AdaGrad Stepsizes: Sharp Convergence Over Nonconvex Landscapes

Rachel Ward [*1 2] Xiaoxia Wu [*1 2] Léon Bottou [2]

Abstract

Adaptive gradient methods such as AdaGrad and its variants update the stepsize in stochastic gradient descent on the fly according to the gradients received along the way; such methods have gained widespread use in large-scale optimization for their ability to converge robustly, without the need to fine-tune parameters such as the stepsize schedule. Yet, the theoretical guarantees to date for AdaGrad are for online and convex optimization. We bridge this gap by providing strong theoretical guarantees for the convergence of AdaGrad over smooth, nonconvex landscapes. We show that the norm version of AdaGrad (AdaGrad-Norm) converges to a stationary point at the $\mathcal{O}(\log(N)/\sqrt{N})$ rate in the stochastic setting, and at the optimal $\mathcal{O}(1/N)$ rate in the batch (non-stochastic) setting – in this sense, our convergence guarantees are "sharp". In particular, both our theoretical results and extensive numerical experiments imply that AdaGrad-Norm is robust to the *unknown Lipschitz constant and level of stochastic noise on the gradient*.

1. Introduction

Consider the problem of minimizing a differentiable function $F : \mathbb{R}^d \to \mathbb{R}$ via stochastic gradient descent (SGD); starting from $x_0 \in \mathbb{R}^d$ and stepsize $\eta_0 > 0$, SGD iterates until convergence

$$x_{j+1} \leftarrow x_j - \eta_j G(x_j), \tag{1}$$

where $\eta_j > 0$ is the stepsize at the jth step and $G(x_j)$ is the stochastic gradient in the form of a random vector satisfying $\mathbb{E}[G(x_j)] = \nabla F(x_j)$ and having bounded variance. SGD is the de facto standard for deep learning optimization problems, or more generally, for large-scale optimization problems where the loss function $F(x)$ can be approximated by the average of a large number n of component functions, $F(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$. It is more efficient to measure a single component gradient $\nabla f_{i_j}(x)$, $\quad i_j \sim \text{Uniform}\{1, 2, \ldots, N\}$, and move in the noisy direction $G_j(x) = \nabla f_{i_j}(x)$, than to compute a full gradient $\nabla F(x) = \frac{1}{n}\sum_{j=1}^n \nabla f_j(x)$ (Bottou & Bousquet, 2008). In the stochastic setting, the question of how to choose the stepsize $\eta > 0$ or stepsize schedule $\{\eta_j\}$ is difficult. The classical Robbins/Monro theory (Robbins & Monro, 1951) says that in order for $\lim_{k\to\infty} \mathbb{E}[\|\nabla F(x_k)\|^2] = 0$, the stepsize schedule should satisfy

$$\sum_{k=1}^{\infty} \eta_k = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} \eta_k^2 < \infty; \tag{2}$$

Yet, these bounds do not necessarily inform how to set the stepsize in practice, where algorithms are run for a finite iterations and the constants in the rate of convergence matter.

1.1. Adaptive Gradient Methods

Adaptive stochastic gradient methods such as *AdaGrad* (introduced independently by Duchi et al. (2011) and McMahan & Streeter (2010)) have been widely used in the past few years. AdaGrad updates the stepsize $\eta_j = \eta/b_j$ on the fly given information of all previous (noisy) gradients observed along the way. The most common variant of AdaGrad updates an entire vector of per-coefficient stepsizes. To be concrete, for optimizing a function $F : \mathbb{R}^d \to \mathbb{R}$, the "coordinate" version of AdaGrad updates d scalar parameters $[b_j]_\ell, \ell = 1, 2, \ldots, d$ at the j iteration – one for each $[x_j]_\ell$ coordinate of $x_j \in \mathbb{R}^d$ – according to $([b_{j+1}]_\ell)^2 = ([b_j]_\ell)^2 + ([\nabla F(x_j)]_\ell)^2$ in the noiseless setting, and $([b_{j+1}]_\ell)^2 = ([b_j]_\ell)^2 + ([G(x_j)]_\ell)^2$ in the noisy gradient setting. This common use makes AdaGrad a variable metric method and has been the object of recent criticism for machine learning applications (Wilson et al., 2017).

One can also consider a variant of AdaGrad which updates only a single (scalar) stepsize according to the sum of squared gradient norms observed so far. In this work, we focus instead on the "norm" version of AdaGrad as a single stepsize adaptation method using the gradient norm information, which we call AdaGrad-Norm. The update in the stochastic setting is as follows: initialize a single scalar $b_0 > 0$; at the jth iteration, observe the random variable G_j

[*]Equal contribution [1]Department of Mathematics, The University of Texas at Austin, USA [2]Facebook AI Research, New York, USA. Correspondence to: Xiaoxia Wu <xwu@math.utexas.edu>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

such that $\mathbb{E}[G_j] = \nabla F(x_j)$ and iterate

$$x_{j+1} \leftarrow x_j - \eta \frac{G(x_j)}{b_{j+1}} \quad \text{with} \quad b_{j+1}^2 = b_j^2 + \|G(x_j)\|^2$$

where $\eta > 0$ is to ensure homogeneity and that the units match. It is straightforward that in expectation, $\mathbb{E}[b_k^2] = b_0^2 + \sum_{j=0}^{k-1} \mathbb{E}[\|G(x_j)\|^2]$; thus, under the assumption of uniformly bounded gradient $\|\nabla F(x)\|^2 \leq \gamma^2$ and uniformly bounded variance $\mathbb{E}_\xi \left[\|G(x;\xi) - \nabla F(x)\|^2\right] \leq \sigma^2$, the stepsize will decay eventually according to $\frac{1}{b_j} \geq \frac{1}{\sqrt{2(\gamma^2+\sigma^2)j}}$. This stepsize schedule matches the schedule which leads to the rates of convergence for SGD in the case of convex but not necessarily smooth functions, as well as smooth but not necessarily convex functions (see, for instance, Agarwal et al. (2009) and Bubeck et al. (2015)). This observation suggests that AdaGrad-Norm should be able to achieve convergence rates for SGD, but *without having to know Lipschitz smoothness parameter of F and the parameter σ a priori* to set the stepsize schedule.

Theoretically rigorous convergence results for AdaGrad-Norm were provided in the convex setting recently (Levy, 2017). Moreover, it is possible to obtain convergence rates in the offline setting by online-batch conversion. However, making such observations rigorous for nonconvex functions is difficult because b_j is itself a random variable which is correlated with the current and all previous noisy gradients; thus, the standard proofs in SGD do not straightforwardly extend to the proofs of AdaGrad-Norm. This paper provides such proof for AdaGrad-Norm.

