Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Differentially Private Matrix Factorization*

Jingyu Hua, Chang Xia, Sheng Zhong
State Key Laboratory for Novel Software Technology,
Department of Computer Science and Technology, Nanjing University, China
huajingyu@nju.edu.cn, changxia656569@gmail.com, zhongsheng@nju.edu.cn

Abstract

Matrix factorization (MF) is a prevailing collaborative filtering method for building recommender systems. It requires users to upload their personal preferences to the recommender for performing MF, which raises serious privacy concerns. This paper proposes a differentially private MF mechanism that can prevent an untrusted recommender from learning any users' ratings or profiles. Our design decouples computations upon users' private data from the recommender to users, and makes the recommender aggregate local results in a privacy-preserving way. It uses the objective perturbation to make sure that the final item profiles satisfy differential privacy and solves the challenge to decompose the noise component for objective perturbation into small pieces that can be determined locally and independently by users. We also propose a third-party based mechanism to reduce noises added in each iteration and adapt our online algorithm to the dynamic setting that allows users to leave and join. The experiments show that our proposal is efficient and introduces acceptable side effects on the precision of results.

1 Introduction

Recommendation systems (RSes), which provide users personalized recommendations of contents and services, have been demonstrated to be extremely valuable for boosting websites' revenue and improving user experience. However, everything has two sides, so do these systems. To offer useful recommendations, a RS usually requires users to supply their personal preferences for various items, which raise serious privacy concerns.

Many early studies [Aïmeur *et al.*, 2008; McSherry and Mironov, 2009; Calandrino *et al.*, 2011] have shown that even those user preferences typically considered insensitive (e.g., ratings of movies and TV viewing history) may be exploited to infer a user's health condition, political inclinations, and even his real identity. What's worse, we have no way to guarantee that recommenders do not abuse user data for financial benefits [Canny, 2002; Nikolaenko *et al.*, 2013]. So, an appealing question is raised: ***is it possible to build a recommendation system without the recommender learning the users' ratings of items?***

There are many collaborative filtering techniques available to build a recommendation system. *Matrix factorization* (abbrev. MF) [Koren *et al.*, 2009; Candès and Recht, 2009] is among the most popular and successful. It wins the famous Netflix prize competition [Koren *et al.*, 2009], and is being used in a lot of real-world recommendation systems. So, this paper aims to propose a privacy-preserving MF scheme.

Just as the name suggests, MF aims to factorize the user-item rating matrix R into the product of two smaller matrices U and V, which are considered to capture latent features of users and items, respectively. The core of this method is to utilize some machine learning algorithm, e.g., *Stochastic Gradient Descent*, to search for U and V that minimize the prediction errors on the set of known ratings. A privacy-preserving MF scheme should guarantee that the execution of the learning algorithm exposes only V to the recommender but never any information about individual ratings or even U.

Nikolaenko *et al.* [2013] first study this problem, and propose an elegant solution through a cryptographic technique known as garbled circuits. Nevertheless, garbled circuits are considered computation expensive even if they have significantly reduced time overheads by exploiting the sparsity characteristic of users' rating matrix. In this paper, we aim to propose a more lightweight solution under the constraint of *differential privacy* [Dwork *et al.*, 2006], which is a recently hot privacy model with provable privacy guarantees but fewer computation overheads. Specifically, we make the following contributions:

(1) We first consider the scenario where the recommender is trusted, which means personal ratings can be exposed to the recommender. The goal here is to ensure that the final item profile matrix V to be published satisfies ϵ-differential privacy. Our proposal utilizes *objective-perturbation technique*, which is first proposed in [Chaudhuri and Monteleoni, 2009] to implement differentially private logistic regression, to achieve this goal. We figure out the required distribution of the noise component for objective perturbation in MF.

(2) We then extend the above proposal to the scenario

*This work was supported by NSFC-61321491, NSFC-61425024, and NSFC-61300235

where the recommender is untrusted. For this purpose, we decouple the computations on private ratings from the recommender to users in each iteration of learning. The recommender is only responsible for aggregating perturbed local results in a differentially private way. We address the challenge to decompose the noise component for objective perturbation into small pieces that can be determined locally and independently by users. We also propose a third-party-based optimization to reduce the noises needed to add in each iteration for improving the precision of results.

(3) The above proposal assumes that every user remains online during the whole process MF. However, MF may last for more than one week so that it is impractical to make such assumption. Therefore, we further present an improvement to make our proposal support the more practical setting that allows users to leave and join during MF.

(4) We conduct experiments based on two datasets. The results demonstrate that our proposal is efficient enough and, moreover, the privacy protection measures bring limited side effects on the precision of learning results.

2 Related Work

There have been some work introducing differential privacy to recommender systems. Nevertheless, most of them [McSherry and Mironov, 2009; Machanavajjhala *et al.*, 2011; Park and Acharya, 2011] consider that the recommender is trusted and allowed to explicitly aggregate users' ratings. Their only goal is to guarantee that the distribution of the results published by the recommender is insensitive to any individual record and thus prevent other parties from inferring a single user's ratings. Xin and Jaakkola [2014] are the first to deal with the untrusted recommender without using cryptographic tools. Similar with our work, they separate sensitive computations that can be executed on the user side from those must be carried out on the server. However, their proposal demands the existence a group of public users who are willing to share their ratings with the recommender, which is not fulfilled in our scenario. In addition, this proposal does not satisfy differential privacy, i.e., is not provably secure.

