Efficient Federated Matrix Factorization Against Inference Attacks

Di Chai, Hong Kong University of Science and Technology, China
Leye Wang, MOE Key Lab of High Confidence Software Technologies, Peking University, China
Kai Chen, Hong Kong University of Science and Technology, China
Qiang Yang, WeBank AI Lab, Hong Kong University of Science and Technology, China

Abstract

In the process of personalized recommendation to the user, the recommendation system usually needs to upload the user's privacy information (such as the user's scoring behavior/scoring data) to the server, and the server uses the user's data for modeling, so as to make personalized recommendation to the user. However, in this way, the user's private information is directly exposed, and the server can use this information to conduct inference attacks on the user, so as to learn the user's private attributes (such as age and gender). Therefore, in this paper, the author proposes an effective federated matrix decomposition method, which can effectively protect users from inference attacks. The key idea is to confuse one user's rating with another's to minimize the leakage of private attributes with a given distortion budget that limits the overhead of recommendation losses and system efficiency. In the process of confusion, differential privacy is applied to control the information leakage between users. Homomorphic encryption is used to protect intermediate results during training. In this paper, we reproduce the paper of Chai et al. and the framework is implemented and tested on real-world data sets(MovieLens 100K). The results show that, compared with no privacy protection, the proposed method can reduce the inference attack accuracy is 28.7% lower on average, and the safety is 12% higher than that of the original paper. while maintaining the recommending performance and only brings an average utility loss of 2.6%, which is only 0.2% lower than the original paper.

Keywords： Federated learning, Inference attack, Matrix factorization.

1 Introduction

In this era of big data, recommendation system has become an indispensable part of our life. As an effective tool to alleviate the problem of "information overload", the recommendation system makes analysis and prediction according to users' behaviors and preferences, and then provides personalized recommendation services to users. To build an effective recommendation system, a large amount of data is usually required. However, these data usually contain users' private information, such as user ratings. Rating information is very private because it represents the user's preferences and indicates the user's private attributes, such as age and gender. Therefore, how to protect the user's privacy information in the recommendation system has become a hot research topic.

Matrix decomposition (MF)[1] is one of the most widely used recommendation techniques. It divides the user's score matrix into two low-dimensional matrices, namely the user's feature matrix and the item's feature matrix. Many researchers have studied privacy protection methods based on MF, such as encryption based methods[2], differential privacy based methods[3], and federated learning based methods[4]. Encryption-based methods protect user privacy by encrypting data, but the complex computation of encryption and decryption brings a lot of time overhead. The method based on differential privacy protects user privacy by minimizing the probability of identifying any user data. Although this algorithm ensures data security to a certain extent and improves communication efficiency, it will lead to a decline in the accuracy of recommendation results. Federated learning approaches protect user privacy by storing user data locally and performing federated model training among all users. Chai et al.[4] presented the first secure federated matrix factorization. They demonstrated that gradients reveal privacy data of users, and therefore homomorphic encryption is necessary to protect intermediate results in federated learning methods.

In this paper, the author focuses on federated matrix factorization for the following reasons[5]:

• Federated learning can collect data from multiple parties, making it easier to meet privacy regularization (e.g. GDPR).

• Federated Learning based solution requires only a semi-honest server, which has more practicability.

• Federated learning-based schemes can be combined with homomorphic encryption and differential privacy approaches to achieve a more secure scheme.

Based on the above reasons, the author proposes an efficient federated matrix decomposition method, called EIFedMF, which can effectively resist inference attacks. The key idea is to learn a obfuscation function that confuses one user's private information with another, minimizing the mutual information between uploaded information and private attributes under a data distortion budget.

2 Related Works

2.1 Homomorphic Encryption

Homomorphic encryption(HE) is a kind of special secret method, its idea originated from private homomorphism, it can not only encrypt data, but also realize the four operations between ciphertext. We say that the HE algorithm is homomorphic over the operation ''★", if the following equation holds[6]:

$$[[m_1]]\bigstar[[m_2]] = [[m_1 \bigstar m_2]] \tag{1}$$

Where $[[\cdot]]$ is the encryption function, m is the message. Because HE has good homomorphic encryption properties, it is also widely used in federated learning to protect the intermediate results in the interaction process of different participants.

2.2 Federated Matrix Factorization

Matrix Factorization(MF) algorithm decomposes a complete rating matrix into represent the product of user-specific latent feature matrix and item feature matrix. Given the rating information $r_{ij} : (i, j) \in \mathcal{M}$ The

calculation formula of user's predicted rating of item:

$$\hat{r}_{ui} = \boldsymbol{U}_{u\cdot}\boldsymbol{V}_{i\cdot}^{T} \tag{2}$$

where $\boldsymbol{U}_{u\cdot} \in \mathbb{R}^{1\times d}$ and $\boldsymbol{V}_{i\cdot}^{T} \in \mathbb{R}^{1\times d}$ are the d-dimensional specific latent feature vector of user u and item i.