Main Contributions Our results make rigorous and precise the observed phenomenon that the convergence behavior of AdaGrad-Norm is *highly adaptable to the unknown Lipschitz smoothness constant and level of stochastic noise on the gradient*: when there is noise, AdaGrad-Norm converges at the rate of $O(\log(N)/\sqrt{N})$, and when there is no noise, the same algorithm converges at the optimal $O(1/N)$ rate like well-tuned batch gradient descent. Moreover, our analysis shows that AdaGrad-Norm converges at these rates for any choices of the algorithm hyperparameters $b_0 > 0$ and $\eta > 0$, in contrast to GD or SGD with fixed stepsize where if the stepsize is set above a hard upper threshold governed by the (generally unknown) smoothness constant L, the algorithm might not converge at all. Finally, we note that the constants in the rates of convergence we provide are explicit in terms of their dependence on the hyperparameters b_0 and η. We list our two main theorems (informally) in the following.

For a differential non-convex function F with L-Lipschitz gradient and $F^* = \inf_x F(x) > -\infty$, Theorem 2.1 implies that AdaGrad-Norm converges to an ε-approximate stationary point with high probability at the rate

$$\begin{aligned}&\min_{\ell \in [N-1]} \|\nabla F(x_\ell)\|^2 \\ \leq &\mathcal{O}\left(\frac{\gamma(\sigma + \eta L + (F(x_0) - F^*)/\eta)\log(N\gamma^2/b_0^2)}{\sqrt{N}}\right).\end{aligned}$$

When there is no noise in the gradient, i.e., $\sigma = 0$, we can improve this rate to the optimal $\mathcal{O}\left(1/N\right)$ rate of convergence (see Theorem 2.2) without the additional log factor.

If the optimal value of the loss function F^* is known and one sets $\eta = F(x_0) - F^*$ accordingly, then the constant in our rate is close to the best-known constant $\sigma L(F(x_0) - F^*)$ achievable for SGD with fixed stepsize $\eta = \eta_1 = \cdots = \eta_N = \min\{\frac{1}{L}, \frac{1}{\sigma\sqrt{N}}\}$ carefully tuned to knowledge of L and σ, as given in Ghadimi & Lan (2013).

Extensive experiments in Section 4 shows that the robustness of AdaGrad-Norm extends from simple linear regression to state-of-the-art models in deep learning, without sacrificing generalization.

1.2. Previous Work

Theoretical guarantees of convergence for AdaGrad were provided in Duchi et al. (2011) in the setting of online convex optimization, where the loss function may change from iteration to iteration and be chosen adversarially. AdaGrad was subsequently observed to be effective for accelerating convergence in the nonconvex setting, and has become a popular algorithm for optimization in deep learning problems. Many modifications of AdaGrad with or without momentum have been proposed, namely, RMSprop (Srivastava & Swersky, 2012), AdaDelta (Zeiler, 2012), Adam (Kingma & Ba, 2015), AdaFTRL(Orabona & Pal, 2015), SGD-BB(Tan et al., 2016), AdaBatch (Defossez & Bach, 2017), SC-Adagrad (Mukkamala & Hein, 2017), AMSGRAD (Reddi et al., 2018), Padam (Chen & Gu, 2018), etc. Extending our convergence analysis to these popular alternative adaptive gradient methods remains an interesting problem for future research.

Regarding the convergence guarantees for the norm version of adaptive gradient methods in the offline setting, the recent work by Levy (2017) introduces a family of adaptive gradient methods inspired by AdaGrad, and proves convergence rates in the setting of (strongly) convex loss functions without knowing the smoothness parameter L in advance. Yet, that analysis still requires the a priori knowledge of a convex set $\mathcal{K}$ with known diameter D in which the global minimizer resides. More recently, Wu et al. (2018) provids convergence guarantees in the non-convex setting for a different adaptive gradient algorithm, WNGrad, which is closely related to AdaGrad-Norm and inspired by weight normalization (Salimans & Kingma, 2016). In fact, the

Algorithm 1 AdaGrad-Norm

```
1: Input: Initialize $x_0 \in \mathbb{R}^d, b_0 > 0, \eta > 0$ and $N$
2: for $j = 1$ to $N$ do
3:     Generate $\xi_{j-1}$ and $G_{j-1} = G(x_{j-1}, \xi_{j-1})$
4:     $b_j^2 \leftarrow b_{j-1}^2 + \|G_{j-1}\|^2$
5:     $x_j \leftarrow x_{j-1} - \frac{\eta}{b_j} G_{j-1}$
6: end for
```

WNGrad stepsize update is similar to AdaGrad-Norm's:

$$\begin{aligned}\text{(WNGrad)} \quad & b_{j+1} = b_j + \|\nabla F(x_j)\|/b_j; \\ \text{(AdaGrad-Norm)} \quad & b_{j+1} = b_j + \|\nabla F(x_j)\|/(b_j + b_{j+1}).\end{aligned}$$

However, the guaranteed convergence in Wu et al. (2018) is only for the batch setting and the constant in the convergence rate is worse than the one provided here for AdaGrad-Norm. Independently, Li & Orabona (2018) also proves the $O(1/\sqrt{N})$ convergence rate for a variant of AdaGrad-Norm in the non-convex stochastic setting, but their analysis requires knowledge of of smoothness constant L and a hard threshold of $b_0 > \eta L$ for their convergence. In contrast to Li & Orabona (2018), we do not require knowledge of the Lipschitz smoothness constant L, but we do assume that the gradient ∇F is uniformly bounded by some (unknown) finite value, while Li & Orabona (2018) only assumes bounded variance $\mathbb{E}_\xi \left[\|G(x;\xi) - \nabla F(x)\|^2\right] \leq \sigma^2$.

1.3. Notation

Throughout, $\|\cdot\|$ denotes the ℓ_2 norm. We use the notation $[N] := \{0, 1, 2, \dots, N\}$. A function $F : \mathbb{R}^d \to \mathbb{R}$ has L-Lipschitz smooth gradient if

$$\|\nabla F(x) - \nabla F(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^d \qquad (3)$$

If $L > 0$ is the smallest number such that the above is satisfied, then we write $F \in C_L^1$ and refer to L as the smoothness constant for F.

2. AdaGrad Convergence

To be clear about the adaptive algorithm, we stateAdaGrad-Norm update in Algorithm 1 for our analysis.

At the kth iteration, we observe a *stochastic gradient* $G(x_k, \xi_k)$, where $\xi_k, k = 0, 1, 2\dots$ are random variables, and such that $\mathbb{E}_{\xi_k}[G(x_k, \xi_k)] = \nabla F(x_k)$ is an unbiased estimator of $\nabla F(x_j)$. We require the following additional assumptions: for each $k \geq 0$,

1. The random vectors ξ_k, $k = 0, 1, 2, \dots$, are independent of each other and also of x_k;
2. $\mathbb{E}_{\xi_k}[\|G(x_k, \xi_k) - \nabla F(x_k)\|^2] \leq \sigma^2$;
3. $\|\nabla F(x_k)\| \leq \gamma$ uniformly.

