Optimal Mini-Batch and Step Sizes for SAGA

Nidham Gazagnadou [1] **Robert M. Gower** [1] **Joseph Salmon** [2]

Abstract

Recently it has been shown that the step sizes of a family of variance reduced gradient methods called the JacSketch methods depend on the expected smoothness constant. In particular, if this expected smoothness constant could be calculated a priori, then one could safely set much larger step sizes which would result in a much faster convergence rate. We fill in this gap, and provide simple closed form expressions for the expected smoothness constant and careful numerical experiments verifying these bounds. Using these bounds, and since the SAGA algorithm is part of this JacSketch family, we suggest a new standard practice for setting the step and mini-batch sizes for SAGA that are competitive with a numerical grid search. Furthermore, we can now show that the total complexity of the SAGA algorithm decreases linearly in the mini-batch size up to a pre-defined value: the optimal mini-batch size. This is a rare result in the stochastic variance reduced literature, only previously shown for the Katyusha algorithm. Finally we conjecture that this is the case for many other stochastic variance reduced methods and that our bounds and analysis of the expected smoothness constant is key to extending these results.

1. Introduction

Consider the empirical risk minimization (ERM) problem:

$$w^* \in \arg\min_{w\in\mathbb{R}^d} \left(f(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w) \right) , \qquad (1)$$

where each f_i is L_i-smooth and f is μ-strongly convex. Each f_i represents a regularized loss over a sampled data point. Solving the ERM problem is often time consuming for large number of samples n, so much so that algorithms scanning through all the data points at each iteration are not competitive. Gradient descent (GD) falls into this category, and in practice its stochastic version is preferred.

Stochastic gradient descent (SGD), on the other hand, allows to solve the ERM incrementally by computing at each iteration an unbiased estimate of the full gradient, $\nabla f_i(w^k)$ for i randomly sampled in $[n] := \{1, 2, \dots, n\}$ (Robbins & Monro, 1951). On the downside, for SGD to converge one needs to tune a sequence of asymptotically vanishing step sizes, a cumbersome and time-consuming task for the user. Recent works have taken advantage of the sum structure in Eq. (1) to design stochastic variance reduced gradient algorithms (Johnson & Zhang, 2013; Shalev-Shwartz & Zhang, 2013; Defazio et al., 2014; Schmidt et al., 2017). In the strongly convex setting, these methods lead to fast linear convergence instead of the slow $\mathcal{O}(1/t)$ rate of SGD. Moreover, they only require a constant step size, informed by theory, instead of sequence of decreasing step sizes.

In practice, most variance reduced methods rely on a mini-batching strategy for better performance. Yet most convergence analysis (with the Katyusha algorithm of Allen-Zhu (2017) being an exception) indicates that a mini-batch size of $b = 1$ gives the best overall complexity, disagreeing with practical findings, where larger mini-batch often gives better results. Here, we show both theoretically and numerically that $b = 1$ is not the optimal mini-batch size for the SAGA algorithm (Defazio et al., 2014).

Our analysis leverages recent results in (Gower et al., 2018), where the authors prove that the iteration complexity and the step size of SAGA, and a larger family of methods called the JacSketch methods, depend on an expected smoothness constant. This constant governs the trade-off between the increased cost of an iteration as the mini-batch size is increased, and the decreased total complexity. Thus if this expected smoothness constant could be calculated a priori, then we could set the optimal mini-batch size and step size. We provide simple formulas for computing the expected smoothness constant when sampling mini-batches without replacement, and use them to calculate optimal mini-batches and significantly larger step sizes for SAGA.

In particular, we provide two bounds on the expected

[1]LTCI, Télécom ParisTech, Université Paris-Saclay, Paris, France [2]IMAG, Univ Montpellier, CNRS, Montpellier, France. Correspondence to: Nidham Gazagnadou <nidham.gazagnadou@telecom-paristech.fr>, Robert M. Gower <robert.gower@telecom-paristech.fr>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

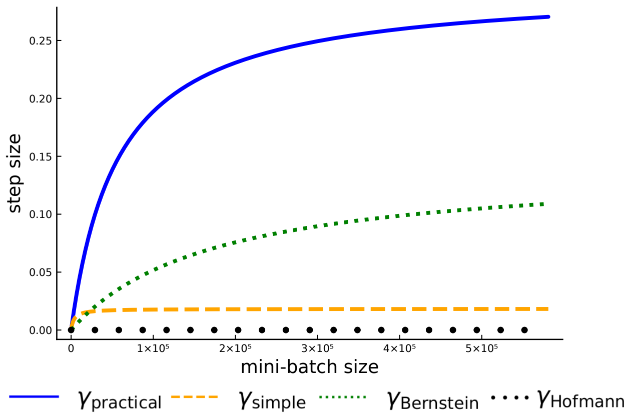

Figure 1: Step size as a function of the mini-batch size for a regularized ($\lambda = 10^{-3}$) logistic regression problem applied to the feature-scaled *covtype.binary* dataset from LIBSVM.

smoothness constant, each resulting in a particular step size formula. We first derive the *simple bound* and then develop a matrix concentration inquality to obtain the refined *Bernstein bound*. We also provide substantial theoretical motivation and numerical evidence for practical estimate of the expected smoothness constant. For illustration, we plot in Figure 1 the evolution of each resulting step size as the mini-batch size grows on a classification problem (Section 5 has more details on our experimental settings).

Furthermore, our bounds provide new insight into the *total complexity*, denoted K_{total} hereafter, of SAGA. For example, when using our *simple bound* we show for regularized generalized linear models (GLM), with $\lambda > 0$ as in Eq. (10), that K_{total} is piecewise linear in the mini-batch size b:

$$K_{\text{total}}(b) = \max\left\{n\frac{b-1}{n-1}\frac{4\bar{L}}{\mu} + \frac{n-b}{n-1}\frac{4L_{\max}}{\mu} + \frac{4b\lambda}{\mu},\right.$$
$$\left. n + \frac{n-b}{n-1}\frac{4(L_{\max}+\lambda)}{\mu}\right\}\log\left(\frac{1}{\epsilon}\right) \quad ,$$

with $L_{\max} := \max_{i\in[n]} L_i$, $\bar{L} := \frac{1}{n}\sum_{i=1}^n L_i$ and $\epsilon > 0$ is the desired precision. This complexity bound, and others presented in Section 3.3 show that SAGA enjoys a linear speedup as we increase the mini-batch size until an optimal one (as illustrated in Figure 2). After this point, the total complexity increases. We use this observation to develop optimal and practical mini-batch sizes and step sizes.

