How Powerful is Graph Filtering for Recommendation

Shaowen Peng peng.shaowen@naist.ac.jp NARA Institute of Science and Technology Nara, Japan

> Kazunari Sugiyama sugiyama-k@g.osaka-seikei.ac.jp Osaka Seikei University Osaka, Japan

ABSTRACT

It has been shown that the effectiveness of graph convolutional network (GCN) for recommendation is attributed to the spectral graph filtering. Most GCN-based methods consist of a graph filter or followed by a low-rank mapping optimized based on supervised training. However, we show two limitations suppressing the power of graph filtering: (1) Lack of generality. Due to the varied noise distribution, graph filters fail to denoise sparse data where noise is scattered across all frequencies, while supervised training results in worse performance on dense data where noise is concentrated in middle frequencies that can be removed by graph filters without training. (2) Lack of expressive power. We theoretically show that linear GCN (LGCN) that is effective on collaborative filtering (CF) cannot generate arbitrary embeddings, implying the possibility that optimal data representation might be unreachable.

To tackle the first limitation, we show close relation between noise distribution and the sharpness of spectrum where a sharper spectral distribution is more desirable causing data noise to be separable from important features without training. Based on this observation, we propose a generalized graph normalization (G²N) with hyperparameters adjusting the sharpness of spectral distribution in order to redistribute data noise to assure that it can be removed by graph filtering without training. As for the second limitation, we propose an individualized graph filter (IGF) adapting to the different confidence levels of the user preference that interactions can reflect, which is proved to be able to generate arbitrary embeddings. By simplifying LGCN, we further propose a simplified graph filtering for CF (SGFCF)¹ which only requires the top-K singular values for recommendation. Finally, experimental results on four datasets with different density settings demonstrate the effectiveness and efficiency of our proposed methods.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06...\$15.00
https://doi.org/XXXXXXXXXXXXXX

Xin Liu xin.liu@aist.go.jp National Institute of Advanced Industrial Science and Technology Tokyo, Japan

> Tsunenori Mine mine@m.ait.kyushu-u.ac.jp Kyushu University Fukuoka, Japan

CCS CONCEPTS

• Information systems \rightarrow Recommender systems.

KEYWORDS

Recommender System, Collaborative Filtering, Graph Convolutional Network

ACM Reference Format:

1 INTRODUCTION

Personalized recommendations have been widely applied to e-commerce, social media platforms, online video sites, etc., and has been indispensable to enrich people's daily life by offering the items user might be interested in based on the data such as user-item interactions, reviews, social relations, temporal information, etc. Among various recommendation scenarios, we focus on collaborative filtering (CF), a fundamental task for recommender systems. Conventional CF methods such as matrix factorization (MF) [22] characterizes users and items as low dimensional vectors and predict the rating via the inner product between the corresponding embedding vectors. Subsequent works replace the linear design of MF with other advanced algorithms such as neural networks [9, 15], attention mechanisms [5, 20], transformer [23, 40], diffusion models [43], etc. to model complex user-item relations.

While the aforementioned advanced recommendation algorithms show superior non-linear ability to model user-item relations, their performance are unstable due to the data sparsity issue in recommendation datasets. Graph Convolutional Networks (GCNs) recently have shown great potential in recommender systems due to the ability of capturing higher-order neighbor signals that can augment the training data to alleviate the sparsity issue. Early GCN-based methods adapt classic GCNs such as vanilla GCN [21] and GraphSage [13] to recommendation [44, 49, 51]. Subsequent works empower GCN by incorporating other advanced algorithms such as contrastive learning [47], learning in hyperbolic space [7, 41], disentangled representation learning [45], etc., slimming GCN model architectures to improve efficiency and scalability [6, 14, 30], and study the effectiveness of GCN [29, 37].

 $^{^{1}}https://github.com/tanatosuu/sgfcf \\$

Recent studies have shown that the effectiveness of GCN for recommendation is mainly attributed to the spectral graph filtering which emphasizes important features (i.e., low frequencies) and filters out useless information. Most existing graph filtering designs can be classified to two categories: (1) Repeatedly propagating the node embeddings across the graph where the embeddings are optimized based on supervisory signals. This type of methods is actually equivalent to a low pass filter followed by a low-rank mapping [12, 14, 29] (see Equation (3)). (2) A simple graph filter without model training [25, 37]. This type of methods only relies on the graph filters to denoise. We can see these two kinds of methods are actually contradictory: the type (2) methods implies that data noise is only distributed in certain frequencies that can be simply removed by graph filters without training, which is contrary to the type (1), that we need model training to further remove data noise. However, we show that neither of them are perfect by pointing out two limitation suppressing the power of graph filtering. Firstly, both of the two designs lack generality. We find that noise distribution varies on datasets with different densities. Particularly, graph filters show poor performance on sparse datasets where data noise is scattered across all frequencies as they denoise by masking certain frequencies while fail to remove intrinsic noise in the features. On the other hand, despite the superior ability of supervised training learning from data polluted by noise, it results in worse performance on dense datasets on which the noise is concentrated in middle frequencies that can be simply removed by graph filters. Moreover, most effective GCN-based methods are basically linear GCNs (LGCNs) without non-linearity. We theoretically show that they are incapable of generating arbitrary embeddings with multi-dimensions, implying the possibility that they cannot generate desirable user/item representations and demonstrating the lack of expressive power.

To tackle the first limitation, we further show the close relation between noise distribution and the sharpness of spectral distribution, that a sharper distribution is more desirable on which noise and important features are separable by graph filtering without supervised training. Based on this observation, we propose a generalized graph normalization (G²N) to adjust the sharpness of spectrum via hyperparameters. As a result, data noise is redistributed through G²N, making the graph filtering generalizable on datasets with different densities. To tackle the second limitation, we propose an individualized graph filter (IGF) which is proved to generate arbitrary embeddings. Specifically, considering interactions do not equally reflect user preference (i.e., more (less) similar of the interacted items implies a higher (lower) consistency between the user's future and past behaviour), the proposed IGF emphasizes different frequencies based on the distinct confidence levels of user preference that interactions can reflect. Finally, by simplifying LGCN, we propose a simplified graph filtering for CF (SGFCF) only requiring the top-*K* singular values. Our main contributions can be summarized as follows::

• We point out two limitations suppressing the power of graph filtering for recommendation: (1) The lack of generality due to the the performance inconsistency of graph filters and supervised training on data with different densities and (2) The lack of expressive power of LGCN that is effective for CF.

- We propose a generalized graph normalization to tackle the first limitation, which redistributes data noise by adjusting the sharpness of spectrum, enabling the graph filtering to denoise without training on datasets with different densities.
- We propose an individualized graph filtering to adapt to distinct confidence levels of user preference that interactions can reflect. It is proved to generate arbitrary data representations thus solves the second limitation.
- Extensive experimental results on four datasets with different density settings demonstrate the efficiency and effectiveness of our proposed method.

2 PRELIMINARIES

We first introduce a commonly used GCN-learning paradigm for CF. Given an interaction matrix with implicit feedbacks containing $|\mathcal{U}|$ users and |I| items: $\mathbf{R} \in \{0,1\}^{|\mathcal{U}| \times |I|}$, we can define a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the node set contains all users and items: $\mathcal{V} = \mathcal{U} + I$, the edge set contains the user-item pairs with interactions: $\mathcal{E} = \mathbf{R}^+$ where $\mathbf{R}^+ = \{r_{ui} = 1 | u \in \mathcal{U}, i \in I\}$. For simplicity, we let $|\mathcal{U}| + |I| = n$. Then, we can define the corresponding adjacency matrix \mathbf{A} of \mathcal{G} and formulate the updating rules as:

$$\mathbf{H}^{(l+1)} = \sigma \left(\hat{\mathbf{A}} \mathbf{H}^{(l)} \mathbf{W}^{(l+1)} \right), \tag{1}$$

where $\sigma(\cdot)$ is an activation function, $\hat{\bf A}$ is a symmetric normalized adjacency matrix defined as follows:

$$\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{0} & \hat{\mathbf{R}} \\ \hat{\mathbf{R}}^T & \mathbf{0} \end{bmatrix},\tag{2}$$

where $\hat{\mathbf{R}} = \mathbf{D}_U^{-\frac{1}{2}} \mathbf{R} \mathbf{D}_I^{-\frac{1}{2}}$ is a normalized interaction matrix, \mathbf{D}_U and \mathbf{D}_I are matrices with diagonal elements representing user and item degrees, respectively. The initial state is the stacked user/item embeddings $\mathbf{H}^{(0)} = \mathbf{E} \in \mathbb{R}^{n \times d}$, where each user u and item i are represented as learnable low-dimensional vectors. $\mathbf{W}^{(l+1)} \in \mathbb{R}^{d \times d}$ is a linear transformation. It has been shown that the following linear GCN (LGCN) removing the activation function and linear transformation is effective for CF [6, 14], with the final representations O generated as:

$$\mathbf{O} = g\left(\hat{\mathbf{A}}\right)\mathbf{E},\tag{3}$$

where $g(\hat{\mathbf{A}})$ is usually defined as a polynomial graph filter:

$$g\left(\hat{\mathbf{A}}\right) = \sum_{l=0}^{L} \theta_l \hat{\mathbf{A}}^l = \mathbf{V} diag\left(\sum_{l=0}^{L} \theta_l \lambda_k^l\right) \mathbf{V}^T, \tag{4}$$

where $\{\theta_0,\cdots,\theta_L\}$ is a set of parameters; λ_k and **V** are an eigenvalue and the stacked eigenvectors, respectively; $diag(\cdot)$ is a diagonalization operator. The rating is predicted as:

$$\hat{r}_{ui} = \mathbf{o}_{u}^{T} \mathbf{o}_{i} \tag{5}$$

DEFINITION 1. (Graph Frequency). Given the eigenvalue and vector pairs $(\lambda_k, \mathbf{v}_k)$ of $\hat{\mathbf{A}}$ where $\lambda_k \in [-1, 1]$, the graph frequency is defined based on the variation of a signal on the graph $\|\mathbf{v}_k - \hat{\mathbf{A}}\mathbf{v}_k\| = 1 - \lambda_k \in [0, 2]$. We call components with small variations low frequencies, components with high variations high frequencies.

