Neural Networks 152 (2022) 499–509

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

A multivariate adaptive gradient algorithm with reduced tuning efforts[☆,☆☆,★,★★]

Samer Saab Jr[a,*], Khaled Saab[b], Shashi Phoha[c], Minghui Zhu[a], Asok Ray[d]

[a] School of Electrical Engineering and Computer Engineering, The Pennsylvania State University, State College, PA, 16802, USA
[b] Department of Electrical Engineering, Stanford University, Stanford, CA, 94305, USA
[c] Applied Research Laboratory, The Pennsylvania State University, State College, PA, 16802, USA
[d] Department of Mechanical Engineering and Mathematics, The Pennsylvania State University, State College, PA, 16802, USA

ARTICLE INFO

Article history:
Received 2 November 2021
Received in revised form 14 March 2022
Accepted 15 May 2022
Available online 21 May 2022

Keywords:
Deep learning
Gradient descent optimization
Adaptive learning rate

ABSTRACT

Large neural networks usually perform well for executing machine learning tasks. However, models that achieve state-of-the-art performance involve arbitrarily large number of parameters and therefore their training is very expensive. It is thus desired to implement methods with small per-iteration costs, fast convergence rates, and reduced tuning. This paper proposes a multivariate adaptive gradient descent method that meets the above attributes. The proposed method updates every element of the model parameters separately in a computationally efficient manner using an adaptive vector-form learning rate, resulting in low per-iteration cost. The adaptive learning rate computes the absolute difference of current and previous model parameters over the difference in subgradients of current and previous state estimates. In the deterministic setting, we show that the cost function value converges at a linear rate for smooth and strongly convex cost functions. Whereas in both the deterministic and stochastic setting, we show that the gradient converges in expectation at the order of $\mathcal{O}(1/\sqrt{k})$ for a non-convex cost function with Lipschitz continuous gradient. In addition, we show that after T iterates, the cost function of the last iterate scales as $\mathcal{O}(\log(T)/T)$ for non-smooth strongly convex cost functions. Effectiveness of the proposed method is validated on convex functions, smooth non-convex function, non-smooth convex function, and four image classification data sets, whilst showing that its execution requires hardly any tuning unlike existing popular optimizers that entail relatively large tuning efforts. Our empirical results show that our proposed algorithm provides the best overall performance when comparing it to tuned state-of-the-art optimizers.

1. Introduction

First-order optimization methods have proven to be effective in large-scale systems, such as deep neural networks, due to their modest computational costs. For example, stochastic gradient descent (SGD) methods (Robbins & Monro, 1951) perform well across many applications in a cost-effective manner. However, SGD scales the gradient uniformly across all model parameters, which may lead to poor performance (Luo, Xiong, Liu, & Sun, 2019). In addition, tuning the hyper-parameters, e.g., step-size, of such non-adaptive optimizers tends to be expensive, which need to be fine-tuned for maximal performance (Probst, Boulesteix, & Bischl, 2019). Various ways of choosing the step-size for SGD have been proposed, which under certain assumptions have convergence guarantees. Constant step-sizes have shown to guarantee convergence to neighborhoods of local optimum, whereas decreasing step-sizes have shown to reach the exact optimum for smooth functions (Ghadimi & Lan, 2013; Gower et al., 2019). However, choosing a near-optimal step-size and decreasing step-size strategy requires a generous amount of tuning, which may be too expensive for models with a large number of parameters.

Adaptive methods have been successfully applied in several machine learning applications (Li & Orabona, 2019) and when training deep neural networks (Duchi, Hazan, & Singer, 2011; Kingma & Ba, 2014; Loizou, Vaswani, Laradji, & Lacoste-Julien, 2020; Vaswani et al., 2019). Despite these application-specific successes, adaptive methods suffer from two limitations. First, adaptive methods, such as Adam, can be non-convergent even

☆ The Walker Fellowship from the Applied Research Laboratory at the Pennsylvania State University.
☆☆ Wu Tsai Neuroscience Stanford Interdisciplinary Graduate Fellowship (SIGF).
★ The U.S. Air Force Office of Scientific Research (AFOSR) under Grant No. FA9550-15-1-040 in the area of dynamic data-driven application systems (DDDAS).
★★ NSF CAREER award ECCS-1846706.
* Corresponding author.
E-mail addresses: samer.saab.1st@gmail.com (S. Saab Jr), ksaab@stanford.edu (K. Saab), sxp26@psu.edu (S. Phoha), muz16@psu.edu (M. Zhu), axr2@psu.edu (A. Ray).

https://doi.org/10.1016/j.neunet.2022.05.016
0893-6080/© 2022 Elsevier Ltd. All rights reserved.

in the convex setting (Luo et al., 2019). The restrictive conditions that guarantee convergence of popular adaptive methods, see, e.g., Zou, Shen, Jie, Zhang, and Liu (2019), may limit their application domains and may be difficult to justify in practice. Second, the existing adaptive methods require tuning of at least one hyper parameter, which does not make them more tuning efficient than SGD.

