FedCORE: Federated Learning for Cross-Organization Recommendation Ecosystem

Zhitao Li[1], Xueyang Wu[2], Weike Pan[1*], Youlong Ding[1], Zeheng Wu[3], Shengqi Tan[3], Qian Xu[2], Qiang Yang[2], Zhong Ming[1]

[1]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China

[2]Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China

[3]WeBank, Shenzhen, China

Problem Definition

Two-partner Cross-Organization Recommendation (COR). We have a set of rating records $\mathcal{R}_{()} = \{(u, i, r_{ui}) | u \in \mathcal{U}_{()}, i \in \mathcal{I}_{()}\}$, where $\mathcal{U}_{()}$ and $\mathcal{I}_{()}$ are the user set and item set of partners *a* and *b*, and r_{ui} is the rating that user *u* assigns to item *i*. For two partners *a* and *b* with rating records $\mathcal{R}_a$ and $\mathcal{R}_b$, the goal is to obtain two recommendation models $f_a(u, i)$ and $f_b(u, i)$ by solving the following optimization problems,

$$\min_{f_a()} \sum_{i \in \mathcal{I}_a, u \in \mathcal{U}_{\backslash a}} (f_a(u, i) - r_{u,i})^2, \tag{1}$$

$$\min_{f_b()} \sum_{i \in \mathcal{I}_b, u \in \mathcal{U}_{\backslash b}} (f_b(u, i) - r_{u,i})^2, \tag{2}$$

where $\mathcal{U}_{\backslash a} = \mathcal{U}_b - \mathcal{U}_a$ denotes the users in partner *b* but unseen in partner *a*, and $\mathcal{U}_{\backslash b}$ is similar.

Notations (1/5)

Table: Some notations and explanations.

n	the number of all users
m	the number of all items
$d \in \mathbb{Z}$	the number of latent dimensions
$U_u \in \mathbb{R}^{1\times d}$	user latent feature vector
$U_u{'}^{*} \in \mathbb{R}^{1\times d}$	recovered user latent feature vector
$\mathbf{U} \in \mathbb{R}^{n\times d}$	user embeddings
$V_i \in \mathbb{R}^{1\times d}$	item latent feature vector
$V_i{'}^{*} \in \mathbb{R}^{1\times d}$	recovered item latent feature vector
$\bar{V}_i \in \mathbb{R}^{1\times d}$	item latent feature vector attacked by partner b
$\mathbf{V} \in \mathbb{R}^{m\times d}$	item embeddings
$\mathbf{W}$	middle-layer parameters
$\nabla\mathbf{W}$	gradient of the middle-layer parameters

Notations (2/5)

Table: Some notations and explanations (cont.).

Notation	Explanation
$r_{ui} \in \{1, 2, \ldots, 5\}$	rating of user *u* to item *i*
$\hat{r}_{ui} \in \{1, 2, \ldots, 5\}$	predicted rating of user *u* to item *i*
$r^{*\prime}_{ui} \in \{1, 2, \ldots, 5\}$	recovered rating of user *u* to item *i*
$y_{ui} \in \{0, 1\}$	indicator variable
$\mathcal{U}_a$	the set of users w.r.t. partner *a*
$\mathcal{U}_{\backslash a} = \mathcal{U}_b \backslash \mathcal{U}_a$	users in partner *b* but unseen in partner *a*
$\mathcal{U}_b$	the set of users w.r.t. partner *b*
$\mathcal{U}_{\backslash b} = \mathcal{U}_a \backslash \mathcal{U}_b$	users in partner *a* but unseen in partner *b*
$\mathcal{U}_{com} = \mathcal{U}_a \cap \mathcal{U}_b$	the set of common users

Notations (3/5)

Table: Some notations and explanations (cont.).

Notation	Explanation
$\mathcal{U} = \mathcal{U}_a \cup \mathcal{U}_b$	the whole set of users
$\mathcal{I}_a$	the set of items w.r.t. partner *a*
$\mathcal{I}_b$	the set of items w.r.t. partner *b*
$\mathcal{I}_{com} = \mathcal{I}_a \cap \mathcal{I}_b$	the set of common items
$\mathcal{I} = \mathcal{I}_a \cup \mathcal{I}_b$	the whole set of items
$\mathcal{R}_a$	rating records in training data w.r.t. partner *a*
$\mathcal{R}_b$	rating records in training data w.r.t. partner *b*
$\mathcal{R}_a^k$	a portion of $\mathcal{R}_a$ in a certain iteration w.r.t. partner *a*
$\mathcal{I}_a^k$	a portion of $\mathcal{I}_a$ in a certain iteration w.r.t. partner *a*

Notations (4/5)

Table: Some notations and explanations (cont.).

Notation	Explanation
$\mathcal{U}_a^k$	a portion of $\mathcal{U}_a$ in a certain iteration w.r.t. partner a
$\nabla \mathcal{V}_i^k$	the set of ∇V_i in a certain iteration trained from $\mathcal{R}_a^k$ w.r.t. partner a
$\nabla \mathcal{W}^k$	the set of $\nabla \mathbf{W}$ in a certain iteration w.r.t. partner a, including the locally trained part from $\mathcal{R}_a^k$ and the received part from partner b
$\nabla \mathcal{U}_u^k$	the set of ∇U_u in a certain iteration w.r.t. partner a, including the locally trained part from $\mathcal{R}_a^k$ and the received part from partner b
$\nabla \mathcal{U}_a^u$	the set of ∇U_u w.r.t. partner a and user u
$\nabla \mathcal{U}_{com}^u$	the set of ∇U_u w.r.t. common user u

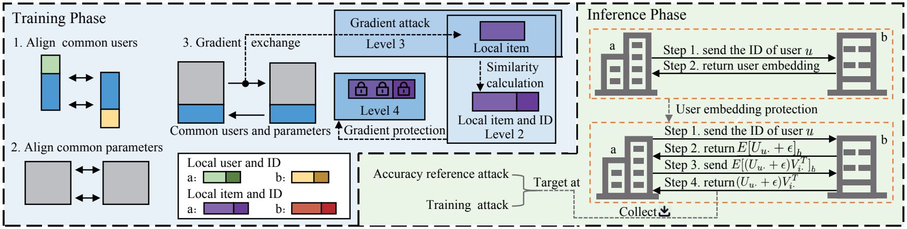
Notations (5/5)

Table: Some notations and explanations (cont.).

Notation	Explanation
$\nabla\tilde{\mathcal{U}}_{com}^{u}$	the set of privatized ∇U_u w.r.t. a common user u
γ	learning rate
α	weight of penalty term
T	total iteration epoch
K	iteration number in each epoch
ϵ	the parameter in differential privacy
Δh	the ℓ_1 sensitivity of a function h
$\mathcal{M}(\mathrm{X})$	an algorithm that takes input X
$\mathrm{Range}(\mathcal{M})$	the value range of algorithm $\mathcal{M}$
$\mathrm{Predict}(\cdot)$	the model prediction function
$E[*]_b$	encryption w.r.t. partner b
$D[*]_b$	decryption w.r.t. partner b

- An absence of study in a general data setting. The prevailing setting in previous works acquiescently only aligns users or items, disregarding the real-world scenarios where both users and items usually overlap across different organizations.
- An absence of study with general recommendation models. Most previous works predominantly hinge on the utilization of a specific model, while lacking study on practices under a generic recommendation framework.
- Unsustainable cross-organization cooperation. A noteworthy shortcoming inherent in almost all previous works lies in the practice of training models only once, thereby terminating the collaboration process prematurely, which undermines the prospects of fostering sustainable commercialization.

