Byzantine Tolerance for Distributed SGD

Presenter: Cong Xie
Advisors: Oluwasanmi Koyejo, Indranil Gupta

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Outline

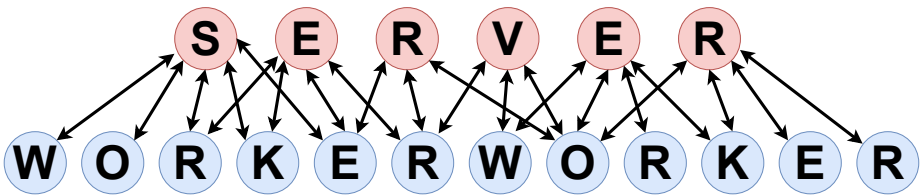
Distributed machine learning

Train ML models on multiple nodes:

Distributed machine learning

Train ML models on multiple nodes:

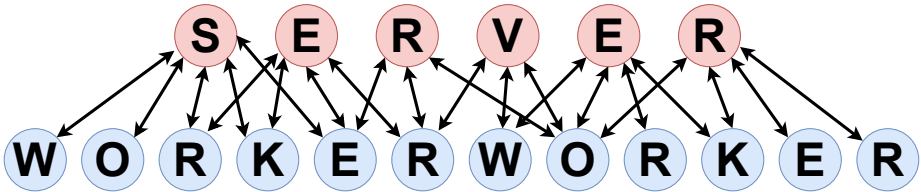
- Server-worker architecture



Distributed machine learning

Train ML models on multiple nodes:

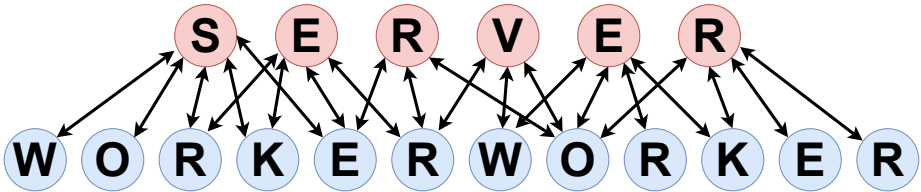
- Server-worker architecture
- Communication



Distributed machine learning

Train ML models on multiple nodes:

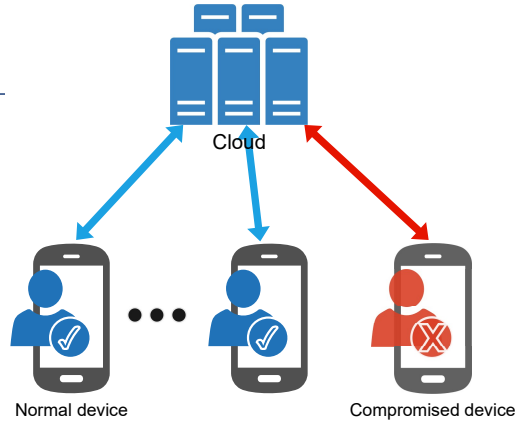
- Server-worker architecture
- Communication
- Security



Outline

Security in distributed ML

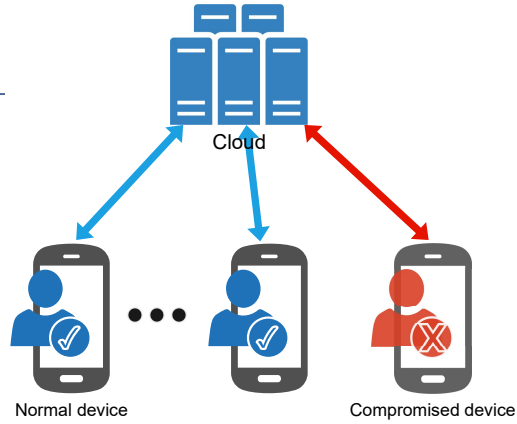
Untrustable workers:



Security in distributed ML

Untrustable workers:

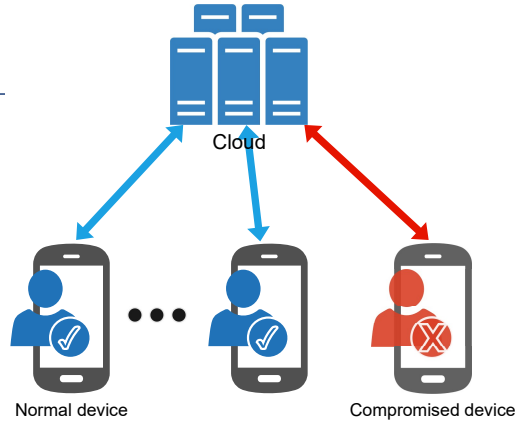
- Hardware/software failures



Security in distributed ML

Untrustable workers:

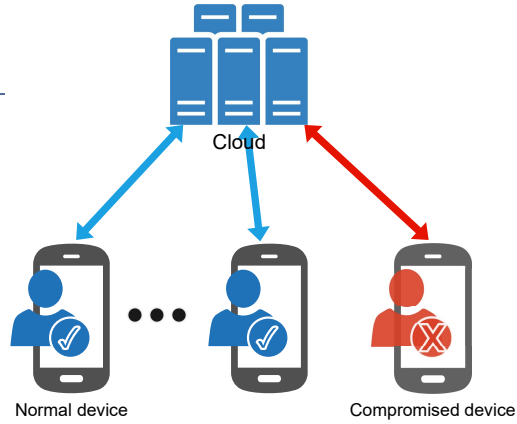
- Hardware/software failures
- Broken communication



Security in distributed ML

Untrustable workers:

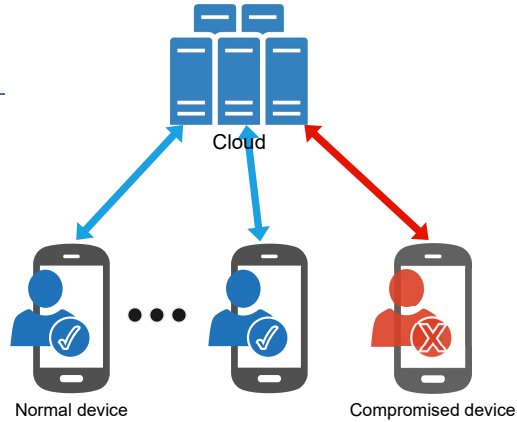
- Hardware/software failures
- Broken communication
- Poisoned local data



Security in distributed ML

Untrustable workers:

- Hardware/software failures
- Broken communication
- Poisoned local data
- Malicious users



Outline

Goal

Security in distributed SGD with:

Goal

Security in distributed SGD with:

- Arbitrary number of faulty workers

Goal

Security in distributed SGD with:

- Arbitrary number of faulty workers
- Untargeted attacks in worst cases

Goal

Security in distributed SGD with:

- Arbitrary number of faulty workers
- Untargeted attacks in worst cases
- Synchronous/asynchronous

Outline

Problem formulation

Optimization:

$$\min_{x \in \mathbb{R}^d} F(x),$$

Problem formulation

Optimization:

$$\min_{x \in \mathbb{R}^d} F(x),$$

where

$$F(x) = \frac{1}{n} \sum_{i \in [n]} \mathbb{E}_{z_i \sim \mathcal{D}_i} f(x; z_i),$$

