COMP3221: Distributed System

2024 Semester 1

Assignment2 Report

Student Name: Zijian Huang
Student ID: 540276745
Student Name: Zheng Shi
Student ID: 500173424

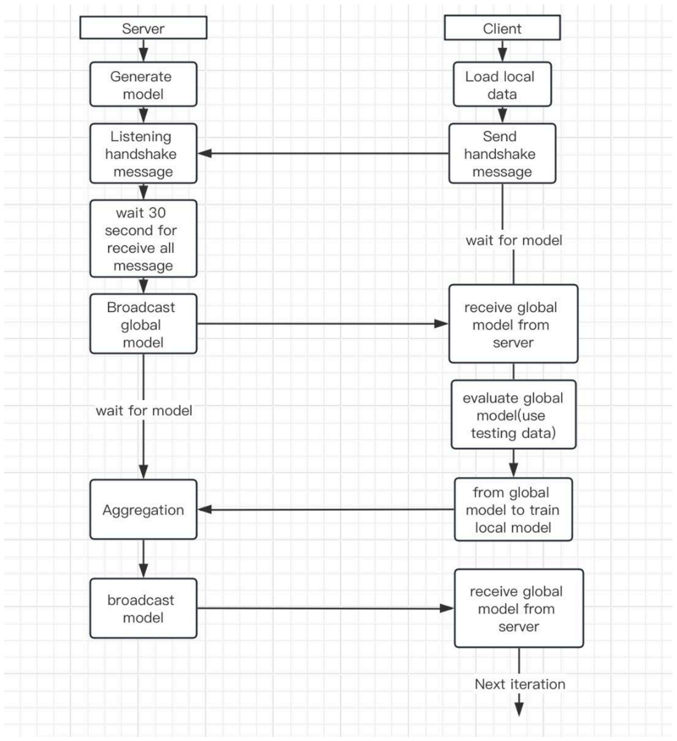
Github link: https://github.com/kennyzzZZ619/assignment2_comp3221.git

1.Federated Learning Algorithm

1.1. Understanding of Algorithm

In this assignment, we use the thought of Federated Learning. It is a distributed machine learning system. It allows multiple clients connected to one server to train the model. Every model is trained at the client using their local data. The Federal Average (FedAvg) algorithm is used in this project.

1.2. Algorithm workflow


Server
Client
Generate model
Load local data
Listening handshake message
Send handshake message
wait 30 second for receive all message
wait for model
Broadcast global model
receive global model from server
wait for model
evaluate global model(use testing data)
Aggregation
from global model to train local model
broadcast model
receive global model from server
Next iteration

This graph shows how we design the working process. The first is the initialization of the global model. This is done by instantiating the "LinearRegressionModel" class. Second step is that every client receives a global model from the server and trains it using the local data. This uses the GD and mini-batch GD to evaluate and train the data. Next step is to update the local model to the server, and the server chooses sub samples to aggregate the model. This step is involved in the weight average of weights, which are determined by the amount of data per client. After aggregation, the server sends a global model to all clients and executes the next iteration.

2.Techniques and methodology for implementation

2.1. Server and client communication

Server using an object called 'FLServer' to initialize the server. Inside this object we define all the functions that can be used in the server. Before establishing communication. We initialize the global model using the "LinearRegressionModel" class. Before entering the while loop to wait for the receiving model. We first start the server socket and build the connection between all clients. Each client uses a thread to handle the handshake message. Ihis step is to register the client on the server.

2.2. Sub-sampling and aggregation

We use the last argument as a sub-sampling identifier. This will pass to the handle_model method and divide model which will be input into the aggregation. We use an array to store the client details which are recorded when registering the clients, and using this array to randomly select which client will connect to the server. Then the selected client sends the model to the server. The client will send the model to the server after training. I mimicked the idea of aggregation in the tutorial and wrote a function in imitation.

2.3. Local model training

The global model is received and trained on a local dataset. Training is performed using Stochastic Gradient Descent (SGD) algorithm with the goal of minimizing the Mean Square Error (MSE) loss. During the local training process, the client will perform several training iterations based on the global model sent by the server and generate an updated local model. And we have two different training methods controlled by last argument input. It manages the size of the data in each epoch. All of the above is encapsulated in a function 'global_iteration' that updates the global_model at each iteration.

