A Personalized Federated Tensor Factorization Framework for Distributed IoT Services QoS Prediction from Heterogeneous Data

Xiaoli Li, Shixuan Li, Yuzheng Li, Yuren Zhou, *Member, IEEE,* Chuan Chen, *Member, IEEE,* and Zibin Zheng, *Senior Member, IEEE*

***Abstract*—With a growing number of alternative IoT services that provide the same functionalities, Quality-of-Service (QoS) prediction has become an important research issue. Conventional central methods require that the historical QoS data is centralized. While the QoS data may be distributed in different edge servers. Due to privacy, these edge servers may not be willing to share their data with others for a better model representation. Besides, as each edge server is in charge of the collecting QoS data from the users in a specific area, the QoS data are likely to be heterogeneous. In this paper, we propose a personalized federated tensor factorization framework for distributed privacy-preserving IoT Services QoS Prediction. We first adopt tensors to represent the QoS data with multi-dimensions, and initialize a personalized model for each edge server. Then, each edge server performs local tensor factorization and exchanges the public component with a parameter master. By constraining the consistency of the global public component and the local personalized public components, the global model can learn information from edge servers during the training process. In addition, we conduct extensive experiments on a real-world QoS dataset, the experimental results demonstrate that the proposed framework is efficient and effective.**

***Index Terms*—Personalized Federated Learning, Distributed QoS prediction, Heterogeneous.**

I. Introduction

A. IoT Services QoS Prediction

The Internet of Things (IoT) is a new paradigm, in which the intelligent devices communicate with each other through the Internet [1]. With the development of IoT, more and more IoT devices, such as smartphones, tablets, wearable devices, and autonomous vehicles, are available [2]. On this foundation, IoT service suppliers offer a multitude of IoT services to realize the intelligence of human lives [3]. IoT services represent a set of end-to-end services that encapsulate application functionalities and information resources, and can adapt to IoT environments. Many of these large numbers of IoT services provide similar functionality. Accordingly, choosing the most appropriate IoT service from a large number of services with similar functions is becoming a challenging problem. QoS, which can describe the non-functional characteristics, is recognized as an important criterion to differentiate IoT services with similar functionality. QoS-based IoT service recommendation, which helps to find the most appropriate IoT services according to the QoS, has become an important research topic. However, these methods are based on known and accurate QoS values. Obtaining the QoS values is a challenging task and has the following critical drawbacks:

(1) Nowadays, vast novel services are emerging on the internet. It is time-consuming to obtain QoS values by invoking all these large volumes of service candidates.

(2) Some properties of QoS, such as response-time, are related to unpredictable and changing user environment. Measuring these QoS properties requires continuous observation and invocation.

To address this critical challenge, the modern QoS prediction techniques are trying to utilize the limited information to predict approximate QoS values for an individual request.

B. Distributed IoT Service QoS Prediction

Conventional QoS prediction methods usually require that the historical QoS data is centralized [4], [5], [6], [7], [8]. However, in many real-world scenarios, the QoS data may be distributed. Qi et al. [9] considered a distributed situation, in which the historical QoS data is distributed in different platforms, that is, QoS values of the same services collected from many users among multiple platforms. Such a phenomenon in the era of IoT is becoming more widespread. With the rapid increment of IoT and 5G networks, smart mobile devices such as smartphones and autonomous vehicles, have become ubiquitous. To guarantee the low user-facing latency in online service, the Online Service Provider (OSP) deploy edge servers close to end users [10], as shown in Fig. 1. Each edge server is responsible for storing and analyzing the QoS data generated by the users who are located in a specific area. A natural approach is to upload these distributed QoS data to a single giant cloud server, and then process centrally to learn a predictive model. However, QoS data are privacy-sensitive [11]. Due to the increasingly strict confidentiality requirements, some edge servers may be not be allowed to send users' raw QoS data to the central server. On the contrary, if each edge server performs prediction training merely based on its local data, it will lead to a non-precise result due to the insufficient user data and cold-start problem. Therefore,

Xiaoli Li is with the Computer School of HuBei University of Arts and Science, Xiangyang, China, and also with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China (e-mail: lixli27@mail2.sysu.edu.cn). Shixuan Li, Yuzheng Li, Yuren Zhou, Chuan Chen and Zibin Zheng are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. e-mail: {lishx7, liyzh23}@mail2.sysu.edu.cn, {zhouyuren, chenchuan, zhzibin}@mail.sysu.edu.cn.

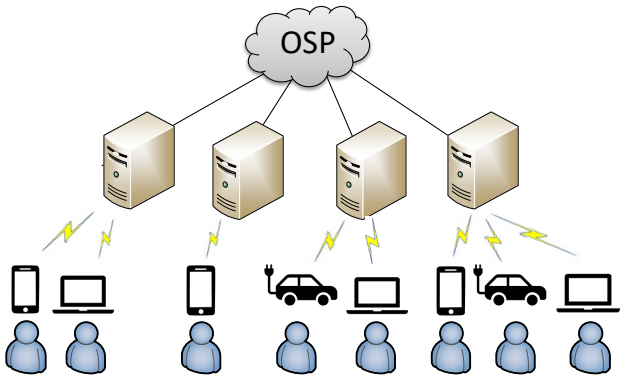


Fig. 1. OSP provides IoT services to users by deploying edge servers close to users, reducing network bandwidth occupation and latency, and enabling users to have a better user experience.

how to learn a prediction model from distributed QoS data via information sharing without infringing user privacy is a challenge.

C. Federated learning

In the past decade, many distributed prediction methods have been proposed for the recommender system. For example, Shmueli et al. [12] proposed protocols of Secure Multiparty Computation (SMC), which can hide private information. However, SMC methods contain complex cryptographic operations, their computational complexity is high, and processing delay is too large. Besides that, Qi et al. [9] introduced amplified Locality-Sensitive Hashing (LSH) and MinHash techniques to calculate the similarity between users across different platforms. And then, Qi et al. [13] proposed a time-aware and privacy-preserving QoS prediction approach, which extends the traditional LSH technique to incorporate the time factor. The LSH approaches can protect the most key private information of users, but they are greatly affected by the sparse data and suffer from cold-start problem. Moreover, because they are memory-based approaches, they cause low scalability.

Recently, federated learning [14] has been proposed, in which the participating clients don't upload their local data to the server, but perform computation based on their local data locally, and then periodically exchange information, so as to train a global model that can be shared collaboratively. Some researchers have proposed to apply Federated Learning to privacy-preserving collaborative data analysis and recommender systems. For example, Chen et al. [15] presented a federated meta-learning framework for the recommendation, which enables information sharing at a higher algorithm level without privacy issues and expansion in model size. Chai et al. [16] proposed a federated matrix factorization framework for the recommendation, and the framework was enhanced with homomorphic encryption to increase security. Duan et al. [17] designed a federated recommendation algorithm, which can provide a more accurate video recommendation service by exploiting the user's profiles. Kim et al. [18] and Ma et al. [19] applied federated tensor factorization for computational phenotyping, in which the hospitals can collaboratively learn meaningful clinical concepts without leakage of patient membership information. Inspired by these works, we apply federated learning to privacy-preserving IoT service QoS prediction. In this paper, our motivation is to enable different edge servers to learn a shared model and provide more accurate QoS prediction, while keeping the QoS data local.

D. Heterogeneity

As shown in Fig. 1, each edge server is in charge of the collecting QoS data from the users in a specific area. The data distribution of different edge servers depends on their users' usages which are likely to be heterogeneous. For example, 1) *Size Heterogeneity*, the number of users on each edge server is uneven; 2) *Scale Heterogeneity*, QoS scale of different edge servers may vary in different ranges, thus each edge server does not follow a common data distribution; 3) *Service Heterogeneity*, the services invoked by users in different edge servers are unbalanced.

For an intuitive understanding, we take the Response Time data (RTdata), a dataset of the real-world QoS dataset *WS-Dream dataset 3* [20], as an example to introduce these heterogeneities in detail. RTdata contains the QoS values of service invocations on *4500* services from *142* users on *64* different time slots. We divide RTdata into *6* clusters according to the QoS values of users, and each edge server takes charge of one cluster of users.

1) *Size Heterogeneity*. Figure 2(a) shows the number of users in each edge server. We can see that the maximum number of users on the edge server is *57*, and the minimum number is *6*. Thus, the user set is extremely unbalanced.

2) *Scale Heterogeneity*. Unlike the subjective values such as movie ratings, QoS values are objective values observed by users in the process of invoking services, which are highly dependent on the underlying network. Users who are geographically close are more likely to share the same IT infrastructure than users who are geographically far away, so they more likely to have similar QoS values or trends [21]. For example, due to the security need or gateway, users in edge server *i* may observe response time more than *3000ms* on all services, while users in edge server *j* with a faster network may observe response time smaller than *200ms* on all services. Thus, the QoS scale of different edge servers is likely to be different. Fig. 2(b) shows the five services with the largest variance ($ws_{305}, ws_{360}, ws_{4250}, ws_{1435}, ws_{2097}$) in the response-time of the six edge servers. We can see that Server5 has the largest QoS scale in these five services, while Server3 has the smallest QoS scale in ws_{305}, ws_{360} and ws_{4250}, and Server 6 has the smallest QoS scale in ws_{1435} and ws_{2097}. Thus, the QoS scale of different edge servers is heterogeneous.

Definition 1: Scale heterogeneity. We quantify the scale heterogeneity of edge server i as the ratio of the QoS values of it to the average QoS values:

$$ScalHete_i = \frac{1}{T} \cdot \sum_{t=0}^{T} \frac{\sum_{i=0}^{m_i} r_{ij}/m_i}{\sum_{i=0}^{m} r_{ij}/m}, \tag{1}$$

(a) Size heterogeneity

(b) Scale heterogeneity

(c) Service heterogeneity

(d) Sparsity and heterogeneity

Fig. 2. The heterogeneities of RTdata: (a) shows the size heterogeneity: the number of users on each edge server is uneven; (b) shows the scale heterogeneity: QoS scale of different edge servers vary in different ranges; (c) shows the service heterogeneity: an example of the class imbalance of the services invoked by users in different edge servers; (d) shows the sparsity and heterogeneity: the value of heterogeneity increases as the density of data decreases.

where T is the number of time slots in RTdata, r_{ij} represents the QoS value of service j observed by user i, m_i represents the number of users of the edge server i, m is the total number of users. And the scale heterogeneity of the whole QoS data is represent as the sum of the weighted scale heterogeneity of each edge server: $ScalHete = \sum_{i=0}^{K} m_i \times ScalHete_i$.

