MURANA: A Generic Framework for Stochastic Variance-Reduced Optimization

Laurent Condat [1] **Peter Richtárik** [1]

Abstract

We propose a generic variance-reduced algorithm, which we call MUltiple RANdomized Algorithm (MURANA), for minimizing a sum of several smooth functions plus a regularizer, in a sequential or distributed manner. Our method is formulated with general stochastic operators, which allow us to model various strategies for reducing the computational complexity. For example, MURANA supports sparse activation of the gradients, and also reduction of the communication load via compression of the update vectors. This versatility allows MURANA to cover many existing randomization mechanisms within a unified framework. However, MURANA also encodes new methods as special cases. We highlight one of them, which we call ELVIRA, and show that it improves upon Loopless SVRG.

1. Introduction

We consider the estimation of the model $x^\star \in \mathbb{R}^d$ arising as the solution of the optimization problem

$$\underset{x\in\mathbb{R}^d}{\text{minimize}} \left\{ R(x) + \frac{1}{M}\sum_{m=1}^{M} F_m(x) \right\}, \tag{1}$$

for some $M \geq 1$, where each convex function F_m is L-smooth, for some $L > 0$, i.e. $\frac{1}{L}\nabla F_m$ is nonexpansive, and $R : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is a proper, closed, convex function (Bauschke & Combettes, 2017), whose proximity operator

$$\mathrm{prox}_{\gamma R} : w \mapsto \underset{x\in\mathbb{R}^d}{\arg\min} \left\{ \gamma R(x) + \tfrac{1}{2}\|x - w\|^2 \right\}$$

is easy to compute, for any $\gamma > 0$ (Parikh & Boyd, 2014; Condat et al., 2019). We suppose that every F_m is μ-strongly convex, for some $\mu \geq 0$, i.e. $F_m - \frac{\mu}{2}\|\cdot\|^2$ is convex. We also suppose that $F := \frac{1}{M}\sum_{m=1}^M F_m$ is μ_F-strongly convex, for some $\mu_F > 0$, with $\mu_F \geq \mu$. We emphasize that μ can be zero, but we require **strong convexity of the global function** F. Since the problem is strongly convex, $x^\star$ exists and is unique.

In a distributed client-server setting, M is the number of parallel computing nodes, with an additional master node communicating with these M nodes. In a non-distributed setting, M is, for instance, the number of data points contributing to some training task. We let $[M] := \{1, \ldots, M\}$.

Randomized optimization algorithms. To formulate our algorithms, we will make use of several sources of randomness of the form

$$d^k = \mathcal{C}^k(\nabla F(x^k) - h^k), \tag{2}$$

where k is the iteration counter, $x^k \in \mathbb{R}^d$ is the model estimate converging to the desired solution $x^\star$, h^k is a control variate converging to $\nabla F(x^\star)$, and $\mathcal{C}^k(r)$ is a shorthand notation to denote a random realization of a stochastic process with expectation r, so that $\mathcal{C}^k(r)$ is a random unbiased estimate of the vector $r \in \mathbb{R}^d$. Although we adopt this notation as if $\mathcal{C}^k$ were a random operator, its argument r does not always have to be known or computed. For instance, if $\mathcal{C}^k(r) = \{\frac{1}{p}r$ with probability p, 0 else$\}$, r is not needed when the output is 0. This means that in (2), $\nabla F(x^k)$ is not computed in that case; this is the key reason why randomness makes it possible to decrease the overall complexity. The distribution of the random variable is not needed, and that is why we lighten the notations by omitting to write the underlying probability space structure. Indeed, we only need to know a constant $\omega \geq 0$ such that, for every $r \in \mathbb{R}^d$,

$$\mathbb{E}\left[\left\|\mathcal{C}^k(r) - r\right\|^2\right] \leq \omega\|r\|^2, \tag{3}$$

where the norm is the 2-norm and $\mathbb{E}$ denotes the expectation. Thus, if r tends to 0, not only does $\mathcal{C}^k(r)$ tend to 0, but the variance tends to 0 as well. Hence, in a step like in (2), d^k will converge to 0 and everything will work out so that the algorithm converges to the exact solution $x^\star$. That is, the proposed algorithm will be **variance reduced** (Gower et al., 2020b). In recent years, variance-reduced algorithms

[1] King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia. Correspondence to: Laurent Condat <see https://lcondat.github.io/>.

This work was presented at the International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2021 (FL-ICML'21). This workshop does not have official proceedings and this paper is non-archival. Copyright 2021 by the author(s).

like SAGA (Defazio et al., 2014) or SVRG (Johnson & Zhang, 2013; Zhang et al., 2013; Xiao & Zhang, 2014) have become the reference for finite-sum problems of the form (1) since they converge to the exact solution but can be M times faster than standard proximal gradient descent, which is typically a huge improvement.

Communication bottleneck in distributed and federated learning. In the age of big data, there has been a shift towards distributed computations, and modern hardware increasingly relies on the power of uniting many parallel units into a single system. Training large machine learning models critically relies on distributed architectures. Typically, the training data is distributed across several workers, which compute, in parallel, local updates of the model. These updates are then sent to a central server, which performs aggregation and then broadcasts the updated model back to the workers, to proceed with the next iteration. But communication of vectors between machines is typically much slower than computation, so **communication is the bottleneck**. This is even more true in the modern machine learning paradigm of **federated learning** (Konečný et al., 2016; McMahan et al., 2017; Kairouz et al., 2019; Li et al., 2020), in which a global model is trained in a massively distributed manner over a network of heterogeneous devices, with a huge number of users involved in the learning task in a collaborative way. Communication can be costly, slow, intermittent and unreliable, and for that reason the users ideally want to communicate the minimum amount of information. Moreover, they also do not want to share their data for privacy reasons. Therefore, **compression** of the communicated vectors, using various sketching, sparsification, or quantization techniques (Alistarh et al., 2017; Wen et al., 2017; Wangni et al., 2018; Albasyoni et al., 2020; Basu et al., 2020; Dutta et al., 2020; Sattler et al., 2020; Xu et al., 2021), has become the approach of choice. In recent works (Tang et al., 2019; Liu et al., 2020; Philippenko & Dieuleveut, 2020; Gorbunov et al., 2020), double, or bidirectional, compression is considered; that is, not only the vectors sent by the workers to the server, but also the model updates broadcast by the server to all workers, are compressed. MURANA accommodates for model or bidirectional compression using the operators $\mathcal{V}^k$; see Section 2.1.

A generic framework. Unbiased stochastic operators with conic variance, like in (3), allow to model a wide range of strategies: they can be used (i) for **sampling**, i.e. to select a subset of functions whose gradient is computed at every iteration, like in SAGA or SVRG, as mentioned above; (ii) for **compression**; in addition to the idea of communicating each vector only with some small probability, we can mention as example the *rand-k* operator, which sends k out of d elements, chosen at random and scaled by $\frac{d}{k}$, of its argument vector; (iii) to model **partial participation** in federated learning, with each user participating in a fraction of the communication rounds only. That is why we formulate MURANA with this type of operators, which have all these applications, and many more.

