Federated Learning
Exploring Aggregation Algorithms

Aicha Slaitane[*]

Abstract

With the increase in the application of machine learning in all aspects of life, concerns about the privacy of the training and testing data have emerged. In response to these concerns, a new technology was introduced: Federated Learning. Federated learning is a type of distributed machine learning. Instead of sending the available datasets to a central server for training, the model parameters are sent to data providers. Each provider (called client) trains the model locally using its local dataset and only sends the updated model parameters to the server, thus preserving data privacy. The server then aggregates all the model updates. There are multiple aggregation algorithms at the level of the server, which is the subject of this report. The performance of a model trained in federated setting is lower than that trained in central setting, as the experimental results show, however the data privacy is prioritized and preserved.

1 Introduction

Machine Learning has an undeniable significance. Its application in various domains has revolutionized the world, rendering intelligent systems more accessible and cost-effective. Machine learning is also extensively used in domains dealing with sensitive data. For example, hospitals train machine learning models using clients' medical data to predict potential illnesses for individuals. In such applications, where the training data is sensitive, there is a greater emphasis on the privacy of individuals' and companies' data. Recently, regulations and laws on data sharing have increased after data privacy concerns have multiplied, especially with prominent big tech companies like Meta [2].

As a response to the increase in data privacy concerns, researchers have proposed different solutions. One prominent solution is federated learning. Unlike traditional machine learning which trains the model centrally using the whole training dataset in a central server, federated learning trains the model locally on each data contributor's local device. This technique enhances data privacy by not sharing individual data with a central server, rather each contributor trains the model locally and only sends the updated model's parameters to the server for aggregation to create a global model. Notably, Google's mobile keyboard prediction system represents one of the first pioneering successes of federated learning [3].

[*]Duke University, Email: {aicha.slaitane}@duke.edu

Recent years have seen an increase in federated learning research given its importance and high potential. Researchers have investigated optimal ways to aggregate the locally trained models' parameters in the central server, personalizing the global model to each client, dealing with challenges stemming from the nature of the non-IID (non-independent and identically distributed) distribution of the clients' data, and more. Some of the most known types of aggregation algorithms include average aggregation, clipped average aggregation, secure aggregation, differential privacy average aggregation, momentum aggregation, weighted aggregation, Bayesian aggregation, and adversarial aggregation [6].

2 Aggregation Approaches

After a single round of training, each client sends the model parameters updates of the model training using the local data at that round to the central server. The server then aggregates the model updates of all the clients to form a global model. These steps are repeated until model convergence. There are many approaches to averaging the parameter updates [6]. We are discussing some of them in this report.

2.1 Average Aggregation

This is the first approach that was ever introduced in the field of federated learning in 2017 [5]. Its main concept is straightforward: average the clients updates. It's simple and easy to implement, however, it's sensitive to outliers and malicious clients and may not perform well in cases of non-IID data. The general formula of this approach is the following:

$$w = \frac{1}{N}\sum_{i=1}^{N} w_i$$

where:

w : global model parameters

N : number of clients (data providers) in a certain round of training

w_i : model parameters of client (data provider) i

2.2 Clipped Average Aggregation

This method is similar to average aggregation, where the average of received messages is calculated, but with an additional step of clipping the model updates to a predefined range before averaging. This approach helps reduce the impact of outliers and malicious clients that may transmit large and malicious updates [8].

$$w = \frac{1}{N}\sum_{i=1}^{N} \text{clip}(w_i, c)$$

where:

w : global model parameters
N : number of clients (data providers) in a certain round of training
w_i : model parameters of client (data provider) i
c : Clipping threshold

2.3 Weighted Aggregation

In this method, the server weights each client's contribution to the final model update depending on client performance or other parameters such as the client's device type, the quality of the network connection, or the similarity of the data to the global data distribution. This can help give more weight to consumers that are more reliable or representative, improving the overall accuracy of the model [7].

$$w = \frac{\sum_{i=1}^{N} a_i \cdot w_i}{\sum_{i=1}^{N} a_i}$$

where:

w : global model parameters
N : number of clients (data providers) in a certain round of training
w_i : model parameters of client (data provider) i
a_i : Weights associated with each parameter

2.4 Differential Privacy Average Aggregation

A layer of differential privacy to the aggregation process to ensure confidentiality of client data. Each client adds random noise to its model update before sending it to the server, and the server compiles the final model by aggregating the updates with the random noise. The amount of noise in each update is carefully tuned to compromise between privacy and model correctness [9].

$$w = \frac{1}{N} \sum_{i=1}^{N} (w_i + b \cdot n_i)$$

where:

w : global model parameters
N : number of clients (data providers) in a certain round of training
w_i : model parameters of client (data provider) i
n_i : random noise vector, drawn from a Laplace distribution with a scale parameter b
b : privacy budget parameter

3 Aggregation Algorithms

In this section, we'll be introduce two examples of federating learning aggregation algorithms: Federated Averaging (FedAvg) and Federated Proximal (FedProx). We'll use these two algorithms to run experiments in the next section.

