applied sciences

MDPI

*Article*

# Collaborative Filtering Model of Graph Neural Network Based on Random Walk

**Jiahao Wang, Hongyan Mei *, Kai Li, Xing Zhang and Xin Chen**

School of Electronics and Information Engineering, Liaoning University of Technology, Jinzhou 121000, China
* Correspondence: liaoning_mhy@126.com

**Abstract:** This paper proposes a novel graph neural network recommendation method to alleviate the user cold-start problem caused by too few relevant items in personalized recommendation collaborative filtering. A deep feedforward neural network is constructed to transform the bipartite graph of user–item interactions into the spectral domain, using a random wandering method to discover potential correlation information between users and items. Then, a finite-order polynomial is used to optimize the convolution process and accelerate the convergence of the convolutional network, so that deep connections between users and items in the spectral domain can be discovered quickly. We conducted experiments on the classic dataset MovieLens-1M. The recall and precision were improved, and the results show that the method can improve the accuracy of recommendation results, tap the association information between users and items more effectively, and significantly alleviate the user cold-start problem.

**Keywords:** recommender systems; graph neural networks; spectral domain graph convolution; collaborative filtering

## 1. Introduction

How users' int determine preferences are perceived and how much they engage with items usually determines the effectiveness of a recommender system (RS). Collaborative filtering is one of the most popular and well-known approaches in RS (CF) [1]. The model and memory-dependent approach has achieved some success. However, some problems still need to be solved: (1) In memory-based recommendations, the matrix sparsity problem arises when there are few user–item interactions. The method's performance decreases as the number of users and items increases. Users are less likely to be recommended if they have never interacted with an item. (2) In model-based recommendations, such as logistic regression (LR) [2], not only is feature crossover and feature filtering not possible, but the performance of the model representation could be better. Later, the researchers improved the model. While performance improved to some extent, it made the data sparser, making the model difficult to converge and increasing the feature weights and training cost. Later, factor decomposers (FM) [3] also faced problems such as combinatorial explosion and difficulty fusing higher-order features. The domain-aware factorization model (FFM) [4] further increased the training cost of the model, complicating parallel training and increasing training time. A recommendation method is proposed to solve these problems, called the graph convolutional collaborative filtering recommendation algorithm based on random wandering (RW-GCF). This method performs spectral convolution between spectral domains. It optimizes the adjacency matrix by random walking to find the hidden information connecting users and items in the graph. In addition, the new approach is to reduce the complexity of the model by optimizing the convolutional kernel, constructing a deep feedforward neural network, and using finite-order polynomials to adjust the size of each spectral domain dynamically, to discover user and item relevance and achieve mitigation of the user cold start problem in collaborative filtering. In addition, the random



**Citation:** Wang, J.; Mei, H.; Li, K.; Zhang, X.; Chen, X. Collaborative Filtering Model of Graph Neural Network Based on Random Walk. *Appl. Sci.* **2023**, *13*, 1786. https://doi.org/10.3390/app13031786

Received: 30 October 2022
Revised: 16 January 2023
Accepted: 17 January 2023
Published: 30 January 2023

walk approach has also played an important role in other fields; for example, in the medical field, Shen L et al. [5] combined the unbalanced double random walk approach, constructed a virus–drug association (VDA) identification framework (VDA-rwlrls), and implemented unbalanced double random walks on the viral network and drug network, respectively, to find clues for the treatment of COVID-19, and made good progress. The authors of [6] applied a two-dimensional random walk Monte Carlo to perform calculations to better understand the spread of COVID-19, and the model designed can predict second and third-wave infections with reasonable accuracy, which helps in planning mitigation strategies during the current pandemic.

## 2. Related Work

### 2.1. Deep-Learning-Based Recommendation System

The first deep learning method used in recommendation systems is the restricted Boltzmann machine (RBM) [7] approach, which uses users' rating preferences for modeling and outperforms traditional matrix decomposition techniques, providing a broad application prospect for deep recommendation systems. For collaborative filtering tasks, Zheng et al. [8] came up with the CF neural autoregressive distribution estimator (CF-NADE) model, which can share parameters across ratings and scalability. The scalability of CF-NADE was significantly improved. Subsequently, Wang Jun et al. [9] used generative and discriminant models for fusion and iterative optimization of the two models, which achieved good results and alleviated the item recommendation problem. The method allows learning domain-specific [10] representations from the interaction matrix of source and target users with the item, and it can effectively alleviate the data sparsity problem. The literature [11] uses a multilayer perceptron (MLP) to model user–item interactions, which further optimizes the problem of CF endogenous feedback and opens up new research avenues for deep learning recommendation algorithms. In addition, convolutional neural networks (CNN) [12], recurrent neural networks (RNN) [13], and deep reinforcement learning (DRL) [14] have been widely used in CF to make a breakthrough in alleviating the cold start and data sparsity problems of CF. Additionally, the algorithm's performance can be improved.

### 2.2. Graph-Based Recommendation System

Another related research direction is using user–item graph structures for the recommendation. The motivation for applying graph neural network methods to recommend systems lies in two aspects: first, most of the data in RS have a graph structure, and second, GNN techniques are potent in capturing connections between nodes and learning representations of graph data; for example, they have made significant progress in learning node representations [15], improving social network performance [16], and optimizing node embeddings [17]. Due to GNN's capacity to learn representations of potential factors from graphs, researchers have proposed graph-based RS, such as semi-supervised learning models for document recommendation, which can measure item similarity by combining multiple graphs. Inspired by graph/node embedding technology, Berg et al. [18] proposed an algorithm based on a graph convolution network, an automatic graph encoder; it uses the structural information of graphs to identify potential users and items to alleviate the matrix completion problem in deep learning. A graph-based regularization method is added by Berg to the matrix decomposition model to learn the graph structure, resulting in traditional graph-based regularization methods. Since then, many graph neural network-based recommendation systems have been proposed, and these methods have also made considerable progress in alleviating CF-related problems.

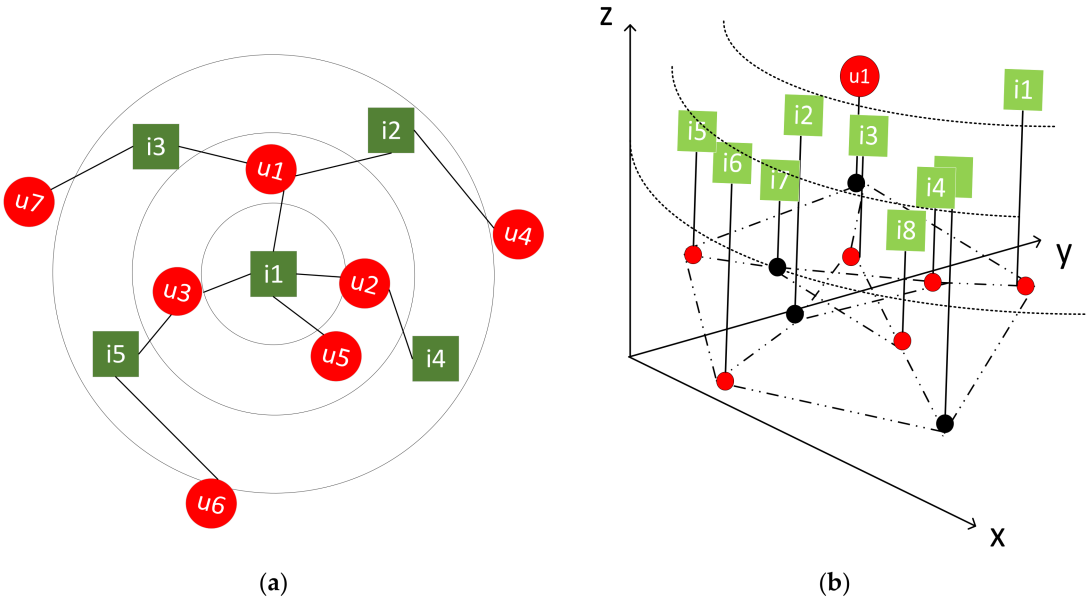
Despite the advantages of traditional CF methods, there are still many shortcomings in dealing with the cold-start problem: firstly, the results are not accurate enough when finding the nearest neighbors for the target users in the cold-start environment, or there are cases when the nearest neighbors cannot be found; secondly, the complexity of the algorithm increases due to the large number of calculations involved, and this problem

also increases the time required to train the model. In contrast, the algorithm proposed in this paper can further improve the performance of the recommendation system by taking into account the potential association information between users and items. This method can be directly learned from the bipartite graph constructed by users and items, with no need to add edge-related information. It can quickly discover implicit information about users and items.

