Interest-Aware Message-Passing Layer-Refined Graph Convolutional Network for Recommendation

Xingyao Yang *, Jinchen Wu and Jiong Yu

School of Software, XinJiang University, Urumqi 830008, China; 107552002971@stu.xju.edu.cn (J.W.)
* Correspondence: yangxy@xju.edu.cn

Abstract: Graph convolutional networks (GCNs) show great potential in recommendation applications, as they have excellent performance in propagation node information propagation and capturing high-order connectivity in user-item interaction graphs. However, in the current recommendation model based on GCN, incoming information from neighbors is aggregated during information propagation, and some of this information may be noisy due to negative information. Additionally, the over-smoothing problem occurs when the model layers are stacked too high. During the embedding learning of users in the graph convolution operation, an important factor is that high-order neighbor users with no interest are involved, leading to similar embedding for users with no interest. These issues can degrade the recommendation performance. To address these problems, this paper proposes a method called IMPLayerGCN. In this method, high-order graph convolution is performed within subgraphs, which are composed of users with similar interests and their interaction items. The higher-order graph convolution is carried out in the subgraph and the layer representation in the process of GCN re-information propagation and node update is refined. The convolution process uses a symmetrical matrix. This approach avoids the spread of negative information from higher-order neighbors to embedded learning.

Keywords: graph convolutional networks; recommender systems; over smoothing; high-order graph convolution; symmetrical matrix

1. Introduction

GCN-based models [1–3] have achieved great success in the field of recommendation systems due to their powerful non-Euclidean structure representation learning ability. The core is that through information iterative propagation, each node uses first-order local convolution operations to aggregate the feature information of its neighboring node. GCN extracts additional information from the node relationship graph, which not only improves the representation of learned users and items, but also alleviates the sparsity problem in the dataset. In the bipartite graph composed of users and items, nodes can obtain more high-order neighborhood information by combining the aggregated information of their neighborhoods with their own embeddings and updating their own embeddings [4]. The updated node embedding also encodes the degree information of all nodes in its higher-order neighborhoods. Therefore, GCN-based recommendation models can usually achieve more desirable recommendation performance.

In the current study, the GCN-based model shows excellent results in the recommendation and node classification tasks. However, the model also has the problem of excessive complexity and redundant calculation in the iterative process. For example, in LightGCN [1], information is linearly aggregated and propagated through renormalized adjacency matrices, which substantially reduces the computational complexity of a large number of user-item interaction graphs, but feature propagation hides embeddings locally in the graph, making user embeddings or item embeddings more similar and eventually predicted in locally connected similar nodes. This is known as the over-smoothing problem.
This can make the preferred items homogeneous across users and lead to degradation of recommendation performance. Chen et al. proposed a model LR-GCCF [5] that removes the nonlinear transformation in GCNs, simplifies the network structure, and introduces the residual network structure. This alleviates the over-smoothing problem and improves the recommendation accuracy.

The above two simplified GCN recommendation models have achieved success in recommendation performance, but from the perspective of graph convolution, the over-smoothing problem in the recommendation field is still not well considered. The GCN-based recommendation model is built on the basis of the user-item graph, connecting users to items based on the history of their interactions with items. User embeddings are learned by iteratively aggregating the messages passed from user nodes and item nodes in the neighborhood, where the passed messages are extracted from the neighbor node embeddings. When stacking k layers, the information of the top k neighbor nodes indirectly connected by the user and the item is also integrated into the embedding learning of the target node. In this way, the information from the higher-order neighborhood enriches the information of the current node. However, not all information from higher-order neighborhoods is positive. In the user-item interaction graph, it is possible that higher-order neighboring user nodes appear to have no common or contradictory interest profiles with the current node, especially when constructed based on implicit feedback, such as when the user initiates a click behavior. Additionally, after stacking convolutional layers, the nodes become more similar, which leads to the problem of over-smoothing. In addition to the problem of over-smoothing, additional noise due to user uncertainty can reduce the recommendation accuracy. Most recommendation models based on user ratings or implicit feedback cannot solve this problem. To reduce the impact of noisy information interactions on recommendation results, one of the existing models reduces noise by removing random effects [6], and random descent cannot distinguish the importance of edges, which hinders the information aggregation of low frequency nodes, resulting in slow convergence of training. Another approach is to learn how to reduce noise in methods [7,8], by calculating attention score between connected nodes to learn edge weights. This method heavily depends on the quality of node representations.

