Information Sciences 482 (2019) 248–265

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Towards a more reliable privacy-preserving recommender system

Jia-Yun Jiang[a,b], Cheng-Te Li[c,d,*], Shou-De Lin[a]

[a] Department of Computer Science and Information Engineering, National Taiwan University, Taiwan
[b] Institute of Information Science, Academia Sinica, Taiwan
[c] Institute of Data Science, National Cheng Kung University, Taiwan
[d] Department of Statistics, National Cheng Kung University, Taiwan

ARTICLE INFO

Article history:
Received 17 November 2017
Revised 29 December 2018
Accepted 31 December 2018
Available online 3 January 2019

Keywords:
Privacy-preserving recommendation
Differential privacy
Secure distributed matrix factorization
Randomized response algorithms

ABSTRACT

This paper proposes a privacy-preserving distributed recommendation framework, *Secure Distributed Collaborative Filtering* (SDCF), to preserve the privacy of value, model and existence altogether. That says, not only the ratings from the users to the items, but also the existence of the ratings as well as the learned recommendation model are kept private in our framework. Our solution relies on a distributed client-server architecture and a two-stage Randomized Response algorithm, along with an implementation on the popular recommendation model, Matrix Factorization (MF). We further prove SDCF to meet the guarantee of *Differential Privacy* so that clients are allowed to specify arbitrary privacy levels. Experiments conducted on numerical rating prediction and one-class rating action prediction exhibit that SDCF does not sacrifice too much accuracy for privacy.

1. Introduction

Collaborative filtering (CF) is one of the most popular models for recommending items [27]. Its basic idea is to make recommendation based on the similarity between users or between items. CF-based models are trained by making use of user feedbacks on items. CF models can be divided into two categories of feedback: numerical and one-class. Numerical feedback consists of numeric values (e.g., ratings between 1 and 5). One-class feedback is a kind of record/action for a specific action (e.g., purchase an item or not). As the training process of CF-based models relies on large-scale data, service providers should collect a huge collection of user feedback records. However, if the servers are untrusted or contain vulnerabilities, the collection of user feedbacks may lead to privacy liability due to data leakage. Even if the servers are curious-but-honest, which means the services are functioning normally, feedback data leakage can still make private attributes and even real identity of users be inferred by hackers [7,10]. There are three categories of data leakage in CF-based recommenders. The first is *Value Leakage*: exposure of the values of feedbacks, such as the rating scores. The second is *Existence Leakage*: exposure of the existence of feedbacks, such as whether a user rates an item. For example, if the attacker knows that user u rates a book i, they can be quite certain that u has read i. The third is *Model Leakage*: exposure of the trained model. Models are important since given the CF model, attackers can estimate the ratings from any user to any item.

[*] Corresponding author.
E-mail address: chengte@mail.ncku.edu.tw (C.-T. Li).

https://doi.org/10.1016/j.ins.2018.12.085
0020-0255/© 2019 Elsevier Inc. All rights reserved.

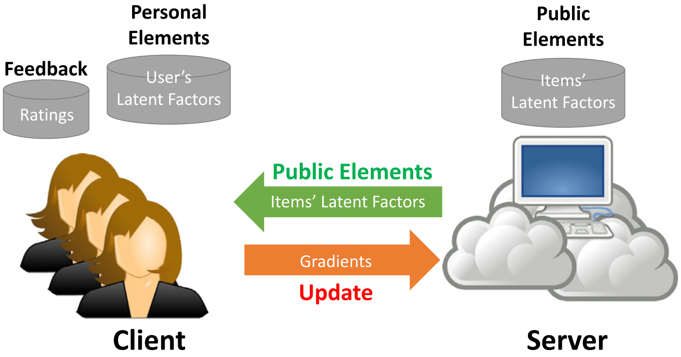


Fig. 1. Secured distributed architecture using MF as an example.

In this paper, we propose a client-server framework, termed *Secure Distributed Collaborative Filtering* (SDCF), to preserve user privacy in CF-based recommendation systems. We elaborate the idea of SDCF in Fig. 1 that uses Matrix Factorization (MF) as an example. First, we separate the *model* into two disjoint parts: personal elements and public elements. Personal elements are user-specific while public elements are those common to every user, such as the item information. We propose to store only public elements at server side, and users keep both personal ratings and personal elements at client side. Such setting is able to avoid *value* and *model* leakage. Apparently not all CF-based models meet such requirement. As will be described later, MF satisfies such condition. The next challenge is how to train a model on aforementioned structure since both parts of elements need to be updated frequently in the training process. Our idea is that during the training, the policy that users download public elements is based on which users are beneficial in locally updating personal elements. Then users send required information to the server so that public elements can be updated. This distributed mechanism naturally prevents both *value* and *model* leakages, because users no longer send their ratings to the server. In addition with only public elements, the server cannot perform recommendation solely. To further prevent the *existence* leakage, we propose a two-stage *Randomized Response algorithm* in SDCF. A strength of SDCF is the guarantee of *Differential Privacy* (DP) [10,11], which is a kind of privacy quantification used in various domains. Equipped With DP, SDCF allows users to adjust their privacy levels based on their own privacy budgets.

Different strategies are leveraged in past studies on privacy-preserving recommender systems. For example, Canny proposed a peer-to-peer network that allows personalized recommendation to be generated by other members without disclosing individual user data [8]. McSherry et al. first applied differential privacy (DP) to CF [22] and is followed by several studies [1,5,6,13,15,19–21,29,33]. Besides, Vallet et al. [30] proposed a decentralized framework of matrix factorization without the need of retaining data at server side. Among these studies, some require servers to trust the training stages [5,20–22,30], and some consider the scenario of completely untrusted servers as our work. [3,8,15,23,29,33]. Nevertheless, most of them rely on particular conditions or resources. For instance, Shen et al. required category information of items [29], Xin et al.'s work relied on a small group of users to be public [33], and the approaches proposed by Nikolaenko et al. [23] Canny [8] need laborious computation for cryptography algorithms. Boutet et al. [6] and Afsharinejad et al. [1] considered to use obfuscated user profiles for k-nearest neighbors, which preserve *existence* privacy; Li et al. [19] improved the performance of collaborative filtering by sharing sanitized data between users. However, in these three studies, the privacy of trained models is not completely protected. The work of Hua et al. is probably the most similar to our solution [15]. It assumes that nothing other than the ratings is considered, and preserves the privacy of *value* and *model* by a distributed gradient-transmission architecture. Nevertheless, it requires *third-party* resources, i.e., additional servers between clients and the server, as its noise sampling method is vulnerable to the *difference attack* [15]. The concern of third-party resources is two-fold. First, the third-party may also be compromised. Second, the employment of additional third-party resources can increase the cost in both system building and maintenance. Table 1 summarizes the features of different solutions to the aforementioned studies.

More recently, some different ideas are proposed to perform privacy-preservation recommendation. An encryption-based method [16] is developed to encrypt user-item rating data while maintaining the predictability of ratings in a centralized setting. CryptoRec [31] casts privacy preserving into feature learning problem by modeling user-item interactions in an item-only latent feature space so that the recommendation can use only pre-trained item information to avoid the leakage of user privacy. Li et al. [18] recommend users to the propagated information without considering user profile and posting content. However, in their settings, the rating profile (i.e., diffusion participation records) can be still potentially inferred. Guo et al. [14] impose differential privacy to social graphs by adding noising links while maintaining the effectiveness of link prediction. Chen et al. [9] develop a privacy-preserving ridge regression (PPRR) over high-dimensional data in distributed context. However, it is not for recommender systems. Qi et al. [24] deploy a time-aware distributed quality service recommendation by developing a privacy-preserving Locality-Sensitive Hashing (LSH) technique. Afsharinejad and Hurley [2] analyze how to create differentially private data sketches of user profiles to protect k-nearest neighbor collaborative from being inferred. Although such various settings and recent approaches are proposed, they can only achieve some part of our goal: simulta-

Table 1
Comparison between Different Solutions under Untrusted Server Scenario.

	without Additional Information	without 3rd Party	Existence Protected	Model Protected	Value Protected
[8]	√	√		√	√
[23]	√		√	√	√
[3]	√	√			√
[33]		√		√	√
[15]	√			√	√
[29]		√	√		√
[1]	√	√	√		√
[6]	√	√	√		√
[19]	√	√			√
SDCF	√	√	√	√	√

neously preserving value, model, and existence privacy, and holding the differential privacy in a distributed recommendation environment.

To the best of our knowledge, SDCF is the first to preserve *value, model*, and *existence* privacy issues and possesses the guarantee of differential privacy. More importantly, all feedback data are kept only at client side, i.e., it will not have any chance to be released or transmitted to the untrusted server or any other malicious client.

We choose Matrix Factorization (MF) [17] to realize SDCF, and develop *Secure Distributed Matrix Factorization* (SDMF), for the following reasons. First, MF is natural to divide the model into personal (user) U and public (item) V parts (matrices). Second, if we use gradient descent-based optimization method to train MF models, the updating of personal elements for different users can be done independently and locally at client side. This implies that no personal information will be transmitted to other clients or to the server. Third, the updating of public elements can be done by aggregating the gradients contributed from clients. It implies that only gradients of the model are transmitted from clients to the server. The attackers cannot recover original ratings nor personal elements even the gradients are intercepted during data transmission. In this paper, we adopt MF with *Stochastic Gradient Langevin Dynamics* (SGLD) [20], instead of the popular Stochastic Gradient Descent method. As will be discussed later, SGLD enjoys the advantage of preventing user latent vectors (i.e., personal elements) from being deciphered. The experiments show that our model can yield reasonably good recommendation results while guaranteeing decent level of privacy. In Section 5, we will further discuss how to use SDCF in *Factorization Machine*-based model [25].

