

FedRec: Federated Recommendation With Explicit Feedback

Guanyu Lin , Feng Liang, Weike Pan , and Zhong Ming , Shenzhen University, Shenzhen 518060, China

Recommendation models have been widely embedded in various online services, while most of which are designed with the assumption that users' original behaviors are available in a central server. This may cause the privacy issue. As a response, we follow a recent work called federated collaborative filtering (FCF) for item recommendation with implicit feedback and propose a novel and generic federated recommendation (FedRec) framework for rating prediction with explicit feedback. Specifically, we federate some basic and advanced factorization-based recommendation models both in batch style and in stochastic style. More importantly, in order to protect the private information of which items each user has rated, as well as not to significantly increase the computational and communication cost, we design two simple but effective strategies, i.e., user averaging and hybrid filling, in which some (instead of all) unrated items are randomly sampled and assigned with some virtual ratings accordingly. Empirical studies on two public datasets show the effectiveness of our FedRec in terms of the closeness of a federated model and an unfederated one, and the usefulness of the two filling strategies.

With the growing number of choices in our daily life, recommendation is becoming more and more important. Actually, recommendation models have been studied for a few decades in the community of recommender systems, among which collaborative filtering (CF)^{5,11} is a very popular technique. However, European Union (EU) has recently enacted General Data Protection Regulation (GDPR)⁴ that forbids commercial companies to collect, deal with, or exchange a user's data without the permission of the corresponding user. Besides, similar regulations are being enacted in the United States and China.¹⁸ That is to say, machine learning techniques, including recommendation models, are facing an isolated and fragmented data problem. As the traditional CF models highly rely on different users' behavior data in order to build a *collaborative* model as some other machine learning models do,¹⁸ CF is unavoidably facing a new challenge about preference modeling and privacy protection.

Fortunately, first proposed by Google, federated machine learning (FML)^{2,10,13} aims at enabling a machine learning model to address this specific challenge. Currently, FML has obtained decent performance in some advanced machine learning fields such as reinforcement learning,¹⁹ transfer learning,¹² and meta learning.³ In general, FML is a new learning paradigm that builds a prediction model by privacy-aware rich interactions between a central server and some local clients. From this perspective, FML has the potential to build a recommendation model with isolated and fragmented data stored in different clients without sacrificing the users' privacy.

In a recent and closely related work,¹ a federated framework for item ranking with implicit feedback (e.g., clicks and examinations) is proposed, which supports both alternative least square (ALS) and stochastic gradient descent (SGD) optimization methods. In particular, in order to model the implicit feedback, all the uninteracted (user, item) pairs are treated as negative feedback. However, this may cause the efficiency issue in both computation and communication. Some other works combine federated learning with meta learning⁹ or gossip learning⁷ for rating prediction and low-rank matrix decomposition, respectively. However, in these

TABLE 1. Some notations and explanations used in the paper.

n	Number of users
m	Number of items
$\mathfrak{R} = \{1, \dots, 5\}$	Rating range
$r_{ui} \in \mathfrak{R}$	Rating of user u to item i
$\mathcal{R} = \{(u, i, r_{ui})\}$	Rating records in training data
\mathcal{R}_u	Rating records w.r.t. user u in \mathcal{R}
$\mathcal{R}^{te} = \{(u, i, r_{ui})\}$	Rating records in test data
\mathcal{I}	The whole set of items
\mathcal{I}_u	Items rated by user u
$\mathcal{I}'_{u'}, \mathcal{I}'_{u'} = \rho \mathcal{I}_u $	Sampled items w.r.t. user u
\mathcal{U}	The whole set of users
\mathcal{U}_i	Users who rated item i
\mathcal{U}'_i	Users w.r.t. sampled item i
$y_{ui} \in \{0, 1\}$	Indicator variable
$d \in \mathbb{R}$	Number of latent dimensions
$U_u \in \mathbb{R}^{1 \times d}$	User-specific latent feature vector
$V_i, W_{i'} \in \mathbb{R}^{1 \times d}$	Item-specific latent feature vector
\hat{r}_{ui}	Predicted rating of user u to item i
γ	Learning rate
ρ	Sampling parameter
λ	Tradeoff parameter
T	Iteration number

two works, the server can easily identify the rating behaviors (i.e., the rated items) when a user uploads the model parameters, which thus may not protect the users' privacy well.

In this article, we follow¹ and design a federated recommendation framework for rating prediction with explicit feedback (e.g., numerical rating scores). Specifically, in our studied problem, we have n users (i.e., clients) and m items, where each user u has rated a set of items \mathcal{I}_u resulting in some rating records $\mathcal{R}_u = \{(u, i, r_{ui}); i \in \mathcal{I}_u\}$. Our goal is then to predict the rating of a user u to each item $j \in \mathcal{I} \setminus \mathcal{I}_u$ without sharing the rating behaviors (i.e., \mathcal{I}_u) or the rating records (i.e., \mathcal{R}_u) with other users and the server. We put some commonly used notations in Table 1.

We can see that our studied problem is very different from those in the paper by Ammad-ud-din *et al.*,¹ and the traditional recommendation works,¹⁷ which requires us to design a rating prediction method in a different *collaborative* manner. In particular, the techniques in the paper by Ammad-ud-din *et al.*¹ cannot be applied directly to our studied problem, because treating all the un-interacted (user, item) pairs as negative feedback will bias the model training and also increase the computational and communication cost. As a response, we first design two strategies, i.e., user averaging (UA) and hybrid filling (HF), to address this issue, and then propose a generic factorization-based federated recommendation framework called FedRec. Under the framework of our FedRec, we federate

PMF¹⁴ in the batch style, as well as PMF and SVD++¹¹ in the stochastic style to show the generality and effectiveness of our solution, where the batch style and stochastic style refer to the ways of calculating the gradients of a user u , i.e., using all the items w.r.t. user u and using only one randomly sampled item w.r.t. user u , respectively.¹⁵

RELATED WORK

Factorization-Based Methods for Rating Prediction With Explicit Feedback

In probabilistic matrix factorization (PMF),¹⁴ the rating of a user $u \in \{1, 2, \dots, n\}$ to an item $i \in \{1, 2, \dots, m\}$ is predicted as the inner product of two learned vectors,

$$\hat{r}_{ui} = U_u \cdot V_i^T \quad (1)$$

where $U_u \in \mathbb{R}^{1 \times d}$ and $V_i \in \mathbb{R}^{1 \times d}$ are the latent feature vectors of user u and item i , respectively.

