Tutorial Part – I

Introduction to Robust Machine-Learning

Principles of Distributed Learning

Rafael Pinot & Nirupam Gupta – Oct. 13 2023

EPFL – Distributed Computing Lab

{firstname}.{lastname}@epfl.ch

Based on a joint works with

Youssef Allouah

Sadegh Farhadkhani

Rachid Guerraoui

Nirupam Gupta

John Stephan

Overview of Part - I

SUB-PART 1. Some Reminders on Distributed Machine Learning

1.1 Reminders on supervised machine learning

1.2 Distributed/federated machine learning

1.3 Motivations for trustworthy machine learning

SUB-PART 2. Robustness to Byzantine Nodes (in Homogeneity)

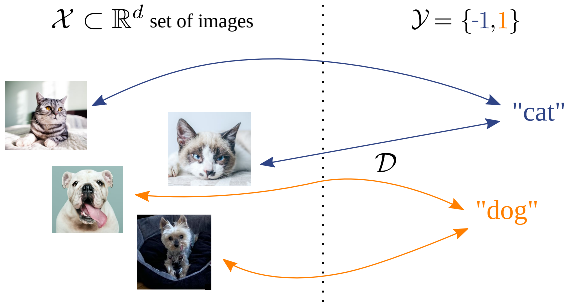
2.1 Brittleness of vanilla methods

2.2 First step towards robustness: aggregation rules

2.3 More advanced tools: noise reduction

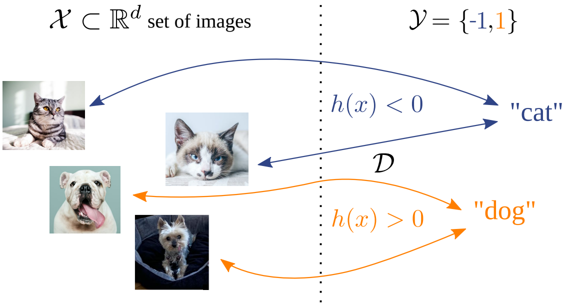
Reminders on Distributed Learning

Supervised Learning (Example of Image Classification)



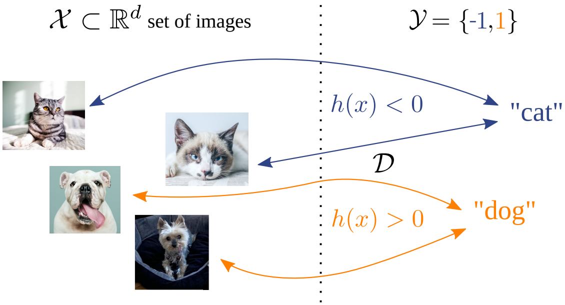
- **Assumption:** A ground-truth distribution $\mathcal{D}$ explains the link between $\mathcal{X}$ and $\mathcal{Y}$

Supervised Learning (Example of Image Classification)



- **Assumption:** A ground-truth distribution $\mathcal{D}$ explains the link between $\mathcal{X}$ and $\mathcal{Y}$
- **Goal:** Use $\mathcal{D}$ to design a mapping $h : \mathcal{X} \rightarrow \mathbb{R}$ matching images $\mathcal{X}$ to labels $\mathcal{Y}$

Supervised Learning (Example of Image Classification)

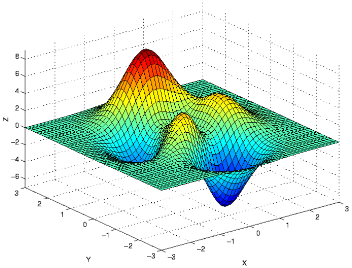


- **Assumption:** A ground-truth distribution $\mathcal{D}$ explains the link between $\mathcal{X}$ and $\mathcal{Y}$
- **Goal:** Use $\mathcal{D}$ to design a mapping $h : \mathcal{X} \to \mathbb{R}$ matching images $\mathcal{X}$ to labels $\mathcal{Y}$

    1) Define a loss function $\ell : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}^+$ and a hypothesis class $\mathcal{H}$

    2) Find $h \in \mathcal{H}$ to minimize the expected error $\mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\ell\left(h(x), y\right)\right]$

Supervised Learning in Practice



- Given a set of m **training examples**:

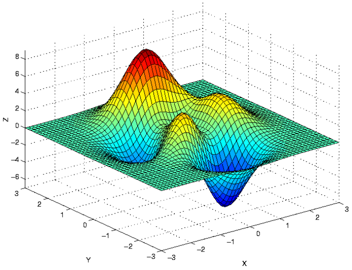
$$\mathcal{S} := \{(x_1, y_1), ..., (x_m, y_m)\} \sim \mathcal{D}^m$$

- Parameterized class $\mathcal{H} := \{h_\theta \mid \theta \in \mathbb{R}^d\}$

- Minimize the **empirical risk**:

$$\mathcal{L}(\theta) := \frac{1}{m} \sum_{i=1}^{m} \ell\left(h_\theta\left(x_i\right), y_i\right)$$

Supervised Learning in Practice



- Given a set of m **training examples**:

$$\mathcal{S} := \{(x_1, y_1), \dots, (x_m, y_m)\} \sim \mathcal{D}^m$$

- Parameterized class $\mathcal{H} := \{h_\theta \mid \theta \in \mathbb{R}^d\}$

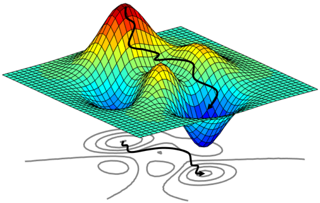
- Minimize the **empirical risk**:

$$\mathcal{L}(\theta) := \frac{1}{m} \sum_{i=1}^{m} \ell\left(h_\theta\left(x_i\right), y_i\right)$$

<u>Learning objective:</u> Assuming $\mathcal{L}$ admits a minimum on $\mathbb{R}^d$, we seek an ε-approximate solution to the empirical risk minimization (ERM), i.e., $\hat{\theta}$ such that

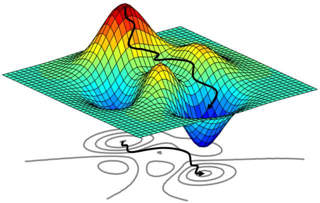
$$\mathcal{L}\left(\hat{\theta}\right) - \mathcal{L}^* \leq \varepsilon, \text{ where } \mathcal{L}^* = \min_{\theta \in \mathbb{R}^d} \mathcal{L}\left(\theta\right).$$