The paper's main contribution can be summarized as follows: (1) Helps users discover more related items. (2) Using finite order polynomials to optimize the convolution process to help users discover items quickly and reduce the computational complexity. (3) The user–item spatial domain information is transformed into a new spectral domain space for the recommendation algorithm.

## 3. Model Framework

In this section, the user–item adjacency matrix is first optimized, relying on the higher-order connectivity of the graph. A random walk method reveals the deep relationships between users and items. Subsequently, the graph Fourier transform is performed on the bipartite graph. The vertices of the bipartite graph (users and items) are dynamically filtered using a new spectral convolution filter to measure the magnitude of each frequency component. Finally, a finite-order polynomial is used to overcome the drawbacks of the convolution operation. The final recommended method, RW-GCF, is introduced by a convolution operation consisting of multiple inter-stacked spectral map convolution layers, which transform the user–item interaction from the spatial domain to the spectral domain—see Figure 1.




(**a**) (**b**)

**Figure 1.** This is a user–project interaction diagram: (**a**) user–project space interaction diagram; (**b**) user–project spectrum domain interaction diagram.

### *3.1. Dichotomous Diagram*

In graph theory, a bipartite user–item graph *G(U,I,E)* is defined as having $N$ vertices and $E$ edges, where $U$ and $I$ are two disjoint sets of vertices of users and items. Each edge $e \in E$ follows the form $e = (u, i)$, where $u \in U, i \in I$, represents the interaction of user $u$ with item $i$ in the training set.

### 3.2. Implicit Feedback Matrix

The implicit feedback matrix $R$ defines $|U| \times |I|$ as follows:

$$R = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1j} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2j} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ r_{i1} & r_{i2} & \cdots & r_{ij} & \cdots & r_{in} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mj} & \cdots & r_{mn} \end{pmatrix} \tag{1}$$

The value is 1 if $u$ and $i$ interact, and 0 if no interaction is recorded.

### 3.3. Adjacency Matrix

For a bipartite graph $G$, its corresponding adjacency matrix $W$ can be defined as:

$$W = \begin{bmatrix} 0 & R \\ R^T & 0 \end{bmatrix} \tag{2}$$

### 3.4. Random Walk Method

In its most basic form, graph theory studies data structured as a graph. Three fundamental components make up a graph: the nodes, the edges, and the weights of the edges. A graph is a mathematical model that depicts a link between entities. Processing signals defined on an undirected connected weighted graph $G = V, E, W$, where $V$ is a finite number of vertices. $V = v1, v2, v3...$ $E$ denotes the collection of edges. $E = e1, e2$, and $e3$ are the edges of the graph, and the edges are abstract representations of the relationship between the vertices. The vertices of the graph might be any actual or abstract individuals.

Given a set of vertices $V$, there are three common ways to construct a graph: the first is to construct a $k$-nearest neighbor graph: and each vertex in the graph is connected to only its $k$-nearest neighbors. The second is constructing a full graph: each node is connected to all other nodes. The third is to construct a hypergraph: the essential feature of a hypergraph is its hyperedges, which can connect more than two vertices (including two). Given a set of vertices $V$, there are three common ways to construct a graph: the first is to construct a $k$-nearest neighbor graph—each vertex in the graph is connected to only its $k$ nearest neighbors. The second is to construct a full graph: each node in the graph is connected to all other nodes. The third is to construct a hypergraph: the basic feature of a hypergraph is its hyperedges, which can connect more than two vertices (including two).

Based on these three main factors, researchers have designed various methods to calculate the correlation between vertices in graphs [19]. This study is implemented by the random walk method [20] in graph embedding. Suppose that to personalize a recommendation to user $u$, a random traversal of the user's item bipartite graph starts at node $U$, which corresponds to user u. When the user walks past a node, the first decision is whether to continue or stop walking. The decision is made with probability $\alpha$. The next step is to determine which direction the user should go. If the user decides to continue, a node is selected from the nodes pointed to by the current node according to a uniform distribution. After several random walks, the probability that a particular item will be visited converges to the value determined by $\alpha$. Assume that an item is likely to be visited, in which case it will be given a greater weight in the final list, so the weight of the item in

the final recommendation list is its probability of being visited. The following equation can be generated by expressing the previous description in the form of an equation, as follows:

$$GEPR(v) = \begin{cases} \alpha \sum\limits_{v' \in \text{in}(v)} \frac{PR(v')}{|\text{out}(v')|} \ (v \neq v_u) \\ (1-\alpha) + \alpha \sum\limits_{v' \in \text{in}(v)} \frac{PR(v')}{|\text{out}(v')|} \ (v = v_u) \end{cases} \tag{3}$$

In the formula, *PR(i) denotes the access* probability of item *i*, *out(i)* denotes the out-degree of item node *i*. $\alpha$determines the probability of further visits. In collaborative filtering, the above relationship between users and items is mainly represented as a two-dimensional matrix (user–item matrix).

### *3.5. Laplace Operator Matrix*

In the normal form of the symmetric matrix, the Laplacian matrix is denoted by $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, where $L$ is the matrix with specification $N \times N$, $D$ is a degree matrix with a size of $N \times N$, and $D_{n\times n} = \sum_j A_{n,j}$. The problem of implicit feedback recommendations is the focus of this study. In the absence of an explicit rating record, the focus is only on whether the user has viewed, liked, or clicked on an item. The set of all items liked by the user is denoted by $I_i^+$, and the set of all other objects is denoted by $I_i^-$.

### *3.6. Relevant Problem Definition*

Provided a user set $U$ and an item set $I$, the purpose is to recommend a sorted list of items of interest to $I_i^-$ for each user who likes/clicks/views the item set $I^+ \subseteq I$.

### *3.7. Graph Fourier Transform*

For a graph $G = \{V, E\}$, where $V, E$ is the set of vertices and edges, each signal vector of the graph is $x \in R^{|V|\times 1}$, where $x_j$ denotes the value of the $j$th signal on the graph

The formula for the standard Fourier transform function is $f$, and the complex exponential expansion is shown in Equation (4):

$$f(t) = f^{-1}[F(w)] = \int_R F(t)e^{2\pi iwt}dt \tag{4}$$

$e^{2\pi iwt}$ forms a standard orthogonal basis.

In order to perform the Fourier transform of the signal $x \in R^{|V|\times 1}$ on the graph, it is necessary to find a set of orthogonal bases, represented by a linear combination of this set of orthogonal bases, so the graph Fourier transform uses the eigenvectors of the Laplace eigenmatrix $U = (\vec{u_1}, \vec{u_2}, \ldots, \vec{u_n})$ as the basis functions of the graph Fourier transform.

