# Graph Convolution Network based Recommender Systems: Learning Guarantee and Item Mixture Powered Strategy

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Inspired by their powerful representation ability on graph-structured data, Graph Convolution Networks (GCNs) have been widely applied to recommender systems, and have shown superior performance. Despite their empirical success, there is a lack of theoretical explorations such as generalization properties. In this paper, we take a first step towards establishing a generalization guarantee for GCN-based recommendation models under inductive and transductive learning. We mainly investigate the roles of graph normalization and non-linear activation, providing some theoretical understanding, and construct extensive experiments to further verify these findings empirically. Furthermore, based on the proven generalization bound and the challenge of existing models in discrete data learning, we propose Item Mixture (IMix) to enhance recommendation. It models discrete spaces in a continuous manner by mixing the embeddings of positive-negative item pairs, and its effectiveness can be strictly guaranteed from empirical and theoretical aspects.

## 1 Introduction

With the explosive growth of online information, Internet users rely on recommender systems to alleviate such information overload [1, 2, 3]. In order to take full advantage of the rich graph structure with high-hop neighbors in recommender systems, recent studies have begun to apply Graph Convolution Networks (GCNs) [4] to recommender systems. As expected, GCN-based models have emerged as one of the most attractive approaches to recommender systems due to their powerful graph representation abilities. However, these works only demonstrate the effectiveness of GCNs empirically and do not provide theoretical guarantees.

**GCN-based Recommendation.** [5] was the first to introduce GCN into recommender systems. Specifically, GCN in their approach was applied to the user-user and item-item graphs to learn auxiliary information, respectively. GC-MC [6] proposed to apply GCN to the user-item interaction graph to learn the relations between users and items. SpectralCF [7] directly discovered all possible connectivity between users and items from the spectral domain of the user-item graph. However, SpectralCF fails to capture the information in high order. NGCF [8] further proposed employing the element-wise product between users and embeddings to augment the item features that users care about and the user features that items are selected by. Some works [8, 9, 10, 11, 12] observed that removing the non-linear activation not only simplifies the update operation, but also retains or enhances the recommendation performance. LightGCN [9] further removed the feature transformation to simplify the design of GCN. However, these changes only depend on the empirical results, without theoretical guarantees. In addition to the user-item interaction matrix, there exist studies combining the social network [13] and knowledge graph [14] to enhance the representations of users and items. In this paper, we mainly consider the user-item interaction matrix, i.e. bipartite graph.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

**Generalization Analysis for GNNs.** Due to the numerical success [15, 5, 16], several theoretical studies have been devoted to establishing generalization guarantees for GNNs. In inductive learning, [17] studied the generalization capability of GNNs by the VC-dimension, which is polynomial w.r.t. the number of parameters and the number of nodes in the graph. [18] provided the first data-dependent bounds via Rademacher Complexity for message passing GNNs and did not allow the analysis of graph information. Subsequently, [19] derived a tighter bound for GNNs based on the PAC-Bayes framework. In transductive learning, [20] analyzed the uniform stability of (single layer) GCNs and thereby deriving their generalization guarantees. They proved that the graph filter, which makes the largest absolute eigenvalue independent of the graph size, satisfies this uniform stability criterion. [21] derived a Rademacher complexity based bound for two layers GCN model. [22] provided a more informative bound according to the relations between normalized graph and input feature at the first layer. However, these works mentioned above only focused on node or graph classification tasks, different from recommendations. The recent works [23, 24] have studied the link prediction and graph embedding tasks; however, they did not concern with the specific models.

**Contributions and paper structure.** The main contributions are shown as follows:

1) We quantify the model complexity for GCN-based recommender systems on the bipartite graph via Covering number. And then, in Section 3, we relate the Covering number of function space with a specific radius to the generalization error under the inductive and transductive settings, leading to the generalization guarantees.

2) Motivated by the proven bound that generalization needs to compromise empirical risk and model complexity and the fact that the recommendation model overfits the sampled discrete data, we propose a simple but effective augmentation strategy, Item mixture (IMix) in Section 4. It makes discrete sample space continuous by mixing positive-negative item pairs about the same user. Notably, we theoretically demonstrate that it is beneficial for recommendation generalization.

3) In Section 5, extensive evaluations are conducted on two datasets, Gowalla and Yelp2018. First, to confirm the previous theoretical foundation, we present the corresponding numerical results, mainly including graph normalization and non-linear activation. Second, we validate the effectiveness of the proposed IMix on different models, and the experimental results consistently show that the recommendation performance is significantly improved after configuring IMix.

We conclude in Section 6. All proofs and more experimental results are provided in the appendix.

# 2 Preliminary

In this section, we first formulate the problem of collaborative filtering (CF) based recommender system. We then describe the GCN-based CF models in detail.

## 2.1 Problem Formulation

We use $u \in \mathcal{U}$ and $i \in \mathcal{I}$ to denote the user and item, and use $\mathcal{R} = \{(u,i)|u \in \mathcal{U}, i \in \mathcal{I}\}$ to denote the collected user-item interactions. Let $\mathcal{Z} = \{(u,i)|u \in \mathcal{U}, i \in \mathcal{I}\}$, $\mathcal{D}$ to be a fixed but unknown distribution over $\mathcal{Z} \times \mathcal{Z} \times \{-1, +1\}$. For any triplet $(z, z', y) \sim \mathcal{D}$, $y = +1$ if the preference between user-item pair $z$ is higher than $z'$, otherwise $y = -1$. Let $f \in \mathcal{F} : \mathcal{U} \times \mathcal{I} \to \mathbb{R}$ be a recommender system. For any user-item pair $z \in \mathcal{Z}$, the model predicts a preference score $f(z)$ by the learned user representation $\boldsymbol{e}_u$ and item representation $\boldsymbol{e}_i$. In this paper, we consider the inner-product as the preference function, i.e., $f(z) = \boldsymbol{e}_u^T \boldsymbol{e}_i$. To optimize the recommendation ability of $f$, we adopt a pairwise ranking loss as follows, for any $(z, z', y) \sim \mathcal{D}$,

$$loss = l(y(f(z) - f(z'))).$$

## 2.2 GCN-based Recommendation

Let $\boldsymbol{x}_u, \boldsymbol{x}_i \in \mathbb{R}^d$ be the user and item feature, where $d$ is the input dimension. Without loss of generality, we fix the collected user-item interactions as matrix $\boldsymbol{R} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$. Denote the normalized graph filter by

