Less than a Single Pass: Stochastically Controlled Stochastic Gradient

Lihua Lei
University of California, Berkeley

Michael I. Jordan
University of California, Berkeley

Abstract

We develop and analyze a procedure for gradient-based optimization that we refer to as *stochastically controlled stochastic gradient* (SCSG). As a member of the SVRG family of algorithms, SCSG makes use of gradient estimates at two scales. Unlike most existing algorithms in this family, both the computation cost and the communication cost of SCSG do not necessarily scale linearly with the sample size n; indeed, these costs are independent of n when the target accuracy is small. An experimental evaluation of SCSG on the MNIST dataset shows that it can yield accurate results on this dataset on a single commodity machine with a memory footprint of only 2.6MB and only eight disk accesses.

1 Introduction

Optimization problems in machine learning are often solved by algorithms that either make use of full gradients (obtained by processing the entire dataset) or stochastic gradients (obtained by processing single data points or mini-batches of data points). The use of the former provides guarantees of eventual convergence and the latter yields advantages in terms of convergence rate, scalability and simplicity of implementation (Hazan et al., 2007; Nemirovski et al., 2009; Rakhlin et al., 2012). An impactful recent line of research has shown that a hybrid methodology that makes use of both full gradients and stochastic gradients can obtain the best of both worlds—guaranteed convergence at favorable rates; (see, e.g. Shalev-Shwartz & Zhang, 2013; Johnson & Zhang, 2013; Defazio, Bach, & Lacoste-Julien, 2014; Lin, Mairal, & Harchaoui, 2015; Allen-Zhu, 2017). The full gradients provide variance control for the stochastic gradients.

Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54.

While this line of research represents significant progress towards the goal of designing scalable, autonomous learning algorithms, there remain some inefficiencies in terms of computation. With the definition of computation and communication cost in Section 2.1, the methods referred to above require $O(n \cdot C(\epsilon, d))$ computation to achieve an ϵ-approximate solution, where n is the number of data points, ϵ is a target accuracy and d is the dimension of the parameter vector. Some methods incur a $O(nd)$ storage cost (Roux et al., 2012; Defazio et al., 2014). The linear dependence on n is problematic in general. Clearly there will be situations in which accurate solutions can be obtained with less than a single pass through the data; indeed, some problems will require a notionally constant number of steps. This will be the case, for example, if the data in a regression problem consist of a fixed number of pairs repeated a large number of times. For deterministic algorithms, the worst case analysis in Agarwal and Bottou (2015) shows that scanning at least a fixed proportion of the data is necessary; however, learning algorithms are generally stochastic and real-world learning problems are generally not worst case. Recently, Woodworth and Srebro (2016) establish a lower bound for the computational cost of randomized algorithms in minimizing the finite sums. However, their result only applies to the case in which $\epsilon \ll \frac{1}{\sqrt{n}}$ and the summands are smooth, leaving open the possibility that smaller computational complexity can be achieved when these assumptions do not hold.

An equally important bottleneck for learning algorithms is the cost of communication. For large data sets that must be stored on disk or distributed across many computing nodes, the communication cost can be significant, even dominating the computation cost. For example, classical stochastic gradient descent (SGD) makes use of random sampling which can incur prohibitive communication cost. There is an active line of research that focuses on communication costs; (see, e.g., Zhang, Wainwright, & Duchi, 2012; Jaggi et al., 2014; Arjevani & Shamir, 2015; Konečnỳ, McMahan, & Ramage, 2015).

In this article, we present a variant of the stochastic

variance reduced gradient (SVRG) method that we refer to as *stochastically controlled stochastic gradient* (SCSG). In contradistinction to SVRG, the theoretical convergence rate of SCSG has a sublinear regime in terms of both computation and communication. The basic idea behind SCSG—that of approximating the full gradient in SVRG via a subsample—has been explored by others, but we present several innovations that yield significant improvements both in theory and in practice. The performance of SCSG is superior to related algorithms when $\epsilon \gg \frac{1}{n}$. This regime is important in machine learning problems, notably in the common situation in which the sample size is large, ($n \in [10^4, 10^9]$), while the required accuracy is low, $\epsilon \in [10^{-4}, 10^{-2}]$. The analysis in this article shows that SCSG is able to achieve the target accuracy in this regime with less than a single pass through the data.

Although SGD can be shown to incur the same computational cost as SCSG for non-strongly-convex functions (for strongly-convex functions we show that SCSG has a better rate than SGD), it requires tuning the stepsize based on the information that is unknown in practice, and it requires fixing the number of steps prior to the implementation. In other words, without intensive tuning, it is not guaranteed to reach an ϵ-approximate solution for general convex functions. In contrast, we propose a simple automatic tuning procedure which guarantees that SCSG finds an ϵ-approximate solution.

Another line of work explores the convergence of first-order methods within one pass of data under a stochastic framework; e.g., streaming SVRG (Frostig et al., 2015) and dynaSAGA(Daneshmand et al., 2016). The analyses in this framework assume that data points are independent and identically distributed, an assumption that is often unrealistic in the streaming setting. In contrast, SCSG does not make such an i.i.d. assumption; indeed, it does not assume any randomness for the observed data. Admittedly, it is interesting to explore SCSG under the i.i.d. framework to understand implications for generalization error, but such an analysis is beyond the scope of the current paper.

The remainder of the paper is organized as follows. In Section 2, we review SVRG, discuss several of its variants and we describe the SCSG algorithm. We provide a theoretical convergence analysis in Section 3. Our analysis involves a data-dependent quantity, G_n, which is defined at the beginning of Section 3. We show in Section 4 that G_n is estimable for generalized linear models. We present an experimental evaluation in Section 5. In Appendix A we compare the computation and communication cost with state-of-art methods. All technical proofs are relegated to the Appendix B. Other relevant issues are discussed in Appendix C.

