Byzantine-robust decentralized optimization

William Cappelletti

EPFL, ING-MATH Master Project

4 February 2021

Presentation plan

Stochastic Optimization

The setting

Data: $\boldsymbol{\xi} \sim \mathcal{D}$

Parameters: $\mathbf{x} \in \mathbb{R}^d$

$\rightarrow$ $f(\mathbf{x}, \boldsymbol{\xi})$

$\mathbb{R}^d \times \mathcal{X} \to \mathbb{R}$

The goal

We are interested in **minimizing** the expected risk w.r.t. $\mathbf{x}$:

$$F(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\xi}} \left[f(\mathbf{x}, x) \right]$$

We do so through the empirical version. With $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n$ iid sample from $\mathcal{D}$:

$$\hat{F}(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}, \boldsymbol{\xi}_i)$$

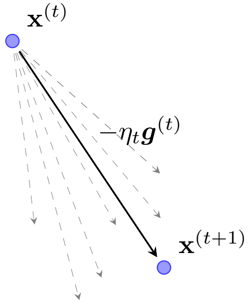
Stochastic Gradient Descent

Algorithm: SGD

Input: $\mathbf{x}^{(0)}$ initial guess, T number of iterations, $\{\eta_t\}_{t<T}$ learning rates.

for $t = 0, \ldots, T-1$ **do**
- Sample $\boldsymbol{\xi}^t \sim \mathcal{D}$;
- $\boldsymbol{g}^{(t)} \leftarrow \nabla f(\mathbf{x}^{(t)}, \boldsymbol{\xi}^t)$;
- $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \eta_t \boldsymbol{g}^{(t)}$;

return $\mathbf{x}^{(T)}$

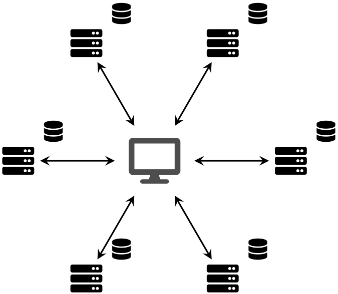


The more samples we include to compute $\boldsymbol{g}^{(t)}$, the more its estimate is precise. We call this mini-batch SGD.

Stochastic Gradient Descent

Limitations of (batch)-SGD

- Slow gradient computations
  - → Parallelization on distributed systems.
- NO Data privacy
  - → Give each node its own data: only share $\mathbf{x}^{(t)}$ and $\boldsymbol{g}_i^{(t)}$.
- Bottlenecks and prone to failures
  - → Need to change framework!



Decentralized Optimization

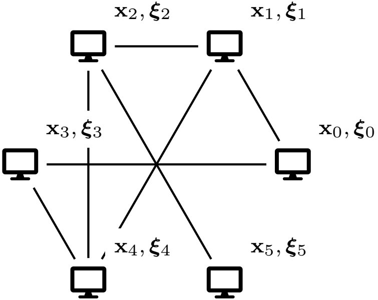
Define a decentralized setting

- We have a bunch of computers $\mathcal{V}$
- They generate a communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- Each node has its own state $\mathbf{x}_i$ and local data $\boldsymbol{\xi}_i$
- **Goal:** minimize local objectives

$$\arg\min_{\mathbf{x}_i, i \in \mathcal{V}} \sum_{i \in \mathcal{V}} \mathbb{E}_{\boldsymbol{\xi}} \left[f(\mathbf{x}_i, \boldsymbol{\xi}) \right]$$

and achieve consensus:

$$\mathbf{x}_i = \mathbf{x}_j \quad \forall i, j \in \mathcal{V}$$

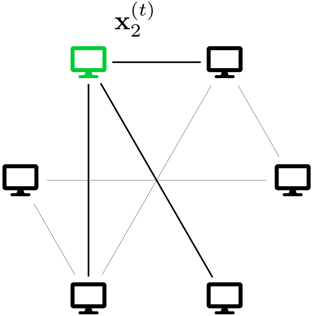


Decentalized Stochastic Gradient Descent

Algorithm: Gossip SGD

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

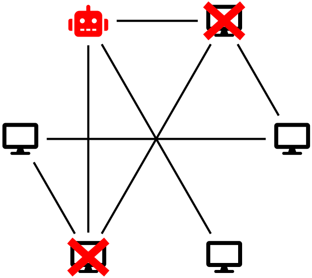
Init $\mathbf{x}_i^{(0)} \leftarrow \mathbf{x}^{(0)}$ for all $i \in \mathcal{V}$;
for $t = 0, \dots, T-1$ **do**
  `// in parallel for each` $i \in \mathcal{V}$
  Collect $\mathbf{X}_i^{(t)} := \left\{ \mathbf{x}_j^{(t)} : j \in \mathcal{N}_i \right\}$;
  $\overline{\mathbf{x}}_i^{(t)} \leftarrow \frac{1}{|\mathcal{N}_i|+1} \left(\sum_j \mathbf{x}_j^{(t)} + \mathbf{x}_i^{(t)} \right)$;
  Sample $\boldsymbol{\xi}_i^t \sim \mathcal{D}$;
  $\boldsymbol{g}_i^{(t)} \leftarrow \nabla f(\overline{\mathbf{x}}_i^{(t)}, \boldsymbol{\xi}_i^t)$;
  Broadcast $\mathbf{x}_i^{(t+1)} \leftarrow \overline{\mathbf{x}}_i^{(t)} - \eta_t \boldsymbol{g}_i^{(t)}$;



New problems

What if?

- What would happen if some node fails?
- What if some malicious entity infiltrates the network?



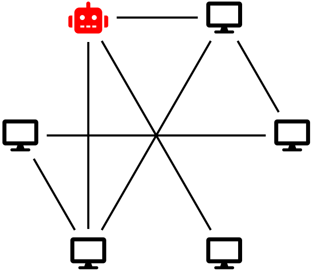
The Byzantine setting

Under the decentralized learning assumptions, we add Byzantine adversaries.

Definition

A *Byzantine agent* i has

- complete knowledge of the network state
- can send an arbitrary message $\mathbf{x}_{i,j}^{(t)}$ to each neighboring node j.



What are the objectives?

For a worker

- → Learn the optimal parameters
- → Speed up convergence w.r.t. working alone

For an attacker

- Break down the system
- Slow down convergence

Robust Decentalized SGD

Algorithm: Robust De-SGD

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

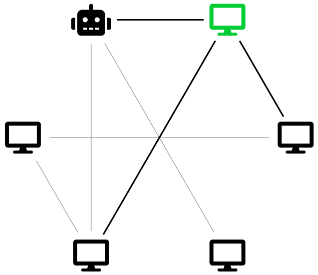
```
Init x_i^(0) ← x^(0) for all i ∈ V ;
for t = 0, ..., T − 1 do
    // in parallel for each i ∈ V
    Collect X_i^(t) := { x_j^(t) : j ∈ N_i } ;
    x̂_i^(t) ← Aggr( x_i^(t), X_i^(t) ) ;
    Sample ξ_i^t ~ D ;
    g_i^(t) ← ∇f(x̂_i^(t), ξ_i^t) ;
    Broadcast x_i^(t+1) ← x̂_i^(t) − η_t g_i^(t) ;
```

- Init $\mathbf{x}_i^{(0)} \leftarrow \mathbf{x}^{(0)}$ for all $i \in \mathcal{V}$;
- **for** $t = 0, \dots, T-1$ **do**
  - // in parallel for each $i \in \mathcal{V}$
  - Collect $\mathbf{X}_i^{(t)} := \left\{ \mathbf{x}_j^{(t)} : j \in \mathcal{N}_i \right\}$;
  - $\hat{\mathbf{x}}_i^{(t)} \leftarrow \mathtt{Aggr}\left(\mathbf{x}_i^{(t)}, \mathbf{X}_i^{(t)} \right)$;
  - Sample $\boldsymbol{\xi}_i^t \sim \mathcal{D}$;
  - $\boldsymbol{g}_i^{(t)} \leftarrow \nabla f(\hat{\mathbf{x}}_i^{(t}, \boldsymbol{\xi}_i^t)$;
  - Broadcast $\mathbf{x}_i^{(t+1)} \leftarrow \hat{\mathbf{x}}_i^{(t)} - \eta_t \boldsymbol{g}_i^{(t)}$;

Function Aggr

Input: A set of vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_M\} \subset \mathbb{R}^d$

Output: $\hat{\mathbf{v}}$ a robust estimate of the mean $\overline{\mathbf{v}}$ of *good nodes*



What has been done ?

