FEDERATED COLLABORATIVE FILTERING FOR PRIVACY-PRESERVING PERSONALIZED RECOMMENDATION SYSTEM

A PREPRINT

Muhammad Ammad-ud-din[†], Elena Ivannikova[*], Suleiman A. Khan[*], Were Oyomno , Qiang Fu , Kuan Eeik Tan , and Adrian Flanagan[†]

EU Cloud R&D Center of Helsinki
Huawei Technologies, FINLAND
[†]*Corresponding author:*{*muhammad.amaduddin,adrian.flanagan*}*@huawei.com*

2019/01/30

ABSTRACT

The increasing interest in user privacy is leading to new privacy preserving machine learning paradigms. In the Federated Learning paradigm, a master machine learning model is distributed to user clients, the clients use their locally stored data and model for both inference and calculating model updates. The model updates are sent back and aggregated on the server to update the master model then redistributed to the clients. In this paradigm, the user data never leaves the client, greatly enhancing the user' privacy, in contrast to the traditional paradigm of collecting, storing and processing user data on a backend server beyond the user's control. In this paper we introduce, as far as we are aware, the first federated implementation of a Collaborative Filter. The federated updates to the model are based on a stochastic gradient approach. As a classical case study in machine learning, we explore a personalized recommendation system based on users' implicit feedback and demonstrate the method's applicability to both the MovieLens and an in-house dataset. Empirical validation confirms a collaborative filter can be federated without a loss of accuracy compared to a standard implementation, hence enhancing the user's privacy in a widely used recommender application while maintaining recommender performance.

1 INTRODUCTION

The General Data Protection Regulation (GDPR) https://gdpr-info.eu/ in the EU (Other jurisdictions are considering similar type legislation and also an increased awareness of users of their data privacy) requires users to be both fully informed about, and consent to the collection, storage, and utilization of their personal data by other parties. GDPR changes the default option of users' personal data being harvested, stored and used to requiring explicit opt-in from the user. A default opt-in requires users to explicitly consent to the collection and use of their personal data which many fail to do. Low user opt-in rates means less data to build high performance machine learning models which in general decrease the model's performance.

Personalized recommendation, a classical application of machine learning models [1] suffers badly from unreliable predictions with diverse consequences in many different domains. For example in health care, an in-accurate recommendation on treatment choices may fail the patient while in e-commerce a poor personalization service may recommend products or content of no interest to a user, resulting in a bad user experience. There exists a need to develop new machine learning methods that offer a privacy-by-design solution where we no longer need to collect and store the users' personal data, hence no longer requiring their explicit opt-in to do so, while still making the users' personal data available for building robust models.

[*]equal contribution

Thanks to advances in technology, user devices (e.g. laptops, mobile phones or tablets) have become an integral part of the machine learning process both in terms of being the source of data used in model building and the means of delivering the results of the model (e.g. recommendations) back to the user. These devices may contain user data ranging from very sensitive personal information (e.g. age, gender, location, who like what etc) to the less sensitive (e.g. downloaded applications or videos watched). The standard approach to model building has been to collect user data from these device and transfer it for processing to backend servers. At the servers, the data may be "anonymized" however, anonymization is not foolproof and can still violate user privacy when integrated with other data [2].

To tackle this privacy problem, a Federated Learning (FL) method has been proposed recently [3]. Federated learning distributes the model learning process to the end clients (i.e. user's devices), making it possible to train a global model from user-specific local models, ensuring that the user's private data never leaves the client device enhancing the users privacy. Their proposed FL method is specific to deep learning models with use cases in image recognition and language modeling tasks. Collaborative Filtering CF (we interchangeably use the abbreviation "CF" for both Collaborative Filtering and Collaborative Filter) is one of the most frequently used matrix factorization models to generate personalized recommendations either independently or combined with other types of models [4]. Particularly, CF utilizes user data to build models and generate recommendations for users, in different contexts [5].

In this paper, we introduce the first Federated Collaborative Filter (FCF) method. We show that federation of the collaborative filter is technically challenging and formulate the updates using a stochastic gradient-based approach. Our method aggregates user-specific gradient updates of the model weights from the clients to update the master model. Specifically, we derive a federated version of the widely used CF method for implicit feedback datasets [6]. However, the proposed method is generalizable and can be extended to recommendation scenarios where explicit rating information is available and can also be applied to a more generic class of matrix factorization models. We compare results between the standard and federated models on a simulated dataset as well as on two real datasets, MovieLens and an in-house service dataset. The findings confirm that the performance of the proposed FCF method is statistically similar compared to the CF model, implying no loss of accuracy while simultaneously enhancing user privacy.

1.1 Contribution

Our original contributions in this work are three fold: (1) we formulate the first federated collaborative filter method, (2) we demonstrate the applicability of the proposed federated method for a classical machine learning application in personalized recommendations, and (3) we empirically demonstrate that collaborative filter can be federated without loss of accuracy.

