Semi-Stochastic Gradient Descent Methods

Jakub Konečný [*] Peter Richtárik [†]

School of Mathematics
University of Edinburgh
United Kingdom

December 4, 2013

Abstract

In this paper we study the problem of minimizing the average of a large number (n) of smooth convex loss functions. We propose a new method, S2GD (Semi-Stochastic Gradient Descent), which runs for one or several epochs in each of which a single full gradient and a random number of stochastic gradients is computed, following a geometric law. The total work needed for the method to output an ε-accurate solution in expectation, measured in the number of passes over data, or equivalently, in units equivalent to the computation of a single gradient of the loss, is $O((\kappa/n)\log(1/\varepsilon))$, where κ is the condition number. This is achieved by running the method for $O(\log(1/\varepsilon))$ epochs, with a single gradient evaluation and $O(\kappa)$ stochastic gradient evaluations in each. The SVRG method of Johnson and Zhang [3] arises as a special case. If our method is limited to a single epoch only, it needs to evaluate at most $O((\kappa/\varepsilon)\log(1/\varepsilon))$ stochastic gradients. In contrast, SVRG requires $O(\kappa/\varepsilon^2)$ stochastic gradients. To illustrate our theoretical results, S2GD only needs the workload equivalent to about 2.1 full gradient evaluations to find an 10^{-6}-accurate solution for a problem with $n = 10^9$ and $\kappa = 10^3$.

1 Introduction

Many problems in data science (e.g., machine learning, optimization and statistics) can be cast as loss minimization problems of the form

$$\min_{x\in\mathbb{R}^d} f(x), \tag{1}$$

where

$$f(x) \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} f_i(x). \tag{2}$$

[*]School of Mathematics, The University of Edinburgh, United Kingdom (e-mail: J.Konecny@sms.ed.ac.uk)

[†]School of Mathematics, The University of Edinburgh, United Kingdom (e-mail: peter.richtarik@ed.ac.uk)
The work of both authors was supported by the Centre for Numerical Algorithms and Intelligent Software (funded by EPSRC grant EP/G036136/1 and the Scottish Funding Council). Both authors also thank the Simons Institute for the Theory of Computing, UC Berkeley, where this work was conceived and finalized. The work of P.R. was also supported by the EPSRC grant EP/I017127/1 (Mathematics for Vast Digital Resources) and EPSRC grant EP/K02325X/1 (Accelerated Coordinate Descent Methods for Big Data Problems).

Here d typically denotes the number of features / coordinates, n the number of examples, and $f_i(x)$ is the loss incurred on example i. That is, we are seeking to find a predictor $x \in \mathbb{R}^d$ minimizing the average loss $f(x)$. In big data applications, n is typically very large; in particular, $n \gg d$.

1.1 Motivation

Let us now briefly review two basic approaches to solving problem (1).

1. *Gradient Descent.* Given $x_k \in \mathbb{R}^d$, the gradient descent (GD) method sets

   $$x_{k+1} = x_k - hf'(x_k),$$

   where h is a stepsize parameter and $f'(x_k)$ is the gradient of f at x_k. We will refer to $f'(x)$ by the name *full gradient.* In order to compute $f'(x_k)$, we need to compute the gradients of n functions. Since n is big, it is prohibitive to do this at every iteration.

2. *Stochastic Gradient Descent (SGD).* Unlike gradient descent, stochastic gradient descent [7, 17] instead picks a random i (uniformly) and updates

   $$x_{k+1} = x_k - hf_i'(x_k).$$

   Note that this strategy drastically reduces the amount of work that needs to be done in each iteration (by the factor of n). Since

   $$\mathbf{E}(f_i'(x_k)) = f'(x_k),$$

   we have an unbiased estimator of the full gradient. Hence, the gradients of the component functions $f_1, \ldots, f_n$ will be referred to as *stochastic gradients.* A practical issue with SGD is that consecutive stochastic gradients may vary a lot or even point in opposite directions. This slows down the performance of SGD. On balance, however, SGD is preferable to GD in applications where low accuracy solutions are sufficient. In such cases usually only a small number of passes through the data (i.e., work equivalent to a small number of full gradient evaluations) are needed to find an acceptable x. For this reason, SGD is extremely popular in fields such as machine learning.

In order to improve upon GD, one needs to reduce the cost of computing a gradient. In order to improve upon SGD, one has to reduce the variance of the stochastic gradients. In this paper we propose and analyze a *Semi-Stochastic Gradient Descent* (S2GD) method. Our method combines GD and SGD steps and reaps the benefits of both algorithms: it inherits the stability and speed of GD and at the same time retains the work-efficiency of SGD.

1.2 Brief literature review

Several recent papers, e.g., Richtárik & Takáč [9], Le Roux, Schmidt & Bach [12, 13], Shalev-Shwartz & Zhang [14] and Johnson & Zhang [3] proposed methods which achieve such a variance-reduction effect, directly or indirectly. These methods enjoy linear convergence rates when applied to minimizing smooth strongly convex loss functions.

The method in [9] is known as Random Coordinate Descent for Composite functions (RCDC), and can be either applied directly to (1)—in which case a single iteration requires $O(n)$ work for a dense problem, and $O(d\log(1/\varepsilon))$ iterations in total—or to a dual version of (1), which requires $O(d)$ work per iteration and $O((n+\kappa)\log(1/\varepsilon))$ iterations in total. Application of a coordinate descent method to a dual formulation of (1) is generally referred to as Stochastic Dual Coordinate Ascent (SDCA) [2]. The algorithm in [14] exhibits this duality, and the method in [15] extends the primal-dual framework to the parallel / mini-batch setting. Parallel and distributed stochastic coordinate descent methods were studied in [10, 1, 11].

Stochastic Average Gradient (SAG) [12] is one of the first SGD-type methods, other than coordinate descent methods, which were shown to exhibit linear convergence. The method of Johnson and Zhang [3], called Stochastic Variance Reduced Gradient (SVRG), arises as a special case in our setting for a suboptimal choice of a single parameter of our method. The Epoch Mixed Gradient Descent (EMGD) method [16] is similar in spirit to SVRG, but achieves a quadratic dependence on the condition number instead of a linear dependence, as is the case with SAG, SVRG and with our method.

For classical work on semi-stochastic gradient descent methods we refer[1] the reader to the papers of Murti and Fuchs [5, 6].

1.3 Outline

We start in Section 2 by describing two algorithms: S2GD, which we analyze, and S2GD+, which we do not analyze, but which exhibits superior performance in practice. We then move to summarizing some of the main contributions of this paper in Section 3. Section 4 is devoted to establishing expectation and high probability complexity results for S2GD in the case of a strongly convex loss. The results are generic in that the parameters of the method are set arbitrarily. Hence, in Section 5 we study the problem of choosing the parameters optimally, with the goal of minimizing the total workload (# of processed examples) sufficient to produce a result of sufficient accuracy. In Section 6 we establish high probability complexity bounds for S2GD applied to a non-strongly convex loss function. Finally, in Section 7 we perform very encouraging numerical experiments on real and artificial problem instances. A brief conclusion can be found in Section 8.

2 Semi-Stochastic Gradient Descent

In this section we describe two novel algorithms: S2GD and S2GD+. We analyze the former only. The latter, however, has superior convergence properties in our experiments.

We assume throughout the paper that the functions f_i are convex and L-smooth.

Assumption 1. *The functions $f_1,\dots,f_n$ have Lipschitz continuous gradients with constant $L>0$ (in other words, they are L-smooth). That is, for all $x,z\in\mathbb{R}^d$ and all $i=1,2,\dots,n$,*

$$f_i(z) \le f_i(x) + \langle f_i'(x), z-x\rangle + \frac{L}{2}\|z-x\|^2.$$

(This implies that the gradient of f is Lipschitz with constant L, and hence f satisfies the same inequality.)

