Carnegie Mellon University

18786 Recitation

Distributed Machine Learning

APR 11, 2025

Haoran Zhang

Acknowledgement: Some slides are borrowed from 18667
https://www.andrew.cmu.edu/course/18-667/

Outline

- Convergence proof for centralized ML
- Distributed ML
- Federated Learning

Important properties of the objective function

(1) L-smooth

An upper bound. F(x) cannot change too fast

(2) μ-strongly convex

A lower bound. F(x) cannot change too slow

Lipschitz Smoothness

- A function $F(\mathbf{x})$ is L-Lipschitz smooth if its gradient is Lipschitz continuous, that is,

$$\|\nabla F(\mathbf{x}) - \nabla F(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\| \text{ for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$$

- Intuition: The slope of the function does not change too quickly – its rate of change is bounded by L
- Example: Is the following function Lipschitz smooth?

$$F(x) = \frac{1}{2}x^2$$
$$F'(x) = x$$
$$|F'(x) - F'(y)| = |x - y|$$

Thus, $F(x)$ is Lipschitz smooth with $L = 1$

Important properties of the objective function

(1) L-smooth

An upper bound. F(x) cannot change too fast

(2) μ-strongly convex

A lower bound. F(x) cannot change too slow

Equivalent Condition to Check L-smoothness

- A function is L-smooth if for any $\mathbf{x}$ and $\mathbf{y}$ it satisfies the following upper bound $\boxed{F(\mathbf{x}) \leq F(\mathbf{y}) + \nabla F(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2}$
- This is an important bound that we are going to frequently use in SGD convergence analysis
- Proof: See Appendix B of this lecture's reading `https://arxiv.org/pdf/1606.04838.pdf`
- For more conditions for Lipschitz smoothness check: `http://xingyuzhou.org/blog/notes/Lipschitz-gradient`

Important properties of the objective function

(1) L-smooth

An upper bound. F(x) cannot change too fast

(2) μ-strongly convex

A lower bound. F(x) cannot change too slow

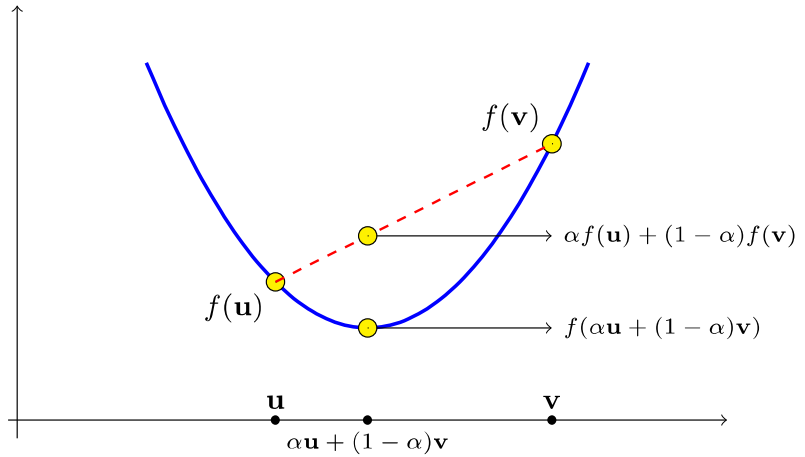
Recall: Convexity

Given $\alpha \in [0,1]$, the combination, $\alpha\mathbf{u} + (1-\alpha)\mathbf{v}$ of the points $\mathbf{u}, \mathbf{v}$ is called a *convex combination*.

DEFINITION 12.2 (Convex Function) Let C be a convex set. A function $f : C \to \mathbb{R}$ is convex if for every $\mathbf{u}, \mathbf{v} \in C$ and $\alpha \in [0,1]$,

$$f(\alpha\mathbf{u} + (1-\alpha)\mathbf{v}) \;\leq\; \alpha f(\mathbf{u}) + (1-\alpha) f(\mathbf{v}) \;.$$

In words, f is convex if for any $\mathbf{u}, \mathbf{v}$, the graph of f between $\mathbf{u}$ and $\mathbf{v}$ lies below the line segment joining $f(\mathbf{u})$ and $f(\mathbf{v})$. An illustration of a convex function, $f : \mathbb{R} \to \mathbb{R}$, is depicted in the following.



The *epigraph* of a function f is the set

$$\text{epigraph(f)} = \{(\mathbf{x}, \beta) : f(\mathbf{x}) \leq \beta\}. \tag{12.1}$$

Source: Textbook on Understanding Machine Learning by Shalev-Schwartz and Ben-David

It is easy to verify that a function f is convex if and only if its epigraph is a convex set. An illustration of a nonconvex function $f : \mathbb{R} \to \mathbb{R}$, along with its epigraph, is given in the following

Important properties of the objective function

(1) L-smooth
An upper bound. F(x) cannot change too fast

(2) μ-strongly convex
A lower bound. F(x) cannot change too slow

Consequences of Strong Convexity

A Lower Bound on the Function

If function $F(\mathbf{x})$ is c-strongly convex then

$$F(\mathbf{x}) \geq F(\mathbf{y}) + \nabla F(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \frac{1}{2} c \|\mathbf{x} - \mathbf{y}\|^2 \text{ for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$$

Bound on the Optimality Gap

$$2c(F(\mathbf{x}) - F(\mathbf{x}^*)) \leq \|\nabla F(\mathbf{x})\|^2 \text{ for all } \mathbf{x} \in \mathbb{R}^d$$

This is called the Polyak-Lojasiewicz (PL) inequality.

Important properties of the objective function

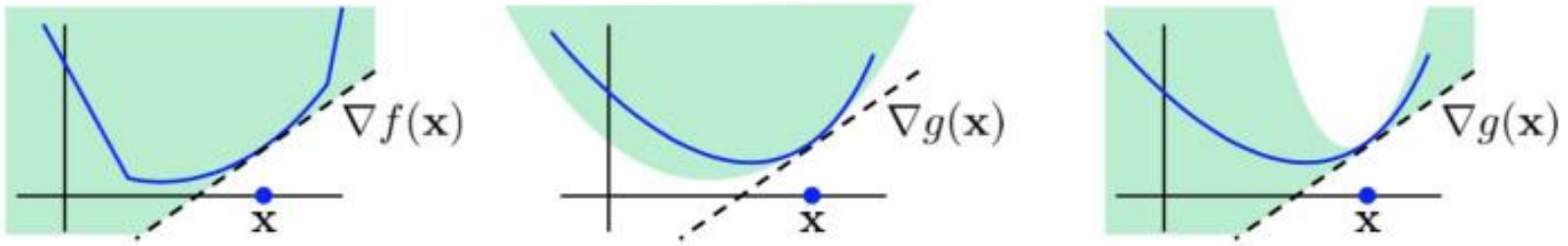
(1) L-smooth

An upper bound. F(x) cannot change too fast

(2) μ-strongly convex

A lower bound. F(x) cannot change too slow

Definition 2.1 (Convex Combination). *A convex combination of a set of n vectors* $\mathbf{x}_i \in \mathbb{R}^p$, $i = 1 \ldots n$ *in an arbitrary real space is a vector* $\mathbf{x}_{\boldsymbol{\theta}} := \sum_{i=1}^{n} \theta_i \mathbf{x}_i$ *where* $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$, $\theta_i \geq 0$ *and* $\sum_{i=1}^{n} \theta_i = 1$.



$f : \mathbb{R}^d \to \mathbb{R}$ — CONVEX FUNCTION

$g : \mathbb{R}^d \to \mathbb{R}$ — STRONGLY CONVEX FUNCTION

$g : \mathbb{R}^d \to \mathbb{R}$ — STRONGLY SMOOTH CONVEX FUNCTION

Figure 2.2: A convex function is lower bounded by its own tangent at all points. Strongly convex and smooth functions are, respectively, lower and upper bounded in the rate at which they may grow, by quadratic functions and cannot, again respectively, grow too slowly or too fast. In each figure, the shaded area describes regions the function curve is permitted to pass through.