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{0} & \tilde{\boldsymbol{R}}_u \\ \tilde{\boldsymbol{R}}_i & \boldsymbol{0} \end{bmatrix} \text{ and } \boldsymbol{A} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{R} \\ \boldsymbol{R}^T & \boldsymbol{0} \end{bmatrix}.$$

Following [25], the existing GCN-based collaborative filtering models can be generalized into two steps. Note that we omit the update formula of items for the sake of brevity.

**(1) Update operation.** The most straightforward layer-wise updations are inspired by the GNN techniques, and they can be summarized as follows,

$$\mathbf{h}_u^{(l+1)} = \phi\left(\mathbf{W}_1^l \mathbf{h}_u^l + \sum_{i \in \mathcal{N}_u} \tilde{a}_{ui} \mathbf{W}_1^l \mathbf{h}_i^l\right), \tag{1}$$

where $\tilde{a}_{ui} = [\widetilde{\boldsymbol{R}}_u]_{ui}$ represents the assigned weights of user $u$ to item $i$. Some recent works [8] further adopt the element-wise product between nodes with their neighbors to leverage the consistency between user preferences and item attributes. This operation is formulated as,

$$\mathbf{h}_u^{(l+1)} = \phi\left(\mathbf{W}_1^l \mathbf{h}_u^l + \sum_{i \in \mathcal{N}_u} \tilde{a}_{ui} \left(\mathbf{W}_1^l \mathbf{h}_i^l + \mathbf{W}_2^l \left(\mathbf{h}_i^l \odot \mathbf{h}_u^l\right)\right)\right). \tag{2}$$

In this paper, we focus on the more complex operation defined in Eq. 2 since the Eq. 1 can be seen as a special case of it when $\boldsymbol{W}_2^l = \boldsymbol{O}$.

**(2) Final node representation.** The CF-based recommender systems require the representation of users and items for the final prediction task. The mainstream approaches include linear combination and concatenation, respectively defined as,

$$\text{Linear combination:} \qquad \boldsymbol{e}_u = \sum\nolimits_{l=0}^{L} \alpha^l \mathbf{h}_u^l,$$
$$\text{Concatenation:} \qquad \boldsymbol{e}_u = \mathbf{h}_u^0 \oplus \mathbf{h}_u^1 \oplus \cdots \oplus \mathbf{h}_u^L,$$

where $\alpha^l$ is a learnable parameter or hyper-parameter, $L$ is the network depth. In this paper, for convenience, we only consider the embeddings of the last layer as the final representation, i.e., $\boldsymbol{e}_u = \mathbf{h}_u^L$. We will prove that the generalization analysis can be easily extended to other integration operations in appendix.

# 3 Generalization for GCN-based Recommender Systems

In this section, we present the main results: generalization bounds for GCN-based recommendation models under inductive and transductive learning settings, respectively.

**Notations.** Let $[L] = \{1, \ldots, L\}$. For a vector $\boldsymbol{v} \in \mathbb{R}^n$, we denote its Euclidean norm by $\|\boldsymbol{v}\|_2 = \sqrt{\boldsymbol{v}^T \boldsymbol{v}}$, and the infinity norm by $\|\boldsymbol{v}\|_\infty = \max_i |\boldsymbol{v}_i|$. For a matrix $\boldsymbol{W} \in \mathbb{R}^{m \times n}$, we denote its Frobenius norm by $\|\boldsymbol{W}\|_F = \sqrt{trace(\boldsymbol{W}^T \boldsymbol{W})}$, the infinity norm by $\|\boldsymbol{W}\|_\infty = \max\limits_{1 \le i \le m} \sum_{j=1}^n |\boldsymbol{W}|_{ij}$, and the spectral norm by $\|\boldsymbol{W}\|_2 = \sigma_{\max}(\boldsymbol{W})$.

**Assumptions.** To establish the generalization guarantee, let us immediately make some mild assumptions that are easy to implement. For the loss function, assuming $l : \mathbb{R} \to [-B, B]$ is any bounded $C_l$-$lipschitz$ continuous function. For the non-linear activation function, we assume $\phi$ is $C_\phi$-$lipschitz$ continuous with some $C_\phi > 0$, $\phi(0) = 0$ . For the weight matrices, we assume that $\boldsymbol{W}_1^l, \boldsymbol{W}_2^l \in \mathbb{R}^{d \times d}$ are shared across layers, and have bounded norms $\|\boldsymbol{W}_1^l\|_2 \le B_1$ and $\|\boldsymbol{W}_2^l\|_2 \le B_2$. For the input feature, we assume $\max\limits_{u \in \mathcal{U}} \|\boldsymbol{x}_u\|_2 \le B_u$ and $\max\limits_{i \in \mathcal{I}} \|\boldsymbol{x}_i\|_2 \le B_i$.

