Asynchronous Distributed Learning with Sparse Communications and Identification

Dmitry Grishchenko, Franck Iutzeler, Jérôme Malick, Massih-Reza Amini

Asynchronous Distributed Learning with Sparse Communications and Identification

Dmitry Grishchenko* Franck Iutzeler* Jérôme Malick† Massih-Reza Amini*

December 10, 2018

Abstract

In this paper, we present an asynchronous optimization algorithm for distributed learning, that efficiently reduces the communications between a master and working machines by randomly sparsifying the local updates. This sparsification allows to lift the communication bottleneck often present in distributed learning setups where computations are performed by workers on local data while a master machine coordinates their updates to optimize a global loss.

We prove that despite its sparse asynchronous communications, our algorithm allows for a fixed stepsize and benefits from a linear convergence rate in the strongly convex case. Moreover, for ℓ_1-regularized problems, this algorithm identifies near-optimal sparsity patterns, so that all communications eventually become sparse. We furthermore leverage on this identification to improve our sparsification technique. We illustrate on real and synthetic data that this algorithm converges faster in terms of data exchanges.

1 Introduction

Distributed learning with costly communications. We consider a distributed learning set-up where n observations are split down over M machines, each machine i having a private subset $\mathcal{S}_i$ of the examples. Standard machine learning approaches usually consider that the entire training set is stored in one single machine or in a datacenter. In contrast, our aim is to learn collaboratively a shared prediction model without moving the training data, hence decoupling the ability to learn from the need to store the data in a centralized way.

Learning over scattered data leads to optimization problems with composite objective of the form

$$\min_{x\in\mathbb{R}^d} \quad \sum_{i=1}^{M} \pi_i f_i(x) + r(x), \tag{1}$$

with $\pi_i = n_i/n$ being the proportion of observations locally stored in machine i, and $f_i(x) = \frac{1}{n_i}\sum_{j\in\mathcal{S}_i} \ell_j(x)$ being the local empirical risk at machine i. This formulation corresponds to a setup *without shared memory* where each machine has access only to its local subset of the data. In this situation, it is natural to consider that the machines, also referred to as *slaves*, perform their computations separately and communicate with a *master* machine. Our general computation framework is the following (see e.g. [15, 10, 11, 22]): one machine gets the current model from the master, improves it by learning from its own data which produces a local model update; only this update is sent to the master, which computes a new shared model.

In large-scale machine learning, stochastic optimization algorithms are very popular, see e.g. the parallel stochastic algorithms of [13, 29]. Such algorithms are highly iterative and then require low-latency and high-throughput connections. In contrast, our distributed setting features machines having significantly higher-latency and lower-throughput connections, and only intermittently available for training. In fact it

*Université Grenoble Alpes
†CNRS, Laboratoire Jean Kuntzmann

is commonly admitted (see e.g. [19, 18]) that in a distributed computing environment, one must focus not only on the data accesses, but also on the size of communication, thus rehabilitating batch algorithms. In the context of this paper, communications are typically the practical bottleneck of the learning process, see e.g. [16, 33].

Contributions. In this paper, we aim at providing a distributed optimization algorithm with fewer communications.

We first propose an asynchronous distributed algorithm, featuring a sparsification of upward communications (slave-to-master). Our sparsification mechanism is based on uniform sampling selection of local update entries. This random technique maintains the linear convergence in the mean-squared error sense, in the strongly convex case. The resulting proximal gradient algorithm is adjustable to various levels of communication costs, machines computational powers, and data distribution evenness. An attractive property of the algorithm is the possibility to use a fixed learning rate that does not depend neither on communication delays nor on the number of machines (one can actually use the same fixed stepsize as for the vanilla proximal-gradient algorithm).

Secondly, in the case of ℓ_1-regularized problems, we prove that this distributed algorithm identifies some sparsity pattern in finite time with probability one, resulting in sparse downward communications (master-to-slave). Thus, all communications are eventually sparse. We furthermore leverage on this identification result to improve our sparsification technique: this approach can be seen as an automatic dimension reduction procedure, resulting in better performance with limited communications.

Outline. The paper is organized as follows. In Section 2, we position our work with respect to the state-of-the-art. In Section 3, we present the new asynchronous distributed algorithm with sparse up-communications, and we provide its exponential convergence rate in the strongly convex case. In Section 4, we prove that the algorithm identifies a near-optimal support for ℓ_1-regularized problems and discuss its corresponding algorithmic implications. Finally, we provide numerical illustrations in section 5 for typical ℓ_1-regularized problems.

2 Related literature

On sparsification. There has been a great interest in sparsity for machine learning both from statistical and algorithmic perspectives (see e.g. [31] and [12] and references therein). Sparsity can be induced by three main families of approaches:

- **Proximal methods.** They consist in using sparsity-inducing regularizations in learning models and solving the optimization problem with proximal-based algorithms. Typically, one uses ℓ_0 or ℓ_1 norms, the proximity operator of which are respectively hard or soft thresholding [32, 27].

- **Coordinate descent.** Block coordinate descent algorithms, recently knew a rebirth in the context of huge-scale learning [23, 28]; these methods can be interpretated as sparsified gradient descent. In the context of ℓ_1-regularized problems, they can be combined with screening techniques for efficiently solving high-dimensional problems, [20].

- **Random sparsification for distributed learning.** The idea of randomly selecting some entries to update has been applied in distributed algorithms. Random selection is used to sparsify local gradients in the synchronous algorithm of [33], to sparsify the variance-reducing term in the stochastic algorithm of [13] and [24], or to sparsify updates in fixed-point iterations [25]. In the context of federated learning, [10] mentions the idea of random sparsification, but without further study. Here we provide a thorough analysis of the idea, and we further study its interplay with the proximal approach and with identification properties of algorithms.

Our algorithm combines these three approaches: it features a proximal ℓ_1 step for the downward communications and a random coordinate updates for the upward communications. To the best of our knowledge, our algorithm is the first to feature such asynchronous two-way sparse communications.

On Identification. Identification of optimal structure has been studied for a long time in the context of constrained convex optimization (see e.g. [34]) and nonsmooth nonconvex optimization (e.g. [9]). Although many deterministic algorithms have been proved to have identification properties, the situation is less clear for random algorithms: in particular (proximal) stochastic gradient methods are known to be unable to identify substructure (see e.g. [14]), while it has been recently proved that (proximal) variance-reduced stochastic gradient methods do have such identification properties [26]. No identification results have yet been reported for asynchronous distributed optimization algorithms.

All known results (including also [14]) use moreover a strong non-degeneracy assumption to establish identification result. The only exception is [6] that present extended identification results without non-degeneracy for a large class of nonsmooth regularizers. Identification results for our distributed algorithm in the ℓ_1 regularization will be based on general identification results of [6].

Finally note that, though identification has been used to accelerate algorithms (by a better tuning of parameters, lower computation complexity of iterations, or high-order accelerations), it is the first time in this paper that it is used as an automatic dimension reduction and sparsification technique.

