# Privacy Enhanced Matrix Factorization for Recommendation with Local Differential Privacy

Hyejin Shin, Sungwook Kim, Junbum Shin, Xiaokui Xiao

**Abstract**—Recommender systems are collecting and analyzing user data to provide better user experience. However, several privacy concerns have been raised when a recommender knows user's set of items or their ratings. A number of solutions have been suggested to improve privacy of legacy recommender systems, but the existing solutions in the literature can protect either items or ratings only. In this paper, we propose a recommender system that protects both user's items and ratings. For this, we develop novel matrix factorization algorithms under local differential privacy (LDP). In a recommender system with LDP, individual users randomize their data themselves to satisfy differential privacy and send the perturbed data to the recommender. Then, the recommender computes aggregates of the perturbed data. This framework ensures that both user's items and ratings remain private from the recommender. However, applying LDP to matrix factorization typically raises utility issues with i) high dimensionality due to a large number of items and ii) iterative estimation algorithms. To tackle these technical challenges, we adopt dimensionality reduction technique and a novel binary mechanism based on sampling. We additionally introduce a factor that stabilizes the perturbed gradients. With MovieLens and LibimSeTi datasets, we evaluate recommendation accuracy of our recommender system and demonstrate that our algorithm performs better than the existing differentially private gradient descent algorithm for matrix factorization under stronger privacy requirements.

**Index Terms**—Local differential privacy, Matrix factorization, Recommender system, Random projection.

## 1 Introduction

With the increased adoption of mobile devices, consumers are increasingly using online markets to make purchases. This flood of data often means that consumers are confronted with many options, which makes it harder to make decisions. Recommender systems help consumers find items of interest and even suggest additional items. For this reason, recommender systems have become more and more common in our everyday life, ranging from movie to product recommendations. Recommender systems are collecting and mining user data for better user experience. However, user data includes personal information and often raises privacy concerns.

In order to generate relevant recommendations, recommender systems collect the (item, rating) pairs from each user. Many early studies [6], [16], [19] have shown that recommenders can abuse user privacy and extract information beyond what is intentionally revealed by the user. Recommenders can infer personal information such as gender, age, and even political orientation, based on the movies a user has watched [29]. Our goal is to protect user's private data including both ratings and items while keeping the quality of recommendation as much as possible.

Differential privacy [9] is a popular tool to address the privacy issue due to its strong privacy guarantee. There have been several works applying differential privacy to recommender systems. Most of them [2], [4], [17], [18], [19] assume the trusted recommender scenario and have focused on the output perturbation of collaborative filtering algorithms, as the classical differential privacy does. However, as recommendation services get more popular, many untrusted recommenders appear online and they may abuse user privacy for purposes beyond its own service. Even for trusted recommenders, there is a major security risk of possessing large amounts of sensitive personal data. Narayanan and Shmatikov [20] demonstrated the danger of accidental leakage of such personal data by deanonymizing some users in the Netflix Prize data set using public IMDb profiles.

As privacy concerns are raised more frequently, local privacy models have been suggested in many application areas for protecting users' privacy from untrusted servers. For recommendation services, there are also a number of the earlier works that adopted local privacy models. Shen and Jin [26], [27] applied differential privacy to protect the set of each user's items, but their randomization algorithm perturbed items likely within the pre-specified categories, due to which the user's category preference (e.g., genre) could be exposed. Recently, Hua et al. [13] also proposed local models where user's private data is perturbed before being submitted to the recommender. However, the existing methods [13], [26], [27] suffer from two deficiencies. First, they were designed to protect either users' ratings [13] or the items rated by user [26], [27], but not both. In many situations, a user's item set is as sensitive as her ratings since inference from which items a user has rated can discover sensitive information like political views or sexual orientation [29]. Thus, it is important to protect both user's ratings and items to guarantee a stronger privacy. Second, they focused on the protection of a single item or rating value among a user's entire items and their ratings. However, in recommendation systems, there are sets of highly correlated items (e.g., movies in the same genre). Masking the presence

- *H. Shin, S. Kim, J. Shin are with Samsung Research, Seoul 06765, Korea. E-mail: {hyejin1.shin, sw14.kim, junbum.shin}@samsung.com*
- *X. Xiao is with School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798. E-mail: xkxiao@ntu.edu.sg*

or absence of a single item among the correlated items is not secure enough to protect users' sensitive information. Also, users often upload a set of ratings at a time. In that sense, the protection of a single rating or item does not fit well in practice. One may try to extend the existing schemes in such a way that it applies differential privacy to every single item and rating. However, this approach introduces a huge error in aggregated data, which results in poor quality of predictions, as we show later in Section 4.1.

In recent years, local differential privacy (LDP) has received considerable attention in research communities and industries as a good alternative to prevent an untrusted server from identifying users or learning any of users' personal information. In LDP, each user randomizes her data to satisfy differential privacy in her own device and then sends the randomized answers to an aggregator. The key idea of LDP is to compute aggregates of users' data without collecting users' personal information. LDP has its roots in randomized response, first proposed by Warner in 1965 [28]. It came to the fore in recent years as Google has shown its applicability through RAPPOR, Google's LDP solution in its Chrome browser [10], [11]. Moreover, since Apple has announced the adoption of LDP to iOS 10 [1], LDP became the most promising privacy protection technique. By definition, it pursues masking whether or not a user appears in the database, known as *per-user privacy*, as introduced in Section 2. In the context of a recommender system, LDP assures that even if a user in the system is changed to another user whose data consists of entirely different ratings and items from the original user, the output of the collaborative filtering algorithm does not reveal meaningful difference to an attacker. However, this stronger privacy requirement could result in a large prediction error when the number of items is large. Reducing this error due to high dimensionality is very challenging. So far, there is no existing recommendation system that guarantees both per-user privacy and recommendation quality under LDP.

The goal of this paper is to build a recommender system that overcomes the limitations of the existing works [13], [26], [27], by guaranteeing strong privacy for each user. We use matrix factorization for recommendation among many collaborative filtering techniques. We develop novel differentially private gradient descent algorithms for matrix factorization. We protect user's private data including items, ratings and profiles by taking a recent LDP solution suggested by Nguyen et al. [21] for gradient perturbation. We show that the final item profiles of our matrix factorization algorithm do not significantly differ with or without a user so that per-user privacy is guaranteed, as we show later in Section 4. We reduce a large perturbation error due to a large number of items by adopting a dimensionality reduction technique through random projection. To the best of our knowledge, this is the first work to propose a LDP solution for matrix factorization that protects user's items and ratings together, guarantees per-user privacy, and keeps the recommendation quality.

Compared to the existing differentially private matrix factorization [13], [26], [27], the contributions of our work are as follows:

- We guarantee a stronger degree of privacy.
  Our matrix factorization algorithms guarantee *per-user privacy*, which is a stronger privacy guarantee than the existing algorithms [13], [26], [27]. In addition, our perturbation algorithms protect not only ratings and user profiles, but also the set of items rated by user. On the other hand, the existing works were targeted to protect either each user's ratings [13] or the items rated by the user [26], [27].
- We greatly improve recommendation accuracy.
  A direct application of LDP on user's data induces a large perturbation error that linearly grows with the number $m$ of items. We adopt random projection for dimension reduction and a recent randomization algorithm [21] to reduce the amount of noise added to gradient. We show that our matrix factorization algorithms estimate item profiles with the error linearly growing with $\log(m)$. We additionally reduce accuracy loss due to iteration by stabilizing the noisy gradient. Gradient descent algorithm runs iteratively. To meet $\epsilon$-differential privacy over $k$ iterations, the privacy budget $\epsilon$ needs to be divided by $k$, so the budget for each iteration becomes $\epsilon/k$. This results in the perturbation error growing linearly with the number $k$ of iterations. In our algorithm, we consider a factor that reduces the variance of the noisy gradient.
- We significantly reduce communication costs.
  In our algorithm, each user transmits only one bit to a server at each iteration while the existing work [13] requests each user to send her data whose length is proportional to the number of items rated by her. Our algorithm additionally reduces communication overheads from a server to users through dimension reduction.

This paper is organized as follows. In Section 2, we briefly introduce matrix factorization and differential privacy as preliminaries. In Section 3, we explain setting and privacy goals of our recommender system. In Section 4, we investigate the limitations of the direct application of LDP to gradient perturbation, then suggest solutions to overcome such limitations, and present the core algorithms for matrix factorization. Section 5 demonstrates the performance of the proposed algorithms with extensive experiments on two large datasets. Section 6 reviews related work. Finally, this paper concludes with some remarks in Section 7.

## 2 PRELIMINARIES

### 2.1 Matrix Factorization

Suppose $n$ users rate a subset of $m$ items (e.g., movies). We denote by $\mathcal{M} \subset \{1, \ldots, n\} \times \{1, \ldots, m\}$ the user/item pairs for which a rating has been generated. Let $M = |\mathcal{M}|$ be the total number of ratings. We denote the rating generated by user $i$ for item $j$ by $r_{ij}$. In a practical setting, the ratings provided are sparse, so $M$ is much smaller than $nm$, while both $n$ and $m$ are large.

Given the ratings $\{r_{ij} : (i,j) \in \mathcal{M}\}$, a recommender system predicts the ratings for items that are not rated by users. There are several ways to do the prediction of unrated items. One of the most popularly used techniques is matrix

TABLE 1
Notations

| Notation | Meaning |
|---|---|
| $n$ | number of users |
| $m$ | number of items |
| $k$ | number of iterations |
| $d$ | number of latent factors |
| $U$ | user profile matrix whose rows are $u_i \in \mathbb{R}^d$ |
| $V$ | item profile matrix whose rows are $v_j \in \mathbb{R}^d$ |
| $\nabla_V^t$ | gradient matrix for $V$ at iteration $t$ |
| $a^*$ | perturbed version of a quantity $a$ |
| $\nabla_V^*$ | gradient matrix for $V$ computed from noisy gradient matrices of $n$ users |
| $(A)_{j,l}$ | $(j,l)$ element of a matrix $A$ |

factorization, which (i) represents each user (resp. item) as a $d$ dimensional vector, referred to as a *profile*, and (ii) models the relevance of an item to a user as the inner product of their profiles. Specifically, matrix factorization computes the user profiles $u_i \in \mathbb{R}^d$, $1 \le i \le n$, and the item profiles $v_j \in \mathbb{R}^d$, $1 \le j \le m$, by minimizing the regularized mean squared error on the set of known ratings

$$\frac{1}{M} \sum_{(i,j)\in\mathcal{M}} (r_{ij} - u_i^T v_j)^2 + \lambda_u \sum_{i=1}^{n} \|u_i\|^2 + \lambda_v \sum_{j=1}^{m} \|v_j\|^2 \quad (1)$$

for $d \ll \min(m,n)$ and positive constants $\lambda_u, \lambda_v$. The unobserved ratings, $r_{ij}$ for $(i,j) \notin \mathcal{M}$, are then predicted by $u_i^T v_j$.