In one part of the paper (Section 4) we also make the following additional assumption:

[1]We thank Zaid Harchaoui who pointed us to these papers a few days before we posted our work to arXiv.

Assumption 2. *The average loss f is μ-strongly convex, $\mu > 0$. That is, for all $x, z \in \mathbb{R}^d$,*

$$f(z) \geq f(x) + \langle f'(x), z - x \rangle + \frac{\mu}{2}\|z - x\|^2. \tag{3}$$

(Note that, necessarily, $\mu \leq L$.)

2.1 S2GD

Algorithm 1 (S2GD) depends on three parameters: stepsize h, constant m limiting the number of stochastic gradients computed in a single epoch, and a $\nu \in [0, \mu]$, where μ is the strong convexity constant of f.

Algorithm 1 Semi-Stochastic Gradient Descent (S2GD)

```
parameters: $m$ = max # of stochastic steps per epoch, $h$ = stepsize, $\nu$ = lower bound on $\mu$
for $j = 0, 1, 2, \dots$ do
    $g_j \leftarrow \frac{1}{n}\sum_{i=1}^n f_i'(x_j)$
    $y_{j,0} \leftarrow x_j$
    Let $t_j \leftarrow t$ with probability $(1-\nu h)^{m-t}/\beta$ for $t = 1, 2, \dots, m$
    for $t = 0$ to $t_j - 1$ do
        Pick $i \in \{1, 2, \dots, n\}$, uniformly at random
        $y_{j,t+1} \leftarrow y_{j,t} - h\left(g_j + f_i'(y_{j,t}) - f_i'(x_j)\right)$
    end for
    $x_{j+1} \leftarrow y_{j,t_j}$
end for
```

The method has an outer loop, indexed by epoch counter j, and an inner loop, indexed by t. In each epoch j, the method first computes g_j—the *full* gradient of f at x_j. Subsequently, the method produces a random number $t_j \in [1, m]$ of steps, following a geometric law, where

$$\beta \overset{\text{def}}{=} \sum_{t=1}^{m} (1 - \nu h)^{m-t}, \tag{4}$$

with only *two stochastic gradients* computed in each step[2]. For each $t = 0, \dots, t_j - 1$, the stochastic gradient $f_i'(x_j)$ is subtracted from g_j, and $f_i'(y_{j,t-1})$ is added to g_j, which ensures that, one has

$$\mathbf{E}(g_j + f_i'(y_{j,t}) - f_i'(x_j)) = f'(y_{j,t}),$$

where the expectation is with respect to the random variable i.

Hence, the algorithm is stochastic gradient descent – albeit executed in a nonstandard way (compared to the traditional implementation described in the introduction).

Note that for all j, the expected number of iterations of the inner loop, $\mathbf{E}(t_j)$, is equal to

$$\xi = \xi(m, h) \overset{\text{def}}{=} \sum_{t=1}^{m} t \frac{(1 - \nu h)^{m-t}}{\beta}. \tag{5}$$

Also note that $\xi \in [\frac{m+1}{2}, m)$, with the lower bound attained for $\nu = 0$, and the upper bound for $\nu h \to 1$.

[2]It is possible to get away with computinge only a *single* stochastic gradient per inner iteration, namely $f_i'(y_{j,t})$, at the cost of having to store in memory $f_i'(x_j)$ for $i = 1, 2, \dots, n$. This, however, will be impractical for big n.

2.2 S2GD+

We also implement Algorithm 2, which we call S2GD+. In our experiments, the performance of this method is superior to all methods we tested, including S2GD. However, we do not analyze the complexity of this method and leave this as an open problem.

Algorithm 2 S2GD+

```
**parameters:** $\alpha \geq 1$ (e.g., $\alpha = 1$)
1. Run SGD for a single pass over the data (i.e., $n$ iterations); output $x$
2. Starting from $x_0 = x$, run a version of S2GD in which $t_j = \alpha n$ for all $j$
```

In brief, S2GD+ starts by running SGD for 1 epoch (1 pass over the data) and then switches to a variant of S2GD in which the number of the inner iterations, t_j, is not random, but fixed to be n or a small multiple of n.

The motivation for this method is the following. It is common knowledge that SGD is able to progress much more in one pass over the data than GD (where this would correspond to a single gradient step). However, the very first step of S2GD is the computation of the full gradient of f. Hence, by starting with a single pass over data using SGD and *then* switching to S2GD, we obtain a superior method in practice.[3]

3 Summary of Results

In this section we summarize some of the main results and contributions of this work.

1. **Complexity for strongly convex f.** If f is strongly convex, S2GD needs

$$\mathcal{W} = O((n + \kappa)\log(1/\varepsilon)) \tag{6}$$

work (measured as the total number of evaluations of the stochastic gradient, accounting for the full gradient evaluations as well) to output an ε-approximate solution (in expectation or in high probability), where $\kappa = L/\mu$ is the condition number. This is achieved by running S2GD with stepsize $h = O(1/L)$, $j = O(\log(1/\varepsilon))$ epochs (this is also equal to the number of full gradient evaluations) and $m = O(\kappa)$ (this is also roughly equal to the number of stochastic gradient evaluations in a single epoch). The complexity results are stated in detail in Sections 4 and 5 (see Theorems 4, 5 and 6; see also (27) and (26)).

2. **Comparison with existing results.** This complexity result (6) matches the best-known results obtained for strongly convex losses in recent work such as [12], [3] and [16]. Our treatment is most closely related to [3], and contains their method (SVRG) as a special case. However, our complexity results have better constants, which has a discernable effect in practice. In Table 1 we compare our results in the strongly convex case with other existing results for different algorithms.

   We should note that the rate of convergence of Nesterov's algorithm [8] is a deterministic result. EMGD and S2GD results hold with high probability. The remaining results hold in

[3]Using a single pass of SGD as an initialization strategy was already considered in [12]. However, the authors claim that their implementation of vanilla SAG did not benefit from it. S2GD does benefit from such an initialization due to it starting, in theory, with a (heavy) full gradient computation.

Algorithm	Complexity/Work
Nesterov's algorithm	$O\left(\sqrt{\kappa} n \log(1/\varepsilon)\right)$
EMGD	$O\left((n+\kappa^2)\log(1/\varepsilon)\right)$
SAG ($n \geq 8\kappa$)	$O\left(n \log(1/\varepsilon)\right)$
SDCA	$O\left((n+\kappa)\log(1/\varepsilon)\right)$
SVRG	$O\left((n+\kappa)\log(1/\varepsilon)\right)$
S2GD	$O\left((n+\kappa)\log(1/\varepsilon)\right)$

Table 1: Comparison of performance of selected methods suitable for solving (1). The complexity/work is measured in the number of stochastic gradient evaluations needed to find an ε-solution.

expectation. Complexity results for stochastic coordinate descent methods are also typically analyzed in the high probability regime [9].

3. **Complexity for convex f.** If f is *not* strongly convex, then we propose that S2GD be applied to a perturbed version of the problem, with strong convexity constant $\mu = O(L/\varepsilon)$. An ε-accurate solution of the original problem is recovered with arbitrarily high probability (see Theorem 8 in Section 6). The total work in this case is

$$\mathcal{W} = O\left((n + L/\varepsilon)\right)\log\left(1/\varepsilon\right)).$$

   We believe this is the first time a reduced-variance version of SGD, other than a coordinate decent method, was analyzed for non-strongly convex loss functions with complexity better than $O(1/\varepsilon^2)$. Rates of the $O(1/\varepsilon)$ variety have already been proved for algorithms such as SAG [13] and MixedGrad [4].