This paper aims to address the aforementioned two limitations of existing adaptive methods. Inspired by the success stories of adaptive methods, and the robustness of gradient descent methods, we propose a novel multivariate adaptive gradient descent method that yields global convergence for a class of optimization problems with competitive empirical performance when compared to the state-of-the art optimizers. More specifically, we analytically and experimentally show that our proposed method provides convergence guarantees on non-convex cost functions without restricting that the gradient is bounded. Furthermore, it can perform comparably to the methods compared in Wilson, Roelofs, Stern, Srebro, and Recht (2017) while fixing its hyper-parameter for all image classification tasks and using a unity step-size, and no hyper-parameter tuning whatsoever on common challenging functions. Therefore, to the best of the authors' knowledge, our method requires less tuning efforts than all aforementioned methods including SGD and adaptive methods.

1.1. Related work

To guarantee convergence, the selection of the step-size is one of the most essential steps in gradient descent methods, which has been investigated for decades (Robbins & Monro, 1951). In Bertsekas (1997) it was shown that under mild assumptions, global convergence can be achieved when the step-size is square summable, but not summable, which was later replaced by $\alpha_k \to 0$ when the gradients are noisy (Gaivoronski, 1994). For example in Needell, Srebro, and Ward (2016), assuming a convex and continuously differentiable cost function with L-Lipschitz gradients, a bound that is inversely proportional to L is derived for the step-size of SGD to guarantee that the error converge linearly to a non-zero term related to the chosen step-size. In Needell and Ward (2016) a step-size relating to both of the extreme Lipschitz bounds of the gradient is proposed, which also guarantees linear convergence to a region around the solution. In any case, a constant step-size guarantees convergence to only a local neighborhood around the solution. Whereas to reach the exact optimum, a decreasing step-size is required (Ghadimi & Lan, 2013; Gower et al., 2019; Karimi, Nutini, & Schmidt, 2016; Lin & Zhou, 2017). For example, it is shown (Bach & Moulines, 2011) that convergence can be guaranteed when the step-size for SGD is proportional to the inverse of the iteration number, where the convergence rate changes according to the interval that the step-size is chosen within. To go a step further, the work in Lei, Hu, Li, and Tang (2019) establishes convergence rates for decreasing step-sizes under more general settings, where the cost function is assumed non-convex with unbounded gradients.

The work in Karimi et al. (2016) compares the convergence rates of a constant step-size strategy to a decreasing step-size. They report that the constant step-size strategy converges the error to some non-zero term that is proportional to the step-size, whereas the decreasing steps-size strategy converges the error to zero. They do state however that for a desired fixed accuracy, it is recommended to use a constant step-size, and decreasing it whenever the accuracy stalls. This is since the $\mathcal{O}(1/k)$ convergence rate of a decreasing step-size strategy matches that of SGD under the assumption of strong convexity. It remains however that adaptive methods decrease the amount of attention or tweaking required during training, and are especially beneficial in machine learning applications (Li & Orabona, 2019).

We introduce some of the most frequently used adaptive optimizers in everyday machine learning applications. AdaGrad adjusts its step size at every iteration by decreasing or increasing the step size for parameters that are connected to frequently or infrequently occurring features, respectively (Duchi et al., 2011). The step size is thus adapted according to the sum of accumulated square gradients, which may be harmful as it aggressively decays the learning rate. Adadelta reduces this aggressive decay in learning rate by restricting the window of past gradient information that is accumulated to some fixed size (Zeiler, 2012). RMSProp however automatically adjusts the learning rate for each parameter using a moving average of the squared gradient, where the gradient is normalized using the magnitude of recent gradient descents. A norm version of the RMSProp algorithm with penalty is proposed for the general nonconvex setting (Xu, Zhang, Zhang, & Mandic, 2021). In contrast, Adam uses additional gradient information by utilizing the first and second moments of the gradient. Unlike in AdaGrad, the learning rate in Adam is scaled according to the exponential moving average of the gradients and square gradients, which was shown to work better than other adaptive methods (Kingma & Ba, 2014). However, even adaptive methods, such as Adam, can be non-convergent in the convex setting (Luo et al., 2019). More recently, Dubey et al. (2019) propose the diffGrad, which adjusts its step-size based on the difference between the present and the immediate past gradient. Therefore, updates are smaller in low gradient changing regions and vice versa. It is worthwhile noting that in some applications where adaptive algorithms achieve better training accuracy than the non-adaptive ones, in the long run, better accuracy on the test data is obtained using the non-adaptive algorithms (Alecsa, Pinţa, & Boros, 2020).

The work in Khan et al. (2018) proposes a method that introduces perturbations to the network parameters during gradient evaluations, as well as estimate uncertainty parameters, which can be implemented in Adam. However, this algorithm adds an additional *precision* hyper-parameter that needs to be approximated (e.g. using Bayesian optimization) prior, as well as require memory to store the uncertainty estimates. The last-iterate convergence of constrained convex functions for an adaptive heavy-ball (HB) method is studied in Tao, Long, Wu, and Tao (2021), where the step-size is updated using an exponential moving average. Using $\beta_{1t} = \frac{t}{t+2}$ and $\beta_{2t} = 1 - \frac{\gamma}{t}$, where t is the epoch number, they could achieve accelerated convergence. Specifically, their method attains a convergence rate of $\mathcal{O}(\frac{1}{\sqrt{t}})$ as oppose to $\mathcal{O}(\frac{\log t}{\sqrt{t}})$ of SGD. However it remains that two hyper-parameters (α and γ) need to be properly selected for the best performance.

