Journal of Machine Learning Research 21 (2020) 1-30 Submitted 6/18; Revised 11/20; Published 12/20

AdaGrad stepsizes: Sharp convergence over nonconvex landscapes

Rachel Ward * RWARD@MATH.UTEXAS.EDU
Xiaoxia Wu * XWU@MATH.UTEXAS.EDU
Department of Mathematics
The University of Texas at Austin
2515 Speedway, Austin, TX, 78712, USA

Léon Bottou LEONB@FB.COM
Facebook AI Research
770 Broadway, New York, NY, 10019, USA

Editor: Mark Schmidt

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 the stepsize schedule. Yet, the theoretical guarantees to date for AdaGrad are for online and convex optimization. We bridge this gap by providing theoretical guarantees for the convergence of AdaGrad for smooth, nonconvex functions. 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, the convergence of AdaGrad-Norm is robust to the choice of all hyper-parameters of the algorithm, in contrast to stochastic gradient descent whose convergence depends crucially on tuning the step-size to the (generally unknown) Lipschitz smoothness constant and level of stochastic noise on the gradient. Extensive numerical experiments are provided to corroborate our theoretical findings; moreover, the experiments suggest that the robustness of AdaGrad-Norm extends to the models in deep learning.

Keywords: nonconvex optimization, stochastic offline learning, large-scale optimization, adaptive gradient descent, convergence

1. Introduction

Consider the problem of minimizing a differentiable non-convex 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

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

∗. Equal Contribution; work done at Facebook AI Research.

where $\eta_j > 0$ is the stepsize at the jth iteration 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 the large-scale optimization problems where the loss function $F(x)$ can be approximated by the average of a large number m of component functions, $F(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x)$. It is more efficient to measure a single component gradient $\nabla f_{i_j}(x), i_j \sim \text{Uniform}\{1, 2, \ldots, m\}$ (or subset of component gradients), and move in the noisy direction $G_j(x) = \nabla f_{i_j}(x)$, than to compute a full gradient $\frac{1}{m}\sum_{i=1}^{m} \nabla f_i(x)$.

For non-convex but smooth loss functions F, (noiseless) gradient descent (GD) with constant stepsize converges to a stationary point of F at rate $\mathcal{O}(1/N)$ with the number of iterations N (Nesterov, 1998). In the same setting, and under the general assumption of bounded gradient noise variance, SGD with constant or decreasing stepsize $\eta_j = \mathcal{O}\left(1/\sqrt{j}\right)$ has been proven to converge to a stationary point of F at rate $\mathcal{O}\left(1/\sqrt{N}\right)$ (Ghadimi and Lan, 2013; Bottou et al., 2018). The $\mathcal{O}(1/N)$ rate for GD is the best possible worst-case dimension-free rate of convergence for any algorithm (Carmon et al., 2019); faster convergence rates in the noiseless setting are available under the mild assumption of additional smoothness (Agarwal et al., 2017; Carmon et al., 2017, 2018). In the noisy setting, faster rates than $\mathcal{O}\left(1/\sqrt{N}\right)$ are also possible using accelerated SGD methods (Ghadimi and Lan, 2016; Allen-Zhu and Yang, 2016; Reddi et al., 2016; Allen-Zhu, 2017; Xu et al., 2018; Zhou et al., 2018; Fang et al., 2018). For instance, Zhou et al. (2018) and Fang et al. (2018) obtain the rate $\mathcal{O}\left(1/N^{2/3}\right)$ without requiring finite-sum structure but with an additional assumptions about Lipschitz continuity of the stochastic gradients, which they exploit to reduce variance.

Instead of focusing on faster convergence rates for SGD, this paper focuses on adaptive stepsizes (Cutkosky and Boahen, 2017; Levy, 2017) that make the optimization algorithm more robust to (generally unknown) parameters of the optimization problem, such as the noise level of the stochastic gradient and the Lipschitz smoothness constant L of the loss function defined as the smallest number $L > 0$ such that $\|\nabla F(x) - \nabla F(y)\| \leq L\|x - y\|$ for all x, y. In particular, the $\mathcal{O}(1/N)$ convergence of GD with fixed stepsize is guaranteed only if the fixed stepsize $\eta > 0$ is carefully chosen such that $\eta \leq 1/L$ – choosing a larger stepsize η, even just by a factor of 2, can result in oscillation or divergence of the algorithm (Nesterov, 1998). Because of this sensitivity, GD with fixed stepsize is rarely used in practice; instead, one adaptively chooses the stepsize $\eta_j > 0$ at each iteration to approximately maximize a decrease of the loss function in the current direction of $-\nabla F(x_j)$ via either line search (Wright and Nocedal, 2006), or according to the Barzilai-Borwein rule (Barzilai and Borwein, 1988) combined with line search.

Unfortunately, in the noisy setting where one uses SGD for optimization, line search methods are not useful, as in this setting the stepsize should not be overfit to the noisy stochastic gradient direction at each iteration. The classical Robbins/Monro theory (Robbins and Monro, 1951) says that in order for $\lim_{k\to\infty} \mathbb{E}[\|\nabla F(x_k)\|^2] = 0$, the stepsize schedule should satisfy

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

However, these bounds do not tell us much about how to select a good stepsize schedule in practice, where algorithms are run for finite iterations and the constants in the rate of convergence matter.

The question of how to choose the stepsize $\eta > 0$ or stepsize or learning rate schedule $\{\eta_j\}$ for SGD is by no means resolved; in practice, a preferred schedule is chosen manually by testing many different schedules in advance and choosing the one leading to smallest training or generalization error. This process can take days or weeks, and can become prohibitively expensive in terms of time and computational resources incurred.

1.1 Stepsize adaptation with AdaGrad-Norm

Adaptive stochastic gradient methods such as *AdaGrad* (introduced independently by Duchi et al. (2011) and McMahan and Streeter (2010)) have been widely used in the past few years. AdaGrad updates the stepsize η_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 (Lafond et al., 2017). To be concrete, for optimizing a function $F : \mathbb{R}^d \to \mathbb{R}$, the "coordinate" version of AdaGrad updates d scalar parameters $b_j(k), k = 1, 2, \ldots, d$ at the j iteration – one for each $x_j(k)$ coordinate of $x_j \in \mathbb{R}^d$ – according to $b_{j+1}(k)^2 = b_j(k)^2 + [\nabla F(x_j)]_k^2$ in the noiseless setting, and $b_{j+1}(k)^2 = b_j(k)^2 + [G_j(k)]^2$ 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 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 optimal 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 a proof for AdaGrad-Norm.

1.2 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 differentiable 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 [1] at the rate

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

  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 and Lan (2013). However, our result requires bounded gradient $\|\nabla F(x)\|^2 \leq \gamma^2$ and our rate constant scales with $\gamma\sigma$ instead of linearly in σ. Nevertheless, our result suggests a good strategy for setting hyperparameters in implementing AdaGrad-Norm practically: given knowledge of F^*, set $\eta = F(x_0) - F^*$ and simply initialize $b_0 > 0$ to be very small.

- When there is no noise $\sigma = 0$, we can improve this rate to an $\mathcal{O}(1/N)$ rate of convergence. In Theorem 2.2, we show that $\min_{j\in[N]} \|\nabla F(x_j)\|^2 \leq \varepsilon$ after
  (1) $N = \mathcal{O}\left(\frac{1}{\varepsilon}\left(((F(x_0) - F^*)/\eta)^2 + b_0\left(F(x_0) - F^*\right)/\eta\right)\right)$ if $b_0 \geq \eta L$,
  (2) $N = \mathcal{O}\left(\frac{1}{\varepsilon}\left(L\left(F(x_0) - F^*\right) + ((F(x_0) - F^*)/\eta)^2\right) + \frac{(\eta L)^2}{\varepsilon}\log\left(\frac{\eta L}{b_0}\right)\right)$ if $b_0 < \eta L$.
  Note that the constant $(\eta L)^2$ in the second case when $b_0 < \eta L$ is not optimal compared to the known best rate constant ηL obtainable by gradient descent with fixed stepsize $\eta = 1/L$ (Carmon et al., 2019); on the other hand, given knowledge of L and $F(x_0) - F^*$, the rate constant of AdaGrad-norm reproduces the optimal constant ηL by setting $\eta = F(x_0) - F^*$ and $b_0 = \eta L$.

1. It is becoming common to define an ε-approximate stationary point as $\|\nabla F(x)\| \leq \varepsilon$ (Agarwal et al., 2017; Carmon et al., 2018, 2019; Fang et al., 2018; Zhou et al., 2018; Allen-Zhu, 2018), but we use the convention $\|F(x)\|^2 \leq \varepsilon$ (Lei et al., 2017; Bottou et al., 2018) to most easily compare our results to those from Ghadimi and Lan (2013); Li and Orabona (2019).