In SVD++,¹¹ the rating of a user u to an item i is estimated by exploiting the other rated items by user u ,

$$\hat{r}_{ui} = U_u \cdot V_i^T + \frac{1}{\sqrt{|\mathcal{I}_u \setminus \{i\}|}} \sum_{i' \in \mathcal{I}_u \setminus \{i\}} W_{i'} \cdot V_i^T \quad (2)$$

where \mathcal{I}_u denotes the items rated by user u , $W_{i'} \in \mathbb{R}^{1 \times d}$ is the latent feature vector of item i' , and $\frac{1}{\sqrt{|\mathcal{I}_u \setminus \{i\}|}}$ is a normalization term. Notice that the difference between SVD++ and PMF is the second term in (2), i.e., $\frac{1}{\sqrt{|\mathcal{I}_u \setminus \{i\}|}} \sum_{i' \in \mathcal{I}_u \setminus \{i\}} W_{i'} \cdot V_i^T$, which is built on the assumption that users with similar rated items will usually have similar taste.

FCF for Item Ranking With Implicit Feedback

In FCF,¹ the authors propose a first federated learning framework for item ranking with implicit feedback. Specifically, they upload an intermediate gradient $\nabla V_{\text{IF}}(u, i)$ to the server instead of the user's original data so as to protect the user's privacy,

$$\nabla V_{\text{IF}}(u, i) = (1 + \alpha y_{ui})(U_u \cdot V_i^T - y_{ui})U_u \quad (3)$$

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 $\alpha > 0$. Notice that all the un-interacted (user, item) pairs w.r.t. a certain user u are treated as negative feedback, i.e., $y_{ui} = 0$ for $i \in \mathcal{I} \setminus \mathcal{I}_u$, as shown in (3), which will protect the user's privacy because the items in \mathcal{I}_u are difficult to be identified by the server. However, this strategy will significantly increase both the computational cost and the communication cost.

As another notice, for the problem of rating prediction with explicit feedback as studied in this article, we usually do not model the unobserved records and will thus have

$$\nabla V_{\text{EF}}(u, i) = y_{ui}(U_u V_i^T - r_{ui})U_u \quad (4)$$

which will cause a leakage of a user u 's privacy because the items in \mathcal{I}_u can then be easily identified by the server. And if we treat all the unobserved records as a negative feedback as that in the paper by Ammad-ud-din *et al.*,¹ we will bias the model training toward lower predicted scores. In a summary, we cannot directly apply FCF¹ to the problem of rating prediction with explicit feedback studied in this article, which motivates us to design a new federated solution.

OUR SOLUTION: FEDERATED RECOMMENDATION

In this section, we describe our proposed solution, i.e., federated recommendation (FedRec), for rating prediction with explicit feedback.

In order to protect users' privacy in rating prediction, in particular of what items user u has rated (i.e., the rating behaviors in \mathcal{I}_u), we propose two simple but effective strategies, i.e., UA and HF. Specifically, we first randomly sample some unrated items $\mathcal{I}'_u \subseteq \mathcal{I} \setminus \mathcal{I}_u$ for each user u , and then assign a virtual rating r'_{ui} to each item $i \in \mathcal{I}'_u$,

$$r'_{ui} = \bar{r}_u = \frac{\sum_{k=1}^m y_{uk} r_{uk}}{\sum_{k=1}^m y_{uk}} \quad (5)$$

$$r'_{ui} = \hat{r}_{ui} \quad (6)$$

where \bar{r}_u denotes the average rating value of a user u to the rated items in \mathcal{I}_u , and \hat{r}_{ui} denotes the predicted rating value of a user u to an unrated item i in \mathcal{I}'_u . We show the details of the two strategies in Algorithm 3, with which we can obtain a virtual rating r'_{ui} for each sampled item $i \in \mathcal{I}'_u$ and then have a combined set of rating records w.r.t. user u , i.e., $\mathcal{R}'_u \cup \mathcal{R}_u$ with $\mathcal{R}'_u = \{(u, i, r'_{ui}), i \in \mathcal{I}'_u\}$.

The combined rating records for each user u can actually hit three birds with one stone, i.e., it can address the privacy issue, the efficiency issue and the accuracy issue. First, with the combined item set, i.e., $\mathcal{I}'_u \cup \mathcal{I}_u$, it will be more difficult for the server to identify what items the corresponding user u has rated, which thus protects the user's privacy in terms of his or her rating behaviors. Second, the way of sampling some unrated items instead of taking all the unrated items in the paper by Ammad-ud-din *et al.*¹ will not significantly increase the computational and communication cost.

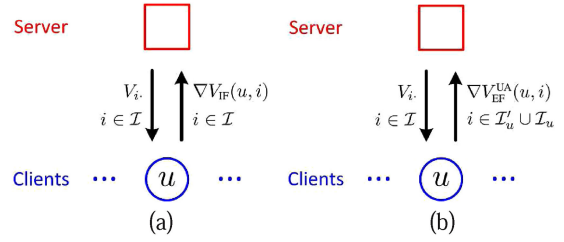


FIGURE 1. Illustration of the interactions between the server and each client in FCF for item ranking with implicit feedback (a) and our FedRec using the user average (UA) strategy for rating prediction with explicit feedback (b).

