Decoupled Behavior-based Contrastive Recommendation

Mengduo Yang
Jie Zhou
yangmd2118@mails.jlu.edu.cn
22251247@zju.edu.cn
Zhejiang University
School of Software Technology
Ningbo, Zhejiang, China

Meng Xi*
Xiaohua Pan
ximeng@zju.edu.cn
panxh@zju-bj.com
Zhejiang University
School of Software Technology
Ningbo, Zhejiang, China
Binjiang Institute of Zhejiang University
Hangzhou, Zhejiang, China

Yi Yuan
icecens@zju.edu.cn
Zhejiang University
School of Software Technology
Ningbo, Zhejiang, China

Ying Li
Yangyang Wu
cnliying@zju.edu.cn
zjuwuyy@zju.edu.cn
Zhejiang University
School of Software Technology
Ningbo, Zhejiang, China
Binjiang Institute of Zhejiang University
Hangzhou, Zhejiang, China

Jinshan Zhang
zhangjinshan@zju.edu.cn
Zhejiang University
School of Software Technology
Ningbo, Zhejiang, China

Jianwei Yin
zjuyjw@zju.edu.cn
Zhejiang University
School of Software Technology
Ningbo, Zhejiang, China
Binjiang Institute of Zhejiang University
Hangzhou, Zhejiang, China

Abstract

Recommender systems are crucial tools in online applications, assisting users in discovering relevant content efficiently. Recent studies demonstrate that contrastive learning (CL) based methods yield significant results in collaborative filtering (CF) recommendations, due to their ability to address the issue of data sparsity. However, two inherent limitations remain unexplored in these methods. a) Since the datasets commonly used are binary (0: no interaction; 1: interaction), current methods only provide rudimentary modeling of user behaviors in binary form, which fails to model complex user-item interactions and relationships in real-world recommendation scenarios. b) Existing CL-based methods mostly construct contrastive views through heuristic-based embedding or structure perturbation, which are prone to introduce noise or discard important information, leading to a decreased representation quality. To address these issues, we propose a Decoupled Behavior-based Contrastive Recommendation model (DBCR) that effectively decouples user behaviors from binary datasets for better user-item interaction modeling. The core idea is to decouple latent user behaviors from unlabelled user-item interactions (binary datasets) and utilize self-supervised contrastive learning to optimize CF-based recommendation jointly. Specifically, we introduce latent behavior variables and embed them into user-item interaction modeling within the generalized expectation-maximization (EM) framework. Moreover, we design a contrastive learning task by constructing a preference view instead of unreasonable perturbation to further improve the learned representation. Experimental results and analyses on three real-world datasets demonstrate the effectiveness of DBCR and its high efficiency, with an average improvement of 16.9% over state-of-the-art methods. Our code is available on https://github.com/Du-danger/DBCR.

CCS Concepts

• **Information systems** → **Recommender systems**.

Keywords

Recommendation, Collaborative Filtering, Behavior Decoupling, Contrastive Learning

ACM Reference Format:
Mengduo Yang, Jie Zhou, Meng Xi, Xiaohua Pan, Yi Yuan, Ying Li, Yangyang Wu, Jinshan Zhang, and Jianwei Yin. 2024. Decoupled Behavior-based Contrastive Recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3627673.3679636

*Meng Xi is the corresponding author.

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679636

1 Introduction

Recently, recommender systems (RS) have emerged as essential tools in web applications, aiding users in swiftly discovering relevant content from a vast amount of online information. These

systems provide personalized recommendations based on user interests, such as recommending items on social networking sites [9, 36], short video platforms [25, 35] and shopping apps [17, 27].

Collaborative Filtering (CF) is one of the most commonly employed methods in recommendation, wherein it utilizes the preferences of similar users to recommend new items to a given user [1, 26, 29]. To implement CF-based recommendation models, one commonly utilized strategy is to reconstruct historical interactions by assigning embeddings to both users and items based on matrix factorization (MF) [10]. To enhance and generalize MF-based methods, some approaches (e.g., NCF, AutoR, DMF) have extended the matrix factorization by introducing neural networks, which improve the quality of user and item embeddings [7, 8, 31].

Recently, with the rise of graph neural network (GNN), researchers have focused on utilizing GNN to propagate embeddings on user-item interaction graphs and learn more effective user/item representations [3, 21]. NGCF [18] and LightGCN [6] are representative examples of GNN-based CF models that have shown promising results in personalized recommendations. These models take advantage of GNN to propagate embeddings on user-item interaction graph, enabling them to capture intricate interdependencies between users and items that traditional CF models might overlook. However, existing GNN-based methods are often faced with the challenges of data sparsity. To address this problem, contrastive learning (CL)-based methods have been proposed, which have great potential in the field of personalized recommendation [15, 34]. The core of this approach is to incorporate self-supervised contrastive learning by maximizing (minimizing) the mutual information of positive (negative) pairs in different views, thus obtaining a more robust and effective representation. Based on this paradigm, some methods have been proposed, such as SGL [26], NCL [12], SimGCL [34] (XSimGCL [33]) have shown satisfactory results and achieved state-of-the-art performance in CF-based recommendation.

Although CL-based methods have achieved state-of-the-art performance in CF-based recommendation, they still have some inherent limitations. **(1)** Primarily, since the datasets commonly used in CF-based recommendation are binary, these methods typically model user behaviors as either 0 (no interaction) or 1 (interaction), which provides rudimentary modeling of user-item interaction and fails to explore the diversity of user behaviors ("like", "dislike", "comment", etc.), thus impeding their ability to model complex relationships between users and items in real-world recommendation scenarios. While several classic datasets already exist that encompass multiple types of feedback, they are relatively scarce compared to binary datasets.[1] In addition, in some certain scenarios, user behaviors may not be easily collected. For instance, in some video platforms (VPs), users are often recommended various game content. When users click on the game content, they are redirecting to external pages to initiate downloads or make in-game purchases. During this period, VPs can only collect the click behaviors, while other behaviors remain untracked. This means downloads and other external behaviors rely on the feedback from the gaming platforms to VPs. However, gaming platforms may not prefer to return such behaviors, like downloads or purchases, to reduce the recommendation fees paid to VPs, which results in difficulties in collecting multi-behavioral data and modeling different relationship between users and items. **(2)** Secondly, existing CL-based methods mostly construct contrastive views through heuristics-based structure or embedding perturbation [26, 34], which may introduce noisy information or discard important information in representation learning. For example, when adding edges between users and items, if these edges do not align with user preferences, they will introduce noise and harm the user and item modeling. This may negatively affect representation learning and reduce the model performance.

To overcome the problems mentioned above, we propose a decoupled behavior-based contrastive recommendation model (DBCR), which aims to decouple latent user behaviors from binary interactions for better modeling of users and items within the generalized expectation-maximization (EM) framework. Specifically, in the E-step, we introduce a latent variable to decouple user behaviors into the user-item interaction modeling and learn the distribution function of behavior variables by capturing preference levels between users and items. In the M-step, we employ the learned behaviors to refine user and item representations through behavior contrastive learning, which maximizes the consistency between user-item interaction and the corresponding behavior variables. In addition, to ensure effective and informative contrastive learning, we design a preference-guided contrastive learning task without unreasonable perturbation on structure and embedding, to further improve the learned representation. Finally, we design a multi-task learning strategy to optimize our model for effective representation learning jointly. In summary, our contributions are mainly as follows:

- We propose a decoupled behavior-based contrastive recommendation model (DBCR) that effectively decouples latent user behaviors from binary interactions, which empowers CL-based methods to model complex relationships between users and items in real-world recommendation scenarios.
- We tackle the behavior decoupling problem by introducing a latent variable to represent user behaviors and learn them alternately through an expectation-maximization (EM) framework, which guarantees that the optimization process is convergent.
- We design a contrastive learning task by constructing a preference view instead of unreasonable perturbation on embedding and structure, ensuring an effective contrastive learning process.
- Experiments conducted on three real-world datasets demonstrate that DBCR surpasses state-of-the-art methods. Experimental analysis further validates the effectiveness of the model, indicating its potential as a reference for practitioners in the field.

