Doctoral Thesis

Towards Effective and Efficient Personalized Recommendation from a Spectral Perspective

Shaowen Peng

February 2024

Department of Social Informatics
Graduate School of Informatics
Kyoto University

Doctoral Thesis
submitted to Department of Social Informatics,
Graduate School of Informatics,
Kyoto University
in partial fulfillment of the requirements for the degree of
DOCTOR of INFORMATICS

Thesis Committee: Takayuki Ito, Professor
Keishi Tajima, Professor
Hisashi Kashima, Professor
Kazunari Sugiyama, Professor (Osaka Seikei University)

Towards Effective and Efficient Personalized Recommendation from a Spectral Perspective*

Shaowen Peng

Abstract

Due to the development of computer hardware and explosive growth of data, personalized recommendation has been applied to many online services such as E-commerce, social network service, short videos and so on, which is ubiquitous in our daily life. Starting from matrix factorization to deep learning based methods, tremendous research effort has been devoted to exploit powerful algorithms to extract user preference from complex user behaviours, and they have shown great potentials and superior performance for recommender systems. However, we notice that limited attention has been paid to the spectrum of data representation containing important information of the recommendation datasets and reflecting how users and items are represented in the embedding space. In this thesis, we analyze and evaluate recommendation algorithms from a spectral perspective. Particularly, we focus on (1) the graph spectrum and (2) the spectrum of user/item representations.

While Graph Convolutional Networks (GCNs) have shown tremendous success in recommender systems and collaborative filtering (CF), the mechanism of how they contribute to recommender systems has not been well studied. Furthermore, GCN-based recommendation methods suffer from expensive computational complexity and poor scalability compared with traditional methods. By analyzing GCN from a spectral perspective (*i.e.,* the graph spectrum), we unveil the effectiveness of GCN for recommendation from the following three aspects.

- We discover that only a small fraction of spectral graph features that emphasize the neighborhood smoothness and difference contribute to the recommendation accuracy, whereas most graph information can be considered as noise that even reduces the performance. What's more, stacking multiple graph convolution

*Doctoral Thesis, Department of Social Informatics, Graduate School of Informatics, Kyoto University, KU-I-DT6960-33-0255, February 2024.

layers (*i.e.,* repeating the neighborhood aggregation) emphasizes smoothed features and filters out noise information in an ineffective way.

- 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.
- The number of required spectral graph features is closely related to the spectral distribution, where important information tends to be concentrated in more (fewer) spectral features on the dataset with a flatter (sharper) distribution.

Based on the above findings, we propose more effective and efficient GCN learning algorithms for recommender systems, which outperform state-of-the-arts and reduce the time and space complexity of existing works.

In another line of our work, we focus on the spectrum of the user/item representations to study what factors contribute to good representations. We shed light on an issue in the existing pair-wise learning paradigm (*i.e.,* the embedding collapse problem), that the representations tend to span a subspace of the whole embedding space, leading to a suboptimal solution and reducing the model capacity. Specifically, optimization on observed interactions is equivalent to a low pass filter causing users/items to have the same representations and resulting in a complete collapse; while negative sampling acts as an unreliable high pass filter to alleviate the collapse by balancing the embedding spectrum but still leads to an incomplete collapse. To tackle this issue, we propose a novel method called DirectSpec, acting as a reliable all pass filter to balance the spectrum distribution of the embeddings during training, ensuring that users/items effectively span the entire embedding space. Additionally, we provide a thorough analysis of DirectSpec from a decorrelation perspective and propose an enhanced variant, DirectSpec$^+$, which employs self-paced gradients to optimize irrelevant samples more effectively.

Keywords: Information Retrieval, Recommender System, Collaborative Filtering, Graph Neural Network, Spectrum

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Background

With the explosive growth of information on the web, we are overwhelmed by the large amount of accessible data nowadays. Recommender system (RS) is considered as an effective strategy to overcome information overload by providing personalized content (*e.g.,* products, music, restaurants, etc.) to distinct users according to their interests and preference. Therefore, it has been extensively applied to online services such as Amazon, Youtube, TikTok, and so on. It has also been reported that recommender systems can significantly boost sales by influencing user behaviours [1, 2]. For instance, 35% of what (out of 310 million) consumers purchase on Amazon and 75% of what they (out of 247 million) watch on Netflix come from product recommendations based on algorithms*. As such, there is no doubt that recommender systems play an increasingly important role to enrich our daily life by providing a variety of choices to users when they are undecided over their next behaviours.

The goal of recommender systems is to predict the relevant scores of users to a group of items based on the optimization over past user-item interactions, and the items with the highest scores are considered as relevant items that are recommended to users. Generally, recommendation datasets at least contain user-

*https://urlzs.com/ZLprh

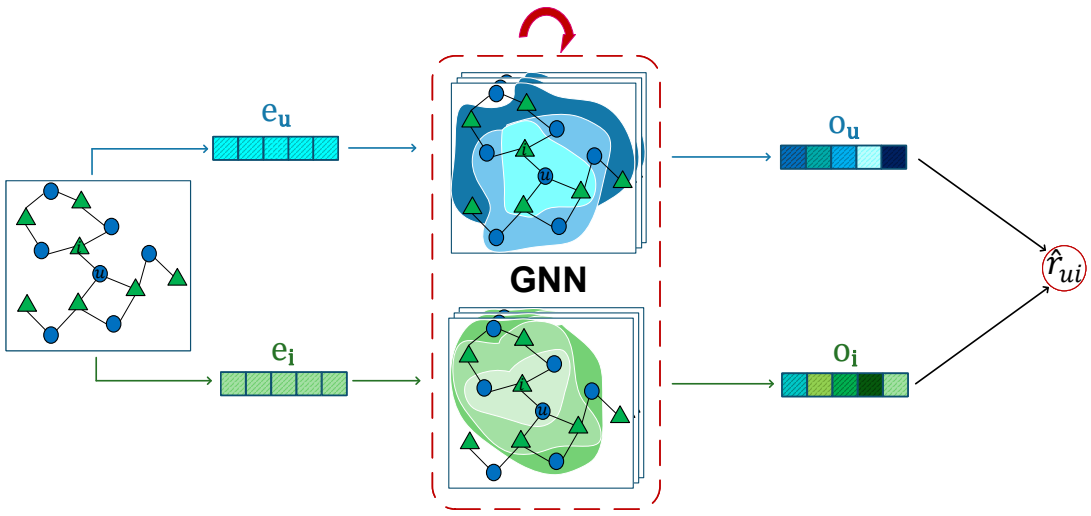


Figure 1.1. A visualization of difference between conventional (*e.g.,* MF) and GNN-based methods. For GNN-based methods, the user/item embeddings are repeatedly updated by aggregating the message from higher-order neighborhood.

ID, item-ID, and interactions (*i.e.,* the record of which users interacted with which items) represented as implicit (*e.g.,* clicks or views) or explicit feedbacks (*e.g.,* ratings or reviews). Additionally, other side information such as demographic data (*i.e.,* age, gender, country, etc.) [3], social relations [4], review data [5], item attributes (*i.e.,* image, genre, price, etc.) [6], and so on can also be included to augment data to better infer the user preference from the historical behaviours.

To accurately and precisely predict users' future behaviours, several kinds of recommendation algorithms have been proposed, such as memory-based [7, 8], model-based [9, 10, 11], content-based [12, 13], hybrid-based [14, 15] methods, and so on. Among them, matrix factorization (MF) [11] is one of the simplest yet effective methods. It characterizes user and item as latent vectors, and estimates the score between a user and an item as the inner product between their latent vectors. Despite its effectiveness, the performance of MF is limited as it simply uses a linear function to model complex user-item relationships. To overcome this issue, subsequent works empower the recommendation algorithms by replacing the linear function with other advanced algorithms such as multilayer perceptrons (MLP) [16, 17], recurrent neural networks (RNNs) [18, 19], attention mechanism [20, 21], transformer [22, 23], etc. and have shown tremendous success

in recommender systems.

Users usually only interact with a very small fraction of items out of millions of them, leading to a sparse training data in practice. Consequently, this issue jeopardizes the effectiveness of recommendation algorithms [24]. The recommender systems that are even equipped with powerful algorithms also show poor performance under extreme data sparsity. In recent years, graph neural networks (GNNs) [25] have attracted considerable attention in various research fields such as node classification [26], anomaly detection [27], molecular science [28], and so on. GNNs have also shown great potential in recommender systems and collaborative filtering (CF) [29, 30] due to the capability of alleviating the sparsity issue by capturing the higher-order collaborative signals. As illustrated in Figure 1.1, unlike traditional recommendation algorithms directly optimizing the sparse user-item interactions, GNNs represent user-item interactions as a bipartite graph and repeatedly aggregate the messages from higher-order neighborood, providing an effective way to augment the training data to alleviate the data sparsity issue.

1.2 Motivation and Challenge

Owing to the dramatic improvement of computing hardware, more powerful algorithms have been proposed and have achieved significant success in various research fields such as computer vision [31], natural language processing [32], speech recognition [33], etc. Starting from memory-based and model-based methods, deep learning based techniques requiring more training parameters and computing resources have been extensively applied to recommender systems as well [17, 34]. However, existing recommendation algorithms are mostly evaluated on certain datasets, their efficiency and effectiveness have not been carefully studied with empirical and theoretical analysis. As such, in this thesis, we comprehensively analyze recommendation algorithms from a spectral perspective. Particularly, we focus on the spectrum information with two aspects:

- Graph Neural Networks (GNNs) recently have been applied to recommender systems, which unlike traditional recommendation methods represent the recommendation data as a graph. We demystify GNN-based recommendation methods by analyzing the graph spectrum of existing methods, and propose more scalable and effective GNN learning algorithms for recommendation.

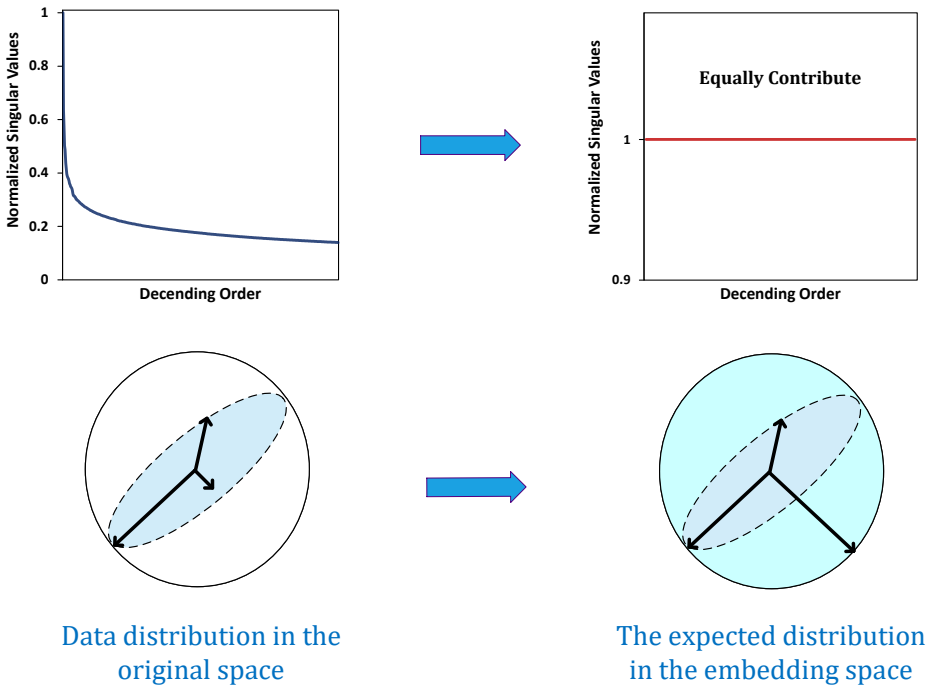


Figure 1.2. An illustration of the data distribution in the original space and expected distribution in the learning embedding space.

- Without depending on specific algorithms, we generally analyze the embedding spectrum of recommendation methods to study what factors contribute to good user/item representations. Particularly, we shed light on a collapse issue causing user and item representations to span a subspace of the whole embedding space, and we propose effective methods to address this issue.

Graph Neural Networks (GNNs) recently have attracted significant attention in recommender systems as they can learn high-quality representation under data sparsity by exploiting higher-order neighborhood. The research efforts are mostly devoted to applying GNN structures to different scenarios of recommendation [35, 36, 30, 37] or empowering GNNs with other advanced algorithms [38, 39, 40], while the following research questions have not been well studied:

- How and why GNNs show superior performance to traditional recommendation algorithms, what designs matter for GNN-based recommender systems?

- Compared with traditional recommendation algorithms, GNNs suffer from high computational cost and poor scalability. It has also been reported that GNN-based methods show slow training convergence [41]. Can GNNs be both effective and efficient for recommender systems?
- GNNs suffer from an over-smoothing issue [42], causing users/items to have the same representations as stacking more layers. As a result, most GNN-based methods remain shallow. Can GNN-based recommendation algorithms benefit more from deep GNNs?

We argue that the above limitations are due to the lack of a deep understanding of GNNs. Given that the foundation of GNNs is deeply rooted in the spectral graph theory [43] and the signal processing on graphs [44], we mainly focus on graph convolutional networks (GCNs) which exploits the adjacency matrix to aggregate messages from neighborhood. By reviewing existing GCN-based methods, we show three kinds of redundancies that significantly affect model efficiency and effectiveness:

1. **Feature redundancy.** Only a very small fraction of spectral features that emphasize the neighborhood smoothness and difference significantly affect the recommendation accuracy, while most graph information is barely contributive to recommendation that can be considered noise added on the graph. Stacking more graph convolution layers can suppress but cannot completely remove the noisy features. Based on this observation, we propose Graph Denoising Decoder (GDE) which only keeps the important graph features to model the graph smoothness and difference without stacking layers.
2. **Structure redundancy.** We show the close connection between GCN-based and low-rank methods (*e.g.,* singular value decomposition (SVD) and MF) that stacking graph convolution layers is to learn low-rank representations by emphasizing (suppressing) the components with larger (smaller) singular values. We then propose a simplified GCN learning paradigm dubbed SVD-GCN showing much fewer complexity than existing GCN-based methods.
3. **Distribution redundancy.** The number of required spectral features contributing to recommendation is highly related to the spectral distribution, where important information tends to be concentrated in more (fewer) graph

features on a flatter (sharper) distribution (*i.e.,* the spectral value drops more slowly (quickly)). To reduce the complexity for retrieving spectral features, we concentrate the important information in fewer spectral features by sharpening the spectral distribution. Specifically, we introduce a renormalized adjacency matrix with a hyperparameter adjusting the sharpness of the spectral distribution to reduce the number of required spectral features.

We then shift our focus to the embedding spectrum. Most of existing recommendation algorithms can be considered as MF variants whose goal is to learn low dimensional representations (with dimension d) from the high dimensional sparse interaction matrix (with dimension $D \gg d$). Figure 1.2 illustrates the top 500 normalized singular value distribution of the interaction matrix of CiteULike. We observe that users/items are predominantly distributed along a few dimensions in the original space while most dimensions barely contribute (*i.e.,* with singular values close to 0) to the representations. Thus, when users and items are mapped into a more compact embedding space, it is expected that redundant dimensions are all removed and each dimension contributes to the user/item representations as equally and uniformly as possible (*i.e.,* the representations make full use of the embedding space). Unfortunately, by analyzing the spectrum of the embedding matrix, we empirically and theoretically show that users/items tend to span a subspace of the whole embedding space (with dimension $d' < d$), where the embeddings collapse along all (complete collapse) or certain dimensions (incomplete collapse). Particularly, optimization solely on observed interactions is equivalent to a low pass filter, where the representations of users and items tend to collapse to a constant vector. Negative sampling is the most common technique to optimize recommendation algorithms without causing an explicit embedding collapse by pushing away the unobserved user-item pairs, and we show that it is equivalent to a high pass filter that alleviates the collapse issue by balancing the embedding spectrum. However, there is no guarantee that negative sampling can completely prevent the collapse, and collapse over certain dimensions still happens on existing pair-wise learning paradigms such as Bayesian personalized ranking (BPR) [45] and binary cross-entropy (BCE) loss [17]. Due to the data sparsity and long tailed distributions, increasing negative sampling ratios is considered as an effective way to to improve representation quality [17, 46], whereas we also demonstrate that it cannot further alleviate the collapse issue by evaluating different negative sampling ratios. We tackle the embedding collapse issue from a

spectral perspective. We observe that the extent of the collapse is closely related to the spectrum distribution of the embedding matrix. Specifically, only one singular value dominates when the representations completely collapse, whereas the singular values are uniformly distributed when the representations make full use of the embedding space. Inspired by this observation, we propose a novel method dubbed DirectSpec acting as an all pass filter to ensuring that all dimensions equally contribute to the representations. We theoretically and empirically show that DirectSpec can completely prevent the embedding collapse without explicitly sampling negative pairs by directly balancing the spectrum distribution, and provide a simple implementation with a complexity only as $\mathcal{O}(B^2d)$ where B is the batch size. Moreover, we shed light on DirectSpec from a decorrelation perspective, and propose an enhanced variant DirectSpec$^+$ which employs self-paced gradients to optimize the irrelevant samples that are highly correlated more effectively. By showing the close connection between DirectSpec and uniformity, we discover that contrastive learning (CL) can alleviate embedding collapse by balancing spectrum distribution in a similar way to DirectSpec, explaining the effectiveness of CL based recommendation algorithms.

1.3 Contributions

The contributions of this thesis can be summarized follows:

- We shed light on the feature redundancy on graph based recommendation that only a very small fraction of spectral features emphasizing neighborhood smoothness and difference are contributive to recommendation, and unveil the effectiveness and weakness of existing GCN-based methods. We propose graph denoising encoder (GDE) which can capture the message from any-hop neighborhood without stacking graph convolution layers.

- We show the close connection between GCN-based and low-rank methods that GCNs contribute to recommendation in a way similar low-rank methods. Particularly, stacking graph convolution layers is to learn a low-rank representation by emphasizing (suppressing) the components with larger (smaller) singular values, and the weighted spectral features is the key making GCN effective instead of the neighborhood aggregation that is considered as the core design of GCNs. We propose a simplified GCN learning paradigm dubbed SVD-GCN which 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.

- We show that the number of required spectral feature is closely related to the spectral distribution, that the datasets with flatter (sharper) spectral distribution tend to require more spectral features. To reduce the computational cost for retrieving spectral features, we concentrate important information from interactions on fewer features by increasing node smoothness to sharpen the spectral distribution, resulting in significant improvement as well.

- We propose a scalable contrastive learning framework by performing augmentation on spectral features where the intensity of the noise added on the features is inversely proportional to their importance and augmenting sparse supervisory signals with abundant higher-order neighborhood signals, resulting in significant improvement.

- We theoretically and empirically show that existing recommendation methods suffer from embedding collapse, that the representations tend to fall into a subspace of the whole embedding space, and analyze the mechanisms causing this issue. We propose a novel method DirectSpec which directly balances the spectrum distribution. We empirically and theoretically show that DirectSpec can prevent embedding collapse.

- Extensive results on public datasets not only show our proposed methods outperform state-of-the-arts but also demonstrate the efficiency and effectiveness of our proposed designs.

1.4 Thesis Outline

The rest of this thesis is organized as follows. In Chapter 2, we conduct comprehensive literature review related to our research. In Chapter 3 to 5, we present our solutions to tackle the aforementioned research questions in GNN, including demystifying how GNNs contribute to recommendation, more effective and efficient GNN methods, and approaches to tackle the over-smoothing issue. In Chapter 6, we propose DirectSpec to tackle the embedding collapse issue in recommendation algorithms. Finally, we conclude our research in Chapter 7.

CHAPTER 2

Literature Review

2.1 Collaborative Filtering

This thesis focuses on the personalized recommendation under the setting of collaborative filtering (CF). CF, a fundamental task for recommender systems, makes predictions based on users' historical interactions. Early memory-based CF methods exploit users that share similar interests or items that tend to be interacted by similar users to infer user preference [8, 47]. The similarity are usually measured as the cosine similarity or Pearson correlation between the users or items interaction vectors. However, directly calculating the similarity is far from effective due to the sparseness and the large size of the datasets. Model-based CF methods [48, 49] are becoming increasingly prevalent as they can tackle the shortcomings of memory-based methods by representing users and items in a more effective way. Especially, Model-based methods usually characterize users/items as low dimensional vectors instead of directly employing the high dimensional interaction matrix, making the algorithms more efficient and scalable. Matrix factorization (MF) [11] is one of the most extensively used model-based methods and the cornerstone of advanced recommendation algorithms. It stems from singular value decomposition (SVD) and compresses the high dimensional interaction matrix to a low dimensional embedding matrix, where users/items are represented as latent vectors and the rating is estimated as the inner product or

cosine similarity between the user and item latent vectors. However, the model expressivity of MF is still limited, due to (1) the lack of auxiliary information provided to complement user-item interactions to help better infer user preference, and (2) the unreasonable assumption that complex user-item relations can be properly modelled by a simple linear function. To tackle the above weakness, subsequent works mostly focus on introducing auxiliary information [50, 51] or employing advanced algorithms to empower MF-based methods [17, 22]. There are still some challenges existing in CF and recommender systems in general that need to be addressed such as: (1) cold start [52]. How to recommend new users and items? (2) Bias in recommendation, such as popularity bias, exposure bias, and so on [53]. (3) Privacy protection. Collecting more detailed user information always leads to more accurate recommendation while is also a threat to users [54].

2.2 Deep Learning Based Recommendation

Deep learning has yielded immense success in recommender systems in recent years. Compared to traditional recommendation models:

- Deep learning is adept at modelling the non-linearity such as user-item relations. Conventional methods such as matrix factorization [11] and factorization machine [55] are essentially linear models. By stacking weight transformation and non-linear activation function, neural networks are capable of approximating any measurable functions to any desired degree of accuracy [56].
- Traditional methods rely heavily on the hand-crafted feature designs. Deep learning, on the other hand, can automatically learn the complex relations between raw input data and the decision space without relying much on the feature engineering, providing a way to effectively learn from side information such as review, temporal and spatial data, social relation, etc.

According to the techniques that have been applied to recommender systems, deep learning based recommendation models can be categorized to: multilayer perceptron (MLP), autoencoder (AE), convolutional neural network (CNN), recurrent neural network (RNN), attention mechanism, transformer, and so on. For instance, wide & deep learning framework jointly trains feed-forward neural networks with embeddings and linear model with feature transformations for generic

recommender systems with sparse inputs [16]. Liang et al. [57] developed variational autoencoder (VAE) for collaborative filtering on implicit feedback data, enabling us to go beyond linear factor models with limited modelling capacity. NCF [17] is a simple and generic neural network architecture fusing generalized matrix factorization (GMF) and MLP to model latent features of users and items. Kand et al. [58] built a self-attention based sequential recommendation model (SASRec), which adaptively assigns weights to previous items at each time step. Zheng et al. [59] proposed a reinforcement learning framework applying a deep Q-Learning structure that can take care of both immediate and future reward for online personalized news recommendation.

From a different perspective, according to the available data that can be used, different task-specific methods has been proposed. For instance, for pure collaborative filtering only exploiting the user-item interactions, most deep learning based models [17, 60] replace the simple model architecture with advanced deep learning based techniques to model the complex user-item relations. Due to the availability of temporal information, self-attention and RNN that are suited to handle sequential data are applied to sequential and session-based recommendation [19, 58, 61]. Deep neural networks and attention mechanisms are employed to multimedia recommendation owing to their superior ability to automatically learn from heterogeneous data [20, 62, 63]. Aside from aforementioned mentioned architectures, convolutional neural networks (CNN) and transformers have shown potentials in explainable recommendation [64, 65]. However, concern has also been raised that the progress achieved by deep learning is not as strong as claimed in the published research [66].

2.3 Graph Neural Network (GNN) and its Applications

2.3.1 Spectral and Spatial GNN

The non-Euclidean nature of graphs prevent deep learning techniques from being directly applied to the graph data. Bruna et al. [67] generalized CNNs to the graph data and proposed a spectral graph convolutional network (GCN) based

on the graph Fourier transform:

$$F = \mathbf{V}^T x, \tag{2.1}$$

where $\mathbf{V}$ is the stacked eigenvectors of the graph Laplacian, x is a graph signal. Note that the eigenvectors $\{\mathbf{v_1}, \cdots\}$ is an orthonormal basis, thus $\mathbf{V}^T x$ is actually the coordinate of the basis $\{\mathbf{v_1}, \cdots\}$. Given the output signal from the graph Fourier transform $\hat{x}$, the inverse graph Fourier transform is defined as follows:

$$F^{-1} = \mathbf{V}\hat{x}. \tag{2.2}$$

Then, the spectral graph convolution is defined as follows:

$$g_\theta * x = \mathbf{V} g_\theta \mathbf{V}^T x, \tag{2.3}$$

where $g_\theta = diag(\theta)$ is a parameterized filter where $diag(\cdot)$ is the diagonalization operation. Due to the spatial non-localization and the expensive learning complexity for retrieving eigenvectors, Defferrard et al. [68] simplified the spectral graph convolution as a polynomial graph filter. By considering g_θ as a function of the eigenvalues λ_i of the graph Laplacian $\mathbf{L}$, the spectral graph convolution can be rewritten as follows:

$$g_\theta * x = \mathbf{V} g_\theta \mathbf{V}^T x = \mathbf{V} diag\left(\sum_{k=0}^{K} \theta_k \lambda_i^k\right) \mathbf{V}^T x = \sum_{k=0}^{K} \theta_k \mathbf{L}^k. \tag{2.4}$$

It further reduced the complexity with Chebyshev polynomial, significantly reducing the learning complexity of the spectral graph convolution. Based on these two research works, Kipf et al. [25] proposed a layer-wise GCN model by further simplifying [68]. A single-layer model architecture is formulated as follows:

$$g_\theta * x = \sigma\left(\hat{\mathbf{A}}\mathbf{X}\mathbf{\Theta}\right), \tag{2.5}$$

where $\sigma(\cdot)$ is an activation function, $\mathbf{X}$ is the feature matrix, $\mathbf{\Theta}$ is the weight matrix, $\hat{\mathbf{A}}$ is a symmetrically normalized adjacency matrix. By stacking multiple layers, Equation (2.5) is able to aggregate the messages from multi-hop neighborhood. [25] is one of the most extensively used GNN architectures.

There are basically two types of GNNs: spatial- and spectral-based GNNs, the difference of them lies in the the message passing design. Spatial-based GNNs focus on the spatial properties of graphs, one representative work is graph attention networks [69]. Unlike [25], it uses attention mechanism to measure the

importance between the target node and its first neighborhood. GraphSAGE [70] is a general inductive framework learning a function that generates embeddings by sampling and aggregating features from nodes' local neighborhood. By theoretically showing that existing GNNs fail to distinguish similar graph structures, Ke et al. [71] proposed Graph Isomorphism Network (GIN) that is as powerful as the Weisfeiler Lehman (WL) graph isomorphism test. Spectral-based GNNs which are also called GCNs, on the other hand, focus on the spectral properties of graphs. They have a solid mathematical foundation in graph signal processing [44] and mostly can only deal with the undirected graphs since the graph Laplacian of directed graphs are asymmetric with complex eigenvalues. The aforementioned works [25, 67, 68] generalizing CNNs to the graph data are representative spectral-based GNNs. It has been shown that GCNs are low pass filters emphasizing the low frequency components [72, 73], while some works show that high frequencies are also important [74, 75] especially on heterophilic graphs. Balcilar et al. [76] bridged the gap between spectral- and spatial-based GNNs by conducting spectral analysis. Unlike conventional GCNs based on polynomial filters, some works improve GCNs by exploiting other expressive filters such as Bernstein approximation [77], ARIMA filter [78], Jacobi basis [79], etc., showing better performance. Despite the superior performance GNNs have shown in various research fields, there are still some challenges and issues that need to be tackled such as: (1) Over-smoothing [42]. The node representations tend to be the same as stacking more layers which instead reduces the performance. As a result, conventional GNNs cannot benefit from deep model architectures. (2) Expressive power [80]. many GNN methods fail to distinguish similar graph structures that cannot be more powerful than WL test. (3) Effectiveness and scalability trade-off. GNNs suffer from expensive learning complexity. The sampling based strategy [70, 81] can reduce the complexity while loses part of the graph information as well. How to trade-off scalability and graph integrity is critical. (4) Generalization on heterophily. Conventional GNNs are equivalent to low pass filters working well on homophilic graphs while perform poorly on heterophilic graphs where connected nodes tend to be different.

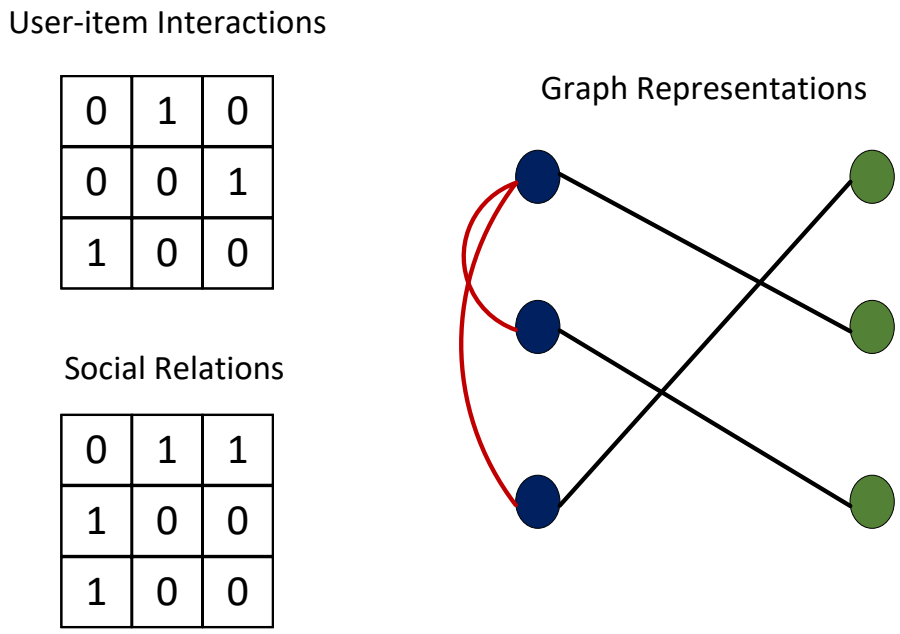


Figure 2.1. An illustration of how the interactions are represented as a graph.

2.3.2 GNN-based Recommendation

Graph neural networks (GNNs) have also been applied to recommender systems in the light of their success in other research fields, including collaborative filtering (CF) [30, 82], social recommendation [35], sequential recommendation [37, 83], multimedia recommendation [84], news recommendation [85, 86], and so on. Since we focus on CF in this thesis, we mainly summarize the GNN learning paradigm for CF in this subsection.

Graph Construction. Given the user-item interaction matrix, it can be represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the node set $\mathcal{V}$ contains all users and items, and the edge set $\mathcal{E}$ is represented by the observed interactions. Without introducing auxiliary information, $\mathcal{G}$ is bipartite where the edges only exist between a user and an item. User-user or item-item can be connected when introducing social relations or item-item knowledges [87, 88]. Figure 2.1 is an example of how interactions are represented as graphs. The graph can be more complex by including external knowledge [89, 90]. For instance, for movie recommendation we have other entities such as actors and directors where there are additional connections

among movies, directors, and actors. Some research works use hypergraphs which are more powerful in representation and provide more information to replace the simple graph [39, 91]. Unlike the simple graph that a edge only contains two nodes, the hyperedge connects any number of nodes in the hypergraphs [92].

Model Architecture. Overall, the message passing design in GNN-based methods can be summarized as follows:

$$\begin{aligned} \mathbf{n}_u^{(l+1)} &= \textbf{Aggregator}\left(\mathbf{h}_i^{(l)} | i \in \mathcal{N}_u\right), \\ \mathbf{h}_u^{(l+1)} &= \textbf{Update}\left(\mathbf{n}_u^{(l+1)}, \mathbf{h}_u^{(l)}\right), \end{aligned} \tag{2.6}$$

where $\mathcal{N}_u$ is the directly connected neighbor to u, $\textbf{Aggregator}(\cdot)$ is a function to aggregate messages from neighborhood i, and $\textbf{Update}(\cdot)$ is to update node embeddings based on their previous $\mathbf{h}_u^{(l)}$ and current embeddings $\mathbf{n}_u^{(l+1)}$. $\textbf{Aggregator}(\cdot)$ is required for most message-passing based GNNs. Usually the sum or mean function is favoured as it is easy to implement without introducing additional complexity [29, 30, 35], some works also use the max pooling [93] or attention mechanism [90] as the aggregator. SpectralCF [82] applies the original spectral graph convolution [67] for recommendation which captures the global neighborhood information. To make GNNs scalable on large graphs, some works sample part of the neighborhood [93, 94] to avoid updating the embeddings via an adjacency matrix, thereby reducing the complexity. Most GNN-based methods especially for CF implement $\textbf{Aggregator}(\cdot)$ via an adjacency matrix, where the updating rule with the matrix and node form are formulated as follows:

$$\begin{aligned} \mathbf{h}_u^{(l+1)} &= \sigma\left(\sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{d_u d_i}} \mathbf{h}_i^{(l)} \mathbf{W}^{(l+1)}\right), \\ \mathbf{H}^{(l+1)} &= \sigma\left(\hat{\mathbf{A}} \mathbf{H}^{(l)} \mathbf{W}^{(l+1)}\right), \end{aligned} \tag{2.7}$$

where $\mathbf{W}^{(l+1)}$ is a weight matrix, $\hat{\mathbf{A}}$ is a symmetric normalized adjacency matrix with $\hat{\mathbf{A}}_{ui} = \frac{1}{\sqrt{d_u d_i}}$ for $(u, i) \in \mathcal{E}$, d_u and d_i are the node degrees for u and i, respectively. Equation (2.7) only applies to implicit feedbacks with only one edge type. For explicit feedbacks where the edges have different weights, the messages with different edge types are accumulated with unshared weights [29, 94]. Chen et al. [95] showed that removing the non-linear activation function enhances the recommendation performance, He et al. [41] further empirically demonstrated the redundancy of both activation functions and weight transformations. It it worth

noting that some works review GNNs from a graph signal processing perspective and do not need to explicitly aggregate neighborhood [80, 96]. The **Update**$(\cdot)$ function is not necessary for GNN-based methods. NGCF [30] uses element-wise multiplication and sum, while Pinsage [93] introduces an attention mechanism to update the embeddings.

The final user and item representations are generated via a **Pooling**$(\cdot)$ function:

$$\mathbf{O} = \mathbf{Pooling}\left(\mathbf{H}^{(0)}, \cdots, \mathbf{H}^{(L)}\right). \tag{2.8}$$

Some works such as GCMC and Pinsage [29, 93] simply use the embeddings from the final layer as the final representation, while most works generate the representations by accumulating embeddings from different layers. Particularly, the concatenate [30, 82, 94] and sum [40, 41] are favoured as they are easy to implement without introducing additional complexity.

Optimization. The works focusing on implicit feedbacks usually choose a Bayesian personalized loss (BPR) [45] formulated as follows:

$$\mathcal{L}_{bpr} = -\sum_{u\in\mathcal{U}} \sum_{(u,i^+)\in,(u,i^-)\notin\mathcal{E}} \ln \sigma\left(\hat{r}_{ui^+} - \hat{r}_{ui^-}\right). \tag{2.9}$$

Here, $\sigma(\cdot)$ is the sigmoid function, $\hat{r}_{ui^+}$ and $\hat{r}_{ui^-}$ are predicted scores usually measured by the inner product between the representation of a user and an item. Since the goal of recommendation with explicit feedbacks is to minimize the difference between the estimated rating and ground truth, the Euclidean distance is favoured:

$$\mathcal{L}_{dist} = \sum_{u\in\mathcal{U}} \sum_{(u,i)\in\mathcal{E}} \left(\hat{r}_{ui} - r_{ui}\right)^2. \tag{2.10}$$

The above supervised learning losses can be combined with other loss functions. For instance, growing effort has been devoted to apply self-supervised learning (SSL) to graph learning [97, 98] including recommender systems. Wu et al. [38] proposed to jointly optimize the supervised learning loss and the following self-supervised learning loss:

$$\begin{aligned}
\mathcal{L}_{user} &= -\sum_{u\in\mathcal{U}} \ln \frac{\exp\left(\mathbf{o'}_u^T \mathbf{o''}_u/\tau\right)}{\sum_{v\in\mathcal{U}} \exp\left(\mathbf{o'}_u^T \mathbf{o''}_v/\tau\right)}, \\
\mathcal{L}_{item} &= -\sum_{i\in\mathcal{I}} \ln \frac{\exp\left(\mathbf{o'}_i^T \mathbf{o''}_i/\tau\right)}{\sum_{v\in\mathcal{I}} \exp\left(\mathbf{o'}_i^T \mathbf{o''}_j/\tau\right)},
\end{aligned} \tag{2.11}$$

where τ is a temperature hyperparameter, $\mathbf{o}'_u$ and $\mathbf{o}''_u$ represent two different views of u. The views are generated by perturbing the graph $\mathcal{G}$ called graph augmentation. Commonly used augmentation operators include node dropping, edge dropping, and so on [38, 99].