Contributions. We propose MUltiple RANdomized Algorithm (MURANA) – a generic template algorithm with several several sources of randomness that can model a wide range of computation, communication reduction strategies, or both at the same time (e.g. by composition, see Proposition 2). MURANA is variance reduced: it converges to the exact solution whatever the variance, which can be arbitrarily large. MURANA generalizes DIANA (Mishchenko et al., 2019; Horváth et al., 2019) in several ways and encompasses SAGA (Defazio et al., 2014) and loopless SVRG (Kovalev et al., 2020) as particular cases; we also give minibatch versions for them. Finally, we propose, as another new particular case of MURANA, a new algorithm, called ELVIRA, which improves upon loopless SVRG.

The proofs can be found in the long version of the paper: arXiv:2106.03056.

2. Proposed framework: MURANA

2.1. Three sources of randomness

We first introduce the **first set of stochastic operators**, $\mathcal{C}_m^k$, for every $k \geq 0$ and $m \in [M]$. In particular, we assume that such that there is a constant $\omega \geq 0$ such that for every $r \in \mathbb{R}^d$,

$$\mathbb{E}[\mathcal{C}_m^k(r)] = r \quad \text{and} \quad \mathbb{E}\left[\left\|\mathcal{C}_m^k(r) - r\right\|^2\right] \leq \omega \|r\|^2. \quad (4)$$

For every $(r, r') \in (\mathbb{R}^d)^2$ and $(m, m') \in [M]^2$, $\mathcal{C}_m^k(r)$ and $\mathcal{C}_{m'}^{k'}(r')$ at two different iteration indexes $k \neq k'$ are independent random variables. However, they can have different laws since only their first and second order statistics matter, as expressed in (4). Note that $\mathcal{C}_m^k(r)$ and $\mathcal{C}_{m'}^k(r')$ with $m \neq m'$ can be **dependent**, so $\left(\mathcal{C}_1^k(r_1), \ldots, \mathcal{C}_M^k(r_M)\right)$ should be viewed as a whole joint random process; this is needed for sampling or partial participation, for instance, where $N < M$ indexes in $[M]$ are chosen at random; see Proposition 1 below.

Next, we introduce the **second set of stochastic operators**, $\mathcal{U}_m^k$, with same properties: for every $k \geq 0$, $m \in [M]$, $r \in \mathbb{R}^d$,

$$\mathbb{E}\left[\mathcal{U}_m^k(r)\right] = r \quad \text{and} \quad \mathbb{E}\left[\left\|\mathcal{U}_m^k(r) - r\right\|^2\right] \leq \chi \|r\|^2, \quad (5)$$

for some constant $\chi \geq 0$, and same dependence properties with respect to m and k as the $\mathcal{C}_m^k$. $\mathcal{C}_m^k$ and $\mathcal{U}_{m'}^k$ can be dependent, and we will see this in the particular case of DIANA, where $\mathcal{U}_m^k = \mathcal{C}_m^k$.

Finally, we introduce the **third set of stochastic operators**,

$\mathcal{V}^k$, which will be applied to the model updates. For every $k \geq 0$ and $r \in \mathbb{R}^d$,

$$\mathbb{E}\left[\mathcal{V}^k(r)\right] = r \quad \text{and} \quad \mathbb{E}\left[\left\|\mathcal{V}^k(r) - r\right\|^2\right] \leq \nu\|r\|^2, \quad (6)$$

for some constant $\nu \geq 0$. The operators $\mathcal{V}^k$ are mutually independent with respect to k, and independent from all operators $\mathcal{C}_m^{k'}$ and $\mathcal{U}_m^{k'}$.

To analyze MURANA, we need to be more precise than just specifying the **marginal gain** ω. So, we introduce the **average gain** $\boldsymbol{\omega}$, characterized as follows: for every $r_m \in \mathbb{R}^d$, $m \in [M]$, and $k \geq 0$,

$$\mathbb{E}\left[\left\|\tfrac{1}{M}\sum_{m=1}^{M}\left(\mathcal{C}_m^k(r_m) - r_m\right)\right\|^2\right] \leq \tfrac{\boldsymbol{\omega}}{M}\sum_{m=1}^{M}\|r_m\|^2. \quad (7)$$

In general, we have $\boldsymbol{\omega} \leq \omega$. But if the operators $\mathcal{C}_m^k$, for $m \in [M]$, are mutually independent, the variance of the sum is the sum of the variances, and we can set $\boldsymbol{\omega} = \omega/M$. Another case of interest is the sampling setting:

Proposition 1 (Marginal and average gains of sampling) *Let $N \in [M]$. Consider that at every iteration k, a random subset $\Omega^k \subset [M]$ of size N is chosen uniformly at random, and $\mathcal{C}_m^k$ maps r_m to $\{\frac{M}{N}r_m$ if $m \in \Omega^k$, 0 otherwise$\}$ (this is sometimes called N-nice sampling (Richtárik & Takáč, 2016; Gower et al., 2020a)). Then (4) is satisfied with $\omega = \frac{M-N}{N}$ and (7) is satisfied with $\boldsymbol{\omega} = \frac{M-N}{N(M-1)}$ (set to 0 if $M = N = 1$).*

Furthermore, the stochastic operators can be composed, which makes it possible to combine random activation with respect to m and compression of the vectors themselves, for instance:

Proposition 2 (Marginal and average gains of composition) *Let $\mathcal{C}_m$ and $\mathcal{C}'_m$ be stochastic operators such that, for every $m \in [M]$ and $r_m \in \mathbb{R}^d$,*

$$\mathbb{E}[\mathcal{C}_m(r_m)] = r_m,\ \mathbb{E}[\|\mathcal{C}_m(r_m) - r_m\|^2] \leq \omega\|r_m\|^2,$$
$$\mathbb{E}[\mathcal{C}'_m(r_m)] = r_m,\ \mathbb{E}[\|\mathcal{C}'_m(r_m) - r_m\|^2] \leq \omega'\|r_m\|^2,$$
$$\mathbb{E}\left[\left\|\tfrac{1}{M}\sum_{m=1}^{M}\left(\mathcal{C}'_m(r_m) - r_m\right)\right\|^2\right] \leq \tfrac{\boldsymbol{\omega}'}{M}\sum_{m=1}^{M}\|r_m\|^2.$$

for some $\omega \geq 0$, $\omega' \geq 0$, $\boldsymbol{\omega}' \geq 0$. Then for every $m \in [M]$ and $r_m \in \mathbb{R}^d$,

$$\mathbb{E}\left[\mathcal{C}'_m(\mathcal{C}_m(r_m))\right] = r_m,$$
$$\mathbb{E}\left[\|\mathcal{C}'_m(\mathcal{C}_m(r_m)) - r_m\|^2\right] \leq (\omega + \omega' + \omega\omega')\|r_m\|^2.$$

Thus, the marginal gain of $\mathcal{C}'_m \circ \mathcal{C}_m$ is $\omega + \omega' + \omega\omega'$.

