Current Research Trends

Mingyi Hong

University Of Minnesota

M. Hong would like to thank Dr. Prashant Khanduri for helping prepare the slides.

Outline

- **Communication Efficient Decentralized Optimization**
  - Motivation
  - Problem
  - Distributed: QSGD, Sparsification
  - Decentralized: Choco-Gossip based approaches
- **Optimization in the presence of Adversaries**
  - Motivation
  - Types of Byzantine Attacks
  - ByzantineSGD
  - Literature
- **Other Issues**

Communication Efficient Decentralized Optimization

Motivation

- Let us consider a distributed or decentralized architecture with m nodes
- **Goal:** To solve:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x}) = \sum_{i=1}^{m} f_i(\boldsymbol{x}) \quad \text{with} \quad f_i(\boldsymbol{x}) = \mathbb{E}_{\xi_i}[g_i(\boldsymbol{x}, \xi_i)] \tag{1.1}$$

- **Typical protocol at each node**
  - Forwards an n-dimensional vector
  - Receives an n-dimensional vector
  - Update its iterate and repeat until convergence

Problem

- **Issue:** In each iteration, every node communicates an n-dimensional vector to its neighbors (or the FC)
  - Dimension n can be potentially **very large**
  - Gradients are **dense** for deep learning applications
  - Results in **network congestion** because of high communication requirements
- **Solution:** Each node communicates a **compressed** gradient
  - **Quantization:** Quantize each entry of the vector into fixed levels
  - **Sparsification:** Only send some entries (prominent ones or randomly selected) of the vector

Example

- Suppose each node uses a 32-bit arithmetic
  - **Uncompressed Communication:** Each node communicates (and receives) $32n$ bits per iteration
  - **Quantization:** 1-bit quantization
    - Only n-bits compared to $32n$-bits
  - **Sparsification:** Sending only k out of n entries
    - $32k$-bits instead of $32n$-bits
- **Trade-off:** Communication vs Convergence
  - Compression $\Rightarrow$ increased variance $\Rightarrow$ **worse** convergence
  - Compression $\Rightarrow$ faster communication $\Rightarrow$ **improved** convergence

Question: Can compression lead to overall faster convergence?

Quantized SGD (QSGD)[1]

- **Parallel implementation**
  - Complete graph with m nodes to solve (1.1)
  - **Homogeneous Data:** Each node have access to data from same distribution, i.e.,

$$f(\boldsymbol{x}) = f_i(\boldsymbol{x}) = \mathbb{E}_{\xi_i}[g_i(\boldsymbol{x}, \xi_i)] \quad \text{with} \quad \xi_i \in \mathcal{D} \quad \forall i \in [m]$$

    - When all the nodes have access to a common database
- Each node receives **quantized** stochastic gradient vectors from other nodes

[1] Alistarh et al., QSGD: Communication-efficient SGD via gradient quantization and encoding, Advances in Neural Information Processing Systems, 2017.

Algorithm: QSGD

Quantized Stochastic Gradient Descent (QSGD)

- **For** each iteration k **do**
- Compute stochastic gradient: $\nabla g_i(\boldsymbol{x}^k, \xi_i^k)$
- Perform gradient compression: $M_i^k \leftarrow \text{Encode}(\nabla g_i(\boldsymbol{x}^k, \xi_i^k))$
- Broadcast M_i^k to other nodes
- **For** each node ℓ
- Receive M_ℓ^k from ℓth node
- Decode received gradients: $\widehat{\nabla g_\ell(\boldsymbol{x}^k)} = \text{Decode}(M_\ell^k)$
- **End**
- Update iterate: $\boldsymbol{x}^{k+1} = \boldsymbol{x} - \frac{\alpha^k}{m} \sum_{\ell=1}^m \widehat{\nabla g_\ell(\boldsymbol{x}^k)}$

Same as SGD other that the Encode and Decode steps

Assumptions

Assumption 1

The function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, convex, L-smooth, and unknown. The algorithm only has access to the stochastic gradients of f, i.e., $\nabla g(\boldsymbol{x}, \xi_i)$.

