FedSVRG Based Communication Efficient Scheme for Federated Learning in MEC Networks

Dawei Chen, Student Member, IEEE, Choong Seon Hong, Senior Member, IEEE, Yiyong Zha, Yunfei Zhang, Xin Liu, and Zhu Han Fellow, IEEE

Abstract—Recently, a novel machine learning technique, federated learning, attracts ever-increasing interests from academia to industry. The main idea of federated learning is to collaboratively train a global optimal machine learning model among all the participants. During the process of parameter updating, the communication cost of the system or network can be extremely huge with a large number of iterations and participants. Although the edge computing paradigm can decrease the latency to a certain extent, how to obtain further delay reduction is still a challenge. Therefore, to address the problem, we firstly model the corresponding problem into a finite-sum optimization problem. Then, we propose a federated stochastic variance reduced gradient based method to decrease the number of iterations between the participants and server from the system perspective, and guarantee the accuracy at the same time. Meanwhile, the corresponding convergence analysis is provided. Finally, we test our proposed method on the linear regression problem and the logistic regression problem. The simulation results show that our proposed method can reduce the communication cost significantly compared with general stochastic gradient descent based federated learning.

Index Terms—Federated learning, multi-access edge computing, stochastic variance reduce gradient.

I. INTRODUCTION

The past decade has witnessed tremendous growth of machine learning techniques and applications, such as face recognition, autonomous driving, disease diagnosis, and object detection and tracking, etc. The prosperity relies heavily on the enormous available data and an ever-increasing number of users. It is anticipated that in 2025, the number of devices connected with the Internet will achieve 80 billion and the total volume of generated data can be up to 180 trillion gigabytes [1]. On the other hand, the powerful and efficient computation hardware design and computing architecture, like parallel high performance graphics processing units (GPUs), also lead to a flourishing machine learning community.

However, traditionally, the limitation of storage memory capacity and computation ability does not allow the machine learning algorithms to be performed on devices. Therefore, cloud computing is proposed and becomes a popular solution to deal with vast data and computation-intensive applications. Conventional cloud computing is constructed in a centralized paradigm, which means all the data have to be uploaded to the central server for further processing. In this case, latency can be an inevitable defect due to the limited bandwidth and long transmission distance. Therefore, to overcome the delay problem, multi-access edge computing (MEC) is proposed to push the data processing to the edge of the network. In MEC, many edge servers will be geographically distributed at the edge of the network. The tasks with relatively lower complexity can be addressed directly within edge servers instead of uploading to a remote centralized data center, which means services are much closer to the users so as to decrease service latency [2].

Even though the delay has been reduced, within an edge server coverage area, there still exists a major problem. If we need to obtain a machine learning model suitable for every user, all the user-owned data have to be transmitted to the edge server for model training, which also leads to expensive communication costs due to large data sizes [3]. Therefore, federated learning is proposed to relieve the transmission pressure. In federated learning, the aim is to obtain a collaboratively trained model while only need to transmit individual model parameters (i.e. weights) instead of raw data to the server [4]. Compared to traditional centralized machine learning techniques, the main benefits of federated learning can be summarized as follows:

- The conventional machine learning needs to accumulate all the datasets, which results in tremendous communication workloads. Whereas, it can be easily moderated by federated learning because instead of transmitting data set, what needs to be exchanged is model parameters, whose data size is much smaller [5].
- The generated individual data are actually non-i.i.d. due to diverse behavior characteristics. Federated learning collects individual models from different users and performs aggregation among them. Therefore, it can be an effective way to tackle non-i.i.d. scenarios [6].