2 RELATED WORK

This work lies at the intersection of three research topics: (i) matrix factorization, (ii) parallel & distributed and, (iii) federated learning. Recently, Alternating Least Squares (ALS) and Stochastic Gradient Descent (SGD) have gained much interest and have become the most popular algorithms for matrix factorization in recommender systems [4]. The ALS algorithm allows learning the latent factor matrices by alternating between updates to one factor matrix while holding the other factor matrix fixed. Each iteration of an update to the latent factor matrices is referred to as an *epoch*. Although the time complexity per epoch is cubic in the number of factors, numerous studies show the ALS is well suited for parallelization [7, 8, 9, 10, 11]. It is not merely a coincidence that ALS is the premiere parallel matrix factorization implementation for CF in Apache Spark (https://spark.apache.org/docs/latest/mllib-collaborative-filtering.html).

Alternatively, SGD has become a widely adopted algorithm to implement matrix factorization for large-scale recommender applications [12, 13]. Contrary to ALS, the solution to latent factor matrices is based on a gradient descent approach, where in each iteration a small step is taken in the direction of the gradient while solving for one factor matrix and fixing the other. Consequently several iterations of gradient descent updates are required to reach the same optimum value. As compared to ALS, the SGD algorithm is efficient and simple, and the time complexity per epoch is linear in the number of factors. However, SGD requires more epochs to produce a stable model with a suitable choice of the learning rate. Furthermore, unlike ALS, parallelization of SGD is challenging, and numerous approaches have been proposed to distribute the learning of matrix factorization [14, 15, 16, 17]. Although these efforts clearly demonstrated the usefulness of parallelizing matrix factorization especially for large-scale recommender system applications, these work only consider the cluster and/or data center settings, and do not assume user data privacy and confidentiality that are inherent to the federated learning paradigm. We adapt the SGD algorithm to federate the seminal CF model [6].

Federated Learning, on the other hand, a distributed learning paradigm essentially assumes user data is not available on central servers and is private and confidential. A prominent direction of research in this domain is based on the weighted averaging of the model parameters [3, 18]. In practice, a master machine learning model is distributed to

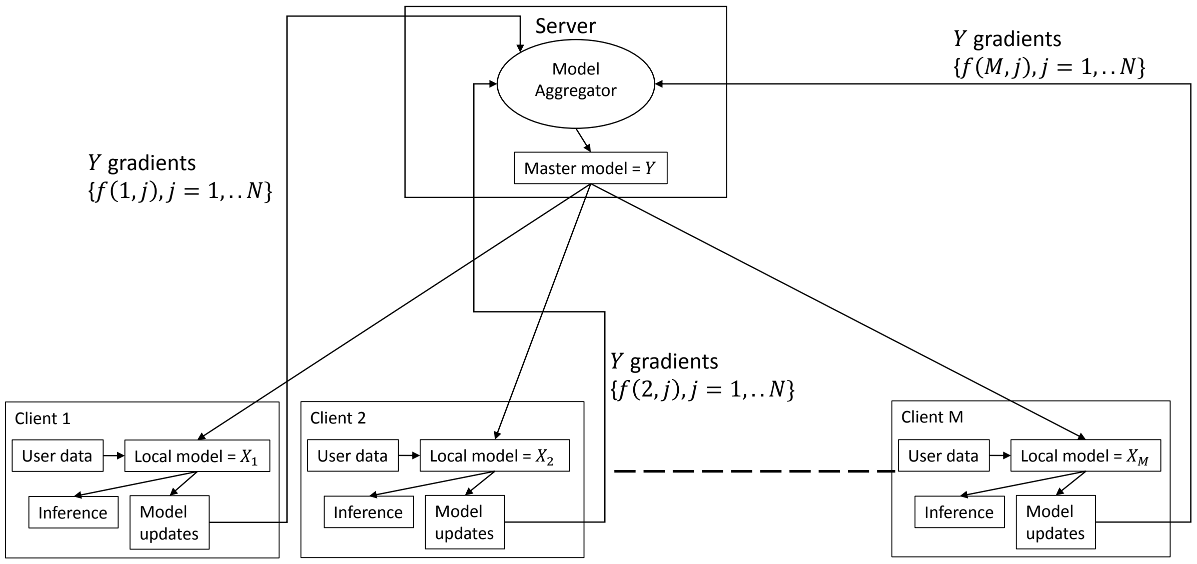

Figure 1: Collaborative Filtering in the Federated Learning Paradigm. The Master Model **Y** (item-factor matrix) is updated on the server and then distributed to the clients. Each user-specific model **X** (user-factor matrix) remains on the local client, and is updated on the client using the local user data and **Y** from the server. The updates through the gradients of **Y** are computed on each client and transmitted to the server where they are aggregated to update the master model **Y**.

user clients. Each client updates the local copy of the model weights using the user's personal data and sends updated weights to the server which uses the weighted average of the clients local model weights to update the master model. This federated averaging approach has recently attracted much attention for deep neural networks, however, the same approach may not be applicable to a wide class of other machine learning models such as matrix factorization. Classical studies based on federated averaging used CNNs to train on benchmark image recognition tasks [3], and LSTM on a language modeling tasks [19, 20]. As a follow-up analysis on federating deep learning models, numerous studies have been proposed addressing the optimization of the communication payloads, noisy, unbalanced [21], non-iid and massively distributed data [22].

Although our work extends the existing distributed matrix factorization and federated learning approaches, primarily we blend SGD-based distributed matrix factorization with the privacy-preserving federated learning. Compared to the commonly used federation of deep learning models, we federate the state-of-the-art CF method which is widely used in personalized recommendation systems.