Satisfies the upper and lower bounds given by

$$F(\mathbf{x}) \leq F(\mathbf{y}) + \nabla F(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2$$

$$F(\mathbf{x}) \geq F(\mathbf{y}) + \nabla F(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \frac{1}{2}c\|\mathbf{x} - \mathbf{y}\|^2 \text{ for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$$

Observe that $L \geq c$ needs to be true for both of the above inequalities to be satisfied

Definition 2.2 (Convex Set). *A set* $\mathcal{C} \in \mathbb{R}^p$ *is considered convex if, for every* $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ *and* $\lambda \in [0, 1]$, *we have* $(1 - \lambda)\mathbf{x} + \lambda\mathbf{y} \in \mathcal{C}$ *as well.*

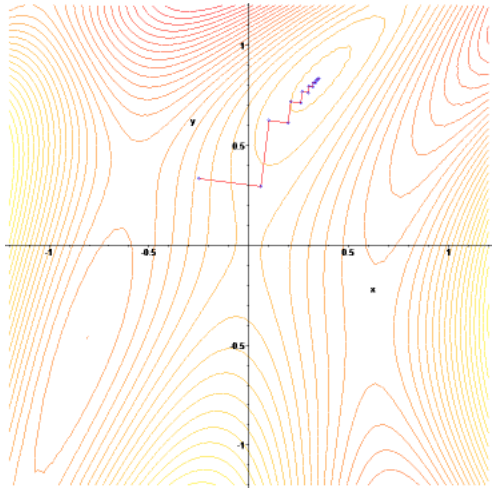
Definition 2.3 (Convex Function). *A function is considered convex if for every* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$ *we have* $f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle$, *where* $\nabla f(\mathbf{x})$ *is the gradient of* f *at* $\mathbf{x}$.

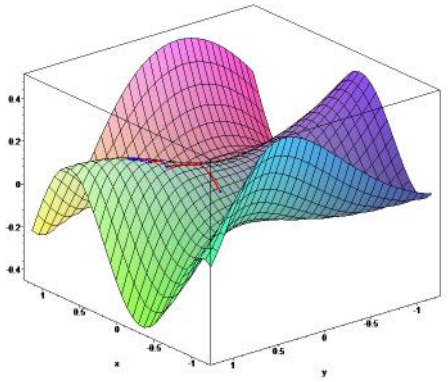
Recap: Gradient Descent (GD)

- GD starts from a random initial point $\mathbf{x}_0$ and updates $\mathbf{x}$ as follows:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla F(\mathbf{x}) \tag{4}$$

  for a small learning rate $\eta > 0$.
- For convex F and small enough η GD is guaranteed to converge to the optimal $\mathbf{x}^*$
- For non-convex functions it can get stuck at local minima





HOW FAST DOES IT CONVERGE to $\mathbf{x}^*$?

Assumptions on the Objective Function

- $F(\mathbf{x})$ is L-Lipschitz smooth. This implies that

$$F(\mathbf{x}) \leq F(\mathbf{y}) + \nabla F(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2$$

- $F(\mathbf{x})$ is c-strongly convex. This implies that

$$F(\mathbf{x}) \geq F(\mathbf{y}) + \nabla F(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \frac{1}{2}c\|\mathbf{x} - \mathbf{y}\|^2 \text{ for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$$

$$2c(F(\mathbf{x}) - F(\mathbf{x}^*)) \leq \|\nabla F(\mathbf{x})\|^2 \text{ for all } \mathbf{x} \in \mathbb{R}^d$$

We are going to show convergence of $F(\mathbf{x}_t)$ to the optimal value $F(\mathbf{x}^*)$ under these conditions

Convergence Analysis of GD

Starting with the Lipschitz smoothness condition with $\mathbf{x}$ replaced by $\mathbf{x}_{t+1}$ and $\mathbf{y}$ replaced by $\mathbf{x}_t$ we have

$$\begin{aligned} F(\mathbf{x}_{t+1}) - F(\mathbf{x}_t) &\leq \nabla F(\mathbf{x}_t)^\top (\mathbf{x}_{t+1} - \mathbf{x}_t) + \frac{L}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \\ &\leq \nabla F(\mathbf{x}_t)^\top (-\eta \nabla F(\mathbf{x}_t)) + \frac{L}{2}\| - \eta \nabla F(\mathbf{x}_t)\|^2 \\ &\leq \eta \left(1 - \frac{L}{2}\eta\right) (-\|\nabla F(\mathbf{x}_t)\|^2) \end{aligned}$$

Now using the strong convexity property $2c(F(\mathbf{x}) - F(\mathbf{x}^*)) \leq \|\nabla F(\mathbf{x})\|^2$

$$\begin{aligned} F(\mathbf{x}_{t+1}) - F(\mathbf{x}_t) &\leq \eta \left(1 - \frac{L}{2}\eta\right) (-\|\nabla F(\mathbf{x}_t)\|^2) \\ &\leq \eta \left(1 - \frac{L}{2}\eta\right) (-2c(F(\mathbf{x}_t) - F(\mathbf{x}^*))) \end{aligned}$$

Convergence Analysis of GD

Now using the strong convexity property $2c(F(\mathbf{x}) - F(\mathbf{x}^*)) \leq \|\nabla F(\mathbf{x})\|^2$

$$F(\mathbf{x}_{t+1}) - F(\mathbf{x}_t) \leq \eta \left(1 - \frac{L}{2}\eta\right) (-\|\nabla F(\mathbf{x}_t)\|^2)$$
$$\leq \eta \left(1 - \frac{L}{2}\eta\right) (-2c(F(\mathbf{x}_t) - F(\mathbf{x}^*)))$$

Assume that $\eta \leq \frac{1}{L}$. Then $\left(1 - \frac{L}{2}\eta\right) \geq \frac{1}{2}$. Thus,

$$F(\mathbf{x}_{t+1}) - F(\mathbf{x}_t) \leq -\eta c(F(\mathbf{x}_t) - F(\mathbf{x}^*))$$
$$F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*) + F(\mathbf{x}^*) - F(\mathbf{x}_t) \leq -\eta c(F(\mathbf{x}_t) - F(\mathbf{x}^*))$$
$$F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*) \leq -\eta c(F(\mathbf{x}_t) - F(\mathbf{x}^*)) + F(\mathbf{x}_t) - F(\mathbf{x}^*)$$
$$F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*) \leq (1 - \eta c)(F(\mathbf{x}_t) - F(\mathbf{x}^*))$$

Convergence Analysis of GD

From the previous slide we have

$$\begin{aligned} F(\mathbf{x}_{t+1}) - F(\mathbf{x}^*) &\leq (1-\eta c)(F(\mathbf{x}_t) - F(\mathbf{x}^*)) \\ &\leq (1-\eta c)^2(F(\mathbf{x}_{t-1}) - F(\mathbf{x}^*)) \text{ continuing recursively} \\ &\vdots \\ &\leq (1-\eta c)^{t+1}(F(\mathbf{x}_0) - F(\mathbf{x}^*)) \end{aligned}$$

And we are done!

Convergence of GD

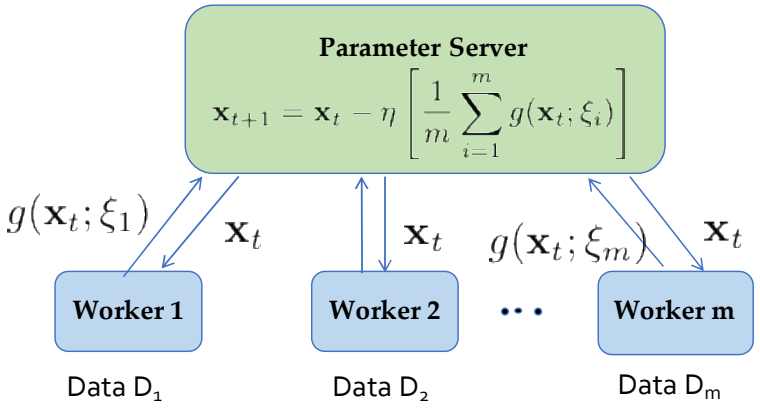
For a c-strongly convex and L-smooth function, if the learning rate $\eta \leq \frac{1}{L}$ and the starting point is $\mathbf{x}_0$ then $F(\mathbf{x}_t)$ after t gradient descent iterations is bounded as

$$F(\mathbf{x}_t) - F(\mathbf{x}^*) \leq (1-\eta c)^t(F(\mathbf{x}_0) - F(\mathbf{x}^*))$$

This proof is only for full-batch GD

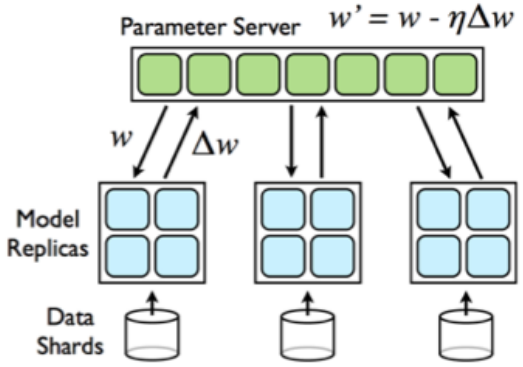
Distributed ML – Why distributed?

