Byzantine-robust decentralized optimization for Machine Learning

William Cappelletti— ING-MATH Master Project, Autumn 2020

Supervisors: Prof. M. Jaggi, Prof. E. Abbé, S.P.R. Karimireddy, L. He

Introduction

The goal : Numerical minimization of a stochastic function f:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left\{ \mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{D}} f(\mathbf{x}, \boldsymbol{\xi}) \right\} \quad \text{with} \quad \begin{cases} \mathbf{x} \in \mathbb{R}^d & \text{parameter vector,} \\ \boldsymbol{\xi} \sim \mathcal{D} & \text{random vector (unknown distibution } \mathcal{D}\text{).} \end{cases}$$

The setting : A *network of computers* with local data $\xi_i \sim \mathcal{D}$ and parameters $\mathbf{x}_i$ collaborates to find the optimal parameters $\mathbf{x}^*$. They can only *share local estimates* $\mathbf{x}_i$.

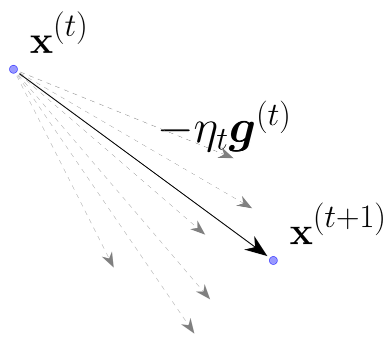
We study : The *convergence rates* of such decentalized algorithms and the *effects of adversarial agents* on the learning process.

Definitions

Stochastic Gradient Descent

We suppose that f is differentiable w.r.t. $\mathbf{x}$.
Our minimization strategy is based on Stochastic Gradient Descent.

Iteratively :
- Sample $\boldsymbol{\xi}^t \sim \mathcal{D}$;
- Compute the stochastic gradient $\boldsymbol{g}^{(t)} := \nabla f(\mathbf{x}^{(t)}, \boldsymbol{\xi}^t)$ at current estimate $\mathbf{x}^{(t)}$;
- Take a step towards $-\boldsymbol{g}^{(t)}$, scaled by a *learning rate* η_t, obtaining a new estimate $\mathbf{x}^{(t+1)}$.



Decentralized SGD

In *decentralized learning* we have a set of computers $\mathcal{V}$ in a communication graph $\mathcal{G} = (\mathcal{V}, E)$.

Properties :
- An edge $(e_{i,j}) = (i,j)$ is in E iff node i can communicate to node j;
- Each node i knows the set of its neighbors $\mathcal{N}_i$;
- Each computer keeps a local parameter vector, or *state*, $\mathbf{x}_i$ and can access local samples $\xi_i \sim \mathcal{D}$.

Algorithm 1: Decentralized SGD

Input: $\mathbf{x}^{(0)}$ initial guess, max T, $\{\eta_t\}_{t<T}$ learning rates, $\mathcal{G} = (\mathcal{V}, E)$ comm. graph

1. Init $\mathbf{x}_i^{(0)} \leftarrow \mathbf{x}^{(0)}$ for all $i \in \mathcal{V}$;
2. **for** $t = 0, \ldots, T-1$ **do**
   - // in parallel for each $i \in \mathcal{V}$
3. Collect $\mathbf{X}_i^{(t)} := \left\{ \mathbf{x}_j^{(t)} : j \in \mathcal{N}_i \right\}$;
4. $\hat{\mathbf{x}}_i^{(t)} \leftarrow$ Aggr $\left(\mathbf{x}_i^{(t)}, \mathbf{X}_i^{(t)}\right)$;
5. Sample $\xi_i^t \sim \mathcal{D}$;
6. $\boldsymbol{g}_i^{(t)} \leftarrow \nabla f(\hat{\mathbf{x}}_i^{(t)}, \xi_i^t)$;
7. Broadcast $\mathbf{x}_i^{(t+1)} \leftarrow \hat{\mathbf{x}}_i^{(t)} - \eta_t \boldsymbol{g}_i^{(t)}$;
8. **end**