2 Assumptions and Algorithm

Throughout this paper we consider the following minimization problem:

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

where $f_i(x)$ are convex functions. Let x^* denote a minimizer of f, $\tilde{x}_0$ denote the initial value and $D_0 = \|\tilde{x}_0 - x^*\|^2$. A point y, possibly random, is called an ϵ-approximate solution if

$$\mathbb{E}f(y) - f(x^*) \le \epsilon.$$

The following assumptions will be used in various contexts in the paper:

A1 f_i is convex with L-Lipschitz gradient,

$$f_i(x) - f_i(y) - \langle \nabla f_i(y), x - y\rangle \le \frac{L}{2}\|x-y\|^2,$$

for some $L < \infty$ and all $i \in \{1, \dots, n\}$;

A2 f_i is strongly convex with

$$f_i(x) - f_i(y) - \langle \nabla f_i(y), x - y\rangle \ge \frac{\mu}{2}\|x-y\|^2,$$

for some $\mu > 0$ and all $i \in \{1, \dots, n\}$;

A3 f is strongly convex with

$$f(x) - f(y) - \langle \nabla f(y), x - y\rangle \ge \frac{\bar{\mu}}{2}\|x-y\|^2,$$

for some $\bar{\mu} > 0$.

If Assumption **A**2 holds, then Assumption **A**3 also holds with $\bar{\mu} \ge \mu$. For unpenalized linear regression where $f_i(x) = (y_i - a_i^T x)^2$, Assumption **A**2 fails to hold for any i if $d > 1$. However, $f(x)$ is strongly convex if $\frac{1}{n}\sum_{i=1}^n a_i a_i^T$ is positive definite. On the other hand, an L_2-regularization term is often added to (1) such that the problem is to minimize $f(x) + \frac{\mu}{2}\|x\|^2$ for some $\mu > 0$, in which case $f_i(x)$ can be replaced by $f_i(x) + \mu/2\|x\|^2$. Thus, Assumption **A**2 is also reasonable for practitioners and it results in tighter convergence analysis and a simpler tuning procedure than Assumption **A**3 (as shown in Section 3). In the presence of strong convexity, $\kappa = L/\mu$ (resp. $L/\bar{\mu}$) denotes the condition number under Assumption **A**2 (resp. **A**3).

2.1 Computation and Communication Cost

To formally compare the algorithms, we need to define the computation cost and the communication cost. In this article we adopt the IFO framework (Agarwal

& Bottou, 2015; Reddi et al., 2016) under which the sampling of an index i and the evaluation of the pair $(f_i(x), \nabla f_i(x))$ incurs one unit of computational cost. Note that we ignore the effect of dimension, treating it as a constant; if desired, we could multiply the IFO complexity by dimension to track dimension dependence.

To define the communication cost, we consider two computational models. In the first model, we have a single machine and the data is too large to fit into the memory and has to be stored on the disk. We assume that accessing one data point from the disk incurs one unit of communication cost. In the second model, we consider a distributed system with a datacenter and multiple worker machines. The data are stored remotely in the workers and the main computation tasks are implemented in the datacenter. This setting has been considered in various recent papers; (see, e.g., Zhang et al., 2012; Arjevani & Shamir, 2015; Konečnỳ et al., 2015). Similar to the first model, we assume that accessing or sampling one data point from a node incurs one unit of communication cost. It turns out that these two models give the same communication cost for algorithms we consider in this paper and hence we do not distinguish them in the following discussion.

Algorithm 1 Stochastic Variance Reduced Gradient (SVRG) Method

Inputs: Stepsize η, number of stages T, initial iterate $\tilde{x}_0$, number of SGD steps m.

Procedure

1: **for** $j = 1, 2, \cdots, T$ **do**
2: $\quad g_j \leftarrow \nabla f(\tilde{x}_{j-1}) = \frac{1}{n}\sum_{i=1}^n \nabla f_i(\tilde{x}_{j-1})$
3: $\quad x_0 \leftarrow \tilde{x}_{j-1}$
4: $\quad N_j \leftarrow m$
5: $\quad$ **for** $k = 1, 2, \cdots, N_j$ **do**
6: $\quad\quad$ Randomly pick $i_k \in \{1, \ldots, n\}$
7: $\quad\quad \nu_{k-1} \leftarrow \nabla f_{i_k}(x_{k-1}) - \nabla f_{i_k}(x_0) + g_j$
8: $\quad\quad x_k \leftarrow x_{k-1} - \eta\nu_{k-1}$
9: $\quad$ **end for**
10: $\quad \tilde{x}_j \leftarrow x_{N_j}$
11: **end for**

Output: (Option 1): $\tilde{x}_T$ (Option 2): $\bar{x}_T = \frac{1}{T}\sum_{j=1}^T \tilde{x}_j$.

2.2 SVRG and Other Related Works

The stochastic variance reduced gradient (SVRG) method blends gradient descent and stochastic gradient descent, using the former to control the effect of the variance of the latter (Johnson & Zhang, 2013). We summarize SVRG in Algorithm 1.

Using the definition from Section 2.1, it is easy to see that the computation cost of SVRG is $O((n+m)T)$. Each step of SVRG involves accessing all data points to compute the full gradient, which incurs a communication cost of n, and sampling m data points for stochastic gradient descent, which incurs a communication cost of m. Thus, the total communication cost is $O((n+m)T)$. As shown in the convergence analysis of (Johnson & Zhang, 2013), m is required to be $\Omega(\kappa)$ to guarantee convergence. Thus, the computation and communication cost of SVRG are both $O((n+\kappa)T)$.