3 Collaborative Filter

CF models the interactions between a user and a set of items. Based on the learned patterns the model recommends new items the user is expected to interact with. In many applications, the number of users N scales to several millions and the number of items M to thousands. However, typically each user interacts with only a few items, leaving the user-item interaction matrix $\mathbf{R} \in \mathcal{R}^{N \times M}$ to be highly sparse. Low-rank factorization approaches similar to [7], [8] and [9] have been shown to handle sparsity well and are able to scale the retrieval to large datasets.

Formally, the CF model is formulated as a linear combination of low-dimensional (K) latent factor matrices $\mathbf{X} \in \mathcal{R}^{K \times N}$ and $\mathbf{Y} \in \mathcal{R}^{K \times M}$ [4] as

$$\mathbf{R} \sim \mathbf{X}^T \mathbf{Y}. \tag{1}$$

where r_{ui} represents the interaction between user u and item i, for $1 \leq u \leq N, 1 \leq i \leq M$. The interactions r_{ui} are generally derived from explicit feedback such as ratings given by a user $r_{ui} \in (1, \dots, 5)$ [7], or implicit feedback

$r_{uj} \geq 1$ when the user u interacted with the item i and is unspecified otherwise [6]. For instance, the implicit interaction could imply the user has watched a video, or bought an item from an online store, or any similar action. In this work, we consider the case of implicit feedback. The prediction for an unspecified $\hat{r}_{ui}$ is then given by

$$\hat{r}_{ui} = \mathbf{x}_u^T \mathbf{y}_i. \tag{2}$$

The implicit feedback scenario introduces a set of binary variables p_{ui} to indicate the preference of a user u for an item i where

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0, \\ 0 & r_{ui} = 0 \end{cases} \Bigg\} \tag{3}$$

In the implicit case, the value $r_{ui} = 0$ can have multiple interpretations such as the user u is not interested in the item i or maybe the user is not aware of the existence of item i. To account for this uncertainty a confidence parameter is commonly introduced [6] as

$$c_{ui} = 1 + \alpha r_{ui} \tag{4}$$

where $\alpha > 0$. the cost function optimizing across all users u and the items i over the confidence levels c_{ui} is then given as

$$J = \sum_u \sum_j c_{ui}(p_{ui} - \mathbf{x}_u^T \mathbf{y}_i)^2 + \lambda \Big(\sum_u \|\mathbf{x}_u\|^2 + \sum_i \|\mathbf{y}_i\|^2 \Big) \tag{5}$$

with a regularization parameter λ. The differential of J with respect to $\mathbf{x}_u, \forall u$ and $\mathbf{y}_i, \forall i$ is given by

$$\frac{\partial J}{\partial \mathbf{x}_u} = -2 \sum_i \left[c_{ui}(p_{uj} - \mathbf{x}_u^T \mathbf{y}_i) \right] \mathbf{y}_i + 2\lambda \mathbf{x}_u. \tag{6}$$

From [6] the optimal solution of $\mathbf{x}_u = \mathbf{x}_{u^*}$ where $\partial J(\mathbf{x}_{u^*}) / \partial \mathbf{x}_u = 0$ is defined as

$$\mathbf{x}_u^* = \left(\mathbf{Y} \mathbf{C}^u \mathbf{Y}^T + \lambda I \right)^{-1} \mathbf{Y} \mathbf{C}^u p(u), \tag{7}$$

where $\mathbf{C}^u \in \mathcal{R}^{N \times N}$ is a diagonal matrix with $\mathbf{C}_{ii}^u = c_{ui}$ and $p(u) \in \mathcal{R}^{N \times 1}$ contains the p_{ui} values for the user u. Similarly for y_i

$$\frac{\partial J}{\partial \mathbf{y}_i} = -2 \sum_u \left[c_{ui}(p_{ui} - \mathbf{x}_u^T \mathbf{y}_i) \right] \mathbf{x}_u + 2\lambda \mathbf{y}_i, \tag{8}$$

$$\mathbf{y}_i^* = \left(\mathbf{X} \mathbf{C}^i \mathbf{X}^T + \lambda I \right)^{-1} \mathbf{X} \mathbf{C}^i p(i) \tag{9}$$

where $\mathbf{C}^i \in \mathcal{R}^{M \times M}$ is a diagonal matrix with $\mathbf{C}_{uu}^i = c_{ui}$ and $p(i) \in \mathcal{R}^{M \times 1}$ contains the p_{ui} values for item i. Using Eqs. 7 and 9 [6] describe a computationally efficient Alternating Least Squares (ALS) algorithm to find the optimum values of $\mathbf{X}, \mathbf{Y}$. The ALS algorithm solves for the $\mathbf{X}, \mathbf{Y}$ factor matrices by alternating between updates. Multiple *epochs* are carried out to update $\mathbf{X}, \mathbf{Y}$, until a suitable convergence criteria is satisfied.

4 Federated Collaborative Filtering

We now introduce the Federated Collaborative Filter (FCF) Algorithm 1 that extends the CF model to the federated mode. The key idea is to carry out model updates such that the user's private interaction data is not transferred to the server. The FCF distributes parts of the model computation to the clients and is illustrated in Figure 1.1. Specifically, the method defines three core components as below.

