Personalized Federated Recommender Systems with Private and Partially Federated AutoEncoders

Qi Le
College of Science and Engineering
University of Minnesota-Twin Cities
Minneapolis, USA
le000288@umn.edu

Enmao Diao
Electrical and Computer Engineering
Duke University
Durhm, USA
enmao.diao@duke.edu

Xinran Wang
College of Science and Engineering
University of Minnesota-Twin Cities
Minneapolis, USA
wang8740@umn.edu

Ali Anwar
College of Science and Engineering
University of Minnesota-Twin Cities
Minneapolis, USA
aanwar@umn.edu

Vahid Tarokh
Electrical and Computer Engineering
Duke University
Durhm, USA
vahid.tarokh@duke.edu

Jie Ding
School of Statistics
University of Minnesota-Twin Cities
Minneapolis, USA
dingj@umn.edu

***Abstract*—Recommender Systems (RSs) have become increasingly important in many application domains, such as digital marketing. Conventional RSs often need to collect users' data, centralize them on the server-side, and form a global model to generate reliable recommendations. However, they suffer from two critical limitations: the personalization problem that the RSs trained traditionally may not be customized for individual users, and the privacy problem that directly sharing user data is not encouraged. We propose Personalized Federated Recommender Systems (PersonalFR), which introduces a personalized autoencoder-based recommendation model with Federated Learning (FL) to address these challenges. PersonalFR guarantees that each user can learn a personal model from the local dataset and other participating users' data without sharing local data, data embeddings, or models. PersonalFR consists of three main components, including AutoEncoder-based RSs (ARSs) that learn the user-item interactions, Partially Federated Learning (PFL) that updates the encoder locally and aggregates the decoder on the server-side, and Partial Compression (PC) that only computes and transmits active model parameters. Extensive experiments on two real-world datasets demonstrate that PersonalFR can achieve private and personalized performance comparable to that trained by centralizing all users' data. Moreover, PersonalFR requires significantly less computation and communication overhead than standard FL baselines.**

***Index Terms*—data heterogeneity, federated learning, personalized recommendation, privacy**

I. Introduction

As a result of the fast rise and widespread use of internet services and applications, Recommender Systems (RSs) have become essential in many fields, including digital marketing, customized health, and data mining [1]. RSs can assist users in making efficient use of available information. Most recent works on RSs require all the data from multiple domains to be shared and the calculation for model training to be performed centrally. However, this centralization has two significant limitations: 1) The individual users may not receive personalized models since the global model that the RSs learned traditionally needs to account for the data heterogeneity among users [2]. 2) As users' data held on the server may be inadvertently disclosed or exploited, the centralized training naturally leads to privacy concerns [3].

Federated learning (FL) [4]–[7] is a distributed machine learning framework that allows users to train models without direct data sharing. By distributing the model training process to local clients, FL utilizes local compute resources and ensures that user data remain on the client's devices. Federated averaging (FedAvg) [5] is a popular training algorithm that allows multiple local updates, which may facilitate convergences and communication efficiency. However, due to potential data distributional heterogeneity [4], [8], FL may need better convergence [9] and to provide personal recommendations for different users.

Motivated by the aforementioned issues, this work proposes Personalized Federated Recommender Systems (PersonalFR), which introduces a personalized AutoEncoder-based recommendation model with Federated Learning (FL). The basic architecture of PersonalFR is training the personalized encoder for each client to establish client-specific mapping information and averaging the decoder on the server side. We aim to use local computing resources, complete local training without data sharing, and provide heterogeneous users with personalized recommendations. Specifically, the server picks a group of accessible clients and provides them with the global decoder. Then, each client updates the local AE model with respect to the local objective function for several epochs. After that, the server leverages Partially Federated Learning (PFL) and Partial Compression (PC) to aggregate local decoders and update the global decoder model. Once the global decoder converges reasonably well, each client uses its customized AE model to obtain recommendations. The suggested PersonalFR with PFL shows faster convergence than standard FL baselines in our extensive experiments. Moreover,

Qi Le, Jie Ding, and Vahid Tarokh were supported in part by the Office of Naval Research under grant number N00014-21-1-2590.

using the PC component, PersonalFR requires substantially less processing and transmission overhead than conventional FL baselines. Our contributions are summarized below.

- We present a new framework of Recommender Systems (RSs), which can provide precise, private, and personalized recommendations without sharing local data, data embeddings, or models.
- By leveraging Partially Federated Learning (PFL), we show that our PersonalFR can outperform FedAvg and achieve private and personalized performance comparable to that trained by centralizing all the users' data.
- We propose Partial Compression (PC) that only computes and transfers the active model parameters. Our experimental results show that the PersonalFR compresses the computation overhead around $1.25\times$ to $1.9\times$ over FedAvg and communication overhead around $2.5\times$ to $27\times$ over FedAvg.