Obviously, communication between users and server happens each time to update global parameters. Although the
transmission workloads have been dramatically decreased by only exchanging parameters, considering from the network and system perspective, the consumption brought by communication can still be huge when the number of participants is sufficiently large [7]. In order to further reduce the communication cost, [8] proposes a method that can directly learn an update from a restricted space parametrized using a smaller number of variables, which is based on the perspective of compression. [9] proposes a communication protocol that is able to compress both the uplink and downlink communications through the combination of different techniques, such as sparsification, ternarization, error accumulation, and optimal Golomb encoding. Likewise, [10] proposes a federated trained ternary quantization algorithm, which optimizes the quantized networks on the clients through a self-learning quantization factor to reduce communication cost. However, all of the literature above is based on the view of data compression. Although communication costs can be reduced, extra computation resources are required as well. Therefore we propose a federated stochastic variance reduced gradient (FedSVRG) based scheme to decrease the total number of interactions between users and edge server while still guarantee the accuracy requirement. Compared with some existing literature [8]–[12], we consider the communication efficiency problem in a MEC scenario from the perspective of optimization in a finite sum form. Besides, we adopt the OFDMA as the communication scheme to perform quantitative analysis for both linear regression and logistic regression problems. To summarize, the main contributions of this work are as follows:

1) In order to analyze the communication cost in the federated learning framework, we model the corresponding communication efficiency problem for edge federated learning into a finite-sum optimization problem.

2) For reducing the communication cost of the network or system, we proposed a FedSVRG based scheme to maintain a lower number of communication interactions but keep the desired accuracy at the same time motivated by improving convergence rate compared with the traditional SGD method.

3) For the experiments, we apply the proposed FedSVRG on the linear regression and logistic regression problems, and compare it with general stochastic gradient descent (SGD) based federated learning. Consequently, we can see that our proposed method performs better, reflected by lower communication costs with certain accuracy.

The structure of the rest paper is as follows. Section II introduces the general federated learning and system model with the corresponding problem formulation. Section III introduces the proposed FedSVRG scheme. Section IV shows the performance of our proposed method compared with general SGD based federated learning. Finally, a conclusion is drawn in Section V.

II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider a MEC server providing services to a multi-user network which contains $N$ users. The purpose of $N$ users is to collaboratively obtain a global machine learning model, which suits all the users within this network. Due to the different characteristics of users and diversity of applications, the distribution of data generated by each user can be different. Here, we define the dataset owned by user $i$ is $D_i$. $D_i$ can be described as feature-label pairs $(x_i, y_i)$, where $x_i \in \mathbb{R}^m$ denotes the input data which can be described by $m$ features and $y_i \in \mathbb{R}$ is the corresponding output or the label. Individually, each participant perform local training process to optimize its own model, which can be written as

$$\min_{\omega \in \mathbb{R}^m} f_i(x_i, \omega; y_i),$$

(1)

where function $f_i$ is the loss function of participant $i$. The typical instances of loss function can be

$$f_i = \frac{1}{2} (x_i^T \omega_i - y_i)^2$$

(2)

for the linear regression problem, and

$$f_i = -\frac{1}{K} \left[ \sum_{j=1}^{K} y_i^j \log(\omega_i^T x_i^j) + (1 - y_i^j) \log (1 - \omega_i^T x_i^j) \right]$$

(3)

for the logistic regression problem, where $j$ denotes the $j$-th data sample of participant $i$ and $K$ is the number of data samples.

Macroscopically, the global machine learning model that can be utilized for all the users within network should be obtained. In this case, the weighted aggregation will be performed in the MEC server to obtain the global model parameters, which can be described as

$$\omega^s = \frac{\sum_{i=1}^{N} D_i \omega_i^s}{D},$$

(4)

where $D = \sum_{i=1}^{N} D_i$ is the total dataset in the network, $s$ is the index of aggregation index, and $\tau$ is the number of local update iterations between each global aggregation. Therefore, the loss function of collaborative machine learning model training can be written as

$$\min_{\omega \in \mathbb{R}^m} J(\omega^s) = \frac{1}{N} \sum_{i=1}^{N} f_i(\omega_i^s).$$

(5)

To solve the problem described in (5), multiple interactions between participants and the MEC server are conducted. In the beginning, the MEC server will distribute an initialized model to each participant. Based on the individual dataset, participants will perform local training after certain iterations. After this, all the participant will upload their locally trained parameters to the MEC server. Received all the parameters, the MEC server performs federated average calculation to obtain a global machine learning model, which will be distributed to each user again after aggregation. When the pre-defined accuracy is achieved after several iterations, the process is accomplished. Overall, the whole procedures are illustrated in Fig. 1.