Practically, our results imply a good strategy for setting the hyperparameters when implementing AdaGrad-norm in practice: set $\eta = (F(x_0) - F^*)$ (assuming 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.

1.3 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 and Swersky, 2012), AdaDelta (Zeiler, 2012), Adam (Kingma and Ba, 2015), AdaFTRL(Orabona and Pal, 2015), SGD-BB(Tan et al., 2016), AdaBatch (Defossez and Bach, 2017), SC-Adagrad (Mukkamala and Hein, 2017), AMSGRAD (Reddi et al., 2018), Padam (Chen and 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 and Kingma, 2016). In fact, the 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 and Orabona (2019) 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 and Orabona (2019), 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 and Orabona (2019) only assumes bounded variance $\mathbb{E}_\xi \left[\|G(x;\xi) - \nabla F(x)\|^2\right] \leq \sigma^2$.

1.4 Future work

This paper provides convergence guarantees for AdaGrad-Norm over smooth, nonconvex functions, in both the stochastic and deterministic settings. Our theorems should shed light on the popularity of AdaGrad as a method for more robust convergence of SGD in nonconvex optimization in that the convergence guarantees we provide are robust to the initial

stepsize η/b_0, and adjust automatically to the level of stochastic noise. Moreover, our results suggest a good strategy for setting hyperparameters in AdaGrad-Norm implementation: set $\eta = (F(x_0) - F^*)$ (if F^* is known) and set $b_0 > 0$ to be a very small value. However, several improvements and extensions should be possible. First, the constant in the convergence rate we present can likely be improved and it remains open whether we can remove the assumption of the uniformly bounded gradient in the stochastic setting. It would be interesting to analyze AdaGrad in its coordinate form, where each coordinate $x(k)$ of $x \in \mathbb{R}^d$ has its own stepsize $\frac{1}{b_j(k)}$ which is updated according to $b_{j+1}(k)^2 = b_j(k)^2 + [\nabla F(x_j)]_k^2$. AdaGrad is just one particular adaptive stepsize method and other updates such as Adam (Kingma and Ba, 2015) are often preferable in practice; it would be nice to have similar theorems for other adaptive gradient methods, and to even use the theory as a guide for determining the "best" method for adapting the stepsize for given problem classes.

1.5 Notation

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

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

We write $F \in \mathbb{C}_L^1$ and refer to L as the smoothness constant for F if $L > 0$ is the smallest number such that the above is satisfied.

2. AdaGrad-Norm convergence

To be clear about the adaptive algorithm, we first state in Algorithm 1 the norm version of AdaGrad we consider throughout in the analysis.

Algorithm 1 AdaGrad-Norm

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

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

1. The random vectors ξ_k, $k = 0, 1, 2, \ldots$, 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. $\|\nabla F(x)\|^2 \leq \gamma^2$ uniformly.

2. $\mathbb{E}_{\xi_k}[G(x_k, \xi_k)] = \nabla F(x_k)$ where $\mathbb{E}_{\xi_k}[\cdot]$ is the expectation with respect ξ_k conditional on previous $\xi_0, \xi_1, \ldots, \xi_{k-1}$

The first two assumptions are standard (see e.g. Nemirovski and 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 learning rate 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 $1/b_k$ is a random variable and dependent on all previous points visited along the way, i.e., $\{\|\nabla F(x_j)\|\}_{j=0}^k$ and $\{\|\nabla G(x_j,\xi_j)\|\}_{j=0}^k$. From this point on, we use the shorthand $G_k = G(x_k,\xi_k)$ and $F_k = \nabla F(x_k)$ for simplicity of notation. The following theorem gives the convergence guarantee to Algorithm 1. We give detailed proof in Section 3.

Theorem 2.1 (AdaGrad-Norm: convergence in stochastic setting) *Suppose $F \in \mathbb{C}_L^1$ and $F^* = \inf_x F(x) > -\infty$. Suppose that the random variables $G_\ell, \ell \geq 0$, satisfy the above assumptions. Then with probability $1-\delta$,*

$$\min_{\ell\in[N-1]} \|\nabla F(x_\ell)\|^2 \leq \min\left\{ \left(\frac{2b_0}{N} + \frac{4(\gamma+\sigma)}{\sqrt{N}}\right)\frac{\mathcal{Q}}{\delta^{3/2}}, \left(\frac{8\mathcal{Q}}{\delta}+2b_0\right)\frac{4\mathcal{Q}}{N\delta} + \frac{8\mathcal{Q}\sigma}{\delta^{3/2}\sqrt{N}} \right\}$$

where

$$\mathcal{Q} = \frac{F(x_0)-F^*}{\eta} + \frac{4\sigma+\eta L}{2}\log\left(\frac{20N(\gamma^2+\sigma^2)}{b_0^2}+10\right).$$

This result implies that AdaGrad-Norm converges for any $\eta > 0$ and starting from any value of $b_0 > 0$. To put this result in context, we can compare to Corollary 2.2 of Ghadimi and Lan (2013) giving the best-known convergence rate for SGD with fixed step-size in the same setting (albeit not requiring Assumption (3) of uniformly bounded gradient): if the Lipschitz smoothness constant L and the variance σ^2 are known a priori, and the fixed stepsize in SGD is set to

$$\eta = \min\left\{\frac{1}{L}, \frac{1}{\sigma\sqrt{N}}\right\}, \quad j = 0,1,\ldots,N-1,$$

then with probability $1-\delta$

$$\min_{\ell\in[N-1]} \|\nabla F(x_\ell)\|^2 \leq \frac{2L(F(x_0)-F^*)}{N\delta} + \frac{(L+2(F(x_0)-F^*))\sigma}{\delta\sqrt{N}}.$$

We match the $O(1/\sqrt{N})$ rate of Ghadimi and Lan (2013), but without a priori knowledge of L and σ, and with a worse constant in the rate of convergence. In particular, the constant in our bound scales according to σ^3 (up to logarithmic factors in σ) while the result for SGD with well-tuned fixed step-size scales linearly with σ. The additional logarithmic factor (by Lemma 3.2) results from the AdaGrad-Norm update using the square norm of the gradient (see inequality (11) for details). The extra constant $\frac{1}{\sqrt{\delta}}$ results from the correlation between

the stepsize b_j and the gradient $\|\nabla F(x_j)\|$. We note that the recent work Li and Orabona (2019) derives an $O(1/\sqrt{N})$ rate for a variation of AdaGrad-Norm without the assumption of uniformly bounded gradient, but at the same time requires a priori knowledge of the smoothness constant $L > 0$ in setting the step-size in order to establish convergence, similar to SGD with fixed stepsize. Finally, we note that recent works (Allen-Zhu, 2017; Lei et al., 2017; Fang et al., 2018; Zhou et al., 2018) provide modified SGD algorithms with convergence rates faster than $O(1/\sqrt{N})$, albeit again requiring priori knowledge of both L and σ to establish convergence.

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 in practice.

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 Section 4.

Theorem 2.2 (AdaGrad-Norm: convergence in deterministic setting) *Suppose that $F \in \mathbb{C}^1_L$ and that $F^* = \inf_x F(x) > -\infty$. Consider AdaGrad-Norm in deterministic setting with following update,*

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

Then $\min_{j\in[N]} \|\nabla F(x_j)\|^2 \leq \varepsilon$ after

(1) $N = 1 + \lceil \frac{1}{\varepsilon}\left(\frac{4(F(x_0)-F^*)^2}{\eta^2} + \frac{2b_0(F(x_0)-F^*)}{\eta}\right)\rceil$ *if* $b_0 \geq \eta L$,

(2) $N = 1 + \lceil \frac{1}{\varepsilon}\left(2L\left(F(x_0)-F^*\right) + \left(\frac{2(F(x_0)-F^*)}{\eta} + \eta L C_{b_0}\right)^2 + (\eta L)^2(1+C_{b_0}) - b_0^2\right)\rceil$
if $b_0 < \eta L$. *Here* $C_{b_0} = 1 + 2\log\left(\frac{\eta L}{b_0}\right)$.

The convergence bound shows that, unlike gradient descent with constant stepsize η which can diverge if the stepsize $\eta \geq 2/L$, AdaGrad-Norm convergence holds for any choice of parameters b_0 and η. The critical observation is that if the initial stepsize $\frac{\eta}{b_0} > \frac{1}{L}$ is too large, the algorithm has the freedom to diverge initially, until b_j grows to a critical point (not too much larger than $L\eta$) at which point $\frac{\eta}{b_j}$ is sufficiently small that the smoothness of F forces b_j to converge to a finite number on the order of L, so that the algorithm converges at an $O(1/N)$ rate. To describe the result in Theorem 2.2, let us first review a classical result (see, for example Nesterov (1998), (1.2.13)) on the convergence rate for gradient descent with fixed stepsize.

