Revisiting LightGCN: Unexpected Inflexibility, Inconsistency, and A Remedy Towards Improved Recommendation (Supplementary Document)

Anonymous Author(s)

1 ADDITIONAL BACKGROUND

In this section, we introduce embedding-based recommendation and neural networks for it. Refer to Table 1 for a symbol table.

1.1 Embedding-based Recommendation

Given a set $\mathcal{U} = \{u_1, \cdots, u_{|\mathcal{U}|}\}$ of users, a set $\mathcal{I} = \{i_1, \cdots, i_{|\mathcal{I}|}\}$ of items, and a user-item interaction matrix $\mathbf{R} \in \{0,1\}^{|\mathcal{U}| \times |\mathcal{I}|}$ where $r_{ui} \in \mathbf{R}$ is 1 if user u purchased item i and 0 otherwise, our objective is to infer the ground-truth preference of each user for each unpurchased item and subsequently recommend items based on the preference. In embedding-based recommendation, each user u and item i are encoded into embedding vectors $\mathbf{e}_u \in \mathbb{R}^d$ and $\mathbf{e}_i \in \mathbb{R}^d$, respectively, within a shared d-dimensional latent space. These embeddings are used to estimate user u's preference for item i through the inner product, i.e., $\hat{r}_{ui} = \mathbf{e}_u^T \mathbf{e}_i$.

1.2 Graph Neural Networks for Recommendation

Graph Neural Networks (GNNs) have emerged as a powerful tool for embedding-based recommendation [3, 7, 10]. The interaction data $\mathbf{R}$ is modeled as a user-item bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ and $\mathcal{E} = \{(u,i) \in \mathcal{U} \times \mathcal{I} : r_{ui} = 1\}$. Throughout the paper, we use $\mathcal{N}_v = \{x \in \mathcal{V} : (v,x) \in \mathcal{E} \text{ or } (x,v) \in \mathcal{E}\}$ to denote the set of neighbors of each node $v \in \mathcal{V}$ and define the degree of v as $|\mathcal{N}_v|$. Then, a two-stage process is employed to derive the embedding of each node u as:

$$\mathbf{e}_u^{(k)} = f_{\text{aggregate}}\left(\{\mathbf{e}_v^{(k-1)} : v \in \mathcal{N}_u \cup \{u\}\}\right), \forall k \in \{1, \cdots, K\},$$

$$\mathbf{e}_u = f_{\text{pool}}\left(\mathbf{e}_u^{(0)}, \cdots, \mathbf{e}_u^{(K)}\right),$$

where K is the predefined number of GNN layers, and the embedding $\mathbf{e}_u^{(0)}$ in the initial layer comprises learnable parameters that are randomly initialized. For each node u, the function $f_{\text{aggregate}}(\cdot)$ aggregates the prior-layer embeddings of the node itself and its neighbors. After propagating across K layers, the function $f_{\text{pool}}(\cdot)$ combines the embeddings derived at each layer, $\mathbf{e}_u^{(0)}, \cdots, \mathbf{e}_u^{(K)}$. The combined embedding is used for making predictions. Here, we categorize the embeddings $\mathbf{e}_u^{(k)}$ into *aggregation-free* (*agg-free*) for $k = 0$ and *aggregation-based* (*agg-based*) for $k \geq 1$, depending on whether neighbor aggregation is employed in their derivation.

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

Conference'17, July 2017, Washington, DC, USA

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1: Frequently-used notations in this paper.

Notation	Definition
$\mathbf{e}^{(0)}$	aggregation-free (agg-free) embeddings
$\mathbf{e}^{(k)}$ $(k \geq 1)$	aggregation-based (agg-based) embeddings
$\|\mathbf{e}\|$	L2 norm of embedding $\mathbf{e}$
$\propto\!\!\!\!\sim$	strong positive linear correlation
α	hyperparameter that adjusts embedding norms
β	hyperparameter that adjusts neighbor weights
γ	ratio between agg-free and agg-based embeddings

Table 2: The Pearson correlation between the norm of the unscaled aggregated embedding $\left\| \sum_{u \in \mathcal{N}_i} |\mathcal{N}_u|^{-0.5} \mathbf{e}_u^{(k)} \right\|$ and the number of neighbors $|\mathcal{N}_i|$ is high for $k \geq 0$ (Observation 1).

Dataset	$k=0$	$k=1$	$k=2$	$k=3$
LastFM	0.9944	0.9800	0.9986	0.9909
MovieLens	0.9829	0.9433	0.9993	0.9984
Gowalla	0.9799	0.9429	0.9747	0.9316
Yelp	0.9905	0.9086	0.9956	0.9014
Amazon	0.9831	0.9289	0.9852	0.9334

2 FURTHER ANALYSIS OF LIGHTGCN

In this section, we provide additional results on our analysis of LightGCN discussed in the main paper. The LightGCN's neighbor aggregation rule for each item i can be written as :

$$\mathbf{e}_i^{(k+1)} = \frac{1}{\sqrt{|\mathcal{N}_i|}} \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)}. \tag{1}$$

2.1 Further Details on Observation 1

Observation 1. *Empirically, the norm of the unscaled aggregated neighbor embeddings in Eq.* (1) *at each k^{th} $(k \geq 0)$ layer exhibits a near linear relationship with the number of neighbors. Specifically,*

$$\left\| \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)} \right\| \propto\!\!\!\!\sim |\mathcal{N}_i|, \tag{2}$$

where the symbol $\propto\!\!\!\!\sim$ denotes a strong positive linear correlation, with a high Pearson correlation coefficient.

Additional observational results. In Figure 2, we present visualizations of Observation 1 across five datasets. These results empirically support our observation that the norm of the unscaled aggregated embedding exhibits a near linear relationship with the number of neighbors, i.e., Eq. (2), for all $k \in \{0, \cdots, K\}$. In addition, in Table 2, we measure the Pearson correlation coefficient between $\left\| \sum_{u \in \mathcal{N}_i} |\mathcal{N}_u|^{-0.5} \mathbf{e}_u^{(k)} \right\|$ and $|\mathcal{N}_i|$, which reveals a high correlation across all layers and datasets.

Table 3: The Pearson correlation between the norm of the scaled aggregated embedding $\|\mathbf{e}_i^{(k)}\|$ and $\sqrt{|\mathcal{N}_i|}$ is observed to be high for $k \geq 1$ (Property 1). However, this correlation is low (i.e., uncorrelated) for $k = 0$ (Property 2).

Dataset	$k=0$	$k=1$	$k=2$	$k=3$
LastFM	0.3459	**0.9509**	**0.7545**	**0.9802**
MovieLens	-0.2476	**0.9372**	**0.8164**	**0.9982**
Gowalla	-0.1515	**0.8266**	**0.7132**	**0.8283**
Yelp	-0.0115	**0.9223**	**0.6778**	**0.9778**
Amazon	0.1190	**0.8867**	**0.7442**	**0.9277**

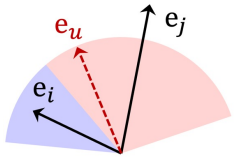


Figure 1: Given two item embeddings, $\mathbf{e}_i$ and $\mathbf{e}_j$, for item i and item j, respectively, along with an embedding $\mathbf{e}_u$ for user u, the inner product $\mathbf{e}_j^T\mathbf{e}_u$ is generally expected be greater than $\mathbf{e}_i^T\mathbf{e}_u$ if $\|\mathbf{e}_j\| > \|\mathbf{e}_i\|$. The red and blue shared areas represent the regions where the inner product is larger for item j and item i, respectively, depending on the location of $\mathbf{e}_u$.

2.2 Why Embedding Norms Matter

We highlight the importance of norm scaling in personalized recommendations. Specifically, the relevance score of item i for user u is computed through the inner product $\hat{r}_{ui} = \mathbf{e}_u^T\mathbf{e}_i$ of the respective embeddings which can be expressed as:

$$\hat{r}_{ui} = \mathbf{e}_u^T\mathbf{e}_i = \|\mathbf{e}_u\|\|\mathbf{e}_i\|\cos(\theta_{ui})$$

where θ_{ui} denotes the angle between $\mathbf{e}_u$ and $\mathbf{e}_i$. Note that the norm of the user embedding does not influence the ranking of recommendations for a given user. Consequently, the ranking is primarily affected by the norm of the item embedding $\mathbf{e}_i$ and the angle θ_{ui} between $\mathbf{e}_u$ and $\mathbf{e}_i$. As illustrated in Figure 1, the embedding $\mathbf{e}_j$ of item j, which has a larger norm compared to the embedding $\mathbf{e}_i$ of item i (i.e., $\|\mathbf{e}_j\| > \|\mathbf{e}_i\|$), is more likely to exhibit larger inner product with an arbitrary embedding $\mathbf{e}_u$ of user u.

2.3 Detailed Observations

In this subsection, we provide our analysis of all datasets, including those not covered in the main paper.

Properties of LightGCN. We uncover unexpected inflexibility and inconsistency in the embedding behavior of LightGCN, and here are the properties.

