北京大学
PEKING UNIVERSITY

Distributed Machine Learning (Part II)

Kun Yuan

Center for Machine Learning Research @ Peking University

Methodologies to save communication

北京大学 PEKING UNIVERSITY

- Global average incurs $O(n)$ comm. overhead; proportional to network size n

  [Decentralized communication]

- Each node interacts with the server at every iteration; proportional to iteration numbers

  [Lazy communication]

- Each node sends a full model (or gradient) to the server; proportional to dimension d

  [Compressed communication]

PART 04

Lazy Communication

Distributed machine learning

A network of n nodes (devices such as GPUs) collaborate to solve the problem:

$$\min_{x\in\mathbb{R}^d} \quad f(x)=\frac{1}{n}\sum_{i=1}^{n} f_i(x), \quad \text{where} \quad f_i(x)=\mathbb{E}_{\xi_i\sim D_i} F(x;\xi_i).$$

- Each component $f_i:\mathbb{R}^d\to\mathbb{R}$ is local and private to node i
- Random variable ξ_i denotes local data that follows distribution D_i
- Each local distribution D_i is different; data heterogeneity exists

Parallel SGD (PSGD)

$$\min_{x\in\mathbb{R}^d} \quad f(x)=\frac{1}{n}\sum_{i=1}^{n} f_i(x), \quad \text{where} \quad f_i(x)=\mathbb{E}_{\xi_i\sim D_i} F(x;\xi_i).$$

PSGD

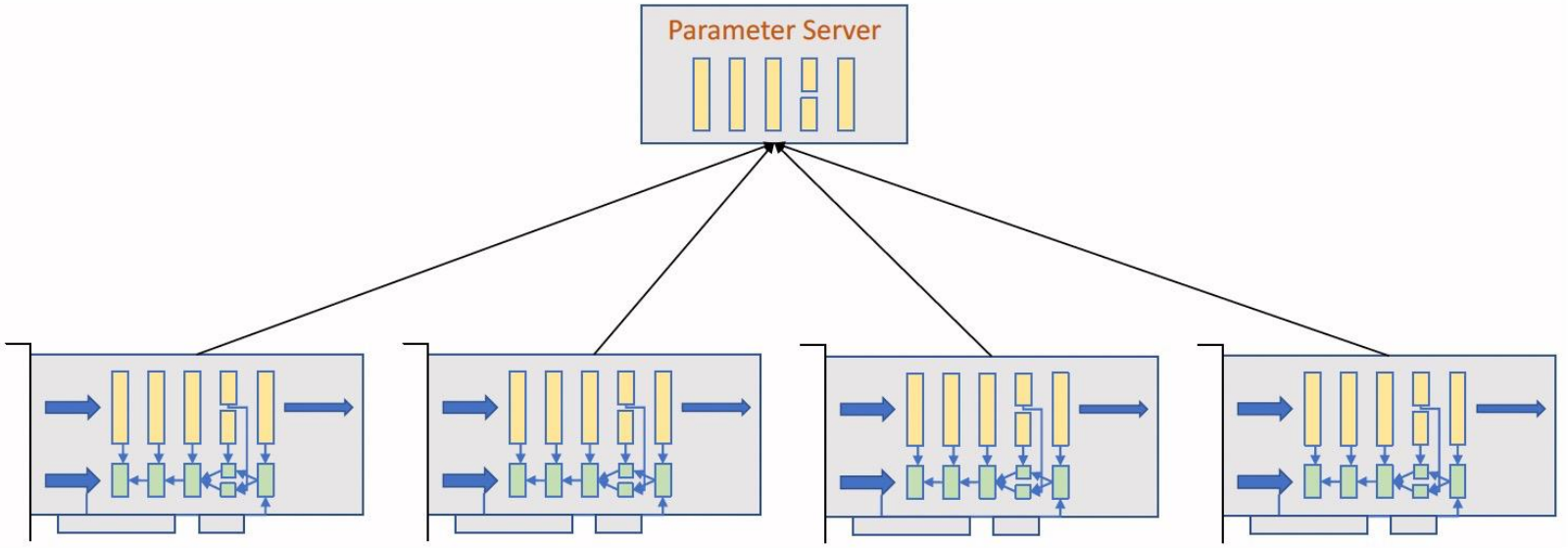
$$g_i^{(k)} = \nabla F(x^{(k)};\xi_i^{(k)}) \qquad \text{(Local compt.)}$$

$$x^{(k+1)} = x^{(k)} - \frac{\gamma}{n}\sum_{i=1}^{n} g_i^{(k)} \qquad \text{(Global comm.)}$$

- Each node i samples data $\xi_i^{(k)}$ and computes gradient $\nabla F(x^{(k)};\xi_i^{(k)})$
- All nodes synchronize (i.e. globally average) to update model x per iteration

Parallel SGD (PSGD): illustration

$$x^{(k+1)} = x^{(k)} - \frac{\gamma}{n}\sum_{i=1}^{n}\nabla F(x^{(k)};\xi_i^{(k)})$$



PSGD convergence rate

北京大学 PEKING UNIVERSITY

Assumption [PSGD assumption]

(1) Each $f_i(x)$ is L-smooth, i.e., $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$ for any x, y;

(2) Each stochastic gradient is unbiased and has bounded variance, i.e.,

$$\mathbb{E}[\nabla F(x; \xi_i)] = \nabla f_i(x), \quad \mathbb{E}\|\nabla F(x; \xi_i) - \nabla f_i(x)\|^2 \leq \sigma^2$$

Theorem [PSGD convergence]

Under the above assumptions and with proper γ, PSGD converges as follows

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

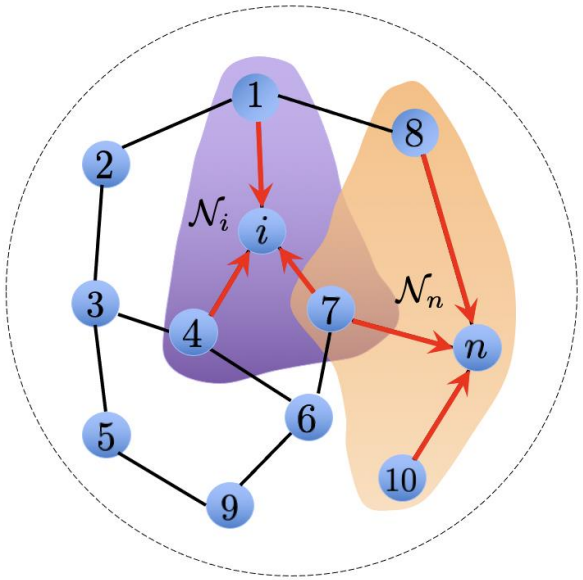
We consider more refined convergence rate to illustrate the benefits of local updates later

Decentralized SGD (DSGD)

- Decentralized SGD saves communication by **replacing global average to partial average**

$$x_i^{(k+\frac{1}{2})} = x_i^{(k)} - \gamma \nabla F(x_i^{(k)}; \xi_i^{(k)}) \quad \text{(Local update)}$$

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} x_j^{(k+\frac{1}{2})} \quad \text{(Partial averaging)}$$



- DSGD = local SGD update + partial averaging [1, 2]
- $\mathcal{N}_i$ is the set of neighbors at node i ; w_{ij} scales information from j to i
- Incurs $O(d_{\max})$ comm. overhead per iteration where $d_{\max} = \max_i\{|\mathcal{N}_i|\}$ is the graph maximum degree

---

[1] C. G. Lopes and A. H. Sayed, "Diffusion least-mean-squares over adaptive networks", ICASSP, 2007

[2] A. Nedich and A. Ozdaglar, " Distributed subgradient methods for multi-agent optimization", IEEE TAC, 2009

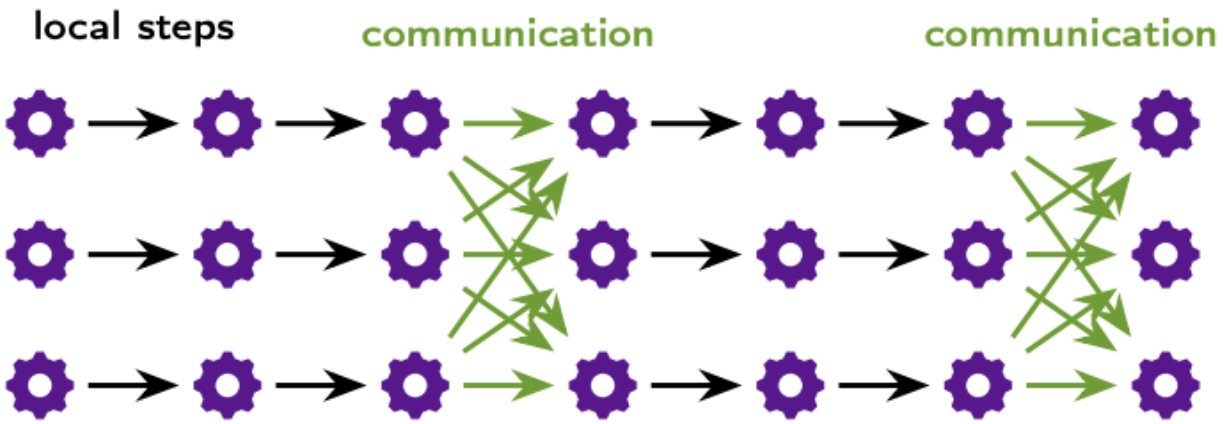
Local SGD

北京大学 PEKING UNIVERSITY

- Local SGD saves communication by **reducing communication frequency**

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

- Communicate once every τ iterations



[S. U. Stich, "Algorithms for Efficient Federated (and Decentralized) Learning", FL-ICML workshop, 2021]

Local SGD is also known as FedAvg

- Local SGD can be rewritten in a different format

$$
\begin{aligned}
&\text{For } r = 0, 1, \cdots, R-1 && \textbf{(R comm. rounds)}\\
&\quad x_i^{(r,0)} = x^{(r)}\\
&\quad \text{For } k = 0, 1, \cdots, K-1 && \textbf{(local update)}\\
&\qquad x_i^{(r,k+1)} = x_i^{(r,k+1)} - \gamma \nabla F(x_i^{(r,k)}, \xi_i^{(r,k)})\\
&\quad x^{(r+1)} = \frac{1}{n}\sum_{i=1}^{n} x_i^{(r,K)} && \textbf{(global comm.)}
\end{aligned}
$$

- Local SGD is also known as **FedAvg** [1,2]

[1] J. Konečný, H. B. McMahan, D. Ramage, P. Richtárik, “Federated Optimization: Distributed Machine Learning for On-Device Intelligence”, 2016

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data”, AISTATS 2017

Local SGD convergence

Assumption [Local SGD assumption]

(1) Each $f_i(x)$ is L-smooth, i.e., $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$ for any x, y;

(2) Each stochastic gradient is unbiased and has bounded variance, i.e.,

$$\mathbb{E}[\nabla F(x; \xi_i)] = \nabla f_i(x), \quad \mathbb{E}\|\nabla F(x; \xi_i) - \nabla f_i(x)\|^2 \leq \sigma^2$$

(3) Gradient dissimilarity can be upper bounded, i.e.,

$$\frac{1}{n}\sum_{i=1}^{n} \|\nabla f_i(x) - \nabla f(x)\|^2 \leq b^2$$

Same as DSGD assumptions

Local SGD convergence

北京大学 PEKING UNIVERSITY

Theorem [Local SGD convergence]

Under the above assumptions, by setting local learning rate and global learning rate properly, it holds that [1]

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

- The number of local updates K **improves** the dominant term in the convergence rate
- Parallel SGD with R communication rounds will converge at rate

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

- Local SGD converges faster than PSGD with the same communication budgets

[1] S. P. Karimireddy et. al., "Scaffold: stochastic controlled averaging for federated", ICML 2020

Local SGD v.s. parallel SGD

- It is intuitive that local SGD converges faster than PSGD; **local SGD runs K times more updates**

Local SGD

For $r = 0, 1, \cdots, R-1$ **(R comm. rounds)**

- $x_i^{(r,0)} = x^{(r)}$
- For $k = 0, 1, \cdots, K-1$ **(local update)**
  - $x_i^{(r,k+1)} = x_i^{(r,k+1)} - \gamma \nabla F(x_i^{(r,k)}, \xi_i^{(r,k)})$
- $x^{(r+1)} = \frac{1}{n} \sum_{i=1}^{n} x_i^{(r,K)}$ **(global comm.)**