Lemma 2.1 *Suppose that $F \in \mathbb{C}^1_L$ and that $F^* = \inf_x F(x) > -\infty$. Consider gradient descent with constant stepsize, $x_{j+1} = x_j - \frac{\nabla F(x_j)}{b}$. If $b \geq L$, then $\min_{j\in[N-1]} \|\nabla F(x_j)\|^2 \leq \varepsilon$ after at most a number of steps*

$$N = \frac{2b(F(x_0) - F^*)}{\varepsilon}.$$

Alternatively, if $b \leq \frac{L}{2}$, then convergence is not guaranteed at all – gradient descent can oscillate or diverge.

Compared to the convergence rate of gradient descent with fixed stepsize, AdaGrad-Norm in the case $b = b_0 \geq \eta L$ gives a larger constant in the rate. But in case $b = b_0 < \eta L$, gradient descent can fail to converge as soon as $b \leq \eta L/2$, while AdaGrad-Norm converges for any $b_0 > 0$, and is extremely robust to the choice of $b_0 < \eta L$ in the sense that the resulting convergence rate remains close to the optimal rate of gradient descent with fixed stepsize $1/b = 1/L$, paying a factor of $\log(\frac{\eta L}{b_0})$ and $(\eta L)^2$ in the constant. Here, the constant $(\eta L)^2$ results from the worst-cast analysis using Lemma 4.1, which assumes that the gradient $\|\nabla F(x_j)\|^2 \approx \varepsilon$ for all $j = 0, 1, \ldots$, when in reality the gradient should be much larger at first. We believe the number of iterations can be improved by a refined analysis, or by considering the setting where x_0 is drawn from an appropriate random distribution.

3. Proof of Theorem 2.1

We first introduce some important lemmas in subsection 3.1 and give the main proof of Theorem 2.1 in Subsection 3.2.

3.1 Ingredients

We first introduce several lemmas that are 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 and Lan (2013), and can be proved by considering the Taylor expansion of F around y.

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

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

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 $a_1 \geq 1$, we have*

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

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 $g(\ell) = a_\ell, 1 \leq \ell \leq T$, and $g(t) = 0$ for $t \geq T$, and replacing sums with integrals. ■

3.2 Main proof

Proof For simplicity, we write $F_j = F(x_j)$ and $\nabla F_j = \nabla F(x_j)$. By Lemma 3.1, for $j \geq 0$,

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

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

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

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

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

However, we do not have a closed form expression for $\mathbb{E}_{\xi_j}[\frac{1}{b_{j+1}}]$. We use the estimate $\frac{1}{\sqrt{b_j^2 + 2\|\nabla F_j\|^2 + 2\sigma^2}}$ as a surrogate for $\mathbb{E}_{\xi_j}[\frac{1}{b_{j+1}}]$ to proceed as we have by Jensen inequality that

$$\mathbb{E}_{\xi_j}\left[\frac{1}{b_{j+1}}\right] \geq \frac{1}{\mathbb{E}_{\xi_j}\left[b_{j+1}\right]} = \frac{1}{\mathbb{E}_{\xi_j}\left[\sqrt{b_j^2 + \|G_j\|^2}\right]} \geq \frac{1}{\sqrt{\mathbb{E}_{\xi_j}\left[b_j^2 + \|G_j\|^2\right]}} \geq \frac{1}{\sqrt{b_j^2 + 2\|\nabla F_j\|^2 + 2\sigma^2}}.$$

where the last inequality is due to $\|G_j\|^2 \leq 2\|\nabla F_j\|^2 + 2\|\nabla F_j - G_j\|^2$. Condition on $\xi_1, \ldots, \xi_{j-1}$ and take expectation with respect to ξ_j,

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

thus,

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

Now, observe the term

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

thus, applying Cauchy-Schwarz,

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

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

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

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

$$\mathbb{E}_{\xi_j}\left[\frac{\sigma\|\nabla F_j\|\|G_j\|}{b_{j+1}\sqrt{b_j^2+2\|\nabla F_j\|^2+2\sigma^2}}\right]\leq\sigma\mathbb{E}_{\xi_j}\left[\frac{\|G_j\|^2}{b_{j+1}^2}\right]+\frac{\|\nabla F_j\|^2}{4\sqrt{b_j^2+2\|\nabla F_j\|^2+2\sigma^2}}.\tag{9}$$

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

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

and, therefore, back to (6),

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

Rearranging,

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

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

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

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

$$\begin{aligned}\sum_{k=0}^{N-1}\mathbb{E}\left[\frac{\|\nabla F_k\|^2}{2\sqrt{b_k^2+2\|\nabla F_k\|^2+2\sigma^2}}\right] &\leq \frac{F_0-F^*}{\eta}+\frac{4\sigma+\eta L}{2}\mathbb{E}\sum_{k=0}^{N-1}\left[\frac{\|G_k\|^2}{b_{k+1}^2}\right] \\ &\leq \frac{F_0-F^*}{\eta}+\frac{4\sigma+\eta L}{2}\log\left(10+\frac{20N\left(\sigma^2+\gamma^2\right)}{b_0^2}\right) \quad (10)\end{aligned}$$

where the second inequality we apply Lemma (3.2) and then Jensen's inequality to bound the summation:

$$\begin{aligned}\mathbb{E}\sum_{k=0}^{N-1}\left[\frac{\|G_k\|^2}{b_{k+1}^2}\right] &\leq \mathbb{E}\left[1+\log\left(1+\sum_{k=0}^{N-1}\|G_k\|^2/b_0^2\right)\right] \\ &\leq \log\left(10+\frac{20N\left(\sigma^2+\gamma^2\right)}{b_0^2}\right). \quad (11)\end{aligned}$$

since

$$\begin{aligned}\mathbb{E}\left[b_k^2-b_{k-1}^2\right] &\leq \mathbb{E}\left[\|G_k\|^2\right] \\ &\leq 2\mathbb{E}\left[\|G_k-\nabla F_k\|^2\right]+2\mathbb{E}\left[\|\nabla F_k\|^2\right] \\ &\leq 2\sigma^2+2\gamma^2. \quad (12)\end{aligned}$$

3.2.1 Finishing the proof of the first bound in Theorem 2.1

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

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

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

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

where the last inequality is due to inequality (12). Thus (10) arrives at the inequality

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

Multiplying by $\frac{2b_0 + 4\sqrt{N}(\gamma+\sigma)}{N}$, the above inequality gives

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

where

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

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

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

where in the second step Jensen's inequality is applied to the concave function $\phi(x) = \min_k h_k(x)$.

3.2.2 Finishing the proof of the second bound in Theorem 2.1

First, observe with probability $1-\delta'$ that

$$\sum_{i=0}^{N-1} \|\nabla F_i - G_i\|^2 \leq \frac{N\sigma^2}{\delta'}.$$

For the denominator on the left hand side of the inequality 10, we let $Z = \sum_{k=0}^{N-1} \|\nabla F_k\|^2$ and so

$$\begin{aligned} b_{N-1}^2 + 2(\|\nabla F_{N-1}\|^2 + \sigma^2) =& b_0^2 + \sum_{i=0}^{N-2} \|G_i\|^2 + 2(\|\nabla F_{N-1}\|^2 + \sigma^2) \\ \leq& b_0^2 + 2\sum_{i=0}^{N-1} \|\nabla F_i\|^2 + 2\sum_{i=0}^{N-2} \|\nabla F_i - G_i\|^2 + 2\sigma^2 \\ \leq& b_0^2 + 2Z + 2N\frac{\sigma^2}{\delta'} \end{aligned}$$

Thus, we further simplify inequality (10),

$$\mathbb{E}\left[\frac{\sum_{k=0}^{N-1} \|\nabla F_k\|^2}{2\sqrt{b_{N-1}^2 + 2\|\nabla F_{N-1}\|^2 + 2\sigma^2}}\right] \leq \frac{F_0 - F^*}{\eta} + \frac{4\sigma + \eta L}{2} \log\left(10 + \frac{20N\left(\sigma^2 + \gamma^2\right)}{b_0^2}\right) \triangleq C_F$$

we have with probability $1 - \hat{\delta} - \delta'$ that

$$\frac{C_F}{\hat{\delta}} \geq \frac{\sum_{k=0}^{N-1} \|\nabla F_k\|^2}{2\sqrt{b_{N-1}^2 + 2\|\nabla F_{N-1}\|^2 + 2\sigma^2}} \geq \frac{Z}{2\sqrt{b_0^2 + 2Z + 2N\sigma^2/\delta'}}$$

