Transactions on Computer Science and Intelligent Systems Research
ISSN: 2960-1800, eISSN: 2960-2238 | Volume 5
AIDML 2024

Multi-Master Model: Better Way to Calculate the Final Weights

Bowen Ma

College of Arts and Science, New York University, New York, United States

bm3150@nyu.edu

Abstract. Under the current age in which data privacy and security become more and more important, as the short-coming of traditional distributed machine learning algorithms, the Federated Learning algorithm (FL), became more apparent. We need a better algorithm framework to handle more complex training scenarios. The focus of this paper is on multi-master Federated Learning, specifically on the last part of how to handle the weights from different master nodes. In this algorithm, instead of using only one central server to receive all weights calculated by the clients, we use multiple master nodes, each taking care of a portion of the data. There is no interaction between clients under different master nodes. Finally, the master nodes process their weights together to get a final weight. In this paper, the number of master nodes is 3, and there are 3 approaches we propose: 1) do a simple average of the weights. 2) do a weighted average. 3) apply the traditional Fedavg algorithm on these weights. As a result, we found that for simple dataset, three approaches to get final weights in multi-master FL perform similarly and are no worse than traditional FedAvg, while for complex real-world data, approach 3 has a slightly better result.

Keywords: Multi-Master model; weights; federated learning.

1. Introduction:

In the current era of artificial intelligence, although we can't say the development of applied algorithms in different fields are close to maturity, they are at least steadily developing. However, the foundation of all these technologies is data. The quantity, quality, and diversity of the data resources are one of the key factors that limit the development of artificial intelligence. However, most data that was used in the AI field is highly sensitive, and they exist in isolated form. Especially in the healthcare industry, as medical data and patients' information are highly sensitive and personal and they are often collected and housed across various medical institutions. It's extremely difficult for different institutions to upload their data to one single server, since the data that contains privacy information might leak out. This presents a significant challenge for the application of artificial intelligence, as traditional AI methods can not address the privacy concern. Additionally, with the introduction of various data privacy protection laws, such as General Data Protection Regulation in 2016, there is an increasing demand for AI solutions that prioritize privacy protection in order to comply with legal requirements.

Given these challenges of data privacy, Federated Learning has become increasingly popular in recent years. It's an encrypted distributed machine learning model, with the emergence of algorithms FedAvg [1], it has successfully met the requirement of training models without sharing data publicly. The most widely used Federate Learning setting is the "server-client structure, which assumes that data is held by clients (ranging from individual mobile devices to entire companies or organizations), while a central parameter server is responsible for the federated learning model learning tasks. During the model training process, training data is stored locally on each client and is not directly shared during training." Its structure is illustrated by Figure 1.

However, when learning on non-iid data, the accuracy of FedAvg significantly decreases [2, 3] .That's when the alternative structure, such as FedProx [4], SCAFFOLD [5], and the focus of this paper: multi-master federate learning structure comes in. Instead of using one central server to handle the federated learning task, it will use multiple master nodes, each handles a portion of the local updates from the clients. Its structure is illustrated by Figure 2 [6]

This paper will focus on the last part of the process, that is, what's the good way to combine the weights from different Master Nodes together to get the best result. We have three potential solutions: 1. Simply averaging their values. 2. Do the weighted average. 3. Perform another round of FedAvg algorithm between the central servers.

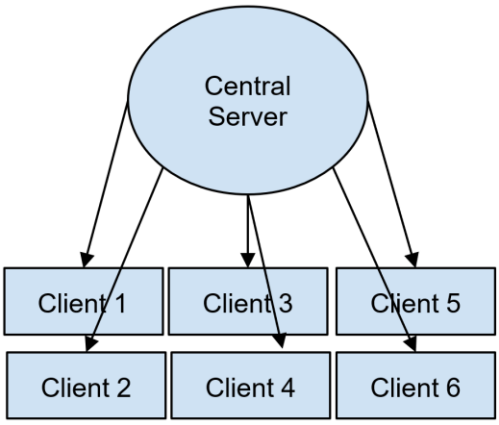

Figure 1. Traditional structure [1]