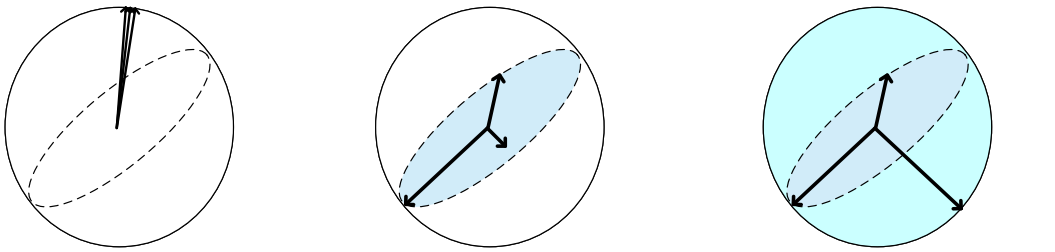
Recent research works also empower GNNs for recommendation by combining them with other advanced algorithms or techniques such as negative sampling [100], learning in hyperbolic space [40], and knowledge distillation [101], etc. Still, there are some challenges in GNN-based recommendation methods that need to be addressed:

- Some issues in GNNs that also affect the recommendation performance. For instance, the over-smoothing issue that makes the representations of distinct nodes be the same as stacking more layers also prevents recommender systems from benefiting from deep GNNs. There are some works proposed to tackle this issue. For instance, LightGCN [41] is shown to share similarities with APPNP [102] which is proven to avoid over-smoothing. Shen et al. [80] and Peng et al. [96] analyzed GNN-based methods from a spectral perspective to address the over-smoothing issue.

- Some issues in recommender systems that are aggravated by GNNs. For instance, Zhao et al. [103] showed that the symmetric neighborhood aggregation adopted in most GCN-based CF methods exacerbates the popularity bias issue. Existing GNN learning paradigms are mostly inherently transductive, causing the cold-start problem to be more difficult to address.

- The expensive complexity and poor scalability compared with traditional recommendation methods. In spite of some improvement has been made to simplify GNN-based recommendation methods [41, 95], it still takes significantly large space and much time to update the node embeddings by multiplying by the adjacency matrix, making the algorithms impractical on large datasets.

- Limited attention has been paid to demystify the mechanism of how GNN works for recommendation. Most existing works simply apply GNN architecture to recommender systems. For instance, SpectralCF incorporates the spectral graph convolution [67], NGCF [30] is highly similar to GCN [25], and LightGCN is inspired by SGC [41] and APPNP [102]. Deeper insights into GNN-based methods are relatively unexplored.



(a) Complete Collapse (b) Dimensional Collapse (c) Expected Representations

Figure 2.2. An illustration of collapse in representation learning. The embedding vectors collapse to the same point for complete collapse, and span a lower dimensional space for dimensional collapse, as opposed to the expected representations making full use of the whole embedding space.

In this thesis, we focus on demystifying how GNNs contribute to recommender systems (CF mainly) by analyzing existing works from a spectral perspective. By providing deep insights into GNN-based methods, we further investigate what designs are necessary for recommendation and what designs are redundant. Based on the comprehensive analysis, we propose simple yet effective GNN learning paradigm for recommendation, significantly reducing the complexity of GNNs while maintaining superior recommendation accuracy.

2.4 Collapse in Representation Learning

The success of machine learning algorithms significantly lies in learning representation, the goal of which is to represent data in an effective and efficient way to make it easier to extract useful information when building predictors [104, 105, 106]. However, it has also been shown that model outputs tend to collapse to the same constant vector (as shown in Figure 2.2 (a)) when only optimizing the model based on the positive pairs [107]. This issue can be well alleviated by self-supervised learning and contrastive learning by exploiting negative samples in an effective way [108, 109]. Due to the heavy computation, research effort has been made to simplify the self-supervised learning algorithms without explicitly sampling negative data [107, 110, 111]. Unfortunately, subsequent works

show that a dimensional collapse (as shown in Figure 2.2 (b)) cannot be ignored that the embedding vectors end up spanning a lower dimensional subspace of the whole embedding space [109, 112]. The collapse issue in representation learning shares similarities to some issues such as over-smoothing in GNN [42], which inspires some researcher works to tackle issues in GNN [113, 114]. Inspired by the aforementioned works tackling collapse in representation learning, we review existing recommendation methods, and show that they suffer from a collapse issue as well. Particularly, the representations tend to collapse to a constant vector when only optimization observed interactions, and an incomplete collapse still exists despite introducing negative samples [17, 45] or even raising the negative sampling ratio [46] where the representations are predominantly distributed in certain dimensions. Most existing works tackle the collapse issue from a spatial perspective by pushing away different users/items [38, 115, 116], whereas we address this issue from a spectral perspective by directly balancing the embedding spectrum, which is demonstrated more effective than existing works.

CHAPTER 3

Graph Feature Denoising for Recommendation

In this chapter [96], we address the research questions of graph convolutional network (GCN) based recommendation methods mentioned in Section 1.2. To unveil the effectiveness of GCNs for recommendation, we first analyze them in a spectral perspective and discover two important findings: (1) only a small fraction of spectral graph features that emphasize the neighborhood smoothness and difference contribute to the recommendation accuracy, whereas most graph information can be considered as noise that even reduces the performance, and (2) repetition of the neighborhood aggregation emphasizes smoothed features and filters out noise information in an ineffective way. Based on the two findings above, we propose a new GCN learning scheme for recommendation by replacing neighborhood aggregation with a simple yet effective Graph Denoising Encoder (GDE), which acts as a band pass filter to capture important graph features. We show that our proposed method alleviates the over-smoothing and is comparable to an indefinite-layer GCN that can take any-hop neighborhood into consideration. Finally, we dynamically adjust the gradients over the negative samples to expedite model training without introducing additional complexity. Extensive experiments on five real-world datasets show that our proposed method not only outperforms state-of-the-arts but also achieves 12x speedup over LightGCN.

3.1 Introduction

Recommender systems have been playing an important role in people's daily life by predicting items the user may be interested in based on the analysis of users' historical records, such as user-item interactions, user reviews, demographic data, etc. Collaborative Filtering (CF), which focuses on digging out the user preference from past user-item interactions, is a fundamental task for recommendation. A common paradigm for CF is to characterize users and items as learnable vectors in a latent space and optimize based on user-item interactions. Matrix factorization (MF) [11] is one of the most widely used embedding-based methods, which estimates the interaction as the inner product between user and item latent vectors. Subsequent works improve MF mostly by: (1) exploiting advanced algorithms such as perceptrons [17, 117], recurrent neural networks [18, 61], memory networks [118], attention mechanisms [20], transformer [22] to model non-linear user-item relations; (2) augment interactions with axillary information

To overcome the drawback of MF that simply exploits a linear function to model complex user behavior, subsequent works exploit advanced algorithms such as perceptrons [17, 117], recurrent neural networks [18, 61], memory networks [118], attention mechanisms [20], transformer [22] to model non-linear user-item relations.

However, the unavailability of capturing the higher-order signals limits the performance of the aforementioned methods due to the data sparsity. Graph convolutional networks (GCNs) [68, 25] have attracted much attention and have shown great potential in various fields including social network analysis [35, 119] and recommender systems [29, 30]. The core idea of GCNs is to augment node representations with (higher-order) neighborhood. Much effort has been devoted to adapt GCNs [25] to CF. For instance, NGCF [30] is inspired and inherits the components from vanilla GCN [25]; EGLN [120] learns an adaptive user-item graph structure to predict potential positive preference. Some research efforts have been made to simplify GCN-based CF methods by removing the non-linearity and embedding transformation [41, 95]. However, we notice that the mechanism of how GCNs contribute to recommendation has not been well studied. To this end, our work focuses on the core component (i.e, neighborhood aggregation) and aims to investigate the following research questions:

- What graph information matters for recommendation?

- How neighborhood aggregation helps recommendation and why repeating it achieves better accuracy?
- Is there a more effective and efficient design to replace neighborhood aggregation?

To answer the aforementioned questions, we review GCNs from a spectral perspective in Section 3.3.1. Specifically, we decompose adjacency matrix into spectral features and discover two important findings: (1) we identify neighborhood smoothness and difference which significantly affect the recommendation accuracy while only account for a small portion of the spectral features, whereas most graph information has no positive effect that can be considered noise added on the graph; (2) stacking layers in GCNs tends to emphasize graph smoothness and depress other information. Based on the two findings above, we unveil the ineffectiveness of neighborhood-aggregation and replace it with our proposed Graph Denoising Decoder (GDE) in Section 3.3.2, which only keeps the important graph features to model the graph smoothness and difference without stacking layers. Our proposed GDE significantly simplifies existing GCN-based CF methods and reduces the running time. The contributions of our work are summarized as follows:

- We unveil the effectiveness of Graph Convolutional Networks (GCNs) for recommendation in a spectral perspective, and shed light on the ineffectiveness of the existing design (*i.e.,* neighborhood aggregation), which provides theoretical and empirical support for our proposed method.
- Compared with existing work that stacks many layers to capture higher-order neighborhood, our proposed GDE is built in a different way by directly capturing the important spectral graph features to emphasize the neighborhood smoothness and difference, which is equipped with a simple yet effective architecture that can incorporate neighborhood signals from any hops.
- We propose to dynamically adjust the magnitude of gradients over the negative samples to tackle the slow convergence issue on GCN-based CF methods, resulting in further improvement and helping expedite the model training.
- Extensive experiments on five real-world datasets not only show our proposed method outperforms state-of-the-arts under extreme data sparsity with less running time but also demonstrate the effectiveness of our proposed designs.

3.2 Preliminaries

3.2.1 Graph Convolutional Network for CF

We first summarize the common GCN paradigm for CF. Given an interaction matrix $\mathbf{R} = \{0,1\}^{M \times N}$ with M users and N items consisting of observed $\mathbf{R}^+$ and unobserved interactions $\mathbf{R}^-$, we define a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the node set $\mathcal{V}$ contains all users and items, $\mathcal{E} = \mathbf{R}^+$ are the edge set. Each user and item is considered as a node of $\mathcal{G}$ and is characterized as a learnable embedding vector $\mathbf{e}_u \in \mathbb{R}^d$ ($\mathbf{e}_i \in \mathbb{R}^d$); by stacking them together we have an embedding matrix $\mathbf{E} \in \mathbb{R}^{(M+N) \times d}$. The goal is to estimate the unobserved interactions $\mathbf{R}^-$, by learning an interaction function given as follows:

$$f(\mathbf{R}^- | \mathcal{G}, \mathbf{R}^+, \mathbf{\Theta}) : \mathcal{V}_u \times \mathcal{V}_i \to \mathbb{R}^+, \tag{3.1}$$

where $\mathbf{\Theta}$ denotes model parameters. Here, $f(\cdot)$ corresponds to a specific GCN model. The matrix and node form of the updating rule are generally formulated as follows:

$$\begin{aligned} \mathbf{H}^{(k+1)} &= \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(k)}\mathbf{W}^{(k+1)}\right), \\ \mathbf{h}_u^{(k+1)} &= \sigma\left(\sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{d_u + 1}\sqrt{d_i + 1}} \mathbf{h}_i^{(k)} \mathbf{W}^{(k+1)}\right), \end{aligned} \tag{3.2}$$

where $\sigma(\cdot)$ is a non-linear activation function; $\mathbf{W}^{(k+1)}$ are the weight matrix at $(k+1)$-th layer, d_u and d_i are the node degree for u and i, respectively; $\mathcal{N}_u$ are nodes directly connected to u, $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, and $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, where $\mathbf{A}$, $\mathbf{D}$ and $\mathbf{I}$ are the adjacency matrix, diagonal degree matrix and identity matrix, respectively. The node embedding is updated by aggregating the neighborhood's current embedding through adjacency matrix starting from the initial state $\mathbf{h}_u^{(0)} = \mathbf{e}_u$. Some works simplify the updating rule by removing the activation function or weight matrix and gain further improvements [95, 41]. The final node embeddings are generated from the previous embeddings via a pooling function:

$$\mathbf{o}_u = \mathbf{pooling}\left(\mathbf{h}_u^{(0)}, \cdots, \mathbf{h}_u^{(K)}\right). \tag{3.3}$$

Common pooling functions are sum, concatenate; some works also output the last layer as the final embeddings. Finally, an interaction between a user and an

item is estimated as follows:

$$\hat{\mathbf{r}}_{ui} = \mathbf{o}_u^T \mathbf{o}_i. \tag{3.4}$$

3.2.2 Graph Signal Processing

We introduce an important definition from graph signal processing. Given a graph signal $\mathbf{s}$, its variation on the graph is defined as:

$$\|\mathbf{s} - \hat{\mathbf{A}}\mathbf{s}\|. \tag{3.5}$$

The variation of a signal on a graph measures the difference between the signal at each node and at its neighborhood [44]. Intuitively, a signal with small variation implies the smoothness between each node and its neighborhood, whereas a signal with large variation stresses the difference between them.

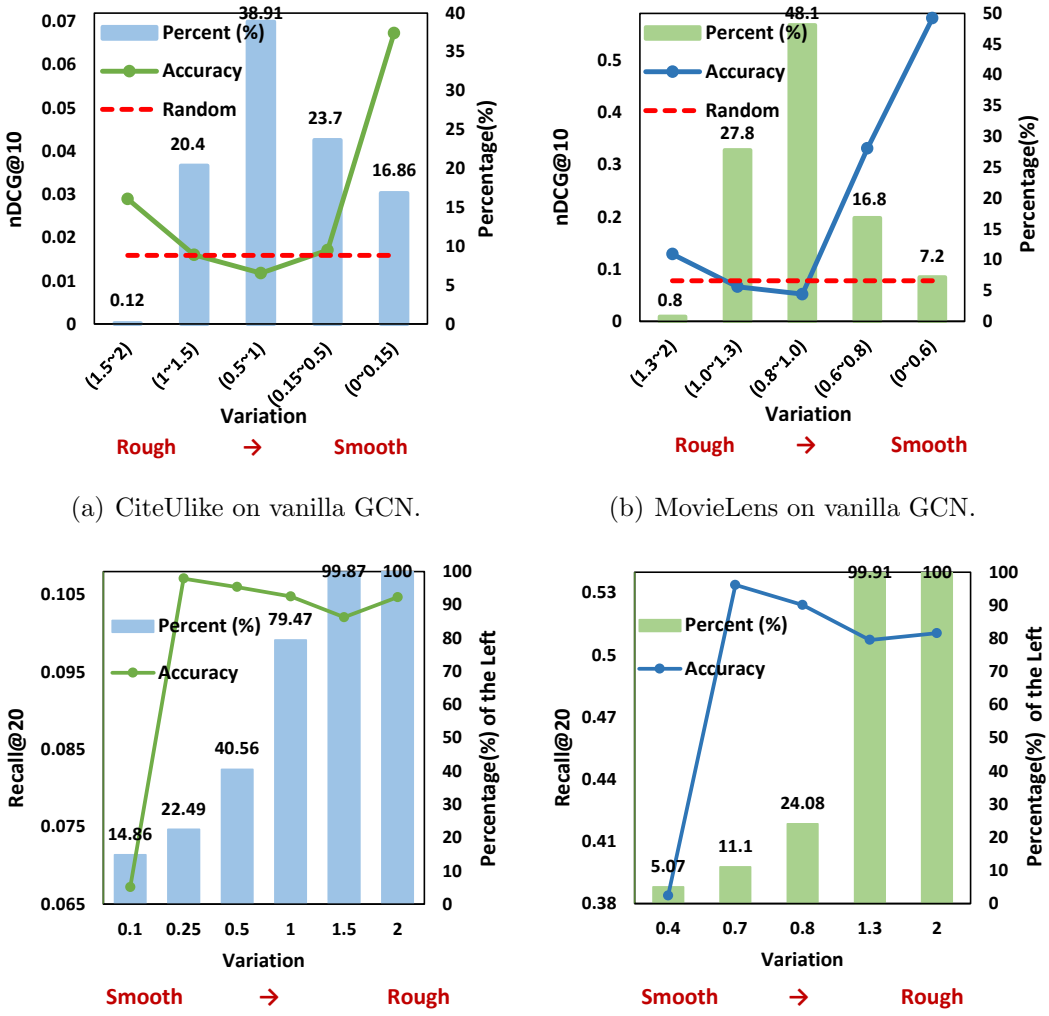
Definition 1. *Given eigenvalue λ_t and eigenvector $\mathbf{v}_t$ of $\hat{\mathbf{A}}$, the variation of an eigenvector on the graph is $\|\mathbf{v}_t - \hat{\mathbf{A}}\mathbf{v}_t\| = 1 - \lambda_t$.*

Where $\lambda_t \in (-1, 1]$ [42]. According to eigendecomposition $\hat{\mathbf{A}} = \mathbf{V} diag(\lambda_t) \mathbf{V}^T = \sum_t \lambda_t \mathbf{v}_t \mathbf{v}_t^T$, the graph information is made up of orthogonal spectral features. Through Definition 1, we can classify these features based on their variations: $\mathbf{v}_t$ with larger eigenvalue is smoother (smaller variation), while the feature with smaller eigenvalue is rougher (larger variation). In this work, we focus on investigating how spectral features with different variations affect recommendation accuracy.

3.3 Methodology

3.3.1 Recap in A Spectral Perspective

As recent works [41, 95] show that GCNs perform better without non-linear activation functions and transformation for CF, the effectiveness of GCNs lies in neighborhood aggregation, which is implemented as the multiplication of adjacency matrix. Following Definition 1, we study how neighborhood aggregation affects and contributes to recommendation accuracy.



(a) CiteUlike on vanilla GCN.

(b) MovieLens on vanilla GCN.

(c) CiteULike on LightGCN.

(d) MovieLens on LightGCN.

Figure 3.1. In (a) and (b), we partition spectral features into different groups based on their variations, it illustrates the percentage of features in different groups and how they contribute to the accuracy which is tested on vanilla GCN; we use a randomly initialized adjacency matrix for 'random' as the benchmark. (c) and (d) show the accuracy where the features with variation $\geq x$ are removed (*e.g.*, the result on $x = 2$ is obtained on the original graph containing all features) on LightGCN.

The Importance of Different Spectral Graph Features

We first define a cropped adjacency matrix as:

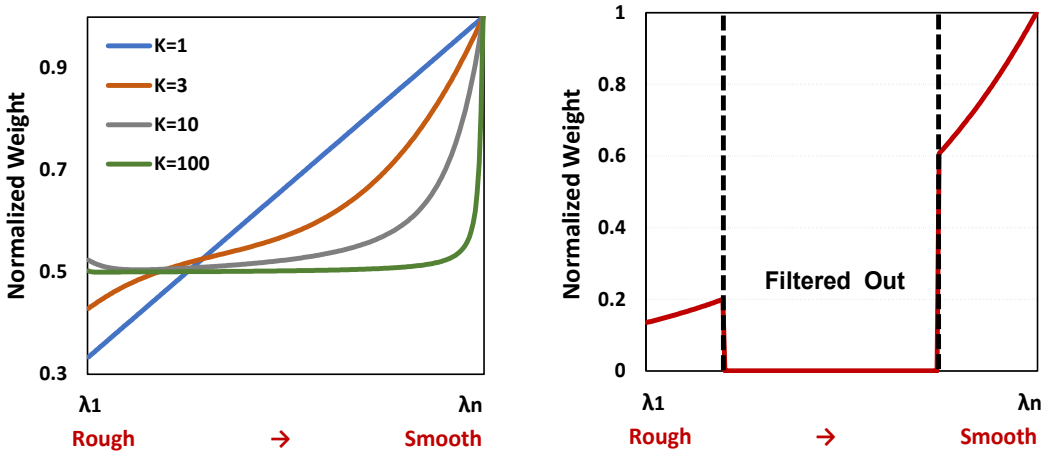
$$\hat{\mathbf{A}}' = \sum_t \mathcal{M}(\lambda_t)\mathbf{v}_t\mathbf{v}_t^T, \tag{3.6}$$

where $\mathcal{M}(\lambda_t) = \{0, \lambda_t\}$ is a binary value function. We only keep the tested features ($\mathcal{M}(\lambda_t) = \lambda_t$) and remove others ($\mathcal{M}(\lambda_t) = 0$) to verify the effectiveness of different spectral features for CF. We replace $\hat{\mathbf{A}}$ with Equation (3.6) and apply it to two classic models: vanilla GCN [25] and LightGCN [41] that have been extensively adopted as baselines for CF, and conduct experiments on two datasets: CiteULike (sparse) and MovieLens-1M (dense) (see Table 3.1 for details). Figure 3.1 shows the results.

In Figure 3.1 (a) and (b), the accuracy is mainly contributed by a small portion of features that are rather smoothed or rough, while most features concentrated in the middle area (83% on CiteULike and 76% on MovieLens) which are not so rough or smoothed contribute as little as the randomly initialized signals to the accuracy and are barely useful. If we ignore the noisy features, it is obvious that the smoother features tend to outweigh the rougher features. In Figure 3.1 (c) and (d), we evaluate the importance of certain features by observing how accuracy changes after removing them. Slight and significant drops in accuracy are identified right after removing the top rough and smoothed features, respectively, while removing other features even results in an improvement.

The above observations raise a question: why the accuracy is mainly contributed by the top smooth and rough features? According to the definition, the two kinds of features actually represent the tendency of user behaviour: homophily and heterophily, that a user tends to interact with both similar others (*i.e.,* in the neighborhood) and different others (*i.e.,* not related to the user) [121], whereas most features that are not either rather smoothed or rough are less helpful to emphasize the two effects, thus contribute less to the accuracy. To summarize the analysis:

- Only a small portion of features that rather smoothed or rough are truly helpful for the recommendation.
- Smoothed features (homophily) outweigh the rough features (heterophily) for recommendation.



(a) on LightGCN. (b) An ideal filter.

Figure 3.2. The normalized weights for distinct graph features.

Note that this finding only applies to recommendation tasks, because the importance of different spectral features varies on tasks according to their data characteristics [74, 122].

Analysis and Limitations of Existing Work

Our analysis is based on LightGCN [41] which only keeps the essential component (*i.e.,* neigbhrhood aggregation), and we can rewrite it as:

$$\mathbf{O} = \sum_{k=0}^{K} \alpha_k \mathbf{H}^{(k)} = \sum_{k=0}^{K} \frac{\hat{\mathbf{A}}^k}{K+1}\mathbf{E} = \left(\sum_t \left(\sum_{k=0}^{K} \frac{\lambda_t^k}{K+1} \right) \mathbf{v}_t \mathbf{v}_t^T \right) \mathbf{E}. \tag{3.7}$$

We can see each spectral feature is weighted by a polynomial filter $\sum_{k=0}^{K} \alpha_k \lambda_t^k$ ($\alpha_k = \frac{1}{K+1}$). Here, we are interested in the weights of distinct graph features and plot them in Figure 3.2 (a), where the weight is normalized as $\frac{\sum_{k=0}^{K} \alpha_k \lambda_t^k}{\sum_{k=0}^{K} \alpha_k}$ (ratio to the maximum). As increasing the layer K, the model depresses the rough features and emphasizes the smoothed features which have been shown important for recommendation in Section 3.3.1. This finding shows the equivalence between exploiting neighborhood signals by stacking layers and emphasizing homophily. We can modify Equation (3.7) to emphasize heterophily as:

$$\mathbf{O} = \sum_{k=0}^{K} \frac{\mathcal{L}^k}{K+1}\mathbf{E}, \tag{3.8}$$

where $\mathcal{L} = \mathbf{I} - \hat{\mathbf{A}}$ is the normalized Laplacian matrix that measures the difference between each node and its neighborhood. It is easy to verify that the eigenvalue of $\mathcal{L}$ is $\lambda'_t = 1 - \lambda_t$, where the rough features are stressed and smoothed features are depressed. It is feasible to combine the two models to capture both the homophily and heterophily in user behaviour, while we argue that existing work suffers from more limitations that we need to propose a new GCN learning scheme to tackle them:

- The smoothed features are emphasized by repeating the multiplication of adjacency matrix, which is computationally expensive.
- The spectral features are weighted through a polynomial filter in a heuristic way, other designs should be discussed.
- The features shown useless still remain even stacking 100 layers, indicating the poor ability to denoise graph information.
- Stacking layers in GCNs results in the over-smoothing issue.
- The model cannot capture heterophily that might facilitate recommendation as the rough features are heavily depressed.

Evidently, a more flexible and effective design is required to replace the neighborhood-aggregation. Thus, here we plot an ideal design in Figure 3.2 (b): the noisy features are unnecessary that should be completely filtered out, while the useful features are reasonably measured according to their importance.

3.3.2 Proposed Method

We first formulate the general idea of GDE. Following previous analysis and Equation (3.6), we can partition an interaction graph $\mathcal{G}$ into smoothed $\mathcal{G}_S$, rough $\mathcal{G}_R$ and noisy $\mathcal{G}_N$ graphs, which are made up of smoothed, rough and noisy spectral features, respectively. The final representations are contributed by the embeddings generated on $\mathcal{G}_S$ and $\mathcal{G}_R$, while the embeddings from $\mathcal{G}_N$ are filtered out. Thus, GDE can be formulated as a band pass filter:

$$\gamma(u/i, \lambda_t) \begin{cases} \neq 0 & \mathbf{v}_t \in \mathcal{G}_S \, or \, \mathcal{G}_R \\ = 0 & \mathbf{v}_t \in \mathcal{G}_N. \end{cases} \tag{3.9}$$

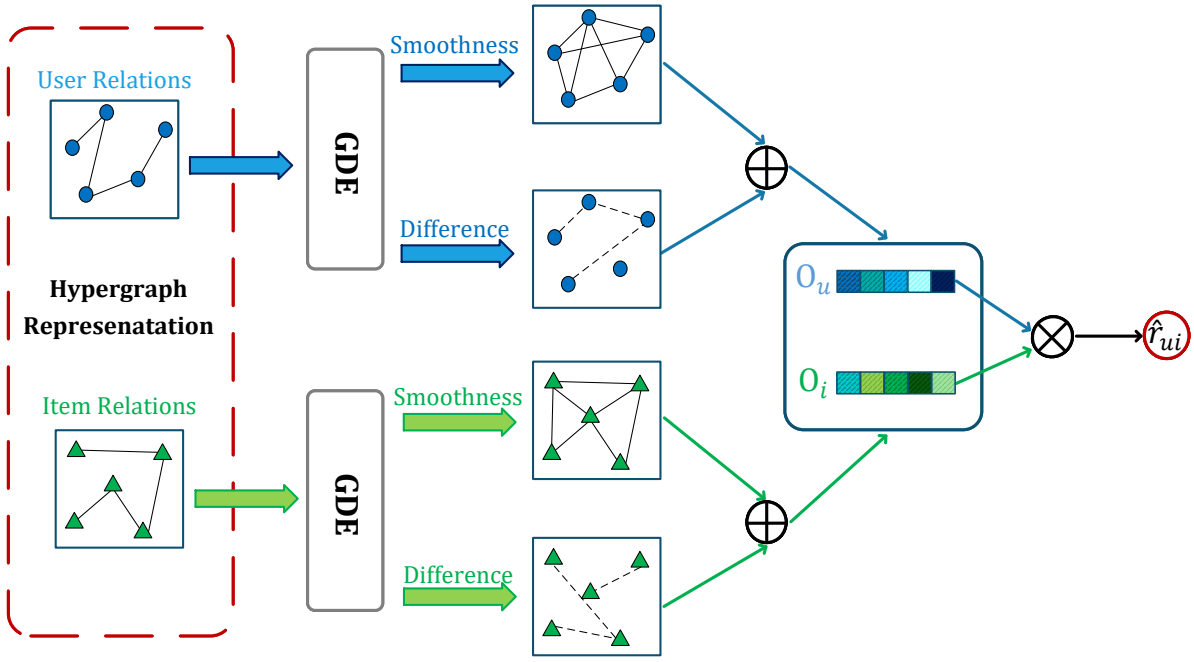


Figure 3.3. The Framework of our proposed GDE.

We replace the polynomial filter by a function $\gamma(\cdot)$. We argue that the importance of a spectral feature not only depends on the variation/eigenvalue but also is related to user/item, thus $\gamma(u, \lambda_t)$ outputs the importance of t-th feature to u. In practice, we only need to compute the feature from $\mathcal{G}_S$ and $\mathcal{G}_R$ as $\mathcal{G}_N$ is not fed into the model in the first place (*i.e.,* we do not need to consider the situation $\gamma(u, \lambda_t) = 0$).

To make graph representation more informative and powerful, we represent user-item interactions as hypergraphs:

$$\begin{aligned} \mathbf{A}_U &= \mathbf{D}_u^{-\frac{1}{2}}\mathbf{R}\mathbf{D}_i^{-1}\mathbf{R}^T\mathbf{D}_u^{-\frac{1}{2}} \quad \in \mathbb{R}^{M\times M}, \\ \mathbf{A}_I &= \mathbf{D}_i^{-\frac{1}{2}}\mathbf{R}^T\mathbf{D}_u^{-1}\mathbf{R}\mathbf{D}_i^{-\frac{1}{2}} \quad \in \mathbb{R}^{N\times N}, \end{aligned} \tag{3.10}$$

where $\mathbf{D}_u, \mathbf{D}_i$ are diagonal degree matrices of users and items. Equation (3.10) is consistent with the propagation matrix of hypergraph neural network [123], the definition and analysis in the previous section is still applicable here. We treat items (users) as hyper-edges when considering user (item) relations.

Graph Denoising Encoder (GDE)

We illustrate the proposed GDE in Figure 3.3, where we propagate embeddings on hypergraphs which emphasize the neighborhood smoothness and difference. The embeddings generated on the smoothed hypergraphs are formulated as follows:

$$\begin{aligned}\mathbf{H}_U^{(s)} &= \left(\mathbf{P}^{(s)} \odot \gamma\left(\mathcal{U}, \pi^{(s)}\right) \mathbf{P}^{(s)^T}\right) \mathbf{E}_U, \\ \mathbf{H}_I^{(s)} &= \left(\mathbf{Q}^{(s)} \odot \gamma\left(\mathcal{I}, \sigma^{(s)}\right) \mathbf{Q}^{(s)^T}\right) \mathbf{E}_I,\end{aligned} \tag{3.11}$$

where $\{\mathbf{P}^{(s)} \in \mathbb{R}^{M \times m_1}, \pi^{(s)} \in \mathbb{R}^{m_1}\}$, $\{\mathbf{Q}^{(s)} \in \mathbb{R}^{N \times n_1}, \sigma^{(s)} \in \mathbb{R}^{n_1}\}$ are top m_1 and n_1 smoothed {features, eigenvalues} for user and item relations ($\mathbf{A}_U$ and $\mathbf{A}_I$), respectively. The term in parentheses represents the node relations on the smoothed graph; $\gamma(\cdot)$ outputs the importance of distinct features to users/items, $\odot$ stands for the element-wise multiplication. $\mathbf{E}_U$ and $\mathbf{E}_I$ are embedding matrices for users and items, respectively. To improve generalization on test sets, we randomly drop out the node relations with a ratio $p \in [0, 1]$. Similarly, we propagate embeddings on the rough hypergraphs with top $m_2 + n_2$ rough features to learn the heterophily:

$$\begin{aligned}\mathbf{H}_U^{(r)} &= \left(\mathbf{P}^{(r)} \odot \gamma\left(\mathcal{U}, \pi^{(r)}\right) \mathbf{P}^{(r)^T}\right) \mathbf{E}_U, \\ \mathbf{H}_I^{(r)} &= \left(\mathbf{Q}^{(r)} \odot \gamma\left(\mathcal{I}, \sigma^{(r)}\right) \mathbf{Q}^{(s)^T}\right) \mathbf{E}_I.\end{aligned} \tag{3.12}$$

Similarly, $\{\mathbf{P}^{(r)}, \pi^{(r)}\}$, $\{\mathbf{Q}^{(r)}, \sigma^{(r)}\}$ are top m_2 and n_2 rough {features, eigenvalues} for user and item relations, respectively. We let $m = m_1 + m_2$, $n = n_1 + n_2$, and generate the final embeddings by:

$$\begin{aligned}\mathbf{O}_U &= \mathbf{pooling}\left(\mathbf{H}_U^{(s)}, \mathbf{H}_U^{(r)}\right), \\ \mathbf{O}_I &= \mathbf{pooling}\left(\mathbf{H}_I^{(s)}, \mathbf{H}_I^{(r)}\right).\end{aligned} \tag{3.13}$$

To keep the model simple and avoid bringing additional complexity, we take summation for pooling function. According to our analysis in Section 3.3.1, stacking layers in GCNs is essentially reweighting the spectral features. Since a single-layer GDE is capable of reasonably weighting the important graph features for recommendation, it is unnecessary to stack more layers.

Measuring the Importance of Graph Features

There are two directions for the design of $\gamma(\cdot)$: a dynamic design by parameterizing $\gamma(\cdot)$, or a well motivated static design by manually adjusting hyper-parameters without introducing parameters. Here, we introduce a instantiation for each of them. For simplicity, we define $\mathbf{P} = [\mathbf{P}^{(s)} \,||\, \mathbf{P}^{(r)}]$, $\pi = [\pi^{(s)} \,||\, \pi^{(r)}]$ and $\mathbf{Q} = [\mathbf{Q}^{(s)} \,||\, \mathbf{Q}^{(r)}]$, $\sigma = [\sigma^{(s)} \,||\, \sigma^{(r)}]$ (|| is the concatenate operation).

• **Option I**. We first encode variation into spectral features and define variation-encoded feature matrices as: $\mathbf{P}' = \mathbf{P}diag(\pi)$, $\mathbf{Q}' = \mathbf{Q}diag(\sigma)$. The importance of a spectral feature to a user is generated via attention mechanism:

$$\gamma(u, \pi_g) = \sigma\left(\mathbf{a}^T \left[\mathbf{W}_U^{(1)} \mathbf{P}_u'^T \,||\, \mathbf{W}_U^{(2)} \mathbf{P}'_g\right]\right), \tag{3.14}$$

where $\mathbf{P}_u'^T$ is a feature vector of u, $\mathbf{P}'_g$ is g-th spectral feature; $\mathbf{W}_U^{(1)} \in \mathbb{R}^{d_1 \times m}$, $\mathbf{W}_U^{(2)} \in \mathbb{R}^{d_1 \times M}$ are transform matrices, attention mechanism is parameterized as a single-layer neural network where $\mathbf{a} \in \mathbb{R}^{2d_1}$. We can learn the importance of a spectral feature to an item similarly. Finally, we normalize the scores across features using the softmax function.