On Asynchronous Distributed Optimization. There is a vast literature on distributed optimization methods without shared memory, based a variety of algorithms such as the ADMM, proximal gradient, SAGA etc. [35, 19, 1, 25, 5]. However, these methods usually rely on restrictive assumptions on the computing system delays, which in turn impact the obtained convergence rates. Asynchronous coordinate descent methods able to handle unbounded delays were recently proposed [8, 30] but use decreasing stepsizes. In contrast, the recent works [22, 21] provide a delay-independent analysis technique that allows to get rid of assumptions on the computing system. We use these techniques in the convergence proofs of our algorithms. It enables us to show the convergence of our distributed randomized method with the same fixed stepsizes as in the vanilla proximal gradient algorithm.

3 Distributed algorithm with sparsification

In this section we present our distributed algorithm for solving (1) with sparse upward communications. Section 3.1 introduces the notation for asynchrony, the proximal gradient iteration, and the details of the algorithm. Section 3.2 studies its convergence under a standard assumption on the model (strong convexity) but no assumption on the computing system. (in particular, no assumption on delays or data distribution).

3.1 Asynchronous distributed proximal algorithm with sparsification

Notation for asynchronous updates. An asynchronous distributed setting allows the algorithm to carry on computation without waiting for slower machines: the machine performs computations based on outdated versions of the main variable, and the master has to gather the slaves inputs into a productive update. We formalize this framework with the following notation.

- *For the master.* We define the time k, as the number of updates the master receives from any of the slaves. Thus, at time k, the master receives some input from an agent, denoted by i^k, updates its global variables $\overline{x}^k$ and x^k, and sends back x^k to slave i^k.
- *For slave i.* At time k, we introduce d_i^k the time elapsed from the last update slave i to the master ($d_i^k = 0$ iff the master gets updates from slave i at time k, i.e. $i^k = i$). We also consider D_i^k the time of the penultimate update. This means that, at time k, the last two moments when i updates were $k - d_i^k$ and $k - D_i^k$.

Sparsification of local updates. In the proposed method, the master machine asynchronously gathers *sparsified* delayed gradient updates from slaves and sends them back the current point. More specifically, each slaves independently computes, using its local subset of the data, a gradient step for a selected subset of coordinates only. At iteration k, this randomly drawn subset of entries of the gradient to be computed by agent i^k is called *mask* and is denoted by $\mathbf{S}^k$ (in bold, to single it out as it is the *only* random variable in

the algorithm). The master machines keeps track of the weighted average of the most recent slave outputs, computes the proximity operator of the regularizer at this average point, and sends this result back to the updating slave i^k. Thus, the iteration of our base algorithm writes as follows with $x_{[j]}$ denoting the j-th coordinate of $x \in \mathbb{R}^d$.

$$x^k_{i[j]} = \begin{cases} \left(x^{k-D^k_i} - \gamma \nabla f_i(x^{k-D^k_i})\right)_{[j]} & \text{if } i = i^k \text{ and } j \in \mathbf{S}^{k-D^k_i}, \\ x^{k-1}_{i[j]} & \text{otherwise} \end{cases}$$

$$x^k = \text{prox}_{\gamma r}\left(\overline{x}^k\right) \qquad \text{with} \quad \overline{x}^k := \sum_{i=1}^M \pi_i x^k_i.$$

This sparsification corresponds to one iteration of a stochastic block-coordinate descent, locally at the worker. However, our algorithm does not correspond to an asynchronous stochastic block-coordinate descent algorithm [17, 30, 25, 28], since the iteration is made from $\overline{x}^k$ aggregating asynchronously all the workers contributions. This originality is inspired from [22, 21]: though it may appear conservative, it actually performs well in practice due to the stability of the produced iterations; the intuition being that combining delayed points is more stable than using a combination of delayed directions; see the numerical comparisons of [21].

Thus, using the terminology [30], the local iterate $x^k_{i[j]}$ of our algorithm suffers from the combination of two kinds of delays: i) a deterministic unbounded delay coming from asynchronous communications, and ii) stochastic delays coming from the local sparsification at worker i. The co-existence of both deterministic and stochastic delays calls for an original mathematical analysis.

Distributed Implementation. We include the above-defined sparsified proximal gradient iteration in a distributed algorithm. The proposed algorithm `SPY` is generic as none of its ingredients (including the stepsize choice) depend on the computing system (data distribution, agents delays, ...). A unique feature of this algorithm is that though each master update relies on only one agent (and thus part of the data), all the data is always implicitly involved in the master variable, with even proportions. This allows the algorithm to cope with the heterogeneity of the computing system. Its presentation uses the following notation: for a vector of $x \in \mathbb{R}^d$ and a subset S of $\{1, .., d\}$, $[x]_S$ denotes the size-d vector with all zeros except for coordinates $j \in S$ and $x_{[S]}$ the size-$|S|$ vector with only the coordinates $j \in S$. Thus we have $x \in \mathbb{R}^d, [x]_S \in \mathbb{R}^d, x_{[S]} \in \mathbb{R}^{|S|}$ and $([x]_S)_{[j]} = x_{[j]}$ iff $j \in S$.

`SPY` algorithm

Master

Initialize $\bar{x}^0$
while *not converged* **do**
  Receive $[\Delta^k]_{\mathbf{S}^{k-D^k_{i_k}}}$ from agent i^k
  $\overline{x}^k \leftarrow \overline{x}^{k-1} + \pi_i [\Delta^k]_{\mathbf{S}^{k-D^k_i}}$
  $x^k \leftarrow \text{prox}_{\gamma r}(\bar{x}^k)$
  Choose sparsity mask $\mathbf{S}^k$
  Send $x^k, \mathbf{S}^k$ to agent i^k
end

Slave i

Initialize $x_i = x_i^+ = x = \bar{x}^0$
while *not interrupted by master* **do**
  $[x_i^+]_{\mathbf{S}} \leftarrow [x - \gamma \nabla f_i(x)]_{\mathbf{S}}$
  $\Delta \leftarrow x_i^+ - x_i$
  Send $[\Delta]_{\mathbf{S}}$ to master
  $[x_i]_{\mathbf{S}} \leftarrow [x_i^+]_{\mathbf{S}}$
  Receive x and $\mathbf{S}$ from master
end

The communications per time are i) a blocking send/receive from a slave to the master (in blue) of size $|\mathbf{S}|$; and ii) a blocking send/receive from the master to the last updating slave (in red) of the current iterate. The up-communication is thus sparse by sparsification and the down-communication cost depends on the structure of x^k, which is the result of a proximal operation of the regularization. In the case where the

ℓ_1-regularizer is used, x^k becomes sparse after some iterations, leading to a two-way sparse algorithm. This particular case will be investigated in details in Section 4.

3.2 Convergence Analysis

We study the convergence properties of the algorithm under standard assumptions on the learning problem (1) and no assumption on the system (neither on delays nor on data distribution).