2 Related Work

2.1 Contrastive Learning for Recommendation

Recently, contrastive learning (CL) has attracted considerable attention in recommendation due to its ability to address the issue of data sparsity [4, 24, 37, 39]. For example, SimpleX [14] designs a cosine similarity contrastive learning task considering first-order neighbor nodes. SGL [26] employs heuristic-based data augmentation methods to generate contrastive views by perturbing nodes/edges. GCA [38] presents joint adaptive data augmentation at structure and attribute levels by removing edges and masking features. However, random perturbations on the graph structure are prone to introduce noise or discard important information. Therefore, CGI

[1] https://github.com/RUCAIBox/RecSysDatasets

[23] selectively drops edges/nodes to construct more practical contrastive views. NCL [12] considers similar semantic prototypes in the embedding space and structural neighbors as positive views. HCCF [29] captures local and global synergies through cross-view contrastive learning enhanced by hyper-graphs. SimGCL [34] experimentally determines that the contribution of data augmentation to SGL is minimal, thus proposing the addition of noise to each layer embedding to generate positive views. To avoid manual construction of contrastive instances, SimGRACE [28] uses different graph encoders as view generators and compares the semantic similarity between the obtained instances. Based on SimGCL, XSimGCL [33] discards the ineffective augmentations and instead employs a noise-based embedding augmentation for CL, with greatly reducing the complexity of GCL.

However, these methods typically model user behaviors as 0 (no interaction) or 1 (interaction) due to the limitation of binary datasets, thus impeding their ability to model complex relationships between users and items in real-world recommendation scenarios. In addition, these methods mostly construct contrastive views based on heuristic embedding or structure perturbation, which may introduce noisy information in representation learning.

2.2 Decoupled Factor-based Recommendation

The pursuit of decoupling latent factors from implicit feedback has become increasingly prominent in recent scholarship. DGCF [20] advances this concept by performing disentangled representation learning via a graph neural network that utilizes an embedding partition strategy. To integrate auxiliary information from the domains of users or items into the recommendation process, DisenHAN [22] seeks to distill disentangled representations for users and items using a heterogeneous graph attention framework. Meanwhile, KGIN [19] is trying to capture latent user intents by employing an item knowledge graph, with the aim of improving the efficacy of recommendations. DCF [5] bifurcates users and items into factor-level representations and employs a factor-level attention scheme to discern underlying intents. Moreover, CDR [2] innovates with a dynamic routing algorithm designed to elucidate correlations among user intentions, thus helping to refine embeddings. However, existing decoupled methods are predicated on fully supervised learning paradigms, which may be constrained by the dearth of dense user-item interaction data in real-world settings. To overcome this impediment, DCCF [15] introduces a novel paradigm that harnesses the power of self-supervised learning to facilitate intent-aware data augmentation.

However, the above methods often disentangle user intents from user-item interactions, while ignoring the diversity of user behaviors. Our method first proposes to use the Expectation-Maximization (EM) algorithm to alternately decouple user behaviors through E-steps and M-steps, with the goal of eventually decoupling latent user behaviors from binary user-item interaction.

3 PRELIMINARIES

3.1 Problem Formulation

In CF-based recommendation, we define $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ and $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ as the sets of users and items, respectively, where M, N are the total number of users and items. Users and items are associated with trainable embedding vectors $\mathbf{E}^u \in \mathbb{R}^{M\times d}$ and $\mathbf{E}^v \in \mathbb{R}^{N\times d}$, where d represents the dimensionality of the embeddings. The user-item interaction matrix $\mathbf{A} \in \mathbb{R}^{M\times N}$ denotes the observed interactions between users and items, where an element $\mathbf{A}_{i,j} = 1$ if the user i has interacted with item j, otherwise $\mathbf{A}_{i,j} = 0$. For convenience, we adopt (u_i, v_j) to represent the interaction between user i and item j. Given the items that user i has interacted with, the goal of recommendation is to predict the items that user i is likely to access in the future, which can be formalized as:

$$f_\theta(\mathbf{A}_{i,:}) \Longrightarrow \mathcal{R}_i, \tag{1}$$

where f is the model function with parameters θ, $\mathbf{A}_{i,:}$ implies the interacted items of user i, and $\mathcal{R}_i$ denotes the item set that user i tends to interact with next. Then, the optimization objective of the recommendation task can be formulated as finding the optimal parameter that maximizes the log-likelihood function of the interacted users and items:

$$\theta' = \arg\max_\theta \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log P_\theta(u_i, v_j). \tag{2}$$

3.2 GNN-based Recommendation Backbone

GNN-based methods usually perform L GNN layers on the interaction graph to learn the representation of users and items. The l-th GNN layer is defined as:

$$z^u_{i,l} = \sigma(\tilde{\mathbf{A}}_{i,:}\mathbf{Z}^v_{l-1}), \quad z^v_{j,l} = \sigma((\tilde{\mathbf{A}}_{:,j})^{\mathrm{T}}\mathbf{Z}^u_{l-1}), \tag{3}$$

where $z^u_{i,l}$ and $z^v_{j,l}$ represent output representation for user i and item j in the l-th layer. We initialize $\mathbf{Z}^u_0$ and $\mathbf{Z}^v_0$ with the embedding $\mathbf{E}^u$ and $\mathbf{E}^v$ respectively. $\sigma(\cdot)$ denotes the LeakyReLU activation function. $\tilde{\mathbf{A}}$ is the normalized interaction matrix, where $\tilde{\mathbf{A}}_{i,j} = \mathbf{A}_{i,j}/\sqrt{D^u_i D^v_j}$ ($D^u_i = \sum \mathbf{A}_{i,:}$ and $D^v_j = \sum \mathbf{A}_{:,j}$). Then, we sum the output representation from all GNN layers to obtain the final user (item) representation and employ a dot product to predict the interaction probability $\hat{y}_{i,j}$ between user i and item j as:

$$z^u_i = \sum_{l=0}^{L} z^u_{i,l}, \quad z^v_j = \sum_{l=0}^{L} z^v_{j,l}, \quad \hat{y}_{i,j} = z^u_i \cdot z^v_j. \tag{4}$$

We bring $\hat{y}_{i,j}$ into Eq. 2 to get a specific optimization objective:

$$\theta' = \arg\max_\theta \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log \hat{y}_{i,j}\,, \tag{5}$$

which is equivalent to minimizing the following BPR loss:

$$\mathcal{L}_{bpr} = -\sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log \phi(\hat{y}_{i,j} - \hat{y}_{i,n}), \tag{6}$$

where $\hat{y}_{i,j}$ and $\hat{y}_{i,n}$ denote the predicted scores for a pair of positive and negative items of user i. $\phi(\cdot)$ is the Sigmoid function.

