A Delay-tolerant Proximal-Gradient Algorithm for Distributed Learning

K. Mishchenko[1], F. Iutzeler[2], J. Malick[2], and M.-R. Amini[2]

[1]KAUST [2]Univ. Grenoble Alpes [3]CNRS

Abstract

Setting:

- Distributed Data distributed across M slaves
- Asynchronous Slaves send their results irregularly
- Regularized Regularized by a non-smooth function, such as ℓ_1-penalty
- Unbalanced Computational power of slaves can be very uneven
- Scarce communications Transmission cost can be very high

Results:

- We propose an algorithm based on gradient and proximal steps
- For strongly convex and smooth functions, rate is linear even when delays are unbounded

Applications:

- ℓ_1 regularized logistic regression, group lasso minibatch processing, etc.

Problem formulation

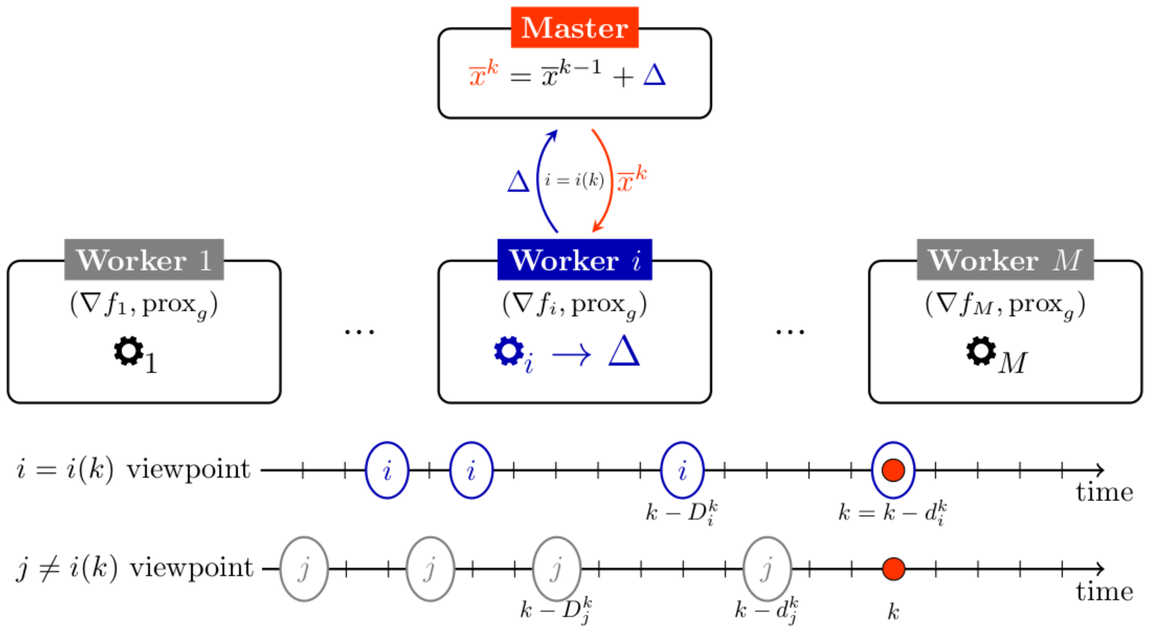
$$\min_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell_i(x) + r(x),$$

Global Objective ↓ Local Data

$$\min_{x\in\mathbb{R}^d} \sum_{i=1}^{M} \underbrace{\left(\frac{n_i}{n}\right)}_{:=\pi_i \text{ proportion}} \underbrace{\left(\frac{1}{n_i}\sum_{j\in\mathcal{S}_i} \ell_j(x)\right)}_{:=f_i(x) \text{ local empirical loss}} + r(x)$$

- $\mathcal{S}_i$ local dataset at i
- Each f_i is convex, L-Smooth $\|\nabla f_i(x) - \nabla f_i(y)\| \leqslant L\|x-y\|$ μ-strongly convex
- r is convex

Asynchronous framework



- *iteration* = receive from a worker + master update + send back
- *time* k = number of iterations
- *delay* d_i^k = time since last exchange with i
  $d_i^k = 0$ iff i updates at time k, $d_i^k = d_i^{k-1} + 1$ elsewhere
- *second delay* D_i^k = time since penultimate exchange with i

DAve communication scheme

$$\overline{x}^k = \overline{x}^{k-1} + \Delta \text{ with } \Delta = \pi_i(x_i^k - x_i^{k-D_i^k}) \text{ for } i = i(k)$$

$$\text{i.e.} \quad \overline{x}^k = \sum_{i=1}^{M} \pi_i x_i^{k-d_i^k} = \sum_{i=1}^{M} \pi_i \blacksquare_i(\overline{x}^{k-D_i^k})$$

- master variable $\overline{x}^k$ = combination of workers last contributions $(x_i^{k-d_i^k})_i$
  π_i = proportion of data at worker i
- one update/time = one worker contribution **but** all workers are always involved in the master variable with even proportions

Revisiting the clock

- *epoch sequence* (k_m) recursively defined by $k_0 = 0$ and

$$\begin{aligned} k_{m+1} &= \min\{k : \text{ each worker made at least 2 updates on the interval } [k_m, k]\} \\ &= \min\{k : k - D_i^k \geq k_m \text{ for all } i = 1, .., M\} \end{aligned}$$

- *epoch time* m = number of epochs

DAve-RPG

Worker i performs **R**epeated **P**roximal **G**radient steps

Master:

