FedRec: Federated Recommendation with Explicit Feedback

Guanyu Lin$^{\#}$, Feng Liang$^{\#}$, Weike Pan* , Zhong Ming*

National Engineering Laboratory for Big Data System Computing Technology
College of Computer Science and Software Engineering
Shenzhen University, Shenzhen 518060, China
{linguanyu20161, liangfeng2018}@email.szu.edu.cn, {panweike, mingz}@szu.edu.cn

Abstract

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.

Keywords: Federated Machine Learning, Federated Recommendation, Explicit Feedback

1 Introduction

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) [Goldberg *et al.*, 1992; Koren, 2008] is a very popular technique. However, European Union (EU) has recently enacted General Data Protection Regulation (GDPR) [EU, 2016] 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 [Yang *et al.*, 2019]. 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 [Yang *et al.*, 2019], CF is unavoidably facing a new challenge about preference modeling and privacy protection.

Fortunately, firstly proposed by Google, federated machine learning (FML) [Konecný *et al.*, 2016; McMahan *et al.*, 2016; Brendan McMahan, 2017] 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 [Zhuo *et al.*, 2019], transfer learning [Liu *et al.*, 2018] and meta learning [Chen *et al.*, 2018]. 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 [Ammad-ud-din *et al.*, 2019], 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 un-interacted (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 learning [Jalalirad *et al.*, 2019] or gossip learning [Hegedüs *et al.*, 2020] for rating prediction and low-rank matrix decomposition, respectively. However, in these 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 paper, we follow [Ammad-ud-din *et al.*, 2019] 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 $j \in \mathcal{I} \backslash \mathcal{I}_u$ 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 [Ammad-ud-din *et al.*, 2019] and the traditional recommendation works [Ricci *et al.*, 2015], which requires us to design a rating prediction method in a different *collaborative* manner. In particular, the techniques in [Ammad-ud-din *et al.*, 2019] can not 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 and hybrid filling, to address this issue, and then propose a generic factorization-based federated recommendation framework called FedRec. Under the framework of our FedRec, we federate PMF [Mnih and Salakhutdinov, 2007] in batch style, as well as PMF and SVD++ [Koren, 2008] in stochastic style to show the generality and effectiveness of our solution, where 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 [Pan and Ming, 2014].

Table 1: Some notations and explanations used in the paper.

n	number of users
m	number of items
$\mathfrak{R} = \{1, \ldots, 5\}$	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
$\mathcal{I}'_u$, $\lvert\mathcal{I}'_u\rvert = \rho\lvert\mathcal{I}_u\rvert$	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
$U_{u\cdot} \in \mathbb{R}^{1\times d}$	user-specific latent feature vector
$V_{i\cdot}, W_{i'\cdot} \in \mathbb{R}^{1\times d}$	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

2 Related Work

2.1 Factorization-based Methods for Rating Prediction with Explicit Feedback

In probabilistic matrix factorization (PMF) [Mnih and Salakhutdinov, 2007], the rating of a user $u \in \{1, 2, \ldots, n\}$ to an item $i \in \{1, 2, \ldots, m\}$ is predicted as the inner product of two learned vectors,

$$\hat{r}_{ui} = U_{u\cdot} V_{i\cdot}^T, \tag{1}$$

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

In SVD++ [Koren, 2008], the rating of a user u to an item i is estimated by exploiting the other rated items by user u,

$$\hat{r}_{ui} = U_{u\cdot} V_{i\cdot}^T + \frac{1}{\sqrt{|\mathcal{I}_u \backslash \{i\}|}} \sum_{i' \in \mathcal{I}_u \backslash \{i\}} W_{i'\cdot} V_{i\cdot}^T, \tag{2}$$

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

2.2 Federated Collaborative Filtering for Item Ranking with Implicit Feedback

In federated collaborative filtering (FCF) [Ammad-ud-din *et al.*, 2019], 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,

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

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 $i \in \mathcal{I} \backslash \mathcal{I}_u$ as shown in Eq.(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 paper, we usually do not model the unobserved records and will thus have,

$$\nabla V_{\text{EF}}(u, i) = y_{ui}(U_{u\cdot} V_{i\cdot}^T - r_{ui}) U_{u\cdot}, \tag{4}$$

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 negative feedback as that in [Ammad-ud-din *et al.*, 2019], we will bias the model training towards lower predicted scores. In a summary, we can not directly apply FCF [Ammad-ud-din *et al.*, 2019] to the problem of rating prediction with explicit feedback studied in this paper, which motivates us to design a new federated solution.