4 Method

The architecture of DBCR within the EM framework is presented in Figure 1. It first conducts GNN-based representation learning to obtain original view representation. Then, it iteratively performs the E-step and M-step to estimate the behavior distribution, obtain a preference view representation, and optimize the model parameters

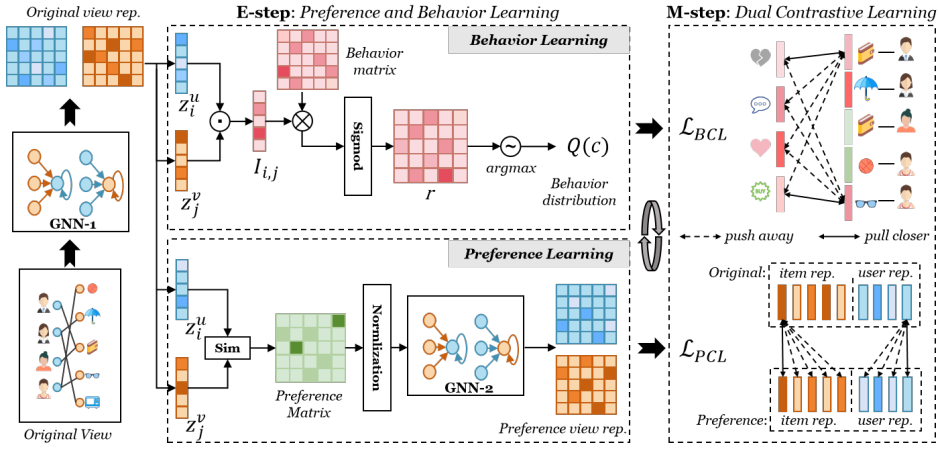

Figure 1: The overview of the proposed DBCR.

θ. During the E-step, it performs behavior and preference learning to estimate the behavior distribution and obtain preference view representation. In the M-step, it optimizes the model parameter θ through behavior contrastive learning (BCL) and preference-guided contrastive learning (PCL). Throughout each iteration, both behavior variables and θ are updated accordingly.

4.1 Latent Behavior Modeling

The main objective of the recommendation task is to optimize Eq. 2. Assume that there are K different user behaviors (e.g., "like", "dislike", "comment", etc.) in a recommender system that forms the behavior variable $C = \{c_k\}_{k=1}^{K}$, then the probability of user i interacting with item j can be rewritten as follows:

$$P_\theta(u_i, v_j) = \mathbb{E}_{(c)}[P_\theta(u_i, v_j, c_k)], \tag{7}$$

where we embed latent behaviors into the user-item interaction modeling. Then, based on Eq. 7, we can rewrite Eq. 2 as follows:

$$\theta' = \arg\max_\theta \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log \mathbb{E}_{(c)}[P_\theta(u_i, v_j, c_k)]. \tag{8}$$

However, optimizing this objective remains a challenge due to the difficulty of integral. To address this, we decouple user-item interaction into K latent user behavior types and adopt a novel approach by constructing a lower-bound function and maximizing it. First, we have:

$$\sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log \mathbb{E}_{(c)}[P_\theta(u_i, v_j, c_k)] = \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log \sum_{k=1}^{K} P_\theta(u_i, v_j, c_k). \tag{9}$$

Since the datasets used are binary, user behaviors are inherently latent. Mathematically, we assume that the behavior variables follow a distribution represented as $Q(c)$, where $\sum Q(c_k) = 1$ and $Q(c_k) \geq 0$. We estimate $Q(c_k)$ in the following E-step and consider it as a constant. Hence, based on the Jensen's inequality, the right term in Eq. 9 can be rewritten as:

$$\begin{aligned}
&= \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log \sum_{k=1}^{K} Q(c_k) \frac{P_\theta(u_i, v_j, c_k)}{Q(c_k)} \\
&\geq \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \sum_{k=1}^{K} Q(c_k) \log \frac{P_\theta(u_i, v_j, c_k)}{Q(c_k)} \\
&\geq \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \sum_{k=1}^{K} \left[Q(c_k) \log P_\theta(u_i, v_j, c_k) - Q(c_k) \log Q(c_k) \right] \\
&\propto \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \sum_{k=1}^{K} Q(c_k) \log P_\theta(u_i, v_j, c_k).
\end{aligned} \tag{10}$$

The inequality will hold with equality when $Q(c_k) = P_\theta(c_k|(u_i, v_j))$. We have found a lower bound of Eq. 9. However, we cannot directly optimize Eq. 10 because $Q(c)$ is unknown. To address this, we innovatively propose a generalized expectation-maximization (EM) approach that iteratively optimizes the objective function and guarantees convergence. The core idea is, to begin with an initial model parameter θ, we estimate the behavior distribution $Q(c)$ in the E-step (Behavior and Preference Learning). Once we have the values of $Q(c)$, we update the parameter θ by maximizing Eq. 9 in M-step (Dual Contrastive Learning). This iterative process is repeated until the likelihood can no longer be increased. Using the EM framework, we can effectively handle the incompleteness caused by the missing behavior distribution $Q(c)$, allowing us to make reliable estimates of both the decoupled behaviors and the model parameters θ.

4.2 E-step: Behavior and Preference Learning

4.2.1 Behavior learning. In this section, we aim to decouple and estimate the behavior type from the user interactions, i.e., the behavior distribution $Q(c)$. We first define the representation of the interaction (u_i, v_j) as follows:

$$\mathcal{I}_{i,j} = z_i^u \odot z_j^v, \tag{11}$$

where $\mathcal{I}_{i,j} \in \mathbb{R}^d$ is the representation of the interaction (u_i, v_j). $z_i^u \in \mathbb{R}^d$ and $z_j^v \in \mathbb{R}^d$ are learned representations of the user i and item

j through Eq. 4, respectively. $\odot$ denotes the element-wise product. Then, we define the learnable vectors $\{\mathcal{B}_k\}_{k=1}^{K}$ ($\mathcal{B}_k \in \mathbb{R}^d$) as the latent behavior embeddings to adaptively learn the relationships between the interacted users and items, where K is the number of latent behaviors. Subsequently, we can obtain the probability of interaction (u_i, v_j) belonging to each behavior type as:

$$r^{i,j} = \phi([\mathcal{I}_{i,j}] \cdot \mathcal{B}^{\mathrm{T}}), \tag{12}$$

where $\mathcal{B} = [\mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_K] \in \mathbb{R}^{K \times d}$ is the behavior vector matrix. $r^{i,j} \in \mathbb{R}^{1 \times K}$ is the behavior probability. For example, $r_{1,k}^{i,j}$ denotes the probability of interaction (u_i, v_j) belonging to the behavior c_k. Then, based on the behavior probability, we can partition the interaction (u_i, v_j) into one of K behavior types as:

$$\mathcal{T}(u_i, v_j) = c_k, \quad \text{if } r_{1,k}^{i,j} = \max(r^{i,j}) \tag{13}$$

where $\mathcal{T}(u_i, v_j)$ denotes the latent behavior type of the interaction (u_i, v_j). Now, we have obtained the behavior types for each interaction in a batch. Naturally, we can obtain the distribution of behaviors $Q(c)$ within the batch, which is expressed as follows:

$$Q(c_k) = P_\theta(c_k|(u_i, v_j)) = \begin{cases} 1, & \text{if } \mathcal{T}(u_i, v_j) = c_k \\ 0, & \text{otherwise} \end{cases}. \tag{14}$$

