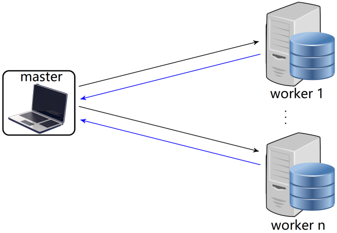
Delay-adaptive Step-sizes for Asynchronous Learning

Xuyang Wu[1], Sindri Magnusson[2], Hamid Reza Feyzmahdavian[3], Mikael Johansson[1]

[1]KTH Royal Institute of Technology
[2]Stockholm University
[3]ABB Corporate Research

ICML 2022

Synchronous and Asynchronous in Distributed Learning



Large sample number/model dimension $\rightarrow$ use of multiple processors

synchronous: all finish comp & comm, next iteration.

- GD: $x_{k+1} = x_k - \frac{\gamma}{n}\sum_{i=1}^{n} \nabla f^i(x_k)$.

inefficient, bottleneck: slowest worker.

asynchronous: some finish comp & comm, next iteration, cause delay.

- IAG:
  $x_{k+1} = x_k - \frac{\gamma}{n}\sum_{i=1}^{n} \nabla f^i(x_{k-\tau_k^i})$.

efficient, do not wait slowest.

Literature Review, Issues, and Idea

- asynchronous, non-diminishing step-size:
  - rely on an upper bound τ of all delays.
  - Issues: τ usually unknown (hard to implement) or large (small step-size, slow convergence)
- **Idea**: step-sizes should rely on actual delays. Poses two questions:
  1. can we measure actual delay? (yes, measured by difference of iteration index)
  2. large gap between delay bound and actual delay? (yes)

Gap between delay bound and actual delay

τ_k: maximal information delay at iteration k.

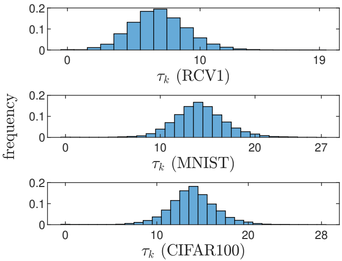


Figure: Real-world delay distribution (8 workers)

most delays are much smaller than maximal delay, good to use actual delay

Main part: delay-adaptive step-sizes for two asynchronous methods

Problem and Algorithm

$$\min_x \underbrace{f(x)}_{\textit{smooth loss}} + \underbrace{R(x)}_{\textit{convex regularizer}}$$

- large sample number: $f(x) := \frac{1}{n}\sum_{i=1}^{n} f^i(x)$
  - **PIAG:** $x_{k+1} = \mathrm{prox}_{\gamma_k R}(x_k - \frac{\gamma_k}{n}\sum_{i=1}^{n} \nabla f_i(x_{k-\tau_k^i}))$.

    $\mathrm{prox}_R(x) = \arg\ \min_y R(y) + \frac{1}{2}\|y - x\|^2$
- large variable dimension: $x = (x^1, \ldots, x^m)$, $R(x) = \sum_{i=1}^{m} R^i(x^i)$.
  - **Async-BCD:** $x_{k+1}^{i_k} = \mathrm{prox}_{\gamma_k R^{i_k}}(x_k^{i_k} - \gamma_k \nabla_{i_k} f(x_{k-\tau_k}))$

    number of blocks and workers can be different.

Convergence Analysis

$$\text{step-size principal:} \quad \gamma_k \le \max(0, \gamma' - \sum_{t=k-\tau_k}^{k-1} \gamma_t). \tag{1}$$

- **PIAG**: $O(\frac{1}{\sum_{t=0}^{k-1} \gamma_t})$, proximal-PL: $O(e^{-\sum_{t=0}^{k-1} \gamma_t})$.
- **Async-BCD:** $O(\frac{1}{\sum_{t=0}^{k-1} \gamma_t})$.

larger step-size integral $\sum_{t=0}^{k-1} \gamma_t$ $\rightarrow$ faster convergence

