Convergence Analysis and Deep Learning Experiments of Federated Learning

Kuiye Ding

Abstract—Deep learning has experienced explosive development in recent years. And data determines the upper limit of deep learning. One of the reasons why Federated Learning was proposed is to solve the problem of "data island". Federated learning can also fully utilize the computing power of edge devices and protect data security. The work of this paper is: 1. Analyze the convergence of federated learning; 2. Experiment federated learning algorithms through simulation. Finally, I demonstrate that the loss function of federated learning possesses convergence. In addition, through experiments, I verified that on the MNIST and CIFAR-10 datasets, the performance of federated learning is quite close to that of general deep learning methods, but with advantages in terms of data privacy protection and edge device arithmetic utilization.

Index Terms—Federated Learning, convergence, simulation

I. Introduction

Federated learning is an innovative distributed machine learning technique that enables multiple participants to collaboratively train a shared model while maintaining data privacy [1]. In this framework, each participant, such as a mobile device or a distributed server, uses its locally stored data to train the model and sends learned model updates (e.g., gradients) to a central server instead of the raw data, thus preserving data privacy and reducing communication costs. The central server aggregates these updates to improve the global model and then distributes them back to individual participants for the next round of training [2]. This approach is not only adaptable and capable of handling non-independent and co-distributed data, but also easily scales to thousands of clients, enhancing the generalization ability and robustness of the model. Federated learning has shown potential for a wide range of applications in a variety of domains, including mobile computing, healthcare, finance, and IoT, due to its advantages in protecting data privacy, reducing data transmission, and enhancing model generalization capabilities [3]. The architecture of federated learning is shown in Figure 1 [4].

In the exploration of this paper, we focus primarily on the convergence properties of federated learning. Specifically, we utilize the gradient descent method, combined with the Lipschitz, to develop a detailed formula derivation process. After rigorous logical arguments, we are finally able to confirm that in the case where all devices are involved in training, both the full gradient descent method and the gradient descent method on a single device are.

In this research paper, I have selected three classical algorithms, CNN (Convolutional Neural Network), MLP (Multi-Layer Perceptual Machine), and Logistic Regression, as the benchmark models, and implemented a federated learning framework based on them. With this framework, I compare the difference in model accuracy between two scenarios with and without federated learning. In addition, I explore the specific impact of key parameters in federation learning on the effectiveness of federation learning.

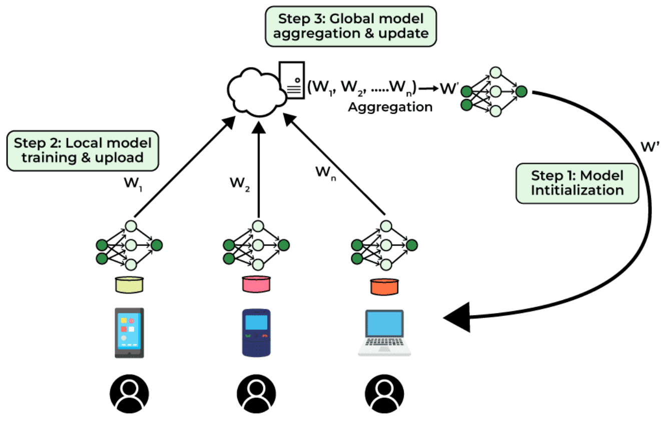
Step 3: Global model aggregation & update
$(W_1, W_2, ...W_n) \rightarrow W'$
Aggregation
Step 2: Local model training & upload
W_1
W_2
W_n
W'
Step 1: Model Intitialization

Fig. 1. Federated Learning

II. Convergence Analysis

In this section, the convergence of federated learning is analyzed, ultimately concluding that the gradient of federated learning is ultimately convergent.

A. Problem Definition

In federated learning, the goal is to minimize the loss function Equation 1.

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \tag{1}$$

Where f is the global loss, F is the loss per client, K denotes the number of clients, n_k denotes the size of the data on the k client, and n denotes the size of the total data.

Two general approaches exist to improve the efficiency of federated learning [1]: 1. Increasing parallelism, i.e., allowing more clients to work independently between each communication round; and 2. Clients increase the amount of computation to handle more gradient computations.

SGD (Stochastic Gradient Descent) and its various variants have indeed occupied a pivotal position in the widespread practice of deep learning [5]. Given this background, in this paper, SGD is chosen as the training strategy when exploring the convergence analysis of federated learning.

The central server aggregates the gradients on each client and applies the updates as shown in Equation 2.

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k \tag{2}$$

Where g_k denotes the gradient descent of client k, and η denotes the learning rate of the matter.

$$\sum_{k=1}^{K} \frac{n_k}{n} g_k = \nabla f(w_t) \tag{3}$$

And in conjunction with Equation 3, so we can get Equation 4.