The rest of the paper is structured as follows. In Section 2 we first introduce variance reduction techniques after presenting our main assumption, the expected smoothnes assumption. We highlight how this assumption is necessary to capture the improvement in iteration complexity, and conclude the section by showing that to calculate the expected smoothness constant we need evaluate an intractable expectation. Which brings us to Section 3 where we directly address

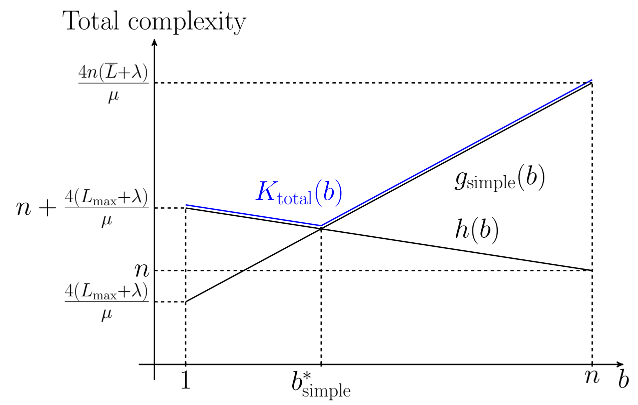
Total complexity
$\frac{4n(\bar{L}+\lambda)}{\mu}$
$n + \frac{4(L_{\max}+\lambda)}{\mu}$
n
$\frac{4(L_{\max}+\lambda)}{\mu}$
$K_{\text{total}}(b)$
$g_{\text{simple}}(b)$
$h(b)$
1
b^*_{simple}
n
b

Figure 2: Optimal mini-batch size b^*_{simple} for the *simple bound*, where $K_{\text{total}}(b) = \max\{g_{\text{simple}}(b), h(b)\}$.

this issue and provide several tractable upper-bounds of the expected smoothness constant. We then calculate optimal mini-batch sizes and step sizes by using our new bounds. Finally, we give numerical experiments in Section 5 that verify our theory on artificial and real datasets. We also show how these new settings for the mini-batch size and step size lead to practical performance gains.

2. Background

2.1. Controlled stochastic reformulation and JacSketch

We can introduce variance reduced versions of SGD in a principled manner by using a *sampling vector*.

Definition 1. We say that a random vector $v \in \mathbb{R}^n$ with distribution $\mathcal{D}$ is a sampling vector if

$$\mathbb{E}_{\mathcal{D}}[v_i] = 1 \quad , \quad \text{for all } i \in [n] \quad .$$

With a sampling vector we can re-write (1) through the following stochastic reformulation

$$w^* = \arg\min_{w\in\mathbb{R}^d} \mathbb{E}_{\mathcal{D}}\left[f_v(w) := \frac{1}{n}\sum_{i=1}^n f_i(w)\cdot v_i\right] , \qquad (2)$$

where $f_v(w)$ is called a *subsampled function*. The stochastic Problem (2) and our original Problem (1) are equivalent :

$$\mathbb{E}_{\mathcal{D}}[f_v(w)] = \frac{1}{n}\sum_{i=1}^n f_i(w)\cdot\mathbb{E}_{\mathcal{D}}[v_i] \overset{\text{Definition 1}}{=} \frac{1}{n}\sum_{i=1}^n f_i(w).$$

Consequently the gradient $\nabla f_v(w)$ is an unbiased estimate of $\nabla f(w)$ and we could use the SGD method to solve (2). To tackle the variance of these stochastic gradients we can further modify (2) by introducing *control variates* which leads to the following *controlled stochastic reformulation*:

$$w^* \in \arg\min_{w\in\mathbb{R}^d} \mathbb{E}_{\mathcal{D}}\left[f_v(w) - z_v(w) + \mathbb{E}_{\mathcal{D}}[z_v(w)]\right] \quad , \quad (3)$$

where $z_v(w) \in \mathbb{R}$ are the control variates. Clearly (3) is also equivalent to (1) since $-z_v(w) + \mathbb{E}_{\mathcal{D}}[z_v(w)]$ has zero expectation. Thus, we can solve (3) using an SGD algorithm where the stochastic gradients are given by

$$g_v(w) := \nabla f_v(w) - \nabla z_v(w) + \mathbb{E}_{\mathcal{D}}[\nabla z_v(w)] \quad . \tag{4}$$

That is, starting from a vector w^0, given a positive step size γ, we can iterate the steps

$$w^{k+1} = w^k - \gamma g_{v^k}(w^k) \quad , \tag{5}$$

where $v^k \sim \mathcal{D}$ are i.i.d. samples at each iteration.

The JacSketch algorithm introduced by Gower et al. (2018) fits this format (5) and uses a linear control $z_v(w) = \frac{1}{n}\langle J^\top w, v\rangle$, where J is a $d \times n$ matrix of parameters. This matrix is updated at each iteration so as to decrease the variance of the resulting stochastic gradients. Carefully updating the covariates through J results in a method that has stochastic gradients with decreasing variance, *i.e.,* $\lim_{w^k \to w^*} \mathbb{E}\left[\|g_{v^k}(w^k) - \nabla f(w^k)\|_2^2\right] = 0$, which is why JacSketch is a stochastic variance reduced algorithm. This is also why the user can set a single constant step size a priori instead of tuning a sequence of decreasing ones. The SAGA algorithm, and all of its mini-batching variants, are instances of the JacSketch method.

2.2. The expected smoothness constant

In order to analyze stochastic variance reduced methods, some form of smoothness assumption needs to be made. The most common assumption is

$$\|\nabla f_i(w) - \nabla f_i(y)\| \leq L_{\max} \|w - y\| \,, \tag{6}$$

for each $i \in [n]$. That is each f_i is uniformly smooth with smoothness constant $L_{\max}$, as is assumed in (Defazio et al., 2014; Hofmann et al., 2015; Raj & Stich, 2018) for variants of SAGA[1]. In the analyses of these papers it was shown that the iteration complexity of SAGA is proportional to $L_{\max}$, and the step size is inversely proportional to $L_{\max}$.

But as was shown in (Gower et al., 2018), we can set a much larger step size by making use of the smoothness of the subsampled functions f_v. For this Gower et al. (2018) introduced the notion of expected smoothness, which we extend here to all sampling vectors and control variates.

