北京大學
PEKING UNIVERSITY

Optimization for Deep Learning

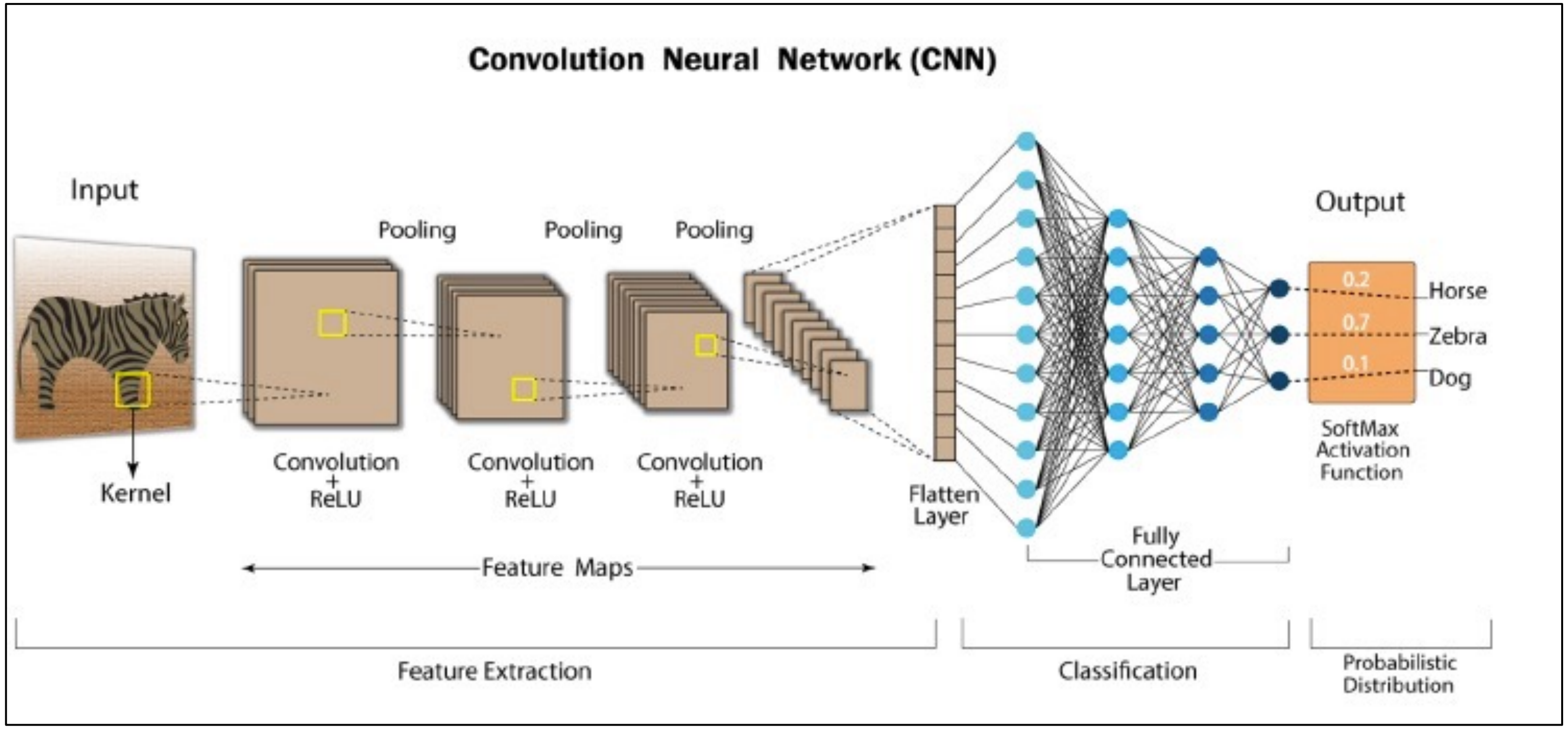
Lecture 1-4: Introduction

Kun Yuan

Peking University

This lecture previews distributed deep learning

Training deep neural network is notoriously difficult



DNN training = non-convexity + **massive dataset** + huge models

Distributed learning

- Training deep neural networks typically requires **massive** datasets; efficient and scalable distributed optimization algorithms are in urgent need