3) *Service Heterogeneity*. If the users of an edge server show a particular preference for a service, the edge server should have more samples of that service. Conversely, if none of the users of an edge server have ever invoked a service, there is no sample of that service in the edge server. Therefore, the distribution of the number of samples per class is different among edge servers. For example, as shown in Fig. 2(c), OSP deployed four edge servers ($\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$, $\mathcal{A}_4$) distributed in different areas. The rows represent the users, and the columns represent the services, a shadow block represents that the corresponding invocation. If the area of edge servers $\mathcal{A}_1$ has a lot of universities, then the users in $\mathcal{A}_1$ are more likely to use teaching service ws_1; if area $\mathcal{A}_2$ has many office buildings, then the users in $\mathcal{A}_2$ prefer business services ws_2; if area $\mathcal{A}_3$ is a tourist destination, then users in this area will use travel service ws_3; while users of $\mathcal{A}_4$ have no obvious preference tendency. The personalized preferences of the users from different edge servers are quite different, the QoS data may be heterogeneous across servers.

Definition 2: Service heterogeneity. For edge server i and j, we quantify the service heterogeneity between them as the ratio of the number of services that edge server i and j have individually invoked but not invoked by each other to the total number of services edge servers i and j have invoked. The definition is as follows: $\left|\overline{W_{it} \cap W_{jt}}\right| / \left|W_{it} \cup W_{jt}\right|$, where W_{it} represents the services invoked by edge servers i in time slot t. Then the service heterogeneity of the dataset of edge server i is as follows:

$$ServHete_i = \frac{1}{K-1} \cdot \frac{1}{T} \cdot \sum_{j \neq i}^{K} \sum_{t=0}^{T} \frac{\left|\overline{W_{it} \cap W_{jt}}\right|}{\left|W_{it} \cup W_{jt}\right|}, \quad (2)$$

where K is the total number of edge servers. And the service heterogeneity of the whole QoS data is represent as the sum of the weighted service heterogeneity of each edge server: $ServHete = \sum_{i=0}^{K} m_i \times ServHete_i$.

Definition 3: Heterogeneity. We take the product of the size heterogeneity and the sum of the scale heterogeneity and the service heterogeneity as the total heterogeneity of an edge server: $Hete_i = m_i/m \times (ScalHete_i + ServHete_i)$. And the heterogeneity of the whole QoS data from distributed edge servers is represent as: $Hete = \sum_{i=0}^{K} m_i \times Hete_i$. A larger value means more heterogeneous.

Sparsity and Heterogeneity. Because each user usually only invokes a few services in a certain period of time, the number

of services invocation is limited, and the QoS data matrix is sparse. We investigated the impact of data sparsity on heterogeneity. We vary the data density from *5* to *50* percent. Fig. 2(d) reports the heterogeneity under different data density. We can see that the data sparseness indeed has a significant influence on heterogeneity. As the density of data decreases, the value of heterogeneity increases.

These heterogeneities will lead to the effect and performance degradation of federated learning algorithm. How to enable the distributed edge servers to train a high-quality centralized model in a decentralized manner under the heterogeneous scenario is a challenge to be solved.

E. Our Motivation

To clarify our motivation, we need to mention two related models. The first one is the uniform global model trained by all the edge servers based on all QoS data via classic federated learning. The second model is the pure local model trained only by one edge server based on its own QoS data.

The uniform global model is trained on the concatenation of all the QoS data distributed in edge servers, it may generalize well on the test data which is similar to the global QoS data distribution; however, it does not perform well for edge servers whose data distributions are very different from the global QoS data distribution. On the other hand, the pure local models are trained on the QoS data of one specific edge server, they match the distribution of corresponding edge servers, however, they do not generalize well for other edge servers due to distribution mismatch. Moreover, due to insufficient user data, the pure local model will lead to problems such as low prediction accuracy and cold-start.

In this paper, our motivation is to enable different edge servers to collaboratively train a shared model and provide more accurate QoS prediction, while keeping the QoS data local. How to train a high-quality centralized model in a decentralized manner when the data distribution is heterogeneous? To address this problem, we propose a personalized federated model for distributed IoT services QoS prediction. We adopt tensors to represent the QoS data with multi-dimensions distributed on different edge servers, and initialize different personalized models for edge servers. Then, each edge server performs local tensor factorization and interacts with the parameter master to learn information from other edge servers during the training process. The personalized federated model is an intermediate model between the pure local models and the uniform global model, which can combine the generalization characteristics of the uniform global model and the distribution matching characteristics of the pure local models, and provide a trade-off between generalization and distribution matching.

Our paper makes the following contributions:

(1) We propose to apply federated learning for distributed IoT service QoS prediction. Federated learning enables different edge servers to learn a shared global model in a decentralized manner, and the edge servers can keep their own QoS data locally. Benefit from federated learning, the cold-start and poor scalability problem can be alleviated.

(2) The heterogeneities of users, services, and scale have been discovered and analyzed on a real-world dataset, and the fact that data sparsity leads to increased heterogeneity is verified.

(3) We adopt tensors to facilitate multi-dimensional QoS data, and design a personalized federated tensor factorization framework for multi-dimensional and heterogeneous distributed QoS data. Our framework balances between generalization and distribution matching, and can find a personalized model that is stylized for each edge server's data.

(4) To evaluate the advantages of our framework, we conduct experiments on a real-world QoS dataset of Web services, and compare our approach with many state-of-the-art methods. The experimental results demonstrate the effectiveness and efficiency of the proposed approach.

The remaining of this paper is organized as follows. Section II surveys related work. We define the problem in Section III. Our method is introduced in Section IV. Experimental results and analysis are summarized in Section V. Finally, we conclude the paper in Section VI.

II. Related work

QoS Prediction. Collaborative Filtering (CF) is one of the most successful QoS prediction techniques. CF-based QoS prediction methods analyze the historical QoS data to predict the unknown QoS values, thus help users find out the appropriate services cost-effectively and efficiently.

Instead of only dealing with the user-service invocations, many QoS prediction methods take advantage of contextual information that goes beyond the user-service matrix, such as time, location, etc. Users who are close to each other are more likely to share the same IT infrastructure (network workload, router, etc.), thus they tend to receive similar objective Web service usage information. By integrating the geographical information, [11], [22] greatly alleviate the problem of data sparsity. The QoS performance of Web services changes over time, by incorporating time information, [20], [6] improve the accuracy of prediction. Ye et al. [23] proposed a robust time-aware personalized QoS prediction method. [21], [7] combined both spatial and temporal information, as the geographically closed user-service pairs may have quite similar QoS trends.

Conventional QoS prediction methods require users to supply their observed QoS values to a central server. However, there is a risk of user privacy leakage. For example, Tang et al. [11] point out that response-time are highly correlated with the users' physical locations, so that the user's location can be inferred from the response-time. The privacy problem has been discussed in several central-CF studies. Randomized Perturbation based approach [24] prevent the leakage of original information by adding noise disturbance to the original data. However, it trades off between privacy and accuracy since adding noise will reduce the accuracy of the prediction. The homomorphic encryption approach [25] carry out computations directly on ciphertexts, which has the same prediction accuracy as non privacy-protection methods. However, this comes at the cost of computational overhead.

Besides the privacy-preserving central-CF methods, some privacy-preserving distributed methods have been proposed.

For example, Shmueli et al. [12] proposed protocols of SMC and Qi et al. [9] [13] introduced the LSH technique. However, they are time-consuming and suffer from cold-start and scalability problems. Different from the above methods, our framework can make full use of both the spatial and temporal information to improve the performance, while keeping the QoS data from different areas locally. In addition, the proposed framework can alleviate the cold-start and poor scalability problem.

Heterogenous Challenges of Distributed QoS Prediction. In a traditional distributed system, the central server has access to all the datasets distributed in sub-systems. Subsequently, the central server split the whole dataset into some subsets with similar distributions for parallel training. However, this is often impractical for distributed Web service QoS prediction. Since the QoS data is privacy sensitive, edge servers may be reluctant to provide their QoS data to the central server due to legal restrictions.

In federated learning, a user's QoS data can only be accessed by the edge server responsible for the user. As the edge servers may be distributed all over the world, users' preferences and IT infrastructures are different. Thus, the edge servers may have local QoS datasets that follow different distributions, i.e., heterogeneity of the number of users, service heterogeneity, and scale heterogeneity.

In the original federated learning algorithm, Federated Averaging algorithm (FedAvg) [26], the local model updates of all clients are simply averaged. Li et al. [27] showed that the heterogeneity of data will slow down the convergence. Zhao et al. [28] showed that the accuracy of FedAvg reduces significantly when local data is heterogeneous and proposed to create a small subset of data which is globally shared between all the clients to increase the test accuracy on highly skewed heterogeneous data. However, a globally shared dataset that consists of a uniform distribution over classes may not always be available. Jeong et al. [29] proposed federated augmentation (FAug), in which the clients train a generative model collectively and augment their local datasets towards yielding IID datasets. FAug needs to guarantee the privacy of the user-generated data which may easily reveal users' privacy sensitive information. Note that sharing or uploading users' local data risks exposing user data privacy, Duan et al. [30] proposed Astraea, a self-balancing federated learning framework, which alleviates the heterogeneities by data augmentation and multi-client rescheduling. However, the mediator will induce high communication overhead. Inspired by FedAvg, Li et al. [31] proposed FedProx, which limits local model updates by penalizing large changes to the current model. It is a simple but efficient framework that can handle heterogeneous federated data, while maintaining the privacy and computational advantages similar to FedAvg.