Property 1. *Empirically, the norms of the agg-based embedding $\mathbf{e}_i^{(k)}$'s at each k^{th} ($k \geq 1$) layer tend to satisfy:*

$$\|\mathbf{e}_i^{(k)}\| \propto \sqrt{|\mathcal{N}_i|}.$$

This relationship symmetrically applies to the agg-based embeddings of each user u, i.e., $\|\mathbf{e}_u^{(k)}\| \propto \sqrt{|\mathcal{N}_u|}$ for $k \geq 1$.

Property 2. *Empirically, the norms of the agg-free embedding $\mathbf{e}_i^{(0)}$'s do **NOT** exhibit a strong linear correlation with $\sqrt{|\mathcal{N}_i|}$'s, in contrast to agg-based embeddings (Property 1).*

Property 3 (Near-uniform Effective Weights). *Empirically, effective weights at each k^{th} ($k \geq 1$) layer tend to be near uniform across neighboring users, i.e.,*

$$\|\mathbf{e}_u^{(k)}\|/\sqrt{|\mathcal{N}_u|} \propto \sqrt{|\mathcal{N}_u|}/\sqrt{|\mathcal{N}_u|} = 1$$

Property 4. *Empirically, for $k = 0$, effective weights tend to decrease with respect to the degrees of neighbors.*

Property 5. *Agg-free ($k = 0$) and agg-based embeddings ($1 \leq k \leq K$) are added with a specific weight ratio of $1 : K$.*

Norm scaling of LightGCN. In Figure 3, we present the visualizations that validate Properties 1 and 2 of LightGCN across five datasets. We can visually confirm that, empirically, the norm of the scaled aggregated embedding $\mathbf{e}_i^{(k)}$ at each k^{th} ($k \geq 1$) layer tends to satisfy $\|\mathbf{e}_i^{(k)}\| \propto \sqrt{|\mathcal{N}_i|}$ (Property 1) while this relationship does not hold when $k = 0$ (Property 2). This is numerically validated in Table 3, where the Pearson correlation coefficients are high when $k \geq 1$, but are low when $k = 0$.

Neighbor weighting of LightGCN. In Figure 4, we present the visualizations that empirically validate Properties 3 and 4 of LightGCN across five datasets. We can visually confirm that the effective weight $\|\mathbf{e}_u^{(k)}\|/\sqrt{|\mathcal{N}_u|}$ at each k^{th} ($k \geq 1$) layer tend to be near uniform across neighboring users (Property 3) while when $k = 0$, it tends to decrease with the degree of the neighbor (Property 4).

3 FURTHER DETAILS ON LIGHTGCN++

In this section, we offer additional information about LightGCN++ to provide a deeper understanding of its mechanisms.

3.1 Generalized Norm Scaling

In this subsection, we provide further details about the generalized neighbor aggregation for LightGCN, which extends its capabilities by allowing flexible norm scaling through the following aggregation rule.

$$\mathbf{e}_i^{(k+1)} = \frac{1}{|\mathcal{N}_i|^{\alpha}} \sum_{u \in \mathcal{N}_i} \frac{1}{|\mathcal{N}_u|^{1-\alpha}} \mathbf{e}_u^{(k)}, \tag{3}$$

where $\alpha \in [0, 1]$ is a controllable hyperparameter that provides flexibility and adaptability in scaling embedding norms.

To build upon this, we first establish that Observation 1 can be generalized to varying values of α in Eq. (3).

Observation 2. *Empirically, the norm of the unscaled aggregated neighbor embeddings in Eq. (3) at each k^{th} ($k \geq 0$) layer exhibits a near linear relationship with the number of neighbors, across varying values of $\alpha \in [0, 1]$. Specifically,*

$$\left\| \sum_{u \in \mathcal{N}_i} \frac{1}{|\mathcal{N}_u|^{1-\alpha}} \mathbf{e}_u^{(k)} \right\| \propto |\mathcal{N}_i|.$$

Results on Observation 2. In Figure 5, we present visualizations of Observation 2 across five datasets. The results empirically support our observation that the norm of the unscaled aggregated embedding in Eq. (3) exhibits a near-linear relationship with the number of neighbors. Specifically,

$$\left\| \sum_{u \in \mathcal{N}_i} \frac{1}{|\mathcal{N}_u|^{1-\alpha}} \mathbf{e}_u^{(k)} \right\| \propto |\mathcal{N}_i|$$

Table 4: The Pearson correlation between the norm of the unscaled aggregated embedding $\left\| \sum_{u \in \mathcal{N}_i} |\mathcal{N}_u|^{\alpha-1} \mathbf{e}_u^{(k)} \right\|$ and $|\mathcal{N}_i|$ is observed to be high for various αs for $k \geq 1$ (Observation 2). Specifically, $k = 2$ is used below.

Dataset	$\alpha = 0.0$	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 1.0$
LastFM	0.9907	0.9967	0.9982	0.9984	0.9985	0.9989
MovieLens	0.9968	0.9993	0.9998	0.9984	0.9957	0.9917
Gowalla	0.9361	0.9432	0.9615	0.9729	0.9803	0.9822
Yelp	0.9777	0.9888	0.9940	0.9958	0.9953	0.9954
Amazon	0.9671	0.9724	0.9751	0.9765	0.9822	0.9864

for $k \geq 0$. Additionally, Table 4 presents the Pearson correlation coefficient between $\left\| \sum_{u \in \mathcal{N}_i} |\mathcal{N}_u|^{\alpha-1} \mathbf{e}_u^{(k)} \right\|$ and $|\mathcal{N}_i|$, using $k = 2$ as an example. The results demonstrate high correlations across various values of α, numerically validating Observation 2.

Preservation of Properties 3 and 4. We explain how Properties 3 and 4 of LightGCN are maintained by the generalized neighbor aggregation rule. Specifically, Eq. (3) can be rewritten as follows:

$$\mathbf{e}_i^{(k+1)} = \frac{1}{|\mathcal{N}_i|^{\alpha}} \sum_{u \in \mathcal{N}_i} \frac{1}{|\mathcal{N}_u|^{1-\alpha}} \mathbf{e}_u^{(k)} = \frac{1}{|\mathcal{N}_i|^{\alpha}} \sum_{u \in \mathcal{N}_i} \frac{\left\|\mathbf{e}_u^{(k)}\right\|}{|\mathcal{N}_u|^{1-\alpha}} \frac{\mathbf{e}_u^{(k)}}{\left\|\mathbf{e}_u^{(k)}\right\|}.$$

This leads to the observation that the effective weight of neighbor u is $\left\|\mathbf{e}_u^{(k)}\right\| / |\mathcal{N}_u|^{1-\alpha}$. The near linear relation $\left\|\mathbf{e}_u^{(k)}\right\| \propto |\mathcal{N}_u|^{1-\alpha}$ for $k \geq 1$ leads to near-uniform effective weight $\left\|\mathbf{e}_u^{(k)}\right\| / |\mathcal{N}_u|^{1-\alpha} \propto |\mathcal{N}_u|^{1-\alpha} / |\mathcal{N}_u|^{1-\alpha} = 1$ for each neighbors u for $k \geq 1$ (Property 3 of LightGCN). On the other hand, when $k = 0$, there is no discernible pattern between $\left\|\mathbf{e}_u^{(k)}\right\|$ and $|\mathcal{N}_u|^{1-\alpha}$, and thus due to the denominator $|\mathcal{N}_u|^{1-\alpha}$, the effective weights tend to decrease w.r.t. the degree of neighbors (Property 4 of LightGCN).

Norm scaling w.r.t. α. In Figure 6, we demonstrate the controllable hyperparameter α in Eq. (3) allows for flexible adjustment of the norms of the agg-based embedding $\mathbf{e}_i^{(k)}$'s ($k \geq 1$). Specifically, for $k \geq 1$, $\left\|\mathbf{e}_i^{(k)}\right\| \propto |\mathcal{N}_i|^{1-\alpha}$ holds.

Effects of α. In Figure 7, we measure NDCG@20 across a range of α values in Eq. (3). The results demonstrate that adhering strictly to $\alpha = 0.5$, as defined in LightGCN, may not yield optimal performance across different datasets. This highlights the importance of adaptable and flexible norm scaling tailored to each dataset to achieve more accurate recommendations.

4 THEORETICAL ANALYSIS

In this section, we provide the theoretical analysis that supports our empirical observations. We first examine the norm of the aggregated embedding of uncorrelated (i.e., independent) neighbors. Then, we generalize this to neighbors with correlations. Lastly, we discuss the space complexity of LightGCN++.

4.1 Norm of Aggregated Embedding of Uncorrelated Neighbors

To demonstrate that our Observation 1 is not mathematically trivial, we show that embeddings form simple distributions do not exhibit the observation. For example, if we simply assume that neighbor embeddings are independently sampled from a normal distribution, the expected norm of the unscaled aggregated embeddings is not linear w.r.t. the number of neighbors.