Adaptive 1: for $\alpha \in (0, 1]$,

$$\gamma_k = \alpha \max\{\gamma' - \sum_{t=k-\tau_k}^{k-1} \gamma_t, 0\}$$

Adaptive 2:

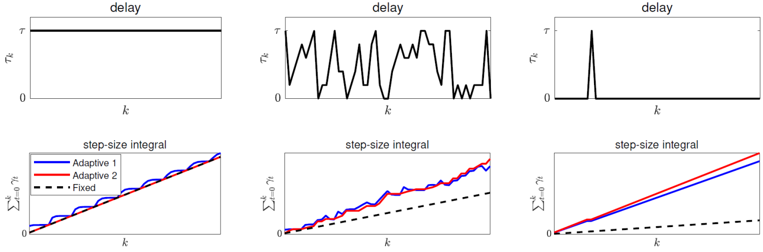
$$\gamma_k = \begin{cases} \frac{\gamma'}{\tau_k+1}, & \frac{\gamma'}{\tau_k+1} \le \gamma' - \sum_{t=k-\tau_k}^{k-1} \gamma_t, \\ 0, & \text{otherwise.} \end{cases}$$

satisfy (1), easy to implement, bounded delay $\rightarrow$ sublinear and linear

Comparison with Fixed Step-size

bounded delay ($\tau_k \leq \tau$), sota fixed: $\frac{\gamma'}{\tau+1}$.

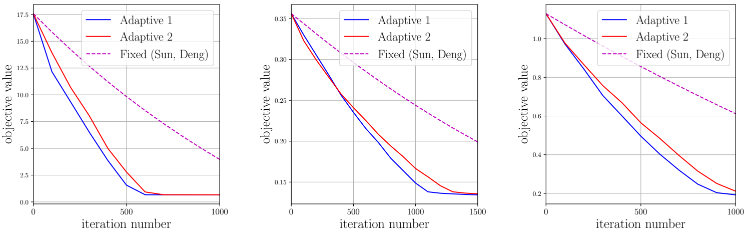
- **worst case**:
  Adaptive 1: $\sum_{t=0}^{k-1} \gamma_t \geq k \cdot \frac{\alpha\gamma'}{\tau+1}$ Adaptive 2: $\sum_{t=0}^{k-1} \gamma_t \geq k \cdot \frac{\tau\gamma'}{(\tau+1)^2}$
- **on delay models**:



(a) constant delay (b) random delay (c) burst delay

Experiment: Logistic Regression

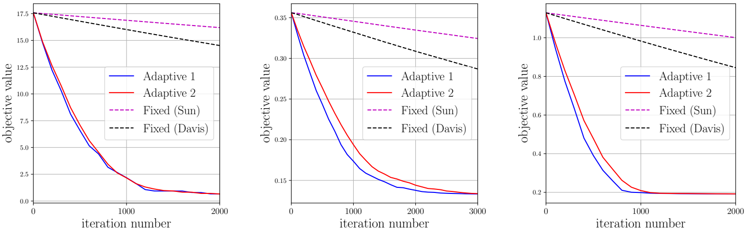
- problem: $\min \frac{1}{N} \sum_{i=1}^{N} \left(\log(1 + e^{-b_i(a_i^T x)}) + \frac{\lambda_2}{2} \|x\|^2 \right) + \lambda_1 \|x\|_1$
- for both algorithms, 8 workers, 10-core machine, 3 datasets, MPI4py.



(a) RCV1 (b) MNIST (c) CIFAR100

Figure: Convergence of **PIAG**

Experiment: Logistic Regression



(a) RCV1 (b) MNIST (c) CIFAR100

Figure: Convergence of **Async-BCD**

Conclusion

- delay-adaptive step-size
  - is implementable (delay-tracking is easy)
  - can significantly accelerate algorithms (validated by theory and experiment).
- the idea of delay-adaptive parameter selection is general, applicable to other asynchronous methods