Berger Scientfic Press

Innovation & Technology Advances, 2024, 2(1), 14-26
https://bergersci.com/index.php/jta/index
DOI: 10.61187/ita.v2i1.30

# GCN-MF: A graph convolutional network based on matrix factorization for recommendation

**Junxi Yang[1], Zongshui Wang[2,*], Chong Chen[1]**

[1]School of Computing, Beijing Information Science and Technology University, Beijing 100096, China
[2]School of Economics and Management, Beijing Information Science and Technology University, Beijing 100096, China

*Correspondence: wangzongshui8@163.com

**Abstract:** With the increasing development of information technology and the rise of big data, the Internet has entered the era of information overload. While users enjoy the convenience brought by big data to their daily lives, they also face more and more information filtering and selection problems. In this context, recommendation systems have emerged, and existing recommendation systems cannot effectively deal with the problem of data sparsity. Therefore, this paper proposes a graph convolutional network based on matrix factorization for recommendation. The embedding layer uses matrix factorization instead of neighborhood aggregation, and the interaction layer uses multi-layer neural networks instead of simple inner products. Finally, on the Movielens-1M, Yelp and Gowalla public data set, NDCG and Recall are better than the existing baseline model, which effectively alleviates the data sparsity problem.

**Keywords:** Recommendation Systems; Graph Convolutional Network; Deep Learning; Matrix Factorization

**How to cite this paper:** Yang, J., Wang, Z., Chen, C. GCN-MF: A graph convolutional network based on matrix factorization for recommendation. Innovation & Technology Advances, 2024, 2(1), 14-26. https://doi.org/10.61187/ita.v2i1.30



## 1. Introduction

Artificial intelligence enables computers to perform various tasks and solve problems by simulating human intelligence. The development of artificial intelligence technology provides strong algorithm support and data processing capabilities for recommendation systems[1]. Recommendation systems are research directions formed by integrating multiple disciplines such as data mining, machine learning, and deep learning. It models and analyzes existing data to predict new users or items. Its purpose is to mine user potential information and establish appropriate matching relationships between users and items. Its mathematical expression is the items set I, the users set U, and the function F represents the accuracy of recommending i to u, as shown in (1).

$$\forall u \in U, r = \operatorname{argmax}(s(u, i), i \in I) \tag{1}$$

At present, recommendation systems are widely used in the field of e-commerce, on the one hand, the recommendation systems use the interaction information between users and items for modeling[2], to recommend the content that the user needs to see the most, effectively solving the problem of excessive information in the Internet leading to user loss and difficulty in finding the target information, on the other hand, the recommendation systems can effectively achieve the goal of the e-commerce platform[3], and the personalized recommendation service for users can attract more users, increase user stickiness, and improve user retention. It can be seen that the research on recommendation systems meets the needs of current Internet platforms and users.

The current recommendation system is closely integrated with deep learning. Firstly, models based on deep learning have stronger expressive ability, and multi-layer neural

networks can mine more feature information. Secondly, deep learning has strong flexibility and can achieve the integration of models and applications. The single hidden layer neural network recommendation model combines the ideas of collaborative filtering and autoencoder[4]. In collaborative filtering, the self-encoding of user vectors and item vectors is completed through co-occurrence matrix, and the user item scores are predicted based on the self-encoding results. The model has certain generalization and expressiveness, but the structure is too simple and the expression ability is insufficient. The Deep Crossing model proposed by Microsoft[5] embeds the original features and inputs them into the neural network layer, where the features are intersected and handed over to the model for completion. This can solve the problem of large-scale automatic feature combinations, but the model can only cross the feature vectors of users and projects. The Product-Based Neural Network (PNN)[6] proposed by Shanghai Tongji University can achieve inner product, outer product, and other operations between different feature domains, improving the problem of insufficient cross over of deep crossing features. However, the approximation operation of outer product affects the expression ability. The Wide & Deep model[7] released by Google is a mixture of single-layer Wide and multi-layer Deep, enhancing the model's memory and generalization ability. Although this model can quickly process and memorize historical behavioral information, manual operations in the Wide section increase overhead. With the rapid development of deep learning, recommendation algorithms based on deep neural networks (MLP), convolutional neural networks (GNN), and recurrent neural networks (RNN) have emerged endlessly [8]. The core of these deep learning models in the field of recommendation systems is to learn the hidden vectors of users and projects, and combine them with collaborative filtering to complete parameter training. Finally, the hidden vectors of users and projects are extracted, and the recommendation process is completed based on predicted scores.

However, the data extracted from recommendation systems has an irregular spatial structure and a large scale, and traditional neural network methods still cannot efficiently process such data. Graph Neural Network (GNN) is a redefined and designed deep learning model for processing non-Euclidean spatial data[9], which can accurately capture the potential connections between such data, model efficiently, and provide users with more accurate and personalized recommendation services. The successful applications of Pinterest Sample and Aggregate (PinSAGE)[10] and Multi-task Multi-view Graph Representation Learning framework (M2GRL)[11] in industrial recommendation systems fully demonstrate the practicality of recommendation systems based on GCN. In recent work, some scholars have fused the recommendation of graph neural networks with other concepts[12] or models[13] to further mine features between users and items and improve recommendation performance. This is an important research direction for improving graph neural networks.

