PORE: Provably Robust Recommender Systems against Data Poisoning Attacks

Jinyuan Jia, *The Pennsylvania State University;*
Yupei Liu, Yuepeng Hu, and Neil Zhenqiang Gong, *Duke University*

https://www.usenix.org/conference/usenixsecurity23/presentation/jia

This paper is included in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.

PORE: Provably Robust Recommender Systems against Data Poisoning Attacks

Jinyuan Jia*[1], Yupei Liu*[2], Yuepeng Hu[2], Neil Zhenqiang Gong[2]
[1]*The Pennsylvania State University,* [2]*Duke University*
[1]*jinyuan@psu.edu,* [2]*{yupei.liu, yuepeng.hu, neil.gong}@duke.edu*

Abstract

Data poisoning attacks spoof a recommender system to make arbitrary, attacker-desired recommendations via injecting fake users with carefully crafted rating scores into the recommender system. We envision a cat-and-mouse game for such data poisoning attacks and their defenses, i.e., new defenses are designed to defend against existing attacks and new attacks are designed to break them. To prevent such cat-and-mouse game, we propose PORE, the first framework to build *provably* robust recommender systems in this work. PORE can transform *any* existing recommender system to be provably robust against *any* untargeted data poisoning attacks, which aim to reduce the overall performance of a recommender system. Suppose PORE recommends top-N items to a user when there is no attack. We prove that PORE still recommends at least r of the N items to the user under any data poisoning attack, where r is a function of the number of fake users in the attack. Moreover, we design an efficient algorithm to compute r for each user. We empirically evaluate PORE on popular benchmark datasets.

1 Introduction

Many web service platforms (e.g., Amazon, YouTube, and TikTok) leverage recommender systems to engage users and improve user experience. Typically, a platform first collects a large amount of rating scores that users gave to items, which is known as a *rating-score matrix*. Then, the platform uses them to build a recommender system that models the complex relationships between user interests and item properties. Finally, the recommender system recommends top-N items to each user that match his/her interests.

However, due to its openness, i.e., anyone can register users and provide rating scores to items, recommender system is fundamentally not robust to *data poisoning attacks* [11,12,18,22,23,27,42]. Specifically, in a data poisoning attack, an attacker creates fake users in a recommender system and assigns carefully crafted rating scores to items. Different data poisoning attacks essentially use different methods to craft the fake users' rating scores. When a recommender system is built based on the *poisoned-rating-score matrix*, which includes the rating scores of both genuine and fake users, it recommends attacker-chosen, arbitrary top-N items to a user. As a result, the recommendation performance (e.g., Precision@N, Recall@N, and F1-Score@N) is substantially degraded. Data poisoning attacks pose severe challenges to the robustness/security of recommender systems.

Many defenses have been proposed to enhance the robustness of recommender systems against data poisoning attacks. In particular, one family of defenses [7,12,41,44,47] aim to detect fake users before building a recommender system. These methods rely on the assumption that the rating scores of fake users and genuine users have statistically different patterns, which they leverage to distinguish between fake and genuine users. Another family of defenses [8, 17, 25, 26, 30, 35, 43] aim to design new methods of training recommender systems such that they have good recommendation performance even if they are trained on a poisoned-rating-score matrix, e.g., using trim learning [17]. However, these defenses only achieve *empirical* robustness, leading to an endless cat-and-mouse game between attacks and defenses: a new empirical defense is proposed to mitigate existing attacks but can be broken by new attacks that adapt to the defense. For instance, fake users can adapt their rating scores such that they cannot be detected based on the rating scores' statistical patterns [11, 12, 18]. As a result, a recommender system's performance is still substantially degraded by strong, adaptive attacks.

Our work: In this work, we aim to end such cat-and-mouse game via proposing PORE, the first framework to build *provably* robust recommender systems against *any* untargeted data poisoning attacks. Suppose, under no attacks, a recommender system algorithm trains a recommender system on a clean rating-score matrix, which recommends a set of top-N items (denoted as Γ_u) to a user u. Under attacks, the recommender system algorithm trains a recommender system on a poisoned-rating-score matrix, which recommends a set of top-N items

*Equal contribution.

(denoted as Γ'_u) to the user. We say the recommender system algorithm is (e,r)-provably robust for the user u if the intersection between Γ_u and Γ'_u includes at least r items when there are at most e fake users, no matter how an attacker crafts the fake users' rating scores. In other words, an (e,r)-provably robust recommender system guarantees that at least r of the recommended top-N items are unaffected by e fake users no matter what rating scores they use. We note that r depends on the number of fake users e and we call r *certified intersection size*. A provably robust recommender system can guarantee a lower bound of recommendation performance under *any* data poisoning attack, i.e., no matter how fake users craft their rating scores.

Suppose a *submatrix* consists of s rows of the rating-score matrix, i.e., a submatrix includes rating scores of s users. Intuitively, when the fraction of fake users is bounded, a randomly sampled submatrix is likely to not contain fake users and thus a recommender system built based on the submatrix is not affected by fake users. Based on this intuition, PORE uses *bagging* [5], a well-known ensemble method, to achieve provable robustness. In particular, PORE aggregates recommendations from multiple base recommender systems to recommend top-N items to each user. Specifically, we can use any recommender system algorithm (called *base algorithm*) to build a recommender system (called *base recommender system*) on a submatrix. Therefore, we could build $\binom{n}{s}$ base recommender systems since there are $\binom{n}{s}$ submatrices, where n is the total number of users. Each base recommender system makes recommendations to users. We denote by p_i the fraction of the $\binom{n}{s}$ base recommender systems that recommend item i to a user u. We call p_i *item probability*.[1] PORE recommends the top-N items with the largest item probabilities to user u.

Our major theoretical result is that we prove PORE is (e,r)-provably robust, no matter what base algorithm is used to train the base recommender systems. Moreover, for any given number of fake users e, we derive the certified intersection size r for each genuine user, which is the solution to an optimization problem. PORE relies on the item probabilities p_i's to make recommendations. Moreover, the optimization problem to calculate r also involves item probabilities. However, it is challenging to compute the exact item probabilities as it requires building $\binom{n}{s}$ base recommender systems. To address the challenge, we design an efficient algorithm to estimate the lower/upper bounds of the item probabilities via building $T \ll \binom{n}{s}$ base recommender systems, where the T base recommender systems can be built in parallel. PORE makes recommendations based on the estimated item probabilities in practice. Moreover, we use the estimated item probabilities to solve the optimization problem to obtain r for each user.

We empirically evaluate PORE on three benchmark datasets, i.e., MovieLens-100k, MovieLens-1M, and MovieLens-10M. Moreover, we consider two state-of-the-art base algorithms, i.e., Item-based Recommendation (IR) [3] and Bayesian Personalized Ranking (BPR) [29], to show the generality of PORE. We also generalize state-of-the-art provably robust defense [19] against data poisoning attacks for machine learning classifiers to recommender systems and compare PORE with it. We have three key observations from our experimental results. First, PORE substantially outperforms the defense generalized from classifiers. Second, when there are no data poisoning attacks, PORE has comparable recommendation performance (i.e., Precision@N, Recall@N, and F1-Score@N) with a standard recommender system built on the entire rating-score matrix. Third, under any data poisoning attacks, PORE can guarantee a lower bound of recommendation performance, while the standard recommender systems cannot.

Our key contributions are summarized as follows:

- We propose PORE, the first framework to build recommender systems that are provably robust against untargeted data poisoning attacks.
- We prove the robustness guarantees of PORE and derive its certified intersection size. Moreover, we design an algorithm to compute the certified intersection size.
- We perform extensive evaluation on popular benchmark datasets using two state-of-the-art base recommender system algorithms.

2 Background

2.1 Recommender Systems

Rating-score matrix: Suppose we have n users and m items which are denoted as $\mathcal{U} = \{u_1, u_2, \cdots, u_n\}$ and $I = \{i_1, i_2, \cdots, i_m\}$, respectively. We use $\boldsymbol{M}$ to denote the rating-score matrix which has n rows and m columns, where a row corresponds to a user and a column corresponds to an item. Essentially, the rating-score matrix $\boldsymbol{M}$ captures the users' interests towards different items. In particular, an entry $\boldsymbol{M}_{ui}$ represents the rating score that the user u gave to the item i. For instance, the rating score could be an integer in the range $[1,5]$, where 1 is the lowest rating score and denotes that the item does not attract the user's interest, and 5 is the largest rating score and denotes that the item attracts the user's interest substantially. We note that our method is applicable to any type of rating scores, e.g., binary, integer-valued, and continuous. $\boldsymbol{M}_{ui} = 0$ means that the user u has not rated the item i yet. For convenience, we denote by $\mathcal{R}$ the domain of a rating score including 0, i.e., $\boldsymbol{M}_{ui} \in \mathcal{R}$.

Top-N recommended items: A recommender system aims to help users discover new items that may arouse his/her interests. A recommender system algorithm takes the rating-score matrix $\boldsymbol{M}$ as input and recommends top-N items to each user that he/she has not rated yet but is potentially interested

[1] Item probability p_i also depends on user u, but we omit it for simplicity.

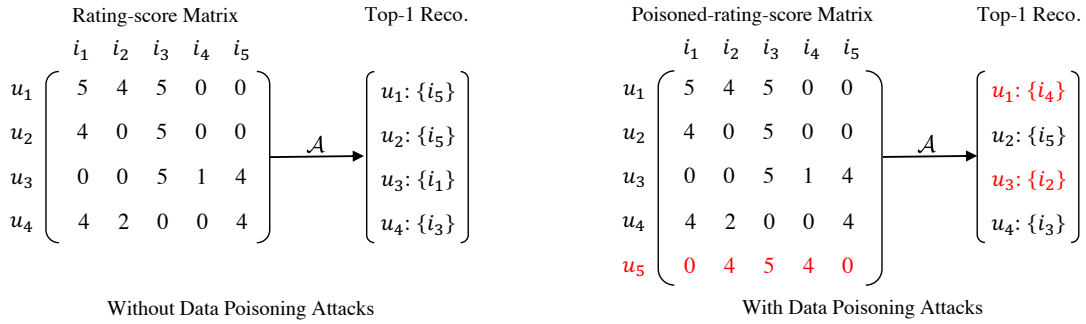

Figure 1: *Left*: a recommender system without data poisoning attacks. *Right*: an attacker manipulates the recommended items for users u_1 and u_3 by injecting a fake user (i.e., u_5) into the system.

in. For simplicity, we use $\mathcal{A}$ to denote a recommender system algorithm. Moreover, we use $\mathcal{A}(\boldsymbol{M}, u)$ to denote the set of top-N items recommended to user u when the recommender system is built by $\mathcal{A}$ on $\boldsymbol{M}$.

Recommender system algorithms: Many algorithms have been proposed to build recommender systems such as Item-based Recommendation (IR) [3, 24, 31], Bayesian Personalized Ranking (BPR) [29], Matrix Factorization [21], Neural Collaborative Filtering (NCF) [16], and LightGCN [15]. For instance, IR calculates the similarities between different items based on their rating scores, predicts users' missing rating scores using such similarities, and recommends a user the N items that the user has not rated yet but have the largest predicted rating scores. Due to its scalability, IR has been widely deployed in industry, e.g., Amazon [24]. According to recent benchmarks released by Microsoft [2], BPR achieves state-of-the-art performance, e.g., BPR even outperforms more complex algorithms such as NCF [16] and LightGCN [15].

2.2 Data Poisoning Attacks

Many studies [11, 12, 18, 22, 23, 27, 42] showed that recommender systems are not robust to data poisoning attacks (Section 6 discusses more details). In a data poisoning attack, an attacker creates fake users in a recommender system and assigns carefully crafted rating scores to them, such that the recommender system, which is built based on the rating scores of genuine and fake users, makes attacker-desired, arbitrary recommendations. For instance, a data poisoning attack could substantially degrade the performance of a recommender system; and a data poisoning attack could promote certain items (e.g., videos on YouTube and products on Amazon) via spoofing a recommender system to recommend them to many genuine users. Figure 1 illustrates data poisoning attacks.

We denote by $\boldsymbol{M}'$ the *poisoned-rating-score matrix*. A data poisoning attack aims to reduce the intersection between $\mathcal{A}(\boldsymbol{M}, u)$ and $\mathcal{A}(\boldsymbol{M}', u)$ via carefully designing the rating scores of the fake users. Different attacks essentially assign different rating scores to the fake users.

3 Problem Formulation

Threat model: We assume an attacker can inject fake users into a recommender system via registering and maintaining fake accounts [36]. We consider an attacker can inject at most e fake users into a recommender system, e.g., because of limited resources to register and maintain fake accounts. However, we assume each fake user can arbitrarily rate as many items as the attacker desires. Moreover, we assume the attacker has whitebox access to the recommender system, e.g., the attacker has access to the rating scores of all genuine users as well as the recommender system algorithm and its parameters. In other words, we consider strong attackers, who can perform any data poisoning attacks.

A poisoned-rating-score matrix $\boldsymbol{M}'$ extends the rating-score matrix $\boldsymbol{M}$ by at most e rows, which correspond to the rating scores of the at most e fake users. Different data poisoning attacks essentially select different rating scores for the fake users and result in different poisoned-rating-score matrix $\boldsymbol{M}'$. We use $\mathcal{L}(\boldsymbol{M}, e)$ to denote the set of all possible poisoned-rating-score matrices when the clean rating-score matrix is $\boldsymbol{M}$ and the number of fake users is at most e. $\mathcal{L}(\boldsymbol{M}, e)$ essentially denotes all possible data poisoning attacks with at most e fake users. Formally, we define $\mathcal{L}(\boldsymbol{M}, e)$ as follows:

$$\mathcal{L}(\boldsymbol{M}, e) = \{\boldsymbol{M}' | \boldsymbol{M}'_{ui} = \boldsymbol{M}_{ui} \text{ and } \boldsymbol{M}'_{vi} \in \mathcal{R}, \forall u \in \mathcal{U}, v \in \mathcal{V}, i \in I\}, \quad (1)$$

where $\mathcal{R}$ is the domain of a rating score, $\mathcal{U}$ is the set of genuine users, $\mathcal{V}$ is the set of at most e fake users (i.e., $|\mathcal{V}| \leq e$), and I is the set of items.