Two popular approaches to solve the nonconvex optimization problem in (1) are (stochastic) gradient descent and alternating least squares. The latter algorithm iteratively solves a least squares optimization by rotating between fixing the $u_i$'s and fixing the $v_i$'s. Gradient descent iteratively adapts the profiles $u_i$ and $v_j$ by the following update formulas:

$$u_i^t = u_i^{t-1} - \gamma_t \cdot \left\{ \nabla_{u_i} \phi(U^{t-1}, V^{t-1}) + 2\lambda_u u_i^{t-1} \right\}, \quad (2)$$

$$v_j^t = v_j^{t-1} - \gamma_t \cdot \left\{ \nabla_{v_j} \phi(U^{t-1}, V^{t-1}) + 2\lambda_v v_j^{t-1} \right\}, \quad (3)$$

where $\gamma_t > 0$ is a learning rate at iteration $t$, $U$ is the user profile matrix whose $i$th row is $u_i$, $V$ is the item profile matrix whose $j$th row is $v_j$, and $\nabla_{u_i}\phi(U,V), \nabla_{v_j}\phi(U,V)$ are the gradients for $u_i$ and $v_j$. $\nabla_{u_i}\phi(U,V)$ (resp. $\nabla_{v_j}\phi(U,V)$) is obtained by taking the derivative of the mean squared error in (1) with respect to $u_i$ (resp. $v_j$). Formally, we have

$$\nabla_{u_i}\phi(U,V) = -\tfrac{2}{M} \textstyle\sum_{j:(i,j)\in\mathcal{M}} v_j (r_{ij} - u_i^T v_j), \quad (4)$$

$$\nabla_{v_j}\phi(U,V) = -\tfrac{2}{M} \textstyle\sum_{i:(i,j)\in\mathcal{M}} u_i (r_{ij} - u_i^T v_j). \quad (5)$$

Note that $\nabla_{u_i}\phi(U,V)$ (resp. $\nabla_{v_j}\phi(U,V)$) captures how the user profile $u_i$ (resp. item profile $v_j$) should be changed to reduce the mean squared error. The learning rate $\gamma_t$ is a constant, typically set to $O(1/t)$ [24]. In Table 1, we list notations frequently used throughout the paper.

### 2.2 Differential Privacy

For our analysis, we adopt differential privacy. Differential privacy was first introduced by Dwork et al. [9] and it aims to preclude adversarial inference about any underlying record from its output. Its definition is as follows:

TABLE 2
Comparison of Privacy Degree - "S" and "E" stand for "Single" and "Entire", respectively.

| | Items | | Ratings | | Iterations | |
|---|---|---|---|---|---|---|
| | S | E | S | E | S | E |
| Ours | ✔ | | ✔ | | ✔ | |
| Hua et al. [13] | × | | ✔ | × | ✔ | × |
| Shen et al. [26], [27] | ✔ | × | × | | n/a | |

***Definition 1.*** A randomized algorithm $\mathcal{A}$ satisfies $\epsilon$-differential privacy if for any two neighboring databases $D$ and $D'$ and any measurable subset $O \subset Range(\mathcal{A})$,

$$Pr[\mathcal{A}(D) \in O] \le e^{\epsilon} Pr[\mathcal{A}(D') \in O],$$

where the probability is over the randomness of $\mathcal{A}$. That is, the probability of generating the same output $O$ using $\mathcal{A}$ on $D$ is at most $e^{\epsilon}$ times smaller than on $D'$.

The privacy parameter $\epsilon$ controls the privacy and utility tradeoff. In this paper, we consider two datasets $D$ and $D'$ to be neighbor if $D$ and $D'$ differ in the entire pairs of items and ratings of one user. On the other hand, Hua et al. [13] and Shen and Jin [26], [27] considered two neighboring datasets $D$ and $D'$ differing in a single rating (or a single item) of one user. The definition of neighboring datasets makes difference in the degree of privacy when publishing the final $V$: Our algorithm guarantees *per-user privacy* while the existing works [13], [26], [27] guarantee *per-item privacy*.

Differential privacy was originally proposed for the setting where a trusted server (i) holds a database where each tuple is collected from a user *without perturbation*, and (ii) releases perturbed information from the dataset using a randomized algorithm. However, if the server is untrusted, then each individual would have to perturb her tuple before it is collected by the server, so as to protect privacy. In that case, the tuple owned by the user is regarded as a singleton database, and we require that the perturbation algorithm $f$ should ensure differential privacy when such a singleton database is given as the input. That is, for any two tuples $x, x' \in Domain(f)$ and any measurable subset $S \subset Range(f)$, we require that

$$Pr[f(x) \in S] \le e^{\epsilon} Pr[f(x') \in S],$$

where the probability is over the randomness of $f$. This variant of differential privacy is referred to as $\epsilon$-local differential privacy ($\epsilon$-LDP) [8]. We focus on $\epsilon$-LDP instead of the original notion of $\epsilon$-differential privacy (Definition 1), as the former provides a much stronger degree of privacy protection for each user against the server.

## 3 System Overview

We assume that the recommender is untrusted and users do not want to share their private data with the recommender. Our goal is to build a recommender system where individual users randomize their data themselves and make a recommendation locally in their devices while the recommender computes aggregates of the perturbed data. Our system aims to achieve the following privacy requirements (see Table 2):

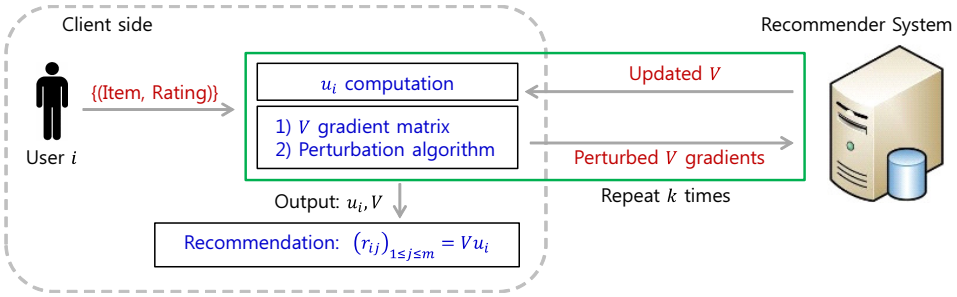



Fig. 1. Overview of Our Recommender System

(i) Privacy of a user's entire ratings and items: Our system requires a perturbation algorithm running on user's device to satisfy $\epsilon$-LDP. This automatically guarantees $\epsilon$-differential privacy when publishing the final $V$, as shown later in Section 4. Our system protects both user's rating and items from any adversaries including the recommender. The existing works protect either ratings [13] or items [26], [27] by guaranteeing $\epsilon$-differential privacy when only single rating or item are changed.

(ii) Privacy over all iterations of gradient descent: Gradient descent iteratively (say $k$ times) computes profiles. The difference of gradients at the $s$-th and $t$-th iterations ($1 \le s < t \le k$) leaks some information about user's ratings or items and leads to inference attack by an adversary. Thus, our system aims to provide $\epsilon$-differential privacy for all the gradients submitted by each user during iterations of gradient descent. The existing works only consider $\epsilon$-differential privacy over two consecutive iterations (i.e., single difference among entire iterations) [13], [23], in which user's privacy over iterations is guaranteed only when each user is involved in the gradient computation at most twice among $k$ iterations.

Figure 1 shows how our system works. In our system, the gradient descent algorithm requires iterative feedbacks between the server and users. We observe from (4) that the user profiles $u_i$ can be computed in user's device without submitting her data to the recommender when the $v_j$ are shared by the recommender, as in [13]. On the other hand, the item profiles $v_j$ are computed after all users' gradients are collected and so its computation is performed by the recommender. At each iteration step, the recommender must give the updated $v_j$ to users and then each user computes her profile $u_i$ in her own device and submits a noisy version of the gradient to the recommender. The recommender then updates the item profiles $v_j$ by aggregating the noisy gradients. These feedbacks between users and the server continue until a fixed number $k$ of iterations is reached. At each iteration, the privacy budget is assigned to be $\epsilon/k$, where $\epsilon$ is given by the recommender. After $k$ iterations, the recommender broadcasts the final item profiles $V$ and then the ratings for unrated items are predicted by $u_i^T v_j$, $j = 1, \ldots, m$, on user's device. The recommendation is finally made to users based on the predicated ratings in their devices.

Since the gradients of individual users include information about the ratings $r_{ij}$ and user profiles $u_i$, we need to protect users' gradients over every iteration. Thus, our randomized perturbation algorithm is designed to meet $\epsilon/k$-LDP at each iteration in order to protect her entire data consisting of items, ratings, and user profiles.

In the local privacy model, $M$ in (4) and (5) cannot be known a priori unless users disclose the numbers of items that they have rated. Therefore, we replace $M$ by $n$ for the development of our private gradient descent algorithms. All parameters, namely $n$, $\gamma_t$, $\lambda_u$, $\lambda_v$, $d$, and $k$ are given by the recommender. The privacy budget $\epsilon$ is also given by the recommender.

## 4 Differentially Private Matrix Factorization

### 4.1 A Simple Solution for Protecting Items and Ratings

The earlier study [13] applied the Laplace mechanism to gradients in order to protect users' ratings. In their algorithms, the recommender requests the users who have rated item $j$ to submit their gradients in a private manner. Therefore, the aggregator can learn whether a user has rated item $j$. In this subsection, we suggest a simple solution to overcome the issue in the existing works.

Let $y_{ij}$ be 1, if user $i$ rates item $j$, and 0, otherwise. Suppose that $r_{ij} > 0$ for item $j$ rated by user $i$. We let $r_{ij} = 0$ for $(i, j) \notin \mathcal{M}$. Then, observe that

$$\sum_{(i,j)\in\mathcal{M}} (r_{ij} - u_i^T v_j)^2 = \sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}(r_{ij} - u_i^T v_j)^2,$$

so that (5) is rewritten as

$$\nabla_{v_j}\phi(U, V) = -2n^{-1}\sum_{i=1}^{n} y_{ij} u_i (r_{ij} - u_i^T v_j). \tag{6}$$

If user $i$ does not rate item $j$, then $y_{ij} = 0$, and hence, her gradient for item $j$ is 0.