1. All the item factor vectors $y_i, i = 1, \ldots, M$ are updated on the server and then distributed to each client u.

2. The user factor vectors $x_u, u \in \{1, \ldots, N\}$ are updated locally on the client u, using the user u's own data and the $y_i, i = 1, \ldots, M$ from the server.
3. The updates through the gradients δy_{ui} are calculated for the item i on each client u and transmitted to the server where the gradients are aggregated to update y_i. This is in contrast to the existing federated learning architectures [23], where the clients directly compute the updates of the parameters y_{ui} which are then aggregated on the server to update the master model.

grad Avg/ epoch

We next discuss the federated model updates in detail.

4.1 Federated User Factor Update

In each update iteration, the server sends the latest item factor vectors $\{\mathbf{y}_i, i = 1, \ldots, M\}$ to each client. The user's own data r_{ui} is used to compute $p(u)$ and $\mathbf{C}^u$ using Eq. 3 and 4, respectively. The method then uses Eq. 7 to update the $\mathbf{x}_u^*$ locally for each client. The updates can be carried out independently for each user u without reference to any other user's data.

4.2 Federated Item Factor Update

To update the item factor vectors using Eq. 9, the user factor vectors $\mathbf{x}_i\ \forall i \in \{1, \ldots, N\}$ and the interaction of the users with items i through the $\mathbf{C}^u$ and $p(u)$ are required. Therefore, the updates of the $\mathbf{y}_i$ cannot be done on the clients and must be carried out on the master server. However, in order to preserve the user's privacy the user-item interactions should remain on the client's own device only, therefore, Eq. 9 cannot be used for computing $\mathbf{y}_i^*$. This is the fundamental problem when federating CF and we present the first solution to address this challenge.

We next adapt and demonstrate a stochastic gradient descent approach to allow for the update of the $\mathbf{y}_i$ vectors on the server, while preserving the user's privacy.

Formally, y_i is updated on the master server as

$$\mathbf{y}_i = \mathbf{y}_i - \gamma \frac{\partial J}{\partial \mathbf{y}_i}, \tag{10}$$

for some gain parameter γ to be determined and $\partial J / \partial \mathbf{y}_i$ as given in Eq. 8. However, Eq. 8 contains a component which is a summation over all users u. We therefore, define $f(u, i)$ as

$$f(u, i) = \left[c_{ui}(p_{ui} - \mathbf{x}_u^T \mathbf{y}_i)\right] \mathbf{x}_u, \tag{11}$$

where $f(u, i)$ is calculated on each client u independently of all the other clients. All the clients then report back the gradient values $f(u, i),\ i = \{1, \ldots, N\}$ to the master server. As a result, Eq. 8 can then be re-written as an aggregation of the client gradients and computed on the master as

$$\frac{\partial J}{\partial \mathbf{y}_i} = -2 \sum_u f(u, i) + 2\lambda \mathbf{y}_i. \tag{12}$$

Finally, the item specific gradient updates in Eq. 12 are then used to update $\mathbf{y}_i$ on the master server using Eq. 10.

The proposed FCF method alternates between solving for $\mathbf{X}$ using Eq. 7 and then solving for $\mathbf{Y}$ using Eq. 10. However, as $\mathbf{Y}$ are updated using a gradient descent approach, multiple iterations of gradient descent updates are required to reach the optimum value of $\mathbf{Y}$. Therefore, a single epoch of FCF consists of an update to $\mathbf{X}$ as in CF and then several gradient descent steps to update $\mathbf{Y}$.

We analyze the Adaptive Moment Estimation (Adam) method [24] for FCF. In Adam the gradient descent is carried out in two separate parts which record an exponentially decaying average of past gradients m_t and squared gradients v_t,

$$m = \beta_1 m + (1 - \beta_1) \frac{\partial J}{\partial \mathbf{y}_i} \tag{13}$$

and

$$v = \beta_2 v + (1 - \beta_2) \left(\frac{\partial J}{\partial \mathbf{y}_i} \right)^2 \tag{14}$$

with $0 < \beta_1, \beta_2 < 1$. Typically, m and v are initialized to 0 values and hence biased towards 0. To counteract these biases, corrected versions of m, v are given by

$$\hat{m} = \frac{m}{1 - \beta_1}, \tag{15}$$

and

$$\hat{v} = \frac{v}{1 - \beta_2}. \tag{16}$$

The updates are then given by

$$\mathbf{y}_i \ = \ \mathbf{y}_i - \frac{\gamma}{\sqrt{\hat{v}} + \epsilon}\hat{m} \tag{17}$$

where $0 < \gamma < 1.0$ is a constant learning rate and $0 < \epsilon \ll 1$ e.g. 10^{-8} to avoid a divide by 0 scenario.

4.3 Privacy-by-Design Solution

Our privacy preserving federated solution does not requires the user's identity to be known at the server. This is primarily because each user sends the updates for $f(u, i)$ which are aggregated in Eq. 12 without reference to the user's identity.

5 DATA

5.1 Simulated data

A simulated dataset of user-item interactions was generated by randomly simulating users, movies, and view events. Specifically, we created a user-item interaction matrix consisting of zeros and ones. The data was required to be 80% sparse with the additional constraints that each user has at least eight view events and each item was watched at least once. The data dimensionality is specified in Table 1.

