A Brief Description of Our Submissions for Track 2 of IDASH 2024

The LARC Team*
Chongqing Institute of Green and Intelligent Technology,
Chinese Academy of Sciences, Chongqing 400714, China

September 10, 2024

1 Problem Statement

In a basic federated learning setting, there are six clients holding different amount of data from TCGA-BRCA dataset, and the parameters of the global model are represented by W. Denoting the ith client as C_i and its local data as D_i, for round t, C_i trains its model on D_i and uploads local pseudo gradient G_i^t to the server S. S then computes linear combinations of pseudo gradients and obtains aggregated gradient as $G_{global}^t = \sum_{i=1}^{6} \alpha_i^t G_i^t$, where α_i^t is a weight assigned to C_i. G_{global}^t is sent back to clients and the model is finally updated by $W^{t+1} = W^t + Adam(G_{global}^t)$, where $Adam()$ indicates the application of Adam optimizer. The goal is to measure and utilize an optimized weight vector $\boldsymbol{\alpha^t}$ for aggregation that facilitates faster and better convergence of the federated learning task.

2 System Overview

We propose a dynamic weighted aggregation algorithm that assigns each client a variable weight based on quality of the client's current update in each round. The pseudo code is listed in Algorithm 1. For round t, each client first sends the pseudo gradient $\boldsymbol{G^t}$ produced by local training to S, on which an initial weighted aggregation result $\hat{G}_{global}^t$ is calculated using weights from the last round. The naive aggregation result is then broadcasted to each client together with the client's corresponding α_i^{t-1}, and C_i calculates two loss, L_{i1}^t and L_{i2}^t, on its local data applying local update and non-local update on the model parameters, respectively. The loss function $L(W, D)$ indicates loss computed on dataset D applying parameters W and is defined in the same way as that of the

*Jingwei Chen, Xiangyu Hu, Qingyu Mo, Haoyong Wang, Wenyuan Wu, Haonan Yuan (in alphabetic order). Corresponding Email: wuwenyuan@cigit.ac.cn

local Cox model training. $\boldsymbol{\Delta L^t}$ is calculated and sent to S, and S computes a proper weight vector $\boldsymbol{\alpha^t}$ based on it, by which the aggregated pseudo gradient is finally computed and broadcasted to every client for update.

The rationale behind our design is that the performance divergence between parameters produced by local training and global weighted aggregation could be indicative to adjusting weights. On one hand, a bigger $(L_{i1}^t - L_{i2}^t)$ implies worse performance of locally trained parameters on D_i compared to aggregated ones excluding C_i's pseudo gradient contribution, and α_i^{t-1} should be set to be smaller to improve the performance of final aggregated parameters on D_i in that case. On the other hand, a smaller $(L_{i1}^t - L_{i2}^t)$ indicates that local pseudo gradient outperforms non-local one more on D_i, and α_i^{t-1} is supposed to be tuned to be bigger. Therefore, the term ΔL_i^t is inversely proportional to the weight α_i^{t-1}, and we use a softmax function to project the loss into $[0, 1]$ so that it can be used as weights.

Such design is however insufficient to produce a proper weight assignment mechanism for two reasons. First, the relative scale between α_i^t is disproportionate to the mere output of softmax. Although the naive algorithm mentioned above can determine the relative order of α_i^t correctly, the relative scale is usually empirical and data-dependent. Second, the softmax function would eliminate α_i^t with small $(L_{i1}^t - L_{i2}^t)$, making its value near to 0, while the fast convergence requires every clients to make contributions to some extent.

To overcome the two challenges that mere softmax function confronts, we introduce two hyper-parameters, q and b, and their ways of influencing weights are illustrated in line 15 and 19 of Algorithm 1. q is responsible for modeling the relative scale, while b can provide α_i^t a base value, $\frac{b}{1+b}$, to ensure that every client contributes to the aggregation.

We show the effectiveness of our approach by experimental results that will be discussed later. the security of the system is promised since no information that contains or directly elicits original data is communicated between clients and server. The extra communication overhead is composed of $\hat{G}^t$, $\boldsymbol{\alpha^{t-1}}$ and $\boldsymbol{\Delta L^t}$ for each round. the latter two are single numbers for each client, and the major extra overhead lies in transmitting vector $\hat{G}^t$ which would increase communication budget by around 50% on the basis of baseline.