Based on the above problems, this paper proposes a layer-refined graph convolutional neural network recommendation algorithm called IMPLayerGCN that focuses on interest transfer. The proposed model first groups the user and the items that interact with the user into different subgraphs, and performs high-order convolution within these subgraphs. For higher-order graph convolutions, only messages from nodes in the same subgraph domain are utilized to learn node embeddings. In this way, the current node contains more and more high-order neighborhood information. At the same time, making the embedding of nodes more similar, this paper uses convolutional layer refinement to ensure that nodes can guarantee self-independent features in deep network embedding. In addition, the updated node embeddings also encode the degree information of all nodes in their high-order neighborhoods. The main contributions of this paper are as follows:

- The constructed user-item graph is grouped into subgraphs by using user characteristics and graph structure information, and the user nodes in the subgraph can have more similar interests than the user nodes in other subgraphs.
- In this paper, we propose a layer refinement graph convolutional neural network recommendation algorithm IMPLayerGCN that focuses on interest transfer. This model refines the hidden layer in the process of network propagation to alleviate the natural noise introduced by external factors in the recommendation scenario and make the user–item interaction graph sparser.
- A large number of experiments are carried out on the e-commerce dataset. The results show that the proposed model has a significant improvement on the same indicators compared to other recent baseline models.
2. Related Work

With the rise of deep learning, collaborative deep learning recommendation systems have been proposed, such as NCF [9], PinSage [10], and other models, to effectively learn the potential embedding of users and items. However, these models cannot capture the result information of user-item interaction graphs. Since GCN was first used to generalize CNN in graph structure data, GCN and its variants have been successfully applied to solve various recommendation problems, such as LightGCN, NGCF [11], GC-MC [12] and other methods. GCN can better represent the bipartite graph based on user-item interaction data. It can not only learn high-order neighbor information but also learns based on the graph structure. Among them, GC-MC and NGCF are designed based on collaborative filtering. In these models, GCN is used to propagate information through a normalized adjacency matrix, and the information received from neighbors is aggregated through nonlinear activation layer and linear conversion layer. LR-GCCF is designed to solve the over-smoothing problem by incorporating residual preference learning in GCN. LightGCN and SGC [13] simplify vanilla GCN by removing the nonlinear activation and feature conversion layers in GCN. Although, the simplified GCN has been effectively improved in computing performance, but it is easy to fall into the shallow suboptimal space due to the over-smoothing problem. Self-supervised methods such as GDCL [14], SelfCF [15], and OCCF [16] use graph augmentation methods for user and item representation, but they mainly use LightGCN as the core, so there is still an over-smoothing problem. To address the over-smoothing problem, UltraGCN [17] skips the infinite message passing of graph convolutions by performing a constrained loss on user–item interactions. IMPGCN [18] groups subgraphs according to users and performs graph convolution operations within the generated subgraphs. However, in the stacking of too many convolutional layers, the problem of nodes becoming similar still occurs.

There are some high-order relationships in the user–item interaction information, which should also be taken into account. Some current GCN models pay attention to the global relationship in the graph nodes. MEIRec [19] utilizes meta-paths to visit neighbor nodes and leverage rich structural information for intent recommendation. SMOGCF [20] directly captures high-order connectivity between adjacent nodes of any order for node information learning. MultiGCCF [21] combines user–user graphs and item–item graphs based on user–item bipartite graphs in embedding learning. A2-GCN [22] emphasizes the importance of attribute information to user preferences and designs an attribute perception attention mechanism to capture the influence of item attribute information on user preferences. This mechanism uses attribute node information connected to an item to help learn the item’s attention vector relative to the target user. CrossGCN [23] builds a crossover feature graph convolutional module, which encodes high-order crossover features in any order and has linear complexity based on feature dimension, so that the graph neural network can achieve better performance in the scene where crossover features are important. DHCN [24] uses hypergraph modeling to capture high-order relationships between items and cross-session information and integrates self-supervision into graph neural network training to enhance hypergraph modeling and improve performance. The indiscriminate use of high-order neighboring nodes is also an important reason for the over-smoothing problem in the GCN-based recommendation model, and noise information is also added to the node information.

3. Methodology

3.1. Framework

3.2. Recap

Let \( U = \{u_1, u_2, u_3, \ldots, u_n\} \) denote a set of users and \( I = \{i_1, i_2, i_3, \ldots, i_m\} \) denote a set of items, where \( n, m \) represent the number of users and items, respectively. We construct a user–item interaction graph \( G = (W, E) \) by the set of users and items, where \( W \) denotes the set of nodes consisting of two types of user nodes and item nodes; \( E \) denotes the set of edges, and if user \( u \) has an interaction on the item \( i \), there is an edge \( e_{u, i} \in E \).
The GCN-based recommendation algorithm is subject to noise or negative information during the propagation of the node embeddings, which can lead to degradation of the final recommendation performance, which is due to the users’ interests usually covering a range of items. Different users may have very different interests, and some items may even contradict each other. Without distinguishing users, information will be embedded and propagated among users who may have completely different interests. To avoid this situation and alleviate the over-smoothing problem, this paper divides the user–item interaction graph $G$ into $g_s$ subgraphs using the strategy of grouping users with similar interests and with their associated sets of interaction items into the same subgraph $G_s$, where $s \in \{1, \ldots, g_s\}$. The IMPLayerGCN framework is shown in Figure 1.