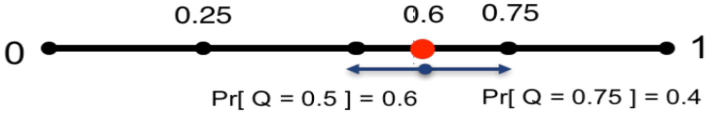
Assumption 2

The stochastic gradient satisfies:

1. *Unbiased: $\mathbb{E}[\nabla g(\boldsymbol{x}, \xi_i)] = \nabla f(\boldsymbol{x})$ (We assumed $\xi_i \sim \mathcal{D}\ \forall i \in [m]$)*
2. *Bounded Second Moment: The stochastic gradient has second moment at most B if $\mathbb{E}[\|\nabla g(\boldsymbol{x}, \xi_i)\|^2] \le B$ for all $\boldsymbol{x} \in \mathbb{R}^n$*
3. *Bounded Variance: The stochastic gradient has bounded variance if $\mathbb{E}[\|\nabla f(\boldsymbol{x}; \xi_i) - \nabla f(\boldsymbol{x})\|^2] \le \sigma^2$ for all $\boldsymbol{x} \in \mathbb{R}^n$*

Quantization Scheme

- **Quantization function:** $Q_s(\boldsymbol{v})$
  - $s \geq 1$ uniformly distributed levels between 0 and 1
  - Unbiased quantization and introduces minimal variance



Quantization with $s = 5$ levels

- **Efficient Coding of Gradients** $Q_s(\boldsymbol{v}) = (\|\boldsymbol{v}\|, \boldsymbol{\sigma}, s \cdot \boldsymbol{\zeta})$
  - $\|\boldsymbol{v}\|$: 32-bit representation
  - $\boldsymbol{\sigma}$: Sign of individual elements: 1-bit representation
  - $s \cdot \boldsymbol{\zeta}$: Integer quantization levels: Use Elias codes[2]

[2]Elias, Universal codeword sets and representations of the integers, IEEE Transactions on Information Theory, 1975.

Stochastic Quantization

- For any $\boldsymbol{v} \in \mathbb{R}^n$ with $\boldsymbol{v} \neq 0$, $Q_s(\boldsymbol{v})$ is defined as

$$Q_s(v_i) = \|\boldsymbol{v}\| \cdot \mathrm{sgn}(v_i) \cdot \zeta_i(\boldsymbol{v}, s) \quad \text{for} \quad i \in [n]$$

  where $\mathrm{sgn}(v_i)$ is the sign function and $\zeta_i(\boldsymbol{v}, s)$'s are independent random variables defined as:

- Let $0 \leq \ell < s$ be s.t. $|v_i|/\|\boldsymbol{v}\| \in [\ell/s, (\ell+1)/s]$, then

$$\zeta_i(\boldsymbol{v}, s) = \begin{cases} \ell/s & \text{w.p.} \quad 1 - p\left(\frac{|v_i|}{\|\boldsymbol{v}\|}, s\right) \\ (\ell+1)/s & \text{otherwise} \end{cases}$$

  with $p(a, s) = as - \ell$ for any $a \in [0, 1]$. $Q(\boldsymbol{v}, s) = 0$ if $\boldsymbol{v} = 0$

Distribution of $\zeta_i(\boldsymbol{v}, s)$ has minimal variance over distributions with support $\{0, 1/s, \ldots, 1\}$ and $\mathbb{E}[\zeta_i(\boldsymbol{v}, s)] = |v_i|/\|\boldsymbol{v}\|$

Main Result

Theorem 1.1

- *Under Assumptions 1 and 2, and* $\alpha^k = 1/(L + \sqrt{m}/\gamma)$ *and to achieve* $\mathbb{E}\left[f\left(\frac{1}{K}\sum_{k=1}^{K} \boldsymbol{x}^k\right)\right] - f^* \leq \epsilon$*, QSGD requires* K *iterations with*

$$K = O\left(R^2 \cdot \max\left\{\frac{2B'}{m\epsilon^2}, \frac{L}{\epsilon}\right\}\right).$$

  with $\gamma = \frac{1}{\sigma}\sqrt{\frac{2}{K}}$, $R^2 = \sup_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{x} - \boldsymbol{x}^0\|$, $\sigma = B'$, $B' = \min\{n/s^2, \sqrt{n}/s\}B$

- *Moreover, if we take* $s = \sqrt{n}$, **QSGD** *requires* $2.8n + 32$ *bits per iteration compared to* $32n$ *required by* **SGD**

Comparison to SGD

Network	Dataset	Params.	Init. Rate	Top-1 (32bit)	Top-1 (QSGD)	Speedup (8 GPUs)
AlexNet	ImageNet	62M	0.07	59.50%	**60.05**% (4bit)	**2.05** ×
ResNet152	ImageNet	60M	1	**77.0**%	76.74% (8bit)	**1.56** ×
ResNet50	ImageNet	25M	1	74.68%	**74.76**% (4bit)	**1.26** ×
ResNet110	CIFAR-10	1M	0.1	93.86%	**94.19**% (4bit)	**1.10** ×
BN-Inception	ImageNet	11M	3.6	-	-	**1.16**× (projected)
VGG19	ImageNet	143M	0.1	-	-	**2.25**× (projected)
LSTM	AN4	13M	0.5	81.13%	**81.15** % (4bit)	**2**× (2 GPUs)