Problem formulation

Optimization:

$$\min_{x \in \mathbb{R}^d} F(x),$$

where

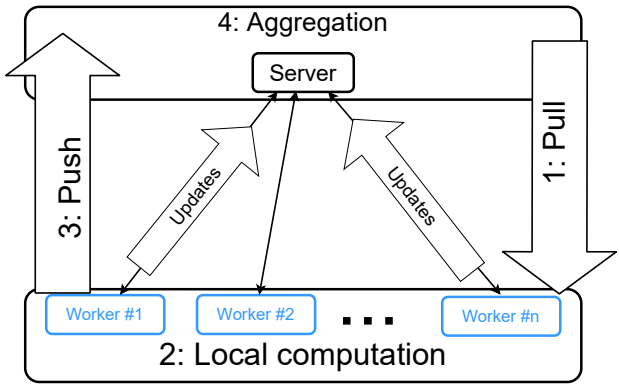
$$F(x) = \frac{1}{n} \sum_{i \in [n]} \mathbb{E}_{z_i \sim \mathcal{D}_i} f(x; z_i),$$

n nodes/devices, with disjoint datasets $\mathcal{D}_i$'s.

Server-worker architecture

Exchange updates:

- Gradients
- Model parameters
- etc.



Synchronous vs. Asynchronous

	Synchronous	Asynchronous
$x_{t+1} =$		
Issue:		

Synchronous vs. Asynchronous

	Synchronous	Asynchronous
$x_{t+1} =$	$Updater\big(x_t, \texttt{Aggr}(\{msg_i\})\big)$	
Issue:		

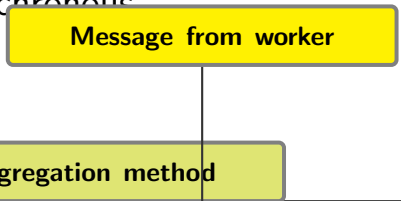
Server-worker architecture

Synchronous vs. Asynchronous

	Synchronous	Asynchronous
$x_{t+1} =$	$Updater\big(x_t, \texttt{Aggr}(\{msg_i\})\big)$	
Issue:		

Server-worker architecture

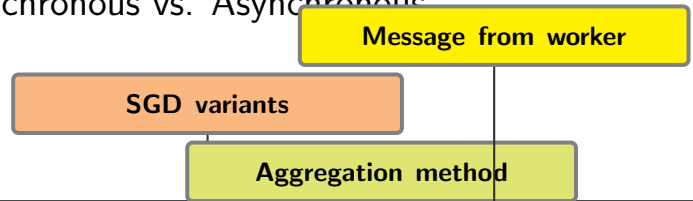
Synchronous vs. Asynchronous



	Synchronous	Asynchronous
$x_{t+1} =$	$Updater\big(x_t, \underline{\texttt{Aggr}}(\{\underline{msg_i}\})\big)$	
Issue:		

Server-worker architecture

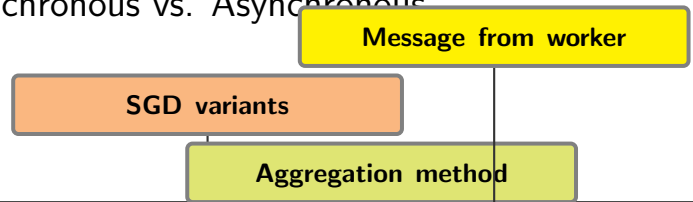
Synchronous vs. Asynchronous



	Synchronous	Asynchronous
$x_{t+1} =$	$Updater\big(x_t, \texttt{Aggr}(\{msg_i\})\big)$	
Issue:		

Server-worker architecture

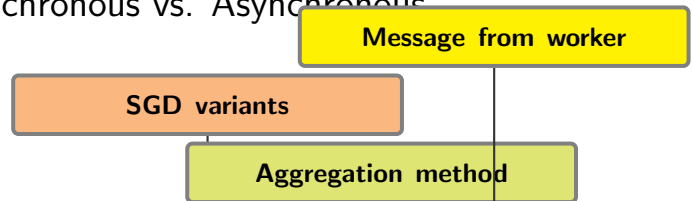
Synchronous vs. Asynchronous



	Synchronous	Asynchronous
$x_{t+1} =$	$Updater\big(x_t, \texttt{Aggr}(\{msg_i\})\big)$	$Updater\,(x_t, msg_i)$
Issue:		

Server-worker architecture

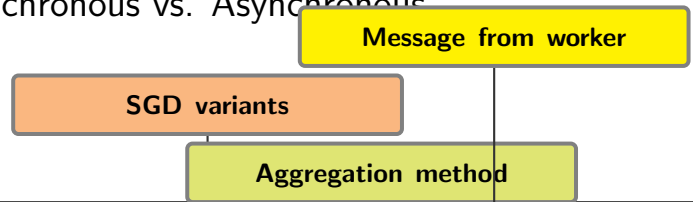
Synchronous vs. Asynchronous



	Synchronous	Asynchronous
$x_{t+1} =$	$Updater\big(x_t, \texttt{Aggr}(\{msg_i\})\big)$	$Updater\,(x_t, msg_i)$
Issue:	Stragglers	

Server-worker architecture

Synchronous vs. Asynchronous



	Synchronous	Asynchronous
$x_{t+1} =$	$Updater\big(x_t, \texttt{Aggr}(\{msg_i\})\big)$	$Updater\,(x_t, msg_i)$
Issue:	Stragglers	Noisy updates

Threat model

Threat model

Byzantine failures on q workers:

Threat model

Byzantine failures on q workers:

- Omniscient

Threat model

Byzantine failures on q workers:

- Omniscient
- Send arbitrary values

Threat model

Byzantine failures on q workers:

- Omniscient
- Send arbitrary values
- Untargeted, in worst cases

Threat model

Byzantine failures on q workers:

- Omniscient
- Send arbitrary values
- Untargeted, in worst cases
- $x^{t+1} = x^t - \gamma \texttt{Aggr}(\{\tilde{v}_i^t : i \in [n]\})$,

$$\tilde{v}_i^t = \begin{cases} *, & \text{if } i\text{th worker is Byzantine,} \\ \nabla f(x^t; z_i \sim \mathcal{D}_i), & \text{otherwise,} \end{cases}$$

Threat model

Byzantine failures on q workers:

- Omniscient
- Send arbitrary values
- Untargeted, in worst cases
- $x^{t+1} = x^t - \gamma \texttt{Aggr}(\{\tilde{v}_i^t : i \in [n]\})$,

$$\tilde{v}_i^t = \begin{cases} *, & \text{if } i\text{th worker is Byzantine,} \\ \nabla f(x^t; z_i \sim \mathcal{D}_i), & \text{otherwise,} \end{cases}$$

Honest servers

Threat model

Byzantine failures on q workers:

- Omniscient
- Send arbitrary values
- Untargeted, in worst cases
- $x^{t+1} = x^t - \gamma \texttt{Aggr}(\{\tilde{v}_i^t : i \in [n]\})$,

$$\tilde{v}_i^t = \begin{cases} *, & \text{if } i\text{th worker is Byzantine,} \\ \nabla f(x^t; z_i \sim \mathcal{D}_i), & \text{otherwise,} \end{cases}$$

