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. [⋆] 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 constraints, etc.

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 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 ($h(x) = \sum_B h_B([\boldsymbol{x}]_B)$) and "simple" in the sense that we have access to $\mathbf{prox}_h \overset{\text{def}}{=} \mathbf{arg\,min}_{\boldsymbol{x}} h(\boldsymbol{x}) + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{z}\|^2$.

$\implies$ includes Lasso, group Lasso or ERM with box constraints.

Variance-reduced stochastic gradient methods are natural candidates: state of the art performance and recent asynchronous variants . The **SAGA** algorithm [4] has an iteration of the form

$$\begin{aligned}&\text{choose random } i \in \{1, \dots, n\}\\ &\boldsymbol{x}^+ = \mathbf{prox}_{\gamma g}(\boldsymbol{x} - \gamma(\nabla f_i(\boldsymbol{x}) - \boldsymbol{\alpha}_i + \overline{\boldsymbol{\alpha}}))\ ,\ \boldsymbol{\alpha}_i^+ = \nabla f_i(\boldsymbol{x})\ .\end{aligned} \qquad \text{(SAGA)}$$

Difficulty of a composite extension

- Existing parallel asynchronous variants of SGD crucially rely on sparse updates.
- 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 crucially rely on the gradient smoothness

A new algorithm: Sparse Proximal SAGA

The algorithm relies on the definitions

- Extended support
- $\boldsymbol{D}_i$
- φ

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

Linear convergence rate

Theorem. Let $\gamma = \frac{a}{5L}$ for any $a \leq 1$ and f be μ-strongly convex ($\mu > 0$). Then Sparse Proximal SAGA converges geometrically in expectation with a rate factor of at least $\rho = \frac{1}{5}\min\{\frac{1}{n}, a\frac{1}{\kappa}\}$. That is, for $\boldsymbol{x}_t$ obtained after t updates, we have the following bound:

$$\mathbb{E}\|\boldsymbol{x}_t - \boldsymbol{x}^*\|^2 \leq (1-\rho)^t C_0\ , \quad \text{with } C_0 := \|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2 + \frac{1}{5L^2}\sum_{i=1}^n \|\boldsymbol{\alpha}_i^0 - \nabla f_i(\boldsymbol{x}$$

Proximal Asynchronous SAGA (ProxASAGA)

Algorithm 1 ProxASAGA (analyzed)

Initialize shared variables $\boldsymbol{x}$ and $(\boldsymbol{\alpha}_i)_{i=1}^n$
loop
  $\hat{\boldsymbol{x}} =$ inconsistent read of $\boldsymbol{x}$
  $\hat{\boldsymbol{\alpha}} =$ inconsistent read of $\boldsymbol{\alpha}$
  Sample i uniformly in $\{1, ..., n\}$
  $S_i :=$ support of ∇f_i
  $T_i :=$ extended support of ∇f_i in $\mathcal{B}$
  $[\overline{\boldsymbol{\alpha}}]_{T_i} = {}^1\!/_n \sum_{j=1}^n [\hat{\boldsymbol{\alpha}}_j]_{T_i}$
  $[\delta\boldsymbol{\alpha}]_{S_i} = [\nabla f_i(\hat{\boldsymbol{x}})]_{S_i} - [\hat{\boldsymbol{\alpha}}_i]_{S_i}$
  $[\hat{\boldsymbol{v}}]_{T_i} = [\delta\boldsymbol{\alpha}]_{T_i} + [\boldsymbol{\Delta}_i\overline{\boldsymbol{\alpha}}]_{T_i}$
  $[\delta\boldsymbol{x}]_{T_i} = [\mathbf{prox}_{\gamma\varphi_i}(\hat{\boldsymbol{x}} - \gamma\hat{\boldsymbol{v}})]_{T_i} - [\hat{\boldsymbol{x}}]_{T_i}$
  for B **in** T_i **do**
    for $b \in B$ **do**
      $[\boldsymbol{x}]_b \leftarrow [\boldsymbol{x}]_b + [\delta\boldsymbol{x}]_b$ ▷ atomic
      if $b \in S_i$ **then**
        $[\boldsymbol{\alpha}_i]_b \leftarrow [\nabla f_i(\hat{\boldsymbol{x}})]_b$
      end if
    end for
  end for
  // ('←' denotes shared memory update.)
end loop

Algorithm 2 ProxASAGA (implemented)

Initialize shared variables $\boldsymbol{x}$, $(\boldsymbol{\alpha}_i)_{i=1}^n$, $\overline{\boldsymbol{\alpha}}$
loop
  Sample i uniformly in $\{1, ..., n\}$
  $S_i :=$ support of ∇f_i
  $T_i :=$ extended support of ∇f_i in $\mathcal{B}$
  $[\hat{\boldsymbol{x}}]_{T_i} =$ 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
  $[\delta\boldsymbol{\alpha}]_{S_i} = [\nabla f_i(\hat{\boldsymbol{x}})]_{S_i} - [\hat{\boldsymbol{\alpha}}_i]_{S_i}$
  $[\hat{\boldsymbol{v}}]_{T_i} = [\delta\boldsymbol{\alpha}]_{T_i} + [\boldsymbol{\Delta}_i\overline{\boldsymbol{\alpha}}]_{T_i}$
  $[\delta\boldsymbol{x}]_{T_i} = [\mathbf{prox}_{\gamma\varphi_i}(\hat{\boldsymbol{x}} - \gamma\hat{\boldsymbol{v}})]_{T_i} - [\hat{\boldsymbol{x}}]_{T_i}$
  for B **in** T_i **do**
    for b **in** B **do**
      $[\boldsymbol{x}]_b \leftarrow [\boldsymbol{x}]_b + [\delta\boldsymbol{x}]_b$ ▷ atomic
      if $b \in S_i$ **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 loop

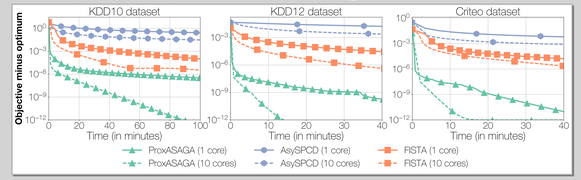
Experimental results

Comparison on 3 large-scale datasets

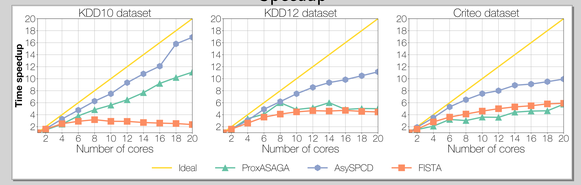
Large-scale logistic regression

Dataset	n	p	$-$	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

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



Speedup



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., Bach, F. & Lacoste-Julien, S. *SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives.* in *NIPS* (2014).