Speed-up achieved by **QSGD** compared to **SGD** while training different networks with the percentages indicating the percentage of time consumed in communication process

SignSGD[3]

- **SignSGD:** 1-bit compression
  - Transmitting just the **sign** of each stochastic gradient
- **SignSGD** works well when
  - Gradients are as **dense**
  - Both gradients and noise are dense in deep learning problem
- **Homogeneous Data:** Each node have access to data from same distribution, i.e.,

$$f(\boldsymbol{x}) = f_i(\boldsymbol{x}) = \mathbb{E}_{\xi_i}[g_i(\boldsymbol{x}, \xi_i)] \text{ with } \xi_i \sim \mathcal{D} \ \forall i \in [m]$$

- **Non-Convex** functions

[3] Bernstein et al., SignSGD: Compressed Optimisation for Non-Convex Problems, Thirty-fifth International Conference on Machine Learning, 2018

Algorithm: SignSGD

SignSGD

- **Input** Learning rate α, initial iterate $\boldsymbol{x}^0$
- **For** $k = 0$ to K **do**
- Compute Stochastic Gradient: $\nabla g(\boldsymbol{x}^k, \xi^k)$ with $|\xi^k| = b_k$
- Update: $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha \; \mathsf{sign}(g(\boldsymbol{x}^k, \xi^k))$
- **End For**

Algorithm: SignSGD with Majority Vote

Distributed SignSGD with Majority Voting (MV)

- **Input** Learning rate α, initial iterate $\boldsymbol{x}^0$, m nodes each with gradient estimate $\nabla g_i(\boldsymbol{x}^k, \xi_i^k)$ for $i \in [m]$ and with $|\xi_i^k| = b_k$
- **For** $k = 0$ to $K - 1$ **do**
- **on** Central Node
- **pull** sign$(\nabla g_i(\boldsymbol{x}^k, \xi_i^k))$ **from** each node
- **push** sign$\left[\sum_{i=1}^m \text{sign}(\nabla g_i(\boldsymbol{x}^k, \xi_i^k))\right]$ **to** each node (MV)
- **on** each worker update:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha \ \text{sign}\left[\sum_{i=1}^m \text{sign}(\nabla g_i(\boldsymbol{x}^k, \xi_i^k))\right]$$

- **End For**

Assumptions

Assumption 3 (Smooth)

The function f *is non-convex and for all* $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ *we have*

$$\|\nabla f_i(\boldsymbol{y}) - \nabla f_i(\boldsymbol{x})\| \leq L_i \|\boldsymbol{y} - \boldsymbol{x}\|$$

for a vector of non-negative constants $\vec{L} = [L_1, \ldots, L_n]$

Assumption 4 (Variance Bound)

The stochastic gradient computed at each $j \in [m]$ *for each dimension* $i \in [n]$ *satisfies*

$$\mathbb{E}[\|\nabla g_j(\boldsymbol{x}, \xi_j)_i - \nabla g_j(\boldsymbol{x})_i\|] \leq \sigma_i^2$$

for a vector of non-negative constants $\vec{\sigma} = [\sigma_1, \ldots, \sigma_n]$

Main Results: SignSGD

Theorem 1.2

For iterations 1 to K under Assumptions 3 and 4, with $\alpha = 1/\sqrt{\|L\|_1 K}$ and mini-batch size $b_k = K$. Let N be the cumulative number of stochastic gradient calls up to step K, i.e., $N = O(K^2)$. Then for **SignSGD** *we have*

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1}\|\nabla f(\boldsymbol{x}^k)\|_1\right]^2 \leq \frac{1}{\sqrt{N}}\left[\sqrt{\|\vec{L}\|_1}\left(f_0 - f^* + \frac{1}{2}\right) + 2\|\vec{\sigma}\|_1\right]^2$$

Note that the norms are ℓ_1-norms!