## 3.1 Covering Number for GCN-based Recommender Systems

In classical learning theory, the complexity of function class is closely related to the generalization ability [26], and the typical complexity measurements are VC-Dimension [27] and Rademacher complexity [28]. In this paper, we adopt the Covering number [29] to investigate the model complexity of GCN-based recommender systems. Based on the properties of bipartite graphs, we study the complexity by combining the recommendations of GCN-based models for any user-item pairs.

**Lemma 1** (Covering number bound)**.** *Under Assumptions, we further assume that* $\|\phi(\mathbf{h}_u^l)\|_\infty, \|\phi(\mathbf{h}_i^l)\|_\infty \le b$ *for any* $u \in \mathcal{U}$, $i \in \mathcal{I}$, *and* $l = [L]$, *let* $\gamma = \|\widetilde{\boldsymbol{A}}\|_\infty$. *Given a sam-*

*ple set $S$ with size $n$, the Covering number of $\mathcal{F}$ over $S$ with specific $\epsilon$ is bounded as*

$$\log \mathcal{N}\left(\mathcal{F}|S, \epsilon, \|\cdot\|\right) \leq d^2 \log\left(1 + \frac{4(\gamma+1)ZB_1\sqrt{nd}}{\epsilon}\right)\left(1 + \frac{4\gamma bZB_2\sqrt{nd}}{\epsilon}\right).$$

*Moreover, when $\epsilon \leq 1 + 4Z\sqrt{nd}\max\{(\gamma+1)B_1, \gamma bB_2\}$,*

$$\begin{aligned}
&\log \mathcal{N}\left(\mathcal{F}|S, \epsilon, \|\cdot\|\right) \leq 2d^2 \log \frac{8Z(\gamma+1)\sqrt{ndB_1B_2b}}{\epsilon}, \ where \\
&Z = C_\phi \mathcal{C}^{2L-1}(B_u + B_i)^2 \frac{(2\mathcal{C})^L - 1}{2\mathcal{C} - 1}, \\
&\mathcal{C} = C_\phi[B_1 + \gamma(B_1 + B_2 b)].
\end{aligned}$$

**Remark 1.** The key idea of the proof is to exploit the properties of the bipartite graph in the recommender system. To investigate the complexity of the recommendation for any user-item pair generated by GCN-based models, we establish the connection between the complexities of function space and weight space. Here $\mathcal{C}$ describes the layer-wise update complexity, which is the important term for the following generalization analysis. According to the norm bound of hidden states, the assumption can be naturally satisfied when the activation function is bounded. And we will discuss the role of unbounded activation in Section 3. Note that the layer-wise complexity reduces to $\mathcal{C} = C_\phi(\gamma+1)B_1$ when ignoring element-wise between users and items, which means that there is no need to discuss the value of $b$.

### 3.2 Generalization Bound for Inductive & Transductive Learning

Transductive and inductive learning are both commonly used in GNNs and recommender systems. The primary difference between these two settings is whether the testing set depends on the training set. The testing samples in inductive learning are drawn from some unknown distribution, while in transductive learning, they are sampled without replacement from a fixed data set. In the following, we first relate the generalization error for two settings to the Covering number of GCN-based recommendations, then derive the final generalization guarantees using Lemma 1.

**Inductive Learning.** Given a training set $S_m = \{(z_i, z_i', y_i)\}_{i=1}^m$, we assume that all samples in $S_m$ are i.i.d. according to $\mathcal{D}$, denoted as $S_m \sim \mathcal{D}^m$. Under this inductive setting, the empirical error over a sample $S_m$ and the corresponding generalization error is defined as follows,

$$\begin{aligned}
&\text{Empirical error:} \qquad \hat{\mathcal{L}}_m(f) = \frac{1}{m}\sum\nolimits_{(z_i, z_i', y_i)\in S_m} l(y_i(f(z_i) - f(z_i')), \\
&\text{Generalization error:} \qquad \mathcal{L}(f) = \mathbb{E}_{(z,z',y)\sim\mathcal{D}}[l(y(f(z) - f(z'))].
\end{aligned} \tag{3}$$

Under inductive learning, we can derive the following generalization error bound by the complexity term according to the Covering number.

**Lemma 2.** *Let $\mathcal{F}$ be a real-valued function class taking values in $[-e, e]$, and assume that $\mathbf{0} \in \mathcal{F}$. Under assumptions, for any function $f$ in a class $\mathcal{F}$, with probability of at least $1-\delta$ over an i.i.d. size-$m$ training set, we have*

$$\mathcal{L}(f) \leq \hat{\mathcal{L}}_m(f) + 4C_l \inf_{\alpha>0}\left(\frac{4\alpha}{\sqrt{m}} + \frac{12}{m}\int_\alpha^{2e\sqrt{m}} \sqrt{\log \mathcal{N}\left(\mathcal{F}|S_m, \epsilon, \|\cdot\|\right)}d\epsilon\right) + 4B\sqrt{\frac{2\log 4/\delta}{m}}.$$

**Transductive Learning.** For the transductive learning setting, the full samples are observed prior to learning. Given a fixed set $S = \{(z_i, z_i', y_i)\}_{i=1}^{m+u}$, a labeled training set $S_m$ with size $m$ is selected from $S$ uniformly at random. The goal of transductive learning is to predict the labels of the samples in testing set $S_u = S \backslash S_m$. Under this transductive setting, the formulation of empirical error $\hat{\mathcal{L}}_m(f)$ is similar to Eq. 3, and the generalization error is defined as follows,

$$\hat{\mathcal{L}}_u(f) = \frac{1}{u}\sum\nolimits_{(z_i, z_i', y_i)\in S_u} l(y_i(f(z_i) - f(z_i')).$$

Following the same strategy adopted in proving Lemma 2, we again need to perform a generalization bound under transductive setting via Covering number of function class. As shown above, the

generalization and empirical error under transductive setting do not depend on any underlying distributions. The important difference induces two versions of generalization analysis. For the transductive learning, we consider the transductive Rademacher complexity [30] defined on both the training and testing data. Then the Covering number bounded generalization error can be derived by the interdependence between model complexity measurements [31, 32].