3 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., user averaging (UA) and hybrid filling (HF). Specifically, we first randomly sample some unrated items $\mathcal{I}'_u \subseteq \mathcal{I} \backslash \mathcal{I}_u$ for each user u, and then assign a virtual rating r'_{ui} to each item $i \in \mathcal{I}'_u$,

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

$$r'_{ui} = \hat{r}_{ui}, \tag{6}$$

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. Firstly, 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. Secondly, the way of sampling some unrated items instead of taking all the unrated items in [Ammad-ud-din *et al.*, 2019] will not significantly increase the computational and communication cost. Thirdly, assigning a virtual rating via an average score or a predicted score instead of a negative score in [Ammad-ud-din *et al.*, 2019] 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 FCF [Ammad-ud-din *et al.*, 2019] 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 FCF [Ammad-ud-din *et al.*, 2019] and explicit feedback in our FedRec.

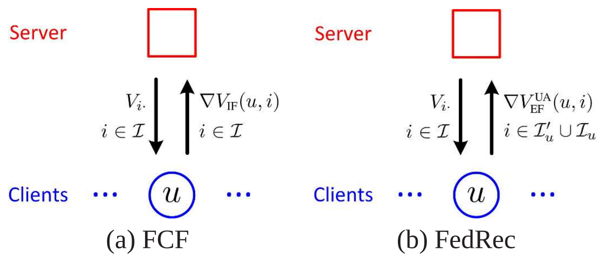

Figure 1: Illustration of the interactions between the server and each client in FCF for item ranking with implicit feedback (left) and our FedRec using the user average (UA) strategy for rating prediction with explicit feedback (right).

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

3.1 FedRec in Batch Style

In this subsection, we describe our FedRec for a basic factorization-based model, i.e., probabilistic matrix factorization (PMF), in 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., $V_{i\cdot}, i = 1, 2, \ldots, m$, with small random values.
- Each client u downloads the item-specific latent feature vectors, i.e., $V_{i\cdot}$, $i = 1, 2, \ldots, m$, 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., $\nabla V^{\text{UA}}_{\text{EF}}(u, i)$, $i \in \mathcal{I}'_u \cup \mathcal{I}_u$, to the server.
- The server updates the item-specific latent feature vectors with $\nabla V^{\text{UA}}_{\text{EF}}(u, i)$ received from the clients.

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 $V_{i\cdot}, i = 1, \ldots, m$ from the server, the user u can calculate the gradient $\nabla U_{u\cdot}$ without referring to other users' data. After that, the user u can update the user-specific latent feature vector $U_{u\cdot}$ locally. That is to say, without sharing the original rating data among users, each user u can update the the model parameter $U_{u\cdot}$.

In particular, we randomly sample some items $\mathcal{I}'_u$ from $\mathcal{I} \backslash \mathcal{I}_u$ 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 user averaging strategy or the hybrid filling strategy, we have the gradient $\nabla U_{u\cdot}$ as follows,

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

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

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

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

which are then uploaded to the server. Notice that the server can not identify which items are from $\mathcal{I}_u$ from the set of the uploaded gradients, i.e., $\nabla V^{\text{UA}}_{\text{EF}}(u, i)$, $i \in \mathcal{I}'_u \cup \mathcal{I}_u$, 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., $\nabla V^{\text{UA}}_{\text{EF}}(u,i)$, $i \in \mathcal{I}'_u \cup \mathcal{I}_u$, $u = 1,2,\ldots,n$, it can then calculate the gradient of item i,

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

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

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

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

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

Algorithm 2 ClientBatch($V_{i\cdot}, i = 1,2,\ldots,m$; u; t), i.e., the algorithm of batch FedRec for PMF in the client.

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

Algorithm 3 ClientFilling($V_{i\cdot}, i = 1,2,\ldots,m$; $U_{u\cdot}$; u; t), i.e., the algorithm of assigning ratings to the sampled unrated items in the client.

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

3.2 FedRec in Stochastic Style

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

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

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

Algorithm 5 ClientStochastic($V_{i\cdot}, i = 1,2,\ldots,m$; u; t), i.e., the algorithm of stochastic FedRec for PMF in the client.

