Knowledge-Based Systems 261 (2023) 110180

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Ranking-based contrastive loss for recommendation systems

Hao Tang [a,b], Guoshuai Zhao [c,*], Yujiao He [c], Yuxia Wu [a], Xueming Qian [a,d]

[a] School of Information and Communication Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[b] China Unicom Shaanxi Branch, Xi'an 710075, China
[c] School of Software Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[d] Ministry of Education's Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an 710049, China

ARTICLE INFO

Article history:
Received 31 October 2021
Received in revised form 7 October 2022
Accepted 4 December 2022
Available online 8 December 2022

Keywords:
Contrastive loss
Recommendation system
Hard samples
Negative samples
Graph convolution network

ABSTRACT

The recommendation system is fundamental technology of the internet industry intended to solve the information overload problem in the big data era. Top-k recommendation is an important task in this field. It generally functions through the comparison of positive pairs and negative pairs based on Bayesian personalized ranking (BPR) loss. We find that the contrastive loss (CL) function used in contrastive learning is well-suited for top-k recommendation. However, there are two problems in the existing loss functions. First, all samples are treated the same, and hard samples are not considered. Second, all nonpositive samples are considered negative samples, which ignores the fact that they are unlabelled data containing items that users may like. Moreover, in our experiments, we find that when items are sorted by their similarities to the user, many negative items (or samples) appear before the positive items. We regard these negative items as hard samples and those at the top as potentially positive samples due to their high level of similarities with users. Therefore, we propose a ranking-based contrastive loss (RCL) function to exploit both hard samples and potentially positive samples. Experimental results demonstrate the effectiveness, broad applicability, and high training efficiency of the proposed RCL function. The code and data are available at https://github.com/haotangxjtu/RCL.

1. Introduction

Recently, recommendation systems have been widely used in many areas, such as news, advertising, e-commerce, social networking, and entertainment. They have become a fundamental technology and an important research area in the big data era. Recommendation algorithms have evolved from traditional methods to deep learning-based methods [1–3], as well as the popular graph convolution network (GCN) [4–6]. Top-k recommendation is one of the basic tasks in recommendation systems [7,8]. It generates a recommendation list by sorting the similarities between users and items. The most popular loss function used is Bayesian personalized ranking (BPR) [9] loss, which aims to maximize the distance between a positive pair and a negative pair. Contrastive loss (CL) is widely used in contrastive learning [10–12], and we find that CL is naturally suitable for recommendation systems due to the same contrastive process. Cross-entropy loss treats top-k recommendation as a classification problem and is used in some cases [1].

However, these loss functions have not considered the following two problems in the recommendation system: (1) All samples have the same weights, and there is no distinction between hard samples and simple samples. However, hard sample mining is of great importance for improving recommendation systems. (2) All nonpositive samples are regarded as negative samples, but those samples with great similarities to users may be potentially positive samples. Mining these potentially positive samples can solve the problem of insufficient positive samples and data sparsity, which will help improve performance.

Both of these problems are related to the processing of negative samples in recommendation systems. How negative samples are viewed and handled in recommendation systems is an important issue. Generally, items that users interact with are viewed as positive samples, which are just the "observed" positive samples. The items that are not interacted with are simply treated as negative samples, while most of them are "unlabelled data", which also contain items that the user probably likes. The main goal of a recommendation system is to retrieve and sort the items a user interacts with from those that are not interacted with (negative sample) and recommend the items that the user probably likes. Therefore, reasonable processing of negative samples can effectively improve the performance of the recommendation system, as shown in existing works. Some negative sampling methods [9,13,14] are combined with loss functions which are

* Corresponding author.
E-mail addresses: th1002@stu.xjtu.edu.cn (H. Tang), guoshuai.zhao@xjtu.edu.cn (G. Zhao), wuyuxia@stu.xjtu.edu.cn (Y. He), qianxm@mail.xjtu.edu.cn (X. Qian).

https://doi.org/10.1016/j.knosys.2022.110180
0950-7051/

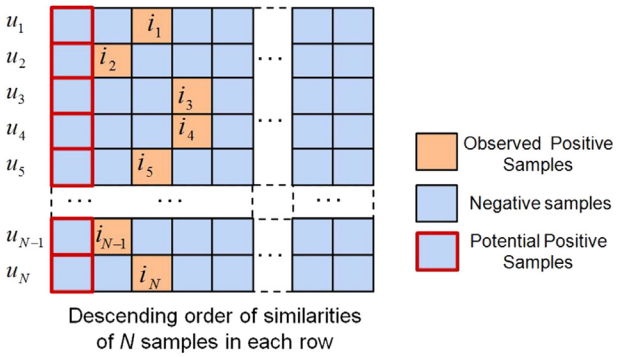


Fig. 1. After ranking by similarities, many negative items appear before the positive ones, which are viewed as hard samples, and those at the top are mined as potentially positive samples. N is the batch size.

characterized by simplicity, efficiency, effectiveness and wide application scenarios. These methods are more suitable for industrial applications at a large scale and becomes the key direction for our research.

Experiments show that CL is well suited for top-k recommendation. However, CL also faces the problems discussed above. Fortunately, we find an interesting phenomenon that helps to address these two challenges. After calculating the similarity of users and all items within a batch and sorting them in descending order in each row, many negative items appear before the observed positive items, as shown in Fig. 1. In the kth row, i_k is the positive item of u_k, but it is not the most similar item. Actually, in the three datasets we used, Yelp2018, Amazon-Book, and Pinterest, the average positions of the observed positive items $i_1, i_2, \ldots, i_N$ are approximately 45, 21, and 52, respectively, in the best baseline method, the light graph convolution network (LightGCN) [15], of 2048 items. This finding is a fundamental but rarely directly discussed issue in top-k recommendation. We are motivated by this and analyse the top-ranked negative items as follows:

- They are negative items but have a high degree of similarity with users, so they are probably hard samples. Those items after the positive samples are easy samples. The distinction between hard samples and simple samples is seldom discussed in top-k recommendation, but it is very useful for improving learning performance.
- The top-ranked items are mined as potentially positive samples. The recommendation system aims to select the most preferred items for the users. The top-ranked items are very similar to the users and they are probably what the user are interested in. The higher the ranking is, the higher the probability that the item will be recommended. We mine the top-ranked items as positive samples in this paper which are called potentially positive samples.

We successfully mine hard samples and potentially positive samples through the interesting phenomena, which helps to solve the problems mentioned above. We propose a ranking-based contrastive loss (RCL) function to make better use of them. On the one hand, we design a similarity-based weighting method where the weights of the hard samples are larger than those of simple samples because of their greater similarity. Therefore, the hard samples are given relatively more attention and can be better optimized. On the other hand, we modify the CL function to use the potentially positive samples directly, which is equivalent to using more positive items. This helps to alleviate the problem of insufficient positive items and the sparsity problem in the recommendation system.

Modifying the loss function is a simple, convenient, and adaptable way to implement the above ideas. There is almost no increase in computation and storage consumption relative to the whole model. Notably, our RCL is a model-agnostic approach and can be applied to any model based on user and item embeddings.

We summarize the contributions of this work as follows:

- We propose a novel RCL function by focusing on hard samples and exploring potentially positive samples. We make CL adaptable to recommendation systems, and it is also a new loss function related to negative samples.
- To distinguish the learning difficulty of samples and pay more attention to hard samples, we devise a weighted contrastive loss (WCL) that forces the model to adaptively focus on different samples.
- We mine potentially positive samples and combine them with CL functions to form potentially positive sample-based CL (PCL) functions, which helps to alleviate the problems of sparsity and lack of positive samples.
- Experiments on real-world datasets demonstrate the superior performance of RCL over the state-of-the-art methods. In addition, we further show the applicability and high training efficiency of our method.

The rest of this paper is organized as follows. In Section 2, a brief review of related works is presented. To verify the effectiveness of RCL, we design pLightGCN_RCL, which combines RCL with the best baseline as our method in Section 3. The experiments and discussions are provided in Section 4. Conclusions are drawn in Section 5.

2. Related work

In this section, we briefly review related studies, which include top-k recommendation, contrastive loss and methods for handling negative samples.

2.1. Top-k recommendation

The top-k recommendation task is to retrieve the top k items that are probably liked by the user [7,16]. The history of user–item interactions can be used to mine the characteristics of users and items. Collaborative filtering (CF) [17–20] is a basic algorithm for mining user preferences and is commonly used for recommendations. In recent years, many deep learning techniques and methods, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), attention mechanisms, and graph convolution networks (GCNs), have been applied to the recommendation field. For example, attention mechanisms combined with autoencoders or graphs are widely used in the recommendation field [21,22]. As a ranking problem, the pairwise loss would be advantageous [13]. BPR [9] loss learns the embeddings of users and items by maximizing the distance between positive pairs and negative pairs, which is the loss function widely used.