Obviously, the communication happens each time when each participant interacts with the MEC server [13]. Suppose each participant shares the same size of the model and the parameter size is denoted as $e_i$ bits, for $\forall i = 1, ..., N$. Assume
each participant is allocated an orthogonal sub-channel so that we can ignore the interference between them. But each interaction between individual user and server will bring communication cost. In this case, the momentary transmission rate for participant $i$ can be described as

$$R_i = B \log_2 \left( 1 + \frac{p_i |G_i|^2}{N_i} \right), \forall i = 1, ..., N,$$  \hspace{1cm} (6)

where $B$ is the sub-channel bandwidth allocated to user $i$, $p_k$ is the transmission power, $|G_i|^2$ is the channel gain between participant $i$ and MEC server, and $N_i$ is the Gaussian noise. Correspondingly, each time to upload the local parameters, the consumed time for each participant can be characterized as

$$T_i = \frac{\epsilon_i}{B \log_2 \left( 1 + \frac{p_i |G_i|^2}{N_i} \right)}, \forall i = 1, ..., N.$$  \hspace{1cm} (7)

To achieve a certain accuracy, or the loss is less than $\epsilon$, the total number of iterations we need is $S$. Therefore, the total communication cost in the network can be defined as

$$C = S \theta \sum_{i=1}^{N} T_i,$$  \hspace{1cm} (8)

where $\theta$ is the unit price for communication per second (i.e. $$/second). Totally, to minimize the communication cost, the problem can be formulated by the following,

$$\min \ C = S \theta \sum_{i=1}^{N} T_i,$$  \hspace{1cm} (9)

s.t. $J(\omega^S) \leq \epsilon.$  \hspace{1cm} (10)

### III. Methodology

To solve (9), the key point is to reduce the total communication interactions. Traditionally, the SGD method is used, which updates the gradient via the following policy,

$$\omega^s = \omega^{s-1} - \alpha_s \nabla J(\omega^{s-1})$$  \hspace{1cm} (11)

where $\alpha_s$ is step size or learning rate on iteration $s$. Assuming the optimal solution $\omega^*$ exists, for the $s$-th iteration, the accuracy can achieve

$$J(\omega^s) - J(\omega^*) = O(1/s),$$  \hspace{1cm} (12)

when function $J$ is convex.

However, because of the existence of randomness in sampling, we need to carefully choose the learning rate $\alpha_s = O(1/s)$ to obtain a sub-linear convergence rate of $O(1/s)$, which means we have to make a trade-off between the slow calculation and fast convergence [14]. Therefore, during the optimizing process, it can be easy to increase the communication cost.

To overcome this defect, we propose a FedSVRG based method. Basically, FedSVRG is made up of two loops of iterations. When performing the inner loop iterations, the average gradient will be re-used for a certain number of iterations instead of updating the new average each time leading to a better convergence rate. Therefore, the number of iterations between the participants and the MEC server can be reduced significantly. At each time, the stored or re-used parameter is denoted as $\hat{\omega}$, which will be utilized for $l$ iterations. The maintained average gradient is

$$\hat{\mu} = \nabla J(\hat{\omega}) = \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\hat{\omega}),$$  \hspace{1cm} (13)

which is calculated throughout the data and will be never changed in inner loop iterations. Then, the updating rule follows the general SGD. At each global aggregation iteration $s = 1, 2, ..., S$, $i_t$ is randomly selected from participants $\{1, ..., N\}$, which can be written as

$$\omega^s = \omega^{s-1} - \alpha_s \left( \nabla f_i(\omega^{s-1}) - \nabla f_i(\hat{\omega}) + \hat{\mu} \right).$$  \hspace{1cm} (14)

