Electronic Research Archive

ERA, 31(2): 793−811.
DOI: 10.3934/era.2023040
Received: 07 October 2022
Revised: 10 November 2022
Accepted: 18 November 2022
Published: 01 December 2022

http://www.aimspress.com/journal/ERA

Research article

An adaptive preference retention collaborative filtering algorithm based on graph convolutional method

Bingjie Zhang[1], Junchao Yu[1], Zhe Kang[1], Tianyu Wei[1], Xiaoyu Liu[1] and Suhua Wang[2,*]

[1] Information Science and Technology, Northeast Normal University, Changchun 130117, China
[2] Computer Department, Changchun Humanities and Sciences College, Changchun 130117, China

* **Correspondence:** Email: wangsuhua@ccrw.edu.cn; Tel: +8643184536330; Fax: +8643184536330.

Abstract: Collaborative filtering is one of the most widely used methods in recommender systems. In recent years, Graph Neural Networks (GNN) were naturally applied to collaborative filtering methods to model users' preference representation. However, empirical research has ignored the effects of different items on user representation, which prevented them from capturing fine-grained users' preferences. Besides, due to the problem of data sparsity in collaborative filtering, most GNN-based models conduct a large number of graph convolution operations in the user-item graph, resulting in an over-smoothing effect. To tackle these problems, Adaptive Preference Retention Graph Convolutional Collaborative Filtering Method (APR-GCCF) was proposed to distinguish the difference among the items and capture the fine-grained users' preferences. Specifically, the graph convolutional method was applied to model the high-order relationship on the user-item graph and an adaptive preference retention mechanism was used to capture the difference between items adaptively. To obtain a unified users' preferences representation and alleviate the over-smoothing effect, we employed a residual preference prediction mechanism to concatenate the representation of users' preferences generated by each layer of the graph neural network. Extensive experiments were conducted based on three real datasets and the experimental results demonstrate the effectiveness of the model.

Keywords: collaborative filtering; recommender system; users' preference; graph neural network

1. Introduction

Recommender systems have attracted considerable attention from both the industry and academia [1,2]. Collaborative filtering (CF) is among the most important recommendation method and depends on a basic assumption that people with similar purchasing experiences in the past will make similar decisions in the future [3]. With the development of graph neural networks (GNN) [4,5], the application of GNN in recommender systems has become increasingly mature [6,7]. In recent years, the history of user-item interactions has been modeled as a user-item graph [8] and applied to knowledge graphs [9,10], heterogeneous graphs [11,12], and social recommendations [13]. In addition, the graph convolutional network (GCN) [14] has been applied extensively to recommender systems [15−19] and delivered promising performance. Compared with the traditional CF methods, the graph neural network has powerful feature extraction and learning capabilities. The graph convolution can capture the high-order neighborhoods information and the complex interaction relationship on the user-item graph, and generate the feature information of the local graph neighborhood of the user/item. First-order neighborhoods are the user's direct neighbors, second-order neighborhoods are the neighbors of the user's neighbors, and so on. Higher neighborhoods than first-order are called high-order neighborhoods. Meanwhile, GNN combines the target nodes own representation with it's neighborhood information to update the new representation of the target node. The graph convolutional method stacks multiple graph convolutional layers and can aggregate features of users and items. Furthermore, exploiting high-order relationships on the user-item graph can alleviate the problem of data sparsity in collaborative filtering [20].

Despite their enormous success, these methods assumed that all interactive items could be used to model user preferences, in which each item contributed the same in generating user preferences. However, the distinctions among the items determine the purchasing behaviors of users. For instance, different items have different attributes, such as functionality and appearance. When users purchase an item, some of them prefer functionality while others prefer its appearance. This prompts us to capture fine-grained preference. A more detailed and specific description of the users' preferences for the item, namely user fine-grained preferences. Similarly, item fine-grained preferences mean more granular features of items, for example, goods will be preferred by users because of their price or quality advantages.

Meanwhile, when the graph convolution operation generates the user's representation, it suffers from an over-smoothing effect on the high-order neighborhood. Therefore, when modeling user preferences, it is important to consider the difference between items to capture the fine-grained preferences of users/items and alleviate the over-smoothing on the user-item graph.

As is shown in Figure 1. Different ratings of items indicate their different contributions to user preferences. If items are similar in terms of importance, it is unclear whether the user assigns an item a high or low rating. However, if we consider differences between items, we may find that assign a high rating because the attributes are more consistent with the characteristics of a career. Hence, failing to discern the differences between items may limit the performance of the recommender system. It is therefore desirable to design a recommender system to adaptively identify differences among items to acquire fine-grained user preferences.

Although it is useful to distinguish the differences among items, three challenges are outstanding in this context. The first challenge stands in the large and complex user-item graph, which makes it difficult to capture the differences between items in the user-item graph. The second challenge involves

ways to incorporate the differences between items into modeling user preferences. It is also difficult to identify items that are important for modeling user preferences from among the large number with which he/she interacts. Besides, the user-item graph usually suffers from the problems of data sparsity and over-smoothing due to the cold start issue inherent in recommender systems. Hence, the third challenge pertains to the means of solving these problems in the user-item graph.

In this paper, we proposed an Adaptive Preference Retention Graph Convolutional Collaborative Filtering Method (APR-GCCF), which is an end-to-end model that differentiates distinctions between items to model user preferences. To address the first two challenges, we proposed an adaptive preference retention mechanism. It contains two important elements. First of all, when users interact with different items, the contributions of different items to the user's representation are different. So, we adopted a trainable weight pair for different neighborhoods that generated corresponding scores and then assign them to capture neighborhoods to the user's local interest. Moreover, when we used GNN to aggregate neighborhoods of user nodes, different neighborhoods would carry different item node information. So, we modeled the user and item preferences through two trainable mapping vectors to capture fine-grained user and item preferences. To address the third, we exploited a residual preference prediction mechanism. Among them, to alleviate the problem of data sparsity, we used a multi-layer GNN to aggregate higher-order neighbors. However, similar to previous work [21], multi-layer GNNs can cause an over-smoothing problem. The essence of over-smoothing is that too much information about the other nodes suppress the information of the nodes themselves. By using the residual network, we connected its embedding and the aggregated new vector when updating the node embedding. This implemented the decoupling of neighborhoods and their information; thus the problem of over-smoothing could be relieved.

The contributions of this paper are as follows:

- We use a residual preference prediction mechanism that exploits high-order relationships and the residual network, which can alleviate the problems of data sparsity and over-smoothing generated by the user-item graph.
- To distinguish the different contributions made by different items to user embedding, we design an adaptive preference retention mechanism to adaptively identify different types of information provided by different items, thus capturing a fine-grained representation of the user's preference.
- We use L0 regularizers to address the overfitting problem and apply a weighted random sampling strategy to select items that are important for user preference modeling. We validate the effectiveness of the proposed framework (APR-GCCF) on three empirically acquired datasets: Amazon, FilmTrust, and Yelp.