Stochastic gradient decent(SGD) is used to train the model. Given the rating information $r_{ui} : (u,i) \in \mathcal{M}$, Equation (3) shows the the optimization problem of MF and Equations (4) – (7) are the parameter updating formulas using SGD.

$$\min_{\boldsymbol{U},\boldsymbol{V}} \frac{1}{\mathcal{M}}(r_{ui} - \boldsymbol{U}_{u\cdot}\boldsymbol{V}_{i\cdot}^{T})^2 + \lambda||\boldsymbol{U}||_2^2 + \mu||\boldsymbol{V}||_2^2 \tag{3}$$

$$\nabla\boldsymbol{U}_{u\cdot} = -2(r_{ui} - \boldsymbol{U}_{u\cdot}\boldsymbol{V}_{i\cdot}^{T})\boldsymbol{V}_{i\cdot} + 2\lambda\boldsymbol{U}_{u\cdot} \tag{4}$$

$$\nabla\boldsymbol{V}_{i\cdot} = -2(r_{ui} - \boldsymbol{U}_{u\cdot}\boldsymbol{V}_{i\cdot}^{T})\boldsymbol{U}_{u\cdot} + 2\mu\boldsymbol{V}_{i\cdot} \tag{5}$$

$$\boldsymbol{U}_{u\cdot} = \boldsymbol{U}_{u\cdot} - \gamma\nabla\boldsymbol{U}_{u\cdot} \tag{6}$$

$$\boldsymbol{V}_{i\cdot} = \boldsymbol{V}_{i\cdot} - \gamma\nabla\boldsymbol{V}_{u\cdot} \tag{7}$$

where $\boldsymbol{U} \in \mathbb{R}^{n\times d}$ and $\boldsymbol{V} \in \mathbb{R}^{m\times d}$ are the d-dimensional user-specific latent matrix and items-specific latent matrix, n is the number of users and m is the number of items, λ and μ are the tradeoff on the regularization term.

In the federated matrix factorization(FedMF) strategy, the model is trained through user joint training. Users keep rating data locally and send the intermediate results (i.e., gradients) to the server. For the interaction matrix $\boldsymbol{X}$, the value of x_{ui} is:

$$x_{ui} = \begin{cases} 1, & \text{if } (u,i) \in \mathcal{R} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Procedure 1 Federated Matrix Factorization with HE According to Matrix $\boldsymbol{X}$

Initialize: The server initializes $\boldsymbol{V}$ and holds the public key and each user u initializes $\boldsymbol{U}_{u\cdot}$ and holds the same secret key

The server keeps the latest $[[\boldsymbol{V}]]$ ▷$[[\cdot]]$ denotes the homomorphic encryption function

Output: $\boldsymbol{U}$ and $\boldsymbol{V}$

User local update:

Each user u downloads $\{[[\boldsymbol{V}_{i\cdot}]]|x_{ui}=1\}$ from server, decrypts and gets $\boldsymbol{V}_{i\cdot}$

LocalGradient$\leftarrow \{0\}^d$,where d is the dimensionality

for $i \in \{i|x_{ui}=1\}$ **do**

  if $(u,i) \in \mathcal{R}$ **then**

    $LocalGradient \leftarrow LocalGradient - 2(r_{ui} - \boldsymbol{U}_{u\cdot}\boldsymbol{V}_{i\cdot}^{T})V_{i\cdot}$

    $\nabla\boldsymbol{V}_{i\cdot} \leftarrow -2(r_{ui} - \boldsymbol{U}_{u\cdot}\boldsymbol{V}_{i\cdot}^{T})\boldsymbol{U}_{u\cdot} + 2\lambda\boldsymbol{V}_{i\cdot}$

  else

    $\nabla\boldsymbol{V}_{i\cdot} \leftarrow \{0\}^d$

  end

  Each user u updates its user-specific latent feature vector:$\boldsymbol{U}_{u\cdot} = \boldsymbol{U}_{u\cdot} - \gamma(LocalGradient + 2\lambda\boldsymbol{U}_{u\cdot})$

  Each user u encrypts and sends $\{[[-\gamma\nabla\boldsymbol{V}_{i\cdot}]]|x_{ui}=1\}$ to the server,where γ is the learning rate

end

2.3 Oblivious Transfer

Oblivious transfer(OT) is a cryptography protocol that is widely used in Secure Multi-Party Computation (SMPC). It was proposed by Rabin in 1981. A typical OT protocal is 1-out-of-2 oblivious transfer, which was proposed by Even et al. in 1985. In this form of casual transmission model, the sender sends two messages (m_1, m_2) to the receiver each time, the receiver provides an input, and obtains the output information according to the input. After the end of the protocol, the receiver gets the desired message (m_1 or m_2), while the sender does not know which message the receiver finally gets.