Third, assigning a virtual rating via an average score or a predicted score instead of a negative score in the paper by Ammad-ud-din *et al.*¹ will not bias the learning process of model parameters much, which is also observed in our empirical studies.

We illustrate the main interactions between the server and each client u of FCF¹ and our FedRec in Figure 1. We can see that the main difference is the content to be uploaded from each client to the server, besides the input of the studied problems, i.e., implicit feedback in FCF¹ and explicit feedback in our FedRec.

Moreover, our FedRec with the UA strategy and the HF strategy is a generic framework that can be used in factorization-based rating prediction models in both the batch style and stochastic style, which will be discussed in the subsequent sections.

FedRec in the Batch Style

In this section, we describe our FedRec for a basic factorization-based model, i.e., PMF, in the batch style.

The interactions between the server and each client, as illustrated in Figure 1, are briefly listed as follows:

- The server randomly initializes the model parameters, i.e., $V_i, i = 1, 2, \dots, m$, with small random values.
- Each client u downloads the item-specific latent feature vectors, i.e., $V_i, i = 1, 2, \dots, m$, from the server.
- Each client u conducts local training with his/her own local data as well as the model parameters downloaded from the server.
- Each client u uploads the gradients, i.e., $\nabla V_{\text{EF}}^{\text{UA}}(u, i), i \in \mathcal{I}'_u \cup \mathcal{I}_u$, to the server.
- The server updates the item-specific latent feature vectors with $\nabla V_{\text{EF}}^{\text{UA}}(u, i)$ received from the clients.

Algorithm 1. The algorithm of batch FedRec for PMF in the server.

```

1: Initialize the model parameters  $V_i, i = 1, 2, \dots, m$ 
2: for  $t = 1, 2, \dots, T$  do
3:   for each client  $u$  in parallel do
4:     ClientBatch( $V_i, i = 1, 2, \dots, m; u; t$ ).
5:   end for
6:   for  $i = 1, 2, \dots, m$  do
7:     Calculate the gradient  $\nabla V_i$  via (9).
8:     Update  $V_i$  via  $V_i \leftarrow V_i - \gamma \nabla V_i$ .
9:   end for
10:  Decrease the learning rate  $\gamma \leftarrow 0.9\gamma$ .
11: end for

```

We will then describe the learning procedures in the client and in the server separately in detail.

Batch FedRec for PMF in the Client

Once a client u has downloaded the item-specific latent feature vectors $V_i, i = 1, \dots, m$ from the server, the user u can calculate the gradient ∇U_u without referring to other users' data. After that, the user u can update the user-specific latent feature vector U_u locally. That is to say, without sharing the original rating data among users, each user u can update the model parameter U_u .

In particular, we randomly sample some items \mathcal{I}'_u from $\mathcal{I} \setminus \mathcal{I}_u$ with $|\mathcal{I}'_u| = \rho|\mathcal{I}_u|$, which are then used to augment the original rating records of user u , i.e., \mathcal{I}_u . Notice that ρ is a sampling parameter, which is fixed as a small number such as $\rho = 3$ in our experiments. With the sampled items \mathcal{I}'_u and the UA strategy or the HF strategy, we have the gradient ∇U_u as follows:

$$\nabla U_u = \frac{\sum_{i \in \mathcal{I}'_u \cup \mathcal{I}_u} [(U_u \cdot V_i^T - y_{ui} r_{ui} - (1 - y_{ui}) r'_{ui}) V_i + \lambda U_u]}{|\mathcal{I}'_u \cup \mathcal{I}_u|} \quad (7)$$

where r'_{ui} is the average rating of user u to the rated items in \mathcal{I}_u in (5) or the predicted rating of user u to an unrated item i from \mathcal{I}'_u in (6).

We can then calculate the gradients, i.e., $\nabla V_{\text{EF}}^{\text{UA}}(u, i), i \in \mathcal{I}'_u \cup \mathcal{I}_u$ locally with user u 's own data and the model parameters downloaded from the server,

$$\nabla V_{\text{EF}}^{\text{UA}}(u, i) = \begin{cases} (U_u \cdot V_i^T - r_{ui}) U_u + \lambda V_i, y_{ui} = 1 \\ (U_u \cdot V_i^T - r'_{ui}) U_u + \lambda V_i, y_{ui} = 0. \end{cases} \quad (8)$$

which are then uploaded to the server. Notice that the server cannot identify which items are from \mathcal{I}_u from the set of the uploaded gradients, i.e., $\nabla V_{\text{EF}}^{\text{UA}}(u, i), i \in$

Algorithm 2. ClientBatch($V_i, i = 1, 2, \dots, m; u; t$), i.e., the algorithm of batch FedRec for PMF in the client.

```

1: Sample items  $\mathcal{I}'_u$  from  $\mathcal{I} \setminus \mathcal{I}_u$  with  $|\mathcal{I}'_u| = \rho|\mathcal{I}_u|$ 
2: ClientFilling( $V_i, i = 1, 2, \dots, m; U_u; u; t$ ).
3: Calculate the gradient  $\nabla U_u$  via (7).
4: Update  $U_u$  via  $U_u \leftarrow U_u - \gamma \nabla U_u$ .
5: for  $i \in \mathcal{I}'_u \cup \mathcal{I}_u$  do
6:   Calculate  $\nabla V_{\text{EF}}^{\text{UA}}(u, i)$  via (8).
7: end for
8: Upload  $\nabla V_{\text{EF}}^{\text{UA}}(u, i)$  with  $i \in \mathcal{I}'_u \cup \mathcal{I}_u$  to the server.