3.1 FedAvg

This algorithm is a fundamental technique used in federated learning, a decentralized approach to training machine learning models across multiple devices or servers that hold local data samples. The key steps of the FedAvg algorithm are as follows [5]:

- Initialization: The central server initializes a global model.

- Client Training: Each client downloads the global model, trains it locally using its own data, and computes a model update.

- Model Update Aggregation: The central server collects the model updates from all clients and aggregates them following the aggregation approach described in section 2.1 to compute a new global model.

- Model Distribution: The updated global model is distributed back to all clients, and the process repeats.

While FedAvg has several advantages, it also comes with some drawbacks and limitations related to communication overhead, network bandwidth usage, stragglers, and model heterogeneity. FedProx, discussed next, addresses the problem of stragglers.

3.2 FedProx

[4] Federated Proximal (FedProx) is a variation of the federated learning algorithm designed to address non-IID (non-identically distributed) data and improve convergence in heterogeneous federated environments. It extends the principles of proximal optimization to federated learning settings. FedProx introduces a regularization term called the proximal term to the loss function, which penalizes deviations of local model updates from a global solution. This regularization encourages clients to stay close to the global model during updates, mitigating the impact of non-IID data distributions and improving convergence speed. The key steps of FedProx are similar to those of FedAvg. The only difference is at the level of aggregation at the central server.

$$\min_{w} F_k(w) + \mu^2 \|w - w_t\|_2^2$$

where:

- mu is the local model update from client proximal coefficient.
- w_t is the model at round t
- w is global model

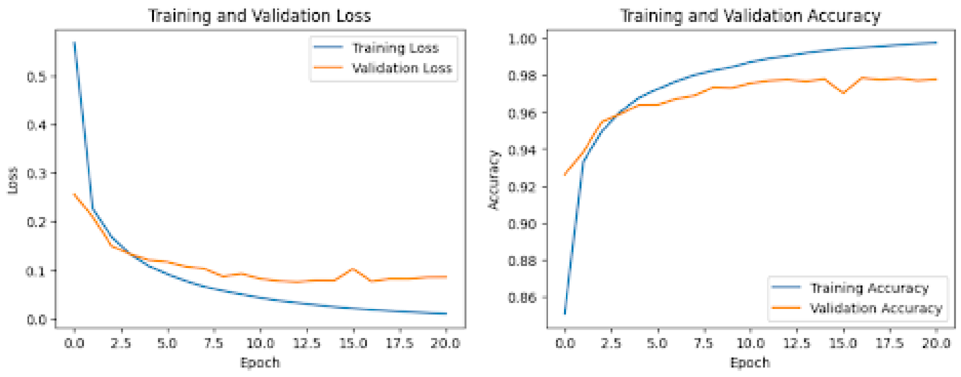

Figure 1: Accuracy of the trained model in a centralized setting

4 Experimental Results

In this section, we show the results of running some experiments to compare the performance of a classification model trained in both centralized and federated settings. The classification task is the standard classification problem in the machine learning literature of handwritten digits using MNIST dataset.

Centralized Setting

The performance of training the model (convolutional neural network) on the MNIST dataset to classify handwritten digits reached more than 95%. Please note that the purpose of this report is not to get a high model performance. Figure 1 shows the results of the centralized training.

Federated Setting

The results of training the convolutional neural network in a federated setting are shown in figure 2 and figure 3 [1]. There are three clients in each training round. They have non-independent and identically distributed datasets. In the context of MNIST dataset, non-idd means that the clients have different digits in their datasets with varying probabilities. This relates to how the datasets in the real world can be.

Figure 2 shows the training with FedAvg as the aggregation algorithm, while Figure 3 shows the results of the training using FedProx.

5 Conclusion

The results of the experiments show that the performance of the CNN classification model trained on the MNIST dataset is better in the centralized setting than in the federated setting. There could be different reasons that can explain this difference including communication overhead. In federated learning, model updates are sent back and forth between the server and clients during each iteration, introducing communication overhead. This communication latency can affect the convergence of the model and may lead to slower training or suboptimal performance compared to the centralized setting.