That is equivalent to solve the following quadratic equation

$$Z^2 - \frac{8C_F^2}{\hat{\delta}^2} Z - \frac{4C_F^2}{\hat{\delta}^2}\left(b_0^2 + \frac{2N\sigma^2}{\delta'}\right) \leq 0$$

which gives

$$\begin{aligned} Z \leq& \frac{4C_F^2}{\hat{\delta}^2} + \sqrt{\frac{16C_F^4}{\hat{\delta}^4} + \frac{4C_F^2}{\hat{\delta}^2}\left(b_0^2 + \frac{2N\sigma^2}{\delta'}\right)} \\ \leq& \frac{8C_F^2}{\hat{\delta}^2} + \frac{2C_F}{\hat{\delta}}\left(b_0 + \frac{\sqrt{2N}\sigma}{\sqrt{\delta'}}\right) \end{aligned}$$

Let $\hat{\delta} = \delta' = \frac{\delta}{2}$. Replacing Z with $\sum_{k=0}^{N-1} \|\nabla F_k\|^2$ and dividing both side with N we have with probability $1-\delta$

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

■

4. Proof of Theorem 2.2

4.1 Lemmas

We will use the following lemma to argue that after an initial number of steps $N = \lceil \frac{(\eta L)^2 - b_0^2}{\varepsilon} \rceil + 1$, either we have already reached a point x_k such that $\|\nabla F(x_k)\|^2 \leq \varepsilon$, or else $b_N \geq \eta L$.

Lemma 4.1 *Fix $\varepsilon \in (0, 1]$ and $C > 0$. For any non-negative $a_0, a_1, \ldots,$ the dynamical system*

$$b_0 > 0; \qquad b_{j+1}^2 = b_j^2 + a_j$$

has the property that after $N = \lceil \frac{C^2 - b_0^2}{\varepsilon} \rceil + 1$ iterations, either $\min_{k=0:N-1} a_k \leq \varepsilon$, or $b_N \geq \eta L$.

Proof If $b_0 \geq \eta C$, we are done. Else $b_0 < C$. Let N be the smallest integer such that $N \geq \frac{C^2 - b_0^2}{\varepsilon}$. Suppose $b_N < C$. Then

$$C^2 > b_N^2 = b_0^2 + \sum_{k=0}^{N-1} a_k > b_0^2 + N \min_{k \in [N-1]} a_k \quad \Rightarrow \quad \min_{k \in [N-1]} a_k \leq \frac{C^2 - b_0^2}{N}$$

Hence, for $N \geq \frac{C^2 - b_0^2}{\varepsilon}$, $\min_{k \in [N-1]} a_k \leq \varepsilon$. Suppose $\min_{k \in [N-1]} a_k > \epsilon$, then from above inequalities we have $b_N > C$. ∎

The following Lemma shows that $\{F(x_k)\}_{k=0}^{\infty}$ is a bounded sequence for any value of $b_0 > 0$.

Lemma 4.2 *Suppose $F \in C_L^1$ and $F^* = \inf_x F(x) > -\infty$. Denote by $k_0 \geq 1$ the first index such that $b_{k_0} \geq \eta L$. Then for all $b_k < \eta L, k = 0, 1, \ldots, k_0 - 1$,*

$$F_{k_0 - 1} - F^* \leq F_0 - F^* + \frac{\eta^2 L}{2} \left(1 + 2 \log \left(\frac{b_{k_0 - 1}}{b_0} \right) \right) \tag{13}$$

Proof Suppose $k_0 \geq 1$ is the first index such that $b_{k_0} \geq \eta L$. By Lemma 3.1, for $j \leq k_0 - 1$,

$$F_{j+1} \leq F_j - \frac{\eta}{b_{j+1}} (1 - \frac{\eta L}{2 b_{j+1}}) \|\nabla F_j\|^2 \leq F_j + \frac{\eta^2 L}{2 b_{j+1}^2} \|\nabla F_j\|^2 \leq F_0 + \sum_{\ell=0}^{j} \frac{\eta^2 L}{2 b_{\ell+1}^2} \|\nabla F_\ell\|^2$$

$$\begin{aligned}
\Rightarrow \quad F_{k_0 - 1} - F_0 &\leq \frac{\eta^2 L}{2} \sum_{i=0}^{k_0 - 2} \frac{\|\nabla F_i\|^2}{b_{i+1}^2} \\
&\leq \frac{\eta^2 L}{2} \sum_{i=0}^{k_0 - 2} \frac{(\|\nabla F_i\| / b_0)^2}{\sum_{\ell=0}^{i} (\|\nabla F_\ell\| / b_0)^2 + 1} \\
&\leq \frac{\eta^2 L}{2} \left(1 + \log \left(1 + \sum_{\ell=0}^{k_0 - 2} \frac{\|\nabla F_\ell\|^2}{b_0^2} \right) \right) \qquad \text{by Lemma 3.2} \\
&\leq \frac{\eta^2 L}{2} \left(1 + \log \left(\frac{b_{k_0 - 1}^2}{b_0^2} \right) \right).
\end{aligned}$$

∎

4.2 Main proof

Proof By Lemma 4.1, if $\min_{k\in[N-1]} \|\nabla F(x_k)\|^2 \leq \varepsilon$ is not satisfied after $N = \lceil \frac{(\eta L)^2 - b_0^2}{\varepsilon} \rceil + 1$ steps, then there exits a first index $1 \leq k_0 \leq N$ such that $\frac{b_{k_0}}{\eta} > L$. By Lemma 3.1, for $j \geq 0$,

$$\begin{aligned} F_{k_0+j} &\leq F_{k_0+j-1} - \frac{\eta}{b_{k_0+j}}(1 - \frac{\eta L}{2b_{k_0+j}})\|\nabla F_{k_0+j-1}\|^2 \\ &\leq F_{k_0-1} - \sum_{\ell=0}^{j} \frac{\eta}{2b_{k_0+\ell}}\|\nabla F_{k_0+\ell-1}\|^2 \\ &\leq F_{k_0-1} - \frac{\eta}{2b_j}\sum_{\ell=0}^{j} \|\nabla F_{k_0+\ell-1}\|^2. \end{aligned} \tag{14}$$

Let $Z = \sum_{k=k_0-1}^{M-1} \|\nabla F_k\|^2$, it follows that

$$\frac{2\left(F_{k_0-1} - F^*\right)}{\eta} \geq \frac{2\left(F_0 - F_M\right)}{\eta} \geq \frac{\sum_{k=k_0-1}^{M-1} \|\nabla F_k\|^2}{b_M} \geq \frac{Z}{\sqrt{Z + b_{k_0-1}^2}}.$$

Solving the quadratic inequality for Z,

$$\sum_{k=k_0-1}^{M-1} \|\nabla F_k\|^2 \leq \frac{4\left(F_{k_0-1} - F^*\right)^2}{\eta^2} + \frac{2\left(F_{k_0-1} - F^*\right) b_{k_0-1}}{\eta}. \tag{15}$$

If $k_0 = 1$, the stated result holds by multiplying both side by $\frac{1}{M}$. Otherwise, $k_0 > 1$. From Lemma 4.2, we have

$$F_{k_0-1} - F^* \leq F_0 - F^* + \frac{\eta^2 L}{2}\left(1 + 2\log\left(\frac{\eta L}{b_0}\right)\right).$$

Replacing $F_{k_0-1} - F^*$ in (15) by above bound, we have

$$\begin{aligned} &\sum_{k=k_0-1}^{M-1} \|\nabla F_k\|^2 \\ &\leq \underbrace{\left(\frac{2\left(F_0 - F^*\right)}{\eta} + \eta L\left(1 + 2\log\left(\eta L/b_0\right)\right)\right)^2 + 2L\left(F_0 - F^*\right) + (\eta L)^2\left(1 + 2\log\left(\frac{\eta L}{b_0}\right)\right)}_{C_M} \end{aligned}$$

Thus, we are assured that

$$\min_{k=0:N+M-1} \|\nabla F_k\|^2 \leq \varepsilon$$

where $N \leq \frac{L^2 - b_0^2}{\varepsilon}$ and $M = \frac{C_M}{\varepsilon}$. ■

5. Numerical experiments

With guaranteed convergence of AdaGrad-Norm and its robustness to the parameters η and b_0, we perform experiments on several data sets ranging from simple linear regression over Gaussian data to neural network architectures on state-of-the-art (SOTA) image data sets including ImageNet. These experiments are not about outperforming the strong baseline of well-tuned SGD, but to further strengthen the theoretical finding that the convergence of AdaGrad-norm requires less hyper-parameter tuning while maintaining a comparable performance as the well-tuned SGD methods.

