RMSE. On Epinions, MOHCN outperforms the best baseline over 1.75% for MAE and 1.60% for RMSE. On Yelp, MOHCN outperforms the best baseline over 1.77% for MAE and 1.86% for RMSE. The results show that our proposed MOHCN which takes into account the user’s dynamic interest evolution and multi-order social relations can make better use of social information, and the recommendation effect is further improved.

C. EXPERIMENT ANALYSIS

1) ABLATION STUDY

In order to study the effects of dynamic interest evolution and multi-order social influence with different $k$-order, we compared the model variants of MOHCN: MOHCN-$x$ which is MOHCN considering $x$-order social influence, MOHCN-E which is MOHCN-2 without considering explicit social relations, and MOHCN-S which is MOHCN-2 without considering dynamic interest evolution. The experimental results on Ciao, Epinions and Yelp are shown in Figure 5. It can be seen that the recommendation effect of MOHCN-2 is significantly better than that of MOHCN-1, and the recommendation effect of MOHCN-3 is better than that of MOHCN-2. Moreover, the recommendation effect of MOHCN-4 is only slightly better than that of MOHCN-3, while on Epinions, MOHCN-4 is slightly worse than MOHCN-3 for RMSE. It shows that multi-order social influence is effective in improving recommendation accuracy.

However, as the order size increases, the time required for recommendation will increase, but the recommendation accuracy will not be greatly improved. Therefore, from the perspective of real-time recommendation, there is no need to set the order size of multi-order social relations too large. What’s more, the number of 1st-order friends of users is small and the number of higher-order friends is large on Ciao and Yelp, and the social relations are dense, so the influence of higher-order friends on users’ interests will become significant. The number of 1st-order friends is large on Epinions, so the influence of higher-order friends on users’ interests is relatively weak.

87646 VOLUME 10, 2022
Furthermore, the best performance of most other GCN-based models is achieved when the order size is two, instead of larger order size. The reason might be that, as the order size increases, those models aggregate too much high-order social information and ignore the important low-order social information, leading to over-smoothing problem, further reduces the model performance. Our multi-order social influence and attention mechanism can alleviate this problem.

As can be seen from Figure 5, the recommendation effect of the regular MOHCN-2 is better than MOHCN-E which is MOHCN-2 without considering explicit social relations.
on all datasets. It indicates that explicit social relations are important for recommendation effect, rather than only considering the user-item relations. As shown in Figure 5(c), the recommendation effect of the regular MOHCN-2 is much better than MOHCN-S which is MOHCN-2 without considering users’ dynamic interest evolution on Epinions. The recommendation effect of MOHCN-2 is also slightly better than MOHCN-S on Ciao and Yelp. It indicates that considering users’ dynamic interest evolution can improve recommendation accuracy, and session-based recommendation can model it satisfactorily. Moreover, the effect of users’ dynamic interest evolution is more obvious on Epinions, because the average clicked history sequence of users on Epinions is longer, which can better capture the users’ dynamic interest evolution, so as to achieve better recommendation results.

2) PERFORMANCE COMPARISON W.R.T EMBEDDING SIZE

We discuss the impact of embedding size on model performance, and the results are shown in Figure 6(a) and Figure 6(b). We take the embedding size as [8, 16, 32, 64, 128, 256]. On Ciao, as the embedding size increases from 8 to 256, the performance of the model will first get better and then worse, and the performance is best when the embedding size is 64. Similarly, on Yelp, as the embedding size increases, the performance of the model will also first get better and then worse, and the performance is also best when the embedding size is 64. If the embedding size is too large, the embedding representation will become sparse, resulting in performance degradation. On Epinions, as the embedding size increases, the performance of the model sharply becomes better, and then slowly gets worse, which works best when the embedding size is 64. Although the effect is similar at 64 and 128, we choose 64 as our embedding size, because a larger embedding size will increase training time and space.

3) PERFORMANCE COMPARISON W.R.T ATTENTION MECHANISMS

We further evaluate the attention mechanisms of the proposed MOHCN as shown in Figure 6(d) and Figure 6(e). There are four different attention mechanisms in MOHCN which can be seen from Section III, including user individual attention \(\alpha\), 1st-order social attention \(\beta\), multi-order social attention \(\gamma\), and item attention \(\mu\). We compare MOHCN with its four variants: MOHCN-\(\alpha\), MOHCN-\(\beta\), MOHCN-\(\gamma\), and MOHCN-\(\mu\). MOHCN-\(\alpha\) represents removing the attention mechanism \(\alpha\) from MOHCN, and the other three variants have the similar meanings. These four attention mechanisms have different effects on the recommendation accuracy. MOHCN-\(\alpha\) and MOHCN-\(\mu\) obtain worse performance than MOHCN, which demonstrates the benefits of user individual attention \(\alpha\) and item attention \(\mu\). When eliminating the attention \(\beta\) and \(\gamma\) from MOHCN, the performance will get worse. It indicates that social relations are important for recommendation as users’ preferences are influenced by their friends. For Ciao, the attention \(\gamma\) has a greater influence than \(\beta\), because the number of 1st-order friends of users is relatively small and the number of higher-order friends is large on Ciao, and the social relations are dense. For Epinions, the attention \(\beta\) is more important than \(\gamma\) since the number of 1st-order friends is large on Epinions. For Yelp, the attention \(\gamma\) has a slightly greater influence than \(\beta\), because the number of 1st-order friends of users is relatively small and the number of higher-order friends is relatively large on Yelp.