A number of variants of SVRG have been studied. For example, a constrained form of SVRG can be obtained by replacing line 8 with a projected gradient descent step (Xiao & Zhang, 2014). A mini-batch variant of SVRG arises when one samples a subset of indices instead of a single index in line 6 and updates the iterates by the average gradient in this batch in line 7 (Nitanda, 2014). Similarly, we can consider implementing the full gradient computation in line 2 using a subsample. This is proposed in Harikandeh et al. (2015), which calculates g_j as $\frac{1}{B}\sum_{i\in\mathcal{I}} \nabla f_i(\tilde{x})$ where $\mathcal{I}$ is a subset of size B uniformly sampled from $\{1, \ldots, n\}$. Harikandeh et al. (2015) heuristically show the potential for significant complexity reduction, but they only prove convergence for $B = \Omega(n)$ and under the stringent condition that $\|\nabla f_i(x)\|$ is uniformly bounded for all x. Similar to Nesterov's acceleration for gradient descent, momentum terms can be added to the SGD steps to accelerate SVRG (Nitanda, 2016; Allen-Zhu, 2017).

Much of this work focuses on the strongly convex case. In the non-strongly convex setting one way to proceed is to add a L_2 regularizer $\frac{\lambda}{2}\|x\|^2$. Tuning λ, however, is subtle and requires multiple runs of the algorithm on a grid of λ (Allen-Zhu & Yuan, 2016). For general convex functions an alternative approach has been presented by Allen-Zhu and Yuan (2016) (they generate N_j by a different scheme in line 4), who also establish a computation rate $O\left(\frac{n}{\epsilon}\right)$. Another approach is discussed by Reddi et al. (2016), who establish convergence for non-strongly convex functions (and non-convex functions with Lipschitz gradient) by considering a weaker stopping criterion based on $\mathbb{E}\|\nabla f(x_k)\|^2$. However, their algorithm relies on calculating a full gradient, involves a complicated stepsize-setting scheme and requires a strong assumption that the $\|\nabla f_i(x)\|$ are uniformly bounded. Other variants of SVRG have been proposed in the distributed computing setting (Lee et al., 2015; Reddi et al., 2015) and in the stochastic setting (Frostig et al., 2015; Daneshmand et al., 2016).

2.3 SCSG

SCSG is similar to the algorithm by Harikandeh et al. (2015) in that it implements the gradient computation on a subsample $\mathcal{I}$ of size B. However, there are three

key differences:

- For non-strongly convex functions, N_j is generated from a geometric distribution with failure probability $\gamma = \frac{1}{B}$ (line 4 of Algorithm 1);
- For strongly convex functions, N_j is generated from a truncated geometric distribution with $P(N_j = k) \propto \gamma^k, \forall k \in \{1, \ldots, m\}$ (with γ and m as determined in Algorithm 3 below);
- The data point index is generated from $\mathcal{I}$ instead of $\{1, \ldots, n\}$ (line 6 of Algorithm 1).

The final point is of particular importance in a practical implementation. By restricting the stochastic gradient step to the batch $\mathcal{I}$, an iteration of SCSG needs only to load that batch into the memory; this can be feasible on a laptop even for large datasets.

We present SCSG in Algorithm 2 and present the method for setting parameters in Algorithm 3.

Algorithm 2 Stochastically Controlled Stochastic Gradient (SCSG) Method

Inputs: Stepsize η, batch size B, number of stages T, initial iterate $\tilde{x}_0$.
Initialization: Get distribution $\mathcal{P}$ by Algorithm 3.
Procedure

1: **for** $j = 1, 2, \cdots, T$ **do**
2: Uniformly sample a batch $\mathcal{I}_j \subset \{1, \cdots, n\}$ with $|\mathcal{I}_j| = B$
3: $g_j \leftarrow \frac{1}{B} \sum_{i \in \mathcal{I}_j} \nabla f_i(\tilde{x}_{j-1})$
4: $x_0 \leftarrow \tilde{x}_{j-1}$
5: Generate $N_j \sim \mathcal{P}$
6: **for** $k = 1, 2, \cdots, N_j$ **do**
7: Randomly pick $i_k \in \mathcal{I}_j$
8: $\nu_{k-1} \leftarrow \nabla f_{i_k}(x_{k-1}) - \nabla f_{i_k}(x_0) + g_j$
9: $x_k \leftarrow x_{k-1} - \eta \nu_{k-1}$
10: **end for**
11: $\tilde{x}_j \leftarrow x_{N_j}$
12: **end for**

Output: (Strongly convex case): $\tilde{x}_T$ (Non-strongly convex case): $\bar{x}_T = \frac{1}{T} \sum_{j=1}^{T} \tilde{x}_j$.

The average computation cost of SCSG is $BT + \sum_{j=1}^{n} N_j$. By the law of large numbers, this is close to $O((B + \mathbb{E}N_1)T)$, which has a similar form to SVRG. Table 1 in Appendix A summarizes the complexity of SCSG as well as 11 existing popular algorithms. The conclusion is that SCSG and SGD are the only two methods which could find an ϵ-approximate solution with less than a single pass of data.

For the general convex case, as discussed in Section 1, SCSG requires much less tuning than SGD and SCSG does not require f_i to be Lipschitz. For the strongly convex case (Assumption **A**3), SCSG has a complexity of $O\left(\left(\frac{1}{\epsilon} + \kappa\right) \log \frac{1}{\epsilon}\right)$, which is strictly less than that of SGD, which is $O\left(\frac{\kappa}{\epsilon} \log \frac{1}{\epsilon}\right)$. In both cases, SCSG outperforms SGD in terms of computation and amount of tuning.

