Optimization for Deep Learning

Lecture 14: Federated Learning

Kun Yuan

Peking University

Main contents in this lecture

- Federated Learning
- Communication-Efficient Algorithms
- Client Sampling

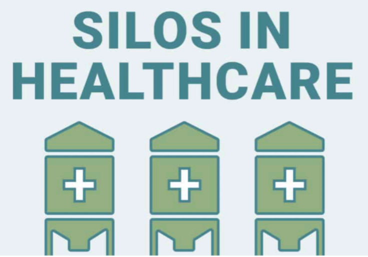
Data silos

- Organizations, companies, and persons collect their own data
- The amount of private data is typically small (except for giant companies)
- It is not enough to train a good model with the private small dataset
- Are we willing to share data with each other to train a good model together?
- Probably not. Private data are valuable and sensitive!

Data silos[1]

[1] Figure is from https://m.thepaper.cn/newsDetail_forward_25528954

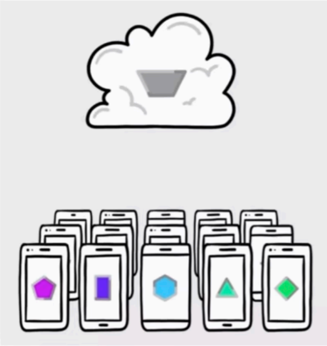
Data silos are everywhere



Cannot share data!

Data silos are more evident in edge AI

- Google wants to train a big machine learning model with mobile data
- Possible solution: collect user's data and train a model in data centers
- Users refuse to share data with Google



Federated learning

- How to collaboratively learn without sharing data? **Federated learning!**
- In Federated learning, n nodes collaborate to solve the following problem

$$\min_{x \in \mathbb{R}^d} \quad \frac{1}{n}\sum_{i=1}^{n} f_i(x) \quad \text{where} \quad f_i(x) = \mathbb{E}_{\xi_i \sim \mathcal{D}_i}[F(x; \xi_i)]$$

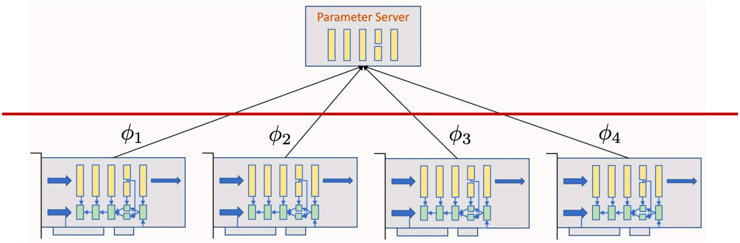
  - Each $f_i(x)$ is a local cost function in node i
  - ξ_i is a random variable representing local data in node i
  - Local distribution $\mathcal{D}_i$ is different from each other

Parallel SGD

$$\begin{aligned}
\phi_i^k &= x^k - \gamma \nabla F(x^k; \xi_i^k) && \text{(local update)} \\
x^{k+1} &= \frac{1}{n} \sum_{i=1}^{n} \phi_i^k && \text{(communication)}
\end{aligned}$$

- Each client samples local data and conducts local update
- The server will collects all local models and return the averaged one
- No data is sharing between clients and the server.

Parameter server



No data is uploaded to server

It is done! Federated learning is essentially parallel SGD!

Wait! You do not consider many issues yet

In Federated Learning

- Server cannot control user's device and data. Users join and leave the learning process at their will (that's why it is called **Federated** learning)
- Communication is way more expensive than computation
- Clients are not stable; they are not available when losing power or signal
- Data distributions are highly non-iid
- Trade-off between fairness and incentive

FedAvg: reduce communication frequency

- Parallel SGD communicates every iteration

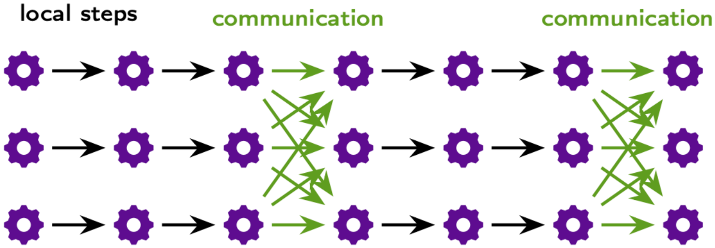
$$\phi_i^k = x^k - \gamma \nabla F(x^k; \xi_i^k) \qquad \text{(local update)}$$

$$x^{k+1} = \frac{1}{n} \sum_{i=1}^{n} \phi_i^k \qquad \text{(communication)}$$

- Federated average (FedAvg) (Konečný et al., 2016) communicates every τ iterations

$$x_i^{k+1} = \begin{cases} x_i^k - \gamma \nabla F(x_i^k; \xi_i^k) & \text{if } \mathrm{mod}(k, \tau) \neq 0 \\ \frac{1}{n} \sum_{i=1}^{n} x_i^k & \text{if } \mathrm{mod}(k, \tau) = 0 \end{cases}$$