Parallel SGD

For $r = 0, 1, \cdots, R-1$ **(R comm. rounds)**

- $\phi_i^{(r+1)} = x^{(r)} - \gamma \nabla F(x^{(r)}, \xi_i^{(r)})$
- $x^{(r+1)} = \frac{1}{n} \sum_{i=1}^{n} \phi_i^{(r+1)}$ **(global comm.)**

Local SGD v.s. mini-batch parallel SGD

- It is more fair to compare local SGD and parallel SGD with batch-size K

Local SGD

For $r = 0, 1, \cdots, R-1$ **(R comm. rounds)**

- $x_i^{(r,0)} = x^{(r)}$
- For $k = 0, 1, \cdots, K-1$ **(local update)**
  - $x_i^{(r,k+1)} = x_i^{(r,k+1)} - \gamma \nabla F(x_i^{(r,k)}, \xi_i^{(r,k)})$
- $x^{(r+1)} = \frac{1}{n} \sum_{i=1}^{n} x_i^{(r,K)}$ **(global comm.)**

Mini-batch Parallel SGD

For $r = 0, 1, \cdots, R-1$ **(R comm. rounds)**

- $\phi_i^{(r+1)} = x^{(r)} - \frac{\gamma}{K} \sum_{k=1}^{K} \nabla F(x^{(r)}, \xi_i^{(r,k)})$
- $x^{(r+1)} = \frac{1}{n} \sum_{i=1}^{n} \phi_i^{(r+1)}$ **(global comm.)**

Local SGD v.s. mini-batch parallel SGD

- Convergence comparison

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

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

- Local SGD is worse than mini-batch parallel SGD due to the **influence of data heterogeneity**

Local SGD v.s. mini-batch parallel SGD

北京大学 PEKING UNIVERSITY

- In real practices, local SGD outperforms mini-batch SGD when data heterogeneity is not severe, i.e., training deep neural network in data-centers

- **Large mini-batch causes severe generalization degradation; while local SGD does not**

Local SGD yields better generalization than huge batches

	Top-1 acc.	local gradients	communication
Mini-batch SGD ($n = 16$, $\tau = 128$)	92.5%	2048	-
Mini-batch SGD ($n = 16$, $\tau = 1024$)	76.3%	16384	$\div$ 8
Local-SGD ($n = 16$, $\tau = 8 \times 128$)	92.0%	16384	$\div$ 8

ResNet-20 on Cifar-10

[1] S. P. Karimireddy et. al., "Scaffold: stochastic controlled averaging for federated", ICML 2020

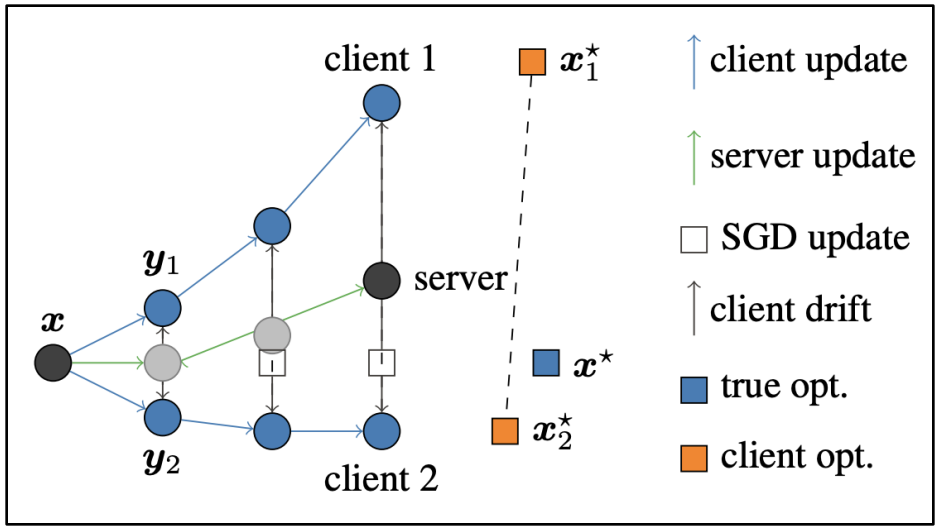
[2] T. Lin et. al., ”Don’t use large mini-batches, use local SGD", ICLR 2020

Data heterogeneity significantly influences local SGD

- In federated learning, local data distributions vary drastically; strong data heterogeneity exists

- Local SGD gets significantly degraded when strong data heterogeneity exists

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



Performance drops significantly

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]

Scaffold: an effective algorithm to remove data heterogeneity

- In local SGD, after K local updates, we have

$$x_i^{(r,K)} = x_i^{(r,0)} - \gamma g_i^{(r)}$$

where $g_i^{(r)} = \frac{\gamma}{K} \sum_{k=0}^{K-1} \nabla F(x_i^{(r,k)}, \xi_i^{(r,k)})$

For $r = 0, 1, \cdots, R-1$ **(R comm. rounds)**

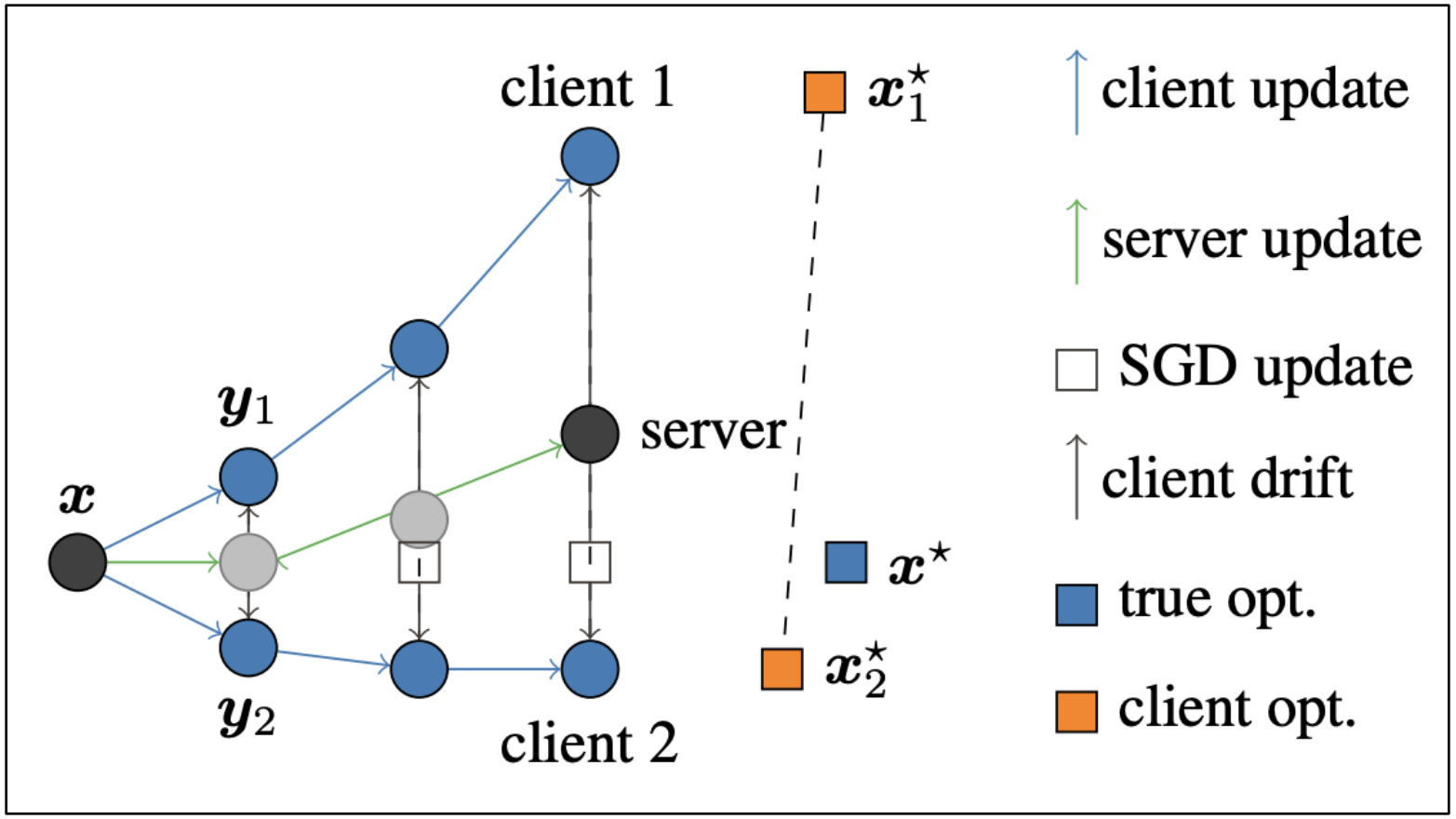
- $x_i^{(r,0)} = x^{(r)}$
- For $k = 0, 1, \cdots, K-1$ **(local update)**
  - $x_i^{(r,k+1)} = x_i^{(r,k+1)} - \gamma \nabla F(x_i^{(r,k)}, \xi_i^{(r,k)})$
- $x^{(r+1)} = \frac{1}{n} \sum_{i=1}^{n} x_i^{(r,K)}$ **(global comm.)**

- The root reason why local SGD suffers from data heterogeneity is

$g_i^{(r)}$ estimates $\nabla f_i(x^{(r)})$ well but is far away from the globally-averaged $(1/n) \sum_{j=1}^{n} \nabla f_j(x^{(r)})$

- $g_i^{(r)}$ is towards the local minimum, not the global minimum

Scaffold: an effective algorithm to remove data heterogeneity



Local update move towards local solutions, not the global solution

Scaffold: an effective algorithm to remove data heterogeneity

- To correct the gradient, Scaffold introduces the following approach

$$x_i^{(r,K)} = x_i^{(r,0)} - \gamma\left(g_i^{(r)} - c_i^{(r)} + c^{(r)}\right)$$

where $c_i^{(r)} = g_i^{(r-1)}$ and $c^{(r)} = (1/n)\sum_{j=1}^{n} c_j^{(r)}$

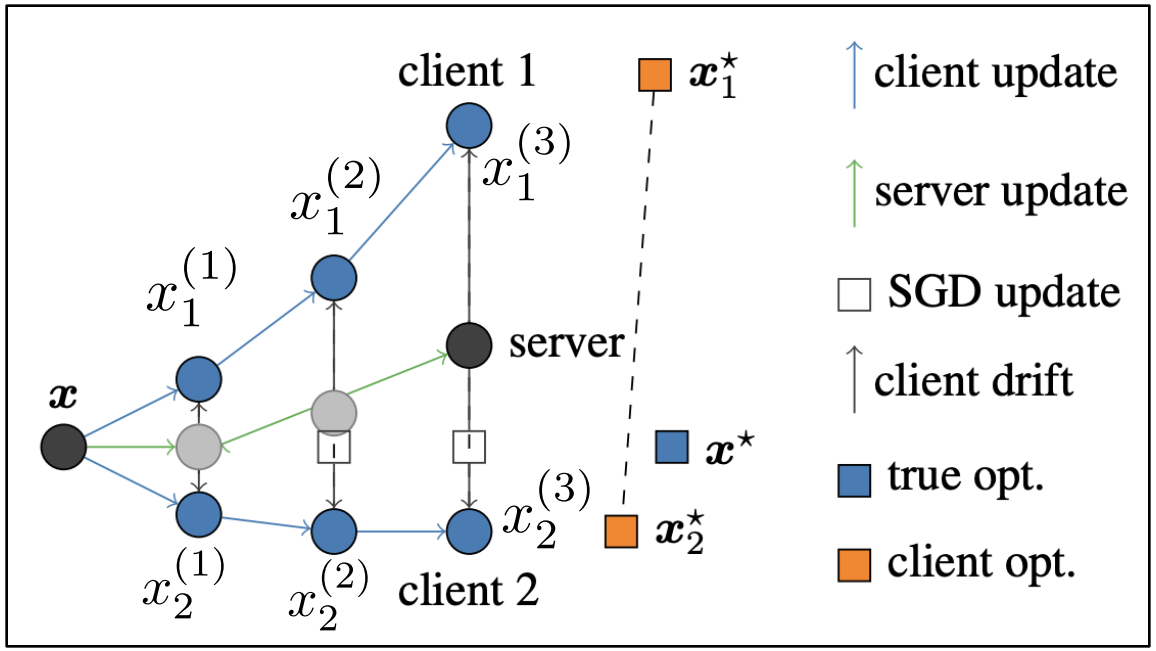
- Suppose $g_i^{(r)} \to \nabla f_i(x^{(r)})$, the corrected gradient $g_i^{(r)} - c_i^{(r)} + c^{(r)} \to \frac{1}{n}\sum_{i=1}^{n} f_i(x^{(r)})$