On the other hand, all of the other methods in Table 1 have a communication cost that is the same as the computation cost since they all sample the stochastic gradient index uniformly from $\{1, \ldots, n\}$ and hence needs to communicate with disk/worker machines. In contrast, SCSG is communication-avoiding since it only needs to communicate when calculating g_j (line 3) and the ensuing SGD steps do not incur communication costs since the batch is fit into the memory. As shown in Table 2 the communication cost of SCSG is free of n and even free of the condition number κ. In other words, in contrast to most other methods, ill-conditioned problems do not overload the communication.

One might argue that the methods listed in Table 1 are not designed for communication efficiency. To make a comparison to algorithms that explicitly aim for communication efficiency, we consider CoCoA (Jaggi et al., 2014), DANE (Shamir et al., 2014) and DiSCO (Zhang & Lin, 2015), and present the results in Table 2 in Appendix A. We found that these methods all involve a tradeoff between the computation cost and communication cost, controlled by the number of workers, and at least one of them will depend on the sample size even when ϵ is large. In contrast, SCSG is able to control the communication cost and the computation cost simultaneously to be independent of the sample size and the number of the worker machines . We will discuss it in more details in Appendix A.

Finally, in contrast to many existing proposals, our tuning procedure is straightforward. Generally, only η and B need to be tuned and a default stepsize is given in Algorithm 3. For certain problems such as generalized linear models, the block size B can also be set automatically. Furthermore, Algorithm 2 is guaranteed to reach an ϵ-approximate solution with our recommended set of parameters, regardless of the initial iterate $\tilde{x}_0$ (a better initial value will accelerate the algorithm by a constant factor). In addition, the stepsize η does not depend on the number of stages T. Thus, the parameters T and $\tilde{x}_0$ are not essential. This is in contradistinction to SGD, which depends crucially on these parameters to achieve the same convergence rate as SCSG (for the general convex case).

3 Convergence Analysis

In this section we present a convergence analysis of SCSG. Note that SCSG is similar to SVRG when $B = n$.

Our theoretical results are similar to those for SVRG in this case; our focus, however, is the case in which $B < n$, where additional sampling variation enters in. To capture the effect of such variation we define the following quantity:

$$G_n = \frac{1}{n}\sum_{i=1}^{n} \|\nabla f_i(x^*)\|^2, \tag{2}$$

where $\|\cdot\|$ denotes the L_2 norm. We then have the following lemma.

Lemma 1 *Let $\mathcal{I} \in \{1, \cdots, n\}$ be a random subset of size B, and define the random variable $g = \frac{1}{B}\sum_{i\in\mathcal{I}} \nabla f_i(x^*)$. Then $\mathbb{E}g = 0$ and*

$$\mathbb{E}\|g\|^2 = \frac{(n-B)G_n}{(n-1)B}.$$

The proof, which appears in Appendix B.1, involves a standard technique for analyzing sampling without replacement. Note that the extra variation vanishes when $B = n$ and in general is inversely proportional to the batch size. To further control the variance, we need to bound G_n. Obviously, G_n is bounded if $\|\nabla f_i(x)\|$ is uniformly bounded as is often assumed in the literature. However, via a more refined analysis, we can discard this stringent condition and only require Assumption **A**1. The next lemma provides a bound on G_n; the proof appears in Appendix B.1.

Lemma 2 *Under Assumption **A**1, for any $x \in \mathbb{R}^d$,*

$$\frac{G_n}{L^2} \le \frac{2}{L^2 n}\sum \|\nabla f_i(\tilde{x}_0)\|^2 + 4\|\tilde{x}_0 - x^*\|^2.$$

If further $f(x) \ge B$ is bounded below by $B \in \mathbb{R}$, then

$$\frac{G_n}{L^2} \le \frac{2}{L^2 n}\sum \|\nabla f_i(\tilde{x}_0)\|^2 + 4(f(\tilde{x}_0) - B).$$

In practice, the loss function is non-negative and Lemma 2 provides a way to estimate G_n. This estimate can be significantly improved in special cases such as Generalized Linear Models. Generally, G_n has the same order as $L^2 D_0$ and we should treat it as $O(1)$ if D_0 is treated as $O(1)$; see Section 4 for a thorough discussion.

3.1 Non-Strongly Convex Case

Our convergence result for non-strongly convex functions is stated in Theorem 1. This theorem is stated for arbitrary $\gamma \in (0, 1)$; the choice in Algorithm 3 is a special case.

Theorem 1 *Assume **A**1 holds. Let $D_0 = \|\tilde{x}_0 - x^*\|^2$ and $\eta = \frac{\theta}{L}$,*

Algorithm 3 Initialization of SCSG

Inputs: Stepsize η, batch size B.
Optional Inputs: gradient Lipschitz constant L, strong convexity modulus μ.
Procedure:
  if μ is unknown **then**
    $\gamma \leftarrow \frac{B-1}{B}$;
    $\mathcal{P} \leftarrow$ geometric distribution with $\mathcal{P}(\{k\}) \propto \gamma^{k-1}$;
  else
    if L is known **then**
      $\gamma \leftarrow 1 - \eta\mu, m = \lceil \frac{1}{2L\mu\eta^2} \rceil$;
    else
      $\gamma \leftarrow \sqrt{1-\eta\mu}, m = \left\lceil \log(\frac{1}{1-\gamma}) / \log(\frac{\gamma}{1-\eta\mu}) \right\rceil$;
    end if
    $\mathcal{P} \leftarrow$ truncated geometric distribution on $\{1, \ldots, m\}$ with $\mathcal{P}(\{k\}) \propto (\gamma/(1-\eta\mu))^k$
  end if
Output: Distribution $\mathcal{P}$.