To sum up, GNN plays an important role in recommendation systems, including the following aspects: 1) Modeling complex relationships: Recommendation systems face a complex network of relationships between users and items. GNN can effectively model these complex relationships, capturing implicit associations between users and items by learning the representations of nodes and edges. This enables the recommendation system to have a more comprehensive and accurate understanding of users' interests and similarities between items. 2) Dealing with cold start issues: In recommendation systems, cold start issues for new users and items are a challenge. GNN can alleviate the cold start problem by utilizing existing user-item relationship networks to learn embedded representations of nodes and edges. By embedding new entities into existing graph structures, graph neural networks can provide preliminary recommendations for new entities based on similarity and association. 3) Remote relationship modeling: The relationship between users and items in a recommendation system is not limited to direct interaction, but also includes implicit and indirect association relationships. Graph neural networks can model

remote relationships, identify hidden associations and interests, and provide more comprehensive and accurate recommendations through multi-hop information dissemination and aggregation.

The remainder of this paper is organized as follows. Section 2 introduces the related work including matrix factorization and graph convolutional networks. Section 3 specifies the model designed in this paper. Section 4 presents the experimental process and results. Section 5 is the conclusion, providing an overview of the work.

## 2. Related Work

### *2.1. Matrix Factorization*

Matrix Factorization[14] is the multiplication of a matrix into several matrices, which can describe the hidden features of the original matrix. Singular value decomposition is a commonly used matrix factorization method, such as (2) matrix M is decomposed into three matrices.

$$\mathrm{M} = Udiag(\partial)V^T \approx U_k diag(\partial)_K V_K^T \tag{2}$$

Where $U$and $V^T$are the unitary matrices, representing left and right singular vectors. $diag(\partial)$ is a diagonal matrix, and the elements on the diagonal are singular values. Singular values are generally arranged in descending order.

Since values with larger (smaller) singular values have a larger (smaller) representation of matrix features, K maximum singular values and singular vectors can be used to approximate the original matrix M.

Matrix factorization has the characteristics of interpretability and dimensionality reduction, so it can be combined with machine learning and deep learning models and applied in various fields[15,16]. In computer vision, image classification uses matrix factorization to extract image features, which can more accurately identify different objects in the image. Matrix factorization in image processing can realize image compression and storage, and improve processing efficiency.

In natural language processing, the matrix factorization in text processing can decompose a large amount of text data into text features and user features, thus realizing text classification, clustering, visualization and emotional analysis. In addition, the matrix factorization can also learn the relationship between words and words, and then mine new words. In a recommendation system, the PMF model based on matrix factorization[17] decomposes the user-item rating matrix into two low-dimensional matrices, optimizes the objective function, and finally predicts the missing values in the original rating matrix to generate a recommendation list. Overall, in the era of big data, using matrix factorization to discover hidden patterns and features in data has become a crucial data processing method.

### *2.2. Graph Convolutional Networks*

In Graph Convolutional Networks (GCN), the interaction data is represented as a bipartite graph, that is, users (items) can interact with items (users), but users (items) do not interact with each other. As shown in **Figure 1**.

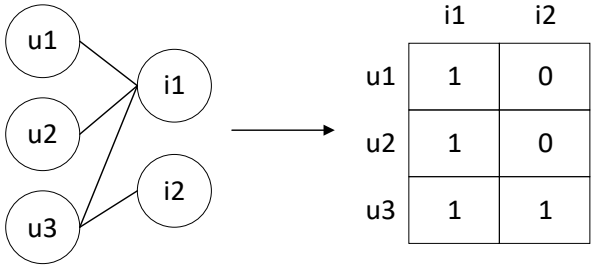


**Figure 1.** Bipartite Graph and Interaction Matrix

The bipartite graph is composed of three users and three items, one user can interact with multiple items, and one item can interact with multiple users. Interaction information can be stored in the interaction matrix of 3 times 2.

Suppose there are user set U of m users and item set I, interaction matrix, and adjacency matrix A of n items defined as (3).

$$\mathrm{A} = \begin{bmatrix} 0 & R \\ R^T & 0 \end{bmatrix} \tag{3}$$

Multi-layer graph convolution network[18] based on hierarchical propagation rules is a classic neighborhood information aggregation neural network model, which is used to solve the problem of graph node classification. Its definition is as follows (4).

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{4}$$

Where A is the adjacency matrix, D is the degree matrix of A, $H^{(l)}$is the input of the L-layer of the graph neural network, $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ is normalization processing, $H^{(l+1)}$is the output of the L-layer, $W^{(l)}$ is the trainable parameter matrix of the L-layer, and $\sigma$ is the activation function.

Neural Graph Collaborative Filtering (NGCF)[19] follows the idea of standard graph convolutional network and collaborative filtering, integrates the bipartite graph of user and item into the embedding, solves the problem of lack of coding in the embedding of collaboration signals, and effectively expresses the higher-order connectivity between users and items. In the initial step, each user and item are associated with an ID embedding, $e_u^{(0)}$ is the initial embedding of user u, and $e_i^{(0)}$ is the initial embedding of item i. Then NGCF uses the interaction of user and item to express the embedding as (5).

