Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates

Dong Yin, Yudong Chen , Kannan Ramchandran, and Peter Bartlett

UC Berkeley & Cornell University

Published in International Conference on Machine Learning. ICML, 2018

Citation:201

Presenter:

Xin Fan

2021/03/31

Outline

Algorithm 1 A General Framework for Byzantine-resilient Distributed SGD

for $t = 1, 2, \ldots$ **do**
  Server:
    Send the global optimization variable to all nodes
  Node:
    Receive the global optimization variable from the server
    Calculate gradient with respect to the local training set
    Send the local gradient to the server
  Server:
    Receive local gradients from all nodes
    Screen the received gradients for Byzantine resilience and aggregate them
    Update the global variable by taking a gradient step using the aggregated gradient
end for

How to design and analyze a Byzantine resilience algorithm?

Outline

- Problem Setup
- Related Work
- Robust Distributed Gradient Descent
- Robust One-round Algorithm
- Experiments
- Conclusion
- Our Discussions

Outline

- **Problem Setup**
- Related Work
- Robust Distributed Gradient Descent
- Robust One-round Algorithm
- Experiments
- Conclusion
- Our Discussions

Problem Setup

- One master machine and m worker machines.
- Each worker machine stores n data points, i.i.d. from D.
- Sometimes $n \gtrsim m$.
- αm worker machines are Byzantine ($0 \leq \alpha < \frac{1}{2}$).
- Normal worker machines communicate with the master machine using some predefined protocol.
- Byzantine machines may send arbitrary or even adversarial messages to the master machine.

Problem Setup

- **Goal**: minimize population risk $F(w) = \mathbb{E}_{z\sim\mathcal{D}} f(w; z)$, $w \in \mathcal{W} \subseteq \mathbb{R}^d$.
- **Focus**: achieving $\boxed{\text{the optimal statistical error rate}}$ w.r.t.:
  - n (the number of data points on each worker machine)
  - m (the number of worker machines)
  - α (the fraction of Byzantine machines)

$$\|\mathbf{w}^T - \mathbf{w}^*\|_2 \le \tilde{\Omega}\left(\frac{\alpha}{\sqrt{n}} + \frac{1}{\sqrt{nm}}\right) = \tilde{\Omega}\left(\frac{1}{\sqrt{n}}\left(\alpha + \frac{1}{\sqrt{m}}\right)\right)$$

Outline

- Problem Setup
- **Related Work**
- Robust Distributed Gradient Descent
- Robust One-round Algorithm
- Experiments
- Conclusion
- Our Discussions

Related Work

- Byzantine distributed learning:

  Feng et al. 2014: $\frac{1}{\sqrt{n}}$ rate, not decaying with α.

  Blanchard et al. 2017: no explicit statistical rates.

  Chen et al. 2017: $O\left(\frac{\sqrt{\alpha}}{\sqrt{n}} + \frac{1}{\sqrt{nm}}\right)$.

Outline

- Problem Setup
- Related Work
- **Robust Distributed Gradient Descent**
- Robust One-round Algorithm
- Experiments
- Conclusion
- Our Discussions

Robust Distributed Gradient Descent

Algorithm 1 Robust Distributed Gradient Descent

Require: Initialize parameter vector $\mathbf{w}^0 \in \mathcal{W}$, algorithm parameters β (for Option II), η and T.

for $t = 0, 1, 2, \ldots, T-1$ **do**

Master machine: send $\mathbf{w}^t$ to all the worker machines.

for all $i \in [m]$ **do in parallel**

Worker machine i: compute local gradient

$$\mathbf{g}^i(\mathbf{w}^t) \leftarrow \begin{cases} \nabla F_i(\mathbf{w}^t) & \text{normal worker machines,} \\ * & \text{Byzantine machines,} \end{cases}$$

send $\mathbf{g}^i(\mathbf{w}^t)$ to master machine.

end for

Master machine: compute aggregate gradient

$$\mathbf{g}(\mathbf{w}^t) \leftarrow \begin{cases} \mathsf{med}\{\mathbf{g}^i(\mathbf{w}^t) : i \in [m]\} & \text{Option I} \\ \mathsf{trmean}_\beta\{\mathbf{g}^i(\mathbf{w}^t) : i \in [m]\} & \text{Option II} \end{cases}$$

update model parameter $\mathbf{w}^{t+1} \leftarrow \Pi_{\mathcal{W}}(\mathbf{w}^t - \eta \mathbf{g}(\mathbf{w}^t))$.

end for

GEORGE MASON UNIVERSITY

Robust Distributed Gradient Descent

- Robust gradient aggregation: $g(w) = \mathcal{R}(g_1, g_2, \ldots, g_m)$
- Coordinate-wise median: $g(w) = \mathrm{med}(g_1, g_2, \ldots, g_m)$
  Take the one-dimensional median on each coordinate.
- Coordinate-wise β-trimmed mean: $g(w) = \mathrm{trmean}_\beta(g_1, g_2, \ldots, g_m)$
  Trimmed mean: remove the largest and smallest β fraction of the elements and then average.