**Lemma 3.** *Let $\mathcal{F}$ be a real-valued function class taking values in $[-e, e]$, and assume that $\mathbf{0} \in \mathcal{F}$. Let $Q_1 = \frac{1}{u} + \frac{1}{m}, Q_2 = \frac{m+u}{(m+u-1/2)(1-1/2(\max(m,u)))}$ and $c_0 < 5.05$. Under assumptions, for any function $f$ in a class $\mathcal{F}$, with probability of at least $1-\delta$ over random partitions of $S$, we have*

$$\mathcal{L}_u(f) \leq \hat{\mathcal{L}}_m(f) + 2C_l \inf_{\alpha>0} \left( \frac{4\alpha}{\sqrt{m+u}} + \frac{12}{m+u} \int_{\alpha}^{2e\sqrt{m+u}} \sqrt{\log \mathcal{N}(\mathcal{F}|S, \epsilon, \|\cdot\|)} d\epsilon \right) + Bc_0 Q_1 \sqrt{\min(m,u)} + 2B\sqrt{\frac{Q_2}{2} Q_1 \ln \frac{1}{\delta}}.$$

Armed with Lemma 2, Lemma 3 and Lemma 1, we now present the generalization error bound for inductive and transductive learning.

**Theorem 1** (Generalization Bound)**.** *Under assumptions, for any function $f$ in a class $\mathcal{F}$, in inductive learning, with probability of at least $1-\delta$, we have,*

$$\mathcal{L}(f) \leq \hat{\mathcal{L}}_m(f) + \frac{24C_l}{\sqrt{m}} \mathcal{C}^{2L} (B_u + B_i)^2 d \sqrt{2\log(8mZ(\gamma+1)\sqrt{dB_1B_2b})} + \frac{16C_l}{m} + 4B\sqrt{\frac{2\log 4/\delta}{m}}. \tag{4}$$

*Accordingly, in transductive learning, with probability of at least $1-\delta$, we have,*

$$\mathcal{L}_u(f) \leq \hat{\mathcal{L}}_m(f) + \frac{24C_l}{\sqrt{m+u}} \mathcal{C}^{2L} (B_u + B_i)^2 d \sqrt{2\log(8(m+u)Z(\gamma+1)\sqrt{dB_1B_2b})} + \frac{4C_l\sqrt{2mu}}{(m+u)^2} + Bc_0 Q_1 \sqrt{\min(m,u)} + 2B\sqrt{\frac{Q_2}{2} Q_1 \ln \frac{1}{\delta}}.$$

**Remark 2.** Theorem 1 states that the gap between generalization error and empirical error converges uniformly to 0 with the rate of $\sqrt{(\log m)/m}$. We first discuss several obvious findings about the dependency of the bound on some terms. For the hidden dimension $d$, the generalization bound scales as $\mathcal{O}(d\sqrt{\log d})$; for the input feature norms $B_u$ and $B_i$, the bound scales as $\mathcal{O}((B_u + B_i)^2\sqrt{\log(B_u + B_i)})$. The most important term is the layer-wise update complexity $\mathcal{C}$, which depends on the value of $\gamma$ according to the graph structure, the spectral norms of weight matrices $B_1$ and $B_2$, and the value of $C_\phi$ and $b$ according to the non-linear activation function. A natural idea is to constrain the weight matrices during training such that the generalization bound would not grow exponentially with the network depth $L$, which is a regularization techniques commonly used in machine learning. The subsequent discussions focus on the $\gamma$, $C_\phi$ and $b$ to investigate the influence of graph normalization and non-linear activation functions.

**Role of graph normalization.** It is easy to see that $\gamma$ is equal to the maximum degree $d_{\max}$ of the user-item interaction graph without graph normalization. Therefore, fixed other terms in $\mathcal{C}$, the generalization bound grows as $O(d_{\max}^L)$. For the random-walk graph, i.e., $\widetilde{\boldsymbol{A}} = \boldsymbol{D}^{-1}\boldsymbol{A}$, $\gamma = 1$, and for symmetric normalized graph, i.e., $\widetilde{\boldsymbol{A}} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$, $\gamma \leq \sqrt{d_{\max}/d_{\min}}$, and hence, the growth is much smaller than the unnormalized graph.

**Role of non-linear activation.** As pointed above, the activation functions affect the generalization performance through two quantities: the Lipschitz constant $C_\phi$ and the norm bound of hidden states $b$. Due to the most activation functions satisfy the condition $C_\phi \leq 1$, and particularly LeakyReLU, ReLU, and Tanh are 1-$Lipschitz$, $C_\phi$ presents negligible influence. For the norm bound $b$, while the bounded activation naturally provides explicit value, the unbounded function can also satisfy the assumption by constraints on the weight matrices. As discussed in Remark 2, weight regularization is widely used in machine learning and can help improve model generalization performance to prevent overfitting. In addition, one recent work [33] gives a new theoretical finding that non-linearity does not contribute to improving expressive power of GNNs. Therefore, this generalization bound provides theoretical support for removing the activation function to simplify the model without incurring a performance penalty.