Where $e_i^{(k)}$ and $e_u^{(k)}$ are the embeddings of user u and item i after propagation through k-layer, LeakyReLU is the activation function,$N_u$ is the item set where user u interacts with item i,$N_i$ is the user set where item i interacts with user u, $W_1$ and $W_2$ are trainable parameter matrices to achieve feature transformation, $\frac{1}{\sqrt{|N_u||N_i|}}$ is normalization processing.

$$\begin{aligned} e_u^{(k+1)} &= LeakyReLU(W_1 e_u^{(k)} + \sum_{i\in N_u} \frac{1}{\sqrt{|N_u||N_i|}}(W_1 e_i^{(k)} + W_2(e_i^{(k)} \odot e_u^{(k)}))) \\ e_i^{(k+1)} &= LeakyReLU(W_1 e_i^{(k)} + \sum_{u\in N_i} \frac{1}{\sqrt{|N_u||N_i|}}(W_1 e_u^{(k)} + W_2(e_u^{(k)} \odot e_i^{(k)}))) \end{aligned} \tag{5}$$

After k-layer propagation, user u and item i each obtain k+1 embeddings. The embeddings are concatenated to obtain the final user and item embedding. Finally, the inner product is used to predict user preference scores for the item.

Although GCN has made some achievements in the field of graph learning, the proposals of Refined Graph Convolution Collaborative Filtering (RGCF)[20] and Linear Residual Graph Convolutional Collaborative Filtering (LR-GCCF)[21] indicate that appropriate simplification of GCN can improve the performance of recommendation tasks. He et al proved through a large number of ablation experiments that the nonlinear activation function and feature transformation matrix of NGCF are redundant operations for collaborative filtering and proposed a recommendation model named Light Graph Convolutional network (LightGCN)[22] that only retains neighborhood aggregation, aiming to reduce the complexity of the model while improving the speed of model training. The embedding updates for each user and item are shown in (6).

$$e_u^{(k+1)} \sum_{i\in N_u} \frac{1}{\sqrt{|N_u||N_i|}} e_i^{(k)} \tag{6}$$

Each layer of embedding is only related to the embedding of the neighborhood, simplifying the model while improving computational efficiency. Compared with NGCF, LightGCN ignores the self-connection of nodes, deletes the activation function and feature transformation matrix, and uses average pooling to obtain the same effect of self-connection, further simplifying the operation process. In the LightGCN model, only the embedding of layer 0 is trained, and multiple layers of embedding are continuously obtained through the formula (6). Finally, the initial and output of each layer are combined to obtain the final embedding as shown in (7).

$$E^{fn} = \left( \sum_{l=0}^{L} \frac{(D^{-\frac{1}{2}} A D^{-\frac{1}{2}})^l}{L+1} \right) E \tag{7}$$

### *2.3. Summary*

In recommendation systems, matrix decomposition can extract hidden features from a large amount of user project interaction data, capture the correlation between users and items, and achieve more accurate recommendations. In graph convolutional networks, LightGCN is a simplified graph convolutional network model based on the NGCF. After experimental verification, LightGCN is superior to the NGCF model in many cases. However, LightGCN still has some drawbacks such as the complexity and waste of time in graph convolutional neighborhood aggregation operations in recommendation tasks, and the simple inner product method restricts user and item interaction. To solve these problems, this paper integrates matrix factorization into a graph convolutional network and proposes an effective collaborative filtering model.

## 3. Model

To solve the problems in the LightGCN model, the Graph Convolutional Network model based on Matrix Factorization (GCN-MF) proposed in this paper is shown in **Figure 2**.

The model in the Fig. consists of four parts: input layer, embedding layer, interaction layer, and output layer. Firstly, input the interaction data from users and items. The embedding layer uses matrix factorization to decompose the interaction data into three matrices M, V, and N. These matrices only take the K largest singular vectors or singular values. M, V, and parameter matrices form the user embedding, while N, V, and parameter matrices form the project embedding. The interaction layer uses multi-layer neural networks to learn the interaction between user embedding and item embedding and finally outputs the prediction results.

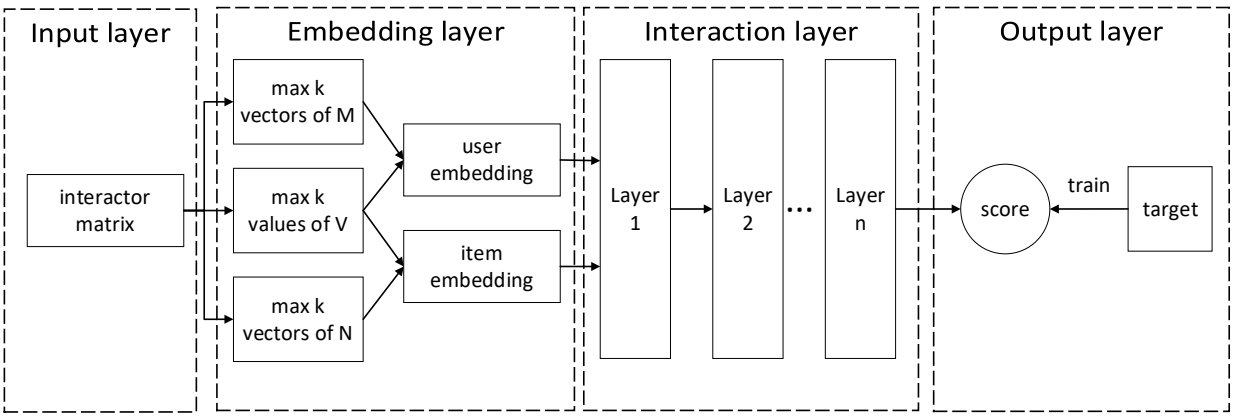


**Figure 2.** GCN-MF3.1. Input Layer and Embedding Layer.

The input layer inputs user item interaction data. Based on matrix factorization, formula (7) representing LightGCN can be split into two terms to obtain formula (8).