Personalized QoS Prediction. From the perspective of edge servers, the motivation for participating in federated learning is to reduce the generalization error on their local data with the help of users' data on other edge servers.

The dominant approach to personalized federated learning is local fine-tuning, in which each edge server using its own local data to tune the global model from previous federated training through several gradient descent steps. Jiang et al. [32] pointed out that the setting of Model Agnostic Meta Learning (MAML) has a number of similarities with the objective of personalization for federated learning and interpreted FedAvg as meta-learning for personalization. Fallah et al. [33] built an initial meta-model, Per-FedAvg, to learn personalized models for each client with convergence guarantees. Similar to fine-tuning, the model can be updated effectively with several gradient steps. However, like MAML, the model requires computing the Hessian term during meta-optimization to update the global model, which is computationally prohibitive. Chen et al. [15] proposed federated meta-learning in recommender systems. The main disadvantage of local fine-tuning is that it minimizes the optimization error, whereas neglects the generalization performance of the personalized model, which results in the personalized model is pruned to overfit.

Another approach is mixing the global and local models. Filip et al. [34] introduced a general approach for federated learning by combining the optimization of the local models and the global model via a mixing parameter. Mansour et al. [35] proposed three personalization approaches: user clustering, data interpolation, and model interpolation. User clustering and data interpolation both require some meta-features from all clients, which makes them impractical due to privacy issues in federated learning. Model interpolation, also used in [36], mixes the local model with the global model.

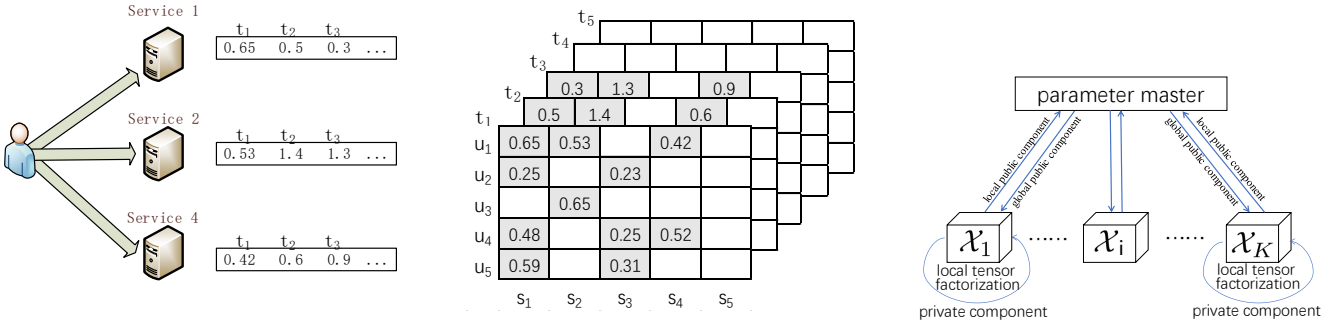
Finally, multi-task learning is also regarded as personalized federal learning. Smith et al. [37] introduced a federated multi-task framework called MOCHA, in which the personalization problem is seen as a multi-task learning problem, and the optimization on each client is considered as a new task.

III. Problem Define

A. QoS Prediction

Figure 3 illustrates a toy example of the QoS prediction problem. In Fig. 3(a), the user u_1 has invoked three IoT services s_1, s_2, and s_4 in the past. u_1 recorded the observed QoS performance of IoT services s_1, s_2, and s_4 with specific invocation time. Fig. 3(b) shows that an edge server form a three-dimensional user-service-time tensor by integrating all the QoS information from the users for whom it is responsible. In this example, there are totally 5 users (from u_1 to u_5), 5 services (from s_1 to s_5), and 5 time intervals (from t_1 to t_5). The tensor is divided into several slices, and each slice represents a time interval. Within the slice, each entry represents the QoS value of the IoT service observed from the user within a specific time interval. It is note worthy that the services set $\{s_1, s_2, ..., s_5\}$ and the time intervals set $\{t_1, t_2, ..., t_5\}$ are the sorted union of sets owned by all edge servers. In Fig. 3(b), the users of the edge server have not invoked service s_5, then the corresponding column is empty. Similarly, if the users of the edge server have not invoked service at time t_4 and t_5, the QoS value of corresponding slices is empty.

Now, we formally define the problem of distributed QoS prediction for IoT services as Fig. 3(c): For simplicity, we assume that there are K edge servers. Given a common IoT



(a) u_1 invoked s_1, s_2, and s_4 in the past (b) The user-service-time tensor of an edge server (c) The framework of distributed QoS prediction

Fig. 3. A toy example of the distributed QoS prediction problem: (a) shows u_1 invoked s_1, s_2, and s_4 in the past; (b) shows a edge server form a three-dimensional user-service-time tensor by integrating all the QoS information from the users for whom it is responsible; (c) shows the framework of distributed QoS prediction.

service set and a common time intervals set, the QoS data at each edge server constructs a locally observed tensor $\mathcal{X}_k$. To predict the missing QoS values, each edge server performs local training based on its own QoS data, and exchange public information with other edge servers through a parameter master, while keeping private information about the users locally.

B. FedAvg

In general, there are two main entities in the FedAvg, i.e., the data owners (participant edge servers) and the model owner (parameter master). Each data owner has a private dataset, and trains a local model based on its dataset. Then the data owner sends only the local model parameters to the parameter master. The parameter master collects local models and averages them to generate a global model.

The goal of classic FedAvg is typically to minimize the following objective function:

$$\min_{w} \mathcal{L} = \sum_{k=1}^{K} p_k F_k(w), \tag{3}$$

where K is the total number of participated edge servers. The edge servers communicate with a parameter master to find a global model w. F_k is the local objective function of the edge server k, it can be defined by empirical risks over local data as follows:

$$F_k(w) = \frac{1}{n_k} \sum_{j_k=1}^{n_k} f_{j_k}(w), \tag{4}$$

where n_k is the number of samples available locally. $p_k = \frac{n_k}{\sum_k n_k}$ is the local weight of F_k. The global model w is trained on the the concatenation of all the samples from all the edge servers and hence is equivalent to minimize the loss on the entire dataset.

C. CP Factorization

A N-way tensor can be rearranged as a matrix; this is called matricization. We denote the mode-n matricization of a tensor $\mathcal{X}$ by $\mathbf{X}_{(n)}$.

Tensor factorization represents the original tensor as a lower dimensional latent matrix. We focus on one of the most well-known tensor factorizations, CANDECOMP/PARAFAC (CP) [38], which can capture multi-linear structure by approximating the original tensor $\mathcal{X}$ to a sum of R rank-one tensors. The rank-R CP tensor factorization is as follows:

$$\mathcal{X} \approx \hat{\mathcal{X}} = \sum_{r=1}^{R} A^{(1)}(:,r) \circ A^{(2)}(:,r) \circ ... \circ A^{(N)}(:,r), \tag{5}$$

where $A^{(n)}$ $(n \in 1, 2..., N)$ is the n-mode factor matrix, and $A^{(n)}(:,r)$ refers to the r-th column of $A^{(n)}$, R is referred as the rank of the $\mathcal{X}$, $\circ$ denotes the outer product.

The objective function of the tensor factorization is to find a set of R normalized rank-one tensors $\{A^{(1)}(:,r) \circ A^{(2)}(:,r) \circ ... \circ A^{(N)}(:,r)\}_{r=1}^{R}$ which best approximates $\mathcal{X}$, i.e., minimizes the Frobenius norm:

$$\begin{aligned} &\min_{\hat{\mathcal{X}}} \mathcal{L} = \left\| \mathcal{X} - \hat{\mathcal{X}} \right\|_F^2 \\ &\text{subject to } \hat{\mathcal{X}} = \sum_{r=1}^{R} A^{(1)}(:,r) \circ ... \circ A^{(N)}(:,r). \end{aligned} \tag{6}$$

IV. Our Method

A. Framework Overview

We adopt tensors to facilitate multi-dimensional QoS data of edge servers. Considering all QoS dimensions uniformly, multi-dimensional QoS data can be modeled well as a multi-dimensional array. We employ an N-dimensional tensor $\mathcal{X}$ to record the historical QoS values, where the $(i_1, i_2, ..., i_N)$ entry of $\mathcal{X}$ is denoted by $x_{i_1 i_2 ... i_N}$. For example, the QoS performance of services is highly related to service invocation time, so the time factor is a critical factor in the QoS pridiction, and can be modeled as a separate dimension. Thus, all these QoS data can form a three-dimensional space: m users, n services, and t periods. The QoS data can be modeled as a tensor $\mathcal{X} \in \mathbb{R}^{m \times n \times t}$. $x_{i_1 i_2 i_3}$ denotes the QoS value for the i_1-th user invoking the i_2-th service at i_3-th time interval.

Then we formally define the overview framework of distributed QoS prediction for IoT services based on federated

tensor factorization as Fig. 3(b). We separate the tensor factorization model into two disjoint parts: private component and public component. The private component consists of users' latent factors that are privacy-sensitive, distinct for different edge servers, and will be kept locally during the federated learning process. While the public component contains services' latent factors which are shared by all edge servers and can be learned collaboratively.

A typical iteration is shown as follows: 1). Edge servers download the global public component from a central parameter master; 2). Each edge server performs local tensor factorization based on its own QoS data to update both private and public components; 3). The edge servers send their local public components to the parameter master; 4) The parameter master aggregates the local public components into a new global public component. As no users' private information will be transmitted to other edge servers, federated tensor factorization can ensure privacy.

B. Personalized Federated Tensor Factorization

We formulate separable objective function on edge servers for federated tensor factorization. For the edge server k, the objective function for tensor factorization in Eq. (6) is reformulated as following:

$$\begin{aligned}&\min_{\hat{\mathcal{X}}_k} F_k = \left\|\mathcal{X}_k - \hat{\mathcal{X}}_k\right\|_F^2 \\ &\text{subject to } \hat{\mathcal{X}}_k = \sum_{r=1}^{R} A_k^{(1)}(:,r) \circ A_k^{(2)}(:,r) \circ ... \circ A_k^{(N)}(:,r).\end{aligned} \tag{7}$$

The edge server k decomposes its local tensor into two disjoint parts: private component $A_k^{(1)}$ and public component $A_k^{(n)}$ $(n = \{2,3,...,N\})$, N is the dimension of tensor $\mathcal{X}_k$.