# 4 Item Mixture Powered Recommendation

The upper bound in Theorem 1 inspires us that the generalization needs to balance empirical risk and model complexity. Most models [34, 8] inevitably lead to excessive complexity in pursuit of high expressiveness, but simple models may lead to higher empirical errors, so it is valuable to reduce the complexity while ensuring model expressiveness. In addition, the recommendation algorithm is trained based on the limited discrete data obtained by sampling, making the model perform well in the discrete space of sampling and prone to overfitting. Considering the above and inspired by Mixup [35], we propose Item Mixture (IMix) to power recommendation, which forces the model to deal with regions between discrete samples continuously by mixing any two positive-negative item pairs. In addition, IMix does not make changes to the model, while we can theoretically prove that it reduces the model complexity, and then enjoys generalization. Below we give specific details.

Since the original representations of items are discrete and unprocessable (e.g., orange or apple), we suggest mixing their embeddings. Specifically, for a triplet $(\boldsymbol{e}_u, \boldsymbol{e}_i, \boldsymbol{e}_{i'})$ computed according to Section 2.2, where user $u$ prefers item $i$ to $i'$, we arbitrarily sample another embedding triplet $(\boldsymbol{e}_u, \boldsymbol{e}_i, \boldsymbol{e}_{i'})$ of the same user. We can obtain mixed embedding triplet $(\boldsymbol{e}_u, \widetilde{\boldsymbol{e}}_i, \widetilde{\boldsymbol{e}}_{i'})$, where $\widetilde{\boldsymbol{e}}_i = \lambda \boldsymbol{e}_i + (1-\lambda)\boldsymbol{e}_j$, and $\widetilde{\boldsymbol{e}}_{i'} = \lambda \boldsymbol{e}_{i'} + (1-\lambda)\boldsymbol{e}_{j'}$. $\lambda \sim Beta(\alpha, \beta)$ for $\alpha, \beta > 0$, and we set $D_\lambda = Beta(\alpha, \beta)$. So in IMix, we will train the recommendation model on the new mixed embeddings. The standard loss and IMix loss are defined as

$$\begin{aligned} &\text{Standard loss:} \qquad \mathcal{L}_m^{std} = \frac{1}{m}\sum\nolimits_{k=1}^{m} l(\boldsymbol{e}_{u_k}^T(\boldsymbol{e}_{i_k} - \boldsymbol{e}_{i'_k})), \\ &\text{IMix loss:} \qquad \mathcal{L}_m^{IMix} = \frac{1}{m}\sum\nolimits_{k=1}^{m} l(\boldsymbol{e}_{u_k}^T(\widetilde{\boldsymbol{e}}_{i_k} - \widetilde{\boldsymbol{e}}_{i'_k})). \end{aligned} \tag{5}$$

Considering the motivation above, IMix preserves the model expressiveness without modification of the model structure. Besides, it transforms the learning of discrete sample pairs into the learning of continuous regions between sample pairs, which may helps to alleviate the overfitting.

Notably, although IMix is inspired by the generalization bound in Theorem 1, it is decoupled from the model structure. Therefore, it is theoretically applicable to any embedding-based recommendation model, and we will show its performance in non-GCN-based models in Appendix A.3.

Next, we theoretically guarantee the effectiveness of the proposed IMix. Following a similar approach in [36], we adopt second-order Taylor expansion to state the IMix training loss is approximately equivalent to standard empirical error with a regularization term.

**Lemma 4.** *Suppose that* $\hat{\Sigma}_X = \frac{1}{m}\sum_{k=1}^{m}(\boldsymbol{e}_{i_k} - \boldsymbol{e}_{i'_k})(\boldsymbol{e}_{i_k} - \boldsymbol{e}_{i'_k})^T$ *and* $\mathcal{D}_x$ *is the empirical distribution of the items in training dataset* $S_m$*. If the loss function* $l$ *is twice differentiable, then the IMix loss defined in Eq. 5 has the following approximation,*

$$\mathcal{L}_m^{mix}(S_m) \approx \mathcal{L}_m^{std}(S_m) + \frac{1}{m}\mathbb{E}_{\lambda\sim\mathcal{D}_\lambda}\left(\frac{(1-\lambda)^2}{\lambda^2}\right)\sum\nolimits_{k=1}^{m} l''(\boldsymbol{e}_{u_k}^T(\boldsymbol{e}_{u_k} - \boldsymbol{e}_{i'_i}))\boldsymbol{e}_{u_k}^T\hat{\Sigma}_X\boldsymbol{e}_{u_k}.$$

Given the above regularized version, let $\tau > 0$ and $\Sigma_X = \mathbb{E}\left[(\boldsymbol{e}_i - \boldsymbol{e}_{i'})(\boldsymbol{e}_i - \boldsymbol{e}_{i'})^T\right]$, we shed light upon the generalization bound by investigating the following function class:

$$\mathcal{F}_\tau^{mix} = \{\mathcal{F}|\mathbb{E}_{(u,i,i')}\left[l''(\boldsymbol{e}_u^T(\boldsymbol{e}_u - \boldsymbol{e}_{i'}))\boldsymbol{e}_u^T\Sigma_X\boldsymbol{e}_u\right] \le \tau\}.,$$

**Remark 3.** Assuming that the loss function $l$ satisfies $l''(x) \ge \rho, x \in \mathbb{R}$ on the basis of the previous assumptions. Then with probability of at least $1-\delta$, for any function $f \in \mathcal{F}_\tau^{mix}$, we have,

$$\mathcal{L}(f) \le \hat{\mathcal{L}}(f) + 4C_l\sqrt{\frac{rank(\Sigma_X)\tau}{\rho m}} + 4B\sqrt{\frac{2\log 4/\delta}{m}}.$$

If we consider the function class with a general regularization technique on the weight matrices, as we discussed above, the constraint on weights will ultimately control the node representation. Therefore, for the general condition, we focus on the function class $\mathcal{F}_\tau^{std} = \{\mathcal{F}|\mathbb{E}_{(u,i,i')}\left[\|\boldsymbol{e}_u\|_2^2 + \|\boldsymbol{e}_i\|_2^2\right] \le \tau\}$, then the similar proof strategy would yield the generalization error bound $\mathcal{L}(f) \le \hat{\mathcal{L}}(f) + 4C_l\sqrt{\frac{\tau^2}{m}} + 4B\sqrt{\frac{2\log 4/\delta}{m}}$. we can conclude that when $rank(\Sigma_X) < \tau$, IMix has a better generalization. It is easily met because $\gamma$ tends to be large to pursue high performance. For example, in the experiment, the embedding size $d$ is 64 (noted that $rank(\Sigma_X) < d$), while the minimum value of $\gamma$ is 130.8.