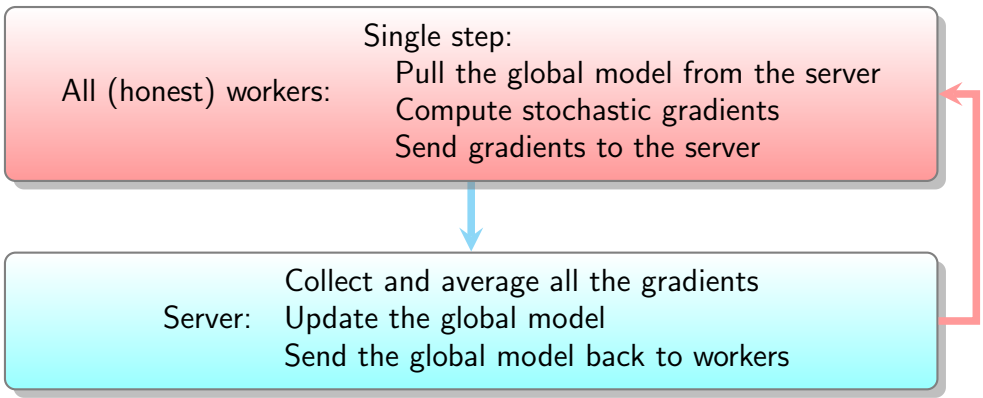
Honest servers

Goal: Prevent arbitrarily bad results

Outline

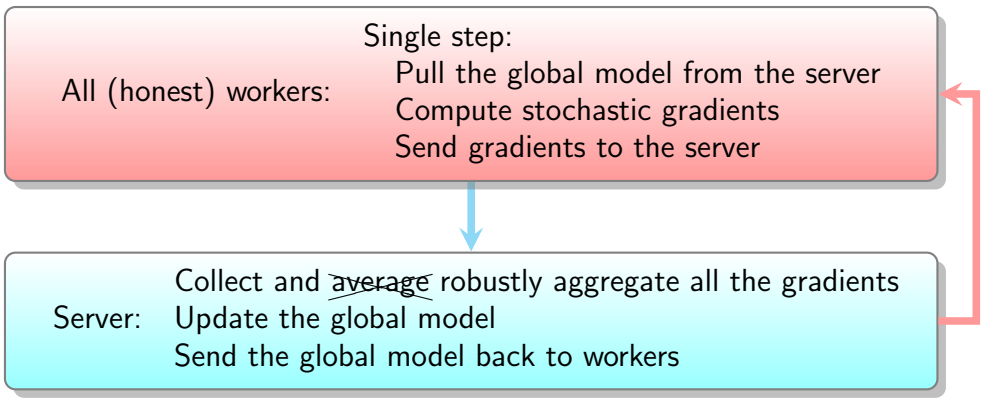
Byzantine-tolerant SGD

Recap: Byzantine workers send arbitrary values to the server



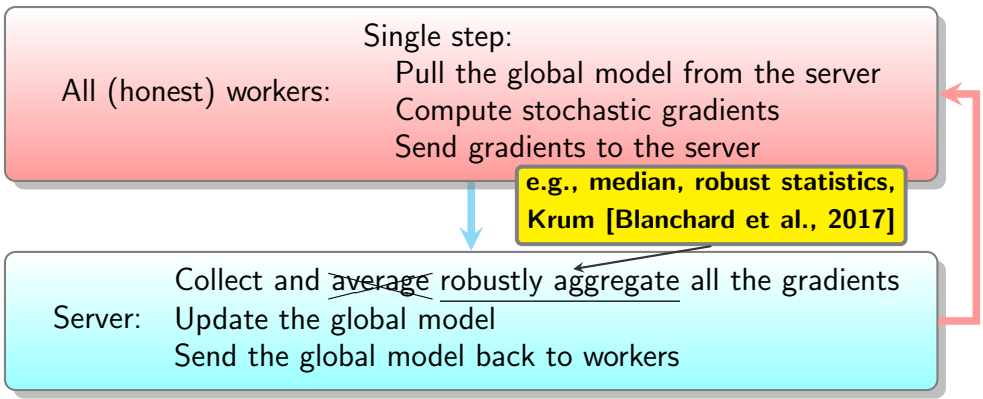
Byzantine-tolerant SGD

Recap: Byzantine workers send arbitrary values to the server



Byzantine-tolerant SGD

Recap: Byzantine workers send arbitrary values to the server



Related work

Median/Trimmed mean [Yin et al., 2018]:

- For each coordinate
- Filter out greatest/smallest values
- Take the average on the remainings

Related work

Median/Trimmed mean [Yin et al., 2018]:

- For each coordinate
- Filter out greatest/smallest values
- Take the average on the remainings

Geometric median [Chen et al., 2017]:

$$\operatorname*{argmin}_{v} \sum_{i \in [n]} \|v - \tilde{v}_i\|$$

Krum [Blanchard et al., 2017]:

$$\texttt{Krum}(\{\tilde{v}_i : i \in [m]\}) = \tilde{v}_k, \quad k = \operatorname*{argmin}_{i \in [n]} KR(\tilde{v}_i),$$

$$KR(\tilde{v}_i) = \sum_{i \to j} \|\tilde{v}_i - \tilde{v}_j\|^2,$$

where $i \to j$ are the indices of the $n - b - 2$ nearest neighbours of $\tilde{v}_i$ in $\{\tilde{v}_j : j \in [n], i \neq j\}$ as measured by squared Euclidean distance.

Bounded distance to the correct average

Outline

Issues of previous work

Issues of previous work

- Previous work: bounded distance to the correct average

Issues of previous work

- Previous work: bounded distance to the correct average
- Issue: bounded distance $\neq$ decrease of loss [Xie et al., 2019a]

Issues of previous work

- Previous work: bounded distance to the correct average
- Issue: bounded distance $\neq$ decrease of loss [Xie et al., 2019a]
- Toy example (sign-flipping attack):

Issues of previous work

- Previous work: bounded distance to the correct average
- Issue: bounded distance $\neq$ decrease of loss [Xie et al., 2019a]
- Toy example (sign-flipping attack):

  $\{-0.3, 1, 2, 3, 4\}$

  Mean: 1.94, positive

Issues of previous work

- Previous work: bounded distance to the correct average
- Issue: bounded distance $\neq$ decrease of loss [Xie et al., 2019a]
- Toy example (sign-flipping attack):

  $\{-0.3, 1, 2, 3, 4\}$

  Mean: 1.94, positive

  $\{-0.3, 1, 2, \cancel{3}, \cancel{4}\}$

Issues of previous work

- Previous work: bounded distance to the correct average
- Issue: bounded distance $\neq$ decrease of loss [Xie et al., 2019a]
- Toy example (sign-flipping attack):

  $\{-0.3, 1, 2, 3, 4\}$

  Mean: 1.94, positive

  $\{-0.3, 1, 2, \cancel{3}, \cancel{4}\}$

  Replace with negative average: $-\frac{-0.3+1+2}{3} = -0.9$

  $\{-0.3, 1, 2, -0.9, -0.9\}$

Issues of previous work

- Previous work: bounded distance to the correct average
- Issue: bounded distance $\neq$ decrease of loss [Xie et al., 2019a]
- Toy example (sign-flipping attack):

  $\{-0.3, 1, 2, 3, 4\}$

  Mean: 1.94, positive

  $\{-0.3, 1, 2, \cancel{3}, \cancel{4}\}$

  Replace with negative average: $-\frac{-0.3+1+2}{3} = -0.9$

  $\{-0.3, 1, 2, -0.9, -0.9\}$

  Median: -0.3, negative!