Initialize $\overline{x}$
while *not converged* **do**
  when a worker finishes:
  Receive adjustment Δ from it
  $\overline{x} \leftarrow \overline{x} + \Delta$
  Send $\overline{x}$ to the agent in return
  $k \leftarrow k + 1$
end
Interrupt all slaves
Output $x = \text{prox}_{\gamma g}(\overline{x})$

Worker i:

Initialize $x = x_i = \overline{x}$,
while *not interrupted by master* **do**
  Receive the most recent $\overline{x}$
  Select a number of repetitions p
  Initialize $\Delta = 0$
  for $q = 1$ ***to*** p **do**
    $z \leftarrow \text{prox}_{\gamma g}(\overline{x} + \Delta)$
    $x_i \leftarrow z - \gamma \nabla f_i(z)$
    $\Delta \leftarrow \Delta + \pi_i (x_i - x_i^{prev})$
    $x_i \leftarrow x_i^{prev}$
  end
  Send the adjustment Δ to the master
end

Convergence results

Take $p = 1$. Then, for $\gamma \in (0, 2/(\mu + L)]$,

$$\forall k \geq k_m, \quad \|x^k - x^\star\|^2 \leq \left(1 - \frac{2\gamma\mu L}{\mu + L}\right)^m \|x^0 - x^\star\|^2$$

where $x^\star$ is the unique minimizer of $\min_x \sum_{i=1}^M \pi_i f_i(x) + g(x)$.

- Usual fixed stepsize, delay-independent
- Linear rate along the *epoch* sequence; delay-tolerant

Take p changing across time and workers. Then, for $\gamma \in (0, 2/(\mu + L)]$,

$$\forall k \geq k_m, \quad \left\|x^k - x^\star\right\|^2 \leqslant \left(1 - \frac{2\gamma\mu L}{\mu + L}\right)^m \Pi_{\ell=1}^m \alpha_\ell \left\|x^0 - x^\star\right\|^2,$$

with epoch improvement

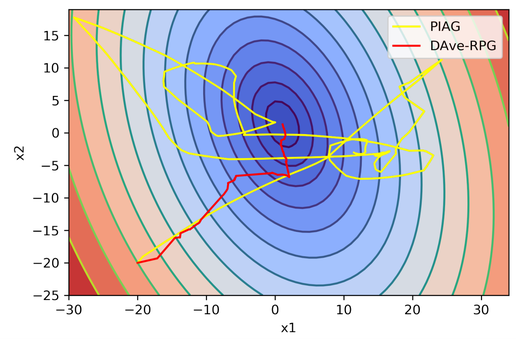
$$\alpha_\ell = \max_{i,k\in[k_\ell, k_{\ell+1})} 1 - \gamma\mu \sum_{q=1}^{p_i^k - 1} (1 - \gamma\mu)^{q-1} \min_i \pi_i^q.$$

- Workers compute more without coordinating
- Compromise Communication/Computation

Comparison with Gradient Combination

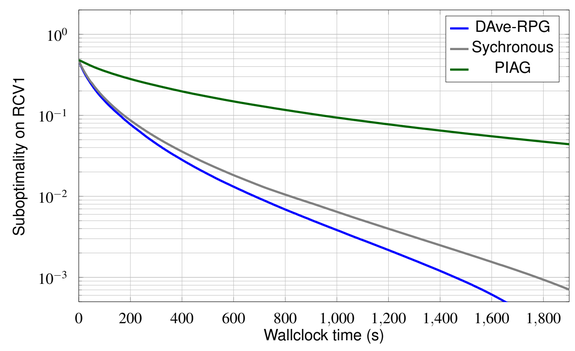
	DAve-RPG ($p = 1$)	PIAG
Combining:	iterates	gradients
Update x^k:	$\text{prox}_{\gamma g}\left(\sum_{i=1}^M \pi_i x^{k-D_i^k} - \gamma \sum_{i=1}^M \pi_i \nabla f_i(x^{k-D_i^k})\right)$	$\text{prox}_{\gamma g}\left(x^{k-1} - \gamma \sum_{i=1}^M \pi_i \nabla f_i(x^{k-D_i^k})\right)$

- Combining *iterates* is more stable than combining *gradients*
- Example: 2D quadratic functions on 5 worker but one worker 10x slower
  Stepsize γ of PIAG is 10x smaller due to delays while the one for DAve-PG stays =



Numerical results

- Logistic regression w/ elastic net $\frac{1}{m}\sum_{j=1}^m \log(1+\exp(-y_j z_j^T x)) + \lambda_1\|x\|_1 + \frac{\lambda_2}{2}\|x\|_2^2$
- 100 machines (1 CPU, 1 GB) in a cluster 10% of the data on machine 1, even on the rest



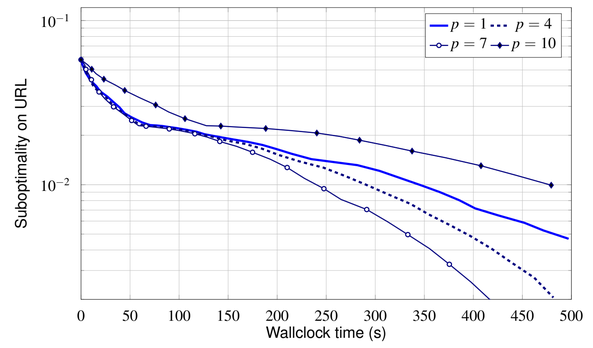


Illustration of delays on
10 evenly loaded machines

