SVD-GCN: A Simplified Graph Convolution Paradigm for Recommendation

Shaowen Peng
swpeng95@gmail.com
Kyoto University
Kyoto, Japan

Kazunari Sugiyama
kaz.sugiyama@i.kyoto-u.ac.jp
Kyoto University
Kyoto, Japan

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

ABSTRACT

With the tremendous success of Graph Convolutional Networks (GCNs), they have been widely applied to recommender systems and have shown promising performance. However, most GCN-based methods rigorously stick to a common GCN learning paradigm and suffer from two limitations: (1) the limited scalability due to the high computational cost and slow training convergence; (2) the notorious over-smoothing issue which reduces performance as stacking graph convolution layers. We argue that the above limitations are due to the lack of a deep understanding of GCN-based methods. To this end, we first investigate what design makes GCN effective for recommendation. By simplifying LightGCN, we show the close connection between GCN-based and low-rank methods such as Singular Value Decomposition (SVD) and Matrix Factorization (MF), where stacking graph convolution layers is to learn a low-rank representation by emphasizing (suppressing) components with larger (smaller) singular values. Based on this observation, we replace the core design of GCN-based methods with a flexible truncated SVD and propose a simplified GCN learning paradigm dubbed SVD-GCN, which only exploits K-largest singular vectors for recommendation. To alleviate the over-smoothing issue, we propose a renormalization trick to adjust the singular value gap, resulting in significant improvement. Extensive experiments on three real-world datasets show that our proposed SVD-GCN not only significantly outperforms state-of-the-arts but also achieves over 100x and 10x speedups over LightGCN and MF, respectively.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Collaborative Filtering, Graph Convolutional Network

ACM Reference Format:
Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. 2022. SVD-GCN: A Simplified Graph Convolution Paradigm for Recommendation. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557462

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557462

1 INTRODUCTION

With rapid development of the Internet and web services, recommender systems have been playing an important role in people's daily life. As a fundamental task for recommendation, Collaborative Filtering (CF) focuses on digging out the user preference from past user-item interactions, and has received much attention for decades. One of the most widely used CF methods, low-rank matrix factorization (MF) [21], characterizes user/item as latent vectors in an embedding space and estimates ratings as the cosine similarity between user and item latent vectors. To overcome the drawback of MF that a linear function is inefficient to capture complex user behaviour, subsequent works incorporate side information (e.g., user reviews, image data, temporal information, etc.) [3, 6, 16] and exploit advanced algorithms [11, 30, 33] to infer user preference.

However, traditional CF methods heavily rely upon the quality of interactions as they can only learn the direct user-item relations. Therefore, they always show poor performance due to the common data sparsity issue in practice. Recently, Graph Convolutional Networks (GCNs) [19] have shown great potential in various fields including social network analysis [5, 36] and recommender systems [2, 38]. Much research effort has been devoted to adapt GCNs for recommendation, such as augmenting GCNs with other advanced algorithms [15, 31, 35], simplifying GCNs to improve training efficiency and model effectiveness [4, 10, 24], and so on. By representing user-item interactions as a bipartite graph, the core idea of GCNs is to repeatedly propagate user and item embeddings on the graph to aggregate higher-order collaborative signals, thereby learning high quality embeddings even with limited interactions. Despite its effectiveness, most existing GCN-based methods suffer from the following limitations:

- The core step of GCNs is implemented by repeatedly multiplying by an adjacency matrix, resulting in high computational cost and poor scalability.
- As shown in many works [22, 40], stacking graph convolution layers tends to cause the overs-smoothing issue, resulting in similar user/item representations and reducing the recommendation accuracy. As a result, most existing GCN-based CF methods remain shallow (two, three layers at most).
- Unlike traditional CF methods, user/item representations are contributed from tremendous higher-order neighborhood, making the model difficult to train. Some GCN-based CF methods such as LightGCN requires about 800 epochs to reach the best accuracy, which further increases the training cost.

We argue that the above limitations are due to the lack of a deep understanding of GCNs. Thus, in this work, we aim to figure out: what is the core design making GCNs effective for recommendation?

Based on our answer to this question, we propose a scalable and simple GCN learning paradigm without above limitations.

To this end, we first dissect LightGCN, a linear GCN-based CF method which only exploits neighborhood aggregation and removes other designs. By simplifying LightGCN, we show that it is closely related to low-rank CF methods such as Singular Value Decomposition (SVD) and low-rank Matrix Factorization (MF), where stacking graph convolution layers is to learn a low-rank representation by emphasizing (suppressing) the components with larger (smaller) singular values. With empirical analysis, we further show that only a very few components corresponding to K-largest singular values contribute to recommendation performance, whereas most information (over 95% on the tested data) are noisy and can be removed. Based on the above analysis, we replace the core component of GCNs (i.e., neighborhood aggregation) with a flexible truncated SVD and propose a simplified GCN learning paradigm dubbed SVD-GCN. Specifically, SVD-GCN only requires a very few (K-largest) singular values (vectors) and model parameters (less than 1% of MF's on the tested data) for prediction. To alleviate the over-smoothing issue, we propose a renormalization trick to adjust the singular value gap, making important features of interactions well preserved, thereby resulting in significant improvement. Furthermore, to make the best of interactions, we augment SVD-GCN with user-user and item-item relations, leading to further improvement. Since the superiority of GCNs over traditional CF methods lies in the ability to augment interactions with higher-order collaborative signals, we only use 20% of the interactions for training to evaluate the robustness and effectiveness of GCN designs. The main contributions of this work are summarized as follows:

- By showing the connection between GCN-based and low-rank CF methods, we provide deep insight into GCN-based CF methods, that they contribute to recommendation in the same way as low-rank methods.
- Distinct from the GCN learning paradigm that most GCN-based methods rigorously sticking to, we propose a simplified formulation of GCNs dubbed SVD-GCN, which only exploits K-largest singular values and vectors and is equipped with a lighter structure than MF.
- To tackle the over-smoothing issue, we propose a renormalization trick to adjust the singular value gap to assure that important features from interactions are well preserved, leading to significant improvement.
- Extensive experiments on three datasets show that our proposed SVD-GCN outperforms state-of-the-art with higher training efficiency and less running time.

2 PRELIMINARIES

2.1 GCN learning paradigm for CF

We summarize a common GCN learning paradigm for CF. Given the user set $\mathcal{U}$, item set $\mathcal{I}$ and an interaction matrix $\mathrm{R} \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$, we define a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the node set $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ contains all users and items, the edge set $\mathcal{E} = \mathrm{R}^+$ is represented by observed interactions, where $\mathrm{R}^+ = \{r_{ui} = 1 | u \in \mathcal{U}, i \in \mathcal{I}\}$. Each user/item is considered as a node on the graph and parameterized as an embedding vector $\mathrm{e}_u / \mathrm{e}_i \in \mathbb{R}^d$. The core idea of GCNs is to update user and item embeddings by propagating them on the graph. The adjacency relations are represented as:

$$\mathrm{A} = \begin{bmatrix} 0 & \mathrm{R} \\ \mathrm{R}^T & 0 \end{bmatrix}. \tag{1}$$

The updating rule of GCNs is formulated as follows:

$$\mathrm{H}^{(l+1)} = \sigma\left(\tilde{\mathrm{A}}\mathrm{H}^{(l)}\mathrm{W}^{(l+1)}\right), \tag{2}$$

where $\tilde{\mathrm{A}} = \mathrm{D}^{-\frac{1}{2}}\mathrm{A}\mathrm{D}^{-\frac{1}{2}}$ is a symmetric normalized adjacency matrix, D is a diagonal node degree matrix. The initial state is $\mathrm{H}^{(0)} = \mathrm{E}$, where $\mathrm{E} \in \mathbb{R}^{(|\mathcal{U}|+|\mathcal{I}|)\times d}$ contains users' and items' embedding vectors. Recent works [4, 10] show the non-linear activation function $\sigma(\cdot)$ and feature transformations $\mathrm{W}^{(l+1)}$ are redundant for CF , the above updating rule can be simplified as follows:

$$\mathrm{H}^{(l)} = \tilde{\mathrm{A}}^l \mathrm{E}. \tag{3}$$

The final embeddings are generated by accumulating the embeddings at each layer through a pooling function:

$$\mathrm{O} = \mathrm{pooling}\left(\mathrm{H}^{(l)} | l = \{0, 1, \cdots, L\}\right). \tag{4}$$