If, in addition, the operators $\mathcal{C}_m$, $m \in [M]$, are mutually independent, then for every $r_m \in \mathbb{R}^d$ and $m \in [M]$, we get

$$\mathbb{E}\left[\left\|\tfrac{1}{M}\sum_{m=1}^{M}\left(\mathcal{C}'_m(\mathcal{C}_m(r_m)) - r_m\right)\right\|^2\right] \quad (8)$$
$$\leq \left(\tfrac{\omega}{M} + \boldsymbol{\omega}'(1+\omega)\right)\tfrac{1}{M}\sum_{m=1}^{M}\|r_m\|^2. \quad (9)$$

Thus, the average gain of $\mathcal{C}'_m \circ \mathcal{C}_m$ in that case is $\frac{\omega}{M} + \boldsymbol{\omega}'(1+\omega)$.

2.2. Proposed algorithms: MURANA and MURANA-D

We propose the MUltiple RANdomized Algorithm (MURANA), described in Algorithm 1, as an abstract mathematical algorithm without regard to the execution architecture, or equivalently, as a sequential algorithm. We also explicitly write MURANA as a distributed algorithm in a client-server architecture, with explicit communication steps, as Algorithm 2, and call it MURANA-D.

If $\mathcal{U}_m^k = \mathcal{C}_m^k = \mathcal{V}^k = \mathrm{Id}$, where Id denotes the identity, and $\omega = \chi = \boldsymbol{\omega} = \nu = 0$, MURANA with $\lambda = \rho = 1$ reverts to standard **proximal gradient descent**: $x^{k+1} = \mathrm{prox}_{\gamma R}\left(x^k - \gamma\nabla F(x^k)\right)$. This baseline algorithm evaluates the full gradient $\nabla F(x^k) = \frac{1}{M}\sum_{m=1}^{M}\nabla F_m(x^k)$ at every iteration, which requires M calls to the gradients ∇F_m. If every gradient call has linear complexity $O(d)$, the complexity is $O(Md)$ per iteration, which is typically much too large.

Thus, the **three sources of randomness** in MURANA are typically used as follows: the operators $\mathcal{C}_m^k$ are used to save computation, by using much less than M, possibly even only 1, gradient calls per iteration, and/or decreasing the communication load by compressing the vectors sent by the nodes to the master for aggregation. The operators $\mathcal{U}_m^k$ control the variance-reduction process, during which each variable h_m^k learns the optimal gradient $\nabla F_m(x^\star)$ along the iterations, using the available computed information. In a distributed setting, the operators $\mathcal{V}^k$ are used for compression during broadcasting, in which the server communicates the model estimate to all nodes, at the beginning of every iteration.

When $\mathcal{U}_m^k = \mathcal{C}_m^k$ for every $m \in [M]$ and $k \geq 0$, we recover the recently proposed DIANA (Mishchenko et al., 2019; Horváth et al., 2019) as a particular case of MURANA-D, but generalized here in several ways, see in Sect. 3. In MURANA, we have **more degrees of freedom** than in DIANA: the stochastic gradient $d^{k+1} + h^k$, which is an unbiased estimate of $\nabla F(x^k)$ and is used to update the model x^k, is obtained from the output of the operators $\mathcal{C}_m^k$, whereas the h_m learn the optimal gradients $\nabla F_m(x^\star)$ using the output of the operators $\mathcal{U}_m^k$. We can think of L-SVRG, see below in Sect. 5, which has these two, different and decoupled, mechanisms: the random choice of the activated

Algorithm 1 MURANA (new)

```
1: input: parameters $\gamma > 0$, $\lambda > 0$, $\rho > 0$, initial vectors
   $x^0 \in \mathbb{R}^d$, $h_m^0 \in \mathbb{R}^d$, for $m \in [M]$.
2: $h^0 := \frac{1}{M}\sum_{m=1}^M h_m^0$
3: for $k = 0, 1, \ldots$ do
4:     for $m \in [M]$ do
5:         $d_m^{k+1} := \mathcal{C}_m^k\big(\nabla F_m(x^k) - h_m^k\big)$
6:         $u_m^{k+1} := \mathcal{U}_m^k\big(\nabla F_m(x^k) - h_m^k\big)$
7:         $h_m^{k+1} := h_m^k + \lambda u_m^{k+1}$
8:     end for
9:     $d^{k+1} := \frac{1}{M}\sum_{m=1}^M d_m^{k+1}$
10:    $\tilde{x}^{k+1} := \mathrm{prox}_{\gamma R}\big(x^k - \gamma(h^k + d^{k+1})\big)$
11:    $x^{k+1} := x^k + \rho\mathcal{V}^k(\tilde{x}^{k+1} - x^k)$
12:    $h^{k+1} := h^k + \frac{\lambda}{M}\sum_{m=1}^M u_m^{k+1}$
13: end for
```

gradient at every iteration and the random decision of taking a full gradient pass. Thus, MURANA is a versatile template algorithm, which covers many diverse tools spread across the literature of randomized optimization algorithms in a single umbrella.

2.3. Convergence results

Let $h_m^\star := \nabla F_m(x^\star)$, for $m \in [M]$. We denote by $\kappa := L/\mu_F$ the conditioning of F.

Theorem 1 (Linear convergence of MURANA) *In MURANA, suppose that $0 < \lambda \leq \frac{1}{1+\chi}$ and $0 < \rho \leq \frac{1}{1+\nu}$, and set $\chi' := \frac{1}{\lambda} - 1 \geq \chi$ and $\nu' := \frac{1}{\rho} - 1 \geq \nu$. Choose $B > 1$. If $\mu = 0$, suppose that $0 < \gamma < \frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$; else, suppose that $0 < \gamma \leq \frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$. Set $\eta := 1 - \gamma\big(\frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}\big)^{-1} \in [0,1)$. Define the Lyapunov function, for every $k \geq 0$,*

$$\Psi^k := \|x^k - x^\star\|^2 + (B^2+B)\gamma^2\boldsymbol{\omega}\frac{1+\chi'}{1+\nu'}\frac{1}{M}\sum_{m=1}^M \|h_m^k - h_m^\star\|^2. \tag{10}$$

Then, for every $k \geq 0$, we have, conditionally to x^k, h^k and all h_m^k,

$$\mathbb{E}\left[\Psi^{k+1}\right] \leq c\Psi^k,$$

where

$$c := 1 - \min\left(\frac{2\gamma}{1+\nu'}\big(\eta\mu_F + (1-\eta)\frac{L\mu}{L+\mu}\big), \frac{1-B^{-2}}{1+\chi'}\right) < 1. \tag{11}$$

Therefore, the algorithm converges linearly with rate c, in expectation; in particular, $\mathbb{E}[\|x^k - x^\star\|^2] \leq c^k\Psi^0$, and $\mathbb{E}[\|h_m^k - \nabla F_m(x^\star)\|^2] \leq c^k\Psi^0$, for every $m \in [M]$ and $k \geq 0$.

