Debiased Contrastive Loss for Collaborative Filtering

Zhuang Liu[1], Yunpu Ma[2], Haoxuan Li[3], Marcel Hildebrandt[4], Yuanxin Ouyang[1(✉)], and Zhang Xiong[3]

[1] State Key Laboratory of Software Development Environment, Beihang University, Beijing, China
{liuzhuang,oyyx}@buaa.edu.cn
[2] Lehrstuhl für Datenbanksysteme und Data Mining, Ludwig-Maximilians-Universität München, Munich, Germany
[3] Engineering Research Center of Advanced Computer Application Technology, Ministry of Education, Beihang University, Beijing, China
xiongz@buaa.edu.cn
[4] University of Munich and Siemens AG, Munich, Germany
Marcel.hildebrandt@Siemens.com

Abstract. Collaborative filtering (CF) is the most fundamental technique in recommender systems, which reveals user preference by implicit feedback. Generally, binary cross-entropy or bayesian personalized ranking are usually employed as the loss function to optimize model parameters. Recently, the sampled softmax loss has been proposed to enhance the sampling efficiency, which adopts an in-batch sample strategy. However, it suffers from the sample bias issue, which unavoidably introduces false negative instances, resulting inaccurate representations of users' genuine interests. To address this problem, we propose a debiased contrastive loss, incorporating a bias correction probability to alleviate the sample bias. We integrate the proposed method into several matrix factorizations (MF) and graph neural network-based (GNN) recommendation models. Besides, we theoretically analyze the effectiveness of our methods in automatically mining the hard negative instances. Experimental results on three public benchmarks demonstrate that the proposed debiased contrastive loss can augment several existing MF and GNN-based CF models and outperform popular learning objectives in the recommendation. Additionally, we demonstrate that our method substantially enhances training efficiency.

Keywords: collaborative filtering · debiased contrastive learning · sample bias · matrix factorization · graph neural network

This work was partially supported by the National Natural Science Foundation of China (No. 61977002), the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A and the State Key Laboratory of Software Development Environment of China (No. SKLSDE-2022ZX-14).

Z. Jin et al. (Eds.): KSEM 2023, LNAI 14119, pp. 94–105, 2023.
https://doi.org/10.1007/978-3-031-40289-0_8

1 Introduction

Recommender system plays an essential role in helping users filter information. Collaborative filtering (CF) is the most popular recommendation technique, which assumes that similar users would exhibit similar preference on items [1]. Many classical CF algorithms employ matrix factorizations (MFs) corresponding to scoring user-item interactions via inner products. In turn, MFs exhibit rather limited expressiveness and the inability to capture complex patterns. As a potential remedy, graph neural networks (GNNs) are nowadays widely employed for the recommendation task. GNNs have the ability to create comprehensive representations of users and items, enabling accurate predictions of new interactions and raising the bar on different benchmark measures [1–4].

To train and optimize the model parameters, most recommendation algorithms adopt binary cross-entropy (BCE) [5] or bayesian personalized ranking (BPR) [6] as the loss function. They are both based on random sample strategies. With the success of contrastive learning [7], sampled softmax (SSM) loss [8,9] is proposed, which adopts the in-batch sample strategy and regards the other instances in the same batch as negatives of the current instance. It significantly improves the sampling efficiency and brings better recommendation performance.

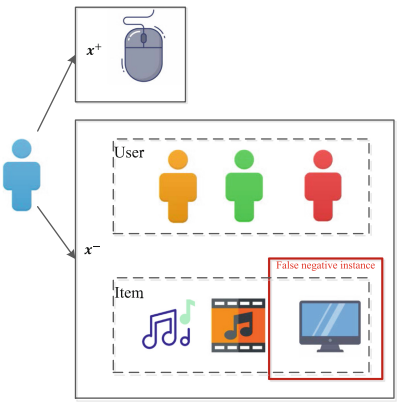

Fig. 1. An example of the sample bias.

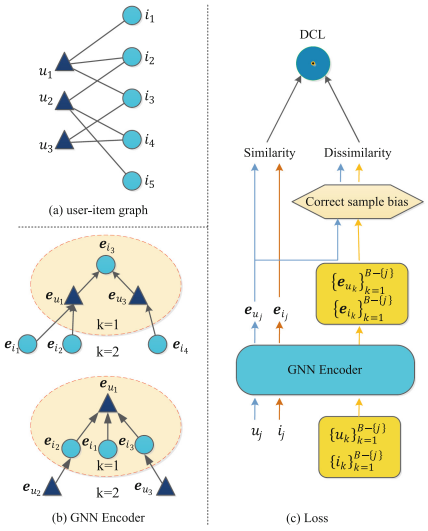

Fig. 2. The architecture of the proposed method.