Assumption 1 (On the problem)**.** *All functions $\{f_i\}_{i=1,..,M}$ are μ-strongly convex and L-smooth, i.e. differentiable with Lipschitz continuous gradient; and r is convex and lower-semi continuous.*

For the analysis, we thus assume that the functions f_i share the same strong convexity constant. In practice, we could include a proportion of ℓ_2 regularization to uniformize the strong convexity. We also assume that the functions f_i share the same smoothness constant, which is also fine as the f_i's are typically averaged, and not summed, risk functions (so that data imbalanced is not an issue here). Note finally that this assumption implies that there is a unique minimizer $x^\star$ to (1).

As for the sparsity masks, we propose two uniform choices. At this point, there is no reason *a priori* to change probabilities between agents or coordinates (this will be changed in Section 4).

Assumption 2 (On the random sparsification)**.** *The sparsity mask selectors $(\mathbf{S}^k)$ are independent and identically distributed random variables such that for a probability $p \in [0,1]$, either:*

- **Option I.** *We select a coordinate in the mask based on the uniform probability p: for all $j, j' \in \{1,..,d\}$ $\mathbb{P}[j \in \mathbf{S}^k] = \mathbb{P}[j' \in \mathbf{S}^k] = p > 0$.*
- **Option II.** *We draw pd coordinates uniformly: $|\mathbf{S}^k| = pd \in \{1,..,d\}$ almost surely and for all $j, j' \in \{1,..,d\}$ $\mathbb{P}[j \in \mathbf{S}^k] = \mathbb{P}[j' \in \mathbf{S}^k]$.*

Furthermore, the delays $(D_i^k)_{i=1,..,M}$ are independent of the future mask selectors $\{\mathbf{S}^\ell\}_{\ell \geq k}$.

The definition of the random set of coordinates in Option I is simple but has a disadvantage: the size of the set can be anything between 0 and n with nonzero probability. Although the size will be pn in expectation, we cannot guarantee that every communication will be sparse. This is why Option II fixes the size of the set $\mathbf{S}$ to $|\mathbf{S}| = pn$ and draw uniformly pn coordinates.

We emphasize that this is a rather light assumption on $\mathbf{S}^k$, as it is independent of both the iterations and computing system. We do not assume that the update delays are bounded or independent of the previous mask selectors $(\mathbf{S}^k)$, we only put the natural assumption that at any k, the delays since the last update may depend on past mask selectors $\{\mathbf{S}^\ell\}_{\ell<k}$ but not on the future ones.

We define an auxiliary sequence of stopping times [1] (k_m) defined iteratively as the first time since the previous terms when all slaves have made at least two updates. Mathematically, the definition is

$$k_0 = 0 \quad \text{and} \quad k_{m+1} = \min\left\{k : k - D_i^k \geq k_m \text{ for all } i\right\}. \tag{3}$$

This sequence will be at the center of the following convergence rate result, since it directly embeds the number of machines and the delays, and thus automatically adapts the various situations. For instance, we treat the usual case of bounded delays as a corollary.

Theorem 1 (Convergence rate)**.** *Suppose that Assumptions 1 and 2 hold. Take $\gamma \in (0, 2/(\mu + L)]$. Then, for all $k \in [k_m, k_{m+1})$,*

$$\mathbb{E}[\|x^k - x^\star\|^2] \;\leq\; \left(1 - 2\frac{\gamma\, p\, \mu\, L}{\mu + L}\right)^m \max_{i=1,..,M} \|x_i^0 - x_i^\star\|^2.$$

[1] To be precise with respect to random stopping time nature of the (k_m), let us define the filtration $\mathcal{F}^k = \sigma(\{\mathbf{S}^\ell\}_{\ell<k})$ so that all variables at time k (x_i^k, $\overline{x}^k$, x^k, d_i^k, D_i^k, etc.) are $\mathcal{F}^k$-measurable but $\mathbf{S}^k$ is not. As for all i, D_i^k is $\mathcal{F}^k$-measurable, k_m is an $\mathcal{F}^{k_m}$ stopping time. The filtration will also appear in the convergence proof.

Proof. The original construction of our algorithm, based on the average of the parameter rather than only gradients, allows us to have simple steps in the proof. We just need to introduce judicious variables, following [22]. First, from the unique solution $x^\star$ of (1), let us define for any slave i the point $x_i^\star := x^\star - \gamma \nabla f_i(x^\star)$. From those, one can derive $\overline{x}^\star := \sum_{i=1}^M \pi_i x_i^\star$. Then, as $0 \in \sum_i \pi_i \nabla f_i(x^\star) + \partial g(x^\star)$ by first-order optimality conditions, we have that $\overline{x}^\star = \sum_{i=1}^M \pi_i x_i^\star = x^\star - \gamma \sum_{i=1}^M \pi_i \nabla f_i(x^\star) \in x^\star + \gamma \partial g(x^\star)$ which directly leads to $\operatorname{prox}_{\gamma r}(\overline{x}^\star) = x^\star$ (see Chap. 16 of [3]).

For a time k and a slave i, we have that $x_i^k = x_i^{k-d_i^k}$ depends on i) $x^{k-D_i^k}$ which is $\mathcal{F}^{k-D_i^k}$-measurable; and ii) $\mathbf{S}^{k-D_i^k}$ which does not have this property but is i.i.d.. We now consider first using Option I of Assumption 2

$$\begin{aligned}\mathbb{E}[\|x_i^k - x_i^\star\|^2 | \mathcal{F}^{k-D_i^k}] &= \mathbb{E}[\|x_i^{k-d_i^k} - x_i^\star\|^2 | \mathcal{F}^{k-D_i^k}] = \sum_{j=1}^d \mathbb{E}[(x_{i[j]}^{k-d_i^k} - x_{i[j]}^\star)^2 | \mathcal{F}^{k-D_i^k}] \\ &= p\|x^{k-D_i^k} - \gamma \nabla f_i(x^{k-D_i^k}) - (x^\star - \gamma \nabla f_i(x^\star))\|^2 + (1-p)\|x_i^{k-D_i^k} - x_i^\star\|^2.\end{aligned}$$

In the other case, when we use Option II of Assumption 2, we have the same bound, since:

$$\begin{aligned}\mathbb{E}[\|x_i^k - x_i^\star\|^2 | \mathcal{F}^{k-D_i^k}] &= \mathbb{E}[\mathbb{E}[\|x_i^k - x_i^\star\|^2 | \mathcal{F}^{k-D_i^k}, \mathbf{S}^{k-D_i^k}], \mathcal{F}^{k-D_i^k}] \\ &= \mathbb{E}[\sum_{j=1}^d p(x_{[j]}^{k-D_i^k} - \gamma \nabla f_i(x^{k-D_i^k})_{[j]} - x_{i[j]}^\star)^2 + (1-p)(x_{i[j]}^{k-D_i^k} - x_{i[j]}^\star)^2 | \mathcal{F}^{k-D_i^k}] \\ &= \sum_{j=1}^d p(x_{[j]}^{k-D_i^k} - \gamma \nabla f_i(x^{k-D_i^k})_{[j]} - x_{i[j]}^\star)^2 + (1-p)(x_{i[j]}^{k-D_i^k} - x_{i[j]}^\star)^2 \\ &= p\|x^{k-D_i^k} - \gamma \nabla f_i(x^{k-D_i^k}) - (x^\star - \gamma \nabla f_i(x^\star))\|^2 + (1-p)\|x_i^{k-D_i^k} - x_i^\star\|^2\end{aligned}$$