In the conditions of Theorem 1, the best choice of the parameter γ in MURANA, minimizing c in (11), is

$$\gamma = \frac{1}{1+\max\left(1-\frac{\mu}{\mu_F}\frac{2L}{L+\mu}, 0\right)}\frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}. \tag{12}$$

When μ is unknown, one should do as if it was true that $\mu = 0$, and set $\gamma = \frac{1}{L}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$.

Further, since $\eta\mu_F + (1-\eta)\frac{L\mu}{L+\mu} \geq \frac{\mu}{2}$, we have

$$c \leq 1 - \min\left(\frac{\gamma\mu}{1+\nu'}, \frac{1-B^{-2}}{1+\chi'}\right).$$

However, μ can be 0, so that to guarantee a linear rate depending on the conditioning κ only, whatever $\mu \geq 0$, γ should not be too close to the upper bound of the allowed interval in Theorem 1. In particular, we have:

Corollary 1 *In MURANA, suppose that $\lambda = \frac{1}{1+\chi}$ and $\rho = \frac{1}{1+\nu}$. Choose $B > 1$. Suppose that $0 < \gamma \leq \frac{1}{L}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$. Then, using Ψ^k defined in (10), with $\chi' = \chi$ and $\nu' = \nu$, we have, for every $k \geq 0$, conditionally on x^k, h^k and all h_m^k,*

$$\mathbb{E}\left[\Psi^{k+1}\right] \leq c'\Psi^k, \tag{13}$$

where $c' := 1 - \min\left(\frac{\gamma\mu_F}{1+\nu}, \frac{1-B^{-2}}{1+\chi}\right) < 1$. Therefore, if $\gamma = \Theta(\frac{1}{L}\frac{1}{1+(1+B)^2\boldsymbol{\omega}})$ and B is fixed, the complexity of MURANA to achieve ϵ-accuracy is

$$\mathcal{O}\big((\kappa(1+\boldsymbol{\omega})(1+\nu) + \chi)\log(1/\epsilon)\big)$$

iterations.

In the conditions of Corollary 1, if $\gamma = \frac{1}{L}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$, then

$$c' = 1 - \min\left(\frac{1}{\kappa}\frac{1}{1+\nu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}, \frac{1-B^{-2}}{1+\chi}\right). \tag{14}$$

Algorithm 2 MURANA-D (new)

```
1: input: parameters $\gamma > 0$, $\lambda > 0$, $\rho > 0$, initial vectors
   $x^0 \in \mathbb{R}^d$, $h_m^0 \in \mathbb{R}^d$.
2: $h^0 := \frac{1}{M}\sum_{m=1}^M h_m^0$, $s^0 := 0$, $x^{-1} = x^0$
3: for $k = 0, 1, \ldots$ do
4:     at master: broadcast $s^k$ to all nodes
5:     for $m \in [M]$, at nodes in parallel, do
6:         $x^k := x^{k-1} + \rho s^k$
7:         $d_m^{k+1} := \mathcal{C}_m^k\big(\nabla F_m(x^k) - h_m^k\big)$
8:         $u_m^{k+1} := \mathcal{U}_m^k\big(\nabla F_m(x^k) - h_m^k\big)$
9:         $h_m^{k+1} := h_m^k + \lambda u_m^{k+1}$
10:        convey $d_m^{k+1}$ and $u_m^{k+1}$ to master
11:    end for
12:    at master:
13:    $h^{k+1} := h^k + \frac{\lambda}{M}\sum_{m=1}^M u_m^{k+1}$
14:    $d^{k+1} := \frac{1}{M}\sum_{m=1}^M d_m^{k+1}$
15:    $\tilde{x}^{k+1} := \mathrm{prox}_{\gamma R}\big(x^k - \gamma(h^k + d^{k+1})\big)$
16:    $s^{k+1} := \mathcal{V}^k(\tilde{x}^{k+1} - x^k)$
17:    $x^{k+1} := x^k + \rho s^{k+1}$
18: end for
```

Algorithm 3 DIANA-PP (new)
(reverts to DIANA if $N = M$)

```
1: input: parameters $\gamma > 0$, $\lambda > 0$, $\rho > 0$, participation level $N \in [M]$, initial vectors $x^0 \in \mathbb{R}^d$, $h_m^0 \in \mathbb{R}^d$, for $m \in [M]$.
2: $h^0 := \frac{1}{M}\sum_{m=1}^M h_m^0$, $s^0 := 0$, $x^{-1} = x^0$
3: for $k = 0, 1, \ldots$ do
4:    pick $\Omega^k \subset [M]$ of size $N$ uniformly at random
5:    at master: broadcast $s^k$ to all nodes
6:    for $m \in \Omega_k$, at nodes in parallel, do
7:       $x^k := x^{k-1} + \rho s^k$
8:       $d_m^{k+1} := \mathcal{C}_m^k\big(\nabla F_m(x^k) - h_m^k\big)$
9:       $h_m^{k+1} := h_m^k + \lambda d_m^{k+1}$
10:      convey $d_m^{k+1}$ to master
11:   end for
12:   for $m \notin \Omega_k$, at nodes in parallel, do
13:      $x^k := x^{k-1} + \rho s^k$
14:      $h_m^{k+1} := h_m^k$
15:   end for
16:   at master:
17:   $d^{k+1} := \frac{1}{M}\sum_{m\in\Omega_k} d_m^{k+1}$
18:   $h^{k+1} := h^k + \lambda d^{k+1}$
19:   $\tilde{x}^{k+1} := \mathrm{prox}_{\gamma R}\big(x^k - \gamma(h^k + d^{k+1})\big)$
20:   $s^{k+1} := \mathcal{V}^k(\tilde{x}^{k+1} - x^k)$
21:   $x^{k+1} := x^k + \rho s^{k+1}$
22: end for
```

To balance the two constants $(1+B)^2$ and $1 - B^{-2}$, one can choose $B = \sqrt{20/3} - 1$, which yields

$$c' \leq 1 - \min\left(\frac{1}{\kappa}\frac{1}{1+\nu}\frac{1}{1+(20/3)\boldsymbol{\omega}}, \frac{3}{5}\frac{1}{1+\chi}\right). \tag{15}$$

Another choice is $B = \sqrt{5} - 1$, so that $(1+B)^2 = 5$ and

$$c' \leq 1 - \min\left(\frac{1}{\kappa}\frac{1}{1+\nu}\frac{1}{1+5\boldsymbol{\omega}}, \frac{1}{3}\frac{1}{1+\chi}\right). \tag{16}$$

3. Particular case: DIANA

When $\mathcal{U}_m^k = \mathcal{C}_m^k$, for every $k \geq 0$ and $m \in [M]$, and $\mathcal{V}^k = \mathrm{Id}$, MURANA-D reverts to DIANA, shown as Algorithm 3 (in the case $N = M$, i.e. full participation). DIANA was proposed by Mishchenko et al. (Mishchenko et al., 2019) and generalized (with $R = 0$) in (Horváth et al., 2019). It was then further extended (still with $R = 0$) to the case of compression of the model during broadcast in (Gorbunov et al., 2020), where it is called 'DIANA with bi-directional quantization'; this corresponds to $\mathcal{V}^k \neq \mathrm{Id}$ here, and we still call the algorithm DIANA in this case. But to date, DIANA was studied for independent operators $\mathcal{C}_m^k$ only, and with $\mu = \mu_F > 0$. Even in this case, our following results are more general than existing ones, for instance with a larger range for γ in comparison with Theorem 1 of (Horváth et al., 2019).