1. *If $B = n$ and $\theta \in (0, \frac{1}{2})$, then*

$$\mathbb{E}(f(\bar{x}_T) - f(x^*)) \le \frac{1}{T}\cdot\frac{LD_0}{2\theta(1-2\theta)\gamma};$$

2. *If $B < n$ and $\theta \in (0, \frac{1}{2})$, then*

$$\mathbb{E}(f(\bar{x}_T) - f(x^*)) \le C_1\frac{1}{T}\cdot\frac{LD_0}{\theta\gamma} + C_2\frac{\theta}{\gamma}\frac{(n-B)}{(n-1)B}\cdot\frac{G_n}{L},$$

for some C_1 and C_2, which only depend on θ. In particular, when $\theta < \frac{1}{5}$, we have $C_1, C_2 \le 2.5$.

Corollary 1 *Assume **A**1 holds and select the batch size B, number of stages T and decay rate γ such that*

$$B \ge \min\left\{n, \left\lceil \frac{10\theta}{\epsilon}\cdot\frac{G_n}{L} \right\rceil\right\}, \gamma = \frac{B-1}{B}, T \ge \left\lceil \frac{5D_0}{\theta\gamma}\cdot\frac{L}{\epsilon} \right\rceil.$$

Letting $\eta = \frac{\theta}{L}$ with $\theta \in (0, \frac{1}{5})$, then $\mathbb{E}(f(\bar{x}_T) - f(x^*)) \le \epsilon$ with

$$\mathbb{E}C_{\text{comp}} = O\left(\min\left\{n, \frac{G_n}{L^2}\cdot\frac{L}{\epsilon}\right\}\frac{D_0 L}{\epsilon}\right),$$

$$\mathbb{E}C_{\text{comm}} = O\left(\min\left\{n, \frac{G_n}{L^2}\cdot\frac{L}{\epsilon}\right\}\frac{D_0 L}{\epsilon}\right).$$

3.2 Strongly Convex Case

First we assume that $f(x)$ is strongly convex in the sense that the Assumption **A**2 holds. Our main result in this case is stated in Theorem 2 for arbitrary decay rate $\gamma \in [1-\eta\mu, 1)$ and truncation parameter m (again the specific choices in Algorithm 3 are a special case).

Theorem 2 *Assume **A**1 and **A**2. Let $D_0 = \|\tilde{x}_0 - x^*\|^2$ and $\eta = \frac{\theta}{L+\mu}$. Assume that one of the following conditions holds:*

(i) $\gamma > 1-\eta\mu,\ m \geq \log\left(\frac{1}{1-\gamma}\right) \Big/ \log\left(\frac{\gamma}{1-\eta\mu}\right)$,

(ii) $\gamma = 1-\eta\mu,\ m \geq \frac{1}{2L\mu\eta^2}$.

Then

1. *if* $B = n$ *and* $\theta \in (0, \frac{1}{2})$,

$$\mathbb{E}(f(\tilde{x}_T) - f(x^*)) \leq 10LD_0 \cdot (2\theta)^{T-1};$$

2. *if* $B < n$ *and* $\theta \in (0, \frac{1}{2})$,

$$\mathbb{E}(f(\tilde{x}_T) - f(x^*)) \leq 10LD_0 \cdot (2\theta)^{\frac{T}{2}-1} + C_3\theta\frac{(n-B)}{(n-1)B} \cdot \frac{G_n}{L}.$$

for some C_3*, which only depends on* θ*. In particular, when* $\theta < \frac{1}{5}$*,* $C_3 \leq 20$.

Similar to the non-strongly convex case, we can derive the computation and the communication cost of SCSG. Corollary 2 summarizes the result for fixed θ with uniform sampling ($\gamma = 1-\mu\eta$).

Corollary 2 *Assume* ***A****1 and* ***A****2. Let* $\kappa = \frac{L}{\mu}$ *denote the condition number and* $\theta \in (0, \frac{1}{5})$*. Select the parameters such that*

$$B \geq \min\left\{n, \left\lceil \frac{40\theta}{\epsilon} \cdot \frac{G_n}{L} \right\rceil\right\}, \gamma = 1-\eta\mu, m = \left\lceil \frac{1}{2L\mu\eta^2} \right\rceil;$$

$$T \geq \left\lceil \log\left(\frac{10D_0L}{\theta\epsilon}\right) \Big/ \log\left(\sqrt{\frac{1}{2\theta}}\right) \right\rceil,$$

Then $\mathbb{E}(f(\tilde{x}_T) - f(x^*)) \leq \epsilon$ *and*

$$\mathbb{E}C_{\text{comp}} = O\left(\left(n \wedge \frac{G_n}{L^2} \cdot \frac{L}{\epsilon} + \kappa\right)\log\frac{D_0L}{\epsilon}\right),$$

$$\mathbb{E}C_{\text{comm}} = O\left(\left(n \wedge \frac{G_n}{L^2} \cdot \frac{L}{\epsilon}\right)\log\frac{D_0L}{\epsilon}\right).$$

One might argue that the analysis in this section requires each f_i to be strongly convex in contrast to much of the literature which only requires f to be strongly convex. We also establish the result by only assuming A3 and state the analysis in Appendix C due to the constraint of length. However, unlike our analysis in Section 3.1 and Section 3.2, in this case the parameters are complicated and depend on the condition number $\bar{\kappa} \triangleq \frac{L}{\bar{\mu}}$ which is hard to obtain in practice. We should emphasize that other methods such as SVRG also face this problem. In particular, SVRG requires

$$\frac{1}{\bar{\mu}\eta(1-2L\eta)m} + \frac{2L\eta}{1-2L\eta} < 1,$$

which implies that $m \succeq \bar{\kappa}$. Thus intensive tuning is required to achieve the theoretical rate. If a case arises in practice in which only Assumption **A3** is satisfied, we recommend treating it as a non-strongly convex function and setting parameters as in Section 3.1.