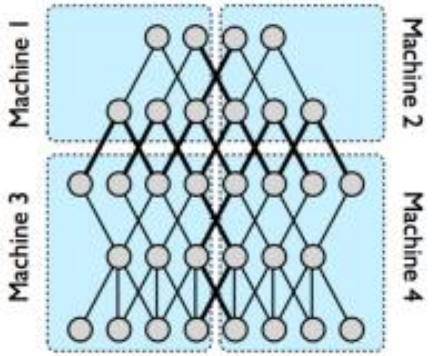
- For large training datasets, it can be prohibitively slow to conduct training at a single node.
- Solution: Split the dataset across m nodes into partitions D_1, D_2, ... D_m and perform data-parallel distributed training, using an algorithm that is called Synchronous Distributed SGD



Distributed ML in the Data-center Setting

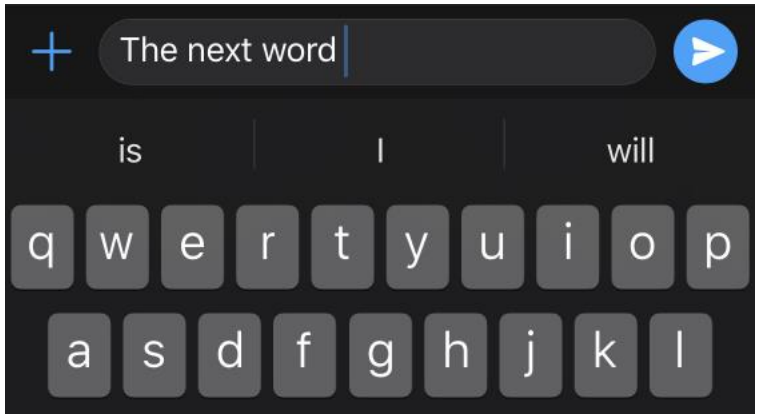
- We have a massive training dataset, which is shuffled and split across multiple nodes (servers in the cloud, often equipped with GPUs)
- A parameter server aggregates gradients from them using synchronous, asynchronous, local-update and/or gradient compression methods that we learned so far





Data Collection at Edge Clients

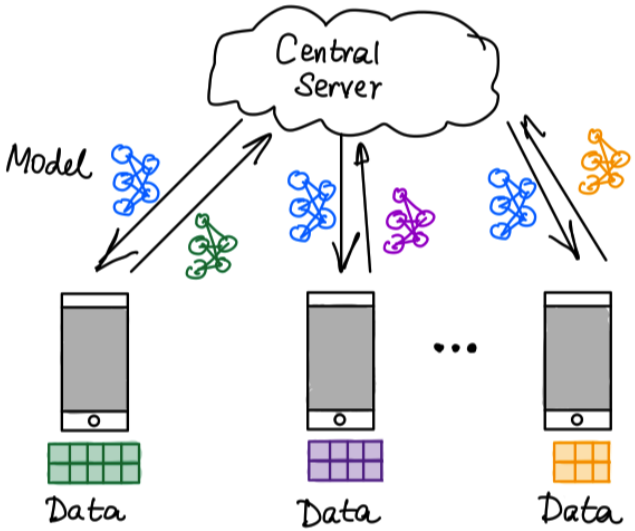
- Edge clients such as cell phone and IoT devices collect massive amounts of data that can be used to train informative ML models
- Consider the next word prediction service on cell phone keyboards
- Training data – What each user types on their phone
- This data can be used to train language models that can accurately predict the next word



Privacy concerns and high communication cost

From Distributed ML to Federated Learning

- Main Idea: Keep the data at the edge client, and bring the model training to the edge
- Sketch of the Algorithm:
  1. The aggregating server sends the current version of the model to available clients
  2. The clients train the model locally for a few iterations and send it back
  3. The server aggregates the models and goes back to step 1



The FedAvg Algorithm

Server Update: Initialize the model $\mathbf{x}_t$, and for each communication round $t = 1, \dots T$. At the t-th round, do the following:

- Select a set $\mathcal{S}_t$ of m out of the K clients, uniformly at random
- Perform ClientUpdate$(i, \mathbf{x}_t)$ at the chosen clients, and receive $\mathbf{x}_{t+1}^{(i)}$ from client $i \in \mathcal{S}_t$
- Aggregate the updates: $\mathbf{x}_{t+1} = \sum_{i \in \mathcal{S}_t} p_i \mathbf{x}_{t+1}^{(i)}$

Client Updates: ClientUpdate$(i, \mathbf{x}_t)$

- Initialize the local model $\mathbf{x}_{t,0}^{(i)} \leftarrow \mathbf{x}_t$ for $\tau_i = \frac{En_i}{B}$ local updates
- For local iteration index $j = 0, \dots, \tau_i - 1$ do the following:
  - Sample minibatch ξ_j from the local dataset $\mathcal{D}_i$, and make the local update
$$\mathbf{x}_{t,j+1}^{(i)} = \mathbf{x}_{t,j}^{(i)} - \eta g(\mathbf{x}_{t,j}^{(i)}, \xi_j)$$
- Return $\mathbf{x}_{t+1}^{(i)} \leftarrow \mathbf{x}_{t,\tau_i}^{(i)}$ to the server

Effect of Data Heterogeneity

- MNIST (handwritten digit dataset) IID experiment – shuffle and partition the data across 100 clients, each receiving 600 examples
- MNIST (handwritten digit dataset) non-IID experiment – the data sorted by labels and divided into 200 shards of size 300 and each of the 100 clients receives 2 shards (at most 2 digits)
- Two different neural networks, a 2-hidden layer perceptron (2NN) and 2-layer convolutional network (CNN) trained on these datasets

2NN	IID		NON-IID	
C	$B = \infty$	$B = 10$	$B = \infty$	$B = 10$
0.0	1455	316	4278	3275
0.1	1474 (1.0×)	87 (3.6×)	1796 (2.4×)	664 (4.9×)
0.2	1658 (0.9×)	77 (4.1×)	1528 (2.8×)	619 (5.3×)
0.5	— (—)	75 (4.2×)	— (—)	443 (7.4×)
1.0	— (—)	70 (4.5×)	— (—)	380 (8.6×)
CNN, $E = 5$				
0.0	387	50	1181	956
0.1	339 (1.1×)	18 (2.8×)	1100 (1.1×)	206 (4.6×)
0.2	337 (1.1×)	18 (2.8×)	978 (1.2×)	200 (4.8×)
0.5	164 (2.4×)	18 (2.8×)	1067 (1.1×)	261 (3.7×)
1.0	246 (1.6×)	16 (3.1×)	— (—)	97 (9.9×)

McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." *Artificial intelligence and statistics*. PMLR, 2017.

Table 1: Effect of the client fraction C on the MNIST 2NN with $E = 1$ and CNN with $E = 5$. Note $C = 0.0$ corresponds to one client per round; since we use 100 clients for the MNIST data, the rows correspond to 1, 10 20, 50, and 100 clients. Each table entry gives the number of rounds of communication necessary to achieve a test-set accuracy of 97% for the 2NN and 99% for the CNN, along with the speedup relative to the $C = 0$ baseline. Five runs with the large batch size did not reach the target accuracy in the allowed time.

