Graph Contrastive Learning with Adaptive Augmentation for Recommendation

Mengyuan Jing, Yanmin Zhu(✉), Tianzi Zang, Jiadi Yu, and Feilong Tang

Shanghai Jiao Tong University, Shanghai, China
✉ Corresponding Author
{jingmy, yzhu, zangtianzi, jiadiyu, tang-fl}@sjtu.edu.cn

Abstract. Graph Convolutional Network (GCN) has been one of the most popular technologies in recommender systems, as it can effectively model high-order relationships. However, these methods usually suffer from two problems: sparse supervision signal and noisy interactions. To address these problems, graph contrastive learning is applied for GCN-based recommendation. The general framework of graph contrastive learning is first to perform data augmentation on the input graph to get two graph views and then maximize the agreement of representations in these views. Despite the effectiveness, existing methods ignore the differences in the impact of nodes and edges when performing data augmentation, which will degrade the quality of the learned representations. Meanwhile, they usually adopt manual data augmentation schemes, limiting the generalization of models. We argue that the data augmentation scheme should be learnable and adaptive to the inherent patterns in the graph structure. Thus, the model can learn representations that remain invariant to perturbations of unimportant structures while demanding fewer resources. In this work, we propose a novel **G**raph **C**ontrastive learning framework with **A**daptive data augmentation for **Rec**ommendation (GCARec). Specifically, for adaptive augmentation, we first calculate the retaining probability of each edge based on the attention mechanism and then sample edges according to the probability with a Gumbel Softmax. In addition, the adaptive data augmentation scheme is based on the neural network and requires no domain knowledge, making it learnable and generalizable. Extensive experiments on three real-world datasets show that GCARec outperforms state-of-the-art baselines.

Keywords: Recommender systems · Graph neural network · Contrastive learning · Self-supervised learning.

1 Introduction

Recommender systems have been an indispensable component in many online services, such as E-commerce platforms and online entertainment applications, for their effectiveness in alleviating information overloading. Recently, graph convolution network (GCN) has become a state-of-the-art method in recommender systems [2, 10, 23, 24], as it can integrate high-order neighbors in the graphs.

Despite the effectiveness, GCN-based recommendation models still suffer from two problems [25]. (1) sparse supervision signal: most GCN-based methods focus on supervised settings [10, 23], where the supervision signal is derived from the observed interactions. However, the observed interactions are very sparse in comparison to the entire interaction space [1, 34], limiting the performance of these methods. (2) noisy interactions. The interaction data usually contain noises, since users often provide implicit feedback like clicks rather than explicit feedback like ratings. There may be cases where users click or purchase items and find they do not like them [28]. Since GCN-based models learn representations by aggregating information from neighbors, noisy interactions will make GCNs fail to learn reliable representations.

To address these problems, we apply graph contrastive learning [32] in GCN-based recommendation. A general framework of graph contrastive learning is to perform data augmentation on the input graph by uniform node/edge dropout to generate two graph views and then maximize the agreement of representations in these views. It extracts additional supervised signals from the input data and learns representations that remain invariant to the perturbations introduced by the data augmentation, making it possible to solve both problems mentioned above. Although several works [15, 25, 33] leverage graph constative learning in GCN-based recommendation, data augmentation schemes, a key component of contrastive learning [27] are still rarely explored in graph contrastive learning-based recommendation methods. Specifically, there are currently two main types of data augmentation schemes. (1)stochastic augmentation [25], such as uniformly dropping a portion of edges or nodes from the input graph; (2) artificially designed augmentation [15, 33], which usually constructs views from additional domain knowledge, such as social networks and knowledge graphs. We argue that these augmentation schemes have two limitations:

First, they ignore the differences in the impact of nodes and edges when performing data augmentation. Taking uniform edge dropout as an example, dropping important edges (e.g., edges that connect items of great interest to a user) for a user will degrade the quality of its representation. Besides, keeping unimportant edges (e.g., the noisy interactions) can be harmful to representation learning. Second, they choose augmentation schemes either based on domain knowledge or performance evaluation on the validation set. However, domain knowledge is not always available and performance evaluation is usually expensive, limiting the generalization of the model to other datasets. Hence, we propose that the data augmentation scheme should be learnable and adaptive to the intrinsic patterns in the graph structure. Such that, it can guide the model to keep the important connective structures while perturbing the unimportant edges in the graph, as well as demanding fewer resources compared with manual selection.

To this end, we propose a novel **G**raph **C**ontrastive learning framework with **A**daptive data augmentation for **Rec**ommendation (GCARec). Specifically, we design a learnable data augmentation scheme based on the neural network, whose parameters are learned from data during the optimization of the contrastive

objective. Meanwhile, as introduced above, the data augmentation scheme should preserve important edges while perturbing unimportant ones. To achieve this goal, we make the preserving probability of important edges higher and the probability of unimportant ones lower. Specifically, the probability is calculated based on the attention mechanism to reflect the importance of each edge. After getting the probability, we sample edges according to them. However, the random sampling strategy is not differentiable and will make the model cannot be trained. Hence, we propose a view generation method based on a Gumbel Softmax [11], which introduces a continuous distribution to approximate categorical samples, to address this issue. After that, we apply contrastive learning based on node self-discrimination to maximize the agreement between the representations in the generated views. Finally, we leverage the contrastive learning task (i.e., node self-discrimination) as the auxiliary task to the recommendation task and jointly train them using the multi-task training strategy.