4 More Detail on G_n

We show that G_n is the key component that determines whether SCSG is able to find an ϵ-approximate solution with less than a single pass of data. If $\|\nabla f_i\|$ is uniformly bounded, we immediately conclude $G_n = O(1)$. If the uniform boundedness is not satisfied, as in numerous applications, Lemma 2 provides a useful bound for G_n. In the stochastic setting (cf. Frostig et al., 2015), where f_i are i.i.d. with $\mathbb{E}\|\nabla f_1(\tilde{x}_0)\|^2 < \infty$, then the law of large numbers implies that

$$\frac{1}{n}\sum \|\nabla f_i(\tilde{x}_0)\|^2 \approx \mathbb{E}\|\nabla f_1(\tilde{x}_0)\|^2,$$

and by Lemma 2, we conclude that

$$\frac{G_n}{L^2} = O\left(D_0 + \frac{\mathbb{E}\|\nabla f_1(\tilde{x}_0)\|^2}{L^2}\right).$$

In many applications, D_0 is treated as $O(1)$. Since the second term is irrelevant to the sample size, it can also be treated as $O(1)$ and hence $\frac{G_n}{L^2} = O(1)$. In summary, G_n is well controlled if the data is homogeneous.

When data is heterogenous or the dimension is scaled as n, then G_n could be large. Heuristically, optimizing the objective with less than a single pass of data in this case is impossible since any subset of data loses a significant amount of information from the whole dataset. Here we present a pathological example in which $G_n = \Omega(n)$[1]. Let $f_i(x) = \|x - e_i\|^2$, where $e_i \in \mathbb{R}^n$ has j-th element equal to 0 and all others equal to 1. In this case, $G_n = n - 1 = \Omega(n)$. Recall that the SCSG has a multiplicative factor $\left(n \wedge \frac{G_n}{L\epsilon}\right)$ for minimizing both general convex functions and strongly convex functions and the factor is equal to n if $G_n = \Omega(n)$. Nonetheless, no algorithm performs well in this case since in general D_0 is also of order n and the performance of other algorithms depends on ϵ through $\frac{D_0L}{\epsilon}$. But such pathological example is arguably not realistic in practice since the goal of optimizing 1 is to extract common features from the data. If the data are extremely heterogeneous, the minimizer of (1) might be meaningless.

4.1 Estimating G_n for Generalized Linear Models

It has been shown in Section 3 that G_n and L determine the block size B, which is the key to reducing

[1]We thank Chi Jin for providing this example.

both the computation and the communication costs in SCSG. For this reason, good estimates of G_n and L are needed for efficiency. Lemma 2 presents a generic bound for G_n, provided L can be estimated. However, we found that more accurate estimates of G_n and L are possible for generalized linear models (GLMs) (McCullagh & Nelder, 1989), which include a broad class of models that are often used in practical applications. Estimation in the GLM setting reduces to solving (1) with $f_i(x) = \rho(y_i; a_i^T x)$, where a_i is the vector of predictors, y_i is the response and ρ is a loss function. This includes popular models such as linear regression, logistic regression, Huber regression and others. For multi-class problem in which $y_i \in \{0, 1, \ldots, K-1\}$ with $K > 2$, $f_i(x)$ can take the form $\rho(y_i; a_i^T x_1, \ldots, a_i^T x_{K-1})$ where $x \in \mathbb{R}^{d(K-1)}$ is the concatenation of $x_1, \ldots, x_{K-1}$; multi-class logistic regression is of this form. First consider the case with only one linear component. Let $\rho_2(z, w)$ denote $\frac{\partial}{\partial w}\rho(z, w)$ and $\rho_{22}(z, w)$ denote $\frac{\partial^2}{\partial w^2}\rho(z, w)$. Then G_n can be expressed as

$$G_n = \frac{1}{n}\sum_{i=1}^n \|\nabla f_i(x^*)\|^2 = \frac{1}{n}\sum_{i=1}^n |\rho_2(y_i, a_i^T x^*)|^2 \|a_i\|^2,$$

and L can be set as

$$L = \sup_i \sup_x \lambda_{\max}\left(\nabla^2 f_i(x)\right) = \sup_i \sup_x |\rho_{22}(y_i, a_i^T x)| \|a_i\|^2.$$

In many cases $\rho_{22}(z, w)$ and $\rho_2(z, w)$ are uniformly bounded; e.g., logistic regression. In the context of robust statistics (Huber, 2011), the loss function has a form $\rho(y_i, a_i^T x) = f(y_i - a_i^T x)$ where f has bounded first-order and second-order derivatives, in which cases the above conditions are satisfied.

Thus for GLMs with bounded $\rho_{22}(z, w)$ and $\rho_2(z, w)$, G_n and L can be bounded by

$$G_n \le M_1 \cdot \frac{1}{n}\sum_{i=1}^n \|a_i\|^2, \quad L \le M_2 \cdot \sup_i \|a_i\|^2. \tag{3}$$

The same argument applies to multi-class GLMs. In particular, via some algebra, it can be shown that $M_1 = 2, M_2 = 1$ for logistic and multi-class logistic regression; see Appendix C.3 for a proof. For moderately large n, we can calculate G_n and L directly. If n is prohibitively large, then we can estimate G_n and L based on a batch of data.