Stochastic Gradient Descent (SGD) in the Centralized Setting



- **Simple** and **efficient** method
- Well understood theoretically
- **Massively used** in practice (especially for deep learning tasks)

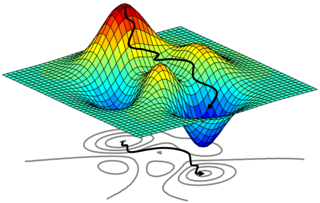
Stochastic Gradient Descent (SGD) in the Centralized Setting



- **Simple** and **efficient** method
- Well understood theoretically
- **Massively used** in practice (especially for deep learning tasks)

- Start with an arbitrary parameter θ_1
- At every step $t = 1, \cdots, T$ do:
  - Sample a data point $(x, y) \sim \text{Unif}(\mathcal{S})$
  - Compute a stochastic gradient $g_t := \nabla_{\theta_t} \ell(y, h_{\theta_t}(x))$
  - Update the parameter $\theta_{t+1} = \theta_t - \gamma g_t$

Stochastic Gradient Descent (SGD) in the Centralized Setting



- **Simple** and **efficient** method
- Well understood theoretically
- **Massively used** in practice
  (especially for deep learning tasks)

Notebook result Bottou et al. (2018): After T iterations of the SGD algorithm, set $\hat{\theta} := \theta_{T+1}$. Then, under reasonable assumptions, $\hat{\theta}$ is an ε-approximate solution to the ERM, with

$$\varepsilon \in \mathcal{O}\left(\frac{\phi(\mathcal{L}, \mathcal{S})}{T}\right)$$

The More Data, The Better

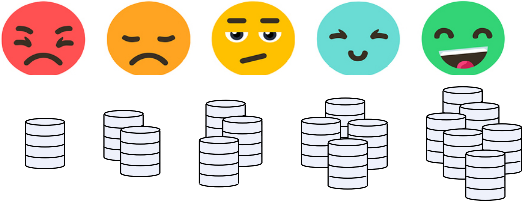
Once we solve the ERM, we wonder how good is $h_{\hat{\theta}}$ w.r.t the “real” risk ?

The More Data, The Better

Once we solve the ERM, we wonder how good is $h_{\hat{\theta}}$ w.r.t the "real" risk ?

Intuitively, **the more data the better**:

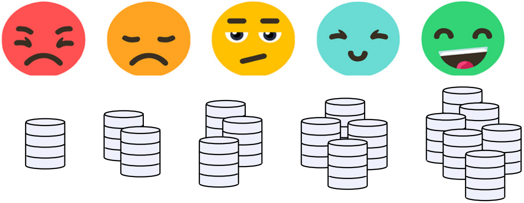
$$\frac{1}{m}\sum_{i=1}^{m}\ell\left(h_{\hat{\theta}}\left(x_i\right),y_i\right)\underset{m\to\infty}{\longrightarrow}\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell\left(h_{\hat{\theta}}(x),y\right)\right]$$



Once we solve the ERM, we wonder how good is $h_{\hat{\theta}}$ w.r.t the "real" risk ?

Intuitively, **the more data the better**:

$$\frac{1}{m}\sum_{i=1}^{m}\ell\left(h_{\hat{\theta}}\left(x_i\right), y_i\right) \underset{m\to\infty}{\longrightarrow} \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell\left(h_{\hat{\theta}}(x), y\right)\right]$$



Even more in **modern day ML** with demanding tasks (e.g. vision or speech)

Federated/Distributed Machine Learning

1. Datacenter distributed learning

→ Train a model on a single massive dataset

→ Distribution **limits computations/storage**

Federated/Distributed Machine Learning

1. Datacenter distributed learning

   → Train a model on a single massive dataset

   → Distribution **limits computations/storage**

2. Cross-silo distributed/federated learning

   → Datacenters are **geo-distributed** by design

   → Keeping data locally is safer

Federated/Distributed Machine Learning

1. <u>Datacenter distributed learning</u>

→ Train a model on a single massive dataset
→ Distribution **limits computations/storage**

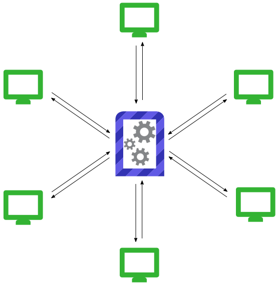
2. <u>Cross-silo distributed/federated learning</u>

→ Datacenters are **geo-distributed** by design
→ Keeping data locally is safer

3. <u>Cross-device distributed/federated learning</u>

→ Same reason for distribution / security requirement
→ **Less computational power** per device
→ More diversity in the data (**heterogeneity**)

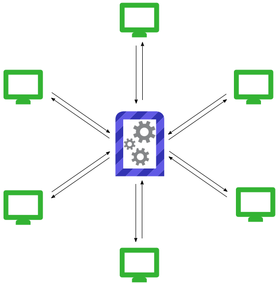
Distributed Machine Learning: Problem Statement



- Server-based communications n computing nodes and a (trusted) central server
- The **nodes** hold the data locally $(\mathcal{S}_i)_{i \in [n]}$

$$\mathcal{L}_i(\theta) := \frac{1}{m} \sum_{(x,y) \in \mathcal{S}_i} \ell\left(h_\theta(x), y\right)$$

- The **server** coordinates the learning



- Server-based communications n computing nodes and a (trusted) central server
- The **nodes** hold the data locally $(\mathcal{S}_i)_{i \in [n]}$

$$\mathcal{L}_i(\theta) := \frac{1}{m} \sum_{(x,y) \in \mathcal{S}_i} \ell\left(h_\theta(x), y\right)$$

- The **server** coordinates the learning

Learning objective: Finding an ε-approximate solution to the ERM for the loss function defined as $\mathcal{L}(\theta) := \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_i(\theta)$

Some Additional Information on the Problem Statement (1/2)

The problem is L-**smooth** and μ-**strongly convex**

Some Additional Information on the Problem Statement (1/2)

The problem is L-**smooth** and μ-**strongly convex**

- $\exists L > 0$ such that for all $\theta,\ \theta' \in \mathbb{R}^d$ and any $(x, y) \sim \mathcal{D}$, the following holds:

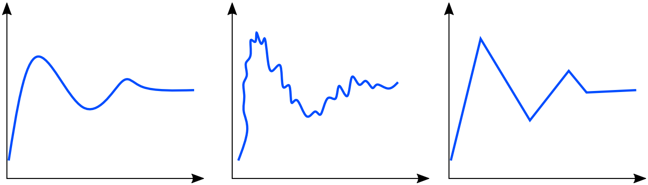
$$\|\nabla \ell(h_\theta(x), y) - \nabla \ell(h_{\theta'}(x), y)\| \leq L \left\| \theta - \theta' \right\|.$$

Some Additional Information on the Problem Statement (1/2)

The problem is L-**smooth** and μ-**strongly convex**

- $\exists L > 0$ such that for all $\theta,\ \theta' \in \mathbb{R}^d$ and any $(x, y) \sim \mathcal{D}$, the following holds:

$$\|\nabla \ell(h_\theta(x), y) - \nabla \ell(h_{\theta'}(x), y)\| \leq L \left\| \theta - \theta' \right\|.$$

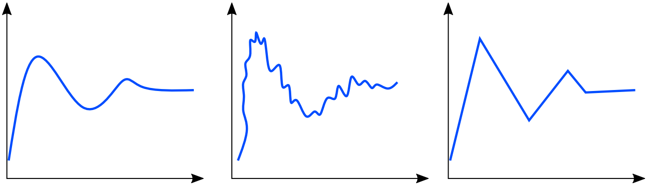


Some Additional Information on the Problem Statement (1/2)

The problem is L-**smooth** and μ-**strongly convex**

- $\exists L > 0$ such that for all $\theta,\ \theta' \in \mathbb{R}^d$ and any $(x, y) \sim \mathcal{D}$, the following holds:

$$\|\nabla \ell(h_\theta(x), y) - \nabla \ell(h_{\theta'}(x), y)\| \leq L \big\|\theta - \theta'\big\|.$$



- $\exists \mu > 0$ such that for all $\theta \in \mathbb{R}^d$, we have

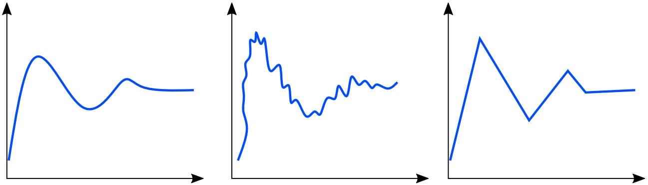
$$\|\nabla \mathcal{L}(\theta)\|^2 \geq 2\mu \left(\mathcal{L}(\theta) - \mathcal{L}^*\right) \quad \text{(Polyak's inequality)}$$

Some Additional Information on the Problem Statement (1/2)

The problem is L-**smooth** and μ-**strongly convex**

- $\exists L > 0$ such that for all $\theta,\ \theta' \in \mathbb{R}^d$ and any $(x, y) \sim \mathcal{D}$, the following holds:

$$\|\nabla \ell(h_\theta(x), y) - \nabla \ell(h_{\theta'}(x), y)\| \leq L \left\|\theta - \theta'\right\|.$$



- $\exists \mu > 0$ such that for all $\theta \in \mathbb{R}^d$, we have

$$\|\nabla \mathcal{L}(\theta)\|^2 \geq 2\mu \left(\mathcal{L}(\theta) - \mathcal{L}^*\right) \quad \text{(Polyak's inequality)}$$

→ **Numerical examples neural-network for image classification**

Some Additional Information on the Problem Statement (2/2)

- All local datasets have the same size m (everything can be adapted)
  $\rightarrow$ We make this assumption, just for simplicity

- **Homogeneity** of the datasets, i.e., $\mathcal{S}_i = \mathcal{S}_j$, for all $i, j \in [n]$
  $\rightarrow$ This is just for my talk, Nirupam will remove this assumption

Some Additional Information on the Problem Statement (2/2)

- All local datasets have the same size m (everything can be adapted)
  $\rightarrow$ We make this assumption, just for simplicity

- **Homogeneity** of the datasets, i.e., $\mathcal{S}_i = \mathcal{S}_j$, for all $i, j \in [n]$
  $\rightarrow$ This is just for my talk, Nirupam will remove this assumption

Bounded stochasticity: There exists $\sigma \geq 0$ such that for all $i \in [n]$ and $\theta \in \mathbb{R}^d$,

$$\frac{1}{m} \sum_{(x,y) \in \mathcal{S}_i} \left\| \nabla \ell \left(h_\theta(x), y \right) - \nabla \mathcal{L}_i(\theta) \right\|^2 \leq \sigma^2$$

Some Additional Information on the Problem Statement (2/2)

- All local datasets have the same size m (everything can be adapted)
  $\rightarrow$ We make this assumption, just for simplicity

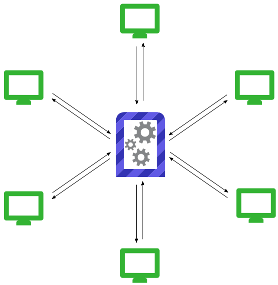
- **Homogeneity** of the datasets, i.e., $\mathcal{S}_i = \mathcal{S}_j$, for all $i, j \in [n]$
  $\rightarrow$ This is just for my talk, Nirupam will remove this assumption

Bounded stochasticity: There exists $\sigma \geq 0$ such that for all $i \in [n]$ and $\theta \in \mathbb{R}^d$,

$$\frac{1}{m} \sum_{(x,y) \in \mathcal{S}_i} \left\| \nabla \ell \left(h_\theta(x), y \right) - \nabla \mathcal{L}_i(\theta) \right\|^2 \leq \sigma^2$$

$\rightarrow$ Every time I sample a point at random in $\mathcal{S}_i$ the gradient estimate has a bounded expected $\|\cdot\|^2$ to the real gradient

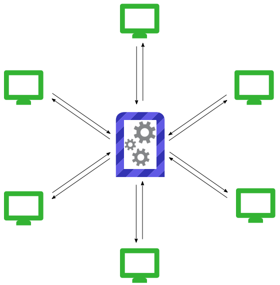
Distributed Stochastic Gradient Descent (DSGD)



Initialize the model at θ_1, then at every step $t = 1, \ldots, T$

- **node** i computes & sends
  $$g_t^{(i)} = \nabla_{\theta_t} \ell\left(y_i, h_{\theta_t}(x_i)\right),$$
  where $(x_i, y_i) \sim \mathcal{S}_i$.
- **Server** updates & broadcast
  $$\theta_{t+1} = \theta_t - \gamma \frac{1}{n} \sum_{i=1}^{n} g_t^{(i)}$$

Distributed Stochastic Gradient Descent (DSGD)



Initialize the model at θ_1, then at every step $t = 1, \dots, T$

- **node** i computes & sends

$$g_t^{(i)} = \nabla_{\theta_t} \ell\left(y_i, h_{\theta_t}(x_i)\right),$$

where $(x_i, y_i) \sim \mathcal{S}_i$.