Table 1: The accuracy (nDCG@10) of SGF and LGCN with different density settings.

Datasets	Methods	80%	60%	40%	20%
CiteULike	SGF	0.2267	0.2222	0.1661	0.0859
	LGCN, $d = 64$	0.1610	0.2023	0.2120	0.1267
	LGCN, $d = 128$	0.1699	0.2067	0.2157	0.1347
	LGCN, $d = 256$	0.1701	0.2110	0.2204	0.1390
	SGF	0.0816	0.1008	0.1106	0.0629
Pinterest	LGCN, $d = 64$	0.0707	0.0912	0.1204	0.1317
Pinterest	LGCN, $d = 128$	0.0706	0.0924	0.1218	0.1338
	LGCN, $d = 256$	0.0699	0.0928	0.1240	0.1341

According to Definition 1, low (high) frequencies emphasize the similarity (dissimilarity) between nodes and their neighborhood. We can adjust the weight of different frequencies via $g(\lambda_k)$ as $g(\hat{\mathbf{A}}) = \sum_k g(\lambda_k) \mathbf{v}_k \mathbf{v}_k^T$. It has been shown that low frequencies are significantly contributive to recommendation [29, 37], and most GCN-based methods can be classified to two categories: (1) a low pass filter followed by a linear mapping and optimization (*i.e.*, LGCN) and (2) a simple graph filter without training [37]. The learning process can be formulated as follows:

$$\mathcal{G} \xrightarrow{\text{Spectrum}} \mathbf{V} \in \mathbb{R}^{n \times n} \xrightarrow{g(\lambda_k)} \mathbf{V}^{(K)} \in \mathbb{R}^{n \times K} \xrightarrow{\mathbf{E}} \mathbf{O} \in \mathbb{R}^{n \times d}, \quad (6)$$

where $\mathbf{V}^{(K)}$ is the top-K low frequency components. Here, we raise two questions on existing works: (1) The underlying assumption of the second type of methods is that data noise is only concentrated in certain frequencies that does not pollute important features (i.e., $\mathbf{V}^{(K)}$), which is contrary to the first type of methods that further training is required to denoise $\mathbf{V}^{(K)}$. Since the two types of methods seem to be contradictory, are they general on different datasets? (2) Despite the effectiveness and simplicity of LGCN, it also has a weakened expressive power compared with vanilla GCN, is it capable of generating desirable data representations? We will answer the two questions in Section 3.

3 ANALYSIS ON GRAPH FILTERING

In this section, we evaluate existing graph filtering designs in terms of generality and expressive power. In the light of the effectiveness of LGCN for CF, our analysis is mainly based on LGCN and a simple graph filter (SGF) $g(\hat{\mathbf{A}})$. The difference between the two models lies in the necessity of model training.

3.1 Generality

3.1.1 Performance Inconsistency under Different Densities. We first evaluate graph filtering on datasets with different densities. We change the data density by adjusting the training ratio x% (the remaining is used as the test set), where $x = \{80, 60, 40, 20\}\%$. We choose an extensively used setting for CF with $\theta_0 = , \cdots , = \theta_L$ [14] where $g(\hat{\mathbf{A}})$ is a low pass filter. We compare LGCN and SGF on two datasets and report the results in Table 1. We observe that the performance of SGF and LGCN consistently decrease and increase as the training data is sparser, respectively. Particularly, SGF tends to be effective on the dense data (e.g., x = 80% and 60%) and significantly outperforms LGCN, indicating the uselessness of

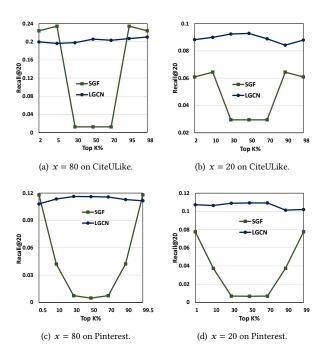


Figure 1: The accuracy (Recall@20) of SGF and LGCN when only considering top-K% low frequencies.

model training. On the other hand, the positive effect of training can be identified on the sparse data (e.g., x = 20% and 40%) as LGCN shows better performance. In addition, despite the improvement brought by increasing the embedding size, the performance tends to converge and still underperforms SGF on the dense data.

3.1.2 Noise Distribution Varies on Densities. The above observations show the inconsistency and lack of generality of LGCN and SGF. Since SGF is a low pass filter, the poor performance on sparse data leads to a reasonable assumption that low frequencies might be polluted by noise. To verify this assumption, we conduct frequency analysis with the following filter:

$$g\left(\hat{\mathbf{A}}\right) = \mathbf{V}^{(K)} \mathbf{V}^{(K)^T},\tag{7}$$

We investigate the noisiness of certain frequencies by increasing K from 0 and observing how accuracy changes after introducing certain frequencies. As the original polynomial filter $\sum_{l}^{L} \lambda_{k}^{l}$ emphasizes more on the lower frequencies leading to biased results, we choose a uniform filter here. We focus on x=80% and x=20% as SGF (LGCN) is most (least) and least (most) effective, respectively. We observe the followings from the results shown in Figure 1:

- On x = 80% ((a) and (c)), the accuracy of SGF increases then significantly drops and rises again as K increases, and the best performance outperforms LGCN, showing that the noise is concentrated in middle frequencies that can be removed by SGF.
- On x = 20% ((b) and (d)), SGF underperforms LGCN as K increases, indicating that the noise is distributed across all frequencies and pollutes the important features as well.

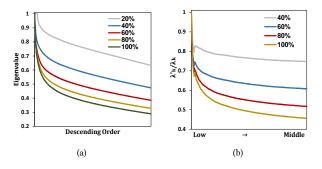


Figure 2: (a) Spectral distribution of CiteULike with different density settings. (b) the eigenvalue ratio (λ'_k/λ_k) of $x = \{40, 60, 80, 100\}$ (λ'_k) to x = 20 (λ_k) .

- The stable performance of LGCN when incorporating noisy frequencies demonstrates the ability of model training of learning from noise. While it tends to be redundant and results in worse performance when the features are not polluted.
- The superior performance when only incorporating low frequencies shows that important graph features are distributed in low frequencies on both settings.

The above observations verify our assumption that the poor performance of SGF on sparse data is attributed to the noise distribution that is across all frequencies. More importantly, both graph filters and supervised training lack generality due to their inconsistent performance on data with different densities, thus a more generic design is required. In addition, we notice that the performance of SGF is highly symmetric with respect to the middle frequency (*i.e.*, $\lambda_{n/2}=0$) which is due to the following theorem and corollary:

Theorem 1. Given P and Q as the left and right singular vectors of $\hat{\mathbf{R}}$, we have the following relation:

$$\mathbf{V} = \begin{bmatrix} \mathbf{P} & \mathbf{P} \\ \mathbf{Q} & -\mathbf{Q} \end{bmatrix} / \sqrt{2}. \tag{8}$$

Particularly, given a eigenvalue $\lambda_k > 0$ with eigenvector $\mathbf{v}_k = [\mathbf{p}_k, \mathbf{q}_k]^T$, there always exists $a - \lambda_k$ with corresponding eigenvector $[\mathbf{p}_k, -\mathbf{q}_k]^T$.

Corollary 1. Let $r_{ui}^{(K)}$ be the rating estimated based on the representation of Equation (7), we have the following relation:

$$r_{ui}^{(K)} = r_{ui}^{(n-K)}. (9)$$

Theorem and Corollary 1 show that low and high frequency are actually highly symmetric due to the bipartivity of \mathcal{G} . Therefore, we can only focus on the low and middle frequencies with $\lambda_k \geq 0$.

3.2 Expressive Power

Despite many existing works that can be summarized as LGCN variants[12, 14, 29] choosing different filters to model user-item relations, its expressive power is still unknown.

Theorem 2. Assuming $\hat{\mathbf{A}}$ has no repeated eigenvalues, then LGCN can produce arbitrary embeddings when d=1. For d>1, each

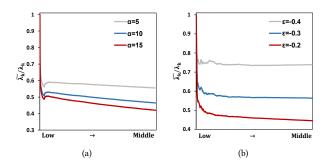


Figure 3: Eigenvalue ratio of $\tilde{\mathbf{A}}$ to $\hat{\mathbf{A}}$ with varying α and ϵ on CiteULike.

dimension requires an individual filter:

$$\mathbf{O} = \mathbf{V} \left(\mathbf{B} \Theta \right) \odot \mathbf{V}^T \mathbf{E}, \tag{10}$$

where $\mathbf{B} \in \mathbb{R}^{n \times n}$ is a full-rank matrix with $\mathbf{B}_{kl} = \lambda_k^l$, $\Theta \in \mathbb{R}^{n \times d}$ is a parameter matrix. Equation (10) degenerates to LGCN when d = 1.

