PERSONALIZING FEDERATED LEARNING USING META-LEARNING

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Matthew Whelan

Spring, 2023

Aidong Zhang, Department of Computer Science

***Abstract*—Federated learning (FL) has emerged as a promising approach for collaborative training of machine learning models in distributed environments, where data is decentralized across multiple devices. This paper presents a study on the implementation and comparison of two commonly used algorithms in FL: Federated Averaging (FedAvg) and Personalized Federated Averaging (Per-FedAvg). FedAvg serves as the baseline algorithm, while Per-FedAvg is an extension that aims to find the optimal solution for the federated learning problem. The performances of both algorithms are evaluated and compared in terms of convergence rate, accuracy, and communication efficiency. The experiments are conducted on a diverse set of distributed datasets, with consideration given to the heterogeneity of devices and data characteristics. The results demonstrate that Per-FedAvg exhibits improved performance over FedAvg in various scenarios, showcasing its potential as an enhanced algorithm for personalized federated learning. This study contributes to the advancement of federated learning methods and provides insights into the benefits of personalized algorithms in distributed machine learning settings.**

I. Introduction

Machine learning has been widely recognized as a powerful tool for solving complex problems across multiple domains. This is due to its ability to recognize patterns and learn from data which surpasses human performance at comparably low costs. However, there is a significant limitation to traditional machine learning methods: the requirement for large amounts of centralized data to train models. This can be a major challenge in distributed environments where data is spread out across multiple devices and cannot be easily aggregated in one location.

To address this challenge, federated learning (FL) has emerged as a promising approach that allows multiple parties to collaboratively train a model without sharing their data directly with each other. Federated learning distributes the model's training across the devices that contain the data, instead of centralizing the data to train the model. This reduces communication cost, as training can be done locally and only model updates need to be communicated to the server. The central server then aggregates the model updates to modify the global model and communicates back the changes, which greatly reduces amount of data needing to be transferred between the client and server. Thus, federated learning can significantly reduce privacy risks and data transfer costs while enabling scalable and flexible machine learning. As low energy devices and privacy concerns become more prevalent, federated learning is gaining interest as an alternative to traditional centralized machine learning.

However, federated learning poses its own unique challenges. The main challenge is heterogeneity of devices and data, as the decentralized setting can include a wide variety of different device hardware and data characteristics. For example, natural language processing (NLP) models are generally trained on mobile devices, namely for next-word keyboard prediction. Training data in this context, i.e. every word a user types on their keyboard, is privacy sensitive and can vary drastically from person to person. Further, since the data is based on the usage of the user, the data is likely to be non-independent and identically distributed (non-IID) across devices. Additionally, some users can have much heavier usage patterns than others, leading to unbalanced amounts of data from user to user.

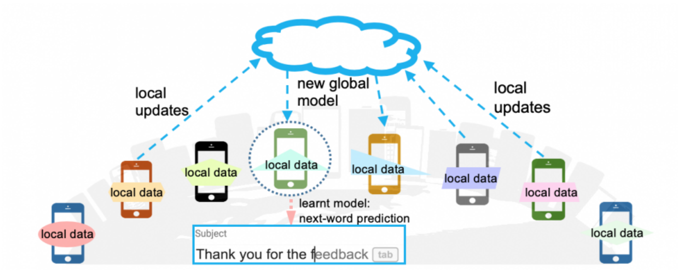

Fig. 1. Federated Learning Setting for Next-Word Prediction

The research conducted in this paper is a part of a National Science Foundation (NSF) grant under the guidance of Professor Aidong Zhang. This research hopes to progress towards the overall goal of implementing a multi-party learning framework to support data sharing in a distributed environment. To help further this goal, this paper implements a commonly used algorithm in FL, FedAvg, Federated Averaging, as a baseline. This paper also implements Per-FedAvg, Personalized Federated Averaging, an extension of FedAvg, and compares the performance of both algorithms.

II. Background

Federated Learning

As mentioned previously, FL is a distributed machine learning paradigm that allows multiple clients to collaboratively train a global model without sharing their local data with a central server. To formally expression this scenario, consider a federated learning problem in the cross-silo setting, where the objective is to minimize the following function:

$$\min_{w\in\mathbb{R}^d} f(w) = \frac{1}{N}\sum_{i=1}^{N} f_i(w) \tag{1}$$

Here, $f_i(w) = \ell(x_i, y_i; w_i)$ where this calculates loss of predicting the true label y_i for the input x_i made with parameters w for client i out of the N total clients. As such, $f(x)$ calculates the average of the local model parameters in order to capture the collective knowledge of all devices.