Using the Laplace eigenvectors as basis functions, since the Laplace eigenvector $U = (\vec{u_1}, \vec{u_2}, \ldots, \vec{u_n})$ is an n-dimensional linearly independent vector, the basis of a space is composed of n linearly independent vectors in an n-dimensional space, and the eigenvectors of the Laplace matrix are also a set of orthogonal basis, as shown in Equation (5), the signal on the graph can be expressed as:

$$x = \hat{x}(\lambda_1)\vec{u_1} + \hat{x}(\lambda_2)\vec{u_2} + \cdots + \hat{x}(\lambda_n)\vec{u_n} \tag{5}$$

The Fourier transform of a graph is defined as the expansion of the eigenvector of the observed graph signal according to the Laplace operator $L$ of the graph, and the eigenvector

can be used as a basis for the spectral domain. Then, the signal ($x \in R^{|V|\times 1}$) of the graph $G$ is shown in Equation (6) and the Fourier transform of the graph is defined as:

$$\hat{x}(l) = \sum_{j=0}^{N-1} x(j)\mu_l(j) \begin{pmatrix} \hat{x}(l_1) \\ \hat{x}(l_2) \\ \vdots \\ \hat{x}(l_n) \end{pmatrix} = \begin{bmatrix} u_1(1) & u_1(2) & \cdots & u_1(4) \\ u_2(1) & u_2(2) & \cdots & u_2(4) \\ \vdots & \vdots & \ddots & \vdots \\ u_n(1) & u_n(2) & \cdots & u_n(n) \end{bmatrix} \begin{pmatrix} x(1) \\ x(2) \\ \vdots \\ x(n) \end{pmatrix} \tag{6}$$

The inverse transformation of the graph Fourier is shown in Equation (7):

$$x(j) = \sum_{j=0}^{N-1} x(l)\mu_l(j) \begin{pmatrix} x(1) \\ x(2) \\ \vdots \\ x(n) \end{pmatrix} = \begin{bmatrix} u_1(1) & u_1(2) & \cdots & u_1(4) \\ u_2(1) & u_2(2) & \cdots & u_2(4) \\ \vdots & \vdots & \ddots & \vdots \\ u_n(1) & u_n(2) & \cdots & u_n(n) \end{bmatrix} \begin{pmatrix} \hat{x}(l_1) \\ \hat{x}(l_2) \\ \vdots \\ \hat{x}(l_n) \end{pmatrix} \tag{7}$$

where $\hat{x}(l)$ denotes the $j$th value of l, $x(j)$ and $\mu_l(j)$ as above, respectively; $\mu_l$ denotes the $l$th vector of $L$; $\hat{x}$ denotes the graph signal that has been converted to spectral domain. For simplicity, the equations in (6) and (7) can be converted to $\hat{x} = U^T x$ and x $= U\hat{x}$, where $U = (\vec{u_1}, \vec{u_2}, \ldots, \vec{u_n})$ is the eigenvector of $L$.

For the bipartite graph $G$, two graph signal vectors exist: $x^u \in R^{|U|\times 1}$ and $x^i \in R^{|I|\times 1}$, associated with user vertices and project vertices, respectively. The transform between the spatial and spectral domains and the inverse transformation from the spectral domain to the spatial domain are shown in Equation (8):

$$\begin{bmatrix} \hat{x}^u \\ \hat{x}^i \end{bmatrix} = U^T \begin{bmatrix} x^u \\ x^i \end{bmatrix} \text{and} \begin{bmatrix} x^u \\ x^i \end{bmatrix} = U^T \begin{bmatrix} \hat{x}^u \\ \hat{x}^i \end{bmatrix} \tag{8}$$

### *3.8. Spectral Convolution Process*

Extensive information about the graph structure exists in the spectral domain, where the Fourier basis formed by the Laplacian eigenvectors can project the graph signal into the orthogonal space, where different frequency domains can reveal different types of connectivity information between users and items. The ability to dynamically adjust the size of each frequency domain is essential for recommendation systems. Drawing on the idea of the convolution theorem to explore the potential relevance of the transformation of the null domain into the spectral domain, the convolution theorem equation is shown in Equation (9):

$$f_1(t) * f_2(t) = f^{-1}[F_1(w) \cdot F_2(w)] \tag{9}$$

where $f$ is the input signal, $f_2(t)$ is the convolution kernel, the input signal corresponding to the graph convolution is $x$, and the convolution kernel is $g_\theta$; then, the graph convolution process is shown in Equation (10):

$$g_\theta * x = f^{-1}(f(x) * f(g)) = U(U^T x * U^T g) \tag{10}$$

where $U^T x$ can be considered as the input map signal, $U^T g$ is the corresponding convolution kernel, and $*$ is the harand product. In this paper, a new convolution filter is designed to facilitate the extraction of more accurate features. That is, $\theta \in R^N$ can be considered as $g_\theta(\wedge) = diag([\theta_0\lambda_{0,}\theta_1\lambda_{1,}\ldots,\theta_{N-1}\lambda_{N-1}])$ when the spatial domain is transformed into the spectral domain, as shown in Equation (11):

$$g_\theta * x = \begin{bmatrix} x^u_{new} \\ x^i_{new} \end{bmatrix} = U g_\theta(\wedge) \begin{bmatrix} \hat{x}^u \\ \hat{x}^i \end{bmatrix} = U g_\theta(\wedge) U^T \begin{bmatrix} x^u \\ x^i \end{bmatrix} \tag{11}$$

where $x^u_{new}$ and $x^i_{new}$ is the new signal learned by the dichotomous graph $G$ filter $g_\theta(\wedge)$, $\wedge = \{\lambda_0, \lambda_1, \ldots, \lambda_{N-1}\}$ denotes the eigenvalues of the Laplacian matrix in the graph $L$.

In Equation (11), the convolution filter $g_\theta(\wedge)$ is placed on the spectrogram signal $\begin{bmatrix} \hat{x}^u \\ \hat{x}^i \end{bmatrix}$, and each value $\theta$ is responsible for enhancing or reducing the component corresponding to each frequency. The eigenvector matrix in the equation is $U$,and the inverse Fourier transform is performed using (11).

### *3.9. Local Filter Polynomial Parameterization*

This filter has two limitations. First, the learning complexity is $O(n)$, which affects the speed of data loading and the length of model training when the number of nodes is too large, and second, each user and item needs a vector to model the depth and complexity of the connection between the user and the item, and the process appears to be more complicated. This first limitation can be overcome by using a polynomial filter, as shown in Equation (12):

$$g_\theta(\wedge) \approx \sum_{K=0}^{K} \theta_K \wedge^k \tag{12}$$

### *3.10. Chebyshev Simplification*

Thanks to the recursive idea of Chebyshev, only the first-order Chebyshev polynomial is considered, and the k-order polynomial can be taken to be of order 1, then each convolution kernel has only one parameter, as shown in Equation (13):

$$g_\theta(\wedge) = \sum_{k=0}^{1} \theta_k(\widetilde{\wedge}) \tag{13}$$

When the order $k$ is 1, the complexity of the filter is reduced. However, in order to make the model more effective, this study tries to extend the order to 2, as shown in Equation (14):

$$g_\theta(\wedge) = \sum_{k=0}^{2} \theta_k(\widetilde{\wedge}) \approx \wedge + \theta_1 \wedge + \theta_2 \wedge^2 \tag{14}$$

In this way, the learning complexity of the filter becomes *O(P)*, where $P$ is a hyperparameter independent of the digital vertices. In particular, to avoid overfitting, this study restricts the order of the polynomial $P$ to 2. This is shown in Equation (15):

$$\begin{bmatrix} \mathbf{x}^u_{new} \\ \mathbf{x}^i_{\text{new}} \end{bmatrix} = \left(\theta'_0 U U^\top + \theta'_1 U \Lambda U^\top + \theta'_2 U \Lambda^2 U\right) \begin{bmatrix} \mathbf{x}^u \\ \mathbf{x}^i \end{bmatrix} \tag{15}$$

In addition, the optimization facilitates further reduction in the number of parameters by making $\theta' = \theta'_0 = \theta'_1 = \theta_2{}'$. where $\theta'$ is a scalar. As shown in Equation (16):

$$\begin{bmatrix} x^u_{new} \\ x^i_{new} \end{bmatrix} = \theta' \left(U U^\top + U \Lambda U^\top + U \Lambda^2 U^\top\right) \begin{bmatrix} x^u \\ x^i \end{bmatrix} \tag{16}$$