Definition 2 (Expected smoothness constant)**.** Consider a sampling vector v with distribution $\mathcal{D}$. We say that the expected smoothness assumption holds with constant $\mathcal{L}$ if for every $w \in \mathbb{R}^d$ we have that

$$\mathbb{E}_{\mathcal{D}}\left[\|\nabla f_v(w) - \nabla f_v(w^*)\|_2^2\right] \leq 2\mathcal{L}(f(w) - f(w^*)) \quad . \tag{7}$$

[1]The same assumption is made in proofs of SVRG (Johnson & Zhang, 2013), S2GD (Konečný & Richtárik, 2017) and the SARAH algorithm (Nguyen et al., 2017).

Remark 1. Note that we refer to any positive constant $\mathcal{L}$ that satisfies (7) as an expected smoothness constant. Indeed $\mathcal{L} \to \infty$ is a valid constant in the extended reals, but as we will see, the smaller $\mathcal{L}$, the better for our complexity results.

Gower et al. (2018) show that the expected smoothness constant plays the same role that $L_{\max}$ does in the previously existing analysis of SAGA, namely that the step size is inversely proportional to $\mathcal{L}$ and the iteration complexity is proportional to $\mathcal{L}$ (see details in Theorem 1). Furthermore, by assuming that f is L–smooth, the expected smoothness constant is bounded

$$L \quad \leq \quad \mathcal{L} \quad \leq \quad L_{\max} \quad , \tag{8}$$

as was proven in Theorem 4.17 in (Gower et al., 2018). Also, the bounds $L_{\max}$ and L are attained when using a uniform single element sampling and a full batch, respectively. And as we will show, the constants $L_{\max}$ and L can be orders of magnitude apart on large dimensional problems. Thus we could set much larger step sizes for larger mini-batch sizes if we could calculate $\mathcal{L}$. Though calculating $\mathcal{L}$ is not easy, as we see in the next lemma.

Lemma 1. *Let v be an unbiased sampling vector. Suppose that $f_v(w) = \frac{1}{n}\sum_{i=1}^n f_i(w) v_i$ is L_v-smooth and each f_i is convex for $i = 1, \ldots, n$. It follows that the expected smoothness constant holds with $\mathcal{L} = \max_{i \in [n]} \mathbb{E}[L_v v_i]$.*

Proof. The proof is given in Appendix A.1. □

If the sampling has a very large combinatorial number of possible realizations — for instance sampling mini-batches without replacement — then this expectation becomes intractable to calculate. This observation motivates the development of functional upper-bounds of the expected smoothness constant that can be efficiently evaluated.

2.3. Mini-batch without replacement: b–nice sampling

Now we will choose a distribution of the sampling vector v based on a mini-batch sampling without replacement. We denote a mini-batch as $B \subseteq [n]$ and its size as $b = |B|$.

Definition 3 (*b-nice sampling*)**.** S is a b-nice sampling if S is a set valued map with a probability distribution given by

$$\mathbb{P}[S = B] = \frac{1}{\binom{n}{b}}, \quad \forall B \subseteq [n] \text{ s.t. } |B| = b \quad .$$

We can construct a sampling vector based on a b-nice sampling by setting $v = \frac{n}{b}\sum_{i \in S} e_i$, where $e_1, \ldots, e_n$ is the canonical basis of $\mathbb{R}^n$. Indeed, v is a sampling vector according to Definition 1 since for every $i \in [n]$ we have

$$v_i = \left(\frac{n}{b}\sum_{j \in S} e_j\right)_i = \frac{n}{b}\mathbb{1}_S(i) \quad , \tag{9}$$

Algorithm 1 JacSketch version of b-nice SAGA

```
Input      : mini-batch size $b$, step size $\gamma > 0$
Initialize : $w^0 \in \mathbb{R}^d$, $J^0 \in \mathbb{R}^{d\times n}$
for $k = 0, 1, 2, \dots$ do
    Sample a fresh batch $B \subseteq [n]$ s.t. $|B| = b$
    $g^k = \frac{1}{n} J^k e + \frac{1}{b}\sum_{i\in B}(\nabla f_i(w^k) - J^k_{:i})$
                                        // update the gradient estimate
    $J^{k+1}_{:i} = \begin{cases} J^k_{:i}, & \text{if } i \notin B \\ \nabla f_i(w^k), & \text{if } i \in B. \end{cases}$
                                        // update the Jacobian estimate
    $w^{k+1} = w^k - \gamma g^k$                       // take a step
return $w^k$
```

where $\mathbb{1}_S$ denotes the indicator function of the random set S. Now taking expectation in (9) gives

$$\mathbb{E}[v_i] = \frac{n}{b}\frac{1}{\binom{n}{b}} \sum_{B\subseteq[n]\,:\,|B|=b} \mathbb{1}_B(i) = \frac{n}{b\binom{n}{b}}\binom{n-1}{b-1} = 1,$$

using $|\{B \subseteq [n] : |B| = b \wedge i \in B\}| = \binom{n-1}{b-1}$.

Here we are interested in the mini-batch SAGA algorithm with b-nice sampling, which we refer to as the b-nice SAGA. In particular, b-nice SAGA is the result of using b-nice sampling, together with a linear model for the control variate $z_v(w)$. Different choices of the control variate $z_v(w)$ also recover popular algorithms such as gradient descent, SGD or the standard SAGA method (see Table 1 for some examples).

A naive implementation of b-nice SAGA based on the JacSketch algorithm is given in Algorithm 1[2].

3. Upper Bounds on the Expected Smoothness

To determine an optimal mini-batch size b^* for b-nice SAGA, we first state our assumptions and provide bounds of the smoothness of the subsampled function. We then define b^* as the mini-batch size that minimizes the total complexity of the considered algorithm, *i.e.,* the total number of stochastic gradients computed. Finally we provide upper-bounds on the expected smoothness constant $\mathcal{L}$, through which we can deduce optimal mini-batch sizes. Many proofs are deferred to the supplementary material.

3.1. Assumptions and notation

We consider that the objective function is a GLM with quadratic regularization controlled by a parameter $\lambda > 0$:

$$w^* \in \arg\min_{w\in\mathbb{R}^d} f(w) = \frac{1}{n}\sum_{i=1}^n \phi_i(a_i^\top w) + \frac{\lambda}{2}\|w\|_2^2 \;, \quad (10)$$

[2]We also provide a more efficient implementation that we used for our experiments in the appendix in Algorithm 2.