However, the in-batch sample strategy does not consider that negative instances may be consistent with the users' interests. This phenomenon, which we refer to as *sample bias*, can cause a significant performance drop [10]. As shown in Fig. 1, a user is keen on electronic products, and items like the monitor in the red

box are consistent with the user's genuine interests. They will likely be sampled as negatives in the same batch and, thus, are described as false negative instances.

In this paper, we propose a debiased contrastive loss (DCL) for the CF model to alleviate the sample bias. In particular, given a user-item pair in a batch, we consider all other instances in the current batch as negatives and then incorporate an explicit correction probability τ^+ to correct the negative scores. Here, τ^+ denotes the proportion of false negatives. We verify the effectiveness of DCL in three public datasets.

To summarize, our main contributions are as follows:

- We propose DCL, a debiased contrastive loss for the general CF model, which can alleviate the sample bias and brings better performance.
- We theoretically analyze that DCL could automatically mine the hard negative instances for users.
- Experimental results on three public datasets demonstrate that our method augments several CF models by a significant margin and outperforms several existing representative loss functions in recommendation while being computationally efficient.

2 Related Work

We briefly review some works closely related to ours, including collaborative filtering and corresponding learning objectives.

Collaborative filtering is the most popular recommendation technique. Many CF algorithms employ MFs corresponding to scoring user-item interactions via products [6,11]. The recent neural recommender models like DMF [12] and NCF [13] enhance the interaction modeling with neural networks. Moreover, formed by user-item interactions, bipartite graphs often serve as a common representation, with edges representing the connections between users and items. With the success of graph neural networks (GNNs), various GNN-based methods have been proposed to improve the recommendation performance, exploiting the user-item interaction graph to infer user preferences. For example, GC-MC [14] applies the graph convolution network (GCN) [15] on the user-item graph. However, it only employs one convolutional layer to exploit the direct connections between users and items. To obtain better performance, NGCF [1] exploits the user-item graph structure by propagating embeddings on it, which allows capturing high-order connectivity. LightGCN [4] is a trimmed-down version of NGCF.

The popular objectives for the CF model mainly consist of BCE loss [5,16], BPR loss [4,6] , and softmax loss [8,9]. BCE loss considers the recommendation task as a binary classification problem. It treats the observed interactions as positives and unobserved interactions as negatives. Instead of scoring the interactions strictly positive or negative, BPR loss aims to encourage the score of an observed interaction to be higher than its unobserved counterparts. To improve the sampling efficiency, [8] proposes CLRec, a cached-batch contrastive loss for recommendation. It designs a first-in-first-out queue to store the instances and fetches those in the queue as negatives. SSM loss [9] is a special version of CLRec, which treats the other instances in the same batch as negatives.

3 Methodology

3.1 Problem Setup

Consider the indexed set of n users $\mathcal{U} = \{u_1, u_2, \cdots, u_n\}$ and m items $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$, respectively. We assume that $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the undirected user-item interaction graph. The vertex set is given by $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$. Moreover, we draw an edge $\{u, i\} \in \mathcal{E}$ if and only if the user $u \in \mathcal{U}$ has interacted with item $i \in \mathcal{I}$. An example of a user-item interaction graph is shown in Fig. 2(a). To ease the notation, we denote with $\mathbf{e}_u \in \mathbb{R}^d$ the embedding of a generic user $u \in \mathcal{U}$ and use $\mathbf{e}_i \in \mathbb{R}^d$ to represent the embedding of an item $i \in \mathcal{I}$, where d is the dimension of embedding vector. Then, the recommendation task consists of predicting the users' preferences towards new items given the user-item interaction graph $\mathcal{G}$. In this paper, we mainly focus on MF and GNN-based CF methods.

3.2 An Overview of the Proposed Method

The architecture of the proposed method is shown in Fig. 2. First, we construct an undirected bipartite user-item graph based on the historical interactions of the users. Then, we use MFs or GNNs to encode the users' behaviors and items, and the GNN encoder is shown in Fig. 2(b). Finally, to boost the performance of various MF and GNN-based CF models, we propose a debiased contrastive loss to learn the embeddings of users and items, as shown in Fig. 2(c). Specifically, given a user-item pair, we treat the instances in the same batch as negatives and propose bias correction probability τ^+ to alleviate the sample bias to maintain state-of-the-art performance.