Function Aggr

Input: A set of vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_M\} \subset \mathbb{R}^d$
Output: $\hat{\mathbf{v}}$ *robust estimate* of the mean $\bar{\mathbf{v}}$

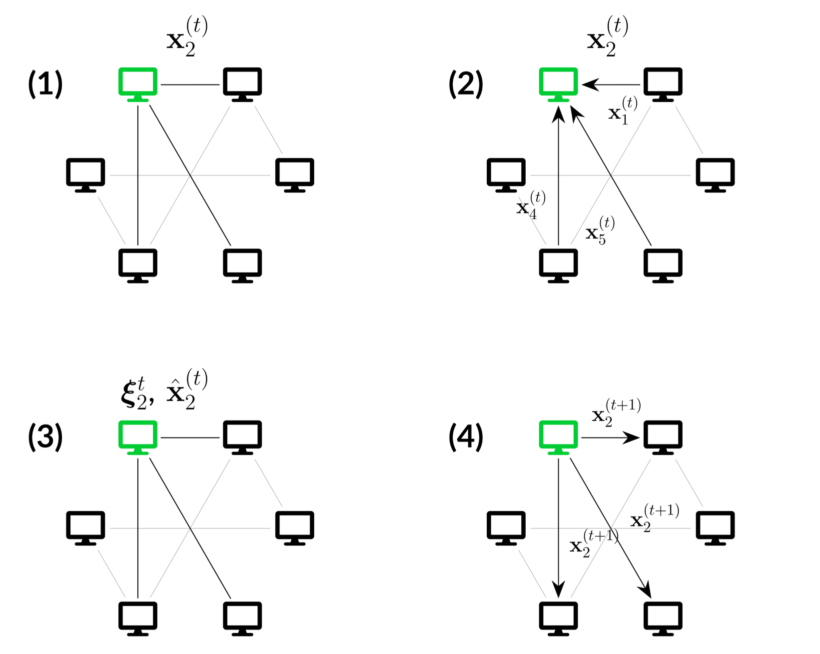


Fig. 2: Representation of one step t of Decentralized SGD. (1) The focus is on a single node and its neighbors. (2) The worker i gathers parameter vectors from neighboring nodes. (3) It aggregates them into $\hat{\mathbf{x}}_i^{(t)}$ and performs an SGD step with a local sample. (4) Finally, it broadcasts its updated parameters.

Regular workers agree on an aggregation strategy `Aggr`, which should be robust to attacks.

If `Aggr` is the arithmetic mean, Algorithm 1 is known as Gossip SGD. If there are *adversarial agents* averaging the parameters leads to severe failure.

Convergence analysis

Theorem 1 Average parameters recursion – Non-convex [2]

Let $\mathbf{X}^{(t)}$ be the set of vectors computed at time t by Algorithm 1 (Decentralized SGD) and let:

- f *be an L-smooth function,*
- *fixed learning rate* $\eta = \frac{1}{\sqrt{T+1}}$,
- $\bar{\mathbf{x}}^{(t)} = \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i^{(t)}$,
- *number of iterations T big enough,*
- $\mathbf{x}^{(0)}$ *be the common starting point,*
- `Aggr` *be a symmetric mixing matrix at all steps.*

Then, without any failure or adversaries in the graph,

$$\frac{1}{T+1}\sum_{t=0}^{T} \left\| \mathbb{E}\nabla f(\bar{\mathbf{x}}^{(t)}) \right\|^2 \le \mathcal{O}\left(\frac{\mathbb{E}\left[f(\bar{\mathbf{x}}^{(0)}) - f(\mathbf{x}^*)\right]}{\sqrt{T+1}} + \left(\frac{1}{N} + \frac{\lambda_2^2}{3}\right)\frac{\sigma^2}{\sqrt{T+1}} \right),$$

with

- N = *number of nodes,*
- λ_2 = *up. bound on second eig.val. of* `Aggr`,
- σ^2 = *up.bound on the trace of the covariance of stoch. gradients* $\mathbb{E}\left\|\nabla f(\mathbf{x}) - \mathbb{E}\nabla f(\mathbf{x})\right\|^2$.

