Non-convex Finite-Sum Optimization Via SCSG Methods

Authored by:

Michael I. Jordan
Lihua Lei
Cheng Ju
Jianbo Chen

Abstract

We develop a class of algorithms, as variants of the stochastically controlled stochastic gradient (SCSG) methods , for the smooth nonconvex finite-sum optimization problem. Only assuming the smoothness of each component, the complexity of SCSG to reach a stationary point with $E —nabla f(x)—{2}le epsilon$ is $O(min{epsilon{-5/3}, epsilon{-1}n{2/3}})$, which strictly outperforms the stochastic gradient descent. Moreover, SCSG is never worse than the state-of-the-art methods based on variance reduction and it significantly outperforms them when the target accuracy is low. A similar acceleration is also achieved when the functions satisfy the Polyak-Lojasiewicz condition. Empirical experiments demonstrate that SCSG outperforms stochastic gradient methods on training multi-layers neural networks in terms of both training and validation loss.

1 Paper Body

We study smooth non-convex finite-sum optimization problems of the form n 1X fi (x) min f (x) = n i=1 x?Rd

(1)

where each component fi (x) is possibly non-convex with Lipschitz gradients. This generic form captures numerous statistical learning problems, ranging from generalized linear models [22] to deep neural networks [19]. In contrast to the convex case, the non-convex case is comparatively under-studied. Early work focused on the asymptotic performance of algorithms [11, 7, 29], with non-asymptotic complexity bounds emerging more recently [24]. In recent years, complexity results have been derived for both gradient methods [13, 2, 8, 9] and stochastic gradient methods [12, 13, 6, 4, 26, 27, 3]. Unlike in the convex case, in the non-convex case one can not expect a gradient-based algorithm to converge to the global minimum if only smoothness is assumed. As a consequence, instead

of measuring functionvalue suboptimality Ef (x) ? inf x f (x) as in the convex case, convergence is generally measured in terms of the squared norm of the gradient; i.e., Ek?f (x)k2 . We summarize the best available rates 1 in Table 1. We also list the rates for Polyak-Lojasiewicz (P-L) functions, which will be defined in Section 2. The accuracy for minimizing P-L functions is measured by Ef (x) ? inf x f (x). 1

It is also common to use Ek?f (x)k to measure convergence; see, e.g. [2, 8, 9, 3]. Our results pcan be readily 2 transferred to this alternative measure by using Cauchy-Schwartz inequality, Ek?f (x)k ? Ek?f (x)k ? , although not vice versa. The rates under this alternative can be made comparable to ours by replacing ? by ?. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Table 1: Computation complexity of gradient methods and stochastic gradient methods for the finite-sum non-convex optimization problem (1). The second and third columns summarize the rates in the smooth and P-L cases respectively. ? is the P-L constant and H? is the variance of a stochastic gradient. These quantities are defined in Section 2. The final column gives additional required ? is assumptions beyond smoothness or the P-L condition. The symbol ? denotes a minimum and O(?) the usual Landau big-O notation with logarithmic terms hidden. Smooth Polyak-Lojasiewicz additional cond. Gradient Methods GD Best available

? O

O n? [24, 13]

n ? 7/8 O ? [9] n ? 5/6 O [9]

n ?

?

Stochastic Gradient Methods

SGD O ?12 [24, 26]

2/3 Best available O n + n ? [26, 27]

1 n2/3 ? 5/3 SCSG O ? ? ?

O

[25, 17]

-

-

smooth gradient smooth Hessian

1 ?2 ?

H? = O(1)

[17]

2/3 n ? n+ O [26, 27] ?

? (1 ? n) + 1 (1 ? n)2/3 O ?? ? ??

