An Even More Optimal Stochastic Optimization Algorithm: Minibatching and Interpolation Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present and analyze an algorithm for optimizing smooth and convex or strongly convex objectives using minibatch stochastic gradient estimates. The algorithm is optimal with respect to its dependence on both the minibatch size and minimum expected loss simultaneously. This improves over the optimal method of Lan [14], which is insensitive to the minimum expected loss; over the optimistic acceleration of Cotter et al. [8], which has suboptimal dependence on the minibatch size; and over the algorithm of Liu and Belkin [15], which is limited to least squares problems and is also similarly suboptimal. Applied to interpolation learning, the improvement over Cotter et al. and Liu and Belkin translates to a linear, rather than square-root, parallelization speedup.

1 Introduction

The massive scale of many modern machine learning models and datasets give rise to complex, high-dimensional training objectives that can be very computationally expensive to optimize. To reduce the computational cost, it is therefore important to devise optimization algorithms that can leverage parallelism to reduce the amount of time needed to train. Stochastic first-order methods, which use stochastic estimates of the gradient of the training objective, are by far the most common approach; and these methods can be directly improved by using minibatch stochastic gradient estimates, which are easy to parallelize across multiple computing cores or devices. Accordingly, we propose and analyze an optimal accelerated minibatch stochastic gradient descent algorithm.

Our analysis exploits that while training machine learning models, it is typically possible to drive the loss either all the way to zero, or at least very close to zero, and the performance of our algorithm improves as the minimum value of the loss approaches zero. This property has multiple names in different contexts. In learning theory, it is common to show fast rates under the assumption of "realizability"—meaning the data can be fit perfectly by the model, i.e. the loss can be driven to zero. Even when the problem is not exactly realizable, it is sometimes possible to derive "optimistic rates" that interpolate between fast rates for realizable learning and the slower agnostic rates [25]. In the context of "interpolation learning," there has recently been great interest in understanding training "overparametrized" models—which have many more parameters than there are training examples, generally meaning that many parameters would attain zero training loss—both in terms of optimization [2–4, 7, 12] and generalization [5, 27, 29]. Finally, in the optimization literature, there have been efforts to prove optimistic rates depending on the minimum value of the objective or on the variance of the stochastic gradients a the minimizer [8, 15–17, 19, 23]. Regardless of the name, these ideas are all based on the same fundamental concept.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Our contributions

In Section 3, we present and analyze an accelerated minibatch SGD algorithm for optimizing smooth and convex objectives using minibatch stochastic gradient estimates. Our method closely resembles the methods of Lan [14] and Cotter et al. [8], but with different hyperparameters and a tighter analysis. Importantly, our algorithm enjoys a linear speedup in the minibatch size all the way up to a critical threshold beyond which larger minibatches do not help. In contrast, Lan and Cotter et al.'s bounds have a worse, sublinear speedup in certain regimes, with a correspondingly higher critical threshold.

In Section 4, we show that a modified version of our algorithm can attain substantially faster convergence when the objective satisfies a certain quadratic growth condition, which is a relaxation of strong convexity. As part of our analysis, we simplify and generalize a technique from Ghadimi and Lan [11], which amounts to a reduction *from* strongly convex optimization *to* convex optimization. The reduction in the other direction is a well-known tool in optimization, but our result shows that it goes both ways and it may be of more general interest.

In Sections 5 and 6, we prove that our methods are optimal with respect to both the minibatch size and also the minimum value of the loss in the settings we consider. We then explain how our guarantees demonstrate a linear speedup in the minibatch size, which improves over a sublinear speedup in previous work [8, 15].

Finally, in Section 7, we extend our results to a related setting where the bound on the minimum value of the loss is replaced by a bound on the variance of the stochastic gradients at the optimum. Under this condition, we establish the optimal error achievable by any learning rule, including non-first-order methods. Our results show that the standard accelerated rate scaling with n^{-2} is not possible given n samples, and the "unaccelerated" n^{-1} rate achieved by SGD is actually optimal. However, we show that our accelerated minibatch SGD method can match the optimal error of SGD using a substantially smaller parallel runtime and the same number of samples.

2 Setting and Background

We consider a generic stochastic optimization problem

$$\min_{w \in \mathbb{R}^d} \{L(w) := \mathbb{E}_{z \sim \mathcal{D}}[\ell(w; z)]\} \tag{1}$$

This problem captures, for instance, supervised machine learning where $z = (x, y)$ is pair of features with a label; $\ell(w; (x, y))$ is the loss of a model parametrized by w on that example; and $L(w)$ is the expected loss. We note that there are two possible interpretations of L depending on what $\mathcal{D}$ corresponds to. We can take $\mathcal{D}$ to be the uniform distribution over a set of training examples, in which case L is the training loss. Alternatively, we can take $\mathcal{D}$ to be the population distribution, in which case L is the population risk. An advantage of the latter view is that optimization guarantees directly imply good performance on the population, however, computing an independent, unbiased stochastic gradient estimate requires a fresh sample from the distribution, so only "one-pass" methods are possible. Nevertheless, our algorithm and analyses apply equally well in either viewpoint.