- **Server** updates & broadcast

$$\theta_{t+1} = \theta_t - \gamma \frac{1}{n} \sum_{i=1}^{n} g_t^{(i)}$$

Textbook result Bertsekas and Tsitsiklis (2015):

Set $\hat{\theta} := \theta_{T+1}$. Then $\hat{\theta}$ is an ε-approximate solution to the ERM, with

$$\varepsilon \in \mathcal{O}\left(\frac{\mathcal{K}_{\mathcal{L}}\sigma^2}{nT}\right), \text{ and } \mathcal{K}_{\mathcal{L}} := \frac{L}{\mu}.$$

Machine learning models are everywhere

- Machine learning models recently gave **outstanding results** (*e.g.* vision, NLP)
- Industries and governments are starting to use them in **critical applications**

Machine learning models are everywhere

- Machine learning models recently gave **outstanding results** (*e.g.* vision, NLP)
- Industries and governments are starting to use them in **critical applications**

Personal Devices

Machine learning models are everywhere

- Machine learning models recently gave **outstanding results** (*e.g.* vision, NLP)
- Industries and governments are starting to use them in **critical applications**

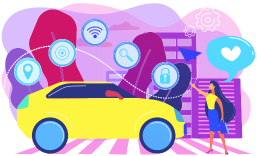
Decision-making Tools

Personal Devices

Machine learning models are everywhere

- Machine learning models recently gave **outstanding results** (*e.g.* vision, NLP)
- Industries and governments are starting to use them in **critical applications**

AI-driven Technologies



Decision-making Tools

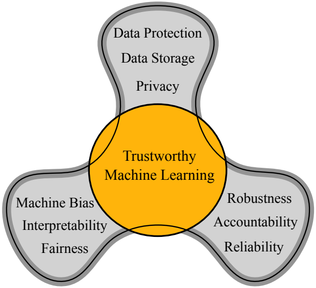
Personal Devices

With Great Power Comes Great Responsibility

But great power comes great responsibility ...

With Great Power Comes Great Responsibility

But great power comes great responsibility ...



- Massive use of learning algorithms raises **societal** and technical issues

- **Since the 80's:** privacy preserving database analysis is a primary concern

- Some more recent and specific to Federated Learning (**Byzantine failures**)

Robustness to Byzantine Nodes

Misbehaving Nodes Are Inevitable

In practice, nodes may misbehave at any point during the learning

Misbehaving Nodes Are Inevitable

In practice, nodes may misbehave at any point during the learning

- **Software bugs** can occur and add errors in the computations

Misbehaving Nodes Are Inevitable

In practice, nodes may misbehave at any point during the learning

- **Software bugs** can occur and add errors in the computations
- **Hardware crashes** also happen, provoking machine crashes and/or errors

Misbehaving Nodes Are Inevitable

In practice, nodes may misbehave at any point during the learning

- **Software bugs** can occur and add errors in the computations
- **Hardware crashes** also happen, provoking machine crashes and/or errors
- Some nodes may have **poisoned or irrelevant** data

Misbehaving Nodes Are Inevitable

In practice, nodes may misbehave at any point during the learning

- **Software bugs** can occur and add errors in the computations
- **Hardware crashes** also happen, provoking machine crashes and/or errors
- Some nodes may have **poisoned or irrelevant** data
- Some machines can get **hacked** by external adversary

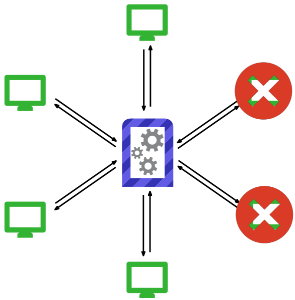
Misbehaving Nodes Are Inevitable

In practice, nodes may misbehave at any point during the learning

- **Software bugs** can occur and add errors in the computations
- **Hardware crashes** also happen, provoking machine crashes and/or errors
- Some nodes may have **poisoned or irrelevant** data
- Some machines can get **hacked** by external adversary

Challenge: We do not know a priori which nodes may misbehave (nor how)

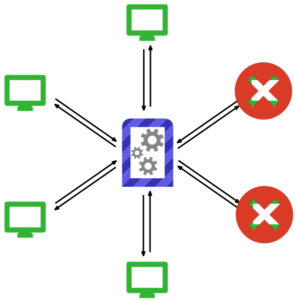
The Byzantine Threat Model



- We consider the **Byzantine threat model** inherited from Lamport et al. (1982)
- Up to $f < n/2$ nodes may be bad
- When i is Byzantine we have

$$g_t^{(i)} = *, \ \forall t \in [T]$$

The Byzantine Threat Model



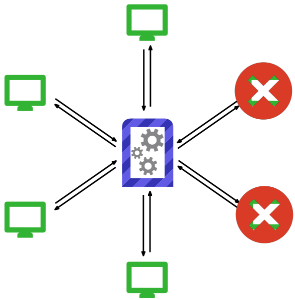
- We consider the **Byzantine threat model** inherited from Lamport et al. (1982)
- Up to $f < n/2$ nodes may be bad
- When i is Byzantine we have

$$g_t^{(i)} = *, \; \forall t \in [T]$$

New objective: Denote H the set of honest (non-Byzantine) nodes. We seek an ε-approximate solution to the ERM for the loss function defined as

$$\mathcal{L}_H(\theta) := \frac{1}{n-f} \sum_{i \in H} \mathcal{L}_i(\theta) \quad \text{(a.k.a. } (f, \varepsilon)\text{-Byzantine resilience)}$$

The Byzantine Threat Model



- We consider the **Byzantine threat model** inherited from Lamport et al. (1982)
- Up to $f < n/2$ nodes may be bad
- When i is Byzantine we have

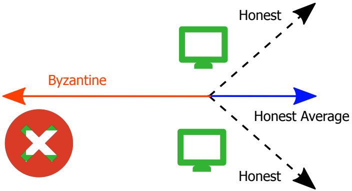
$$g_t^{(i)} = *, \; \forall t \in [T]$$

New objective: Denote H the set of honest (non-Byzantine) nodes. We seek an ε-approximate solution to the ERM for the loss function defined as

$$\mathcal{L}_H(\theta) := \frac{1}{n-f} \sum_{i \in H} \mathcal{L}_i(\theta) \quad \text{(a.k.a. } (f, \varepsilon)\text{-Byzantine resilience)}$$

$\rightarrow$ **Despite the f Byzantine players (and not knowing H a priori)**

Is DSGD Byzantine Robust?