Multiple edge servers contain different sets of users but the same set of services, time intervals, etc. The horizontal concatenation of the local users' latent factor matrices $A_k^{(1)}$ $(k \in 1,2,...,K)$ forms the global users' latent factor matrix $A^{(1)}$ as follows:

$$A^{(1)} = \begin{bmatrix} A_1^{(1)}; \\ \vdots \\ A_K^{(1)} \end{bmatrix}. \tag{8}$$

Consequently, we represent $\hat{\mathcal{X}}$ in Eq. (6) with respect to $\hat{\mathcal{X}}_k$ as:

$$\hat{\mathcal{X}} = \begin{bmatrix} \hat{\mathcal{X}}_1 \\ \vdots \\ \hat{\mathcal{X}}_K \end{bmatrix} = \begin{bmatrix} \sum_r A_1^{(1)}(:,r) \circ A_1^{(2)}(:,r) \circ ... \circ A_1^{(N)}(:,r) \\ \vdots \\ \sum_r A_K^{(1)}(:,r) \circ A_K^{(2)}(:,r) \circ ... \circ A_K^{(N)}(:,r) \end{bmatrix}. \tag{9}$$

In the conventional federated learning, the public component is shared by all edge servers, and the local services' latent factor matrices of $A_k^{(n)}$ $(n = \{2,3,...,N\})$ from all edge servers are set equal to the global factor services' latent matrices $A^{(n)}$ $(n = \{2,3,...,N\})$:

$$A^{(n)} = A_1^{(n)} = A_2^{(n)} = ... = A_K^{(n)}, (n = \{2,3,...,N\}), \tag{10}$$

where the goal is to find a global public componet $A^{(n)}, n = \{2,\ 3,\ ...\ ,\ N\}$ and some private componets $A_k^{(1)}, k = \{1,\ 2,\ ...\ ,\ K\}$ to minimize the loss on the concatenation of all the samples from all the edge servers, the objective function of Eq. (3) is reformulated as:

$$\begin{aligned}\min_{A^{(n)},A_k^{(1)}} \mathcal{L} &= \sum_{k=1}^{K} p_k F_k(A^{(n)}), (n = \{2,3,...,N\}) \\ &= \sum_{k=1}^{K} p_k \left\|\mathcal{X}_k - \hat{\mathcal{X}}_k\right\|_F^2 \\ \text{subject to } \hat{\mathcal{X}}_k &= \sum_{r=1}^{R} A_k^{(1)}(:,r) \circ A^{(2)}(:,r) ... \circ A^{(N)}(:,r).\end{aligned} \tag{11}$$

Obviously, when the local distribution of edge server k is highly correlated with the global distribution, the global model is preferable. Otherwise, when the local distribution of edge server k be quite different from the global distribution, the edge server k may lead other edge servers toward the optima of its local objective and move further away from the initial global model. In such settings, using the global model as a local model may not be valid for other edge servers. As the QoS data in different edge servers are heterogenous, our personalized federated tensor factorization framework can use the information of the global model to compensate for the small amount of local training data of edge servers, while minimizing the harm caused by the heterogeneity of the local data distribution.

Instead of initializing the same model for each server, we initialize different models for each server seperatelly and introduce an l_2-norm regularized form for each edge server:

$$\begin{aligned}&\min_{\hat{\mathcal{X}}_k} F_k = \left\|\mathcal{X}_k - \hat{\mathcal{X}}_k\right\|_F^2 + \sum_{n=2}^{N} \gamma \left\|A_k^{(n)} - A^{(n)}\right\|_F^2, \\ &\text{subject to } \hat{\mathcal{X}}_k = \sum_{r=1}^{R} A_k^{(1)}(:,r) \circ A_k^{(2)}(:,r) \circ ... \circ A_k^{(N)}(:,r),\end{aligned} \tag{12}$$

where $n = \{2,3,...,N\}$, and γ are regularization parameters that controls the strength of the global model to the personalized models. As the local public components of all edge server should be closed to the global public component, adding a quadratic penalty can adjust the local deviation when the local public components are far away from the global public component. The personalized federated learning can be formulated as:

$$\begin{aligned}\min_{A^{(n)},A_k} \mathcal{L} = \sum_{k=1}^{K} p_k \Big(&\left\|\mathcal{X}_k - \sum_{r=1}^{R} A_k^{(1)}(:,r) \circ ... \circ A_k^{(N)}(:,r)\right\|_F^2 \\ &+ \sum_{n=2}^{N} \gamma \left\|A_k^{(n)} - A^{(n)}\right\|_F^2 \Big), (n = \{2,3,...,N\}).\end{aligned} \tag{13}$$

C. Federated Optimization

Our goal is to find the local personalized models

$A_k = \{A_k^{(1)}, A_k^{(2)}, \ ... \ , A_k^{(N)}\}$, $(k \in \{1, 2, ..., K\})$ and the global model $A^{(n)}$ $(n = \{2, ..., N\})$ that minimize the objective function in Eq. (13). In this paper, we solve the optimization updates using Elastics Averaging Stochastic Gradient Descent (EASGD) [39] approach to solve the optimization problem. EASGD is Stochastic Gradient Descent (SGD) [40] based algorithm, which scales well to sparse tensors as the computation is bounded by the number of non-zeros.

1) Local Personalized Models Update: Each edge server updates the local model (including local private component and public component) by solving the subproblem of Eq. (13).

- Local Private Component. The local private components, users' latent factor matrices, are distinct for different edge server. $A_k^{(1)}$ is the users' latent factor matrix for each edge server k. We write the gradient equation of $A_k^{(1)}$as:
$$\frac{\partial \mathcal{L}}{\partial A_k^{(1)}} = (\hat{\mathbf{X}}_{k(1)} - \mathbf{X}_{k(1)}) A_k^{(-1)}, \tag{14}$$
where $\hat{\mathbf{X}}_{k(1)}$ and $\mathbf{X}_{k(1)}$ are the mode-1 matricization of tensor $\hat{\mathcal{X}}_k$ and $\mathcal{X}_k$ respectively, and $A_k^{(-1)} = A_k^{(N)} \odot ... \odot A_k^{(2)}$. The symbol $\odot$ denotes the Khatri-Rao product [41]. For each edge server k, it keep its own users' latent factors locally, and updates the users' latent factors based on its local dataset as follows:
$$A_k^{(1),t+1} = A_k^{(1),t} - \alpha_t \frac{\partial \mathcal{L}}{\partial A_k^{(1)}}, \tag{15}$$
where α_t is the learning rate at the t-th iteration, which controls the speed of gradient descent iteration.
- Local Public Component. The local public components are services' latent factor matrices, the gradient equation of which is as:
$$\frac{\partial \mathcal{L}}{\partial A_k^{(n)}} = (\hat{\mathbf{X}}_{k(n)} - \mathbf{X}_{k(n)}) A^{(-n)} + \gamma (A_k^{(n)} - A^{(n)}), \tag{16}$$
where $A_k^{(n)}$ $(n = \{2, 3, ..., N\})$ are the local services' latent factor matrices of the edge server k, $A^{(n)}$ $(n = \{2, 3, ..., N\})$ are the global services' latent factor matrices learned by all edge server, and $A^{(-n)} = A^{(N)} \odot ... \odot A^{(n+1)} \odot A^{(n-1)} \odot ... \odot A^{(1)}$ for $n = \{2, 3, ...N\}$.
For each edge server k, it sends its current local services' latent factor matrics $A_k^{(n),t}$ $(n = \{2, 3, ..., N\})$ to the master, and receives the master's global services' latent matrics $A^{(n),t}$ $(n = \{2, 3, ..., N\})$. Then the edge server k performs multiple local tensor factorization based on its local dataset to get an approximate minimum to the objective function in Eq. (13) as follows:
$$A_k^{(n),t+1} = A_k^{(n),t} + \alpha_t \frac{\partial \mathcal{L}}{\partial A^{(n)}}, n \in \{2, ..., N\}. \tag{17}$$

2) Global Model Update: The master broadcasts its current global services' latent matrics
$A^{(n),t}$ $(n = \{2, 3, ..., N\})$ to all edge servers; and receives all local services' latent matrics $A_k^{(n),t}$ $(n = \{2, 3, ..., N\})$ sent by edge servers. Then we update the master's global services' latent matrics $A^{(n),t+1}$ $(n = \{2, 3, ..., N\})$ as follows:
$$\begin{aligned} A^{(n),t+1} &= A^{(n),t} + \alpha_t \frac{\partial \mathcal{L}}{\partial A^{(n)}} \\ &= A^{(n),t} + \alpha_t (\sum_{n=2}^{N} \gamma (A_k^{(n),t} - A^{(n),t})). \end{aligned} \tag{18}$$

Algorithm 1 illustrates the personalized federated tensor factorization.

Algorithm 1 the personalized federated tensor factorization

Input: $A_k^{(n),t}$ $(t = 0, n = \{1, 2, ..., N\}, k = \{1, 2, ..., K\})$, $A^{(n),t}$ $(t = 0, n = \{2, ..., N\})$, $\gamma \succ 0$, α_t.
Output: $A_k^{(1)}$ $(k = \{1, 2, ..., K\})$, $A^{(n)}$ $(n = \{2, ..., N\})$.

for $t = 0$ to T **do**
  //Run on the parameter master:
  Broadcast its current global public component $A^{(n),t}$ $(n = \{2, ..., N\})$ to all edge servers;
  Receive all local public components $A_k^{(n),t}$ $(n = \{2, ..., N\})$ sent by edge servers;
  Update the global public componen $A^{(n),t+1}$ $(n = \{2, ..., N\})$ via Eq. (18).

