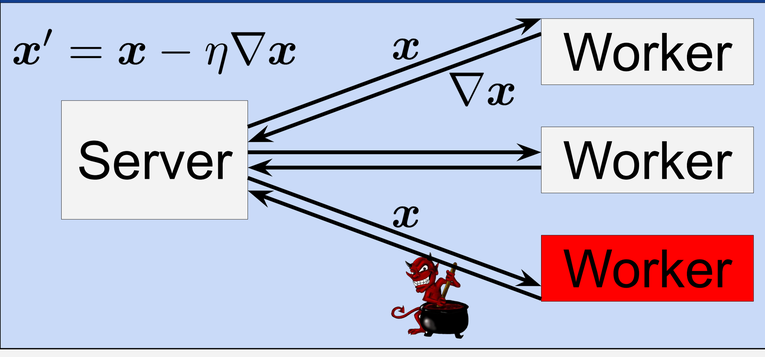
Asynchronous Byzantine Machine Learning (the case of SGD)

Georgios Damaskinos, El Mahdi El Mhamdi
Rachid Guerraoui, Rhicheek Patra, Mahsa Taziki

ICML

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

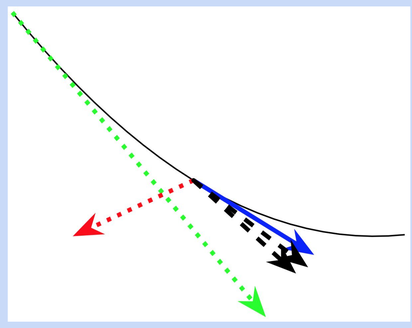
1. Byzantine workers



Examples:

- crash
- software bug
- corrupted data
- security flaw

2. Asynchronous - Byzantine SGD



cost function: thin black curve
actual ***g***: solid blue
non-stale honest ***g***: black dashed
stale honest ***g***: green dotted
poisoning ***g***: red dotted

3. Notation

t	Model epoch incremented after each update
l_p	Timestamp of the last model used by worker p
$\boldsymbol{x}_t$	Model (parameter vector) at epoch t
γ_t	Learning rate at epoch t
$\boldsymbol{g}_p$	Each gradient is a tuple $[\boldsymbol{g}_p, l]$ denoting that a worker p computed the gradient $\boldsymbol{g}_p$ w.r.t $\boldsymbol{x}_l$
$\mathcal{G}_t$	Set of gradients that the server receives in epoch t
τ_{tl}	Staleness value for a gradient $[\boldsymbol{g}, l]$ at epoch t ($\tau_{tl} \triangleq k - l$)
ξ	Mini-batch of training examples

4. *Kardam*

- Byzantine resilience against f/n workers, f <= n/3
- Optimal slowdown: $\frac{n-2f}{n-f} \leq SL \leq \frac{n-f}{n}$
- Provable (almost sure) convergence

5. Filtering component

a. <u>Lipschitz filter</u>
empirical Lipschitz coefficient at worker p:

$$\hat{K}_p = \frac{\|\boldsymbol{g}_p - \boldsymbol{g}_p^{prev}\|}{\|\boldsymbol{x}_{l_p} - \boldsymbol{x}_{l_p}^{prev}\|}$$

empirical Lipschitz coefficient at server after response from p:

$$\hat{K}_t^p = \frac{\|\boldsymbol{g}_p - \boldsymbol{g}_q\|}{\|\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\|}$$

$$\hat{K}_t^p \leq \hat{K}_t \triangleq quantile_{\frac{n-f}{n}} \{\hat{K}_p\}_{p \in P}$$

b. <u>Frequency filter</u>
limits the number of successive gradients from a single worker to a value of f

a ^ b => *correct cone*:

$$\langle \mathbb{E}_\xi \boldsymbol{G}(\boldsymbol{x}; \xi), \boldsymbol{\nabla} Q(\boldsymbol{x}) \rangle > \Omega((\|\boldsymbol{\nabla} Q(x_t)\| - \sqrt{d}\sigma)\|\boldsymbol{\nabla} Q(x_t)\|)$$

6. Dampening component

SGD **generic staleness-aware** update rule:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \gamma_t \sum_{[\boldsymbol{G}(\boldsymbol{x}_l; \xi_m), l] \in \mathcal{G}_t} \Lambda(\tau_{tl}) \cdot \boldsymbol{G}(\boldsymbol{x}_l; \xi_m)$$

