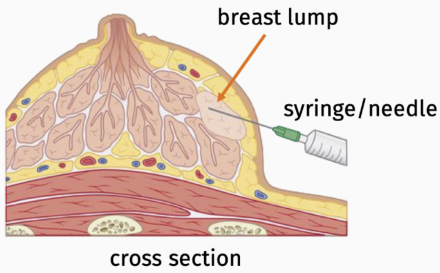
CS-GY 6923: Lecture 8
Federated Learning

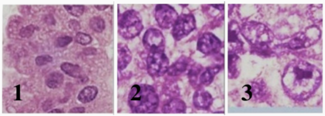
NYU Tandon School of Engineering, Akbar Rafiey

Motivating problem

Breast Cancer Biopsy: Determine if a breast lump in a patient is malignant (cancerous) or benign (safe).

- Collect cells from lump using fine needle biopsy.
- Stain and examine cells under microscope.
- Based on certain characteristics (shape, size, cohesion) determine if likely malignant or not).





Motivating problem

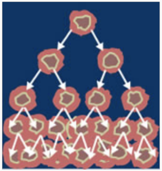
Demo: `demo_breast_cancer.ipynb`

Data: UCI machine learning repository

Breast Cancer Wisconsin (Original) Data Set

Download: Data Folder, Data Set Description

Abstract: Original Wisconsin Breast Cancer Database



Data Set Characteristics:	Multivariate	**Number of Instances:**	699	**Area:**	Life
Attribute Characteristics:	Integer	**Number of Attributes:**	10	**Date Donated**	1992-07-15
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	564320

`https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original)`

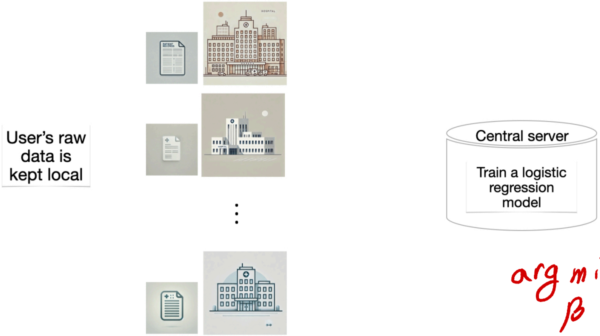
Logistic regression

- **Loss function**: "Logistic loss" aka "binary cross-entropy loss"

$$L(\beta) = -\sum_{i=1}^{n} y_i \log(h_\beta(\mathbf{x})) + (1 - y_i) \log(1 - h_\beta(\mathbf{x}))$$

- Do GD or SGD: $\nabla L(\beta) = \boldsymbol{X}^T (h(\boldsymbol{X}\beta) - \boldsymbol{y})$
- In this setting, all the data is collected on one central server.

Motivating problem



$$\arg\min_{\beta} L(\beta)$$

Question: How should we learn β in this case?

Federated (Decentralized) Learning

Some challenges with centralized settings

- Privacy concern: in many applications individuals do not trust the central server. Individuals want to keep their raw data local
- Computational concern: collecting all the data at one central server and doing computation could be infeasible.

Federated Learning

A decentralized learning paradigm where data remains local while models are trained collaboratively.

Communication-Efficient Learning of Deep Networks from Decentralized Data

H. Brendan McMahan **Eider Moore** **Daniel Ramage** **Seth Hampson** **Blaise Agüera y Arcas**
Google, Inc., 651 N 34th St., Seattle, WA 98103 USA

Sum decomposable loss functions

- Typical loss function in machine learning:

$$L(\beta) = \frac{1}{n} \sum_{j=1}^{n} \ell(\boldsymbol{\beta}, \mathbf{x}_j, y_j)$$

  where $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ are the training data point.
- In the FL setting the data points are distributed among different clients i.e., each client has its own local data.