Because the expectation of $\nabla f_i(\omega) - \hat{\mu}$ over $i$ equals to zero, we have

$$E[\omega^s | \omega^{s-1}] = \omega^{s-1} - \alpha_s \nabla J(\omega^{s-1}),$$  \hspace{1cm} (15)

and the form is the same as general SGD. However, compared with SGD, each step only relies on derivative $\nabla f_i$, so that the computation cost can be reduced $N$ times. By defining the auxiliary function, we can rewrite the gradients updating rule in (14)

$$\tilde{f}_i(\omega) = f_i(\omega) - (\nabla f_i(\hat{\omega}) - \hat{\mu})^T \omega,$$  \hspace{1cm} (16)


<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub-channel Bandwidth $B$</td>
<td>$20 \text{ MHz}$</td>
</tr>
<tr>
<td>Transmission Power $p$</td>
<td>$23 \text{ dBm}$</td>
</tr>
<tr>
<td>Channel Gain $c$</td>
<td>$4$</td>
</tr>
<tr>
<td>Gaussian Noise $N$</td>
<td>$-96 \text{ dBm}$</td>
</tr>
<tr>
<td>Unit Price $\theta$</td>
<td>$0.01 \text{$/second}$</td>
</tr>
</tbody>
</table>

since $E_i(\nabla f_i(\omega) - \tilde{\mu}) = 0$ and $J(\omega) = \frac{1}{N} \sum_{i=1}^{N} f_i(\omega) = \frac{1}{N} \sum_{i=1}^{N} \tilde{f}_i(\omega)$. When both $\omega$ and $\omega^*$ converge to the optimal parameters $\omega^*$, we can see $\tilde{\mu}$ approaches to zero, which means the variance of rule (14) is reduced [15]. Therefore, when $\nabla f_i(\omega) \rightarrow \nabla f_i(\omega^*)$, we have

$$\nabla f_i(\omega^{*-1}) - \nabla f_i(\omega) + \tilde{\mu} \rightarrow \nabla f_i(\omega^*) - \nabla f_i(\omega^*) \rightarrow 0.$$  

(17)

Because of the benefit of variance reduction, for SVRG, the learning rate $\alpha_s$ does not need to decay so that a relatively larger step size can be utilized, which leads to faster convergence [11]. The algorithm of SVRG is summarized in Algorithm 1. One remark that needs to be mentioned is that we have two options to decide the value of $\omega^s$. Practically, both of utilizing the latest calculated value $\omega^s$ and using the random intermediate value $\omega^s$ are fine.

IV. SIMULATION RESULTS

For the simulation part, we test our proposed FedSVRG for linear regression and logistic regression. The communication model parameters settings are as summarized in Table I. Meanwhile, for each experiment, we utilize general SGD as the baseline method to compare the performance with FedSVRG. The iteration stops either the epoch index achieves the maximum steps or the loss achieves optimality gap tolerance.

A. Linear Regression

We randomly generate data that totally contains 1,200 samples and the data is one-dimensional. The boundary is pre-defined and we set the standard deviation as 0.15 to randomly add noise into the boundary function so as to generate the dataset. The training data set contains 600 data points (i.e. $N = 600$) and the rest 600 data samples form the test set. The number of the maximum epoch is set as 100 and the rest 600 data samples form the test set.

