Semi-Stochastic Gradient Descent Methods

Centre for Numerical Algorithms and Intelligent Software

Jakub Konečný   Peter Richtárik

University of Edinburgh

Simons Institute for the Theory of Computing

1. Introduction

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

$$\min_{x\in\mathbb{R}^d} f(x), \qquad \text{where} \qquad f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x). \tag{P}$$

Assumption 1. The functions $f_1, f_2, \dots, f_n$ have Lipschitz continuous gradients with constant $L > 0$. That is, for all $x, z \in \mathbb{R}^d$ we have

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

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

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

There are two basic approaches to solving this problem. In first place, **Gradient Descent** (GD) iteration sets

$$x_{j+1} = x_j - hf'(x_j),$$

where h is a stepsize parameter and $f'(x_j)$ is the gradient of f at x_j.
If n is large, it is prohibitive to evaluate full gradient at each iteration. **Stochastic Gradient Descent** (SGD) picks $i \in \{1, 2, \dots, n\}$ uniformly at random, and sets

$$x_{j+1} = x_j - hf_i'(x_j).$$

SGD drastically reduces the amount of work that needs to be done in each iteration (by factor of n), but for fixed step size converges only to certain neighbourhood of optimal solution. GD enjoys linear convergence, but iteration complexity depends on n.

The aim of this work is to provide an algorithm for solving (P), which has linear rate of convergence, but retains the work efficiency of SGD.

2. The Algorithm (S2GD)

In the S2GD algorithm, we compute full gradient (g_j) once, followed by a random number of updates, where we use two stochastic gradients in each of them.

Algorithm (S2GD)

parameters: $m = \max$ # of stochastic steps per epoch; h = stepsize; ν = lower bound on μ; initial point x_0

for $j = 0, 1, 2, \dots$ **do**

  $g_j \leftarrow \frac{1}{n}\sum_{i=1}^{n} f_i'(x_j)$ ▷ Compute full gradient

  $y_{j,0} \leftarrow x_j$

  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** ▷ $\beta = \sum_{t=1}^{m}(1-\nu h)^{m-t}$

    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

3. Rate of convergence

Theorem 1

Consider the S2GD algorithm applied to solving problem (P). *Choose* $0 \le \nu \le \mu$, $0 < h < 1/2L$, *and let* m *be sufficiently large so that*

$$c := \frac{(1-\nu h)^m}{\beta\mu h(1-2Lh)} + \frac{2(L-\mu)h}{1-2Lh} < 1,$$

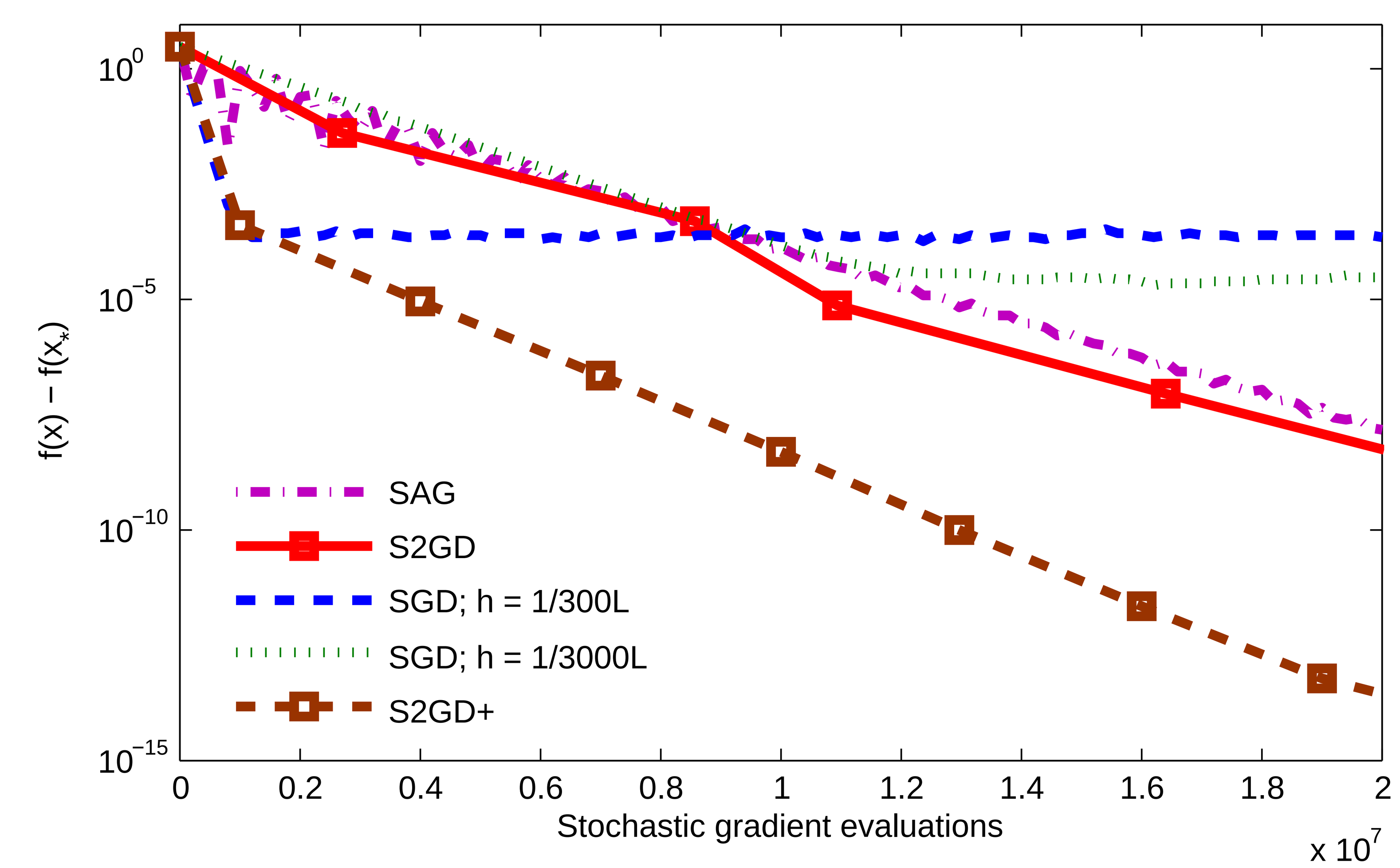
Then, we have the following convergence in expectation:

$$\mathbf{E}(f(x_j) - f(x_*)) \le c^j (f(x_0) - f(x_*)).$$

It is worth noting two special cases. With $\nu = 0$ we recover the result of [1], and with $\nu = \mu$, c can be written in the form

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

5. Experiments



The Figure presents practical performance of different stochastic methods on least squares problem with $n = 10^6, \kappa = 10^5$.

SAG is Stochastic Average Gradient of [3]. S2GD is our proposed algorithm. S2GD+ is algorithm we propose, but do not analyse. The algorithm runs SGD for 1 pass through the data, and then runs S2GD with fixed size of the inner loop. Note that the S2GD+ solves the problem to machine precision in just 20 passes through data. This is a vast improvement over the full Gradient Descent, which would need $O(10^5)$ passes through data.

6. References

[1] Johnson R., Zhang T.: Accelerating Stochastic Gradient Descent using Predictive Variance Reduction, *Advances in Neural Information Processing Systems*, 2013

[2] Konečný J, Richtárik P.: Semi-Stochastic Gradient Descent Methods, *arXiv:1312.1666*, 2013

[3] Schmidt M., Le Roux N., Bach F.: Minimizing Finite Sums with the Stochastic Average Gradient, *arXiv:1309.2388*, 2013

4. Optimal Choice of Params

The natural question is, if I require ϵ-accuracy, what are the parameters m, h I should use, and for how many iterations j should I run the algorithm? One can see this as a 3-dimensional work minimization problem dependent on parameters j, m, h, subject to achieving ϵ-accuracy.

While it is not possible to obtain closed form solution for this problem, we provide the following suboptimal values of parameters. Fix j, let $\Delta = \epsilon^{1/j}$ and $\kappa = \frac{L}{\mu}$.

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

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

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

$$\mathcal{W}(j^*) = O\left((n+\kappa)\log\left(\frac{1}{\epsilon}\right)\right)$$

In order to illustrate the strength of our method, we provide a comparison of work needed to solve a problem with $n = 10^9$ for small values of j, and different values of κ and ϵ.
$\mathcal{W}_\mu$ and $\mathcal{W}_0$ is the work needed with parameter $\nu = \mu$ and $\nu = 0$, measured in evaluations of f_i'.

	$\varepsilon = 10^{-6}, \kappa = 10^3$	
j	$\mathcal{W}_\mu(j)$	$\mathcal{W}_0(j)$
1	$116n$	$10^7 n$
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	$7.58n$	$10^4 n$
3	**3.18n**	$51.0n$
4	$4.03n$	$6.03n$
5	$5.01n$	**5.32n**
6	$6.00n$	$6.09n$

	$\varepsilon = 10^{-6}, \kappa = 10^6$	
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	$21.0n$	$23.5n$

	$\varepsilon = 10^{-6}, \kappa = 10^9$	
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	$1210n$	$3078n$