The set up

$x_1 \; \cdots\cdots \; x_n$

$P_1 = \{2, 3, 10\} \qquad P_2 = \{12, 4, 100\}, \; \ldots$

The loss can be broken down to sum of the clients' local losses.

$$L(\boldsymbol{\beta}) =$$

$$\frac{1}{n}\left(\sum_{j\in\mathcal{P}_1}\ell_1(\boldsymbol{\beta},\mathbf{x}_j,y_j)+\sum_{j\in\mathcal{P}_2}\ell_2(\boldsymbol{\beta},\mathbf{x}_j,y_j)+\cdots+\sum_{j\in\mathcal{P}_K}\ell_K(\boldsymbol{\beta},\mathbf{x}_j,y_j)\right)$$

The set up

We assume there are K clients over which the data is partitioned, with $\mathcal{P}_k$ the set of indexes of data points on client k, with $n_k = |\mathcal{P}_k|$.

Objective: $\min_{\boldsymbol{\beta}} L(\boldsymbol{\beta}) = \sum_{k=1}^{K} \frac{n_k}{n} L_k(\boldsymbol{\beta})$

$L_k(\boldsymbol{\beta}) = \frac{1}{n_k} \sum_{j \in \mathcal{P}_k} L_k(\boldsymbol{\beta}, \mathbf{x}_j, y_j)$ is a user-specified loss function on client k local training dataset.

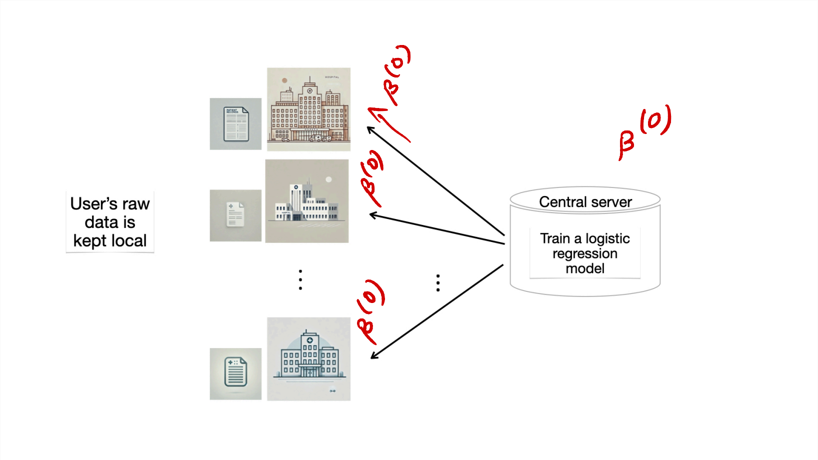
Algorithmic framework

Recall that Gradient Descent is a first order optimization method: Given a function L to minimize, we need to have:

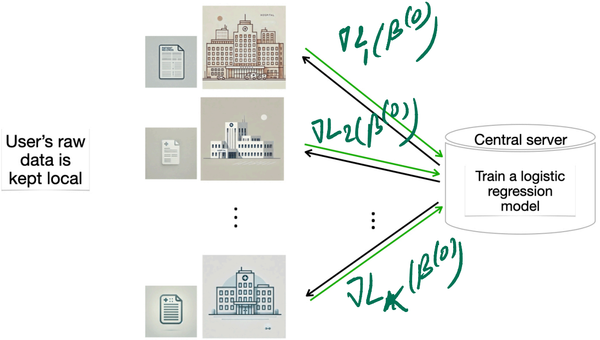
- Function oracle: Evaluate $L(\boldsymbol{\beta})$ for any $\boldsymbol{\beta}$
- Gradient oracle: Evaluate $\nabla L(\boldsymbol{\beta})$ for any $\boldsymbol{\beta}$.

Idea: We can actually compute the full gradient $\nabla L(\boldsymbol{\beta})$ without collecting clients raw data. How ?

Basic FedGD algorithm



Basic FedGD algorithm



$$\nabla L(\beta^{(0)}) = \sum_{k=1}^{K} \nabla L_k(\beta^{(0)})$$

Basic FedGD algorithm

Server chooses a starting model $\beta^{(0)}$.

For $i = 0, \dots, T-1$:

- Server broadcast the current model $\beta^{(i)}$ to all clients
  - All clients in parallel do:

    Compute local gradient:

    $\nabla L_k(\beta^{(i)}) = \frac{n_k}{n} \sum_{j \in \mathcal{P}_k} \nabla \ell_k(\beta^{(i)}, \mathbf{x}_j, y_j)$

    Send $\nabla L_k(\beta^{(i)})$ to server
- Server does the aggregation $\nabla L(\beta^{(i)}) = \sum_{k=1}^{K} \nabla L_k(\beta^{(i)})$
- Server updates $\beta^{(i+1)} = \beta^{(i)} - \eta \nabla L(\beta^{(i)})$

Return $\beta^{(T)}$.