H? = O(1)

As in the convex case, gradient methods have better dependence on ? in the non-convex case but worse dependence on n. This is due to the requirement of computing a full gradient. Comparing the complexity of SGD and the best achievable rate for stochastic gradient methods, achieved via variance-reduction

methods, the dependence on ? is significantly improved in the latter case. However, unless ? ¡¡ n?1/2 , SGD has similar or even better theoretical complexity than gradient methods and existing variance-reduction methods. In practice, it is often the case that n is very large (105 ? 109) while the target accuracy is moderate (10?1 ? 10?3). In this case, SGD has a meaningful advantage over other methods, deriving from the fact that it does not require a full gradient computation. This motivates the following research question: Is there an algorithm that ? achieves/beats the theoretical complexity of SGD in the regime of modest target accuracy; ? and achieves/beats the theoretical complexity of existing variance-reduction methods in the regime of high target accuracy? The question has been partially answered in the convex case by [21] in their formulation of the stochastically controlled stochastic gradient (SCSG) methods. When the target accuracy is low,

SCSG has the same O ??2 rate as SGD but with a much smaller data-dependent constant factor (which does not even require bounded gradients). When the target accuracy is high, SCSG achieves the same rate as the best non-accelerated methods, O(n?). Despite the gap between this and the optimal rate, SCSG is the first known algorithm that provably achieves the desired performance in both regimes. In this paper, we generalize SCSG to the non-convex setting which, surprisingly, provides a completely affirmative answer to the question raised before. By only assuming smoothness of each component as in almost all other works, SCSG is always O ??1/3 faster than SGD and is never worse than recently developed stochastic gradient methods that achieve the best rate.When ? ¿¿ n1 , SCSG is at least O((?n)2/3) faster than the best SVRG-type algorithms. Comparing with the gradient methods, SCSG has a better convergence rate provided ? ¿¿ n?6/5 , which is the common setting in practice. Interestingly, there is a parallel to recent advances in gradient methods; [9] improved the classical O(??1) rate of gradient descent to O(??5/6); this parallels the improvement of SCSG over SGD from O(??2) to O(??5/3). Beyond the theoretical advantages of SCSG, we also show that SCSG yields good empirical performance for the training of multi-layer neural networks. It is worth emphasizing that the mechanism by which SCSG achieves acceleration (variance reduction) is qualitatively different from other speed-up 2

techniques, including momentum [28] and adaptive stepsizes [18]. It will be of interest in future work to explore combinations of these various approaches in the training of deep neural networks. The rest of paper is organized as follows: In Section 2 we discuss our notation and assumptions and we state the basic SCSG algorithm. We present the theoretical convergence analysis in Section 3. Experimental results are presented in Section 4. All the technical proofs are relegated to the Appendices.

2

Notation, Assumptions and Algorithm

We use k ? k to denote the Euclidean norm and write min{a, b} as a ? b for brevity throughout the ? which hides logarithmic terms, will only be used to maximize readibility in paper. The notation O, our presentation but will not be used in the formal analysis. We define computation cost using the IFO

framework of [1] which assumes that sampling an index i and P accessing the pair (?fi (x), fi (x)) incur a unit of cost. For brevity, we write ?fI (x) for 1 i?I ?fi (x). Note that calculating ?fI (x) incurs —I— units of computational cost. x is called —I— an ?-accurate solution iff Ek?f (x)k2 ? ?. The minimum IFO complexity to reach an ?-accurate solution is denoted by Ccomp (?). Recall that a random variable N has a geometric distribution, N ? Geom(?), if N is supported on the non-negative integers 2 with P (N = k) = ? k (1 ? ?),

?k = 0, 1, . . .

An elementary calculation shows that EN ?Geom(?) =

? . 1??

(2)

To formulate our complexity bounds, we define f ? = inf f (x), x

?f = f (? x0) ? f ? .

Further we define H? as an upper bound of the variance of stochastic gradients, i.e. n

H? = sup x

1X k?fi (x) ? ?f (x)k2 . n i=1

(3)

The assumption A1 on the smoothness of individual functions will be made throughout this paper. A1 fi is differentiable with k?fi (x) ? ?fi (y)k ? Lkx ? yk for some L ¡ ? and all i ? {1, . . . , n}. As a direct consequence of assumption A1, it holds for any x, y ? Rd that L L ? kx ? yk2 ? fi (x) ? fi (y) ? h?fi (y), x ? yi ? kx ? yk2 . 2 2

(4)

In this paper, we also consider the following Polyak-Lojasiewicz (PL) condition [25]. It is weaker than strong convexity as well as other popular conditions that appeared in optimization literature; see [17] for an extensive discussion. A2 f (x) satisfies the P-L condition with ? ¿ 0 if k?f (x)k2 ? 2?(f (x) ? f (x?)) where x? is the global minimum of f . 2

Here we allow N to be zero to facilitate the analysis.