### *3.11. Learnable Parameter Optimization*

For the second limitation, the user and project inputs $x^u \in R^{|u| \times 1}$ and $x^i \in R^{|I| \times 1}$ are transformed into $C$ dimensional map signals, $x^u \in R^{|u| \times c}$ and $x^i \in R^{|I| \times c}$, and to facilitate the computation and reduce the complexity of the loop, the convolution filters are boosted and transformed into a convolution filter matrix $C$ with input channels $F$ and filters $\Theta \in R^{C \times F}$, making the final spectral convolution operation as shown in (17):

$$\begin{bmatrix} x^u_{new} \\ x^i_{new} \end{bmatrix} = g_\theta * x = \sigma(U U^\top + U \Lambda U^\top + U \Lambda^2 U^\top \begin{bmatrix} \mathrm{x^u} \\ x^i \end{bmatrix} \Theta) \tag{17}$$

where $x^u \in R^{|u|\times c}$ and $x^i \in R^{|I|\times c}$ represent the convolution results learned from the spectral domain of the user and the project using the filter $\Theta$, respectively; $\sigma$ also represents the logistic regression function, and the feedforward flow is shown in Figure 2.

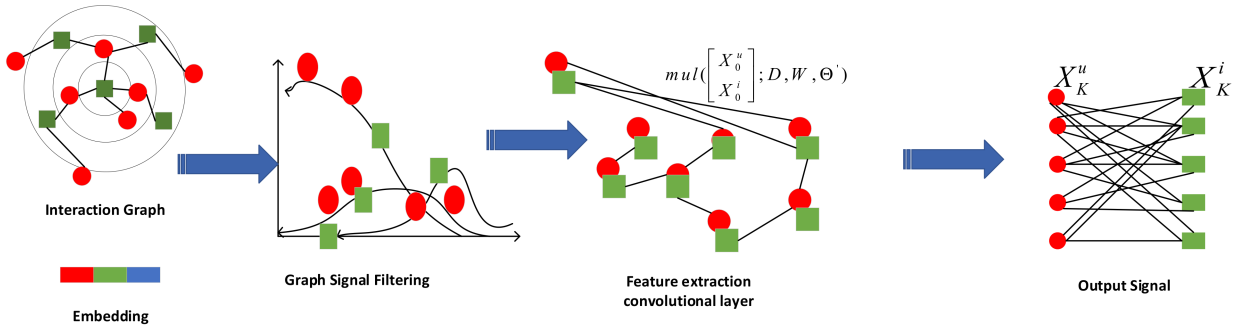



**Figure 2.** RW-GCF convolutional signal processing flow chart.

### *3.12. Multi-Layer Stacking*

Given a user vector $x^u$ and a project vector $x^i$, the new graph signals ($x^u_{new}$ and $x^{\mathrm{i}}_{new}$) are the result of learning the convolution from the spectral domain by means of the parameter matrix $\Theta \in R^{C\times F}$ in Equation (19). First, randomly initialize the user vector $x^u_0$ and the item vector $x^i_0$, using $x^u_0$ and $x^i_0$ as input, the $K$ layer RW-SCF can be expressed as (18):

$$H^{X^u_K}_{X^{\mathrm{i}}_K} = f(x^u_{new}, x^i_{new}, U, \wedge, \Theta_{K-1}) \tag{18}$$

where $\Theta_{K-1} \in R^{F\times F}$ is the filter parameter matrix of the $K$ layer; $\mathrm{x}^u_K$ and $\mathrm{x}^i_K$ denote the convolutional filtering results of the $K$ layer.

In order to utilize all the attributes within the hidden layers in RW-GCF, this study further relates them to the ultimate potential factors of users and items as shown in (19):

$$\Phi^{\mathrm{u}} = [X^u_0, X^u_1, \ldots, X^u_k] \tag{19}$$

where $\Phi^{\mathrm{u}} \in R^{|u|\times(C+KF)}$ and $\Phi^{\mathrm{i}} \in R^{|i|\times(C+KF)}$.

A loss function called BPR [21] is used to evaluate the implicit recommendation of the model. It is an unbiased loss function that takes into account the various factors that influence the rating process. It creates a triple $(r, j, j')$,, where $j$ denotes the items that user $r$ liked, clicked, or viewed, and $j'$ denotes the items that user $r$ did not like, click, or view. The loss function of RW-GCF is illustrated by maximizing the preference difference between $j$ and $j'$ given the user matrix $I^u$ and the item matrix $I^i(r, j, j')$.

$$L = \operatorname{argmin}(\Phi^u, \Phi^i) \sum_{(r,j,j')} -\ln\sigma(pos - neg) + \lambda_{\mathrm{reg}}(||r_u||^2_2 + ||j_i||^2_2 + ||j_{i'}||^2_2) \tag{20}$$

where $pos$ and $neg$ denote the potential factors of users and items, $\lambda_{reg}$ denotes the coefficients of the regularization term, and the training data are shown in (21):

$$\Gamma = \left\{(r, j, j') \middle| r \in U \wedge j \in I^+_i \wedge j' \in I^-_i\right\} \tag{21}$$

### *3.13. Optimization Functions and Prediction Results*

In this paper, we use a first-order gradient optimization algorithm for stochastic self-scalar functions based on low-order moment adaptation—Adam [22]. The Adam method is relatively easy to implement and is very efficient due to its low memory requirements. It considers the first- and second-order moment estimates of the gradient and then computes the adaptive learning rate. The Adam method is also very effective in smoothing and online setups. It combines the advantages of AdaGrad [23] and RMSProp [24], which are known

to be very useful when dealing with large data sets and complex algorithms. The method is also ideal for very noisy and sparse gradient problems. The hyper-parameters can also be interpreted intuitively, and the implementation usually does not require parametrization.

### 3.14. Algorithm Design

Step 1 Input user term implicit matrix $R$, neighborhood matrix $W$, batch size $B$, number of iterations $E$, and potential factor dimension $C$. The number of layers of convolution kernel $F$, learning rate $l_r$, regularization factor $\lambda_{\text{reg}}$,and number of iterations convergence k.

Step 2 Construct the implicit feedback matrix of users and items, and use the Gaussian mixture function $N(0.01, 0.02)$ to obtain the initial $X^u$, $X^i$ values to ensure the stability of the initial data distribution.

Step 3 Obtain the vector of size $B$ in the dataset and loop through $E$ iterations.

Step 4 The potential items preferred by users are mined by the random walk method of graph embedding, and the probability values of the items are converged and stabilized by setting appropriate parameters through several computational iterations. Then, use the matrix decomposition method to optimize the user–item adjacency matrix.

Step 5 Optimize the convolution process by Equations (14) and (15), using Chebyshev's first-order truncation and setting the size of the field of feeling to $k$, then simplify the convolution kernel by using a second-order finite-order polynomial for the convolution operation so that each convolution kernel $\Theta$ has only one parameter of the learnable system, and according to Equation (17), loop several times $K$ to obtain $X_K^{\mathrm{u}}$ and $X_K^i$.

Step 6 embeds $\left[X_0^i, X_1^i, \ldots, X_k^i\right]$ and $\left[X_0^u, X_0^u, \ldots, X_k^u\right]$ into the potential hidden vector, respectively.

Step 7 Optimize the gradient by the backpropagation algorithm.

Step 8 Update the gradient by Adam's algorithm.

Step 9 outputs the model parameters at $X_{0,}^{\mathrm{i}} X_0^u, \Theta_0, \Theta_1 \ldots \Theta_{K-1}$.

## 4. Experiments and Analysis of Results

### 4.1. Data Set

MovieLens-1M [25]: contains over 1,000,209 ratings from over 3900 movies. This study converts the rating data into implicit data, where 0 and 1 only indicate whether the user likes the item or not, and a dataset with a 3.0% density was selected and retained for this study.