5.2 MovieLens

The MovieLens rating datasets have been collected and made available by the GroupLens research group (https://grouplens.org/). The dataset [25] used in this research work contains 1,000,209 anonymous ratings of 3952 movies made by 6040 users who joined MovieLens in the year of 2000. The data contains user-item interactions characterized by the user id, movie id, and explicit rating information. We transform the explicit ratings to the implicit feedback scenario of our proposed approach. Specifically, we assume that users watched the videos they rated and are not aware of the remaining ones. As a result, the explicit rating wherever available irrespective to its value, is transformed to one while missing ratings are assigned zeros. The transformation, therefore, represents the real implicit feedback data.

Algorithm 1 Federated Collaborative Filter (FCF)

```
input: M clients
  FL Server
  Initialize: $\mathbf{Y}$
  for each client $m \in M$ in parallel do
    $\nabla\mathbf{Y}^{(m)}$ = ClientUpdate($\mathbf{Y}$)
    $\mathbf{Y} = \mathbf{Y} - \gamma \sum_m \nabla\mathbf{Y}^{(m)}$ using Eqs. 10 & 12
  end for
  FL Client
  function: ClientUpdate($\mathbf{Y}$)
  update user factor $\mathbf{x}_u$ using Eq. 7
  compute item factor $\mathbf{Y}^{(m)}$ gradients: $\nabla\mathbf{Y}^{(m)}$ using Eq. 11
  return $\nabla\mathbf{Y}^{(m)}$ to server
```

Dataset	# users	# movies	# view events
Simulated	5000	40	8
MovieLens	6040	3952	20
In-house Production Data	6077	1543	20

Table 1: Overview of the datasets used in the study, where # view events refers to the minimal number of videos each user watched.

5.3 In-house Production Dataset

The in-house production dataset consists of a data snapshot extracted from the database that contains user view events. We carried out experiments using a subset of the anonymized dataset that was created by selecting view events for users who watched more than 20 videos and for videos that were viewed by more than 100 users. The overview of the dataset is listed in Table 1.

6 EXPERIMENTS AND RESULTS

6.1 Experimental setup

The CF and FCF models have three hyper-parameters in common: α, λ and the number of factors K. Whereas, the hyper-parameters related to the learning rate are specific to FCF only. To chose optimal hyper-parameters on real datasets, we used five-fold cross validation using Bayesian optimization approach [26]. We specified bounds for $K \in [2, 4]$, α & $\lambda \in [1, 5]$ and learned the optimal hyper-parameters using the python implementation[6]. We used a standard CF model to select the common hyper-parameters, whereas the parameters related to the Adam's learning rate were optimally chosen using the FCF model. Similar to CF, we used the Bayesian optimization to infer the FCF specific hyper-parameters using the bounds for β_1 & $\eta \in [0.0999, 0.50], \beta_2 \in [0.0999, 0.80], \epsilon \in [1e-8, 1e-5]$. The number of gradient descent iterations per epoch was set to 10.

For the simulated data experiments, we also wanted to evaluate how varying several hyper-parameters effect the model's behavior. Therefore, these were varied during the experiments. However, when not specified, the default values used for hyper-parameters in simulated data experiments were $\alpha = 1.0$, $\lambda = 1.0$, $\gamma = 0.05$, $K = 4$ and the number of gradient descent iterations per epoch as 20. All experiments were run for 20 epochs.

[6]https://github.com/fmfn/BayesianOptimization

6.2 Convergence Analysis

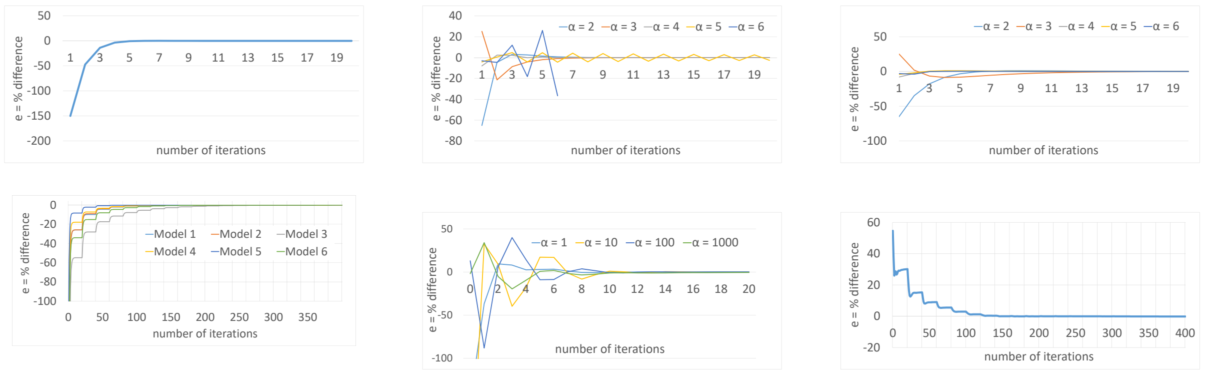

Figure 2: Convergence analysis of FCF model. The y-axis represents the % difference between the elements of latent factors $\mathbf{Y}_{\text{FCF}}$ to $\mathbf{Y}_{\text{CF}}$ whereas the x-axis shows iterations. With 20 gradient descent iterations per epoch the vertical lines indicate the start and end of an epoch. **Top-Left**, $\alpha = 1, \gamma = 0.05$, epoch $= 1$; **Top-Middle**, $\alpha = 2 - 6, \gamma = 0.05$, epoch $= 1$; **Top-Right**, $\alpha = 2 - 6, \gamma = 0.025$, epoch $= 2$; **Bottom-Left**, $\alpha = 10, \gamma = 0.05$, epoch $= 20$; **Bottom-Middle**, $\alpha = 1 - 1000, \gamma = 0.2$, epoch $= 1$ using Adam adaptive learning rate; **Bottom-Right**, $\alpha = 10, \gamma = 0.2$, epoch $= 20$ using Adam adaptive learning rate.