4. **Optimal parameters.** We derive formulas for optimal parameters of the method which (approximately) minimize the total workload, measured in the number of stochastic gradients computed (counting a single full gradient evaluation as n evaluations of the stochastic gradient). In particular, we show that the method should be run for $O(\log(1/\varepsilon))$ epochs, with stepsize $h = O(1/L)$ and $m = O(\kappa)$. No such results were derived for SVRG in [3].

5. **One epoch.** In the case when S2GD is run for 1 epoch only, effectively limiting the number of full gradient evaluations to 1, we show that S2GD with $\nu = \mu$ needs

$$O(n + (\kappa/\varepsilon)\log(1/\varepsilon))$$

   work only (see Table 2). This compares favorably with the optimal complexity in the $\nu = \mu$ case (which reduces to SVRG), where the work needed is

$$O(n + \kappa/\varepsilon^2).$$

6. **Special cases.** GD and SVRG arise as special cases of S2GD, for $m = 1$ and $\nu = 0$, respectively.[4]

[4]While S2GD reduces to GD for $m = 1$, our *analysis* does not say anything meaningful in the $m = 1$ case - it is too coarse to cover this case. This is also the reason behind the empty space in the "Complexity" box column for GD in Table 2.

Parameters	**Method**	**Complexity**
$\nu = \mu$, $j = O(\log(\frac{1}{\varepsilon}))$ & $m = O(\kappa)$	Optimal S2GD	$O((n+\kappa)\log(\frac{1}{\varepsilon}))$
$m = 1$	GD	—
$\nu = 0$	SVRG [3]	$O((n+\kappa)\log(\frac{1}{\varepsilon}))$
$\nu = 0$, $j = 1$, $m = O(\frac{\kappa}{\varepsilon^2})$	Optimal SVRG with 1 epoch	$O(n + \frac{\kappa}{\varepsilon^2})$
$\nu = \mu$, $j = 1$, $m = O(\frac{\kappa}{\varepsilon}\log(\frac{1}{\varepsilon}))$	Optimal S2GD with 1 epoch	$O(n + \frac{\kappa}{\varepsilon}\log(\frac{1}{\varepsilon}))$

Table 2: Summary of complexity results and special cases. Condition number: $\kappa = L/\mu$ if f is μ-strongly convex and $\kappa = 2L/\varepsilon$ if f is *not* strongly convex and $\epsilon \leq L$.

7. **Low memory requirements.** Note that SDCA and SAG, unlike SVRG and S2GD, need to store all gradients f_i' (or dual variables) throughout the iterative process. While this may not be a problem for a modest sized optimization task, this requirement makes such methods less suitable for problems with very large n.

8. **S2GD+.** We propose a "boosted" version of S2GD, called S2GD+, which we do not analyze. In our experiments, however, it performs vastly superior to all other methods we tested, including GD, SGD, SAG and S2GD. S2GD alone is better than both GD and SGD if a highly accurate solution is required. The performance of S2GD and SAG is roughly comparable, even though in our experiments S2GD turned to have an edge.

4 Complexity Analysis: Strongly Convex Loss

For the purpose of the analysis, let

$$\mathcal{F}_{j,t} \overset{\text{def}}{=} \sigma(x_1, x_2, \dots, x_j; y_{j,1}, y_{j,2}, \dots, y_{j,t}) \tag{7}$$

be the σ-algebra generated by the relevant history of S2GD. We first isolate an auxiliary result.

Lemma 3. *Consider the S2GD algorithm. For any fixed epoch number j, the following identity holds:*

$$\mathbf{E}(f(x_{j+1})) = \frac{1}{\beta}\sum_{t=1}^{m}(1-\nu h)^{m-t}\mathbf{E}\left(f(y_{j,t-1})\right). \tag{8}$$

Proof. By the tower property of expectations and the definition of x_{j+1} in the algorithm, we obtain

$$\begin{aligned}\mathbf{E}(f(x_{j+1})) &= \mathbf{E}\left(\mathbf{E}(f(x_{j+1}) \mid \mathcal{F}_{j,m})\right) = \mathbf{E}\left(\sum_{t=1}^{m}\frac{(1-\nu h)^{m-t}}{\beta}f(y_{j,t-1})\right)\\ &= \frac{1}{\beta}\sum_{t=1}^{m}(1-\nu h)^{m-t}\mathbf{E}\left(f(y_{j,t-1})\right).\end{aligned}$$

□

We now state and prove the main result of this section.

Theorem 4. *Let Assumptions 1 and 2 be satisfied. Consider the S2GD algorithm applied to solving problem* (1). *Choose* $0 \leq \nu \leq \mu$, $0 < h < \frac{1}{2L}$, *and let* m *be sufficiently large so that*

$$c \overset{def}{=} \frac{(1-\nu h)^m}{\beta \mu h (1-2Lh)} + \frac{2(L-\mu)h}{1-2Lh} < 1. \tag{9}$$

Then we have the following convergence in expectation:

$$\mathbf{E}\left(f(x_j) - f(x_*)\right) \leq c^j (f(x_0) - f(x_*)). \tag{10}$$

Before we proceed to proving the theorem, note that in the special case with $\nu = 0$, we recover the result of Johnson and Zhang [3] (with a minor improvement in the second term of c where L is replaced by $L - \mu$), namely

$$c = \frac{1}{\mu h(1-2Lh)m} + \frac{2(L-\mu)h}{1-2Lh}. \tag{11}$$

If we set $\nu = \mu$, then c can be written in the form

$$c = \frac{(1-\mu h)^m}{(1-(1-\mu h)^m)(1-2Lh)} + \frac{2(L-\mu)h}{1-2Lh}. \tag{12}$$

Clearly, the latter c is a major improvement on the former one. We shall elaborate on this further later.

Proof. It is well-known [8, Theorem 2.1.5] that since the functions f_i are L-smooth, they necessarily satisfy the following inequality:

$$\|f_i'(x) - f_i'(x_*)\|^2 \leq 2L\left[f_i(x) - f_i(x_*) - \langle f_i'(x_*), x - x_*\rangle\right].$$

By summing these inequalities for $i = 1, \ldots, n$, and using $f'(x_*) = 0$, we get

$$\frac{1}{n}\sum_{i=1}^{n} \|f_i'(x) - f_i'(x_*)\|^2 \leq 2L\left[f(x) - f(x_*) - \langle f'(x_*), x - x_*\rangle\right] = 2L(f(x) - f(x_*)). \tag{13}$$

Let $G_{j,t} \overset{\text{def}}{=} g_j + f_i'(y_{j,t-1}) - f_i'(x_j)$ be the direction of update at j^{th} iteration in the outer loop and t^{th} iteration in the inner loop. Taking expectation with respect to i, conditioned on the σ-algebra $\mathcal{F}_{j,t-1}$ (7), we obtain[5]

$$\begin{aligned}
\mathbf{E}\left(\|G_{j,t}\|^2\right) &= \mathbf{E}\left(\|f_i'(y_{j,t-1}) - f_i'(x_*) - f_i'(x_j) + f_i'(x_*) + g_j\|^2\right) \\
&\leq 2\mathbf{E}\left(\|f_i'(y_{j,t-1}) - f_i'(x_*)\|^2\right) + 2\mathbf{E}\left(\|\left[f_i'(x_j) - f_i'(x_*)\right] - f'(x_j)\|^2\right) \\
&= 2\mathbf{E}\left(\|f_i'(y_{j,t-1}) - f_i'(x_*)\|^2\right) \\
&\quad + 2\mathbf{E}\left(\|f_i'(x_j) - f_i'(x_*)\|^2\right) - 4\mathbf{E}\left(\langle f'(x_j), f_i'(x_j) - f_i'(x_*)\rangle\right) + 2\|f'(x_j)\|^2 \\
&= 4L\left[f(y_{j,t-1}) - f(x_*) + f(x_j) - f(x_*)\right] - 2\|f'(x_j)\|^2 - 4\langle f'(x_j), f'(x_*)\rangle \\
&\overset{(3)+(13)}{\leq} 4L\left[f(y_{j,t-1}) - f(x_*)\right] + 4(L-\mu)\left[f(x_j) - f(x_*)\right].
\end{aligned} \tag{14}$$

[5] For simplicity, we supress the $\mathbf{E}(\cdot \mid \mathcal{F}_{j,t-1})$ notation here.