One way to protect "which items are rated by a user" is to perturb the item vector $Y_i = (y_{ij})_{1\le j\le m}$ using the randomized response mechanism [28]. Specifically, we generate $y_{ij}^*$ from $y_{ij}$ as follows:

$$y_{ij}^* = \begin{cases} 0, & \text{with probability } p/2, \\ 1, & \text{with probability } p/2, \\ y_{ij}, & \text{with probability } 1-p \end{cases} \tag{7}$$

for $j = 1, \ldots, m$.

To protect $g_{ij} = (g_{ijl})_{1 \le l \le d} = -2u_i(r_{ij} - u_i^T v_j)$, individual users perturb the $g_{ij}$ in their devices such that

$$g^*_{ijl} = g_{ijl} + \eta_{ijl}, \tag{8}$$

where the $\eta_{ijl}$ are independent random variables that follow a Laplace distribution with scale $\sigma$. If each user submits $\{(y^*_{ij}, g^*_{ij1}, \ldots, g^*_{ijd}) : j = 1, \ldots, m\}$ to the server, then it can be shown that the mean of the individual gradients, $\nabla_{v_j} = n^{-1} \sum_{i=1}^n y_{ij} g_{ij}$, could be estimated by

$$\nabla^*_{v_j} = \frac{1}{n} \sum_{i=1}^n \left( \frac{y^*_{ij} - p/2}{1-p} \right) g^*_{ij}.$$

The server updates the item profiles $v_j$ with the mean of the noisy gradients, i.e., $v_j^t = v_j^{t-1} - \gamma_t \cdot \{\nabla^{*,t-1}_{v_j} + 2\lambda_v v_j^{t-1}\}$.

If we set $p = 2/(1 + e^{\epsilon_1/k})$ and $\sigma = \frac{\Delta d}{\epsilon_2/k}$ in each iteration with $\Delta = \max_{i,j} r_{ij} - \min_{i,j} r_{ij}$ and $\epsilon = \epsilon_1 + \epsilon_2$, we can show that the final $V$ satisfies $\epsilon$-differential privacy for two datasets $D$ and $D'$ differing in one rating of one user. It indicates that this algorithm can protect only a single item or rating of a user. To protect the entire items and ratings of a user, we have to set $p = 2/(1 + e^{(\epsilon_1/k)/m})$ and $\sigma = \frac{m\Delta d}{\epsilon_2/k}$ in each iteration. In that case, the final $V$ satisfies $\epsilon$-differential privacy for two datasets $D$ and $D'$ differing in one user's entire data, so that the matrix factorization algorithm guarantees per-user privacy. However, this stronger privacy requirement results in the estimation error for $V$ linearly growing with $m$, given as $O\left(\frac{md\sqrt{\log(md)}}{(\epsilon_1/k)(\epsilon_2/k)\sqrt{n}}\right)$. Since $m$ is typically large in recommender systems, this estimation error is unacceptably large, which deteriorates the recommendation quality. Therefore, in the next two subsections, we develop new matrix factorization algorithms that guarantee per-user privacy, but reduce the estimation error.

### 4.2 Attempt 1: Gradient Perturbation for Per-User Privacy

To reduce the error in the $V$ estimation, we apply the randomized method proposed by Nguyen et al. [21]. In particular, the method considers the scenario where each user has a multiple-dimensional tuple to be collected by a server, and it requires that, when submitting her tuple, the user randomly selects one dimension, and submits a perturbed version of her tuple value on the selected dimension, while ignoring all other dimensions. Nguyen et al. show that the perturbed values collected by the server can be used to estimate the mean of all users' values on each dimension, and the estimation error is significantly smaller than the case when each user submits her values on all dimensions.

Algorithm 1 shows our matrix factorization algorithm that utilizes Nguyen et al.'s method [21]. The key idea of Algorithm 1 is that each user sends the perturbed gradient for a randomly selected element from the $m \times d$ gradient matrix. Based on the analysis in [21], this randomization technique reduces the estimation error of the gradients $\nabla_{v_j}$ at each iteration from $O(m)$ to $O(\sqrt{m})$. Theorem 2 shows the accuracy of the final item profiles after $k$ iterations.

Since each user submits a noisy gradient only for a randomly selected item, and its value is either $B$ $(= md\frac{e^{\epsilon/k}+1}{e^{\epsilon/k}-1})$ or $-B$ regardless of user's items, no adversary can learn about which items are rated by her or how much she likes those items. Theorem 1 proves the strong privacy guarantee of our algorithm.

**Algorithm 1** Differentially Private Gradient Descent

**input:** Predefined iteration number $k$ and privacy parameter $\epsilon$
**output:** Item profile matrix $V \in \mathbb{R}^{m \times d}$

Initialize $U$, $V$ and a counter $iter = 0$
**while** $iter \le k$ **do**
  Initialize $\nabla^*_V \in \{0\}^{m \times d}$
  **for** $i = 1$ to $n$ **do**
    Initialize $x^*_i \in \{0\}^{m \times d}$
    Sample $j$ uniformly at random from $\{1, 2, \ldots, m\}$
    Sample $l$ uniformly at random from $\{1, 2, \ldots, d\}$
    Compute $(x_i)_{j,l} = -2y_{ij}u_{il}(r_{ij} - u_i^T v_j)$
    If $(x_i)_{j,l} \notin [-1, 1]$, project $(x_i)_{j,l}$ onto $[-1, 1]$
    Draw $T \sim Bernoulli\left(\frac{(x_i)_{j,l}(e^{\epsilon/k}-1)+e^{\epsilon/k}+1}{2(e^{\epsilon/k}+1)}\right)$
    **if** $T = 1$ **then** $(x^*_i)_{j,l} = md\frac{e^{\epsilon/k}+1}{e^{\epsilon/k}-1}$
    **else** $(x^*_i)_{j,l} = -md\frac{e^{\epsilon/k}+1}{e^{\epsilon/k}-1}$
    **end if**
    Compute $\nabla^*_V = \nabla^*_V + x^*_i$
  **end for**
  Compute $\nabla^*_V = \nabla^*_V / n$
  $iter = iter + 1$
  $V = V - \gamma_t \cdot \{\nabla^*_V + 2\lambda_v V\}$
  **for** $i = 1$ to $n$ **do**
    Derive $\nabla_{u_i}$ in (4) and $u_i = u_i - \gamma_t \cdot \{\nabla_{u_i} + 2\lambda_u u_i\}$
  **end for**
**end while**
**return** $V$

***Lemma 1.*** Let $x_i$ be the non-private gradient matrix of user $i$ and $x^*_i$ a noisy gradient matrix of user $i$ in Algorithm 1. Assume that $x_i \in [-1, 1]^{m \times d}$. Then, each submission of user's gradients satisfies $\epsilon/k$-LDP, so noisy gradients submitted by each user over $k$ iterations satisfy $\epsilon$-LDP.

*Proof.* Similarly to Nguyen et al. [21], we have

$$\frac{Pr[x^*_i = v | x_i]}{Pr[x^*_i = v | x'_i]} \le \frac{\max_{x_i}\{(x_i)_{j,l}(e^{\epsilon/k} - 1) + e^{\epsilon/k} + 1\}}{\min_{x'_i}\{(x'_i)_{j,l}(e^{\epsilon/k} - 1) + e^{\epsilon/k} + 1\}} = e^{\epsilon/k}. \quad \square$$

***Theorem 1.*** Algorithm 1 satisfies $\epsilon$-differential privacy.

*Proof.* Suppose that two datasets $D$ and $D'$ differ in the ratings of user $p$. For $w \in \mathbb{R}^{m \times d}$, observe that

$$\begin{aligned} Pr\left[\nabla^*_V = w | D\right] &= Pr\left[\sum_{i=1}^n x^*_i = nw \middle| D\right] \\ &= \sum_{\substack{\tilde{w}_1, \ldots, \tilde{w}_n \\ \tilde{w}_1 + \cdots + \tilde{w}_n = nw}} \prod_{i=1}^n Pr\left[x^*_i = \tilde{w}_i | D\right] \\ &= \sum_{\substack{\tilde{w}_1, \ldots, \tilde{w}_n \\ \tilde{w}_1 + \cdots + \tilde{w}_n = nw}} \prod_{i=1}^n Pr\left[x^*_i = \tilde{w}_i | x_i\right], \end{aligned}$$

where the probability space is over the coin flips of the binary randomization in Algorithm 1. Thus, we have

$$\frac{Pr[\nabla_V^* = w|D]}{Pr[\nabla_V^* = w|D']} \leq \max_{\substack{\tilde{w}_1,\ldots,\tilde{w}_n \\ \tilde{w}_1+\cdots+\tilde{w}_n=nw}} \prod_{i=1}^{n} \frac{Pr\left[x_i^* = \tilde{w}_i|x_i\right]}{Pr\left[x_i^* = \tilde{w}_i|x_i'\right]} = \max_{\tilde{w}_p} \frac{Pr\left[x_p^* = \tilde{w}_p|x_p\right]}{Pr\left[x_p^* = \tilde{w}_p|x_p'\right]} \leq e^{\epsilon/k},$$

which means that the derived $V$ at each iteration satisfies $\epsilon/k$-differential privacy. Therefore, for the final item profile $V$, we have

$$\frac{Pr[V = \bar{V}|D]}{Pr[V = \bar{V}|D']} \leq \max_{a_1,..,a_k} \prod_{t=1}^{k} \frac{Pr[\nabla_V^{*,t} = a_t|D]}{Pr[\nabla_V^{*,t} = a_t|D']} \leq e^{\epsilon}. \quad \square$$

Theorem 1 states that when aggregating noisy gradients submitted by users at each iteration, the aggregator cannot distinguish the final item profiles $V$ with or without a user. Thus, Algorithm 1 guarantees per-user privacy and can completely prevent re-identification attacks.

