Membership Inference Attacks Against Recommender Systems

Minxing Zhang[1,2*] Zhaochun Ren[1*†] Zihan Wang[1*] Pengjie Ren[1]
Zhumin Chen[1] Pengfei Hu[1] Yang Zhang[2†]
[1]Shandong University [2]CISPA Helmholtz Center for Information Security

ABSTRACT

Recently, recommender systems have achieved promising performances and become one of the most widely used web applications. However, recommender systems are often trained on highly sensitive user data, thus potential data leakage from recommender systems may lead to severe privacy problems.

In this paper, we make the first attempt on quantifying the privacy leakage of recommender systems through the lens of membership inference. In contrast with traditional membership inference against machine learning classifiers, our attack faces two main differences. First, our attack is on the user-level but not on the data sample-level. Second, the adversary can only observe the ordered recommended items from a recommender system instead of prediction results in the form of posterior probabilities. To address the above challenges, we propose a novel method by representing users from relevant items. Moreover, a shadow recommender is established to derive the labeled training data for training the attack model. Extensive experimental results show that our attack framework achieves a strong performance. In addition, we design a defense mechanism to effectively mitigate the membership inference threat of recommender systems.[1]

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Security and privacy**;

KEYWORDS

membership inference attack, recommender system, membership leakage

ACM Reference Format:
Minxing Zhang, Zhaochun Ren, Zihan Wang, Pengjie Ren, Zhunmin Chen, Pengfei Hu, Yang Zhang. 2021. Membership Inference Attacks Against Recommender Systems. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3460120.3484770

*These authors contributed equally to this work.
†Corresponding author.
[1]Our code is available at https://github.com/minxingzhang/MIARS.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea.
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8454-4/21/11…$15.00
https://doi.org/10.1145/3460120.3484770

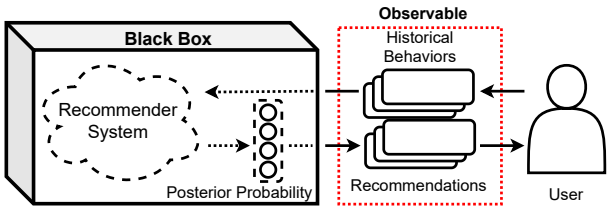


Figure 1: An example of recommender systems.

1 INTRODUCTION

As one of the most prevalent services in current web applications, recommender systems have been applied in various scenarios, such as online shopping, video sharing, location recommendation, etc. A recommender system is essentially an information filtering system, relying on machine learning algorithms to predict user preferences for items. One mainstream method in this space is collaborative filtering, which is based on traditional methods such as matrix factorization and latent factor model, predicting a user's preference from their historical behaviors combined with other users' similar decisions [19, 41]. Another is the content-based recommendation [35, 51]. This approach aims to distinguish users' likes from dislikes based on their metadata (such as descriptions of the items and profiles of the users' preferences). Recent advancement of deep learning techniques further boosts the performance of recommender systems [16].

The success of recommender systems lies in the large-scale user data. However, the data in many cases contains sensitive information of individuals, such as shopping preference, social relationship [1], and location information [44]. Recently, various research has shown that machine learning models, represented by machine learning classifiers, are prone to privacy attacks [6, 8, 10, 18, 22, 28–32, 34, 37, 39, 43, 45]. However, the privacy risks stemming from recommender systems have been left largely unexplored.

1.1 Our Contributions

In this paper, we take the first step quantifying the privacy risks of recommender systems through the lens of membership inference. Compared to previous membership inference attacks against machine learning classifiers [39, 43], our attack faces two main differences. First, the goal of our attack is to determine whether a user's data is used by the target recommender. This indicates our attack is on the user-level while most of the previous attacks focus on the sample-level [39, 43]. Unlike sample-level membership inference, user-level membership inference has a broader scope as mentioned in previous works [45], and it can help us gain a comprehensive understanding of recommender systems' privacy risks. Second, from the adversary's perspective, only ranking lists

of items are available from the target recommender which raises several technical challenges:

- In our attack, as Figure 1 depicts, the adversary can only observe lists of items, even though recommender systems have already calculated posterior probabilities before making decisions. This setting is prevalent in the real world, such as the service provided by Amazon or Netflix. Besides, exposing less information can protect the intellectual property of recommendation service providers [39]. On the contrary, in classical membership inference attacks against classifiers, posterior probabilities used for decisions can be accessed by the adversary [31, 37, 39, 43].
- Several recent membership inference attacks against classifiers focus on the decision-only (i.e., label-only) scenario [10, 28]. However, these studies either rely on the target model to label a shadow dataset [28] or use adversarial examples [10, 28], which are not practical when targeting recommender systems in the real world. Therefore, we aim for a new method to extract information from decision results for the attack model.
- Unlike classical classifiers, the outputs of recommender systems are ranking lists of items other than unordered labels. In that case, the order information plays an important role, and can substantially facilitate user preference predictions. Therefore, it is necessary for our attack model to capture order information from recommended items, which is still ignored by previous membership inference attack methods.

Threat Model. The goal of the adversary is to conduct a membership inference attack against the target recommender system, inferring whether a target user's data is used to train the target recommender or not. However, such attacks can lead to severe security and privacy risks. Specifically, when a recommender system is trained with data from people with certain sensitive information, such as health status, knowing a user's data being part of the recommender's training data directly leaks their private information. Moreover, membership inference also gains the adversary information about the target recommender's training data, which may jeopardize the recommender's intellectual property, since collecting high-quality data often requires a large amount of efforts [18, 27]. From a different angle, a user can also use membership inference as a tool to audit whether their data is used by the target recommender.

We assume the adversary has black-box access to the target recommender, the most difficult setting for the adversary [43]. Instead of posterior probabilities for recommendations, only relevant items for users are available, such as rating or purchase and recommended items. Due to the knowledge, a shadow recommender is established to derive labeled training data, for the attack model better inferring membership status in the target recommender.

Attack Method. For user-level membership inference, we need to summarize each user's *feature vector*, based on interactions between the target recommender and them, as the input to the attack model. However, compared to the previous work of membership inference against classifiers, the adversary can only observe the recommended items from a recommender system instead of posterior probabilities as prediction results. Thus, in the first step, the adversary constructs a user-item matrix for ratings with a dataset used to generate feature vectors. Then, they factorize this matrix into two low-dimensional matrices, namely user matrix and item matrix. Each item's feature vector can be represented by the corresponding row vector in the item matrix. For each user, the adversary extracts two sets of items (one set contains items the user is recommended and the other contains the items the user interacted with) and calculates these two sets' center vectors, respectively. The difference between these two center vectors for each user describes how accurate the recommender is for this user. In that case, lower difference indicates a user's data is more likely to be used to train the recommender. Therefore, we use this difference as the input to the attack model, i.e., user feature vector. The adversary generates all the labeled training dataset for their attack model with the help of the shadow recommender. To launch the attack, the adversary generates the target user's feature vector following the same steps and obtains the prediction from the attack model.

Evaluation. To evaluate our attack, the adversary is assumed to have a shadow dataset that comes from the same distribution as the target recommender's training data, and know the target recommender's algorithm. These assumptions are gradually relaxed based on our empirical evaluation.

Our experiments are performed on three benchmark recommendation datasets, i.e., the Amazon Digital Music (ADM) [15], Lastfm-2k (lf-2k) [4], and Movielens-1m (ml-1m) [13]. The recommendation algorithms we focus on include Item-Based Collaborative Filtering (Item), Latent Factor Model (LFM), and Neural Collaborative Filtering (NCF). Evaluation results demonstrate that our attack is able to achieve an excellent performance.

- In general, when the adversary knows the distribution of the target dataset and the target recommender's algorithm, the attack performance is extremely strong. For instance, when the target recommender uses NCF on the ADM dataset, our attack achieves an AUC of 0.987. Also, when the target algorithm is Item or NCF, our attack achieves better performances.
- When the adversary is not aware of the target recommender's algorithm, attack performances are reduced but still strong. For instance, on the lf-2k dataset, when the target recommender uses Item and the shadow recommender uses NCF, the attack performance decreases from an AUC of 0.929 to an AUC of 0.827. On the other hand, in some cases, the attack performances even increase. For instance, on the ml-1m dataset, when the target recommender uses Item and the shadow recommender uses LFM, the attack's AUC increases from 0.871 to 0.931.
- We further relax the assumption of the shadow dataset. Evaluation shows that even under such a scenario, our attack still achieves good performances in general. Note that, in some cases, when the adversary knows less about the target recommender, the attack even performs better. This demonstrates the good generalization ability of our attack.

In conclusion, the experimental results show the effectiveness of our attack, indicating that recommender systems are indeed vulnerable to privacy attacks.

Defense. To mitigate the recommender's privacy risk, we propose a defense mechanism, namely *Popularity Randomization*. Popularity Randomization is deployed when the target recommender recommends items to its non-member users. The normal strategy in such case is to provide non-member users with the most popular items. However, to defend the attack, we enlarge the set of popular items, and randomly select a subset of them for recommendation. Intuitively, while preserving the recommendation performance for non-member users, this approach enriches the randomness of the recommendation.

Experimental results show that Popularity Randomization can effectively mitigate membership inference. For instance, Popularity Randomization (with a 0.1 ratio of recommendations to candidates) decreases the attack performances by more than 12%, 33%, and 41% when the target algorithm is Item, LFM or NCF respectively. Through analyses, we observe that Popularity Randomization has the greatest impact on the attack targeting on NCF.

2 METHOD

In this section, we first present some necessary definitions in Section 2.1, and then introduce the threat model for the membership inference attack against recommender systems in Section 2.2. Next, we give overviews for recommender systems in Section 2.3 and our attack model in Section 2.4. Finally, we detail the proposed membership inference attack methods in Section 2.5.

