Robust Collaborative Learning with Linear Gradient Overhead

Sadegh Farhadkhani Rachid Guerraoui Nirupam Gupta

Lê-Nguyên Hoang Rafaël Pinot *John Stephan*

{sadegh.farhadkhani@epfl.ch}

EPFL

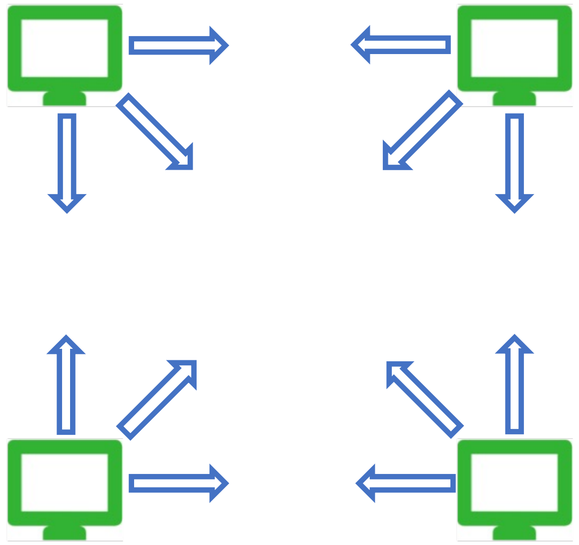
ICML
International Conference
On Machine Learning

Collaborative Learning

- n nodes, each with a local dataset.
- Communicate through asynchronous channels.
- Goal: Collaboratively solving a common ML task without sharing the data:

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} Q^{(i)}(\theta)$$

$Q^{(i)}$: local loss of node i (non-convex)



D-SGD

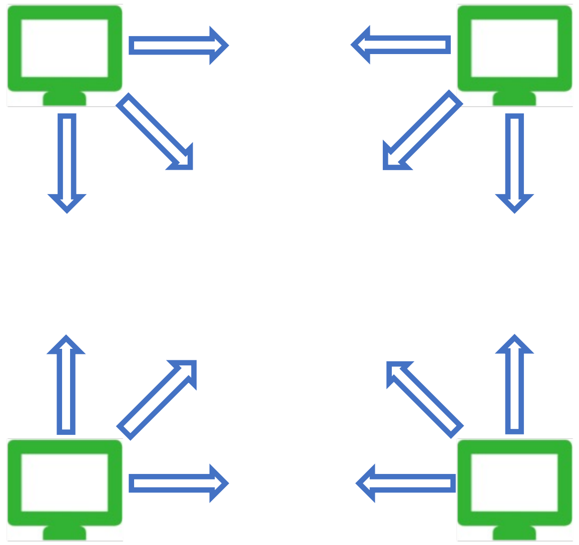
Node i at learning iteration t:

- Local phase:
  - take a local step using stochastic gradient $g_t^{(i)}$:

$$\theta_{t+1/2}^{(i)} \leftarrow \theta_t^{(i)} - \gamma g_t^{(i)}$$

- Coordination phase:
  - Broadcast $\theta_{t+1/2}^{(i)}$
  - Compute the average of received models:

$$\theta_{t+1}^{(i)} \leftarrow \underset{j \in R^{(i)}}{\mathrm{AVG}} \left(\theta_{t+1/2}^{(j)}\right)$$

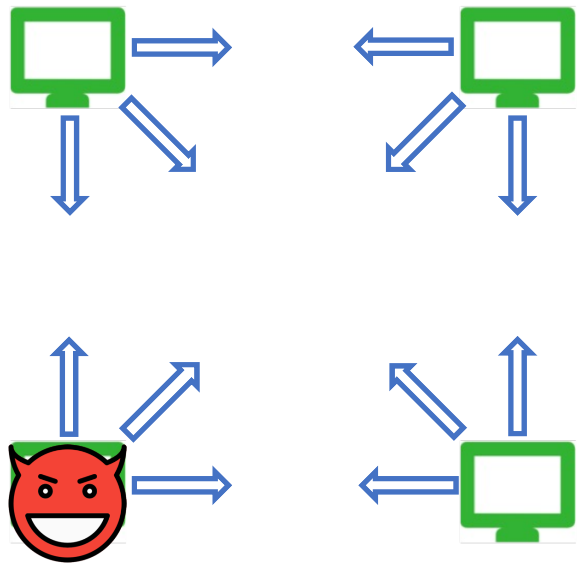


Convergence rate: $\mathcal{O}\left(\frac{\sigma^2}{\underbrace{n}\epsilon^2}\right)$