***Theorem* 2.** Suppose that the learning rate $\gamma_t$ decreases as iteration goes and satisfies $\sum_{t=1}^{\infty} \gamma_t^2 < \infty$. Let $V^*$ be the item profile matrix derived by Algorithm 1 and $V$ the item profile matrix in non-private manner after $k$ iterations. With at least $1-\beta$ probability,

$$\|V^* - V\|_{\max} = O\left(\frac{\sqrt{md\log(md/\beta)}}{(\epsilon/k)\sqrt{n}}\right).$$

*Proof.* First observe that $V^* - V = -\sum_{t=1}^{k} \gamma_t c_t (\nabla_V^{*,t} - \nabla_V^t)$ with $c_t = \prod_{j=t+1}^{k}(1 - 2\lambda_v\gamma_j)$. The random variables $(\nabla_V^{*,t})_{j,l}, 1 \leq t \leq k$, are independent and bounded by $O\left(\frac{md}{\epsilon/k}\right)$. Also, $(\nabla_V^{*,t})_{j,l}$ is an unbiased estimator of $(\nabla_V^t)_{j,l}$, and $\text{Var}((\nabla_V^{*,t})_{j,l}) = O\left(\frac{md}{(\epsilon/k)^2 n}\right)$. Then, by Bernstein's inequality, we have

$$\begin{aligned}
&Pr\left[\|\sum_{t=1}^{k} \gamma_t c_t(\nabla_V^{*,t} - \nabla_V^t)\|_{\max} > \lambda \;\middle|\; D\right] \\
&\leq \sum_{j=1}^{m}\sum_{l=1}^{d} Pr\left[\left|\sum_{t=1}^{k} \gamma_t c_t((\nabla_V^{*,t})_{j,l} - (\nabla_V^t)_{j,l})\right| > \lambda \;\middle|\; D\right] \\
&\leq 2 \cdot md \cdot \exp\left(-\frac{\lambda^2}{2\sum_{t=1}^{k}\gamma_t^2 c_t^2 \text{Var}((\nabla_V^{*,t})_{j,l}) + \frac{2}{3}\lambda md\frac{e^{\epsilon/k}+1}{e^{\epsilon/k}-1}}\right) \\
&= O\Big(md \cdot \exp\big(-n\lambda^2(\epsilon/k)^2/md\big)\Big)
\end{aligned}$$

since $\sum_{t=1}^{k} \gamma_t^2 < \infty$ and $|c_t| \leq 1$ for small $\lambda_v \downarrow 0$. Taking $\lambda = O\left(\frac{\sqrt{md\log(md/\beta)}}{(\epsilon/k)\sqrt{n}}\right)$, the desired rate for $\|V^* - V\|_{\max}$ is achieved. $\square$

Theorem 2 states that, with Algorithm 1, we could reduce the estimation error for $V$, compared to the direct application of LDP with randomization algorithms (7) and (8).

With Algorithm 1, each user sends only one element that is randomly selected from the $m \times d$ matrix of gradients for updating $V$ to the aggregator and so transmits one bit to the aggregator. This greatly reduces communication costs in multiparty computation between users and server, compared to the existing work [13] where each user is requested to send her perturbed gradients whose length is proportional to the number of items rated by her.

The update of $V$ with the full gradient can be done when all the users are involved in the gradient computation. However, participation of all the users to the gradient computation is difficult in practice. Stochastic gradient descent (SGD), which updates $V$ with the gradient of a single user at each iteration, is the most popular algorithm in practice, but its large variability makes the convergence of iterative algorithm slow. When a noisy gradient of a single user is used for updating, updates would be bouncing even more, resulting in a slower convergence. A practical compromise between conventional gradient descent and stochastic gradient descent is a mini-batch gradient descent. With a mini-batch gradient descent, we can allow users to leave and join the multiparty computation whenever they want.

**Optimization (Variance Reduction).** From the experiment, we observe the gradients perturbed by Algorithm 1 could have excessively large error when the privacy budget $\epsilon/k$ at each iteration is small. In particular, for a fixed $\epsilon$, as the number $k$ of iterations increases, the privacy budget at each iteration decrease, and hence, the variation in a noisy gradient becomes larger. Thus, the noisy gradient could explode, which makes the gradient descent algorithm diverge, since $\frac{e^{\epsilon/k}+1}{e^{\epsilon/k}-1} = O(\frac{k}{\epsilon})$. To stablize the iterative algorithm, we change the learning rate from $\gamma_t$ to $\gamma_t/f_k$, where $f_k$ is a function of $k$ that increases faster than $k$. The update formula in Algorithm 1 becomes $V = V - (\gamma_t/f_k) \cdot \{\nabla_V^* + 2\lambda_v V\}$. This formula can also be interpreted as the update with the smaller gradients $\frac{1}{f_k}\nabla_V^*$ and learning rate $\gamma_t$. Interestingly, the resulting $V$ with the correction factor $\frac{1}{f_k}$ still satisfies $\epsilon$-differential privacy. Although the factor $\frac{1}{f_k}$ plays a role in reducing the variability in the gradients due to the perturbation, it induces a bias. Specifically, it can be shown that

$$\begin{aligned}
&\mathbb{E}\left[\left|\frac{1}{f_k}(\nabla_V^*)_{j,l} - (\nabla_V)_{j,l}\right|^2 \;\middle|\; D\right] \\
&= \frac{1}{f_k^2} Var[(\nabla_V^*)_{j,l}] + \left\{\frac{f_k - 1}{f_k}(\nabla_V)_{j,l}\right\}^2 \\
&= O\left(\frac{k^2}{f_k^2} \cdot \frac{md}{\epsilon^2 n}\right) + O(1)
\end{aligned}$$

and so we have

$$\|V^* - V\|_{\max} = O\left(\frac{k}{f_k} \cdot \frac{\sqrt{md\log(md/\beta)}}{\epsilon\sqrt{n}}\right) + O(1).$$

As we pick a larger $f_k$, the bias gets larger. So, we have to decide $f_k$ appropriately to compromise between variance and bias. When we choose $f_k = k$, it greatly reduces the estimation error of $V^*$ compared to Algorithm 1 without correction factor, but we observe from the experiment that the performance degrades very slightly as the number of iterations increases. This is because a rigorous bound for $\|V^* - V\|_{\max}$ with finite $k$ is given by $O\left(\sqrt{\sum_{t=1}^{k}\gamma_t^2 c_t^2} \cdot \frac{\sqrt{md\log(md/\beta)}}{\epsilon\sqrt{n}}\right) + O\left(\frac{k-1}{k}\sum_{t=1}^{k}\gamma_t c_t \|\nabla_V^t\|_{\max}\right)$. In our experiment, we set $f_k = k^2$. Thus, for small $\epsilon$ such that $\frac{k\epsilon\sqrt{n}}{\sqrt{md}} = o(1)$, the

correction factor significantly reduces the estimation error for $V$ by $O\left(\frac{\sqrt{md\log(md/\beta)}}{k\epsilon\sqrt{n}}\right)$. On the other hand, for large $\epsilon$ such that $\frac{\sqrt{md}}{k\epsilon\sqrt{n}} = o(1)$, the estimation error for $V$ is $O(1)$, i.e., the estimation error is dominated by the bias. In that case, there is no more improvement no matter how large $\epsilon$ is. Thus, for large $\epsilon$, the correction factor is ineffective and its use is not recommended. We also empirically observe the advantage of our algorithm with the correction factor in the experiments, particularly when $\epsilon$ is small.

### 4.3 Attempt 2: Accuracy Improvements via Dimension Reduction

Although Algorithm 1 enhances privacy and reduces the estimation error to be linear to $\sqrt{m}$, the error may still be excessively large since, in practice, $m$ is an enormous number, typically ranging from $10^4$ to $10^6$. To improve the accuracy of Algorithm 1, we consider applying dimensionality reduction. In particular, we adopt *random projection* [14], for two reasons: (i) random projection enables us to reduce the dimensionality of users' data without prior knowledge of the data; (ii) random projection has the restricted isometry property so that the distances between the points in a randomly selected subspace of suitably high dimension are approximately preserved [14].

For $q \ll m$, let $\Phi$ be a $q \times m$ random matrix whose elements $\phi_{kj}$ are independent random variables from Gaussian distribution or Bernoulli distribution with mean 0 and variance $1/q$. From the Johnson-Lindenstrauss lemma [14], we have

$$(1-\delta)\|r_i - Vu_i\|^2 \le \|\Phi(r_i - Vu_i)\|^2 \le (1+\delta)\|r_i - Vu_i\|^2$$

for $q = O(\delta^{-2}\log n)$. Since $(\Phi a)^T(\Phi b) = (\|\Phi(a+b)\|^2 - \|\Phi a\|^2 - \|\Phi b\|^2)/2$, we have

$$|(\Phi a)^T(\Phi b) - a^T b| \le \frac{3}{2}\delta(\|a\|^2 + \|b\|^2)$$

for any $a, b \in \mathbb{R}^m$. It follows that

$$\sum_{j=1}^{m} y_{ij}(r_{ij} - u_i^T v_j)^2 = r_i^T\Phi^T\Phi r_i - 2r_i^T\Phi^T\Phi V u_i + u_i^T V^T\Phi^T\Phi D_i\Phi^T\Phi V u_i + O(\delta),$$

where $D_i = \mathrm{diag}(y_{i1},\ldots,y_{im})$, $r_i = (r_{ij})_{1\le j\le m}$ and $V = (v_{jl}) = [v_1,\ldots,v_m]^T$ with the $d$-dimensional vector $v_j$. Letting $z_i = \Phi r_i = (z_{ik})_{1\le k\le q}$, $B = \Phi V$, and $W_i = \Phi D_i\Phi^T = (w_{i,kk'})_{1\le k,k'\le q}$, the problem of finding the item profiles $V$ becomes the problem of finding $B = [b_1,\ldots,b_q]^T$, where $b_k \in \mathbb{R}^d$, that minimizes

$$n^{-1}\sum_{i=1}^{n}\left\{z_i^T z_i - 2z_i^T B u_i + u_i^T B^T W_i B u_i\right\} + \lambda_v\sum_{k=1}^{q}\|b_k\|^2$$

for given $\{u_i \in \mathbb{R}^d, 1 \le i \le n\}$. Then, the update formula is

$$B^t = B^{t-1} - \gamma_t\cdot\left\{\nabla_B\psi(U^{t-1}, B^{t-1}) + 2\lambda_v B^{t-1}\right\}, \quad (9)$$

where $\nabla_B\psi(U,B) = -2n^{-1}\sum_{i=1}^{n}(z_i - W_i B u_i)u_i^T$.