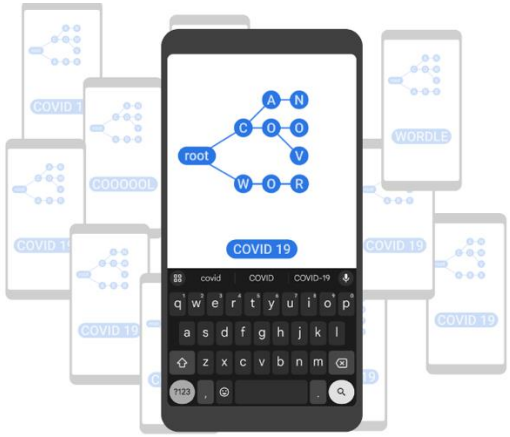
2NN	IID		NON-IID	
C	$B = \infty$	$B = 10$	$B = \infty$	$B = 10$
0.0	1455	316	4278	3275
0.1	1474 (1.0×)	87 (3.6×)	1796 (2.4×)	664 (4.9×)
0.2	1658 (0.9×)	77 (4.1×)	1528 (2.8×)	619 (5.3×)
0.5	— (—)	75 (4.2×)	— (—)	443 (7.4×)
1.0	— (—)	70 (4.5×)	— (—)	380 (8.6×)
CNN, $E = 5$				
0.0	387	50	1181	956
0.1	339 (1.1×)	18 (2.8×)	1100 (1.1×)	206 (4.6×)
0.2	337 (1.1×)	18 (2.8×)	978 (1.2×)	200 (4.8×)
0.5	164 (2.4×)	18 (2.8×)	1067 (1.1×)	261 (3.7×)
1.0	246 (1.6×)	16 (3.1×)	— (—)	97 (9.9×)

McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." *Artificial intelligence and statistics*. PMLR, 2017.

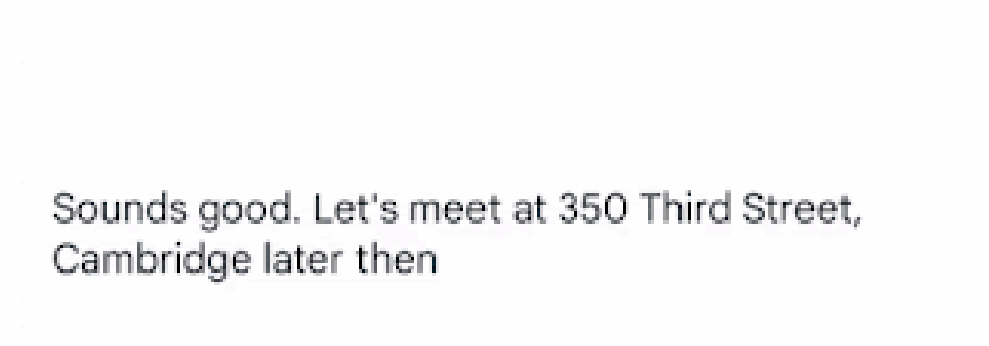
Multi-Model Federated Learning

Examples: Multiple FL applications on one device.

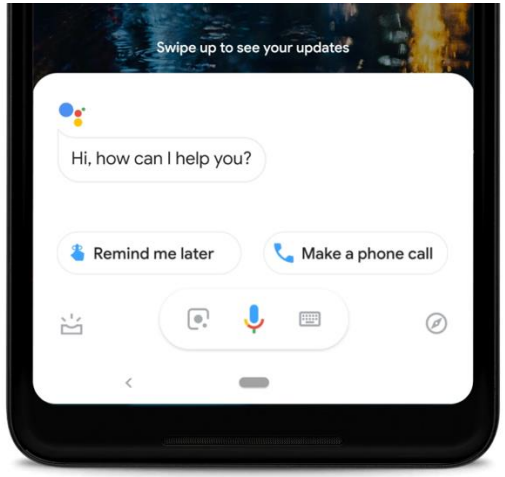
Keyboard prediction



Predicting text selection



Speech model



Source: federated.withgoogle.com

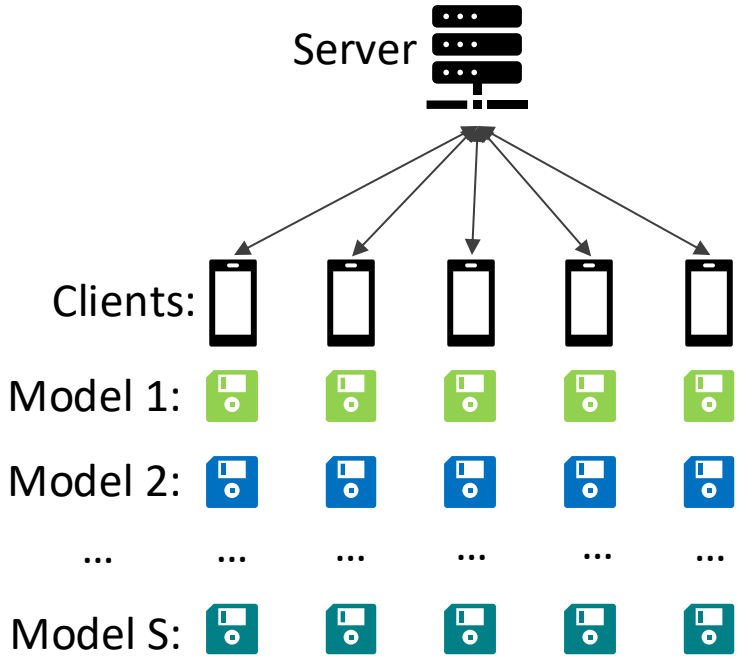
Multi-Model Federated Learning

Key assumptions from previous work [1]

In each round, the server only allows partial participation, and each active client can only train one model.

1) Partial Participation: reduce communication cost

2) Only train one model: computational constraints



Multi-model federated learning

[1] Bhuyan, Neelkamal, Sharayu Moharir, and Gauri Joshi. “Multi-model federated learning with provable guarantees.” *EAI International Conference on Performance Evaluation Methodologies and Tools*. Cham: Springer Nature Switzerland, 2022.

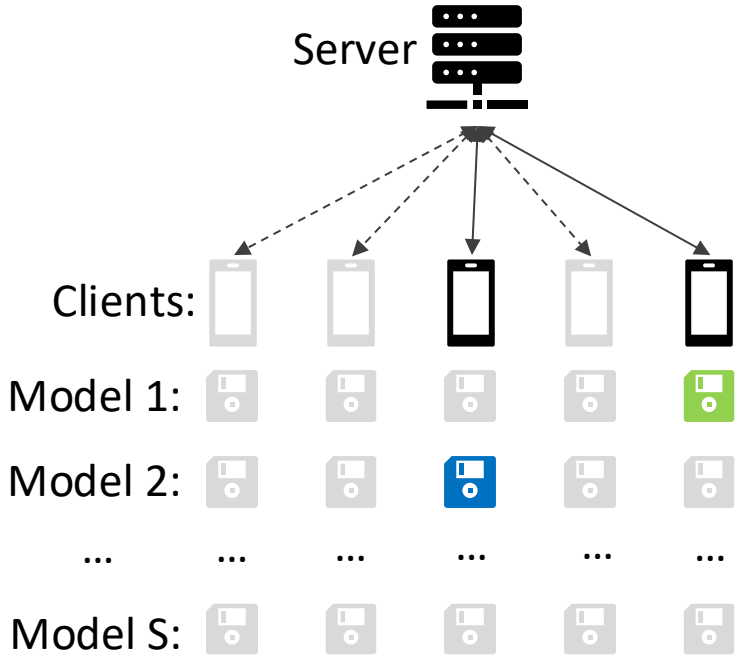
Multi-Model Federated Learning

Key assumptions from previous work [1]

In each round, the server only allows partial participation, and each active client can only train one model.