3.3 Debiased Contrastive Loss

To solve the problems of sample bias, we propose a debiased contrastive loss for CF models, as illustrated in Fig. 2(c). Given an arbitrary user u (an item i), we can obtain its embedding $\mathbf{e}_u$ (or $\mathbf{e}_i$) based on MF or GNN models. The recommender system is supposed to retrieve the top k items relevant to the user u by retrieving the top k candidate $\{\mathbf{e}_i\}_{i \in I}$ similar to $\mathbf{e}_u$. Most implementations use the inner product or the cosine similarity $\phi(u, i) =< \mathbf{e}_u, \mathbf{e}_i >$ as the similarity score. Considering the efficiency of sampling, the SSM loss emerges as a special softmax loss, which samples a subset of unobserved items as negatives, and further leading to

$$\mathcal{L}_{SSM} = \frac{1}{\mid \mathcal{D} \mid} \sum_{(u,i) \in \mathcal{D}} -\log \frac{e^{\phi(u,i)}}{e^{\phi(u,i)} + \sum_{l=1}^{L} e^{\phi(u,i_l)}}, \tag{1}$$

where $\mathcal{D} = \{(u, i) : u \in \mathcal{U}, i \in \mathcal{I} \quad \text{and} \quad g_{ui} = 1\}$ denotes the training set, $\{i_l\}_{l=1}^{L}$ are negative items that the user u has not interacted with, and L denotes the number of negative items for u.

Among the negative instances $\{i_l\}_{l=1}^{L}$, false negative items may fall into the user's area of interest. To address the sample bias, we develop a debiased contrastive loss. We assume a negative item in $\{i_l\}_{l=1}^{L}$ is actually a false negative with probability τ^+. Therefore, we can re-write the negative score $e^{\phi(u,i_l')} = \tau^+ e^{\phi(u,i)} + (1-\tau^+)e^{\phi(u,i')}$, where i' is the true negative item for user u. Here, we equally treat each negative item and employ the average score of all L negatives to approximate $e^{\phi(u,i_l')}$, namely, $e^{\phi(u,i_l')} = \frac{1}{L}\sum_{l=1}^{L} e^{\phi(u,i_l)}$. In this way, for each item i_l in $\{i_l\}_{l=1}^{L}$, we can obtain the corrected negative score for a user-item pair (u,i) as follows

$$g\left(u,i,\{i_l\}_{l=1}^{L}\right) = \frac{1}{1-\tau^+}\left(\frac{1}{L}\sum_{l=1}^{L} e^{\phi(u,i_l)} - \tau^+ e^{\phi(u,i)}\right), \tag{2}$$

where the second term in brackets can be regarded as the positive score for user u. We basically reduce the energy of negative samples by the false negative rate τ^+ times the energy of the positive sample. Note that we further scale the negative score by the inverse of $1-\tau^+$ (i.e., the probability of sampling a true negative example). Therefore, the debiased contrastive loss is given by

$$\mathcal{L}_{DCL} = \frac{1}{\mid \mathcal{D} \mid}\sum_{(u,i)\in\mathcal{D}} -\log\frac{e^{\phi(u,i)}}{e^{\phi(u,i)} + Lg\left(u,i,\{i_l\}_{l=1}^{L}\right)}. \tag{3}$$

Following the sampling strategy in [9], we consider the other (2B-2) instances in the current minibatch as negatives. Here, we treat users and items in the current minibatch as negative instances for the following reason: GNN models treat users and items equally in the sense that both of them are processed by fusing embeddings from their high-order neighbors. Therefore, users and items can be selected equally in the negative sampling process. Thus, we can modify Eq. (2) so that

$$g\left(u_j,i_j,\{u_k\}_{k=1}^{\mathcal{B}-\{j\}},\{i_k\}_{k=1}^{\mathcal{B}-\{j\}}\right) = \frac{1}{1-\tau^+}\left[\frac{\sum_{k=1}^{B-1}\left(e^{\phi(u_j,u_k)} + e^{\phi(u_j,i_k)}\right)}{2B-2} - \tau^+ e^{\phi(u_j,i_j)}\right], \tag{4}$$

where j is the index of the user or item in a minibatch. $\{u_k\}_{k=1}^{\mathcal{B}-\{j\}}$ and $\{i_k\}_{k=1}^{\mathcal{B}-\{j\}}$ denote the other $B-1$ instances in a minibatch for a user and an item.