II. Related Work

A. Recommender Systems

Recommendation Systems (RSs) may be divided into three rough categories [10]: content-based filtering, collaborative filtering, and hybrid methods. Content-based filtering [11] predicts users' preferences based on their side information, e.g., personal information. Collaborative filtering [11] leverages user-item interactions and the items that users with comparable preferences favored to infer users' preferences for particular items. The hybrid methods [12] combine content-based filtering and collaborative filtering. Our proposed method is a collaborative filtering recommender system that uses user-item interactions.

B. Personalized Federated Learning

Personalized Federated Learning has been a solution to modeling heterogeneous data to provide users with personalized models. There are two rough categories of personalized federated learning [13]: global model personalization and personalized models. The first category trains a good globally-shared FL model, and then the trained global FL model is adapted locally for each FL client [14]. For example, FAug [15] tries to mitigate statistical heterogeneity across clients' datasets. MAML [16] seeks to develop a globally generalizable model. The second category focuses on training personalized FL models for clients. For example, FedMD [17] and SPIDER [18] aim to establish client-specific model architectures via knowledge distillation and neural architectural search, respectively. FedAMP [19] utilizes similarity between clients' data distributions to enhance the performance of tailored models. Self-FL [13] automatically tunes clients' local model initialization, training steps, and server aggregation based on balancing the inter-client and intra-client model uncertainty from a Bayesian hierarchical modeling perspective. We refer to [13] for a more detailed literature review on personalized federated learning. The existing federated personalization methods often require additional resources in computation and communication. This is particularly a concern for many practical applications of recommender systems that require a large model to address a large scale of users, items, and ratings heterogeneously distributed among users [10]. It has motivated our work to study a more efficient personalized federated learning solution to recommender systems.

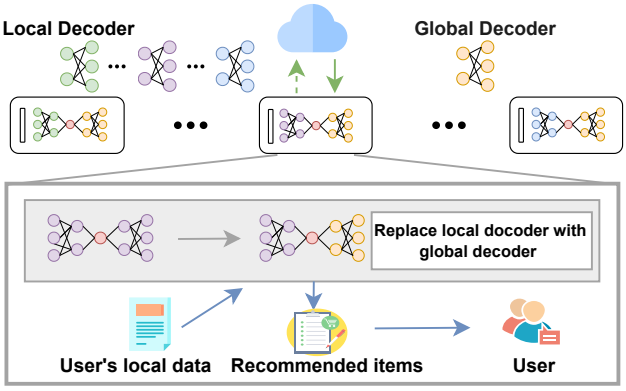


Fig. 1: Personalized Federated Recommender Systems. User can obtain improved personal recommendations by leveraging the data from global domain without sharing the local data and encoder part of the model.

III. Method

In this section, we present the framework architecture for PersonalFR that offers personalized recommendations shown in Figure 1.

A. Problem Formulation

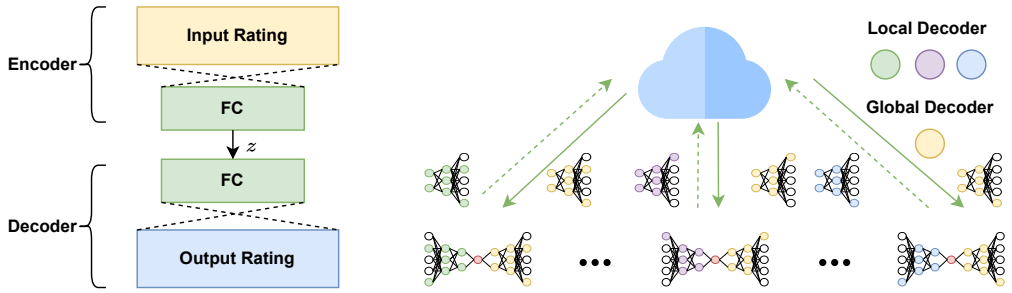
Our PersonalFR is a rating-based collaborative filtering recommender system that predicts users' explicit preference ratings for items based on user-item interactions. We define $U \triangleq \{u_1, \ldots, u_k\}$ be the set of k users and $V \triangleq \{v_1, \ldots, v_n\}$ be the set of n items. Then, we have a user-item interaction matrix $R = [r_{i,j}]_{1\le i\le k, 1\le j\le n} \in \mathbb{R}^{k\times n}$, where each element $r_{i,j}$ indicates the rating score that user u_i assigns to the item v_j. Utilizing a recommender system, we can predict $\hat{r}_{i,j}$ for user u_i attributing to the item v_j. The objective of training a recommender system is to minimize the average of following training loss for all the rating scores

$$\sum_{1\le i\le k, 1\le j\le n, r_{i,j}\neq 0} \ell(\hat{r}_{i,j}, r_{i,j}) \tag{1}$$

over model parameters that define $\hat{r}_{i,j}$'s, where $\ell(\cdot)$ is the loss function. When calculating the loss value, our PersonalFR will mask out the unrated user-item interactions and minimize the average of the loss values between the rated user-item interactions and the fitted user-item interactions. We will use quadratic loss for regression and cross-entropy loss for classification in the experimental study.