Provably robust recommender system algorithm: We say a recommender system algorithm is provably robust against data poisoning attacks if a certain number of its recommended

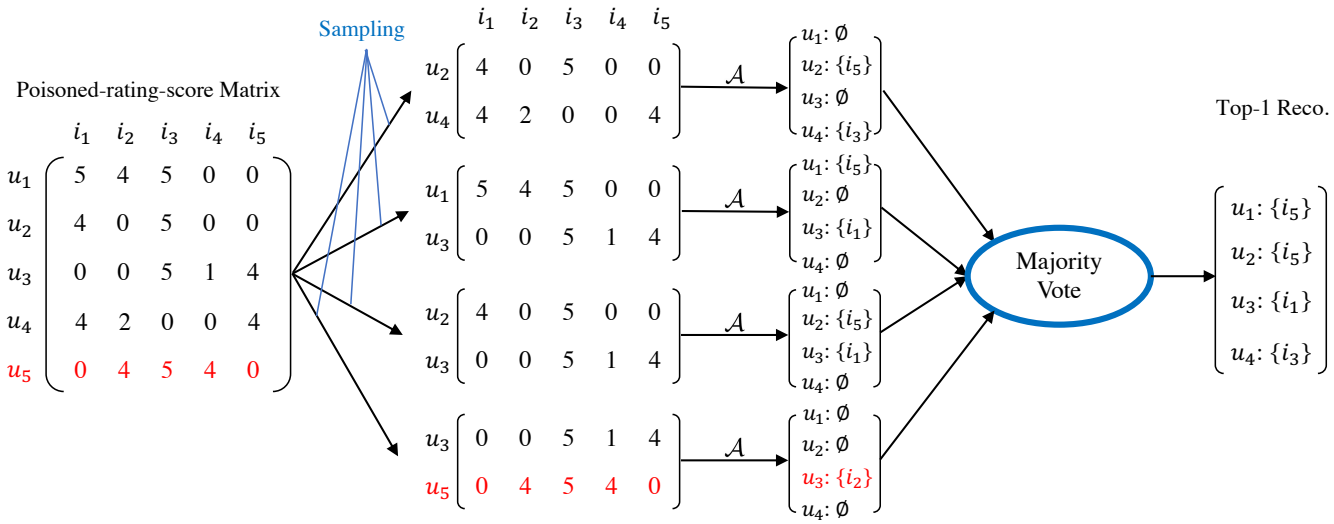

Figure 2: Robustness of our ensemble recommender system against data poisoning attacks.

top-N items for a user are provably unaffected by any data poisoning attacks. Specifically, given a set of items I_u, we say a recommender system algorithm $\mathcal{A}$ is provably robust for a user u if it satisfies the following property:

$$\min_{\boldsymbol{M}' \in \mathcal{L}(\boldsymbol{M},e)} |I_u \cap \mathcal{A}(\boldsymbol{M}', u)| \geq r, \tag{2}$$

where $\mathcal{L}(\boldsymbol{M},e)$ is the set of all possible poisoned-rating-score matrices (i.e., all possible data poisoning attacks with at most e fake users), $|I_u \cap \mathcal{A}(\boldsymbol{M}', u)|$ is the size of the intersection between I_u and the top-N items recommended to u by $\mathcal{A}$ under attacks, and r is called *certified intersection size*. Note that r may depend on the user u and the number of fake users e, but we omit its explicit dependency on u and e for simplicity.

When I_u is the set of top-N items recommended to user u by $\mathcal{A}$ under no attacks, i.e., $I_u = \mathcal{A}(\boldsymbol{M}, u)$, our provable robustness means that at least r of the N items in $\mathcal{A}(\boldsymbol{M}, u)$ are still recommended to u under any attacks with at most e fake users. When I_u is the set of ground truth test items for u (i.e., the set of items that u is indeed interested in), our provable robustness means that at least r of the ground truth test items are recommended to u under attacks. As we will discuss more details in experiments, r in the latter case can be used to derive a lower bound of recommendation performance such as Precision@N, Recall@N, and F1-Score@N under any data poisoning attacks. We formally define a provably robust recommender system algorithm as follows:

Definition 1 ((e,r)-Provably Robust Recommender System)**.** *Suppose the number of fake users is at most e and a data poisoning attack can arbitrarily craft the rating scores for the fake users. We say a recommender system algorithm $\mathcal{A}$ is (e,r)-provably robust for a user u if its certified intersection size for u is at least r, i.e., if Equation (2) is satisfied.*

4 Our PORE

We first give an overview of our PORE, then define our ensemble recommender system and show it is (e,r)-provably robust, and finally describe our algorithm to compute the certified intersection size r for each user.

4.1 Overview

Our key intuition is that, when the number of fake users is bounded, a random subset of a small number of users is likely to only include genuine users. Therefore, a recommender system built using the rating scores of such a random subset of users is not affected by fake users. Based on the intuition, PORE builds multiple recommender systems using random subsets of users and takes majority vote among them to recommend items for users.

Specifically, we create multiple submatrices from a rating-score matrix, where each submatrix contains the rating scores of s different randomly selected users, i.e., s rows randomly selected from the rating-score matrix. Then, we use an arbitrary base algorithm to build a recommender system (called *base recommender system*) on each submatrix and use it to recommend items for users. Finally, we build an *ensemble recommender system*, which takes a majority vote among the base recommender systems as the final recommended items for each user. Figure 2 shows our ensemble recommender system and its robustness against data poisoning attacks.

Next, we first formally define our ensemble recommender system in PORE. Then, we show PORE is provably robust against data poisoning attacks. In particular, given an arbitrary set of items I_u for a user u, we prove that at least r of the top-N items recommended to u by PORE are guaranteed to be among I_u under any data poisoning attacks. Moreover, we

derive the certified intersection size r. Finally, we design an algorithm to compute the certified intersection sizes r for all users simultaneously.

4.2 Our Ensemble Recommender System

Item probability p_i: Given a rating-score matrix $\boldsymbol{M}$, we randomly sample a submatrix with s rows of $\boldsymbol{M}$, i.e., the submatrix consists of the rating scores of s different randomly selected users. For convenience, we denote by $\boldsymbol{X}$ the sampled submatrix. Then, we use an arbitrary base algorithm $\mathcal{A}$ to build a base recommender system on the sampled submatrix $\boldsymbol{X}$. We use the base recommender system to recommend top-N' items (denoted as $\mathcal{A}(\boldsymbol{X},u)$) to each user u in the sampled submatrix. Since the submatrix is randomly sampled, the recommended top-N' items are also random. To consider such randomness, we denote by p_i the probability that item i is recommended to a user u. Formally, we define p_i as follows: $p_i = \Pr(i \in \mathcal{A}(\boldsymbol{X},u))$. We call p_i *item probability*.

Note that we consider $\mathcal{A}(\boldsymbol{X},u)$ is an empty set when u is not in the sampled submatrix $\boldsymbol{X}$, as many recommender systems make recommendations for users in the rating-score matrix that was used to build the recommender systems. Since a base recommender system is built using s rows of the rating-score matrix $\boldsymbol{M}$, we can build $\binom{n}{s}$ base recommender systems in total, where n is the total number of users/rows in $\boldsymbol{M}$. Essentially, our item probability p_i is the fraction of the $\binom{n}{s}$ base recommender systems that recommend item i to the user u.

Poisoned item probability p'_i: Under data poisoning attacks, the rating-score matrix $\boldsymbol{M}$ becomes a poisoned version $\boldsymbol{M}'$. We denote by $\boldsymbol{Y}$ a random submatrix with s rows sampled from $\boldsymbol{M}'$. Moreover, we define *poisoned item probability* $p'_i = \Pr(i \in \mathcal{A}(\boldsymbol{Y},u))$, i.e., p'_i is the probability that the item i is in the top-N' items recommended to u when the base recommender system is built based on $\boldsymbol{Y}$.

Our ensemble recommender system $\mathcal{T}$: Our ensemble recommender system (denoted as $\mathcal{T}$) recommends the top-N items with the largest item probabilities p_i's for a user u. Essentially, our ensemble recommender system takes a majority vote among the $\binom{n}{s}$ base recommender systems. In particular, our ensemble recommender system essentially recommends the top-N items that are the most frequently recommended by the $\binom{n}{s}$ base recommender systems to a user. For simplicity, we use $\mathcal{T}(\boldsymbol{M},u)$ to denote the set of top-N items recommended to the user u by our ensemble recommender system $\mathcal{T}$ when the rating-score matrix is $\boldsymbol{M}$. Note that N' and N are different parameters, i.e., N' is the number of items recommended to a user by a base recommender system while N is the number of items recommended to a user by our ensemble recommender system. We will explore their impact on our ensemble recommender system in experiments.

Under data poisoning attacks, our ensemble recommender system $\mathcal{T}$ uses the poisoned item probabilities to make recommendations. Specifically, $\mathcal{T}(\boldsymbol{M}',u)$ is the set of top-N items with the largest poisoned item probabilities p'_i's that are recommended to u by $\mathcal{T}$ under attacks.

4.3 Deriving the Certified Intersection Size

We show that our ensemble recommender system algorithm $\mathcal{T}$ is (e,r)-provably robust. In particular, for any given number of fake users e, we can derive the certified intersection size r of $\mathcal{T}$ for any user u. Specifically, given an arbitrary set of items I_u, we show that at least r of the recommended top-N items $\mathcal{T}(\boldsymbol{M}',u)$ are in I_u when there are at most e fake users, no matter what rating scores they use. Later, we can replace I_u as our desired sets of items. Formally, we aim to show the following: $\min_{\boldsymbol{M}' \in \mathcal{L}(\boldsymbol{M},e)} |I_u \cap \mathcal{T}(\boldsymbol{M}',u)| \geq r$, where $\mathcal{L}(\boldsymbol{M},e)$ is the set of all possible poisoned-rating-score matrices and denotes all data poisoning attacks with at most e fake users. Next, we first overview our main idea and then show our theorem.

Overview of our derivation: Our proof is based on the *law of contraposition*. Suppose we have a statement: $U \longrightarrow V$, whose contraposition is $\neg V \longrightarrow \neg U$. The law of contraposition means that a statement is true if and only if its contraposition is true. We define a predicate V as $V : |I_u \cap \mathcal{T}(\boldsymbol{M}',u)| \geq r$. The predicate V is true if at least r of the top-N items recommended by $\mathcal{T}$ for u are in I_u when the poisoned-rating-score matrix is a given $\boldsymbol{M}'$. Then, we derive a necessary condition (denoted as $\neg U$) for $\neg V$ to be true, i.e., we have $\neg V \longrightarrow \neg U$. By the law of contraposition, we know that V is true if U is true. Roughly speaking, U means that the rth largest poisoned item probability for items in I_u is larger than the $(N-r+1)$th largest poisoned item probability for items in $I \setminus I_u$, when the poisoned-rating-score matrix is $\boldsymbol{M}'$.

The challenge in deriving the condition for U to be true is that it is hard to compute the poisoned item probabilities p'_i's due to the complexity of recommender system. To address the challenge, we resort to derive a lower bound of p'_i for each $i \in I_u$ and an upper bound of p'_j for each $j \in I \setminus I_u$. In particular, we derive the lower/upper bounds of poisoned item probabilities using lower/upper bounds of item probabilities. We consider lower/upper bounds of item probabilities instead of their exact values, because it is challenging to compute them exactly. Suppose we have a lower bound $\underline{p_i}$ of p_i for each $i \in I_u$ and an upper bound $\overline{p}_j$ of p_j for each $j \in I \setminus I_u$, i.e., we have the following:

$$p_i \geq \underline{p_i} \text{ and } p_j \leq \overline{p}_j, \tag{3}$$

where $i \in I_u$ and $j \in I \setminus I_u$. In the next section, we design an algorithm to estimate such lower/upper bounds of item probabilities. Given the lower/upper bounds $\underline{p_i}$ and $\overline{p}_j$, we derive a lower bound of p'_i for each $i \in I_u$ and an upper bound of p'_j for each $j \in I \setminus I_u$ via a variant Neyman-Pearson Lemma [28] that we develop. Our variant is applicable to multiple functions, while the standard Neyman-Pearson Lemma is only applicable to one function.

Next, we show our intuition to derive the upper and lower bounds (please refer to the proof of Theorem 1 for formal analysis) of the poisoned item probabilities. We denote by Φ the union of the domain spaces of $\boldsymbol{X}$ and $\boldsymbol{Y}$, i.e., each element in Φ is a submatrix with s rows sampled from $\boldsymbol{M}$ or $\boldsymbol{M}'$. Our idea is to find subsets in Φ such that we can apply our variant of the Neyman-Pearson Lemma to derive the upper/lower bounds of the poisoned item probabilities. Moreover, the upper/lower bounds are related to the probabilities that the random submatrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ are in the subsets, which can be easily computed. We denote by P (or A) the set of submatrices sampled from $\boldsymbol{M}$ (or $\boldsymbol{M}'$) that include the user u.

Deriving a lower bound of p'_i, $i \in I_u$: We can find a subset $C_i \subseteq P$ such that we have $\Pr(\boldsymbol{X} \in C_i) = p^*_i \triangleq \frac{\lfloor p_i \cdot \binom{n}{s} \rfloor}{\binom{n}{s}}$. Note that we can find such a subset because p^*_i is an integer multiple of $1/\binom{n}{s}$. Then, via our variant of the Neyman-Pearson Lemma, we can derive a lower bound of p'_i using the probability that the random submatrix $\boldsymbol{Y}$ is in the subset C_i, i.e., we have: $p'_i \geq \Pr(\boldsymbol{Y} \in C_i),\ \forall i \in I_u$.