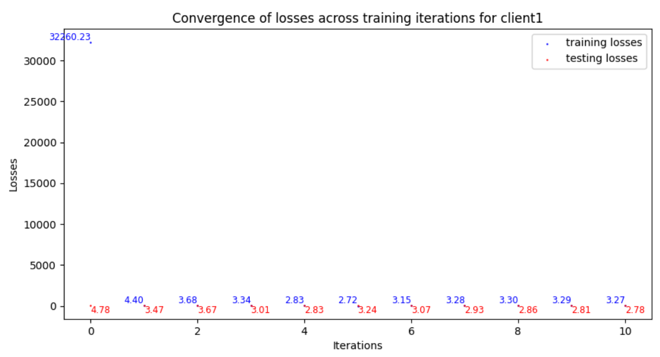
3.Machine Learning Model & Dataset Description

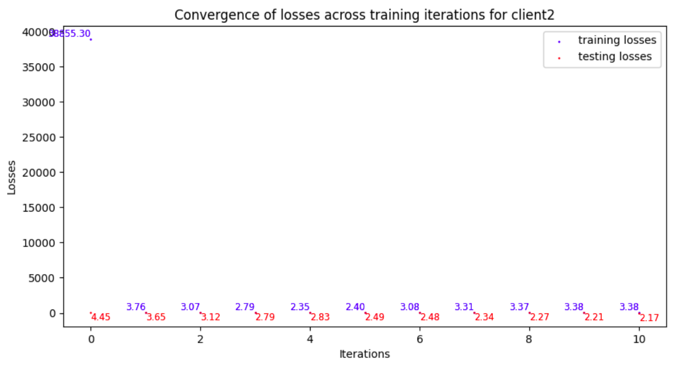
The model we select is the linear regression model with stochastic gradient descent. The model starts from server and client with randomly initialized parameters. In the global iterations, the models are updated to minimize the loss function.

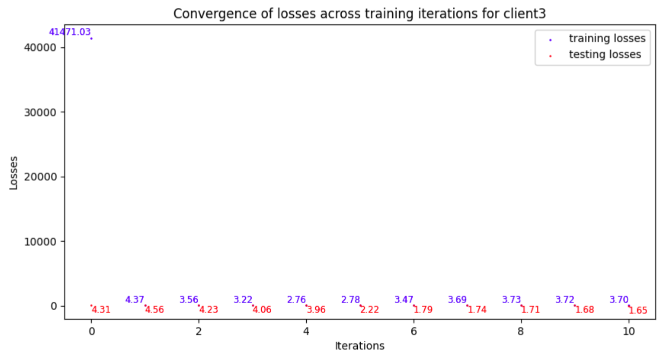
The dataset is the California Housing Dataset with 8 numerical features and 1 numerical target variable.

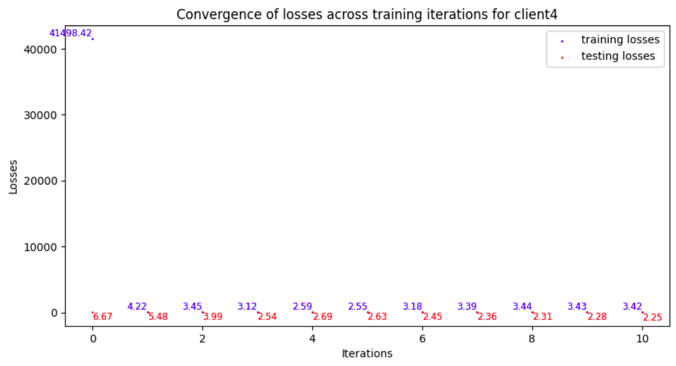
5.Experiment Results & Analysis

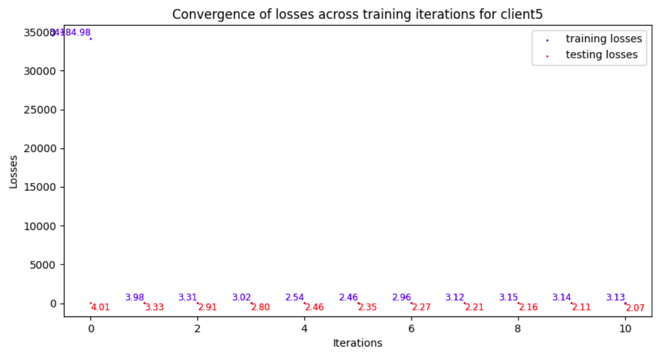
Global Linear Regression Model with Small-Batch GD: As illustrated by the figures below, the training loss, while initially high, converges with the testing losses as the number of global iterations increases.