We now use the μ-strong convexity and L-smoothness of f_i to write (see e.g. Lemma 3.11 of [4]),

$$\begin{aligned}&\|x^{k-D_i^k} - \gamma \nabla f_i(x^{k-D_i^k}) - (x^\star - \gamma \nabla f_i(x^\star))\|^2 \\ &\qquad \leq \left(1 - \frac{2\gamma\mu L}{\mu + L}\right) \left\|x^{k-D_i^k} - x^\star\right\|^2 - \gamma \left(\frac{2}{\mu + L} - \gamma\right) \left\|\nabla f_i(x^{k-D_i^k}) - \nabla f_i(x^\star)\right\|^2.\end{aligned}$$

Thus, for any $\gamma \in (0, 2/(\mu + L)]$, one can drop the last non-negative term,

$$\begin{aligned}\mathbb{E}[\|x_i^k - x_i^\star\|^2 | \mathcal{F}^{k-D_i^k}] &\leq p\left(1 - \frac{2\gamma\mu L}{\mu + L}\right) \left\|x^{k-D_i^k} - x^\star\right\|^2 + (1-p)\|x_i^{k-D_i^k} - x_i^\star\|^2 \\ &\leq p\left(1 - \frac{2\gamma\mu L}{\mu + L}\right) \left\|\overline{x}^{k-D_i^k} - \overline{x}^\star\right\|^2 + (1-p)\|x_i^{k-D_i^k} - x_i^\star\|^2,\end{aligned}$$

where we used that $\|x^{k-D_i^k} - x^\star\|^2 = \|\operatorname{prox}_{\gamma r}(\overline{x}^{k-D_i^k}) - \operatorname{prox}_{\gamma r}(\overline{x}^\star)\|^2 \leq \|\overline{x}^{k-D_i^k} - \overline{x}^\star\|^2$ by definition and non-expansiveness of the proximity operator of r. Taking full expectation on both sides, we get

$$\mathbb{E}\|x_i^k - x_i^\star\|^2 \leq p\left(1 - \frac{2\gamma\mu L}{\mu + L}\right) \mathbb{E}\left\|\overline{x}^{k-D_i^k} - \overline{x}^\star\right\|^2 + (1-p)\mathbb{E}\|x_i^{k-D_i^k} - x_i^\star\|^2.$$

Then, using that $\overline{x}^{k-D_i^k} - \overline{x}^\star = \sum_{i=1}^M \pi_i (x_i^{k-D_i^k} - \overline{x}_i^\star)$ and the convexity of $\|\cdot\|^2$, we get

$$\begin{aligned}
\mathbb{E}\|x_i^k - x_i^\star\|^2 &\leq p\left(1 - \frac{2\gamma\mu L}{\mu+L}\right)\sum_{j=1}^M \pi_j \mathbb{E}\left\|x_j^{k-D_i^k} - x_j^\star\right\|^2 + (1-p)\mathbb{E}\|x_i^{k-D_i^k} - x_i^\star\|^2 \\
&\leq p\left(1 - \frac{2\gamma\mu L}{\mu+L}\right)\max_{j=1,..,M} \mathbb{E}\left\|x_j^{k-D_i^k} - x_j^\star\right\|^2 + (1-p)\max_{j=1,..,M} \mathbb{E}\|x_j^{k-D_i^k} - x_j^\star\|^2 \\
&\leq \left(1 - 2\frac{\gamma p\mu L}{\mu+L}\right)\max_{j=1,..,M} \mathbb{E}\left\|x_j^{k-D_i^k} - x_j^\star\right\|^2.
\end{aligned}$$

Let $c_k = \max_{i=1,..,M} \mathbb{E}\left\|x_j^k - x_j^\star\right\|^2$ and $\alpha = 2\gamma p\mu L/(\mu+L)$, then the above result implies the following bound: $c_k \leq (1-\alpha)\max_{j=1,..,M} c_{k-D_j^k}$. Using the definition of the sequence (k_m), we get

$$\begin{aligned}
c_{k_m} &\leq (1-\alpha)\max_j c_{k_m - D_j^{k_m}} \leq (1-\alpha)\max_{\ell\in[k_{m-1},k_m)} c_\ell \\
c_{k_m+1} &\leq (1-\alpha)\max(c_{k_m}, \max_{\ell\in[k_{m-1},k_m)} c_\ell) \leq (1-\alpha)\max_{\ell\in[k_{m-1},k_m)} c_\ell.
\end{aligned}$$

Thus for all $k \geq k_m$, $c_k \leq (1-\alpha)\max_{\ell\in[k_{m-1},k_m)} c_\ell$. This implies that the sequence $\widetilde{c}_m := \max_{\ell\in[k_m,k_{m+1})} c_\ell$ is exponentially decreasing:

$$\widetilde{c}_m \leq (1-\alpha)\widetilde{c}_{m-1} \leq (1-\alpha)^m \widetilde{c}_0 \leq (1-\alpha)^m \max_{i=1,..,M} \|x_i^0 - x_i^\star\|^2.$$

Finally, it suffices to use once again the non-expansivity of the proximity operator of r and the definitions to get that for all $k \in [k_m, k_{m+1})$,

$$\mathbb{E}\|x^k - x^\star\|^2 \leq \mathbb{E}\|\overline{x}^k - \overline{x}^\star\|^2 \leq \sum_{i=1}^M \pi_i \mathbb{E}\|x_i^k - x_i^\star\|^2 \leq c_k \leq (1-\alpha)^m \max_{i=1,..,M} \|x_i^0 - x_i^\star\|^2,$$

which concludes the proof. □

This results shows that the proposed algorithm converges linearly at a rate that only depends on the function properties but neither on the number of machines nor on the delays as they are directly embedded in the sequence (k_m). When we consider bounded delays which is the standard assumption, the particularization of the previous result leads to a rate depending as expected on the bound on delays, as formalized in the following corollary.