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

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

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

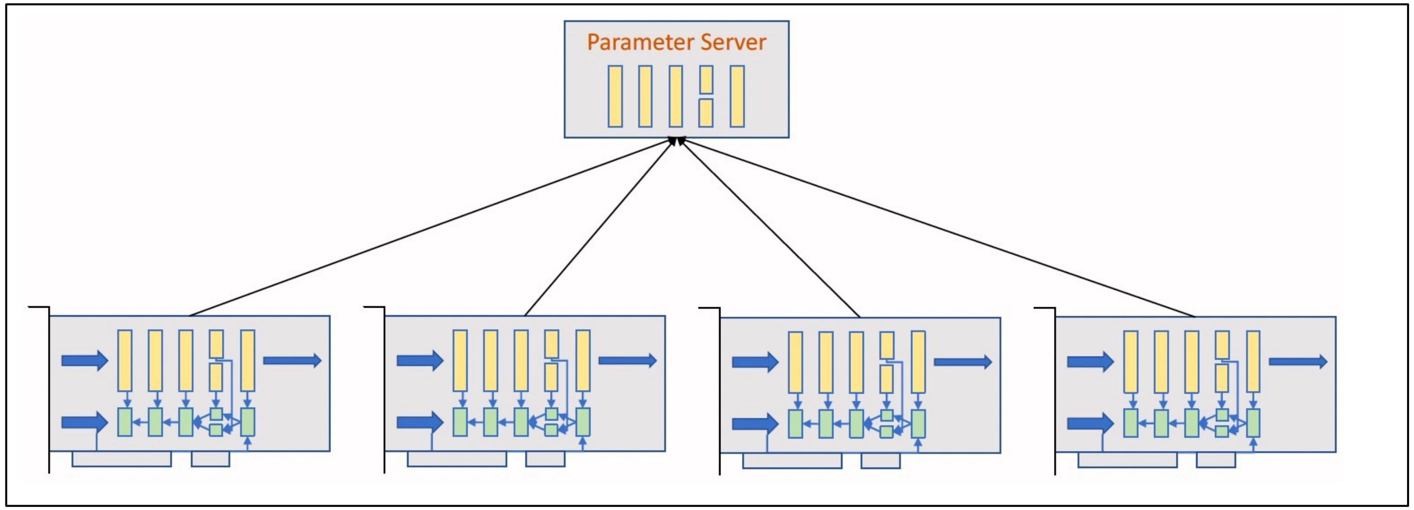
Vanilla parallel stochastic gradient descent (PSGD)

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

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

- 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

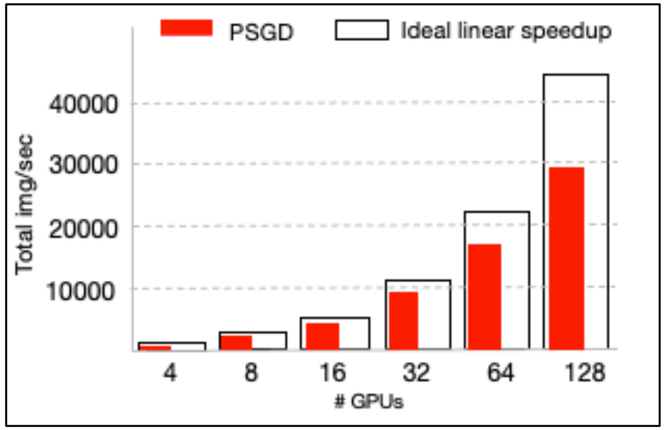
Vanilla parallel stochastic gradient descent (PSGD)