The first two assumptions are standard (see e.g. Nemirovski & Yudin (1983); Nemirovski et al. (2009); Bottou et al. (2018)). The third assumption is somewhat restrictive as it rules out strongly convex objectives, but is not an unreasonable assumption for AdaGrad-Norm, where the adaptive stepsize is a cumulative sum of all previous observed gradient norms.

Because of the variance in gradient, the AdaGrad-Norm stepsize $\frac{\eta}{b_k}$ decreases to zero roughly at a rate between $\frac{1}{\sqrt{2(\gamma^2+\sigma^2)k}}$ and $\frac{1}{\sigma\sqrt{k}}$. It is known that AdaGrad-Norm stepsize decreases at this rate (Levy, 2017), and that this rate is optimal in k in terms of the resulting convergence theorems in the setting of smooth but not necessarily convex F, or convex but not necessarily strongly convex or smooth F. Still, standard convergence theorems for SGD do not extend straightforwardly to AdaGrad-Norm because the stepsize is a random variable and dependent on all previous points visited along the way. From this point on, we use the shorthand $G_k = G(x_k, \xi_k)$ for simplicity of notation.

Theorem 2.1. *Suppose $F \in C_L^1$ and $F^* = \inf_x F(x) > -\infty$. Suppose that the random variables $G_\ell, \ell \geq 0$, satisfy the above assumptions. Then with probability $1 - \delta$,*

$$\min_{\ell \in [N-1]} \|\nabla F_\ell\|^2 \leq \left(\frac{2b_0}{N} + \frac{2\sqrt{2}(\gamma + \sigma)}{\sqrt{N}}\right)\frac{\mathcal{Q}}{\delta^{3/2}}$$

$$\textit{and} \quad \min_{k \in [N-1]} \|\nabla F_k\|^2 \leq \frac{4\mathcal{Q}}{N\delta}\left(\frac{8\mathcal{Q}}{\delta} + 2b_0\right) + \frac{8\mathcal{Q}\sigma}{\delta^{3/2}\sqrt{N}}$$

where

$$\mathcal{Q} = \frac{F_0 - F^*}{\eta} + \frac{4\sigma + \eta L}{2}\log\left(\frac{20N(\gamma^2 + \sigma^2)}{b_0^2} + 10\right).$$

Due to page limit, we give proof for the first bound, i.e.,

$$\min_{\ell \in [N-1]} \|\nabla F_\ell\|^2 \leq \left(\frac{2b_0}{N} + \frac{2\sqrt{2}(\gamma + \sigma)}{\sqrt{N}}\right)\frac{\mathcal{Q}}{\delta^{3/2}}$$

in Section 3 while deferring the second one to the appendix.

This result implies that AdaGrad-Norm converges starting from any value of b_0 for a given η. To put this result in context, we can compare to Corollary 2.2 of Ghadimi & Lan (2013), which implies that under the same conditions and Assumption (1) and (2) but not (3), if the Lipschitz constant L and the variance σ are known a priori, and the step-size is

$$\eta_j = \frac{1}{b_j} = \frac{1}{b} = \min\left\{\frac{1}{L}, \frac{1}{\sigma\sqrt{N}}\right\}, \quad j = 0, 1, \dots, N-1,$$

then with probability $1 - \delta$

$$\min_{\ell \in [N-1]} \|\nabla F_\ell\|^2 \leq \frac{2L(F_0 - F^*)}{N\delta} + \frac{(L + 2(F_0 - F^*))\sigma}{\delta\sqrt{N}}.$$

Thus, essentially, we match the $O(1/\sqrt{N})$ rate of Ghadimi & Lan (2013), but *without a priori knowledge of* L *and* σ. The constant in the $O(1/\sqrt{N})$ rate of AdaGrad-Norm scales according to σ^2 (up to a logarithmic factors in σ) while the results with well-tuned stepsize scales linearly with σ.

We reiterate however that the main emphasis in Theorem 2.1 is on the robustness of the AdaGrad-Norm convergence to its hyperparameters η and b_0, compared to plain SGD's dependence on its parameters η and σ. Although the constant in the rate of our theorem is not as good as the best-known constant for stochastic gradient descent with well-tuned fixed stepsize, our result suggests that *implementing AdaGrad-Norm allows one to vastly reduce the need to perform laborious experiments to find a stepsize schedule with reasonable convergence when implementing SGD*. Practically, our results imply a good strategy for setting the hyperparameters when implementing AdaGrad-Norm: set $\eta = (F(x_0) - F^*)$ (if F^* is known) and set $b_0 > 0$ to be a very small value. If F^* is unknown, then setting $\eta = 1$ should work well for a wide range of values of L, and in the noisy case with σ^2 strictly greater than zero.

We note that for the second bound in 2.1, in the limit as $\sigma \to 0$ we recover an $O(\log(N)/N)$ rate of convergence for noiseless gradient descent. We can establish a stronger result in the noiseless setting using a different method of proof, removing the additional log factor and Assumption 3 of uniformly bounded gradient. We state the theorem below and defer our proof to the appendix.

Theorem 2.2. *Suppose that* $F \in C_L^1$ *with* $F^* = \inf_x F(x) > -\infty$. *Consider AdaGrad in deterministic setting with following update,*

$$x_j = x_{j-1} - \frac{\eta}{b_j} \nabla F_{j-1} \quad \textit{with} \quad b_j^2 = b_{j-1}^2 + \|\nabla F_{j-1}\|^2$$

Then $\min_{j \in [N]} \|\nabla F(x_j)\|^2 \le \varepsilon$ *for*
if $\frac{b_0}{\eta} \ge L$ *:*

$$N = 1 + \left\lceil \frac{2(F(x_0) - F^*)(b_0 + 2(F(x_0) - F^*)/\eta)}{\eta\varepsilon} \right\rceil,$$

if $\frac{b_0}{\eta} < L$ *:*

$$N = 1 + \left\lceil \frac{(\eta L)^2 - b_0^2}{\varepsilon} + \frac{4\left((F(x_0) - F^*)/\eta + \left(\frac{3}{4} + \log\left(\frac{\eta L}{b_0}\right)\right)\eta L\right)^2}{\varepsilon} \right\rceil.$$

3. Proof of Theorem 2.1

We first introduce two important lemmas in Subsection 3.1 and give the proof of the first bound in Subsection 3.2, and defer the proof of the second bound to the appendix.

3.1. Ingredients of The Proof

The following two lemmas will be used in the proof for Theorem 2.1. We repeatedly appeal to the following classical Descent Lemma, which is also the main ingredient in Ghadimi & Lan (2013), and can be proved by considering the Taylor expansion of F around y.