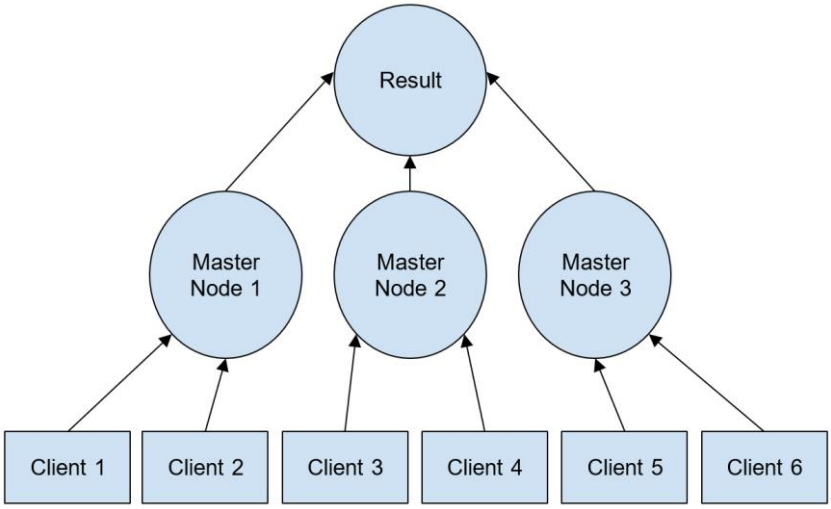

Figure 2. Multi-Master structure [6]

2. Motivation

We will use MNIST dataset to see how the traditional FedAvg algorithm performs poorly on heterogeneous data. In order to simulate real life non-iid scenarios, we will separate the data by Dirichlet partition and Diversity partition. The Dirichelt partition is to partition the indices of samples in the original dataset according to the Dirichlet distribution of the particular attribute. The Diversity partition is to partition the indices of samples according to its types of particular attribute. As Figure 3 shows, for the Dirichlet partition, each user has a limited number of types, and their distribution is highly unequal. So this corresponds to Distribution-based label imbalance. As Figure 3 shows, in the Diversity partition, the distribution of each type in the workers is the same, so it corresponds to Quantity-based label imbalance. Based on Figure 3, we can see that as the data become more heterogeneous (the hyperparameter decreases), the traditional FedAvg algorithm performs worse. It takes a longer time to converge and also results in a lower score compared to IID. That's because of the Client Drift. When the data is IID, the average model weights will be close to the optimal weight w*, because it has equal distance between local optimal w*1 and w*2. However in the non-IID case, the optimal weight w* does not have equal distance between the local optimals, so if we simply average the local weights, the result is not what we want (Figure 4) [3]. Because of the limitation of

FedAvg on heterogeneous data, we will be focusing on another structure that was designed to fix this problem: multi-master Federate learning, and try to make improvements on that model.