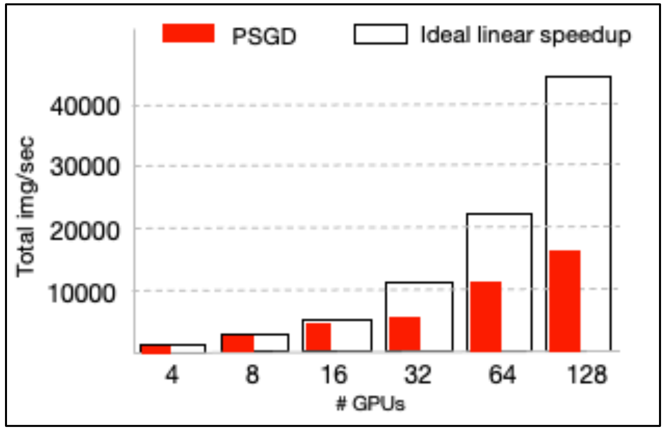
- Global average incurs $O(n)$ comm. overhead; **proportional to network size n**
- When network size n is large, PSGD suffers severe communication overhead

PSGD cannot achieve linear speedup due to comm. overhead

- PSGD cannot achieve ideal linear speedup in throughput due to comm. overhead
- Larger comm-to-compt ratio leads to worse performance in PSGD



Small comm.-to-compt. ratio



Large comm.-to-compt. ratio

- How can we accelerate PSGD? **We must reduce communication overhead.**

B. Ying, K. Yuan, H. Hu, Y. Chen and W. Yin, “BlueFog: Make decentralized algorithms practical for optimization and deep learning”, arXiv: 2111. 04287, 2021

Methodologies to save communication

PEKING UNIVERSITY

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

  [Communication compression]

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

  [Lazy communication]

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

  [Decentralized communication]

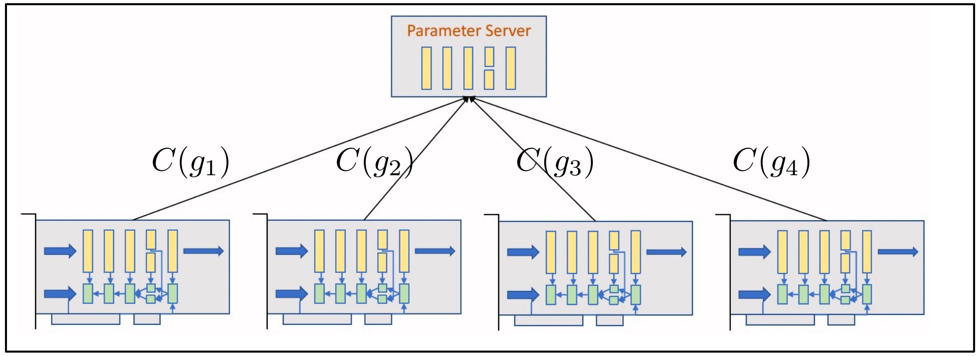
- Each node has to be synchronized with each other during each iteration

  [Asynchronous communication]

Communication compression

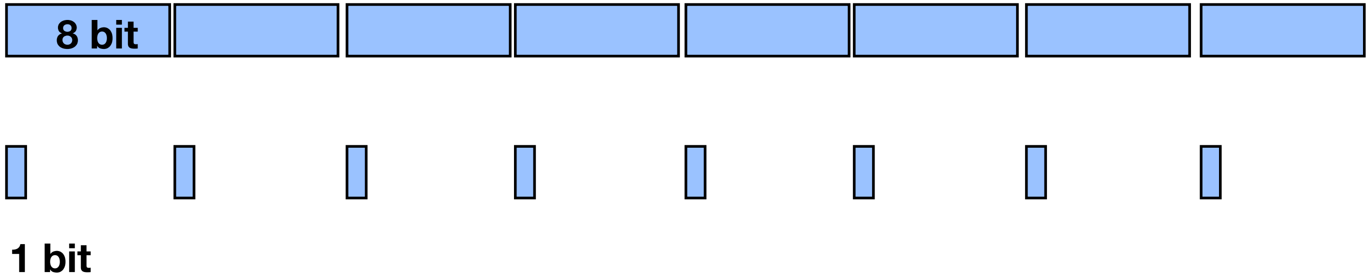
- A basic (but not state-of-the-art) algorithm is QSGD [Alistarh et. al., 2017]

