UVA | SCHOOL *of* DATA SCIENCE

Overview of Federated Learning

Distributed Computing
School of Data Science
University of Virginia

Agenda

> Federated Learning (FL) Background

> Federated Averaging (FedAvg)

> FL in Production

> Non-IID Data

> Other Considerations and Challenges

FL Background

Decentralized vs Distributed Data

Decentralized
- Data is spread across machines
- No single entity controls all data
- **No single entity sees all data**
- Nodes can have their own policies and permissions

Distributed
- Data is spread across machines
- **System has central authority**

Setting

> Goal is to learn single, global ML model from decentralized, sensitive data which cannot be moved from clients

> Clients may be mobile devices (*cross-device*) or organizations (*cross-silo*)

Sampling

> Number of clients may be massive (mobile phones)

> For each round of training, algorithms randomly sample a small set of clients *C* from total pool

Primitives

> Algorithms use primitives to separate concerns

> Examples:
  - sum over selected clients
  - broadcast to selected clients

Data Generating Mechanism

> Data generated locally

> Data remains decentralized

Orchestration

> Central server organizes training

> Central server never sees raw data

Lifecycle

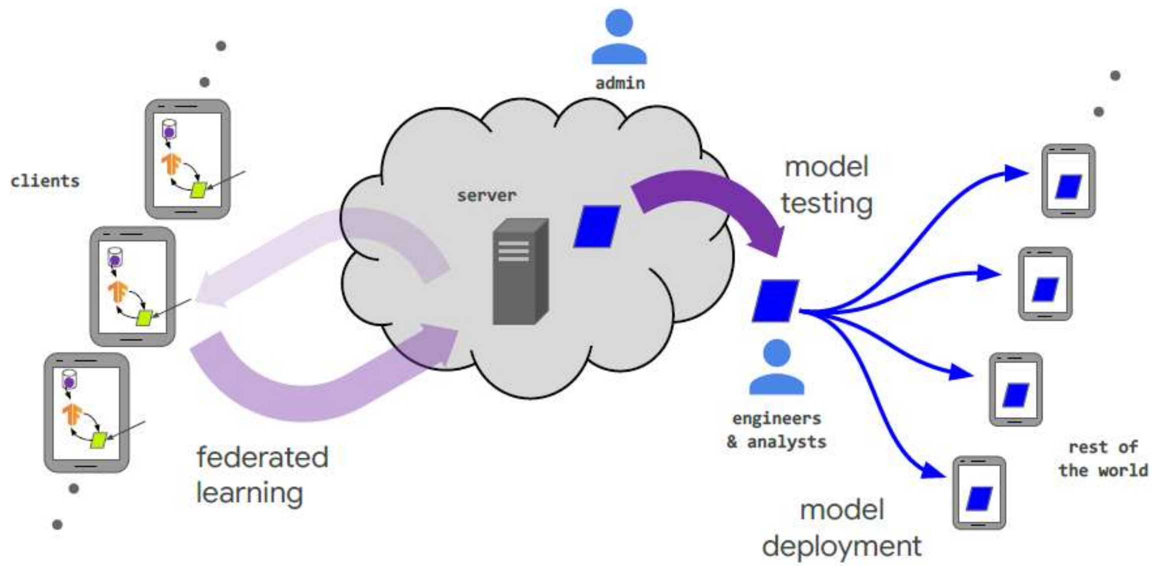

Figure 1: The lifecycle of an FL-trained model and the various actors in a federated learning system. This figure is revisited in Section 4 from a threat models perspective.

Source: Advances and Open Problems in Federated Learning. Kairouz et. al.

Federated Averaging (FedAvg)

FederatedAveraging

Paper below introduces algorithm:
FederatedAveraging (FedAvg)

> Local SGD on each client
> Updated local models averaged by server
to form updated global model

Communication-Efficient Learning of Deep Networks from Decentralized Data. McMahon et. al.

FedAvg Details

Each client minimizes local loss function

$$F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w).$$

These are aggregated to global loss

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w)$$

FedAvg Algorithm

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

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

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

Design Parameter: Updates at Client

> Each round of FL requires sending model weights or gradients over network

> Performing more updates at client can:
- reduce communication cost
- improve quality of local models

Design Parameter: Updates at Client, contd.

There is a tradeoff as downsides may include:

> Models drifting apart
> Slower convergence
> Global model may become biased if there is overfitting

Design Parameter: Updates at Client, contd.

These factors need to be balanced

Rules of Thumb for production:

> (cross-device) 1–5 local epochs per round usually optimal

> (cross-silo) 5–50 local epochs per round often improves convergence

Design Parameter: Aggregation

FedAvg takes weighted avg of local models, in proportion to size of local datasets

$$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^{k}$$

FL in Production

Production Frameworks – NVIDIA FLARE

Open-source FL framework

Focused on cross-silo FL (hospitals, institutions, enterprises)

Built for production

Production Frameworks – NVIDIA FLARE

Production users include:

> Mayo Clinic
> Mass General Brigham
> NVIDIA partners in oncology & radiology
> Global medical imaging consortia

Production Frameworks – NVIDIA FLARE

> Privacy filters can be applied

> Jobs submitted to controller

Production Frameworks - FedML

Why it's used:
> Cloud-native, lightweight, easy deployment.
> Handles on-device, edge, server-to-server FL.
> Offers MLOps, dashboards, experiment tracking.

Production users:
> Mobile OEMs
> IoT manufacturers
> Startups doing network optimization or personalization

Non-IID Data

IID Data

Many models require observations which are independent, identically distributed (IID)

FL is easiest when data is IID

But often, FL is conducted on non-IID data

Non-IID Data

Client-partitioned data may be non-IID:

> Two clients *i* and *j* may have different distributions
- Feature distribution skew
- Label distribution skew

Non-IID Data, contd.

> Same label, different features across *i*

> Same features, different label across *I*

> For single client, data may not be in random order (ordered by frame in video)

> For single client, distn may change over time

Data Imbalance

Different clients may hold different amounts of data

Dataset Shift

Clients contributing to FL model training are subject to criteria

This may create bias: clients where model will be deployed may be different

Example: training on devices with more memory than is needed for inference

Other Considerations and Challenges

Incentive Mechanisms

Often need to encourage honest participation

Particularly true when clients are competitors and there are free-riders

Can provide:

- Monetary incentives

- Model performance in proportion to contribution

Straggler Problem

Local-update SGD methods have all clients perform the same number of local updates

This can introduce bottleneck if any client unpredictably slows down or fails (*straggler*).

Straggler Problem: Some Solutions

1) Deadline-based participation

Server sets max round time
> Clients finishing in time upload their updates
> Stragglers ignored for the round

Straggler Problem: Some Solutions

2) Asynchronous FL

> Server doesn't sync rounds
> Client sends updates when ready
> Server updates global model with rule like:

$$w_{t+1} = w_t - \eta \cdot g_i$$

Multi-Task: Local Fine Tuning

One popular approach to FL models with non-IID:

1. Begin with FL training of global model
2. Deploy model to each client
3. Fine tune client models locally

Multi-Task: Models Trained on Subsets

Another approach is to train models on specific subsets

This falls between global model and local models

Subsets may be formed by geo, characteristics, clusters, etc.

References

Communication-Efficient Learning of Deep Networks from Decentralized Data. McMahon et. al.
Purpose: Introduced Federated Learning

Advances and Open Problems in Federated Learning. Kairouz et. al.
Purpose: Broad paper surveying challenges