4.2.2 Preference learning. Through our observation, the user preference level for an item is strongly correlated with the similarity between the user and item representations. That is, the user preference level for one item is higher when the similarity between the user and item representations is greater. Thus, we assume that the preference levels can be measured as the similarity between the representation of users and items to some extent. Specifically, we obtain the preference matrix $\mathcal{P} \in \mathbb{R}^{M \times N}$ between users and items based on the learned representation learned from Eq. 4 as:

$$\mathcal{P}_{i,j} = \begin{cases} \mathrm{sim}(z_i^u, z_j^v), & \text{if } \mathbf{A}_{i,j} = 1 \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

where $\mathrm{sim}(\cdot,\cdot)$ is a similarity function (e.g., dot product), and $\mathcal{P}_{i,j} \in \mathcal{P}$ is the preference level of user i for item j, which can effectively capture the relationship and preference between users and items. Hence, to generate semantic representation for better contrastive learning, we perform the GNN-based representation learning on the preference view, i.e., the preference matrix $\mathcal{P}$. First, we normalize the preference matrix as:

$$\hat{\mathcal{P}} = (\mathbf{D}^p)^{1/2} \mathcal{P} (\mathbf{D}^p)^{1/2}, \tag{16}$$

where $\mathbf{D}^p$ is the diagonal degree matrix of $\mathcal{P}$. Then, we define the GNN-based representation learning on the preference view $\hat{\mathcal{P}}$ as:

$$\begin{aligned} \mathbf{H}_l^u = \sigma(\hat{\mathcal{P}} \mathbf{H}_{l-1}^v), \mathbf{H}_l^v = \sigma((\hat{\mathcal{P}})^{\mathrm{T}} \mathbf{H}_{l-1}^u), \\ \mathbf{H}^u = \sum_{l=0}^{L} \mathbf{H}_l^u, \mathbf{H}^v = \sum_{l=0}^{L} \mathbf{H}_l^v, \end{aligned} \tag{17}$$

where $\mathbf{H}^u$ and $\mathbf{H}^v$ are the learned representations of users and items from the preference view. $h_i^u \in \mathbf{H}^u$ ($h_j^v \in \mathbf{H}^v$) is the representation of user i (item j). We initialize $\mathbf{H}_0^u$ and $\mathbf{H}_0^v$ as $\mathbf{E}^u$ and $\mathbf{E}^v$, respectively.

4.3 M-step: Dual Contrastive Learning

4.3.1 Behavior Contrastive Learning. Through the E-step, we have obtained the interaction representation and estimate the behavior distribution function $Q(c)$. To maximize the objective in Eq. 10, we also need to calculate $P_\theta(u_i, v_j, c_k)$. Here, we assume that the prior distribution over behaviors is uniform, the conditional distribution of (u_i, v_j) given c_k follows isotropic Gaussian with $L2$ normalization to make the final objection easier to solve. Then we can rewrite $P_\theta(u_i, v_j, c_k)$ as follows:

$$\begin{aligned} P_\theta(u_i, v_j, c_k) &= P_\theta(c_k) P_\theta((u_i, v_j)|c_k) = \frac{1}{K} \cdot P_\theta((u_i, v_j)|c_k) \\ &\propto \frac{1}{K} \cdot \frac{\exp(-(\mathcal{I}_{i,j} - \mathcal{B}_k)^2)}{\sum_{k'=1}^{K} \exp(-(\mathcal{I}_{i,j} - \mathcal{B}_{k'})^2)} \\ &\propto \frac{1}{K} \cdot \frac{\exp(\mathcal{I}_{i,j} \cdot \mathcal{B}_k))}{\sum_{k'=1}^{K} \exp(\mathcal{I}_{i,j} \cdot \mathcal{B}_{k'})}, \end{aligned} \tag{18}$$

where $\mathcal{I}_{i,j}$ and $\mathcal{B}_k$ are representations of interaction (u_i, v_j) and behavior c_k, respectively. Based on Eq. 14 and 18, maximizing the lower bound in Eq. 10 is equivalent to minimize the following loss:

$$\mathcal{L}_{bcl} = \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1}^{\mathcal{T}(u_i,v_j)=c_k} -\log \frac{\exp(\mathrm{sim}(\mathcal{I}_{i,j}, \mathcal{B}_k))}{\sum_{k'=1}^{K} \exp(\mathrm{sim}(\mathcal{I}_{i,j}, \mathcal{B}_{k'}))}. \tag{19}$$

We can see that Eq. 19 is similar to general contrastive learning, which aims to maximize the mutual information between two view representations. While Eq. 19 maximizes the mutual information between one interaction and its corresponding behavior type.

4.3.2 Preference-guided Contrastive Learning. Existing CL-based methods often generate contrastive views by perturbing graph structure or embedding. However, this approach carries the risk of introducing harmful noise or discarding crucial information, hindering representation learning. In contrast, we construct a preference view without handcrafted perturbation. It is important to emphasize that, instead of perturbing structure and embedding, our approach only involves a preference-ware representation, which ensures that the CL process is informative. Then, we design a preference-guided contrastive loss by contrasting representations from the original view and the preference view, which is defined as follows:

$$\mathcal{L}_{pcl}^u = \sum_{i=1}^{M} -\log \frac{\exp(\mathrm{sim}(z_i^u, h_i^u)/\tau)}{\sum_{i'=1}^{M} \exp(\mathrm{sim}(z_i^u, h_{i'}^u)/\tau)} \tag{20}$$

where $\mathcal{L}_{pcl}^u$ is the preference-guided contrastive loss for users. z_i^u and h_i^u are the learned representations of user i from the original view and the preference view, respectively. The item-side preference-guided contrastive loss $\mathcal{L}_{pcl}^v$ is defined in the same way.

4.4 Inference and Optimization

Inference. We adopt an inner product to predict the interaction probability $\hat{y}_{i,j}$ between user i and item j as:

$$\hat{y}_{i,j} = z_i^u \cdot z_j^v. \tag{21}$$

Optimization. We employ a multi-task training approach to train our proposed model, which involves optimizing the main recommendation task via Eq. 6, the BCL task via Eq. 19, and the PCL task

via Eq. 20. Specifically, we define the overall loss function as:

$$\begin{aligned} \mathcal{L}_{pcl} =& \mathcal{L}^u_{pcl} + \mathcal{L}^v_{pcl}, \\ \mathcal{L} =& \mathcal{L}_{bpr} + \lambda_1 \mathcal{L}_{bcl} + \lambda_2 \mathcal{L}_{pcl} + \lambda_3 ||\Theta||_2^2, \end{aligned} \tag{22}$$

where Θ denotes the learnable parameter sets. λ_1, λ_2, and λ_3 are the weights for behavior contrastive loss, preference-guided contrastive loss, and regularization loss, respectively.