### 3.3. Layer Refined for Subgraph Learning

The model in this paper takes reference from the LightGCN and improves on its design. Let $\mathcal{N} = |W|$ represent the number of node sets $W$ containing items and users and $\mathcal{M} = |E|$ represent the number of interaction edge sets, and denote the adjacency matrix as $B \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}}$ and the diagonal matrix as $D$. In the field of recommendation, the user–item interaction graph can be processed using a bipartite graph with the source node as the user and the target node as the item. Given that $\mathcal{N} = \mathcal{N}_u + \mathcal{N}_i$, the number of users is denoted as $\mathcal{N}_u$ and the number of items is denoted as $\mathcal{N}_i$. The node embedding in the self-layer is denoted as $X^{(0)} \in \mathbb{R}^{\mathcal{N} \times T}$, where $T$ is the node embedding size. $X^{(k)}$ denotes the node embedding representation in the $k$-th layer. The transition matrix is defined as $D^{-1/2}BD^{-1/2}$ in LightGCN. The simplified graph convolution layer is defined as:

$$X^{(l+1)} = (D^{-1/2}BD^{-1/2})X^{(l)} \quad (1)$$

The node embedding of the $l+1$-th hidden layers is only linearly aggregated from the $l$-th layer using the transition matrix $D^{-1/2}BD^{-1/2}$. Existing GCN-based recommendation models typically aggregate the self-layer and all hidden layers obtained from each propagation to update the final node embedding. After the above operation, a function
READOUT is used to aggregate the features of all layers of a node to obtain the final node representation \( \mathbf{X} \). The procedure is as follows:

\[
\mathbf{X} = \text{READOUT}(\mathbf{X}^{(0)}, \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(L)})
\]  

where the function \( \text{READOUT} \) can be any differentiable function, such as the mean function used in LightGCN for the final embedding update.

After iterative aggregation of features from higher-order nearest neighbors, the nodes will not be able to maintain their unique features and cause the embedding to become increasingly similar, thus leading to over-smoothing of the problem. Additionally, they do not distinguish the heterogeneous features of higher-order nodes during aggregation, and the noise from higher-order neighbors affects the current node embedding learning. Recommendation models such as LightGCN and SGC achieve optimal performance at most at 3 or 4 layers. In addition to the over-smoothing effect, this paper argues that nodes also receive noisy or negative information during embedding information propagation, which also affects the final performance. This is because users’ interests usually cover a range of items, and different users may have very different interests, and without differentiating between users, embedding propagation may be performed among users with different interests in order to learn embeddings during the graph convolution process. To avoid this issue, this paper groups users with similar interests into the same subgraph and limits the propagation of embedding information to within subgraphs. A node in a subgraph can only make use of information from neighboring nodes in that subgraph. Because the items that a user interacts with all belong to that user’s subgraph, the user can still obtain information from all their connected items. For the item node, its direct user neighbors can be distributed in different subgraphs. According to the above graph convolution process, similarly, an akin symmetric transition matrix \( \mathbf{\hat{B}}_s = \mathbf{D}_s^{-1/2} \mathbf{B}_s \mathbf{D}_s^{-1/2} \) is used in this paper for subgraph convolution, where the neighbourhood matrix \( \mathbf{B}_s \in \mathbb{R}^{N_s \times N_s} \) in the subgraph is constructed from the user–item interaction matrix \( \mathbf{A}_s \in \mathbb{R}^{N_s \times N_i} \) as follows:

\[
\mathbf{B}_s = \left( \begin{array}{cc}
0 & \mathbf{A}_s \\
\mathbf{A}_s^T & 0
\end{array} \right)
\]  

In matrix \( \mathbf{B}_s \), a non-zero entry \( A_{ui} \in \mathbf{B}_s \) indicates that user \( u \) has had a previous interaction record with item \( i \) with a value of 1, otherwise \( A_{ui} = 0 \). Some recent studies \([25,26]\) have shown that GCN does not scale well to deep architectures as the stacking of multi-layer graph convolution leads to a high complexity in backpropagation. In practice, many GCN-based recommendation models achieve good recommendation performance when the number of convolution layers is 2 \([10,27]\). However, the final goal of our task is to learn information embedding that represents the attributes of each user or item, and the information propagated through multiple layers is often too smooth to distinguish between different nodes. During information propagation in subgraph \( G_s \), this paper designs a layer-refined mechanism to dynamically extract information from the self-layer and focus on the unique interests or attributes of the current user or item. The information propagation in this model is defined as:

\[
\mathbf{X}_s^{(l+1)} = \mathbf{\hat{B}}_s \mathbf{X}_s^{(l)}
\]

\[
\mathbf{X}_s^{(l+1)} = (\mathbf{a}^{(l+1)} + \epsilon)\mathbf{X}_s^{(l+1)}
\]

where \( \mathbf{X}_s^{(l)} \) denotes the node’s final embedding obtained after the \( l \)-th layer of graph convolution in subgraph \( G_s \), \( \mathbf{a}^{(l+1)} \in \mathbb{R}^{N_s} \) is the node similarity vector between the current layer and the self-layer, and \( \epsilon \) is a quantity that prevents the zero vector from being too infinitesimal in \( \mathbf{X}_s^{(l+1)} \).