In this work, we use the cosine similarity with temperature $t>0$ as the similarity function, i.e., $\phi(u,i) = \frac{f(u,i)}{t}$, where $f(u,i)$ denotes the cosine similarity. In addition, the value of $f(u,i)$ change in range $[-1,1]$, so Eq. (4) attains its theoretical minimum at $e^{-1/t}$ [17]. Therefore, in implementation, we constrain Eq. (4) to be greater than $e^{-1/t}$ to prevent calculating the logarithm of a negative number. Finally, the debiased contrastive loss for recommendation algorithm in this work is defined as

$$\mathcal{L}_{DCL} = \frac{1}{\mid \mathcal{D} \mid}\sum_{(u_j,i_j)\in\mathcal{D}} -\log\frac{e^{\phi(u_j,i_j)}}{e^{\phi(u_j,i_j)} + (2B-2)g}. \tag{5}$$

3.4 Theoretical Analysis

Proposition 1. *DCL can automatically mine the hard negative instance, where the hardness level for each negative instance adaptively depends on the predicted score and can also be controlled by the temperature t.*

Proof. To analyze the ability of mining hard negative instances for DCL, we resort the gradient with respect to negative instances. First, we re-write the $\mathcal{L}_{DCL}$:

$$\begin{aligned}\mathcal{L}_{DCL} &= \frac{1}{\mid \mathcal{D} \mid} \sum_{(u_j,i_j)\in\mathcal{D}} -\log \frac{e^{\phi(u_j,i_j)}}{e^{\phi(u_j,i_j)} + (2B-2)g} \\ &= \frac{1}{\mid \mathcal{D} \mid} \sum_{(u_j,i_j)\in\mathcal{D}} \log\left[1 + \frac{(2B-2)g}{e^{\phi(u_j,i_j)}}\right].\end{aligned} \tag{6}$$

The gradients *w.r.t.* $f(u_j, i_k)$ and $f(u_j, u_k)$ are respectively computed as

$$\frac{\partial \mathcal{L}_{DCL}}{\partial f(u_j,i_k)} = \frac{1}{\mid \mathcal{D} \mid} \sum_{(u_j,i_j)\in\mathcal{D}} \frac{1}{t} \frac{e^{f(u_j,i_k)}}{(1-\tau^+)\left[e^{\phi(u_j,i_j)} + (2B-2)g\right]} \tag{7}$$

$$\frac{\partial \mathcal{L}_{DCL}}{\partial f(u_j,u_k)} = \frac{1}{\mid \mathcal{D} \mid} \sum_{(u_j,i_j)\in\mathcal{D}} \frac{1}{t} \frac{e^{f(u_j,u_k)}}{(1-\tau^+)\left[e^{\phi(u_j,i_j)} + (2B-2)g\right]} \tag{8}$$

Therefore, for each negative instance, the gradient is proportional to the predicted score $f(u_j, i_k)$ or $f(u_j, u_k)$. A harder negative instance with larger predicted score has larger gradient, which indicates that DCL aims to optimize harder negative instance. Moreover, the larger the predicted score, the larger the gradient, and the harder the negative instance. In addition, we can tune the temperature t to control the gradient and further control the hardness level of negative instances, which is consistent with [18].

3.5 Model Training

To effectively learn the parameters of our method, we train the debiased contrastive loss $\mathcal{L}_{DCL}$ with regularization in an end-to-end fashion as follows

$$\mathcal{L} = \mathcal{L}_{DCL} + \lambda\left(||\mathbf{e}_u||_2^2 + ||\mathbf{e}_i||_2^2\right), \tag{9}$$

where β is a coefficient that balances between the two losses. λ controls the L_2 regularization strength.

4 Experiments

4.1 Experimental Settings

Datasets. We conduct experiments on three real-world datasets: **Yelp2018** is adopted from the 2018 edition of the Yelp challenge. Local businesses like restaurants and bars are considered as items. **Amazon-Book** consists of book

review data crawled from Amazon.com. **Steam** is a video game dataset collected from Steam, where items correspond to video games. We apply the preprocessing to Yelp2018 and Amazon-Book detailed in [1]. Concerning the Steam dataset, we follow the common practice detailed in [19]. The statistics of the processed datasets are summarized in Table 1. These datasets vary significantly in their domains, size, and sparsity. Following [1], we randomly select 80% of historical interactions of each user and assign them to the training set. The remainder comprises the test set.