1.2. Contributions

We make the following contributions:

- *Development of a novel multivariate adaptive gradient descent method*, named MADAGRAD, whose learning rate is adaptively adjusted for each element of its parameter. The learning rate computation is based on the element-by-element ratio of the difference in current and past parameters to the difference in current and past subgradients of the cost function, and thus does not pose any significant increase in computational complexity.
- *Smooth and strongly convex*: In the non-stochastic setting, we show that the cost function converges at a linear rate for smooth and μ-strongly convex cost functions.
- *Smooth and non-convex*: Under both deterministic and stochastic settings, we show that the gradient converges in expectation at the order of $\mathcal{O}(1/\sqrt{k})$ for a non-convex cost function with Lipschitz continuous gradient. Of note, under both settings, we do not assume boundedness of the subgradients.

- *Non-smooth and strongly-convex*: In the stochastic setting, we show that after T iterates, the cost function of the last iterate scales as $\mathcal{O}(\log(T)/T)$ for non-smooth strongly convex cost functions with bounded subgradients.
- *Tuning reduction*: We show that for the non-stochastic setting, convergence can be achieved for smooth convex or non-convex cost functions while setting $\gamma_k = 1, \forall k$ and $\chi_k = \phi_k$. That is, absolutely no tuning is needed.

In addition, we empirically validate the proposed method on smooth and strongly convex function, convex with non-twice-differentiable gradients, a smooth non-convex function, non-smooth convex function, and four image classification tasks using deep neural networks. We show that our proposed method displays superior convergence properties on all toy examples. We test our algorithm on MNIST, QMNIST, and CIFAR-10, and CIFAR-100. We show that MADAGRAD with reduced tuning efforts yields the best overall performance for deep neural networks when comparing it to the state-of-the-art *tuned* optimizers such as NAG, diffGrad, Adam and SGDm.

2. Proposed adaptive method

We consider the following unconstrained optimization problem

$$\min_{x\in\mathbb{R}^d} f(x), \tag{1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a differentiable function that is bounded from below, and assume that (1) has a solution x^*, where we denote its optimal value by $f(x^*)$. For constrained optimization problems, the interested readers are referred to Li and Bian (2021) and the references therein.

We denote by $\mathscr{G}_k \triangleq \mathscr{G}(x_k; \zeta_k)$ a stochastic subgradient of $f(x)$ at x_k depending on a random variable ζ_k, indexed by fixed k, we have ζ operates on sample space Ω, σ-algebra $\mathcal{E}$, and probability measure $P_\zeta : \mathcal{E} \to [0, 1]$, such that $\zeta : (\Omega, \mathcal{E}, P_\zeta) \to (\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$, where $\mathcal{B}(\mathbb{R}^d)$ is the Borel set on $\mathbb{R}^d$. Thus, we have $\mathbb{E}[\mathscr{G}_k] \in \partial f(x_k)$, where $\partial f(x)$ denotes the set of subgradients of f at the point x. The proposed optimizer is given by:

$$x_{k+1} = x_k - \gamma_k \chi_k \odot \mathscr{G}_k, \tag{2}$$

where $\odot$ denotes the element-wise multiplication of two vectors, $x_k \in \mathbb{R}^d$, and the ith element of $\chi_k \in \mathbb{R}^d$ is defined as follows:

$$\chi_k = \begin{cases} 1, & \phi_k \geq 1 \\ \epsilon, & \phi_k \leq \epsilon \\ \phi_k, & \text{otherwise,} \end{cases} \tag{3}$$

where $\phi_k \triangleq \left|\frac{(x_k - x_{k-1})_i}{(\mathscr{G}_k - \mathscr{G}_{k-1})_i}\right|$, $0 < \epsilon < 1$, and γ_k is a scaling step size. In this context, each element of x_k is assigned a variable step size. We define the matrix B_k such that $\chi_k \odot \mathscr{G}_k = B_k \mathscr{G}_k$, where

$$B_k = \begin{bmatrix} (\chi_k)_1 & 0 & \cdots & 0 \\ 0 & (\chi_k)_2 & \cdots & 0 \\ \vdots & & \ddots & \\ 0 & \cdots & 0 & (\chi_k)_d \end{bmatrix}.$$

Thus, B_k is a positive-definite diagonal matrix, where the ith diagonal element of B_k is equal to $(\chi_k)_i > 0\ \forall i$.

In the deterministic setting, we assume that f is differentiable and replace $\mathscr{G}_k$ with $\nabla f(x_k)$, and we use a constant step size, that is, $\gamma_k = \gamma,\ \forall k$.

