World Wide Web (2024) 27:10
https://doi.org/10.1007/s11280-024-01242-x

Invariant representation learning to popularity distribution shift for recommendation

Ming He[1] · Han Zhang[1] · Zihao Zhang[1] · Chang Liu[1]

Received: 25 October 2023 / Revised: 6 December 2023 / Accepted: 19 December 2023 /
Published online: 2 February 2024

Abstract
Recommender systems often suffer from severe performance drops due to popularity distribution shift (PDS), which arises from inconsistencies in item popularity between training and test data. Most existing methods aimed at mitigating PDS focus on reducing popularity bias, but they usually require inaccessible information or rely on implausible assumptions. To solve the above problem, in this work, we propose a novel framework called **I**nvariant **R**epresentation **L**earning (**IRL**) to PDS. Specifically, for simulating diverse popularity environments where popular items and active users become even more popular and active, or conversely, we apply perturbations to the user-item interaction matrix by adjusting the weights of popular items and active users in the matrix, without any prior assumptions or specialized information. In different simulated popularity environments, dissimilarities in the distribution of representations for items and users occur. We further utilize contrastive learning to minimize the dissimilarities among the representations of users and items under different simulated popularity environments, resulting in invariant representations that remain consistent across varying popularity distributions. Extensive experiments on three real-world datasets demonstrate that IRL outperforms state-of-the-art baselines in effectively alleviating PDS for recommendation.

Keywords Recommender systems · Popularity distribution shift · Invariant representation learning · Contrastive learning

This article belongs to the Topical Collection: *Special Issue on Advancing recommendation systems with foundation models*
Guest Editors: Kai Zheng, Renhe Jiang, and Ryosuke Shibasaki

✉ Ming He
heming@bjut.edu.cn

Han Zhang
han1254@emails.bjut.edu.cn

Zihao Zhang
zzh2000@emails.bjut.edu.cn

Chang Liu
liuchang123@emails.bjut.edu.cn

1 Faculty of Information Technology, Beijing University of Technology, Beijing, Pingleyuan 100124, China

1 Introduction

Conventional recommendation models inherently assume that training data (historical interactions) and test data (future interactions) are drawn from the same distribution. However, this assumption often proves to be incorrect in practical recommendation scenarios. The diversity in human behaviors across demographics, regions, and time [1] results in an inconsistency in the popularity distribution between the training and test data, referred to as the Popularity Distribution Shift (PDS).

To illustrate the PDS, we conduct a case study using data from the KuaiRand [2] dataset, as shown in Figure 1. The interaction data is divided into two equal portions based on chronological order, referred to as Data1 and Data2, capturing interactions from two distinct time periods. Leveraging Data1, we compute the popularity of each item, measured by the number of interactions with each item. Subsequently, we categorize the item IDs into four groups based on their popularity, labeled as Head, Mid1, Mid2, and Tail, in descending order of popularity. Next, we compute the average popularity of each group in both Data1 and Data2. The figure highlights a significant shift in item popularity between the two time periods, where initially popular items become less popular, while some of the long-tail items start to gain attention.

PDS can hinder the recommendation models performance in real-world scenarios. During training, these models employ empirical risk minimization techniques [3] to minimize the prediction loss over the training data distribution. Consequently, less attention has been paid to long-tail items and inactive users [4]. This results in a minority of popular items and active users dominating the parameter optimization process in these models, and the model embedding latent space exhibits uneven distribution with bias towards popular items and users [5, 6]. Particularly when utilizing graph convolution operations for feature extraction, where high-degree nodes have a substantial influence on refining nearby neighbors and making them more similar in the representation space [5]. Although this approach enhances prediction accuracy within the training data popularity distribution, it can hinder the optimal performance of user preferences when applied on an online platform.

Researchers are actively investigating strategies to address the PDS issue and enhance the generalization of recommendation models. These strategies include regularization methods [7–10], reweighting techniques [11–13], and causal-embedding approaches [14, 15]. However, these methods share a common constraint in that they require prior knowledge of the target popularity distribution or an assumption of an unbiased uniform distribution in the test data. As a result, these methods can be challenging to implement in real-world scenarios when there is limited prior knowledge available.

Recent research has explored contrastive learning and invariant learning [16–20] to maintain consistent representations despite changes in popularity distribution. However, traditional contrastive learning methods can introduce noise or irrelevant information through data augmentation. Some partially invariant learning approaches assume multiple training data environments [17], which may not align with real-world scenarios with minimal external environmental changes during data collection. An alternative perspective suggests separating bias factors from invariant representations [18, 19], but identifying suitable indicators for these bias factors can be difficult or invalid. For example, [18] applies the popularity of items as a supervisory signal to decouple invariant representations from popularity-related representations. However, the decoupling approach might result in overlooking intrinsic excellent features in items crucial for their popularity. To address these limitations, we propose a novel approach that creates diverse popularity environments by perturbing data directionally. Different from existing work [18] that solely establishes an explicit bias signal and straightforwardly

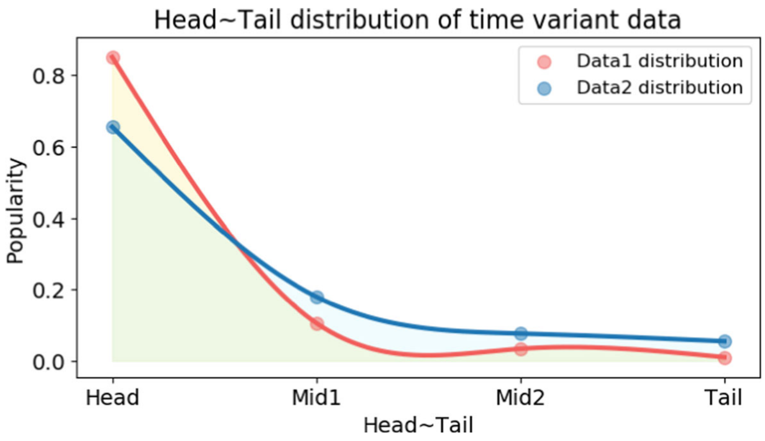

Figure 1 A motivating case from the KuaiRand [2] data illustrates how the distribution of popularity shifts in real-world scenarios

decouples it from the feature representation, we employ contrastive learning to derive stable feature representations resistant to variations in popularity environments. Our approach simplifies popularity bias handling and enhances the stability of representation learning.

