Mitigating Popularity Bias in Recommendation with Unbalanced Interactions: A Gradient Perspective

Weijieying Ren *, Lei Wang *, Kunpeng Liu #, Ruocheng Guo ♯, LIM Ee Peng♭ and Yanjie Fu §

*Department of Computer Science, University of Central Florida, USA
♭Department of Computer Science, Singapore Management University, Singapore
♯Department of Computer Science, Portland State University, USA
§ByteDance AI lab, UK

wjyren@knights.ucf.edu, lei.wang.2019@phdcs.smu.edu.sg, kunpeng@pdx.edu
ruocheng.guo@bytedance.com, eplim@smu.edu.sg, yanjie.fu@ucf.edu

Abstract—Recommender systems learn from historical user-item interactions to identify preferred items for target users. These observed interactions are usually unbalanced following a long-tailed distribution. Such long-tailed data lead to popularity bias to recommend popular but not personalized items to users. We present a gradient perspective to understand two negative impacts of popularity bias in recommendation model optimization: (i) the gradient direction of popular item embeddings is closer to that of positive interactions, and (ii) the magnitude of positive gradient for popular items are much greater than that of unpopular items. To address these issues, we propose a simple yet efficient framework to mitigate popularity bias from a gradient perspective. Specifically, we first normalize each user embedding and record accumulated gradients of users and items via popularity bias measures in model training. To address the popularity bias issues, we develop a gradient-based embedding adjustment approach used in model testing. This strategy is generic, model-agnostic, and can be seamlessly integrated into most existing recommender systems. Our extensive experiments on two classic recommendation models and four real-world datasets demonstrate the effectiveness of our method over state-of-the-art debiasing baselines.

Index Terms—Recommendation system, Popularity bias.

I. INTRODUCTION

Recommender systems (RS) provide users with newly suggested items by mining user preferences from historical user-item interactions [1]–[4]. Each observed user-item interaction represents an implicit feedback (e.g., purchases, clicks, browses) indicating the user’s preference on the item. We call this a positive interaction. To train a RS model, we can uniformly select a few items having no interaction with the user as negative items. A RS model based on such implicit feedback data is designed to predict positive items with higher prediction scores than negative items. Generally, each item will receive a positive gradient if it has observed interactions with a user and obtain a negative gradient if it is selected as a negative sample.

Nevertheless, recent studies [1], [4], [5] found that in most real-world application scenarios, user-item interactions in recommender systems are highly imbalanced following a long-tailed distribution [6], [7]. Only a small number of items are popular as they frequently interact with users and thus are more likely favored by users. When such long-tailed data are used for training a RS model, popularity bias arises in the trained model, i.e., popular items are overly recommended to all users, while some less popular items may be more relevant to some specific users. Popularity bias can result in low visibility of unpopular items, reduced diversity of markets and damage consumer experience. Mitigating popularity bias is therefore highly demanded.

To deal with popularity bias, one prominent direction is to design a re-weighting loss, e.g., inverse propensity score (IPS), to balance overall item distributions [8], [9]. Another alternative is to conduct knowledge transfer [6] with the assumption that knowledge represented in popular items should be beneficial to the user preference mining in less popular items. Recently, more sophisticated models are designed to preserve user specific tastes via the usage of side information [10], autoencoder model [11] and to disentangle user preference following pre-defined prior heuristics [12]. However, previous works either require complex model design [11], degenerate embedding representation or rely on strong assumption [12],...
which is lack of generalization ability. Unlike the aforementioned strategies, we approach the popularity bias by analyzing gradient distortion in long-tailed data, attributed to the combination of gradients generated by positive and negative items. We discover that (i) the overall gradient direction of popular items’ embeddings tend to be closer to the gradient of positive interactions, and (ii) the magnitude of positive gradient for popular items is far larger than that of unpopular items.

To illustrate the above challenges, we visualize the related variables in the training process of Movielens-10M dataset in Fig.1. Specifically, the cosine similarity between the accumulated positive gradient (given by $\sum_{t=1}^{T} \nabla_t L'(u_+, i_+)$, where $t$ is the iteration step) and overall gradient of each item is visualized in Fig. 1(a) for items sorted by inverse popularity order. We also show the cosine similarity between the negative gradient and overall gradient of items. The high similarity (i.e., close to 1) between the overall gradient and the accumulated positive gradient of popular items indicates that the overall gradient of each popular item is shifted towards the direction of its positive interactions. The opposite can be observed for unpopular items. Fig. 1(b) shows the norm of accumulated positive and negative gradients for items sorted in the descending popularity order. In this figure, popular items exhibit large accumulated positive gradient norm values than their accumulated negative gradient norm values. In other words, the accumulated gradients of popular items increase the norm of their corresponding item embedding vectors and results in tendency to rank high in recommendation results.