Issues of previous work

- Previous work: bounded distance to the correct average
- Issue: bounded distance $\neq$ decrease of loss [Xie et al., 2019a]
- Toy example (sign-flipping attack):

  $\{-0.3, 1, 2, 3, 4\}$

  Mean: 1.94, positive

  $\{-0.3, 1, 2, \cancel{3}, \cancel{4}\}$

  Replace with negative average: $-\frac{-0.3+1+2}{3} = -0.9$

  $\{-0.3, 1, 2, -0.9, -0.9\}$

  Median: -0.3, negative!

  Mean: 0.18, positive

Issues of previous work

[Xie et al., 2019a]

Theorem

Consider the worst case: $n - 2q = 1$. *The server receives* $(n - q)$ *correct gradients* $\mathcal{V} = \{v_1, \ldots, v_{n-q}\}$ *and* q *Byzantine gradients* $\mathcal{U} = \{u_1, \ldots, u_q\}$. $\mathbb{E}[v_i] = g, \forall i \in [n - q]$, $\mathbb{E}[((v_i)_j - g_j)^2] \geq \sigma^2, \forall i \in [n - q], j \in [d]$, *where* $(v_i)_j$ *is the* j*th coordinate of* v_i, *and* g_j *is the* j*th coordinate of* g. *When* $\max_{j \in [d]} |g_j| < \frac{\sigma}{\sqrt{n-q-1}}$, *there exist Byzantine gradients* $\mathcal{U} = \{u_1, \ldots, u_q\}$ *such that* $\langle g, \quad \mathbb{E}\left[\texttt{Median}(\mathcal{V} \cup \mathcal{U})\right] \rangle < 0$.

Issues of previous work

[Xie et al., 2019a]

Theorem

Consider the worst case: $n - 2q = 3$. *The server receives* $(n - q)$ *correct gradients* $\mathcal{V} = \{v_1, \ldots, v_{n-q}\}$ *and* q *Byzantine gradients* $\mathcal{U} = \{u_1, \ldots, u_q\}$. $\mathbb{E}[v_i] = g, \forall i \in [n - q]$. *The mean of correct gradients is* $\bar{v} = \frac{1}{n-q} \sum_{i \in [n-q]} v_i$. *The correct gradients are bounded by* $\|v_i - \bar{v}\|^2 \leq \|\bar{v}\|^2, \forall i \in [n - q]$. *Furthermore, we assume that* $v_i \neq v_j, \forall i \neq j, i, j \in [n - q]$, *and* $\exists \beta$ *such that* $\|v_i - v_j\|^2 \geq \beta^2, \forall i \neq j, i, j \in [n - q]$. *We take* $u_1 = u_2 = \cdots = u_q = -\epsilon\bar{v}$, *where* ϵ *is a small positive constant value such that* $\epsilon^2 \|\bar{v}\|^2 \leq \beta^2$. *When* $(n - q)$ *is large enough:* $n - q > \frac{2(\epsilon+2)^2}{\epsilon^2} + 2$, *we have* $\langle g, \quad \mathbb{E}\left[\texttt{Krum}(\mathcal{V} \cup \mathcal{U})\right] \rangle < 0$.

Motivation

- Previous work:

  Guarantee bounded distance to the correct average
  Bounded number of Byzantine workers ($\leq 50\%$, $2q < n$)

Motivation

- Previous work:
  - Guarantee bounded distance to the correct average
  - Bounded number of Byzantine workers ($\leq 50\%$, $2q < n$)
- Issues:
  - Bounded distance $\neq$ decrease of loss [Xie et al., 2019a]
  - Number of Byzantine workers $\geq 50\%$
  - Asynchronous training

Motivation

- Previous work:
  - Guarantee bounded distance to the correct average
  - Bounded number of Byzantine workers ($\leq 50\%$, $2q < n$)
- Issues:
  - Bounded distance $\neq$ decrease of loss [Xie et al., 2019a]
  - Number of Byzantine workers $\geq 50\%$
  - Asynchronous training
- Solution:
  - Estimate the decrease of loss

Byzantine-tolerant SGD: new approaches

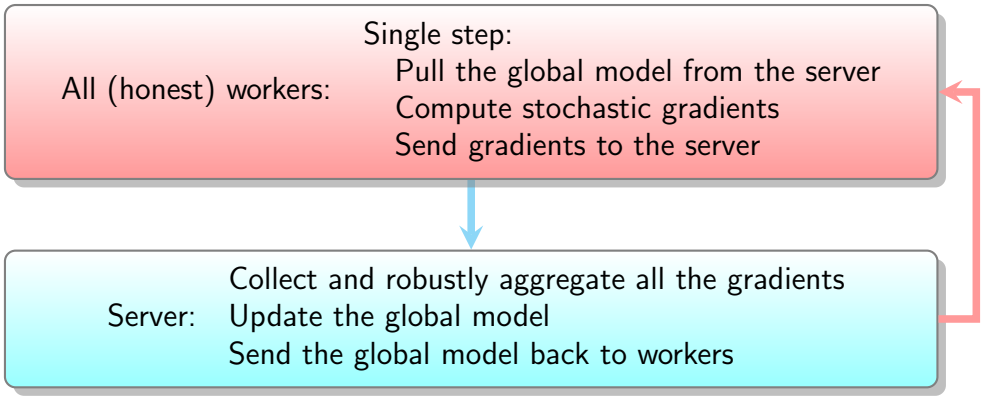
Byzantine-tolerant SGD: new approaches

- Validation on server:
  - A validation dataset separated from training data
  - Sample a batch from the validation set
  - Estimate the decrease of loss for each update

Byzantine-tolerant SGD: new approaches

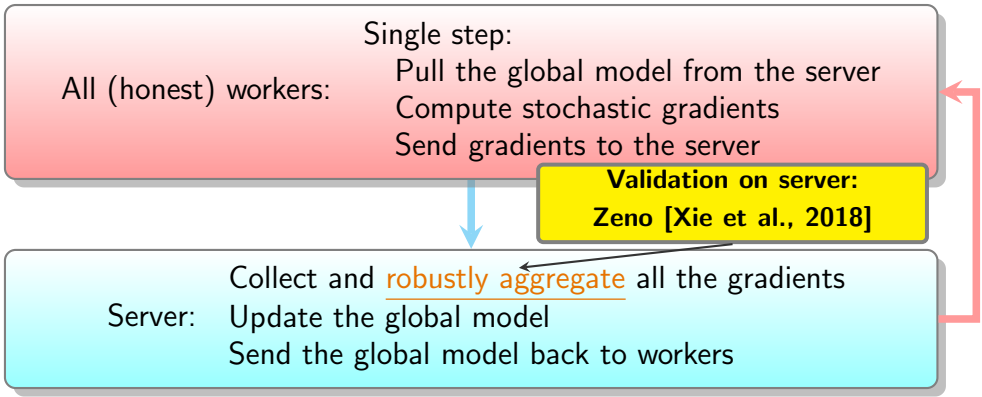
- Validation on server:
  - A validation dataset separated from training data
  - Sample a batch from the validation set
  - Estimate the decrease of loss for each update
- Zeno [Xie et al., 2018]:
  - Synchronous SGD with validation on server
  - Number of Byzantine workers $\geq 50\%$