Table 1: Algorithms covered by JacSketch and corresponding sampling vector v and control variates z_v.

Parameters	v	$\nabla z_v(w)$
GD	$e = e_1 + \cdots + e_n$	$\nabla f_i(w)$
SGD	$ne_i,\ i \sim \frac{1}{n}$	0
SAGA	$ne_i,\ i \sim \frac{1}{n}$	$J_{:i}$
b-nice SAGA	$(n/b)\sum_{i\in S} e_i$	$(1/b)\sum_{i\in S} J_{:i}$

with $\|\cdot\|_2$ is the Euclidean norm, $\phi_1, \dots, \phi_n$ are convex functions and $a_1, \dots, a_n$ a sequence of observations in $\mathbb{R}^d$. This framework covers regularized logistic regression by setting $\phi_i(z) = \log(1 + \exp(-y_i z))$ for some binary labels $y_i, \dots, y_n$ in $\{\pm 1\}$, ridge regression if $\phi_i(z) = (z - y_i)^2/2$ for real observations $y_i, \dots, y_n$, and conditional random fields for when the y_i's are structured outputs.

We assume that the second derivative of each ϕ_i is uniformly bounded, which holds for our aforementioned examples.

Assumption 1 (Bounded second derivatives)**.** There exists $U \geq 0$ such that $\phi_i''(x) \leq U, \forall x \in \mathbb{R}, \forall i \in [n]$.

For a batch $B \subseteq [n]$, we rewrite the subsampled function as

$$f_B(w) := \frac{1}{|B|}\sum_{i\in B} \phi_i(a_i^\top w) + \frac{\lambda}{2}\|w\|_2^2 \;,$$

and its second derivative is thus given by

$$\nabla^2 f_B(w) = \frac{1}{|B|}\sum_{i\in B} \phi_i''(a_i^\top w) a_i a_i^\top + \lambda I_d \;, \quad (11)$$

where I_d denotes the identity matrix of size d.

For a symmetric matrix M, we write $\lambda_{\max}(M)$ (resp. $\lambda_{\min}(M)$) for its largest (resp. smallest) eigenvalue. Assumption 1 directly implies the following.

Lemma 2 (Subsample smoothness constant)**.** *Let $B \subset [n]$, and let $A_B = [a_i]_{i\in B}$ denote the column concatenation of the vectors a_i with $i \in B$. The smoothness constant of the subsampled loss function $\frac{1}{|B|}\sum_{i\in B}\phi_i(a_i^\top w)$ is given by*

$$L_B := \frac{U}{|B|}\lambda_{\max}\left(\sum_{i\in B} a_i a_i^\top\right) = \frac{U}{|B|}\lambda_{\max}\left(A_B A_B^\top\right). \quad (12)$$

Proof. The proof follows from Assumption 1 as

$$\frac{1}{|B|}\sum_{i\in B}\nabla^2\phi_i(a_i^\top w) = \frac{1}{|B|}\sum_{i\in B}\phi_i''(a_i^\top w)a_i a_i^\top \preceq \frac{U}{|B|}A_B A_B^\top. \qquad \square$$

Combined with (11), we get that f_B is $(L_B + \lambda)$-smooth.

Another key quantity in our analysis is the strong convexity parameter.

Definition 4. The strong convexity parameter is given by

$$\mu := \min_{w \in \mathbb{R}^d} \lambda_{\min}\left(\nabla^2 f(w)\right).$$

Since we have an explicit regularization term with $\lambda > 0$, f is strongly convex and $\mu \geq \lambda > 0$.

We additionally define L_i, resp. L, as the smoothness constant of the individual function $\phi_i(a_i^\top w)$, resp. the whole function $\frac{1}{n}\sum_{i=1}^n \phi_i(a_i^\top w)$. We also recall the definitions of the maximum of the individual smoothness constants by $L_{\max} := \max_{i\in[n]} L_i$ and their average by $\bar{L} := \frac{1}{n}\sum_{i=1}^n L_i$. The three constants satisfies

$$L \quad \leq \quad \bar{L} \quad \leq \quad L_{\max}. \tag{13}$$

The proof of (13) is given in Lemma 10 in the appendix.

3.2. Path to the optimal mini-batch size

Our starting point is the following theorem taken from combining Theorem 3.6 and Eq. (103) in (Gower et al., 2018)[3].

Theorem 1. *Consider the iterates w^k of Algorithm 1. Let the step size be given by*

$$\gamma = \frac{1}{4}\frac{1}{\max\left\{\mathcal{L}+\lambda, \frac{1}{b}\frac{n-b}{n-1}L_{\max} + \frac{\mu}{4}\frac{n}{b}\right\}}. \tag{14}$$

Given an $\epsilon > 0$, if $k \geq K_{\text{iter}}(b)$ where

$$K_{\text{iter}}(b) := \left\{\frac{4(\mathcal{L}+\lambda)}{\mu}, \frac{n}{b} + \frac{n-b}{n-1}\frac{4(L_{\max}+\lambda)}{b\mu}\right\}\log\left(\frac{1}{\epsilon}\right), \tag{15}$$

then $\mathbb{E}\left[\left\|w^k - w^*\right\|^2\right] \leq \epsilon C$, where $C > 0$ is a constant [4].

Through Theorem 1 we can now explicitly see how the expected smoothness constant $\mathcal{L}$ controls both the step size and the resulting iteration complexity. This is why we need bounds on $\mathcal{L}$ so that we can set the step size. In particular, we will show that the expected smoothness constant is a function of the mini-batch size b. Consequently so is the step size, the iteration complexity and the *total complexity*. We denote K_{total} the total complexity defined as the number of stochastic gradients computed, hence with (15),

$$\begin{aligned} K_{\text{total}}(b) &= bK_{\text{iter}}(b) \\ &= \max\left\{\frac{4b(\mathcal{L}+\lambda)}{\mu}, n + \frac{n-b}{n-1}\frac{4(L_{\max}+\lambda)}{\mu}\right\}\log\left(\frac{1}{\epsilon}\right). \end{aligned} \tag{16}$$

Once we have determined $\mathcal{L}$ as a function of b, we will calculate the mini-batch size b^* that optimizes the total complexity $b^* \in \arg\min_{b\in[n]} K_{\text{total}}(b)$.