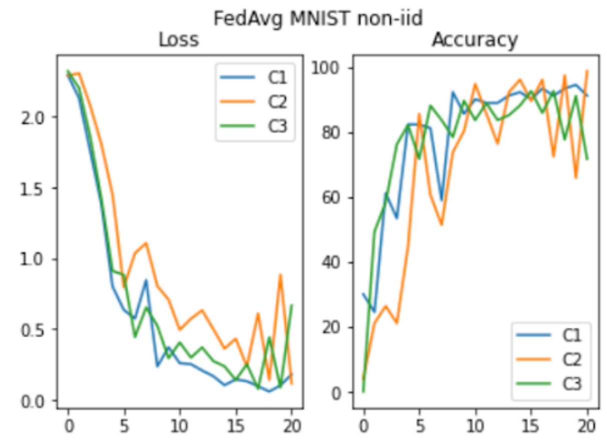

Figure 2: Training with FedAvg

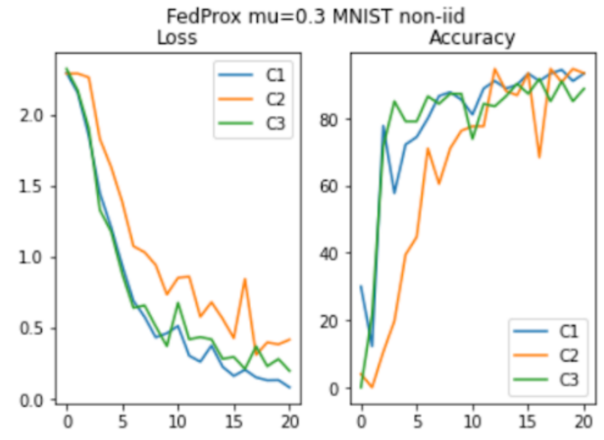

Figure 3: Training with FedProx

Overall, the observed difference in performance between centralized and federated settings highlights the trade-offs and challenges associated with distributed learning paradigms like federated learning.

6 Future Work

Federated Learning is one of the solutions to preserve the data privacy and can be a substitute to machine learning. It was applied in many prominent applications like Google's GBoard prediction feature [3]. However, there is still a need to incentivize clients to participate in federated training. Some clients may have concerns about model fairness and the cost of training, so designing mechanisms to encourage them to participate in the training is important to train a more robust and accurate global machine learning model. Their participation would thus contribute to social welfare. One approach to this problem is through game theory, particularly coalitional game theory. In coalitional game theory, the focus is on what groups of clients, rather than what individual clients, would achieve if they work towards the same goal. Given a set of clients, a coalitional game defines how well each group (or coalition) of clients can do for itself. Thus, delving deeper in this field and drawing connections from it to federated learning is an interesting future work direction.

References

[1] AI4health winter school - Practical Session on Handling heterogeneity in the analysis of biomedical information. Example of FedAvg and FedProx for two datasets: MNIST iid and MNIST non-iid.

[2] Diane Bartz. Meta, us government spar in court over toughened privacy order. *Reuters*, October 2023. https://www.reuters.com/legal/meta-us-government-spar-court-over-toughened-privacy-order-2023-10-17/.

[3] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *ArXiv*, 2018. Accessed February 20, 2024.

[4] Tian Li, Anit K. Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018. Accessed April 30, 2024.

[5] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the Artificial Intelligence and Statistics*, volume 54, pages 1273–1282, April 2017.

[6] Md Moshawrab, Mohamed Adda, Abdenour Bouzouane, Hany Ibrahim, and Abdelhakim Raad. Reviewing federated learning aggregation algorithms: Strategies, contributions, limitations and future perspectives. *Electronics*, 12:2287, 2023.

[7] Jorge Reyes, Luigi Di Jorio, Cédric Low-Kam, and Marta Kersten-Oertel. Precision-weighted federated learning. *arXiv preprint arXiv:2107.09627*, 2021.

[8] Ting Wang, Zhicheng Zheng, and Feng Lin. Federated learning framework based on trimmed mean aggregation rules. *SSRN Electronic Journal*, 2022.

[9] Kaixiang Wei, Jie Li, Meng Ding, Chunxiao Ma, Hong-Han Yang, Farhad Farokhi, Shi Jin, Q.S.Q. Tony, and H. Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.