Corollary 2. *Suppose that Assumptions 1 and 2 hold. Suppose furthermore that the delays are bounded over time and machines: $d_i^k \leq d$ for all k and i. Take $\gamma \in (0, 2/(\mu+L)]$. Then, for all k,*

$$\mathbb{E}[\|x^k - x^\star\|^2] \;\leq\; \left(1 - 2\frac{\gamma\, p\, \mu\, L}{(2d+1)(\mu+L)}\right)^k \max_{i=1,..,M} \|x_i^0 - x_i^\star\|^2.$$

Proof. From definition (3), we have that $k_{m+1} - k_m \leq 2d+1$, by the boundedness assumption. Therefore $k_m \leq (2d+1)m$. For all $k \in [k_m, k_{m+1})$, we deduce $k/(2d+1) \leq m$. Notice also that for all $u \in (0,1)$, the concavity of the function $u \mapsto (1-u)^{1/(2d+1)}$ gives that $(1-u)^{1/(2d+1)} \leq 1 - u/(2d+1)$. The bound of Theorem 1 thus leads to

$$\left(1 - 2\frac{\gamma\, p\, \mu\, L}{(\mu+L)}\right)^m \leq \left(1 - 2\frac{\gamma\, p\, \mu\, L}{(\mu+L)}\right)^{k/(2d+1)} \leq \left(1 - 2\frac{\gamma\, p\, \mu\, L}{(2d+1)(\mu+L)}\right)^k.$$

This gives the rate depending on the bound d. □

In Theorem 1 and its corollary, it is important to notice that the stepsize γ can be taken in the usual range for (proximal) gradient descent and in particular does not depend on the delays in any way. (In fact, the delays and the computing system appear nowhere in the algorithm). This is in contrast with existing asynchronous algorithms, even for the works with the most realistic assumptions; see [7, 1, 17, 30] and references therein. The traditional stepsizes were previsously known to usually work in practice (see e.g. [30, Footnote 1]). Our original proof technique allows us to establish it theoretically.

We also emphasize that even though the probability p of coordinate selection of Assumption 2 appears in the rate, it does not appear neither in the range of admissible stepsizes. Thus, with the same stepsize as in the classical proximal gradient, sparsification is possible at the expense of an accordingly slower rate.

4 Identification and Sparsity for ℓ_1-regularized problems

In the case of ℓ_1-regularized learning problems, i.e. with $r(x) = \lambda\|x\|_1$ in problem (1), our algorithm reaches some sparsity structure, which leads to algorithmic improvements.

4.1 Sparsity Identification and Resulting Sparsification of Communications

Regularization with ℓ_1-norm induces sparsity [2]. It has been shown that proximal-gradient methods identify such sparsity patterns which means that they produce iterates that reach the same support as the optimal solution of (1). Unfortunately, randomness may break this identification property. For example, it is well-known that for the proximal stochastic gradient descent, this sparse structure may not be identified with probability one; see e.g. [14] and a counter-example in the recent article [26].

For our algorithm, we proved in Section 3.2 its L2 convergence without any assumption on delays. This yields the convergence in probability, which unfortunately does not implies substructure identification in general, as it is the case for the prox-SGD. In this section, we prove that our algorithm identifies a near-optimal substructure under an additional light assumption over the delays. To the best of our knowledge, this is the first time that such a property is shown for an asynchronous algorithm.

Assumption 3 (Additional assumption for identification)**.** *The number of iterations between two full updates cannot grow exponentially, i.e., there is C such that for any $\varepsilon > 0$ and all m, $k_{m+1} - k_m \leq C(1+\varepsilon)^m$. This assumption is mild and subsumes the usual bounded delay assumption.*

For a vector $x \in \mathbb{R}^n$, we define its *support* $\mathrm{supp}(x) \subseteq \{1, .., d\}$ as the set $\mathrm{supp}(x) = \{i \in \{1, .., d\} : x_{[i]} \neq 0\}$, and we note $|\mathrm{supp}(x)|$ the size of the support of x. We show that for ℓ_1-regularized problems, the master iterates (x^k) of our algorithm reaches the sparsity pattern of the solution (or a slightly enlarged one) in finite time with probability one.

Theorem 3 (Identification)**.** *Let Assumptions 1, 2, and 3 hold. Then, for $r(x) = \lambda\|x\|_1$, the algorithm identifies a near-optimal support in finite time with probability one: there is K such that*

$$\mathrm{supp}(x^\star) \subseteq \mathrm{supp}(x^k) \subseteq \mathrm{supp}(y_\varepsilon^\star) \qquad \textit{for all } k \geq K,$$

where $y_\varepsilon^\star = \mathrm{prox}_{\gamma(1-\varepsilon)r}(\overline{x}^\star - x^\star)$ for any $\varepsilon > 0$. Furthermore, if the problem is non-degenerate, i.e. $-\sum_{i=1}^M \pi_i \nabla f_i(x^\star) \in \mathrm{ri}\ \partial r(x^\star)$ then, the algorithm identifies the optimal support: w.p. 1, $\exists K$:

$$\mathrm{supp}(x^k) = \mathrm{supp}(x^\star) \qquad \textit{for all } k \geq K.$$

Proof. The proof is divided into two parts: i) the proof of the almost sure convergence of iterates; and ii) the use of general identification results. First, from the proof of Theorem 1, one get that

$$\mathbb{E}\|x^k - x^\star\|^2 \leq \mathbb{E}\|\overline{x}^k - \overline{x}^\star\|^2 \leq C(1-\alpha)^m, \tag{4}$$

for some C and $\alpha \in]0,1]$. Notice that the exponent is not the iteration number but rather the number of stopping times (k_m) before k. Thus,

$$\mathbb{E}\left[\sum_{k=1}^{+\infty} \|x^k - x^\star\|^2\right] \leq \mathbb{E}\left[\sum_{m=1}^{+\infty} \sum_{k=k_m}^{k_{m+1}-1} \|x^k - x^\star\|^2\right]$$
$$\leq C \sum_{m=1}^{+\infty} (k_{m+1} - 1 - k_m)(1-\alpha)^m.$$

Assumption 3 now allows us to bound $k_{m+1} - 1 - k_m$ and to get that this sum is finite almost surely:

$$1 = \mathbb{P}\left[\sum_{k=1}^{+\infty} \|x^k - x^\star\|^2 < +\infty\right] \leq \mathbb{P}\left[\|x^k - x^\star\|^2 \to 0\right],$$

which in turn implies that (x^k) converges almost surely to $x^\star$. The same reasoning holds for $\overline{x}^k$ from (4). Thus, as $x^k = \text{prox}_{\gamma r}(\overline{x}^k)$, one has that $u^k := (\overline{x}^k - x^k)/\gamma \in \partial r(x^k)$ and (u_k) converges almost surely to $u^\star := (\overline{x}^\star - x^\star)/\gamma$.