\[
\mathbf{a}^{(l+1)} = \text{SIM}(\mathbf{X}_s^{(l+1)}, \mathbf{X}_s^{(0)})
\]
In this paper, the cosine similarity is used as the SIM function for this model. The cosine similarity function between two row vectors $x_i = X^{(l+1)} |i, :|$ and $x_j = X^{(0)} |j, :|$ is calculated as follows:

$$\text{SIM}(x_i, x_j) = \frac{x_i \cdot x_j}{\max(||x_i||_2, ||x_j||_2)}$$

(7)

This function increases the fusion of hidden layers that are similar to the self-layer and reduces the effect of hidden layers that are different from the self-layer. Finally, the node’s final embedding is updated by removing the self-layer, as the information from the self-layer has been refined in the hidden layer. Equation (2) can be simplified as follows:

$$X = \text{READOUT}(X^{(1)}, X^{(2)} \ldots, X^{(L)})$$

(8)

Similarly, the node’s final embedding is obtained by refining the information in the self-layer in the sub-layer convolution as follows:

$$X^{(k)}_s = \text{READOUT}(X^{(1)}_s, X^{(2)}_s \ldots, X^{(l+1)}_s)$$

(9)

This paper uses sum aggregation to perform node updates, where $k = l + 1$. After obtaining the node’s final embedding of each subgraph, the final node embedding of each subgraph is aggregated to obtain the final $k$-layer embedding in the following process:

$$X^{(k)} = \sum_{s \in G} X^{(k)}_s$$

(10)

Finally, this model combines the embeddings of all the layers to obtain a final representation of the user and the item in the following process:

$$X = \alpha_0 X^{(0)} + \alpha_1 X^{(1)} + \cdots + \alpha_K X^{(K)}$$

(11)

where $\alpha_k \geq 0$ is the hyperparameter on the $k$-th layer, which denotes the importance of the $k$-th layer embedding in constituting the final embedding. As with LightGCN, its value is uniformly set to $\frac{1}{K+1}$. In the prediction phase, the final node embedding $X$ is applied to rank the candidate items for each target user, and the Top-K score is as follows:

$$\hat{r}_{ui} = x_u x_i^T, x_u \in X, x_i \in X$$

(12)

where $x_u \in \mathbb{R}^T$ and $x_i \in \mathbb{R}^T$ denote the final embedding of user $u$ and item $i$, respectively.

3.4. Generating the Subgraph

In this section, it will be described how to generate subgraph $G_s$ from user-item interaction graph $G$. The subgraph is used to group users with similar interests in the model. The user grouping is described as a classification task; that is, each user is classified into a group, and each user is represented by a feature vector. The graph structure and user embedding are fused. The process is defined as follows:

$$M_u = \sigma(W_1 (\epsilon_u^{(0)} + \epsilon_u^{(1)}) + b_1)$$

(13)

where, $M_u$ denote the user feature obtained by feature fusion, $\epsilon_u^{(0)}$ denotes the embedded representation of user ID, $\epsilon_u^{(1)}$ denotes the feature representation obtained by local neighbors in the aggregation graph, that is, user embedding after propagation in the first layer. $W_1 \in \mathbb{R}^{d \times d}$ denotes the weight matrix of the fusion method, $d$ denotes the embedding dimension, $b_1 \in \mathbb{R}^{1 \times d}$ denotes the deviation vector of the fusion method. $\sigma$ denotes the activation function. In this paper, LeakyReLU is used as the activation function because it can encode both positive and negative signals. In order to classify users into different
subgraphs, this paper uses the user features obtained by the two-layer neural network to convert into a prediction vector. The process is defined as follows:

\[ h_u = \sigma(W_2 M_u + b_2) \]  

\[ p_u = W_3 h_u + b_3 \]  

where, \( p_u \) is the prediction vector. The position of the maximum value in \( p_u \) indicates the group or subgraph to which the user belongs. \( W_2 \in \mathbb{R}^{d \times d} \) and \( W_3 \in \mathbb{R}^{d \times N} \) denote the weight matrix of the two layers, \( b_2 \in \mathbb{R}^{1 \times d} \) and \( b_3 \in \mathbb{R}^{1 \times N} \) denote the offset vectors of the second layer. For users with similar embedding, Equations (14) and (15) will generate similar prediction vectors; that is, they are grouped into the same group. The purpose of subgraph generation is to construct a matrix representing the user–item adjacencies in the subgraph based on the user grouping results and the original user–item Laplacian matrix. For the matrix of each subgraph, if the corresponding user is not in the user group, this paper filters out the user–item adjacency in the Laplacian matrix of the original user–item graph based on the obtained user group information.

