



LightAD: accelerating AutoDebias with adaptive sampling

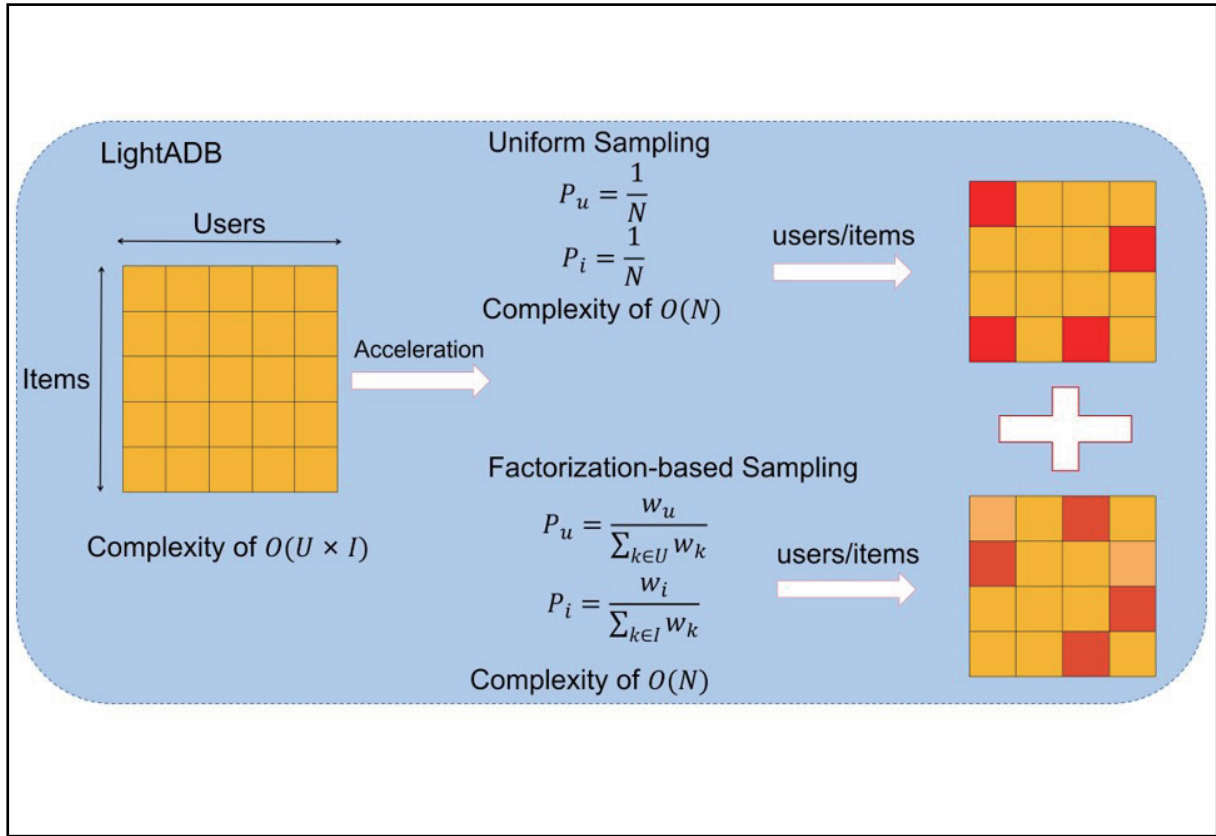
Yang Qiu, Hande Dong, Jiawei Chen , and Xiangnan He

School of Information Science and Technology, University of Science of Technology of China, Hefei 230027, China

 Correspondence: Jiawei Chen, E-mail: cjwustc@ustc.edu.cn

© 2024 The Author(s). This is an open access article under the CC BY-NC-ND 4.0 license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

Graphical abstract




The sampling framework of LightAD.

Public summary

- Biases can significantly affect the performance of the recommended model. AutoDebias is a generic and effective solution in tackling biases, but it suffers from a huge computational cost. This study proposes an improved framework named LightAD that can reduce the computational cost of the AutoDebias by sampling.
- Through theoretical analysis, we prove the unbiasedness of the LightAD. And compared with the simple uniform sampling, our sampling strategy can reduce the sampling variance and accelerate the model convergence.
- The experimental results on three datasets show that our sampling strategy achieves similar performance to AutoDebias framework, and greatly improves the algorithm efficiency.

LightAD: accelerating AutoDebias with adaptive sampling

Yang Qiu, Hande Dong, Jiawei Chen , and Xiangnan He

School of Information Science and Technology, University of Science of Technology of China, Hefei 230027, China

✉ Correspondence: Jiawei Chen, E-mail: cjwustc@ustc.edu.cn

© 2024 The Author(s). This is an open access article under the CC BY-NC-ND 4.0 license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).



Cite This: *JUSTC*, 2024, 54(4): 0405 (9pp)



Read Online

Abstract: In recommendation systems, bias is ubiquitous because the data are collected from user behaviors rather than from reasonable experiments. AutoDebias, which resorts to metalearning to find appropriate debiasing configurations, i.e., pseudolabels and confidence weights for all user-item pairs, has been demonstrated to be a generic and effective solution for tackling various biases. Nevertheless, setting pseudolabels and weights for every user-item pair can be a time-consuming process. Therefore, AutoDebias suffers from an enormous computational cost, making it less applicable to real cases. Although stochastic gradient descent with a uniform sampler can be applied to accelerate training, this approach significantly deteriorates model convergence and stability. To overcome this problem, we propose LightAutoDebias (short as LightAD), which equips AutoDebias with a specialized importance sampling strategy. The sampler can adaptively and dynamically draw informative training instances, which results in better convergence and stability than does the standard uniform sampler. Several experiments on three benchmark datasets validate that our LightAD accelerates AutoDebias by several magnitudes while maintaining almost equal accuracy.

Keywords: recommendation; bias; debias; sampling

CLC number: TP181

Document code: A

1 Introduction

In the era of information explosion, recommender systems (RSs) that connect users with the right items have been recognized as the most effective means to alleviate information overload^[1]. Existing recommendation methods mainly focus on developing a machine learning model to better fit the collected user behavior data^[2,3]. However, because the behavior data in practice are observational rather than experimental, various biases occur. For example, user behaviors are subject to what was exposed to the users (aka, exposure bias) or what were the public opinions (aka, conformity bias), deviating from reflecting the real interests of users. Consequently, blindly fitting the RS model without tackling the bias issues will lead to unacceptable performance^[4,5].