As the core of MF is a machine learning algorithm, our work is also quite related to another area: *differentially private machine learning* [Chaudhuri and Monteleoni, 2009; Chaudhuri *et al.*, 2011]. Work in this area aims to implement differentially private versions of prevailing machine learning algorithms such as logistic regression and support vector machines. These algorithms mostly focus on the single-party setting, where all the user data has been aggregated on a singe trusted party. Their goal is to guarantee that any published model that is learned from the data satisfies differential privacy. Rajkumar and Agarwal [2012] propose an elegant differentially private stochastic gradient descent algorithm for the multi-party setting. Note that stochastic gradient descent is a key solution to MF. Unfortunately, we cannot directly apply this proposal since the scenario we consider in this paper is not completely the same with theirs although both of them involve multi-parties. The biggest difference is that the parties in their setting are organizations owning a large number of users's data, while the ones in our setting are individual users owning their own data. As we mentioned in Sec. 4.2, if we directly apply their approach, the precision of the obtain model will be extremely low. Moreover, their proposal can only guarantee a slightly weaker form of differential privacy instead of the original form.

3 Preliminaries

3.1 Matrix Factorization

This paper considers the standard setting of collaborating filtering, in which n users rate a small subset of m items. We denote by $R = [r_{ij}]_{n\times m}$ the full rating matrix, and by $\mathcal{R} \subset [n] \times [m]$ the user-item pairs that have values in R, i.e., for which ratings have been generated. Note that R is usually extremely sparse, i.e., $W = |\mathcal{R}| = \Theta(m + n) \ll m \times n$. The goal of a recommendation system is to predict the blank ratings for the user-item pairs in $[n] \times [m] \setminus \mathcal{R}$.

MF is one of the most popular methods for solving this problem. In its basic form, each user i is characterized by a profile vector $u_i \in \mathbb{R}^d$, and each item j is characterized by a profile vector $v_j \in \mathbb{R}^d$. User i's rating of item j, which is denoted by r_{ij}, is then approximated by the inner product of u_i and v_j, i.e., $u_i^T v_j$. The dimension d of these vectors are very small, say 20 to 100. The recommender computes the profile matrix $U = [u_i^T]_{i\in[n]}$ and $V = [v_j^T]_{i\in[m]}$ by performing the following *regularized least squares minimization (RLSM)*:

$$\min_{U,V} \mathcal{C}(U,V) = \frac{1}{M} \sum_{(i,j)\in\mathcal{R}} (r_{ij} - u_i^T v_j)^2 + \lambda \sum_{i\in[n]} \|u_i\|_2^2 + \mu \sum_{j\in[m]} \|v_j\|_2^2, \tag{1}$$

where both λ and μ are positives. The same as the work in Nikolaenko *et al.* [2013], we focus on the popular approach, stochastic gradient descent (abbr. SGD), to minimize Equation (1). Denote by $E(U,V)$ the squared prediction error. i.e., $E(U,V) = \sum_{(i,j)\in\mathcal{R}} (r_{ij} - u_i^T v_j)^2$. SGD iteratively learns U and V based on the following updating rules:

$$u_i(t) = u_i(t-1) - \gamma \cdot (\nabla_{u_i}(U(t-1), V(t-1)) + 2\lambda u_i(t-1)), \tag{2}$$

$$v_j(t) = v_i(t-1) - \gamma \cdot (\nabla_{v_j}(U(t-1), V(t-1)) + 2\mu v_j(t-1)), \tag{3}$$

where $\gamma > 0$ is a small gain factor and

$$\nabla_{u_i}(U,V) = -2 \sum_{j:(i,j)\in\mathcal{R}} v_j (r_{ij} - u_i^T v_j), \tag{4}$$

$$\nabla_{v_j}(U,V) = -2 \sum_{i:(i,j)\in\mathcal{R}} u_i (r_{ij} - u_i^T v_j). \tag{5}$$

The initial $U(0)$ and $V(0)$ are composed of uniformly random norm 1 rows.

3.2 Differential Privacy

Differential privacy becomes a hot privacy model recently as it provides provable privacy guarantees but introduces fewer computation overheads. It requires that the output of any computation based on a underlying database is insensitive to

the removal and the addition of a particular record. This indicates that it is privacy harmless for a record owner to include his record in the database. We formally give the definition of different privacy below [Mohammed *et al.*, 2011].

Definition 1 (ϵ-differential privacy) *A randomized algorithm Ag is differentially private if and only if any two databases $\mathbb{D}$ and $\mathbb{D}'$ contain at most one different record (i.e., $|\mathbb{D} \triangle \mathbb{D}'| \leq 1$), and for any possible anonymized output $O \in Range(Ag)$: $Pr[Ag(\mathbb{D}) = O] \leq e^{\epsilon} \times Pr[Ag(\mathbb{D}') = O]$, where the probability is taken over the randomness of Ag.*

In the above equation, ϵ is positive, and it is believed that the smaller its value is, the stronger the privacy guarantee is.

3.3 Setting

This paper considers a recommender system including two types of actors: users and the recommender. Every user $i \in [n]$ wants to preserve her ratings. The recommender is untrusted, which means he is curious about and even may misuse users' ratings for making profits.

Our objective is to design a differentially private MF scheme to allow the recommender to obtain the accurate item profile matrix, i.e., V, but neither the user profile matrix, i.e., U, nor the individual ratings. Note that we should prevent the recommender from getting U since they can derive users' ratings precisely by computing the product of U and V. As Nikolaenko *et al.* [2013] mentioned, once V is learned and published, a user i can infer her user profile u_i by solving (1) w.r.t. u_i via ridge regression. With both u_i and V, she can further predict her preferences on all the items locally.

4 Differentially Private Matrix Factorization

4.1 Scenario with Trusted Recommender

We first consider the simple centralized scenario, where the recommender is assumed to have collected the ratings of users and wants to learn and publish V satisfying ϵ-differential privacy. In this scenario, the recommender is considered to be trusted and never disclose users' privacy. Matrix U must be kept secret, otherwise the attacker can predict a specific user's ratings of all the items as long as she knows which vector in U corresponds to this user.