In Fig. 2(a), we can see that the performance of SGD and FedSVRG are almost the same except the test mean square error (MSE) of FedSVRG is slightly better, which is 0.002 lower than the SGD method. For Fig. 2(b), the yellow dot represents the optimal point, where the function in (5) achieves the minimum value. The red and blue dots describe the solution after each epoch and the corresponding lines show the route of gradient descent. Obviously, FedSVRG is possessed of a faster convergence speed, which iterates only 4 epochs to achieve the optimal point at (0.58, 2.24). While SGD spends 9 epochs to arrive its optimal solution at (0.58, 2.23). Fig. 2(c) shows the relationship between the loss and the corresponding cost during the iterating process. Evidently, the loss curve of FedSVRG decreases faster than SGD, which describes the same property reflected in Fig. 2(b). At the same time, at each epoch with the same loss value, FedSVRG always gives the more economical solution for the communication cost, which demonstrates the effectiveness of our proposed method.

B. Logistic Regression

1) Two-class Classification: To generate the data for two-class classification problem, the centroid points are pre-
FedSVRG can reduce communication cost at the equivalent linear regression and two-class classification problems, the loss general SGD method. Simultaneously, in Fig. 4, as is in a better accuracy for 81.2%, which is 0.3% higher than the respective. Statistically, we can find that FedSVRG achieves the simulation results are shown in Table III and Fig. 4, and the loss achieves optimality gap tolerance is set as 10^{-36}. The corresponding simulation results are displayed in Table II and Fig. 3, respectively. Compared with general SGD methods, obviously, FedSVRG obtains 89.2% accuracy, which is 0.5% higher. At the same time, for a certain loss value, FedSVRG always gives a solution with lower communication cost, which is the same as the performance with regard to the linear regression problem. Besides, looking into the slopes of two curves, we can find that the loss of FedSVRG reduces faster, which implies a faster convergence rate as well.

2) Multi-class Classification: For multi-class classification problem, the number of generated data classes is set as five. Each class contains 300 samples, and each data point is three-dimensional. Besides, we set the standard deviation as 0.15 to randomly add noise to the centroid points. Likewise, we separated the dataset into training dataset and test dataset equally. The number of the maximum epoch is set as 1,000 and the loss achieves optimality gap tolerance is set as 10^{-12}. The simulation results are shown in Table III and Fig. 4, respectively. Statistically, we can find that FedSVRG achieves a better accuracy for 81.2%, which is 0.3% higher than the general SGD method. Simultaneously, in Fig. 4, as is in linear regression and two-class classification problems, the loss curve for FedSVRG decreases faster, which also reflects that FedSVRG can reduce communication cost at the equivalent loss value level.

In addition, we also test our method on the public real dataset CIFAR 10, which contains 60,000 tiny images (32 × 32, 3 channels) in total. We divide the dataset as 50,000 images as the training dataset and 10,000 images as the test dataset. For the proposed method, we assume the number of users is five and they have the same data size which means the dataset will be distributed equally among them. We set the learning rate to 0.002, the local batch size is 2000, and the activation function is Relu. The simulation results are shown in Table IV and Fig. 5, respectively. The results show that accuracy achieves a little higher for FedSVRG compared with SGD, where the difference is 2.33%. For the loss, the result keeps consistent with previous experiments, where the loss FedSVRG decreases faster and this demonstrates the convergence has been improved. Accordingly, the cost can be less compared with SGD with regard to the same loss or accuracy level.

V. CONCLUSION

In order to decrease the communication cost or the number of interactions between users and MEC servers, this work firstly models the communication cost minimization problem into a finite-sum optimization problem. Then, we propose a FedSVRG based communication efficient scheme to solve the corresponding issue and provide the convergence analysis at the same time. In the simulation part, the proposed FedSVRG are tested for linear regression and logistic regression problems, and the performance is compared with the general SGD method. Finally, we can find that FedSVRG gives better accuracy and converges faster, which correspondingly leads to a lower communication cost.

ACKNOWLEDGMENT

We would like to thank Dr. Mingyi Hong for the discussion and comments that greatly improved the manuscript. This work is partially supported by NSF EARS-1839818, CNS-1717454, CNS-1731424, and CNS-1702850.
REFERENCES