We consider optimizing objectives on $\mathbb{R}^d$, but we are most interested in dimension-free analysis that does not explicitly depend on the dimension, and our algorithm's guarantees hold even in infinite dimension. Indeed, in many applications including machine learning, the dimension can be very large, so an explicit reliance on the dimension would often be useless.

We study the family of optimization algorithms that attempt to minimize (1) using T sequential stochastic gradient estimates with minibatches of size b of the form

$$g(w_t) = \frac{1}{b} \sum_{i=1}^{b} \nabla \ell(w_t; z^i) \quad \text{for i.i.d. } z^1, \ldots, z^b \sim \mathcal{D} \tag{2}$$

at points $w_1, \ldots, w_T$ of the algorithm's choice. Later, we will argue that our proposed method is optimal with respect to all of the algorithms in this family. Because it is easy to parallelize the computation of the b stochastic gradients in the minibatch, T roughly captures the runtime of the algorithm, while $n = bT$, the total number of samples used, captures its sample complexity.

In our analysis, we will rely on several assumptions about the losses ℓ and the expected loss L.

Assumption 1. *For almost every $z \sim \mathcal{D}$, $\ell(w; z)$ is non-negative, convex, and H-smooth w.r.t. w, i.e.*

$$\forall_{w,u,z}\ \ell(u; z) + \langle \nabla \ell(u; z),\ w - u \rangle \leq \ell(w; z) \leq \ell(u; z) + \langle \nabla \ell(u; z),\ w - u \rangle + \frac{H}{2}\|w - u\|^2$$

This assumption holds, for instance, for training linear models with a smooth, convex, and non-negative loss function, as in least squares problems or logistic regression.

Assumption 2. *The expected loss L is convex and has minimum value $L^* = \min_w L(w)$, which is attained at a point w^* with $\|w^*\| \leq B$.*

The minimum value of the loss, L^*, is a key quantity in our analysis, and we will prove an "optimistic rate" for our algorithm, meaning it performs increasingly well as $L^* \to 0$. The key idea is to use L^* to bound the variance of the stochastic gradient estimates at the point w^*. Following prior work [8, 24, 25], we observe that under Assumptions 1 and 2, we can upper bound (see Lemma 3 in Appendix A)

$$\mathbb{E}\|\nabla \ell(w^*; z) - \nabla L(w^*)\|^2 \leq 2HL^* \tag{3}$$

The H-Lipschitzness of $\nabla \ell$ then allows us to upper bound the variance of the stochastic gradients at other, non-minimizing points w too.

This approach is in contrast to a common practice in the optimization literature of simply assuming that the variance of the stochastic gradient estimates is uniformly upper bounded, i.e.

$$\sup_w \mathbb{E}\|\ell(w; z) - \nabla L(w)\|^2 \leq \sigma^2 \tag{4}$$

This bound is very convenient for analysis, but it can fail to hold even in very simple cases like least squares regression, where the variance grows with $\|w\|^2$. Therefore, while faster rates can sometimes be ensured under the condition (4), this assumption may be too strong and may not correspond well with problems of interest.

For many optimization problems, particularly those arising from training machine learning models, the minimum of the loss, L^*, can be expected to be small. For example, machine learning models are often trained in the interpolation regime [2-5, 7, 12, 27, 29], where the training loss is under-determined, so there are many settings of the parameters which achieve exactly zero training loss. When $\mathcal{D}$ is the empirical distribution (see above), the interpolation regime therefore corresponds to $L^* = 0$. However, even if $\mathcal{D}$ is the population distribution, machine learning problems are often realizable, or nearly realizable, meaning the population risk can also be driven to zero or close to it.

Assumption 3. *The expected loss L is convex; it has minimum value $L^* = \min_w L(w)$; $L(0) - L^* \leq \Delta$; and L satisfies the following growth condition for all w:*

$$L(w) - L^* \geq \frac{\lambda}{2} \min_{w^* \in \arg\min_w L(w)} \|w - w^*\|^2$$

In the optimization literature, it is often possible to show enhanced guarantees under the favorable condition of strong convexity. However, the assumption of strong convexity is somewhat at odds with our goal of studying the interpolation or near-interpolation setting. For example, the simplest example of interpolation learning is underdetermined least squares, where the number of observations is less than the dimension; in this case, the empirical covariance is degenerate so the problem is not strongly convex. Furthermore, assuming *both* strong convexity *and* $L^* \approx 0$ can be very strong, and implies very strong constraints on the objective function which may often fail to hold.

We consider instead a relaxation of strong convexity, Assumption 3, which only requires that the objective grow faster than the squared distance to the set of minimizers (which need not be a singleton) [6, 10, 18]. This condition *does* hold for underdetermined least squares—with λ equal to the smallest *non-zero* eigenvalue of the covariance—and for many other problems that have non-unique minimizers. We also show that this weaker assumption is anyways sufficient to achieve faster rates—even linear convergence in certain regimes.