We apply the objective perturbation method, which is first proposed by [Chaudhuri and Monteleoni, 2009], to fulfill the above goal. Its basic idea is to achieve differential privacy by randomly perturbing the objective function instead of perturbing the output of the leaning algorithm. Specifically, in our case, the objective in (1) is perturbed as follows:

$$\begin{aligned}\min_V \widetilde{\mathcal{C}}(V) =& \frac{1}{M} \sum_{(i,j)\in\mathcal{R}} (r_{ij} - u_i^T v_j)^2 + \lambda \sum_{i\in[n]} \|u_i\|_2^2 \\ &+ \mu \sum_{j\in[m]} \|v_j\|_2^2 + \frac{1}{M} \sum_{j=1}^{m} \eta_j^T v_j,\end{aligned} \tag{6}$$

where $N = [\eta_j]_{d\times m}$ is a noise matrix for objective perturbation. Because U is kept confidential, the recommender can first perform the original RLSM in (1), and then use the obtained U as a constant profile matrix when minimize (6). That is why (6) contains only one matrix variable, i.e., V.

Theorem 1 *Let $[min_r, max_r]$ ($min_r, max_r \in \mathbb{R}$) be the range of users' rating values. If each vector $\eta_j \in N$ in (6) is independently and randomly picked from the density function $p(\eta_j) \propto e^{-\frac{\epsilon\|\eta_j\|}{2\Delta}}$, where $\Delta = max_r - min_r$, the derived V satisfies ϵ-differential privacy.*

Proof 1 *Let $D = \{r_{i_1j_1}, r_{i_2j_2}, \cdots, r_{i_{W-1}j_{W-1}}, r_{pq}\}$ and $D' = \{r_{i_1j_1}, r_{i_2j_2}, \cdots, r_{i_{W-1}j_{W-1}}, r'_{pq}\}$ be two sets of ratings differing at one record r_{pq}. Here, $\{(i_1,j_1),(i_2,j_2),\cdots,(i_{W-1},j_{W-1},(p,q))\} = \mathcal{R}$. Let N and N' be the noise matrices in (6) when training with D and D', respectively. Obviously, $\widetilde{\mathcal{C}}(V)$ is differentiable anywhere.*

Let $\bar{V}$ be the derived item profile matrix minimizes both the optimization problems, we have $\forall j \in \{1,2,\cdots,m\}$, $\nabla_{v_j}\widetilde{\mathcal{C}}(D,\bar{v_j}) = \nabla_{v_j}\widetilde{\mathcal{C}}(D',\bar{v_j}) = 0$. Thereby,

$$\eta_j - 2\sum_{i:(i,j)\in\eta} u_i(r_{ij} - u_i^T\bar{v_j}) = \eta'_j - 2\sum_{i:(i,j)\in\eta} u_i(r'_{ij} - u_i^T\bar{v_j}). \tag{7}$$

If $j \neq q$, we can derive from (7) that:

$$\eta_j = \eta'_j.$$

So, $\forall j \neq q$, $\|\eta_j\| = \|\eta'_j\|$.

If $j = q$, we can derive that:

$$\begin{aligned}\eta_j - 2u_p(r_{pq} - u_p^T\bar{v_q}) &= \eta'_j - 2u_p(r'_{pq} - u_p^T\bar{v_q}) \\ \eta_j - \eta'_j &= 2u_p(r_{pq} - r'_{pq}).\end{aligned}$$

Since $\|u_p\| \leq 1$ and $|r_{pq} - r'_{pq}| \leq \Delta$, we obtain that $\|\eta_j - \eta'_j\| \leq 2\Delta$.

Therefore, for any pair of r_{pq} and r'_{pq},

$$\begin{aligned}\frac{Prob[V=\bar{v_j}|D]}{Prob[V=\bar{v_j}|D']} &= \frac{\prod_{j\in\{1,2,\cdots,m\}} p(\eta_j)}{\prod_{j\in\{1,2,\cdots,m\}} p(\eta'_j)} \\ &= e^{-\frac{\epsilon(\sum_{j\in\{1,2,\cdots,m\}}\|\eta_j\| - \sum_{j\in\{1,2,\cdots,m\}}\|\eta'_j\|)}{2\Delta}} \\ &= e^{-\frac{\epsilon(\|\eta_q\| - \|\eta'_q\|)}{2\Delta}}.\end{aligned} \tag{8}$$

As $\|\eta_q - \eta'_q\| \leq 2\Delta$, the above ratio is no more than e^{ϵ}.

In this scenario, we do not jointly optimize (6) w.r.t. U and V because it would invalidate Theorem 1. We cannot guarantee that the values of u_i at the both sides of Eqa (7) are equal in this case. As a result, it would significantly increase the difficulty of deriving the distribution of eta, which is critical for our proposal.

4.2 Scenario with Untrusted Recommender

We now try to extend our proposal to the scenario described in Sec. 3.3. In this new setting, the central recommender is considered to be non-trusted. Therefore, users can no longer feel free to send their raw ratings of items to the recommender.

Fortunately, we observe from (2) that the adaption of a user's profile vector u_i only relies on user i's own ratings rather than those of other users, which means all these computations can be performed locally instead of centrally on the recommender side. Therefore, user profile vectors can be kept

secret on the user side, and thus protected against the recommender. Certainly, user i has to request the immediate profile vectors of items that he has rated from the recommender in each round. So long as these item profile vectors satisfying ϵ-differential privacy, the user profile vector derived from them must satisfy ϵ-differential privacy as well, which means no user can learn private ratings of others in this process.

Different from the user profile vectors, the adaption of each item profile vector v_j in each iteration relies on multiple users' ratings of item j as well as their private user profile vectors, which means such computations involve multi-parties. We may leverage existing secure multi-party computing methods based on cryptographic techniques to protect user privacy in this process, however these methods usually rely on too many heavyweight cryptographic operations that are far from practical. In this paper, we try to implement a multi-party version of the objective perturbation method described in the last section to address this challenge.