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



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

Quantization

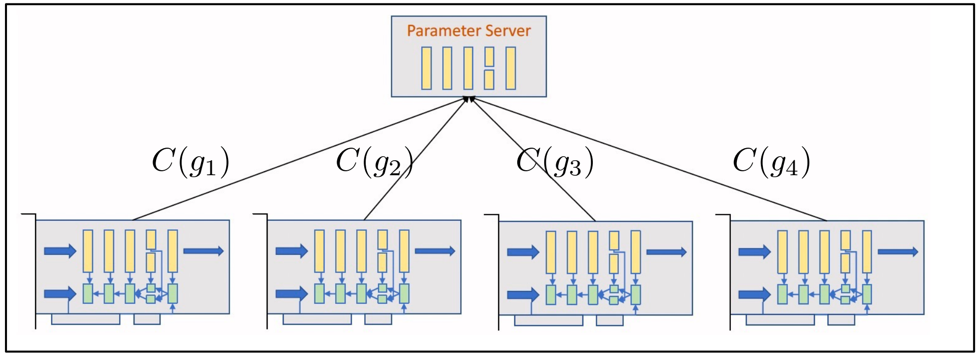


Communication compression

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

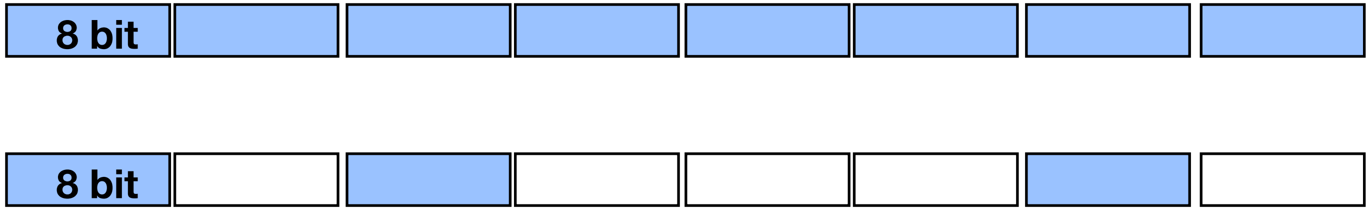
$$g_i^{(k)} = \nabla F(x_i^{(k)}; \xi_i^{(k)})$$

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



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

Sparsification



Communication compression

- How to develop effective communication compression strategies?
- How does communication compression effect the convergence rate?
- Is there any advanced optimization algorithm that can handle compression error better?

We leave these questions to the main lecture.

Lazy communication (Federated Average)

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

$$x_i^{(k+1)} = \begin{cases} x_i^{(k+\frac{1}{2})} & \text{if } \mathrm{mod}(k, \tau) \neq 0 \\ \frac{1}{n}\sum_{j=1}^{n} x_j^{(k+\frac{1}{2})} & \text{if } \mathrm{mod}(k, \tau) = 0 \end{cases}$$ (Lazy comm.)

- Nodes communicate once every τ iterations [Konecny et .al. 2015, 2016]
- Or nodes communicate when necessary, i.e., [Chen et. al. 2018; Liu et.al., 2019]

Lazy compression

- How does lazy communication affect the convergence rate?
- How does data heterogeneity affect the convergence rate?
- How to tune the lazy communication period?
- How to develop efficient algorithms to overcome the data heterogeneity issue?

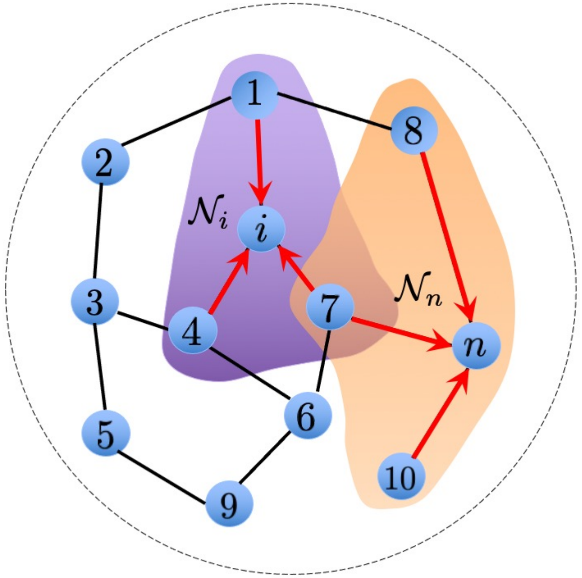
We leave these questions to the main lecture.