• **Option II**. We first analyze what static design could lead to better results. By considering $\gamma(\cdot)$ as a one variable continuous function of eigenvalues (*i.e.,* ignore the effect from user/item first), we can rewrite the term in parentheses in Equations (3.11) and (3.12) as follows according to the Taylor series:

$$\mathbf{P}diag(\sum_{k=0}^{K} \alpha_k \pi^k)\mathbf{P}^T = \sum_{k=0}^{K} \alpha_k \mathbf{P}diag(\pi^k)\mathbf{P}^T = \sum_{k=0}^{K} \alpha_k \bar{\mathbf{A}}_U^k, \tag{3.15}$$

where $\bar{\mathbf{A}}_U^k$ can be considered as an adjacency matrix with noisy features being removed, $\bar{\mathbf{A}}_U^k = \mathbf{A}_U^k$ when $m \to M$. K is the highest order with non-zero derivative, $\alpha_k = \frac{\gamma^{(k)}(0)}{k!}$ is the coefficient of k-th order Maclaurin expansion. On the other hand, from a spatial perspective, K is also the order of the farthest incorporated neighborhood and α_k is the contribution of k-th order neighborhood. Intuitively, we hope the model can capture neighbor signals as far as possible with positive contributions to user/item representations, implying that $\alpha_k > 0$, $K \to \infty$. In other words, $\gamma(\cdot)$ should be infinitely differentiable whose any-order derivative is positive. To satisfy this condition and after extensive experiments (shown in Section 3.4.4), we use a exponential kernel: $\gamma(\pi) = e^{\beta\pi}$, where $\beta \in \mathbb{R}$ controls the extent of the emphasis over different features (*i.e.,* a larger (smaller) β emphasize

the neighbor smoothing (difference) more). We can rewrite Equation (3.15) as $\mathbf{A}'_U = \sum_{k=0}^{\infty} \frac{\beta^k}{k!} \bar{\mathbf{A}}_U^k$, and is comparable to a GCN with infinite layers if we further rephrase GDE in the form of the GCN paradigm for CF:

$$\begin{aligned} \mathbf{H}^{(k+1)} &= \bar{\mathbf{A}}_U \mathbf{H}^{(k)}, \\ \mathbf{O}_U &= \lim_{K \to \infty} \sum_{k=0}^{K} \frac{\beta^k}{k!} \mathbf{H}^{(k)}. \end{aligned} \tag{3.16}$$

Furthermore, the importance of a spectral feature might vary on different users/items as well. For instance, the users/items with low degrees are isolated on the graph, thus the smoothing effect should be emphasized more than the nodes with high degrees. To this end, we modify as $\gamma(u, \pi_g) = e^{(\beta + (-log(d_u)))\pi_g}$ to adapt to different users/items.

In our experiments, we choose **Option II** as it shows improvement over **Option I** across all datasets. Here, we attempt to analyze the limitations of an adaptive design. Firstly, with a parameterized design, the terms in parentheses in Equations (3.11) and (3.12) need to be repeated during each epoch of training, which is computationally expensive and unnecessary when using an static design. Secondly, we notice that introducing parameters to model the importance of features results in even worse convergence and accuracy. We speculate the reason is due to the sparseness of the datasets. Unlike other tasks, the available data for CF is only the user/item ID, which is difficult to learn the data intrinsic characteristics in an adaptive manner. We will compare the two designs in Section 3.4.4.

3.3.3 Discussion

Over-Smoothing

Definition 2. *A model suffers from the over-smoothing if any spectral features dominate as the model layer K is large enough:*

$$\lim_{K \to \infty} \frac{|\gamma(\lambda_t)|}{\mathbf{max}\{|\gamma(\lambda_1)|, \cdots, |\gamma(\lambda_{M+N})|\}} \to 0. \tag{3.17}$$

Over-smoothing in GCNs [42, 124] refers to overweight of the smoothest feature when increasing the layer K, eventually all features except the smoothest one loses and it results in the same user/item representations. Most GCN-based CF

methods suffer from this limitation and remain shallow, take LightGCN as an example:

$$\text{LightGCN}: \lim_{\substack{\alpha_k=\frac{1}{K+1} \\ K\to\infty}} \frac{\gamma(\lambda_t)}{\gamma(\lambda_{\max})} = \frac{\sum_{k=0}^{K} \frac{\lambda_t^k}{K+1}}{\sum_{k=0}^{K} \frac{1}{K+1}} \to 0. \tag{3.18}$$

The key to prevent over-smoothing is to properly and reasonably model the importance of features too assure any features are not overweighted. On the other hand, instead of controlling the weight through neighborhood aggregation, we adjust the weight of different features through a flexible and light design (*i.e.,* **Option II**). According to Definition 2, it is easy to verify that GDE does not suffer from over-smoothing: $\frac{e^{\beta\pi_g}}{e^{\beta\pi_{\max}}} \neq 0$, $\frac{e^{\beta\sigma_h}}{e^{\beta\sigma_{\max}}} \neq 0$.

Time Complexity

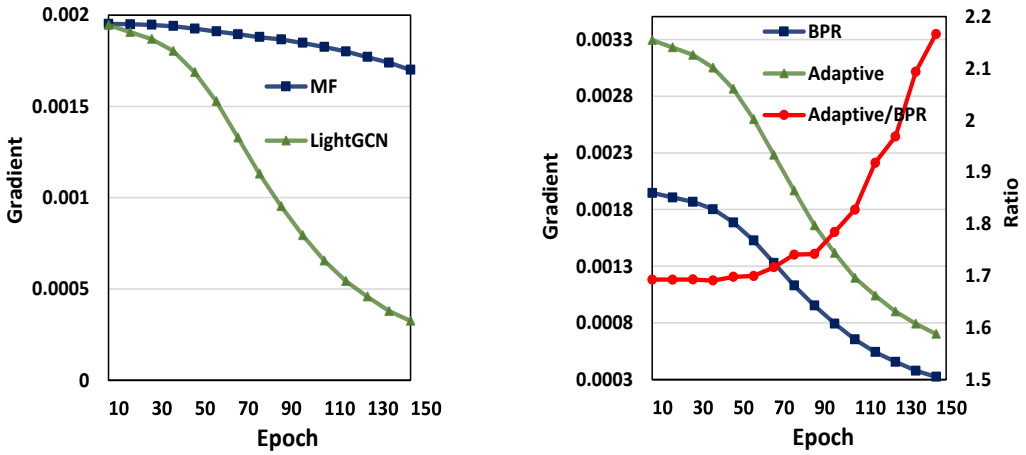
The complexity of our model mainly comes from retrieving of required graph features (preprocessing) and the training. We can calculate spectral features through algorithms such as Lanczos method and LOBPCG [125, 126] with GPU implementation. For instance, the complexity of Lanczos method is $\mathcal{O}(m^2M + m|\mathcal{E}_{\mathbf{A}_U}| + n^2N + n|\mathcal{E}_{\mathbf{A}_I}|)$ [125], where $\mathbf{A}_U$ and $\mathbf{A}_I$ are sparse, and $m \ll M$, $n \ll N$, $\mathcal{E}_{\mathbf{A}_U}$ and $\mathcal{E}_{\mathbf{A}_I}$ are edges of $\mathbf{A}_U$ and $\mathbf{A}_I$, respectively. Since $\mathbf{A}_U$ and $\mathbf{A}_I$ are sparse and required features only account for a small portion, the complexity can be controlled at a low level. During training, each user/item can be considered as a multi-hot vector, and the final embedding is generated in a node level rather than a matrix level. The complexity for GDE is $\mathcal{O}((1-p)(M+N)dc|\mathbf{R}^+|)$, where c is the number of epochs.

3.3.4 Optimization

It has been reported that some GCN-based methods show slow training convergence. We argue that the issue lies in the commonly used BPR loss for pairwise learning [45]:

$$\mathcal{L}_{BPR} = -\sum_{(u,i,j)\in T} \ln \sigma(\hat{\mathbf{r}}_{ui} - \hat{\mathbf{r}}_{uj}), \tag{3.19}$$

where $T = \{(u,i,j)|(u,i) \in \mathbf{R}^+, (u,j) \in \mathbf{R}^-\}$, $\sigma(\cdot)$ is a sigmoid function. The loss optimizes based on sampled triples by maximizing the difference between observed interactions and unobserved interactions. Rendle and Frendenthaler [127]



(a) The gradients on negative samples vanishes more quickly on LightGCN than MF.

(b) This issue is eased by adaptively adjusting the gradients on negative samples.

Figure 3.4. A training issue of pairwise learning and a solution to it.

show that the positive rating $\hat{\mathbf{r}}_{ui}$ increases quickly due to the tailed item distributions (*i.e.,* observed interactions are more likely to be sampled than unobserved ones). As a result, the term controlling the gradient magnitude $(1 - \sigma(\hat{\mathbf{r}}_{ui} - \hat{\mathbf{r}}_{uj}))$ decreases quickly and prevents the model from learning from (negative) training pairs. Figure 3.4 (a) shows that LightGCN suffers more from this issue than MF, which explains why it requires many epochs to converge. To avoid bringing additional complexity, we propose to adaptively adjust the gradient to expedite model training:

$$\delta_{uj} = 1 - \log(1 - \min(\sigma(\hat{\mathbf{r}}_{uj}), \xi)). \tag{3.20}$$

Since this issue is due to the over-sampling of positive items, we adjust the gradients over negative samples. δ_{uj} is a self-paced but non-trainable parameter changing according to $\hat{\mathbf{r}}_{uj}$; when $\sigma(\hat{\mathbf{r}}_{uj})$ deviates too far from 0, δ_{uj} becomes large to accelerate the model training. $\min(\sigma(\hat{r}_{uj}), \xi)$ is to prevent δ_{uj} being too large, and we set $\xi = 0.99$. The loss function is enhanced as follows:

$$\mathcal{L}_{adapt} = - \sum_{(u,i,j) \in T} \ln \sigma(\hat{\mathbf{r}}_{ui} - \delta_{uj} \hat{\mathbf{r}}_{uj}) + \lambda \left\| \Theta \right\|^2, \tag{3.21}$$

where λ controls regularization strength. We compare two loss functions in Figure 3.4 (b) by showing the gradient with respect to $\hat{\mathbf{r}}_{uj}$. Obviously, the negative items consistently receive larger gradients from the adaptive loss than the plain BPR loss, thereby helping accelerate training.

3.4 Experiments

In this section, we comprehensively evaluate GDE. In particular, we aim to answer the following research questions:

- **RQ1**: Does GDE outperform other baselines?
- **RQ2**: Is GDE more efficient than GCN-based methods?
- **RQ3**: Does the proposed designs show positive effects? How do hyper-parameters affect the performance?

3.4.1 Experimental Setup

Datasets

The descriptions of datasets are listed as follows. The statistics of all five datasets are summarized in Table 3.1.

- **Pinterest**: This is an implicit feedback dataset [17] for content-based image recommendation, where users can pin image they are interested in.
- **CiteULike-a**: This dataset* is collected from CiteULike which allows users to create their own collections of articles.
- **MovieLens**: These two datasets (1M and 100K)† have been widely used to evaluate CF algorithms. Since it is an explicit feedback dataset while we focus on implicit feedback, we hide all ratings.
- **Gowalla**: The interactions in this dataset [30] are check-ins which record the locations the user has visited.

Evaluation Metrics

We adopt two widely-used metrics: Recall and nDCG for personalized ranking [128]. Recall measures the ratio of recommended items in the test set; nDCG considers the position of items by assigning a higher weight to the item ranking

*https://github.com/js05212/citeulike-a
†https://grouplens.org/datasets/movielens/

Table 3.1. Statistics of datasets

Datasets	#User	#Item	#Interactions	Density%
CiteULike-a	5,551	16,981	210,537	0.223
MovieLens-1M	6,040	3,952	1,000,209	4.190
MovieLens-100K	943	1,682	100,000	6.305
Pinterest	37,501	9,836	1,025,709	0.278
Gowalla	29,858	40,981	1,027,370	0.084

Table 3.2. Overall performance comparison in terms of nDCG@20 and Recall@20. Improv.% denotes the improvements over the best baselines.

	CiteULike		**Pinterest**		**MovieLens-1M**		**Gowalla**		**MovieLens-100K**	
	nDCG	**Recall**	**nDCG**	**Recall**	**nDCG**	**Recall**	**nDCG**	**Recall**	**nDCG**	**Recall**
BPR	0.0591	0.0527	0.0861	0.0809	0.4849	0.4578	0.0907	0.0743	0.4935	0.4641
Ease	0.0846	0.0801	0.0695	0.0639	0.3249	0.3000	0.0670	0.0642	0.3523	0.3214
LCFN	0.0662	0.0590	0.0937	0.0873	0.5197	0.4898	0.1132	0.0980	0.5199	0.4898
GF-CF	0.0836	0.0811	0.0776	0.0755	0.4789	0.4562	0.0537	0.0567	0.4048	0.3793
ELGN	0.1125	0.1027	0.1176	0.1098	0.5418	0.5133	0.1249	0.1138	0.5347	0.5100
LightGCN	0.1149	0.1066	0.1143	0.1069	0.5261	0.5031	0.1327	0.1224	0.5418	0.5133
SGL-ED	0.1070	0.0985	0.1185	0.1094	0.5314	0.5035	0.1561	0.1353	0.5321	0.5044
GDE	**0.1339***	**0.1224***	**0.1240***	**0.1147***	**0.5715***	**0.5423***	**0.1632***	**0.1449***	**0.5731***	**0.5400***
GDE-d	0.1126	0.1026	0.1157	0.1080	0.5578	0.5306	0.1462	0.1341	0.5582	0.5280
Improv.%	+16.54	+14.82	+3.29	+4.46	+5.48	+5.65	+4.55	+7.10	+5.77	+5.20
p-value	4.10e-9	2.77e-8	2.94e-4	3.71e-7	6.71e-7	5.44e-4	6.15e-5	5.44e-4	3.88e-6	4.29e-6

higher. The recommendation list is generated by ranking unobserved items and truncating at position k. As the success of GCNs lies in the ability of exploiting high-order neighbor to tackle data sparsity which is common in practice, we use only 20% of the user-item pairs for training to evaluate the model stability with limited interactions, and leave the remaining for test; we randomly select 5% from the training data as validation set for hyper-parameter tuning. We report the average accuracy on test sets.

Baselines

We compare our proposed method with the following CF methods. The architecture settings are based on the reported results in each paper:

- BPR [45]: This method proposes a pair-wise ranking loss by maximizing the

difference between observed and unobserved interactions.

- EASE [129]: This is a neighborhood-based method which is considered as a SLIM [130] variant with a closed form solution.
- LCFN [91]: This model proposes a low pass graph convolution to replace the vanilla graph convolution and initializes the embeddings with pretrained MF. We set $F = 0.1$ on CiteULike and $F = 0.005$ on other datasets.
- LightGCN [41]: Unlike other GCN methods, this model exploits a light GCN model for CF by removing non-linear activation functions and transformation layers where the model complexity is the same as MF.
- GF-CF [80]: This is a GCN-based CF method without model optimization, exploiting low frequency components and has a low time complexity.
- EGLN [120]: This model uses an adaptive user-item graph structure and deigns a local-global consistency optimization function via mutual information maximization to better serve CF. We set $\alpha = 0.1$ and $\beta = 0.1$.
- SGL-ED [38]: This model contrasts different node views that are generated by randomly masking the edge connections on the graph, and incorporate the proposed self-supervised loss into LightGCN [41]. We set $\tau = 0.2$, $\lambda_1 = 0.1$ and $p = 0.1$.

We remove popular GCN-based methods such as GCMC [29], SpectralCF [82], Pinsage [93], NGCF [30] as the baselines above have shown superiority over them.

Implementation details

We implemented the proposed model based on PyTorch[‡], and released the code on Github[§]. For all models, the optimizer is SGD; the embedding size d is set to 64 and $d_1 = 16$; the regularization rate is set to 0.01 on all datasets; the learning rate is tuned amongst $\{0.001, 0.005, 0.01, \cdots\}$; the drop ratio of node relations is tune amongst $p = \{0.1, 0.2, \cdots\}$; without specification, the model parameters are initialized with Xavier Initialization [131]; the batch size is set to 256. We report the hyper-paramter setting: $\frac{m}{M}/\frac{n}{N} = \{0.01, 0.05, 0.1, 0.2, \cdots\}$, $\beta = \{1, 1.5, 2, 2.5, \cdots\}$ in the next subsection.

[‡]https://pytorch.org/

[§]https://github.com/tanatosuu/GDE

3.4.2 Overall Comparison (RQ1)

We report the performance of baselines and our GDE variants in Table 3.2, where GDE-d is a GDE variant using Option I. We observe the followings:

- Overall, GCN-based methods outperform traditional CF methods when data suffers from extreme sparsity, indicating the effectiveness of GCNs to tackle the data sparsity. EGLN, LightGCN, and SGL-ED alternately achieve the best baseline, while our proposed GDE consistently outperforms all baselines across all datasets, indicating the superiority and stability of our proposed method.

- Among GCN-based methods, LCFN and GF-CF show relatively poor performance compared with other methods. We speculate that the parameterized kernel used in LCFN fails to learn the importance of low frequency components, such an design even reduces the accuracy and hinders the convergence. As for GF-CF, a reasonable explanation is GF-CF fails to perform stably with limited interactions due to the lack of model optimization. The sparse setting adopted in our work increases the difficulty to learn from data and to perform stably, which is also critical to evaluate recommendation models.

- Since our work improves the GCN architecture for CF, it is more fair to compare with pure GCN-based methods such as LightGCN. The improvement of GDE over LighGCN is more significant on sparse data (*e.g.,* 23.0% on Gowalla in terms of nDCG@20) than dense data (*e.g.,* 5.8% on 100K), indicating the effectiveness of GDE to tackle data sparsity.

- Although GDE-d shows competitive performance over baselines, it still significantly underperforms GDE, indicating that it is difficult to learn the data characteristic in an adaptive manner, while a well motived static design might result in a promising accuracy. We will compare the two designs in terms of complexity and performance in detail in the latter section.

3.4.3 Efficiency of GDE (RQ2)

Figure 3.5 shows the preprocessing time. We can see the model shows superior performance with only a small portion of spectral features. For instance, GDE outperforms the best baseline with less than 10%, 5%, 1%, 5% spectral features

Table 3.3. The comparison of the training time (seconds) per epoch on Gowalla.

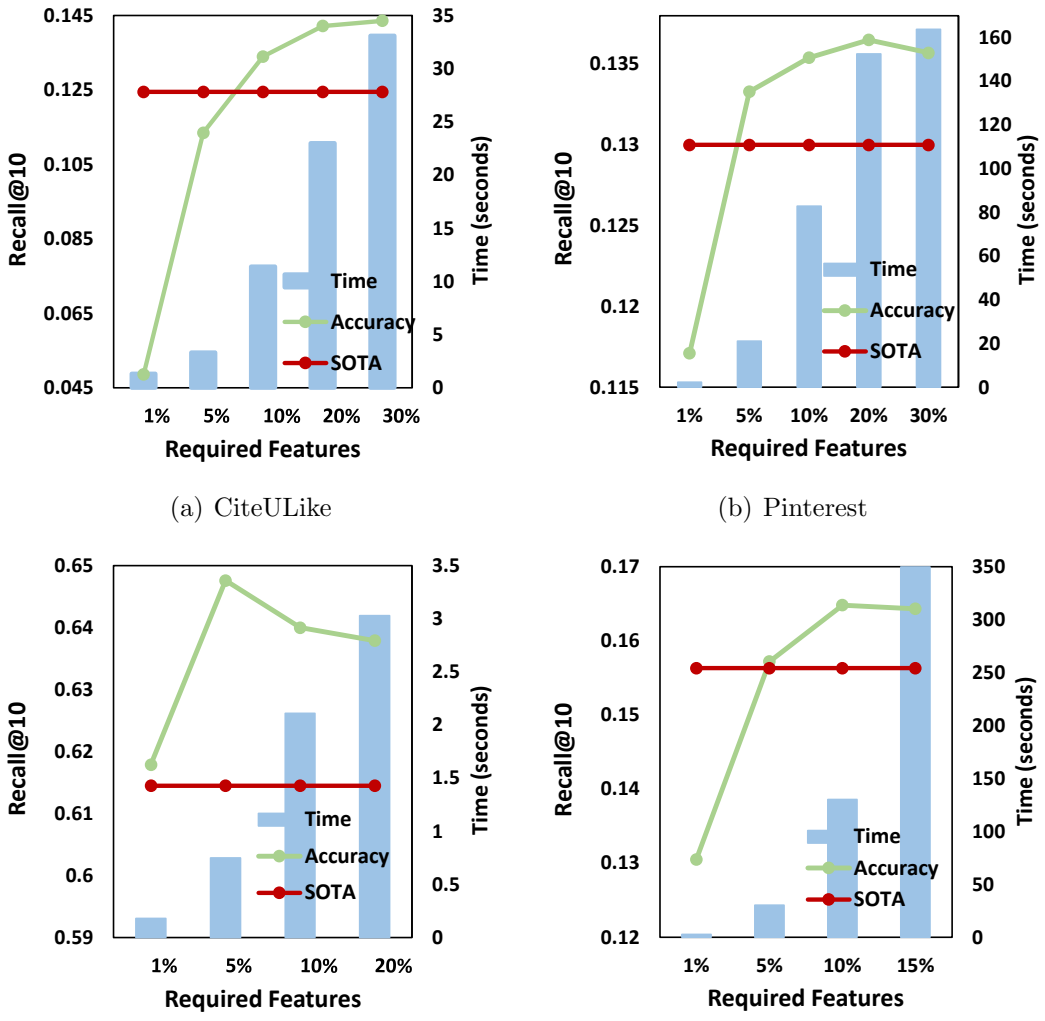
	CiteULike	Pinterest	ML-1M	Gowalla	Epochs
ELGN	13.90	268.7	26.5	770.1	260
LightGCN	1.17	7.7	8.0	9.8	600
LCFN	33.71	154.3	4.8	269.7	350
BPR	0.44	1.9	1.8	3.3	450
GDE	0.39	1.8	1.8	3.1	120
GDE-d	3.15	26.7	3.3	34.5	180

on the datasets in Figure 3.5 (a) to (d), respectively, that justifies our previous analysis that only a very small portion of graph features is useful for recommendation. In practice, we can adjust the number of graph features in order to balance between the computational complexity and model accuracy. Table 3.3 shows the training times per epoch of several methods. Since the batch size is different on baselines, we test with fixed batch size to compare their time complexity fairly. The training epochs are obtained on the optimal batch-size settings reported in the papers. Since GF-CF requires no model training and SGL-ED is compared with LightGCN in terms of time complexity in the original literature, we ignore them. Among GCN baselines, LightGCN is the most efficient one, as it only keeps the core component (*i.e.,* neighborhood aggregation) for training. On the other hand, the training time of GDE barely increases with the size of datasets, which is as fast as BPR and much faster than LightGCN; GDE-d takes more training time as Equations (3.11) and (3.12) need to be repeated during each epoch of training. Overall, the running epochs of GDE, LightGCN, BPR are 120, 600, 450, and the whole running times are 502s (including preprocessing time), 5880s, 1480s, respectively; GDE has around 12x, 3x speed-up compared with LightGCN and BPR, respectively.

3.4.4 Study of GDE (RQ3)

How Rough and Smoothed Features Contribute?

As shown in Figure 3.6, the rough features behave like inherent features of the graph as it contributes to the accuracy without much model training and the training loss barely drops. On the other hand, the accuracy of the smoothed



(a) CiteULike

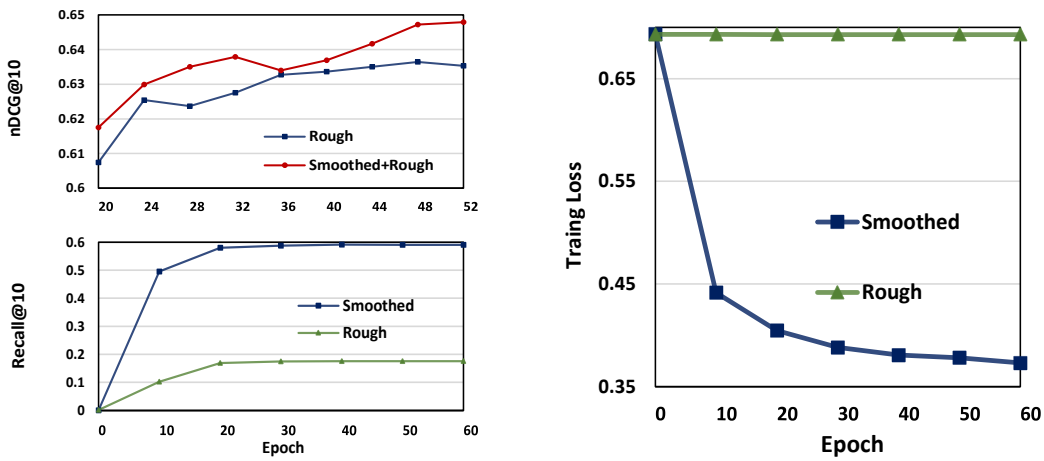
(b) Pinterest

(c) MovieLens-1M

(d) Gowalla

Figure 3.5. How the accuracy and preprocessing time (y-axis) change with the required spectral features (x-axis). SOTA represents the accuracy of the best baseline (*i.e.,* LightGCN on CiteULike, SGL-ED on Pinterest and Gowalla, ELGN on ML-1M).

features significantly increases as the training proceeds, in the meanwhile the training loss drops sharply. Normally, we set $m_2 \leq 0.1 \times m_1$, since the accuracy is mainly contributed by the smoothed features while the rough features show a slight improvement to the accuracy. We also notice the improvement of rough features tends to be significant on dense datasets (*e.g.,* MovieLens) and is hard to be identified on sparse datasets (*e.g.,* CiteULike). A reasonable explanation is



(a) Accuracy (b) Training Loss

Figure 3.6. How rough and smoothed features affect accuracy and training loss as the training proceeds.

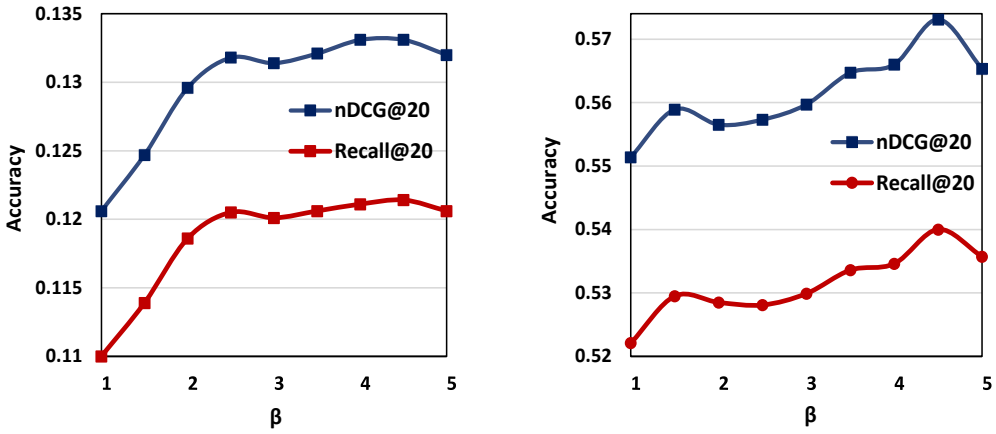
Table 3.4. The accuracy of different designs to measure the importance of spectral features on CiteULike.

Design	Function	nDCG@10	Property		
			Increasing	Pos Coef.	Infinite
Static	$\log(\alpha\lambda_t)$	0.1343	✓	×	✓
	$\sum_k \alpha_k \lambda_t^k$	0.1434	✓	✓	×
	$\frac{1}{1-\alpha\lambda_t}$	0.1464	✓	✓	✓
	$e^{\beta\lambda_t}(\beta > 0)$	0.1518	✓	✓	✓
	$e^{\beta\lambda_t}(\beta < 0)$	0.0322	×	×	✓
Dynamic	Attention	0.1296			

that the rough features can help emphasize the difference among nodes on dense datasets, where nodes are relatively connected to each other, whereas they are less important on sparse datasets as the nodes are already isolated on the graph.

What Kind of $\gamma(\cdot)$ Design Works Better?

Table 3.4 lists some designs for weighting functions $\gamma(\cdot)$, and (1) Increasing, (2) Pos Coef, and (3) Infinite refer to: if it (1) is an increasing function, (2) has positive coefficients of Taylor series and (3) is infinitely differentiable, respectively. From the top to bottom, we set $\alpha = 10$, $\alpha_k = 1$, $\alpha = 0.9$, $\beta = 4$, $\beta =$ -2. We can see the importance of the three properties is (1)≫(2)¿(3). The model shows



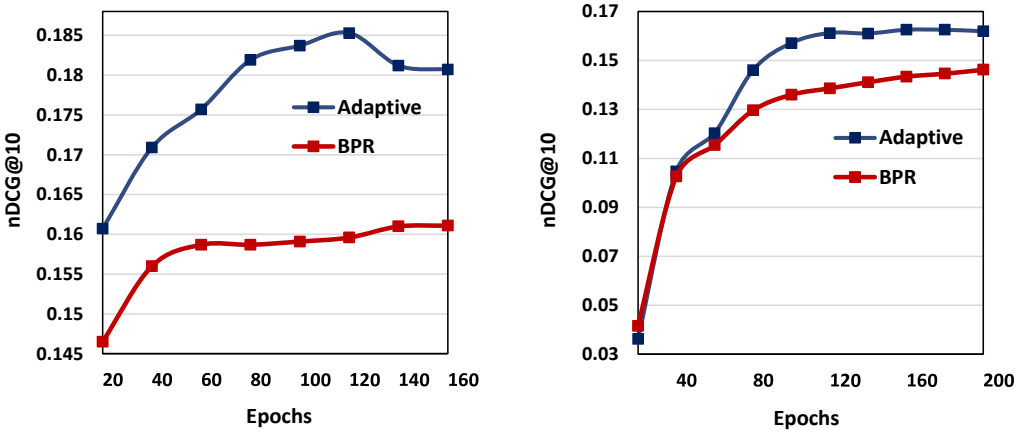
(a) CiteULike. (b) MovieLens-100K.

Figure 3.7. Accuracy of GDE with varying β.

poor performance when $\gamma(\cdot)$ is a decreasing function (*i.e.,* the rougher features have higher importance), justifying our previous analysis that the smoother features are more important. Overall, the designs that satisfy all three properties outperform other designs. On the other hand, we notice that the dynamic design underperforms the static designs with an increasing function, which demonstrates that the dynamic design fails to learn the importance of different features, otherwise it should perform closely to above static designs. In addition, the running time comparison in Table 3.3 also shows that a static design runs much faster than the dynamic design. Based on the above analysis and experimental results, we conclude that a static design is more effective and efficient for CF.

Effect of β

Figure 3.7 shows the accuracy of GDE with varying β on two datasets, where similar trends are observed on other datasets as well. We observe consistent improvements when the smooth features are emphasised (β becomes larger). Particularly, the best accuracy is achieved at $\beta = 4.5$ on both datasets, and the accuracy drops by 9.4%, 3.5% at $\beta = 1$ compared with the best accuracy on CiteULike and 100K, respectively. We speculate the degradation is larger on CiteULike is because the smoothed features are more important on sparse data, on which users and items have fewer interactions and are more isolated on the graph.



(a) GDE (b) LightGCN

Figure 3.8. The adaptive loss helps accelerate GCN training.

Table 3.5. The accuracy of several GDE-variants evaluated by Recall@20.

	CiteULike	Pinterest	ML-1M	Gowalla	ML-100K
plain	0.1146	0.1109	0.5303	0.1353	0.5287
drop	0.1198	0.1116	0.5390	0.1382	0.5400
drop+adaptive	0.1224	0.1147	0.5423	0.1449	0.5275
LightGCN	0.1066	0.1069	0.5031	0.1224	0.5133

Ablation Study

As shown in Table 3.5, we propose three variants: (1) with BPR loss and without dropout (plain), (2) with BPR loss and dropout (drop), (3) with adaptive loss and dropout (drop+adaptive).

The effect of dropout. Edge dropout is a commonly used technique to improve generalization on test data. We can identify the positive effect of it as a model with dropout shows improvements over the model without it across all datasets. In addition, we observe that the improvements on dense data (*e.g.,* MovieLens 100K) tend to be more significant than sparse data (*e.g.,* Pinterest).

The effect of adaptive loss. As shown in Figure 3.8, the adaptive loss largely reduces the training epochs of GCN-based methods including but not limited to GDE. From the reported results in Table 3.5, the adaptive loss also results in improvements on 4 out of 5 datasets. We speculate that the reason it reduces the accuracy on 100K is closely related to the data density; since we propose the

adaptive loss to help alleviate the undersampling of the negative samples, where such issue is not serious on dense data as the negative samples are more likely to be sampled.

The effectiveness of GDE. It is fair to separate the improvement from the adaptive loss from overall improvements, as it can be applied on other GCN-based methods as well. We can see GDE with a common loss still outperforms competitive baselines including LightGCN, proving the effectiveness of our proposed designs.

3.5 Summary

In this chapter [132], we explored how GCNs facilitate recommendation and what graph information matters for recommendation from a spectral perspective. We especially showed how distinct spectral graph features contribute to the accuracy, found that only a very small portion of spectral features that emphasize the neighborhood smoothness and difference are truly helpful for recommendation. We then unveiled the effectiveness of GCNs by showing that stacking layers in GCNs emphasizes the smoothness. Based on the two important findings above, we pointed out the limitations of existing GCN-based CF methods and proposed a Graph Denoising Encoder (GDE) to replace neighborhood aggregation with a simple yet effective architecture. Finally, to tackle a slow convergence issue on GCN-based methods, we proposed an adaptive loss to dynamically adjust the gradients over negative samples, accelerating the model training and resulting in improvement. Extensive experiments conducted on five datasets not only demonstrate the effectiveness and efficiency of our proposed methods but also justifies our analysis and findings. We believe that the insights of our work are inspirational to future developments of GCN architecture designs for recommender systems. In future work, we plan to analyze the potential of GCNs from other perspectives and apply our proposed models to other recommendation tasks.

CHAPTER 4

A Simplified Graph Convolution Paradigm for Recommendation

In this chapter, we demystify GCNs by showing close connection between GCN-based and low-rank CF methods such as Singular Value Decomposition (SVD) and Matrix Factorization (MF), that stacking graph convolution layers is to learn a low-rank representation by emphasizing (suppressing) components corresponding to 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.

4.1 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) [11], 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.) [20, 4, 133] and exploit advanced algorithms [17, 22, 134] 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) [25] have shown great potential in various fields including social network analysis [119, 35] and recommender systems [29, 93]. Much research effort has been devoted to adapt GCNs for recommendation, such as augmenting GCNs with other advanced algorithms [40, 38, 39], simplifying GCNs to improve training efficiency and model effectiveness [41, 95, 135], 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 [42, 136], 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.

4.2 preliminaries

4.2.1 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 $\mathbf{R} \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$, 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 $\mathcal{E} = \mathbf{R}^+$ is represented by observed interactions, where $\mathbf{R}^+ = \{r_{ui} = 1 | u \in \mathcal{U}, i \in \mathcal{I}\}$. Each user/item is considered as a node on the graph and parameterized as an embedding vector $\mathbf{e}_u / \mathbf{e}_i \in \mathbb{R}^d$. The core idea of GCNs is to update user and item embeddings by propagating them on the graph. The adjacency relations are represented as:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^T & \mathbf{0} \end{bmatrix}. \tag{4.1}$$

The updating rule of GCNs is formulated as follows:

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l+1)}\right), \tag{4.2}$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is a symmetric normalized adjacency matrix, $\mathbf{D}$ is a diagonal node degree matrix. The initial state is $\mathbf{H}^{(0)} = \mathbf{E}$, where $\mathbf{E} \in \mathbb{R}^{(|\mathcal{U}|+|\mathcal{I}|)\times d}$ contains users' and items' embedding vectors. Recent works [41, 95] show the non-linear activation function $\sigma(\cdot)$ and feature transformations $\mathbf{W}^{(l+1)}$ are redundant for CF , the above updating rule can be simplified as follows:

$$\mathbf{H}^{(l)} = \tilde{\mathbf{A}}^l\mathbf{E}. \tag{4.3}$$

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

$$\mathbf{O} = \mathbf{pooling}\left(\mathbf{H}^{(l)} | l = \{0, 1, \cdots, L\}\right). \tag{4.4}$$

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

$$\hat{r}_{ui} = \mathbf{o}_u^T \mathbf{o}_i. \tag{4.5}$$

4.2.2 Low-Rank Methods

Low rank representation plays a fundamental role in modern recommender systems [11]. The core idea of low-rank methods is inspired by Singular Value Decomposition (SVD):

$$\mathbf{R} = \mathbf{U} diag\left(s_k\right) \mathbf{V}^T \approx \sum_{k=1}^{K} s_k \mathbf{u}_k \mathbf{v}_k^T. \tag{4.6}$$

The interaction matrix can be decomposed to three matrices, where the column of [$\mathbf{U}$ and $\mathbf{V}$ (i.e., $\mathbf{u}_k$ and $\mathbf{v}_k$)] 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 $\mathbf{R}$ with only K-largest singular values. Alternatively, we can learn low-rank representations in a dynamical way through matrix factorization (MF) [11]:

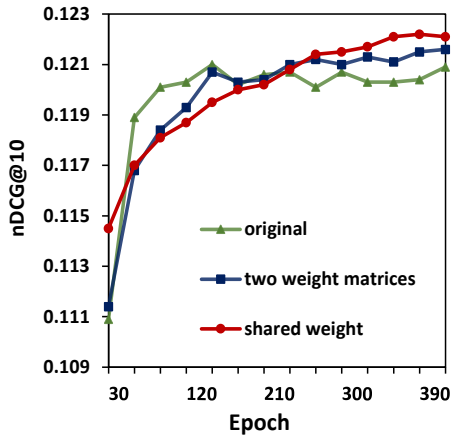
$$\min \sum_{(u,i) \in \mathbf{R}^+} \left\| r_{ui} - \mathbf{e}_u^T \mathbf{e}_i \right\|_2^2 + \lambda \left(\|\mathbf{e}_u\|_2^2 + \|\mathbf{e}_i\|_2^2 \right), \tag{4.7}$$

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).