Specifically, after the objective is perturbed as (6), the updating rule of v_j in (3) becomes

$$\begin{aligned} v_j(t) = v_i(t-1) + \gamma \cdot (\nabla_{v_j}(U(t-1), V(t-1)) \\ + 2\mu v_j(t-1) - \eta_j), \end{aligned} \tag{9}$$

where η_j is a random noise vector whose density function $p(\eta_j) = k \cdot e^{-\frac{\epsilon \|\eta_j\|}{2\Delta}}$. Suppose that $User_j = \{i_1, i_2, \cdots, i_k\}$ are k users that have rated item j. Our proposal decomposes this update and make these users each compute a piece of information, i.e., her local gradient $\widehat{\nabla}^{i_s}_{v_j}(t) = \nabla^{i_s}_{v_j}(t) + \eta^{i_s}_j$, where $\nabla^{i_s}_{v_j}(t) = -2u_i(t)(r_{ij} - u_i(t)^T v_j(t))$ and $\sum_{s=1,2,\cdots,k} \eta^{i_s}_j = \eta_j$. They then forward these partial results to the recommender, who combines them to update $v_j(t)$. However, this extension encounters two challenges in order to satisfy ϵ-differential privacy:

(1) According to Theorem 1, the objective perturbation will preserve differential privacy if each noise vector $\eta_j \in N$ has the density

$$p(\eta_j) \propto e^{-\frac{\epsilon \|\eta_j\|}{2\Delta}}. \tag{10}$$

However, in the distributed scenario, η_j is no longer selected individually by the recommender. So, the first challenge is how the k users can independently select $\eta^{i_s}_j$ such that the resulting sum $\sum_{s=1,2,\cdots,k} \eta^{i_s}_j$ follows the distribution of (10)?

(2) The objective perturbation guarantees that publishing final V does not breach ϵ-differential privacy. Nevertheless, in the current setting, the untrusted recommender obtain not only final V but also a series of immediate results (i.e., $\widehat{\nabla}^{i_s}_{v_j}$) from each user after each iteration. Although each of these results include a noise vector, the recommender can easily eliminate their effect: Considering the following difference between results of consecutive iterations

$$\begin{aligned} \widehat{\nabla}^{i_s}_{v_j}(u_{i_s}(t), v_j(t)) - \widehat{\nabla}^{i_s}_{v_j}(u_{i_s}(t-1), v_j(t-1)) = \\ \nabla^{i_s}_{v_j}(u_{i_s}(t), v_j(t)) - \nabla^{i_s}_{v_j}(u_{i_s}(t-1), v_j(t-1)), \end{aligned} \tag{11}$$

the recommender learns the difference between the true local gradients and the effect of $\eta_{i_s j}$ has been eliminated. This may help the recommender predict some information about user i_s's ratings, and thus compromise privacy constraint. We name this attack *difference attack*.

Our solution to the first problem relies on the lemma below.

Lemma 1 *If each element in every η_j of (6) is independently and randomly picked from Laplace $(0, 2\Delta\sqrt{d}/\epsilon)$, the derived item profile matrix V satisfies ϵ-differential privacy.*

Proof 2 *We still follow the proof of 1 and come to (8). Suppose that $\eta_q = [x_1, x_2, \cdots, x_d]^T$. As each element x_i is independently randomly picked from the Laplace $(0, 2\Delta\sqrt{d}\epsilon)$ distribution, its probability density function is $p(x_i) = \frac{\epsilon}{4\Delta\sqrt{d}} e^{-\frac{\epsilon|x_i|}{2\Delta\sqrt{d}}}$.*

$$\begin{aligned} \frac{Prob[V = \bar{v_j}|D]}{Prob[V = \bar{v_j}|D']} &= \frac{\prod_{j\in\{1,2,\cdots,m\}} p(\eta_j)}{\prod_{j\in\{1,2,\cdots,m\}} p(\eta'_j)} = \frac{p(\eta_q)}{p(\eta'_q)} \\ &= e^{-\frac{\epsilon \sum_{k=1,2,\cdots,d} |x_k|}{2\Delta\sqrt{d}}} / e^{-\frac{\epsilon \sum_{k=1,2,\cdots,d} |x'_k|}{2\delta\sqrt{d}}} \\ &= e^{\frac{\epsilon \sum_{k=1,2,\cdots,d} (|x_k| - |x'_k|)}{2\Delta\sqrt{d}}} \\ &\le e^{\frac{\epsilon\sqrt{d \sum_{k=1,2,\cdots,d} (x_k - x'_k)^2}}{2\Delta\sqrt{d}}} \\ &= e^{\frac{\epsilon\sqrt{d}\|\eta_q - \eta'_q\|}{2\Delta\sqrt{d}}} \le e^{\epsilon} \end{aligned}$$

So, we obtain the conclusion.

Besides this lemma, we also have the following theorem:

Theorem 2 *If random number $h \sim Exponential(1)$, and random number $c \sim N(0,1)$ independent of h, then $X = \mu + b\sqrt{2h}c \sim Laplace(\mu, b)$. Here, the notation "$\sim$" means "distributed as".*

Proof 3 *Please see [Kotz* et al.*, 2001].*

We can then use the above theorem to guide each user $i_s \in User_j$ to pick $\eta^j_{i_s}$ independently such that the resulting sum η_j satisfying (10). In particular, before the first round of iteration, the recommender picks a random number vector $H_j \in \mathbb{R}^d$, of which each element $H_j[l] \sim Exponential(1)$, and sends it to every user in $User_j$. Then, user i_s independently selects a random vector C_{i_s}, where each element $C_{i_s}[l] \sim N(0, 1/k)$ and computes $\eta^{i_s}_j$ according to

$$\eta^{i_s}_j[l] = \frac{2\Delta\sqrt{d}}{\epsilon} \cdot \sqrt{2H_j[l]} C_{i_s}[l]. \tag{12}$$

She will use $\eta^{i_s}_j$ to compute $\widehat{\nabla}^{i_s}_{v_j}$ in each round. It is easy to prove that the resulted sum of $\eta^{i_s}_j$ on k users satisfies (10).