4.5 Convergence Analysis

To prove the convergence of DBCR under the generalized EM framework, we only need to prove that the value of our log-likelihood function keeps increasing as the number of iterations increases, i.e.,

$$\sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log P_{\theta^{s+1}}(u_i, v_j) \geq \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log P_{\theta^s}(u_i, v_j), \tag{23}$$

where s denotes the iteration number. The optimization goal in the $s+1$ round can be expressed as:

$$L(\theta, \theta^s) = \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \sum_{k=1}^{K} P_{\theta^s}(c_k|(u_i, v_j)) \log P_\theta(u_i, v_j, c_k), \tag{24}$$

where θ^s are the obtained model parameters by optimizing the model in s rounds. Then, we define:

$$H(\theta, \theta^s) = \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \sum_{k=1}^{K} P_{\theta^s}(c_k|(u_i, v_j)) \log P_\theta(c_k|(u_i, v_j)). \tag{25}$$

Subtract the above two formulas $L(\theta, \theta^s)$ and $H(\theta, \theta^s)$, and get:

$$\begin{aligned} & L(\theta, \theta^s) - H(\theta, \theta^s) \\ =& \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \sum_{k=1}^{K} P_{\theta^s}(c_k|(u_i, v_j)) \log \frac{P_\theta(u_i, v_j, c_k)}{P_\theta(c_k|(u_i, v_j))} \\ =& \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \sum_{k=1}^{K} P_{\theta^s}(c_k|(u_i, v_j)) \log P_\theta(u_i, v_j) \\ =& \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log P_\theta(u_i, v_j). \end{aligned} \tag{26}$$

In the final formula obtained from the above formula, we let θ equal to θ^s and θ^{s+1} respectively, and get:

$$\begin{aligned} & \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log P_{\theta^{s+1}}(u_i, v_j) - \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log P_{\theta^s}(u_i, v_j) \\ =& [L(\theta^{s+1}, \theta^s) - L(\theta^s, \theta^s)] - [H(\theta^{s+1}, \theta^s) - H(\theta^s, \theta^s)]. \end{aligned} \tag{27}$$

To prove the convergence of the EM algorithm, we only need to prove that the right side of the above formula is non-negative. Since θ^{s+1} makes $L(\theta, \theta^s)$ maximal, then we have:

$$L(\theta^{s+1}, \theta^s) - L(\theta^s, \theta^s) \geq 0. \tag{28}$$

For the second part, based on the Jensen's inequality, we have:

$$\begin{aligned} & H(\theta^{s+1}, \theta^s) - H(\theta^s, \theta^s) \\ =& \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \sum_{k=1}^{K} P_{\theta^s}(c_k|(u_i, v_j)) \log \frac{P_{\theta^{s+1}}(c_k|(u_i, v_j))}{P_{\theta^s}(c_k|(u_i, v_j))} \\ \leq& \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log \sum_{k=1}^{K} P_{\theta^s}(c_k|(u_i, v_j)) \frac{P_{\theta^{s+1}}(c_k|(u_i, v_j))}{P_{\theta^s}(c_k|(u_i, v_j))} \\ =& \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log \sum_{k=1}^{K} P_{\theta^{s+1}}(c_k|(u_i, v_j)) = 0, \end{aligned} \tag{29}$$

where the final result 0 uses the property that the probability distribution accumulates to 1. Finally, we have:

$$\sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log P_{\theta^{s+1}}(u_i, v_j) - \sum_{i=1}^{M} \sum_{\mathbf{A}_{i,j}=1} \log P_{\theta^s}(u_i, v_j) \geq 0. \tag{30}$$

Thus, the convergence of our method is proved, as shown in Eq. 23.

4.6 Time Complexity Analysis

In this section, we analyze the time complexity of our model during both the pre-processing and the batch training stages. We also compare it with the state-of-art methods as presented in Table 1.

4.6.1 Complexity of the Pre-processing. All the methods need to normalize the adjacency matrix, which has an $O(E)$ time complexity. Note that LightGCL needs to perform Singular Value Decomposition (SVD) to generate its contrastive view, which leads to an extra computational cost of $O(qE)$.

4.6.2 Complexity of the Batch Training.

- *Augmentation.* Our method performs behavior learning and preference view constructing, which leads to an $O(Ed + BKd)$ time complexity. For the other methods, SGL performs two graph augmentations, which is $O(2\rho E)$; DCCF requires distill latent intent factors across all layers, which is $(O(2ELd))$. NCL need perform clustering to get the prototypes, which is $O((M+N)d)$; Both SimGCL and XSimGCL add noise on the learned embeddings, which are also $O((M+N)d)$.
- *GNN.* Our method performs GNN on both original view and preference view, which is an $O(4ELd)$ time complexity. A three-GNN architecture is adopted in both SGL and SimGCL to learn two different augmentations, which is $O(6ELd)$.
- *BPR Loss.* All methods are trained with the BPR loss and each batch contains B interactions, with B random negative interaction, so they have exactly the same time cost in this regard.
- *Contrastive Loss.* Our method contrasts the positive with negative samples for behavior and preference contrastive learning in a batch, which is $O(Bd+B(T+K)d)$ complexity; DCCF contrasts all nodes across all layers per-batch, leading to $O(B(M+N)Ld)$ complexity; NCL need perform K_2-prototype contrastive learning, which brings extra $O(BK_2d)$ complexity.

To sum up, XSimGCL is the current lightweight yet effective method. And compared to all these methods, our DBCR demonstrates significant competitiveness in terms of time complexity.

Table 1: Comparisons of computational complexity with baselines. In the table, E, L and d denote the number of edges, the number of GNN layers, and the embedding size; $\rho \in (0, 1]$ is the edge keep rate in SGL; q, K, K_1 and K_2 are the SVD-required rank in LightGCL, the number of behavior types in DBCR, the number of latent intents in DCCF and the number of prototypes in NCL. M and N represent the number of users and items; B and T are the batch size and node number in a batch.

Stage	LightGCN	SGL	LightGCL	DCCF	NCL	SimGCL	XSimGCL	DBCR
Pre-processing	$O(E)$	$O(E)$	$O((q+1)E)$	$O(E)$	$O(E)$	$O(E)$	$O(E)$	$O(E)$
Augmentation	-	$O(2\rho E)$	-	$O(2ELd)$	$O((M+N)d)$	$O((M+N)d)$	$O((M+N)d)$	$O(BKd + Ed)$
GNN	$O(2ELd)$	$O(2ELd + 4\rho ELd)$	$O(2ELd + 2q(M+N)Ld)$	$O(2ELd + K_1(M+N)Ld)$	$O(2ELd)$	$O(6ELd)$	$O(2ELd)$	$O(4ELd)$
BPR Loss	$O(2Bd)$	$O(2Bd)$	$O(2Bd)$	$O(2Bd)$	$O(2Bd)$	$O(2Bd)$	$O(2Bd)$	$O(2Bd)$
Contrastive Loss	-	$O(Bd + BTd)$	$O((Bd + BTd)L)$	$O(B(M+N)Ld)$	$O(Bd + B(T+K_2)d)$	$O(Bd + BTd)$	$O(Bd + BTd)$	$O(Bd + B(T+K)d)$

5 Evaluation

To verify the effectiveness of DBCR, we try to answer the following research questions:

- **RQ1:** How effective is DBCR to boost recommendation?
- **RQ2:** How efficient is the performance of DBCR?
- **RQ3:** Do the proposed components enhance the performance?
- **RQ4:** How does DBCR perform on the real multi-behavior dataset?
- **RQ5:** How do different hyper-parameter settings affect DBCR?

5.1 Experimental Setup

5.1.1 Datasets. We evaluate all the methods on three real-world datasets [1], including Gowalla, Amazon-book, and Tmall. Detailed statistics of the datasets are presented in Table 2.

Table 2: Description of binary datasets used in evaluations.