Table 1. Statistics of the datasets.

Dataset	Yelp2018	Amazon-Book	Steam
#Users	31668	52643	281428
#Items	38048	91599	13044
#Interactions	1561406	2984108	3488885
Density	0.00130	0.00062	0.00095

Evaluation Metrics. To evaluate the quality of the recommendation algorithms, we adopt two ranking-based metrics: (1) recall@K measures the average number of items that the users interact with that are ranked among the top-K candidates. (2) ndcg@K is a precision-based metric that accounts for the predicted position of the ground truth instance. Thereby, all items, except for the ones with which the user has interacted, are considered for the ranking. Following [1,4], we set $K = 20$ and report the average metrics for all users in the test set. Concerning both metrics, larger values indicate better performances.

Baselines. We first verify whether our proposed DCL can augment existing MF-based or GNN-based recommendation methods: **MF** [6], **DMF** [12], **GCN** [15], **GC-MC** [14], **PinSage** [2], **NGCF** [1], **LR-GCCF** [3], **LightGCN** [4], and **LightGCN-single** [4]. The detail of all baselines can be found in the original publications. For a fair comparison, all baselines optimize the BPR loss. For each baseline, we retain the base model to encode the user behaviors and items and change the loss function to DCL. We append ++ to the name of each method to indicate that the results are obtained with our proposed DCL (e.g., NGCF++). Then, we choose LightGCN-single as the backbone and compare DCL with several popular loss functions for recommendation: **BCE** [5], **BPR** [6], **SSM** [9].

Parameter Settings. We optimize all models using Adam and employ Xavier initializations. The embedding size is fixed to 128 and the batch size to 2048 for all methods. Grid search is applied to choose the learning rate and the L_2 regularization coefficient λ over the ranges $\{10^{-4}, 10^{-3}, 10^{-2}\}$ and $\{10^{-6}, 10^{-5}, \cdots, 10^{-2}\}$. We set the number of layers $K = 1$ for GC-MC, and

$K = 2$ for other GNN-based methods. We use the cosine similarity with temperature as the similarity function, and set $t = 0.07$ for the Amazon-Book and $t = 0.1$ for both Yelp2018 and Steam. We use normalizations to stabilize debiased contrastive learning. The bias correction probability τ^+ is tuned on the sets $\{0, 0.1, 0.01, 0.001, 0.0001, 0.00001\}$. All other hyperparameters are set according to the suggestions from the settings specified in the original publications. All models are trained on a single NVIDIA GeForce GTX TITAN X GPU.

4.2 Overall Performance Comparison

Combine Our Method into Several Existing Recommendation Baselines. Table 2 summarizes the best results of all considered methods with (indicated by ++) and without our proposed DCL on the three benchmark datasets. The percentages in right corners indicate the relative improvements of our methods compared to each baseline. Bold scores indicate the best performance in each column; Bold percentages in right corners indicate the most significant relative improvements.

Table 2. A comparison of the overall performance among all considered baseline methods with (indicated by ++) and without our proposed DCL.