This paper uses the 1-out-of-2 oblivious transfer,which can be naively implemented by doing n times of 1-out-of-2 oblivious transfer. And we adopt the protocol of Naor et al.[7], which reduces the complexity of 1-out-of-n OT from $O(n)$ to $O(\log(n))$.

3 Method and Implementation Details

3.1 Notation

Table 1: Some Notations

n	user number
m	item number
$\mathcal{R} = (u, i)$	The set of (u, i) pairs in the training set
$\boldsymbol{X} \in \mathbb{R}^{n \times m}$	the matrix of users interaction with items
$\boldsymbol{U} \in \mathbb{R}^{n \times d}$	user-specific latent feature matrix
$\boldsymbol{V} \in \mathbb{R}^{m \times d}$	item-specific latent feature matrix
$\boldsymbol{U}_{u\cdot} \in \mathbb{R}^{1 \times d}$	user-specific latent feature vector
$\boldsymbol{V}_{i\cdot} \in \mathbb{R}^{1 \times d}$	item-specific latent feature vector
d	dimensional of latent feature vector
$\mathcal{I}_u$	the set of items rated by user u
λ, μ	the tradeoff on the regularization term
γ	learning rate

3.2 Overview

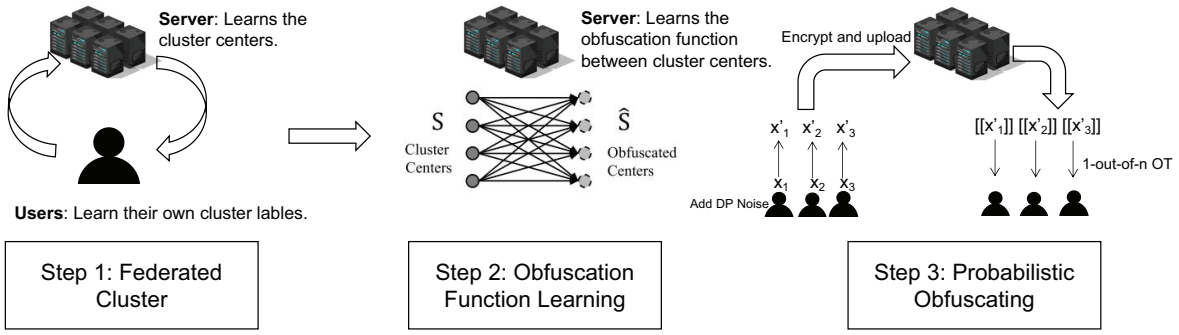

Figure 1: Overview of the method

3.3 Reduce the Information Leakage

The average information leakage of a set of input $\boldsymbol{X}$ regarding a private target Y is given by the $I(\boldsymbol{X}; Y)$, which is the mutual information between $\boldsymbol{X}$ and Y[8].

In this paper, we want to reduce the mutual information:

$$I(\hat{\boldsymbol{X}};Y)=\sum_{\hat{x}\in\hat{\boldsymbol{X}},y\in Y}p(\hat{x},y)\log(\frac{p(\hat{x},y)}{p(\hat{x})p(y)}) \tag{9}$$

where $\hat{\boldsymbol{X}}$ is the obfuscated rating-pair matrix and Y is users' private attribute.

Given the rating-pair matrix $\boldsymbol{X}$, we use an obfuscation function $p_{\hat{\boldsymbol{X}}|\boldsymbol{X}}$ to generate $\hat{\boldsymbol{X}}$. Then the joint probability $p(\hat{x},y)$, and the marginal probability $p(\hat{x})$ can be represented as:

$$\begin{aligned}p_{\hat{\boldsymbol{X}},Y}(\hat{x},y)&=\sum_{x\in\boldsymbol{X}}p_{\hat{\boldsymbol{X}}|\boldsymbol{X}}(\hat{x}|x)p_{\boldsymbol{X},Y}(x,y)\\p_{\hat{\boldsymbol{X}}}(\hat{x})&=\sum_{x\in\boldsymbol{X},y\in Y}p_{\hat{\boldsymbol{X}}|\boldsymbol{X}}(\hat{x}|x)p_{\boldsymbol{X},Y}(x,y)\end{aligned} \tag{10}$$