Theorem 2 shows that a shared global filter $[\theta_1,\cdots,\theta_n]$ cannot generate arbitrary multidimensional embeddings, and it is apparent one-dimensional representation is not enough to characterize users and items, which also demonstrates the incapability of LGCN to generate the optimal representations and shows the poor expressive power despite its effectiveness. Theorem 2 can be directly applied to SGF by setting d=n. Since the output has n dimensions, it requires n different filters to fit arbitrary embeddings.

4 METHODOLOGY

4.1 Sharpness of Spectrum Matters

Ideally, if the spectrum is a low pass filter defined as follows:

$$\lambda_k = \begin{cases} \gg 0 & k \le K \\ \approx 0 & k > K, \end{cases} \tag{11}$$

where $\lambda_1 > \cdots > \lambda_{n/2}$. Most of the energy is concentrated in the top-K low frequencies while the information distributed in the middle frequencies is trivial and noisy to data representations. In this case, the important features and noise can be separated by graph filtering without training. Inspired by this observation, we report the eigenvalue distribution with different density settings in Figure 2 (a). We observe that the denser data on which the noise and important features tend to be separable has a sharper spectrum. Particularly, in Figure 2 (b), we can see that the eigenvalue closer to the middle frequency tends to drop faster than the low frequency. This means that the energy of middle frequencies is close to 0 while low frequencies stay important to data representation, causing the important features and noise to be distributed in different frequencies instead of mixed up together. Therefore, a sharper spectral distribution is more desirable as it is closer to an ideal low pass filter. Furthermore, consider a rank-K approximation consisting of the top-K low frequencies: $\hat{\mathbf{A}}_K = \sum_{k=1}^K \lambda_k \mathbf{v}_k \mathbf{v}_k^T$, then we have: Theorem 3. Let $\tilde{\mathbf{A}}$ be a normalized adjacency matrix of \mathcal{G} with a sharper spectral distribution than $\hat{\mathbf{A}} \colon 1 = \frac{\tilde{\lambda}_1}{\lambda_1} \ge \cdots \ge \frac{\tilde{\lambda}_K}{\lambda_K} > \frac{\tilde{\lambda}_{K+1}}{\tilde{\lambda}_{K+1}} \ge \cdots \ge \frac{\tilde{\lambda}_{n/2}}{\lambda_{n/2}}$. Then $\tilde{\mathbf{A}}_K$ is a better approximation than $\hat{\mathbf{A}}_K$ with the following equation measuring the quality of an approximation:

$$\mathbf{appro}\left(\tilde{\mathbf{A}}_K\right) = \frac{\left|\tilde{\mathbf{A}}_K\right|_F^2}{\left|\tilde{\mathbf{A}}\right|_F^2},\tag{12}$$

where $|\cdot|_F$ stands for the Frobenius norm.

Equation (12) measures to what extent $\tilde{\mathbf{A}}_K$ can recover $\tilde{\mathbf{A}}$ where $\operatorname{\mathbf{appro}}\left(\tilde{\mathbf{A}}_K\right) \in [0,1]$. As the spectral distribution becomes sharper, $\tilde{\mathbf{A}}_K$ is a better approximation indicating that more important information is concentrated in the top-K low frequencies while the remaining information is more trivial that is distributed in the middle frequencies, causing them to be more separable.

4.2 Generalized Graph Normalization (G²N)

The analysis in Section 4.1 provides a solution to tackle the dilemma mentioned in Section 3.1, that graph filters and supervised training perform inconsistently on data with different density settings which is attributed to the varied noise distribution. If we are able to generate a desirable spectrum in a way that clearly differentiates important features from noise, it becomes possible to depend exclusively on graph filters for recommendation. Since the spectrum is closely related to how we normalize the graph, we study what graph normalization leads to a desirable spectrum in this subsection.

According to Definition 1, we know that putting more emphasis on low frequencies leads to a higher similarity between nodes and neighborhood. Thus, it is reasonable to assume that more energy is concentrated in the low frequencies if we increase the similarity by modifying the graph normalization, leading to a sharper spectrum. Since $g(\lambda_k)$ shares the same eigenspace with λ_k , we can simply define the similarity on $g(\hat{\mathbf{A}}) = \hat{\mathbf{A}}$:

$$\hat{\mathbf{A}}_{u*} \sum_{v \in \mathcal{N}_{u}^{2}} \hat{\mathbf{A}}_{v*}^{T} = (\mathbf{V}_{u*} \odot \lambda) \mathbf{V}^{T} \left(\left(\sum_{v} \mathbf{V}_{v*} \odot \lambda \right) \mathbf{V}^{T} \right)^{T}$$

$$= (\mathbf{V}_{u*} \odot \lambda) \left(\sum_{v} \mathbf{V}_{v*} \odot \lambda \right)^{T}, \tag{13}$$

where λ is a vector containing all eigenvalues, V_{u*} refers to the u-th row of V, \odot is the operator for element-wise multiplication, \mathcal{N}_u^2 stands for the second-order neighborhood as only the second-order neighbors have similarity with u (there is no similarity between a user and an item).

DEFINITION 2. (Variation on the second-order Graph). The variation of the eigenvectors on the second-order graph is defined as:

$$\|\mathbf{v}_k - \hat{\mathbf{A}}^2 \mathbf{v}_k\| = 1 - \lambda_k^2 \in [0, 1].$$
 (14)

Interpretation of Equation (13). Definition 2 measures the difference between the signal samples of eigenvectors at each node (\mathbf{V}_{uk}) and at its second-order neighbors $(\sum_v \mathbf{V}_{vk})$. Intuitively, \mathbf{v}_k with $|\lambda_k| \to 1$ implies that the nodes are similar to their second-order neighborhood: $|\mathbf{V}_{uk} - \sum_v \mathbf{V}_{vk}| \to 0$, while the middle frequency with $|\lambda_k| \to 0$ emphasizes the difference between them.

Consider λ as a band-pass filter, if the node similarity increases, the components with $|\lambda_k| \to 1$ and $|\lambda_k| \to 0$ should be correspondingly emphasized and suppressed to make the equation hold, respectively, leading to a sharper spectrum. In other words, the sharpness of the spectral distribution is closely related to the average node similarity defined on the normalized adjacency matrix.

Then, our question is transformed to: how do we increase the node similarity through a new graph normalization? The original setting is defined as $\hat{\mathbf{A}}_{ui} = \frac{1}{\sqrt{d_u}\sqrt{d_i}}$. Here, we define a renormalized adjacency matrix with $\tilde{\mathbf{A}}_{ui} = w(d_u)w(d_i)$, and the average similarity between users and items can be defined as follows:

$$SIM_{U} = \frac{\sum_{u,v \in \mathcal{U}} \hat{\mathbf{A}}_{u*} \hat{\mathbf{A}}_{v*}^{T}}{|\mathcal{U}|^{2}} = \sum_{i \in I} 2w(d_{i})^{2} \frac{\sum_{u,v \in \mathcal{N}_{i}} w(d_{u})w(d_{v})}{|\mathcal{U}|^{2}},$$

$$SIM_{I} = \frac{\sum_{i,j \in I} \hat{\mathbf{A}}_{*i}^{T} \hat{\mathbf{A}}_{*j}}{|I|^{2}} = \sum_{u \in \mathcal{U}} 2w(d_{u})^{2} \frac{\sum_{i,j \in \mathcal{N}_{u}} w(d_{i})w(d_{j})}{|I|^{2}},$$
(15)

where N_u/N_i is the set of first-hop neighborhood of u/i. We can see that the user/item with a higher degree has a larger impact on the average similarity (proportional to d_u^2/d_i^2), leading to the conclusion that the higher weights over the high-degree nodes results in higher node similarity. Based on the original design, we can propose two new designs with higher weights over high-degree nodes: (1) $w(d_u) = \frac{1}{\sqrt{d_u + \alpha}}$, and (2) $w(d_u) = d_u^\epsilon$, and propose a generalized graph normalization as follows:

$$\tilde{\mathbf{A}} = (\mathbf{D} + \alpha \mathbf{I})^{\epsilon} \mathbf{A} (\mathbf{D} + \alpha \mathbf{I})^{\epsilon}, \qquad (16)$$

where $\alpha \ge 0$, $\epsilon \in [-0.5, 0]$. $\tilde{\mathbf{A}} = \hat{\mathbf{A}}$ when $\alpha = 0$ and $\epsilon = -0.5$.

Theorem 4. Given $\tilde{\lambda}_k$ as the eigenvalue of \tilde{A} , we have:

$$d_{max} (d_{max} + \alpha)^{2\epsilon} \lambda_k \ge \tilde{\lambda}_k \ge d_{min} (d_{min} + \alpha)^{2\epsilon} \lambda_k, \quad (17)$$

where d_{min} and d_{max} are the min and max node degree, respectively.

Particularly, increasing α and ϵ shrinks and scales the eigenvalue, respectively, making the spectrum not normalized any more. As we focus on the sharpness of the spectrum, we normalize $\tilde{\lambda}_k$ and visualize $\tilde{\lambda}_k/\lambda_k$ to observe how our proposed G^2N adjusts the spectrum shown in Figure 3. We observe that the eigenvalue closer to the middle frequency drops more quickly, while the one closer to the low frequency tends to remain unchanged, and such a trend is more obvious as α or ϵ increases. The results in Figure 3 indicate that G^2N can generate a desirable spectrum which is more equivalent to an ideal low pass filter assuring that data noise and important features are linearly separable without further training.