3

2.1

Generic form of SCSG methods

The algorithm we propose in this paper is similar to that of [14] except (critically) the number of inner loops is a geometric random variable. This is an essential component in the analysis of SCSG, and, as we will show below, it is key in allowing us to extend the complexity analysis for SCSG to the non-convex case. Moreover, that algorithm that we present here employs a mini-batch procedure in the inner loop and outputs a random sample instead of an average of the iterates. The pseudo-code is shown in Algorithm 1. Algorithm 1 (Mini-Batch) Stochastically Controlled Stochastic Gradient (SCSG) method for smooth non-convex finite-sum objectives Inputs: Number of stages T , initial iterate x ?0 , stepsizes (?j)Tj=1 , batch sizes (Bj)Tj=1 , mini-batch sizes (bj)Tj=1 . Procedure 1: for j = 1, 2, ? ? ? , T do 2: Uniformly sample a batch Ij ? {1, ? ? ? , n} with —Ij — = Bj ; 3: gj ? ?fIj (? xj?1); (j) 4: x0 ? x ?j?1 ; 5: Generate Nj ? Geom (bj /(Bj + bj)); 6: for k = 1, 2, ? ? ? , Nj do 7:

Randomly pick I?k?1 ? [n] with —I?k?1 — = bj ; (j) (j) (j) 8: ?k?1 ? ?fI?k?1 (xk?1) ? ?fI?k?1 (x0) + gj ; (j)

(j)

(j)

9: xk ? xk?1 ? ?j ?k?1 ; 10: end for (j) 11: x ?j ? xNj ; 12: end for

Output: (Smooth case) Sample x ??T from (? xj)Tj=1 with P (? x?T = x ?j) ? ?j Bj /bj ; (P-L case) x ?T . As seen in the pseudo-code, the SCSG method consists of multiple epochs. In the j-th epoch, a minibatch of size Bj is drawn uniformly from the data and a sequence of mini-batch SVRG-type updates are implemented, with the total number of updates being randomly generated from a geometric distribution, with mean equal to the batch size. Finally it outputs a random sample from {? xj }Tj=1 . This is the standard way, proposed by [23], as opposed to computing arg minj?T k?f (? xj)k which requires additional overhead. By (2), the average total cost is T X

(Bj + bj ? ENj) =

j=1

T X

(Bj + bj ?

i=1

T X Bj Bj .)=2 bj j=1

(5)

Define T (?) as the minimum number of epochs such that all outputs afterwards are ?-accurate solutions, i.e. T (?) = min{T : Ek?f (? x?T 0)k ? ? for all T 0 ? T }. Recall the definition of Ccomp (?) at the beginning of this section, the average IFO complexity to reach an ?-accurate solution is T (?) X ECcomp (?) ? 2 Bj . j=1

2.2

Parameter settings

The generic form (Algorithm 1) allows for flexibility in both stepsize, ?j , and batch/mini-batch size, (Bj , bj). In order to minimize the amount of tuning needed in practice, we provide several default settings which have theoretical support. The settings and the corresponding complexity results are summarized in Table 2. Note that all settings fix bj = 1 since this yields the best rate as will be shown in Section 3. However, in practice a reasonably large mini-batch size bj might be favorable due to the acceleration that could be achieved by vectorization; see Section 4 for more discussions on this point. 4

Table 2: Parameter settings analyzed in this paper. Bj bj Type of Objectives ECcomp (?)

2/3 1 O 1? ? n 1 Smooth O ?5/3 ? n?

3 1 n2/3 ? 5/3 j2 ?n 1 Smooth O ? ? ?

1 1 1 1 ? (? n) + (? n)2/3 O ?? ? n 1 Polyak-Lojasiewicz O ?? ? ??