In this study, we propose a novel learning framework called **IRL** (**I**nvariant **R**epresentation **L**earning). IRL comprises three key modules: the Matrix Directional Perturbation (MDP) module, the Cross-Environment Contrastive (CEC) module, and the Inter-Environment Constraint (IEC) module. Firstly, the MDP module identifies popular items and active users, referred to as popular nodes, and adjusts their weights when constructing interaction matrices. This leads to the creation of traditional, popularity-enhanced, and popularity-attenuated interaction matrices. Subsequently, convolution operations are applied separately to these matrices, generating node representations for various simulated popularity environments. The CEC module plays a crucial role in enhancing feature consistency across different simulated popularity contexts, ultimately yielding the invariant features we aim for. Finally, the IEC module enforces the convergence of the interaction probability distribution by using distribution constraints. This enhances prediction accuracy and helps mitigate any adverse effects arising from the perturbation of interaction matrices.

The key contributions of this work are as follows:

- We present a novel approach to simulate various popularity environments by applying directional perturbation on the interaction matrix.
- We propose the IRL framework which obtains invariant feature representation through cross-environment contrastive learning and inter-environment interaction distribution constraints.
- We implement IRL on LightGCN [21] and conduct extensive experiments on three real-world datasets, demonstrating its effectiveness.

2 Related work

2.1 Popularity debiasing in recommendation

Popularity bias is the phenomenon where popular items receive recommendations more frequently than expected, a challenge extensively examined in recommender systems research.

Several strategies have emerged to mitigate this bias. In one approach, researchers introduce penalty terms to balance recommendation accuracy and coverage [7–10]. For instance, [9] addresses missing target labels with self-training regularizers, and [7] employs intra-list diversity as a regularization method. Another strategy adjusts the loss of training instances using inverse propensity scores. Recent studies focus on unbiased propensity estimators like [11] and [12] to reduce propensity score variance without relying on observed frequencies. Counterfactual inference techniques also play a role in mitigating item popularity's influence. For example, [14] uses backdoor adjustment to address imbalanced item group distributions, while [15] employs do-calculus to handle confounding popularity bias. However, many debiasing methods assume unrealistic access to popularity information during testing, relying on uniform distribution or prior knowledge of test data. *However, our approach does not require such prior information or unbiased assumptions.*

2.2 Invariant learning

Invariant learning techniques [22–24] assume data heterogeneity across various environments, with the goal of capturing predictive representations that remain consistent in diverse settings. Some methods have expanded on this by relaxing traditional invariance assumptions [25], while others have introduced novel approaches. For example, [26] combines invariant learning principles with the information bottleneck concept, and [27] has developed an effective weighting method to enhance invariance, leading to improved generalization in machine learning tasks. In the realm of recommendation systems, [17] assumes the existence of different environments and leverages the Expectation-Maximization (EM) algorithm to allocate interactions to these environments, mitigating bias. [18] obtains invariant representations by isolating bias factors, and it has demonstrated superior efficacy in mitigating the problem of PDS. Different from [18] that merely employs a decoupling method for separating invariant representations, we adopt a contrastive learning method to bring representations closer in different popularity environments and obtain invariant representations. It's important to know that many of these methods mentioned above often require unbiased uniform data during training or rely on assumptions about the distribution of environments in the training data. Additionally, the identification or definition of a bias factor is also a notably intricate endeavor. Our approach stands out by creating diverse environmental states through data augmentation. *The innovative technique proposed in this work eliminates the need for unbiased data during model training, concurrently avoids making unrealistic environmental assumptions, and discards the need to identify bias factors.*

2.3 Graph contrastive learning for recommendation

A promising line of recent studies has incorporated contrastive learning (CL) into graph-based recommenders, to address the label sparsity issue with self-supervision signals. Particularly, [28] and [29] perform data augmentation over graph structure and embeddings with random dropout operations. However, such stochastic augmentation may drop important information, which may make the sparsity issue of inactive users even worse. Furthermore, some recent alternative CL-based recommenders, such as [30] and [31], design heuristic-based strategies to construct views for embedding contrasting. Cai et al. [32] exclusively utilizes singular value decomposition for contrastive augmentation. However, regardless of the methods used, the data augmentation directions in these models are uncontrollable. *Different from these meth-*

ods, the data augmentation mode of our proposed interaction matrix carries interpretable semantic information, meaning that our data augmentation is controllable.

3 Methodology

3.1 Invariant presentation learning

Due to varying popularity environments, these vectors' latent spaces exhibit different biases [6], i.e., during training, the differences in item popularity and user activity levels can lead to feature representation biases towards popular items or active users in a specific popularity environment. Even worse, the commonly used graph convolution operations tend to amplify such biases [5]. We address the model representation bias by modifying the training process. Our ultimate goal is to diminish the disparities in different latent spaces and obtain stable invariant representations.

To illustrate this process visually, we use Figure 2. As shown, due to the graph convolution operation, feature representations are likely to be biased toward active nodes. Therefore, user preferences P_1, P_2, and P_3, obtained from training data in different popularity environments, deviate from the true ideal preference P. By simulating preferences in various popularity contexts and applying contrastive learning, all preference representations eventually converge to a common point. We term this point the ideal point of invariant preferences. Ultimately, all representations move closer to this ideal point, achieving invariant representation learning.

3.2 Framework

In this section, we provide a detailed technical overview of our proposed APDS. The overall framework is presented in Figure 3. The **Matrix Directional Perturbation (MDP)** module generates traditional interaction matrices from real training data and adjusts the weights for popular items and active users, resulting in enhanced and attenuated matrices. These matrices are assumed to be obtained from simulated environments (as indicated by dashed lines in Figure 3). We set the initial user and item representation embeddings as $\mathbf{E}_u \in \mathbb{R}^{M\times d}$ and