The proposed method to "federating" the CF can be seen as a variation of the standard ALS algorithm. The ALS algorithm learns the $\mathbf{X}, \mathbf{Y}$ factors by alternating between updates to $\mathbf{X}, \mathbf{Y}$ using Eq. 7 and 9 respectively. Each iteration of an update to both $\mathbf{X}, \mathbf{Y}$ is referred to as an *epoch*. In CF, each ALS alteration of the updates between $\mathbf{X}, \mathbf{Y}$ is convex, hence the rate of convergence of the model is expected to be quick and consistent.

Our proposed FCF method uses a similar approach but instead alternates between updating $\mathbf{X}$ using Eq. 7 and then solving for $\mathbf{Y}$ using gradient descent Eq. 8. The update of $\mathbf{Y}$, therefore, requires several iterations of gradient descent to reach the optimum value. Naturally, the FCF is also run in epochs where each epoch consists of an update to $\mathbf{X}$ as in CF and then several gradient descent steps to update $\mathbf{Y}$.

We demonstrate that FCF and CF models converge to the same optimum. The solutions are compared by calculating the average percentage difference between the elements of latent factors as

$$e = \frac{\sum_i \sum_k (\mathbf{Y}_{\text{CF}}(i,k) - \mathbf{Y}_{\text{FCF}}(i,k))/\mathbf{Y}_{\text{CF}}(i,k)}{M \times K} \times 100$$

where Y_{FCF} is the item-factor matrix of FCF and Y_{CF} is the item-factor matrix of CF.

First we compare the results after one epoch of the CF and FCF training, with both initialized with the same $\mathbf{X}, \mathbf{Y}$. To ensure consistency, $\mathbf{X}$ is updated in both cases resulting in the same values of $\mathbf{X}$ for both algorithms initially. The $\mathbf{Y}_{FCF}$ is updated using Eq. 10 and compared to the learned $\mathbf{Y}_{CF}$. We found that the FCF model converges after 5 gradient descent iterations to $\approx 0\%$ difference between the $\mathbf{Y}_{CF}$ and $\mathbf{Y}_{FCF}$ (Figure 2, top-left). The result confirms that the two models convergence to the same optimal. This is typically the case in any form of iterative optimization process. However, it does indicate that the FCF algorithm converges to the optimum ALS solution for this simple scenario.

Typically, the optimization of gradient descent approach depends on the appropriate choice of learning rate parameter γ [14], therefore, we next investigated the effects of γ on varying the value of implicit confidence parameter α (in Eq. 4). We observed that with $\gamma = 0.05$, stable convergence of the $\mathbf{Y}_{FCF}$ factor matrices was achieved for smaller values of $\alpha \in \{1, 2, 3, 4\}$, however, the convergence became un-stable for the larger α values (Figure 2, top-middle). However, decreasing the value of γ by half, results in a stable convergence for all the values of α (Figure 2, top-right).

We next compared the model parameters by increasing the number of training epochs from 1 to 20 to evaluate if the gradient descent estimation of $\mathbf{Y}_{\text{FCF}}$ converges globally, for higher values of $\alpha = 10$. To obtain robust evaluation, for FCF, we trained 6 models with the same hyper-parameters but different initialization of the factors $\mathbf{X}_{\text{FCF}}, \mathbf{Y}_{\text{FCF}}$, and evaluated the percentage difference at each gradient descent step of each epoch.

The results demonstrate that all 6 FCF model runs converged to the optimal solution of CF as shown in Figure 2, bottom-left, although at different rates. Specifically, 5 out of the 6 model runs converged within 6 epochs to the final $\mathbf{Y}_{\text{CF}}$ solution. These simulations indicate that the convergence is quite robust although the rates of convergence can vary.

For consistency purpose, we repeated the experiments with a higher $\gamma = 0.1$ (Supplementary Figure S1), which also led to convergence of all the models, however, required more iterations.

Several approaches can be used to stabilize the gradient descent methods including adaptively changing the learning rate γ or introducing gradient checking [24, 27]. Therefore, to tackle the diverging gradients, we introduce an adaptive learning rate based on Adam optimizer [24] into the FCF model. Using cross-validation, we identified values of $\beta_1 = 0.4$, $\beta_2 = 0.99$ and $\gamma = 0.2$ to produce stable convergence, where β_1 and β_2 are as defined in Eq. 13 and 14. The convergence for a single epoch and different values of α ranging form $1 - 1000$ are shown in Figure 2, bottom-middle. The results confirm that the convergence is stable across several orders of magnitude of the confidence parameter α, as the algorithm convergences around ~ 10 iterations in all cases. Finally, we also analyzed the adaptation of Adam's learning rate on the model's convergence over 20 epochs with $\alpha = 10$ and a substantially high $\gamma = 0.2$ confirming that the model converged to the CF solution within $6 - 7$ epochs (Figure 2, bottom-right).