B. AutoEncoder

AutoEncoder (AE) has many successful applications in RSs [12], [20]. AE encodes a high-dimensional input signal into a low-dimensional hidden representation, and then decodes it into an output representation. The structure of AE is shown in Figure 2. In addition, AE considers rating matrices



(a) (b)

Fig. 2: (a) Illustration of AutoEncoder. The input rating dimension is the number of items of that domain, while the output dimension is the same as the input dimension. (b) Illustration of Partially Federated Learning and Partial Compression. Only the active parameters, as shown in Expression (3), will be transmitted.

as tabular data, where rows represent subjects and columns represent features. We define $\mathcal{X} \triangleq \{x_1, \ldots, x_k\}$ as the set of all user rating vectors, where $x_i \triangleq (r_{i,1}, \ldots, r_{i,n}) \in \mathbb{R}^n$ denotes the rating scores that user u_i assigns to the items $v_1 \ldots v_n$. All user rating vectors are sparse vectors, where the unrated user-item interactions are zeros. We define $\hat{\mathcal{X}} \triangleq \{\hat{x}_1, \ldots, \hat{x}_k\}$ as the set of all predicted user rating vectors, where $\hat{x}_i \triangleq (\hat{r}_{i,1}, \ldots, \hat{r}_{i,n}) \in \mathbb{R}^n$ represents the predicted rating scores that user u_i gives to the items $v_1 \ldots v_n$. Our AE takes the vector x_i as the input and produces the vector $\hat{x}_i$ as the output. The output vector has the same dimension as the input vector. In particular, AE consists of an encoder $z = E(x_i) : R^n \rightarrow R^{d_{\text{hidden}}}$ and a decoder $\hat{x}_i = D(z) : R^{d_{\text{hidden}}} \rightarrow R^n$, where n and d_{hidden} represent the dimensions of the input/output vector and the latent vector z, respectively.

C. Personalized Federated Recommender Systems (PersonalFR)

In this section, two critical components of PersonalFR are described. The first key component is the personalized partial update of the client and server model, which we refer to as Partially Federated Learning (PFL). The second key component is the optimized computation and communication during the training process, which we call Partial Compression (PC). PC is built upon the PFL, and these elements contribute together to make PersonalFR perform better and need fewer resources than FedAvg for recommender systems. Furthermore, we summarize the pseudocode of the PersonalFR in Algorithm 1 and present the workflow of PersonalFR at the end of this section.

Before describing the details of two key components, PFL, and PC, we first introduce the general settings and notations. Let $\mathcal{M}$ be the set of the clients, whose cardinality is M. Each client owns a unique encoder and shares a global decoder. For a generic decoder, we define the vector $H \triangleq (h_1, \ldots, h_q) \in \mathbb{R}^q$ as the output of the last hidden layer, $\mathcal{W} \triangleq \{W_1, \ldots, W_p\}$ as the full set of weights, $\mathcal{B} \triangleq \{B_1, \ldots, B_p\}$ as the full set of biases, where p is the number of layers. Each element of $\mathcal{W}$ is a matrix, and each element of the $\mathcal{B}$ is a vector. In the sequel, for the above quantities associated with a particular client m, we will put a subscript m to highlight such association.

Let $\mathcal{M}$ be the set of the clients, whose cardinality is M. For client $m \in \mathcal{M}$, we use $\mathcal{X}_m$ and $\hat{\mathcal{X}}_m$ to represent all the user rating vectors and all the predicted user rating vectors within the client m, respectively, such that $\mathcal{X} = \cup_{m \in \mathcal{M}} \mathcal{X}_m$ and $\hat{\mathcal{X}} = \cup_{m \in \mathcal{M}} \hat{\mathcal{X}}_m$. For client m, the output vector $\hat{x}_{m,i} = (\hat{r}_{i,1}, \ldots, \hat{r}_{i,n}) \in \mathbb{R}^n$ from the AE model is generated from

$$\hat{x}_{m,i} = W_{m,p} \cdot H_m + B_{m,p}, \tag{2}$$

where $W_{m,p} \in \mathbb{R}^{n \times q}$ and $B_{m,p} \in \mathbb{R}^n$ represent the weight matrix and the bias of the output layer associated with the client m, respectively.

