Making Users Indistinguishable: Attribute-wise Unlearning in Recommender Systems

Yuyuan Li
College of Computer Science,
Zhejiang University
Hangzhou, China
11821022@zju.edu.cn

Chaochao Chen*
College of Computer Science,
Zhejiang University
Hangzhou, China
zjuccc@zju.edu.cn

Xiaolin Zheng
College of Computer Science,
Zhejiang University
Hangzhou, China
xlzheng@zju.edu.cn

Yizhao Zhang
College of Computer Science,
Zhejiang University
Hangzhou, China
22221337@zju.edu.cn

Zhongxuan Han
College of Computer Science,
Zhejiang University
Hangzhou, China
zxhan@zju.edu.cn

Dan Meng
OPPO Research Institute
Shenzhen, China
mengdan90@163.com

Jun Wang
OPPO Research Institute
Shenzhen, China
junwang.lu@gmail.com

ABSTRACT

With the growing privacy concerns in recommender systems, recommendation unlearning, i.e., forgetting the impact of specific learned targets, is getting increasing attention. Existing studies predominantly use training data, i.e., model inputs, as the unlearning target. However, we find that attackers can extract private information, i.e., gender, race, and age, from a trained model even if it has not been explicitly encountered during training. We name this unseen information as *attribute* and treat it as the unlearning target. To protect the sensitive attribute of users, Attribute Unlearning (AU) aims to degrade attacking performance and make target attributes indistinguishable. In this paper, we focus on a strict but practical setting of AU, namely Post-Training Attribute Unlearning (PoT-AU), where unlearning can only be performed after the training of the recommendation model is completed. To address the PoT-AU problem in recommender systems, we design a two-component loss function that consists of i) distinguishability loss: making attribute labels indistinguishable from attackers, and ii) regularization loss: preventing drastic changes in the model that result in a negative impact on recommendation performance. Specifically, we investigate two types of distinguishability measurements, i.e., user-to-user and distribution-to-distribution. We use the stochastic gradient descent algorithm to optimize our proposed loss. Extensive experiments on three real-world datasets demonstrate the effectiveness of our proposed methods.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Collaborative filtering**; • **Security and privacy** → **Social network security and privacy**.

KEYWORDS

Recommender Systems, Collaborative Filtering, Attribute Unlearning

ACM Reference Format:
Yuyuan Li, Chaochao Chen, Xiaolin Zheng, Yizhao Zhang, Zhongxuan Han, Dan Meng, and Jun Wang. 2023. Making Users Indistinguishable: Attribute-wise Unlearning in Recommender Systems. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29–November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3581783.3612418

1 INTRODUCTION

Recommendation unlearning has gained increasing interest in recent years. On the one hand, recommender systems have been widely applied in practice with great success, having a substantial influence on people's lifestyles [12, 25, 41]. The success lies in their ability to extract highly personalized information from user data. On the other hand, people have grown more aware of privacy concerns in personalized recommendations, and demand their sensitive information be protected. As one of the protective measures, *Right to be Forgotten* [8, 9, 16] requires recommendation platforms to enable users to withdraw their individual data and its impact, which impulses the study of machine/recommendation unlearning.

Existing studies of machine unlearning mainly use training data, i.e., model inputs, as the unlearning target [38]. We name this type of unlearning task as Input Unlearning (IU). In the recommendation

*Corresponding author.

MM '23, October 29–November 3, 2023, Ottawa, ON, Canada.

ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3612418

Table 1: Difference between input unlearning and attribute unlearning in recommender systems.

	Input Unlearning	**Attribute Unlearning**
Unlearning target	Input data (used in training)	Latent attribute (**not** used in training)
Applicability of retraining from scratch	Ground truth	Not applicable

scenarios, the input data can be a user-item interaction matrix. With different unlearning targets, IU can be user-wise, item-wise, and instance-wise [11]. IU benefits multiple parties, e.g., data providers and model owners, because the target data can be i) the specified data that contains users' sensitive information, and ii) the dirty data that is polluted by accidental mistakes or intentional attack [34].

Extensive studies on IU cannot obscure the importance of Attribute Unlearning (AU), where attributes represent the inherent properties, e.g., gender, race, and age of users that have **not** been used for training (Table 1: difference in unlearning target). Due to the information extraction capabilities of recommender systems, AU is especially valuable in the context of recommendation. Although recommendation models did not see the latent attribute, the research found that basic machine learning models can successfully infer users' attributes from the user embedding learned by collaborative filtering models [18], which is also known as Attribute Inference Attack (AIA) [3, 31]. Therefore, from the perspective of privacy preservation, AU is as important as IU in recommender systems. However, existing IU methods cannot be applied in AU. As illustrated in Table 1, retraining from scratch (ground truth for IU) is unable to unlearn the latent attribute, i.e., not applicable for AU, since it is not utilized during training at all.

Existing but limited research on AU has focused on In-Training AU (InT-AU) [18, 24], where unlearning is performed during model training (as shown in the left of Figure 1). In this paper, we focus on a more strict AU setting, namely Post-Training Attribute Unlearning (PoT-AU), where we can only manipulate the model, i.e., updating parameters, after the training is fully completed and have no knowledge about training data or other training information (as shown in the right of Figure 1). Compared with InT-AU, this setting is more practical, because of i) data accessibility: we may not get access to the training data or other information after training due to regulations, and ii) deployment overhead: non-interference with the original training process is more flexible and reduces deployment overhead. As shown in Figure 1, there are two goals for PoT-AU in recommender systems, where user embedding is the target of attacking and unlearning. The primary goal (**Goal #1**) is to make the target attribute indistinguishable to AIA, or more directly, to degrade the attacking performance. The other goal (**Goal #2**) is to maintain the recommendation performance. This goal is equally important as the primary one, since both users and recommendation platforms want to avoid having a negative impact on the original recommendation tasks.

To achieve the above two goals in the PoT-AU problem, we consider it as an optimization problem on user embedding which is the crucial parameter in the recommendation model that contains user information. We then design a two-component loss function that consists of i) *distinguishability loss*: measuring the distinguishable degree of users with different attribute labels, and ii) *regularization loss*: measuring the divergence of current parameters and the original ones. Each component is individually devised for one goal in the PoT-AU problem. We introduce the general design principle of each component, which is applicable to all recommendation models that have user embeddings. Specifically, we mainly focus on the design of distinguishability loss and investigate two types of distinguishability measurements from different perspectives, i.e., User-to-User (U2U) and Distribution-to-Distribution (D2D). U2U loss measures the distinguishability by the weighted embedding divergence of each user-pairs if the two users in the pair have different attribute labels. D2D loss regards all users within the same attribute label as a distribution and measures the distinguishability by the distance of different distributions. We implement our proposed loss functions with effective and computationally efficient components, and adopt stochastic gradient descent to optimize them.