To alleviate the unfavorable effect of biases, a surge of recent work has been devoted to debiasing solutions. For example, the inverse propensity score (IPS) was proposed to re-weight training samples for expectation-unbiased learning^[6,7]; Data imputation was explored to fill missing values^[8,9]; Knowledge distillation was also used to bridge the models trained on biased data and unbiased data. More recently, AutoDebias^[10] proposed to optimize objective function on unbiased data has applied a metalearning technique, which has been demonstrated to be a generic and effective solution for recommendation debiasing. AutoDebias gives both pseudolabels and confidence weights for each user-item pair and is flexible enough to address various biases and outperform previous work by a large margin.

Although effective at managing diverse biases, AutoDebias suffers from serious inefficiency. Specifically, the training

of AutoDebias involves calculations for each user-item pair. As such, the complexity of AutoDebias is linear with the product of the number of users and items, which can easily scale to one billion or even larger. This crucial defect severely limits the applicability and potential of AutoDebias. Although stochastic gradient descent with a uniform sampler can be employed for acceleration, this treatment significantly deteriorates model convergence and stability, leading to inferior recommendation performance. Hence, how to accelerate AutoDebias without reducing its accuracy is still an open problem.

To this end, in this work, we propose LightAutoDebias (short as LightAD), which equips AutoDebias with a carefully designed adaptive sampling strategy. We make three important improvements: (i) We adopt a dynamic sampling distribution that favors the informative training instances with large confidence weights. Further theoretical analysis proves that such a distribution could reduce the sampling variance and accelerate convergence. (ii) We develop a customized factorization-based sampling strategy to support fast sampling from the above dynamic distribution. (iii) Because the model may not reliably estimate the confidence weights in the early training stage, we propose a self-paced strategy that adopts uniform sampling at the beginning while gradually tilting toward factorization-based sampling along the training process.

To summarize, the contributions of this work are as follows:

- We identify the serious inefficiency issue of AutoDebias, which is imperative to be conquered.
- We propose a dynamic and adaptive sampler for AutoDebias, which yields provably better convergence and

stability than those of the standard uniform sampler.

- We conduct extensive experiments on three benchmark datasets, validating that our LightAD accelerates AutoDebias by several magnitudes while maintaining almost equal recommendation accuracy.

The remainder of this paper is organized as follows. We first provide a review of related works in Section 2. We then define the problem and recap AutoDebias in Section 3. We elaborate upon the proposed method in detail in Section 4. After that, we theoretically demonstrate the unbiasedness and effectiveness of our approach in Section 5. The experimental results and discussion are presented in Section 6. Finally, we conclude the paper and present some directions for future work are presented in Section 7.

2 Related work

The fact that biases exist in practical RSs and can strongly compromise the performance of RSs has been proven by many scholars. Since we study how to apply unbiased sampling strategies to improve the efficiency issues in this paper, we first make a general classification of biases in data and review some related work on tackling them. Then, we included some specific sampling methods in the debiasing area.

2.1 Biases in recommendation

Intuitively, biases in an RS indicate that models lose the ability to provide accurate recommendations for users. In general, we can roughly divide them into the following categories:

Selection bias. Selection bias occurs when users are free to choose which items to rate, thus the collected ratings are not a representative sample of all the ratings^[11]. Intuitively, the data we used in RS are not missing at random. The research of Marlin et al.^[12] fully proved the presence of selection bias. Schnabel et al.^[6] considered introducing used inversed propensity scores (IPSSs) to reweight the observed data. Steck et al.^[13] designed a novel unbiased metric named ATOP to remedy selection bias. Considering that users can freely rate items they like, Hernández et al.^[8] proposed imputation-based methods that directly impute the missing entries with pseudolabels and then weights down the contribution of these missing ratings.

Conformity bias. Conformity bias occurs as users tend to align themselves with others in the group regardless of whether it goes against their own will^[11]. In other words, our rating of some products may succumb to public opinion and therefore not reflect our real interests. Krishnan et al.^[14] conducted comparative experiments suggesting that social influence bias is an important factor in RS. Liu et al.^[15] took conformity considered and directly used rating data behavior as an important feature to quantify conformity bias. In addition, Tang et al.^[16] and Hao et al.^[17] leveraged social factors more directly to produce the final prediction and introduce specific parameters to eliminate the impact of conformity bias.

Exposure bias. Exposure bias exists when users are only exposed to a small portion of the items provided^[11]. Exposure bias is readily comprehensible as the unobserved data can be attributed to users not liking the item or simply not seeing the item. One simple and straightforward strategy to address exposure bias is to consider all noninteraction data as negative

samples and specify their confidence. Hu et al.^[18] put forward the classic WMF model in which the unobserved data are assigned lower weights. Similarly, Pan et al.^[19] went a step further and specified the data confidence with examined user behavior. Another way to address exposure bias is to design an exposure-based probabilistic model that is able to capture whether a user has been exposed to an item. Liang introduced the EXMF model of users' activity through an exposure variable and proposed a generative process of implicit feedback. Chen et al.^[20] assumed that users frequently come into contact with specific communities and thus modelled confidence weights using a community-based strategy.

Position bias. Position bias emerges as users are inclined to interact with items in a higher position on the recommendation list^[11]. When dealing with position bias, many researchers assuming that users' click behavior occurs if and only if an item is relevant and examined by users^[21, 22]. Another conventional strategy is to apply a model and its variants^[23–25], which assumes that users examine items from the top to the bottom of the recommendation list. Hence, the click behavior relies only on the relevance of all the items that displayed to users.

Popularity bias. Popularity bias occurs when popular items are recommended more often than their popularity^[11]. Due to the common long tail phenomenon in recommendation data; recommendation models usually pay more attention to popular items and hence give higher scores to popular items than their real value. Kamishima et al.^[26] used a mean-match regularizer in their models to offset popularity bias. Zheng et al.^[27] applied counterfactual reasoning to address the popularity bias and assumed that users' click behavior depends mainly on their interest and items' popularity. Krishnan et al.^[28] introduced adversarial learning in RS to seek a trade-off between the popular item-biased reconstruction objective and the accuracy of long-tail item recommendations.

2.2 Sampling

In addition to the methods mentioned above, sampling is a promising debiasing method that has been studied by many scholars. Specifically, we can use various sampling strategies to determine which data are used to update the parameters and how often in the training of our models. Steffen et al.^[29] employed the uniform negative sampler as the basic strategy to improve efficiency. Several neural-based methods^[30–32] also consider using negative sampler as an efficient way to increase the model performance. In response to exposure bias, Yu et al.^[33] proposed to over sampled popular negative samples because they had a greater chance of being exposed. Nevertheless, these samplers follow a predetermined sampling distribution which may dynamically change according to the model's state in each iteration; increasing the likelihood generating low-quality samples. Dae et al.^[34], Steffen et al.^[35], and Ding et al.^[36] proposed an adaptive negative sampler, for the purpose of over sampling "difficult" instances to increase learning efficiency. One disadvantage of these heuristic methods is that they fail to capture the real negative instances. To address this problem, Ding et al.^[37] explored leveraging side information to enhance the studied the performance of samplers; Chen et al.^[38] took advantage of

social information in their sampling strategy.