Deriving an upper bound of p'_j, $j \in I \setminus I_u$: We first find a subset $C'_j \subseteq P$ such that we have the following: $\Pr(\boldsymbol{X} \in C'_j) = \overline{p}^*_j = \frac{\lceil \overline{p}_j \cdot \binom{n}{s} \rceil}{\binom{n}{s}}$. Given the subset C'_j, we further define a subset $C_j = C'_j \cup (A \setminus P)$. Then, based on our variant of the Neyman-Pearson Lemma, we derive an upper bound of p'_j using the probability that the random submatrix $\boldsymbol{Y}$ is in the subset C_j, i.e., we have:

$$p'_j \leq \Pr(\boldsymbol{Y} \in C_j). \tag{4}$$

In our derivation, we further improve the upper bound via jointly considering multiple items in $I \setminus I_u$. Suppose $\mathcal{H}_c \subseteq I \setminus I_u$ is a set of c items. We denote $\overline{p}_{\mathcal{H}_c} = \sum_{j \in \mathcal{H}_c} \overline{p}_j$. Then, we can find a subset $C'_{\mathcal{H}_c}$ such that we have the following: $\Pr(\boldsymbol{X} \in C'_{\mathcal{H}_c}) = \overline{p}^*_{\mathcal{H}_c} \triangleq \frac{\lceil (\overline{p}_{\mathcal{H}_c}/N') \cdot \binom{n}{s} \rceil}{\binom{n}{s}}$. Given the subset $C'_{\mathcal{H}_c}$, we further define a subset $C_{\mathcal{H}_c} = C'_{\mathcal{H}_c} \cup (A \setminus P)$. Then, we have the following upper bound for the smallest poisoned item probability in the set $\{p'_j | j \in \mathcal{H}_c\}$:

$$\min_{j \in \mathcal{H}_c} p'_j \leq \frac{N' \cdot \Pr(\boldsymbol{Y} \in C_{\mathcal{H}_c})}{c}. \tag{5}$$

Finally, we can combine the upper bounds in Equation (4) and (5) to derive an upper bound of the $(N-r+1)$th largest poisoned item probability in $I \setminus I_u$. Note that we don't jointly consider multiple items in I_u when deriving the lower bounds for poisoned item probabilities in I_u because it does not improve the lower bounds.

Formally, we have the following theorem:

Theorem 1. *Suppose we have a rating-score matrix $\boldsymbol{M}$, a user u, and an arbitrary set of k items $I_u = \{\mu_1, \mu_2, \cdots, \mu_k\}$. Furthermore, we have a lower bound $\underline{p_i}$ for each $i \in I_u$ and an upper bound $\overline{p}_j$ for each $j \in I \setminus I_u$ that satisfy Equation (3). Without loss of generality, we assume $\underline{p_{\mu_1}} \geq \underline{p_{\mu_2}} \geq \cdots \geq \underline{p_{\mu_k}}$. Under any data poisoning attacks with at most e fake users, we have the following guarantee: $\min_{\boldsymbol{M}' \in \mathcal{L}(\boldsymbol{M}, e)} |I_u \cap \mathcal{T}(\boldsymbol{M}', u)| \geq r$, where r is the solution to the following optimization problem or 0 if it does not have a solution:*

$$r = \operatorname*{argmax}_{r' \in \{1,2,\cdots,\min(k,N)\}} r'$$
$$s.t.\ \underline{p^*_{\mu_{r'}}} > \min\Big(\min_{c=1}^{N-r'+1} \frac{N' \cdot (\overline{p}^*_{\mathcal{H}_c} + \sigma)}{c}, \overline{p}^*_{v_1} + \sigma \Big), \tag{6}$$

where $n' = n + e$, $\sigma = \frac{s}{n'} \cdot \frac{\binom{n'}{s}}{\binom{n}{s}} - \frac{s}{n}$, $\underline{p^*_{\mu_{r'}}} = \frac{\lfloor p_{\mu_{r'}} \cdot \binom{n}{s} \rfloor}{\binom{n}{s}}$, $\mathcal{H}_c = \{v_1, v_2, \cdots, v_c\}$ is the set of c items that have the smallest item-probability upper bounds among the $N - r' + 1$ items with the largest item-probability upper bounds in $I \setminus I_u$, v_1 is the item in $\mathcal{H}_1$, $\overline{p}_{\mathcal{H}_c} = \sum_{j \in \mathcal{H}_c} \overline{p}_j$, $\overline{p}^*_{\mathcal{H}_c} = \frac{\lceil (\overline{p}_{\mathcal{H}_c}/N') \cdot \binom{n}{s} \rceil}{\binom{n}{s}}$, and $\overline{p}^*_{v_1} = \frac{\lceil \overline{p}_{v_1} \cdot \binom{n}{s} \rceil}{\binom{n}{s}}$.

Proof. See Appendix A. □

4.4 Computing the Certified Intersection Size

Given a base algorithm $\mathcal{A}$, a rating-score matrix $\boldsymbol{M}$, a set of genuine users $\mathcal{U} = \{u_1, u_2, \cdots, u_n\}$, a set of items I_u for each genuine user u, the maximum number of fake users e, and a sampling size s, we aim to compute the certified intersection size of our ensemble recommender system for each user in $\mathcal{U}$. The key to compute the certified intersection size is to solve r in the optimization problem in Equation (6). Specifically, given a user u, the key challenge to solve the optimization problem in Equation (6) is how to estimate the item-probability lower bounds $\underline{p_i}$ for $\forall i \in I_u$ and upper bounds $\overline{p}_j$ for $\forall j \in I \setminus I_u$. One naive way is to build $\binom{n}{s}$ base recommender systems and compute the exact item probabilities. However, such approach is computationally infeasible as $\binom{n}{s}$ is huge. To address the challenge, we design an algorithm to estimate lower/upper bounds of item probabilities via building $T << \binom{n}{s}$ base recommender systems. Next, we introduce estimating the lower/upper bounds of the item probabilities, solving r using the estimated item-probability bounds, and our complete algorithm.

Estimating the item-probability lower/upper bounds: We randomly sample T submatrices from $\boldsymbol{M}$, where each submatrix contains s rows of $\boldsymbol{M}$. For simplicity, we denote them as $\Gamma_1, \Gamma_2, \cdots, \Gamma_T$. Then, we build a base recommender system for each submatrix Γ_t using the base algorithm $\mathcal{A}$, where $t = 1, 2, \cdots, T$. Given a user u, we use each base recommender system to recommend top-N' items for the user. We denote by $\mathcal{A}(\Gamma_t, u)$ the set of top-N' items recommended to the user u by the base recommender system built on the submatrix Γ_t. Note that $\mathcal{A}(\Gamma_t, u)$ is empty if the user u is not in the submatrix Γ_t.

Algorithm 1: BINARYSEARCH

Input: $e, s, N', N, I_u, \{\underline{p_i} | i \in I_u\}$, and $\{\overline{p}_j | j \in I \setminus I_u\}$
Output: r_u
$low, high \leftarrow 1, \min(|I_u|, N)$
while $low < high$ **do**
  $r' = \lceil (low + high)/2 \rceil$
  if VERIFYCONSTRAINT$(r', e, s, N', N, I_u, \{\underline{p_i} | i \in I_u\}, \{\overline{p}_j | j \in I \setminus I_u\}) == 1$ **then**
    $low \leftarrow r'$
  else
    $high \leftarrow r' - 1$
  end if
end while
if VERIFYCONSTRAINT$(r', e, s, N', N, I_u, \{\underline{p_i} | i \in I_u\}, \{\overline{p}_j | j \in I \setminus I_u\}) == 1$ **then**
  return r'
else
  return 0
end if

We denote by T_i the frequency of an item i among the recommended top-N' items of the T base recommender systems, i.e., T_i is the number of base recommender systems whose top-N' recommended items for u include i. Based on the definition of item probability p_i, the frequency T_i follows a binomial distribution with parameters T and p_i, i.e., we have the following: $\Pr(T_i = t) = \binom{T}{t} \cdot p_i^t \cdot (1 - p_i)^{T-t}$, where $t = 0, 1, \cdots, T$. Our goal is to estimate a lower or upper bound of p_i given T_i and T. This is essentially a *binomial proportion confidence interval* estimation problem. Therefore, we can leverage the standard Clopper-Pearson method [10] to estimate a lower or upper bound of p_i from a given T_i and T. Formally, we have the following:

$$\underline{p_i} = \text{Beta}(\frac{\alpha_u}{m}; T_i, T - T_i + 1), i \in I_u, \quad (7)$$

$$\overline{p}_j = \text{Beta}(1 - \frac{\alpha_u}{m}; T_j, T - T_j + 1), j \in I \setminus I_u, \quad (8)$$

where $1 - \alpha_u/m$ is the confidence level for estimating the lower/upper bound of one item probability, m is the total number of items, and $\text{Beta}(\beta; \varsigma, \vartheta)$ is the βth quantile of the Beta distribution with shape parameters ς and ϑ. Based on *Bonferroni correction* in statistics, the *simultaneous confidence level* of estimating the lower/upper bounds of the m item probabilities is at least $1 - \alpha_u$. Given an item set $\mathcal{H}_c$ defined in Theorem 1, we can estimate $\overline{p}_{\mathcal{H}_c}$ as $\overline{p}_{\mathcal{H}_c} = \min(\sum_{j \in \mathcal{H}_c} \overline{p}_j, N' - \sum_{i \in I_u} \underline{p_i})$, where both $\sum_{j \in \mathcal{H}_c} \overline{p}_j$ and $N' - \sum_{i \in I_u} \underline{p_i}$ are upper bounds of $p_{\mathcal{H}_c}$, and we use the smaller one.

Solving the optimization problem: We note that Equation (6) has the following property: its left-hand side and right-hand side respectively decreases and increases as r' increases. Thus, given the estimated item-probability bounds, we can efficiently solve the optimization problem in Equation (6) via binary search to obtain r_u for each user u. Algorithm 1 shows our BINARYSEARCH algorithm. The function VERIFYCONSTRAINT verifies whether the constraint in Equation (6) is satisfied for a given r' and returns 1 if so.

Algorithm 2: COMPUTE r

Input: $\boldsymbol{M}$, s, T, $\mathcal{A}$, N', α, e, N, $\mathcal{U}$, and $\{I_u | u \in \mathcal{U}\}$
Output: r_u for each user $u \in \mathcal{U}$
$\Gamma_1, \Gamma_2, \cdots, \Gamma_T \leftarrow$ RANDOMSAMPLE$(\boldsymbol{M}, s)$
for u **in** $\mathcal{U}$ **do**
  counts$[i] \leftarrow \sum_{t=1}^{T} \mathbb{I}(i \in \mathcal{A}(\Gamma_t, u)), i \in \{1, 2, \cdots, m\}$
  $\underline{p_i}, \overline{p}_j \leftarrow$ BOUNDEST(counts, $\frac{\alpha}{n}$), $i \in I_u, j \in I \setminus I_u$
  $r_u =$ BINARYSEARCH$(e, s, N', N, I_u, \{\underline{p_i} | i \in I_u\}, \{\overline{p}_j | j \in I \setminus I_u\})$
end for
return $\{r_u | u \in \mathcal{U}\}$

Complete algorithm: Algorithm 2 shows our complete algorithm to compute the certified intersection size r_u for each user $u \in \mathcal{U} = \{u_1, u_2, \cdots, u_n\}$. The function RANDOMSAMPLE randomly samples T submatrices, each of which contains s rows sampled from $\boldsymbol{M}$ uniformly at random. BOUNDEST estimates the item-probability bounds with a confidence level $1 - \frac{\alpha}{n}$, i.e., $\alpha_u = \frac{\alpha}{n}$, for each user $u \in \mathcal{U}$, based on Equation (7) - (8). BINARYSEARCH solves the optimization problem in Equation (6) via binary search to obtain r_u for u based on the estimated item-probability bounds. Note that Algorithm 2 requires a clean rating-score matrix $\boldsymbol{M}$, which may be sampled from the clean data distribution.

Due to randomness, the estimated item-probability bounds may be incorrect, e.g., an estimated item-probability lower bound is larger than the true item probability for some item and some user or an estimated item-probability upper bound is smaller than the true item probability for some item and some user. When such estimation error happens for a user u, the solved certified intersection size r_u is incorrect for u. Since the simultaneous confidence level of estimating the item-probability bounds in our algorithm is at least $1 - \frac{\alpha}{n}$ for any user $u \in \mathcal{U}$, the probability of having at least one incorrectly estimated item-probability bound for any user $u \in \mathcal{U}$ is at most $\frac{\alpha}{n}$. Moreover, our following theorem shows that the probability of having an incorrect certified intersection size r_u for at least one user in $\mathcal{U}$ is bounded by α:

Theorem 2. *The probability that our Algorithm 2 returns an incorrect r_u for at least one user in $\mathcal{U}$ is at most α.*

Proof. See Appendix B. □

5 Evaluation

5.1 Experimental Setup

Datasets: We mainly evaluate PORE on MovieLens-100k and MovieLens-1M benchmark datasets [1, 14], which con-

sist of around 100,000 and 1,000,000 rating scores, respectively. Specifically, MovieLens-100k contains 943 users and 1,682 items, where each user rated 106 items on average. MovieLens-1M contains 6,040 users and 3,952 items, where each user on average rated 166 items. Following [3], for each user, we sample 75% of its rating scores as training data and treat its remaining rated items as test items. The users' training data form the rating-score matrix and are used to build recommender systems, while their test items are used to evaluate the performance of the recommended top-N items.

Base algorithms: PORE is applicable to any base algorithm. To show such generality, we evaluate two base algorithms, i.e., Item-based Recommendation (IR) [3] and Bayesian Personalized Ranking (BPR) [29]. We adopt their public implementations [3]. We adopt IR because it has been widely deployed in industry [24]. We adopt BPR because it achieves state-of-the-art performance according to recent benchmarks released by Microsoft [2].

Evaluation metrics: When there are no data poisoning attacks, *Precision@N*, *Recall@N*, and *F1-Score@N* are standard metrics to evaluate the performance of a recommender system. We denote by $\mathcal{E}_u$ the set of test items for a user u. Precision@N for a user u is the fraction of the top-N items recommended for u that are in $\mathcal{E}_u$, Recall@N for u is the fraction of the items in $\mathcal{E}_u$ that are in the top-N items recommended for u, while F1-Score@N for a user u is the harmonic mean of the user's Precision@N and Recall@N. The Precision@N (or Recall@N or F1-Score@N) of a recommender system algorithm is the users' average Precision@N (or Recall@N or F1-Score@N).