To further analyze the underlying reasons of the above empirical findings, we review the optimization process of the classical pairwise ranking loss used in implicit RS, i.e., BPR loss [13]. It is clear that item (or user) embedding vectors are mainly determined by their accumulated gradients. Considering a popular item $i$ illustrated in Fig. 2, vectors $\overrightarrow{AC}$ and $\overrightarrow{AB}$ represent its accumulated positive and negative gradient respectively. The overall gradients $\overrightarrow{AD}$ is shifted towards $\overrightarrow{AC}$ due to the latter’s large magnitude, making the learned item embedding only facilitate the learning of positive interaction (vice versa for unpopular items). The accumulated gradients are derived by summing up gradient values across all iterations when optimizing the objective function $L'(u, i)$. Consequently, We identify three factors which may account for the reason of the existence of popularity bias in implicit RS: 1) item popularity, 2) user activity, and 3) user conformity.

Our Approach. Motivated by mitigating popularity bias, we found that the combined analysis for gradients generated by positive and negative interactions are key solutions. Hence, we propose the concept of user/item embedding adjustment. In the training stage, we only normalize user embeddings to mitigate the influence of high activity users and record the accumulated gradients of users and items. In the testing stage, we propose to mitigate bias by adjusting the learned user and item embeddings to mitigate the influence by item popularity and user conformity. Our main contributions are as follows:

- **A Novel Gradient Perspective to Understand Popularity Bias Mechanism.** Classical methods focus on understanding popularity bias from balancing the overall data distribution, e.g., popular and unpopular groups. In this paper, we provide new findings that 1) the overall gradient direction of popular (unpopular) items is shifted to be closer to the gradient on positive (negative) interactions; 2) positive gradient magnitude of popular items is usually much larger than that of unpopular items. We analyze such phenomenon from the gradient perspective, and provide a new insight on balancing the positive and negative gradients for the same item.

- **Gradient Based Embedding Adjustment Approach.** Motivated by the new findings, we identify three essential factors, namely item popularity, user conformity, and user activeness and analyze how user and item embeddings are related to them during the optimization steps. We develop a simple yet novel embedding post-doc adjustment method. This framework is explainable and interpretable.

- **Extensive Experiments with Real-world Datasets.** We conduct extensive experiments to evaluate the effectiveness of our proposed methods in terms of overall accuracy, generalisation, and efficiency against state-of-the-art baselines. We also conducted ablation studies to evaluate the components of our proposed methods.

II. PRELIMINARIES

**Problem Statement.** Suppose we have a user set $U$, an item set $I$ and an user-item interaction matrix $Y \in \{0, 1\}^{U \times I}$, where $Y_{ui}$ indicates whether $u \in U$ has made an interaction (e.g., adoption or purchase) with item $i \in I$ or not. The goal of a recommender is to learn a scoring function $r(u, i)$ to recommend a list of $k$ items to a user $u$:

$$\hat{y}_{ui} = r(u, i) = P_u^T Q_i$$

where $\hat{y}_{ui}$ is the predicted score, $P_u$ is the user representation vector for the user $u$, and $Q_i$ is the item representation vector for the item $i$. $P_u (Q_i)$ is derived from the user encoder $f(u(\Theta_U))$ (the item encoder $f(i(\Theta_I))$). Note that the user or item encoder is agnostic to model, which can be the embedding matrix [14], graph neural network [15], or other models. For simplicity, we refer to $P_u (Q_i)$ as user embedding (item embedding) in the rest of the paper.
Why does the embedding magnitude matter in RS problems? In the testing stage, the recommender system returns the personalized ranking of items $R_u$ for a user $u$:

$$R_u = \{(i, \text{rank}(y_{ui})) \}_{i \in I}$$

$$= \{(i, \text{rank}(P_u^T Q_i)) \}_{i \in I}$$

$$= \{(i, \text{rank}([Q_i | \cos(P_u, Q_i)]) \}_{i \in I}$$

where $\text{rank}$ is a ranking function based on a set of scoring function values between $u$ and all items $i \in I$. In Eq. 2, we omit $\|P_u\|$ as $R_u$ involves the same user $u$. In other words, the personalized ranking result during the testing stage only depends on the magnitude of item embedding $Q_i$ and its cosine similarity with the user embedding $P_u$.

How are the embedding magnitudes generated in RS? In the training stage, we often adopt a pairwise ranking loss, e.g., Bayesian Personalized Ranking (BPR) loss [13] and Binary cross entropy (BCE) [16] loss. Without loss of generality, we revisit BPR loss functions for recommender system with implicit feedback. In realistic scenarios, only positive implicit feedback (e.g., clicks and purchases) are observed. In implicit RS, we assume that user $u$'s preference over an item $i$ with an observed $(u, i)$ interaction is ranked higher than the unobserved $(u, j)$ interaction as a negative sample, where the item $j$ is uniformly selected from item set $I$ that does not have observed interactions with the user. Without loss of generality, we represent BPR loss here to maximize the likelihood of observing such pairwise ranking relations:

$$L_{BPR} = \sum_{(u,i,j) \in D_s} \log(\sigma(\hat{y}_{ui} - \hat{y}_{uj})) - \lambda_\Theta \|\Theta\|^2$$

where $D_s$ is a set of $(u, i, j)$ triplets with $(u, i)$ and $(u, j)$ as positive and negative samples respectively. $\Theta$ is the model parameters and $\sigma(x) = \frac{1}{1 + \exp(-x)}$ is the sigmoid function. The gradient of BPR loss with respect to the model parameters $\Theta$ (e.g., user/item embeddings) can be written as:

$$\frac{\partial L_{BPR}}{\partial \Theta} = \sum_{(u,i,j) \in D_s} \frac{\partial \log(\sigma(\hat{y}_{ui} - \hat{y}_{uj}))}{\partial \Theta} - \lambda_\Theta \frac{\partial \|\Theta\|^2}{\partial \Theta}$$

$$\propto -e^{-(\hat{y}_{ui} - \hat{y}_{uj})} \cdot \frac{\partial (\hat{y}_{ui} - \hat{y}_{uj})}{\partial \Theta} - \lambda_\Theta \Theta$$

