Auxiliary Offloading Optimization Method for Hierarchical Federated Learning

Yingzhao Qiu[1(✉)], Shaoyong Guo[1], Tao Shen[2], Yan Feng[3], and Fenhua Bai[3]

[1] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
449242404@qq.com, syguo@bupt.edu.cn
[2] Kunming University of Science and Technology, Kunming, China
shentao@kust.edu.cn
[3] Yunnan Provincial Academy of Science and Technology, Kunming, China
bofenhua@stu.kust.edu.cn

Abstract. Materials genetic data engineering is booming, and with it comes the problem of data islands and other issues faced by the materials genetic database. While federated learning solves the problem of material genetic data engineering, it also has the problems of low model accuracy and high delay. In this paper, an auxiliary offloading optimization method for hierarchical federated learning is designed. Under the hierarchical federated learning system model based on cloud-edge-end, the data offloading algorithm under unbalanced data balances the data volume of each client, reduces the local gradient difference and the model accuracy of hierarchical federated learning is improved; under the condition of limited resources, a low-delay-based hierarchical federated learning offloading strategy is designed to computing offloading, which reduces the delay of hierarchical federated learning in the training and updating process. Through experiment analysis, the auxiliary offload optimization method of hierarchical federated learning proposed in this paper has higher model accuracy and lower delay than traditional federated learning methods.

Keywords: Federated learning · Edge computing · Computation offloading · Delay · Materials genetic data engineering

1 Introduction

As one of the three key research areas of materials genetic data engineering, database plays an important role and significance in the accelerated design of materials [1]. However, the data requirements of different enterprises and governments are different, and the standardized data format requirements are not fully complied with, which is easy to cause problems such as data islands in the material gene databases of enterprises and governments. Federated learning, as a distributed machine learning framework, was first proposed by Google in 2016 [2], which allows data to be kept locally on the client side without transferring the data to a central location, only the model parameters are kept on the client and central Shared between servers, while solving the problem of data silos,

Q. Liu et al. (Eds.): CENet 2022, LNEE 961, pp. 146–155, 2022.
https://doi.org/10.1007/978-981-19-6901-0_16

it reduces the huge communication cost incurred by centralized machine learning in the process of data transmission [3]. However, there are many aspects to be optimized in the application of federated learning in the material genetic data engineering network. The first is the model accuracy problem of federated learning. Compared with centralized machine learning, federated learning allows the client to train its own data set locally, allowing the same epoch training in the local environment of the client with different data sizes and different computing conditions, as well as the local model. The parameters are aggregated and the results are used as the design of global model parameters, which all have an impact on the model accuracy to a certain extent [4]. The second is the delay problem. Different clients have different amounts of data, and it is often the case that the client that needs to be trained needs to wait for a long time for the client that has not been trained to complete the training before the upper-layer server can aggregate the updated model parameters [5]. At the same time, in the dynamic environment of the wireless network where the material genetic data project is located, the status of the client is constantly changing, and some clients often have unstable problems such as disconnection and reconnection, and are prone to resource constraints, such as Insufficient transmission power, computing power, etc.

For these issues, some scholars have made relevant research work. In terms of model accuracy, the author of the paper [6] proposes that a set of sharable data can be transmitted while the training model is delivered to the client, and it is proved that the model accuracy of federated learning can be improved by this. However, the generation of the sharable data requires manual intervention for optimal sampling of the data, which will increase the consumption of human resources. In terms of delay, the author of the paper [7] proposed an asynchronous update framework based on federated learning. All clients will not wait for untrained clients after their own training is completed, but will directly train themselves. The model parameters are uploaded to the upper-layer server. For unstable clients in wireless networks, the paper [8] proposes that the connection with unstable dynamic clients can be cut off directly. For the problem of limited resources, the author of the paper [9] proposed a federated learning algorithm capable of client selection. In each round of training, only some clients are selected for training to improve the computing power of the client. However, in the process of solving the problems raised by the papers [6–9], they all made a trade-off for the original client, and did not fully consider discarding the data contained in the client, which caused the accuracy of the federated learning model. Certain influence.