Methods	Dataset					
	Yelp2018		Amazon-Book		Steam	
	recall@20	ndcg@20	recall@20	ndcg@20	recall@20	ndcg@20
MF	0.0445	0.0361	0.0306	0.0231	0.0632	0.0303
MF++	$0.0502_{+12.81\%}$	$0.0403_{+11.63\%}$	$0.0489_{\mathbf{+59.80\%}}$	$0.0386_{\mathbf{+67.10\%}}$	$0.0759_{+20.09\%}$	$0.0390_{+28.71\%}$
DMF	0.0419	0.0347	0.0268	0.0215	0.0877	0.0441
DMF++	$0.0615_{\mathbf{+46.78\%}}$	$0.0507_{\mathbf{+46.11\%}}$	$0.0408_{+52.24\%}$	$0.0333_{+54.88\%}$	$0.1063_{+21.21\%}$	$0.0547_{+24.04\%}$
GCN	0.0457	0.0371	0.0312	0.0242	0.0866	0.0435
GCN++	$0.0649_{+42.01\%}$	$0.0533_{+43.67\%}$	$0.0486_{+55.77\%}$	$0.0381_{+57.44\%}$	$0.1089_{\mathbf{+25.75\%}}$	$0.0568_{\mathbf{+30.57\%}}$
GC-MC	0.0498	0.0411	0.0346	0.0258	0.1023	0.0518
GC-MC++	$0.0622_{+24.90\%}$	$0.0504_{+22.63\%}$	$0.0457_{+32.08\%}$	$0.0360_{+39.53\%}$	$0.1161_{+13.49\%}$	$0.0605_{+16.80\%}$
PinSage	0.0458	0.0369	0.0403	0.0309	0.0849	0.0455
PinSage++	$0.0479_{+4.59\%}$	$0.0392_{+6.23\%}$	$0.0469_{+16.38\%}$	$0.0369_{+19.42\%}$	$0.1044_{+22.97\%}$	$0.0540_{+18.68\%}$
NGCF	0.0584	0.0489	0.0365	0.0271	0.1129	0.0597
NGCF++	$0.0646_{+10.62\%}$	$0.0532_{+8.79\%}$	$0.0452_{+23.84\%}$	$0.0348_{+28.41\%}$	$0.1149_{+1.77\%}$	$0.0597_{+1.51\%}$
LR-GCCF	0.0602	0.0493	0.0373	0.0283	0.1035	0.0534
LR-GCCF++	$\mathbf{0.0711}_{+18.11\%}$	$0.0558_{+13.18\%}$	$0.0460_{+23.32\%}$	$0.0340_{+20.14\%}$	$0.1086_{+4.93\%}$	$0.0565_{+5.81\%}$
LightGCN	0.0630	0.0519	0.0453	0.0346	0.1159	0.0611
LightGCN++	$0.0677_{+7.46\%}$	$\mathbf{0.0562}_{+8.29\%}$	$\mathbf{0.0537}_{+18.54\%}$	$\mathbf{0.0424}_{+22.54\%}$	$0.1185_{+2.24\%}$	$0.0622_{+1.80\%}$
LightGCN-single	0.0633	0.0506	0.0466	0.0358	0.1246	0.0650
LightGCN-single++	$0.0680_{+7.42\%}$	$0.0556_{+9.88\%}$	$0.0532_{+14.16\%}$	$0.0418_{+16.76\%}$	$\mathbf{0.1369}_{+9.87\%}$	$\mathbf{0.0704}_{+8.31\%}$

On the one hand, DCL exhibits the largest performance gains when combined with the most basic model, such as MF, DMF, and GCN. Especially on Yelp2018, DMF++ exhibits a performance increase of 46.78% recall@20 and 46.11% ndcg@20. On Amazon-Book, MF++ outperforms MF with 59.80% recall@20 and 67.10% ndcg@20, respectively. On Steam, compared with GCN,

GCN++ obtains relative performance gains of 25.75% and 30.57% concerning recall@20 and ndcg@20, respectively. On the other hand, DCL shows the best performance in some state-of-the-art models. For example, on Yelp2018, LR-GCCF++ reaches 0.0711 concerning recall@20, and LightGCN++ reaches 0.0562 concerning ndcg@20. On Amazon-Book, LightGCN++ shows the best performance, and on Steam, LightGCN-single++ consistently outperforms all the other baselines.

GCN and GC-MC indicate their superior capabilities to incorporate information from the user-item graph compared to MF and DMF. By correcting sampling bias, GCN++ and GC-MC++ consistently outperform GCN and GC-MC, respectively. Compared to GCN and GC-MC, PinSage and NGCF further consider higher-order neighborhoods. Moreover, PinSage++ and NGCF++ consistently outperform PinSage and NGCF, but the relative improvements are slightly worse than the ones of GCN++ and GC-MC++.

These results show that debiased contrastive learning positively affects recommendation performances. In the following subsections, we focus on LightGCN-single++ and analyze its performance in greater detail.

Compare with Different Loss Functions. Table 3 summarizes the best results of all considered loss functions on three benchmark datasets *w.r.t.* LightGCN-single. Bold scores are the best in each column, while underlined scores are the second best. The last row shows the improvements in our DCL relative to the best baseline.

Table 3. A comparison of the overall performance among different loss functions.

