Federated Learning-driven Trust Prediction for Mobile Edge Computing-based IoT Systems

Jiahui Bai, Hai Dong[§]
School of Computing Technologies, RMIT University, Melbourne, Australia
Email: s3735049@student.rmit.edu.au, hai.dong@rmit.edu.au

***Abstract*—We propose a federated learning-based data-driven trust prediction method to meet the demand of high-accuracy IoT service trustworthiness prediction in Mobile Edge Computing (MEC) with low convergence time. Our research focuses on the mixture distribution and heterogeneity features of IoT trust information in distributed MEC environments and formulates the task of distributed IoT trust prediction on top of MEC network topologies as a federated optimization problem. We then employ Federated Expectation-Maximization to mitigate the federated optimization problem by taking into account the data mixture distribution and heterogeneity. We conduct a series of experiments upon simulated MEC-based IoT environments crafted on top of a real-world IoT dataset. The experimental results show that our proposed methods can achieve better balance between prediction accuracy and model training efficiency than a state-of-the-art data-driven MEC-based IoT service trust prediction method and a Federated Averaging-based method.**

***Index Terms*—IoT Service Trust, Data-driven Trust Prediction, Mobile Edge Computing-based IoT Systems, Federated Expectation-Maximization**

I. Introduction

IoT services base on cloud computing and IoT technology provides data collection, storage, analysis and application services through connecting various devices, sensors and systems. Trust of an Internet of Things (IoT) service is used to measure the belief how much a service consumer believes a service provider can deliver the requested IoT service [1]. Trust is a critical prerequisite for IoT to provide safe and reliable services [2]. Untrustworthy IoT devices may provide low-quality even malicious services, which can generate high risks for service consumers.

There are various ways to measure service trust in IoT systems. For example, trust values can be generated upon service consumers' subjective evaluation and quality of service (QoS) measured by third parties [3]. These types of information can be used to measure service trust in terms of certain trust attributes, such as availability, reliability, risk, etc. These trust attributes can be aggregated to model the trustworthiness of IoT services in a specific application context. However, the *domain-specific modelling of IoT service trust* requires extensive domain knowledge, which is expensive in term of labour cost [2] [4]. *Data-driven approaches* use machine learning or statistical analysis to learn the relationship among trust information. In comparison to the model-driven approaches, the data-driven approaches do not require expert advice and domain-specific expertise. In addition, the data-driven approaches can be applied to model multiple contexts, which are suitable for addressing diverse needs of various IoT service domains [5].

Trust measurement and prediction also poses challenges to network infrastructure. Traditionally trust information is stored in centralized clouds for service providers and consumers to share. With the broad application of IoT technologies, the numbers of IoT devices, services, providers and customers keep sharply increasing, leading to rapid generation of trust-related information [6] [7]. In this regard, the network capacity (e.g. network bandwidth) of centralized clouds cannot meet the requirement of transmitting high-volume and ever-growing trust information from IoT devices to cloud data centers. Since IoT business is sensitive to *network delay*, it usually needs to provide services within *tens of milliseconds* [7]. For example, intelligent transportation systems need to receive and analyze local messages from in-vehicle applications and roadside sensors, to issue hazard warnings in a short time. This allows adjacent vehicles to receive data in a very short time, enabling drivers to react in a timely manner [7]. Therefore, a centralized cloud-based infrastructure is no longer a viable solution, considering that it cannot provide high-speed information transmission and low-latency service delivery.

Mobile Edge Computing (MEC) is a distributed computing paradigm that moves computing, storage and network resources close to data sources and end users. It enables efficient transmission and processing of rapidly generated high-volume IoT service trust information by allowing cellular base stations to provide computing and storage resources to IoT devices nearby [6] [7]. In this way, MEC can mitigate the bottleneck of transmitting IoT service trust information to centralized clouds and deliver low-latency service. This is significant for improving the efficiency and quality of IoT services.