Theorem 1. *Let M be a set of d-dimensional embeddings $\{\mathbf{x}_i\}_{i=1}^{|M|}$, where each dimension of $\mathbf{x}_i \in M$ is independently drawn from a normal distribution $N(0, \sigma^2)$. If the embeddings are uncorrelated (i.e., independent) sample-wise, the expected L2 norm of the unscaled aggregated embeddings in M is proportional to $\sqrt{|M|}$:*

$$\mathbb{E}\left[\left\| \sum_{\mathbf{x}_i \in M} \mathbf{x}_i \right\|\right] \propto \sqrt{|M|}$$

Proof. Let $\mathbf{X}_j^{(M)} = \sum_{\mathbf{x}_i \in M} \mathbf{x}_{ij}$. Based on the property of the sum of normal distributions, $\mathbf{X}_j^{(M)} \sim N(0, |M|\sigma^2)$. Thus, $\mathbf{X}_1^{(M)} / \sqrt{|M|\sigma^2}$, $\cdots$, $\mathbf{X}_d^{(M)} / \sqrt{|M|\sigma^2}$ are d independent random variables from $N(0,1)$. This implies that the following statistic is distributed according to the chi distribution with d degrees of freedom:

$$\sqrt{\sum_{j=1}^{d} \left(\frac{\mathbf{X}_j^{(M)}}{\sqrt{|M|\sigma^2}} \right)^2} = \frac{1}{\sqrt{|M|\sigma^2}} \left\| \mathbf{X}^{(M)} \right\|.$$

From the expected value of the chi distribution:

$$\mathbb{E}\left[\frac{1}{\sqrt{|M|\sigma^2}} \left\| \mathbf{X}^{(M)} \right\| \right] = \sqrt{2} \frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)},$$

where $\Gamma(\cdot)$ is the gamma function. Thus, the expected value of L2 norm $\left\|\mathbf{X}^{(M)}\right\|$ of $\mathbf{X}^{(M)}$ is:

$$\mathbb{E}\left[\left\| \mathbf{X}^{(M)} \right\| \right] = \sqrt{2|M|\sigma^2} \frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)} \propto \sqrt{|M|}.$$

This completes the proof. □

From Theorem 1, it is evident that embeddings that are simply sampled independently from a normal distribution do not exhibit the linear relationship in Eq. (9), implying that Observation 1 is indeed non-trivial. Next, we examine the variance of the L2 norm of the aggregated embedding.

Theorem 2. *Let M be a set of d-dimensional embeddings $\{\mathbf{x}_i\}_{i=1}^{|M|}$, where each dimension of $\mathbf{x}_i \in M$ is independently drawn from a normal distribution $N(0, \sigma^2)$. If the embeddings are uncorrelated (i.e., independent) sample-wise, the variance of the L2 norm of the unscaled aggregated embeddings in M is:*

$$\mathbb{V}ar\left[\left\| \sum_{\mathbf{x}_i \in M} \mathbf{x}_i \right\|\right] = |M|\sigma^2 \left(d - \left(\sqrt{2} \frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)} \right)^2 \right)$$

Proof. Let $\mathbf{X}_j^{(M)} = \sum_{\mathbf{x}_i \in M} \mathbf{x}_{ij}$. Then, from the proof of Theorem 1, $\mathbf{X}_1^{(M)} / \sqrt{|M|\sigma^2}, \cdots, \mathbf{X}_d^{(M)} / \sqrt{|M|\sigma^2}$ are d independent random variables from $N(0,1)$. This implies that the following statistic is distributed according to the chi distribution with d degrees of freedom. From the variance of the chi distribution:

$$\mathbb{V}\text{ar}\left[\frac{1}{\sqrt{|M|\sigma^2}} \left\| \mathbf{X}^{(M)} \right\| \right] = d - \mathbb{E}\left[\frac{1}{\sqrt{|M|\sigma^2}} \left\| \mathbf{X}^{(M)} \right\| \right]^2 = d - \left(\sqrt{2} \frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)} \right)^2.$$

From the properties of variance, we have:

$$\mathbb{V}\mathrm{ar}\left[\left\|\mathbf{X}^{(M)}\right\|\right] = |M|\sigma^2\left(d - \left(\sqrt{2}\frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)}\right)^2\right).$$

This completes the proof. □

Since the expected value of the chi distribution is close to $\sqrt{d-\frac{1}{2}}$ for large d, the following approximation holds in the high-dimensional space:

$$\mathbb{V}\mathrm{ar}\left[\left\|\sum_{\mathbf{x}_i\in M}\mathbf{x}_i\right\|\right] \approx \frac{|M|\sigma^2}{2}.$$

This indicates that the variance linearly increases with the number of neighbors, i.e., $|M|$.

4.2 Norm of Aggregated Embedding of Correlated Neighbors (Generalization)

We have observed that when embeddings are independently sampled from a normal distribution, the expected norm of the unscaled aggregated embeddings is sublinear w.r.t. the number of neighbors. Here, we demonstrate how the correlation between neighbors affects the linearity between the L2 norm of the aggregated embedding and the number of neighbors. Specifically, we introduce the correlation coefficient ρ between neighbors when computing the L2 norm of the aggregated pairs.

Theorem 3. *Let M be a set of d-dimensional embeddings $\{\mathbf{x}_i\}_{i=1}^{|M|}$ consisting of random variables. Assume that (1) at each j^{th} dimension, $\mathbf{x}_{1j},\cdots,\mathbf{x}_{|M|j}$ are drawn from a multivariate normal distribution $N(0,\Sigma)$ with $\Sigma_{ii}=\sigma^2\ \forall i, \Sigma_{ik}=\rho\sigma^2\ \forall i\neq k$ for some $\rho\geq 0$ (i.e., each $\mathbf{x}_{ij}\sim N(0,\sigma^2)$ and each pair $\mathbf{x}_{ij}$ and $\mathbf{x}_{kj}$ have correlation coefficient ρ), and (2) the dimensions are mutually independent and thus i.i.d. Then, the expected L2 norm of the unscaled aggregated embeddings in M follows the proportionality:*

$$\mathbb{E}\left[\left\|\sum_{\mathbf{x}_i\in M}\mathbf{x}_i\right\|\right] \propto \sqrt{|M|(1-\rho)+|M|^2\rho}.$$

Note that if $\rho=1$, then $\mathbb{E}\left[\left\|\sum_{\mathbf{x}_i\in M}\mathbf{x}_i\right\|\right]\propto|M|$.

Proof. Let $\mathbf{X}_j^{(M)}=\sum_{\mathbf{x}_i\in M}\mathbf{x}_{ij}$. Based on the property of the sum of normal distributions, the expected value of $\mathbf{X}_j^{(M)}$ is 0. The variance of $\mathbf{X}_j^{(M)}$ is:

$$\begin{aligned}\mathbb{V}\mathrm{ar}\left[\mathbf{X}_j^{(M)}\right] &= \sum_{\mathbf{x}_i\in M}\mathbb{V}\mathrm{ar}\left[\mathbf{x}_{ij}\right] + \sum_{\substack{\mathbf{x}_i,\mathbf{x}_k\in M\\ i\neq k}}\mathbb{C}\mathrm{ov}\left(\mathbf{x}_{ij},\mathbf{x}_{kj}\right)\\ &= |M|\sigma^2 + |M|(|M|-1)\rho\sigma^2\\ &= \sigma^2\left(|M|(1-\rho)+|M|^2\rho\right).\end{aligned}$$

Thus, $\mathbf{X}_j^{(M)}/\sqrt{\sigma^2\left(|M|(1-\rho)+|M|^2\rho\right)}$ for $j=1\cdots d$ are d independent random variables from $N(0,1)$. This implies that the following statistic is distributed according to the chi distribution with d degrees of freedom:

$$\sqrt{\sum_{j=1}^{d}\left(\frac{\mathbf{X}_j^{(M)}}{\sqrt{\sigma^2\left(|M|(1-\rho)+|M|^2\rho\right)}}\right)^2} = \frac{\left\|\mathbf{X}^{(M)}\right\|}{\sqrt{\sigma^2\left(|M|(1-\rho)+|M|^2\rho\right)}}.$$

From the expected value of the chi distribution:

$$\mathbb{E}\left[\frac{\left\|\mathbf{X}^{(M)}\right\|}{\sqrt{\sigma^2\left(|M|(1-\rho)+|M|^2\rho\right)}}\right] = \sqrt{2}\frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)},$$

where $\Gamma(\cdot)$ is the gamma function. Thus, the expected value of L2 norm $\left\|\mathbf{X}^{(M)}\right\|$ of $\mathbf{X}^{(M)}$ is:

$$\begin{aligned}\mathbb{E}\left[\left\|\mathbf{X}^{(M)}\right\|\right] &= \sqrt{2\sigma^2\left(|M|(1-\rho)+|M|^2\rho\right)}\frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)}\\ &\propto \sqrt{|M|(1-\rho)+|M|^2\rho}.\end{aligned}$$

This completes the proof. □

Theorem 3 suggests that if the embeddings are independent of each other (i.e., $\rho=0$), as assumed in Theorem 1, the norm of the aggregated embedding is sublinear w.r.t. the number of neighbors, i.e., $\mathbb{E}\left[\left\|\mathbf{X}^{(M)}\right\|\right]\propto\sqrt{|M|}$. Conversely, if the embeddings exhibit complete linear relationships with each other (i.e., $\rho=1$), the norm of the aggregated embedding is linear with the number of neighbors, i.e., $\mathbb{E}\left[\left\|\mathbf{X}^{(M)}\right\|\right]\propto|M|$. This indicates how the degree of correlation between embeddings influences the linearity of their aggregated norm. We conjecture that Observation 1 is attributed to the strong correlations between user/item embeddings. Next, we examine the variance of the L2 norm of the aggregated embedding.