The GCN-based CF model has also been widely studied and has recently achieved good performance [15,23–26]. For example, neural graph collaborative filtering (NGCF) [23] exploits the structure of the user–item graph and its high-order connectivity by propagating embeddings on it. Linear residual graph convolutional collaborative filtering (LR-GCCF) [24] is a lightweight linear model proposed to alleviate the oversmoothing problem while dealing with sparse data in graph convolution. LightGCN [15] retains only the most important part of the GCN, namely, neighbourhood aggregation, to be more concise and suitable for recommendations and for achieving better performance. Disentangled graph collaborative filtering (DGCF) [25] focuses on the user's intention to adopt different items by modelling a distribution

over intents for each user–item interaction. Neural graph personalized ranking (NGPR) [26] is proposed to directly make use of the user–item interaction information in embedding learning by incorporating the user–item interaction graph. LR-GCCF and LightGCN can be seen as the development of NGCF, and we select LightGCN as our baseline.

2.2. Contrastive loss

CL is one of the most useful functions to mine the relationship between samples, and it is designed to narrow the distance between positive samples and enlarge the distance between negative samples [10,27,28]. They are widely used in contrastive learning and many tasks [29,30].

Triplet loss [31,32] uses exactly one positive sample and one negative sample at each anchor and attempts to moderate overfitting by the max-margin operation. There are several derivative forms of CL based on multiple negative samples, such as N-pair loss [33], noise-contrastive estimation (InfoNCE) loss [34,35], and normalized temperature-scaled cross-entropy loss (NT-Xent) [10,11]. The N-pair loss [33] considers multiple negative samples for each anchor to solve the problem of the slow convergence of triplet loss. The InfoNCE loss is commonly applied to measure the distance or similarity by its mutual information in many works of representation learning [34,35]. NT-Xent [10,11] regards all the $2N-2$ samples except the target and its positive sample as negative samples to fully tap the potential relationships between samples. The slight difference is that the denominator of NT-Xent contains only negative samples, while that of the N-pair loss and InfoNCE loss is the sum of positive and negative samples. They share the same idea with a minor difference in form.

Some improvements on CL have been proposed, such as soft contrastive learning [36] and debiased contrastive learning [28]. However, they are proposed for the contrastive learning framework and are generally applied in the computer vision (CV) domain. CL needs to adapt to the recommendation system, which is based on graph-based contrastive learning and has problems in this domain [37,38]. Self-supervised graph learning (SGL) [39] is proposed for recommendations, but it focuses on the application of the contrastive learning framework rather than the CL function. Moreover, compared with the complex contrastive learning framework, which requires pretraining and fine-tuning or multitask learning [39] and requires data augmentation with an increased computational burden, our improved CL function is simple, more efficient, and has wide adaptability.

2.3. Methods for handling negative samples

Our proposed loss function is related to appropriately viewing and handling negative samples. Negative sampling strategies are often used in combination with loss functions. BPR [9] is the most basic and widely used loss function, which assumes that the samples obey a uniform distribution and generally samples one or several negative samples. Neural network-based collaborative filtering (NNCF) [13] proposes a general hybrid recommendation framework with a "graph-based" loss function. This function uses nonpositive samples within the same batch as negative samples, which greatly increases the number of negative samples used and achieves a significant performance improvement. The negative sampling method is also adopted by CL functions [10]. The efficient neural matrix factorization framework (ENMF) [14] proposes a whole-data-based method that employs a nonsampling strategy. The loss function for optimization is derived through mathematical reasoning.

Models combining negative sampling strategies are also often proposed. Generative adversarial networks (GANs), reinforcement learning, and more auxiliary information are adopted by models to mine hard samples to improve performance. To sample true negative instances of high quality, simplify and robustify negative sampling (SRNS) [40] designs a two-step sampling scheme that constantly alternates between score-based memory updates and variance-based sampling. Auxiliary information helps mine hard samples. In real-world scenarios, platforms can easily collect whether the recommended (i.e., exposed) item has been interacted with by a user. Reinforced negative sampling (RNS) [41] integrates the exposure data into the negative sampler by using rich information about the negative preferences of users. Similar to RNS, IRGAN [42] and AdvIR [43] are also GAN-based structures that are much more complicated for training and application. As a result, such methods place higher demands on model design, model training, and data resources. For example, with more hyperparameters to fine-tune, a pretraining model may be needed, which is more likely to suffer mode collapse. The negative sampling-based loss functions are more intuitive and simpler; they are also model- and data-independent and applicable to a wide range of applications.

Compared with the above top-k recommendation methods, we propose a loss function for recommendations, and, in theory, RCL can be applied to most of them. Compared with the existing CL functions, we propose an improved CL function to make it more adaptable to the recommendation system by combining the problems in the recommendation system. Compared with the existing negative sampling methods, we propose a simple and effective method based on a loss function that gives more attention to hard samples by weighting and mines potentially positive items. Our approach is also a positive-unlabelled (PU) learning method that learns only from positive samples and unlabelled data. Compared to previous approaches in the field of PU learning [44], we propose a new approach in combination with the specific problem of recommendation systems: identifying hard samples and exploiting potentially positive samples. In particular, our approach solves the problem by combining the CL function. RCL focuses on different problems in the recommendation field compared to our previous work MSCL [45] which solves the problem of different importance of samples and underutilization of positive samples.

3. Methodology

The RCL function is applicable to all embedding-based recommendation methods in theory. To verify and discuss its effectiveness, we implement RCL with the state-of-the-art GCN-based model LightGCN [15]. The architecture of the proposed framework is shown in Fig. 2. We first construct a user–item graph based on the historical interaction of users and items. Then, the LightGCN framework is applied to obtain the embeddings of users and items. Similarities between users and items from the same batch are calculated by cosine similarity based on embeddings. Then, they are sorted in descending order. Thus, the items preceding the positive item are seen as hard samples, while the items following the positive item are simple samples. Some top-ranked items (in light brown) of the row are treated as potentially positive samples. Then, the RCL function is proposed based on them. We first briefly describe the basic method and then focus on the RCL function in detail.

3.1. Basic method

LightGCN [15], the state-of-the-art method for top-k recommendation, is selected as our basic method. We briefly introduce LightGCN here. A user–item graph is built based on the user's

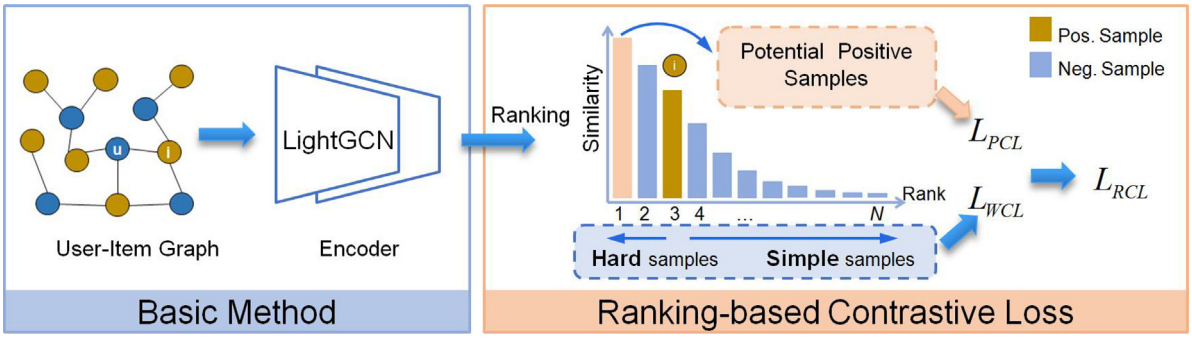


Fig. 2. An illustration of our method. The figure displays the processing of each user–item pair within the batch.

interaction history. After that, embeddings of users and items are obtained by the following GCN:

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

$$\boldsymbol{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \boldsymbol{e}_u^{(k)} \tag{2}$$

where u, i denote the user and item, respectively, and $e_u^{(k)}$, $e_i^{(k)}$ are embeddings of u, i of the kth layer. Specifically, $k = 0$ represents the initialized latent vector. $\mathcal{N}_u$ and $\mathcal{N}_i$ represent the set of the neighbours of targets u and i, respectively. The final embeddings of users and items are calculated as follows:

$$\boldsymbol{e}_u = \sum_{k=0}^{K} \alpha_k \boldsymbol{e}_u^{(k)} \tag{3}$$

$$\boldsymbol{e}_i = \sum_{k=0}^{K} \alpha_k \boldsymbol{e}_i^{(k)} \tag{4}$$

where α_k are weight parameters for each layer, and K is the maximum number of layers. In LightGCN, α_k parameters are set to $1/(k+1)$ because the authors find that learning α_k does not lead to improvements. Under the CL function, we succeed in improving the performance by treating α_k as learnable parameters and they are automatically optimized and learned during the training process similar to the other parameters, such as the embeddings. This modified method is named pLightGCN.