In summary, the contributions of our work are as follows:

1. We propose a novel graph contrastive learning framework with adaptive data augmentation, which encourages the model to learn important structural information in the graph.
2. We designed a learnable data augmentation scheme that not only requires less human effort but also easily generalizes to different graph-based recommendation scenarios.
3. Extensive experiments are conducted on three real-world datasets. The results indicate that our model outperforms the state-of-art methods and demonstrates the effectiveness of the adaptive data augmentation scheme.

The remainder of this paper is organized as follows. We first introduce the notations and GCN-based recommendation in Section 2. We then present details of our model in Section 3. Settings and results of conducted experiments are introduced in Section 4. We summarize the related work in Section 5. Finally, we conclude this work and future work in Section 6.

2 Preliminaries

Notations. Let $\mathcal{U} = \{u_1, u_2, \cdots, u_m\}(|\mathcal{U}| = m)$ and $\mathcal{I} = \{i_1, i_2, \cdots, i_n\}(|\mathcal{I}| = n)$ denote the set of users and items, respectively. Let $\mathcal{O}^+ = \{y_{u,i} | u \in \mathcal{U}, i \in \mathcal{I}\}$ denote the observed interactions, where $y_{u,i}$ indicates that user u has interacted with item i. Moreover, a user-item interaction graph is constructed, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ is the set of nodes and $\mathcal{E} = \{(u, i) | y_{u,i} \in \mathcal{O}^+, u \in \mathcal{U}, i \in \mathcal{I}\}$ is the edge set.

Recap GCN. Generally, at each layer of GCN-based recommendation models, two key computations, i.e., aggregate and update, are involved to generate the node representations, which can be formulated as follows:

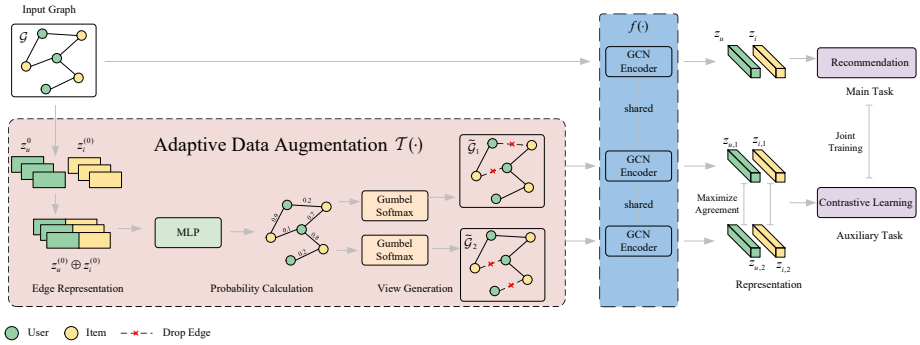


Fig. 1. Overall framework illustration of our proposed GCARec. The upper is the recommendation task, which predicts the preference score of users on items. The bottom is the contrastive learning task, which aims to maximize the agreement of representations in different views.

$$
\begin{aligned}
a_u^{(l)} &= f_{\text{aggregate}}(\{z_i^{(l-1)} | i \in \mathcal{N}_u\}), \\
z_u^{(l)} &= f_{\text{update}}(z_u^{(l-1)}, a_u^{(l)}),
\end{aligned} \tag{1}
$$

where $\mathcal{N}_u$ denotes the neighbors of node u and $a_u^{(l)}$ is the aggregated representation of $\mathcal{N}_u$. $z_u^{(l)}$ is the representation of node u at the l-th layer. $z_u^{(0)}$ is initialized by the learnable embedding. To generate the representation of node u, it first aggregates the representations of $\mathcal{N}_u$ and then updates the representation of u from its representation at $(l-1)$-th layer and the aggregated representations. It can be seen that $z_u^{(l)}$ encodes the l-hop neighbors of u. To produce the final representation for the recommendation, a readout function may be used:

$$
z_u = f_{\text{readout}}\left(\left\{z_u^{(l)} \mid l = [0, \cdots, L]\right\}\right), \tag{2}
$$

where L is the number of layers in the GCN-based model.

After getting the final representations, a prediction layer is built to calculate the preference score, which indicates how likely user u would adopt item i. For fast retrieval, the inner product usually is adopted:

$$
\hat{y}_{u,i} = z_u^T z_i, \tag{3}
$$

where z_u and z_i are final representations of u and i, respectively.

3 Methodology

In this section, we introduce our method in detail. Firstly, in Section 3.1, we describe the overall framework of GCARec. Then in Section 3.2 and Section 3.3,

we present the adaptive data augmentation method to generate different views and the contrastive learning based on node self-discrimination. Finally, in Section 3.4, the multi-task training strategy is introduced.

3.1 The Contrastive Learning Framework

Our framework follows the general contrastive learning paradigms, which seek to achieve maximum agreement between representations of different views. Figure 1 is the illustration of the framework. The basic idea is two folds. On the one hand, we generate two graph views by performing data augmentation on the input graph. On the other hand, we employ a contrastive loss function to conduct the contrastive learning task (i.e., node self-discrimination), which encourages representations of the same nodes in the two different views to be similar, while representations of different nodes in those views to be distinct.