FedAvg: illustration[2]



[2] This figure is from https://sstich.ch/files/Stich21-flworkshop.pdf

FedAvg: implementation

Algorithm 1: FedAvg

Server input: initial x and global learning rate γ_g;
Client input: local learning rate γ_ℓ;
for *each round* $r = 1, \cdots, R$ **do**
  server **communicates** x to all clients $i \in [n]$;
  for *each client* $i = 1, \cdots, n$ ***in parallel*** **do**
    initialize local model $y_i \leftarrow x$;
    for *local step* $k = 1, \cdots, K$ **do**
      sample $\xi_i \sim \mathcal{D}_i$ and updates $y_i \leftarrow y_i - \gamma_\ell \nabla F(y_i, \xi_i)$;
    each client **communicates** model change $\Delta y_i = y_i - x$ to the server;
  server updates $x \leftarrow x + \frac{\gamma_g}{n} \sum_{i=1}^{n} \Delta y_i$

FedAvg: assumptions

Assumption 1

Each $f_i(x)$ is L-smooth. Each local stochastic gradient $\nabla F(x;\xi)$ is unbiased and has bounded variance σ^2.

Assumption 2

We assume the difference between each local gradient $\nabla f_i(x)$ and the globally averaged gradient $\nabla f(x)$ is bounded above as follows

$$\frac{1}{n}\sum_{i=1}^{n}\|\nabla f_i(x) - \nabla f(x)\|^2 \le \zeta^2, \quad \forall x \in \mathbb{R}^d.$$

FedAvg: convergence

Theorem 1 (Karimireddy et al. (2020))

Under Assumptions 1-2, by setting local learning rate and global learning rate properly, it holds that

$$\frac{1}{R}\sum_{r=1}^{R}\mathbb{E}\|\nabla f(x^R)\|^2 = \mathcal{O}\left(\frac{\sigma}{\sqrt{nKR}} + \frac{\zeta^{2/3}}{R^{2/3}} + \frac{1}{R}\right)$$

- For parallel SGD with $T = KR$ iterations (and comm. rounds), it holds that

$$\frac{1}{R}\sum_{k=1}^{T}\mathbb{E}\|\nabla f(x^k)\|^2 = \mathcal{O}\left(\frac{\sigma}{\sqrt{nKR}} + \frac{1}{KR}\right)$$

  when $R \to \infty$, FedAvg and parallel SGD converge at the same rate $\frac{\sigma}{\sqrt{nKR}}$, but FedAvg requires K times fewer rounds of communications.

Data heterogeneity affects FedAvg

- Data heterogeneity ζ^2 can significantly influences the FedAvg performance

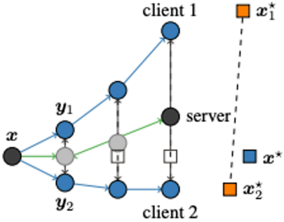


Table 2. Ablation results for varying degrees of data heterogeneity.

Method	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 2.5$	homog
FedAvg	45.0±0.2	52.9±0.1	54.4±0.2	54.9±0.4
FedProx	45.2±0.3	53.1±0.3	54.5±0.3	54.8±0.5
MOON	46.5±0.5	55.0±0.5	56.3±0.6	56.3±0.5

[M. Mendieta et. al., CVPR 2022]

Two sources of bias in federated learning

- There are two sources of bias in federated learning
- **Inner variance:** stochastic gradient within each client

$$\mathbb{E}\|F(x;\xi_i) - \nabla f_i(x)\|^2 \le \sigma^2, \quad \forall i \in [n].$$

- **Outer variance:** data heterogeneity due to different local data distributions

$$\frac{1}{n}\sum_{i=1}^{n} \|\nabla f_i(x) - \nabla f(x)\|^2 \le \zeta^2, \quad \forall\, x \in \mathbb{R}^d.$$

where $\nabla f(x) = (1/n)\sum_{i=1}^{n} \nabla f_i(x)$.