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla f(\mathbf{w}_t) \tag{4}$$

The gradient descent is computed separately for each client and finally summarized by the central server and returned to each client, this approachFederatedAveraging (orFedAvg). [1] The amount of computation is controlled by three key parameters: C, the fraction of clients that perform computation on each round; E, then number of training passes each client makes over its local dataset on each round; and B, the local minibatch size used for the client updates.

B. Assuming

- Assume that the loss function $F_k(\mathbf{w})$ for each client is convex, but not strongly convex.
- The gradient $\nabla F_k(\mathbf{w})$ of $F_k(\mathbf{w})$ is Lipschitz continuous.
- Full device participation.

C. Convergence

I will use gradient descent to optimize the global loss function. The update rule for gradient descent is:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla F(\mathbf{w}_t) \tag{5}$$

where $\mathbf{w}$ is the model parameter at the t iteration, η is the learning rate, and $F(\mathbf{w}_t)$ is the client gradient of the loss function.

According to the Lipschitz continuum (or Taylor's formula expansion), Equation 6 can be obtained. where L is a constant.

$$F(\mathbf{w}_{t+1}) \leqslant F(\mathbf{w}_t) + \langle \nabla F(\mathbf{w}_t), \mathbf{w}_{t+1} - \mathbf{w}_t \rangle + \frac{L}{2} ||\mathbf{w}_{t+1} - \mathbf{w}_t||^2 \tag{6}$$

Substituting into the updated Equation 5 for gradient descent yields:

$$F(\mathbf{w}_{t+1}) \leqslant F(\mathbf{w}_t) - \eta ||\nabla F(w_t)||^2 + \frac{\eta^2 L}{2} ||\nabla F(\mathbf{w}_t)||^2. \tag{7}$$

Assuming that $\mathbf{w}^*$ is the optimal parameter of the model while satisfying $0 < \eta < \frac{2}{L}$,one obtains the linear convergence of the objective function values in Equation 8.

$$F(\mathbf{w}_{t+1}) - F(\mathbf{w}^*) \leq \left(1 - \frac{\eta L}{2}\right) (F(\mathbf{w}_t) - F(\mathbf{w}^*)) \tag{8}$$

The recurrence relation Equation 8 leads to Equation 9.

$$F(\mathbf{w}_{t+1}) - F(\mathbf{w}^*) \leq \left(1 - \frac{\eta L}{2}\right)^t (F(\mathbf{w}_0) - F(\mathbf{w}^*)) \tag{9}$$

$F(\mathbf{w}_0)$ is the objective function value at the initial moment; $F(\mathbf{w}^*)$ is the objective function value corresponding to the optimal solution.

Equation 9 shows that the objective function value $F(\mathbf{w}_t)$ is gradually approaching the optimal solution $F(\mathbf{w}^*)$ with the number of iterations t, and that the rate of convergence is controlled by the learning rate η and the constant L of the objective function. With this formula, it can be seen that the gradient descent method will converge at an exponential rate in each iteration step, as long as we choose the appropriate learning rate η.

III. Experiments

The graphics card model used for the experiments in this paper is NVIDIA A100-SXM4-80GB. The number of clients is set to 100, each client epochs is set to 5, batch size is set to 10, and learning rate is set to 0.01.

The experiments in this paper consist of three families of models on two datasets [1]: 1) a simple two-layer perceptron with 2 hidden layers with 200 units each and ReLU activation, which we call the NN. 2) a CNN with two 5x5 convolutional layers (the first with 32 channels and the second with 64, each followed by 2x2 max pooling), a fully connected layer with 512 units and ReLU activations, and a final softmax output layer.3) A logistic regression function with a linear layer and an activation function sigmod. In this paper, we use two ways to distribute the data to the clients: IID, where the data is shuffled and then distributed to 100 clients, each of which receives 600 examples; and non-IID, where we first sort the data by numerical labels, divide it into 200 fragments of size 300, and assign 2 fragments to each of the 100 clients.

The code for the experimental part of this paper is borrowed from [6] [7]. The code for this article has been open sourced to https://github.com/Day333/federated-learning-test.