Thus, we generalize DIANA to arbitrary operators $\mathcal{C}_m^k$, to the general setting $0 \leq \mu \leq \mu_F$, to the presence of a regularizer R, and to possible randomization, or compression, of the model updates. As a direct application of Theorem 1 with $\chi = \omega$, we have:

Theorem 2 (Linear convergence of DIANA) *In DIANA, suppose that* $0 < \lambda \leq \frac{1}{1+\omega}$ *and* $0 < \rho \leq \frac{1}{1+\nu}$*, and set* $\omega' := \frac{1}{\lambda} - 1 \geq \omega$ *and* $\nu' := \frac{1}{\rho} - 1 \geq \nu$*. Choose* $B > 1$*. If* $\mu = 0$*, suppose that* $0 < \gamma < \frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$*; else, suppose that* $0 < \gamma \leq \frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$*. Set* $\eta := 1 - \gamma\big(\frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}\big)^{-1} \in [0,1)$*. We define the Lyapunov function, for every* $k \geq 0$*,*

$$\Psi^k := \|x^k - x^\star\|^2 + (B^2+B)\gamma^2\boldsymbol{\omega}\frac{1+\omega'}{1+\nu'}\frac{1}{M}\sum_{m=1}^M \|h_m^k - h_m^\star\|^2. \tag{17}$$

Then, for every $k \geq 0$*, we have, conditionally to* x^k*,* h^k *and all* h_m^k*,*

$$\mathbb{E}\left[\Psi^{k+1}\right] \leq c\Psi^k,$$

where

$$c := 1 - \min\left(\frac{2\gamma}{1+\nu'}\big(\eta\mu_F + (1-\eta)\frac{L\mu}{L+\mu}\big), \frac{1-B^{-2}}{1+\omega'}\right). \tag{18}$$

Corollary 2 *In DIANA, suppose that* $\lambda = \frac{1}{1+\omega}$ *and* $\rho = \frac{1}{1+\nu}$*. Choose* $B > 1$*. Suppose that* $0 < \gamma \leq \frac{1}{L}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$*. Then, using* Ψ^k *defined in (17), with* $\omega' = \omega$ *and* $\nu' = \nu$*, we have, for every* $k \geq 0$*, conditionally to* x^k*,* h^k *and all* h_m^k*,*

$$\mathbb{E}\left[\Psi^{k+1}\right] \leq c'\Psi^k, \tag{19}$$

where $c' := 1 - \min\left(\frac{\gamma\mu_F}{1+\nu}, \frac{1-B^{-2}}{1+\omega}\right) < 1$*. Therefore, if* $\gamma = \Theta(\frac{1}{L}\frac{1}{1+(1+B)^2\boldsymbol{\omega}})$ *and* B *is fixed, the complexity of DIANA to achieve* ϵ*-accuracy is*

$$\mathcal{O}\big((\kappa(1+\boldsymbol{\omega})(1+\nu) + \omega)\log(1/\epsilon)\big)$$

iterations.

Partial participation in DIANA. We make use of the possibility of having dependent stochastic operators and we use the composition of operators $\mathcal{C}'^k_m \circ \mathcal{C}_m^k$, like in Proposition 2, with the $\mathcal{C}'^k_m$ being sampling operators like in Proposition 1. This yields DIANA-PP, shown as Algorithm 3. Since DIANA-PP is a particular case of DIANA with such composed operators, we can apply Theorem 2, with ω, the marginal gain of the composed operators here, replaced by $\omega + \frac{M-N}{N}(1+\omega)$ and $\boldsymbol{\omega}$ replaced by $\frac{\omega}{M} + \frac{M-N}{N(M-1)}(1+\omega)$:

Theorem 3 (Linear convergence of DIANA-PP) *In DIANA-PP, suppose that the* $\mathcal{C}_m^k$*,* $m \in [M]$ *are mutually independent and set* $\boldsymbol{\omega} := \frac{\omega}{M} + \frac{M-N}{N(M-1)}(1+\omega)$*. Suppose that* $0 < \lambda \leq \frac{N}{M}\frac{1}{1+\omega}$ *and set* $\omega' := \frac{N}{M\lambda} - 1 \geq \omega$*. Suppose that* $0 < \rho \leq \frac{1}{1+\nu}$ *and set* $\nu' := \frac{1}{\rho} - 1 \geq \nu$*. Choose* $B > 1$*.*

If $\mu = 0$, suppose that $0 < \gamma < \frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$; else, suppose that $0 < \gamma \leq \frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$. Set $\eta := 1 - \gamma\left(\frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}\right)^{-1} \in [0,1)$. We define the Lyapunov function, for every $k \geq 0$,

$$\Psi^k := \|x^k - x^\star\|^2 + (B^2+B)\gamma^2\boldsymbol{\omega}\frac{1+\omega'}{1+\nu'}\frac{1}{N}\sum_{m=1}^M \|h_m^k - h_m^\star\|^2. \quad (20)$$

Then, for every $k \geq 0$, we have, conditionally to x^k, h^k and all h_m^k,

$$\mathbb{E}\left[\Psi^{k+1}\right] \leq c\Psi^k,$$

where

$$c := 1 - \min\left(\frac{2\gamma}{1+\nu'}\left(\eta\mu_F + (1-\eta)\frac{L\mu}{L+\mu}\right), \frac{1-B^{-2}}{1+\omega'}\right). \quad (21)$$

Corollary 3 *In DIANA-PP, suppose that the $\mathcal{C}_m^k$, $m \in [M]$ are mutually independent and set $\boldsymbol{\omega} := \frac{\omega}{M} + \frac{M-N}{N(M-1)}(1+\omega)$. Suppose that $\lambda = \frac{N}{M}\frac{1}{1+\omega}$ and $\rho = \frac{1}{1+\nu}$. Choose $B > 1$. Suppose that $0 < \gamma \leq \frac{1}{L}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$. Then, using Ψ^k defined in (20), with $\omega' = \omega$ and $\nu' = \nu$, we have, for every $k \geq 0$, conditionally to x^k, h^k and all h_m^k,*

$$\mathbb{E}[\Psi^{k+1}] \leq c'\Psi^k,$$

where

$$c' := 1 - \min\left(\frac{\gamma\mu_F}{1+\nu}, \frac{N}{M}\frac{1-B^{-2}}{1+\omega}\right).$$

Therefore, if $\gamma = \Theta(\frac{1}{L}\frac{1}{1+(1+B)^2\boldsymbol{\omega}})$ and B is fixed, the asymptotic complexity of DIANA to achieve ϵ-accuracy is

$$\mathcal{O}\Big(\Big(\kappa\big(1+\tfrac{\omega}{M}+\tfrac{M-N}{NM}(1+\omega)\big)(1+\nu)+\tfrac{M}{N}(1+\omega)\Big)\log(1/\epsilon)\Big)$$

iterations.