Overall of Our Solution



Training Phase
1. Align common users
2. Align common parameters
3. Gradient exchange
Common users and parameters
Gradient attack
Level 3
Local item
Similarity calculation
Local item and ID
Level 2
Level 4
Gradient protection
Local user and ID
a:
b:
Local item and ID
a:
b:
Accuracy reference attack
Training attack
Target at
Collect
Inference Phase
a
b
Step 1. send the ID of user u
Step 2. return user embedding
User embedding protection
Step 1. send the ID of user u
Step 2. return $E[U_{u\cdot} + \epsilon]_b$
Step 3. send $E[(U_{u\cdot} + \epsilon)V_{i\cdot}^T]_b$
Step 4. return $(U_{u\cdot} + \epsilon)V_{i\cdot}^T$

Figure: The overall framework of our FedCORE, including three key components, namely: collaborative training and inference, studying privacy leakage concerns, and our privacy protection approach.

Overall of Our Solution

- We are the first that study the ecosystem problem of cross-organization federated recommendation, for which we propose a novel solution called FedCORE.
- We analyze the privacy leakage problem in federated recommendation and design several attack methods to illustrate the potential risks.
- We implement a privacy-preserving method for collaborative training and design a cryptography-based protocol to protect user privacy in the inference phase.
- We conduct extensive experiments on three real-world datasets and two seminal recommendation models to study the effectiveness of cooperation and privacy protection in our FedCORE.

Differential Privacy

- **Differential Privacy.** For all $\mathcal{S} \subset \mathsf{Range}(\mathcal{M})$, and for all adjacent datasets X, X' that differ by one instance, a randomized algorithm $\mathcal{M}(\mathrm{X})$ is considered as (ϵ, δ)-differentially private if the following equation holds,

$$\Pr(\mathcal{M}(\mathrm{X}) \in \mathcal{S}) \leq e^{\epsilon} \Pr\left(\mathcal{M}\left(\mathrm{X}'\right) \in \mathcal{S}\right) + \delta, \tag{3}$$

  where $\mathcal{S}$ denotes a subset of the output of the algorithm $\mathcal{M}$, $\mathsf{Range}(\mathcal{M})$ denotes the set of output of the algorithm $\mathcal{M}$, ϵ denotes the privacy budget parameter in differential privacy and δ denotes a small positive number typically close to zero.

Differential Privacy

- The Laplace mechanism [Dwork et al., 2014] is a popular mechanism used to guarantee ϵ-differential privacy. the Laplace mechanism can be formulated as a function $\mathsf{Privatize}(\mathrm{X}, \epsilon, \Delta h)$,

$$\mathsf{Privatize}(\mathrm{X}, \epsilon, \Delta h) = \mathrm{X} + \epsilon, \quad \epsilon \sim \mathsf{Laplace}(\frac{\Delta h}{\epsilon}). \tag{4}$$

Note that Δh is defined as the ℓ_1 sensitivity of a function h,

$$\Delta h = \max_{\mathrm{X},\mathrm{X}'} \left\| h(\mathrm{X}) - h\left(\mathrm{X}'\right) \right\|_1 . \tag{5}$$

The operation $\mathsf{privatize}(\mathrm{X}, \epsilon, \Delta h)$ provides ϵ-DP protection to the output X with the sensitivity Δh. The privacy budget ϵ is inversely proportional to the noise scale, which means (i) when $\epsilon = 0$, the output is totally overwhelmed by the noise resulting in no utility and (ii) when for $\epsilon = \infty$, there is no protection.

DP for our FedCORE.

- Similar to the operations in [McMahan et al., 2017] [Abadi et al., 2016], we are able to bound the sensitivity by clipping the gradient with a certain scale.
- More specifically, we clip the gradients by ℓ_1 norm of scale 0.05, where in most cases the difference between any two gradients is $\Delta h = 0.1$, i.e., in our FedCORE, the privatization process is implemented as follows,

$$\mathrm{PrivateCOR}(\mathrm{X}, \epsilon) = \mathrm{Privatize}(\mathrm{Clip}(\mathrm{X}, 0.05), \epsilon, 0.1), \tag{6}$$

  where $\mathrm{Clip}(a, b)$ is a function that clips the ℓ_1 norm of an input a by a scale b.

Privacy Definition

We define four levels of privacy protection in the cross-organization federated recommendation as follows.

- **Level 1**: The data partner leaks some complete raw data, features, and IDs corresponding to the features.
- **Level 2**: The data partner leaks some complete features and IDs corresponding to the features.
- **Level 3**: The data partner leaks some complete features but does not know the corresponding IDs. Alternatively, the data partner leaks some features with noise at most, but knows the corresponding IDs.
- **Level 4**: The data partner leaks at most some features with noise, and does not know the corresponding IDs.

Privacy Definition

Table: Four levels of privacy protection.

	Raw data	Feature	ID
Level 1	🔓	🔓	🔓
Level 2	🔒	🔓	🔓
Level 3	🔒	🔒	🔓
	🔒	🔓	🔒
Level 4	🔒	🔒	🔒

- 🔒: Protected, 🔓: Unprotected.

Privacy Definition

- In the case of transmitting plaintext gradients, the protection level of cross-user federated recommendation is level 2, while the protection level of cross-organization federated recommendation is level 3.
- With certain protection methods, such as differential privacy, secret sharing, and homomorphic encryption, the protection level of cross-user federated recommendation can reach level 3 or level 4, and the protection level of cross-organization federated recommendation can reach level 4.

Collaborative Training Framework

In general, the optimization problem of our FedCORE in the training phase is as follows,

$$\min_{\Theta} \sum_{u \in \mathcal{U}_a} \sum_{i \in \mathcal{I}_a} y_{ui} f_{ui} + \sum_{u \in \mathcal{U}_b} \sum_{i \in \mathcal{I}_b} y_{ui} f_{ui}, \tag{7}$$

where $y_{ui} = 1$ if the rating record $(u, i, r_{ui}) \in \mathcal{R}_a$ or $\mathcal{R}_b$ and $y_{ui} = 0$ otherwise, $\mathcal{U}_a \cap \mathcal{U}_b = \mathcal{U}_{com}$, $\mathcal{I}_a \cap \mathcal{I}_b = \mathcal{I}_{com}$. f_{ui} depends on a specific recommendation algorithm, e.g., $f_{ui} = \frac{1}{2}(r_{ui} - \hat{r}_{ui})^2 + \frac{\alpha}{2}\|U_u\|^2 + \frac{\alpha}{2}\|V_i\|^2$ in PMF, where $\hat{r}_{ui}$ denotes the predicted rating of user u to item i, and α is the weight of the penalty term.