Firstly, we generate two graph views using adaptive data augmentation $\mathcal{T}(\cdot)$, denote as $\tilde{\mathcal{G}}_1 = \mathcal{T}(\tilde{\mathcal{G}})$ and $\tilde{\mathcal{G}}_2 = \mathcal{T}(\tilde{\mathcal{G}})$, where $\mathcal{G}$ is the input graph. Then, $\tilde{\mathcal{G}}_1$ and $\tilde{\mathcal{G}}_2$ are fed to the GCN encoder $f(\cdot)$, after which the encoder outputs representations of nodes in the two generated views. $Z_1 = f(\tilde{\mathcal{G}}_1)$ and $Z_2 = f(\tilde{\mathcal{G}}_2)$ denote the representations in the two views, respectively. Specifically, we adopt LightGCN [10] as the GCN encoder in this work.

Next, we perform the contrastive learning task on the representation of generated views. Since we focus on the data augmentation, we simply treat the same nodes in different views as positive pairs and different nodes in different views as negative pairs. To be specific, for a node $u \in \mathcal{U}$, $\{(z_{u,1}, z_{u,2})|u \in \mathcal{U}, z_{u,1} \in Z_1, z_{u,2} \in Z_2\}$ is the positive pair and $\{(z_{u,1}, z_{u',2})|u, u' \in \mathcal{U}, u' \neq u, z_{u,1} \in Z_1, z_{u',2} \in Z_2\}$ are the negative pairs. Technically, we adopt the InfoNCE [7] as the contrastive loss.

Finally, we adopt a multi-task training strategy to improve the recommendation performance. In specific, the recommendation task is the main task and the contrastive learning task is the auxiliary task.

3.2 Adaptive Augmentation

The user-item interaction graph contains several collaborative filtering signals, as it is constructed based on observed interaction. For example, the first-order neighbors reflect user interest and the second-order neighbors exhibit behavior similarity. Hence, it is useful to mine the inherent patterns in the user-item interaction graph structure. In addition, contrastive learning that maximizes the agreement between views aims to learn representations that remain invariant to the perturbations introduced by data augmentation [30]. Therefore, the data augmentation scheme should preserve important connective structures in the user-item interaction graph while perturbing unimportant ones.

To achieve this goal, we propose an adaptive data augmentation scheme that tends to retain the important edges and perturb possibly unimportant edges in the user-item interaction graph. To be specific, we sample edges in the graph with lower preserving probability for unimportant edges and higher preserving

probability for important ones. Compared with methods that randomly corrupt views, our method puts more emphasis on important structures, which can guide the model to mine the inherent patterns in the graph structure. Two main processes are included in our proposed adaptive data augmentation: probability calculation and view generation.

Formally, we first calculate the preserving probability of each edge (u, i) as $p_{u,i}$. Then, we sample two subsets $\tilde{\mathcal{E}}_1$ and $\tilde{\mathcal{E}}_2$ from the original edge set $\mathcal{E}$ with the probability of each edge to generate two views. We will introduce the details as follows.

Probability Calculation. As we aim to corrupt unimportant edges and keep important structures in the user-item interaction graph, $p_{u,i}$ is required to reflect the importance of edge (u, i). In GCARec, we calculate the preserving probability of each edge based on the attention mechanism, which has been shown to be effective in modeling the importance of the edge. Following GAT [20], we perform self-attention on the nodes using a Multi-Layer Perception (MLP), and then calculate $p_{u,i}$ according to the following equations:

$$\begin{aligned} \alpha_{u,i} &= \mathbf{W}(z_u^{(0)} \oplus z_i^{(0)}) + \mathbf{b}, \\ p_{u,i} &= \frac{\exp(\alpha_{u,i})}{1 + \exp(\alpha_{u,i})} \end{aligned} \tag{4}$$

where $\alpha_{u,i}$ is the attention coefficient, which indicates the importance of i to u. $\oplus$ is the concatenation operation. $\mathbf{W}$ and $\mathbf{b}$ are trainable parameters. $z_u^{(0)}$ and $z_i^{(0)}$ are representations initialized by the learnable embeddings.

View Generation. After getting the preserving probabilities, we can randomly sample edges according to their preserving probabilities. However, random sampling is not differentiable and makes model cannot be trained well via back-propagation. Inspired by [17], we adopt Gumbel Softmax to be the differential surrogate to address this issue. It can be described as follows:

$$g(u, i) = \frac{\exp\left((\log(p_{u,i}) + g_1)/\tau_1\right)}{\sum_{y=0}^{1} \exp\left((\log(p_{u,i}^{y}(1 - p_{u,i})^{1-y}) + g_y)/\tau_1\right)}, \tag{5}$$

where g_y is the noise, which is i.i.d sampled from a Gumbel distribution: $g = -\log(-\log(x))$ and $x \sim \text{Uniform}(0, 1)$. τ_1 is the temperature parameter. When $\tau_1 \to 0$, $\mathcal{G}$ approximates to a one-hot vector. The edge subset of the generated view is $\tilde{\mathcal{E}} = \{(u, i) | g(u, i) > p, (u, i) \in \mathcal{E}\}$, where the threshold p is a hyperparameter to control the removal of unimportant edges.