Here, although H_j is picked by the recommender, he has no ways to predict the real value of $\eta^{i_s}_j$ due to the existence of C_{i_s} determined by the user locally. Thus, it is infeasible for him to remove this noise piece from $\widehat{\nabla}^{i_s}_{v_j}$. In addition, users can easily monitor whether the recommender generates $C_{i_s}[l]$ following $Exponential(1)$. Revealing one user's piece noise to the recommender can only harm himself, not give the recommender extra information about the values of other users.

For the scend challenge, Rajkumar and Agarwal [2012] propose a smart solution by adding another noise vector $\rho^{i_s}_j(t)$ to each local gradient:

$$\widetilde{\nabla}^{i_s}_{v_j}(t) = \nabla^{i_s}_{v_j}(t) + \eta^{i_s}_j + \rho^{i_s}_j(t). \tag{13}$$

Note that $\rho_j^{i_s}(t)$ should be re-sampled during each iteration. They show that the privacy leak due to (11) can be prevented if $\rho_j^{i_s}(t)$ can guarantee that $\nabla_{v_j}^{i_s}(t) + \rho_j^{i_s}(t)$ satisfies ϵ-differential privacy. In our case, it is easy to prove this requirement is fulfilled if the value of $\rho_j^{i_s}(t)$ is generated according to the density $p(\rho_j^{i_s}(t)) \propto e^{-\frac{\epsilon\|\rho_j^{i_s}(t)\|}{2\Delta}}$.

Unfortunately, if each local gradient value sent to the recommender includes such noise, the utility of V that we obtain is significantly affected according to our experiment. The previous work does not encounter this problem because it considers a different scenario: the parties participating in the distributed learning are organizations having aggregated the records of many users, rather than end users. As a result, the number of parties is far smaller than that in our scenario, which indicates that much fewer noises are introduced in each iteration. Thus, the utility of the final result is affected not as significantly as in our case.

We exploit a semi-honest third-party to address this problem. Here, semi-honest means that this party follows the protocol but is curious about user's privacy. In particular, our solution is composed of the following four steps:

(1) When user i_s requests $v_j(t)$ from the recommender in the beginning of the t-th iteration, the server generates a random noise vector $\psi_j^{i_s}(t)$, of which the elements are i.i.d. on $[0, P]$. Here, we denote by P a big integer. It then returns $\psi_j^{i_s}(t)$ together with $v_j(t)$ to user i_s.

(2) User i_s first computes $\widetilde{\nabla}_{v_j}^{i_s}(t) = \nabla_{v_j}^{i_s}(t) + \eta_j^{i_s} + \widetilde{\rho}_j^{i_s}(t)$. Note that the noise vector $\widetilde{\rho}_j^{i_s}(t))$ here is different from $\rho_j^{i_s}(t)$ in (13). We will describe its valuation later. Afterwards, the user further computes $\phi_j^{i_s}(t) = \widetilde{\nabla}_{v_j}^{i_s}(t) + \psi_j^{i_s}(t) \bmod P$, and forwards it to the third-party.

(3) The third-party aggregates the results from users in $User_j$ and computes $\phi_j(t) = \sum_{s=1}^{k} \phi_j^{i_s}(t) \bmod P$. The result is forwarded to the recommender.

(4) The recommender computes $\widetilde{\nabla}_{v_j}(t) = \phi_j(t) - \sum_{s=1}^{k} \psi_j^{i_s}(t) \bmod P$, and uses it to update $v_j(t)$.

Due to the existence of $\psi_j^{i_s}(t)$, it is infeasible for the curious third-party to learn any useful information about the real value of $\nabla_{v_j}^{i_s}(t)$ in the above process provided that it does not collude with the recommender. Thereby, the rating value $r_{i_s j}$ is also protected against the third-party. By applying this optimization, to defend against the difference attack we mentioned earlier, we require $\sum_{s=1}^{k}(\nabla_{v_j}^{i_s}(t) + \widetilde{\rho}_j^{i_s}(t))$ rather than every $\nabla_{v_j}^{i_s}(t) + \rho_j^{i_s}(t)$ to satisfy ϵ-differential privacy. It is easy to prove that so long as

$$\widetilde{\rho}_j(t) = \sum_{s=1}^{k} \widetilde{\rho}_j^{i_s}(t) \propto e^{-\frac{\epsilon\|\widetilde{\rho}_j(t)\|}{2\Delta}},$$

such requirement is fulfilled. Each user u_{i_s} can use the same method proposed earlier for picking $\eta_{i_s}^j(t)$ to pick $\widetilde{\rho}_j^{i_s}(t)$ based on Theorem 2. It is easy to demonstrate that $\widetilde{\rho}_j^{i_s}(t)$ is much smaller than $\rho_j^{i_s}(t)$ with a high probability, which means our scheme above can significantly reduce the side effects on the utility of the learning result.

4.3 Further Practical Consideration

By now, our proposal requires every user to stay online all the time during MF. Nevertheless, this is impractical as MF may last for one or several weeks and there must exist users' leaving and joining during such a long time. Fortunately, according to our experiments on Netflix datasets, SGD-based MF is robust against dynamic changes of users. In the experiments, we make users leave and join with specific proabilites before every iteration, and the adaption in each learning is just based on the data of currently online users. The results show that the precision of the final results is slightly affected.

However, our protection proposal confronts a big challenge under this dynamic setting: it is difficult for different groups of users to produce random variables with the same sum. Note that, in our current design, every noise vector η_j for objective perturbation equals the sum of random vectors locally picked by all the users in $User_j$. This vector should remain constant in different iterations. Nevertheless, in the dynamic setting, as online users in $User_j$ are always changing, η_j will vary in differential iterations, which result in that final V no longer satisfies ϵ-differential privacy.

We make use of the semi-honest third-party described earlier to address this problem. In this improvement, before starting MF, we add an additional initialization phase to determine every η_j in advance. In particular, this initialization works as follows:

(1) The recommender randomly picks k online users, i.e., $UO = \{u_{i_1}, u_{i_2}, \cdots, u_{i_k}\}$, and sends each $u_{i_s} \in UO$ a random vector $\beta_j^{i_s}$, of which the elements are i.i.d. on $[0, P]$.