- Local update drives each local model towards the global minimum

S. P. Karimireddy et. al., "Scaffold: stochastic controlled averaging for federated", ICML 2020

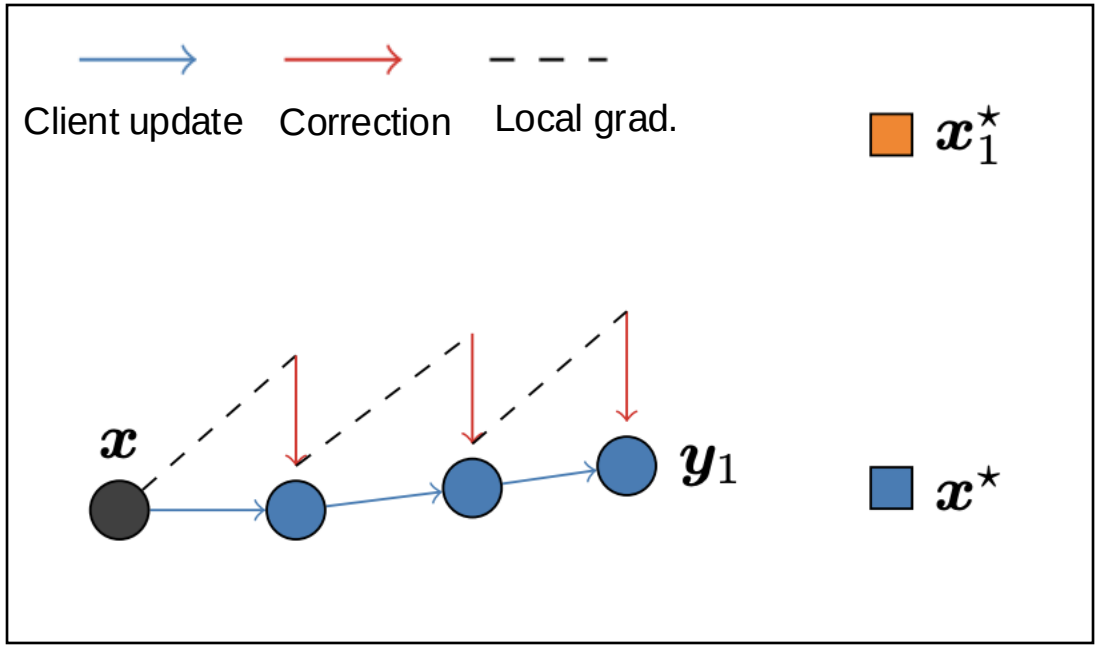
Local SGD v.s. Scaffold

Local SGD



$$x_i^\star \neq x^\star \Longrightarrow (1/n)\sum_{i=1}^{n} x_i^\star \neq x^\star$$

Scaffold



$$x_i^\star \to x^\star \Longrightarrow (1/n)\sum_{i=1}^{n} x_i^\star \to x^\star$$

Scaffold recursions

北京大学 PEKING UNIVERSITY

Scaffold algorithm

For $r = 0, 1, \cdots, R-1$ **(R comm. rounds)**

$x_i^{(r,0)} = x^{(r)} \quad c_i^{(r+1)} = 0$

For $k = 0, 1, \cdots, K-1$ **(K local updates)**

$x_i^{(r,k+1)} = x_i^{(r,k)} - \gamma\left(\nabla F(x_i^{(r,k)}, \xi_i^{(r,k)}) - c_i^{(r)} + c^{(r)}\right)$

$c_i^{(r+1)} \mathrel{+}= \nabla F(x_i^{(r,k)}, \xi_i^{(r,k)})/K$

$x^{(r+1)} = \frac{1}{n}\sum_{i=1}^{n} x_i^{(r,K)} \quad c^{(r+1)} = \frac{1}{n}\sum_{i=1}^{n} c_i^{(r)}$ **(global comm.)**

S. P. Karimireddy et. al., “Scaffold: stochastic controlled averaging for federated learning", ICML 2020

Scaffold convergence

Assumption [Scaffold assumptions]

(1) Each $f_i(x)$ is L-smooth, i.e., $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$ for any x, y;

(2) Each stochastic gradient is unbiased and has bounded variance, i.e.,

$$\mathbb{E}[\nabla F(x; \xi_i)] = \nabla f_i(x), \quad \mathbb{E}\|\nabla F(x; \xi_i) - \nabla f_i(x)\|^2 \leq \sigma^2$$

Theorem [Scaffold convergence]

Under the above assumptions and with proper γ, Scaffold converges as follows

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

Scaffold removes the influence of data heterogeneity

Convergence comparison

北京大学 PEKING UNIVERSITY

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

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

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

- We cannot show the superiority of Scaffold to mini-batch parallel SGD
- It is believed in the community that local updates **cannot** bring theoretical convergence benefits within the family of general non-convex and smooth problems [1]

[1] B. Woodworth et. al., "Is Local SGD Better than Minibatch SGD?", ICML 2020

Provable benefits under additional assumptions

- in the standard complexity classes, no benefit of local steps!
- however, on quadratic functions we hope to be better!
  - IID setting: (same Hessian, matching local minima)

algorithm	rounds
mini-batch SGD batch size τ	$\mathcal{O}\left(\frac{\sigma^2}{n\tau\mu\epsilon} + \frac{L}{\mu}\log\frac{1}{\epsilon}\right)$
local SGD τ local steps	$\mathcal{O}\left(\frac{\sigma^2}{n\tau\mu\epsilon} + \frac{L}{\tau\mu}\log\frac{1}{\epsilon}\right)$

  - non-IID setting: (same Hessian, different local minima)

SCAFFOLD τ local steps	$\tilde{\mathcal{O}}\left(\frac{\sigma^2}{n\tau\mu\epsilon} + \frac{L}{\tau\mu}\log\frac{1}{\epsilon}\right)$

Remarkable: benefit from parallelism and serial updates!

Scaffold performs well in non-iid scenarios

Communication rounds to reach 0.5 test accuracy for logistic regression on EMNIST

		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

S. P. Karimireddy et. al., "Scaffold: stochastic controlled averaging for federated", ICML 2020

Client sampling

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**)

- Clients are not stable; they are not available when out of power or signal

This implies that not all clients can participate in the training or learning process

Client sampling: sample s <= n clients per iteration during the training process

Does not change the algorithm and analysis too much

Client sampling

- Client sampling does not change the algorithm and analysis too much

Scaffold with client sampling

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

- When n = s, the above results reduce to Scaffold with full-client sampling

- Fewer sampled clients lead to slower convergence performance

- The above results hold for **uniform** client sampling; Arbitrary client sampling is less explored

S. P. Karimireddy et. al., "Scaffold: stochastic controlled averaging for federated", ICML 2020

Momentum benefits local SGD simply and provably

- Momentum is widely-used in federated learning and local SGD
- Its theoretical benefits to federated learning algorithms are not well-understood

Algorithm 1 FEDAVG-M: FEDAVG with momentum

Require: initial model x^0 and gradient estimate g^0, local learning rate η, global learning rate γ, momentum β

for $r = 0, \cdots, R-1$ **do**
  for each client $i \in \{1, \ldots, N\}$ in parallel **do**
    Initialize local model $x_i^{r,0} = x^r$
    for $k = 0, \cdots, K-1$ **do**
      Compute $g_i^{r,k} = \beta \nabla F(x_i^{r,k}; \xi_i^{r,k}) + (1-\beta) g^r$ ▷ $\beta = 1$ implies FEDAVG
      Update local model $x_i^{r,k+1} = x_i^{r,k} - \eta g_i^{r,k}$
    end for
  end for
  Aggregate local updates $g^{r+1} = \frac{1}{\eta N K} \sum_{i=1}^{N} \left(x^r - x_i^{r,K} \right)$
  Update global model $x^{r+1} = x^r - \gamma g^{r+1}$
end for

$$\beta \to 0 \quad \Longrightarrow \quad g_i^{(r,k)} \to g^{(r)}$$

Beta tunes how local gradient changes towards global gradient

Z. Cheng, et. al., "Momentum Benefits NON-IID Federated Learning Simply and Provably", ICLR 2024

Momentum benefits local SGD simply and provably

北京大学 PEKING UNIVERSITY

- For local SGD, momentum provably removes data heterogeneity; no need for Scaffold

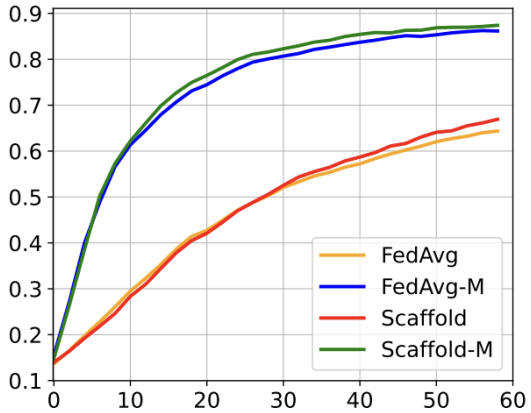
Algorithm	Convergence Rate $\mathbb{E}[\|\nabla f(\hat{x})\|^2] \lesssim$	Assumptions
FEDAVG		
(Yu et al., 2019b)	$\left(\frac{L\Delta\sigma^2}{NKR}\right)^{1/2} + \left(\frac{L\Delta G}{R}\right)^{2/3} + \frac{L\Delta}{R}$	Bounded grad.
(Koloskova et al., 2020)	$\left(\frac{L\Delta\sigma^2}{NR}\right)^{1/2} + \left(\frac{L\Delta K\zeta}{R}\right)^{2/3} + \frac{L\Delta K}{R}$	Bounded hetero.
(Karimireddy et al., 2020b)	$\left(\frac{L\Delta\sigma^2}{NKR}\right)^{1/2} + \left(\frac{L\Delta\zeta}{R}\right)^{2/3} + \frac{L\Delta}{R}$	Bounded hetero.
(Yang et al., 2021)	$\left(\frac{L\Delta\sigma^2}{NKR}\right)^{1/2} + \frac{L\Delta}{R}$	Bounded hetero.[1]
FEDCM[2] (Xu et al., 2021b)	$\left(\frac{L\Delta(\sigma^2+NKG^2)}{NKR}\right)^{1/2} + \left(\frac{L\Delta(\sigma/\sqrt{K}+G)}{R}\right)^{2/3}$	Bounded grad. Bounded hetero.
LED (Alghunaim, 2023)	$\left(\frac{L\Delta\sigma^2}{NKR}\right)^{1/2} + \left(\frac{L\Delta\sigma}{\sqrt{K}R}\right)^{2/3} + \frac{L\Delta}{R}$	–
VRL-SGD[2] (Liang et al., 2019)	$\left(\frac{L\Delta\sigma^2}{NKR}\right)^{1/2} + \left(\frac{L\Delta\sigma}{\sqrt{K}R}\right)^{2/3} + \frac{L\Delta}{R}$	–
FEDAVG-M (Thm. 1)	$\left(\frac{L\Delta\sigma^2}{NKR}\right)^{1/2} + \frac{L\Delta}{R}$ **No data heterogeneity**	

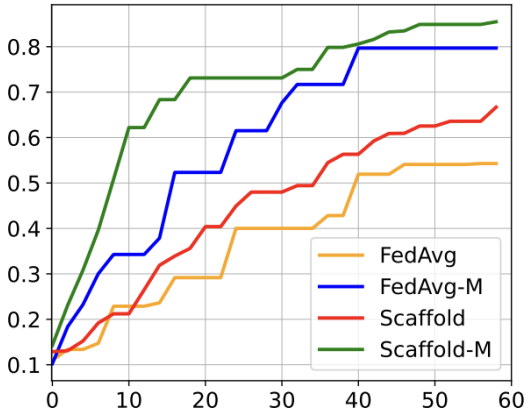
Z. Cheng, et. al., “Momentum Benefits NON-IID Federated Learning Simply and Provably”, ICLR 2024