Now that the almost sure convergence of (x^k) and (u_k) with $u^k \in \partial r(x^k)$, we can use Theorem 1 of [6] to get the first part of the result. The right hand side is obtained from Eq. (2.8) of [6] by noticing that a coordinate i of the right term is equal to zero for function $\lambda\|\cdot\|_1$ if and only if $u^\star_{[i]} \in]-\lambda, \lambda[$. As for any $\varepsilon > 0$, $\text{prox}_{(1-\varepsilon)r}(u^\star)_{[i]} = 0$ if and only if $u^\star_{[i]} \in [-(1-\varepsilon)\lambda, (1-\varepsilon)\lambda] \subset]-\lambda, \lambda[$, then, the sparsity pattern of $y^\star_\varepsilon = \text{prox}_{\gamma(1-\varepsilon)r}(\gamma u^\star)$ is slightly larger than the right one of (2.8).

Finally, if $u^\star = -\sum_{i=1}^M \pi_i \nabla f_i(x^\star) \in \text{ri}\ \partial r(x)$, then, for any i such that $x_{[i]} = 0$, $u^\star_{[i]} \in \text{ri}\ \partial\lambda|x_{[i]}| =]-\lambda, \lambda[$ so the sparsity patterns of the right and left bounds match. □

This result implies that for problems having a sparse solution, the point sent to slaves will eventually become sparse. Thus for ℓ_1 regularized learning problems, our distributed algorithm has, structurally, two-way sparse communications. In addition to the sparsification of communications, the identification can be further leveraged as an algorithmic advantage, as explained in the next section.

Remark 1 (Extensions to other types of sparsity)**.** *The identification result of Theorem 3 is formulated with an ℓ_1-regularization in order to focus on the sparsity in the main text of the paper however, it can be easily extended to a vast number of regularizations, as the total variation, $\ell_{1,2}$ norm, etc. In general, any mirror-stratifiable function [6] will produce structure-identifying iterates. The technical ingredients used for this identification are*

$$\begin{cases} \overline{x}^k \longrightarrow \overline{x}^\star \text{ almost surely} \\ x^k = \textbf{prox}_{\gamma r}(\overline{x}^k) \end{cases},$$

which are obtained i) by the linear convergence of our algorithm in L2 and ii) by its proximal construction. Then, the steps of Theorem 3 can be mimicked for mirror-stratifiable function r. Thus extensions of our algorithm could actually identify low-complexity manifolds for a more general class of functions. For clarity, we only consider the case of the ℓ_1 norm in the paper.

4.2 Automatic Support Adaptation

Upon noticing that the (or most of) iterates outside the solution pattern will stay null after some time, it is natural to foster the coordinates in the mask of the master iterate x^k in terms of update. Practically, this means giving them a higher probability in the sparsity mask selector $\mathbf{S}^k$.

Assumption 4 (On the sparsification with support identification)**.** *The sparsity mask selectors $(\mathbf{S}^k)$ are random variables such that $\mathbb{P}[j \in \mathbf{S}^k] = 1$ if $j \in \text{supp}(x^k)$ and either:*

Option III. $\mathbb{P}[j' \in \mathbf{S}^k] = p > 0$ *for all* $j' \notin \text{supp}(x^k)$.

Option IV. $|\mathbf{S}^k| = |\text{supp}(x^k)| + p(d - |\text{supp}(x^k)|) \in \{1, .., d\}$ *almost surely and* $\mathbb{P}[j' \in \mathbf{S}^k] = \mathbb{P}[j'' \in \mathbf{S}^k]$ *for all* $j', j'' \notin \text{supp}(x^k)$.

Furthermore, the delays $(D_i^k)_{i=1,..,M}$ *are independent of the future mask selectors* $\{\mathbf{S}^\ell\}_{\ell \geq k}$.

SPY-DR algorithm

Slave i

Initialize $x_i = x_i^+ = x = \bar{x}^0$
while *not interrupted by master* **do**
  $[x_i^+]_{\mathbf{S}\setminus\text{supp}(x)} \leftarrow [x - \gamma\nabla f_i(x)]_{\mathbf{S}\setminus\text{supp}(x)}$
  $[x_i^+]_{\text{supp}(x)} \leftarrow p[x - \gamma\nabla f_i(x)]_{\text{supp}(x)} + (1-p)[x_i]_{\text{supp}(x)}$
  $\Delta \leftarrow x_i^+ - x_i$
  Send $[\Delta]_{\mathbf{S}}$ to master
  $[x_i]_{\mathbf{S}} \leftarrow [x_i^+]_{\mathbf{S}}$
  Receive x and $\mathbf{S}$ from master
end

This algorithm is similar to SPY except for handling of the support of the received point by a slave (the master part remains the same). The following result gives convergence and identification with the same rate as Theorem 1 but SPY-DR may be more efficient in practice since in the first iterations, the mask of the master point is usually full and thus so are the updates, resulting in a faster convergence and identification.

Theorem 4 (Convergence rate of SPY-DR)**.** *Let Assumptions 1, 3, and 4 hold. Take* $\gamma \in (0, 2/(\mu + L)]$. *Then, for all* $k \in [k_m, k_{m+1})$,

$$\mathbb{E}\|x^k - x^\star\|^2 \leq \left(1 - 2\frac{\gamma p \mu L}{\mu + L}\right)^m \max_{i=1,..,M} \|x_i^0 - x_i^\star\|^2$$

and identification in the sense of Theorem 3 holds.

Proof. With the same notations as in the proof of Theorem 1, we have

$$\begin{aligned}
\mathbb{E}[\|x_i^k - x_i^\star\|^2 | \mathcal{F}^{k-D_i^k}] &= \mathbb{E}[\|x_i^{k-d_i^k} - x_i^\star\|^2 | \mathcal{F}^{k-D_i^k}] = \sum_{j=1}^d \mathbb{E}[([x_i^{k-d_i^k} - x_i^\star]_{[j]})^2 | \mathcal{F}^{k-D_i^k}] \\
&= \sum_{j \notin \text{supp}(x^{k-d_i^k})} p\Big([x^{k-D_i^k} - \gamma\nabla f_i(x^{k-D_i^k}) - x_i^\star]_{[j]}\Big)^2 + (1-p)\Big([x_i^{k-D_i^k} - x_i^\star]_{[j]}\Big)^2 \\
&\quad + \sum_{j \in \text{supp}(x^{k-d_i^k})} \Big([p(x^{k-d_i^k} - \gamma\nabla f_i(x^{k-d_i^k})) + (1-p)x_i^{k-D_i^k}]_{[j]} - [x_i^\star]_{[j]}\Big)^2 \\
&\leq \sum_{j \in \text{supp}(x^{k-d_i^k})} p\Big([x^{k-D_i^k} - \gamma\nabla f_i(x^{k-D_i^k}) - x_i^\star]_{[j]}\Big)^2 + (1-p)\Big([x_i^{k-D_i^k} - x_i^\star]_{[j]}\Big)^2 \\
&\quad + \sum_{j \notin \text{supp}(x^{k-d_i^k})} p\Big([x^{k-D_i^k} - \gamma\nabla f_i(x^{k-D_i^k}) - x_i^\star]_{[j]}\Big)^2 + (1-p)\Big([x_i^{k-D_i^k} - x_i^\star]_{[j]}\Big)^2 \\
&= p\|x^{k-D_i^k} - \gamma\nabla f_i(x^{k-D_i^k}) - (x^\star - \gamma\nabla f_i(x^\star))\|^2 + (1-p)\|x_i^{k-D_i^k} - x_i^\star\|^2
\end{aligned}$$

where the inequality follows from convexity of $\|\cdot\|^2$. The rest of the proof follows the same arguments as the one of Theorem 1. The identification part also follows unchanged. □

