FR-FMSS: Federated Recommendation via Fake Marks and Secret Sharing

Zhao Hao Lin¹,², Wei Ke Pan¹,²* and Zhong Ming¹,²*
2017112126@email.szu.edu.cn, {panweike, mingz}@szu.edu.cn

¹National Engineering Laboratory for Big Data System Computing Technology
Shenzhen University, Shenzhen, P.R. China

²College of Computer Science and Software Engineering
Shenzhen University, Shenzhen, P.R. China
Problem Definition

- Privacy-aware cross-user federated recommendation framework
  - Input: Some rating records \( \mathcal{R}_u = \{(u, i, r_{ui})|i \in \mathcal{I}_u\} \), where each user \( u \) has interacted with a set of items \( \mathcal{I}_u \). Note that \( \mathcal{R}_u \) is an unordered set in rating prediction and item ranking or an ordered sequence in sequential recommendation.

- Goal: Predict the rating of user \( u \) to each item in \( \mathcal{I} \setminus \mathcal{I}_u \) or to rank the items in \( \mathcal{I} \setminus \mathcal{I}_u \) for user \( u \) without leaking the rating behavior (i.e., \( \mathcal{I}_u \)) or the rating records (i.e., \( \mathcal{R}_u \)), which is very different from traditional collaborative filtering.
Related Work

Challenges

- Protecting user rating behavior, i.e., the items that the user has interacted with. If a user $u$ uploads $\nabla \Theta_i^s(u)$ to the server, the server will know user $u$ has interacted with item $i$. We call this kind of parameter that may leak user rating behavior as ID-sensitive parameter.

- Protecting user rating values. A recent work [Chai et al., 2019] shows that the server can obtain the rating values as long as it knows the gradients of the item-specific latent feature matrix uploaded by the same user in two continuous iterations.
Federated Collaborative Filtering (FCF)

In FCF [Ammad-ud-din et al., 2019], the authors propose the first federated learning framework for item ranking with implicit feedback.
Federated Collaborative Filtering (FCF)

Specifically, they upload an intermediate gradient $\nabla V_i.(u)$ to the server instead of the user’s original data hoping to protect the user’s privacy,

$$\nabla V_i.(u) = (1 + \alpha y_{ui})(U_u V_i^T - y_{ui})U_u,$$

(1)

where $y_{ui} \in \{0, 1\}$ is an indicator variable for a rating record $(u, i, r_{ui})$ in the training data, and $1 + \alpha y_{ui}$ is a confidence weight with $\alpha > 0$. However, this strategy may not actually protect the privacy of users, because the gradients uploaded by the clients to the server are real. The server may infer the user’s rating values based on the uploaded gradient [Chai et al., 2019].
FedRec: Federated Recommendation with Explicit Feedback

In FedRec [Lin et al., 2020], the authors propose a federated recommendation algorithm for rating prediction with explicit feedback.
FedRec: Federated Recommendation with Explicit Feedback

FedRec [Lin et al., 2020] samples some unrated items and assigns some fake ratings to prevent the server from inferring the users’ rating behavior. We call this technique fake items. They propose two strategies, i.e., user averaging (UA) and hybrid filling (HF) to generate fake gradients. Specifically, they first randomly sample some unrated items $I'_u \subseteq I \setminus I_u$ for each user $u$, assign a fake rating $r'_{ui}$ to each item $i \in I'_u$, and then calculate the fake gradient via $r'_{ui}$.

Note that FedRec do not modify the gradients of the rated items, which may lead to the leakage of the users’ rating values [Chai et al., 2019]. Furthermore, the generated fake ratings will bring noise to the recommendation system and reduce its accuracy.
In this work [Li et al., 2016], the authors use secret sharing to implement a federated version of item-based collaborative filtering.

---

**Step 1 random numbers of**

\[
r_{ui} r_{uj}, r_{ui}^2, r_{uj}^2; \quad i,j \in I_u \text{ or } i,j \in I_{\bar{u}}
\]

**Step 2 random numbers of**

\[
r_{ui} r_{ui}, r_{ui}^2, r_{uj}^2; \quad i,j \in I_{\bar{u}} \text{ or } i,j \in I_u
\]
Dongsheng Li et al.

Secret sharing:

1. Each client divides a value into several random numbers.
2. Each client keeps a copy of the random number, and sends the others to other clients.
3. Each client adds the received random numbers to that it keeps.
4. The server accumulates all the received values.

Note that the result obtained is the same as that obtained without using secret sharing.