More concretely, the gradients for updating a matrix factorization model are:

$$\frac{\partial (\hat{y}_{ui} - \hat{y}_{uj})}{\partial \Theta} = \begin{cases} (Q_i - Q_j) & \text{if } \Theta = P_u \\ P_u & \text{if } \Theta = Q_i \\ -P_u & \text{if } \Theta = Q_j \\ 0 & \text{otherwise} \end{cases}$$

**Failure case analysis.** For BPR loss optimization, we notice that for each user-item pair $(u, i)$ with positive interaction, the model will randomly sample a negative pair $(u, j)$. For a popular item $i$ which matches mainstream preference, it is frequently updated by a positive gradient $P_u$. It is only updated by a negative gradient $-P_u$ when it serves as a negative sample. In other words, item $i$ will be updated with positive gradients more frequently than negative gradients during the optimization.

| User Group | #Users | pop!4u | unp!4u |
|------------|--------|--------|--------|
| All        | 37,962 | 17.3   | 4.3    |
| Active     | 20,801 | 25.1   | 6.6    |
| Inactive   | 17,161 | 7.9    | 1.7    |

| Item Group | #Items | act_u4u | ina_u4u |
|------------|--------|---------|---------|
| All        | 4,819  | 136.6   | 34.2    |
| Popular    | 328    | 1,591.6 | 414.0   |
| Unpopular  | 4,491  | 30.3    | 6.4     |

### III. Motivation: Popularity Bias from a Gradient Perspective

In this section, we discuss popularity bias from a gradient update perspective, i.e., how the item popularity, user activity and user conformity influence the gradient magnitude and shift the overall gradient to the positive interaction direction for popular items.

Based on Eq.5, we can derive that user (item) embeddings affect the magnitude and direction of the item (user) embeddings. For simplicity, we define the update of the latent vector of positive items $Q_i^t$, negative items $Q_j^t$, and users $P_u^t$ in the $t^{th}$ iteration step for a data example $(u, i, j)$ with the learning rate $lr$ as follows:

$$Q_i^t = Q_i^{t-1} - lr \cdot P_u^{t-1},$$

$$Q_j^t = Q_j^{t-1} - lr \cdot (-P_u^{t-1}),$$

$$P_u^t = P_u^{t-1} - lr \cdot (Q_i^{t-1} - Q_j^{t-1})$$

For supporting the analysis of popularity bias from a gradient perspective. We take the Movielens-10M as an example, and statistics is shown in Table I. We sort items by their number of interactions in descending order and select sorted items until the number of interactions of selected items up to 80% of total interactions in the training set (in fact, less than 20% items account for more than 80% interactions in many real scenarios). We assume this selected items as popular items and the remaining items as unpopular items. In the same way, we can get active users and inactive users.

**Influence of User Activity on Item Side.** Let the latent vector of item $i$ in the 0-th iteration be $Q_i^0$. Then we can obtain the equivalent form of $Q_i^t$ as:

$$Q_i^t = Q_i^0 - lr \cdot \sum_{(u,i) \in Y_i^{+,t-1}} P_u - \sum_{(u',i) \in Y_i^{-,t-1}} P_{u'}$$

where $Y_i^{+,t-1}$ and $Y_i^{-,t-1}$ denote the set of use-item pairs in which the item $i$ works as the positive and negative items in training set before the iteration step $t$, respectively. Without loss of generalization, we simplify all equations later by ignoring iteration steps $t$ and $t-1$. We let $\Delta_i$ represent the increment of $Q_i^t$:

$$\Delta_i = \sum_{(u,i) \in Y_i^+} P_u - \sum_{(u',i) \in Y_i^-} P_{u'}$$
Since users in both $Y_i^+$ and $Y_i^-$ consist of active users and inactive users, equation 8 can be extended as:

$$\Delta_i = \sum_{(u,i) \in Y_{i,act}^+} P_u + \sum_{(u,i) \in Y_{i,act}^-} P_u - \sum_{(w',i) \in Y_{i,act}^+} P_{w'} - \sum_{(w',i) \in Y_{i,act}^-} P_{w'}$$

(9)

where $Y_{i,act}^+$ ($Y_{i,act}^-$) is the set of use-item pairs for item $i$ in which users are active (inactive) users.

As shown in Table I, for items in any group (all, popular, and unpopular), the number of positive active users is much larger than inactive users. Besides, the number of negative active users is much smaller than inactive users via negative sampling over uniform distribution. Thus, the items embeddings are easily dominated by active users Inferential Failure, according to the cumulative gradients calculated by Eq. 8. We can derive that the higher the popularity of the item is, the larger the magnitude of its corresponding embedding will be. Moreover, the distance between popular items might be close in latent space due to the user conformity.