The idea behind our proposed algorithm is that the elements of χ_k are positive and not larger than one. Thus, for sufficiently small γ, convergence can be attained. Although our optimizer (2) is considered as a first-order method, it mimics Newton's method where the diagonal matrix B_k (composed of elements of ϕ_k) is used instead of the inverse of the Hessian matrix, see, e.g., Saab and Shen (2019). For example, whenever $f(x)$ is quadratic with positive diagonal quadratic matrix, B_k is considered to be a reasonable approximation of the inverse of the Hessian matrix. In fact, it can be shown that if $f(x)$ is a quadratic function with a positive diagonal quadratic matrix, then the optimal solution is reached in one iteration whenever $\gamma = 1, \forall k$ and $\chi_k = \phi_k$ in (3).

MADAGRAD algorithm is summarized in Algorithm 1.

Algorithm 1 MADAGRAD

```
1:  Choose $\gamma_k$; e.g., $\gamma_k = 1, \forall k$, and $0 \leq \epsilon << 1$
2:  Initialize: $x^- \leftarrow x_{-1}$, $x \leftarrow x_0$ with $x_{-1} \neq x_0$, and get $g^- \leftarrow \mathscr{G}_{-1}$
3:  for iteration $k = 1, 2, \ldots$ do
4:      $g \leftarrow \mathscr{G}$
5:      $\Delta g \leftarrow g - g^-$
6:      $\Delta x \leftarrow x - x^-$
7:      $g^- \leftarrow g$
8:      $x^- \leftarrow x$
9:      for $i = 1, 2, \ldots, p$ do
10:         $(\phi)_i \leftarrow \left|\frac{(\Delta x)_i}{(\Delta g)_i}\right|$
11:         $(\chi)_i \leftarrow (\phi)_i$
12:         if $(\chi)_i > 1$, $(\chi)_i \leftarrow 1$ end if
13:         if $(\chi)_i < \epsilon$, $(\chi)_i \leftarrow \epsilon$ end if
14:         $(x)_i \leftarrow (x)_i - \gamma(\chi)_i(\mathscr{G})_i$
15:     end for
16: end for
```

Examining Algorithm 1, we make the following observations:

- Little memory is required; in particular, storing only the vectors $\mathscr{G} \in \mathbb{R}^p$ and $x \in \mathbb{R}^p$ only for one iteration at a time.
- It is computationally efficient since it requires basic scalar operations of the elements of few p-dimensional vectors such as addition, division, multiplication, and absolute value.

3. Global convergence

We assume that $f(x)$ is L-smooth throughout this section. We use a sufficiently small $\gamma_k = \gamma,\ \forall k$ for the deterministic setting (Theorems 1 and 2), and a decreasing step-size given by $\gamma_k \propto 1/\sqrt{k+1}$ for the stochastic setting (Theorem 3) to guarantee convergence. However, such an aggressively decaying step-size may yield poor performance, whereas a fixed step size, $\gamma_k = 1,\ \forall k$, as analytically reflected in Corollaries 1 and 2, and Propositions 1 and 2, performs well in practice as illustrated in our experimental results. Theorem 1, Corollary 1 and Proposition 1 assume strong convexity of $f(x)$ whereas Theorem 2, Corollary 2, Proposition 2 and Theorem 3 are applicable to non-convex functions.

3.1. Deterministic setting

Motivation: Global convergence requires sufficiently small γ_k and the corresponding range depends on the value of L (Cohen, Kaur, Li, Kolter, & Talwalkar, 2021) for L-smooth cost functions, which is practically inaccessible. A small step-size is usually expected when using gradient descent (GD) methods and also required for methods using momentum. For example, the global convergence of GD requires sufficiently small step size, α, and the corresponding range depends on the value of L for L-smooth cost functions. If the cost function is quadratic, it is shown that the vanilla GD diverges whenever its step size $\alpha > \frac{2}{\lambda_i}$, where λ_i is any eigenvalue of the quadratic matrix, and NAG diverges whenever its step size $\alpha > \frac{1}{\lambda_i}\frac{2+2\beta}{1+2\beta} > \frac{4}{3}\frac{1}{\lambda_i}$, where $0 \leq \beta < 1$ is NAG momentum hyper-parameter (Cohen et al., 2021). The step-size of the Heavy-Ball (HB) method needs to satisfy $\alpha \in \left(0, \frac{2(1-\beta)}{L}\right)$, where $\beta \in (0,1)$ is the momentum hyper-parameter (Ghadimi, Feyzmahdavian, & Johansson, 2015) for deterministic setting and the similar conditions apply for stochastic setting (Yang, Lin, & Li, 2016) for SGD, stochastic HB (SHB), and stochastic Nesterov accelerated gradient (SNAG) method. On the other hand, we claim that our method does not require tuning of γ_k; by simply setting $\gamma_k = 1, \forall k$ since ϕ_k tends to locally approximate the corresponding value of "$1/L$" for each element of the gradient. This choice of $\gamma_k = 1$ is also justified experimentally in the following section. However, we rationalize our claim analytically in Corollaries 1 and 2, and Propositions 1 and 2 with some tight constraints on the cost function.