We summarize the main contributions of this paper as follows:

- To the best of our knowledge, we are the first to study the PoT-AU problem in recommender systems, which is a more strict and practical problem than InT-AU. We identify two essential goals of PoT-AU in recommender systems, i.e., making attributes indistinguishable, and maintaining recommendation performance.
- To address the PoT-AU problem, we propose a two-component loss function, i.e., a linear combination of distinguishability loss and regularization loss. Each component is devised to achieve one of the above goals separately. From two different perspectives, we design two types of distinguishability loss, i.e., U2U and D2D.
- We implement two examples, i.e., U2U-R and D2D-R, of our proposed loss function and conduct extensive experiments on three real-world datasets to comprehensively evaluate the effectiveness of our proposed methods in terms of both unlearning (**Goal #1**) and recommendation (**Goal #2**).
- To better understand the mechanism of our proposed method, we analyze the user embedding changes before and after unlearning, finding the connection between these changes and the performance difference between U2U and D2D. Finally, we also discuss the potential variations of different distinguishability losses.

2 PRELIMINARIES

2.1 Recommendation Model

Today's recommender systems can provide personalized recommendations based on collaborative information across users and items. Collaborative filtering is an acknowledged algorithm for analyzing this information [43].In this paper, we choose the widely applied collaborative filtering which is based on matrix factorization (MF-based CF). The basic idea of MF-based CF is to decompose the user-item interaction matrix into two low-rank embedding matrices, i.e., user embedding and item embedding. In the PoT-AU problem, we use user embedding as the target of attacking and unlearning.

2.2 Attacking Setting

The process of attacking in PoT-AU problem is also known as AIA, which consists of three stages, i.e., exposure, training, and inference.

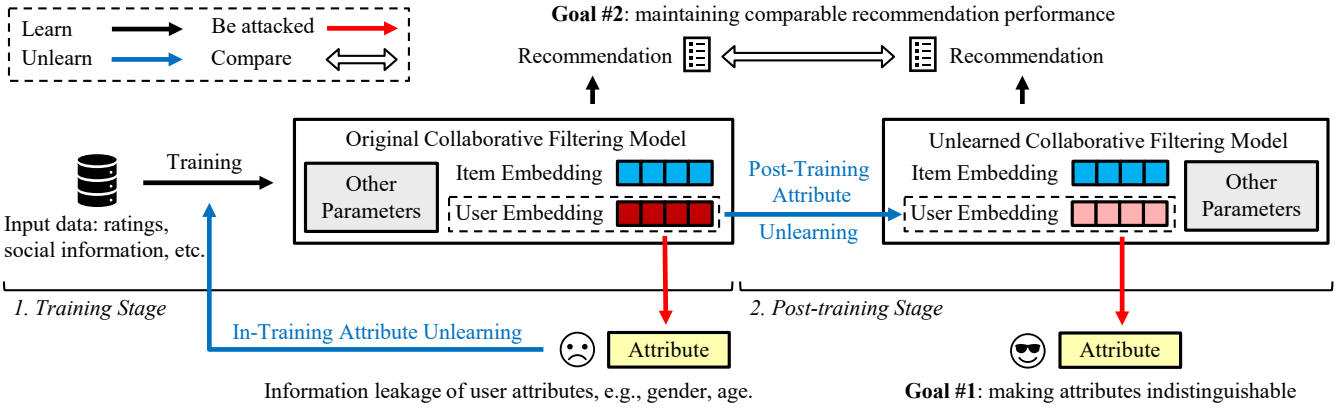


Figure 1: Overview of Post-Training Attribute Unlearning (PoT-AU) in recommender systems.

In the exposure stage, we assume that attackers follow the grey-box attack. In other words, not all model parameters but only users' embedding and their corresponding attribute information are exposed to attackers. In the training stage, we assume that attackers train the attacking model on a shadow dataset, which can be generated by sampling from the original users or users from the same distribution [40]. Although shadow-dataset training will inevitably reduce attacking performance, this assumption is reasonable, since the full-dataset setting is too strong and impractical. Regarding the attack as a classification task, attacking models use user embedding as input data and attribute information as labels. For conciseness, we focus on binary classification in this paper. We will discuss the generalization to multi-class classification in Section 6. In the inference stage, attackers use their trained attacking models to make predictions.

3 POST-TRAINING ATTRIBUTE UNLEARNING

In this section, we first further explain our motivation as well as the process of the PoT-AU problem in recommender systems. Then we consider the PoT-AU problem as an optimization problem on user embedding and propose a novel two-component loss function to address it.

3.1 Motivation

As shown in Figure 1, we divide the whole process of PoT-AU into two stages, i.e., training stage and post-training stage. In the training stage, the recommender system trains an original collaborative filtering model on input data. To align with the post-training setting, we leave this stage untamed and assume that one has no information in this stage except the user embedding and their attributes. In the post-training stage, we generate new user embedding by unlearning the original one. The new embedding, i.e., user embedding after unlearning, is supposed to achieve two goals simultaneously. **Goal #1** (unlearning) is to make target attributes distinguishable so as to protect attribute information from attackers. **Goal #2** (recommendation) is to maintain the original recommendation performance so as not to influence users' initial requirements.

Compared with in-training setting (InT-AU), the post-training setting (PoT-AU) is more challenging. Firstly, PoT-AU allows no interference with the training process, which means InT-AU methods, i.e., adding network block [24], and adversarial training [18], are not applicable. Secondly, even though PoT-AU cuts down the connection with the training process, directly manipulating user embedding by adding artificially-designed noise, e.g., differential privacy [1], is inappropriate, because i) it will inevitably degrade recommendation performance, and ii) its unlearning ability is not promising, as the functional mechanism of attacking models, including complex machine learning models, is not well-understood. Thirdly, PoT-AU prohibits access to the input data and other training information, as the said data could be either unavailable or under protection. As a result, the said data cannot be used for fine-tuning user embedding, e.g., adding noise to the embedding and then fine-tuning to boost recommendation performance.

3.2 Two-Component Loss Function

One of the feasible solutions is to envision the final desired user embedding and temporally ignore the intermediate manipulation and transformation. Thus, we regard PoT-AU as an optimization problem on user embedding. In other words, we aim to design a proper loss function and let optimization techniques do the rest.

There are two goals in PoT-AU problem, i.e., **Goal #1**: unlearning and **Goal #2**: recommendation. Thus, we propose a two-component loss function in which one component is individually devised for one goal. Formally, we combine the two components linearly with a trade-off parameter α to balance both goals:

$$L(\theta) = \ell_u + \alpha \ell_r, \tag{1}$$

where $\theta \in \mathbb{R}^{N \times K}$ denotes user embedding (N is the number of users, and K is the number of embedding dimensions), ℓ_u and ℓ_r represent the loss for **Goal #1** and **Goal #2** respectively.

3.3 Distinguishability Loss

The core difficulty of designing a proper two-component loss function lies in defining distinguishability loss ℓ_u. It is related to the primary goal of PoT-AU, i.e., **Goal #1**. We define the distinguishability from two perspectives, i.e., User-to-User (U2U) and Distribution-to-Distribution (D2D). Following the definitions, we design two

types of distinguishability losses respectively. For conciseness, we assume the target attribute has binary labels: S_1 and S_2.

3.3.1 ***User-to-user Loss.*** The attacker infers users' attributes according to their embedding. From a U2U perspective, making attributes indistinguishable means letting users with different attribute labels have similar embedding. This idea is statistically expressed as a positive correlation between the divergence of attribute information and the similarity of user embedding. In other words, there is a negative correlation between the divergence of attribute information and the divergence of user embedding, which can be formally expressed as:

$$Div(\theta_i, \theta_j) \leq \frac{\delta}{Div(v_i, v_j)} \text{ for } i \in S_1, j \in S_2, \tag{2}$$

where Div can be certain divergence functions, v_i denotes the attribute information of user i, and δ is a constant. The attribute information is usually regarded as the attribute labels, but there are more advanced options in the PoT-AU setting, which we will discuss in Section 3.5.