2.1 Definitions

We present the following definitions for the attack process:

- **Target Recommender**, trained on the **Target Dataset**, is the recommender system attacked by the adversary.
- **Shadow Recommender**, trained on the **Shadow Dataset**, is a recommender system built to infer the membership status of the target recommender and generate training data for the attack model.
- **Members** are the users whose data is used to train the recommender, while **Non-Members** are the ones whose data is not used.
- **Personalized Recommendation Algorithms** learn members' preferences from historical behaviors (such as purchases or ratings), which are also called **Interactions**. **Non-Personalized Recommendation Algorithms** are based on the predetermined rule, such as selecting the most popular or highest-rated items. According to different recommendation methods, members and non-members are provided with recommended items, which are also called **Recommendations**.
- **Feature Vectors** show the latent features, indicating item attributes or user preferences.
- **Attack Model** is used to infer whether the target user is a member, and trained on the dataset generated from the shadow recommender.

2.2 Threat Model

Adversary's Goal. The adversary aims to infer whether a user's data is used by a target recommender. In fact, knowing a certain user's data being used by a recommender system directly leaks their private information. Besides, knowing a user being part of the dataset can also allow the adversary to gain extra information of the target recommender's dataset. This directly violates the intellectual property of the target recommender, since it is very expensive to collect high-quality training data. Alternatively, membership inference can also be used as an auditing tool by a user to find out whether their data is used by the target recommender.

Adversary's Knowledge. We assume an adversary has only black-box access to the target recommender. That is, adversary can only observe the items recommended to a target user (i.e., *recommendations*), and the user's history (i.e., *interactions*), such as rating and purchase, instead of posterior probabilities for recommendation predictions. In that case, the adversary needs to profile users by their interactions and recommendations. Meanwhile, a shadow recommender is built to generate labeled data for the attack model, since ground truth membership is unavailable from the target recommender.

2.3 Recommender Systems

In this section, the framework of recommender systems is briefly introduced.

Recommendation algorithms output recommended items based on the information learnt from input. In the paper, two types of recommendation algorithms are mainly involved: personalized and non-personalized recommendation algorithms. For members, items are recommended according to the preferences of members. Meanwhile, lacking non-members' data, non-personalized recommendation algorithms are conducted, and provide most popular items for non-members. Specifically, Item-Based Collaborative Filtering (Item) [40], Latent Factor Model (LFM) and Neural Collaborative Filtering (NCF) [16] are adopted as the personalized recommendation algorithms for members. As for the non-personalized recommendation algorithm, the most popular items are provided to non-members, which is also called the popularity recommendation algorithm in the paper. We briefly introduce the above algorithms as follows:

- **Item** calculates the similarity between items aiming to find the ones which are closed to users' likes.
- **LFM** builds a latent space to bridge user preferences and item attributes.
- **NCF** combine the deep learning technology with collaborative filtering to enhance the recommendation performances.
- Users are provided with the most popular items by the **popularity recommendation algorithm**.

In general, a recommender system $\mathcal{A}_{RS}$ learns user preferences from the interactions, sometimes with the external knowledge (such as gender and location information) for users. According to the predicted preferences, the recommender system provides users with multiple items. This procedure can be formulated as:

$$\mathcal{A}_{RS} : (\mathcal{I}_{RS}, \mathcal{K}_{RS}) \rightarrow \mathcal{R}_{RS},$$

where $\mathcal{A}_{RS}$ is a recommender system learning the preferences of users from their interactions $\mathcal{I}_{RS}$ and the external knowledge $\mathcal{K}_{RS}$. And $\mathcal{R}_{RS}$ denotes recommended items to users. In the paper, we mainly use the interactions of users. Thus, we define a recommender

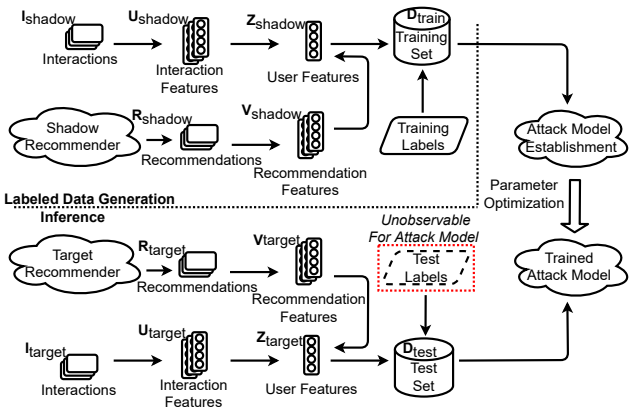


Figure 2: The framework of the membership inference attack against a recommender system.

system as:

$$\mathcal{A}_{RS} : \mathcal{I}_{RS} \rightarrow \mathcal{R}_{RS},$$

where $\mathcal{I}_{RS}$ is a set of lists of interactions for users and $\mathcal{R}_{RS}$ is a set of ordered lists of recommendations for users. Concretely, $\mathcal{I}_{RS} = \{L_I^n\}_{n=1}^{N_u}$ and $\mathcal{R}_{RS} = \{L_R^n\}_{n=1}^{N_u}$, where L_I^n is the list of interactions and L_R^n is the ordered list of recommendations for the n^{th} user, and N_u is the number of users.

2.4 Attack Overview

In this section, we give an overview of our attack. As Figure 2 demonstrated, the attack process follows three steps: Labeled Data Generation, Attack Model Establishment, and Parameter Optimization.

Labeled Data Generation. To represent items, an item matrix is derived, by factorizing a user-item rating matrix. Due to the black-box access to the target recommender for the adversary, a shadow recommender is built to generate labeled training data for the attack model. Moreover, we represent interactions and recommendations of users using corresponding feature vectors. After that, a user is profiled by the difference between two centers of their interactions and recommendations. And each user is labeled with 1 or 0, indicating they are a member or non-member.

Attack Model Establishment. Inspired by [39], a two-hidden-layer Multi-Layer Perceptron (MLP) is utilized as the attack model A_{attack} to infer membership status. Each hidden layer is followed by a ReLU activation layer. And a softmax function is used as the output layer to predict the probability of the membership.

Parameter Optimization. After *Labeled Data Generation* and *Attack Model Establishment*, as shown in Figure 2, the adversary updates parameters of the attack model. In the inference stage, the test dataset for target users are established following the same steps as training data generation. The membership status for target users is inferred by the trained attack model.

2.5 Membership Inference Attack

In this section, we detail our proposed membership inference attack against a recommender system. As mentioned before, the whole attack consists of three steps to achieve the adversary's goal: Labeled Data Generation, Attack Model Establishment, and Parameter Optimization.

Labeled Data Generation. Training data is required during the $\mathcal{A}_{attack}$ optimization process. However, the adversary cannot obtain membership status directly from the target recommender $\mathcal{A}_{target}$. To address this problem, a shadow recommender $\mathcal{A}_{shadow}$ is developed to mimic the dataset and recommendation algorithm of the target recommender.

As mentioned in Section 1.1, only recommended item lists from target recommender systems can be observed. Inspired by the previous works [17, 24], *matrix factorization* is adopted to project users and items into a shared latent space. Specifically, a $p \times q$ user-item matrix $\mathbf{M}^f$ is built using ratings of users to items, where p and q are the number of users and items respectively. Values in $\mathbf{M}^f$ are ratings ranging from 1 to 5, indicating how much users prefer these items. Then, $\mathbf{M}^f$ is factorized into two low-dimensional matrices, namely latent user matrix $\mathbf{M}^{user} \in \mathcal{R}^{p \times l}$ and latent item matrix $\mathbf{M}^{item} \in \mathcal{R}^{q \times l}$, where we denote l as the dimension of the latent feature space. We apply matrix factorization to find optimized $\mathbf{M}^{user}$ and $\mathbf{M}^{item}$ by minimizing the loss function L_{MF}:

$$L_{MF} = \left\| \mathbf{M}^f - \hat{\mathbf{M}}^f \right\|_2 \quad where \quad \hat{\mathbf{M}}^f = \mathbf{M}^{item} \cdot \mathbf{M}^{user\mathrm{T}},$$

where $\hat{\mathbf{M}}^f$ is a predicted user-item matrix which contains the predicted scores of users rating items. Besides, $\mathbf{M}^{user}$ and $\mathbf{M}^{item}$ present the predicted preferences of users and the predicted attributes of items, respectively. Each row of the item matrix $\mathbf{M}^{item}$ represents the feature vector of the corresponding item. Note that, since $\mathbf{M}^{user}$ may not cover all users in the shadow and target recommenders, $\mathbf{M}^{user}$ is not used to represent users.

To this end, training data for the attack model can be generated from $\mathcal{A}_{shadow}$. The shadow dataset $\mathcal{D}_{shadow}$ are split into two disjoint sets for members and non-members, which are denoted by $\mathcal{D}^{in}_{shadow}$ and $\mathcal{D}^{out}_{shadow}$, respectively. These datasets are composed of 3-tuples in the form of $(uID, iID, score)$, indicating scores rated by users to items. For instance, a 3-tuple $(2, 3, 4)$ in datasets means that the 2^{nd} user rates the 3^{rd} item a score of 4. Ratings in $\mathcal{D}^{in}_{shadow}$ and $\mathcal{D}^{out}_{shadow}$ can be seen as interactions of users to items, and sets of interaction lists for members and non-members can be obtained, denoted as $\mathcal{I}^{in}_{shadow}$ and $\mathcal{I}^{out}_{shadow}$, respectively. In that case, each user has a list of interactions. For example, if a user rates the 2^{nd}, 4^{th}, 6^{th} and 8^{th} items, the corresponding interaction list is $\{2, 4, 6, 8\}$.

Next, $\mathcal{A}_{shadow}$ is established to mimic $\mathcal{A}_{target}$, and provides users with recommendations according to their preferences. The sets of recommendation lists for members and non-members are denoted by $\mathcal{R}^{in}_{shadow}$ and $\mathcal{R}^{out}_{shadow}$, respectively. Similar as interactions, each user is associated with a list of recommendations. However, $\mathcal{R}^{in}_{shadow}$ and $\mathcal{R}^{out}_{shadow}$ are sets of *ordered* lists of recommendations. Formally, the recommendation process can be formulated as follows:

$$\mathcal{A}_{shadow} : f_{per}(\mathcal{I}^{in}_{shadow}) = \mathcal{R}^{in}_{shadow}$$

$$\mathcal{A}_{shadow} : f_{pop}(\mathcal{I}^{in}_{shadow}) = \mathcal{R}^{out}_{shadow},$$

where f_{per} performs a personalized recommendation algorithm based on the behaviors of members. Meanwhile, since non-members'

data is unavailable to $\mathcal{A}_{shadow}$, f_{pop} performs the popularity recommendation algorithm (a non-personalized recommendation algorithm) based on the statistical results from $\mathcal{I}^{in}_{shadow}$. Besides, $\mathcal{I}^{in}_{shadow}$ is a set of lists of interactions for members, and $\mathcal{R}^{in}_{shadow}$ and $\mathcal{R}^{out}_{shadow}$ are sets of ordered lists of recommended items for members and non-members respectively.

