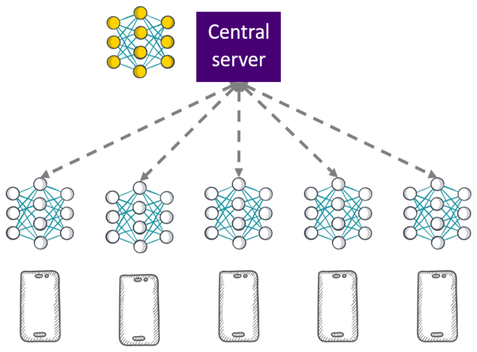
TP 3: Introduction to Federated Learning -Part I-

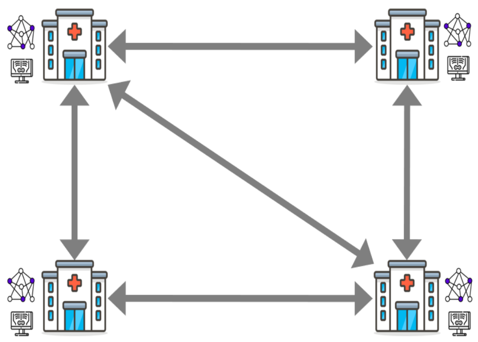
Othmane MARFOQ, Chuan Xu 14/03/2021

Introduction

Federated learning (FL) [5] "involves training statistical models over remote devices or siloed data centers, such as mobile phones or hospitals, while keeping data localized" [3] because of privacy concerns or limited communication resources. The definition implicitly distinguishes two different settings [1] (see Fig. 1): the *cross-device* scenario including a large number (millions or even more) of unreliable mobile/edge devices with limited computing capabilities and slow Internet connections, and the *cross-silo* scenario with at most a few hundreds of reliable data silos with powerful computing resources and high-speed access links.



(a) Cross-Device Federated Learning



(b) Cross-Silo Federated Learning

Figure 1: Two setting of federated learning: While cross-device FL (right) uses a server-client architecture, cross-silo FL (right) usually relies on peer-to-peer communications

Problem Formulation

Consider K clients and a loss function l. Client $1 \leq k \leq K$ has an underlying (unknown) distribution $\mathcal{D}_k$ over $\mathcal{X} \times \mathcal{Y}$, and wants to learn a parametric model w minimizing its true risk

$$\mathcal{L}_{\mathcal{D}_i}(w) = \mathbb{E}_{(x,y)\sim\mathcal{D}_i} l(h_w(x), y) \tag{1}$$

Client i does not have access to the underlying distribution $\mathcal{D}_k$. Instead it has access to n_k samples drawn i.i.d. from $\mathcal{D}_k$, denoted

$$\mathcal{S}_k = \left\{\xi_k^{(i)} = (x_k^{(i)}, y_k^{(i)})\right\}_{i=1}^{n_i}$$

Usually $n_k \ll n := \sum_{k'=1}^{K} n_{k'}$, thus collaboration between the K clients is needed in order to train better models.

The canonical federated learning problem involves learning a *single, global* statistical model from data stored on the K clients, without moving raw data, i.e., data is processed locally. In particular the goal is to minimize the following objective problem, corresponding to the empirical risk associated to problem 1

$$F(w) := \sum_{k=1}^{K} \frac{n_k}{n} F_k(w), \tag{2}$$

with

$$F_k(w) = \frac{1}{n_k} \sum_{i=1}^{n_k} l\left(h_w\left(x_k^{(i)}, y_k^{(i)}\right)\right) \tag{3}$$

Challenges

Federated learning poses multiple challenges that do not present in other classical problems, such as distributed learning in data center settings or traditional private data analyses [3, 5].

- **Expensive Communication**: Mobile devices are frequently offline or on slow or expensive connections, typically limited by an upload bandwidth of 1 MB/s or less.
- **Statistical Heterogeneity** The training data on a given client is typically based on the usage of the mobile device by a particular user, and hence any particular user's local dataset will not be representative of the population distribution. Moreover, some users will make much heavier use of the service or app than others, leading to varying amounts of local training data.
- **Systems Heterogeneity** The storage, computational, and communication capabilities of each device in federated networks may differ due to variability in hardware (CPU, GPU, memory), network connectivity (3G, 4G, 5G, wifi), and power (battery level).
- **Privacy Concerns** Finally, privacy is often a major concern in federated learning applications. Federated learning makes a step towards protecting data generated on each device by sharing model updates, e.g., gradient information, instead of the raw data. However, communicating model updates throughout the training process can nonetheless reveal sensitive information, either to a third-party, or to the central server [6, 4].

Federated Averaging

A simple, yet efficient, widely used algorithm in the context of federated learning is *federated averaging* (Alg. 1). Federated averaging (FedAvg) has three key parameters controlling the amount of computation at each round: C, the fraction of clients that perform computation on each round; E, then number of training steps each client makes over its local dataset on each round; and B, the local minibatch size used for the client updates.

Algorithm 1: Federated Averaging [5]

```
1  initialize $w_0$;
2  for iterations $t = 1, \dots, T$ do
3      $m \leftarrow \max(1, C \cdot K)$ ;
4      $\mathcal{S}_t \leftarrow$ (random set of $m$ clients);
5      server broadcasts $w_t$ to clients $k \in \mathcal{S}_t$;
6      for client $k \in \mathcal{S}_t$ in parallel do
7          $w_{t+1}^k \leftarrow$ ClientUpdate($k$, $w_t^k$) ;
8      end
9      $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
10 end
11 Function ClientUpdate($k$, $w$):
12     for $j = 1, \dots, E$ do
13         sample batch $\xi$ from $\mathcal{D}_k$ ;
14         $w \leftarrow w - \eta \nabla l(w; \xi)$;
15     end
16     return $w$;
```

Question

- What is the difference between the algorithm we have implemented for Parameter-Server architecture in TD2 and FedAvg described above?

Tasks

- First, clone / download the code corresponding to this session from the class repository. If you have already cloned the project previously, just do "git pull".
- In this session, we will consider MNIST [2] dataset as example. We prepared a script that splits this dataset across clients in a non-iid fashion (`data/generate_data.py`). Run this script in order to download and split the dataset.
- Complete `Learner` class in `learner.py`.
- Complete `Server` class in `server.py`
- Complete `Client` class in `client.py`
- Complete the function `main` in `main.py` by precising the model, criterion, metric and device
- reserve resources using `oarsub`.
- run the code using `python -m torch.launch.distributed`

References

[1] Peter Kairouz et al. *Advances and Open Problems in Federated Learning*. 2021. arXiv: `1912.04977 [cs.LG]`.

[2] Yann LeCun and Corinna Cortes. “MNIST handwritten digit database”. In: (2010). URL: `http://yann.lecun.com/exdb/mnist/`.

[3] Tian Li et al. “Federated Learning: Challenges, Methods, and Future Directions”. In: *IEEE Signal Processing Magazine* 37.3 (May 2020), pp. 50–60. ISSN: 1558-0792. DOI: `10.1109/msp.2020.2975749`. URL: `http://dx.doi.org/10.1109/MSP.2020.2975749`.

[4] Lingjuan Lyu et al. *Privacy and Robustness in Federated Learning: Attacks and Defenses*. 2020. arXiv: `2012.06337 [cs.CR]`.

[5] H. Brendan McMahan et al. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. 2017. arXiv: `1602.05629 [cs.LG]`.

[6] H. Brendan McMahan et al. *Learning Differentially Private Recurrent Language Models*. 2018. arXiv: `1710.06963 [cs.LG]`.