Theorem 4. *Let M be a set of d-dimensional embeddings $\{\mathbf{x}_i\}_{i=1}^{|M|}$ consisting of random variables. Assume that (1) at each j^{th} dimension, $\mathbf{x}_{1j},\cdots,\mathbf{x}_{|M|j}$ are drawn from a multivariate normal distribution $N(0,\Sigma)$ with $\Sigma_{ii}=\sigma^2\ \forall i, \Sigma_{ik}=\rho\sigma^2\ \forall i\neq k$ for some $\rho\geq 0$ (i.e., each $\mathbf{x}_{ij}\sim N(0,\sigma^2)$ and each pair $\mathbf{x}_{ij}$ and $\mathbf{x}_{kj}$ have correlation coefficient ρ), and (2) the dimensions are mutually independent and thus i.i.d. Then, the variance of the L2 norm of the unscaled aggregated embeddings in M is:*

$$\mathbb{V}ar\left[\left\|\sum_{\mathbf{x}_i\in M}\mathbf{x}_i\right\|\right] = \sigma^2\left(|M|(1-\rho)+|M|^2\rho\right)\left(d-\left(\sqrt{2}\frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)}\right)^2\right)$$

Proof. Let $\mathbf{X}_j^{(M)}=\sum_{\mathbf{x}_i\in M}\mathbf{x}_{ij}$. Then, from the proof of Theorem 3, $\mathbf{X}_j^{(M)}/\sqrt{\sigma^2\left(|M|(1-\rho)+|M|^2\rho\right)}$ for $j=1,\cdots,d$ are d independent random variables from $N(0,1)$. This implies that the following statistic is distributed according to the chi distribution with d degrees of freedom. From the variance of the chi distribution:

Table 5: Comparison of LightGCN, its enhanced methods (i.e., ALGCN [13], r-AdjNorm [16], and SSM [11]), and LightGCN++ reveals that LightGCN++ incorporates key design properties that existing methods have overlooked.

		LightGCN [3]	ALGCN [13]	r-AdjNorm [16]	SSM [11]	**LightGCN++ (Ours)**
Effective weights	Definition	$\|\mathbf{e}_u^{(k)}\|/\sqrt{\|\mathcal{N}_u\|}$	$\|\mathbf{e}_u^{(k)}\|/\|\mathcal{N}_u\|$	$\|\mathbf{e}_u^{(k)}\|/\|\mathcal{N}_u\|^{1-\alpha}$	$\|\mathbf{e}_u^{(k)}\|/\|\mathcal{N}_u\|^{\beta}$	$1/\|\mathcal{N}_u\|^{\beta}$
	P1	✗	✗	✗	✓	✓
	P2	✗	✗	✗	✗	✓
Layer-wise Aggregation	Definition	$\frac{1}{K+1}\sum_{k=0}^{K}\mathbf{e}_i^{(k)}$	$\frac{1}{K+1}\sum_{k=0}^{K}\mathbf{e}_i^{(k)}$	$\frac{1}{K+1}\sum_{k=0}^{K}\mathbf{e}_i^{(k)}$	$\frac{1}{K+1}\sum_{k=0}^{K}\mathbf{e}_i^{(k)}$	$\gamma\mathbf{e}_i^{(0)}+(1-\gamma)\sum_{k=1}^{K}\mathbf{e}_i^{(k)}$
	P3	✗	✗	✗	✗	✓

$$\mathbb{V}\mathrm{ar}\left[\frac{1}{\sqrt{\sigma^2\left(|M|(1-\rho)+|M|^2\rho\right)}}\left\|\mathbf{X}^{(M)}\right\|\right] =$$

$$d-\mathbb{E}\left[\frac{1}{\sqrt{\sigma^2\left(|M|(1-\rho)+|M|^2\rho\right)}}\left\|\mathbf{X}^{(M)}\right\|\right]^2 = d-\left(\sqrt{2}\frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)}\right)^2.$$

From the properties of variance, we have:

$$\mathbb{V}\mathrm{ar}\left[\left\|\mathbf{X}^{(M)}\right\|\right] = \sigma^2\left(|M|(1-\rho)+|M|^2\rho\right)\left(d-\left(\sqrt{2}\frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)}\right)^2\right).$$

This completes the proof. □

Since the expected value of the chi distribution is close to $\sqrt{d-\frac{1}{2}}$ for large d, the following approximation holds in the high-dimensional space:

$$\mathbb{V}\mathrm{ar}\left[\left\|\sum\nolimits_{\mathbf{x}_i\in M}\mathbf{x}_i\right\|\right] \approx \frac{\sigma^2\left(|M|(1-\rho)+|M|^2\rho\right)}{2}.$$

5 DETAILS ON RELATED WORKS

The propagation rule of LightGCN has been enhanced in various directions. In this section, We begin by reviewing the properties of LightGCN. Then, we examine whether existing methods for enhancing LightGCN address these limitations.

Properties of LightGCN. We begin by reviewing properties of LightGCN that we have identified: **inflexibility** and **inconsistency** which are summarized as follows:

- **P1. Near-uniform neighbor weights.** When $k \geq 1$, effective weights are near-uniform across neighbors (Property 1).
- **P2. Uncontrollable neighbor weights.** When $k = 0$, effective weights are not adjustable (Property 2).
- **P3. Inflexible layer-wise norm scaling.** While norm scaling of embeddings at $k = 0$ and $k \geq 1$ exhibits disparities (Properties 3 & 4), they are aggregated with a fixed ratio (Property 5).

Existing LightGCN-enhanced methods. We assess whether existing methods for enhancing LightGCN address the limitations discussed above. Recently, the focus of most research has been on adjusting the embedding norms (i.e., magnitudes) [11, 13, 16]. However, these studies overlook key limitations of LightGCN, which we have empirically discovered in this paper as summarized in Table 5.

Xu et al. [13] generalized the left norm of LightGCN's propagation rule to be a function of the user/item popularity, as follows:

$$\mathbf{e}_i^{(k+1)} = \sum_{u\in\mathcal{N}_i} f(|\mathcal{N}_i|)\frac{1}{|\mathcal{N}_u|}\mathbf{e}_u^{(k)}$$

where $f(|\mathcal{N}_i|)$ is a function of $|\mathcal{N}_i|$, e.g., $f(|\mathcal{N}_i|) = |\mathcal{N}_i|^{0.5}$ or $f(|\mathcal{N}_i|) = \log_2(|\mathcal{N}_i|+1)$. While the function $f(\cdot)$ allows for flexible adjustment of the embedding norms, its effective weights remain inflexible and cannot be adjusted.

Zhao et al. [16] introduced a hyperparameter α as follows:

$$\mathbf{e}_i^{(k+1)} = \sum_{u\in\mathcal{N}_i}\frac{1}{|\mathcal{N}_i|^{\alpha}|\mathcal{N}_u|^{1-\alpha}}\mathbf{e}_u^{(k)},$$

which is essentially equivalent to Eq. (6). This approach provides flexibility in scaling embedding norms by introducing α. However, the effective weights are constant (i.e., uniform) when $k \geq 1$, as discussed in Section 4.1. [1]

Wu et al. [11] introduced another hyperparameter β into the right term of the propagation rule as follows:

$$\mathbf{e}_i^{(k+1)} = \sum_{u\in\mathcal{N}_i}\frac{1}{|\mathcal{N}_i|^{\alpha}|\mathcal{N}_u|^{\beta}}\mathbf{e}_u^{(k)}.$$

While β can be used to flexibly adjust the effective weights of neighbors, its role was overlooked in [11], and it was introduced to control the norm of embeddings along with α. In addition, its effective weight $\|\mathbf{e}_u^{(k)}\|/|\mathcal{N}_u|^{\beta}$ is uncontrollable when $k = 0$, as discussed in Section 3.4. [2]

Commonly, prior works have primarily focused on adjusting the norms of embeddings. In contrast, LightGCN++ is designed to address two unexpected and unexplored properties of LightGCN: (1) **inflexibility**: the near-uniform weights of neighbors and (2) **inconsistency**: disparities in norm scaling between layers at $k = 0$ and $k \geq 1$. As summarized in Table 5, LightGCN++ extends beyond adjusting embedding norms and addresses the two crucial limitations of LightGCN.

Furthermore, LightGCN has been enhanced through long-range propagation [5, 8]. Specifically, Mao et al. [8] approximated the limit of infinite-layer graph convolution of LightGCN. Kapralova et al. [5] adapted personalized PageRank into LightGCN's propagation rule. These studies have not investigated the unique properties of LightGCN, which we discussed in Section 3 in the main paper.

[1] Recall our empirical observation that, in $\mathbf{e}_i^{(k+1)} = \sum_{u\in\mathcal{N}_i}\frac{1}{|\mathcal{N}_i|^{\alpha}|\mathcal{N}_u|^{\beta}}\mathbf{e}_u^{(k)}$, the effective weight $\|\mathbf{e}_u^{(k)}\|/|\mathcal{N}_u|^{\beta}$ becomes a constant if $\beta = 1-\alpha$ (Section 4.1).