Using item feature representations, we can vectorize the interaction and recommendation sets as follows:

$$\mathcal{I}_{shadow} \xrightarrow{vec} \mathcal{U}_{shadow}$$
$$\mathcal{R}_{shadow} \xrightarrow{vec} \mathcal{V}_{shadow},$$

where $\mathcal{U}_{shadow}$ and $\mathcal{V}_{shadow}$ are sets of lists of the feature vectors for the corresponding items in $\mathcal{I}_{shadow}$ and $\mathcal{R}_{shadow}$.

Given that each user has a list of interactions and is provided with an ordered list of recommendations, the adversary is able to represent users by their relevant items. To be specific, for the i^{th} user, the representation is generated with the following two steps:

1) Center vectors of the interactions' and recommendations' feature vectors of the i^{th} user are calculated:
$$\overline{\mathcal{U}}_i = \sum_j \mathcal{U}_{i,j}/N_i^{int}$$
$$\overline{\mathcal{V}}_i = \sum_j \mathcal{V}_{i,j}/N_i^{rec},$$
where $\overline{\mathcal{U}}_i$ and $\overline{\mathcal{V}}_i$ are the center vectors of the feature vectors for the interactions and recommendations of the i^{th} user, and N_i^{int} and N_i^{rec} are the corresponding quantities. Besides, $\mathcal{U}_{i,j}$ and $\mathcal{V}_{i,j}$ are the feature vectors for the j^{th} interaction and recommendation of the i^{th} user, respectively.
2) The difference between the two center vectors are obtained:
$$\mathbf{z}_i = \overline{\mathcal{U}}_i - \overline{\mathcal{V}}_i.$$
In the paper, we employ $\mathbf{z}_i$ as the feature vector for the i^{th} user, which takes not only the user's history but also the predicted preference into consideration.

Meanwhile, each user is assigned a label of 1 or 0, indicating their membership (i.e., 1 means member and 0 means non-member). The training dataset $\mathcal{D}_{train} = \{(feature_i, label_i)\}_{i=1}^{N}$ contains feature vectors and labels of all users, where the pair $(feature_i, label_i)$ denotes the feature vector and label for the i^{th} user.

Attack Model Establishment. Inspired by [39], a MLP is established as the attack model $\mathcal{A}_{attack}$. The output of $\mathcal{A}_{attack}$ is a 2-dimension vector representing probabilities for the membership status. For the i^{th} user, the prediction can be formulated as follows:

$$\mathbf{h}_1 = \text{ReLU}(\mathbf{W}_1\mathbf{z}_i + \mathbf{b}_1)$$
$$\mathbf{h}_2 = \text{ReLU}(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$$
$$\mathbf{y}_i = \text{softmax}(\mathbf{h}_2),$$

where $\mathbf{z}_i$ is the input of $\mathcal{A}_{attack}$ as well as the i^{th} user's feature vector in our attack. And $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are the parameters updated in the training process. ReLU$(\cdot)$ is an activation function working on the outputs of two hidden layers, and softmax$(\cdot)$ is used for normalization which is required by the cross-entropy loss. Besides, $\mathbf{h}_1$ and $\mathbf{h}_2$ are the results of two hidden layers after ReLU$(\cdot)$. And $\mathbf{y}_i$ is the predicted result for the input $\mathbf{z}_i$, which is a 2-dimension vector indicating the possibilities of $\mathbf{z}_i$ belonging to members and non-members, respectively.

Parameter Optimization. In this section, the parameter optimization process for the attack model is described. Stochastic gradient descent is adopted to update parameters, aiming to minimize the cross-entropy loss function L_{MLP}:

$$L_{MLP} = -\sum_{i=1}^{N_{train}} (\mathrm{y}_i^* \log \mathrm{y}_i' + (1-\mathrm{y}_i^*)\log(1-\mathrm{y}_i')),$$

where y_i^* is the ground truth label for the i^{th} target user. And y_i' is the predicted possibility of the i^{th} target user belonging to members. Besides, N_{train} is the size of training data.

Test data $\mathcal{D}_{test}$ for the attack model is generated from the target recommender in the same way as the training data. The trained attack model $\mathcal{A}'_{attack}$ conduct a prediction given a target user feature vector $\mathbf{z}_{target}$, i.e., $\mathcal{A}'_{attack}(\mathbf{z}_{target}) = \mathbf{y}_{target}$, where $\mathbf{y}_{target} = \begin{pmatrix} a \\ b \end{pmatrix}$ is a 2-dimension vector, and the values of a and b indicate the probabilities that the target user belongs to non-members and members respectively. According to the predicted results, the adversary infers the membership status of the target user. Concretely, when $a < b$, the target user is predicted to be a member. Otherwise, they are predicted to be a non-member.

3 EXPERIMENTS

In this section, we first demonstrate experimental setup, including recommendation methods, datasets, preprocessing process, evaluation metrics, implementation details, and notations in Section 3.1. Then, we evaluate the performances of original recommender systems in Section 3.2. Moreover, we investigate membership inference attacks against recommender systems in Section 3.3 and conduct detailed analyses on the influences of hyperparameters in Section 3.4. Finally, we present extensive analysis to comprehensively investigate the attack model in Section 3.5.

3.1 Experimental Setup

Recommendation Methods. Personalized recommendation algorithms are adopted for members, including Item-Based Collaborative Filtering (Item) [40], Latent Factor Model (LFM) and Neural Collaborative Filtering (NCF) [16]. Meanwhile, due to the lack of non-members' data, a recommender system provides non-members with the most popular items, which is named the *popularity recommendation algorithm* in our paper.

Datasets. We utilize three real-world datasets in our experiments, including Amazon Digital Music (ADM) [15], Lastfm-2k (lf-2k) [4], and Movielens-1m (ml-1m) [13], to evaluate our attack strategies. All these datasets are commonly-used benchmark datasets for evaluating recommender systems. Note that only ratings in these datasets are used for our evaluation in the experiments. Scores range from 1 to 5, which indicates how much users like musics (ADM and lf-2k) or movies (ml-1m).

Preprocessing. For each dataset, we divide it into three disjoint subsets, i.e. a shadow dataset, a target dataset and a dataset for extracting item features. Then, the following processing methods are implemented to these subsets:

- To generate feature vectors for users, the dataset for item feature should contain all items of the target and shadow recommenders.
- For the shadow or target dataset, we further divide it into two disjoint parts, which are used to conduct recommendations to members and non-members, respectively. Moreover, following the previous work [16], we filter out the users who have less than 20 interactions.

In our experiments, recommender systems conduct recommendations based on implicit feedback. We assign values of 1 to the user-item pairs when there exist interactions between these users and items. And other user-item pairs are assigned 0. In LFM and NCF, recommender systems require both positive and negative instances. We randomly sample negative user-item pairs from the pairs scoring 0 and regard the pairs assigned 1 as positive instances. We keep the same number of negative samples as positive samples for the dataset balance.

Evaluation Metrics. We use AUC (area under the ROC curve) as the metric to evaluate attack performances. Following the definition of the attack, we regard members as *positive data points* and non-members as *negative data points*. AUC indicates the proportion of the prediction results being positive to negative. For example, if the attack model utilizes Random Guess to conduct a membership inference, the AUC is close to 0.5.

Implementation Details. We build a MLP with 2 hidden layers as the attack model. The first hidden layer has 32 units and the second layer has 8 units, both followed by a ReLU layer. And we utilize a softmax layer as the output layer. For the optimizer, we employ Stochastic Gradient Descent (SGD) with a learning rate of 0.01 and a momentum of 0.7. Besides, we use cross entropy as the loss function and the model is trained for 20 epochs.

In the paper, members are recommended by Item, LFM and NCF while non-members are recommended by the popularity recommendation algorithm. Note that, Item and the popularity recommendation algorithm do not need the iterative process of updating parameters. The detailed model configurations of LFM and NCF are shown as follows:

- **LFM.** We adopt the SGD algorithm to update parameters with a learning rate of 0.01 and conduct LFM with a regularization coefficient of 0.01 to enhance the model's generalization ability. Then we train the model for 20 epochs.
- **NCF.** We use Adam as the optimizer with a learning rate of 0.001. And we build the MLP part with 3 hidden layers containing 64, 32 and 16 hidden units respectively. Meanwhile, the embedding size of the Generalized Matrix Factorization (GMF) part is 8 [16]. In addition, the number of negative samples corresponding to per positive sample is set to 4. Then we train the model for 20 epochs with a batch size of 256.

Notations. To clarify the experimental settings, notations are demonstrated in Table 1, where "∗" stands for any algorithm or dataset used to construct or train the shadow or target model. For example, "A∗" could be the combination of "ADM+Item", "ADM+LFM" or "ADM+NCF". Note that, not all possible combinations are listed due to the space limit.

Table 1: Notations for different settings. "∗" stands for any algorithm or dataset used to construct or train the shadow or target model.

Notation	Illustrations
A∗	Trained on the ADM dataset.
L∗	Trained on the lf-2k dataset.
M∗	Trained on the ml-1m dataset.
∗I	Implemented by Item algorithm.
∗L	implemented by LFM algorithm.
∗N	implemented by NCF algorithm.
AI∗∗	The shadow recommender is implemented by Item algorithm on the ADM dataset.
∗∗AI	The target recommender is implemented by Item algorithm on the ADM dataset.
AIMN	The shadow recommender is implemented by Item algorithm on the ADM dataset, and the target recommender is implemented by NCF algorithm on the ml-1m dataset.

