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

Fabian Pedregosa
UC Berkeley
Berkey, California
f@bianp.net

Rémi Leblond
INRIA and Ecole Normale Superieure
Paris, France
remi.leboond@inria.fr

Simon Lacoste–Julien
MILA and DIRO, Montreal University
Montreal, Canada
slacoste@iro.umontreal.ca

ABSTRACT

Due to their simplicity and excellent performance, parallel asynchronous variants of stochastic gradient descent have become popular methods to solve a wide range of large-scale optimization problems on multi-core architectures. Yet, despite their practical success, support for nonsmooth objectives is still lacking, making them unsuitable for many problems of interest in machine learning, such as the Lasso, group Lasso or empirical risk minimization with convex constraints. In this work, we propose and analyze ProxAsaga, a fully asynchronous sparse method inspired by Saga, a variance reduced incremental gradient algorithm. The proposed method is easy to implement and significantly outperforms the state of the art on several nonsmooth, large-scale problems. We prove that our method achieves a theoretical linear speedup with respect to the sequential version under assumptions on the sparsity of gradients and block-separability of the proximal term. Empirical benchmarks on a multi-core architecture illustrate practical speedups of up to 12x on a 20-core machine.

KEYWORDS

parallel optimization, machine learning, stochastic optimization

ACM Reference Format:
Fabian Pedregosa, Rémi Leblond, and Simon Lacoste–Julien. 2018. Breaking the Nonsmooth Barrier: A Scalable Parallel Method for Composite Optimization. In *Proceedings of SysML conference (SYSML'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.475/123_4

1 SETTING

The class of problems that we consider are problems of the form:

$$\arg\min_{\boldsymbol{x}\in\mathbb{R}^p} f(\boldsymbol{x}) + h(\boldsymbol{x}), \quad \text{with } f(\boldsymbol{x}) := \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{x}),$$

where each f_i is convex with L-Lipschitz gradient, the average function f is μ-strongly convex and h is convex but potentially nonsmooth. We further assume that h is "simple" in the sense that we have access to its proximal operator and that it is block-separable, that is, it can be decompose block coordinate-wise as $h(\boldsymbol{x}) = \sum_{B\in\mathcal{B}} h_B([\boldsymbol{x}]_B)$, where h_B only depends on coordinates in block B and $\mathcal{B}$ is a partition of the coefficients into subsets which will call *blocks*.

This template models a broad range of problems arising in machine learning and signal processing: the finite-sum structure of f includes the least squares or logistic loss functions; the proximal term h includes penalties such as the ℓ_1 or group lasso penalty. Furthermore, this term can be extended-valued, thus allowing for convex constraints through the indicator function.

2 SPARSE PROXIMAL SAGA

Our first contribution is to develop a sparse variant of the Saga algorithm [1] in which the cost per iteration is only proportional to the nonzeros in the partial gradient. We denote by T_i the "extended support" of ∇f_i, which is defined as the smallest set of blocks in $\mathcal{B}$ that contain the support of ∇f_i. Let $d_B := n/n_B$, where $n_B := \sum_i \mathbb{1}\{B \in T_i\}$ is the number of times that $B \in T_i$. For simplicity we assume $n_B > 0$, as otherwise the problem can be reformulated without block B. We also define $\varphi_i(\boldsymbol{x}) := \sum_{B\in T_i} d_B h_B(\boldsymbol{x})$.

Following Leblond et al. [2], we will also replace the dense gradient estimate $\boldsymbol{u}_i$ by the sparse estimate $\boldsymbol{v}_i := \nabla f_i(\boldsymbol{x}) - \boldsymbol{\alpha}_i + \boldsymbol{D}_i\overline{\boldsymbol{\alpha}}$, where $\boldsymbol{D}_i$ is the diagonal matrix defined block-wise as $[\![\boldsymbol{D}_i]\!]_{B,B} = d_B\mathbb{1}\{B \in T_i\}\boldsymbol{I}_{|B|}$. We now describe the Sparse Proximal Saga algorithm. As the original Saga algorithm, it maintains two moving quantities: the current iterate $\boldsymbol{x} \in \mathbb{R}^p$ and a table of historical gradients $(\boldsymbol{\alpha}_i)_{i=1}^n$, $\boldsymbol{\alpha}_i \in \mathbb{R}^p$. At each iteration, the algorithm samples an index $i \in \{1, \dots, n\}$ and computes the next iterate $(\boldsymbol{x}^+, \boldsymbol{\alpha}^+)$ as:

$$\begin{aligned} \boldsymbol{v}_i &= \nabla f_i(\boldsymbol{x}) - \boldsymbol{\alpha}_i + \boldsymbol{D}_i\overline{\boldsymbol{\alpha}}\,; \\ \boldsymbol{x}^+ &= \mathbf{prox}_{\gamma\varphi_i}\big(\boldsymbol{x} - \gamma\boldsymbol{v}_i\big)\,;\ \boldsymbol{\alpha}_i^+ = \nabla f_i(\boldsymbol{x})\,, \end{aligned} \tag{SPS}$$

where in a practical implementation the vector $\overline{\boldsymbol{\alpha}}$ is updated incrementally at each iteration.

3 PROXIMAL ASYNCHRONOUS SAGA

Our algorithm maintains two quantities in shared memory to which the different processors will have access: the vector of coefficients $\boldsymbol{x} \in \mathbb{R}^p$ and a table of memory terms $\boldsymbol{\alpha}$, which is of size $n \times p$ in the general case but can be compressed to size n for generalized linear models such as logistic regression. In our algorithm, each processor samples a data index i and performs asynchronous the update described in Algorithm 1.

4 ANALYSIS AND EXPERIMENTS

We prove that our method achieves a theoretical linear speedup with respect to the sequential version under assumptions on the sparsity of gradients and block-separability of the proximal term.