[2] The linearity $\|\mathbf{e}_u^{(k)}\| \propto |\mathcal{N}_u|^{1-\alpha}$ is observed only when $k \geq 1$ (Property 2).

6 DETAILS ON EXPERIMENTAL SETTINGS

In this section, we offer a detailed description of the settings used in our experiments.

6.1 Experimental Settings

Datasets. We used five benchmark datasets, LastFM, MovieLens, Gowalla, Yelp, and Amazon to conduct experiments. The statistics of each dataset are reported in Table 6.

Baselines. We compared LightGCN++ against the following baseline methods:

- **BPRMF** [9] optimizes the BPR loss to learn the embeddings for users and items using matrix factorization (MF).
- **NeuMF** [4] uses an MLP instead of the dot product in the MF to learn the matching function between users and items.
- **NGCF** [10] incorporates both feature transformation and nonlinearities in its GNN framework.
- **LR-GCCF** [2] removes nonlinearities while incorporating feature transformation within its GNN framework.
- **HCCF** [12] exploits contrastive learning to integrate an explicit interaction graph with the learned implicit hypergraph structure.
- **LightGCL** [1] utilizes an SVD-reconstructed graph as an augmented view in its contrastive learning framework.
- **LightGCN** [3] simplifies NGCF by removing feature transformation and nonlinearities (see Section 2.1 in the main paper for more details).
- **NCL** [6] incorporates both structural and semantic neighbors for each node to construct contrastive pairs.
- **SimGCL** [15] applies random noise into embeddings to create an augmented view for its contrastive learning framework.
- **XSimGCL** [14] simplifies SimGCL by directly applying noises into the embeddings that are used for making predictions.

The source code utilized for each baseline is listed in Table 7. For LR-GCCF [2], we manually implemented it based on the official PyTorch code of LightGCN. For the methods that equip LightGCN (i.e., NCL [6], SimGCL [15], XSimGCL [14]), we integrated their original contrastive loss functions into the LightGCN's framework.

Hyperparameter search. The dataset was divided into training, validation, and test sets following a 7:1:2 ratio. We then searched the hyperparameter settings for each method that yields the best NDCG@20 for the validation set. We initialized the learnable parameters (i.e., embeddings $\mathbf{E}$ at the initial layer) following a normal distribution. The embedding dimension is set to 64, the batch size is set to 2048, and the learning rate is set to 0.001 with a regularization coefficient λ of 0.0001. For GNN-based models, we used the number of layers $K = 2$. The search space for model-specific hyperparameters for each method is conducted as follows:

- **HCCF** [12]: The four hyperparameters, λ_1 (weight decay), λ_2 (weight for contrastive learning loss), τ (temperature for contrastive loss), γ (dropout edge preservation ratio) are selected from $\{0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$, $\{10^{-3}, 10^{-2}, 10^{-1}, 0.1, 0.2, 0.3\}$, $\{0.1, 0.3, 1, 3, 10\}$, and $\{0.25, 0.5, 0.75, 1.0\}$, respectively.

Table 6: Dataset Statistics.

Dataset	# User	# Item	# Interaction	Density
LastFM	1,885	17,388	91,779	0.00280
MovieLens	6,039	3,628	836,478	0.03817
Gowalla	29,858	40,981	1,027,370	0.00084
Yelp	31,668	38,048	1,561,406	0.00129
Amazon	52,643	91,599	2,704,860	0.00056

Table 7: Source code URLs of baselines.

Method	Source Code URL
BPRMF [9]	https://github.com/gusye1234/LightGCN-PyTorch
NeuMF [4]	https://github.com/guoyang9/NCF
NGCF [10]	https://github.com/huangtinglin/NGCF-PyTorch
LR-GCCF [2]	https://github.com/gusye1234/LightGCN-PyTorch
HCCF [12]	https://github.com/akaxlh/HCCF
LightGCL [1]	https://github.com/HKUDS/LightGCL
LightGCN [3]	https://github.com/gusye1234/LightGCN-PyTorch
NCL [6]	https://github.com/RUCAIBox/NCL
SimGCL [15]	https://github.com/Coder-Yu/SELFRec
XSimGCL [14]	https://github.com/Coder-Yu/SELFRec

- **LightGCL** [1]: The four hyperparameters, λ_1 (weight of contrastive loss), λ_2 (weight of L2 regularization [3]), τ (temperature for contrastive loss), and p (dropout rate) are selected from $\{0.001, 0.01, 0.1\}$, $\{10^{-8}, 10^{-7}\}$, $\{0.2, 0.5\}$, and $\{0.0, 0.25\}$.
- **NCL** [6]: There are three hyperparameters, λ_1 (weight of structure-contrastive loss), λ_2 (weight of prototype-contrastive loss), and K (number of prototypes). We tune them from $\{0.001, 0.0001\}$, $\{0.001, 0.0001\}$, and $\{100, 1000\}$, respectively.
- **SimGCL** [15]: The two hyperparameters, ϵ (magnitude of the noise) and λ (weight of contrastive loss) are selected from $\{0.01, 0.05, 0.1, 0.5\}$ and $\{0.01, 0.05, 0.1, 0.5\}$, respectively.
- **XSimGCL** [14]: The three hyperparameters, ϵ (magnitude of the noise), λ (weight of contrastive loss), and ℓ^* (target layer to contrast with the final embedding) are selected from $\{0.01, 0.05, 0.1, 0.5\}$, $\{0.01, 0.05, 0.1, 0.5\}$, and $\{1, 2\}$, respectively.
- **LightGCN++ (proposed)**: The three hyperparameters, α, β, and γ are selected from $\{0.4, 0.5, 0.4\}$, $\{-0.1, 0.0, 0.1\}$, and $\{0.0, 0.1, 0.2\}$.

Implementation. We implemented LightGCN++ based on the official PyTorch implementation of LightGCN [4]. It is easy to implement based on LightGCN's framework, with the following modifications:

- Instead of normalizing with $\mathbf{D}^{-0.5}\mathbf{A}\mathbf{D}^{-0.5}$, where $\mathbf{A}$ is the adjacency matrix and $\mathbf{D}$ is a diagonal degree matrix, we apply normalization using $\mathbf{D}^{-\alpha}\mathbf{A}\mathbf{D}^{-\beta}$ using hyperparameters α and β.
- At the beginning of each k^{th} ($k \geq 0$) layer, we normalize the embedding $\mathbf{e}_i^{(k)}$ to a unit vector $\mathbf{e}_i^{(k)} / \left\|\mathbf{e}_i^{(k)}\right\|$.
- After aggregating neighbors over K layers, we apply a weighted sum to $\mathbf{e}_i^{(0)}$ and $\mathbf{e}_i^{(1)} + \cdots + \mathbf{e}_i^{(K)}$ using the hyperparameter γ.

For **reproducibility**, we make our code and dataset available at https://anonymous.4open.science/r/LightGCNpp-F38B (anonymized).

6.2 Hyperparameter Tuning Strategies

Here, we share some simple strategies for tuning α, β, and γ in LightGCN++ for its practical usability.

[3] In the official implementation of LightGCL, the computation of L2 regularization differs from that of LightGCN. Thus, we manually tuned this hyperparameter instead of using the default setting (i.e., 0.0001).

[4] https://github.com/gusye1234/LightGCN-PyTorch

Table 8: Runtime (in seconds) of LightGCN and LightGCN++ per epoch. LightGCN++ is marginally slower than LightGCN.

Dataset	LightGCN	LightGCN++	Increase
LastFM	0.9137	0.9345	1.0227 ×
MovieLens	10.0144	10.0232	1.0008 ×
Gowalla	13.2507	13.3094	1.0044 ×
Yelp	21.2809	22.0242	1.0349 ×
Amazon	46.2871	48.7372	1.0529 ×

- **Tuning α.** We advise users to adjust α based on the long-tailed characteristics of the item popularity (i.e., degree) distribution. Specifically, increasing α (i.e., $\alpha \to 1$) leads to a fairer recommendation that is equally likely to recommend both popular and unpopular items. In contrast, decreasing α (i.e., $\alpha \to 0$) biases the system towards recommending more popular items.
- **Tuning β.** According to our experiments, setting $0 \geq \beta \geq 1$ generally leads to improvements. This suggests that reducing the influence of high-degree neighbors enhances accuracy.
- **Tuning γ.** We recommend that users begin by tuning γ from 0, which excludes the embeddings at the initial layer (i.e., $\mathbf{e}_i^{(0)}$) from the layer-wise aggregation. Then, gradually increasing γ may enhance performance, depending on the dataset.

6.3 Complexity Analysis

We analyze the complexity of LightGCN++. Since it does not introduce any extra trainable parameters compared to LightGCN, its $O(|\mathcal{V}|Kd)$ space complexity remains unchanged. The time complexity of each phase of LightGCN++ is:

- The time complexity of computing the degree of each node (i.e., $|\mathcal{N}_i|$'s and $|\mathcal{N}_u|$'s) is $O(|\mathcal{E}|)$.
- While LightGCN takes $O(|\mathcal{E}|Kd)$ time for graph convolution, LightGCN++ involves embedding normalization at each layer, resulting in a time complexity of $O(|\mathcal{E}|Kd + |\mathcal{V}|Kd)$. Typically, since $|\mathcal{V}| \ll |\mathcal{E}|$, the increase in time complexity is minimal.
- The time complexity of computing the BPR loss is $O(|\mathcal{E}|d)$.