3.3 Contrastive Learning

As we introduced in Section 3.1, we treat the same nodes in the different views as positive pairs and different nodes in the different views as negative pairs. The loss function of the contrastive learning task can be described as follows:

$$\mathcal{L}_{gcl} = \mathcal{L}_{gcl}^{user} + \mathcal{L}_{gcl}^{item}, \tag{6}$$

$$\mathcal{L}_{gcl}^{user} = \sum_{u \in \mathcal{U}} -\log \frac{\exp(sim(z_{u,1}, z_{u,2})/\tau)}{\exp(sim(z_{u,1}, z_{u,2})/\tau) + \sum_{u' \neq u, u' \in \mathcal{U}} sim(z_{u,1}, z_{u',2})/\tau)}, \tag{7}$$

$$\mathcal{L}_{gcl}^{item} = \sum_{i \in \mathcal{I}} -\log \frac{\exp(sim(z_{i,1}, z_{i,2})/\tau)}{\exp(sim(z_{i,1}, z_{i,2})/\tau) + \sum_{i' \neq i, i' \in \mathcal{I}} sim(z_{i,1}, z_{i',2})/\tau)}, \tag{8}$$

where $\mathcal{L}_{gcl}^{user}$ and $\mathcal{L}_{gcl}^{item}$ are the contrastive losses of the user side and item side, respectively. $sim(\cdot, \cdot)$ is the discriminator function, which takes two vectors as the input and then scores the similarity between them. In this work, we set it as cosine similarity function, i.e., $sim(z_1, z_2) =< z_1, z_2 > /(||z_1|| \cdot ||z_2||)$. τ is the temperature to amplify the effect of discrimination.

3.4 Multi-task Training

We leverage a multi-task training strategy to optimize the main recommendation task and the auxiliary contrastive learning task jointly.

$$\mathcal{L} = \mathcal{L}_{main} + \lambda_1 \mathcal{L}_{gcl} + \lambda_2 ||\Theta||_2^2, \tag{9}$$

where Θ is the set of model parameters. λ_1 and λ_2 are hyperparameters to control the strengths of contrastive loss and L_2 regularization, respectively. $\mathcal{L}_{main}$ is the loss function of the main recommendation task. In this work, we adopt Bayesian Personalized Ranking (BPR) loss [19]:

$$\mathcal{L}_{main} = \sum_{(u,i,j) \in O} -\log \sigma \left(\hat{y}_{u,i} - \hat{y}_{u,j} \right), \tag{10}$$

where $\hat{y}_{u,i} = z_u^T z_i$ is the preference score. $\sigma(\cdot)$ is the sigmoid function. $\mathcal{O} = \{(u, i, j) \mid (u, i) \in \mathcal{O}^+, (u, j) \in \mathcal{O}^-\}$ denotes the training data, and $\mathcal{O}^-$ is the unobserved interactions.

4 Experiments

To evaluate the effectiveness of our proposed GCARec, we conduct extensive experiments by answering the following questions:

- **RQ1**: Does GCARec outperform state-of-the-art methods for the top-K recommendation?
- **RQ2**: What are the benefits of our proposed adaptive data augmentation scheme for recommendation performance?
- **RQ3**: Is our model sensitive to hyperparameters? How do different hyperparameters influence the recommendation performance?

4.1 Experimental Setup

Datasets. We adopt three real-world datasets, including MovieLens-1M[1], Retailrocket[2] and Yelp2018 [23]. Detailed statistics of them are summarized in Table 1.

Table 1. Statistics of the datasets.

Dataset	#Users	#Items	#interactions	Density
MovieLens-1M	5,949	2,810	571,531	0.03419
Retailrocket	259,531	86,053	931,549	0.00004
Yelp2018	31,668	38,048	1,561,406	0.00130

- **MovieLens-1M**. This dataset contains the ratings (1-5 stars) of users for movies, which were collected through the MovieLens website.
- **Retailrocket**. This dataset contains the interactions of users on a real-world e-commerce website. The interactions include clicks, adding to carts and transactions with items.
- **Yelp2018**. This dataset is the 2018 edition of the Yelp challenge. It contains businesses, reviews and user data. We view the businesses such as restaurants as items.

Evaluation Metrics. To evaluate all methods, we randomly split the interactions into training, validation, and testing sets with a ratio of 7:1:2. Items that the user has not interacted with are treated as negative items. For each user in the test set, each method produces users' preference scores for all items, excluding the positive items used in the training set. We employ the widely-used $Recall@K$ and $NDCG@K$ as evaluation metrics for the top-K recommendation, where K is set to 2, 6 and 10.

Compared Methods. We compare GCARec with the following methods:

- **POP**. This method recommends items according to item popularity. The popularity of an item is the number of its interactions. This is a non-personalized method but is still adopted in some scenarios.
- **BPR** [19]. This is a matrix factorization method. It is optimized by Bayesian personalized ranking (BPR) loss to make the preference score of positive items higher than negative items.
- **NGCF** [23]. This is a graph-based recommendation method, which incorporates the second-order neighbors.