Finally, an interaction is estimated as the inner product between a user's and an item's final embedding:

$$\hat{r}_{ui} = \mathrm{o}_u^T \mathrm{o}_i. \tag{5}$$

2.2 Low-Rank Methods

Low rank representation plays a fundamental role in modern recommender systems [17]. The core idea of low-rank methods is inspired by Singular Value Decomposition (SVD):

$$\mathrm{R} = \mathrm{U} diag\left(s_k\right) \mathrm{V}^T \approx \sum_{k=1}^{K} s_k \mathrm{u}_k \mathrm{v}_k^T. \tag{6}$$

The interaction matrix can be decomposed to three matrices, where the column of [U and V (i.e., u_k and v_k)] and s_k are [left and right singular vectors] and singular value, respectively; $s_1 > s_2 > \cdots \geq 0$; $diag(\cdot)$ is the diagonalization operation. Since the components with larger (smaller) singular values contribute more (less) to interactions, we can approximate R with only K-largest singular values. Alternatively, we can learn low-rank representations in a dynamical way through matrix factorization (MF) [21]:

$$\min \sum_{(u,i)\in \mathrm{R}^+} \left\| r_{ui} - \mathrm{e}_u^T \mathrm{e}_i \right\|_2^2 + \lambda \left(\|\mathrm{e}_u\|_2^2 + \|\mathrm{e}_i\|_2^2 \right), \tag{7}$$

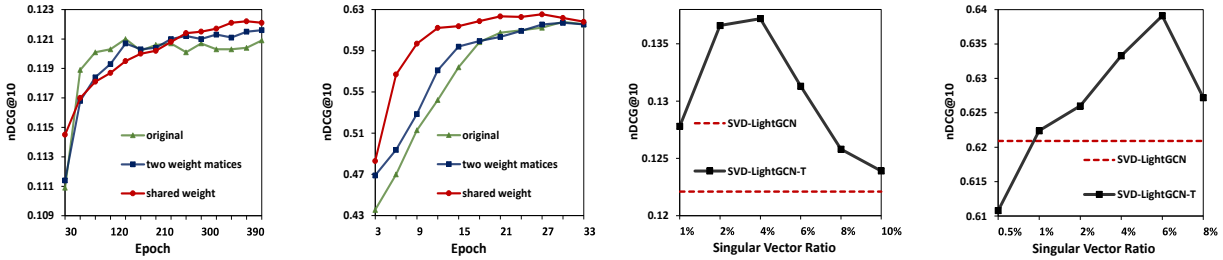
where λ is the strength for regularization. Each user and item is represented as a trainable vector with dimension $d \leq \min(|\mathcal{U}|, |\mathcal{V}|)$. By optimizing the following objective function, the model is expected to learn important features from interactions (e.g., components corresponding to d-largest singular values).

3 METHODOLOGY

3.1 Connections Between GCNs and SVD

As activation functions and feature transformations have been shown ineffective for CF [10], we focus on LightGCN whose final embeddings are generated as follows:

$$\mathrm{O} = \sum_{l=0}^{L} \frac{\mathrm{H}^{(l)}}{L+1} = \left(\sum_{l=0}^{L} \frac{\tilde{\mathrm{A}}^l}{L+1} \right) \mathrm{E}, \tag{8}$$



(a) The accuracy of three variants on CiteULike evaluated by nDCG@10.
(b) The accuracy of three variants on ML-100K.
(c) Comparison between SVD-LightGCN and SVD-LightGCN-T on CiteULike.
(d) Comparison between SVD-LightGCN and SVD-LightGCN-T on ML-100K.

Figure 1: Some empirical results on two datasets (CiteULike and ML-100K).

where the pooling function is $\frac{1}{L+1}$. If we take a closer look at the power of adjacency matrix $\tilde{\mathrm{A}}^l$, we have the following observation:

$$\tilde{\mathrm{A}}^l = \begin{cases} \begin{bmatrix} \left(\tilde{\mathrm{R}}\tilde{\mathrm{R}}^T\right)^{\frac{l}{2}} & 0 \\ 0 & \left(\tilde{\mathrm{R}}^T\tilde{\mathrm{R}}\right)^{\frac{l}{2}} \end{bmatrix} & l = \{0, 2, 4, \cdots\} \\ \begin{bmatrix} 0 & \tilde{\mathrm{R}}\left(\tilde{\mathrm{R}}^T\tilde{\mathrm{R}}\right)^{\frac{l-1}{2}} \\ \mathrm{R}^T\left(\tilde{\mathrm{R}}\tilde{\mathrm{R}}^T\right)^{\frac{l-1}{2}} & 0 \end{bmatrix} & l = \{1, 3, 5, \cdots\}. \end{cases} \tag{9}$$

Following the definition of $\tilde{\mathrm{A}}$, $\tilde{\mathrm{R}} = \mathrm{D}_U^{-\frac{1}{2}}\mathrm{R}\mathrm{D}_I^{-\frac{1}{2}}$, where D_U and D_I are the node degree matrices for users and items, respectively. Then, we can split Equation (8) as follows:

$$\begin{aligned} \mathrm{O}_U &= \frac{\sum_{l=\{0,2,4,\cdots\}}\left(\tilde{\mathrm{R}}\tilde{\mathrm{R}}^T\right)^{\frac{l}{2}}\mathrm{E}_U + \sum_{l=\{1,3,5,\cdots\}}\tilde{\mathrm{R}}\left(\tilde{\mathrm{R}}^T\tilde{\mathrm{R}}\right)^{\frac{l-1}{2}}\mathrm{E}_I}{L+1}, \\ \mathrm{O}_I &= \frac{\sum_{l=\{0,2,4,\cdots\}}\left(\tilde{\mathrm{R}}^T\tilde{\mathrm{R}}\right)^{\frac{l}{2}}\mathrm{E}_I + \sum_{l=\{1,3,5,\cdots\}}\tilde{\mathrm{R}}^T\left(\tilde{\mathrm{R}}\tilde{\mathrm{R}}^T\right)^{\frac{l-1}{2}}\mathrm{E}_U}{L+1}. \end{aligned} \tag{10}$$

The first and second terms represent the messages from homogeneous (even-hops) and heterogeneous (odd-hops) neighborhood, O_U and O_I are final embeddings for user and items, E_U and E_I are embedding matrices for users and items, respectively. Similar to the definition in Section 2.2, let P, Q, and σ_k denote the stacked left, right singular vectors, and singular value for $\tilde{\mathrm{R}}$, respectively, and we formulate the following theorem.

Theorem 1. *The adjacency relations in Equation (10) can be rewritten as the following forms:*

$$\begin{aligned} \left(\tilde{\mathrm{R}}\tilde{\mathrm{R}}^T\right)^l &= \mathrm{P}diag\left(\sigma_k^{2l}\right)\mathrm{P}^T, \\ \left(\tilde{\mathrm{R}}^T\tilde{\mathrm{R}}\right)^l &= \mathrm{Q}diag\left(\sigma_k^{2l}\right)\mathrm{Q}^T, \end{aligned} \tag{11}$$

$$\begin{aligned} \tilde{\mathrm{R}}\left(\tilde{\mathrm{R}}^T\tilde{\mathrm{R}}\right)^{\frac{l-1}{2}} &= \mathrm{P}diag\left(\sigma_k^l\right)\mathrm{Q}^T, \\ \mathrm{R}^T\left(\tilde{\mathrm{R}}\tilde{\mathrm{R}}^T\right)^{\frac{l-1}{2}} &= \mathrm{Q}diag\left(\sigma_k^l\right)\mathrm{P}^T. \end{aligned} \tag{12}$$