Under data poisoning attacks, Precision@N, Recall@N, and F1-Score@N are insufficient to evaluate a recommender system algorithm. This is because they may be different under different data poisoning attacks, and it is infeasible to enumerate all possible attacks. To address the challenge, we propose to evaluate a recommender system algorithm using *certified Precision@N*, *certified Recall@N*, and *certified F1-Score@N* under attacks. Like the standard Precision@N (or Recall@N or F1-Score@N), calculating our certified Precision@N, certified Recall@N, and certified F1-Score@N also only requires a clean rating-score matrix and thus does not depend on any specific data poisoning attack. Certified Precision@N (or certified Recall@N or certified F1-Score@N) is a *lower bound* of Precision@N (or Recall@N or F1-Score@N) under any data poisoning attacks with at most e fake users. For instance, a certified Precision@N of 0.3 means that a recommender system achieves at least Precision@N of 0.3 when the number of fake users is at most e, no matter what rating scores they use. Specifically, our certified Precision@N for a user u is the least fraction of the top-N recommended items for u that are guaranteed to be in $\mathcal{E}_u$ when there are at most e fake users; our certified Recall@N for u is the least fraction of u's test items $\mathcal{E}_u$ that are guaranteed to be in the top-N recommended items; while certified F1-Score@N for a user is the harmonic mean of the user's certified Precision@N and certified Recall@N. Formally, we have the following for a user u:

$$\text{Certified Precision@}N = \min_{\boldsymbol{M}' \in \mathcal{L}(\boldsymbol{M},e)} \frac{|\mathcal{E}_u \cap \mathcal{T}(\boldsymbol{M}', u)|}{N}, \quad (9)$$

$$\text{Certified Recall@}N = \min_{\boldsymbol{M}' \in \mathcal{L}(\boldsymbol{M},e)} \frac{|\mathcal{E}_u \cap \mathcal{T}(\boldsymbol{M}', u)|}{|\mathcal{E}_u|}, \quad (10)$$

$$\text{Certified F1-Score@}N = \min_{\boldsymbol{M}' \in \mathcal{L}(\boldsymbol{M},e)} \frac{2 \cdot |\mathcal{E}_u \cap \mathcal{T}(\boldsymbol{M}', u)|}{|\mathcal{E}_u| + N}, \quad (11)$$

where $|\cdot|$ is the size of a set. We can compute the certified Precision@N, certified Recall@N, and certified F1-Score@N for each user by Algorithm 2. In particular, we can compute certified intersection size r_u for each user u using Algorithm 2 by letting $I_u = \mathcal{E}_u$. Given r_u, the certified Precision@N, certified Recall@N, and certified F1-Score@N for a user u are at least $\frac{r_u}{N}$, $\frac{r_u}{|\mathcal{E}_u|}$, and $\frac{2r_u}{|\mathcal{E}_u|+N}$, respectively. A recommender system algorithm's certified Precision@N (or Recall@N or F1-Score@N) is the average of the *genuine users'* certified Precision@N (or Recall@N or F1-Score@N).

Compared methods: We note that PORE is the first provably robust recommender system algorithm against data poisoning attacks. Therefore, there are no prior recommender system algorithms we can compare with in terms of *certified* Precision@N, Recall@N, and F1-Score@N. However, Jia et al. [19] showed that bagging can be used to build provably robust defense against data poisoning attacks for *machine learning classifiers*, which we extend to recommender systems and compare with our PORE. Roughly speaking, given a training dataset, bagging trains multiple base classifiers, each of which is trained on a random subset of training examples in the training dataset. Given a testing input, bagging uses each base classifier to predict its label and takes a majority vote among the predicted labels as the final predicted label for the testing input. Jia et al. showed that bagging can guarantee that the predicted label for an input is provably unaffected by a bounded number of fake training examples injected into the training dataset.

We generalize their provable guarantee to derive certified intersection size for each genuine user in recommender systems, which can be then used to compute certified Precision@N, Recall@N, and F1-Score@N for bagging. Specifically, we treat a user as a testing input, an item as a label, and a base recommender system as a base classifier in the terminology of bagging. Like our PORE, the generalized bagging builds T base recommender systems and takes majority vote among them to recommend top-N items to each user. Note that since a base classifier predicts one label for a testing input, we set $N' = 1$, i.e., a base recommender system (i.e., a base classifier) recommends top-1 item (i.e., predicts one label) for a user (i.e., a testing input). Finally, for each user, bagging recommends him/her the N items with the largest (poisoned)

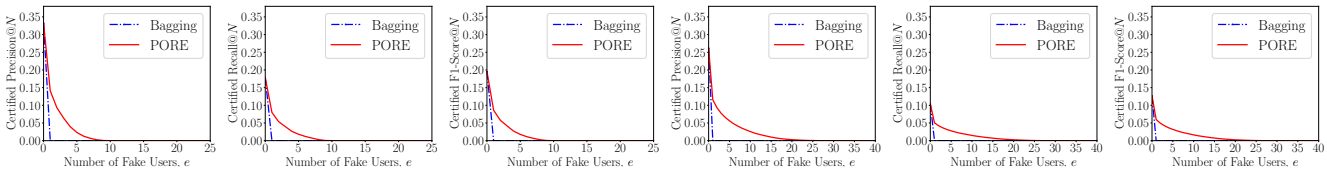

Figure 3: Our PORE outperforms bagging when extended from classifiers to recommender systems on MovieLens-100k (left three) and MovieLens-1M (right three), where $N = 10$ and base algorithm is IR.

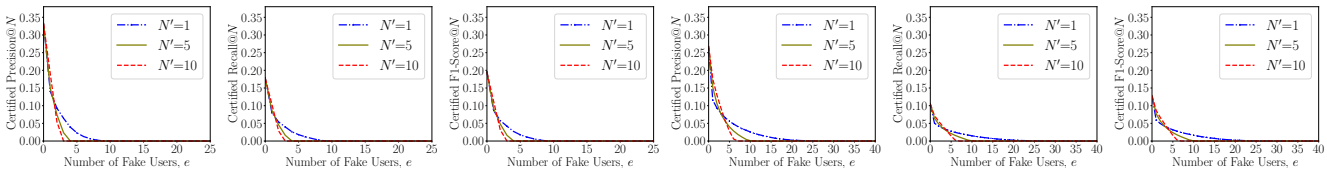

Figure 4: Impact of N' on the certified Precision@N, certified Recall@N, and certified F1-Score@N of ensemble IR for MovieLens-100k (left three) and MovieLens-1M (right three), where $N = 10$.

item probabilities. Given a set of items I_u for a user u, we denote by $\underline{p_i}$ a lower bound of item probability p_i of item $i \in I_u$. We use $\overline{p}_l$ to denote the largest upper bound of item probabilities of items in $I \setminus I_u$, i.e., $\overline{p}_l = \max_{j \in I \setminus I_u} \overline{p}_j$. We can estimate these item-probability bounds using our method in Equation 7 and 8. Given $\underline{p_i}$ and $\overline{p}_l$, we can compute an integer Z_i based on Theorem 1 in bagging [19]. Roughly speaking, bagging can guarantee the poisoned item probability p'_i is larger than p'_l under any data poisoning attacks with at most Z_i fake users. Therefore, given at most e fake users, the certified intersection size of bagging for a user u can be computed as $\min\{\sum_{i \in I_u} \mathbb{I}(Z_i \geq e), N\}$, where $\mathbb{I}$ is an indicator function.

We note that when bagging is extended to recommender systems, both N' and N can only be 1 when using the techniques in [19] to derive its provable robustness guarantees. PORE can be viewed as an extension of bagging to recommender systems, but N' and N can be arbitrary positive integers. Due to such differences, we propose new techniques to derive the robustness guarantee of PORE. Our major technical contribution is to derive a better guarantee for bagging applied to recommender systems.

Parameter setting: PORE has the following parameters: N' is the number of items recommended by a base recommender system for a user, N is the number of items recommended by our ensemble recommender system for a user, T is the number of base recommender systems, $1-\alpha$ is the confidence score, and s is the number of rows sampled from the rating-score matrix in each submatrix. Unless otherwise mentioned, we adopt the following default parameter settings: $N' = 1$, $N = 10$, $T = 100,000$, $\alpha = 0.001$, $s = 200$ for MovieLens-100k, and $s = 500$ for MovieLens-1M. We will study the impact of each parameter on the certified Precision@N, certified Recall@N, and certified F1-Score@N of PORE while fixing the remaining parameters to their default settings. We call our ensemble recommender system *ensemble IR* (or *ensemble BPR*) when the base algorithm is IR (or BPR). By default, we use IR as the base algorithm because of its scalability.

5.2 Experimental Results

We report Precision@N/Recall@N/F1-Score@N under no attacks (i.e., $e = 0$), while we report certified Precision@N /Recall@N/F1-Score@N under attacks (i.e., $e \geq 1$).

Our PORE outperforms bagging [19]: Figure 3 compares our PORE with bagging on the two datasets. We find that our PORE substantially outperforms bagging when extended from classifiers to recommender systems. The reason is that PORE jointly considers multiple items when deriving the certified intersection size. In contrast, bagging can only consider each item independently when extended to recommender systems, and thus achieve a suboptimal certified intersection size.

Impact of N': Figure 4 shows the impact of N'. We have two observations. First, our method has similar Precision@N/Recall@N/F1-Score@N for different N' when there are no attacks (i.e., $e = 0$). Second, a smaller N' achieves a lower certified Precision@N, certified Recall@N, or certified F1-Score@N when e is small (e.g., $e = 1$), but the curve has a longer tail. In other words, a smaller N' is more robust against data poisoning attacks as the number of fake users e increases. The reason is that an attack has a smaller manipulation space when N' is smaller. This observation is also consistent with our theoretical result in Equation (6). Specifically, given the same item-probability lower/upper bounds, a smaller N' may lead to a larger certified intersection size. Therefore, we set N' to be 1 by default in our experiments.

Impact of N: Figure 5 shows the impact of N. The results show that N achieves a tradeoff between Precision@N under no attacks (i.e., $e = 0$) and robustness under attacks. Specifically, a smaller N achieves a higher Precision@N under no attacks but the certified Precision@N decreases more quickly as e increases. The certified Recall@N increases as N increases. The reason is that more items are recommended to each user as N increases. The certified F1-Score@N drops more quickly as e increases when N is smaller, because the certified Recall@N drops more quickly.

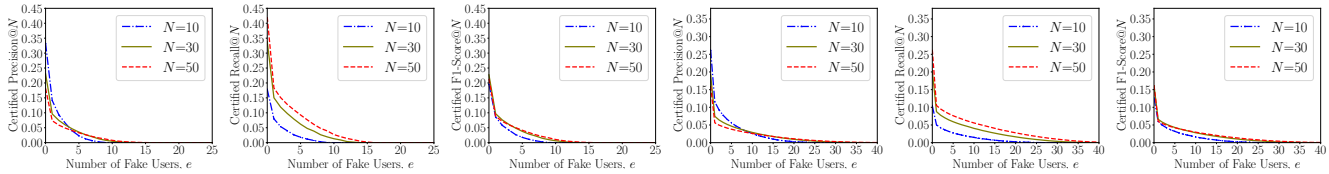

Figure 5: Impact of N on the certified Precision@N, certified Recall@N, and certified F1-Score@N of ensemble IR for MovieLens-100k (left three) and MovieLens-1M (right three).

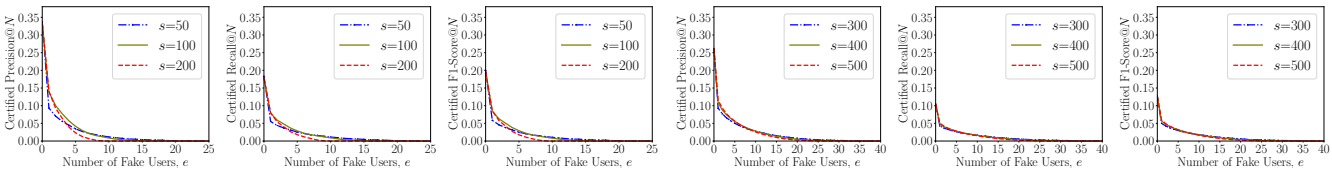

Figure 6: Impact of s on the certified Precision@N, certified Recall@N, and certified F1-Score@N of ensemble IR for MovieLens-100k (left three) and MovieLens-1M (right three), where $N = 10$.

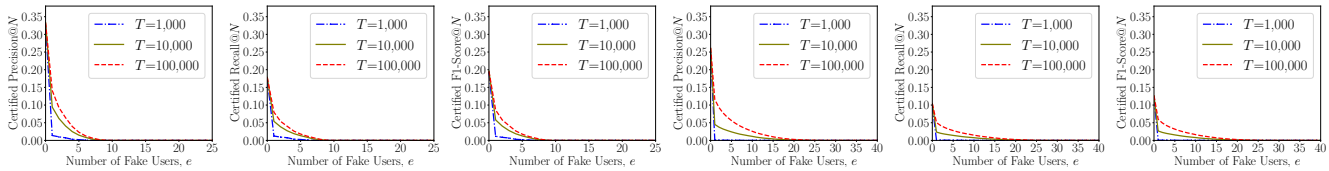

Figure 7: Impact of T on the certified Precision@N, certified Recall@N, and certified F1-Score@N of ensemble IR for MovieLens-100k (left three) and MovieLens-1M (right three), where $N = 10$.

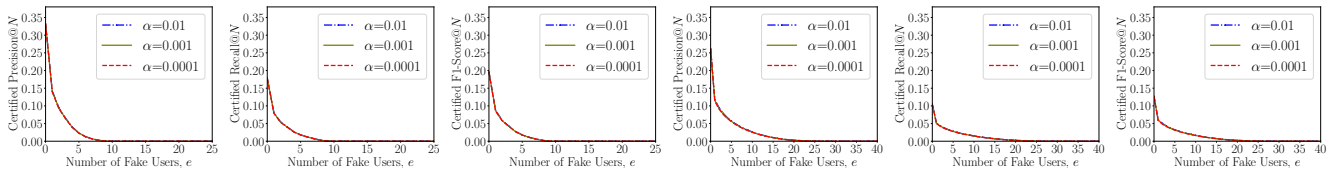

Figure 8: Impact of α on the certified Precision@N, certified Recall@N, and certified F1-Score@N of ensemble IR for MovieLens-100k (left three) and MovieLens-1M (right three), where $N = 10$.

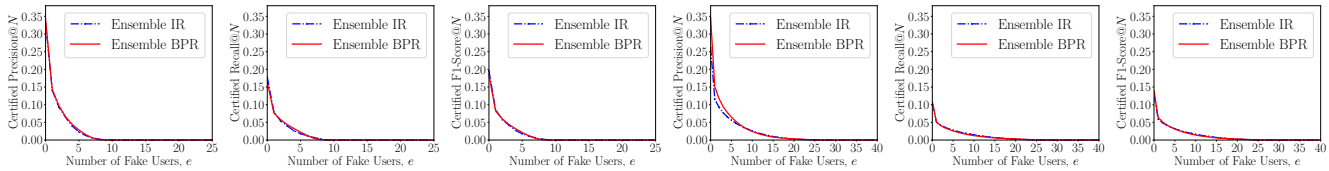

Figure 9: Comparing the certified Precision@N, certified Recall@N, and certified F1-Score@N of ensemble IR and ensemble BPR for MovieLens-100k (left three) and MovieLens-1M (right three), where $N = 10$.