In this paper, referring to the existing related work, based on the hierarchical federated learning system, an auxiliary offload optimization method is designed, which can realize the dynamic environment of wireless network and face the situation of limited resources and ensure the optimal performance of hierarchical federated learning. High model accuracy and low delay. The auxiliary offloading optimization method analyzes whether to offload computation to a certain client, fully considers the delay problem of hierarchical federated learning, and limits the total transmission power and the total computing power of the client to adapt to the resource constraints in the wireless network condition. And by offloading some data on some clients to the edge server, and arranging for the edge server to use the amount of data offloaded by the client to join the model training process of federated learning with the client, the training efficiency

of federated learning is improved and the cost of federated learning is reduced. Time delay. By balancing the amount of data remaining on the client side and the amount of data received by the edge server, the problem of differences in the local gradients of the same epoch during the model update process between the client and the edge server is reduced, so as to improve the accuracy of the federated learning model. To a certain extent, the problem that the client cannot upload the local model parameters in a centralized manner is alleviated. During the parameter aggregation process, the training time limit of the client is set, so that the dynamic client with timeout and under-time will not perform model aggregation in this round of update, and the connection with the dynamic client with continuous timeout and under-time exceeds a certain number of times will be cut off.

The rest of the paper is arranged as follows: the second part presents the system model of the hierarchical federated learning system; the third part analyzes and designs the auxiliary offloading optimization method; the fourth part designs the experimental content and analyzes the experimental results; The fifth part is the summary and outlook.

2 System Model

2.1 The Hierarchical Federated Learning System Model Based on Cloud-Edge-End

As far as the reality is concerned, each terminal device, with its own storage capacity and computing power, is often unable to perform a large number of computing tasks on its own, while edge servers often have strong computing power and lack data resources for training. In order to make full use of the idle computing resources of edge servers, this paper designs a layered federated learning system based on cloud-edge-end architecture. It is divided into three layers of "cloud-edge-end" in the model architecture, and it is divided into two processes during the operation process. A training update process is performed to address insufficient storage and computing power of end devices. The architecture diagram of the hierarchical federated learning system is shown in Fig. 1.

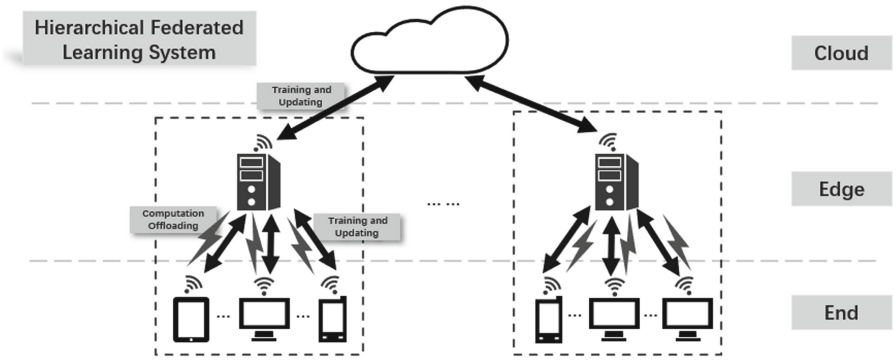

Fig. 1. Hierarchical federated learning system architecture diagram

For the "cloud-edge-end" three-layer structure of the hierarchical federated learning system, "cloud" is the cloud layer; "edge" is the edge server layer, assuming that the set of edge servers it owns is $A^{'}$, there are total A edge servers and each server is represented by the symbol a. "end" is the terminal layer, assuming that there are a total of K clients, each client is represented by the symbol k, and its set is K, and the local data set stored by each client is set as D_k^a, the client's local dataset size represented by D_k^a. Then for each edge server a, there are K_a clients, with the set K_a. The size of the total local data set of the client corresponding to each edge server a is $D_a = \sum_{k \in \mathrm{K}_a} D_k^a$, with the set D_a.

Terminal layer: It consists of mobile phones, computers, wearable devices and other wireless network edge intelligent terminals, which are collectively referred to as clients in this article. These clients are enabled by the integration of advanced sensors with higher computing power and widespread internet availability, generating massive amounts of data every day.