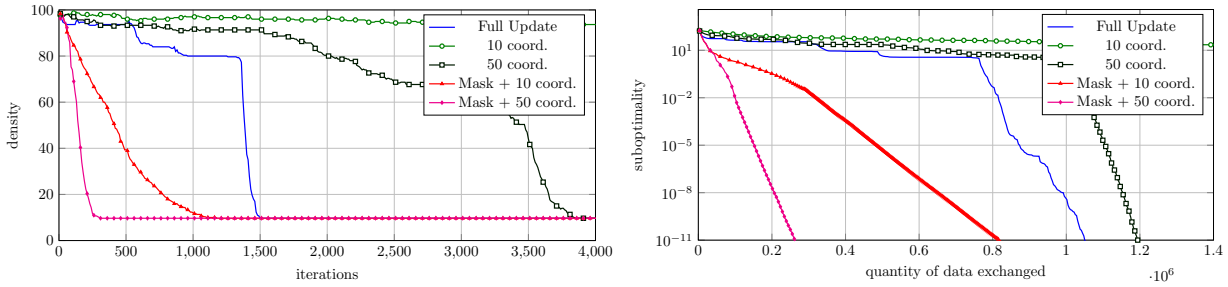

Figure 1: Performance of the algorithms on the synthetic lasso problem: identification properties (at the left) and convergence with respect to communication cost (at the right).

5 Numerical Illustrations

In this section, we illustrate the communication gain provided by our random sparsification algorithms on two classic ℓ_1 regularized empirical risk minimization problems.

Problems. We first consider a synthetic LASSO problem

$$\min_{x\in\mathbb{R}^d} \quad \|Ax-b\|^2+\lambda_1\|x\|_1$$

with $n=10,000$ examples and $d=300$ features. A is generated from the standard normal distribution, $b=Ax_0+e$ where x_0 is a 90% sparse vector and e is taken from the normal distribution with standard deviation 0.01. We take λ_1 to reach the sparsity of 90%.

We also examine the regularized logistic regression with elastic net

$$\min_{x\in\mathbb{R}^d} \quad \frac{1}{n}\sum_{j=1}^{n}\log(1+\exp(-y_j z_j^\top x))+\lambda_1\|x\|_1+\frac{\lambda_2}{2}\|x\|_2^2$$

on two data-sets from the LibSVM repository: the *madelon* data-set ($n=2,000$ $d=500$) with hyperparameters $\lambda_2=0.01$ and $\lambda_1=0.001$, chosen to reach an 70% sparsity; the *epsilon* dataset ($n=100,000$ $d=2,000$) with parameters $\lambda_1=8.10^{-4}$ and $\lambda_2=0.01$ to reach an 40% sparsity.

Experimental set-up. We used Python, and MPI (Message Passing Interface) for the distributed communications framework. To communicate sparse vectors, we send a list of coordinates then their values, as usual in sparse communications.

We run our experiments on a machine with 32 cores and 256 Gb of RAM: one core plays the role the master, 10 cores are the slaves. The data sets are split evenly between the 10 slaves, each having access only to its own part.

Algorithms. We compare three algorithms:

- 'Full update': `SPY` without any sparsification;
- 'xxx coord.': `SPY` where the sparsification mask is taken as xxx randomly chosen coordinates;
- 'Mask + xxx coord.': `SPY-DR` where the mask takes all the coordinates in the support of the received point plus xxx randomly chosen other ones.

We display the performance of the algorithms in two ways: i) size of support vs number of iterations, showing the identification properties; and ii) functional suboptimality vs communication cost, modelled as the number of couples (coordinate, value) sent from and to the master.

Experimental results. Figure 1 reports the convergence of the algorithms for the lasso problem. We observe that the use of the mask in the coordinate selection enables faster decrease of the support of the

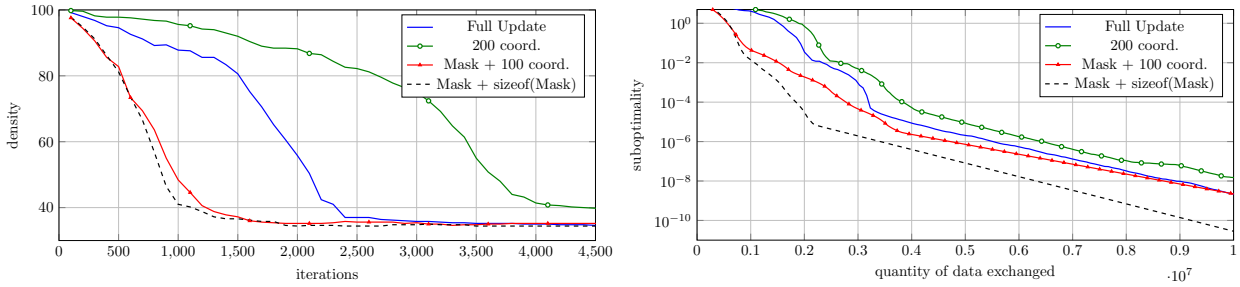

Figure 2: Performance of the algorithms on the logistic regression problem with madelon dataset: identification properties (at the left) and convergence with respect to communication cost (at the right).

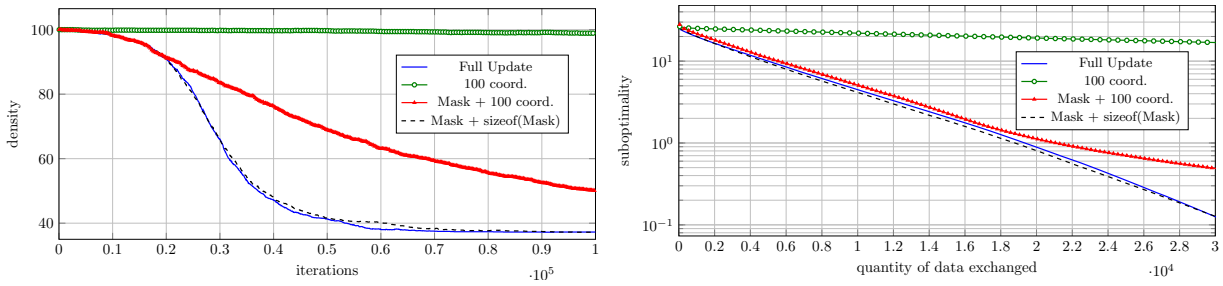

Figure 3: Performance of the algorithms on the logistic regression problem with the epsilon dataset: identification properties (at the left) and convergence with respect to communication cost (at the right).

iterates ('density') compared to uniform selection. Thanks to this faster identification, the iteration cost in terms of quantity of data exchanged decreases quickly at the same time as the iterates reach a sparse neighborhood of the optimal solution, which makes the mask coordinate selection converge faster than the full update.