The following parts of this paper were organized as follows: In Section 2, relevant works related to collaborative filtering and graph convolution neural networks were reviewed. In Section 3, we defined the formulations of the recommendation problem in our paper. The framework proposed was presented in Section 4 and three empirically acquired datasets were verified in Section 5. Finally, we summarize our work here and suggested directions for future research in the area in Section 6.

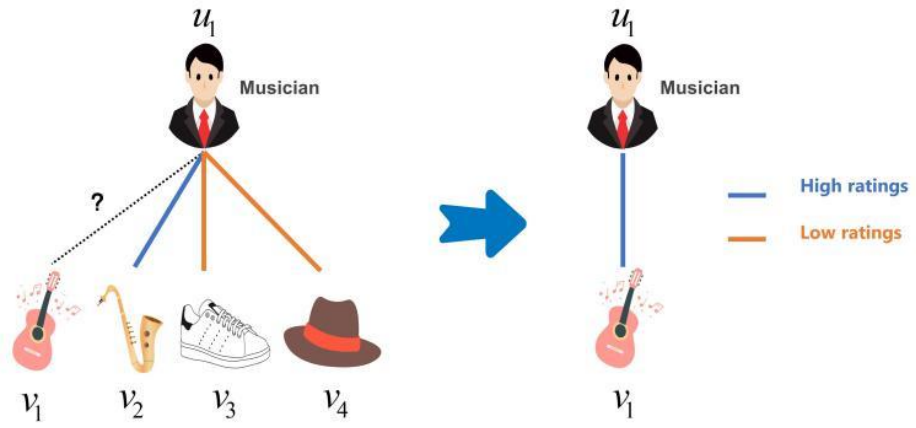


Figure 1. The impact of items on users when they purchase them.

2. Related works

This section focuses on two areas of research directly related to our study: collaborative filtering and graph neural networks.

2.1. Collaborative filtering

Collaborative filtering (CF) assumed that users with similar behaviors would have similar preferences for items. Most collaborative filtering algorithms that have been proposed in recent years used matrix factorization (MF) for the recommendation. Matrix factorization (MF) involved mapping users and items into a shared latent space and learning vector information to make recommendations. The PMF [22] utilized Gaussian distribution to model latent factors related to users and items, and then used their product to predict item ratings. BiasedMF [3] improved the PMF by combining user/item bias and a global bias. SVD++ [23] utilized implicit feedback (such as the number of clicks and identification of items by users as their favorite) to model user preferences. Deep learning models have yielded impressive performance in recommender systems in recent years. [24] combined collaborative filtering and neural networks to solve the cold start problem. NCF [9] employed deep learning to learn complicated internal interactions between users and items. AutoRec [25] utilized an encoder layer to project the user's/item's observed ratings into a latent space and reconstructs them using a decoder layer. Another relevant method involved modeling the history of user-item interactions as a user-item graph [8] to predict user preferences. The NGCF [26] adopted graph neural networks to capture higher-order connectivity and extracted a collaborative signal for a user-item graph. The LR-GCCF [27] exploited the residual network to solve the problem of over-smoothing in the user-item graph, and eliminated the nonlinearity in the GNN to improve recommendation performance. The MCCF [28] decomposed the user's motivations for purchasing items. Meanwhile, MCCF used node-level and component-level attention mechanisms to generate user/item preferences.

2.2. Graph neural networks

Graph neural networks were first proposed by [29−31]. The graph convolutional network (GCN) has lately achieved remarkable success in various tasks of graph-based analysis. It aggregated features derived from neighbors of the target node and propagated the information related to it in the graph to update new node embeddings. While different from the GCN, GraphSAGE [32] also aggregated features from local neighbors and used uniform sampling to generate a node embedding. The GAT [33] employed an attention mechanism [34] to adaptively assign weights to a node's neighbors when performing an aggregate operation. The GIN [35] operated through an injection-based neighborhood aggregation scheme. It applied a learnable mechanism of combining features to preserve the local graph structure and features of the nodes. The HAN [36] has considered the difference in importance among nodes in the convolution process, and the SGC [37] reduced the complexity of the GCN by removing feature transformations and nonlinear activations. It was used as a low-pass filter in the spectral domain to reduce the effects of over-smoothing as well. The developers of the DAGNN [38] claimed that the effects of entanglement of representational transformation and information propagation primarily affected the performance of the GNN and devised an adaptive fusion weight to adjust the effect of over-smoothing on it.

3. **Preliminaries**

In this section, we will introduce the notation used in this paper. We classically modeled the history of user-item interactions as a user-item graph $G = \{U, V, R, E\}$ [28]. Let $U = \{u_1, u_2, \dots, u_n\}$ and $V = \{v_1, v_2, \dots, v_m\}$ denoted the sets of users and items, respectively, where n was the number of users and m was the number of items. We assumed that $R \in \mathbb{R}^{n\times m}$ was the user-item graph and may contain several ordinal rating levels $\{1, \dots, R\}$. Each edge $e = (u_i, v_j, r_{ij}) \in E$ showed that user u_i assigned rating r_{ij} to item v_j. If u_i assigned a rating to v_j, r_{ij} is the rating, otherwise, $r_{ij} = 0$. Generally, we let R_{u_i} be the set of items that have interacted with the user u_i, R_{v_j} be the set of users who v_j have interacted with. We used an embedding vector $s^{u_i} \in R^d$ and an embedding vector $h^{v_j} \in R^d$ to denote a user u_i and an item v_j, respectively, where d was the dimension of the embedding vector. The mathematical notations used in this paper are given in Table 1. Then the task of rating prediction can be formulated as:

Input: The user-item interaction matrix R.

Output: Each user's missing rating for the item.

Table 1. Notations.

Notations	Description
$\mathbf{u_i}$	User i
$\mathbf{u_j}$	Item j
$\mathbf{s^{u_i}}$	The embedding of user u_i.
$\mathbf{h^{v_j}}$	The embedding of item v_j.
$\mathbf{R_{u_i}}$	The set of items with which user u_i interacts.
$\mathbf{R_{v_j}}$	The set of users who have interacted with item v_j.
$\mathbf{e_{u_i}}$	The new aggregated representation of user u_i, obtained by combining the neighborhood of u_i with its own features.
$\mathbf{e_{v_j}}$	The new aggregated representation of item v_j, obtained by combining the neighborhood of v_j and its own features.
$\mathbf{a^{u_i}}$	The preference retention score of user u_i.
$\mathbf{a^{v_j}}$	The preference retention score of item v_j.
$\mathbf{A^{u_i}}$	Preference representation of user u_i.
$\mathbf{A^{v_j}}$	Preference representation of item v_j.
K	The number of GNN layers.
$\mathbf{g_{ij}^k}$	User preference for the k-th layer.
W, b	Weight and bias in the MLP.
$\oplus$	The concatenation operator of two vectors.
$\hat{\mathbf{r}}_{\mathbf{ij}}$	The predicted rating of item v_j by user u_i.
λ	The regularized penalty coefficient.