Robust Distributed Gradient Descent

- **Definition**
- Variance of a random vector x:

$$\mathrm{var}(\mathrm{x}) = \mathbb{E}\|\mathrm{x} - \mathbb{E}\mathrm{x}\|^2$$

- Absolute skewness of a random variable X:

$$\gamma(X) = \frac{\mathbb{E}|X - \mathbb{E}X|^3}{\mathrm{var}(X)^{3/2}}$$

- v-sub-exponential random variable X:

$$\mathbb{E}e^{\lambda(X - \mathbb{E}X)} \leq e^{\frac{1}{2}v^2\lambda^2}, \qquad \forall\, |\lambda| < \frac{1}{v}$$

Robust Distributed Gradient Descent

- **Definition**
- Variance of a random vector x:

$$\mathrm{var}(\mathrm{x}) = \mathbb{E}\|\mathrm{x} - \mathbb{E}\mathrm{x}\|^2$$

- Absolute skewness of a random variable X:

$$\gamma(X) = \frac{\mathbb{E}|X - \mathbb{E}X|^3}{\mathrm{var}(X)^{3/2}}$$

- v-sub-exponential random variable X:

$$\mathbb{E}e^{\lambda(X-\mathbb{E}X)} \leq e^{\frac{1}{2}v^2\lambda^2}, \qquad \forall\, |\lambda| < \frac{1}{v}$$

- **Smoothness:** Population loss function $F(w)$ is L_F-smooth, and for every z, each partial derivative $\partial_k f(w; z)$ is L_k-Lipschitz.

- **Bounded parameter space:** $\forall\, w, w' \in W$, $\|w - w'\|_2 \leq D$.

- **Minimizer (convex loss):** $w^* = \mathrm{argmin}_{w \in W} F(w)$, $\nabla F(w^*) = 0$.

Robust Distributed Gradient Descent

Median-based Gradient Descent

- $\text{var}\big(\nabla F(w)\big) \le V^2, \forall\, w \in W.$
- $\gamma\big(\partial_k f(w;z)\big) \le S, \forall\, k \in [d],\ w \in W.$
- $\alpha + \tilde{O}\left(\frac{1}{\sqrt{m}} + \frac{1}{\sqrt{n}}\right) \le \frac{1}{2} - \epsilon.$
- Step size $\eta = 1/L_F$.

- **Strongly convex loss function**
- **Theorem 1** Suppose that $F(w)$ is λ_F-strongly convex. Then, w.h.p., after $O\left(\frac{L_F}{\lambda_F}\log\left(\frac{\lambda_F}{\Delta}\|w^0 - w^*\|_2\right)\right)$ parallel iterations, the median-based gradient descent algorithm outputs $\widehat{w}$ with

$$\|\widehat{w} - w^*\|_2 \le O\left(\frac{\Delta}{\lambda_F}\right),$$

where $\Delta = \tilde{O}\left(V\left(\frac{\alpha}{\sqrt{n}} + \sqrt{\frac{d}{nm}} + \frac{S}{n}\right)\right)$.

$$n \gtrsim m$$

$$\tilde{\Omega}\left(\frac{\alpha}{\sqrt{n}} + \frac{1}{\sqrt{nm}}\right) = \tilde{\Omega}\left(\frac{1}{\sqrt{n}}\left(\alpha + \frac{1}{\sqrt{m}}\right)\right)$$

24

Robust Distributed Gradient Descent

Median-based Gradient Descent

- **Non-strongly convex loss function**
- **Theorem 2** Suppose that $F(w)$ is convex. Then, w.h.p., after $O\left(\frac{L_F}{\Delta}\|w^0 - w^*\|_2\right)$ parallel iterations, the median-based gradient descent algorithm outputs $\widehat{w}$ with

$$F(\widehat{w}) - F(w^*) \le O(\Delta\|w^0 - w^*\|_2),$$

where $\Delta = \tilde{O}\left(V\left(\frac{\alpha}{\sqrt{n}} + \sqrt{\frac{d}{nm}} + \frac{S}{n}\right)\right)$.