2. Preliminaries

2.1. Differential privacy (DP)

Differential privacy provides a mathematical definition to quantify the privacy preserved by an algorithm [10,11]. Here we explain the meaning of DP in recommender systems. If users aim to preserve the *value privacy*, a privacy-preserving algorithm is supposed to lower down the server's confidence in identifying the actual value of each rating.

Definition 1 (differential privacy)**.** A randomized algorithm A with domain $\mathbb{R}^n$ is ϵ-differentially private, if and only if any two datasets $X, X' \in \mathbb{R}^n$ contain at most one different record (i.e., the hamming distance $d(X, X') = 1$) for possible anonymized output $O \subseteq Range(A)$:

$$\frac{Pr[A(X) \in O]}{Pr[A(X') \in O]} \leq e^{\epsilon},$$

where the probability Pr is taken over the randomness of algorithm A, and ϵ is positive. Lower values of ϵ indicates higher degree of privacy guaranteed.

With Definition 1, we can measure the difference between the outcomes with and without the presence of an element determined by ϵ. However, there is a trade-off between user privacy and the utility of an algorithm. Our model is designed to be a good privacy-preserving algorithm that can maintain the performance while increasing the degree of privacy.

2.2. Matrix factorization (MF)

Let $\mathcal{U}$ and $\mathcal{V}$ be a user set and an item set, and each r_{ij} indicates an observed score that a user i rated an item j in the rating matrix $R \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{V}|}$. MF aims to make $UV^T \approx R$, where $U \in \mathbb{R}^{|\mathcal{U}| \times K}$ and $V \in \mathbb{R}^{|\mathcal{V}| \times K}$ for a given dimension number K. The objective is to find U and V such that the loss function $L(U, V)$ can be minimized:

$$\min_{U,V} L(U, V) = \min_{U,V} \sum_{i,j \in R} (r_{ij} - u_i v_j^T)^2 + \lambda(||U||_2^2 + ||V||_2^2), \quad (1)$$

where λ is a given regularization factor to prevent overfitting. Stochastic Gradient Descent (SGD) is a popular technique to minimize the objective function of MF. It aims to iteratively learn u_i and v_j according to the following updating rules:

$$\begin{aligned}
\hat{\nabla}_{u_i} L(u_i, v_j) &= e_{ij} v_j + \lambda u_i, \\
\hat{\nabla}_{v_j} L(u_i, v_j) &= e_{ij} u_i + \lambda v_j, \\
u_i &\leftarrow u_i - \eta \hat{\nabla}_{u_i} L(u_i, v_j), \\
v_j &\leftarrow v_j - \eta \hat{\nabla}_{v_j} L(u_i, v_j),
\end{aligned} \tag{2}$$

where $e_{ij} = u_i {v_j}^T - r_{ij}$ is the error of the prediction and η is the learning rate. These updating rules allow the gradients of u_i and v_j to be computed independently, and thus enable the deployment in a distributed environment. However, applying SGD in a distributed manner may expose u_i (i.e., *model* leakage) during gradients transmission. To remedy this problem, we introduce SGLD below.

2.3. Stochastic gradient Langevin dynamics (SGLD)

SGLD [32] is an optimization method, which combines SGD and Langevin Dynamics that samples from the posterior distribution to prevent overfitting. Specifically, from the Bayesian view of MF, we can convert the objective into the maximization of posterior distribution of U and V given the observed ratings and parameters:

$$p(U, V | R, \lambda_r, \Lambda_u, \Lambda_v) \propto p(R | U, V, \lambda_r) p(U | \Lambda_u) p(V | \Lambda_v), \tag{3}$$

where λ_r is the global regularization term, Λ_u and Λ_v are diagonal matrices generated by Gamma distribution for the regularization of u_i and v_j. Since a hypothesis of normal distribution $\mathcal{N}$ can be adopted in Eq. (3), the log-likelihood can be derived from:

$$\begin{aligned}
F(U, V) &= \ln p(U, V | R, \lambda_r, \Lambda_u, \Lambda_v) \\
&= \ln p(R | U, V, \lambda_r) + \ln p(U | \Lambda_u) + \ln p(V | \Lambda_v) + C \\
&= \ln \mathcal{N}(R | UV^T, \lambda_r^{-1}) + \ln \mathcal{N}(U | 0, \Lambda_U^{-1}) + \ln \mathcal{N}(V | 0, \Lambda_V^{-1}) + C,
\end{aligned} \tag{4}$$

where C is a constant. To maximize the log-likelihood and also to follow the normal distribution, by adding noises into gradients to avoid overfitting, the updating rules of SGLD in the tth iteration become:

$$\begin{aligned}
&\hat{\nabla}_{u_i} F(u_i, v_j) = e_{ij} v_j + u_i \Lambda_u, \\
&\hat{\nabla}_{v_j} F(u_i, v_j) = e_{ij} u_i + v_j \Lambda_v, \\
&\eta_t = \frac{\eta_0}{t^\gamma}, \\
&u_i \leftarrow u_i - \eta_t \hat{\nabla}_{u_i} F(u_i, v_j) + \mathcal{N}(0, \eta_t \mathtt{I}), \\
&v_j \leftarrow v_j - \eta_t \hat{\nabla}_{v_j} F(u_i, v_j) + \mathcal{N}(0, \eta_t \mathtt{I}),
\end{aligned} \tag{5}$$

where η_0 is the initial learning rate, γ is the decay factor, and I is the identical matrix of size K. As Liu et al. [20] had proved that introducing noises to SGLD can guarantee differential privacy. As will be shown later on, our model using SGLD can avoid the exposure of *model*.

3. Methodology

3.1. Framework overview

The overview of the proposed SDMF framework is exhibited in Fig. 2. It is a distributed framework consisting of several clients and the server. The key idea is to divide the model into the private (user) part and a public (item) part, and transmit only gradient values to avoid leakage. Since MF learns two matrices, item and user latent factors, by iteratively performing updates, we consider item latent factors as the public elements in SDCF. Therefore the public part is the only information stored in the server, as illustrated in Fig. 1. In addition, user latent factors are treated as personal elements and thus are stored with their ratings at client side only. With SGD to optimize MF's objective (Eq. (1)), we define the gradients (denoted by $grad(v_j)$ for the latent factors of item i as: $grad(v_j) = \eta \hat{\nabla}_{v_j} L(u_i, v_j)$. The updating rule in Eq. (2) can be rewritten as:

$$v_j \leftarrow v_j - grad(v_j).$$

The gradients are the only things transmitted from clients to the server. Specifically, in each iteration, each client needs to download the newest V from the server, then calculates the gradients for u_i and v_j. The gradients of u_i are directly used to update u_i stored at the client side while $grad(v_j)$ is sent to the server to update v_j. However, we further look into the formation of $grad(v_j)$ and find that

$$grad(v_j) = \eta(e_{ij} u_j + \lambda v_j).$$

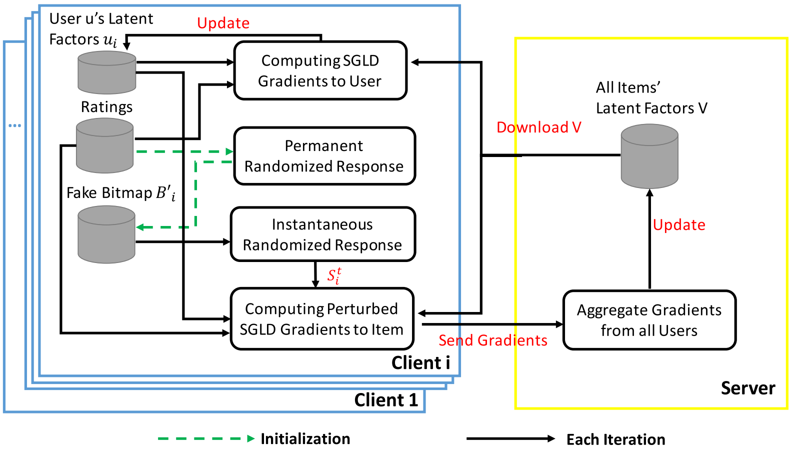

User u's Latent Factors u_i
Update
Computing SGLD Gradients to User
Ratings
Permanent Randomized Response
Fake Bitmap B'_i
Instantaneous Randomized Response
S_i^t
Computing Perturbed SGLD Gradients to Item
Client i
Client 1
...
All Items' Latent Factors V
Download V
Update
Aggregate Gradients from all Users
Send Gradients
Server
Initialization
Each Iteration

Fig. 2. The overview of SDMF framework. First, each client reads the ratings from their own database and performs PRR to calculate the fake bitmap B'_i to initialize the training. For every iteration, each client will do three steps: (1) Download the items' factors V. (2) Read the ratings and user's factors u_i, and calculate the gradients for updating u_i. (3) Read the ratings, the B'_i and u_i, and calculate the gradients for updating V.

Since v_j, η, and λ are public, the server can obtain $e_{ij}u_j$. When u_j converges, the server will be able to approximate each u_j given $e_{ij}u_j$ for different items within a few iterations. It is because u_j is the greatest common divisor (GCD) of $e_{ij}u_j$ values.

To avoid the exposure of u_i, we leverage SGLD instead of SGD to protect the gradients by adding Gaussian noises. The gradients of items can be re-written as:

$$\begin{aligned} grad(v_j) &= \eta_t \hat{\nabla}_{v_j} F(u_i, v_j) - \mathcal{N}(0, \eta_t \mathrm{I}) \\ &= \eta_t (e_{ij}u_i + v_j \Lambda_v) - \mathcal{N}(0, \eta_t \mathrm{I}). \end{aligned} \tag{6}$$

Aside from differential privacy proved by Liu et al. [20], the noises introduced make it hard to recover u_i from $grad(v_j)$ by GCD. In addition, since the noises in each iteration are sampled with different values, we are immune to the *difference attack* mentioned in the work of Hua et al. [15]. Note that the difference attack is an attack that removes the noises by calculating the difference between gradients of two consecutive iterations. Therefore, even if the gradients are exposed, the privacy of *model* and *value* can still be preserved.