```

$\mathcal{I}'_u \cup \mathcal{I}_u$ easily. Hence, the privacy of user u 's rating behaviors is protected.

Batch FedRec for PMF in the Server

Once the server has received the gradients, i.e., $\nabla V_{\text{EF}}^{\text{UA}}(u, i), i \in \mathcal{I}'_u \cup \mathcal{I}_u, u = 1, 2, \dots, n$, it can then calculate the gradient of item i ,

$$\nabla V_i = \frac{\sum_{u \in \mathcal{U}'_i \cup \mathcal{U}_i} \nabla V_{\text{EF}}^{\text{UA}}(u, i)}{|\mathcal{U}'_i \cup \mathcal{U}_i|} \quad (9)$$

where $\mathcal{U}'_i \cup \mathcal{U}_i$ denotes the users that have rated or virtually rated item i (which cannot be distinguished by the server).

We depict the learning process of the server in Algorithm 1 and that of each client in Algorithm 2.

FedRec in Stochastic Style

In this section, we follow the batch style and obtain the algorithms in the stochastic style, which are shown in Algorithms 4 and 5. Notice that there are two differences compared with that in the batch style. First, at each iteration t , the server samples one user at a time, i.e., n times in total. Second, the server updates V_i after receiving $\nabla V_{\text{EF}}^{\text{UA}}(u, i)$ without parameter aggregation.

Complexity Analysis

The Complexity of FedRec in the Batch Style

For each client u , the worst computational complexity is $T_{\text{local}} \times [(|\mathcal{I}_u| + |\mathcal{I}'_u|) \times d + d] \times |\mathcal{I}'_u| \times d + (|\mathcal{I}_u| + |\mathcal{I}'_u|) \times d + d + (|\mathcal{I}_u| + |\mathcal{I}'_u|) \times d$, which can be rewritten as $[T_{\text{local}} \times (1 + \rho) + 2 + 3\rho] \times d \times |\mathcal{I}_u| + (T_{\text{local}} + 1) \times d$ for $|\mathcal{I}'_u| = \rho|\mathcal{I}_u|$ at each iteration. For the server, the computational complexity is $1 + \sum_{i=1}^m d \times [|\mathcal{U}_i \cup \mathcal{U}'_i| + 1] = 1 + d \times m + d \times |\mathcal{R}| \times (1 + \rho)$ at each iteration. Because ρ, d , and T_{local} are usually fixed, we can see that the computational complexity for each client u and the server are linear with the number of rated

Algorithm 3. ClientFilling($V_i, i = 1, 2, \dots, m; U_u; u; t$), i.e., the algorithm of assigning ratings to the sampled unrated items in the client.

```

1: if strategy == HF then
2:   for  $t_{\text{local}} = 1, 2, \dots, T_{\text{local}}$  do
3:     Calculate the gradient  $\nabla U_u$  via (7).
4:     update  $U_u$  via  $U_u \leftarrow U_u - \gamma \nabla U_u$ .
5:   end for
6:   Assign  $r'_{ui}$  to each item  $i \in \mathcal{I}'_u$  via HF, i.e., use (5) when
    $t < T_{\text{predict}}$  and use (6) when  $t \geq T_{\text{predict}}$ .
7: else
8:   Assign  $r'_{ui}$  to each item  $i \in \mathcal{I}'_u$  via UA, i.e., use (5).
9: end if

```

Algorithm 4. The algorithm of stochastic FedRec for PMF in the server.

```

1: Initialize the model parameters  $V_i, i = 1, 2, \dots, m$ 
2: for  $t = 1, 2, \dots, T$  do
3:   for  $t_2 = 1, 2, \dots, n$  do
4:     Randomly sample a user  $u$  from  $\mathcal{U}$ .
5:     ClientStochastic( $V_i, i = 1, 2, \dots, m; u; t$ ).
6:     for  $i \in \mathcal{I}'_u \cup \mathcal{I}_u$  do
7:       Update  $V_i$  via  $V_i \leftarrow V_i - \gamma \nabla V_{\text{EF}}^{\text{UA}}(u, i)$ .
8:     end for
9:   end for
10:  Decrease the learning rate  $\gamma \leftarrow 0.9\gamma$ .
11: end for

```

items (i.e., $|\mathcal{I}_u|$) and the number of rating records (i.e., $|\mathcal{R}|$), respectively.

For the communication complexity, there is only one communication between each client u and the server at each iteration. The sole communication is that the server gets $\nabla V_{\text{EF}}^{\text{UA}}(u, i)$ that contains $4 \times d \times |\mathcal{I}_u| \times (1 + \rho)$ bytes information from each client u by invoking the function ClientBatch ($V_i, i = 1, 2, \dots, m; u; t$), and the server sends $4 \times d \times m$ bytes information to each client u . To summarize, the communication complexity of each client u is $4 \times d \times [|\mathcal{I}_u| \times (1 + \rho) + m]$ bytes at each iteration, which is linear with the number of rated items by user u . Besides, the communication complexity at each iteration for the server is $\sum_{u=1}^n 4 \times d \times [|\mathcal{I}_u| \times (1 + \rho) + m] = 4 \times d \times [|\mathcal{R}| \times (1 + \rho) + m \times n]$, which is linear with the number of rating records in the training data.