  //Run on edge server k, (k = {1,2,...,K}):
  Send the current local public component $A_k^{(n),t}$ $(n = \{2, ..., N\})$ to the master;
  Receive the master's global public component $A^{(n),t}$ $(n = \{2, ..., N\})$;
  Update the local private component via Eq.(15);
  Update the local public component via Eq.(17).
end for

V. Evaluation

In this section, we introduce the real-world datasets, the performance metrics, and the compared methods, then report experimental results and parameter analysis.

A. Datasets

We utilize RTdata [1] to validate the proposed framework. We split the 142 users into K edge servers. We model each part of the RTdata as a tensor set $\mathcal{X} \in \mathbb{R}^{m_k \times 4500 \times 64}$, where m_k represents the number of users of the edge server *k*. We randomly removed some values from each edge server, and then used the removed QoS values to evaluate the prediction accuracy of different prediction approaches. Data density is the percentage of unremoved values in these datasets.

Table I shows the heterogeneity of data at different data densities when K is 3, 6, and 10 respectively. In addition, we simulate two different data distributions, IID (Independent and Identically Distributed) and Non-IID (Not Independent and Identically Distributed). In the IID scenario, we randomly assign the same amount of user data to each edge server. In the Non-IID scenario, we cluster RTdata based on users' QoS values, and each cluster represents the QoS data of an edge server.

From Table I, we can see that the heterogeneity increases with the increase of the number of edge servers and the decrease of data density. This is because the larger the K or the lower the density, the smaller the data volume of a single

[1] http://inpluslab.com/wsdream/

TABLE I
THE HETEROGENEITY OF DATA AT DIFFERENT DATA DENSITIES WHEN K IS 3, 6, AND 10.

K	Heterogeneity		Denisty=5%	Denisty=10%	Denisty=15%	Denisty=20%	Denisty=30%	Denisty=50%
3	IID	ServHete	0.035	0.031	0.030	0.029	0.029	0.028
		ScalHete	0.332	0.334	0.334	0.334	0.334	0.334
		Hete	0.367	0.365	0.363	0.363	0.362	0.361
	Non-IID	ServHete	0.307	0.081	0.035	0.026	0.022	0.020
		ScalHete	0.343	0.343	0.343	0.343	0.343	0.343
		Hete	0.650	0.424	0.377	0.368	0.365	0.363
6	IID	ServHete	0.058	0.037	0.027	0.024	0.021	0.018
		ScalHete	0.168	0.168	0.168	0.168	0.168	0.168
		Hete	0.225	0.204	0.195	0.192	0.189	0.186
	Non-IID	ServHete	0.644	0.405	0.269	0.190	0.112	0.063
		ScalHete	0.157	0.155	0.154	0.154	0.153	0.153
		Hete	0.802	0.560	0.424	0.344	0.265	0.215
10	IID	ServHete	0.737	0.507	0.335	0.218	0.103	0.049
		ScalHete	0.100	0.100	0.100	0.100	0.100	0.100
		Hete	0.750	0.544	0.390	0.286	0.183	0.136
	Non-IID	ServHete	0.772	0.597	0.475	0.390	0.279	0.169
		ScalHete	0.140	0.138	0.137	0.137	0.136	0.135
		Hete	0.884	0.711	0.590	0.508	0.399	0.294

edge server, and the smaller the number of services that any two edge servers have invoked in common, resulting in greater service heterogeneity.

B. Accuracy Metrics

Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics are used to measure the prediction QoS values of our method. MAE is defined as

$$MAE = \frac{\sum i,j \left|(r_{ij} - \hat{r}_{ij})\right|}{\rho}, \tag{19}$$

where $\hat{r}_{ij}$ is the predicted QoS value, and ρ is the number of predicted values in test data. RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum i,j (r_{ij} - \hat{r}_{ij})^2}{\rho}}. \tag{20}$$

In RMSE, the errors are squared before they are averaged, thus the RMSE gives a relatively high weight to large errors.

C. Compared methods

To study the prediction performance, we compare our approach (named as PerFed) with other well-known approaches: HDOP [42], SerRec$_{time-LSH}$ [13], and FedAvg [14].

HDOP. HDOP is a central tensor factorization method, which requires all data logging into a server.

CP-alone. CP-alone means pure local models that the edge servers train on their own datasets via CP, and do not cooperate with others.

SerRec$_{\text{time-LSH}}$. SerRec$_{time-LSH}$ is a time-aware and privacy-preserving service recommendation approach based on Locality-Sensitive Hashing (LSH). And it is designed for distributed QoS prediction.

FedAvg. FedAvg is the most used method in federated learning, in which the master combines all the local updates by taking the average.

PerFed. Our framework, personalized federated (PerFed) model, is an intermediate model between the pure local models and the uniform global model, which can find a personalized model that is stylized for each edge server's data.

D. Performance Comparisons

In order to validate the feasibility of our proposal in terms of accuracy, convergence, efficiency, and privacy-preservation.

1) Accuracy: Table II shows the MAE and RMSE results of different prediction methods on RTdata under the different proportions of training data. It should be noted that the prediction results of method $SerRec_{time\text{-}LSH}$ are actually the same under the different number of edge servers, because the QoS values data in each edge server are independently hashed offline, and the number of edge servers does not substantially affect the final results [43].

As indicated in Table II, the prediction accuracy of CP-alone is the lowest when K is 6 and 10 (i.e., both MAE and RMSE are the highest). This is due to insufficient user data and cold-start problems, which lead to over-fitting of the purely local model on the training set, resulting in low prediction accuracy. However, when K is 3, as each edge server has a larger amount of data, the accuracy of CP-alone is similar to that of SerRec$_{time\text{-}LSH}$, or even better. The prediction accuracy of HDOP is the highest, as it is a conventional central tensor factorization method, which puts all data together to train a model. HDOP employs all QoS

TABLE II
THE MAE AND RMSE RESULTS OF DIFFERENT PREDICTION METHODS ON RTDATA UNDER DIFFERENT PROPORTION OF TRAINING DATA: PERFED PERFORMS BETTER THAN CP-ALONE, SERREC$_{time\text{-}LSH}$, AND FEDAVG. BESIDES, THE ACCURACIES OF PERFED IS VERY CLOSE TO THE PERFORMANCE OF HDOP.

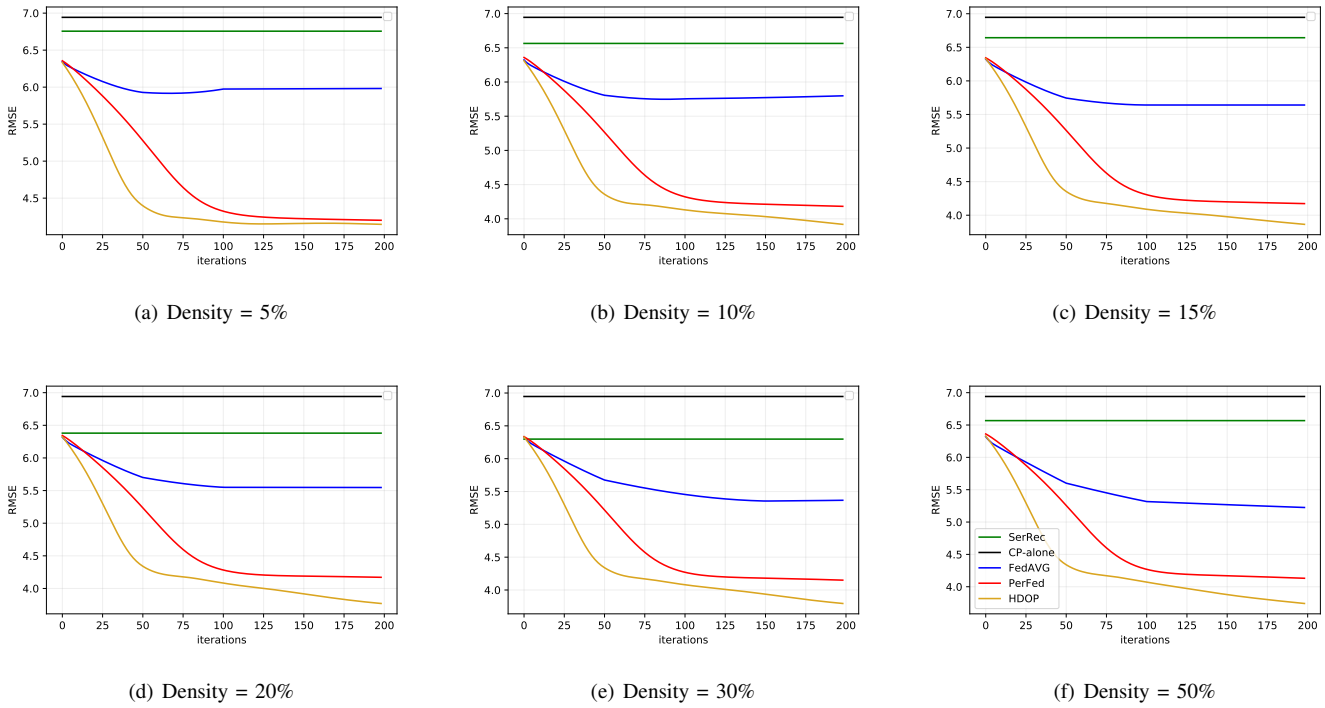
K	Heterogeneity	Methods	Denisty=5%		Denisty=10%		Denisty=15%		Denisty=20%		Denisty=30%		Denisty=50%	
			MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
3	IID	CP-alone	2.907	6.354	2.918	6.287	2.894	6.289	2.923	6.254	2.931	6.245	2.901	6.268
		FedAvg	2.744	5.901	2.74	5.958	2.701	5.866	2.763	6.034	2.729	5.95	2.683	5.818
		PerFed	**2.702**	**5.72**	**2.672**	**5.637**	**2.652**	**5.584**	**2.657**	**5.604**	**2.653**	**5.556**	**2.651**	**5.561**
	Non-IID	CP-alone	3.129	5.716	3.117	5.727	3.118	5.755	3.106	5.72	3.111	5.698	3.118	5.741
		FedAvg	2.388	4.292	2.372	4.291	2.14	4.102	2.137	4.069	2.121	4.036	2.105	4.034
		PerFed	**2.378**	**4.179**	**2.222**	**4.28**	**2.125**	**4.056**	**2.129**	**4.04**	**2.104**	**4.025**	**2.088**	**4.015**
6	IID	CP-alone	3.761	7.094	3.422	6.825	3.683	6.774	3.409	6.525	3.416	6.524	3.417	6.528
		FedAvg	3.008	6.145	2.949	5.983	2.967	6.091	2.97	6.13	2.954	6.073	2.93	6.054
		PerFed	**2.913**	**5.642**	**2.853**	**5.484**	**2.844**	**5.465**	**2.828**	**5.492**	**2.832**	**5.487**	**2.82**	**5.52**
	Non-IID	CP-alone	3.27	6.943	3.727	6.945	3.273	6.946	3.27	6.942	3.273	6.947	3.269	6.941
		FedAvg	2.406	4.261	2.393	4.235	2.27	4.133	2.249	4.094	2.243	4.123	2.229	4.089
		PerFed	**2.384**	**4.194**	**2.308**	**4.076**	**2.225**	**3.985**	**2.233**	**3.974**	**2.211**	**3.936**	**2.217**	**3.916**
10	IID	CP-alone	3.387	6.999	3.386	6.984	3.383	6.982	3.373	6.989	3.38	6.979	3.385	6.99
		FedAvg	2.383	4.322	2.405	4.395	2.376	4.48	2.365	4.247	2.372	4.353	2.306	4.148
		PerFed	**2.364**	**4.228**	**2.404**	**4.255**	**2.365**	**4.226**	**2.363**	**4.191**	**2.369**	**4.185**	**2.304**	**4.097**
	Non-IID	CP-alone	3.28	7.051	3.266	7.021	3.247	7.039	3.236	7.013	3.239	7.015	3.228	7.035
		FedAvg	2.503	6.155	2.505	6.008	2.604	5.929	2.731	5.871	3.02	5.83	2.553	4.794
		PerFed	**2.294**	**4.393**	**2.245**	**4.292**	**2.225**	**4.254**	**2.216**	**4.212**	**2.168**	**4.181**	**2.159**	**4.184**
$SerRec_{time\text{-}LSH}$			3.07	6.756	3.076	6.564	2.9	6.643	2.843	6.38	2.771	6.298	3.191	6.568
HDOP			2.24	4.138	2.151	3.919	2.034	3.773	2.053	3.77	2.067	3.795	2.027	3.741