However, simply turning to MEC infrastructure is inadequate to fully solve the problem of trust prediction in IoT. In fact, the introduction of novel trust information processing techniques is also crucial to overcome the typical challenges posed by MEC-based IoT systems. In such systems, trust information is typically accumulated in a massively distributed manner within each MEC environment, creating distributed data silos. Since the trust information generated by each MEC environment is different, it will lead to non-identical

[§]Corresponding Author

J. Bai and H. Dong, "Federated Learning-driven Trust Prediction for Mobile Edge Computing-based IoT Systems," 2023 IEEE International Conference on Web Services (ICWS), Chicago, IL, USA, 2023, pp. 131-137, doi: 10.1109/ICWS60048.2023.00031.

and independent data distribution (non-IID), which severely impacts the performance of trust measurement [8]. Therefore, for the same services provided in different MEC environments, different trust characteristics need to be considered for the performance of the trust prediction model.

There are existing works, such as [5] [9] [10], that mainly focuses on predicting the trustworthiness of MEC-base IoT services. These works adopt machine learning to complete trust prediction models. However, the existing work in this field suffers from the following key challenges, which originate from the intrinsic characteristics of IoT service trust information in distributed MEC environments and the low-latency requirement of IoT service trust prediction.

1) The IoT service trust information heterogeneity problem resulted from a mixture of various data distributions in different MEC environments has not been properly addressed by existing data-driven trust prediction approaches. IoT service trust information based on MEC is context-dependent. Different MEC environments would generate IoT service trust information with diverse data distributions and sample sizes, leading to a data heterogeneity problem. However, most of the existing works are not specifically designed to address such data heterogeneity problem.
2) None of the existing data-driven approaches can achieve enough training efficiency while meeting the requirement of high accuracy. IoT service trust prediction requires efficiently training a trust model with high performance in an MEC environment to meet the requirement of quick service selection decision-making and service delivery. However, the existing data-driven IoT service trust prediction models cannot meet the requirements of high accuracy and low training overhead at the same time, evident in our experiment.

This paper aims to deliver a federated learning-based approach to more effectively predict the trustworthiness of IoT services in distributed MEC environments. The core idea of federated learning is to allow the learning model on each device to be trained locally based on the local data stored in the devices, and to transmit the local model weights to a server to train a global model [11]. Our specific contributions addressing the challenges outlined above are summarized below.

1) To address the data heterogeneity problem, we model the IoT service trust prediction problem in MEC environments as a federated optimization problem with a mixture of different data distributions (i.e., mixture distribution of IoT trust information). Our objective is to optimize the global prediction model with diverse data distributions in different MEC environments. Driven by this objective, we employ a Federated Expectation-Maximization (FedEM) framework to train a family of distributed IoT service trust prediction models. The FedEM framework can adaptively balance the data imbalance problem among various MEC environments, thereby solving the data heterogeneity problem.
2) Our FedEM-based framework can achieve high-accuracy service trust prediction in MEC-based IoT systems with relatively short training time. The FedEM framework is able to find the relationship of shared implicit features between the underlying data features in different MEC environments. These learned relationships enable effective knowledge sharing among the distributed IoT service trust prediction models, thereby improving the model accuracy and convergence speed.
3) We compare the performance of the proposed methods with the state of the art distributed data-driven IoT trust prediction methods upon a real-world dataset. Our methods show better balance between accuracy and convergence speed than the existing methods.

This paper is organized as follows: In Section 2, we formally define the problem setting and present a mathematical framework enabling data-driven IoT service trust prediction in distributed MEC environments. Section 3 introduces the preliminary. The detail of the proposed solution is presented in Section 4, where we describe the implementation of the mathematical framework introduced in Section 3. Section 5 provides a comprehensive evaluation of our proposed solution through a series of experiments. Finally, Section 6 concludes our work and provide potential directions for future research.

II. Problem formulation

This section defines the problem of IoT service trust prediction in distributed MEC environments through mathematical formulas.