As user embedding θ is the target to optimize, Equation (2) can be interpreted as the divergence of embedding $Div(\theta_i, \theta_j)$ is bounded by a scale of $\delta/Div(v_i, v_j)$. That is, the greater the divergence between the attribute information of users i and j, the tighter the upper bound, and hence the smaller the divergence in embedding between users is likely to be. Assuming that the divergence function is non-negative, we can rewrite Equation (2) as:

$$Div(\theta_i, \theta_j) \cdot Div(v_i, v_j) \leq \delta. \tag{3}$$

Equation (3) reflects the concept of U2U-perspective distinguishability, where δ is the upper bound of the distinguishability degree. We name this type of distinguishability measurement as U2U loss and define it as follows.

Definition 1 (User-to-user distinguishability). *Given the divergence of user embedding $Div(\theta_i, \theta_j)$ and the divergence of user attribute $Div(v_i, v_j)$, we define user-to-user distinguishability as:*

$$\ell_{u,U} = \sum_{i \in S_1} \sum_{j \in S_2} Div(\theta_i, \theta_j) \cdot Div(v_i, v_j). \tag{4}$$

Note that the divergence of embedding can be different from that of attribute, which leads to more possibility of U2U loss. Interpreting $Div(v_i, v_j)$ as a weight term, Equation (4) shows that U2U loss is a weighted divergence loss of user embedding. The larger the weight, the more stringent the measure of embedding divergence is likely to be, which reflects our design principle, i.e., letting users with different attribute labels have similar embedding.

3.3.2 ***Distribution-to-distribution Loss.*** We consider the user embedding with the same attribute label as a distribution, e.g., P_{θ_1} denotes the embedding distribution of users with label S_1. Inspired by Theorem 1, the distinguishability of user performance can be bounded by the $\mathcal{H}$-divergence of their distributions $div_{\mathcal{H}}(P_{\theta_1}, P_{\theta_2})$.

Theorem 1. *[Bound on Domain Risk [4]] Let $\mathcal{H}$ be a hypothesis class of VC dimension d. With probability $1-\epsilon$ over the choice of samples $\Theta_1 \sim P(\theta_1)^n$ and $\Theta_2 \sim P(\theta_2)^n$, for every $\eta \in \mathcal{H}$:*

$$R_{\Theta_1}(\eta) - R_{\Theta_2}(\eta) \leq \hat{div}_{\mathcal{H}}(\Theta_1, \Theta_2) + \beta + \gamma, \tag{5}$$

with $\beta = \sqrt{\frac{4}{n}(d \log \frac{2en}{d} + \log \frac{4}{\epsilon})} + 4\sqrt{\frac{1}{n}(d \log \frac{2n}{d} + \log \frac{4}{\epsilon})}$, and $\gamma \geq \inf_{\eta^* \in \mathcal{H}}[R_{P_{\theta_1}}(\eta^*) + R_{P_{\theta_2}}(\eta^*)]$.

In Theorem 1, $\hat{div}_{\mathcal{H}}$ is the empirical $\mathcal{H}$-divergence, and R is the domain risk which represents the recommendation loss in our scenarios, reflecting the performance of users. For practical consideration, it is worth noting that the embeddings of all users are trained together without any attribution information. As a result, the shapes of the embedding distribution tend to be similar across different attribute labels. The difference in distributions mainly comes from their distance. Therefore, we use $dist(P_{\theta_1}, P_{\theta_2})$ to approximate $div_{\mathcal{H}}(P_{\theta_1}, P_{\theta_2})$ and omit the constants for simplicity. We name this type of distinguishability measurement as D2D loss and define it as follows.

Definition 2 (Distribution-to-distribution distinguishability). *Given two distributions of embedding from users with different attribute labels P_{θ_1} and P_{θ_2}, we define distribution-to-distribution distinguishability as the distance between two distributions:*

$$\ell_{u,D} = Dist(P_{\theta_1}, P_{\theta_2}). \tag{6}$$

Thus, distributional distances, e.g., KL divergence [21] and Maximum Mean Discrepancy (MMD) [44], can be used to measure the degree of D2D distinguishability.

3.4 Regularization Loss

To achieve **Goal #2**, we add regularization loss ℓ_r in Equation (1). We name ℓ_r as regularization loss instead of recommendation loss. The reason is that we cannot use common recommendation loss, e.g., binary cross-entropy [48] and Bayesian personalized ranking [39]) in the post-training setting, as the training data is not available. As a result, we can only use the regularization term to bound the user embedding with the original one, preventing a drastic change in user embedding. The idea behind this is that we deem that closer model parameters will lead to closer model performance. Formally, the regularization loss is defined as $\ell_r = R(\theta, \theta^*)$, where θ^* denotes the original user embedding before unlearning.

3.5 Implementation

3.5.1 ***Regularization Loss.*** For the regularization loss ℓ_r, we choose the commonly used regularizer, Frobenius norm [5]. Formally,

$$R(\theta, \theta^*) = \|\theta - \theta^*\|_F^2 = \sum_{i=1}^{N} \sum_{j=1}^{K} (\theta_{i,j} - \theta^*_{i,j})^2. \tag{7}$$

3.5.2 ***Distinguishability Loss.***

U2U. For the U2U distinguishability loss, we also use Frobenius norm to measure the divergence of user embedding. Formally, $Div(\theta_i, \theta_j) = \|\theta_i - \theta_j\|_F^2$. As for attribute divergence, we use the inverse adjacent matrix. To be specific,

$$Div(v_i, v_j) = 1 - A(S_i, S_j), \quad A(S_i, S_j) = \begin{cases} 1 & \text{if } S_i = S_j. \\ 0 & \text{if } S_j \neq S_j. \end{cases} \tag{8}$$

Due to the property that $Div(v_i, v_j) = 0$ if $S_i = S_j$, we can rewrite U2U loss as:

$$\ell_{u,U} = \sum_{i=1}^{N}\sum_{j=1}^{N} \|\theta_i - \theta_j\|_F^2 \cdot Div(v_i, v_j) = 2\mathrm{Tr}(\theta^\top \mathcal{L}_D \theta), \quad (9)$$

where $\mathcal{L}_D$ is the Laplacian matrix of divergence $Div(v_i, v_j)$. This tensorized expression is more efficient for GPU devices to compute.

Although the choice of attribute divergence is not the main focus of this paper, we would like to have a brief discussion about it. Using the inverse adjacent matrix as attribute divergence is simple and effective, but it loses a certain amount of attribute information and there are more choices with potential. A straightforward choice is to transform attribute labels into one-hot vectors and apply classification losses, e.g., cross-entropy [50] and cosine similarity [15]. Rather than using one-hot vectors, a more advanced approach would be to use the predicted softmax vector from attacking models instead. This approach can not only preserve more information about attribute labels, but also adopt adversarial training while updating user embedding, which means $Div(v_i, v_j)$ is adaptively updated as $Div(\theta_i, \theta_j)$ is updated.