value information (global information) to construct a global model for making value prediction, it is good at estimating the overall structure that relates simultaneously to all users. SerRec$_{time\text{-}LSH}$ performs better than CP-alone when K is 6 and 10, as it achieves lower MAE and RMSE than CP-alone. This is because SerRec$_{time\text{-}LSH}$ alleviates the cold-start problem via integrating the distributed QoS data. These two federated learning methods, FedAvg and PerFed, both achieve lower MAE and RMSE than SerRec$_{time\text{-}LSH}$. This is because the key step of SerRec$_{time\text{-}LSH}$ is to find out similar neighbors for each user by leveraging users' historic QoS values. It just makes use of local information but leads to the loss of global structure, and it is greatly affected by the sparse data problem. Besides, the accuracies of PerFed are close to the performance of HDOP. This validates the effectiveness of the federated learning methods to the problem of distributed QoS prediction. In the two federated learning methods, PerFed performs better than FedAvg. This is because FedAvg doesn't consider heterogeneous data, and PerFed can effectively avoid the global model following the local deviation, which is more suitable for the sparse and heterogeneous data.

2) Convergence: Fig. 4 shows the RMSE in terms of iterations on RTdata with different proportions of training data when K is 6. And we can see that the RMSE of CP-alone is almost constant, that is because the number of features is constant, the information the algorithm can learn from it is capped. SerRec$_{time\text{-}LSH}$ performs better than CP-alone. FedAvg performs better than SerRec$_{time\text{-}LSH}$. However, we can see that from Fig. 4(f) to Fig. 4(a), the curve of FedAvg is smoother. As mentioned in Section 1.4, the heterogeneity increases with the decrease of data density. Thus, this indicates that the heterogeneity of data will slow down the convergence of FedAvg. PerFed get faster convergence than FedAvg, and the convergence rate is close to that of the centralized method. The experimental results show that PerFed can achieve higher accuracy and stable performance than FedAvg. Comparing Fig. 4(a) with Fig. 4(f), it is found that the curve of PerFed in Fig. 4(a) is closer to the curve of HDOP than that in Fig. 4(f). This shows that the proposed PerFed performs better when the data is more sparse, that is, the data is more heterogeneous.

3) Efficiency: PerFed uses an iterative approach which requires some global iterations to achieve a global accuracy level. Each global iteration consists of two major stages: computation and communication. In the computation phase, all edge servers solve their local problems, and in the communication phase, they transmit their updates to the master. We note that the time of the master aggregates the local model updates is much smaller than the local training time of the edge server, thus model update time is not considered in this work. Similarly, since the downlink bandwidth is larger than uplink bandwidth, the downlink time is negligible compared to uplink time. Therefore, the total runtime can be formulated as: $T_{glob} = T * (T_{com} + T_{cmp})$, where T denotes the total number of global training iterations, and T_{com}, T_{cmp} refer to the time taken by communication and computation, respectively.

The main computation of PerFed is to evaluate the gradients.



(a) Density = 5% (b) Density = 10% (c) Density = 15%

(d) Density = 20% (e) Density = 30% (f) Density = 50%

Fig. 4. The RMSE in terms of iterations on RTdata under the different proportions of training data: the RMSE of CP-alone is almost constant, the heterogeneity of data will slow down the convergence of FedAvg, PerFed get faster convergence than FedAvg, and the convergence rate is close to that of the centralized method HDOP.

Because of the sparsity of QoS data, the computational complexity of evaluating the gradients is $O(\rho_{max}R(m+n+t))$ for each iteration, where ρ_{max} is the maximum number of available items of QoS data of all edge servers, R is the rank, and m, n, t are three dimensions: m users, n services, t time periods. The communication time is equal to the amount of data transferred s divided by the transmission rate r: $T_{com} = T*s/r$. The data size of each edge server needed to transfer to the master is: $s_k = R(m+n+t)$. In summary, the global time of PerFed is: $O(T*\rho_{max}R(m+n+t))+O(T*R(m+n+t)/r)$.

FedAvg, like PerFed, is a federated learning algorithm, which requires the same time as PerFed. The computation time of CP-alone is the same as that of PerFed, but it does not have the step of communication transmission. However, in view of its poor effect, we will not compare it with its efficiency.

As for the central method HDOP, which require all edge server send their raw data to the master, the main computation is $O(\rho_{total}R(m+n+t))$, where ρ_{total} is the total number of available items of QoS data of all edge servers. And the communication time is $O(mnt/r)$, thus the global time is as: $O(\rho_{total}R(m+n+t))+O(R(mnt/r)$. As for SerRec$_{time\text{-}LSH}$, the main computation is the similarity computation, which is $O(m^2nt)$. And the data needed to transfer is the hashing table, which is equal to ρ_{total}, thus the communication time is ρ_{total}/r.

Figure. 5 shows the global time costs of the three methods under different transmission speed when K is 6. PerFed-50, PerFed-100, and PerFed-150 represent PerFed with a different number of global training iterations. We can see that while the time cost of PerFed-150 is a little higher than that of

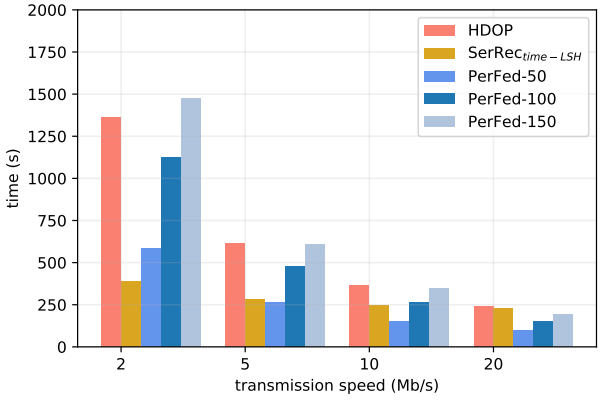
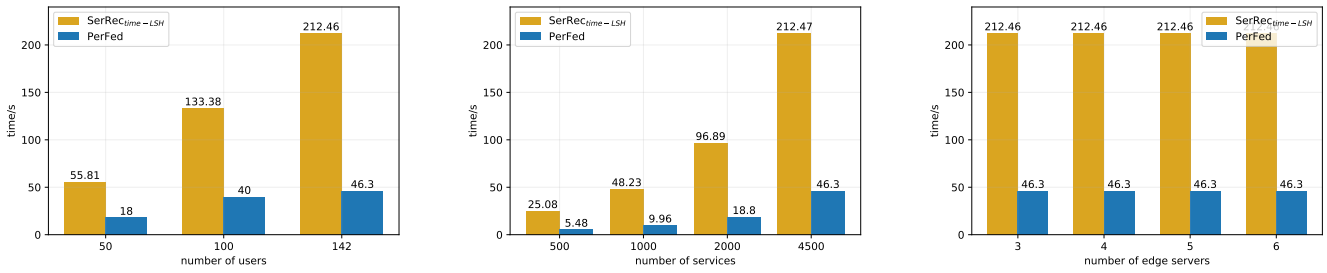


Fig. 5. Global time costs under different transmission speed.