Robust Distributed Gradient Descent

Median-based Gradient Descent

- **Non-convex loss function**
- **Theorem 3** W.h.p., after $O\left(\frac{L_F}{\Delta^2}\left(F(w^0) - F(w^*)\right)\right)$ parallel iterations, the median-based gradient descent algorithm outputs $\widehat{w}$ with

$$\|\nabla F(\widehat{w})\|_2 \leq O(\Delta),$$

where $\Delta = \tilde{O}\left(V\left(\frac{\alpha}{\sqrt{n}} + \sqrt{\frac{d}{nm}} + \frac{S}{n}\right)\right)$.

Robust Distributed Gradient Descent

Trimmed-mean-based Gradient Descent

- $\forall\, k \in [d],\ w \in W, \partial_k f(w; z)$ is v-sub-exponential.
- $\alpha \le \beta \le \frac{1}{2} - \epsilon$.
- Step size $\eta = 1/L_F$.

- **Strongly convex loss function**
- **Theorem 4** Suppose that $F(w)$ is λ_F-strongly convex. Then, w.h.p., after $O\left(\frac{L_F}{\lambda_F}\log\left(\frac{\lambda_F}{\Delta'}\|w^0 - w^*\|_2\right)\right)$ parallel iterations, the median-based gradient descent algorithm outputs $\widehat{w}$ with

$$\|\widehat{w} - w^*\|_2 \le O\left(\frac{\Delta'}{\lambda_F}\right),$$

where $\Delta' = \tilde{O}\left(vd\left(\frac{\beta}{\sqrt{n}} + \sqrt{\frac{1}{nm}}\right)\right)$.

GEORGE MASON UNIVERSITY

Robust Distributed Gradient Descent

Trimmed-mean-based Gradient Descent

- **Non-strongly convex loss function**
- **Theorem 5** Suppose that $F(w)$ is convex. Then, w.h.p., after $O\left(\frac{L_F}{\Delta'}\|w^0 - w^*\|_2\right)$ parallel iterations, the median-based gradient descent algorithm outputs $\widehat{w}$ with
$$F(\widehat{w}) - F(w^*) \leq O(\Delta'\|w^0 - w^*\|_2),$$
where $\Delta' = \tilde{O}\left(vd\left(\frac{\beta}{\sqrt{n}} + \sqrt{\frac{1}{nm}}\right)\right)$.

Robust Distributed Gradient Descent

Trimmed-mean-based Gradient Descent

- **Non-convex loss function**
- **Theorem 6** W.h.p., after $O\left(\frac{L_F}{\Delta'^2}\left(F(w^0) - F(w^*)\right)\right)$ parallel iterations, the median-based gradient descent algorithm outputs $\widehat{w}$ with

$$\|\nabla F(\widehat{w})\|_2 \leq O(\Delta'),$$

where $\Delta' = \tilde{O}\left(vd\left(\frac{\beta}{\sqrt{n}} + \sqrt{\frac{1}{nm}}\right)\right)$.

Robust Distributed Gradient Descent

Comparison

	median GD	trimmed mean GD
Statistical error rate	$\widetilde{\mathcal{O}}(\frac{\alpha}{\sqrt{n}} + \frac{1}{\sqrt{nm}} + \frac{1}{n})$	$\widetilde{\mathcal{O}}(\frac{\alpha}{\sqrt{n}} + \frac{1}{\sqrt{nm}})$
Distribution of $\partial_k f(\mathbf{w}; \mathbf{z})$	Bounded skewness	Sub-exponential
α known?	No	Yes

Robust Distributed Gradient Descent

Proof Sketch

- **Statistics part**
  - Bound the error of robust gradient estimation for a fixed $w \in W$.
  - For median, use Berry-Esseen-type inequality to bound the asymptotic convergence rate to normal distribution.
  - For trimmed mean, use standard Bernstein-type inequality.
  - Due to complicated probabilistic dependence, we need careful covering net argument to prove a uniform bound $\forall\, w \in W$.

- **Optimization part**
  - Convert the problem to gradient descent with bounded adversarial noise:
  - $w \leftarrow w - \eta g(w)$ with $\|g(w) - \nabla F(w)\| \leq \Delta, \forall\, w \in W$.

