An Introduction to Federated Learning

Andreas Tritsarolis, 26/01/2024
Outline

- From Centralized to Decentralized Learning
- Federated Learning (FL)
- Categories of Federated Learning
- Federated vs Distributed Machine Learning
- Privacy Preservation Techniques
- Optimization in Federated Learning (FedAvg)
- Advances and Open Problems
- Programming Frameworks
From Centralized to Decentralized Learning

• Data is **Everywhere**…
  • Edge devices upload vast amounts of data per minute

• …and often **decentralized** across many parties

• Training a Machine Learning (ML) model
  • Requires a centralized dataset processed in a tightly integrated system

• Centralizing Process → **Privacy** nightmare
  • High Costs (e.g. storage, bandwidth, etc.)
  • High Sensitivity (e.g. finance, health, etc.)

• Delegating ML Training → **Performance** nightmare
  • Dataset too small → Underfitting/Overfitting
  • Features’ distribution → Biased target distribution
Federated Learning

- Federated Learning (FL)
  - Train a centralized model on decentralized data
- Edge devices collaboratively train an ML model
  - Keep the data decentralized
- Every participant keeps control of its own data
  - Harder to extract sensitive information


Federated Learning (cont.)

- Edge devices send model updates…
  - Client-Server architecture via Hub and Spokes
  - Decentralised Architecture via peer-to-peer
  - Hybrid Architecture

- Our scope of interest → Client-Server

Your phone personalizes the model locally, based on your usage (A). Many users' updates are aggregated (B) to form a consensus change (C) to the shared model, after which the procedure is repeated. [source]
Optimization in Federated Learning

- **Baseline → Federated Average**
- **Consider**
  - A set of $K$ parties (clients)
  - Each party $k$ holds a dataset $D_k$ of $n_k$ points
  - $D = D_1 \cup \ldots \cup D_K$ → the joint users’ dataset
    - $n = \sum_k n_k$ → #points
  - Aim → Minimize $F(\theta; D)$, w.r.t $\theta$, where

\[
F(\theta; D) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(\theta; D_k) \quad \text{and} \quad F_k(\theta; D_k) = \sum_{d \in D_k} f(\theta; d)
\]

Algorithm 1: FederatedAveraging. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**
- initialize $w_0$
- for each round $t = 1, 2, \ldots$ do
  - $m \leftarrow \max(C \cdot K, 1)$
  - $S_t \leftarrow$ (random set of $m$ clients)
  - for each client $k \in S_t$ in parallel do
    - $w_{t+1}^k \leftarrow$ ClientUpdate($k, w_t$)
    - $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

**ClientUpdate($k, w$):** // Run on client $k$
- $B \leftarrow$ (split $P_k$ into batches of size $B$)
- for each local epoch $i$ from 1 to $E$ do
  - for batch $b \in B$ do
    - $w \leftarrow w - \eta \nabla \ell(w; b)$
- return $w$ to server

Optimization in Federated Learning (cont.)

- Reddi et al. generalize the aforementioned scheme
  - Allow usage of adaptive optimization schemes (e.g. Adam)
- Clients use SGD
- Server use AdaGRAD, YOGI, Adam
- Advantages
  - Maintain communications costs (same as FedAvg)
  - Work in cross-device settings

Advances and Open Problems

• In non-i.i.d datasets → client drift
  • FedAvg fails to converge

• Prevent client drift → lowers convergence rate
  • Use fewer local updates
  • Smaller learning rates

Advances and Open Problems (cont.)

• (If) data distributions are very different
  • learning a single model $\rightarrow$ requires a (very) large number of parameters

• Lift the “one size fits all” requirement
  • i.e., the learned model should perform well for all parties

• Allow each party $k$ to learn a (simpler) personalized model $\theta_k$
  • design the objective so as to enforce some kind of collaboration


Advances and Open Problems (cont.)

- Including (among others)
  - Robustness against adversaries
    - e.g., Gradient leaks
  - Incentive mechanisms
    - e.g. Blockchain
  - Model diversity
    - Ensure fairness without access to sensitive attributes
  - Communication costs
    - Improve efficiency → Lower bandwidth costs
    - Accessible to edge devices

http://arxiv.org/abs/1610.05492


Experimental results on the AT&T Dataset with no DP
Programming Frameworks

- PySyft
  - pip install syft

- TensorFlow Federated
  - pip install --upgrade tensorflow-federated

- Flower
  - pip install flwr

- Opacus
  - pip install opacus

- TenSEAL
  - pip install tenseal
Programming Frameworks (cont.)

- **Flower: A Friendly Federated Learning Research Framework**
  - Originated from the University of Oxford
  - A unified approach to federated learning
  - Supports a wide variety of frameworks
    - TensorFlow, PyTorch, Scikit-Learn
  - Secure Aggregation → **Salvia**

<table>
<thead>
<tr>
<th>Feature</th>
<th>TFF</th>
<th>Syft</th>
<th>FedScale</th>
<th>LEAF</th>
<th>Flower</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-node simulation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Multi-node execution</td>
<td>*</td>
<td>✓</td>
<td>(✓)</td>
<td>***</td>
<td>✓</td>
</tr>
<tr>
<td>Scalability</td>
<td>*</td>
<td>**</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Heterogeneous clients</td>
<td>(✓)</td>
<td>***</td>
<td>**</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ML framework-agnostic</td>
<td>****</td>
<td>****</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Communication-agnostic</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Language-agnostic</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Baselines</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Labels: * Planned / ** Only simulated
*** Only Python-based / **** Only PyTorch and/or TF/Keras