Main Results: SignSGD with Majority Voting

Theorem 1.3

- **SignSGD with Majority Vote** *with* m *workers converges at least as fast as* **SignSGD**
- *Further Assuming that the noise in each component of the stochastic gradient is unimodal and symmetric about the mean we have for* **SignSGD with Majority Vote**

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1}\|\nabla f(\boldsymbol{x}^k)\|_1\right]^2 \leq \frac{1}{\sqrt{N}}\left[\sqrt{\|\vec{L}\|_1}\left(f_0 - f^* + \frac{1}{2}\right) + \frac{2}{\sqrt{m}}\|\vec{\sigma}\|_1\right]^2$$

Comparison of SignSGD to SGD

- **Define:**
  - Density of a vector: $\phi(\boldsymbol{v}) = \frac{\|\boldsymbol{v}\|_1^2}{n\|\boldsymbol{v}\|_2^2}$
  - Lower bound on gradient density: $\phi(\nabla g)$
  - $R_1 = \frac{\sqrt{\phi(\vec{L})}}{\phi(\nabla g)}$ and $R_2 = \frac{\phi(\vec{\sigma})}{\phi(\nabla g)}$ with
- Convergence of **SignSGD** can be rephrased as:

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1}\|\nabla f(\boldsymbol{x}^k)\|_2\right]^2 \leq \frac{2}{\sqrt{N}}\left[R_1 L\left(f_0 - f^* + \frac{1}{2}\right)^2 + 4R_2\sigma^2\right]$$

with $\sigma^2 = \|\vec{\sigma}\|_2^2$ and $L = \|\vec{L}\|_\infty$

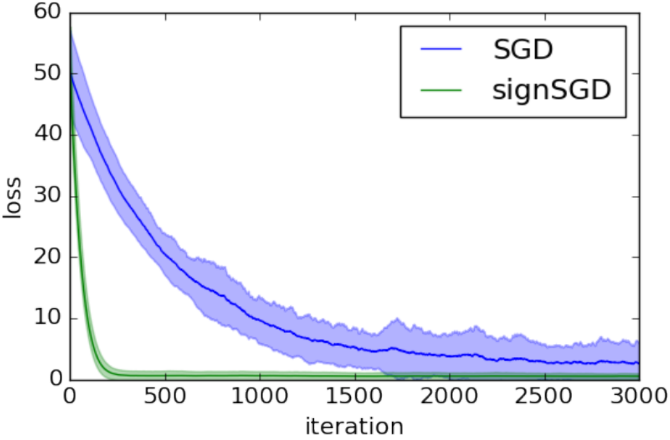
- For **SGD**

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1}\|\nabla f(\boldsymbol{x}^k)\|_2^2\right] \leq \frac{1}{\sqrt{N}}\left[2L(f_0 - f^*) + \sigma^2\right]$$

Comparison of SignSGD to SGD

- $R_1 \gg 1$ and $R_2 \gg 1$
  - **SGD** is better suited than **SignSGD**
  - **Curvature and the stochasticity are much denser than the typical gradient**
- NOT$[R_1 \gg 1]$ and NOT$[R_2 \gg 1]$
  - **SignSGD** is as fast or faster than **SGD**
  - **Neither curvature nor stochasticity are much denser than the gradient**

SGD vs SignSGD



Toy example with $f(x) = \frac{1}{2}\|x\|_2^2$ for $x \in \mathbb{R}^{100}$.

Literature

- **Lossy compression:** Low precision arithmetic[4]
- **Sparsification**[5]
- **Three levels**[6]

---

[4]Abadi et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv 2016.

[5]De Sa, Christopher M., et al. "Taming the wild: A unified analysis of hogwild-style algorithms." Advances in neural information processing systems. 2015.

[6]Wen et al., Terngrad: Ternary gradients to reduce communication in distributed deep learning, NeurIPS 2017.

Compression for Decentralized Optimization: Choco-Gossip[7]

- **Decentralized** networks
- General compression
  - Quantize or Sparsify, Biased or unbiased
- **Choco-Gossip:** Decentralized consensus with **compression**
  - Achieves **exact consensus** while converging to the true solution
- **Past Approaches:** Only **neighborhood convergence** or required **fairly accurate compression**
- **Choco-SGD**: Decentralized SGD based on **Choco-Gossip**

[7] Koloskova et al., Decentralized Stochastic Optimization and Gossip Algorithms with Compressed Communication, 36th International Conference on Machine Learning, 2019.