We first formally present the definitions that are used for attaining convergence.

Definition 1. A differentiable function $f(x)$ is called L-smooth if and only if it has a Lipschitz continuous gradient, i.e., if and only if there exists $L < \infty$ such that $\forall x, y \in \mathbb{R}^d$:

$$\|\nabla f(y) - \nabla f(x)\| \le L\|y - x\|. \tag{4}$$

Definition 2. The cost function $f(x)$ is μ-strongly convex, if it is differentiable, and there exists $\mu > 0$ such that the following inequality holds $\forall x, y \in \mathbb{R}^d$,

$$f(y) - f(x) \ge \langle y - x, \nabla f(x)\rangle + \frac{\mu}{2}\|y - x\|^2. \tag{5}$$

Theorem 1. *Suppose $f(x)$ is L-smooth and μ-strongly convex, and let $0 < \gamma_k = \gamma < \frac{2\epsilon}{L}, \forall k$. Then, the sequence $\{x_k\}$ generated by the algorithm described by (2) satisfies*

$$f(x_k) - f(x^*) \le q^k(f(x_0) - f(x^*)), \tag{6}$$

where the convergence factor $q = 1 + 2\mu\gamma(\frac{L}{2}\gamma - \epsilon) \in (0, 1)$.

Proof. Since $f(x)$ is L-smooth, it follows that (Nesterov, 2003)

$$f(x_{k+1}) - f(x_k) \le \langle \nabla f(x_k), (x_{k+1} - x_k)\rangle + \frac{L}{2}\|x_{k+1} - x_k\|^2.$$

We have $x_{k+1} - x_k = -\gamma B_k \nabla f(x_k)$ where $\epsilon I \preceq B_k \preceq I$, which leads to

$$\begin{aligned}\langle \nabla f(x_k), x_{k+1} - x_k\rangle &= -\gamma\langle \nabla f(x_k), B_k\nabla f(x_k)\rangle \\ &\le -\gamma\epsilon\|\nabla f(x_k)\|^2\end{aligned}$$

and $\|x_{k+1} - x_k\|^2 = \gamma^2\|B_k\nabla f(x_k)\|^2 \le \gamma^2\|\nabla f(x_k)\|^2$. Therefore, we obtain

$$f(x_{k+1}) - f(x_k) \le \left(\frac{L}{2}\gamma^2 - \gamma\epsilon\right)\|\nabla f(x_k)\|^2.$$

With $0 < \gamma < \frac{2\epsilon}{L}$, we have $\frac{L}{2}\gamma^2 - \gamma\epsilon < 0$ and since $f(x_k)$ is μ-convex, then $\|\nabla f(x_k)\|^2 \ge 2\mu(f(x_k) - f(x^*))$, thus

$$f(x_{k+1}) - f(x_k) \le 2\mu\gamma(\frac{L}{2}\gamma - \epsilon)(f(x_k) - f(x^*)).$$

Subtracting $f(x_*)$ from both sides leads to

$$f(x_{k+1}) - f(x^*) \le (1 + 2\mu\gamma(\frac{L}{2}\gamma - \epsilon))(f(x_k) - f(x^*)) \tag{7}$$

or equivalently, $f(x_k) - f(x^*) \le q^k(f(x_0) - f(x^*))$, where $q = 1 + 2\mu\gamma(\frac{L}{2}\gamma - \epsilon)$; and $0 < q < 1$ since $0 < \gamma < \frac{2\epsilon}{L}$. It is noted that for $0 < \gamma < \frac{\epsilon}{L}$, $\frac{dq(\gamma)}{d\gamma} = 2\mu(L\gamma - \epsilon) < 0$ for $0 < \gamma < \frac{\epsilon}{L}$. At $\gamma = 0$ or $\gamma = \frac{2\epsilon}{L}$, we have $q = 1$, and at $\gamma = \frac{\epsilon}{L}$, we have $q = 1 - \frac{\mu\epsilon^2}{L}$. Since $0 < \mu \le L$[1] and $0 \le \epsilon < 1$, then at their limit values, $\mu = L$ and $\epsilon = 1$, we have $q = 0$, which is the minimum value of q. □

Corollary 1. *If $L < 2$, by setting $\gamma_k = 1$ and ϵ such that $\frac{L}{2} < \epsilon < 1$, then we attain the linear convergence in (6) with convergence factor $q = 1 + 2\mu(\frac{L}{2} - \epsilon) \in (0, 1)$.*

Proof. Since $\gamma = 1$ and $\frac{L}{2} < \epsilon < 1$, then the driving condition of (7) is met. That is, $q = 1 + 2\mu(\frac{L}{2} - \epsilon) \in (0, 1)$. □

Theorem 2. *Suppose $f(x)$ is L-smooth but not necessarily convex, and $0 < \gamma_k = \gamma < \frac{2\epsilon}{L}, \forall k$. Then, the sequence $\{x_k\}$ generated by the algorithm described by (2), satisfies*

$$\min_{k=0,\dots,K}\|\nabla f(x_k)\|^2 \le \frac{f(x_0) - f(x^*)}{\left(\gamma\epsilon - \frac{L\gamma^2}{2}\right)}\frac{1}{K+1}. \tag{8}$$