HetRec [26]: contains over 855,598 ratings from over 10,197 movies. Handling of data is the same as MovieLens-1M, and a dataset with a 0.3% density was selected and retained for this study.

### 4.2. Baselines

In order to verify the validity of RW-GCF, this study compared it with six representative models.

NCF [9]: A fusion of neural collaborative filtering, matrix decomposition, and multilayer perceptron (MLP) for learning from user–project interactions. MLP enables NCF to model the nonlinearity between users and projects.

GCMC [18]: Graph convolution matrix complementary, which uses graph autoencoders to learn the underlying factors of the user and the project, i.e., the connectivity information of the two parts of the interaction graph.

SpectralCF [21]: A collaborative filtering algorithm that performs convolution operations directly on the spectral domain to improve the user's cold start problem.

NGCF [27]: a user–project interaction graph with three GNN layers designed to refine the representation of users and projects using information from their nearest third-order neighbors.

Light-GCN [28]: is a CF method based on GNN, able to choose GCN as aggregation technique, constant function as activation, i.e., to remove nonlinear computation, and mean as layer combination function.

DGCF [29]: A deconfined GNN model that uses neighbor routing and embedding propagation to deconfine the potential factors at the top edge of the graph.

### *4.3. Parameter Setting*

The following Table 1 is selected for the optimal hyperparameters after many experiments.

**Table 1.** This is a table of parameter settings.

| Hyper-Parameters | Numerical Value |
|---|---|
| Average value (μ) | 0.01 |
| Variance ($\sigma$) | 0.02 |
| Number of convolution layers ($K$) | 3 |
| Figure signal size ($C$) | 16 |
| Number of filter layers ($F$) | 16 |
| Regularization term ($\lambda$) | 0.001 |
| Training Batch ($B$) | 1024 |
| Number of iterations ($E$) | 230 |
| Learning rate ($l$) | 0.001 |
| Dropout | 0 |
| Number of convergence iterations ($Ci$) | 15 |
| Convergence rate ($\alpha$) | 0.55 |
| Number of iterations convergence ($k$) | 15 |

### *4.4. Evaluation Methodology*

The retrieval capability of the recommendation model should be extended to measure the accuracy of the recommendation system. in this study, recall@M, and p@M(precision) were used to evaluate the performance of the recommender system. The proportion of relevant items retrieved from all items was calculated using recall@M; p@M(precision) indicated the number of correctly recommended items among the top M items. Then, the definition of Recall@M and precision@M for each user is shown in Equations (22) and (23):

$$\text{Recall@M} = \frac{\text{Number of items correctly predicted}}{\text{Recommended list length}} \tag{22}$$

$$\text{Precision@M} = \frac{\text{Number of items correctly predicted}}{\text{The length of the actual user click list}} \tag{23}$$

### *4.5. Experimental Tests*

Tables 2 and 3 show the best records for recall@20 and precision@20 on the MovieLens-1M and HetRec datasets.

**Table 2.** Model Comparison: MovieLens-1M.

| Metrics | Recall | | | | | Precision | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Entity | recall@20 | recall@40 | recall@60 | recall@80 | recall@100 | p@20 | p@40 | p@60 | p@80 | p@100 |
| NGCF | 0.1366 | 0.1912 | 0.2333 | 0.2665 | 0.2952 | 0.0418 | *0.0295* | *0.0241* | *0.0207* | *0.0185* |
| Light-GCN | 0.1321 | 0.2021 | 0.2217 | 0.2454 | 0.2779 | 0.0443 | *0.0312* | *0.0297* | *0.0231* | *0.0202* |
| DGCF | 0.1499 | 0.2076 | 0.2378 | 0.2712 | 0.3122 | 0.0457 | *0.0331* | *0.0281* | *0.0246* | *0.0199* |
| GCMC | 0.1559 | 0.1822 | 0.2447 | 0.2551 | 0.2702 | 0.0497 | *0.0374* | *0.0331* | *0.0297* | *0.0245* |
| NCF | 0.1702 | 0.1949 | 0.2422 | 0.2551 | 0.2828 | 0.0505 | *0.0399* | *0.0348* | *0.0326* | *0.0273* |
| SpectralCF | 0.2373 | 0.2614 | 0.2969 | 0.3402 | 0.3714 | 0.0701 | *0.0527* | *0.0501* | *0.0499* | *0.0374* |
| RW-GCF | 0.2378 | 0.2725 | 0.3027 | 0.3429 | 0.3817 | 0.0845 | *0.0629* | *0.0601* | *0.0546* | *0.0477* |

**Table 3.** Model Comparison: HetRec.

| Metrics Entity | Recall | | | | | Precision | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **recall@20** | **recall@40** | **recall@60** | **recall@80** | **recall@100** | **p@20** | **p@40** | **p@60** | **p@80** | **p@100** |
| NGCF | 0.1284 | 0.1874 | 0.2276 | 0.2601 | 0.2901 | 0.0401 | *0.0277* | *0.0216* | *0.0199* | *0.0167* |
| Light-GCN | 0.1279 | 0.1916 | 0.2199 | 0.2577 | 0.2703 | 0.0427 | *0.0301* | *0.0264* | *0.0207* | *0.0183* |
| DGCF | 0.1402 | 0.2037 | 0.2264 | 0.2685 | 0.3049 | 0.0432 | *0.0328* | *0.0256* | *0.0232* | *0.0177* |
| GCMC | 0.1459 | 0.1722 | 0.2347 | 0.2451 | 0.2602 | 0.0488 | *0.0367* | *0.0309* | *0.0254* | *0.0221* |
| NCF | 0.1599 | 0.1837 | 0.2301 | 0.2413 | 0.2717 | 0.0491 | *0.0382* | *0.0327* | *0.0301* | *0.0255* |
| SpectralCF | 0.1977 | 0.2216 | 0.2614 | 0.2761 | 0.3299 | 0.0683 | *0.0508* | *0.0491* | *0.0467* | *0.0423* |
| RW-GCF | 0.2021 | 0.2342 | 0.2787 | 0.2901 | 0.3402 | 0.0801 | *0.0611* | *0.0593* | *0.0508* | *0.0466* |

As can be seen from the figure, the algorithm RW-GCF proposed in this paper outperforms the other seven algorithms after validation on the dataset MovieLens-1M. Compared with algorithms NCF, GCMC, NGCF, Light-GCN, and DGCF, all algorithms showed significant improvement in recall and a 2% improvement compared with SpectralCF, the most cutting-edge recommendation algorithm in the field of spectroscopy. In terms of accuracy improvement, the performance is still the best compared with NCF, GCMC, NGCF, Light-GCN, and DGCF, so the RW-GCF algorithm has the best recommendation result in the cold start environment. After analysis, the reason for this result is that the algorithm RW-GCF can perform the convolution operation in the graph domain directly, which not only relies on the unique properties of the graph to optimize the potential properties of the adjacency matrix, thus revealing the neighborhood information of the graph, but also reveals the connectivity information hidden in the graph domain. It can also reveal the hidden connectivity information in the graphs, and realize the deep connection between users by using the popular convolution method to perform the computation for fast discovery. Secondly, to demonstrate the generality of the algorithm, this study also conducted experimental validation on the HetRec dataset and found that the performance of the proposed method is still the best.