**Influence of Item Popularity on User Side.** Similar to analyze items from the gradient perspective, we can obtain accumulated gradients $\Delta_u$ of user $u$ by:

$$\Delta_u = \sum_{(u,i) \in Y_{i,act}^+} Q_i + \sum_{(u,i) \in Y_{i,act}^-} Q_i - \sum_{(u,j) \in Y_{i,unp}^+} Q_j - \sum_{(u,j) \in Y_{i,unp}^-} Q_j$$

(10)

For users in any group (all, active, inactive), the number of positive popular items is much larger than unpopular items. The number of negative popular items is however much smaller than unpopular items in randomly negative sampling. Therefore, popular items significantly dominate user embeddings due to the accumulated gradients defined in Eq. 10.

IV. PROPOSED SOLUTION

In this section, we present our method. Specifically, in the training stage, our method normalizes only user embeddings after learning the user and item embeddings using LGN [15] or MF [14]. We then apply post-hoc debiasing on user and item embeddings in the testing stage to mitigate item popularity bias. We first give our definition of an unbiased item (user) embedding in implicit RS. We next give a detailed description about how to mitigate user activity in the training stage, and how to mitigate item popularity and user conformity influence.

- **Disentangled embedding learning.** We first define four factors influencing user and item embedding learning.
- **A Post-hoc debiasing method.** Following the gradient optimization process, we analyze how the four factors propagate popularity bias into embedding updating phase and design a specific debiasing term for each of the four factors accordingly.

### A. Unbiased Embedding Definition

As described in Section 3, there are three factors exacerbating the imbalance issue between accumulated positive and negative gradients of the each item: (1) user’s high activity interacting with many items; (2) item’s high popularity involved in more positive interactions. (3) user’s conformity towards other mainstream users. For popular items, its overall gradient is shifted towards the positive gradient direction, making the popular items heavily exposed. The unpopular items in contrast are under-exposed. The large magnitude of popular item vectors results in higher ranking scores for positive interactions. Since updating of user and item latent vectors is intertwined in the implicit RS optimization process, we propose to adjust both user and item latent vectors to mitigate popular bias.

Given the item embedding $Q_i$ and user embedding $P_u$ trained using vanilla MF [14] (or LGN [15]) model, we represent the item embedding as:

$$Q_i = Q_i^{pop} + Q_i^{int}$$

(11)

where $Q_i^{pop}$ captures the item’s popularity and $Q_i^{int}$ captures the item’s genuine features. Similarly, we represent a user embedding $P_u$ as:

$$P_u = \|P_u^{act}\|_2(P_u^{conf} + P_u^{int})$$

(12)

where $P_u^{act}$ captures the users’ activeness, $P_u^{conf}$ signifies the user’s conformity to other mainstream users, and $P_u^{int}$ denotes the user’s genuine interest. Notably, taking the product of $P_u^{act}$ and $P_u^{conf} + P_u^{int}$ is justified by the fact that an active user is inclined to have more observed interactions [17]. Formally, the unbiased matching score for a user and an item is:

$$\hat{y}_{ui} = P_u^{int} \cdot Q_i^{int}$$

(13)

In the following sections, we will describe how we mitigate $P_u^{act}$ in the training stage and eliminate $Q_i^{pop}$ and $P_u^{conf}$ in the testing stage.

### B. Normalizing User Embeddings in Training

As an active user will have more observed interactions with items, their accumulated gradients based on these items will likely be larger than those of other less active users. For example, if users $u_1$ and $u_2$ share similar item preferences with $u_1$ being more active, the embedding vectors of $u_1$ and $u_2$ will have similar updating direction but the vector norm of user $u_1$ will be much larger. Such large embedding contributes to the gradient updating of its corresponding items, aggravating the imbalance issue between accumulated positive and negative gradients of items.

To reduce the bias propagation of highly active users, we normalize each user embedding $\hat{P}_u$ into a unit vector during the training stage:

$$\hat{P}_u = P_u / \|P_u^{act}\|_2$$

(14)

where $\hat{P}_u$ is the learned user embedding after normalization. To keep the notation simple, we use $\|P_u\|$ to denote $\|P_u^{act}\|_2$. The complete training procedure is shown in Algorithm 1.
Algorithm 1 Training

**Require:** Backbone user encoder $f(\cdot|\Theta_U)$, backbone item encoder $f(\cdot|\Theta_I)$, user set $U$, item set $I$, and user-item interactions $Y$.