Proof. Since $f(x)$ is a L-smooth function, we have

$$\begin{aligned}f(x_{k+1}) - f(x_k) &\le \langle \nabla f(x_k), x_{k+1} - x_k\rangle + \frac{L}{2}\|x_{k+1} - x_k\|^2 \\ &= -\gamma\langle \nabla f(x_k), B_k\nabla f(x_k)\rangle \\ &+ \frac{L\gamma^2}{2}\|B_k\nabla f(x_k)\|^2.\end{aligned}$$

Every element $(\chi_k)_i$ is upper-bounded by 1, and lower-bounded by ϵ. Thus, we have $\epsilon I \preceq B_k \preceq I$ and $\|B_k\| \le 1$. For compactness of presentation, we define $g_k \triangleq \nabla f(x_k)$. Since B_k is a diagonal matrix, we can write

$$\begin{aligned}f(x_{k+1}) - f(x_k) &\le -\gamma\langle B_k g_k, g_k\rangle + \frac{L\gamma^2}{2}\|g_k\|^2 \\ &\le -\left(\gamma\epsilon - \frac{L\gamma^2}{2}\right)\|g_k\|^2,\end{aligned}$$

or

$$\left(\gamma\epsilon - \frac{L\gamma^2}{2}\right)\|g_k\|^2 \le f(x_k) - f(x_{k+1}).$$

Summing this inequality for $k = 0, \dots, K$ while cancelling common terms, we obtain

$$\begin{aligned}\sum_{k=0}^{K}\left(\gamma\epsilon - \frac{L\gamma^2}{2}\right)\|g_k\|^2 &\le f(x_0) - f(x_{K+1}) \\ &\le f(x_0) - f(x^*).\end{aligned}$$

The last inequality is based on $f(x^*) \le f(x_{K+1})$. Note that

$$\min_{k=0,\dots,K}\|g_k\|^2\sum_{k=0}^{K}\left(\gamma\epsilon - \frac{L\gamma^2}{2}\right) \le \sum_{k=0}^{K}\left(\gamma\epsilon - \frac{L\gamma^2}{2}\right)\|g_k\|^2.$$

Since $\gamma < \frac{2\epsilon}{L}$ and thus $\sum_{k=0}^{K}(\gamma\epsilon - \frac{L\gamma^2}{2}) > 0$, it follows that

$$\begin{aligned}\min_{k=0,\dots,K}\|g_k\|^2 &\le \frac{f(x_0) - f(x^*)}{\sum_{k=0}^{K}\left(\gamma\epsilon - \frac{L\gamma^2}{2}\right)} \\ &= \frac{f(x_0) - f(x^*)}{(K+1)\left(\gamma\epsilon - \frac{L\gamma^2}{2}\right)}.\end{aligned}$$

This ends the proof. □

Corollary 2. *If $L < 2$, by setting $\gamma_k = \gamma = 1$ and ϵ such that $\frac{L}{2} < \epsilon < 1$, then we attain the convergence in (8).*

[1] Since $f(x)$ is L-smooth, then $\|\nabla f(y) - \nabla f(x)\| \le L\|y - x\|$. In addition, if $f(x)$ is μ-strongly convex, then from Theorem 2.1.10 (Nesterov, 2003) we have $\forall\, x, y \in \mathbb{R}^d$, $\frac{1}{2\mu}\|\nabla f(y) - \nabla f(x)\|^2 \ge f(y) - f(x) - \langle \nabla f(x), y - x\rangle$. From Definition 2, we have $f(y) - f(x) \ge \langle y - x, \nabla f(x)\rangle + \frac{\mu}{2}\|y - x\|^2$. Therefore, $\|\nabla f(y) - \nabla f(x)\| \ge \mu\|y - x\|$; hence $0 < \mu \le L$.

Proof. Since $\gamma = 1$ and $\frac{L}{2} < \epsilon < 1$, then the driving condition of (8), $\sum_{k=0}^{K}\big(\gamma\epsilon - \frac{L\gamma^2}{2}\big) > 0$ is met. □

Let $(B_k)_{ii} = (\chi_k)_i$. Define $L_{i,k}$, $\bar{L}_k$, and $\underline{L}_k$, such that $(B_k)_{ii} = \frac{1}{L_{i,k}}$, $\bar{L}_k \triangleq \max_i L_{i,k}$ and $\underline{L}_k \triangleq \min_i L_{i,k}$.