Related Work

The foundation of much of the work on optimal stochastic optimization algorithms is the accelerated SGD variant, AC-SA, of Lan [14]. This algorithm is analyzed in a different setting, where L is

Algorithm 1 Accelerated Minibatch SGD

$w_0^{\text{ag}} = w_0 = 0$
for $t = 0, 1, \ldots, T-1$ **do**
  $\beta_t = 1 + \frac{t}{6}$ and $\gamma_t = \gamma(t+1)$ for $\gamma = \min\left\{\frac{1}{12H}, \frac{b}{24H(T+1)}, \sqrt{\frac{bB^2}{HL^*T^3}}\right\}$
  $w_t^{\text{md}} = \beta_t^{-1} w_t + (1 - \beta_t^{-1}) w_t^{\text{ag}}$
  $\tilde{w}_{t+1} = w_t - \gamma_t g_t(w_t^{\text{md}})$ where $g_t(w_t^{\text{md}}) = \frac{1}{b}\sum_{i=1}^{b} \nabla \ell(w_t^{\text{md}}; z_t^i)$ for i.i.d. $z_t^1, \ldots, z_t^b \sim \mathcal{D}$
  $w_{t+1} = \min\left\{1, \frac{B}{\|\tilde{w}_{t+1}\|}\right\} \tilde{w}_{t+1}$
  $w_{t+1}^{\text{ag}} = \beta_t^{-1} w_{t+1} + (1 - \beta_t^{-1}) w_t^{\text{ag}}$
Return: w_T^{ag}

H-smooth, B-bounded, and convex and where the stochastic gradients have uniformly bounded variance (4), and under these conditions, it is optimal, with guarantee

$$\mathbb{E}L(\hat{w}) - L^* \le c \cdot \left(\frac{HB^2}{T^2} + \frac{\sigma B}{\sqrt{bT}}\right) \tag{5}$$

In our setting, however, there is no such explicit bound on the gradient variance, and at the same time, Lan's analysis does not exploit the bound on L^* to achieve a better rate. Under our Assumptions 1 and 2, it is possible to use the variance upper bound $\sigma^2 = 2H^2B^2 + 4HL^*$, which yields

$$\mathbb{E}L(\hat{w}) - L^* \le c \cdot \left(\frac{HB^2}{T^2} + \frac{HB^2}{\sqrt{bT}} + \sqrt{\frac{HB^2L^*}{bT}}\right) \tag{6}$$

However, we will later show that this guarantee is suboptimal, and the denominator of the second term can be improved to bT.

In the most directly comparable existing work, Cotter et al. [8] propose and analyze a minibatch SGD algorithm and an accelerated variant similar to AC-SA under our Assumptions 1 and 2. Their accelerated method is guaranteed to achieve suboptimality

$$\mathbb{E}L(\hat{w}) - L^* \le c \cdot \left(\frac{HB^2}{T^2} + \frac{HB^2}{\sqrt{b}T} + \frac{HB^2\sqrt{\log T}}{bT} + \sqrt{\frac{HB^2L^*}{bT}}\right) \tag{7}$$

While this is optimal (up to a minor $\sqrt{\log T}$ factor) in the special case $b = 1$, we show that the $\sqrt{b}T$ dependence in the second term can be improved to bT. We also extend our analysis to the Assumptions 1 and 3.

In other related work, Liu and Belkin [15] propose a different minibatch accelerated SGD method. In the special case of least squares objectives—where $\ell(w; (x, y)) = \frac{1}{2}(\langle w, x\rangle - y)^2$—that also satisfy Assumptions 1 and 3 with $L^* = 0$, they show that their algorithm attains suboptimality

$$\mathbb{E}L(\hat{w}) - L^* \le c \cdot \Delta \exp\left(-c' \cdot \frac{T}{\sqrt{H/\lambda} + H/(\lambda\sqrt{b})}\right) \tag{8}$$

As with Cotter et al., we show that the $\sqrt{b}$ dependence of this guarantee is suboptimal and can be improved to b. In addition, our analysis goes beyond the special case of least squares with $L^* = 0$.

In other related work, Zhang et al. [31] study algorithms in a related setting where the losses ℓ are Lipschitz rather than smooth, and where the dimension of the problem is sufficiently small relative to the other problem parameters. Zhang and Zhou [30] study our smooth setting but with an additional restriction that ℓ is Lipschitz. Srebro et al. [25] study the performance of the empirical risk minimizer under Assumptions 1 and 2 as well as stochastic first-order methods with $b = 1$. Their methods are optimal for $b = 1$, but they do not analyze the effect of the minibatch size.