Combined with the above joint and marginal probability, the mutual information between the obfuscated rating-pair matrix $\hat{X}$ and the private attribute Y is:

$$I(\hat{\boldsymbol{X}};Y)=\sum_{\hat{x}\in\hat{\boldsymbol{X}},y\in Y}\left[\sum_{x\in\boldsymbol{X}}p_{\hat{\boldsymbol{X}}|\boldsymbol{X}}(\hat{x}|x)p_{\boldsymbol{X},Y}(x,y)\right]\cdot\log(\frac{\sum_{x'\in\boldsymbol{X}}p_{\hat{\boldsymbol{X}}|\boldsymbol{X}}(\hat{x}|x')p_{\boldsymbol{X},Y}(x',y)}{p(y)\cdot\sum_{x''\in\boldsymbol{X},y'\in Y}p_{\hat{\boldsymbol{X}}|\boldsymbol{X}}(\hat{x}|x'')p_{\boldsymbol{X},Y}(x'',y')}) \tag{11}$$

For a given dataset with privacy attribute Y, we can empirically determine $p_Y(y)$ and $p_{\boldsymbol{X},Y}(x,y)$. By minimizing the mutual information between the $\hat{\boldsymbol{X}}$ and Y, the information leakage is reduced.

3.4 Bound the Data Distortions

The discrepancy between $\boldsymbol{X}$ and $\hat{\boldsymbol{X}}$ can be categorized into two situations:

(1) $x_{ui}=0,\hat{x}_{ui}=1$: User u did not rate item i, but after the obfuscation, we need to upload user u's encryption gradient for item i. The gradient upload is an additional overhead.

(2) $x_{ui}=1,\hat{x}_{ui}=0$: User u interacts with item i, but after the obfuscation, we don't need to upload user u's encryption gradient for item i. So we lose an effective gradient, which leads to a recommendation loss.

In this paper, we measure recommendation loss by the amount of data lost and efficiency overhead by the amount of data increased. For the given vector x_u and the obfuscation vector $\hat{x}_u$, we use Equation (12) to measure therecommendation loss, and use Equation (13) to quantify the time consumption overhead.

$$dist_{rec}(x_u,\hat{x}_u)=\frac{||x_u||-||x_u-\hat{x}_u||_{x_u>\hat{x}_u}}{||x_u||} \tag{12}$$

$$dist_{eff}(x_u,\hat{x}_u)=\frac{||\hat{x}_u-||x_u||}{||x_u||} \tag{13}$$

where $||\cdot||$ is the L1-norm operation, $||\cdot||_{condition}$ is the L1-norm that satisfies the $condition$.

3.5 Obfuscation Function Learning

To achieve the goal of minimizing mutual information between the obfuscated rating-pair matrix $\hat{\boldsymbol{X}}$ and the private data Y, we constrain the data distortion between $\hat{\boldsymbol{X}}$ and $\boldsymbol{X}$ to maintain the utility. We can learn an

obfuscation function $p_{\hat{\boldsymbol{X}}|\boldsymbol{X}}$ through the following constrained optimization:

$$\begin{aligned} \min_{p_{\hat{\boldsymbol{X}}|\boldsymbol{X}}} \quad & I(\hat{\boldsymbol{X}};Y) \\ s.t. \quad & \begin{cases} 0 \leqslant p_{\hat{\boldsymbol{X}}|\boldsymbol{X}}(\hat{x}|x) \leqslant 1 \\ \sum\limits_{\hat{x}} p_{\hat{\boldsymbol{X}}|\boldsymbol{X}}(\hat{x}|x) = 1 \\ E_{\hat{\boldsymbol{X}},\boldsymbol{X}}(dist_{rec}(x,\hat{x})) \leqslant \alpha \\ E_{\hat{\boldsymbol{X}},\boldsymbol{X}}(dist_{eff}(x,\hat{x})) \leqslant \beta \end{cases} \end{aligned} \tag{14}$$

where α and β are two hyperparameters controlling the utility of $\hat{\boldsymbol{X}}$ in recommendation andefficiency.

As we all know, in the real world, the number of users and item is very large, if we want to calculate a user confusion matrix is very difficult.