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

3.3 Complexity Analysis

The Complexity of FedRec in Batch Style

For each client u, the worst computational complexity is $T_{\text{local}} \times [(|\mathcal{I}_u| + |\mathcal{I}'_u|) \times d + d] + |\mathcal{I}'_u| \times d + (|\mathcal{I}_u| + |\mathcal{I}'_u|) \times d + d + (|\mathcal{I}_u| + |\mathcal{I}'_u|) \times d$ 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 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 $\nabla V^{\text{UA}}_{\text{EF}}(u,i)$ that contains $4 \times d \times |\mathcal{I}_u| \times (1 + \rho)$ bytes information from each client u by invoking the function ClientBatch($V_{i\cdot}, i = 1,2,\ldots,m$; 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.

The Complexity of FedRec in Stochastic Style

Similarly, for algorithms in 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 $4 \times d \times [|\mathcal{R}| \times (1+\rho) + m \times n]$, respectively, for each client u and the server at each iteration t. But the computational complexities in stochastic style are different from that in 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 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 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.

The Comparison of Complexity with FCF

From FCF and [Hu *et al.*, 2008], 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 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 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 batch style and stochastic style are comparable with that of FCF, and the computational complexities of the server of our FedRec in both batch and stochastic style are smaller than 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 $\nabla V_{\text{IF}}(u,i)$ for $i \in \mathcal{I}$. In our FedRec both in 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 hybrid filling strategy in clients are comparable (i.e., both are linear with $|\mathcal{I}_u|$).

3.4 Discussions

Although we have described and analysed our FedRec in the context of a basic matrix factorization model PMF [Mnih and Salakhutdinov, 2007], our proposed framework and learning algorithms can be applied to more advanced factorization-based models such as SVD++ in Eq.(2), which is included in our empirical studies.

4 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 user averaging strategy and hybrid filling strategy; and (iii) the convergence property of both the federated and unfederated models.

In implementation, we use advanced object-oriented programming and multi-threading 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*.

4.1 Datasets and Evaluation Metrics

We adopt two public MovieLens datasets [Harper and Konstan, 2016] from GroupLens[1]. The first dataset is extracted from MovieLens 100K, which consists of 100,000 ratings from 943 users for 1,682 movies. The second dataset is extracted from MovieLens 1M, which consists of 1,000,209 ratings from 6,040 users for 3,952 movies.

Firstly, we divide each dataset into five equal parts. Secondly, we take four parts as the training data and the left one as the test data. We repeat the above 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[2].

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 follow [Ammad-ud-din *et al.*, 2019] and compute the mean difference (MD),

$$\text{MD} = |\frac{\text{M}_{\text{F}}(model) - \text{M}_{\text{UF}}(model)}{\text{M}_{\text{UF}}(model)}| \times 100\%, \qquad (10)$$

where $\text{M}_{\text{F}}(model)$ and $\text{M}_{\text{UF}}(model)$ 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 standard deviation range (STDR),

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

[1]https://grouplens.org/

[2]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/

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		

where $\text{STD}_{\text{F}}(model)$ and $\text{STD}_{\text{UF}}(model)$ 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.

4.2 Baselines and Parameter Settings

For the basic recommendation models in our experiments, we use PMF [Mnih and Salakhutdinov, 2007] in batch style, as well as PMF [Mnih and Salakhutdinov, 2007] and SVD++ [Koren, 2008] in stochastic style. We denote them as PMF(B), PMF(S) and SVD++(S), respectively. We choose more models in stochastic style because stochastic algorithms are more commonly used in both academia and industry. Notice that we do not include FCF [Ammad-ud-din *et al.*, 2019] 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 paper.

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 Eq.(6) and the iteration number T_{local} for locally training $U_{u\cdot}$ both from $\{5, 10, 15\}$. All the hyper parameters are searched according to the MAE performance on the first copy of each dataset.

4.3 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: (i) 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 (ii) the overall relative performance among the models are PMF(B), PMF(S) < SVD++(S), which is consistent in previous studies [Pan and Ming, 2017].