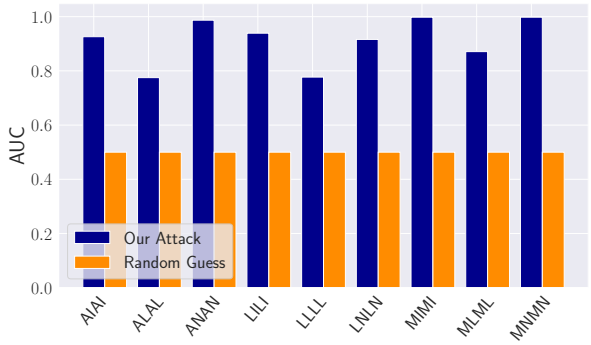


Figure 3: The attack performances under the assumption I.

In the experiments, there are two kinds of combinations in the paper (i.e., 2-letter and 4-letter combinations). For the 2-letter combinations, the first letter, i.e., "A," "L" or "M", indicates the shadow (or target) dataset, and the second letter, i.e., "I," "L" and "N", indicates the recommendation algorithm. For the 4-letter combinations, the first two letters represent the dataset and algorithm of the shadow recommender and the last two letters denote the dataset and algorithm of the target recommender. For instance, "AIMN" means that the adversary establishes a shadow recommender with Item on the ADM dataset to attack a target recommender implemented by NCF on the ml-1m dataset.

3.2 Recommendation Performance

We adopt HR@k as the metric to evaluate the recommendation performance, where $k = 100$ is consistent with the experimental setting and Hit Rate (HR) presents the proportion of recommendations including the ground truth. We can see from the results in Table 2 that, in general, recommender systems achieve the best

Table 2: The HR@100 of the shadow and target recommenders.

ADM	Item	LFM	NCF
shadow	0.222	0.119	0.116
target	0.224	0.204	0.123
lf-2k	Item	LFM	NCF
shadow	0.652	0.478	0.625
target	0.650	0.468	0.637
ml-1m	Item	LFM	NCF
shadow	0.943	0.856	0.721
target	0.951	0.860	0.713

Table 3: The AUC of the attack model against the ADM dataset, under the assumption II.

Shadow Algorithm	Target Algorithm		
	Item	LFM	NCF
Item	**0.926**	**0.885**	0.750
LFM	0.843	0.775	0.554
NCF	0.513	0.494	**0.987**

Table 4: The AUC of the attack model against the lf-2k dataset, under the assumption II.

Shadow Algorithm	Target Algorithm		
	Item	LFM	NCF
Item	**0.939**	0.796	0.793
LFM	0.732	0.777	0.774
NCF	0.827	**0.809**	**0.916**

Table 5: The AUC of the attack model against the ml-1m dataset, under the assumption II.

Shadow Algorithm	Target Algorithm		
	Item	LFM	NCF
Item	**0.998**	0.792	0.706
LFM	0.931	0.871	0.670
NCF	0.976	**0.914**	**0.998**

performance on the ml-1m dataset. Specifically, the shadow recommender obtains a hit rate of 0.856 when using LFM on the ml-1m dataset.

3.3 Attack Performance

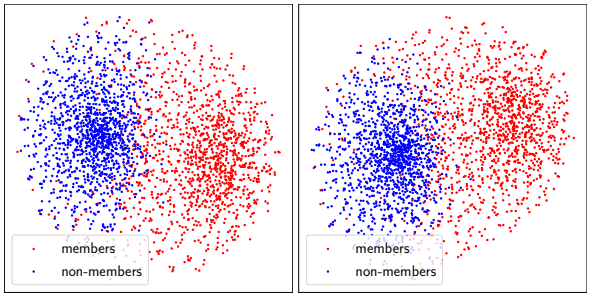
We perform experiments on the ADM, lf-2k and ml-1m datasets with three typical recommendation algorithms, including Item, LFM and NCF. Experimental results show that our method is able to achieve strong attack performances. We draw the following conclusions:

Assumption I. First of all, the target recommender's dataset distribution and algorithm are available. And, in the paper, these information is the most knowledge that the adversary can gain from the target recommender. The complete results are shown in Figure 3, in which we compare our attack with Random Guess. Then, data points (members and non-members) are visualized in a 2-dimension space by t-distributed Stochastic Neighbor Embedding (t-SNE) [50]. Figure 4 shows the results of the shadow and target distributions for two datasets, where the red points represent members and the blue points represent non-members. According to the attack and visualization results, we conclude that:

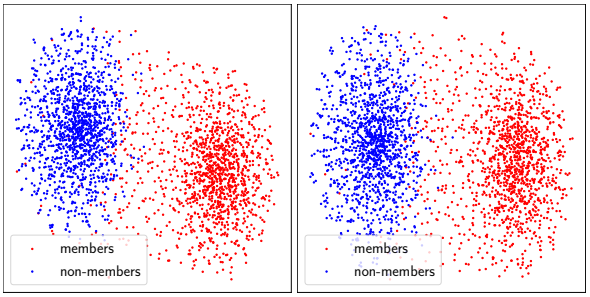
- In general, under the assumption that the shadow recommender knows the algorithm and dataset distribution of the target recommender, our attack is very strong. There are two main reasons for the effectiveness. First, data points of members and non-members are tightly clustered separately. Due to the different recommendation methods for members and non-members, generally, the interactions and recommendations of members are more relevant. In that case, the intra-cluster distance of members and non-members is much smaller than the inter-cluster distance between them, so that members can be easily distinguished. Second, as the adversary has the most knowledge of the target recommender, the shadow recommender can well mimic the target recommender. Thus the attack model, which is trained on the ground truth membership generated from the shadow recommender, is able to conduct a membership inference accurately.
- When the target recommender uses Item or NCF, our attack performs considerably better on all datasets. Specifically, an average AUC of the attack aiming at Item or NCF is 18% and 20% higher respectively. Compared to the visualization result of LFM, the dissimilarity of the shadow and target distributions of Item or NCF is smaller. Thus the attack model can easily deal with the data points which are similar to its training data. Besides, our attack performs better when the target dataset is the ml-1m dataset. This is because the user-item matrix of the ml-1m dataset is the densest among all three datasets, which enormously facilitates the item vectorization and attack model training.

Assumption II. To this end, we relax the assumption so that the adversary only has a shadow dataset in the same distribution as the target dataset. The experimental results are shown in Table 3, Table 4 and Table 5, where the attack results of the previous assumption are listed at the diagonals. Then we depict the visualization results of data points in the shadow and target distributions by t-SNE to show the relationship between members and non-members. In Figure 5, the red points are members and the blue points are non-members, which are all from the lf-2k dataset. We can see from the attack performances as well as the comparisons between the shadow and target distributions that:

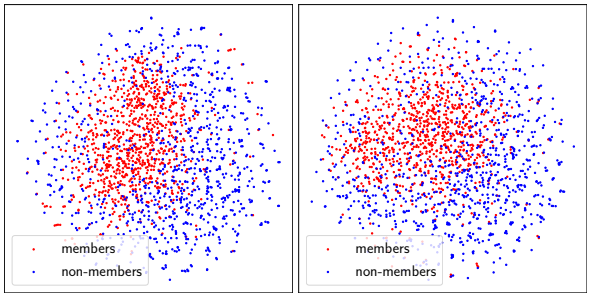
- When the adversary only gains the knowledge about the target dataset distribution, the attack performances drop as expected but are still strong. For instance, on the ADM dataset, when the target recommender uses Item but the shadow recommender uses LFM, the attack performance drops from an AUC of 0.926 to an AUC of 0.843. Decreases



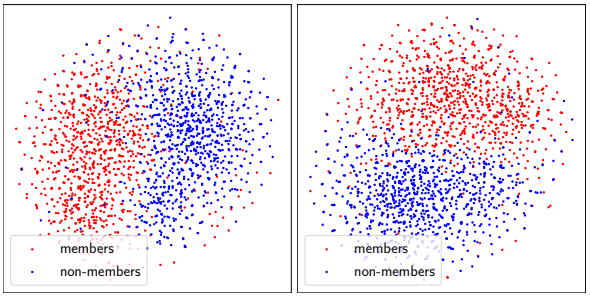
(a) ADM_NCF_shadow (left) and ADM_NCF_target (right)



(b) ADM_LFM_shadow (left) and ADM_LFM_target (right)

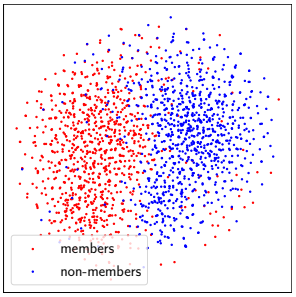


(c) lf-2k_Item_shadow (left) and lf-2k_Item_target (right)

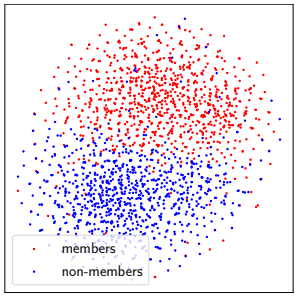


(d) lf-2k_LFM_shadow (left) and lf-2k_LFM_target (right)

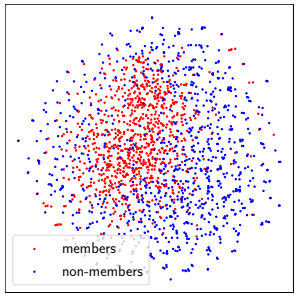
Figure 4: Visualization results by t-SNE, where red points denote members and blue points represent non-members. For the ADM dataset, visualization results, (a) when the shadow and target recommenders are implemented by NCF, and (b) when LFM is used as the shadow and target recommenders, are demonstrated. For the lf-2k dataset, visualization results, (c) when the shadow and target recommenders are implemented by Item, and (d) when LFM is adopted as the shadow and target recommenders, are shown.



(a) lf-2k_LFM_shadow



(b) lf-2k_LFM_target



(c) lf-2k_Item_shadow

Figure 5: Visualization results by t-SNE, where red points are members and blue points are non-members. For the lf-2k dataset, visualization results, (a) when the shadow recommender is implemented by LFM, (b) when LFM is employed as the target recommender, and (c) when Item is used as the shadow recommender, are demonstrated.

also appear on the lf-2k dataset and the ml-1m dataset. That is to say, even with different recommendation methods, the attack model can still benefit from the similar distributions of the target and shadow datasets to conduct the memebership inference accurately.