To reduce the complexity, we cluster the users into a limited and fixed number of groups. Then we learn the obfuscation function between clusters instead of individual users. Specifically, we firstly cluster users into k sets according to the matrix $\boldsymbol{X}$, and then we learn the obfuscation function between different groups of users using the cluster centroids $\boldsymbol{S} = \{s_0, s_1, ..., s_k\}$. We can get a new constrained optimization：

$$\begin{aligned} \min_{p_{\hat{\boldsymbol{S}}|\boldsymbol{S}}} \quad & I(\hat{\boldsymbol{S}};Y) \\ s.t. \quad & \begin{cases} 0 \leqslant p_{\hat{\boldsymbol{S}}|\boldsymbol{S}}(\hat{s}|s) \leqslant 1 \\ \sum\limits_{\hat{s}} p_{\hat{\boldsymbol{S}}|\boldsymbol{S}}(\hat{s}|s) = 1 \\ E_{\hat{\boldsymbol{S}},\boldsymbol{S}}(dist_{rec}(s,\hat{s})) \leqslant \alpha \\ E_{\hat{\boldsymbol{S}},\boldsymbol{S}}(dist_{eff}(s,\hat{s})) \leqslant \beta \end{cases} \end{aligned} \tag{15}$$

It has been proved by Calmon et al.[9] that the constrained optimization in Equations (14-15) is a convex optimization problem.

3.5.1 Federated Clustering

We propose a federated clustering method based on the k-means algorithm.

3.6 Probabilistic Obfuscation

After obtaining the cluster centers, the server will solve the convex optimization in Equation (15) and send the obfuscation function $p_{\hat{\boldsymbol{S}}|\boldsymbol{S}}$ to each user. In addition, we also need to fill in the gap between clusters and users to perform obfuscation on individual users. We firstly obfuscate one user from his own cluster s to another cluster s', then randomly select one user's data from cluster s' as the obfuscation result.

However, there are some drawbacks to this obfuscation: (1) User u have direct access to user w's data, which is not secure. (2) User u's selection of user w is transparent to the server.

To improve the security of the system, we use differential privacy(DP) and oblivious transfer(OT) to design secure model, which consists of three steps:

(1) Step 1: User u adds DP noise to local data x_u getting x'_u, then encrypts and sends $[[x'_u]]$ to the server.

(2) Step 2: User u get the local cluster s, obfuscate to another cluster $\hat{s}$ according to the obfuscation funtion

Procedure 2 Federated k-means Algorithm

Initialize: The server holds the public key, initializes cluster centers $C \in \{0,1\}^{k\times m}$, k is the number of clusters and m is the number of items. Users hold the same secret key.
Output:Converged cluster centers $\boldsymbol{C}$.
The server sends encypted centers $[[\boldsymbol{C}]]$ to all users
for $u = 1, \ldots, n$ **do**
  User u downloads $[[\boldsymbol{C}]]$ from server, decrypts and gets $\boldsymbol{C}$
  User u decides local cluster label $L_u = argmin_w \|x_u - C_w\|_2^2$
  The location of user u in the cluster:$\boldsymbol{H}_{u\cdot} = \{0\}^k, \boldsymbol{H}_{u\cdot}[L_u] \leftarrow 1$
  User u encrypts and uploads $[[\boldsymbol{H}_{u\cdot}]]$
end
The server counts the number of users in each cluster
The server compute $[[\boldsymbol{H}]] = \sum_{u\cdot}[[\boldsymbol{H}_{u\cdot}]]$, and return $[[\boldsymbol{H}]]$ to all users
for $u = 1, \ldots, n$ **do**
  User u downloads $[[\boldsymbol{H}]]$ from server, decrypts and gets $\boldsymbol{H}$
  User u compute local contribution on cluster center:$\boldsymbol{C}_{u\cdot} \leftarrow \{0\}^{k\times m}, \boldsymbol{C}_{u\cdot}[L_u] \leftarrow \dfrac{x_u}{\boldsymbol{H}[L_u]}$
  User u encrypts and uploads $[[\boldsymbol{C}_{u\cdot}]]$
end
The server computs the new centers $[[\boldsymbol{C}]] \leftarrow \sum_{u=1}^{n}[[\boldsymbol{C}_{u\cdot}]]$
Repeat the algorithm until the cluster centers are converged

$p_{\hat{\boldsymbol{S}}|\boldsymbol{S}}$.

(3) Step 3: User u performs the 1-out-of-n OT with the server to randomly get a user w from cluster $\hat{s}$, then uses x'_w as the obfuscation result.

3.6.1 Differential Privacy

User u adds DP noise to local data, such that the randomized output on each cluster satisfies the definition of ε-differential privacy (ε-DP).

