Universe Scientific Publishing

DOI:10.18686/ahe.v7i29.10721

Guiding Federated Learning with Low Rank Metrics

Toby Yang **Ahmed Elkoushy**

University of Toronto Toronto Ontario Canada M5S 1A1

Abstract: Federated learning (FL) is a distributed machine learning method that protects the privacy of local data, and thus is used in scenarios like medical applications. However, FL algorithms are known to demonstrate deteriorated performance when the local data of each client is not independently and identically distributed (Non-IID). In this work, we explore the possibility of guiding the aggregation process in FL using low-rank based layer-wise metrics, and propose an aggregation framework that allows the use of any layer-wise metric to guide the aggregation process. Specifically, a layer-wise metric is used as the weight for the weighted averaging process in the aggregation, and the metric is optionally smoothed by exponential averaging.

In this work, we propose two low-rank layer-wise metrics. However, experiments do not show significant improvement of the proposed methods over the baseline method (FedAvg), thus this paper mainly serves as a record of our exploration and the experimental results.

Keywords: Federated learning (FL); Low rank; Non-IID data; Explainable metrics

1. Introduction

Federated Learning (FL) trains distributed machine learning models, preserving data privacy in cross-silo settings like healthcare. Non-IID data often leads to client drift, harming knowledge preservation [4, 6]. We focus on guiding aggregation with explainable low rank factorized metrics in FL, enabling layer-wise metric usage.

2. Related Works

2.1 Federated Learning Problem Formulation

Traditionally, most machine learning optimization problems (involving one client) are of the following form:

$$\min_{w\in\mathbb{R}^d} \quad f(w) \quad \text{where} \quad f(w) \equiv \frac{1}{n}\sum_{i=1}^{n} f_i(w) \tag{1}$$

Where f(w) is defined to be the empirical loss and is simply the average of the loss on each data point l(xi, yi) or l(ˆyi, yi) where yˆi refers to the output of the model.

In the federated case, we would like to construct a problem where we could leverage already well defined optimization methods such as SGD. [5] showed that one can form local subproblems of similar structure on each client and define the following local loss:

$$F_k(w) \equiv \frac{1}{n_k}\sum_{i\in P_k} f_i(w) \tag{2}$$

Where k refers to the kth client from K total clients, P_k is the data partition for client k where nk = $|P_k|$ and $f_i(w)$ is the same as before but only specific to a data point within P_k. The goal is to use local optimization algorithms and define the global loss in terms of all of the local subproblems in order to come up with some kind of aggregation technique to merge the results of the local optimization. Traditionally, this is done through the Federated Averaging algorithm. If $n \equiv \sum n_k$, the global empirical loss can be expressed as the following according to [5]:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) = \sum_{k=1}^{K} \frac{n_k}{n} \cdot \frac{1}{n_k}\sum_{i\in P_k} f_i(w) \tag{3}$$

Finally, the new federated optimization problem becomes

$$\min_{w \in \mathbb{R}^d} \quad f(w) \quad \text{where} \quad f(w) \equiv \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \tag{4}$$

2.2 Federated Averaging

As previously discussed, FedAvg is the traditional aggregation algorithm in which each client is weighted according to its corresponding fraction $\frac{n_k}{n}$ as seen in (4) [8]. In practice, this is done by having each client run a local optimizer such (typically SGD or a variant) and every E epochs, a communication round is done and the aggregation step is performed. This translates to two different weight update steps:

$$w_{t+1}^k \leftarrow w_t^k + \eta \cdot \nabla F_k(w_t) \quad \text{or} \quad w_{t+1}^k \leftarrow ClientUpdate(w_t^k) \quad \text{Update Step for Client K} \tag{5}$$

$$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k \quad \text{Update Step for the Server after E epochs} \tag{6}$$