Each user sends $\nabla_B^i = (z_i - W_i B u_i)u_i^T$ to the server for updating $B$. Since $\nabla_B^i = \Phi\nabla_V^i + O(\delta)$ and we use the same random matrix $\Phi$ for all the users, the server can easily recover a sparse matrix $\nabla_V^i$ whose elements are $-2y_{ij}u_{il}(r_{ij} - u_i^T v_j)$ from $\nabla_B^i$ using sparse recovery algorithms like compressive sensing. In that case, the aggregator can learn user's items and ratings. To avoid such issue, we suggest to send a noisy gradient for a randomly selected element from $q \times d$ matrix in a similar manner to Algorithm 1. The aggregator then collects all the users' gradients in the reduced dimension and updates $B$. These feedbacks between users and server go until $k$ iterations.

**Algorithm 2** Differentially Private Gradient Descent with Dimension Reduction

**input:** Positive integer $q$, predefined iteration number $k$, and privacy parameter $\epsilon$
**output:** Item profile matrix $V \in \mathbb{R}^{m\times d}$

Generate a $q\times m$ random matrix $\Phi$ whose entries are drawn from Gaussian distribution with mean 0 and standard deviation $1/\sqrt{q}$ and send $\Phi$ to users
Initialize $U$, $V$ and a counter $iter = 0$
**while** $iter \le k$ **do**
  Initialize $\nabla_B^* \in \{0\}^{q\times d}$
  **for** $i = 1$ to $n$ **do**
    Initialize $x_i^* \in \{0\}^{q\times d}$
    Derive $\nabla_V^i = \{-2y_{ij}u_i(r_{ij} - u_i^T v_j)\}_{1\le j\le m}$
    Compute $x_i = \Phi\nabla_V^i$
    Sample $s$ uniformly at random from $\{1,2,\ldots,q\}$
    Sample $l$ uniformly at random from $\{1,2,\ldots,d\}$
    If $(x_i)_{s,l} \notin [-1,1]$, project $(x_i)_{s,l}$ onto $[-1,1]$
    Draw $T \sim Bernoulli\left(\frac{(x_i)_{s,l}(e^{\epsilon/k}-1)+e^{\epsilon/k}+1}{2(e^{\epsilon/k}+1)}\right)$
    **if** $T = 1$ **then** $(x_i^*)_{s,l} = qd\frac{e^{\epsilon/k}+1}{e^{\epsilon/k}-1}$
    **else** $(x_i^*)_{s,l} = -qd\frac{e^{\epsilon/k}+1}{e^{\epsilon/k}-1}$
    **end if**
    Compute $\nabla_B^* = \nabla_B^* + x_i^*$
  **end for**
  Compute $\nabla_B^* = \nabla_B^*/n$ and send $\nabla_B^*$ to users
  $iter = iter + 1$
  **for** $i = 1$ to $n$ **do**
    Derive $\nabla_{u_i}$ in (4) and $u_i = u_i - \gamma_t\cdot\{\nabla_{u_i} + 2\lambda_u u_i\}$
    $V = V - \gamma_t\cdot\{\Phi^\dagger\nabla_B^* + 2\lambda_v V\}$
  **end for**
**end while**
**return** $V$

Once the aggregator obtains a final $B$ using noisy gradients submitted by users, the aggregator shares $B$ with users and the user device then computes $\hat{z}_i = Bu_i$. With the measurements $z_i$, the user device can predict the unobserved ratings $r_{ij}$ for $(i,j) \notin \mathcal{M}$ by sparse recovery algorithms. It is natural to assume sparsity in items preferred by each user. Thus, one can predict the ratings $r_{ij}, (i,j) \notin \mathcal{M}$ by solving the $l_1$-norm optimization problem:

$$\begin{aligned} &\text{minimize } \|x_i\|_1 \\ &\text{subject to } \hat{z}_i = \Phi x_i. \end{aligned} \quad (10)$$

In this case, the estimated ratings $r_{ij}, (i,j) \notin \mathcal{M}$ are given by compressive sensing algorithm.

From the experiments, we observe that a recovery of the ratings from $\hat{z}_i$ produces large errors when $q$ is smaller than the number of items rated by user. Since the sparsity level of ratings varies from user to user, setting optimal $q$ is not easy. Thus, we alternatively recover the gradient $\nabla_V \in \mathbb{R}^{m\times d}$

from $\nabla_B \in \mathbb{R}^{q\times d}$ since $\nabla_B \approx \Phi\nabla_V$. Although there is a recovery error in this procedure as well, we observe that the error in gradient is less sensitive, so the latter approach is more robust. When the server shares the initial $V$ with users, each user computes the gradient $\nabla_V^i$ and generates a $q \times d$ matrix $\nabla_B^i = \Phi\nabla_V^i$. Then, she sends $x_i^* = \nabla_B^{i,*}$ using Algorithm 1 by replacing $m$ with $q$ to the server. The server aggregates $x_i^*$, $i = 1, \ldots, n$, and shares $\nabla_B^* = \frac{1}{n}\sum_{i=1}^n x_i^*$ with users. Then, the full gradient $\nabla_V$ is recovered from $\nabla_B$ as follows:

$$\tilde{\nabla}_V^* = \Phi^\dagger \nabla_B^*, \tag{11}$$

where $\Phi^\dagger = \Phi^T(\Phi\Phi^T)^{-1}$ is the pseudo-inverse of $\Phi$. With the gradient $\tilde{\nabla}_V^*$, the item profile matrix $V$ can be updated and shared by the recommender. However, sharing a $m \times d$ matrix $V$ at every iteration increases communication cost. Thus, in our system, the recommender shares a $q \times d$ matrix $\nabla_B^*$ with users and each user recovers $\nabla_V$ from $\nabla_B$ in her device using (11). The item profile matrix $V$ is updated in each user's device and the recommendation is made with the final $V$. We summarize our proposal in Algorithm 2, and establish its privacy guarantees in the following.

***Corollary 1.*** Algorithm 2 satisfies $\epsilon$-differential privacy.

*Proof.* Let $x_i, x_i' \in [-1, 1]^{q\times d}$ be any two gradient matrices of user $i$. Also, let $x_i^*$ be a perturbed gradient matrix of user $i$ by Algorithm 2. As shown in Theorem 1, we have

$$\frac{Pr[x_i^* = v|x_i]}{Pr[x_i^* = v|x_i']} \leq e^{\epsilon/k}.$$

Now let $D$ and $D'$ be two datasets differing in the ratings of user $p$. For $w \in \mathbb{R}^{m\times d}$, observe that

$$\begin{aligned} Pr\left[\Phi^\dagger\nabla_B^{*,t} = w|D\right] &= Pr\left[\frac{1}{n}\sum_{i=1}^n x_i^{*,t} = \Phi w\middle|D\right] \\ &= \sum_{\substack{\tilde{w}_1,\ldots,\tilde{w}_n \\ \tilde{w}_1+\cdots+\tilde{w}_n = n\Phi w}} \prod_{i=1}^n Pr\left[x_i^{*,t} = \tilde{w}_i|x_i\right], \end{aligned}$$

where the probability space is over the coin flips of the binary randomization in Algorithm 2. Note that a random matrix $\Phi$ is generated by a server and then shared with users, so we treat $\Phi$ as a given matrix. As in Theorem 1, for the final item profile $V$, we have

$$\begin{aligned} \frac{Pr[V = \bar{V}|D]}{Pr[V = \bar{V}|D']} &\leq \max_{a_1,\ldots,a_k} \prod_{t=1}^k \frac{Pr[\Phi^\dagger\nabla_B^{*,t} = a_t|D]}{Pr[\Phi^\dagger\nabla_B^{*,t} = a_t|D']} \\ &\leq \max_{a_1,\ldots,a_k} \prod_{t=1}^k \max_{\tilde{w}_p^t} \frac{Pr\left[x_p^{*,t} = \tilde{w}_p^t|x_p\right]}{Pr\left[x_p^{*,t} = \tilde{w}_p^t|x_p'\right]} \\ &\leq e^\epsilon. \quad \square \end{aligned}$$

***Corollary 2.*** Suppose that the learning rate $\gamma_t$ decreases as iteration goes and satisfies $\sum_{t=1}^\infty \gamma_t^2 < \infty$. Let $V^*$ be the item profile matrix derived by Algorithm 2 and $V$ the item profile matrix in non-private manner after $k$ iterations. With at least $1 - \beta$ probability,

$$\|V^* - V\|_{\max} = O\left(\frac{qd\sqrt{\log(qd/\beta)}}{(\epsilon/k)\sqrt{n}}\right).$$

*Proof.* Observe that $V^* - V = -\sum_{t=1}^k \gamma_t c_t(\Phi^\dagger\nabla_B^{*,t} - \nabla_V^t)$ with $c_t = \prod_{j=t+1}^k(1 - 2\lambda_v\gamma_j)$ for $1 \leq t < k$ and $c_k = 1$. From the Johnson-Lindenstrauss lemma [14], we have

$$\begin{aligned} &\|\sum_{t=1}^k \gamma_t c_t(\Phi^\dagger\nabla_B^{*,t} - \nabla_V^t)\|_{\max} \\ &\leq \|\sum_{t=1}^k \gamma_t c_t(\Phi^\dagger\nabla_B^{*,t} - \nabla_V^t)\|_F \\ &\leq (1-\delta)^{-1/2}\|\sum_{t=1}^k \gamma_t c_t(\nabla_B^{*,t} - \Phi\nabla_V^t)\|_F \\ &\leq (1-\delta)^{-1/2}\sqrt{qd}\|\sum_{t=1}^k \gamma_t c_t(\nabla_B^{*,t} - \Phi\nabla_V^t)\|_{\max}. \end{aligned}$$

Now note that the random variables $(\nabla_B^{*,t})_{j,l}, 1 \leq t \leq k$, are independent and bounded by $O\left(\frac{qd}{\epsilon/k}\right)$, $(\nabla_B^{*,t})_{j,l}$ is an unbiased estimator of $(\Phi\nabla_V^t)_{j,l}$, and $\mathrm{Var}((\nabla_B^{*,t})_{j,l}) = O\left(\frac{qd}{(\epsilon/k)^2 n}\right)$. Similarly to the proof of Theorem 2, by Bernstein's inequality, we have

$$\begin{aligned} &Pr\left[\|\sum_{t=1}^k \gamma_t c_t(\nabla_B^{*,t} - \Phi\nabla_V^t)\|_{\max} > \lambda \,\middle|\, D\right] \\ &\leq 2 \cdot qd \cdot \exp\left(-\frac{\lambda^2}{2\sum_{t=1}^k \gamma_t^2 c_t^2 \mathrm{Var}((\nabla_B^{*,t})_{j,l}) + \frac{2}{3}\lambda qd\frac{e^{\epsilon/k}+1}{e^{\epsilon/k}-1}}\right) \\ &= O\Big(qd \cdot \exp\big(-n\lambda^2(\epsilon/k)^2/qd\big)\Big). \end{aligned}$$

Taking $\lambda = O\left(\frac{\sqrt{qd\log(qd/\beta)}}{(\epsilon/k)\sqrt{n}}\right)$, we have

$$\|\sum_{t=1}^k \gamma_t c_t(\nabla_B^{*,t} - \Phi\nabla_V^t)\|_{\max} = O\left(\frac{\sqrt{qd\log(qd/\beta)}}{(\epsilon/k)\sqrt{n}}\right),$$

so the desired rate for $\|V^* - V\|_{\max}$ is achieved. $\square$

Corollary 1 and 2 state that our matrix factorization algorithm guarantees both per-user privacy and utility. In addition, our algorithm is much more efficient than the existing work. In our algorithm, each user transmits one bit to the server. Further, we reduce the communication cost from the server to users with the use of random projection. In the existing multiparty computations between the server and users for matrix factorization [13], the server needs to send a $m \times d$ matrix $V$ of item profiles to users at each iteration. On the other hand, our algorithm requires the server to send a $q \times d$ matrix $\nabla_B$ of noisy gradients to users. Taking $q = O(\log m)$, our algorithm lowers communication cost drastically.