Basic FedGD algorithm

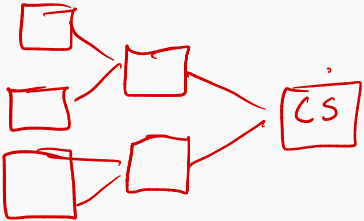
Question: What are some drawbacks of the Basic FedGD algorithm?

Basic FedGD algorithm

Question: What are some drawbacks of the Basic FedGD algorithm?

- Requires many communication rounds between the server and clients
- What if some of the clients are not available (not participating)
- Does not use much of clients' computation power

Reducing communication rounds



How can we reduce the number of communication rounds ?

Reducing communication rounds

How can we reduce the number of communication rounds ?

Clients can take several local steps.

Basic FedGD+

Server chooses a starting model $\beta^{(0)}$.

For $i = 0, \ldots, (T-1)/\tau$:

- Server broadcast the current model $\beta^{(i)}$ to clients
  - Clients in parallel do:
    - $\boldsymbol{w}_k^{(0)} = \boldsymbol{\beta}^{(i)}$
    - For $j = 0, \ldots, \tau - 1$:

      Compute $\nabla L_k(\boldsymbol{w}_k^{(j)})$

      Local GD update $\boldsymbol{w}_k^{(j+1)} = \boldsymbol{w}_k^{(j)} - \eta \nabla L_k(\boldsymbol{w}_k^{(j)})$

      $\hookrightarrow$ local gradient
    - Send ???? to server
- Server updates ????

Return $\beta^{(T)}$.

Basic FedGD+

Server chooses a starting model $\beta^{(0)}$.

For $i = 0, \dots, (T-1)/\tau$:

- Server broadcast the current model $\beta^{(i)}$ to clients
    - Clients in parallel do:
        - $\boldsymbol{w}_k^{(0)} = \boldsymbol{\beta}^{(i)}$
        - For $j = 0, \dots, \tau - 1$:

          Compute $\nabla L_k(\boldsymbol{w}_k^{(j)})$

          Local GD update $\boldsymbol{w}_k^{(j+1)} = \boldsymbol{w}_k^{(j)} - \eta \nabla L_k(\boldsymbol{w}_k^{(j)})$

Local model after GD

        - Send $\boldsymbol{w}_k^{(\tau)}$ to server
- Server updates $\beta^{(i+1)} = \sum_{k=1}^{K} \frac{n_k}{n} \boldsymbol{w}_k^{(\tau)}$

Return $\beta^{(T)}$.

How can we reduce the number of participating clients in each round?

Reducing number of participating clients

How can we reduce the number of participating clients in each round?

(Unbiased) client sampling.

FedAvg

Server chooses a starting model $\beta^{(0)}$.

For $i = 0, \ldots, (T-1)/\tau$:

- Server broadcast the current model $\beta^{(i)}$ to random subset of active clients
    - Each sampled client in parallel do:
        - $\boldsymbol{w}_k^{(0)} = \beta^{(i)}$ → global model
        - For $j = 0, \ldots, \tau - 1$:

          Compute $\nabla L_k(\boldsymbol{w}_k^{(j)})$

          Local GD update $\boldsymbol{w}_k^{(j+1)} = \boldsymbol{w}_k^{(j)} - \eta \nabla L_k(\boldsymbol{w}_k^{(j)})$
        - Send $\boldsymbol{w}_k^{(\tau)}$ to server
- Server updates $\beta^{(i+1)} = \sum_{k=1}^{K} \frac{n_k}{n} \boldsymbol{w}_k^{(\tau)}$

local steps

can be SGD

Return $\beta^{(T)}$.