Similar to most top-k recommendation models, LightGCN is trained using the traditional BPR loss as follows

$$L_{BPR} = \sum_{(u,i,j) \in D} -\log \sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right) \tag{5}$$

where i, j denote positive and negative items, U, I is the set of users and items and $D = \{(u, i, j), u \in U, i, j \in I\}$; $\sigma(\cdot)$ is the sigmoid function; and $\hat{y}_{ui}$ is the inner product of the user and item as same as the model prediction in Formula (13). In this paper BPR is replaced by RCL.

3.2. Ranking-based Contrastive Loss (RCL)

According to recent works [10,11], we use the NT-Xent loss as the original contrastive loss function and adapt it to top-k recommendation:

$$sim(u, i) = \boldsymbol{e}_u^{\top} \boldsymbol{e}_i / \|\boldsymbol{e}_u\| \|\boldsymbol{e}_i\| \tag{6}$$

$$L_{CL} = -\frac{1}{N} \sum_{(u,i) \in D} \log \frac{\exp\left(sim(u, i^+)/\tau\right)}{\sum_{i \in I^-} \exp\left(sim(u, i)/\tau\right)} \tag{7}$$

where i^+ is the positive sample of target user u, I^- is the set of negative samples, and N denotes the batch size. $sim(u, i)$ is the cosine similarity of the (u, i) pair based on their embeddings. We follow the sampling strategy used in contrastive learning [10,11] in that the other $N-1$ items except the positive sample i^+ in the same batch are regarded as negative items for user u.

3.2.1. Highlighting the hard samples by Weighted CL (WCL)

Hard sample mining is important for the recommendation field. In contrast to previous works [27,39], which point out that the original CL function has the inherent ability to be mined from the perspective of the gradient, we improve the CL function by extrinsically and explicitly mining hard samples. All these methods contribute large and meaningful gradients for hard samples in optimization and guide node representation learning.

For negative samples, the greater the similarity is, the more difficult it is to learn; the lower the degree of similarity is, the easier it is to learn. Thus, the similarities are proportional to the errors/losses and are used as weights in our method. In this way, the weights of hard samples are increased, making the model focus more on the learning of hard samples during training. Since positive samples generally have a high degree of similarity, their learning can also be facilitated by similarity weighting. Therefore, we consistently weighted all samples according to their similarities. We first convert the cosine similarity values to positive values by using the *softplus* function. Other functions, such as softmax and sigmoid, are also effective, but we find that softplus is better experimentally. The formulas are as follows:

$$f(u, i) = (softplus(sim(u, i)))^{\gamma} \exp(sim(u, i)/\tau) \tag{8}$$

$$L_{WCL} = -\frac{1}{N} \sum_{(u,i) \in D} \log \frac{f\left(u, i^+\right)}{\sum_{i \in I^-} f(u, i)} \tag{9}$$

where $softplus(x) = log(1 + exp(x))$, and γ is a positive hyperparameter, which helps adjust the samples' weights while training.

The loss function L_{WCL} takes the similarity, $sim(u, i)$, as the weight to automatically focus on the learning of hard samples. Here, we give a concrete example to demonstrate the influence of weighting by similarity. Suppose there is a simple sample and a hard sample with similarity values of -0.9 and 0.9, respectively. We take $\gamma = 3$ and calculate the weights of the two samples by $(softplus(sim(u, i)))^{\gamma}$, which are 0.04 and 1.91 times those of the unweighted case when $\gamma = 0$. We observe that the weight of the simple sample decreases, while the weight of the hard sample increases, and the gap between the two becomes larger: the weight of the hard sample is approximately 47 times that of the simple sample. Thus, the hard sample proportion increases in loss, and hard samples receive more attention and are better learned in the learning process.

3.2.2. Mining Potentially Positive samples on CL (PCL)

We use the top-ranked samples as potentially positive samples for the following reasons. (1) Negative samples are unlabelled data that contain what users may like. Therefore, we mine the possible positive samples from negative samples. (2) The goal of recommendations is to retrieve and recommend what the user may like from unlabelled data, which meets the basic principle of recommendation. (3) There must exist (with different probabilities) samples that users may like, i.e., potentially positive samples. This is the basic assumption of the existence of the recommendation system. Otherwise, the recommendation is unreasonable. (4) Existing methods also support our approach by mining and exploiting potentially positive samples. For example, graph data augmentation (GAUG) [46] enables the augmentation of graph data by dynamically generating positive samples through a generator, thereby increasing the number of connections in the original graph.

Therefore, the top-ranked items are sampled as potentially positive items. Instead of actually making the sparse graph denser by establishing links in graphs as implemented GAUG [46], we use these positive samples by improving the loss function and define L_P as

$$L_P = -\frac{1}{N}\sum_{(u,i)\in D}\log\frac{\sum_{j\in T}\exp\left(sim\left(u,j\right)/\tau\right)}{\sum_{i\in I^-}\exp\left(sim\left(u,i\right)/\tau\right)} \tag{10}$$

where $sim\left(u,j\right)$ is the similarity and T is the set of top j items with the greatest similarities. We combine L_P with the original CL function as L_{PCL},

$$\begin{aligned} L_{PCL} =& L_{CL} + L_P \\ =& -\frac{1}{N}\sum_{(u,i)\in D}(\log\frac{\exp\left(sim(u,i^+)/\tau\right)}{\sum_{i\in I^-}\exp\left(sim(u,i)/\tau\right)} \\ &+\lambda\log\frac{\sum_{j\in T}\exp\left(sim\left(u,j\right)/\tau\right)}{\sum_{i\in I^-}\exp\left(sim\left(u,i\right)/\tau\right)}) \end{aligned} \tag{11}$$

where λ is a hyperparameter to balance the contribution of L_P.

3.2.3. Combining WCL and PCL as RCL

The previous section specifies how the two improvements are implemented, and next, we combine them as the ranking-based CL function:

$$\begin{aligned} L_{RCL} =& -\frac{1}{N}\sum_{(u,i)\in D}(\log\frac{f\left(u,i^+\right)}{\sum_{i\in I^-}f\left(u,i\right)} \\ &+\lambda\log\frac{\sum_{j\in T}f\left(u,j\right)}{\sum_{i\in I^-}f\left(u,i\right)}) \end{aligned} \tag{12}$$

where $f\left(u,i\right)$ is the weight function of WCL and the function after the plus sign using potentially positive samples in the set T is PCL.

In RCL, potentially positive samples are also part of the hard samples, which are a remining of the hard samples. That is to say the top few samples are used both as hard samples and as potentially positive samples. The reason why some top-ranked samples are difficult to learn is that they are probably unseen positive samples rather than negative ones. Moreover, better ranking quality can be obtained by weighted learning based on WCL, and only the top of them are reused as potentially positive samples. We find that WCL or PCL performs better on different datasets, meaning that one of the factors can play a dominant role. By balancing WCL and PCL with hyperparameters, RCL is able to take advantage of both and adapt to different datasets to achieve better results.

Table 1
Complexity analyses.

Component		Complexity
LightGCN	Adjacency matrix	$O(2\|E\|)$
	Graph convolution	$O\left(2\|E\|Lds\frac{\|E\|}{N}\right)$
Loss	BPR	$O(2\|E\|ds)$
	CL	$O(N\|E\|ds)$
	RCL	$O(N\|E\|ds)$

3.3. Model prediction

The model prediction is defined as the inner product of the user and item final embeddings:

$$\hat{y}_{ui} = \boldsymbol{e}_u^T\boldsymbol{e}_i \tag{13}$$

Following LightGCN, the L2 regularization of parameters is added in the training loss function, and the top-k items are recommended to users. Overall, we replace the BPR function with RCL and keep all other details the same. Our approach with the RCL function is named pLightGCN-RCL.