Recap: Byzantine-tolerant SGD with robust aggregation



Zeno

Recap: Byzantine-tolerant SGD with robust aggregation



Zeno

Zeno

- Stochastic descent score:

  $Score_{\gamma,\rho}(u, x) = f_r(x) - f_r(x - \gamma u) - \rho\|u\|^2$

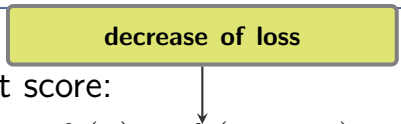
  $f_r(x)$: loss function on a random batch of validation data

  u: candidate gradient

  x: current global model

  γ, ρ: hyperparameters

Zeno



- Stochastic descent score:

$$Score_{\gamma,\rho}(u, x) = f_r(x) - f_r(x - \gamma u) - \rho\|u\|^2$$

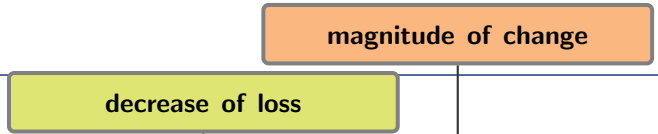
$f_r(x)$: loss function on a random batch of validation data

u: candidate gradient

x: current global model

γ, ρ: hyperparameters

Zeno



- Stochastic descent score:

$$Score_{\gamma,\rho}(u, x) = f_r(x) - f_r(x - \gamma u) - \rho\|u\|^2$$

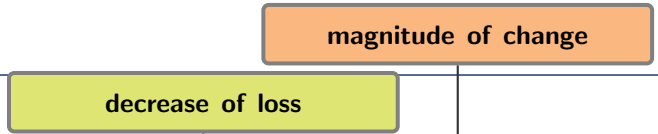
$f_r(x)$: loss function on a random batch of validation data

u: candidate gradient

x: current global model

γ, ρ: hyperparameters

Zeno



- Stochastic descent score:

  $Score_{\gamma,\rho}(u, x) = f_r(x) - f_r(x - \gamma u) - \rho\|u\|^2$

  $f_r(x)$: loss function on a random batch of validation data

  u: candidate gradient

  x: current global model

  γ, ρ: hyperparameters

- Zeno:

  Sort the gradients with the score

  Remove bottom-b, $b \geq q$

Why Zeno works?

Zeno

Why Zeno works?

- Bounded score

Zeno

Why Zeno works?

- Bounded score

  Score no worse than a correct gradient

Zeno

Why Zeno works?

- Bounded score

  Score no worse than a correct gradient

- Similarity between validation and training

Zeno

Zeno

- Gradients from workers: $v_1, \ldots, v_n$

Zeno

- Gradients from workers: $v_1, \ldots, v_n$

$$v_i = \nabla f(x),\ \mathbb{E}[v_i] = \nabla F(x)$$

Zeno

- Gradients from workers: $v_1, \ldots, v_n$

  $$v_i = \nabla f(x),\ \mathbb{E}[v_i] = \nabla F(x)$$

- Replace q of them by arbitrary vectors: $\tilde{v}_1, \ldots, \tilde{v}_n$

Zeno

- Gradients from workers: $v_1, \ldots, v_n$

  $$v_i = \nabla f(x),\ \mathbb{E}[v_i] = \nabla F(x)$$

- Replace q of them by arbitrary vectors: $\tilde{v}_1, \ldots, \tilde{v}_n$
- Sort by score, re-index:

Zeno

- Gradients from workers: $v_1, \ldots, v_n$

  $v_i = \nabla f(x)$, $\mathbb{E}[v_i] = \nabla F(x)$
- Replace q of them by arbitrary vectors: $\tilde{v}_1, \ldots, \tilde{v}_n$
- Sort by score, re-index:

  $\tilde{v}_{(1)}, \ldots, \tilde{v}_{(n)}$, $Score_{\gamma,\rho}(\tilde{v}_{(1)}, x) \geq \ldots \geq Score_{\gamma,\rho}(\tilde{v}_{(n)}, x)$

Zeno

- Gradients from workers: $v_1, \ldots, v_n$

  $v_i = \nabla f(x)$, $\mathbb{E}[v_i] = \nabla F(x)$
- Replace q of them by arbitrary vectors: $\tilde{v}_1, \ldots, \tilde{v}_n$
- Sort by score, re-index:

  $\tilde{v}_{(1)}, \ldots, \tilde{v}_{(n)}$, $Score_{\gamma,\rho}(\tilde{v}_{(1)}, x) \geq \ldots \geq Score_{\gamma,\rho}(\tilde{v}_{(n)}, x)$

  $v_{(1)}, \ldots, v_{(n)}$, $Score_{\gamma,\rho}(v_{(1)}, x) \geq \ldots \geq Score_{\gamma,\rho}(v_{(n)}, x)$

Zeno

- Gradients from workers: $v_1, \ldots, v_n$

  $v_i = \nabla f(x)$, $\mathbb{E}[v_i] = \nabla F(x)$
- Replace q of them by arbitrary vectors: $\tilde{v}_1, \ldots, \tilde{v}_n$
- Sort by score, re-index:

  $\tilde{v}_{(1)}, \ldots, \tilde{v}_{(n)}$, $Score_{\gamma,\rho}(\tilde{v}_{(1)}, x) \geq \ldots \geq Score_{\gamma,\rho}(\tilde{v}_{(n)}, x)$

  $v_{(1)}, \ldots, v_{(n)}$, $Score_{\gamma,\rho}(v_{(1)}, x) \geq \ldots \geq Score_{\gamma,\rho}(v_{(n)}, x)$

  Filter out the bottom-b, $b \geq q$

Zeno

- Gradients from workers: $v_1, \ldots, v_n$

  $v_i = \nabla f(x)$, $\mathbb{E}[v_i] = \nabla F(x)$
- Replace q of them by arbitrary vectors: $\tilde{v}_1, \ldots, \tilde{v}_n$
- Sort by score, re-index:

  $\tilde{v}_{(1)}, \ldots, \tilde{v}_{(n)}$, $Score_{\gamma,\rho}(\tilde{v}_{(1)}, x) \geq \ldots \geq Score_{\gamma,\rho}(\tilde{v}_{(n)}, x)$

  $v_{(1)}, \ldots, v_{(n)}$, $Score_{\gamma,\rho}(v_{(1)}, x) \geq \ldots \geq Score_{\gamma,\rho}(v_{(n)}, x)$

  Filter out the bottom-b, $b \geq q$

  Result: $\frac{1}{n-b} \sum_{i=1}^{n-b} \tilde{v}_{(i)}$

Zeno

Bounded score:

Zeno

Bounded score:

- $\forall i \in [n - q]$:

Zeno

Bounded score:

- $\forall i \in [n - q]$:
- $Score_{\gamma,\rho}(\tilde{v}_{(i)}, x) \geq Score_{\gamma,\rho}(v_{(i)}, x)$

Zeno

Bounded score:

- $\forall i \in [n-q]$:
- $Score_{\gamma,\rho}(\tilde{v}_{(i)}, x) \geq Score_{\gamma,\rho}(v_{(i)}, x)$
- $\tilde{v}_{(i)}$ has a score no worse than a correct one

Zeno

Bounded score:

- $\forall i \in [n-q]$:
- $Score_{\gamma,\rho}(\tilde{v}_{(i)}, x) \geq Score_{\gamma,\rho}(v_{(i)}, x)$
- $\tilde{v}_{(i)}$ has a score no worse than a correct one
- Proof by contradiction

Zeno

Bounded score:

- $\forall i \in [n-q]$:
- $Score_{\gamma,\rho}(\tilde{v}_{(i)}, x) \geq Score_{\gamma,\rho}(v_{(i)}, x)$
- $\tilde{v}_{(i)}$ has a score no worse than a correct one
- Proof by contradiction
- Connection from candidates to correct gradients

Zeno

In the worst cases:

Zeno

In the worst cases:

- Converge to a critical point with some error

Zeno

In the worst cases:

- Converge to a critical point with some error
- Run T iterations, take $\gamma = \frac{1}{L\sqrt{T}}$

Zeno

In the worst cases:

- Converge to a critical point with some error
- Run T iterations, take $\gamma = \frac{1}{L\sqrt{T}}$
- $\frac{\sum_{t=0}^{T-1} \mathbb{E}\|\nabla F(x_t)\|^2}{T} \leq$

Zeno

In the worst cases:

- Converge to a critical point with some error
- Run T iterations, take $\gamma = \frac{1}{L\sqrt{T}}$
- $\frac{\sum_{t=0}^{T-1} \mathbb{E}\|\nabla F(x_t)\|^2}{T} \leq \mathcal{O}\left(\frac{1}{(1-c)\sqrt{T}}\right)$

Converge with T ↗

Zeno

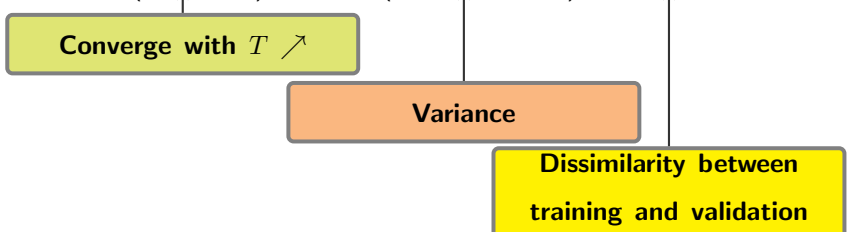
In the worst cases:

- Converge to a critical point with some error
- Run T iterations, take $\gamma = \frac{1}{L\sqrt{T}}$
- $$\frac{\sum_{t=0}^{T-1} \mathbb{E}\|\nabla F(x_t)\|^2}{T} \leq \mathcal{O}\left(\frac{1}{(1-c)\sqrt{T}}\right) + \mathcal{O}\left(\frac{(b-q+1)(n-q)}{(n-b)^2}\right)$$

Zeno

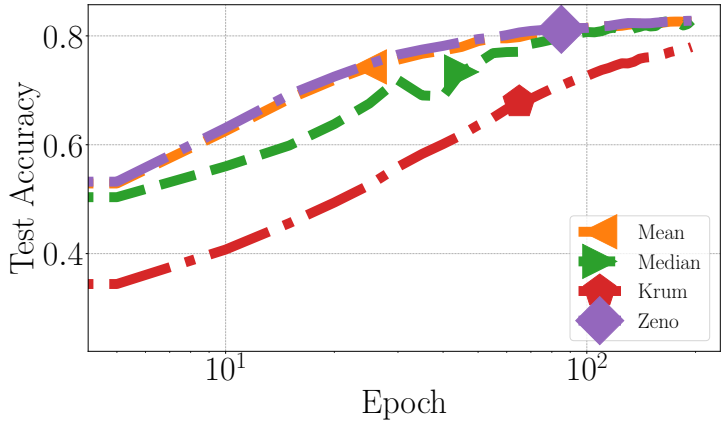
In the worst cases:

- Converge to a critical point with some error
- Run T iterations, take $\gamma = \frac{1}{L\sqrt{T}}$
- $$\frac{\sum_{t=0}^{T-1} \mathbb{E}\|\nabla F(x_t)\|^2}{T} \leq \mathcal{O}\left(\frac{1}{(1-c)\sqrt{T}}\right) + \mathcal{O}\left(\frac{(b-q+1)(n-q)}{(n-b)^2}\right) + \frac{e}{1-c}$$



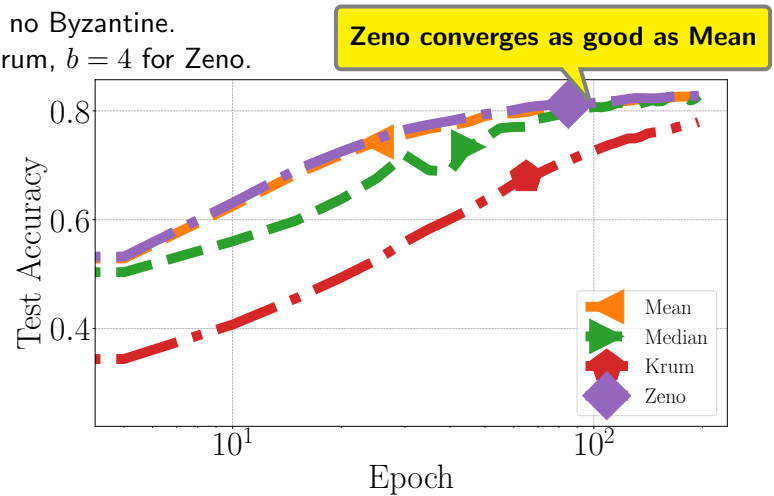
Zeno

Convergence without attacks on CNN and CIFAR-10.
20 workers, no Byzantine.
$b = 4$ for Krum, $b = 4$ for Zeno.



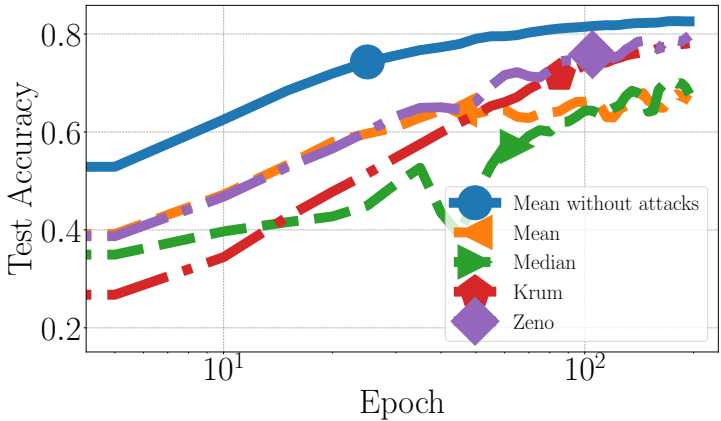
Zeno

Convergence without attacks on CNN and CIFAR-10.
20 workers, no Byzantine.
$b = 4$ for Krum, $b = 4$ for Zeno.