As shown in Table I, the observable data associated with a typical recommender system is highly sparse. As a result, for every client, only a tiny fraction of items, say $\{v_1, \ldots, v_{n'}\}$, where $n' \ll n$, are rated by at least one user of that client. Thus, only the submatrix $W'_{m,p} \in \mathbb{R}^{n' \times q}$ of the weight matrix $W_{m,p}$ and the subvector $B'_{m,p} \in \mathbb{R}^{n'}$ of the bias vector $B_{m,p}$ connected to the rated items will be effectively used to predict rating scores. In line with that, only $W'_{m,p}$ and $B'_{m,p}$ will be updated in the back-propagation during the local training.

Then, we elaborate on more details of PFL, which trains a personalized encoder for each client to generate client-specific encoder mapping and averages the decoder on the server side to gain model improvement from other clients' models. More specifically, for each client, PFL trains its personalized encoder together with the global decoder on the local dataset. Then, only the decoder will be processed during the transmission and server aggregation. This is different from FedAvg that transfers and averages the whole model. The unshared personalized encoder of each client can extract the unique features of each client's input. Consequently, the procedure for calculating predicted rating scores differs for FedAvg and PersonalFR. In particular, for client m in FedAvg, the predicted rating scores are $\hat{x}_{m,i} = D(E(x_i)) \in \mathbb{R}^n$ for user u_i attributing to all n items, where $D(\cdot)$ is the global decoder and $E(\cdot)$ is the global encoder. In contrast, in PersonalFR, the predicted rating scores are $\hat{x}_{m,i} = D(E_m(x_i)) \in \mathbb{R}^n$ for user u_i attributing to the all n items, where $D(\cdot)$ is the global decoder and $E_m(\cdot)$ is the personalized encoder of the client containing user

Algorithm 1: PersonalFR: Personalized Federated Recommender Systems

Input: Data $\mathcal{X}$ distributed on M local clients ($\mathcal{X} = \mathcal{X}_1 \cup \cdots \cup \mathcal{X}_m$), fraction C of selected clients per communication round, local minibatch size B, learning rate η, total number of communication rounds T, number of local epochs K, local encoders distributed on M clients parameterized by E_m ($m = 1, \ldots, M$), globally shared decoder parameterized by D_g.

System executes:
- Initialize D_g^0
- **for** each client $m \in \mathcal{M}$ **in parallel do**
  - Initialize E_m^0
  - Send indices of rated user-item interactions I_m to the server
- **end**
- Server records all clients' indices $I = \{I_1, \ldots, I_m\}$
- **for** each communication round $t = 1, \ldots, T$ **do**
  - $\mathcal{M}^t \leftarrow \max(C \cdot M, 1)$ clients sampled from $\mathcal{M}$
  - Initialize decoder parameters set $\mathcal{S}^t \leftarrow \varnothing$
  - **for** each client $m \in \mathcal{M}^t$ **in parallel do**
    - $E_m^t \leftarrow E_m^{t-1}$
    - $D_{m,\text{active}}^t \leftarrow$ Server sends active parameters based on D_g^{t-1} and I_m(See Expression (3))
    - $E_m^t, D_{m,\text{active}}^t \leftarrow \text{ClientUpdate}(E_m^t, D_{m,\text{active}}^t)$
    - $\mathcal{S}^t \leftarrow \mathcal{S}^t \cup \{D_{m,\text{active}}^t\}$
  - **end**
  - $D_g^t \leftarrow \text{ServerAggregation}(D_g^{t-1}, \mathcal{S}^t)$
- **end**

ClientUpdate $(E_m^t, D_{m,\text{active}}^t)$**:**
- $B_m \leftarrow$ Split local data $\mathcal{X}_m$ into batches of size B
- **for** each local epoch $l = 1, \ldots, K$ **do**
  - **for** batch $b_m \in B_m$ **do**
    - $E_m^t \leftarrow E_m^t - \eta \nabla_E L(D_{m,\text{active}}^t(E_m^t), b_m)$
    - $D_{m,\text{active}}^t \leftarrow D_{m,\text{active}}^t - \eta \nabla_D L(D_{m,\text{active}}^t(E_m^t), b_m)$
    - (L: total loss for b_m based on Formula (1))
  - **end**
- **end**
- Return E_m^t, $D_{m,\text{active}}^t$

ServerAggregation $(D_g^{t-1}, \mathcal{S}^t)$**:**
- $D_g^t \leftarrow D_g^{t-1}$
- **for** each active parameters $D_{m,\text{active}}^t \in \mathcal{S}^t$ **do**
  - $D_m^t \leftarrow$ Use $D_{m,\text{active}}^t$ and D_m^{t-1} to update D_m^t according to Equation (5)
  - $D_g^t \leftarrow D_g^t + \frac{1}{|\mathcal{S}^t|} D_m^t$ ($|\mathcal{S}^t|$: cardinality of $\mathcal{S}^t$)
- **end**
- Return D_g^t

u_i. Moreover, it is worth mentioning that keeping the private personalized encoder for all clients not only helps us improve the local model's performance but also helps us with some security parts. For example, it would be challenging to recover the user data by possible attackers knowing the parameters of the globally shared decoder and the output vector for a client.