3 Preliminary

In this section, we first formulate the task of recommendation systems and then expose the nature of bias from the point of view of risk discrepancy. Finally, we analyze the temporal and spatial complexity of AutoDebias to demonstrate the challenges we face in deploying this algorithm.

3.1 Problem definition

Suppose we have a recommender system with user set \mathcal{U} (including n users) and item set \mathcal{I} (including m items). Let $r \in \mathcal{R}$ be the users' feedback on items, which can be binary (implicit feedback) or the numerical ratings (explicit feedback). Let D_T denote the historical user behavior, which can be seen as a set of triplets $\{(u_k, i_k, r_k)\}_{1 \leq k \leq |D_T|}$ generated from an unknown training distribution $p_T(u, i, r)$ over the space $\mathcal{U} \times \mathcal{I} \times \mathcal{R}$. For convenience, we use $\delta(\cdot, \cdot)$ to represent the predefined error function, e.g., the MSE, cross-entropy, and Hinge loss. The objective of recommendation system is used to learn a proper function f_θ from D_T that minimizes the following true risk:

$$L(f) = \mathbb{E}_{p_U(u, i, r)} [\delta(f(u, i), r)], \quad (1)$$

where $p_U(u, i, r)$ denotes the ideal unbiased data distribution for model evaluation. Regrettably, p_U is not accessible in most cases. Instead, training is often performed on the training dataset D_T , which optimizes the following empirical risk:

$$\hat{L}_T(f) = \frac{1}{|D_T|} \sum_{k=1}^{|D_T|} \delta(f(u_k, i_k), r_k). \quad (2)$$

According to the PAC learning theory^[39], if and only if $\hat{L}_T(f)$ is an unbiased estimator, i.e., $\mathbb{E}_{p_T}[\hat{L}_T(f)] = L(f)$, the learned model will be approximately optimal as the training data size increases. However, since the collected behavior data often contain bias, the training data distribution p_T cannot be consistent with the unbiased test one p_U . In other words, the unbiasedness cannot hold in many cases. Therefore, directly training a recommendation model on D_T with Eq. (2) without considering the inherent biases would easily lead to inferior results.

3.2 AutoDebias brief

To eliminate the distribution discrepancy, Ref. [10] puts forward AutoDebias, a general proposed a debiasing framework, which aims at addressing various biases. The main idea of AutoDebias transforms the aforementioned empirical risk functions into:

$$\hat{L}_T(f|\phi) = \frac{1}{|D_T|} \sum_{k=1}^{|D_T|} w_k^{(1)} \delta(f(u_k, i_k), r_k) + \sum_{u \in \mathcal{U}, i \in \mathcal{I}} w_{ui}^{(2)} \delta(f(u, i), m_{ui}), \quad (3)$$

where AutoDebias imputes the pseudolabels m_{ui} for each user-item pair, and gives diverse confidence weights $w_k^{(1)}$ for each training instance. When the set of hyperparameters $\phi \equiv \{w^{(1)}, w^{(2)}, m\}$ is properly specified, $\hat{L}_T(f|\phi)$ can be an unbiased estimator, even if the training dataset D_T contains any

type of bias.

AutoDebias also devises a metlearning-based strategy to learn the appropriate ϕ . Specifically, they first propose reparameterizing ϕ with a concise linear meta model:

$$\begin{aligned} w_k^{(1)} &= \exp(\varphi_1^T [\mathbf{e}_{u_k} \circ \mathbf{e}_{i_k} \circ \mathbf{e}_{r_k}]), \\ w_{ui}^{(2)} &= \exp(\varphi_2^T [\mathbf{e}_u \circ \mathbf{e}_i \circ \mathbf{e}_{O_{ui}}]), \\ m_{ui} &= \sigma(\varphi_3^T [\mathbf{e}_{r_{ui}} \circ \mathbf{e}_{O_{ui}}]), \end{aligned} \quad (4)$$

where \mathbf{e}_u , \mathbf{e}_i , \mathbf{e}_r , and $\mathbf{e}_{O_{ui}}$ denote the one-hot vectors of the user u , item i , feedback r and observation O_{ui} , respectively. This process can reduce the number of parameters and encode useful debiasing information. They then leveraged a meta-learning strategy to optimize ϕ by utilizing a small set of uniform data. Specifically, AutoDebias updates ϕ and f_θ in an alternative fashion: making an assumed update of θ with $\theta'(\phi) = \theta - \eta_1 \nabla_\theta \hat{L}_T(f_\theta|\phi)$; validating the performance of θ' on the uniform data with $\hat{L}_U(f_\theta)$, i.e., $\hat{L}_U(f_\theta) = \frac{1}{|D_U|} \sum_{i=1}^{|D_U|} \delta(f_\theta(u_i, i_i), r_i)$,

which in turn gives a feedback signal (gradient) to update the meta model ϕ : $\varphi \leftarrow \varphi - \eta_2 \nabla_\varphi \hat{L}_U(f_{\theta'(\phi)})$; given the updated ϕ , updating θ actually: $\theta \leftarrow \theta - \eta_1 \nabla_\theta \hat{L}_T(f_\theta|\phi)$.

3.3 Complexity analysis of AutoDebias

Now, we conduct time complexity analyses of AutoDebias. The time complexity mainly comes from the following two parts:

Calculation on observed training instances. As depicted by the first part of the Eq. (3), the calculation only involves the instances in D_T . As such, the complexity of this part is linear with the number of the observed training instances, i.e., $O(|D_T|)$.

Calculation on imputed training instances. As depicted by the second part of the Eq. (3), the complexity involves the calculation over all user-item pairs, which is $O(nm)$. The number of users and items can be easily scaled to millions or more in practice, AutoDebias would be computationally infeasible.

Considering the sparse nature of real-world recommendation data, the second part has become the bottleneck of AutoDebias. A naive solution for acceleration employs stochastic gradient descent and a uniform sampler, where the gradient can be quickly calculated on the sampled portion of the dataset. However, we find that this treatment would significantly deteriorate model convergence and stability, leading to inferior recommendation performance. In fact, not all user-item pairs are equally important in model training. We observe fairly scattered $w^{(2)}$ in practice. The uniform sampler easily samples the uninformative training instances with small $w^{(2)}$, which limits contribution to the training process. As such, we need a new sampler that can adaptively draw informative training instances.