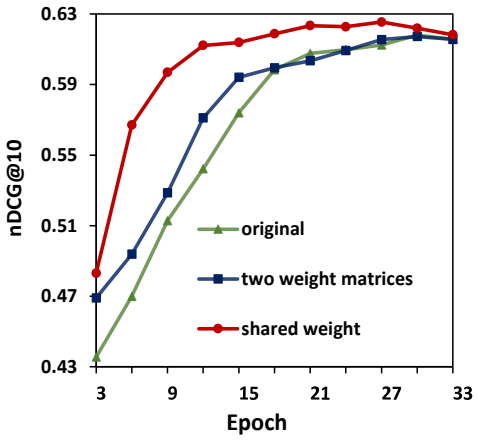
4.3 Methodology

4.3.1 Connections Between GCNs and SVD

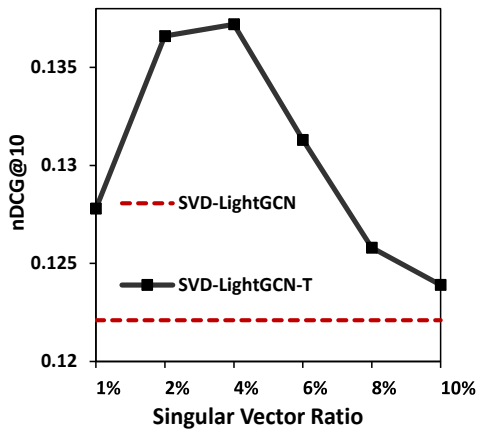
As activation functions and feature transformations have been shown ineffective for CF [41], we focus on LightGCN whose final embeddings are generated as



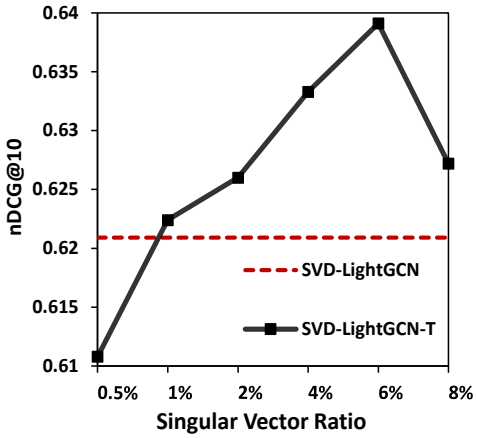
(a) The accuracy of three variants on CiteULike evaluated by nDCG@10.



(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 4.1. Some empirical results on two datasets (CiteULike and ML-100K).

follows:

$$\mathbf{O} = \sum_{l=0}^{L} \frac{\mathbf{H}^{(l)}}{L+1} = \left(\sum_{l=0}^{L} \frac{\tilde{\mathbf{A}}^{l}}{L+1} \right) \mathbf{E}, \tag{4.8}$$

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

$$\tilde{\mathbf{A}}^l = \begin{cases} \begin{bmatrix} \left(\tilde{\mathbf{R}}\tilde{\mathbf{R}}^T\right)^{\frac{l}{2}} & \mathbf{0} \\ \mathbf{0} & \left(\tilde{\mathbf{R}}^T\tilde{\mathbf{R}}\right)^{\frac{l}{2}} \end{bmatrix} & l = \{0, 2, 4, \cdots\} \\ \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{R}}\left(\tilde{\mathbf{R}}^T\tilde{\mathbf{R}}\right)^{\frac{l-1}{2}} \\ \mathbf{R}^T\left(\tilde{\mathbf{R}}\tilde{\mathbf{R}}^T\right)^{\frac{l-1}{2}} & \mathbf{0} \end{bmatrix} & l = \{1, 3, 5, \cdots\}. \end{cases} \tag{4.9}$$

Following the definition of $\tilde{\mathbf{A}}$, $\tilde{\mathbf{R}} = \mathbf{D}_U^{-\frac{1}{2}}\mathbf{R}\mathbf{D}_I^{-\frac{1}{2}}$, where $\mathbf{D}_U$ and $\mathbf{D}_I$ are the node degree matrices for users and items, respectively. Then, we can split Equation (4.8) as follows:

$$\begin{aligned} \mathbf{O}_U &= \frac{\sum_{l=\{0,2,4,\cdots\}}\left(\tilde{\mathbf{R}}\tilde{\mathbf{R}}^T\right)^{\frac{l}{2}}\mathbf{E}_U + \sum_{l=\{1,3,5,\cdots\}}\tilde{\mathbf{R}}\left(\tilde{\mathbf{R}}^T\tilde{\mathbf{R}}\right)^{\frac{l-1}{2}}\mathbf{E}_I}{L+1}, \\ \mathbf{O}_I &= \frac{\sum_{l=\{0,2,4,\cdots\}}\left(\tilde{\mathbf{R}}^T\tilde{\mathbf{R}}\right)^{\frac{l}{2}}\mathbf{E}_I + \sum_{l=\{1,3,5,\cdots\}}\tilde{\mathbf{R}}^T\left(\tilde{\mathbf{R}}\tilde{\mathbf{R}}^T\right)^{\frac{l-1}{2}}\mathbf{E}_U}{L+1}. \end{aligned} \tag{4.10}$$

The first and second terms represent the messages from homogeneous (even-hops) and heterogeneous (odd-hops) neighborhood, $\mathbf{O}_U$ and $\mathbf{O}_I$ are final embeddings for user and items, $\mathbf{E}_U$ and $\mathbf{E}_I$ are embedding matrices for users and items, respectively. Similar to the definition in Section 4.2.2, let $\mathbf{P}$, $\mathbf{Q}$, and σ_k denote the stacked left, right singular vectors, and singular value for $\tilde{\mathbf{R}}$, respectively, and we formulate the following theorem.

Theorem 1. *The adjacency relations in Equation (4.10) can be rewritten as the following forms:*

$$\begin{aligned} \left(\tilde{\mathbf{R}}\tilde{\mathbf{R}}^T\right)^l &= \mathbf{P}diag\left(\sigma_k^{2l}\right)\mathbf{P}^T, \\ \left(\tilde{\mathbf{R}}^T\tilde{\mathbf{R}}\right)^l &= \mathbf{Q}diag\left(\sigma_k^{2l}\right)\mathbf{Q}^T, \end{aligned} \tag{4.11}$$

$$\begin{aligned} \tilde{\mathbf{R}}\left(\tilde{\mathbf{R}}^T\tilde{\mathbf{R}}\right)^{\frac{l-1}{2}} &= \mathbf{P}diag\left(\sigma_k^{l}\right)\mathbf{Q}^T, \\ \mathbf{R}^T\left(\tilde{\mathbf{R}}\tilde{\mathbf{R}}^T\right)^{\frac{l-1}{2}} &= \mathbf{Q}diag\left(\sigma_k^{l}\right)\mathbf{P}^T. \end{aligned} \tag{4.12}$$

Following Theorem 1, we can rewrite Equation (4.10) as:

$$\begin{aligned}\mathbf{O}_U &= \mathbf{P}diag\left(\frac{\sum_{l=\{0,2,\cdots\}}\sigma_k^l}{L+1}\right)\mathbf{P}^T\mathbf{E}_U + \mathbf{P}diag\left(\frac{\sum_{l=\{1,3,\cdots\}}\sigma_k^l}{L+1}\right)\mathbf{Q}^T\mathbf{E}_I,\\ \mathbf{O}_I &= \mathbf{Q}diag\left(\frac{\sum_{l=\{0,2,\cdots\}}\sigma_k^l}{L+1}\right)\mathbf{Q}^T\mathbf{E}_I + \mathbf{Q}diag\left(\frac{\sum_{l=\{1,3,\cdots\}}\sigma_k^l}{L+1}\right)\mathbf{P}^T\mathbf{E}_U.\end{aligned} \tag{4.13}$$

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

$$\mathbf{P}diag\left(\frac{\sum_{l=\{0,2,\cdots\}}\sigma_k^l}{L+1}\right)\mathbf{P}^T = \sum_k \frac{\sum_{l=\{0,2,\cdots\}}\sigma_k^l}{L+1}\mathbf{p}_k\mathbf{p}_k^T. \tag{4.14}$$

$\frac{\sum_{l=\{0,2,\cdots\}}\sigma_k^l}{L+1}$ 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 4.2 (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.*

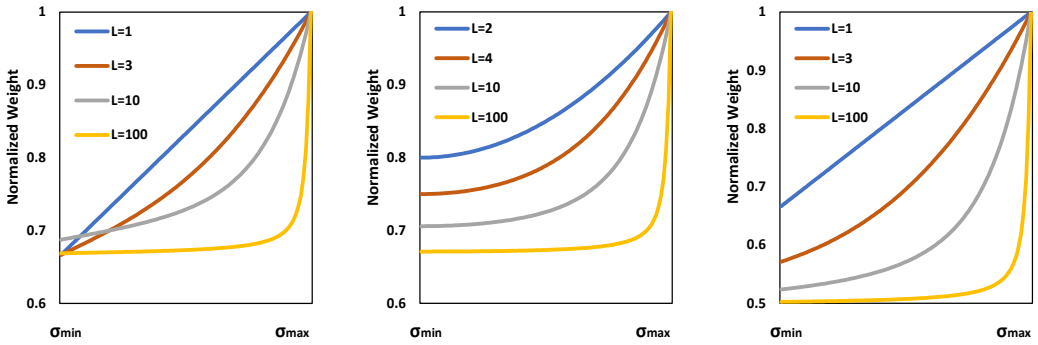
We further observe that:

$$\mathbf{O}_u = \left(\mathbf{p}_{u*}^T \odot \frac{\sum_{l=\{0,2,\cdots\}}\sigma^l}{L+1}\right)\mathbf{P}^T\mathbf{E}_U + \left(\mathbf{p}_{u*}^T \odot \frac{\sum_{l=\{1,3,\cdots\}}\sigma^l}{L+1}\right)\mathbf{Q}^T\mathbf{E}_I, \tag{4.15}$$

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

Assumption 1. $\mathbf{P}^T\mathbf{E}_U$ *and* $\mathbf{Q}^T\mathbf{E}_I$ *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 (4.13); (2) we simply replace $\mathbf{P}^T\mathbf{E}_U$ and $\mathbf{P}^T\mathbf{E}_I$ with two different weight matrices; (3) we use a shared weight matrix based on (2).



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

Figure 4.2. Normalized weights of singular vectors.

The results in Figure 4.1 (a) and (b) show that the performance of the three models are fairly close, and thus: (1) neighborhood 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 (4.13) and simplify it as:

$$\begin{aligned} \mathbf{O}_U &= \mathbf{P} diag\left(\frac{\sum_{l=0}^{L}\sigma_k^l}{L+1}\right)\mathbf{W}, \\ \mathbf{O}_I &= \mathbf{Q} diag\left(\frac{\sum_{l=0}^{L}\sigma_k^l}{L+1}\right)\mathbf{W}, \end{aligned} \tag{4.16}$$

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., $\mathbf{p}_k$ and $\mathbf{q}_k$); then, we learn a condensed embeddings of the singular vectors through a feature transformation $\mathbf{W}$. Figure 4.2 (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* $\mathbf{W} = \mathbf{I}$ *and* $l = L = \frac{1}{2}$ *(fixed to a square root).*

4.3.2 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 4.1 (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 [42].

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

As shown from Figure 4.2 (b), the larger singular values are further emphasized as increasing the model layers. Following Theorem 2, if we further increase the layer L:

$$\lim_{L\to\infty} = \frac{\sum_{l=0}^{L} \frac{\sigma_k^l}{L+1}}{\sum_{l=0}^{L} \frac{\sigma_{\max}^l}{L+1}} \to 0, \tag{4.17}$$

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: $\dot{\mathbf{R}} = (\mathbf{D}_U + \alpha\mathbf{I})^{-\frac{1}{2}}\mathbf{R}(\mathbf{D}_I + \alpha\mathbf{I})^{-\frac{1}{2}}$ where $\alpha \geq 0$.

Theorem 3. *Given the singular value* $\dot{\sigma}_k$ *of* $\dot{\mathbf{R}}$, $\dot{\sigma}_{\max} \leq \frac{d_{\max}}{d_{\max}+\alpha}$ *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 (4.8) as:

$$\mathbf{O} = \left(\sum_{l=0}^{L} \alpha_l \tilde{\mathbf{A}}^l \right) \mathbf{E}, \tag{4.18}$$

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.

4.3.3 SVD-GCN

Based on the analysis in Section 4.3.2, we formulate the user and item representations as follows:

$$\begin{aligned} \mathbf{O}_U &= \dot{\mathbf{P}}^{(K)} diag\left(\psi\left(\dot{\sigma_k}\right)\right) \mathbf{W}, \\ \mathbf{O}_I &= \dot{\mathbf{Q}}^{(K)} diag\left(\psi\left(\dot{\sigma_k}\right)\right) \mathbf{W}, \end{aligned} \tag{4.19}$$

where $\dot{\mathbf{P}}^{(K)}$ and $\dot{\mathbf{Q}}^{(K)}$ are composed of K-largest left and right singular vectors of $\dot{\mathbf{R}}$, 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 4.3.2, extensive experiments show that an exponential kernel [137] 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 4.4.3. 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:

$$\begin{aligned} \mathbf{o}_u &= \dot{\mathbf{p}}_u^T \odot \left(e^{\beta\dot{\sigma}}\right) \mathbf{W}, \\ \mathbf{o}_i &= \dot{\mathbf{q}}_i^T \odot \left(e^{\beta\dot{\sigma}}\right) \mathbf{W}, \end{aligned} \tag{4.20}$$

where $\dot{\mathbf{p}}_u^T$ and $\dot{\mathbf{q}}_i^T$ are the rows of $\dot{\mathbf{P}}^{(K)}$ and $\dot{\mathbf{Q}}^{(K)}$, respectively; $\dot{\sigma}$ is a vector containing all singular values. Note that the element-wise multiplication does not involve parameters thus can be precomputed. Then, inspired by BPR loss [45], we formulate the loss function as follows:

$$\mathcal{L}_{main} = \sum_{u \in \mathcal{U}} \sum_{(u,i^+) \in \mathbf{R}^+, (u,i^-) \notin \mathbf{R}^+} \ln \sigma \left(\mathbf{o}_u^T \mathbf{o}_{i^+} - \mathbf{o}_u^T \mathbf{o}_{i^-}\right). \tag{4.21}$$

As shown in Equation (4.10), 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 $\mathbf{R}_U = \dot{\mathbf{R}}^T\dot{\mathbf{R}}$ and $\mathbf{R}_I = \dot{\mathbf{R}}\dot{\mathbf{R}}^T$, respectively. According to Equation (4.26) in Section 4.5, the eigenvectors of $\mathbf{R}_U$ and $\mathbf{R}_I$ are actually $\dot{\mathbf{R}}$'s left and right singular vectors, respectively; and the eigenvalues are both the square of $\dot{\mathbf{R}}$'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 $\mathcal{G}_U$:

$$\mathcal{L}_{user} = \sum_{u \in \mathcal{U}} \sum_{(u,u^+) \in \mathcal{E}_U, (u,u^-) \notin \mathcal{E}_U} \ln \sigma \left(\mathbf{o}_u^T \mathbf{o}_{u^+} - \mathbf{o}_u^T \mathbf{o}_{u^-}\right). \tag{4.22}$$

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

$$\mathcal{L}_{item} = \sum_{i \in \mathcal{I}} \sum_{(i,i^+) \in \mathcal{E}_I, (i,i^-) \notin \mathcal{E}_I} \ln \sigma \left(\mathbf{o}_i^T \mathbf{o}_{i^+} - \mathbf{o}_i^T \mathbf{o}_{i^-} \right). \tag{4.23}$$

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

$$\begin{aligned}
&\text{SVD-GCN-B}: \ \mathcal{L} = \mathcal{L}_{\text{main}} + \lambda \left\| \Theta \right\|_2^2, \\
&\text{SVD-GCN-U}: \ \mathcal{L} = \mathcal{L}_{\text{main}} + \gamma \mathcal{L}_{\text{user}} + \lambda \left\| \Theta \right\|_2^2, \\
&\text{SVD-GCN-I}: \ \mathcal{L} = \mathcal{L}_{\text{main}} + \zeta \mathcal{L}_{\text{item}} + \lambda \left\| \Theta \right\|_2^2, \\
&\text{SVD-GCN-M}: \ \mathcal{L} = \mathcal{L}_{\text{main}} + \gamma \mathcal{L}_{\text{user}} + \zeta \mathcal{L}_{\text{item}} + \lambda \left\| \Theta \right\|_2^2,
\end{aligned} \tag{4.24}$$

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 $\mathbf{W}$.

4.3.4 Discussion

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 [138], with a complexity as: $\mathcal{O}(K \left|\mathbf{R}^+\right| + K^2 \left|\mathcal{U}\right| + K^2 \left|\mathcal{I}\right|)$. 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 $\mathcal{O}(c \left|\mathbf{R}^+\right| (K+1)d)$, which is comparable to MF: $c \left|\mathbf{R}^+\right| d$, 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.

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 [135] simplifies LightGCN by replacing neighborhood aggregation with a weighted MF and shows lower complexity:

$$\max \sum_{u \in \mathcal{U}, i \in \mathcal{N}_u} \beta_{u,i} \mathbf{e}_u^T \mathbf{e}_i, \tag{4.25}$$

Table 4.1. 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

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.

4.4 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.

4.4.1 Experimental Settings

Datasets and Evaluation Metrics

We use five public datasets in this work, where the results of Figure 4.1 are based on CiteULike* and ML-100K [139]. To demonstrate the effectiveness of our proposed methods on more datasets and to justify the previous analysis, we

*https://github.com/js05212/citeulike-a

evaluate SVD-GCN on three other datasets: Gowalla [30], Yelp [140], and ML-1M [139]. Since we focus on implicit feedback, we only keep user/item ID and transform feedbacks to binary ratings. Table 4.1 lists statistics of datasets.

We adopt two widely-used metrics: Recall and nDCG [128] 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.

Baselines

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

- BPR [45]: This is a stable and classic MF-based method, exploiting a Bayesian personalized ranking loss for personalized rankings.
- EASE [129]: This is a neighborhood-based method with a closed form solution and show superior performance to many traditional CF methods.
- LightGCN [41]: 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 [91]: 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 [38]: 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 [41]. We set $\tau = 0.2$, $\lambda_1 = 0.1$, $p = 0.1$, and use a three-layer architecture.

- UltraGCN [135]: 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 [93], NGCF [30], and SpectralCF [82] as aforementioned baselines have already shown superiority over them.

Implementation Details

We implemented the proposed model based on PyTorch[†] and released the code on Github[‡]. 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 [131]; the batch size is set to 256. We report other hyperparameter settings in the next subsection.

4.4.2 Comparison

Performance

We report the accuracy of baselines and our proposed GCN-SVD variants in Table 4.2, 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.

[†] `https://pytorch.org/`

[‡] `https://github.com/tanatosuu/svd_gcn`

Table 4.2. Overall performance comparison. Improv.% denotes the improvements over the best baselines.

Datasets	Methods	nDCG@10	nDCG@20	Recall@10	Recall@20
Yelp	BPR	0.0388	0.0374	0.0371	0.0370
	Ease	0.0360	0.0362	0.0346	0.0368
	LCFN	0.0617	0.0627	0.0613	0.0653
	UltraGCN	0.0417	0.0403	0.0404	0.0403
	LightGCN	0.0751	0.0710	0.0725	0.0698
	SGL-ED	0.0817	0.0794	0.0784	0.0792
	SVD-GCN-S	0.0919	0.0895	0.0894	0.0903
	SVD-GCN-B	0.0898	0.0876	0.0866	0.0879
	SVD-GCN-U	0.0923	0.0897	0.0888	0.0898
	SVD-GCN-I	0.0930	0.0907	0.0897	0.0910
	SVD-GCN-M	**0.0941**	**0.0917**	**0.0908**	**0.0921**
	Improvement%	+15.18	+15.49	+15.82	+16.29
ML-1M	BPR	0.5521	0.4849	0.5491	0.4578
	Ease	0.3773	0.3249	0.3682	0.3000
	LCFN	0.5927	0.5197	0.5887	0.4898
	UltraGCN	0.5326	0.4688	0.5302	0.4434
	LightGCN	0.5917	0.5261	0.5941	0.5031
	SGL-ED	0.6029	0.5314	0.6010	0.5035
	SVD-GCN-S	0.6458	0.5702	0.6466	0.5421
	SVD-GCN-B	0.6480	0.5724	0.6484	0.5443
	SVD-GCN-U	0.6571	**0.5791**	**0.6571**	**0.5495**
	SVD-GCN-I	**0.6574**	0.5770	0.6565	0.5465
	SVD-GCN-M	0.6521	0.6705	0.6490	0.5377
	Improvement%	+9.04	+8.98	+9.33	+9.14
Gowalla	BPR	0.1086	0.0907	0.0917	0.0743
	Ease	0.0722	0.0670	0.0680	0.0642
	LCFN	0.1305	0.1132	0.1144	0.0980
	UltraGCN	0.0977	0.0815	0.0841	0.0681
	LightGCN	0.1477	0.1327	0.1368	0.1224
	SGL-ED	0.1789	0.1561	0.1563	0.1353
	SVD-GCN-S	0.1900	0.1677	0.1690	0.1484
	SVD-GCN-B	0.1820	0.1607	0.1628	0.1428
	SVD-GCN-U	0.1875	0.1654	0.1667	0.1460
	SVD-GCN-I	0.1857	0.1646	0.1662	0.1466
	SVD-GCN-M	**0.1905**	**0.1681**	**0.1693**	**0.1487**
	Improvement%	+6.48	+7.69	+8.32	+9.90

Table 4.3. 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

- 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.

Training Efficiency

The results shown in this subsection are obtained on a machine equipped with AMD Ryzen 9 5950X and GeForce RTX 3090. Figure 4.3 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 4.3 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.

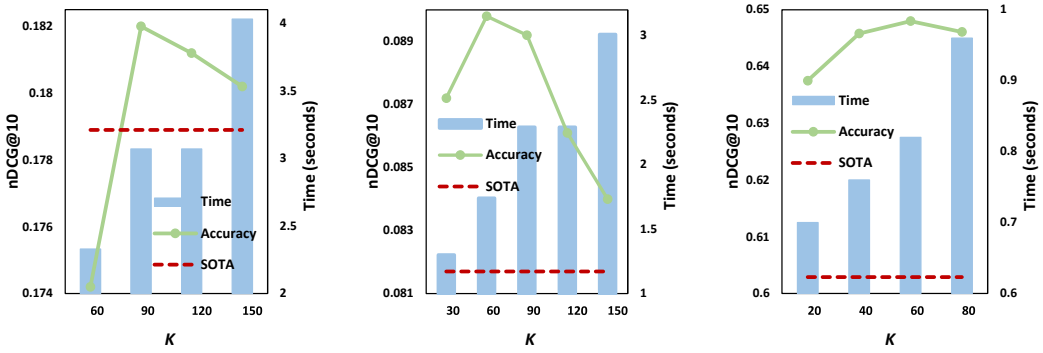
4.4.3 Model Analysis

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 4.4 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).

Do We Need Feature Transformation?

By comparing SVD-GCN-S and SVD-GCN-B, we can see $\mathbf{W}$ 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 $\mathbf{W}$ 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



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

Figure 4.3. How the preprocessing time and accuracy (nDCG@10) vary on K on SVD-GCN-B.

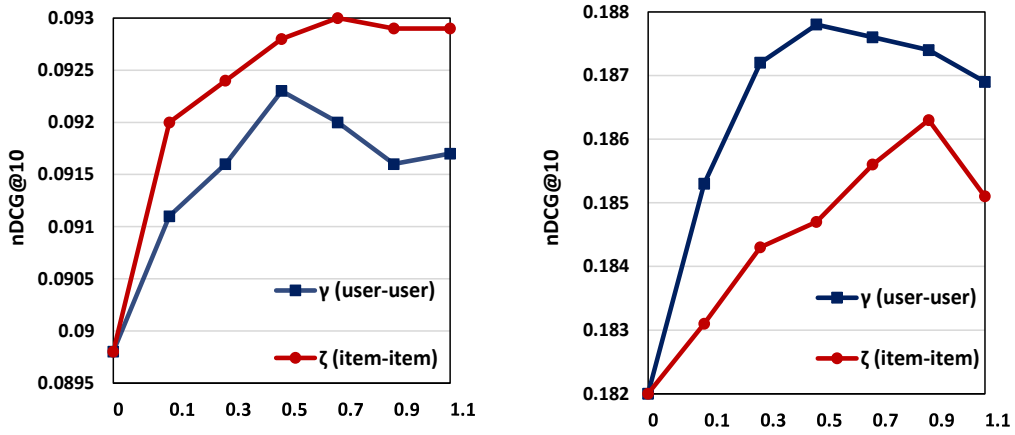
relations are much denser thus is easier to learn. Overall, SVD-GCN can achieve superior accuracy without any model training, implying that the key design making GCN effective for recommendation lies in a good low-rank representation.

Effect of Renormalization Trick

We have two observations from Figure 4.5 (a): as increasing α (i.e., shrinking the singular value gap), (1) the accuracy increases first then drops, reaches the best at $\alpha = 3$; (2) the model tends to require fewer singular vectors. In Figure 4.5 (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.

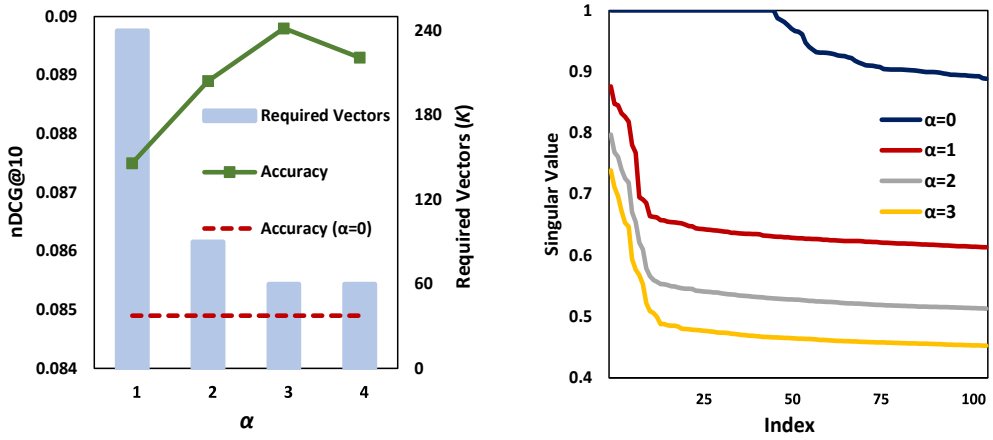
Effect of β

Figure 4.6 shows the accuracy with varying β. The accuracy first increases as increasing β, then starts dropping after reaching the best performance at $\beta = 2.5$



(a) Yelp. (b) Gowalla.

Figure 4.4. Effect of γ and ζ.

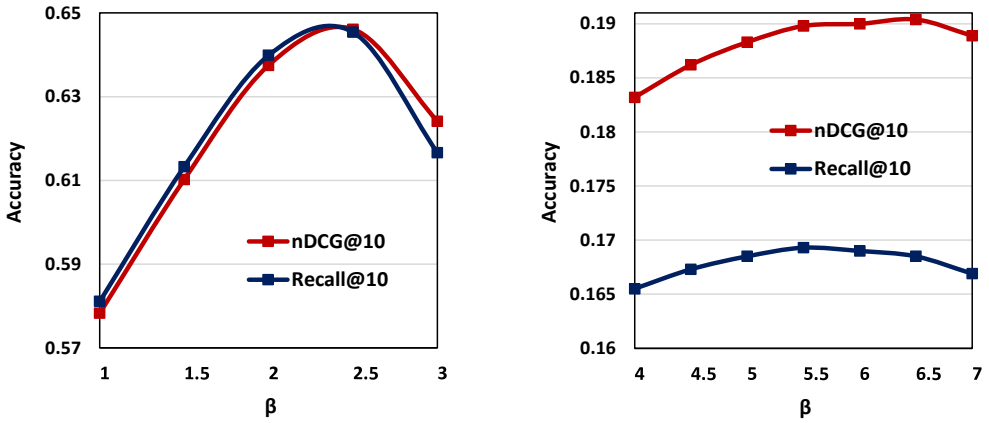


(a) How K and accuracy (nDCG@10) vary on α on SVD-GCN-B.

(b) Distribution of top 100 singular values with varying α.

Figure 4.5. Effect of renormalization trick on Yelp.

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.



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

Figure 4.6. Effect of β on SVD-GCN-S.

The Choice of Weighting Function

We show the accuracy of SVD-GCN-S with different weighting functions in Table 4.4. 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 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)$\gg$(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)$\gg$(2)$>$(3), and the functions satisfying all three properties perform the best.

Table 4.4. Accuracy of different weighting functions on Yelp.

Design	Function	nDCG@10	Property		
			(1) Increasing	(2) Pos Coef.	(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			

4.5 Proofs

4.5.1 Proofs of Theorem 1

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

$$\begin{aligned} \tilde{\mathbf{R}}\tilde{\mathbf{R}}^T &= \mathbf{P} diag\left(\sigma_k^2\right) \mathbf{P}^T, \\ \tilde{\mathbf{R}}^T\tilde{\mathbf{R}} &= \mathbf{Q} diag\left(\sigma_k^2\right) \mathbf{Q}^T. \end{aligned} \tag{4.26}$$

By repeating the above Equations l times, we obtain Equation (4.11).

For simplicity, we let $\mathbf{R}' = \tilde{\mathbf{R}}\left(\tilde{\mathbf{R}}^T\tilde{\mathbf{R}}\right)^{\frac{l-1}{2}}$, and $\mathbf{R}'^T = \mathbf{R}^T\left(\tilde{\mathbf{R}}\tilde{\mathbf{R}}^T\right)^{\frac{l-1}{2}}$. We let $\mathbf{P}'$, $\mathbf{Q}'$ and σ_k' denote the stacked left singular vectors, right singular vectors and singular value for $\mathbf{R}'$, respectively. Following Equation (4.26), we can derive the following equations:

$$\begin{aligned} \mathbf{R}'\mathbf{R}'^T &= \left(\tilde{\mathbf{R}}\tilde{\mathbf{R}}^T\right)^l = \mathbf{P} diag\left(\sigma_k^{2l}\right) \mathbf{P}^T = \mathbf{P}' diag\left(\sigma_k'^2\right) \mathbf{P}'^T, \\ \mathbf{R}'^T\mathbf{R}' &= \left(\tilde{\mathbf{R}}^T\tilde{\mathbf{R}}\right)^l = \mathbf{Q} diag\left(\sigma_k^{2l}\right) \mathbf{Q}^T = \mathbf{Q}' diag\left(\sigma_k'^2\right) \mathbf{Q}'^T. \end{aligned} \tag{4.27}$$

It is easy to observe that $\mathbf{P}' = \mathbf{P}$, $\mathbf{Q}' = \mathbf{Q}$ and $\sigma_k' = \sigma_k^l$. Then, according to SVD, we derive Equation (4.12).

□

4.5.2 Proofs of Theorem 2 and 3

Proof. We first introduce Rayleigh quotients [141]:

$$\lambda_{\min} \leq \mathbf{x}^T\tilde{\mathbf{A}}\mathbf{x} \leq \lambda_{\max} \quad s.t.\, |\mathbf{x}| = 1, \tag{4.28}$$

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

$$1 - \mathbf{x}^T \tilde{\mathbf{A}} \mathbf{x} = \mathbf{x}^T \mathbf{x} - \mathbf{x}^T \tilde{\mathbf{A}} \mathbf{x} = \sum_{(u,i) \in \mathcal{E}} \left(\frac{x_u}{\sqrt{d_u}} - \frac{x_i}{\sqrt{d_i}} \right)^2 \geq 0. \tag{4.29}$$

In the meanwhile, we have the following observation:

$$\tilde{\mathbf{A}} \begin{bmatrix} \mathbf{p}_k \\ \mathbf{q}_k \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{R}} \mathbf{q}_k \\ \tilde{\mathbf{R}}^T \mathbf{p}_k \end{bmatrix} = \sigma_k \begin{bmatrix} \mathbf{p}_k \\ \mathbf{q}_k \end{bmatrix}, \tag{4.30}$$

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

Given $\dot{\mathbf{R}}$, we can define the corresponding adjacency matrix $\dot{\mathbf{A}}$. Since the relation in Equation (4.30) still holds between $\dot{\mathbf{R}}$ and $\dot{\mathbf{A}}$, we only need to prove $\dot{\lambda}_{\max} \leq \frac{d_{\max}}{d_{\max}+\alpha}$.

$$\begin{aligned} \mathbf{x}^T \dot{\mathbf{A}} \mathbf{x} &= \sum_{(u,i) \in \mathcal{E}} \frac{2 x_u x_i}{\sqrt{d_u + \alpha}\sqrt{d_i + \alpha}} \leq \sum_{u \in \mathcal{V}} \frac{d_u}{d_u + \alpha} x_u^2, \\ &= 1 - \sum_{u \in \mathcal{V}} \frac{\alpha}{d_u + \alpha} x_u^2 \leq 1 - \frac{\alpha}{d_{\max} + \alpha} = \frac{d_{\max}}{d_{\max} + \alpha}. \end{aligned} \tag{4.31}$$

$=$ holds when $\alpha = 0$. When $\alpha > 0$, $\dot{\lambda}_{\max} < \frac{d_{\max}}{d_{\max}+\alpha}$, since $\mathbf{x}$ 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 $1 - \sum_{u\in\mathcal{V}} \frac{\alpha}{d_u+\alpha} x_u^2 = 1 - \frac{\alpha}{d_{\max}+\alpha}$.

□

4.6 Summary

In this chapter, 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 4.4.3, 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.

CHAPTER 5

Removing Distribution Redundancy in Graph Recommendation

In Chapter 3 and 4, we showed that only a small fraction of spectral features are contributive to recommendation accuracy. In this chapter [142], we show that the number of required spectral features is closely related to the spectral distribution where important information tends to be concentrated in more (fewer) spectral features on a flatter (sharper) distribution, making the complexity unpredictable. To reduce the complexity for retrieving spectral features, we propose a renormalized adjacency matrix with a hyper-parameter adjusting the sharpness of the spectral distribution. We further propose a scalable contrastive learning framework to alleviate data sparsity and to boost model generalization, leading to significant improvement. Extensive experiments on three real-world datasets demonstrate the effectiveness and efficiency of our proposed designs.

5.1 Introduction

Graph convolutional networks (GCNs) [25] have shown great potentials in recommender systems and collaborative filtering (CF) [93, 30]. Due to the limited

attention that has been paid to demystify GCN-based recommendation algorithms as well as the expensive computational complexity and poor scalability of existing GCN-based methods, we presented two solutions: GDE and SVD-GCN showing lower complexity and higher efficiency in Chpter 3 and 4. Most importantly, we showed that only a small fraction of spectral features are contributive to recommendation while most features are noisy and useless. However, we notice that the number of required spectral features actually varies on datasets after conducting extensive experiments on different datasets, making the complexity of our proposed algorithms unpredictable since calculating spectral features requires additional complexity.