# 5 Experiments

In this section, we conduct extensive experiments to validate our theoretical findings and the effectiveness of the proposed IMix.

**Dataset.** We follow existing work [8, 9], using two datasets: (1) Gowalla, contains 1,027,370 check-in information from 29,858 users to 40,981 locations; (2) Yelp2018, comprises 31,668 users who have reviewed 38,048 items (e.g., restaurants, bars) with 1,561,406 times. The training set, validation set, and test set are randomly sampled in a ratio of $7:1:2$.

**Model Structure.** Two competitive GCN-based algorithms, NGCF [8] and LightGCN [9], are used to construct experiments. Among them, NGCF explicitly models the high-order connectivity between users and items through message construction and message aggregation, thereby improving the expressiveness of embedding. LightGCN, on the other hand, is keenly aware that the nonlinear activation function and feature transformation of NGCF have no specific semantics in the recommendation, so it is simplified to only a domain message aggregation module. Correspondingly, these models are configured with the proposed IMix denoted as NGCF-IMix and LightGCN-IMix, respectively.

We use the public implementation version of LightGCN [1] and follow the default parameters of their algorithms unless otherwise specified. More details are provided in Appendix A.1.

**Metrics.** The recommendation performance is measured by the positive items' rankings in the test set. We use two ranking metrics widely used in recommender systems [37, 38, 39]: Recall and NDCG. Recall at a cutoff K, denoted as Recall@K, denotes the proportion of positive items predicted to be correct. NDCG at a cutoff K, denoted as NDCG@K, measures the reward for the positive items' position in the top-K recommendation list. For these two metrics, we truncate the ranked list K to 20.

## 5.1 Numerical Discussion

In this subsection, we discuss the implication of the generalization bound defined in Theorem. 1, and the results are built on NGCF because it is more similar to traditional GCNs.

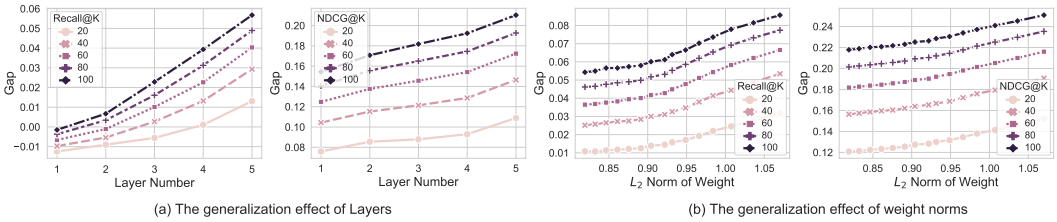
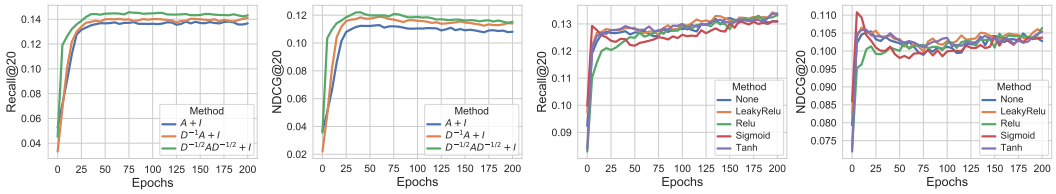



Figure 1: (a): the effect of number of layers $L$ on Generalization Performance on Gowalla. (b): the effect of weight norms $B_1$, $B_2$ on Generalization Performance on Gowalla.

**Quantitative Analysis.** We investigate the effects of two major values on the generalization gap, which represents the differences between recommendation performances in training and testing sets. We consider the number of layers $L$, as shown in the left part of Fig. 1. As the number of layers increases, the generalization gaps of ndcg@K and recall@K significantly increase. The empirical results imply that the generalization error is proportional to the number of layers, consistent with the theoretical analysis. We also consider the spectral norms of weight matrices $B_1$ and $B_2$, as shown in the right part of Fig. 1. We compare the generalization abilities with different weight norms. As the norms increase, the generalization gaps increase steadily. Compared with the number of layers, we can observe that the weight norms increase more slowly. The theoretical reason is that the generalization performance is $L$-th order with respect to the weight norms $B_1$ and $B_2$, and exponential with respect to the number of layers $L$.

**The Role of Normalized Graph.** We compare the generalization performances with different normalized graphs, as shown in the left part of Fig. 2. It is clear that the two normalized graphs have higher generalizations than the unnormalized graph. This result is consistent with the theoretical findings. The comparison between two normalized graphs depends on the specific user-item interaction graph.