Impact of s: Figure 6 shows the impact of s. We have two observations. First, our method achieves similar Precision@N, Recall@N, and F1-Score@N for different s when there are no attacks (i.e., $e = 0$). Second, a larger s achieves a larger certified Precision@N, certified Recall@N, or certified F1-Score@N when e is small, but they decrease more quickly as e increases. This is because it's more likely to sample fake users in a submatrix when s is larger.

Impact of T and α: Figure 7 and 8 show the impact of T and α, respectively. We have the following observations. First, Precision@N, Recall@N, or F1-Score@N is similar for different T when there are no attacks. In other words, a small T is enough for our ensemble recommender system to achieve good recommendation performance when there are no attacks. Second, certified Precision@N, certified Recall@N, or certified F1-Score@N increases as T or α increases. The reason is that a larger T or α can produce tighter estimated item-probability lower/upper bounds, based on which we may compute larger certified intersection sizes r in our Algorithm 2. Therefore, we use a larger T by default in our experiments to better show the certified Precision@N, certified Recall@N, and certified F1-Score@N of PORE. We also observe that the certified Precision@N, certified Recall@N, and certified F1-Score@N are insensitive to α once it is small enough.

Ensemble IR vs. ensemble BPR: Figure 9 compares ensemble IR and ensemble BPR. The results show that they achieve similar certified Precision@N/Recall@N/F1-Score@N. One exception is that ensemble BPR achieves higher certified

Table 1: Precision@10, Recall@10, and F1-Score@10 of IR, Ensemble IR, BPR, and Ensemble BPR under no attacks.

(a) MovieLens-100k

Algorithm	Precision@10	Recall@10	F1-Score@10
IR	0.330753	0.176385	0.193783
Ensemble IR	0.332556	0.178293	0.195624
BPR	0.349841	0.181807	0.199426
Ensemble BPR	0.352280	0.173296	0.193362

(b) MovieLens-1M

Algorithm	Precision@10	Recall@10	F1-Score@10
IR	0.270116	0.104350	0.127704
Ensemble IR	0.262616	0.103638	0.126191
BPR	0.324449	0.118385	0.144765
Ensemble BPR	0.362945	0.119441	0.151509

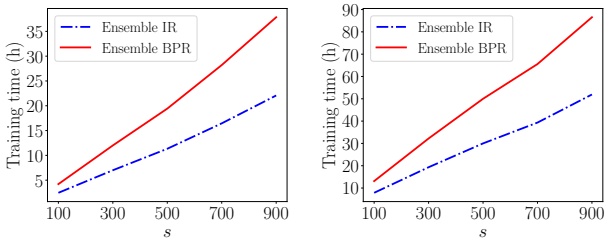

Figure 10: Training time of PORE as a function of s on MovieLens-100k (left) and MovieLens-1M (right).

Precision@N on MovieLens-1M dataset.

Standard recommender system vs. ensemble recommender system under no attacks: Table 1 compares standard recommender systems and our ensemble recommender systems with respect to the standard Precision@N, Recall@N, and F1-Score@N when there are no attacks, where $s = 300$ for MovieLens-100k and $s = 1,000$ for MovieLens-1M. A standard recommender system leverages IR (or BPR) to train a single recommender system on the entire rating-score matrix. The results show that our ensemble recommender system achieves comparable performance with a standard recommender system when there are no attacks. Ensemble BPR achieves higher Precision@N than ensemble IR on both datasets. The reason is that BPR achieves higher precision than IR at training the base recommender systems.

Training time of ensemble IR and BPR: PORE trains T base recommender systems, each of which is trained using rating scores of s users. Figure 10 shows the training time of ensemble IR/BPR as a function of s, while Figure 11 shows the impact of T on the training time of ensemble IR/BPR. As expected, the training time of ensemble IR/BPR increases linearly as s or T increases. This is because a larger s means each base recommender system is trained using more data, while a larger T means more base recommender systems are

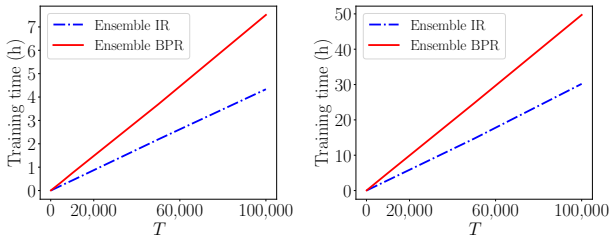

Figure 11: Training time of PORE as a function of T on MovieLens-100k (left) and MovieLens-1M (right).

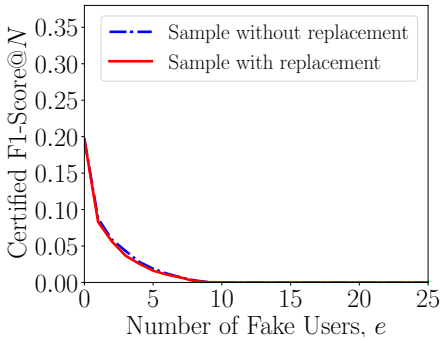

Figure 12: Comparing certified F1-Score@N of sampling with and without replacement for ensemble IR on MovieLens-100k, where $N = 10$.

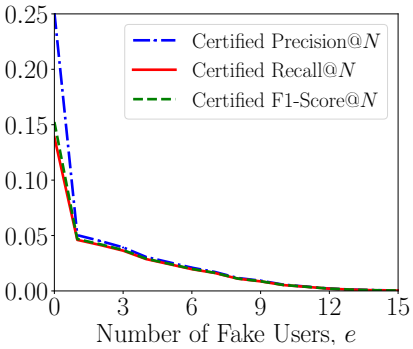

Figure 13: Certified Precision@N, certified Recall@N, and certified F1-Score@N of ensemble IR for MovieLens-10M, where $N = 10$.

trained. Ensemble IR is more efficient than ensemble BPR because IR is more efficient than BPR.

Sampling with vs. without replacement: PORE randomly samples a submatrix without replacement, i.e., the sampled s users are different in a submatrix. Sampling with replacement means that the s users may have duplicates, i.e., a submatrix may include rating scores of less than s unique users. We can extend our theoretical guarantees to sampling with replacement. The theoretical analysis is similar to sampling without replacement, so we omit it for simplicity. Figure 12 compares the certified F1-Score@N of sampling with and

without replacement for ensemble IR on MovieLens-100k dataset. Our results show that sampling with and without replacement achieves comparable certified F1-Score@N. They also achieve comparable certified Precision@N and certified Recall@N, which we omit for simplicity.

Evaluation on a large dataset: We also evaluate PORE on MovieLens-10M [1], which consists of around 10,000,000 rating scores. We set $s = 5{,}000$, $T = 1{,}000$, and IR as the base recommender system algorithm, and we adopt the default settings for other parameters. We set a smaller T than our previous experiments because it is more expensive to train each base recommender system on MovieLens-10M. Figure 13 shows the results. Our results indicate that our method is applicable to a large dataset and can derive certified performance guarantees against data poisoning attacks. We note that the certified Precision@N/Recall@N/F1-Score@N of bagging also reduces to 0 with just 1 fake user.

6 Related Work

Data poisoning attacks to recommender systems: Many data poisoning attacks to recommender systems [6,9,11,12,18, 22,23,27,32–34,38,39,42,45,46] have been proposed. Early attacks are algorithm-agnostic, i.e., the crafted rating scores of fake users do not depend on the recommender system algorithm [6,22,27]. Recently, more advanced data poisoning attacks [11, 12, 18, 23, 33, 39, 42, 46] have been optimized for specific recommender system algorithms. For instance, Yang et al. [42] proposed to inject fake co-visitations to poison association rule based recommender systems, Fang et al. proposed optimized data poisoning attacks to graph based recommender systems [12] and matrix factorization based recommender systems [11], and Huang et al. [18] proposed data poisoning attacks optimized to deep learning based recommender systems.

Empirical defenses against data poisoning attacks to recommender systems: One family of defenses [7,12,41,44,47] aim to detect fake users via analyzing their abnormal rating score patterns. The key assumption is that the rating scores of fake users and genuine users have different patterns. Fo instance, Burke [7] extracted features from each user's rating scores and trained a classifier to predict whether a user is fake or not. Another family of defenses [8, 17, 25, 26, 30, 35, 43] try to train more robust recommender systems. For instance, adversarial training [13], which was developed to train robust machine learning classifiers, has been extended to train robust recommender systems by multiple work [35, 43]. However, none of the above defenses provides provable robustness guarantees. In particular, they cannot derive certified Precision@N, certified Recall@N, and certified F1-Score@N. As a result, they can still be attacked by adaptive attacks.

Ensemble recommender systems: Ensemble methods have been explored to improve the empirical performance of recommender systems [4, 37, 40]. For instance, around a decade ago, the winning teams [4, 37] in the well-known Netflix competition on predicting user rating scores for movies blended multiple base recommender systems built by different base algorithms. However, these studies are different from ours. The key difference is that they didn't derive the provable robustness guarantees for their ensemble recommender systems. Moreover, they use different ways to aggregate the base recommender systems, e.g., they aggregated the rating scores predicted by the base recommender systems.

Provably robust classifiers against data poisoning attacks: Several works [19, 20] proposed certified defenses against data poisoning attacks for machine learning classifiers. The key difference between classifiers and recommender systems is that an input has a single ground-truth label in a classifier while a user has multiple ground-truth items in a recommender system. As a result, these methods achieve suboptimal certified robustness guarantees when generalized to recommender systems as shown in our experimental results.

7 Discussion and Limitations

Theoretical guarantees: For any given number of fake users, our method can derive a certified intersection size r for each user. Note that our theoretical guarantee still holds even if fake users rate new items. However, when the fraction of fake users is large (e.g., 49%), the derived r and the corresponding certified Precision@N/Recall@N/F1-Score@N may reduce to 0. As the first step on provably robust recommender systems, our method can derive a non-zero r against a moderate number of fake users. It is still an open challenge to derive a non-trivial r for a large fraction of fake users. Essentially, there is a trade-off between performance without attack and robustness, which is controlled by s (number of users in each submatrix). When the fraction of fake users is large, using a small s makes the ensemble recommender system more robust but less accurate.

There are several directions to further improve the theoretical guarantees, i.e., derive a larger r for a given fraction of fake users. First, we considered a very strong threat model, where each fake user can arbitrarily rate all items. For instance, in MovieLens-100k, each fake user can rate up to 1,682 items, which accounts for 1.7% of the rating scores from all genuine users. Fake users that rate a large number of items can be easily detected, as genuine users often rate a small number of items. Therefore, one way to further improve theoretical guarantee is to consider fake users that rate a bounded number of items. Second, PORE is applicable to any base recommender system algorithm without considering the knowledge of the base algorithm. Therefore, the second possible way to derive better theoretical guarantees is to consider the domain knowledge of a specific base algorithm.

Base algorithms and voting mechanisms: We focus on using the same base algorithm to train each base recommender

system in this work. We note that the base recommender systems can be trained using different base algorithms. In particular, our theoretical guarantee holds for any (randomized) base algorithm. Therefore, given a set of base algorithms, we can randomly pick one to train a base recommender system. Moreover, we can view each base recommender system is trained using a randomized base algorithm sampled from the set of base algorithms. PORE uses hard voting when aggregating the items recommended by the base recommender systems. Hard voting has to be used to derive the theoretical guarantee of the ensemble recommender system.

Targeted data poisoning attacks: In this work, we focus on untargeted data poisoning attacks, which aim to reduce the overall performance of a recommender system. Our method can guarantee a lower bound of recommendation performance against any untargeted data poisoning attacks. Targeted data poisoning attacks aim to promote specific attacker-chosen items (called *target items*) [42]. It is an interesting future work to derive provable robustness guarantees against such attacks. Specifically, given a fraction of fake users, we aim to derive an upper bound of the number of genuine users, to which the target items are recommended.

8 Conclusion and Future Work

In this work, we show that PORE can turn an arbitrary base recommender system algorithm to be provably robust against data poisoning attacks via ensembling multiple base recommender systems built by the base algorithm on random subsamples of the rating-score matrix. Our ensemble recommender system guarantees a certain fraction of its top-N items recommended to a user is unaffected by fake users no matter how the attacker crafts their rating scores. Our empirical evaluation confirms that our ensemble recommender system provides provable robustness guarantees. Interesting future work includes deriving better provable robustness guarantees by bounding the number of items that fake users can rate and incorporating the knowledge of the base recommender system algorithm as well as deriving provable robustness guarantees against targeted data poisoning attacks.

Acknowledgements

We thank the anonymous reviewers and shepherd for their constructive comments. This work was supported by NSF under grant No. 2131859, 2112562, 2131859, 1937786, and 1937787, as well as ARO grant No. W911NF2110182.

References

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

[2] Recommenders. https://github.com/microsoft/recommenders.

[3] Andreas Argyriou, Miguel González-Fierro, and Le Zhang. Microsoft recommenders: Best practices for production-ready recommendation systems. In *WWW*, 2020.

[4] Robert M Bell, Yehuda Koren, and Chris Volinsky. The bellkor 2008 solution to the netflix prize. *Statistics Research Department at AT&T Research*, 1, 2008.

[5] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[6] Robin Burke, Bamshad Mobasher, Runa Bhaumik, and Chad Williams. Segment-based injection attacks against collaborative filtering recommender systems. In *ICDM*, 2005.

[7] Robin Burke, Bamshad Mobasher, Chad Williams, and Runa Bhaumik. Classification features for attack detection in collaborative recommender systems. In *KDD*, 2006.

[8] Huiyuan Chen and Jing Li. Adversarial tensor factorization for context-aware recommendation. In *RecSys*, 2019.

[9] Konstantina Christakopoulou and Arindam Banerjee. Adversarial attacks on an oblivious recommender. In *RecSys*, 2019.

[10] Charles J Clopper and Egon S Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4), 1934.

[11] Minghong Fang, Neil Zhenqiang Gong, and Jia Liu. Influence function based data poisoning attacks to top-n recommender systems. In *WWW*, 2020.

[12] Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. Poisoning attacks to graph-based recommender systems. In *ACSAC*, 2018.