Momentum benefits local SGD simply and provably

- For Scaffold, momentum improves its convergence rate
- Can achieve up to $n^{2/9}$ times speedup compared to vanilla Scaffold; state-of-the-art rate

Algorithm	Convergence Rate $\mathbb{E}[\|\nabla f(\hat{x})\|^2] \lesssim$	Assumptions
SCAFFOLD (Karimireddy et al., 2020b)	$\left(\frac{L\Delta\sigma^2}{SKR}\right)^{1/2} + \frac{L\Delta}{R}\left(\frac{N}{S}\right)^{2/3}$	–
SCAFFOLD-M (Thm. 3)	$\left(\frac{L\Delta\sigma^2}{SKR}\right)^{1/2} + \frac{L\Delta}{R}\left(1+\frac{N^{2/3}}{S}\right)$	–



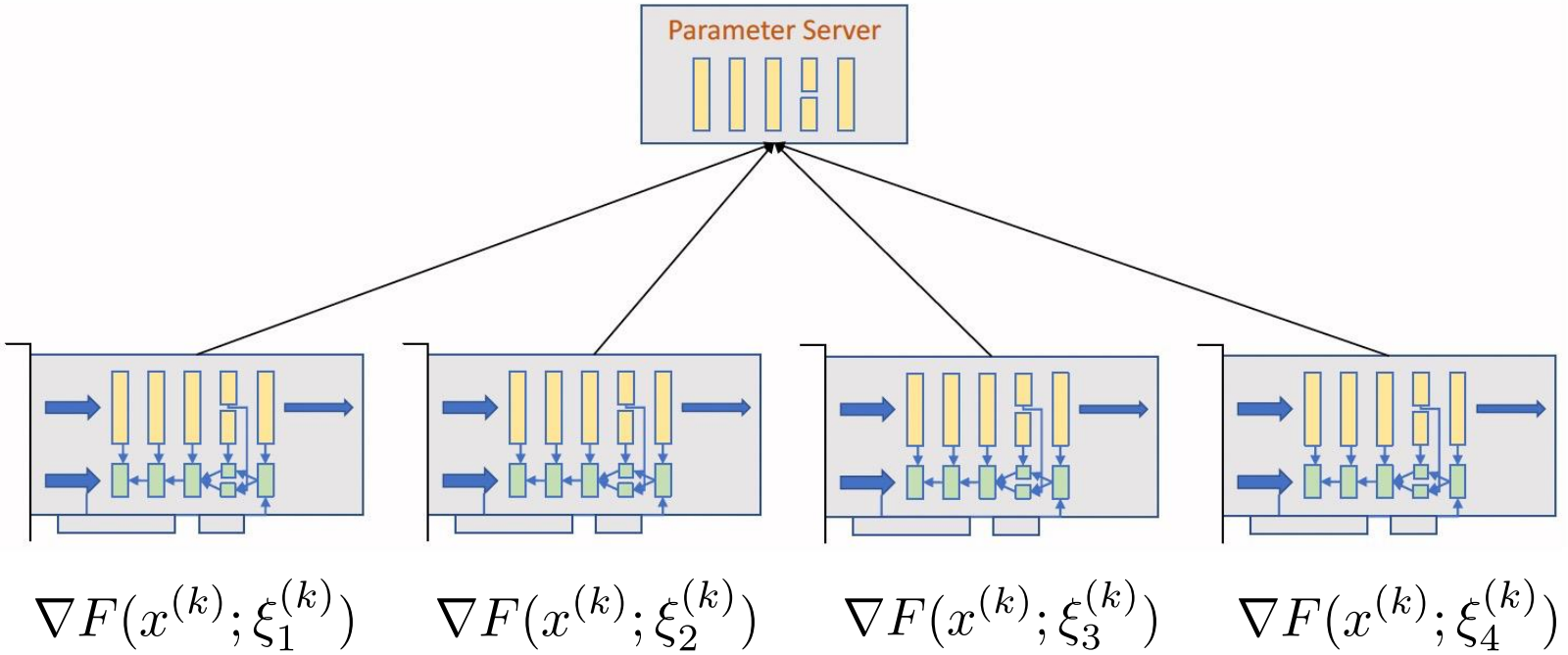


PART 05

Compressed Communication

Parallel SGD (PSGD)

$$x^{(k+1)} = x^{(k)} - \frac{\gamma}{n}\sum_{i=1}^{n}\nabla F(x^{(k)};\xi_i^{(k)})$$

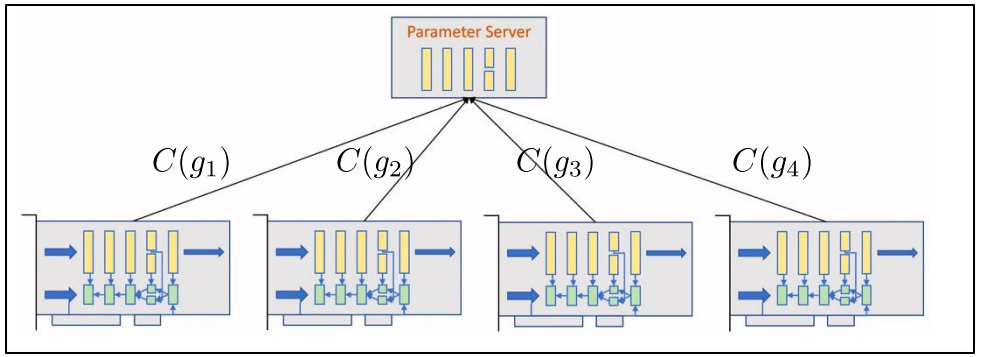


Ccommunication compression

- A basic (but not state-of-the-art) algorithm is QSGD [Alistarh et. al., 2017]

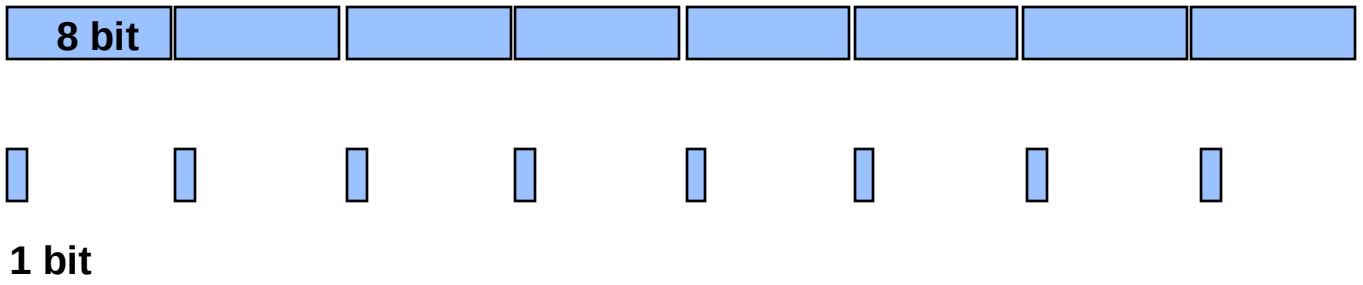
$$g_i^{(k)} = \nabla F(x_i^{(k)}; \xi_i^{(k)})$$

$$x^{(k+1)} = x^{(k)} - \frac{\gamma}{n} \sum_{j=1}^{n} C(g_j^{(k)})$$



- $C(\cdot)$ is a compressor. It can quantize or sparsify the full gradient

Quantization

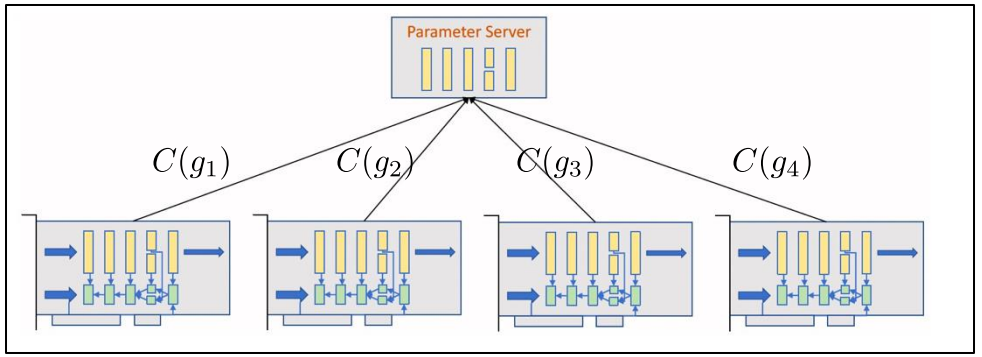


Ccommunication compression

- A basic (but not state-of-the-art) algorithm is QSGD [Alistarh et. al., 2017]

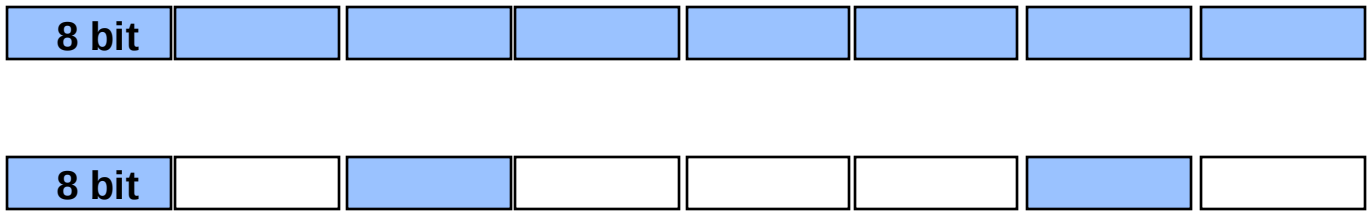
$$g_i^{(k)} = \nabla F(x_i^{(k)}; \xi_i^{(k)})$$

$$x_i^{(k+1)} = x_i^{(k)} - \frac{\gamma}{n} \sum_{j=1}^{n} C(g_j^{(k)})$$



- $C(\cdot)$ is a compressor. It can quantize or sparsify the full gradient

Sparsification



Unbiased compressor

- Compressor class. Most compressors in literature are either **unbiased** or **contractive**

- We let $\mathcal{U}_\omega$ denote the set of unbiased compressors satisfying Assumption 3

Assumption (Unbiased compressor) The compression operator $C : \mathbb{R}^d \to \mathbb{R}^d$ satisfies

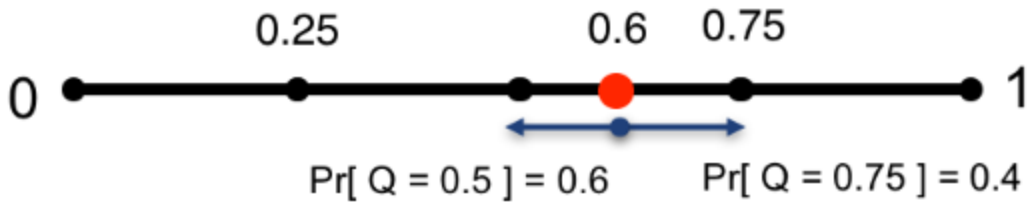
$$\mathbb{E}[C(x)] = x, \quad \mathbb{E}[\|C(x) - x\|^2] \leq \omega \|x\|^2, \quad \forall\, x \in \mathbb{R}^d$$

for constant $\omega \geq 0$, where the expectation is taken over the randomness of the compression operator C.

- Identity operator I (i.e. no compression) is an unbiased compressor with $\omega = 0$.