Complexity of FedRec in Stochastic Style

Similarly, for algorithms in the stochastic style, if we assume each user is sampled one time at each iteration t , the communication complexities are $4 \times d \times [|\mathcal{I}_u| \times (1 + \rho) + m]$ and $4 \times d \times [|\mathcal{R}| \times (1 + \rho) + m \times$

Algorithm 5. ClientStochastic($V_i, i = 1, 2, \dots, m; u; t$), i.e., the algorithm of stochastic FedRec for PMF in the client.

```

1: Sample items  $\mathcal{I}'_u$  from  $\mathcal{I} \setminus \mathcal{I}_u$  with  $|\mathcal{I}'_u| = \rho |\mathcal{I}_u|$ 
2: ClientFilling( $V_i, i = 1, 2, \dots, m; U_u; u; t$ ).
3: for  $i \in \mathcal{I}'_u \cup \mathcal{I}_u$  do
4:   Calculate the gradient  $\nabla U_u = (U_u V_i^T - y_{ui} r_{ui} - (1 - y_{ui}) r'_{ui}) V_i + \lambda U_u$ .
5:   Update  $U_u$  via  $U_u \leftarrow U_u - \gamma \nabla U_u$ .
6:   Calculate  $\nabla V_{\text{EF}}^{\text{UA}}(u, i)$  via (8).
7: end for
8: Upload  $\nabla V_{\text{EF}}^{\text{UA}}(u, i)$  with  $i \in \mathcal{I}'_u \cup \mathcal{I}_u$  to the server.

```

$n]$, respectively, for each client u and the server at each iteration t . But the computational complexities in the stochastic style are different from that in the batch style. The computational complexity for the server is $\sum_{u=1}^n [1 + (|\mathcal{I}_u| + |\mathcal{I}'_u|) \times d] = n + d \times |\mathcal{R}| \times (1 + \rho)$ at each iteration t , which is similar to that in the batch style. The worst computational complexity for each client u is $T_{\text{local}} \times [(|\mathcal{I}_u| + |\mathcal{I}'_u|) \times d + d] + |\mathcal{I}'_u| \times d + (|\mathcal{I}_u| + |\mathcal{I}'_u|) \times 3 \times d = [T_{\text{local}} \times (1 + \rho) + 3 + 4\rho] \times d \times |\mathcal{I}_u| + T_{\text{local}} \times d$ at each iteration t , which is higher than that in the batch style.

Moreover, comparing lines 3–5 in Algorithm 1 with lines 3–5 in Algorithm 4, we can see that the former (i.e., batch style) is more efficient because it can run the clients in parallel.

Comparison of Complexity With FCF

From FCF and the paper by Hu *et al.*,⁸ we can see that the computational complexities of a client u and the server are $d^2 \times |\mathcal{I}_u| + d^3 + m \times d$ and $1 + (d \times n \times m)$, respectively. Moreover, we can find that (i) the highest order term of the computational complexities of a client u and the server are $d \times (d \times |\mathcal{I}_u| + m)$ and $n \times m \times d$ in FCF, respectively; (ii) the highest order term of the computational complexities of a client u and the server in our FedRec in the batch style are $[T_{\text{local}} \times (1 + \rho) + 2 + 3\rho] \times d \times |\mathcal{I}_u|$ and $d \times [m + |\mathcal{R}| \times (1 + \rho)]$, respectively; (iii) the highest order term of the computational complexities of a client u and the server in our FedRec in the stochastic style are $[T_{\text{local}} \times (1 + \rho) + 3 + 4\rho] \times d \times |\mathcal{I}_u|$ and $n + d \times |\mathcal{R}| \times (1 + \rho)$, respectively; (iv) and T_{local}, ρ , and d are usually fixed, n, m , and $|\mathcal{R}|$ are usually dependent on a dataset, and $|\mathcal{R}|$ is often far smaller than $n \times m$ in a dataset. Hence, the computational complexities of each client of our FedRec in both the batch style and stochastic style are comparable with that of FCF, and the computational complexities of the server of our FedRec in both

the batch and stochastic styles are smaller than those of FCF.

In FCF, the communication complexities of each client u and the sever at each iteration t are $8 \times m \times d$ and $8 \times n \times m \times d$, respectively. In FCF, we have to upload $\nabla V_{\text{F}}(u, i)$ for $i \in \mathcal{I}$. In our FedRec both in the stochastic style and batch style, we only upload $\nabla V_{\text{EF}}^{\text{UA}}(u, i)$ for $i \in \mathcal{I}'_u \cup \mathcal{I}_u$. Hence, we reduce the communication complexity of $4 \times d \times [m - |\mathcal{I}_u| \times (1 + \rho)]$ at each iteration t for each client and $4 \times d \times [n \times m - |\mathcal{R}| \times (1 + \rho)]$ for the server.

After thorough comparison, we can see that all the complexities of our FedRec are smaller than that of FCF, though the worst computational complexities using the HF strategy in clients are comparable (i.e., both are linear with $|\mathcal{I}_u|$).

Discussions

Although we have described and analyzed our FedRec in the context of a basic matrix factorization model PMF,¹⁴ our proposed framework and learning algorithms can be applied to more advanced factorization-based models such as SVD++ in (2), which is included in our empirical studies.

EXPERIMENTS

We study the effectiveness of our proposed factorization-based federated rating prediction framework FedRec by conducting experiments on two public datasets. In particular, we focus on the following three points: (i) the performance difference between a federated model and an unfederated one, as well as the applicability of our FedRec to different recommendation models; (ii) the impact of the sampling parameter ρ in our proposed UA strategy and HF strategy; and (iii) the convergence property of both the federated and unfederated models.

In implementation, we use advanced object-oriented programming and multithreading methods in Java to simulate the federation system. Specifically, each user u (i.e., client) is represented by an object of class *Client*, and the server is represented by an object of class *Server*.

Datasets and Evaluation Metrics

We adopt two public MovieLens datasets⁶ from GroupLens. (<https://grouplens.org/>) The first dataset is extracted from MovieLens 100K, which consists of 100 000 ratings from 943 users for 1682 movies. The second dataset is extracted from MovieLens 1M, which consists of 1 000 209 ratings from 6040 users for 3952 movies.