We empirically demonstrate that LightGCN++ is marginally slower than LightGCN in the main text, and thus it is still light and fast. While the complexity for normalizing the adjacency matrix and computing the BPR loss remains the same, LightGCN++ takes additional time to normalize the embeddings of users and items during the graph convolution at each layer. Thus, the time complexity for graph convolution in LightGCN++ is $O(|\mathcal{E}|Kd + |\mathcal{V}|Kd)$, whereas in LightGCN, it is $O(|\mathcal{E}|Kd)$. However, as shown in Table 8, empirically, this additional computation does not significantly increase the runtime, and thus LightGCN++ takes advantage of the speed of LightGCN. Experiments were conducted on a server with RTX 3090Ti GPUs.

7 ADDITIONAL EXPERIMENTAL RESULTS

In this section, we provide additional experimental results.

7.1 Additional Results of LightGCN++

Different Ns for Recall@N and NDCG@N. Tables 12 and 13 present the performance of LightGCN++ and its baselines across five datasets, evaluating the performance in terms of top-10 and top-40 recommendations, respectively.

Table 9: The searched hyperparameters for each dataset.

Dataset	α	β	γ
LastFM	0.6	-0.1	0.2
MovieLens	0.5	0.0	0.0
Gowalla	0.6	-0.1	0.2
Yelp	0.6	-0.1	0.1
Amazon	0.6	-0.1	0.2

7.2 Parameter Sensitivity Analysis

We examine the influence of the controllable hyperparameters α, β, and γ on the performance of LightGCN++. We evaluate the performance of LightGCN++ for $\alpha \in \{0.0, 0.1, \cdots, 1.0\}$, $\beta \in \{-0.25, -0.2, \cdots, 0.25\}$, and $\gamma \in \{0.0, 0.1, \cdots 1.0\}$. We include results in Figures 8, 9, and 10, regarding the sensitivity analysis for parameters α, β, and γ, respectively. The results indicate the importance of the flexible and adaptive adjustment for each dataset.

Optimal α, β, and γ. In Table 9, we report the best hyperparameters searched from $\alpha \in \{0.4, 0.5, 0.6\}$, $\beta \in \{-0.1, 0.0, 0.1\}$, and $\gamma \in \{0.0, 0.1, 0.2\}$, for each dataset. Notably, $\alpha = 0.6$, $\beta = -0.1$, and $\gamma = 0.2$ worked best in three out of five datasets, and we recommend the users use this configuration as default.

7.3 Potential Extensions of LightGCN++

He et al. [3] reveal that feature transformation (FT) significantly reduces the effectiveness of GNN-based recommender systems, introducing unnecessary complexity. In this subsection, we further elaborate on the potential extension of adopting feature transformation (FT) and nonlinear activations (NA).

Feature Transformation in LightGCN++. LightGCN++ can be extended by incorporating FT as follows:

$$\mathbf{e}_i^{(k+1)} = \frac{1}{|\mathcal{N}_i|^\alpha} \sum_{u \in \mathcal{N}_i} \frac{1}{|\mathcal{N}_u|^\beta} \frac{\mathbf{W}^{(k)} \mathbf{e}_u^{(k)}}{\left\| \mathbf{W}^{(k)} \mathbf{e}_u^{(k)} \right\|}$$

where $\mathbf{W}^{(k)}$ is the trainable weight matrix used for FT at each k^{th} layer. Note that the projected embedding of each neighbor u is normalized, and an adjustable effective weight of $1/|\mathcal{N}_u|^\beta$ is applied. Moreover, the term, $1/|\mathcal{N}_i|^\alpha$ is used for adjusting embedding norms.

Nonlinear activation in LightGCN++. LightGCN++ can be extended by incorporating NA together with FT as follows:

$$\mathbf{e}_i^{(k+1)} = \phi\left(\frac{1}{|\mathcal{N}_i|^\alpha} \sum_{u \in \mathcal{N}_i} \frac{1}{|\mathcal{N}_u|^\beta} \frac{\mathbf{W}^{(k)} \mathbf{e}_u^{(k)}}{\left\| \mathbf{W}^{(k)} \mathbf{e}_u^{(k)} \right\|} \right),$$

where $\phi(\cdot)$ is the nonlinear activation function (e.g., Leaky ReLU), and $\mathbf{W}^{(k)}$ is the weight matrix at the k^{th} layer.

Experimental results. In Table 10, we report the performance (in terms of NDCG@20) for both LightGCN and LightGCN++ when equipped with FT and/or NA. For NA, we used the Leaky ReLU. The results show that LightGCN++ effectively reduces the performance degradation encountered in LightGCN with FT and/or NA.

Table 10: While LightGCN encounters significant performance degradation (in terms of NDCG@20) upon incorporating feature transformation (FT) and/or nonlinear activation (NA), LightGCN++ effectively mitigates this degradation.

Dataset	LastFM	MovieLens	Gowalla	Yelp	Amazon
LightGCN	0.2427	0.3010	0.1426	0.0449	0.0274
+ FT	0.1869	0.2814	0.1075	0.0388	0.0201
Degradation	22.99%	6.51%	24.61%	13.58%	26.64%
+ FT + NA	0.1980	0.2862	0.1083	0.0377	0.0188
Degradation	18.41%	**4.91%**	24.05%	16.03%	31.38%
LightGCN++	0.2624	0.3275	0.1469	0.0529	0.0294
+ FT	0.2429	0.3098	0.1420	0.0494	0.0282
Degradation	**7.43%**	**5.40%**	**3.33%**	**6.61%**	**4.08%**
+ FT + NA	0.2457	0.3099	0.1362	0.0486	0.0280
Degradation	**6.36%**	5.37%	**7.28%**	**8.12%**	**4.76%**

Table 11: LightGCN++'s adaptive pooling for layer-wise embedding aggregation, tuning γ, is more effective than both mean pooling and learnable pooling approaches.

Dataset	LastFM	MovieLens	Gowalla	Yelp	Amazon
Mean Pooling	0.2614	0.3217	0.1436	0.0507	0.0282
Learnable Pooling	0.2590	0.3227	0.1389	0.0524	0.0283
Adaptive Pooling	**0.2624**	**0.3275**	**0.1469**	**0.0529**	**0.0294**

7.4 Learnable Embedding Pooling

To evaluate the effectiveness of the adaptive layer-wise embedding aggregation approach used in LightGCN++, we compare it with two intuitive pooling methods. Specifically, mean pooling assigns equal importance across all layers, whereas learnable pooling learns individual weights for each layer. As shown in Table 11, adaptive pooling, upon fine-tuning γ, achieves the best results in terms of NDCG@20. This indicates that addressing the disparities in norm scaling between agg-free and agg-based embeddings by properly balancing them is effective for generating the final embeddings.

8 FUTURE DIRECTIONS

Lastly, we share potential future directions of this work as follows:

- **Enhancement of layer-wise pooling.** While we have distinguished between agg-free (k=0) and agg-based (k=1) embeddings for simplicity, more enhanced pooling strategies can potentially improve performance.
- **Enhancement from deeper layers.** Inspired by UltraGCN [8], LightGCN++ could benefit from incorporating propagations from deeper layers.
- **Further analysis on LightGCN.** While LightGCN has empirically shown the effectiveness of removing feature transformation and non-linear activation, it lacks theoretical justification for these improvements. We aim to delve deeper into this topic.
- **Extensions to general GNNs.** The concept of effective weights presents an opportunity to investigate the propagation mechanisms of general GNNs in areas beyond recommender systems.

REFERENCES

[1] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2022. LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation. In *ICLR*.

[2] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *AAAI*.

[3] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*.

[4] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*.

[5] Milena Kapralova, Luca Pantea, and Andrei Blahovici. 2023. LightGCN: Evaluated and Enhanced. *arXiv preprint arXiv:2312.16183* (2023).

[6] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *WWW*.

[7] Kang Liu, Feng Xue, and Richang Hong. 2022. RGCF: Refined graph convolution collaborative filtering with concise and expressive embedding. *Intelligent Data Analysis* 26, 2 (2022), 427–445.

[8] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: ultra simplification of graph convolutional networks for recommendation. In *CIKM*.

[9] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.

[10] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*.

[11] Jiancan Wu, Xiang Wang, Xingyu Gao, Jiawei Chen, Hongcheng Fu, Tianyu Qiu, and Xiangnan He. 2022. On the effectiveness of sampled softmax loss for item recommendation. *TOIS* (2022).

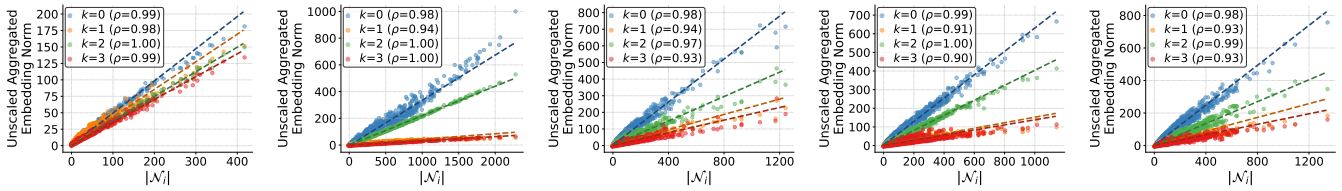
[12] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Huang. 2022. Hypergraph contrastive collaborative filtering. In *SIGIR*.