Comments

With a fixed learning rate η the suboptimality of the average parameters $\bar{\mathbf{x}}^{(T)}$, i.e. the squared norm of their gradient, decrases sub-linearly in the number of iterations T.

The second term depends on the bound σ^2 on gradients stochasticity. We can reduce this noise by increasing the number of agents N, and by reducing λ_2, the second eigenvalue of `Aggr`.

λ_2 is 0 for a fully connected graph and, indicatively, increases the fewer edges are in the graph, getting to one for disconnected ones.

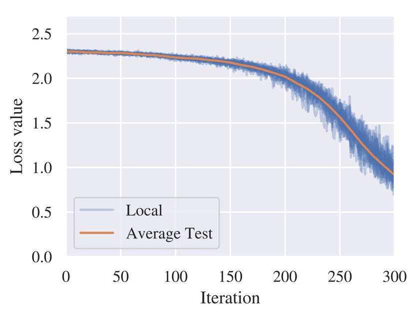


Fig. 3: 20 independent nodes.

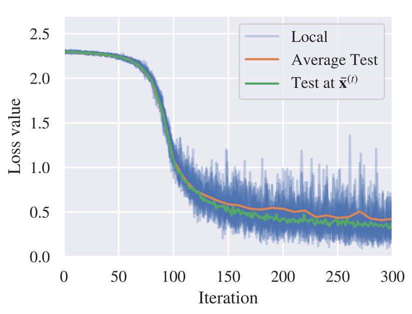


Fig. 4: 20 nodes, gossip average on fully connected graph.

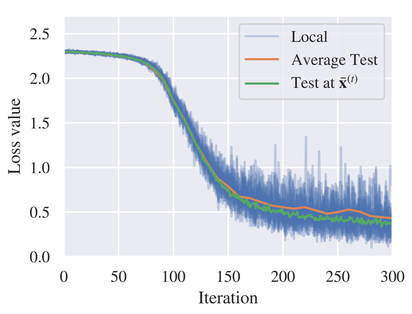


Fig. 5: 20 nodes, gossip average on 4-regular graph.

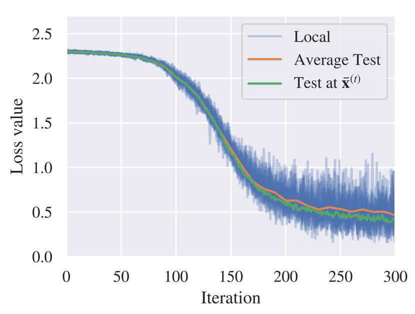


Fig. 6: 20 nodes, gossip average on cycle graph.

Figures. Experiments on training a CNN on the MNIST dataset confirm the theoretical analysis.

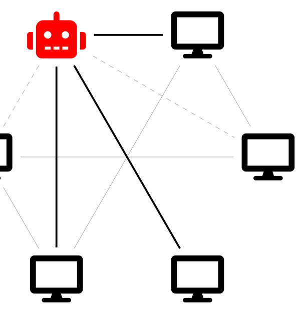
Byzantine robustness

We give a very broad definition, to allow for *powerful adversaries* and *worst case* scenarios.

Byzantine agents

Definition. Let $\mathcal{G} = (\mathcal{V}, E)$ be a communication graph, a node i is a *Byzantine agent* if

1. It has *complete knowledge* of the states of all nodes in the network, at each moment;
2. It can send *arbitrary messages* $\mathbf{x}_{i,j}^{(t)}$ to each neighboring node j.



Objective : Slow down, or even disrupt, the learning procedure.

Can good workers *benefit from collaboration*, by implementing a "good" `Aggr` function?

There is no agreement in the literature about the characterization of robustness. This precludes a theoretical comparison of different methods. We resort to *experimental analysis*.

Experiments

We analyze the learning curves of Decentralized SGD for the classification of MNIST through a Convolutional Neural Network.