- An interesting finding is that, the attack on the ml-1m dataset achieves the best overall performance (i.e., 0.873 in terms of average AUC), and the attack performance on the ADM dataset is the worst (i.e., 0.747 in terms of average AUC). This is because the user-item matrix built from the ml-1m dataset is the densest while the matrix from the ADM dataset is the sparsest. Intuitively, the attack model can learn more information from a denser user-item matrix, leading to a better attack performance.

In addition, we do acknowledge that, in some cases, the attack performances are not ideal. For instance, the attack model against LFM achieves a poor performance (see Table 3). Comparing to the other two recommendation algorithms, LFM has higher model complexity, which makes it harder for the adversary to build a similar shadow model.

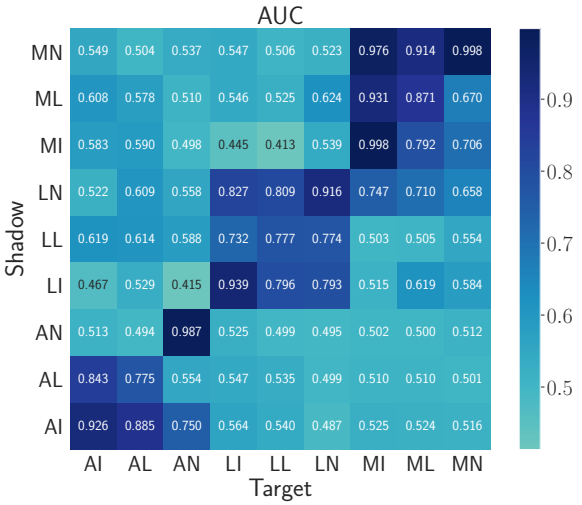
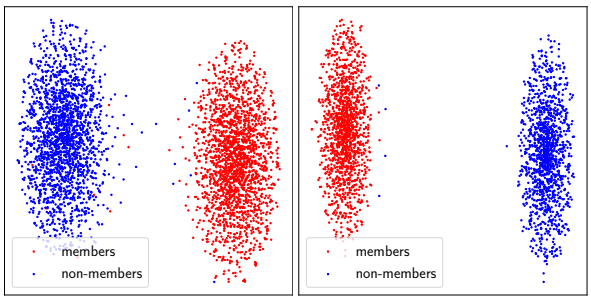
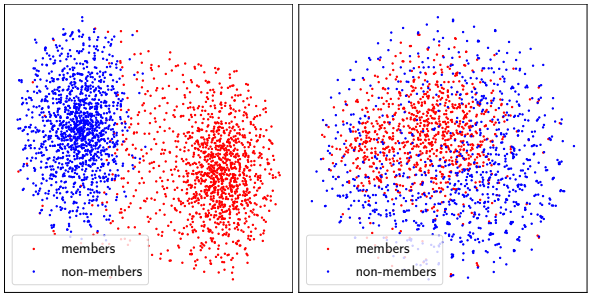


Figure 6: The attack performances under the assumption III. The x-axis indicates the target recommender's datasets (the first letter, i.e., "A," "L" and "M") and algorithms (the second letter, i.e., "I," "L" and "N"), and similarly the y-axis represents the shadow recommender's datasets and algorithms.



(a) ml-1m_LFM_shadow (left) and ADM_Item_target (right)



(b) ADM_LFM_shadow (left) and lf-2k_Item_target (right)

Figure 7: The visualization results of MLAI v.s. ALLI

Assumption III. Finally, we further conduct evaluations when the adversary neither has a shadow dataset in the same distribution as the target dataset nor knows the target algorithm. All experimental results are shown in Figure 6. Note that, the attack results of the assumption I are listed at the back-diagonal and the attack performances of the assumption II are shown in the three 3×3 block back-diagonal matrices. Analysing the results, we draw conclusions that:

- Even under the minimum assumption, our attack can still achieve strong performances in most cases. For instance, when the target recommender is established by LFM on the ml-1m dataset and the adversary uses NCF to build a shadow recommender on the lf-2k dataset, our attack achieves an AUC of 0.710.
- In some cases, when the adversary knows less information about the target recommender, the attack even achieves better performances. For instance, when the adversary builds a shadow recommender by NCF on the lf-2k dataset to mimic the target recommender which uses Item on the ml-1m dataset, our attack achieves an AUC of 0.747. Meanwhile, with the knowledge that the target dataset is the ml-1m dataset, the adversary uses LFM to establish a shadow recommender when the target recommender uses NCF, our attack only achieves an AUC of 0.670. To explain this, we adopt the t-SNE algorithm to visualize user feature vectors for the "MLAI" and "ALLI" attacks. The visualization results in Figure 7 show that the distributions of feature vectors generated by the shadow model "ML" and target model "AL" are more similar than the distributions generated by "AI" and "LI". Therefore, the "MLAI" attack performs better than the "ALLI" attack.

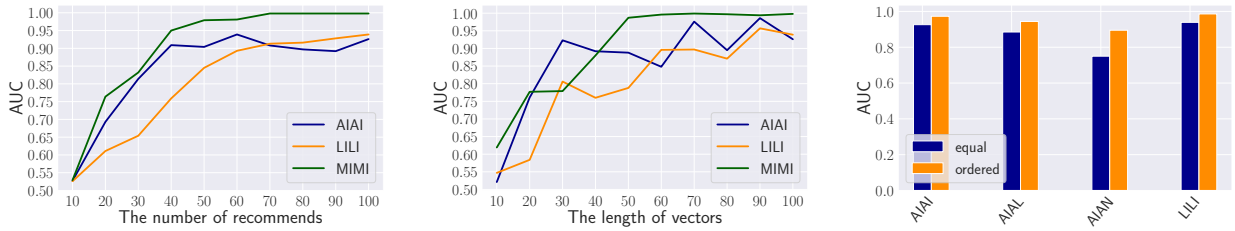
In summary, our attack can effectively conduct a membership inference against recommender systems, even with the limited knowledge.

3.4 Hyperparameters

In this section, we analyse the influences of hyperparameters, including the number of recommendations k, the length of vectors l and the weights of recommendations. Figure 8 shows the experimental results.

The Number of Recommendations k. We evaluate our experiments with different values of k from 10 to 100, in order to explore the influence of k on the attack. Figure 8a shows the attack performance against the number of recommendations. When the number of recommendations is less than 50, the attack performance improves with the increase of k. Then the performance maintains stable when k goes beyond 50. These results show that the attack model gains more information when the number of recommendations increases. However, the attack model cannot gain more information infinitely when the number of recommendations is large enough.

The Length of Vectors l. We evaluate our experiments with different values of l from 10 to 100, in order to explore the influence of l on the attack. Figure 8b shows the attack performance against the length of vectors. Similar to Figure 8a, when the length of vectors is less than 50, the attack performance improves with the increase of l. Then, in general, no obvious improvement of the performance is observed when l goes beyond 50. These results show that the representation power of the attack model becomes stronger, as a larger length of vectors can provide more dimensional perspectives. However, the attack model cannot improve its representation power infinitely when the length of vectors is large enough.

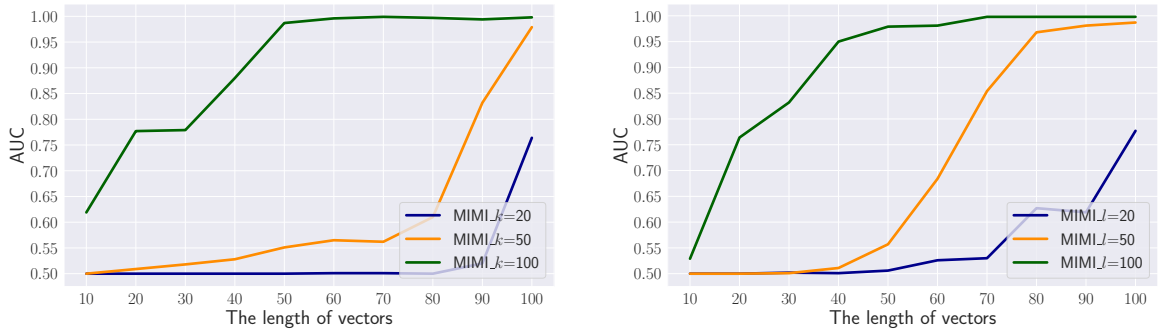


(a) The attack performances against the number of recommendations k.

(b) The attack performances against the length of vectors l.

(c) The attack performances against the weights of recommendations.

Figure 8: The attack performances of analysing the influences of hyperparameters.



(a) The attack performances with different l, when $k = 20$, $k = 50$ and $k = 100$.

(b) The attack performances with different k, when $l = 20$, $l = 50$ and $l = 100$.

Figure 9: The attack performances to investigate "which is more important, k or l?"

The Weights of Recommendations. We evaluate our experiments with different designs for the weights of recommendations. In the real world, the items recommended to a user are provided in the form of an ordered sequence. And, compared to the items at the back of the sequence, the ones in front of the sequence are more likely to be preferred by the user. Thus we evaluate two methods of assigning weights to items at different positions in the sequence. One is that all items are assigned the same weight of $\frac{1}{k}$. And the other is to assign a weight of $\frac{k-i+1}{\sum_{n=1}^{k} n}$ to the i^{th} item in the sequence. Same as mentioned above, we denote k as the number of recommendations. As shown in Figure 8c, we find that considering the order of recommendations can obviously promote attack performances.

3.5 Extensive Analysis

In this section, we study five interesting questions and give further results in order to comprehensively investigate our attack method.

Which Is More Important, the Length of Feature Vector l or the Number of Recommendations k? In addition to the analyses about hyperparameters in Section 3.4, we also investigate "which is more important, k and l?". Two more experiments are conducted on the "MIMI" attack, with different l when k is set to 20, 50 and 100 (see Figure 9a), and with different k when l is set to 20, 50 and 100 (see Figure 9b). As the results show, both the number of recommendations (k) and the length of vectors (l) influence attack performances substantially. Specifically, when k reduces from 100 to 20, the AUC score drops from 0.998 to 0.764. Similarly, as l reduces from 100 to 20, the AUC score descends from 0.998 to 0.817.