Decentralized communication

- To break $O(n)$ comm. overhead, we replace global average with partial average

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

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



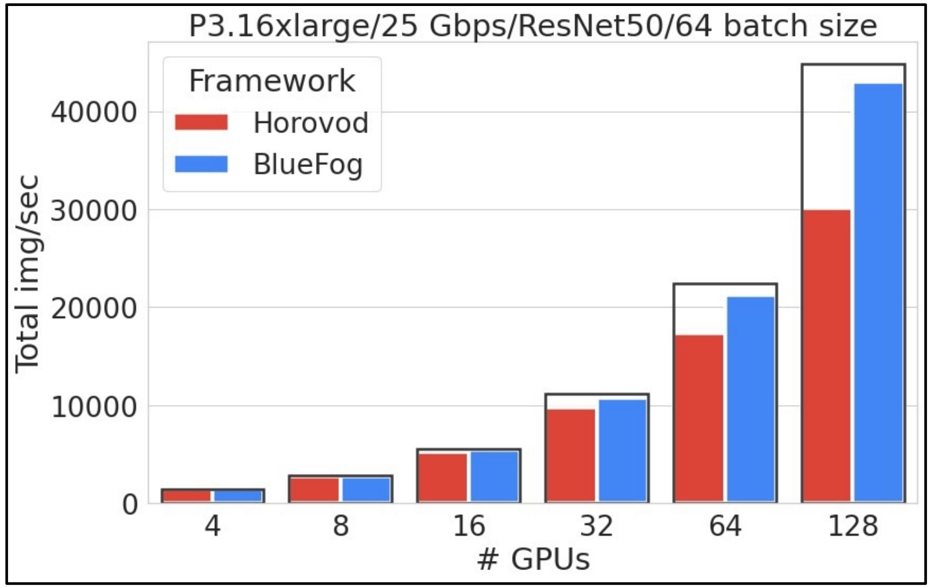
- DSGD = local SGD update + partial averaging [LS08]

- $\mathcal{N}_i$ is the set of neighbors at node i ; w_{ij} scales information from j to i

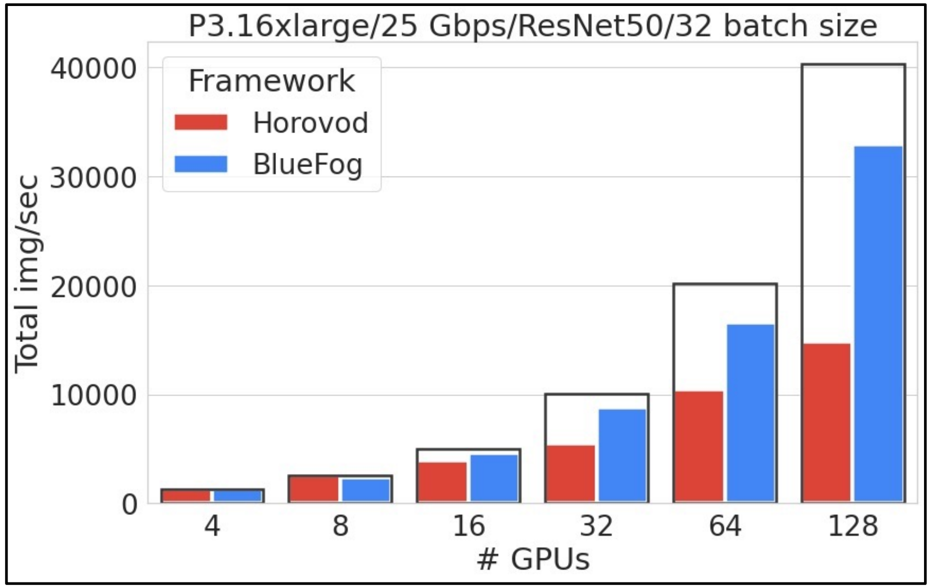
- Incurs $O(d_{\max})$ comm. overhead per iteration where $d_{\max} = \max_i\{|\mathcal{N}_i|\}$ is the graph maximum degree