## 5. Discussion

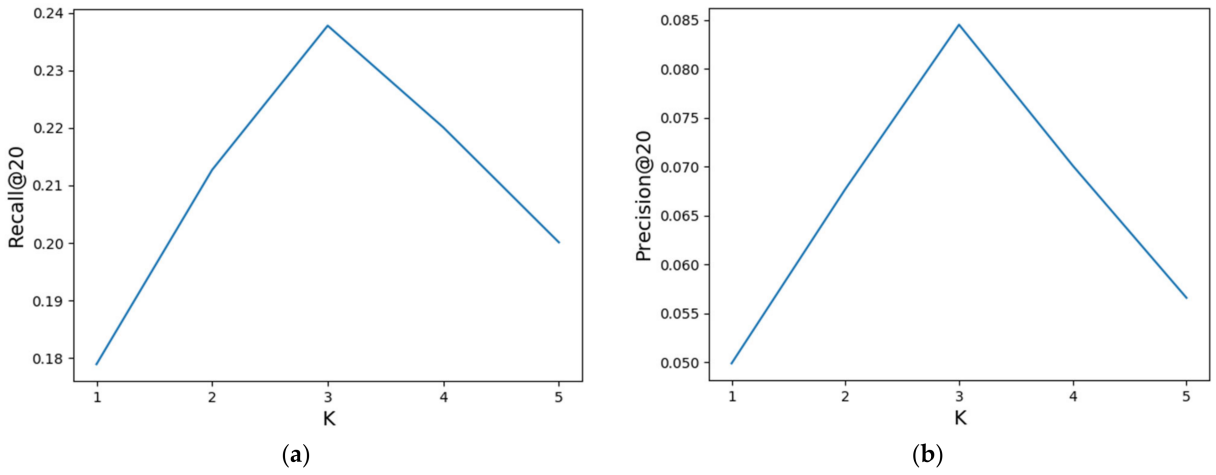
### *5.1. Parameter Discussion*

In this section, the effects of the parameters involved in the studied algorithm are discussed. Since the algorithm achieves good results on both datasets, a more detailed comparative study of the hyper-parameters is carried out on the MovieLens-1M dataset. In the proposed method, the number of convolutional layers, the dimensionality of the graph signal, and the number of filters all play an important role in the performance of the recommendation algorithm; therefore, an experimental approach is used in this study to verify the values of the relevant parameters K, C, and F. Regarding the data distribution of the test and training sets, this study follows the randomization principle to ensure the probability balance. In this study, 80% of the items were selected as the training set, and the remaining items were used as the test set. In addition, the validation set derived from the training set of each data set was used to locate the optimal hyper-parameters.

#### 5.1.1. Optimal Number of Convolution Layers

In this study, the effect of the *K* values of the convolutional layers on the recommended performance is first discussed. Figure 3 shows the recall@20 and precision@20 of the proposed method at different *K* values. The results in Figure 3 show that the performance of the algorithm is relatively low when the number of convolutional layers is small. As the number of convolutional layers increases, the performance of the proposed algorithm gradually improves but reaches a peak and then starts to decrease. It indicates that as the number of convolutional layers increases, the perceptual field of the spectral convolutional network becomes gradually larger, but a continuous increase will risk gradient disappear-

ance and gradient explosion, and it is experimentally verified that the evaluation index will reach a peak when the value of $K$ is 3, which is the best result, as shown in Figure 3 below.

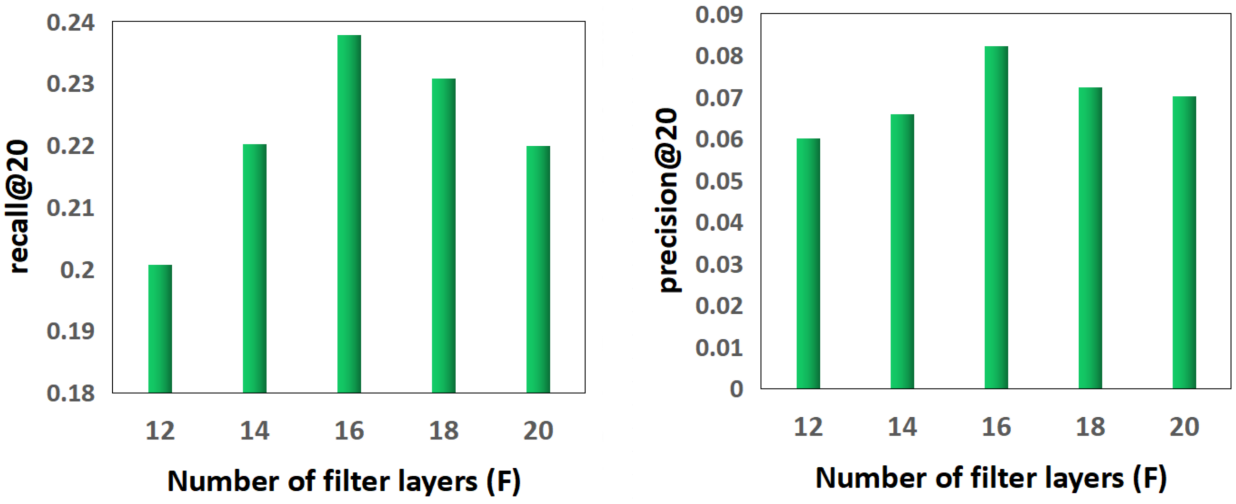



(**a**) (**b**)

**Figure 3.** Effects of hyper-parameter $K$ in terms of Recall@20 and Precision@20 in the dataset of MovieLens-1M: (**a**) effects of hyper-parameter $K$ in terms of Recall@20; (**b**) effects of hyper-parameter $K$ in terms of Precision @20.

#### 5.1.2. Optimal Number of Filters

Secondly, the number of filters $F$ is discussed; in this lab, the number of filters is firstly adjusted. This is because each vertex of a user or item is represented by a scalar feature. However, each user and item needs a vector to model the depth and complexity of the connection between the user and the item. A reasonable choice of the value of F also plays an important role in constructing a suitable graph signal and thus also plays a driving role in improving the model performance. This study found that there is a significant difference in the effect on the efficiency of the algorithm when the number of filters is between 12and 20; the number of filters has a crucial role in constructing the convolutional layers of the graph convolutional neural network. The initial value of the filter is 12, and 2 is added each time. The algorithm can achieve the best effect when $F$ is 16, recall@20 is 0.2387, and p@20 is 0.0885, as shown in Figure 4 below.




(**a**) (**b**)

**Figure 4.** This is a filter parameter tuning diagram: (**a**) effects of hyper-parameter $F$ in terms of Recall@20; (**b**) effects of hyper-parameter $F$ in terms of Precision@20.

5.1.3. Initializing Gaussian Distribution

When initializing the parameters, a Gaussian distribution is used for the assignment because, in the training process, the parameters may not always conform to the Gaussian distribution, and a series of normalization methods will occur in an attempt to pull the parameters in training back to the Gaussian distribution to some extent. The reasonable selection of the mean and variance of the Gaussian distribution can accelerate the convergence of the network and enable the model to quickly calculate a stable value. For the selection of mean and variance, this study has been verified and found that a mean value of 0.01 and a variance of 0.01 can make the evaluation efficiency of this model reach the best value; the selection of the mean and variance records are shown in Table 4.

**Table 4.** Gaussian matrix optimal hyper-parameters.

| | $\mu$ = 0.07 $\sigma$ = 0.015 | $\mu$ = 0.09 $\sigma$ = 0.018 | $\mu$ = 0.01 $\sigma$ = 0.02 | $\mu$ = 0.02 $\sigma$ = 0.04 | $\mu$ = 0.04 $\sigma$ = 0.06 |
|---|---|---|---|---|---|
| recall@20 | 0.0209 | 0.2102 | 0.2378 | 0.2214 | 0.2199 |
| precision@20 | 0.0701 | 0.0766 | 0.0845 | 0.0801 | 0.0794 |

5.1.4. Optimal Number of Iterations

This study found that if a recommendation is given to the user, then a random roam is made from the node corresponding to the user in the form of a bipartite graph. If the decision is made to continue the roaming, then a node from the node to which the current node points is randomly selected as the next node to be roamed according to a uniform distribution. In this way, after several random walks, the probability of each item node being visited converges to a single number. The weight of an item in the final recommendation list is the probability of that item node being visited. This study finds that the number of iterations gradually begins to stabilize at 12, and when 15 is reached as a stable value in order to obtain the most complete list of item recommendations, the probability of each item being visited reaches convergence, and the probability of visit remains stable, at which point the number of iterations is the best parameter to make the model evaluation efficiency optimal; this study carries out the following statistics on the relevant records, as shown in Figure 5 below.