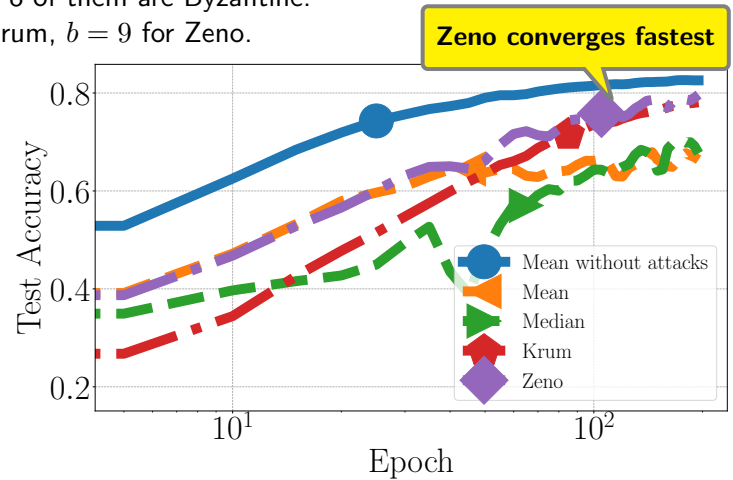
Zeno

Convergence with label-flipping attacks on CNN and CIFAR-10.
20 workers, 8 of them are Byzantine.
$b = 8$ for Krum, $b = 9$ for Zeno.



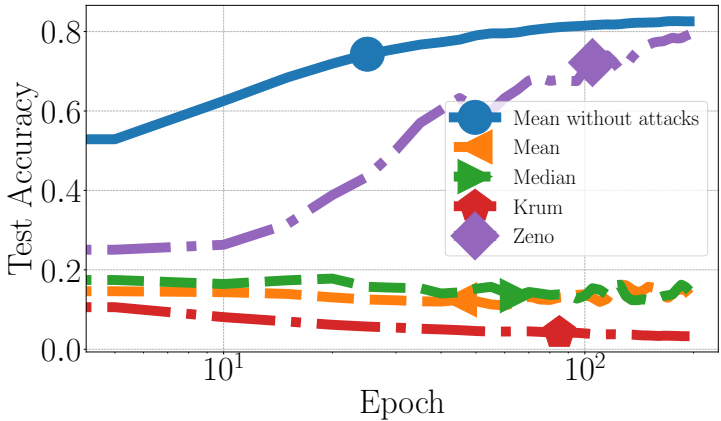
Zeno

Convergence with label-flipping attacks on CNN and CIFAR-10.
20 workers, 8 of them are Byzantine.
$b = 8$ for Krum, $b = 9$ for Zeno.



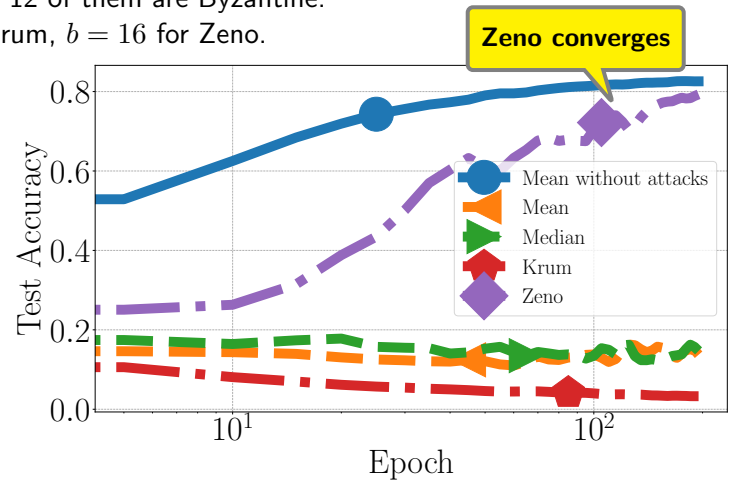
Zeno

Convergence with label-flipping attacks on CNN and CIFAR-10.
20 workers, 12 of them are Byzantine.
$b = 8$ for Krum, $b = 16$ for Zeno.



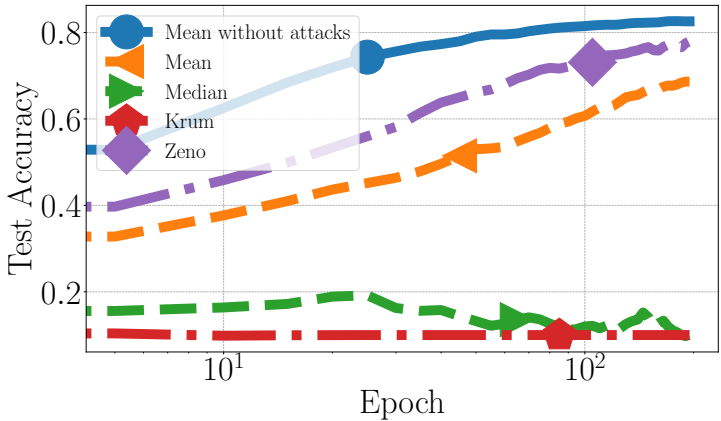
Zeno

Convergence with label-flipping attacks on CNN and CIFAR-10.
20 workers, 12 of them are Byzantine.
$b = 8$ for Krum, $b = 16$ for Zeno.



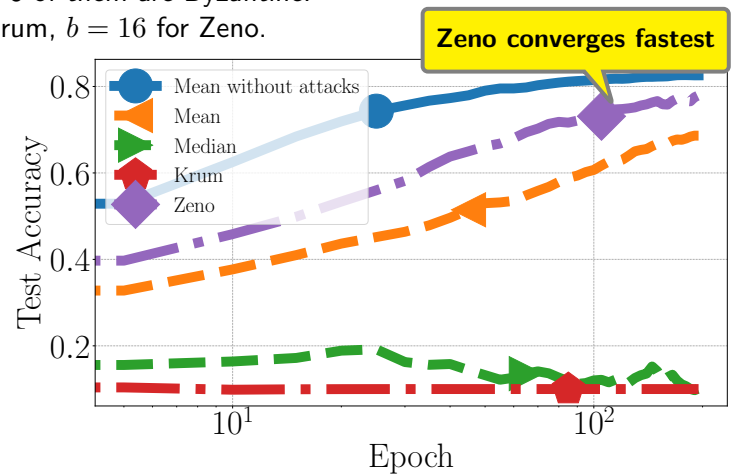
Zeno

Convergence with sign-flipping attacks on CNN and CIFAR-10.
20 workers, 8 of them are Byzantine.
$b = 8$ for Krum, $b = 16$ for Zeno.



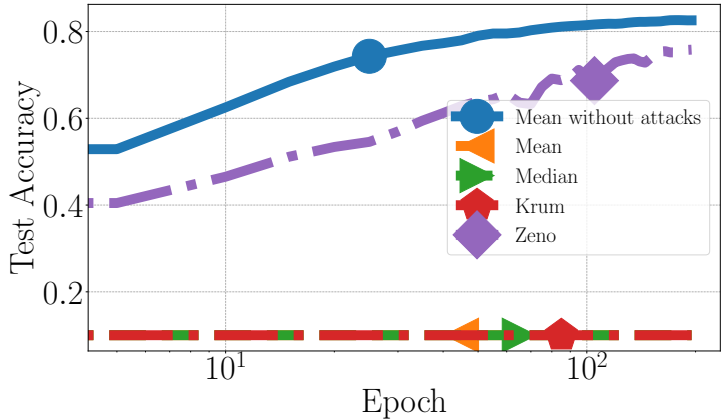
Zeno

Convergence with sign-flipping attacks on CNN and CIFAR-10.
20 workers, 8 of them are Byzantine.
$b = 8$ for Krum, $b = 16$ for Zeno.