3. Low Rank Metrics

3.1 Low Rank Factorization

Within a Convolutional Neural Network (CNN), the weights of a convolution layer take the form of 4-D tensors of dimension $k \times k \times n_i \times n_o$ where ni is the number of input channels and no is the number of output channels. The weight W_l for convolution layer l acts as a learned encoder for feature extraction and so [3] was able to unfold the tensor W on the input channel or output channel to obtain a 2D matrix $W_l \in^{whno \times ni}$ or $W_l \in^{whni \times no}$. This matrix is then separated from noise using Variational Bayesian Matrix Factorization to obtain a low-rank singular value decomposition $\hat{W}_l = U\Lambda V^T$ alongside some noise matrix:

W_l[Weight Matrix] $\rightarrow \hat{W}_l$[Low-Rank Structure] + E_l[Noise Perturbation]

3.2 Stable Rank

According to [3], the stable rank is defined as the norm energy of the singular values of a given matrix, however they propose a modified definition of stable rank for the low rank m×n matrix $\hat{W}_l$ given by the following:

$$s(\hat{W}_l) = \frac{1}{n\sigma_1^2(\hat{W}_l)} \sum_{k=1}^{n'} \sigma_k^2(\hat{W}_l) \tag{7}$$

Where $\sigma_1 \geq \sigma 2 \geq ... \geq \sigma_{n'}$ are the low rank singular values in descending order and $n \leq m$. This measure is normalized by $n\sigma^2(\hat{W}_l)$ and $s(\hat{W}l) \in [0, 1]$. A higher measure is theoretically indicative of a better encoder.

3.3 Temporal Knowledge Gain

We define temporal knowledge gain of client i S for layer l as the difference of stable rank between round t and round t + 1 as evaluated on the low rank weight structure:

$$KG(\hat{W}_l^i) = s(\hat{W}_l^{i,t+1}) - s(\hat{W}_l^{i,t}) \tag{8}$$

If this difference is positive, it indicates an increase in stable rank and a relative improvement in themapping done on layer l for client i. The opposite holds true if it is negative and the magnitude of the knowledge gain represents the amount of knowledge lost or gained. Since $s(\hat{W}l) \in [0,1]$, $KG \in$

$[-1,1]$ with 1 being the desirable extreme representing a complete gain of knowledge.

3.4 Uniqueness Score

The goal of this metric is to compare the learning of one client in comparison to how consistent it is with other clients in order to combat client drift. In order to measure this, we use cosine similarity. Consider a set of clients S and two clients j and k S. The low rank weight tensors of layer l are concatenated into vectors in the following manner:

$$\hat{w}_l^j = \begin{bmatrix} | \\ \hat{W}_l^j \\ | \end{bmatrix} \quad \text{and} \quad \hat{w}_l^k = \begin{bmatrix} | \\ \hat{W}_l^k \\ | \end{bmatrix}$$

To calculate the similarity between the weights of these two clients we then simply compute

$$Sim(\hat{w}_l^j, \hat{w}_l^k) = \frac{\hat{w}_l^j \cdot \hat{w}_l^k}{\|\hat{w}_l^j\| \|\hat{w}_l^k\|} \tag{9}$$

We then define the uniqueness score in the following manner:

$$U(\hat{W}_l^i) = \sum_{k \in S} (1 - Sim(\hat{w}_l^{i,t}, \hat{w}_l^{k,t})) \tag{10}$$

Various intuitions about metric's role in aggregation exist due to clients' diverse training on unique samples. There are multiple intuitions at play here in terms of how this metric can be used to guide aggregation. Given that each client is being trained on different samples, each client may develop new patterns that other clients cannot develop. Discovery of these useful patterns leads to rapid divergence by responsible clients. Prioritizing the most unique client for new information may be beneficial. Sometimes, minimizing drift by favoring agreeing clients is useful. The most useful approach may change, but for simplicity, we use the uniqueness score (defined in (10)) for aggregation. Cosine similarity applies to any vectors; low rank factorization isn't strictly needed but aids noise reduction for better results.