1) Partial Participation: reduce communication cost

2) Only train one model: computational constraints

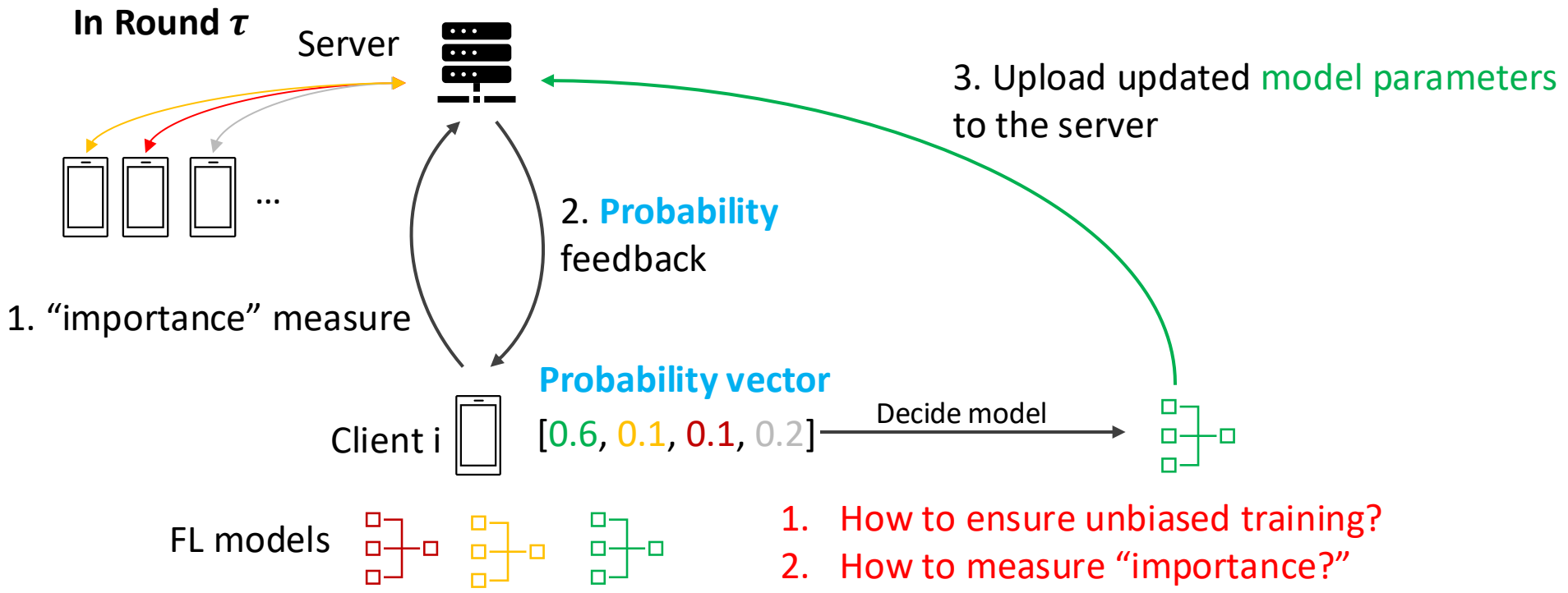


Multi-model federated learning

[1] Bhuyan, Neelkamal, Sharayu Moharir, and Gauri Joshi. “Multi-model federated learning with provable guarantees.” *EAI International Conference on Performance Evaluation Methodologies and Tools*. Cham: Springer Nature Switzerland, 2022.

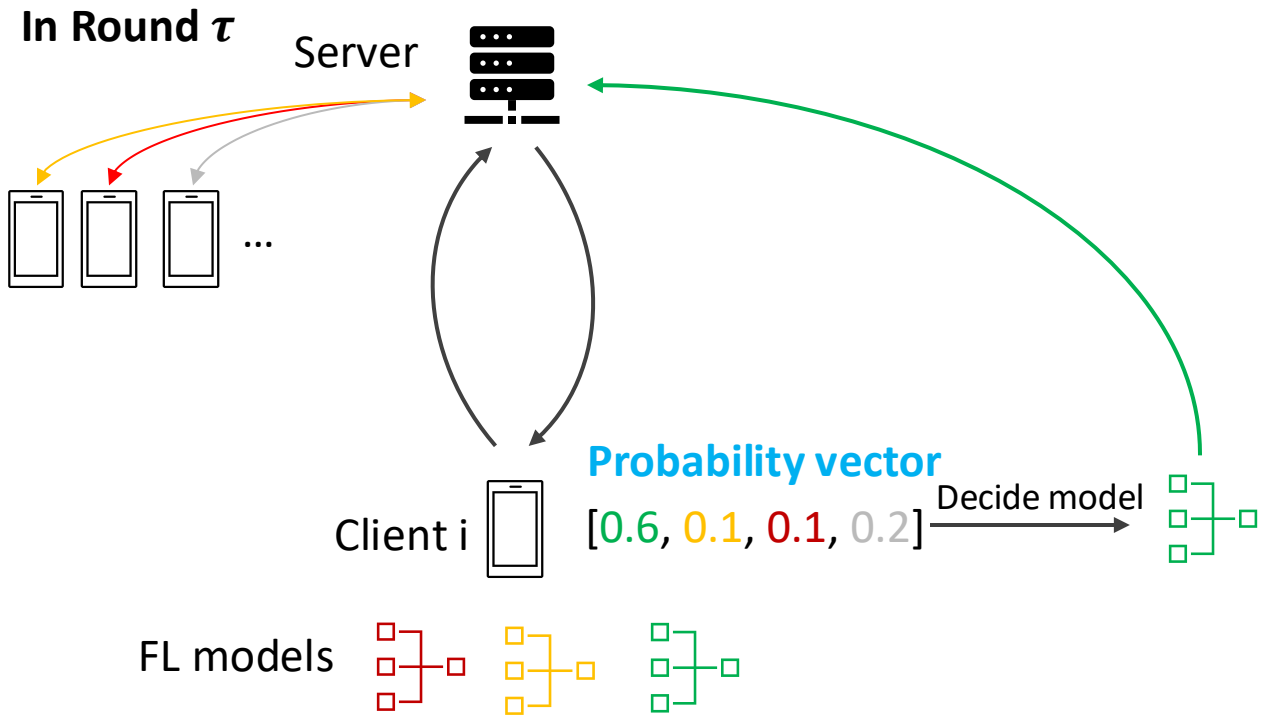
MMFL Optimal Variance-Reduced Sampling

Idea: the server prefers selecting more "important" clients.



1. How to ensure unbiased training?
2. How to measure "importance?"

MMFL Optimal Variance-Reduced Sampling



In each global round (Aggregation):

$$w_s^{\tau+1} = w_s^{\tau} - \sum_{i \in \mathcal{A}_{\tau,s}} \frac{d_{i,s}}{p_{s|i}^{\tau}} U_{i,s}^{\tau}$$

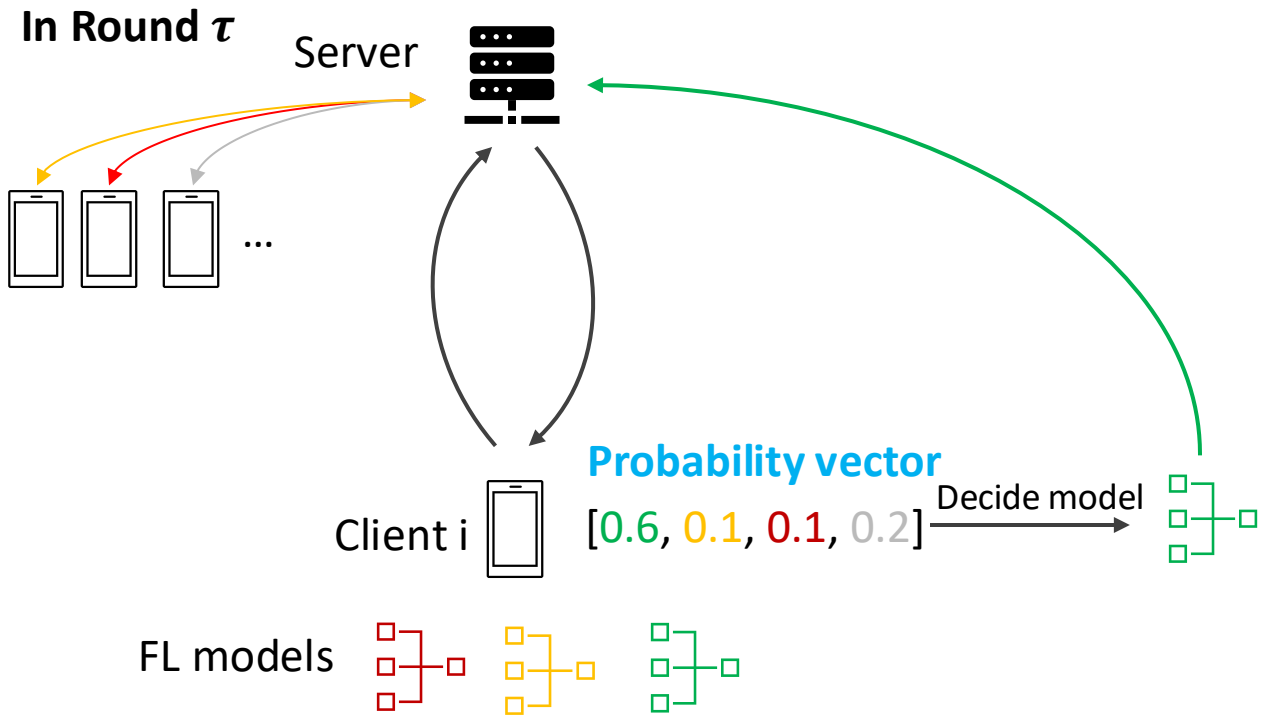
$d_{i,s} = \frac{n_{i,s}}{\sum_{j=1}^{N} n_{j,s}}$: dataset size ratio.