Lemma 3.1 (Descent Lemma)**.** *Let* $F \in C_L^1$. *Then,*

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

We will also use the following lemmas concerning sums of non-negative sequences.

Lemma 3.2. *For any non-negative* $a_1, \cdots, a_T$, *and* $a_1 \ge 1$, *we have*

$$\sum_{\ell=1}^{T} \frac{a_\ell}{\sum_{i=1}^{\ell} a_i} \le \log\left(\sum_{i=1}^{T} a_i\right) + 1. \tag{4}$$

Proof. The lemma can be proved by induction. That the sum should be proportional to $\log\left(\sum_{i=1}^{T} a_i\right)$ can be seen by associating to the sequence a continuous function $g : \mathbb{R}^+ \to \mathbb{R}$ satisfying $g(\ell) = a_\ell, 1 \le \ell \le T$, and $g(t) = 0$ for $t \ge T$, and replacing sums with integrals. □

3.2. The Proof

Proof. By Descent Lemma 3.1, for $j \ge 0$,

$$\begin{aligned} \frac{F_{j+1} - F_j}{\eta} &\le -\langle \nabla F_j, \frac{G_j}{b_{j+1}} \rangle + \frac{\eta L}{2b_{j+1}^2}\|G_j\|^2 \\ &= -\frac{\|\nabla F_j\|^2}{b_{j+1}} + \frac{\langle \nabla F_j, \nabla F_j - G_j \rangle}{b_{j+1}} + \frac{\eta L \|G_j\|^2}{2b_{j+1}^2}. \end{aligned}$$

At this point, we cannot apply the standard method of proof for SGD, since b_{j+1} and G_j are correlated random variables and thus, in particular, for the conditional expectation

$$\begin{aligned} \mathbb{E}_{\xi_j}\left[\frac{\langle \nabla F_j, \nabla F_j - G_j \rangle}{b_{j+1}}\right] &\ne \frac{\mathbb{E}_{\xi_j}\left[\langle \nabla F_j, \nabla F_j - G_j \rangle\right]}{b_{j+1}} \\ &= \frac{1}{b_{j+1}} \cdot 0; \end{aligned}$$

If we had a closed form expression for $\mathbb{E}_{\xi_j}[\frac{1}{b_{j+1}}]$, we would proceed by bounding this term as

$$\begin{aligned} &\left|\mathbb{E}_{\xi_j}\left[\frac{1}{b_{j+1}}\langle \nabla F_j, \nabla F_j - G_j \rangle\right]\right| \\ =&\left|\mathbb{E}_{\xi_j}\left[\left(\frac{1}{b_{j+1}} - \mathbb{E}_{\xi_j}\left[\frac{1}{b_{j+1}}\right]\right)\langle \nabla F_j, \nabla F_j - G_j \rangle\right]\right| \\ \le&\mathbb{E}_{\xi_j}\left[\left|\frac{1}{b_{j+1}} - \mathbb{E}_{\xi_j}\left[\frac{1}{b_{j+1}}\right]\right| \|\langle \nabla F_j, \nabla F_j - G_j \rangle\|\right]. \end{aligned} \tag{5}$$

Since we do not have a closed form expression for $\mathbb{E}_{\xi_j}[\frac{1}{b_{j+1}}]$ though, we use the estimate $\frac{1}{\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}}$ as a surrogate for $\mathbb{E}_{\xi_j}[\frac{1}{b_{j+1}}]$ to proceed. Condition on $\xi_1, \ldots, \xi_{j-1}$ and take expectation with respect to ξ_j,

$$0 = \frac{\mathbb{E}_{\xi_j}\left[\langle \nabla F_j, \nabla F_j - G_j \rangle\right]}{\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}} = \mathbb{E}_{\xi_j}\left[\frac{\langle \nabla F_j, \nabla F_j - G_j \rangle}{\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}}\right]$$

thus,

$$\begin{aligned}
&\frac{\mathbb{E}_{\xi_j}[F_{j+1}] - F_j}{\eta} \\
\leq &\mathbb{E}_{\xi_j}\left[\frac{\langle \nabla F_j, \nabla F_j - G_j\rangle}{b_{j+1}} - \frac{\langle \nabla F_j, \nabla F_j - G_j\rangle}{\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}}\right] \\
&- \mathbb{E}_{\xi_j}\left[\frac{\|\nabla F_j\|^2}{b_{j+1}}\right] + \mathbb{E}_{\xi_j}\left[\frac{L\eta\|G_j\|^2}{2b_{j+1}^2}\right] \\
\leq &\mathbb{E}_{\xi_j}\left[\left(\frac{1}{\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}} - \frac{1}{b_{j+1}}\right)\langle \nabla F_j, G_j\rangle\right] \\
&- \frac{\|\nabla F_j\|^2}{\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}} + \frac{\eta L}{2}\mathbb{E}_{\xi_j}\left[\frac{\|G_j\|^2}{b_{j+1}^2}\right] \qquad (6)
\end{aligned}$$

Now, observe the term in (6)

$$\begin{aligned}
&\frac{1}{\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}} - \frac{1}{b_{j+1}} \\
= &\frac{(\|G_j\| - \|\nabla F_j\|)(\|G_j\| + \|\nabla F_j\|) - \sigma^2}{b_{j+1}\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}\left(\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2} + b_{j+1}\right)} \\
\leq &\frac{|\|G_j\| - \|\nabla F_j\||}{b_{j+1}\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}} + \frac{\sigma}{b_{j+1}\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}}
\end{aligned}$$

thus, applying Cauchy-Schwarz to the first term in (6),

$$\begin{aligned}
&\mathbb{E}_{\xi_j}\left[\left(\frac{1}{\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}} - \frac{1}{b_{j+1}}\right)\langle \nabla F_j, G_j\rangle\right] \\
\leq &\mathbb{E}_{\xi_j}\left[\frac{|\|G_j\| - \|\nabla F_j\||\,\|G_j\|\|\nabla F_j\|}{b_{j+1}\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}}\right] \\
&+ \mathbb{E}_{\xi_j}\left[\frac{\sigma\|G_j\|\|\nabla F_j\|}{b_{j+1}\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}}\right] \qquad (7)
\end{aligned}$$