As we have shown in Lemma 1, computing a precise bound on $\mathcal{L}$ can be computationally intractable. This is why we focus on finding upper bounds on $\mathcal{L}$ that can be computed, but also tight enough to be useful. To verify that our bounds are sufficiently tight, we will always have in mind the bounds $L \leq \mathcal{L} \leq L_{\max}$ given in (8). In particular, after expressing our bounds of $\mathcal{L} = \mathcal{L}(b)$ as a function of b,we would like the bounds (8) to be attained for $\mathcal{L}(1) = L_{\max}$ and $\mathcal{L}(n) = L$.

3.3. Expected smoothness

All bounds we develop on $\mathcal{L}$ are based on the following lemma, which is a specialization of (1) for b-nice sampling.

Proposition 1 (Expected smoothness constant)**.** *For the b-nice sampling, with $b \in [n]$, the expected smoothness constant is given by*

$$\mathcal{L} = \frac{1}{\binom{n-1}{b-1}} \max_{i=1,\ldots,n}\left\{\sum_{B\subseteq[n]\,:\,|B|=b\wedge i\in B} L_B\right\}. \tag{17}$$

Proof. Let S the b-nice sampling as defined in Definition 3 and let $v = \frac{n}{b}\sum_{j\in S} e_j$ be its corresponding sampling vector. Note that

$$f_v(w) = \frac{1}{n}\sum_{i\in[n]} f_i(w)v_i = \frac{1}{b}\sum_{i\in S} f_i(w) = f_S(w).$$

Finally from Lemma 1, we have that:

$$\begin{aligned} \mathcal{L} &= \mathbb{E}\left[L_v v_i\right] \overset{(9)}{=} \mathbb{E}\left[L_S \frac{n}{b}\mathbb{1}_{\{i\in S\}}\right] \\ &= \frac{1}{\binom{n}{b}}\frac{n}{b}\sum_{B\subseteq[n]\,:\,|B|=b} L_B \mathbb{1}_{\{i\in B\}} \\ &= \frac{1}{\binom{n}{b}}\frac{n}{b}\sum_{\substack{B\subseteq[n]\,: \\ |B|=b\wedge i\in B}} L_B = \frac{1}{\binom{n-1}{b-1}}\sum_{\substack{B\subseteq[n]\,: \\ |B|=b\wedge i\in B}} L_B. \end{aligned}$$

Taking the maximum over all $i \in [n]$ gives the result. □

The first bound we present is technically the simplest to derive, which is why we refer to it as the *simple bound.*

Theorem 2 (Simple bound)**.** *For a b-nice sampling S, for $b \in [n]$, we have that*

$$\mathcal{L} \leq \mathcal{L}_{\text{simple}}(b) := \frac{n}{b}\frac{b-1}{n-1}\bar{L} + \frac{1}{b}\frac{n-b}{n-1}L_{\max}\;, \tag{18}$$

[3]Note that λ has been added to every smoothness constant since the analysis in Gower et al. (2018) depends on the $(L+\lambda)$-smoothness of f and the $(L_B+\lambda)$-smoothness of the subsampled functions f_B.

[4]Specifically, let $J^0 \in \mathbb{R}^{d\times n}$ be the initiated Jacobian of the b-nice SAGA Algorithm 1. Then this constant is

$$C := \left\|w^0 - w^*\right\|^2 + \frac{\gamma}{2L_{\max}}\sum_{i\in[n]}\left\|J^0_{:i} - \nabla f(w^*)\right\|^2.$$

Proof. The proof, given in Appendix A.2, starts by using the that $L_B \leq \frac{1}{b}\sum_{j\in B} L_j$ for all subsets B, which follows from repeatedly applying Lemma 8 in the appendix. The remainder of the proof follows by straightforward counting arguments. $\square$

The previous bound interpolates, respectively for $b = 1$ and $b = n$, between $L_{\max}$ and $\bar{L}$. On the one hand, we have that $\mathcal{L}_{\text{simple}}(b)$ is a good bound for when b is small, since $\mathcal{L}_{\text{simple}}(1) = L_{\max}$. Though $\mathcal{L}_{\text{simple}}(b)$ may not be a good bound for large b, since $\mathcal{L}_{\text{simple}}(n) = \overline{L} \geq L$, thanks to (13). Thus $\mathcal{L}_{\text{simple}}(b)$ does not achieve the left-hand side of (8). Indeed $\overline{L}$ can be far from L. For instance[5], if $f(w) = \frac{1}{n}\sum_{i\in[n]}\frac{1}{2}(a_i^\top w - b_i)^2$ is a quadratic function, then we have that $\overline{L} = \frac{1}{n}\text{Tr}\left(AA^\top\right)$ and $L = \frac{1}{n}\lambda_{\max}(AA^\top)$. Thus if the eigenvalues of $AA^\top$ are all equal then $\overline{L} = dL$. Alternatively, if one eigenvalue is significantly larger than the rest then $\overline{L} \approx L$.

Due to this shortcoming of $\mathcal{L}_{\text{simple}}$, we now derive the *Bernstein bound*. This bound explicitly depends on L instead of $\bar{L}$, and is developed through a specialized variant of a matrix Bernstein inequality (Tropp, 2012; 2015) for sampling *without* replacement in Appendix C.

Theorem 3 (Bernstein bound)**.** *The expected smoothness constant is upper bounded by*

$$\mathcal{L} \leq \mathcal{L}_{\textit{Bernstein}}(b) := 2\frac{b-1}{b}\frac{n}{n-1}L + \frac{1}{b}\left(\frac{n-b}{n-1} + \frac{4}{3}\log d\right)L_{\max}. \tag{19}$$

Checking again the bounds of $\mathcal{L}_{\text{Bernstein}}(b)$, we have on the one hand that $\mathcal{L}_{\text{Bernstein}}(1) = \left(1 + \frac{4}{3}\log d\right)L_{\max} \geq L_{\max}$, thus there is a little bit of slack for b small. On the other hand, using $\frac{1}{n}L_{\max} \leq L$ (see Lemma 10 in appendix), we have that

$$\mathcal{L}_{\text{Bernstein}}(n) = 2L + \frac{1}{n}\frac{4}{3}\log d\, L_{\max} \leq \left(2 + \frac{4}{3}\log d\right)L,$$

which depends only logarithmically on d. Thus we expect the *Bernstein bound* to be more useful in the large d domains, as compared to the *simple bound*. We confirm this numerically in Section 5.1.