In summary, the comprehensive simulations confirm that FCF models converged to the CF model's optimal solution, although at different rates. importantly, the Adam's learning rate is needed to stabilize the convergence in the presence of increasing implicit confidence parameter α.

6.3 Recommendation Performance

We next evaluate the recommendation performance of our proposed Federated Collaborative Filter in comparison to the standard Non-Federated Collaborative Filter [6]. To comprehensively compare the performance metrics we use the standard evaluation metrics [28], **Precision**, **Recall**, **F1**, and **Mean Average Precision (MAP)** and **Root Means Square Error (RMSE)** for the top 10 predicted recommendations. For completeness the metrics are defined in Supplementary Material. We also compute "**diff %**" between performance metrics of FCF and CF as

$$\textbf{diff \%} = \left|\frac{\text{MetricMean(FCF)} - \text{MetricMean(CF)}}{\text{MetricMean(CF)}}\right| \times 100$$

	CF	FCF	diff %
	Movie-Lens		
Precision	0.3008 ± 0.0079	0.2993 ± 0.0083	0.4987
Recall	0.1342 ± 0.0044	0.134 ± 0.0046	0.149
F1	0.1552 ± 0.0047	0.1548 ± 0.0049	0.2577
MAP	0.2175 ± 0.008	0.2155 ± 0.0082	0.9195
RMSE	0.6988 ± 0.056	0.6994 ± 0.0558	0.0859
	In-House		
Precision	0.0916 ± 0.0173	0.0914 ± 0.0172	0.2183
Recall	0.1465 ± 0.0289	0.146 ± 0.0289	0.3413
F1	0.1104 ± 0.0214	0.11 ± 0.0213	0.3623
MAP	0.0669 ± 0.017	0.0664 ± 0.0171	0.7474
RMSE	0.8076 ± 0.0316	0.8083 ± 0.0323	0.0867
	Simulated		
Precision	0.2014 ± 0.0057	0.2013 ± 0.0059	0.0497
Recall	0.8867 ± 0.0196	0.8863 ± 0.0199	0.0451
F1	0.3208 ± 0.0088	0.3207 ± 0.009	0.0312
MAP	0.5805 ± 0.0341	0.5798 ± 0.0341	0.1206
RMSE	0.5387 ± 0.0193	0.5391 ± 0.0192	0.0743

Table 2: Comparison of the test set performance metrics between Collaborative Filter (CF) and Federated Collaborative Filter (FCF) using different metrics averaged over all users. The values denote the **mean** ± **standard deviation** across 10 different model builds. The **diff %** refers to the percentage difference between the **mean** values of CF and FCF.

Table 2 shows test set performance metrics averaged across users over 10 rounds of the model rebuilds for the two real datasets and simulated dataset. In each model build, the data for each user was randomly divided into 60% training, 20% validation and 20% test sets. While validation and training sets were used to find optimal hyper-parameters and to learn model parameters, for example, the latent factors, the test set was purely employed to predict recommendations and to evaluate the performance scores on unseen user data. The results confirm that the FCF and CF model results are very similar in terms of test set recommendation performance metrics. On average, the percentage difference **diff %** CF and FCF across any of the five metrics is less than 0.5%. The standard deviations **std** are also small suggesting the multiple

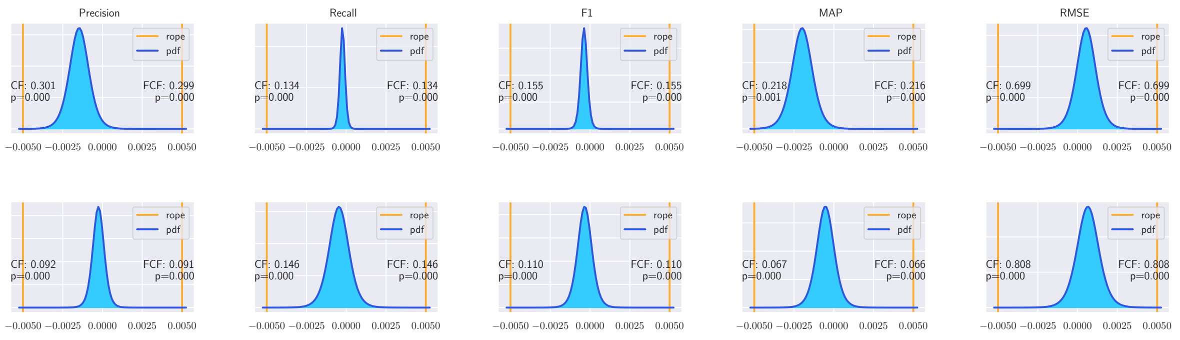

Figure 3: Comparisons between Collaborative Filter (CF) and Federated Collaborative Filter (FCF) in form of posterior distributions drawn from a correlated Bayesian t-test, on MovieLens (top row) and in-house production (bottom row) datasets. The various performance metrics, Precision, Recall, F1, MAP and RMSE are shown in columns. The vertical lines (rope) define a region of practical equivalence where the mean difference in performance is no more than $\pm 0.5\%$. The area under this distribution in the interval [-0.005, 0.005] is 0.999 confirming that the performance of two models is statistically similar.

runs converge to stable and comparable solutions. To further evaluate the statistical significance of the similarity between the results of CF and FCF, we performed a Bayesian correlated t-test [29, 30]. The input to the Bayesian t-test is lists of test set performance scores (for instance Precision, Recall, F1, MAP or RMSE) of the two models, obtained from 10 rebuilds. The test outputs a posterior distribution that describes the mean difference in performance scores between the CF and FCF models. Since posterior is a probability density function, it can be integrated to infer a probability of the hypothesis (that is the mean difference in recommendation performance scores is smaller than 0.5%).