By applying the inequality $ab \leq \frac{\lambda}{2}a^2 + \frac{1}{2\lambda}b^2$ with $\lambda = \frac{2\sigma^2}{\sqrt{b_j^2+\|\nabla F_j\|^2+\sigma^2}}$, $a = \frac{\|G_j\|}{b_{j+1}}$, and $b = \frac{|\|G_j\|-\|\nabla F_j\||\|\nabla F_j\|}{b_j^2+\|\nabla F_j\|^2}$, the first term in (7) can be bounded as

$$\begin{aligned}
&\mathbb{E}_{\xi_j}\left[\frac{|\|G_j\| - \|\nabla F_j\||\,\|G_j\|\|\nabla F_j\|}{b_{j+1}\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}}\right] \\
\leq &\frac{\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}}{4\sigma^2}\frac{\|\nabla F_j\|^2\mathbb{E}_{\xi_j}\left[(\|G_j\| - \|\nabla F_j\|)^2\right]}{b_j^2 + \|\nabla F_j\|^2 + \sigma^2} \\
&+ \frac{\sigma^2}{\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}}\mathbb{E}_{\xi_j}\left[\frac{\|G_j\|^2}{b_{j+1}^2}\right] \\
\leq &\frac{\|\nabla F_j\|^2}{4\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}} + \sigma\mathbb{E}_{\xi_j}\left[\frac{\|G_j\|^2}{b_{j+1}^2}\right] \qquad (8)
\end{aligned}$$

where the last inequality due to the fact that

$$|\|G_j\| - \|\nabla F_j\|| \leq \|G_j - \nabla F_j\|.$$

Similarly, applying the inequality $ab \leq \frac{\lambda}{2}a^2 + \frac{1}{2\lambda}b^2$ with $\lambda = \frac{2}{\sqrt{b_j^2+\|\nabla F_j\|^2+\sigma^2}}$, $a = \frac{\sigma\|G_j\|}{b_{j+1}}$, and $b = \frac{\|\nabla F_j\|}{\sqrt{b_j^2+\|\nabla F_j\|^2+\sigma^2}}$, the second term of the right hand side in equation (7) is bounded by

$$\begin{aligned}
&\mathbb{E}_{\xi_j}\left[\frac{\sigma\|\nabla F_j\|\|G_j\|}{b_{j+1}\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}}\right] \\
\leq &\sigma\mathbb{E}_{\xi_j}\left[\frac{\|G_j\|^2}{b_{j+1}^2}\right] + \frac{\|\nabla F_j\|^2}{4\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}}. \qquad (9)
\end{aligned}$$

Thus, puting inequalities (8) and (9) back into (7) gives

$$\begin{aligned}
&\mathbb{E}_{\xi_j}\left[\left(\frac{1}{\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}} - \frac{1}{b_{j+1}}\right)\langle \nabla F_j, G_j\rangle\right] \\
\leq\ &2\sigma\mathbb{E}_{\xi_j}\left[\frac{\|G_j\|^2}{b_{j+1}^2}\right] + \frac{\|\nabla F_j\|^2}{2\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}}
\end{aligned}$$

and, therefore, back to (6),

$$\begin{aligned}
\frac{\mathbb{E}_{\xi_j}[F_{j+1}] - F_j}{\eta} \leq &\frac{\eta L}{2}\mathbb{E}_{\xi_j}\left[\frac{\|G_j\|^2}{b_{j+1}^2}\right] + 2\sigma\mathbb{E}_{\xi_j}\left[\frac{\|G_j\|^2}{b_{j+1}^2}\right] \\
&- \frac{\|\nabla F_j\|^2}{2\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}}
\end{aligned}$$

Rearranging,

$$\begin{aligned}
\frac{\|\nabla F_j\|^2}{2\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}} \leq &\frac{F_j - \mathbb{E}_{\xi_j}[F_{j+1}]}{\eta} \\
&+ \frac{4\sigma + \eta L}{2}\mathbb{E}_{\xi_j}\left[\frac{\|G_j\|^2}{b_{j+1}^2}\right]
\end{aligned}$$

Applying the law of total expectation, we take the expectation of each side with respect to $\xi_{j-1}, \xi_{j-2}, \ldots, \xi_1$, and arrive at the recursion

$$\begin{aligned}
&\mathbb{E}\left[\frac{\|\nabla F_j\|^2}{2\sqrt{b_j^2 + \|\nabla F_j\|^2 + \sigma^2}}\right] \\
\leq &\frac{\mathbb{E}[F_j] - \mathbb{E}[F_{j+1}]}{\eta} + \frac{4\sigma + \eta L}{2}\mathbb{E}\left[\frac{\|G_j\|^2}{b_{j+1}^2}\right].
\end{aligned}$$

Taking $j = N$ and summing up from $k = 0$ to $k = N - 1$,

$$\sum_{k=0}^{N-1} \mathbb{E}\left[\frac{\|\nabla F_k\|^2}{2\sqrt{b_k^2 + \|\nabla F_k\|^2 + \sigma^2}}\right] \leq \frac{F_0 - F^*}{\eta} + \frac{4\sigma + \eta L}{2}\mathbb{E}\sum_{k=0}^{N-1}\left[\frac{\|G_k\|^2}{b_{k+1}^2}\right]. \quad (10)$$

For term of left hand side in equation (10), we apply Hölder's inequality,

$$\frac{\mathbb{E}|XY|}{(\mathbb{E}|Y|^3)^{\frac{1}{3}}} \leq \left(\mathbb{E}|X|^{\frac{3}{2}}\right)^{\frac{2}{3}}$$

$$\text{with } X = \left(\frac{\|\nabla F_k\|^2}{\sqrt{b_k^2 + \|\nabla F_k\|^2 + \sigma^2}}\right)^{\frac{2}{3}}$$

$$\text{and } Y = \left(\sqrt{b_k^2 + \|\nabla F_k\|^2 + \sigma^2}\right)^{\frac{2}{3}} \quad \text{to obtain}$$

$$\mathbb{E}\left[\frac{\|\nabla F_k\|^2}{2\sqrt{b_k^2 + \|\nabla F_k\|^2 + \sigma^2}}\right] \geq \frac{\left(\mathbb{E}\|\nabla F_k\|^{\frac{4}{3}}\right)^{\frac{3}{2}}}{2\sqrt{\mathbb{E}\left[b_k^2 + \|\nabla F_k\|^2 + \sigma^2\right]}} \geq \frac{\left(\mathbb{E}\|\nabla F_k\|^{\frac{4}{3}}\right)^{\frac{3}{2}}}{2\sqrt{b_0^2 + 2(k+1)(\gamma^2 + \sigma^2)}}$$

where the last inequality is due to

$$\mathbb{E}\left[b_k^2 - b_{k-1}^2\right] \leq \mathbb{E}\left[\|G_k\|^2\right] \leq 2\mathbb{E}\left[\|G_k - \nabla F_k\|^2\right] + 2\mathbb{E}\left[\|\nabla F_k\|^2\right] \leq 2\sigma^2 + 2\gamma^2.$$

As for the second term of right hand side in equation (10), we apply Lemma (3.2) and then Jensen's inequality to bound the final summation:

$$\mathbb{E}\sum_{k=0}^{N-1}\left[\frac{\|G_k\|^2}{b_{k+1}^2}\right] \leq \mathbb{E}\left[1 + \log\left(1 + \sum_{k=0}^{N-1}\|G_k\|^2/b_0^2\right)\right] \leq \log\left(10 + \frac{20N\left(\sigma^2 + \gamma^2\right)}{b_0^2}\right).$$