3.4. Time complexity analyses

We analyse the complexity of our method following SGL [39], which also uses LightGCN as the baseline, mainly focusing on the complexity of the loss function. Suppose the edge in the user–item interaction graph is $|E|$. Let L denote the number of GCN layers and d, N, s denote the embedding size, the batch size, and the number of training epochs, respectively. The complexity of CL is $O(Ndes)$, which is mainly the inner product. The complexity of RCL contains three parts: sorting, weighting and adding potentially positive samples. It is sorted once per epoch. The complexity of the sorting algorithm is generally $O(NlogN)$, so the overall complexity of sorting is $O(NlogN|E|/Ns) = O((logN)|E|s)$. By using the existing calculations of CL, both WCL and PCL add a very limited number of additions, multiplication, and divisions. Therefore, their complexity is both $O(N|E|s)$. The overall complexity of RCL is $O(Nd|E|s) + O(logN|E|s) + O(N|E|s) + O(N|E|s) = O((Nd+logN+2N)|E|s)$. Considering that generally $N = 2048$, $d = 64$, the latter two are much smaller than the first one, so the final complexity is simplified to $O(Nd|E|s)$. That is, it has the same time complexity as CL. This is consistent with the intuition that we add only simple operations such as weighting and summation to CL. All the components and complexities are shown in Table 1.

For comparison, the complexity of BPR is $O(2|E|ds)$. RCL is $O(N/2)$ times larger than the computational cost of BPR. However, there is no significant time difference between them in practice which is shown in the Training Efficiency at last of the next section.

4. Experiments

4.1. Experimental settings

4.1.1. Datasets

To evaluate the effectiveness of RCL, we conduct experiments on three benchmark datasets: Yelp2018 [15,23], Amazon-Book [15,23], and Pinterest [1,47]. These three datasets are collected for local businesses, books, and content-based image recommendations. For the first two datasets, we use the same datasets and splits of LightGCN [15] for comparison purposes. For Pinterest, we follow the same dataset preprocessing strategy of LightGCN. The statistics of the processed datasets are summarized in Table 2.

Table 2
Statistics of the datasets.

Dataset	Users	Items	Interactions	Density
Yelp2018	31,668	38,048	1,561,406	0.00130
Amazon-Book	52,643	91,599	2,984,108	0.00062
Pinterest	55,187	9916	1,500,809	0.00274

4.1.2. Evaluation metrics

We adopt two widely used evaluation protocols: Recall@K and NDCG@K, where K = 20 following NGCF and LightGCN [15,23]. The Recall measures the number of items that the user likes in the test data that have been successfully predicted in the top-k ranking list. Normalized discounted cumulative gain (NDCG) is a measure of ranking quality that considers the position of the item in the ranking; the higher the position is, the higher the score.

4.1.3. Hyperparameter settings

We implemented our proposed method on the official code of LightGCN.[1] Our code and data are available at GitHub.[2] Similar to LightGCN, the embedding size is fixed to 64 for all models. We optimize the model with adaptive moment estimation (ADAM) [48] and use the default batch size of 2048. Grid search is applied to choose the learning rate and the L2 regularization coefficient over the ranges {0.0001, 0.001, 0.01} and {1e–5, 1e–4, . . . , 1e–2}, respectively. In most cases, the optimal values are 0.001 and 1e-4. The three main parameters introduced by RCL are λ, γ, and τ, and we tune them within the ranges of {0.1, 0.2, . . . , 1.0}, {1, 2, 3}, and {0.1, 0.2, 0.5, 1.0}, respectively. The top-2 items in the sorted N items are used as potentially positive samples. The early stopping and validation strategies are the same as in NGCF and LightGCN.

4.1.4. Compared methods

We make comparisons with two types of methods. These compared methods are introduced in the related work in Section 2.

Popular top-k recommendation methods. Matrix factorization (MF) is the most fundamental method in the recommendation field, and most methods can be seen as developments of MF. NGCF [23], LR-GCCF [24], and LightGCN [15] have recently become strong methods based on GCNs. These graph-based methods have shown superior performance to several methods in the original works.

Loss-function-related methods. NNCF [13] and ENMF [14] are adopted here because they also propose loss functions focusing on different sample processing strategies.

4.2. Performance comparison

4.2.1. Overall comparison with different methods

In Table 3, we summarize the performance in the two groups, as previously introduced. For fair comparisons, the parameters of the compared methods are adjusted by referring to the range given in the original article,hyperparameter discussions, as well as the parameters used in the provided code. The values in Table 3 are low because full ranking with all item candidates is used for evaluation following NGCF and LightGCN. Walid et. al. [49] point out that most metrics are inconsistent under the sampling way which can lead to false discoveries, and sampling should be avoided as much as possible during evaluation. Thus, full ranking with all candidate items are used for evaluation and the results are lower than those of the sampled method [1].

The best results are shown in bold, while underlined scores are the second-best results. Our method consistently obtains optimal results on the three datasets. Random seeds have little impact on the results; for example, the standard deviations of the Recall on the three datasets are ±0.0002, ±0.0001, and ±0.0001. To measure the overall performance on the different datasets, we calculated the average performances of the three datasets of the two metrics [50], which are placed in the last column of the table.

Among all methods, MF is the most basic and performs the worst. The GCN-based methods, NGCF, LR-GCCF and LightGCN, explore higher-order connectivity on graphs, consider the effect of multihop neighbours and thus perform much better and show the superiority of graph convolution-based methods. LR-GCCF and LightGCN improve on the shortcomings of NGCF, with performance significantly better than that of NGCF, which is consistent with the original paper. LightGCN is the optimal approach and is selected as the basis of our method. In particular, LightGCN has the same number of parameters as MF with only user and item embeddings.

Compared with the traditional BPR-based methods, the latter three negative sampling-based loss function methods are also significantly effective. Overall, both NNCF and ENMF are more effective than NGCF and LR-GCCF. They are sometimes able to outperform LightGCN. This shows the potential for improving negative sampling methods with loss functions. ENMF performs better than NNCF because it uses more negative samples. We observe a difference in the performance of NNCF and ENMF on the first two datasets. NNCF performs well on Amazon-Book, but poorly on Yelp2018, and with a large margin. The performance of ENMF is the opposite. This shows that the method based on negative sampling will show instability on different datasets.

LightGCN, NNCF, and ENMF achieve suboptimal results on the three datasets, while our proposed method is the best on all datasets. For RCL, each sample is better weighted for optimization, and potentially positive items are explored for the key problem of recommendation. RCL combines and balances the two factors, and, thus, it performs the best on different datasets. The improvements of our method over the suboptimal results on the three datasets are shown in the last row. These improvements are obvious because the suboptimal results are closer to the third ones. Overall, the average improvement over the three datasets are 14.76% on Recall and 16.72% on NDCG. Our method performs much better than the others on the Amazon-Book dataset.

4.2.2. Comparison with different layers of LightGCN

The results of pLightGCN-RCL and LightGCN with different numbers of layers (1 to 4) on Recall and NDCG metrics are shown in Fig. 3.

Overall, pLightGCN-RCL outperforms LightGCN by a large margin in all layers, which shows the effectiveness of RCL. Specifically, on the 2-layer setting, the Recall improvements on Yelp2018, Amazon-Book, and Pinterest are 11.89%, 38.44%, and 12.93%, respectively, while the NGCD values are 11.90%, 47.93%, and 15.62%, respectively.

Compared to LightGCN, pLightGCN-RCL can obtain optimal results with fewer layers. As the layers increase, we find that pLightGCN-RCL generally reaches excellent results in the 2-layer setting and then changes flatly, while the performance of LightGCN generally rises, which means it needs more layers. Therefore, fewer layers (2 layers) and less computation are suitable for pLightGCN-RCL. Because α_k allows better learning of the importance of each layer, two layers are appropriate. There is no need for multiple layers of GCN because of oversmoothing issues [15].

[1] https://github.com/gusye1234/LightGCN-PyTorch.
[2] https://github.com/haotangxjtu/RCL.

Table 3
Performance comparison.