Example: random quantization

- Random quantization with 5 levels:



E[Q] = 0.6 × 0.5 + 0.4 * 0.75 = 0.6

Unbiased: $$\mathbb{E}[Q(x)] = \frac{Q_+(x) - x}{Q_+(x) - Q_-(x)} \cdot Q_-(x) + \frac{x - Q_-(x)}{Q_+(x) - Q_-(x)} \cdot Q_+(x) = x$$

Distributed compressed SGD

北京大学 PEKING UNIVERSITY

- Compressed algorithm is easy to develop with unbiased compressors

$$g_i^{(k)} = \nabla F(x_i^{(k)}; \xi_i^{(k)})$$
$$x_i^{(k+1)} = x^{(k)} - \frac{\gamma}{n}\sum_{j=1}^{n} C(g_j^{(k)})$$

- For simplicity, we assume $g_i^{(k)} = \nabla f_i(x^{(k)})$

$$\mathbb{E}[f(x^{(k+1)})] \le \mathbb{E}[f(x^{(k)})] + \mathbb{E}[\langle \nabla f(x^{(k)}), x^{(k+1)} - x^{(k)} \rangle] + \frac{L}{2}\mathbb{E}\|x^{(k+1)} - x^{(k)}\|^2 \quad \textbf{(L-smooth)}$$
$$\le \mathbb{E}[f(x^{(k)})] - \gamma\mathbb{E}[\langle \nabla f(x^{(k)}), \frac{1}{n}\sum_{i=1}^{n} C\big(\nabla f_i(x^{(k)})\big)\rangle] + \frac{\gamma^2 L}{2}\mathbb{E}\|\frac{1}{n}\sum_{i=1}^{n} C\big(\nabla f_i(x^{(k)})\big)\|^2$$
$$\le \mathbb{E}[f(x^{(k)})] - \gamma\mathbb{E}\|\nabla f(x^{(k)})\|^2 + \frac{\gamma^2 L}{2n^2}\sum_{i=1}^{n}\mathbb{E}\|C\big(\nabla f_i(x^{(k)})\big)\|^2 \quad \textbf{(Unbiased compressor)}$$

Distributed compressed SGD: convergence sketch

$$\mathbb{E}[f(x^{(k+1)})] \le \mathbb{E}[f(x^{(k)})] + \mathbb{E}[\langle \nabla f(x^{(k)}), x^{(k+1)} - x^{(k)} \rangle] + \frac{L}{2}\mathbb{E}\|x^{(k+1)} - x^{(k)}\|^2 \quad \textbf{(L-smooth)}$$

$$\le \mathbb{E}[f(x^{(k)})] - \gamma \mathbb{E}[\langle \nabla f(x^{(k)}), \frac{1}{n}\sum_{i=1}^{n} C\big(\nabla f_i(x^{(k)})\big) \rangle] + \frac{\gamma^2 L}{2}\mathbb{E}\|\frac{1}{n}\sum_{i=1}^{n} C\big(\nabla f_i(x^{(k)})\big)\|^2$$

$$\le \mathbb{E}[f(x^{(k)})] - \gamma \mathbb{E}\|\nabla f(x^{(k)})\|^2 + \frac{\gamma^2 L}{2n^2}\sum_{i=1}^{n}\mathbb{E}\|C\big(\nabla f_i(x^{(k)})\big)\|^2 \quad \textbf{(Unbiased compressor)}$$

$$= \mathbb{E}[f(x^{(k)})] - \gamma \mathbb{E}\|\nabla f(x^{(k)})\|^2 + \frac{\gamma^2 L}{2n^2}\sum_{i=1}^{n}\mathbb{E}\|C\big(\nabla f_i(x^{(k)})\big) - \nabla f_i(x^{(k)}) + \nabla f_i(x^{(k)})\|^2$$

$$\le \mathbb{E}[f(x^{(k)})] - \gamma \mathbb{E}\|\nabla f(x^{(k)})\|^2 + \frac{\gamma^2 L}{n^2}\sum_{i=1}^{n}\mathbb{E}\|C\big(\nabla f_i(x^{(k)})\big) - \nabla f_i(x^{(k)})\|^2 + \frac{\gamma^2 L}{n^2}\sum_{i=1}^{n}\mathbb{E}\|\nabla f_i(x^{(k)})\|^2$$

$$\le \mathbb{E}[f(x^{(k)})] - \gamma \mathbb{E}\|\nabla f(x^{(k)})\|^2 + \frac{(1+\omega)\gamma^2 L}{n^2}\sum_{i=1}^{n}\mathbb{E}\|\nabla f_i(x^{(k)})\|^2 \quad \textbf{(Bounded compression)}$$

$$= \mathbb{E}[f(x^{(k)})] - \gamma \mathbb{E}\|\nabla f(x^{(k)})\|^2 + \frac{(1+\omega)\gamma^2 L}{n^2}\sum_{i=1}^{n}\mathbb{E}\|\nabla f_i(x^{(k)}) - \nabla f(x^{(k)}) + \nabla f(x^{(k)})\|^2$$

$$\le \mathbb{E}[f(x^{(k)})] - \left(\gamma - \frac{2(1+\omega)\gamma^2 L}{n^2}\right)\mathbb{E}\|\nabla f(x^{(k)})\|^2 + \frac{2(1+\omega)\gamma^2 L b^2}{n^2} \quad \textbf{(Bounded data hetero.)}$$

Distributed compressed SGD: convergence property

Assumption [Compressed SGD assumption]

(1) Each $f_i(x)$ is L-smooth, i.e., $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$ for any x, y;

(2) Each stochastic gradient is unbiased and has bounded variance, i.e.,

$$\mathbb{E}[\nabla F(x;\xi_i)] = \nabla f_i(x), \quad \mathbb{E}\|\nabla F(x;\xi_i) - \nabla f_i(x)\|^2 \leq \sigma^2$$

(3) Gradient dissimilarity can be upper bounded, i.e.,

$$\frac{1}{n}\sum_{i=1}^{n}\|\nabla f_i(x) - \nabla f(x)\|^2 \leq b^2$$

Same as DSGD and Local SGD assumptions

Distributed compressed SGD: convergence property

Theorem [Compressed SGD convergence]

Under the above assumptions and with proper γ, compressed SGD with unbiased compressors will converge as follows

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\|\nabla f(x^{(k)})\|^2 = \mathcal{O}\left(\sqrt{\frac{(1+\omega)\sigma^2+\omega b^2}{nK}}+\frac{1+\omega/n}{K}\right)$$

Convergence performance can be further improved by DIANA [1] and MARINA [2]

[1] K. Mishchenko et. al., “Distributed Learning with Compressed Gradient Differences”, 2019

[2] E. Gorbunov, et. al., "MARINA: Faster Non-Convex Distributed Learning with Compression", 2021

Contractive compressor

- We let $\mathcal{C}_\delta$ denote the set of unbiased compressors satisfying Assumption 4

Assumption (Contractive compressor) The compression operator $C : \mathbb{R}^d \to \mathbb{R}^d$ satisfies

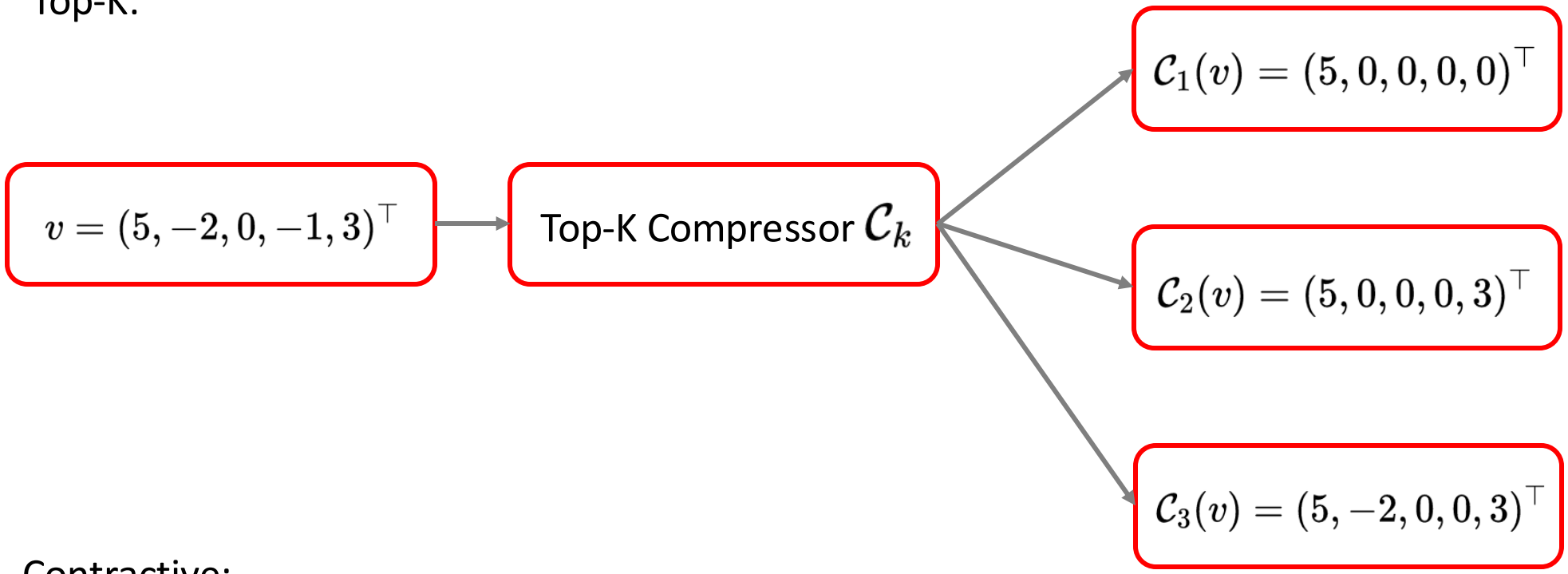
$$\mathbb{E}[\|C(x) - x\|^2] \leq (1 - \delta)\|x\|^2, \quad \forall\, x \in \mathbb{R}^d$$

for constant $\delta \in (0, 1]$, where the expectation is taken over the randomness of the compression operator C.

- Identity operator I (i.e. no compression) is a contractive compressor with $\delta = 1$.

Example: Top-K compressor

- Top-K:



Contractive:

$$\|\mathcal{C}_k(v) - v\|^2 = \|v\|^2 - \|\mathcal{C}_k(v)\|^2 \leq (1 - k/d)\|v\|^2, \text{ i.e., } \delta = k/d \in (0, 1]$$

Distributed compressed SGD may diverge with contractive compressor

- DC-SGD may diverge with contractive compressors

$$x^{(k+1)} = x^{(k)} - \frac{\gamma}{n}\sum_{i=1}^{n} C\left(\nabla f_i(x^{(k)})\right)$$

- Check the divergent examples in [1]

- The root reason is that contractive compressor introduces non-vanishing distortions

$$\mathbb{E}\|C\left(\nabla f_i(x^{(k)})\right) - \nabla f_i(x^{(k)})\|^2 \leq (1-\delta)\|\nabla f_i(x^{(k)})\|^2$$

- Since $\nabla f_i(x^{(k)})$ will not converge to 0, $C\left(\nabla f_i(x^{(k)})\right)$ will not converge to $\nabla f_i(x^{(k)})$

[1] A. Beznosikov et.al., “On Biased Compression for Distributed Learning”, JMLR 2023

Error feedback: corrects the distortion

- A smart idea: we do not compress the gradient. Instead, we compress the **gradient difference**

$$x^{(k+1)} = x^{(k)} - \frac{\gamma}{n}\sum_{i=1}^{n} g_i^{(k)}$$

$$g_i^{(k+1)} = g_i^{(k)} + C\left(\nabla f_i(x^{(k+1)}) - g_i^{(k)}\right)$$

- The compression distortion gets diminished, and $g_i^{(k)} \to \nabla f_i(x^{(k)})$

$$\begin{aligned}\mathbb{E}\|g_i^{(k+1)} - \nabla f_i(x^{(k+1)})\|^2 &= \mathbb{E}\|C\left(\nabla f_i(x^{(k+1)}) - g_i^{(k)}\right) - \left(\nabla f_i(x^{(k+1)}) - g_i^{(k)}\right)\|^2 \\ &\leq (1-\delta)\mathbb{E}\|g_i^{(k)} - \nabla f_i(x^{(k+1)})\|^2\end{aligned}$$