Dataset	#Users	#Items	#Interactions	Density
Gowalla	50,821	57,440	1,172,425	4.0e-4
Amazon-book	78,578	77,801	2,240,156	3.7e-4
Tmall	47,939	41,390	2,357,450	1.2e-3

5.1.2 Baselines. To evaluate the proposed DBCR, we compare it with the state-of-the-art baselines presented in four groups:
❶ **General neural methods**: NCF [8] and AutoR [16].
❷ **GNN-based methods**: NGCF [18] and LightGCN [6].
❸ **CL-based methods**: SLRec [32], SGL [26], HCCF [29], NCL [12], LightGCL [1], SimGCL [34] and XSimGCL [33].
❹ **Decoupling-based methods**: DisenGCN [13], DisenHAN [22], CDR [2], DGCF [20], DGCL [11] and DCCF [15].

5.1.3 Implementations and Metrics. Our method is implemented using the PyTorch framework, leveraging the Adam Optimizer for optimization purposes with a learning rate set to 0.001. The hyper-parameters K, λ_1, λ_2, and λ_3 are tuned from the range of [1, 10], [0.1, 0.5], [0.1, 0.5], and [1e-7, 1e-6], respectively. To ensure a fair comparison, we set the initial embedding size d to 32 for all methods. For graph-based models, we select the number of GNN layers L from {1, 2, 3}. We use two commonly used top-K evaluation metrics (Recall@K and NDCG@K) and follow an all-rank approach, where positive items from the test set are ranked together with all un-interacted items for each user [18, 30]. All experiments are conducted on a machine featuring a 32-core CPU, a 24GB NVIDIA 4090 GPU, 125GB memory, and powered by the Linux OS.

5.2 Performance Analysis (RQ1)

To address research question **RQ1**, we present a performance comparison of different recommendation methods in Table 3. The following key observations are made.

❶ ***DBCR consistently outperforms all baselines on all datasets.*** Compared to the state-of-the-art method DCCF, our approach achieves an average improvement of 16.9% on three datasets. We attribute this success to our model's effective decoupling of latent user behaviors from binary interactions through the EM framework: preference and behavior learning (E-step) and dual contrastive learning (M-step). This allows us to capture the intricate diversity of user-item interactions and model complex relationships between users and items in real-world recommendation scenarios. Furthermore, our approach designs a preference-guided contrastive learning task without perturbing structure and embedding, which ensures that the CL process is preference-aware and informative.

❷ ***Compared to Recall@40 (NDCG@40), DBCR achieves a greater improvement in Recall@20 (NDCG@20).*** For example, on the Gowalla dataset, DBCR achieves an improvement of 14.2% in Recall@20, and a 12.2% enhancement in Recall@40. Similarly, it shows an 16.9% increase in NDCG@20 and a 15.4% uplift in NDCG@40. Comparable improvements can be also observed on Amazon-book and Tmall datasets. Overall experiments have shown that our model yields a more notable improvement in Metric@20 than in Metric@40. This suggests that our model is better at ranking items of interest for users within the top 20 recommended results, while the top 40 recommendations may include more items that users are not interested in. Thus, DBCR is more effective in capturing user interests. Overall, our results indicate the effectiveness of our model in accurately ranking user-relevant items within the top 20 recommendations, demonstrating its ability to understand and predict user interests while effectively filtering out unrelated items.

5.3 Efficiency Study (RQ2)

In this section, we analyze the batch training efficiency of our model in comparison to seven competitive baselines, as shown in Figure 2. Specifically, we report the actual training time on the Gowalla dataset and make the following observations.

❶ ***For the convergence epoch and per-batch training time, DBCR is comparable to the state-of-the-art GCL-based methods.*** No-GCL method LightGCN is trained with hundreds of epochs. By contrast, GCL-based methods (SGL, LightGCL, DCCF, NCL, SimGCL and XSimGCL) need the fewest epochs to reach convergence, due to the contrastive learning for accelerating convergence.

Table 3: Comparing various state-of-the-art recommendation baselines on benchmark datasets.

Datasets	Gowalla				Amazon-book				Tmall			
Metrics	Recall@20	Recall@40	NDCG@20	NDCG@40	Recall@20	Recall@40	NDCG@20	NDCG@40	Recall@20	Recall@40	NDCG@20	NDCG@40
NCF	0.1247	0.1910	0.0659	0.0832	0.0468	0.0771	0.0336	0.0438	0.0383	0.0647	0.0252	0.0344
AutoR	0.1409	0.2142	0.0716	0.0905	0.0546	0.0914	0.0354	0.0482	0.0336	0.0611	0.0203	0.0295
NGCF	0.1413	0.2072	0.0813	0.0987	0.0532	0.0866	0.0388	0.0501	0.0420	0.0751	0.0250	0.0365
LightGCN	0.1799	0.2577	0.1053	0.1255	0.0732	0.1148	0.0544	0.0681	0.0555	0.0895	0.0381	0.0499
SLRec	0.1529	0.2200	0.0926	0.1102	0.0544	0.0879	0.0374	0.0490	0.0549	0.0888	0.0375	0.0492
SGL	0.1814	0.2589	0.1065	0.1267	0.0774	0.1204	0.0578	0.0719	0.0574	0.0919	0.0393	0.0513
HCCF	0.1818	0.2601	0.1061	0.1265	0.0824	0.1282	0.0625	0.0776	0.0623	0.0986	0.0425	0.0552
NCL	0.1831	0.2624	0.1089	0.1293	0.0846	0.1318	0.0656	0.0802	0.0647	0.0986	0.0446	0.0568
LightGCL	0.1827	0.2606	0.1082	0.1284	0.0840	0.1285	0.0648	0.0794	0.0633	0.0974	0.0447	0.0565
SimGCL	0.1927	0.2699	0.1139	0.1344	0.0907	0.1365	0.0696	0.0839	0.0680	0.1053	0.0480	0.0609
XSimGCL	0.1933	0.2709	0.1145	0.1350	0.0911	0.1368	0.0701	0.0844	0.0693	0.1072	0.0489	0.0621
DisenGCN	0.1379	0.2003	0.0798	0.0961	0.0481	0.0776	0.0353	0.0451	0.0422	0.0688	0.0285	0.0377
DisenHAN	0.1437	0.2079	0.0829	0.0997	0.0542	0.0865	0.0407	0.0513	0.0416	0.0682	0.0283	0.0376
CDR	0.1364	0.1943	0.0812	0.0963	0.0564	0.0887	0.0419	0.0526	0.520	0.0833	0.0356	0.0465
DGCF	0.1784	0.2515	0.1069	0.1259	0.0688	0.1073	0.0513	0.0640	0.0544	0.0867	0.0372	0.0484
DGCL	0.1793	0.2483	0.1067	0.1247	0.0677	0.1057	0.0506	0.0631	0.0526	0.0845	0.0359	0.0469
DCCF	0.1883	0.2648	0.1127	0.1325	0.0894	0.1347	0.0686	0.0834	0.0672	0.1048	0.0473	0.0602
DBCR	**0.2203**	**0.3039**	**0.1338**	**0.1558**	**0.1118**	**0.1631**	**0.0876**	**0.1041**	**0.0786**	**0.1212**	**0.0558**	**0.0706**
Imprv.	14.2%	12.2%	16.9%	15.4%	22.7%	19.2%	25.0%	23.3%	13.4%	13.1%	14.1%	13.7%

DBCR ranked third in the number of epochs to get converged. Consistent with our analysis on time complexity in section 4.6, our DBCR approach ranks third in batch training time among GCL-based methods, following behind NCL and XSimGCL.