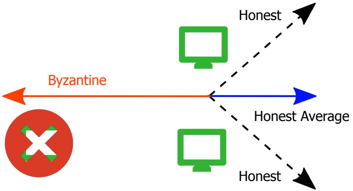


Recall update rule at the server:

$$\theta_{t+1} = \theta_t - \gamma \frac{1}{n} \sum_{i=1}^{n} g_t^{(i)}$$

The average is arbitrarily manipulable by a **single** Byzantine node.

Is DSGD Byzantine Robust?



Recall update rule at the server:

$$\theta_{t+1} = \theta_t - \gamma \frac{1}{n} \sum_{i=1}^{n} g_t^{(i)}$$

The average is arbitrarily manipulable by a **single** Byzantine node.

A standard approach to confer Byzantine robustness:

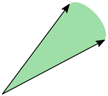
Replace the simple averaging with a **non-linear** aggregation rule $F : \mathbb{R}^{d \times n} \to \mathbb{R}^d$:

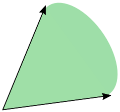
$$\theta_{t+1} = \theta_t - \gamma \, F\left(g_t^{(1)}, \ldots, g_t^{(n)}\right)$$

$\rightarrow$ Choosing F is close to the **robust mean estimation** problem

One of the main challenges is uncertainty:

$$\mathbb{E}\left[\left\|g_t^{(i)} - \mathbb{E}\left[g_t^{(i)}\right]\right\|^2\right] \leq \sigma^2$$

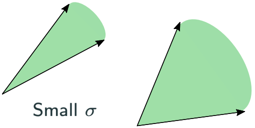

Small σ


Big σ

Range of plausible gradients for an honest worker

One of the main challenges is uncertainty:

$$\mathbb{E}\left[\left\|g_t^{(i)} - \mathbb{E}\left[g_t^{(i)}\right]\right\|^2\right] \leq \sigma^2$$



Range of plausible gradients for an honest worker

Essentially: the bigger σ, the harder it is to defend against Byzantine nodes

If $\sigma^2 = 0$ (i.e., gradient descent) $\rightarrow$ **F = Majority voting**.	When $\sigma^2 > 0$ (i.e., SGD) $\rightarrow$ **F = ?**

Some Famous Aggregation Rules 1/2

Simple coordinate-wise solutions ($n = 5$, $f = 1$, $d = 2$):

Coordinate-wise median (CW-Med)

→ Compute the median on each coordinates.

$$\text{CW-Med}\begin{pmatrix} 3 & 1 & 3 & 6 & 8 \\ 6 & 2 & 4 & 3 & 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

Simple coordinate-wise solutions ($n = 5,\ f = 1,\ d = 2$):

Coordinate-wise median (CW-Med)

→ Compute the median on each coordinates.

$$\text{CW-Med}\begin{pmatrix} 3 & 1 & \textcircled{3} & 6 & 8 \\ 6 & 2 & 4 & \textcircled{3} & 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

Coordinate-wise trimmed mean (CW-TM)

→ Remove f biggest and f smallest coordinates on each dimension, and then average.

$$\text{CW-TM}\begin{pmatrix} 3 & \cancel{1} & 3 & 6 & \cancel{8} \\ \cancel{6} & 2 & 4 & 3 & \cancel{1} \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \end{pmatrix}$$

Some Famous Aggregation Rules 1/2

Simple coordinate-wise solutions ($n = 5$, $f = 1$, $d = 2$):

Coordinate-wise median (CW-Med)

→ Compute the median on each coordinates.

$$\text{CW-Med}\begin{pmatrix} 3 & 1 & 3 & 6 & 8 \\ 6 & 2 & 4 & 3 & 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

Coordinate-wise trimmed mean (CW-TM)

→ Remove f biggest and f smallest coordinates on each dimension, and then average.

$$\text{CW-TM}\begin{pmatrix} 3 & \cancel{1} & 3 & 6 & \cancel{8} \\ \cancel{6} & 2 & 4 & 3 & \cancel{1} \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \end{pmatrix}$$

Both these solutions have been analyzed, e.g., in Yin et al. (2018).

Some Famous Aggregation Rules 2/2

More sophisticated aggregations:

Geometric median Chen et al. (2017)

→ Output a vector that realizes the geometric median of the send gradients, i.e.,

$$\mathrm{GM}\left(v_1, \dots, v_n\right) \in \mathrm{argmin}_{v \in \mathbb{R}^d} \sum_{i=1}^{n} \|v - v_i\|.$$

Some Famous Aggregation Rules 2/2

More sophisticated aggregations:

Geometric median Chen et al. (2017)

→ Output a vector that realizes the geometric median of the send gradients, i.e.,

$$\mathrm{GM}\left(v_1, \ldots, v_n\right) \in \operatorname{argmin}_{v \in \mathbb{R}^d} \sum_{i=1}^{n} \|v - v_i\|.$$

MDA Rousseeuw (1985)

→ Choose a set of indices S^* with cardinality $n - f$ and with the smallest *diameter*. Then average over S^*, i.e.,

$$\mathrm{MDA}\left(v_1, \ldots, v_n\right) = \frac{1}{n-f} \sum_{i \in S^*} v_i.$$

Some Famous Aggregation Rules 2/2

More sophisticated aggregations:

Geometric median Chen et al. (2017)
→ Output a vector that realizes the geometric median of the send gradients, i.e.,

$$\mathrm{GM}\left(v_1, \ldots, v_n\right) \in \operatorname{argmin}_{v \in \mathbb{R}^d} \sum_{i=1}^{n} \|v - v_i\|.$$

MDA Rousseeuw (1985)
→ Choose a set of indices S^* with cardinality $n - f$ and with the smallest *diameter*. Then average over S^*, i.e.,

$$\mathrm{MDA}\left(v_1, \ldots, v_n\right) = \frac{1}{n-f} \sum_{i \in S^*} v_i.$$

But also **MeaMed** Xie et al. (2018), **Krum, Multi-Krum** Blanchard et al. (2017) ...