SYSML'17, February 2017, Stanford, California USA

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

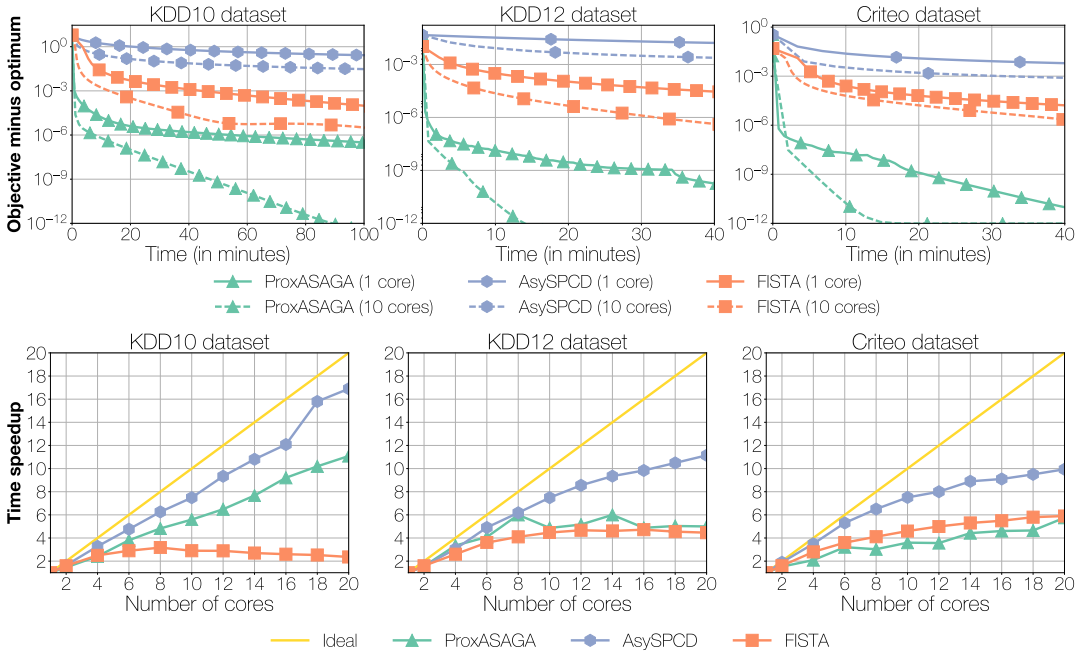

Figure 1: Convergence for asynchronous stochastic methods for $\ell_1 + \ell_2$-regularized logistic regression. Top: Suboptimality as a function of time for different asynchronous methods using 1 and 10 cores. Bottom: Running time speedup as function of the number of cores. ProxAsaga achieves significant speedups over its sequential version while being orders of magnitude faster than competing methods. AsySPCD achieves the highest speedups but it also the slowest overall method.

Algorithm 1 ProxAsaga

Initialize shared variables $\boldsymbol{x}$, $(\boldsymbol{\alpha}_i)_{i=1}^n$, $\overline{\boldsymbol{\alpha}}$
loop
  Sample i uniformly in $\{1, \dots, n\}$
  $S_i :=$ 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
  $[\delta\boldsymbol{\alpha}]_{S_i} = [\nabla f_i(\hat{\boldsymbol{x}})]_{S_i} - [\hat{\boldsymbol{\alpha}}_i]_{S_i}$
  $[\hat{\boldsymbol{v}}]_{T_i} = [\delta\boldsymbol{\alpha}]_{T_i} + [\boldsymbol{D}_i\overline{\boldsymbol{\alpha}}]_{T_i}$
  $[\delta\boldsymbol{x}]_{T_i} = [\mathbf{prox}_{\varphi_i}(\hat{\boldsymbol{x}} - \gamma\hat{\boldsymbol{v}})]_{T_i} - [\hat{\boldsymbol{x}}]_{T_i}$
  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
  $\boldsymbol{\alpha}_i \leftarrow \nabla f_i(\hat{\boldsymbol{x}})$ ▷ atomic
end loop

Our main results states that ProxAsaga obtains (under assumptions) a theoretical linear speedup with respect to its sequential version. Empirical benchmarks reported Figure 1 show that this method dramatically outperforms state of the art alternatives on large sparse datasets, while the empirical speedup analysis illustrates the practical gains as well as its limitations.

Definition 4.1. Let $\Delta := \max_{l=1..p} |\{i : l \in T_i\}|/n$. This is the normalized maximum number of data points that share a specific block in their extended support. For example, if a block is present in all T_i, then $\Delta = 1$. If no two T_i share the same block, then $\Delta = 1/n$.

Corollary 4.2 (Speedup). *Suppose* $\tau \leq \frac{1}{10\sqrt{\Delta}}$. *If* $\kappa \geq n$, *then using the 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 its sequential counterpart. Thus* ProxAsaga *is linearly faster than its sequential counterpart up to a constant factor. Note that in both cases* the step size does not depend on τ.

Furthermore, if $\tau \leq 6\kappa$, *we can use a universal step size of* $\Theta(1/L)$ *to get a similar rate for* ProxAsaga *than Sparse Proximal* Saga, *thus making it adaptive to local strong convexity since the knowledge of* κ *is not required.*

These speedup rates are comparable with the best ones obtained in the smooth case, including Niu et al. [3], Reddi et al. [4], even though unlike these papers, we support inconsistent reads and nonsmooth objective functions. The one exception is Leblond et al. [2], where the authors prove that their algorithm, Asaga, can obtain a linear speedup even without sparsity in the well-conditioned regime. In contrast, ProxAsaga always requires some sparsity. Whether this property for smooth objective functions could be extended to the composite case remains an open problem.

REFERENCES

[1] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. 2014. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*.

[2] Rémi Leblond, Fabian Pedregosa, and Simon Lacoste-Julien. 2017. ASAGA: asynchronous parallel SAGA. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017)* (2017).

[3] Feng Niu, Benjamin Recht, Christopher Re, and Stephen Wright. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*.

[4] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alexander J Smola. 2015. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Advances in Neural Information Processing Systems*.