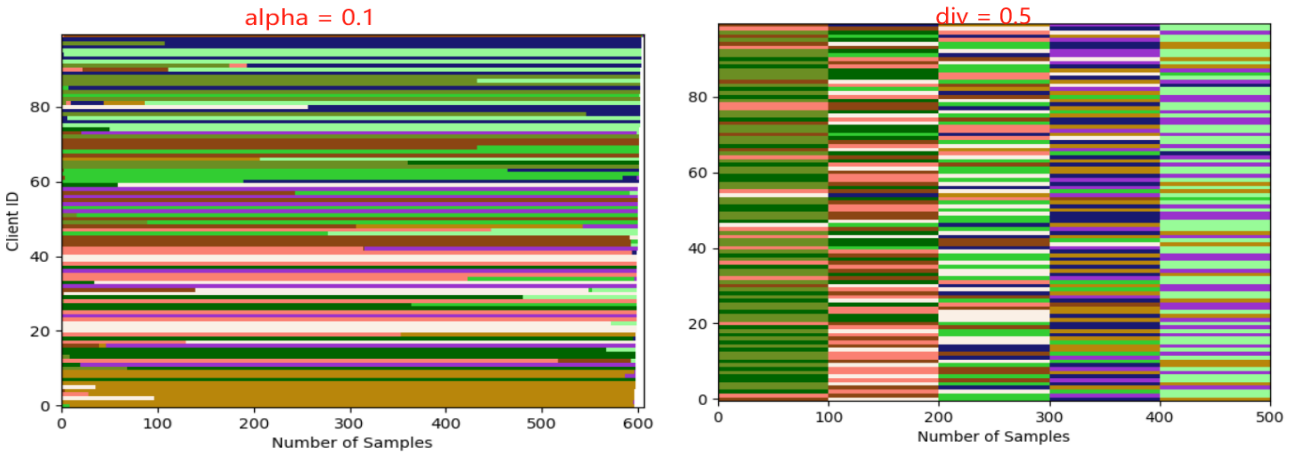

Figure 3. Dirichlet distribution and Diversity Distribution [3]

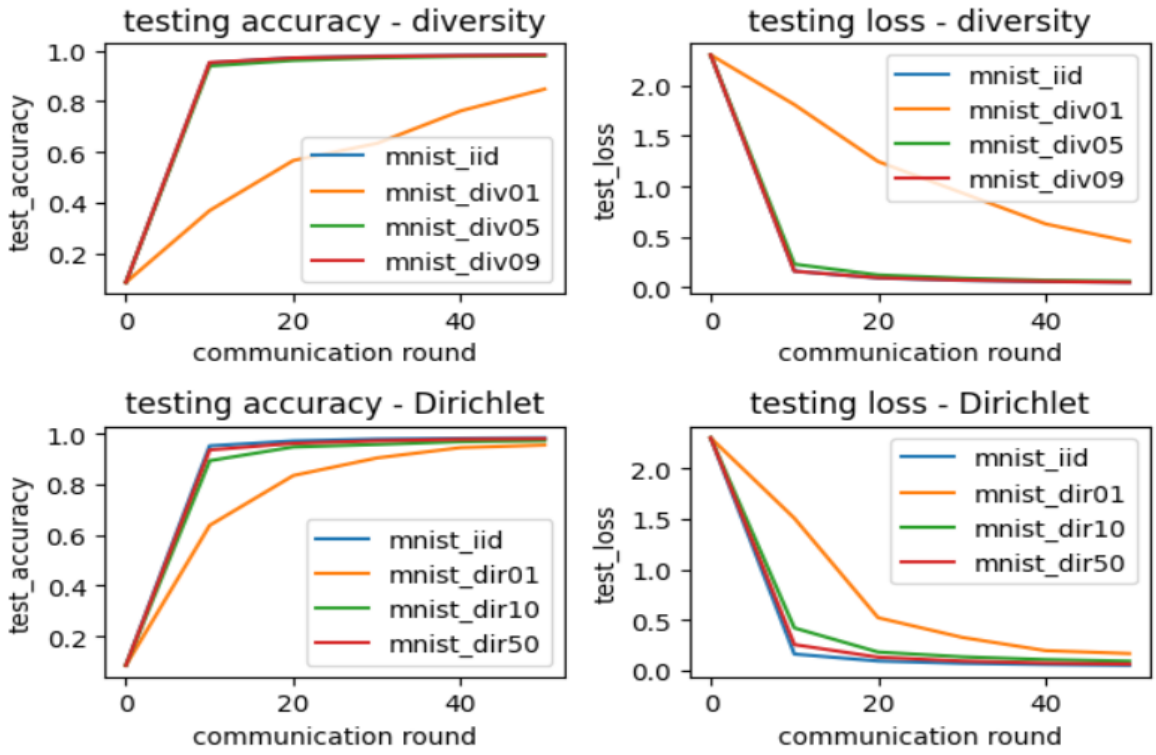

Figure 4. Ttraditional results [3]

3. Method

The main problem for the traditional Fed Avg algorithm on Non-IID data happened on the stage when the central server combined all weights from different clients together. So the focus of the paper is also on this part. For the Multi-Master model, when multiple master nodes receive value from their individual clients, there are three ways we could use to calculate the final weights.

3.1. Three ways to calculate the final weights:

Calculate the arithmetic mean of their weights.

Calculate the weighted average, based on the different ratio of data in the data distribution.