D2D. For the D2D distinguishability loss, we apply MMD with radial kernels [45] to measure the distance of two distributions. MMD satisfies several properties that are required as a distance measurement, including non-negativity and exchange invariance, i.e., $Dist(P_{\theta_1}, P_{\theta_2}) = Dist(P_{\theta_2}, P_{\theta_1})$.

3.5.3 ***Summary.*** Incorporating different distinguishability losses, we propose two loss functions for the PoT-AU problem. The U2U-R loss computes as:

$$L_U(\theta) = \ell_{u,U} + \alpha R(\theta, \theta^*) = 2\mathrm{Tr}(\theta^\top \mathcal{L}_D \theta) + \alpha\|\theta - \theta^*\|_F^2. \quad (10)$$

The D2D-R loss computes as:

$$L_D(\theta) = \ell_{u,D} + \alpha R(\theta, \theta^*) = \mathrm{MMD}(P_{\theta_1}, P_{\theta_2}) + \alpha\|\theta - \theta^*\|_F^2. \quad (11)$$

We apply the stochastic gradient descent algorithm [6] to optimize our proposed losses.

4 EXPERIMENTS

To comprehensively evaluate our proposed methods, we conduct experiments on three benchmark datasets and observe the performance in terms of unlearning and recommendation. We also investigate the efficiency and robustness of our proposed loss functions. We further analyze the change in user embedding before and after unlearning to better understand the mechanism of our proposed methods.

4.1 Experimental Settings

4.1.1 ***Datasets.*** Experiments are conducted on three publicly accessible datasets that contain both input data, i.e., user-item interactions, and user attributes, i.e., gender. Following [18], the provided gender information of the users are limited to females and males.

- **MovieLens 100K (ML-100K)**[1]: MovieLens is one of the most widely used datasets in the recommendation [26, 27]. They collected users' ratings towards movies as well as other attributes, e.g., gender, age, and occupation. ML-100K is the version containing 100 thousand ratings.
- **MovieLens 1M (ML-1M)**: This version has 1 million ratings.
- **LFM-2B**[2]: This dataset collected more than 2 billion listening events, which is used for music retrieval and recommendation tasks [37].LFM-2B also contains user attributes including gender, age, and country.

We filter out the users and items that have less than 5 interactions. Specifically, we use 80% of ratings for training, 10% as the validation set for tuning hyper-parameters, and the rest for testing. Table 2 summarizes the statistics of three datasets.

Table 2: Summary of datasets.

Dataset	Attribute	User #	Item #	Rating #	Sparsity
ML-100K	Total	943	1,682	100,000	93.695%
	Female	273		73,824	83.923%
	Male	670		26,176	97.677%
ML-1M	Total	6040	3,950	1,000,209	95.814%
	Female	1,079		246,896	94.207%
	Male	4,331		753,313	95.597%
LFM-2B	Total	19,972	99,639	2,829,503	99.858%
	Female	4,415		444,076	99.899%
	Male	15,557		2,385,427	99.846%

4.1.2 ***Recommendation Models.*** We test our proposed methods on two different recommendation models. As mentioned in Section 2.1, we focus on collaborative filtering models and use the *user embedding* as the attacking and unlearning target.

- **NMF** [30]: Neural Matrix Factorization (NMF) is one of the representative well-known models based on matrix factorization.
- **LightGCN** [29]: Light Graph Convolution Network (LightGCN) is the state-of-the-art collaborative filtering model which improves recommendation performance by simplifying the graph convolution network.

4.1.3 ***Attackers.*** We randomly expose 10% user embedding and their corresponding labels as the *shadow dataset* to train attackers, leaving the rest for testing. We choose two easy-implement and powerful machine learning models as attackers.

- **MLP** [19]: Multilayer Perceptron is a simplified three-layer neural network, which is a widely used classifier.
- **XGB** [14]: XGBoost is an acknowledged classifier in the industry. It is so powerful that it has been used in numerous machine learning since it was proposed [13].

4.1.4 ***Unlearning Methods.*** There are lots of studies on machine unlearning, but they are not applicable to the PoT-AU problem. To the best of our knowledge, we are the first to study the PoT-AU problem. Although InT-AU setting is different from ours, i.e., PoT-AU, comparing with InT-AU methods would be helpful to provide a comprehensive understanding of the AU problem. Thus, we compared our proposed U2U-R and D2D-R with the original user embedding and other InT-AU methods.

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

[2] http://www.cp.jku.at/datasets/LFM-2b

Table 3: Results of unlearning performance (performance of attackers). The top results are highlighted in bold. The lower the attacking performance, the better the unlearning performance.

NMF		MLP				XGB			
		Acc	Precision	Recall	AUC	Acc	Precision	Recall	AUC
ML-100K	Original	0.6455	0.6735	0.5893	0.6465	0.6364	0.6333	0.6786	0.6356
	U2U-R	**0.4818**	0.9997	0.0013	**0.5002**	0.9909	0.9828	0.9999	0.9906
	D2D-R	0.5182	0.6333	0.3115	0.5434	0.5091	0.5593	0.5410	**0.5052**
	Retrain	0.5091	1.0000	**0.0003**	0.5006	**0.3818**	**0.3965**	**0.4107**	0.6187
	Adv-InT	0.4883	**0.5758**	0.2054	0.5135	0.5933	0.6409	0.5662	0.5957
ML-1M	Original	0.7485	0.7403	0.7125	0.7464	0.7310	0.7361	0.6625	0.7269
	U2U-R	**0.4737**	1.0000	**0.0002**	**0.5001**	0.9831	0.9763	0.9981	0.9845
	D2D-R	0.4795	**0.4167**	0.6338	0.5019	0.5029	**0.4186**	0.5070	**0.5035**
	Retrain	0.4883	0.4868	1.0000	0.5028	**0.4437**	0.4598	**0.4819**	0.5261
	Adv-InT	0.4912	0.4557	0.4161	0.5129	0.6079	0.6551	0.5577	0.6110
LFM-2B	Original	0.7181	0.6906	0.7501	0.7192	0.6767	0.6419	0.7422	0.6790
	U2U-R	**0.5188**	0.5641	**0.1654**	0.5188	0.9991	0.9873	0.9924	0.9993
	D2D-R	0.5263	0.5128	0.6154	0.5283	0.5075	0.4960	**0.4769**	0.5068
	Retrain	0.5226	**0.5021**	0.9531	0.5382	**0.5038**	**0.4853**	0.5156	**0.5042**
	Adv-InT	0.5205	0.5243	0.6207	**0.5187**	0.5825	0.6279	0.5557	0.5812
LightGCN		MLP				XGB			
		Acc	Precision	Recall	AUC	Acc	Precision	Recall	AUC
ML-100K	Original	0.6220	0.6363	0.6512	0.6205	0.6585	0.6596	0.7209	0.6553
	U2U-R	**0.4391**	0.9993	**0.0012**	**0.5011**	0.9756	0.95838	0.9998	0.9722
	D2D-R	0.5244	0.4737	0.4865	0.5210	0.5047	0.5213	0.5132	**0.5045**
	Retrain	0.4545	**0.4483**	0.4815	0.5449	**0.4795**	**0.4952**	**0.4962**	0.5107
	Adv-InT	0.5375	0.4907	0.5221	0.5326	0.5785	0.6032	0.6007	0.5864
ML-1M	Original	0.7076	0.6871	0.6957	0.7069	0.6754	0.6524	0.6646	0.6748
	U2U-R	0.5175	0.5134	0.9943	**0.5090**	0.9874	0.9761	0.9973	0.9889
	D2D-R	0.5234	0.5281	0.5434	0.5232	0.5146	0.5205	0.5145	0.5146
	Retrain	**0.4737**	**0.4641**	**0.4201**	0.5269	**0.5029**	**0.4969**	**0.4793**	**0.5027**
	Adv-InT	0.5117	0.4908	0.8209	0.5272	0.5917	0.6544	0.5725	0.6037
LFM-2B	Original	0.7218	0.6712	0.7903	0.7261	0.6541	0.6143	0.6935	0.6566
	U2U-R	**0.4812**	0.4773	0.9981	0.5071	0.9997	0.9989	0.9874	0.9998
	D2D-R	0.5062	0.5151	**0.4928**	**0.5064**	**0.5113**	0.5286	**0.5312**	**0.5103**
	Retrain	0.5489	**0.4247**	0.5000	0.5458	0.5263	**0.4459**	0.5323	0.5226
	Adv-InT	0.5439	0.5139	0.5714	0.5454	0.5879	0.6124	0.5548	0.5734