In this work, the algorithm uses secret sharing to calculate $\sum_{u \in U} r_{ui} r_{uj}$, $\sum_{u \in U} r_{ui}^2$, and $\sum_{u \in U} r_{uj}^2$, and then the server calculates the cosine similarity as follows,

$$
cos(i, j) = \frac{\sum_{u \in U} r_{ui} r_{uj}}{\sqrt{\sum_{u \in U} r_{ui}^2} \sqrt{\sum_{u \in U} r_{uj}^2}}
$$

(2)

However, when a client receives the data sent by other clients, it will know the rated items of the senders, i.e., leaking user rating behavior.
Our Solution: FR-FMSS

Our FR-FMSS is a cross-user federated recommendation framework. To protect the users’ rating behavior and rating values, it uses two techniques, i.e., fake marks and secret sharing.

**Fake marks.** The first type of fake marks is called fake items, which has been used in FedRec [Lin et al., 2020], SDCF [Jiang et al., 2019], and FedRec++ [Liang et al., 2021]. The second one is called fake ratings, which can be used in federating a more sophisticated method such as MF-MPC [Pan and Ming, 2017]. Note that if an algorithm only uses fake marks to protect user privacy, it may leak the users’ rating values, such as FedRec and FedRec++.

**Secret sharing.** Secret sharing is an important technology of cryptography [Shamir, 1979], which has been used in the work [Li et al., 2016] and sharedMF [Ying, 2020]. However, if an algorithm only uses secret sharing to protect user privacy, it may leak the users’ rating behavior, such as the above two works.
### Notations

**Table: Some notations and explanations.**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{R}$</td>
<td>rating records in training data, $\mathcal{R} = {(u, i, r_{ui})}$</td>
</tr>
<tr>
<td>$\mathcal{R}_u$</td>
<td>rating records w.r.t. user $u$ in $\mathcal{R}$</td>
</tr>
<tr>
<td>$\mathcal{U}$</td>
<td>user set</td>
</tr>
<tr>
<td>$\mathcal{I}$</td>
<td>item set</td>
</tr>
<tr>
<td>$\mathcal{M}$</td>
<td>mark set of the shared parameters</td>
</tr>
<tr>
<td>$\mathcal{M}_u$</td>
<td>mark set that the user $u$ wants to update, $\mathcal{M}_u \subset \mathcal{M}$</td>
</tr>
<tr>
<td>$\mathcal{M}_u'$</td>
<td>fake mark set that the user $u$ samples, $\mathcal{M}_u' \subset \mathcal{M} \setminus \mathcal{M}_u$ and $</td>
</tr>
<tr>
<td>$\tilde{\mathcal{M}}_u$</td>
<td>mark set that the user $u$ receives from other clients</td>
</tr>
<tr>
<td>$\mathcal{N}_u$</td>
<td>set of users who receive data sent by the user $u$ in secret sharing</td>
</tr>
<tr>
<td>$\Theta^p_u$</td>
<td>the private parameters w.r.t. user $u$</td>
</tr>
<tr>
<td>$\Theta^s_m$</td>
<td>the shared parameters w.r.t. mark $m$</td>
</tr>
<tr>
<td>$\nabla \Theta^s_m(u)$</td>
<td>the gradient of $\Theta^s_m$ calculated by the user $u$, which will be sent to the server</td>
</tr>
<tr>
<td>$\tilde{\nabla \Theta}^s_m(\tilde{u}, u)$</td>
<td>the random number of $\nabla \Theta^s_m(u)$, which will be sent to the user $\tilde{u}$</td>
</tr>
<tr>
<td>$\nabla \Theta^p(u)$</td>
<td>the sum of the gradients of $\Theta^p_u$ calculated by the user $u$ using $\mathcal{R}_u$</td>
</tr>
<tr>
<td>$Y_m$</td>
<td>the number of users who want to update $\Theta^s_m$</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>the learning rate</td>
</tr>
<tr>
<td>$\rho$</td>
<td>the sampling parameter</td>
</tr>
</tbody>
</table>
Interactions between the Server and each Client

- Step 1: $\Theta^s_m(u)$ for $m \in M_u \cup M'_u \cup \overline{M}_u$
- Step 2: $\nabla \Theta^s_m(u, \overline{u})$, $m \in M_{\overline{u}} \cup M'_{\overline{u}}$
- Step 3: $\nabla \Theta^s_m(\overline{u})$, $m \in M_{\overline{u}} \cup M'_{\overline{u}}$
Advantages of FR-FMSS

We extend fake items to fake marks and combine it with secret sharing to protect user privacy, which really protects users’ rating behavior and rating values.