[1] `https://github.com/kuandeng/LightGCN`




(a) The generalization effect of normalized graph (b) The generalization effect of activation function

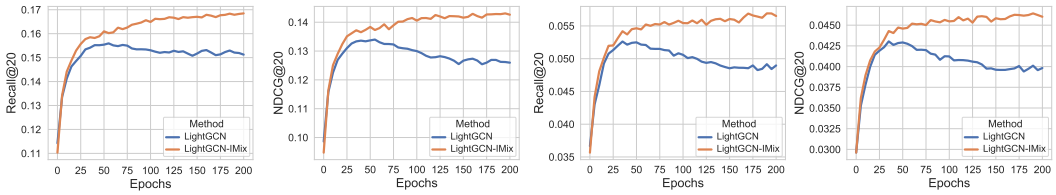
Figure 2: (a): the effect of normalized graph on Generalization Performance on Gowalla. (b): the effect of activation function on Generalization Performance on Gowalla.

**The Role of Non-linear Activation Function.** We study the role of non-linear activation function in GCN-based recommender systems. Since most activation functions are all 1-Lipschitz continuous, we compare the generalization performances with the popular activation functions, including LeakyReLU, ReLU, Sigmoid, and Tanh. Furthermore, we compare these results with the model without activation function. As shown in the right part of Fig. 2, these models present similar performances, which provide theoretical support for removing activation functions. Although the Sigmoid presents great recommendations at first, it overfitted as the training continued, while the performance of other models is still improved, and their optimal performance tends to be unified.

## 5.2 Performance Analysis of IMix

Table 1: Overall evaluation. All models' performance is improved after configured with IMix.

| Dataset | | Gowalla | | Yelp2018 | |
|---|---|---|---|---|---|
| **Layer #** | **Method** | **Recall@20** | **NDCG@20** | **Recall@20** | **NDCG@20** |
| **1 layer** | NGCF | 0.1308 | 0.1032 | 0.0419 | 0.0336 |
| | NGCF-IMix | 0.1451(+10.93%) | 0.1146(+11.05%) | 0.0492(+17.42%) | 0.0360(+7.14%) |
| **2 layer** | NGCF | 0.1274 | 0.1050 | 0.0431 | 0.0348 |
| | NGCF-IMix | 0.1454(+14.13%) | 0.1164(+10.86%) | 0.0491(+13.92%) | 0.0391(+12.36%) |
| **3 layer** | NGCF | 0.1303 | 0.1082 | 0.0435 | 0.0353 |
| | NGCF-IMix | 0.1496(+14.81%) | 0.1191(+10.07%) | 0.0504(+15.86%) | 0.0405(+14.73%) |
| **1 layer** | LightGCN | 0.1556 | 0.1340 | 0.0526 | 0.0427 |
| | LightGCN-IMix | 0.1699(+9.19%) | 0.1436(+7.16%) | 0.0562(+6.84%) | 0.0458(+7.26%) |
| **2 layer** | LightGCN | 0.1672 | 0.1425 | 0.0564 | 0.0462 |
| | LightGCN-IMix | 0.1765(+5.56%) | 0.1507(+5.75%) | 0.0621(+10.10%) | 0.0507(+9.74%) |
| **3 layer** | LightGCN | 0.1759 | 0.1500 | 0.0602 | 0.0492 |
| | LightGCN-IMix | 0.1793(+1.93%) | 0.1524(+1.6%) | 0.0644(+6.98%) | 0.0527(+7.11%) |




(a) Gowalla (b) Yelp2018

Figure 3: Test curves (Recall@20 and NDCG@20) for 200 epochs of training.

**Effectiveness Validation.** We apply the proposed IMix to NGCF and LightGCN and perform a detailed comparison, as shown in Table 1. The percentage of relative improvement is shown in parentheses. Obviously, the proposed augmentation strategy (corresponding to NGCF-IMix and LightGCN-IMix) significantly improves the performance of the corresponding models (NGCF and LightGCN). For example, in the Gowalla dataset, the best performance of NGCF on Recall@20 is 0.1308, while with IMix configured, the performance improves to an encouraging 0.1496. In addition,

the improvements of IMix on NGCF are larger than that on LightGCN. We reasonably believe that LightGCN is more expressive (better performance), and the benefit of the boosting strategy decreases accordingly. Nonetheless, the performance on LightGCN is also a pleasant surprise.

Besides, Fig. 3 presents the performance curves regarding Recall and NDCG for the test set during the training of LightGCN-based models (LightGCN and LightGCN-IMix). Throughout the training process, LightGCN-IMix consistently outperforms LightGCN. After 50 epochs, LightGCN shows different degrees of overfitting. On the contrary, the performance of LightGCN-IMix is still further improved, and their gap is gradually enlarged. We reasonably believe that the proposed strategy adds a data-dependent regularization term to LightGCN (Lemma 4), which helps to alleviate overfitting.

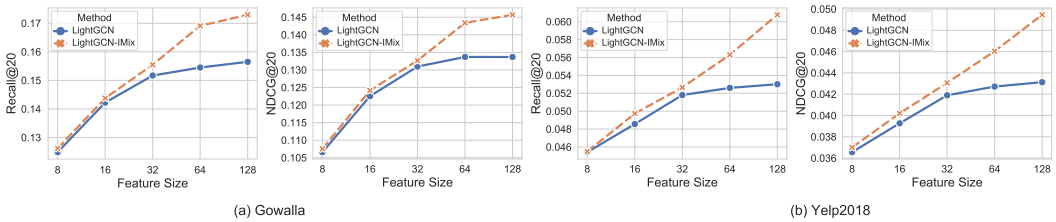



Figure 4: Performance comparison of different embedding sizes.

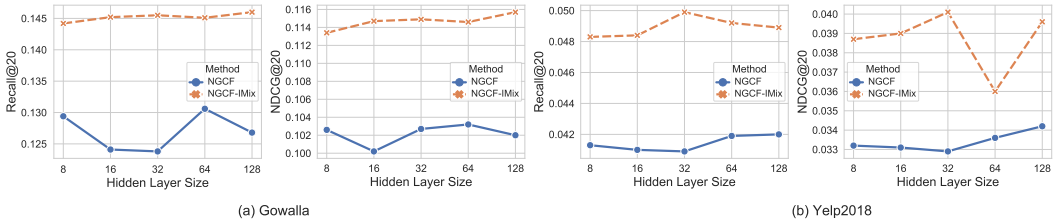



Figure 5: Performance comparison of different hidden layer size.