Next, we show how PC is built upon the PFL to further reduce computation and communication costs. In PersonalFR, we only update and transfer the active parameters D_{active} of a generic decoder within the training process. The active parameters $D_{m,\text{active}}$ for the client m can be expressed as

$$D_{m,\text{active}} \triangleq \{W_{m,1}, B_{m,1}, \ldots, W'_{m,p}, B'_{m,p}\}, \tag{3}$$

which represents all the weight matrices and bias vectors that will be updated during the local back-propagation of the decoder D_m. Here, $W_{m,p}$ and $B_{m,p}$ in $D_{m,\text{active}}$ are the weight matrix and bias vector of the output layer, respectively. On the other hand, in FedAvg, all the parameters of the decoder of client m need to be updated and transferred during the training stage, which can be represented by

$$D_m \triangleq \{W_{m,1}, B_{m,1}, \ldots, W_{m,p}, B_{m,p}\} \tag{4}$$

$D_{m,\text{active}}$ only occupies a tiny portion, e.g., as small as 3.7% (in our experiment study), of D_m. This is because the weight matrix $W_{m,p}$ and bias vector $B_{m,p}$ of the output layer occupy a significant portion of decoder parameters. By shrinking them to $W'_{m,p}$ and $B'_{m,p}$ using PC, we can greatly reduce computing and communication resources. Besides, during the server aggregation step at communication round t, we need first to update W_p^t and B_p^t of the output layer of D_m^t for each participating client. More particularly, for client m, we update $W_{m,p}^t$ and $B_{m,p}^t$ using $W_{m,p}^{\prime t}$ and $B_{m,p}^{\prime t}$, respectively. The following equation illustrates the procedure

$$\begin{aligned} W_{m,p}^t &= W_{m,p}^{\prime t} \cup (W_{m,p}^{t-1} \backslash W_{m,p}^{\prime t-1}) \\ B_{m,p}^t &= B_{m,p}^{\prime t} \cup (B_{m,p}^{t-1} \backslash B_{m,p}^{\prime t-1}), \end{aligned} \tag{5}$$

where $W_{m,p}^{t-1}$ represents the weight matrix of the output layer of D_m^{t-1} at communication round $t-1$, $W_{m,p}^{\prime t}$ represents the submatrix of $W_{m,p}^t$ connected to the rated items of client m at communication round t, and $W_{m,p}^{t-1} \backslash W_{m,p}^{\prime t}$ represents the set of parameters in $W_{m,p}^{t-1}$ but not in $W_{m,p}^{\prime t}$.

At last, we summarize the execution flow of our PersonalFR system. Algorithm 1 presents the pseudocode of the PersonalFR. In the beginning, we initialize the AE models for all clients. Then, to find the active parameters, each client needs to send the indices of the rated user-item interactions to the server, and the server records all clients' indices. Each client can now locate its active parameters and begin using the PC. Afterward, the server selects a batch of available clients and sends the current global decoder. Throughout several local epochs, each client trains the local AE model with respect to the local objective function and uploads the new local decoder parameters using PC to the server. Last, the server updates the global decoder model by utilizing the selected clients' active parameters and the previously recorded indices. The above training procedure is repeated until the global decoder converges. Finally, in the prediction stage, each client utilizes its personalized AE model to obtain recommendations.

IV. Experiments

A. Experimental Setup

Models and Datasets. We conduct our experiments on two public datasets: MovieLens1M (ML1M) [21], which is a dataset of movie ratings, and Anime [22], which is a dataset of anime ratings. The detailed attributes of these two datasets

are listed in Table I. We filter out users and items with fewer than 20 ratings for the Anime dataset and pick the first 6000 users. The details of hyperparameters for model training are listed in Table II. We have the following control settings.

1) Different number of clients. We evaluate the performance of PersonalFR and FedAvg under various numbers of clients and data heterogeneity scenarios. While the numbers of clients are different, the total amounts of the available data remain the same for ML1M and Anime datasets, respectively. As the number of clients increases, each client will own fewer users and less available data.