Following Theorem 1, we can rewrite Equation (10) as:

$$\begin{aligned} \mathrm{O}_U &= \mathrm{P}diag\left(\frac{\sum_{l=\{0,2,\cdots\}}\sigma_k^l}{L+1}\right)\mathrm{P}^T\mathrm{E}_U + \mathrm{P}diag\left(\frac{\sum_{l=\{1,3,\cdots\}}\sigma_k^l}{L+1}\right)\mathrm{Q}^T\mathrm{E}_I, \\ \mathrm{O}_I &= \mathrm{Q}diag\left(\frac{\sum_{l=\{0,2,\cdots\}}\sigma_k^l}{L+1}\right)\mathrm{Q}^T\mathrm{E}_I + \mathrm{Q}diag\left(\frac{\sum_{l=\{1,3,\cdots\}}\sigma_k^l}{L+1}\right)\mathrm{P}^T\mathrm{E}_U. \end{aligned} \tag{13}$$

Now the final embeddings are contributed from $\tilde{\mathrm{R}}$'s singular vectors and values instead of neighborhood. Note that:

$$\mathrm{P}diag\left(\frac{\sum_{l=\{0,2,\cdots\}}\sigma_k^l}{L+1}\right)\mathrm{P}^T = \sum_k \frac{\sum_{l=\{0,2,\cdots\}}\sigma_k^l}{L+1}\mathrm{p}_k\mathrm{p}_k^T. \tag{14}$$

$\frac{\sum_{l=\{0,2,\cdots\}}\sigma_k^l}{L+1}$ and $\frac{\sum_{l=\{1,3,\cdots\}}\sigma_k^l}{L+1}$ can be considered as weights of singular vectors when considering even and odd hop neighbors, respectively. We illustrate the normalized weights in Figure 2 (a) and (b), and make the following observation:

Observation 1. *As stacking more graph convolution layers, the goal of GCNs is to learn a low-rank representation by stressing (suppressing) more components with larger (smaller) singular values.*

We further observe that:

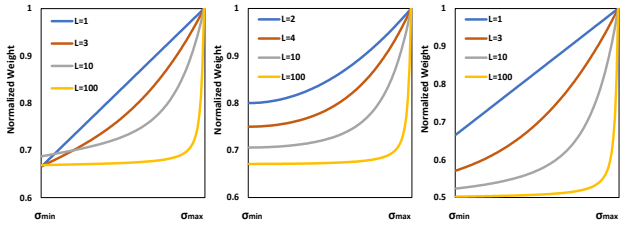
$$\mathrm{O}_u = \left(\mathrm{p}_{u*}^T \odot \frac{\sum_{l=\{0,2,\cdots\}}\sigma^l}{L+1}\right)\mathrm{P}^T\mathrm{E}_U + \left(\mathrm{p}_{u*}^T \odot \frac{\sum_{l=\{1,3,\cdots\}}\sigma^l}{L+1}\right)\mathrm{Q}^T\mathrm{E}_I, \tag{15}$$

where σ is a vector containing all singular values, p_{u*}^T is the u-th row vector, $\odot$ represents the element-wise multiplication. We can see $\mathrm{P}^T\mathrm{E}_U$ and $\mathrm{Q}^T\mathrm{E}_I$ are common terms for distinct users/items, what makes representations unique lies in the term in parentheses.

Assumption 1. $\mathrm{P}^T\mathrm{E}_U$ *and* $\mathrm{Q}^T\mathrm{E}_I$ *are redundant.*

On the other hand, the above two terms play a important role constituting the core design of GCNs (i.e., neighborhood aggregation), replacing or removing them leads to a new learning paradigm without explicitly aggregating neighborhood. To verify this assumption, we evaluate three models: (1) the original model Equation (13); (2) we simply replace $\mathrm{P}^T\mathrm{E}_U$ and $\mathrm{P}^T\mathrm{E}_I$ with two different weight matrices; (3) we use a shared weight matrix based on (2).

The results in Figure 1 (a) and (b) show that the performance of the three models are fairly close, and thus: (1) neighborhood



(a) Odd-hop neighbors. (b) Even-hop neighbors. (c) SVD-LightGCN.

Figure 2: Normalized weights of singular vectors.

aggregation is not necessary for GCNs; (2) The power of GCNs for CF does not heavily rely on model parameters, since reducing parameters (by half) does not reduce the accuracy and even results in faster convergence. Based on the model (3), we can merge the two terms in Equation (13) and simplify it as:

$$\begin{aligned} \mathrm{O}_U &= \mathrm{P} diag\left(\frac{\sum_{l=0}^{L}\sigma_k^l}{L+1}\right)\mathrm{W}, \\ \mathrm{O}_I &= \mathrm{Q} diag\left(\frac{\sum_{l=0}^{L}\sigma_k^l}{L+1}\right)\mathrm{W}, \end{aligned} \tag{16}$$

and name it SVD-LightGCN. We can interpret it as a two-step procedure. We first obtain a weighted singular matrices by assigning the weight $\frac{\sum_{l=0}^{L}\sigma_k^l}{L+1}$ to singular vectors (i.e., p_k and q_k); then, we learn a condensed embeddings of the singular vectors through a feature transformation W. Figure 2 (c) shows the goal of SVD-LightGCN is also to learn a low-rank representation, where the weights of singular vectors are adjustable through L. We also observe that:

Observation 2. *SVD is a special case of SVD-LightGCN where* $\mathrm{W} = \mathrm{I}$ *and* $l = L = \frac{1}{2}$ *(fixed to a square root).*

3.2 Analysis on SVD-LightGCN

Training Efficiency. Observation 1 provides an alternative way to build GCNs, that we can directly focus on the weights over singular vectors instead of stacking layers. However, retrieving all singular vectors is computationally expensive and not applicable on large datasets as well. On the other hand, Observation 1 implies that most small singular values are not so helpful for recommendation. To further verify this observation, we compare SVD-LightGCN and SVD-LightGCN-T which only exploits K-largest singular values and vectors, and report the accuracy of them in Figure 1 (c) and (d), where x-axis represents the singular vector ratio: $\frac{K}{\min(|\mathcal{U}|,|\mathcal{I}|)}$. We can see SVD-LightGCN-T with only the top 1% largest singular values and vectors outperforms SVD-LightGCN which exploits all singular vectors, and the best accuracy is achieved at 4% on CiteULike, 6% on ML-100K. This finding not only shows that most small singular values and vectors are noisy that even reduces the performance, but also helps largely reduce the training cost and improve the training efficiency. For instance, retrieving 4% of the singular vectors and values only takes 1.8s on CiteULike, the learning parameters (Kd) are merely 1% of that of MF and LightGCN ($|\mathcal{U}|\,d + |\mathcal{I}|\,d$).

Over-Smoothing. Users and items tend to have the same representations when the model layer L is large enough [22].

Theorem 2. *The maximum singular value of* $\tilde{\mathrm{R}}$ *is 1.*

As shown from Figure 2 (b), the larger singular values are further emphasized as increasing the model layers. Following Theorem 2, if we further increase the layer L:

$$\lim_{L\to\infty} = \frac{\sum_{l=0}^{L}\frac{\sigma_k^l}{L+1}}{\sum_{l=0}^{L}\frac{\sigma_{\max}^l}{L+1}} \to 0, \tag{17}$$

where the weights of any singular vectors are reduced to 0 compared with the largest one $\sigma_{\max}$, where user/item representations are only contributed by the largest singular vector. Thus, increasing model layers does not necessarily lead to better representations and might instead cause information loss. The over-smoothing issue lies in the gap between singular values, where it is enlarged as stacking layers, which suppresses some important information that matters for recommendation. To alleviate this issue, we define a renormalized interaction matrix as: $\dot{\mathrm{R}} = (\mathrm{D}_U + \alpha\mathrm{I})^{-\frac{1}{2}}\mathrm{R}(\mathrm{D}_I + \alpha\mathrm{I})^{-\frac{1}{2}}$ where $\alpha \geq 0$.

Theorem 3. *Given the singular value* $\dot{\sigma}_k$ *of* $\dot{\mathrm{R}}$, $\dot{\sigma}_{\max} \leq \frac{d_{\max}}{d_{\max}+\alpha}$ *where* $d_{\max}$ *is the maximum node degree.*

The maximum singular value becomes smaller as increasing α, indicating a smaller gap. On the other hand, a too small gap fails to emphasize the difference of importance of different components (i.e., the component with a larger singular value is more important). Thus, we can adjust α to regulate the gap to assure that important information is well preserved and to adapt to different datasets.