**Sensitivity w.r.t. Model Hyperparameters.** We also explore the impact of the model's hyperparameters: different feature sizes (dimension of $x_u$ and $x_i$) and hidden layer sizes (dimension of $W_1^l$ and $W_2^l$). Here, the number of layers is set to 1, and the results are shown in Fig. 4 and Fig. 5.

In Fig. 4, we find that the larger the feature size, the more beneficial it is to the recommendation. Moreover, LightGCN-IMix is better than LightGCN, and even more, the better the performance of the original model, the better the improvement after configuring IMix, which highlights the superiority of the proposed IMix.

Fig. 5 reveals that the hidden layer size does not cause a significant change in model performance, while when the hidden layer size is only 8, the performance is sufficiently satisfactory, which provides support for the practicality of GCN-based algorithms. In addition, with similar conclusions to Fig. 4 and Table 1, the IMix-configured models outperform the original models in all cases. In conclusion, we validate the effectiveness of IMix.

## 6 Conclusion and Future Work

GCNs achieve superior empirical success in recommender systems but lack theoretical understanding. In this paper, we provide a model complexity analysis for GCN-based recommendations via Covering number, and then induce the generalization error bounds in inductive and transductive learning. We report several theoretical findings and again verify them via experimental results. Inspired by the generalization analysis, we introduce an augmentation strategy, Item Mixture, to improve the recommendation generalization by mixing item embeddings.

There are several interesting future directions. First, it would be interesting to further analyze the user-item bipartite graph with social relationship and knowledge graph. The second is to extend our analysis to investigate the plausibility of the other GNN-based variants. The third is to study the role of graph spectral properties in generalization ability.

# References


[1] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[2] 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*, pages 974–983, 2018.

[3] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. Parameter-efficient transfer from sequential behaviors for user modeling and recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1469–1478, 2020.

[4] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[5] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.

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

[7] 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*, pages 311–319, 2018.

[8] 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*, pages 165–174, 2019.

[9] 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*, pages 639–648, 2020.

[10] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:27–34, 04 2020.

[11] Le Wu, Yonghui Yang, Kun Zhang, Richang Hong, Yanjie Fu, and Meng Wang. Joint item recommendation and attribute inference: An adaptive graph convolutional network approach. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 679–688, 2020.

[12] Zhiwei Liu, Lin Meng, Fei Jiang, Jiawei Zhang, and Philip S Yu. Deoscillated graph collaborative filtering. *arXiv preprint arXiv:2011.02100*, 2020.

[13] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference*, pages 417–426, 2019.

[14] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 968–977, 2019.

[15] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.

[16] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

[17] Franco Scarselli, Ah Chung Tsoi, and Markus Hagenbuchner. The vapnik–chervonenkis dimension of graph and recursive neural networks. *Neural Networks*, 108:248–259, 2018.

[18] Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pages 3419–3430. PMLR, 2020.

[19] Renjie Liao, Raquel Urtasun, and Richard Zemel. A pac-bayesian approach to generalization bounds for graph neural networks. *arXiv preprint arXiv:2012.07690*, 2020.

[20] Saurabh Verma and Zhi-Li Zhang. Stability and generalization of graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1539–1548, 2019.

[21] Shaogao Lv. Generalization bounds for graph convolutional neural networks via rademacher complexity. *arXiv preprint arXiv:2102.10234*, 2021.

[22] Pascal Esser, Leena Chennuru Vankadara, and Debarghya Ghoshdastidar. Learning theory can (sometimes) explain generalisation in graph neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[23] Yihan Gao, Chao Zhang, Jian Peng, and Aditya Parameswaran. The importance of norm regularization in linear graph embedding: Theoretical analysis and empirical demonstration. *arXiv preprint arXiv:1802.03560*, 2018.

[24] Atsushi Suzuki, Atsushi Nitanda, Linchuan Xu, Kenji Yamanishi, Marc Cavazza, et al. Generalization bounds for graph embedding using negative sampling: Linear vs hyperbolic. *Advances in Neural Information Processing Systems*, 34, 2021.

[25] Shiwen Wu, Fei Sun, Wentao Zhang, and Bin Cui. Graph neural networks in recommender systems: a survey. *arXiv preprint arXiv:2011.02260*, 2020.

[26] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[27] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

[28] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[29] Ding-Xuan Zhou. The covering number in learning theory. *Journal of Complexity*, 18(3):739–767, 2002.

[30] Ran El-Yaniv and Dmitry Pechyony. Transductive rademacher complexity and its applications. *Journal of Artificial Intelligence Research*, 35:193–234, 2009.

[31] Minshuo Chen, Xingguo Li, and Tuo Zhao. On generalization bounds of a family of recurrent neural networks. *arXiv preprint arXiv:1910.12947*, 2019.

[32] Peter Bartlett, Dylan J Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1706.08498*, 2017.

[33] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

[34] 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*, pages 726–735, 2021.

[35] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[36] Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization? *arXiv preprint arXiv:2010.04819*, 2020.

[37] Haoyu Wang, Defu Lian, and Yong Ge. Binarized collaborative filtering with distilling graph convolutional networks. *arXiv preprint arXiv:1906.01829*, 2019.

[38] 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*, pages 2020–2029, 2020.

[39] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. Disentangled graph collaborative filtering. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 1001–1010, 2020.

## Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]