Above we have used the bound $\|x' + x''\|^2 \leq 2\|x'\|^2 + 2\|x''\|^2$ and the fact that

$$\mathbf{E}(G_{j,t} \mid \mathcal{F}_{j,t-1}) = f'(y_{j,t-1}). \tag{15}$$

We now study the expected distance to the optimal solution (a standard approach in the analysis of gradient methods):

$$\begin{aligned}
\mathbf{E}(\|y_{j,t} - x_*\|^2 \mid \mathcal{F}_{j,t-1}) \quad &= \quad \|y_{j,t-1} - x_*\|^2 - 2h\langle \mathbf{E}(G_{j,t} \mid \mathcal{F}_{j,t-1}), y_{j,t-1} - x_* \rangle \\
&\qquad\quad + h^2 \mathbf{E}(\|G_{j,t}\|^2 \mid \mathcal{F}_{j,t-1}) \\
&\overset{(14)+(15)}{\leq} \quad \|y_{j,t-1} - x_*\|^2 - 2h\langle f'(y_{j,t-1}), y_{j,t-1} - x_* \rangle \\
&\qquad\quad + 4Lh^2\left[f(y_{j,t-1}) - f(x_*)\right] + 4(L-\mu)h^2\left[f(x_j) - f(x_*)\right] \\
&\overset{(3)}{\leq} \quad \|y_{j,t-1} - x_*\|^2 - 2h\left[f(y_{j,t-1}) - f(x_*)\right] - \nu h\|y_{j,t-1} - x_*\|^2 \\
&\qquad\quad + 4Lh^2\left[f(y_{j,t-1}) - f(x_*)\right] + 4(L-\mu)h^2\left[f(x_j) - f(x_*)\right] \\
&= \quad (1 - \nu h)\|y_{j,t-1} - x_*\|^2 - 2h(1 - 2Lh)[f(y_{j,t-1}) - f(x_*)] \\
&\qquad\quad + 4(L-\mu)h^2[f(x_j) - f(x_*)].
\end{aligned} \tag{16}$$

By rearranging the terms in (16) and taking expectation over the σ-algebra $\mathcal{F}_{j,t-1}$, we get the following inequality:

$$\begin{aligned}
&\mathbf{E}(\|y_{j,t} - x_*\|^2) + 2h(1 - 2Lh)\mathbf{E}(f(y_{j,t-1}) - f(x_*)) \\
&\qquad \leq (1 - \nu h)\mathbf{E}(\|y_{j,t-1} - x_*\|^2) + 4(L-\mu)h^2\mathbf{E}(f(x_j) - f(x_*)).
\end{aligned} \tag{17}$$

Finally, we can analyze what happens after one iteration of the outer loop of S2GD, i.e., between two computations of the full gradient. By summing up inequalities (17) for $t = 1, \ldots, m$, with inequality t multiplied by $(1 - \nu h)^{m-t}$, we get the left-hand side

$$\begin{aligned}
LHS &= \mathbf{E}(\|y_{j,m} - x_*\|^2) + 2h(1 - 2Lh)\sum_{t=1}^{m}(1 - \nu h)^{m-t}\mathbf{E}(f(y_{j,t-1}) - f(x_*)) \\
&\overset{(8)}{=} \mathbf{E}(\|y_{j,m} - x_*\|^2) + 2\beta h(1 - 2Lh)\mathbf{E}(f(x_{j+1}) - f(x_*)),
\end{aligned}$$

and the right-hand side

$$\begin{aligned}
RHS &= (1 - \nu h)^m \mathbf{E}(\|x_j - x_*\|^2) + 4\beta(L-\mu)h^2\mathbf{E}(f(x_j) - f(x_*)) \\
&\overset{(3)}{\leq} \frac{2(1 - \nu h)^m}{\mu}\mathbf{E}(f(x_j) - f(x_*)) + 4\beta(L-\mu)h^2\mathbf{E}(f(x_j) - f(x_*)) \\
&= 2\left(\frac{(1 - \nu h)^m}{\mu} + 2\beta(L-\mu)h^2\right)\mathbf{E}(f(x_j) - f(x_*)).
\end{aligned}$$

Since $LHS \leq RHS$, we finally conclude with

$$\mathbf{E}(f(x_{j+1}) - f(x_*)) \quad \leq \quad c\mathbf{E}(f(x_j) - f(x_*)) - \frac{\mathbf{E}(\|y_{j,m} - x_*\|^2)}{2\beta h(1 - 2Lh)} \quad \leq \quad c\mathbf{E}(f(x_j) - f(x_*)).$$

$\square$

Since we have established linear convergence of expected values, a high probability result can be obtained in a straightforward way using Markov inequality.

Theorem 5. *Consider the setting of Theorem 4. Then, for any* $0 < \rho < 1$, $0 < \varepsilon < 1$ *and*

$$s \geq \frac{\log\left(\frac{1}{\varepsilon\rho}\right)}{\log\left(\frac{1}{c}\right)}, \tag{18}$$

we have

$$\mathbf{P}\left(\frac{f(x_j) - f(x_*)}{f(x_0) - f(x_*)} \leq \varepsilon\right) \geq 1 - \rho. \tag{19}$$

Proof. This follows directly from Markov inequality and Theorem 4:

$$\mathbf{P}(f(x_j) - f(x_*) > \varepsilon(f(x_0) - f(x_*)) \overset{(10)}{\leq} \frac{\mathbf{E}(f(x_j) - f(x_*))}{\varepsilon(f(x_0) - f(x_*))} \leq \frac{c^s}{\varepsilon} \overset{(18)}{\leq} \rho$$

□

This result will be also useful when treating the non-strongly convex case.

5 Optimal Choice of Parameters

The goal of this section is to provide insight into the choice of parameters of S2GD; that is, the number of epochs (equivalently, full gradient evaluations) j, the maximal number of steps in each epoch m, and the stepsize h. The remaining parameters (L, μ, n) are inherent in the problem and we will hence treat them in this section as given.

In particular, ideally we wish to find parameters j, m and h solving the following optimization problem:

$$\min_{j,m,h} \quad \tilde{\mathcal{W}}(j, m, h) \overset{\text{def}}{=} j(n + 2\xi(m, h)), \tag{20}$$

subject to

$$\mathbf{E}(f(x_j) - f(x_*)) \leq \varepsilon(f(x_0) - f(x_*)). \tag{21}$$

Note that $\tilde{\mathcal{W}}(j, m, h)$ is the *expected work*, measured by the number number of stochastic gradient evaluations, performed by S2GD when running for j epochs. Indeed, the evaluation of g_j is equivalent to n stochastic gradient evaluations, and each epoch further computes on average $2\xi(m, h)$ stochastic gradients (see (5)). Since $\frac{m+1}{2} \leq \xi(m, h) < m$, we can simplify and solve the problem with ξ set to the conservative upper estimate $\xi = m$.