?j Version 1 Version 2 Version 3

3 3.1

1 2LB 2/3 1 2/3 2LBj 1 2/3 2LBj

Convergence Analysis One-epoch analysis

First we present the analysis for a single epoch. Given j, we define ej = ?fIj (? xj?1) ? ?f (? xj?1).

(6)

(j)

(j)

As shown in [14], the gradient update ?k is a biased estimate of the gradient ?f (xk) conditioning on the current random index ik . Specifically, within the j-th epoch, (j)

(j)

(j)

(j)

(j)

EI?k ?k = ?f (xk) + ?fIj (x0) ? ?f (x0) = ?f (xk) + ej . This reveals the basic qualitative difference between SVRG and SCSG. Most of the novelty in our (j) analysis lies in dealing with the extra term ej . Unlike [14], we do not assume kxk ? x? k to be bounded since this is invalid in unconstrained problems, even in convex cases. By careful analysis of primal and dual gaps [cf. 5], we find that the stepsize ?j should scale as 2 (Bj /bj)? 3 . Then same phenomenon has also been observed in [26, 27, 4] when bj = 1 and Bj = n. 2

Theorem 3.1 Let ?j L = ?(Bj /bj)? 3 . Suppose ? ? 16 and Bj ? 9 for all j, then under Assumption A1, 13 bj 5L 6I(Bj ¡ n) 2 Ek?f (? xj)k ? ? E(f (? xj?1) ? f (? xj)) + ? H? . (7) ? Bj Bj The proof is presented in Appendix B. It is not surprising that a large mini-batch size will increase the theoretical complexity as in the analysis of mini-batch SGD. For this reason we restrict most of our subsequent analysis to bj ? 1. 3.2

Convergence analysis for smooth non-convex objectives

When only assuming smoothness, the output x ??T is a random element from (? xj)Tj=1 . Telescoping (7) over all epochs, we easily obtain the following result. Theorem 3.2 Under the specifications of Theorem 3.1 and Assumption A1,

P ?1 ?2 T 5L ?f + 6 bj 3 Bj 3 I(Bj ¡ n) H? j=1 ? . Ek?f (? x?T)k2 ? 1 PT ? 13 3 j=1 bj Bj This theorem covers manyexisting results. When Bj = n and bj = 1, Theorem 3.2 implies that

L?f L?f and hence T (?) = O(1+ ?n1/3). This yields the same complexity bound T n1/3 2/3 n L?f ECcomp (?) = O(n +) as SVRG [26]. On the ? other hand, when bj = Bj ? B for some L?f H? ? 2 B ¡ n, Theorem 3.2 implies that Ek?f (? xT)k = O T + B . The second term can be made ?

L?f L?f H? O(?) by setting B = O H? . Under this setting T (?) = O and EC (?) = O . 2 comp ? ?

Ek?f (? x?T)k2 = O

This is the same rate as in [26] for SGD. 5

However, both of the above settings are suboptimal since they either set the batch sizes Bj too large or set the mini-batch sizes bj too large. By Theorem 3.2, SCSG can be regarded as an interpolation between SGD and SVRG. By leveraging these two parameters, SCSG is able to outperform both methods. We start from considering a constant batch/mini-batch size Bj ? B, bj ? 1.

Similar to SGD and ? SCSG, B should be at least O(H?). In applications like the training of neural networks, the required accuracy is moderate and hence a small batch size suffices. This is particularly important since the gradient can be computed without communication overhead, which is the bottleneck of SVRG-type algorithms. As shown in Corollary 3.3 below, the complexity of SCSG beats both SGD and SVRG. Corollary 3.3 (Constant batch sizes) Set

bj ? 1,

Bj ? B = min

12H? ,n , ?

?j ? ? =

1 2

6LB 3

.

Then it holds that

ECcomp (?) = O

? 32 ! H? L?f H ?n + ? ?n . ? ? ?

Assume that L?f , H? = O(1), the above bound can be simplified to

32 !

1 1 1 ?n + ? ?n =O ECcomp (?) = O ? ? ?

2

n3 5 ? ? ?3 1

! .