3 A Better Accelerated Minibatch SGD Method

The method we propose, Algorithm 1, is essentially identical to the AC-SA algorithm of Lan [14] and to the AG algorithm of Cotter et al. [8]; the difference is that we use different stepsize and momentum

parameters and provide a tighter analysis for our setting. Our Accelerated Minibatch SGD algorithm provides the following guarantee:

Theorem 1. *Let ℓ and L satisfy Assumptions 1 and 2, then Algorithm 1 guarantees*

$$\mathbb{E}L(w_T^{ag}) - L^* \leq c \cdot \left(\frac{HB^2}{T^2} + \frac{HB^2}{bT} + \sqrt{\frac{HB^2 L^*}{bT}} \right)$$

We prove this in Appendix A using a similar approach as the analysis for AC-SA of Lan [14]. As discussed previously, Lan's analysis relies on a uniform upper bound on the stochastic gradient variance on the set $\{w : \|w\| \leq B\}$. While an upper bound can be derived in our setting, it resembles H^2B^2 which is too large to achieve good performance. Our analysis, in contrast, exploits the fact that the points $w_0^{\text{md}}, \ldots, w_{T-1}^{\text{md}}$ at which the stochastic gradients are actually computed approach w^* as the algorithm proceeds, which implies that the stochastic gradient variance decreases over time. So, an analysis based on a uniform upper bound on the stochastic gradient variance is too pessimistic and our analysis improves by more tightly controlling the variance.

It is difficult to identify precisely why our analysis improves in its dependence on the minibatch size, b, compared with Cotter et al.'s bound. However, the primary difference between our analyses is that Cotter et al. use stepsizes $\gamma_t \propto t^p$ for $p \leq 1$, while our stepsizes scales linearly with t. Our choice leads to a somewhat simpler proof, which may explain our tighter bound.

4 The Reduction and Faster Rates

Our algorithm can also be extended to the setting of Assumptions 1 and 3 using an argument resembling that of Ghadimi and Lan [11]. Their proof is based on a simple idea, which we will show amounts to a reduction from strongly convex optimization to convex optimization. However, their argument is fairly complex and specific to their problem. In this section, we simplify and generalize their approach, and we present it in a way that will hopefully be convenient for future use. We then apply this idea to provide an enhanced guarantee for Algorithm 1 under Assumptions 1 and 3.

Convex optimization algorithms guarantee reducing the value of the objective by an amount that depends on some measure of the distance between the initialization and a minimizer of the objective. The key idea in Ghadimi and Lan's analysis is that when the objective is λ-strongly convex, reducing the value of the objective implies reducing the *distance* to the minimizer:

$$\|w - w^*\|^2 \leq \frac{2(L(w) - L^*)}{\lambda} \tag{9}$$

Therefore, if we apply an algorithm for convex objectives to a strongly convex objective several times in succession, each time reducing the suboptimality by a constant factor, then (1) roughly $\log 1/\epsilon$ applications will suffice to reach ϵ-suboptimality and (2) reducing the suboptimality by a constant factor will get no harder with each application since the distance to the optimum is decreasing. Using this idea allows us to take an algorithm with a guarantee for convex objectives and slightly modify it in order to derive an algorithm with a corresponding, better guarantee for strongly convex objectives. We also show that strong convexity can be replaced by a weaker condition that generalizes Assumption 3.

In order to present the reduction, we first define

Definition 1. *Given $\psi : \mathbb{R}^d \to \mathbb{R}_+$, we say that L satisfies the (λ, ψ)-growth condition (hereafter (λ, ψ)-GC) if for all w*

$$L(w) - L^* \geq \lambda \psi(w)$$

As an example, the (λ, ψ)-GC for $\psi(w) = \frac{1}{2}\|w - w^*\|^2$ is equivalent to λ-strong convexity. However, the growth condition can be more general, for instance, the condition in Assumption 3 is equivalent to the (λ, ψ)-GC for $\psi(w) = \arg\min_{w^* \in \arg\min_w L(w)} \|w - w^*\|^2$.

The second ingredient of the reduction is the "time" needed by an algorithm to optimize convex or strongly convex objectives:

Definition 2. *Let $\mathcal{L}$ be any set of convex functions. We define $\mathsf{Time}(\epsilon, B, \psi, \mathcal{L}, \mathcal{A})$ to be the time needed by the algorithm $\mathcal{A}$ to find a point $\hat{w}$ with $\mathbb{E}L(\hat{w}) - L^* \leq \epsilon$ given a point w_0 with $\mathbb{E}\psi(w_0) \leq B$,*

Algorithm 2 $\mathsf{GC2Cvx}(\mathcal{A}, \theta)$

Given: w_0 s.t. $\mathbb{E}L(w_0) - L^* \leq \Delta$
for $t = 1, 2, \ldots, T = \lceil \log_\theta \frac{\Delta}{\epsilon} \rceil$ **do**
  Set w_t to be the output of $\mathcal{A}$ initialized a w_{t-1} after $\mathsf{Time}\big(\theta^{-t}\Delta, \theta^{1-t}\frac{\Delta}{\lambda}, \psi, \mathcal{L}, \mathcal{A}\big)$
Return w_T

for any $L \in \mathcal{L}$. Similarly, we define $\mathsf{Time}_\lambda(\epsilon, \Delta, \psi, \mathcal{L}, \mathcal{A})$ to be the time needed by $\mathcal{A}$ to find $\hat{w}$ with $\mathbb{E}L(\hat{w}) - L^* \leq \epsilon$ given a point w_0 with $\mathbb{E}L(w_0) - L^* \leq \Delta$, for any $L \in \mathcal{L}$ that also satisfies the (λ, ψ)-GC.