3.5. Model Optimization

The task objective of this paper is to recommend a set of top k items to the target user to satisfy the user’s preference. Optimization uses a two-by-two learning approach. In order to perform pairwise learning, a triple \( \{ u, i^+, i^- \} \) needs to be constructed, where \( u \) denotes the user, \( i^+ \) denotes the positive item, \( i^- \) denotes the negative item, there is an observed interaction between \( u \) and \( i^+ \), and there is an unobserved interaction between \( u \) and \( i^- \). In this method, positive items should be ranked higher than negative items. The objective function is expressed as:

\[ \arg \min_{(u,i^+,i^-) \in O} -\ln \phi(\hat{r}_{ui^+} - \hat{r}_{ui^-}) + \lambda \| \Theta \|^2_2 \]  

where \( O = \{ (u,i^+,i^-) | (u,i^+) \in \mathcal{R}^+, (u,i^-) \in \mathcal{R}^- \} \) denotes the training set. \( \mathcal{R}^+ \) denotes the set of users \( u \) and \( i^+ \) observed to have interactions in the training dataset. \( \mathcal{R}^- \) denotes the sampled unobserved interaction set. \( \lambda \) and \( \Theta \) denote the regularization weights and parameters of the model, respectively. \( L_2 \) regularization is used to prevent overfitting. A small batch optimizer, Adam, is used to optimize the prediction model and update the model parameters. Specifically, for a randomly sampled batch of triples \( \{ u, v^+, v^- \} \in O \), the representations of these users and items are first learned by propagation rules, and then the model parameters are updated using the gradient of the loss function.

4. Experiments

4.1. Datasets

In order to evaluate the effectiveness of IMPLayerGCN, the experiments in this chapter are conducted on two benchmark datasets. The first dataset is the Grocery and Gourmet Food dataset provided by Amazon, denoted as the Amazon dataset. Each entry in the dataset records the rating value of the user on the item. The second dataset is the Taobao dataset, which contains users’ click, purchase, add-items-to-cart, product preference, and other behaviors on the e-commerce website Taobao. After processing the above two raw datasets, the interaction data, item data, and user data are recorded separately. The basic statistics of the datasets are presented in Table 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Count of Users</th>
<th>Count of Items</th>
<th>Interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>115,144</td>
<td>39,688</td>
<td>1,025,169</td>
</tr>
<tr>
<td>Taobao</td>
<td>82,633</td>
<td>136,710</td>
<td>4,230,631</td>
</tr>
</tbody>
</table>

Table 1. Statistics of the experimental datasets.
For the two datasets, this paper adopts the strategy to retain the user and item items with at least 10 interactions, and as can be seen from Table 1, the datasets differ in size and sparsity. The datasets used for training, validation, and testing are set as follows: First, the user–item interactions for each dataset are sorted in chronological order according to the user–item interaction timestamp. Then, the top 70% of user–item interactions are used as training data, the next 10% are used as validation data, and the remaining 20% are used as test data. In the validation and test datasets, cold-start users and items that did not appear in the interaction data were removed. For the observed user–item interactions, they are regarded as positive instances, and for the methods that adopt the pairwise learning strategy, we randomly sample a negative instance that the user has not used before and pairs it with each positive instance.

4.2. Baselines

IMPLayerGCN is compared with the following state-of-the-art baseline models in two datasets.

- LightGCN [1] is a simplified graph convolutional network with only linear propagation and aggregation among neighbors. It averages the hidden layer embeddings for predicted end-user/item embedding calculations.
- A2-GCN [22] learns final embeddings using node classification and graph classification.
- CrossGCN [23] obtains cross-features on the basis of traditional matrix factorization methods and enhances feature learning capabilities.
- DHCN [24] uses the hypergraph to model user and item information, maximizes the interaction information between graph network learning nodes by constructing the line graph of the hypergraph, and integrates self-supervised learning into the training of the network for recommendation.

4.3. Experimental Settings

In this experiment, the user and item embedding values of the model are fixed at 64, the Adam optimizer is used, and the Xavier method is used to initialize the embedding parameters. In order to make a fair comparison, the comparison model is based on the hyperparameters tuned by each model in the published paper. If no hyperparameters are specified, the range of values is determined in the ablation experiment in this paper. For the model IMPLayerGCN in this paper, the number of sublayers is set to 4, and the adjustment range of the coefficient $\lambda$ of the $L_2$ regularization term is set to $\{0.01, 0.001, 0.0001, 0.00001\}$.