2) Explicit versus implicit feedback [23]. The explicit feedback is the default rating (1-5 for the ML1M dataset, 1-10 for the Anime dataset). In contrast, the implicit feedback is the binarized rating (positive if greater than 3.5 for the ML1M dataset, positive if greater than 8 for the Anime dataset). We regard the explicit feedback as the regression task and the implicit feedback as the binary classification task. We use the l_2-norm as the loss function and the Root Mean Square Error (RMSE) as the evaluation metric for explicit feedback. We use the cross-entropy as the loss function and the Normalized Discounted Cumulative Gain (NDCG) as the evaluation metric for implicit feedback.

3) With versus without compression. We contrast the computation and communication costs of PersonalFR runs on the ML1M and Anime datasets with those of FedAvg.

4) Ablation studies. For each dataset, we train on 80% of the available data and test on the remaining 20%. Four random experiments are conducted to report the standard errors of performance metrics, which are all smaller than 4×10^{-3}.

Baseline We compare the proposed method with two baselines, 'Joint' and 'FedAvg.' 'Joint' refers to the centralized case where a single entity owns all data. The 'FedAvg' denotes the case where the standard FL baseline is applied. Our method aims to outperform the 'FedAvg' case and perform competitively with the 'Joint' case.

Dataset	k	n	sparsity
ML1M	6040	3706	96%
Anime	69600	9927	99%

TABLE I: Detailed attributes of the ML1M and Anime datasets. Each dataset contains k users and n items. Sparsity means the percentage of unrated user-item interactions over all user-item interactions.

B. Experimental Results

Tables III and IV lists the experimental outcomes. The standard deviations are shown in brackets with four random experiments. In Figures 3, 4, and 5, we depict evaluations across various contexts. We offer thorough explanations below.

Effect of the number of clients III As shown in Figure 3, we record the learning curves of FedAvg and PersonalFR of the ML1M and Anime datasets for explicit and implicit feedback. As a result, our PersonalFR converges faster and better than FedAvg while keeping a personalized encoder. Moreover, the performance of PersonalFR using explicit and implicit feedback is close to centralized training, as shown

Model	AutoEncoder							
Hidden size	[n,256,128], [128,256,n]							
Global Epoch	800							
Local Epoch K	5							
Momentum	0.9							
Weight decay	5.00E-04							
Number of clients M	1		100		300		6040 (1 User / Client)	
Data	ML1M	Anime	ML1M	Anime	ML1M	Anime	ML1M	Anime
Optimizer	Adam [24]		SGD					N/A
Local Batch Size B	500	100	10					
Learning rate	1.00E-03		1.00E-01					

TABLE II: Hyperparameters of our experiments for training local models. The size of our encoder is [n,256,128], where n is the size of the input. The size of our decoder is [128,256,n], where n is the size of the output, the same size as the input.

in Table III. If we compare the performance of FedAvg and PersonalFR on 300 clients with that of 100 clients, we observe a decline in performance for either explicit or implicit feedback. This decline is perhaps because FedAvg has the known issue of gradient divergence; namely, the directions of the gradient updates generated from each selected client can be significantly different [8], [9], especially when clients' data are heterogeneous. As the number of clients increases, data distributional heterogeneity and gradient divergence become increasingly problematic.

To further demonstrate our PersonalFR method, we test the situation where each user obtains a private and personalized model for the ML1M dataset. In the ML1M dataset, there are 6040 users, so we have 6040 clients. We summarize the results in Table IV and show the learning curve for explicit and implicit feedback in Figure 4. Under this scenario, our PersonalFR performs significantly better than the FedAvg for explicit feedback, indicating that personalized encoder mapping is increasingly helpful for predicting user preferences. Moreover, in the experiments, the hyperparameters are not tuned for optimal performance in the federated setting. Thus, the experimental results could be potentially improved further.

Explicit versus implicit feedback As shown in Table III and IV, our PersonalFR outperforms the FedAvg in most scenarios and achieves competitive results in the 'Joint' scenario. Furthermore, we can observe that when the number of clients increases, the performance drops of FedAvg and PersonalFR for explicit feedback are more significant than those of implicit feedback according to Table III and Table IV. Moreover, for both explicit and implicit feedback, the performance of PersonalFR drops less than that of FedAvg. Especially for explicit feedback, the performance of PersonalFR drops significantly less than that of FedAvg. Therefore, PersonalFR has much more advantages than FedAvg for explicit feedback.