We denote dateset $\boldsymbol{D}$ as a subset of users' data from one cluster. Based on $\boldsymbol{D}$, we build dataset $\boldsymbol{D'}$ by randomly drop or add one user from the same cluster. We denote function $f : \boldsymbol{D} \rightarrow \boldsymbol{D'}$ as the random pick function. Define function $A(\boldsymbol{D}) = (f(\boldsymbol{D}) + N) \mod 2$, where $N \in \{0,1\}^m$ is discreate noise. And we generate N using Equation (16), where $1 \leqslant i \leqslant m$. We set $b = \dfrac{\Delta f}{\varepsilon}, \Delta f = \max_{x\in\boldsymbol{D}, y\in\boldsymbol{D'}} \sum_{i=1}^{m} |x_i - y_i|$, then function A satisfies the definition of ε-DP.

$$Pr(N_i = 0) = \frac{1}{1 + e^{-\frac{1}{b}}}, Pr(N_i = 1) = \frac{e^{-\frac{1}{b}}}{1 + e^{-\frac{1}{b}}} \tag{16}$$

Assume $f(\boldsymbol{D}) = (x_1, x_2, ..., x_m)^T$ and $f(\boldsymbol{D'}) = (y_1, y_2, ..., y_m)^T$. Denote $\boldsymbol{O} = (z_1, z_2, ..., z_3)$ as a possible

output of function A. Then we have:

$$
\begin{aligned}
Pr[A(\boldsymbol{D})=\boldsymbol{O}] &= \prod_{i=1}^{m} \frac{1}{1+e^{-\frac{\varepsilon}{\Delta f}}} e^{-\frac{\varepsilon}{\Delta f}|x_i-z_i|} \\
Pr[A(\boldsymbol{D}')=\boldsymbol{O}] &= \prod_{i=1}^{m} \frac{1}{1+e^{-\frac{\varepsilon}{\Delta f}}} e^{-\frac{\varepsilon}{\Delta f}|y_i-z_i|} \\
\frac{Pr[A(\boldsymbol{D})=\boldsymbol{O}]}{Pr[A(\boldsymbol{D}')=\boldsymbol{O}]} &= \frac{\prod_{i=1}^{m} \frac{1}{1+e^{-\frac{\varepsilon}{\Delta f}}} e^{-\frac{\varepsilon}{\Delta f}|x_i-z_i|}}{\prod_{i=1}^{m} \frac{1}{1+e^{-\frac{\varepsilon}{\Delta f}}} e^{-\frac{\varepsilon}{\Delta f}|y_i-z_i|}} \\
&= \prod_{i=1}^{m} e^{-\frac{\varepsilon}{\Delta f}(|x_i-z_i|-|y_i-z_i|)} \\
&= e^{\varepsilon \cdot \frac{\prod_{i=1}^{m}(|y_i-z_i|-|x_i-z_i|)}{\Delta f}}
\end{aligned}
\tag{17}
$$

According to the triangle inequality, we can know $|a+b|-|a| \leqslant |b|$. So we have:

$$
\begin{aligned}
\frac{Pr[A(\boldsymbol{D})=\boldsymbol{O}]}{Pr[A(\boldsymbol{D}')=\boldsymbol{O}]} &= e^{\varepsilon \cdot \frac{\prod_{i=1}^{m}(|y_i-z_i|-|x_i-z_i|)}{\Delta f}} \\
&\leqslant e^{\varepsilon \cdot \frac{\prod_{i=1}^{m}(|y_i-x_i|)}{\Delta f}} \\
&\leqslant e^{\varepsilon}
\end{aligned}
\tag{18}
$$

Thus, function A satisfies the definition of ε-DP.

In order to blind the server with which user is finally chosen as the obfuscation result, we use 1-out-of-n oblivious transfer technology which guarantees that the server learns nothing about the user u' choice and the user u only learn the data he chooses.

3.7 Privacy Analysis

We assume the opponent is a semi-honest server, and the method mentioned in this paper can effectively protect the privacy of the entire model:

(1) Federated kmeans clustering: The cluster centers leak no private information because it represents the character that belongs to a group of users, which is not private.

(2) Obfuscation Function Learning: the server learns the obfuscation function based on the clusters' centers. We already know, the cluster centers leak no private information. In the same way, the obfuscation function also leaks nothing private.

Procedure 3 Apply DP Noise(Compute Δf)

Initialize: ε
Output:Δf for each user
for $u = 1, \ldots, n$ **do**
  User u encrypts $[[x_u]]$ and cluster label L_u,send them to server
end
for $u = 1, \ldots, n$ **do**
  for $w = 1, \ldots, n$ **do**
    The server computes$[[D_{uw}]] = [[x_u]] - [[x_w]]$ ▷user u and user w
    The server random shuffles $[[D_{uw}]]$ ▷$[[D_{uw}]]$ is a vector
  end
  The server random shuffles $[[D_u]]$ together with $[[L]]$
  The server sends $[[D_u]]$ and $[[L]]$ to user u
end
for $u = 1, \ldots, n$ **do**
  User u decrypts $[[D_u]]$ and $[[L]]$,$\Delta f \leftarrow 0$
  for $w_1 = 1, \ldots, n$ **do**
    for $w_2 = w_1 + 1, \ldots, n$ **do**
      if $L_{u\cdot} = L_{w_1} = L_{w_2}$ **then**
        $dist = \sum_{i=1}^{m} |D^i_{L_{w_1} L_{w_2}}|$
        if $\Delta f < dist$ **then**
          $\Delta f = dist$
        end
      end
    end
  end
  User u gets Δf,and applies DP noise locally
end

4 Experimental Results

4.1 Experiment Settings

We use the MovieLens dataset that contains ratings of 1682 movies made by 943 users, and we focus on protecting users'gender, age, and occupation in this dataset.

