Breaking the Nonsmooth Barrier: A Scalable Parallel Method for Composite Optimization

Fabian Pedregosa[†,‡], Rémi Leblond[†], Simon Lacoste–Julien[*]

[†]INRIA and École Normale Supérieure, Paris, France. [‡]Currently at UC Berkeley [*]MILA and DIRO, Université de Montréal, Canada



Summary

Optimization methods need to be adapted to the **parallel** setting to leverage modern computer architectures.

Highly efficient variants of stochastic gradient descent have been recently proposed, such as Hogwild [1], Kromagnon [2], ASAGA [3].

They assume that the objective function is smooth, so are inapplicable to problems such as Lasso, optimization with convex constraints, etc.

Main contributions:

1. **Sparse Proximal SAGA**, a sparse variant of the linearly-convergent proximal SAGA algorithm.
2. **ProxASAGA**, the first parallel asynchronous variance-reduced method that supports *nonsmooth* composite objective functions.

Problem Setting

Objective: develop parallel asynchronous method for problems of the form

$$\underset{\boldsymbol{x}\in\mathbb{R}^p}{\text{minimize}} \; \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{x}) + h(\boldsymbol{x}) \; ,$$

- f_i is differentiable with L-Lipschitz gradient.
- h is block-separable ($h(x) = \sum_B h_B([x]_B)$) and "simple" in the sense that we have access to

$$\mathbf{prox}_{\gamma h} \stackrel{\text{def}}{=} \mathbf{arg\,min}_{\boldsymbol{x}}\, \gamma h(\boldsymbol{x}) + \tfrac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|^2 \; .$$

$\Longrightarrow$ includes Lasso, group Lasso or box constraints.

Variance-reduced stochastic gradient methods are natural candidates due to their state of the art performance on these problems and recent asynchronous variants.

The **SAGA algorithm** [4] maintains current iterate $\boldsymbol{x} \in \mathbb{R}^p$ and historical gradients $\boldsymbol{\alpha} \in \mathbb{R}^{n\times p}$. At each iteration, sample $i \in \{1, \ldots, n\}$ and compute $(\boldsymbol{x}^+, \boldsymbol{\alpha}^+)$ as

$$\boldsymbol{x}^+ = \mathbf{prox}_{\gamma h}(\boldsymbol{x} - \gamma(\nabla f_i(\boldsymbol{x}) - \boldsymbol{\alpha}_i + \overline{\boldsymbol{\alpha}})) \, ; \;\; \boldsymbol{\alpha}_i^+ = \nabla f_i(\boldsymbol{x}) \; .$$

Difficulty of a Composite Extension

- Existing methods exhibit best performance when updates are sparse.
- Even in the presence of sparse gradients, the SAGA update is not sparse due to the presence of $\overline{\boldsymbol{\alpha}}$ and $\mathbf{prox}$.
- Existing convergence proofs bound noise from asynchrony using the Lipschitz constant of the gradient. This property does not extend to composite case.

A New Sequential Algorithm: Sparse Proximal SAGA

The algorithm relies on the following quantities

- Extended support T_i: set of blocks that intersect with ∇f_i.
$$T_i \stackrel{\text{def}}{=} \{B : \text{supp}(\nabla f_i) \cap B \neq \varnothing,\; B \in \mathcal{B}\}$$
- For each block $B \in \mathcal{B}$, $d_B \stackrel{\text{def}}{=} n/n_B$, where $n_B := \sum_i \mathbb{1}\{B \in T_i\}$ is the number of T_i that contain B.
- $\boldsymbol{D}_i$ is a diagonal matrix defined block-wise as $[\boldsymbol{D}_i]_{B,B} \stackrel{\text{def}}{=} d_B \mathbb{1}\{B \in T_i\} \boldsymbol{I}_{|B|}$.
- φ_i is a block-wise reweighting of h: $\varphi_i \stackrel{\text{def}}{=} \sum_{B\in T_i} d_B h_B(\boldsymbol{x})$