As discussed in Section 4, for small $\epsilon$, we can further improve the recommendation accuracy by using the correction factor while $\epsilon$-differential privacy is still satisfied. Note that $\|V^* - V\|_{\max} = O\left(\frac{qd\sqrt{\log(qd/\beta)}}{k\epsilon\sqrt{n}}\right) + O(1)$ when applying correction factor $\frac{1}{k^2}$ to Algorithm 2. Since $\log(n) \approx \log(m)$ in a typical recommender system, we can estimate the item profiles with the error up to nearly $O(\log(m)d/(k\epsilon\sqrt{n}))$.

## 5 EXPERIMENTS

We now assess the performance of our proposed algorithms. We evaluate our two algorithms that guarantee per-user privacy: (i) differentially private gradient descent (private GD); (ii) differentially private gradient descent with dimensionality reduction (private GD-DR). We additionally evaluate a non-private version of GD (non-private GD) for baseline and Hua et al's method for comparison. To evaluate the prediction accuracy, we compute prediction RMSE by 10-fold cross validation. We measure prediction RMSEs over various privacy budgets $\epsilon$ and various numbers $k$ of iterations. Remark that we have tested the algorithm in Section 4.1 with $p = 2/(1+e^{\epsilon/(2km)})$ and $\sigma = \frac{m\Delta d}{\epsilon/(2k)}$ as well, but found that it diverges even when $\epsilon$ is large, and hence, we omit it from the figures.

### 5.1 Experimental Setup

**Datasets.** We use two datasets: MovieLens [12] and LibimSeTi [5]. Among several versions of MovieLens, we use the version that consists of 20M ratings of 26,744 movies submitted by 138,493 users. The range of ratings is 0.5 to 5. The LibimSeTi dating recommendation dataset used for our analysis contains 13,689,250 ratings of 26,509 profiles made by 135,359 users[1]. The ratings for this data are on a 1-10 scale.

**Choice of parameters.** For both MovieLens and LibimSeTi, we set $\lambda_u = \lambda_v = 10^{-8}$ for all methods and $q = 2700$ for dimension reduction. We use $d = 15, 20$ for low rank approximation of MovieLens and LibimSeTi, respectively. We set a learning rate $\gamma_t$ to $O(1/t)$ at iteration $t$. We consider the values for privacy budget $\epsilon$ from 0.05 to 1.6 and various values for the number $k$ of iterations.

**Comparisons with competitors.** This paper is the first work to protect the entire ratings and items of a single user (i.e., guarantee per-user privacy). On the other hand, Hua et al. [13] and Shen and Jin [26], [27] provide privacy for a single rating or a single item of a single user. In addition, our system guarantees $\epsilon$-differential privacy for each user over $k$ iterations while Hua et al. [13] supports $\epsilon$-differential privacy over two consecutive iterations. As summarized in Table 2, our system pursues much stronger privacy than the existing solutions [13], [26], [27]. Therefore, there is no existing work to fairly compare with ours. Nonetheless, we compare ours with Hua et al. [13] to show how good the performance of our approach is, even under stronger privacy requirements. Note that we do not compare with Shen and Jin [26], [27] because their approaches require additional information on items (public category set for items), which may not always exist. For example, for online dating recommendation, an individual user corresponds to an item and there is no publicly available category set for individual users.

1. The original LibimSeTi dataset contains 17,359,346 anonymous ratings of 168,791 profiles made by 135,359 users. Since the original LibimSeTi data is very sparse with 0.07% density, we chose the profiles having more than 140 ratings so that density of the dataset is 0.38%.

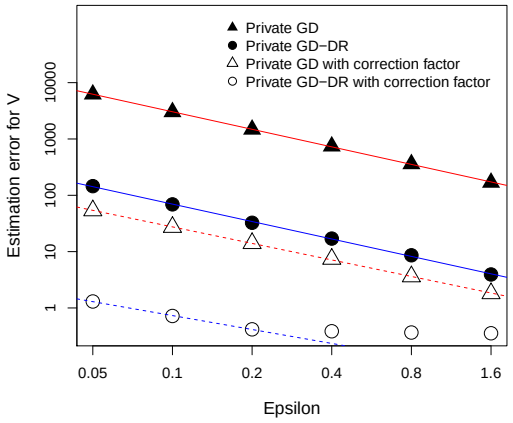



Fig. 2. MoviLens: Estimation errors $\|V^* - V\|_{\max}$ for Private GD and Private GD-DR without and with correction factor.

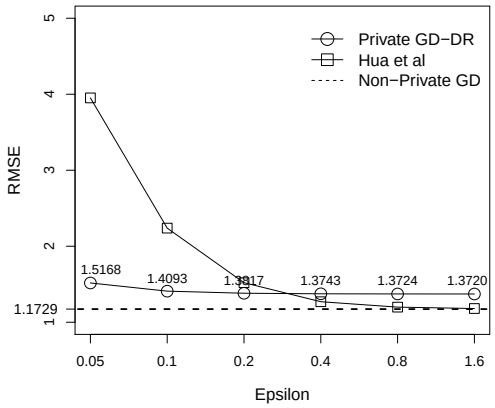
### 5.2 Evaluation Results

**Effects of dimensionality reduction and correction factor.** The number of items in both MovieLens and LibimSeTi datasets is quite large. To demonstrate the effects of dimensionality reduction and the correction factor $1/k^2$ on accuracy of our private gradient descent algorithms, we run private GD and private GD-DR without correction factor and with correction factor and measure the estimation error of $V^*$ due to the perturbation, given by $\|V^* - V\|_{\max} = \max_{j,l} |v^*_{jl} - v_{jl}|$. For this, we use the entire MovieLens with 20M ratings. Figure 2 displays the estimation errors of the resulting item profiles after 10 iterations. From Figure 2, we observe that the estimation errors $\|V^* - V\|_{\max}$ are inversely proportional to the privacy budget $\epsilon$ with the slope values $-1.035$ (red solid line), $-0.975$ (red dotted line), and $-1.028$ (blue solid line) for private GD (Algorithm 1) without and with correction factor and private GD-DR (Algorithm 2) without correction factor, respectively. However, for private GD-DR with correction factor, the estimation errors $\|V^* - V\|_{\max}$ reduce linearly over $\epsilon$ with the slope value $-0.821$ (blue dotted line) until $\epsilon = 0.2$ and then no significant improvement in the accuracy of $V$ estimation is made although the privacy budget $\epsilon$ increases. This phenomenon for private GD-DR (Algorithm 2) with correction factor agrees with our theoretical observation $\|V^* - V\|_{\max} = O\left(\frac{qd\sqrt{\log(qd/\beta)}}{k\epsilon\sqrt{n}}\right) + O(1)$. We also observe that private GD produces unsatisfactory accuracy while private GD-DR improves accuracy drastically by reducing the input dimension. In addition, we observe that the correction factor significantly improves the accuracy of item profiles. All of these agree with our theoretical observations in Section 4.

**Prediction accuracy over various privacy budgets.** We measure prediction RMSEs by 10-fold cross validation for both MovieLens and LibimSeTi over various privacy budgets $\epsilon$. Since Figure 2 demonstrates that the correction factor $1/k^2$ greatly improves the accuracy, we implement private GD-DR after including the correction factor to Algorithm 2. Hereafter we call it private GD-DR. For comparison, we implement the method proposed by Hua et al. [13] as well.

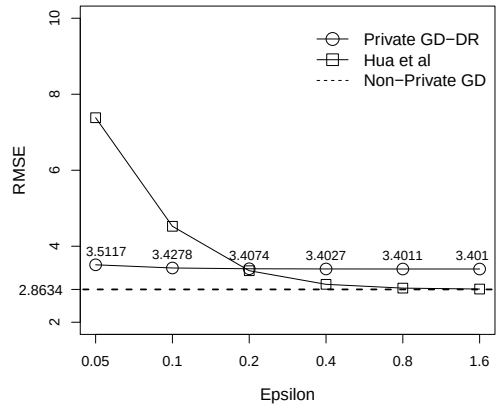
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2805356, IEEE Transactions on Knowledge and Data Engineering






| | $\epsilon$ | | | | |
|---|---|---|---|---|---|
| | 0.1 | 0.2 | 0.4 | 0.8 | 1.6 |
| Non-Private GD | 1.1729 | | | | |
| Private GD-DR | 1.4093 | 1.3817 | 1.3743 | 1.3724 | 1.3720 |
| Hua et al | 2.2385 | 1.5203 | 1.2702 | 1.1989 | 1.1806 |

Fig. 3. MovieLens - Prediction RMSEs for Private GD-DR and Hua et al.'s method. The values given in the table are RMSEs for Non-Private GD, Private GD-DR, and Hua et al.'s method over various $\epsilon$ values. Remark that private GD-DR satisfies $\epsilon$-differential privacy for the item profile matrix $V$ and $\epsilon$-LDP for a user's entire data, whereas Hua et al.'s method satisfies $m\epsilon$-differential privacy for $V$ and $10m\epsilon$-LDP for a user's entire data.




| | $\epsilon$ | | | | |
|---|---|---|---|---|---|
| | 0.1 | 0.2 | 0.4 | 0.8 | 1.6 |
| Non-Private GD | 2.8634 | | | | |
| Private GD-DR | 3.4278 | 3.4074 | 3.4027 | 3.4011 | 3.4010 |
| Hua et al | 4.5249 | 3.3563 | 2.9978 | 2.8980 | 2.8735 |

Fig. 4. LibimSeTi - Prediction RMSEs for Private GD-DR and Hua et al's method. The values given in the table are RMSEs for Non-Private GD, Private GD-DR, and Hua et al.'s method over various $\epsilon$ values. Remark that private GD-DR satisfies $\epsilon$-differential privacy for the item profile matrix $V$ and $\epsilon$-LDP for a user's entire data, whereas Hua et al.'s method satisfies $m\epsilon$-differential privacy for $V$ and $10m\epsilon$-LDP for a user's entire data.