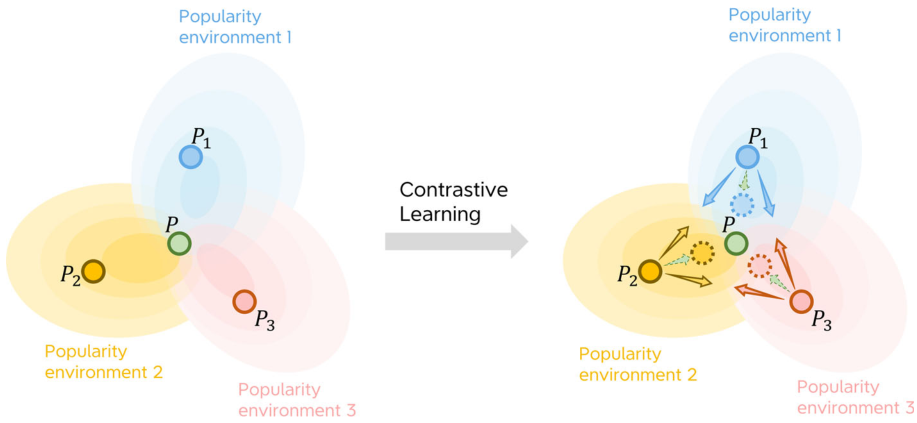

Figure 2 The basic idea of IRL

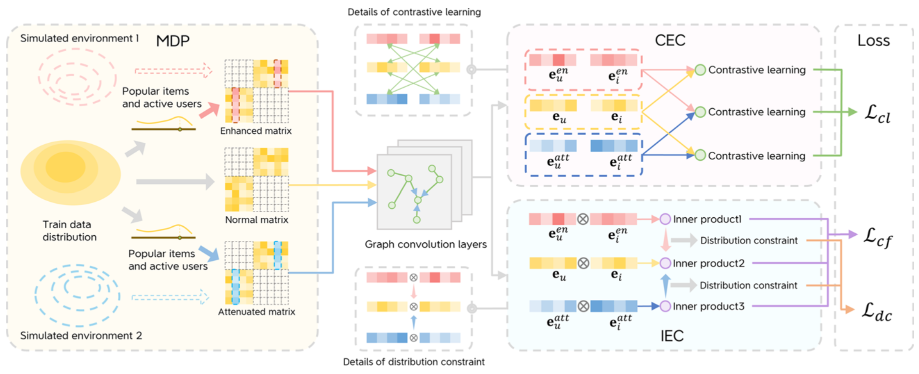

Figure 3 The overall framework of IRL

$\mathbf{E}_i \in \mathbb{R}^{N\times d}$, where M and N represent the number of users and items, and d represents the embedding size. After performing graph convolution operations with different matrices, we obtain normal user and item vectors ($\mathbf{e}_u$ and $\mathbf{e}_i$), enhanced vectors ($\mathbf{e}_u^{en}$ and $\mathbf{e}_i^{en}$), and attenuated vectors ($\mathbf{e}_u^{att}$ and $\mathbf{e}_i^{att}$). These vectors are then used in the **Cross-Environment Contrastive (CEC)** module and **Inter-Environment Constraint (IEC)** module for contrastive learning, interaction probability computation, and distribution constraint.

3.2.1 Matrix directional perturbation

In this module, we start by obtaining the interaction matrix $\mathbf{R} \in \mathbb{R}^{M\times N}$, where M represents user number and N represents item number. Each element at row m and column n represents the interaction number between the m-th user and the n-th item. We then transpose $\mathbf{R}$ to get $\mathbf{R}^T$ and create an $(M+N)\times(M+N)$ square matrix, $\mathbf{A}$, by placing $\mathbf{R}$ in the upper-right and $\mathbf{R}^T$ in the lower-left corners, with all other elements set to zero. Next, we compute the symmetrically normalized matrix $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$. Here, $\mathbf{D}$ is a diagonal matrix with diagonal elements representing the sum of corresponding row elements in matrix $\mathbf{A}$. This normalized matrix is then used as input for the graph convolutional layers of LightGCN.

To perturb the matrix $\mathbf{R}$, we calculate the popularity of both users and items based on their respective interaction counts, sort them in descending order of popularity, and select the top 20% as active users and popular items. Initially, we multiply the row elements of $\mathbf{R}$ corresponding to the columns of popular items by a factor t (where t is a hyperparameter). This operation effectively amplifies all elements in the columns of popular items by t, resulting in matrix $\mathbf{R}'$. Similarly, we amplify the rows corresponding to active users by a factor of t, resulting in matrix $\mathbf{R}''$. Combining $\mathbf{R}'$ and $\mathbf{R}''^T$ forms a new adjacency matrix $\mathbf{A}^{en}$, same as the process of constructing $\mathbf{A}$, referred to as the enhanced matrix. We illustrate this process in the left part of Figure 4. The method for obtaining the attenuated matrix $\mathbf{A}^{att}$ is similar, with the operation of multiplying by t replaced by multiplying by $\frac{1}{t}$. We also show the process in the right part of Figure 4.

3.2.2 Cross-environment contrastive

After obtaining different matrices, we have completed the step of introducing artificial data augmentation to simulate changes in the popularity environment. The different interaction

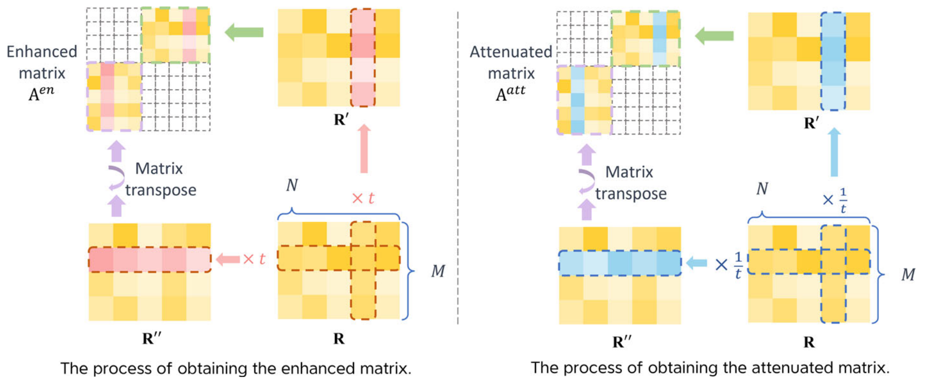

Figure 4 The process of obtaining the enhanced and attenuated matrix

matrices obtained in MDP module can be regarded as interaction information collected from various environments, Specifically, as depicted in Figure 3, these environments correspond to simulated environment 1 and simulated environment 2. *In environment 1, popular items become even more popular, while in environment 2, popular items become less popular*.

We employ a contrastive learning approach to obtain genuine invariant representations to reduce the distances between representations from different environments. The distances between different representations are calculated using InfoNCE [33] loss:

$$\mathcal{L}_{cl_1}^{\mathcal{B}} = \sum_{(u,i)\in\mathcal{B}} (f_{en}(\mathbf{z}_u, \mathbf{z}_i, \mathcal{B}) + f_{en}(\mathbf{z}_i, \mathbf{z}_u, \mathcal{B}) + f_{att}(\mathbf{z}_u, \mathbf{z}_i, \mathcal{B}) + f_{att}(\mathbf{z}_i, \mathbf{z}_u, \mathcal{B})), \tag{1}$$

$$\mathcal{L}_{cl_2}^{\mathcal{B}} = \sum_{(u,i)\in\mathcal{B}} (f_{att}(\mathbf{z}_u^{en}, \mathbf{z}_i^{att}, \mathcal{B}) + f_{att}(\mathbf{z}_i^{en}, \mathbf{z}_u^{att}, \mathcal{B})), \tag{2}$$

function $f_{en}(\cdot,\cdot,\cdot)$ and $f_{att}(\cdot,\cdot,\cdot)$ in the above equation is defned as:

$$f_s(\mathbf{z}_u, \mathbf{z}_i, \mathcal{B}) = -\log \frac{\exp(\mathbf{z}_u^\top \mathbf{z}_i^s/\tau)}{\sum_{\_,j\in\mathcal{B}} \exp(\mathbf{z}_u^\top \mathbf{z}_j^s/\tau)}, \tag{3}$$

where $\mathcal{B}$ represents a batch of user and item IDs, and $\mathbf{z}$ represents the result obtained after applying L_2 normalization to the vectors, e.g., $\mathbf{z}_i = \frac{\mathbf{e}_i}{||\mathbf{e}_i||_2}$, τ is the hyperparameter.