For the first logistic regression problem, Figure 2 shows similar behavior as in the lasso case, with mask-algorithm communicating only relevant data in view of both identification and optimization.

For the second logistic regression problem, Figure 3 shows a slightly different behavior. Since the optimal sparsity is only 40%, strategies based on randomly selecting a small fixed number of coordinates do not achieve identification fast enough, thus degrading the performance. However, the adaptive approach consisting in selecting all coordinates in the mask plus the same number randomly outside of it reach the same performance than the full algorithm without sparsification. Thus, in this less favorable context, our algorithm proves to be robust to moderately sparse solutions.

Concluding remarks on numerical illustrations. Without sparsification, the algorithm exhibits a fast support identification but with many communications as the slave-to-master exchanges are not sparse. On the other side, with harsh sparsification, the identification is very slow which is damageable for the convergence speed. Thus, to efficiently harness sparsification, one has to use limited sparsification before identification (to identify quickly) and then sparsify more, privileging the coordinates in the identified support; which is the idea of `SPY-DR` and can be seen as automatic dimension reduction.

6 Conclusion

We presented an asynchronous distributed algorithm with sparse communications. We proved that the algorithm converges linearly with an efficient fixed stepsize and identifies near-optimal substructure. For ℓ_1-regularized problems, the sparsification allows a gain in convergence with respect to communication exchanges. Thus, this work advocates the interest of the ℓ_1-regularization for learning from scattered data, beyond statistical considerations and feature selection.

References

[1] Arda Aytekin, Hamid Reza Feyzmahdavian, and Mikael Johansson. Analysis and implementation of an asynchronous optimization algorithm for the parameter server. *arXiv preprint arXiv:1610.05507*, 2016.

[2] Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, et al. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, 2012.

[3] Heinz H Bauschke and Patrick L Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer Science & Business Media, 2011.

[4] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[5] Clément Calauzènes and Nicolas Le Roux. Distributed saga: Maintaining linear convergence rate with limited communication. *arXiv preprint arXiv:1705.10405*, 2017.

[6] Jalal Fadili, Jerome Malick, and Gabriel Peyré. Sensitivity analysis for mirror-stratifiable convex functions. *arXiv preprint arXiv:1707.03194*, 2017.

[7] Hamid Reza Feyzmahdavian, Arda Aytekin, and Mikael Johansson. A delayed proximal gradient method with linear convergence rate. In *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on*, pages 1–6. IEEE, 2014.

[8] Robert Hannah and Wotao Yin. On unbounded delays in asynchronous parallel fixed-point algorithms. *arXiv preprint arXiv:1609.04746*, 2016.

[9] Warren L Hare and Adrian S Lewis. Identifying active constraints via partial smoothness and prox-regularity. *Journal of Convex Analysis*, 11(2):251–266, 2004.

[10] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[11] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[12] John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10(Mar):777–801, 2009.

[13] Rémi Leblond, Fabian Pedregosa, and Simon Lacoste-Julien. Asaga: Asynchronous parallel saga. *arXiv preprint arXiv:1606.04809*, 2016.

[14] Sangkyun Lee and Stephen J Wright. Manifold identification in dual averaging for regularized stochastic online learning. *Journal of Machine Learning Research*, 13(Jun):1705–1744, 2012.

[15] Mu Li, David G Andersen, and Alexander Smola. Distributed delayed proximal gradient methods. In *NIPS Workshop on Optimization for Machine Learning*, volume 3, page 3, 2013.

[16] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

[17] Ji Liu, Stephen J Wright, Christopher Ré, Victor Bittorf, and Srikrishna Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *The Journal of Machine Learning Research*, 16(1):285–322, 2015.

[18] Chenxin Ma, Martin Jaggi, Frank E Curtis, Nathan Srebro, and Martin Takáč. An accelerated communication-efficient primal-dual optimization framework for structured machine learning. *arXiv preprint arXiv:1711.05305*, 2017.

[19] Chenxin Ma, Virginia Smith, Martin Jaggi, Michael Jordan, Peter Richtarik, and Martin Takac. Adding vs. averaging in distributed primal-dual optimization. In *International Conference on Machine Learning*, pages 1973–1982, 2015.

[20] M. Massias, A. Gramfort, and J. Salmon. Dual Extrapolation for Faster Lasso Solvers. In *ICML*, 2018.

[21] Konstantin Mishchenko, Franck Iutzeler, and Jérôme Malick. A distributed flexible delay-tolerant proximal gradient algorithm. *arXiv preprint arXiv:1806.09429*, 2018.

[22] Konstantin Mishchenko, Franck Iutzeler, Jérôme Malick, and Massih-Reza Amini. A delay-tolerant proximal-gradient algorithm for distributed learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 3587–3595, 2018.

[23] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[24] Fabian Pedregosa, Rémi Leblond, and Simon Lacoste-Julien. Breaking the nonsmooth barrier: A scalable parallel method for composite optimization. In *Advances in Neural Information Processing Systems*, pages 55–64, 2017.

[25] Zhimin Peng, Yangyang Xu, Ming Yan, and Wotao Yin. Arock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38(5):A2851–A2879, 2016.

[26] Clarice Poon, Jingwei Liang, and Carola-Bibiane Schönlieb. Local convergence properties of saga/prox-svrg and acceleration. *arXiv preprint arXiv:1802.02554*, 2018.

[27] Jineng Ren, Xingguo Li, and Jarvis Haupt. Communication-efficient algorithm for distributed sparse learning via two-way truncation. *arXiv preprint arXiv:1709.00537*, 2017.

[28] Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *The Journal of Machine Learning Research*, 17(1):2657–2681, 2016.

[29] Farnood Salehi, Patrick Thiran, and L Elisa Celis. Stochastic dual coordinate descent with bandit sampling. *arXiv preprint arXiv:1712.03010*, 2017.

[30] Tao Sun, Robert Hannah, and Wotao Yin. Asynchronous coordinate descent under more realistic assumptions. In *Advances in Neural Information Processing Systems*, pages 6183–6191, 2017.

[31] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[32] Jialei Wang, Mladen Kolar, Nathan Srebro, and Tong Zhang. Efficient distributed learning with sparsity. *arXiv preprint arXiv:1605.07991*, 2016.

[33] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. *arXiv:1710.09854*, 2017.

[34] Stephen J Wright. Identifiable surfaces in constrained optimization. *SIAM Journal on Control and Optimization*, 31(4):1063–1079, 1993.

[35] Ruiliang Zhang and James Kwok. Asynchronous distributed admm for consensus optimization. In *International Conference on Machine Learning*, pages 1701–1709, 2014.