Algorithm of DFedRec(b) I

Algorithm 1 Collaborative Training of FedCORE.

```
 1: Step 1. Common user ID alignment. Each partner aligns the user set $\mathcal{U}$ and does not align
    the item set $\mathcal{I}$.
 2: Step 2. Common parameters initialization alignment. Each partner keeps the initialization
    values of the user embeddings and the middle-layer parameters $\mathbf{W}$ consistent.
 3: Step 3. Co-training via gradient exchange w.r.t. partner $a$.
 4: for $t = 1, 2, \ldots, T$ do
 5:     for $k = 1, 2, \ldots, K$ do
 6:         Sample a portion of rating records $\mathcal{R}_a^k$
 7:         for $r_{ui} \in \mathcal{R}_a^k$ (parallely) do
 8:             Compute the gradients $\nabla U_u$, $\nabla V_i$ and $\nabla \mathbf{W}$ based on a specific model
 9:         end for
10:         Send $\nabla \mathbf{W}$ to partner $b$
11:         for $u \in \mathcal{U}_a^k \cap \mathcal{U}_{com}$ (parallely) do
12:             Compute privatized $\nabla \tilde{\mathcal{U}}_{com}^u$ via Eq.(12), Eq.(13) or Eq.(14)
13:             send $\nabla \tilde{\mathcal{U}}_{com}^u$ to partner $b$
14:         end for
15:         Synchronize() /*Wait for all partners to complete the exchange of gradients of the
            users' latent feature vectors.*/
```

Algorithm of DFedRec(b) II

```
16:         $\mathbf{W} = \mathbf{W} - \lambda \frac{\sum_{\nabla \mathbf{W}' \in \nabla \mathcal{W}^k} \nabla \mathbf{W}'}{|\nabla \mathcal{W}^k|}$
17:         **for** $i \in \mathcal{I}_a^k$ **do**
18:             $V_i = V_i - \lambda \frac{\sum_{\nabla V_i' \in \nabla \mathcal{V}_i^k} \nabla V_i'}{|\nabla \mathcal{V}_i^k|}$
19:         **end for**
20:         **for** $u \in \mathcal{U}_a^k$ **do**
21:             $U_u = U_u - \lambda \frac{\sum_{\nabla U_u' \in \nabla \mathcal{U}_u^k} \nabla U_u'}{|\nabla \mathcal{U}_u^k|}$
22:         **end for**
23:     **end for**
24: **end for**
```

Data Attack Methods

- In this section, we introduce how to attack the common user gradients to obtain the complete item embedding, thereby reducing the protection level of cross-organization collaborative training with plaintext gradients to level 3.
- Subsequently, we propose that the protection level can be further reduced from level 3 to level 2 to some extent through similarity calculation. We assume that in a two-partner COR scenario, partner *b* is honest but curious, and tries to attack partner *a* by recovering some useful information from the shared gradients $\nabla U_u, u \in \mathcal{U}_{com}$.

Data Attack Methods

Firstly, we introduce how to attack the common user gradients to obtain the complete item embedding. The general form of the attack method on the common user gradients for different recommendation methods is as follows,

$$V_i'^{*}, r_{ui}^{*}{}' = \text{Attack}(\nabla \mathbf{W}, \mathbf{W}, \nabla U_u, U_u), \tag{8}$$

where $u \in \mathcal{U}_{com}$, $i \in \mathcal{I}_a$, $V_i'^{*}$ denotes the recovered item latent feature vector and $r_{ui}^{*}{}'$ denotes the recovered rating of user u to item i from the attack method.

Data Attack Methods

In PMF, the function Attack(·) can solve the following linear equation system,

$$\nabla U_u = -(r_{ui} - U_u V_i^T) V_i + \alpha U_u. \tag{9}$$

When partner *a* computes a gradient $\nabla U_u, u \in \mathcal{U}_{com}$ via Eq.(9) and sends it to partner *b* for synchronous update, partner *b* can completely restore the value of $V_i, i \in \mathcal{I}_a$. This can be done by solving the following linear equation system,

$$\begin{cases} \nabla U_{u1} = & -(r_{ui} - U_u {V_i'^*}^T) {V_{i1}'}^* + \alpha U_{u1} \\ \nabla U_{u2} = & -(r_{ui} - U_u {V_i'^*}^T) {V_{i2}'}^* + \alpha U_{u2} \\ \vdots & \\ \nabla U_{ud} = & -(r_{ui} - U_u {V_i'^*}^T) {V_{id}'}^* + \alpha U_{ud} \end{cases} \tag{10}$$

where each equation corresponds to one dimension.

Data Attack Methods

In NeuCF, the function Attack(·) can be based on the idea of DLG [Zhu et al., 2019]. However, unlike the original DLG, the data to be restored in DLG [Zhu et al., 2019] is a matrix-level image, while in our setup it is vector-level item embeddings. This makes the attack more likely to succeed. The optimization problem is shown as follows,

$$
\begin{aligned}
&{V_i'}^*, {r_{ui}^*}' \\
&= \underset{V_i', r_{ui}'}{\arg\min} \left\|(\nabla \mathbf{W}', \nabla U_u') - (\nabla \mathbf{W}, \nabla U_u)\right\|^2 \\
&= \underset{V_i', r_{ui}'}{\arg\min} \|\|\|(\frac{\partial \ell\left(F\left(V_i', \mathbf{W}\right), {r_{ui}}'\right)}{\partial \mathbf{W}}, \frac{\partial \ell\left(F\left(V_i', \mathbf{W}\right), {r_{ui}}'\right)}{\partial U_u}) \\
&\quad - (\nabla \mathbf{W}, \nabla U_u)\|\|\|^2
\end{aligned} \tag{11}
$$

where ${V_i'}^* \in \mathbb{R}^{1\times d}$ is the recovered item latent feature vector and ${r_{ui}'}^*$ is the recovered rating of user u to item i.

Data Attack Methods

Secondly, we introduce how to reduce the protection level from level 3 to level 2 to some extent by calculating the similarity.

- **Step 1.** Partner b receives the gradient ∇U_u from partner a and attempts to find $\bar{V}_i$ using the attack method described in Eqs.(10) and (11).
- **Step 2.** Partner b calculates the cosine similarity between $\bar{V}_i$ and the latent feature vectors of all items belonging to partner b in order to determine the likely ground-truth ID of $\bar{V}_i$. The item with the highest cosine similarity is considered the most probable ground-truth ID.
- **Step 3.** Partner b calculates r_{ui} for all users $u \in \mathcal{U}_{com}$ and item $i = i'$ using $\bar{V}_i$ and U_u, $u \in \mathcal{U}_{com}$.

Privacy Protection

To address the issue mentioned above, we propose three strategies to make it more difficult for an adversary to gain useful information from the shared gradients.

- **Differential Privacy (DP).** We apply DP (discussed in Section 4) to privatize the gradients $\nabla\mathcal{U}_{com}$ before transferring them to the other partner. In Eq.(12), we utilize the privatization function $\mathrm{Privatize}_{tr}$ defined in Eq.(6) to privatize the exposed gradients as $\nabla\tilde{U}_u$. As a result, the similarity between $\bar{V}_i, i \in \mathcal{I}_a$ attacked by partner b and $V_i, i \in \mathcal{I}_b$ is reduced,

$$\nabla\tilde{\mathcal{U}}^{u}_{com} = \mathrm{PrivateCOR}(\nabla\mathcal{U}_{com}, \epsilon), \quad u \in \mathcal{U}_{com}. \tag{12}$$

Privacy Protection

In the data attack method for PMF, we must have at least as many equations as unknown variables in order to solve a system of linear equations like Eq.(10). Therefore, we can make it more difficult to solve the equation system by increasing the number of unknown variables.