Setting

- We randomly sample Erdos-Renyi graphs on 15 nodes, with an edge probability of 0.4 (average of 5.6 neighbors per node).
- Byzantine agents are added to such graphs and allowed to communicate to each regular node.

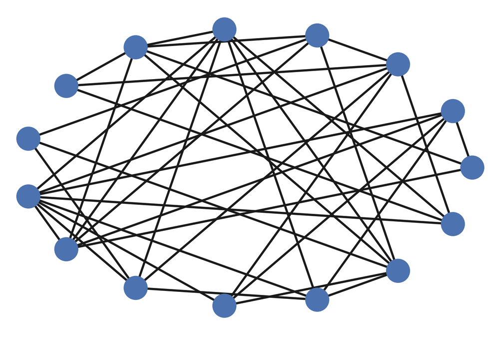


Fig. 7: Byzantine-free (15,0.4)-ER sample graph.

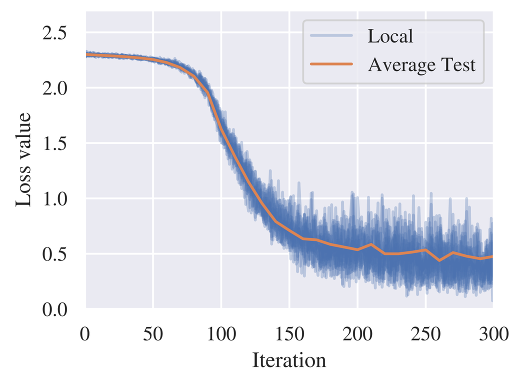


Fig. 8: Gossip SGD on (15,0.4)-ER graph.

Attacks

- `Gauss` attack: send a random sample from a multivariate standard Gaussian distribution.
- `LittleIsEnough` attack: estimate the mean and variance of the vectors shared by the good workers and send arbitrary erroneous messages which could go undetected [1].

DKrum

Choose as $\hat{\mathbf{x}}_i^{(t)}$ the vector which is closer to its $M-b-2$ neighboring vectors in euclidean distance [3]. M is the number of neighbors (counting oneself) and b is the up. bound on Byz. nodes; if $M < b+2$ revert to Local SGD.

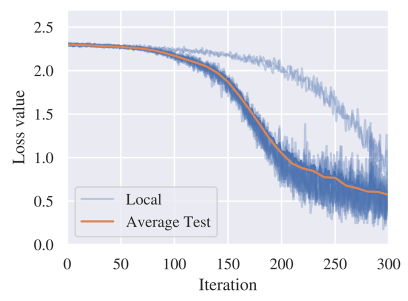


Fig. 9: 3 Byz: `Gauss`.

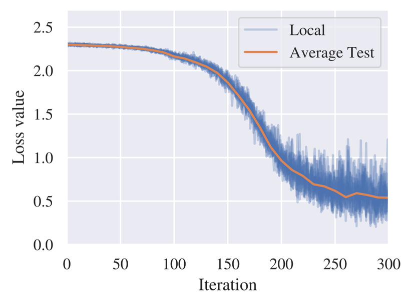


Fig. 10: 7 Byz: `Gauss`.

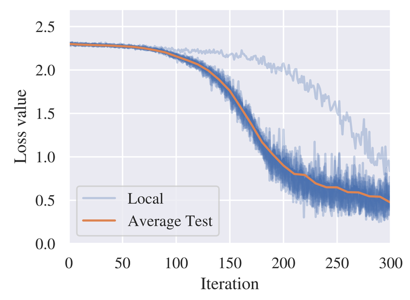


Fig. 11: 3 Byz: `LittleIsEnough`.

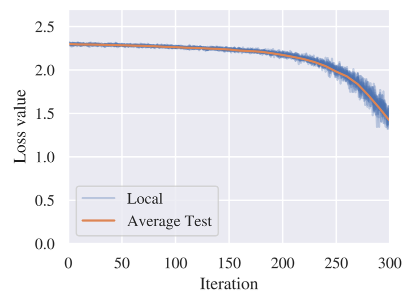


Fig. 12: 4 Byz: `LittleIsEnough`.