Model

- **Goal:** Solve (1.1) over a decentralized network
- **Network:** Undirected network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with $|\mathcal{V}| = m$ nodes and $|\mathcal{E}|$ edges
  - Matrix $W = [w_{ij}] \in [0,1]^{m \times m}$ is symmetric doubly stochastic
  - Denote by $\delta > 0$ the spectral gap of W
  - Denoting $\beta = \|I - W\|_2 \in [0, 2]$
- **Heterogeneous Data:** Each node have access to data from potentially different distributions, i.e.,

$$f_i(\boldsymbol{x}) = \mathbb{E}_{\xi_i}[g_i(\boldsymbol{x}, \xi_i)] \text{ with } \xi_i \sim \mathcal{D}_i \ \forall i \in [m]$$

Compression

A general notion of compression operator Q:

Assumption 5 (Compression Operator)

We assume the compression operator $Q : \mathbb{R}^n \to \mathbb{R}^n$ *satisfies for all* $\boldsymbol{x} \in \mathbb{R}^n$

$$\mathbb{E}_Q \|Q(\boldsymbol{x}) - \boldsymbol{x}\|^2 \leq (1 - \omega)\|\boldsymbol{x}\|^2$$

for $\omega > 0$*. Here,* $\mathbb{E}_Q$ *is the expectation over the randomness of* Q

- **QSGD** discussed earlier satisfies Assumption 5
- **Sparsification** satisfies Assumption 5

Choco-SGD: Based on Choco-Gossip

Algorithm: Choco-SGD

- **Initialize:** Initial iterate $\boldsymbol{x}_i^0 \in \mathbb{R}^n \; \forall i \in [m]$, consensus step-size γ, SGD step-size α^k, Graph $\mathcal{G}$, W, initialize $\hat{\boldsymbol{x}}_i^0 = 0 \; \forall i \in [m]$
- **For** $k = 0$ to $K - 1$ **do**
-     Compute stochastic gradient $\boldsymbol{d}_i^k = \nabla g_i(\boldsymbol{x}_i^k, \xi_i^k)$
-     $\boldsymbol{x}_i^{k+\frac{1}{2}} = \boldsymbol{x}_i^k - \alpha^k \boldsymbol{d}_i^k$
-     $\boldsymbol{q}_i^k = Q(\boldsymbol{x}_i^{k+\frac{1}{2}} - \hat{\boldsymbol{x}}_i^k)$
-     Send $\boldsymbol{q}_i^k$ and receive $\boldsymbol{q}_j^k$ from neighbours
-     Maintain: $\hat{\boldsymbol{x}}_j^{k+1} = \boldsymbol{q}_j^k + \hat{\boldsymbol{x}}_j^k$ for all $j \in \mathcal{N}_i$
-     $\boldsymbol{x}_i^{k+1} = \boldsymbol{x}_i^{k+\frac{1}{2}} + \gamma \sum_{j:\{i,j\} \in \mathcal{E}} w_{ij} \left(\hat{\boldsymbol{x}}_j^{k+1} - \hat{\boldsymbol{x}}_i^{k+1} \right)$
- **End For**

Intuition: Choco-SGD

- Note that at every iteration $k \in [K]$ each node $i \in [m]$
  - Maintains a **compressed estimate** of $\boldsymbol{x}_j^k$ denoted as:

$$\hat{\boldsymbol{x}}_j^k \;\; \text{for} \;\; j : \{i, j\} \in \mathcal{E} \;\; (\text{including}\{i\} \in \mathcal{E})$$

  - This is accomplished by receiving **compressed error estimate**

$$\boldsymbol{q}_j^k = Q(\boldsymbol{x}_i^{k+\frac{1}{2}} - \hat{\boldsymbol{x}}_i^k) \;\; \text{for} \;\; j : \{i, j\} \in \mathcal{E} \;\; (\text{including}\{i\} \in \mathcal{E})$$

  at each $k \in [K]$

- **Idea behind Choco-SGD:** Note that averaging the update equation across all $i \in [m]$, yields the standard SGD update

$$\bar{\boldsymbol{x}}^{k+1} = \bar{\boldsymbol{x}}^{k+\frac{1}{2}} = \bar{\boldsymbol{x}}^k - \alpha^k \bar{\boldsymbol{d}}^k$$

where $\bar{\boldsymbol{x}}^k = \frac{1}{m} \sum_{i=1}^m \boldsymbol{x}_i^k$ and $\bar{\boldsymbol{d}}^k$ is defined in a similar fashion