Linear speed-up: improves with the nubmer of nodes

Byzantine Threat Model

- Up to f nodes are faulty and send arbitrary vectors to others.



Byzantine Threat Model

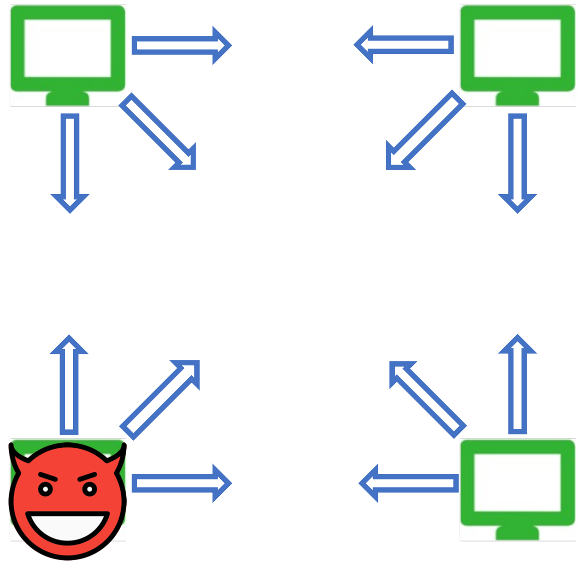
- Up to f nodes are faulty and send arbitrary vectors to others.

Can we converge in the presence of faulty nodes?

➢ Yes (El-Mhamdi et al. NeurIPS 2021).

How fast?

➢ Previous works: Orders of magnitude slower than D-SGD.



Our Result

Convergence (to an optimum ball) with rate

$$\mathcal{O}\left(\frac{\sigma^2 (1 + f)}{n\epsilon^2}\right).$$

- Linear overhead in the number of faults.
- Recovers D-SGD for $f = 0$.
- Conjectured to be tight.

(α, λ)-reduction: a new mixing criterion to analyze the non-linear mixing of non-faulty nodes instead of the spectral gap.

Our Algorithm: MoNNA

Correct node i at learning iteration t:

- Local phase:
  - Polyak's (Mo)mentum for the local update:

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

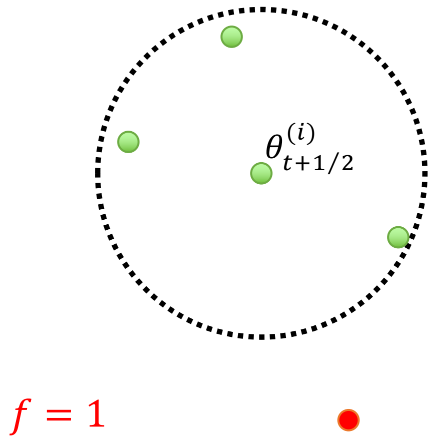
$$\theta_{t+1/2}^{(i)} \leftarrow \theta_t^{(i)} - \gamma m_t^{(i)}$$

- Coordination phase:
  - Broadcast $\theta_{t+1/2}^{(i)}$
  - Nearest Neighbor Averaging (NNA):