4. Adaptive preference retention collaborative filtering algorithm based on graph convolutional method

4.1. Overview

In this part, we proposed Adaptive Preference Retention Graph Convolutional Collaborative Filtering Method (APR-GCCF) which was a general graph neural network based on the CF model for the recommendation. The general framework of the APR-GCCF was shown in Figure 2. APR-GCCF consisted of two major components. The first component, an adaptive preference retention mechanism, is shown in the red dotted line and blue dotted line of Figure 2 used the GNN to assemble the user's/item's neighborhood to learn the user/item embedding and took the user of a trainable mapping vector m_1/m_2 that was shared between layers of the GNN generated retain scores $a^0, \ldots, a^k$ to

differentiate among users/items. The second component was a residual preference prediction mechanism as shown in the black dotted line of Figure 2. It utilized a residual network to concatenate user's/item's preferences representations from an adaptive preference retention mechanism. Finally, the user-item interaction ratings were predicted through MLP.

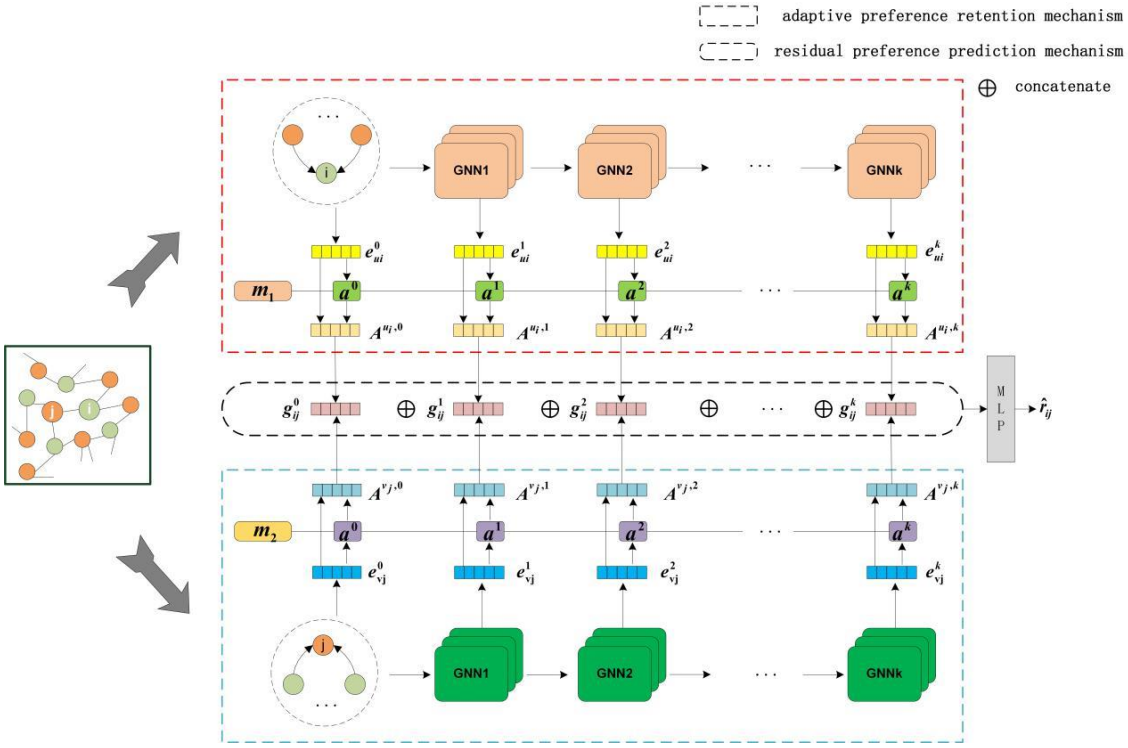


Figure 2. The framework of APR-GCCF.

4.2. Adaptive preference retention mechanism

In this paper, we aimed to learn the fine-grained preference A_{u_i} of user u_i and representation A_{v_j} of item v_j from user-item graph. It was claimed that user u_i's preference relied on his/her characteristics s^{u_i} and neighborhood $h^{a,k}$, where $a \in R_{u_i}$. So, we combined u_i's characteristics with features of u_i's neighborhood through an aggregation function $\text{Aggre}_{\text{user}}$. When aggregating the features of the user to generate item v_j's embedding, a similar aggregation function $\text{Aggre}_{\text{item}}$ was used. The aggregation process as follows:

$$e_{u_i}^{k+1} = \text{Aggre}_{\text{user}}(s^{u_i,k}, \sum_{a \in R_{u_i}} h^{a,k}) \tag{1}$$

$$e_{v_j}^{k+1} = \text{Aggre}_{\text{item}}(h^{v_j,k}, \sum_{b \in R_{v_j}} s^{b,k}) \tag{2}$$

where $e_{u_i}^{k+1}$ denoted the user u_i's embedding of GNN at layer k that is generated by user aggregation function $\text{Aggre}_{\text{user}}$. $k \in (1, \ldots, K)$ indicated the number of layers in GNN. R_{u_i} was the set of items that user u_i has interacted with. R_{v_j} represented the set of users that have interacted with the item v_j, $s^{u_i,k}$ denoted the vector of user u_i at k-th layer. $h^{v_j,k}$ was the vector of item v_j at k-th layer. Then we will discuss how to define aggregation functions $\text{Aggre}_{\text{user}}$ and $\text{Aggre}_{\text{item}}$.

We adopted aggregation functions that is different from the traditional graph convolutional

networks on the user-item graph to capture the user's/item's representations. To be specific, the attention mechanism was applied to aggregate the neighborhood of user/item, as below:

$$e_{u_i}^{k+1} = \mathrm{Relu}(W_{user} \cdot \left(s^{u_i,k} \oplus \sum_{a \in R_{u_i}} \alpha_{i,a} h^{a,k}\right) + b_{user}) \tag{3}$$

$$e_{v_j}^{k+1} = \mathrm{Relu}(W_{item} \cdot \left(h^{v_j,k} \oplus \sum_{b \in R_{v_j}} \alpha_{j,b} s^{b,k}\right) + b_{item}) \tag{4}$$

where $\alpha_{i,a}$ denoted the attention coefficient of items in the user's neighborhood when characterizing user u_i's representation from the user-item interaction history R_{u_i}. $\alpha_{j,b}$ denoted the attention weight of users in the item v_j's neighborhood when characterizing item v_j's representation from the item-user interaction history R_{v_j}. $\mathrm{Relu}(\cdot)$ was the Relu activate function. W_{users} / W_{items} and b_{user}/b_{item} were weight and bias of aggregation functions, respectively. $\oplus$ is the concatenation operation. Take the aggregation function of user u_i as an example, we took user u_i's representation $s^{u_i,k}$ and features of item interacted with user u_i as input, then use the Softmax function to produce the attention coefficient $\alpha_{i,a}$ as:

$$\alpha_{i,a} = \frac{\exp\left(\mathrm{Relu}(W_{i,j} \cdot (s^{u_i,k} \oplus h^{a,k}))\right)}{\sum_{a \in R_{u_i}} \exp\left(\mathrm{Relu}(W_{i,j} \cdot (s^{u_i,k} \oplus h^{a,k}))\right)} \tag{5}$$

$$\alpha_{j,b} = \frac{\exp\left(\mathrm{Relu}(W_{j,i} \cdot (s^{u_i,k} \oplus h^{a,k}))\right)}{\sum_{b \in R_{v_j}} \exp\left(\mathrm{Relu}(W_{j,i} \cdot (s^{u_i,k} \oplus h^{a,k}))\right)} \tag{6}$$

where $W_{i,j}$ and $W_{j,i}$ were the attention weight of $\alpha_{i,a}$ and $\alpha_{j,b}$.

The GNN can capture information on the neighborhood of a given node through the feature propagation mechanism. The nodes were assumed to contribute similarly to the neighborhood aggregation process, which led to the inevitable loss of fine-grained user preference-related information. In practice, the purchasing behavior of the user was usually determined by differences among items. Therefore, different items should make different contributions to generate user embedding. This motivated us to design an adaptive preference retention mechanism to adaptively identify different kinds of information provided by different items.

By taking original user embedding generated by the GNN as input, we took advantage of a trainable mapping vector shared between layers of the GNN to learn the differences among items, and to output their preference retention scores, such as $a^{u_i,k}$ and $a^{v_j,k}$. These preference retention scores were used for representations that carried information on the differences among items in neighborhoods spanning a variety of ranges, and also for measuring the feature-related information of the items to identify their contributions as follows:

$$a^{u_i,k} = \mathrm{Relu}\left(m_1 \cdot e_{u_i}^{k} + n_1\right) \tag{7}$$

$$a^{v_j,k} = \mathrm{Relu}\left(m_2 \cdot e_{v_j}^{k} + n_2\right) \tag{8}$$

where $a^{u_i,k}$ was the preference retention scores of the user u_i at k-th layer, and $a^{u_i,k}$ was the preference retention scores of v_j same as u_i. m_1 and m_2 indicated trainable mapping vectors shared by the processes of user and item preference modeling, respectively, and n_1 and n_2 were biases. $e_{u_i}^{k}$ and $e_{v_j}^{k}$ are user's/item's embeddings generated by GNN at the k-th layer.

Finally, we confirmed differences among the items by using the preference retention scores and preserving the differences in user embeddings automatically to model fine-grained user preferences. The generated user/item preference embeddings $A^{u_i,k}$/$A^{v_j,k}$ were as follows:

$$A^{u_i,k} = a^{u_i,k} \odot e^{k}_{u_i} \tag{9}$$

$$A^{v_j,k} = a^{v_j,k} \odot e^{k}_{v_j} \tag{10}$$

where $\odot$ represented element-wise multiplication, and $A^{u_i,k}$ and $A^{v_j,k}$ were the preference embeddings of the user and the item in layer k.

When users interact with different items, the contributions of different items to the user's representation are different. As shown in Figure 1, the user's current preference is shoes, so shoes are critical to modeling user preferences, and the contribution of musical instruments to the user's representation will be much smaller, therefore, the user's rating of shoes is higher than that of musical instruments. We have added an attention mechanism to the aggregation process of the graph neural networks, namely Eqs (5) and (6), and model the user's representation by assigning corresponding preference scores $\alpha_{i,a}$ and $\alpha_{j,b}$ to different items. At the same time, we regard the user's neighbors of different levels of user interaction as the user's local interest. We believe that the local interest also plays a vital role in determining the final user's preferences, so we adopt a trainable weight pair for different neighborhoods generate corresponding scores, namely Eqs (7) and (8), and assign them to each neighborhood to generate the final user representation.

When we used GNN to aggregate the high-order neighborhoods of user nodes, different neighborhoods would carry different item node information. We modeled the user and item preferences through two trainable mapping vectors m_1 and m_2. We use the adaptively adjusted weight m_1/m_2 shared between GNN layers was used to generate retention scores for different neighborhoods. These retention scores were applied to the user's/item's multi-level neighborhood to save the difference of the item/user nodes in each level of the user's/item's neighborhood. When modeling user preference, we considered the impact of the differences among items for user preference, i.e., the user's purchasing behavior was affected by the attributes of the items. Hence, we utilized m_1 to adaptively identify the differences in items and retain their different features. Similarly, when modeling item preference, we thought that it was different decision-making procedures that determined a user's decision to buy an item. We thus used m_2 to distinguish among these procedures to capture fine-grained item preferences.

With the adaptive preference retention mechanism, APR-GCCF enabled the adaptive identification of differences between items and generated a fine-grained representation of users'/items' preferences.

The preference embeddings for the user and the item have been introduced in the previous part. Next, we will show how the residual preference prediction mechanism was used to generate predictions.

4.3. Residual preference prediction mechanism

With a predefined depth K, the traditional GNN-based recommender systems stop at the k-th layer, where output of the user embedding was regarded as e_i^k and that of the item embedding was e_j^k. In practice, e_i^k and e_j^k can catch the fine-grained preferences of user and item. Their inner product was used to predict the ratings $\hat{r}_{ij}$ as follows:

$$\hat{r}_{ij} = e_i^k \odot e_j^k \tag{11}$$

where $\odot$ denoted element-wise multiplication.

Most GNN-based recommendation models won't achieve the best performance until k = 2 or k = 3 (He et al., 2020; Xiang Wang, He, Wang, et al., 2019; Ying et al., 2018). The performance dropped

quickly when K continued to increase. We think that the reason was that the embedding of each user/item node was smoothed by higher-order neighbors in the user-item graph. Thus, it was the problem of over-smoothing. In the collaborative filtering task, there was few user-item interactions, and consequently rendered the effect of over-smoothing severe. To solve these problems, a residual preference prediction mechanism was proposed. We performed a concatenation operation on user and item preference embeddings $A^{u_i,k}$ and $A^{v_j,k}$ generated by the adaptive preference retention mechanism in layer k as follows:

$$g_{ij}^{k} = A^{u_i,k} \oplus A^{v_j,k} \tag{12}$$

where $\oplus$ was the concatenation operation, k was the layer number of GNN. The residual preference prediction mechanism could help alleviate effect of over-smoothing by using deeper GNN layers to enhance the expressiveness of user preference. Hence, we used residual preference prediction mechanism to connects all layers of the results of prediction:

$$g_{ij} = g_{ij}^{0} \oplus g_{ij}^{1} \oplus \cdots \oplus g_{ij}^{k} \tag{13}$$

where g_{ij} denoted the concatenation of preference-related predictions in all layers, g_{ij}^{0} represented the initially predicted preference used as input for the user and the item, and $\oplus$ denoted the concatenation operation of the embeddings. The previous process was reasonable in that each user's neighbors were different, and thus the practice that integrating each layer's representation to generate the final user preference was advantageous.