Method	Yelp2018		Amazon-Book		Pinterest		Average	
	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
MF	0.0441	0.0353	0.0329	0.0249	0.1061	0.0743	0.0610	0.0448
NGCF	0.0579	0.0477	0.0344	0.0263	0.1257	0.0894	0.0727	0.0545
LR-GCCF	0.0591	0.0485	0.0378	0.0292	0.1277	0.0907	0.0749	0.0561
LightGCN	0.0639	0.0525	0.0411	0.0315	0.1389	0.0983	0.0813	0.0608
NNCF	0.0597	0.0490	0.0430	0.0337	0.1305	0.0960	0.0792	0.0596
ENMF	0.0627	0.0515	0.0362	0.0285	0.1446	0.1043	0.0812	0.0614
pLightGCN-RCL(ours)	**0.0696**	**0.0564**	**0.0575**	**0.0476**	**0.1528**	**0.1110**	**0.0933**	**0.0717**
Improvement	8.92%	7.43%	33.72%	41.25%	5.67%	6.42%	14.76%	16.72%

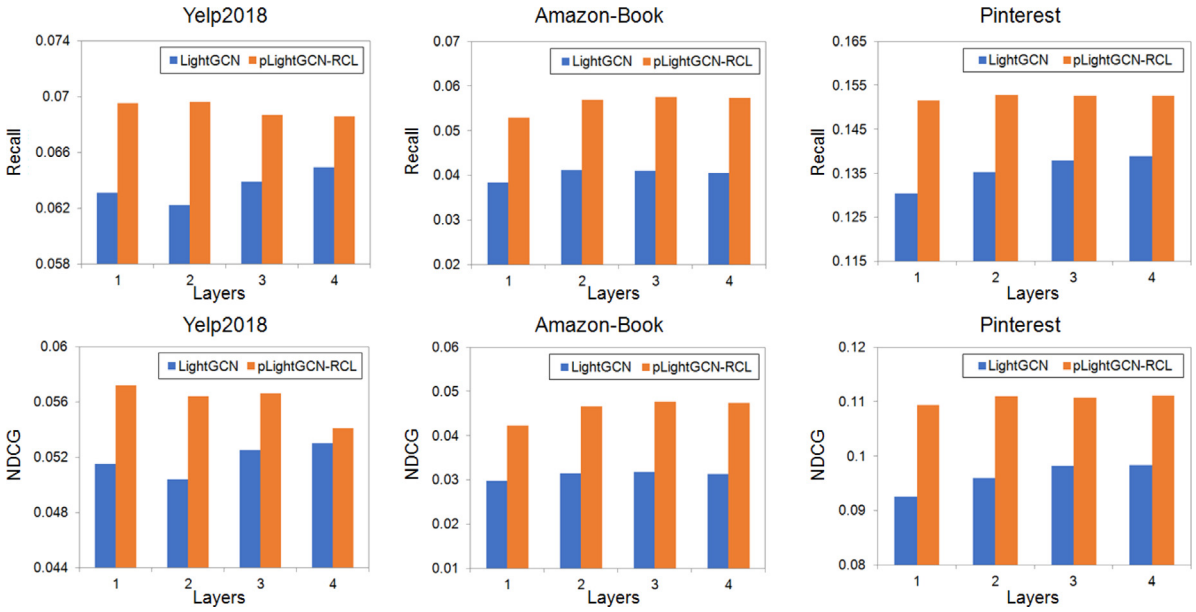


Fig. 3. Performance comparison with different numbers of LightGCN layers.

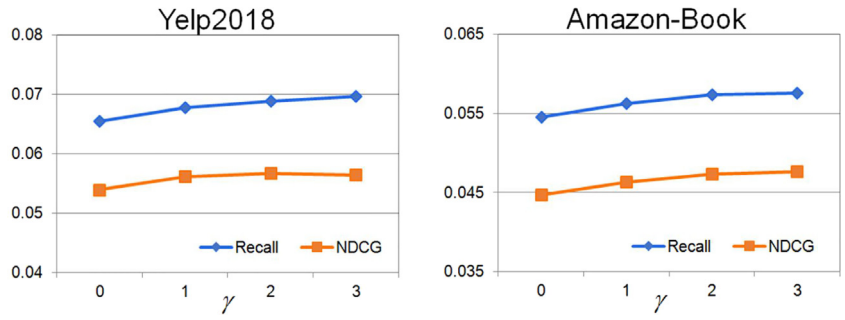


Fig. 4. Impact of γ on the weight function.

4.3. Hyperparameter sensitivity analysis

Experimental findings indicate that Pinterest behaves similarly to Yelp2018 in terms of hyperparameters, so we chose Yelp2018 and Amazon-Book as representative datasets to present the results. In addition, as shown in the figure, the performance changes of different datasets vary regularly with the hyperparameters. Thus, the hyperparameters can be easily adjusted.

4.3.1. Impact of γ on the weight function

Fig. 4 shows that both metrics generally increase as γ increases. The weight is not added when $\gamma = 0$, and all datasets gain the worst performance. This illustrates the effectiveness of the weighting methods and the correctness of mining hard samples. The results in the figure are consistent with the theoretical analysis. The larger γ is, the greater the increase in the weight of the hard samples and the greater the decrease of the simple samples. Therefore, a better effect will generally be achieved when γ is added and has a large value. In addition, 3 can be used as a default setting for each dataset.

4.3.2. Impact of λ of potentially positive samples

As seen from Fig. 5, $\lambda = 0$ is the case without potentially positive samples. Better results can be obtained by adjusting it. Yelp2018 rises and then falls, reaching the best results when $\lambda = 0.2$ (Pinterest is 0.3). Amazon-Book continues to increase, obtaining the best result at $\lambda = 1$. This shows the effectiveness of adding potentially positive samples. The λ indicates how important the

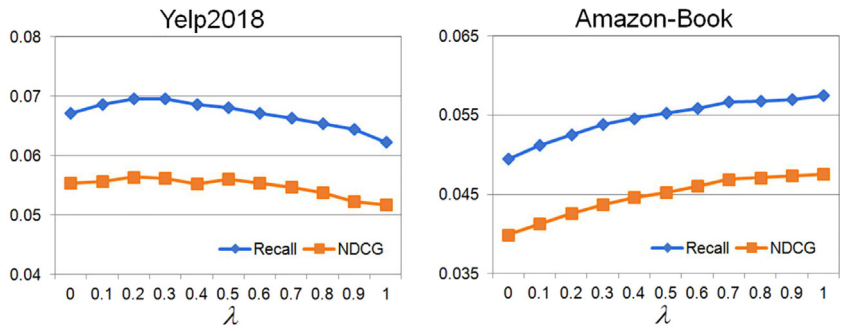


Fig. 5. Impact of the coefficient λ of potentially positive samples.

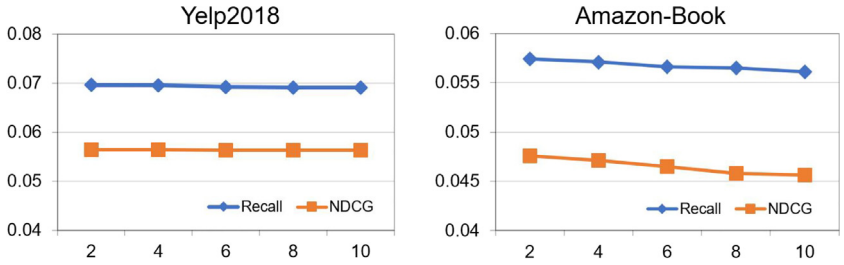


Fig. 6. Impact of the number of potentially positive samples.

Table 4
Ablation study.

Methods	BPR	CL	para	WCL	PCL	Yelp2018		Amazon-Book		Pinterest	
						Recall	NDCG	Recall	NDCG	Recall	NDCG
LightGCN	✓					0.0622	0.0504	0.0410	0.0318	0.1353	0.0960
pLightGCN	✓		✓			0.0427	0.0335	0.0270	0.0204	0.1100	0.0772
LightGCN-CL		✓				0.0631	0.0521	0.0452	0.0355	0.1396	0.1003
pLightGCN-CL		✓	✓			0.0646	0.0534	0.0516	0.0417	0.1418	0.1024
pLightGCN-WCL		✓	✓	✓		0.0672	0.0554	0.0495	0.0399	0.1476	0.1069
pLightGCN-PCL		✓	✓		✓	0.0655	0.0539	0.0545	0.0447	0.1456	0.1049
pLightGCN-RCL		✓	✓	✓	✓	0.0696	0.0564	0.0575	0.0476	0.1528	0.1110

For convenience, we name the methods for each row, which are displayed in the first column. BPR, CL, WCL, PCL, and RCL are loss functions. The para term indicates that we set LightGCN α_k parameters as trainable parameters. The ablation analysis is performed on the optimal number of layers (2, 3, and 2 in the three datasets).

potentially positive example is in the overall picture, where it is relatively more important for Amazon-Book. This confirms the previous finding that Amazon-Book benefits the most from potentially positive samples. The adjustment of λ to fit different datasets can deepen the understanding of the data and the reason for the performance improvement.

4.3.3. Impact of the number of potentially positive samples

The experimental results are shown in Fig. 6. Both the Recall and NDCG slightly decline with the increase of the number of potentially positive samples, with Yelp2018 decreasing insignificantly. This decline may be because the similarity of the positive samples used for the user decreases as the number increases. Therefore, the number of potentially positive samples has little effect on the results, and more positive samples do not lead to performance gains. We use 2 potentially positive samples in all experiments.

4.4. Ablation study

4.4.1. Effectiveness of CL functions

The first four methods in Table 4 are used to illustrate the improvement of LightGCN under the CL function. The first line is the original LightGCN method, where α_k is set to $1/(k+1)$. The comparison of the three methods, LightGCN, LightGCN-CL, and pLightGCN-CL, shows the effectiveness of α_k and CL with no exception. The performance of pLightGCN drops significantly compared with LightGCN. Thus, α_k can improve the performance, but it is important to use CL at the same time; otherwise, the performance will become significantly worse with BPR loss.