A First Step Towards Resilience

Common notion of (f, κ)-robust averaging:

For any set of n vectors $v_1, \dots, v_n \in \mathbb{R}^d$ and any subset $S \subseteq [n]$ of size $n - f$,

$$\left\| F\left(v_1, \dots, v_n\right) - \overline{v}_S \right\|^2 \leq \frac{\kappa}{n-f} \sum_{i \in S} \left\| v_i - \overline{v}_S \right\|^2,$$

where $\overline{v}_S := \frac{1}{n-f} \sum_{i \in S} v_i$

A First Step Towards Resilience

Common notion of (f, κ)-robust averaging:

For any set of n vectors $v_1, \dots, v_n \in \mathbb{R}^d$ and any subset $S \subseteq [n]$ of size $n - f$,

$$\left\| F\left(v_1, \dots, v_n\right) - \overline{v}_S \right\|^2 \leq \frac{\kappa}{n-f} \sum_{i \in S} \left\| v_i - \overline{v}_S \right\|^2,$$

where $\overline{v}_S := \frac{1}{n-f} \sum_{i \in S} v_i$

Rationale: Ensure that the distance between the result of the aggregation rule and the average of honest workers' inputs is bounded by their *variance* times a factor κ.

A First Step Towards Resilience

Common notion of (f, κ)-robust averaging:

For any set of n vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ and any subset $S \subseteq [n]$ of size $n - f$,

$$\left\| F\left(v_1, \ldots, v_n\right) - \overline{v}_S \right\|^2 \leq \frac{\kappa}{n-f} \sum_{i \in S} \left\| v_i - \overline{v}_S \right\|^2,$$

where $\overline{v}_S := \frac{1}{n-f} \sum_{i \in S} v_i$

Rationale: Ensure that the distance between the result of the aggregation rule and the average of honest workers' inputs is bounded by their *variance* times a factor κ.

Quick sanity check: If $\sigma^2 = 0$ the honest workers are identical (full gradients)

$$\sum_{i \in S} \left\| v_i - \overline{v}_S \right\|^2 = 0$$

$\rightarrow$ The aggregation rule should mimic the majority voting scheme.

A First Step Towards Resilience

Common notion of (f, κ)-robust averaging:

For any set of n vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ and any subset $S \subseteq [n]$ of size $n - f$,

$$\left\| F\left(v_1, \ldots, v_n\right) - \overline{v}_S \right\|^2 \leq \frac{\kappa}{n-f} \sum_{i \in S} \left\| v_i - \overline{v}_S \right\|^2,$$

where $\overline{v}_S := \frac{1}{n-f} \sum_{i \in S} v_i$

$\rightarrow$ This definition is satisfied by many existing aggregation rules.

A First Step Towards Resilience

Common notion of (f, κ)-robust averaging:

For any set of n vectors $v_1, \dots, v_n \in \mathbb{R}^d$ and any subset $S \subseteq [n]$ of size $n - f$,

$$\left\|F\left(v_1, \dots, v_n\right) - \overline{v}_S\right\|^2 \leq \frac{\kappa}{n-f} \sum_{i \in S} \left\|v_i - \overline{v}_S\right\|^2,$$

where $\overline{v}_S := \frac{1}{n-f} \sum_{i \in S} v_i$

→ This definition is satisfied by many existing aggregation rules.

Agg.	CW-TM	GM	CW-Med	L.B.
κ	$\mathcal{O}\left(\frac{f}{n-2f}\right)$	$\mathcal{O}\left(1 + \frac{f}{n-2f}\right)$	$\mathcal{O}\left(1 + \frac{f}{n-2f}\right)$	$\Omega\left(\frac{f}{n-2f}\right)$

Applies to **Krum**, **Multi-Krum** Blanchard et al. (2017) and **MeaMed** Xie et al. (2018).

A First Step Towards Resilience

Common notion of (f, κ)-robust averaging:

For any set of n vectors $v_1, \dots, v_n \in \mathbb{R}^d$ and any subset $S \subseteq [n]$ of size $n - f$,

$$\left\| F\left(v_1, \dots, v_n\right) - \overline{v}_S \right\|^2 \leq \frac{\kappa}{n-f} \sum_{i \in S} \left\| v_i - \overline{v}_S \right\|^2,$$

where $\overline{v}_S := \frac{1}{n-f} \sum_{i \in S} v_i$

Convergence result in the homogeneous case:

A First Step Towards Resilience

Common notion of (f, κ)-robust averaging:

For any set of n vectors $v_1, \dots, v_n \in \mathbb{R}^d$ and any subset $S \subseteq [n]$ of size $n - f$,

$$\|F(v_1, \dots, v_n) - \overline{v}_S\|^2 \leq \frac{\kappa}{n-f} \sum_{i \in S} \|v_i - \overline{v}_S\|^2,$$

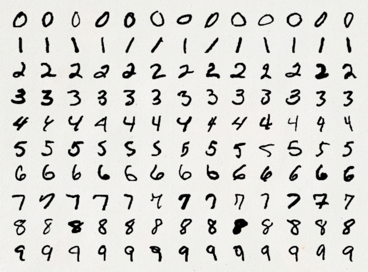
where $\overline{v}_S := \frac{1}{n-f} \sum_{i \in S} v_i$

Convergence result in the homogeneous case:

If F is an (f, κ)-robust averaging, setting $\hat{\theta} := \theta_{T+1}$, the algorithm satisfies (f, ε)-Byzantine resilience with

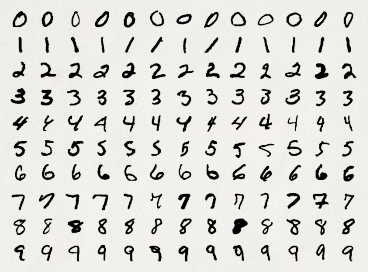
$$\varepsilon \in \mathcal{O}\left(\frac{\mathcal{K}_{\mathcal{L}_H} \sigma^2}{(n-f)T} + \kappa\sigma^2\right)$$

Learning task: MNIST hand-written digit image classification task with $n = 15$ nodes out of which $f = 5$ might be Byzantine.