Justification. The following properties are verified

$\varphi_i(\boldsymbol{x})$ is zero outside T_i	$\boldsymbol{D}_i\boldsymbol{x}$ is zero outside T_i	(sparsity)
$\mathbb{E}_i\,\varphi_i = h$	$\mathbb{E}_i\,\boldsymbol{D}_i = \boldsymbol{I}$	(unbiasedness)

Algorithm. As SAGA, it maintains current iterate $\boldsymbol{x} \in \mathbb{R}^p$ and table of historical gradients $\boldsymbol{\alpha} \in \mathbb{R}^{n\times p}$. At each iteration, it samples an index $i \in \{1, \ldots, n\}$ and computes next iterate $(\boldsymbol{x}^+, \boldsymbol{\alpha}^+)$ as

$$\boldsymbol{v}_i = \nabla f_i(\boldsymbol{x}) - \boldsymbol{\alpha}_i + \boldsymbol{D}_i\overline{\boldsymbol{\alpha}}$$
$$\boldsymbol{x}^+ = \mathbf{prox}_{\gamma\varphi_i}(\boldsymbol{x} - \gamma\boldsymbol{v}_i) \, ; \;\; \boldsymbol{\alpha}_i^+ = \nabla f_i(\boldsymbol{x})$$

Features

- Per Iteration cost in $\mathcal{O}(|T_i|)$.
- Easy to implement (compared to the lagged update approach [5]).
- Amenable to parallelization.

Convergence Analysis

For step size $\gamma = \frac{1}{5L}$ and f μ-strongly convex ($\mu > 0$), Sparse Proximal SAGA converges geometrically in expectation. At iteration t we have

$$\mathbb{E}\|\boldsymbol{x}_t - \boldsymbol{x}^*\|^2 \leq (1 - \tfrac{1}{5}\min\{\tfrac{1}{n}, \tfrac{1}{\kappa}\})^t C_0 \; ,$$

with $C_0 = \|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2 + \frac{1}{5L^2}\sum_{i=1}^n \|\boldsymbol{\alpha}_i^0 - \nabla f_i(\boldsymbol{x}^*)\|^2$ and $\kappa = \frac{L}{\mu}$ (condition number).

Implications

- Same convergence rate than SAGA with cheaper updates.
  - In the "big data regime" ($n \geq \kappa$): rate in $\mathcal{O}(1/n)$.
  - In the "ill-conditioned regime" ($n \leq \kappa$): rate in $\mathcal{O}(1/\kappa)$.
- Adaptivity to strong convexity, i.e., no need to know strong convexity parameter to obtain linear convergence.

A New Parallel Algorithm: Proximal Asynchronous SAGA (ProxASAGA)

Proximal Asynchronous SAGA (ProxASAGA) runs Sparse Proximal SAGA asynchronously and without locks and updates $\boldsymbol{x}$, $\boldsymbol{\alpha}$ and $\overline{\boldsymbol{\alpha}}$ in shared memory.

All read/write operations to shared memory are *inconsistent*, i.e., no vector-level locks while reading/writing.

keep doing in parallel
    Sample i uniformly in $\{1, ..., n\}$
    $[\hat{\boldsymbol{x}}]_{T_i}$ = inconsistent read of $\boldsymbol{x}$ on T_i
    $\hat{\boldsymbol{\alpha}}_i$ = inconsistent read of $\boldsymbol{\alpha}_i$
    $[\overline{\boldsymbol{\alpha}}]_{T_i}$ = inconsistent read of $\overline{\boldsymbol{\alpha}}$ on T_i
    $[\delta\boldsymbol{\alpha}]_{S_i} = [\nabla f_i(\hat{\boldsymbol{x}})]_{S_i} - [\hat{\boldsymbol{\alpha}}_i]_{S_i}$
    $[\hat{\boldsymbol{v}}]_{T_i} = [\delta\boldsymbol{\alpha}]_{T_i} + [\boldsymbol{D}_i\overline{\boldsymbol{\alpha}}]_{T_i}$
    $[\delta\boldsymbol{x}]_{T_i} = [\mathbf{prox}_{\gamma\varphi_i}(\hat{\boldsymbol{x}} - \gamma\hat{\boldsymbol{v}})]_{T_i} - [\hat{\boldsymbol{x}}]_{T_i}$
    for B **in** T_i **do**
        for b **in** B **do**
            $[\boldsymbol{x}]_b \leftarrow [\boldsymbol{x}]_b + [\delta\boldsymbol{x}]_b$ ▷ atomic
            if $b \in \text{supp}(\nabla f_i)$ **then**
                $[\overline{\boldsymbol{\alpha}}]_b \leftarrow [\overline{\boldsymbol{\alpha}}]_b + 1/n[\delta\boldsymbol{\alpha}]_b$ ▷ atomic
            end if
        end for
    end for
    $\boldsymbol{\alpha}_i \leftarrow \nabla f_i(\hat{\boldsymbol{x}})$ (scalar update) ▷ atomic
end parallel loop