3.2. Randomized response (RR) algorithm

Nevertheless, the *existence* is still possibly exposed due to the action that user i sends $grad(v_j)$ of item j to the server in SGLD. Thus, instead of sending gradients of all rated items, we send the gradients of some unrated items to the server so that exact rated items are unknown. We implement such idea with the technique of randomized response. A straightforward solution consists of two steps. First, we create a binary matrix to represent the existence of ratings, as illustrate in the left part of Fig. 3, in which each user i maintains a binary vector B_i whose element B_{ij} indicates whether user i rates item j. Second, in tth iteration, the client does not send the gradients of all rated items, but the gradients of a set of perturbed items determined by a bit array S_i^t. This naive algorithm is shown in the upper right of Fig. 3. Specifically, the bit array S_i^t of user i is used to protect the user's existence by randomizing the rated and unrated items. If $S_{ij}^t = 1$, item j's gradient will be sent to the server, otherwise not. We generate the bit array S_i^t of item i based on its binary vector B_i, given by:

$$P(S_{ij}^t = 1) = \begin{cases} p, & \text{if } B_{ij} = 0 \\ q, & \text{if } B_{ij} = 1 \end{cases}, \forall j \in \mathcal{V}, \tag{7}$$

where p and q are two given probabilities to determine the number of perturbed unrated and rated items in S_i^t. Higher p leads to larger disturbance in the randomization while higher q preserves the actual rating behavior. Note that in each iteration t, the bit array S_i^t of item i is re-generated.

Unfortunately, such simple randomization is vulnerable to the *average attack*. Specifically, untrusted server may collect the gradients for a sufficient number of iterations T and obtain the rating actions by averaging the derived bit array of user i with the following formula:

$$\frac{1}{T}\sum_{t=1}^{T}(S_{ij}^t) \approx \begin{cases} p, & \text{if } B_{ij} = 0 \\ q, & \text{if } B_{ij} = 1 \end{cases} \tag{8}$$

To address this, we aim at realizing the idea of randomized response (RR), which is originally presented by RAPPOR [12], in order to construct a secured and distributed recommender system. Note that we are the first that implements the concept

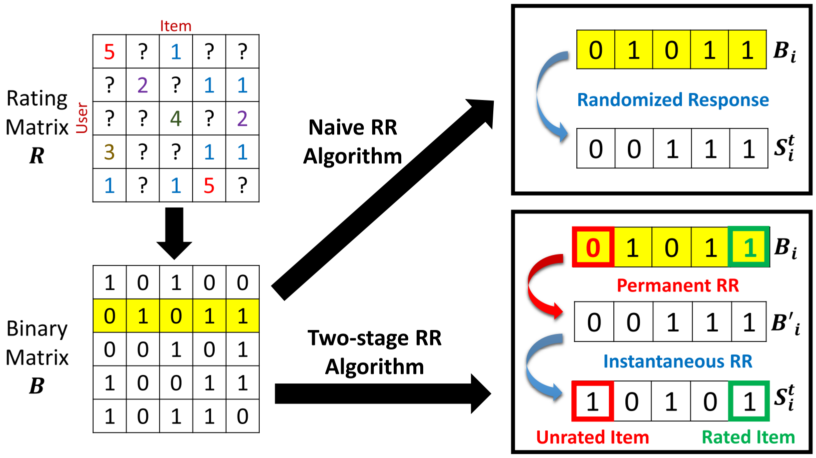


Fig. 3. Representing the existence of ratings by a binary matrix (left). Using RR to sample a noised bit vector S_i^t from binary vector B_i of user i (upper right). Two-stage RR algorithm (lower right).

of randomized response in distributed recommendation. The proposed two-stage RR algorithm is to strengthen the perturbation of rated and unrated items via the binary vector B_i and the bit array S_{ij}. The *Permanent Randomized Response* (PRR) stage prevents rating exposure caused by the average attack over multiple iterations while the *Instantaneous Randomized Response* (IRR) stage preserves privacy in every single iteration. Note that PRR will be executed only once during initialization and IRR will be executed in the beginning of every iteration.

3.2.1. Permanent Randomized Response (PRR)

We generate a perturbed binary vector B'_i to permanently replace the original B_i so that we can apply the RR mechanism based on not only the actual rating actions but also some fake “1”s and “0”s. With such strategy, the average attacker will not be able to derive B_i since all information sent to server are generated from both fake and actual rating actions. To implement such idea, when initializing the optimization process, we sample the perturbed binary vector B'_i for each user i by

$$B'_{ij} = \begin{cases} 1, \text{with probability } \frac{1}{2}f \\ 0, \text{with probability } \frac{1}{2}f \\ B_{ij}, \text{with probability } 1-f \end{cases}, \forall j \in \mathcal{V}, \tag{9}$$

where the parameter f is a probability specified by the client to control the strength of perturbation, and thus determines the percentage of actual rating actions in B'_i. With a higher f, it will be more private with more perturbation but provide less information for the prediction model. Then B'_i is fixed and becomes the input of IRR in each iteration. Note that instead of letting the client provide the probability parameter f, the value of f is automatically determined by the differential privacy parameter ϵ_I, which will be introduced and discussed in Section 3.4.

3.2.2. Instantaneous Randomized Response (IRR)

With the perturbed binary vector B'_i derived from PRR, IRR is to implement the naive approach by generating the bit array S_i^t. Specifically, for each iteration t, we sample a bit array S_i^t for every user i based on two probabilities p and q using Eq. (7), in which the original binary vector B_i is replaced by the perturbed version B'_i. Then the client sends gradients of user i to every item j if $S_{ij}^t = 1$. Note that, since the bit array S_i^t is generated by the perturbed binary vector B'_i in Eq. (9), the average attack at server side can at most recover the perturbed version B'_i. We discuss the properties of DP for both PRR and IRR in Section 3.4.

3.2.3. Number of gradients sent to the server

Given that we can determine the degree of privacy by adjusting the probability parameters f, p and q, in this subsection, we discuss how these parameters affect the number of gradients sent to the server that influences the transmission cost of a distributed recommender system. Recall in Eq. (7) that the probabilities p and q determine the perturbed rated and unrated items via the bit array. We can estimate the number of gradients sent by first calculating the probabilities of sending the gradients of unrated and rated items, denoted by $p^\star$ and $q^\star$, respectively.

Lemma 1. *The probability of sending gradients of* <u>*unrated*</u> *item j to the server is:*

$$p^\star = P(S_{ij}^t = 1 | B_{ij} = 0)$$

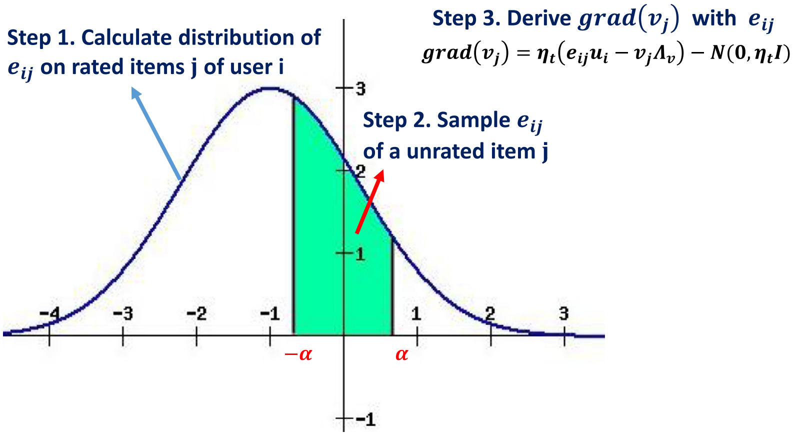

Step 1. Calculate distribution of e_{ij} on rated items j of user i
Step 3. Derive $grad(v_j)$ with e_{ij}
$grad(v_j) = \eta_t(e_{ij}u_i - v_j\Lambda_v) - N(0, \eta_t I)$
Step 2. Sample e_{ij} of a unrated item j
3
2
1
-1
-4
-3
-2
-1
1
2
3
$-\alpha$
α

Fig. 4. Steps to sample e_{ij} for unrated items.

$$\begin{aligned}
&= P(B'_{ij} = 1 | B_{ij} = 0) P(S^t_{ij} = 1 | B'_{ij} = 1) \\
&\quad + P(B'_{ij} = 0 | B_{ij} = 0) P(S^t_{ij} = 1 | B'_{ij} = 0) \\
&= \frac{1}{2} f \times q + \left(\frac{1}{2} f + (1 - f)\right) \times p.
\end{aligned} \tag{10}$$

The probability of sending gradients of <u>*rated*</u> *item j to the server is:*

$$\begin{aligned}
q^\star &= P(S^t_{ij} = 1 | B_{ij} = 1) \\
&= P(B'_{ij} = 1 | B_{ij} = 1) P(S^t_{ij} = 1 | B'_{ij} = 1) \\
&\quad + P(B'_{ij} = 0 | B_{ij} = 1) P(S^t_{ij} = 1 | B'_{ij} = 0) \\
&= \left(\frac{1}{2} f + (1 - f)\right) \times q + \frac{1}{2} f \times p.
\end{aligned} \tag{11}$$

Let the number of items rated by user i be h_i, the expected number of gradients sent to the server is

$$z = h_i \times q^\star + (|\mathcal{V}| - h_i) \times p^\star. \tag{12}$$

It is obvious that the fewer gradients to send, the smaller the transmission cost will be. The server will get less information to update the model. Therefore, our model can flexibly allow users to adjust these parameters according to their needs. In our experiments, for each user i, we set z to be $\frac{|R|}{|\mathcal{U}|}$, which is the average number that each user has rated, so that the total amount of gradients that the server receives is the same as the original MF, and we can make a fair comparison. Note that with z and Theorem 2, which allows users to specify a privacy budget ϵ_I, we can obtain $p^\star$ and $q^\star$.