Existing algorithms – Trimmed Mean

Function TrimmedMean

Input: b upper bound on # Byzantine, set $\{\mathbf{x}_1, \dots, \mathbf{x}_M\} \subset \mathbb{R}^d$

Init an empty $\hat{\mathbf{x}}$;
foreach $k \in [d]$ **do**
  Sort $\left\{[\mathbf{x}_1]_k, \dots, [\mathbf{x}_M]_k\right\}$ as $\{x_{(1)}, \dots, x_{(M)}\}$;
  `// Average, discarding the lowest and highest` b`:`
  $[\hat{\mathbf{x}}]_k \leftarrow \frac{1}{M-2b} \sum_{k=b+1}^{M-b} x_{(k)}$;
return $\hat{\mathbf{x}}$

Average each coordinate by excluding extreme values

PROS

- Easy to understand

CONS

- Each node needs at least $2b$ neighbors

BRIDGE

Applying TrimmedMean to the parameters from neighboring nodes and always including the local parameters corresponds to the BRIDGE algorithm.

Existing algorithms – Krum and Bulyan

Function Krum

Input: b upper bound on # Byzantine,
set $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\} \subset \mathbb{R}^d$

```
foreach i ∈ [M] do
    Identify the M − b − 2 closest points
     to x_i into Ñ_i;
    s_i ← Σ_{j∈Ñ_i} ‖x_i − x_j‖² ;
return x̂ ← arg min_{i∈[M]} {s_i}
```

Function Bulyan

Input: b upper bound on # Byzantine,
set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\} \subset \mathbb{R}^d$

```
Select ← ∅ ;
while |Select| < M − 2b do
    x_s ← Krum(X \ Select) ;
    Select ← Select ∪ {x_s} ;
return x̂ ← TrimmedMean(Select)
```

- `Krum`: Find a candidate which is central even after removing nodes
- `Bulyan`: Take another `Aggr` rule and make it stronger

PROS

- *Convergence* in the parameter server setting

CONS

- Very strict assumptions for analysis.
- Convergence does not imply optimality.
- Each node needs at least $2b$ neighbors, $> 4b$ for Bulyan.

Existing algorithms – ByGARS

Algorithm: ByGARS++

Init $\mathbf{x}_i^{(0)} \leftarrow \mathbf{x}^{(0)}$ for all i ;
Init $\mathbf{q}_i^{(0)} \leftarrow \mathbf{0}$ for all i ;
foreach i *good worker, at step* t **do**
  Collect $H_t := \left\{ \mathbf{g}_j^{(t)} : j \in \mathcal{N}_i \right\}$;
  Sample $\boldsymbol{\xi}_i^t \sim \mathcal{D}$;
  $\boldsymbol{g}_i^{(t)} \leftarrow \nabla f(\hat{\mathbf{x}}_i^{(t)}, \boldsymbol{\xi}_i^t)$;
  $\mathbf{x}_i^{(t+1)} \leftarrow \mathbf{x}_i^{(t)} - \eta_t H_t \boldsymbol{q}_i^{(t)}$;
  $\mathbf{q}_i^{(t+1)} \leftarrow (1 - \alpha^t)\mathbf{q}^t + \alpha^t H_t \boldsymbol{g}_i^{(t)}$;
  Broadcast $\boldsymbol{g}_i^{(t)}$;

Score neighbors by how much they align to the validation grad

PROS

- Validate recieved *gradients* against local
- Memory of the past through *scores* $\mathbf{q}_i$