Assumptions

Assumption 6

Each $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i \in [m]$ is assumed to be L-smooth and μ-strongly convex and variance of each node $i \in [m]$ is bounded for all $\boldsymbol{x} \in \mathbb{R}^n$ as

$$\mathbb{E}_{\xi_i} \|\nabla f_i(\boldsymbol{x}, \xi_i) - \nabla f_i(\boldsymbol{x})\|^2 \leq \sigma_i^2$$
$$\mathbb{E}_{\xi_i} \|\nabla f_i(\boldsymbol{x}, \xi_i)\|^2 \leq G^2$$

where $\mathbb{E}_{\xi_i}$ denotes expectation over $\xi_i \sim \mathcal{D}_i$. Moreover, we denote

$$\bar{\sigma}^2 = \frac{1}{m} \sum_{i=1}^{m} \sigma_i^2$$

Main Result

Theorem 1.4

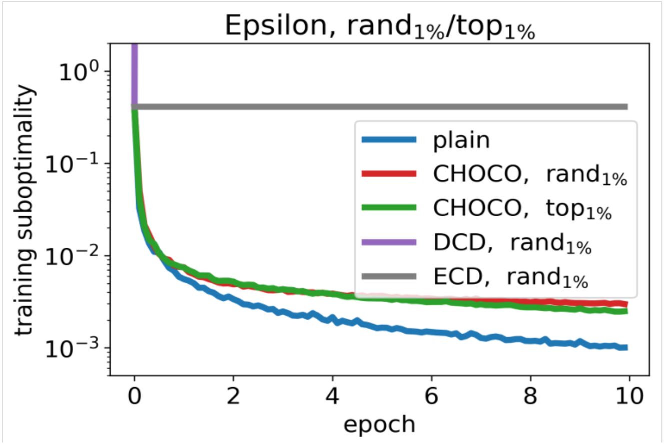
Under Assumptions 6 and 5 the algorithm **Choco-SGD** *with step-sizes* $\alpha^k = \frac{4}{\mu(a+k)}$ *with* $a \geq \max\{410/\delta^2\omega, 16\kappa\}$ *for* $\kappa = L/\mu$ *and* $\gamma = \frac{\delta^2\omega}{16\delta+\delta^2+4\beta^2+2\delta\beta^2-8\delta\omega}$ *converges with a rate*

$$f(\boldsymbol{x}^K_{\textsf{avg}} - f^*) = O\left(\frac{\bar{\sigma}^2}{\mu m K}\right) + O\left(\frac{\kappa G^2}{\mu\omega^2\delta^4 K^2}\right) + O\left(\frac{G^2}{\mu\omega^3\delta^6 K^3}\right)$$

where $\boldsymbol{x}^K_{\textsf{avg}} = \frac{1}{S_K}\sum_{k=0}^{K-1} w_k \bar{\boldsymbol{x}}^k$ *and* $S_K = \sum_{k=0}^{K-1} w_k$

For large K, the term $O\left(\frac{\bar{\sigma}^2}{\mu m K}\right)$ dominates recovering the rate of SGD

Comparison: Choco-SGD vs SGD, ECD and DCD[8]



Comparison of **Choco-SGD** with Plain **SGD** with **ECD-SGD** and **DCD-SGD** proposed in [Tang et al. 2018] with rand$_{1\%}$ sparsification and top$_{1\%}$ for Choco-SGD for RCV1 dataset

[8] Tang et al., Decentralized training over decentralized data, ICML 2018

Extensions

- **Non-Convex functions**[9]
- **Push-Sum + Choco-Gossip**[10]
- **Gradient Descent for Strongly Convex function**[11]

[9] Singh, et al., SPARQ-SGD: Event-Triggered and Compressed Communication in Decentralized Stochastic Optimization." arXiv 2019.

[10] Taheri, et al., Quantized Decentralized Stochastic Learning over Directed Graphs." arXiv 2020.

[11] Liu et al., Linear Convergent Decentralized Optimization with Compression." arXiv 2020.

Optimization in the Presence of Adversaries

The Byzantine Generals Problem[12]

- Adversaries also referred to as **Byzantines**
- Byzantine army communicating only by messages
- One or more generals may be traitors who will try to confuse the other generals

Goal: The algorithm must ensure that the generals agree upon a common battle plan

[12] Lamport et al., The Byzantine Generals Problem", ACM 1982

Optimization in Presence of Byzantines

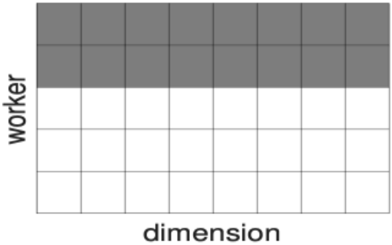
- **Byzantine nodes:** A fraction of multiple WNs might be adversarial
  - Forward **arbitrary** vectors instead of gradients to the server
  - Can collaborate and adversely effect the algorithm's performance