- **U2U-R** (PoT-AU): This is the two-component loss function with user-to-user loss as distinguishability loss, i.e., Equation (10).
- **D2D-R** (PoT-AU): This is the two-component loss function with distribution-to-distribution loss as distinguishability loss, i.e., Equation (11).
- **Original**: This is the original user embedding before unlearning.
- **Retrain** [49] (InT-AU): This method adds a regularizer to the original loss to achieve fairness. Inspired by it, we add our proposed D2D loss to the original recommendation loss and retrain the model from scratch. Note that we only use D2D loss because incorporating U2U loss in this setting would be computationally prohibitive.
- **Adv-InT** [18] (InT-AU): This method uses adversarial training to achieve InT-AU for variational auto-encoder. We also apply the idea of adversarial training to our tested recommendation models, i.e., NMF and LightGCN, and name it Adv-InT.

We run all models 10 times and report the average results. Due to the space limit, we report the hyper-parameter settings and hardware information in Appendix A.

4.2 Results and Discussions

4.2.1 ***Unlearning Performance.*** The performance of unlearning is evaluated by the performance of attackers. We train attackers on the shadow training set, and report their performance on the testing set. To comprehensively evaluate attacking performance, we report four metrics, including Accuracy (Acc), precision, recall, and Area Under the ROC Curve (AUC) [17], in Table 3. To visually analyze the results, we also use t-SNE [46] to reduce dimension (Figure 3 in

Table 4: Results of recommendation performance. The top results are highlighted in bold.

NMF		NDCG@5	HR@5	NDCG@10	HR@10
ML-100K	Original	0.4308	0.6098	**0.4179**	**0.4491**
	U2U-R	0.4174	0.6015	0.4073	0.4425
	D2D-R	**0.4320**	**0.6102**	0.4147	0.4489
	Retrain	0.4159	0.6004	0.4109	0.4439
	Adv-InT	0.4213	0.5997	0.4097	0.4453
ML-1M	Original	**0.5146**	**0.7517**	**0.5054**	**0.5761**
	U2U-R	0.4316	0.6957	0.4430	0.5443
	D2D-R	0.5110	0.7497	0.5033	0.5750
	Retrain	0.5095	0.7421	0.4962	0.5673
	Adv-InT	0.4913	0.7395	0.4875	0.5514
LFM-2B	Original	0.2082	**0.8510**	**0.3773**	**0.8130**
	U2U-R	0.1342	0.6572	0.2268	0.6236
	D2D-R	**0.2111**	0.8479	0.3746	0.8103
	Retrain	0.2033	0.8414	0.3695	0.8071
	Adv-InT	0.2012	0.8335	0.2706	0.8075
LightGCN		NDCG@5	HR@5	NDCG@10	HR@10
ML-100K	Original	**0.4394**	**0.6161**	**0.4195**	**0.4496**
	U2U-R	0.4274	0.6106	0.4079	0.4418
	D2D-R	0.4384	0.6131	0.4180	0.4492
	Retrain	0.4211	0.6051	0.4092	0.4450
	Adv-InT	0.4285	0.6063	0.4134	0.4443
ML-1M	Original	**0.4554**	**0.7170**	**0.4674**	**0.5590**
	U2U-R	0.4296	0.6983	0.4445	0.5452
	D2D-R	0.4534	0.7168	0.4657	0.5586
	Retrain	0.4471	0.7176	0.4666	0.5567
	Adv-InT	0.4485	0.7106	0.4625	0.5538
LFM-2B	Original	**0.2234**	**0.8898**	**0.4025**	**0.8471**
	U2U-R	0.2148	0.8807	0.3898	0.8332
	D2D-R	0.2228	0.8894	0.4018	0.8461
	Retrain	0.2230	0.8885	0.3996	0.8449
	Adv-InT	0.2213	0.8827	0.3943	0.8425

Table 5: Running time of unlearning methods.

Time (s)		U2U-R	D2D-R	Retrain	Adv-InT
ML-100K	NMF	5.78	2.69	117.39	223.21
	LightGCN	11.86	5.99	284.65	567.24
ML-1M	NMF	66.65	28.80	629.09	928.55
	LightGCN	159.22	60.99	1402.22	2103.57
LFM-2B	NMF	110.05	45.00	1116.19	1457.31
	LightGCN	364.73	147.54	2993.87	3503.16

Appendix B.2). We have the following observations from the above results. Firstly, attackers achieve an average accuracy of 0.68 on the original embedding, indicating that information on the user's attribute in user embedding is released to the attackers. Secondly, all methods can degrade the attacking performance of MLP. U2U-R, D2D-R, Retrain, and Adv-InT decrease the AUC by 27.11%, 25.00%, 24.16%, and 24.37%, respectively, on average. Thirdly, U2U-R cannot fool XGB and increase the attacking performance significantly instead (average AUC is up to 0.99). We will further analyze U2U-R's different performance w.r.t MLP and XGB in Section 4.2.5. D2D-R can decrease the AUC of XGB by 24.41% on average. In comparison, Retrain and Adv-InT can only decrease the AUC by 20.93% and 11.84%, respectively.

Summary: Compared with U2U-R, D2D-R is more effective in unlearning. D2D-R protects the user's attributes by making them indistinguishable to the attacker, outperforming the compared methods.

4.2.2 ***Recommendation Performance.*** We use two common metrics, i.g., Normalized Discounted Cumulative Gain (NDCG) and Hit Ratio (HR), to evaluate recommendation performance [28, 48]. We truncate the ranked list at 5 and 10 for both metrics. As shown in Table 4, unlearning methods also affect recommendation performance. Compared with the original performance, U2U-R has an average degradation of 5.30% on NDCG and 3.77% on HR. We analyze the reasons for degradation in Section 4.2.5. Interestingly, D2D-R increases the performance at an average of 6.28% on NDCG and 1.89% on HR. D2D loss, which is devised to make attributes (gender) indistinguishable, could accidentally diminish the negative gender discrimination to enhance recommendation performance.

Summary: U2U-R degrades recommendation performance to some extent, while D2D-R outperforms all compared methods and even slightly outperforms the original user embedding.