When the target accuracy is high, one might consider a sequence of increasing batch sizes. Heuristically, a large batch is wasteful at the early stages when the iterates are inaccurate. Fixing the batch 3 size to be n as in SVRG is obviously suboptimal. Via an involved analysis, we find that Bj ? j 2 gives the best complexity among the class of SCSG algorithms. Corollary 3.4 (Time-varying batch sizes) Set bj ? 1,

n 3 o Bj = min dj 2 e, n ,

?j =

1 2

.

6LBj3 Then it holds that

ECcomp (?) = O min

!

? 2 5 5 n3 H 5 ? 53 ? 3 3 (L?f) + (H) log + ? (L?f + H log n) . ,n 5 ? ? ?3 (8) 1

The proofs of both Corollary 3.3 and Corollary 3.4 are presented in Appendix C. To simplify the bound (8), we assume that L?f , H? = O(1) in order to highlight the dependence on ? and n. Then (8) can be simplified to ! ! !

2 2 2 3 log n 3 3 5 5 1 1 n 1 n 1 n 5 ? ? 3 + ECcomp (?) = O ? n3 + =O =O . 5 log 5 ? n 5 ? ? ? ? ? ?3 ?3 ?3

3 The log-factor log5 1? is purely an artifact of our proof. It can be reduced to log 2 +? 3 3 ? ¿ 0 by setting Bj ? j 2 (log j) 2 +? ; see remark 1 in Appendix C. 3.3

1 ?

for any

Convergence analysis for P-L objectives

When the component fi (x) satisfies the P-L condition, it is known that the global minimum can be found efficiently by SGD [17] and SVRG-type algorithms [26, 4]. Similarly, SCSG can also achieve this. As in the last subsection, we start from a generic result to bound E(f (? xT) ? f ?) and then consider specific settings of the parameters as well as their complexity bounds. 6

1

Theorem 3.5 Let ?j =

5Lbj3 1

1

. Then under the same settings of Theorem 3.2,

??Bj3 +5Lbj3

E(f (? xT) ? f ?) ? ?T ?T ?1 . . . ?1 ? ?f + 6?H? ?

T X ?T ?T ?1 . . . ?j+1 ? I(Bj ¡ n) 1

j=1

2

.

??Bj + 5Lbj3 Bj3

The proofs and additional discussion are presented in Appendix D. Again, Theorem 3.5 covers existing complexity bounds for both SGD and SVRG. In fact, when Bj = bj ? B as in SGD, via some calculation, we obtain that !

T L H? ? E(f (? xT) ? f) = O ? ?f + . ?+L ?B ?

L The second term can be made O(?) by setting B = O(H ??), in which case T (?) = O(? log ?

?f ?

). As

O(LH ?2 ?

a result, the average cost to reach an ?-accurate solution is ECcomp (?) =), which is the same as [17]. On the other hand, when Bj ? n and bj ? 1 as in SVRG, Theorem 3.5 implies that ! T

L ? ? ?f . E(f (? xT) ? f) = O 1 ?n 3 + L

2/3 This entails that T (?) = O (1 + ?n11/3) log 1? and hence ECcomp (?) = O (n + n ?) log 1? , which is the same as [26]. By leveraging the batch and mini-batch sizes, we obtain a counterpart of Corollary 3.3 as below. Corollary 3.6 Set

bj ? 1,

Bj ? B = min

12H? ,n , ??

?j ? ? =

1 2

6LB 3

Then it holds that (ECcomp (?) = O

!

32) 1 H? H? ?f ?n + ?n . log ?? ? ?? ?

Recall the results from Table 1, SCSG is O ?1 + (??)11/3 faster than SGD and is never worse than SVRG. When both ? and ? are moderate, the acceleration of SCSG over SVRG is significant. Unlike the smooth case, we do not find any possible choice of setting that can achieve a better rate than Corollary 3.6.