[1] https://grouplens.org/datasets/movielens/
[2] https://www.kaggle.com/retailrocket/ecommerce-dataset

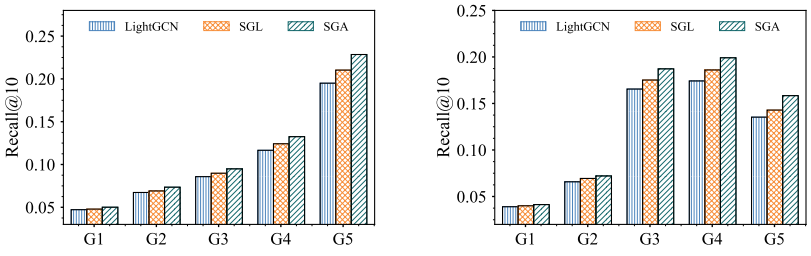
Table 2. Performance comparison of all compared methods on three datasets. The best results are bolded and the best results of baselines are underlined.* indicates the significance level p-value$<$0.01 compared with the best baseline.

Dataset	Metric	POP	BPRMF	NGCF	LightGCN	SGL	GCARec	Improve.
MovieLens-1M	Recall@2	0.0212	0.0404	0.0420	0.0452	0.0487	**0.0517***	+6.16%
	NDCG@2	0.1540	0.2585	0.2663	0.2796	0.2938	**0.3080***	+4.83%
	Recall@6	0.0518	0.0989	0.1004	0.1104	0.1170	**0.1238***	+5.81%
	NDCG@6	0.1418	0.2358	0.2422	0.2565	0.2675	**0.2799***	+4.64%
	Recall@10	0.0781	0.1446	0.1464	0.1587	0.1689	**0.1772***	+6.43%
	NDCG@10	0.1388	0.2310	0.2342	0.2520	0.2600	**0.2711***	+3.17%
Retailrocket	Recall@2	0.0018	0.0210	0.0471	0.0650	0.0782	**0.0861***	+10.10%
	NDCG@2	0.0017	0.0393	0.0442	0.0600	0.0732	**0.0800***	+9.29%
	Recall@6	0.0050	0.0631	0.0635	0.1352	0.1616	**0.1742***	+7.78%
	NDCG@6	0.0031	0.0615	0.0635	0.0908	0.1103	**0.1182***	+7.16%
	Recall@10	0.0071	0.1253	0.1277	0.1830	0.2131	**0.2272***	+6.63%
	NDCG@10	0.0038	0.0730	0.0754	0.1056	0.1288	**0.1358***	+5.43%
Yelp2018	Recall@2	0.0026	0.0099	0.0105	0.0130	0.0150	**0.0158***	+5.33%
	NDCG@2	0.0112	0.0463	0.0512	0.0620	0.0707	**0.0742***	+4.95%
	Recall@6	0.0063	0.0245	0.0256	0.0318	0.0366	**0.0377***	+3.01%
	NDCG@6	0.0104	0.0413	0.0444	0.0545	0.0622	**0.0645***	+3.70%
	Recall@10	0.0095	0.0374	0.0383	0.4660	0.0535	**0.0548***	+2.43%
	NDCG@10	0.0109	0.0439	0.0465	0.0564	0.0647	**0.0665***	+2.78%

- **LightGCN** [10]. This is a graph-based recommendation method, which simplifies the design of GCN. It discards the feature transformation and nonlinear activation in GCN and only uses the neighbor aggregation.
- **SGL** [25].This is a state-of-the-art graph-based recommendation method using graph contrastive learning. It designs three data augmentation methods, including node dropout, edge dropout and random walk. In our experiments, we adopt the edge dropout.

Parameter settings All models are trained from scratch and optimized by the Adam optimizer. The learning rate is fixed to 0.001 and the batch size is 1024. Parameters are initialized by Xavier initializer [6]. The early stopping strategy is adopted, i.e., models are stopped if the *Recall*@10 on the validation data does not increase for 50 consecutive epochs. The embedding size is fixed to 64 for all models. For NGCF[3], LightGCN[4] and SGL[5], we use the implementation provided by their authors on Github. For all baselines, we tune the parameters and report the best performance. We tune λ_1, λ_2, τ and p within the ranges of $\{0, 0.1, 0.2, \cdots, 0.5\}$, $\{0.005, 0.01, 0.05, 0.1, 0.5, 1.0\}$, $\{0.1, 0.2, 0.5, 1.0\}$ and $\{0, 0.1, 0.2, \cdots, 0.5\}$, respectively. The temperature τ_1 in Gumbel Softmax

[3] https://github.com/xiangwang1223/neural_graph_collaborative_filtering
[4] https://github.com/kuandeng/LightGCN
[5] https://github.com/wujcan/SGL



(a) MovieLens-1M (b) Retailrocket

Fig. 2. Performance comparison over different user groups.

is initialized to 10. Following [11], we adopt the following annealing schedule:

$$\tau_1 \leftarrow \max(0.3, \exp(-rt)), \tag{11}$$

where t is the global training step. r is the decay rate and set to 10^{-4}. In addition, τ_1 is updated every 500 batches. The code is released at https://github.com/myjing/GCARec.