=> convergence rate bound

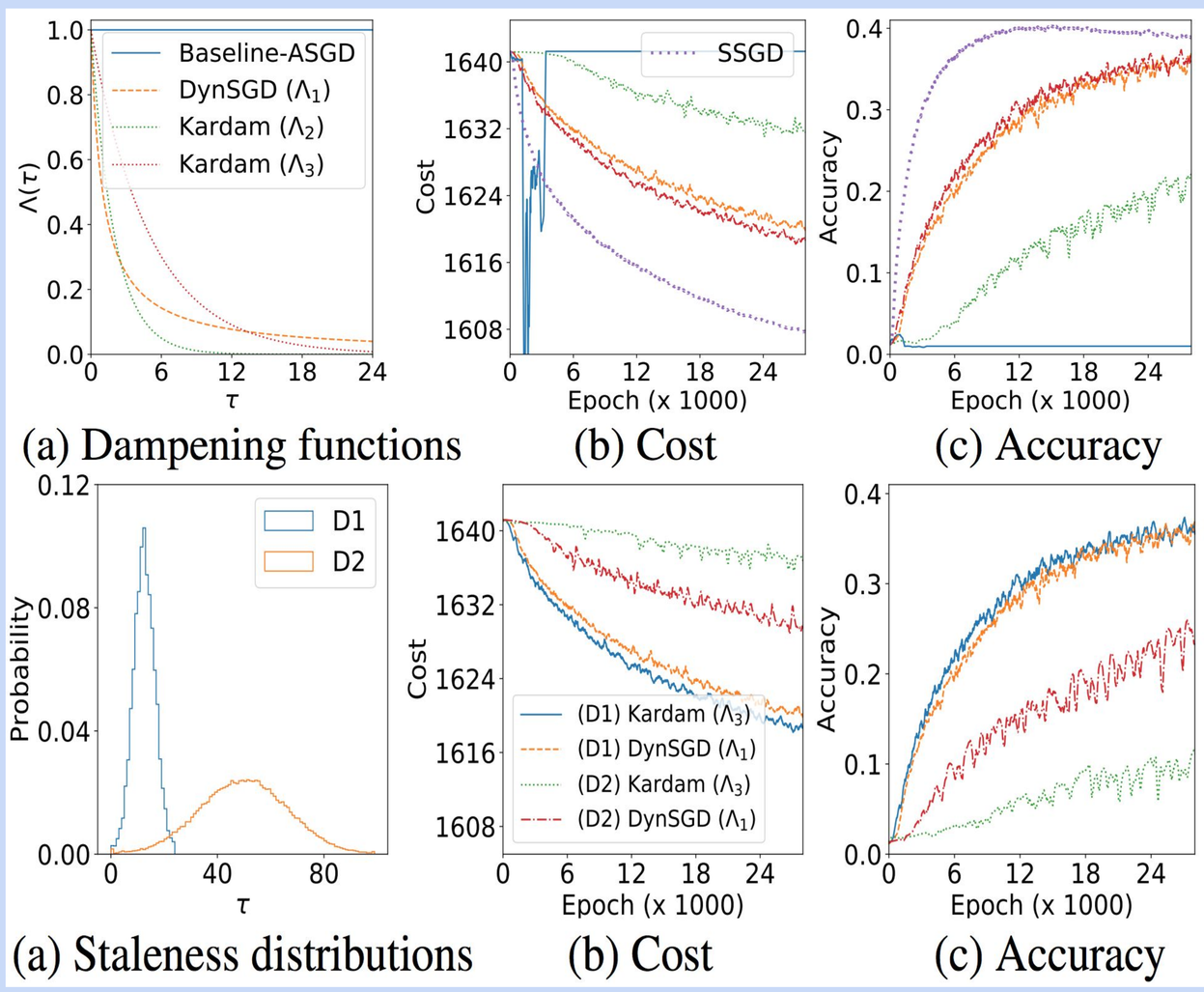
7. Evaluation

<u>Setup</u>

- CIFAR-100
- CNN
- f = 3, n = 10 workers

Parameters	Input	Conv1	Pool1	Conv2	Pool2	FC1	FC2	FC3
Kernel size	32×32×3	3×3×16	3×3	3×3×64	4×4	384	192	100
Strides		1×1	3×3	1×1	4×4			

- Baseline-ASGD: no dampening component
- SSGD: ideal (synchronous) SGD
- $\Lambda_1 = 1/(\tau+1)$, $\Lambda_2 = \exp(0.5\tau)$, $\Lambda_3 = \exp(0.2\tau)$



<u>Slowdown</u>

Λ_1: 27.9 % filtered gradients (D_1)
Λ_3: 19.6 % filtered gradients (D_1)

Take away

- gradient filtering => Byzantine resilience
- gradient dampening => Asynchronous convergence
- Asynchrony can be viewed as Byzantine behavior

International Conference on Machine Learning (ICML) 2018, Stockholm, Sweden

Contact: firstname.lastname@epfl.ch