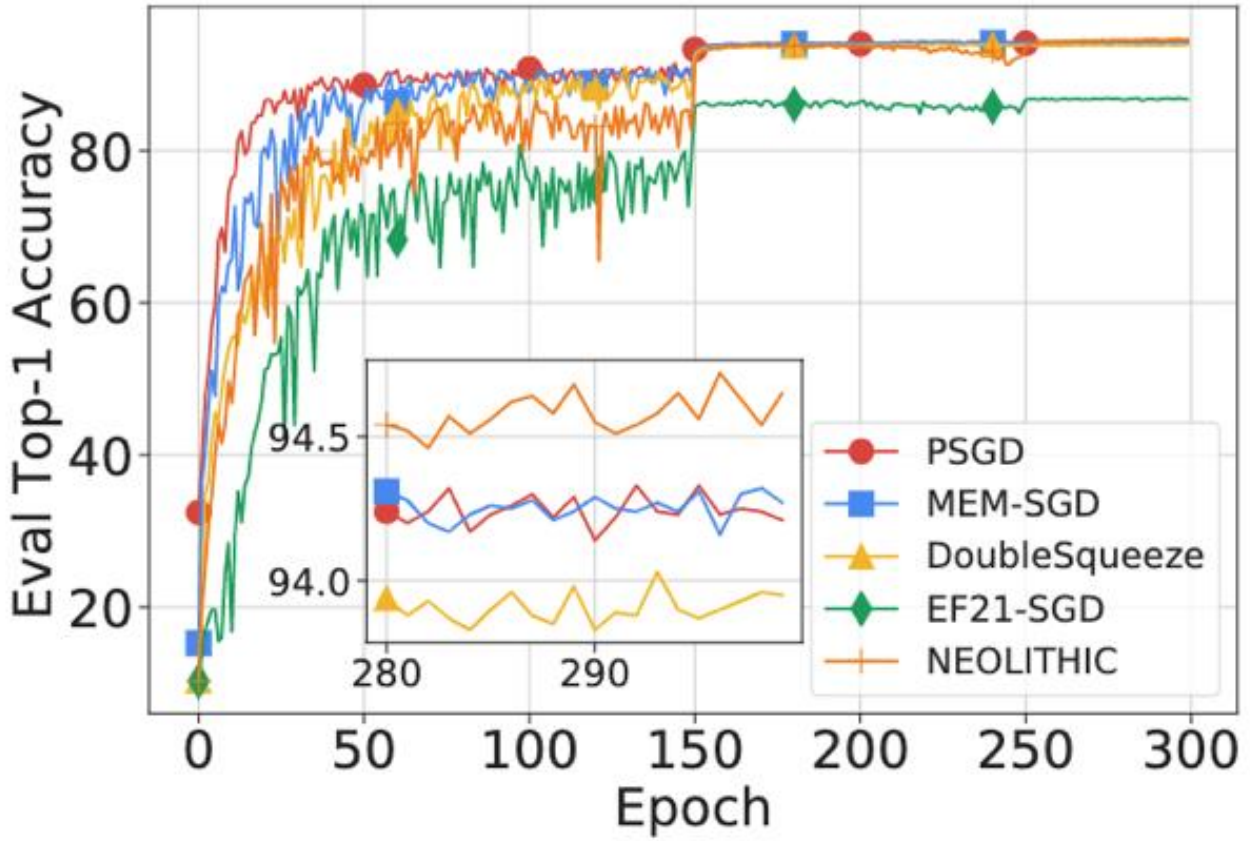
- If $\nabla f_i(x^{(k)}) \to \nabla f_i(x^\star)$, we will have $g_i^{(k)} \to \nabla f_i(x^{(k)})$ and hence distortion gets diminished

- This algorithm is named as Error Feedback [1]

[1] P. Richtárik, et. al., "EF21: A New, Simpler, Theoretically Better, and Practically Faster Error Feedback", NeurIPS 2021

Deep learning experiments

- 8 workers, **1% compression ratio** (top-k compressors), minibatch=128, R=2, ResNet18/ResNet20



Deep learning experiments

北京大学 PEKING UNIVERSITY

Cifar10; 8 workers; **5% compression ratio**

METHODS	RESNET18	RESNET20
PSGD	93.99 ± 0.52	91.62 ± 0.13
MEM-SGD	94.35 ± 0.01	91.27 ± 0.08
DOUBLE-SQUEEZE	94.11 ± 0.14	90.73 ± 0.02
EF21-SGD	87.37 ± 0.49	65.82 ± 4.86
NEOLITHIC	$\mathbf{94.63 \pm 0.09}$	$\mathbf{91.43 \pm 0.10}$

北京大学
PEKING UNIVERSITY

The end of the story? Not Yet !

Compressed algorithms converge slower

- The optimal iteration complexity of distributed algorithms with unbiased compression

(With compression) $$\Theta\left(\frac{\sigma^2}{\mu n\epsilon}+(1+\omega)\sqrt{\frac{L}{\mu}}\ln\left(\frac{1}{\epsilon}\right)\right)$$

- In contrast, the accelerated distributed SGD without any compression achieves optimal complexity

(Without compression) $$\Theta\left(\frac{\sigma^2}{\mu n\epsilon}+\sqrt{\frac{L}{\mu}}\ln\left(\frac{1}{\epsilon}\right)\right)$$

- Algorithm with compression needs **more rounds of communication** than without compression

- But compression **saves communication per round**

北京大学
PEKING UNIVERSITY

Can compression save total communication?

Optimal total communication cost of compressed algorithms

北京大学 PEKING UNIVERSITY

- Recall the definition of unbiased compression

Assump. 3 (Unbiased compressor) The compression operator $C : \mathbb{R}^d \to \mathbb{R}^d$ satisfies

$$\mathbb{E}[C(x)] = x, \quad \mathbb{E}[\|C(x) - x\|^2] \leq \omega \|x\|^2, \quad \forall\, x \in \mathbb{R}^d$$

for constant $\omega \geq 0$, where the expectation is taken over the randomness of the compression operator C.

- It typically holds that $\omega \gg 1$

Optimal total communication cost of compressed algorithms

北京大学 PEKING UNIVERSITY

- For compressed algorithm with unbiased compression, the following proposition establishes the lower bound of the **per-iteration communication cost** [1]

Proposition 1

Let d be the deminsion of the input variable x. For any compressor C satisfying Assumption 1, the per-round communciation cost of $C(x)$ is lower bounded by

$$\Omega\left(\frac{d}{1+\omega}\right)$$

This lower bound is **tight** and can be achieved by random-K compressor

- The more aggressive the compressor is, the more per-round communication cost it saves

Y. He, et. al., "Unbiased Compression Saves Communication in Distributed Optimization: When and How Much?", NeurIPS 2023

Optimal total communication cost of compressed algorithms

北京大学 PEKING UNIVERSITY

- The **optimal total communication cost (TCC)** of algorithms with communication compression

Per-round comm. number of comm. rounds

$$\mathrm{TCC} = \Theta\left(\frac{d}{1+\omega}\cdot\left[\frac{\sigma^2}{\mu n\epsilon} + (1+\omega)\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\epsilon}\right)\right]\right)$$

$$= \Theta\left(\frac{d\sigma^2}{(1+\omega)\mu n\epsilon} + d\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\epsilon}\right)\right)$$

Optimal total communication cost of non-compressed algorithms

- The **optimal total communication cost (TCC)** of algorithms without communication compression

Per-round comm. number of comm. rounds

$$\mathrm{TCC} = \Theta\left(d \cdot \left[\frac{\sigma^2}{\mu n \epsilon} + \sqrt{\frac{L}{\mu}} \log\left(\frac{1}{\epsilon}\right)\right]\right)$$

$$= \Theta\left(\frac{\sigma^2 d}{\mu n \epsilon} + d\sqrt{\frac{L}{\mu}} \log\left(\frac{1}{\epsilon}\right)\right)$$

Compression v.s. non-compression

- Recall the optimal total communication cost of algorithms with compression and non-compression

Unbiased compression

$$\Theta\left(\frac{d\sigma^2}{(1+\omega)\mu n\epsilon}+d\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\epsilon}\right)\right)$$

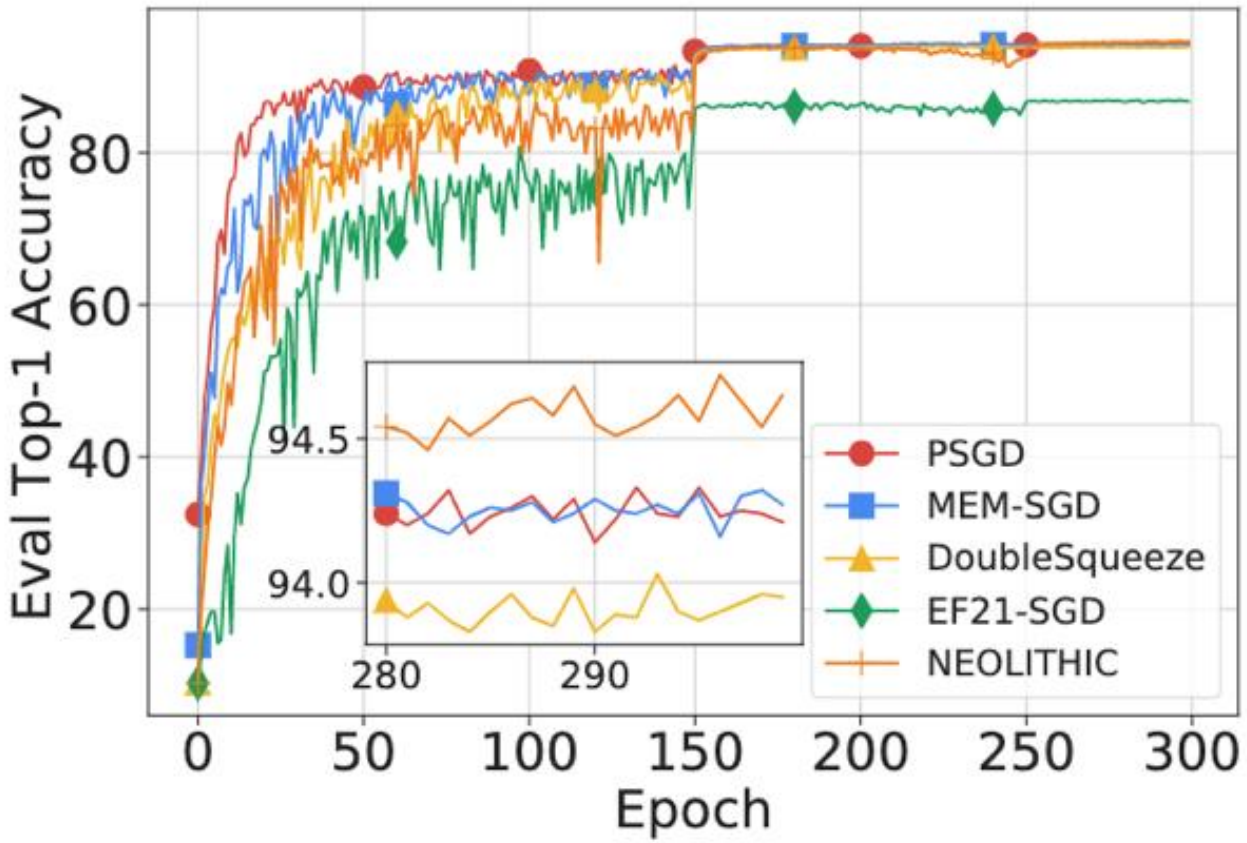
Non-compression

$$\Theta\left(\frac{\sigma^2 d}{\mu n\epsilon}+d\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\epsilon}\right)\right)$$

- When noise σ is large or ϵ is small, the first term dominates; **compression saves communication**!
- When $\sigma \to 0$, the second term dominates; **compression does not save communication**!