4.3 Individualized Graph Filtering (IGF)

Owing to G^2N , we can only rely on graph filters for recommendation. By applying the results in Theorem 2, we can empower SGF via the following enhanced model which is capable of generating arbitrary embeddings:

$$\mathbf{O} = \mathbf{V} \left(\mathbf{B} \Theta \right) \odot \mathbf{V}^{T} = \left(\mathbf{V} \odot \begin{bmatrix} g_{1}(\lambda_{k}) \\ \vdots \\ g_{n}(\lambda_{k}) \end{bmatrix} \right) \left(\mathbf{V} \odot \begin{bmatrix} g_{1}(\lambda_{k}) \\ \vdots \\ g_{n}(\lambda_{k}) \end{bmatrix} \right)^{T}, \quad (18)$$

$$\mathbf{B} \Theta = \mathbf{B} \left[\Theta_{1}, \cdots, \Theta_{n} \right] = \left[g_{1}^{2}(\lambda_{k}), \cdots, g_{n}^{2}(\lambda_{k}) \right].$$

Table 2: Statistics of datasets.

Datasets	#User	#Item	#Interactions	Density%
CiteULike	5,551	16,981	210,537	0.223
Pinterest	37,501	9,836	1,025,709	0.278
Yelp	25,677	25,815	731,672	0.109
Gowalla	29,858	40,981	1,027,370	0.084

Without loss of generality, we assume the filter $B\Theta$ is non-negative. Here, each row of V can be considered as a user/item feature vector, and $[g_1(\lambda_k), \cdots, g_n(\lambda_k)]^T$ is the corresponding individualized filters related to users/items. There are different ways to implement the individualized filter, we present our solution as follows.

Definition 3. a (an) user's/item's homophilic ratio measuring the similarity of the the past interactions is defined as follows:

$$homo(u) = \frac{\sum_{i,j \in \mathcal{N}_{u}} \mathbb{1}^{\mathcal{D}_{\mathcal{G}/u}(i,j) < \delta}}{|\mathcal{N}_{u}|^{2}},$$

$$homo(i) = \frac{\sum_{u,v \in \mathcal{N}_{i}} \mathbb{1}^{\mathcal{D}_{\mathcal{G}/i}(u,v) < \delta}}{|\mathcal{N}_{i}|^{2}},$$
(19)

where $\mathcal{D}_{\mathcal{G}}(\cdot,\cdot)$ is the graph distance², $\mathcal{D}_{\mathcal{G}/u}(i,j)$ measures the distance between i and j which does not pass through u, and $\mathbb{1}$ is an indicator function producing 1 if two nodes are close enough (i.e., $< \delta$).

Intuitively, if a user's interactions are similar (*i.e.*, high homophilic ratio), then it is possible his/her future behaviour is consistent with the past interactions. While it is hard to rely on a user's past interactions if they are quite different (low homophilic ratio). Given that different frequencies emphasize the similarity between nodes and neighborhood with different degrees, it is reasonable to implement the individualized filter based on the homophilic ratio. After evaluating multiple graph filters, we choose a monomial filter: $g(\lambda_k) = \lambda_k^{\beta}$, map the homophilic ratio to $[\beta_1, \beta_2]$ via a linear function, where the min and max of the homophilic ratio is mapped to β_1 and β_2 , respectively. Then, the individualized filter is implemented as $g_u(\lambda_k) = \lambda_k^{\beta_u}$, where $\beta_u \in [\beta_1, \beta_2]$ is determined by $\mathbf{homo}(u)$.

4.4 Simplified Graph Filtering for CF (SGFCF)

Theorem 5. The prediction matrix of SGF can be formulated as follows:

$$\mathbf{O}_{U}\mathbf{O}_{I}^{T} = (\mathbf{P}diag(\psi(\sigma_{k})))(\mathbf{Q}diag(\omega(\sigma_{k})))^{T}, \tag{20}$$

where σ_k is the singular value of $\hat{\mathbf{R}}, \psi(\sigma_k) = \sum_{l=\{0,2,\cdots\}} \sigma_k^l, \omega(\sigma_k) = \sum_{l=\{1,3,\cdots\}} \sigma_k^l$.

Here, $\psi(\sigma_k)$ and $\omega(\sigma_k)$ are equivalent to low pass filters making no difference. By applying G^2N and IGF, we can design our SGFCF by modifying Equation (20) as follows:

$$\mathbf{O}_{U}\mathbf{O}_{I}^{T} = \left(\tilde{\mathbf{P}}^{(K)} \odot G(\tilde{\sigma}_{k})\right) \left(\tilde{\mathbf{Q}}^{(K)} \odot G(\tilde{\sigma}_{k})\right) + \gamma \tilde{\mathbf{R}} \tilde{\mathbf{R}}^{T} \tilde{\mathbf{R}}, \tag{21}$$

where $G(\tilde{\sigma}_k) = [g_1(\tilde{\sigma}_k), \cdots, g_n(\tilde{\sigma}_k)]^T$; $\tilde{\mathbf{P}}^{(K)}$ and $\tilde{\mathbf{Q}}^{(K)}$ are the top-K left and right singular vectors corresponding to low frequencies, $\tilde{\sigma}_k$ is the singular value of $\tilde{\mathbf{R}} = (\mathbf{D}_U + \alpha \mathbf{I})^{\epsilon} \mathbf{R} (\mathbf{D}_I + \alpha \mathbf{I})^{\epsilon}$, respectively. In practice, we notice that non-low frequencies are not completely noisy, thus we add a term $\bar{\mathbf{R}}\bar{\mathbf{R}}^T\bar{\mathbf{R}}$ containing all frequencies.

4.5 Discussion

Compared with existing GCN-based methods, our proposed SGFCF mainly differs from them in three aspects: (1) We provide a closed-form solution with complexity only as: $O(K |\mathbf{R}^+| + K^2 |\mathcal{U}| + K^2 |\mathcal{I}|)$. (2) Our method is generalizable on datasets with different densities. (3) Compared with the methods that can be summarized as LGCN, our method is proven to have stronger expressive power which is capable of generating arbitrary embeddings. Particularly, compared with non-parametric methods such as GFCF [37], our superiority comes from two aspects: (1) GFCF is equivalent to a low pass filter which does not consider the varied noise distribution on data with different densities, thus is not generalizable. (2) It can be summarized as a LGCN showing poor expressive power. We will empirically demonstrate our superiority in Section 5.

5 EXPERIMENT

5.1 Experimental Setup

5.1.1 Datasets and Evaluation Metrics. We evaluate our method on four datasets in this work, the statistics are summarized in Table 2. CiteULike³ is collected from a social bookmarking service CiteULike which allows users to bookmark and share research articles; Pinterest [15] is constructed for evaluating content-based image recommendation; Yelp [16] is from the Yelp Challenge data; Gowalla [44] is a check-in dataset which records the locations users have visited. We focus on x = 80% and x = 20%, randomly select 5% as validation set, and leave the remaining for test. We adopt Recall and nDCG [18], two widely used evaluation metrics for personalized recommendation. The recommendation list is generated by ranking unobserved items and truncating at position k = 10.

5.1.2 Implementation. We use stochastic gradient descent (SGD) as the optimizer for training-based models. The embedding size d is set to 64, the regularization rate γ is set to 0.01 on all datasets, the learning rate is tuned with step size 0.1, the model parameters are initialized with Xavier initialization [11] and the batch size is set to 256. $\alpha \ge 0$ and $\epsilon \in [-0.5, 0]$ are tuned with step size 1 and 0.02, respectively. Other hyperparameters in this work are all tuned with step size 0.1. We set $\delta = 2$ which is the smallest graph distance between homogeneous nodes; we use a monomial filter: $g(\lambda_k) = \lambda_k^{\beta}$, $\beta_1 \le \beta$ and $\beta_2 \ge \beta$ are tuned after determining the best β .

5.1.3 Baselines. We compare our proposed methods with competitive baselines which can be categorized to training-based and non-parametric (*i.e.*, training-free) methods. For training-based methods, we choose BPR [33] and DirectAU [42] implemented on MF, and seven GCN-based methods: LightGCN [14], SGL-ED [47], LightGCL [3], XSimGCL [50], DCCF [32], GDE [29], and JGCF [12]. Additionally, we adopt four non-parametric methods: EASE [39], GF-CF [37], PGSP [25], and BPSM [8]. Note that the hyperparameters are properly set after conducting tuning for them on the datasets used in this work.

 $^{^2\}mathrm{The}$ number of edges in the shortest path between two nodes.

 $^{^3}$ https://github.com/js05212/citeulike-a

			Citel	JLike			Ye	lp	
		x=8	80%	x=2	20%	x=8	30%	x =2	20%
Metho	Method		Recall	nDCG	Recall	nDCG	Recall	nDCG	Recall
	BPR	0.1620	0.1778	0.0674	0.0621	0.0487	0.0607	0.0635	0.0600
	DirectAU	0.2102	0.2260	0.1348	0.1280	0.0721	0.0872	0.0857	0.0839
	LightGCN	0.1610	0.1777	0.1273	0.1206	0.0572	0.0721	0.0751	0.0725
	SGL-ED	0.1890	0.2117	0.1217	0.1167	0.0676	0.0837	0.0817	0.0784
Training-Based	XSimGCL	0.2024	0.2229	0.1360	0.1289	0.0691	0.0847	0.0837	0.0809
	LightGCL	0.2096	0.2214	0.1270	0.1224	0.0673	0.0836	0.0682	0.0661
	DCCF	0.1641	0.1819	0.0791	0.0740	0.0626	0.0746	0.0668	0.0627
	GDE	0.1890	0.2055	0.1518	0.1429	0.0653	0.0805	0.0866	0.0839
	JGCF	0.1557	0.1752	0.1438	0.1386	0.0626	0.0789	0.0895	0.0868

EASE

GFCF

BPSM

PGSP

SGFCF

Improv.%

Training-Free

Ours

0.2368

0.2405

0.2333

0.2357

0.2663

+8.69

0.2468

0.2562

0.2466

0.2501

0.2797

+9.17

Table 3: Performance comparison on CiteULike and Yelp. Improv.% denotes the improvement over the best baseline.