To summarize, DIANA is the particular case of DIANA-PP with full participation, i.e. $N = M$. Its convergence with general, possibly dependent, operators $\mathcal{C}_m^k$, is established in Theorem 2 and Corollary 2. DIANA-PP is more general than DIANA, since it allows for partial participation, but its convergence is established in Theorem 3 and Corollary 3 only when the operators $\mathcal{C}_m^k$, $m \in [M]$, are mutually independent.

4. Particular case: SAGA

When $\mathcal{U}_m^k = \mathcal{C}_m^k$, for every $k \geq 0$ and $m \in [M]$, and these operators are set as dependent sampling operators like in Proposition 1, and $\mathcal{V}^k = \mathrm{Id}$, MURANA becomes Minibatch-SAGA, shown as Algorithm 4. We have $1 + \omega = \frac{M}{N}$, $\boldsymbol{\omega} = \frac{M-N}{N(M-1)}$, and we set $\lambda = \frac{1}{1+\omega} = \frac{N}{M}$ and $\rho = 1$. Minibatch-SAGA is SAGA (Defazio et al., 2014) if $N = 1$ and proximal gradient descent if $N = M$, so Minibatch-SAGA interpolates between these two regimes for $1 < N < M$. This algorithm was called 'minibatch SAGA with τ-nice sampling' in (Gower et al., 2020a), with their τ being our N, but studied only with $R = 0$. It was called 'q-SAGA' in (Hofmann et al., 2015) with their q being our N, but studied only with $\mu_F = \mu$. Thus, our convergence results Theorem 4 and Corollary 4, deferred to the long version of the paper by lack of space, are new, to the best of our knowledge.

Algorithm 4 Minibatch-SAGA
(reverts to SAGA if $N = 1$)

```
1: input: parameter $\gamma > 0$, sampling size $N \in [M]$, initial vectors $x^0 \in \mathbb{R}^d$, $h_m^0 \in \mathbb{R}^d$, for $m \in [M]$.
2: $h^0 := \frac{1}{M}\sum_{m=1}^M h_m^0$
3: for $k = 0, 1, \ldots$ do
4:     pick $\Omega^k \subset [M]$ of size $N$ uniformly at random
5:     for $m \in \Omega_k$ do
6:         $h_m^{k+1} := \nabla F_m(x^k)$
7:     end for
8:     for $m \in [M]\backslash\Omega_k$ do
9:         $h_m^{k+1} := h_m^k$
10:    end for
11:    $d^{k+1} := \frac{1}{N}\sum_{m\in\Omega^k}(h_m^{k+1} - h_m^k)$
12:    $x^{k+1} := \mathrm{prox}_{\gamma R}\left(x^k - \gamma(h^k + d^{k+1})\right)$
13:    $h^{k+1} := h^k + \frac{N}{M}d^{k+1}$
14: end for
```

5. Particular case: L-SVRG

Like SAGA, SVRG (Johnson & Zhang, 2013; Zhang et al., 2013) (sometimes called prox-SVRG (Xiao & Zhang, 2014) if $R \neq 0$) is a variance-reduced (Gower et al., 2020b) randomized algorithm, well suited to solve (1), since it can be up to M times faster than proximal gradient descent. Recently, the loopless-SVRG (L-SVRG) algorithm was proposed (Kovalev et al., 2020), which is similar to SVRG, but with the outer loop of epochs replaced by a coin flip performed in each iteration, designed to trigger with a small probability, e.g. $1/M$, the computation of the full gradient of F. In comparison with SVRG, the analysis of L-SVRG is simpler and L-SVRG is more flexible; for instance, there is no need to know μ_F to achieve the $\mathcal{O}\big((\kappa + M)\log(1/\epsilon)\big)$ complexity. In SVRG and L-SVRG, in addition to the full gradient passes computed once in a while, one gradient is computed at every iteration. A minibatch version of L-SVRG, with N instead of 1 gradients picked at every iteration, was called 'L-SVRG with τ-nice sampling' in (Qian et al., 2019); we call it Minibatch-L-SVRG, shown as Algorithm 5.

Minibatch-L-SVRG is a particular case of MURANA, with the $\mathcal{C}_m^k$, $m \in [M]$, set as dependent sampling operators like in Proposition 1, and $\mathcal{V}^k = \mathrm{Id}$, $\rho = 1$. Thus, like for

Algorithm 5 Minibatch-L-SVRG
(reverts to L-SVRG if $N = 1$)

```
1: input: parameter $\gamma > 0$, sampling size $N \in [M]$, proba. $p \in (0,1]$, initial $x^0 \in \mathbb{R}^d$.
2: $h^0 := \frac{1}{M}\sum_{m=1}^M \nabla F_m(x^0)$, $y^0 := x^0$
3: for $k = 0, 1, \ldots$ do
4:     pick $\Omega^k \subset [M]$ of size $N$ uniformly at random
5:     $d^{k+1} := \frac{1}{N}\sum_{m\in\Omega^k} \nabla F_m(x^k) - \nabla F_m(y^k)$
6:     $x^{k+1} := \mathrm{prox}_{\gamma R}\left(x^k - \gamma(h^k + d^{k+1})\right)$
7:     Pick randomly $a^k := \{1$ with probability $p$,
8:         $0$ with probability $1-p\}$
9:     if $a^k = 1$ then
10:        $h^{k+1} := \frac{1}{M}\sum_{m=1}^M \nabla F_m(x^k)$,
11:        $y^{k+1} := x^k$
12:    else
13:        $h^{k+1} := h^k$, $y^{k+1} := y^k$
14:    end if
15: end for
```

Algorithm 6 ELVIRA (new)

```
1: input: parameter $\gamma > 0$, sampling size $N \in [M]$, proba. $p \in (0,1]$, initial $x^0 \in \mathbb{R}^d$.
2: $h^0 := \frac{1}{M}\sum_{m=1}^M \nabla F_m(x^0)$, $y^0 := x^0$
3: for $k = 0, 1, \ldots$ do
4:     Pick randomly $a^k := \{1$ with probability $p$,
5:         $0$ with probability $1-p\}$
6:     if $a^k = 1$ then
7:         $h^{k+1} := \frac{1}{M}\sum_{m=1}^M \nabla F_m(x^k)$
8:         $x^{k+1} := \mathrm{prox}_{\gamma R}\left(x^k - \gamma h^{k+1}\right)$
9:         $y^{k+1} := x^k$
10:    else
11:        pick $\Omega^k \subset [M]$ of size $N$ uniformly at random
12:        $d^{k+1} := \frac{1}{N}\sum_{m\in\Omega^k}\left(\nabla F_m(x^k) - \nabla F_m(y^k)\right)$
13:        $x^{k+1} := \mathrm{prox}_{\gamma R}\left(x^k - \gamma(h^k + d^{k+1})\right)$
14:        $h^{k+1} := h^k$, $y^{k+1} := y^k$
15:    end if
16: end for
```