- With the marks set sent to other clients, i.e., $\mathcal{M}_u \cup \mathcal{M}'_u$, and the marks set sent to the server, i.e., $\mathcal{M}_u \cup \mathcal{M}'_u \cup \bar{\mathcal{M}}_u$, it will be more difficult for the clients and the server to identify what marks the corresponding user $u$ has interacted with, which thus protects the users’ privacy in terms of rating behavior.

- Because secret sharing modifies the actually calculated gradients, it will be more difficult for the clients and the server to infer the rating values of the corresponding user $u$ from their received data, which thus protects the users’ privacy in terms of rating values.

- Because secret sharing generates fake gradients without introducing noise, our FR-FMSS is a lossless method.

- It is generic, which means that it can be applied to many un-federated recommendation algorithms.
FR-FMSS in a Client \( u \)

1. The client \( u \) calculates \( \nabla \Theta^s_m \) and \( \nabla \Theta^p_u \) with the backbone recommendation algorithm.

2. The client \( u \) randomly samples some marks \( M'_u \) from \( M \setminus M_u \) with 
\(| M'_u | = \rho | M_u | \), and assigns \( \nabla \Theta^s_m \leftarrow 0, m \in M'_u \).

3. The client \( u \) randomly samples \( N_u \) from \( U \setminus \{ u \} \)

4. The client \( u \) generates \( | N_u | + 1 \) random numbers that meet 
\[ \nabla \Theta^s_m(u) = \sum_{\tilde{u} \in N_u \cup \{ u \}} \nabla \Theta^s_m(\tilde{u}, u), \] and then keeps \( \nabla \Theta^s_m(u, u) \) and sends \( \nabla \Theta^s_m(\tilde{u}, u), \tilde{u} \in N_u \) to the client \( \tilde{u} \)

5. After all the clients have sent data, the client \( u \) adds the received random numbers and that it keeps as follows, 
\[ \nabla \Theta^s_m(u) \leftarrow \nabla \Theta^s_m(u, u) + \sum_{\tilde{u}} \nabla \Theta^s_m(u, \tilde{u}) \]

6. The client \( u \) uploads \( \nabla \Theta^s_m(u), m \in M_u \cup M'_u \cup \tilde{M}_u \) to the server, and then updates its private parameters.
FR-FMSS in the Server

1. The server sends the shared parameters and lets the clients start training.
2. Once the server has received all the gradients sent by the clients, it calculates \( \nabla \Theta_m^s = \sum_{u \in \mathcal{U}} \nabla \Theta_m^s(u) \).
3. The server updates \( \Theta_m^s \) using \( \nabla \Theta_m^s \).

We depict the learning process of the server in Algorithm 1 and that of each client in Algorithm 2.
Algorithm 1 The algorithm of FR-FMSS in the server.

1: Initialize the shared parameters $\Theta^s_m, m \in \mathcal{M}$
2: for $t = 1, 2, \ldots, T$ do
3:   for each client $u$ in parallel do
4:     ClientTrain($\Theta^s_m, m \in \mathcal{M}; t; \gamma$), i.e., Algorithm 2
5:   end for
6: for $m \in \mathcal{M}$ do
7:   Receive $Y_m(u), u \in \mathcal{U}$
8:   $Y_m \leftarrow \sum_{u \in \mathcal{U}} Y_m(u)$
9:   Receive $\nabla \Theta^s_m(u), u \in \mathcal{U}$
10:  $\nabla \Theta^s_m = \sum_{u \in \mathcal{U}} \nabla \Theta^s_m(u)$
11:  $\Theta^s_m \leftarrow \Theta^s_m - \gamma \frac{\nabla \Theta^s_m}{Y_m}$
12: end for
13: Decrease the learning rate $\gamma \leftarrow 0.9 \gamma$
14: end for
Algorithm 2 ClientTrain($\Theta^s_m, m \in M; t; \gamma$)