In some cases such as least squares regression, $\rho_{22}(z, w)$ is bounded but $\rho_2(z, w)$ is not. We can improve the bound in Lemma 2 by taking advantage of the explicit form of the solution. In particular, we have $x^* = (A^T A)^{-1} A^T y$ and it is easy to show that $G_n \le \sup \|a_i\|^2 \cdot \frac{\|y\|^2}{n}$.

5 Experiments

In this section, we illustrate the performance of SGSC by implementing it for multi-class logistic regression on the MNIST dataset[2]. We normalize the data into the range $[0, 1]$ by dividing each entry by 256. No regularization term is added and so the function to be minimized is

$$f(x) = \frac{1}{n}\sum_{i=1}^n \{\log(1 + \sum_{k=1}^{K-1} e^{a_i^T x_k}) - \sum_{k=1}^{K-1} I(y_i = k) a_i^T x_k\},$$

where $n = 60000, K = 10$, $y_i \in \{0, 1, \ldots, 9\}$, $a_i \in \mathbb{R}^{785}$ including $28 \times 28 = 784$ pixels plus an intercept term 1 and $x = (x_1, \ldots, x_9) \in \mathbb{R}^{785 \times 9} = \mathbb{R}^{7065}$. Direct computation shows that $G_n = 174.25$ and $L = 292.82$. Thu the highest stepsize with a convergence guarantee is $\eta_0 = \frac{1}{2L} = 0.0017$. We treat this stepsize as a benchmark and also try three more aggressive stepsizes, $\eta_1 = 4\eta_0, \eta_2 = 10\eta_0, \eta_3 = 30\eta_0$.

Suppose that our target accuracy is $\epsilon = 0.001$. Corollary 1 then implies that $B = 2220$. Based on this benchmark, we try three batch sizes $B \in \{250, 2500, 7500\}$ as well as SVRG with $m = 60000$. We record the iterates x_t every half pass through the data where t denotes the number of passes. Although the theory provides a guarantee for $\mathbb{E}(f(x_t) - f(x^*)) \le \epsilon$, the optimal value x^* is unknown in practice. Instead we report $\|\nabla f(\bar{x}_t)\|^2$ as an accuracy measure. This value generally has the same order as $f(x_t) - f(x^*)$. To illustrate the average performance, we run the algorithm for 20 times in each case and report the norm of the gradient evaluated at the average of iterates. In all cases, the initial value is set to be a zero vector. The top row of Figure 1 displays the results. It is clear that SCSG with a batch size as small as 250 converges faster than SVRG in the first 50 passes. To achieve an accuracy of 0.001, SCSG requires less than five passes of data while SVRG requires around ten passes through the data. In the best-tuned case ($\eta = 4\eta_0$), SCSG only requires two passes. Moreover, it is seen that performance degrades for $\eta = 30\eta_0$. This implies that our benchmark stepsize is valid in this case; in practice, we suggest starting from the benchmark stepsize η_0 and trying multiples of η_0 in $[\eta_0, 30\eta_0]$.

Since our theory indicates that SCSG is able to achieve good accuracy even with less than a single pass through the dataset, we repeat the above procedure while recording the iterates $\bar{x}_t$ every 0.1 of a pass through the data, where the batch size B is selected from $\{250, 1000, 2500\}$. The bottom row of Figure 1 displays the results. It is seen that $B = 250$ yields the fastest convergence in the first pass and in the best

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

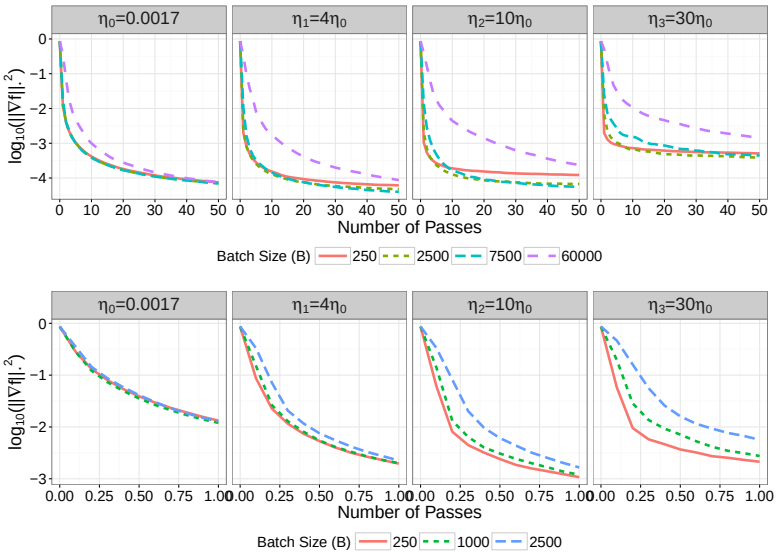

Figure 1: Plot of $\|\nabla f(x_t)\|_2^2$ versus the number of data passes with different batch sizes and stepsizes: (Top) performance in first 50 passes; (Bottom) performance within one pass.

tuned case, $\eta = 10\eta_0$, an accuracy of 0.01 is achieved within only 0.25 passes through the data for $B = 250$ and $B = 1000$. Note that a batch of MNIST dataset with size 1000 requires no more than 2.6MB of memory and it involves roughly $0.25n/2B = 7.5 < 8$ accesses of the disk. In other words, a commodity machine with a very modest memory suffices to achieve reasonable accuracy.

6 Discussion

We propose SCSG as a member of the SVRG family of algorithms, proving its superior performance in terms of both computation and communication cost. Both complexities are independent of sample size when the required accuracy is small, for generalized linear models which are widely used in practice. The real data example also validates our theory.