$$E_U^{fn} M\ diag\left(\frac{\sum_{l=\{0,2,4\ldots\}} \partial_k^l}{L+1}\right) M^T E_U = M\ diag\left(\frac{\sum_{l=\{1,3,5\ldots\}} \partial_k^l}{L+1}\right) N^T E_I$$
$$E_I^{fn} N\ diag\left(\frac{\sum_{l=\{0,2,4\ldots\}} \partial_k^l}{L+1}\right) N^T E_I + N\ diag\left(\frac{\sum_{l=\{1,3,5\ldots\}} \partial_k^l}{L+1}\right) M^T E_U \quad (8)$$

Where $E_U$ and $E_I$ represent the embeddings of users and items, $E_U^{fianl}$ and $E_I^{fianl}$ represent the final embeddings, $M$, $N$represent the singular vectors of $\tilde{R}$, and $\partial_k^l$ represent the singular values of $\tilde{R}$.

According to (8), the final embedding representation is only determined by the singular vector and singular value, rather than neighborhood aggregation. Therefore, in (8), this paper retains M, N, and diagonal matrices, and replaces $M^T E_U$ and $N^T E_I$ with the learnable weight matrix W to obtain the formula, as shown in (9).

$$E_U^{fn} = M\ diag\left(\frac{\sum_{l=0}^{L} \partial_k^l}{L+1}\right) W_1$$
$$E_I^{fn} = N\ diag\left(\frac{\sum_{l=0}^{L} \partial_k^l}{L+1}\right) W_2 \quad (9)$$

Firstly, a weighted singular matrix is obtained by assigning singular value weights to the singular vector, and then the embedding of the singular vector is learned through the feature transformation matrix W, the weight of singular vectors can be adjusted through L.

In recent years, low rank representations[23] have played an increasingly important role in deep learning. The goal of formula (9) is to learn low rank representations. As the number of convolutional layers L increases, graph neural networks pay more attention to larger singular values to learn low rank representations and ignore smaller singular values. When $L \to \infty$, the weight of the maximum singular value vector approaches 1 while the weight of other singular value vectors approaches 0. Therefore, excessive model layers can lead to the loss of some important information and the problem of over-smoothing[24]. The noise present in most small singular values and singular vectors can have a negative impact on the model, increasing computational complexity while reducing recommendation efficiency. Therefore, according to the idea of approximate substitution in matrix factorization, larger singular values and singular vectors are more important in representing matrix features. Only the first K largest singular values and singular vectors can approximate the original matrix.

LightGCN uses a simple polynomial to represent the weight function. This paper uses $f(\tilde{\partial}_k)$ to represent the weight function to obtain the GCN-MF embedding layer model. The weight function can be set to different functions, for example, set $f(\tilde{\partial}_k)$ to $\frac{1}{1-\gamma\partial_k}$ [25]. Finally, the formula is shown in (10).

$$E_U^{fn} = \widetilde{M}^{(K)}\ diag\left(f(\tilde{\partial}_k)\right) W_1$$
$$E_I^{fn} = \widetilde{N}^{(K)}\ diag\left(f(\tilde{\partial}_k)\right) W_2 \quad (10)$$

Where $\gamma$ is a parameter used to adjust the importance of singular values, $\widetilde{M}^{(K)}$ and $\widetilde{N}^{(K)}$ are composed of the largest K left and right singular vectors by $\tilde{R}$, and $\tilde{R}$ is the normalized representation of the interaction matrix R.

Unlike traditional graph convolutional networks that update the embeddings of all users and items multiple times in the form of parameter matrices, the GCN-MF embedding layer uses K maximum singular values and singular vectors instead of neighborhood aggregation to learn the embeddings of users and items, reducing model parameters and complexity, and having stronger scalability.

### *3.2. Interaction Layer and Output Layer*

After the embedding layer, the vector representation of users and items is obtained, and then input into the interaction layer. The interaction layer and output layer use Neural Collaborative Filtering (NCF)[26], and the simple inner product form of the original model is replaced by the Multi-Layer Perceptron (MLP). The reason is that multi-layer neural network is a structure that can approach any continuous function through training and parameter adjustment[27], which can solve the limitation of simple inner product, to better capture the complex interaction between users and items, and improve the accuracy of model prediction. The formula of the interaction layer is (11).

$$\propto = a_l(h_l^T(a_{l-1}(\dots a_2\left(h_2^T\begin{bmatrix}E_U^{fn}\\E_I^{fn}\end{bmatrix}+b_2\right)\dots))+b_l) \tag{11}$$

Where $h_l^T$ represents the parameter matrix of each layer, and $b_l$ is the bias of each layer, and $a_l$ represents the activation function of each layer of the neural network, and $\propto$ is the output of the interaction layer.

The output layer only has one neuron connected to $\propto$, and the final prediction result is the scoring matrix. The Formula is (12).

$$\hat{y}_{ui} = a(h^T \propto) \tag{12}$$

Where $h^T$ is the weight of the output layer and $a$ is the activation function of the output layer.