We are deliberately vague about the precise meaning of "time" here. Typically, it would correspond to the number of iterations of the algorithm, but it could also count the number of times the algorithm accesses a certain oracle, or even the wall-clock time of an implementation of the algorithm.

With these definitions in hand, we present the reduction, Algorithm 2, which guarantees:

Theorem 2. *For any algorithm $\mathcal{A}$ and $\theta > 1$, $\mathsf{GC2Cvx}(\mathcal{A}, \theta)$ defined in Algorithm 2 guarantees*

$$\mathsf{Time}_\lambda(\epsilon, \Delta, \psi, \mathcal{L}, \mathsf{GC2Cvx}(\mathcal{A}, \theta)) \leq \sum_{t=1}^{\lceil \log_\theta \frac{\Delta}{\epsilon} \rceil} \mathsf{Time}\left(\theta^{-t}\Delta, \theta^{1-t}\frac{\Delta}{\lambda}, \psi, \mathcal{L}, \mathcal{A}\right)$$

Proof. By the definition of $\mathsf{Time}\big(\theta^{-t}\Delta, \theta^{1-t}\frac{\Delta}{\lambda}, \psi, \mathcal{L}, \mathcal{A}\big)$, if $\mathbb{E}\psi(w_{t-1}) \leq \theta^{1-t}\frac{\Delta}{\lambda}$ at each iteration, then $\mathbb{E}L(w_t) - L^* \leq \theta^{-t}\Delta$ for each t, and $\mathbb{E}L(w_T) - L^* \leq \theta^{-T}\Delta \leq \epsilon$. We now prove by induction that the condition $\mathbb{E}\psi(w_{t-1}) \leq \theta^{1-t}\frac{\Delta}{\lambda}$ always holds.

As the base case, the (λ, ψ)-GC implies that

$$\lambda\psi(w_0) \leq L(w_0) - L^* \implies \mathbb{E}\psi(w_0) \leq \frac{\Delta}{\lambda} \tag{10}$$

Now, suppose that for all $t' < t$, $\mathbb{E}\psi(w_{t'}) \leq \theta^{-t'}\frac{\Delta}{\lambda}$. Then, by the definition of $\mathsf{Time}\big(\theta^{-t}\Delta, \theta^{1-t}\frac{\Delta}{\lambda}, \psi, \mathcal{L}, \mathcal{A}\big)$, we have $\mathbb{E}L(w_t) - L^* \leq \theta^{-t}\Delta$ so by the (λ, ψ)-GC

$$\lambda\psi(w_t) \leq L(w_t) - L^* \implies \mathbb{E}\psi(w_t) \leq \frac{\mathbb{E}L(w_t) - L^*}{\lambda} \leq \theta^{-t}\frac{\Delta}{\lambda} \tag{11}$$

This completes the proof. □

To give a better sense of Theorem 2 and how it can be applied, we discuss several examples in Appendix B. This reduction complements a standard tool in the optimization toolbox, which goes in the opposite direction. Specifically, it is very common to take an algorithm for optimizing λ-strongly convex objectives and to apply it to convex objectives by optimizing the λ-strongly convex surrogate $L_\lambda(w) = L(w) + \frac{\lambda}{2}\|w\|^2$. Sometimes, this approach is slightly suboptimal, and the similar, multistep procedure of Allen-Zhu and Hazan [1] is required, nevertheless, it was already known that strongly convex optimization is, for this reason, "easier" than convex optimization. Our result shows that the reverse is also true, so strongly convex and convex optimization are, in a certain sense, equally hard.

In combination, these two reductions going in each direction appear to be optimal in a certain way. In all of the examples we have tried, composing both reductions—converting a convex algorithm to a strongly convex one, and then converting that strongly convex algorithm back to a convex one—results in only a constant-factor degradation in the guarantee. Furthermore, we have consistently observed that applying the reduction, Algorithm 2, to an optimal algorithm for convex optimization yields an optimal algorithm for strongly-convex optimization, and we conjecture that this holds universally.

Before moving on, we note that the reduction Algorithm 2 can result in an unusual method. For example, the algorithm $\mathsf{GC2Cvx}(\text{Alg 1}, e)$ analyzed below involves repeated invocations of Algorithm 1, resulting in the stepsize and momentum parameters being reset periodically. As a whole, it resembles Algorithm 1, however it has a strange, non-monotonic stepsize and momentum schedule. Therefore, while this technique can be useful for deriving new optimization algorithms with better

theoretical guarantees, these methods may not always be aethetically pleasing or practical, and it may still be useful to study more "natural" methods that can be directly analyzed under the (λ, ψ)-GC.

Finally, we apply this technique to derive a variant of Algorithm 1 for use under Assumptions 1 and 3, which we prove in Appendix C.