Methods	Dataset					
	Yelp2018		Amazon-Book		Steam	
	recall@20	ndcg@20	recall@20	ndcg@20	recall@20	ndcg@20
BCE	0.0591	0.0482	0.0437	0.0335	0.1134	0.0581
BPR	0.0633	0.0506	0.0466	0.0358	0.1246	0.0650
SSM	<u>0.0673</u>	<u>0.0542</u>	<u>0.0511</u>	<u>0.0405</u>	<u>0.1328</u>	<u>0.0689</u>
DCL	**0.0680**	**0.0556**	**0.0532**	**0.0418**	**0.1369**	**0.0704**
Improv	1.04%	2.58%	4.11%	3.21%	3.09%	2.18%

Table 3 shows that the model with BCE loss achieves the worst performance on all three datasets, indicating that it is unsuitable to take the top-k recommendation as a classification problem. In particular, it may introduce noise when fitting the exact value of ratings. BPR loss outperforms BCE loss, verifying the effectiveness of capturing relative relations among items. While SSM loss further outperforms BCE and BPR loss, indicating that it conducts in-batch sampling and efficiently uses negative instances. However, a few negative instances may constitute false negatives. Thus, we apply the bias correction probability τ^+ to

control sample bias and define it as DCL. We find that DCL loss further improves the performance of the recommendation compared to SSM loss. Specifically, on the Amazon-Book dataset, DCL achieves 4.11% improvement for recall@20 and 3.21% improvement for ndcg@20. In summary, adequate negative instances and an effective sample bias control are essential to improve the performance of recommendation algorithms.

4.3 Effectiveness Analysis

Figure 3 shows the test performance regarding recall@20 for each epoch of LightGCN-single and LightGCN-single++. Due to the space limitation, we omit the performance concerning ndcg@20, which has a similar trend as recall@20. Since the convergence speed of the LightGCN-single is relatively slow, we include a small sub-figure in the lower right corner of each figure.

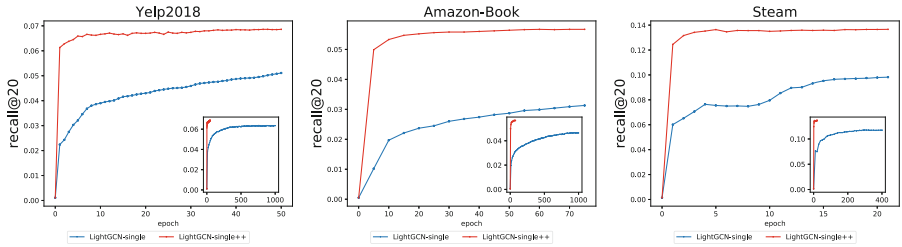

Fig. 3. The test performance after each training epoch of LightGCN-single and LightGCN-single++.

We observe that LightGCN-single++ exhibits faster convergence than LightGCN-single on all three datasets. Specifically, to achieve the best performance, LightGCN-single requires thousands of epochs on Yelp2018 and Amazon-Book and about 400 epochs on Steam. While LightGCN-single++ only requires 50–70 epochs to achieve the best performance on Yelp2018 and Amazon-Book dataset. Moreover, it only requires 18 epochs to converge on the Steam dataset. It is reasonable that even though we sample thousands of negatives for each user, this does not increase the memory overhead due to the reuse of the instances in the current minibatch. These results indicate that a debiased sampling strategy not only improves training efficiency but also improves the recommendation performance. This finding provides a new optimization perspective for the design of future recommendation algorithms.

4.4 Parameter Sensitivity Analysis

In this subsection, we examine the robustness with respect to the bias correction probability τ^+. We analyze this hyperparameter by fixing the remaining hyperparameters at their optimal value. Figure 4 shows the recall@20 and ndcg@20

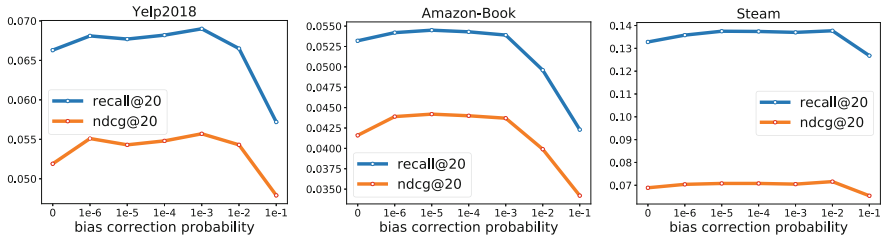

Fig. 4. Performance of LightGCN-single++ obtained with different bias correction probabilities τ^+.

for LightGCN-single++ with the bias correction probability τ^+ varying from 0 to 0.1 while keeping other optimal hyper-parameters unchanged.