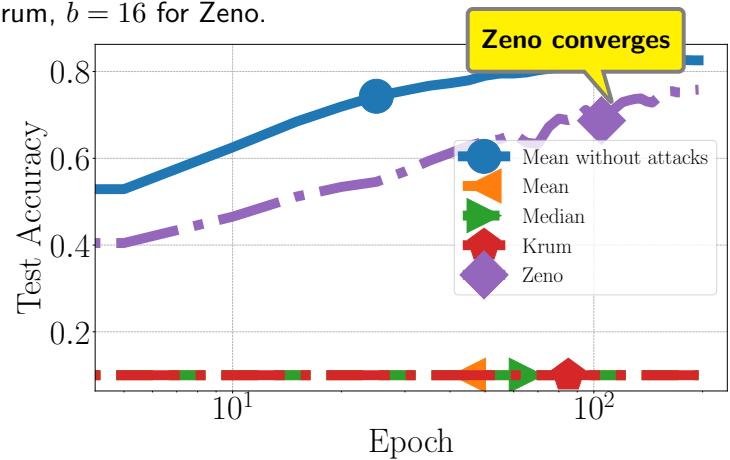
Zeno

Convergence with sign-flipping attacks on CNN and CIFAR-10.
20 workers, 12 of them are Byzantine.
$b = 8$ for Krum, $b = 16$ for Zeno.



Zeno

Convergence with sign-flipping attacks on CNN and CIFAR-10.
20 workers, 12 of them are Byzantine.
$b = 8$ for Krum, $b = 16$ for Zeno.



Zeno

Summary of Zeno:

Zeno

Summary of Zeno:

- Prevent catastrophic failures in the worst cases

Zeno

Summary of Zeno:

- Prevent catastrophic failures in the worst cases
- Tolerate Byzantine workers $\geq 50\%$

Zeno

Summary of Zeno:

- Prevent catastrophic failures in the worst cases
- Tolerate Byzantine workers $\geq 50\%$
- Aggregation is unnecessary:

Zeno

Summary of Zeno:

- Prevent catastrophic failures in the worst cases
- Tolerate Byzantine workers $\geq 50\%$
- Aggregation is unnecessary:

  Next: asynchronous training

Byzantine-tolerant SGD: asynchronous training

Byzantine-tolerant SGD: asynchronous training

- Asynchronous training:
  - Sorting is infeasible
  - Validation (computing the loss of batch) is expensive

Byzantine-tolerant SGD: asynchronous training

- Asynchronous training:
  - Sorting is infeasible
  - Validation (computing the loss of batch) is expensive
- Zeno++ [Xie et al., 2019b]:
  - Hard threshold instead of sorting
  - Inner-product validation
  - Full asynchrony

Zeno++

Zeno++

- Recap: stochastic descent score:

$$Score_{\gamma,\rho}(u, x) = f_r(x) - f_r(x - \gamma u) - \rho\|u\|^2$$

Zeno++

- Recap: stochastic descent score:

$$Score_{\gamma,\rho}(u, x) = f_r(x) - f_r(x - \gamma u) - \rho\|u\|^2$$

- (Approximated) stochastic descent score:

$$Score_{\gamma,\rho}(u, x) \approx \gamma \langle \nabla f_r(x), u \rangle - \rho\|u\|^2$$

x: current global model

Zeno++

- Recap: stochastic descent score:

$$Score_{\gamma,\rho}(u, x) = \gamma\langle \nabla f_r(x), u\rangle - \rho\|u\|^2$$

1st-order Taylor's expansion

- (Approximated) stochastic descent score:

$$Score_{\gamma,\rho}(u, x) \approx \gamma\langle \nabla f_r(x), u\rangle - \rho\|u\|^2$$

x: current global model

Zeno++

- Recap: stochastic descent score:

$$Score_{\gamma,\rho}(u, \cdot) - \rho\|u\|^2$$

1st-order Taylor's expansion

- (Approximated) stochastic descent score:

$$Score_{\gamma,\rho}(u, x) \approx \gamma\langle \nabla f_r(x), u\rangle - \rho\|u\|^2$$

x: ~~current~~ stale global model

Zeno++

- Recap: stochastic descent score:

  $Score_{\gamma,\rho}(u, x) = \gamma\langle \nabla f_r(x), u\rangle - \rho\|u\|^2$

  1st-order Taylor's expansion

- (Approximated) stochastic descent score:

  $Score_{\gamma,\rho}(u, x) \approx \gamma\langle \nabla f_r(x), u\rangle - \rho\|u\|^2$

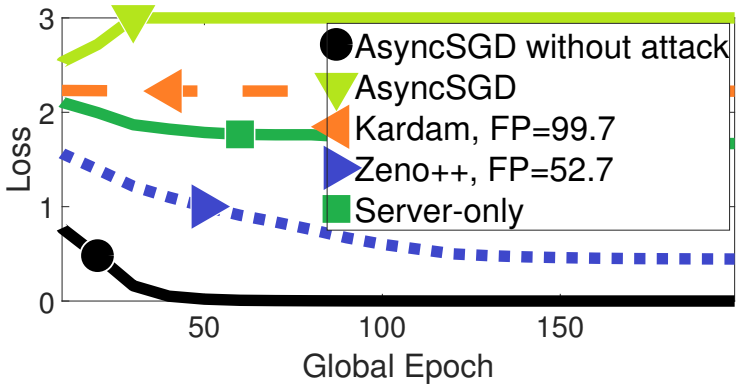
  x: ~~current~~ stale global model

- Threshold:

  $Score_{\gamma,\rho}(u, x) \geq \epsilon$

  Trade-off: $\epsilon \leq 0$

Convergence with sign-flipping attacks on CNN and CIFAR-10.
8 out of the 10 workers are Byzantine.



References I

[Blanchard et al., 2017] Blanchard, P., Guerraoui, R., Stainer, J., et al. (2017).

Machine learning with adversaries: Byzantine tolerant gradient descent.

In *Advances in Neural Information Processing Systems*, pages 118–128.

[Chen et al., 2017] Chen, Y., Su, L., and Xu, J. (2017).

Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent.

POMACS, 1:44:1–44:25.

[Xie et al., 2018] Xie, C., Koyejo, O., and Gupta, I. (2018).

Zeno: Distributed Stochastic Gradient Descent with Suspicion-based Fault-tolerance.

In *ICML*.

References II

[Xie et al., 2019a] Xie, C., Koyejo, O., and Gupta, I. (2019a).
Fall of Empires: Breaking Byzantine-tolerant SGD by Inner Product Manipulation.
In *UAI*.

[Xie et al., 2019b] Xie, C., Koyejo, O., and Gupta, I. (2019b).
Zeno++: robust asynchronous SGD with arbitrary number of Byzantine workers.
ArXiv, abs/1903.07020.

[Yin et al., 2018] Yin, D., Chen, Y., Ramchandran, K., and Bartlett, P. (2018).
Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates.
arXiv preprint arXiv:1803.01498.