Theorem 3. *Let ℓ and L satisfy Assumptions 1 and 3, then the output of* GC2Cvx(*Alg 1*, e) *guarantees*

$$\mathbb{E}L(\hat{w}) - L^* \leq c \cdot \left(\Delta \exp\left(-\frac{c'\sqrt{\lambda}T}{\sqrt{H}} \right) + \Delta \exp\left(-\frac{c'\lambda bT}{H} \right) + \frac{HL^*}{\lambda bT} \right)$$

5 The Optimality of Our Algorithms

Here, we argue that the guarantees in Theorems 1 and 3 are optimal in the setting we consider. First, it is well-known that even without any noise, i.e. $\ell(w; z) = L(w)$, and even with $L^* = 0$, any first-order method will have error at least [20]

$$L(\hat{w}) - L^* \geq c \cdot \frac{HB^2}{T^2} \qquad \text{or} \qquad L(\hat{w}) - L^* \geq c \cdot \Delta \exp\left(-c' \cdot \frac{\sqrt{\lambda}T}{\sqrt{H}} \right) \tag{12}$$

under Assumptions 1 and 2 or Assumptions 1 and 3, respectively (in fact, the latter holds even under the stronger condition of λ-strong convexity). Therefore, the first terms in our method's guarantees in Theorems 1 and 3 are tight, and correspond precisely to the rate achieved by accelerated gradient descent [22].

For the remaining second and third terms in each guarantee, we prove the following lower bound:

Theorem 4. *There exists a loss $\ell(w; (x, y))$ such that for any learning algorithm, there exists a distribution over (x, y) pairs such that ℓ and L satisfy Assumptions 1 and 2, and such that this algorithm's output given n i.i.d. samples from the distribution will have error at least*

$$\mathbb{E}L(\hat{w}) - L^* \geq c \cdot \left(\frac{HB^2}{n} + \sqrt{\frac{HB^2L^*}{n}} \right)$$

Similarly, there exists a loss $\ell(w; (x, y))$ such that for any learning algorithm, there exists a distribution over (x, y) pairs such that ℓ and L satisfy Assumptions 1 and 3, and such that this algorithm's output given n i.i.d. samples from the distribution will have error at least

$$\mathbb{E}L(\hat{w}) - L^* \geq c \cdot \left(\Delta \cdot \mathbb{1}_{n \leq \frac{H}{2\lambda}} + \min\left\{ \frac{HL^*}{\lambda n}, \Delta \right\} \right)$$

We prove this in Appendix D using an argument similar to that of Srebro et al. [25], based on simple least squares problems. The standard lower bound for deterministic first-order optimization plus Theorem 4 establish that our method is an optimal first-order algorithm under Assumptions 1 and 2 and they essentially establish that our method is optimal under Assumptions 1 and 3. In the latter case, the only gap is between $\Delta \exp\left(-c' \cdot \frac{\lambda bT}{H}\right)$ and $\Delta \cdot \mathbb{1}_{bT \leq \frac{H}{2\lambda}}$, which is exponentially small, and nearly negligible.

However, even stronger than proving that our algorithm is optimal amongst first-order methods, Theorem 4 shows that the second and third terms of our guarantee are actually optimal amongst *all* algorithms that use only $n = bT$ samples from the distribution, not just first-order methods. Therefore, for b small enough that the first term is lower-order, our algorithm is actually an optimal learning rule amongst all possible learning rules.

6 The Minibatch Parallelization Speedup

Here, it will be convenient to rewrite our guarantees in Theorems 1 and 3 in terms of how large T must be in order to guarantee error less than ϵ. Under Assumptions 1 and 2, and Assumptions 1 and

3, respectively, this is

$$T(\epsilon) = c \cdot \left(\sqrt{\frac{HB^2}{\epsilon}} + \frac{1}{b}\left(\frac{HB^2}{\epsilon} + \frac{HB^2 L^*}{\epsilon^2} \right) \right) \tag{13}$$

$$T(\epsilon) = c \cdot \left(\sqrt{\frac{H}{\lambda}} \log \frac{\Delta}{\epsilon} + \frac{1}{b}\left(\frac{H}{\lambda} \log \frac{\Delta}{\epsilon} + \frac{HL^*}{\lambda\epsilon} \right) \right) \tag{14}$$

Written this way, it is easy to see that the number of iterations needed by our algorithm to reach accuracy ϵ decreases linearly with the minibatch size b up until the first term becomes larger than the second and third terms. This linear speedup [8, 9] continues until

$$b \gtrsim \sqrt{\frac{HB^2}{\epsilon}} + \frac{L^* \sqrt{HB^2}}{\epsilon^{3/2}} \quad \text{and} \quad b \gtrsim \sqrt{\frac{H}{\lambda}} + \frac{L^* \sqrt{HB^2}}{\epsilon} \tag{15}$$

in each case respectively (ignoring log factors). Once b passes these thresholds, no more improvement is possible by increasing the minibatch size, and the iteration complexity is dominated by the first term, which corresponds to the time needed to reach ϵ error using exact accelerated gradient descent [22]. In other words, once the minibatch size is this large, the algorithm performs essentially the same as if there were no noise in the gradients.