Assuming a set of arbitrary trust features, they can be organized into vectorized form x_i in $\mathbb{R}^d$, where d represents the dimension of x_i. The impact of each trust feature on the overall trust value is represented by the coefficient vector w in $\mathbb{R}^d$, and the mapping function tr defines how to combine each trust feature with its respective weight coefficients to obtain the overall trust value y. Any arbitrary trust model can be represented as follows:

$$\hat{y_i} = \mathrm{tr}\left(x_i; w\right) \qquad \text{where} \qquad \mathrm{tr} : \mathbb{R}^d \times \mathbb{R} \Rightarrow \mathbb{R} \tag{1}$$

where $\hat{y_i}$ represents the predicted value for the i-th instance, and tr represents that it takes a vector in $\mathbb{R}^d$ and a scalar in $\mathbb{R}$ and produces a scalar in $\mathbb{R}$ as output.

Specifically, the trust prediction model is designed for an MEC topology comprising multiple MEC environments. Let $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ be a set of MEC topology with m MEC environments. Each MEC environment has its own local dataset. Let $D = \{d_1, d_2, \ldots, d_m\}$ be a set of local datasets of m MEC environments. The local data distribution $\{\mathcal{D}_t\}_{t\in\mathcal{T}}$ is usually not uniform, so each data distributions $\mathcal{D}_t$ train separate model weights $w_{\mathcal{D}_t} \in \mathcal{W}$. Finding the best set of w values can be generally defined as a loss minimization problem in the following form:

$$\min_{w\in\mathcal{W}} \mathcal{L}_{\mathcal{D}_t}(w) = \frac{1}{\mathcal{T}} \sum_{t=1}^{\mathcal{T}} \frac{n_t}{n} \mathbb{E}_{\mathcal{D}_t} \mathcal{L}_{t_{\mathcal{D}_t}}(w) \tag{2}$$

where n_t denotes the sampling of the local dataset of the t-th MEC environment, $\mathbb{E}_{\mathcal{D}_t}$ represents the expected value after taking a sample for the data set $\mathcal{D}_t$ distributed on the MEC environment t, n is a round of sampling of the total data set, and w_{r+1}^t represents a new round of the t-th local model weights. In addition, we need to find the potential weight relationship between MEC environment through the local data distribution differences. The data in $t \in \mathcal{T}$ generates a local distribution $\mathcal{D}_t$ from the local dataset. $\mathcal{L}_{t_{\mathcal{D}_t}}$ represents the local loss function of the t-th MEC environment under its specific data distribution. For each independent MEC environment, $\mathcal{L}_t$ can be further expressed in the following form:

$$\mathcal{L}_t(w) = \frac{1}{n_t} \sum_{i=1}^{n_t} \ell_i(w) \tag{3}$$

where ℓ_i is denoted as the loss function for individual data sampling in the local MEC environment. Each MEC environment updates the local model as follows:

$$w_{r+1}^t = w_r^t - \eta \nabla \mathcal{L}_t(w) \tag{4}$$

where η is the learning rate, and ∇ denotes the gradient operation. Then the MEC environments upload their trained model weights to the central cloud. The central cloud aggregates the model parameters of different MEC environments, expressed as follows:

$$w_{r+1} = \sum_{t=1}^{\mathcal{T}} \frac{n_t}{n} w_{r+1}^t \tag{5}$$

III. Preliminary

A. Federated Expectation-Maximization

FedEM is a type of federated multi-task learning (MTL) algorithm, which is designed to replace the personalized learning model. It adopts MTL to model the relationship between different clients, thereby increasing the sample size of nodes and improving performance. In FedEM, there are two broad assumptions about local data. FedEM considers each local data distribution $\mathcal{D}_t$ to be a mixture of M underlying distributions $\tilde{\mathcal{D}}_m, 1 \leq m \leq M$. FedEM makes two broad assumptions about their local data [12]:

Assumption 1: There are M underlying (independent) distributions $\tilde{\mathcal{D}}_m, 1 \leq m \leq M$, such that for $t \in \mathcal{T}$, $\mathcal{D}_t$ is mixture of the distributions $\left\{\tilde{\mathcal{D}}_m\right\}_{m=1}^{M}$ with weights $\pi_t^* = [\pi_{t1}^*, \ldots, \pi_{tM}^*] \in \Delta^M$, which denotes the M-dimensional simplex.

$$z_t \sim \mathcal{M}\left(\pi_t^*\right), \quad \left((\mathbf{x}_t, y_t) \mid z_t = m\right) \sim \tilde{\mathcal{D}}_m, \quad \forall t \in \mathcal{T} \tag{6}$$

where $\mathcal{M}(\pi)$ represents a categorical distribution parameterized by π.