The simple inner product of MF is a special form of NCF. Set the number of neural network layers of the interaction layer in the NCF to 0, set the parameter matrix of the output layer to identity matrix, and select the identity function as the activation function to become the inner product form of MF. When selecting a multi-layer neural network and nonlinear activation function, compared with the simple inner product form of MF, GCN-MF's interaction layer and output layer have a more powerful expression ability to learn the interaction between users and items.

## 4. Experiments

### *4.1. Experimental Environment*

Experimental environment setup: The programming language is Python, the language version is Python 3.8.10, and the operating system used is window10. The code is developed by the deep learning framework Python 1.13.1. The server used in the laboratory is NVIDIA RTX6000 GPU with 48GB of memory.

### *4.2. Data Set and Evaluation Indicators*

The data set is from Movielens-1M, Yelp and Gowalla. Including user, item and interaction information. The information of the data set is shown in **Table 1**.

**Table 1.** Data set.

| | **Movielens-1M** | **Yelp** | **Gowalla** |
|---|---|---|---|
| User | 6040 | 31,668 | 29,858 |
| Item | 3952 | 38,048 | 40,981 |
| Interaction | 1,000,209 | 1,561,406 | 1,027,370 |
| Sparsity | 95.81% | 99.87% | 99.916% |

Movielens-1M contains 1,000,209 interactions between 6040 users and 3952 items, with a sparsity of 95.81%. Yelp contains 1,561,406 interactions between 31,668 users and 38,048 items, with a sparsity of 99.87%. Gowalla contains 1,027,370 interactions between 29,858 users and 10,981 items, with a sparsity of 99.916%. In this experiment, 80% of the

data was randomly selected as the training set, and the remaining 20% was used as the testing set.

The most commonly used evaluation indicator in recommendation systems are Recall@k and NDCG@k[28].

The Recall is the ratio of the predicted number of positive samples to the total number of positive samples. A higher Recall indicates a more accurate model prediction, as shown in (13).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{13}$$

Where TP represents the predicted number of positive samples and FN represents the number of positive samples that were not predicted.

The NDCG is an indicator that measures the quality of sorting results. Firstly, the scores of each position in the sorting results are normalized, and then the cumulative scores of each position are calculated in order of position. Finally, the results are obtained by dividing the cumulative scores by the theoretical scores. The higher the NDCG, the more accurate the model prediction is, as shown in (14).

$$\text{NDCG} = \frac{\sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)}}{\sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{log_2(i+1)}} \tag{14}$$

Where $|REL|$ represents a set composed of the first P results, arranged in descending order of correlation, $rel_i$ represents the correlation degree at position i.

### *4.3. Comparing Algorithms and Parameter Settings*

The comparative algorithms used in this experiment include Bayesian Personalized Ranking optimization for Matrix Factorization (BPR-MF), NGCF, LightGCN, and Ultra Graph Convolutional network (UltraGCN). The application fields and advantages of the four methods are as follows.

BPR-MF[29], Bayesian Personalized Ranking (BPR) is a general personalized sorting optimization method. BPR-MF directly trains users to embed projects and then recommends them according to the collaborative filtering framework. The goal is to get the correct sorting of items by users.

NGCF[19], a neural network model that uses multi-layer graph convolution to learn user and project embedding, updates user and item embeddings through neighborhood aggregation, feature transformation and nonlinear activation of graph convolution network, and then recommends according to collaborative filtering framework.

LightGCN[22], a neural network model that uses multi-layer graph convolution to learn user and item embeddings, is a simplified version of NGCF. It deletes feature transformation and nonlinear activation, only retains neighborhood aggregation to update user and item embeddings, and then recommends according to the collaborative filtering framework.

UltraGCN[30], an ultra-simplified graph convolution network model, is improved from the single-layer LightGCN. It approximates the result of multi-layer graph convolution through constraint loss, skips the multi-layer message delivery mechanism, and then recommends it according to the collaborative filtering framework.

Among them, BPR-MF is a classic recommendation model based on matrix factorization, while the latter three are cutting-edge recommendation models based on graph convolutional networks. Therefore, this article chooses the above four methods for comparison.

In this experiment, the Xavier method is used to initialize the embedding parameters. The basic parameters of the model are set as follows: the user (item) embedding size is 64,

the batch size is 1024, the L2 regularization coefficient is 1e-4, the decay rate is 0, the number of the epoch is 1000, the number of network layers of NGCF and LightGCN is 3, and the other parameters are consistent with the parameters provided in the original papers.

### *4.4. Experimental Results.*

The methods of graph neural network models are superior to BPR-MF, indicating the effectiveness of graph neural network models. It is feasible to alleviate the problem of data sparsity by enhancing information exchange through high-order connectivity.

NGCF has the lowest performance among all methods of graph network models, as it is the model designed for recommendation in graph convolutional networks. Initially, graph convolution networks were designed for graph classification tasks, with preserved feature transformations and nonlinear activation, making the model redundant, increasing training difficulty, and reducing recommendation performance. LightGCN, on the other hand, removes feature transformations and nonlinear activations from NGCF, learns user and item embeddings through linear propagation on user item interaction graphs. This simple model is easier to train and implement than NGCF. However, the truly complex operation in graph convolution networks is neighborhood aggregation, where multiple updates of user and item embeddings have a significant impact on recommendation results. **Table 2** shows the results of comparative experiments.