While it may not be practical to fully parallelize the computation of the minibatches across b machines for large b, our results nevertheless represent a substantial speedup and an opportunity to parallelize computation with absolutely no cost to the algorithm's theoretical performance. This also improves significantly over previous work. For instance, the algorithm of Cotter et al. [8] requires

$$T(\epsilon) = c \cdot \left(\sqrt{\frac{HB^2}{\epsilon}} + \frac{1}{\sqrt{b}} \frac{HB^2}{\epsilon} + \frac{1}{b} \frac{HB^2 L^*}{\epsilon^2} \right) \tag{16}$$

Therefore, their analysis exhibits three regimes rather than our two: first a linear speedup for small b, then a $1/\sqrt{b}$ speedup for intermediate b, and finally no speedup once $b \gtrsim \frac{HB^2}{\epsilon} + \frac{\sqrt{HB^2}L^*}{\epsilon^{3/2}}$. Therefore, even under the favorable condition that $L^* = 0$, Cotter et al.'s method has a sublinear parallelization speedup from minibatching, and the minibatch size needed to match the performance of accelerated gradient descent can be as much as $\sqrt{HB^2/\epsilon}$ times larger. Under Assumption 3 the algorithm and analysis of Liu and Belkin [15] exhibits as similar sublinear speedup from minibatching.

7 Stochastic Optimization with Bounded Variance at the Optimum

So far, we have considered optimizing objectives where the instantaneous losses are non-negative and the value of the minimum of the expected loss is bounded and small, but in other contexts we may want to understand the complexity of optimization in terms of bounds on the variance of the stochastic gradients. In the optimization literature, it is common to assume that the variance of the stochastic gradients is bounded uniformly on the entire space, i.e. $\sup_w \mathbb{E}\|\nabla \ell(w; z)\|^2 \leq \sigma^2$. When, in addition to this variance bound, the objective L is H-smooth and convex, and has a minimizer with norm at most B, then it has long been known that T steps of SGD achieves error [20]

$$\mathbb{E}L(w_T) - L^* \leq \frac{HB^2}{T} + \frac{\sigma B}{\sqrt{T}} \tag{17}$$

However, the assumption of uniformly upper bounded variance can be strong, and it turns out that when ℓ is also H-smooth, the σ in SGD's guarantee can easily be replaced with σ_*, an upper bound on the standard deviation of the variance at the minimizer specifically, i.e. $\mathbb{E}\|\nabla \ell(w^*; z)\|^2 \leq \sigma_*^2$ [26]. Indeed, for many other *non-accelerated* algorithms, the weaker bound σ_* suffices and a global bound is unnecessary [13, 28].

However, it was not clear whether it is possible to make this substitution of σ for σ_* for *accelerated* methods too. In fact, for a long time it was not even clear that the T^{-2} accelerated rate was possible in the stochastic setting at all until Lan [14]'s optimal stochastic first-order algorithm, which guarantees

$$\mathbb{E}L(w_T) - L^* \leq \frac{HB^2}{T^2} + \frac{\sigma B}{\sqrt{T}} \tag{18}$$

Can we go farther and replace this σ with σ_* too? Our lower bound, Theorem 4, shows that the answer is no:

Corollary 1. *For any algorithm, there exists an objective $L(w) = \mathbb{E}_z \ell(w; z)$ such that L has a minimizer with norm less than B, ℓ is H-smooth and convex, and $\mathbb{E}\|\ell(w^*; z)\|^2 \le \sigma_*^2$, and such that the output of the algorithm given n samples is at least*

$$\mathbb{E}L(\hat{w}) - L^* \ge c \cdot \left(\frac{HB^2}{n} + \frac{\sigma_* B}{\sqrt{n}} \right)$$

A similar result also holds in the strongly convex setting. This corollary, in essence, shows that it is impossible to achieve an accelerated n^{-2} rate under these conditions, so Lan's accelerated SGD algorithm relied crucially on the uniformly bounded variance. In fact, the corollary applies to any algorithm, not just first-order methods. However, there is good news, which is that our guarantees for *minibatch* accelerated SGD can be applied also in this setting:

Theorem 5. *Let $L(w) = \mathbb{E}_z \ell(w; z)$ have a minimizer with norm at most B, let ℓ be H-smooth and convex, and let $\mathbb{E}\|\nabla \ell(w^*; z)\|^2 \le \sigma_*^2$. Then Algorithm 1 guarantees*

$$\mathbb{E}L(w_T^{ag}) - L^* \le c \cdot \left(\frac{HB^2}{T^2} + \frac{HB^2}{bT} + \frac{\sigma_* B}{\sqrt{bT}} \right)$$

Let $L(w) = \mathbb{E}_z \ell(w; z)$ satisfy $L(w) - L^* \ge \frac{\lambda}{2} \min_{w^* \in \arg\min_w L(w)} \|w - w^*\|^2$ for all w, let $L(0) - L^* \le \Delta$, let ℓ be H-smooth and convex, and let $\mathbb{E}\|\nabla \ell(w^*; z)\|^2 \le \sigma_*^2$. Then $\mathsf{GC2Cvx}(Alg 1, e)$ guarantees

$$\mathbb{E}L(\hat{w}) - L^* \le c \cdot \left(\Delta \exp\left(-\frac{c' \sqrt{\lambda} T}{\sqrt{H}} \right) + \Delta \exp\left(-\frac{c' \lambda b T}{H} \right) + \frac{\sigma_*^2}{\lambda b T} \right)$$

The first part is included in the proof of Theorem 1 in Appendix A, and the second part is basically identical to the proof of Theorem 3.