What Is the Impact of the Dataset Size? Considering the size of the training dataset imposes huge impacts on machine learning models, we conduct evaluations regarding the size of the shadow dataset. Specifically, the size of the shadow dataset is reduced to 90%, 80%, and 70% of the original size. Note that, the ratio of members to non-members keep unchanged for the dataset balance. For the "LLLL" attack, the AUC scores of the attack performances are decreased to 0.633, 0.714, and 0.746, respectively, when the size of the shadow dataset is 70%, 80%, and 90% of the original size. Comparing to the original AUC of 0.777, we can conclude that a larger shadow dataset usually leads to a better-trained attack model.

Why Use an MLP as the Attack Model? To demonstrate the effectiveness of our attack model, we evaluate the attacks utilizing K-Means to distinguish non-members from members on the lf-2k and ml-1m datasets. The results are shown in Table 6 and Table 7, where Item, LFM, and NCF in the first column are shadow algorithms for our attack, and K-Means is used to cluster non-members and members. Since there are only two classes, i.e., members and non-members, the number of classes K for K-Means is set to 2. From the results, we can conclude that our attack outperforms K-Means largely, indicating the validity of our attack model. For instance, when the K-Means algorithm infers the membership status from

Table 6: The AUC of the K-Means algorithm on the lf-2k dataset.

	Target Algorithm		
	Item	LFM	NCF
Item	**0.939**	0.796	0.793
LFM	0.732	0.777	0.774
NCF	0.827	**0.809**	**0.916**
K-Means	*0.649*	*0.717*	*0.730*

Table 7: The AUC of the K-Means algorithm on the ml-1m dataset.

	Target Algorithm		
	Item	LFM	NCF
Item	**0.998**	0.792	0.706
LFM	0.931	0.871	0.670
NCF	0.976	**0.914**	**0.998**
K-Means	*0.805*	*0.720*	*0.719*

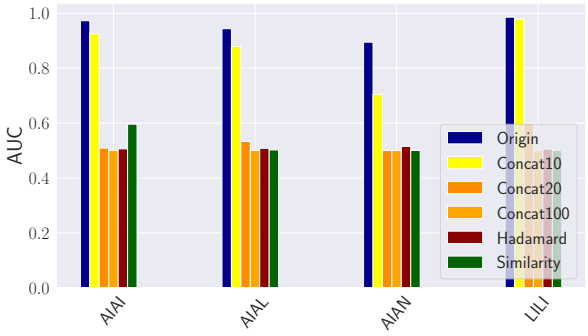


Figure 10: The attack performances with different user feature generation methods

the target recommender "LI", the performance is much worse than our attack (0.649 v.s. 0.939).

Which Is the Best User Feature Generation Method? To further verify the effectiveness of our attack, we adopt different aggregation methods to generate user feature vectors. Besides the method used in our attack, **Origin**, we evaluate 5 user feature generation methods.

- **Concat10** concatenates feature vectors of the first 10 interactions and the first 10 recommendations for each user.
- **Concat20** concatenates feature vectors of the first 20 interactions and the first 20 recommendations for each user.
- **Concat100** concatenates feature vectors of the first 20 interactions and all the recommendations (i.e., $k = 100$) for each user.[2]

[2]As the number of interactions for different users are different, using all interactions for concatenation will result in different lengths of user feature vectors. Therefore, we only take the first 20 interactions into consideration since the least number of interactions in datasets is 20.

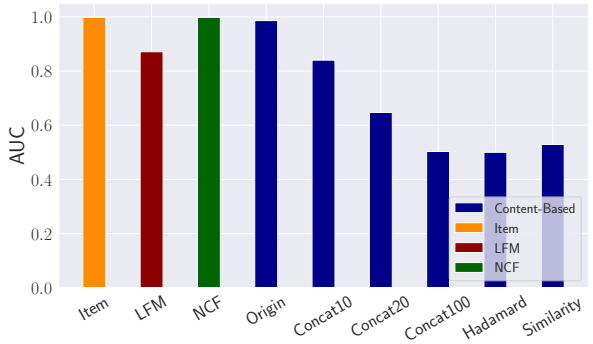


Figure 11: The attack performances for the content-based recommender system on the ml-1m dataset.

- **Hadamard** respectively conducts Hadamard products on feature vectors of all the interactions and recommendations to obtain two vectors for each user. Afterwards, following the similar steps in Section 2.5, we use the difference of these two vectors as the user feature vector.
- **Similarity** first conducts dot products between feature vectors of each recommendation and all interactions, then concatenates the average of dot product results for each recommendation into the user feature vector.

The results are shown in Figure 10. We find that our method outperforms all the other aggregation methods. For instance, on the settings of AIAI, our method outperforms Concat10 and Similarity by 5% and 63%. These results demonstrate the effectiveness of our user feature generation method in the attack process.

Is It Possible to Generalize Our Attacks to Content-Based Recommender Systems? To further verify the effectiveness of our attack, we conduct evaluations on the membership inference attacks against a content-based recommender system. Different from the recommendation algorithms used in the previous attacks, a content-based recommender system aims to distinguish users' likes from dislikes based on their *metadata* (such as description of items and profiles of users) [51]. The evaluation is conducted on the ml-1m dataset with information about items and users, under the assumption I, i.e., the target recommender's algorithm and dataset distribution are available.

The results are depicted in Figure 11. We conclude that our attack achieves a strong performance against a content-based recommendation algorithm (i.e., 0.986 in terms of AUC), indicating a high generalization ability of our attack model. Furthermore, we evaluate this attack using different feature generation methods mentioned above. Results show that our aggregation method also outperforms other baselines on a content-based recommendation system.

3.6 Summary

The experimental results show that our attack can conduct an effective membership inference against recommender systems. When the adversary knows the algorithm and dataset distribution of the

target recommender, the attack achieves the strongest performance. Later in the experiments, we gradually relax the assumptions and show that our attack can still effectively conduct the membership inference, demonstrating that our attack has a good generalization ability.

Furthermore, we explore the influences of hyperparameters. With the increase of the number of recommendations k and the length of vectors l, the attack performance improves or maintains stable, however, the cost continuously increases. In that case, we are able to find a balance, where the attack performance is strong and the cost is affordable. And the exploration of the weights of recommendations shows that the more information is available, the more powerful the attack is.

4 DEFENSE

The above experiments show the effectiveness of our attack. Meanwhile, to defend the membership inference against recommender systems, we also propose a countermeasure, named Popularity Randomization, to mitigate the attack risk. In the original setting in Section 2.5, non-members are provided with the most popular items. As a result, feature vectors of non-members are extremely similar and easily distinguished from members. To address this problem, we increase the randomness of non-members' recommendations. Specifically, we first select candidates from the most popular items. Then, a random selection is conducted on the candidates, i.e., we randomly pick 10% of candidates as recommendations for non-members. The detailed methodology is demonstrated in Appendix A.1.

To evaluate the effectiveness of the defense mechanism, we conduct experiments under the assumption that the dataset distribution and algorithm of the target recommender are available. Figure 12 shows the attack performances before and after deploying the defense mechanism. The blue bar denotes the attack performances with the original setting, i.e., the popularity recommendation algorithm. And the orange bar represents the attack performances with the defense mechanism, i.e., Popularity Randomization. From the results, we conclude that Popularity Randomization considerably decreases the performance of our attack. Specifically, the defense mechanism decreases the AUC scores of the attack model by more than 12%, 33% and 41% respectively when the target recommender uses Item, LFM or NCF. With the defense strategy, attacking the target recommender using LFM achieves the lowest AUC score on all three datasets. When the target recommender using LFM, our attack with the defense mechanism only achieves 0.513, 0.501, and 0.500 AUC scores on the ADM dataset, the lf-2k dataset and the ml-1m dataset, respectively (detailed explanations are demonstrated in Appendix A.2). Besides, as Figure 12 shows, the attack performances against NCF decrease most hugely. For instance, the AUC score of the attack against NCF on the ADM dataset drops from 0.987 to 0.576. In contrast, the attack with Popularity Randomization against Item can still achieve strong performances. For instance, on the ADM dataset, the attack with Popularity Randomization can attain 0.812 in terms of AUC when the target algorithm is Item. Compared to the attack with the original setting, the defense mechanism only achieves a 12% drop in the attack performance. Item is the simplest one among the three recommendation methods, which makes it easier for the adversary to build a similar shadow recommender with the target recommender. This leads to a stronger attack but more ineffective defense. In contrast, the other two recommender systems have more complex model structures, leading to substantial decreases in attack performances with the defense strategy.

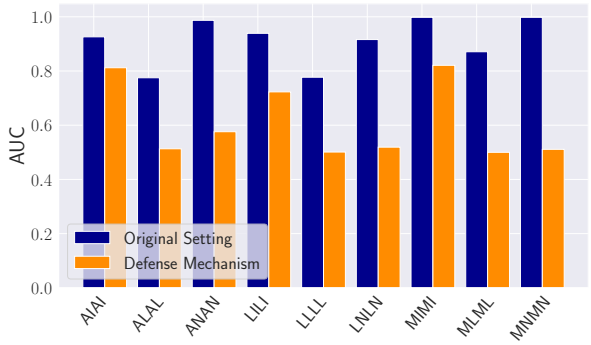


Figure 12: Comparisons of attack performances before and after deploying the defense mechanism.

In addition, visualization results and impacts on recommendation performances are comprehensively analyzed in Appendix A.2 and Appendix A.3, respectively.

5 RELATED WORK

Membership Inference. The goal of membership inference is to infer whether a target data sample is used to train a machine learning model [6, 22, 26, 28, 30, 39, 42, 43, 52]. Shokri et al. [43] propose the first membership inference attack in this domain. The authors have made several key assumptions for the adversary, such as multiple shadow models and a shadow dataset which comes from the same distribution as the target model's training datasset. Salem et al. [39] gradually relax these assumptions and broaden the scenarios of membership inference attacks. Later, Nasr et al. [31] conduct a comprehensive membership privacy assessment in both centralized and federated learning setting. In particular, they propose the first membership inference when the adversary has white-box to the target model. Other research has shown that membership inference is effective under other machine learning settings, such as generative models [14], federated learning [8, 29], and natural language models [45]. Besides, a plethora of other attacks have been proposed against machine learning models [3, 5, 7, 12, 20, 21, 33, 48, 49].