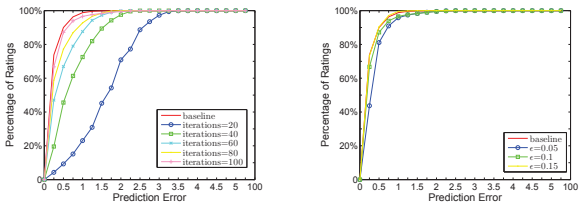
(2) Then, user u_{i_s} picks the local noise vector $\eta_j^{i_s}$ the same as before, and computes $\tau_j^{i_s} = \eta_j^{i_s} + \beta_j^{i_s} \bmod P$. The result is forwarded to the third-party.

(3) The third-party aggregates the results from each user and computes $\widetilde{\tau}_j = \alpha_j + \sum_{s=1}^{k} \tau_j^{i_s} \bmod P$, where α_j is another random vector, of which the elements are i.i.d. on $[0, P]$. The result is sent to the recommender.

(4) The recommender then derives $\widehat{\tau}_j = \alpha_j + \sum_{s=1}^{k} \eta_j^{i_s} \bmod P$ by removing each $\beta_j^{i_s}$, and keeps it secret.

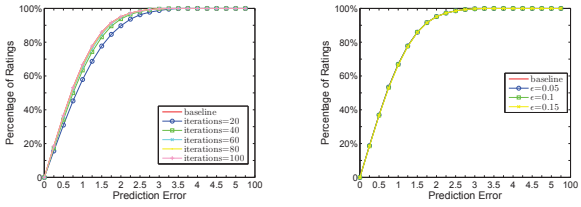
When MF starts, every iteration executes the same as described in the end of Sec. 4.2 except three exceptions: First, in Step 1, if there are k' online users collaborating to update v_j, the recommender randomly divides $\widehat{\tau}_j$ into k' pieces, and sends every user u_{i_s} in this group a piece $\widehat{\tau}_j^{i_s}$. Second, user u_{i_s} will use $\widehat{\tau}_j^{i_s}$ to replace $\eta_j^{i_s}$ in Step 2. Third, in Step 3 the third-party minuses α_j from $\phi_j(t)$ before uploading.

It is easy to find that by applying the above adaption, η_j no longer varies in different iterations and equals the sum of $\eta_j^{i_s}$, determined by the k online users in the initialization process. Moreover, it is infeasible for the recommender or the third-party to infer the true value of η_j if they do not collude. The recommender may compute $\widetilde{\nabla}_{v_j}(t) - \widehat{\tau}_j$ to remove η_j from $\widetilde{\nabla}_{v_j}(t)$. However, this operation will introduce a new and more random noise, i.e., $-\alpha_j$, which can better protect the user privacy. The situation is the same for the third-party.



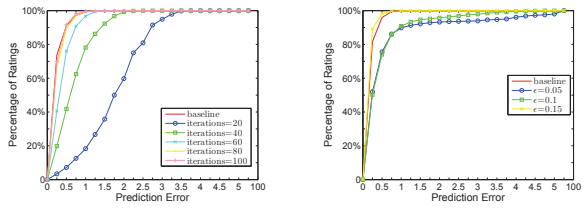
(a) $\epsilon = 0.1$, change iterations (b) 100 iterations, change ϵ

Figure 1: CDF of prediction errors for Netflix in S1: the recommender is trusted



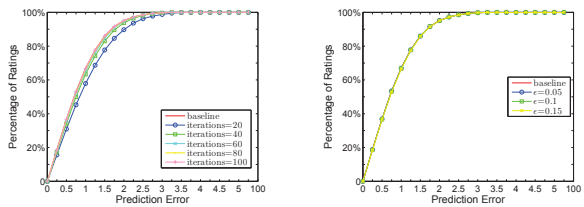
(a) $\epsilon = 0.1$, change iterations (b) 100 iterations, change ϵ

Figure 2: CDF of prediction errors for MovieLens in S1: the recommender is trusted



(a) $\epsilon = 0.1$, change iterations (b) 100 iterations, change ϵ

Figure 3: CDF of prediction errors for Netflix in S2: the recommender is untrusted but the online users are static



(a) $\epsilon = 0.1$, change iterations (b) 100 iterations, change ϵ

Figure 4: CDF of prediction errors for MovieLens in S2: the recommender is untrusted but the online users are static

5 Evaluation

We now evaluate the performance of our proposal based on two public datasets. The first one is reduced from the well-known Netflix dataset. Specifically, it is composed of 191668 users' ratings of 100 movies. The movies were selected uniformly & randomly from the original dataset, and all the users' ratings of these movies were included. We repeated each experiment by selecting different sets of movies, and did not observe obvious differences among the results. The second one is MovieLens 100k, which consists of 943 users' ratings of 1682 movies. For simplicity, we below refer to them as Netflix and MoveLens, respectively. The range of ratings is on a 5 star range for both.

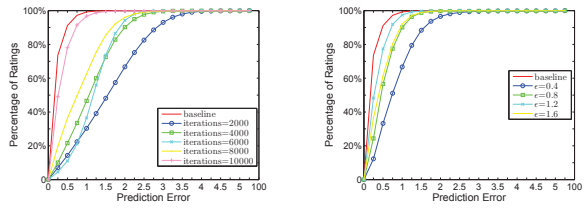
It is obvious that our privacy-protection mechanism mainly brings two kinds of side-effects on MF: First, it introduces inaccuracies due to the perturbations of the objective and intermediate results. Second, it brings communication overheads among users, the recommender and the third-party in each iteration, which dominate the time costs of the whole system. So, our evaluations mainly use MF accuracy and communication overheads as the key metrics.