The residual preference prediction mechanism connected the user representations generated by the GNN of each layer with different neighborhood information to produce the final unified embedding of the user. At the same time, we obtained the final unified embedding of the item through the same operation, and combined the embedding of the user and the item to generate a prediction.

Equation (9) was equivalent to embedding the nodes with differences in each layer to generate a new embedding. This was quite reasonable because the user-item interaction subgraph was different. As a result, the nodes in its neighborhood were different, and the contribution to the final representation of the node was different as well. Therefore, we integrated the representations of each layer to generate the final user/item embedding which contained richer information.

Once the final preference has been obtained, we used it as input to the MLP to predict the rating r_{ij} from u_i to v_j as:

$$c_1 = \mathrm{Relu}\left(W_1 g_{ij} + b_1\right) \tag{14}$$

$$c_2 = \mathrm{Relu}\left(W_2 c_1 + b_2\right) \tag{15}$$

...

$$\hat{r}_{ij} = W_l c_{l-1} + b_l \tag{16}$$

where W_l was a weight vector, b_l was a bias vector, and l was the hidden layer index. As we can see, the layer-wise aggregation was the main operation. For the k-th GNN layer, the aggregation function has computational complexity $O(nm \times d)$, where n was the number of users and m was the number of items, and d was the dimension of embedding vector.

We used a graph convolution operation in the user-item graph to aggregate the user's representation. By obtaining the user's high-level neighborhood information, we could mine hidden user-item interaction relationships, which can increase the density of the data to a certain extent.

However, aggregating the high-level neighborhood information through multi-layer GNN will cause the problem of over-smoothing. The essence of over-smoothing is that too much information about the other nodes suppress the information of the nodes themselves. A residual network was used to inject the user's previous information into the high-order neighborhoods. Specifically, we connected its embedding and the aggregated new vector together when updating the node embedding. This implemented the decoupling of neighborhoods and their information; thus the problem of over-smoothing could be relieved.

In the following section, some optimization strategies will be discussed.

4.4. Optimization

Objective function: We focused on the task of predicting ratings. The ultimate goal was thus to minimize the difference between the predicted rating and the ground-truth rating:

$$L_r = \frac{1}{2|O|}\sum_{(i,j)\in O}\left(\hat{r}_{ij} - r_{ij}\right)^2 \tag{17}$$

where O was the set of users and items, and $\hat{r}_{ij}$ and r_{ij} were the predicted ratings and the ground-truth ratings.

L0 regularization was used to alleviate the problem of overfitting. The final objective function was:

$$\min_{\Theta} L = L_r + \lambda \parallel \theta \parallel_0 \tag{18}$$

where λ is the L0 regularization parameter and θ represented the set of model parameters.

Weighted random sampling strategy: To model the representation of user/item, the user's/item's neighborhoods were aggregated by applying GNN. However, a large number of interactions occurred between items and users. So, we usually understood user preferences using only some items with what he/she interacted. In Eq (1), we did not aggregate all of the items that interacted with the user because highly-rated items better-reflected user preferences. Hence, we adopted a weighted random sampling strategy to emphasize more on such items.

We regarded the average degree of the node N_i (N_j) of the user (item) as the threshold. When the number of nodes neighboring a given node exceeded the threshold, we chose neighborhoods using a random weighted sampling strategy; otherwise, all neighborhood nodes would be retained. The procedure was as follows:

$$w = \text{Random}\ (0,1) \tag{19}$$

$$s = w^{\frac{1}{r}} \tag{20}$$

where w was a random number in (0, 1), r was the rating, and s indicated a generated weighted sampling score. We sampled all nodes in the neighborhood, arranged them in descending order, and took the highest N_i (N_j) as the neighborhood of the user/item.

5. **The proposed method**

We conducted experiments on three empirically acquired datasets to verify the effectiveness of our proposed model and answered the following questions:

- Q1: How does the performance of our model compare with state-of-the-art collaborative filtering models?

- Q2: How effective is the preference learned from the adaptive preference retention mechanism? Can it be used for residual preference prediction?
- Q3: How do different hyperparameters affect the model?

5.1. Datasets

We used three empirically acquired datasets to validate our model: Amazon, FilmTrust, and Yelp. Their main properties were listed in Table 2. We took 80% of each dataset as the training set and the other 20% as the test set.

- **Amazon:** The Amazon is a widely used product recommendation dataset containing 65,170 ratings from 1000 users on 1000 items.
- **FilmTrust:** FilmTrust is a movie sharing and rating website containing 35,497 ratings from 1508 users on 2071 items.
- **Yelp:** This is a dataset to recommend local businesses containing 30,838 ratings from 1286 users on 2614 items.

5.2. Baseline

To verify the effectiveness of our model, we used the following methods for comparison:

- PMF [22]: This is the most popular matrix factorization model for a CF-based recommender systems.
- BiasedMF [3]: This is a factorization model that considers biases to model user and item preferences.
- SVD++ [23]: It utilizes implicit feedback to capture fine-grained user preferences.
- AutoRce [25]: It utilizes an encoder layer to project the user's/item's observed ratings into a latent space, and then reconstructs them using a decoder layer.
- NGCF [26]: It is a deep learning method that uses the GNN to extract higher-order connectivity for recommendation.
- LR-GCCF [27]: The model uses a residual network to alleviate over-smoothing in the user-item grap.
- MCCF [28]: It decomposes the user's motivation for purchasing, and uses node-level and component-level attention mechanisms to generate user/item preferences.

Table 2. The statistics of the datasets used in this study.

Dataset	#User	#Items	#Interactions	Density	Rating Scale
Amazon	1000	1000	65,170	6.5175%	1−5
FilmTrust	1508	2071	35,497	1.136%	0.5−4
Yelp	1286	2614	30,838	0.917%	1−5

Table 3. Comparison of performance of recommendation algorithms on three empirically acquired datasets with seven baselines. The best performances were presented in bold, and the underlined scores represented the second-best performances.