4.2.3 ***Efficiency.*** We use running time to evaluate the efficiency of unlearning methods. From Table 5, we observe that i) our proposed PoT-AU methods (U2U-R and D2D-R) significantly outperform InT-AU methods (Retrain and Adv-InT). This is because PoT-AU methods can be viewed as a fine-tuning process on an existing model, providing them with inherent efficiency compared to InT-AU methods; ii) By incorporating our proposed distinguishability loss to the original recommendation loss and retraining from scratch, Retrain outperforms Adv-InT. By serving as a baseline method, Retrain provides a new path for InT-AU methods to explore.

4.2.4 ***Effect of Trade-off Parameter.*** To investigate the robustness of our proposed two-component loss function, we study the effect of α, i.e., trade-off parameter. As shown in Figure 2 (Appendix B.1), we use AUC and NDCG@10 to represent unlearning and recommendation performance respectively. We observe that our proposed methods, especially U2U-R, are robust with different α. Reducing the value of α results in insignificant performance change for D2D-R, i.e., enhances unlearning performance and decreases recommendation performance.

4.2.5 ***Analysis of Embedding.*** To understand the mechanism of our proposed methods and compare the difference between two types of distinguishability losses, we analyze the distributions of user embedding. Specifically, we report the histograms for each dimension of user embedding where the users are grouped by gender. Figure 4 illustrates the user embedding of LightGCN trained on different datasets. From it, we have the following observations.

- Compared with original user embedding, U2U-R and D2D-R both change the distributions of user embedding. According to numeric results, i.e., Tables 3 and 4, these changes affect unlearning and recommendation performance.

- U2U loss does make users with different gender behave more similarly. However, U2U-R enhances XGB's attacking performance, instead of degrading it. We observe that the current implementation of U2U-R bounds the loss so tightly that it destroys the original distributions, making them collapse to the mean of the original embedding. This results in two *needle-shaped* distributions, which makes MLP difficult to distinguish them. As for XGB, it has a powerful fitting ability and is sometimes over-fitting. But in this case, XGB can over-fit the mean in the training set, and accurately distinguish needle-shaped distributions in the testing set. This observation also provides a reasonable explanation for U2U-R's performance drop in the recommendation task (Table 4).
- D2D loss is effective in enlarging the overlapping area of two gender distributions, which means it narrows the distance between two distributions. At the same time, it does not deform the original distributions. This brings superior performance in both unlearning and recommendation.

Summary: With the analysis of embedding distribution, we find that the performance degradation of U2U-R in the recommendation task is caused by a sharp change in embedding distribution. U2U distinguishability loss is bounded so tightly that it makes all users within the distribution move toward the mean.

5 RELATED WORK

5.1 Machine Unlearning

Machine unlearning aims to remove the influence of particular training data on a learned model [38]. Existing unlearning methods can be classified into the following two approaches.

Exact Unlearning: This approach aims to ensure that the target is unlearned as completely as retraining from scratch. Cao and Yang [10] transformed training data points into a reduced number of summations to enhance unlearning efficiency. Recently, Bourtoule et al. [7] proposed a partition-aggregation unlearning framework, i.e. SISA, which partitions the dataset into disjoint subsets, trains one model on each subset, and aggregates all models. This design reduces the retraining overhead to subsets.

Approximate Unlearning: This approach aims to estimate the influence of unlearning target, and directly remove the influence through parameter manipulation, i.e., updating parameters with the purpose of unlearning [20, 23, 42, 47]. In this approach, the influence of unlearning target is evaluated by influence function [32, 33], which is found to be fragile in deep learning [2]. Recent studies also point out that the influence of individual training data on deep models is intractable to compute analytically [22].

5.2 Recommendation Unlearning

RecEraser was proposed to achieve unlearning in recommender systems [11]. Following SISA's partition-aggregation framework, RecEraser groups similar data together, instead of random partitioning. In addition, RecEraser uses an attention-based aggregator to further enhance performance. Similarly, LASER also groups similar data together [36]. However, instead of parallel training, LASER employs sequential training. While this modification significantly enhances model utility, it comes at the cost of reduced efficiency. Lately, approximate unlearning is also investigated in recommendation to enhance efficiency [35].

5.3 Attribute Unlearning

Existing studies predominately focus on unlearning the input data samples, ignoring the latent attributes that are irrelevant to the training process. AU is first studied by [24] to unlearn particular attributes of a facial image, e.g., smiling, big nose, and mustache, by adding network blocks. As recommender systems potentially capture users' sensitive information, e.g., gender, race, and age, it is non-trivial to study AU in the recommendation scenario.

However, splitting the model into a feature extractor and a classifier, and adding a network block between them may not be universally applicable in the context of recommendation [24]. Adversarial training was used to achieve AU in recommender systems [18]. However, it is under the setting of In-Training AU (InT-AU), which manipulates the model parameters during training. Different from InT-AU setting, our PoT-AU setting is more strict and practical because i) we can only manipulate the model parameters when training is completed, ii) as the training data or other training information, e.g., gradients, are usually protected or discarded after training, we cannot get access to them to enhance performance, and iii) it is more flexible for recommendation platforms to manipulate the model based on unlearning requests without interfering with the original training process.

6 CONCLUSIONS

In this paper, we study the Post-Training Attribute Unlearning (PoT-AU) problem in recommender systems, which aims to protect users' attribute information instead of input data. To the best of our knowledge, we are the first to study this problem, which is more strict and practical than In-Training Attribute Unlearning (InT-AU) problem. There are two goals in the PoT-AU problem, i.e., making attributes indistinguishable, and maintaining comparable recommendation performance. To achieve the above two goals, we propose a two-component loss function, which consists of distinguishability loss and regularization loss, to optimize user embedding. We design two types of distinguishability losses from different perspectives, i.e., User-to-User (U2U) and Distribution-to-Distribution (D2D). We conduct extensive experiments on three real-world datasets to evaluate the effectiveness of our proposed methods, i.e., U2U-R and D2D-R. Results indicate that i) both methods achieved the unlearning goal, but D2D-R significantly outperformed U2U-R, and ii) D2D-R had a negative impact on recommendation performance, but D2D-R can enhance it. Generally speaking, D2D-R achieves both goals in the PoT-AU problem.

In this work, we focus on the attribute with binary labels. In the future, we will study the multiple-labels case and further exploit different implementations of the current U2U design. One of the advantages of U2U is that it can be directly generalized to the multiple-label case without modification.

ACKNOWLEDGMENTS

This work was supported in part by the "Ten Thousand Talents Program" of Zhejiang Province for Leading Experts (No. 2021R52001), and the National Natural Science Foundation of China (No. 72192823).

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] S Basu, P Pope, and S Feizi. 2021. Influence Functions in Deep Learning Are Fragile. In *ICLR*.

[3] Ghazaleh Beigi, Ahmadreza Mosallanezhad, Ruocheng Guo, Hamidreza Alvari, Alexander Nou, and Huan Liu. 2020. Privacy-aware recommendation with private-attribute protection using adversarial learning. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 34–42.

[4] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Mach. Learn.* 79, 1-2 (2010), 151–175. https://doi.org/10.1007/s10994-009-5152-4

[5] Albrecht Böttcher and David Wenzel. 2008. The Frobenius norm and the commutator. *Linear algebra and its applications* 429, 8-9 (2008), 1864–1885.