- **Average of Gradients (AG).** we can have a simple but effective defense method by the gradients of the users' latent feature vectors locally,

$$\nabla \tilde{\mathcal{U}}_{com}^{u} = \frac{\sum_{\nabla U'_u \in \nabla \mathcal{U}_a^u} \nabla U'_u}{|\nabla \mathcal{U}_a^u|}, u \in \mathcal{U}_{com}. \tag{13}$$

It is worth noting that averaging the gradient makes the attack method, i.e., DLG [Zhu et al., 2019], more difficult to succeed, which also has a protective effect on the deep learning-based recommendation methods.

Privacy Protection

The aforementioned two defense strategies can be integrated to provide stronger privacy protection. We propose to combine two methods as follows,

- **Mixture of DP and AG (DP+AG).** We propose to combine two methods as follows,

$$\nabla\tilde{\mathcal{U}}^u_{com} = \text{PrivateCOR}\left(\frac{\sum_{\nabla U'_u \in \nabla\mathcal{U}^u_a} \nabla U'_u}{|\nabla\mathcal{U}^u_a|}, \epsilon\right), \quad u \in \mathcal{U}_{com}. \tag{14}$$

In order to choose the strategy of AG, a mini-batch update is used during gradient descent. And DP will also affect the recommendation performance.

Privacy Definition

- For a business organization, it is often more important to attract new users than to expand the item pool. Therefore, a reasonable approach is for partners *a* and *b* to use their local data to solve the cold-start user problem by collaboratively recommending reasonable items to users $u \in \mathcal{U}_b$ using the model prediction function $\mathrm{Predict}(\cdot)$, e.g., the dot product $U_u V_i^T$ for $i \in \mathcal{I}_a$ in PMF.
- As mentioned above, partner *a* can recommend reasonable items to users $u, u \in \mathcal{U}_b$ with the help of $U_u, u \in \mathcal{U}_b$. However, if $U_u, u \in \mathcal{U}_b$ are sent directly to partner *a*, the privacy of partner *b* is compromised, which undermines the goal of sustainable commercialization. This process is shown in the upper right corner of Figure 1. Therefore, in the inference phase, it is important to protect the user latent feature vectors U_u for $u \in \mathcal{U}_b$.

Collaborative Inference Framework

Algorithm 2 Collaborative inference of FedCORE w.r.t. partner *a*.

```
1: Step 1. Send the ID of user $u$ to whom partner $a$ wants to recommend items.
2: Step 2. In order to protect $U_u, u \in \mathcal{U}_b$, partner $b$ returns the encrypted user latent feature vector $E[U_u + \epsilon]_b$.
3: Step 3. Partner $a$ sends $\text{Predict}(E[U_u + \epsilon], V_i, \mathbf{W})$ to partner $b$.
4: Step 4. Partner $b$ computes $D[\text{Predict}(E[U_u + \epsilon], V_i, \mathbf{W})]_b$ and returns $\text{Predict}(U_u, V_i, \mathbf{W})$ to partner $a$.
```

Data Attack Methods

- Although the above approach avoids directly transmitting the user feature vectors, there is still a risk of leakage. We propose two types of attack methods that aim to ‘steal’ the users’ latent vectors from the other partner.
- It is worth noting that one of the attack methods, the accuracy reference (AR) attack, is specific to matrix factorization-based models such as PMF and some deep learning-based models in which the final rating is calculated using the dot product between the user and item latent vectors [Luo et al., 2022, Yuan et al., 2022, Wu et al., 2022].

Data Attack Methods

- **Training (TR) attack.** In the inference phase, partner *a* only lacks knowledge of the user latent feature vectors U_u for $u \in \mathcal{U}_b$. Therefore, partner *a* can attempt to 'steal' these unknown vectors by directly training them on the historical requested ratings $\overline{\mathcal{R}}_b$ from partner *b* for each user *u* in the inference phase.
  The optimization problem is as follows.

  $$U_u'^{*} = \arg\min_{U_u'} \sum_{r_{ui} \in \overline{\mathcal{R}}_b} \left\| r_{ui} - \mathrm{Predict}(U_u', V_i, \mathbf{W}) \right\|^2 , \tag{15}$$

  where $u \in \mathcal{U}_b$ and the function $\mathrm{Predict}(\cdot)$ depends on the specific recommendation algorithm.

Data Attack Methods

- **Accuracy reference (AR) attack.** In PMF, assuming that the dimension of U_u is d, partner a can inquire ratings of d different items w.r.t. user u. The user latent feature vector U_u can be obtained by solving the following system of linear equations.

$$\begin{cases} r_{u1} = & U_u'^* V_1^T \\ r_{u2} = & U_u'^* V_2^T \\ \quad\vdots & \\ r_{ud} = & U_u'^* V_d^T \end{cases} \tag{16}$$

where each equation corresponds to an item and its rating w.r.t. user u.

Privacy Protection

In the inference phase, the acquisition of an individual user's anticipated item rating is imperative for the execution of specific business operations by the collaborating entity. Consequently, privacy preservation strategies that rely on the amalgamation of numerous ratings prove infeasible in this context.

- **Add noise to user embeddings.** If we apply differential privacy to the user embeddings in step 2 of the collaborative inference framework, the final inferred ratings will also be noisy. This makes the attack results biased. The formula is as follows,

$$U'_u = \mathrm{PrivateCOR}(U_u, \epsilon), \quad u \in \mathcal{U}_b. \tag{17}$$

Collaborative Training Framework

- In the training phase, the total communication count is $T \times K \times |\mathcal{U}_a^k \cap \mathcal{U}_{com}|$ via Eq.(12), and $T \times K \times |\{r_{ui}\}|$ via Eq.(13) and Eq.(14), where $r_{ui} \in \mathcal{R}_a^k$ and $u \in \mathcal{U}_{com}$. The size of $|\{r_{ui}\}|$ is smaller than $|\mathcal{U}_a^k \cap \mathcal{U}_{com}|$ due to the utilization of averaging the user latent feature vector gradients in Eq.(13) and Eq.(14).
- In the inference phase, every request for a user rating undergoes four distinct stages, including two transmissions and two receptions. Therefore, for each instance of inference, the communication count stands at 2.

Research Questions

- RQ1) What is the influence of the unaligned items?
- RQ2) What is the protective efficacy of our FedCORE in the training phase?
- RQ3) How does our FedCORE perform in the inference phase?
- RQ4) How does our FedCORE perform with different proportions of overlapping users?
- The source code and scripts to reproduce all the results are publicly available at https://github.com/seek-up-well/FedCORE-code.