Edge server layer: consists of edge servers. The edge server layer runs on the terminal layer and is connected to the cloud platform. Participate in the computation offloading process and the training update process in a hierarchical federated learning system.

Cloud layer: consists of cloud platforms. The cloud platform does not directly contact the client at the terminal layer, but can indirectly contact the client connected to it through the edge server. However, the cloud platform will not participate in the computation offloadinging process of the hierarchical federated learning system, nor can it obtain the local data of the client and the offloading data of the edge server. It can only collect the local model parameters uploaded by the edge server during the training and updating process, and aggregate them into The global model parameters are delivered to the edge server.

2.2 Computation Offloading Process of Hierarchical Federated Learning System

In the computation offloadinging process of the hierarchical federated learning system proposed in this paper, the client k uploads its own data size and other information D_k^a to the corresponding edge server a, and the edge server a performs a calculation of the offload information, including the need for Calculate the set of uninstalled clients K_a^+ and the specific amount of data U_k^a to be uninstalled by the client k, and send it to each client. After the client k receives the uninstallation information from the edge server a, unload the data volume of U_k^a to the edge server a to complete the computation offloadinging process of the hierarchical federation system. At this time, the offload data set received by the edge server is $\mathrm{U}_a = \bigcup_{k \in \mathrm{K}_a} \mathrm{U}_k^a$, and its size is $U_a = \sum_{k \in \mathrm{K}_a} U_k^a$.

In the process of computing offloading, whether each client needs to perform data offloading, and the amount of data that should be offloaded, this paper will analyze in Chapter 3, and summarize it as an auxiliary offloading optimization method for hierarchical federated learning.

2.3 Training and Updating Process of Hierarchical Federated Learning System

After the client k receives the federated learning training task initialization phase and completes the computation offloadinging process, the client k will use the data left in the local area after offloading to train the local model, and the edge server a will also use it

in the computation offloadinging process. The data unloaded U_k^a by the client k is used for edge model training. After the local training is completed, the client k will upload the trained model parameters ω_k^a to the edge server within the transmission time limit, and the edge server a will also complete the training of the edge model parameters ω_a^e, and will upload the local model parameters. Aggregate with edge model parameters to obtain local model parameters ω_a, and upload them to the cloud platform; after receiving the local model parameters of all edge servers, the cloud platform performs a global model aggregation operation, and finally aggregates the global model. The model parameters ω are downloaded to the edge server a, and then sent to the corresponding client k by the edge server a.

After the client k completes the local model training, upload the local model parameters to the upstream communication stage of the edge server a. Considering that in the dynamic environment of the wireless network, the client is in an unstable state that is prone to disconnection and reconnection. This paper sets a transmission time limit to limit the time-out and out-of-time clients to participate in the training update process of hierarchical federated learning. Assuming that the total number of communication rounds of hierarchical federated learning is N, In the n round of communication, use l_n to indicate the start time of the uplink communication phase, $l_n^{'}$ to indicate the end of the uplink communication phase, and l_k^t to indicate the time when the client k completes the local model training. Set the transfer time limit to:

$$l_n < l_k^t < l_n^{'} \tag{1}$$

In each round of communication, it is determined whether the client k has completed the uplink communication phase within the transmission time limit.

At the same time, set the maximum number of consecutive times that the client is allowed to be out of the transmission time limit to be L. If the number of consecutive times that the client is not within the transmission time limit after a certain round of communication is over L, it will be regarded as an invalid client and will never participate in this Any process of hierarchical federated learning training.

In the aggregation stage, the local model parameters under the edge server a are defined as:

$$\omega_a = \frac{\sum_{k \in \mathrm{K}_a} (D_k^a - U_k^a)\omega_k^a + U_a \omega_a^e}{D_a} \tag{2}$$

where $D_a = \sum_{k \in K_a} D_k^a$ is the size of the local data set of all clients under the edge server a.