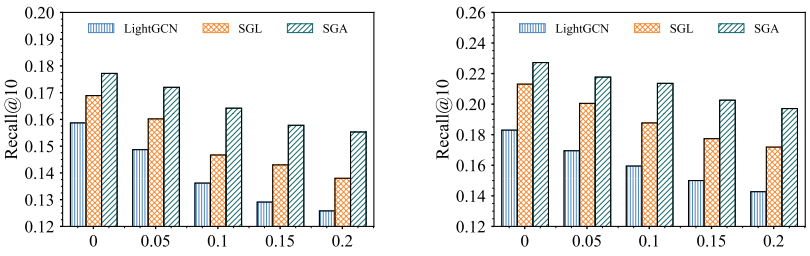
4.2 Performance Comparison (RQ1)

The performance comparison of our proposed GCARec and compared methods on three datasets is shown in Table 2. From this, we can find that: (1) Graph-based methods achieve better performance compared with conventional methods, i.e., POP and BPR. This shows that incorporating the high-order neighbors helps to improve recommendation performance. LightGCN outperforms NGCF on all datasets, demonstrating the effectiveness of removing the feature transformation and nonlinear activation. (2) SGL and GCARec consistently perform better than other baselines, which shows the effectiveness of graph contrastive learning. (3) Overall, GCARec consistently outperforms other baselines on all datasets. Compared with SGL, it shows the effectiveness of the adaptive data augmentation than the uniform dropout.

4.3 Further Study of GCARec

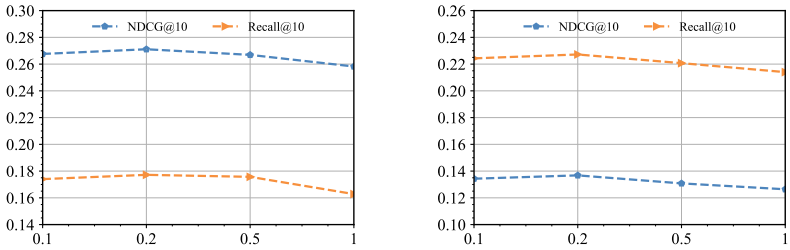
In this section, we first study the benefits of GCARec from two aspects: (1) data sparsity and (2) robustness to noises. Then, we investigate the effect of hyperparameters including τ, λ_1 and p. Due to the space limitation, we only report the results on MovieLens-1M and Retailrocket, while having similar observations in Yelp2018.

Effect of Data sparsity (RQ2). To verify the effectiveness of GCARec to solve the data sparsity problem, we divide users into five groups according to



(a) MovieLens-1M (b) Retailrocket

Fig. 3. Performance *w.r.t.* noise ratio.

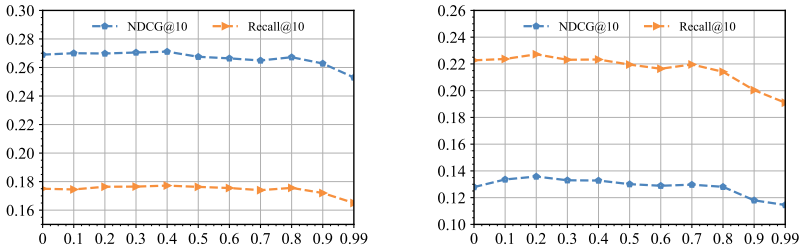


(a) MovieLens-1M (b) Retailrocket

Fig. 4. Performance *w.r.t.* different τ.

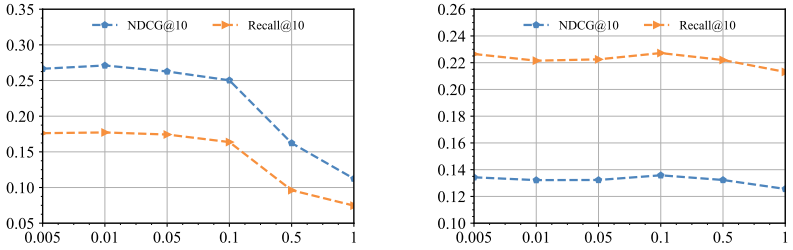
their interaction numbers and make the total number of interactions in each group the same. The larger the GroupID, the larger the average number of user interactions, i.e., the lower the sparsity level. Figure 2 shows the results of *Recall*@10 on these five groups. From it, we can see that our GCARec consistently outperforms LightGCN and SGL, showing its effectiveness in solving the problem of data sparsity. In addition, as the sparse lever increases, the performance improvement from GCARec increases. This shows that the adaptive data augmentation scheme facilitates GCARec to make recommendations on sparse data.

Robustness to Noisy Interactions(RQ2). To further verify the robustness of GCARec against noisy interactions, we add 5%, 10%, 15% and 20% adversarial examples (negative interactions) to the training set. The testing set is kept unchanged. The results are shown in Figure 3. We can observe that adding noise data degrades the performance of all models, but the degradation of the performance of GCARec is smaller than that of GCARec and LightGCN. Moreover, the larger the ratio of noise data, the larger the performance degradation gap



(a) MovieLens-1M (b) Retailrocket

Fig. 5. Performance *w.r.t.* different p.



(a) MovieLens-1M (b) Retailrocket

Fig. 6. Performance *w.r.t.* λ_1.

between GCARec and LightGCN. This shows that adaptive data augmentation schemes can identify inherent patterns in the graph structure more effectively.