For practical use, we develop an automatic parameter-tuning procedure. For a dataset for which scanning is possible, we recommend using the benchmark stepsize and batch size produced by automatic tuning and running SCSG for a few steps. If the desired accuracy is not achieved, then double the stepsize or the batch size. The doubling continues until the batch cannot fit into memory. For massive data for which scanning is too costly, we recommend estimating G_n and L on a random batch and repeating the above procedure.

We plan to explore several variants of SCSG in future work. For example, a non-uniform sampling scheme can be applied to SGD steps to leverage the Lipschitz constants L_i as in SVRG. More interestingly, we can consider a better sampling scheme for $\mathcal{I}_j$ by putting more weight on influential observations.

As a final comment, we found that the previous complexity analysis focuses on the high-accuracy computation for which the dependence on the sample size n and condition number κ is of major concern. The low-accuracy regime is unfortunately under-studied theoretically even though it is commonly encountered in practice. We advocate taking all three parameters, namely n, κ and ϵ, into consideration and distinguishing the analyses for high-accuracy computation and low-accuracy computation as standard practice in the literature.

Acknowledgements

We thank the anonymous reviewers for their helpful comments and Chi Jin for his valuable example, which greatly improved this work. This work was supported by the Mathematical Data Science program of the Office of Naval Research under grant N00014-15-1-2670.

References

Agarwal, A., & Bottou, L. (2015). A lower bound for the optimization of finite sums. In *International conference on machine learning* (p. 78-86).

Allen-Zhu, Z. (2017). Katyusha: The first direct acceleration of stochastic gradient methods. In *Stoc.*

Allen-Zhu, Z., & Yuan, Y. (2016). Improved SVRG for non-strongly-convex or sum-of-non-convex ob-

jectives. In *International conference on machine learning*.

Arjevani, Y., & Shamir, O. (2015). Communication complexity of distributed convex learning and optimization. In *Advances in neural information processing systems* (pp. 1756–1764).

Daneshmand, H., Lucchi, A., & Hofmann, T. (2016). Starting small-learning with adaptive sample sizes. In *International conference on machine learning* (pp. 1463–1471).

Defazio, A., Bach, F., & Lacoste-Julien, S. (2014). SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems* (pp. 1646–1654).

Frostig, R., Ge, R., Kakade, S. M., & Sidford, A. (2015). Competing with the empirical risk minimizer in a single pass. In *Conference on learning theory*.

Harikandeh, R., Ahmed, M. O., Virani, A., Schmidt, M., Konečnỳ, J., & Sallinen, S. (2015). Stop wasting my gradients: Practical SVRG. In *Advances in neural information processing systems* (pp. 2242–2250).

Hazan, E., Agarwal, A., & Kale, S. (2007). Logarithmic regret algorithms for online convex optimization. *Machine Learning*, *69*(2-3), 169–192.

Huber, P. J. (2011). *Robust statistics*. Springer.

Jaggi, M., Smith, V., Takác, M., Terhorst, J., Krishnan, S., Hofmann, T., & Jordan, M. I. (2014). Communication-efficient distributed dual coordinate ascent. In *Advances in neural information processing systems* (pp. 3068–3076).

Johnson, R., & Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems* (pp. 315–323).

Konečnỳ, J., McMahan, B., & Ramage, D. (2015). Federated optimization: Distributed optimization beyond the datacenter. *ArXiv e-prints abs/1511.03575*.

Lee, J., Ma, T., & Lin, Q. (2015). Distributed stochastic variance reduced gradient methods. *ArXiv e-prints abs/1507.07595*.

Lin, H., Mairal, J., & Harchaoui, Z. (2015). A universal catalyst for first-order optimization. In *Advances in neural information processing systems* (pp. 3384–3392).

McCullagh, P., & Nelder, J. A. (1989). *Generalized Linear Models*. CRC Press.

Nemirovski, A., Juditsky, A., Lan, G., & Shapiro, A. (2009). Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, *19*(4), 1574–1609.

Nitanda, A. (2014). Stochastic proximal gradient descent with acceleration techniques. In *Advances in neural information processing systems* (pp. 1574–1582).

Nitanda, A. (2016). Accelerated stochastic gradient descent for minimizing finite sums. In *Proceedings of the 19th international conference on artificial intelligence and statistics* (pp. 195–203).

Rakhlin, A., Shamir, O., & Sridharan, K. (2012). Making gradient descent optimal for strongly convex stochastic optimization. In *International conference on machine learning* (pp. 449–456).

Reddi, S. J., Hefny, A., Sra, S., Póczos, B., & Smola, A. J. (2015). On variance reduction in stochastic gradient descent and its asynchronous variants. In *Advances in neural information processing systems* (pp. 2629–2637).

Reddi, S. J., Hefny, A., Sra, S., Póczós, B., & Smola, A. J. (2016). Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*.

Roux, N. L., Schmidt, M., & Bach, F. (2012). A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in neural information processing systems* (pp. 2663–2671).

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

Shamir, O., Srebro, N., & Zhang, T. (2014). Communication-efficient distributed optimization using an approximate Newton-type method. In *International conference on machine learning* (Vol. 32, pp. 1000–1008).

Woodworth, B. E., & Srebro, N. (2016). Tight complexity bounds for optimizing composite objectives. In *Advances in neural information processing systems* (pp. 3639–3647).

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

Zhang, Y., & Lin, X. (2015). Disco: Distributed optimization for self-concordant empirical loss. In *International conference on machine learning* (pp. 362–370).

Zhang, Y., Wainwright, M. J., & Duchi, J. C. (2012). Communication-efficient algorithms for statistical optimization. In *Advances in neural information processing systems* (pp. 1502–1510).