Perturbed Iterate Framework

Problem: Analysis of asynchronous parallel algorithms is *hard*.

Solution: Cast them as sequential algorithms working on *perturbed* inputs. Distinguish:

- $\hat{\boldsymbol{x}}_t$: inconsistent vector. Counter t is incremented when a core *finishes reading* the parameters (after read labeling [3]).
- $\boldsymbol{x}_t$: the *virtual* iterate defined by $\boldsymbol{x}_{t+1} \stackrel{\text{def}}{=} \boldsymbol{x}_t - \gamma\boldsymbol{g}_t$ with $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{v}, i) = \frac{1}{\gamma}(\hat{\boldsymbol{x}}_t - \mathbf{prox}_{\gamma\varphi_i}(\hat{\boldsymbol{x}}_t - \gamma\hat{\boldsymbol{v}}_i))$.

Interpret $\hat{\boldsymbol{x}}_t$ as a noisy version of $\boldsymbol{x}_t$ due to asynchrony. Generalization of perturbed iterate framework [2, 3] to composite objectives.

Analysis preliminaries

Definition (measure of sparsity). Let $\Delta := \max_{B\in\mathcal{B}} |\{i : T_i \ni B\}|/n$. This is the normalized maximum number of times that a block appears in the extended support. We always have $1/n \leq \Delta \leq 1$.

Definition (delay bound). τ is a uniform bound on the *maximum delay* between two iterations processed concurrently.

Convergence guarantee of ProxASAGA

Suppose $\tau \leq \frac{1}{10\sqrt{\Delta}}$. Then:

- If $\kappa \geq n$, then with step size $\gamma = 1/36L$, ProxASAGA converges geometrically with rate factor $\Omega(\frac{1}{\kappa})$.
- If $\kappa < n$, then using the step size $\gamma = 1/36n\mu$, ProxASAGA converges geometrically with rate factor $\Omega(\frac{1}{n})$.

In both cases, the convergence rate is the same as Sparse Proximal SAGA $\Longrightarrow$ ProxASAGA is **linearly faster** up to constant factor. In both cases the **step size does not depend on** τ.