Furthermore, the weighting function is a crucial design as it controls the weights of singular vectors, while LightGCN adopts a polynomial in a heuristic way. Let $\psi(\cdot)$ denotes the weighting function. Basically, we can parameterize $\psi(\cdot)$ with advanced algorithms to dynamically learn the weights of singular vectors. Alternatively, if we consider $\psi(\cdot)$ as a static continuous function of singular values σ_k, it is expected to weight the singular vectors through a function with easy-to-adjust hyperparameters instead of by repeatedly increasing the model layer L. In addition, by replacing the polynomial in LightGCN with $\psi(\cdot)$, following the Taylor series $\psi(\sigma_k) = \sum_{l=0}^{L}\alpha_l\sigma_k^l$, we can rewrite Equation (8) as:

$$\mathrm{O} = \left(\sum_{l=0}^{L}\alpha_l\tilde{\mathrm{A}}^l\right)\mathrm{E}, \tag{18}$$

where α_l is $\psi(\sigma_k)$'s l-th order derivative at 0, L is $\psi(\cdot)$'s highest order. From a spatial perspective, α_l is also the contribution of l-th order neighborhood, and L corresponds to the farthest neighborhood being incorporated. Intuitively, it is expected that user/item representations are constructed from as many positive neighborhood signals as possible (i.e., $\alpha_l > 0$ and $L \to \infty$), implying that $\psi(\cdot)$ is infinitely differentiable with any-order derivatives positive.

3.3 SVD-GCN

Based on the analysis in Section 3.2, we formulate the user and item representations as follows:

$$\begin{aligned} \mathrm{O}_U &= \dot{\mathrm{P}}^{(K)} diag\left(\psi\left(\dot{\sigma_k}\right)\right) \mathrm{W}, \\ \mathrm{O}_I &= \dot{\mathrm{Q}}^{(K)} diag\left(\psi\left(\dot{\sigma_k}\right)\right) \mathrm{W}, \end{aligned} \tag{19}$$

where $\dot{\mathrm{P}}^{(K)}$ and $\dot{\mathrm{Q}}^{(K)}$ are composed of K-largest left and right singular vectors of $\dot{\mathrm{R}}$, respectively. Our initial attempt is to dynamically model the importance of singular vectors through a neural network given singular values as the input. However, we found that such a design underperforms static designs in most cases, and speculate that the reason is due to the data sparsity on CF. Unlike other recommendation tasks with rich side information, the only available data is the user/item ID besides interactions, which increases the difficulty to learn the intrinsic data characteristics. Based on previous analysis in Section 3.2, extensive experiments show that an exponential kernel [20] achieves superior accuracy on the tested data, thus we set $\psi(\dot{\sigma_k}) = e^{\beta\dot{\sigma_k}}$, where β is a hyperparameter to adjust the extent of emphasis over larger singular values (i.e., a larger (smaller) β emphasizes the importance of larger (smaller) singular values more). We will also compare different $\psi(\cdot)$ designs in Section 4.3. Unlike conventional GCNs updating all user/item embeddings simultaneously in a matrix form resulting in a large spatial complexity, we can train SVD-GCN in a node form with more flexibility as:

$$\begin{aligned} \mathrm{o}_u &= \dot{\mathrm{p}}_u^T \odot \left(e^{\beta\dot{\sigma}}\right) \mathrm{W}, \\ \mathrm{o}_i &= \dot{\mathrm{q}}_i^T \odot \left(e^{\beta\dot{\sigma}}\right) \mathrm{W}, \end{aligned} \tag{20}$$

where $\dot{\mathrm{p}}_u^T$ and $\dot{\mathrm{q}}_i^T$ are the rows of $\dot{\mathrm{P}}^{(K)}$ and $\dot{\mathrm{Q}}^{(K)}$, respectively; $\dot{\sigma}$ is a vector containing all singular values. Note that the element-wise multiplication does not involve parameters thus can be pre-computed. Then, inspired by BPR loss [26], we formulate the loss function as follows:

$$\mathcal{L}_{main} = \sum_{u\in\mathcal{U}} \sum_{(u,i^+)\in\mathrm{R}^+,(u,i^-)\notin\mathrm{R}^+} \ln\sigma\left(\mathrm{o}_u^T\mathrm{o}_{i^+} - \mathrm{o}_u^T\mathrm{o}_{i^-}\right). \tag{21}$$

As shown in Equation (10), in GCN-based CF methods, user/item representations are contributed from three kinds of information flows: user-item, user-user, and item-item relations. Thus, besides the user-item relations, homogeneous (i.e., user-user and item-item) relations also help increase model effectiveness. We define a user-user $\mathcal{G}_U = (\mathcal{V}_U, \mathcal{E}_U)$, and an item-item graph $\mathcal{G}_I = (\mathcal{V}_I, \mathcal{E}_I)$, where $\mathcal{V}_U = \mathcal{U}$ and $\mathcal{V}_I = \mathcal{I}$; $\mathcal{E}_U = \{(u,g)|g \in \mathcal{N}_i, i \in \mathcal{N}_u\}$ and $\mathcal{E}_I = \{(i,h)|h \in \mathcal{N}_u, u \in \mathcal{N}_i\}$, where $\mathcal{N}_u$ and $\mathcal{N}_i$ are the sets of directly connected neighbors for u and i, respectively. Naturally, we can define the normalized adjacency matrix of $\mathcal{G}_U$ and $\mathcal{G}_I$ as $\mathrm{R}_U = \dot{\mathrm{R}}^T\dot{\mathrm{R}}$ and $\mathrm{R}_I = \dot{\mathrm{R}}\dot{\mathrm{R}}^T$, respectively. According to Equation (26) in Section 7, the eigenvectors of R_U and R_I are actually $\dot{\mathrm{R}}$'s left and right singular vectors, respectively; and the eigenvalues are both the square of $\dot{\mathrm{R}}$'s singular values. Thus, $\mathcal{G}$, $\mathcal{G}_U$ and $\mathcal{G}_I$ are closely connected. We formulate the following loss to learn the relations on $\mathcal{G}_U$:

$$\mathcal{L}_{user} = \sum_{u\in\mathcal{U}} \sum_{(u,u^+)\in\mathcal{E}_U,(u,u^-)\notin\mathcal{E}_U} \ln\sigma\left(\mathrm{o}_u^T\mathrm{o}_{u^+} - \mathrm{o}_u^T\mathrm{o}_{u^-}\right). \tag{22}$$

Similarly, we learn the relations on $\mathcal{G}_I$ via the following loss:

$$\mathcal{L}_{item} = \sum_{i\in\mathcal{I}} \sum_{(i,i^+)\in\mathcal{E}_I,(i,i^-)\notin\mathcal{E}_I} \ln\sigma\left(\mathrm{o}_i^T\mathrm{o}_{i^+} - \mathrm{o}_i^T\mathrm{o}_{i^-}\right). \tag{23}$$

Finally, we propose the following four SVD-GCN variants:

$$\begin{aligned} &\text{SVD-GCN-B}: \ \mathcal{L} = \mathcal{L}_{\text{main}} + \lambda\left\|\Theta\right\|_2^2, \\ &\text{SVD-GCN-U}: \ \mathcal{L} = \mathcal{L}_{\text{main}} + \gamma\mathcal{L}_{\text{user}} + \lambda\left\|\Theta\right\|_2^2, \\ &\text{SVD-GCN-I}: \ \mathcal{L} = \mathcal{L}_{\text{main}} + \zeta\mathcal{L}_{\text{item}} + \lambda\left\|\Theta\right\|_2^2, \\ &\text{SVD-GCN-M}: \ \mathcal{L} = \mathcal{L}_{\text{main}} + \gamma\mathcal{L}_{\text{user}} + \zeta\mathcal{L}_{\text{item}} + \lambda\left\|\Theta\right\|_2^2, \end{aligned} \tag{24}$$

where Θ denotes the model parameters. Besides the above variants, to evaluate the effect of the feature transformation, we propose a non-parametric method SVD-GCN-S by removing W.