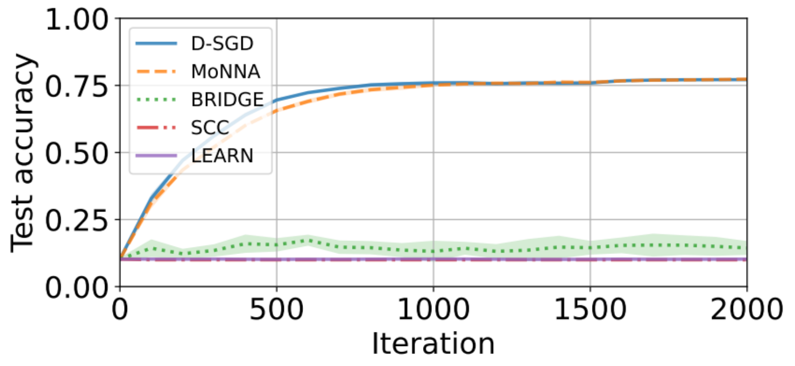
$$\theta_{t+1}^{(i)} \leftarrow \underset{j \in R^{(i)}}{\mathrm{NNA}}\left(\theta_{t+1/2}^{(j)}\right)$$

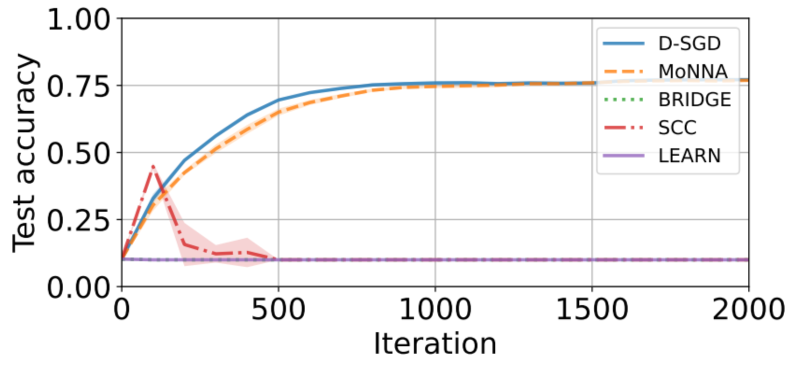
NNA at node i:

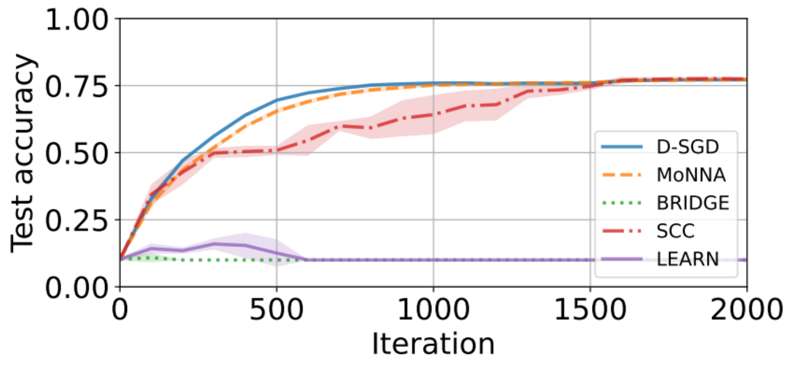
1. Filter out f vectors that are furthest from $\theta_{t+1/2}^{(i)}$.
2. Average the remaining vectors.

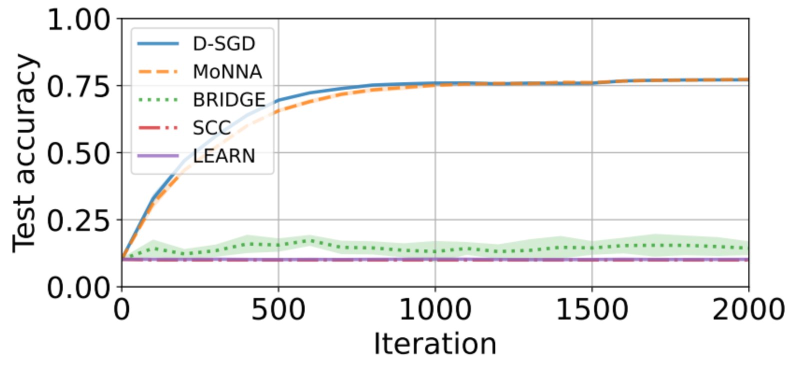


Experiments: MoNNA vs. Previous Works









Thank you!