Some Numerical Observations: Model Setting

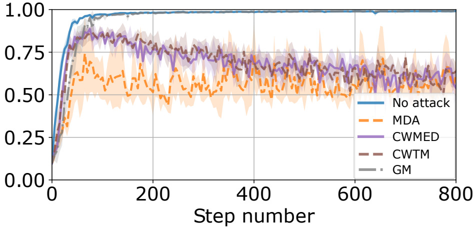
Learning task: MNIST hand-written digit image classification task with $n = 15$ nodes out of which $f = 5$ might be Byzantine.

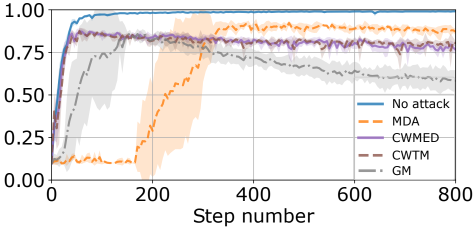


Adversarial behaviors: The Byzantine nodes apply either of the following:

- *Label-flipping*: shift the label of each image **0123456789** $\rightarrow$ **1234567890**
- *Sign-flipping*: send the inverse of the local gradient $\mathbf{g}_t^{(i)} \rightarrow -\mathbf{g}_t^{(i)}$

Training accuracy of a CNN along the learning procedure on MNIST. On the **left** *label-flipping* attack and on the **right** *sign-flipping* attack.

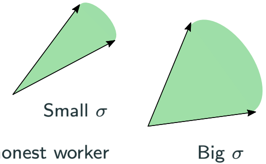




Why is There Still a Gap?

Recall the challenge of uncertainty:

$$\mathbb{E}\left[\left\|g_t^{(i)} - \mathbb{E}\left[g_t^{(i)}\right]\right\|^2\right] \leq \sigma^2$$

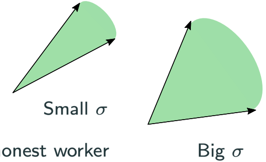


Range of plausible gradients for an honest worker

Why is There Still a Gap?

Recall the challenge of uncertainty:

$$\mathbb{E}\left[\left\|g_t^{(i)} - \mathbb{E}\left[g_t^{(i)}\right]\right\|^2\right] \leq \sigma^2$$

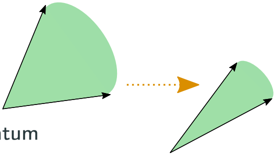


Range of plausible gradients for an honest worker

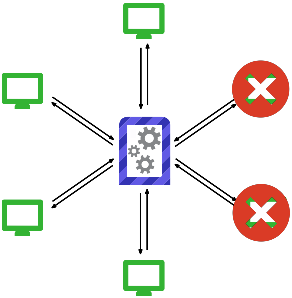
Main idea: Let us get the uncertainty smaller to obtain better convergence!

Option 1: Use mini-batch SGD

Option 2: Use distributed momentum



Option 1: Mini-batch SGD & Dynamic Sampling



→ **Each honest worker** i

- Samples a batch of b_t points

$$B_t^{(i)} = \{(x_1, y_1), \ldots, (x_{b_t}, y_{b_t})\}$$

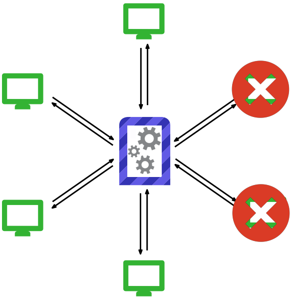
- Computes and send

$$g_t^{(i)} = \frac{1}{b_t} \sum_{(x,y) \in B_t^{(i)}} \nabla \ell(h_{\theta_t}(x), y)$$

→ **Server** updates & broadcasts

$$\theta_{t+1} = \theta_t - \gamma F\left(g_t^{(1)}, \ldots, g_t^{(n)}\right)$$

Option 1: Mini-batch SGD & Dynamic Sampling



→ **Each honest worker** i

- Samples a batch of b_t points

$$B_t^{(i)} = \{(x_1, y_1), \ldots, (x_{b_t}, y_{b_t})\}$$

- Computes and send

$$g_t^{(i)} = \frac{1}{b_t} \sum_{(x,y) \in B_t^{(i)}} \nabla \ell(h_{\theta_t}(x), y)$$

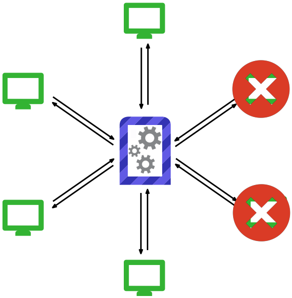
→ **Server** updates & broadcasts

$$\theta_{t+1} = \theta_t - \gamma F\left(g_t^{(1)}, \ldots, g_t^{(n)}\right)$$

Pros and cons of the method:

- Reduces the noise at every step $\sigma^2 \rightarrow \frac{\sigma^2}{b_t}$
- Need large b_t to work which inflates the computationnal cost of the method

Controlling Uncertainty Drift with Momentum



- **Honest node** i computes

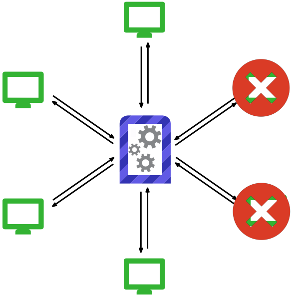
$$m_t^{(i)} = \beta m_{t-1}^{(i)} + (1-\beta) g_t^{(i)},$$

where $m_0^{(i)} = 0$ and $\beta \in [0,\, 1)$.

- **Server** updates & broadcast

$$\theta_{t+1} = \theta_t - \gamma F\left(m_t^{(1)}, \dots, m_t^{(n)}\right)$$

Controlling Uncertainty Drift with Momentum



- **Honest node** i computes

$$m_t^{(i)} = \beta m_{t-1}^{(i)} + (1-\beta) g_t^{(i)},$$

where $m_0^{(i)} = 0$ and $\beta \in [0, 1)$.