5.1 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 $\frac{650}{b_0}$, (b) SGD-DecaySqrt: decaying stepsize $\eta_j = \frac{650}{b_0\sqrt{j}}$, and (c) AdaGrad-Coodinate: update the d parameters $b_j(k), k = 1, 2, \ldots, d$ at each iteration j, one for each coordinate of $x_j \in \mathbb{R}^d$. Figure 1 plots $\|A^T\left(Ax_j - y\right)\|/m$ (GradNorm) and the effective learning rates at iterations 10, 2000, and 5000, and as a function of b_0, for each of the four methods. The effective learning rates are $\frac{650}{b_j}$ (AdaGrad-Norm), $\frac{650}{b_0}$(SGD-Constant), $\frac{650}{b_0\sqrt{j}}$(SGD-DecaySqrt), and the median of $\{b_j(\ell)\}_{\ell=1}^d$ (AdaGrad-Coordinate).

We can see in Figure 1 how AdaGrad-Norm and AdaGrad-Coordinate auto-tune the learning rate adaptively 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 and AdaGrad-Coordinate still converge with good speed while SGD-Constant and SGD-DecaySqrt diverge. When b_0 is initialized too large (stepsize too small), surprisingly AdaGrad-Norm and AdaGrad-Coordinate converge 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). Comparing AdaGrad-Coordinate and AdaGrad-Norm, AdaGrad-Norm is

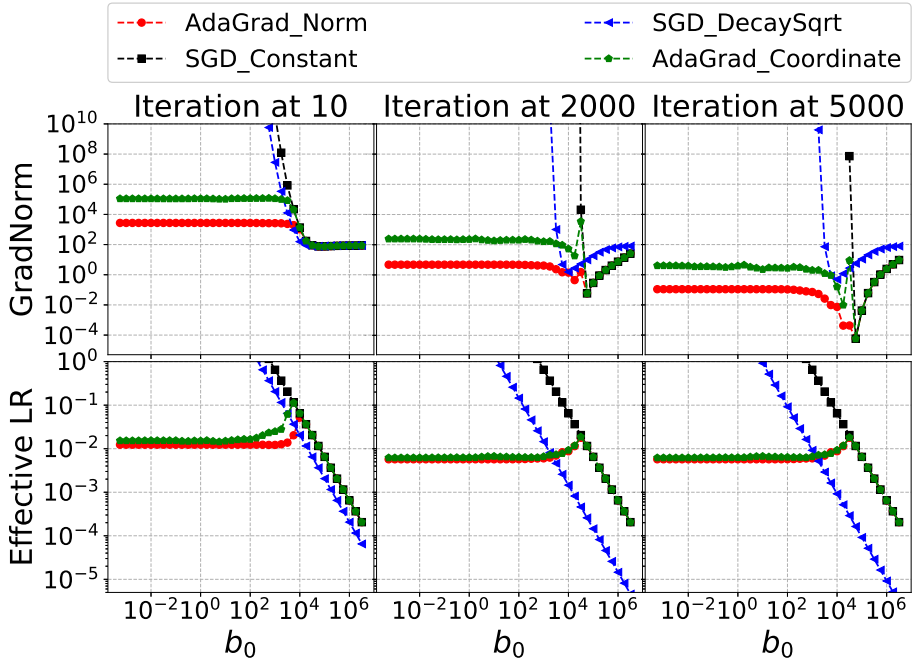

Figure 1: Gaussian Data – Stochastic Setting. The top 3 figures plot the square of the gradient norm for linear regression, $\|A^T(Ax_j - y)\|/m$, w.r.t. b_0, at iterations 10, 2000 and 5000 (see title) respectively. The bottom 3 figures plot the corresponding effective learning rates (median of $\{b_j(\ell)\}_{\ell=1}^d$ for AdaGrad-Coordinate), w.r.t. b_0, at iteration 10, 2000 and 5000 respectively (see title).

more robust to the initialization b_0 but is not better than AdaGrad-Coordinate when the initialization b_0 is close to the optimal value of L.

Figure 2 explores the batch gradient descent setting, when there is no variance $\sigma = 0$ (i.e., using the whole data sample for one iteration). The experimental setup in Figure 2 is the same as Figure 1 except for the sample size m of each iteration. Since the line-search method (GD-LineSearch) is one of the most important algorithms in deterministic gradient descent for adaptively choosing the step-size at each iteration, we also compare to this method – see Algorithm 2 in the appendix for our particular implementation of Line-Search. We see that the behavior of the four methods, AdaGrad-Norm, AdaGrad-Coordinate, GD-Constant, and GD-DecaySqrt, are very similar to the stochastic setting, albeit AdaGrad-Coordinate here is worse than in the stochastic setting. Among the five methods in the plot, GD-LineSearch performs the best but with significantly longer computational time, which is not practical in large-scale machine learning problems.

5.2 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 –

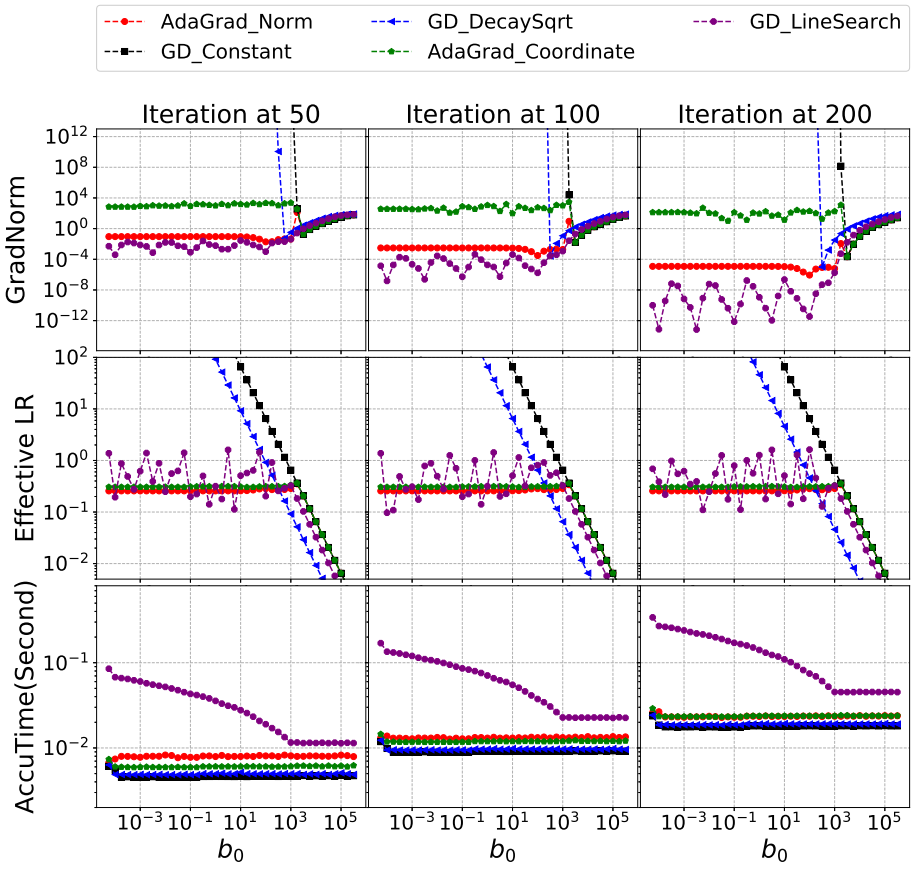

Figure 2: Gaussian Data - Batch Setting. The y-axis and x-axis in the top and middle 3 figures are the same as in Figure 1. The bottom 3 figures plot the accumulated computational time (AccuTime) up to iteration 50, 100 and 200 (see title), as a function of b_0.