Datasets

- MovieLens 100K (ML100K)
- MovieLens 1M (ML1M)
- A randomly constructed subset of Netflix with 5,000 users and 5,000 items[1] (NF5K5K)

[1] http://www.netflix.com/

Datasets

We process each dataset with the following steps:

- We randomly divide each dataset into 60% training data, 20% validation data, and 20% test data.
- We randomly split the users in each part (i.e., training, validation, and test data) into two groups, i.e., 25% of the users assigned to group *a* and 25% assigned to group *b*, and the remaining 50% of the users serve as common users for both groups. We use a similar process for dividing the items into two groups.
- To simulate the same user preferences across different platforms, we assign 20% of the data from common users and common items to both group *a* and group *b*.

Datasets

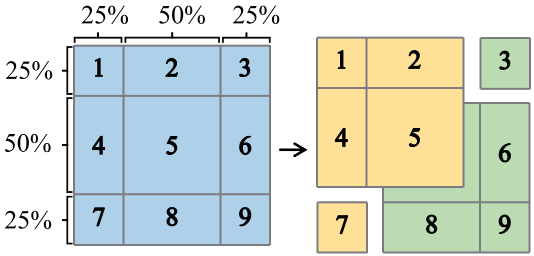


Figure: Illustration of data processing for two partners.

Datasets

- To better explain how we divide the data for simulation, we use a schematic diagram shown in Figure 2.
- The blue squares represent the original dataset, and the yellow and green squares represent the two simulated data after separation. In each square, the rows represent users and the columns represent items.
- The original data, represented by the blue squares, is divided into 9 parts, each identified by a number.
- After separation, parts 1, 2, and 4 are unique to partner *a*, and parts 6, 8, and 9 are unique to partner *b*.
- For part 5, each partner receives 40% at random and also shares 20% of the data from common users and common items with the other partner, as mentioned above.
- Parts 3 and 7 are used to study the recommendation performance in the inference phase but are not involved in the training phase.

Datasets

- In RQ1, different ratios (e.g., 0.1, 0.2) of common items are not aligned, meaning they are mapped to a new set of item IDs.
- In RQ2 and RQ3, all common items are not aligned, meaning they are all mapped to a new set of item IDs.

Parameters setting

- For all experiments, we set the total number of iterations T to 50, and the batch size is set to 256.
- In order to directly compare the effects of the two algorithms when under attack, we use the same optimizer SGD.
- We search for the best value of the learning rate $\gamma \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$ for PMF and NeuCF using the validation data of each dataset.

Evaluation Metrics

- For performance evaluation of each algorithm, we use two commonly used metrics, i.e., mean absolute error (MAE) and root mean square error (RMSE), and report the average performance on the five copies of each dataset.

Influence of Entirely Unaligned Items

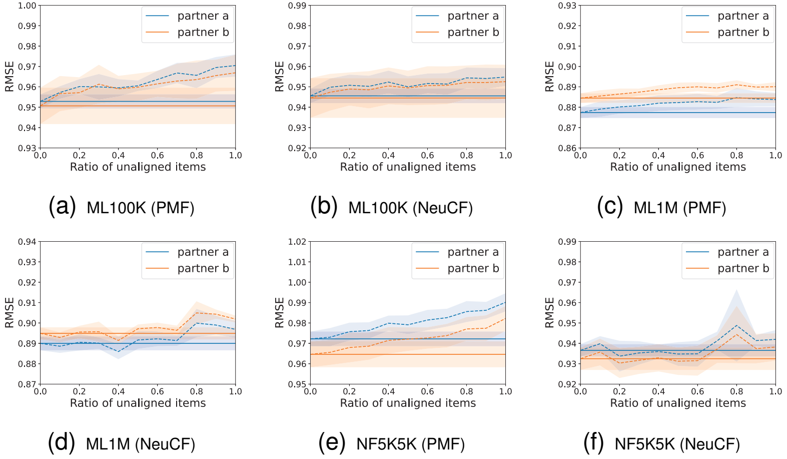


Figure: Recommendation performance w.r.t. different amounts of unaligned items between partners a and b, where a solid line denotes the performance of the corresponding entirely aligned items, and a dashed line denotes the performance of the unaligned items with different ratios.

Influence of Entirely Unaligned Items

- From a modeling perspective, NeuCF performs better than PMF in terms of the average performance gap on all three datasets. This indicates that the influence of unaligned items is related to the complexity of the model, and NeuCF has more parameters, which makes it more robust and able to perform better.
- From the perspective of the dataset, the experimental results are better for ML1M than for ML100K. It is reasonable to conclude that as the quantity of data (i.e., the number of users and items, and their interactions) increases, the influence of entirely unaligned items on the recommendation performance becomes weaker.

Protective Efficacy in the Training Phase

- Because the latent feature vectors of the same item ID tend to be similar, even if they are associated with different and unaligned IDs, partner b may still know the ID i corresponds to in the real world. Therefore, the rating records of partner a are likely to be leaked.
- To demonstrate this risk of privacy disclosure, for every latent vector of a common item in partner a, we calculate its cosine similarity with all the latent feature vectors of the items in partner b at various iterations of the training process, e.g., $t = 0, k = 1$, $t = 0, k = 5$, etc.

Protective Efficacy in the Training Phase

Table: Number of items with top 1% and 5% cosine similarity values in PMF w.r.t. a same item ID. Note that there are 842, 1976 and 2500 common and unaligned items in these three datasets, respectively.

Dataset	t	k	Top 1%	Top 5%
ML100K	0	0	4.80 ±0.40	22.40 ±1.20
	0	5	4.80 ±0.40	22.00 ±1.26
	0	20	5.00 ±0.63	22.40 ±1.62
	1	0	4.60 ±1.50	21.40 ±1.20
	5	0	13.20 ±2.56	56.00 ±6.84
	20	0	25.40 ±3.72	89.60 ±12.18
	49	0	71.80 ±7.98	154.8 ±5.42
ML1M	0	0	11.00 ±0.00	76.00 ±0.00
	0	5	11.00 ±0.00	75.60 ±0.49
	0	20	11.20 ±0.40	75.80 ±0.75
	1	0	19.00 ±1.41	114.0 ±2.61
	5	0	44.40 ±7.23	169.8 ±11.41
	20	0	185.4 ±10.52	473.0 ±20.68
	49	0	455.0 ±13.08	783.8 ±14.30
NF5K5K	0	0	18.00 ±0.00	98.40 ±0.80
	0	5	18.00 ±0.00	98.40 ±1.20
	0	20	18.40 ±0.80	97.00 ±0.63
	1	0	20.40 ±2.06	98.40 ±3.20
	5	0	44.80 ±5.04	187.6 ±12.22
	20	0	117.0 ±8.32	326.6 ±9.89
	49	0	265.6 ±6.15	505.8 ±13.93

Protective Efficacy in the Training Phase

Table: Number of items with top 1% and 5% cosine similarity values in NeuCF w.r.t. a same item ID. Note that there are 842, 1976 and 2500 common and unaligned items in these three datasets, respectively.