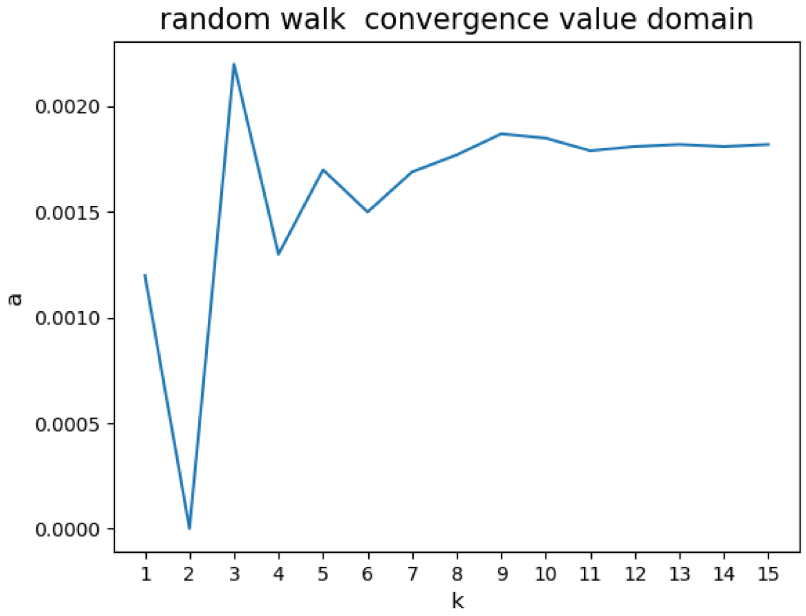



**Figure 5.** Random walk convergence value.

#### 5.1.5. Best Batch Size

In this study, the number of data to be thrown into the model for training is a value between 1 and the total number of training samples; it is not good if the batch size is too large or too small. If the value is too large, assuming Batch_Size=100000, throwing 100,000 pieces of data into the model at one time will probably cause memory overflow and prevent normal training. If the Batch_Size is too small compared to the normal data set, the training data will be very difficult to converge, which will lead to underfitting. By increasing the Batch_Size, the relative processing speed will be faster, and the memory required will be increased in order to achieve better training results.

We generally need to increase the number of training sessions for all samples while increasing the batch size to achieve the best results. Increasing batch size generally increases the number of training sessions for all samples, which also leads to an increase in time consumption, so we need to find a suitable batch size value to achieve the best balance between overall model efficiency and memory capacity. In this study, the best value for the model evaluation efficiency is selected when the batch size is 1024 through several validations, and the model efficiency can reach a peak when the batch size is equal to 1024, which is the best parameter that can be selected for the batch size. The relevant records are shown in Figure 6.

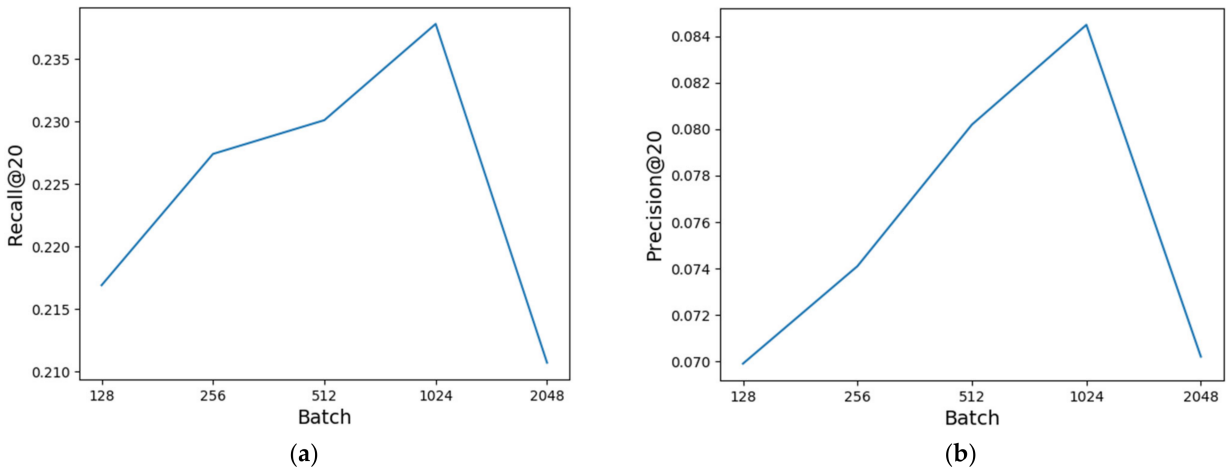



**Figure 6.** Effects of hyper-parameter batch in terms of Recall@20 and Precision@20 in the dataset of MovieLens-1M: (**a**) effects of hyper-parameter *K* in terms of Recall@20; (**b**) effects of hyper-parameter batch in terms of Precision @20.

#### 5.1.6. Optimal Epoch Selection

Epoch, as an important hyper-parameter, determines the number of times the learning algorithm works on the entire training data set. In the process of model training, the complete process of running the model to complete a forward and backward propagation of all the data is called an epoch, and in the process of gradient descent model training, the neural network gradually goes from the unfitted state to the optimally fitted state, and after reaching the optimal state, it enters the overfitting state. If the epoch is 1, it means that each sample in the training dataset has a chance to update the internal model parameters, which will lead to low overall efficiency because of the long loading time of the samples; as such, the epoch should be increased appropriately, but this does not mean the larger, the better. According to the domain-related experience and experimental validation, the final epoch of 230 is chosen as the best value, which results in the best evaluation efficiency of the model. The relevant records are shown in Figure 7.

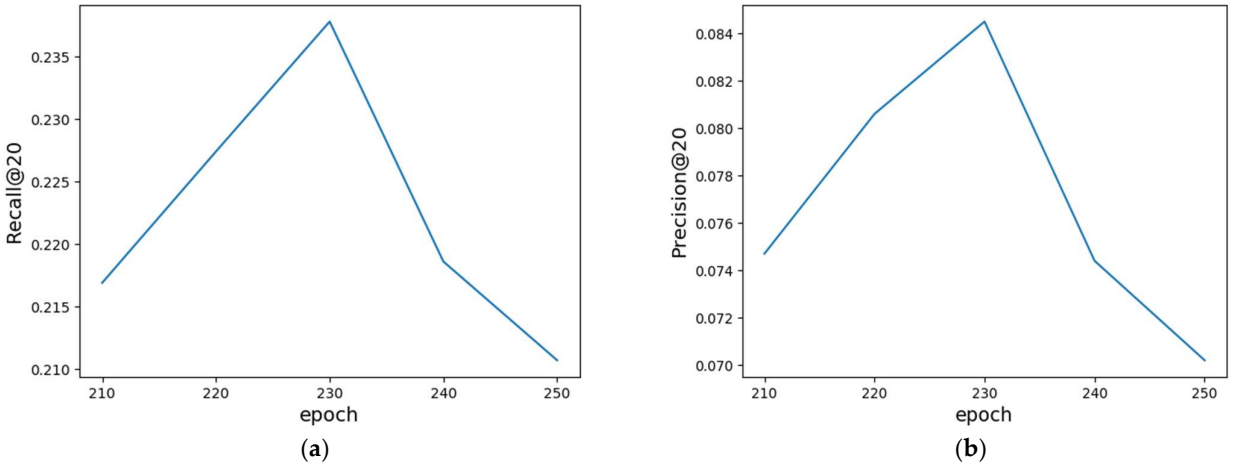



**Figure 7.** Effects of hyper-parameter epoch in terms of Recall@20 and Precision@20 in the dataset of MovieLens-1M: (**a**) effects of hyper-parameter epoch in terms of Recall@20; (**b**) effects of hyper-parameter epoch in terms of Precision @20.