To tackle this limitation, we show that the number of required spectral features is related to the sharpness of the spectral distribution, that fewer (more) features are required on a sharper (flatter) spectrum. Inspired by this observation, we propose a renormalized adjacency matrix with a hyper-parameter adjusting the sharpness of spectrum to reduce the number of required features. Furthermore, By analyzing how graph contrastive learning (GCL) works for recommendation, we propose a scalable contrastive learning framework to boost model generalization and robustness and to augment sparse supervisory signal with rich higher-order neighborhood signals. Finally, we comprehensively evaluate the proposed designs on three datasets with respect to efficiency and effectiveness. The main contributions of this work can be summarized as follows:

- To reduce the computational cost for retrieving spectral features, we concentrate important information from interactions on fewer features by increasing node smoothness to sharpen the spectral distribution, resulting in significant improvement as well.
- We propose a scalable contrastive learning framework by performing augmentation on spectral features and augmenting sparse supervisory signals with abundant higher-order neighborhood signals, resulting in significant improvement.
- Extensive experiments on three real-world datasets demonstrate the effectiveness and efficiency of our proposed designs.

5.2 Preliminaries

The notations used in this chapter is consistent with Chapter 3 and 4. We use SVD-GCN proposed in Chapter 4 as the baseline model:

$$\begin{aligned} \mathbf{O}_U &= \mathbf{P}^{(K)} diag\left(\Delta\left(\sigma_k\right)\right) \mathbf{W}, \\ \mathbf{O}_I &= \mathbf{Q}^{(K)} diag\left(\Delta\left(\sigma_k\right)\right) \mathbf{W}, \end{aligned} \tag{5.1}$$

with the following three components:

- Stacked top-K smoothest left and right singular vectors of $\hat{\mathbf{R}}$: $\mathbf{P}^{(K)} \in \mathbb{R}^{|\mathcal{U}| \times K}$ and $\mathbf{Q}^{(K)} \in \mathbb{R}^{|\mathcal{I}| \times K}$.
- GCNs use a polynomial to weight the spectral features. Here, we abstract it as a nonparametric function $\Delta(\cdot)$, since the dynamic choice has been shown ineffective [96].
- A feature transformation $\mathbf{W} \in \mathbb{R}^{K \times d}$, where $K \ll \min(|\mathcal{U}|, |\mathcal{I}|)$.

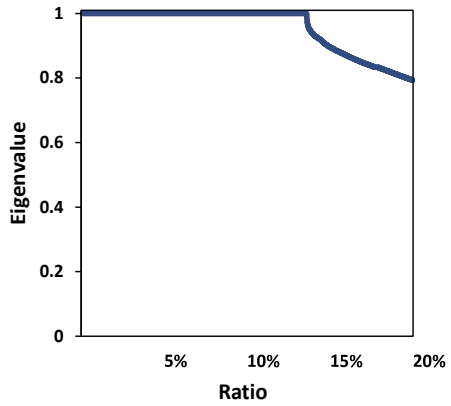
5.3 Methodology

5.3.1 Removing Distribution Redundancy

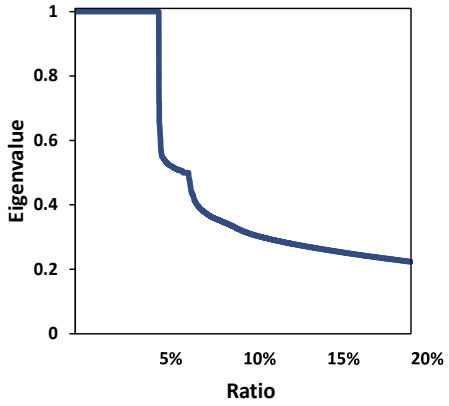
Figure 5.1 (a) and (b) illustrate the spectral distributions of CiteULike and ML-1M, on which our previously proposed GDE [96] reaches the best performance with top 30% and 5% smoothed features, respectively. On CiteULike, over 10% components correspond to the largest eigenvalue, and the spectral value drops slowly; while the spectral distribution on ML-1M is sharper and the spectral value drops more quickly. Recall our previous analysis that the feature with a larger spectral value (*i.e.,* smoother feature) tends to be more important to recommendation. Thus, we can make the following observation:

Observation 3. *A flatter (sharper) spectral distribution (i.e., the spectral value drops more slowly (quickly)) implies the graph is composed of more (fewer) smoothed components and thus requires more (fewer) features.*

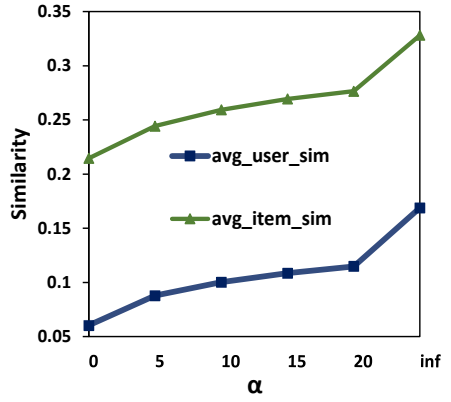
Apparently, it requires more computational cost for retrieving spectral features on a flatter spectral distribution, while we hope the important information can



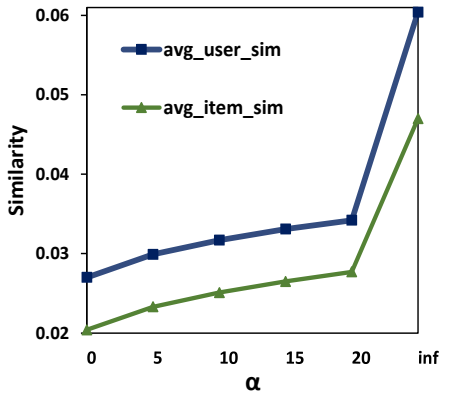
(a) Top 20% eigenvalue distribution on CiteULike.



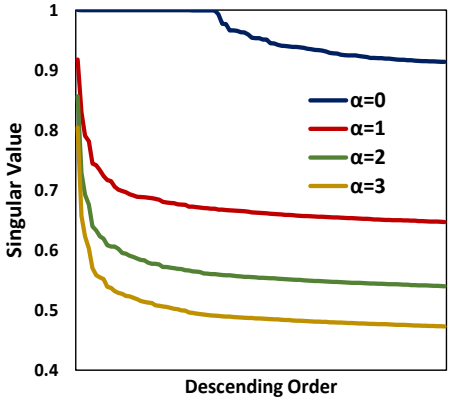
(b) Top 20% eigenvalue distribution on ML-1M.



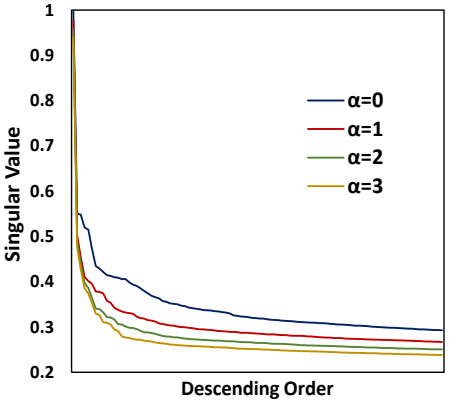
(c) The average similarity increases as α becomes larger on CiteULike.



(d) The average similarity increases as α becomes larger on ML-1M..



(e) Top 100 singular values with varying α on CiteULike.



(f) Top 100 singular values with varying α on ML-1M..

Figure 5.1. Experimental results for tackling distribution redundancy.

be concentrated in as few features as possible to reduce the cost. Then, we raise a simple question: **Can we reduce the required features K?** This question is equivalent to: How can we sharpen the distribution? Without considering the specific recommendation algorithm, a user or an item can be represented as the corresponding row of the adjacency matrix (*e.g.,* $\hat{\mathbf{A}}_{u*}$ for u and $\hat{\mathbf{A}}_{i*}$ for i), and the similarity can be simply measured as their dot product or cosine similarity. And we can see that only homogeneous nodes (*i.e.,* user-user and item-item) connected to the common nodes (*e.g.,* the users that interacted with common items, and the items that are interacted by common users) have similarities. Consider the average similarity between a node u and other nodes:

$$\begin{aligned}\hat{\mathbf{A}}_{u*}\sum_{z\in\mathcal{N}_u^2}\hat{\mathbf{A}}_{z*}^T &= (\mathbf{V}_{u*}\odot\lambda)\,\mathbf{V}^T\left(\left(\sum_z \mathbf{V}_{z*}\odot\lambda\right)\mathbf{V}^T\right)^T,\\ &= (\mathbf{V}_{u*}\odot\lambda)\left(\sum_z \mathbf{V}_{z*}\odot\lambda\right)^T,\end{aligned} \tag{5.2}$$

where $\mathcal{N}_u^2$ is the node set of second-order neighbor who have similarities with u, λ is a vector containing all eigenvalues.

Definition 3. *(Variation on Second-order Graphs). The variation of the eigenvectors on the second-order graph can be defined as:*

$$\left\|\mathbf{v}_k - \hat{\mathbf{A}}^2\mathbf{v}_k\right\| = 1-\lambda_k^2 \in [0,1] \tag{5.3}$$

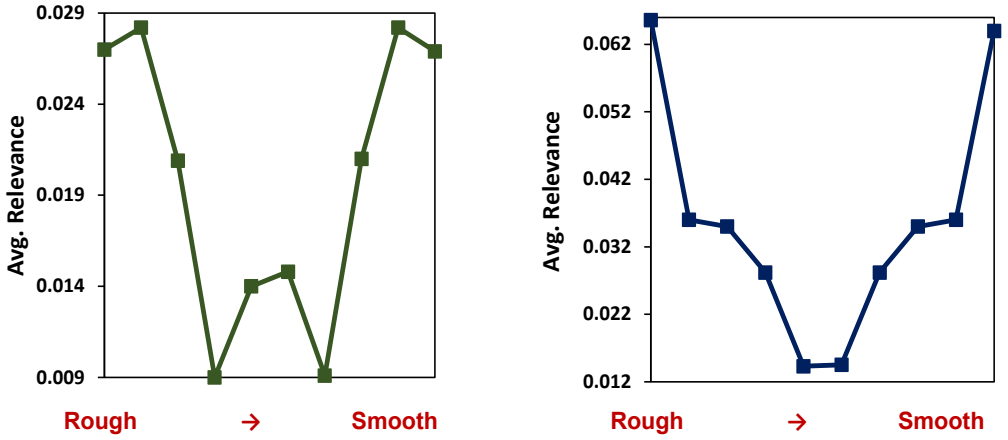
Interpretation of Equation (5.2). Definition 3 measures the difference between the signal samples of eigenvectors at each node ($\mathbf{V}_{uk}$) and at its second-order neighbor ($\sum_z \mathbf{V}_{zk}$). Intuitively, $\mathbf{v}_k$ with $|\lambda_k| \to 1$ implies that the nodes are similar to their second-order neighborhood: $|\mathbf{V}_{uk} - \sum_z \mathbf{V}_{zk}| \to 0$, while $\mathbf{v}_k$ with $|\lambda_k| \to 0$ emphasizes the difference between $\mathbf{V}_{uk}$ and $\sum_z \mathbf{V}_{zk}$. Consider λ as a band-pass filter, with the spectral distribution becoming sharper, the components with $|\lambda_k| \to 1$ and $|\lambda_k| \to 0$ are emphasized and suppressed, respectively, leading to a higher similarity between the nodes and their second-order neighbor who have non-zero similarities with them. In other words, the sharpness of the spectral distribution is closely related to the average node similarity defined on the normalized adjacency matrix. On the other hand, the obvious difference

between ML-1M and CiteULike is the data density, where the users/items of ML-1M have more interactions that are easier to have similarities with other the users/items, thus resulting in a sharper spectral distribution. Then, the original question can be transformed to: **How do we increase the average node similarity (defined on the interaction matrix) to sharpen the spectral distribution?**

Without changing the interactions, the key to increase the average similarity lies in the weight of the adjacency relation: $\hat{\mathbf{A}}_{ui} = \frac{1}{\sqrt{d_u}\sqrt{d_i}}$. To this end, we define a renormalized adjacency matrix with $\bar{\mathbf{A}}_{ui} = w(d_u)w(d_i)$ and investigate what node weights lead to a higher average similarity. Intuitively, the average similarity between users and items are defined as follows:

$$\begin{aligned} ave\_sim\_user &= \frac{\sum_{u,v\in\mathcal{U}} \hat{\mathbf{A}}_{u*}\hat{\mathbf{A}}_{v*}^T}{|\mathcal{U}|^2} = \sum_{i\in\mathcal{I}} 2w(i)^2 \frac{\sum_{u,v\in\mathcal{N}_i} w(u)w(v)}{|\mathcal{U}|^2}, \\ ave\_sim\_item &= \frac{\sum_{i,j\in\mathcal{I}} \hat{\mathbf{A}}_{*i}^T\hat{\mathbf{A}}_{*j}}{|\mathcal{I}|^2} = \sum_{u\in\mathcal{U}} 2w(u)^2 \frac{\sum_{i,j\in\mathcal{N}_u} w(i)w(j)}{|\mathcal{I}|^2}. \end{aligned} \tag{5.4}$$

Without loss of generality, we assume that user preference is not related to the node degree, or such a relation is uniform across the board. For instance, if we consider the popularity bias, we assume that all users are uniformly have the tendency to interact with popular items. In other words, $w(u)w(v)$ or $w(i)w(j)$ follow the same distribution that can be considered fixed. Then, we can see the item with a higher degree affects more to the average similarity (proportional to d_i^2) when we consider user similarity. Similarly, the user with a higher degree affects more to the average similarity when we consider the item similarity, leading to the conclusion that the higher weights over the high-degree nodes results in higher node similarity. Theoretically, $w(z)$ with a higher weight over high-degree nodes than the original setting $w(z) = \frac{1}{\sqrt{z}}$ results in higher average similarity. In this work, we set $\bar{\mathbf{A}}_{ui} = \frac{1}{\sqrt{d_u+\alpha}\sqrt{d_i+\alpha}}$, where $\alpha \in \mathbb{R}^+$. To avoid introducing too many different notations, we still use α here for simplicity. The range of α here is different from the α in $w(z) = \frac{1}{z^\alpha}$. Note that since $\left(\frac{1}{\sqrt{z}}\right)'$ is monotonically increasing, the weights of $\frac{1}{\sqrt{z+\alpha}}$ over high-degree nodes are higher than that of $\frac{1}{\sqrt{z}}$, and are emphasized more by setting α larger. Figure 5.1 (c) and (d) show that the average user and item similarities constantly increase as increasing α; in Figure 5.1 (e) and (f), we can observe a sharper distribution as increasing α, thus the number of required features (*i.e.,* K) is expected to be reduced. Note that



(a) on CiteULike. (b) On ML-1M

Figure 5.2. Arranging spectral features in rough → smooth order, we evenly divide them into 10 groups and calculate the average relevance (taking the absolute value of cosine similarity) between the eigenvectors of the original and the perturbed graphs (we randomly drop the edge with probability 0.1). The smaller (larger) relevance implies that the noise is more intense (weaker) on the group of features.

there is a trade-off between the reduction of K and the integrity of interactions: a too large α would sabotage the original interactions.

By applying the renormalized adjacency matrix to SVD-GCN, we name it simplified graph denoising encoder (SGDE) and optimize it with BPR loss [45]:

$$\mathcal{L}_{main} = \sum_{u \in \mathcal{U}} \sum_{(u,i^+) \in \mathbf{R}^+, (u,i^-) \notin \mathbf{R}^+} \ln \sigma \left(\mathbf{o}_u^T \mathbf{o}_{i^+} - \mathbf{o}_u^T \mathbf{o}_{i^-} \right). \tag{5.5}$$

5.3.2 Contrastive Simplified Graph Denoising Encoder (CSGDE)

Recently, graph contrastive learning (GCL) has received much attention including recommender systems [38, 143]. The core idea is to maximize and minimize the agreement of two views from the same and different node(s), respectively, where the views are generated by perturbing the original graph such as randomly dropping out edges or nodes. However, existing GCL learning paradigm is computationally expensive as generating multiple node views basically requires multiple

times the complexity of GCNs. In this section, we aim to incorporate GCL into our method without bringing too much complexity. To this end, we first attempt to analyze how GCL works for recommendation. We focus on edge dropping as it gains more improvement than other augmentations [38]. As shown in Figure 5.2, the edge-dropping noise tends to attack noisy components in the middle area while the smoothed and rough components tend to be preserved. By maximizing the agreements between embeddings from perturbed graphs, the dissimilar components (*i.e.,* noisy features) tend to be filtered out. However, there is no guarantee that the edge-dropping noise always attack the noisy spectral features as we can see the spectral features in the middle area on CiteULike shows higher relevance than ML-1M. To summarize, the limitations of GCL are that: (1) The noise added on the graph is uncontrollable, (2) computationally expensive; (3) GCL ignores the latent relations that indirectly connected users/items might be as well closely related to the target user/item, as it only maximizes the views from the same node. To this end, we propose a feature augmentation by adding random noise on the weighted spectral features:

$$\begin{aligned} \mathbf{O}_U &= \mathcal{N}\left(\mathbf{P}^{(K)} diag\left(\Delta\left(\sigma_k\right)\right), \mu\right)\mathbf{W}, \\ \mathbf{O}_I &= \mathcal{N}\left(\mathbf{Q}^{(K)} diag\left(\Delta\left(\sigma_k\right)\right), \mu\right)\mathbf{W}, \end{aligned} \tag{5.6}$$

where the noise is generated from normal distribution $\mathcal{N}(0, \mu)$ with μ as the standard deviation. Since $\Delta(\cdot)$ outputs the feature weight according to their importance to recommendation, the more (less) important features have larger (smaller) weights thus are less (more) perturbed by the noise. Then, Equation (5.6) emphasizes the important features and tends to filter out the noisy ones. Instead of only maximizing the views from the same node, we incorporate higher-order neighbor signals:

$$\mathcal{L}_{user} = \sum_{u \in \mathcal{U}} \sum_{(u,u^+) \in, (u,u^-) \notin \mathcal{E}^L_{\mathcal{A}_U}} \ln \sigma\left(\mathbf{o}_u^T \mathbf{o}_{u^+} - \mathbf{o}_u^T \mathbf{o}_{u^-}\right), \tag{5.7}$$

where $\mathcal{E}^L_{\mathcal{A}_U}$ is the edge set considering $\{1, \cdots, L\}$ hop neighbors of $\mathbf{A}_U$. Although users are not directly connected, they still might show similar interests and should be close on the embedding space if they are close on the graph. Since the InfoNCE loss brings additional complexity, we stick to the BPR loss here. Similarly, the contrastive loss for higher-order item signals are generated as:

$$\mathcal{L}_{item} = \sum_{i \in \mathcal{I}} \sum_{(i,i^+) \in, (i,i^-) \notin \mathcal{E}^L_{\mathcal{A}_I}} \ln \sigma\left(\mathbf{o}_i^T \mathbf{o}_{i^+} - \mathbf{o}_i^T \mathbf{o}_{i^-}\right), \tag{5.8}$$

where $\mathcal{E}^L_{\mathcal{A}_I}$ is the edge set considering $\{1, \cdots, L\}$ hop neighbors of $\mathbf{A}_I$. Finally, the model is optimized by the following loss:

$$\mathcal{L} = \mathcal{L}_{main} + \delta\mathcal{L}_{user} + \zeta\mathcal{L}_{item} + \gamma \left\|\Theta\right\|_2^2, \tag{5.9}$$

where Θ denotes the model parameters, δ and ζ are hyperparameters controlling the effect from higher-order neighbors. We additionally propose a robust SGDE (RSGDE) by only applying Equation (5.6) to SGDE to investigate how feature augmentation solely affects the performance.

5.4 Experiments

In this section, we comprehensively evaluate our proposed methods in terms of effectiveness and efficiency. Particularly, we aim to answer the following research questions:

- Do our proposed methods outperform other competitive baselines as well as our previously proposed GDE?
- How are the efficiency of our proposed methods, especially compared with GCN-based methods?
- How do hyperparameters affect model performance? Do the proposed designs positively affect the model performance?

5.4.1 Experimental Settings

Datasets and Evaluation Metrics

We list statistics of datasets in Table 5.1. Two MovieLens datasets: ML-1M and ML-100K are collected by GroupLens* and have been widely used to evaluate CF algorithms. CiteULike† is collected from CiteULike which allows users to create their own collections of articles. Gowalla [30] is a check-in dataset which records the locations users have visited. Yelp [140] is the Yelp Challenge data for user ratings on businesses. Since we focus on implicit feedbacks, we remove other auxiliary infromation such as ratings and reviews and only leave user/item IDs.

*`https://grouplens.org/datasets/movielens/`

†`https://github.com/js05212/citeulike-a`

Table 5.1. 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

To further verify our previous observations and generalize on other datasets, we evaluate on ML-1M, Yelp, and Gowalla.

We adopt two widely used evaluation metrics: Recall and nDCG [128] to evaluate model performance. Recall measures the ratio of the relevant items in the recommended list to all relevant items in test sets, while nDCG takes the rank into consideration by assigning higher scores to relevant items ranked 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 higher-order neighbor signals 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 hyperparameter tuning and report the average accuracy on test sets.

Baselines

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

- BPR [45]: This is a stable and classic MF-based method, exploiting a Bayesian personalized ranking loss for personalized rankings.
- EASE [129]: This is a neighborhood-based method with a closed form solution and show superior performance to many traditional CF methods.
- LightGCN [41]: This method removes activations functions and feature transformation, only leaves neighborhood aggregation for recommendation. We use

a three-layer architecture as the baseline.

- LCFN [91]: To remove the noise from interactions for recommendation, this method uses a low pass graph convolution to replace the spectral graph convolution. We set $F = 0.005$ and use a single-layer architecture.
- SGL-ED [38]: This model explores self-supervised learning based on LightGCN [41], by maximizing the agreements of multiple views from the same node, where the node views are generated by performing noise such as randomly removing the edges or nodes on the original graph. We set $\tau = 0.2$, $\lambda_1 = 0.1$, $p = 0.1$, and use a three-layer architecture.
- UltraGCN [135]: This model simplifies LightGCN by replacing neighborhood aggregation with a weighted MF, which shows faster convergence and less complexity.
- GDE [96]: This method only uses a very few graph features for recommendation without stacking layers, showing less complexity and higher efficiency than conventional GCN-based methods.

We remove some popular GCN-based methods such as Pinsage [93], NGCF [30], and SpectralCF [82] as the aforementioned baselines have already shown superiority over them.

Implementation Details

We implemented the proposed model based on PyTorch[‡] and released the code on Github[§]. For all models, we use SGD as the optimizer, 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 amongst $\{0.001, 0.005, 0.01, \cdots\}$; without specification, the model parameters are initialized with Xavier Initialization [131]; the batch size is set to 256. We report other hyperparameter settings in the next subsection.

5.4.2 Performance Comparison

We report the accuracy of our proposed SGDE variants and other baselines in Table 5.2, and make the following observations:

[‡]`https://pytorch.org/`

[§]`https://github.com/tanatosuu/GDE`

Table 5.2. Overall performance comparison. Improv.% denotes the improvements over the best baselines.

Datasets	Methods	nDCG@10	nDCG@20	Recall@10	Recall@20
Yelp	BPR	0.0388	0.0374	0.0371	0.0370
	Ease	0.0360	0.0362	0.0346	0.0368
	LCFN	0.0617	0.0627	0.0613	0.0653
	UltraGCN	0.0417	0.0403	0.0404	0.0403
	LightGCN	0.0751	0.0710	0.0725	0.0698
	SGL-ED	0.0817	0.0794	0.0784	0.0792
	GDE	0.0866	0.0850	0.0839	0.0860
	SGDE	0.0900	0.0877	0.0870	0.0880
	RSGDE	0.0947	0.0919	0.0917	0.0924
	CSGDE	**0.0966**	**0.0938**	**0.0933**	**0.0939**
	Improv.%	+11.55	+10.35	+11.20	+9.19
ML-1M	BPR	0.5521	0.4849	0.5491	0.4578
	Ease	0.3773	0.3249	0.3682	0.3000
	LCFN	0.5927	0.5197	0.5887	0.4898
	UltraGCN	0.5326	0.4688	0.5302	0.4434
	LightGCN	0.5917	0.5261	0.5941	0.5031
	SGL-ED	0.6029	0.5314	0.6010	0.5035
	GDE	0.6482	0.5681	0.6471	0.5376
	SGDE	0.6491	0.5730	0.6496	0.5445
	RSGDE	0.6559	0.5771	0.6554	0.5468
	CSGDE	**0.6581**	**0.5798**	**0.6583**	**0.5502**
	Improv.%	+1.52	+2.06	+1.73	+2.34
Gowalla	BPR	0.1086	0.0907	0.0917	0.0743
	Ease	0.0722	0.0670	0.0680	0.0642
	LCFN	0.1305	0.1132	0.1144	0.0980
	UltraGCN	0.0977	0.0815	0.0841	0.0681
	LightGCN	0.1477	0.1327	0.1368	0.1224
	SGL-ED	0.1789	0.1561	0.1563	0.1353
	GDE	0.1857	0.1632	0.1657	0.1449
	SGDE	0.1820	0.1607	0.1628	0.1428
	RSGDE	0.1917	0.1690	0.1691	0.1485
	CSGDE	**0.1950**	**0.1712**	**0.1714**	**0.1496**
	Improv.%	+5.01	+4.90	+3.44	+3.24

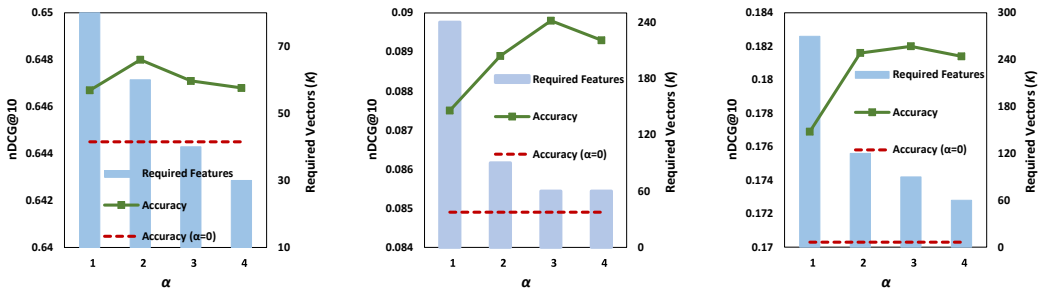
- Overall, GCN-based methods tend to show better performance over traditional CF methods, demonstrating the superiority of GCNs for CF especially when the data is extremely sparse, as GCNs can augment interactions with rich higher-order neighbor signals.

- SGL-ED performs the best among GCN-based baselines, which demonstrates the effectiveness of self-supervised learning for CF. UltraGCN shows relatively poor performance, despite the superior performance reported in the original paper. According to our previous analysis in Section 4.2.3, that UltraGCN is basically a weighted MF which loses the ability to leverage higher-order neighborhood, explaining why it performs poorly when data is sparse (with 20% interactions for training).

- By comparing SGDE, RSGDE, and CSGDE, we can see that the improvement from feature augmentation is more significant than leveraging higher-order neighbor signals, indicating that a well-designed data augmentation helps learn robust and generalizable representations.

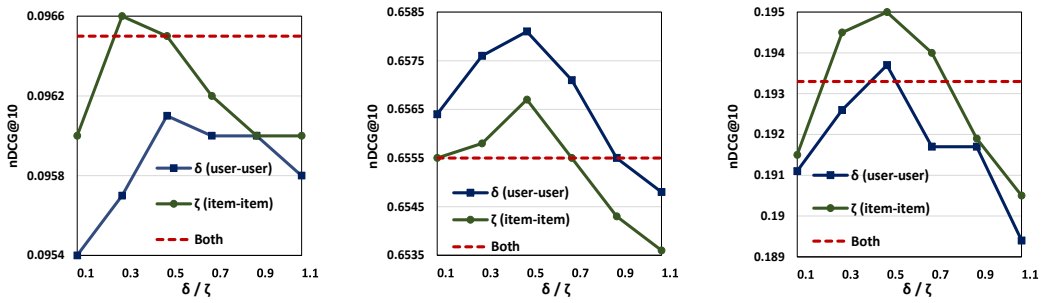
- SGDE achieves similar performance to GDE on ML-1M, outperforms GDE on Yelp, and slightly underperforms GDE on Gowalla. Note that GDE is trained with an adaptive loss which gains improvement over GDE with BPR loss, SGDE still outperforms GDE on Gowalla when both models are trained with the same BPR loss. RSGDE and CSGDE show consistent improvements over GDE, demonstrating the effectiveness of our proposed contrastive learning framework. For instance, the improvements of CSGDE over GDE on Yelp, ML-1M, and Gowalla are 11.6%, 1.5%, and 5.0%, respectively, in terms of nDCG@10.

- Compared with conventional GCN-based methods such as LightGCN, CSGDE outperforms it by 28.6%, 11.2%, and 32.0%, in terms of nDCG@10, on Yelp, ML-1M, and Gowalla, respectively, demonstrating the superiority and effectiveness of our proposed designs.

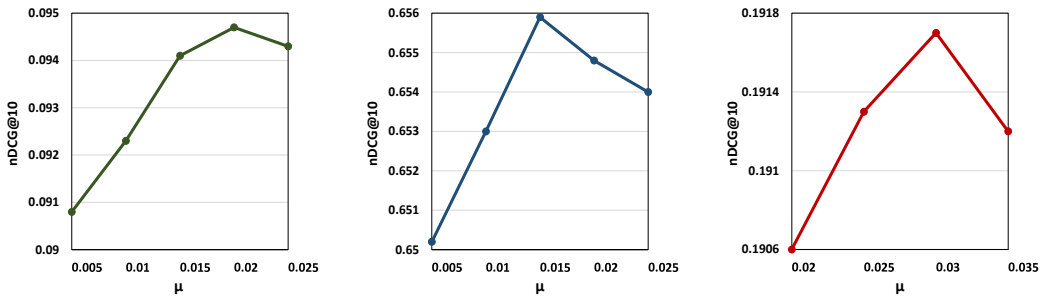


(a) ML-1M ($K = 120$ when $\alpha = 0$). (b) Yelp ($K = 2000$ when $\alpha = 0$). (c) Gowalla ($K = 3000$ when $\alpha = 0$).

Figure 5.3. How the accuracy of SGDE and the number of required features change with α.



(a) How the accuracy changes with δ and ζ on Yelp. (b) How the accuracy changes with δ and ζ on ML-1M. (c) How the accuracy changes with δ and ζ on Gowalla.



(d) How the accuracy changes with μ on Yelp. (e) How the accuracy changes with μ on ML-1M. (f) How the accuracy changes with μ on Gowalla.

Figure 5.4. Effect of the contrastive loss.

Table 5.3. How the accuracy changes with L.

	L=1	L=2	L=3	L=4
Yelp	0.0965	0.0963	0.0962	0.0964
ML-1M	0.6581	-	-	-
Gowalla	0.1924	0.1925	0.1949	0.1950

5.4.3 Model Analysis

Distribution Redundancy

We report how the number of required features K and the accuracy change with α in Figure 5.3. We can see K =120, 2000, and 3000 on ML-1M, Yelp, and Gowalla at $\alpha = 0$, and it constantly decreases as increasing α, where the best accuracy is achieved at α =2, 3, and 3, with K reduced to 60, 60, and 90, respectively. For instance, we can observe a 13x speed-up on Yelp by comparing the processing times for $K = 60$ and $K = 2000$ which are 1.74s and 24.94s, respectively, and the best accuracy at $\alpha = 3$ outperforms that at $\alpha = 0$ by 10.0%, indicating the important information can be concentrated in fewer spectral features by increasing α. Overall, we can both boost the efficiency and effectiveness by setting α in a reasonable way (*i.e.,* there is a trade-off between sharpening the distribution and keeping the original data uncontaminated).

Contrastive Loss

We report effect of the contrastive loss in Figure 5.4. In Figure 5.4 (a) - (c), we show how the accuracy changes with δ and ζ, 'Both' represents the accuracy of the model with the best settings of δ and ζ. We observe the followings:

- The accuracy increases first then drops as constantly increasing the hyperparameters and is mostly maximized at 0.5 on three datasets (excluding $\zeta = 0.3$ on Yelp). The contrastive loss tends to achieve a larger improvement on the sparser data, as the improvement is 0.3% on the denser data ML-1M, and nearly 2% on the other two datasets.

- incorporating both user and item homogeneous relations does not lead to an improvement compared with incorporating either of them. A reasonable explanation is that incorporating either relations can help optimize another re-

lation as well. For instance, consider the target users and neighbor users $(u, u^+) \in \mathcal{E}^l_{\mathcal{A}_U}, l = \{1, \cdots, L\}$, let i and i^+ be the items u and u^+ have interacted, respectively, then the possible distance between i and i^+ is $l-1 (l > 1)$, l, or $l+1$. When u and u^+ are optimized to be close, since i and i^+ are optimized to be near to u and u^+, respectively, then i and i^+ are pulled to be close as well. Thus, the item higher-order relations are also optimized to some extent when considering the user higher-order relations. Similarly, we can draw the same conclusion if we consider the relations between neighbor items and the target items. As a result, incorporating both relations results in overfitting, showing a worse performance on the data used in this work.

- We observe that the model performs better with user relations on ML-1M, and with item relations on Gowalla and Yelp. If we define the scale of relations as the number of all possible relations (*i.e.,* $|\mathcal{U}|^2$ for user relations and $|\mathcal{I}|^2$ for item relations), then this observation shows that optimizing the relations with a larger scale might lead to larger improvement.

Figure 5.4 (d) - (f) show how the accuracy changes with standard variation μ where a larger (smaller) μ implies intenser (weaker) noise, and the maximum accuracy is reached at 0.015, 0.02, and 0.03 on Yelp, ML-1M, and Gowalla respectively, where the noise tends to be more intense on sparser datasets when reaching the best accuracy. Table 5.3 shows how the accuracy changes with L. Since almost all users and items are connected when $L > 1$ on ML-1M (*i.e.,* all users and items can sampled as positive), we only report the accuracy with $L = 1$. The accuracy gradually increases as increasing L on Gowalla, while the best accuracy is achieved at $L = 1$ on Yelp, which might be due to the data density since Gowalla is sparser than Yelp. Overall, higher-order relations provide auxiliary information to help extract user preference.

5.5 Summary

In this chapter, we unveiled the distribution redundancy of GCN-based recommendation methods by showing that the number of required spectral feature is related to the spectral distribution, where a dataset with a flatter distribution tends to requires more spectral features when reaching the best performance, resulting in more computational cost. We define a renormalized adjacency matrix

with a hyper-parameter adjusting the sharpness of the spectral distribution to reduce the number of required spectral features, making the important information on the graph be concentrated in fewer features. By analyzing how GCL works for recommendation, we further proposed a scalable contrastive learning framework. Particularly, we performed feature augmentation by adding noise on the spectral feature where the intensity is according to their importance, and augmented sparse supervisory signals with higher-order neighbor. Experimental results on three datasets demonstrated the effectiveness and efficiency of our proposed SGDE variants over our previously proposed GDE as well as competitive baselines including both GCN-based and traditional CF methods. In the future, we will continue efforts to boost training efficiency and model scalability of GCN-based recommendation methods.

CHAPTER 6

Balancing Embedding Spectrum for Recommendation

In this chapter, we shed light on an issue in the existing pair-wise learning paradigm (*i.e.,* the embedding collapse problem) mentioned in Section 1.2, that the representations tend to span a subspace of the whole embedding space, leading to a suboptimal solution and reducing the model capacity. Specifically, optimization on observed interactions is equivalent to a low pass filter causing users/items to have the same representations and resulting in a complete collapse; while negative sampling acts as an unreliable high pass filter to alleviate the collapse by balancing the embedding spectrum but still leads to an incomplete collapse. To tackle this issue, we propose a novel method called DirectSpec, acting as a reliable all pass filter to balance the spectrum distribution of the embeddings during training, ensuring that users/items effectively span the entire embedding space. Additionally, we provide a thorough analysis of DirectSpec from a decorrelation perspective and propose an enhanced variant, DirectSpec$^+$, which employs self-paced gradients to optimize irrelevant samples more effectively. Moreover, we establish a close connection between DirectSpec$^+$ and uniformity, demonstrating that contrastive learning (CL) can alleviate the collapse issue by indirectly balancing the spectrum. Finally, we implement DirectSpec and DirectSpec$^+$ on two popular recommender models: MF and LightGCN. Our experimental results

demonstrate its effectiveness and efficiency over competitive baselines.