FedAvg

In terms of its algorithmic implementation, **Federated Averaging** [1] dictates a global model is to be trained by averaging the local models from each device or node, regardless of their individual characteristics or preferences. This is generally done in a distributed environment, where data is spread out across devices and the data is kept private.

The general steps behind FedAvg include the following:

1) Initialization and Client Selection: central server initializes a global model and a set of clients is selected to participate in each round of federated learning.

2) Local Model Training: each selected client downloads the current global model from the central server and trains the model locally using its own data. This is performed using local optimization algorithms such as Stochastic Gradient Descent (SGD).
3) Local Model Update: after local training, each client generates as a model update that represents the changes made to the local model during training, which can be in the form of model parameters, gradients or other information.
4) Model Aggregation and Global Model Update (FedAvg Step): Client model updates are sent to the central server for aggregation using the FedAvg algorithm, where the weighted average of the updates replaces the previous global model
5) Iteration: Steps 2-4 are repeated for multiple rounds, so that the global model can be iteratively improved through collaboration until the global model converges or reaches the maximum specified number of rounds

Per-FedAvg

Personalized Federated Averaging [2] is an extension of the FedAvg algorithm, meaning it allows each client to have its own personalized model by utilizing a global, meta model. This meta model leverages the collective knowledge and insights gained from the base models to improve overall performance. The premise is current or new clients can adapt this meta model to their own local dataset by performing one or a few steps of gradient descent, known as few-shot learning. In this way, prior knowledge from a larger set of related clients can be leveraged to learn how to quickly adapt to new clients with only a few training examples.

This process is done through using the Model-Agnostic Meta-Learning (MAML) problem formulation discussed in [3] using a two step procedure. The first step is the inner loop (personalization step), where the model is exposed to a few training examples from a specific task and performs gradient updates to adapt its parameters. The second step is the outer loop (global update step), where the updated parameters are fine-tuned for a specific task based on the performance from the previous step to optimize fast adaption across tasks.

Since this algorithm relies on few-shot learning principles, to simplify the problem expression, we assume each user takes the initial model and updates it using one step of gradient descent with respect to its own loss function. As such, the problem formulated in (1) changes to:

$$\min_{w \in \mathbb{R}^d} F(w) = \frac{1}{N} \sum_{i=1}^{N} f_i(w - \alpha \nabla f_i(w)) \tag{2}$$

where $\alpha \geq 0$ is the learning rate or step size. This problem formulation allows for the convergence upon a meta-model which is trained in a way that one step of local gradient leads to a good model for every individual user. Per-FedAvg follows a similar scheme as FedAvg for solving equation (2); as detailed in [2], the first step is to compute the gradient of local functions - the gradient ∇F_i:

$$\nabla F_i(w) = (I - \alpha \nabla^2 f_i(w)) \nabla f_i(w - \alpha \nabla f_i(w)) \tag{3}$$

where $\nabla^2 f_i(w)$ represents the Hessian matrix, which is comprised of second-order partial derivatives of $f_i(w)$. Calculating the Hessian matrix can be very computationally expensive, so Per-FedAvg uses two different approximation techniques.

The first follows the concepts in First-Order MAML (FO-MAML) in [3], which replaces the gradient estimation with its first-order approximation therefore ignoring the Hessian term ($I - \alpha \nabla^2 f_i(w)$ term). This is referred to as Per-FedAvg (FO).

The second technique follows the idea of HF-MAML, proposed in [4], where the Hessian-vector product in the MAML global update step (second step) is replaced by difference of gradients using the following approximation:

$$\nabla^2 \phi(w) u \approx \frac{\nabla \phi(u + \delta v) - \nabla \phi(u - \delta v)}{\delta}$$

This is referred to as Per-FedAvg (HF).

A. *Related Work*

There are many other significant advancements being made in the FL space that address its main challenges.

As referenced earlier, meta-learning, also known as "learning to learn," is a subfield of machine learning that focuses on algorithms that can learn from and generalize across multiple interrelated learning tasks. The goal of meta-learning is to enable models to acquire knowledge or prior experience from a set of related tasks, allowing them to quickly adapt and perform well on new, unseen tasks. One popular meta-learning algorithm, ARUBA, incorporates adaptive regularization during the training process, which is shown to improve the performance of the FedAvg algorithm [5].

Another related subarea is multi-task learning, where multiple related tasks are split between clients to be jointly learned and optimized to achieve personalization. MOCHA [6] is a federated learning algorithm that presents a solution to overcome data heterogeneity and communication efficiency. Instead of treating each client as an independent learning task, MOCHA leverages multi-task learning techniques to collaboratively learn a set of separate but related models simultaneously across clients. This algorithm is the first of its kind to address certain statistical and systems challenges within the distributed multi-task learning setting, such as high communication cost, stragglers, and fault tolerance. The methods introduced in this paper were studied in detail, formed a foundational set of knowledge for this research and remain a topic of interest in future extensions of this work.