1. Initialize the user and item encoder parameters $\Theta_U$ and $\Theta_I$ respectively;
2. $P = 0$ and $Q = 0$;
3. for $t = 1$ to $T$ do
4. $Y_t^+ \leftarrow$ SampleMiniBatch($Y, m$)
5. $Y_t^- \leftarrow$ NegativeSampler($Y_t^+$)
6. for $k = 1$ to $m$ do
7. $u, i, j = Y_{t,k}^+, Y_{t,k}^-$
8. $P_u, Q_i, Q_j = f(u(\Theta_U), f(i(\Theta_U)), f(j(\Theta_U)))$
9. $P_u = P_u + ||P_u||$
10. $Q_i = Q_i - lr \cdot (Q_i - Q_j)$
11. $Q_j = Q_j - lr \cdot (-P_u)$
12. $y_{ui} = P_u \cdot Q_i$
13. $y_{uj} = P_u \cdot Q_j$
14. $L_k = \log (\sigma(y_{ui} - y_{uj})) - \lambda_{\Theta}||\Theta||^2$
15. end for
16. end for
17. $P = \frac{1}{T} \sum_{t=m}^{T} P_u$
18. $Q = \frac{1}{T} \sum_{t=m}^{T} Q_i$
19. end for

**Algorithm 2 Adjusting Popularity Bias in Inference**

**Input:** Backbone user encoder $f(\cdot|\Theta_U)$, backbone item encoder $f(\cdot|\Theta_I)$, user $u$, item $i$, recorded user conformity $P$, and recorded item popularity $Q$.

**Output:** $\hat{y}_{ui}$

1. $P_u, Q_i, Q_j = f(u(\Theta_U), f(i(\Theta_U)), f(j(\Theta_U)))$
2. $Q_i^{\text{int}} = Q_i - \alpha_1 \cdot \cos(Q_i, \hat{Q}_i) \cdot ||Q_i|| \cdot \hat{Q}$
3. $P_u^{\text{int}} = P_u - \alpha_2 \cdot \cos(P_u, \hat{P}) \cdot ||P_u|| \cdot \hat{P}$
4. $\hat{y}_{ui} = P_u^{\text{int}} \cdot Q_i^{\text{int}}$

C. Mitigating Popularity Bias in Inference

1) Mitigating item popularity influence: For a popular item, its positive interactions with users are frequently observed among the training samples, shifting the learned embedding to the positive interaction direction. As depicted in Fig.3, the motivation here is to modify the overall learned item embedding $\tilde{A}D$ to an unbiased item vector $AD'$. There are a number of ways to do so. In this paper, we propose to modify the biased accumulated gradient $\tilde{A}D$ only through its accumulated positive gradients $\hat{A}C$ and keep $\tilde{A}B$ unchanged. Nevertheless, the iterative optimization of user/item embedding makes it unclear how much other items and relevant user embeddings contribute to the accumulated positive gradients for each item, worsening the imbalance between $\tilde{A}B$ and $\hat{A}C$. Hence, the important question here is how to approximate the biased embedding $P^{\text{pop}}$ in Eq.12. Our analysis in Section 3 suggests that 1) for popular items, their overall accumulated gradient (learned) embeddings $(\tilde{A}D)$ are shifted towards its positive gradient direction $(\hat{A}C)$; 2) Iterative optimization of user and item embedding makes item embeddings are intertwined with other relevant item information. Besides, user conformity and activity makes their item embeddings reflects similar user preference. Thus, popular items share similar positive gradient directions. Based on the two findings, we use the mean embedding of popular items’ embedding to represent such biased embedding $Q^{\text{pop}}$. The direction of such biased vector is named as popular direction $\hat{Q}$.

Thus, we decompose the an item embedding $Q_i$ into a genuine features embedding $Q_i^{\text{int}}$ and a projection on the positive direction $\hat{Q}$. Projection vector here refers to item popularity embedding $Q_i^{\text{pop}}$ in Eq.11. In the test phase, we adjust the learned item embedding $Q_i$ and derive its genuine representation $Q_i^{\text{int}}$:

$$Q_i^{\text{int}} = Q_i - Q_i^{\text{pop}} = Q_i - \alpha_1 \cdot \cos(Q_i, \hat{Q}_i) \cdot ||Q_i|| \cdot \hat{Q}$$ (15)

where $\alpha_1$ is a scaling parameter which controls the magnitude of the modified positive vector $\alpha \hat{C}$. In this paper, we choose the value of $\alpha_1$ based on the validation set.

2) Mitigating user conformity: Different from item embeddings which are updated by both positive and negative gradients, a user embedding is only updated by the gradient $Q_i - Q_j$, in which items $i$ and $j$ appear in observed and non-observed interactions with the user, respectively. A user embedding is inevitably influenced by the preference of other users during the gradient updates with BPR loss, and thus account for user conformity.

For example, consider a scenario where three users interact with an item $i_1$, while the first user $u_1$ also interacts with items $i_2$ and $i_3$. Since $i_1$ has multiple observed interactions, it receives the accumulated gradients based on three users. Whenever we update user embedding $\nabla_u L(u, i) = Q_{i_1}$. The embedding of $Q_{i_1}$ with large magnitude will increase the magnitude of latent vector of $u$.

The situation here is similar to item popularity bias, we therefore adopt a similar way to extract user’s true interest by subtracting the user conformity projection:

$$P_u = P_u - \alpha_2 \cdot \cos(P_u, \hat{P}) \cdot ||P_u|| \cdot \hat{P}$$ (16)

where $P$ denotes the mean or normalized embedding of mainstream users’ embedding. The term $\alpha_2 \cdot \cos(P_u, \hat{P}) \cdot ||P_u|| \cdot \hat{P}$ captures the user conformity embedding $P_u^{\text{con}}$ in Eq.12. $\alpha_2$ is a scaling parameter. We show inference procedure in Algorithm 2.