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[14] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *ACM TIIS*, 5(4), 2015.

[15] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*, 2020.

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, 2017.

[17] Seira Hidano and Shinsaku Kiyomoto. Recommender systems robust to data poisoning using trim learning. In *ICISSP*, 2020.

[18] Hai Huang, Jiaming Mu, Neil Zhenqiang Gong, Qi Li, Bin Liu, and Mingwei Xu. Data poisoning attacks to deep learning based recommender systems. In *NDSS*, 2021.

[19] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Intrinsic certified robustness of bagging against data poisoning attacks. In *AAAI*, 2021.

[20] Jinyuan Jia, Yupei Liu, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified robustness of nearest neighbors against data poisoning and backdoor attacks. In *AAAI*, 2022.

[21] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[22] Shyong K Lam and John Riedl. Shilling recommender systems for fun and profit. In *WWW*, 2004.

[23] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *NeurIPS*, 2016.

[24] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1), 2003.

[25] Yang Liu, Xianzhuo Xia, Liang Chen, Xiangnan He, Carl Yang, and Zibin Zheng. Certifiable robustness to discrete adversarial perturbations for factorization machines. In *SIGIR*, 2020.

[26] Bhaskar Mehta, Thomas Hofmann, and Wolfgang Nejdl. Robust collaborative filtering. In *RecSys*, 2007.

[27] Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Chad Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *TOIT*, 7(4), 2007.

[28] Jerzy Neyman and Egon Sharpe Pearson. Ix. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London.*, 1933.

[29] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv:1205.2618*, 2012.

[30] Jeff J Sandvig, Bamshad Mobasher, and Robin Burke. Robustness of collaborative recommendation based on association rule mining. In *RecSys*, 2007.

[31] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[32] Carlos E Seminario and David C Wilson. Attacking item-based recommender systems with power items. In *RecSys*, 2014.

[33] Junshuai Song, Zhao Li, Zehong Hu, Yucheng Wu, Zhenpeng Li, Jian Li, and Jun Gao. Poisonrec: An adaptive data poisoning framework for attacking black-box recommender systems. In *ICDE*, 2020.

[34] Jiaxi Tang, Hongyi Wen, and Ke Wang. Revisiting adversarially learned injection attacks against recommender systems. In *RecSys*, 2020.

[35] Jinhui Tang, Xiaoyu Du, Xiangnan He, Fajie Yuan, Qi Tian, and Tat-Seng Chua. Adversarial training towards robust multimedia recommender system. *TKDE*, 32(5), 2019.

[36] Kurt Thomas, Damon McCoy, Chris Grier, Alek Kolcz, and Vern Paxson. Trafficking fraudulent accounts: The role of the underground market in twitter spam and abuse. In *USENIX Security Symposium*, 2013.

[37] Andreas Töscher, Michael Jahrer, and Robert M Bell. The bigchaos solution to the netflix grand prize. *Netflix prize documentation*, 2009.

[38] Soumya Wadhwa, Saurabh Agrawal, Harsh Chaudhari, Deepthi Sharma, and Kannan Achan. Data poisoning attacks against differentially private recommender systems. In *SIGIR*, 2020.

[39] Chenwang Wu, Defu Lian, Yong Ge, Zhihao Zhu, and Enhong Chen. Triple adversarial learning for influence based poisoning attack in recommender systems. In *KDD*, 2021.

[40] Mingrui Wu. Collaborative filtering via ensembles of matrix factorizations. In *KDD Cup and Workshop 2007*, 2007.

[41] Zhiang Wu, Junjie Wu, Jie Cao, and Dacheng Tao. Hysad: A semi-supervised hybrid shilling attack detector for trustworthy product recommendation. In *KDD*, 2012.

[42] Guolei Yang, Neil Zhenqiang Gong, and Ying Cai. Fake co-visitation injection attacks to recommender systems. In *NDSS*, 2017.

[43] Feng Yuan, Lina Yao, and Boualem Benatallah. Adversarial collaborative neural network for robust recommendation. In *SIGIR*, 2019.

[44] Fuzhi Zhang and Quanqiang Zhou. Hht–svm: An online method for detecting profile injection attacks in collaborative recommender systems. *Knowledge-Based Systems*, 2014.

[45] Hengtong Zhang, Yaliang Li, Bolin Ding, and Jing Gao. Practical data poisoning attack against next-item recommendation. In *WWW*, 2020.

[46] Hengtong Zhang, Changxin Tian, Yaliang Li, Lu Su, Nan Yang, Wayne Xin Zhao, and Jing Gao. Data poisoning attack against recommender system using incomplete and perturbed data. In *KDD*, 2021.

[47] Sheng Zhang, Amit Chakrabarti, James Ford, and Fillia Makedon. Attack detection in time series for recommender systems. In *KDD*, 2006.

A Proof of Theorem 3.1

Recall that, given a rating-score matrix $\boldsymbol{M}$ and its poisoned version $\boldsymbol{M}'$, $\boldsymbol{X}$ and $\boldsymbol{Y}$ are respectively two submatrices with s rows randomly sampled from $\boldsymbol{M}$ and $\boldsymbol{M}'$ without replacement. We use Φ to denote the domain space of $\boldsymbol{Y}$, i.e., each element in Φ is a submatrix with s rows sampled from $\boldsymbol{M}'$. Note that since $\boldsymbol{M}$ is a submatrix of $\boldsymbol{M}'$, the domain space of $\boldsymbol{X}$ is a subset of Φ. For simplicity, we define the following notations. Suppose we have $\boldsymbol{Z} \in \Phi$ and another rating-score matrix $\boldsymbol{W}$, we say $\boldsymbol{Z} \prec \boldsymbol{W}$ (or $\boldsymbol{Z} \not\prec \boldsymbol{W}$) if $\boldsymbol{Z}$ is (or is not) in the domain space created by sampling s rows from $\boldsymbol{W}$. We have $\boldsymbol{X} \prec \boldsymbol{M}$ and $\boldsymbol{Y} \prec \boldsymbol{M}'$ based on our defined notations. Similarly, give a user u and $\boldsymbol{Z} \in \Phi$, we say $u \vdash \boldsymbol{Z}$ if $\boldsymbol{Z}$ contains rating scores of user u. We say $u \nvdash \boldsymbol{Z}$ if $\boldsymbol{Z}$ does not contain user u's rating scores.

The following lemma generalizes the Neyman-Pearson Lemma [28] to multiple functions:

Lemma 1. *Let $\boldsymbol{X}$, $\boldsymbol{Y}$ be two random variables with probability densities $Pr(\boldsymbol{X}=\boldsymbol{Z})$ and $Pr(\boldsymbol{Y}=\boldsymbol{Z})$, where $\boldsymbol{Z} \in \Phi$. Let $g_1, g_2, \cdots, g_\gamma : \Phi \to \{0,1\}$ be γ random or deterministic functions. Let η be an integer such that: $\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z}) \le \eta, \forall \boldsymbol{Z} \in \Phi$, where $g_l(1|\boldsymbol{Z})$ denotes the probability that $g_l(\boldsymbol{Z})=1$. Similarly, we use $g_l(0|\boldsymbol{Z})$ to denote the probability that $g_l(\boldsymbol{Z})=0$. Then, we have the following:*

(1) If $\Phi' \subseteq \{\boldsymbol{Z} \in \Phi : g_1(1|\boldsymbol{Z}) = g_2(1|\boldsymbol{Z}) = \cdots = g_\gamma(1|\boldsymbol{Z}) = 0\}$, $O_1 = \{\boldsymbol{Z} \in \Phi \setminus \Phi' : Pr(\boldsymbol{Y}=\boldsymbol{Z}) < \rho \cdot Pr(\boldsymbol{X}=\boldsymbol{Z})\}$ and $O_2 = \{\boldsymbol{Z} \in \Phi \setminus \Phi' : Pr(\boldsymbol{Y}=\boldsymbol{Z}) = \rho \cdot Pr(\boldsymbol{X}=\boldsymbol{Z})\}$ for some $\rho > 0$. Assuming we have $O_3 \subseteq O_2$, and $O = O_1 \cup O_3$. Then, if we have $\frac{\sum_{l=1}^{\gamma} Pr(g_l(\boldsymbol{X})=1)}{\eta} \ge Pr(\boldsymbol{X} \in O)$, then $\frac{\sum_{l=1}^{\gamma} Pr(g_l(\boldsymbol{Y})=1)}{\eta} \ge Pr(\boldsymbol{Y} \in O)$.

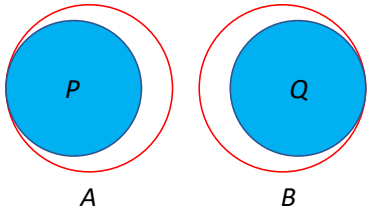

Figure 14: Illustration of subset P, Q, A, and B.

(2) If $\Phi' \subseteq \{\boldsymbol{Z} \in \Phi : g_1(1|\boldsymbol{Z}) = g_2(1|\boldsymbol{Z}) = \cdots = g_\gamma(1|\boldsymbol{Z}) = 0\}$, $O_1 = \{\boldsymbol{Z} \in \Phi : Pr(\boldsymbol{Y}=\boldsymbol{Z}) > \rho \cdot Pr(\boldsymbol{X}=\boldsymbol{Z})\}$ and $O_2 = \{\boldsymbol{Z} \in \Phi : Pr(\boldsymbol{Y}=\boldsymbol{Z}) = \rho \cdot Pr(\boldsymbol{X}=\boldsymbol{Z})\}$ for some $\rho > 0$. Assuming we have $O_3 \subseteq O_2$, and $O = O_1 \cup O_3$. Then, if we have $\frac{\sum_{l=1}^{\gamma} Pr(g_l(\boldsymbol{X})=1)}{\eta} \le Pr(\boldsymbol{X} \in O)$, then $\frac{\sum_{l=1}^{\gamma} Pr(g_l(\boldsymbol{Y})=1)}{\eta} \le Pr(\boldsymbol{Y} \in O)$.

Proof. We first prove part (1). For convenience, we denote the complement of O as O^c. Then, we have the following:

$$\frac{\sum_{l=1}^{\gamma} \Pr(g_l(\boldsymbol{Y})=1)}{\eta} - \Pr(\boldsymbol{Y} \in O) \tag{12}$$

$$= \int_{\Phi} \frac{\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z})}{\eta} \cdot \Pr(\boldsymbol{Y}=\boldsymbol{Z}) d\boldsymbol{Z} - \int_{O} \Pr(\boldsymbol{Y}=\boldsymbol{Z}) d\boldsymbol{Z} \tag{13}$$

$$= \int_{O^c} \frac{\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z})}{\eta} \cdot \Pr(\boldsymbol{Y}=\boldsymbol{Z}) d\boldsymbol{Z} + \int_{O} \frac{\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z})}{\eta} \cdot \Pr(\boldsymbol{Y}=\boldsymbol{Z}) d\boldsymbol{Z} - \int_{O} \Pr(\boldsymbol{Y}=\boldsymbol{Z}) d\boldsymbol{Z} \tag{14}$$

$$= \int_{O^c} \frac{\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z})}{\eta} \cdot \Pr(\boldsymbol{Y}=\boldsymbol{Z}) d\boldsymbol{Z} - \int_{O} (1 - \frac{\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z})}{\eta}) \cdot \Pr(\boldsymbol{Y}=\boldsymbol{Z}) d\boldsymbol{Z} \tag{15}$$

$$= \int_{O^c \setminus \Phi'} \frac{\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z})}{\eta} \cdot \Pr(\boldsymbol{Y}=\boldsymbol{Z}) d\boldsymbol{Z} + \int_{\Phi'} \frac{\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z})}{\eta} \cdot \Pr(\boldsymbol{Y}=\boldsymbol{Z}) d\boldsymbol{Z} - \int_{O} (1 - \frac{\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z})}{\eta}) \cdot \Pr(\boldsymbol{Y}=\boldsymbol{Z}) d\boldsymbol{Z} \tag{16}$$

$$\ge \rho \cdot [\int_{O^c \setminus \Phi'} \frac{\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z})}{\eta} \cdot \Pr(\boldsymbol{X}=\boldsymbol{Z}) d\boldsymbol{Z} + \int_{\Phi'} \frac{\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z})}{\eta} \cdot \Pr(\boldsymbol{X}=\boldsymbol{Z}) d\boldsymbol{Z} - \int_{O} (1 - \frac{\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z})}{\eta}) \cdot \Pr(\boldsymbol{X}=\boldsymbol{Z}) d\boldsymbol{Z}] \tag{17}$$

$$= \rho \cdot [\int_{O^c} \frac{\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z})}{\eta} \cdot \Pr(\boldsymbol{X}=\boldsymbol{Z}) d\boldsymbol{Z} + \int_{O} \frac{\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z})}{\eta} \cdot \Pr(\boldsymbol{X}=\boldsymbol{Z}) d\boldsymbol{Z} - \int_{O} \Pr(\boldsymbol{X}=\boldsymbol{Z}) d\boldsymbol{Z}] \tag{18}$$

$$= \rho \cdot [\int_{\Phi} \frac{\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z})}{\eta} \cdot \Pr(\boldsymbol{X}=\boldsymbol{Z}) d\boldsymbol{Z} - \int_{O} \Pr(\boldsymbol{X}=\boldsymbol{Z}) d\boldsymbol{Z}] \tag{19}$$

$$= \rho \cdot [\frac{\sum_{l=1}^{\gamma} \Pr(g_l(\boldsymbol{X})=1)}{\eta} - \Pr(\boldsymbol{X} \in O)] \tag{20}$$

$$\ge 0. \tag{21}$$

We have Equation (17) from (16) due to the fact that $\forall \boldsymbol{Z} \in O^c \setminus \Phi', \Pr(\boldsymbol{Y}=\boldsymbol{Z} \ge \rho \cdot \Pr(\boldsymbol{X}=\boldsymbol{Z}), \forall \boldsymbol{Z} \in \Phi', g_1(1|\boldsymbol{Z}) =$

$g_2(1|\boldsymbol{Z}) = \cdots = g_\gamma(1|\boldsymbol{Z}) = 0$, $1 - \frac{\sum_{l=1}^{\gamma} g_l(1|\boldsymbol{Z})}{\eta} \geq 0$, and $\Pr(\boldsymbol{Y} = \boldsymbol{Z}) \leq \rho \cdot \Pr(\boldsymbol{X} = \boldsymbol{Z}), \forall \boldsymbol{Z} \in O$. Similarly, we can prove the part (2). We omit the details for simplicity. □

