Research on Federated learning algorithms under communication constraints

Xiaoying Zhong¹, Zhiwei Zhou², Yihe Deng³ and Weikai Liu*
Wuhan Institute of Technology, Wuhan, Hubei, 430200, China
*Corresponding author’s e-mail: 3163266328@qq.com

Abstract: Federated learning is a kind of distributed machine learning, due to the limited communication resources of Internet of things devices, it is very inefficient to train various neural network models through federated learning (FL) mode. In this paper, we propose two new efficient federated learning algorithms: static quantization federated average algorithm (SQFedAvg) and dynamic quantization federated average algorithm (DQFedAvg). They are composed of two parts, which are optimized locally on the client side, and then the updated parameters are quantified and sent to the server, thus reducing computational storage and communication costs and speeding up the training process. Finally, two models: convolutional neural network (CNN) model, a two-layer perceptron (2NN) model and two standard detection data sets (CIFAR-10, MINIST) are used to verify the effectiveness and efficiency of the proposed federated learning algorithm.

1. Introduction
In the past few decades, artificial intelligence technology has made great progress in various real-world applications, The development of these artificial intelligence-based applications relies heavily on a large amount of data (also known as big data). In the current implementation, the collected real-time data is uploaded to the cloud server in a centralized manner and processed by the cloud server in a unified manner[1]. However, the centralized processing of cloud servers has aroused public attention to privacy-sensitive data sharing and may suffer unaccepted long delays and high traffic burdens. In order to alleviate these challenges, it is best to separate model training from the need for remote collection and centralized processing of raw data.

This paper studies a learning algorithm whose goal is to train a high-quality centralized model, while the training data is still distributed on a large number of clients. It is called Federated Learning (FL)[2],[3]. In FL, each client calculates the update to the local model based on its local training data set, and then uploads the updated local model parameters to the FL server. The FL server aggregates updates from all clients to improve the global model, and sends the newly updated global model back to the client. These steps will be repeated many times until convergence or the required accuracy is reached. However, due to the constant communication between the client and the central server to exchange a large number of model parameters, the communication time, the number of communication rounds, and the total number of bits required to transmit data are relatively high, resulting in high communication overhead. In fact, communication overhead and communication efficiency have become the bottleneck for making full use of distributed computing resources to accelerate machine learning model training[5],[6].

In this context, communication-efficient federated learning methods have recently become more and more popular. More and more researchers are focusing on reducing the cost of federated learning
communication [7]. Researchers from Google, Konecny et al., proposed a method to reduce communication costs [8]: the client only transmits the locally calculated model parameter updates to the central server, and does not need to transmit the complete local model. Although this method can reduce communication costs to a certain extent, it cannot be applied to more complex services in practice. On this basis, Reisizadeh et al. proposed a cycle averaging and compression processing method FedPAQ[9]. FedPAQ allows participants to perform local training before synchronizing with the server, and only sends updates from active participants back to the central server, and only the compressed version of the local information is sent, and the server decompresses and aggregates the received compressed version. The two methods have obvious effects in reducing the number of communication rounds and the amount of transmitted data, but they cannot guarantee sufficient accuracy. H. Brendan McMahan et al. proposed to apply gradient optimization algorithms SGD and Adam to the gradient update of the federated average algorithm [10], thereby reducing a certain number of communication rounds, but the effect is not obvious.

Aiming at the communication constraints and algorithm accuracy problems mentioned above, this paper proposes two weighting methods on the basis of the above-mentioned documents: SQFedAvg and DQFedAvg. That is, before the client sends the updated model parameters to the server, a quantization process is performed on the parameters, so that the amount of data that needs to be transmitted is greatly reduced, thereby reducing the number of bits in the number of transmissions. The server performs a decoding operation after receiving the quantized update parameters, and then performs aggregation. These steps will be repeated many times until convergence or the required accuracy is reached.

2. Materials and Methods

2.1. Quantification method

2.1.1. Static quantization. In order to reduce communication overhead, this paper proposes a method to improve communication efficiency-quantification. The simplest quantization is scalar quantization. Usually the operation of scalar quantization can be given by the following description:

$$Q(x) = g\left(\lfloor f(x) \rfloor\right)$$

(1)

Where $x$ is a real number (such as the gradient or the weight in a neural network), $\lfloor x \rfloor$ is the rounded down function, the generated integer $i = \lfloor f(x) \rfloor$, and the sum of $f(x)$ and $g(i)$ is arbitrary Real-valued function. The integer $i$ is sometimes called a quantitative index.