In this section, we conduct experiments to show the effectiveness and efficiency of the proposed framework. Specifically, we aim to answer the following research questions:
Table II

| Dataset          | User | Item | Interaction | Sparsity |
|------------------|------|------|-------------|----------|
| Movielens-10M    | 37,962 | 4819 | 1,371,473 | 0.007496 |
| Netflix          | 32,450 | 8432 | 2,212,690 | 0.008086 |
| Adresa           | 13,485 | 744  | 116,321   | 0.01594  |
| Gowalla          | 29,858 | 40,981| 1,027,370 | 0.008480 |

- **RQ1**: (a) Does our proposed method outperform state-of-the-art recommendation methods under the non-IID setting? (b) Furthermore, does it generalize well on both BPR and BCE loss? (c) Is it more efficient than state-of-the-art methods?
- **RQ2**: How do different components (e.g., normalization, user conformity subtrahend, item popularity subtrahend, and these subtrahends’ hyper-parameters) affect the result of our method?
- **RQ3**: (a) How well does our method mitigate the popularity bias? (b) How does the performance of our method change with the intervention proportion of test data?

### A. Experiment Settings

**Datasets and Data Preprocessing.** Our experiments utilize four publicly available datasets: 1) Movielens-10M is a dataset with 10M movie ratings. 2) Netflix is from an open competition and it covers film ratings. 3) Adresa is a popular dataset for news recommendation. 4) Gowalla is a dataset covering location-based user check-in behaviors.

We summarize statistics of the datasets in Table II. To investigate the extent to which the debiased models reduce popularity bias, we replace the conventional evaluation strategy, where the test set consists of IID samples from the observed interactions, with a sampling strategy to construct a debiased test set. Specifically, we randomly sample interactions with equal probability given to all items to construct the test and validation data. We then use the remaining data that come with popularity bias as the training data. We follow DICE to process Movielens-10M and Netflix datasets and MACR to process Adresa and Gowalla for a fair comparison. More specifically, we sample 30% unbiased interactions regarding items as the test set, 10% as the validation set, and the remaining 60% as training data for Movielens-10M and Netflix. We prepare 10% unbiased data as the test set, another 10% unbiased data as the validation set, and the rest as the training set for Adresa and Gowalla.

**Baselines.** We implement our method over the classical MF and the state-of-the-art LGN to investigate the effectiveness and generalization ability of our proposed approach across different backbone recommendation models. We compare our methods with the following baselines. Matrix factorization (MF) is a representative collaborative filtering method. LGN is a state-of-the-art collaborative filtering based on a graph convolution network. IPS is a widely used re-weighting method based on causal inference.

We show the effect of hyper-parameters $\alpha_1$ and $\alpha_2$ in detail in Appendix.

1https://grouplens.org/datasets/movielens/

2We show the effect of hyper-parameters $\alpha_1$ and $\alpha_2$ in detail in Appendix.

weighting score, i.e., inverse propensity score is set as the inverse of corresponding item popularity value. It eliminates popularity bias by imposing large weight to unpopular items. **CausE** is a domain adaptation algorithm that learns a causal embedding from a small uniform dataset. DICE is the state-of-the-art disentangled method for learning causal embedding to cope with popularity bias problem. It designs a framework with causal-specific data to disentangle interest and popularity with pre-defined guidelines. DICE leverages popularity bias in the inference. We use the code provided by authors. MACR addresses popularity bias from a causal-effect view. The counterfactual inference is performed to estimate the direct effect from item properties to the ranking score, which is removed to eliminate the popularity bias.

**Implementation Details.** We evaluate the effectiveness performance by three widely-used metrics, i.e., top-k Recall, Hit Ratio (HR@k), and top-k Normalized Discounted Cumulative Gain (NDCG@k), which are used in DICE and MACR. For MACR on Movielens-10M and Netflix, we use the source code provided by their authors and set the batch size and embedding size as 128 for a fair comparison. For the reported results of the remaining baselines, we refer to DICE and MACR directly. Note that our method is based on two backbones (i.e., MF and LGN). The hyper-parameters of backbones are set following the suggestions from DICE for Movielens-10M and Netflix and MACR for Adresa and Gowalla to guarantee a fair comparison with DICE and MACR. $\alpha_1$ and $\alpha_2$ are determined on validation set based on grid search in the search list formed by taking values from 0 to 2 with a 0.2 increment.

### B. Overall Comparison (RQ1)

The recommendation performance (i.e., Recall@20, HR@20, and NDCG@20) of different models are presented in Table III. The methods with best performance are marked with boldface fonts and the second best ones are underlined. Based on the results, we make the following observations:

- As shown in Table III, our proposed method consistently outperforms all the baselines on all the datasets w.r.t. all the evaluation metrics. The state-of-the-art baselines DICE and MACR incorporate extra modules to reduce popularity bias, which demands additional prior knowledge and efforts to exploit good architecture by trial and error during the training stage. In contrast, our method only needs to store the aggregated influence of the dominating items and users that is caused by gradient based updates. It efficiently handles popularity bias by performing a post-hoc subtraction calculation in the inference step, which eventually leads to better performance than baselines.