4

Experiments

We evaluate SCSG and mini-batch SGD on the MNIST dataset with (1) a three-layer fully-connected neural network with 512 neurons in each layer (FCN for short) and (2) a standard convolutional neural network LeNet [20] (CNN for short), which has two convolutional layers with 32 and 64 filters of size 5 ? 5 respectively, followed by two fully-connected layers with output size 1024 and 10. Max pooling is applied after each convolutional layer. The MNIST dataset of handwritten digits has 50, 000 training examples and 10, 000 test examples. The digits have been size-normalized and centered in a fixed-size image. Each image is 28 pixels by 28 pixels. All experiments were carried out on an Amazon p2.xlarge node with a NVIDIA GK210 GPU with algorithms implemented in TensorFlow 1.0. Due to the memory issues, sampling a chunk of data is costly. We avoid this by modifying the inner loop: instead of sampling mini-batches from the whole dataset, we split the batch Ij into Bj /bj mini-batches and run SVRG-type updates sequentially on each. Despite the theoretical advantage of setting bj = 1, we consider practical settings bj ¿ 1 to take advantage of the acceleration obtained 7

by vectorization. We initialized parameters by TensorFlow?s default Xavier uniform initializer. In all experiments below, we show the results corresponding to the best-tuned stepsizes. We consider three algorithms: (1) SGD with a fixed batch size B ? {512, 1024}; (2) SCSG with a fixed batch size B ? {512, 1024} and a fixed mini-batch size b = 32; (3) SCSG with time-varying batch sizes Bj = dj 3/2 ? ne and bj = dBj /32e. To be clear, given T epochs, the IFO complexity PT of the three algorithms are T B, 2T B and 2 j=1 Bj , respectively. We run each algorithm with 20 passes of data. It is worth mentioning that the largest batch size in Algorithm 3 is d2751.5 e = 4561, which is relatively small compared to the sample size 50000. We plot in Figure 1 the training and the validation loss against the IFO complexity?i.e., the number of passes of data?for fair comparison. In all cases, both versions of SCSG outperform SGD, especially in terms of training loss. SCSG with time-varying batch sizes always has the best performance and it is more stable than SCSG with a fixed batch size. For the latter, the acceleration is more significant after increasing the batch size to 1024. Both versions of SCSG provide strong evidence that variance reduction can be achieved efficiently without evaluating the full gradient.

4

2

6

8

10

#grad / n

12

10-1
4
2
6
8
10
#grad / n
12
14
0
2
4
6
8
10
#grad / n
12
14
100
10-1
0
2
4
6
8
10
#grad / n
12
14
10-1
10-2 0
2
4
6
8
10
#grad / n
12
14
100
10-1 0
2
4
6
8
10

#grad / n
12
FCN
Training Log-Loss
Training Log-Loss
10-2
14
100
0
10-1
FCN
SGD (B = 512) SCSG (B = 512, b = 32) SCSG (B = j1.5, B/b = 32)
100
14
Validation Log-Loss
Validation Log-Loss
0
SGD (B = 1024) SCSG (B = 1024, b = 32) SCSG (B = j1.5, B/b = 32)
100
Validation Log-Loss
10-2
Training Log-Loss
10-1
CNN
Validation Log-Loss
Training Log-Loss
CNN
SGD (B = 512) SCSG (B = 512, b = 32) SCSG (B = j1.5, B/b = 32)
100
SGD (B = 1024) SCSG (B = 1024, b = 32) SCSG (B = j1.5, B/b = 32)
100
10-1
10-2 0
2
4
#grad / n
6
8
10
12
14
2
4
#grad / n
6
8

10
12
14
100
10-1 0

Figure 1: Comparison between two versions of SCSG and mini-batch SGD of training loss (top row) and validation loss (bottom row) against the number of IFO calls. The loss is plotted on a log-scale. Each column represents an experiment with the setup printed on the top.