[13] Ronghai Xu, Haijun Zhao, Zhi-Yuan Li, and Chang-Dong Wang. 2023. ALGCN: Accelerated Light Graph Convolution Network for Recommendation. In *DASFAA*.

[14] Junliang Yu, Xin Xia, Tong Chen, Lizhen Cui, Nguyen Quoc Viet Hung, and Hongzhi Yin. 2023. XSimGCL: Towards extremely simple graph contrastive learning for recommendation. *TKDE* (2023).

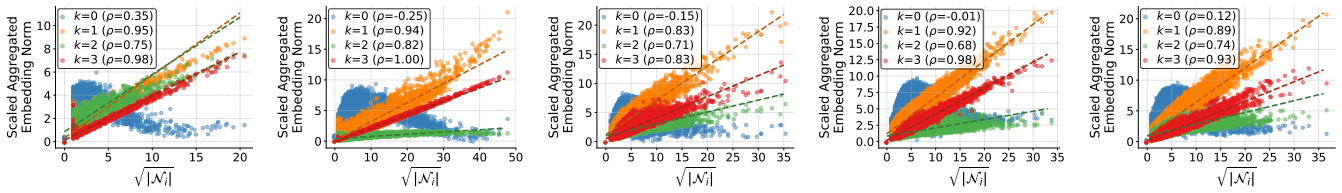
[15] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *SIGIR*.

[16] Minghao Zhao, Le Wu, Yile Liang, Lei Chen, Jian Zhang, Qilin Deng, Kai Wang, Xudong Shen, Tangjie Lv, and Runze Wu. 2022. Investigating accuracy-novelty performance for graph-based collaborative filtering. In *SIGIR*.



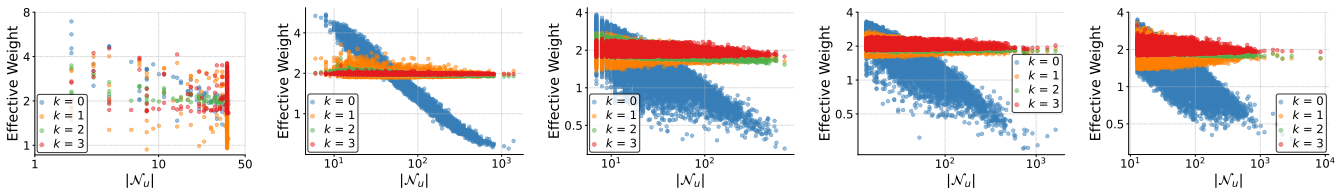
(a) LastFM (b) MovieLens (c) Gowalla (d) Yelp (e) Amazon

Figure 2: The norm of the unscaled aggregated embedding ($\sum_{u\in\mathcal{N}_i}|\mathcal{N}_u|^{-0.5}\mathbf{e}_u^{(k)}$) in Eq. (1) tends to be proportional to the number of neighbors $|\mathcal{N}_i|$ for $k \geq 0$ (Observation 1). The symbol ρ represents a Pearson correlation coefficient.



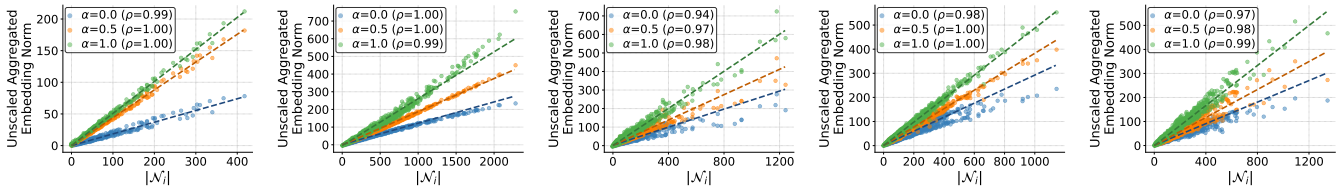
(a) LastFM (b) MovieLens (c) Gowalla (d) Yelp (e) Amazon

Figure 3: The norm of the scaled aggregated embedding ($\mathbf{e}_i^{(k)}$) in Eq. (1) tends to be proportional to $\sqrt{|\mathcal{N}_i|}$ for $k \geq 1$ (Property 1). This relationship does not hold when $k = 0$ (Property 2). The symbol ρ represents a Pearson correlation coefficient.



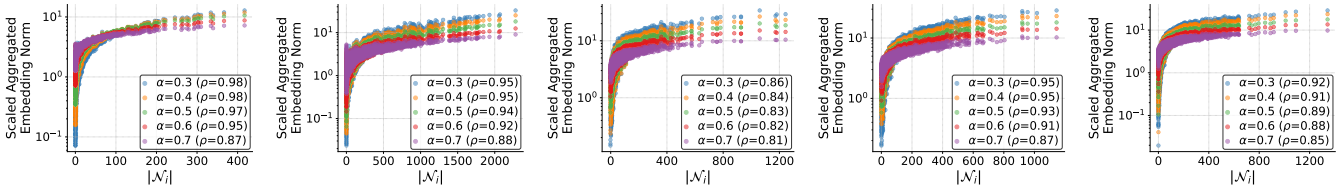
(a) LastFM (b) MovieLens (c) Gowalla (d) Yelp (e) Amazon

Figure 4: The effective weight $\|\mathbf{e}_u^{(k)}\|/\sqrt{|\mathcal{N}_u|}$ of neighbor u tends to be uniform when $k \geq 1$ (Property 3). When $k = 0$, the effective weight decreases with respect to the degree (Property 4).



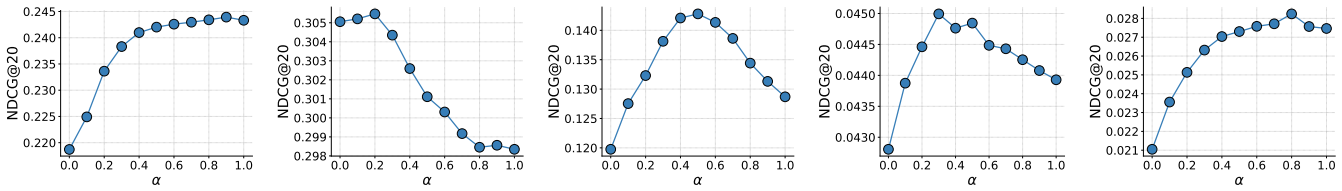
(a) LastFM (b) MovieLens (c) Gowalla (d) Yelp (e) Amazon

Figure 5: The norm of the aggregated neighbor embeddings ($\sum_{u\in\mathcal{N}_i}|\mathcal{N}_u|^{\alpha-1}\mathbf{e}_u^{(k)}$) in Eq. (3) is proportional to the number of neighbors $|\mathcal{N}_i|$ for $k \geq 0$ (Observation 2) in five datasets. The symbol ρ represents a Pearson correlation coefficient.



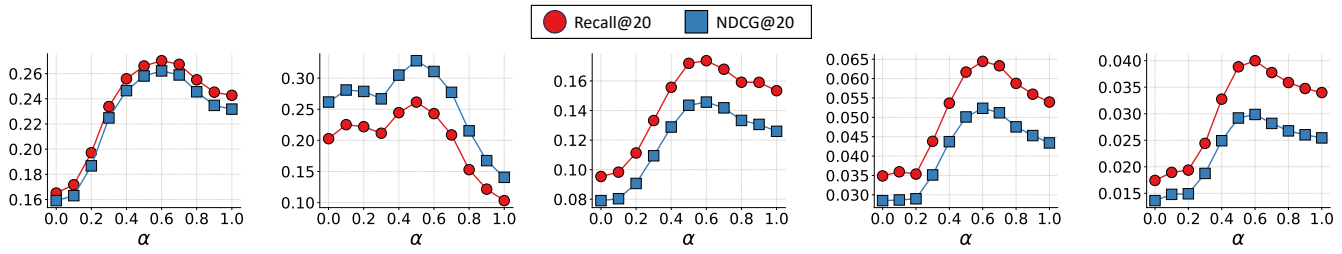
(a) LastFM (b) MovieLens (c) Gowalla (d) Yelp (e) Amazon

Figure 6: The controllable parameter α in Eq. (3) allows for flexible adjustment of norm scaling of agg-based embeddings. The symbol ρ represents a Pearson correlation coefficient.