Results :
- Robust to `Gaussian` attack, but weak vs `LittleIsEnough`.
- Three `LittleIsEnough` attackers do the same damage as seven `Gaussian`.
- With four `LittleIsEnough` it performs worse than Local SGD (see Fig. 3).

BRIDGE

Compute $\hat{\mathbf{x}}_i^{(t)}$ coordinatewise by discarding the b (up. bound on Byz. nodes) lowest and highest values (*trimming*) and averaging the remaining with the local estimate [4]. If a node has less than $2b+1$ neighbors, it reverts to Local SGD.

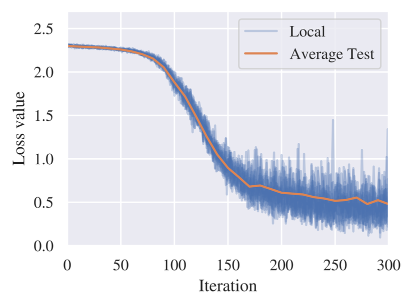


Fig. 13: 3 Byz: `Gauss`.

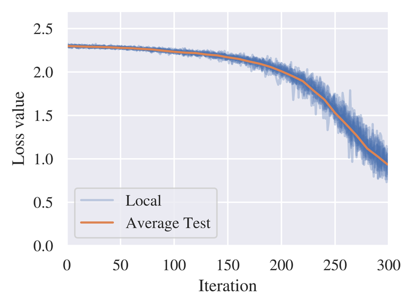


Fig. 14: 4 Byz: `Gauss`.

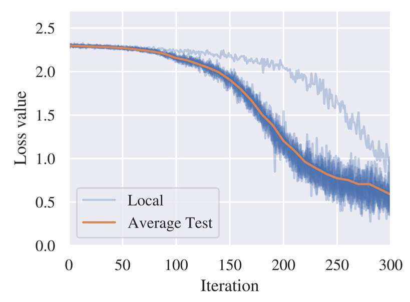


Fig. 15: 2 Byz: `LittleIsEnough`.

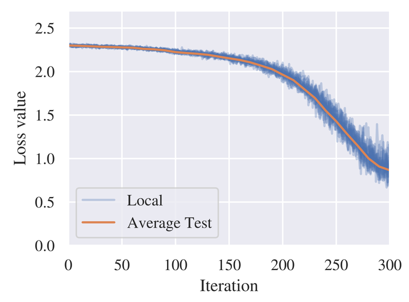


Fig. 16: 3 Byz: `LittleIsEnough`.

Results :
- This algorithm is very restrictive, as with four adversaries it reverts to Local SGD.
- Average convergence against three `Gauss` adversaries is better than `DKrum`.
- `LittleIsEnough` attack disrupts `BRIDGE` more than `DKrum`, with one fewer Byz. agent.

Conclusions

Many more attacks and defences have been proposed. Their assumptions are very different and, as we see with the proposed examples, there is a tradeoff between the imposed restrictions and the convergence speed. Also, the same defence can be robust against some attacks and very weak against others.
A univocal characterization of robustness would help in comparing weaknesses and strengths of different methods. Furtermore, it could help in deriving convergence bounds in line with those we present for a Byzantine-free setting.

References

[1] Moran Baruch, Gilad Baruch, and Yoav Goldberg. *A Little Is Enough: Circumventing Defenses For Distributed Learning*. 2019. arXiv: 1902.06156 [cs.LG].

[2] William Cappelletti. "Byzantine-robust decentralized optimization for Machine Learning". MA thesis. EPFL, 2021.

[3] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. *The Hidden Vulnerability of Distributed Learning in Byzantium*. 2018. arXiv: 1802.07927 [stat.ML].

[4] Zhixiong Yang and Waheed U Bajwa. "BRIDGE: Byzantine-resilient decentralized gradient descent". In: *arXiv preprint* (2019). arXiv: 1908.08098.

LaTeX TikZposter