It is the unpowerful BPR that results in the model converging to a poor local optimum. For example, on the Yelp2018 dataset with $k = 2$, we obtain α_0, α_1, and α_2 values under BPR of 0.122, 0.605, and 0.2723, respectively, while those under RCL are 0.019, 0.075, and 0.906, respectively. Obviously, the second plays the most important role under RCL, and thus, better results are achieved. This is consistent with the LightGCN-single variant in LightGCN [15], which uses only the second layer as the final embedding and achieves better results. Even after pretraining some epochs and then starting to use α_k, there is still a significant drop in performance with our additional experiments. With the help of the CL function, LightGCN-CL can achieve quicker and better convergence. These comparisons clearly show the superiority of CL over BPR.

It should be noted that the number of α_k parameters is very small, which is equal to the number of layers. These very few but critical parameters improve the performance by giving the weights of the layers explicitly, which can be seen in Section 4.5, Layer Importance.

Table 5
Applicability of RCL.

Methods	Yelp2018		Amazon-Book		Pinterest	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
MF-BPR	0.0441	0.0353	0.0329	0.0249	0.1061	0.0743
MF-RCL	0.0622(41.04%)↑	0.0508(43.91%)↑	0.0477(44.98%)↑	0.0355(42.57%)↑	0.1403(32.23%)↑	0.0999(34.45%)↑
NGCF-BPR	0.0579	0.0477	0.0344	0.0263	0.1257	0.0894
NGCF-RCL	0.0685(18.31%)↑	0.0565(18.45%)↑	0.0537(56.10%)↑	0.0425(61.60%)↑	0.1466(16.63%)↑	0.1054(17.90%)↑
LR-GCCF-BPR	0.0591	0.0485	0.0378	0.0292	0.1277	0.0907
LR-GCCF-RCL	0.0683(15.57%)↑	0.0559(15.26%)↑	0.0484(28.04%)↑	0.0381(30.48%)↑	0.1477(15.66%)↑	0.1065(17.42%)↑
LightGCN-BPR	0.0639	0.0525	0.0411	0.0315	0.1389	0.0983
pLightGCN-RCL	**0.0696**(7.24%)↑	**0.0564**(6.42%)↑	**0.0575**(39.90%)↑	**0.0476**(51.11%)↑	**0.1528**(10.01%)↑	**0.1110**(12.92%)↑

The performance of LightGCN on Yelp2018 and Amazon-Book is taken from the original paper. The best results are shown in bold. Performance improvements relative to BPR loss are shown in brackets.

4.4.2. Effectiveness of the proposed improvements on CL

The last four rows of Table 4 are used to illustrate the effectiveness of RCL improvements.

Effectiveness of WCL. The overall performance of pLightGCN-CL vs. pLightGCN-WCL and pLightGCN-PCL vs. pLightGCN-RCL illustrates the effectiveness of this idea.

Effectiveness of PCL. Both pLightGCN-CL vs. pLightGCN-PCL and pLightGCN-WCL vs. pLightGCN-RCL illustrate the effectiveness of the potentially positive samples. In addition, Amazon-Book benefits most from this idea.

Effectiveness of combining WCL and PCL. The comparison of pLightGCN-RCL vs. pLightGCN-WCL and pLightGCN-RCL vs. pLightGCN-PCL show that when both losses are combined, the performance clearly increases, rather than decreasing or saturating. WCL and PCL solve the problem of recommendation system from different perspectives, so both of them can still continue to improve the effect when used together. In addition, RCL still has a large improvement on the Pinterest dataset relative to WCL and PCL, showing that the two ideas may have a mutually reinforcing effect on some datasets. And with explicit samples mining and target constraints, the reasons for improvements are explainable. This not only facilitates practical applications but also can inspire other new methods.

Sensitivity analysis on different datasets. Overall, Amazon-Book is more sensitive to potentially positive samples because the two metrics significantly improved. Yelp2018 and Pinterest are more sensitive to adding weights. This indicates that our two ideas will vary depending on the different datasets, but they are generally effective.

In conclusion, the improvements proposed in this paper are effective, and the RCL function that combines the two improvements is always more effective and performs best.

4.5. Applicability of RCL

We replace BPR with RCL to illustrate the applicability of RCL. In Table 5, "*-BPR" is the original method with BPR loss, and "*-RCL" is the method with the RCL function. We make the following observations:

All methods using RCL gain improvements by a large margin compared with those using BPR loss. NGCF, LRGCCF, and LightGCN have recently become the best three methods, and RCL makes them more competitive.

MF is the most fundamental method based on e_u, e_i, while the other methods are based on the transformation of e_u, e_i. Theoretically, RCL is suitable for all embedding-based methods. Therefore, the success of MF-RCL indicates that RCL can be applied to a wide range of methods in this field.

We sort the four methods according to their performance in ascending order. If we observe the consecutive three rows, method-BPR, method-RCL, and adjacent methods-BPR, we find that method-RCL always performs the best. This means that the improvements of RCL over BPR on the same method are always greater than the improvements between the adjacent methods with the original BPR loss. That is, experiments show that our proposed RCL outperforms the new method. In particular, MF is the most basic method in the recommendation field, and RCL makes MF much better than competing methods NGCF and LR-GCCF with BPR.

In particular, MF is the most basic method in the recommendation domain, and RCL significantly improves the performance of the most basic method MF. Overall, MF-RCL outperforms NGCF-BPR, LR-GCCF-BPR, and LightGCN-BPR on three datasets, with only one exception on Yelp2018. This shows the powerful ability of RCL to optimize the latent vectors directly. As the simplest recommendation method, MF-RCL can be widely used in large-scale industrial scenarios. It also shows the fundamental role and importance of the loss function for recommendation, which can be adapted to many models and significantly improve the results.

In addition, we find that the improvements on the three datasets are different, which may be because of distinct dataset characteristics, especially because there is a substantial improvement on the Amazon-Book dataset with RCL.

4.6. Discussion

4.6.1. RCL is suitable for top-k recommendation

BPR is directly optimized for ranking, which usually uses one or several comparison pairs, while RCL employs thousands. RCL improves the quality of comparison by our ideas. In particular, the ranking is optimized by assigning different weights according to different positions, and the learning and use of top-ranked samples are also enhanced. Therefore, RCL is more suitable for ranking tasks than BPR in theory. Experimentally, the NDCG of RCL has been significantly improved compared with BPR in all test datasets in Table 5.

NDCG is a ranking-related metric that is more meaningful for ranking and top-k recommendation tasks than Recall. In Table 5, the mean improvements of the four methods of Recall on Yelp2018, Amazon-Book, and Pinterest are 20.54%, 42.26%, 18.63%, respectively, while those of NDCG are 21.01%, 46.64%, 20.67%, respectively. The boost of NDCG is generally greater than that of Recall, also indicating that RCL is more suitable for top-k recommendations.

4.6.2. Layer importance

By introducing the CL function, we successfully learn α_k, as shown in Fig. 7. To the best of our knowledge, this is the first quantitative representation of the importance of each layer in the GCN domain.

We conduct a normalization such that $\sum_{k=0}^{K} \alpha_k = 1$. α_k represents the embedding weight of the kth layer, and α_0 represents

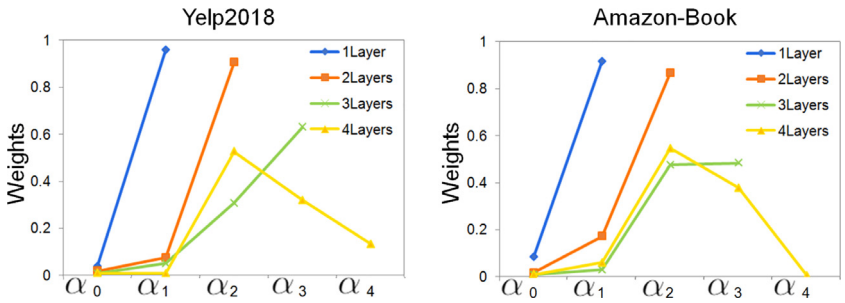


Fig. 7. Layer Importance. For the first time, α_k shows the importance of each layer quantitatively.