As shown in Table I, it is the CNN model that performs best on both the MNIST dataset and the CIFAR-10 dataset, which is to be expected.CNN is not only a larger

TABLE I
Model Accuracy and Loss for Different Datasets

	Eopch	10		50		100	
Model	Dataset	ACC	LOSS	ACC	LOSS	ACC	LOSS
LR	MNIST	90.17	15.76	90.94	15.57	91.2	15.53
	CIFAR-10	34.33	20.52	35.55	20.44	35.9	20.42
MLP	MNIST	95.74	1.37	97.05	1.01	97.31	0.99
	CIFAR-10	41.73	16.45	44.98	15.65	45.58	15.42
CNN	MNIST	98.51	0.52	98.98	0.33	99.04	0.3
	CIFAR-10	58.27	12.05	66.22	9.73	71.61	8.33

TABLE II
Federated Learning Model Accuracy and Loss for Different Datasets

	Eopch	10		50		100	
Model	Dataset	IDD	Non-IDD	IDD	Non-IDD	IDD	Non-IDD
LR	MNIST	89.5	82.64	90.49	89.25	90.81	88.59
	CIFAR-10	35.97	27.86	37.86	32.84	38.72	33.74
MLP	MNIST	94.66	74.49	97.19	92.99	97.77	94.67
	CIFAR-10	46.04	20.12	51.3	36.92	53.03	42.58
CNN	MNIST	96.73	82.24	98.43	89	98.82	94.56
	CIFAR-10	45.95	20.31	52.89	41.68	42.38	55.36

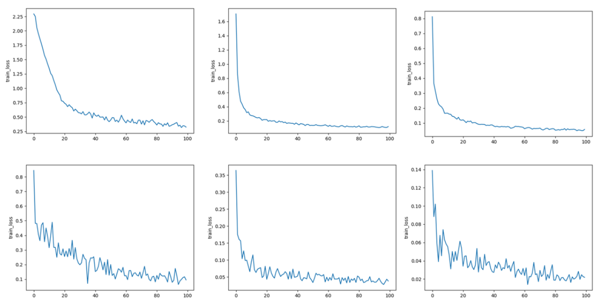

Fig. 2. Loss Comparison. The first line is IID and the second line is Non-IID.

model with larger parameters, but also a more advanced approach. The highest obtained on the MNIST dataset is 99.04% and the best performance on the CIFAR-10 dataset is 71.61%. While the best results for traditional machine learning models were 91.2% and 35.9%. Next, use the federated learning approach to see how the results differ.

The experimental results in Tables I and II show that federated learning is very close to traditional deep learning methods in terms of effectiveness. Taking the MNIST dataset as an example, the accuracy achieved with the CNN method is 99.04%, while the accuracy under the federated learning framework reaches 98.82%, with a gap of only 0.22 percentage points between the two.

Meanwhile, in terms of data distribution strategy, the IDD (Independent and Distributed Data) method shows a clear advantage over the Non-IDD (Non-Independent and Distributed Data) method. This is visualized in the comparison of loss curves presented in Figure 2. The figure not only reveals the superiority of the IDD approach in terms of loss reduction, but also provides a profound reminder that when implementing federated learning, we should ensure that the data of each client remains independently distributed as much as possible to maximize the potential of federated learning and enhance its effectiveness.

In addition, I conducted experiments by adjusting the parameter C to 0.1,0.5 and 1 respectively. With all other conditions being consistent, the accuracy is 94.64%, 94.78% and 94.78% respectively. There is a slight increase in accuracy from $C = 0.1$ to $C = 0.5$ and no change to $C = 1$. But the time consuming increases exponentially, so it is not better to have more number of clients.

IV. Conclusion

This study confirms the convergence of federated learning and its effectiveness through theoretical analysis and deep learning experiments on MNIST and CIFAR-10 datasets. The results show that federated learning achieves accuracy close to traditional deep learning, with CNN outperforming other models. The IID data distribution strategy proves more effective than non-IID, emphasizing the importance of data handling in federated learning. Future work should focus on scalability, non-IID data challenges, and integrating complex models to enhance federated learning's performance.

Acknowledgment

Thanks to the authors of all the references.

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas, "Communication-efficient learning of deep networks from decentralized data," vol. 54, pp. 1273–1282, 20–22 Apr 2017.

[2] wikipedoa, "Federated learning," $https://en.wikipedia.org/wiki/Federated_learning$, 21 October 2024, at 14:01 (UTC).

[3] Yesha Shastri, "A Step-by-Step Guide to Federated Learning in Computer Vision," https://www.v7labs.com/blog/federated-learning-guide, Feb 3, 2023.

[4] ," https://www.geeksforgeeks.org/collaborative-learning-federated-learning/, 20 Mar, 2024.

[5] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao, "Advances and open problems in federated learning," Foundations and Trends® in Machine Learning, vol. 14, no. 1–2, pp. 1–210, 2021.

[6] shaoxiongji, "federated-learning," https://github.com/shaoxiongji/federated-learning, 2018.

[7] litian96, "FedProx," https://github.com/litian96/FedProx, 2019.