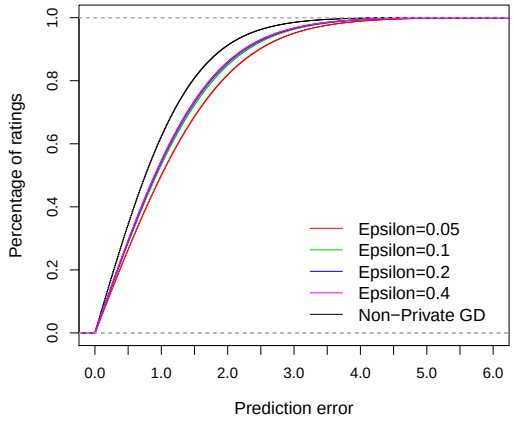



Fig. 5. MovieLens - CDFs of the prediction errors $|\hat{r}_{ij} - r_{ij}|$ for Private GD-DR.

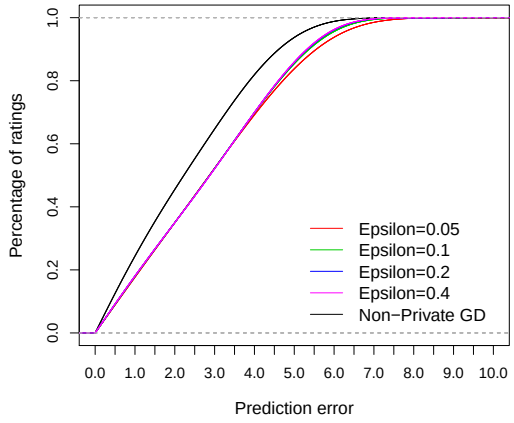



Fig. 6. LibimSeTi - CDFs of the prediction errors $|\hat{r}_{ij} - r_{ij}|$ for Private GD-DR.

Figures 3 and 4 display the resulting prediction RMSEs after running $k = 10$ iterations. For both datasets, the prediction performance of our algorithm is much better than Hua et al.'s method when the privacy budget $\epsilon$ is very small and quite competitive when $\epsilon \geq 0.4$, although our privacy requirement is much stronger. Note that our algorithm guarantees $\epsilon$-differential privacy for the item profile matrix $V$ and $\epsilon$-LDP for a user's entire data, whereas Hua et al.'s method provides $m\epsilon$-differential privacy for $V$ and $mk\epsilon$-LDP for a user's entire data. Recall that, while Hua et al.'s method guarantees privacy for a user's single rating and privacy for gradients over two consecutive iterations, our algorithm guarantees not only privacy for a user's entire ratings and set of items, but also privacy for gradients over whole iterations. Remark that the bias caused by the correction factor dominates the estimation error as $\epsilon$ increases, as shown in Figure 2. This is why the performance of our algorithm is not significantly improved as $\epsilon$ increases from 0.2 to 1.6. Also, note that the RMSE for LibimSeTi is larger, because its rating scale is larger than that of MovieLens, and its rating matrix is sparser than that of MovieLens.

To see how accurately the ratings are predicted by our algorithm, we also compute the prediction errors of ratings from 10-fold cross validation and draw its cumulative distribution function (CDF) in Figures 5 and 6. Note that, for 10-fold cross validation, we partition the dataset into 10 parts, where 9 parts (training set) are used for estimating profiles $U$ and $V$ and one part (test set) is used for evaluating the prediction performance. After training and testing 10 times, the prediction errors of ratings are given by $|\hat{r}_{ij} - r_{ij}|$ for

$(i, j) \in \mathcal{M}$.

To evaluate the recommender systems, we additionally measure utility in terms of F-score for top 10 set. F-score is the harmonic mean of precision and recall. We calculate it by comparing top 10 items recommended by a differentially private matrix factorization algorithm (e.g., private GD-DR) to top 10 items chosen by matrix factorization algorithm on the raw data (i.e., non-private GD). Figures 7 and 8 show the values for F-score for MovieLens and LibimSeTi datasets. For MovieLens dataset, F-score gets larger as $\epsilon$ increases, meaning that the items recommended by private GD-DR or Hua et al.'s method become more similar to the items recommended by non-private GD as privacy level is lower. On the other hand, for LibimSeTi dataset, F-scores of private GD-DR are almost the same over $\epsilon$ – it grows slowly for $\epsilon \leq 0.2$ and then stays the almost same value for $\epsilon > 0.2$. The different performances of private GD-DR over two datasets in terms of $F$-score are somewhat expected from Figures 5 and 6, where the prediction errors for MovieLens dataset decrease faster than those for LibimiSeTi dataset as $\epsilon$ increases. Note again that, although Hua et al.'s method performs better than our algorithm for larger $\epsilon$, our algorithm guarantees much stronger privacy than Hua et al.'s method does.

**Prediction accuracy over various numbers of iterations.** We additionally evaluate prediction RMSEs of Algorithm 2 over various numbers of iterations from 1 to 50, displayed in Figure 9. Without the correction factor, for fixed $\epsilon$, RMSE grows as the number $k$ of iterations increases. This is because $\epsilon$ is the total privacy budget over $k$ iterations and so the privacy budget at each iteration is $\epsilon/k$ in order to protect all the information about ratings, set of items rated by user, and user profiles across $k$ iterations. As explained in Section 4, the amount of noise added to the gradients increases as $k$ gets larger. This makes the gradient descent algorithm useless. One may increase the total privacy budget as $k$ increases, but this causes privacy breach as iteration goes. Our algorithm with the correction factor resolves this issue and achieves both privacy and utility. Figure 9 shows that RMSE decreases as iteration goes when $\epsilon = 0.1$. We observe a similar result with LibimSeTi dataset.

**Communication cost.** We analyze the communication cost of private GD-DR and Hua et al.'s method at each iteration in Table 3. For private GD-DR, each user always transmits one bit to server regardless of how many items she rates. For Hua et al's method, each user's uploading data sizes are about 8KB in average for both MovieLens and LibimSeTi. Server transmits the updated $V$ of $dm$ elements to each user in Hua et al's method, which are about 1.5MB for MovieLens and 2MB for LibimSeTi. In private GD-DR server transmits its reduced form of $dq$ elements with $q = O(\log m)$, which are about 0.15MB for MovieLens and 0.2MB for LibimSeTi with $q = 2700$.

TABLE 3
Communication Cost at Each Iteration

| | a user to server | server to a user |
|---|---|---|
| Private GD-DR | 1 bit | $O(d \log m)$ |
| Hua et al. | $O(dm_i)$ | $O(dm)$ |

$m_i$: the number of items rated by user $i$

## 6 Related Work

Differential privacy [9] is a strong and mathematically rigorous privacy standard. It aims to ensure that the output of the algorithm does not significantly depend on any particular user's data. Earlier models of differential privacy were developed for the trusted server setting, but more recent research increasingly considers local differential privacy (LDP) due to its stronger privacy protection.

Erlingsson et al. [10] propose RAPPOR to collect and analyze users' data without privacy violation. RAPPOR builds on the concept of randomized response, a surveying technique developed by Warner [28] for collecting statistics on sensitive topics. The key idea of RAPPOR is to perturb the user's data before sending them to Google such that the perturbation algorithm satisfies differential privacy. However, RAPPOR is limited to simple data analysis such as counting. It cannot compute aggregates on numeric attributes, or handle sophisticated analysis such as regression, classification, clustering and matrix factorization. Moreover, the accuracy of RAPPOR deteriorates quickly when the number of attributes increases.

Recently, Nguyen et al. [21] develop practical LDP solutions that go beyond RAPPOR. Their randomization mechanisms enable analysis of both categorical and numerical data as well as machine learning tasks based on stochastic gradient descent (SGD). Our solution is built upon Nguyen et al.'s technique for SGD, and we make contributions in three aspects. First, we provide a rigorous analysis of the error in the item profile matrix (see Theorem 2 and Corollary 2) produced by our solution, whereas Nguyen et al. do not conduct any theoretical analysis on the accuracy of the their SGD method. Second, based on our analysis, we propose a variance reduction technique using a correction factor in the learning rate (see Sections 4.2 and 4.3), which is not considered by Nguyen et al. Third, we utilize our SGD algorithm to build a recommendation system and comprehensively evaluate its performance on benchmark datasets; in contrast, Nguyen et al. consider a generic setting of SGD without demonstrating on how their method can be used for recommendation.

There are also recent works that adopt local privacy models in constructing recommendation systems. Polat et al. [22] and Xin and Jaakkola [30] deal with untrusted recommenders by introducing randomized mechanisms to perturb user data locally. However, their methods do not satisfy differential privacy. Shen and Jin [26], [27] propose a differentially private perturbation algorithm to inject instance-based noises so that the set of user's items is protected. However, their approach cannot protect users' category preference such as genre. Hua et al. [13] propose a gradient perturbation algorithm for differentially private matrix factorization to prevent an untrusted recommender from learning any user's ratings or profiles. Similar to ours, they suggest a recommender system that decouples computations upon users' private data from the recommender to users, and makes the recommender aggregate local results in a privacy-preserving manner. However, Hua et al. [13] only aim to protect users' ratings, which motivate us to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2805356, IEEE Transactions on Knowledge and Data Engineering



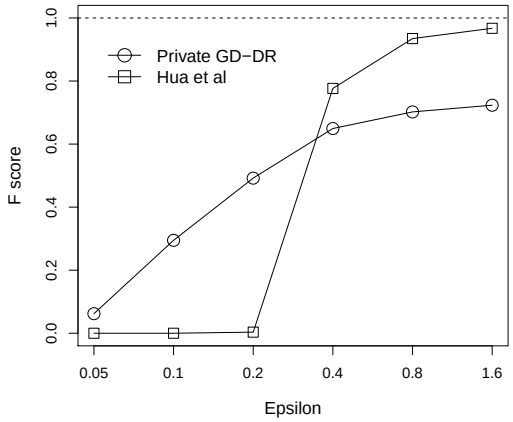



Fig. 7. MovieLens - F-score for Private GD-DR and Hua et al.'s method. Remark that private GD-DR satisfies $\epsilon$-differential privacy for the item profile matrix $V$ and $\epsilon$-LDP for a user's entire data, whereas Hua et al.'s method satisfies $m\epsilon$-differential privacy for $V$ and $10m\epsilon$-LDP for a user's entire data.