6.1 Introduction

Recommender systems have penetrated into our daily life, we can see them everywhere such as e-commerce [144], short-video [21], social network [4], and so on. Collaborative filtering (CF), a fundamental technique for recommendation to discover user preference based on the historical data, has attracted much attention in the last decades. The most extensively used CF technique, matrix factorization (MF) [11] which represents users and items as low dimensional latent vectors and the rating is estimated as the inner product between latent vectors, has been the cornerstone of modern recommender systems. Since MF estimates the rating with a simple linear function, subsequent works mostly focus on designing more powerful and complex algorithms to model non-linear user-item relations, including but not limited to multi-layer perceptron (MLP) [17], attention mechanism [58], reinforcement learning [59], transformer [22], diffusion model [145] graph neural network (GNN) [93, 132], etc., and have shown tremendous success.

Although different kinds of methods have been proposed, most of them can be considered as MF variants whose goal is to learn low dimensional representations (with dimension d) from the high dimensional sparse interaction matrix (with dimension $D \gg d$). Figure 6.1 illustrates the top 500 normalized singular value distribution of the interaction matrix of CiteULike (see Section 6.5 for data description). We observe that users/items are predominantly distributed along a few dimensions in the original space while most dimensions barely contribute (*i.e.,* with singular values close to 0) to the representations. Thus, when users and items are mapped into a more compact embedding space, it is expected that redundant dimensions are all removed and each dimension contributes to the user/item representations as equally and uniformly as possible (*i.e.,* the representations make full use of the embedding space).

Existing recommendation methods are mostly optimized by pulling observed user-item pairs closer than unobserved ones. Here, a simple yet fundamental question arises: Can user/item representations of existing work make full use of all dimensions? Unfortunately, by analyzing the spectrum of the embedding matrix, we empirically and theoretically show that users/items tend to span a subspace of the whole embedding space (with dimension $d' < d$), where the embeddings

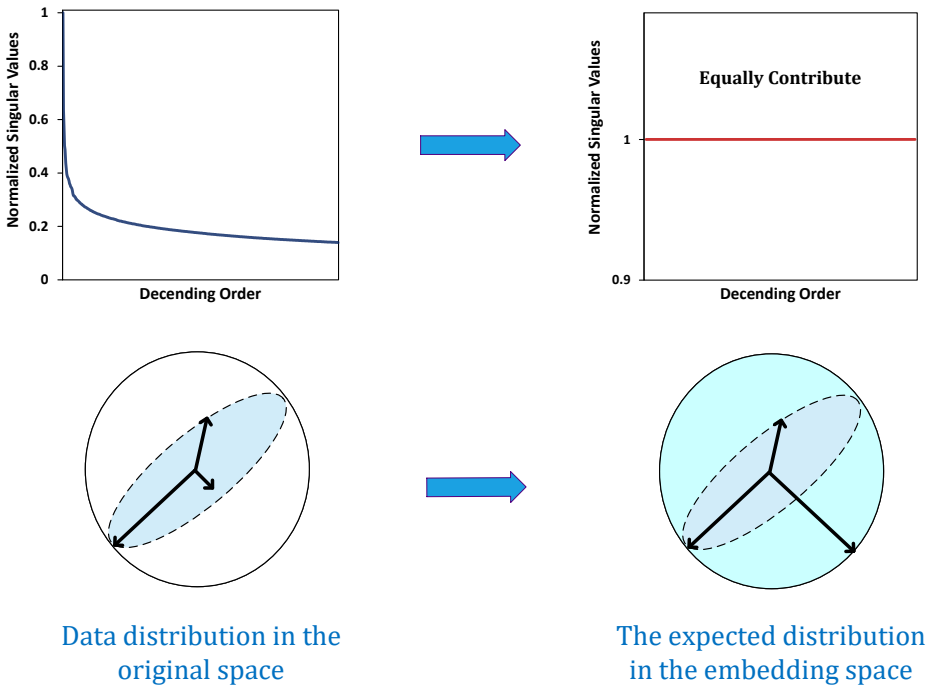


Figure 6.1. An illustration of the data distribution in the original space and expected distribution in the learning embedding space.

collapse along all (complete collapse) or certain dimensions (incomplete collapse). Particularly, optimization solely on observed interactions is equivalent to a low pass filter, where the representations of users and items tend to collapse to a constant vector. Negative sampling is the most common technique to optimize recommendation algorithms without causing an explicit embedding collapse by pushing away the unobserved user-item pairs, and we show that it is equivalent to a high pass filter that alleviates the collapse issue by balancing the embedding spectrum. However, the effectiveness of negative sampling in completely preventing the embedding collapse cannot be guaranteed, and collapse over certain dimensions still happens on existing pair-wise learning paradigms such as Bayesian personalized ranking (BPR) [45] and binary cross-entropy (BCE) loss [17]. Due to the data sparsity and long tailed distributions, increasing negative sampling ratios is considered as an effective way to to improve representation quality [17, 46], whereas we also demonstrate that it cannot further alleviate the collapse issue by evaluating different negative sampling ratios.

In this work, we tackle the embedding collapse issue from a spectral perspective. We observe that the extent of the collapse is closely related to the spectrum distribution of the embedding matrix. Specifically, only one singular value dominates when the representations completely collapse, whereas the singular values are uniformly distributed when the representations make full use of the embedding space. Inspired by this observation, we propose a novel method dubbed DirectSpec acting as an all pass filter to ensuring that all dimensions equally contribute to the representations. We theoretically and empirically show that DirectSpec can completely prevent the embedding collapse without explicitly sampling negative pairs by directly balancing the spectrum distribution, and provide a simple implementation with a complexity only as $\mathcal{O}(B^2d)$ where B is the batch size. Moreover, we shed light on DirectSpec from a decorrelation perspective, and propose an enhanced variant DirectSpec$^+$ which employs self-paced gradients to optimize the irrelevant samples that are highly correlated more effectively. By showing the close connection between DirectSpec and uniformity, we discover that contrastive learning (CL) can alleviate embedding collapse by balancing spectrum distribution in a similar way to DirectSpec, explaining the effectiveness of CL based recommendation algorithms. Finally, we implement DirectSpec and DirectSpec$^+$ on two popular baselines: MF [11] and LightGCN [41], and experimental results show that DirectSpec$^+$ improves BPR and LightGCN by up to 52.6% and 41.8% in terms of nDCG@10, respectively. The contribution of this work can be summarized as follows:

- We theoretically and empirically show that existing recommendation methods suffer from embedding collapse, that the representations tend to fall into a subspace of the whole embedding space, and analyze the mechanisms causing this issue.

- We propose a novel method DirectSpec which directly balances the spectrum distribution. We empirically and theoretically show that DirectSpec can prevent embedding collapse.

- We unveil the effectiveness of CL by showing that uniformity, a key design of CL can alleviate collapse issue by indirectly balancing the spectrum distribution that can be considered as a special case of DirectSpec.

- Extensive results on three datasets demonstrate the efficiency and effectiveness

of our proposed methods.

6.2 Preliminaries

Given the user set $\mathcal{U}$ and item set $\mathcal{I}$, the interaction matrix is defined as $\mathbf{R} \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$, the observed interactions are represented as $\mathbf{R}^+ = \{r_{ui} = 1 | u \in \mathcal{U}, i \in \mathcal{I}\}$. Users and items are first mapped to low dimensional vectors $\mathbf{H} \in \mathbb{R}^{(|\mathcal{U}|+|\mathcal{I}|)\times d}$ through an encoder, where the encoder can be as simple as a linear mapping [11] or advanced algorithms such as MLPs [17], GNNs [30], etc. Let $\mathbf{h}_u$ and $\mathbf{h}_i$ be the u's and i's representations, respectively. The goal of CF is to predict unobserved interactions $\mathbf{R}^- = \{r_{ui} = 0 | u \in \mathcal{U}, i \in \mathcal{I}\}$ estimated as the inner product between the user and item representations: $\hat{r}_{ui} = \mathbf{h}_u^T \mathbf{h}_i$. The model parameters Θ are optimized through a loss function $\mathcal{L}$ formulated as follows:

$$\arg\min_{\mathbf{\Theta}} \mathcal{L}(\hat{r}_{ui}, r_{ui}). \tag{6.1}$$

The loss function measures the difference between the estimated score and the ground truth. BPR and BCE loss are two extensively used learning frameworks for CF methods:

$$\begin{aligned} \mathcal{L}_{BPR} &= \sum_{(u,i)\in\mathbf{R}^+,(u,j)\in\mathbf{R}^-} -\ln \sigma(\hat{r}_{ui} - \hat{r}_{uj}), \\ \mathcal{L}_{BCE} &= \sum_{u\in\mathcal{U},i\in\mathcal{I}} -r_{ui} \ln \sigma(\hat{r}_{ui}) - (1 - r_{ui}) \ln \left(1 - \sigma(\hat{r}_{ui})\right), \end{aligned} \tag{6.2}$$

where $\sigma(\cdot)$ is the sigmoid function. BPR loss maximizes the difference between observed and unobserved user-item pairs, while BCE loss directly pulls the observed pairs close and pushes the unobserved ones away from each other. The embeddings are a low dimensional approximation of the sparse high dimensional interaction matrix, thus they should contain diverse and rich information representing the user-item relations. Rank is a commonly used metric to measure the dimension of a matrix, while it fails to tell the difference between the embedding matrix (1) with a 'sphere' distribution that users/items are uniformly distributed in each dimension of the space and (2) with a 'spheroid' distribution that users/items are predominantly distributed over some dimensions and insignificantly distributed over other dimensions. Although both matrices have the same rank, apparently (1) contains more diverse information than (2). Here, we introduce another tool:

Definition 4. ***Effective Rank.*** *Given singular values of the embedding matrix* $\mathbf{H}$*:* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d \geq 0$*, let* $p_k = \frac{\sigma_k}{\sum_k \sigma_k}$*, then the effective rank is defined as follows:*

$$erank(\mathbf{H}) = exp\left(H\left(p_1, \cdots, p_d\right)\right), \tag{6.3}$$

where $H\left(p_1, \cdots, p_d\right) = -\sum_k p_k \log p_k$ *is the Shannon entropy.*

Compared with rank, erank takes the singular value distribution into consideration: the more uniform (sharper) of the distribution, the higher (lower) of the erank [146]. The embedding matrix contains the least information when there is only one leading non-zero singular values ($\sigma_2 = \cdots = \sigma_d = 0$, and $erank(\mathbf{H}) = rank(\mathbf{H}) = 1$), while erank is maximized when each dimension equally contributes to the representations: $\sigma_1 = \cdots = \sigma_d$ ($erank(\mathbf{H}) = rank(\mathbf{H}) = d$).

6.3 Embedding Collapse in CF

6.3.1 Complete Collapse

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the node set contains all users and items: $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$, the edge set is represented by observed interactions: $\mathcal{E} = \mathbf{R}^+$. Let us first consider a log loss function (as it is extensively used) that only optimizes the observed interactions: $\mathcal{L} = -\sum \ln \sigma(\hat{r}_{ui}), (u, i) \in \mathbf{R}^+$.

Proposition 4. *Suppose* G *is connected, then* $\mathbf{h}_k \approx \mathbf{h}_z$ *for arbitrary nodes* k *and* z *when* $\mathcal{L}$ *completely converges.*

By calculating the gradient over the embeddings, the parameters are updated through stochastic gradient descent (SGD) as follows:

$$\begin{aligned} \mathbf{H}^{(l+1)} &= \mathbf{H}^{(l)} - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{H}^{(l)}} = \mathbf{H}^{(l)} + \alpha \mathbf{A}\mathbf{H}^{(l)}, \\ \mathbf{h}_u^{(l+1)} &= \mathbf{h}_u^{(l)} - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{h}_u^{(l)}} = \mathbf{h}_u^{(l)} + \alpha \sum_{(u,i) \in \mathbf{R}^+} \mathbf{A}_{ui} \mathbf{h}_i^{(l)}, \end{aligned} \tag{6.4}$$

where $\alpha \in (0, 1)$, $\mathbf{A}_{ui} = 1 - \sigma(\hat{r}_{ui})$ for $(u, i) \in \mathbf{R}^+$, and $\mathbf{A}$ can be considered as an adjacency matrix of $\mathcal{G}$. Equation (6.4) is similar to the message passing in GNN [25], which makes the nodes similar to their neighborhood. Repeating Equation (6.4) multiple times can further reach the higher-order neighborhood, causing indirectly connected nodes to be similar. While $\mathbf{A}_{ui}$ controlling the magnitude

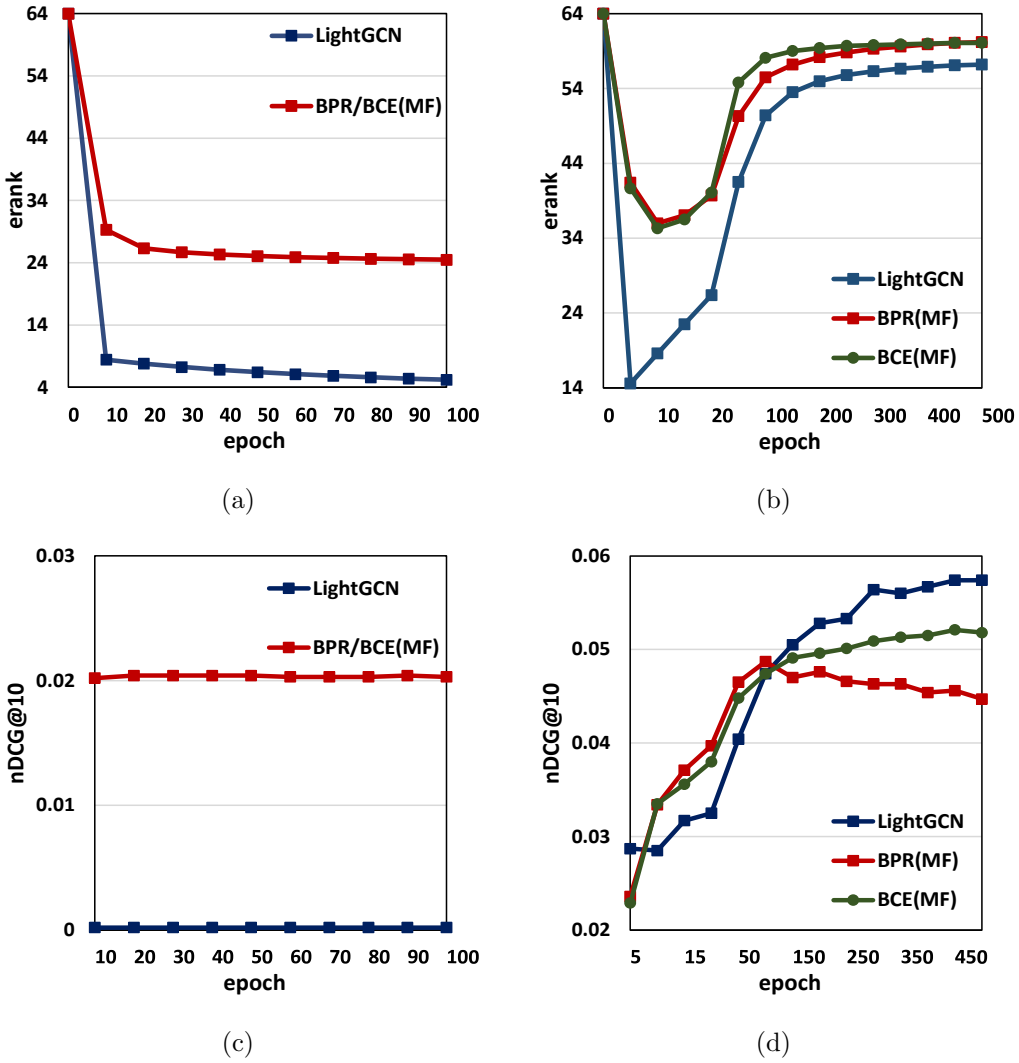


Figure 6.2. Embedding collapse on Yelp ($d = 64$). (a) and (b) show how embeddings completely and incompletely collapse, respectively; (c) and (d) show how nDCG@10 changes as the training proceeds under the two situations.

of the gradient tends to vanish as training proceeds, and eventually $\mathcal{L}$ converges when $\mathbf{A}$ degenerates into a zero matrix. Note that this issue does not exist on other loss functions such as Euclidean distance loss: $\sum \|\mathbf{h}_u - \mathbf{h}_i\|_2^2$. If we fix the magnitude of gradients unchanged, then Equation (6.4) can be rewritten as:

$$\mathbf{H}^{(l)} = (\mathbf{I} + \alpha\mathbf{A})^l \, \mathbf{H}^{(0)}. \tag{6.5}$$

Definition 5. ***Graph Filtering.*** *Let* $\lambda_1 > \lambda_2 > \cdots$ *be the eigenvalues of* $\mathbf{A}$.

Then $\mathcal{F}^L(\mathbf{A})$ *is a low pass filter if* $|\mathcal{F}^L(\lambda_i)| > |\mathcal{F}^L(\lambda_j)|$ *for* $\lambda_i > \lambda_j$*, and* $\mathcal{F}^H(\mathbf{A})$ *is a high pass filter if* $|\mathcal{F}^H(\lambda_i)| > |\mathcal{F}^H(\lambda_j)|$ *for* $\lambda_i < \lambda_j$*.*

Suppose $\mathbf{A}$ is normalized such as $|\lambda_k| \leq 1$, then we can see that $\mathbf{I} + \alpha\mathbf{A}$ is equivalent to a low pass filter. According to the spectral decomposition with $\mathbf{v}_k$ as the eigenvector:

$$(\mathbf{I} + \alpha\mathbf{A})^l = \sum_k (1 + \alpha\lambda_k)^l \mathbf{v_k}\mathbf{v_k}^T, \tag{6.6}$$

it is obvious that $\frac{(1+\alpha\lambda_k)^l}{(1+\alpha\lambda_1)^l} \to 0$ $(k \neq 1)$ as l is large enough, resulting in $\text{rank}((\mathbf{I} + \alpha\mathbf{A})^l) \to 1$, then:

$$\text{rank}\left(\mathbf{H}^{(l)}\right) \leq \min\left(\text{rank}\left((\mathbf{I} + \alpha\mathbf{A})^l\right), \text{rank}\left(\mathbf{H}^{(0)}\right)\right) = 1, \tag{6.7}$$

showing that all nodes have the same embedding representations. Considering that the gradient vanishing hinders the convergence of Equation (6.5), the representations of distinct nodes would not be completely the same.

In Figure 6.2 (a) and (c), we evaluate LightGCN and MF on how the embeddings completely collapse on Yelp when only considering the observed interactions, the parameters are initialized with Xavier initialization. We have the following observations:

- Before the training starts, the embeddings are uniformly distributed in the embedding space since $erank(\mathbf{H}) \approx d$.
- The erank monotonically decreases on both models, and the accuracy barely changes throughout the training, indicating that the embeddings collapse as the training starts and the issue becomes more serious as training proceeds.
- The erank decreases more rapidly on LightGCN than MF. Consider a K-th layer LightGCN, then Equation (6.5) can be rewritten as:

$$\mathbf{H}^{(l)} = (\mathbf{I} + \alpha\mathbf{A})^l \sum_{k=0}^{K} \mathbf{A}^k\mathbf{E}. \tag{6.8}$$

  Here, $\mathbf{E}$ is the initial stacked user/item embeddings sent to the encoder; we ignore the difference between the adjacency matrix used in Equation (6.5) and LightGCN for simplicity. We can see that increasing the layer K causes the loss function to converge faster, indicating that the over-smoothing in GNN [42] aggravates the collapse issue.

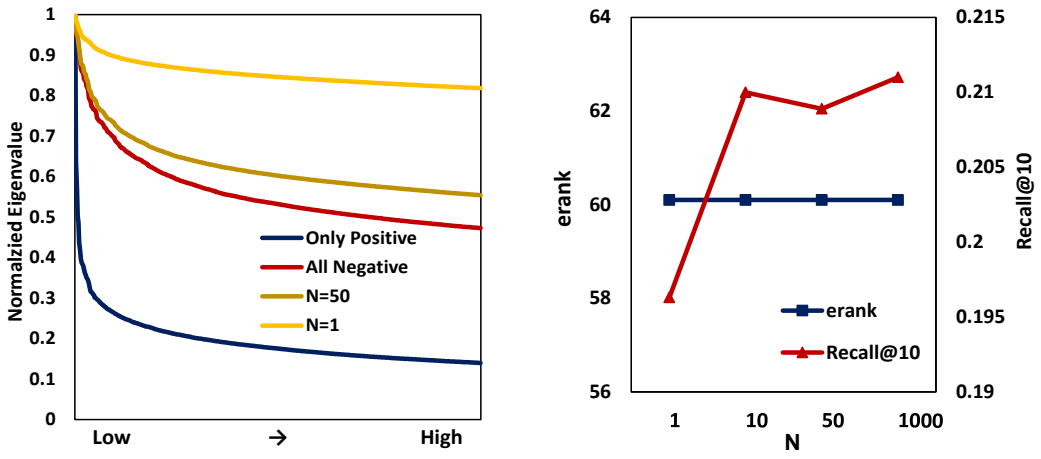
- The erank tends to converge to a value larger than 1. Besides the gradient vanishing issue mentioned above, Proposition 1 is based on the assumption that $\mathcal{G}$ is connected, which always does not hold on the real-world recommendation datasets. In other words, disconnected nodes (*i.e.,* no reachable paths between them) do not have the same representations.

6.3.2 Incomplete Collapse

Existing recommendation algorithms mostly exploit unobserved interactions for optimization. Naturally, we raise a question: Can existing pair-wise learning paradigm completely prevent the collapse? Similarly, we evaluate LightGCN (with BPR), MF (with BPR), and MF (with BCE) on Yelp, and show how erank and accuracy change as training proceeds in Figure 6.2 (b) and (d). The erank plunges at first, showing a trend similar to Figure 6.2 (a). Gradually, the erank increases and tends to converge to a value lower than d, indicating that the embeddings are negligibly distributed over some dimensions. LightGCN drops more rapidly than MF and converges to a smaller value. In the meanwhile, the accuracy also increases accordingly, and shows a trend similar to erank. It is obvious that existing learning paradigms can alleviate the collapse issue but still cannot completely prevent it. We attempt to analyze how negative sampling alleviates the collapse issue and its weakness. Take the BCE loss as an example, the parameters are updated through SGD as follows:

$$\begin{aligned}
\mathbf{H}^{(l+1)} &= \mathbf{H}^{(l)} + \alpha \left(\mathbf{A} - \bar{\mathbf{A}}\right) \mathbf{H}^{(l)}, \\
\mathbf{h}_u^{(l+1)} &= \mathbf{h}_u^{(l)} + \alpha \sum_{(u,i)\in\mathbf{R}^+} \mathbf{A}_{ui} \mathbf{h}_i^{(l)} - \alpha \sum_{(u,j)\in\mathbf{R}^-} \bar{\mathbf{A}}_{uj} \mathbf{h}_j^{(l)},
\end{aligned} \tag{6.9}$$

where $\bar{\mathbf{A}}_{uj} = \sigma(\hat{r}_{uj})$ for $(u, j) \in \mathbf{R}^-$. Intuitively, as the direction of the gradients are opposite to that of observed interactions, the disconnected nodes (unobserved interactions) are pushed away from the target users/items, preventing the representations from collapsing. Furthermore, as shown previously in Section 6.3.1 that optimization on observed interactions is equivalent to a low pass filter, Equation (6.9) can be disentangled to a low pass $\mathbf{I} + \alpha\mathbf{A}$ and a high pass filter $\mathbf{I} - \alpha\bar{\mathbf{A}}$ (suppose $\mathbf{A}$ and $\bar{\mathbf{A}}$ are symmetrically normalized), where a high pass filter can balance the embeddings spectrum by reducing the weights on low frequency and raising the importance of high frequency components, leading to a more uniform distribution. To verify our analysis, we visualize the spectrum of $\mathbf{A} - \bar{\mathbf{A}}$ in Figure



(a) (b)

Figure 6.3. (a) the normalized eigenvalue distribution (Top 500) of $\mathbf{A} - \bar{\mathbf{A}}$ on CiteULike; (b) how erank and recall@10 changes with varying negative sampling ratios N on CiteULike.

6.3 (a), where we fix the elements of $\mathbf{A}$ and $\bar{\mathbf{A}}$ to 1 for simplicity since they are changing during training. $\bar{\mathbf{A}} = 0$ (zero matrix) for 'All positive', and $\bar{\mathbf{A}}$ contains all unobserved interactions for 'All negative'. It is obvious that $\mathbf{A} - \bar{\mathbf{A}}$ has a more balanced spectrum than $\mathbf{A}$, leading to a higher erank. However, due to the varying elements, there is no guarantee that repeating Equation (6.9) can consistently alleviate the collapse issue and eventually completely prevents it. The experimental results also show that the representations still suffer from an incomplete collapse. A plausible solution is to raise the negative sampling ratio as unobserved interactions usually account for over 95% of all interactions that cannot be sufficiently sampled during training, whereas the spectrum of $\mathbf{A} - \bar{\mathbf{A}}$ is the most uniform when the negative sampling ratio $N = 1$, and raising the ratio seems to aggravate the collapse issue again as shown in Figure 6.3 (a). We further evaluate BCE with different negative sampling ratios and report the results in Figure 6.3 (b). The accuracy increases by raising the negative sampling ratio $N > 1$, while the erank remains unchanged, showing that introducing more negative samples fails to further alleviate or prevent the embedding collapse.

6.4 Methodology

6.4.1 Balancing Embedding Spectrum

The spectrum distribution can directly reflect the extent of embedding collapse. Intuitively, a uniform distribution indicates that different dimensions equally contribute to the embeddings and leads to a high effective rank. Therefore, a reliable solution is an all pass filter $\mathcal{F}^A$ such that $\mathcal{F}^A(\sigma_i)=\mathcal{F}^A(\sigma_j)$ for $\sigma_i \neq \sigma_j$. By parameterizing $\mathcal{F}^A$, it is straightforward to directly balance the spectrum by optimizing the following loss:

$$\min \sum_{k \neq z} \|\sigma'_k - \sigma'_z\|_2^2, \tag{6.10}$$

where σ'_k is the singular value of $\mathcal{F}^A(\mathbf{H})$. However, such a design requires the computation of singular values during each training and introduces more model parameters that might hinder the model convergence, thus we stick to a simple yet effective design in this work. Consider an affinity graph $\mathcal{G}' = (\mathcal{V}, \hat{\mathbf{A}})$, where the node features are represented by the embedding matrix $\mathbf{H}$ and the adjacency relation is measured by the similarity score between nodes: $\hat{\mathbf{A}}_{ui} = \mathbf{h}_u^T \mathbf{h}_i$, we define the massage passing on $\mathcal{G}'$ as follows:

$$\mathcal{F}^A(\mathbf{H}) = \mathbf{H} - \alpha \hat{\mathbf{A}}^L \mathbf{H}, \tag{6.11}$$

where $\alpha \in \mathbb{R}^+$, $L \in \mathbb{N}$ is the matrix power.

Proposition 5. $\overbrace{\mathcal{F}^A \cdots \mathcal{F}^A}^{T}(\mathbf{H})$ *(Repeating* $\mathcal{F}^A(\mathbf{H})$ *T times) is equivalent to an all pass filter where* $\text{erank}(\overbrace{\mathcal{F}^A \cdots \mathcal{F}^A}^{T}(\mathbf{H})) = d$.

We can rewrite Equation (6.11) according to singular value decomposition (SVD) as follows:

$$\begin{aligned} \mathcal{F}^A(\mathbf{H}) = \mathbf{H} - \alpha \left(\mathbf{H}\mathbf{H}^T\right)^L \mathbf{H} &= \mathbf{P} diag\left(\sigma_k\right) \mathbf{Q}^T - \alpha \mathbf{P} diag\left(\sigma_k^{2L+1}\right) \mathbf{Q}^T \\ &= \mathbf{P} diag\left(\sigma_k \left(1 - \alpha \sigma_k^{2L}\right)\right) \mathbf{Q}^T, \end{aligned} \tag{6.12}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are stacked singular vectors, $diag(\cdot)$ is the diagonalization operation. It is obvious that $1 - \alpha\sigma_i^{2L} > 1 - \alpha\sigma_j^{2L}$ for $\sigma_i < \sigma_j$. Thus $1 - \alpha\sigma_k^{2L}$ can be considered as rescaled factors: the larger singular values corresponding

to the lower frequency components are multiplied by smaller weights, leading to a more uniform distribution when $L > 0$. L and α control the weight multiplied on the singular value of $\mathbf{H}$, thus directly affect the efficiency of spectrum balancing. A larger (smaller) L makes the rescaled factors smaller (larger) on the large singular values, and α controls the norms of the rescaled factors that has a similar effect to L. Since σ_k^L increases fast, a large L could reduce the algorithm's efficiency where the large singular values in the original distribution are multiplied by too small weights, causing a sharp singular value distribution similar to the original one. In addition, too large L or α could also break the non-negativity of the singular value, thus we need to assure that $1 - \alpha\sigma_1^{2L} > 0$. According to the definition of Entropy and erank, a more uniform distribution results in a larger erank, indicating that $erank(\mathcal{F}^A(\mathbf{H})) \geq erank(\mathbf{H})$ where $=$ holds when $erank(\mathbf{H}) = d$. By sufficiently repeating Equation (6.12), eventually $erank(\mathbf{H}) = d$. We summarize the above analysis and propose Algorithm 1 to tackle the embedding collapse. We run Algorithm 1 on a randomly generated matrix with the code torch.randn$(10, 10)$*, and report the results with different setting of α, L, and T in Table 6.1, where $1 - \alpha\sigma_1^{2L} < 0$ when $L > 3$. We can see that the erank tends to converge to $d = 10$ as we increase the iteration T, scaling hyperparameter α, and the order L, demonstrating the effectiveness of Algorithm 1 balancing the embedding spectrum.

Algorithm 1 is impractical since the complexity is $\mathcal{O}(TLD^2d)$ where $D = |\mathcal{U}| + |\mathcal{I}|$. We propose a simplified implementation of DirectSpec formulated in Algorithm 2. Since the singular value increases fast with L which could break the non-negativity of the singular value, and in practice we do not need the erank to strictly be d (we will explain in Section 6.4.3), we can set $T = 1$ and $L = 1$. In addition, instead of updating the whole embedding matrix, we iteratively rebalance the spectrum of a smaller matrix with a size in accordance with the batch size.

6.4.2 A Decorrelation Perspective

In this subsection, we show how users and items are represented in the embedding space through our proposed DirectSpec from a spatial perspective.

*`https://pytorch.org/`

Algorithm 1: Balancing Embedding Spectrum via Message Passing

Input: Embedding matrix $\mathbf{H}$; matrix power L; the number of iteration T; scaling hyperparameter $\alpha \in \mathbb{R}^+$

for $t = 1$ **to** T **do**

$\mathbf{H} \leftarrow \mathbf{H} - \alpha \hat{\mathbf{A}}^L \mathbf{H}$;

end

Return H

Algorithm 2: DirectSpec

Input: Embedding matrix $\mathbf{H}$; batch size B; scaling hyperparameter $\alpha \in \mathbb{R}^+$

sample the users and items: $\mathcal{U}_B$, $\mathcal{I}_B$;

generate the normalized embeddings: $\mathbf{H}_B^U$, $\mathbf{H}_B^I$;

$\mathbf{H}_B^U \leftarrow \mathbf{H}_B^U - \alpha \mathbf{H}_B^U {\mathbf{H}_B^U}^T \mathbf{H}_B^U$;

$\mathbf{H}_B^I \leftarrow \mathbf{H}_B^I - \alpha \mathbf{H}_B^I {\mathbf{H}_B^I}^T \mathbf{H}_B^I$;

Return $\mathbf{H}_B^U$, $\mathbf{H}_B^I$

Proposition 6. *Algorithm 2 is equivalent to optimizing the following loss:*

$$L_{ds} = \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{U}} \left\| \mathbf{h}_u^T \mathbf{h}_v \right\|^2 + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \left\| \mathbf{h}_i^T \mathbf{h}_j \right\|^2 . \tag{6.13}$$

Table 6.1. An toy example demonstrating the effectiveness of Algorithm 1. The matrix is randomly generated with a size 10. Without specification, $T = 50$, $\alpha = 1e-3$, and $L = 1$.

T	**0**	**1**	**10**	**100**	**500**	**1000**
erank	7.64	7.72	8.11	8.59	9.57	9.75
α	**0**	**1e-3**	**2e-3**	**5e-3**	**1e-2**	**2e-2**
erank	7.64	8.43	8.71	9.09	9.35	9.59
L	**0**	**1**	**2**	**3**	**-**	**-**
erank	7.64	7.80	8.45	8.83	-	-

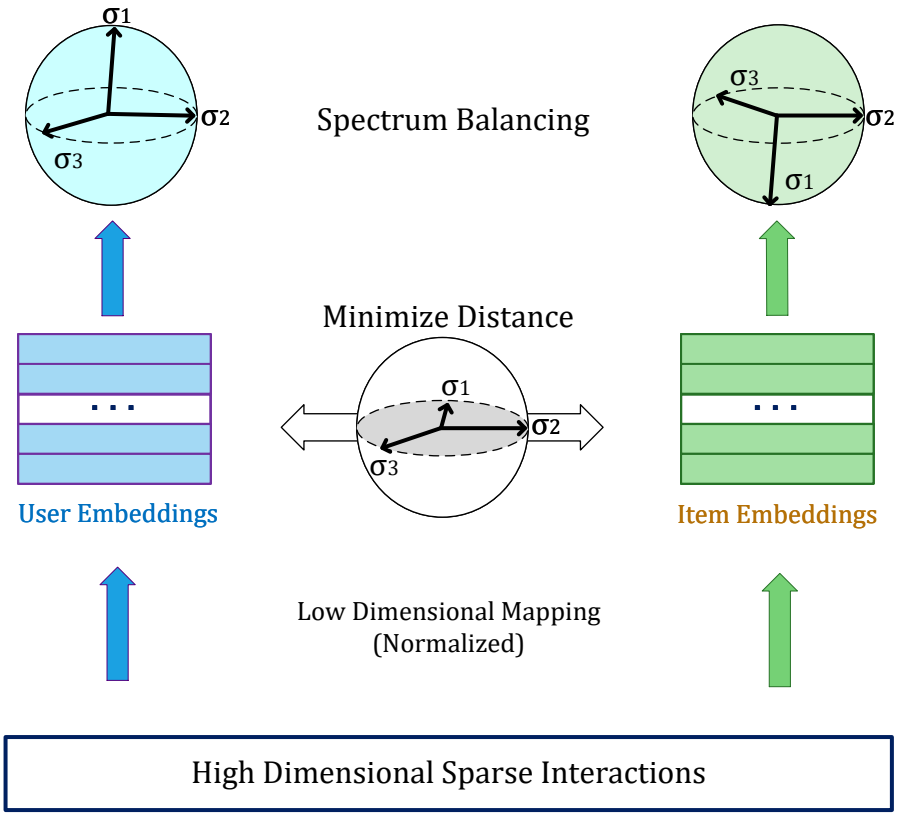


Figure 6.4. An illustration of the proposed DirectSpec.