$U_{i,s}^{\tau} = \eta_\tau \sum_{t=1}^{K} \nabla f_{i,s}^{t,\tau}$: local update.

$p_{s|i}^{\tau}$: probability of assigning client i to model s.

$\mathcal{A}_{\tau,s}$: set of assigned clients for model s.

MMFL Optimal Variance-Reduced Sampling



In each global round (Aggregation):

$$w_s^{\tau+1} = w_s^{\tau} - \sum_{i \in \mathcal{A}_{\tau,s}} \frac{d_{i,s}}{p_{s|i}^{\tau}} U_{i,s}^{\tau}$$

Unbiased Training:

$$\mathbb{E}\left[\sum_{i \in \mathcal{A}_{\tau,s}} \frac{d_{i,s}}{p_{s|i}^{\tau}} U_{i,s}^{\tau}\right]$$

$$= \mathbb{E}\left[\sum_{i=1}^{N} \frac{d_{i,s}}{p_{s|i}^{\tau}} U_{i,s}^{\tau} 1_{i \in \mathcal{A}_{\tau,s}}\right]$$

$$= \sum_{i=1}^{N} d_{i,s} U_{i,s}^{\tau}$$

MMFL optimal variance-reduced sampling

Aggregation:

$$w_s^{\tau+1} = w_s^{\tau} - \sum_{i \in \mathcal{A}_{\tau,s}} \frac{d_{i,s}}{p_{s|i}^{\tau}} U_{i,s}^{\tau}$$

Random Variable X

$\mathbb{E}[X]$ is given.

MMFL optimal variance-reduced sampling

Aggregation:

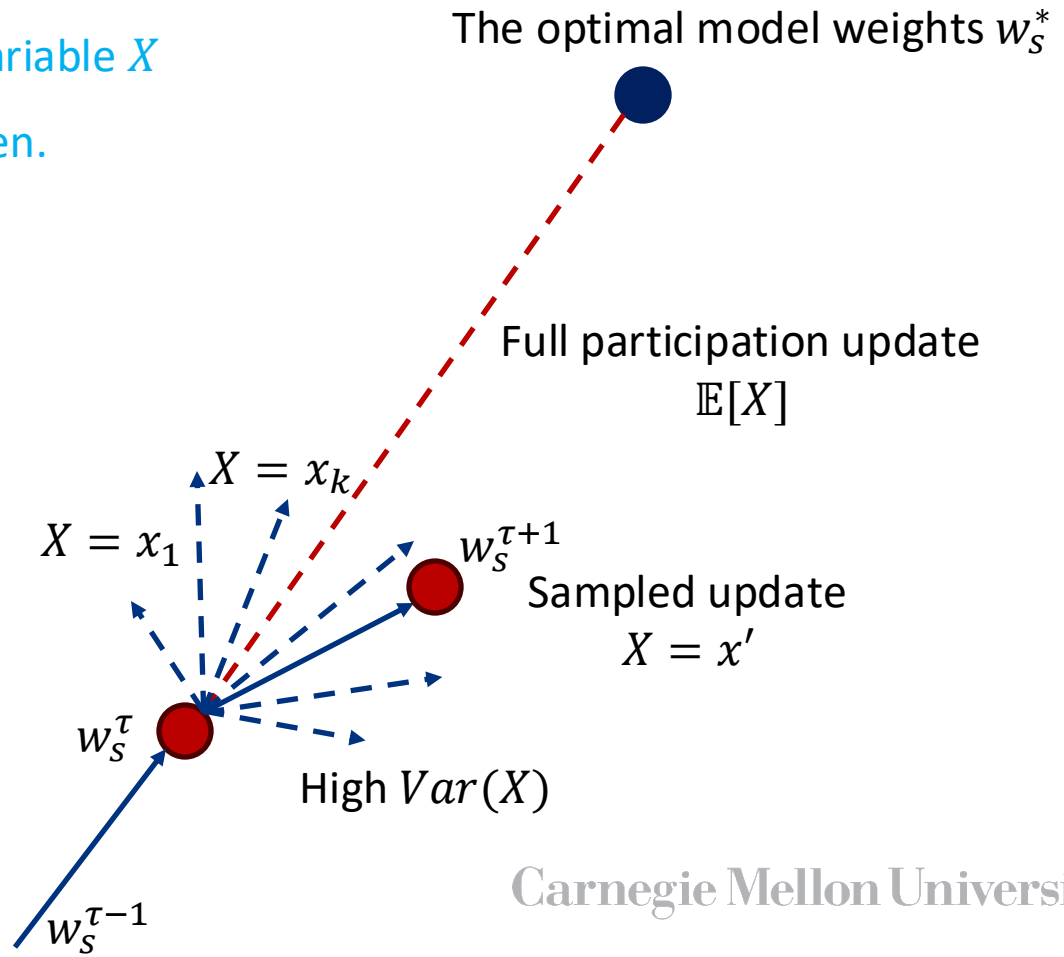
$$w_s^{\tau+1} = w_s^{\tau} - \sum_{i \in \mathcal{A}_{\tau,s}} \frac{d_{i,s}}{p_{s|i}^{\tau}} U_{i,s}^{\tau}$$

Random Variable X

$\mathbb{E}[X]$ is given.

High variance of X can make the training unstable…
Therefore, define our objective:

$$\min_{\{p_{s|i}^{\tau}\}} \sum_{s=1}^{S} \mathbb{E}_{\mathcal{A}_{\tau,s}} \left[\| \sum_{i \in \mathcal{A}_{\tau,s}} \frac{d_{i,s}}{p_{s|i}^{\tau}} U_{i,s}^{\tau} - \sum_{i=1}^{N} d_{i,s} U_{i,s}^{\tau} \|^2 \right]$$



MMFL optimal variance-reduced sampling

Aggregation:

$$w_s^{\tau+1} = w_s^{\tau} - \sum_{i \in \mathcal{A}_{\tau,s}} \frac{d_{i,s}}{p_{s|i}^{\tau}} U_{i,s}^{\tau}$$