Data: total rounds T, q, b, D_i, $\alpha_i^{-1} = 1$, $i \in [1, 6]$
Result: W^T
Randomly initialize model parameters W^0 and share to all the clients;
for *t in range(T)* **do**
  for *i in range(6)* **do**
    C_i locally trains on D_i and obtains G_i^t;
    C_i sends G_i^t to S;
  end
  S computes $\hat{G}_{global}^t = \sum_{i=1}^{6} \alpha_i^{t-1} G_i^t$;
  S broadcasts $\hat{G}_{global}^t$ to $C_1 - C_6$ and sends them their respective $\alpha_1^{t-1} - \alpha_6^{t-1}$;
  for *i in range(6)* **do**
    C_i calculates $L_{i1}^t = L(W^t + Adam(\alpha_i^{t-1} G_i^t), D_i)$,
    $L_{i2}^t = L(W^t + Adam(\hat{G}_{global}^t - \alpha_i^{t-1} G_i^t), D_i)$;
    C_i sends $\Delta L_i^t = L_{i1} - L_{i2}$ to S;
  end
  S computes softmax scores p^t given six clients' $q\Delta L_i^t$ as input: $p_i^t = softmax(-q\Delta L_i^t)$, and the maximum of scores is denoted as $max(\boldsymbol{p^t})$;
  for *i in range(6)* **do**
    S computes $\alpha_i^t = \frac{\frac{p_i^t}{max(\boldsymbol{p^t})} + b}{1+b}$;
  end
  S calculates $G_{global}^t = \sum_{i=1}^{6} \alpha_i^t G_i^t$ and broadcasts to $C_1 - C_6$;
  for *i in range(6)* **do**
    C_i updates its parameters as $W^{t+1} = W^t + Adam(G_{global}^t)$;
  end
end

Algorithm 1: Workflow of our weighted aggregation algorithm.

3 Running Guideline

The source code of our algorithm can be found here. Our algorithm is written in "solution" function in "todo.py", and the hyper-parameters are defined in the first two lines of it. We have also edited few lines in "fed_opt.py" to access necessary variables, and they are marked with annotations in the file.

Our testing platform is Windows 11 and Python 3.9, and we recommend committee to implement the project in this setting. No other packages than those already covered in "requirement.txt" of the given baseline code are needed to be downloaded. The command to install those packages for us is however different from the original instruction and is "Directory of your python interpreter.exe -m pip install -r requirements.txt" in our setting. We suggest that committee uses this command if packages cannot be installed successfully. Once

the environment is built, directly running "federated.py" should work and produce an output of about 0.7428.

4 Performance

The experiment result is discussed in this section. We also implement [1] and [2] for comparison. The dataset we use is the official TCGA-BRCA dataset given by iDASH committee which contains 900 data points, and we keep the data distribution of the 6 clients fixed. Besides, all the configurations concerning local training and aggregation process are maintained, and the aggregation round is set to be 5 by default. We use all the data for training and test the model on training dataset. The result is shown in Table 1 when the hyper-parameter is set as $q = 19$ and $b = 0.5$. We also randomly split the dataset on each client into training set and test set by the ratio of $5 : 1$ for 10 times, and the average c-index of the federated learning result is shown in Table 2. According to the result, the performance of our algorithm using either training set or independent set as testing set outperforms baseline and other SOTA methods.

Method	c-index
Baseline	69.02%
[1]	68.83%
[2]	72.06%
Ours	74.28%

Table 1: C-index result using training set for test.

Method	c-index
Baseline	65.08%
[2]	66.79%
Ours	69.15%

Table 2: C-index result using individual testing set.

References

[1] H. Wu and P. Wang. Fast-convergent federated learning with adaptive weighting. *IEEE Transactions on Cognitive Communications and Networking*, 7(4):1078–1088, 2021.

[2] Y. Xia, D. Yang, W. Li, A. Myronenko, D. Xu, H. Obinata, H. Mori, P. An, S. Harmon, E. Turkbey, et al. Auto-fedavg: learnable federated averaging for multi-institutional medical image segmentation. *arXiv preprint arXiv:2104.10195*, 2021.