Solo Project: Federated Learning

March 30, 2024

1 Objectives

- Learn the basics and processes of Federated Learning.
- Simulate and implement the interactions between clients and the cloud server in Federated Learning.

2 Project Statement

The experimental task is to simulate a simple BloodMNIST/CIFAR10 (pick any one of the two) classification in horizontal Federated Learning mode.

The whole project is divided into the following three steps to help you improve the functionality and get closer to how the federated learning framework works in the real world.

2.1 Stage 1

In federated learning, we suppose that one server and N clients train a classification task collaboratively, which updates model parameters by **a) direct data access** or **b) ".pth" file reading and writing**.

In stage 1, all clients participate in each round of updates, as shown in Figure 1. In one round of global iteration, the global model is first sent to all clients, and then each client completes several local rounds of model training on their own private dataset. After that, all clients upload their respective local models to the server, and the server aggregates the model parameters of all clients through the average operation to obtain the global model, which is tested on the test dataset.

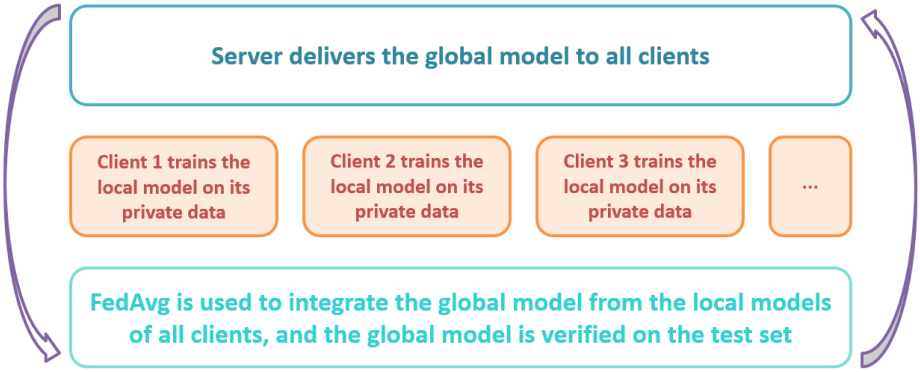

Figure 1: Flowchart of Stage 1

2.2 Stage 2

Based on stage 1, let M out of N clients be selected to participate in the update in each global round, which is called the partial participation mode.

1. Please randomly select M clients to participate in the update in each global round.
2. The program should support different M.

2.3 Stage 3

In stage 1, the server and clients interact through the ".pth" file reading and writing. Here, the server and clients interact using Socket communication. When the server is running, clients can connect to the server. The communication between the client and server should be implemented using Internet-domain socket.

Specifically, the tasks of the server and each client are as follows:

The server:

- Listen on the port, accept local models from each client, and gradually complete model aggregation.
- Send the global model to each client after aggregation.
- Using TCP protocol is suggested.

The client:

- Train locally on its own private dataset and send local model parameters to the server.
- Receive global model parameters from the server.

3 Implementation Details

- The clients' local private training data will be given together later, with the file named as Client1, Client2, ···, Client20 (there are 20 clients, $N = 20$). In Stage 1, all 20 clients will participate in the model parameter update. In Stage 2, M of the 20 clients will participate in the model parameter update. In addition, we will provide a Test dataset for the aggregated global model performance evaluation.
- Print out as much information as possible to show the intermediate results of your programs.
- Python packages you might need for Stage 3:

  import socket

  You can go online and search for "python socket file transfers" for reference.

4 Material to be submitted

- Submit a PDF experiment report to describe design ideas, experimental procedures, running results, reference materials, etc. Using *Latex* is suggested.
- Compress the PDF report together with the source code into the *FL+StudentID.tar* file. Use meaningful names for the file so that the contents of the file are obvious. Enclose a *README* file that lists the files you have submitted along with an explanation and show that how your program works.
- Send your *FL+StudentID.tar* file to the Canvas.
- Due date: **May 12, 2024**, submit on-line before **PM 24:00**.