3.3. The gradients of unrated items

Now we have only one remaining issue: what to transmit to the server for an item not rated by the user? For a rated item j, the user i can simply send her gradient $grad(v_j) = \eta_t(e_{ij}u_i + v_j\Lambda_v) - \mathcal{N}(0, \eta_t \mathrm{I})$, where e_{ij} is the difference between the predicted and the observed values, $u_i v_j^T - r_{ij}$, as mentioned in Eq. (2). However, if item j is unrated, we cannot get e_{ij} because r_{ij} does not exist. Therefore, we sample "fake" e_{ij} from the distribution of observed e_{ij} of user i. As pointed out in the Bayesian view of SGLD (Eq. (3)), the probabilities of U and V given the observed ratings are assumed to be in normal distributions. Consequently, we can shift $\mathcal{N}(R|UV^T, \lambda_r^{-1})$ in Eq. (4) to be zero mean to obtain the distribution of e_{ij}, which is also a normal distribution. Therefore, we develop the sampling method for unrated item to have three steps, as illustrated in Fig. 4. First, given a certain user i, we can count the sample mean μ and standard deviation σ from the rated items so the distribution of e_i can be derived. Second, we sample e_{ij} for each unrated item j from $\mathcal{N}(\mu, \sigma)$ within a given range $[-\alpha, \alpha]$. Third, the gradients $grad(v_j)$ with sampled e_{ij} can be derived.

Since e_{ij} is sampled from the distribution of the rated items, the attacker cannot distinguish "fake" gradients from actual ones. Besides, the bound in α can constrain the range of e_{ij} to avoid imposing serious noise on the recommendation model. Furthermore, we will discuss how α can be set to control the privacy level in Section 3.4.

3.4. Differential privacy in our framework

An important feature in our framework is that the level of privacy is controllable based on the parameter f in PRR, the probabilities p, and q in IRR, as well as the range factor α in computing gradients of unrated items. In this section, we aim to provide theoretical analysis on how the Differential Privacy (DP) is related to these parameters so that one can determine the correspondent parameters for arbitrary privacy levels. Although the total privacy budget scales up with the number of iterations, we can focus on the privacy level of each iteration. The reason is that the number of iterations that SGLD requires to converge to a certain performance is varied in different datasets. Therefore, we keep it a decision that should be made by the service providers according to their particular goals.

DP of two-stage randomized response algorithm. To control the privacy level of the proposed two-stage RR algorithm, we will set a few privacy budgets to control the difficulty for the server to identify the rated items according to the number of times the gradients of items are sent from the user. In the following, we prove the DP in PRR and IRR with the privacy budgets ϵ_P and ϵ_I, respectively, and derive the relation between these privacy budgets and the probabilities *f*, *p*, and *q*.

Theorem 1. *With a privacy budget ϵ_P, PRR is ϵ_P-differentially private if*

$$\epsilon_P = 2h_i \ln\left(\frac{1-\frac{1}{2}f}{\frac{1}{2}f}\right).$$

Proof. Given the number of rated items h_i and the item set $\mathcal{V}$, for any binary vector B_i with $|\mathcal{V}|$ elements and containing h_i "1"s, without loss of generality, we can set B_i as $b^\star = \{b_1 = 1, ..., b_{h_i} = 1, b_{h_i+1} = 0, ..., b_{|\mathcal{V}|} = 0\}$.

Assume O_B derived from Eq. (9) in PRR, for all $b' \in O_B$ (for simplicity, we let b' be any possible B_i'), given $B_i = b^\star$, the probability of B_i being changed into some $B_i' = b'$ is

$$\begin{aligned} P(B_i' = b' | B_i = b^\star) = & \left(1 - \frac{1}{2}f\right)^{b_1'} \left(\frac{1}{2}f\right)^{1-b_1'} \\ & \times \cdots \times \left(1 - \frac{1}{2}f\right)^{b_{h_i}'} \left(\frac{1}{2}f\right)^{1-b_{h_i}'} \\ & \times \left(\frac{1}{2}f\right)^{b_{h_i+1}'} \left(1 - \frac{1}{2}f\right)^{1-b_{h_i+1}'} \\ & \times \cdots \times \left(\frac{1}{2}f\right)^{b_{|\mathcal{V}|}'} \left(1 - \frac{1}{2}f\right)^{1-b_{|\mathcal{V}|}'} \end{aligned}$$

Then for any pair of B_1 and B_2 in any possible B_i, we can derive the ratio in differential privacy (Definition 1) as below:

$$\begin{aligned} \frac{P(B_i' \in O_B | B_i = B_1)}{P(B_i' \in O_B | B_i = B_2)} &= \frac{\sum\limits_{b' \in O_B} P(B_i' = b' | B_i = B_1)}{\sum\limits_{b' \in O_B} P(B_i' = b' | B_i = B_2)} \\ &\leq \max_{b' \in S} \frac{P(B_i' = b' | B_i = B_1)}{P(B_i' = b' | B_i = B_2)} \\ &= \left(1 - \frac{1}{2}f\right)^{2(b_1' + b_2' + \cdots + b_{h_i}' - b_{h_i+1}' - b_{h_i+2}' - \cdots - b_{2h}')} \\ &\times \left(\frac{1}{2}f\right)^{2(b_{h_i+1}' + b_{h_i+2}' + \cdots + b_{2h_i}' - b_1' - b_2' - \cdots - b_{h_i}')} \\ &\leq \left(\frac{1-\frac{1}{2}f}{\frac{1}{2}f}\right)^{2h_i} = e^{\epsilon_P}, \text{when } \epsilon_P = 2h_i \ln\left(\frac{1-\frac{1}{2}f}{\frac{1}{2}f}\right) \end{aligned}$$

□

Theorem 2. *With a privacy budget ϵ_I, IRR is ϵ_I-differentially private, if*

$$\epsilon_I = h_i \ln\left(\frac{q^\star(1-p^\star)}{p^\star(1-q^\star)}\right), \tag{13}$$

where $p^\star$ and $q^\star$ follow Lemma 1.

Proof. Assume O_S derived from Eq. (7) in IRR, for $s \in O_S$ (for simplicity, we let s be any possible S_i^t), given $B_i = b^\star$, the probability of B_i being changed into some $S_i^t = s$ is

$$
\begin{aligned}
P(S_i^t = s | B_i = b^\star) = {} & (q^\star)^{s_1}(1-q^\star)^{1-s_1} \times \cdots \times (q^\star)^{s'_{h_i}}(1-q^\star)^{1-s'_{h_i}} \\
& \times (p^\star)^{s'_{h_i+1}}(1-p^\star)^{1-s'_{h_i+1}} \times \cdots \times (p^\star)^{s'_{|\mathcal{V}|}}(1-p^\star)^{1-s'_{|\mathcal{V}|}}.
\end{aligned}
$$

Then for any pair of B_1 and B_2 in any possible B_i, we can derive the ratio in differential privacy (Definition 1.) as below:

$$
\begin{aligned}
\frac{P(S_i^t \in O_S | B_i = B_1)}{P(S_i^t \in O_S | B_i = B_2)} &= \frac{\sum\limits_{s \in O_S} P(S_i^t = s | B_i = B_1)}{\sum\limits_{s \in O_S} P(S_i^t = s | B_i = B_2)} \le \max_{s \in O_S} \frac{P(S_i^t = s | B_i = B_1)}{P(S_i^t = s | B_i = B_2)} \\
&\le \left(\frac{q^\star(1-p^\star)}{p^\star(1-q^\star)} \right)^{h_i} = e^{\epsilon_I}, \text{ when } \epsilon_I = h_i \ln\left(\frac{q^\star(1-p^\star)}{p^\star(1-q^\star)} \right).
\end{aligned}
$$

□

Practically, ϵ_I is smaller than ϵ_P because ϵ_P is the privacy guarantee of the worst case suffering from the average attack over several iterations. In our implementation, we set $\epsilon_P = 2\epsilon_I$ and find f by ϵ_P. Consequently, given ϵ_I specified by the client, we can obtain p and q by solving Eqs. (10)–(13).

DP of computation of gradients to unrated items. Here we explain how to use the range factor α to control the difficulty for the attacker. The goal is to distinguish fake gradients of unrated items from gradients of rated items at server side according to the numerical values of gradients. Note that such privacy level is different from the DP of Randomized Response algorithm, which is the difficulty of detecting rated items according to discrete results of being sampled or not.