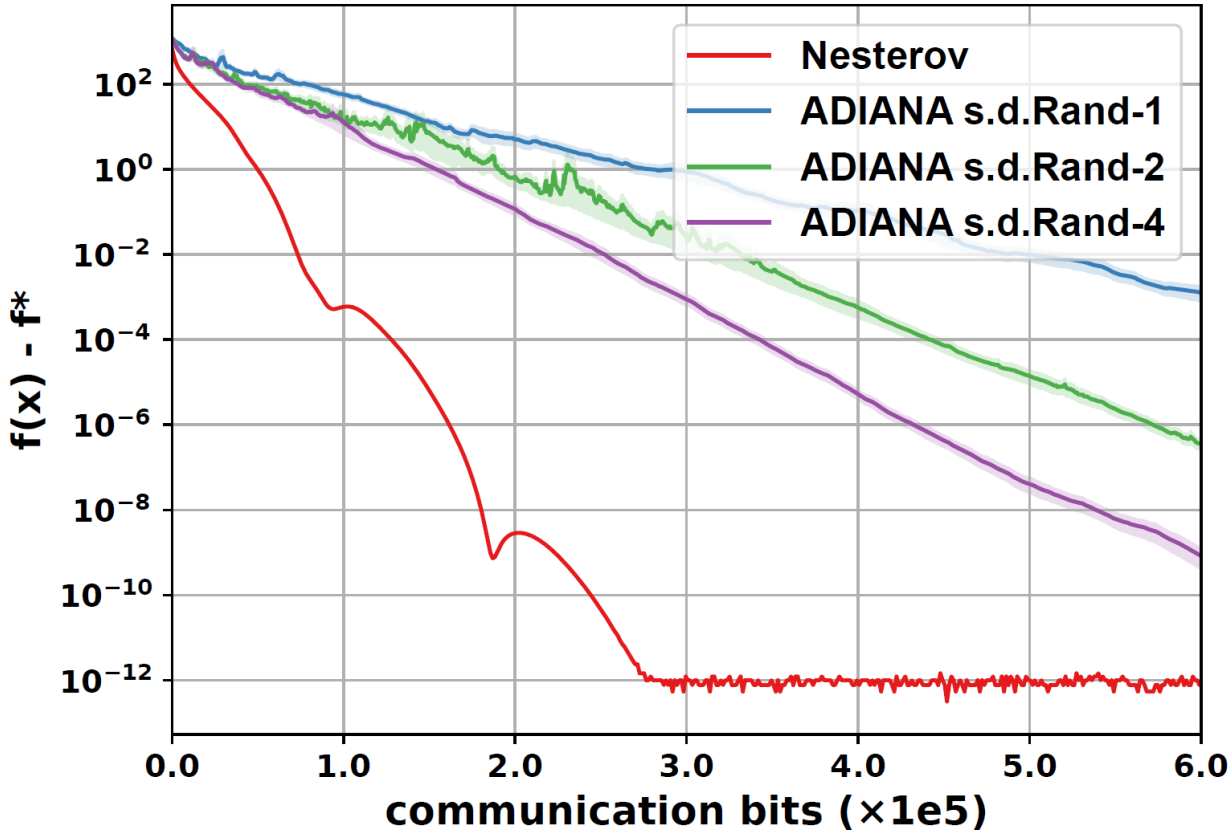
Compression saves communication in stochastic optimization

- 8 workers, **1% compression ratio** (top-k compressors), minibatch=128, R=2, ResNet18/ResNet20



Use 1% communication to reach the same accuracy

Compression does not communication in deterministic optimization



Deterministic least square

Unbiased compressor **does not save** communication

Summary

Compression saves total communication cost when

- Gradient noise is large
- A very accurate solution is required

Compression cannot save total communication cost when

- No gradient noise
- A less accurate solution is required

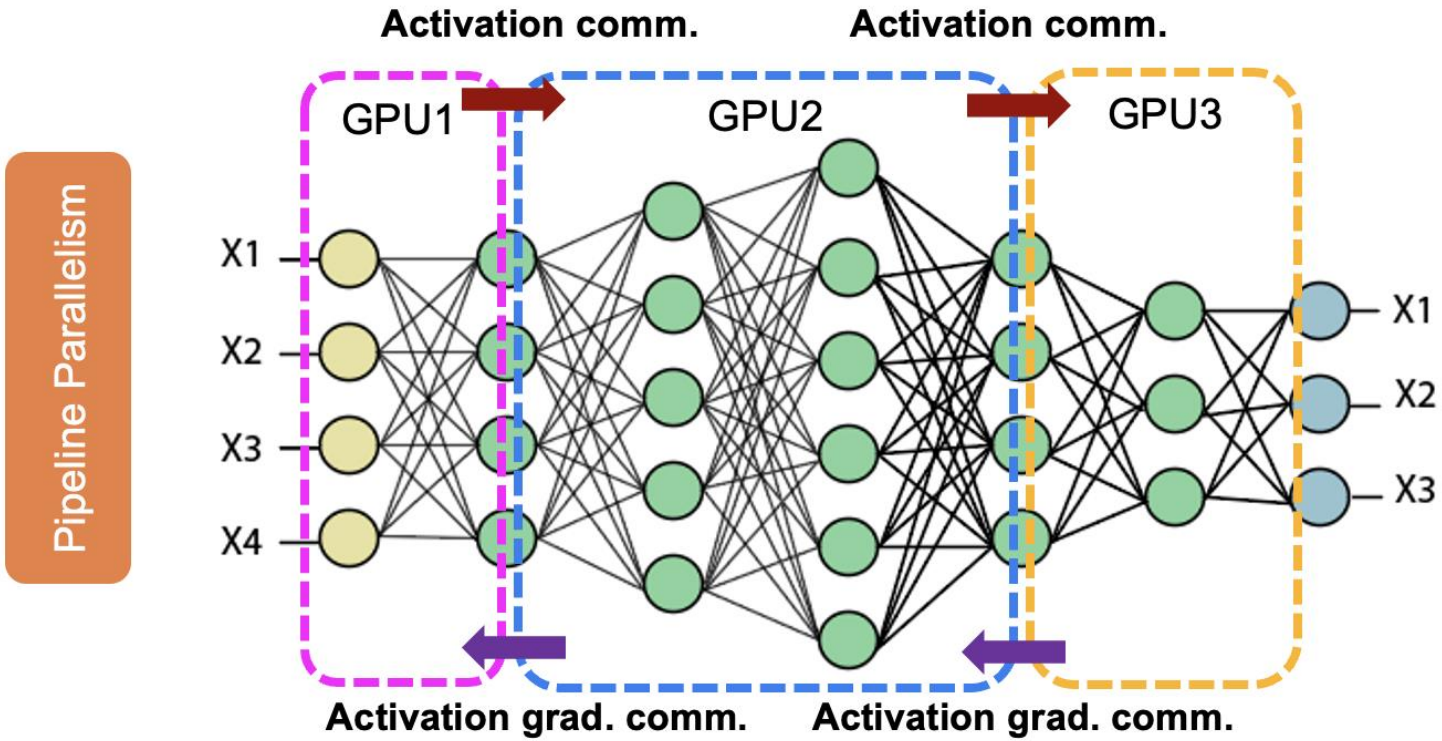
北京大学
PEKING UNIVERSITY

PART 06

Discussions

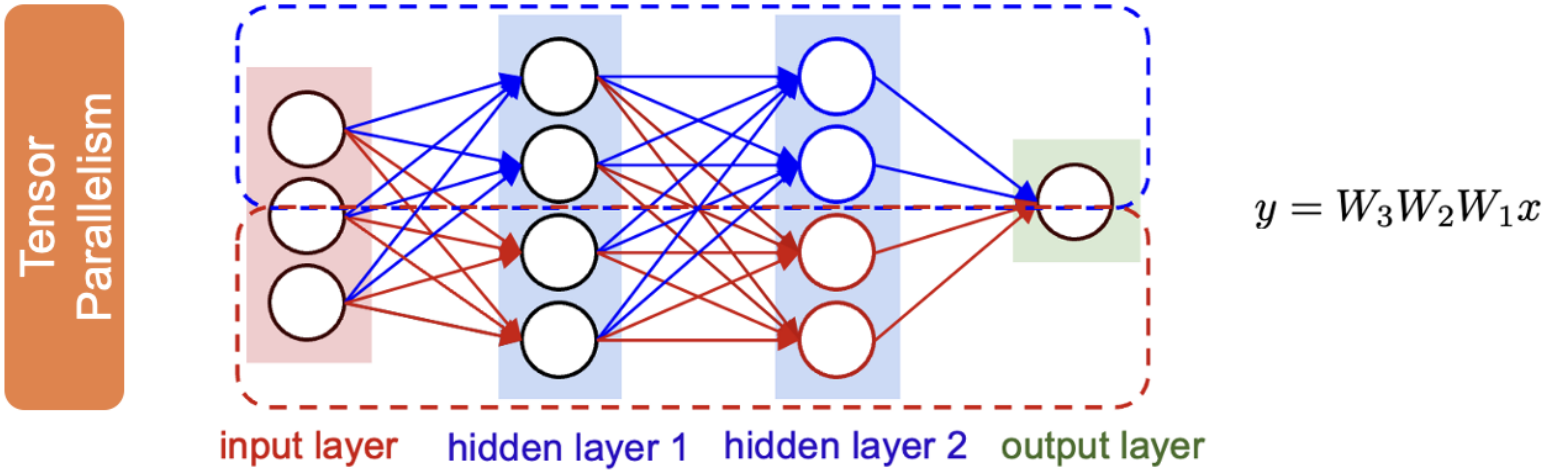
Too much data parallel! How about model parallel?

- 90% of distributed machine learning algorithms are on data parallel (as we discussed in our talk)
- Roughly speaking, communication issue for data parallelism is almost resolved
- However, communication overhead for model parallelism is almost untouched



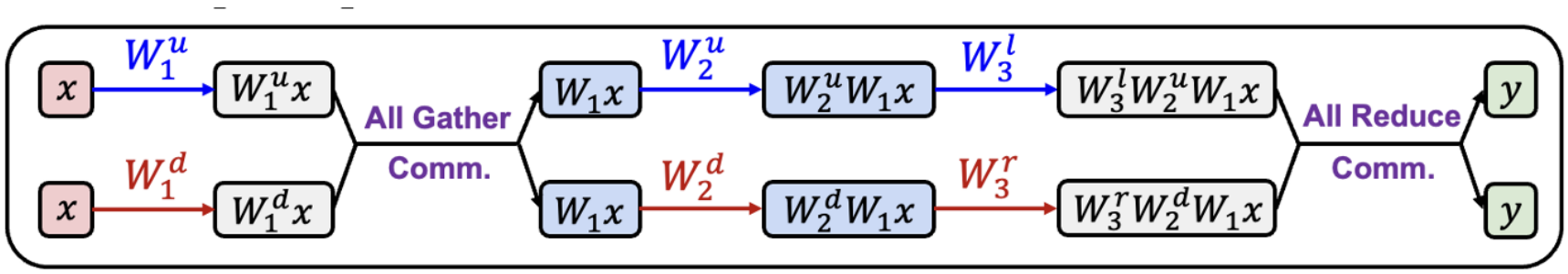
Too much data parallel! How about model parallel?

- However, communication overhead for model parallelism is almost untouched



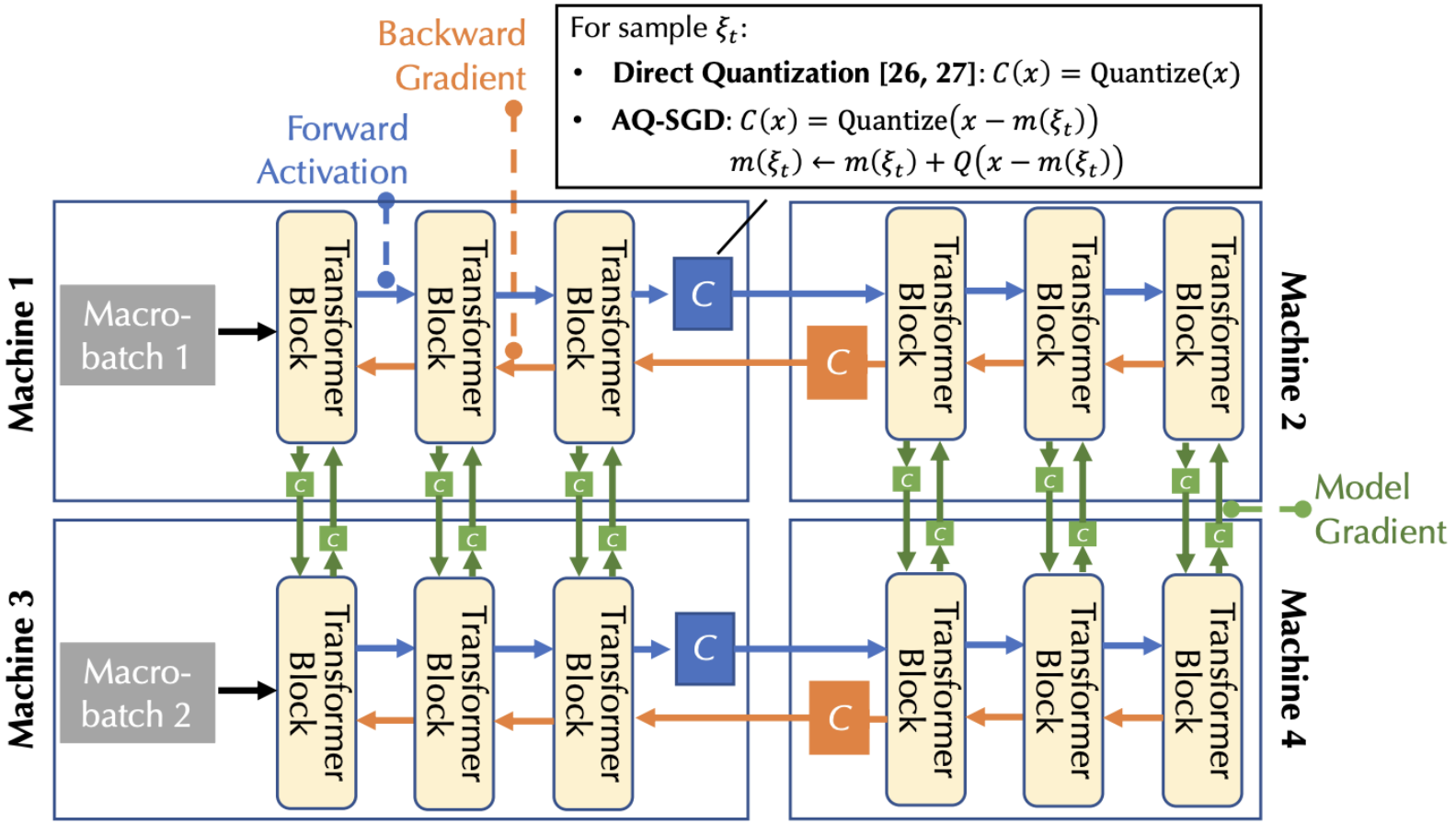
$$y = W_3 W_2 W_1 x$$

Computational graph



Communication compression in pipeline parallelism

- A very recent work [1] studies activation compression in pipeline parallelism



[1] J. Wang, et. al., "Fine-tuning Language Models over Slow Networks using Activation Quantization with Guarantees", NeurIPS 2022