Item-Based Recommendation Algorithms. Item-based recommendation techniques have been applied in various scenarios [11, 23, 40]. Sarwar et al. [40] explore item-based collaborative filtering (CF) techniques which enhance the scalability and quality of the CF-based algorithms. Besides, Deshpande and Karypis [11, 23] present item-based top-N recommendation algorithms to promote the efficiency and performance.

Latent Factor Models. LFM aims to find some latent factors and is commonly implemented by Matrix Factorization (MF) [24, 25, 36, 38]. Polat et al. [36] combine SVD-based Collaborative Filtering with privacy to achieve accurate predictions while preserving

privacy. Later, Salakhutdinov et al. [38] propose the Probabilistic Matrix Factorization which scales linearly with the number of observations and performs well on very sparse and imbalanced datasets. Koren [24] presents an integrated model that combines the neighborhood and LFM, which optimizes a global cost function and integrates implicit feedback into the model. Furthermore, Koren [25] presents a methodology for modeling time drifting user preference in the context of recommender systems.

Neural Collaborative Filtering. With the advancement of deep learning techniques, recommendation algorithms with neural networks has been in blossom [2, 9, 16, 46, 47]. He et al. [16] propose the first framework for collaborative filtering based on neural networks to model latent features of users and items. They show that MF can be interpreted as a specialization of NCF and utilize a MLP to endow NCF modelling with a high level of non-linearities. Later, Bai et al. [2] present a model which integrates neighborhood information into NCF, namely Neighborhood-based Neural Collaborative Filtering. Another recent work is that Chen et al. [9] design a Joint Neural Collaborative Filtering model which enables deep feature learning and deep user-item interaction modeling to be tightly coupled and jointly optimized in a single neural network.

6 DISCUSSION

In the previous evaluations, our attack shows its effectiveness as well as strong generalization ability. Moreover, the proposed defense mechanism, Popularity Randomization, can also mitigate the attack performances considerably. Furthermore, to obtain a comprehensive understanding of membership inference attacks, in this section, we focus on three important factors that largely influence attack performances: the choice of datasets, the selection of recommendation algorithms, and distributions of generated user features. The detailed explanations are demonstrated as follows:

The Choice of Datasets. The dataset with a denser user-item matrix leads to better attack performances. The richer information in a denser user-item matrix considerably facilitates the process of the item vectorization and attack model training. From the results and analyses under the assumption II in Section 3.3, we can see that the attack on the ml-1m dataset achieves the best overall performance (i.e., 0.873 in terms of average AUC) as the user-item matrix built from this dataset is the densest.

The Selection of Recommendation Algorithms. It is easier for the adversary to attack against a recommender system with a simpler model structure. As the results (see Table 3) under the assumption II show, the attack against LFM achieves poor performances. Comparing to the other two recommendation algorithms, LFM has higher model complexity, which makes it harder to attack. Meanwhile, defending a simpler recommender system is more difficult. As the evaluations of the defense (see Figure 12) in Section 4 show, attacks against Item perform strongly even with the defense mechanism. This is because Item has the simplest structure among the three recommendation algorithms. In short, a recommender system established by a simple algorithm structure is usually more vulnerable to membership inference attacks.

Distributions of Generated User Features. The combination of the dataset and recommender algorithm also matters. Higher attack performances can be obtained, when the distribution of user feature vectors generated by the shadow recommender system is more similar to the distribution generated by the target recommender system. Trained with samples from a similar distribution, attack models are able to conduct an accurate inference. Specifically, in Figure 6, the attack of "MLAI" achieves a better performance than the one of "ALLI" (0.608 v.s. 0.547). And we see from the visualization results in Figure 7 that the above advantage comes from the smaller difference of feature distributions between the shadow recommender "ML" and the target recommender "AI" than the one between "AL" and "LI". In summary, training data with distributions similar to the target data can boost the attack performances.

7 CONCLUSION

Recommender systems have achieved tremendous success in real-world applications. However, data used by recommender systems is highly sensitive. In that case, successfully inferring a user's membership status from a target recommender may lead to severe privacy consequences.

In this paper, to investigate the privacy problem in recommender systems, we design various attack strategies of membership inference. To the best of our knowledge, ours is the first work on the membership inference attacks against recommender systems. Comparing to membership inference attacks on data sample-level classifiers, for recommender systems, our work focuses on the user-level membership status, which cannot be directly obtained from the system outputs. To address these challenges, we propose a novel membership inference attack scheme, the core of which is to obtain user-level feature vectors based on the interactions between users and the target recommender, and input these feature vectors into attack models. Extensive experiment results show the effectiveness and generalization ability of our attack. To remedy the situation, we further propose a defense mechanism, namely Popularity Randomization. Our empirical evaluations demonstrate that Popularity Randomization can largely mitigate the privacy risks.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful feedback. This work was supported by the Natural Science Foundation of China (61902219, 61972234, 62072279, 62102234), the Helmholtz Association within the project "Trustworthy Federated Data Analytics" (TFDA) (funding number ZT-I-OO1 4), the National Key R&D Program of China with grant No. 2020YFB1406704, the Key Scientific and Technological Innovation Program of Shandong Province (2019JZZY010129), Shandong University multidisciplinary research and innovation team of young scholars (No. 2020QNQT017), and the Tencent WeChat Rhino-Bird Focused Research Program (JR-WXG2021411). All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Michael Backes, Mathias Humbert, Jun Pang, and Yang Zhang. walk2friends: Inferring Social Links from Mobility Profiles. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1943–1957. ACM, 2017.

[2] Ting Bai, Ji-Rong Wen, Jun Zhang, and Wayne Xin Zhao. A Neural Collaborative Filtering Model with Interaction-based Neighborhood. In *ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1979–1982. ACM, 2017.

[3] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion Attacks against Machine Learning at Test Time. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 387–402. Springer, 2013.

[4] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. Second Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec2011). In *ACM Conference on Recommender Systems (RecSys)*, pages 387–388. ACM, 2011.

[5] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *USENIX Security Symposium (USENIX Security)*, pages 267–284. USENIX, 2019.

[6] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting Training Data from Large Language Models. *CoRR abs/2012.07805*, 2020.

[7] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *IEEE Symposium on Security and Privacy (S&P)*, pages 39–57. IEEE, 2017.

[8] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. GAN-Leaks: A Taxonomy of Membership Inference Attacks against Generative Models. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 343–362. ACM, 2020.

[9] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten de Rijke. Joint Neural Collaborative Filtering for Recommender Systems. *ACM Transactions on Information Systems*, 2019.

[10] Christopher A. Choquette Choo, Florian Tramèr, Nicholas Carlini, and Nicolas Papernot. Label-Only Membership Inference Attacks. *CoRR abs/2007.14321*, 2020.

[11] Mukund Deshpande and George Karypis. Item-Based Top-N Recommendation Algorithms. *ACM Transactions on Information Systems*, 2004.

[12] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Grag. Badnets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *CoRR abs/1708.06733*, 2017.

[13] F. Maxwell Harper and Joseph A Konstan. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems*, 2015.

[14] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. LOGAN: Evaluating Privacy Leakage of Generative Models Using Generative Adversarial Networks. *Symposium on Privacy Enhancing Technologies Symposium*, 2019.

[15] Ruining He and Julian McAuley. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *The Web Conference (WWW)*, pages 507–517. ACM, 2016.

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural Collaborative Filtering. In *International Conference on World Wide Web (WWW)*, pages 173–182. ACM, 2017.

[17] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 549–558. ACM, 2016.

[18] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. Stealing Links from Graph Neural Networks. In *USENIX Security Symposium (USENIX Security)*. USENIX, 2021.

[19] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining Collaborative Filtering Recommendations. In *ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 241–250. ACM, 2000.

[20] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High Accuracy and High Fidelity Extraction of Neural Networks. In *USENIX Security Symposium (USENIX Security)*, pages 1345–1362. USENIX, 2020.

[21] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 19–35. IEEE, 2018.

[22] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. MemGuard: Defending against Black-Box Membership Inference Attacks via Adversarial Examples. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 259–274. ACM, 2019.

[23] George Karypis. Evaluation of Item-Based Top-N Recommendation Algorithms. In *ACM International Conference on Information and Knowledge Management (CIKM)*, pages 247–254. ACM, 2001.

[24] Yehuda Koren. Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 426–434. ACM, 2008.

[25] Yehuda Koren. Collaborative Filtering with Temporal Dynamics. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 447–456. ACM, 2009.

[26] Klas Leino and Matt Fredrikson. Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference. In *USENIX Security Symposium (USENIX Security)*, pages 1605–1622. USENIX, 2020.

[27] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. How to Prove Your Model Belongs to You: A Blind-Watermark based Framework to Protect Intellectual Property of DNN. In *Annual Computer Security Applications Conference (ACSAC)*, pages 126–137. ACM, 2019.

[28] Zheng Li and Yang Zhang. Membership Leakage in Label-Only Exposures. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2021.

[29] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting Unintended Feature Leakage in Collaborative Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 497–512. IEEE, 2019.

[30] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine Learning with Membership Privacy using Adversarial Regularization. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 634–646. ACM, 2018.

[31] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1021–1035. IEEE, 2019.

[32] Milad Nasr, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, and Nicholas Carlini. Adversary Instantiation: Lower Bounds for Differentially Private Machine Learning. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2021.

[33] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. SoK: Towards the Science of Security and Privacy in Machine Learning. In *IEEE European Symposium on Security and Privacy (Euro S&P)*, pages 399–414. IEEE, 2018.

[34] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable Private Learning with PATE. In *International Conference on Learning Representations (ICLR)*, 2018.

[35] Michael J. Pazzani and Daniel Billsus. Content-Based Recommendation Systems. In *The Adaptive Web, Methods and Strategies of Web Personalization*, pages 325–341. Springer, 2007.