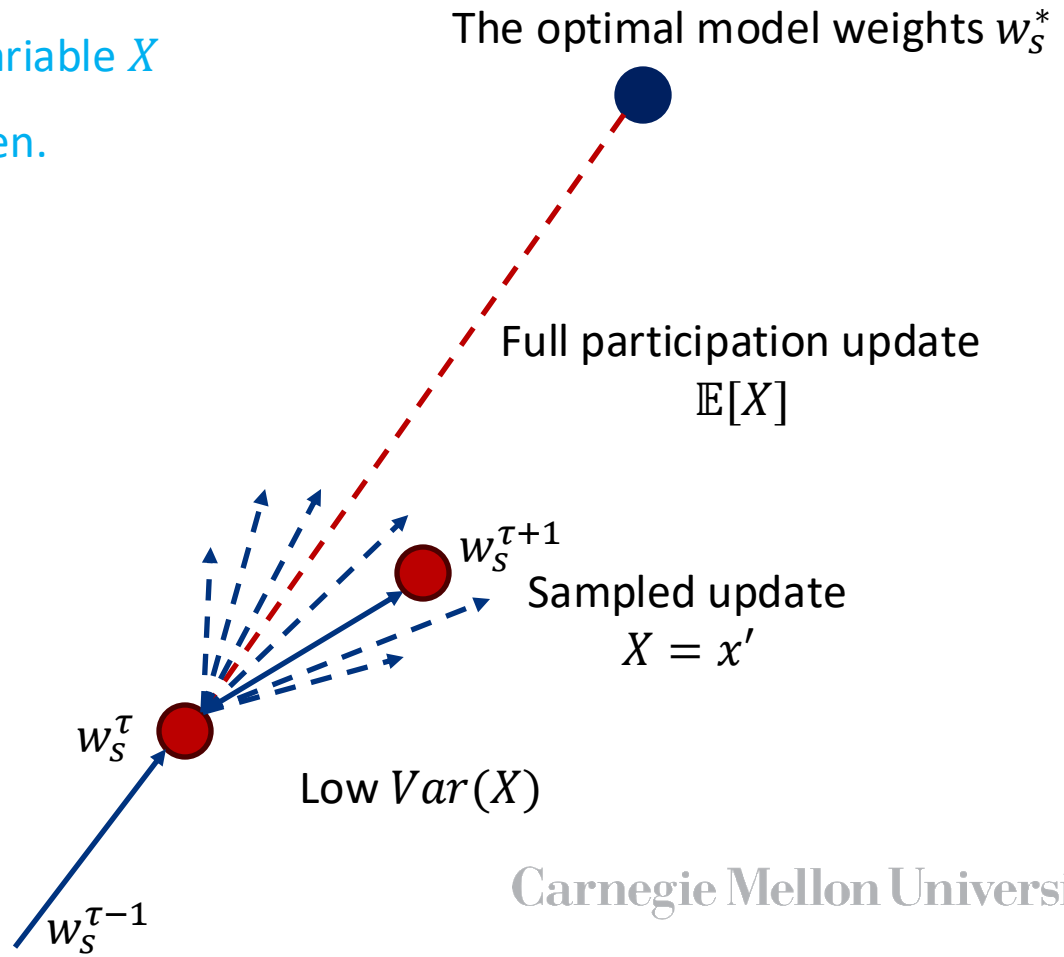
Random Variable X

$\mathbb{E}[X]$ is given.

High variance of X can make the training unstable…
Therefore, define our objective:

$$\min_{\{p_{s|i}^{\tau}\}} \sum_{s=1}^{S} \mathbb{E}_{\mathcal{A}_{\tau,s}} \left[\| \sum_{i \in \mathcal{A}_{\tau,s}} \frac{d_{i,s}}{p_{s|i}^{\tau}} U_{i,s}^{\tau} - \sum_{i=1}^{N} d_{i,s} U_{i,s}^{\tau} \|^2 \right]$$

Notice: variance is an ideal objective to stabilize the training, but there could be other factors… (will further discuss later)



MMFL Optimal Variance-Reduced Sampling

Minimizing the variance of update

$$\min_{\{p^{\tau}_{s|i}\}} \sum_{s=1}^{S} \mathbb{E}_{\mathcal{A}_{\tau,s}} \left[\| \sum_{i \in \mathcal{A}_{\tau,s}} \frac{d_{i,s}}{p^{\tau}_{s|i}} U^{\tau}_{i,s} - \sum_{i=1}^{N} d_{i,s} U^{\tau}_{i,s} \|^2 \right]$$

$$\text{s.t. } p^{\tau}_{s|i} \geq 0, \sum_{s=1}^{S} p^{\tau}_{s|i} \leq 1, \sum_{s=1}^{S} \sum_{i=1}^{N} p^{\tau}_{s|i} = m \quad \forall i, s$$

τ: global round number
i: client index
s: model index
m: expected number of active clients
$d_{i,s}$: dataset size ratio
t: local epoch number
$\mathcal{A}_{\tau,s}$: set of active clients

MMFL Optimal Variance-Reduced Sampling

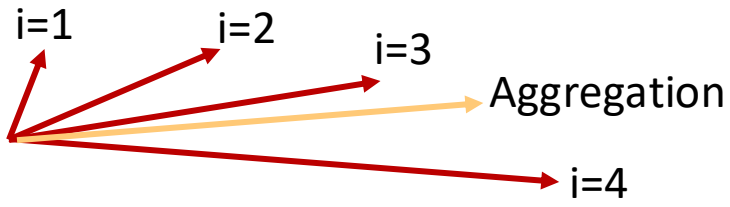
Closed-form solution of the problem

$$p_{s|i}^{\tau} = \begin{cases} (m-N+k)\frac{\|\tilde{U}_{i,s}^{\tau}\|}{\sum_{j=1}^{k} M_j^{\tau}} & \text{if } i = 1, 2, \cdots, k, \\ \frac{\|\tilde{U}_{i,s}^{\tau}\|}{M_i^{\tau}} & \text{if } i = k+1, \cdots, N. \end{cases} \quad (5)$$

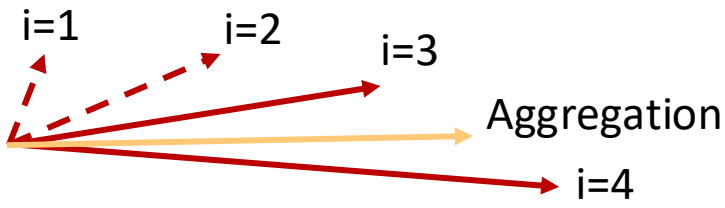
where $\|\tilde{U}_{i,s}^{\tau}\| = \|d_{i,s}U_{i,s}^{\tau}\|$ and $M_i^{\tau} = \sum_{s=1}^{S} \|\tilde{U}_{i,s}^{\tau}\|$. We reorder clients such that $M_i^{\tau} \leq M_{i+1}^{\tau}$ for all i, and k is the largest integer for which $0 < (m-N+k) \leq \frac{\sum_{j=1}^{k} M_j^{\tau}}{M_k^{\tau}}$.

τ: global round number
i: client index
s: model index
m: expected number of active clients
$d_{i,s}$: dataset size ratio
t: local epoch number
$\mathcal{A}_{\tau,s}$: set of active clients

Full participation (N=4)



Partial participation (active=2)