Given a user u and Φ, we define the following subsets:

$$P = \{\boldsymbol{Z} \in \Phi | \boldsymbol{Z} \prec \boldsymbol{M}, u \vdash \boldsymbol{Z}\}, Q = \{\boldsymbol{Z} \in \Phi | \boldsymbol{Z} \prec \boldsymbol{M}, u \nvdash \boldsymbol{Z}\} \quad (22)$$

$$A = \{\boldsymbol{Z} \in \Phi | \boldsymbol{Z} \prec \boldsymbol{M}', u \vdash \boldsymbol{Z}\}, B = \{\boldsymbol{Z} \in \Phi | \boldsymbol{Z} \prec \boldsymbol{M}', u \nvdash \boldsymbol{Z}\} \quad (23)$$

It is easy to verify that $P \cap Q = \emptyset$ and $A \cap B = \emptyset$. Moreover, since M' contains rating scores for all the user in $\boldsymbol{M}$, we have $P \subseteq A$ and $Q \subseteq B$. Figure 14 shows an illustration of these four subsets. Based on our sampling without replacement, we have the following probability mass functions for $\boldsymbol{X}$ and $\boldsymbol{Y}$ in Φ:

$$\Pr(\boldsymbol{X} = \boldsymbol{Z}) = \begin{cases} \frac{1}{\binom{n}{s}}, & \text{if } \boldsymbol{Z} \in P \cup Q, \\ 0, & \text{otherwise.} \end{cases} \quad (24)$$

$$\Pr(\boldsymbol{Y} = \boldsymbol{Z}) = \begin{cases} \frac{1}{\binom{n'}{s}}, & \text{if } \boldsymbol{Z} \in A \cup B, \\ 0, & \text{otherwise.} \end{cases} \quad (25)$$

Given the probability mass functions, we have the following probabilities:

$$\Pr(\boldsymbol{X} \in P) = 1 - \binom{n-1}{s} \cdot \frac{1}{\binom{n}{s}} = \frac{s}{n}, \quad \Pr(\boldsymbol{X} \in A) = \frac{s}{n}, \quad (26)$$

$$\Pr(\boldsymbol{Y} \in P) = \frac{s}{n} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}}, \quad \Pr(\boldsymbol{Y} \in A) = \frac{s}{n'}. \quad (27)$$

We have $\Pr(\boldsymbol{X} \notin P) = \binom{n-1}{s} \cdot \frac{1}{\binom{n}{s}}$ because there are overall $\binom{n-1}{s}$ possible submatrices if the user u is not sampled from $\boldsymbol{M}$. Then, we can compute $\Pr(\boldsymbol{X} \in P)$ based on the fact that $\Pr(\boldsymbol{X} \in P) + \Pr(\boldsymbol{X} \notin P) = 1$. We obtain $\Pr(\boldsymbol{X} \in A)$ based on the fact that $P \subseteq A$ and $A \cap B = \emptyset$. Similarly, we can compute the probability of $\boldsymbol{Y}$ in these subsets.

We will leverage the law of contraposition to prove our theorem. Suppose we have a statement: $U \longrightarrow V$, then, its contraposition is $\neg V \longrightarrow \neg U$. The law of contraposition tells us that a statement is true if and only if its contraposition is true. We define the following two predicates:

$$U : \underline{p^*_{\mu_r}} > \min(\min_{c=1}^{N-r+1} \frac{N' \cdot (\overline{p}^*_{\mathcal{H}_c} + \sigma)}{c}, \overline{p}^*_{\nu_1} + \sigma) \quad (28)$$
$$\text{and } 1 \leq r \leq \min(k, N),$$

$$V : |I_u \cap \mathcal{T}(\boldsymbol{M}', u)| \geq r, \quad (29)$$

where $\sigma = \frac{s}{n'} \cdot \frac{\binom{n'}{s}}{\binom{n}{s}} - \frac{s}{n}$.

Deriving the necessary condition: We assume $\neg V$ is true, i.e., $|I_u \cap \mathcal{T}(\boldsymbol{M}', u)| < r$. If $r = 0$ or $r > \min(k, N)$, then, $\neg U$ is true. Next, we consider $1 \leq r \leq \min(k, N)$ and we will show $p^*_{\mu_r} \leq \min(\min_{c=1}^{N-r+1} \frac{N' \cdot (\overline{p}^*_{\mathcal{H}_c} + \sigma)}{c}, \overline{p}^*_{\nu_1} + \sigma)$ is true. If $|I_u \cap \mathcal{T}(\boldsymbol{M}', u)| < r$, then, there exist at least $k - r + 1$ items in I_u that are not recommended to the user u by our ensemble recommender system when taking $\boldsymbol{M}'$ as input. In other words, there exist at least $N - r + 1$ items in $I \setminus I_u$ appears in the recommended item set $\mathcal{T}(\boldsymbol{M}', u)$. For simplicity, we use $\mathcal{D}_r$ and $\mathcal{V}_r$ to denote subsets of $k - r + 1$ items in I_u and $N - r + 1$ items in $I \setminus I_u$, respectively. Formally, we have the following:

$$\exists \mathcal{D}_r, \mathcal{V}_r, \text{ s.t. } \mathcal{D}_r \cap \mathcal{T}(\boldsymbol{M}', u) = \emptyset, \mathcal{V}_r \subseteq \mathcal{T}(\boldsymbol{M}', u), \quad (30)$$

The above equation means that the poisoned item probabilities of the items in $\mathcal{V}_r$ are no smaller than the poisoned item probabilities of the items in $\mathcal{D}_r$. In other words, we have the following for $\mathcal{D}_r$ and $\mathcal{V}_r$:

$$\max_{i \in \mathcal{D}_r} \Pr(i \in \mathcal{A}(\boldsymbol{Y}, u)) \leq \min_{j \in \mathcal{V}_r} \Pr(j \in \mathcal{A}(\boldsymbol{Y}, u)). \quad (31)$$

Moreover, since there exist $\mathcal{D}_r$ and $\mathcal{V}_r$, we have the following necessary condition if $|I_u \cap \mathcal{T}(\boldsymbol{M}', u)| < r$ and $1 \leq r \leq \min(k, N)$:

$$\min_{\mathcal{D}_r} \max_{i \in \mathcal{D}_r} \Pr(i \in \mathcal{A}(\boldsymbol{Y}, u)) \leq \max_{\mathcal{V}_r} \min_{j \in \mathcal{V}_r} \Pr(j \in \mathcal{A}(\boldsymbol{Y}, u)). \quad (32)$$

Next, we will derive the lower bound of left-hand side and the upper bound of the right-hand side for the above equation.

Deriving a lower bound of $\min_{\mathcal{D}_r} \max_{i \in \mathcal{D}_r} \Pr(i \in \mathcal{A}(\boldsymbol{Y}, u))$: For $\forall i \in \mathcal{D}_r$, based on Equation (3), we have the following:

$$\underline{p^*_i} \triangleq \frac{\lfloor p_i \cdot \binom{n}{s} \rfloor}{\binom{n}{s}} \leq \underline{p_i} \leq \Pr(i \in \mathcal{A}(\boldsymbol{X}, u)) \quad (33)$$

Based on the definition of $\boldsymbol{X}$, we have:

$$\Pr(i \in \mathcal{A}(\boldsymbol{X}, u)) \geq \underline{p^*_i}. \quad (34)$$

For $\forall i \in \mathcal{D}_r$, we define the function $g_i(\boldsymbol{Z}) = \mathbb{I}(i \in \mathcal{A}(\boldsymbol{Z}, u))$, where $\mathbb{I}$ is an indicator function. Specifically, we have $g_i(\boldsymbol{Z}) = 0$ for $\forall \boldsymbol{Z} \in B$ based on the definition of B in Equation (23), which will be used when we leverage Lemma 1 to derive the lower bound of $\Pr(g_i(\boldsymbol{Y}) = 1)$. We have $\Pr(g_i(\boldsymbol{X}) = 1) \geq \underline{p^*_i}$ based on Equation (34) and the definition of g_i. For $\forall i \in \mathcal{D}_r$, we can find $C_i \subseteq P$ such that we have the following:

$$\Pr(\boldsymbol{X} \in C_i) = \underline{p^*_i}. \quad (35)$$

Note that we can find such a subset because $\underline{p^*_i}$ is an integer multiple of $\frac{1}{\binom{n}{s}}$. Then, we have the following:

$$\Pr(g_i(\boldsymbol{X}) = 1) \geq \Pr(\boldsymbol{X} \in C_i). \quad (36)$$

For simplicity, we define the following quantity:

$$\tau = \binom{n}{s} / \binom{n'}{s}. \quad (37)$$

Next, we will apply Lemma 1 to obtain the lower bound of $\Pr(g_i(\boldsymbol{Y}) = 1)$. In particular, we let $\Phi' = B$ since we have $g_i(\boldsymbol{Z}) = 0$ for $\forall \boldsymbol{Z} \in B$. Note that we omit Q since we have $Q \subseteq B$. Then, we have $A = \Phi \setminus \Phi'$. Based on Equation (24) - (25), we have $\Pr(\boldsymbol{Y} = \boldsymbol{Z}) = \tau \cdot \Pr(\boldsymbol{X} = \boldsymbol{Z})$ if $\boldsymbol{Z} \in P$ and $\Pr(\boldsymbol{Y} =$

$\boldsymbol{Z}) > \tau \cdot \Pr(\boldsymbol{X} = \boldsymbol{Z})$ if $\boldsymbol{Z} \in A \setminus P$. We let $O_1 = \emptyset$ since there is no subset that satisfies $\Pr(\boldsymbol{Y} = \boldsymbol{Z}) < \tau \cdot \Pr(\boldsymbol{X} = \boldsymbol{Z})$. Furthermore, we let $O_2 = P$, $O_3 = C_i \subseteq O_2$, $\eta = 1$, and $\gamma = 1$. Finally, we let $O = O_1 \cup O_3 = C_i$. We can apply Lemma 1 based on the condition in Equation (36) and we have the following:

$$\Pr(g_i(\boldsymbol{Y}) = 1) \geq \Pr(\boldsymbol{Y} \in C_i). \tag{38}$$

Based on the definition of g_i, we have the following:

$$\begin{aligned} &\Pr(i \in \mathcal{A}(\boldsymbol{Y}, u)) && (39)\\ =&\Pr(g_i(\boldsymbol{Y}) = 1) && (40)\\ \geq&\Pr(\boldsymbol{Y} \in C_i) && (41)\\ =&\Pr(\boldsymbol{X} \in C_i) \cdot \tau && (42)\\ =&\underline{p_i^*} \cdot \tau. && (43) \end{aligned}$$

For simplicity, we denote $\mathcal{D}_r = \{d'_1, d'_2, \cdots, d'_z\}$, where $z = k - r + 1$. Without loss of generality, we assume $\underline{p_{d'_1}} \geq \cdots \geq \underline{p_{d'_z}}$. We have the following:

$$\max_{i \in \mathcal{D}_r} \Pr(i \in \mathcal{A}(\boldsymbol{Y}, u)) \geq \max_{i \in \mathcal{D}_r} \underline{p_i^*} \cdot \tau = \underline{p_{d'_1}^*} \cdot \tau. \tag{44}$$

Then, we have the following:

$$\min_{\mathcal{D}_r} \max_{i \in \mathcal{D}_r} \Pr(i \in \mathcal{A}(\boldsymbol{Y}, u)) = \min_{\mathcal{D}_r} \underline{p_{d'_1}^*} \cdot \tau \tag{45}$$

Therefore, when $\mathcal{D}_r = \{\mu_r, \mu_{r+1}, \cdots, \mu_k\}$, $\underline{p_{d'_1}^*} \cdot \tau$ reaches the minimal value which is $\underline{p_{\mu_r}^*} \cdot \tau$. In other words, we have the following:

$$\min_{\mathcal{D}_r} \max_{i \in \mathcal{D}_r} \Pr(i \in \mathcal{A}(\boldsymbol{Y}, u)) \geq \underline{p_{\mu_r}^*} \cdot \tau. \tag{46}$$

Deriving an upper bound of $\max_{\mathcal{V}'_r} \min_{j \in \mathcal{V}'_r} \mathbf{Pr}(j \in \mathcal{A}(\boldsymbol{Y}, u))$: For $\forall j \in \mathcal{V}_r$, given Equation (3), we have the following:

$$\overline{p}_j^* = \frac{\lceil \overline{p}_j \cdot \binom{n}{s} \rceil}{\binom{n}{s}} \geq \overline{p}_j \geq \Pr(j \in \mathcal{A}(\boldsymbol{X}, u)). \tag{47}$$

Suppose we have $\mathcal{V}'_r = \{v'_1, v'_2, \cdots, v'_w\}$, where $w = N - r + 1$. Without loss of generality, we assume the following:

$$\overline{p}_{v'_1} \leq \overline{p}_{v'_2} \leq \cdots \leq \overline{p}_{v'_w}. \tag{48}$$

We first derive an upper bound of $\min_{j \in \mathcal{V}'_r} \Pr(j \in \mathcal{A}(\boldsymbol{Y}, u))$. Given an arbitrary item $j \in \mathcal{V}'_r$, we have the following inequality based on Equation (47) and our definition of $\boldsymbol{X}$:

$$\Pr(j \in \mathcal{A}(\boldsymbol{X}, u)) \leq \overline{p}_j^*. \tag{49}$$

Given an item $j \in \mathcal{V}'_r$, we define the function $g_j(\boldsymbol{Z}) = \mathbb{I}(j \in \mathcal{A}(\boldsymbol{Z}, u))$. We have $\Pr(g_j(\boldsymbol{X}) = 1) \leq \overline{p}_j^*$ based on Equation (49) and the definition of g_j. Then, we can leverage Lemma 1 to derive an upper bound for $\Pr(g_j(\boldsymbol{X}) = 1)$. In particular, we can find $C'_j \subseteq P$ such that we have the following:

$$\Pr(\boldsymbol{X} \in C'_j) = \overline{p}_j^*. \tag{50}$$

We let $C_j = C'_j \cup (A \setminus P)$. Since we have $\Pr(\boldsymbol{X} \in A \setminus P) = \Pr(\boldsymbol{X} \in A) - \Pr(\boldsymbol{X} \in P) = 0$, we have the following:

$$\Pr(\boldsymbol{X} \in C_j) = \Pr(\boldsymbol{X} \in C'_j) + \Pr(\boldsymbol{X} \in A \setminus P) = \overline{p}_j^*. \tag{51}$$