Thus (10) arrives at the inequality

$$\min_{k\in[N-1]}\left(\mathbb{E}\left[\|\nabla F_k\|^{\frac{4}{3}}\right]\right)^{\frac{3}{2}}\frac{N}{2\sqrt{b_0^2 + 2N(\gamma^2 + \sigma^2)}} \leq \frac{F_0 - F^*}{\eta} + \frac{4\sigma + \eta L}{2}\left(\log\left(1 + \frac{2N\left(\sigma^2 + \gamma^2\right)}{b_0^2}\right) + 1\right).$$

Multiplying by $\frac{2b_0 + 2\sqrt{2N}(\gamma+\sigma)}{N}$, the above inequality gives

$$\min_{k\in[N-1]}\left(\mathbb{E}\left[\|\nabla F_k\|^{\frac{4}{3}}\right]\right)^{\frac{3}{2}} \leq \underbrace{\left(\frac{2b_0}{N} + \frac{2\sqrt{2}(\gamma+\sigma)}{\sqrt{N}}\right)\mathcal{Q}}_{C_N}$$

where

$$\mathcal{Q} = \frac{F_0 - F^*}{\eta} + \frac{4\sigma + \eta L}{2}\log\left(\frac{20N\left(\sigma^2 + \gamma^2\right)}{b_0^2} + 10\right).$$

Finally, the bound is obtained by Markov's Inequality:

$$\mathbb{P}\left(\min_{k\in[N-1]}\|\nabla F_k\|^2 \geq \frac{C_N}{\delta^{3/2}}\right) = \mathbb{P}\left(\min_{k\in[N-1]}\left(\|\nabla F_k\|^2\right)^{2/3} \geq \left(\frac{C_N}{\delta^{3/2}}\right)^{2/3}\right) \leq \delta\frac{\mathbb{E}\left[\min_{k\in[N-1]}\|\nabla F_k\|^{4/3}\right]}{C_N^{2/3}} \leq \delta$$

where the second inequality applies Jensen's inequality to the concave function $\phi(x) = \min_k h_k(x)$. □

4. Numerical Experiments

With guaranteed convergence of AdaGrad-Norm and its strong robustness to the choice of hyper-parameters η and b_0, we perform experiments on several data sets ranging from simple linear regression on Gaussian data to neural network architectures on state- of-the-art (SOTA) image data sets including ImageNet.

4.1. Synthetic Data

In this section, we consider linear regression to corroborate our analysis, i.e.,

$$F(x) = \frac{1}{2m}\|Ax - y\|^2 = \frac{1}{(m/n)}\sum_{k=1}^{m/n}\frac{1}{2n}\|A_{\xi_k}x - y_{\xi_k}\|^2$$

where $A \in \mathbb{R}^{m\times d}$, m is the total number of samples, n is the mini-batch (small sample) size for each iteration, and $A_{\xi_k} \in \mathbb{R}^{n\times d}$. Then AdaGrad-Norm update is

$$x_{j+1} = x_j - \frac{\eta A_{\xi_j}^T\left(A_{\xi_j}x_j - y_{\xi_j}\right)/n}{\sqrt{b_0^2 + \sum_{\ell=0}^{j}\left(\|A_{\xi_\ell}^T\left(A_{\xi_\ell}x_\ell - y_{\xi_\ell}\right)\|/n\right)^2}}.$$

We simulate $A \in \mathbb{R}^{1000\times 2000}$ and $x^* \in \mathbb{R}^{1000}$ such that each entry of A and x^* is an i.i.d. standard Gaussian. Let $y = Ax^*$. For each iteration, we independently draw a small sample of size $n = 20$ and x_0 whose entries follow i.i.d. uniform in $[0, 1]$. The vector x_0 is same for all the methods so as to eliminate the effect of random initialization in weight vector. Since $F^* = 0$, we set $\eta = F(x_0) - F^* = \frac{1}{2m}\|Ax_0 - b\|^2 = 650$. We vary the initialization $b_0 > 0$ as to compare with plain SGD using (a) SGD-Constant: fixed stepsize $\frac{650}{b_0}$, and (b) SGD-DecaySqrt: decaying stepsize $\eta_j = \frac{650}{b_0\sqrt{j}}$. Figure 1 plots

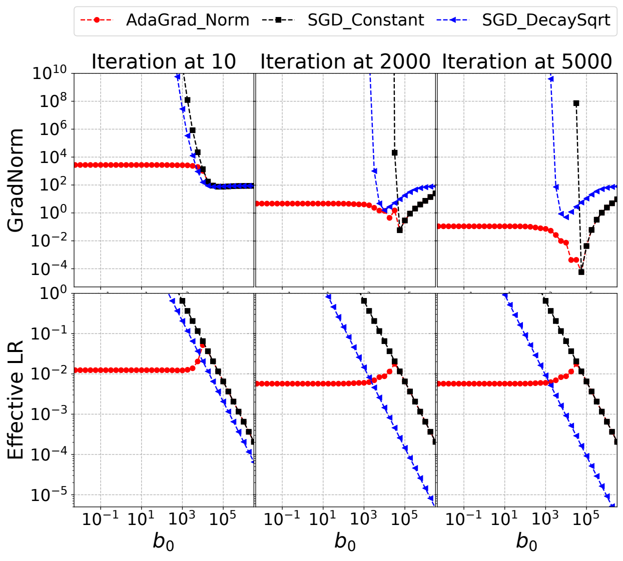

Figure 1: Gaussian Data. The top 3 figures plot the gradient norm for linear regression, $\|A^T(Ax_j - y)\|/m$, w.r.t. b_0 at iteration 10, 2000 and 5000 (see title) respectively. The bottom 3 figures plot the corresponding effective learning rates (LR) w.r.t. b_0 at iteration 10, 2000 and 5000 respectively. Note black curve overlaps with red curve when $b_0 >= 56234$.

$\|A^T(Ax_j - y)\|/m$ (GradNorm) and the effective learning rates at iterations 10, 2000, and 5000, and as a function of b_0, for each of the three methods. The effective learning rates are $\frac{650}{b_j}$ (AdaGrad-Norm), $\frac{650}{b_0}$(SGD-Constant), and $\frac{650}{b_0\sqrt{j}}$(SGD-DecaySqrt).