After the local model parameter aggregation of the edge servers is completed, it is necessary to perform a global aggregation of the edge model parameters of all edge servers on the cloud platform to obtain a global model and broadcast it to all edge servers. The edge servers broadcast to their corresponding client. Global model parameters can be defined as:

$$\omega = \frac{\sum_{a \in A'} D_a \omega_a}{\sum_{a \in A} D_a} \tag{3}$$

3 Auxiliray Offloading Optimiaztion Method

3.1 Data Offloading Algorithm Under Unbalanced Data

At the beginning of hierarchical federated learning, all clients have the same model parameters and model structure. With the progress of the hierarchical federated learning training, the client generates new model parameters after the training is completed, which are the local model parameters of the client. The local model parameters are uploaded to the edge server and aggregated into local model parameters. In this process, the model parameters have changed, that is, there is a weight divergence phenomenon between the local model parameters and the local model parameters. The main reason for this is the difference in the probability distance between the data distribution on the client and the actual distribution of the entire data set. Different clients have different resource conditions, such as computing hardware, dataset size, etc. If all clients perform the same epoch on the local model with their local datasets of different sizes, the local gradients will differ significantly.

Therefore, under the assumption that the client computing hardware and other conditions are the same, we control the amount of data offloading during the calculation and offloading process of the client, so that the client k corresponding to the edge server a can be calculated and unloaded after the offloading. The size of the data volume on the cilent $D_k^a - U_k^a$ is basically the same as the total size of the offload data U_a received by the edge server. Define K_a^+ as the set of clients that need to be offloaded under the edge server a, there are K_a^+ clients, K_a^- is the set of clients that need to be offloaded under the edge server a, there are K_a^- client, there is:

$$U_a = \frac{\sum_{k \in \mathrm{K}_a^+} D_k^a}{K_a^+ + 1} \tag{4}$$

Therefore, for the client k corresponding to the edge server a, the amount of offloading is:

$$U_k^a = \begin{cases} D_k^a - \frac{\sum_{k \in \mathrm{K}_a^+} D_k^a}{K_a^+ + 1}, & k \in \mathrm{K}_a^+ \\ 0, \quad \mathrm{k} \in \mathrm{K}_\mathrm{a}^- \end{cases} \tag{5}$$

3.2 Low-Delay-Based Hierarchical Federated Learning Offloading Strategy

In this section, the goal of minimizing the delay is to determine the offloading threshold θ_k^a of the hierarchical federated learning under the condition of limited resources, and summarize it as a low-delay-based hierarchical federated learning offloading strategy. As shown in Fig. 2, the delay of hierarchical federated learning consists of several parts. Assuming that the delay in each round of training update process is similar, simply use T_u to represent the offloading delay in the calculation of the offloading process, and use T_m, T_e, T_g represent the model training delay in one round of training update process, the delay of local model parameter aggregation and the delay of global model parameter aggregation.

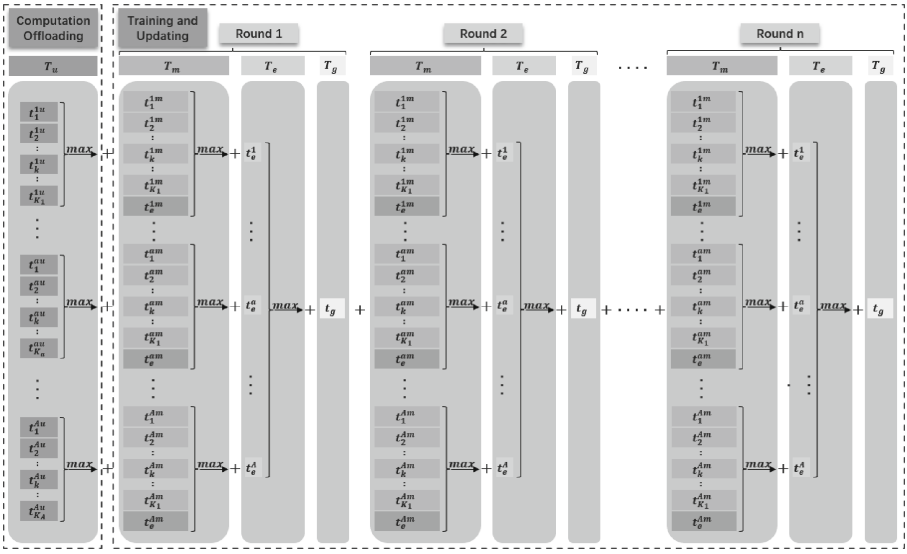

Fig. 2. Delay of hierarchical federated learning system

By limiting the global parameter aggregation delay T_g to a fixed duration, the edge part delay expression corresponding to each edge server a in the hierarchical federated learning process can be deduced as:

$$T^a = \max_{k \in \mathrm{K}_a} \frac{U_k^a}{Blog_2(1 + p_k^{au} g_k^a)} + N\{max\left\{ \max_{k \in \mathrm{K}_a} \frac{\tau C\left(D_k^a - U_k^a\right)}{\sqrt{\frac{p_k^{am}}{\varsigma_k^a}}}, \frac{\tau C U_a}{h_a} \right\} + t_e^a + T_g\} \tag{6}$$

where p_k^{au} is the transmission power allocated to the client k under the edge server a, B is the bandwidth of the link, $g_k^a = \check{g}_k^a / N_0$, $\check{g}_k^a = v|\, r_k^a|^2$, N_0 is the variance of the complex Gaussian white channel noise, $\check{g}_k^a$is the channel gain of the client k under the edge server a, v is the loss parameter of the large-scale fading path, and r_k^a is the edge server a under the client k and the client under the data offload period. A constant channel response is assumed. τ is the number of epochs per round of training, C is the number of CPU cycles required to train 1 bit of data, p_k^{am} and ς_k^a are the computing power and effective capacitance coefficient of the client k under the edge server a, h_a is the edge CPU frequency of the server a. t_e^a is the aggregation delay of local model parameters of the edge server a.

$\sum_{k \in \mathrm{K}_a} p_k^{au}$ and $\sum_{k \in \mathrm{K}_a} p_k^{am}$ are both limited within the total power P_a, which means, $\sum_{k \in \mathrm{K}_a} p_k^{au} \leq P_a$, $\sum_{k \in \mathrm{K}_a} p_k^{am} \leq P_a$. We use Lagrangian function and KKT condition to solve it, and get the offloading threshold θ_k^a of the client k under the edge server a. Therefore, we can set the algorithm of the computing offloading part of the hierarchical federated learning.

4 Experiment and Result Analysis

4.1 Experiment Setup

In this section, experiments are carried out to verify the improvement of the model accuracy and delay of the hierarchical federated learning system AOHF using the auxiliary offload optimization method proposed in this paper. The dataset used in the experiment is the MINST dataset of the National Institute of Standards and Technology, which contains handwritten Arabic numerals 0–9, and performs the image classification task of numerals 0–9. Among them, 60,000 training samples and 10,000 test samples were obtained from 250 different high school students and Census Bureau staff in a 1:1 ratio. The experiment is based on the open source python machine learning library PyTorch. When simulating AOHF, the total number of edge servers is set as $A = 5$, and the total number of clients is $K = 200$. At the same time, in terms of experimental conditions, some resources are limited, and the offline time of the client is randomly set to simulate the dynamic environment under the wireless network. Set the power constraint of the system to $P_a = 1$ W, the number of CPU cycles required to train 1-bit data is $C = 1000$, and the CPU frequency of each edge server during the training update process is set to $h_a = 10^{11}$ times/s, The effective capacitance coefficient of each client is set to $\varsigma_k^a = 10^{-18}$, the bandwidth is set to B = 10 Hz, and the transmission time limit is set to $l_n = 0.1\,\text{s} < l_k^t < l_n^{'} = 2\,\text{s}$. The maximum number of consecutive times that the terminal is not within the transmission time limit is $L = 3$.

4.2 Model Accuracy

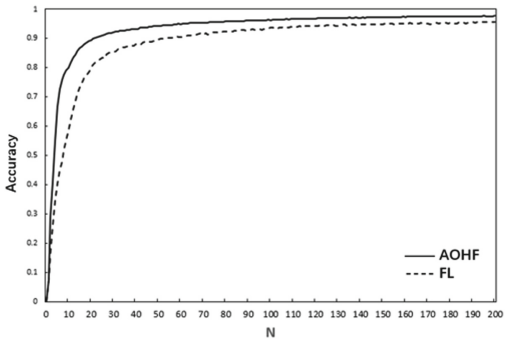

Fig. 3. Comparison of accuracy between AOHF and FL

The model accuracy of federated learning is reflected in the accuracy rate. As shown in Fig. 3, compared with traditional federated learning, AOHF has a certain improvement in accuracy. Compared with the first subsection of the third part of this paper, the judgment made when analyzing and discussing the data offloading algorithm under unbalanced data is: Consistently, the client-owned dataset size is one reason why local gradients will vary significantly during training and updating.