Here we use $pm(x, y)$, $pm(x)$ and $pm(y)$ to represent the probability density functions corresponding to $\tilde{\mathcal{D}}_m$. Additionally, we assume that the marginals over X are equivalent.

Assumption 2: For all $m \in [M]$, we have $p_m(\mathbf{x}, y) = p_m(\mathbf{x})$.

FedEM can hold without strictly requiring Assumption 2, but in most cases, it can be expressed as the following problem:

$$\forall t \in \mathcal{T}, \quad \underset{h_t \in \mathcal{H}}{minimize}\, \mathcal{L}_{\mathcal{D}_t}(h_t) \tag{7}$$

However, under Assumption 2, FedEM can be used with discriminative models such as neural networks.

FedEM aims to estimate the optimal parameters of the components $\Theta^* = (\theta_m^*)_{1 \leq m \leq M}$ and mixture weights $\Pi^* = (\pi_t^*)_{1 \leq t \leq T}$ by minimizing the negative log-likelihood $f(\Theta, \Pi)$. However, this is a non-convex problem that requires a sophisticated approach to solve. One such approach is the Expectation-Maximization (EM) algorithm, which is commonly used in mixture modeling. The algorithm alternates between two steps: the Expectation step (E-step) and the Maximization step (M-step).

E-step:

$$q_t^{k+1}\left(z_t^{(i)} = m\right) \propto \pi_{tm}^k \cdot \exp\left(-l\left(h_{\theta_m^k}\left(\mathbf{x}_t^{(i)}\right), y_t^{(i)}\right)\right), \quad t \in [T], m \in [M], i \in [n_t] \tag{8}$$

M-step:

$$\pi_{tm}^{k+1} = \frac{\sum_{i=1}^{n_t} q_t^{k+1}\left(z_t^{(i)} = m\right)}{n_t}, \quad t \in [T], \quad m \in [M] \tag{9}$$

$$\theta_m^{k+1} \in \underset{\theta \in \mathbb{R}^d}{\arg\min} \sum_{t=1}^{T} \sum_{i=1}^{n_t} q_t^{k+1}\left(z_t^{(i)} = m\right) l\left(h_\theta\left(\mathbf{x}_t^{(i)}\right), y_t^{(i)}\right), \quad m \in [M] \tag{10}$$

During the Expectation step of FedEM, the algorithm calculates the probabilities of each data point belonging to each component by updating the distribution q_t over the latent variables $z_t^{(i)}$ for every data point, given the current estimates of the parameters $\{\Theta, \Pi\}$. In this way, the algorithm determines the probabilities of each component to each data point.

During the Maximization step, the algorithm updates the parameters $\{\Theta, \Pi\}$ by maximizing the expected log-likelihood (Equation 10), where the expectation is taken with respect to the current latent variables' distributions. Specifically, this step involves finding the optimal parameters of the components and the mixture weights, which can be done using standard optimization techniques. The resulting estimates are then used to update the distributions over the latent variables in the subsequent Expectation step.

Finally, the model weight update is performed in the MEC environment (see Equation 10). This equation represents an update step in the context of a machine learning algorithm, where θ_m^{k+1} is the updated value of a model weight θ for a independent distributions m in iteration $k+1$. The goal of the update is to minimize the empirical risk over a set of training examples, where the risk is measured using the loss function l and the predicted values are obtained using the hypothesis function h_θ. The sum over t and i indicates that the empirical risk is computed over all training examples in all T

datasets of clients, with n_t representing the number of training examples in the t-th dataset. The term $q_t^{k+1}(z_t^{(i)} = m)$ is a weight assigned to the i-th training example in the t-th dataset, based on its assigned cluster $z_t^{(i)}$ in the previous iteration k. This weight is used to give more importance to examples that belong to the current cluster m, and less importance to examples that belong to other clusters. The optimization problem is solved using the argmin operator, which finds the value of θ that minimizes the empirical risk over all training examples in the current cluster m.