In Propositions 1 and 2, we impose assumptions on $\bar{L}_k$ and $\underline{L}_k$, which cannot be generally verified a priori. However, the objective of these propositions is only to show that with $\gamma_k = \gamma = 1$, $\forall k$, we can attain convergence. Nonetheless, we provide an example to show how the conditions of Propositions 1 and 2 can be met. Let $f(x_k) = \frac{1}{2}x_k^T A x_k + b^T x_k + c$, where A is a diagonal matrix with extreme eigenvalues L and $\mu \le L$. It can be shown that for this special case, $(\chi_k)_i = \frac{1}{A_{ii}}$, $\forall k$. Therefore, $\bar{L}_k = L$ and $\underline{L}_k = \mu$. Let $L = 1.2$ and $\mu = 1$. We find that $-1 < 2\mu\big(\frac{L}{2}\frac{1}{\underline{L}_k^2} - \frac{1}{\bar{L}_k}\big) = -0.47 < 0$, $\forall k$, which implies that the condition of Proposition 1 is satisfied. In addition, $\frac{2\underline{L}_k^2}{\bar{L}_k} = 1.67 > L$, $\forall k$, which implies that the condition of Proposition 2 is also satisfied.

Proposition 1. *Suppose $f(x)$ is L-smooth and μ-strongly convex, and let $\gamma_k = 1, \forall k$ and $\chi_k = \phi_k$ in (3). If there exists $\eta_0 < 0$ such that $-1 < 2\mu\big(\frac{L}{2}\frac{1}{\underline{L}_k^2} - \frac{1}{\bar{L}_k}\big) \le \eta_0 < 0$, then the sequence $\{x_k\}$ generated by the algorithm described by (2) satisfies*

$$f(x_k) - f(x^*) \le \rho^k (f(x_0) - f(x^*)), \tag{9}$$

where the convergence factor $\rho = 1 + \eta_0 \in (0, 1)$.

Proof. Since $f(x)$ is L-smooth, we have $\|\nabla f(x_{k+1}) - \nabla f(x_k)\| \le L\|x_{k+1} - x_k\|$ $\forall k$. We have $\frac{1}{\bar{L}_k}I \preceq B_k \preceq \frac{1}{\underline{L}_k}I$, and we get the two following inequalities

$$\begin{aligned}\langle \nabla f(x_k), x_{k+1} - x_k\rangle &= -\langle \nabla f(x_k), B_k \nabla f(x_k)\rangle \\ &\le -\frac{1}{\bar{L}_k}\|\nabla f(x_k)\|^2,\end{aligned}$$

and $\|x_{k+1} - x_k\|^2 = \|B_k \nabla f(x_k)\|^2 \le \frac{1}{\underline{L}_k^2}\|\nabla f(x_k)\|^2$.

Since $f(x)$ is L-smooth, it follows that

$$\begin{aligned}f(x_{k+1}) - f(x_k) &\le \langle \nabla f(x_k), (x_{k+1} - x_k)\rangle + \frac{L}{2}\|x_{k+1} - x_k\|^2 \\ &\le \Big(\frac{L}{2}\frac{1}{\underline{L}_k^2} - \frac{1}{\bar{L}_k}\Big)\|\nabla f(x_k)\|^2.\end{aligned} \tag{10}$$

Since $\eta_0 < 0$ (and $\mu > 0$), then $\frac{L}{2}\frac{1}{\underline{L}_k^2} - \frac{1}{\bar{L}_k} < 0$. Since $f(x)$ is μ-strongly convex, we obtain

$$f(x_{k+1}) - f(x_k) \le 2\mu\Big(\frac{L}{2}\frac{1}{\underline{L}_k^2} - \frac{1}{\bar{L}_k}\Big)(f(x_k) - f(x^*)).$$

Subtracting $f(x^*)$ on both sides leads us to

$$f(x_{k+1}) - f(x^*) \le \Big(1 + 2\mu\Big(\frac{L}{2}\frac{1}{\underline{L}_k^2} - \frac{1}{\bar{L}_k}\Big)\Big)(f(x_k) - f(x^*)). \tag{11}$$

Since $-1 < 2\mu\big(\frac{L}{2}\frac{1}{\underline{L}_k^2} - \frac{1}{\bar{L}_k}\big) \le \eta_0 < 0$, this ends the proof. □

Proposition 2. *Suppose that $f(x)$ is L-smooth function and not necessarily convex. Let $\gamma_k = 1, \forall k$ and $\chi_k = \phi_k$ in (3). If $L < \frac{2\underline{L}_k^2}{\bar{L}_k}$ $\forall k$, then the sequence $\{x_k\}$ generated by the algorithm described by (2) satisfies*

$$f(x_{k+1}) - f(x^*) < f(x_k) - f(x^*), \forall k. \tag{12}$$

Proof. By assuming that $L < \frac{2\underline{L}_k^2}{\bar{L}_k}$, we get $\frac{L}{2}\frac{1}{\underline{L}_k^2} - \frac{1}{\bar{L}_k} < 0$, then from (10), we have $f(x_{k+1}) < f(x_k)$. Subtracting $f(x^*)$ from both sides leads to (12). □

In Propositions 1 and 2, the assumption of $L < \frac{2\underline{L}_k^2}{\bar{L}_k}$ and $\mu \le \frac{\bar{L}_k}{2}$ limits the class of cost functions.