CONS

- Not sharing parameters (adapted from distributed)
- Analysis only assume (almost) fixed multiplicative noise
- Does not use local grad to move

Existing algorithms – MOZI

Algorithm: MOZI

Input: $\mathbf{x}^{(0)}$, max T, $\{\eta_t\}_{t<T}$, # byz ngbs b_i, tolerance ϵ

Init $\mathbf{x}_i^{(0)} \leftarrow \mathbf{x}^{(0)}$ for all $i \in \mathcal{V}$;
for $t \in [T-1]$ **do**
  `// in parallel for each` $i \in \mathcal{V}$
  Collect $\mathbf{x}_j^{(t)}$ for $j \in \mathcal{N}_i$;
  Sample $\boldsymbol{\xi}_i^t \sim \mathcal{D}$;
  $l_i^t \leftarrow f(\mathbf{x}_i^{(t)}, \boldsymbol{\xi}_i^t)$;
  $\boldsymbol{g}_i^{(t)} \leftarrow \nabla f(\mathbf{x}_i^{(t)}, \boldsymbol{\xi}_i^t)$;
  for $j \in \mathcal{N}_i$ **do**
    $d_{i,j} \leftarrow \left\|\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}\right\|$;
  $Close \leftarrow \arg\min_{\substack{\mathcal{N}^* \subseteq \mathcal{N}_i, \\ |\mathcal{N}^*| = M - b_i}} \sum_{j \in \mathcal{N}^*} d_{i,j}$;
  $Sel \leftarrow \emptyset$;
  for $j \in Close$ **do**
    $l_j^t \leftarrow f(\mathbf{x}_j^{(t)}, \boldsymbol{\xi}_i^t)$;
    if $l_i^t - l_j^t \geq \epsilon$ **then**
      $Sel \leftarrow Sel \cup \{j\}$;
  if Sel *is* $\emptyset$ **then** $Sel \leftarrow \{\arg\min_{j \in Close} l_j^t\}$;
  $\mathcal{R}_i^t \leftarrow \frac{1}{|Sel|} \sum_{j \in Sel} \mathbf{x}_i^t$;
  Broadcast $\mathbf{x}_i^{(t+1)} \leftarrow \alpha \mathbf{x}_i^{(t)} + (1-\alpha)\mathcal{R}_i^t - \eta_t \boldsymbol{g}_i^{(t)}$;

Select a pool of canditates and further filter out those with higher loss than local estimate

PROS

- Check candidates' *distance* AND *loss value*
- Does at least as well as being alone

CONS

- Compute loss at many values
- Need to know b_i for each node
- Many hyperparameters (α, η, ϵ)
- Not very elegant

Existing algorithms – Summary

- Many different assumptions and definitions:
  Lack of a unified framework.
- Most methods have been adapted directly from the federated learning setting.
- Almost all are based on euclidean distance
- Analysis only focuses on the average of the parameters, or some linear combination of the local losses.
- Analysis is always asymptothical and is almost never compared to Local SGD.

Convergence Analysis

We would like to bound the function suboptimality, under reasonable assumptions.
Proof idea :

- Approximate Byzantine-resilient DeSGD by a Byzantine-free weighted Gossip SGD algorithm:

$$\widehat{\mathbf{X}}_{\mathcal{R}}^{(t)} := \texttt{Aggr}(\mathbf{X}^{(t)}) \approx \mathbf{X}_{\mathcal{R}}^{(t)} \boldsymbol{M}_t$$

💡 $\boldsymbol{M}_t$ is a weighted mixing matrix given by the graph.

- Compute finite time convergence rates for weighted Gossip SGD:

1. Focus on the average "good" parameters $\overline{\mathbf{x}}^{(t)} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i(t)$
2. Obtain recursion for mean error term $F(\overline{\mathbf{x}}^{(T)}) - F(\mathbf{x}^*)$
3. Bound the consensus variation $\Xi_T = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}\left\|\hat{\mathbf{x}}_i^{(T)} - \overline{\mathbf{x}}^{(T)}\right\|^2$ and its weighted time average $\sum_{t=0}^{T} w_t \Xi_t$.
4. Combine 2. and 3. to bound the suboptimality $\mathbb{E}\left\|\nabla F(\overline{\mathbf{x}}^{(T)}, \boldsymbol{\xi})\right\|^2$.