1: $\nabla \Theta^p(u) \leftarrow 0$
2: for $m \in M_u$ do
3:     $\nabla \Theta^s_m(u) \leftarrow 0$
4:     $Y_m(u) \leftarrow 0$
5: end for
6: for $m \in M_u$ do
7:     Calculate $\nabla \Theta^s_m$ and $\nabla \Theta^p_u$ with the recommendation algorithm
8:     $\nabla \Theta^s_m(u) \leftarrow \nabla \Theta^s_m(u) + \nabla \Theta^s_m$
9:     $Y_m(u) \leftarrow 1$
10:    $\nabla \Theta^p(u) \leftarrow \nabla \Theta^p(u) + \nabla \Theta^p_u$
11: end for
12: Sample fake marks $M'_u$ from $M \setminus M_u$ with $|M'_u| = \rho |M_u|$
13: for $m \in M'_u$ do
14:     $\nabla \Theta^s_m(u) \leftarrow 0$
15:     $Y_m(u) \leftarrow 0$
16: end for
17: Sample $N_u$ from $U \setminus \{u\}$
18: for $m \in M'_u \cup M_u$ do
Algorithm of FR-FMSS in the Client II

19: Generate $|\mathcal{N}_u| + 1$ random numbers that meet $\nabla \Theta^s_m(u) = \sum_{\tilde{u} \in \mathcal{N}_u \cup \{u\}} \nabla \Theta^s_m(\tilde{u}, u)$
20: Keep $\nabla \Theta^s_m(u, u)$ and send $\nabla \Theta^s_m(\tilde{u}, u), \tilde{u} \in \mathcal{N}_u$ to the client $\tilde{u}$
21: Generate $|\mathcal{N}_u| + 1$ random numbers that meet $Y_m(u) = \sum_{\tilde{u} \in \mathcal{N}_u \cup \{u\}} Y_m(\tilde{u}, u)$
22: Keep $Y_m(u, u)$ and send $Y_m(\tilde{u}, u), \tilde{u} \in \mathcal{N}_u$ to the client $\tilde{u}$
23: end for
24: Synchronize()
25: $\tilde{\mathcal{M}}_u \leftarrow \emptyset$
26: for $m \in \mathcal{M}$ do
27: if receive a set of $\nabla \Theta^s_m(u, \tilde{u}), u \in \mathcal{N}_u$ then
28: \[ \tilde{\mathcal{M}}_u \leftarrow \tilde{\mathcal{M}}_u \cup \{m\} \]
29: \[ \nabla \Theta^s_m(u) \leftarrow \nabla \Theta^s_m(u, u) + \sum_{\tilde{u}} \nabla \Theta^s_m(u, \tilde{u}) \]
30: \[ Y_m(u) \leftarrow Y_m(u, u) + \sum_{\tilde{u}} Y_m(u, \tilde{u}) \]
31: end if
32: end for
33: Upload $\nabla \Theta^s_m(u), Y_m(u), m \in \mathcal{M}_u \cup \mathcal{M}'_u \cup \tilde{\mathcal{M}}_u$ to the server
34: $\Theta^p_u \leftarrow \Theta^p_u - \gamma \frac{\nabla \Theta^p(u)}{|\tilde{\mathcal{M}}_u|}$
FR-FMSS Specialization

1. We judge whether each parameter of the un-federated algorithm is a shared parameter or a private parameter.
2. We find out the ID-sensitive parameters of the un-federated algorithm.
3. We design the intermediate values that the clients upload to the server according to the parameter update rules of the un-federated algorithm.
4. We specialize Algorithm 1 and Algorithm 2 according to the shared parameters, private parameters, ID-sensitive parameters, and intermediate values.
Specialization of MF-MPC for Rating Prediction

1. We set $U_u$ and $b_u$ in MF-MPC as private parameters $\Theta^p_u$ in our FR-FMSS, and set $V_i$, $b_i$, $M^r_i$, and $\mu$ as shared parameters $\Theta^s_m$ in our FR-FMSS.

2. We find that $V_i$, $b_i$, and $M^r_i$ are ID-sensitive parameters.

3. We use gradient descent to update the parameters of MF-MPC, so we set $\nabla V_i(u)$, $\nabla b_i(u)$, $\nabla M^r_i(u)$, and $\nabla \mu(u)$ in MF-MPC as $\nabla \Theta^s_m(u)$ in our FR-FMSS.