In the Top-K recommendation, this paper uses two widely used metrics to evaluate the model, namely NDCG@K and Recall@K, with $K = 10, 20, 50$. To simplify the representation, denote NDCG@K as N@K and Recall@K as R@K.

4.4. Results

In this section, the comparison results of different baseline models and our model IMPLayerGCN will be given. Under $n$ different scenarios in the Amazon and Taobao datasets, the performance of our proposed IMPLayerGCN model is compared with other baseline models. The results are shown in Tables 2 and 3.

Table 2. The comparison of Experimental results in Amazon.

<table>
<thead>
<tr>
<th>Model</th>
<th>N@10</th>
<th>N@20</th>
<th>N@50</th>
<th>R@10</th>
<th>R@20</th>
<th>R@50</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2-GCN</td>
<td>0.0411</td>
<td>0.0541</td>
<td>0.0801</td>
<td>0.0545</td>
<td>0.0736</td>
<td>0.0929</td>
</tr>
<tr>
<td>DHCN</td>
<td>0.0420</td>
<td>0.0554</td>
<td>0.0806</td>
<td>0.0572</td>
<td>0.0782</td>
<td>0.1038</td>
</tr>
<tr>
<td>LightGCN</td>
<td>0.0425</td>
<td>0.0557</td>
<td>0.0811</td>
<td>0.0584</td>
<td>0.0766</td>
<td>0.1157</td>
</tr>
<tr>
<td>CrossGCN</td>
<td>0.0502</td>
<td>0.0632</td>
<td>0.0985</td>
<td>0.0658</td>
<td>0.0854</td>
<td>0.1316</td>
</tr>
<tr>
<td>IMPLayerGCN</td>
<td>0.0539</td>
<td>0.0777</td>
<td>0.1034</td>
<td>0.0699</td>
<td>0.0998</td>
<td>0.1470</td>
</tr>
</tbody>
</table>
Table 3. The comparison of experimental results in Taobao.

<table>
<thead>
<tr>
<th>Model</th>
<th>N@10</th>
<th>N@20</th>
<th>N@50</th>
<th>R@10</th>
<th>R@20</th>
<th>R@50</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2-GCN</td>
<td>0.1243</td>
<td>0.1292</td>
<td>0.1395</td>
<td>0.1351</td>
<td>0.1471</td>
<td>0.1574</td>
</tr>
<tr>
<td>DHCN</td>
<td>0.1239</td>
<td>0.1292</td>
<td>0.1384</td>
<td>0.1359</td>
<td>0.1489</td>
<td>0.1692</td>
</tr>
<tr>
<td>LightGCN</td>
<td>0.1275</td>
<td>0.1337</td>
<td>0.1403</td>
<td>0.1391</td>
<td>0.1551</td>
<td>0.1617</td>
</tr>
<tr>
<td>CrossGCN</td>
<td>0.1320</td>
<td>0.1475</td>
<td>0.1668</td>
<td>0.1415</td>
<td>0.1631</td>
<td>0.1880</td>
</tr>
<tr>
<td>IMPLayerGCN</td>
<td>0.1394</td>
<td>0.1560</td>
<td>0.1741</td>
<td>0.1419</td>
<td>0.1762</td>
<td>0.2094</td>
</tr>
</tbody>
</table>

According to Tables 2 and 3, the model IMPLayerGCN in this paper has achieved better performance than other models in the Top-K recommendation. Compared with LightGCN, NDCG in Amazon dataset and Taobao dataset increased by 31.27% and 16.70%, respectively, and Recall increased by 25.68% and 15.04%, respectively. Compared with recent CrossGCN, NDCG in the Amazon dataset and Taobao dataset increased by 11.76% and 5.25%, respectively, and Recall increased by 11.60% and 6.56%, respectively. The reason is that IMPLayerGCN divides the user–item graph into subgraphs, which can enable user nodes in the subgraphs to obtain more similar interest information and pay more attention to the learning of users’ own interests. Meanwhile, as performing high-order convolution in each subgraph and using node information in the same graph domain to learn node embedding, the current node contains more high-order information. The refinement of convolution layer in our model ensures that the current node has its own features in the deep hidden layer propagation. Although A2-GCN filters out items unrelated to users through item attributes, some high-order relationships between nodes and graphs generate noise information. CrossGCN obtains the cross features of nodes by the traditional matrix factorization method, and focuses on the learning ability of enhanced features. However, it can only model local feature interactions, which ignores the capture of global features and then enables nodes to access limited node information. DHCN uses hypergraphs to model users and items and their interaction order and captures high-order information between nodes. In this process of global information propagation, external noise is introduced, resulting in a decline in recommendation performance. LightGCN captures the high-order relations between nodes with stacking too many convolution layers, which makes the nodes similar to each other. Thus, LightGCN can only learn coarse-grained high-order relationships and cannot avoid the influence of negative information of high-order neighborhoods on current node information. Under each evaluation metric, LightGCN has achieved better performance than the baseline model.