In computer applications, a known quantization is uniform quantization (static quantization). In uniform quantization, "$M$ "bit can be used to represent the accuracy of quantization. When $M = 8$, 32-bit floating-point type can be quantized into 8-bit integer by the following quantization formula:

$$Q(x) = \left\lfloor \frac{2^{M-1}x + 0.5}{2^{M-1}} \right\rfloor$$

(2)

In this formula, 0.5 represents the offset, and 0.5 is added to make each quantized representation located in the middle of the input area, so as to ensure the quantization to the nearest point, thereby reducing the quantization error. Since the distribution of gradient values and weights calculated in reality is basically not uniform, the uniform quantization may bring a relatively large loss of accuracy.
2.1.2. Dynamic quantization. Based on the literature, [7], [8], [9], [10], this paper further considers the method of dynamic quantification.

Let $W_t \in [W_t^{\min}, W_t^{\max}]$ denote the floating-point weight to be transmitted in the neural network, $Q_t \in [Q_{\min}, Q_{\max}]$ denote the quantized integer weight, let $X_{sf}$ denote the scale factor, and $Z_q$ denote the zero point in the quantized value. Then $X_{sf}$ and $Z_q$ can be expressed by the following formula:

$$X_{sf} = \frac{W_t^{\max} - W_t^{\min}}{Q_{\max} - Q_{\min}}$$

(3)

$$Z_q = \begin{cases} 
Q_{\max}, & W_t^{\max} \cdot \frac{1}{X_{sf}} \in (-\infty, 0) \\
Q_{\max} - W_t^{\max} \cdot \frac{1}{X_{sf}}, & W_t^{\max} \cdot \frac{1}{X_{sf}} \in (0, Q_{\max} - Q_{\min}) \\
Q_{\min}, & W_t^{\max} \cdot \frac{1}{X_{sf}} \in (Q_{\max} - Q_{\min}, +\infty) 
\end{cases}$$

(4)

Therefore, the weight can be quantized from 32-bit floating point (FP32) to 8-bit integer (int8) by the following formula (3):

$$Q_t = \left\lfloor Z_q + \frac{W_t}{X_{sf}} \right\rfloor$$

(5)

The participant sends the quantized weight to the server. On the contrary, the server can decode int8 into FP32 by the following formula after receiving the quantized weight:

$$W_t' = X_{sf} \left( W_t - Z_q \right)$$

(6)

Therefore, the dynamic quantization process is equivalent to a process of encoding and decoding [11]. Since there is a decoding operation in the central server, the dynamic quantization error will be greatly reduced.

3. Results and Discussion

In this section, compare the algorithm proposed in this paper with some classic federated learning algorithms, such as federated learning algorithm based on stochastic gradient descent (Fedsgd), federated learning algorithm based on Adam optimization algorithm (Fedadam), and federated average algorithm (FedAvg) for numerical simulation. Reflect the advantages of the algorithm in this paper.

3.1. Experimental setup

The simulation experiment in this paper is based on two very classic detection data sets (MNIST and CIFAR-10) and two models, model 1: a multilayer perceptron, with two hidden layers, each layer has 200 units, abbreviated as 2NN. model 2: a CNN with two convolutional layers, a fully connected layer, and a final softmax output layer. For the learning process, set the static learning rate of all settings to 0.01, the number of quantization bits $b$ to 8, the number of iterations $E$ of the client is set to 10, the mini-batch size $B$ used when the client is updated is set to 10, SGD The momentum is 0.5, and Adam’s weight decay is set to $1e^-4$. This paper establishes two indicators to compare the proposed algorithm with the baseline. The first is to test the convergence accuracy of the global model trained on the data set. The second is the number of communication rounds required to achieve convergence accuracy.
3.2. Results

![Figure 1. Comparison of Convergence Accuracy of different algorithms](image1)

Figure 1. Comparison of Convergence Accuracy of different algorithms

![Figure 2. Comparison of communication rounds required by different algorithms to achieve convergence Accuracy](image2)

Figure 2. Comparison of communication rounds required by different algorithms to achieve convergence Accuracy

3.3. Analysis of experimental results

Test accuracy: For the final convergence accuracy, the convergence accuracy achieved by DQFedAvg is close to FedAvg. Although the accuracy of the model after quantization is not optimal, the accuracy achieved by the optimization algorithm we use is sufficient to meet our needs. Because our goal is to evaluate our optimization method, not to achieve the highest possible accuracy on this task.