4. Proposed Framework

The low rank metrics mentioned are among many possible ones to explore factorized weights and assess learning quality. Our focus is on temporal knowledge gain and similarity score in our experiments. This framework operates per layer, modifying federated averaging to introduce smoothness in weighted averaging and implement aggregation using a specified low rank metric. For general purposes, we will refer to a low rank metric as some measure on the weights M(Wˆ i) for layer l and client i. That means, in our example,M(Wˆ il) can be equal to KG(Wˆ il) or U(Wˆ il). Note that for a certain round t, M(Wˆ il) might include past values for weights from previous rounds such as in the case for knowledge gain and can alsoinclude weights of other clients such as is required to calculate uniqueness score. M($\hat{W}^1$l) is simply a freehand that shows that the measurement is being done to rank the client itself.

These metrics work layer by layer, converting 4D tensors in convolutional blocks for low rank measurements. Linear layers are weighted similarly to convolutional layers, each requiring individual weighted aggregation, unlike FedAvg and other strategies.

Algorithm 1 Framework for Metric Guided Aggregation

Inputs: exponential decay factor β, exponential update factor ζ
Initialize: $w^0 = [W_1^0, W_2^0, ..., W_L^0]$ for L layers, $\alpha_l^{i,0} = 1$ for all layers l and clients i
    w^n: global model parameters at round n.
    $\alpha_l^{i,n}$: exponential averaging memory variable for layer l, client model i, at round n.
for round $t = 0, 1, ..., T-1$ **do**
    for client $i \in S$ in parallel **do**
        $w_i^{t+1} \leftarrow ClientUpdate(w_i^t)$ ▷ Local update step
    for client $i \in S$ **do**
        Initialize $\alpha_p^i = 1$ ▷ Keep track of visited conv layer.
    α_p^i: temporary variable used for keeping track the score of the previous conv layer
        for layer $l = 1, 2, ..., L$ in $w_i^{t+1} = [W_1^{i,t+1}, W_2^{i,t+1}, ..., W_L^{i,t+1}]$ **do**
            if $dim(W_l^{i,t+1}) = 4$ **then** ▷ If convolutional layer
                $\alpha_p^i \leftarrow \beta \cdot \alpha_l^{i,t} + \zeta \cdot M(\hat{W}_l^i)$ ▷ assign the smoothed score to the temp variable
            $\alpha_l^{i,t+1} \leftarrow \alpha_p^i$ ▷ move the temp value into the exponential averaging memory
            $\bar{\alpha_l}^i \leftarrow \frac{N_i}{\sum_{k \in S} N_k} \cdot \alpha_l^{i,t+1}$ ▷ calculate a normalized score based on number of data samples per client
    for layer $l = 1, 2, ..., L$ in $w_i^{t+1} = [W_1^{i,t+1}, W_2^{i,t+1}, ..., W_L^{i,t+1}]$ **do**
        // Normalize $\bar{\alpha_l}^i$ for each client i and aggregate
        $W_l^{t+1} \leftarrow \sum_{i \in S} \bar{\bar{\alpha}}_l^i \cdot W_l^i$ where $\bar{\bar{\alpha}}_l^i = \bar{\alpha}_l^i / \sum_{i \in S} \bar{\alpha}_l^i$
    $w^{t+1} \leftarrow [W_1^{t+1}, W_2^{t+1}, ..., W_L^{t+1}]$ ▷ Concatenate updated layers

To prevent discrepancies, the calculated score is normalized for weighted averaging. Multiple metrics can be applied by chaining multiplication (see normalization in Algorithm 1). Scores are often chained with the number of client samples, similar to FedAvg. Note, in the implementation, any zero/negative α values are limited to 1e-3 for stability, ensuring no negative or zero α in aggregation but allowing very small values for specific clients. Furthermore, as seen in line 10 of Algorithm 1, the factor β is introduced in order to increase thesmoothness. This hyperparameter can be tuned in order to obtain the desired stability.