IV. FedEM-based IoT service trust prediction

We employ the FedEM algorithm [12] to better solve the heterogeneity problem caused by the mixture distribution and non-IID of the MEC topology and converge as much as possible in the shortest number of communications. This algorithm allows the model training and EM steps to be performed MEC environment-side, thus accomplishing federated optimization. Compared with other federated learning methods, the FedEM algorithm is more effective in dealing with non-IID and mixture distribution and has a higher convergence speed. For MEC-based IoT systems, FedEM can be trained locally in each MEC environment, and then perform global model aggregation on the server side, thereby realizing trust prediction tasks across MEC environments.

In addition, the federated optimization based on FedEM can ensure the fast convergence of the model, which is a critical feature. Federated optimization algorithms encounter many problems when training across multiple devices, including network communication delays between devices, data heterogeneity, and data inhomogeneity [11]. These issues can lead to situations where training is unstable or may fail to converge. In contrast, the FedEM algorithm can be trained based on the local data of the MEC environment, avoiding the problem of data inhomogeneity and heterogeneity, thereby improving the convergence speed and accuracy of the model. Therefore, using the FedEM algorithm for federated optimization can better guarantee the convergence and accuracy of the model.

We adopted a multi-layer perceptron (MLP) as our training model, which consists of an input layer, a hidden layer with multiple ReLU activation function perceptrons and an output layer with a single perceptron with sigmoid activation function. During training, each local MEC environment trains its own MLP to implement a binary classifier. These local binary classifiers will be combined to generate a global trust prediction model through a federated optimization algorithm. Due to the different data distribution and characteristics of each local environment, the performance of local classifiers will also vary. Therefore, we need to use federated optimization to balance the contributions of different local classifiers to generate a more accurate and robust global trust prediction model.

We provide the Algorithm 1 to illustrate our proposed federated optimization algorithm. Lines 1 to 4 initialize the model parameters in the central cloud and the mixing weights for each MEC environment, respectively. The algorithm has

Algorithm 1 FedEM-based IoT Trust Prediction in MEC

Input : Set of MEC environments $\mathcal{T}$; Set of local MEC environment data $D_{1:\mathcal{T}}$; Number of mixture distributions M; Number of communication rounds K

Output: θ_m^K; $m \in [M]$

1: *Central cloud initialize* θ_0 *for* $1 \leq m \leq M$ *to the* $\mathcal{T}$ *MEC environments*
2: **for all** $t = 1$ to $\mathcal{T}$ in parallel over $\mathcal{T}$ MEC environments **do**
3: *initialize* π_t^0
4: **end for**
5: **for all** $k = 1$ to K **do**
6: *Central cloud broadcasts* θ_{k-1} *for* $1 \leq m \leq M$ *to the* $\mathcal{T}$ *MEC environments*
7: **for all** $t = 1$ to $\mathcal{T}$ in parallel over $\mathcal{T}$ MEC environments **do**
8: **for all** $m = 1$ to M **do**
9: // E-step:
10: **for all** $i = 1$ to n_t **do**
11: $q_t^k\left(z_t^{(i)} = m\right) \leftarrow \frac{\pi_{tm}^k \cdot \exp\left(-l\left(h_{\theta_m^k}\left(\mathbf{x}_t^{(i)}\right), y_t^{(i)}\right)\right)}{\sum_{m'=1}^{M} \pi_{tm'}^k \cdot \exp\left(-l\left(h_{\theta_{m'}^k}\left(\mathbf{x}_t^{(i)}\right), y_t^{(i)}\right)\right)}$
12: **end for**
13: // M-step:
14: $\pi_{tm}^k \leftarrow \frac{\sum_{i=1}^{n_t} q_t^k\left(z_t^{(i)}=m\right)}{n_t}$
15: *The indices* I *are sampled uniformly from 1 to* $|D|$
16: $\theta_{tm}^k \leftarrow \theta_{tm}^{k-1} - \eta_{k-1} \sum_{i \in I} q_t^k\left(z_t^{(i)} = m\right) \cdot \nabla_\theta l\left(h_{\theta_m^k}\left(\mathbf{x}_t^{(i)}\right), y_t^{(i)}\right)$
17: **end for**
18: *MEC environment* t *sends* θ_{tm}^k *and* $1 \leq m \leq M$ *to the central cloud*
19: **end for**
20: **for all** $m = 1$ to M **do**
21: $\theta_{tm}^k \leftarrow \sum_{t=1}^{\mathcal{T}} \frac{n_t}{n} \cdot \theta_{m,t}^k$
22: **end for**
23: **end for**