First, we divide each dataset into five equal parts. Second, we take four parts as the training data and the left one as the test data. We repeat the abovementioned procedure for five times and have five copies of training data and test data for each dataset. Finally, we report the averaged recommendation performance on those five copies of test data. (For reproducibility, we have made the data, code and scripts used in the experiments publicly available at <http://csse.szu.edu.cn/staff/panwk/publications/FedRec/>)

We adopt mean absolute error (MAE) and root mean square error (RMSE) as the evaluation metrics, which are commonly used in the community of recommender systems.

In order to compare the performance between a federated model and an unfederated one, we follow¹ and compute the mean difference (MD):

$$\text{MD} = \left| \frac{M_{\text{F}}(\text{model}) - M_{\text{UF}}(\text{model})}{M_{\text{UF}}(\text{model})} \right| \times 100\% \quad (10)$$

where $M_{\text{F}}(\text{model})$ and $M_{\text{UF}}(\text{model})$ denote the mean of the performance of a *model* under a federated framework and an unfederated framework, respectively.

Besides, in order to study the equivalence of the conversion in our proposed federated framework, we introduce a metric called the standard deviation range (STDR),

$$\text{STDR} = \left| \frac{\text{STD}_{\text{F}}(\text{model}) + \text{STD}_{\text{UF}}(\text{model})}{M_{\text{UF}}(\text{model})} \right| \times 100\% \quad (11)$$

where $\text{STD}_{\text{F}}(\text{model})$ and $\text{STD}_{\text{UF}}(\text{model})$ denote the metric of standard deviation of a *model* under a federated framework and an unfederated framework, respectively.

We can see that MD stands for the real difference and STDR represents the maximum deviation that can be caused by the instability of the recommendation model itself. If the former is smaller than the latter, it is reasonable to say that the federated framework can convert an unfederated model to a federated one equivalently though there may still be some instability of the algorithm itself.

Baselines and Parameter Settings

For the basic recommendation models in our experiments, we use PMF¹⁴ in the batch style, as well as PMF¹⁴ and SVD++¹¹ in the stochastic style. We denote them as PMF(B), PMF(S), and SVD++(S), respectively. We choose more models in the stochastic style because stochastic algorithms are more commonly used in both academia and industry. Notice that we do

TABLE 2. Recommendation performance of the federated versions (via our FedRec with $\rho = 0$) and the unfederated versions of PMF(B), PMF(S) and SVD++(S) on MovieLens 100K and MovieLens 1M.

Style	Models	Framework	MAE			RMSE		
			Value	MD	STDR	Value	MD	STDR
MovieLens 100K								
Batch	PMF	Unfederated	0.7418±0.0046	0.00%	1.26%	0.9424±0.0059	0.01%	1.31%
		Federated	0.7418±0.0048			0.9424±0.0064		
Stochastic	PMF	Unfederated	0.7497±0.0043	0.01%	1.13%	0.9551±0.0054	0.02%	1.09%
		Federated	0.7498±0.0042			0.9553±0.0051		
	SVD++	Unfederated	0.7215±0.0034	0.08%	0.96%	0.9228±0.0049	0.05%	1.07%
		Federated	0.7221±0.0035			0.9233±0.0050		
MovieLens 1M								
Batch	PMF	Unfederated	0.7195±0.0013	0.03%	0.35%	0.9108±0.0014	0.03%	0.32%
		Federated	0.7193±0.0012			0.9106±0.0015		
Stochastic	PMF	Unfederated	0.6829±0.0019	0.01%	0.46%	0.8701±0.0021	0.01%	0.41%
		Federated	0.6829±0.0012			0.8700±0.0015		
	SVD++	Unfederated	0.6619±0.0010	0.01%	0.33%	0.8493±0.0013	0.01%	0.31%
		Federated	0.6620±0.0012			0.8493±0.0014		

not include FCF¹ in the empirical studies, because it is designed for item ranking with implicit feedback, instead of rating prediction with explicit feedback, as studied in this article.

For parameter configurations, we set the number of latent features $d = 20$ and the iteration number $T = 100$ for all the models, and fix the learning rate for stochastic models as $\gamma = 0.01$ and that for the batch model as $\gamma = 0.8$. For each unfederated model, we choose the best value of the tradeoff parameter of the regularization term λ from $\{0.1, 0.01, 0.001\}$. For each federated model, we use the same value of λ with that of the corresponding unfederated model. For models with different values of the sampling parameter $\rho \in \{0, 1, 2, 3\}$, we choose the best value of the iteration number T_{predict} for starting filling the sampled unrated items via (6) and the iteration number T_{local} for locally training U_u , both from $\{5, 10, 15\}$. All the hyperparameters are searched according to the MAE performance on the first copy of each dataset.

Results

Generality of FedRec

We report the recommendation performance of PMF(B), PMF(S), and SVD++(S) implemented in a federated manner (via our FedRec) and in an unfederated manner in Table 2, from which we can have the following observations: 1) the value of MD is lower than the corresponding value of STDR on both evaluation metrics of MAE and RMSE for all the models, which shows that our FedRec is a generic framework for federated recommendation and is able to convert an unfederated model to a federated one equivalently in spite of the instability of the model itself; and 2) the overall

relative performance among the models are PMF(B), PMF(S) < SVD++(S), which is consistent in previous studies.¹⁶

Impact of Sampling Parameter

Furthermore, we study the effectiveness of our UA and HF strategies with different values of the sampling parameter $\rho \in \{0, 1, 2, 3\}$. We report the recommendation performance in Figure 2, and can have the following observations: 1) the recommendation performance decreases (i.e., the value of RMSE increases) with a larger value of ρ when using the UA strategy, which is expected because it will introduce some noise to the data; 2) the recommendation performance decreases or increases very slightly with a larger value of ρ when using the HF strategy, which means that we address the privacy issue well without sacrificing the recommendation accuracy much; and 3) the overall relative performance of PMF(B), PMF(S), and SVD++(S) are similar to that in Table 2.