The two algorithms being used in our experimentation will be termed as FedKG where $M(\hat{W}^i_l) = KG(\hat{W}^i l)$ where KG is as defined in (8) and FedUnique where $M(\hat{W}^i_l) = U(\hat{W}^i l)$ where the Uniqueness Score is as defined in (10).

5. Experiments

We experimented with various algorithms and hyper parameters with our proposed framework andexplored how much each algorithm improve the final accuracy in an non-IID dataset setting and howthey perform in combination.

5.1 Experimental Setup

This work primarily focuses on image classification tasks, using the widely accepted Resnet-18 model and Adam optimizer. Experiments utilize CIFAR10, CIFAR100, and TinyImageNet200 datasets, partitioned non-IID using Dirichlet distribution. Each subset corresponds to one client's private data. Implementation leverages the Flower framework and Dirichlet partitioning with FedML. FedAvg serves as the baseline for comparison. Hyper-parameter search is constrained to 250 training epochs, with the best combination used in the final experiment. Each experiment averages accuracy from three trials. See Figure 1.

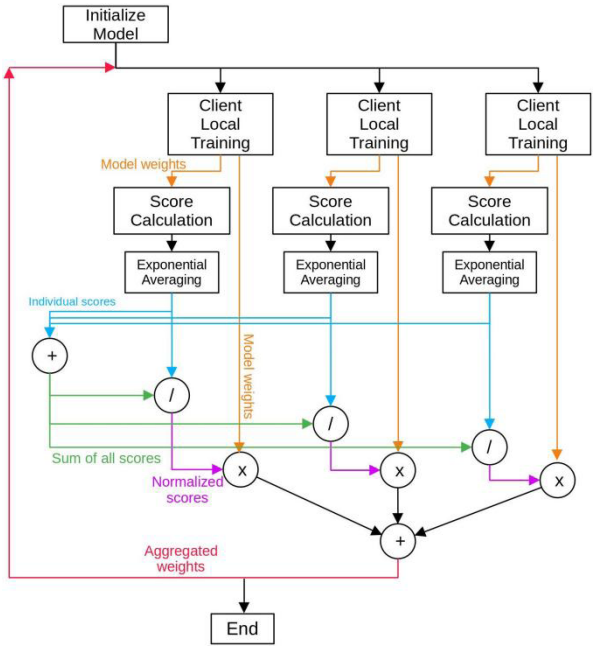

Figure 1: Flowchart of the framework.

5.2 Generating Non-IID Data

Non-IID covers various data distribution types, including horizontal and vertical splitting in sample and label spaces. Our experiment focuses on horizontally split data that is Non-IID in both spaces, resulting in similar content and format among local datasets with varying sample counts per class. We created Non-IID datasets using a customized Dirichlet partitioning algorithm based on FedML. We simulated two Non-IID levels with $\alpha = 0.3$ and $\alpha = 0.0001$, as illustrated in Figure 2.

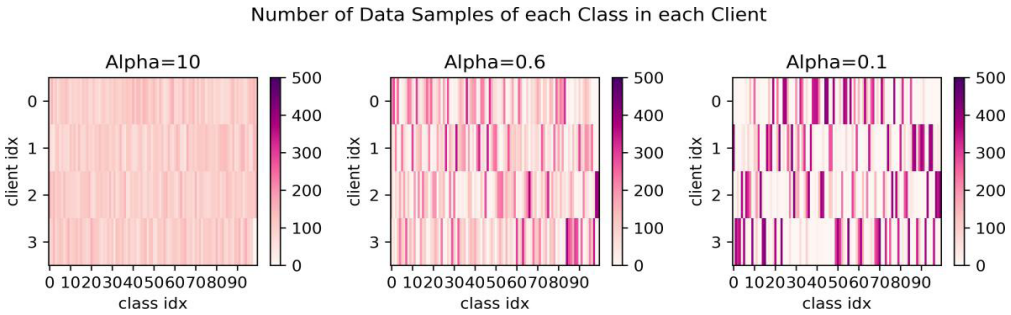

Figure 2: Non-IID data distribution visualization. Darker color indicates more data per class label per client. This representation reflects the data proportion, not the actual scale of CIFAR100 dataset samples.