CNN
scsg (B=j1.5, B/b=16) sgd (B=j1.5)
10 0 Validation Log Loss
10 0 Training Log Loss
CNN
scsg (B=j1.5, B/b=16) sgd (B=j1.5)
10 -1
10 -1
0
50
100 150 Wall Clock Time (in second)
200
0
50
100 150 Wall Clock Time (in second)
200

Figure 2: Comparison between SCSG and mini-batch SGD of training loss and validation loss with a CNN loss, against wall clock time. The loss is plotted on a log-scale. Given 2B IFO calls, SGD implements updates on two fresh batches while SCSG replaces the second batch by a sequence of variance reduced updates. Thus, Figure 1 shows that the gain due to variance reduction is significant when the batch size is fixed. To further explore this, we compare SCSG with time-varying batch sizes to SGD with the same sequence of batch sizes. The results corresponding to the best-tuned constant stepsizes are plotted in Figure 3a. It is clear that the benefit from variance reduction is more significant when using time-varying batch sizes. We also compare the performance of SGD with that of SCSG with time-varying batch sizes against wall clock time, when both algorithms are implemented in TensorFlow and run on a Amazon p2.xlarge node with a NVIDIA GK210 GPU. Due to the cost of computing variance reduction terms in SCSG, each update of SCSG is slower per iteration compared to SGD. However, SCSG makes faster progress 8

in terms of both training loss and validation loss compared to SCD in wall clock time. The results are shown in Figure 2.

10-2 0
2
4
6

8
10
#grad / n
12
14
SGD SCSG
10-1
10-2 0
2
4
6
8
10
#grad / n
12
CNN
14
B/b = 2.0 B/b = 5.0 B/b = 10.0 B/b = 16.0 B/b = 32.0
100
10-1
10-2
0
2
4
6
8
10
#grad / n
12
14
FCN Training Log-Loss
10-1
FCN 100
Training Log-Loss
SGD SCSG
Training Log-Loss
Training Log-Loss
CNN 100
B/b = 2.0 B/b = 5.0 B/b = 10.0 B/b = 16.0 B/b = 32.0
100
10-1
0
2
4
6
8

10
#grad / n
12
14
(b) SCSG with different Bj /bj
(a) SCSG and SGD with increasing batch sizes

Finally, we examine the effect of Bj /bj , namely the number of mini-batches within an iteration, since it affects the efficiency in practice where the computation time is not proportional to the batch size. Figure 3b shows the results for SCSG with Bj = dj 3/2 ? ne and dBj /bj e ? {2, 5, 10, 16, 32}. In general, larger Bj /bj yields better performance. It would be interesting to explore the tradeoff between computation efficiency and this ratio on different platforms.

5
Conclusion and Discussion

We have presented the SCSG method for smooth, non-convex, finite-sum optimization problems. SCSG is the first algorithm that achieves a uniformly better rate than SGD and is never worse than SVRG-type algorithms. When the target accuracy is low, SCSG significantly outperforms the SVRG-type algorithms. Unlike various other variants of SVRG, SCSG is clean in terms of both implementation and analysis. Empirically, SCSG outperforms SGD in the training of multi-layer neural networks. Although we only consider the finite-sum objective in this paper, it is straightforward to extend SCSG to the general stochastic optimization problems where the objective can be written as E??F f (x; ?): at the beginning of j-th epoch a batch of i.i.d. sample (?1 , . . . , ?Bj) is drawn from the distribution F and Bj 1 X ?f (? xj?1 ; ?i) (see line 3 of Algorithm 1); gj = Bj i=1 (k) (k) at the k-th step, a fresh sample (??1 , . . . , ??bj) is drawn from the distribution F and (j)

?k?1 =

bj bj 1 X 1 X (j) (k) (j) (k) ?f (xk?1 ; ??i) ? ?f (x0 ; ??i) + gj bj i=1 bj i=1

(see line 8 of Algorithm 1).

Our proof directly carries over to this case, by simply suppressing the term I(Bj ¡ n), and yields ? ?5/3) for smooth non-convex objectives and the bound O(? ? ?1 ??1 ? ??5/3 ??2/3) for the bound O(? P-L objectives. These bounds are obtained simply by setting n = ? in our convergence analysis. Compared to momentum-based methods [28] and methods with adaptive stepsizes [10, 18], the mechanism whereby SCSG achieves acceleration is qualitatively different: while momentum aims at balancing primal and dual gaps [5], adaptive stepsizes aim at balancing the scale of each coordinate, and variance reduction aims at removing the noise. We believe that an algorithm that combines these three techniques is worthy of further study, especially in the training of deep neural networks where the target accuracy is modest.