Recap: variance-reduction

- Recall the finite-sum optimization problem

$$\min_{x \in \mathbb{R}^d} \quad f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

- Stochastic variance-reduced gradient

$$g_k(x) = \nabla f_{i_k}(x) - u_{i_k}^k + \bar{u}^k, \text{ where } \bar{u}^k = \frac{1}{m} \sum_{j=1}^{n} u_i^k$$

  - quantity $\{u_j\}_{j=1}^n$ are n auxiliary variables with each $u_j \in \mathbb{R}^d$
  - index $i_k \in [n]$ is sampled uniformly randomly.

Recap: SAGA algorithm

Initialize x_0 arbitrarily; let $u_j^0 = \nabla f_j(x_0)$ and $\bar{u}^0 = \frac{1}{n}\sum_{j=1}^n u_j^0$

For $k = 0, 1, 2, ..., T-1$:

sample $i_k \in \{1, 2, \cdots, n\}$ uniformly randomly

update $g_k = \nabla f_{i_k}(x_k) - u_{i_k}^k + \bar{u}_k$

update $x_{k+1} = x_k - \gamma g_k$

update $u_{i_k}^{k+1} = \nabla f_{i_k}(x_k)$ and $u_{i_k}^{k+1} = u_{i_k}^k$ if $j \neq i_k$

update $\bar{u}^{k+1} = \frac{1}{n}\sum_{j=1}^n u_j^{k+1}$

Output x_T.

SAGA converges **without** assuming $\frac{1}{n}\sum_{i=1}^n \|\nabla f_i(x) - \nabla f(x)\|^2 \leq \zeta^2$

Extending SAGA to federated learning

- Recall the federated learning problem

$$\min_{x \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} f_i(x) \quad \text{where} \quad f_i(x) = \mathbb{E}_{\xi_i \sim \mathcal{D}_i}[F(x; \xi_i)]$$

- It is more challenging than finite-sum optimization since we cannot access $\nabla f_i(x)$ directly; we can only access stochastic gradient $\nabla F(x; \xi_i)$

- How to extend SAGA to the above problem setting? **Mini-batch!**

Extending SAGA to federated learning

Initialize x_0 arbitrarily; let $u_j^0 = \nabla f_j(x_0)$ and $\bar{u}^0 = \frac{1}{n}\sum_{j=1}^n u_j^0$

For $k = 0, 1, 2, ..., T-1$:

- sample $i_k \in \{1, 2, \cdots, n\}$ uniformly randomly
- sample a batch of data to update $\nabla F_B^k = \frac{1}{B}\sum_{b=1}^B \nabla F(x_k; \xi_{i_k}^b)$
- update $g_k = \nabla F_B^k - u_{i_k}^k + \bar{u}_k$
- update $x_{k+1} = x_k - \gamma g_k$
- update $u_{i_k}^{k+1} = \nabla F_B^k$ and $u_{i_k}^{k+1} = u_{i_k}^k$ if $j \neq i_k$
- update $\bar{u}^{k+1} = \frac{1}{n}\sum_{j=1}^n u_j^{k+1}$

Output x_T.

Extending SAGA to federated learning

We can further replace mini-batch with **local update**

For $k = 0, 1, 2, ..., T-1$:

sample $i_k \in \{1, 2, \cdots, n\}$ uniformly randomly

sample a batch of data $\mathcal{B} = \{\xi_{i_k}^b\}_{b=0}^{B-1}$ from local distribution $\mathcal{D}_{i_k}$

initialize $x_k^0 = x_k$

For $b = 0, 1, 2, ..., B-1$:

update $g_k^b = \nabla F(x_k^b; \xi_{i_k}^b) - u_{i_k}^k + \bar{u}_k$

update $x_k^{b+1} = x_k^b - \gamma g_k^b$

update $x^{k+1} = x_k^{B-1}$

update $u_{i_k}^{k+1} = \frac{1}{B} \sum_{b=0}^{B-1} \nabla F(x_k^b; \xi_{i_k}^b)$ and $u_{i_k}^{k+1} = u_{i_k}^k$ if $j \neq i_k$

update $\bar{u}^{k+1} = \frac{1}{n} \sum_{j=1}^{n} u_j^{k+1}$

Output x_T.