- Our proposed method is both general and effective. It can easily be incorporated into various backbone RS models such as MF, LGN, and any other general recommendation models. At the same time, it achieves promising performance on debiased test set. Compared to classical MF, our proposed method that uses MF as the backbone model achieves average 37.33%, 33.87%, and 51.79% improve-
Our proposed method has stronger generalization ability, as to the remaining baselines for mitigating popularity bias, Figure 4 shows how our method using MF as the backbone performs when applied with BPR or BCE loss on Movielens-10m and Netflix datasets.

Table III

| Dataset    | MovieLens-10M | Netflix |
|------------|---------------|---------|
| Method     | Recall@20     | HR@20   | NDCG@20 |
| Original   | 0.128         | 0.442   | 0.084   |
| IPS        | 0.133         | 0.443   | 0.085   |
| CausE      | 0.115         | 0.406   | 0.074   |
| DICE       | 0.163         | 0.519   | 0.108   |
| MACR       | 0.138         | 0.469   | 0.089   |
| Ours       | 0.173         | 0.536   | 0.116   |

Table IV

| Method    | MovieLens-10M | Netflix |
|-----------|---------------|---------|
| MF        | 47            | 77      |
| DICE      | 124           | 207     |
| MACR      | 57 (1.21x)    | 87 (1.29x) |
| Ours      | 51 (1.08x)    | 78 (1.01x) |
| LGN       | 126           | 201     |
| DICE      | 307 (2.34x)   | 454 (2.25x) |
| MACR      | 304 (2.41x)   | 447 (2.22x) |
| Ours      | 129 (1.02x)   | 203 (1.01x) |

Figure 4. Performance (Recall@20, HR@20, and NDCG@20) of MF and Our Method with BPR and BCE loss on MovieLens-10m and Netflix datasets.

C. Further Analysis

In this section, we continue to evaluate our proposed method by answering the remaining research questions:

1) Does it generalize well on both BPR and BCE loss? (RQ1(b)): Figure 4 shows how our method using MF as the backbone performs when applied with BPR or BCE loss on MovieLens-10M and Netflix. Overall, subtracting user conformity bias and item popularity bias in the testing stage provides substantially better performance than the models using vanilla user and item embeddings. This observation validates our view that the optimization of BPR and BCE incorporates the user conformity bias and the item popularity bias into user and item embeddings, respectively, making the recommendation results deviating from users’ true interest. The proposed post-hoc design is an elegant and effective way to remove bias.

2) Is our method more efficient than state-of-the-art baselines? (RQ1(c)): We investigate the efficiency of our proposed method by comparing the running time on the same device against the two state-of-the-art methods, i.e., DICE and MACR, with MF and LGN backbones. As shown in Table IV, we observe that our method achieves comparable running time with classical MF and LGN methods, and it is significantly more efficient than DICE and MACR. With special causal learning design and direct supervision on disentanglement, DICE takes twice more time than our proposed method due to the additional cost of parameter updating. MACR includes additional modules to reconstruct user and item frequencies, and is therefore less efficient. Another interesting side observation is that DICE always suffers from excessive computation time compared with MACR. We conjecture that the additional two rule-based strategies of DICE are partially responsible for the excess time consumed.
In this study, we investigate whether our method performs better when removing the user conformity and item popularity bias. Specifically, we compare our method without user norm (i.e., MF w/o norm and LGN w/o norm), where normalization on user embedding is removed; our method with α1 = 0, where we just simply drop the user conformity embedding term in the inference; our method with α2 = 0 that drops the item popularity embedding term.

The results in Table V show that removing any component would lead to worse performance. It indicates that all three components contribute to improving the model performance under non-IID circumstances. Omitting user embedding normalization would reduce accuracy in all metrics on both Movielens-10M and Adressa datasets. This validates the usefulness of normalizing user embedding magnitudes in the training stage, i.e., promoting items with fewer interactions benefits recommendation performance. It is clear that compared with the variant without item popularity bias, the model performs much worse when removing the user conformity bias in most cases. One possible reason is that user has more impact than item for these two datasets.

4) How does our method mitigate the popularity bias? (RQ2(a)): In this study, we investigate whether our method can remove the effect of popularity bias by the methods, i.e., MF (LGN), Our_MF (Our_LGN) w/o norm, and Our_MF (Our_LGN) on Movielens-10M. For fine-grained debias analysis, we first sort items in inverse popularity order. We then divide items into five groups, each of the first four groups accounts for 5% of items, and we put the rest into the last group. Group 1 represents the most popular item group, and Group 5 represents the least popular item group. For each group, we calculate the average recommended frequency and recall@20. We show the recommended frequency of each group by methods in Figure 5 and recall in Figure 6.

Conventional recommendation methods have the most significant recommended frequency and perform best in group 1 but yield a significant performance drop in group 2 and further drops in other groups. It indicates that conventional recommender systems tend to recommend popular items to unrelated users due to user conformity and popularity bias. In contrast, conventional recommender systems with our proposed post-hoc method have less recommended frequency and performance in the most popular group but achieve a significantly higher recommended frequency and better recommendation performance from group 2 to group 4. It shows that our proposed method effectively reduces user conformity and item popularity bias.

In group 5, all methods achieve similar recommended frequency and performance. The possible reason is that these least popular items rarely appear in the training set so that the representation of these items cannot be learned effectively. In some cases, our method achieves less recommended frequency but higher recall than our method w/o norm. It means that our method can still more accurately recommend items to users.