Problem: To design optimization algorithms which are resilient to Byzantine attacks

- **Next:** Types of Byzantine attacks

Classic Byzantine Attacks

- A **fraction of WNs are Byzantines**[13]
  - The set of Byzantine nodes is fixed
- A **fraction of WNs are Byzantines**, however, the set of Byzantine nodes is **not fixed**[14]
  - Nodes can change alliances



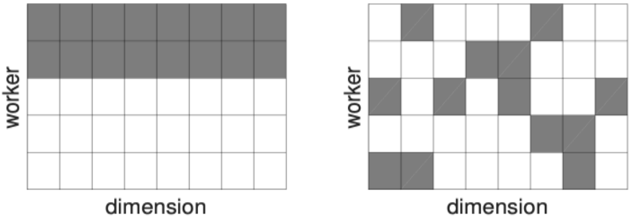
Classical Byzantine Attacks

[13] Alistarh et el., Byzantine Stochastic Gradient Descent, NeurIPS, 2018.

[14] Blanchard et al., Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent, NeurIPS, 2017.

Generalized Byzantine Attacks

- **Dimensional Byzantine Attacks**[15]
  - The attacker only affects certain **dimensions** of a node's vector



(a) Classic Byzantine (b) Generalized Byzantine

Dimension based attacks

[15] Xie et el., Generalized Byzantine-tolerant SGD, arxiv, 2018.

Byzantine SGD[16]

- **Problem:** Solve (1.1) in a distributed fashion
- **Model:**
  - Number of nodes: m out of which β-fraction are Byzantines
  - Nodes interact via a server node (star network)
  - Each node forwards their stochastic gradient to the server and receives a aggregated direction
    - Server **filters** the alleged Byzantine nodes
  - Classical Byzantine attack model
- **Goal:** The algorithm must ensure convergence in the presence of Byzantines!
  - By designing effective filtering rule

[16] Alistarh et al., Byzantine Stochastic Gradient Descent, NeurIPS, 2018.

Notations

- **Homogeneous Data:** Each node have access to data from same distribution, i.e.,

$$f(\boldsymbol{x}) = f_i(\boldsymbol{x}) = \mathbb{E}_{\xi_i}[g_i(\boldsymbol{x}, \xi_i)] \text{ with } \xi_i \sim \mathcal{D} \; \forall i \in [m]$$

- The set of good nodes: $\mathcal{G}$
  - $\mathcal{G}$ is not known to the server
- **Estimate** of the good set at each iteration: $\mathcal{G}^k$ for $k \in [K]$

Assumptions

Assumption 7

For differentiable $f : \mathbb{R}^n \to \mathbb{R}$ we have

- *f is μ-strongly convex*
- *f is L-Lipschitz smooth*
- *f is B-Lipschitz continuous (Gradient of f is bounded)*

At each iteration, each node $i \in [m]$ computes $\nabla_i^k \in \mathbb{R}^n$ as

Assumption 8

At each iteration $k \in [K]$ for every $i \in \mathcal{G}$, we have

- $\nabla_i^k = \nabla g_i(\boldsymbol{x}^k, \xi_i^k)$ *for a random sample* $\xi_i^k \sim \mathcal{D}$
- $\|\nabla_i^k - \nabla f(\boldsymbol{x}^k)\| \leq \mathcal{V}$

For each $k \in [K]$ and $i \notin \mathcal{G}$ the vector ∇_i^k can be adversarially chosen

Filtering Rule

- Let us first define two **statistics** the central node maintains for filtering the Byzantine nodes
  - $A_i = \sum_{t=1}^{k} \langle \nabla_i^k, \boldsymbol{x}^t - \boldsymbol{x}^1 \rangle$
  - $B_i = \sum_{t=1}^{k} \nabla_i^t$
- Note that A_i and B_i accumulate over time
  - The filtering rule relies on the **Martingale concentration**
  - Relies on the fact the for $i \in \mathcal{G}$ and $k \in [K]$ the stochastic gradients ∇_i^k are **chosen independently**
- **Vector Median:** Finally, we define vector median of a set of vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ as any vector $\boldsymbol{v}_i$ such that

$$|\{j \in [m] : \|\boldsymbol{v}_j - \boldsymbol{v}_i\| \leq \mathfrak{T}_v\}| > m/2$$

where $\mathfrak{T}_v > 0$ is the diameter of the norm-ball

Algorithm: Byzantine SGD

Input: Learning rate α, Initial iterate $\boldsymbol{x}^1$, constants $\mathfrak{T}_A$, $\mathfrak{T}_B > 0$