Convergence Analysis

Theorem (Average parameters recursion – Non-convex)

- *Let f be an L-smooth function*
- *Let number of iterations T big enough*
- *Take fixed learning rate $\eta = \frac{1}{\sqrt{T+1}}$*
- *Let average of parameters $\overline{\mathbf{x}}^{(t)} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i(t)$*
- *Suppose* `Aggr` $\equiv M_t$ *symmetric mixing matrix*

Then,

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

with

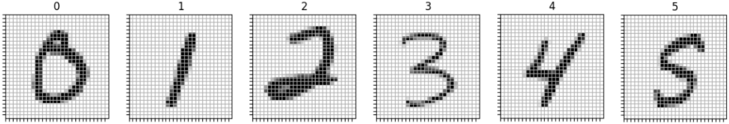
$N =$ *number of nodes*
$\lambda_2 =$ *up. bound on second eig.val. of M_t*

$$\sigma^2 = \sup \mathbb{E}\left\|\nabla f(\mathbf{x}) - \mathbb{E}\nabla f(\mathbf{x})\right\|^2$$

Note: Stricter bound for *strongly convex* objectives.

Experimental analysis

We perform experiments on the MNIST dataset. We train a Convolutional Neural Network for the Handwritten Digit Classification task.



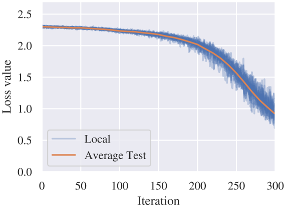
- $T = 300$ iterations
- Learning rate $\eta = 0.2$ for 100 steps then $\eta = 0.1$
- Minibatches of 32 samples per node

Objectives

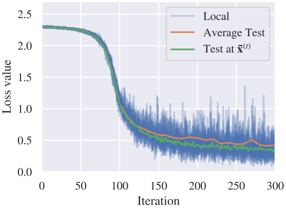
1. Understand the implications of the convergence bound.
2. Study the effects of Byzantine attacks on some aggregation rules.

Experimental analysis – Byzantine free

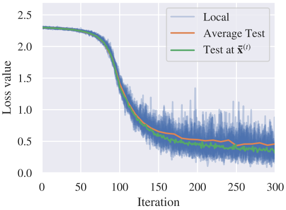
We analyze how connectivity changes the learning curve (recall theorem)



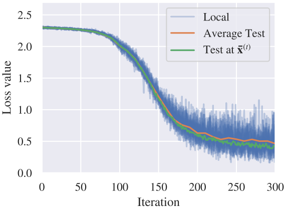
(i) 20 indep. nodes



(ii) 20 nodes fully connected



(iii) 8-regular graph, 20 nodes



(iv) Cycle graph, 20 nodes

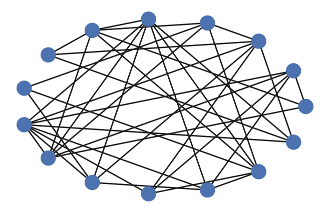
Experimental analysis – Byzantine robustness

Objective : Analyze the learning curves of two Robust DeSGD methods against two different attacks

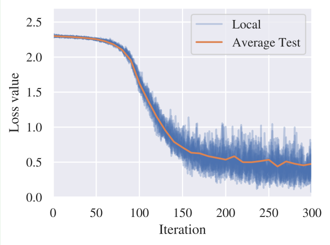
Procedure

- Randomly sample $(15, 0.4)$-Erdos-Renyi graphs

Example



(i) Byzantine-free $(15, .4)$-ER graph



(ii) Gossip SGD on $(15, .4)$-ER graph

Experimental analysis – Byzantine robustness

Objective : Analyze the learning curves of two Robust DeSGD methods against two different attacks.