Theorem 3. *With a privacy budget ϵ_g, gradients for unrated items are ϵ_g-differentially private, if*

$$
\epsilon_g = \ln\left(\left(\frac{1}{2} erf\left(\frac{x-\mu}{\sigma\sqrt{2}} \right) \Big|_{-\alpha}^{\alpha} \right)^{-1} \right), \tag{14}
$$

where α is the range factor, and erf() is the Gauss Error Function:

$$
erf(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} e^{-t^2} dt.
$$

Proof. Based on Section 3.3, we have the normal distribution of the observed e_{ij}:

$$
f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}},
$$

where μ and σ are mean and standard deviation of the observed e_{ij}. Assume e' is any possible e_{ij}, then for any possible item pair of j_1 and j_2, we derive the ratio in differential privacy (Definition 1.) as below:

$$
\begin{aligned}
\frac{P(e_{ij} = e' | j = j_1)}{P(e_{ij} = e' | j = j_2)} &\le \frac{\frac{f(e')}{\int_{-\alpha}^{\alpha} f(x)dx}}{f(e')} = \frac{1}{\int_{-\alpha}^{\alpha} f(x)dx} = \frac{1}{\frac{1}{2} erf\left(\frac{x-\mu}{\sigma\sqrt{2}} \right) \Big|_{-\alpha}^{\alpha}} \\
&\le e^{\epsilon_g}, \text{ when } \epsilon_g = \ln\left(\left(\frac{1}{2} erf\left(\frac{x-\mu}{\sigma\sqrt{2}} \right) \Big|_{-\alpha}^{\alpha} \right)^{-1} \right).
\end{aligned}
$$

□

Since it is difficult to directly solve Eq. (14), in our implementation, we set

$$
\alpha_{Max} = Max(|\mu + 2\sigma|, |\mu - 2\sigma|)
$$

so that $[-\alpha, \alpha]$ can cover more than 95% of the ratings in the distribution. Then we do binary search in $(0, \alpha_{Max})$ to find an α that

$$
e^{\epsilon_g - \delta} \le \left(\frac{1}{2} erf\left(\frac{x-\mu}{\sigma\sqrt{2}} \right) \Big|_{-\alpha}^{\alpha} \right)^{-1} \le e^{\epsilon_g},
$$

where δ is a very small number (e.g. 10^{-6} in our experiments) so that

$$
\epsilon_g \approx \ln\left(\left(\frac{1}{2} erf\left(\frac{x-\mu}{\sigma\sqrt{2}} \right) \Big|_{-\alpha}^{\alpha} \right)^{-1} \right).
$$

3.5. Implementation details of SDMF

We further explain the procedure on the client and the server during the training process of SDMF.

Algorithm 1 Client (i, t).

```
Require: Data Input: {$\mathcal{V}$, $h_i, R_i$, $B_i$}, Model Input: {$\eta_0, \gamma$, $\Lambda_u$, $\Lambda_v$}, Privacy Input: {$\epsilon_g$, $\epsilon_I$, $\epsilon_P$}
if $t = 0$ then
    Initialize $u_i$
    Find $f$ with $\epsilon_P$
    $B_i' \leftarrow PRR(B_i, f)$
    Find $p$ and $q$ with $\epsilon_I$ and $h_c$
if $t > 0$ then
    $\eta_t \leftarrow \frac{\eta_0}{t^\gamma}$
    Download $V$ from Server
    $S_c(t) \leftarrow IRR(B_i', p, q)$
    $\bar{u_i} \leftarrow 0$
    for $j \in \mathcal{V}$ do
        if $B_{ij} = 1$ then
            $e_{ij} \leftarrow u_i v_j^T - r_{ij}$
            $\bar{u_i} \leftarrow \bar{u_i} + (\eta_t(e_{ij}v_j + u_i\Lambda_u) - \mathcal{N}(0, \eta_t\mathtt{I}))$
    $\bar{u_i} \leftarrow \frac{\bar{u_i}}{|R_i|}$
    Calculate $\mu$ and $\sigma$ of $\{e_i j | \forall B_{ij} = 1\}$
    Find $\alpha$ with $\mu$, $\sigma$ and $\epsilon_g$
    for $j \in \mathcal{V}$ do
        if $S_i(t)_j = 1$ and $B_{ij} = 1$ then
            $grad(v_j) \leftarrow \eta_t(e_{ij}u_i + v_j\Lambda_v) - \mathcal{N}(0, \eta_t\mathtt{I})$
            Send $(j, grad(v_j))$ to Server
        if $S_i(t)_j = 1$ and $B_{ij} = 0$ then
            while True do
                Sample $e_{ij} \sim N(\mu, \sigma)$
                if $\alpha > x > -\alpha$ then
                    break
            $grad(v_j) \leftarrow \eta_t(e_{ij}u_i + v_j\Lambda_v) - \mathcal{N}(0, \eta_t\mathtt{I})$
            Send $(j, grad(v_j))$ to Server
    Send "finish" to Server
    $u_i \leftarrow u_i - \bar{u_i}$
```

3.5.1. Client side

A list of step-by-step actions in each iteration t is shown in Algorithm 1. A client first initializes its latent factors. Then it decides the two probabilities used in IRR, p and q, and conducts PRR as Eq. (8) when $t = 0$. In every iteration t, it downloads the item latent factors and calculates the prediction error e_{ij} of the rated item. After generating the mean and standard deviation of e_{ij}, the client can find a suitable α. Later, the client will conduct IRR as Equation to generate S_i^t of this iteration t. For every item j that $S_i^t = 1$, if it is a rated item, the client sends the server j and $grad(v_j)$ as in general SGLD; if it is an unrated item, sample e_{ij} in $N(\mu, \sigma)$ (and re-sample if it is not within a bound of $[-\alpha, \alpha]$), sending $grad(v_j)$ calculated with this sampled e_{ij} and j to the server. After the client finishes the sending of *gradients* to the server, it sends a “finish” signal to the server and updates its user latent factors with the average updating calculated before.

3.5.2. Server side

Algorithm 2 shows how the server performs training, obtains *gradients*, and updates item latent factors. Comparing with Algorithm 1, it can be seen that the server needs to do very little computation. The server is only in charge of controlling the start of a new iteration and updating item latent factors with whatever $grad(v_j)$ it receives. In this process, nothing but the item latent factors need to be stored, thereby the privacy is preserved. Besides, we can make all clients compute *gradients* of all their ratings at the same time in each iteration. With such parallelization, the time that the server has to wait for all gradients will be largely reduced, comparing to a centralized training of MF.

Note that the stopping criteria can be customized to fit different services, and here we provide two commonly-used ones: the convergence of model and the maximum number of iterations. First, if the stopping criterion is set to be the convergence of the model, we can make each client send only one additional bit to the server, in which such a bit represents whether their own latent factors have converged. When every client sets the bit to “True” and the item latent factors also converge, the server can stop the training process. The second criteria is simply setting a maximum number of iterations to stop the training process no matter the latent factors converge or not. As the privacy budget scales with the number of iterations, there are trade-offs between performance and privacy. It can be seen that the convergence of model can reach better performance while setting a large enough number of iterations guarantees the privacy.

Algorithm 2 Server.

```
Require: $\mathcal{U}$, $\mathcal{V}$
Initialize $V$
$t \leftarrow 0$
for $i \in \mathcal{U}$ do
    Call Client i to run Client($i$, $t$)
$t \leftarrow 1$
while not reach stop criteria do
    Initialize $\bar{V}$ with all zero
    Initialize $Count = 0$
    for $i \in \mathcal{U}$ do
        Call Client i to run Client(i, t)
    while True do
        Wait until receiving a $(j, grad(v_j))$
        $\bar{V}_j \leftarrow \bar{V}_j + grad(v_j)$
        $Count \leftarrow Count + 1$
        if receive ”finish” from all of the Clients then
            break
    $V \leftarrow V - \frac{\bar{V}}{Count}$
    $t \leftarrow t + 1$
```

3.6. Discussion

In the following, we will discuss two practical issues of the proposed model that should be considered in real-world application.

First, as trade-offs for privacy, the clients have to do more computation and transmission in our proposed framework than in the conventional scenario. For each client, the time complexity for computing the gradients in a single iteration is $O(h_i K)$, where h_i is the rated items by user i and K is the latent dimension number. For the server, the time complexity for updating item latent factors is $O(\sum_{\forall i \in \mathcal{U}} h_i K)$. However, the actual computation time may depend on the devices that the clients and the server use. As for the transmission overhead, in every iteration, since the space complexity for each client to download item latent factors is $\mathcal{V}K$, the space complexity for uploading the gradients is $O(h_i K)$.

The second issue should be considered is that some clients may be offline in some iterations. It is difficult to force all the clients to always online or online at the same time. So there will be some iterations that only some clients do updating actions to the latent factors. However, service providers can give some daily rewards or incentives to encourage the clients to be regularly online every day to increase the number of online users. Besides, the more frequently online users will get better prediction because they contribute more updates according to their rating data. Therefore, the service providers can use and promote such fact to encourage users to be online. Note that in fact, in modern commercial services or social media platforms, various practical incentive mechanisms had been adopted and deployed to attract users and boost the consumption volume.

4. Evaluation

We conduct experiments to evaluate the utility of SDMF. Specifically, the experiments aim at examining the trade-off between privacy and performance of the proposed SDMF. The evaluation is designed to answer three questions: (1) Can SDMF maintain high accuracy while preserving certain level of privacy? (2) How does the privacy budgets ϵ_I and ϵ_g affect the accuracy in the proposed two-stage RR method? (3) How does SDMF perform on numerical and one-class rating prediction tasks? Note that the experiments here focus on the effect of using DP to preserve *existence* privacy, so we do not compare with other DP methods such as [15,29] since their DP algorithms aim to preserve *value* privacy, which SDMF can completely preserve with the architecture.

4.1. Datasets & evaluation settings

We use three popular public rating datasets for the experiments: two MovieLens datasets[1] (ML-100K and ML-1M) and Netflix data [4]. Note that we subsample Netflix data to a subset with 10,000 users and 5000 items, all with more than 10 ratings to avoid cold-start users, who will contribute little due to the RR algorithm. The statistics of MovieLens-100K, MovieLens-1M and the subsampled Netflix dataset are shown in Table 2.

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

Table 2
Statistics for the datasets.