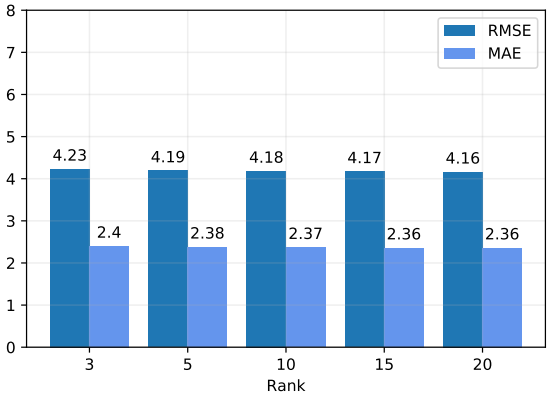
HDOP, the time costs of PerFed-50 and PerFed-100 are lower than that of HDOP. From Fig. 4, we can see that the best results can be achieved in 100-125 rounds generally. This means that Perfed has a lower time cost than HDOP. On the other hand, the time cost of PerFed-100 is larger than that of SerRec$_{time\text{-}LSH}$ when the transmission speed is below $5Mb/s$. However, when the transmission speed is greater than $10Mb/s$, the time cost of PerFed-100 is close to or even lower than that of SerRec$_{time\text{-}LSH}$.

4) Privacy: We think there are three categories of information that should be considered private in QoS recommenders: (1) QoS Values: the true QoS values of services; (2) Invoking

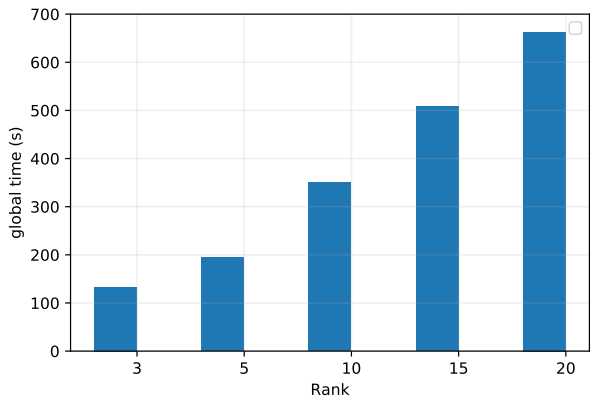


(a) Compute time with the number of users (b) Compute time with the number of services (c) Compute time with the the number of servers

Fig. 6. The compute time of PerFed and SerRec$_{time\text{-}LSH}$ with the growth of the number of users, services and edge servers.

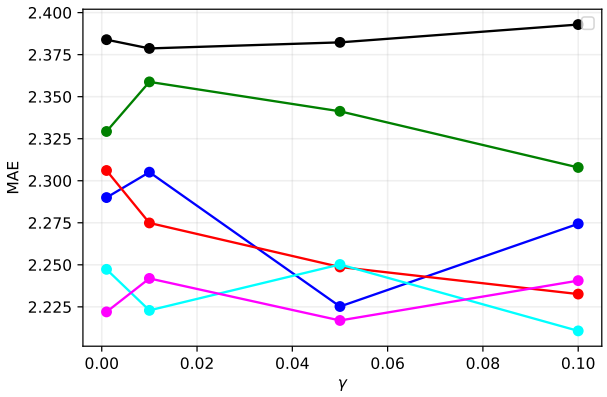


(a) The accuracy under different R

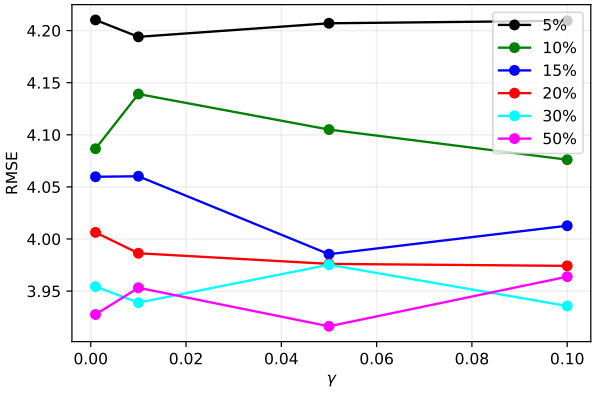


(b) The global time under different R

Fig. 7. The impact of parameter R.



(a) The MAE under different γ



(b) The RMSE time under different γ

Fig. 8. The impact of parameter γ.

Existence: whether a user invokes a service or not; (3) Trained Model: Models are important, since given the model, attackers can estimate the QoS values from any user to any service. We consider the Honest-But-Curious adversary model [44] where malicious nodes can collude to predict interests but cannot cheat in the protocol. HDOP is centralized, thus it's difficult to guarantee the fairness and independence of the central server. SerRec$_{time\text{-}LSH}$ eliminates the use of a centralized server, and can protect the most key private information of users and only publish the less sensitive user index table to the public. However, curious edge servers may be able to snoop on the personal information of users on other edge servers. For example, curious users can extract the information of interest to the users from the profiles exchanged with other edge servers.

TABLE III
THE PRIVACY COMPARISON OF THE THREE METHODS: PERFED CAN AVOID QOS VALUE, INVOKING EXISTENCE, AND TRAINED MODEL LEAKAGE.

Methods	QoS Values	Invoking Existence	Trained Model
HDOP	×	×	×
SerRec$_{time\text{-}LSH}$	✓	×	×
PerFed	✓	✓	✓

Moreover, SerRec$_{time\text{-}LSH}$ has to utilize the actual QoS values of similar friends to make QoS prediction. This may result in privacy leakage risks. The proposed framework PerFed separates traditional tensor factorization into private components and public components. The edge servers maintain their QoS values and train their private components independently and locally. They only send their public components to the master. Intuitively, such a setting is able to avoid QoS values leakage and no users' personal information will be transmitted to other edge servers. The master only receives public components and it is difficult for the master to recover original QoS values nor private information from the public components. Table II shows the privacy comparison of the three methods. ✓ indicates that the corresponding information can be protected, whereas × indicates that the corresponding information may be leaked. We can see that PerFed can avoid QoS value, invoking existence, and trained model leakage.

E. Parameter Analysis

1) Impact of Scalability: In order to demonstrate the impact of scalability in a distributed situation, we compare the compute time costs of PerFed with SerRec$_{time\text{-}LSH}$, both of the two methods are suitable for distributed QoS prediction. We separate the scalability into three situations: a) The number of users per edge server is increasing; b) The number of services provided by servers is expanding; c) the number of edge servers deployed by OSP is broadening. Fig. 6(a), Fig. 6(b), and Fig. 6(c) show the compute time costs of the two methods with the growth of the number of users, services, and edge servers, respectively. In Fig. 6(a), we set the number of edge servers and services are 6 and 4500, respectively. And the number of users per edge server varies from 50 to 142. We cluster the selected users into 6 groups. In Fig. 6(b), we set the number of edge servers and users are 6 and 142, respectively. We cluster the 142 users into 6 groups, and randomly remove the QoS data corresponding to a certain amount of services. In Fig. 6(c), we keep the number of services and users unchanged as 142 and 4500 respectively. And we cluster the users into 6 edge servers, and then select some edge servers to train. We can see that the time cost of PerFed is much smaller than the cost of SerRec$_{time\text{-}LSH}$ in all the three situations. In Fig. 6(c), because all the edge servers are training parallelly, the computing time depends on the slowest one, all the sets of edge servers we selected include the slowest one, so the overall time is unchanged.

2) Impact of R: According to Section 3.3, R is the rank of the tensor. To study the impact of parameter R, we vary the values of R from 3 to 20. Fig. 7 shows the accuracy and global time of our method with 5% training data on RTdata. As shown in Fig. 7(a), there is a small decrease in the values of MAE and RMSE when the value of R is increased, indicating that a bigger value of R generates more accurate prediction results. And Fig. 7(b) shows that the global time increases significantly with the increase of R. We try to select a small R with high prediction accuracy since a smaller R can reduce the global time. In our framework, we set R to be 5.

3) Impact of γ: As mentioned in Section 4.2, γ controls the proportions of the global model to the personalized models of the loss function $\mathcal{L}$. We set rank = 5 in this experiment and vary the hyper-parameter γ from 0.001 to 0.1. Fig. 8 and Table III show the impact of parameter γ on the prediction results under different data densities. We observe that there is little difference in the value of MAE and RMSE under different γ, and all the values of γ can obtain a good prediction effect. The accuracy of PerFed increases (i.e., MAE and RMSE of PerFed is decreased) with the decrease of heterogeneity (i.e., the increase of data density). This is a normal phenomenon, suggesting that the heterogeneity does indeed have a negative impact on the predicted accuracy.

VI. CONCLUSIONS AND FUTURE WORK

In the distributed environment, it is very important to be able to integrate QoS data distributed on multiple edge servers while protecting users' privacy information. Besides, the QoS data are often sparse, and heterogeneous. In view of these challenges, we design a distributed privacy-preserving prediction method. We utilize tensor to facilitate multi-dimensional QoS data, and propose a personalized federated tensor factorization framework to alleviate the heterogeneities by the trade-off between generalization and distribution matching. We conduct experiments on a real-world QoS dataset of Web services, and the experimental results validate the effectiveness and efficiency of our proposed framework.

Although PerFed is efficient and accurate, it has some limitations in the practice of the communication delay. Thus, in the future, we will investigate asynchronous update strategies to perform an elastic aggregate of the local models.

ACKNOWLEDGMENT

The research is supported by the National Key R&D Program of China (2020YFB1006001), the National Natural Science Foundation of China (11801595), the Guangdong Basic and Applied Basic Research Foundation (2019A1515011043), the Natural Science Foundation of Guangdong (2018A030310076) and the Innovative Research Foundation of Ship General Performance (25622112).

REFERENCES

[1] A. Whitmore, A. Agarwal, and L. D. Xu, "The internet of things - A survey of topics and trends," *Inf. Syst. Frontiers*, vol. 17, no. 2, pp. 261–274, 2015.

[2] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, "Context-aware qos prediction with neural collaborative filtering for internet-of-things services," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4532–4542, 2020. [Online]. Available: https://doi.org/10.1109/JIOT.2019.2956827

[3] M. E. Khanouche, S. Mouloudj, and M. Hammoum, "Two-steps qos-aware services composition algorithm for internet of things," in *Proceedings of the 3rd International Conference on Future Networks and Distributed Systems, ICFNDS 2019, Paris, France, July 01-02, 2019*. ACM, 2019, pp. 26:1–26:6.