Table 4: Performance comparison on Pinterest and Gowalla. Improv.% denotes the improvement over the best baseline.

0.1313

0.0902

0.0886

0.1121

0.1542

+1.58

0.1211

0.0896

0.0872

0.1105

0.1478

+3.43

0.0719

0.0644

0.0622

0.0643

0.0824

+14.29

0.0859

0.0806

0.0748

0.0789

0.0998

+14.45

0.0360

0.0722

0.0113

0.0732

0.0963

+7.60

0.0346

0.0725

0.0110

0.0717

0.0930

+3.56

		Pinterest					Gow	alla	
		x=80%		x=2	20%	x=80%		x=20%	
Metho	d	nDCG	Recall	nDCG	Recall	nDCG	Recall	nDCG	Recall
	BPR	0.0668	0.0780	0.0976	0.0946	0.1164	0.1186	0.1086	0.0917
	DirectAU	0.0840	0.0964	0.1338	0.1289	0.1286	0.1349	0.1864	0.1648
	LightGCN	0.0699	0.0828	0.1297	0.1263	0.0987	0.1074	0.1477	0.1368
	SGL-ED	0.0755	0.0888	0.1355	0.1300	0.1343	0.1417	0.1789	0.1563
Training-Based	XSimGCL	0.0842	0.0973	0.1396	0.1346	0.1288	0.1392	0.1861	0.1643
	LightGCL	0.0717	0.0826	0.1342	0.1287	0.1368	0.1436	0.1524	0.1329
	DCCF	0.0767	0.1002	0.1241	0.1151	0.1179	0.1181	0.1452	0.1244
	GDE	0.0748	0.0862	0.1403	0.1351	0.1261	0.1313	0.1847	0.1645
	JGCF	0.0765	0.0885	0.1363	0.1305	0.1173	0.1260	0.1845	0.1631
	EASE	0.0853	0.0964	0.1064	0.1020	0.1350	0.1440	0.1405	0.1247
Tuoining Enos	GFCF	0.0859	0.0981	0.0851	0.0857	0.1387	0.1483	0.1662	0.1474
Training-Free	BPSM	0.0858	0.0950	0.0567	0.0558	0.1432	0.1482	0.0737	0.0694
	PGSP	0.0876	0.1000	0.1264	0.1219	0.1395	0.1462	0.1640	0.1450
02280	SGFCF	0.0923	0.1052	0.1437	0.1384	0.1432	0.1527	0.1973	0.1762
Ours	Improv.%	+5.36	+4.80	+2.42	+2.44	+0.00	+2.97	+5.85	+6.92

5.2 Comparison

5.2.1 Accuracy. We report the accuracy of baselines and our method in Table 3 and 4. We have the following observations:

- Overall, GCN-based methods show better performance especially on sparse data, indicating the superior ability to tackle data sparsity by incorporating higher-order neighborhood. For instance, LightGCN underperforms BPR on two datasets with x = 80% while significantly outperforms BPR on four datasets with sparse setting x = 20%.
- Comparing non-parametric methods (*i.e.*, Ease, GFCF, PGSP and BPSM) with training-based methods, we can see that non-parametric methods tend to be effective on dense data (*e.g.*, x = 80%) and show relatively poor performance on sparse data (*e.g.*, x = 20%). For instance, GFCF achieves the best baseline on x = 80% while shows the second worst performance on x = 20% on CiteULike. The relatively poor performance of GFCF
- on sparse data also verifies our previous analysis in Section 3.1 that a simple low pass filter cannot work well on datasets with different densities due to the varied noise distribution.
- Contrary to non-parametric methods, GCN-based methods requiring training such as LightGCN, GDE, and JGCF show superior performance on sparse data while are less effective on dense data, which further verifies the analysis in Section 3.1 showing the performance inconsistency of graph filters and supervised training on data with different densities.
- Our proposed SGFCF, significantly outperforms competitive baselines almost across all datasets, demonstrating the effectiveness of our proposed method. Particularly, SGFCF outperforms GFCF which shares similarities with our designs by 14.0% on x=80% and by 41.0% on x=20% on average, in terms of nDCG@10. The larger improvement on sparser data further proves the superiority of our proposed designs over GFCF.

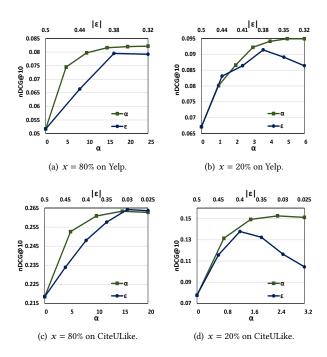


Figure 4: How performance changes with varying α and ϵ .

Table 5: Comparison on training time.

Dataset	Ye	elp	Gowalla			
Method	x=80%	x=80% x=20%		x=20%		
DirectAU	3.33×10^{3} s	7.92×10^{2} s	4.53×10^{3} s	3.30×10^{4} s		
LightGCN	$1.13 \times 10^4 s$	2.73×10^{3} s	5.09×10^{3} s	1.22×10^{3} s		
GFCF	71.0s	1.47×10^{2} s	48.1s	2.40×10^{2} s		
SGFCF	6.3s	2.4s	12.3s	5.5s		

5.2.2 Efficiency. We report the training time of SGFCF and several baselines that have been shown efficient in Table 5, where the results are obtained on a server equipped with AMD Ryzen 9 5950X and GeForce RTX 3090. Despite the light architecture compared with other GCN-based methods, LightGCN still requires much more training time than DirectAU whose complexity is comparable to MF. However, it requires hundreds of training epochs before convergence for training-based methods due to the non-convex loss functions, causing them to be inefficient compared with GFCF which does not require model training. Our proposed SGFCF achieves over 10x and 1000x speedup over GFCF and LightGCN, respectively. Particularly, the higher efficiency of SGFCF over GFCF is attributed to G²N. By generating a desirable spectrum via G²N, graph information is concentrated in fewer low frequency components, reducing the number of required spectral features *K*. For instance, K = 100, 50, and 90 on CiteULike, Yelp, and Gowalla with x = 20%when the accuracy is maximized, as opposed to K = 512 on GFCF.

5.3 Study of SGFCF

5.3.1 Effect of α and ϵ . We study how accuracy changes with α and ϵ and report the results in Figure 4. Since we showed that either α or

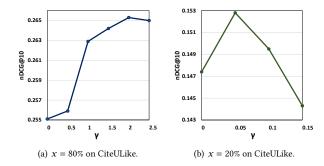


Figure 5: How performance changes with varying γ .

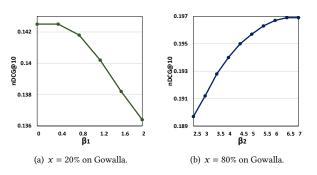


Figure 6: How performance changes with β_1 and β_2 .

 ϵ can adjust the sharpness of spectrum in Section 4.2, we separately tune them and choose the better one, where we fix $\alpha=0$ and $\epsilon=-0.5$ when studying the other one. We can observe that the accuracy is more sensitive to ϵ than α . Note that $\alpha \to +\infty$ and $\epsilon=0$ when nodes are equally weighted, thus $\frac{1}{\sqrt{d_u+\alpha}}$ is a smoother function than d^ϵ_u since $\alpha \in [0,+\infty]$ while $\epsilon \in [-0.5,0]$, explaining why α overall performs better than ϵ . Moreover, the hyperparameter value is larger on the dense setting x=80% than the sparse setting x=20%. A reasonable explanation is that nodes have smaller degrees on the sparse setting on average, on which the model performance is more sensitive to changes in hyperparameters. We can also see the difference between the values on x=20% and x=80% when the best performance is achieved is roughly consistent with their node degree difference (*i.e.*, around 4 times).

5.3.2 Effect of γ . We study the effect of γ and report the results in Table 6 and Figure 5. We can see that introducing γ leads to a better performance, demonstrating that middle frequency components still contain some information contributing to the data representations. Particularly, the improvement on the dense setting x=80% tends to be more significant than that on the sparse setting x=20%. Intuitively, the sparser data are composed of fewer spectral features. For instance, only K=20 spectral features are required on Yelp with x=20% when the best performance is achieved, as opposed to K=300 with X=80%, and we observe a similar trend on other datasets. This observation implies that the middle frequencies are more noisy and useless on sparse datasets. The results in Figure 5

Table 6: The improvement% of γ in terms of nDCG@10.

	Yelp	CiteULike	Pinterest	Gowalla
x=80%	2.35	4.00	0.55	3.66
x=20%	0.21	3.66	0.42	0.42

further verifies our observation, that the accuracy is more sensitive to the change in *y* on sparse setting.

5.3.3 Effect of IGF. We show how accuracy changes with β_1 and β_2 in Figure 6, where we fix $\beta_2 = 2.0$ in (a) and $\beta_1 = 2.5$ in (b). We can observe that an individualized filter adapting to different users/items shows better performance than a shared graph filter.

5.3.4 Impact of δ . Intuitively, the homophilic ratio of distinctive users/items tends to shift towards 1 as increasing δ , making their difference more insignificant. Meanwhile, a larger δ also results in higher computational complexity. As shown in Table 7, the best performance is achieved at $\delta=2$ on most settings except the slight improvement on CiteULike with x=20%. Thus, we set $\delta=2$ considering the trade-off between effectiveness and efficiency.