DSGD is more communication-efficient than PSGD

- DSGD (BlueFog) has **better linear speedup** than PSGD (Horovod) due to its small comm. overhead



Small comm.-to-compt. ratio



Large comm.-to-compt. ratio

B. Ying, K. Yuan, H. Hu, Y. Chen and W. Yin, "BlueFog: Make decentralized algorithms practical for optimization and deep learning", arXiv: 2111. 04287, 2021

Lazy compression

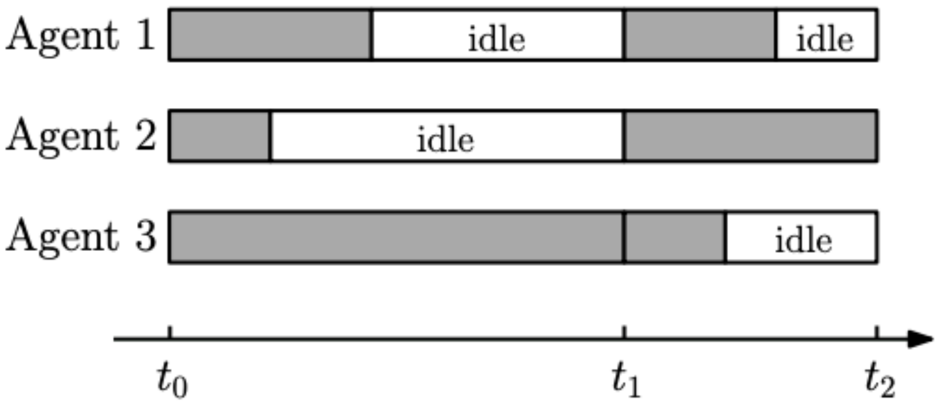
北京大学 PEKING UNIVERSITY

- How does graph affect the convergence rate?
- How does data heterogeneity affect the convergence rate?
- How to develop efficient graph that can accelerate the convergence rate?
- How to develop efficient algorithms to overcome the data heterogeneity issue?

We leave these questions to the main lecture.

Asynchronous communication

- Synchronization across nodes causes severe idle time



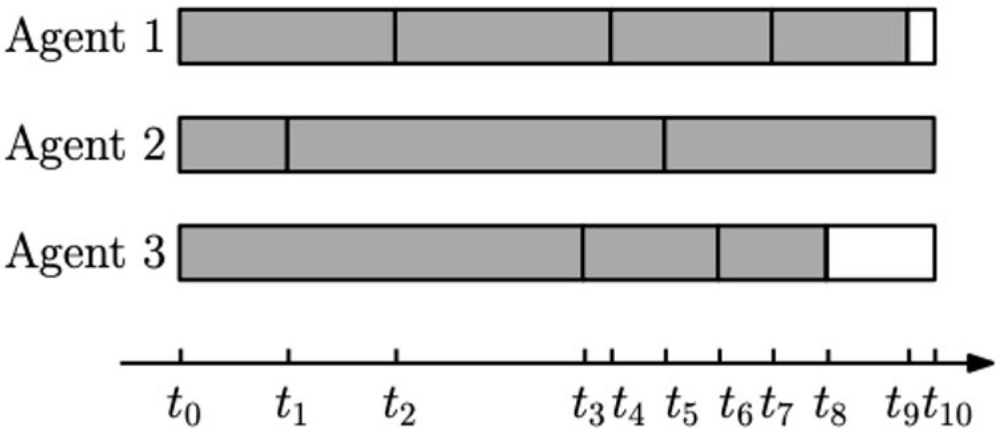
Synchronous comm.