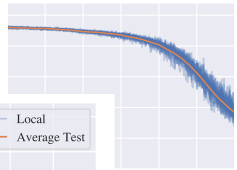
Procedure

- Randomly sample $(15, 0.4)$-Erdos-Renyi graphs
- Add Byzantine agents and allow them to communicate to each regular node

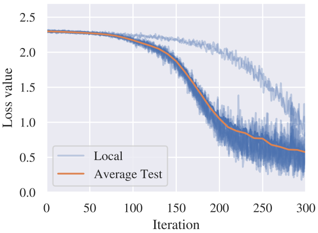
Byzantine attacks

- `Gauss`: send a random sample from a multivariate standard Gaussian distribution
- `LittleIsEnough`: estimate the mean and variance of the vectors shared by the good workers and send an erroneous message which could go undetected

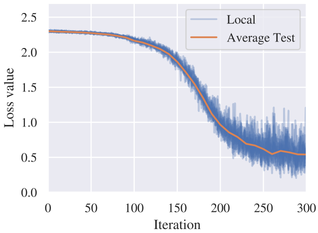
Experimental analysis – DKrum



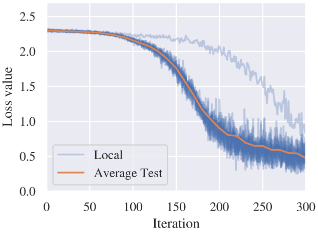
Local SGD



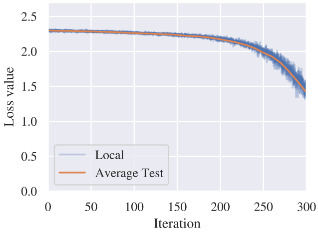
(i) 3 Byzantines-Gauss



(ii) 7 Byzantines-Gauss

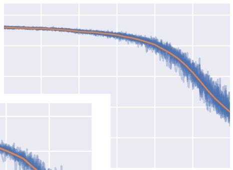


(iii) 3 Byz.-LittleIsEnough

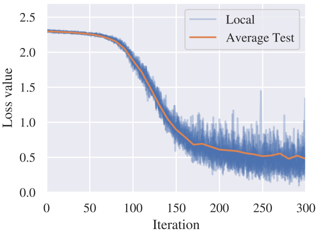


(iv) 4 Byz.-LittleIsEnough

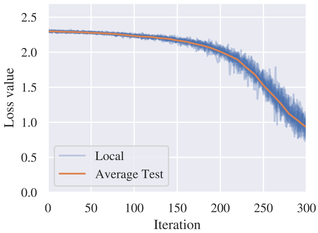
Experimental analysis – BRIDGE



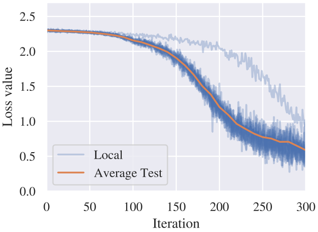
Local SGD



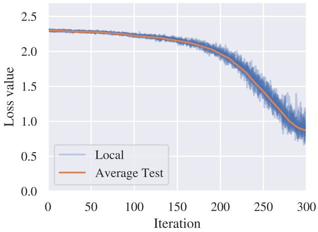
(i) 3 Byzantines-Gauss



(ii) 4 Byzantines-Gauss



(iii) 2 Byz.-LittleIsEnough



(iv) 3 Byz.-LittleIsEnough

Experimental analysis – Comments

- We only showed two methods among many others
- Tradeoff between *restrictive assumptions* and *convergence speed*
- A defence strategy 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

Recap

- Motivate and define Decentralized SGD
- Introduce Byzantine adversaries
- Review variants of DeSGD which claim robustness
- Prove convergence rates in Byzantine-free setting
- Analyse experimentally the learning curves for MNIST-classification with different graphs and settings

Future work

- Find a unifying characterization of Byzantine robustness
  - → Allow for varying number of Byzantine and regular nodes. Include proposed methods by limiting the assumptions
- Approximate `Aggr` in linear form with only good nodes, bounding the error introduced by Byzantine agents
  - → This let us easily generalize the convergence bounds
- Generalize convergence proof to non-symmetric mixing
  - → Very few graphs allow symmetric weghts, but the results seem to hold for a general setting.
- Find a proof strategy to bound local convergence rates
  - → Allows to compare proposed methods against Local SGD

Thank you for your attention!