Acknowledgments The authors thank Zeyuan Allen-Zhu, Chi Jin, Nilesh Tripuraneni and anonymous reviewers for helpful discussions.

2 References

[1] Alekh Agarwal and Leon Bottou. A lower bound for the optimization of finite sums. ArXiv e-prints abs/1410.0723, 2014. 9

[2] Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. Finding approximate local minima for nonconvex optimization in linear time. arXiv preprint arXiv:1611.01146, 2016. [3] Zeyuan Allen-Zhu. Natasha: Faster stochastic non-convex optimization via strongly non-convex parameter. arXiv preprint arXiv:1702.00763, 2017. [4] Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. ArXiv e-prints abs/1603.05643, 2016. [5] Zeyuan Allen-Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. arXiv preprint arXiv:1407.1537, 2014. [6] Zeyuan Allen-Zhu and Yang Yuan. Improved SVRG for non-strongly-convex or sum-of-nonconvex objectives. ArXiv e-prints, abs/1506.01972, 2015. [7] Dimitri P Bertsekas. A new class of incremental gradient methods for least squares problems. SIAM Journal on Optimization, 7(4):913?926, 1997. [8] Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. Accelerated methods for non-convex optimization. arXiv preprint arXiv:1611.00756, 2016. [9] Yair Carmon, Oliver Hinder, John C Duchi, and Aaron Sidford. " convex until proven guilty": Dimension-free acceleration of gradient descent on non-convex functions. arXiv preprint arXiv:1705.02766, 2017. [10] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12(Jul):2121?2159, 2011. [11] Alexei A Gaivoronski. Convergence properties of backpropagation for neural nets via theory of stochastic gradient methods. part 1. Optimization methods and Software, 4(2):117?134, 1994. [12] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. SIAM Journal on Optimization, 23(4):2341?2368, 2013. [13] Saeed Ghadimi and Guanghui Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. Mathematical Programming, 156(1-2):59?99, 2016. [14] Reza Harikandeh, Mohamed Osama Ahmed, Alim Virani, Mark Schmidt, Jakub Kone?cn‘y, and Scott Sallinen. Stop wasting my gradients: Practical SVRG. In Advances in Neural Information Processing Systems, pages 2242?2250, 2015. [15] Matthew D Hoffman, David M Blei, Chong Wang, and John William Paisley. Stochastic variational inference. Journal of Machine Learning Research, 14(1):1303?1347, 2013. [16] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In Advances in Neural Information Processing Systems, pages 315?323, 2013. [17] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximalgradient methods under the polyak-?ojasiewicz condition. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 795?811. Springer, 2016. [18] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014. [19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521(7553):436?444, 2015. [20] Yann LeCun, L?on Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-

based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278?2324, 1998. [21] Lihua Lei and Michael I Jordan. Less than a single pass: Stochastically controlled stochastic gradient method. arXiv preprint arXiv:1609.03261, 2016. [22] Peter McCullagh and John A Nelder. Generalized Linear Models. CRC Press, 1989. 10

[23] 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. [24] Yurii Nesterov. Introductory lectures on convex optimization: A basic course. Kluwer Academic Publishers, Massachusetts, 2004. [25] Boris Teodorovich Polyak. Gradient methods for minimizing functionals. Zhurnal Vychislitel?noi Matematiki i Matematicheskoi Fiziki, 3(4):643?653, 1963. [26] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. arXiv preprint arXiv:1603.06160, 2016. [27] Sashank J Reddi, Suvrit Sra, Barnab?s P?czos, and Alex Smola. Fast incremental method for nonconvex optimization. arXiv preprint arXiv:1603.06159, 2016. [28] Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton. On the importance of initialization and momentum in deep learning. ICML (3), 28:1139?1147, 2013. [29] Paul Tseng. An incremental gradient (-projection) method with momentum term and adaptive stepsize rule. SIAM Journal on Optimization, 8(2):506?531, 1998. [30] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and R in Machine Learning, 1(1?2):1?305, 2008. variational inference. Foundations and Trends

11