3.2.3 Inter-environment constraint

The next step involves calculating the inner products between user and item embeddings under different environments separately:

$$y_{u,i}^n = \mathbf{e}_u^\top \mathbf{e}_i, \tag{4}$$

$$y_{u,i}^e = \mathbf{e}_u^{en\top} \mathbf{e}_i^{en}, \tag{5}$$

$$y_{u,i}^a = \mathbf{e}_u^{att\top} \mathbf{e}_i^{att}. \tag{6}$$

We employ the Bayesian Personalized Ranking (BPR) loss [3], which is a pairwise loss that encourages the prediction of an observed entry to be higher than its unobserved counterparts:

$$\mathcal{L}_{cf}^{u,i,i_{neg}} = \sum_{t\in\{n,e,a\}} -\ln\sigma(y_{u,i}^t - y_{u,i_{neg}}^t), \tag{7}$$

where i_{neg} represents a randomly sampled item that the user has not interacted with and σ represents a sigmoid function.

Finally, in order to prevent the model's predictions from deviating excessively from the true distribution, we introduce Kullback-Leibler Divergence to estimate distribution constraints on these three dot products:

$$\mathcal{L}_{dc}^{u,i} = KL(sg(\sigma(y_{u,i}^{n})), \sigma(y_{u,i}^{e})) + KL(sg(\sigma(y_{u,i}^{n})), \sigma(y_{u,i}^{a})) \tag{8}$$

where the sg is a stop gradient operator.

Algorithm 1 The overall training process of IRL.

```
Input: user-item interaction graph $\mathcal{G}$, interaction matrix $\mathbf{R} \in \mathbb{R}^{M \times N}$, where $M$ is the number of users and $N$
    is the number of items, embedding size $d$, and the number of layers $l$ in the convolutional layer;
Output: user embeddings $\mathbf{E}_u$ and item embeddings $\mathbf{E}_i$;
 1: function LightGCN($\mathbf{R}_1, \mathbf{R}_2$)
 2:     $\mathbf{E}^{(0)} \leftarrow \begin{pmatrix} \mathbf{E}_u \\ \mathbf{E}_i \end{pmatrix} \in \mathbb{R}^{(M+N)\times d}$
 3:     $\mathbf{A} \leftarrow \begin{pmatrix} \mathbf{0}^{M\times N} & \mathbf{R}_1 \\ \mathbf{R}_2^T & \mathbf{0}^{N\times M} \end{pmatrix} \in \mathbb{R}^{(M+N)\times(M+N)}$
 4:     for each $i \in [1, l]$ do
 5:         $\mathbf{E}^{(i)} \leftarrow (\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{E}^{(i-1)}$
 6:     end for
 7:     $\mathbf{E} \leftarrow \frac{1}{l}\sum_{i=1}^{l} \mathbf{E}^{(i)}$
 8:     return $\mathbf{E}[:M], \mathbf{E}[M:]$
 9: end function
10: // Model training
11: Statistically obtain popular items $\mathbf{I}_{pop}$ and active users $\mathbf{U}_{pop}$;
12: Multiply the rows corresponding to $\mathbf{U}_{pop}$ in $\mathbf{R}$ by $t$ to obtain $\mathbf{R}^{en'}$;
13: Multiply the columns corresponding to $\mathbf{I}_{pop}$ in $\mathbf{R}$ by $t$ to obtain $\mathbf{R}^{en''}$;
14: Multiply the rows corresponding to $\mathbf{U}_{pop}$ in $\mathbf{R}$ by $\frac{1}{t}$ to obtain $\mathbf{R}^{att'}$;
15: Multiply the columns corresponding to $\mathbf{I}_{pop}$ in $\mathbf{R}$ by $\frac{1}{t}$ to obtain $\mathbf{R}^{att''}$;
16: while IRL not converge do
17:     $\mathcal{B}_{user}, \mathcal{B}_{item}, \mathcal{B}_{item_{neg}} \leftarrow$ Sample a mini-batch of data;
18:     $\mathcal{B} \leftarrow (\mathcal{B}_{user}, \mathcal{B}_{item})$;
19:     // Obtain the complete vector representations under different environments
20:     $\mathbf{E}_u, \mathbf{E}_i \leftarrow$ LightGCN($\mathbf{R}, \mathbf{R}$);
21:     $\mathbf{E}_u^{en}, \mathbf{E}_i^{en} \leftarrow$ LightGCN($\mathbf{R}^{en'}, \mathbf{R}^{en''}$);
22:     $\mathbf{E}_u^{att}, \mathbf{E}_i^{att} \leftarrow$ LightGCN($\mathbf{R}^{att'}, \mathbf{R}^{att''}$);
23:     Calculate the $\mathcal{L}_{cf}$ according to (9);
24:     Calculate the $\mathcal{L}_{cl}$ according to (10);
25:     Calculate the $\mathcal{L}_{dc}$ according to (11);
26:     Update IRL by minimizing the loss in (12).
27: end while
```

3.3 Model train and inference

During the training process, the model takes a batch of input data, including user IDs, item IDs for positive samples (representing user interactions), and sampled negative item IDs (indicating items with which users have never interacted). These are denoted as $\mathcal{B}_{user}$, $\mathcal{B}_{item}$, and $\mathcal{B}_{item_{neg}}$, respectively. We combine user IDs and item IDs into a single data batch, referred to as $\mathcal{B}_{inter}$, and define $\mathcal{B}_{bpr}$ as a batch containing user IDs, item IDs, and negative item

IDs. Subsequently, the CL loss, BPR loss, and the distribution constraint loss are calculated separately:

$$\mathcal{L}_{cf} = \sum_{(u,i,i_{neg})\in\mathcal{B}_{bpr}} \mathcal{L}_{cf}^{u,i,i_{neg}}, \tag{9}$$

$$\mathcal{L}_{cl} = \alpha \cdot \mathcal{L}_{cl_1}^{\mathcal{B}_{inter}} + \beta \cdot \mathcal{L}_{cl_2}^{\mathcal{B}_{inter}}, \tag{10}$$

$$\mathcal{L}_{dc} = \sum_{(u,i)\in\mathcal{B}_{inter}} \mathcal{L}_{dc}^{u,i}. \tag{11}$$

The final loss of the model is:

$$\mathcal{L} = \mathcal{L}_{cf} + \mathcal{L}_{cl} + \gamma \cdot \mathcal{L}_{dc}. \tag{12}$$

The overall training process of IRL is shown in Algorithm 1.