Scaffold: remove the influence of data heterogeneity

- Scaffold is a well-known federated learning algorithm that can remove the influence of data heterogeneity
- Scaffold is essentially a **distributed** version of SAGA with local update

Scaffold: remove the influence of data heterogeneity

Algorithm 2: Scaffold

Server input: initial x and $\bar{u}$ and global learning rate γ_g;
Client input: u_i and local learning rate γ_ℓ;
for *each round* $r = 1, \cdots, R$ **do**
  server **communicates** $(x, \bar{u})$ to all clients $i \in [n]$;
  for *each client* $i = 1, \cdots, n$ ***in parallel*** **do**
    initialize local model $y_i \leftarrow x$;
    for *local step* $k = 1, \cdots, K$ **do**
      sample $\xi_i \sim \mathcal{D}_i$ and updates $g_i = \nabla F(y_i; \xi_i) - u_i + \bar{u}$;
      update $y_i \leftarrow y_i - \gamma_\ell g_i$;
    update $u_i^+ \leftarrow u_i - \bar{u} + \frac{1}{K\gamma_\ell}(x - y_i)$; $// u_i^+ = \frac{1}{K}\sum_k \nabla F(y_i; \xi_i^b)$
    communicates $(\Delta y_i, \Delta u_i) = (y_i - x, u_i^+ - u_i)$ to the server;
    update $c_i \leftarrow c_i^+$;
  server updates $x \leftarrow x + \frac{\gamma_g}{n}\sum_{i=1}^n \Delta y_i$ and $c \leftarrow c + \frac{1}{n}\sum_{i=1}^n \Delta u_i$

Scaffold: convergence

Theorem 2

Under Assumptions 1, by setting local learning rate and global learning rate properly, it holds that

$$\frac{1}{R}\sum_{r=1}^{R}\mathbb{E}\|\nabla f(x^R)\|^2 = \mathcal{O}\left(\frac{\sigma}{\sqrt{nKR}} + \frac{1}{R}\right)$$

- No need to assume data heterogeneity bound
- Compared to the convergence rate of FedAvg

$$\frac{1}{R}\sum_{r=1}^{R}\mathbb{E}\|\nabla f(x^R)\|^2 = \mathcal{O}\left(\frac{\sigma}{\sqrt{nKR}} + \frac{\zeta^{2/3}}{R^{2/3}} + \frac{1}{R}\right)$$

we find Scaffold has removed the influence of data heterogeneity

Scaffold: simulation

		non-IID data				IID data	
	Epochs	0% similarity (sorted)		10% similarity		100% similarity (i.i.d.)	
		Num. of rounds	Speedup	Num. of rounds	Speedup	Num. of rounds	Speedup
SGD	1	317	(1×)	365	(1×)	416	(1×)
SCAFFOLD	1	**77**	(4.1×)	62	(5.9×)	60	(6.9×)
	5	152	(2.1×)	20	(18.2×)	10	(41.6×)
FedAvg	1	258	(1.2×)	74	(4.9×)	83	(5×)
	5	428	(0.7×)	34	(10.7×)	10	(41.6×)

less rounds with drift correction

no drift correction needed

What if not all clients are available during training?

Both FedAvg and Scaffold supports **partial client participation**

Algorithm 3: FedAvg with partial client participation

Server input: initial x and global learning rate γ_g;
Client input: local learning rate γ_ℓ;
for *each round* $r = 1, \cdots, R$ **do**
  server **communicates** x to all clients $i \in [n]$;
  a subset of clients $\mathcal{S} \subseteq [n]$ is sampled;
  for *each client* $i \in \mathcal{S}$ ***in parallel*** **do**
    initialize local model $y_i \leftarrow x$;
    for *local step* $k = 1, \cdots, K$ **do**
      sample $\xi_i \sim \mathcal{D}_i$ and updates $y_i \leftarrow y_i - \gamma_\ell \nabla F(y_i, \xi_i)$;
    each client **communicates** model change $\Delta y_i = y_i - x$ to the server;
  server updates $x \leftarrow x + \frac{\gamma_g}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta y_i$

FedAvg with partial client participation: convergence

Theorem 3

Under Assumptions 1-2, by setting local learning rate and global learning rate properly, FedAvg converges as follows

$$\frac{1}{R}\sum_{r=1}^{R}\mathbb{E}\|\nabla f(x^R)\|^2 = \mathcal{O}\left(\frac{\sqrt{\sigma^2 + K\zeta^2(1-\frac{m}{n})}}{\sqrt{mKR}} + \frac{\zeta^{2/3}}{R^{2/3}} + \frac{1}{R}\right)$$

- m is the number of clients that participate in the training process
- It is observed that partial client participation amplifies the influence of data heterogeneity and slows down the convergence

References I

J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International conference on machine learning*. PMLR, 2020, pp. 5132–5143.