To verify the effectiveness of the model’s middle layer refinement module in alleviating the over-smoothing problem, it is first necessary to know the coverage rate of nodes involved in the number of convolutional layers in different graphs. In the two datasets Amazon and Taobao, it is shown that when the convolutional layers are stacked up to seven layers, and a node has collected the information of almost all nodes, as shown in Figure 2.

The message passing strategy adopted in the convolutional layer is the same as that adopted in LightGCN, and the number of layers is increased from two to seven. The experimental results are shown in Figures 3 and 4.
Figure 2. The average coverage of nodes involves in different graph convolution layers.

In Figures 3 and 4, IMPlayerGCN performs better than LightGCN in both datasets when more than two layers are stacked. This indicates that the proposed model can learn better embeddings through the layer refinement module. Secondly, the performance of LightGCN reaches its peak when it is stacked with three or four layers. If the number of layers is increased, the performance will decrease significantly, indicating that the deep structure of LightGCN is over-smooth. The IMPlayerGCN shows better performance on deeper networks.

Figure 3. The comparison results of IMPlayerGCN and LightGCN under different layers in Taobao dataset.
Combined with the comparison results of IMLayerGCN and LightGCN in two datasets under different layers, the results are shown in Figure 2. It can be shown that the IMLayerGCN model can effectively alleviate the over-smoothing problem. IMLayerGCN-n indicates models with n subgraphs, respectively. For example, n is 2, which means that the user-item graph is divided into 2 subgraphs in the IMLayerGCN model. IMLayerGCN works best when the number of subgraphs is 2, and when the number of convolutional layers is no more than 3. This is because in the process of embedding and propagation of node information, the nodes in the subgraph of IMLayerGCN-2 can access more nodes in a short distance than those with three or four subgraphs. IMLayerGCN-3 has the best performance when the number of convolution layers is more than three. After three-layer convolution, the number of nodes involved in embedding propagation increases sharply. In general, each node in the graph with the number of subgraphs is two should be able to access more nodes than that in the graph with the number of subgraphs is three and four, but the IMLayerGCN-2’s performance improvements are minor. This shows that there is still noise information in the embedding propagation of user interest at the coarse level. With more subgraphs, on the one hand, users with similar interests can be distinguished at a more granular level to better extract information from higher-level neighbors. On the other hand, the moleculegram method cuts off connections with other short-distance nodes that provide more value, so IMLayerGCN’s performance at three and four subgraphs differs less from that at two subgraphs as the number of layers increases. Therefore, in order to further study the influence of subgraphs, the average coverage rate of each node and other performance indicators of LightGCN and IMLayerGCN are tested. The number of convolutional layers in LightGCN is set to three, and the number of convolutional layers in IMLayerGCN-3 is set to six. These are the optimal performance settings of the two models in Amazon. The experimental results in recall and node coverage are shown in Figure 5.

By grouping users with similar interests into subgraphs, information can propagate within subgraphs in IMLayerGCN, allowing for more layers of graph convolution to extract positive information from higher-order neighbors. In contrast, LightGCN is limited by the negative information of the high order neighborhood and can only improve by more than three levels. Comparing the performance of different subgraphs, the performance of IMLayerGCN significantly improves with increased coverage. By stacking six layers, the user nodes in the subgraph are connect to almost all other nodes in the entire graph, meaning that users in the subgraph interact with almost all of the items in the graph. Furthermore, IMLayerGCN can achieve high coverage without over-smoothing, indicating that the item embedding learned in the graph helps the users to embed learning in the graph, and the information extracted from the subgraph during the graph convolution helps to embed learning of all nodes in the subgraph.
To prevent overfitting, the regularization coefficient $\lambda$ is introduced as it is an important model of super parameters, with a range of $\{0.01, 0.001, 0.0001, 0.00001\}$. Different values of the regularization coefficient were set in the Taobao and Amazon datasets to verify the influence of $\lambda$ on the model. The experimental results are shown in Figure 6.

Figure 6. The result of different regularization coefficients.

In Figure 6, the model shows the best performance in the Taobao dataset when the regularization coefficient value is 0.001, while in the Amazon dataset, the best performance is achieved when the value is 0.0001. The experiment revealed that the model is not sensitive to changes in the pair, indicating that it has good generalization ability.
4.5. Ablation Study

In this section, we design three variants of the model to verify the effect of using graph structure information and first-order propagation information on the model in constructing subgraphs: IMPLayerGCN-SI, IMPLayerGCN-1st, and IMPLayerGCN-LR.