Minibatch-SAGA, we have $1+\omega = \frac{M}{N}$ and $\boldsymbol{\omega} = \frac{M-N}{N(M-1)}$. Let $p \in (0,1]$. The $\mathcal{U}_m^k$ are all the same random operator $\mathcal{U}^k$, which maps x to $\{\frac{1}{p}x$ with probability p, 0 otherwise$\}$. We have $\chi = \frac{1-p}{p}$ and we set $\lambda = \frac{1}{1+\chi} = p$. Hence, by applying Theorem 1, we get:

Theorem 5 (Linear convergence of Minibtach-L-SVRG) *Set $\boldsymbol{\omega} = \frac{M-N}{N(M-1)}$ and choose $B > 1$. In Minibtach-L-SVRG, if $\mu = 0$, suppose that $0 < \gamma < \frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$; else, suppose that $0 < \gamma \leq \frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$. Set $\eta := 1 - \gamma\left(\frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}\right)^{-1} \in [0,1)$. We define the Lyapunov function, for every $k \geq 0$,*

$$\Psi^k := \|x^k - x^\star\|^2 + (B^2+B)\gamma^2\boldsymbol{\omega}\frac{1}{pM}\sum_{m=1}^M \|h_m^k - h_m^\star\|^2. \tag{22}$$

Then, for every $k \geq 0$, we have, conditionally to x^k, h^k and all h_m^k,

$$\mathbb{E}\left[\Psi^{k+1}\right] \leq c\Psi^k,$$

where

$$c := 1 - \min\left(2\gamma\left(\eta\mu_F + (1-\eta)\frac{L\mu}{L+\mu}\right), p(1-B^{-2})\right).$$

Corollary 5 *Set $\boldsymbol{\omega} = \frac{M-N}{N(M-1)}$ and choose $B > 1$. In Minibatch-L-SVRG, suppose that $0 < \gamma \leq \frac{1}{L}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$. Then, using Ψ^k defined in (22), we have, for every $k \geq 0$, conditionally to x^k, h^k and all h_m^k,*

$$\mathbb{E}\left[\Psi^{k+1}\right] \leq c'\Psi^k,$$

where

$$c' := 1 - \min\left(\gamma\mu_F, p(1-B^{-2})\right).$$

For instance, with $N = 1$, $B = \sqrt{5} - 1$ (we have seen this choice before), and $\gamma = \frac{1}{6L}$, we have $c' \leq 1 - \min(\frac{1}{6\kappa}, \frac{p}{3})$; this is very similar to the rate $1 - \min(\frac{1}{6\kappa}, \frac{p}{2})$ given in Theorem 5 of (Kovalev et al., 2020).

Therefore, if $\gamma = \Theta(\frac{1}{L})$, the asymptotic complexity of Minibatch-L-SVRG to achieve ϵ-accuracy is $\mathcal{O}\big((\kappa + \frac{1}{p})\log(1/\epsilon)\big)$ iterations and $\mathcal{O}\big((N\kappa + pM\kappa + \frac{N}{p} + M)\log(1/\epsilon)\big)$ gradient calls, since there are $2N + pM$ gradient calls per iteration in expectation. This is the same as Minibatch-SAGA if $p = \Theta(\frac{N}{M})$.

6. Particular case: ELVIRA (new)

It is a pity not to use the full gradient in L-SVRG to update x^k, when it is computed. And even with $p = 1$, which means the full gradient computed at every iteration, L-SVRG does not revert to proximal gradient descent. We correct these drawbacks by proposing a new algorithm, called ELVIRA, which improves upon L-SVRG; it is shown as Algorithm 6. The novelty is that whenever a full gradient pass is computed, it is used just after to update the estimate x^{k+1} of the solution.

ELVIRA is a particular case of MURANA as follows: $\mathcal{V}^k = \mathrm{Id}$, $\rho = 1$, and the $\mathcal{U}_m^k$ are set like in Minibatch-L-SVRG. The $\mathcal{C}_m^k$ depend on the $\mathcal{U}_m^k$ and are set as follows: if the full gradient is not computed, $\mathcal{C}_m^k$ are sampling operators like in Proposition 1, Minibatch-L-SVRG and Minibatch-SAGA. Otherwise, the $\mathcal{C}_m^k$ are set to the identity. We have $\chi = \frac{1-p}{p}$ and we set $\lambda = \frac{1}{1+\chi} = p$. Moreover, $\boldsymbol{\omega} = \frac{M-N}{N(M-1)}(1-p)$. For instance, if $N = 1$ and $p = \frac{1}{M}$, we have $\boldsymbol{\omega} = \frac{M-1}{M}$, instead of $\boldsymbol{\omega} = 1$ with L-SVRG. Hence, by applying Theorem 1, we get:

Theorem 6 (Linear convergence of ELVIRA) *Set* $\boldsymbol{\omega} = \frac{M-N}{N(M-1)}(1-p)$ *and choose* $B > 1$. *In ELVIRA, if* $\mu = 0$, *suppose that* $0 < \gamma < \frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$; *else, suppose that* $0 < \gamma \leq \frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$. *Set* $\eta := 1 - \gamma\left(\frac{2}{L+\mu}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}\right)^{-1} \in [0,1)$. *We define the Lyapunov function, for every* $k \geq 0$,

$$\Psi^k := \|x^k - x^\star\|^2 + (B^2+B)\gamma^2\boldsymbol{\omega}\frac{1}{pM}\sum_{m=1}^M \|h_m^k - h_m^\star\|^2. \tag{23}$$

Then, for every $k \geq 0$, *we have, conditionally to* x^k, h^k *and all* h_m^k,

$$\mathbb{E}\left[\Psi^{k+1}\right] \leq c\Psi^k,$$

where

$$c := 1 - \min\Big(2\gamma\big(\eta\mu_F + (1-\eta)\frac{L\mu}{L+\mu}\big), p(1-B^{-2})\Big).$$

Corollary 6 *Set* $\boldsymbol{\omega} = \frac{M-N}{N(M-1)}(1-p)$ *and choose* $B > 1$. *In ELVIRA, suppose that* $0 < \gamma \leq \frac{1}{L}\frac{1}{1+(1+B)^2\boldsymbol{\omega}}$. *Then, using* Ψ^k *defined in* (23), *we have, for every* $k \geq 0$, *conditionally to* x^k, h^k *and all* h_m^k,

$$\mathbb{E}\left[\Psi^{k+1}\right] \leq c'\Psi^k, \tag{24}$$

where

$$c' := 1 - \min\Big(\gamma\mu_F, p(1-B^{-2})\Big).$$

For instance, with $N = 1$, $B = \sqrt{5} - 1$ (we have seen this choice before), and $\gamma = \frac{1}{6L}$, we have $c' \leq 1 - \min(\frac{1}{6\kappa}, \frac{p}{3})$, which is the same result as above for Minibatch-L-SVRG.