V. CONCLUSION

In this paper, we propose a Multi-order Hypergraph Convolutional Neural Network for dynamic social recommendation system, which uses hypergraph to model user-item interaction graphs and user-user social graphs together. To compensate for the lack of social information of some users, we combine the implicit social relations obtained from user-item interaction graph with the explicit social relations from user-user social graph. We are the first to fill the gap by modeling session-based data as a multi-order hypergraph structure dynamically, which can better encode high-order data correlation. Moreover, we innovatively model the influence of multi-order friends on users in social networks to solve the problem of data sparsity and over-smoothing problem of other high-order social models. To further improve recommendation accuracy, Gated Recurrent Units are utilized to model the users’ session-based dynamic individual interests. The experimental results on two real-world datasets prove the effectiveness of our proposed MOHCN compared to the state-of-the-art.

REFERENCES

[1] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” IEEE Trans. Neural Netw., vol. 20, no. 1, pp. 61–80, Jan. 2009.
[2] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive survey on graph neural networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 1, pp. 4–24, Jan. 2021.
[3] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, “Hypergraph neural networks,” in Proc. AAAI Conf. Artif. Intell., vol. 33, 2019, pp. 3558–3565.
[4] S. Bandyopadhyay, K. Das, and M. N. Murty, “Line hypergraph convolution network: Applying graph convolution for hypergraphs,” 2020, arXiv:2002.03392.
[5] I. Wang, K. Ding, L. Hong, H. Liu, and J. Caverlee, “Next-item recommendation with sequential hypergraphs,” in Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2020, pp. 1101–1110.
[6] J. Yu, H. Yin, J. Li, Q. Wang, N. Q. V. Hung, and X. Zhang, “Self-supervised multi-channel hypergraph convolutional network for social recommendation,” in Proc. Web Conf., vol. 35, Apr. 2021, pp. 4503–4511.
[7] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, “Session-based social recommendation via dynamic graph attention networks,” in Proc. 12th ACM Int. Conf. Web Search Data Mining, Jan. 2019, pp. 555–563.
[8] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, “Graph neural networks for social recommendation,” in Proc. World Wide Web Conf. (WWW), 2019, pp. 417–426.
[9] Q. Wu, L. Jiang, X. Gao, X. Yang, and G. Chen, “Feature evolution based multi-task learning for collaborative filtering with social trust,” in Proc. 28th Int. Joint Conf. Artif. Intell., Aug. 2019, pp. 3877–3883.
[10] W. Fan, Q. Li, and M. Cheng, “Deep modeling of social relations for recommendation,” in Proc. 32nd AAAI Conf. Artif. Intell. (AAAI), 2018, pp. 8075–8076.
[11] Y. Liu, L. Chen, X. He, J. Peng, Z. Zheng, and J. Tang, “Modelling higher-order social relations for item recommendation,” IEEE Trans. Knowl. Data Eng., vol. 34, no. 9, pp. 4385–4397, Sep. 2022.
[12] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, “Light- 
GCN: Simplifying and powering graph convolution for recom-
mandation,” in Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., 
Jul. 2020, pp. 639–648.

[13] J. Li, H. Peng, Y. Cao, Y. Dou, H. Zhang, P. Yu, and L. He, “High-
order attention-enhancing heterogeneous graph neural networks,” 
IEEE Trans. Knowl. Data Eng., early access, Apr. 23, 2021, doi:
10.1109/TKDE.2021.3074654.

[14] R. Dey and F. M. Salem, “Gate-variants of gated recurrent unit (GRU)
neural networks,” in Proc. IEEE 60th Int. Midwest Symp. Circuits Syst. 
(MWSCAS), Aug. 2017, pp. 1597–1600.

[15] L. Wu, X. He, X. Wang, K. Zhang, and M. Wang, “A survey on 
accuracy-oriented neural recommendation: From collaborative filtering to 
information-rich recommendation,” 2021, arXiv:2104.13030.

[16] P. V. Marsden and N. E. Friedkin, “Network studies of social influence,” 
Sociol. Methods Res., vol. 22, no. 1, pp. 127–151, 1993.

[17] J. Jamali and M. Ester, “A matrix factorization technique with trust prop-
gation for recommendation in social networks,” in Proc. 4th ACM Conf. 
Recommender Syst., 2010, pp. 135–142.

[18] B. Yang, Y. Lei, J. Liu, and W. Li, “Social collaborative filtering by trust,” 
IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 8, pp. 1633–1647, 
Aug. 2016.

[19] J. Tang, S. Wang, X. Hu, D. Yin, Y. Bi, Y. Chang, and H. Liu, “Rec-
ommendation with social dimensions,” in Proc. IEEE 10th Int. Conf. 
Comput. Netw., Aug. 2016.

[20] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, “DiffNet++: 
A neural influence and interest diffusion network for social recom-
mandation,” IEEE Trans. Knowl. Data Eng., early access, Dec. 31, 2021, doi:
10.1109/TKDE.2020.3048414.

[21] R. van den Berg, T. N. Kipf, and M. Welling, “Graph convolutional matrix 
completion,” 2017, arXiv:1706.02263.

[22] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, 
“Graph convolutional neural networks for web-scale recommen-
der systems,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. 
Discovery Data Mining, Jul. 2018, pp. 974–983.

[23] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, 
and M. Grohe, “Weisfeiler and Leman go neural: Higher-order graph neu-
rnal networks,” in Proc. 42nd ACM SIGIR Conf. Res. Develop. Inf. Retr., 
Jul. 2019, pp. 235–244.

[24] L. Wu, J. Li, P. Sun, R. Hong, Y. Ge, and M. Wang, “DiffNet++: 
A neural influence and interest diffusion network for social recommen-
dation,” IEEE Trans. Knowl. Data Eng., early access, Dec. 31, 2021, doi:
10.1109/TKDE.2020.3048414.

[25] R. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 
2014, arXiv:1412.6980.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and 
I. Goodfellow, “Dropout: A simple way to prevent neural networks 
from overfitting,” J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929–1958, 
Jan. 2014.