[36] Huseyin Polat and Wenliang Du. SVD-based Collaborative Filtering with Privacy. In *ACM Symposium on Applied Computing (SAC)*, pages 791–795. ACM, 2005.

[37] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs Black-box: Bayes Optimal Strategies for Membership Inference. In *International Conference on Machine Learning (ICML)*, pages 5558–5567. PMLR, 2019.

[38] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic Matrix Factorization. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1257–1264. NIPS, 2007.

[39] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.

[40] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-Based Collaborative Filtering Recommendation Algorithms. In *International Conference on World Wide Web (WWW)*, pages 285–295. ACM, 2001.

[41] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative Filtering Recommender Systems. In *The Adaptive Web, Methods and Strategies of Web Personalization*, pages 291–324. Springer, 2007.

[42] Virat Shejwalkar and Amir Houmansadr. Membership Privacy for Machine Learning Models Through Knowledge Transfer. In *AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 2021.

[43] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *IEEE Symposium on Security and Privacy (S&P)*, pages 3–18. IEEE, 2017.

[44] Reza Shokri, Georgios Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. Quantifying Location Privacy. In *IEEE Symposium on Security and Privacy (S&P)*, pages 247–262. IEEE, 2011.

[45] Congzheng Song and Vitaly Shmatikov. Auditing Data Provenance in Text-Generation Models. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 196–206. ACM, 2019.

[46] Peijie Sun, Le Wu, and Meng Wang. Attentive Recurrent Social Recommendation. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 185–194. ACM, 2018.

[47] Peijie Sun, Le Wu, Kun Zhang, Yanjie Fu, Richang Hong, and Meng Wang. Dual Learning for Explainable Recommendation: Towards Unifying User Preference Prediction and Review Generation. In *The Web Conference (WWW)*, pages 837–847. ACM, 2020.

[48] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble Adversarial Training: Attacks and Defenses. In *International Conference on Learning Representations (ICLR)*, 2017.

[49] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *USENIX Security Symposium (USENIX Security)*, pages 601–618. USENIX, 2016.

[50] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 2008.

[51] Bogdan Walek and Vladimir Fojtik. A Hybrid Recommender System for Recommending Relevant Movies Using An Expert System. *Expert Systems with Applications*, 2020.
[52] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In *IEEE Computer Security Foundations Symposium (CSF)*, pages 268–282. IEEE, 2018.

A DEFENSE

In this section, we demonstrate the defense methodology in detail (Appendix A.1), discuss about visualization results (Appendix A.2), and analyze the impacts of Popularity Randomization on original recommendation performances (Appendix A.3).

A.1 Methodology

In the original setting in Section 2.5, non-members are provided with the most popular items. However, the most popular items for different users are the same, resulting in that feature vectors of non-members are extremely similar and easily distinguished. To address this problem, an intuitive way is to increase the randomness of non-members' recommendations. In the paper, we propose a defense mechanism named *Popularity Randomization*. Specifically, a larger number of the most popular items are first selected as candidates. Then, we conduct a random selection, i.e., we randomly pick 10% of candidates as recommendations for non-members. Note that all recommendations for non-members are still the most popular items. In that case, recommendations to different non-members are diverse and thus the similarities among non-members' feature vectors are decreased. The above defense strategy can be formulated as follows:

First, a recommender system $\mathcal{A}_{RS}$ sorts all items by popularity, i.e.,

$$\mathcal{A}_{RS} : \mathcal{D}_{original} \xrightarrow{Sort} \mathcal{S}_{sorted},$$

where $\mathcal{D}_{original}$ is the original dataset and $\xrightarrow{Sort}$ is to sort items by popularity. And $\mathcal{S}_{sorted}$ is an ordered sequence, including all items from $\mathcal{D}_{original}$. In $\mathcal{S}_{sorted}$, higher items are more popular.

Second, based on the number of recommendations and the preset ratio, the recommender system selects a number of the most popular items as candidates, which can be defined as the following functions:

$$\alpha_{PR} = \frac{N_{rec}}{N_{cand}},$$

$$\mathcal{S}_{cand} = \{x | x \in \mathcal{S}_{sorted}, Ind_x \leq N_{cand}\},$$

where α_{PR} is the ratio of recommendations to candidates. $\mathcal{S}_{cand}$ is a sequence containing all candidates selected by the recommender system from $\mathcal{S}_{sorted}$. N_{rec} and N_{cand} are the numbers of the recommendations and the candidates. And Ind_x is the index of x in $\mathcal{S}_{sorted}$.

Finally, non-members are randomly provided with items from candidates as recommendations, which can be defined as the following function:

$$f(\mathcal{S}_{cand}) = \mathcal{R}^{out} \quad (|\mathcal{R}^{out}| = N_{rec}),$$

where $f(\mathcal{S}_{cand})$ is to randomly select recommendations from $\mathcal{S}_{cand}$. And $|\mathcal{R}^{out}|$, which is equal to N_{rec}, is the number of recommendations to non-members.

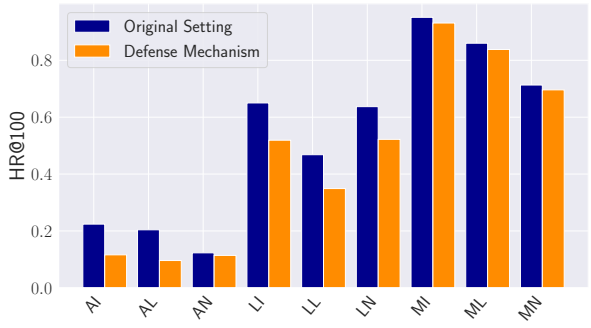


Figure 13: Comparisons of recommendation performances before against after deploying the defense mechanism.

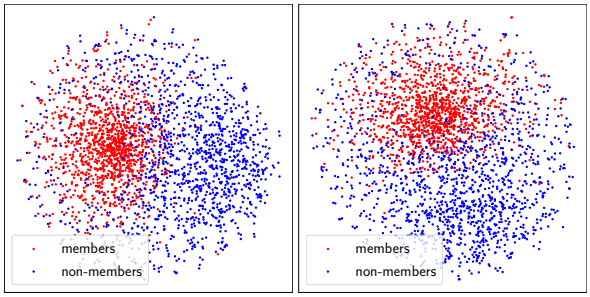
A.2 Visualization Results

In this section, we visualize user feature vectors by t-SNE to show the differences of the distributions between members and non-members in the shadow and target datasets. In Figure 14, the red points represent feature vectors of members and the blue points denote feature vectors of non-members. We conclude from the comparison between Figure 14a and Figure 14b that Popularity Randomization can decrease the differences between the feature vectors of members and non-members, and hinder the attack performance considerably. We find a similar phenomenon in Figure 14c and Figure 14d. These visualization results show the effectiveness of Popularity Randomization.

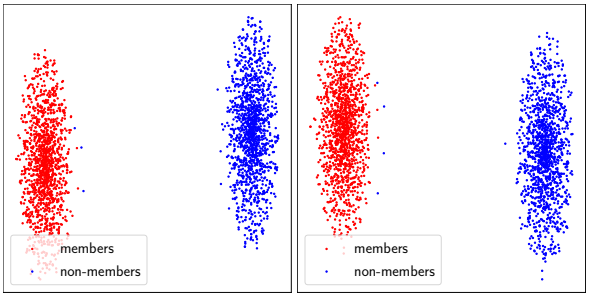
Moreover, as Figure 12 shows, with the defense mechanism, the attack against LFM can only reach comparable performances with Random Guess. In contrast, when the target algorithm is Item, the attack achieves strong performances. To explain this, as Figure 14a and Figure 14c shows, the areas of the red and blue points rarely overlap when using Item. However, there exist many red points in the area of the blue points when using LFM, leading to a poor attack performance.

A.3 Impacts on Recommendation Performances

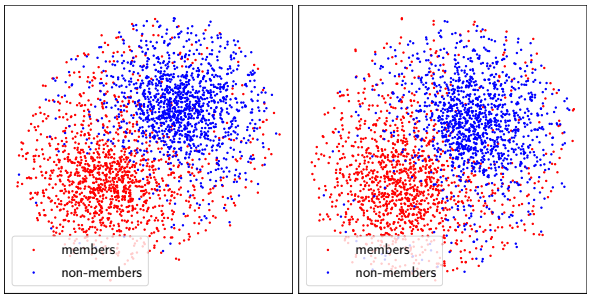
We next investigate the impacts on the performance of the target recommender with Popularity Randomization. In Figure 13, the results show that the performances of the recommender only slightly drops. For instance, when the target recommender uses Item on the ml-1m dataset, Popularity Randomization only achieves a 2% drop in the recommendation performance. Therefore, our proposed defense strategy can mitigate the attack risk effectively while preserving the original recommendation performances.



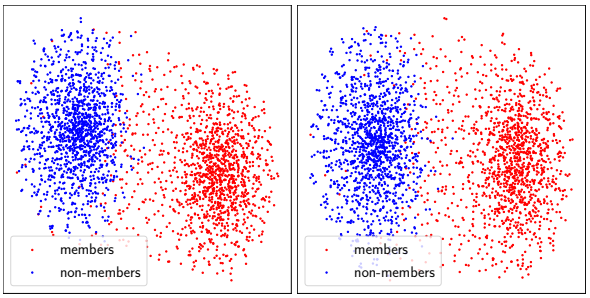
(a) ADM_Item_shadow (left) and ADM_Item_target (right) with the defense mechanism.



(b) ADM_Item_shadow (left) and ADM_Item_target (right) without the defense mechanism.



(c) ADM_LFM_shadow (left) and ADM_LFM_target (right) with the defense mechanism.



(d) ADM_LFM_shadow (left) and ADM_LFM_target (right) without the defense mechanism.

Figure 14: Data distributions by t-SNE, in which red points represent members and blue points denote non-member. (a) Data points in the shadow and target recommenders using Item on the ADM dataset with the defense mechanism. (b) Data points in the shadow and target recommenders using Item on the ADM dataset without the defense mechanism. (c) Data points in the shadow and target recommenders using LFM on the ADM dataset with the defense mechanism. (d) Data points in the shadow and target recommenders using LFM on the ADM dataset without the defense mechanism.