With versus without compression We conduct experiments for PC with various numbers of client scenarios. We show the results in Figure 4. Our personalFR can compress the computation overhead around $1.25\times$ to $1.9\times$ over FedAvg and communication overhead around $2.5\times$ to $27\times$ over FedAvg according to the number of clients. Furthermore, the results show that as the number of clients increases, the local dataset

Dataset		ML1M		Anime	
Metric		RMSE(↓)	NDCG(↑)	RMSE(↓)	NDCG(↑)
Joint	Upper Bound	0.8591(0.0003)	0.8466(0.0024)	1.1926(0.0007)	0.8576(0.0018)
100 Clients	FedAvg	0.8629(0.0009)	0.8514(0.0012)	1.2303(0.0014)	0.8606(0.0041)
	PersonalFR	**0.8613(0.0004)**	**0.8538(0.0018)**	**1.2121(0.0004)**	**0.8621(0.0024)**
300 Clients	FedAvg	0.8731(0.0006)	0.8486(0.0025)	1.2438(0.0012)	0.8603(0.0025)
	PersonalFR	**0.8602(0.0007)**	**0.8529(0.0017)**	**1.2247(0.0007)**	**0.8611(0.0024)**

TABLE III: Results of ML1M and Anime datasets for explicit and implicit feedback. ↓ indicates the smaller the better, while ↑ indicates the larger the better.

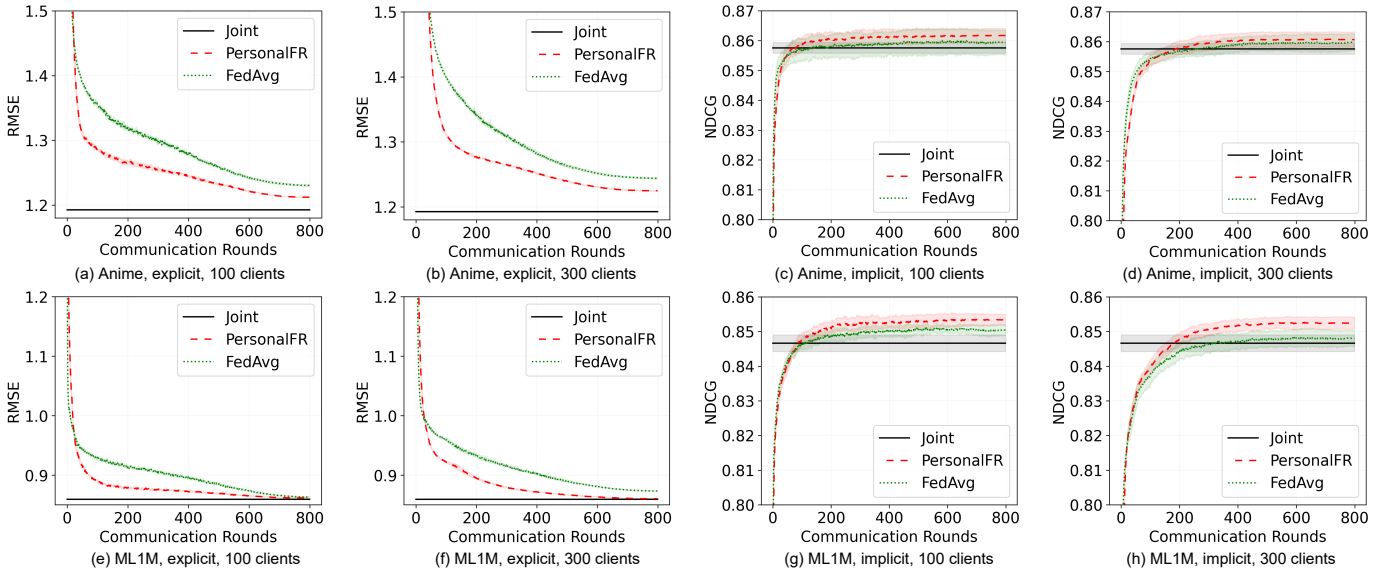


Fig. 3: Learning curves of the ML1M and Anime datasets for explicit feedback measured with RMSE and implicit feedback measured with NDCG trained by FedAvg and PersonalFR, respectively. (a-d) Anime dataset. (e-h) ML1M dataset.

Dataset		ML1M	
Metric		RMSE(↓)	NDCG(↑)
Joint	Upper Bound	0.8591(0.0003)	0.8466(0.0024)
6040 Clients (1 User / Client)	FedAvg	0.9508(0.0008)	0.8321(0.0054)
	PersonalFR	**0.8983(0.0017)**	**0.8379(0.0021)**

TABLE IV: Results of ML1M for explicit and implicit feedback under 6040 clients (1 User/Client) situation

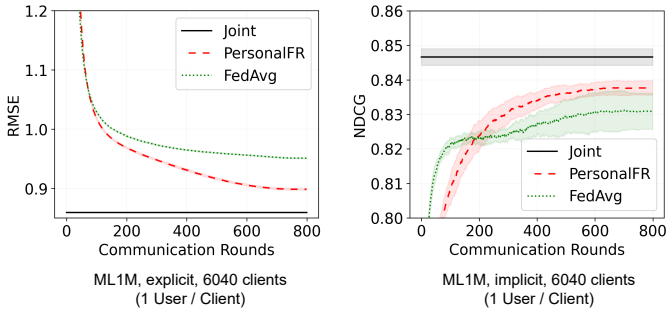


Fig. 4: Learning curves of the ML1M dataset for explicit (left) and implicit (right) feedback measured with RMSE trained by FedAvg and PersonalFR under 6040 clients(1 User / Client) situation, respectively.

for each client becomes sparser, meaning that fewer rated items are observed, and the proposed PC achieves a more significant reduction in computation and communication.