Table 2 shows the processing of attribute labels.

In the federated user clustering, we set $k = 10$, The α and β in Equation (15) are set to 0.2 and 1.0, ε is set to 0.01. We use SEAL-CKKS encryption as the homomorphic encryption method, and the poly modulus degree is set to 8192, he bandwidth of the communication is 1 GB/s. The programming language is Python,

Procedure 4 Secure Random Select Using OT

for $u = 1, \ldots, n$ **do**
  User u encrypts $[[x_u]]$ and cluster $[[L_u]]$,sends them to server
end
The server holds $[[x_u]]$ as the messages and sends $[[L]]$ to all the users
for $u = 1, \ldots, n$ **do**
  User u decrypts $[[L]]$ and random selects one index s_u such that $L_{s_u} = L_u$
  User u performs an 1-out-if-n OT with the server,and gets x_{s_u}
end

Table 2: Procecssing of Attributes

Attribute	#Classes	Values
Gender	2	{Male, Female}
Age	5	$\{\leqslant 18, (18, 30], (30, 40], (40, 50], 50 <\}$
Occuption	21	administrator, artist, doctor, etc.

and we use SEAL-Python in the implementation, which is a python binding for the Microsoft SEAL library.

The instructions for the resulting table:

- EIFedMF-Single: Protect a single attribute.
- EIFedMF-All: Protect all the attributes.
- PartText Solution(PT): Users use real rating-pair without any protection.
- Random Increase (RI-p): Users randomly pick p of non-rated items as obfuscation to resist the inference attacks.
- Random Flip (RF-p): Users randomly flip the values of the matrix $\boldsymbol{X}$ with probability p.

4.2 Results and Analysis

4.2.1 Inference Attack Accuracy

We have performed inference attack experiments using three types of machine learning models: Naive Bayesian(NB), Support Vector Machine(SVM), and Gradient Boosting Decision Tree (GBDT) and We use the 10-fold cross-validation method.

Table 3 shows the average inference attack accuracy.

Table 3: Inference Attack Accuracy

Attribute	Model	PT	RI-0.1	RI-0.3	RI-0.5	RI-0.7	RI-0.9	RF-0.1	RF-0.3	RF-0.5	RF-0.7	RF-0.9	EIFedMF Single	EIFedMF All
Gender	NB	0.69565	0.66383	0.67234	0.65852	0.69565	0.7105	0.66066	0.69672	0.65856	0.67441	0.64583	0.56834	0.57159
	SVM	0.72216	0.7105	0.7105	0.7105	0.7105	0.7105	0.7105	0.7105	0.7105	0.7105	0.7105	0.67127	0.64368
	GBDT	0.73342	0.72228	0.7016	0.68869	0.69723	0.67444	0.71094	0.70435	0.69565	0.70764	0.70543	0.64941	0.62077
Age	NB	0.42099	0.39765	0.42734	0.40083	0.39237	0.41781	0.40929	0.39345	0.37752	0.40511	0.41673	0.27573	0.0.28631
	SVM	0.44857	0.42524	0.41781	0.41781	0.41781	0.41781	0.421	0.41781	0.41781	0.41781	0.42312	0.35423	0.34674
	GBDT	0.48233	0.41623	0.43704	0.40455	0.39661	0.40318	0.43582	0.4045	0.36947	0.39088	0.43772	0.31615	0.31442
Occupation	NB	0.20893	0.18238	0.1824	0.19406	0.18029	0.2089	0.20894	0.18558	0.17181	0.20891	0.20891	0.09866	0.09862
	SVM	0.23118	0.20785	0.20785	0.20785	0.20785	0.20785	0.20785	0.20785	0.20785	0.20785	0.2089	0.16968	0.14634
	GBDT	0.19069	0.19299	0.16394	0.16989	0.15294	0.1685	0.18477	0.15725	0.14613	0.14876	0.16714	0.11933	0.10616

It can be seen from Table 3, the attack accuracy of EIFedMF is significantly lower than PT. We can conclude that at the attack level, EIFedMF is safe. Compared with PT, the inference attack accuracy is 28.7% lower on average.