[6] Léon Bottou. 2012. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*. Springer, 421–436.

[7] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *Proceedings in the 42nd IEEE Symposium on Security and Privacy (SP)*.

[8] Department of Justice California. 2018. California Consumer Privacy Act. https://oag.ca.gov/privacy/ccpa.

[9] Government Canada. 2019. Personal Information Protection and Electronic Documents Act (S.C. 2000, c. 5). Website. https://laws-lois.justice.gc.ca/ENG/ACTS/P-8.6/index.html.

[10] Yinzhi Cao and Junfeng Yang. 2015. Towards making systems forget with machine unlearning. In *Proceedings in the 36th IEEE Symposium on Security and Privacy (SP)*. 463–480.

[11] Chong Chen, Fei Sun, Min Zhang, and Bolin Ding. 2022. Recommendation unlearning. In *Proceedings of the ACM Web Conference 2022*. 2768–2777.

[12] Chaochao Chen, Huiwen Wu, Jiajie Su, Lingjuan Lyu, Xiaolin Zheng, and Li Wang. 2022. Differential private knowledge transfer for privacy-preserving cross-domain recommendation. In *Proceedings of the ACM Web Conference 2022*. 1455–1465.

[13] T Chen, T He, and M Benesty. 2022. Machine learning challenge winning solutions. https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions

[14] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, et al. 2015. Xgboost: extreme gradient boosting. *R package version 0.4-2* 1, 4 (2015), 1–4.

[15] Najim Dehak, Reda Dehak, James R Glass, Douglas A Reynolds, Patrick Kenny, et al. 2010. Cosine similarity scoring without score normalization techniques.. In *Odyssey*. 15.

[16] Council EU. 2014. Council regulation (eu) on 2012/0011. Website. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52012PC0011.

[17] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters* 27, 8 (2006), 861–874.

[18] Christian Ganhör, David Penz, Navid Rekabsaz, Oleg Lesota, and Markus Schedl. 2022. Unlearning Protected User Attributes in Recommendations with Adversarial Training *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2142–2147. https://doi.org/10.1145/3477495.3531820

[19] Matt W Gardner and SR Dorling. 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment* 32, 14-15 (1998), 2627–2636.

[20] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. 2020. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9304–9312.

[21] Jacob Goldberger, Shiri Gordon, Hayit Greenspan, et al. 2003. An Efficient Image Similarity Measure Based on Approximations of KL-Divergence Between Two Gaussian Mixtures.. In *ICCV*, Vol. 3. 487–493.

[22] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. 2021. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11516–11524.

[23] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. 2020. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*. 3832–3842.

[24] Tao Guo, Song Guo, Jiewei Zhang, Wenchao Xu, and Junxiao Wang. 2022. Efficient Attribute Unlearning: Towards Selective Removal of Input Attributes from Feature Representations. *arXiv preprint arXiv:2202.13295* (2022).

[25] Zhongxuan Han, Xiaolin Zheng, Chaochao Chen, Wenjie Cheng, and Yang Yao. 2023. Intra and Inter Domain HyperGraph Convolutional Network for Cross-Domain Recommendation. In *Proceedings of the ACM Web Conference 2023*. 449–459.

[26] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm Transactions on Interactive Intelligent Systems (TIIS)* 5, 4 (2015), 1–19.

[27] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th International Conference on World Wide Web (WWW)*. 507–517.

[28] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*. 1661–1670.

[29] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[30] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*. 173–182.

[31] Jinyuan Jia and Neil Zhenqiang Gong. 2018. Attriguard: A practical defense against attribute inference attacks via adversarial machine learning. In *27th {USENIX} security symposium ({USENIX} security 18)*. 513–529.

[32] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*. 1885–1894.

[33] Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. 2019. On the accuracy of influence functions for measuring group effects. In *Advances in neural information processing systems*, Vol. 32.

[34] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data poisoning attacks on factorization-based collaborative filtering. *Advances in neural information processing systems* 29 (2016).

[35] Yuyuan Li, Chaochao Chen, Xiaolin Zheng, Yizhao Zhang, Biao Gong, Jun Wang, and Linxun Chen. 2023. Selective and collaborative influence function for efficient recommendation unlearning. *Expert Systems with Applications* (2023), 121025. https://doi.org/10.1016/j.eswa.2023.121025

[36] Yuyuan Li, Xiaolin Zheng, Chaochao Chen, and Junlin Liu. 2022. Making recommender systems forget: Learning and unlearning for erasable recommendation. *arXiv preprint arXiv:2203.11491* (2022).

[37] Alessandro B Melchiorre, Navid Rekabsaz, Emilia Parada-Cabaleiro, Stefan Brandl, Oleg Lesota, and Markus Schedl. 2021. Investigating gender fairness of recommendation algorithms in the music domain. *Information Processing & Management* 58, 5 (2021), 102666.

[38] Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. 2022. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299* (2022).

[39] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 452–461.

[40] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2019. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *2019 Network and Distributed Systems Security (NDSS) Symposium*.

[41] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web*. Springer, 291–324.

[42] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. 2021. Remember What You Want to Forget: Algorithms for Machine Unlearning. In *Advances in 34th Neural Information Processing Systems (NeurIPS)*.

[43] Yue Shi, Martha Larson, and Alan Hanjalic. 2014. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)* 47, 1 (2014), 1–45.

[44] Alexander J Smola, A Gretton, and K Borgwardt. 2006. Maximum mean discrepancy. In *13th International Conference, ICONIP 2006, Hong Kong, China, October 3-6, 2006: Proceedings*.

[45] Ilya O Tolstikhin, Bharath K Sriperumbudur, and Bernhard Schölkopf. 2016. Minimax estimation of maximum mean discrepancy with radial kernels. *Advances in Neural Information Processing Systems* 29 (2016).

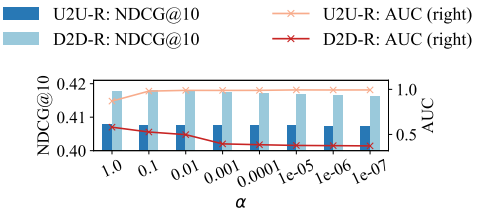
[46] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[47] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. 2023. Machine Unlearning of Features and Labels. In *Network and Distributed System Security (NDSS) Symposium 2023*.

[48] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems.. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, Vol. 17. 3203–3209.

[49] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P Gummadi. 2019. Fairness constraints: A flexible approach for fair classification. *The Journal of Machine Learning Research* 20, 1 (2019), 2737–2778.

[50] Zhilu Zhang and Mert Sabuncu. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems* 31 (2018).



(a) ML-100K: NMF

(b) ML-1M: NMF

(c) LFM-2B: NMF

(d) ML-100K: LightGCN

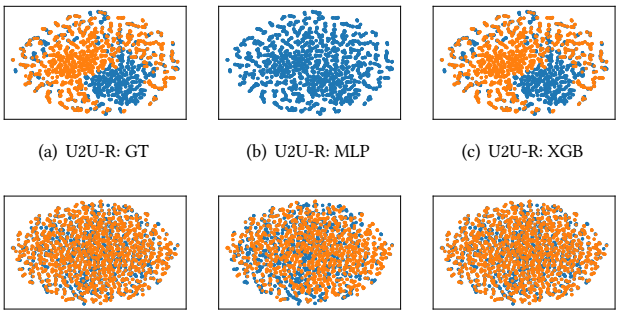
(e) ML-1M: LightGCN

(f) LFM-2B: LightGCN

Figure 2: Effect of α w.r.t. unlearning (AUC) and recommendation (NDCG@10) performance.