Parameter Sensitivity Analysis (RQ3). There are three important hyper-parameters used in GCARec: (1) τ defined in Eq. 7 and Eq. 8; (2) p which determines the generation of graph views; (3) λ_1 which controls the strength of contrastive learning loss. To analyze the parameter sensitivity of GCARec, we select representative values for them. When investigating the effect of τ, we fix $p = 0.4, \lambda = 0.01$ on MovieLens-1M and $p = 0.2, \lambda = 0.1$ on Retailrocket. When investigating the effect of p, we fix $\lambda = 0.01$ on MovieLens-1M, $\lambda = 0.1$ on Retailrocket, $\tau = 0.2$ on both datasets. When investigating the effect of λ_1, the settings of p and τ are as the same as those in the previous cases. The results are shown in Figure 4, Figure 5 and Figure 6.

From Figure 4, we can observe that GCARec is sensitive to τ. Too large (e.g., 1.0) or too small a value (e.g., 0.1) of τ will reduce the performance of the model. This is consistent with the experimental results in [25]. From Figure 5, it can be found that the performance of GCARec is relatively stable when p is not too large. Therefore, overall, our model is not sensitive to p. When $p > 0.9$, almost all

edges of the graph will be dropped, resulting in isolated nodes in the augmented graph views. In this case, it is difficult to aggregate useful information from neighbors. As a result, the representations learned in generated views are not sufficiently distinctive, and this will make it difficult to optimize the contrastive learning objective. From Figure 6, it can be seen that GCARec is sensitive to λ_1. We need to choose λ_1 carefully for different datasets. Moreover, a small value of λ_1 can lead to desirable performance, while a too large value of λ_1 can lead to huge performance degradation.

5 Related Work

In this section, we summarize the related works in two research lines: graph-based recommendation and self-supervised learning in recommender systems.

5.1 Graph-based Recommendation

Graph neural networks (GNNs) have gained considerable attention in recommender systems due to their effectiveness in handling structural data and exploring structural information. In particular, GCN, which propagates user and item embedding over the user-item interaction graph, has driven numerous graph-based recommendation models, such as NGCF [23] and LightGCN [10]. Recently, attention mechanisms are introduced into GCN-based recommendation models [4], which learn different weights to different neighbors, to model the importance of the different neighbors to represent the node. In addition to models that only utilize the user-item interaction graph, some graph-based recommendation models use other graphs like DHCN [29] over the session-based graph and DiffNet [26] over the social network. Knowledge graph has also attracted a surge of attention recently [22].

All of these works focus on supervised settings for model training. However, supervised learning relies heavily on expensive labeled data, making them usually suffer from the problem of sparse supervision signal and noisy interactions. Our work explores graph contrastive learning for solving these problems.

5.2 Self-supervised Learning in Recommender Systems

Self-supervised learning [14] is a new paradigm to learn prior knowledge from unlabeled data. It was firstly used in the field of computer vision [3, 9, 35] and natural language processing [5, 12] for tasks like image classification and text classification. Inspired by the success of these works, SSL has been applied in graph representation learning recently [8, 16, 18, 21]. In this line of research, graph contrastive learning is the dominant method, which maximizes agreement between multiple views that are generated from the raw graph through data augmentation.

Inspired by the success of graph contrastive learning, several works apply self-supervised learning in recommendation. S^3-Rec [36] proposes four optimization objectives to learn attributes, item, subsequence and subsequence correlations. DNN-SSL [31] adopts a two-tower DNN architecture using uniform feature masking and dropout. SGL [25] utilizes uniform node/edge dropout and random walk on user-item interaction graph and applies contrastive learning to the graph-based recommendation. NCL [13] incorporates structural neighbors and semantic neighbors into contrastive pairs. It should be noted that our research focus is different from that of NCL. NCL focuses on sampling strategies, while we focus on data augmentation schemes. They are separate components of contrastive learning. In addition, some studies apply self-supervised learning in other recommendation scenarios such as social recommendation [33].

In this work, we focus on applying graph contrastive learning in the graph-based recommendation method. Compared with SGL, we propose an adaptive and flexible data augmentation scheme that tends to keep the important structures and perturb possibly unimportant edges in the graph. This adaptive augmentation is helpful for the model to preserve the fundamental structure pattern in the graph.

6 Conclusion and Future Work

In this work, we propose a graph contrastive learning for recommendation with adaptive data augmentation. In particular, we propose a learnable and adaptive data augmentation scheme, which tends to retain the important structures and perturb possibly unimportant edges in the user-item interaction graph while generaling easily to other graph-based recommendation. To make the data augmentation scheme learnable, we design it based on the neural network. In specific, we first identify the importance of each edge based on the attention mechanism and calculate the preserving probability based on the importance. Then, edges are sampled according to the importance with a Gumbel Softmax. Extensive experiments on three public datasets demonstrate the effectiveness of our proposed GCARec.

In future work, we will explore other scenarios such as sequential recommendation to apply our work. Besides, we will consider the other part of contrastive learning, i.e., neighbor sampling, to improve the recommendation performance.

Acknowledgements This research is supported in part by the 2030 National Key AI Program of China 2018AAA0100503, National Science Foundation of China (No. 62072304, No. 61772341, No. 61832013), Shanghai Municipal Science and Technology Commission (No. 19510760500, No. 21511104700, No. 19511120300), the Oceanic Interdisciplinary Program of Shanghai Jiao Tong University (No. SL2020MS032), Scientific Research Fund of Second Institute of Oceanography, the open fund of State Key Laboratory of Satellite Ocean Environment Dynamics, Second Institute of Oceanography, MNR, GE China, and Zhejiang Aoxin Co. Ltd.