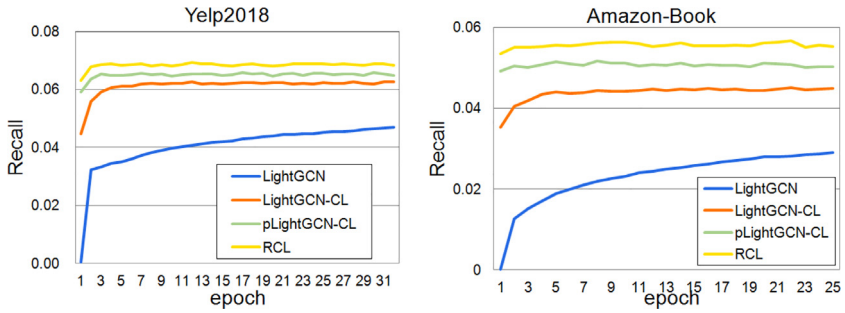


Fig. 8. Training Efficiency. The vertical axis is the test performance of each epoch of LightGCN, LightGCN-CL, pLightGCN-CL and RCL.

the original embedding. As the number of layers increases, the importance of layers 2 and 3 increases, and the importance of layers 0 and 1 decreases significantly. The greater the number of layers, the less important layers 0 and 1 are. When $k = 4$, layer 2 is the most important, and the weight of layer 4 markedly decreases. This is consistent with the oversmoothing problems faced by multilayer GCNs in which performance gains are no longer realized. However, we are surprised to find that layers 0 and 1 are of low importance. This finding can clearly explain why better results can be obtained in LightGCN [15] by using only layer 2.

This also demonstrates the advantages of GCNs over traditional methods that GCNs exploit the graph structure to aggregate information of higher-order neighbours. The results show that, 2 or 3 is the appropriate number of layers in general, and the second layer is important. In case of 4 layers, the weights of the third and fourth layers decrease rapidly, and the fourth layer plays almost no role. This also verifies the theory of GCNs from a quantitative perspective: more layers are not always better due to the over-smoothing problem.

4.6.3. Training efficiency

Experiments indicate a significant improvement in both the training efficiency and performance of the RCL. Fig. 8 shows the test performance of each epoch of LightGCN, LightGCN-CL, pLightGCN-CL and LightGCN-RCL in terms of Recall. For comparison purposes, the figure shows the performance of LightGCN in the early stages of the training process. For LightGCN, more than 700 epochs are required to obtain the optimal results, while RCL achieves the best results at 12 and 22 epochs on the two datasets. In all compared methods in Fig. 8, it is obvious that RCL has the fastest convergence and the best results. RCL outperforms the other methods on the Amazon-Book dataset with just one iteration. Such significant training efficiency gains are found on all the methods in Table 5 with the RCL function.

Meanwhile, the training time per epoch is almost unchanged. Specifically, for the three datasets using BPR, the time consumptions per epoch are approximately 13 s, 11 s, and 64 s, and those of RCL are approximately 15 s, 13 s, and 67 s. RCL requires more inner product computation and sorting than BPR. The batch sizes are typically 1024 and 2048, which are not large numbers. Thus, it can be accelerated by deep learning frameworks and the graphics processing units (GPUs) with almost no increase in time consumption. RCL can easily utilize negative samples from the same batch, unlike BPR, which requires a negative sampling process.

RCL significantly improves the convergence speed with almost no increase in training time, which greatly improves training efficiency. We believe this is mainly because N-1 negative samples are used for learning simultaneously, as demonstrated in the NNCF [13].

4.7. Compared to MSCL

MSCL [45] is our previous work whose Recall on Yelp2018, Amazon-Book and Pinterest datasets are 0.0691, 0.0580 and 0.1525 respectively, and NDCG are 0.0568, 0.0466 and 0.1104 respectively. This RCL-based method has similar performance but slightly outperforms the MSCL-based method overall, as it shows better results than MSCL in four of the six results. The RCL focuses on mining the samples from multiple perspectives while the MSCL addresses the issue of how to make the best use of the available samples by distinguishing the different importance of the samples and using a combination of multiple positive samples. RCL mines hard samples and potentially positive samples which are important problems for recommendation systems. RCL is a new approach for sample mining just based on interaction data, which is more intuitive, simple, and can be applied to various recommendation methods in theory.

5. Conclusion and future work

In this paper, we propose a simple and effective loss function RCL for recommendation systems. The interesting phenomena that many negative items are before the positive ones after ranking by similarities inspire us to mine hard samples and potentially

positive samples conveniently. RCL is proposed based on the CL by weighting and using potentially positive samples. RCL shows significant improvements in performance and training efficiency and has good adaptability.

Some directions related to this paper are worth exploring. On the one hand the mining and utilization of samples in recommendation systems is very important, on the other hand the key to CL is to use samples to make better comparisons, so they can be well combined and achieve good results. A good loss function is more valuable for the recommendation because the loss function can directly optimize the model to improve the performance and can be widely used with many methods. CL can still be improved and applied to recommendation systems for different tasks. For example, we find that under the constraint of CL, the spatial distribution of sample embeddings is too compact leading to over-smoothing of features, which could be the basis of a future work.

CRediT authorship contribution statement

Hao Tang: Methodology, Software, Data curation, Writing – original draft, Formal analysis. **Guoshuai Zhao:** Conceptualization, Methodology, Formal analysis, Supervision. **Yujiao He:** Writing – review & editing. **Yuxia Wu:** Methodology, Formal analysis. **Xueming Qian:** Resources, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

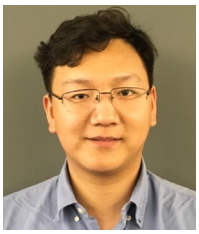
This work was supported in part by the NSFC, China under Grants 61902309; in part by the Fundamental Research Funds for the Central Universities, China (xzd012022006); in part by China Postdoctoral Science Foundation (2020M683496); in part by the National Postdoctoral Innovative Talents Support Program, China (BX20190273); in part by the Humanities and Social Sciences Foundation of Ministry of Education, China under Grant 16XJAZH003; and in part by the Science and Technology Program of Xi'an, China under Grant 21RGZN0017. (*Corresponding author* : *Guoshuai Zhao*.)

