mS2GD: Mini-Batch Semi-Stochastic Gradient Descent in the Proximal Setting

Jakub Konečný
University of Edinburgh
United Kingdom, EH9 3FD
J.Konecny@sms.ed.ac.uk

Jie Liu
Lehigh University
Bethlehem, PA 18015
jie.liu@lehigh.edu

Peter Richtárik
University of Edinburgh
United Kingdom, EH9 3FD
peter.richtarik@ed.ac.uk

Martin Takáč
Lehigh University
Bethlehem, PA 18015
takac.mt@gmail.com

Abstract

We propose a mini-batching scheme for improving the theoretical complexity and practical performance of semi-stochastic gradient descent applied to the problem of minimizing a strongly convex composite function represented as the sum of an average of a large number of smooth convex functions, and simple nonsmooth convex function. Our method first performs a deterministic step (computation of the gradient of the objective function at the starting point), followed by a large number of stochastic steps. The process is repeated a few times with the last iterate becoming the new starting point. The novelty of our method is in introduction of mini-batching into the computation of stochastic steps. In each step, instead of choosing a single function, we sample b functions, compute their gradients, and compute the direction based on this. We analyze the complexity of the method and show that the method benefits from two speedup effects. First, we prove that as long as b is below a certain threshold, we can reach predefined accuracy with less overall work than without mini-batching. Second, our mini-batching scheme admits a simple parallel implementation, and hence is suitable for further acceleration by parallelization.

1 Introduction

The problem we are interested in is to minimize a sum of two convex functions,

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

where f is the average of a large number of smooth convex functions $f_i(x)$, i.e.,

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{2}$$

We further make the following assumptions:

Assumption 1. *The regularizer $R : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is convex and closed. The functions $f_i : \mathbb{R}^d \to \mathbb{R}$ are differentiable and have Lipschitz continuous gradients with constant $L > 0$. That is,*

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|,$$

for all $x, y \in \mathbb{R}^d$, where $\|\cdot\|$ is L2 norm.

Hence, the gradient of f is also Lipschitz continuous with the same constant L.

Assumption 2. *P is strongly convex with parameter $\mu > 0$, i.e., $\forall x, y \in \mathrm{dom}(R)$,*

$$P(y) \geq P(x) + \xi^T(y - x) + \frac{\mu}{2}\|y - x\|^2, \ \ \forall \xi \in \partial P(x), \tag{3}$$

where $\partial P(x)$ is the subdifferential of P at x.

Let $\mu_f \geq 0$ and $\mu_R \geq 0$ be the strong convexity parameters of f and R, respectively (we allow both of them to be equal to 0, so this is not an additional assumption). We assume that we have lower bounds available ($\nu_f \in [0, \mu_f]$ and $\nu_R \in [0, \mu_R]$).

Related work There has been intensive interest and activity in solving problems of (1) in the past years. An algorithm that made its way into many applied areas is FISTA [1]. However, this method is impractical in large-scale setting (big n) as it needs to process all n functions in each iteration. Two classes of methods address this issue – randomized coordinate descent methods [13, 15, 16, 11, 5, 19, 9, 10, 14, 4] and stochastic gradient methods [22, 12, 6, 20]. This brief paper is closely related to the works on stochastic gradient methods with a technique of explicit variance reduction of stochastic approximation of the gradient. In particular, our method is a mini-batch variant of S2GD [8]; the proximal setting was motivated by SVRG [7, 21].

A typical stochastic gradient descent (SGD) method will randomly sample i^{th} function and then update the variable x using $\nabla f_i(x)$ — an estimate of $\nabla f(x)$. Important limitation of SGD is that it is inherently sequential, and thus it is difficult to parallelize them. In order to enable parallelism, mini-batching—samples multiple examples per iteration—is often employed [17, 3, 2, 23, 20, 18].

Our Contributions. In this work, we combine the variance reduction ideas for stochastic gradient methods with mini-batching. In particular, develop and analyze mS2GD (Algorithm 1) – a mini-batch proximal variant of S2GD [8]. To the best of our knowledge, this is the first *mini-batch* stochastic gradient method with reduced variance for problem (1). We show that the method enjoys twofold benefit compared to previous methods. Apart from admitting a parallel implementation (and hence speedup in clocktime in an HPC environment), our results show that in order attain a specified accuracy ϵ, our mini-batching scheme can get by with less gradient evaluations. This is formalized in Theorem 2, which predicts more than linear speedup up to some b — the size of the mini-batches. Another advantage, compared to [21], is that we do not need to average the t_k points x in each loop, but we instead simply continue from the last one (this is the approach employed in S2GD [8]).