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.[3] We did not find it practical to compute the norm of the gradient for the entire neural network during back-propagation. Instead, we adapt 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.[4]

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 (LogReg), a multilayer network

3. The code we used is originally from `https://github.com/pytorch/examples/tree/master/imagenet`
4. `https://github.com/xwuShirley/pytorch/blob/master/torch/optim/adagradnorm.py`

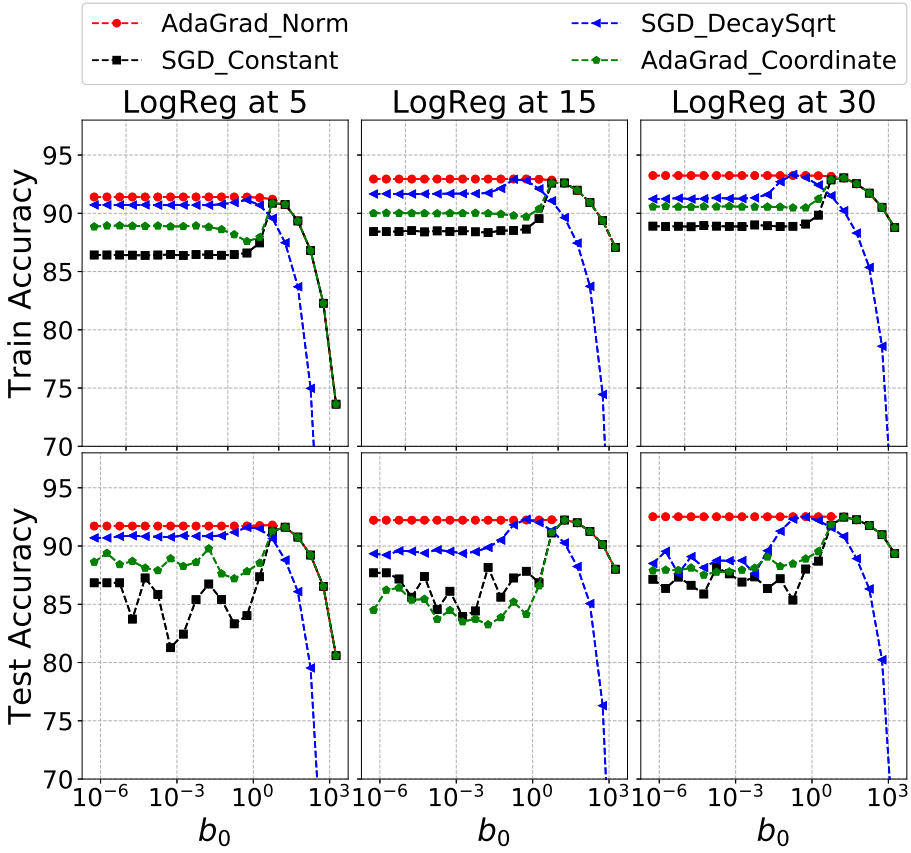

Figure 3: MNIST. In each plot, the y-axis is the train or test accuracy and the x-axis is b_0. The 6 plots are for logistic regression (LogReg) with average at epoch 1-5, 11-15 and 26-30. The title is the last epoch of the average. Note green and red curves overlap when b_0 belongs to $[10, \infty)$

with two fully connected layers (FulConn2) with 100 hidden units and ReLU activations, and a convolutional neural network (see Table 2 in the appendix for details). For CIFAR10, our model is ResNet-18 (He et al., 2016). For both data sets, we use 256 images per iteration (2 GPUs with 128 images/GPU, 234 iterations per epoch for MNIST and 196 iterations per epoch for CIFAR10). For ImagetNet, we use ResNet-50 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.

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

Experimental Details In order to make the setting match our assumptions, we make several changes, which are not practically meaningful scenarios but serve only for corroborating our theorems.

For the experiment in MNIST, we do not use bias, regularization (zero weight decay), dropout, momentum, batch normalization (Ioffe and Szegedy, 2015), or any other added features that help achieving SOTA performance (see Figure 3 and Figure 4). However, the architecture of ResNet by default is built with the celebrated batch normalization (Batch-Norm) method as important layers. Batch-Norm accomplishes the auto-tuning property by normalizing the means and variances of mini-batches in a particular way during the forward-propagation, and in return is back-propagated with projection steps. This projection phenomenon is highlighted in weight normalization (Salimans and Kingma, 2016; Wu et al., 2018). Thus, in the ResNet-18 experiment on CIFAR10, we are particularly interested in how Batch-Norm interacts with the auto-tuning property of AdaGrad-Norm. We disable the learnable scale and shift parameters in the Batch-Norm layers [5] and compare the default setup in ResNet (Goyal et al., 2017). The resulted plots are located in Figure 4 (bottom left and bottom right). In the ResNet-50 experiment on ImageNet, we also depart from the standard set-up by initializing the weights of the last fully connected layer with i.i.d. Gaussian samples with mean zero and variance 0.03125. Note that the default initialization for the last fully-connected layer of ResNet50 is an i.i.d. Gaussian distribution with mean zero and variance of 0.01. Instead, we use variance 0.03125 in that the AdaGrad-Norm algorithm takes the norm of the gradient. The initialization of Gaussian distribution with higher variance results in larger accumulative gradient norms, which is likely to make AdaGrad-Norm robust to small initialization of b_0. To some extent, AdaGrad-Norm could be sensitive to the model's initialization. But how much sensitive the AdaGrad-Norm, or more generally the adaptive gradient methods, to the initialization of the model could be a potential future direction.

For all experiments, same initialized vector x_0 is used for the same model so as to eliminate the effect of random initialization in weight vectors. We set $\eta = 1$ in all AdaGrad implementations, noting that in all these problems we know that $F^* = 0$ and we measure that $F(x_0)$ is between 1 and 10. Indeed, we approximate the loss using a sample of 256 images to be $\frac{1}{256}\sum_{i=1}^{256} f_i(x_0)$: 2.4129 for logistic regression, 2.305 for two-layer fully connected model, 2.301 for convolution neural network, 2.3848 for ResNet-18 with disable learnable parameter in Batch-Norm, 2.3459 for ResNet-18 with default Batch-Norm, and 7.704 for ResNet-50. We vary the initialization b_0 while fixing all other parameters and plot the training accuracy and testing accuracy after different numbers of epochs. We compare AdaGrad-Norm with initial parameter b_0 to (a) SGD-Constant: fixed stepsize $\frac{1}{b_0}$, (b) SGD-DecaySqrt: decaying stepsize $\eta_j = \frac{1}{b_0\sqrt{j}}$ (c) AdaGrad-Coordinate: a vector of per-coefficient stepsizes. [6]

Observations and Discussion In all experiments shown in Figures 3, 4, and 5, we fix b_0 and compare the accuracy for the four algorithms; the convergence of AdaGrad-Norm is much better even for small initial values b_0, and shows much stronger robustness than the alternatives. In particular, Figures 3 and 4 show that the AdaGrad-Norm's accuracy is extremely robust (as good as the best performance) to the choice of b_0. At the same time, the SGD methods and AdaGrad-Coordinate are highly sensitive. For Figure 5, the range of