For MF accuracy, we mainly consider two metrics. The first is the cumulative distribution function (CDF) of prediction errors, i.e., $\{|r'_{ij} - r_{ij}||(i,j) \in \mathcal{R}\}$. We denote by r'_{ij} the ratings predicted by the inner product of u_i and v_j derived from MF, and by $r_{i,j}$ the real ratings of users. The second one is the mean increase of prediction errors compared with the baseline, i.e., the original MF scheme without any protection running 100 iterations. We do not take [Nikolaenko *et al.*, 2013]'s mechanism as the baseline because its computation loads are several orders of magnitude greater than ours although they used a much smaller dataset. In addition, as they introduce few noises, their accuracy infinitely approaches the unprotected MF.

We implement our proposals for all the three scenarios considered in this paper: (S1) the recommender is trusted; (S2) the recommender is untrusted but the online users are static; (S3) the recommender is untrusted and the online users are dynamic. For each scenario, we vary ϵ and the iteration counts. The gain factor γ is 2^{-5}. The parameters λ and μ in (1) are both set to 0.001. The dimension d of all the profile vectors is set to 50.

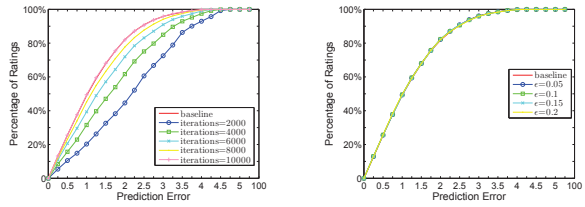
The results of the first metric for S1 are present in Fig. 1 and Fig. 2. We find that the precision increases with the iteration number and the privacy budget ϵ. Specifically, the side effects of objective perturbation on MF precision become negligible when $\epsilon \geq 0.15$ for Netflix after 70 iterations. According to Fig. 7(a), the mean increase of prediction errors is below 0.03 compared with the baseline. The results for MovieLens are much better: the MF precision becomes extremely approaching the baseline just after 40 iterations when $\epsilon = 0.05$. According to Fig. 8(b), the mean increase of prediction errors even slightly deceases compared with the baseline. This is because the precision of the raw MF (i.e., the baseline) is not so high that introducing noises in learning may help some elements escape from local optimums.

The results for S2 are present in Fig. 3 and Fig. 4. We can find that the results of this scenario are very close to those of S1 for both datasets, which indicates that the additional noises added in each iteration for defending against the difference attack (Please refer to Sec. 4.2) will not significantly affect MF precision as well. In particular, the mean increase



(a) $\epsilon = 0.8$, change iterations (b) 10000 iterations, change ϵ

Figure 5: CDF of prediction errors for Netflix in S3: the recommender is untrusted and the online users are dynamic



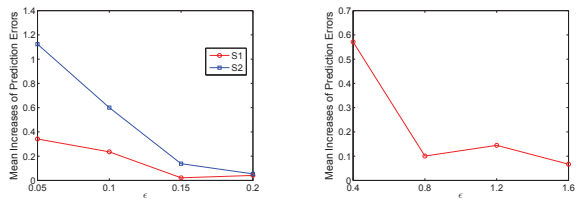
(a) $\epsilon = 0.1$, change iterations (b) 10000 iterations, change ϵ

Figure 6: CDF of prediction errors for MovieLens in S3: the recommender is untrusted and the online users are dynamic

of prediction errors for Netflix is around 0.14 when $\epsilon = 0.15$ (Please see Fig. 7(a)), and this value declines to near 0 for MovieLens (Please see Fig. 8(a)).

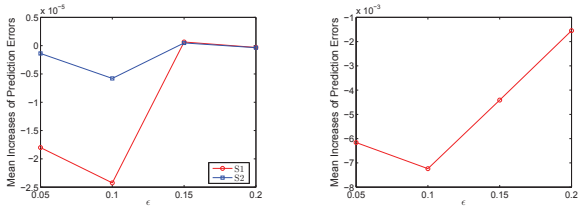
For S3, we first assume that the users get online and offline with probability of 0.005 and 0.01, respectively, after each iteration. The recommender randomly selects at most 1000 and 100 of the online users to participate in the learning in each iteration for Netflix and MoveLens, respectively. The results are shown in Fig. 5 and Fig. 6. We find that for Netflix if we slightly relax the privacy requirement, i.e., improve ϵ from 0.1 to 0.8, the precision of the results can also become extremely close to the baseline after 10000 iterations. In particular, according to Fig. 7(b) the mean increase of prediction errors is below 0.1 when $\epsilon = 0.8$. The increase of iteration rounds is due to that each iteration only covers the data of 1000 online users but not all the 191668 users. In addition, although the iteration round increases, the time cost for each iteration greatly decreases. The performance on MovieLens is still much better than Netflix: the curves in Fig. 6 has become indistinguishable from the baseline after 10000 iterations even when $\epsilon \geq 0.05$.

We also evaluate our proposals when the joining and leaving probabilities are varied. The results are present in Fig. 9 and Fig. 10. We can find that as the joining probability increases between $[0.005, 0.02]$, the mean increase of prediction errors compared with the baseline declines. This is easy to understand because a larger joining probability indicates more offline users getting online in each iteration, which results in that the MF process can cover a wider set of users after the same number of iterations. Compared with the joining probability, the effect of the leaving probability is not so



(a) Scenario 1 and 2 (100 iterations) (b) Scenario 3 (10000 iterations)

Figure 7: Mean increases of prediction errors for Netflix compared with the baseline



(a) Scenario 1 and 2 (100 iterations) (b) Scenario 3 (10000 iterations)

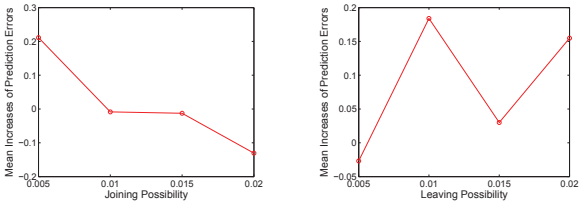
Figure 8: Mean increases of prediction errors for MovieLens compared with the baseline

obvious, especially for Netflix. This is because the base number of online users are much smaller than that of the offline users in the beginning, and thus the slight changing of the leaving probability cannot greatly change the composition of users participating in each iteration.