- **Server** updates & broadcast

$$\theta_{t+1} = \theta_t - \gamma F\left(m_t^{(1)}, \dots, m_t^{(n)}\right)$$

Pros and cons of the method:

- Initially introduced to accelerate the learning when $\sigma^2 = 0$, see Polyak (1964)
- Can also be used to **control uncertainty** when $\sigma^2 > 0$
- Way harder to analyze (momentum drift vs noise reduction)

Option 2: Controlling Uncertainty with Momentum

Key ingredient of the analysis: In short, we have that

$$\mathbb{E}\left[\left\|m_t^{(i)} - \overline{m}_t\right\|^2\right] \in \mathcal{O}\left(\sigma^2 \ (1-\beta)\right), \text{ where } \overline{m}_t := \frac{1}{(n-f)} \sum_{i \in H} m_t^{(i)}.$$

Then choosing the right momentum coefficient β (essentially) yields the result

$\rightarrow$ Eluding quite some technicalities (especially on the momentum drift)

Option 2: Controlling Uncertainty with Momentum

Key ingredient of the analysis: In short, we have that

$$\mathbb{E}\left[\left\|m_t^{(i)} - \overline{m}_t\right\|^2\right] \in \mathcal{O}\left(\sigma^2 \, (1-\beta)\right), \text{ where } \overline{m}_t := \frac{1}{(n-f)} \sum_{i \in H} m_t^{(i)}.$$

Then choosing the right momentum coefficient β (essentially) yields the result

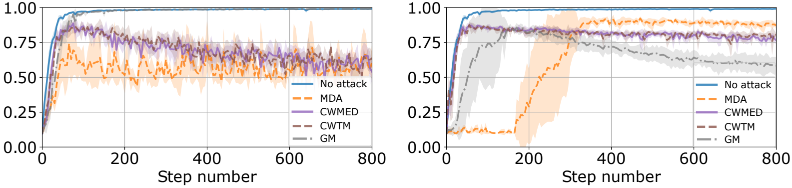
$\rightarrow$ Eluding quite some technicalities (especially on the momentum drift)

Convergence result in the homogeneous case Farhadkhani et al. (2022, 2023):

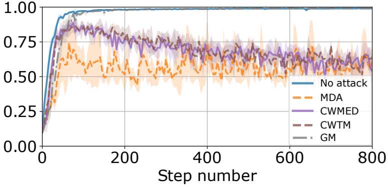
Assumes F is an (f, κ)-robust averaging and β is chosen "well-enough". Then setting $\hat{\theta} := \theta_{T+1}$, the algorithm satisfies (f, ε)-Byzantine resilience with

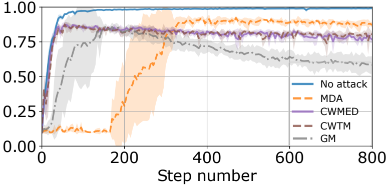
$$\varepsilon \in \mathcal{O}\left(\left(\kappa + \frac{1}{(n-f)}\right) \frac{\mathcal{K}_{\mathcal{L}_H} \sigma^2}{T}\right)$$

Impact of the Momentum on Byzantine Resilience

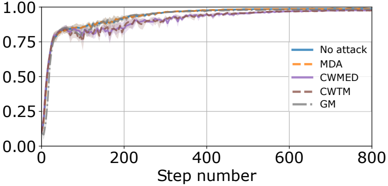


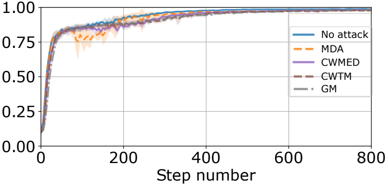
Same setting as before. **Up** without momentum ($\beta = 0$)





Same setting as before. **Up** without momentum ($\beta = 0$) and **down** with momentum ($\beta = 0.99$)





Take-home messages

Concluding Remarks

General take-home messages:

- Distributed Learning unlocks the use of machine learning in high-stack applications and is driving research forward...
- ... but standard distributed learning solutions are not robust to misbehaving nodes (a.k.a. Byzantine)

Concluding Remarks

General take-home messages:

- Distributed Learning unlocks the use of machine learning in high-stack applications and is driving research forward...
- ... but standard distributed learning solutions are not robust to misbehaving nodes (a.k.a. Byzantine)

Technical solutions:

- First of all, use robust aggregation rules (median, trimmed mean, etc.).
- Second, reduce the noise of the stochastic gradients (e.g., using momentum).

Concluding Remarks

General take-home messages:

- Distributed Learning unlocks the use of machine learning in high-stack applications and is driving research forward...
- ... but standard distributed learning solutions are not robust to misbehaving nodes (a.k.a. Byzantine)

Technical solutions:

- First of all, use robust aggregation rules (median, trimmed mean, etc.).
- Second, reduce the noise of the stochastic gradients (e.g., using momentum).

Future directions:

- Understand the impact of data heterogeneity in (robust) federated learning
- How to combine robustness with other concerns (**privacy**, fairness, bias, etc.)

$\rightarrow$ **Nirupam's talk**

Thanks for listening!

References

Bertsekas, D. and Tsitsiklis, J. (2015). *Parallel and distributed computation: numerical methods*. Athena Scientific.

Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 119–129. Curran Associates, Inc.

Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311.

Chen, Y., Su, L., and Xu, J. (2017). Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):1–25.

Farhadkhani, S., Guerraoui, R., Gupta, N., Hoang, L.-N., Pinot, R., and Stephan, J. (2023). Robust collaborative learning with linear gradient overhead.

Farhadkhani, S., Guerraoui, R., Gupta, N., Pinot, R., and Stephan, J. (2022). Byzantine machine learning made easy by resilient averaging of momentums. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S., editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 6246–6283. PMLR.

Lamport, L., Shostak, R., and Pease, M. (1982). The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401.

Polyak, B. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4:1–17.

Rousseeuw, P. J. (1985). Multivariate estimation with high breakdown point. *Mathematical statistics and applications*, 8(37):283–297.

Xie, C., Koyejo, O., and Gupta, I. (2018). Generalized byzantine-tolerant sgd.

Yin, D., Chen, Y., Kannan, R., and Bartlett, P. (2018). Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. PMLR.