4 Method

In this section, we present LightAD which equips AutoDebias with a carefully designed adaptive sampling strategy. LightAD adopts an uneven sampling distribution and factorization-based efficient sampling strategy. We will

detail LightAD in the following parts.

4.1 Fast informative sampler

As mentioned before, the second part of Eq. (3) has become the bottleneck of AutoDebias. Here we leverage a sampler to accelerate the training, where the gradient can be estimated from a much smaller sampled training dataset. Formally, the learning with the sampler can be formulated as follows:

$$S \sim \text{Multinomial}(N, p_s(u, i)), \quad (5)$$

which draws a small set of users and items, i.e., S , with the sample size N and the distribution $p_s(u, i)$. In this way, a recommendation model can be trained efficiently with sampled instances, which can be expressed as:

$$\hat{L}_r(f|\phi) = \frac{1}{|D_T|} \sum_{k=1}^{|D_T|} w_k^{(1)} \delta(f(u_k, i_k), r_k) + \frac{1}{|S|} \sum_{u,i \in S} \delta(f(u, i), m_{ui}), \quad (6)$$

where the confidence weights $w_{ui}^{(2)}$ are offset by the sampling distribution. Now, the question lies in which sampling distribution to use and how to implement fast sampling.

Importance-aware sampling distribution. Note that different data may have different confidence weights $w^{(2)}$. This suggests that the data sampling distribution is highly important since it determines which data are used to update parameters and how often it is. Intuitively, the informative data that have a larger $w^{(2)}$ should be sampled with a larger sampling probability. The reason lies in the fact that they would bring larger gradients and make a greater contribution to training, which could speed up model convergence. In fact, our theoretical analyses presented in Section 5.2 prove that sampling with the distribution $p_s(u, i) \propto w_{ui}^{(2)}$ can reduce the sampling variance.

Fast factorization-based sampling. Now the question lies in how to perform fast sampling from the distribution $p_s(u, i) \propto w_{ui}^{(2)}$. Direct sampling from $p_s(u, i)$ would be highly time-consuming, as it involves a large instance space. Moreover, the sampling distribution would evolve as the training proceeds. To address this problem, we propose a subtle fast factorization-based sampling algorithm. The merit of the algorithm lies in the decomposition nature of the hyperparameters $w_{ui}^{(2)}$. In fact, we have:

$$w_{ui}^{(2)} = \exp(\varphi_2^T [e_u \circ e_i]) = \exp(\varphi_2^T [e_u]) * \exp(\varphi_2^T [e_i]) \equiv w_u^{(2)} * w_i^{(2)}, \quad (7)$$

where e_u and e_i denote the one-hot vectors of user ID u and item ID i . The $w_{ui}^{(2)}$ can be decomposed as the product of the item-based weight $w_i^{(2)}$ and user-based weight $w_u^{(2)}$. As such, we could separate the sampling of users and items as follows:

$$\begin{aligned} E_{S_1, S_2}[\hat{L}_r(f|\phi)] &= \frac{1}{|D_T|} \sum_{k=1}^{|D_T|} w_k^{(1)} \delta(f(u_k, i_k), r_k) + E_{S_1} \left[\frac{\alpha \sum_{u,i} w_{ui}^{(2)}}{|S_1|} \sum_{u,i \in S_1} \delta(f(u, i), m_{ui}) \right] + E_{S_2} \left[\frac{(1-\alpha)nm}{|S_2|} \sum_{u,i \in S_2} w_{ui}^{(2)} \delta(f(u, i), m_{ui}) \right] = \\ &= \frac{1}{|D_T|} \sum_{k=1}^{|D_T|} w_k^{(1)} \delta(f(u_k, i_k), r_k) + \frac{\alpha \sum_{u,i} w_{ui}^{(2)} |S_1|}{|S_1|} E_{u,i \sim p_s} [\delta(f(u, i), m_{ui})] + \frac{(1-\alpha)nm |S_2|}{|S_2|} E_{u,i \sim U} [w_{ui}^{(2)} \delta(f(u, i), m_{ui})] = \end{aligned}$$

$$\begin{aligned} S_u &\sim \text{Multinomial}(N, p_u), \\ S_i &\sim \text{Multinomial}(N, p_i), \end{aligned} \quad (8)$$

where $p_u \propto w_u^{(2)}$ and $p_i \propto w_i^{(2)}$. In this way, we can find that the sampling distribution for a user-item pair is proportional to $w_{ui}^{(2)}$, without requiring direct sampling from the large user-item space. As such, the sampling complexity could be reduced from $O(nm)$ to $O(n+m)$.

4.2 Self-paced sampling strategy.

Note that in the early training stage, the model may not yield a reliable estimate of the confidence weights. Directly learning a model from an informative sampler may lead to suboptimal results. To address this problem, we propose a self-paced strategy that adopts a uniform distribution at the beginning while gradually alters the distribution as the training proceeds. Formally, we adopt the following sampling strategy:

$$\begin{aligned} S_1 &\sim \text{Multinomial}(\alpha N, p_s(u, i)), \\ S_2 &\sim \text{Multinomial}((1-\alpha)N, U(u, i)), \end{aligned} \quad (9)$$

where $U(u, i)$ denotes the uniform distribution. α controls the ratio of the instances sampled from the informative sampler and the uniform sampler, which would gradually increase as the training proceeds. In this work, we simply set α with $\min(e/\beta, 1)$, where e denotes the current epoch, and β controls the increase rate of α , while leaving other choices as future work. After obtaining S_1 and S_2 , the model can be updated with the following objective functions:

$$\begin{aligned} \hat{L}_r(f|\phi) &= \frac{1}{|D_T|} \sum_{k=1}^{|D_T|} w_k^{(1)} \delta(f(u_k, i_k), r_k) + \frac{\alpha \sum_{u,i} w_{ui}^{(2)}}{|S_1|} \cdot \\ &\quad \sum_{u,i \in S_1} \delta(f(u, i), m_{ui}) + \frac{(1-\alpha)nm}{|S_2|} \sum_{u,i \in S_2} w_{ui}^{(2)} \delta(f(u, i), m_{ui}). \end{aligned} \quad (10)$$

5 Theoretical analyses

The former section elaborates how LightAD accelerates AutoDebias. In this section, we conduct theoretical analyses to answer the following questions: Does the unbiasedness still hold with our proposed sampling strategy? Does the proposed informative sampler reduce the sampling variance?