We next evaluate the communication overheads of our proposal. In S1, MF is performed completely by the recommender, who does not communicate with users. Thus, our proposal introduces no communication overheads. However, the proposals designed for S2 and S3 require cooperations among the recommender, the users and the third-party, and thus do bring communication overheads. Since S2 is not practical, we only consider S3 here.

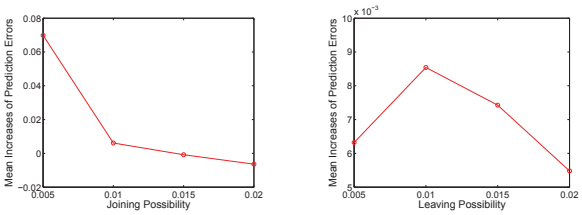
In this scenario, according to our simulation, if a user rated no more than 20 items, she only needs to download at most 12KB data from the recommender and uploads at most 5KB data to the third-party in each iteration. This indicates that our proposal is efficient enough to be deployed on mobile devices. In addition, for Netflix if the total number of items that the 1000 online users rated are at most 10000 (the number in the real-world is usually much smaller than this value), the total data that the third-party has to send to the recommender in each iteration is less than 3MB, which is still very efficient.

According to our experiment on Netflix, the average times for doing optimization on each client and the third party in each iteration are about 5.7×10^{-4}ms and 400ms, respectively. It is infeasible to conduct real experiments to measure the time costs for communication since we are lack of a testbed with 1000 concurrent users. Nevertheless, the litera-



(a) Leaving probability=0.01 (b) Joining probability=0.01

Figure 9: Effects of joining and leaving probabilities in S3 for Netflix (10000 iterations, $\epsilon = 0.8$)



(a) Leaving probability=0.01 (b) Joining probability=0.01

Figure 10: Effects of joining and leaving probabilities in S3 for MovieLens (10000 iterations, $\epsilon = 0.1$)

ture [Kimpe *et al.*, 2012; Crovella *et al.*, 1999] shows that it is not so difficult for an average provider to build a web system that can handle 1000 concurrent users (each may download or upload a file about 10kb) with the average response time below 1s. So, it is safe to assume that the total time for each iteration can be confined to 5s. Although this is slower than the baseline (0.46s), it is much faster than [Nikolaenko *et al.*, 2013]'s mechanism which took 2.5hr to finish one iteration based on a smaller dataset. For MovieLens, since there are up to 100 users participating in learning in each iteration, the communication overheads are no greater than those of Netflix according to our analysis.

6 Conclusion and Future Work

We conclude that a differentially private MF mechanism is feasible when the central server is untrusted. The results show that our proposals have little side effects on the precision of results and the efficiency is also high. People can leverage these proposals to build recommender systems in a privacy-preserving way. To the best of our knowledge, we are the first to apply differential privacy to the task of MF.

A promising future work is to adapt our proposal to improved variants of the raw MF, such as WNMF (i.e., Weighted Non-negative MF). According to our experiments, such mechanism may significantly improve the precision of MF.

References

Esma Aïmeur, Gilles Brassard, José M Fernandez, and et al. Alambic: a privacy-preserving recommender system for electronic commerce. *International Journal of Information Security*, 7(5):307–334, 2008.

Joseph A Calandrino, Ann Kilzer, Arvind Narayanan, and et al. "you might also like": Privacy risks of collaborative filtering. In *Proceedings of the 32nd IEEE Symposium on Security and Privacy (SP)*, pages 231–246. IEEE, 2011.

Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

John Canny. Collaborative filtering with privacy. In *Proceedings of the 23rd IEEE Symposium on Security and Privacy (SP)*, pages 45–57. IEEE, 2002.

Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems*, pages 289–296, 2009.

Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *The Journal of Machine Learning Research*, 12:1069–1109, 2011.

Mark E Crovella, Robert Frangioso, and Mor Harchol-Balter. Connection scheduling in web servers. Technical report, Boston University Computer Science Department, 1999.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and et al. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography*, pages 265–284. Springer, 2006.

Dries Kimpe, Philip Carns, Kevin Harms, and et al. Aesop: Expressing concurrency in high-performance system software. In *Proceedings of the 7th International Conference on Networking, Architecture and Storage (NAS)*, pages 303–312. IEEE, 2012.

Yehuda Koren, Robert Bell, and et al. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

Samuel Kotz, Tomasz Kozubowski, and Krzysztof Podgorski. *The Laplace distribution and generalizations: a revisit with applications to communications, exonomics, engineering, and finance*. Number 183. Springer Science & Business Media, 2001.

Ashwin Machanavajjhala, Aleksandra Korolova, and Atish Das Sarma. Personalized social recommendations: accurate or private. *Proceedings of the VLDB Endowment*, 4(7):440–450, 2011.

Frank McSherry and Ilya Mironov. Differentially private recommender systems: building privacy into the net. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 627–636. ACM, 2009.

Noman Mohammed, Rui Chen, Benjamin Fung, and et al. Differentially private data release for data mining. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 493–501. ACM, 2011.

Valeria Nikolaenko, Stratis Ioannidis, Udi Weinsberg, and et al. Privacy-preserving matrix factorization. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 801–812. ACM, 2013.

Yubin Park and Ayan Acharya. Differentially private recommendation systems: Practical implementation and interpretation. 2011.

Arun Rajkumar and Shivani Agarwal. A differentially private stochastic gradient descent algorithm for multiparty classification. In *International Conference on Artificial Intelligence and Statistics*, pages 933–941, 2012.

Yu Xin and Tommi Jaakkola. Controlling privacy in recommender systems. In *Advances in Neural Information Processing Systems*, pages 2618–2626, 2014.