😊

Appendix

Assumptions

Assumption 1 (Bounded gradients)

The stochastic component $\boldsymbol{\delta}(\mathbf{x}) = \boldsymbol{g}(\mathbf{x}) - \hat{\boldsymbol{g}}(\mathbf{x})$ follows a distribution $(0, \Sigma_{\mathbf{x}})$ and has bounded squared norm, for all $\mathbf{x} \in \mathbb{R}^d$ I.e. for all $\mathbf{x} \in \mathbb{R}^d$

$$\mathbb{E} \left\| \boldsymbol{\delta}(\mathbf{x}) \right\|^2 \leq \sigma^2. \tag{1}$$

Assumption 2 (Byzantine-free)

All of the N agents in the graph are regular workers following Decentalized SGD.

Assumption 3 (Symmetric mixing)

We suppose that the mixing matrices $\boldsymbol{M}_t$ are symmetric, and thus doubly stochastic, for all $t \geq 0$.

Assumptions

Assumption 4 (Nonnul spectral gap)

The second eigenvalue $\lambda_{2,T}$ of $\boldsymbol{M}_t$ is striclty smaller than 1 for all $t \geq 0$.
Note that since $\lambda_{2,T} < 1$, then $\lambda_{2,T}^2 < \lambda_{2,T}$. This implies that the spectral gap ρ_t of $\boldsymbol{M}_t^2$, defined as the difference between the first two eigenvalues, is always greater than zero.
Also, there exist a positive lower bound $\rho = \inf_t \{\rho_t\}$ on the spectrals gaps.

Assumption 5 (Smoothness)

The empirical risk function f is L-smooth, as defined in (**??**), with respect to the parameter vector $\mathbf{x}$, for any random vector $\boldsymbol{\xi}$.

Lemmas

Useful notation

$$\Xi_T = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left\|\hat{\mathbf{x}}_i^{(T)} - \overline{\mathbf{x}}^{(T)}\right\|^2,$$
$$r_T = \mathbb{E}\left[f(\overline{\mathbf{x}}^{(T)}) - f(\mathbf{x}^*)\right] = F(\overline{\mathbf{x}}^{(T)}) - F(\mathbf{x}^*),$$
$$e_T = \left\|\nabla F(\overline{\mathbf{x}}^{(T)}, \boldsymbol{\xi})\right\|^2.$$

Lemma (Error recursion - Non-convex)

Let assumptions 1, 2, 3, 4, 5 hold. The average of the parameters at iteration T produced by De-SGD, with constant leraning rate η satisfies

$$r_{T+1} \leq r_T + \left(L\eta^2 - \frac{\eta}{2}\right) e_T + \frac{L^2\eta + 2L^3\eta^2}{2}\Xi_T + \frac{L\eta^2}{2N}\sigma^2. \quad (2)$$

Lemmas

Lemma (Consensus convergence - Non-convex)

Let assumptions 1, 2, 3, 4, 5 hold. With $\rho_T = 1 - \lambda_{T,2}^2$ the spectral gap of the squared mixing matrix $\boldsymbol{M}_T^2$, we have

$$\Xi_T \leq \left(1 - \frac{\rho_T}{2} + \frac{6L^2\eta^2}{\rho_T}\right)\Xi_{T-1} + \frac{6\eta^2}{\rho_T} e_{T-1} + (1-\rho_T)\,\eta^2\sigma^2. \quad (3)$$

Furthermore, if we use a fixed learning rate $\eta \leq \frac{\rho}{2\sqrt{6}L}$, with ρ a lower bound on the spectral gaps; and we define a series of weights $\{w_t\}_{t\geq 0} \subset \mathbb{R}_+$ such that $w_{t+1} \leq w_t \left(1 + \frac{\rho}{8}\right)$, we can bound

$$\sum_{t=0}^{T} w_t \Xi_t \leq \frac{48L}{\rho^2}\eta^2 \sum_{t=0}^{T} w_t e_t + 4\eta^2\sigma^2 \left(\frac{1}{\rho} - 1\right) W_T. \quad (4)$$