Convergence of FedRec

We also study the convergence of all the federated versions and the unfederated versions of PMF(B), PMF(S), and SVD++(S) in Figure 3, including 1) the original unfederated versions of PMF(B), PMF(S), and SVD++(S); 2) the federated versions (with $\rho = 0$) of PMF(B), PMF(S), and SVD++(S); and 3) the federated versions (with $\rho \in \{1, 2, 3\}$ and the HF strategy) of PMF(B), PMF(S), and SVD++(S). We can see that all the methods have almost the same tendency of convergence, i.e., they converge with about 20 iterations, which shows that our FedRec does not have a significant affect on the convergence.

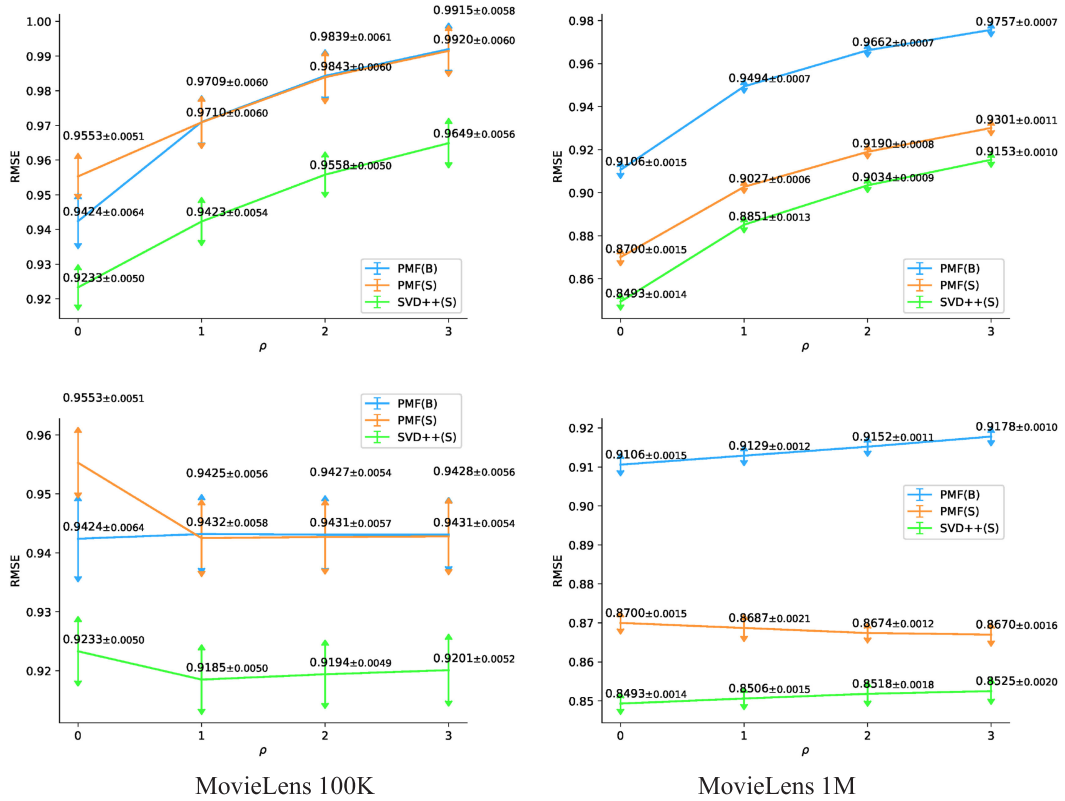


FIGURE 2. Recommendation performance of the federated versions (via our FedRec) of PMF(B), PMF(S), and SVD++(S) with different values of ρ when using the UA strategy (top) and the HF strategy (bottom) on MovieLens 100K and MovieLens 1M.

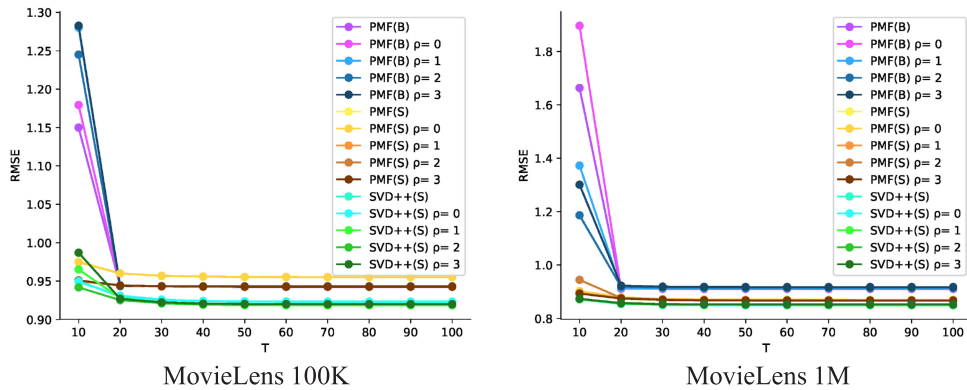


FIGURE 3. Recommendation performance of (i) the original unfederated versions of PMF(B), PMF(S), and SVD++(S), (ii) the federated versions (with $\rho = 0$) of PMF(B), PMF(S), and SVD++(S), and (iii) the federated versions (with $\rho \in \{1, 2, 3\}$ and the HF strategy) of PMF(B), PMF(S), and SVD++(S), with different iteration numbers on MovieLens 100K and MovieLens 1M.