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

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

Asynchronous communication

- Asynchronization reduces idle time, but it cause delayed gradient



Asynchronous comm.

$$x^{(1)} = x^{(0)} - \gamma \nabla F(x^{(0)}, \xi_2^{(0)})$$

$$x^{(2)} = x^{(1)} - \gamma \nabla F(x^{(0)}, \xi_1^{(0)})$$

$$x^{(3)} = x^{(2)} - \gamma \nabla F(x^{(0)}, \xi_3^{(0)})$$

$$\vdots$$

$$x^{(k+1)} = x^{(k)} - \gamma \boxed{\nabla F(x^{(k-\tau)}, \xi_i^{(k-\tau)})}$$

Delayed gradient

Asynchronous compression

- How does delayed gradient affect the convergence rate?
- How does data heterogeneity affect the convergence rate?
- How to develop efficient algorithms to handle delayed gradient?
- How to develop efficient algorithms to overcome the data heterogeneity issue?

We leave these questions to the main lecture.

Byzantine distributed learning

- In the above scenarios, we assume all nodes are honest. They collaborate to learn better

- In some Byzantine scenario, some nodes are malicious but **we do not know their identities**

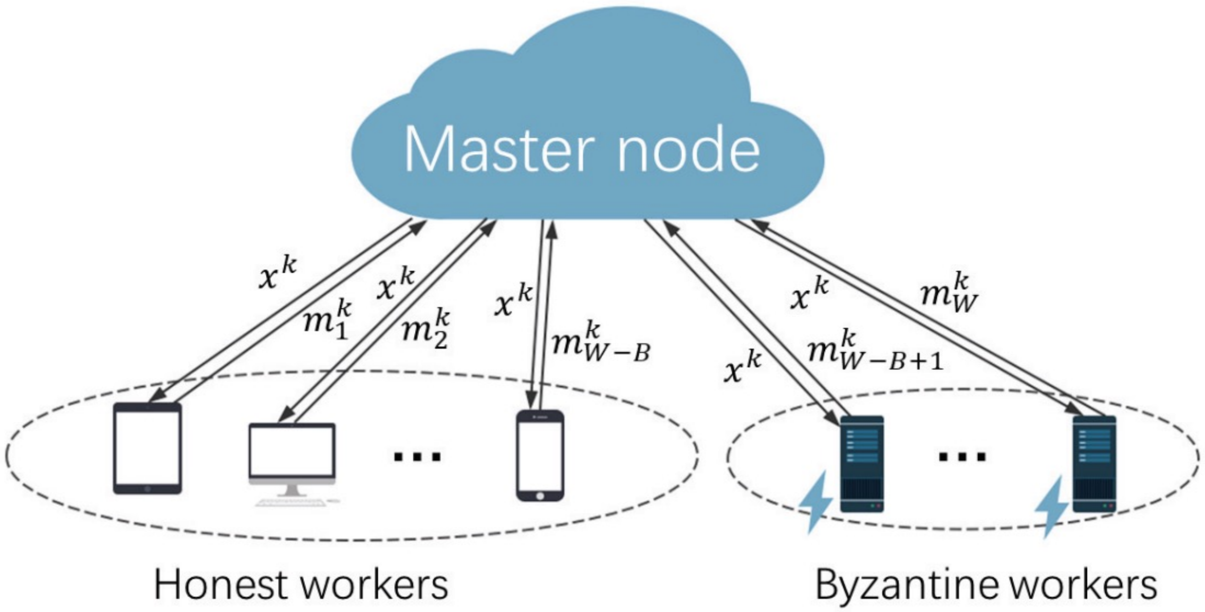


Figure is from (Wu et. al., 2020)

Byzantine distributed learning

Existence of the Byzantine node brings significant challenge in algorithm design

- How to develop algorithms to avoid the attacks from the Byzantine node?
- How does the number of Byzantine nodes affect the convergence rate?
- How does data heterogeneity affect the Byzantine-robust algorithms

We leave these questions to the main lecture.

Summary

- Distributed deep learning is a very hot research topic
- It is widely used in training large language models and federated learning
- Some of the most important topics are:
    - Compressed/decentralized/lazy/asynchronous communication
    - Byzantine learning