During inference, given a user u and an item i, we index the corresponding vectors from the entire embeddings after convolutional operations. We obtain the interaction prediction score directly through a dot product operation and then rank them in descending order.

4 Experiments

In this section, we seek to address the following research inquiries:

- **RQ1:** How does IRL perform compared with other debiasing strategies and popularity generalization baselines?
- **RQ2:** How does the hyperparameter t, which controls the environment simulation, affect the model performance?
- **RQ3:** How do the different components affect the model performance?
- **RQ4:** How to evaluate if the model has learned invariant representations?

4.1 Experimental settings

4.1.1 Datasets

We perform experiments on three real-world datasets: Yahoo! R3 [34], Coat [35], and KuaiRand [2]. Both the Coat and Yahoo! R3 datasets consist of two components: a biased dataset of regular user interactions and an unbiased uniform dataset obtained through a randomized trial. In this trial, users engaged with randomly selected items. The KuaiRand dataset consists of two temporal segments of data. The first segment includes interactions collected from April 8th to April 21st, 2022, under a standard recommendation strategy. The second segment encompasses interactions gathered from April 22nd to May 8th, 2022, with two types of data collected under both the standard recommendation strategy and a random intervention recommendation strategy. We refer to these three datasets as kuai-1, kuai-2, and kuai-random, respectively.

For Coat and Yahoo! R3, user-item feedback is in the form of ratings ranging from 1 to 5 stars. Ratings equal to or greater than 4 are categorized as positive feedback, while the rest are considered negative feedback. In the case of KuaiRand, positive samples are determined based on the “IsClick" signal provided by the platform. During training, we label the dataset consisting of kuai-1 and kuai-2 as Kuai-time (indicating that this dataset is designed to assess the model's effectiveness in handling popularity shifts caused by temporal changes), and we refer to the dataset consisting of kuai-1, kuai-2, and kuai-random as Kuai-random. The statistical information is outlined in Table 1.

To demonstrate the model's ability to learn invariant preferences and alleviate the impact of PDS, we conduct experiments on three datasets with unbiased test sets: Yahoo! R3, Coat, and Kuai-random (utilizing kuai-1 and kuai-2 as the training set and kuai-random as the test set). To further emphasize the model's effectiveness in alleviating PDS in the real world, we conduct experiments on Kuai-time, that is, using kuai-1 as the training set and kuai-2 as the test set.

4.1.2 Evaluation metrics

We employ the all-ranking strategy, which involves ranking all items, excluding the positive ones in the training set, by the CF model for each user. To assess the quality of the recommendations, we utilize two commonly used metrics: Recall@K, and Normalized Discounted Cumulative Gain (NDCG@K), with K set by default to 20.

NDCG@K measures the quality of recommendation through discounted importance based on position.

$$DCG_u@K = \sum_{(u,v)\in D_{test}} \frac{I(\hat{z}_{u,v} \leq K)}{\log(\hat{z}_{u,v}+1)}$$

$$NDCG@K = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}} \frac{DCG_u@K}{IDCG_u@K},$$

in these expressions, $IDCG_u@K$ represents the ideal discounted cumulative gain for user u at position K. $\mathcal{U}$ refers to the group of users, D_{test} represents the test data, and $z_{u,v}$ indicates the position of item v in the recommended ranking list for user u.

Recall@K measures how many items recommended to user will be interacted.

$$Recall_u@K = \frac{\sum_{(u,v)\in D_{test}} I(\hat{z}_{u,v} \leq K)}{|D^u_{test}|}$$

$$Recall@K = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}} Recall_u@K,$$

where D^u_{test} is the set of all interactions of the user u in test data D_{test}.

4.1.3 Baselines

We compare our method, IRL, with the following state-of-the-art baseline methods. All of these methods are constructed on the LightGCN framework and are designed to address popularity debiasing or popularity domain generalization.

Table 1 Dataset statistics

	Coat	Yahoo! R3	kuai-1	kuai-2	kuai-random
#Users	290	14,382	26,210	25,877	27,285
#Items	295	1,000	16,637	15,193	21,946
#Interactions	2,776	129,748	1,141,112	295,497	1,186,059

- **LightGCN** [21]: A simplified graph-based recommendation model that prioritizes user-item interactions for enhanced efficiency.
- **sam+reg** [8]: This methodology encompasses two crucial components, with one focusing on addressing distribution imbalances and the other dedicated to reducing biased correlations between predicted user-item relevance and item popularity.
- **IPS-CN** [13]: Building upon IPS, which addresses popularity bias by re-weighting each training instance according to item popularity, IPC-CN enhances this approach through the inclusion of normalization techniques aimed at achieving reduced variance.
- **CausE** [36]: This approach utilizes a small unbiased dataset to simulate the training process under a completely random recommendation policy.
- **MACR** [37]: This method incorporates popularity bias into the causal impact of item popularity on prediction scores by employing two modules to capture item popularity and user conformity effects, influencing the ultimate predictions.
- **CD^{2}AN** [38]: This model uses Pearson correlation to separate item properties from item popularity and introduces unexposed items to align popularity distributions between hot and long-tail items.
- **s-DRO** [39]: This model improves the Distributionally Robust Optimization (DRO) framework by adding real-time streaming optimization to reduce the impact of popularity bias on ERM.
- **InvCF** [18]: This method disentangles user preferences from item popularity, obtaining unbiased preference representations without relying on predefined popularity distributions.