Robust Collaborative Learning with Linear Gradient Overhead

Sadegh Farhadkhani [1] Rachid Guerraoui [1] Nirupam Gupta [1]
Lê Nguyên Hoang [2 3] Rafael Pinot [1] John Stephan [1]

Abstract

Collaborative learning algorithms, such as *distributed SGD* (or D-SGD), are prone to faulty machines that may deviate from their prescribed algorithm because of software or hardware bugs, poisoned data or malicious behaviors. While many solutions have been proposed to enhance the robustness of D-SGD to such machines, previous works either resort to strong assumptions (*trusted* server, *homogeneous* data, specific noise model) or impose a gradient computational cost that is several orders of magnitude higher than that of D-SGD. We present MoNNA, a new algorithm that (a) is provably robust under standard assumptions and (b) has a gradient computation overhead that is linear in the fraction of faulty machines, which is conjectured to be tight. Essentially, MoNNA uses *Polyak's momentum* of local gradients for *local updates* and *nearest-neighbor averaging (NNA)* for *global mixing*, respectively. While MoNNA is rather simple to implement, its analysis has been more challenging and relies on two key elements that may be of independent interest. Specifically, we introduce the mixing criterion of (α, λ)-*reduction* to analyze the *non-linear mixing* of non-faulty machines, and present a way to control the tension between the momentum and the model *drifts*. We validate our theory by experiments on image classification and make our code available at `https://github.com/LPD-EPFL/robust-collaborative-learning`.

1. Introduction

Collaborative learning allows multiple machines (or *nodes*), each with a local dataset, to learn local models that offer a high accuracy on the union of their local datasets (Boyd et al., 2011). This paradigm facilitates the training of complex models over a large volume of data, while addressing concerns on data locality and ownership. The general task of collaborative learning can be formulated as follows. Consider a *parameter space* $\mathbb{R}^d$, a *data space* $\mathcal{X}$ and a *loss function* $q : \mathbb{R}^d \times \mathcal{X} \to \mathbb{R}$. Given a parameter $\theta \in \mathbb{R}^d$, a data point $x \in \mathcal{X}$ incurs a loss of value $q(\theta, x)$. The system comprises n nodes. Each node i samples data from distribution $\mathcal{D}_i$, and thus has a *local loss function* $Q^{(i)}(\theta) := \mathbb{E}_{x \sim \mathcal{D}_i}[q(\theta, x)]$. The goal for each node i is to compute $\theta_*^{(i)}$ minimizing the *global average loss*, i.e.,

$$\theta_*^{(i)} \in \arg\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{j=1}^{n} Q^{(j)}(\theta) \,. \tag{1}$$

Collaborative learning with D-SGD. The most standard way of solving the optimization problem (1) is through the use of the celebrated distributed SGD (D-SGD) method (Tang et al., 2018; Koloskova et al., 2020). Each node maintains a local parameter, approximating a solution of the optimization problem (1), which is updated iteratively in two phases. In the first phase, also called *the local phase*, each node updates its current parameter *partially* using a stochastic estimate of its local loss function's gradient. In the second phase, also called *the coordination phase*, the nodes exchange their partially updated parameters with each other over a network, and then each node replaces its current parameter by the *average* of all the partially updated parameters. While the former is essential for reducing the local loss functions, the latter yields reduction in the global average loss function. Alternately, as is the case in *federated learning* (Kairouz et al., 2021), the nodes may rely on a *trusted* coordinator (called the *server*) to execute the coordination phase involving the averaging operation.

Robustness issue. D-SGD is not very robust: a handful of faulty nodes, deviating from their prescribed algorithm, may prevent the remaining non-faulty (or *correct*) nodes from computing a valid solution (Su & Vaidya, 2016). Such behavior may result from software and hardware bugs, poisoned data, or malicious adversaries controlling part of the network. We consider a setting where at most f (out of n) nodes in the system are faulty and assume that these can

[1]EPFL, Lausanne, Switzerland. [2]Tournesol, [3]Calicarpa, Switzerland. Correspondence to: Sadegh Farhadkhani <sadegh.farhadkhani@epfl.ch>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).