FedRec: Federated Recommendation with Explicit Feedback

Guanyu Lin[1,2#], Feng Liang[1,2#], Weike Pan[1,2*] and Zhong Ming[1,2*]

{linguanyu20161, liangfeng2018}@email.szu.edu.cn, {panweike, mingz}@szu.edu.cn

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

[2]College of Computer Science and Software Engineering
Shenzhen University, Shenzhen, P.R. China

Notations (1/2)

Table: Some notations and explanations.

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

Notations (2/2)

Table: Some notations and explanations (cont.).

$d \in \mathbb{R}$	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

Problem Definition

- Privacy-aware rating prediction with explicit feedback
  - Input: Some rating records $\mathcal{R}_u = \{(u, i, r_{ui}); i \in \mathcal{I}_u\}$, where each user u has rated a set of items $\mathcal{I}_u$
  - Goal: Predict the rating of 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., $\mathcal{R}_u$), which is very different from traditional collaborative filtering

Probabilistic Matrix Factorization (PMF)

In PMF [Mnih and Salakhutdinov, 2007], the rating of user u to item i 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 latent feature vectors of user u and item i, respectively.

SVD++

In SVD++ [Koren, 2008], the rating of user u to 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.

Federated Collaborative Filtering (FCF)

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

Federated Collaborative Filtering (FCF)

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 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 FCF, we will bias the model training towards lower predicted scores.
In a summary, we can not directly apply FCF to the problem of rating prediction with explicit feedback studied in this paper, which also motivates us to design a new federated solution.

Challenges

- The privacy challenge. There may be a leakage of user u's privacy because the items in $\mathcal{I}_u$ may be easily identified by the server.
- The computational and communication challenge. Treating all the un-interacted (user, item) pairs as negative feedback as that in FCF will bias the model training and will also increase the computational and communication cost.

Our Solution: Federated Recommendation (FedRec)

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 $i \in \mathcal{I}'_u$,

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

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

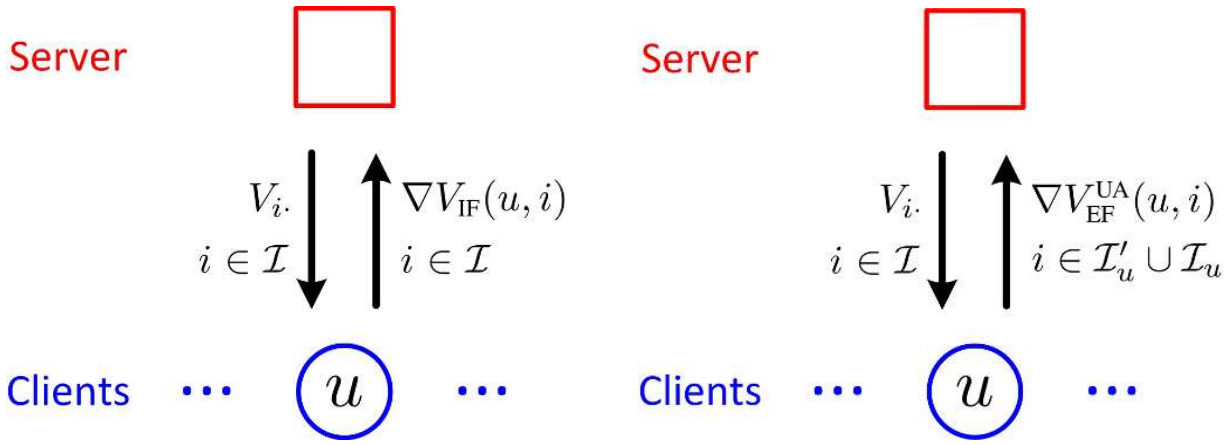
where $\bar{r}_u$ denotes the average rating value of a user u to the rated items in $\mathcal{I}_u$, and $\hat{r}_{ui}$ denotes the predicted rating value of a user u to an unrated item i in $\mathcal{I}'_u$. We show the details of the two strategies in Algorithm 3, with which we can obtain a virtual 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 $\mathcal{R}'_u = \{(u, i, r'_{ui}), i \in \mathcal{I}'_u\}$.

Advantages of FedRec

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 users' privacy in terms of rating behaviors.
- Secondly, the way of sampling some un-interacted items instead of taking all un-interacted items in FCF will not significantly increase the communication and computational cost.
- Thirdly, assigning a virtual rating value via an average score or a predicted score instead of a negative score in FCF will not bias the learning process of model parameters much.

Comparison between FCF and FedRec



(a) FCF (b) FedRec

- 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 and explicit feedback in our FedRec.

FedRec in Bacth Style

The interactions between the server and each client 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., $\nabla V_{\text{EF}}^{\text{UA}}(u, i), i \in \mathcal{I}_u' \cup \mathcal{I}_u$, 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.

Batch FedRec for PMF in the Client

- We randomly sample some items $\mathcal{I}'_u$ from $\mathcal{I}\backslash\mathcal{I}_u$ with $|\mathcal{I}'_u| = \rho|\mathcal{I}_u|$. 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,

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

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

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

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

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

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 of Batch FedRec for PMF in the Server

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

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

Algorithm of Batch FedRec for PMF in the Client (1/2)

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 $\nabla V_{\text{EF}}^{\text{UA}}(u, i)$ via Eq.(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.