4.2.2 Data utility

In this paper, we use root mean square error(RMSE) as a measure of utility. We use 10-fold cross-validation, where 90% of the dataset for training, and 10% for testing.

Table 4 shows the the average result.

Table 4: Inference Attack Accuracy

Attribute	PT	RI-(0.1∼0.9)	RF-0.1	RF-0.3	RF-0.5	RF-0.7	RF-0.9	EIFedMF Gender	EIFedMF Age	EIFedMF Occupation	EIFedMF All
RMSE	0.95033	0.95033	0.95298	0.9604	0.97412	1.00881	1.21912	0.97589	0.97523	0.97291	0.97427

All the valid data of PartText(PT) and random increase(RI) obfuscation are uploaded to the server, so we can draw a conclusion that these two parts have no data utility loss. For random filp(RF) obfuscation, the valid data increases with the increase of obfuscation probability p. It is worth noting that although RI and RF with flipping probability smaller than 0.5 have better data utility than EIFedMF, they cannot protect the users against inference attacks very well, especially when using strong attack models, such as GBDT and SVM. EIFedMF had an average utility loss of 2.6% from baseline.

4.2.3 Efficiency

In this paper, the efficiency depends on the number of items uploaded.

Table 5 shows the average number of items that need to be uploaded by each user under FullText(FL), PartText(PT) and EIFedMF modes.

Table 5: The Number of Uploaded Items

	FT	PT	EIFedMF
Number	1682	106	841

In this paper, we measure efficiency by the maximum amount of time it takes a user to upload his/her items to the server. Table 6 shows the maximum time consumed by a user for a round of upload items.

Table 6: The cost of time

	FT	PT	EIFedMF
Cost of time(seconds)	916.34	58.446	449.616

Compared with FullText, the number of items that need to be uploaded by EIFedMF has been reduced by nearly 50%. The efficiency of the whole system has been greatly improved.

5 Conclusion and Future Work

With the promulgation of general data protection regulation (GDPR) and other privacy and data protection laws and regulations, enterprises and other organizations need to obtain user authorization to collect users' rating data and other privacy information. In addition, users' awareness of privacy protection is constantly improving. Users may be reluctant to share their private information, making it difficult to obtain data. The modeling process of recommendation system is very dependent on data, and privacy protection has become the research hotspot of recommendation system. We propose an obfuscation based method based on federated matrix factorization. We learn an obfuscation function to obfuscate users' rating-pair matrix such that the mutual information between the obfuscated result and the private attribute is minimized under a certain distortion

budget. Compared with PT, the inference attack accuracy is 28.7% lower on average, and the safety is 12% higher than that of the original paper. while maintaining the recommending performance and only brings an average utility loss of 2.6%, which is only 0.2% lower than the original paper.

Against more power attack models, when the server knows more data, e.g., users' social network, more powerful attack models could be applied. Thus, protecting users' privacy against these powerful models will be one of our essential research problems.

In the future, we will pay more attention to how to balance security, efficiency and utility, and how to create greater practical value.

References

[1] KOREN Y, BELL R, VOLINSKY C. Matrix factorization techniques for recommender systems[J]. Computer, 2009, 42(8): 30-37.

[2] KIM S, KIM J, KOO D, et al. Efficient privacy-preserving matrix factorization via fully homomorphic encryption[C]//Proceedings of the 11th ACM on Asia conference on computer and communications security. 2016: 617-628.

[3] BERLIOZ A, FRIEDMAN A, KAAFAR M A, et al. Applying differential privacy to matrix factorization[C]//Proceedings of the 9th ACM Conference on Recommender Systems. 2015: 107-114.

[4] CHAI D, WANG L, CHEN K, et al. Secure federated matrix factorization[J]. IEEE Intelligent Systems, 2020, 36(5): 11-20.

[5] CHAI D, WANG L, CHEN K, et al. Efficient Federated Matrix Factorization against Inference Attacks[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2022.

[6] ACAR A, AKSU H, ULUAGAC A S, et al. A survey on homomorphic encryption schemes: Theory and implementation[J]. ACM Computing Surveys (Csur), 2018, 51(4): 1-35.

[7] NAOR M, PINKAS B. Oblivious transfer and polynomial evaluation[C]//Proceedings of the thirty-first annual ACM symposium on Theory of computing. 1999: 245-254.

[8] YANG D, QU B, CUDRÉ-MAUROUX P. Privacy-preserving social media data publishing for personalized ranking-based recommendation[J]. IEEE Transactions on Knowledge and Data Engineering, 2018, 31(3): 507-520.

[9] Du PIN CALMON F, FAWAZ N. Privacy against statistical inference[C]//2012 50th annual Allerton conference on communication, control, and computing (Allerton). 2012: 1401-1408.