Therefore, if $\gamma = \Theta(\frac{1}{L})$, the complexity of ELVIRA is $\mathcal{O}\big((\kappa + \frac{1}{p})\log(1/\epsilon)\big)$ iterations and $\mathcal{O}\big((N\kappa + pM\kappa + \frac{N}{p} + M)\log(1/\epsilon)\big)$ gradient calls, since there are $2N(1-p)+pM$ gradient calls per iteration in expectation. If in addition $p = \Theta(\frac{N}{M})$, the complexity becomes $\mathcal{O}\big((\kappa + \frac{M}{N})\log(1/\epsilon)\big)$ iterations and $\mathcal{O}\big((N\kappa + M)\log(1/\epsilon)\big)$ gradient calls. So, the complexity of ELVIRA is the same as that of Minibatch-L-SVRG; but in practice, one can expect ELVIRA to be faster, because its variance is strictly lower. And it has the same low-memory requirements. ELVIRA reverts to proximal gradient descent in 3 cases: (i) if $p = 1$, (ii) if $N = M$, (iii) if $M = 1$.

References

Albasyoni, A., Safaryan, M., Condat, L., and Richtárik, P. Optimal gradient compression for distributed and federated learning. Preprint arXiv:2010.03246, 2020.

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Proc. of 31st Conf. Neural Information Processing Systems (NIPS)*, pp. 1709–1720, 2017.

Basu, D., Data, D., Karakus, C., and Diggavi, S. N. Qsparse-Local-SGD: Distributed SGD With Quantization, Sparsification, and Local Computations. *IEEE Journal on Selected Areas in Information Theory*, 1(1):217–226, 2020.

Bauschke, H. H. and Combettes, P. L. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, New York, 2nd edition, 2017.

Bubeck, S. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8: 231–357, 2014.

Condat, L., Kitahara, D., Contreras, A., and Hirabayashi, A. Proximal splitting algorithms: Relax them all! preprint arXiv:1912.00137, 2019.

Defazio, A., Bach, F., and Lacoste-Julien, S. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Proc. of 28th Conf. Neural Information Processing Systems (NIPS)*, pp. 1646–1654, 2014.

Dutta, A., Bergou, E. H., Abdelmoniem, A. M., Ho, C. Y., Sahu, A. N., Canini, M., and Kalnis, P. On the discrepancy between the theoretical analysis and practical implementations of compressed communication for distributed deep learning. In *Proc. of AAAI Conf. Artificial Intelligence*, pp. 3817–3824, 2020.

Gorbunov, E., Kovalev, D., Makarenko, D., and Richtárik, P. Linearly converging error compensated SGD. arXiv:2010.12292, 2020.

Gower, R. M., Richtárik, P., and Bach, F. Stochastic quasi-gradient methods: Variance reduction via Jacobian sketching. *Math.Program.*, 2020a.

Gower, R. M., Schmidt, M., Bach, F., and Richtárik, P. Variance-reduced methods for machine learning. *Proc. of the IEEE*, 108(11):1968–1983, nov 2020b.

Hofmann, T., Lucchi, A., Lacoste-Julien, S., and McWilliams, B. Variance reduced stochastic gradient descent with neighbors. In *Proc. of 29th Conf. Neural Information Processing Systems (NIPS)*, pp. 1509–1519, 2015.

Horváth, S., Kovalev, D., Mishchenko, K., Stich, S., and Richtárik, P. Stochastic distributed learning with gradient quantization and variance reduction. preprint arXiv:1904.05115, 2019.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Proc. of 27th Conf. Neural Information Processing Systems (NIPS)*, pp. 315–323, 2013.

Kairouz, P. et al. Advances and open problems in federated learning. Preprint arXiv:1912.04977, 2019.

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. Paper arXiv:1610.05492, presented at the NIPS Workshop on Private Multi-Party Machine Learning, 2016.

Kovalev, D., Horváth, S., and Richtárik, P. Don't jump through hoops and remove those loops: SVRG and Katyusha are better without the outer loop. In *Proc. of 31st Int. Conf. Algorithmic Learning Theory (ALT)*, volume PMLR 117, pp. 451–467, 2020.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 3(37):50–60, 2020.

Liu, X., Li, Y., Tang, J., and Yan, M. A double residual compression algorithm for efficient distributed learning. In *Proc. of 23rd Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, volume PMLR 108, pp. 133–143, 2020.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Agüera y Arcas, B. Communication-efficient learning of deep networks from decentralized data. In *Proc. of 20th Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2017.

Mishchenko, K., Gorbunov, E., Takáč, M., and Richtárik, P. Distributed learning with compressed gradient differences. preprint arXiv:1901.09269, 2019.

Parikh, N. and Boyd, S. Proximal algorithms. *Foundations and Trends in Optimization*, 3(1):127–239, 2014.

Philippenko, C. and Dieuleveut, A. Bidirectional compression in heterogeneous settings for distributed or federated learning with partial participation: tight convergence guarantees. preprint arXiv:2006.14591, 2020.

Qian, X., Qu, Z., and Richtárik, P. L-SVRG and L-Katyusha with arbitrary sampling. arXiv:1906.01481, 2019.

Richtárik, P. and Takáč, M. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156:433–484, 2016.

Sattler, F., Wiedemann, S., K.-R. Müller, and Samek, W. Robust and communication-efficient federated learning from non-i.i.d. data. *IEEE Trans. Neural Networks and Learning Systems*, 31(9):3400–3413, 2020.

Tang, H., Yu, C., Lian, X., Zhang, T., and Liu, J. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In *Proc. of Int. Conf. Machine Learning (ICML)*, pp. 6155–6165, 2019.

Wangni, J., Wang, J., Liu, J., and Zhang, T. Gradient sparsification for communication-efficient distributed optimization. In *Proc. of 32nd Conf. Neural Information Processing Systems (NeurIPS)*, pp. 1306–1316, 2018.

Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. TernGrad: Ternary gradients to reduce communication in distributed deep learning. In *Proc. of 31st Conf. Neural Information Processing Systems (NIPS)*, pp. 1509–1519, 2017.

Xiao, L. and Zhang, T. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Xu, H., Ho, C.-Y., Abdelmoniem, A. M., Dutta, A., Bergou, E. H., Karatsenidis, K., Canini, M., and Kalnis, P. GRACE: A compressed communication framework for distributed machine learning. In *Proc. of 41st IEEE Int. Conf. Distributed Computing Systems (ICDCS)*, 2021.

Zhang, L., Mahdavi, M., and Jin, R. Linear convergence with condition number independent access of full gradients. In *Proc. of 27th Conf. Neural Information Processing Systems (NIPS)*, 2013.