Algorithm of Batch FedRec for PMF in the Client (2/2)

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**

FedRec in Stochastic Style

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 $\nabla V_{\text{EF}}^{\text{UA}}(u, i)$ without parameter aggregation.

Algorithm of Stochastic FedRec for PMF in the Server

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

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

Algorithm of Stochastic FedRec for PMF in the Client

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 $\nabla V_{\mathrm{EF}}^{\mathrm{UA}}(u, i)$ via Eq.(8).
7: **end for**
8: Upload $\nabla V_{\mathrm{EF}}^{\mathrm{UA}}(u, i)$ with $i \in \mathcal{I}'_u \cup \mathcal{I}_u$ to the server.

Discussions

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

Datasets

- We adopt two public MovieLens datasets [Harper and Konstan, 2015] 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].

[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/`

Recommendation Models

- Batch style
  - PMF [Mnih and Salakhutdinov, 2007], denoted as PMF(B)
- Stochastic style
  - PMF [Mnih and Salakhutdinov, 2007], denoted as PMF(S)
  - SVD++ [Koren, 2008], denoted as SVD++(S)

We choose more models in stochastic style because stochastic algorithms are more commonly used in both academia and industry.

Parameter Settings

- For parameter configurations, we set the number of latent features as $d = 20$ 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 $\{0.1, 0.01, 0.001\}$ and use the same value of λ on the corresponding federated 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. both from $\{5, 10, 15\}$.

Evaluation Metrics (1/3)

- Mean Absolute Error (MAE)

$$MAE = \sum_{(u,i,r_{ui}) \in \mathcal{R}^{te}} |r_{ui} - \hat{r}_{ui}| / |\mathcal{R}^{te}|$$

- Root Mean Square Error (RMSE)

$$RMSE = \sqrt{\sum_{(u,i,r_{ui}) \in \mathcal{R}^{te}} (r_{ui} - \hat{r}_{ui})^2 / |\mathcal{R}^{te}|}$$

Evaluation Metrics (2/3)

- Mean Difference (MD)

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

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.

- MD stands for the real difference of a *model* under a federated framework and an unfederated framework.

Evaluation Metrics (3/3)

- Standard Deviation Range (STDR)

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

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.

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

Results (1/6)

Table: 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.

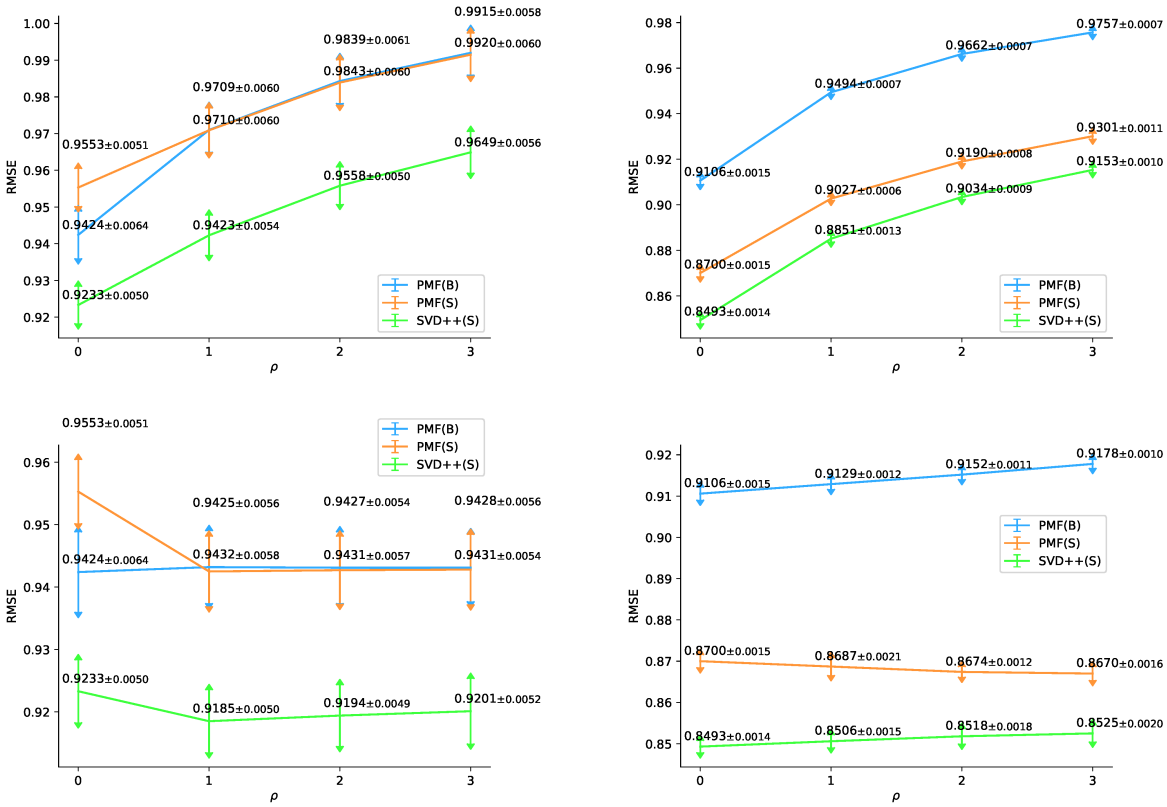
Style	Models	Framework	MAE			RMSE		
			Value	MD	STDR	Value	MD	STDR
MovieLens 100K								
Batch	PMF	Unfederated	0.7418±0.0046	0.00%	1.26%	0.9424±0.0059	0.01%	1.31%
		Federated	0.7418±0.0048			0.9424±0.0064		
Stochastic	PMF	Unfederated	0.7497±0.0043	0.01%	1.13%	0.9551±0.0054	0.02%	1.09%
		Federated	0.7498±0.0042			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		