2 Proximal Algorithms

A popular *proximal gradient* approach to solving (1) is to form a sequence $\{y_k\}$ via

$$y_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left[U_k(x) \stackrel{\text{def}}{=} f(y_k) + \nabla f(y_k)^T(x - y_k) + \frac{1}{2h}\|x - y_k\|^2 + R(x) \right].$$

Note that U_k in an upper bound on P if $h > 0$ is a stepsize parameter satisfying $1/h \geq L$. This procedure can be equivalently written using the *proximal operator* as follows:

$$y_{k+1} = \mathrm{prox}_{hR}(y_k - h\nabla f(y_k)), \qquad \mathrm{prox}_R(z) \stackrel{\text{def}}{=} \arg\min_{x \in \mathbb{R}^d} \{\tfrac{1}{2}\|x - z\|^2 + R(x)\}.$$

In large-scale setting it is more efficient to instead consider the *stochastic proximal gradient* approach, in which the proximal operator is applied to a stochastic gradient step:

$$y_{k+1} = \mathrm{prox}_{hR}(y_k - hv_k), \tag{4}$$

where v_k is a stochastic estimate of $\nabla f(y_k)$. Of particular relevance to our work are the the SVRG [7], S2GD [8] and Prox-SVRG [21] methods where the stochastic estimate of $\nabla f(y_k)$ is of the form

$$v_k = \nabla f(x) + \tfrac{1}{nq_{i_k}}(\nabla f_{i_k}(y_k) - \nabla f_{i_k}(x)), \tag{5}$$

where x is an "old" reference point for which the gradient $\nabla f(x)$ was already computed in the past, and $i_k \in \{1, 2, \ldots, n\}$ is a random index equal to i with probability $q_i > 0$. Notice that v_k is an unbiased estimate of the gradient:

$$\mathbf{E}_i[v_k] \stackrel{(5)}{=} \nabla f(x) + \sum_{i=1}^n q_i \tfrac{1}{nq_i}(\nabla f_i(y_k) - \nabla f_i(x)) \stackrel{(2)}{=} \nabla f(y_k).$$

SVRG, S2GD and Prox-SVRG, as well as our new mS2GD method, update the points y_k in an inner loop, and the reference point x in an outer loop. This ensures that v_k has low variance, which ultimately leads to extremely fast convergence.

3 Mini-batch S2GD

We now describe the mS2GD method (Algorithm 1).

Algorithm 1 mS2GD

1: **Input:** m (max # of stochastic steps per epoch); $h > 0$ (stepsize); $x_0 \in \mathbb{R}^d$ (starting point); minibatch size $b \in [n]$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Compute and store $g_k \leftarrow \nabla f(x_k) = \frac{1}{n}\sum_i \nabla f_i(x_k)$
4: Initialize the inner loop: $y_{k,0} \leftarrow x_k$
5: Let $t_k \leftarrow t \in \{1, 2, \ldots, m\}$ with probability q_t given by (6)
6: **for** $t = 0$ to $t_k - 1$ **do**
7: Choose mini-batch $A_{kt} \subset [n]$ of size b, uniformly at random
8: Compute a stoch. estimate of $\nabla f(y_{k,t})$: $v_{k,t} \leftarrow g_k + \frac{1}{b}\sum_{i \in A_{kt}} (\nabla f_i(y_{k,t}) - \nabla f_i(x_k))$
9: $y_{k,t+1} \leftarrow \mathrm{prox}_{hR}(y_{k,t} - h v_{k,t})$
10: **end for**
11: Set $x_{k+1} \leftarrow y_{k,t_k}$
12: **end for**

The main step of our method (Step 8) is given by the update (4), however with the stochastic estimate of the gradient instead formed using a *mini-batch* of examples $A_{kt} \subset [n]$ of size $|A_{kt}| = b$. We run the inner loop for t_k iterations, where $t_k = t \in \{1, 2, \ldots, m\}$ with probability q_t given by

$$q_t \overset{\text{def}}{=} \frac{1}{\gamma}\left(\frac{1-h\mu_f}{1+h\nu_R}\right)^{m-t}, \quad \text{with} \quad \gamma \overset{\text{def}}{=} \sum_{t=1}^{m}\left(\frac{1-h\mu_f}{1+h\nu_R}\right)^{m-t}. \tag{6}$$

4 Complexity Result

In this section, we state our main complexity result and comment on how to optimally choose the parameters of the method.