6 RELATED WORK

6.1 Collaborative Filtering

Collaborative filtering is a fundamental task for recommender systems as it provides recommendations by learning from the useritem historical interactions without rely on specific knowledge or user and item profiles. The underling assumption of CF is that similar users tend to have similar preference [2, 35]. Matrix factorization [22], one of the simplest yet effective methods for CF, characterizes users and items as learnable low-dimensional vectors, where the rating between a user and an item is estimated as the inner product between user and item vectors. Most CF methods can be considered as enhanced MF variants addressing drawbacks of MF which can be mainly classified into three categories: (1) Due to the limited available data, some works incorporate side information to help infer user preference. Rendle et al. [34] introduces temporal information and combine MF with Markov chain (MC) to predict users' next behaviour. Ma et al. [26] integrates social relations and user-item interactions with MF. [24] is an enhanced MF to incorporate geological information. (2) To address the drawback that MF uses a simple linear function to model complex user-item relations, much effort has been devoted to exploit advanced algorithms to learn form user-item interactions, such as multilayer perceptron [15, 48], autoencoder [36], attention mechanism [5], transformer [40], etc. (3) Due to the data sparsity, negative sampling is critical to generate desirable data representations. Therefore, a lot of effective sampling strategies have been proposed [27, 28, 42].

6.2 GCN-based CF methods

Graph convolution networks (GCNs) have shown great potential in recommender systems and collaborative filtering (CF). Early attempts simply apply classic GCN architectures for CF which do not necessarily show desirable performance in recommendation. For instance, SpectralCF [51] shares similarities with Cheb-Net [10], the designs of NGCF [44] are based on GCN [21], and

Table 7: The effect of δ evaluated by nDCG@10.

Settings	Datasets	$\delta = 2$	$\delta = 4$	$\delta = 6$
x=80%	CiteULike	0.2658	0.2631	0.2629
	Yelp	0.0825	0.0823	0.0824
	Pinterest	0.0919	0.0919	0.0919
	Gowalla	0.1432	0.1386	0.1386
	CiteULike	0.1542	0.1547	0.1548
x=20%	Yelp	0.0963	0.0960	0.0953
X=20%	Pinterest	0.1436	0.1438	0.1439
	Gowalla	0.1971	0.1951	0.1932

PinSage [49] is closely related to GraphSAGE [13]. Subsequent works show the redundancy of GCN-based methods such as non-linearity and linear transformation [6, 14], demystify how GCNs contribute to recommendation [30, 31] and analyze the expressive power of GCN for recommendation [4, 37]. Furthermore, research effort has also be devoted to empower GCN with other advanced algorithms, such as transformer [23], sampling strategy [17], contrastive learning [19, 47], etc. and achieve further improvement.

Spectral-bsed GCNs, focusing the spectral domain of graphs, have also received much attention [1, 46]. By analyzing GCN from a perspective of graph signal processing, recent works show that GCN is essentially a low pass filter, and low/high frequencies are significantly contributive to recommendation accuracy [29, 37]. Based on this finding, several spectral GCN-based methods have been proposed and show superiority, that can be classified into two categories: (1) non-parametric graph filters [25, 37] and (2) graph filters combined with supervised training [12, 29]. However, we empirically demonstrated that they fail to perform well on datasets with different densities due to the varied noise distribution.

7 CONCLUSION

In this work, we addressed two limitations of existing GCN-based methods: the lack of generality and expressive power. We proposed a generalized graph normalization (G^2N) to adjust the sharpness of spectrum, making graph filtering generalizable on datasets with different densities, and an individualized graph filtering (IGF), where we emphasize different frequencies based on the homophilic ratio measuring the distinct confidence levels of user preference that interactions can reflect, which is proved to generate arbitrary embeddings. Finally, we proposed a simplified graph filtering for CF (SGFCF) requiring only the top-K singular values. Extensive experimental results on four datasets demonstrated the effectiveness and efficiency of our proposed designs. In future work, we plan to analyze the potential of GCNs from other perspectives and apply our proposed method to other recommendation tasks.

Acknowledgement

This paper is based on results obtained from the project, "Research and Development Project of the Enhanced infrastructures for Post-5G Information and Communication Systems" (JPNP20017), commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

REFERENCES

- [1] Muhammet Balcilar, Renton Guillaume, et al. 2021. Analyzing the Expressive Power of Graph Neural Networks in a Spectral Perspective. In *ICLR*'21.
- [2] John S Breese, David Heckerman, and Carl Kadie. 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In UAI'98. 43–52.
- [3] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2023. LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation. In ICLR'23.
- [4] Xuheng Cai, Lianghao Xia, Xubin Ren, and Chao Huang. 2023. How Expressive are Graph Neural Networks in Recommendation?. In CIKM'23. 173–182.
- [5] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. In SIGIR'17. 335–344.
- [6] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In AAAI-20. 27–34.
- [7] Mengru Chen, Chao Huang, Lianghao Xia, Wei Wei, Yong Xu, and Ronghua Luo. 2023. Heterogeneous Graph Contrastive Learning for Recommendation. In WSDM'23. 544–552.
- [8] Jeongwhan Choi, Seoyoung Hong, Noseong Park, and Sung-Bae Cho. 2023.Blurring-Sharpening Process Models for Collaborative Filtering. In SIGIR'23. 1096–1106.
- [9] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural Networks for Youtube Recommendations. In RecSys'16. 191–198.
- [10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In Neurips'16. 3844–3852.
- [11] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In AISTATS'10. 249–256.
- [12] Jiayan Guo, Lun Du, Xu Chen, Xiaojun Ma, Qiang Fu, Shi Han, Dongmei Zhang, and Yan Zhang. 2023. On Manipulating Signals of User-Item Graph: A Jacobi Polynomial-Based Graph Collaborative Filtering. In KDD'23. 602–613.
- [13] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In Neurips'17. 1024–1034.
- [14] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In SIGIR'20. 639–648.
- [15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In WWW'17. 173–182.
- [16] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In SIGIR'16. 549–558.
- [17] Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. 2021. MixGCF: An Improved Training Method for Graph Neural Network-based Recommender Systems. In KDD'21. 665–674.
- [18] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. TOIS 20, 4 (2002), 422–446.
- [19] Yangqin Jiang, Chao Huang, and Lianghao Huang. 2023. Adaptive Graph Contrastive Learning for Recommendation. In KDD'23. 4252–4261.
- [20] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive Sequential Recommendation. In ICDM'18. 197–206.
- [21] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In ICLR'17.
- [22] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. Computer 8 (2009), 30–37.
- [23] Chaoliu Li, Lianghao Xia, Xubin Ren, Yaowen Ye, Yong Xu, and Chao Huang. 2023. Graph Transformer for Recommendation. In SIGIR'23. 1680–1689.
- [24] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. 2014. GeoMF: Joint Geographical Modeling and Matrix Factorization for Point-of-Interest Recommendation. In KDD'14. 831–840.
- [25] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, Li Shang, and Ning Gu. 2023. Personalized Graph Signal Processing for Collaborative Filtering. In WWW'23. 1264–1272.
- [26] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. 2008. Sorec: Social Recommendation Using Probabilistic Matrix Factorization. In ICDM'08. 931–940.

- [27] Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. 2021. SimpleX: A Simple and Strong Baseline for Collaborative Filtering. In CIKM'21. 1243–1252.
- [28] Seongmin Park, Mincheol Yoon, Jae-woong Lee, Hogun Park, and Jongwuk Lee. 2023. Toward a Better Understanding of Loss Functions for Collaborative Filtering. In CIKM'23, 2034–2043.
- [29] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. 2022. Less is More: Reweighting Important Spectral Graph Features for Recommendation. In SIGIR'22. 1273–1282.
- [30] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. 2022. SVD-GCN: A Simplified Graph Convolution Paradigm for Recommendation. In CIKM'22. 1625– 1634
- [31] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. 2024. Less is More: Removing Redundancy of Graph Convolutional Networks for Recommendation. TOIS 42, 3 (2024), 1–26.
- [32] Xubin Ren, Lianghao Xia, Jiashu Zhao, Dawei Yin, and Chao Huang. 2023. Disentangled Contrastive Collaborative Filtering. In SIGIR'23. 1137–1146.
- [33] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In UAI'09. 452– 461
- [34] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-Basket Recommendation. In WWW'10. 811–820.
- [35] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In WWW'01. 285–295.
- [36] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders Meet Collaborative Filtering. In WWW'15. 111–112.
- [37] Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, B Khaled Letaief, and Dongsheng Li. 2021. How Powerful is Graph Convolution for Recommendation?. In CIKM'21. 1619–1629.
- [38] Daniel Spielman. 2012. Spectral graph theory. Combinatorial scientific computing 18 (2012).
- [39] Harald Steck. 2019. Embarrassingly Shallow Autoencoders for Sparse Data. In WWW'19. 3251–3257.
- [40] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In CIKM'19. 1441–1450.
- [41] Jianing Sun, Zhaoyue Cheng, Saba Zuberi, Felipe Pérez, and Maksims Volkovs. 2021. HGCF: Hyperbolic Graph Convolution Networks for Collaborative Filtering. In WWW'21, 593–601.
- [42] Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. 2022. Towards Representation Alignment and Uniformity in Collaborative Filtering. In KDD'22. 1816–1825.
- [43] Wenjie Wang, Yiyan Xu, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. 2023. Diffusion Recommender Model. In SIGIR'23. 832–841.
- [44] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In SIGIR'19. 165–174.
- [45] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled Graph Collaborative Filtering. In SIGIR'20. 1001–1010.
- [46] Xiyuan Wang and Muhan Zhang. 2022. How Powerful are Spectral Graph Neural Networks. In ICML'22. 23341–23362.
- [47] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-Supervised Graph Learning for Recommendation. In SIGIR'21. 726–735.
- [48] Hong-Jian Xue, Xinyu Dai, et al. 2017. Deep Matrix Factorization Models for Recommender Systems. In IJCAI-17. 3203–3209.
- [49] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In KDD'18. 974–983.
- [50] Junliang Yu, Xin Xia, Tong Chen, Lizhen Cui, Nguyen Quoc Viet Hung, and Hongzhi Yin. 2024. XSimGCL: Towards Extremely Simple Graph Contrastive Learning for Recommendation. TKDE 36, 2 (2024), 913–926.
- [51] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S Yu. 2018. Spectral Collaborative Filtering. In RecSys'18. 311–319.