- IMPLayerGCN-SI: It removes graph structure information from the subgraph generation module.
- IMPLayerGCN-1st: It uses first-order propagation in subgraph learning.
- IMPLayerGCN-LR: It removes the layer refinement module.

Experiments on model variants were conducted in two datasets, and the experimental results are shown in Tables 4 and 5.

The results in Tables 4 and 5 show that the comprehensive performance of IMPLayerGCN is higher than that of its variants, and the performance of its variant model is better than that of the baseline model, which indicates that it is effective to add graph structure information to the construction of subgraph module, and the graph structure information can enrich node information. Applying first-order propagation to subgraph embedding leads to slightly lower results because first-order convolution contains all embedded information of users and items, which makes nodes in the subgraph learn noise information. After adding a layer refinement convolution structure to the subgraph, users can focus on their own interests and alleviate the influence of other similar users’ interests.

Table 4. The comparison of experimental results in Amazon

<table>
<thead>
<tr>
<th>Model</th>
<th>N@10</th>
<th>N@20</th>
<th>N@50</th>
<th>R@10</th>
<th>R@20</th>
<th>R@50</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPLayerGCN-SI</td>
<td>0.0531</td>
<td>0.0698</td>
<td>0.0966</td>
<td>0.0663</td>
<td>0.0794</td>
<td>0.1331</td>
</tr>
<tr>
<td>IMPLayerGCN-1st</td>
<td>0.0489</td>
<td>0.0637</td>
<td>0.0912</td>
<td>0.0656</td>
<td>0.0789</td>
<td>0.1304</td>
</tr>
<tr>
<td>IMPLayerGCN-LR</td>
<td>0.0522</td>
<td>0.0643</td>
<td>0.0948</td>
<td>0.0675</td>
<td>0.0856</td>
<td>0.1367</td>
</tr>
<tr>
<td>IMPLayerGCN</td>
<td>0.0539</td>
<td>0.0777</td>
<td>0.1034</td>
<td>0.0699</td>
<td>0.0998</td>
<td>0.1470</td>
</tr>
</tbody>
</table>

Table 5. The comparison of experimental results in Taobao

<table>
<thead>
<tr>
<th>Model</th>
<th>N@10</th>
<th>N@20</th>
<th>N@50</th>
<th>R@10</th>
<th>R@20</th>
<th>R@50</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPLayerGCN-SI</td>
<td>0.1342</td>
<td>0.1503</td>
<td>0.1718</td>
<td>0.1368</td>
<td>0.1632</td>
<td>0.1891</td>
</tr>
<tr>
<td>IMPLayerGCN-1st</td>
<td>0.1321</td>
<td>0.1488</td>
<td>0.1670</td>
<td>0.1379</td>
<td>0.1613</td>
<td>0.1834</td>
</tr>
<tr>
<td>IMPLayerGCN-LR</td>
<td>0.1356</td>
<td>0.1536</td>
<td>0.1715</td>
<td>0.1418</td>
<td>0.1730</td>
<td>0.1976</td>
</tr>
<tr>
<td>IMPLayerGCN</td>
<td>0.1394</td>
<td>0.1560</td>
<td>0.1741</td>
<td>0.1419</td>
<td>0.1762</td>
<td>0.2094</td>
</tr>
</tbody>
</table>

5. Conclusions

In the recommendation model based on GCN, the indiscriminate use of higher-order nodes may introduce negative information in embedding propagation, leading to performance degradation when stacking more layers. To address this, the IMPLayerGCN model proposed in this paper learns user and item embeddings by employing higher-order graph convolution within subgraphs. Subgraphs are formed by a designed subgraph generation algorithm that groups users with similar interests and their interaction items into the same graph. In IMPLayerGCN, the embedding of a node learned in a subgraph only contributes to the embedding learning of other nodes in that subgraph. This way, IMPLayerGCN can effectively avoid bringing noisy information to the embedding learning. Experiments on large-scale real datasets demonstrate that IMPLayerGCN can be improved by stacking more layers to take advantage of information from higher-order neighbors, resulting in better performance compared to baseline experiments.

Author Contributions: X.Y., J.W., and J.Y. conceived the experiment; J.W. and J.Y. conducted the experiment; X.Y. and J.W. analyzed the results. All authors reviewed the manuscript. All authors have read and agreed to the published version of the manuscript.
Funding: This work was supported by the Education Department Project of Xinjiang Uygur Autonomous Region XJEDU2016S035, the Doctoral Research Start-up Foundation of Xinjiang University BS150237, and the National Natural Science Foundation of China 62262064, 61862060.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.


Conflicts of Interest: The authors declare no conflict of interest.

References


**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.