5.3 Accuracy Results

We conducted experiments on six datasets: CIFAR10, CIFAR100, TinyImageNet200, with Dirichlet alpha=0.3 and 0.0001. Each algorithm underwent three trials for each setting, and the highest test accuracy per trial was averaged. Test accuracy is the mean of an epoch's accuracy across all four clients. Results are depicted in Figure 3.

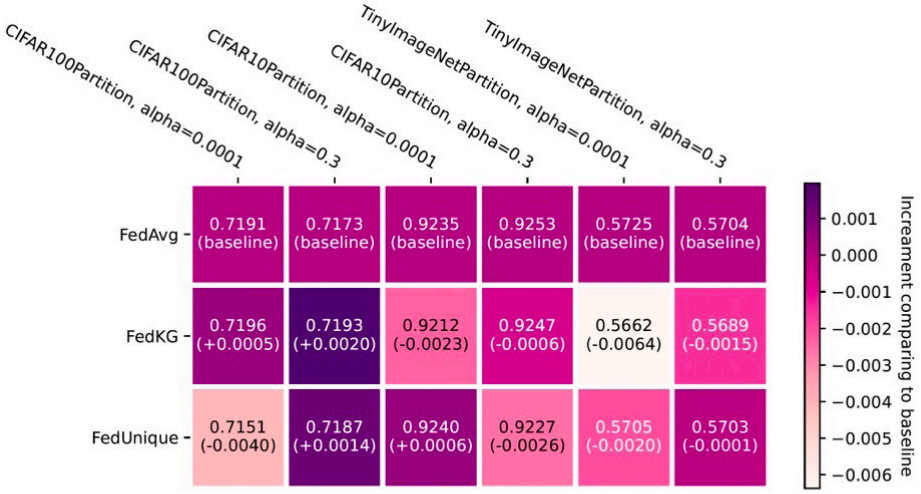

Figure 3: Experiment result. The background colour shows the improvement in accuracy compared to the baseline, deeper colour is preferable. The accuracy shown is the average of 3 trials with random initialization.

6. Conclusion

We introduced a federated learning framework enabling layer-wise low-rank metric usage to guide aggregation and explored its effectiveness. However, experiment results didn't reveal a strong pattern. Performance of FedKG and FedUnique compared to FedAvg depends on the dataset and lacks statistical significance (<1%). This paper mainly serves as an exploration record. Additional methods explored but not included can be found in the appendices.

References:

[1]Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D. Lane. Flower: A friendly federated learning research framework. CoRR, abs/2007.14390, 2020.

[2]Chaoyang He, Songze Li, Jinhyun So, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, RameshRaskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. Fedml: A research library and benchmark for federated machine learning. CoRR, abs/2007.13518, 2020.

[3]Mahdi S Hosseini, Mathieu Tuli, and Konstantinos N Plataniotis. Exploiting explainable metrics for augmented sgd. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10296–10306, 2022.

[4]Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. CoRR, abs/1909.06335, 2019.

[5]Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated op- timization: Distributed machine learning for on-device intelligence. CoRR, abs/1610.02527, 2016.

[6]Gihun Lee, Yongjin Shin, Minchan Jeong, and Se-Young Yun. Preservation of the global knowledge by not-true self knowledge distillation in federated learning. CoRR, abs/2106.03097,2021.

[7]Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos:An experimental study. CoRR, abs/2102.02079, 2021.

[8]H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. 2016.

[9]Bjarne Pfitzner, Nico Steckhan, and Bert Arnrich. Federated learning in a medical context: A systematic literature review, 07 2020.