Table 8: Ablation study on SGFCF evaluated by nDCG@10.

Dataset	Yelp		Citel	JLike	Gowalla	
Setting	x=80%	x=20%	x=80%	x=20%	x=80%	x=20%
SGFCF	0.0824	0.0963	0.2667	0.1542	0.1432	0.1973
w/o IGF	0.0820	0.0949	0.2651	0.1532	0.1388	0.1922
w/o IGF & G ² N	0.0537	0.0729	0.2287	0.0924	0.1255	0.1313

Table 9: Performance comparison on x=40% and 60%.

Setting			Methods				
Dataset	Metric		LightGCN	GFCF	JGCF	SGFCF	Improv.%
x=40%	nDCG	0.2120	0.2021	0.2095	0.2440	+15.09	
CiteULike	X=40%	Recall	0.1989	0.1951	0.1994	0.2298	+15.54
	x=60%	nDCG	0.2023	0.2502	0.2039	0.2739	+9.47
		Recall	0.1969	0.2425	0.1986	0.2622	+8.12
	x=40%	nDCG	0.1204	0.1275	0.1294	0.1371	+5.95
Pinterest	X=40%	Recall	0.1170	0.1221	0.1241	0.1313	+5.80
	x=60%	nDCG	0.0912	0.1064	0.1025	0.1117	+4.98
	X=60%	Recall	0.0891	0.1022	0.0990	0.1067	+4.40

Table 10: Performance (nDCG@10) on different graph filters.

Dataset	Yelp		Pint	erest	Gowalla	
Setting	tting $ x=80\% $		x=80%	x=20%		
Monomial	0.0820	0.0949	0.0922	0.1435	0.1388	0.1922
Exp	0.0816	0.0940	0.0919	0.1432	0.1385	0.1922
Markov	0.0802	0.0945	0.0904	0.1406	0.1335	0.1714
Jacobi	0.0770	0.0784	0.0856	0.1425	0.1381	0.1909

A SUPPLEMENTARY EXPERIMENTS

A.1 Ablation Study

To demonstrate the effectiveness of our proposed designs, we compare three variants: (1) SGFCF, (2) SGFCF without IGF, and (3) SGFCF without both G²N and IGF, and report the results in Table 8. We observe that removing either of G²N and IGF results in performance degradation, showing that both G²N and IGF are contributive to model performance. Particularly, IGF tends to be more effective on the sparse setting, indicating that it might require stronger expressive power to generate the optimal representations on the sparse data. Moreover, G²N contributes more to the accuracy than IGF, as the poor performance of graph filters is mainly due to the varied noise distribution polluting the low frequencies being important to the data representations.

A.2 Experiments on Other Density Settings

To further verify the effectiveness of our methods, we compare our SGFCF with several competitive baselines on x=40% and 60%, and report the results in Table 9. We can observe that GFCF without training shows better performance on the denser setting x=60%, while training-based methods JGCF and LightGCN achieve better results on the sparser setting x=40% as supervised training shows superior ability learning from noisy data. Our proposed SGFCF outperforms all baselines across the board, demonstrating the generality of our proposed designs.

A.3 Graph Filter Designs

We compare four commonly used graph filters for recommendation: monomial filter, exponential diffusion kernel, Markov diffusion kernel, and Jacobi polynomials (with detailed introduction as follows) and report the results in Table 10. Overall, the monomial filter: λ_k^β with the simplest design shows better performance than other filters. Due to the important features that are concentrated in only a few low frequencies via G^2N , a simple increasing function is already able to appropriately emphasize different frequencies according to their importance instead of complicated band-pass filters such as the Markov diffusion kernel and Jacobi polynomials.

A.3.1 Monomial Filter. The eigenvalue of $\hat{\mathbf{A}}^l$ is λ_k^l , here we extend it to λ_k^β where $\beta \geq 0$. It is a simple increasing function.

A.3.2 Exponential Diffusion Kernel. The exponential diffusion kernel is defined as:

$$\exp(\beta \hat{\mathbf{A}}) = \sum_{l=0}^{\infty} \frac{\beta^l \hat{\mathbf{A}}^l}{l!},$$
(22)

where the corresponding eigenvalue is $e^{\beta \lambda_k}$.

A.3.3 Markov Diffusion Kernel. The Markov diffusion kernel is defined as: $\frac{1}{L}\sum_{l=0}^L \hat{\mathbf{A}}^l$ where the corresponding eigenvalue is $\frac{\sum_{l=0}^L \lambda_k^l}{L}$.

A.3.4 Jacobi Polynomials. The Jacobi basis for $k \ge 2$ is defined as:

$$P_k^{a,b}(\hat{\mathbf{A}})\mathbf{E} = \theta_k \hat{\mathbf{A}} P_{k-1}^{a,b}(\hat{\mathbf{A}})\mathbf{E} + \theta_k' P_{k-1}^{a,b}(\hat{\mathbf{A}})\mathbf{E} - \theta_k'' P_{k-2}^{a,b}(\hat{\mathbf{A}})\mathbf{E}, \quad (23)$$
here
$$\theta_k = \frac{(2k+a+b)(2k+a+b-1)}{2k(k+a+b)},$$

$$\theta_k' = \frac{(2k+a+b-1)(a^2-b^2)}{2k(k+a+b)(2k+a+b-2)},$$

$$\theta_k'' = \frac{(k+a-1)(k+b-1)(2k+a+b)}{k(k+a+b)(2k+a+b-2)}.$$

For k < 2,

$$P_0^{a,b}(\hat{\mathbf{A}})\mathbf{E} = \mathbf{E},$$

 $P_0^{a,b}(\hat{\mathbf{A}})\mathbf{E} = \mathbf{E} = \frac{a-b}{2}\mathbf{E} + \frac{a+b+2}{2}\hat{\mathbf{A}}\mathbf{E}.$ (25)

The final representations are generated by summing up the embeddings from each iteration: $\sum_{k=0}^L P_k^{a,b}(\hat{\mathbf{A}})$.

B PROOFS

B.1 Proof of Theorem 1 and Corollary 1

PROOF. Let \mathbf{p}_k , \mathbf{q}_k , and σ_k be the left, right singular vector, and the singular value of $\hat{\mathbf{R}}$, we can show the following relations:

$$\begin{bmatrix} \mathbf{0} & \hat{\mathbf{R}} \\ \hat{\mathbf{R}}^{T} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{k} \\ \mathbf{q}_{k} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{R}} \mathbf{q}_{k} \\ \hat{\mathbf{R}}^{T} \mathbf{p}_{k} \end{bmatrix} = \sigma_{k} \begin{bmatrix} \mathbf{p}_{k} \\ \mathbf{q}_{k} \end{bmatrix},$$

$$\begin{bmatrix} \mathbf{0} & \hat{\mathbf{R}} \\ \hat{\mathbf{R}}^{T} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{k} \\ -\mathbf{q}_{k} \end{bmatrix} = \begin{bmatrix} -\hat{\mathbf{R}} \mathbf{q}_{k} \\ \hat{\mathbf{R}}^{T} \mathbf{p}_{k} \end{bmatrix} = -\sigma_{k} \begin{bmatrix} \mathbf{p}_{k} \\ \mathbf{q}_{k} \end{bmatrix},$$
(26)

where σ_k and $-\sigma_k$ are two eigenvalues of $\hat{\mathbf{A}}$, with $[\mathbf{p}_k,\mathbf{q}_k]$ and $[\mathbf{p}_k,-\mathbf{q}_k]$ as the corresponding eigenvectors, respectively. According to Theorem 1, it is easy to show $r_{ui}^{(K)} = r_{ui}^{(n-K)} = \mathbf{P}^{(K)}\mathbf{Q}^{(K)}$. Corollary 1 implies that the rating estimated by the high frequencies

are exactly opposite to the symmetric (to $\lambda_{n/2}=0$) low frequencies based on Equation (7).