❷ ***The total training time of DBCR is also competitive compared with the existing SOTA methods.*** SGL takes the longest time to finish the computation in a single batch, which is almost several times that of LightGCN on all the datasets. XSimGCL ranks second due to its lightweight contrastive learning architecture. Our DBCR ranks third due to its effective and efficient contrastive learning strategy for better model optimization, which also shows a significant improvement in recommendation performance.

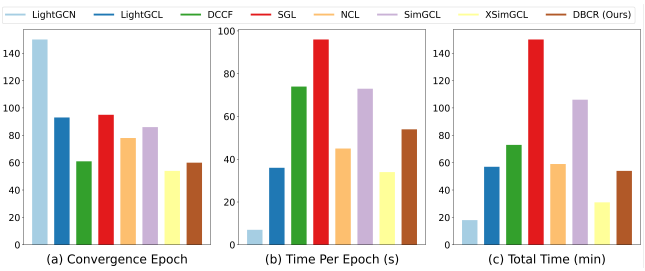

Figure 2: The training epoch and time of different methods.

5.4 Ablation Study (RQ3)

In our model, two essential components contribute to model performance: behavior contrastive learning (BCL) and preference-guided contrastive learning (PCL). To answer research question **RQ2**, we conduct extensive experiments to verify the effect of the components in DBCR. Table 4 reveals the performance of the different variants of DBCR. (A) is our final model, and (B) to (D) are DBCR removed certain components: w/o BCL is the baseline LightGCN with using PCL; w/o PCL is the baseline LightGCN with using BCL; Only BPR is the baseline LightGCN.

❶ ***Behavioral decoupling (BCL) is important for modeling users and items.*** Compared (A) with (B), we find that without the proposed BCL, the performance drops significantly, demonstrating the effectiveness of BCL, which can effectively capture the intricate diversity of user-item interactions and model complex relationships in recommendation scenarios.

❷ ***PCL plays a crucial role in better contrastive learning and representation enhancement.*** Compared (A) with (C), we find that PCL is important for representation learning, which leverages preference view representation to construct a preference-guided contrastive learning task, resulting in a more informative and effective contrastive learning process than current methods.

❸ ***Compared with DBCR, only BPR (i.e., LightGCN) exhibits poor performance.*** When comparing (A) with (D), it becomes evident that relying solely on the main BPR task leads to subpar performance. This observation serves to underscore not only the efficacy of our model but also the critical significance of each individual component within the proposed framework.

Table 4: Performance with different variants (@20).

Model	Metric	Gowalla	Amazon-book	Tmall
(A) DBCR	Recall	**0.2203**	**0.1118**	**0.0786**
	NDCG	**0.1338**	**0.0876**	**0.0558**
(B) w/o BCL	Recall	0.2051	0.0986	0.0664
	NDCG	0.1255	0.0781	0.0502
(B) w/o PCL	Recall	0.1993	0.0943	0.0636
	NDCG	0.1207	0.0751	0.0478
(D) Only BPR	Recall	0.1799	0.0732	0.0555
	NDCG	0.1082	0.0544	0.0381

5.5 Verification of Behavior Decoupling (RQ4)

To validate the effectiveness of behavior decoupling, we conduct additional experiments using a real multi-behavior IJCAI dataset [24], which contains four types of user behaviors: Page View, Favorite, Cart, and Buy. We integrate these four types of behaviors into a unified user-item interaction matrix and conduct experiments on it. The experimental results are as follows.

❶ ***Performance with different*** K**.** Table 5 indicates that the optimal performance is achieved when $K = 4$, which coincides with the number of user behaviors 4 in the IJCAI dataset. It also demonstrates that our model can effectively learn from a multi-behavior dataset, capturing the nuances and distinctions between different types of user-item interactions

Table 5: Performance with different K on IJCAI dataset.

K	1	2	3	**4**	5
NDCG@10	0.166	0.168	0.170	**0.172**	0.171
K	6	7	8	9	10
NDCG@10	0.169	0.167	0.169	0.168	0.168

❷ ***Performance with different methods on the real multi-behavior dataset.*** Table 6 demonstrates the superior performance of the proposed DBCR method compared to the existing state-of-the-art approach on the IJCAI dataset, which is a real multi-behavior dataset. This underscores the effectiveness of our method in addressing such multi-behavior datasets, thereby reinforcing its practical relevance and potential for broader applicability.

Table 6: Performance with various methods on IJCAI dataset.

Methods	LightGCN	SGL	LightGCL	NCL
NDCG@10	0.124	0.123	0.130	0.134
Methods	DCCF	SimGCL	XSimGCL	**DBCR**
NDCG@10	0.143	0.154	0.156	**0.172**

5.6 Hyper-parameter Analysis (RQ5)

To investigate the impact of hyper-parameters on the model performance, we conduct the following experiments.

❶ ***Analysis of the latent behavior number*** K**.** We conduct experiments to explore how many latent behavior variables are effective (i.e., K) as shown in Figure 3(a). We observe that the best number of latent behaviors varies across different datasets. For example, on the Gowalla dataset, the best value of K is 3, while the best value of K on the Amazon dataset is 6. It can be explained by that different recommendation platforms provide users with different services, corresponding to different behaviors, such as purchasing behavior on an e-commerce site, while creating/watching videos on a video-sharing platform. This also demonstrates our model can effectively capture behavior differences in real-world applications and achieve superior performance.

❷ ***Analysis of loss hyper-parameters*** λ_1 ***and*** λ_2**.** As shown in Figure 3(b) and 3(c), we explore the sensitivity of our model with two crucial hyper-parameters: the weights for behavior contrastive loss λ_1 and preference-guided contrastive loss λ_2. We find DBCR reaches its best performance when increasing λ_1 to 0.1, and then it starts to deteriorate as λ_1 becomes larger. λ_2 has the same situation. These findings suggest that contrastive learning as an auxiliary task, the weight of its loss should also be controlled within a reasonable range, so as not to exert excessive influence on the main task. Hence, we select the configuration with $\lambda_1 = 0.1$ and $\lambda_2 = 0.2$ as the optimal choice. In this configuration, the model strikes an effective balance between the main recommendation loss, behavior contrastive loss, and preference-guided contrastive loss, which facilitates a more comprehensive and robust optimization process.

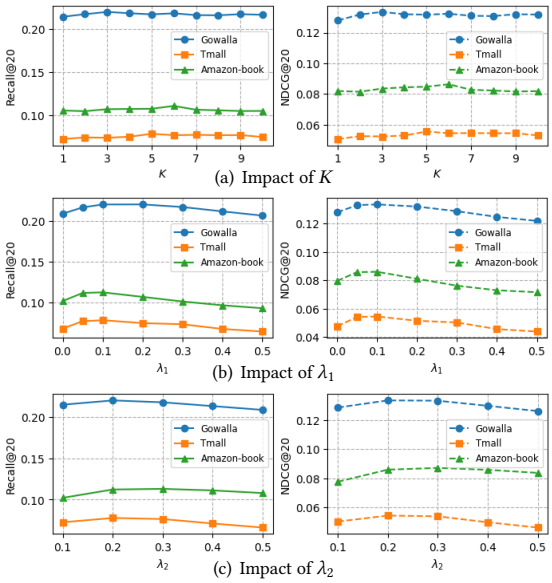

Figure 3: Influence of hyper-parameters on three datasets.