Dataset	#Users	#Items	#Ratings
MovieLens-100K	943	1682	100,000
MovieLens-1M	6040	3900	1,000,209
Netflix (subsampled)	10,000	5000	573,595

Task 1: Numerical Rating Prediction. We randomly split both rating datasets into 80% as the training set and 20% as the test set. This random splitting is repeated for 30 times to obtain the average *Root-Mean-Square Error* (RMSE). In order to examine how different privacy levels affect the performance, we vary the controllable parameters, i.e., the privacy budgets ϵ_g and ϵ_I in SDMF. As data quality (or noise introduced) is determined by ϵ-differential privacy, higher values of ϵ_g and ϵ_I lead to weaker privacy protection, but maintain the accuracy of the recommendation model. Hence, we choose to have a wide range of privacy budgets so that the trade-off between performance and privacy can be observed. The privacy budgets ϵ_g and ϵ_I are chosen from {4, 1, 0.25, 0.0625}, resulting in 16 combinations of privacy budgets in total. Note that we set $\epsilon_P = 2\epsilon_I$ so that we do not show the results of different values of ϵ_P.

We follow Hua et al. [15] to set $K = 50$, which is a K that can reach good performance on MovieLens dataset and Netflix dataset. The other settings of parameters are: $\gamma = 0.6$, and, $(\Lambda_u, \Lambda_v) \sim Gamma(1, 100)$. The learning rate η_0 is 5×10^{-6} for MovieLens-100K, 5×10^{-7} for MovieLens-1M, and 5×10^{-7} for Netflix, which are tuned to have better performance based on a validation set made by randomly splitting the training set into 80%–20%. Besides, SDMF will be compared to the baselines and competitive methods as listed in the following.

- **Non-private MF:** Performing recommendation using the original MF with SGLD. We compare this non-private version with SDMF to understand how the noises introduced for preserving privacy affect performance.
- **Input Perturbation SGLD (ISGLD):** We make a comparison to a naïve strategy of simply adding noises to ratings. Note that this solution can only preserve value privacy, not model privacy nor existence privacy. We add Laplacian noises to ratings and train MF using SGLD on these perturbed ratings to derive ISGLD. The ϵ-differential privacy of adding Laplacian noises to ratings has been proved in [5]. We compare SDMF with ISGLD with privacy budget ϵ set to 4 and 2, while the former is the largest number we set for privacy budgets in SDMF and the latter is what reported to have comparable performance as item average baseline by Berlioz et al. [5].
- **SDMF $\alpha = \infty$:** This is an SDMF given no constraint on the sampled fake gradient. Hence, it would lead to the highest privacy level for given fixed ϵ_g and ϵ_I in SDMF.

Note that the meanings of our ϵ, ϵ_I and ϵ_g are different from the privacy budgets in previous studies [1,15,28,29,33], because their purposes are to preserve the privacy of *value* and *model* while our privacy budgets are used for preserving the privacy of *existence*. It is meaningless and unfair to compare our framework with them. Therefore, we only present the results of ISGLD to reflect the level of the increasing of error after the decreasing of privacy budgets.

Task 2: One-Class Rating Action Prediction. To conduct experiments of SDMF on the task of one-class feedback, we choose to predict the rating actions via Bayesian Personalized Ranking Matrix Factorization (BPRMF) [26] in ML-100K, ML-1M, and the subsampled Netflix dataset. That says, BPRMF is considered MF technique in our SDMF. The split of training and testing is set by the leave-one-out strategy, which is the same as BPRMF[26]. Specifically, we first randomly select one rating action of each user to be added into the testing set while the training set consists of all rating actions except those in the testing set. This process is repeated for 10 times to generate the average Area under ROC Curve (AUC), which indicates the correctness of pair-wise ranking between a positive (rated) and a negative (unrated) sample. While the gradients of BPRMF is similar to the basic MF, we can accordingly apply BPRMF to SDMF by changing the error e_{ij} in MF to $\frac{-e^{-x_{ijj'}}}{1+e^{-x_{ijj'}}}$ for rated items and $\frac{e^{-x_{ijj'}}}{1+e^{-x_{ijj'}}}$ for unrated items, where $x_{ijj'} = u_i v_j^T - u_i v_{j'}^T$ denotes the distance between the predicted ranking scores of a rated item j and an unrated item j'. The updating rules of using BPRMF in SDMF, termed **SD-BPRMF**, is shown in Lemma 2. Since the optimization of BPRMF computes gradients for unrated items, SD-BPRMF does not need the "fake" errors, and the privacy budget ϵ_g is not applied here.

Lemma 2. *For a rated item j rated by user i, SD-BPRMF randomly samples an unrated item j′ and updates its latent factors with*

$$u_i \leftarrow u_i - \eta_0\left(\frac{e^{-x_{ijj'}}}{1+e^{-x_{ijj'}}}(-v_j + v_{j'}) + u_i\Lambda_u\right) + \mathcal{N}(0, \eta_t \mathtt{I}),$$

$$v_j \leftarrow v_j - \eta_0\left(\frac{-e^{-x_{ijj'}}}{1+e^{-x_{ijj'}}}(u_i) + v_j\Lambda_v\right) + \mathcal{N}(0, \eta_t \mathtt{I}),$$

$$v_{j'} \leftarrow v_{j'} - \eta_0\left(\frac{e^{-x_{ijj'}}}{1+e^{-x_{ijj'}}}(u_i) + v_{j'}\Lambda_v\right) + \mathcal{N}(0, \eta_t \mathtt{I}).$$

We compare the performance of SD-BPRMF with the privacy budget $\epsilon_I \in \{4, 1, 0.25, 0.0625\}$ and a non-private version of BPRMF using SGLD as the baseline. The settings of parameters are: $K = 10$, $(\Lambda_u, \Lambda_v) \sim Gamma(1, 100)$, $\gamma = 0.6$, and $\epsilon_P = 2\epsilon_I$. The learning rate η_0 is 5×10^{-6} for both MovieLens datasets (MovieLens-100K and MovieLens-1M), and 10^{-7} for Netflix dataset, which are chosen according to the validation set.

4.2. Experimental results

Since the number of training iterations is positively correlated with the transmission overhead in our framework, there is a trade-off between recommendation performance and computational cost while the privacy cost is scaled up by the number of iterations as well. Therefore, we show the experimental results by presenting the learning curve (i.e., RMSE vs. “number of iterations” up to 100) for each model. Based on the results, service providers are allowed to choose the most appropriate numbers in their applications.

Task 1*.* The learning curves on ML-100K are in Fig. 5a and b, the ones on ML-1M are in Fig. 6a and b, and the ones on Netflix are in Fig. 7a and b. It can be seen that the distance between ISGLD ($\epsilon = 4$) and ISGLD ($\epsilon = 2$) gets much greater than the ranges that the same curves in SDMF spread. Such results reflect that the performance of SDMF is less sensitive to the privacy budget than ISGLD. This result also shows that SDMF has the advantage to maintain the accuracy while still enjoying the additional protection on *existence* and *model* privacy. While comparing two privacy budgets, in Figs. 5a, 6a, and 7a, we can see ϵ_g has a negative relation to RMSE, which means more privacy yields lower accuracy on prediction. Although SDMF of $\alpha = \infty$ can already provide comparable performance with ISGLD ($\epsilon = 4$), controlling α enjoys arbitrary adjustment of the balance between privacy and accuracy to handle different circumstances. On the other hand, in Figs. 5b, 6b, and 7b, ϵ_I has less influence on the performance than ϵ_g. The difference between the privacy budgets is caused by their different meanings. Since ϵ_g is for the privacy level of the fake gradients, it directly affects the amount of noises that are injected to the model. In addition, ϵ_I is used to control the frequency of updating a rated and unrated item. Since SGLD has a nature to take the advantage of the randomness to prevent overfitting, the decreasing of ϵ_I will not largely increase the errors. It turns out to indicate that our two-stage RR algorithm can achieve good privacy without sacrificing too much accuracy. In a nutshell, the RMSE values eventually saturate to reasonable quality as the iteration number increases, and the performance of $\epsilon_g = 4$ is indeed comparable to the non-private baseline. Such results imply that the proposed SDMF can preserve the rating privacy of users while maintaining the performance of recommender systems. In our opinion, we believe the distributed framework that directly preserves the *value* privacy of data without adding large amount of noises is the key to our superior performance comparing to previous studies.

Task 2*.* The results of predicting one-class actions on ML-100K, ML-1M, and Netflix are shown in Fig. 8a,b and c, respectively. Comparing to the non-private baseline, the loss values in AUC on the three datasets are 0.03, 0.07, and 0.05. Similar to Task 1, ϵ_I has little influence on the performance. However, the gap between private version and non-private version is greater than that in Task 1. In our opinion, this is caused by the loss of information due to PRR (Section 3.2.1). In one-class rating action prediction, what PRR does is equivalent to deleting some rating actions from the training set, and thus generates more performance damage than in numerical rating prediction. Nevertheless, as we are the first to exploit differential privacy in one-class recommendation task, such loss is acceptable because the accuracy is still high enough to be with certain utility.