5.1 Unbiasedness of LightAD

It has been proven that the empirical risk function of AutoDebias can be an unbiased estimator of the true risk. Now we connect the objective of LightAD (Eq. (10)) with AutoDebias, which reveals the unbiasedness of LightAD. In fact, we have:

$$\frac{1}{|D_T|} \sum_{k=1}^{|D_T|} w_k^{(1)} \delta(f(u_k, i_k), r_k) + \alpha \sum_{u,i} \frac{w_{ui}^{(2)} \sum_{u,i} w_{ui}^{(2)}}{\sum_{u,i} w_{ui}^{(2)}} \delta(f(u, i), m_{ui}) + (1 - \alpha) nm \sum_{u,i} w_{ui}^{(2)} \delta(f(u, i), m_{ui}) = \hat{L}_s(f|\phi). \quad (11)$$

From the above mathematical deduction, we can safely draw the conclusion that the objective of LightAD is indeed to use an unbiased estimator of AutoDebias and the ideal true risk.

5.2 Variance reduction

How does the conventional uniform sampler perform? Does the proposed informative sampler reduce the variance? In this section, we aim to provide a theoretical answer to these problems.

Let us first formulate the variance of the estimated objective with the sampler. In fact, we have the following upper-bound of the variance:

$$\begin{aligned} \text{Var}_s[\hat{L}_s] &= \frac{1}{|S|} \text{Var}_{p_s} \left[\frac{w_{ui}^{(2)}}{p_s(u, i)} \delta(f(u, i), m_{ui}) \right] \leq \\ &\frac{1}{|S|} \left(\sum_{u,i} \frac{\text{Var}_{ui}[w_{ui}^{(2)}] + E_{ui}[w_{ui}^{(2)}]^2}{p_s(u, i)} \right) \times E_{p_s}[\delta(f(u, i), m_{ui})]^2 - \\ &\frac{1}{|S|} E_{p_s}[w_{ui}^{(2)} \delta(f(u, i), m_{ui})]^2. \end{aligned} \quad (12)$$

For a uniform sampler, the variance of the objective is subject to the variance of weights $w_{ui}^{(2)}$, which is unsatisfactory. As the training proceeds, we observe a diverse distribution of $w_{ui}^{(2)}$ with large variance. The gradient from \hat{L}_s with a uniform sampler fluctuates heavily, making the training process highly unstable.

To address this problem, we should choose a better sampling distribution to reduce the upper bound of the estimation variance. In fact, we have:

$$\begin{aligned} \sum_{ui} \frac{(w_{ui}^{(2)})^2}{p_s(u, i)} &= \sum_{ui} \frac{(w_{ui}^{(2)})^2}{p_s(u, i)} \sum_{ui} p_s(u, i) \geq \\ &\left(\sum_{ui} \sqrt{\frac{(w_{ui}^{(2)})^2}{p_s(u, i)}} \sqrt{p_s(u, i)} \right)^2 = E_{ui}[w_{ui}^{(2)}]^2, \end{aligned} \quad (13)$$

where we employ the Cauchy inequality, and the equation holds if and only if $p_s(u, i) \propto w_{ui}^{(2)}$. This means that for the informative sampler whose sampling distribution is proportional to the confidence weights, the estimation variance achieves the lowest upper bound. The estimation variance is only subject to the mean of the confidence weights and independent of the variance of the confidence weights. The informative sampler would bring better stability and convergence.

6 Experiments

In this section, we first describe the experimental settings, and then present the detailed experiments with the purpose of answering the following questions:

RQ1: How does our LightAD perform compared with AutoDebias in recommendation performance?

RQ2: How does the proposed LightAD accelerate AutoDebias?

RQ3: How does the hyperparameter β affect the recommendation performance?

RQ4: How does LightAD perform when dealing with the simultaneous presence of various biases?

6.1 Experimental setup

We conduct our experiments with two publicly accessible datasets and one synthetic dataset: Yahoo!R3, Coat, and Simulation. The concrete statistics of the three datasets are summarized in Table 1.

Yahoo!R3. This dataset contains two types of ratings collected from two sources. The first source collects ratings of users' normal interactions with music in Yahoo! services (i.e., users pick and rate items according to their own preferences), which can be considered as a stochastic logging policy. Thus, this type of data is obsessed with bias. The second kind of data is ratings from randomly selected songs collected during an online survey (i.e., uniform logging policy). Approximately, it can be regarded as unbiased and reflects the real interest of users.

Coat. This is a dataset collected by Ref. [7], which simulates customers shopping behaviors for coats in an online service. In this dataset, users were first given a web-shop interface and then asked to select the coat they wanted to buy most. Shortly afterwards, they were requested to rate 24 of the self-selected coats and 16 of the randomly picked ones on a five-point scale. Therefore, this dataset consists of self-selected ratings and the uniformly selected ratings which can be regarded as biased data and unbiased data respectively.

Simulation. A great advantage of AutoDebias is that it can handle multiple biases simultaneously, which means good universality in complex scenario. We need to verify that this universality also persists in our model. Since there is no public dataset that satisfies such a scenario, we borrow from AutoDebias and adopt a synthetic dataset Simulation where both position bias and selection bias are taken into account. Unlike the previous two datasets, this one contains the position information of items when the interaction occurs. In short, this dataset also includes a large amount of biased data and a relatively small amount of unbiased data. The rationale for the way of data synthesis can be found in Refs. [40, 41].

To be consistent with AutoDebias, we follow the experimental setup with them in training, validating, and testing. Specifically, we use the biased data D_T to train our base model and uniform data D_U to assist in debiasing and testing. Given the paucity of uniform data, we split D_U into three parts: 5% to assist in training hyperparameters D_{ut} , 5% for validation to tune the hyperparameters D_{uv} , and the

Table 1. Statistics of the datasets.

Dataset	Users	Items	Training	Validation	Test
Coat	290	300	6960	232	4176
Yahoo!R3	15400	1000	311704	2700	48600
Simulation	500	500	12500	3750	67500

remaining 90% for evaluating the model D_{uc} . Additionally, we binarize the rating in three datasets with threshold 3, i.e., we treat the rating values that are larger than 3 as positive feedback and label them with $r = 1$.

Evaluation protocols. The evaluation metrics in this work are the negative logarithmic loss (NLL), the area under the ROC curve (AUC), and normalized discounted cumulative gain (NDCG):

NLL. This metric evaluates the performance of the predictions:

$$NLL = -\frac{1}{|D_{ve}|} \sum_{(u,i,r) \in D_{ve}} \log(1 + e^{-r \cdot f_{\theta}(u,i)}), \quad (14)$$

where D_{ve} denotes the validation set D_{uv} or the test set D_{uc} .

AUC. This metric evaluates the performance of rankings:

$$AUC = \frac{\sum_{(u,i) \in D_{ve}^+} \widehat{Rank}_{u,i} - (|D_{ve}^+| + 1)(|D_{ve}^+|)/2}{(|D_{ve}^+|) * (|D_{ve}| - |D_{ve}^+|)}, \quad (15)$$

where $|D_{ve}^+|$ denotes the number of positive data in D_{ve} , and $\widehat{Rank}_{u,i}$ denotes the rank position of a positive feedback (u,i) .