6 Conclusion and Limitation

In this work, an effective recommendation method (DBCR) is proposed, which aims to decouple latent user behaviors from binary interactions for better modeling of users and items. The core of DBCR is to learn user behavior distribution function from binary user-item interactions and utilize self-supervised contrastive learning to optimize the model within the Expectation-Maximization (EM) framework. Additionally, an informative contrastive learning task is designed based on the preference view to further improve the representation. For future work, we aim to address the limitation by extending our model to tackle even more complex and larger-scale recommendation scenarios. Additionally, we plan to incorporate external knowledge for a more comprehensive modeling of latent user behaviors. These enhancements will further improve the performance and effectiveness of our approach.

7 Acknowledgement

This work is supported by the National Key R&D Program of China 2023YFC3306303, the Key R&D Program of Zhejiang Province 2023C01213, and the Regional Innovation and Development Joint Fund of the National Natural Science Foundation of China (U22A6001).

References

[1] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2023. LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

[2] Hong Chen, Yudong Chen, Xin Wang, Ruobing Xie, Rui Wang, Feng Xia, and Wenwu Zhu. 2021. Curriculum Disentangled Recommendation with Noisy Multi-feedback. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. 26924–26936.

[3] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 27–34.

[4] Yongjun Chen, Zhiwei Liu, Jia Li, Julian J. McAuley, and Caiming Xiong. 2022. Intent Contrastive Learning for Sequential Recommendation. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*. ACM, 2172–2182.

[5] Yudong Chen, Xin Wang, Miao Fan, Jizhou Huang, Shengwen Yang, and Wenwu Zhu. 2021. Curriculum Meta-Learning for Next POI Recommendation. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*. ACM, 2692–2702.

[6] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*. ACM, 639–648.

[7] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *IEEE Trans. Knowl. Data Eng.* 30, 12 (2018), 2354–2366.

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW, Perth, Australia, April 3-7, 2017*. ACM, 173–182.

[9] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*. ACM, 135–142.

[10] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[11] Haoyang Li, Xin Wang, Ziwei Zhang, Zehuan Yuan, Hang Li, and Wenwu Zhu. 2021. Disentangled Contrastive Learning on Graphs. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. 21872–21884.

[12] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving Graph Collaborative Filtering with Neighborhood-enriched Contrastive Learning. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*. ACM, 2320–2329.

[13] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 4212–4221.

[14] Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. 2021. SimpleX: A Simple and Strong Baseline for Collaborative Filtering. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*. ACM, 1243–1252.

[15] Xubin Ren, Lianghao Xia, Jiashu Zhao, Dawei Yin, and Chao Huang. 2023. Disentangled Contrastive Collaborative Filtering. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*. ACM, 1137–1146.

[16] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*. ACM, 111–112.

[17] Jianling Wang, Raphael Louca, Diane Hu, Caitlin Cellier, James Caverlee, and Liangjie Hong. 2020. Time to Shop for Valentine's Day: Shopping Occasions and Sequential Recommendation in E-commerce. In *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*. ACM, 645–653.

[18] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*. ACM, 165–174.

[19] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning Intents behind Interactions with Knowledge Graph for Recommendation. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. ACM / IW3C2, 878–887.

[20] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled Graph Collaborative Filtering. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*. ACM, 1001–1010.

[21] Xiao Wang, Ruijia Wang, Chuan Shi, Guojie Song, and Qingyong Li. 2020. Multi-Component Graph Convolutional Collaborative Filtering. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 6267–6274.

[22] Yifan Wang, Suyao Tang, Yuntong Lei, Weiping Song, Sheng Wang, and Ming Zhang. 2020. DisenHAN: Disentangled Heterogeneous Graph Attention Network for Recommendation. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*. ACM, 1605–1614.

[23] Chunyu Wei, Jian Liang, Di Liu, and Fei Wang. 2022. Contrastive Graph Structure Learning via Information Bottleneck for Recommendation. In *NeurIPS*.

[24] Wei Wei, Chao Huang, Lianghao Xia, Yong Xu, Jiashu Zhao, and Dawei Yin. 2022. Contrastive Meta Learning with Behavior Multiplicity for Recommendation. In *WSDM '22: The Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event / Tempe, AZ, USA, February 21 - 25, 2022*. ACM, 1120–1128.

[25] Wei Wei, Chao Huang, Lianghao Xia, and Chuxu Zhang. 2023. Multi-Modal Self-Supervised Learning for Recommendation. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*. ACM, 790–800.

[26] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised Graph Learning for Recommendation. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*. ACM, 726–735.

[27] Liang Wu, Diane Hu, Liangjie Hong, and Huan Liu. 2018. Turning Clicks into Purchases: Revenue Optimization for Product Search in E-Commerce. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*. ACM, 365–374.

[28] Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, and Stan Z. Li. 2022. SimGRACE: A Simple Framework for Graph Contrastive Learning without Data Augmentation. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*. ACM, 1070–1079.

[29] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy X. Huang. 2022. Hypergraph Contrastive Collaborative Filtering. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*. ACM, 70–79.

[30] Lianghao Xia, Chao Huang, and Chuxu Zhang. 2022. Self-Supervised Hypergraph Transformer for Recommender Systems. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*. ACM, 2100–2109.

[31] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. ijcai.org, 3203–3209.

[32] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix X. Yu, Ting Chen, Aditya Krishna Menon, Lichan Hong, Ed H. Chi, Steve Tjoa, Jieqi (Jay) Kang, and Evan Ettinger. 2021. Self-supervised Learning for Large-scale Item Recommendations. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*. ACM, 4321–4330.

[33] Junliang Yu, Xin Xia, Tong Chen, Lizhen Cui, Nguyen Quoc Viet Hung, and Hongzhi Yin. 2024. XSimGCL: Towards Extremely Simple Graph Contrastive Learning for Recommendation. *IEEE Trans. Knowl. Data Eng.* 36, 2 (2024), 913–926. https://doi.org/10.1109/TKDE.2023.3288135

[34] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are Graph Augmentations Necessary?: Simple Graph Contrastive Learning for Recommendation. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*. ACM, 1294–1303.

[35] Ruohan Zhan, Changhua Pei, Qiang Su, Jianfeng Wen, Xueliang Wang, Guanyu Mu, Dong Zheng, Peng Jiang, and Kun Gai. 2022. Deconfounding Duration Bias in Watch-time Prediction for Video Recommendation. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*. ACM, 4472–4481.

[36] Fanjin Zhang, Jie Tang, Xueyi Liu, Zhenyu Hou, Yuxiao Dong, Jing Zhang, Xiao Liu, Ruobing Xie, Kai Zhuang, Xu Zhang, Leyu Lin, and Philip S. Yu. 2022. Understanding WeChat User Preferences and "Wow" Diffusion. *IEEE Trans. Knowl. Data Eng.* 34, 12 (2022), 6033–6046.

[37] Shengyu Zhang, Lingxiao Yang, Dong Yao, Yujie Lu, Fuli Feng, Zhou Zhao, Tat-Seng Chua, and Fei Wu. 2022. Re4: Learning to Re-contrast, Re-attend, Re-construct for Multi-interest Recommendation. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*. ACM, 2216–2226.

[38] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph Contrastive Learning with Adaptive Augmentation. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. ACM / IW3C2, 2069–2080.

[39] Ding Zou, Wei Wei, Ziyang Wang, Xian-Ling Mao, Feida Zhu, Rui Fang, and Dangyang Chen. 2022. Improving Knowledge-aware Recommendation with Multi-level Interactive Contrastive Learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*. ACM, 2817–2826.