**Table 2.** Overall Performance Comparison.

| | **Movielens-1M** | | **Yelp** | | **Gowalla** | |
|---|---|---|---|---|---|---|
| | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 |
| MF-BPR | 0.2017 | 0.1989 | 0.0553 | 0.0468 | 0.1563 | 0.1384 |
| NGCF | 0.2486 | 0.2343 | 0.0564 | 0.0475 | 0.1577 | 0.1391 |
| LightGCN | 0.2527 | 0.2454 | 0.0628 | 0.0547 | 0.1619 | 0.1424 |
| UltraGCN | <u>0.2739</u> | <u>0.2675</u> | <u>0.0672</u> | <u>0.0586</u> | <u>0.1654</u> | <u>0.1460</u> |
| GCN-MF | **0.2833** | **0.2772** | **0.0687** | **0.0598** | **0.1678** | **0.1482** |
| Improve | 3.43% | 3.62% | 2.23% | 2.04% | 1.45% | 1.51% |

UltraGCN performs best among all baseline models because it uses weighted MF instead of neighborhood aggregation operations and allows for more appropriate edge weight allocation, and flexible adjustment of relative importance between different types of relationships, thereby improving LightGCN's multi-layer messaging mechanism, demonstrating faster convergence speed and lower complexity. Compared to LightGCN, the model is easier to implement and more efficient, However, UltraGCN is only composed of single-layer graph convolution networks, so it can only utilize single-order neighborhood information and ignore the high-order information interaction of graph convolution networks, which will limit recommendation performance in cases of limited interaction.

GCN-MF outperformed all baseline models on three common datasets. Compared to UltraGCN, GCN-MF on Movielens-1M Recall@20 and NDCG@20 increased by 3.43% and 3.62%, on Yelp Recall@20 and NDCG@20 increased by 2.23% and 2.04%, on Gowalla Recall@20 and NDCG@20 increased by 1.45% and 1.51% respectively. The lower the sparsity on the data set, the higher the improvement on NDCG@20 and Recall@20. To some extent, it alleviates the problem of data sparsity.

GCN-MF uses matrix factorization to replace the neighborhood aggregation of graph convolutional networks with K largest singular vectors and singular values in the embedding layer and uses a multi-layer neural network instead of simple inner product operation in the interaction layer to further capture the interaction information between users and items, thus improving the multi-level message transmission mechanism of LightGCN.

The experimental results show that GCN-MF in this paper is an efficient and feasible collaborative filtering framework model.

**Table 3** shows the model training time, including the time of each epoch, the number of epoch and the training time on movielens-1M.

**Table 3.** Training time comparison on movielens-1M.

| Methods | Time/Epoch | Epoch | Training time |
|---|---|---|---|
| MF-BPR | 1.32s | 25 | 33s |
| LightGCN | 5.31s | 400 | 1816s |
| UltraGCN | 1.63s | 85 | 138.55s |
| GCN-MF | 1.83s | 95 | 173.85s |

The training time of LightGCN is 1816s, because lightGCN is a traditional GCN model. And the convergence speed of UltraGCN and GCN-MF aggregated in the improvement field is faster, 138.55s and 173.85s respectively. The training speed is increased by more than 10 times. From the training time, it is further verified that the simplification of graph convolution networks can indeed improve the recommendation efficiency of collaborative filtering tasks.

### *4.5. Parameter Analysis*

**Figure 3** shows the impact of parameter K on the results of movielens-1M, which is related to the number of singular values and singular vectors.

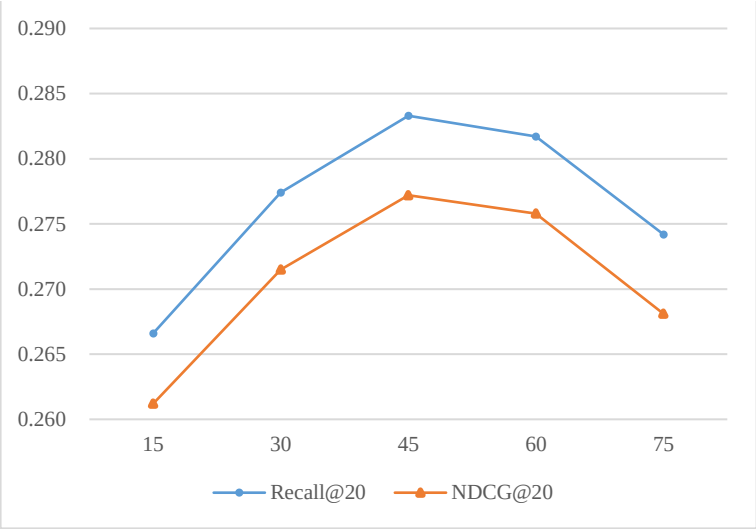


**Figure 3.** Impact of K.

The analysis interval of K is [15, 75]. When K is taken as 15, a small number of singular values and singular vectors cannot effectively represent matrix features. As K increases, the NDCG and Recall of the GCN-MF model show an upward trend. When K is taken as 45, the model achieves good results. If the value of K continues to increase, it will lead to a downward trend in the evaluation index. The reason is that if smaller singular values and singular vectors participate in the operation, it will not only increase the computational complexity of the model. Moreover, the noise of these smaller singular values and singular vectors will have an impact on the indicators, indicating that only taking the first 45 maximum singular values and singular vectors can improve the recommendation results of the model.