3.4 Discussion

3.4.1 Model Complexity. The complexity of SVD-GCN mainly comes from two parts. We first retrieve K singular vectors through SVD for the low-rank matrix [8], with a complexity as: $O(K\left|\mathrm{R}^+\right| + K^2\left|\mathcal{U}\right| + K^2\left|\mathcal{I}\right|)$. We run the algorithm on GPU and only require a very few singular vectors, which only costs several seconds. Except for SVD-GCN-S, other variants require training with time complexity as $O(c\left|\mathrm{R}^+\right|(K+1)d)$, which is comparable to MF: $c\left|\mathrm{R}^+\right|d$, where c denotes the number of epochs. On the other hand, the model parameters of MF is $\frac{|\mathcal{U}|+|\mathcal{I}|}{K}$ time that of GCN-SVD. Overall, SVD-GCN is lighter than MF, and we will show more quantitative results in terms of efficiency in Section 4.2.

3.4.2 Comparison with GCN-based CF Methods. Compared with conventional GCN-based methods, GCN-SVD replaces neighborhood aggregation with a truncated SVD and significantly reduces the model parameters. Overall, SVD-GCN is equipped with a lighter structure and more scalable. Recent proposed work UltraGCN [24] simplifies LightGCN by replacing neighborhood aggregation with a weighted MF and shows lower complexity:

$$\max \sum_{u\in\mathcal{U},i\in\mathcal{N}_u} \beta_{u,i}\mathrm{e}_u^T\mathrm{e}_i, \tag{25}$$

where $\beta_{u,i}$ is obtained from single-layer LightGCN. However, UltraGCN improves based on single-layer LightGCN, which can only exploit the first order neighborhood and losses the ability of incorporating high-order neighborhood to augment training interactions. On the other hand, SVD-GCN is derived from any-layer LightGCN and we further generalize it to the situation of infinite layers, hence maximizes the power of GCNs.

4 EXPERIMENTS

In this section, we comprehensively evaluate our proposed SVD-GCN. The rest of this section is organized as follows: we introduce experimental settings in Section 4.1, compare baselines with SVD-GCN in terms of recommendation accuracy and training efficiency in Section 4.2; in Section 4.3, we dissect SVD-GCN to show the effectiveness of our proposed designs and how different hyperparameter settings (i.e., K, α, β, γ, and ζ) affect performance.

Table 1: Statistics of datasets

Datasets	#User	#Item	#Interactions	Density%
CiteULike	5,551	16,981	210,537	0.223
ML-100K	943	1,682	100,000	6.305
ML-1M	6,040	3,952	1,000,209	4.190
Yelp	25,677	25,815	731,672	0.109
Gowalla	29,858	40,981	1,027,370	0.084

4.1 Experimental Settings

4.1.1 Datasets and Evaluation Metrics. We use five public datasets in this work, where the results of Figure 1 are based on CiteULike[1] and ML-100K [9]. To demonstrate the effectiveness of our proposed methods on more datasets and to justify the previous analysis, we evaluate SVD-GCN on three other datasets: Gowalla [34], Yelp [12], and ML-1M [9]. Since we focus on implicit feedback, we only keep user/item ID and transform feedbacks to binary ratings. Table 1 lists statistics of datasets.

We adopt two widely-used metrics: Recall and nDCG [14] to evaluate our methods. Recall measures the ratio of the relevant items in the recommended list to all relevant items in test sets, while nDCG takes the ranking into consideration by assigning higher scores to items ranking higher. The recommendation list is generated by ranking unobserved items and truncating at position k. Since the advantage of GCN-based methods over traditional CF methods is the ability of leveraging high-order neighborhood to augment training data, thereby alleviating the data sparsity, we only use 20% of interactions for training and leave the remaining for test to evaluate the model robustness and stability; we randomly select 5% from the training set as validation set for hyper-parameter tuning and report the average accuracy on test sets.

4.1.2 Baselines. We compare our methods with the following competing baselines, where the hyperparameter settings are based on the results of the original papers:

- BPR [26]: This is a stable and classic MF-based method, exploiting a Bayesian personalized ranking loss for personalized rankings.
- EASE [29]: This is a neighborhood-based method with a closed form solution and show superior performance to many traditional CF methods.
- LightGCN [10]: This method uses a light GCN architecture for CF by removing activations functions and feature transformation. We use a three-layer architecture as the baseline.
- LCFN [39]: This model replaces the original graph convolution with a low pass graph convolution to remove the noise from interactions for recommendation. We set $F = 0.005$ and use a single-layer architecture.
- SGL-ED [35]: This model generates different node views by randomly removing the edge connections and maximizes their agreements, and the proposed self-supervised loss is implemented on LightGCN [10]. We set $\tau = 0.2$, $\lambda_1 = 0.1$, $p = 0.1$, and use a three-layer architecture.
- UltraGCN [24]: This model simplifies LightGCN by replacing neighborhood aggregation with a weighted MF, which shows faster convergence and less complexity.

We remove some popular GCN-based methods such as Pinsage [38], NGCF [34], and SpectralCF [41] as aforementioned baselines have already shown superiority over them.

4.1.3 Implementation Details. We implemented the proposed model based on PyTorch[2] and released the code on Github[3]. For all models, We use SGD as the optimizer, the embedding size d is set to 64, the regularization rate λ is set to 0.01 on all datasets, the learning rate is tuned amongst $\{0.001, 0.005, 0.01, \cdots, 1\}$; without specification, the model parameters are initialized with Xavier Initialization [7]; the batch size is set to 256. We report other hyperparameter settings in the next subsection.

4.2 Comparison

4.2.1 Performance. We report the accuracy of baselines and our proposed GCN-SVD variants in Table 2, and have the following observations:

- Overall, GCN-based methods outperforms traditional CF methods, indicating the effectiveness of GCNs for CF and demonstrating the importance of augmenting training interactions by incorporating high-order neighborhood information, thereby alleviating data sparsity.
- Among all baselines, SGL-ED achieves the best across all datasets, while our proposed SVD-GCNs show consistent improvements over SGL-ED, indicating the effectiveness and superiority over conventional GCN designs. UltraGCN shows relatively poor performance among GCN-based methods. As shown in our previous analysis in Section 3.4.2, UltraGCN improves based on single-layer GCN which fails to leverage the higher-order neighborhood, thus cannot perform stably with limited interactions.
- Since our key contribution is to replace neighborhood aggregation, the improvement is more clear if we compare with pure GCN-based methods such as LightGCN. SVD-GCN outperforms LightGCN on Yelp, ML-1M, and Gowalla by 53.6%, 11.7%, and 29.0%, respectively, in terms of nDCG@10. The improvements over sparse data tend to be more significant, indicating the stability of SVD-GCN under extreme data sparsity.
- Among SVD-GCN variants, the basic model SVD-GCN-B and SVD-GCN-S already outperform all baselines by a large margin. In addition, introducing user-user and item-item relations results in further improvement. We also notice that mixing user-user and item-item relations does not necessarily leads to better accuracy, and we speculate that the reason might be related to the data density. On the dense data such as ML-1M where the user-item interactions are relatively sufficient, the improvement by introducing user-user and item-item relations is not as significant as that of sparser datasets, and incorporating both relations even performs worse; while on the sparest data Gowalla, introducing auxiliary relations shows consistent improvements.

4.2.2 Training Efficiency. The results shown in this subsection are obtained on a machine equipped with AMD Ryzen 9 5950X and

[1] https://github.com/js05212/citeulike-a

[2] https://pytorch.org/

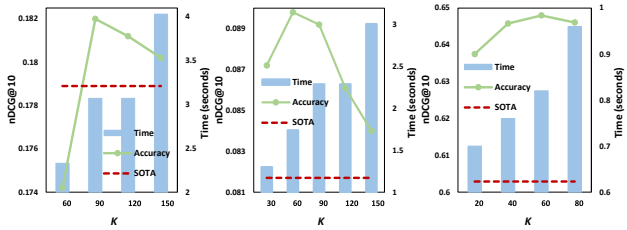
[3] https://github.com/tanatosuu/svd_gcn

Table 2: Overall performance comparison.