In view of (10), accuracy constraint (21) is satisfied if c (which depends on h and m) and j satisfy

$$c^j \leq \varepsilon. \tag{22}$$

We therefore instead consider the parameter fine-tuning problem

$$\min_{j,m,h} \mathcal{W}(j, m, h) \overset{\text{def}}{=} j(n + 2m) \qquad \text{subject to} \qquad c \leq \varepsilon^{1/j}. \tag{23}$$

In the following we (approximately) solve this problem in two steps. First, we fix j and find (nearly) optimal $h = h(j)$ and $m = m(j)$. The problem reduces to minimizing m subject to $c \le \varepsilon^{1/j}$ by fine-tuning h. While in the $\nu = 0$ case it is possible to obtain closed form solution, this is not possible for $\nu > \mu$. However, it is still possible to obtain a formula for $h(j)$ leading to $m(j)$ which depends on ε in the correct way. We then plug the formula for $m(j)$ obtained this way back into (23), and study the quantity $\mathcal{W}(j, m(j), h(j)) = j(n + 2m(j))$ as a function of j.

Theorem 6 (Choice of parameters)**.** *Fix the number of epochs $j \ge 1$, error tolerance $0 < \varepsilon < 1$, and let $\Delta = \varepsilon^{1/j}$. If we run S2GD with the stepsize*

$$h = h(j) \overset{def}{=} \frac{1}{\frac{4}{\Delta}(L-\mu) + 2L} \tag{24}$$

and

$$m \ge m(j) \overset{def}{=} \begin{cases} \left(\frac{4(\kappa-1)}{\Delta} + 2\kappa\right) \log\left(\frac{2}{\Delta} + \frac{2\kappa-1}{\kappa-1}\right), & \textit{if} \quad \nu = \mu, \\ \frac{8(\kappa-1)}{\Delta^2} + \frac{8\kappa}{\Delta} + \frac{2\kappa^2}{\kappa-1}, & \textit{if} \quad \nu = 0, \end{cases} \tag{25}$$

then $\mathbf{E}(f(x_j) - f(x_*)) \le \varepsilon(f(x_0) - f(x_*))$.

In particular, if we choose $j^* = \lceil \log(1/\varepsilon) \rceil$, then $\frac{1}{\Delta} \le \exp(1)$, and hence $m(j^*) = O(\kappa)$, leading to the workload

$$\mathcal{W}(j^*, m(j^*), h(j^*)) = \lceil \log\left(\tfrac{1}{\varepsilon}\right) \rceil (n + O(\kappa)) = O\left((n+\kappa)\log\left(\tfrac{1}{\varepsilon}\right)\right). \tag{26}$$

Proof. We only need to show that $c \le \Delta$, where c is given by (12) for $\nu = \mu$ and by (11) for $\nu = 0$. The stepsize h is chosen so that

$$c_2 \overset{\text{def}}{=} \frac{2(L-\mu)h}{1-2Lh} = \frac{\Delta}{2},$$

and hence it only remains to verify that $c - c_2 \le \frac{\Delta}{2}$. In the $\nu = 0$ case, $m(j)$ is chosen so that $c - c_2 = \frac{\Delta}{2}$. In the $\nu = \mu$ case, $c - c_2 = \frac{\Delta}{2}$ holds for $m = \log\left(\frac{2}{\Delta} + \frac{2\kappa-1}{\kappa-1}\right) / \log\left(\frac{1}{1-H}\right)$, where $H = \left(\frac{4(\kappa-1)}{\Delta} + 2\kappa\right)^{-1}$. We only need to observe that c decreases as m increases, and apply the inequality $\log\left(\frac{1}{1-H}\right) \ge H$.

□

We now comment on the above result:

1. **Workload.** Notice that for the choice of parameters j^*, $h = h(j^*)$, $m = m(j^*)$ and any $\nu \in [0, \mu]$, the method needs $O(n \log(1/\varepsilon))$ computations of the full gradient (note this is independent of κ), and $O(\kappa \log(1/\varepsilon))$ computations of the stochastic gradient. This result, and special cases thereof, are summarized in Table 2.

2. **Simpler formulas for m.** If $\kappa \ge 2$, we can instead of (25) use the following (slightly worse but) simpler expressions for $m(j)$, obtained from (25) by using the bounds $1 \le \kappa - 1$, $\kappa - 1 \le \kappa$ and $\Delta < 1$ in appropriate places (e.g., $\frac{8\kappa}{\Delta} < \frac{8\kappa}{\Delta^2}$, $\frac{\kappa}{\kappa-1} \le 2 < \frac{2}{\Delta^2}$):

$$m \ge \tilde{m}(j) \overset{\text{def}}{=} \begin{cases} \frac{6\kappa}{\Delta} \log\left(\frac{5}{\Delta}\right), & \text{if} \quad \nu = \mu, \\ \frac{20\kappa}{\Delta^2}, & \text{if} \quad \nu = 0. \end{cases} \tag{27}$$

3. **Optimal stepsize in the $\nu = 0$ case.** Theorem 6 does not claim to have solved problem (23); the problem in general does not have a closed form solution. However, in the $\nu = 0$ case a closed-form formula can easily be obtained:

$$h(j) = \frac{1}{\frac{4}{\Delta}(L-\mu)+4L}, \qquad m \geq m(j) \overset{\text{def}}{=} \frac{8(\kappa-1)}{\Delta^2} + \frac{8\kappa}{\Delta}. \tag{28}$$

   Indeed, for fixed j, (23) is equivalent to finding h that minimizes m subject to the constraint $c \leq \Delta$. In view of (11), this is equivalent to searching for $h > 0$ maximizing the quadratic $h \to h(\Delta - 2(\Delta L + L - \mu)h)$, which leads to (28).

   Note that both the stepsize $h(j)$ and the resulting $m(j)$ are slightly larger in Theorem 6 than in (28). This is because in the theorem the stepsize was for simplicity chosen to satisfy $c_2 = \frac{\Delta}{2}$, and hence is (slightly) suboptimal. Nevertheless, the dependence of $m(j)$ on Δ is of the correct (optimal) order in both cases. That is, $m(j) = O\left(\frac{\kappa}{\Delta}\log(\frac{1}{\Delta})\right)$ for $\nu = \mu$ and $m(j) = O\left(\frac{\kappa}{\Delta^2}\right)$ for $\nu = 0$.

4. **Stepsize choice.** In cases when one does not have a good estimate of the strong convexity constant μ to determine the stepsize via (24), one may choose suboptimal stepsize that does not depend on μ and derive similar results to those above. For instance, one may choose $h = \frac{\Delta}{6L}$.

In Table 3 we provide comparison of work needed for small values of j, and different values of κ and ε. Note, for instance, that for any problem with $n = 10^9$ and $\kappa = 10^3$, S2GD outputs a highly accurate solution ($\varepsilon = 10^{-6}$) in the amount of work equivalent to 2.12 evaluations of the full gradient of f!

6 Complexity Analysis: Convex Loss

If f is convex but not strongly convex, we define $\hat{f}_i(x) \overset{\text{def}}{=} f_i(x) + \frac{\mu}{2}\|x - x_0\|^2$, for small enough $\mu > 0$ (we shall see below how the choice of μ affects the results), and consider the perturbed problem

$$\min_{x \in \mathbb{R}^d} \hat{f}(x), \tag{29}$$

where

$$\hat{f}(x) \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \hat{f}_i(x) = f(x) + \frac{\mu}{2}\|x - x_0\|^2. \tag{30}$$

Note that $\hat{f}$ is μ-strongly convex and $(L+\mu)$-smooth. In particular, the theory developed in the previous section applies. We propose that S2GD be instead applied to the perturbed problem, and show that an approximate solution of (29) is also an approximate solution of (1) (we will assume that this problem has a minimizer).