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

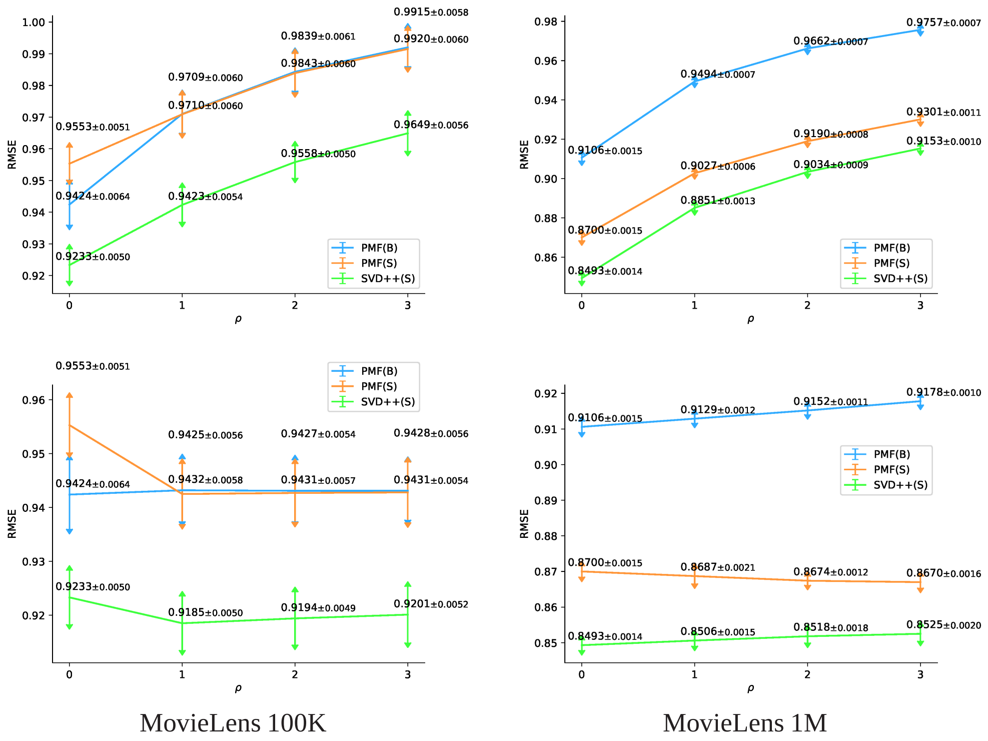

Figure 2: Recommendation performance of the federated versions (via our FedRec) of PMF(B), PMF(S) and SVD++(S) with different values of ρ when using the user averaging strategy (top) and the hybrid filling strategy (bottom) on MovieLens 100K and MovieLens 1M.

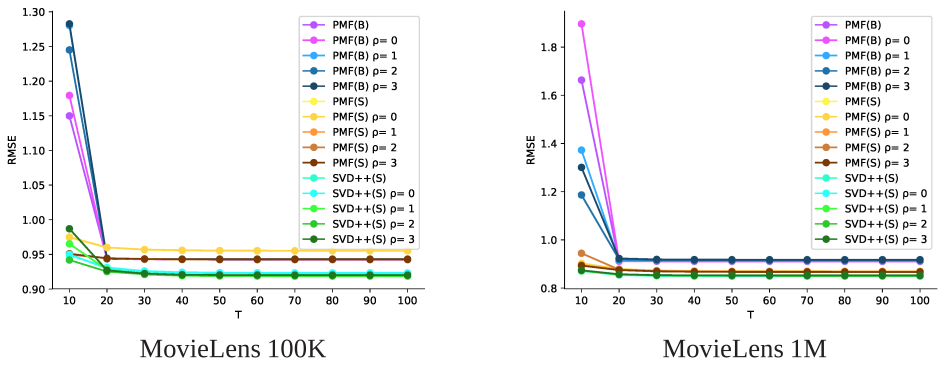

Figure 3: Recommendation performance of (i) the original unfederated versions of PMF(B), PMF(S) and SVD++(S), (ii) the federated versions (with $\rho = 0$) of PMF(B), PMF(S) and SVD++(S), and (iii) the federated versions (with $\rho \in \{1, 2, 3\}$ and the hybrid filling strategy) of PMF(B), PMF(S)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

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.

5 Conclusions and Future Work

In this paper, we follow a recent work on federated collaborative filtering (FCF) for item ranking with implicit feedback in recommender systems [Ammad-ud-din *et al.*, 2019], 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 batch style and in stochastic style to showcase the generality of our FedRec. More importantly, we propose two simple but effective strategies (i.e., user averaging and hybrid filling) 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.