Perform another FedAvg algorithm for weights on different master nodes.

3.2. Structure of the model:

1) Master Node:

The job of master node is to monitor the federate learning process, and keep track of the participating clients. In the end, it also has the responsibility to aggregate all client's upload models [7].

2) Client Nodes:

The job of client nodes is to simulate the federate learning process, each node will be given a specific portion of the data that they need to be trained on. After finishing the training phase, all client nodes need to send the trained model weights back to the master node.

4. Experiment

4.1. Experiment setting

First we will be using MNIST data to perform multi-master federate learning, and we will use the Dirichlet partition with alpha = 0.1 to simulate the non-iid data distribution. Each user will only have two types of images as their data and the distribution of these two types will be highly randomized. We will use 3 methods mentioned above to handle the weights in each different master node.

Second, we will use a real life dataset from Kaggle about hospital's data for predicting whether a patient will survive or not, implement both Centralized Federated learning and the Multi-Master Federate learning, and use the same 3 methods to get the final weights and see the accuracy differences.

4.2. Experiment Dataset

1) MNIST Dataset : It's a classical handwritten digit recognition dataset of digits from 0 to 9, each represented by a 28 x 28 pixel grayscale image.

2) Kaggle real-life dataset: The data contains patients' private, sensitive information and also the information of the hospital they belong to. We want to train a model to predict a person's morality without sending his information to the central master node. For the MNIST dataset, there is nothing we need to preprocess. For the kaggle dataset, it contains categorical features and NA value. I first create one-hot encoding for these variables, and then just drop the rows that have NA value, which decrease the data size from 91713 to 76935.

4.3. Training

1) MNIST data training: This paper first computed the traditional centralized federated learning algorithm, each worker performs training independently. Then they send the computed weights back to the central server. The central server will combine the weights together to get the best final results.

Then for the multi-master case, the number of master nodes is set 3. After separating the data with the Dirichlet partition, this work randomly distributes the value into 3 groups to simulate the multi-master scenario. Each master performs its own federate learning, and when all three master nodes get the weights, this paper uses the three methods in III.B to get the final results.

2) Kaggle data training: In the centralized federated learning case, each worker has its own preprocessed data and performs training independently, after calculation, each worker will send the computed weights, instead of the data, back to the master node. Then the only master node will combine their weights together and reach the best local updates, while preserving the data privacy. In traditional federate learning algorithm, the way master node achieve this is by using User Data Size Weighted Average, which involves picking samples from the Dirichlet distribution and compute the posterior mean and variances, this helps the model to be more stable and less likely to be affected by some sudden change.

For the multi-master federated learning structure, the number of master nodes is equal to the number of different hospitals. We first need to separate data into different clusters. One cluster represents data coming from one specific hospital, so the data inside each cluster all share similar characteristics and

format, and each data's property will not be mitigated with other data. It also provides fault-tolerance to the model. The failure of one master node will not crush the entire training process. Inside each cluster, we again use the Dirichlet partition to separate the data. Then, similar to the Fed Avg algorithm, inside each cluster, each worker will do the training, sampling the posterior distribution using Dirichlet distribution and the calculation of means and variances. In the end, the workers send the weights back to the corresponding master node. Because all data inside one cluster share similar data properties and format, the training process will be facilitated. After all the master nodes compute the average weights for each parameter, we can perform three methods in 3.2 to aggregate these cluster-level weights to compute the final weights.

4.4. Experiment Result

In this section, we will be making the analysis of the results we get after performing multi-master federate learning on MNIST dataset and kaggle dataset using 3 different approaches to handle the final weights.

For the traditional MNIST dataset, because it is a simple dataset and we separate each master node with equal size, the result of using the simple average and using the weighted average to get final weights are the same. And the result of doing another Fedavg algorithm to get the final weights is a little better than the other two results (Table I).