Let $\hat{x}_*$ be the (necessarily unique) solution of the perturbed problem (29). The following result describes an important connection between the original problem and the perturbed problem.

Lemma 7. *If $\hat{x} \in \mathbb{R}^d$ satisfies $\hat{f}(\hat{x}) \leq \hat{f}(\hat{x}_*) + \delta$, where $\delta > 0$, then*

$$f(\hat{x}) \leq f(x_*) + \frac{\mu}{2}\|x_0 - x_*\|^2 + \delta.$$

	$\varepsilon = 10^{-3}, \kappa = 10^3$	
j	$\mathcal{W}_\mu(j)$	$\mathcal{W}_0(j)$
1	**1.06n**	$17.0n$
2	$2.00n$	**2.03n**
3	$3.00n$	$3.00n$
4	$4.00n$	$4.00n$
5	$5.00n$	$5.00n$

	$\varepsilon = 10^{-6}, \kappa = 10^3$	
j	$\mathcal{W}_\mu(j)$	$\mathcal{W}_0(j)$
1	$116n$	10^7n
2	**2.12n**	$34.0n$
3	$3.01n$	**3.48n**
4	$4.00n$	$4.06n$
5	$5.00n$	$5.02n$

	$\varepsilon = 10^{-9}, \kappa = 10^3$	
j	$\mathcal{W}_\mu(j)$	$\mathcal{W}_0(j)$
2	$7.58n$	10^4n
3	**3.18n**	$51.0n$
4	$4.03n$	$6.03n$
5	$5.01n$	**5.32n**
6	$6.00n$	$6.09n$

	$\varepsilon = 10^{-3}, \kappa = 10^6$	
j	$\mathcal{W}_\mu(j)$	$\mathcal{W}_0(j)$
2	$4.14n$	$35.0n$
3	**3.77n**	$8.29n$
4	$4.50n$	**6.39n**
5	$5.41n$	$6.60n$
6	$6.37n$	$7.28n$

	$\varepsilon = 10^{-6}, \kappa = 10^6$	
j	$\mathcal{W}_\mu(j)$	$\mathcal{W}_0(j)$
4	$8.29n$	$70.0n$
5	**7.30n**	$26.3n$
6	$7.55n$	$16.5n$
8	$9.01n$	**12.7n**
10	$10.8n$	$13.2n$

	$\varepsilon = 10^{-9}, \kappa = 10^6$	
j	$\mathcal{W}_\mu(j)$	$\mathcal{W}_0(j)$
5	$17.3n$	$328n$
8	**10.9n**	$32.5n$
10	$11.9n$	$21.4n$
13	$14.3n$	**19.1n**
20	$21.0n$	$23.5n$

	$\varepsilon = 10^{-3}, \kappa = 10^9$	
j	$\mathcal{W}_\mu(j)$	$\mathcal{W}_0(j)$
6	$378n$	$1293n$
8	**358n**	$1063n$
11	$376n$	**1002n**
15	$426n$	$1058n$
20	$501n$	$1190n$

	$\varepsilon = 10^{-6}, \kappa = 10^9$	
j	$\mathcal{W}_\mu(j)$	$\mathcal{W}_0(j)$
13	$737n$	$2409n$
16	**717n**	$2126n$
19	$727n$	$2025n$
22	$752n$	**2005n**
30	$852n$	$2116n$

	$\varepsilon = 10^{-9}, \kappa = 10^9$	
j	$\mathcal{W}_\mu(j)$	$\mathcal{W}_0(j)$
15	$1251n$	$4834n$
24	**1076n**	$3189n$
30	$1102n$	$3018n$
32	$1119n$	**3008n**
40	$1210n$	$3078n$

Table 3: Comparison of work sufficient to solve a problem with $n = 10^9$, and various values of κ and ε. The work was computed using formula (23), with $m(j)$ as in (27). The notation $\mathcal{W}_\nu(j)$ indicates what parameter ν was used.

Proof. The statement is almost identical to Lemma 9 in [9]; its proof follows the same steps with only minor adjustments. □

We are now ready to establish a complexity result for non-strongly convex losses.

Theorem 8. *Let Assumption 1 be satisfied. Choose $\mu > 0$, $0 \le \nu \le \mu$, stepsize $0 < h < \frac{1}{2(L+\mu)}$, and let m be sufficiently large so that*

$$\hat{c} \overset{def}{=} \frac{(1-\nu h)^m}{\beta \mu h (1 - 2(L+\mu)h)} + \frac{2Lh}{1-2(L+\mu)h} < 1. \tag{31}$$

Pick $x_0 \in \mathbb{R}^d$ and let $\hat{x}_0 = x_0, \hat{x}_1, \ldots, \hat{x}_j$ be the sequence of iterates produced by S2GD as applied to problem (29). Then, for any $0 < \rho < 1$, $0 < \varepsilon < 1$ and

$$j \ge \frac{\log(1/(\varepsilon\rho))}{\log(1/\hat{c})}, \tag{32}$$

we have

$$\mathbf{P}\left(f(\hat{x}_j) - f(x_*) \leq \varepsilon(f(x_0) - f(x_*)) + \frac{\mu}{2}\|x_0 - x_*\|^2\right) \geq 1 - \rho. \tag{33}$$

In particular, if we choose $\mu = \epsilon < L$ and parameters j^*, $h(j^*)$, $m(j^*)$ as in Theorem 6, the amount of work performed by S2GD to guarantee (33) is

$$\mathcal{W}(j^*, h(j^*), m(j^*)) = O\left((n + \tfrac{L}{\varepsilon})\log(\tfrac{1}{\varepsilon})\right),$$

which consists of $O(\frac{1}{\varepsilon})$ full gradient evaluations and $O(\frac{L}{\epsilon}\log(\frac{1}{\varepsilon}))$ stochastic gradient evaluations.

Proof. We first note that

$$\hat{f}(\hat{x}_0) - \hat{f}(\hat{x}_*) \overset{(30)}{=} f(\hat{x}_0) - \hat{f}(\hat{x}_*) \leq f(\hat{x}_0) - f(\hat{x}_*) \leq f(x_0) - f(x_*), \tag{34}$$

where the first inequality follows from $f \leq \hat{f}$, and the second one from optimality of x_*. Hence, by first applying Lemma 7 with $\hat{x} = \hat{x}_j$ and $\delta = \varepsilon(f(x_0) - f(x_*))$, and then Theorem 5, with $c \leftarrow \hat{c}$, $f \leftarrow \hat{f}$, $x_0 \leftarrow \hat{x}_0$, $x_* \leftarrow \hat{x}_*$, we obtain

$$\begin{aligned}\mathbf{P}\left(f(\hat{x}_j) - f(x_*) \leq \delta + \frac{\mu}{2}\|x_0 - x_*\|^2\right) &\overset{\text{(Lemma 7)}}{\geq} \mathbf{P}\left(\hat{f}(\hat{x}_j) - \hat{f}(\hat{x}_*) \leq \delta\right) \\ &\overset{(34)}{\geq} \mathbf{P}\left(\frac{\hat{f}(\hat{x}_j) - \hat{f}(\hat{x}_*)}{\hat{f}(\hat{x}_0) - \hat{f}(\hat{x}_*)} \leq \varepsilon\right) \overset{(19)}{\geq} 1 - \rho.\end{aligned}$$

The second statement follows directly from the second part of Theorem 6 and the fact that the condition number of the perturbed problem is $\kappa = \frac{L+\epsilon}{\epsilon} \leq \frac{2L}{\epsilon}$. □

7 Numerical Experiments

In this section we conduct computational experiments to illustrate some aspects of the performance of our algorithm. In Section 7.1 we consider L2-regularized logistic regression with the MNIST[6] dataset, consisting of hand-written digits. It is sufficient for our purposes to perform classification on the "5 vs 8" case only. In Section 7.2 we consider L2-regularized least squares with artificial data. All experiments were performed in MATLAB.