Theorem 1. *Let Assumptions 1 and 2 be satisfied and let* $x_* \overset{def}{=} \arg\min_x P(x)$. *In addition, assume that the stepsize satisfies* $0 < h < \min\{\frac{1-h\mu_f}{1+h\nu_R}\frac{1}{4L\alpha(b)}, \frac{1}{L}\}$ *and that* m *is sufficiently large so that*

$$\rho \overset{def}{=} \frac{\left(\frac{1-h\mu_f}{1+h\nu_R}\right)^m \frac{1}{\mu} + \frac{4h^2 L\alpha(b)}{1+h\nu_R}\left(\gamma + \left(\frac{1-h\mu_f}{1+h\nu_R}\right)^{m-1}\right)}{\gamma h\left\{\frac{1}{1+h\nu_R} - \frac{4hL\alpha(b)}{1-h\mu_f}\right\}} < 1, \tag{7}$$

where $\alpha(b) = \frac{n-b}{b(n-1)}$. *Then mS2GD has linear convergence in expectation:*

$$\mathbf{E}(P(x_k) - P(x_*)) \le \rho^k (P(x_0) - P(x_*)).$$

If we consider the special case $\nu_f = 0$, $\nu_R = 0$ (i.e., if the algorithm does not have any nontrivial good lower bounds on μ_f and μ_R), we obtain

$$\rho = \frac{1}{\mu h(1 - 4Lh\alpha(b))m} + \frac{4Lh\alpha(b)\,(m+1)}{(1 - 4Lh\alpha(b))m}. \tag{8}$$

In the special case when $b = 1$ we get $\alpha(b) = 1$ in the above theorem, the rate given by (8) exactly recovers the rate achieved by Prox-SVRG [21]. If moreover $R = 0$, this rate is very similar (albeit very slightly weaker) to the one achieved by S2GD (see Eq (12) in [8]) in the case when the method does not have access to any nontrivial lower bound on μ.

4.1 Mini-batch speedup

In order to be able to see the speed-up we can gain from the mini-batch strategy, and due to many parameters in the complexity result (Theorem 1) we need to fix some of the parameters. For simplicity, we will use ν_f and ν_R equal to 0, so we can analyse (8) instead of (7). Let us consider the case when we also fix k (number of outer iterations). Once the parameter k is fixed and in order to get some ϵ accuracy, we get the value of ρ which will guarantee the result.

Let us now fix target decrease in single epoch $\rho = \rho_*$. For any $1 \leq b \leq n$, define (h_*^b, m_*^b) to be the optimal pair stepsize-size of the inner loop, such that $\rho < \rho_*$. This pair is optimal in the sense that m_*^b is the smallest possible — because we are interested in minimizing the computational effort, thus minimizing m. If we set $b = 1$, we recover the optimal choice of parameters without mini-batching. If $m_*^b \leq m_*^1/b$, then we can reach the same accuracy with less evaluations of gradient of a function f_i. The following theorem states the formula for h_*^b and m_*^b. Equation (9) shows that as long as the condition $\tilde{h}^b \leq \frac{1}{L}$ is satisfied, m_*^b is decreasing at a rate faster than $1/b$. Hence, we can attain the same accuracy with less work, compared to the case when $b = 1$.

Theorem 2. *Fix target $\rho = \rho_*$, where ρ is given by* (8) *and $\rho_* \in (0, 1)$. If we consider the mini-batch size b fixed, then the choice of stepsize h_*^b and size of inner loop m_*^b that minimizes the work done — the number of gradients evaluated — while having $\rho \leq \rho_*$, is given by:*

$$\tilde{h}^b := \sqrt{\left(\tfrac{1+\rho}{\rho\mu}\right)^2 + \tfrac{1}{4\mu\alpha(b)L}} - \tfrac{1+\rho}{\rho\mu}.$$

If $\tilde{h}^b \leq \frac{1}{L}$ then $h_*^b = \tilde{h}^b$ and

$$m_*^b = \frac{4}{\left(\sqrt{\frac{\rho^2\mu}{\alpha(b)L} + 4(1+\rho)^2} - 2(1+\rho)\right)} = 8\alpha(b)L\frac{1+\rho+\sqrt{\frac{1}{4\alpha(b)L}\mu\rho^2 + (1+\rho)^2}}{\mu\rho^2}. \quad (9)$$

Otherwise $h_*^b = \frac{1}{L}$ and $m_*^b = \frac{L/\mu + 4\alpha(b)}{\rho - 4\alpha(b)(1+\rho)}$.