V. Conclusion and Future Work

In this work, we propose Personalized Federated Recommender Systems (PersonalFR), which combines a personalized AutoEncoder-based recommendation model with Federated Learning (FL). We demonstrate that by using Partially Federated Learning (PFL), our PersonalFR can surpass the FedAvg and obtain private and customized performance close to that achieved by centralizing all user data. Furthermore, the PersonalFR requires far less computation and communication overhead than the FedAvg by applying Partial Compression (PC). An interesting future problem is to address the performance decline that occurs as the number of clients increases. In addition, one may examine the newly developed approach to federated recommender systems in the presence of various adversarial attacks.

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, 2005.

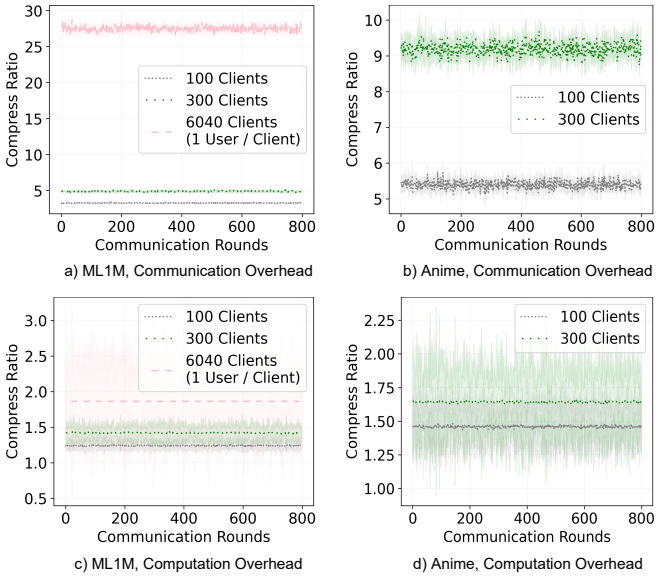


Fig. 5: The compress ratio of the computation and communication overhead of PersonalFR runs on the ML1M and Anime datasets compared to those of the FedAvg. (a-b) Communication overhead. (c-d) Computation overhead.

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, 2020.

[3] B. Zhang, N. Wang, and H. Jin, "Privacy concerns in online recommender systems: influences of control and user data input," in *Proc. SOUPS*, 2014, pp. 159–173.

[4] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017, pp. 1273–1282.

[6] E. Diao, J. Ding, and V. Tarokh, "HeteroFL: Computation and communication efficient federated learning for heterogeneous clients," *Proc. ICLR*, 2020.

[7] ——, "SemiFL: Communication efficient semi-supervised federated learning with unlabeled clients," *Proc. NeurIPS*, 2022.

[8] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *Proc. ICML*, 2020, pp. 5132–5143.

[9] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[10] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender systems: an introduction*. Cambridge University Press, 2010.

[11] B. Rocca, "Introduction to recommender systems: Overview of some major recommendation algorithms," *Towards Data Science*, 2019.

[12] E. Diao, V. Tarokh, and J. Ding, "Privacy-preserving multi-target multi-domain recommender systems with assisted autoencoders," *arXiv preprint arXiv:2110.13340*, 2022.

[13] H. Chen, J. Ding, E. Tramel, S. Wu, A. K. Sahu, S. Avestimehr, and T. Zhang, "Self-aware personalized federated learning," *Proc. NeurIPS 2022*, 2022.

[14] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," *arXiv preprint arXiv:2002.10619*, 2020.

[15] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.

[16] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," *Proc. NeurIPS*, vol. 33, pp. 3557–3568, 2020.

[17] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," *arXiv preprint arXiv:1910.03581*, 2019.

[18] E. Mushtaq, C. He, J. Ding, and S. Avestimehr, "SPIDER: Searching personalized neural architecture for federated learning," *arXiv preprint arXiv:2112.13939*, 2021.

[19] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, and Y. Zhang, "Personalized cross-silo federated learning on non-iid data." in *AAAI*, 2021, pp. 7865–7873.

[20] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proc. ICWWW*, 2015, pp. 111–112.

[21] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[22] CooperUnion, "Anime recommendations database," http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/.

[23] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. ICDM*. Ieee, 2008, pp. 263–272.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.