III. Approach

In terms of actual implementation, this project utilized PyTorch, a widely used machine learning framework, within a Python runtime environment. To facilitate the simulation of federated learning runs, the research leveraged the distributed communication package **torch.distributed**. This package enabled the distribution of the training process across multiple

GPUs within a single machine, harnessing parallel processing capabilities for improved performance and a more realistic testbed than traditional single-processor configurations.

To execute runs of the algorithms, the suite, created by Xidong Wu of University of Pittsburgh, uses the argparse package to convert command line arguments to code. This allows for customization of key parameters, such as the learning rate (α), momentum rate (β) and specifying if using the Hessian approximation. The suite incorporates many functions, including `partition_dataset()`, to partition and modify the heterogeneity of the dataset. The suite provides all of the necessary neural network models (using PyTorch) for compatibility with the supported datasets. The datasets supported include MNIST, FashionMNIST and CIFAR-10, all with three separate settings for heterogeneity [7]. In the first level (0), all clients share the same data so each client is homogenous. In the second level (1), each client has class data and divides it equally across clients, simulating a middling heterogeneity scenario. In the third level (2), each client has partial class data that is not divided equally among clients, which simulates high heterogeneity.

As stated previously, Per-FedAvg uses few-shot learning principles to frame the problem such that each user takes the initial model and updates it using one step of gradient descent with respect to its own loss function. Normally, the algorithm is tested with respect to the test data for performance evaluation either directly before or after a communication round, i.e. when the model aggregation between clients occurs. But, for Per-FedAvg, a helper function, `train_one_step()`, runs one step of stochastic gradient descent with respect to the test data before evaluating the model on the test dataset.

The suite along with all other code for this project is available at https://github.com/matthewhelan/Federated-Learning. Xidong Wu's expertise and efforts in developing the suite have been instrumental in enabling the customization and execution of these algorithms [7].

IV. Results

The three algorithms focused on are FedAvg, Per-FedAvg (FO) and Per-FedAvg (HF). For a fair comparison, the output of the FedAvg method is updated with one step of stochastic gradient descent with respect to the test data, using `train_one_step()`. This is because both of the Per-FedAvg implementations utilize this function to personalize the model before evaluation.

In determining run scenarios for these algorithms, a few key variables are available to modify. First, to change the total amount of clients simulated, the `--worker-size`, representing N, is varied between 2 and 10 to simulate different client scenarios. Second, the `--lr`, α, is varied from 0.01 and 0.001 to change the convergence speed and robustness. For consistency, the neural network design and non-specified hyperparameters are unchanged. The test accuracy results along with the 95% confidence intervals are shown in Table 1 for the MNIST dataset with low heterogeneity.

TABLE I
Comparison of Test Accuracy in percentage (%)

Hyperparameters	**FedAvg**	**Per-FedAvg (FO)**	**Per-FedAvg (HF)**
α = 0.01, N = 2	87.02 $\pm$ 0.02	89.81 $\pm$ 0.04	94.74 $\pm$ 0.03
α = 0.01, N = 10	84.85 $\pm$ 0.01	90.62 $\pm$ 0.07	96.23 $\pm$ 0.06
α = 0.001, N = 2	89.54 $\pm$ 0.03	93.92 $\pm$ 0.03	95.63 $\pm$ 0.06
α = 0.001, N = 10	84.14 $\pm$ 0.05	93.72 $\pm$ 0.03	97.17 $\pm$ 0.03

There are a few main key takeaways that can be gathered from Table 1. The first is the personalized variants of FedAvg, Per-FedAvgs, perform better than FedAvg for all test scenarios. For increasing α from 0.001 to 0.01, there is a marginal decrease in performance, but not substantial. But, in the case of Per-FedAvg (FO), the performance seems to decrease the most when increasing α, a trend also noted in [2]. For increasing N from 2 to 10, there is a slight, but not consistent decrease in performance. Overall, it's clear that Per-FedAvg (FO) generally performs better than FedAvg and Per-FedAvg (HF) generally performs the best out of all methods.

In other test runs changing the heterogeneity, similar trends can be found. Per-FedAvg (HF) consistently outranked the other methods, with Per-FedAvg (FO) continuing to struggle greatly with higher α values, when compared to Per-FedAvg (HF). One reason why Per-FedAvg (HF) seems to outperform Per-FedAvg (FO) in these runs is because of its tendency to adapt better to user data. This method seems to better capture the characteristics of each client's distribution, resulting in improved personalized solutions.