Theorems

Theorem (Average parameters recursion – Non-convex)

Let assumptions 1, 2, 3, 4, 5 hold. Denote by $\overline{\mathbf{x}}^{(t)}$ *the average of the parameters across all nodes at iteration* t *and let* $\mathbf{x}^{(0)}$ *be the common starting point. Then, taking a fixed learning rate* $\eta \leq \min\left\{\frac{\rho}{12\sqrt{2}L}, \frac{\rho}{12\sqrt{2}L^{1.5}}\right\}$*, we have the following convergence rate:*

$$\frac{1}{T+1}\sum_{t=0}^{T} e_t \leq \frac{4r_0}{(T+1)\eta} + 2L\eta\left(\frac{1}{N} + \frac{1-\rho}{3}\right)\sigma^2. \tag{5}$$

Corollary

Under Theorem 23 conditions, with fixed learning rate $\eta = \frac{1}{\sqrt{T+1}}$*, we have*

$$\frac{1}{T+1}\sum_{t=0}^{T} e_t \leq \tilde{\mathcal{O}}\left(\frac{r_0}{\sqrt{T+1}} + \left(\frac{1}{N} + \frac{1-\rho}{3}\right)\frac{\sigma^2}{\sqrt{T+1}}\right). \tag{6}$$

Strongly convex case

Useful notation

$$\Xi_T = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left\|\hat{\mathbf{x}}_i^{(T)} - \overline{\mathbf{x}}^{(T)}\right\|^2$$
$$r_T = \mathbb{E}\left\|\overline{\mathbf{x}}^{(T)} - \mathbf{x}^*\right\|^2$$
$$e_T = \mathbb{E}\left[f(\overline{\mathbf{x}}^{(T)}) - f(\mathbf{x}^*)\right] = F(\overline{\mathbf{x}}^{(T)}) - F(\mathbf{x}^*)$$

Assumption 6 (Strong convexity)

The risk function f is μ-strongly convex with respect to the parameter vector $\mathbf{x}$, for all random vectors $\boldsymbol{\xi} \in \mathcal{X}$.
We say that f is μ-*strongly convex*, for some $\mu > 0$ if $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ the following holds:

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x}\rangle + \frac{\mu}{2}\|\mathbf{y} - \mathbf{x}\|^2. \tag{7}$$

Strongly convex case – Theorem

Theorem (Average convergenge rate – μ-convex)

Let assumptions 1, 2, 3, 4, 5, 6 hold. Denote by $\overline{\mathbf{x}}^{(t)}$ the average of the parameters across all nodes at iteration t and let $\mathbf{x}^{(0)}$ be the common starting point. Then, taking a fixed learning rate $\eta_t = \eta \leq \frac{\rho}{8\sqrt{6}L}$, and a sequence of weights $w_t = \left(1 - \frac{\eta\mu}{2}\right)^{-(t+1)}$. we have the following convergence rate:

$$\frac{1}{W_T}\sum_{t=0}^{T} w_t e_t + \frac{\mu}{2} r_{T+1} \leq \frac{r_0}{\eta} \exp\left\{-(T+1)\frac{\eta\mu}{2}\right\} + \eta\left(\frac{1-\rho}{2} + \frac{1}{N}\right)\sigma^2. \quad (8)$$

Corollary

With a fixed learning rate $\eta \leq \min\left\{\frac{2\ln(T^2)}{\mu T}, \frac{\rho}{8\sqrt{6}L}\right\}$, we have

$$\frac{1}{W_T}\sum_{t=0}^{T} w_t e_t + \frac{\mu}{2} r_{T+1} \leq \tilde{\mathcal{O}}\left(\frac{r_0}{T} + \left(\frac{1-\rho}{2} + \frac{1}{N}\right)\frac{\sigma^2}{T}\right). \quad (9)$$