CONCLUSIONS AND FUTURE WORK

In this article, we follow a recent work on FCF for item ranking with implicit feedback in recommender systems,¹ and propose a novel and generic federated

recommendation (FedRec) framework for rating prediction with explicit feedback. Our FedRec aims to federate some traditional rating-oriented recommendation models without sacrificing users' privacy, i.e., each user's rated items. Specifically, we federate some

factorization-based models both in the batch style and in the stochastic style to showcase the generality of our FedRec. More importantly, we propose two simple but effective strategies (i.e., UA and HF) for virtual rating estimation, and introduce a sampling parameter ρ to achieve a balance between the computational/communication efficiency and the protection of users' privacy. Empirical studies on two public datasets show the effectiveness of our proposed framework for converting an unfederated model to a federated one, as well as the usefulness of the two strategies for recommendation accuracy and privacy protection.

For future works, we are interested in federating more recommendation models so as to further generalize our proposed framework. Besides, we are also interested in studying novel federated transfer learning methods for cross-domain recommendation.

ACKNOWLEDGMENTS

The authors would like to thank the editors and reviewers for their constructive and expert comments. This work was supported by the National Natural Science Foundation of China under Grant 61872249, Grant 61836005, and Grant 61672358. G. Lin and F. Liang are co-first authors.

REFERENCES

1. M. Ammad-ud-din *et al.*, "Federated collaborative filtering for privacy-preserving personalized recommendation system," *CoRR*, abs/1901.09888, 2019. [Online]. Available: <https://arxiv.org/abs/1901.09888>
2. D. R. B. McMahan, "Federated learning: Collaborative machine learning without centralized training data," 2017. [Online]. Available: <https://ai.googleblog.com/2017/04/federated-learning-collaborative.html>, Accessed: Apr. 27, 2019.
3. F. Chen, Z. Dong, Z. Li, and X. He, "Federated meta-learning for recommendation," *CoRR*, abs/1802.07876, 2018. [Online]. Available: <https://arxiv.org/abs/1802.07876v1>
4. EU. "Regulation (EU) 2016/679 of the European parliament and of the council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation) (text with EEA relevance)," 2016. [Online]. Available: <https://eur-lex.europa.eu/eli/reg/2016/679/oj>, Accessed: Apr. 27, 2019.
5. D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, no. 12, pp. 61–70, 1992.
6. F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interactive Intell. Syst.*, vol. 5, no. 4, 2016, Art. no. 19.
7. I. Hegedüs, G. Danner, and M. Jelasity, "Decentralized recommendation based on matrix factorization: A comparison of gossip and federated learning," in *Proc. Int. Workshops ECML PKDD*, volume 1167 *Commun. Comput. Inf. Sci.*, Springer, 2020 pp. 317–332.
8. Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. 8th IEEE Int. Conf. Data Mining*, 2008, vol. 8, pp. 263–272.
9. A. Jalalirad, M. Scavuzzo, C. Capota, and M. R. Sprague, "A simple and efficient federated recommender system," in *Proc. 6th IEEE/ACM Int. Conf. Big Data Comput., Appl. Technologies*, 2019, pp. 53–58.
10. J. Konecný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *CoRR*, abs/1610.02527, 2016. [Online]. Available: <https://arxiv.org/abs/1610.02527>
11. Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 426–434.
12. Y. Liu, T. Chen, and Q. Yang, "Secure federated transfer learning," *CoRR*, abs/1812.03337, 2018. [Online]. Available: <https://arxiv.org/abs/1812.03337v1>
13. H. B. McMahan, E. Moore, D. Ramage, and B. A. Y. Arcas, "Federated learning of deep networks using model averaging," *CoRR*, abs/1602.05629, 2016. [Online]. Available: <https://arxiv.org/abs/1602.05629v1>
14. A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. 21st Int. Conf. Neural Inf. Process. Syst.*, 2007, pp. 1257–1264.
15. W. Pan and Z. Ming, "Interaction-rich transfer learning for collaborative filtering with heterogeneous user feedback," *IEEE Intell. Syst.*, vol. 29, no. 6, pp. 48–54, Nov.–Dec. 2014.
16. W. Pan and Z. Ming, "Collaborative recommendation with multiclass preference context," *IEEE Intell. Syst.*, vol. 32, no. 2, pp. 45–51, Mar.–Apr. 2017.
17. F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*, 2nd ed. New York, NY, USA: Springer, 2015.
18. Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, 2019, Art. no. 19.

19. H. H. Zhuo, W. Feng, Q. Xu, Q. Yang, and Y. Lin, "Federated reinforcement learning," *CoRR*, abs/1901.08277, 2019. [Online]. Available: <https://arxiv.org/abs/1901.08277v1>


GUANYU LIN is currently an undergraduate student with the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include collaborative recommendation, federated learning, and counterfactual learning. He received the B. S. degree in computer science and technology from Shenzhen University. Contact him at linguanyu20161@email.szu.edu.cn.

FENG LIANG is currently working toward the master's degree with the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include collaborative recommendation, federated learning, and transfer learning. He received the B.S. degree in software engineering from the Guilin

University of Electronic Technology, Guilin, China. Contact him at liangfeng2018@email.szu.edu.cn.

WEIKE PAN is currently an Associate Professor with the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include transfer learning, recommender systems, and statistical machine learning. He received the Ph.D. degree in computer science and engineering from the Hong Kong University of Science and Technology, Hong Kong, China. He is the corresponding author. Contact him at panweike@szu.edu.cn.

ZHONG MING is currently a Professor with the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include software engineering and Web intelligence. He received the Ph.D. degree in computer science and technology from Sun Yat-Sen University, Guangzhou, China. He is the corresponding author. Contact him at mingz@szu.edu.cn.



CG&A
www.computer.org/cga

IEEE Computer Graphics and Applications bridges the theory and practice of computer graphics. Subscribe to *CG&A* and

- stay current on the latest tools and applications and gain invaluable practical and research knowledge,
- discover cutting-edge applications and learn more about the latest techniques, and
- benefit from *CG&A*'s active and connected editorial board.

75 YEARS
IEEE COMPUTER SOCIETY

IEEE