NDCG@k. This metric evaluates the quality of recommendation through discounted importance based on ranking position:

$$\begin{aligned} DCG_u @ k &= \sum_{(u,i) \in D_{ve}} \frac{\mathbf{I}(\widehat{Rank}_{u,i} \leq k)}{\log(\widehat{Rank}_{u,i} + 1)}, \\ NDCG @ k &= \frac{1}{|U|} \sum_{u \in U} \frac{DCG_u @ k}{IDCG_u @ k}, \end{aligned} \quad (16)$$

where $IDCG_u @ k$ is a normalizer that ensures NDCG is between 0 and 1.

We implement our experiment with Pytorch and the code is available at <https://github.com/Chris-Mraz/LightAD>. We perform grid search to tune the hyperparameters for all candidate methods on the validation set.

6.2 Performance comparison (RQ1)

Baselines. To demonstrate the effectiveness of our proposed

sampled method, we use matrix factorization (MF)^[42] as the backbone and the following models as our baselines:

Matrix factorization (MF)^[42]. MF is one classical model in RS. In this model, the preference of user u for item i can be formulated as $R_{u,i} = U_u^T V_i + u_i + u_j + \mu$. In our experimental setup, we train the MF model with biased, uniform, and combined dataset, i.e., D_r , D_u , and $D_r + D_u$, respectively.

Inverse propensity score (IPS)^[6]. IPS is a counterfactual-based recommendation method that estimates the propensity score via naive bayes.

Doubly robust (DR)^[43]. DR combines the ideas of the IPS and data imputation. The debiasing capacity of this model can be ensured if either the imputed data or propensity score are accurately estimated

CausE and KD-label. CausE and KD-label are excellent methods that transfer the unbiased information with a teacher model using knowledge distillation techniques. Both of them are the best performing models in Ref. [5].

AutoDebias. AutoDebias is a generic and effective solution for recommendation debiasing.

We also test three versions of our LightAD for ablation studies: (i) LightAD-uniform, which improves AutoDebias with a uniform sampler. (ii) LightAD-fixed, where we remove the selfpaced strategy and fix α as a constant (i.e., $\alpha = 0.5$). (iii) LightAD-selfpaced, the complete model proposed in Section 4.2.

Table 2 shows the performance of our LightAD and the baselines with respect to three evaluation metrics on two datasets. We make the following two observations:

(I) As a whole, our LightAD-selfpaced outperforms other baselines by a large margin. Contrast with AutoDebias, our model achieves similar performance with AutoDebias in both datasets and some metrics even exceed it. This can be attributed to the fact that AutoDebias gives the same attention to all samples during training, which will hinder the capture of truly useful information. It's worth noting that the learning of sampling strategy is only conducted on a very small amount of data instances i.e., the same order of magnitude as the biased instances, which is far less than $O(nm)$.

(II) In terms of all the sampling strategies, LightAD with

Table 2. Comparison results on explicit feedback data. The boldface font denotes the winner in that column.

Model	Yahoo!R3			Coat		
	NLL	AUC	NDCG@5	NLL	AUC	NDCG@5
MF (biased)	-0.587	0.727	0.547	-0.546	0.757	0.484
MF (uniform)	-0.503	0.568	0.435	-0.641	0.540	0.362
MF (combined)	-0.579	0.729	0.551	-0.538	0.750	0.492
IPS	-0.450	0.725	0.556	-0.531	0.755	0.484
DR	-0.444	0.723	0.552	-0.513	0.760	0.501
CausE	-0.585	0.729	0.553	-0.529	0.762	0.500
KD-label	-0.587	0.727	0.547	-0.546	0.757	0.484
AutoDebias	-0.419	0.748	0.648	-0.529	0.766	0.501
LightAD-uniform	-0.410	0.734	0.586	-0.505	0.710	0.473
LightAD-fixed	-0.406	0.737	0.632	-0.499	0.742	0.489
LightAD-selfpaced	-0.425	0.748	0.641	-0.510	0.765	0.517

simple uniform sampling obtains the worst performance and convergence. This is consistent with our intuition that uniform sampling suffers from high variance. The fact that LightAD-fixed outperforms LightAD-uniform proves that the informative sampler could partly address this problem. LightAD-selfpaced achieves the best performance which validates the effectiveness of the proposed selfpaced strategy.

6.3 Efficiency comparison (RQ2)

As we mention in Section 3.3, the calculation on imputed training instances consumes enormous computing resources and restricts the practicality of the AutoDebias. In this part, we test this notion through ablation experiments. Table 3 shows the time cost of the ablated methods with different components being deleted. From the table, we can conclude that the introduction of these components $\{w^{(1)}, w^{(2)}, m\}$ has different effects on the efficiency of our model. $w^{(1)}$ has much less impact on model efficiency than the other two. This is consistent with our previous analysis and demonstrates the necessity of our sampling.

To verify the efficiency promotion of our methods, we conduct experiments in two dimensions: the time cost in each epoch and the overall time cost. To ensure the reliability of the experiment as much as possible, we validated the experiment under the same conditions i.e., the same hyperparameter configuration and computing resource. The statistics of training time for these four models are shown in Table 4. From this table, we can find that employing samplers could indeed accelerate the training of AutoDebias. More interestingly, employing a uniform sampler achieves fast calculation in each epoch while requiring more time cost in total. We ascribe it to the fact that informative sampler could improve convergence and reduce the number of the required epochs. To prove this, we plotted the training process of LightAD-fixed and LightAD-uniform. As shown in the Fig. 1, it takes less than 20 epochs for LightAD-fixed to converge while taking more than 40 epochs to achieve the analogous level in LightAD-uniform. The experimental results are consistent with our previous theoretical proof.

6.4 Hyperparameter analysis (RQ3)

Note that in LightAD-selfpaced, we gradually increase the

Table 3. Ablation study of time cost per training epoch on Yahoo!R3.

Model	$w^{(1)}$	m	$w^{(2)}$	Epoch (s)
MF	×	×	×	0.02
AutoDebias- $w^{(1)}$	√	×	×	0.07
AutoDebias- $w^{(1)}m$	√	√	×	0.92
AutoDebias	√	√	√	1.3

Table 4. Training time on Yahoo!R3.

Model	Epoch (s)	Total time (s)
AutoDebias	1.30	1260.50
LightAD-uniform	0.14	74.10
LightAD-fixed	0.34	107.36
LightAD-selfpaced	0.20	29.05

contribution of the informative sampler while hyperparameter β controls the increasing ratio. In this subsection, we adjust the value of β and explore how it affects recommendation performance. The results are presented in Fig. 2. As we can see, as β becomes larger, with few exceptions, the performance will increase first and then drop when β surpasses a threshold. The reason is that too small (or large) β would make the model focus on informative sampler too early (or late). Setting proper β makes the model achieve the best performance.