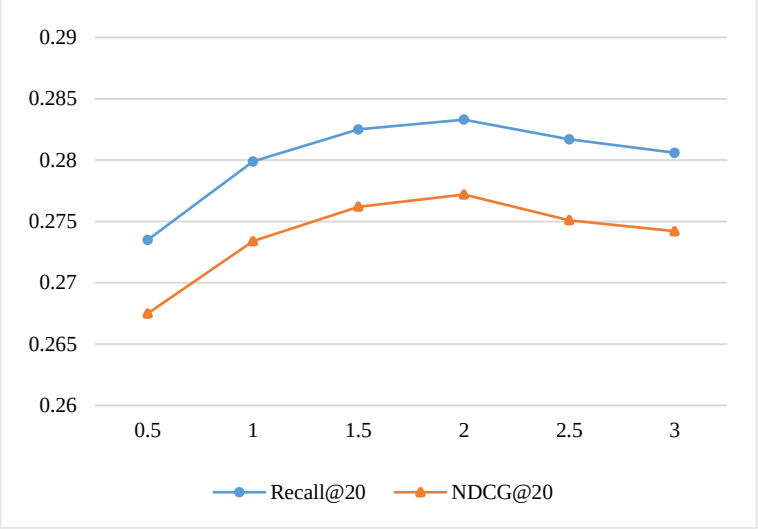


**Figure 4.** Impact of γ.

**Figure 4** shows the impact of parameter γ on the results of movielens-1M, which is related to singular value to singular vector weight.

The analysis interval of γ is [0.5, 3]. Starting from 0.5, as the value increases, the NDCG and Recall of the GCN-MF model show an upward trend. When the value is 2, the model achieves good results. If the value continues to increase, it will lead to a downward trend in the evaluation indicators. This indicates that when the value is 2, the recommended results of the model can be improved.

**Figure 5** shows the impact of parameter L on the results of movielens-1M, which is related to the number of neural network layers.

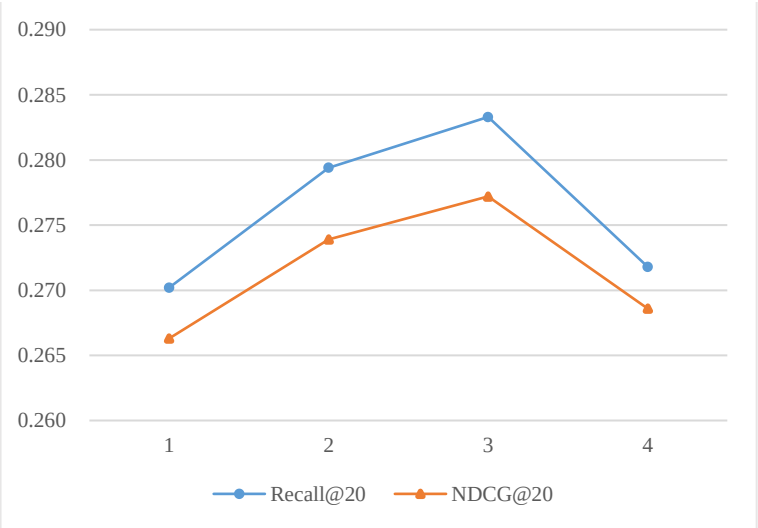


**Figure 5.** Impact of L.

The analysis interval of L is [1, 4]. When L is taken as 1, it cannot effectively represent the interaction between users and projects. As the number of layers increases, the NDCG and Recall of the GCN-MF model show an upward trend, indicating that adding multi-layer neural networks to learn the interaction between users and items can improve the prediction results. When L is taken as 3, the model achieves good results. If the number of layers continues to increase, the evaluation indicators will show a downward trend. The reason is that the overly complex model will not only increase the training cost, but

also cause the problem of over-fitting. This indicates that when the number of interaction layers L is 3, the recommendation effect can be significantly improved.

## 5. Conclusion

This paper proposes a graph convolutional network recommendation model that combines matrix factorization and graph network into the collaborative filtering framework and improves the efficiency of recommendation. Experiments were conducted on the real data set, using Recall and NDCG as evaluation indicators for the model. The proposed model was compared with baseline models such as MF-BPR, NGCF, LightGCN, UltraGCN. The experimental results showed that the proposed model outperformed the baseline models and provided users with more accurate recommendation results.

In future work, firstly, the model proposed in this paper can be combined with other models to achieve more efficient recommendations. Secondly, this model only uses user and item interaction bipartite graphs to mine higher-order information of users and items. It can be considered to incorporate different auxiliary information, such as comment text, user interests, etc. into the model to further improve recommendation results.

In addition to the recommendation field, artificial intelligence technology is also being applied in medical diagnosis, traffic control, and agricultural intelligence. With the innovation of algorithms and the improvement of computing power, the application field of artificial intelligence will further expand and deepen.

## Funding

This research was supported by the Social Science Foundation of Beijing (22GLB028).