7.1 L2-regularized logistic loss

Consider labeled training data $(a_i, y_i) \in \mathbb{R}^d \times \{0, 1\}$, for $i = 1, 2, \ldots, n$, where a_i are examples and y_i labels. The L2-regularized logistic loss for the ith training example is defined as

$$f_i(x) = -\left[y_i \log\left(\frac{1}{1 + \exp(-x^T a_i)}\right) + (1 - y_i)\log\left(1 - \frac{1}{1 + \exp(-x^T a_i)}\right)\right] + \frac{\lambda}{2}\|x\|^2,$$

where $\lambda > 0$ is a regularization parameter.

The MNIST dataset consists of $n = 11,272$ training examples, each being a $d = 785$ dimensional vector. We set $\lambda = 0.05$ and, using this as an estimate of μ, results in a well conditioned problem with $\kappa \approx 320$.

[6]http://yann.lecun.com/exdb/mnist/

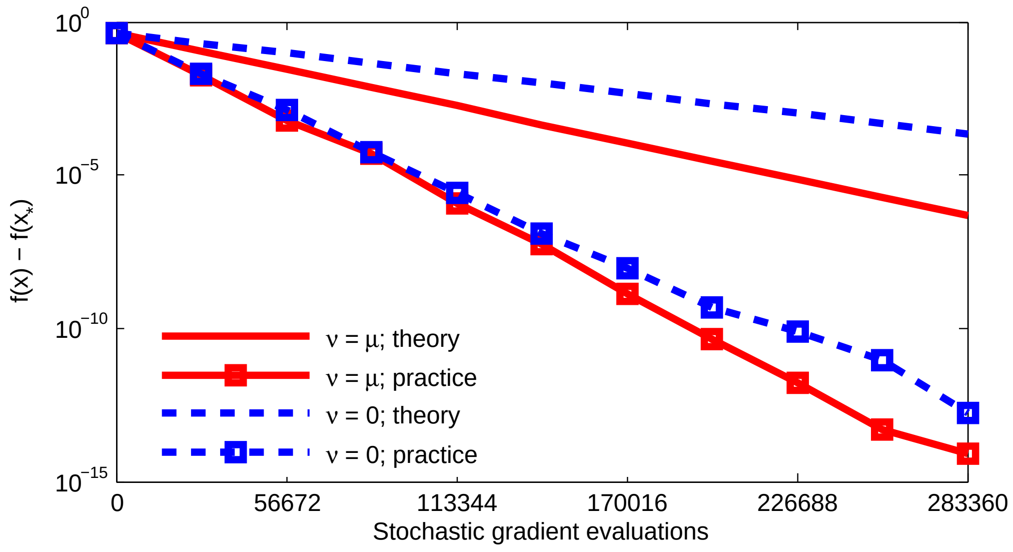

Figure 1: Logistic regression on MNIST dataset. Comparison of theoretical result and practical performance for cases $\nu = \mu$ (full line) and $\nu = 0$ (dashed line). Each marker represents state after a whole epoch.

In Figure 1 we present a comparison of our theory and the practical performance of S2GD for two choices of the parameter ν: $\nu = \mu$ and $\nu = 0$. Recall that the latter choice is essentially the SVRG method of Johnson and Zhang [3].

The figure demonstrates linear convergence of S2GD in practice, with the convergence rate being significantly better than the already strong theoretical result. We can observe surprising convergence to machine precision in work equivalent to evaluating full gradient only 25 times. This is property standard SGD does not possess (as we show later) and for problems of certain size is impossible with gradient descent, since we cannot compute enough gradients to reach such a precision. Our method is also an improvement over [3], both in theory and practice.

Note that we have implemented the methods following the theory, without any engineering tricks. Hence, further improvement is possible.

We chose the parameters h and m of the method numerically in order to minimize total work $\mathcal{W}$. The values are $m = 11,686$ and $h = 1/14.1L$ for $\nu = \lambda$, resulting in $c = 0.2548$, and $m = 17,062$, $h = 1/13.9L$ in the case $\nu = 0$, resulting in $c = 0.4716$. Note that the theoretical decrease factor c is much better for $\nu = \mu$ than in the SVRG case ($\nu = 0$). Indeed, the predicted drop in the residual is doubled, while at the same time only about two thirds of the stochastic gradient computations required by SVRG are needed.

Note that the theoretical decrease factor c is much better for $\nu = \mu$ than in the SVRG case ($\nu = 0$). Indeed, the predicted drop in the residual is doubled, while at the same time only about two thirds of the stochastic gradient computations required by SVRG are needed.

We include these in the same plot since we obtained the values as average of several hundreds of runs, and the average number of stochastic updates is almost the same in both cases. Note that due to this fact, the practical performance is essentially the same, perhaps with bigger variance with $\nu = 0$. However, the important difference is in the theoretical bound, matching the theoretical insight from Table 3, saying that for $\kappa \ll n$, the number of full gradient evaluations needed in case $\nu = 0$ is about twice as big as in case $\nu = \mu$.

7.2 L2-regularized least squares

Figure 2 presents a comparison of the theoretical rate and practical performance on a larger problem with artificial data, with a condition number we can control (and choose it to be poor). In particular, we consider the L2-regularized least squares with

$$f_i(x) = (a_i^T x - b_i)^2 + \frac{\lambda}{2}\|x\|^2,$$

for some $a_i \in \mathbb{R}^d$, $b_i \in \mathbb{R}$ and $\lambda > 0$ is the regularization parameter.

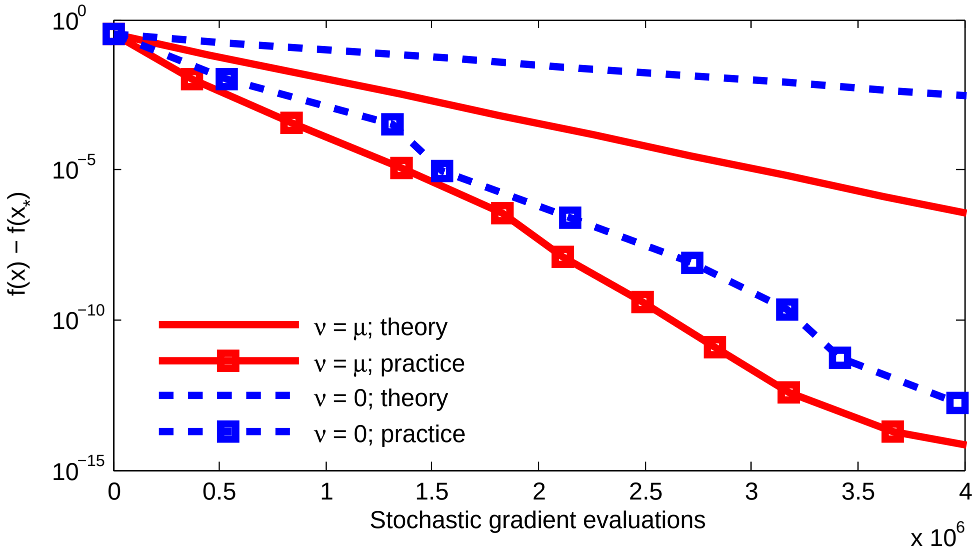

Figure 2: Least squares with $n = 10^5$, $\kappa = 10^4$. Comparison of theoretical result and practical performance for cases $\nu = \mu$ (full red line) and $\nu = 0$ (dashed blue line).