Figure 3 shows the posterior distributions of the differences on MovieLens and in-house production datasets (in the case of simulated data, we show similar plots in Appendix). Consider, the results for evaluating the similarity in Precision values of the two models. The vertical lines (rope) define a region of practical equivalence where the difference in precision is no more than $\pm 0.5\%$ i.e. $[-0.005, 0.005]$. The FCF model has an average precision of 0.299 and CF 0.301. The plot shows the distribution of differences. The area under this distribution in the interval $(-\infty, 0.005)$ is $9.888e - 07$, which represents the probability of CF being better than FCF. Similarly, the area in the interval $(0.005, \infty)$ equals $1.000e - 04$ and denotes the probability that FCF being better than CF. The area between $[-0.005, 0.005]$, the defined region of practical equivalence (rope), is 0.999, which is interpreted as probability of being equivalent under then $\pm 0.5\%$ difference threshold. The results confirm that the precision values of CF and FCF are equivalent with a probability of 0.999. Moreover, all probability density functions in Figure 3 have more than 99% of the area within the rope, therefore, with no loss of generality, it can be claimed that the FCF and CF recommendation performance is statistically similar.

7 CONCLUSION

We introduced the first federated collaborative filtering method for privacy-preserving personalized recommendations. We presented the algorithm to federate the standard collaborative filter using stochastic gradient descent based approach. The convergence analysis demonstrated that the federated model achieves robust and stable solutions by incorporating an adaptive learning rate. The empirical evidence proved that the federated collaborative filter achieves statistically similar recommendation performance compared to the standard method. As a practical outcome, the results establish that the federated model can provide similar quality of recommendations as the widely used standard collaborative filter while fully preserving the user's privacy.

Future Work: We consider this work as a first step towards privacy-preserving recommendation systems with a federated model. However, we aim to explore multiple directions in this line of research in the future. A simulator-based analysis of the real-world scenario with updates arriving from clients in a continuously asynchronous fashion (online learning) could help benchmark the system. In addition, analysis on the communication payloads and efficiency could help evaluate the other practical aspects of such systems. Another important focus area for future studies is security in federated models. We intend to investigate the effects of attacks and threats by incorporating recently proposed methods [31, 32, 33] for secure federated learning.

References

[1] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[2] Latanya Sweeney. Simple demographics often identify people uniquely. *Health (San Francisco)*, 671:1–34, 2000.

[3] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[4] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[5] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[6] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[7] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *International Conference on Algorithmic Applications in Management*, pages 337–348. Springer, 2008.

[8] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable collaborative filtering approaches for large recommender systems. *Journal of machine learning research*, 10(Mar):623–656, 2009.

[9] Po-Lung Chen, Chen-Tse Tsai, Yao-Nan Chen, Ku-Chun Chou, Chun-Liang Li, Cheng-Hao Tsai, Kuan-Wei Wu, Yu-Cheng Chou, Chung-Yi Li, Wei-Shih Lin, et al. A linear ensemble of individual and blended models for music rating prediction. In *Proceedings of the 2011 International Conference on KDD Cup 2011-Volume 18*, pages 21–60. JMLR. org, 2011.

[10] Sebastian Schelter, Christoph Boden, Martin Schenck, Alexander Alexandrov, and Volker Markl. Distributed matrix factorization with mapreduce using a series of broadcast-joins. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 281–284. ACM, 2013.

[11] Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit S Dhillon. Parallel matrix factorization for recommender systems. *Knowledge and Information Systems*, 41(3):793–819, 2014.

[12] Martin Zinkevich, John Langford, and Alex J Smola. Slow learners are fast. In *Advances in neural information processing systems*, pages 2331–2339, 2009.

[13] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[14] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.

[15] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

[16] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77. ACM, 2011.

[17] Benjamin Recht and Christopher Ré. Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation*, 5(2):201–226, 2013.

[18] Adrian Nilsson, Simon Smith, Gregor Ulm, Emil Gustavsson, and Mats Jirstrand. A performance evaluation of federated learning algorithms. In *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning, DIDL@Middleware 2018, Rennes, France, December 10, 2018*, pages 1–8, 2018.

[19] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *CoRR*, abs/1811.03604, 2018.

[20] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *CoRR*, abs/1812.02903, 2018.

[21] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S. Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 4427–4437, 2017.

[22] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *CoRR*, abs/1806.00582, 2018.

[23] 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.

[24] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

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

[26] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[27] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[28] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutirrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109 – 132, 2013.

[29] Giorgio Corani and Alessio Benavoli. A bayesian approach for comparing cross-validaed algorihms on muliple daa ses. *Machine Learning*, 100(2-3):285–304, 2015.

[30] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

[31] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *CoRR*, abs/1807.00459, 2018.

[32] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *CoRR*, abs/1808.04866, 2018.

[33] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin B. Calo. Analyzing federated learning through an adversarial lens. *CoRR*, abs/1811.12470, 2018.