Data sets	Metrics	PMF	Biased MF	SVD++	I-AutoRec	NGCF	LR-GCCF	MCCF	APR-GCCF	Improve
Amazon	**RMSE**	0.9636	0.9446	0.9430	0.9582	0.9318	0.9467	0.9325	0.8986	3.6%
	MAE	0.7354	0.7257	0.7132	0.7324	0.6944	0.7155	0.6827	0.6664	2.4%
Film Trust	**RMSE**	1.0145	0.9129	0.9177	0.9307	0.9064	0.9151	0.8977	0.8859	1.3%
	MAE	0.7263	0.7010	0.7265	0.7188	0.6989	0.6917	0.7053	0.6809	1.6%
Yelp	**RMSE**	0.3909	0.3636	0.3625	0.3402	0.3767	0.3860	0.3341	0.3253	2.6%
	MAE	0.1881	0.1509	0.1548	0.1422	0.1352	0.1669	0.1973	0.0879	35%

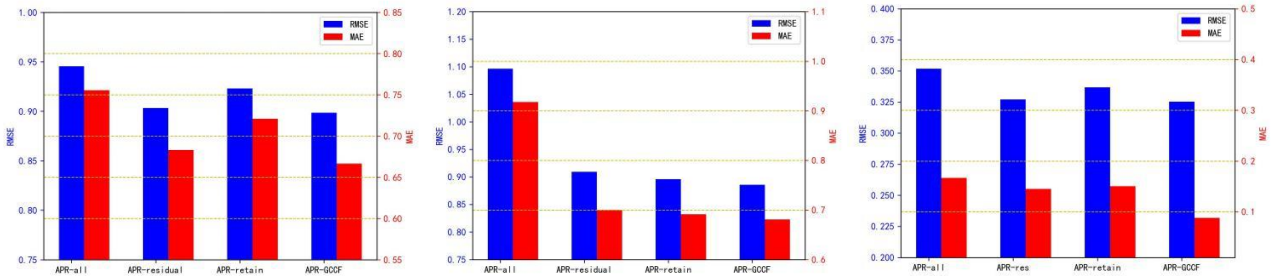
5.3. Implementation

We adopted two widely used evaluation metrics, respectively, the RMSE (root mean squared error) and MAE (mean absolute error) to assess the predicted ratings of items. We set the number of layers of the graph neural network to $\{1, 2, 3, 4, 5\}$ and the number of dimensions of the embedding d to $\{16, 32, 64, 128, 256\}$. We randomly initialized the model parameters using a Gaussian distribution N (0, 0.1) and took Adam as the optimizer. The batch used for training was $\{64, 128, 256, 512\}$ and the learning rate was $\{0.00005, 0.0001, 0.0005, 0.001, 0.01\}$. The dropout rate was $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. We used L0 regularization and the regularization parameters from [39].

5.4. Baseline comparison (Q1)

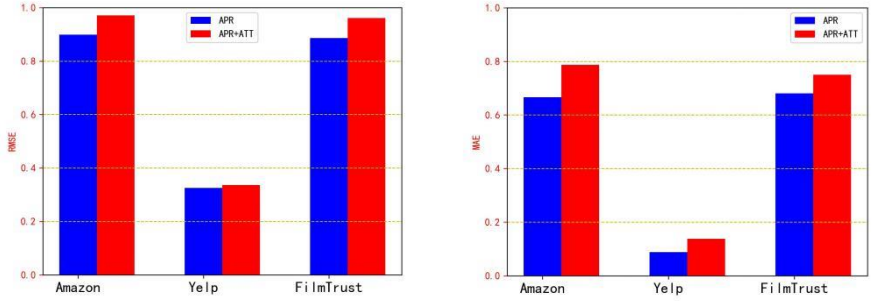
We compared the performance of all methods. Table 3 shows the errors incurred by all of them in the experiments.

We have the following observations: 1) Our model, the APR-GCCF, generally outperformed all the other baselines in terms of the RMSE and MAE, which showed its superiority in the recommendation. 2) The APR-GCCF further improved recommendation performance compared with the NGCF and MCCF. This suggested that considering the differences among items when modeling user preferences helped improve recommendation performance. 3) Our method consistently outperformed the LR-GCCF. Although the LR-GCCF used a residual network to predict user interest, it neglected the varying contributions of items to this outcome. 4) The APR-GCCF achieved better performance than the PMF, BiasedMF, and SVD++. Because these three methods were based on the MF, whereas the APR-GCCF was based on the graph neural networks, and took the user-item graph as input to model the user preference. 5) Of the baselines, the NGCF, LR-GCCF, MCCF, and APR-GCCF belonged to graph neural network-based methods, which indicated that graph neural networks were powerful in terms of learning representations for the user-item graph.



(a) Amazon (b) FilmTrust (c) Yelp

Figure 3. Comparison of performance of variants of the APR-GCCF model.



(a) RMSE (b) MAE

Figure 4. Performance of the APR-GCCF and APR+ATT in terms of the RMSE and MAE.

5.5. Ablation study (Q2)

We designed three variants of the proposed model: APR-all, APR-residual, and APR-retain. APR-residual and APR-retain involved the proposed method without the residual preference mechanism and the adaptive preference retention mechanism, respectively, and APR-all consisted of the proposed method without both. We conducted experiments using them on the three empirically acquired datasets and compared the results with those of our proposed model.

The experimental results were shown in Figure 3.

Figure 3 illustrated that APR-all, APR-residual, and APR-retain delivered worse performance than the proposed method in varying degrees on the three datasets. There were three reasons. First, when we removed the adaptive and the residual preference prediction mechanisms, our model degenerated into a normal GNN that yielded poorer performance in terms of collaborative filtering. Second, the GNN treated the contributions of all items as the same because it did not have the adaptive preference retention mechanism, and thus could not distinguish among the items. Finally, when the residual preference prediction mechanism was removed, the GNN could not handle the problem of over-smoothing, so this degraded its performance.

Effectiveness of adaptive preference retention mechanism: APR-residual delivered worse results than APR-GCCF, which indicated that considering the differences among items is important for learning user preferences and improving recommendation-related performance.

To distinguish the adaptive preference retention mechanism and the attention mechanism, we designed an attention-based variant of the proposed method called APR+ATT, i.e., we used an attention mechanism instead of the adaptive preference retention mechanism. We experimented it on three datasets and compared its results with those of the proposed APR-GCCF, as shown in Figure 4.

The APR+ATT was inferior to our model on both evaluation metrics on the three datasets. This is because the attention mechanism usually pays more attention to items with larger weights, and cannot effectively retain the difference information between items. Besides, using the attention mechanism to model user representations, items with higher weights were usually assigned more contributions to the user representation, while items with lower weights may be ignored. So, this cannot reflect the difference information between the items. It did not consider the differences among items, which was an important reason for the inefficiency of APR+ATT.

Effectiveness of residual preference prediction mechanism: Figure 3 showed that the APR-all delivered worse than APR-retain. This verifies the effectiveness of the residual preference prediction mechanism. We asserted that the traditional GNN suffered from data sparsity and over-smoothing during propagation in higher-order neighborhoods. We integrated the aggregated results of each layer of the GNN into the final prediction by using a residual network, and this helped avoid data sparsity and over-smoothing, thus improving performance in terms of recommendations.