Dataset	t	k	Top 1%		Top 5%	
			NCF	GMF	NCF	GMF
ML100K	0	0	4.80 ±0.40	2.20 ±0.40	33.00 ±4.00	28.20 ±0.40
	0	5	4.40 ±1.20	2.20 ±1.20	32.80 ±3.76	28.60 ±0.49
	0	20	6.60 ±1.74	2.20 ±1.74	33.00 ±2.90	28.20 ±0.40
	1	0	12.80 ±2.04	3.00 ±2.04	63.80 ±6.31	29.00 ±0.89
	5	0	15.80 ±4.96	3.80 ±4.96	73.60 ±6.15	29.80 ±1.17
	20	0	20.20 ±3.82	4.60 ±3.82	79.20 ±11.2	30.40 ±3.20
	49	0	19.60 ±5.04	5.20 ±5.04	81.20 ±10.8	31.20 ±2.99
ML1M	0	0	14.00 ±0.00	20.00 ±0.00	72.00 ±0.00	70.00 ±0.00
	0	5	14.60 ±1.20	19.20 ±1.20	71.80 ±4.62	69.60 ±0.49
	0	20	13.60 ±1.74	20.00 ±1.74	75.60 ±2.65	69.40 ±0.49
	1	0	53.40 ±4.96	20.60 ±4.96	198.2 ±6.62	66.80 ±1.72
	5	0	102.4 ±14.6	20.60 ±14.6	314.6 ±30.0	69.00 ±4.15
	20	0	185.0 ±15.4	21.40 ±15.4	484.8 ±19.1	80.20 ±2.48
	49	0	205.4 ±18.8	23.40 ±18.8	503.4 ±15.2	82.20 ±3.43
NF5K5K	0	0	15.40 ±1.20	15.40 ±1.20	82.00 ±8.0	73.00 ±0.00
	0	5	16.40 ±2.33	15.40 ±2.33	84.40 ±6.89	73.20 ±0.40
	0	20	19.00 ±3.10	15.40 ±3.10	89.80 ±6.97	73.00 ±0.00
	1	0	46.80 ±4.31	15.00 ±4.31	177.2 ±11.8	73.00 ±0.63
	5	0	47.00 ±6.23	14.40 ±6.23	191.2 ±8.61	73.00 ±1.41
	20	0	47.80 ±4.26	13.80 ±4.26	204.2 ±6.14	72.20 ±1.33
	49	0	49.80 ±4.49	13.60 ±4.49	206.2 ±6.79	72.60 ±1.36

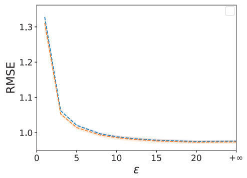
Protective Efficacy in the Training Phase

- We report the number of items with top 1% and 5% cosine similarity with respect to a same but not aligned item ID in Tables 7 and 8.
- We can see that the numbers increase as the training progresses, which means thateven if the item IDs are not aligned, the latent vectors of the same item ID are close, posing a threat to privacy protection.
- Additionally, from the perspective of different models, NeuCF is much less similar than PMF. We believe this is due to the intermediate layer parameters taking on some modeling capability, so that the input features become less important.

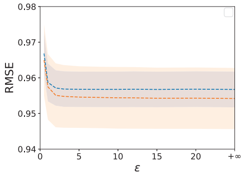
Protective Efficacy in the Training Phase

- We then study the influence of different amounts of privacy budget in the training phase and report the results in Figure 4. As the privacy budget ϵ_1 increases, the recommendation performance becomes better. The choice of $\epsilon_1 = 15$ is a good trade-off between recommendation performance and privacy protection.

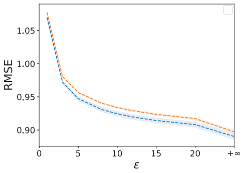
Protective Efficacy in the Training Phase



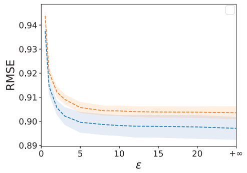
(a) ML100K (PMF)



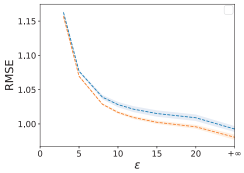
(b) ML100K (NeuCF)



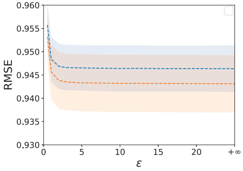
(c) ML1M (PMF)



(d) ML1M (NeuCF)



(e) NF5K5K (PMF)



(f) NF5K5K (NeuCF)

Figure: Recommendation performance with different values of the privacy budget ϵ_1 between partners *a* and *b* w.r.t different datasets and models.

Protective Efficacy in the Training Phase

To study the defense effect of our method, we randomly select 100 common users who rate at least five different items. All metrics are described as follows:

- Sim(1) is computed between the items' latent feature vectors from $V_i, i \in \mathcal{I}_a$ stored in partner a and $V_i'^{*}, i \in \mathcal{I}_a$ attacked by partner b.
- Sim(2) is computed between the items' latent feature vectors from $V_i, i \in \mathcal{I}_a$ stored in partner a and $V_i, i \in \mathcal{I}_b$ with the same items' IDs.
- Sim(3) is computed between $V_i'^{*}, i \in \mathcal{I}_a$ attacked by partner b and the items' latent feature vectors from $V_i, i \in \mathcal{I}_b$ with the same item IDs.
- RMSE is computed from the train data w.r.t. partner a and common users.

Protective Efficacy in the Training Phase

Table: Cosine similarity and recommendation performance of different methods for uploading gradient. The baseline refers to that of uploading a single gradient w.r.t. PMF.

Dataset	Method	Sim(1)	Sim(2)	Sim(3)	RMSE
ML100K	baseline	1.000 $\pm$0.000	0.992 $\pm$0.046	0.992$\pm$0.046	0.950
	AG	0.300 $\pm$0.924	0.297 $\pm$0.918	0.994$\pm$0.007	2.364
	DP	0.311 $\pm$0.237	0.309 $\pm$0.236	0.992$\pm$0.046	1.962
	DP+AG	0.070 $\pm$0.339	0.068 $\pm$0.338	0.994$\pm$0.007	2.401
ML1M	baseline	1.000 $\pm$0.000	0.990 $\pm$0.014	0.990$\pm$0.014	0.868
	AG	0.261 $\pm$0.850	0.254 $\pm$0.852	0.991$\pm$0.012	2.435
	DP	0.425 $\pm$0.205	0.422 $\pm$0.204	0.990$\pm$0.014	2.728
	DP+AG	0.115 $\pm$0.416	0.112 $\pm$0.413	0.991$\pm$0.012	2.801
NF5K5K	baseline	1.000 $\pm$0.000	0.985 $\pm$0.076	0.985$\pm$0.076	0.991
	AG	0.267 $\pm$0.945	0.264 $\pm$0.938	0.992$\pm$0.013	2.440
	DP	0.267 $\pm$0.258	0.264 $\pm$0.256	0.985$\pm$0.076	1.810
	DP+AG	0.057 $\pm$0.274	0.057 $\pm$0.275	0.992$\pm$0.013	2.334

Protective Efficacy in the Training Phase

Table: Cosine similarity and recommendation performance of different methods for uploading gradient. The baseline refers to that of uploading a single gradient w.r.t. NeuCF.