4.3 Delay

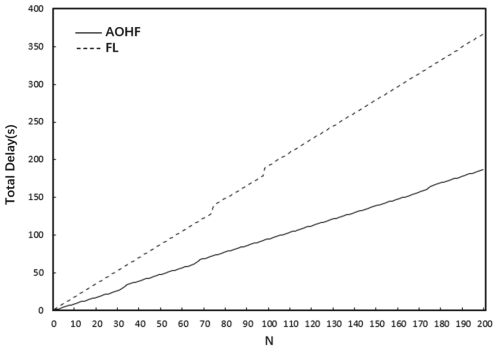

Fig. 4. Comparison of total delay between AOHF and FL

Figure 4 shows the delay comparison between AOHF and traditional federated learning. It can be seen from the figure that, on the one hand, the delays of AOHF and traditional federated learning gradually increase with the total number of communication rounds, and basically show a linear relationship. It shows that the delay of each communication round is basically the same; on the other hand, it can also be clearly seen that the delay of AOHF is lower than that of traditional federated learning, which is consistent with the analysis result of determining the AOHF offloading threshold.

5 Conclusion

In this paper, an auxiliary offloading optimization method for hierarchical federated learning is designed, which can solve the data island problem of the material gene database to a certain extent in the dynamic environment of the resource-constrained wireless network in which the material gene data project is located, and at the same time guarantees The continuity of hierarchical federated learning, and through the data offloading algorithm under unbalanced data and the low-delay-based hierarchical federated learning offloading strategy, the model accuracy of hierarchical federated learning is improved, the delay is reduced, and the material gene is improved. Data engineering data learning and training efficiency.

Acknowledgment. This work was supported in Major science and technology special project of Science and Technology Department of Yunnan Province (202002AB080001–8).

References

1. Yang, L., Hang, S., Chai, F., Luo, X., Duan, L.: Material database and application status of data mining technology. Materials China **38**(7), 672–681 (2019)

2. McMahan, H.B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: 20 th International Conference on Artificial Intelligence and Statistics, pp. 1273–1282. Fort Lauderdale, Florida, USA (2017)
3. Dhakal, S., Prakash, S., Yona, Y., Talwar, S., Himayat, N.: Coded federated learning. In: 2019 IEEE Globecom Workshops, pp. 1–6. Hawaii, USA (2019)
4. Liu, S., Yu, J., Deng, X., Wan, S.: FedCPF: an efficient-communication federated learning approach for vehicular edge computing in 6G communication networks. IEEE Trans. Intell. Transp. Syst. **23**(2), 1616–1629 (2022)
5. Xie, C., Koyejo, S., Gupta, I.: Asynchronous federated optimization. In: 12th Annual Workshop on Optimization for Machine Learning, pp. 28–39, Online (2022)
6. Jamali-Rad, H., Abdizadeh, M., Singh, A.: Federated Learning With Taskonomy for Non-IID Data. In: IEEE Transactions on Neural Networks and Learning Systems, pp. 1–12 (2022)
7. Chen, Y., Ning, Y., Rangwala, H.: Asynchronous Online Federated Learning for Edge Devices with Non-IID Data. In: 2020 IEEE International Conference on Big Data, pp. 15–24, IEEE, Online (2020)
8. Idrissi, M.J., Berrada, I., Noubir, G.: FEDBS: Learning on Non-IID Data in Federated Learning using Batch Normalization. In: 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI), pp. 861–867, IEEE (2021)
9. Yang, H., Fang, M., Liu, J.: Achieving linear speedup with partial worker participation in non-IID federated learning. In: 2021 International Conference on Learning Representations, pp. 1–23 (2021)