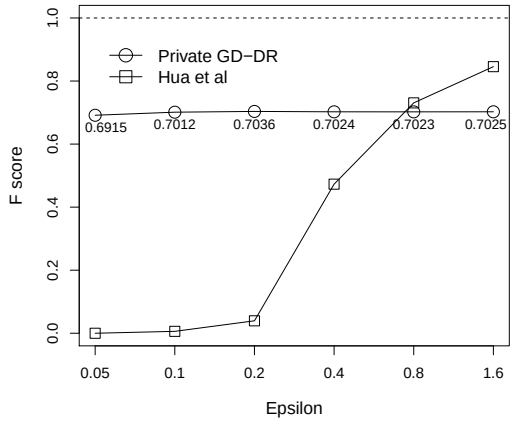
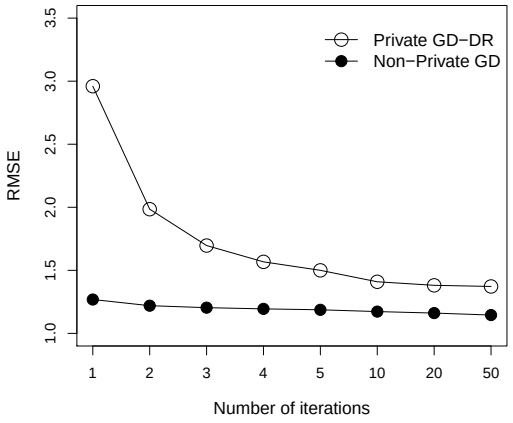



Fig. 8. LibimSeTi - F-score for Private GD-DR and Hua et al.'s method. Remark that private GD-DR satisfies $\epsilon$-differential privacy for the item profile matrix $V$ and $\epsilon$-LDP for a user's entire data, whereas Hua et al.'s method satisfies $m\epsilon$-differential privacy for $V$ and $10m\epsilon$-LDP for a user's entire data.




| | $k$ | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 3 | 5 | 10 | 20 | 50 |
| Non-Private GD | 1.269 | 1.204 | 1.187 | 1.173 | 1.161 | 1.145 |
| Private GD-DR | 2.960 | 1.697 | 1.500 | 1.409 | 1.381 | 1.372 |

Fig. 9. MovieLens - Prediction RMSEs of Private GD-DR over $k$ iterations when $\epsilon = 0.1$. The values in the table are RMSEs for Private GD-DR and Non-Private GD.

develop a private method to protect both the set of items and ratings.

More generally, there are recent works applying LDP to machine learning algorithms [8] and histogram construction [3], [7], [25]. Duchi et al. [8] propose a randomization mechanism that reduces the effect of a large number of attributes and investigate the tradeoff between privacy guarantees and the utility of the resulting statistical estimators under LDP based on the information theory. Bassily and Smith [3] propose an optimal mechanism for frequency estimations over a large categorical domain under LDP. Qin et al. [25] show that Bassily and Smith's mechanism can be combined with the classic randomized response to achieve improved accuracy for heavy hitter estimation over set-valued data. Kairouz et al. [15] also study the fundamental tradeoff between local differential privacy and information theoretic utility functions. However, these works have focused only on theoretical investigation. Chen et al. [7] extend Bassily and Smith's method to the private spatial data aggregation problem for guaranteeing personalized local differential privacy.

Finally, there are several works on differentially private recommender systems assuming trusted servers. McSherry and Mironov [19] develop a differentially private method for collaborative filtering algorithms. Specifically, they apply differential privacy to count, average, and covariance matrix perturbations. Liu et al. [17] develop a differentially private matrix factorization algorithm by using Bayesian posterior sampling via Stochastic Gradient Langevin Dynamics and enhances privacy by guaranteeing per-user privacy and calibrating each user's privacy. Similarly to Liu et al. [17], Balu and Furon [2] apply Bayesian posterior sampling approach to develop a differentially private matrix factorization algorithm assuming a trusted recommender.

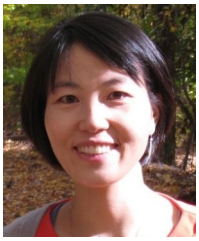
## 7 Conclusion

We develop novel matrix factorization algorithms under local differential privacy (LDP). The proposed algorithms enhance privacy by guaranteeing per-user privacy and completely protecting users' items and ratings. We overcome the limitation of the private matrix factorization framework under LDP due to high dimensionality, by incorporating dimensionality reduction techniques and the randomization mechanism of Nguyen et al. [21]. Since the proposed randomization mechanisms request each user to transmit only one bit to a server, the communication cost from users to the server is drastically reduced. In addition, dimensionality reduction even reduces communication overhead from the server to users.

## References

[1] https://www.wired.com/2016/06/apples-differential-privacy-collecting-data/
[2] R. Balu and T. Furon. Differentially private matrix factorization using sketching techniques. In *IH&MMSec*, pages 57–62, 2016.

[3] R. Bassily and A. Smith. Local, private, efficient protocols for succinct histograms. In *STOC*, pages 127–135, 2015.
[4] A. Berlioz, A. Friedman, M. A. Kaafar, R. Boreli, and S. Berkovsky. Applying differential privacy to matrix factorization. In *RecSys*, pages 107–114, 2015.
[5] L. Brozovsky and V. Petricek. Recommender system for online dating service. In *Znalosti*, 2007. (Available: http://www.occamslab.com/petricek/data/)
[6] J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten, and V. Shmatikov. "You might also like": privacy risks of collaborative filtering. In *IEEE S&P*, pages 231–246, 2011.
[7] R. Chen, H. Li, A. K. Qin, S. P. Kasiviswanathan, and H. Jin. Private spatial data aggregation in the local setting. In *ICDE*, pages 289–300, 2016.
[8] J. C. Duchi, M. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *FOCS*, pages 429–438, 2013.
[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.
[10] U. Erlingsson, V. Pihur, and A. Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, pages 1054–1067, 2014.
[11] G. G. Fanti, V. Pihur, and U. Erlingsson. Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries. *PoPETS*, 3, 2016.
[12] F. M. Harper and Joseph A. Konstan. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5, 1–19, 2015.
[13] J. Hua, C. Xia, and S. Zhong. Differentially private matrix factorization. In *IJCAI*, pages 1763–1770, 2015.
[14] W. B. Johnson and J. Lindenstrauss. Extensions of Lipshitz mapping into Hilbert space. In *Conference in modern analysis and probability, volume 26 of Contemporary Mathematics*, pages 189–206, 1984.
[15] P. Kairouz, S. Oh, and P. Viswanath. Extremal mechanisms for local differential privacy. In *NIPS*, pages 2879–2887, 2014.
[16] S. K. Lam, D. Frankowski, and J. Riedl. Do you trust your recommendations? An exploration of security and privacy issues in recommender systems. In *ETRICS*, pages 14–29, 2006.
[17] Z., Liu, Y.-X. Wang, and A. J. Smola. Fast differentially private matrix factorization. In *RecSys*, pages 171-178, 2015.
[18] A. Machanavajjhala, A. Korolova, and A. D. Sarma. Personalized social recommendations: accurate or private. *Proceedings of the VLDB Endowment*, 4(7), pages 440–450, 2011.
[19] F. McSherry and L. Mironov. Differentially private recommender systems: building privacy into the Netflix prize contenders. In *SIGKDD*, pages 627–636, 2009.
[20] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE S&P*, pages 111–125, 2008.
[21] T. Nguyen, X. Xiao, Y. Yang, S. C. Hui, H. Shin, and J. Shin, Collecting and analyzing data from smart device users with local differential privacy. arXiv:1606.05053 [cs.DB], 2016.
[22] H. Polat and W. Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *ICDM*, pages 625-628, 2003.
[23] A. Rajkumar and S. Agarwal. A differentially private stochastic gradient descent algorithm for multiparty classification. In *AISTATS*, pages 933–941, 2012.
[24] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22, 400–407, 1951.
[25] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren. Heavy Hitter Estimation over Set-Valued Data with Local Differential Privacy. In *CCS*, 2016.
[26] Y. Shen and H. Jin, Privacy-preserving personalized recommendation: An instance-based approach via differential privacy. In *ICDM*, pages 540–549, 2014.
[27] Y. Shen and H. Jin, EpicRec: Towards practical differentially private framework for personalized recommendation. In *CCS*, pages 180–191, 2016.
[28] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60, 63–69, 1965.
[29] U. Weinsberg, S. Bhagat, S. Ioannidis, and N. Taft. Blurme: inferring and obfuscating user gender based on ratings. In *RecSys*, pages 195–202, 2012.
[30] Y. Xin and T. Jaakkola. Controlling privacy in recommender systems. In *NIPS*, pages 2618–2626, 2014.



**Hyejin Shin** received the PhD Degree in Statistics from Texas A&M University in 2006. She is currently a principal engineer at Samsung Research. Before joining Samsung, she was an assistant professor in the Department of Mathematics and Statistics at Auburn University and a Member of Technical Staff at Bell Labs. Her research interests include statistical modeling, statistical machine learning, and data privacy.

**Sungwook Kim** received the Ph.D. degree in mathematics at Seoul National University (SNU) in 2012. He is currently a senior engineer at Samsung Research. His research interests include computational number theory, cryptography, and information security and privacy.

**Junbum Shin** received the Ph.D. degree in computer science from Korea Advanced Institute of Science and Technology in 2003. He is currently a principal engineer at Samsung Research. His research interests include security protocol, data privacy, data and software protection, system security, cryptography, and formal method.

**Xiaokui Xiao** received the PhD degree in computer science and engineering from the Chinese University of Hong Kong in 2008. He is currently an Associate Professor at the Nanyang Technological University (NTU), Singapore. Before joining NTU in 2009, he was a postdoctoral associate at the Cornell University. His research interests include data privacy, spatial databases, graph databases, and parallel computing.