Dataset	Method	Sim(1)	Sim(2)	Sim(3)	RMSE
		NCF	NCF	NCF	
ML100K	baseline	0.724 ±0.443	0.320 ±0.373	0.432±0.331	1.415
	AG	0.036 ±0.300	0.025 ±0.271	0.432±0.331	2.106
	DP	0.453 ±0.444	0.222 ±0.398	0.432±0.331	1.868
	DP+AG	0.034 ±0.273	0.035 ±0.259	0.432±0.331	2.135
ML1M	baseline	0.158 ±0.477	0.088 ±0.411	0.595±0.349	1.914
	AG	0.005 ±0.313	-0.006 ±0.310	0.668±0.285	2.113
	DP	0.123 ±0.415	0.088 ±0.385	0.668±0.285	2.014
	DP+AG	0.024 ±0.318	0.014 ±0.305	0.668±0.285	2.122
NF5K5K	baseline	0.673 ±0.488	0.230 ±0.397	0.372±0.375	1.252
	AG	-0.013 ±0.308	-0.009 ±0.281	0.372±0.375	1.713
	DP	0.312 ±0.423	0.127 ±0.337	0.372±0.375	1.557
	DP+AG	0.005 ±0.302	0.000 ±0.295	0.372±0.375	1.70

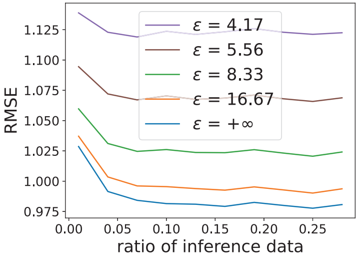
Protective Efficacy in the Training Phase

- Sim(1), which represents the highest similarity score between the original and reconstructed item vectors, consistently registers the highest value in both PMF and NeuCF when gradients are unprotected.
- With the implementation of differential privacy (DP), adversarial gradient (AG), and the combined DP+AG strategies, there is a notable reduction in Sim(1), thereby impeding the attacker's ability to accurately deduce the IDs of the genuine items.
- Among these, the DP+AG strategy emerges as the most potent in shielding the training process against gradient-based intrusions.

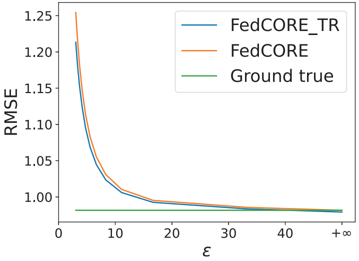
Performance in the Inference Phase

We investigate both the defense performance and recommendation performance in the inference phase. All baselines are described as follows:

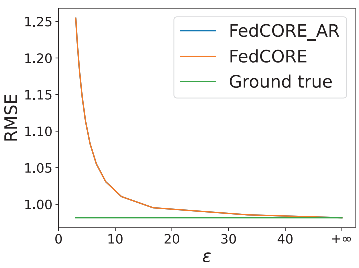
- AvgFilling: uses the global average rating to perform recommendations for new users.
- FedCORE: protects inference performance by adding noise to user embeddings.
- FedCORE_AR: conducts an attack via the accuracy reference (AR) attack, adds noise to user embeddings, and uses the attack users' vector for the recommendation.
- FedCORE_AR2: conducts an attack via the accuracy reference (AR) attack, adds noise to inferred ratings, and uses the attacked users' vector for the recommendation.
- FedCORE_TR: conducts an attack via the training (TR) attack, adds noise to user embeddings, and uses the attack users' vector for the recommendation.



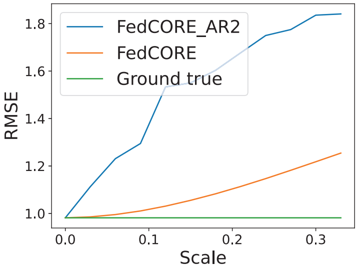
(a) The training attack performance of FedCORE_TR.



(b) The training attack performance of FedCORE_TR and defense performance of FedCORE.

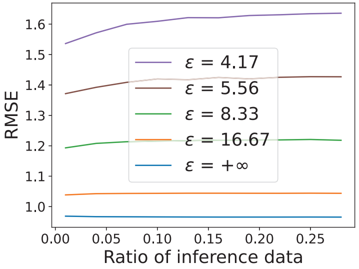


(c) The accuracy attack performance of FedCORE_AR and defense performance of FedCORE.

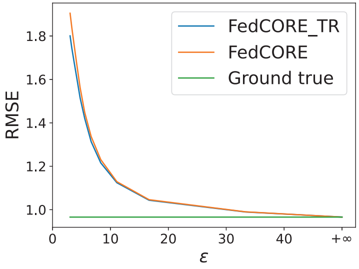


(d) The accuracy attack performance of FedCORE_AR2 and defense performance of FedCORE.

Figure: Recommendation performance and defense performance of our FedCORE with PMF w.r.t. different amounts of privacy budget and different ratios of inference data on ML100K.



(a) The training attack performance of FedCORE_TR.



(b) The training attack performance of FedCORE_TR and defense performance of FedCORE.

Figure: Recommendation performance and defense performance of our FedCORE with NeuCF w.r.t. different amounts of privacy budget and different ratios of inference data on ML100K.

Performance in the Inference Phase

- In Figures 5(b) and 6(b), we assess the inference efficacy of FedCORE against FedCORE_TR. Our observations confirm that the integrity of the original user embeddings can be maintained through the strategic introduction of noise.
- Figure 5(c) demonstrates that the performance of FedCORE_AR in the PMF setting aligns with that of FedCORE.
- The outcomes illustrated in Figure 5(d) suggest that the introduction of minimal noise into the ratings can thoroughly thwart an attack, resulting in an inference performance markedly inferior to that of FedCORE.

Performance in the Inference Phase

Table: Recommendation performance of our FedCORE w.r.t. PMF and partner *a*.

Algorithm	MAE	RMSE
	ML100K	
AvgFilling	$0.9654_{\pm 0.0075}$	$1.1637_{\pm 0.0067}$
FedCORE	$0.752_{\pm 0.0141}$	$0.9594_{\pm 0.0142}$
PMF	$0.7328_{\pm 0.0068}$	$0.9488_{\pm 0.0125}$
	ML1M	
AvgFilling	$0.9374_{\pm 0.0019}$	$1.1173_{\pm 0.0029}$
FedCORE	$0.7082_{\pm 0.002}$	$0.9072_{\pm 0.0019}$
PMF	$0.6925_{\pm 0.0019}$	$0.874_{\pm 0.0028}$
	NF5K5K	
AvgFilling	$0.9125_{\pm 0.0081}$	$1.0866_{\pm 0.0068}$
FedCORE	$0.7791_{\pm 0.011}$	$1.0115_{\pm 0.0114}$
PMF	$0.7479_{\pm 0.0049}$	$0.9925_{\pm 0.0128}$

Performance in the Inference Phase

Table: Recommendation performance of our FedCORE w.r.t. NeuCF and partner *a*.