Remark 2. The simple bound is relatively tight for b small, while the Bernstein bound is better for large b and large d. Fortunately, we can obtain a more refined bound by taking the minimum of the *simple* and the *Bernstein bounds*. This is highlighted numerically in Section 5.

Next we propose a *practical estimate* of $\mathcal{L}$ that is tight for both small and large mini-batch sizes.

Definition 5 (Practical estimate)**.**

$$\mathcal{L}_{\text{practical}}(b) := \frac{n}{b}\frac{b-1}{n-1}L + \frac{1}{b}\frac{n-b}{n-1}L_{\max} \ . \tag{20}$$

[5]We numerically explore such extreme settings in Section 5.

Indeed $\mathcal{L}_{\text{practical}}(1) = L_{\max}$ and $\mathcal{L}_{\text{practical}}(n) = L$, achieving both limits of (8). The downside to $\mathcal{L}_{\text{practical}}(b)$ is that it is not an upper bound of $\mathcal{L}$. Rather, we are able to show that $\mathcal{L}_{\text{practical}}(b)$ is very close to a valid smoothness constant, but it can be slightly smaller. Our theoretical justification for using $\mathcal{L}_{\text{practical}}(b)$ comes from a mid step in the proof of the Bernstein bound which is captured in the next lemma.

Lemma 3. *Let $a_j \in \mathbb{R}^d$ for $j \in [n]$ and let S^i be a $(b-1)$-nice sampling over $[n] \setminus \{i\}$, for every $i \in [n]$. It follows that*

$$\mathcal{L} \leq \mathcal{L}_{\textit{practical}}(b) + U\max_{i\in[n]}\mathbb{E}\left[\lambda_{\max}\left(N_i\right)\right] \ , \tag{21}$$

with $N_i := \frac{1}{b}\sum_{j\in S^i} a_j a_j^\top - \frac{1}{b}\frac{b-1}{n-1}\sum_{j\in[n]\setminus\{i\}} a_j a_j^\top$.

Proof. The proof is given in Appendix A.3. $\square$

Lemma 3 shows that the expected smoothness constant is upper-bounded by $\mathcal{L}_{\text{practical}}(b)$ and an additional term. In this additional term we have the largest eigenvalue of a random matrix. This matrix is zero in expectation, and we also find that its eigenvalues oscillate around zero. Indeed, we provide extensive experiments in Section 5 confirming that $\mathcal{L}_{\text{practical}}(b)$ is very close to $\mathcal{L}$ given in (17).

4. Optimal Mini-Batch Sizes

Now that we have established the *simple* and the *Bernstein bounds*, we can minimize the total complexity (16) in the mini-batch size.

For instance for the *simple bound*, given $\epsilon > 0$ and plugging in (18) into (16) gives

$$K_{\text{total}}(b) \leq \max\left\{g_{\text{simple}}(b), h(b)\right\}\log\left(\frac{1}{\epsilon}\right) \ ,$$

where $g_{\text{simple}}(b) := \frac{4\left(n\bar{L} - L_{\max} + (n-1)\lambda\right)}{\mu(n-1)}b + \frac{4n\left(L_{\max} - \bar{L}\right)}{\mu(n-1)}$, and $h(b) := -\frac{4(L_{\max}+\lambda)}{\mu(n-1)}b + n\left(1 + \frac{1}{n-1}\frac{4(L_{\max}+\lambda)}{\mu}\right)$.

Remark 3. The right-hand side term $h(b)$ is common to all our bounds since it does not depend on $\mathcal{L}$. It linearly decreases from $h(1) = n + \frac{4(L_{\max}+\lambda)}{\mu}$ to $h(n) = n$.

We note that $g_{\text{simple}}(b)$ is a linearly increasing function of b, because $L_{\max} \leq n\bar{L}$ (as proven in Lemma 10). One can easily verify that $g_{\text{simple}}(b)$ and $h(b)$ cross, as presented in Figure 2, by looking at initial and final values:

- At $b = 1$, $g_{\text{simple}}(1) = \frac{4}{\mu}(L_{\max} + \lambda) = h(1) - n$. So, $g_{\text{simple}}(1) \leq h(1)$.
- At $b = n$, $g_{\text{simple}}(n) = \frac{4(\bar{L}+\lambda)}{\mu}n = \frac{4(\bar{L}+\lambda)}{\mu}h(n)$. Since $\bar{L} \geq \mu$, we get $g_{\text{simple}}(n) \geq h(n)$.

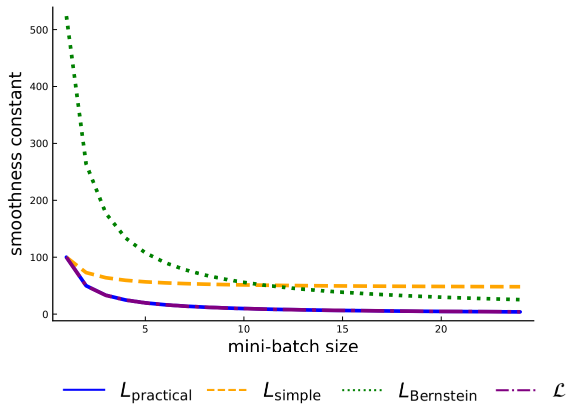

Figure 3: Expected smoothness constant $\mathcal{L}$ and its upper-bounds as a function of the mini-batch size b for ridge regression on the unscaled *staircase eigval* dataset ($\lambda = 10^{-3}$).

Consequently, solving $g_{\text{simple}}(b) = h(b)$ in b gives the optimal mini-batch size

$$b^*_{\text{simple}} = \left\lfloor 1 + \frac{\mu(n-1)}{4(\bar{L}+\lambda)} \right\rfloor . \tag{22}$$

For the *Bernstein bound*, plugging (19) into (16) leads to

$$K_{\text{total}}(b) \le \max\left\{g_{\text{Bernstein}}(b), h(b)\right\} \log\left(\frac{1}{\epsilon}\right) , \tag{23}$$

where

$$g_{\text{Bernstein}}(b) := \tfrac{4}{\mu(n-1)}\left(2nL - L_{\max} + (n-1)\lambda\right) b + \tfrac{4n}{\mu(n-1)}\left(L_{\max} - 2L\right) + \tfrac{16}{3\mu}\log(d) L_{\max} .$$

The function $g_{\text{Bernstein}}$ is also linearly increasing in b and its initial and final values are

- At $b{=}1$, $g_{\text{Bernstein}}(1) = (1 + \frac{4}{3}\log d)\frac{4L_{\max}}{\mu} + \frac{4\lambda}{\mu}$.
- At $b{=}n$, $g_{\text{Bernstein}}(n) {=} n\frac{4(2L+\lambda)}{\mu} + \frac{16}{3\mu}(L_{\max}{+}\lambda)\log(d)$. Since $L \ge \mu$, we get $g_{\text{Bernstein}}(n) \ge h(n)$.