We can see in Figure 1 how AdaGrad-Norm auto-tunes the learning rate to a certain level to match the unknown Lipschitz smoothness constant and the stochastic noise so that the gradient norm converges for a significantly wider range of b_0 than for either SGD method. In particular, when b_0 is initialized too small, AdaGrad-Norm still converges with good speed while SGD-Constant and SGD-DecaySqrt diverge. When b_0 is initialized too large (stepsize too small), surprisingly AdaGrad-Norm converges at the same speed as SGD-Constant. This possibly can be explained by Theorem 2.2 because this is somewhat like the deterministic setting (the stepsize controls the variance σ and a smaller learning rate implies smaller variance).

4.2. Image Data

In this section, we extend our numerical analysis to the setting of deep learning and show that the robustness of AdaGrad-Norm does not come at the price of worse generalization – an important observation that is not explained by our current theory. The experiments are done in PyTorch (Paszke et al., 2017) and parameters are by default if no specification is provided.[1] We did not find it practical to compute the norm of the gradient for the entire neural network during back-propagation.[2] Instead, we adopt a stepsize for each neuron or each convolutional channel by updating b_j with the gradient of the neuron or channel. Hence, our experiments depart slightly from a strict AdaGrad-Norm method and include a limited adaptive metric component. Details in implementing AdaGrad-Norm in a neural network are explained in the appendix and the code is also provided.[3]

Datasets and Models We test on three data sets: MNIST (LeCun et al., 1998), CIFAR-10 (Krizhevsky, 2009) and ImageNet (Deng et al., 2009), see Table 1 in the appendix for detailed descriptions. For MNIST, our models are a logistic regression (LR), a multilayer network with two fully connected layers (FC) with 100 hidden units and ReLU activations. For CIFAR10, our models are LeNet (with ReLU as activation functions instead of sigmoid function, see Table 2 in the appendix for details) and ResNet-18 (He et al., 2016). For both data sets, we use simple SGD without momentum and set mini-batch of 128 images per iteration (2 GPUs with 64 images/GPU, 468 iterations per epoch for MNIST and 390 iterations per epoch for CIFAR10) and repeat five trails to avoid the random initialization effect. For ImagetNet, we use ResNet-50 with no momentum and 256 images for one iteration (8 GPUs with 32 images/GPU, 5004 iterations per epoch). Note that we do not use accelerated methods such as adding momentum in the training. In the setting of ResNet, the differences from the standard setup are the batch normalization layers where we set no learnable parameters in batch normalization so as to minimize the effect of this optimization method. In addition, we set the initialization of weights in the last fully connected layer to be i.i.d. Gaussian with zero mean and variance $1/2048$. [4]

We pick these models for the following reasons: (1) LR with MNIST represents the smooth loss function; (2) FC with MNIST represents non-smooth loss function; (3) LeNet in CIFAR10 belongs to a class of simple shallow network architectures; (4) ResNet-18 in CIFAR10 represents a complicated network architecture involving many other added features for SOTA; (5) ResNet-50 in ImageNet represents large-scale data and a very deep network architecture.

We set $\eta = 1$ in AdaGrad-Norm implementations, noting that in all these problems we know that $F^* = 0$ and measure that $F(x_0)$ is between 1 and 10. Fixing all other parameters,

[1]The code we used is originally from `https://github.com/pytorch/examples/tree/master/imagenet`

[2]The computation time to obtain the gradient norm of the entire neural network is large.

[3]AdaGrad-Norm `https://github.com/xwuShirley/pytorch/blob/master/torch/optim/adagradnorm.py`

[4]Set *nn.BatchNorm2d(planes,affine=False)*, and use regularization (weight decay 0.0001) for both LeNet and ResNet.

we vary the initialization b_0 and plot the training and testing accuracy after different numbers of epochs. We compare AdaGrad-Norm with initial parameter b_0^2 to SGD with (a) $\frac{1}{b_0}$ (SGD-Constant) and (b) $\eta_j = \frac{1}{b_0\sqrt{j}}$ (SGD-DecaySqrt).

Observations The experiments shown in Figures 2, 3 and 4 clearly verify that AdaGrad-Norm convergence is extremely robust to the choice of b_0, while the SGD methods are much more sensitive. In all except the LeNet experiment, AdaGrad-Norm convergence degrades very slightly even for very small initial values b_0. Similar to Synthetic Data, when b_0 is initialized too small, AdaGrad-Norm still converges with good speed, while SGDs do not. When b_0 is initialized in the range of well-tune stepsizes, AdaGrad-Norm gives almost the same accuracy as constant SGD.

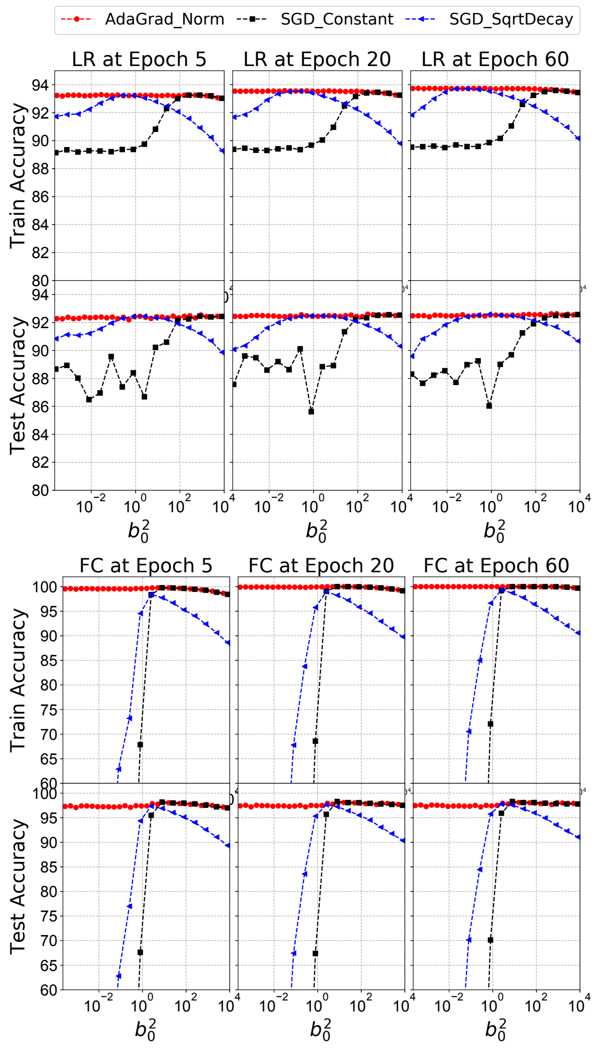

Figure 2: MNIST. In each plot, y-axis is train or test accuracy and x-axis is b_0^2. The top 6 plots are for logistic regression (LR) with snapshots at epoch 5, 20 and 60 (see title). The bottom 6 plots are for two fully connected (FC) layer.