Outline

- Problem Setup
- Related Work
- Robust Distributed Gradient Descent
- **Robust One-round Algorithm**
- Experiments
- Conclusion
- Our Discussions

GEORGE MASON UNIVERSITY

Robust One-round Algorithm

- **Quadratic loss function:**

$$f(w;z) = \frac{1}{2} w^T H w + p^T w + c$$
$$F(w) = \frac{1}{2} w^T H_F w + p_F^T w + c_F$$

- **Theorem 7** Suppose that $\alpha + \tilde{O}\left(\frac{1}{\sqrt{m}} + \frac{1}{\sqrt{n}}\right) \leq \frac{1}{2} - \epsilon$. Then, for strongly convex quadratic loss function, w.h.p., the output of the robust one-round algorithm satisfies

$$\|\widehat{w} - w^*\|_2 \leq \tilde{O}\left(\frac{\alpha}{\sqrt{n}} + \frac{1}{\sqrt{nm}} + \frac{1}{n}\right).$$

Outline

- Problem Setup
- Related Work
- Robust Distributed Gradient Descent
- Robust One-round Algorithm
- **Experiments**
- Conclusion
- Our Discussions

GEORGE MASON UNIVERSITY

Experiments

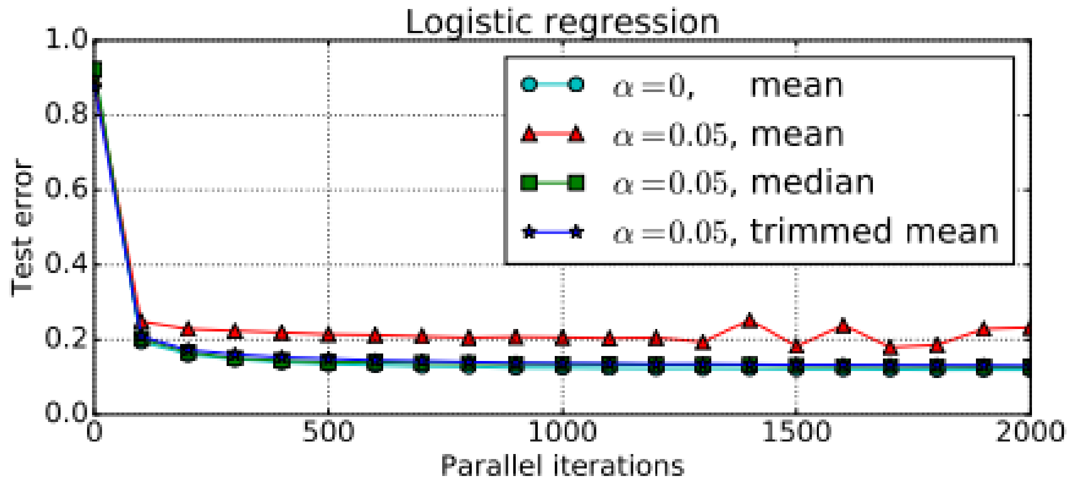
- Byzantine: replace every training label y on these machines with 9 − y, e.g., 0 is replaced with 9, 1 is replaced with 8, etc, and the Byzantine machines simply compute gradients based on these data.
  - Robust gradient descent (MNIST)
  - Logistic regression, $m = 40$:

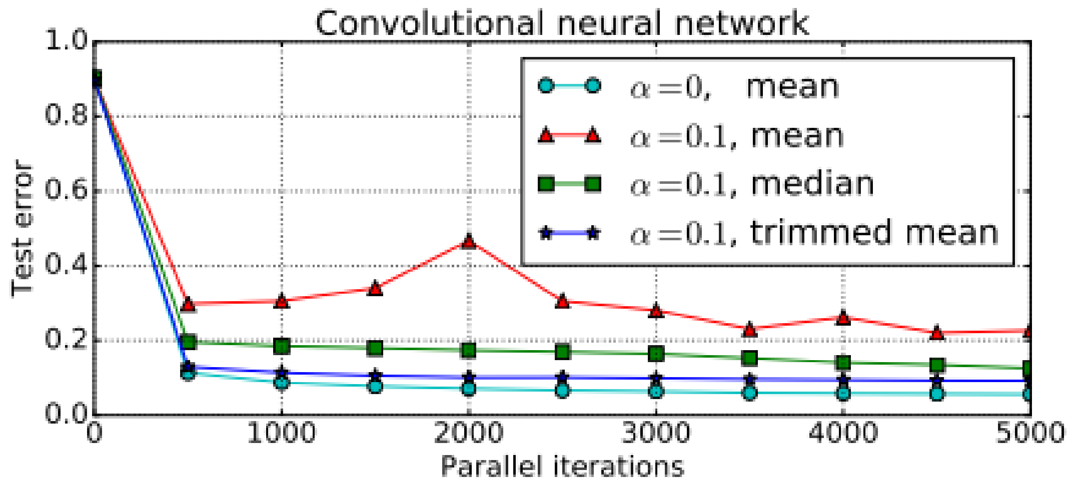
α	0	0.05		
Algorithm	mean	mean	median	trimmed mean
Test accuracy (%)	88.0	76.8	87.2	86.9