Table 1. Accuracy results for MNIST dataset

MNIST	**Centralized**	**Approach 1**	**Approach 2**	**Approach 3**
Accuracy	93%	94.61%	94.61%	94.89%

We can see that there is no big difference between the traditional Fedavg algorithm and the multi-master Fedavg algorithm. This is not a surprise result. Since we know for the MNIST dataset, it only contains 28 x 28 grayscale pixels and the final digits. There is no outlier, no special case. And the dataset is properly organized. Even though the paper separate the data in a heterogeneous way, that is, each client only contains a few digits' images, when they get distributed to different master nodes, the master nodes will have sufficient data that they can learn from so the performance will not be too bad. From here, we can make one conclusion that when the dataset is simple and small, the advantage of a multi-master algorithm is not obvious, but still, it's not worse than the traditional Fedavg algorithm, no matter which approach you use. Additionally, among three different approaches, approach 3 has a little higher accuracy than the other 2 approaches.

For the kaggle dataset about hospital's data for predicting whether a patient will survive or not, we could see the accuracy of three different approaches (Table II). Using a traditional centralized federated learning algorithm, and getting final weights by averaging the different weights from different master nodes has accuracies lower than 90%. And using another Fedavg algorithm for the final weights performs the best. It's understandable that Centralized and Approach 1 perform not so well, because each hospital has a different amount of data, so when their specific master node calculates the value, it will have different weights. Then if we simply do an algorithm average, the performance will not be so good. The node with higher weights will dominate the results [8-10]. It seems like performing another round of Fed Avg algorithm among the master nodes could have a better performance.

Table 2. Accuracy results for Kaggle dataset

kaggle data	**Centralized**	**Approach 1**	**Approach 2**	**Approach 3**
Accuracy	88%	83%	83.2%	93.68%

5. Conclusion

This paper performed the traditional centralized federated learning algorithm and the multi-master federate learning algorithm on MNIST dataset and one Kaggle real life dataset. Because of the poor performance of the traditional Fed Avg algorithm on heterogeneous data, this paper wants to find a better model that could handle more complicated situations. So the focus is on the Multi-Master model, and the crucial aspect of this paper is the computation of the final weights among all master nodes. This paper tried three different methods to calculate the final weights, and make a comparison of their accuracy. In the end, the accuracy table proves that for a simple dataset such as MNIST, the multi-master algorithm's performance is similar to the traditional centralized federated learning algorithm, with three methods performing quite similar. For the complex dataset, the result shows that while providing fault tolerance, privacy preservation and data security, using approach 3, which is to perform another round of Fed Avg on all weights in Master Nodes can have the best result.

Reference

[1] H. B.McMahan, , E. Moore, Ramage,. Communication-efficient learning of Deep Networks from Decentralized Data. arXiv.org. https://arxiv.org/abs/1602.05629 ,2023

[2] S.,Bharati, M. R. H Mondal, P. Podder, Federated learning: Applications, challenges and Future Directions. arXiv.org. https://arxiv.org/abs/2205.09513 2022

[3] Q. Li,,Y. Diao, Q. Chen, Federated learning on Non-IID Data Silos: An experimental study. arXiv.org. https://arxiv.org/abs/2102.02079 2021

[4] T. Li,,A. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar). Federated optimization in Heterogeneous Networks. arXiv.org. https://arxiv.org/abs/1812.06127 2020.

[5] S. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich,. Scaffold: Stochastic controlled averaging for Federated learning. arXiv.org. https://arxiv.org/abs/1910.06378 2021.

[6] N. Bhuyan, S. Moharir, S. Multi-model federated learning. arXiv.org. https://arxiv.org/abs/2201.02582 2022

[7] T. Sun, D.Li, B. Wang, Decentralized Federated averaging. arXiv.org. https://arxiv.org/abs/2104.11375 2021..

[8] Y. Zhao, M. Li L.,N. Suda, N., Civin, D., & Chandra, V. (2022, July 21). Federated learning with non-IID data. arXiv.org. https://arxiv.org/abs/1806.00582

[9] J. Hasan, Security and privacy issues of Federated Learning. arXiv.org. https://arxiv.org/abs/2307.12181

[10] Z. Qu, X. Li, X., Xu, J., Tang, B., Lu, Z., & Liu, Y. On the convergence of Multi-Server Federated Learning with Overlapping Area. arXiv.org. https://arxiv.org/abs/2208.07893 2022