Communication rounds required to achieve convergence accuracy: It can be observed in the figure, in testing the CNN model with the MNIST data set, compared with the baseline FedAvg, the number of communication rounds required for SQFedAvg and DQFedAvg to achieve convergence accuracy is reduced by 17.69% and 20.14%, respectively. Compared with Fedsgd and Fedadam, The number of communication rounds required for SQFedAvg to achieve convergence accuracy has been reduced by 48.95% and 54.17%, respectively. The communication rounds required for DQFedAvg to achieve convergence accuracy were reduced by 50.63% and 55.68%, respectively.

In addition, DQFedAvg has a greater improvement in accuracy than SQFedAvg. For example, when using the MNIST data set to test the CNN model, the convergence accuracy of DQFedAvg is 10.75% higher than that of SQFedAvg. Similarly, it can be clearly seen from the figure that on other data sets or other models, the communication rounds are basically the same. In the case of DQFedAvg, the accuracy that can be achieved when it converges is significantly higher than that of SQFedAvg.

4. Conclusions

In this article, for federated learning under communication constraints, two efficient quantization methods are proposed: static quantization and dynamic quantization. The experimental results show that, compared with the traditional federated learning algorithm, the two quantization methods can
save a lot of communication costs, and the accuracy meets certain requirements. Finally, through the comparison of the two quantization methods, it is concluded that dynamic quantification can better meet the needs and is more feasible than static quantification. In addition, studying how to save the communication overhead in the process of federated learning has very important practical significance for the practical application of federated learning, especially on mobile devices with limited communication resources.

References
[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas. (2017) “Communication-efficient learning of deep networks from decentralized data,” [J]. in Proc. Intl. Conf. Artificial Intell. and Stat., Fort Lauderdale, FL: 1273-1282.
[2] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi. (2019) “Error feedback fixes signsgd and other gradient compression schemes,” [J]. in Proc. Intl. Conf. Machine Learn., Long Beach, CA: 3252-3261.
[3] WANG J Z, KONG L W, HUANG Z C, et al. (2020) A Review of Federated Learning Algorithms [J]. Big Data Research, 2020, 6(6):64-82.
[4] G. Qu and N. Li. (2018) “Harnessing smoothness to accelerate distributed optimization,” [J]. IEEE Trans. Control Netw. Syst., vol. 5, no. 3: 1245-1260.
[5] Sindri Magnússon, Hossein Shokri-Ghadikolae, and Na Li. (2020) “On Maintaining Linear Convergence of Distributed Learning and Optimization Under Limited Communication,” [J]. IEEE Transactions on signal processing, vol. 68: 6104-6106.
[6] YANG Qiang. (2020) Federated learning: the last on kilometer of artificial intelligence [J]. CAAI transactions on intelligent systems, 15(1): 183-186.
[7] X. Lian, Y. Huang, Y. Li, and J. Liu. (2015) “Asynchronous parallel stochastic gradient for nonconvex optimization,” [J]. in Advances in Neural Information Processing Systems: 2737-2745.
[8] M. Chiang and T. Zhang. (2016) “Fog and IoT: An overview of research opportunities,” [J]. IEEE Internet of Things Journal, vol. 3, no. 6: 854-864.
[9] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, et al. (2019) “Adaptive Federated Learning in Resource Constrained Edge Computing Systems,” [J]. IEEE Journal on selected areas in communications, vol. 37, no. 6: 1205-1207.
[10] Chenxin Ma, Jakub Konečný, Martin Jaggi, Virginia Smith, Michael I Jordan, Peter Richtárik, and Martin Takáč. (2017) Distributed optimization with arbitrary local solvers. [J]. Optimization Methods & Software, 32(4): 813-848.