4.2 Performance comparison (RQ1)

All baseline models can be divided into two categories: The Popularity Generalization methods (CD^{2}AN, sDRO, InvCF) and the Popularity Debiasing methods (sam+reg, IPS-CN, CausE, MACR). Table 2 summarizes the best results of all the models on all benchmark datasets. The results obtained on unbiased test sets, gathered using random exposure strategies in Yahoo! R3, Coat, and Kuai-random, illustrate whether the models can capture users'

Table 2 The performance comparison on Yahoo! R3, Coat, and KuaiRand datasets

Dataset	Yahoo! R3		Coat		Kuai-time		Kuai-random	
Metrics / Model	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
LightGCN [21]	0.1478	0.0686	0.2658	0.1574	0.1002	0.0932	0.0019	0.0043
sam+rg [8]	0.1498	0.0693	0.2659	0.1569	0.1211	0.1025	0.0014	<u>0.0060</u>
IPS-CN [13]	0.1331	0.0612	0.2474	0.1771	0.0935	0.1108	0.0024	0.0055
CauseE [36]	0.1490	0.0693	0.2479	0.1689	0.1357	0.0954	0.0018	0.0047
MACR [37]	0.1499	0.0691	0.0939	0.0584	0.1178	0.117	<u>0.0026</u>	0.0052
sDRO [39]	0.1426	0.0660	0.2415	0.1790	0.1264	0.0998	0.0021	0.0045
CD^{2}AN [38]	0.1397	0.0638	0.2245	0.1708	0.1093	0.1086	0.0013	0.0059
InvCF [18]	<u>0.1515</u>	<u>0.0718</u>	<u>0.2686</u>	<u>0.1819</u>	<u>0.1462</u>	<u>0.1196</u>	0.0024	<u>0.0060</u>
IRL (ours)	**0.1617**	**0.0788**	**0.2885**	**0.1906**	**0.1541**	**0.1275**	**0.0027**	**0.0063**
Imp.%	6.76%	9.75%	7.44%	4.80%	5.42%	6.63%	3.84%	3.98%

The best results are highlighted in bold while the second best ones are underlined

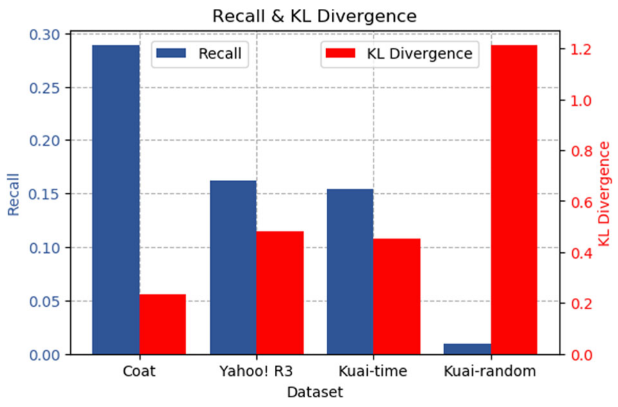

Figure 5 The relationship between KL divergence of popularity distribution across different data training sets and test sets and the Recall values

latent and invariant preferences. Meanwhile, in real-world applications, the popularity distribution dynamically changes over time. Therefore, we establish the Kuai-time dataset based on temporal variations to showcase the model's performance when dealing with popularity shifts in real deployment environments. From Table 2, we can ascertain that IRL outperforms the baseline models in all datasets, signifying that learning from invariant representations can substantially improve recommendation performance.

Simultaneously, we observe that as the degree of popularity shift between the training and test datasets increases, there is a noticeable decrease in the model's performance. As depicted in Figure 5, we calculate the Kullback-Leibler (KL) divergence of the popularity distribution of items between the training and test sets of various datasets. It is evident that on the Coat dataset, the KL divergence is minimal, and the model performs optimally. With an increase in KL divergence, there is a substantial decline in the model's Recall values (Figure 6).

Additionally, due to the model's matrix perturbation pre-processing, training efficiency maintains a linear relationship with LightGCN. This accelerates training, tuning, and deployment. In contrast, the baseline model, particularly suboptimal InvCF, requires extensive negative sample sampling for contrastive learning during training. This approach can be

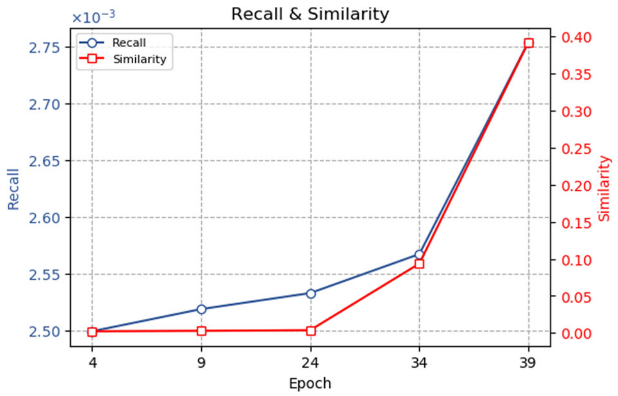

Figure 6 The relationship between model performance and similarity in vector representations

Table 3 Time cost of one epoch for InvCF and IRL

	Yahoo! R3	Kuai-time	Kuai-random
InvCF	6.9s	78.9s	82.6s
IRL	0.7s	6.8s	7.5s
Imp.%	89.85%	91.38%	90.92%

costly on larger graphs and introduce noisy signals [40]. Experiments on a server with 1 NVIDIA GeForce RTX 4090 GPU recorded the average time for our model and InvCF to complete one training epoch on various datasets, detailed in Table 3. Training time for the Coat dataset is excluded due to its small size.

4.3 Hyperparameter sensitivity (RQ2)

In Section 3, we have explained the perturbation of the interaction matrix by the hyperparameter t to introduce variations in the popularity environment. Utilizing contrastive learning, we mitigate the sensitivity of embeddings to popularity, ultimately achieving invariant representations for users and items. Adjusting various t values (with other parameters modified during the experiments), we document the model's evaluation results on Recall@20 and NDCG@20, presenting a summary in Figure 7. Figure 7(a) and (b) document the evaluation results of different metrics on the Yahoo! R3, Coat, and Kuai-time datasets. Owing to significant differences in the model's performance on the Kuai-random dataset compared to the preceding three datasets, we separately display the results of the two evaluation metrics for the Kuai-random dataset in Figure 7(c).

Figure 7 illustrates that the majority of the model's evaluation metrics across various datasets attain their optimal values at $t = 4$. A minority of results exhibit variations; for example, on the Coat dataset, the model attains the optimal Recall@20 value at $t = 5$, while on the Kuai-random dataset, it simultaneously achieves optimal NDCG@20 values at $t = 3$ and $t = 4$. For overall optimal model performance, we fix $t = 4$ in subsequent experiments. Additionally, the line chart intuitively demonstrates that the model's performance initially improves with increased perturbation strength. However, excessive perturbation in the popularity environment leads to a gradual decrease in the model's performance. Excessive perturbation may result in a significant deviation from the real environment, causing the model embeddings to shift towards an unrealistic vector distribution.