Results (2/6)

Observations:

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

Results (3/6)



MovieLens 100K MovieLens 1M

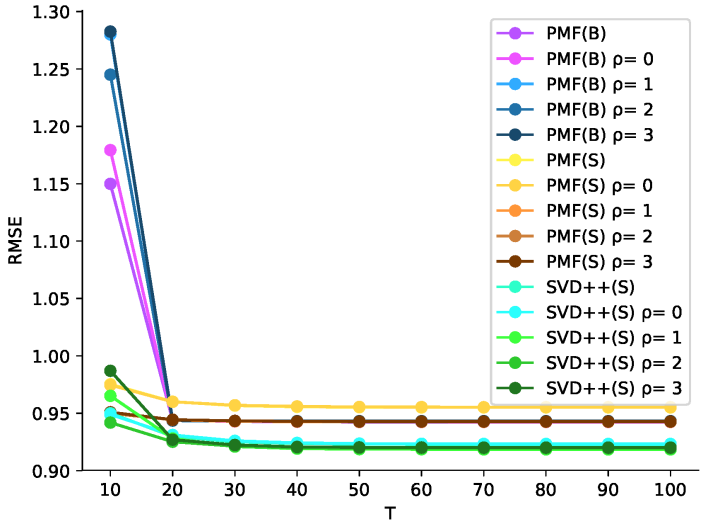
Figure: Recommendation performance of the federated versions (via our FedRec) of PMF(B), PMF(S) and SVD++(S) with different values of ρ when using the user averaging strategy (top) and the hybrid filling strategy (bottom) on MovieLens 100K and MovieLens 1M.

Results (4/6)

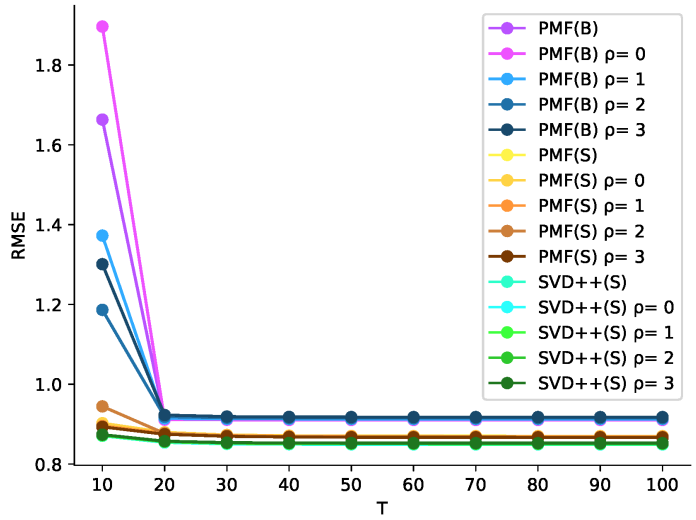
Observation:

- 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;
- 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 . . .

Results (5/6)



MovieLens 100K



MovieLens 1M

Figure: 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.

Results (6/6)

Observations:

- 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.

Conclusions

- We follow a recent work on federated collaborative filtering (FCF) on item ranking with implicit feedback [Ammad-ud-din et al., 2019], and propose a generic federated recommendation (FedRec) framework for rating prediction with explicit feedback.
- 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.
- We federate some factorization-based models in both batch style and stochastic style to showcase the generality of our FedRec.

Future Work

- Horizontal federated learning
  - Design more federated recommendation models so as to further generalize our proposed framework
- Vertical federated learning
  - Develop federated transfer learning methods for cross-domain recommendation

Thank you!

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.

Ammad-ud-din, M., Ivannikova, E., Khan, S. A., Oyomno, W., Fu, Q., Tan, K. E., and Flanagan, A. (2019).

Federated collaborative filtering for privacy-preserving personalized recommendation system.
CoRR, abs/1901.09888.

Harper, F. M. and Konstan, J. A. (2015).

The movielens datasets: History and context.
ACM Transactions on Interactive Intelligent Systems, 5(4):19:1–19:19.

Koren, Y. (2008).

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.

Mnih, A. and Salakhutdinov, R. R. (2007).

Probabilistic matrix factorization.
In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, pages 1257–1264.