5) How does our method’s performance change along with intervention proportion of test data? (RQ2(b)): In previous experiments, all methods are tested on the 100% intervened test data. This study investigates how methods perform as the proportion of intervened test data changes. We evaluate MF and our method (MF) with five different proportions of intervened test data, i.e., 0%, 50%, 75%, 90%, 100%. 0% means training and test data have the same distribution. In some cases, our method achieves less recommended frequency but higher recall than our method w/o norm. It means that our method can still more accurately recommend items to users.
10% intervened test data, which verifies the effectiveness of
our method under non-IID circumstances.

6) Effect of the hyper-parameters of subtrahends: Our post-
hoc method has two hyper-parameters $\alpha_1$ and $\alpha_2$ for
the user conformity subtrahend and the item popularity subtrahend
respectively. We experiment with our method using MF as
the backbone on Movielens-10M to investigate their effect.
Figure 8 shows how our method’s performance (recall@20)
changes as $\alpha_1$ and $\alpha_2$ change from 0 to 2 over the interval.

As we can see, the model performs well when $\alpha_1$ is
between 0.4 to 0.6 and performs increasingly better while
increasing $\alpha_2$ in most cases. It suggests that the appropriate
amount of mitigating user conformity bias and item popular-
ity bias benefits the recommendation system under non-IID
circumstances. Large $\alpha_1$ yields a drop in performance while
large $\alpha_2$ improves the recommendation performance, which
indicates that excessive user conformity debiasing hurts the
performance. One possible reason is that user normalization
overly constrains the magnitude of user embeddings.

![Figure 8. Effect of $\alpha_1$ and $\alpha_2$ in our method (MF) for mitigating the user conformity bias and the item popularity bias respectively.](image)

VI. RELATED WORK

In recent years, recommendation systems have achieved
great success due to the merit of deep learning methods, most
of which aim at the development of novel machine learning
models to fit user behavior data. When it comes to real-
world scenarios, the distribution of user and item interaction is
inevitably long-tailed, making popularity bias a long-standing
challenge [22]–[24].

The first core idea in mitigating popularity bias is to balance
the data distribution such that popular and unpopular items are
equally important in the training process. Inverse propensity
weighting (IPW) [8], [25] is widely used to assign weights
inversely proportional to item popularity in the empirical
risk of a RS model [26]. While such reweighting methods
achieve significant improvement for under-represented items,
they come with a high potential risk of overfitting and suffer
from high variance [27], [28]. Recent work [29]–[32] studies
this problem from a fairness perspective [33], [34], and enforce
the algorithm to produce a fair allocation of exposure based
on merit. Other strategies following the first pipeline is to transfer
the knowledge learned from popular groups to unpopular
groups [35]. Such methods often introduce extra noise to
unpopular items and modify the initial dataset distribution.
Recent work [36] investigates the bias from the perspective
of causal inference and proposes a counterfactual reasoning
method using Do-calculus [37]. They propose a different
causal graph based on prior knowledge, however, the essence
is to re-balance the distribution for popular and unpopular
groups with a causal language.

Another line of research is to preserve specific user and
item information intentionally, with the goal to make the
unpopular items more representative [10]. With auxiliary text
information, [10] an autoencoder layer [38], [39] is added
while learning user and item representations with text-based
Convolutional Neural Networks. Latest work [21] regards a
personalized recommendation system as a treatment policy
utility maximization problem and proposes a domain adap-
tation method named CausE. [40] propose a meta-learning
strategy and transfer unbiased data representation from a uni-
form logged data. However, the availability and high-quality
of auxiliary dataset is the bottleneck for such approaches.

The third categories try to solve this problem through rank-
ing adjustment [41]–[43]. These approaches result in a trade-
off between the recommendation accuracy and the coverage
of unpopular items. They typically suffer from accuracy drop
due to give consideration to the long-tail in a brute manner

Different from previous method, in this paper we do not
focus on balancing the distribution of popular and unpopular
items groups or making unpopular items more representative.
We provide a new insight that unbalancing between the
positive and negative gradients of the same item incurs the
popularity bias. Some related works [44] discuss the gradient
vanish of unpopular items from a gradient perspective in im-
licit recommendation system. However, they still focus on the
unbalanced distribution between popular and unpopular item
groups, which actually fall into the former three categories.

VII. CONCLUSION

In this paper, we propose a simple, efficient, and general
framework to mitigate popularity bias from a gradient (first-
order information) perspective. In our approach, we record
accumulated gradients of users and items as popularity bias
during the training stage. Then we address popularity bias by
gradient-level calibration during the testing stage as a post-hoc
debiasing strategy. We implement our proposed method over
MF and LGN on four real-world datasets. Extensive experi-
ments demonstrate that our approach outperforms competitive
baselines, e.g., DICE and MACR. As part of the future work,
we will consider applying our approach to more complex
recommendation tasks, such as sequential recommendation
and conversational recommendation.

VIII. ACKNOWLEDGEMENT

This research was partially supported by the National
Science Foundation (NSF) via the grant numbers: 2040950,
2006889, 2045567.

446