The most obvious observation from Fig. 4 is that the recall@20 and ndcg@20 have the same trend: they both increase first and then decrease as τ^+ increases. On Yelp2018 and Amazon-Book, recall@20 and ndcg@20 increase steadily to specific high values of τ^+. If we continue to increase τ^+ further, the performance drops suddenly. Moreover, on the Steam dataset, when we vary τ^+ from 0.01 to 0.1, the performance starts to drop. It shows we can correct the sample bias by carefully selecting τ^+ in an appropriate interval. Empirically, we find that when the hyperparameter τ^+ exceeds a specific range, the performance will deteriorate since more samples are considered false negative.

For a more fine-grained analysis, when setting $\tau^+ = 0$, we do not correct the sample bias and treat the other $(2B - 2)$ users and items in the current minibatch as true negative instances. We find that the performance is relatively weak across all three datasets. It indicates that among these negative samples, false negative instances need to be taken into account. When we set τ^+ to a small value to correct the sample bias, recall@20 and ndcg@20 improve. We can obtain the best performance when setting τ^+ to $1e^{-3}$, $1e^{-5}$, and $1e^{-2}$ on Yelp2018, Amazon-Book, and the Steam datasets, respectively.

5 Conclusion

In this work, we presented a general debiased contrastive loss for collaborative filtering, aiming to alleviate the sample bias. Concretely, we treat the other users and items in the current minibatch as negative instances and employ a bias correction probability τ^+ to control the proportion of false negative samples. Experimental results on three real-world datasets showed that our method significantly augments several MF and GNN-based CF models and outperforms popular learning objectives. In addition, we also find that our method improves training efficiency significantly. In future works, we plan to explore personalized τ^+ for different users to adaptively alleviate sample bias and examine whether the proposed method can be extended to different recommender settings, such as sequential recommendation tasks.

References

1. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: The 42nd International ACM SIGIR Conference (2019)
2. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: KDD, pp. 974–983. ACM (2018)
3. Chen, L., Wu, L., Hong, R., Zhang, K., Wang, M.: Revisiting graph based collaborative filtering: a linear residual graph convolutional network approach. In: AAAI, pp. 27–34. AAAI Press (2020)
4. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: Lightgcn: simplifying and powering graph convolution network for recommendation. In: SIGIR, pp. 639–648. ACM (2020)
5. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.-S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, pp. 173–182 (2017)
6. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. CoRR, abs/1205.2618 (2012)
7. Barbano, C.A., Dufumier, B., Tartaglione, E., Grangetto, M., Gori, P.: Unbiased supervised contrastive learning. In: The Eleventh International Conference on Learning Representations (2023)
8. Zhou, C., Ma, J., Zhang, J., Zhou, J., Yang, H.: Contrastive learning for debiased candidate generation in large-scale recommender systems (2021)
9. Wu, J., et al.: On the effectiveness of sampled softmax loss for item recommendation. arXiv preprint arXiv:2201.02327 (2022)
10. Chuang, C.-Y., Robinson, J., Lin, Y.-C., Torralba, A., Jegelka, S.: Debiased contrastive learning. CoRR, abs/2007.00224 (2020)
11. Koren, Y., Bell, R.M., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
12. Xue, H.-J., Dai, X., Zhang, J., Huang, S., Chen, J.: Deep matrix factorization models for recommender systems. In: IJCAI, Melbourne, Australia, vol. 17, pp. 3203–3209 (2017)
13. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.-S.: Neural collaborative filtering. CoRR, abs/1708.05031 (2017)
14. van den Berg, R., Thomas, N.K., Welling, M.: Graph convolutional matrix completion. CoRR, abs/1706.02263 (2017)
15. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (Poster). OpenReview.net (2017)
16. Shan, Y., Hoens, T.R., Jiao, J., Wang, H., Yu, D., Mao, J.C.: Deep crossing: web-scale modeling without manually crafted combinatorial features. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 255–262 (2016)
17. Chuang, C.-Y., Robinson, J., Lin, Y.-C., Torralba, A., Jegelka, S.: Debiased contrastive learning. Adv. Neural. Inf. Process. Syst. **33**, 8765–8775 (2020)
18. Wang, F., Liu, H.: Understanding the behaviour of contrastive loss. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2495–2504 (2021)
19. Sun, F., et al.: BERT4Rec: sequential recommendation with bidirectional encoder representations from transformer. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 1441–1450 (2019)