Federated Learning & Data Privacy, 2025-2026

First Lab - 25 September 2025

Welcome to the first lab session of the Federated Learning & Data Privacy course! In this lab, we will implement the FedAvg algorithm, understand its components, and explore the impacts of its parameters on the model's performance. Let's start.

EXERCISE 1 - Get Familiar with the Code

Objective: Understand the structure and components of FedAvg through UML diagramming.

- **Setup**: Clone the repository TP1 from https://gitlab.inria.fr/arodio/fedcourse24 (https://gitlab.inria.fr/arodio/fedcourse24).
- **UML Diagram**:
    - Draw the UML diagram of the provided codebase. Focus on identifying the classes, their attributes, methods, and the relationships between them.
    - Use online tools like www.planttext.com for creating the UML diagram.
    - Add brief descriptions to each component to explain its functionality.
      Share your UML Diagram

EXERCISE 2 - Complete the Code

Objective: Understand the core mechanics of FedAvg by implementing the missing code.

- **Client Local Training**:
    - Complete the step method in client.py to perform local training on client data.
    - Complete the write_logs method in client.py to record training metrics.
- **Aggregator Orchestration**:
    - Complete the mix method in aggregator.py to aggregate client models.
    - Complete the update_clients method in aggregator.py to synchronize the updated global model with the clients.
- **Training Process**:
    - Implement the training loop in train.py to run the FedAvg algorithm.
- **Run the Experiment**:
    - Once you've completed the code, run the FedAvg algorithm using the script run.sh and observe the output.
      Share the complete code

EXERCISE 3 - The Effect of Local Epochs

Objective: Analyze how the number of local epochs affects the model's performance in a federated learning setting.

- **Experiment**:
    - Run FedAvg for different numbers of local epochs (e.g., 1, 5, 10, 50, 100).
    - Record the test accuracy for each setting.
- **Plot**:
    - Create a plot with the local epochs on the x-axis and test accuracy on the y-axis.
    - Interpret the plot and draw your conclusions. How does the number of local epochs influence the learning process and the final model accuracy? Were you expecting this result? Motivate your answer.
      Share the plot and your answers

BONUS EXERCISE - Complex Dataset & Model

Objective: Challenge yourself by integrating a more complex dataset and a deep learning model into the framework.

- **Implementation**:
    - Integrate the CIFAR10 dataset into the learning process.
    - Implement the MobileNet model using the PyTorch framework.
- **Training & Evaluation**:
    - Train the model using the FedAvg algorithm and evaluate its performance.
    - Test different hyperparameters setups.
    - Plot the training curves and comment the effect of the different hyperparameters choices.

Good luck, and don't hesitate to ask questions and collaborate with your peers!

You can send your plots, answers and code to: francesco.diana@inria.fr (mailto:francesco.diana@inria.fr).
You can both upload the code on GitHub and share the repository with me, or send me the zip file.

IMPORTANT: you have time until 08/10/2025 to send me your solution. Late answers will be penalized.

Exercise 3:

Plot interpretation:

- As we can see from the result, when the number of local epochs is 1, the client frequently sync with the server which leads to slow local progress, since each client only performs a small update before aggregation. And the global model need more communication rounds to be converged.

- When the number of local epochs is 5 or higher, the client trains longer locally before communicating. As we can see that the result of model training on 1 local epochs after 5 communication rounds and those on 5 local epochs after 1 communication round are equivalent. This accelerate global model training, need fewer rounds to be converged.