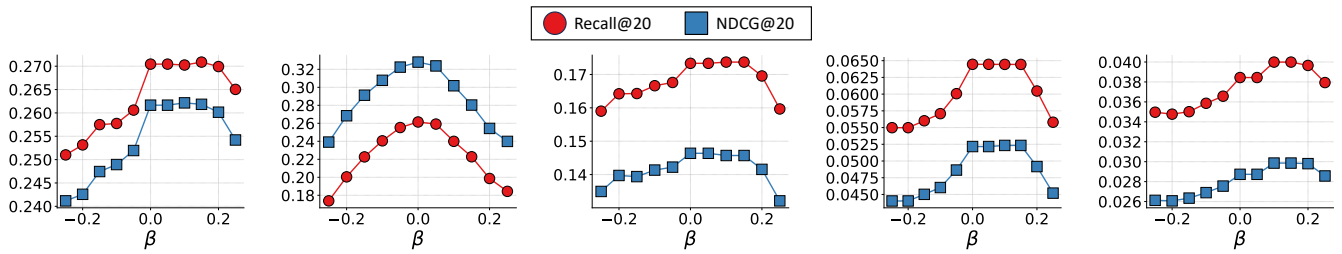
(a) LastFM (b) MovieLens (c) Gowalla (d) Yelp (e) Amazon

Figure 7: The value of α in Eq. (3), which determines the scaling of embedding norms, varies to achieve the best performance (in terms of NDCG@20) across different datasets.



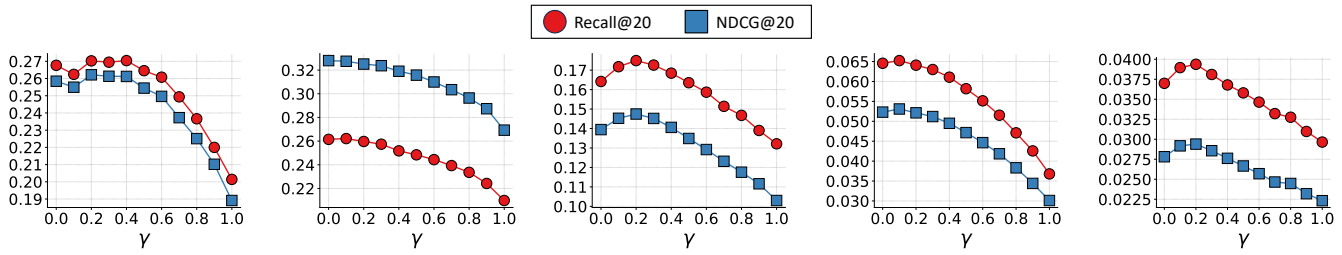
(a) LastFM (b) MovieLens (c) Gowalla (d) Yelp (e) Amazon

Figure 8: Impact of α on the performance of LightGCN++. The optimal α is different across datasets.



(a) LastFM (b) MovieLens (c) Gowalla (d) Yelp (e) Amazon

Figure 9: Impact of β on the performance of LightGCN++. The optimal β is different across datasets.



(a) LastFM (b) MovieLens (c) Gowalla (d) Yelp (e) Amazon

Figure 10: Impact of γ on the performance of LightGCN++. The optimal γ is different across datasets.

Table 12: LightGCN++ consistently and significantly outperforms LightGCN in terms of Recall@10 and NDCG@10. State-of-the-art methods enhanced with LightGCN++ (i.e., NCL++, SimGCL++, and XSimGCL++) also outperform their counterparts with LightGCN. * and ** denote $p < 0.01$ and $p < 0.001$ for a one-tailed t-test, indicating that a method with LightGCN++ significantly outperforms its counterpart with LightGCN. For each dataset, the best performance is in bold and the second-best is underlined.

Dataset	LastFM		MovieLens		Gowalla		Yelp		Amazon	
Metric	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10
BPRMF [9]	0.1361	0.1532	0.1291	0.2677	0.0895	0.0899	0.0209	0.0242	0.0165	0.0170
NeuMF [4]	0.1498	0.1688	0.1345	0.2707	0.0842	0.0871	0.0207	0.0232	0.0131	0.0135
NGCF [10]	0.1562	0.1769	0.1420	0.2872	0.0916	0.0942	0.0264	0.0294	0.0176	0.0177
LR-GCCF [2]	0.1367	0.1584	0.1025	0.2281	0.0675	0.0753	0.0224	0.0260	0.0095	0.0109
HCCF [12]	0.1520	0.1730	0.1397	0.2900	0.0827	0.0912	0.0365	0.0414	0.0190	0.0197
LightGCL [1]	0.1741	0.1997	0.1460	0.2907	0.1183	0.1274	0.0364	0.0411	0.0239	0.0243
LightGCN [3]	0.1741	0.1996	0.1495	0.2977	0.1182	0.1284	0.0321	0.0363	0.0206	0.0209
LightGCN++	0.1889**	0.2171**	0.1667**	0.3243**	0.1219**	0.1321**	0.0380**	0.0428**	0.0221**	0.0223**
Improvement	8.50%	8.77%	11.51%	8.94%	3.13%	2.88%	18.38%	17.91%	7.28%	6.70%
NCL [6]	0.1768	0.2024	0.1507	0.3002	0.1193	0.1283	0.0342	0.0385	0.0221	0.0222
NCL++	**0.1894****	**0.2177****	0.1677**	**0.3257****	**0.1234****	**0.1331****	**0.0401****	**0.0451****	0.0241**	0.0239**
Improvement	7.13%	7.56%	11.28%	8.49%	3.44%	3.74%	17.25%	17.14%	9.05%	7.66%
SimGCL [15]	0.1795	0.2049	0.1647	0.3186	0.1185	0.1274	0.0379	0.0427	0.0239	0.0242
SimGCL++	0.1880**	0.2151**	**0.1679****	0.3254**	0.1191	0.1283	0.0385**	0.0435**	0.0259**	0.0258**
Improvement	4.74%	4.98%	1.94%	2.13%	0.51%	0.71%	1.58%	1.87%	8.37%	6.61%
XSimGCL [14]	0.1801	0.2062	0.1657	0.3216	0.1173	0.1253	0.0380	0.0427	0.0226	0.0227
XSimGCL++	0.1893**	0.2173**	0.1671*	0.3242*	0.1188**	0.1283**	0.0397**	0.0447**	**0.0262****	**0.0263****
Improvement	5.11%	5.38%	0.84%	0.81%	1.28%	2.39%	4.47%	4.68%	15.93%	15.86%

Table 13: LightGCN++ consistently and significantly outperforms LightGCN in terms of Recall@40 and NDCG@40. State-of-the-art methods enhanced with LightGCN++ (i.e., NCL++, SimGCL++, and XSimGCL++) also outperform their counterparts with LightGCN. * and ** denote $p < 0.01$ and $p < 0.001$ for a one-tailed t-test, indicating that a method with LightGCN++ significantly outperforms its counterpart with LightGCN. For each dataset, the best performance is in bold and the second-best is underlined.

Dataset	LastFM		MovieLens		Gowalla		Yelp		Amazon	
Metric	Recall@40	NDCG@40	Recall@40	NDCG@40	Recall@40	NDCG@40	Recall@40	NDCG@40	Recall@40	NDCG@40
BPRMF [9]	0.2784	0.2235	0.3192	0.2914	0.1950	0.1238	0.0631	0.0401	0.0507	0.0303
NeuMF [4]	0.3065	0.2457	0.3320	0.2998	0.1767	0.1163	0.0652	0.0401	0.0409	0.0244
NGCF [10]	0.3218	0.2586	0.3439	0.3134	0.1986	0.1281	0.0794	0.0496	0.0544	0.0322
LR-GCCF [2]	0.2717	0.2248	0.2591	0.2433	0.1352	0.0948	0.0668	0.0427	0.0280	0.0179
HCCF [12]	0.3152	0.2529	0.3366	0.3131	0.1680	0.1165	0.1046	0.0672	0.0590	0.0355
LightGCL [1]	0.3480	0.2854	0.3540	0.3219	0.2389	0.1643	0.1035	0.0662	0.0697	0.0422
LightGCN [3]	0.3481	0.2854	0.3586	0.3265	0.2364	0.1640	0.0928	0.0592	0.0620	0.0371
LightGCN++	0.3745**	0.3083**	0.3852**	0.3534**	0.2448**	0.1693**	0.1074**	0.0690**	0.0665**	0.0398**
Improvement	7.58%	8.02%	7.42%	8.24%	3.55%	3.23%	15.73%	16.55%	7.26%	7.28%
NCL [6]	0.3502	0.2878	0.3597	0.3281	0.2389	0.1647	0.0969	0.0621	0.0658	0.0394
NCL++	0.3754**	0.3093**	**0.3861****	**0.3545****	**0.2490****	**0.1712****	**0.1120****	**0.0720****	0.0706**	0.0422**
Improvement	7.20%	7.47%	7.34%	8.05%	4.23%	3.95%	15.58%	15.94%	7.29%	7.11%
SimGCL [15]	0.3547	0.2914	0.3810	0.3480	0.2388	0.1641	0.1080	0.0692	0.0680	0.0416
SimGCL++	0.3734**	0.3068**	0.3858**	0.3541**	0.2405*	0.1652*	0.1081	0.0697*	0.0716**	0.0439**
Improvement	5.27%	5.28%	1.26%	1.75%	0.71%	0.67%	0.09%	0.72%	5.29%	5.53%
XSimGCL [14]	0.3575	0.2937	0.3830	0.3504	0.2339	0.1610	0.1074	0.0689	0.0657	0.0397
XSimGCL++	**0.3758****	**0.3094****	0.3856*	0.3534**	0.2419**	0.1657**	0.1112**	0.0716**	**0.0742****	**0.0452****
Improvement	5.12%	5.35%	0.68%	0.86%	3.42%	2.92%	3.54%	3.92%	12.94%	13.85%