- $\mathcal{G}^1 \leftarrow [m]$
- **For** $k = 1$ to K **do**
- Receive $\nabla_i^k \in \mathbb{R}^n$ from $i \in [m]$
- Maintain statistics A_i and B_i
- Compute $A_{\text{med}} = \text{median}\{A_1, \ldots, A_m\}$
- Compute B_{med} from B_i's with diameter $\mathfrak{T}_B$ (see previous slide)
- Compute ∇_{med} from ∇_i^k,s with diameter $2\mathcal{V}$ (see previous slide)
- Filtering Rule:
$$\mathcal{G}^k \leftarrow \{i \in \mathcal{G}^{k-1} : |A_i - A_{\text{med}}| \le \mathfrak{T}_A \cap \|B_i - B_{\text{med}}\| \le \mathfrak{T}_B \cap \|\nabla_i^k - \nabla_{\text{med}}\| \le 4\mathcal{V}\}$$
- $$\boldsymbol{x}^{k+1} = \operatorname*{argmin}_{\boldsymbol{y}:\|y-\boldsymbol{x}^1\|\le D} \left\{\frac{1}{2}\|\boldsymbol{y} - \boldsymbol{x}^k\|^2 + \alpha\Big\langle \frac{1}{m}\sum_{i\in\mathcal{G}^k} \nabla_i^k, y - \boldsymbol{x}^k \Big\rangle\right\}$$
- **End For**

Main Result

Theorem 1.5

Under Assumption 7 with $\mu = 0$ (convex function) and with β-fraction of Byzantine nodes with $\beta < 1/2$, the ByzantineSGD finds a point $\boldsymbol{x}$ with $f(\boldsymbol{x}) - f^* \leq \epsilon$ in K iterations where

$$K = \tilde{O}\left(\frac{1}{\epsilon} + \frac{1}{\epsilon^2 m} + \frac{\beta^2}{\epsilon^2}\right)$$

or

$$K = \tilde{O}\left(\frac{1}{\mu} + \frac{1}{\mu\epsilon m} + \frac{\beta^2}{\mu\epsilon}\right) \text{ if } f(x) \text{ is } \mu\text{-strongly convex}$$

Note that $\beta = 0$ leads to the guarantees of standard parallel SGD!

Past: Deterministic Optimization Approaches

- **Byzantine Gradient Descent**[17]
  - Classical Byzantine model, Strongly convex functions, Median based aggregation
- **Approximate Gradient Descent**[18]
  - Classical Byzantine model, Strongly convex functions, Direction with most spread
- **Robust Distributed Gradient Descent**[19]
  - Generalized Byzantine model, Strongly convex, convex and non-convex functions, Coordinate wise median and trimmed mean

[17] Chen et al., Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent, Proc. ACM 2017.

[18] Su et al., Securing Distributed Machine Learning in High Dimensions, arxiv 2018.

[19] Yin et al., Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates, ICML 2018.

Past: Stochastic Gradient Descent

- **Krum Based Approaches**[20]
  - Classical Byzantine model, Non-convex functions, Geometric median based aggregation
- **Phocas**[21]
  - Generalized Byzantine model, Non-convex and strongly convex functions, Coordinate wise trimmed mean
- **Robust Gradient Aggregation (RSA)**[22]
  - Classical Byzantine model, Strongly convex functions, Model based aggregation, **Heterogeneous** data

[20] Blanchard et al.,Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent, NeurIPS, 2017

[21] Xie et al., "Phocas: dimensional Byzantine-resilient stochastic gradient descent", arxiv 2018

[22] Li et al., RSA: Byzantine-Robust Stochastic Aggregation Methods for Distributed Learning from Heterogeneous Datasets, AAAI, 2019.

Past: Decentralized Gradient Decent

- **BRIDGE**[23] and **ByRDiE**[24]
  - Generalized Byzantine models, Strongly convex and convex functions resp., Distance based aggregation
- **Recent Survey**[25]

[23] Yang et al., BRIDGE: Byzantine-resilient decentralized gradient descent, arXiv 2019.

[24] Yang et al., ByRDiE: Byzantine-resilient distributed coordinate descent for decentralized learning, IEEE Transactions on Signal and Information Processing over Networks 2019.

[25] Yang et al., Adversary-resilient distributed and decentralized statistical inference and machine learning: An overview of recent advances under the Byzantine threat model, IEEE Signal Processing Magazine 2020.