	Yelp				ML-1M				Gowalla			
	nDCG@k		Recall@k		nDCG@k		Recall@k		nDCG@k		Recall@k	
	k=10	k=20	k=10	k=20	k=10	k=20	k=10	k=20	k=10	k=20	k=10	k=20
BPR	0.0388	0.0374	0.0371	0.0370	0.5521	0.4849	0.5491	0.4578	0.1086	0.0907	0.0917	0.0743
Ease	0.0360	0.0362	0.0346	0.0368	0.3773	0.3249	0.3682	0.3000	0.0722	0.0670	0.0680	0.0642
LCFN	0.0617	0.0627	0.0613	0.0653	0.5927	0.5197	0.5887	0.4898	0.1305	0.1132	0.1144	0.0980
UltraGCN	0.0417	0.0403	0.0404	0.0403	0.5326	0.4688	0.5302	0.4434	0.0977	0.0815	0.0841	0.0681
LightGCN	0.0679	0.0680	0.0669	0.0704	0.5917	0.5261	0.5941	0.5031	0.1477	0.1327	0.1368	0.1224
SGL-ED	0.0817	0.0794	0.0784	0.0792	0.6029	0.5314	0.6010	0.5035	0.1789	0.1561	0.1563	0.1353
SVD-GCN-S	0.0919	0.0895	0.0894	0.0903	0.6458	0.5702	0.6466	0.5421	0.1900	0.1677	0.1690	0.1484
SVD-GCN-B	0.0898	0.0876	0.0866	0.0879	0.6480	0.5724	0.6484	0.5443	0.1820	0.1607	0.1628	0.1428
SVD-GCN-U	0.0923	0.0897	0.0888	0.0898	0.6571	**0.5791**	**0.6571**	**0.5495**	0.1875	0.1654	0.1667	0.1460
SVD-GCN-I	0.0930	0.0907	0.0897	0.0910	**0.6574**	0.5770	0.6565	0.5465	0.1857	0.1646	0.1662	0.1466
SVD-GCN-M	**0.0941**	**0.0917**	**0.0908**	**0.0921**	0.6521	0.5705	0.6490	0.5377	**0.1905**	**0.1681**	**0.1693**	**0.1487**
Improv.%	+15.18	+15.49	+15.82	+16.29	+9.04	+8.98	+9.33	+9.14	+6.48	+7.69	+8.32	+9.90

Table 3: Training time comparison on Gowalla.

Model	Time/Epoch	Epochs	Running Time	Parameters
LightGCN	6.43s	600	3,858s	4.5m
UltraGCN	2.55s	90	229.5s	4.5m
BPR	1.04s	250	260.0s	4.5m
SVD-GCN-S	0.00s	0	3.07s	0.0k
SVD-GCN-B	1.28s	8	13.31s	5.7k
SVD-GCN-U	2.06s	8	19.55s	5.7k
SVD-GCN-I	2.18s	8	20.51s	5.7k
SVD-GCN-M	3.05s	8	27.47s	5.7k



(a) Gowalla (b) Yelp. (c) MovieLens.

Figure 3: How the preprocessing time and accuracy (nDCG@10) vary on K on SVD-GCN-B.

GeForce RTX 3090. Figure 3 shows how the preprocessing time and accuracy change with K, where SOTA is the best baseline. The best accuracy is achieved at $K = 90$, $K = 60$, and $K = 60$, where the preprocessing time is 3.07s, 0.82s, and 1.74s, on Gowalla, ML-1M, and Yelp, respectively. Overall, only 1% singular vectors are required on ML-1M, and less than 0.5% singular vectors are required on Gowalla and Yelp, when the model reaches the best accuracy.

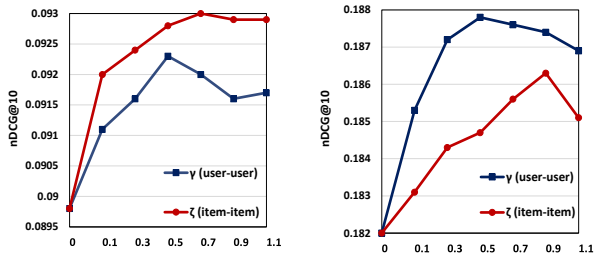
Table 3 shows the training time and running epochs of several methods, where the running time includes both preprocessing and training time. Overall, LightGCN is the most time consuming model (3,858s) as it is a conventional GCN model; SVD-GCN-S is the most time efficient model (3.07s) since it does not require model optimization and shows over 1000x speed-up over LightGCN. BPR is the fastest model (1.04s) in terms of training time per epoch, while it still requires hundreds epochs to reach the best accuracy due to the large amount of parameters need to be optimized. Although SVD-GCN variants (excluding SVD-GCN-S) are slightly slower than BPR on training time per epoch, they show fast training convergence as the model parameters are only 0.08% of that of BPR.

4.3 Model Analysis

4.3.1 How Homogeneous Relations Affect Performance? The direct comparison between SVD-GCN-B and SVD-GCN-U, SVD-GCN-I, and SVD-GCN-M demonstrates the positive effect of homogeneous relations. Furthermore, Figure 4 shows how different γ and ζ affect the accuracy, where the accuracy increases first then drops as constantly increasing the value of γ and ζ. The best accuracy is achieved at $\gamma = 0.5$, while the optimal ζ (0.9 on Gowalla and 0.7 on Yelp) is larger than γ. One reasonable explanation is that item-item relations are usually sparser (0.21% on Gowalla and 0.33% on Yelp) than user-user relations (0.41% on Gowalla and 0.48% on Yelp).

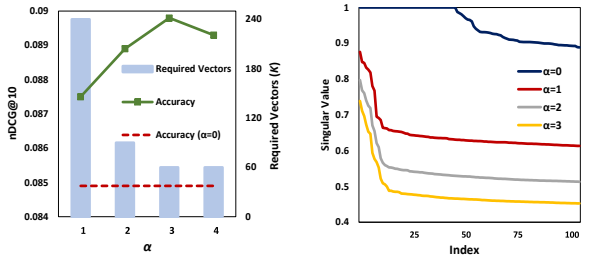
4.3.2 Do We Need Feature Transformation? By comparing SVD-GCN-S and SVD-GCN-B, we can see W results in worse accuracy on Gowalla and Yelp and only a slight improvement on ML-1M, which shows that feature transformation does not help much learn user-item interactions. On the other hand, we can identify the positive effect of W when incorporating user-user and item-item relations, which leads to improvement compared with SVD-GCN-B. We speculate that the ineffectiveness of feature transformation is related to the data density, where the intrinsic characteristic of sparse data such as user-item interactions is difficult to learn, while user-user and item-item relations are much denser thus is easier to learn. Overall, SVD-GCN can achieve superior accuracy without any model training, implying that the key design making GCN effective for recommendation lies in a good low-rank representation.

4.3.3 Effect of Renormalization Trick. We have two observations from Figure 5 (a): as increasing α (i.e., shrinking the singular value gap), (1) the accuracy increases first then drops, reaches the best at



(a) Yelp. (b) Gowalla.

Figure 4: Effect of γ and ζ.



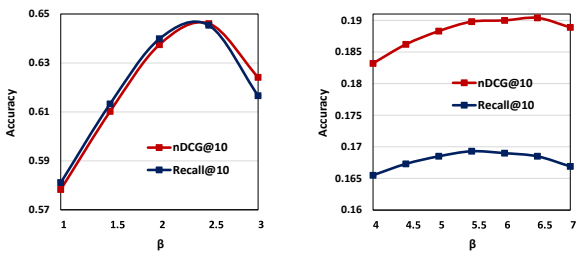
(a) How K and accuracy (nDCG@10) vary on α on SVD-GCN-B. (b) Distribution of top 100 singular values with varying α.

Figure 5: Effect of renormalization trick on Yelp.

$\alpha = 3$; (2) the model tends to require fewer singular vectors. In Figure 5 (b), as increasing α, (1) the maximum singular value becomes smaller, which is consistent with Theorem 3; (2) singular values drops more quickly, which explains why fewer singular vectors are required. For instance, the model with $\alpha = 0$ has more large singular values which contribute significantly to the interactions compared with the model with $\alpha > 0$, thus more singular vectors are required; while the important large singular values are fewer as increasing α. In other words, the important information is concentrated in fewer top singular values when we constantly increase α. Surprisingly, we have the same observation on other datasets. Theoretical analysis on this interesting phenomenon is beyond the scope of this work, we leave it for future work.

4.3.4 Effect of β. Figure 6 shows the accuracy with varying β. The accuracy first increases as increasing β, then starts dropping after reaching the best performance at $\beta = 2.5$ on ML-1M, $\beta = 6.0$ on Gwoalla; there is a similar trend on Yelp that the best accuracy is achieved at $\beta = 4.0$. We observe that β tends to be larger on sparser data, implying that the large singular values are more important on the sparser data. We speculate that there is less useful information on sparser datasets, thus the small singular values contain more noise and should be depressed more than denser datasets.