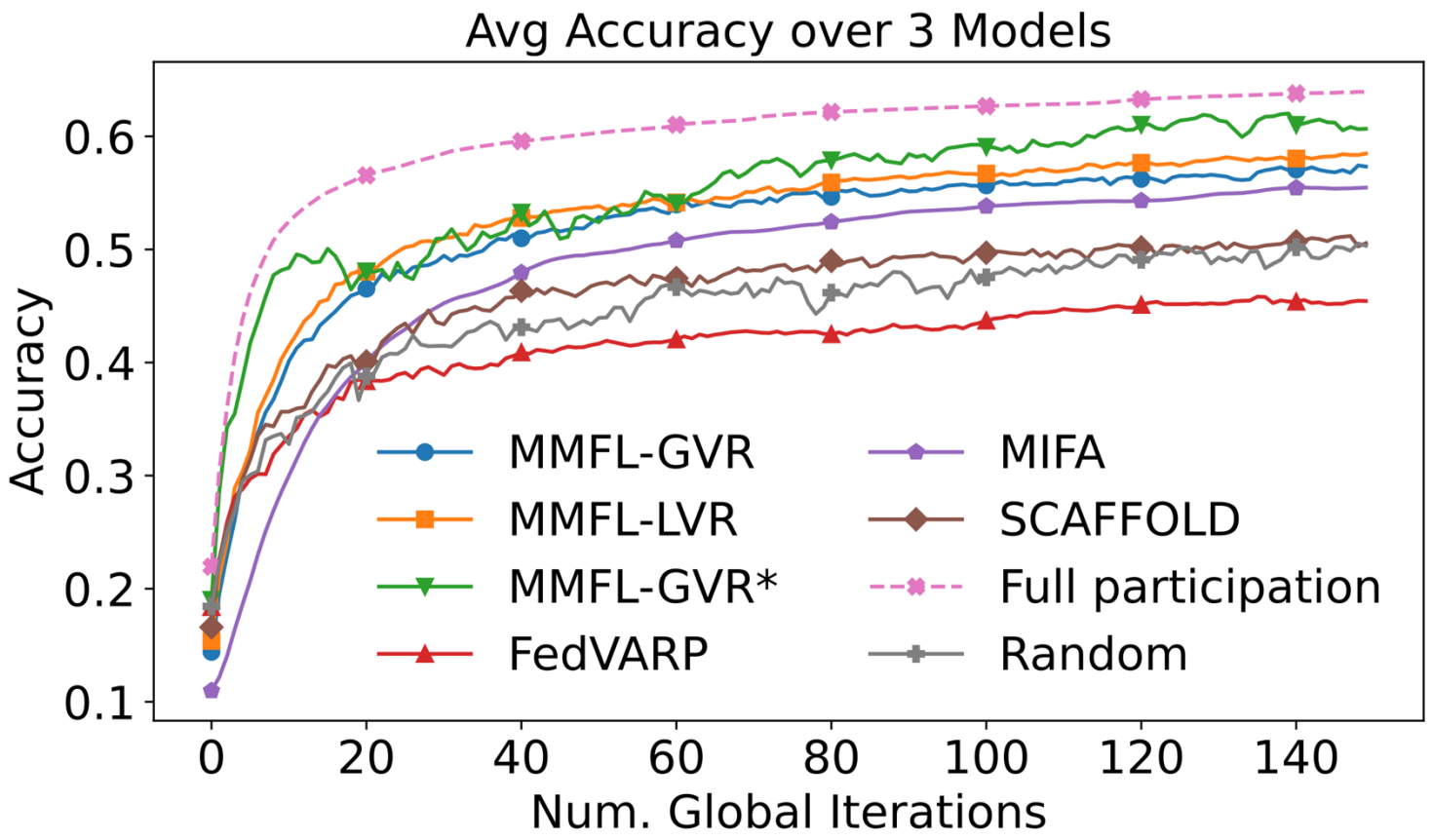
Proof: https://tinyurl.com/mmflos

Experiments

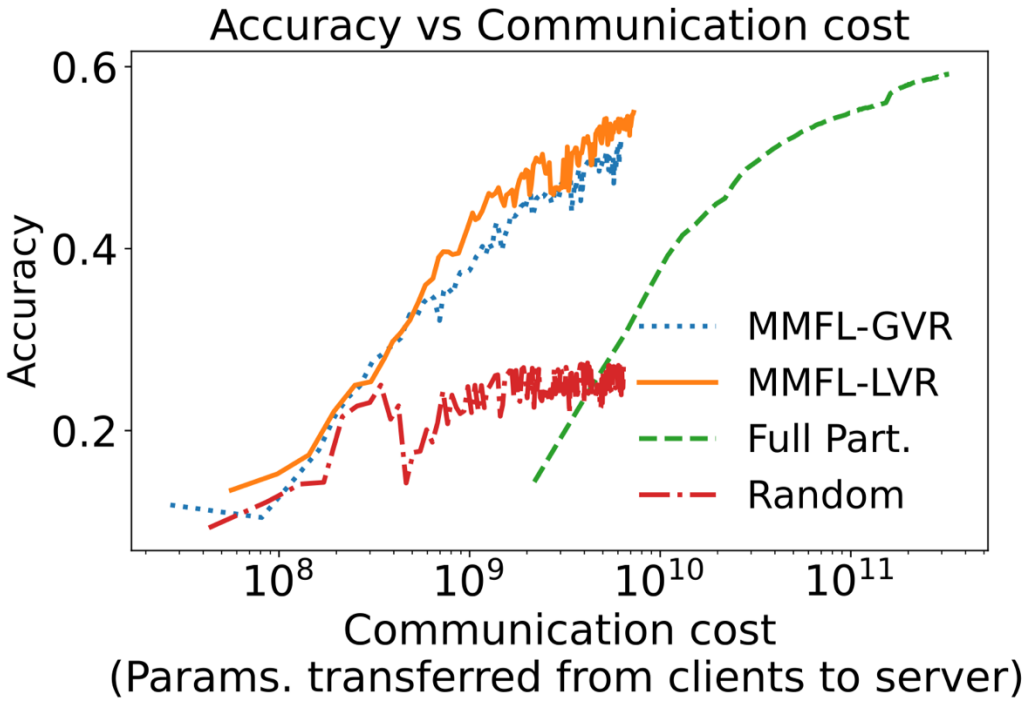
3 Models: all Fashion-MNIST.
N=120 clients
m=12 (active rate=0.1)
Each client: 30% labels.

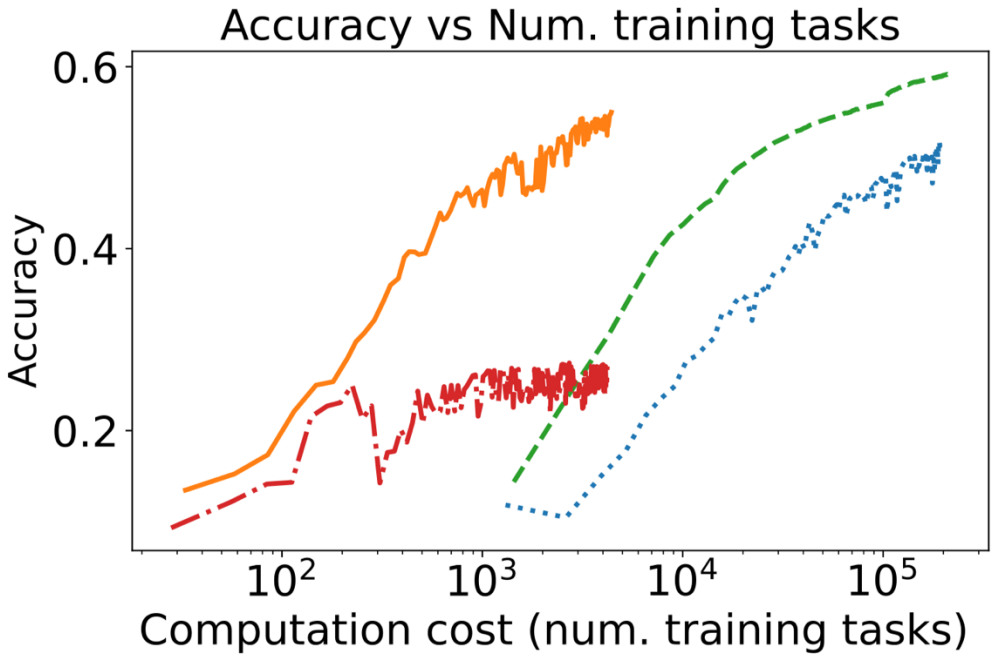
For each model: 10% high-data clients, 90% low-data clients. 10% clients hold 52.6% data of each task.

25% clients: $B_i = 3$
50% clients: $B_i = 2$
25% clients: $B_i = 1$



Experiments





Experiments

3 Models: all Fashion-MNIST.

5 Models: two Fashion-MNIST, one CIFAR-10, one EMNIST, one Shakespeare.

10% clients only have data for S-1 models.

TABLE I

FINAL AVERAGE MODEL ACCURACY RELATIVE TO THAT FROM FULL PARTICIPATION (THEORETICALLY THE BEST UNDER THE SAME LOCAL TRAINING SETTINGS).

Methods	3 tasks	5 tasks	Comm. Cost	Comp. Cost	Mem. Cost
FedVARP [30]	0.712±.14	0.690±.19	Low	Low	High
MIFA [31]	0.868±.18	0.835±.18	Low	Low	High
SCAFFOLD [32]	0.794±.14	0.650±.24	Low	Low	Low
Random	0.778±.19	0.749±.23	Low	Low	Low
Full Participation	1.000±.13	1.000±.14	High	High	Low
MMFL-GVR	0.893±.14	0.842±.20	Low	High	Low
MMFL-LVR	0.912±.15	0.849±.16	Low	Low	Low
MMFL-GVR*	**0.960**±.15	**0.869**±.18	Low	High	High