6.5 Universality verification (RQ4)

To demonstrate the universality of our LightAD, we conduct experiments on Simulation. We compare our three sampling strategies with two SOTA methods in mitigating both position bias and selection bias:

Dual learning algorithm (DLA)^[44]. DLA treated the learning of the propensity function and the ranking list as a dual problem and designed specific EM algorithms to learn the two models, which is one SOTA method in mitigating position bias.

Heckman ensemble (HeckE)^[45]. Ascribed bias to the reality that users can only view a truncated list of recommendations. HeckE adopted a two-step method in which it first

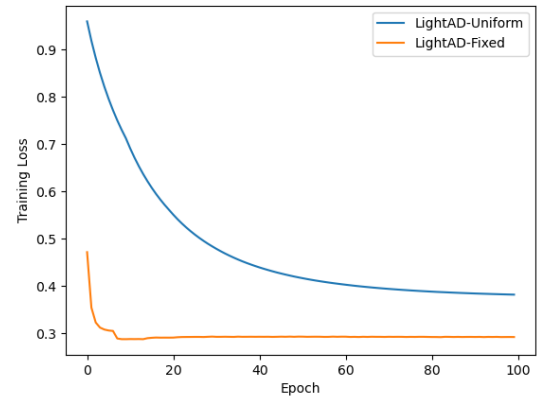


Fig. 1. Training process of LightAD-fixed and LightAD-uniform.

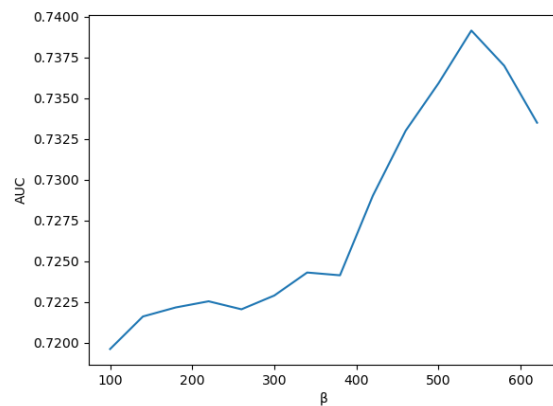


Fig. 2. The influence of hyperparameter β on self-paced sampling strategy.

designed a Probit model to estimate the probability of a item being examined and then took advantage of this probability to modify their model.

Different from the models in Section 6.2, we add position information in the modeling of $w_k^{(1)}$ and left other experimental settings unchanged, i.e., $w_k^{(1)} = \exp(\varphi_1^T[\mathbf{x}_{u_k} \circ \mathbf{x}_{i_k} \circ \mathbf{e}_{r_k} \circ \mathbf{e}_{p_k}])$, where \mathbf{e}_{p_k} denotes the feature vectors of position information. Table 5 shows the performance of our LightAD and the baselines on Simulation. We can find LightAD vastly outperform SOTA methods which verifies that LightAD can handle various biases and their combination.

Table 5. Comparison results on simulation dataset.

	NLL	AUC	NDCG@5
DLA	-0.693	0.572	0.595
HeckE	-0.692	0.587	0.657
LightAD-uniform	-0.731	0.603	0.665
LightAD-fixed	-0.745	0.604	0.680
LightAD-selfpaced	-0.685	0.611	0.700

7 Conclusions

In this work, we identify the inefficiency issue of AutoDebias, and propose to accelerate AutoDebias with a carefully designed sampling strategy. The sampler could adaptively and dynamically draw informative training instances, which brings provably better convergence and stability than the standard uniform sampler. Extensive experiments on three benchmark datasets validate that our LightAD accelerates AutoDebias by several magnitudes while maintaining almost equal recommendation performance and universality.

One promising research direction for future work is to explore pruning strategy for AutoDebias. AutoDebias involves a large number of debiasing parameters, and of course not all parameters are necessary. Leveraging AutoML or lottery hypothesis to locate important parameters while leave others would significantly reduce the time and space cost, mitigate over-fitting and potentially boost recommendation performance.

Acknowledgements

This work was supported by the Fundamental Research Funds for the Central Universities (WK2100000019).

Conflict of interest

The authors declare that they have no conflict of interest.

Biographies

Yang Qiu is currently an Algorithm Researcher in IDATA (Hefei). He received his master's degree from the University of Science and Technology of China in 2023. His research interests primarily revolve recommended system and large language model.

Jiawei Chen is currently a "Hundred Talent" research fellow in the College of Computer Science and Technology at Zhejiang University. He

was a postdoctor research fellow under the supervision of Prof. Xiangnan He at University of Science and Technology of China during July 2020 to July 2022. His research interests mainly focus on information retrieval, data mining, and machine learning, particularly in recommender systems, robustness, and graph neural networks.

References

- [1] He X, Liao L, Zhang H, et al. Neural collaborative filtering. In: WWW'17: Proceedings of the 26th International Conference on World Wide Web. Perth, Australia: ACM, **2017**: 173–182.
- [2] Yuan F, He X, Karatzoglou A, et al. Parameter-efficient transfer from sequential behaviors for user modeling and recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM, **2020**: 1469–1478.
- [3] Sun F, Liu J, Wu J, et al. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. New York: ACM, **2019**: 1441–1450.
- [4] Abdollahpouri H, Burke R, Mobasher B. Controlling popularity bias in learning-to-rank recommendation. In: Proceedings of the Eleventh ACM Conference on Recommender Systems. New York: ACM, **2017**: 42–46.
- [5] Liu D, Cheng P, Dong Z, et al. A general knowledge distillation framework for counterfactual recommendation via uniform data. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM, **2020**: 831–840.
- [6] Schnabel T, Swaminathan A, Singh A, et al. Recommendations as treatments: Debiasing learning and evaluation. In: Proceedings of the 33rd International Conference on Machine Learning. New York: ACM, **2016**: 1670–1679.
- [7] Wang X, Bendersky M, Metzler D, et al. Learning to rank with selection bias in personal search. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. New York: ACM, **2016**: 115–124.
- [8] Hernández-Lobato J M, Houlby N, Ghahramani Z. Probabilistic matrix factorization with non-random missing data. In: Proceedings of the 31st International Conference on International Conference on Machine Learning-Volume 32. New York: ACM, **2014**: II-1512–II-1520.
- [9] Steck H. Evaluation of recommendations: Rating-prediction and ranking. In: Proceedings of the 7th ACM Conference on Recommender Systems. New York: ACM, **2013**: 213–220.
- [10] Chen J, Dong H, Qiu Y, et al. AutoDebias: Learning to debias for recommendation. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM, **2021**: 21–30.
- [11] Chen J, Dong H, Wang X, et al. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems*, **2023**, 41: 1–39.
- [12] Marlin B, Zemel R S, Roweis S, et al. Collaborative filtering and the missing at random assumption. arXiv: 1206.5267, **2012**.
- [13] Steck H. Training and testing of recommender systems on data missing not at random. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, **2010**: 713–722.
- [14] Krishnan S, Patel J, Franklin M J, et al. A methodology for learning, analyzing, and mitigating social influence bias in recommender systems. In: Proceedings of the 8th ACM Conference on Recommender Systems. New York: ACM, **2014**: 137–144.
- [15] Liu Y, Cao X, Yu Y. Are you influenced by others when rating? Improve rating prediction by conformity modeling. In: Proceedings of the 10th ACM Conference on Recommender Systems. New York:

- ACM, **2016**: 269–272.
- [16] Tang J, Gao H, Liu H. mTrust: Discerning multi-faceted trust in a connected world. In: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining. New York: ACM, **2012**: 93–102.
- [17] Ma H, King I, Lyu M R. Learning to recommend with social trust ensemble. In: Proceedings of the 32nd international ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM, **2009**: 203–210.
- [18] Hu Y, Koren Y, Volinsky C. Collaborative filtering for implicit feedback datasets. In: 2008 Eighth IEEE International Conference on Data Mining. Pisa, Italy: IEEE, **2008**: 263–272.
- [19] Pan R, Scholz M. Mind the gaps: Weighting the unknown in large-scale one-class collaborative filtering. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). New York: ACM, **2009**: 667–676.
- [20] Chen J, Wang C, Zhou S, et al. Fast adaptively weighted matrix factorization for recommendation with implicit feedback. *Proceedings of the AAAI Conference on Artificial Intelligence*, **2020**, 34: 3470–3477.
- [21] Dupret G E, Piwowarski B. A user browsing model to predict search engine click data from past observations. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Singapore: ACM, **2008**: 331–338.
- [22] Zhang W V, Jones R. Comparing click logs and editorial labels for training query rewriting. In: WWW 2007 Workshop on Query Log Analysis: Social And Technological Challenges. Banff, Canada: ACM, **2007**: 43.
- [23] Craswell N, Zoeter O, Taylor M, et al. An experimental comparison of click position-bias models. In: Proceedings of the 2008 International Conference on Web Search and Data Mining. New York: ACM, **2008**: 87–94.
- [24] Guo F, Liu C, Kannan A, et al. Click chain model in web search. In: Proceedings of the 18th International Conference on World Wide Web. New York: ACM, **2009**: 11–20.
- [25] Zhu Z A, Chen W, Minka T, et al. A novel click model and its applications to online advertising. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining. New York: ACM, **2010**: 321–330.
- [26] Kamishima T, Akaho S, Asoh H, et al. Correcting popularity bias by enhancing recommendation neutrality. In: Proceedings of the 8th ACM Conference on Recommender Systems. Foster City, USA: ACM, **2014**.
- [27] Zheng Y, Gao C, Li X, et al. Disentangling user interest and conformity for recommendation with causal embedding. In: Proceedings of the Web Conference 2021. New York: ACM, **2021**: 2980–2991.
- [28] Krishnan A, Sharma A, Sankar A, et al. An adversarial approach to improve long-tail performance in neural collaborative filtering. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. New York: ACM, **2018**: 1491–1494.
- [29] Rendle S, Freudenthaler C, Gantner Z, et al. BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. Montreal, Canada: AUAI Press, **2009**: 452–461.
- [30] Wu Y, DuBois C, Zheng A X, et al. Collaborative denoising auto-encoders for top-N recommender systems. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. San Francisco, USA: ACM, **2016**: 153–162.
- [31] He X, Deng K, Wang X, et al. 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. Xi'an, China: ACM, **2020**: 639–648.
- [32] He X, Du X, Wang X, et al. Outer product-based neural collaborative filtering. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence. Stockholm, Sweden: IJCAI Organization, **2018**: 2227–2233.
- [33] Yu H-F, Bilenko M, Lin C-J. Selection of negative samples for one-class matrix factorization. In: Proceedings of the 2017 SIAM International Conference on Data Mining. Houston, USA: SIAM, **2017**: 363–371.
- [34] Park D H, Chang Y. Adversarial sampling and training for semi-supervised information retrieval. In: WWW'19: The World Wide Web Conference. New York: ACM, **2019**: 1443–1453.
- [35] Rendle S, Freudenthaler C. Improving pairwise learning for item recommendation from implicit feedback. In: WSDM'14: Proceedings of the Seventh ACM International Conference on Web Search and Data Mining. New York, USA: ACM, **2014**: 273–282.
- [36] Ding J, Quan Y, He X, et al. Reinforced negative sampling for recommendation with exposure data. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence. Macao, China: IJCAI Organization, **2019**: 2230–2236.
- [37] Ding J, Feng F, He X, et al. An improved sampler for Bayesian personalized ranking by leveraging view data. In: Companion Proceedings of The Web Conference 2018. Lyon, France: ACM, **2018**: 13–14.
- [38] Chen J, Wang C, Zhou S, et al. SamWalker: Social recommendation with informative sampling strategy. In: WWW'19: The World Wide Web Conference. New York: ACM, **2019**: 228–239.
- [39] Haussler D. Probably Approximately Correct Learning. Palo Alto, USA: AAAI Press, **1990**.
- [40] Sun W, Khenissi S, Nasraoui O, et al. Debiasing the human-recommender system feedback loop in collaborative filtering. In: WWW'19: The World Wide Web Conference. San Francisco, USA: ACM, **2019**: 645–651.
- [41] Gleich D F, Lim L H. Rank aggregation via nuclear norm minimization. In: KDD'11: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Diego, USA: ACM, **2011**: 60–68.
- [42] Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems. *Computer*, **2009**, 42: 30–37.
- [43] Wang X, Zhang R, Sun Y, et al. Doubly robust joint learning for recommendation on data missing not at random. In: ICML'19: Proceedings of the 36th International Conference on Machine Learning. Long Beach, USA: PMLR, **2019**: 6638–6647.
- [44] Ai Q, Bi K, Luo C, et al. Unbiased learning to rank with unbiased propensity estimation. In: SIGIR'18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. New York: ACM, **2018**: 385–394.
- [45] Ovaisi Z, Ahsan R, Zhang Y, et al. Correcting for selection bias in learning-to-rank systems. In: Proceedings of The Web Conference 2020. New York: ACM, **2020**: 1863–1873.