4.3.5 The Choice of Weighting Function. We show the accuracy of SVD-GCN-S with different weighting functions in Table 4. For dynamic designs, we use a neural network to attempt to model the importance of singular vectors with singular values as the input, while it underperforms most static designs, showing that the dynamic design is not suitable for the weighting function. For



(a) ML-1M. (b) Gowalla.

Figure 6: Effect of β on SVD-GCN-S.

Table 4: Accuracy of different weighting functions on Yelp.

Design	Function	nDCG@10	Property (1) Increasing	Property (2) Pos Coef.	Property (3) Infinite
Static	$\log(\beta\sigma_k)$	0.0882	✓	×	✓
	$\sum_l^L \sigma_k^l$	0.0899	✓	✓	×
	$\frac{1}{1-\beta\sigma_k}$	0.0919	✓	✓	✓
	$e^{\beta\sigma_k}(\beta > 0)$	0.0919	✓	✓	✓
	$e^{\beta\sigma_k}(\beta < 0)$	0.0828	×	×	✓
Dynamic	Neural Network	0.0850			

static designs, following the previous analysis in Section 3.2, we list some properties that matter to accuracy: (from left to right) if the function (1) is increasing, (2) has positive taylor coefficients, (3) is infinitely differentiable, and evaluate some functions, where the setting of β is based on the best accuracy of each function. We can see the importance of the three properties is (1)≫(2)>(3). (1) implies that the larger singular values are assigned higher weights, which is important according to the previous analysis; (2) and (3) suggest if the model can capture neighborhood from any-hops with positive contributions. Overall, the importance of the three properties is (1)≫(2)>(3), and the functions satisfying all three properties perform the best.

5 RELATED WORK

Collaborative Filtering (CF) is an extensively used technique in modern recommender systems. Early memory-based CF methods [27] predict user preference by computing the similarity between users or items. Later on, model-based methods become prevalent [21] which characterizes users and items as latent vectors and calculate their dot products to predict the unobserved ratings. Subsequent works focus on modeling complex user-item interactions with advanced algorithms, such as neural network [11, 37], attention mechanism [18], transformer [30], and so on. Behind the learning in Euclidean space, some methods [32] explore the potential of learning in non-Euclidean space. On another line, auxiliary information such as social relations [23], review data [1], temporal information [18] etc. is also well incorporated to obtain a better understanding of user preference.

The data sparsity issue on recommendation datasets limits the aforementioned traditional CF methods. The development of GCNs helps alleviate this issue by incorporating higher-order neighborhood to facilitate user/item representations, and thus much effort

has been made to adapt GCNs to recommendation. Early work such as GC-MC [2] accumulates messages from different neighbors based on the rating for explicit feedbacks; SpectralCF [41] adapts the original graph convolution to CF with implicit feedbacks; NGCF [34] improves based on vanilla GCN [19] by additionally encoding the interactions via an element-wise multiplication. To improve the scalability on large-scale datasets, Ying et al. [38] defines a flexible graph convolution on spatial domain without passing messages with adjacency matrix. By showing the redundancy of feature transformation and non-linear activation function, LightGCN [10] only keeps neighborhood aggregation for recommendation. Recent works fuse other research topics into GCNs, such as contrastive learning [35, 42], learning in hyperbolic space [31], negative sampling [13], graph signal processing [25], etc. and achieves further success.

Despite the superior performance that the aforementioned GCN-based methods have achieved, the computational cost of GCNs is much larger than traditional CF methods, making them unscalable on large-scale datasets. Although some works [4, 10] reduces the cost to some extent by removing feature transformation and non-linear activation functions, while the complexity mainly comes from the neighborhood aggregation, which is implemented by multiplying by an adjacency matrix. One recent work UltraGCN [24] further simplifies GCNs by replacing the neighborhood aggregation with a weighted MF, where the weight is obtained from a single-layer LightGCN, which significantly reduces the complexity. However, such a simplification degrades the power of GCNs as it can only capture the first-order neighborhood, and the experimental results also show its ineffectiveness under extreme sparsity. On the other hand, our proposed SVD-GCN is based on comprehensive theoretical and empirical analysis on LightGCN with any layers, whose superiority and effectiveness have been demonstrated through extensive experimental results.

6 CONCLUSION

In this work, we proposed a simplified and scalable GCN learning paradigm for CF. We first investigated what design makes GCN effective. Particularly, by further simplifying LightGCN, we showed that stacking graph convolution layers is to learn a low-rank representation by emphasizing (suppressing) more components with larger (smaller) singular values. Based on the close connection between GCN-based and low-rank methods, we proposed a simplified GCN formulation by replacing neighborhood aggregation with a truncated SVD, which only exploits K-largest singular values and vectors for recommendation. To alleviate over-smoothing issue, we proposed a renormalization trick to adjust the singular value gap, resulting in significant improvement. Extensive experimental results demonstrated the training efficiency and effectiveness of our propose methods.

We leave two questions for future work. Firstly, since SVD-GCN-S already achieves superior performance and feature transformation only shows positive effect learning user-user and item-item relations, we aim to incorporate user-user and item-item relations without introducing any model parameters (i.e., we improve based on SVD-GCN-S). In addition, we attempt to explain the phenomenon in Section 4.3.3, that why shrinking the singular value gap causes singular values to drop more quickly, thereby making important information to be concentrated in fewer singular vectors.

7 PROOFS

7.1 Proofs of Theorem 1

Proof. Following SVD, we know any two singular vectors are orthonormal (i.e., $\mathrm{P}\mathrm{P}^T = \mathrm{I}$ and $\mathrm{Q}\mathrm{Q}^T = \mathrm{I}$), thus it is easy to derive the following equations:

$$\begin{aligned} \tilde{\mathrm{R}}\tilde{\mathrm{R}}^T &= \mathrm{P} diag\left(\sigma_k^2\right)\mathrm{P}^T, \\ \tilde{\mathrm{R}}^T\tilde{\mathrm{R}} &= \mathrm{Q} diag\left(\sigma_k^2\right)\mathrm{Q}^T. \end{aligned} \tag{26}$$

By repeating the above Equations l times, we obtain Equation (11).

For simplicity, we let $\mathrm{R}' = \tilde{\mathrm{R}}\left(\tilde{\mathrm{R}}^T\tilde{\mathrm{R}}\right)^{\frac{l-1}{2}}$, and $\mathrm{R}'^T = \mathrm{R}^T\left(\tilde{\mathrm{R}}\tilde{\mathrm{R}}^T\right)^{\frac{l-1}{2}}$. We let P', Q' and σ_k' denote the stacked left singular vectors, right singular vectors and singular value for R', respectively. Following Equation (26), we can derive the following equations:

$$\begin{aligned} \mathrm{R}'\mathrm{R}'^T &= \left(\tilde{\mathrm{R}}\tilde{\mathrm{R}}^T\right)^l = \mathrm{P} diag\left(\sigma_k^{2l}\right)\mathrm{P}^T = \mathrm{P}' diag\left(\sigma_k'^2\right)\mathrm{P}'^T, \\ \mathrm{R}'^T\mathrm{R}' &= \left(\tilde{\mathrm{R}}^T\tilde{\mathrm{R}}\right)^l = \mathrm{Q} diag\left(\sigma_k^{2l}\right)\mathrm{Q}^T = \mathrm{Q}' diag\left(\sigma_k'^2\right)\mathrm{Q}'^T. \end{aligned} \tag{27}$$

It is easy to observe that $\mathrm{P}' = \mathrm{P}$, $\mathrm{Q}' = \mathrm{Q}$ and $\sigma_k' = \sigma_k^l$. Then, according to SVD, we derive Equation (12).

