Efficient Client Selection in Federated Learning

Background

Federated Learning (FL) is a distributed machine learning paradigm where multiple clients collaboratively train a global model without sharing their private data. However, clients in real-world FL systems may have heterogeneous computational capabilities and data characteristics. Random client selection, while simple, can lead to suboptimal performance in such scenarios.

Objective

In this exercise, you will work with a simulated FL framework that supports model training over a dataset split across multiple clients. Your goal is to analyze and understand the impact of client heterogeneity—particularly in terms of data distribution and compute performance—on FL training dynamics, and then design a better client selection strategy.

Task Breakdown

1. Per-Client Data Analysis

You are provided with a dataset split across multiple clients. Start by performing data analysis on each client's local dataset to answer:

- What is the label distribution (class imbalance) per client?
- How many samples does each client have?
- Are there significant differences in data size or class diversity across clients?

Make a Jupyter notebook with plots and analysis of the above three points (and anything else you want to report).

2. Profiling Client Training Time

Perform an ablation study by measuring the **memory used(using top/htop)** and **training time per minibatch** for different batch sizes for 100 mini-batches of local training.

- Do you see any variability in the minimatch times and memory usage? Why?
- What should be the optimal minibatch size to use from what you have observed?

Make a Jupyter notebook showing memory and minibatch times plots with respect to different batch sizes

3. Baseline FL Experiment with Random Client Selection

Using the provided FL simulation code:

- Plug the optimal batch size and the corresponding minibatch timings into the FL simulation framework config for use in the next steps.
- Run 20 rounds of FL with **random client selection,** selecting **3 clients out of 12 every round**.

Summarize what you observe—do some clients contribute more to the model's performance? Is training time well-utilized?

4. Design a Smarter Client Selection Strategy

Devise a new client selection algorithm that aims to **maximize accuracy gain per unit time**. You can consider factors such as:

- Client training speed (minibatch time)
- Data diversity or label distribution
- Sample size
- Historical contribution to model accuracy
- Whatever else you think will help.

Implement and integrate this strategy into the simulation framework.

- Plot accuracy vs. round and accuracy vs. time for random and your novel client selection algorithm.
- Plot the variation of client train times per round for the first 20 rounds.
- Run random client selection and your novel client selection to convergence. Plot a **moving average with a window of 5 rounds** for the two client selection strate. Set target accuracy as min(max_accuracy_random, max_accuracy_your_algo).
- Fairness amongst clients using Jain’s fairness index.
- BONUS: Plot rounds to target accuracy on y-axis and average_roundtime on x-axis.

5. Reflection and Discussion

Prepare a short summary discussing:

- What aspects of client heterogeneity were most critical?
- Why does your strategy outperform (or didn't outperform) random selection?
- Trade-offs between fairness, speed, and accuracy in FL?

Deliverables

1. **Source Code Repository**
    - Make a directory called **data_analysis**, put the notebook and plots for Section1 in this directory.
    - Make a directory called **profile_on_rbpi**, put all scripts, data files and plots for Section2 in this directory.
    - Make a directory called **simulation**, put your updated simulation code with all configs etc in this directory.

2. **Technical Report (Max 5 pages)**
    - 0.5-1 page on Section 1.
    - 0.5-1 page on Section 2.
    - 1-2 pages on Section 4.
    - Analysis and conclusion.
    - Add necessary plots. If you want to add an algorithm, please do so in the appendix.

3. **Demo Presentation (5 min)**
    - Present your algorithm with clear reasoning for each module.
    - Presenting results.
    - Discussion of challenges, solutions, and future improvements.