B.2 Proof of Theorem 2

PROOF. We first consider d = 1. If LGCN can generate arbitrary embeddings, then the following Equation holds:

$$\mathbf{V}^T \mathbf{O} = diag\left(q(\lambda_k)\right) \mathbf{V}^T \mathbf{E},\tag{27}$$

which is equivalent to the following Equation:

$$\begin{bmatrix} \frac{(\mathbf{V}^T \mathbf{O})_1}{(\mathbf{V}^T \mathbf{E})_1} \\ \vdots \\ \frac{(\mathbf{V}^T \mathbf{O})_n}{(\mathbf{V}^T \mathbf{E})_n} \end{bmatrix} = \begin{bmatrix} \sum_l \theta_l \lambda_1^l \\ \vdots \\ \sum_l \theta_l \lambda_n^l \end{bmatrix} = \mathbf{B} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}, \tag{28}$$

where $\mathbf{B} \in \mathbb{R}^{n \times L}$, $\mathbf{B}_{kl} = \lambda_k^l$. By extending the order of $g(\lambda_k)$ to n (*i.e.*, L = n), \mathbf{B} becomes a full rank matrix as long as $\hat{\mathbf{A}}$ has no repeated eigenvalue, thus we can always find a solution to satisfy Equation (28). To generalize the above result to the situation of d > 1, multiple filters are required:

$$\begin{bmatrix} \frac{(\mathbf{V}^{T}\mathbf{O})_{11}}{(\mathbf{V}^{T}\mathbf{E})_{11}}, \cdots, \frac{(\mathbf{V}^{T}\mathbf{O})_{1d}}{(\mathbf{V}^{T}\mathbf{E})_{1d}} \\ \vdots & \vdots \\ \frac{(\mathbf{V}^{T}\mathbf{O})_{n1}}{(\mathbf{V}^{T}\mathbf{E})_{n1}}, \cdots, \frac{(\mathbf{V}^{T}\mathbf{O})_{nd}}{(\mathbf{V}^{T}\mathbf{E})_{nd}} \end{bmatrix} = \mathbf{B} \begin{bmatrix} \theta_{11}, \cdots, \theta_{1d} \\ \vdots \\ \theta_{n1}, \cdots, \theta_{nd} \end{bmatrix}, \tag{29}$$

By applying Equation (29) to LGCN, we get Equation (10). \Box

B.3 Proof of Theorem 3

PROOF. We first show that the Frobenius norm can be defined as follows:

$$|\hat{\mathbf{A}}|_F^2 = \operatorname{trace}\left(\hat{\mathbf{A}}^T \hat{\mathbf{A}}\right) = \sum_{k=1}^n \lambda_k^2$$
 (30)

According to Theorem 1, $\sum_{k=1}^{n/2} \lambda_k^2 = \sum_{k=n/2}^n \lambda_k^2$ where $\lambda_{n/2} = 0$. Let $\frac{\tilde{\lambda}_k^2}{\lambda_k^2} = \alpha_k$, then $\alpha_1 \geq \cdots \geq \alpha_K > \alpha_{K+1} \geq \cdots \geq \alpha_{n/2}$, and the following relation holds:

$$2\frac{\left|\tilde{\mathbf{A}}_{K}\right|_{F}^{2}}{\left|\tilde{\mathbf{A}}\right|_{F}^{2}} = \frac{\sum_{k=1}^{K} \tilde{\lambda}_{k}^{2}}{\sum_{k=1}^{n/2} \tilde{\lambda}_{k}^{2}} = \frac{\sum_{k=1}^{K} \alpha_{k} \lambda_{k}^{2}}{\sum_{k=1}^{K} \alpha_{k} \lambda_{k}^{2} + \sum_{j=K+1}^{n/2} \alpha_{j} \lambda_{j}^{2}}$$

$$\geq \frac{\alpha_{K} \sum_{k=1}^{K} \lambda_{k}^{2}}{\alpha_{K} \sum_{k=1}^{K} \lambda_{k} + \sum_{k=1}^{K} \alpha_{k} \lambda_{k}^{2} + \sum_{j=K+1}^{n/2} \alpha_{j} \lambda_{j}^{2}}$$

$$\geq \frac{\alpha_{K} \sum_{k=1}^{K} \lambda_{k}^{2}}{\alpha_{K} \sum_{k=1}^{K} \lambda_{k} + \alpha_{K+1} \sum_{j=K+1}^{n/2} \lambda_{j}^{2}}$$

$$\geq \frac{\sum_{k=1}^{K} \lambda_{k}^{2}}{\sum_{k=1}^{N/2} \lambda_{k}^{2}} = 2\frac{\left|\hat{\mathbf{A}}_{K}\right|_{F}^{2}}{\left|\hat{\mathbf{A}}\right|_{F}^{2}}$$
(31)

The inequalities above are based on the observation that $\frac{x}{x+y} = 1 - \frac{y}{x+y}$ decreases as x decreases and y increases for x > 0 and y > 0.

B.4 Proof of Theorem 4

PROOF. According to Courant-Fischer Theorem [38], we have:

$$\lambda_k = \max_{\dim(S)=k} \min_{\mathbf{x} \in S} \mathbf{x}^T \hat{\mathbf{A}} \mathbf{x} \qquad \text{s.t.} |\mathbf{x}| = 1$$
 (32)

where the eigenvalue is arranged in descending order. Then,

$$\begin{split} \tilde{\lambda}_k &= \max_{\dim(S)=k} \min_{\mathbf{x} \in S} \mathbf{x}^T \tilde{\mathbf{A}} \mathbf{x} = \max_{\dim(S)=k} \min_{\mathbf{x} \in S} \sum_{(u,i) \in \mathcal{E}} \frac{2\mathbf{x}_u \mathbf{x}_i}{(d_u + \alpha)^{-\epsilon} (d_i + \alpha)^{-\epsilon}} \\ &= \max \min_{(u,i) \in \mathcal{E}} \frac{2\mathbf{x}_u \mathbf{x}_i}{(d_u)^{0.5} (d_i)^{0.5}} \sqrt{\frac{d_u d_i}{(d_u + \alpha) (d_i + \alpha)}} \left((d_u + \alpha) (d_i + \alpha) \right)^{0.5 + \epsilon} \end{split}$$

$$\leq d_{max} (d_{max} + \alpha)^{2\epsilon} \left(\max \min \mathbf{x}^T \hat{\mathbf{A}} \mathbf{x} \right) = d_{max} (d_{max} + \alpha)^{2\epsilon} \lambda_k.$$
(33)

In the second last step, $\frac{d_u}{d_u + \alpha}$ increases over d_u , thus $\sqrt{\frac{d_u d_i}{(d_u + \alpha)(d_i + \alpha)}} \le \frac{d_{max}}{d_{max} + \alpha}$. It is evident that $((d_u + \alpha)(d_i + \alpha))^{0.5 + \epsilon} \le (d_{max} + \alpha)^{1 + 2\epsilon}$. Similarly, we can prove $\tilde{\lambda}_k \ge d_{min}(d_{min} + \alpha)^{2\epsilon} \lambda_k$.

B.5 Proof of Theorem 5

Proof. The power of the adjacency matrix can be rewritten as follows:

$$\hat{\mathbf{A}}^{l} = \left\{ \begin{array}{ccc} \left[\left(\hat{\mathbf{R}} \hat{\mathbf{R}}^{T} \right)^{\frac{l}{2}} & \mathbf{0} \\ \mathbf{0} & \left(\hat{\mathbf{R}}^{T} \hat{\mathbf{R}} \right)^{\frac{l}{2}} \right] & l = \{0, 2, 4, \cdots\} \\ \\ \left[\begin{array}{ccc} \mathbf{0} & \hat{\mathbf{R}} \left(\hat{\mathbf{R}}^{T} \hat{\mathbf{R}} \right)^{\frac{l-1}{2}} \\ \hat{\mathbf{R}}^{T} \left(\hat{\mathbf{R}} \hat{\mathbf{R}}^{T} \right)^{\frac{l-1}{2}} & \mathbf{0} \end{array} \right] & l = \{1, 3, 5, \cdots\}. \end{array} \right.$$

$$(34)$$

According to singular value decomposition (SVD): $\hat{\mathbf{R}} = \mathbf{P} diag(\lambda_k) \mathbf{Q}^T$, it is easy to show the following relations:

$$\left(\hat{\mathbf{R}}\hat{\mathbf{R}}^{T}\right)^{\frac{l}{2}} = \mathbf{P}diag\left(\sigma_{k}^{l}\right)\mathbf{P}^{T}, \quad \left(\hat{\mathbf{R}}^{T}\hat{\mathbf{R}}\right)^{\frac{l}{2}} = \mathbf{Q}diag\left(\sigma_{k}^{l}\right)\mathbf{Q}^{T}, \\
\hat{\mathbf{R}}\left(\hat{\mathbf{R}}^{T}\hat{\mathbf{R}}\right)^{\frac{l-1}{2}} = \mathbf{P}diag\left(\sigma_{k}^{l}\right)\mathbf{Q}^{T}, \quad \hat{\mathbf{R}}^{T}\left(\hat{\mathbf{R}}\hat{\mathbf{R}}^{T}\right)^{\frac{l-1}{2}} = \mathbf{Q}diag\left(\sigma_{k}^{l}\right)\mathbf{P}^{T}.$$
(35)

Then, the final embeddings of SGF can be formulated as follows:

$$\mathbf{O} = \sum_{l=0}^{L} \hat{\mathbf{A}}^{l} = \begin{bmatrix} \mathbf{P}diag(\psi(\sigma_{k}))\mathbf{P}^{T} & \mathbf{P}diag(\omega(\sigma_{k}))\mathbf{Q}^{T} \\ \mathbf{Q}diag(\omega(\sigma_{k}))\mathbf{P}^{T} & \mathbf{Q}diag(\psi(\sigma_{k}))\mathbf{Q}^{T} \end{bmatrix}, \quad (36)$$

where $\psi(\sigma_k)=\sum_{l=\{0,2,\cdots\}}\sigma_k^l$, $\omega(\sigma_k)=\sum_{l=\{1,3,\cdots\}}\sigma_k^l$. Then, the ratings are estimated as:

$$\mathbf{O}_{U}\mathbf{O}_{I}^{T} = \mathbf{P}diag(\phi(\lambda_{k})\omega(\lambda_{k}))\mathbf{Q}^{T}$$
(37)