References

1. Bayer, I., He, X., Kanagal, B., Rendle, S.: A generic coordinate descent framework for learning from implicit feedback. In: WWW. pp. 1341–1350 (2017)
2. Berg, R.v.d., Kipf, T.N., Welling, M.: Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263 (2017)
3. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: ICML. pp. 1597–1607 (2020)
4. Chen, W., Gu, Y., Ren, Z., He, X., Xie, H., Guo, T., Yin, D., Zhang, Y.: Semi-supervised user profiling with heterogeneous graph attention networks. In: IJCAI. vol. 19, pp. 2116–2122 (2019)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
6. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS. pp. 249–256 (2010)
7. Gutmann, M., Hyvärinen, A.: Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In: AISTATS. pp. 297–304 (2010)
8. Hassani, K., Khasahmadi, A.H.: Contrastive multi-view representation learning on graphs. In: ICML. pp. 4116–4126. PMLR (2020)
9. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: CVPR. pp. 9729–9738 (2020)
10. 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 (2020)
11. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144 (2016)
12. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942 (2019)
13. Lin, Z., Tian, C., Hou, Y., Zhao, W.X.: Improving graph collaborative filtering with neighborhood-enriched contrastive learning. arXiv preprint arXiv:2202.06200 (2022)
14. Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., Tang, J.: Self-supervised learning: Generative or contrastive. TKDE (2021)
15. Long, X., Huang, C., Xu, Y., Xu, H., Dai, P., Xia, L., Bo, L.: Social recommendation with self-supervised metagraph informax network. In: CIKM. pp. 1160–1169 (2021)
16. Peng, Z., Huang, W., Luo, M., Zheng, Q., Rong, Y., Xu, T., Huang, J.: Graph representation learning via graphical mutual information maximization. In: Proceedings of The Web Conference 2020. pp. 259–270 (2020)
17. Qin, Y., Wang, P., Li, C.: The world is binary: Contrastive learning for denoising next basket recommendation. In: SIGIR. pp. 859–868 (2021)
18. Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., Tang, J.: Gcc: Graph contrastive coding for graph neural network pre-training. In: SIGKDD. pp. 1150–1160 (2020)
19. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618 (2012)
20. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. stat **1050**, 20 (2017)

21. Velickovic, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. ICLR (Poster) **2**(3), 4 (2019)
22. Wang, X., He, X., Cao, Y., Liu, M., Chua, T.S.: Kgat: Knowledge graph attention network for recommendation. In: SIGKDD. pp. 950–958 (2019)
23. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: SIGIR. pp. 165–174 (2019)
24. Wang, X., Jin, H., Zhang, A., He, X., Xu, T., Chua, T.S.: Disentangled graph collaborative filtering. In: SIGIR. pp. 1001–1010 (2020)
25. Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., Xie, X.: Self-supervised graph learning for recommendation. In: SIGIR. pp. 726–735 (2021)
26. Wu, L., Sun, P., Fu, Y., Hong, R., Wang, X., Wang, M.: A neural influence diffusion model for social recommendation. In: SIGIR. pp. 235–244 (2019)
27. Wu, M., Zhuang, C., Mosse, M., Yamins, D., Goodman, N.: On mutual information in contrastive learning for visual representations. arXiv preprint arXiv:2005.13149 (2020)
28. Wu, Z., Xiong, Y., Yu, S.X., Lin, D.: Unsupervised feature learning via non-parametric instance discrimination. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3733–3742 (2018)
29. Xia, X., Yin, H., Yu, J., Wang, Q., Cui, L., Zhang, X.: Self-supervised hypergraph convolutional networks for session-based recommendation. In: AAAI. vol. 35, pp. 4503–4511 (2021)
30. Xiao, T., Wang, X., Efros, A.A., Darrell, T.: What should not be contrastive in contrastive learning. arXiv preprint arXiv:2008.05659 (2020)
31. Yao, T., Yi, X., Cheng, D.Z., Yu, F., Chen, T., Menon, A., Hong, L., Chi, E.H., Tjoa, S., Kang, J., et al.: Self-supervised learning for large-scale item recommendations. In: CIKM. pp. 4321–4330 (2021)
32. You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., Shen, Y.: Graph contrastive learning with augmentations. In: NIPS. vol. 33, pp. 5812–5823 (2020)
33. Yu, J., Yin, H., Gao, M., Xia, X., Zhang, X., Viet Hung, N.Q.: Socially-aware self-supervised tri-training for recommendation. In: SIGKDD. pp. 2084–2092 (2021)
34. Zang, T., Zhu, Y., Liu, H., Zhang, R., Yu, J.: A survey on cross-domain recommendation: taxonomies, methods, and future directions. arXiv preprint arXiv:2108.03357 (2021)
35. Zbontar, J., Jing, L., Misra, I., LeCun, Y., Deny, S.: Barlow twins: Self-supervised learning via redundancy reduction. In: ICML. pp. 12310–12320. PMLR (2021)
36. Zhou, K., Wang, H., Zhao, W.X., Zhu, Y., Wang, S., Zhang, F., Wang, Z., Wen, J.R.: S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In: CIKM. pp. 1893–1902 (2020)