[4] M. Tang, W. Liang, Y. Yang, and J. Xie, "A factorization machine-based qos prediction approach for mobile service selection," *IEEE Access*, vol. 7, pp. 32 961–32 970, 2019.

[5] L. Chen, F. Xie, Z. Zheng, and Y. Wu, "Predicting quality of service via leveraging location information," *Complexity*, vol. 2019, 2019.

[6] S. Li, J. Wen, F. Luo, and G. Ranzi, "Time-aware qos prediction for cloud service recommendation based on matrix factorization," *IEEE Access*, vol. 6, pp. 77 716–77 724, 2018.

[7] Q. Zhou, H. Wu, K. Yue, and C.-H. Hsu, "Spatio-temporal context-aware collaborative qos prediction," *Future Generation Computer Systems*, 2019.

[8] S. Ding, Y. Li, D. Wu, Y. Zhang, and S. Yang, "Time-aware cloud service recommendation using similarity-enhanced collaborative filtering and arima model," *Decision Support Systems*, vol. 107, pp. 103–115, 2018.

[9] L. Qi, X. Zhang, W. Dou, C. Hu, C. Yang, and J. Chen, "A two-stage locality-sensitive hashing based approach for privacy-preserving mobile service recommendation in cross-platform edge environment," *Future Generation Computer Systems*, vol. 88, pp. 636–643, 2018.

[10] H. Yin, X. Zhang, H. H. Liu, Y. Luo, C. Tian, S. Zhao, and F. Li, "Edge provisioning with flexible server placement," *IEEE Trans. Parallel Distributed Syst.*, vol. 28, no. 4, pp. 1031–1045, 2017.

[11] M. Tang, Y. Jiang, J. Liu, and X. F. Liu, "Location-aware collaborative filtering for qos-based service recommendation," in *2012 IEEE 19th International Conference on Web Services*. IEEE, 2012, pp. 202–209.

[12] E. Shmueli and T. Tassa, "Secure multi-party protocols for item-based collaborative filtering," in *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, 2017, pp. 89–97.

[13] L. Qi, R. Wang, C. Hu, S. Li, Q. He, and X. Xu, "Time-aware distributed service recommendation with privacy-preservation," *Information Sciences*, vol. 480, pp. 354–364, 2019.

[14] J. Konecný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *CoRR*, vol. abs/1610.02527, 2016. [Online]. Available: http://arxiv.org/abs/1610.02527

[15] F. Chen, Z. Dong, Z. Li, and X. He, "Federated meta-learning for recommendation," *CoRR*, vol. abs/1802.07876, 2018.

[16] D. Chai, L. Wang, K. Chen, and Q. Yang, "Secure federated matrix factorization," *CoRR*, vol. abs/1906.05108, 2019.

[17] S. Duan, D. Zhang, Y. Wang, L. Li, and Y. Zhang, "Jointrec: A deep-learning-based joint cloud video recommendation framework for mobile iot," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1655–1666, 2020.

[18] Y. Kim, J. Sun, H. Yu, and X. Jiang, "Federated tensor factorization for computational phenotyping," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada*. ACM, 2017, pp. 887–895.

[19] J. Ma, Q. Zhang, J. Lou, J. C. Ho, L. Xiong, and X. Jiang, "Privacy-preserving tensor factorization for collaborative health data analysis," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China*. ACM, 2019, pp. 1291–1300.

[20] Y. Zhang, Z. Zheng, and M. R. Lyu, "Wspred: A time-aware personalized qos prediction framework for web services," in *IEEE 22nd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2011, pp. 210–219.

[21] X. Wang, J. Zhu, Z. Zheng, W. Song, Y. Shen, and M. R. Lyu, "A spatial-temporal qos prediction approach for time-aware web service recommendation," *ACM Transactions on the Web (TWEB)*, vol. 10, no. 1, p. 7, 2016.

[22] J. Liu, M. Tang, Z. Zheng, X. F. Liu, and S. Lyu, "Location-aware and personalized collaborative filtering for web service recommendation," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 686–699, 2016.

[23] F. Ye, Z. Lin, C. Chen, Z. Zheng, H. Huang, and E. Yilmaz, "Outlier resilient collaborative web service qos prediction," in *The Web Conference (WWW)*, 2021.

[24] X. Zhu, X.-Y. Jing, D. Wu, Z. He, J. Cao, D. Yue, and L. Wang, "Similarity-maintaining privacy preservation and location-aware low-rank matrix factorization for qos prediction based web service recommendation," *IEEE Transactions on Services Computing*, 2018.

[25] S. Badsha, X. Yi, I. Khalil, D. Liu, S. Nepal, E. Bertino, and K.-Y. Lam, "Privacy preserving location-aware personalized web service recommendations," *IEEE Transactions on Services Computing*, 2018.

[26] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, 2017, pp. 1273–1282.

[27] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=HJxNAnVtDS

[28] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *CoRR*, vol. abs/1806.00582, 2018.

[29] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *CoRR*, vol. abs/1811.11479, 2018.

[30] M. Duan, D. Liu, X. Chen, Y. Tan, J. Ren, L. Qiao, and L. Liang, "Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications," in *37th IEEE International Conference on Computer Design, ICCD 2019, Abu Dhabi, United Arab Emirates, November 17-20, 2019*. IEEE, 2019, pp. 246–254. [Online]. Available: https://doi.org/10.1109/ICCD46524.2019.00038

[31] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*, I. S. Dhillon, D. S. Papailiopoulos, and V. Sze, Eds. mlsys.org, 2020. [Online]. Available: https://proceedings.mlsys.org/book/316.pdf

[32] Y. Jiang, J. Konecný, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," *CoRR*, vol. abs/1909.12488, 2019. [Online]. Available: http://arxiv.org/abs/1909.12488

[33] A. Fallah, A. Mokhtari, and A. E. Ozdaglar, "Personalized federated learning: A meta-learning approach," *CoRR*, vol. abs/2002.07948, 2020. [Online]. Available: https://arxiv.org/abs/2002.07948

[34] F. Hanzely and P. Richtárik, "Federated learning of a mixture of global and local models," *CoRR*, vol. abs/2002.05516, 2020.

[35] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," *CoRR*, vol. abs/2002.10619, 2020. [Online]. Available: https://arxiv.org/abs/2002.10619

[36] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," *CoRR*, vol. abs/2003.13461, 2020. [Online]. Available: https://arxiv.org/abs/2003.13461

[37] V. Smith, C. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 4424–4434.

[38] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *Siam Review*, vol. 51, no. 3, pp. 455–500, 2009.

[39] S. Zhang, A. Choromanska, and Y. LeCun, "Deep learning with elastic averaging SGD," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems, Quebec, Canada*, 2015, pp. 685–693.

[40] G. Mann, R. T. McDonald, M. Mohri, N. Silberman, and D. Walker, "Efficient large-scale distributed training of conditional maximum entropy models," in *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing System, British Columbia, Canada*, 2009, pp. 1231–1239.

[41] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup, "Scalable tensor factorizations with missing data," in *Proceedings of the SIAM International Conference on Data Mining, Columbus, Ohio, USA*, 2010.

[42] S. Wang, Y. Ma, B. Cheng, R. Chang *et al.*, "Multi-dimensional qos prediction for service recommendations," *IEEE Transactions on Services Computing*, vol. 12, no. 1, pp. 47–57, 2019.

[43] L. Qi, X. Zhang, W. Dou, and Q. Ni, "A distributed locality-sensitive hashing-based approach for cloud service recommendation from multi-source data," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2616–2624, 2017.

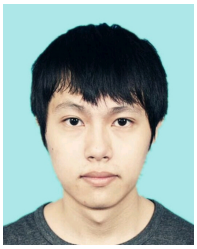
[44] O. Goldreich, "Cryptography and cryptographic protocols," *Distrib. Comput.*, vol. 16, no. 2–3, p. 177–199, Sep. 2003. [Online]. Available: https://doi.org/10.1007/s00446-002-0077-1

Xiaoli Li received the Master's degree in computer architecture from University of Electronic Science and Technology of China, Chengdou, China, in 2011, and the Ph.D. degree from the Sun Yat-sen University, Guangdong, China, in 2022. She is currently an Associate Professor with the Computer school of HuBei University of Arts and Science. Her research interests include services computing, software engineering, cloud computing, machine learning and federated learning.

Shixuan Li received the Master's degree in the School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou, China, in 2021. She received a bachelor degree in Netwook Engineering from Sun Yat-Sen University, Guangzhou, China, in 2019. Her research interests include machine learning, federated learning, and software engineering.

Yuzheng Li received the B.E. degrees from the Sun Yat-sen University, Guangdong, China, in 2018, and the Master's degree in the School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou, China, in 2021. His current research interests include federated learning, blockchain, statistical machine learning, multi-view learning and optimization.

Yuren Zhou received the B.Sc. degree in mathematics from Peking University, Beijing, China, in 1988, and the M.Sc. degree in mathematics and the Ph.D. degree in computer science from Wuhan University, Wuhan, China, in 1991 and 2003, respectively.,He is currently a Professor with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. His current research interests include design and analysis of algorithms, evolutionary computation, and social networks.



Chuan Chen received the B.S. degree from Sun Yat-sen University, Guangzhou, China, in 2012, and the Ph.D. degree from Hong Kong Baptist University, Hong Kong, in 2016. He is currently an Associate Professor with the School of Computer Science and Engineering, Sun Yat-Sen University. He published over 50 international journal and conference papers. His current research interests include machine learning, numerical linear algebra, and numerical optimization.

Zibin Zheng received his PhD degree from the Chinese University of Hong Kong in 2011. He is currently a professor in the School of Computer Science and Engineering at Sun Yat-sen University, China. He has published over 150 international journal and conference papers, including three ESI highly cited papers. According to Google Scholar, his papers have more than 13,590 citations, with an H-index of 54. His research interests include blockchain, smart contract, services computing, software reliability.