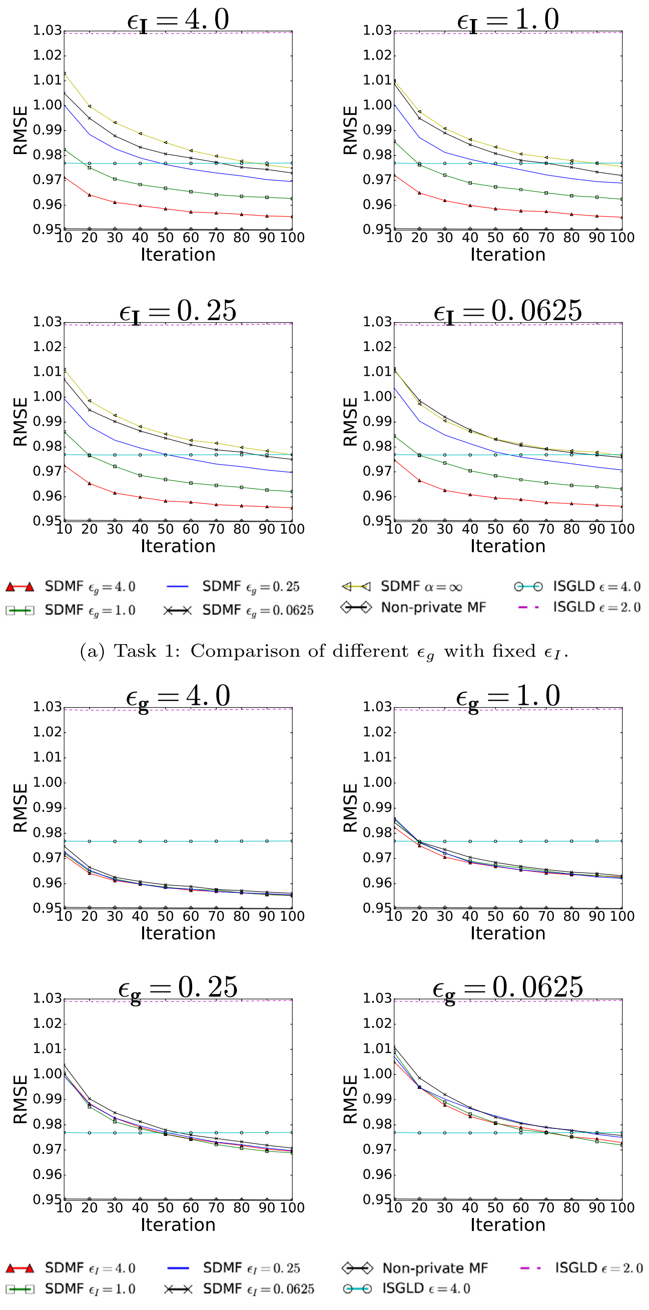
5. Application to factorization machine

Here we discuss how to apply the proposed SDCF to Factorization Machines (FM) [25], which is another widely-used and robust model for recommendation systems. We choose FM not only because it can be optimized with gradient descent but also because many of CF-based methods such as MF, SVD++, and K-Nearest Neighbors (KNN) can actually be expressed as Factorization Machines with different settings of parameters [25]. The basic idea of FM is to learn the interactions between variables. To simply explain FM, let us assume the input data of the prediction task is a matrix $X \in \mathbb{R}^{n \times m}$, where $x \in \mathbb{R}^m$ describes a sample with m variables. To model the interactions of d-order, for the cth column of total m columns in X, FM will learn a bias weight w_c and d latent vectors $v_c^l \in \mathbb{R}^{k_l}$, where k_l is the number of dimensions for the lth latent vector. Therefore, the predicted target (e.g., the rating scores), denoted by $\hat{y}(x)$, is defined as

$$\hat{y}(x) = w_0 + \sum_{c=1}^{m} w_c x_c + \sum_{l=2}^{d} \sum_{c_1=1}^{m} \cdots \sum_{c_d=c_{d-1}+1}^{m} \left(\prod_{\zeta=1}^{l} x_{c_\zeta} \right) \sum_{a=1}^{k_l} \left(\prod_{\zeta=1}^{l} v_{c_\zeta a}^{l} \right).$$

For recommendation systems, each rating will be formulated as an x such that the information about the user and the item can be represented in a format of dummy code as shown in below:

$$x = (\underbrace{0, \cdots, 0, 1, 0, \cdots, 0}_{|\mathcal{U}|}, \underbrace{0, \cdots, 0, 1, 0, \cdots, 0}_{|\mathcal{V}|}, \underbrace{x_{|\mathcal{U}|+|\mathcal{V}|+1}, \cdots, x_m}_{\text{other variables}}),$$



(a) Task 1: Comparison of different ϵ_g with fixed ϵ_I.

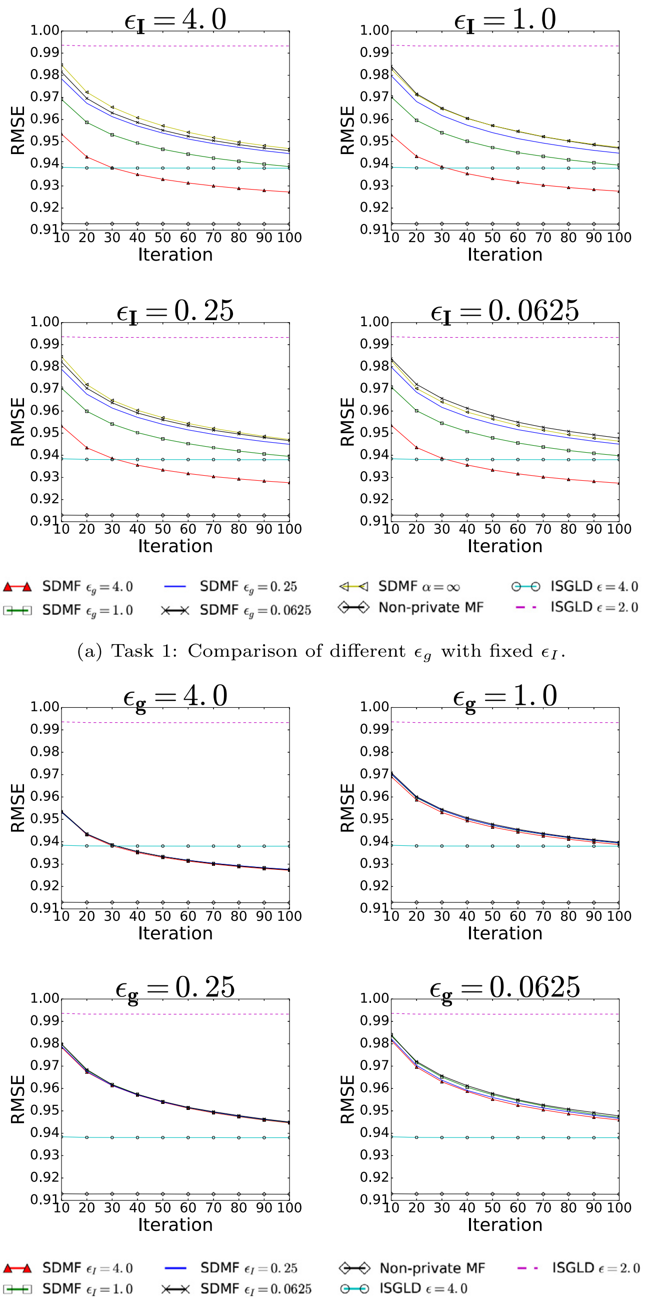
(b) Task 1: Comparison of different ϵ_I with fixed ϵ_g.

Fig. 5. MovieLens-100K.

where other variables describe information such as time, age of user, categories of item, and other attributes about the rating. To incorporate SDCF with FM, we can consider v_c^I and w_c for all $c \le |\mathcal{U}|$ as personal elements while other bias weights and latent vectors are treated as public elements. In this way, users can download public elements, update their own v_c^I and w_c, and then send gradients for other bias weights and latent vectors to the server.

6. Conclusions and future work

This paper proposes a framework, SDCF, to preserve the privacy of *value*, *model*, and *existence* in recommender systems under a distributed setting. The key contribution is the proposed two-stage Randomized Response algorithm, which is able to compute *gradients* of matrix factorization for unrated items in a secure manner under client-server context. More impor-

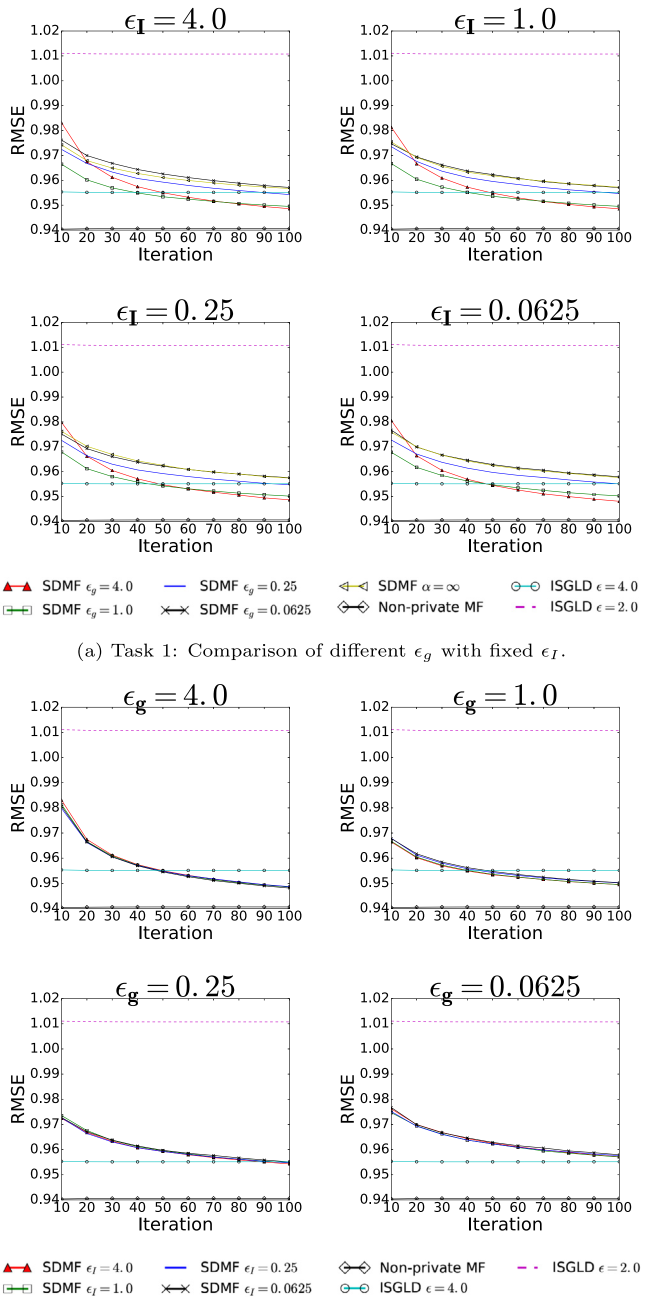


(a) Task 1: Comparison of different ϵ_g with fixed ϵ_I.