a quadruple loop nest. The outermost loop, from lines 5 to 23, iterates on the number of communication rounds K. The second loop, from lines 7 to 19, is executed in parallel on each MEC environment t. The third loop, from lines 8 to 17, updates each mixture distribution m. The fourth loop, from lines 10 to 12, updates the local mixing weights for each MEC environment, which refers to the E-step. Specifically, the probability of mixture distribution $q_t^k(z_t^{(i)} = m)$ is computed. The coefficient π_{tm}^k of the m-th mixture distribution is then computed in the M-step. The model parameters θ_{tm}^k are updated by computing stochastic gradients. The MEC environment sends the updated parameter θ_{tm}^k to the central cloud represented by line 18. The central cloud aggregates the updates of all the MEC environments represented from line 20 to line 22. Then, the algorithm goes back to line 4

and starts a new round. The central cloud broadcasts the new model weights θ_{k-1} for each mixture distribution. The above steps are repeated until the specified number of communication rounds K is reached.

V. Evaluation

We design the following experiments to evaluate our contributions:

1) To evaluate whether our method can solve the problem of heterogeneous trust information in IoT services caused by the mixture distribution, we use a public IoT dataset to compare the accuracy between our method and the baseline methods to verify the generalization performance of our method. We then split the dataset in a way that mimics the mixture data distribution in MEC-based IoT systems and evaluate if our method can address the heterogeneity problem caused by the mixture distribution.
2) To verify whether our method can achieve higher efficiency while meeting the requirement of high accuracy, we compare the convergence time and the required number of training iterations between our FedEM model and the baseline methods.

Our experimental evaluation is performed on a computer equipped with an Intel Xeon E5-2686 2.30GHz CPU, 60 GB RAM, and an NVIDIA RTX A4000. All comparison models are implemented using the Python programming language. To implement federated learning, we use the PyTorch (1.12) library[1]. Additionally, we use the CVXPY (1.2.2) library[2] to implement another data-driven IoT service trust prediction model (i.e., S-ADMM [5]). The hyperparamter values of all the models are tuned to achieve their optimal performance. The source code of our implementation can be found from[3].

By implementing all the models in Python and utilizing the aforementioned libraries, we are able to conduct experiments in a standardized and reproducible manner, ensuring that our results are accurate and reliable.

A. Dataset

UNSW-NB15[4]: This dataset contains a collection of network traffic data. The data consists of both normal network activities and synthetic attack behavior network activities that were generated in a laboratory setting. This dataset contains around 2 million records, with each record consisting of 49 features. Each record corresponds to a result, which could be either a benign transaction or one of nine different types of attacks. For the purpose of our experiments, we marked the nine different types of attacks as harmful and all the benign transactions as benign.

To simulate a topology of distributed MEC environments with mixture distribution, we divide dataset into 100 sets of MEC environment with varying local datasets lengths and different data distributions. The training and test datasets were split in an 80:20 ratio to ensure that the models were evaluated using a fair and consistent approach. The length of the training data set is within the range of $[34, 27338]$, and the length of the test data set is within the range of $[9, 6835]$. It is worth noting that the trust value 0 of an IoT service represents a benign sample, while 1 represents a harmful sample in the federated learning dataset. We have taken the step of splitting the dataset and generating a mixed distribution before running the models, to ensure that all models are tested on the same MEC topology and to eliminate any potential result differences due to the use of different data. This ensures that our experimental dataset conforms to the assumption of mixed data distribution. All the data are first divided into two clusters. For each cluster, its samples are divided into 100 MEC environments using Dirichlet Distribution. For each MEC environment, samples are selected from each cluster such that the number of samples for each MEC environment conforms to the Dirichlet distribution of the belonged cluster. Specifically, when the Dirichlet distribution is used as the prior distribution of the multinomial distribution, if there are samples available, we can use Bayesian inference to update the posterior distribution to obtain a new probability distribution. This new probability distribution can be viewed as a mixture distribution.