References

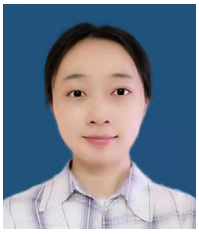
[1] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, Tat-Seng Chua, Neural collaborative filtering, in: WWW, 2017, pp. 173–182.
[2] Yuxia Wu, Ke Li, Guoshuai Zhao, Xueming QIAN, Personalized long- and short-term preference learning for next POI recommendation, IEEE Trans. Knowl. Data Eng. 34 (4) (2022) 1944–1957.
[3] Guoshuai Zhao, Peiliang Lou, Xueming Qian, Xingsong Hou, Personalized location recommendation by fusing sentimental and spatial context, Knowl.-Based Syst. 196 (2020) 105849.
[4] Yuxia Wu, Lizi Liao, Gangyi Zhang, Wenqiang Lei, Guoshuai Zhao, Xueming Qian, Tat-Seng Chua, State graph reasoning for multimodal conversational recommendation, IEEE Trans. Multimed. (2022).
[5] Hao Tang, Guoshuai Zhao, Xuxiao Bu, Xueming Qian, Dynamic evolution of multi-graph based collaborative filtering for recommendation systems, Knowl.-Based Syst. 228 (2021) 107251.
[6] Ruiping Yin, Kan Li, Guangquan Zhang, Jie Lu, A deeper graph neural network for recommender systems, Knowl.-Based Syst. 185 (2019).
[7] Hyunsung Lee, Sangwoo Cho, Yeongjae Jang, Jaekwang Kim, Honguk Woo, Differentiable ranking metric using relaxed sorting for Top-K recommendation, IEEE Access 9 (2021) 114649–114658.
[8] Mingming Yang, Songhua Xu, A novel deep quantile matrix completion model for top-N recommendation, Knowl.-Based Syst. 228 (2021) 107302.
[9] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, Lars Schmidt-Thieme, BPR: Bayesian personalized ranking from implicit feedback, in: UAI, 2009, pp. 452–461.
[10] Ting Chen, Simon Kornblith, Mohammad Norouzi, Geoffrey E. Hinton, A simple framework for contrastive learning of visual representations, in: ICML, Vol. 119, in: Proceedings of Machine Learning Research, PMLR, 2020, pp. 1597–1607.
[11] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, Yang Shen, Graph contrastive learning with augmentations, in: NeurIPS, 2020.
[12] Phuc H. Le-Khac, Graham Healy, Alan F. Smeaton, Contrastive representation learning: A framework and review, IEEE Access 8 (2020) 193907–193934.
[13] Ting Chen, Yizhou Sun, Yue Shi, Liangjie Hong, On sampling strategies for neural network-based collaborative filtering, in: KDD, 2017, pp. 767–776.
[14] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, Shaoping Ma, Efficient neural matrix factorization without sampling for recommendation, ACM Trans. Inf. Syst. 38 (2) (2020) 14:1–14:28.
[15] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, Meng Wang, LightGCN: Simplifying and powering graph convolution network for recommendation, in: SIGIR, ACM, 2020, pp. 639–648.
[16] Bo Yang, Jing Chen, Zhongfeng Kang, Dongsheng Li, Memory-aware gated factorization machine for top-N recommendation, Knowl.-Based Syst. 201–202 (2020) 106048.
[17] Lei Xu, Chunxiao Jiang, Yan Chen, Yong Ren, K. J. Ray Liu, User participation in collaborative filtering-based recommendation systems: A game theoretic approach, IEEE Trans. Cybern. 49 (4) (2019) 1339–1352.
[18] Guoshuai Zhao, Zhidan Liu, Yulu Chao, Xueming Qian, CAPER: context-aware personalized emoji recommendation, IEEE Trans. Knowl. Data Eng. 33 (9) (2021) 3160–3172.
[19] Rui Chen, Qingyi Hua, Yan-shuo Chang, Bo Wang, Lei Zhang, Xiangjie Kong, A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks, IEEE Access 6 (2018) 64301–64320.
[20] Ruiqin Wang, Yunliang Jiang, Jungang Lou, ADCF: Attentive representation learning and deep collaborative filtering model, Knowl.-Based Syst. 227 (2021) 107194.
[21] Shi-Ting Zhong, Ling Huang, Chang-Dong Wang, Jian-Huang Lai, Philip S. Yu, An autoencoder framework with attention mechanism for cross-domain recommendation, IEEE Trans. Cybern. 52 (6) (2022) 5229–5241.
[22] Junmei Hao, Yujie Dun, Guoshuai Zhao, Yuxia Wu, Xueming Qian, Annular-graph attention model for personalized sequential recommendation, IEEE Trans. Multimed. 24 (2022) 3381–3391.
[23] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, Tat-Seng Chua, Neural graph collaborative filtering, in: SIGIR, 2019, pp. 165–174.
[24] Lei Chen, Le Wu, Richang Hong, Kun Zhang, Meng Wang, Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach, in: AAAI, AAAI Press, 2020, pp. 27–34.
[25] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, Tat-Seng Chua, Disentangled graph collaborative filtering, in: SIGIR, 2020, pp. 1001–1010.
[26] Zhibin Hu, Jiachun Wang, Yan Yan, Peilin Zhao, Jian Chen, Jin Huang, Neural graph personalized ranking for Top-N Recommendation, Knowl.-Based Syst. 213 (2021) 106426.
[27] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, Dilip Krishnan, Supervised contrastive learning, in: NeurIPS, 2020.
[28] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, Stefanie Jegelka, Debiased contrastive learning, in: NeurIPS, 2020.
[29] Dapeng Feng, Songfang Han, Hang Xu, Xiaodan Liang, Xiaojun Tan, Point-guided contrastive learning for monocular 3-D object detection, IEEE Trans. Cybern. (2021) 1–13.
[30] Jun Zhao, Minglai Shao, Hao Peng, Hong Wang, Bo Li, Xudong Liu, Porn2Vec: A robust framework for detecting pornographic websites based on contrastive learning, Knowl.-Based Syst. 228 (2021) 107296.
[31] Gal Chechik, Varun Sharma, Uri Shalit, Samy Bengio, Large scale online learning of image similarity through ranking, J. Mach. Learn. Res. 11 (2010) 1109–1135.
[32] Florian Schroff, Dmitry Kalenichenko, James Philbin, Facenet: A unified embedding for face recognition and clustering, in: CVPR, 2015, pp. 815–823.
[33] Kihyuk Sohn, Improved deep metric learning with multi-class n-pair loss objective, in: NIPS, 2016, pp. 1857–1865.
[34] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, Yoshua Bengio, Learning deep representations by mutual information estimation and maximization, in: ICLR, 2019.
[35] Philip Bachman, R. Devon Hjelm, William Buchwalter, Learning representations by maximizing mutual information across views, in: NeurIPS, 2019, pp. 15509–15519.

[36] Janine Thoma, Danda Pani Paudel, Luc Van Gool, Soft contrastive learning for visual localization, in: NeurIPS, 2020.
[37] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, Yang Shen, Graph contrastive learning with augmentations, in: NeurIPS, 2020.
[38] Kaveh Hassani, Amir Hosein Khas Ahmadi, Contrastive multi-view representation learning on graphs, in: ICML, Vol. 119, PMLR, 2020, pp. 4116–4126.
[39] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, Xing Xie, Self-supervised graph learning for recommendation, in: SIGIR, ACM, 2021, pp. 726–735.
[40] Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, Depeng Jin, Simplify and robustify negative sampling for implicit collaborative filtering, in: NeurIPS, 2020.
[41] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, Depeng Jin, Reinforced negative sampling for recommendation with exposure data, in: IJCAI, ijcai.org, 2019, pp. 2230–2236.
[42] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, Dell Zhang, IRGAN: A minimax game for unifying generative and discriminative information retrieval models, in: SIGIR, ACM, 2017, pp. 515–524.
[43] Dae Hoon Park, Yi Chang, Adversarial sampling and training for semi-supervised information retrieval, in: WWW, ACM, 2019, pp. 1443–1453.
[44] Jessa Bekker, Jesse Davis, Learning from positive and unlabeled data: a survey, Mach. Learn. 109 (4) (2020) 719–760.
[45] Hao Tang, Guoshuai Zhao, Yuxia Wu, Xueming Qian, Multisample-based contrastive loss for top-k recommendation, IEEE Trans. Multimed. (2021).
[46] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver J. Woodford, Meng Jiang, Neil Shah, Data augmentation for graph neural networks, in: AAAI, AAAI Press, 2021, pp. 11015–11023.
[47] Xue Geng, Hanwang Zhang, Jingwen Bian, Tat-Seng Chua, Learning image and user features for recommendation in social networks, in: ICCV, IEEE Computer Society, 2015, pp. 4274–4282.
[48] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, in: ICLR, 2015.
[49] Walid Krichene, Steffen Rendle, On sampled metrics for item recommendation, in: KDD, ACM, 2020, pp. 1748–1757.
[50] Guoshuai Zhao, Xueming Qian, Chen Kang, Service rating prediction by exploring social mobile users' geographical locations, IEEE Trans. Big Data 3 (1) (2017) 67–78.

Hao Tang received a B.E. degree from Information Engineering University, Zhengzhou, China, in 2011, and an M.E. degree from Shandong University of Science and Technology, Qingdao, China, in 2013. After working for 5 years, he is currently working toward a Ph.D. degree at SMILES LAB, Xi'an Jiaotong University, Xi'an, China. His current research interests include recommendation systems and graph neural networks.



Guoshuai Zhao received the B.E. degree from Heilongjiang University, Harbin, China, in 2012, the M.S. degree and Ph.D. degree from Xi'an Jiaotong University, Xi'an, China, in 2015 and 2019 respectively. He was an intern with the Social Computing Group at Microsoft Research Asia from January 2017 to July 2017, and was a visiting scholar with Northeastern University, U.S., from October 2017 to October 2018 and with MIT, U.S., from June 2019 to December 2019. Now he is an Associate Professor with Xi'an Jiaotong University. His research interests include social media big data analysis, recommendation systems, and natural language generation.



Yujiao He received the B.E. degree from Xi'an Jiaotong University, Xi'an, China, in 2020, and is expected to get the MS degree from Xi'an Jiaotong University in 2023. Her current research interests include social media big data analysis and recommender systems.

Yuxia Wu received the B.S. degree from Zhengzhou University, Henan, China, in 2014, the M.S. degree from the Fourth Military Medical University, Xi'an, China, in 2017, and is currently working toward the Ph.D. degree at Xi'an Jiaotong University, Xi'an, China. She is mainly engaged in the research of social multimedia mining and recommendation systems.

Xueming Qian received the B.S.and M.S. degrees from Xi'an University of Technology, Xi'an, China, in 1999 and 2004, respectively, and the Ph.D. degree from the School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an, China, in 2008. He was awarded the Microsoft Fellowship in 2006. From 1999 to 2001, he was an Assistant Engineer at Shannxi Daily. Since 2008, he has been an Associate Professor in the School of Electronics and Information Engineering, Xi'an Jiaotong University. Now, he is an Associate Professor in the School of Electronics and Information Engineering, Xi'an Jiaotong University. He is the Director of SMILES LAB. He was a Visiting Scholar at Microsoft Research Asia from August 2010 to March 2011. His research interests include social media big data mining and search. He is a member of the IEEE, ACM, and Senior Member of CCF.