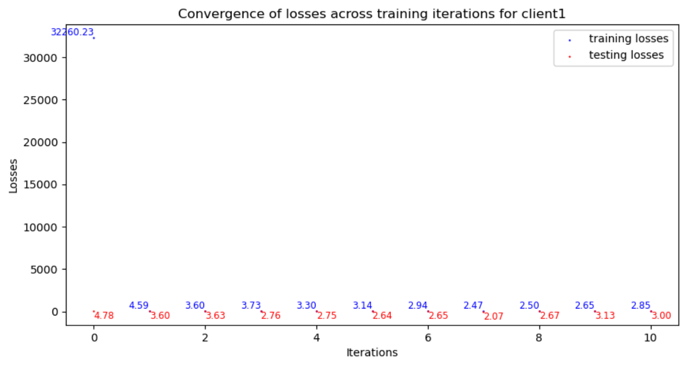


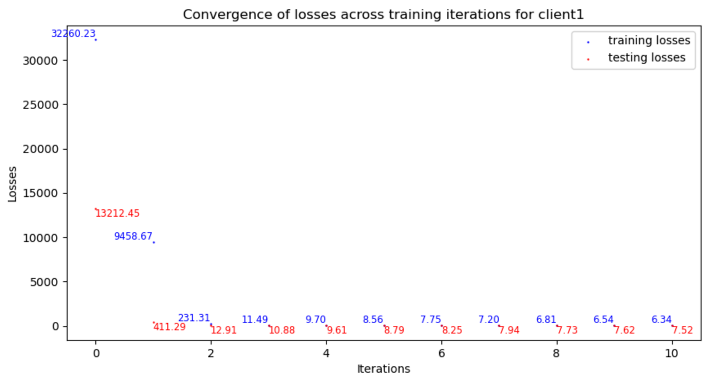




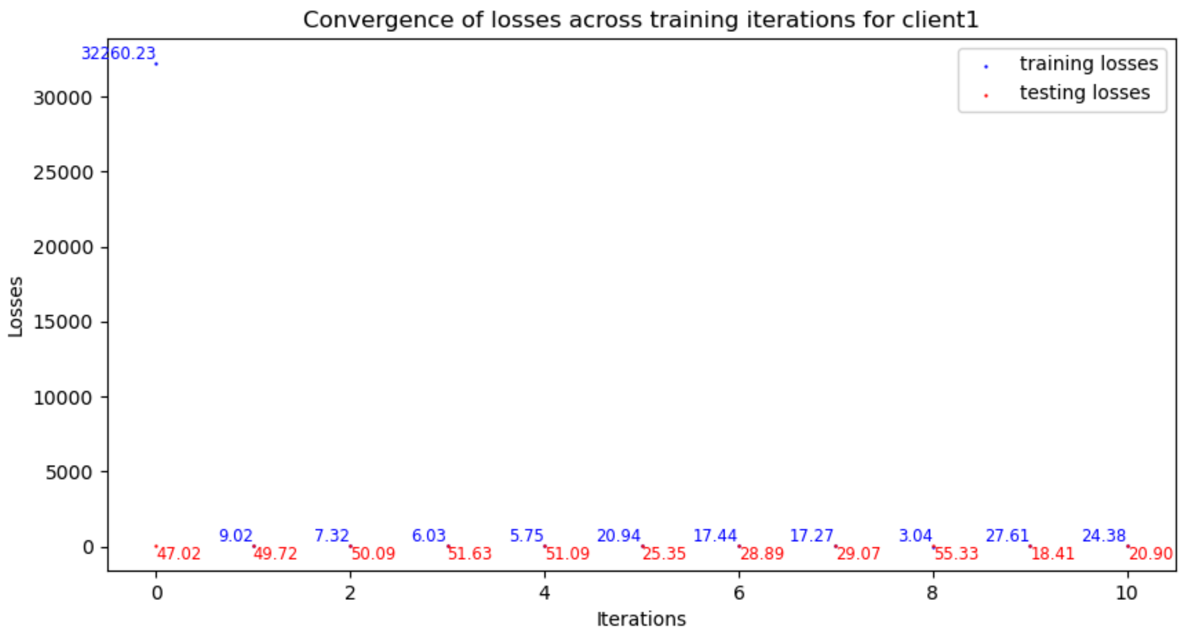


When subsampling is activated and models from only 3 clients are aggregated, the subsequent losses are different from that without subsampling. This is shown by the lower left figure.