6 Acknowledgment

We thank the editors and reviewers for their constructive and expert comments. We thank the support of National Natural Science Foundation of China Nos. 61872249, 61836005 and 61672358. Guanyu Lin and Feng Liang are co-first authors, and Weike Pan and Zhong Ming are co-corresponding authors for this work.

7 About the Authors

Guanyu Lin, co-first author, is an undergraduate student in the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering at Shenzhen University, China. His research interests include collaborative recommendation, federated learning, and counterfactual learning. Lin has recently received a BS in computer science and technology from Shenzhen University, China. Contact him at linguanyu20161@email.szu.edu.cn.

Feng Liang, co-first author, is a master's student in the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, China. His research interests include collaborative recommendation, federated learning, and transfer learning. Liang received a BS in software engineering from the Guilin University of Electronic Technology, China. Contact him at liangfeng2018@email.szu.edu.cn.

Weike Pan, co-corresponding author, is an associate professor in the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering at Shenzhen University, China. His research interests include transfer learning, recommender systems, and statistical machine learning. Pan received a PhD in computer science and engineering from the Hong Kong University of Science and Technology, China. Contact him at panweike@szu.edu.cn.

Zhong Ming, co-corresponding author, is a professor in the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering at Shenzhen University, China. His research interests include software engineering and Web intelligence. Ming received a PhD in computer science and technology from Sun Yat-Sen University, China. Contact him at mingz@szu.edu.cn.

References

[Ammad-ud-din *et al.*, 2019] Muhammad Ammad-ud-din, Elena Ivannikova, Suleiman A. Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. Federated collaborative filtering for privacy-preserving personalized recommendation system. *CoRR*, abs/1901.09888, 2019.

[Brendan McMahan, 2017] Daniel Ramage Brendan McMahan. Federated learning: Collaborative machine learning without centralized training data. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, 2017. Accessed April 27, 2019.

[Chen *et al.*, 2018] Fei Chen, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning for recommendation. *CoRR*, abs/1802.07876, 2018.

[EU, 2016] 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). https://eur-lex.europa.eu/eli/reg/2016/679/oj, 2016. Accessed April 27, 2019.

[Goldberg *et al.*, 1992] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[Harper and Konstan, 2016] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):19, 2016.

[Hegedüs *et al.*, 2020] István Hegedüs, Gábor Danner, and Márk Jelasity. Decentralized recommendation based on matrix factorization: A comparison of gossip and federated learning. In *International Workshops of ECML PKDD 2019*, volume 1167 of *Communications in Computer and Information Science*, pages 317–332. Springer, 2020.

[Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, volume 8, pages 263–272, 2008.

[Jalalirad *et al.*, 2019] Amir Jalalirad, Marco Scavuzzo, Catalin Capota, and Michael R. Sprague. A simple and efficient federated recommender system. In *Proceedings of the 6th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*, BDCAT, pages 53–58, 2019.

[Konecný *et al.*, 2016] Jakub Konecný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *CoRR*, abs/1610.02527, 2016.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434, 2008.

[Liu *et al.*, 2018] Yang Liu, Tianjian Chen, and Qiang Yang. Secure federated transfer learning. *CoRR*, abs/1812.03337, 2018.

[McMahan *et al.*, 2016] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

[Mnih and Salakhutdinov, 2007] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, pages 1257–1264, 2007.

[Pan and Ming, 2014] Weike Pan and Zhong Ming. Interaction-rich transfer learning for collaborative filtering with heterogeneous user feedback. *IEEE Intelligent Systems*, 29(6):48–54, 2014.

[Pan and Ming, 2017] Weike Pan and Zhong Ming. Collaborative recommendation with multiclass preference context. *IEEE Intelligent Systems*, 32(2):45–51, 2017.

[Ricci *et al.*, 2015] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems Handbook*. Springer, 2nd edition, 2015.

[Yang *et al.*, 2019] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):12:1–12:19, 2019.

[Zhuo *et al.*, 2019] Hankz Hankui Zhuo, Wenfeng Feng, Qian Xu, Qiang Yang, and Yufeng Lin. Federated reinforcement learning. *CoRR*, abs/1901.08277, 2019.