The lower bound above showed that it is impossible to achieve error like n^{-2} using n samples. However, Theorem 5 shows it *is* possible to achieve error like T^{-2} with parallel runtime T using our minibatch accelerated SGD method. In other words, while n steps of SGD matches the lower bound in Corollary 1 and therefore attains the smallest possible error using n samples, our method is able to more quickly attain this same optimal error using $n = bT$ samples with $b \gg 1$, so the parallel runtime T can be as small as $\sqrt{n}$, i.e. a quadratic improvement in the runtime vs. SGD. Since our upper and lower bounds in this section match, this also tightly bounds the complexity of stochastic first-order optimization with a bound on σ_*.

References

[1] Zeyuan Allen-Zhu and Elad Hazan. Optimal black-box reductions between optimization objectives. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/1f50893f80d6830d62765ffad7721742-Paper.pdf

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *arXiv preprint arXiv:1811.04918*, 2018.

[3] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

[4] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

[5] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116 (32):15849–15854, 2019.

[6] Jérôme Bolte, Trong Phong Nguyen, Juan Peypouquet, and Bruce W Suter. From error bounds to the complexity of first-order descent methods for convex functions. *Mathematical Programming*, 165(2): 471–507, 2017.

[7] Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32:2937–2947, 2019.

[8] Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. Better mini-batch algorithms via accelerated gradient methods. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1647–1655. Curran Associates, Inc., 2011. URL http://papers.nips.cc/paper/4432-better-mini-batch-algorithms-via-accelerated-gradient-methods.pdf.

[9] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(Jan):165–202, 2012.

[10] Dmitriy Drusvyatskiy and Adrian S Lewis. Error bounds, quadratic growth, and linear convergence of proximal methods. *Mathematics of Operations Research*, 43(3):919–948, 2018.

[11] Saeed Ghadimi and Guanghui Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization, ii: shrinking procedures and optimal algorithms. *SIAM Journal on Optimization*, 23(4):2061–2089, 2013.

[12] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8580–8589, 2018.

[13] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized sgd with changing topology and local updates. In *International Conference on Machine Learning*, pages 5381–5393. PMLR, 2020.

[14] Guanghui Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1-2):365–397, 2012. URL https://pdfs.semanticscholar.org/1621/f05894ad5fd6a8fcb8827a8c7aca36c81775.pdf.

[15] Chaoyue Liu and Mikhail Belkin. Mass: an accelerated stochastic method for over-parametrized learning. *arXiv preprint arXiv:1810.13395*, 2018.

[16] Siyuan Ma, Raef Bassily, and Mikhail Belkin. The power of interpolation: Understanding the effectiveness of sgd in modern over-parametrized learning. In *International Conference on Machine Learning*, pages 3325–3334. PMLR, 2018.

[17] Eric Moulines and Francis Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. *Advances in neural information processing systems*, 24:451–459, 2011.

[18] Ion Necoara, Yu Nesterov, and Francois Glineur. Linear convergence of first order methods for non-strongly convex optimization. *Mathematical Programming*, 175(1):69–107, 2019.

[19] Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. *Advances in neural information processing systems*, 27:1017–1025, 2014.

[20] Arkadii Semenovich Nemirovsky and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

[21] Yurii Nesterov. Introductory lectures on convex optimization: a basic course. 2004.

[22] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

[23] Mark Schmidt and Nicolas Le Roux. Fast convergence of stochastic gradient descent under a strong growth condition. *arXiv preprint arXiv:1308.6370*, 2013.

[24] Shai Shalev-Shwartz and Yoram Singer. Online learning: Theory, algorithms, and applications. 2007.

[25] Nathan Srebro, Karthik Sridharan, and Ambuj Tewari. Optimistic rates for learning with a smooth loss. *arXiv preprint arXiv:1009.3896*, 2010.

[26] Sebastian U Stich. Unified optimal analysis of the (stochastic) gradient method. *arXiv preprint arXiv:1907.04232*, 2019.

[27] Alexander Tsigler and Peter L Bartlett. Benign overfitting in ridge regression. *arXiv preprint arXiv:2009.14286*, 2020.

[28] Blake Woodworth, Kumar Kshitij Patel, and Nathan Srebro. Minibatch vs local sgd for heterogeneous distributed learning. *arXiv preprint arXiv:2006.04735*, 2020.

[29] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[30] Lijun Zhang and Zhi-Hua Zhou. Stochastic approximation of smooth and strongly convex functions: Beyond the $o(1/t)$ convergence rate. In *Conference on Learning Theory*, pages 3160–3179. PMLR, 2019.

[31] Lijun Zhang, Tianbao Yang, and Rong Jin. Empirical risk minimization for stochastic convex optimization: $o(1/n)$-and $o(1/n^2)$-type of risk bounds. In *Conference on Learning Theory*, pages 1954–1979. PMLR, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]