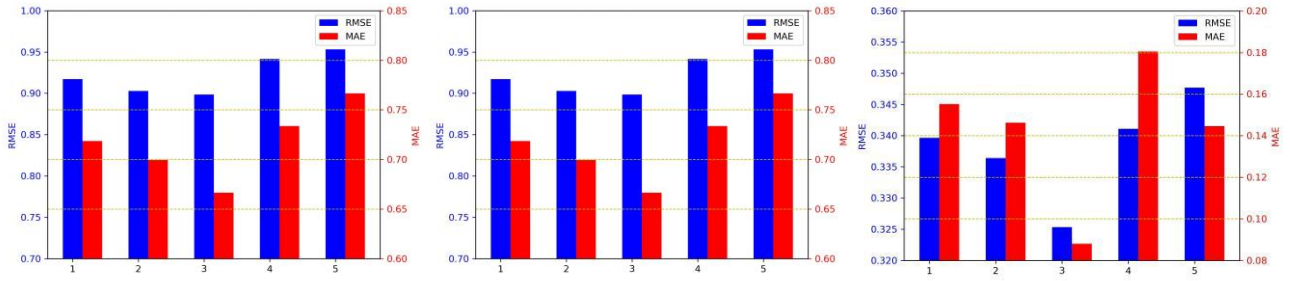
5.6. *Effect of hyperparameters (Q3)*

Since the number of GNN layers is critical to the model, we investigated its impact on performance. We also analyzed the effects of the learning rate and the number of embedding dimensions on the APR-GCCF.

The effect of the number of GNN layers: In the graph convolution clustering process, the number of layers of the graph convolution is 1, which means that the first-order neighborhoods are aggregated. The number of layers is k, which means that the kth-order neighborhoods are aggregated. To explore the effect of the number of layers on the model, we varied the number of GNN layers k in the range $\{1, 2, 3, 4, 5\}$ while keeping the other parameters constant. Figure 5 implied the performance obtained using different numbers of graph convolutional layers on the three datasets. When k = 3, the model achieved the best performance on all three datasets. As the number of graph convolutional layers increased, performance gradually worsened due to over-smoothing and overfitting.

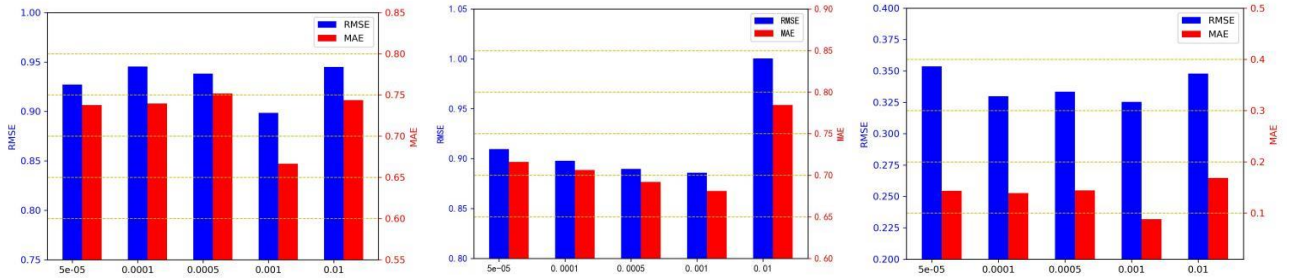
The effect of the learning rate: We delved into the effects of different learning rates on the model on the three datasets, as shown in Figure 6. We set the initial learning rate lr to 0.01 and gradually reduced it. The performance of the model was optimal when lr = 0.001, and degraded as the learning rate decreased. We thus used an appropriate learning rate to reduce the complexity of the model.

The impact of the number of embedded dimensions: The number of embedded dimensions d was a key parameter controlling the APR-GCCF. Figure 7 showed that the proposed method attained the best performance on Amazon and FilmTrust at $d = 64$, and on Yelp at $d = 32$. This difference might have been acquired owing to the varying sparsity of the datasets. In general, as d increased, recommendation performance gradually improves, with more embedded dimensions yielding a stronger representation. But the model's performance worsened when the number of dimensions was larger than the optimal value.



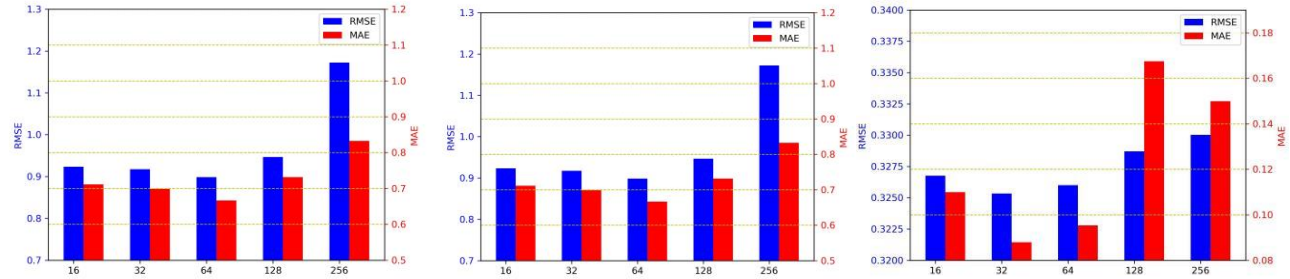
(a) Amazon (b) FilmTrust (c) Yelp

Figure 5. The effect of the number of layers of the GCN on the three datasets.



(a) Amazon (b) FilmTrust (c) Yelp

Figure 6. The effect of learning rate on the three datasets.



(a) Amazon (b) FilmTrust (c) Yelp

Figure 7. The effects of embedded dimensions on the three datasets.

6. Conclusions and future work

We proposed a graph network model called the APR-GCCF to model collaborative filtering to predict the ratings assigned by users to items to make recommendations to them. The APR-GCCF was composed of two parts. An adaptive preference retention mechanism was used to adaptively distinguish the items. We retained differences among these items to model fine-grained user/item preferences. In addition, to reduce data sparsity and the effect of over-smoothing introduced by higher-order layers of the GNN, we designed a residual preference prediction mechanism that used a residual

network to concatenate the user/item preference generated by each layer of the GNN. We conducted extensive experiments on three real-world datasets to validate the effectiveness of our approach.

Although the APR-GCCF has achieved success, information is collected from adjacent nodes without distinguishing which path the information comes from. However, this problem is necessary for improved model capability and interpretability. In future work, we will investigate this problem.