## 6. Conclusions

The existence of implicit connectivity information in the spectral domain is important for establishing connections between users and items in recommender systems. In this study, a graph convolutional collaborative filtering recommendation algorithm based on a random walk and matrix decomposition is proposed to optimize the adjacency matrix by random walk, exploit the higher order connectivity of the graph, explore the user association items, achieve to alleviate the matrix sparsity problem, and then construct a neural network by finite order polynomials, which can reduce the computation and optimize the model performance, and allow the method to learn the potential factors of users and items directly from the spectral domain. The experimental results show that this method outperforms other advanced algorithms. The performance of the model can be further improved in the future by changing the spectral convolution method, such as spectral graph attention network (GAT), graph wavelet neural network (GWNN), and simple spectral graph convolution (S2GC), or adding heterogeneous information and item content of the graph structure.

**Author Contributions:** Conceptualization, H.M. and J.W.; methodology, H.M.; software, J.W. and K.L.; validation, J.W. and K.L.; formal analysis, X.Z. and X.C.; investigation, H.M.; resources, H.M.; data curation, H.M.; writing—original draft preparation, H.M.; writing—review and editing, H.M.; visualization, J.W.; supervision, H.M.; project administration, H.M.; funding acquisition, H.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by the Innovation Project of Postgraduate Education Reform of Liaoning University of Technology (YJG2021020), Liaoning Natural Science Foundation Mentoring Program Project (2019-ZD-0700), Liaoning Education Department Scientific Research Project (JZL202015404, LJKZ0625) and Liaoning Higher Education Innovation Talent Support Project (LR2019034).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data used to support the findings of this study are available from the corresponding author upon request.

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

## References


1. Huang, L.W.; Jiang, B.T.; Lu, S.; Liu, Y.; Li, D.Y. A review of deep learning-based recommender systems. *J. Comput. Sci.* **2018**, *41*, 1619–1647.
2. Tansitpong, P. Identifying key drivers in airline recommendations using logistic regression from web scraping. In Proceedings of the 2020 the 3rd International Conference on Computers in Management and Business, Tokyo, Japan, 31 January–2 February 2020; pp. 112–116.
3. Chen, C.; Zhang, M.; Ma, W.; Liu, Y.; Ma, S. Efficient non-sampling factorization machines for optimal context-aware recommendation. In Proceedings of the Web Conference 2020, Taiwan, China, 20–24 April 2020; pp. 2400–2410.
4. Elbadrawy, A.; Karypis, G. Domain-aware grade prediction and top-n course recommendation. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; pp. 183–190.
5. Shen, L.; Liu, F.; Huang, L.; Liu, G.; Zhou, L.; Peng, L. VDA-RWLRLS: An anti-SARS-CoV-2 drug prioritizing framework combining an unbalanced bi-random walk and Laplacian regularized least squares. *Comput. Biol. Med.* **2022**, *140*, 105119. [CrossRef] [PubMed]
6. Mahapatra, D.P.; Triambak, S. Towards predicting COVID-19 infection waves: A random-walk Monte Carlo simulation approach. *Chaos Solit. Fractals* **2022**, *156*, 111785. [CrossRef] [PubMed]
7. Salakhutdinov, R.; Mnih, A.; Hinton, G. Restricted Boltzmann machines for collaborative filtering. In Proceedings of the 24th International Conference on Machine Learning, Corvalis, OR, USA, 20–24 June 2007; pp. 791–798.
8. Zheng, Y.; Tang, B.; Ding, W.; Zhou, H. A neural autoregressive approach to collaborative filtering. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 1 June 2016; pp. 764–773.
9. Wang, J.; Yu, L.; Zhang, W.; Gong, Y.; Zhang, D. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 515–524.
10. Liu, H.; Guo, L.; Li, P.; Zhao, P.; Wu, X. Collaborative filtering with a deep adversarial and attention network for cross-domain recommendation. *Inf. Sci.* **2021**, *565*, 370–389. [CrossRef]
11. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.S. Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 173–182.
12. Sheikh Fathollahi, M.; Razzazi, F. Music similarity measurement and recommendation system using convolutional neural networks. *Int. J. Multimed. Inf. Retr.* **2021**, *10*, 43–53. [CrossRef]
13. Guo, Q.; Yu, Z.; Wu, Y.; Liang, D.; Qin, H.; Yan, J. Dynamic recursive neural network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 5147–5156.
14. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep reinforcement learning: A brief survey. *IEEE Signal Process. Mag.* **2017**, *34*, 26–38. [CrossRef]
15. Zhang, J.; Shi, X.; Zhao, S.; King, I. Star-gcn: Stacked and reconstructed graph convolutional networks for recommender systems. *arXiv* **2019**, arXiv:1905.13129.
16. Wu, L.; Sun, P.; Fu, Y.; Hong, R.; Wang, X.; Wang, M. A neural influence diffusion model for social recommendation. In Proceedings of the 42th International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, 21–25 July 2019; pp. 235–244.
17. Wang, X.; He, X.; Cao, Y.; Liu, M.; Chua, T.S. Kgat: Knowledge graph attention network for recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 950–958.
18. Berg, R.; Kipf, T.N.; Welling, M. Graph convolutional matrix completion. *arXiv* **2017**, arXiv:1706.02263.
19. Fouss, F.; Pirotte, A.; Renders, J.M.; Saerens, M. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. Knowl. Data Eng.* **2007**, *19*, 355–369. [CrossRef]
20. Haveliwala, T.H. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Trans. Knowl. Data Eng.* **2003**, *15*, 784–796. [CrossRef]
21. Zheng, L.; Lu, C.T.; Jiang, F.; Zhang, J.; Yu, P.S. Spectral collaborative filtering. In Proceedings of the 12th ACM Conference on Recommender Systems, Vancouver, BC, Canada, 2–7 October 2018; pp. 311–319.
22. Jais IK, M.; Ismail, A.R.; Nisa, S.Q. Adam optimization algorithm for wide and deep neural network. *Knowl. Eng. Data Sci.* **2019**, *2*, 41–46. [CrossRef]
23. Mukkamala, M.C.; Hein, M. Variants of rmsprop and adagrad with logarithmic regret bounds. In Proceedings of the International Conference on Machine Learning, Sydney, NSW, Australia, 6–11 August 2017; pp. 2545–2553.
24. Kurbiel, T.; Khaleghian, S. Training of deep neural networks based on distance measures using RMSProp. *arXiv* **2017**, arXiv:01911.2017.
25. Harper, F.M.; Konstan, J.A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst. (TIIS)* **2015**, *5*, 1–19. [CrossRef]
26. Brusilovsky, P.; Cantador, I.; Koren, Y.; Kuflik, T.; Weimer, M. Workshop on information heterogeneity and fusion in recommender systems (HetRec 2010). In Proceedings of the Fourth ACM Conference on Recommender Systems, Barcelona, Spain, 26–30 September 2010; pp. 375–376.

27. Wang, X.; He, X.; Wang, M.; Feng, F.; Chua, T.S. Neural graph collaborative filtering. In Proceedings of the 42th International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, 21–25 July 2019; pp. 165–174.
28. He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; Wang, M. Lightgcn: Simplifying and powering graph convolution network for recommendation. In Proceedings of the 43th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, China, 25–30 July 2020; pp. 639–648.
29. Wang, X.; Jin, H.; Zhang, A.; He, X.; Xu, T.; Chua, T.S. Disentangled graph collaborative filtering. In Proceedings of the 43th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, China, 25–30 July 2020; pp. 1001–1010.