5 Experiments

In this section we present a preliminary experiment, and an insight into the possible speedup by parallelism. Figure 1 shows experiments on L2-regularized logistic regression on the RCV1 dataset. [1] We compare S2GD (blue, squares) and mS2GD (green circles) with mini-batch size $b = 8$, without any parallelism. The figure demonstrates that one can achieve the same accuracy with less work. The green dashed line is the ideal (most likely practically unachievable) result with parallelism (we divide passes through data by b). For comparison, we also include SGD with constant stepsize (purple, stars), chosen in hindsight to optimize performance. Figure 2 shows the possible speedup in terms of work done, formalized in Theorem 2. Notice that up to a certain threshold, we do not need any more work to achieve the same accuracy (red straight line is ideal speedup; blue curvy line is what mS2GD achieves).

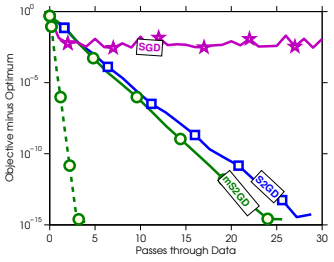

Figure 1: mS2GD needs fewer data passes than other methods (RCV1 data).

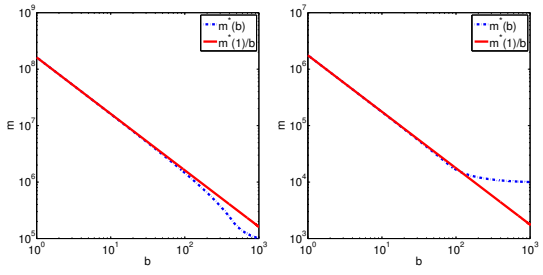

Figure 2: Speedup from parallelism for $\rho = 0.01$ (left) and $\rho = 0.1$ (right). Parameters: $L = 1, n = 1000, \mu = 1/n$.

[1] Available at http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/.

References

[1] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2(1):183–202, 2009.

[2] Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *NIPS*, pages 1647–1655, 2011.

[3] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *JMLR*, 13(1):165–202, 2012.

[4] Olivier Fercoq, Zheng Qu, Peter Richtárik, and Martin Takáč. Fast distributed coordinate descent for non-strongly convex losses. In *IEEE Workshop on Machine Learning for Signal Processing*, 2014.

[5] Olivier Fercoq and Peter Richtárik. Accelerated, parallel and proximal coordinate descent. *arXiv:1312.5799*, 2013.

[6] Martin Jaggi, Virginia Smith, Martin Takáč, Jonathan Terhorst, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. *NIPS*, 2014.

[7] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *NIPS*, pages 315–323, 2013.

[8] Jakub Konečný and Peter Richtárik. Semi-stochastic gradient descent methods. *arXiv:1312.1666*, 2013.

[9] Jakub Mareček, Peter Richtárik, and Martin Takáč. Distributed block coordinate descent for minimizing partially separable functions. *arXiv:1406.0238*, 2014.

[10] Ion Necoara and Dragos Clipici. Distributed coordinate descent methods for composite minimization. *arXiv:1312.5302*, 2013.

[11] Ion Necoara and Andrei Patrascu. A random coordinate descent algorithm for optimization problems with composite objective function and linear coupled constraints. *Comp. Optimization and Applications*, 57(2):307–337, 2014.

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

[13] Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optimization*, 22:341–362, 2012.

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

[15] Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

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

[17] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming: Series A and B- Special Issue on Optimization and Machine Learning*, pages 3–30, 2011.

[18] Shai Shalev-Shwartz and Tong Zhang. Accelerated mini-batch stochastic dual coordinate ascent. In *NIPS*, pages 378–385, 2013.

[19] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *JMLR*, 14(1):567–599, 2013.

[20] Martin Takáč, Avleen Singh Bijral, Peter Richtárik, and Nathan Srebro. Mini-batch primal and dual methods for SVMs. *ICML*, 2013.

[21] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *arXiv:1403.4699*, 2014.

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

[23] Peilin Zhao and Tong Zhang. Accelerating minibatch stochastic gradient descent using stratified sampling. *arXiv:1405.3080*, 2014.