5. Set *nn.BatchNorm2d(planes,affine=False)*
6. We use *torch.optim.adagrad*

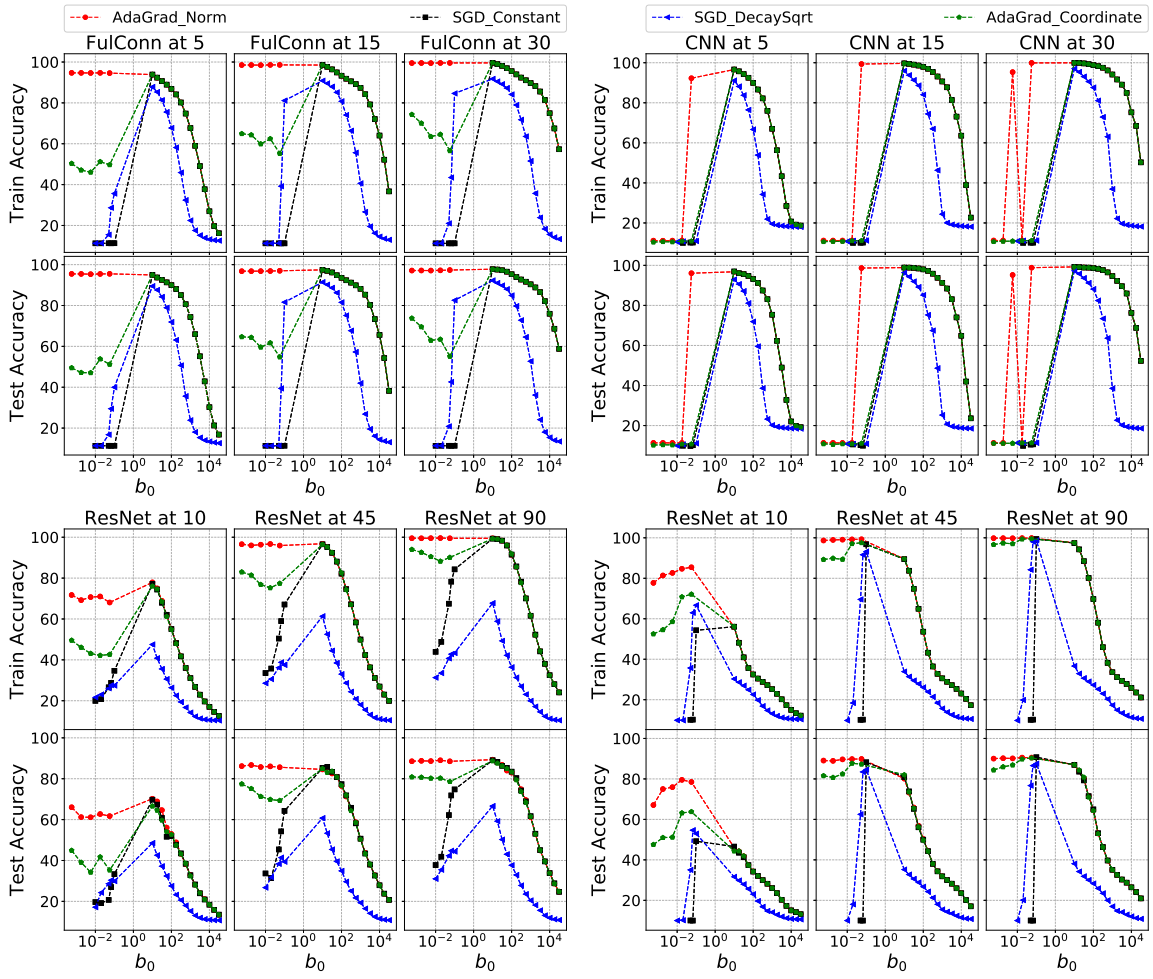

Figure 4: In each plot, the y-axis is the train or test accuracy and the x-axis is b_0. Top left 6 plots are for MNIST using the two-layer fully connected network (ReLU activation). Top right 6 plots are for MNIST using convolution neural network (CNN). Bottom left 6 plots are for CIFAR10 using ResNet-18 with disabling learnable parameter in Batch-Norm. Bottom right 6 plots are for CIFAR10 using ResNet-18 with default Batch-Norm. The points in the (top) bottom plot are the average of epoch (1-5) 6-10, epoch (11-15) 41-45 or epoch (26-30) 86-90. The title is the last epoch of the average. Note green, red and black curves overlap when b_0 belongs to $[10, \infty)$. Better read on screen.

parameters b_0 for which AdaGrad-Norm attains its best performance is also larger than the corresponding range for SGD-Constant and AdaGrad-Coordinate but sub-optimal for small values of b_0. It is likely to indicate that for ImageNet training, AdaGrad-Norm does not remove the need to tune b_0 but makes the hyper-parameter search for b_0 easier. Note that the best test accuracy in Figure 5 is substantially lower than numbers in the literature, where

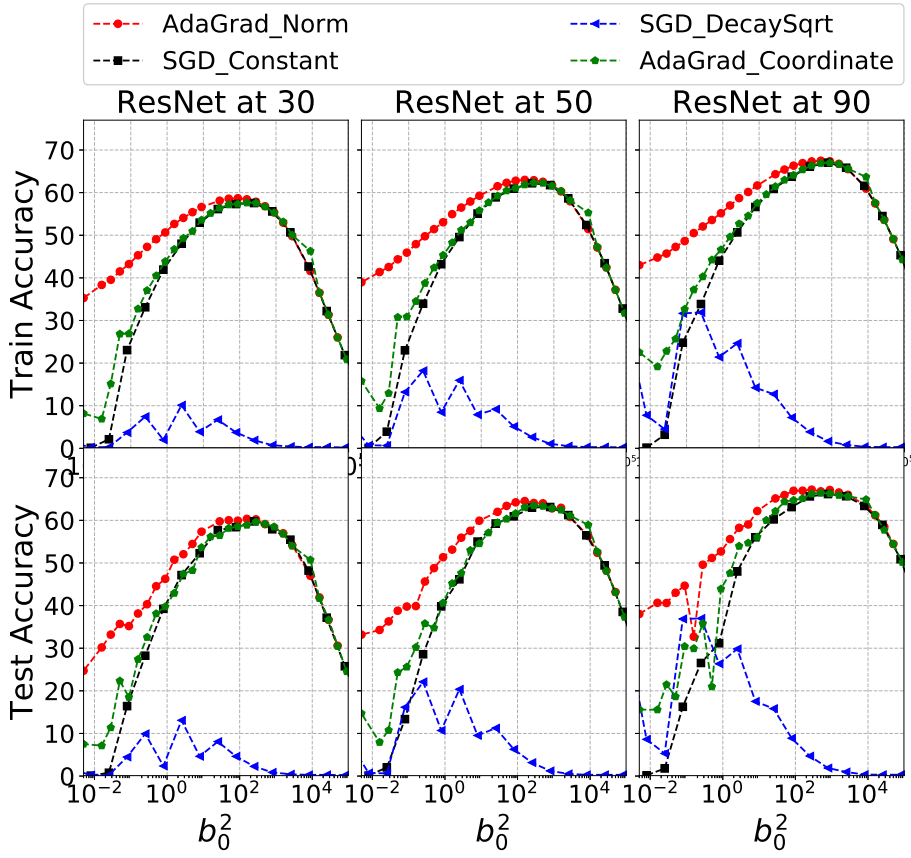

Figure 5: ImageNet trained with model ResNet-50. The y-axis is the average train or test accuracy at epoch 26-30, 46-50, 86-90 w.r.t. b_0^2. Note no momentum is used in the training. See **Experimental Details**. Note green, red and black curves overlap when b_0 belongs to $[10, \infty)$.

optimizers for ResNet-50 on ImageNet attain test accuracy around 76% (Goyal et al., 2017), about 10% better than the best result in Figure 5. This is mainly because (a) we do not apply momentum methods, and perhaps more critically (b) both SGD and AdaGrad-Norm do not use the default decaying scheduler for η as in Goyal et al. (2017). Instead, we use a constant rate $\eta = 1$. Our purpose is not to achieve the comparable state-of-the-art results but mainly to verify that AdaGrad-Norm is less sensitive to hyper-parameter and requires less hyper-parameter tuning.

Similar to the Synthetic Data, when b_0 is initialized in the range of well-tuned stepsizes, AdaGrad-Norm gives almost the same accuracy as SGD with constant stepsize; when b_0 is initialized too small, AdaGrad-Norm still converges with good speed (except for CNN in MNIST), while SGDs do not. The divergence of AdaGrad-Norm with small b_0 for CNN in MNIST (Figure 4, top right) can be possibly explained by the unboundedness of gradient norm in the four-layer CNN model. In contrast, the 18-layer or 50-layer ResNet model is

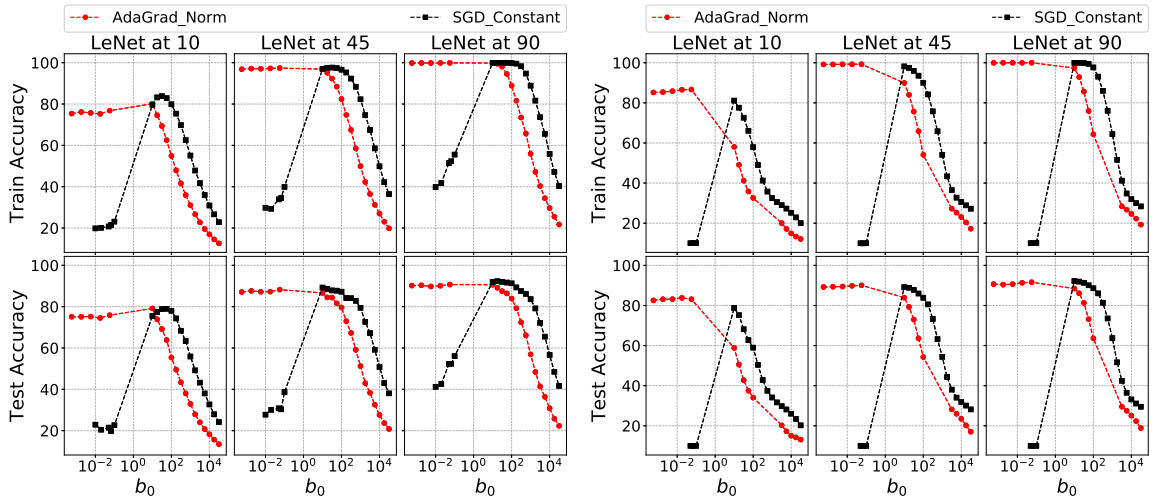

Figure 6: The performance of SGD and AdaGrad-Norm in presence of momentum (see Algorithm 3). In each plot, the y-axis is train or test accuracy and x-axis is b_0. Left 6 plots are for CIFAR10 using ResNet-18 with disabling learnable parameter in Batch-Norm. Right 6 plots are for CIFAR10 using ResNet-18 with default Batch-Norm. The points in the plot are the average of epoch 6-10, epoch 41-45 and epoch 86-90, respectively. The title is the last epoch of the average. Better read on screen.

very robust to all range of b_0 in experiments (Figure 4, bottom), which is due to Batch-Norm that we further discuss in the next paragraph.