(b) Task 1: Comparison of different ϵ_I with fixed ϵ_g.

Fig. 6. MovieLens-1M.

tantly, we have also proven that the proposed two-stage RR algorithm can theoretically meet differential privacy. Experimental results conducted on well-known MovieLens and Netflix datasets demonstrate that the ability of SDCF to be applied to both prediction tasks of numerical ratings and one-class rating actions without sacrificing too much accuracy. Moreover, low influence to the performance of the proposed parameter ϵ_I exhibits the effectiveness of Instantaneous Randomized Response in the training process. Therefore, we suggest that Randomized Response algorithms can be further applied to more tasks in machine learning to preserve privacy. Moreover, we show that the proposed randomization method can be also applied to a more general recommender, Factorization Machine, which allows features of users and items to boost recommendation accuracy. Last, to improve the feasibility of SDCF for practical usage, in the future we plan to work on the scenario of active learning to lower down the transmission overhead and the scenario of online learning to renew the model

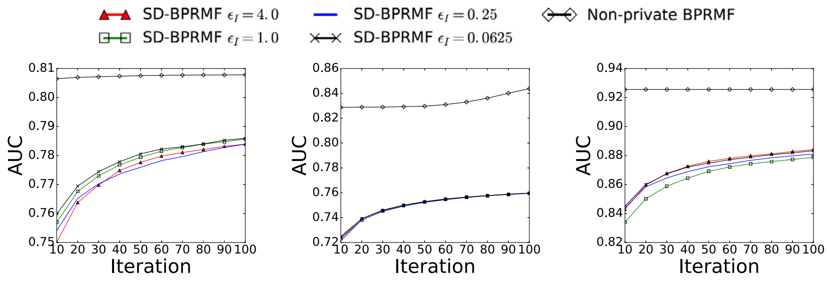


(a) Task 1: Comparison of different ϵ_g with fixed ϵ_I.

(b) Task 1: Comparison of different ϵ_I with fixed ϵ_g.

Fig. 7. Netflix (subsampled).

without re-training from scratch. Besides, we plan to extend this framework to content-based recommendation models as well as other models that also use gradient descent to learn latent representations (e.g., deep learning framework) so that the applications will not be limited to only recommender systems. In addition, we also plan to boost the efficiency of the proposed SDCF framework from two aspects. First, real-world recommender systems usually need to tackle a large number of users and items that make the user-item rating matrix too big. The efficiency of matrix factorization naturally become a critical issue. Therefore, our framework will further incorporate the technique of parallel SGD [34] that partitions the big matrix for fast SGD. Second, we will implement the idea of Fully Homomorphic Encryption [16] as a kind of preprocessing for the raw user-item ratings. By doing so, the computation cost of the proposed distributed perturbation mechanism can be lowered down by decreasing the volume of perturbation.



(a) ML-100K (b) ML-1M (c) Netflix (subsampled)

Fig. 8. Task 2: Comparison of different ϵ_I.

Acknowledgements

This work was sponsored by Ministry of Science and Technology (MOST) of Taiwan under grant 107-2636-E-006-002 and 107-2218-E-006-040, and also by Academia Sinica under grant AS-TP-107-M05.

References

[1] A. Afsharinejad, N. Hurley, Performance analysis of a privacy constrained knn recommendation using data sketches, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, ACM, 2018, pp. 10–18.
[2] A. Afsharinejad, N. Hurley, Performance analysis of a privacy constrained knn recommendation using data sketches, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, in: WSDM '18, 2018, pp. 10–18.
[3] A. Basu, J. Vaidya, H. Kikuchi, T. Dimitrakos, Privacy-preserving collaborative filtering on the cloud and practical implementation experiences, in: IEEE CLOUD, 2013, pp. 406–413.
[4] J. Bennett, S. Lanning, N. Netflix, The netflix prize, In KDD Cup and Workshop in Conjunction with KDD, 2007.
[5] A. Berlioz, A. Friedman, M.A. Kaafar, R. Boreli, S. Berkovsky, Applying differential privacy to matrix factorization, in: Proceedings of the 9th ACM Conference on Recommender Systems, ACM, 2015, pp. 107–114.
[6] A. Boutet, D. Frey, R. Guerraoui, A. Jégou, A.-M. Kermarrec, Privacy-preserving distributed collaborative filtering, Computing 98 (8) (2016) 827–846.
[7] J.A. Calandrino, A. Kilzer, A. Narayanan, E.W. Felten, V. Shmatikov, “you might also like:” privacy risks of collaborative filtering, in: 2011 IEEE Symposium on Security and Privacy, IEEE, 2011, pp. 231–246.
[8] J. Canny, Collaborative filtering with privacy, in: Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on, IEEE, 2002, pp. 45–57.
[9] Y.-R. Chen, A. Rezapour, W.-G. Tzeng, Privacy-preserving ridge regression on distributed data, Inf. Sci. 451–452 (2018) 34–49.
[10] C. Dwork, F. McSherry, K. Nissim, A. Smith, Calibrating noise to sensitivity in private data analysis, in: Theory of Cryptography Conference, Springer, 2006, pp. 265–284.
[11] C. Dwork, A. Roth, et al., The algorithmic foundations of differential privacy, Found. Trends Theor. Comput. Sci. 9 (3–4) (2014) 211–407.
[12] Ú. Erlingsson, V. Pihur, A. Korolova, Rappor: Randomized aggregatable privacy-preserving ordinal response, in: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security, ACM, 2014, pp. 1054–1067.
[13] A. Friedman, S. Berkovsky, M.A. Kaafar, A differential privacy framework for matrix factorization recommender systems, User Model. User-Adapt. Interact. 26 (5) (2016) 425–458.
[14] T. Guo, J. Luo, K. Dong, M. Yang, Differentially private graph-link analysis based social recommendation, Inf. Sci. 463–464 (2018) 214–226.
[15] J. Hua, C. Xia, S. Zhong, Differentially private matrix factorization, in: Proceedings of the 24th International Conference on Artificial Intelligence, AAAI Press, 2015, pp. 1763–1770.
[16] J. Kim, D. Koo, Y. Kim, H. Yoon, J. Shin, S. Kim, Efficient privacy-preserving matrix factorization for recommendation via fully homomorphic encryption, ACM Trans. Privacy Secur. (TOPS) 21 (4) (2018).
[17] Y. Koren, R. Bell, C. Volinsky, et al., Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.
[18] C.-T. Li, Y.-J. Lin, M.-Y. Yeh, Forecasting participants of information diffusion on social networks with its applications, Inf. Sci. 422 (2018) 432–446.
[19] J. Li, J.-J. Yang, Y. Zhao, B. Liu, M. Zhou, J. Bi, Q. Wang, Enforcing differential privacy for shared collaborative filtering, IEEE Access 5 (2017) 35–49.
[20] Z. Liu, Y.-X. Wang, A. Smola, Fast differentially private matrix factorization, in: Proceedings of the 9th ACM Conference on Recommender Systems, ACM, 2015, pp. 171–178.
[21] A. Machanavajjhala, A. Korolova, A.D. Sarma, Personalized social recommendations: accurate or private, Proc. VLDB Endow. 4 (7) (2011) 440–450.
[22] F. McSherry, I. 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, ACM, 2009, pp. 627–636.
[23] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, D. Boneh, Privacy-preserving matrix factorization, in: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, ACM, 2013, pp. 801–812.
[24] L. Qi, R. Wang, C. Hu, S. Li, Q. He, X. Xu, Time-aware distributed service recommendation with privacy-preservation, Inf. Sci. 480 (2019) 354–364.
[25] S. Rendle, Factorization machines with libfm, ACM Trans. Intell. Syst. Technol. (TIST) 3 (3) (2012) 57.
[26] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, in: Proceedings of the twenty–fifth conference on uncertainty in artificial intelligence, AUAI Press, 2009, pp. 452–461.
[27] F. Ricci, L. Rokach, B. Shapira, Introduction to Recommender Systems Handbook, Springer, 2011.
[28] Y. Shen, H. Jin, Privacy-preserving personalized recommendation: an instance-based approach via differential privacy, in: 2014 IEEE International Conference on Data Mining, IEEE, 2014, pp. 540–549.
[29] Y. Shen, H. Jin, Epicrec: Towards practical differentially private framework for personalized recommendation, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016, pp. 180–191.
[30] D. Vallet, A. Friedman, S. Berkovsky, Matrix factorization without user data retention, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2014, pp. 569–580.
[31] J. Wang, A. Arriaga, Q. Tang, P.Y. Ryan, Facilitating privacy-preserving recommendation-as-a-service with machine learning, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, in: CCS '18, 2018, pp. 2306–2308.

[32] M. Welling, Y.W. Teh, Bayesian learning via stochastic gradient langevin dynamics, in: Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 681–688.

[33] Y. Xin, T. Jaakkola, Controlling privacy in recommender systems, in: Advances in Neural Information Processing Systems, 2014, pp. 2618–2626.

[34] Y. Zhuang, W.-S. Chin, Y.-C. Juan, C.-J. Lin, A fast parallel SGD for matrix factorization in shared memory systems, in: Proceedings of the 7th ACM Conference on Recommender Systems, in: RecSys '13, 2013, pp. 249–256.