Algorithm	MAE	RMSE
	ML100K	
AvgFilling	0.9654$\pm$0.0075	1.1637$\pm$0.0067
FedCORE	0.7432$\pm$0.0151	0.9459$\pm$0.015
NeuCF	0.7396$\pm$0.0069	0.9444$\pm$0.0156
	ML1M	
AvgFilling	0.9374$\pm$0.0019	1.1173$\pm$0.0029
FedCORE	0.6991$\pm$0.0045	0.8923$\pm$0.0057
NeuCF	0.6985$\pm$0.0027	0.8796$\pm$0.0026
	NF5K5K	
AvgFilling	0.9125$\pm$0.0081	1.0866$\pm$0.0068
FedCORE	0.7393$\pm$0.0039	0.9464$\pm$0.007
NeuCF	0.7225$\pm$0.0032	0.9421$\pm$0.0081

Performance in the Inference Phase

- The resultant training on a single machine outcomes serve as an upper bound for the federated methods in terms of recommendation performance, augmenting the established baseline, e.g., PMF and NeuCF.
- Our FedCORE is only slightly worse than the single machine-trained methods, e.g., PMF and NeuCF. These methods surpass our FedCORE, primarily attributed to learning from the centralized data.
- Our FedCORE achieves significantly better recommendation performance than AvgFilling.

Performance of Different Proportions of Overlapping Users

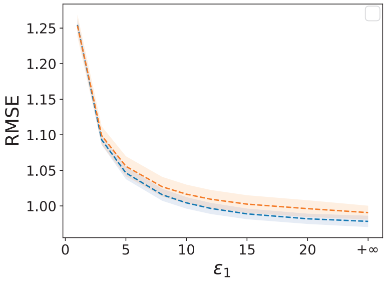
- In this section, we examine the performance of our FedCORE on data with different alignment ratios. We include new experimental results using different data partitions. Specifically, we change the co-user ratio to 35% and 15%.
- Since different data partitions mainly affect the training phase, we focus on the performance in the training phase under different data proportions.
- The results are shown in Table 13, Figure 7 and Table 14. These results clearly demonstrate that the different proportions of data have, in essence, negligible effects on the experimental results. Note that the results are similar on ML1M and NF5K5K, as well as that using NeuCF.

Performance of Different Proportions of Overlapping Users

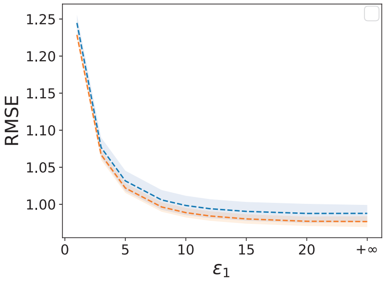
Table: Number of items with top 1% and 5% cosine similarity values w.r.t. a same item ID from ML100K. Note that the numbers of common and unaligned items are 842 and 1178 in different data alignment ratios, respectively.

Ratio	t	k	Top 1%	Top 5%
15%	0	0	4.80 $\pm$0.40	11.00 $\pm$0.00
	5	0	3.80 $\pm$1.60	15.20 $\pm$2.14
	20	0	5.00 $\pm$2.53	19.00 $\pm$2.28
	49	0	15.60 $\pm$1.62	44.60 $\pm$4.45
25%	0	0	4.80 $\pm$0.40	22.40 $\pm$1.20
	5	0	13.20 $\pm$2.56	56.00 $\pm$6.84
	20	0	25.40 $\pm$3.72	89.60 $\pm$12.18
	49	0	71.80 $\pm$7.98	154.80 $\pm$5.42
35%	0	0	3.80 $\pm$0.40	36.00 $\pm$0.00
	5	0	40.20 $\pm$4.17	149.00 $\pm$14.09
	20	0	154.80 $\pm$6.11	326.00 $\pm$10.90
	49	0	192.40 $\pm$9.41	331.20 $\pm$15.66

Performance of Different Proportions of Overlapping Users



(a) Ratio 15%.



(b) Ratio 35%.

Figure: Recommendation performance on ML100K w.r.t. different values of the privacy budget ϵ_1 between partners *a* and *b* with different data alignment ratios.

Performance of Different Proportions of Overlapping Users

Table: Cosine similarity values and recommendation performance of different methods and data alignment ratios in uploading the gradients. The baseline refers to that of uploading a single gradient w.r.t. PMF.

Ratio	Method	Sim(1)	Sim(2)	Sim(3)	RMSE
15%	baseline	1.000 ±0.000	0.950 ±0.083	0.950±0.083	0.956
	AG	0.291 ±0.920	0.275 ±0.877	0.946±0.094	2.383
	DP	0.311 ±0.238	0.297 ±0.233	0.950±0.083	2.002
	DP+AG	0.105 ±0.323	0.094 ±0.310	0.946±0.094	2.372
25%	baseline	1.000 ±0.000	0.992 ±0.046	0.992±0.046	0.950
	AG	0.300 ±0.924	0.297 ±0.918	0.994±0.007	2.364
	DP	0.311 ±0.237	0.309 ±0.236	0.992±0.046	1.962
	DP+AG	0.070 ±0.339	0.068 ±0.338	0.994±0.007	2.401
35%	baseline	1.000 ±5.566	0.993 ±8.000	0.993±0.080	0.984
	AG	0.265 ±0.924	0.267 ±0.922	0.994±0.066	2.385
	DP	0.307 ±0.233	0.305 ±0.233	0.993±0.080	1.970
	DP+AG	0.075 ±0.330	0.075 ±0.331	0.994±0.066	2.331

Conclusions and Future Work

- We address a new and important problem, i.e., cross-organization federated recommendation, and propose a novel and privacy-aware solution called federated cross-organization recommendation ecosystem (FedCORE).
- Our FedCORE allows different organizations to collaboratively train on cross-silo data and infer the preferences of their cold-start users.
- We conduct extensive experiments on three real-world datasets and two seminal recommendation models to study the effectiveness of cooperation and privacy protection in our proposed ecosystem.
- For future works, we plan to generalize our FedCORE to recommendation problems involving users' sequential and multi-behavior data.

Thank you!

- We thank the handling editors and reviewers for their effort and constructive expert comments, as well as the support of the National Natural Science Foundation of China (Nos. 62172283 and 62272315) and the National Key Research and Development Program of China (No. 2018AAA0101100).

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016).
Deep learning with differential privacy.
In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318.

Dwork, C., Roth, A., et al. (2014).
The algorithmic foundations of differential privacy.
Foundations and Trends in Theoretical Computer Science, 9(3–4):211–407.

Luo, J., He, M., Lin, X., Pan, W., and Ming, Z. (2022).
Dual-task learning for multi-behavior sequential recommendation.
In *Proceedings of the 31th ACM International Conference on Information and Knowledge Management*, pages 1379–1388.

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. (2017).
Learning differentially private recurrent language models.
arXiv preprint arXiv:1710.06963.

Wu, C., Wu, F., Qi, T., Liu, Q., Tian, X., Li, J., He, W., Huang, Y., and Xie, X. (2022).
FEEDREC: News feed recommendation with various user feedbacks.
In *Proceedings of the 31th International Conference on World Wide Web*, pages 2088–2097.

Yuan, E., Guo, W., He, Z., Guo, H., Liu, C., and Tang, R. (2022).
Multi-behavior sequential transformer recommender.
In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1642–1652.

Zhu, L., Liu, Z., and Han, S. (2019).
Deep leakage from gradients.
32.