Yet, it is unclear whether $g_{\text{Bernstein}}(1)$ is dominated by $h(1)$. This is why we need to distinguish two cases to minimize the total complexity, which leads to the following solution

$$b^*_{\text{Bernstein}} = \begin{cases} \left\lfloor 1 + \frac{\mu(n-1)}{4(2L+\lambda)} - \frac{4}{3}\log d \frac{n-1}{n}\frac{L_{\max}}{2L+\lambda} \right\rfloor, & \text{if } \frac{4}{3}\frac{4L_{\max}}{\mu}\log d \le n, \\ 1, & \text{otherwise} . \end{cases}$$

In the first case, the problem is well-conditioned and $g_{\text{Bernstein}}$ and h do cross at a mini-batch size between 1 and n. In the second case, the total complexity K_{total} is governed by $g_{\text{Bernstein}}$ because $g_{\text{Bernstein}}(b) \ge h(b)$ for all $b \in [n]$, and the resulting optimal mini-batch size is $b = 1$.

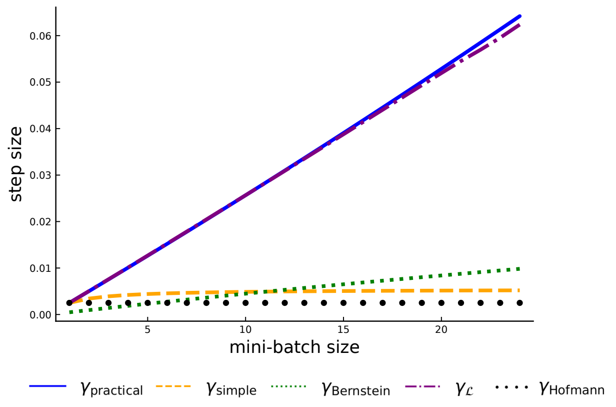

Figure 4: Step size estimates as a function the mini-batch size b for ridge regression on the unscaled *staircase eigval* dataset ($\lambda = 10^{-3}$).

5. Numerical Study

All the experiments were run in Julia and the code is freely available on https://github.com/gowerrobert/StochOpt.jl.

5.1. Upper-bounds of the expected smoothness constant

First we experimentally verify that our upper-bounds hold and how much slack there is between them and $\mathcal{L}$ given in Equation (17). For ridge regression applied to artificially generated small datasets, we compute Equation (17) and compare it to our *simple* and *Bernstein bounds*, and our *practical* estimate. Our data are matrices $A \in \mathbb{R}^{d\times n}$ defined as follows

- *uniform* $(n = 24, d = 50) : [A]_{ij} \sim \mathcal{U}([0,1))$,
- *alone eigval* $(n = d = 24) : A = \text{diag}\,(1,\dots,1,100)$,
- *staircase eigval* $(n = d = 24)$:
$$A = \text{diag}\left(1, 10\sqrt{1/n}, \dots, 10\sqrt{(n-2)/n}, 10\right) .$$

In Figure 4 we see that $\mathcal{L}_{\text{practical}}$ is arbitrarily close to $\mathcal{L}$, making it hard to distinguish the two line plots. This was the case in many other experiments, which we defer to Appendix E.1. For this reason, we use $\gamma_{\text{practical}}$ in our experiments with the SAGA method.

Furthermore, in accordance with our discussion in Section 3.3, we have that $\mathcal{L}_{\text{simple}}$ and $\mathcal{L}_{\text{Bernstein}}$ are close to $\mathcal{L}$ when b is small and large, respectively. In Appendix E.2 we show, by applying ridge and regularized logistic regression to publicly available datasets from LIBSVM[6] and the UCI repository[7], that the *simple bound* performs better than the

[6] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
[7] https://archive.ics.uci.edu/ml/datasets/

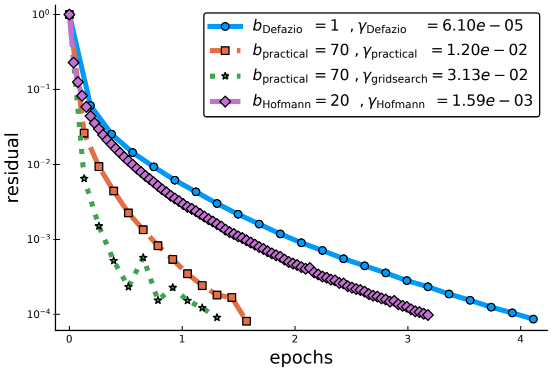

Figure 5: Comparison of SAGA settings for ridge regression on the unscaled *slice* dataset ($\lambda = 10^{-1}$).

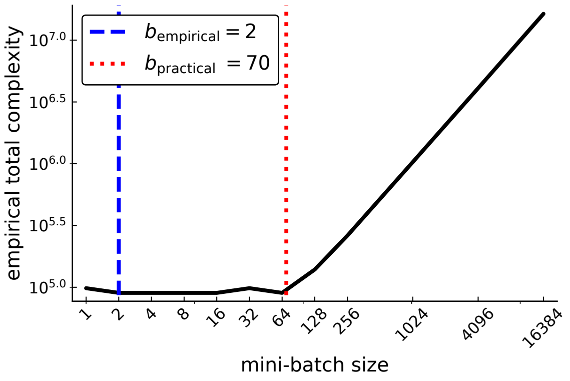

Figure 6: Total complexity versus the mini-batch size for ridge regression on the unscaled *slice* dataset ($\lambda = 10^{-1}$).

Bernstein bound when $n \gg d$, and conversely for d slightly smaller than n, larger than n or when scaling the data.