A few asides

- We can still prove convergence for convex functions. (Under the same assumptions: bounded gradient norm, bounded radius.)
- The updates the server receives at each round is equal, in expectation, to the full update. (verify).
- We can use Stochastic Gradient Descent instead of GD
- Using linearity of expectation, we can prove unbiased estimation of our algorithm

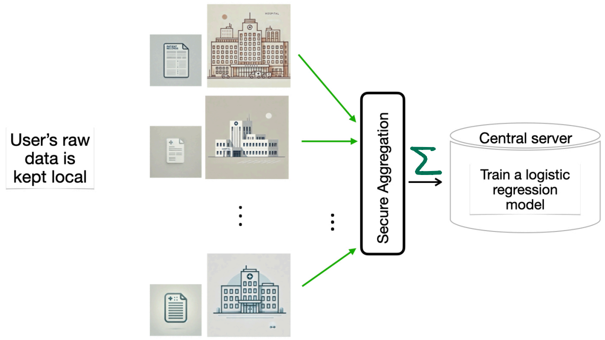
Some privacy concerns

- In some settings the server is not trusted at all, an adversary
- Clients do not want their individual updates be given to the server
- Adversarial attacks on gradients and models is a very active research area
  - Gradients or model parameters can leak sensitive information.

Question: How can we address these concerns ?

Secure Aggregation

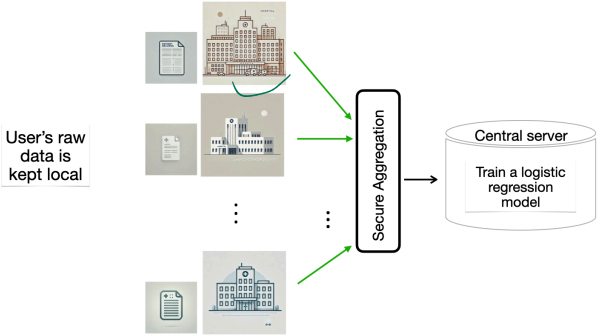
Enables clients to submit vector inputs, such that the server (an aggregator) can only decipher the combined update, not individual updates.



Secure Aggregation

Practical implementations using Secure Multi-Party Computation (MPC), Differential Privacy, Homomorphic Encryption, etc.

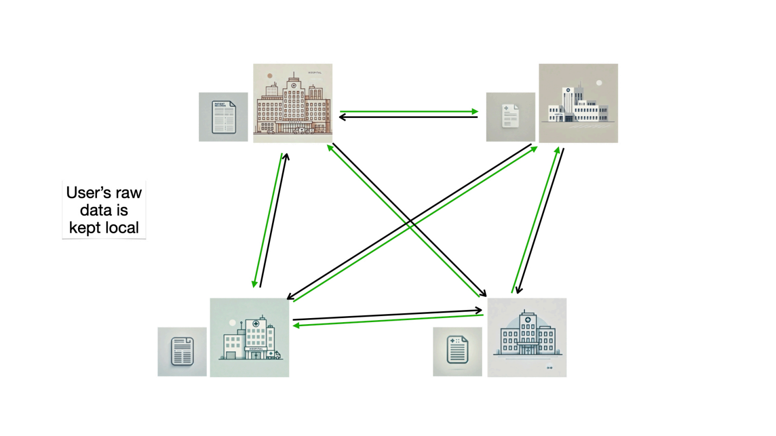
add local noise



FL without a central server

It is more difficult to analyze the convergence and behavior.

It requires privacy preserving communication over the entire graph.



Further reading on FL

Foundations and Trends® in Machine Learning

Advances and Open Problems in Federated Learning

Suggested Citation: Peter Kairouz, H. Brendan McMahan, *et al.* (2021), "Advances and Open Problems in Federated Learning", Foundations and Trends® in Machine Learning: Vol. 14, No. 1–2, pp 1–210. DOI: 10.1561/2200000083.

Peter Kairouz
Google Research
Kairouz@google.com

H. Brendan McMahan
Google Research

et al.

Advantages and challenges of FL

Advantages:

- Preserves data privacy by keeping data local.
- Enables collaborative training across multiple organizations or devices.
- Reduces risks of centralized data breaches.
- Facilitates training on diverse, real-world data without data sharing.

Challenges:

- Communication overhead between clients and server.
- Handling heterogeneous (non-iid) data distributions.
- Ensuring fairness across participants with varying data quality or quantity.
- Potentially high computational demands on client sides.