  - CNN, $m = 10$:

α	0	0.1		
Algorithm	mean	mean	median	trimmed mean
Test accuracy (%)	94.3	77.3	87.4	90.7

Experiments





Experiments

- Robust one-round algorithm
- Logistic regression, $m = 10$:

α	0	0.1	
Algorithm	mean	mean	median
Test accuracy (%)	91.8	83.7	89.0

Outline

- Problem Setup
- Related Work
- Robust Distributed Gradient Descent
- Robust One-round Algorithm
- Experiments
- **Conclusion**
- Our Discussions

Conclusion

- Median-based gradient descent achieves order optimal rate if $n \gtrsim m$.
- Trimmed-mean-based gradient descent achieves order optimal rate under stronger probabilistic assumptions.
- Median-based one-round algorithm achieves order optimal rate for quadratic loss functions if $n \gtrsim m$.

Outline

- Problem Setup
- Related Work
- Robust Distributed Gradient Descent
- Robust One-round Algorithm
- Experiments
- Conclusion
- **Our Discussions**

Our Discussions

SUMMARY OF SOME RECENT RESULTS CONCERNING BYZANTINE-RESILIENT DISTRIBUTED MACHINE LEARNING

Algorithm	**Convergence Rate**	**Statistical Learning Rate**	**Condition on** (M, b)
Coordinate-wise Median (CM) [15]	$\mathcal{O}\left(c^t\right)$	$\mathcal{O}\left(\frac{b}{M\sqrt{N}} + \frac{1}{\sqrt{MN}} + \frac{1}{N}\right)$	$M \geq 2b+1$
Coordinate-wise Trimmed Mean (CTM) [15]	$\mathcal{O}\left(c^t\right)$	$\mathcal{O}\left(\frac{b}{M\sqrt{N}} + \frac{1}{\sqrt{MN}}\right)$	$M \geq 2b+1$
GeoMed [16]	$\mathcal{O}\left(c^t\right)$	$\mathcal{O}\left(\frac{\sqrt{b}}{\sqrt{MN}}\right)$	$M \geq 2b+1$
Krum [17]	N/A	N/A	$M \geq 2b+3$
Multi-Krum [17]	N/A	N/A	$M \geq 2b+m+2$
Bulyan [18]	N/A	N/A	$M \geq 4b+3$
Zeno/Zeno++ [20], [21]	$\mathcal{O}\left(c^t\right) + \mathcal{O}\left(1\right)$	N/A	$M \geq b+1$
RSA [23]	$\mathcal{O}\left(\frac{1}{t}\right) + \mathcal{O}\left(1\right)$	N/A	$M \geq b+1$
signSGD [24]	—	N/A	$M \geq 2b+1$

Algorithm	CM, CTM, Zeno/Zeno++	GeoMed	Krum, Multi-Krum	Bulyan
Screening Complexity	$\mathcal{O}(Md)$	$\mathcal{O}\left(Md + bd\log^3\left(\frac{1}{\gamma}\right)\right)^\star$	$\mathcal{O}(M^2d)$	$\mathcal{O}(M^2d + Md)$

Screening computational complexity for GeoMed is for computing $(1+\gamma)$-approximate geometric median [16].

Our Discussions

- GeoMed algorithm
  - uses the geometric median of local gradients as the screening and aggregation rule.
- Krum algorithm,
  - finds the local gradient that has the smallest distance to its $M - b - 2$ closest gradients and uses this gradient for the update step.
- Multi-Krum
  - finds $m \in \{1, \ldots, M\}$ local gradients using the Krum principle and uses an average of these gradients for update.
- The Bulyan algorithm
  - a two-stage algorithm. First, it recursively uses vector median methods such as Geometric median and Krum to select $M - 2b$ local gradients. And then eliminate $2b$ values that are farthest from the coordinate-wise median.
- Zeno/Zeno++
  - has a ground truth dataset (the true gradient)
- RSA
  - add a regularization term.
- SignSGD
  - voted mechanism