By calculating the gradient over user and item embeddings:

$$\frac{\partial L_{ds}}{\mathbf{h}_u} = 4\sum_v \left(\mathbf{h}_u^T\mathbf{h}_v\right)\mathbf{h}_v,\ \frac{\partial L_{ds}}{\mathbf{h}_i} = 4\sum_j \left(\mathbf{h}_i^T\mathbf{h}_j\right)\mathbf{h}_j, \tag{6.14}$$

the parameter updating through SGD can be formulated as follows:

$$\begin{aligned}\mathbf{H}_U &\leftarrow \mathbf{H}_U - \alpha\left(\mathbf{H}_U\mathbf{H}_U^T\right)\mathbf{H}_U,\\ \mathbf{H}_I &\leftarrow \mathbf{H}_I - \alpha\left(\mathbf{H}_I\mathbf{H}_I^T\right)\mathbf{H}_I,\end{aligned} \tag{6.15}$$

where $\mathbf{H}_U$ and $\mathbf{H}_I$ are user and item embedding matrices, respectively. Therefore, Equation (6.13) is equivalent to DirectSpec. Intuitively, two users/items are (positively or negatively) correlated if $|\mathbf{h}_u^T\mathbf{h}_v| > 0$ or $|\mathbf{h}_i^T\mathbf{h}_j| > 0$, then we can see that the goal of Equation (6.13) is to decorrelate/orthogonalize different users and items (*e.g.*, $\mathbf{h}_u^T\mathbf{h}_v \to 0$ and $\mathbf{h}_i^T\mathbf{h}_j \to 0$) instead of simply pushing them away.

Proposition 7. *The global minimum of Equation (6.13) can be reached when* $d > \max(|\mathcal{U}|, |\mathcal{I}|)$.

Suppose that n user or item embeddings are perfectly decorrelated (*i.e.,* any two embeddings are orthogonal), then they are linearly independent as well:

$$\alpha_1 \mathbf{h}_1 + \cdots + \alpha_n \mathbf{h}_n = 0, \tag{6.16}$$

since we get $\alpha_k = 0$ by multiplying Equation (6.16) by $\mathbf{h}_k^T$, $k = \{1, \cdots, n\}$. Therefore, any two users/items being orthogonal implies that all embedding vectors are linearly independent, requiring d to be greater than $\max(|\mathcal{U}|, |\mathcal{I}|)$.

From a different perspective, the matrices $\mathbf{H}_U\mathbf{H}_U^T$ and $\mathbf{H}_I\mathbf{H}_I^T$ represent the similarity scores between users and items, respectively, and we have the following relations according to SVD:

$$\mathbf{H}_U\mathbf{H}_U^T = \mathbf{P}diag(\sigma_k^2)\mathbf{P}^T,\ \mathbf{H}_I\mathbf{H}_I^T = \mathbf{Q}diag(\sigma_k^2)\mathbf{Q}^T. \tag{6.17}$$

For simplicity, we use the same notations denoting singular values and vectors of $\mathbf{H}_U$ and $\mathbf{H}_I$, since we do not emphasize their difference here. In the meanwhile, $\mathbf{P}$ and $\mathbf{Q}$ are also the eigenvectors of $\mathbf{H}_U$ and $\mathbf{H}_I$, respectively. Let λ_k denote the eigenvalue of $\mathbf{H}_U\mathbf{H}_U^T$ and $\mathbf{H}_I\mathbf{H}_I^T$, we have $\lambda_k = \sigma_k^2$. When $(\mathbf{H}_U\mathbf{H}_U^T)_{uv} \to 0$ and $(\mathbf{H}_I\mathbf{H}_I^T)_{ij} \to 0$, $(\mathbf{H}_U\mathbf{H}_U^T)$ and $(\mathbf{H}_I\mathbf{H}_I^T)$ are optimized to be identity matrices with uniform spectrum distributions: $\lambda_1 = \lambda_2 = \cdots = 1$, thus $\mathbf{H}_U$'s and $\mathbf{H}_I$'s spectra are also uniform, leading to a maximum erank. The above observations reveal that unobserved pairs should stay irrelevant instead of simply being pushed away like BCE and BPR loss, which is more consistent with users' true preference over unobserved interactions. We will empirically compare DirectSpec and negative sampling in Section 6.5.2.

6.4.3 DirectSpec$^+$

As the observed pairs should be deeply correlated, perfect user/item decorrelation is not the goal of personalized recommendation, explaining why we do not need (1) the erank of the embedding to strictly be d and (2) d to be large enough to reach the perfect decorrelation. The training objective should balance between user/item correlation and decorrelation:

$$L = \sum_{(u,i)\in\mathbf{R}^+} \ln \sigma\left(\mathbf{h}_u^T\mathbf{h}_i\right) + L_{ds}. \tag{6.18}$$

The first term pulls users/items to be close to each other, while the second term offsets the negative effect from the first term by orthogonalizing them. From Equation (6.14), we can see that all samples are optimized with the same pace. To enhance the efficiency, we can penalize more on the highly correlated user/item pairs (*i.e.,* $\mathbf{h}_u^T\mathbf{h}_v \to 1$ and $\mathbf{h}_i^T\mathbf{h}_j \to 1$) by improving Equation (6.13) as:

$$L_{ds} = \sum_{u\in\mathcal{U}}\sum_{v\in\mathcal{U}} \exp\left(\mathbf{h}_u^T\mathbf{h}_v/\tau\right) + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{I}} \exp\left(\mathbf{h}_i^T\mathbf{h}_j/\tau\right). \tag{6.19}$$

Compared with Equation (6.14), $\mathbf{h}_u^T\mathbf{h}_v \to 1$ and $\mathbf{h}_i^T\mathbf{h}_j \to 1$ are pushed away at a faster pace (*i.e.,* larger gradients); τ is a temperature controlling the strength of penalties on highly correlated samples. Note that Equation (6.19) is equivalent to a pairwise Gaussian potential: $\exp\left(\frac{\mathbf{h}_u^T\mathbf{h}_v}{\tau}\right) = \exp\left(\frac{-\|\mathbf{h}_u-\mathbf{h}_v\|^2}{2\tau} + \frac{1}{\tau}\right)$, and the minimum is reached when users/items are orthogonal to each other in the embedding space when d is large enough according to Proposition 3 in [147]. Thus, the training objective of the enhanced algorithm is consistent with the original DirectSpec. Furthermore, we notice that some unobserved pairs are correlated as well to some extent, such as the users showing similar preference or items interacted by similar users that should be decorrelated with a slower pace than other irrelevant pairs. Here, we propose two adaptive temperature designs to adapt to the pairs with different degrees of correlations:

i. We dynamically learn the temperature through an attention mechanism such as $\mathbf{\Gamma}_{uv} = \sigma(\mathbf{W}^T[\mathbf{h}_u||\mathbf{h}_v])$ for u and v, where $||$ stands for the concatenate operation and $\mathbf{W} \in \mathbb{R}^{2d}$ is the transform matrix.

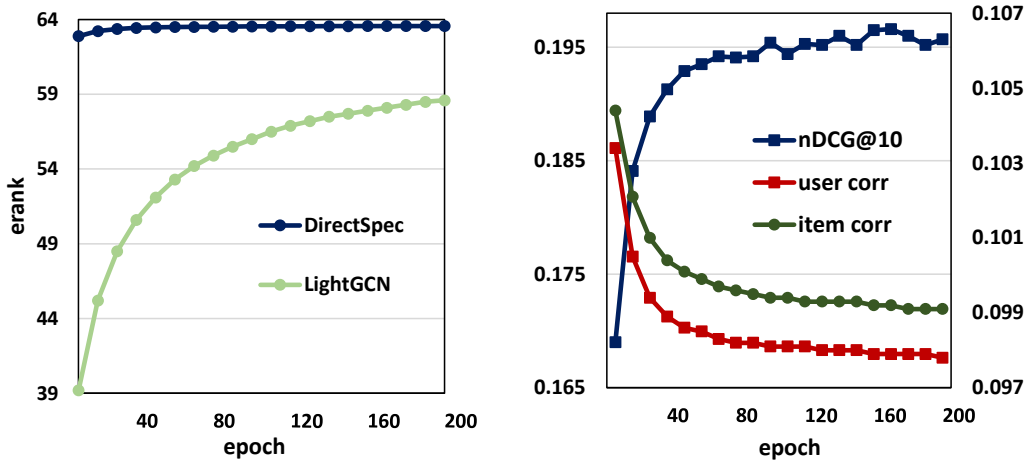
ii. We define the graph distance between two nodes as the minimum length of path between them on $\mathcal{G}$, then an unparameterrized design is defined as follows:

$$\mathbf{\Gamma}_{uv} = \begin{cases} \tau_0 & d_{\mathcal{G}}(u,v) \le K \\ \\ \tau_1 & otherwise, \end{cases} \tag{6.20}$$

where $d_{\mathcal{G}}(u,v)$ is the graph distance between u and v. Here, the pairs with $d_{\mathcal{G}}(u,v) \le K$ are considered correlated, and $\tau_0 \le \tau_1$.

By calculating the gradients of Equation (6.19), we can incorporate its parameter-updating formula into Algorithm 2, and propose DirectSpec$^+$:

$$\begin{aligned} \mathbf{H}_B^U &\leftarrow \mathbf{H}_B^U - \alpha \cdot \text{softmax}\left(\mathbf{H}_B^U {\mathbf{H}_B^U}^T \odot \mathbf{\Gamma}\right)\mathbf{H}_B^U, \\ \mathbf{H}_B^I &\leftarrow \mathbf{H}_B^I - \alpha \cdot \text{softmax}\left(\mathbf{H}_B^I {\mathbf{H}_B^I}^T \odot \mathbf{\Gamma}\right)\mathbf{H}_B^I. \end{aligned} \tag{6.21}$$



(a) (b)

Figure 6.5. (a): DirectSpec$^+$ can prevent embedding collapse on CiteULike; (b): users/items are decorrelated as the training of DirectSpec$^+$ proceeds.

To regulate the magnitude of the gradient, we rewrite it as a Softmax function. Other steps of DirectSpec$^+$ are consistent with Algorithm 2. In Figure 6.5 (a), we compare the erank of LightGCN and DirectSpec, and show that DirectSpec can prevent the embeddings from collapsing in the first place as $erank \approx d$ throughout the training; In Figure 6.5 (b), 'user/item corr' is short for user/item correlation measured by Equation (6.13), reflecting the correlation between users/items. We can see that users/items tend to be more orthogonal as training proceeds while the accuracy increases, demonstrating that decorrelating users/items is a key to prevent embedding collapse instead of simply pushing them as far away as possible.

6.4.4 Discussion

Connection Between DirectSpec$^+$ and Uniformity Loss

Contrastive learning has shown great potential in recommender systems [38, 148], which adopts an InfoNCE loss:

$$L_{cl} = -\sum_u \ln \frac{\exp\left(\mathbf{h}_u^T \mathbf{h}_{u^+}/\tau\right)}{\sum_v \exp\left(\mathbf{h}_u^T \mathbf{h}_v/\tau\right)} - \sum_i \ln \frac{\exp\left(\mathbf{h}_i^T \mathbf{h}_{i^+}/\tau\right)}{\sum_j \exp\left(\mathbf{h}_i^T \mathbf{h}_j/\tau\right)}, \tag{6.22}$$

Table 6.2. Comparison of time complexity.

Complexity/Model	DirectSpec	CL	GCL
Self-supervised	$c\|\mathbf{R}^+\|Bd$	$c\|\mathbf{R}^+\|\left(\|\mathcal{U}\|+\|\mathcal{I}\|\right)d$ or $c\|\mathbf{R}^+\|Bd$	$c\|\mathbf{R}^+\|\left(\frac{L\|\mathbf{R}^+\|}{B}+\|\mathcal{U}\|^2+\|\mathcal{I}\|^2\right)d$
Supervised	$c\|\mathbf{R}^+\|d$	$2c\|\mathbf{R}^+\|d$ or $c\|\mathbf{R}^+\|Bd$	$2c\|\mathbf{R}^+\|d$

that can be decoupled to alignment and uniformity loss. Alignment minimizes the distance between positive pairs (*i.e.,* (u, u^+) and (i, i^+)). In the meanwhile, embeddings from different users/items are pushed away via uniformity loss, which has been shown necessary for desirable user/item representations [115, 149]. To unveil the effectiveness of uniformity, we fix alignment: $\mathbf{h}_u^T\mathbf{h}_{u^+} = 1$ and $\mathbf{h}_i^T\mathbf{h}_{i^+} = 1$, and let $L_{cl}(u) = -\ln\frac{\exp(1/\tau)}{\sum_v \exp(\mathbf{h}_u^T\mathbf{h}_v/\tau)}$. Then, the gradients over the parameters are calculated as follows:

$$\begin{aligned}\frac{\partial L_{cl}}{\partial \mathbf{h}_u} &= \frac{\partial L_{cl}(u)}{\partial \mathbf{h}_u} + \sum_{v\neq u}\frac{\partial L_{cl}(v)}{\partial \mathbf{h}_u} \\ &= \sum_v \frac{\exp\left(\mathbf{h}_u^T\mathbf{h}_v/\tau\right)}{\sum_k \exp\left(\mathbf{h}_u^T\mathbf{h}_k/\tau\right)}\mathbf{h}_v/\tau + \sum_v \frac{\exp\left(\mathbf{h}_u^T\mathbf{h}_v/\tau\right)}{\sum_k \exp\left(\mathbf{h}_v^T\mathbf{h}_k/\tau\right)}\mathbf{h}_v/\tau,\end{aligned} \tag{6.23}$$

thus the parameters are updated via SGD as:

$$\mathbf{H}_U \leftarrow \mathbf{H}_U - \alpha\cdot\left(\text{softmax}\left(\mathbf{H}_U{\mathbf{H}_U}^T/\tau\right) + \mathbf{S}\mathbf{D}^{-1}\right)\mathbf{H}_U, \tag{6.24}$$

where $\mathbf{S} = \exp\left(\mathbf{H}_U{\mathbf{H}_U}^T/\tau\right)$, and $\mathbf{D}$ is a diagonal matrix with $D_{uu} = \sum_v \mathbf{S}_{uv}$, $\text{softmax}\left(\mathbf{H}_U{\mathbf{H}_U}^T/\tau\right) = \mathbf{D}^{-1}\mathbf{S}$. Item embeddings are updated in a way similar to Equation (6.24). We can see that Equation (6.24) is the same as DirectSpec$^+$ if we ignore the term $\mathbf{S}\mathbf{D}^{-1}$, revealing that uniformity can alleviate collapse by indirectly rebalancing the embedding spectrum.

Complexity

We compare the complexity of DirectSpec with graph contrastive learning (GCL) and CL in Table 6.2, where B and c denote the batch size and the required epochs, respectively. All the three methods adopt a multi-task training strategy, where

Table 6.3. Statistics of datasets.

Datasets	#User	#Item	#Interactions	Density%
CiteULike	5,551	16,981	210,537	0.223
Yelp	25,677	25,815	731,672	0.109
Gowalla	29,858	40,981	1,027,370	0.084

the complexity comes from the supervised and self-supervised learning task (if we consider the process of spectrum balancing in DirectSpec as a self-supervised learning task). For self-supervised learning task, GCL has the expensive computational cost as it generates two node views on perturbed graphs, whose time complexity is twice that of the backbone GCN (here, we use LightGCN). CL has lower time complexity than GCL as it does not exploit graph structures, and the complexity can be reduced to that of DirectSpec: $c|\mathbf{R}^+|Bd$ if it adopts a mini-batch training [148]. For supervised learning task, we ignore the computational cost of the encoder (*i.e.,* the complexity for generating the embeddings) as DirectSpec and CL can also be implemented on other algorithms (*e.g.,* MLP, GNN, transformer, etc.). Compared with most of the works adopting BPR or CL loss which still rely on negative samples, DirectSpec only samples the positive pairs and is equipped with a simple loss. Overall, DirectSpec shows lower time complexity over GCL and CL methods.

6.5 Experiments

In this section, we comprehensively evaluate our proposed methods on three public datasets. We implement DirectSpec on two popular baselines: MF and LightGCN. Particularly, we introduce data description and implementation details in Section 6.5.1; we compare DirectSpec with other competitive baselines in terms of accuracy and efficiency in Section 6.5.2. Finally, we conduct model analysis in Section 6.5.3, including detailed analysis and experimental results on how DirectSpec prevents embeddings from collapsing without relying on negative samples, and how different settings of hyperparameters affect model performance.

6.5.1 Experimental Settings

Datasets

We use the following three publicly available real-world datasets in this work, where the statistics of them are summarized in Table 6.3.

- **CiteULike**[†]: This is an scholarly article recommendation dataset. Users are allowed to create their own collections of articles including abstracts, titles, and tags, etc.
- **Yelp** [140]: This is a business dataset from Yelp Challenge data. The items are point of interests (POIs), users can leave reviews and ratings.
- **Gowalla** [30]: This is a check-in datasets recording which locations users have visited.

Since we focus on CF for implicit feedbacks, we remove auxiliary information such as reviews, tags, geological and item information, etc, and transform explicit ratings to 0-1 implicit feedbacks.

Evaluation Metrics

We adopt two widely used evaluation metrics for personalized rankings: Recall and normalized discounted cumulative gain (nDCG) [128] to evaluate model performance. The recommendation list is generated by ranking unobserved items and truncating at position k. Recall measures the ratio of the relevant items in the recommendation list to all relevant items in test sets, while nDCG takes the rank into consideration by assigning higher scores to the relevant items ranked higher. We use 80% of the interactions for training and randomly select 10% from the training set as validation set for hyper-parameter tuning, the rest 20% data is used for test sets; we report the average accuracy on test sets.

Baselines

We compare DirectSpec with the following competitive baselines:

- BPR [45]: This is a stable and classic MF-based method, exploiting a Bayesian personalized ranking loss for personalized rankings.

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

- CL-Rec [148]: This is a CL-based method. Since there are no item features on the datasets used in this work, we remove data augmentation and use BPR as the main loss to better compare with our methods.
- DMF [150]: This is a neural network based method with interaction vectors as the input and BCE loss for optimization. Since multiple layers does not result in improvement on our datasets, we adopt a single-layer architecture.
- CCL [46]: The proposed cosine contrastive loss (CCL) maximizes the similarity between positive pairs and minimizes the similarity of negative pairs below the margin m. After parameter tuning, we set $\mathcal{N} = 1000$, $w = 300$ on all datasets, $m = 0.1$, 0.3, and 0.3 on CiteULike, Yelp, and Gowalla, respectively, and we choose MF as the encoder.
- DirectAU [115]: This method directly optimizes alignment and uniformity and shows superior performance. Following the original paper, we choose MF and LightGCN as the encoder, and set $\gamma = 5.0$, 0.5, and 1.5 on CiteULike, Yelp, and Gowalla, respectively.
- LightGCN [41]: This is a linear GNN method that only keeps neighborhood aggregation for recommendation. We employ a three-layer architecture as our baseline.
- SGL-ED [38]: This model explores self-supervised learning by maximizing the agreements of multiple views from the same node, where the node views are generated by adding noise such as randomly removing the edges or nodes on the original graph. We set $\tau = 0.2$, $\lambda_1 = 0.1$, $p = 0.1$, and use a three-layer architecture.
- GDE [96]: This method only keeps a very few graph features for recommendation without stacking layers, showing less complexity and higher efficiency than conventional GCN-based methods.

Implementation Details

We implemented our DirectSpec based on PyTorch and the code is released publicly[‡]. SGD is adopted as the optimizer for all models, the embedding size d is

[‡]https://github.com/tanatosuu/directspec

set to 64, the regularization rate is set to 0.01 on all datasets, the learning rate is tuned amongst $\{0.001, 0.005, 0.01, \cdots\}$; without specification, the model parameters are initialized with Xavier Initialization [131]; the batch size is set to 256. We report other hyperparameter settings in the next subsection.

6.5.2 Comparison

Performance

We report overall performance in Table 6.4, and observe the followings:

- Overall, GCN-based methods show better performance on sparse data (Yelp and Gowalla) than dense data (CiteULike). For instance, DirectSpec$^+$(GCN) achieves better and worse than DirectSpec$^+$(MF) on Gowalla and CiteULike, respectively, and their performance is close on Yelp. LightGCN underperforms BPR on CiteULike and Gowalla, which might be attributed to the slow convergence that has been reported in the original paper. Among baselines, DirectAU achieves the best on CiteULike and Yelp, while SGL-ED outperforms other baselines on Gowalla. Our proposed DirectSpec$^+$ implemented on both LightGCN and MF consistently show improvements over all baselines.

- CL-Rec is even inferior to BPR, indicating that it is hard for traditional CF task without any side information to benefit from CL. In the meanwhile, the GCL-based method (SGL-ED) works much better as we can exploit the graph structure containing rich topological information as the input, as opposed to traditional CF task that only uses user/item ID as input.

- DirectAu directly regulating the uniformity shows relatively superior performance. Inspired by [147], it adopts a Radial Basis Function (RBF) kernel which shares a similar form to the uniformity loss. Since we have shown that the uniformity loss can tackle embedding collapse by indirectly balancing the spectrum distribution in Section 6.4.4, the effectiveness of DirectAU demonstrates our previous analysis.

- Since our methods are implemented on MF and LightGCN, the effectiveness of our methods can be further demonstrated by directly comparing with them. DirectSpec(MF) and DirectSpec$^+$(MF) outperform BPR by 14.3% and 35.8%,

Table 6.4. Performance comparison. The best baseline is underlined. "$*$" indicates statistical significance at $p < 0.01$ for a one-tailed t-test.

Datasets	Methods	nDCG@10	nDCG@20	Recall@10	Recall@20
CiteULike	BPR	0.1620	0.1773	0.1778	0.2190
	DMF	0.1442	0.1640	0.1646	0.2183
	CL-REC	0.1523	0.1662	0.1671	0.2069
	CCL(MF)	0.1545	0.1642	0.1716	0.1996
	DirectAU(MF)	<u>0.2102</u>	<u>0.2252</u>	<u>0.2260</u>	<u>0.2693</u>
	DirectAU(GCN)	0.1926	0.2109	0.2116	0.2604
	LightGCN	0.1610	0.1781	0.1771	0.2190
	SGL-ED	0.1890	0.2065	0.2117	0.2588
	GDE	0.1890	0.2061	0.2055	0.2528
	DirectSpec(MF)	0.1688	0.1849	0.1827	0.2270
	DirectSpec(GCN)	0.1693	0.1863	0.1875	0.2334
	DirectSpec$^+$(MF)	**0.2197***	**0.2354***	**0.2352***	**0.2818***
	DirectSpec$^+$(GCN)	0.2038	0.2213	0.2213	0.2704
	p-value	2.51e-4	2.25e-5	1.17e-4	1.54e-5
Yelp	BPR	0.0487	0.0583	0.0607	0.0869
	DMF	0.0543	0.0649	0.0694	0.0986
	CL-REC	0.0476	0.0566	0.0588	0.0833
	CCF(MF)	0.0509	0.0593	0.0617	0.0846
	DirectAU(MF)	<u>0.0721</u>	<u>0.0848</u>	<u>0.0872</u>	<u>0.1219</u>
	DirectAU(GCN)	0.0695	0.0819	0.0854	0.1194
	LightGCN	0.0572	0.0690	0.0721	0.1045
	SGL-ED	0.0676	0.0794	0.0837	0.1166
	GDE	0.0653	0.0771	0.0805	0.1129
	DirectSpec(MF)	0.0689	0.0804	0.0839	0.1156
	DirectSpec(GCN)	0.0712	0.0832	0.0861	0.1192
	DirectSpec$^+$(MF)	0.0743	0.0864	**0.0909***	0.1249
	DirectSpec$^+$(GCN)	**0.0745***	**0.0868***	**0.0909***	**0.1252***
	p-value	3.37e-4	2.56e-3	3.17e-5	4.29e-4
Gowalla	BPR	0.1164	0.1255	0.1186	0.1483
	DMF	0.1121	0.1227	0.1186	0.1504
	CL-REC	0.1116	0.1198	0.1123	0.1401
	CCL(MF)	0.1269	0.1349	0.1295	0.1573
	DirectAU(MF)	0.1286	0.1402	0.1349	0.1710
	DirectAU(GCN)	0.1298	0.1409	0.1363	0.1711
	LightGCN	0.0987	0.1098	0.1074	0.1399
	SGL-ED	<u>0.1343</u>	<u>0.1462</u>	<u>0.1417</u>	<u>0.1779</u>
	GDE	0.1261	0.1367	0.1313	0.1656
	DirectSpec(MF)	0.1133	0.1225	0.1183	0.1480
	DirectSpec(GCN)	0.1160	0.1251	0.1191	0.1490
	DirectSpec$^+$(MF)	0.1389	0.1509	0.1447	**0.1829***
	DirectSpec$^+$(GCN)	**0.1400***	**0.1513***	**0.1453***	0.1819
	p-value	1.89e-8	1.70e-7	1.35e-7	1.54e-5

Table 6.5. Training time (seconds) per epoch.

Model/Data	Citeulike	Yelp	Gowalla
BPR	2.37	8.53	13.28
DirectSpec$^+$(MF)	3.78 (1.59x)	13.15 (1.54x)	19.10 (1.44x)
LigtGCN	8.78	43.18	79.79
DirectSpec$^+$(GCN)	9.49 (1.08x)	45.70 (1.06x)	81.20 (1.02x)
SGL-ED	22.20 (2.53x)	182.54 (4.23x)	401.96 (5.04x)

on average in terms of nDCG@10, respectively. In the meanwhile, the improvement of DirectSpec(GCN) and DirectSpec$^+$(GCN) over LightGCN is 15.7% and 32.9% on average in terms of nDCG@10, respectively.

- Overall, the improvement of DirectSpec over MF and LightGCN is Yelp> Gowalla>CiteULike (33.0%, 7.4%, and 4.7% on average in terms of nDCG@10, respectively). Figure 6.6 compares the extent of collapse on three datasets using MF and LightGCN (with only positive samples). We can observe that Yelp suffers more from the collapse while CiteULike suffers less, which is consistent with the improvement of DirectSpec. This observation reveals that the dataset suffering more from collapse tends to benefit more from DirectSpec.

- DirectSpec$^+$ shows significant improvement over DirectSpec across all datasets. As introduced in Section 6.4.2, DirectSpec$^+$ is more effective tackling embedding collapse than DirectSpec by effectively penalizing the hard negative samples with user/item pairs highly correlated. The superior performance demonstrates its effectiveness.

Efficiency

We report the training time per epoch of BPR, LightGCN, and DirectSpec$^+$ (MF and GCN) in Table 6.5. The experiments are all conducted on Intel(R) Core(TM) i9-10980XE CPU and NVIDIA RTX A6000 GPU. We first compare BPR and DirectSpec$^+$(MF), and observe that it takes roughly 0.5x additional running time, which is acceptable considering the significant improvement. DirectSpec$^+$(GCN) almost shows no additional running time over LightGCN, due to the reason that the complexity of DirectSpec$^+$ is mainly determined by the batch size rather than

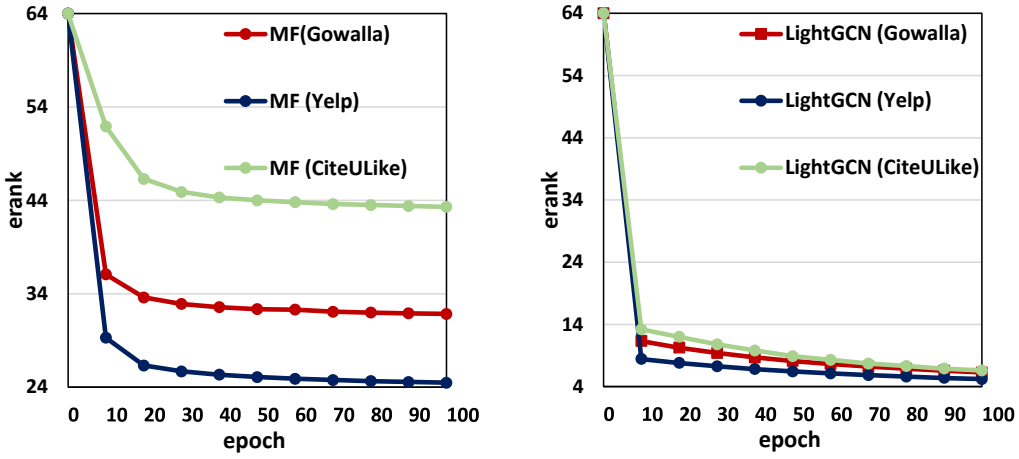
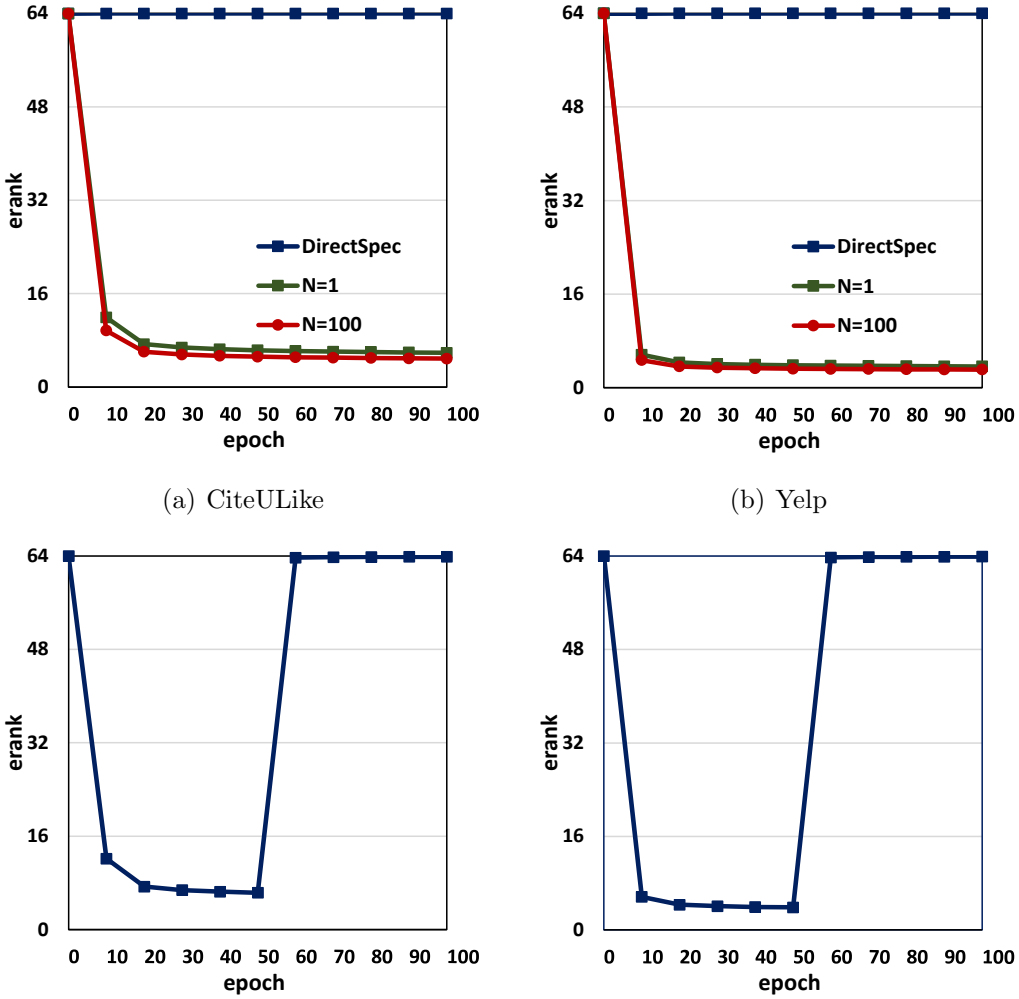


Figure 6.6. The extent of collapse on three datasets.

the size of users and items. Since LightGCN has a higher complexity than BPR, the proportion of DirectSpec$^+$(LightGCN)'s running time to that of LightGCN is much smaller than the proportion of DirectSpec$^+$(MF)'s running time to that of BPR. This finding can explain another observation that the additional time is smaller on larger datasets (*i.e.,* Gowalla<Yelp<CiteULike). The complexity of DirectSpec$^+$ is unchanged while LightGCN and BPR accordingly take more running time on larger datasets, thus the proportion of DirectSpec$^+$'s running time to that of LightGCN/BPR becomes smaller. Meanwhile, SGL-ED is more computationally expensive on larger datasets, and DirectSpec$^+$(LightGCN) is more efficient than it with more significant improvement over LightGCN.

Comparison with Negative Sampling

The results in Table 6.4 demonstrates the superiority of DirectSpec over negative sampling based methods including BPR and CCL in terms of performance. Furthermore, we compare how these two methods alleviate collapse issue by optimizing them solely based on unobserved interactions. In Figure 6.7 (a) and (b), we observe that the erank of MF (BCE) consistently drops without optimization on the observed interactions, resulting in a complete collapse, and the erank drops more quickly as increasing the negative sampling ratios, whereas the erank of DirectSpec remains unchanged throughout the training. This observation indicates that there exists a dynamic balance between optimization on observed and unobserved interactions, the lack of either of them cannot prevent the em-

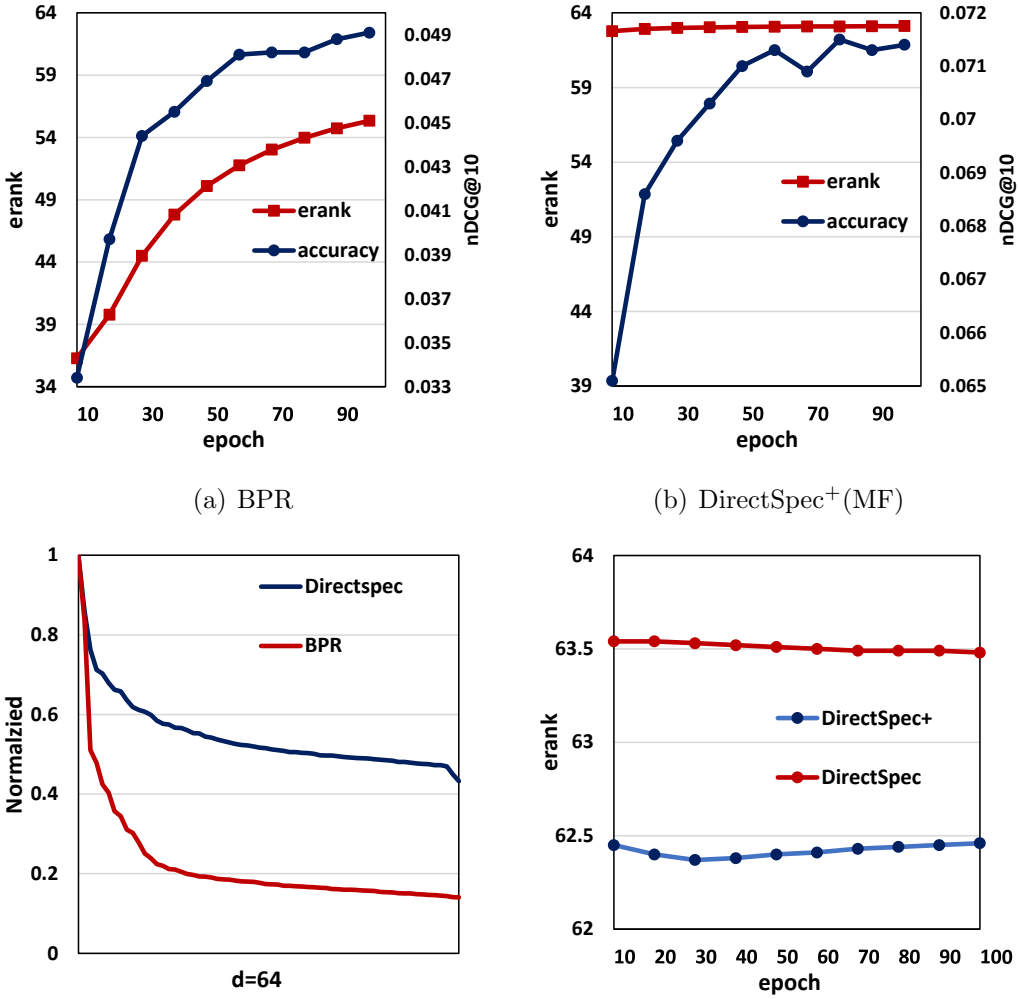


(a) CiteULike (b) Yelp

(c) CiteULike (d) Yelp

Figure 6.7. All models are optimized without using observed interactions. In (a) and (b), we show how erank changes on DirectSpec (MF) and MF (BCE) where N is the negative sampling ratio. In (c) and (d), we use BCE (MF) for the first 50 epochs and DirectSpec for the last 50 epochs.

beddings from collapsing. Furthermore, since negative sampling is equivalent to a high pass filter, it can only alleviate the collapse caused by low pass filters (*e.g.*, optimization on observed interactions), while DirectSpec is equivalent to an all pass filter that can tackle the collapse under any situations. Figure 6.7 (c) and (d) demonstrate that DirectSpec can recover the embeddings from collapsing caused by negative sampling.