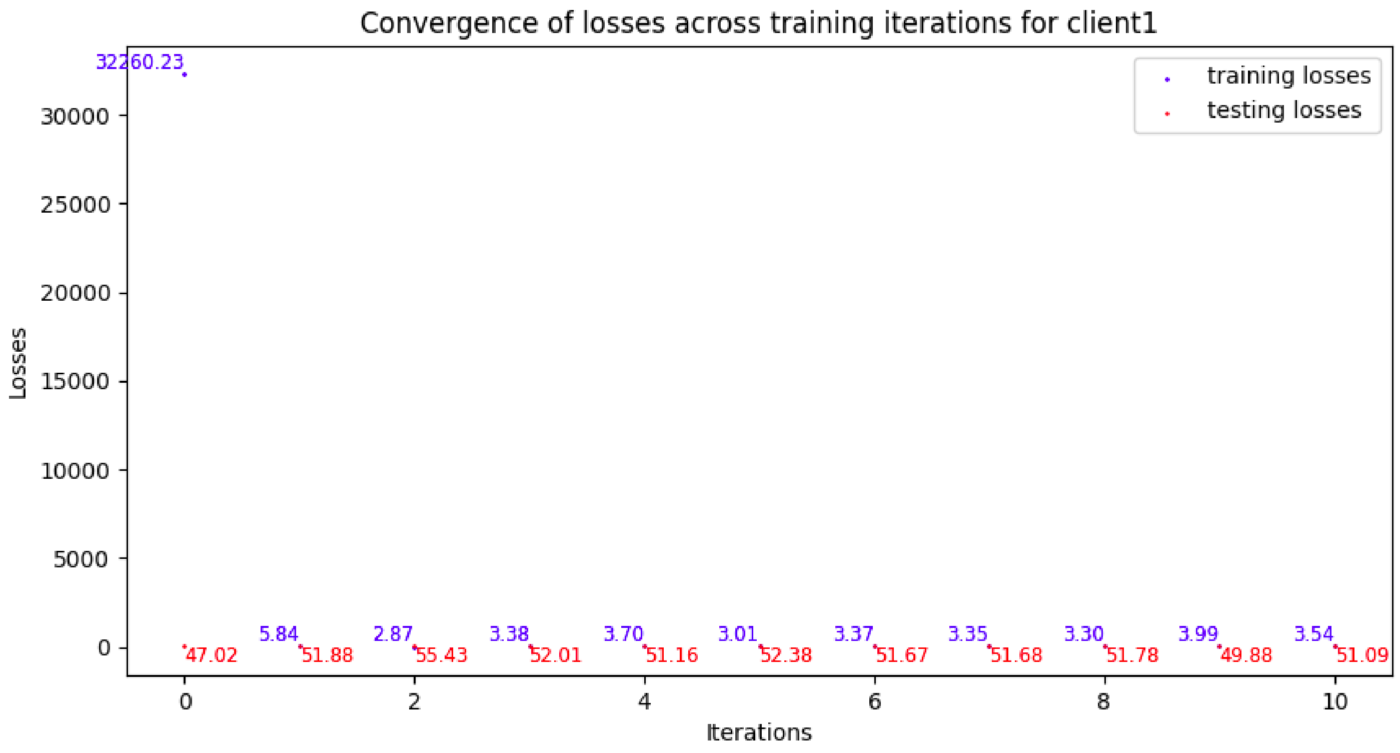




The upper right figure shows the loss converging slower with K=3 and GD as the optimization method. Regarding client subsampling, full client participation (M=K) theoretically improves the generalization of the model. But at such a time when the data distribution is extremely uneven, it can lead to a slight bias. But we found a problem, when the learning rate is 0.005, the server parameter is (6000 2 demo) and the client parameter is (6001 1 demo), we will find that the ten iterations TRAIN LOSSES are getting higher and higher, which also indicates the problem. (show as below) This again demonstrates that high learning rates may lead to rapid convergence, but may also exceed the optimal point, leading to unstable or divergent model performance.



After changing the learning rate from 0.001 to 0.005, the testing losses increase.


Convergence of losses across training iterations for client1
training losses
testing losses
Losses
Iterations