□

7.2 Proofs of Theorem 2 and 3

Proof. We first introduce Rayleigh quotients [28]:

$$\lambda_{\min} \le \mathrm{x}^T\tilde{\mathrm{A}}\mathrm{x} \le \lambda_{\max} \quad s.t.\ |\mathrm{x}| = 1, \tag{28}$$

where $\lambda_{\min}$ and $\lambda_{\max}$ are the minimum and maximum eigenvalues of $\tilde{\mathrm{A}}$, respectively. Then, we can show $\lambda_{\max} = 1$:

$$1 - \mathrm{x}^T\tilde{\mathrm{A}}\mathrm{x} = \mathrm{x}^T\mathrm{x} - \mathrm{x}^T\tilde{\mathrm{A}}\mathrm{x} = \sum_{(u,i)\in\mathcal{E}}\left(\frac{x_u}{\sqrt{d_u}} - \frac{x_i}{\sqrt{d_i}}\right)^2 \ge 0. \tag{29}$$

In the meanwhile, we have the following observation:

$$\tilde{\mathrm{A}}\begin{bmatrix}\mathrm{p}_k\\ \mathrm{q}_k\end{bmatrix} = \begin{bmatrix}\tilde{\mathrm{R}}\mathrm{q}_k\\ \tilde{\mathrm{R}}^T\mathrm{p}_k\end{bmatrix} = \sigma_k\begin{bmatrix}\mathrm{p}_k\\ \mathrm{q}_k\end{bmatrix}, \tag{30}$$

which implies that $\sigma_k \in \{\lambda_{\min}, \cdots, \lambda_{\max}\} \le 1$ with $[\mathrm{p}_k, \mathrm{q}_k]^T$ as the eigenvector. By observing the eigenvector of $\lambda_{\max}$, if $\lambda_{\max}$ is also a singular value, we have: $\mathrm{p}_k = \sqrt{\mathrm{D}_U}1$ and $\mathrm{q}_k = \sqrt{\mathrm{D}_I}1$ where 1 is a vector with all 1 elements. It is easy to verify the solution satisfies SVD: $\tilde{\mathrm{R}}\mathrm{q}_k = \mathrm{p}_k$, thus $\sigma_{\max} = 1$.

Given $\dot{\mathrm{R}}$, we can define the corresponding adjacency matrix $\dot{\mathrm{A}}$. Since the relation in Equation (30) still holds between $\dot{\mathrm{R}}$ and $\dot{\mathrm{A}}$, we only need to prove $\dot{\lambda}_{\max} \le \frac{d_{\max}}{d_{\max}+\alpha}$.

$$\begin{aligned} \mathrm{x}^T\dot{\mathrm{A}}\mathrm{x} &= \sum_{(u,i)\in\mathcal{E}}\frac{2x_ux_i}{\sqrt{d_u+\alpha}\sqrt{d_i+\alpha}} \le \sum_{u\in\mathcal{V}}\frac{d_u}{d_u+\alpha}x_u^2, \\ &= 1 - \sum_{u\in\mathcal{V}}\frac{\alpha}{d_u+\alpha}x_u^2 \le 1 - \frac{\alpha}{d_{\max}+\alpha} = \frac{d_{\max}}{d_{\max}+\alpha}. \end{aligned} \tag{31}$$

= holds when $\alpha = 0$. When $\alpha > 0$, $\dot{\lambda}_{\max} < \frac{d_{\max}}{d_{\max}+\alpha}$, since x takes different values at $\sum_{(u,i)\in\mathcal{E}} \frac{2x_u x_i}{\sqrt{d_u+\alpha}\sqrt{d_i+\alpha}} = \sum_{u\in\mathcal{V}} \frac{d_u}{d_u+\alpha} x_u^2$ and $1 - \sum_{u\in\mathcal{V}} \frac{\alpha}{d_u+\alpha} x_u^2 = 1 - \frac{\alpha}{d_{\max}+\alpha}$.

□

REFERENCES

[1] Yang Bao, Hui Fang, and Jie Zhang. 2014. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-14)*. 2–8.

[2] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).

[3] 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 *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR'17)*. 335–344.

[4] 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 *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI-20)*. 27–34.

[5] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *Proceedings of the 28th International Conference on World Wide Web (WWW'19)*. 417–426.

[6] Shanshan Feng, Gao Cong, Bo An, and Yeow Meng Chee. 2017. Poi2vec: Geographical Latent Representation for Predicting Future Visitors. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-17)*. 102–108.

[7] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS'10)*. 249–256.

[8] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM review* 53, 2 (2011), 217–288.

[9] F Maxwell Harper and Joseph A Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5, 4 (2015), 1–19.

[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 639–648.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW'17)*. 173–182.

[12] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *Proceedings of the 39th international ACM SIGIR conference on Research and development in Information Retrieval (SIGIR'16)*. 549–558.

[13] 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 *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD'21)*. 665–674.

[14] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[15] Shuyi Ji, Yifan Feng, Rongrong Ji, Xibin Zhao, Wanwan Tang, and Yue Gao. 2020. Dual channel hypergraph collaborative filtering. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'20)*. 2020–2029.

[16] Meng Jiang, Peng Cui, Rui Liu, Qiang Yang, Fei Wang, Wenwu Zhu, and Shiqiang Yang. 2012. Social Contextual Recommendation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM'12)*. 45–54.

[17] Ruoming Jin, Dong Li, Jing Gao, Zhi Liu, Li Chen, and Yang Zhou. 2021. Towards a Better Understanding of Linear Models for Recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD'21)*. 776–785.

[18] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive Sequential Recommendation. In *In Proceedings of IEEE International Conference on Data Mining (ICDM'18)*. 197–206.

[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)*.

[20] Risi Imre Kondor and John D. Lafferty. 2002. Diffusion Kernels on Graphs and Other Discrete Input Spaces. In *Proceedings of the 19th International Conference on Machine Learning (ICML'02)*. 315–322.

[21] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 8 (2009), 30–37.

[22] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*. 3538–3545.

[23] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. 2008. Sorec: Social Recommendation Using Probabilistic Matrix Factorization. In *Proceedings of the 17th ACM conference on Information and Knowledge Management (ICDM'08*. 931–940.

[24] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM'21)*. 1253–1262.

[25] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. 2022. Less is More: Reweighting Important Spectral Graph Features for Recommendation. In *Proceedings of the 45th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR'22)*. 1273–1282.

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI'09)*. 452–461.

[27] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW'01)*. 285–295.

[28] Daniel Spielman. 2012. Spectral graph theory. *Combinatorial scientific computing* 18 (2012).

[29] Harald Steck. 2019. Embarrassingly shallow autoencoders for sparse data. In *Proceedings of the 28th International Conference on World Wide Web (WWW'19)*. 3251–3257.

[30] 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 *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM'19)*. 1441–1450.

[31] Jianing Sun, Zhaoyue Cheng, Saba Zuberi, Felipe Pérez, and Maksims Volkovs. 2021. HGCF: Hyperbolic Graph Convolution Networks for Collaborative Filtering. In *Proceedings of the 30th International Conference on World Wide Web (WWW '21)*. 593–601.

[32] Lucas Vinh Tran, Yi Tay, Shuai Zhang, Gao Cong, and Xiaoli Li. 2020. HyperML: A Boosting Metric Learning Approach in Hyperbolic Space for Recommender Systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM'20)*. 609–617.

[33] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. 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 (SIGIR'17)*. 515–524.

[34] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval (SIGIR'19)*. 165–174.

[35] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-Supervised Graph Learning for Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'21)*. 726–735.

[36] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A Neural Influence Diffusion Model for Social Recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 235–244.

[37] Hong-Jian Xue, Xinyu Dai, et al. 2017. Deep Matrix Factorization Models for Recommender Systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17)*. 3203–3209.

[38] 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 *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'18)*. 974–983.

[39] Wenhui Yu and Zheng Qin. 2020. Graph Convolutional Network for Recommendation with Low-pass Collaborative Filters. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. 10936–10945.

[40] Lingxiao Zhao and Leman Akoglu. 2020. PairNorm: Tackling Oversmoothing in GNNs. In *Proceedings of the 8th International Conference on Learning Representations (ICLR'20)*.

[41] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S Yu. 2018. Spectral Collaborative Filtering. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys'18)*. 311–319.

[42] Ding Zou, Wei Wei, Xian-Ling Mao, Ziyang Wang, Minghui Qiu, Feida Zhu, and Xin Cao. 2022. Multi-Level Cross-View Contrastive Learning for Knowledge-Aware Recommender System. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'22)*. 1358–1368.