(a) BPR

(b) DirectSpec^{+}(MF)

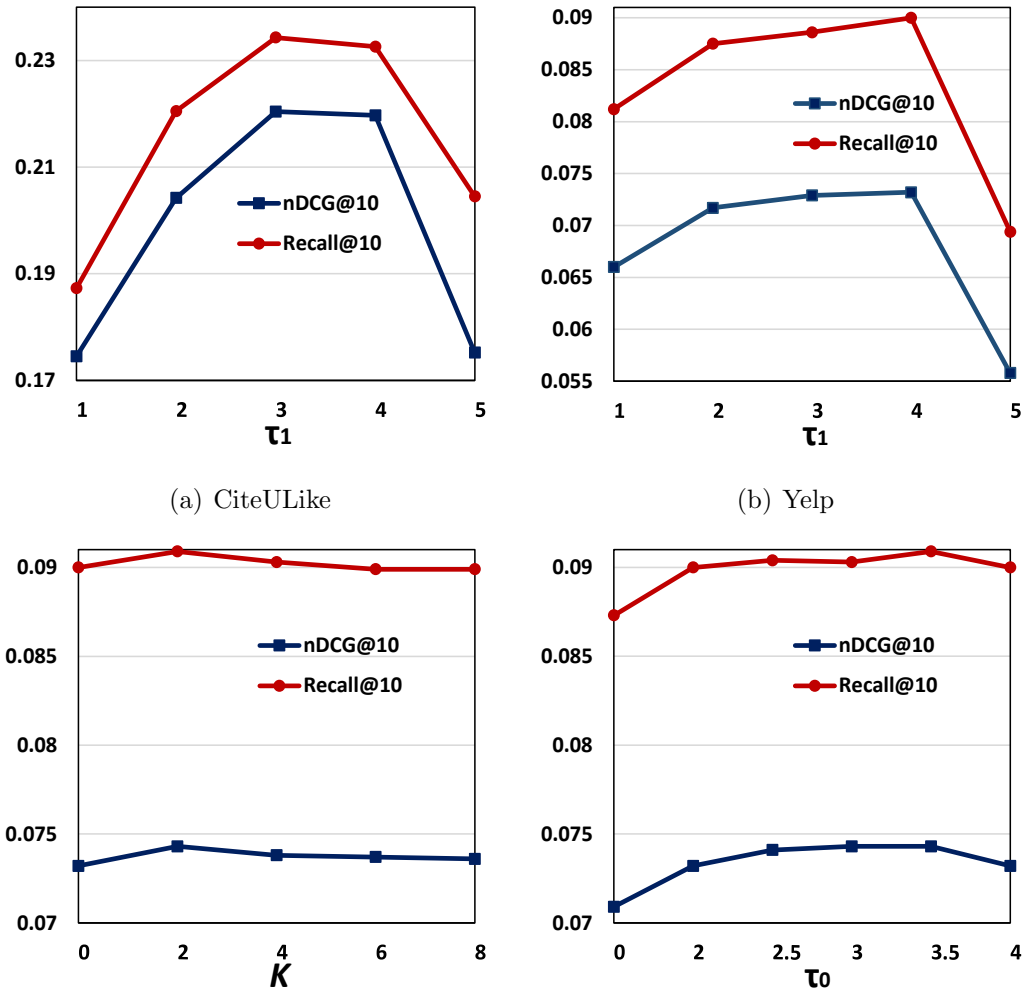
(c) Singular value distribution of the embedding matrix

(d) erank comparison

Figure 6.8. How DirectSpec prevents collapse (on Yelp).

Table 6.6. Comparison between static and dynamic temperature design.

	Citeulike		**Yelp**		**Gowalla**	
	nDCG@10	**Recall@10**	**nDCG@10**	**Recall@10**	**nDCG@10**	**Recall@10**
Static (i)	0.2197	0.2352	0.0743	0.0909	0.1389	0.1447
Dynamic (ii)	0.1593	0.1693	0.0485	0.0611	0.1126	0.1144



(a) CiteULike

(b) Yelp

(c) Yelp

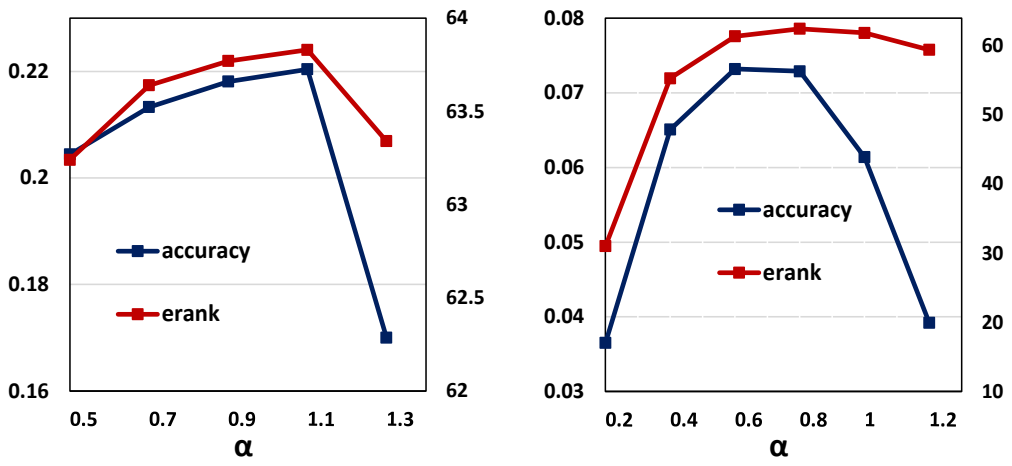
(d) Yelp

Figure 6.9. (a) and (b) show how the accuracy changes with τ_1, (c) displays the sensitivity of k, and (d) illustrates how the accuracy changes with τ_0.

6.5.3 Model Analysis

Can DirectSpec Prevent Collapse?

In Figure 6.8 (a) and (b), we report how the accuracy and erank change as training proceeds on BPR and DirectSpec$^+$(MF). For BPR, the erank increases slowly and tends to converge to a value smaller than d, resulting in an incomplete embedding collapse. While the erank of DirectSpec$^+$(MF) is close to d at the beginning of training and remains almost unchanged throughout the training. Figure 6.8



(a) CiteULike (b) Yelp

Figure 6.10. Impact of α on accuracy (nDCG@10) and erank.

(c) illustrates the normalized singular value distribution of the embedding matrix when the accuracy of BPR and DirectSpec$^+$(MF) is maximized. We can see that most singular values of BPR's embeddings are less than 0.2, barely contributing to the representations; while DirectSpec$^+$(MF)'s singular values are mostly larger than 0.5, indicating that the representations are contributed by more dimensions. One may raise a concern, that DirectSpec still suffers from embedding collapse as the erank is still slightly smaller than d ($d = 64 > erank(\mathbf{H}) > 62$ in our experiments) when the accuracy is maximized. As shown in Section 4.2, the goal of DirectSpec is equivalent to decorrelating users and items, while it is obvious that not all users/items are irrelevant, considering the relations of observed interactions (*e.g,* users share similar/opposite interests are positively/negatively related; items that are interacted by similar users or users that have opposite interests are also positively or negatively correlated, respectively). Thus, a higher erank does not necessarily lead to a better accuracy. From a spectral perspective, all dimensions are not completely equally important to users/item. Some dimensions might contain more important information while others contain less. In other words, the goal of DirectSpec is to assure that all dimensions literally contribute to the representations, as opposed to the representations of BPR or LightGCN that are only predominantly distributed along a few dimensions. In Figure 6.8 (d), DirectSpec seems to show a better ability of balancing spectrum as it has a higher erank, while it shows inferior performance to DirectSpec$^+$. As shown in

Equation (6.14), all samples are decorrelated with the same pace on DirectSpec, thus the similar or dissimilar (*i.e.,* sharing opposite interests) users/items that should be correlated to some extent are also perfectly orthogonalized, leading to underfitting.

Adaptive Temperature Design

We first compare the dynamic and static designs proposed in Section 4.3, and report the results in Table 6.6. We observe that the static design outperforms the dynamic design by a large margin on all datasets, and the reason might be attributed to the data scarcity on CF task, making the algorithm difficult to learn the inherent user/item relations from the interactions data in a dynamic way. Therefore, DirectSpec is implemented with a static temperature design, which avoids bringing additional complexity. We first tune τ_1 by fixing $\tau_0 = \tau_1$ and report how the accuracy changes with τ_1 in Figure 6.9 (a) and (b). The accuracy first increases and then drops as increasing τ_1, which is maximized at $\tau_1 = 3$ on CiteULike and $\tau_1 = 4$ on Yelp. τ_1 controls the strength of penalties on highly correlated unobserved samples (*i.e.,* hard negatives), the samples that are not highly correlated tend to be ignored when τ_1 is set too large, while a too small τ_1 fails to optimize the highly correlated samples in an effective way. On the other hand, among all unobserved interactions, some are correlated to some extent that should be decorrelated with a slower pace which we use an another hyperparameter τ_0 to control. In Figure 6.9 (c), we observe that optimizing the user/item pairs whose graph distance are $K = 2$ with a slower pace τ_0 results in a better accuracy, and the accuracy drops after introducing the pairs farther away (*i.e.,* $K > 2$) as they are too far away on the graph that should be considered irrelevant. In Figure 6.9 (d), DirectSpec achieves the best performance at $\tau_0 = 3.5$ which is slightly smaller than $\tau_1 = 4$ on Yelp (see Figure 6.9 (b)), and further decreasing $\tau_0 = 3.5$ leads to a worse performance. This finding indicates that the correlation between the pairs that are close on the graph are still significantly smaller than the observed interactions.

Impact of α

According to Equation (6.12), α controls the extent of spectrum balancing. A larger (smaller) α implies that large singular values are multiplied by smaller

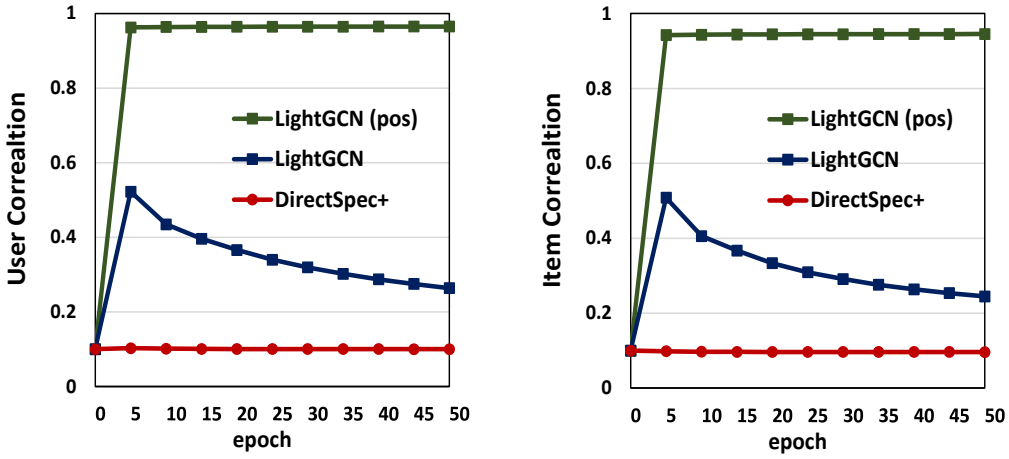


Figure 6.11. Correlations of users (left) and items (right) during the training.

(larger) weights, leading to a more (less) uniform spectrum distribution. Preposition 1 does not hold when α is too large: $1 - \alpha\sigma_1^2 < 0$ (with $L = 1$). Since it takes additional time to calculate σ_1 during each training, we use the grid search for optimal α. As shown in Figure 6.10 (a) and (b), both of the accuracy and erank first increase and then drop, showing a similar trend. Specifically, the speed of collapse tends to outrun the speed of spectrum balancing when α is small, and the drop in erank as increasing α demonstrates our analysis in Section 4.1 that a large α instead results in a sharp distribution. In addition, the best performance is achieved at $\alpha = 1.1$ on CiteULike and $\alpha = 0.6$ on Yelp, and the erank and accuracy are more sensitive when α is small on Yelp, which might be related to the fact that Yelp suffers more from the collapse issue than other datasets.

User/Item Correlation

As analyzed in Section 6.4.2, the user/item correlation can reflect the extent of embedding collapse, and the correlation here is measured by Equation (6.13) similar to Figure 6.5 (b). As shown in Figure 6.11, the correlation is low before training, while users/items become highly correlated immediately after training starts on LightGCN(pos) which only optimizes positive (*i.e.,* observed) pairs, causing user and items to have similar representations and resulting in a complete collapse. For LightGCN using BPR loss, the correlation increases fast first and then gradually drops, whereas the correlation keeps low throughout the training

on DirectSpec$^+$, demonstrating its effectiveness to tackle embedding collapse. Since GNN-based methods also suffer from the over-smoothing issue, the above observation also indicates that our proposed method can alleviate this issue very well. We also notice that the correlation did not converges to 0, due to the reason that $\mathbf{H}_U\mathbf{H}_U^T$ and $\mathbf{H}_I\mathbf{H}_I^T$ are full rank matrices when they are optimized to identity matrices (*i.e.,* completely decorrelated) according to Equation (6.13), while the embedding matrix is merely a low-rank approximation: $d \ll min(|\mathcal{U}|, |\mathcal{I}|)$, thus the correlation would not converge to 0 unless $d \geq \max(|\mathcal{U}|, |\mathcal{I}|)$.

6.6 Summary

In this chapter, we showed that existing CF methods mostly suffer from an embedding collapse issue. Particularly, optimization solely on the observed interactions is equivalent to a low pass filtering, causing the representations of users and items to collapse to a constant vector. Optimizing unobserved interactions (*i.e.,* negative sampling) can alleviate this issue by acting as a high pass filter to balance the embedding spectrum, while an incomplete collapse still exists where the embeddings are distributed along certain dimensions instead of making full use of all dimensions. We demonstrated that increasing the negative sampling ratio which is considered as an efficient way to improve the representation quality cannot further alleviate the collapse issue. To tackle this issue, we proposed a DirectSpec which acts as an all pass filter to directly balances the spectrum distribution to assure that all dimensions can contribute to the user/item embeddings as equally as possible. We conducted comprehensive analysis on DirectSpec from a decorrelation perspective and further proposed an enhanced variant DirectSpec$^+$ to efficiently penalize irrelevant samples with self-paced gradients. In addition, we shed light on the close connection between DirectSpec and contrastive learning, that uniformity, a key design contributing to recommendation, can alleviate collapse issue by indirectly balancing spectrum distribution, that can be considered as a special case of DirectSpec$^+$. Finally, extensive experiments on three publicly available datasets demonstrated the effectiveness and efficiency of our proposed methods. In future, we plan to focus on different perspectives to tackle embedding collapse to achieve much better efficiency and effectiveness.

CHAPTER 7

Conclusion

7.1 Conclusion

In this thesis, we investigated recommendation methods from a spectral perspective. Particularly, we focus on two topics: (1) demystifying the mechanisms of how graph convolutional networks (GCNs) work for recommendation and the designs of more efficient and effective GCN learning paradigm, and (2) investigating the factors contributing to high-quality representations that have remained relatively unexplored. The contribution can be summarized as follows:

- In Chapter 3, we showed how distinct spectral graph features contribute to the accuracy and found that only a very small fraction of spectral features that emphasize the neighborhood smoothness and difference are contributive to recommendation. We then unveiled the effectiveness of GCNs by showing that stacking layers in GCNs emphasizes the smoothness. Based on the two important findings above, we pointed out the limitations of existing GCN-based methods and proposed a Graph Denoising Encoder (GDE) equipped with a simple yet effective architecture. Finally, to tackle a slow convergence issue on GCN-based methods, we proposed an adaptive loss to dynamically adjust the gradients over negative samples, accelerating the model training and resulting in further improvement.

- In Chapter 4, we proposed a simplified and scalable GCN learning paradigm for CF. By simplifying LightGCN [41], 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 the 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.

- In Chapter 5, we unveiled the distribution redundancy of GCN-based methods by showing that the number of required spectral feature is closely related to the spectral distribution, where a dataset with a flatter distribution tends to requires more spectral features when reaching the best performance, resulting in more computational cost. We define a renormalized adjacency matrix with a hyper-parameter adjusting the sharpness of the spectral distribution to reduce the number of required spectral features, making the important information on the graph be concentrated in fewer features. By analyzing how graph contrastive learning (GCL) works for recommendation, we further proposed a scalable contrastive learning framework.

- In Chapter 6, we showed that existing CF methods mostly suffer from an embedding collapse issue. Particularly, optimization solely on the observed interactions causes the representations of users and items to collapse to a constant vector. Optimizing unobserved interactions (*i.e.,* negative sampling) can alleviate this issue by acting as a high pass filter to balance the embedding spectrum, while an incomplete collapse still exists where the embeddings are distributed along certain dimensions instead of making full use of all dimensions. To tackle this issue, we proposed a DirectSpec which acts as an all pass filter to directly balances the spectrum distribution to assure that all dimensions can contribute to the user/item embeddings as equally as possible. We conducted comprehensive analysis on DirectSpec from a decorrelation perspective and further proposed an enhanced variant DirectSpec$^+$ to efficiently penalize irrelevant samples with self-paced gradients.

7.2 Future Work

Firstly, the approaches proposed in this thesis are mainly based on collaborative filtering (CF). We aim to extend our proposed methods to other recommendation tasks, which is not easy as the research findings made on CF does not necessarily hold on other tasks. Therefore, we will need to modify and improve our algorithms to adapt to other recommendation tasks. Secondly, in the light of great potential of graph neural networks (GNN), we will continue to improve GNN-based recommendation methods. Particularly, we focus on the scalability and efficiency of the model architecture considering the expensive complexity of GNNs compared with traditional recommendation methods. In addition, spectrum contains important information of graphs and embeddings which is relatively unexplored by existing works. We will keep improving recommendation methods by empirically and theoretically studying the spectrum information.

Acknowledgements

Time flies. I feel like I just walked into Yoshida campus of Kyoto university for the first time yesterday while now I have reached the end of my Ph.D. journey. Looking back on the path I have walked, I realized I could not have come all the way here without the support and help of so many people.

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Kazunari Sugiyama for guiding me throughout my doctoral studies. It was him who continuously encouraged me and taught me how to write solid research papers when my submissions were ruthlessly rejected time after time. I have learnt a lot from his passion and diligence towards research and education.

I would like to thank my advisors Profs. Keishi Tajima and Takayuki Ito for their constructive advices and suggestions to help improve my research works.

Moreover, I want to thank my collaborators Assoc. Prof. Tsunenori Mine and Dr. Xin Liu. Assoc. Prof. Mine is my supervisor when I was in Kyushu university pursing my master degree. After I went to Kyoto university, he still generously provided computational resource for me. Dr. Liu shares a lot of similarity with me on research interests, I always enjoy the research discussion with him and benefit a lot from his valuable advices.

Lastly, I would like to extend my appreciation to my family, especially my parents Shujie Peng, Xiaomei Wu, and my beloved girl friend Xinyang Liang, for their unconditional love and support.

Shaowen Peng, February 2024

References

[1] Bhavik Pathak, Robert Garfinkel, Ram D Gopal, Rajkumar Venkatesan, and Fang Yin. Empirical analysis of the impact of recommender systems on sales. *Journal of Management Information Systems*, pages 159–188, 2010.

[2] Zhijie Lin. An empirical investigation of user and system recommendations in e-commerce. *Decision Support Systems*, 68:111–124, 2014.

[3] Mohammad Yahya H Al-Shamri. User profiling approaches for demographic recommender systems. *Knowledge-Based Systems*, 100:175–187, 2016.

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

[5] Yang Bao, Hui Fang, and Jie Zhang. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-14)*, pages 2–8, 2014.

[6] Karen HL Tso-Sutter, Leandro Balby Marinho, and Lars Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC'08)*, pages 1995–1999, 2008.

[7] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th conference on Uncertainty in artificial intelligence*, pages 43–52, 1998.

[8] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW'01)*, pages 285–295, 2001.

[9] Chumki Basu, Haym Hirsh, William Cohen, et al. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, pages 714–720, 1998.

[10] Mark O'Connor and Jon Herlocker. Clustering items for collaborative filtering. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems*, 1999.

[11] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[12] Michelle Keim Condliff, David D Lewis, David Madigan, and Christian Posse. Bayesian mixed-effects models for recommender systems. In *Proceedings of the 22nd International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR'99)*, pages 23–30.

[13] Prem Melville, Raymond J Mooney, Ramadass Nagarajan, et al. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-02)*, pages 187–192, 2002.

[14] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, pages 313–331, 1997.

[15] David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI'00)*, pages 473–480, 2000.

[16] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In

Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS'16), pages 7–10, 2016.

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

[18] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining (WSDM'17)*, pages 495–503, 2017.

[19] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. Sequential user-based recurrent neural network recommendations. In *Proceedings of the 11th ACM Conference on Recommender Systems (RecSys'17)*, pages 152–160, 2017.

[20] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 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)*, pages 335–344, 2017.

[21] Shang Liu, Zhenzhong Chen, Hongyi Liu, and Xinghai Hu. User-video co-attention network for personalized micro-video recommendation. In *Proceedings of the 28th international conference on World Wide Web (WWW'19)*, pages 3020–3026, 2019.

[22] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM'19)*, pages 1441–1450, 2019.

[23] Chaoliu Li, Lianghao Xia, Xubin Ren, Yaowen Ye, Yong Xu, and Chao Huang. Graph transformer for recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'23)*, pages 1680–1689, 2023.

[24] Nouhaila Idrissi and Ahmed Zellou. A systematic literature review of sparsity issues in recommender systems. *Social Network Analysis and Mining*, 10:1–23, 2020.

[25] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)*, 2017.

[26] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *Proceedings of the 8th International Conference on Learning Representations (ICLR'20)*, 2020.

[27] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining (SDM'19)*, pages 594–602. SIAM, 2019.

[28] Yuyang Wang, Jianren Wang, Zhonglin Cao, and Amir Barati Farimani. Molecular contrastive learning of representations via graph neural networks. *Nature Machine Intelligence*, 4(3):279–287, 2022.

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

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

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*, pages 770–778, 2016.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*, 2017.

[33] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'13)*, pages 6645–6649, 2013.

[34] Sheng Li, Jaya Kawale, and Yun Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM'15)*, pages 811–820, 2015.

[35] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 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)*, pages 235–244, 2019.

[36] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys'18)*, pages 297–305, 2018.

[37] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI-19)*, pages 346–353, 2019.

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

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

[40] Jianing Sun, Zhaoyue Cheng, Saba Zuberi, Felipe Pérez, and Maksims Volkovs. Hgcf: Hyperbolic graph convolution networks for collaborative

filtering. In *Proceedings of the 30th International Conference on World Wide Web (WWW'21)*, pages 593–601, 2021.

[41] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*, pages 639–648, 2020.

[42] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 3538–3545, 2018.

[43] Fan RK Chung. *Spectral Graph Theory*, volume 92. American Mathematical Soc., 1997.

[44] Aliaksei Sandryhaila and Jose MF Moura. Discrete signal processing on graphs: Frequency analysis. *IEEE Transactions on Signal Processing*, 62(12):3042–3054, 2014.

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

[46] Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. Simplex: A simple and strong baseline for collaborative filtering. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM'21)*, pages 1243–1252, 2021.

[47] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.

[48] John Canny. Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR'02)*, pages 238–245, 2002.

[49] Xiaoyuan Su and Taghi M Khoshgoftaar. Collaborative filtering for multi-class data using belief nets algorithms. In *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, pages 497–504, 2006.

[50] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. Sorec: Social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and Knowledge Management (CIKM'08)*, pages 931–940, 2008.

[51] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. Geomf: Joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)*, pages 831–840, 2014.

[52] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR'02)*, pages 253–260, 2002.

[53] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems (TOIS)*, 41(3):1–39, 2023.

[54] Arjan JP Jeckmans, Michael Beye, Zekeriya Erkin, Pieter Hartel, Reginald L Lagendijk, and Qiang Tang. Privacy in recommender systems. *Social Media Retrieval*, pages 263–281, 2013.

[55] Steffen Rendle. Factorization machines. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM'10)*, pages 995–1000, 2010.

[56] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[57] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the*

27th International Conference on World Wide Web (WWW'18), pages 689–698, 2018.

[58] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *Proceedings of the 18th IEEE International Conference on Data Mining (ICDM'18)*, pages 197–206, 2018.

[59] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 27th International Conference on World Wide Web (WWW'18)*, pages 167–176, 2018.

[60] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web (WWW'15)*, pages 111–112, 2015.

[61] Balázs Hidasi, Alexandros Karatzoglou, et al. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[62] Lei Zheng, Vahid Noroozi, and Philip S Yu. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the 10th International Conference on Web Search and Data Mining (WSDM'17)*, pages 425–434, 2017.

[63] Wei Niu, James Caverlee, and Haokai Lu. Neural personalized ranking for image recommendation. In *Proceedings of the 11th International Conference on Web Search and Data Mining (WSDM'18)*, pages 423–431, 2018.

[64] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *Proceedings of the 11th ACM Conference on Recommender Systems (RecSys'17)*, pages 297–305, 2017.

[65] Lei Li, Yongfeng Zhang, and Li Chen. Personalized transformer for explainable recommendation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL'21)*, pages 4947–4957, 2021.

[66] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys'19)*, pages 101–109, 2019.

[67] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR'14)*, 2014.

[68] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*, pages 3844–3852, 2016.

[69] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*, 2018.

[70] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*, pages 1024–1034, 2017.

[71] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proceedings of the 7th International Conference on Learning Representations (ICLR'19)*, 2019.

[72] Hoang NT and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

[73] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML'19)*, pages 6861–6871, 2019.

[74] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the 35th*

AAAI Conference on Artificial Intelligence (AAAI-21), pages 3950–3957, 2021.

[75] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *Proceedings of the 9th International Conference on Learning Representations (ICLR'21)*, 2021.

[76] Muhammet Balcilar, Renton Guillaume, et al. Analyzing the expressive power of graph neural networks in a spectral perspective. In *Proceedings of the 9th International Conference on Learning Representations (ICLR'21)*, 2021.

[77] Mingguo He, Zhewei Wei, Hongteng Xu, et al. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. In *Proceedings of the 35th International Conference on Neural Information Processing Systems (NeurIPS'21)*, volume 34, pages 14239–14251, 2021.

[78] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 44(7):3496–3507, 2022.

[79] Xiyuan Wang and Muhan Zhang. How powerful are spectral graph neural networks. In *Proceedings of the 39th International Conference on Machine Learning (ICML'22)*, pages 23341–23362, 2022.

[80] Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, B Khaled Letaief, and Dongsheng Li. How powerful is graph convolution for recommendation? In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM'21)*, pages 1619–1629, 2021.

[81] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'19)*, pages 257–266, 2019.

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

[83] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*, pages 169–178, 2020.

[84] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, and Tat-Seng Chua. Graph-refined convolutional network for multimedia recommendation with implicit feedback. In *Proceedings of the 28th ACM International Conference on Multimedia (MM'20)*, pages 3541–3549, 2020.

[85] Linmei Hu, Chen Li, Chuan Shi, Cheng Yang, and Chao Shao. raph neural news recommendation with long-term and short-term interest modeling. *Information Processing & Management*, 57(2):102142, 2020.

[86] Suyu Ge, Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. Graph enhanced representation learning for news recommendation. In *Proceedings of the 29th international conference on World Wide Web (WWW'20)*, pages 2863–2869, 2020.

[87] Kartik Sharma, Yeon-Chang Lee, Sivagami Nambi, Aditya Salian, Shlok Shah, Sang-Wook Kim, and Srijan Kumar. A survey of graph neural networks for social recommender systems. *arXiv preprint arXiv:2212.04481*, 2022.

[88] Mengru Chen, Chao Huang, Lianghao Xia, Wei Wei, Yong Xu, and Ronghua Luo. Heterogeneous graph contrastive learning for recommendation. In *Proceedings of the 16th ACM International Conference on Web Search and Data Mining (WSDM'23)*, pages 544–552, 2023.

[89] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM'18)*, pages 417–426, 2018.

[90] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'19)*, pages 950–958, 2019.

[91] Wenhui Yu and Zheng Qin. Graph convolutional network for recommendation with low-pass collaborative filters. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*, pages 10936–10945, 2020.

[92] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Proceedings of the 20th International Conference on Neural Information Processing Systems (NIPS'06)*, pages 1601–1608, 2006.

[93] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 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)*, pages 974–983, 2018.

[94] Muhan Zhang and Yixin Chen. Inductive matrix completion based on graph neural networks. In *Proceedings of the 8th International Conference on Learning Representations (ICLR'20)*, 2020.

[95] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI-20)*, number 01, pages 27–34, 2020.

[96] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. 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)*, pages 1273–1282, 2022.

[97] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep Graph Infomax. In *Proceedings of the 7th International Conference on Learning Representations (ICLR'19)*, 2019.

[98] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS'20)*, pages 5812–5823, 2020.

[99] Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. Data augmentation for deep graph learning: A survey. *ACM SIGKDD Explorations Newsletter*, 24(2):61–77, 2022.

[100] Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. 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)*, pages 665–674, 2021.

[101] Lianghao Xia, Chao Huang, Jiao Shi, and Yong Xu. Graph-less collaborative filtering. In *Proceedings of the 32nd International Conference on World Wide Web (WWW'23)*, pages 17–27, 2023.

[102] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *Proceedings of the 7th International Conference on Learning Representations (ICLR'19)*, 2019.

[103] Minghao Zhao, Le Wu, Yile Liang, Lei Chen, Jian Zhang, Qilin Deng, Kai Wang, Xudong Shen, Tangjie Lv, and Runze Wu. Investigating accuracy-novelty performance for graph-based collaborative filtering. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'22)*, pages 50–59, 2022.

[104] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS'12)*, volume 25, 2012.

[105] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations (ICLR'13)*, 2013.

[106] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *Proceedings of the 7th International Conference on Learning Representations (ICLR'19)*, 2019.

[107] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'21)*, pages 15750–15758, 2021.

[108] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*, pages 1597–1607, 2020.

[109] Tianyu Hua, Wenxiao Wang, Zihui Xue, Sucheng Ren, Yue Wang, and Hang Zhao. On feature decorrelation in self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR'21)*, pages 9598–9608, 2021.

[110] Jean-Bastien Grill, Florian Strub, Florent Altché, Tallec, et al. Bootstrap your own latent a new approach to self-supervised learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS'20)*, pages 21271–21284, 2020.

[111] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*, pages 12310–12320, 2021.

[112] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. In *Proceedings of the 10th International Conference on Learning Representations (ICLR'22)*, 2022.

[113] Yifei Zhang, Hao Zhu, Zixing Song, Piotr Koniusz, and Irwin King. Spectral feature augmentation for graph contrastive learning and beyond. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI-23)*, pages 11289–11297, 2023.

[114] Xiaojun Guo, Yifei Wang, Tianqi Du, and Yisen Wang. Contranorm: A contrastive learning perspective on oversmoothing and beyond. In *Proceedings of the 11th International Conference on Learning Representations (ICLR'23)*, 2023.

[115] Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. Towards representation alignment and uniformity in collaborative filtering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'22)*, pages 1816–1825, 2022.

[116] Bingchao Wu, Yangyuxuan Kang, Bei Guan, and Yongji Wang. We are not so similar: Alleviating user representation collapse in social recommendation. In *Proceedings of the 2023 ACM International Conference on Multimedia Retrieval (ICMR'23)*, pages 378–387, 2023.

[117] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys'16)*, pages 191–198, 2016.

[118] Travis Ebesu, Bin Shen, and Yi Fang. Collaborative memory network for recommendation systems. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR'18)*, pages 515–524, 2018.

[119] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *Proceedings of the 28th International Conference on World Wide Web (WWW'19)*, pages 417–426, 2019.

[120] Yonghui Yang, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. Enhanced graph learning for collaborative filtering via mutual information maximization. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'21)*, pages 71–80, 2021.

[121] Pouria Ramazi, James Riehl, and Ming Cao. Homophily, heterophily and the diversity of messages among decision-making individuals. *Royal Society open science*, 5(4):180027, 2018.

[122] Haohan Wang, Xindi Wu, Zeyi Huang, and Eric P Xing. High-frequency component helps explain the generalization of convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'20)*, pages 8684–8694, 2020.

[123] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI-19)*, pages 3558–3565, 2019.

[124] Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'20)*, pages 338–348, 2020.

[125] Johan Paratte and Lionel Martin. Fast eigenspace approximation using random signals. *arXiv preprint arXiv:1611.00938*, 2016.

[126] Andreas Stathopoulos and Kesheng Wu. A block orthogonalization procedure with constant synchronization requirements. *SIAM Journal on Scientific Computing*, 23(6):2165–2182, 2002.

[127] Steffen Rendle and Christoph Freudenthaler. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM'14)*, pages 273–282, 2014.

[128] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

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

[130] Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM'11)*, pages 497–506, 2011.

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

[132] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. Svd-gcn: A simplified graph convolution paradigm for recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM'22)*, pages 1625–1634, 2022.

[133] Shanshan Feng, Gao Cong, Bo An, and Yeow Meng Chee. Poi2vec: Geographical latent representation for predicting future visitors. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 102–108, 2017.

[134] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 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)*, pages 515–524, 2017.

[135] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. Ultragcn: Ultra simplification of graph convolutional networks for recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM'21)*, pages 1253–1262, 2021.

[136] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in {gnn}s. In *Proceedings of the 8th International Conference on Learning Representations (ICLR'20)*, 2020.

[137] Risi Imre Kondor and John D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the 19th International Conference on Machine Learning (ICML'02)*, pages 315–322, 2002.

[138] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[139] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):1–19, 2015.

[140] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 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)*, pages 549–558, 2016.

[141] Daniel Spielman. Spectral graph theory. *Combinatorial scientific computing*, 18, 2012.

[142] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. Less is more: Removing redundancy of graph convolutional networks for recommendation. *ACM Trans. Inf. Syst.*, 42(3), jan 2024.

[143] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. Knowledge graph contrastive learning for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'22)*, pages 1434–1443, 2022.

[144] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'18)*, pages 839–848, 2018.

[145] Wenjie Wang, Yiyan Xu, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. Diffusion recommender model. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'23)*, pages 832–841, 2023.

[146] Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European Signal Processing Conference*, pages 606–610, 2007.

[147] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceed-*

ings of the 37th International Conference on Machine Learning (ICML'20), pages 9929–9939, 2020.

[148] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix Yu, Ting Chen, Aditya Menon, Lichan Hong, Ed H. Chi, Steve Tjoa, Jieqi (Jay) Kang, and Evan Ettinger. Self-supervised learning for large-scale item recommendations. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM'21)*, pages 4321–4330, 2021.

[149] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'22)*, pages 1294–1303, 2022.

[150] Hong-Jian Xue, Xinyu Dai, et al. Deep matrix factorization models for recommender systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17)*, pages 3203–3209, 2017.

Selected List of Publications

- **Journals**

  [1] Shaowen Peng, Kazunari Sugiyama, and Tsuneori Mine. Less is More: Removing Redundancy of Graph Convolutional Networks for Recommendation. *ACM Transactions on Information Systems (TOIS), 42(3), pages 85:1-85:26, 2024.* https://dl.acm.org/doi/full/10.1145/3632751

  [2] Shaowen Peng, Kazunari Sugiyama, Xin Liu, and Tsuneori Mine. Balancing Embedding Spectrum for Recommendation. *(TOIS) (Under Review).*

- **International Conferences and Workshops**

  [3] Shaowen Peng, Kazunari Sugiyama, and Tsuneori Mine. Less is More: Reweighting Important Spectral Graph Features for Recommendation. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1273-1282, July 2022. https://dl.acm.org/doi/10.1145/3477495.3532014

  [4] Shaowen Peng, Kazunari Sugiyama, and Tsuneori Mine. SVD-GCN: A Simplified Graph Convolution Paradigm for Recommendation. *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, 1625–1634, October 2022. https://dl.acm.org/doi/10.1145/3511808.3557462