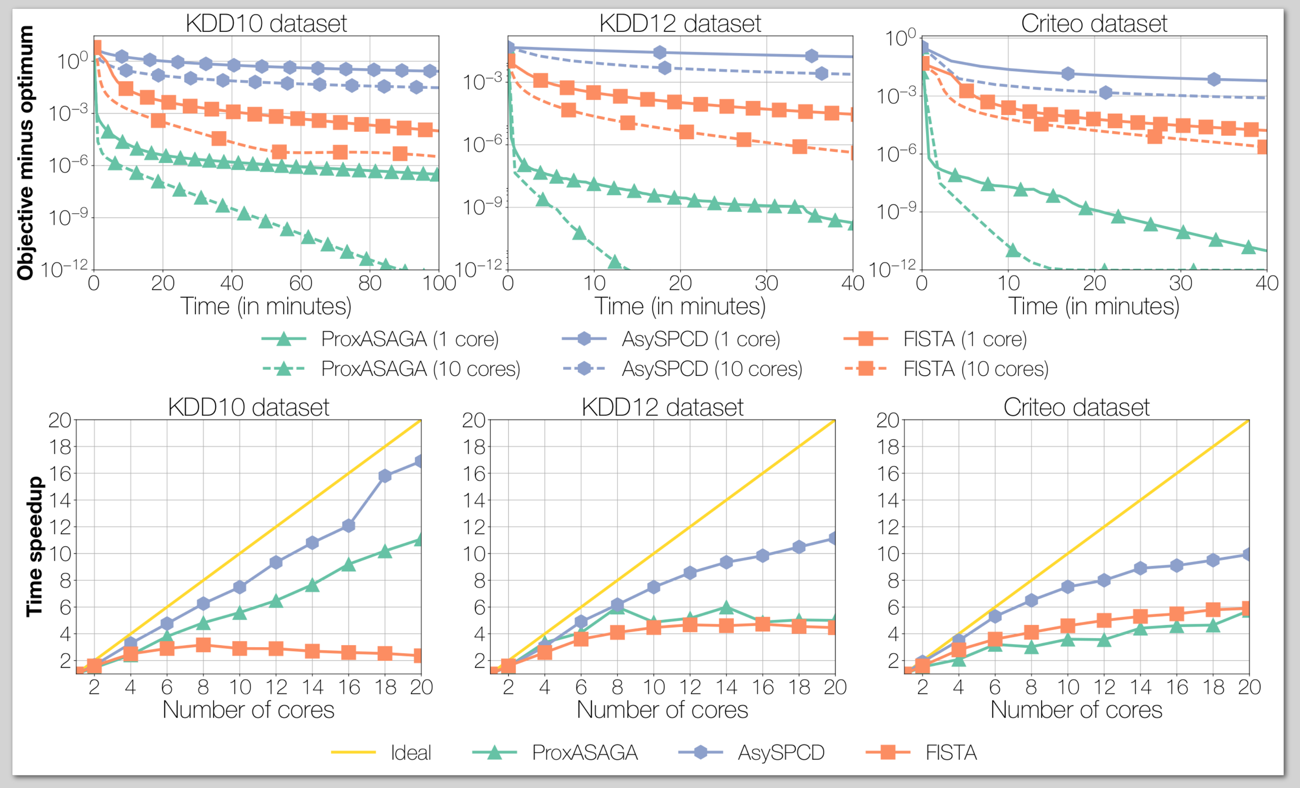
If $\tau \leq 6\kappa$, a universal step size of $\Theta(1/L)$ achieves a similar rate than Sparse Proximal SAGA, making it adaptive to local strong convexity (knowledge of κ not required).

Experimental results

Comparison on 3 large-scale datasets on an elastic-net regularized logistic regression model:

$$\underset{x}{\text{minimize}} \; \frac{1}{n}\sum_{i=1}^n \log\left(1 + \exp(-b_i\boldsymbol{a}_i^\intercal\boldsymbol{x})\right) + \frac{\lambda_1}{2}\|\boldsymbol{x}\|_2^2 + \lambda_2\|\boldsymbol{x}\|_1 \quad ,$$

Dataset	n	p	density	L	Δ
KDD 2010	19,264,097	1,163,024	10^{-6}	28.12	0.15
KDD 2012	149,639,105	54,686,452	2×10^{-7}	1.25	0.85
Criteo	45,840,617	1,000,000	4×10^{-5}	1.25	0.89



Highlights: ProxASAGA significantly outperforms existing methods, significant speedup (6x to 12x) over the sequential version.

References

1. Niu, F., Recht, B., Re, C. & Wright, S. *Hogwild: A lock-free approach to parallelizing stochastic gradient descent.* in *NIPS* (2011).
2. Mania, H. *et al.* Perturbed iterate analysis for asynchronous stochastic optimization. *SIAM Journal on Optimization* (2017).
3. Leblond, R., Pedregosa, F. & Lacoste-Julien, S. ASAGA: asynchronous parallel SAGA. *AISTATS* (2017).
4. Defazio, A. *et al. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives.* in *NIPS* (2014).
5. Schmidt, M., Le Roux, N. & Bach, F. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming* (2016).