Acknowledgments

This study was supported by the Jilin Provincial Development and Reform Commission (contract number 2022C046-5) and the Scientific Research Project of The Education Department of Jilin Province in 2023, NO. JJKH20231514K.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. S. Milano, M. Taddeo, L. Floridi, Ethical aspects of multi-stakeholder recommendation systems, *Inf. Soc.*, **37** (2021), 35−45. https://doi.org/10.1080/01972243.2020.1832636
2. H. Tang, G. Zhao, X. Bu, X. Qian, Dynamic evolution of multi-graph based collaborative filtering for recommendation systems, *Knowledge-Based Syst.*, **228** (2021), 107251. https://doi.org/10.1016/j.knosys.2021.107251
3. Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, *Computer*, **42** (2009), 30−37. https://doi.org/10.1109/MC.2009.263
4. G. Datta, P. A. Beerel, Can deep neural networks be converted to ultra low-latency spiking neural networks, in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021. https://doi.org/10.23919/DATE54114.2022.9774704
5. C. Gao, X. Wang, X. He, Y. Li, Graph neural networks for recommender system, in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, (2022), 1623−1625. https://doi.org/10.1145/3488560.3501396
6. S. Jang, H. Lee, S. Cho, S. Woo, S. Lee, Ghost graph convolutional network for skeleton-based action recognition, in *2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, 2021. https://doi.org/10.1109/ICCE-Asia53811.2021.9641919
7. J. Xu, L. Chen, M. Lv, C. Zhan, S. Chen, J. Chang, HighAir: a hierarchical graph neural network-based air quality forecasting method, preprint, arXiv:2101.04264. 86.
8. V. Kalofolias, X. Bresson, M. Bronstein, P. Vandergheynst, Matrix completion on graphs, preprint, arXiv:14081717.
9. X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T. Chua, Neural collaborative filtering, in *Proceedings of the 26th International Conference on World Wide Web*, (2017), 173−182. https://doi.org/10.1145/3038912.3052569
10. X. Wei, J. Liu, Effects of nonlinear functions on knowledge graph convolutional networks for recommender systems with yelp knowledge graph, Lamar University, Beaumont, (2021), 185−199. https://doi.org/10.5121/csit.2021.110715

11. W. Li, L. Ni, J. Wang, C. Wang, Collaborative representation learning for nodes and relations via heterogeneous graph neural network, *Knowledge-Based Syst.*, **255** (2022), 109673. https://doi.org/10.1016/j.knosys.2022.109673
12. C. Huang, Recent advances in heterogeneous relation learning for recommendation, preprint, arXiv:211003455.
13. H. B. Kang, R. Kocielnik, A. Head, J. Yang, M. Latzke, A. Kittur, et al., From who you know to what you read: augmenting scientific recommendations with implicit social networks, in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, (2022), 1−23. https://doi.org/10.1145/3491102.3517470
14. T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, preprint, arXiv:1609.02907.
15. X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, Lightgcn: simplifying and powering graph convolution network for recommendation, in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, (2020), 639−648. https://doi.org/10.1145/3397271.3401063
16. B. Jin, C. Gao, X. He, D. Lin, Y. Li, Multi-behavior recommendation with graph convolutional networks, in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, (2020), 659−668. https://doi.org/10.1145/3397271.3401072
17. L. Ma, Y. Li, J. Li, W. Tan, Y. Yu, M. A. Chapman, Multi-scale point-wise convolutional neural networks for 3D object segmentation from LiDAR point clouds in large-scale environments, *IEEE Trans. Intell. Transp. Syst.*, **22** (2019), 821−836. https://doi.org/10.1109/TITS.2019.2961060
18. X. Wang, H. Jin, A. Zhang, X. He, T. Xu, T. Chua, Disentangled graph collaborative filtering, in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, (2020), 1001−1010. https://doi.org/10.1145/3397271.3401137
19. Y. Zheng, C. Gao, L. Chen, D. Jin, Y. Li, DGCN: diversified recommendation with graph convolutional networks, in *Proceedings of the Web Conference*, (2021), 401−412. https://doi.org/10.1145/3442381.3449835
20. R. Raziperchikolaei, Y. J. Chung, Simultaneous learning of the inputs and parameters in neural collaborative filtering, preprint, arXiv:2203.07463.
21. R. Yin, K. Li, G. Zhang, J. Lu, A deeper graph neural network for recommender systems. *Knowledge-Based Syst.*, **185** (2019), 105020. https://doi.org/10.1016/j.knosys.2019.105020
22. R. R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in *Proceedings of the 20th International Conference on Neural Information Processing Systems*, (2007), 1257−1264.
23. Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2008), 426−434. https://doi.org/10.1145/1401890.1401944
24. J. Wei, J. He, K. Chen, Y. Zhou, Z. Tang, Collaborative filtering and deep learning based recommendation system for cold start items, *Expert Syst. Appl.*, **69** (2017), 29−39. https://doi.org/10.1016/j.eswa.2016.09.040
25. S. Sedhain, A. K. Menon, S. Sanner, L. Xie, Autorec: autoencoders meet collaborative filtering, in *Proceedings of the 24th International Conference on World Wide Web*, (2015), 111−112. https://doi.org/10.1145/2740908.2742726

26. X. Wang, X. He, M. Wang, F. Feng, T. Chua, Neural graph collaborative filtering, in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, (2019), 165−174. https://doi.org/10.1145/3331184.3331267
27. L. Chen, L. Wu, R. Hong, K. Zhang, M. Wang, Revisiting graph based collaborative filtering: a linear residual graph convolutional network approach, in *Proceedings of the AAAI Conference on Artificial Intelligence*, **34** (2020), 27−34. https://doi.org/10.1609/aaai.v34i01.5330
28. X. Wang, R. Wang, C. Shi, G. Song, Q. Li, Multi-component graph convolutional collaborative filtering, in *Proceedings of the AAAI Conference on Artificial Intelligence*, **34** (2020), 6267−6274. https://doi.org/10.1609/aaai.v34i04.6094
29. C. Zhang, W. Li, D. Wei, Y. Liu, Z. Li, Network dynamic GCN influence maximization algorithm with leader fake labeling mechanism, *IEEE Trans. Comput. Social Syst.*, **2022** (2022), 1−9. https://doi.org/10.1109/TCSS.2022.3193583
30. F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, *IEEE Trans. Neural Networks*, **20** (2008), 61−80. https://doi.org/10.1109/TNN.2008.2005605
31. M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in *2005 IEEE International Joint Conference on Neural Networks*, (2005), 729−734. https://doi.org/10.1109/IJCNN.2005.1555942
32. W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, (2017), 1025−1035.
33. P. Velikovi, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, preprint, arXiv:1710.10903.
34. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al., Attention is all you need, in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017. Available from: https://papers.nips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
35. K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks, preprint, arXiv:181000826.
36. X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, et al., Heterogeneous graph attention network, in *The World Wide Web Conference*, (2019), 2022−2032. https://doi.org/10.1145/3308558.3313562
37. F. Wu, T. Zhang, A. H. de Souza, C. Fifty, T. Yu, K. Q. Weinberger, Simplifying graph convolutional networks, **2019** (2019), 6861−6871. https://doi.org/10.48550/arXiv.1902.07153
38. M. Liu, H. Gao, S. Ji, Towards deeper graph neural networks, in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, (2020), 338−348. https://doi.org/10.1145/3394486.3403076
39. C. Louizos, M. Welling, D. P. Kingma, Learning sparse neural networks through L_0 regularization, preprint, arXiv:171201312.