The CIFAR-10 dataset runs give interesting insights. Since this dataset is much larger and more complex than the MNIST dataset, the accuracies gained are much lower than the values shown in Table 1. But, as a result of this, Per-FedAvg achieves a much more significant gain on the other methods. These results are also consistent with the FashionMNIST dataset runs. Figure 2 shows how Per-FedAvg (HF) seems to converge fastest to the optimal solution and seems to achieve the best test accuracy out of the three methods in a scenario with $\alpha = 0.01$ and $N = 2$ under a low heterogeneity setting.

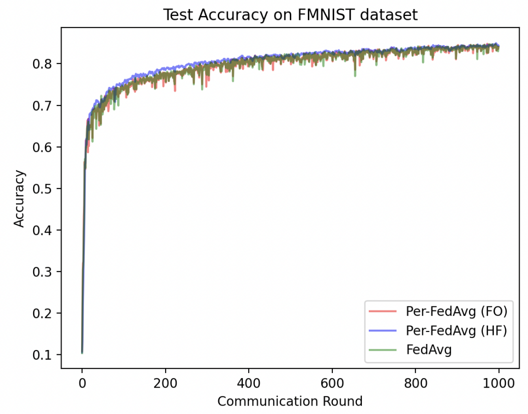

Fig. 2. Comparison of Test Accuracy on FMNIST dataset

Thus, it seems like the Hessian approximation, or lack thereof, makes a difference in the Per-FedAvg implementations. Per-FedAvg (HF) utilizes an approximation of the Hessian-vector product, replacing it with a difference of gradients. Evidently, this approximation is generally much more accurate than the complete omission of an approximation in the case of Per-FedAvg (FO). This means a more efficient computation of the Hessian information, which can lead to faster convergence and better optimization performance overall.

V. Conclusion

This paper focuses on the Federated Learning (FL) problem and its related challenges in a distributed setting. Specifically, this paper looked at the implementations of two popular FL algorithms: FedAvg and Per-FedAvg. FedAvg serves as the baseline algorithm, while Per-FedAvg is an extension designed to address the specific challenges of the FL problem. The two relevant versions of Per-FedAvg include Per-FedAvg (FO), which ignores the second-order (Hessian) term, and Per-FedAvg (HF), which uses an approximation technique to estimate the Hessian term.

The study evaluates and compares the performance of both algorithms for various distributed datasets, taking into account the heterogeneity of devices and data characteristics. The experiments reveal that Per-FedAvg outperforms FedAvg in multiple scenarios, demonstrating its potential as an improved algorithm for personalized federated learning. Further, the results also show that Per-FedAvg (HF) is the best performing method out of the two Per-FedAvg methods due to its better approximation, tendency to adapt to user data and ability to handle larger learning rates.

The findings contribute to the advancement of federated learning methods by highlighting the benefits of personalized algorithms in distributed machine learning settings. The study provides valuable insights into the performance of different FL approaches and their implications for practical applications. By addressing the limitations of traditional FL algorithms, Per-FedAvg (HF) offers a promising solution for achieving more efficient and accurate collaborative learning in decentralized environments.

Acknowledgement

I would like to express my sincere gratitude to Xidong Wu and Jianhui Sun from the University of Pittsburgh and University of Virginia, respectively, for the invaluable support and guidance throughout my undergraduate research. I would also like to extend my thanks to Professor Aidong Zhang for her continuous support, mentorship, and encouragement, and the National Science Foundation (NSF) for funding this project.

References

[1] McMahan, B., Moore, E., Ramage, D., Hampson, S., and Aguera y Arcas, B. (2017). Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS).

[2] Fallah, A., Mokhtari, A., and Ozdaglar, A. (2020). Personalized federated learning: A meta-learning approach. arXiv preprint arXiv:2002.07948.

[3] Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 34th International Conference on Machine Learning (pp. 1126–1135). Sydney, Australia.

[4] Fallah, A., Mokhtari, A., and Ozdaglar, A. (2020). On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In International Conference on Artificial Intelligence and Statistics (pp. 1082–1092).

[5] Khodak, M., Balcan, M.-F. F., and Talwalkar, A. S. (2019). Adaptive gradient-based meta-learning methods. In Advances in Neural Information Processing Systems (pp. 5915–5926).

[6] Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. (2017). Federated multi-task learning. In Advances in Neural Information Processing Systems (pp. 4424–4434).

[7] Wu, X., Huang, F., Hu, Z., and Huang, H. (2023). Faster Adaptive Federated Learning. Retrieved from https://arxiv.org/pdf/2212.00974.pdf