A EXPERIMENTAL SETTINGS

Hardware Information: All models and algorithms are implemented with Python 3.8 and PyTorch 1.9. We run all experiments on an Ubuntu 20.04 LTS System server with 48-core CPU, 256GB RAM and NVIDIA GeForce RTX 3090 GPU.



(a) U2U-R: GT (b) U2U-R: MLP (c) U2U-R: XGB

(d) D2D-R: GT (e) D2D-R: MLP (f) D2D-R: XGB

Figure 3: Gender prediction of different user embedding where GT denotes ground truth, the orange and blue dots represent the predictions of female and male respectively. The user embedding is trained in LightGCN on ML-1M.

Recommendation Models: To obtain the optimal performance, we use grid search to tune the hyper-parameters. For model-specific hyper-parameters, we follow the suggestions from their original papers. All model parameters are initialized with a Gaussian distribution $\mathcal{N}(0, 0.01^2)$. Specifically, we set the learning rate to 0.001 and the embedding size K to 16. The number of epochs is set to 50 for NMF and 400 for LightGCN.

Attackers: For MLP, we set the L2 regularization weight to 1.0 and the maximal iteration to 1000, leaving the other hyper-parameters at their defaults in scikit-learn 1.1.3. For XGB, we use the xgboost package, setting the hyper-parameters as their default values.

Our Methods: We set α and the learning rate to 1e-4 and 0.001 respectively for both methods. The number of epochs is set to 5,000 and 1,000 for U2U-R and D2D-R respectively.

B EXPERIMENTAL RESULTS

B.1 Effect of Trade-off Parameter

The effect of α is reported in Figure 2.

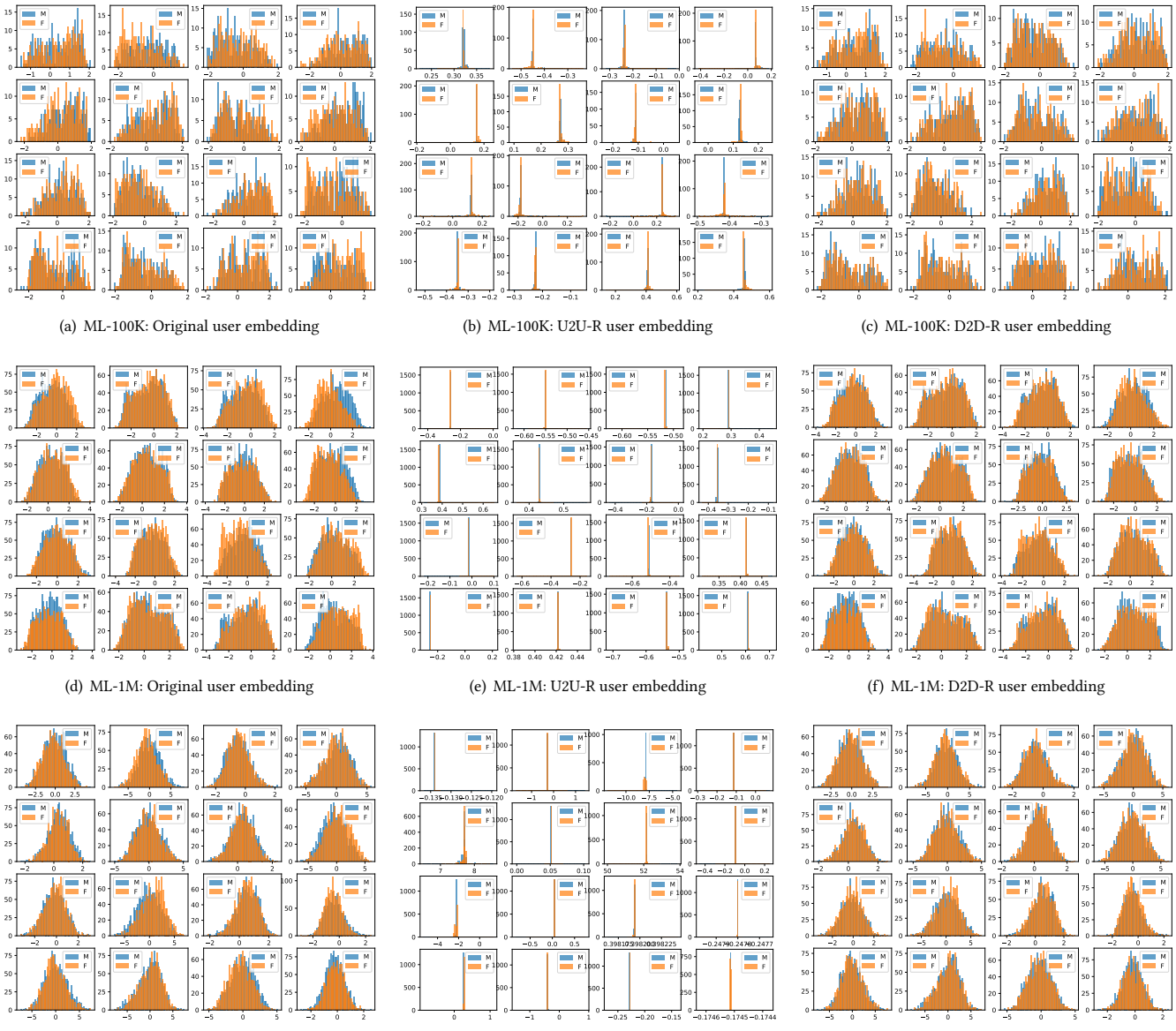
B.2 Visualization of Unlearning Performance

To visually analyze the results, we also reduce the embedding dimension to 2 using t-SNE [46] and plot the distribution of gender prediction in Figure 3. Recall that the aim of unlearning is to degrade the performance of attackers. From Figure 3, we observe that U2U-R displays a significant degree of diversity towards different attackers, i.e., MLP and XGB.

- For MLP attack, U2U-R fools the attacker by flipping all labels into the same class, which significantly changes the distribution of user embedding. This observation can also be illustrated in Figure 4.
- For XGB attack, U2U-R cannot degrade the attacking performance. The visualized outcome of U2U-R, i.e., Figure 3(c), depicts minimal disparity from the actual ground truth, i.e., Figure 3(a).

B.3 Analysis of Embedding

The histograms of user embedding are reported in Figure 4.



(a) ML-100K: Original user embedding (b) ML-100K: U2U-R user embedding (c) ML-100K: D2D-R user embedding

(d) ML-1M: Original user embedding (e) ML-1M: U2U-R user embedding (f) ML-1M: D2D-R user embedding

(g) LFM-2B: Original user embedding (h) LFM-2B: U2U-R user embedding (i) LFM-2B: D2D-R user embedding

Figure 4: The distributions of different user embedding on different datasets (LightGCN). Each mini-plot represents one dimension of user embedding where F and M denote female and male respectively. To compare the two distributions, i.e., female and male, more accurately, we down-sample the males so that the number of males is equal to the number of females.