B. Other Approaches Evaluated

1) Federated Averaging: Federated Averaging (FedAvg) is one of commonly used federated learning algorithms. Its basic idea is to distribute the model to each device, and perform local training on each device. The trained local model parameters are then sent back to the server. A main model in the server takes the average of these parameters and sets them as its new weight parameters and passes them back to the devices for the next iteration. This process is iterated until the model converges [11]. Although FedAvg is a federated learning algorithm, it has no ability to capture the relationship between MEC environments and its related data distribution [12].

2) Stochastic Alternating Method of Multipliers: The Stochastic Alternating Method of Multipliers (S-ADMM) is a distributed optimization algorithm designed to solve large-scale convex optimization problems. S-ADMM transforms the original problem into a network lasso problem [5]. The work of [5] realizes MEC-based IoT service trust prediction through S-ADMM. S-ADMM is able to randomly sample the local data, so as to offset the non-IID problem caused by different data distributions.

C. Key Performance Indicators

We use several key performance indicators (KPIs) to comprehensively evaluate and compare the performance of the above methods, including accuracy, number of communication rounds, elapsed time and variance of accuracy.

Accuracy measures the number of correctly predicted samples in a given set divided by the total number of samples in the MEC environments. We employ this metric to compare the

[1]https://pytorch.org
[2]https://www.cvxpy.org
[3]https://github.com/SHVleV9CYWkK/MEC-IoT-Trust-FedEM
[4]https://research.unsw.edu.au/projects/unsw-nb15-dataset

TABLE I
PERFORMANCE COMPARISON AMONG ALL THE MODELS

Dataset	Model	Maximum Accuracy	Elapsed Time	Number of Communication Rounds
UNSW-NB15	FedAvg	98%	152.44s	10
	S-ADMM	88%	156.38s	22
	FedEM	98%	105.75s	5

overall accuracy of different models and to determine which model is most effective at predicting IoT service trust.

Elapsed time is adopted to compare the convergence speed of all the candidate algorithms. By measuring the time required to reach convergence, we can determine which model is most effective in a given MEC environment.

Number of communication rounds is adopted to evaluate the data transmission overhead of each model during the training process. We evaluated the convergence rate of each model and determined which model required the lowest number of communication rounds to achieve the best results.

D. Results and Discussion

1) Accuracy: The experimental results show that our method outperforms the other candidate methods in terms of accuracy. Specifically, on the UNSW-NB15 test set, FedEM achieved a 98.5% accuracy at round 25, while FedAvg and S-ADMM achieved 97.51% and 85.9%, respectively. Figure 1 clearly shows that FedEM consistently outperforms S-ADMM in terms of accuracy. Further details regarding the hyperparameters can be found in the respective section. In addition, the convergence stability of FedAvg from the eighth round to the twelfth round on the UNSW-NB15 test set is not stable. It is worth noting that S-ADMM cannot converge stably in the twenty-fifth round on the UNSW-NB15 test set, and it may need more rounds. We can also find that the accuracy of S-ADMM cannot always be higher than that of FedEM. Overall, our proposed method has been proven to be effective in achieving higher accuracy when compared to FedAvg and S-ADMM.