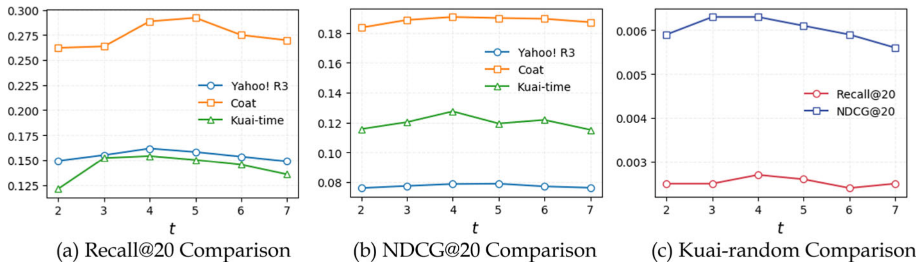

Figure 7 Model evaluation metrics under different hyperparameter t values

Table 4 The results of ablation experiments for IRL on different datasets

Yahoo! R3/Recall		Coat/Recall	
w/o cl	w/o dc	w/o cl	w/o dc
0.1546$^{\downarrow 4.39\%}$	0.1590$^{\downarrow 1.66\%}$	0.2731$^{\downarrow 5.33\%}$	0.2865$^{\downarrow 0.69\%}$
Kuai-time/Recall		Kuai-random/Recall	
w/o cl	w/o dc	w/o cl	w/o dc
0.1414$^{\downarrow 8.24\%}$	0.1495 $^{\downarrow 2.98\%}$	0.0026 $^{\downarrow 3.70\%}$	0.0026 $^{\downarrow 3.70\%}$
Yahoo! R3/NDCG		Coat/NDCG	
w/o cl	w/o dc	w/o cl	w/o dc
0.0720 $^{\downarrow 8.62\%}$	0.0755 $^{\downarrow 4.18\%}$	0.1678 $^{\downarrow 11.9\%}$	0.1799 $^{\downarrow 5.61\%}$
Kuai-time/NDCG		Kuai-random/NDCG	
w/o cl	w/o dc	w/o cl	w/o dc
0.1167 $^{\downarrow 8.47\%}$	0.1237 $^{\downarrow 2.98\%}$	0.0061 $^{\downarrow 3.17\%}$	0.0061 $^{\downarrow 3.17\%}$

The red arrows and their corresponding data represent the extent of the model's performance degradation when the corresponding module is missing

4.4 Ablation study (RQ3)

We conduct ablation studies to analyze the effects of MDP, CEC, and IEC.

Through experimentation, we have determined that setting $t = 4$ during matrix perturbation yields the best performance across all datasets. Therefore, in all ablation experiments, we maintain t in the MDP module at the default value of 4, while adjusting the other hyperparameters (α, β, γ, and τ) to suit each specific dataset. To investigate the roles of CEC and IEC, we individually disable CEC and IEC by setting $\alpha = \beta = 0$ and $\gamma = 0$. The experimental results conducted without contrastive learning (i.e., w/o cl) and distribution constraints (i.e., w/o dc) are summarized in Table 4.

Table 4 demonstrates that the exclusion of the cross-environment contrastive learning module (CEC) leads to a significant decline in performance. This highlights the crucial role of cross-environment contrastive learning in the training process and reaffirms the foundational concept of invariant representation learning. Furthermore, the distribution constraint on interactions guarantees that the model's predictions stay within a realistic and plausible range, mitigating potential deviations brought about by the incorporation of contrastive learning.

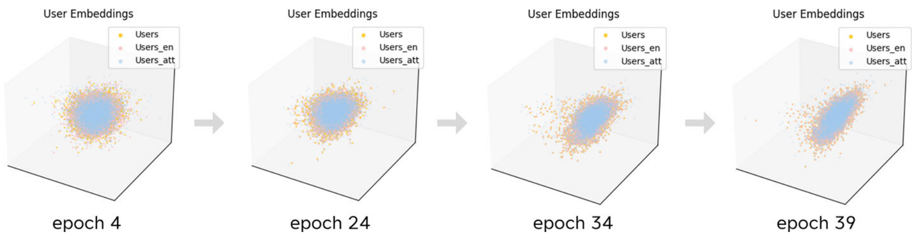

Figure 8 The distribution of user embedding vectors changing with training epochs

4.5 Case study (RQ4)

In this section, we use the Kuai-random dataset as an example. In the training process, every 5 epochs (starting from epoch 0), we assess the model's performance on the test dataset to determine whether to save the current model state. The model attains its peak performance during the 39th epoch. Following the completion of training, we assess and document the model's performance across various epochs and visualize the embedding distribution information.

During training, interaction matrices, in conjunction with graph convolutional layers, transform the initial user and item vectors into their final representations. After passing through multiple convolutional layers, we obtain user embeddings tailored to various simulated environments: red for enhanced popularity, blue for reduced popularity, and yellow for the real environment (Figure 8). As training advances, vector distributions shift from dispersion to convergence. By the 39th round, they distinctly deviate from the 4th round, indicating the convergence of feature representations during training, moving towards invariance. We sample user representations, calculate cosine similarity, and present the average similarity between vectors at each round, along with model Recall values (Figure 6). As vectors from different environments converge, the model's performance gradually improves.

5 Conclusion

In this paper, our newly proposed IRL framework perturbs the interaction matrix to simulate diverse popularity environments. Subsequently, convolution operations are applied to derive user and item representations under various environmental conditions. These representations then undergo contrastive learning to achieve invariant representations, effectively mitigating the negative impact of PDS caused by changes in popularity distribution. Extensive experiments have consistently demonstrated the effectiveness of our IRL, surpassing other baseline methods. In our future research, we plan to explore automated methods for determining enhancement and attenuation coefficients in matrix perturbation, with the aim of further enhancing our recommendation system.

Author Contributions Supervision, M.H.; Writing-original draft, H.Z; Writing-review & editing, Z.Z. and C.L.

Funding Not applicable.

Availability of Data and Materials All datasets used in this study are publicly available.

Declaration

Competing interest The authors declare no competing interests.

Ethics approval Not applicable.