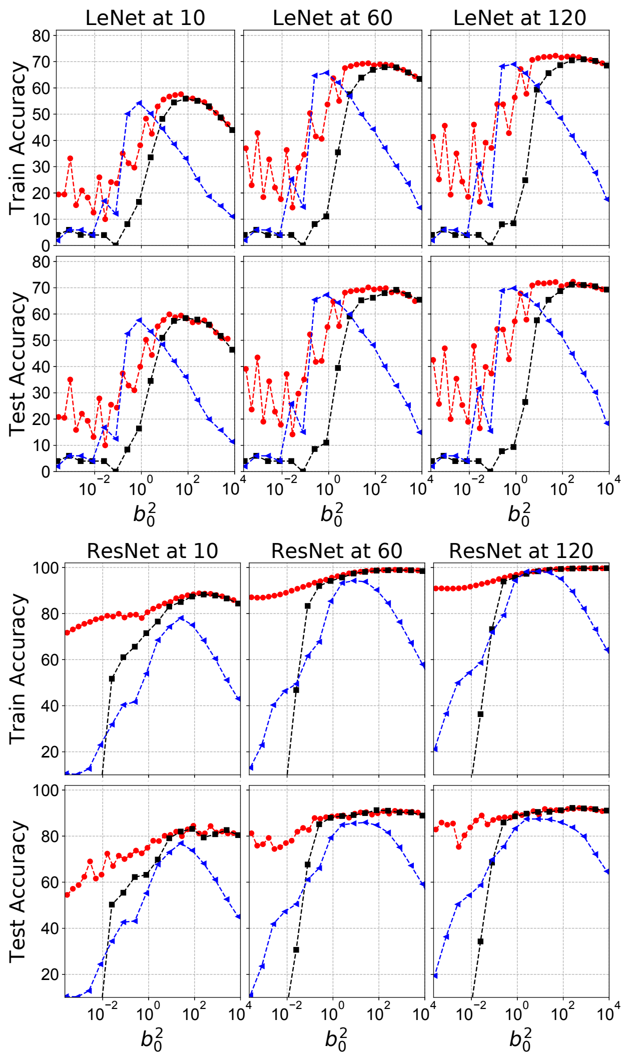

Figure 3: CIFAR10. Top 6 plots use LeNet and bottom 6 plots use ReNet-18. See Figure 2 for instruction.

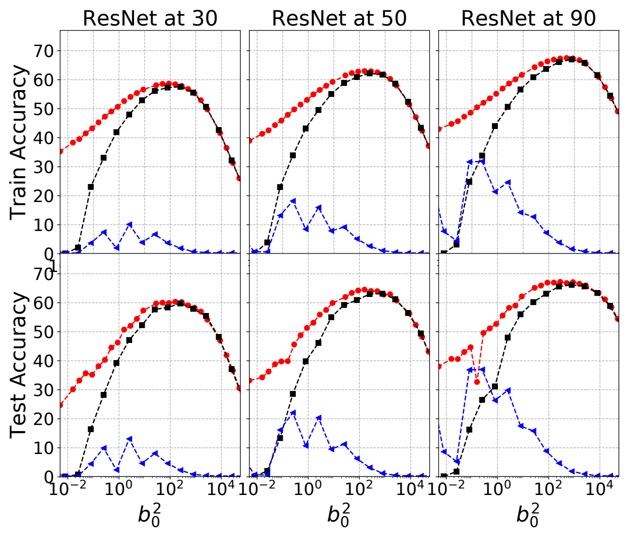

Figure 4: ImageNet with ReNet-50. See Figure 2 for instruction.

Acknowledgments

Special thanks to Kfir Levy for pointing us to his work, to Francesco Orabona for reading a previous version and pointing out a mistake, and to Krishna Pillutla for discussion on the unit mismatch in AdaGrad. We thank Arthur Szlam and Mark Tygert for constructive suggestions. We also thank Francis Bach, Alexandre Defossez, Ben Recht, Stephen Wright, and Adam Oberman. We appreciate the help with the experiments from Priya Goyal, Soumith Chintala, Sam Gross, Shubho Sengupta, Teng Li, Ailing Zhang, Zeming Lin, and Timothee Lacroix. Finally, we thank the anonymous reviewers for their helpful comments.

References

Agarwal, A., Wainwright, M., Bartlett, P., and Ravikumar, P. Information-theoretic lower bounds on the oracle complexity of convex optimization. In *Advances in Neural Information Processing Systems*, pp. 1–9, 2009.

Bottou, L. and Bousquet, O. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pp. 161–168, 2008.

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *SIAM Reviews*, 60(2):223–311, 2018. URL http://leon.bottou.org/papers/bottou-curtis-nocedal-2018.

Bubeck, S. et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

Chen, J. and Gu, Q. Closing the generalization gap of adaptive gradient methods in training deep neural networks. *arXiv preprint arXiv:1806.06763*, 2018.

Defossez, A. and Bach, F. Adabatch: Efficient gradient aggregation rules for sequential and parallel stochastic gradient methods. *arXiv preprint arXiv:1711.01761*, 2017.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Li., F. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Ghadimi, S. and Lan, G. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Levy, K. Online to offline conversions, universality and adaptive minibatch sizes. In *Advances in Neural Information Processing Systems*, pp. 1612–1621, 2017.

Li, X. and Orabona, F. On the convergence of stochastic gradient descent with adaptive stepsizes. *arXiv preprint arXiv:1805.08114*, 2018.

McMahan, H. B. and Streeter, M. Adaptive bound optimization for online convex optimization. *COLT 2010*, pp. 244, 2010.

Mukkamala, M. C. and Hein, M. Variants of RMSProp and Adagrad with logarithmic regret bounds. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2545–2553, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Nemirovski, A. and Yudin, D. Problem complexity and method efficiency in optimization. 1983.

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

Orabona, F. and Pal, D. Scale-free algorithms for online linear optimization. In *ALT*, 2015.

Paszke, A., Gross, S., Chintala, S.and Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., and Antiga, L.and Lerer, A. Automatic differentiation in pytorch. 2017.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ryQu7f-RZ.

Robbins, H. and Monro, S. A stochastic approximation method. In *The Annals of Mathematical Statistics*, volume 22, pp. 400–407, 1951.

Salimans, T. and Kingma, D. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 901–909, 2016.

Srivastava, G. H. N. and Swersky, K. Neural networks for machine learning-lecture 6a-overview of mini-batch gradient descent, 2012.

Tan, C., Ma, S., Dai, Y.-H., and Qian, Y. Barzilai-borwein step size for stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 685–693, 2016.

Wilson, A., Roelofs, R., Stern, M., Srebro, N., and Recht, B. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pp. 4151–4161, 2017.

Wu, X., Ward, R., and Bottou, L. WNGrad: Learn the learning rate in gradient descent. *arXiv preprint arXiv:1803.02865*, 2018.

Zeiler, M. ADADELTA: an adaptive learning rate method. In *arXiv preprint arXiv:1212.5701*, 2012.