**Federated Learning & Data Privacy, 2024-2025**

**Third Lab - 11 February 2025**

Welcome to the third lab session of the Federated Learning & Data Privacy course! Today we will see how to implement Federated Learning in real network systems.

**EXERCISE 5 - Federated Learning with Flower**

**Objective**: Gain practical experience with the [Flower federated learning framework](https://flower.ai/docs/framework/index.html) by deploying a real, distributed federated learning experiment. Explore personalized federated learning algorithms and compare their performance against the standard FedAvg.

**EXERCISE 5.1 - Federated Learning on Real Networks**

**Goal**: Deploy a federated learning system using PyTorch and Flower to understand the setup and execution of federated learning in a networked environment.

**Setup**

1. **Clone the Flower repository and set up the PyTorch quickstart example**:

git clone --depth=1 https://github.com/adap/flower.git && mv flower/examples/quickstart-pytorch . && rm -rf flower && cd quickstart-pytorch

1. **Install the dependencies**:

pip install -e .

**Run Federated Learning**

* **Launch the simulation**

flwr run .

* Observe the federated training process initiated by PyTorch through Flower.

You can find complete instructions [here](https://github.com/adap/flower/tree/main/examples/quickstart-pytorch).

**EXERCISE 5.2 - Tackling Data Heterogeneity with FedProx**

Because the training time is quite long and consumes quite a lot of resources, I only do 20 rounds to show the results

**Objective**: Understand how the FedProx algorithm addresses the challenges posed by data heterogeneity in federated learning and compare its performance with the FedAvg algorithm.

* **FedProx Overview**: FedProx modifies the local training objective by introducing a proximal term, which aims to reduce local model drift by penalizing significant deviations from the global model. Review the FedProx algorithm [Federated Optimization in Heterogeneous Networks (Algorithm 2)](https://arxiv.org/abs/1812.06127).
* **Instructions**: Follow the tutorial on FedProx available at [Flower's documentation](https://flower.ai/docs/baselines/fedprox.html). Run experiments trying different number of rounds and different mu values.
* **Analysis**: Discuss the observed differences in performance between FedAvg and FedProx. Are there specific configurations (e.g., number of local epochs) where FedProx particularly outperforms FedAvg?

**RESULT :**

**Code Link** [**: https://github.com/DinhHuy1405/Federated-Learning-2024---2025/tree/main/TP3**](https://github.com/DinhHuy1405/Federated-Learning-2024---2025/tree/main/TP3)

A graph of a number of points

AI-generated content may be incorrect.

**Observations from the Experiment**

The graph provide results plots **validation accuracy** over **20 rounds** for:

1. **FedAvg (μ = 0)** → Traditional federated averaging.
2. **FedProx (μ = 2)** → FedProx with a strong regularization term.
3. **FedProx (μ = 0)** → Equivalent to FedAvg.

**Key Insights:**

1. **FedAvg (μ = 0) vs. FedProx (μ = 0):**
   * The two models should ideally behave the same, but **FedProx (μ = 0)** slightly outperforms **FedAvg (μ = 0)** in later rounds.
   * This could be due to small variations in implementation or stochastic effects.
2. **FedAvg (μ = 0) vs. FedProx (μ = 2):**
   * **FedAvg (μ = 0)** achieves **higher accuracy** over time compared to **FedProx (μ = 2)**.
   * **FedProx (μ = 2)** starts with similar accuracy but **converges more slowly**.
   * This indicates that a strong proximal term (μ = 2) **reduces client drift** but also **restricts adaptation**, leading to lower overall accuracy.
3. **FedProx (μ = 2) vs. FedProx (μ = 0):**
   * **FedProx (μ = 0) consistently outperforms FedProx (μ = 2)**.
   * This suggests that for this **CIFAR-10 dataset**, enforcing a strong **proximal constraint (μ = 2)** might overly restrict local model updates, preventing optimal learning.

**Analysis of Data Heterogeneity**

* **FedProx aims to mitigate data heterogeneity** by preventing local model drift.
* **However, choosing an appropriate μ value is crucial**:
  + **Small μ (close to 0)**: Allows more flexibility but may suffer from **client drift**.
  + **Large μ (e.g., μ = 2)**: Reduces drift but slows convergence and may lead to **suboptimal performance**.

**Impact of Local Epochs**

* If the number of **local epochs** per client increases, **FedProx may perform better than FedAvg**.
* **Why?**
  + When local training lasts multiple epochs, FedAvg tends to diverge more due to data heterogeneity.
  + **FedProx mitigates this by keeping local updates closer to the global model**.
  + Thus, in experiments with **higher local epochs**, we may see **FedProx outperform FedAvg**.

**BONUS EXERCISE - Personalized Federated Learning**

Because the training time is quite long and consumes quite a lot of resources, I only do 20 rounds to show the results

**Goal**: Evaluate a personalized federated learning algorithm using Flower, showing its possible advantages over the FedAvg algorithm.

**Choose one of the two proposed personalization algorithms:**

1. **Federated Learning with Personalization Layers (FedPer)**
   * **Overview**: FedPer implements personalization by allowing some neural network layers to be client-specific, making the model closer to individual data distributions.
   * **Instructions**: Follow the tutorial on FedPer available at [Flower's documentation](https://flower.ai/docs/baselines/fedper.html).

**Evaluation**

* Reproduce the tutorial and compare the results with FedAvg to highlight the benefits of personalization.

**RESULT:**

**Code Link:** [**https://github.com/DinhHuy1405/Federated-Learning-2024---2025/tree/main/TP3**](https://github.com/DinhHuy1405/Federated-Learning-2024---2025/tree/main/TP3)

A graph of a graph with a line graph

AI-generated content may be incorrect.

**Observations:**

1. **FedAvg (μ = 0)**
   * **Achieves the highest accuracy (~52%)** after 20 rounds.
   * Demonstrates a steady improvement in validation accuracy, indicating effective generalization across all clients.
   * Works best in scenarios where **data heterogeneity is moderate**.
2. **FedProx (μ = 2)**
   * Performs significantly worse than FedAvg, **reaching only ~25% accuracy**.
   * Converges slowly, highlighting **the trade-off between stability and adaptability**.
   * The **proximal term (μ)** helps prevent extreme local divergence but **limits learning** in less heterogeneous settings.
3. **FedPer**
   * **Performs the worst (~12%)**, showing minimal improvement over 20 rounds.
   * The use of **client-specific layers** does not improve performance in this setting, likely due to:
     + **Insufficient training rounds** for personalization layers to adapt.
     + **High variance in client data** making generalization difficult.
   * FedPer might be **more effective in highly heterogeneous datasets** where personalization is crucial.

**Key Insights**

* **FedAvg is the best approach for general federated learning scenarios** where personalization is not required.
* **FedProx can help stabilize training in highly non-IID datasets**, but excessive proximal constraints **can hinder learning**.
* **FedPer struggles in this setting** but might **perform better with a more diverse dataset** or **additional local training epochs**.