We consider an instance with $n = 100,000$, $d = 1,000$ and $\kappa = 10,000$. We run the algorithm with both parameters $\nu = 0$ and $\nu = \lambda$ — our best estimate of μ. Again, we chose parameters m and h as a (numerical) solution of the work-minimization problem (20), obtaining $m = 261,063$ and $h = 1/11.4L$ for $\nu = \lambda$ and $m = 426,660$ and $h = 1/12.7L$ for $\nu = 0$. The practical performance is obtained after single run of the S2GD algorithm. The experiments demonstrate a relationship between theory and practice similar to that in the previous section: Figure 2 confirms consistency of the results obtained in Figure 1.

7.3 SGD, S2GD, S2GD+ and SAG

Here we compare the practical performance of SGD, S2GD, S2GD+ and SAG on a larger, sparse (90% zeros) problem with $n = 10^6$, $\kappa = 10^5$ and $d = 200$; see Figure 3. The size limitations are due to us running the experiments in MATLAB on a notebook. As before, we set the parameters m and h numerically in an optimal way.

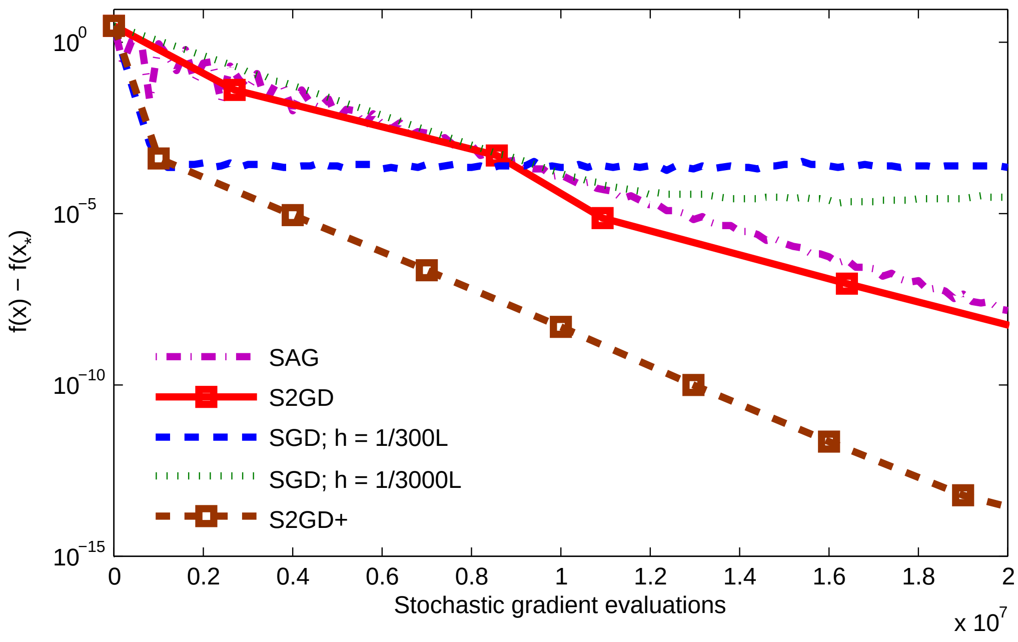

Figure 3: Least squares with $n = 10^6$, $\kappa = 10^5$. Comparison of S2GD, SGD (with different step sizes), SAG and S2GD+ (Algorithm 2). Note that SAG and S2GD have similar performance, with S2GD slightly outperforming SAG. However, S2GD+ is vastly superior to all.

We can see, that S2GD and SAG, run with parameters as analyzed in theory, exhibit essentially the same performance. Both methods can benefit from engineering and implementation tricks, which we do not explore in this work.

The performance of SGD with relatively big step size is better in the first few iterations (1 pass over the data), but quickly slows down and oscillates at low precision – this is due to the inherent variance of the stochastic gradient. If we aim for a higher precision, one would need to decrease the step size and thus make SGD slower than S2GD or SAG. The advantage of S2GD and SAG is linear convergence at all distances from the solution. S2GD is slower than SGD in the beginning, because it has to compute the full gradient before actually making a single step.

Allowing deviations from theory, performance *significantly superior* to all other methods is offered by Algorithm 2 (S2GD+), where we first run SGD for a single pass through the data, to find a good solution quickly, and then continue with S2GD with constant size of the inner loop $m = n$. Note that we evaluate function value in S2GD only after each epoch, as illustrated with markers in the figures.

8 Conclusion

We have developed a new semi-stochastic gradient descent method (S2GD) and analyzed its complexity for smooth convex and strongly convex loss functions. Our methods need $O((\kappa/n)\log(1/\varepsilon))$ work only, measured in units equivalent to the evaluation of the full gradient of the loss function, where $\kappa = L/\mu$ if the loss is L-smooth and μ-strongly convex, and $\kappa \leq 2L/\varepsilon$ if the loss is merely L-smooth.

To the best of our knowledge, stochastic gradient descent methods have not been previously shown to exhibit comparable convergence guarantees for non-strongly convex functions. Our results in the strongly convex case match or improve on a few very recent results, while at the same time generalizing and simplifying the analysis.

Additionally, we proposed S2GD+ —a method which equips S2GD with an SGD pre-processing step—which in our experiments exhibits superior performance to all methods we tested. We leave the analysis of this method as an open problem.

References

[1] Olivier Fercoq and Peter Richtárik. Smooth minimization of nonsmooth functions with parallel coordinate descent methods. *arXiv:1309.5885*, 2013.

[2] C-J. Hsieh, K-W. Chang, C-J. Lin, S.S. Keerthi, , and S. Sundarajan. A dual coordinate descent method for large-scale linear SVM. In *ICML*, 2008.

[3] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, 2013.

[4] Mehrdad Mahdavi and Rong Jin. Mixedgrad: An $o(1/t)$ convergence rate algorithm for stochastic smooth optimization. *arXiv:1307.7192v1*, 2013.

[5] Kurt Marti and E. Fuchs. On solutions of stochastic programming problems by descent procedures with stochastic and deterministic directions. *Methods of Operations Research*, 33:281–293, 1979.

[6] Kurt Marti and E. Fuchs. Rates of convergence of semi-stochastic approximation procedures for solving stochastic optimization problems. *Optimization*, 17(2):243–265, 1986.

[7] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[8] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer, 2004.

[9] Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, pages 1–38, 2012.

[10] Peter Richtárik and Martin Takáč. Parallel coordinate descent methods for big data optimization. *arXiv:1212.0873*, 2012.

[11] Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *arXiv:1310.2059*, 2013.

[12] Nicolas Le Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. *arXiv:1202.6258*, 2012.

[13] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *arXiv:1309.2388*, 2013.

[14] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *arXiv:1209.1873*, 2012.

[15] Martin Takáč, Avleen Bijral, Peter Richtárik, and Nati Srebro. Mini-batch primal and dual methods for SVMs. In *ICML*, 2013.

[16] Lijun Zhang, Mehrdad Mahdavi, and Rong Jin. Linear convergence with condition number independent access of full gradients. *NIPS*, 2013.

[17] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML*, 2004.