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

1. Hanafi, M., Suryana, N., Bin, S., et al. Paper survey and example of collaborative filtering implementation in recommender system. Journal of Theoretical and Applied Information Technology, 2017, 3195(16), 4001-4014.
2. Ye, X., Yuan, P., Guo, X., et al. Collaborative filtering recommendation algorithm based on user interest and project cycle. Journal of Nanjing University of Science and Technology, 2018, 42(4), 392-400. https://doi.org/10.14177/j.cnki.32-1397n.2018.42.04.002
3. Sarwar, B., Karypis, G., Konstan, J., et al. Analysis of recommendation algorithms for e-commerce. In proceedings of the 2nd ACM Conference on Electronic Commerce, 2000: 158-167. https://doi.org/10.1145/352871.352887
4. Sedhain, S., Menon, A. K., Sanner, S., Xie, L. 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
5. Ying, S., Hoens, T. R., Jian, J., et al. Deep Crossing: Web-Scale Modeling without Manually Crafted Combinatorial Features. In proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, 255-62. https://doi.org/10.1145/2939672.2939704
6. Qu, Y., Han, C. Product-Based Neural Networks for User Response Prediction. In proceedings of 2016 IEEE 16th International Conference on Data Mining (ICDM), 2016, 1149-5. https://doi.org/10.1109/ICDM.2016.0151
7. Cheng, H-T., KOC, L., Harmsen, J., et al. Wide & deep learning for recommender systems. In proceedings of the 1st workshop on deep learning for recommender systems, 2016, 7-10. https://doi.org/10.1145/3041021.3054227
8. Zhu, H., Li, X., Zhang, P., et al. Learning tree-based deep model for recommender systems. In proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018, 1079-1088. https://doi.org/10.1145/3219819.3219826
9. Wu, Z., Pan, S., Chen, F., et al. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 2020, 32(1),4-24. https://doi.org/10.1109/TNNLS.2020.2978386
10. Ying, R., He, R., Chen, K., et al. Graph convolutional neural networks for web-scale recommender systems. In proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 2018, 974-983. https://doi.org/10.1145/3219819.3219890

11. Wang, M., Lin, Y., Lin, G., et al. M2GRL: A multi-task multi-view graph representation learning framework for web-scale recommender systems. In proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. 2020, 2349-2358. https://doi.org/10.1145/3394486.3403284
12. Huang, Z., Lin, Z., Gong, Z., et al. A two-phase knowledge distillation model for graph convolutional network-based recommendation. International Journal of Intelligent Systems, 2022, 37(9), 5902-5923. https://doi.org/10.1002/int.22819
13. Chen, L., Bi, X., Fan, G., et al. A multitask recommendation algorithm based on DeepFM and Graph Convolutional Network. Concurrency and Computation: Practice and Experience, 2023, e7498. https://doi.org/10.1002/cpe.7498
14. He, X., Zhang, H., Kan, M.Y., et al. Fast matrix factorization for online recommendation with implicit feedback. In proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. 2016, 549-558. https://doi.org/10.1145/2911451.2911489
15. Li, C., Che, H., Leung M.F., et al. Robust multi-view non-negative matrix factorization with adaptive graph and diversity constraints. Information Sciences, 2023, 634: 587-607. https://doi.org/10.1109/CCBD.2016.012
16. Wang, S., Zhang, Y., Lin, X., et al. Learning matrix factorization with scalable distance metric and regularizer. Neural Networks, 2023, 161, 254-266. https://doi.org/10.1016/j.neunet.2023.01.034
17. Salakhutdinov, R., Mnih, A. Probabilistic matrix factorization. In proceedings of the 20th International Conference on Neural Information Processing Systems, 2007, 1257–1264.
18. Kipf, T. N., Welling, M. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016. https://doi.org/10.1109/ACCESS.2021.3060173
19. Wang, X., He, X., Wang, M., et al. 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
20. Liu, K., Xue, F., Hong, R. RGCF: Refined graph convolution collaborative filtering with concise and expressive embedding. Intelligent Data Analysis, 2022, 26(2), 427-445. https://doi.org/10.3233/IDA-205725
21. Chen, L., Wu, L., Hong, R., et al. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In proceedings of the AAAI conference on artificial intelligence. 2020, 34(01), 27-34. https://doi.org/10.1609/aaai.v34i01.5330
22. He, X., Deng, K., Wang, X., et al. 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
23. Jin, R., Li, D., Gao, J., et al. Towards a better understanding of linear models for recommendation. In proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021, 776-785. https://doi.org/10.1145/3447548.3467428
24. Li, Q., Han, Z., Wu, X. Deeper insights into graph convolutional networks for semi-supervised learning. In proceedings of the AAAI conference on artificial intelligence. 2018, 32(1),3528-3538. https://doi.org/10.48550/arXiv.1801.07606
25. Risi, I. K. Diffusion kernels on graphs and other discrete input spaces. In Proceedings of the 19th International Conference on Machine Learning, 2002, 315-322.
26. He, X., Liao, L., Zhang, H., et al. Neural collaborative filtering. In proceedings of the 26th International Conference on World Wide Web. 2017, 173-182. https://doi.org/10.1145/3038912.3052569
27. Hornik, K., Stinchcombe, M., White, H. Multilayer feedforward networks are universal approximators. Neural networks, 1989, 2(5), 359-366. https://doi.org/10.1016/0893-6080(89)90020-8
28. He, R., McAuley, J. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In proceedings of the 25th international conference on world wide web. 2016: 507-517. https://doi.org/10.1145/2872427.2883037
29. Rendle, S., Freudenthaler, C., Gantner, Z., et al. BPR: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618, 2012. https://doi.org/10.48550/arXiv.1205.2618
30. Mao, K., Zhu, J., Xiao, X., et al. UltraGCN: ultra simplification of graph convolutional networks for recommendation. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 2021: 1253-1262. https://doi.org/10.1145/3459637.3482291