Communication compression in pipeline parallelism

- Problem formulation

$$\min_{x\in\mathbb{R}^d} \quad f(x) := \mathbb{E}_{\xi\sim\mathcal{D}} F(b(a(\xi, x^{(a)}), x^{(b)})),$$

- When using communication compression, distortion is introduced **during** forward-backward process, which is fundamentally different from data parallelism that introduces distortion **after** FB process

$$x_{t+1} = x_t - \gamma \cdot (g_\xi(x_t) + \Delta_\xi(x_t)),$$

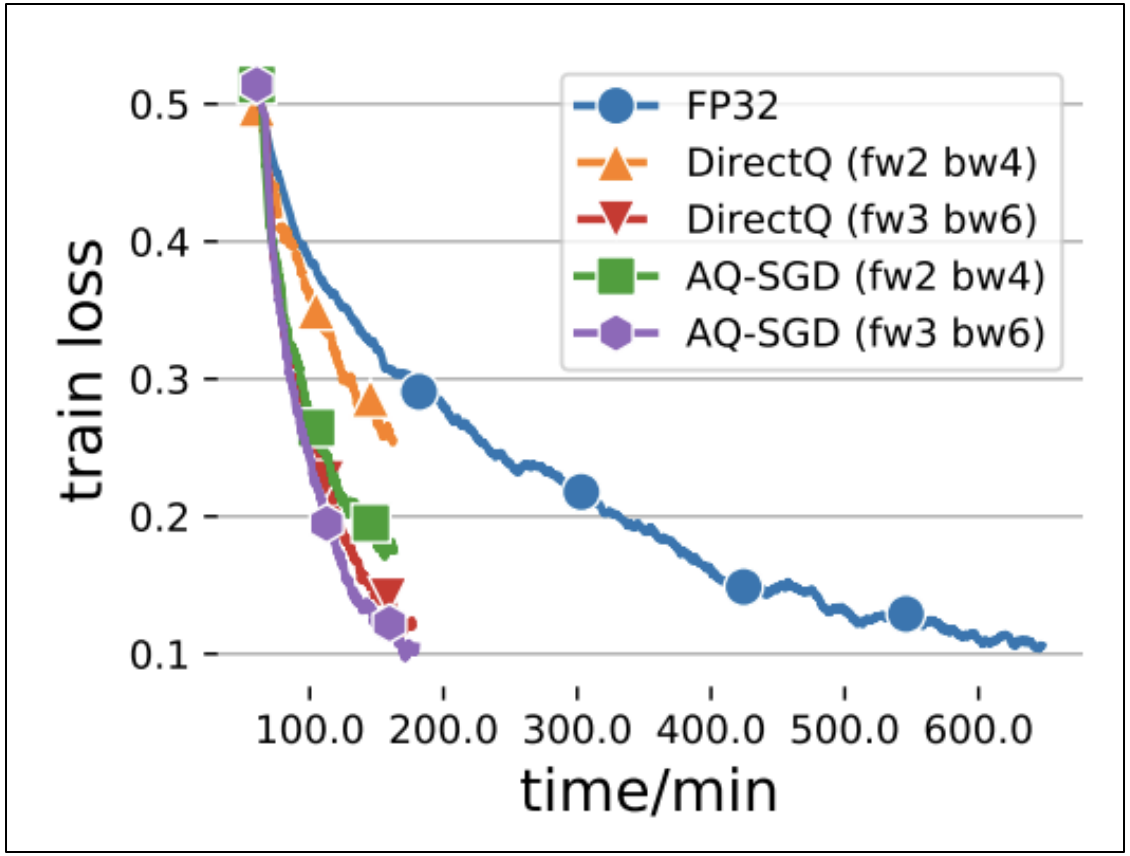
where the distortion is defined as

$$\begin{aligned}
\Delta_\xi^{(Q)}(x_t) &= Q(\nabla_a (f\circ b)|_{(m(\xi,x_t^{(a)}),x_t^{(b)})} \cdot \nabla_{x^{(a)}} a|_{x_t^{(a)}} - \nabla_a (f\circ b)|_{(m(\xi,x_t^{(a)}),x_t^{(b)})} \cdot \nabla_{x^{(a)}} a|_{x_t^{(a)}}, \\
\Delta_\xi^{(a)}(x_t) &= \nabla_a (f\circ b)|_{(m(\xi,x_t^{(a)}),x_t^{(b)})} \cdot \nabla_{x^{(a)}} a|_{x_t^{(a)}} - \nabla_a (f\circ b\circ a)|_{(x_t^{(a)},x_t^{(b)})} \\
\Delta_\xi^{(b)}(x_t) &= \nabla_{x^{(b)}} (f\circ b)|_{(m(\xi,x_t^{(a)}),x_t^{(b)})} - \nabla_{x^{(b)}} (f\circ b)|_{(a(\xi,x_t^{(a)}),x_t^{(b)})},
\end{aligned}$$

[1] J. Wang, et. al., “Fine-tuning Language Models over Slow Networks using Activation Quantization with Guarantees”, NeurIPS 2022

Communication compression in pipeline parallelism

Significant time saving



Perhaps too much theory? Need more practitioner

北京大学 PEKING UNIVERSITY

- 90% of distributed machine learning algorithms focus on convergence theories
- More research is needed to carefully evaluate communication-saving optimizers in real LLM training and finetuning tasks
- Transformers are standard models and Adam are standard optimizers in LLM
- We need to carefully design and evaluate communication-efficient optimizers for Transformer + Adam
- Communication-efficient optimizers for general non-convex problems have less practical values

Learn to distributed optimization

- Decentralized SGD
- Compressed SGD
- Local SGD or federated averaging

All these algorithms are designed by hands. They are not designed automatically.

Are these algorithms optimal? Probably not. Are there better algorithms? Probably yes.

How to design better algorithms? We do not know yet.

Learn-to-learn

- Learn-to-learn: Replace hand-designed update rules with a learned update rule

- Classical gradient-based algorithm

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t) \,.$$

- Learned gradient-based algorithm

$$\theta_{t+1} = \theta_t + g_t(\nabla f(\theta_t), \phi) \,.$$

g_t is a mapping to be learned. It is parameterized by ϕ . When taking a gradient, it outputs a better update direction

[1] M. Andrychowicz et. al., “Learning to learn by gradient descent by gradient descent”, NIPS 2016

Learned optimizer

- We can use RNN to learn a “good” update direction per iteration
- For each update, RNN takes in a gradient $\nabla f(\theta_t)$, outputs a learned update direction g_t
- Then the optimizee updates θ

$$h_t = p(U\nabla f(\theta_t) + Vh_{t-1})$$
$$g_t = q(Wh_t)$$

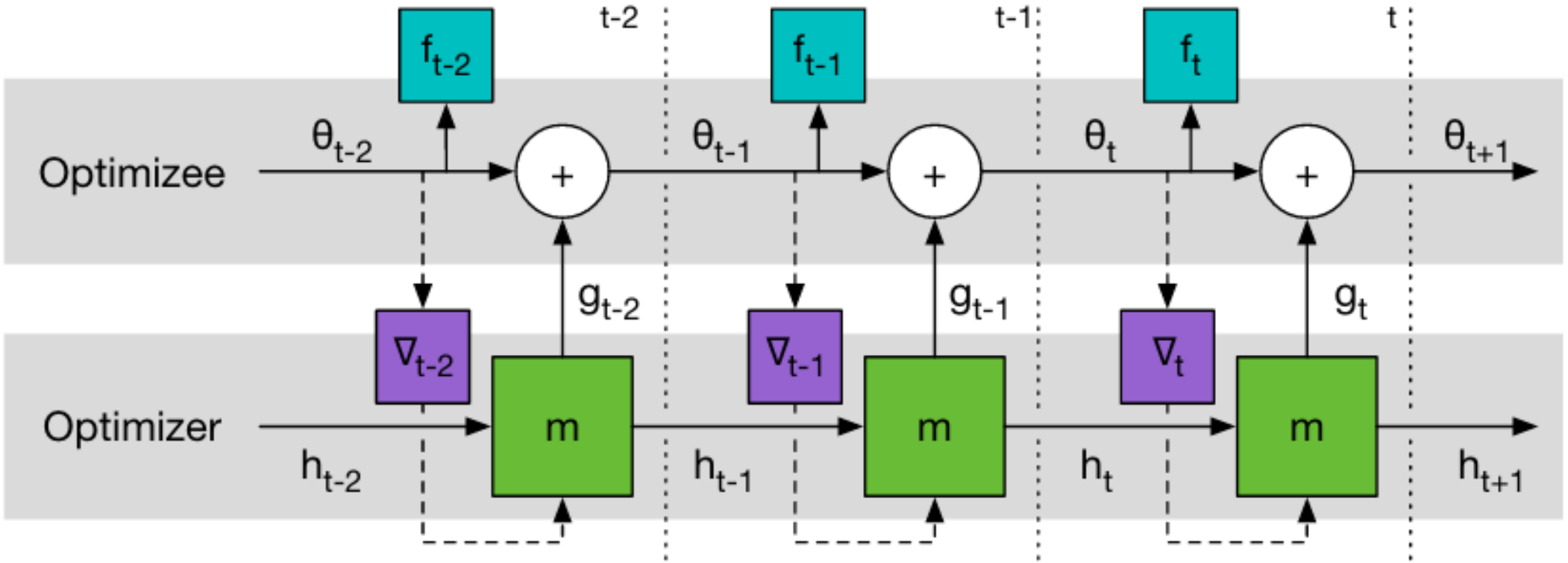
RNN; U, V, W are to be learned

$$\theta_{t+1} = \theta_t + g_t$$

Optimizee

[1] M. Andrychowicz et. al., “Learning to learn by gradient descent by gradient descent”, NIPS 2016

Learned optimizer



Learn to distributed optimization

- Decentralized SGD
- Compressed SGD
- Local SGD or federated averaging

Can we learn better distributed algorithms than those discussed in this lecture?

Can we learn better decentralized topology, or communication compressors?

How to effectively combine optimization and learning to develop algorithms?

Robust distributed optimization

北京大学 PEKING UNIVERSITY

- Distributed training over thousands of GPUs is extremely challenging
- Two major challenges: **stability** and **scalability**
- Stability: very common that some GPU crashes during training LLMs
  - Meta OPT-175B: 175+ job restarts caused by hardware failures in 2 months
  - InternLM: Waste 41700 GPU hours due to training crashes (>80% are infrastructure failures)
- Stability in LLM training is very important, we should develop algorithms that are tolerant with hardware failures

北京大学
PEKING UNIVERSITY

Thank you!

Kun Yuan homepage: https://kunyuan827.github.io/

We have openings for PostDocs