5.2. Related step size estimation

Different bounds on $\mathcal{L}$ also give different step sizes (14). Plugging in our estimates $\mathcal{L}_{\text{simple}}$, $\mathcal{L}_{\text{Bernstein}}$ and $\mathcal{L}_{\text{practical}}$ into (14) gives the step sizes γ_{simple}, $\gamma_{\text{Bernstein}}$ and $\gamma_{\text{practical}}$, respectively. We compare our resulting step sizes to $\gamma_{\mathcal{L}}$ where $\mathcal{L}$ is given by Eq. (17) and to the step size given by Hofmann et al. (2015), which is $\gamma_{\text{Hofmann}}(b) = \frac{K}{2L_{\max}(1+K+\sqrt{1+K^2})}$, where $K := \frac{4bL_{\max}}{n\mu}$. We can see in Figure 4, that for $b = 1$, all the step sizes are approximately the same, with the exceptions of the Bernstein step size. For $b > 5$, all of our step sizes are larger than $\gamma_{\text{Hofmann}}(b)$, in particular $\gamma_{\text{practical}}(b)$ is significantly larger. These observations are verified in other artificial and real data examples in Appendices E.3 and E.4.

5.3. Comparison with previous SAGA settings

Here we compare the performance of SAGA when using the mini-batch size and step size $b = 1, \gamma_{\text{Defazio}} := 1/3(n\mu + L_{\max})$ given in (Defazio et al., 2014), $b = 20$ and $\gamma_{\text{Hofmann}} = 20/n\mu$ given in Hofmann et al. (2015), to our new practical mini-batch size $b_{\text{practical}} = \left\lfloor 1 + \frac{1}{4L}\mu(n-1) \right\rfloor$ and step size $\gamma_{\text{practical}}$. Our goal is to verify how much our parameter setting can improve practical performance. We also compare with a step size $\gamma_{\text{gridsearch}}$ obtained by grid search over odd powers of 2. These methods are run until they reach a relative error of 10^{-4}.

We find in Figure 5 that our parameter settings $(\gamma_{\text{practical}}, b_{\text{practical}})$ significantly outperforms the previously suggested parameters, and is even comparable to grid search. In Appendix E.5, we show that the settings $(\gamma_{\text{Hofmann}}, b = 20)$ can lead to very poor performance compared to our settings. We also show that our settings are performing very well both in terms of epochs and time. Finally, our estimate $b_{\text{practical}}$ can be closed to n (see Figure 21), meaning that our theory can also predict when to use gradient desent rather than stochastic gradient methods.

5.4. Optimality of our mini-batch size

In the last experiment, detailed in Appendix E.6, we show that our estimation of the optimal mini-batch size $b_{\text{practical}}$ is close to the best one found through a grid search. We build a grid of mini-batch sizes[8] and compute the empirical complexity required to achieve a relative error of 10^{-4}, as in Section 5.3. In Figure 6 we can see that the empirical complexity of the optimal mini-batch size calculated through grid search is very close to the resulting empirical complexity of using $b_{\text{practical}}$. What is even more interesting, is that $b_{\text{practical}}$ seems to predict a regime change (except in Figure 27a), where using a larger mini-batch size results in a much larger empirical complexity.

6. Conclusions

We have explained the crucial role of the expected smoothness constant $\mathcal{L}$ in the convergence of a family of stochastic variance-reduced descent algorithms. We have developed functional upper-bounds of this constant to build larger step sizes and closed-form optimal mini-batch values for the b-nice SAGA algorithm. Our experiments on artificial and real datasets showed the validity of our upper-bounds and the improvement in the total complexity using our step and optimal mini-batch sizes. Our results suggest a new parameter setting for mini-batch SAGA, that significantly outperforms previous suggested ones, and is even comparable with a grid search approach, without the computational burden of the later.

[8]Our grid is $\{2^i, i = 0, \ldots, 14\}$, with $2^{16}, 2^{18}$ and n being added when needed.

Acknowledgements

This work was supported by grants from DIM Math Innov Région Ile-de-France (ED574 - FMJH) and by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, in a joint call with Gaspard Monge Program for optimization, operations research and their interactions with data sciences.

References

Allen-Zhu, Z. Katyusha: The First Direct Acceleration of Stochastic Gradient Methods. In *STOC*, 2017.

Bach, F. Sharp analysis of low-rank kernel matrix approximations. In *COLT 2013 - The 26th Annual Conference on Learning Theory*, pp. 185–209, 2013.

Chang, C.-C. and Lin, C.-J. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

Defazio, A., Bach, F., and Lacoste-Julien, S. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems 27*, pp. 1646–1654. 2014.

Dheeru, D. and Karra Taniskidou, E. UCI machine learning repository, 2017.

Gower, R. M., Richtárik, P., and Bach, F. Stochastic quasi-gradient methods: Variance reduction via Jacobian sketching. *arXiv preprint arXiv:1805.02632*, 2018.

Gross, D. and Nesme, V. Note on sampling without replacing from a finite collection of matrices. *arXiv preprint arXiv:1001.2738*, 2010.

Hoeffding, W. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

Hofmann, T., Lucchi, A., Lacoste-Julien, S., and McWilliams, B. Variance reduced stochastic gradient descent with neighbors. In *Advances in Neural Information Processing Systems*, pp. 2305–2313, 2015.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, pp. 315–323. Curran Associates, Inc., 2013.

Konečný, J. and Richtárik, P. Semi-stochastic gradient descent methods. *Frontiers in Applied Mathematics and Statistics*, 3:9, 2017.

Nesterov, Y. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer Publishing Company, Incorporated, 1 edition, 2014.

Nguyen, L. M., Liu, J., Scheinberg, K., and Takáč, M. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2613–2621. PMLR, Aug 2017.

Raj, A. and Stich, S. U. SVRG meets SAGA: k-SVRG — a tale of limited memory. *arXiv:1805.00982*, 2018.

Robbins, H. and Monro, S. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

Schmidt, M., Le Roux, N., and Bach, F. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1):83–112, Mar 2017.

Shalev-Shwartz, S. and Zhang, T. Stochastic dual coordinate ascent methods for regularized loss. *Journal of Machine Learning Research*, 14(1):567–599, February 2013.

Tropp, J. A. Improved analysis of the subsampled randomized Hadamard transform. *Advances in Adaptive Data Analysis*, 3(01n02):115–126, 2011.

Tropp, J. A. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012.

Tropp, J. A. An introduction to matrix concentration inequalities. *Foundations and Trends in Machine Learning*, 8(1-2):1–230, 2015.