4. We specialize Algorithm 1 and Algorithm 2 based on the above settings.

Note that the mark of $V_i$ and $b_i$ is item ID $i$, and the mark of $M^r_i$ is a combination of item ID $i$ and rating $r$, so we must use both fake items and fake ratings to federate MF-MPC.
Specialization of eALS for Item Ranking with Implicit Feedback

1. We set $U_u$ in eALS as $\Theta^p_u$ in our FR-FMSS, and set $V_i$, $S^V$, and $S^U$ as $\Theta^s_m$ in our FR-FMSS.

2. We find that only $V_i$ is ID-sensitive parameter.

3. Because eALS does not use gradient descent to update its parameters, we set $A_{if}(u) = [\omega_{ui} r_{ui} - (\omega_{ui} - c_i) \hat{r}_{ui}] U_{uf}$, $B_{if}(u) = (\omega_{ui} - c_i) U^2_{uf}$, and $S^U(u)$ in eALS as $\nabla \Theta^s_m(u)$ in our FR-FMSS. Then we rewrite the update rule of $V_{if}$ as

$$V_{if} = \frac{\sum_{u \in R_i} A_{if}(u) - c_i \sum_{k \neq f} V_{ik} S^U_{kf}}{\sum_{u \in R_i} B_{if}(u) + c_i S^U_{if} + \lambda}$$

so that the server can use the received $A_{if}(u)$, $B_{if}(u)$, and $S^U(u)$ to update $V_{if}$.

4. We specialize Algorithm 1 and Algorithm 2 based on the above settings.
Specialization of Fossil for Sequential Recommendation

1. We set $\eta_k^u$ in Fossil as $\Theta_u^p$ in our FR-FMSS, and set $V_i$, $b_i$, $W_i$, and $\eta_k$ in Fossil as $\Theta_m^s$ in our FR-FMSS.

2. We find that $V_i$, $b_i$, and $W_i$ are ID-sensitive parameters.

3. We use gradient descent to update the parameters of Fossil, so we set $\nabla V_i(u)$, $\nabla b_i(u)$, $\nabla W_i(u)$, and $\nabla \eta_k(u)$ in Fossil as $\nabla \Theta_m^s(u)$ in our FR-FMSS.

4. We specialize Algorithm 1 and Algorithm 2 based on the above settings.

Because we use batch gradient decent to update Fossil, all the items in a user’s rated item sequence will be used to calculate the gradients in one iteration, which means that the order information of the rated item sequence is transparent to the server. Therefore, the steps for applying FR-FMSS to Fossil are very similar to those of MF-MPC.
Discussions

Because our FR-FMSS uses fake marks and secret sharing, the server and all the clients can infer no private information from their received data. Moreover, a client will send random numbers to several randomly sampled clients. Therefore, unless the server colludes with all the clients, the privacy of each client is hard to be inferred. This means that our FR-FMSS is more secure than the algorithms that require third-party servers or denoising clients such as FedGNN [Wu et al., 2021] and FedRec++ [Liang et al., 2021].

The increased communication cost is
\[ \sum_{u \in U} |N_u||\mathcal{M}_u \cup \mathcal{M}'_u||\nabla \Theta^s_m(\tilde{u}, u)| + \sum_{u \in U} |\mathcal{M}'_u \cup \tilde{\mathcal{M}}_u \setminus \mathcal{M}_u||\nabla \Theta^s_m(u)|. \]

Because a client only needs to send \( \nabla \Theta^s_m(\tilde{u}, u) \) to a small number of clients and \( \rho = \frac{|\mathcal{M}'_u|}{|\mathcal{M}_u|} \) is also very small, the increased communication cost is not high.
Conclusions

We propose a generic cross-user federated recommendation framework called FR-FMSS, which mainly makes use of two techniques, i.e., fake marks and secret sharing.

Our FR-FMSS can be applied to a variety of un-federated recommendation algorithms. We take MF-MPC, eALS and Fossil as examples to show how to apply our FR-FMSS to a specific algorithm.
We are interested in extending our FR-FMSS to cross-organization federated recommendation. We are also interested in studying efficient interactions methods among the server and clients, in terms of both computation and communication complexity.
Thank you!

We thank the anonymous reviewers for their expert and constructive comments and suggestions. We thank the support of National Natural Science Foundation of China Nos. 61836005 and 61872249, and Shenzhen Basic Research Fund No. JCYJ20200813091134001.
Federated collaborative filtering for privacy-preserving personalized recommendation system.
CoRR, abs/1901.09888.

Secure federated matrix factorization.

Towards a more reliable privacy-preserving recommender system.

An algorithm for efficient privacy-preserving item-based collaborative filtering.

Fedrec++: Lossless federated recommendation with explicit feedback.

Fedrec: Federated recommendation with explicit feedback.
IEEE Intelligent Systems, pages 1–1.

Collaborative recommendation with multiclass preference context.

How to share a secret.
Fedgnn: Federated graph neural network for privacy-preserving recommendation.
CoRR, abs/2102.04925.

Ying, S. (2020).
Shared MF: A privacy-preserving recommendation system.