Based on $\Pr(g_j(\boldsymbol{X}) = 1) \leq \overline{p}_j^*$, we have the following:

$$\Pr(g_j(\boldsymbol{X}) = 1) \leq \Pr(\boldsymbol{X} \in C_j). \tag{52}$$

Next, we will apply Lemma 1 to obtain the upper bound of $\Pr(g_j(Y) = 1)$. In particular, we let $\Phi' = B$ since we have $g_j(\boldsymbol{Z}) = 0$ for $\forall \boldsymbol{Z} \in B$. Then, we have $A = \Phi \setminus \Phi'$. Based on Equation (24) - (25), we have $\Pr(\boldsymbol{Y} = \boldsymbol{Z}) = \tau \cdot \Pr(\boldsymbol{X} = \boldsymbol{Z})$ if $\boldsymbol{Z} \in P$ and $\Pr(\boldsymbol{Y} = \boldsymbol{Z}) > \tau \cdot \Pr(\boldsymbol{X} = \boldsymbol{Z})$ if $\boldsymbol{Z} \in A \setminus P$. We let $O_1 = A \setminus P$, $O_2 = P$, $O_3 = C'_j \subseteq O_2$, $\eta = 1$, and $\gamma = 1$. Finally, we let $O = O_1 \cup O_3 = C_j$. We can apply Lemma 1 based on the condition in Equation (52) and we have the following:

$$\begin{aligned} &\Pr(j \in \mathcal{A}(\boldsymbol{Y}, u)) && (53)\\ =&\Pr(g_j(\boldsymbol{Y}) = 1) && (54)\\ \leq&\Pr(\boldsymbol{Y} \in C_j) && (55)\\ =&\Pr(\boldsymbol{Y} \in C'_j) + \Pr(\boldsymbol{Y} \in A \setminus P) && (56)\\ =&\Pr(\boldsymbol{X} \in C'_j) \cdot \tau + \Pr(\boldsymbol{Y} \in A) - \Pr(\boldsymbol{Y} \in P) && (57)\\ =&\overline{p}_j^* \cdot \frac{\binom{n}{s}}{\binom{n'}{s}} + \frac{s}{n'} - \frac{s}{n} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}}. && (58) \end{aligned}$$

Given Equation (48), we have the following:

$$\min_{j \in \mathcal{V}'_r} \Pr(j \in \mathcal{A}(\boldsymbol{Y}, u)) \leq \overline{p}_{v'_1}^* \cdot \frac{\binom{n}{s}}{\binom{n'}{s}} + \frac{s}{n'} - \frac{s}{n} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}}. \tag{59}$$

Therefore, when $\mathcal{V}'_r$ contains the set of items in $I \setminus I_u$ that have the largest probability bounds, which we denote as $\mathcal{V}'_r = \{v_1, v_2, \cdots, v_w\}$ where $\overline{p}_{v_1} \leq \overline{p}_{v_2} \leq \cdots \leq \overline{p}_v$, the upper bound of $\max_{\mathcal{V}'_r} \min_{j \in \mathcal{V}'_r} \Pr(j \in \mathcal{A}(\boldsymbol{Y}, u))$ reaches the maximum value. Formally, we have the following:

$$\max_{\mathcal{V}'_r} \min_{j \in \mathcal{V}'_r} \Pr(j \in \mathcal{A}(\boldsymbol{Y}, u)) \leq \overline{p}_{v_1}^* \cdot \frac{\binom{n}{s}}{\binom{n'}{s}} + \frac{s}{n'} - \frac{s}{n} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}}. \tag{60}$$

Next, we will derive another upper bound for $\Pr(g_j(Y) = 1)$ via jointly considering multiple items. In particular, we use $\mathcal{H}_c$ to denote an arbitrary subset of $\mathcal{V}'_r$ that contains c items, i.e., $\mathcal{H}_c \subseteq \mathcal{V}'_r$. We denote $\overline{p}_{\mathcal{H}_c} = \sum_{j \in \mathcal{H}_c} \overline{p}_j$, i.e., the summation of probability upper bounds for the items in $\mathcal{H}_c$. Then, for each item $j \in \mathcal{H}_c$, we define $g_j(\boldsymbol{Z}) = \mathbb{I}(j \in \mathcal{A}(\boldsymbol{Z}, u))$. Given a positive integer η, we define the following quantity:

$$\overline{p}_{\mathcal{H}_c}^* = \frac{\lceil (\overline{p}_{\mathcal{H}_c} / \eta) \cdot \binom{n}{s} \rceil}{\binom{n}{s}}. \tag{61}$$

We can find $C'_{\mathcal{H}_c} \subseteq P$ such that we have the following:

$$\Pr(\boldsymbol{X} \in C'_{\mathcal{H}_c}) = \overline{p}_{\mathcal{H}_c}^*. \tag{62}$$

Then, we define $C_{\mathcal{H}_c} = C'_{\mathcal{H}_c} \cup (A \setminus P)$ and we have the following:

$$\Pr(\boldsymbol{X} \in C_{\mathcal{H}_c}) = \Pr(\boldsymbol{X} \in C'_{\mathcal{H}_c}) + \Pr(\boldsymbol{X} \in A \setminus P) = \overline{p}^*_{\mathcal{H}_c}. \tag{63}$$

Based on the definition of $g_j(\boldsymbol{Z})$, we have the following:

$$\frac{\sum_{j \in \mathcal{H}_c} \Pr(g_j(X) = 1)}{\eta} \leq \frac{\sum_{j \in \mathcal{H}_c} \overline{p}_j}{\eta} \leq \overline{p}^*_{\mathcal{H}_c} = \Pr(\boldsymbol{X} \in C_{\mathcal{H}_c}). \tag{64}$$

Next, we will leverage Lemma 1 to derive an upper bound for $\sum_{j \in \mathcal{H}_c} \Pr(g_j(\boldsymbol{Y}) = 1)$. Given a rating-score matrix $\boldsymbol{Z}$ as input, the recommender system algorithm $\mathcal{A}$ recommends N' items to a user. Therefore, we have $\sum_{j \in \mathcal{H}_c} \mathbb{I}(j \in \mathcal{A}(\boldsymbol{Z}, u)) \leq N'$, i.e., $\sum_{j \in \mathcal{H}_c} g_j(\boldsymbol{Z}) \leq N'$. Based on this, we let $\eta = N'$. Since there are c items in $\mathcal{H}_c$, we let $\gamma = c$. We let $\Phi' = B$ since we have $g_j(\boldsymbol{Z}) = 0$ for $\forall \boldsymbol{Z} \in B$. Then, we have $A = \Phi \setminus \Phi'$. Based on Equation (24) - (25), we have $\Pr(\boldsymbol{Y} = \boldsymbol{Z}) = \tau \cdot \Pr(\boldsymbol{X} = \boldsymbol{Z})$ if $\boldsymbol{Z} \in P$ and $\Pr(\boldsymbol{Y} = \boldsymbol{Z}) > \tau \cdot \Pr(\boldsymbol{X} = \boldsymbol{Z})$ if $\boldsymbol{Z} \in A \setminus P$. We let $O_1 = A \setminus P$, $O_2 = P$, and $O_3 = C'_{\mathcal{H}_c} \subseteq O_2$. Finally, we let $O = O_1 \cup O_3 = C_{\mathcal{H}_c}$. We can apply Lemma 1 based on the condition in Equation (64) and we have the following:

$$\sum_{j \in \mathcal{H}_c} \Pr(j \in \mathcal{A}(\boldsymbol{Y}, u)) \tag{65}$$

$$= \sum_{j \in \mathcal{H}_c} \Pr(g_j(\boldsymbol{Y}) = 1) \tag{66}$$

$$\leq N' \cdot \Pr(\boldsymbol{Y} \in C_{\mathcal{H}_c}) \tag{67}$$

$$= N' \cdot (\Pr(\boldsymbol{Y} \in C'_{\mathcal{H}_c}) + \Pr(\boldsymbol{Y} \in A \setminus P)) \tag{68}$$

$$= N' \cdot (\Pr(\boldsymbol{X} \in C'_{\mathcal{H}_c}) \cdot \tau + \Pr(\boldsymbol{Y} \in A) - \Pr(\boldsymbol{Y} \in P)) \tag{69}$$

$$= N' \cdot (\overline{p}^*_{\mathcal{H}_c} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}} + (\frac{s}{n'} - \frac{s}{n} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}})). \tag{70}$$

Then, we have the following:

$$\min_{j \in \mathcal{V}_r} \Pr(j \in \mathcal{A}(\boldsymbol{Y}, u)) \tag{71}$$

$$\leq \min_{j \in \mathcal{H}_c} \Pr(j \in \mathcal{A}(\boldsymbol{Y}, u)) \tag{72}$$

$$\leq \frac{\sum_{j \in \mathcal{H}_c} \Pr(j \in \mathcal{A}(\boldsymbol{Y}, u))}{c} \tag{73}$$

$$\leq N' \cdot (\overline{p}^*_{\mathcal{H}_c} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}} + (\frac{s}{n'} - \frac{s}{n} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}}))/c. \tag{74}$$

We have Equation (72) from (71) because $\mathcal{H}_c \subseteq \mathcal{V}_r$ and Equation (73) from (72) because the smallest value is no larger than the average value in a set. We note that the upper bound of $\min_{j \in \mathcal{V}_r} \Pr(j \in \mathcal{A}(\boldsymbol{Y}, u))$ is non-decreasing as $\overline{p}_{\mathcal{H}_c}$ increases. Based on Equation (48), the upper bounds reaches the minimal value when $\mathcal{H}_c = \{v'_1, v'_2, \cdots, v'_c\}$. Taking all possible c into consideration, we have the following:

$$\min_{j \in \mathcal{V}_r} \Pr(j \in \mathcal{A}(\boldsymbol{Y}, u)) \tag{75}$$

$$\leq \min_{c=1}^{N-r+1} N' \cdot (\overline{p}^*_{\mathcal{H}_c} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}} + (\frac{s}{n'} - \frac{s}{n} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}}))/c,$$

where $\mathcal{H}_c = \{v'_1, v'_2, \cdots, v'_c\}$. Similarly, the upper bound of $\max_{\mathcal{V}_r} \min_{j \in \mathcal{V}_r} \Pr(j \in \mathcal{A}(\boldsymbol{Y}, u))$ reaches the maximum value when $\mathcal{V}_r$ contains the $N - r + 1$ items among all items in $I \setminus I_u$ that have the largest probability upper bounds, which we denote as $\mathcal{V}_r = \{v_1, v_2, \cdots, v_w\}$, where $\overline{p}_{v_1} \leq \overline{p}_{v_2} \leq \cdots \leq \overline{p}_{v_w}$ and $w = N - r + 1$. Formally, we have the following:

$$\max_{\mathcal{V}_r} \min_{j \in \mathcal{V}_r} \Pr(j \in \mathcal{A}(\boldsymbol{Y}, u)) \tag{76}$$

$$\leq \min_{c=1}^{N-r+1} N' \cdot (\overline{p}^*_{\mathcal{H}_c} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}} + (\frac{s}{n'} - \frac{s}{n} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}}))/c, \tag{77}$$

where $\mathcal{H}_c = \{v_1, v_2, \cdots, v_c\}$. Since we have Equation (32) when $\neg U$ is true and $1 \leq r \leq \min(k, N)$, we have the following:

$$\underline{p^*_{\mu_r}} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}} \leq \min(\min_{c=1}^{N-r+1} \frac{N' \cdot (\overline{p}^*_{\mathcal{H}_c} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}} + (\frac{s}{n'} - \frac{s}{n} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}}))}{c}, \overline{p}^*_{v_1} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}} + \frac{s}{n'} - \frac{s}{n} \cdot \frac{\binom{n}{s}}{\binom{n'}{s}}). \tag{78}$$

where $\mathcal{V}_r = \{v_1, v_2, \cdots, v_{N-r+1}\}$ and $\mathcal{H}_c = \{v_1, v_2, \cdots, v_c\}$. The Equation (78) is equivalent to the following:

$$\underline{p^*_{\mu_r}} \leq \min(\min_{c=1}^{N-r+1} \frac{N' \cdot (\overline{p}^*_{\mathcal{H}_c} + \sigma)}{c}, \overline{p}^*_{v_1} + \sigma), \tag{79}$$

where $\sigma = \frac{s}{n'} \cdot \frac{\binom{n'}{s}}{\binom{n}{s}} - \frac{s}{n}$.

Applying the law of contraposition: We leverage the law of contraposition and we have the following: if we have $1 \leq r \leq \min(k, N)$ and the following:

$$\underline{p^*_{\mu_r}} > \min(\min_{c=1}^{N-r+1} \frac{N' \cdot (\overline{p}^*_{\mathcal{H}_c} + \sigma)}{c}, \overline{p}^*_{v_1} + \sigma), \tag{80}$$

where $\sigma = \frac{s}{n'} \cdot \frac{\binom{n'}{s}}{\binom{n}{s}} - \frac{s}{n}$. Then, we have $|I_u \cap \mathcal{T}(\boldsymbol{M}', u)| \geq r$. The Equation (80) is satisfied for $\forall \boldsymbol{M}' \in \mathcal{L}(\boldsymbol{M}, e)$. We can find the maximum value of r, where $1 \leq r \leq \min(k, N)$, that satisfies the Equation (80), which is essentially the optimization problem in the Equation (6). We reach the conclusion.

B Proof of Theorem 3.2

Based on Equation (7) - (8) and Boole's inequality in probability theory, we have the following probability:

$$\Pr((p_i \geq \underline{p_i}, \forall i \in I_u) \wedge (p_j \leq \overline{p}_j, \forall j \in I \setminus I_u)) \geq 1 - \alpha_u, \tag{81}$$

where $R \wedge S$ is true if and only if R is true and S is true. Note that there is no randomness in our optimization problem in Equation (6). Therefore, the probability that our Algorithm 2 computes an incorrect r_u for the user u is at most α_u. Recall that we set $\alpha_u = \frac{\alpha}{n}$ in our Algorithm 2. Based on the Boole's inequality, we know the probability that our Algorithm 2 computes an incorrect r_u for at least one user among all users in $\mathcal{U}$ is at most α.