We are interested in the experiments of Batch-Norm by default and Batch-Norm without learnable parameters because we want to understand how AdaGrad-Norm interacts with models that already have the built-in feature of auto-tuning stepsize such as Batch-Norm. First, comparing the outcomes of Batch-Norm with the default setting (Figure 4, bottom right) and without learnable parameters (Figure 4, bottom left), we see the learnable parameters (scales and shifts) in Batch-Norm can be very helpful in accelerating the training. Surprisingly, the best stepsize in Batch-Norm with default for SGD-Constant is at $b_0 = 0.1$ (i.e., $\eta = 10$). While the learnable parameters are more beneficial to AdaGrad-Coordinate, AdaGrad-Norm seems to be affected less. Overall, combining the two auto-tuning methods (AdaGrad-Norm and Batch-Norm) give good performance.

At last, we add momentum to the stochastic gradient descent methods as empirical evidence to showcase the robustness of adaptive methods with momentum shown in Figure 6. Since SGD with 0.9 momentum is commonly used, we also set 0.9 momentum for our implementation of AdaGrad-Norm. See Algorithm 3 in the appendix for details. The results (Figure 6) show that AdaGrad-Norm with momentum is highly robust to initialization while SGD with momentum is not. SGD with momentum does better than AdaGrad-Norm when the initialization b_0 is greater than the Lipschitz smoothness constant. When b_0 is smaller

than the Lipschitz smoothness constant, AdaGrad-Norm performs as well as SGD with the best stepsize (0.1).

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 owe particular gratitude to the reviewers and the editor for their suggestions and comments that significantly improved the paper.

References

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

N. Agarwal, Z. Allen-Zhu, B. Bullins, and T. Hazan, E.and Ma. Finding approximate local minima faster than gradient descent. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 1195–1199, 2017. ISBN 978-1-4503-4528-6.

Z. Allen-Zhu. Natasha: Faster non-convex stochastic optimization via strongly non-convex parameter. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 89–97. JMLR. org, 2017.

Z. Allen-Zhu. Natasha 2: Faster non-convex optimization than sgd. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2675–2686. 2018.

Z. Allen-Zhu and Y. Yang. Improved svrg for non-strongly-convex or sum-of-non-convex objectives. In *International conference on machine learning*, pages 1080–1089, 2016.

J. Barzilai and J. Borwein. Two-point step size gradient method. *IMA Journal of Numerical Analysis*, 8:141–148, 1988.

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Reviews*, 60(2):223–311, 2018.

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

Y. Carmon, J. Duchi, O. Hinder, and A Sidford. "convex until proven guilty": Dimension-free acceleration of gradient descent on non-convex functions. In *International Conference on Machine Learning*, pages 654–663. PMLR, 2017.

Y. Carmon, J. Duchi, O. Hinder, and A. Sidford. Accelerated methods for nonconvex optimization. *SIAM Journal on Optimization*, 28(2):1751–1772, 2018.

Y. Carmon, J. Duchi, O. Hinder, and A. Sidford. Lower bounds for finding stationary points i. *Mathematical Programming*, pages 1–50, 2019.

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

A. Cutkosky and K. Boahen. Online learning without prior information. *Proceedings of Machine Learning Research vol*, 65:1–35, 2017.

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

J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

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

C. Fang, C. J. Li, Z. Lin, and T. Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 689–699. 2018.

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

S. Ghadimi and G. Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1-2):59–99, 2016.

P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, and K. Jia, Y.and He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

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

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

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

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

J. Lafond, N. Vasilache, and L. Bottou. Diagonal rescaling for neural networks. Technical report, arXiV:1705.09319, 2017.

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

L. Lei, Cheng J., J. Chen, and M. Jordan. Non-convex finite-sum optimization via scsg methods. In *Advances in Neural Information Processing Systems*, pages 2348–2358, 2017.

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

X. Li and F. Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 983–992. PMLR, 2019.

B. McMahan and M. Streeter. Adaptive bound optimization for online convex optimization. *Conference on Learning Theory*, page 244, 2010.

M. C. Mukkamala and M. Hein. Variants of RMSProp and Adagrad with logarithmic regret bounds. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2545–2553, 2017.

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

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

Y. Nesterov. Introductory lectures on convex programming volume i: Basic course. 1998.

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

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

S. J. Reddi, S. Sra, B. Póczos, and A. Smola. Fast incremental method for smooth nonconvex optimization. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 1971–1977. IEEE, 2016.

S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

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

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

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

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

A. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4148–4158, 2017.

S. Wright and J. Nocedal. *Numerical Optimization*. Springer New York, New York, NY, 2006. ISBN 978-0-387-40065-5.

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

Yi Xu, Rong Jin, and Tianbao Yang. First-order stochastic algorithms for escaping from saddle points in almost linear time. In *Advances in Neural Information Processing Systems*, pages 5530–5540, 2018.

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

D. Zhou, P. Xu, and Q. Gu. Stochastic nested variance reduced gradient descent for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 3925–3936, 2018.

Appendix A. Tables

Table 1: Statistics of data sets. DIM is the dimension of a sample

Dataset	Train	Test	Classes	Dim
MNIST	60,000	10,000	10	28×28
CIFAR-10	50,000	10,000	10	32×32
ImageNet	1,281,167	50,000	1000	Various

Table 2: Architecture for four-layer convolution neural network (CNN)

Layer type	Channels	Out Dimension
5×5 conv relu	20	24
2×2 max pool, str.2	20	12
5×5 conv relu	50	8
2×2 max pool, str.2	50	4
FC relu	N/A	500
FC relu	N/A	10

Appendix B. Implementing Algorithm 1 in a neural network

In this section, we give the details for implementing our algorithm in a neural network. In the standard neural network architecture, the computation of each neuron consists of an elementwise nonlinearity of a linear transform of input features or output of previous layer:

$$y = \phi(\langle w, x\rangle + b), \tag{16}$$

where w is the d-dimensional weight vector, b is a scalar bias term, x,y are respectively a d-dimensional vector of input features (or output of previous layer) and the output of current neuron, $\phi(\cdot)$ denotes an element-wise nonlinearity.

For fully connected layer, the stochastic gradient G in Algorithm 1 represents the gradient of the current neuron (see the green curve, Figure 7). Thus, when implementing our algorithm in PyTorch, AdaGrad Norm is one learning rate associated to one neuron for fully connected layer, while SGD has one learning rate for all neurons.

For convolution layer, the stochastic gradient G in Algorithms 1 represents the gradient of each channel in the neuron. For instance, there are 6 learning rates for the first layer in the LeNet architecture (Table 1). Thus, AdaGrad-Norm is one learning rate associated to one channel.

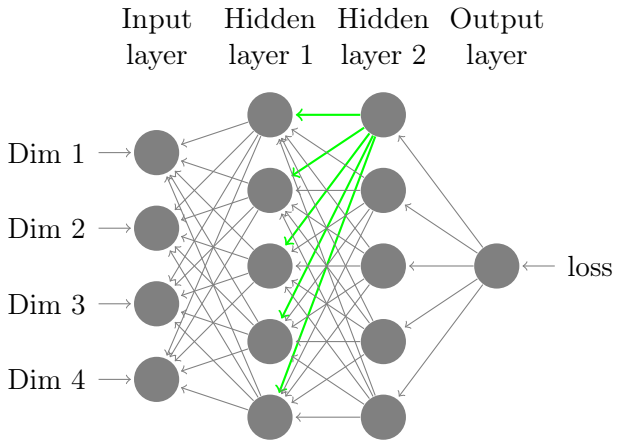

Figure 7: An example of backpropagation of two hidden layers. Green edges represent the stochastic gradient G in Algorithm 1 .

Algorithm 3 AdaGrad-Norm with momentum in PyTorch

Input: Initialize $x_0 \in \mathbb{R}^d, b_0 > 0, v_0 \leftarrow 0, j \leftarrow 0, \beta \leftarrow 0.9$, and the total iterations N.
for $j = 0, 1, \ldots, N$ **do**
  Generate ξ_j and $G_j = G(x_j, \xi_j)$
  $v_{j+1} \leftarrow \beta v_j + (1-\beta) G_j$
  $x_{j+1} \leftarrow x_j - \frac{v_{j+1}}{b_{j+1}}$ with $b_{j+1}^2 \leftarrow b_j^2 + \|G_j\|^2$
end for

Algorithm 2 Gradient Descent with Line Search Method

function LINE-SEARCH($x, b_0, \nabla F(x)$)
  $x_{new} \leftarrow x - \frac{1}{b_0}\nabla F(x)$
  while $F(x_{new}) > F(x) - \frac{b_0}{2}\|\nabla F(x)\|^2$ **do**
    $b_0 \leftarrow 2b_0$
    $x_{new} \leftarrow x - \frac{1}{b_0}\nabla F(x)$
  end while
  return x_{new}
end function