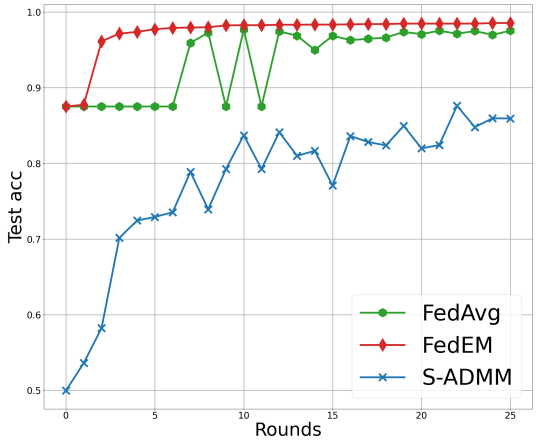

Fig. 1. Accuracy of all the models on the UNSW-NB15 dataset

2) Convergence: Our experimental results show that our proposed FedEM method achieves high accuracy with fewer communication times than the conventional FedAvg and S-ADMM methods. As shown in Table 1, our method requires significantly fewer rounds of communications to converge. This indicates that our method needs less communication overheads for model training.

Next, we compare these methods in terms of the convergence time. First, we recorded the maximum accuracy of FedAvg and S-ADMM over 25 rounds and the time for reaching their highest accuracy. We then use the highest accuracy of these two models as a threshold and record the time that FedEM reaches this accuracy. It can be seen from Table 1 that FedEM requires significantly less time than the other two models to reach the maximum accuracy of these two models. These results demonstrate that our proposed FedEM method achieves high accuracy in significantly reduced running time, namely, a better balance between convergence speed and accuracy, compared to the conventional methods.

In conclusion, the experimental results preliminarily prove that the our proposed trust prediction methods based on FedEM can address the IoT service trust information heterogeneity problem resulted from a mixture of various data distributions in different MEC environments. These methods can achieve high prediction accuracy with fast convergence speed and less communication overheads.

VI. CONCLUSION AND FUTURE WORK

This paper proposes a federated learning-based method for data-driven IoT service trust prediction considering the presence of mixture distributions in distributed mobile edge computing (MEC) environments. We argue that training a trust prediction model in the MEC setting can be interpreted as a federated optimization problem. We propose a Federated Expectation-Maximization (FedEM)-based method to address the problem of mixture distributions. Finally, we conduct a series of simulation experiments to verify the feasibility and effectiveness of the method in distributed MEC environments. The experimental results show that the proposed methods can achieve high training efficiency while ensuring high prediction accuracy than the state-of-the-art data-driven MEC-based IoT service trust prediction method and a Federated Averaging (FedAvg)-based method.

ACKNOWLEDGMENT

This research was supported by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (project DP220101823).

REFERENCES

[1] Y. L. Sun, Z. Han, W. Yu, and K. R. Liu, "A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2006, pp. 1–13.

[2] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for internet of things," *J. Netw. Comput. Appl.*, vol. 42, pp. 120–134, 2014.

[3] Y. Wang, "Trust quantification for networked cyber-physical systems," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2055–2070, 2018.

[4] U. Jayasinghe, A. Otebolaku, T.-W. Um, and G. M. Lee, "Data centric trust evaluation and prediction framework for iot," in *ITU Kaleidoscope: Challenges for a Data-Driven Society (ITU K)*, 2017, pp. 1–7.

[5] P. Abeysekara, H. Dong, and A. K. Qin, "Data-driven trust prediction in mobile edge computing-based iot systems," *IEEE Transactions on Services Computing*, vol. 16, no. 1, pp. 246–260, 2023.

[6] M. T. Beck, M. Werner, S. Feld, and T. Schimper, "Mobile edge computing: A taxonomy," 2014.

[7] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[8] 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.

[9] P. Abeysekara, H. Dong, and A. K. Qin, "Edge intelligence for real-time iot service trust prediction," *IEEE Transactions on Services Computing*, pp. 1–14, 2023.

[10] P. Abeysekara, H. Dong, and A. Qin, "Machine learning-driven trust prediction for mec-based iot services," in *IEEE International Conference on Web Services (ICWS)*, 2019, pp. 188–192.

[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *International Conference on Artificial Intelligence and Statistics*, vol. 54, 2017, pp. 1273–1282.

[12] O. Marfoq, G. Neglia, A. Bellet, L. Kameni, and R. Vidal, "Federated multi-task learning under a mixture of distributions," *Advances in Neural Information Processing Systems*, vol. 34, pp. 15 434–15 447, 2021.