References

1. He, Y., Wang, Z., Cui, P., Zou, H., Zhang, Y., Cui, Q., Jiang, Y.: Causpref: causal preference learning for out-of-distribution recommendation. In: Proceedings of the ACM Web Conference 2022, pp. 410–421 (2022)

2. Gao, C., Li, S., Zhang, Y., Chen, J., Li, B., Lei, W., Jiang, P., He, X.: Kuairand: an unbiased sequential recommendation dataset with randomly exposed videos. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, pp. 3953–3957 (2022)
3. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. arXiv:1205.2618 (2012)
4. Chen, J., Dong, H., Wang, X., Feng, F., Wang, M., He, X.: Bias and debias in recommender system: a survey and future directions. ACM Trans. Inf. Syst. **41**(3), 1–39 (2023)
5. Chen, J., Wu, J., Chen, J., Xin, X., Li, Y., He, X.: How graph convolutions amplify popularity bias for recommendation? arXiv:2305.14886 (2023)
6. Hong, Y., Yuan, X., Li, X.: Dcl4rec: an effective debiased contrastive learning framework for long-tail sequential recommendation. Available at SSRN 4558746
7. Abdollahpouri, H., Burke, R., Mobasher, B.: Controlling popularity bias in learning-to-rank recommendation. In: Proceedings of the Eleventh ACM Conference on Recommender Systems, pp. 42–46 (2017)
8. Boratto, L., Fenu, G., Marras, M.: Connecting user and item perspectives in popularity debiasing for collaborative recommendation. Inf. Process. Manag. **58**(1), 102387 (2021)
9. Chen, Z., Xiao, R., Li, C., Ye, G., Sun, H., Deng, H.: Esam: discriminative domain adaptation with non-displayed items to improve long-tail performance. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 579–588 (2020)
10. Zhu, Z., He, Y., Zhao, X., Zhang, Y., Wang, J., Caverlee, J.: Popularity-opportunity bias in collaborative filtering. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, pp. 85–93 (2021)
11. Chen, J., Dong, H., Qiu, Y., He, X., Xin, X., Chen, L., Lin, G., Yang, K.: Autodebias: learning to debias for recommendation. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 21–30 (2021)
12. Ding, S., Wu, P., Feng, F., Wang, Y., He, X., Liao, Y., Zhang, Y.: Addressing unmeasured confounder for recommendation with sensitivity analysis. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 305–315 (2022)
13. Gruson, A., Chandar, P., Charbuillet, C., McInerney, J., Hansen, S., Tardieu, D., Carterette, B.: Offline evaluation to make decisions about playlistrecommendation algorithms. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp. 420–428 (2019)
14. Wang, W., Feng, F., He, X., Wang, X., Chua, T.-S.: Deconfounded recommendation for alleviating bias amplification. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 1717–1725 (2021)
15. Zhang, Y., Feng, F., He, X., Wei, T., Song, C., Ling, G., Zhang, Y.: Causal intervention for leveraging popularity bias in recommendation. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 11–20 (2021)
16. Yu, J., Xia, X., Chen, T., Cui, L., Hung, N.Q.V., Yin, H.: Xsimgcl: towards extremely simple graph contrastive learning for recommendation. IEEE Trans. Knowl, Data Eng (2023)
17. Wang, Z., He, Y., Liu, J., Zou, W., Yu, P.S., Cui, P.: Invariant preference learning for general debiasing in recommendation. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 1969–1978 (2022)
18. Zhang, A., Zheng, J., Wang, X., Yuan, Y., Chua, T.-S.: Invariant collaborative filtering to popularity distribution shift. In: Proceedings of the ACM Web Conference 2023, pp. 1240–1251 (2023)
19. Wang, W., Lin, X., Wang, L., Feng, F., Ma, Y., Chua, T.-S.: Causal disentangled recommendation against user preference shifts. ACM Trans. Inf, Syst (2023)
20. Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Zhang, D., Le Priol, R., Courville, A.: Out-of-distribution generalization via risk extrapolation (rex). In: International Conference on Machine Learning, pp. 5815–5826, PMLR (2021)
21. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: 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, pp. 639–648 (2020)
22. Arjovsky, M., Bottou, L., Gulrajani, I., Lopez-Paz, D.: Invariant risk minimization. arXiv:1907.02893 (2019)
23. Bühlmann, P.: Invariance, causality and robustness (2020)
24. Liu, J., Hu, Z., Cui, P., Li, B., Shen, Z.: Heterogeneous risk minimization. In: International Conference on Machine Learning, pp. 6804–6814, PMLR (2021)
25. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: International Conference on Machine Learning, pp. 1180–1189, PMLR (2015)

26. Ahuja, K., Caballero, E., Zhang, D., Gagnon-Audet, J.-C., Bengio, Y., Mitliagkas, I., Rish, I.: Invariance principle meets information bottleneck for out-of-distribution generalization. Adv. Neural Inf. Process. Syst. **34**, 3438–3450 (2021)
27. Liu, E.Z., Haghgoo, B., Chen, A.S., Raghunathan, A., Koh, P.W., Sagawa, S., Liang, P., Finn, C.: Just train twice: improving group robustness without training group information. In: International Conference on Machine Learning, pp. 6781–6792, PMLR (2021)
28. Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., Xie, X.: Self-supervised graph learning for recommendation. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 726–735 (2021)
29. Yu, J., Yin, H., Xia, X., Chen, T., Cui, L., Nguyen, Q.V.H.: Are graph augmentations necessary? simple graph contrastive learning for recommendation. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1294–1303 (2022)
30. Xia, L., Huang, C., Xu, Y., Zhao, J., Yin, D., Huang, J.: Hypergraph contrastive collaborative filtering. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 70–79 (2022)
31. Lin, Z., Tian, C., Hou, Y., Zhao, W.X.: Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In: Proceedings of the ACM Web Conference 2022, pp. 2320–2329 (2022)
32. Cai, X., Huang, C., Xia, L., Ren, X.: Lightgcl: simple yet effective graph contrastive learning for recommendation. arXiv:2302.08191 (2023)
33. Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv:1807.03748 (2018)
34. Marlin, B.M., Zemel, R.S.: Collaborative prediction and ranking with non-random missing data. In: Proceedings of the Third ACM Conference on Recommender Systems, pp. 5–12 (2009)
35. Schnabel, T., Swaminathan, A., Singh, A., Chandak, N., Joachims, T.: Recommendations as treatments: Debiasing learning and evaluation. In: International Conference on Machine Learning, pp. 1670–1679, PMLR (2016)
36. Bonner, S., Vasile, F.: Causal embeddings for recommendation. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp. 104–112 (2018)
37. Wei, T., Feng, F., Chen, J., Wu, Z., Yi, J., He, X.: Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 1791–1800 (2021)
38. Chen, Z., Wu, J., Li, C., Chen, J., Xiao, R., Zhao, B.: Co-training disentangled domain adaptation network for leveraging popularity bias in recommenders. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 60–69 (2022)
39. Wen, H., Yi, X., Yao, T., Tang, J., Hong, L., Chi, E.H.: Distributionally-robust recommendations for improving worst-case user experience. In: Proceedings of the ACM Web Conference 2022, pp. 3606–3610 (2022)
40. Zhou, X., Zhou, H., Liu, Y., Zeng, Z., Miao, C., Wang, P., You, Y., Jiang, F.: Bootstrap latent representations for multi-modal recommendation. In: Proceedings of the ACM Web Conference 2023, pp. 845–854 (2023)