3.2. Stochastic setting

In what follows, we denote by $\mathbb{E}_k[\cdot]$ the expectation over the randomness in $\zeta_{[k]} \triangleq (\zeta_1, \dots, \zeta_k)$ and let $\mathbb{E}_{k|k-1}[\cdot]$ denote the expectation over ζ_k given $\zeta_1, \dots, \zeta_{k-1}$ fixed. We use $\mathbb{E}[\cdot]$ to denote all randomness.

3.2.1. Smooth and non-convex functions

The subsequent theorem is useful for selecting the number of iterations a priori, K, where the step size is fixed depending on the value of K.

Theorem 3. *Suppose $f(x)$ is L-smooth but not necessarily convex, with $\mathbb{E}\big[\mathscr{G}(x;\zeta) - \nabla f(x)\big] = 0$, $\mathbb{E}[\zeta_i^T\zeta_j] = 0\, \forall i \ne j$, and $\mathbb{E}\big[\|\mathscr{G}(x;\zeta) - \nabla f(x)\|^2\big] \le \delta^2$, $\forall x$. Then, by setting $\gamma_k = \min\{\frac{\epsilon}{L}, \frac{C}{\delta\sqrt{K+1}}\}$ for $k = 0, 1, \dots, K$, the sequence $\{x_k\}$ generated by the algorithm described by (2), satisfies*

$$\begin{aligned}\min_{k=0,\dots,K} \mathbb{E}\Big[\|\nabla f(x_k)\|^2\Big] &\le \frac{D_f}{\epsilon(K+1)}\max\Big\{\frac{L}{\epsilon}, \frac{\delta\sqrt{K+1}}{C}\Big\} \\ &\quad + \frac{L\delta^2}{2\epsilon}\min\Big\{\frac{\epsilon}{L}, \frac{C}{\delta\sqrt{K+1}}\Big\},\end{aligned} \tag{13}$$

where $D_f \triangleq \mathbb{E}[f(x_0) - f(x^*)]$.

Proof. The proof follows a similar approach as in Ghadimi and Lan (2013).

Since $f(x)$ is a L-smooth function, we have

$$\begin{aligned}f(x_{k+1}) - f(x_k) &\le \langle \nabla f(x_k), x_{k+1} - x_k\rangle + \frac{L}{2}\|x_{k+1} - x_k\|^2 \\ &= -\gamma_k\langle \nabla f(x_k), B_k\mathscr{G}_k\rangle + \frac{L\gamma_k^2}{2}\|B_k\mathscr{G}_k\|^2.\end{aligned}$$

Every element $(\chi_k)_i$ is upper-bounded by 1, and lower-bounded by ϵ. Therefore, we have $\epsilon I \preceq B_k \preceq I$ and $\|B_k\| \le 1$. By defining $g_k \triangleq \nabla f(x_k)$ and $\delta_k \triangleq \mathscr{G}_k - g_k$. Since B_k is a diagonal matrix, we can write

$$\begin{aligned}f(x_{k+1}) - f(x_k) &\le -\gamma_k\langle B_k g_k, \delta_k\rangle - \gamma_k\langle B_k g_k, g_k\rangle \\ &\quad + \frac{L\gamma_k^2}{2}\|\mathscr{G}_k\|^2 \\ &\le -\gamma_k\langle B_k g_k, \delta_k\rangle - \gamma_k\epsilon\|g_k\|^2 + \frac{L\gamma_k^2}{2}\|\mathscr{G}_k\|^2.\end{aligned}$$

We have

$\|\mathscr{G}_k\|^2 = \langle g_k + \delta_k, g_k + \delta_k\rangle = \|g_k\|^2 + \|\delta_k\|^2 + 2\langle g_k, \delta_k\rangle$. Thus,

$$\begin{aligned}f(x_{k+1}) - f(x_k) &\le -\gamma_k\langle B_k g_k, \delta_k\rangle - \Big(\gamma_k\epsilon - \frac{L\gamma_k^2}{2}\Big)\|g_k\|^2 \\ &\quad + \frac{L\gamma_k^2}{2}\|\delta_k\|^2 + L\gamma_k^2\langle g_k, \delta_k\rangle.\end{aligned}$$

Note that B_k is bounded, $\mathbb{E}[\delta_k] = 0$, $\mathbb{E}[\zeta_i^T\zeta_j] = 0\ \forall i \ne j$, and x_k is a function of the history of $\zeta_{[k-1]}$. Thus $\mathbb{E}[\langle g_k, \delta_k\rangle|\zeta_{[k-1]}] = 0$, and $\mathbb{E}[\langle B_k g_k, \delta_k\rangle|\zeta_{[k-1]}] = 0$. Taking the conditional expectation on both sides, we obtain

$$\mathbb{E}[f(x_{k+1}) - f(x_k)] \le -\Big(\gamma_k\epsilon - \frac{L\gamma_k^2}{2}\Big)\mathbb{E}\Big[\|g_k\|^2\Big] + \frac{L\gamma_k^2}{2}\|\delta_k\|^2$$

or

$$\Big(\gamma_k\epsilon - \frac{L\gamma_k^2}{2}\Big)\mathbb{E}\Big[\|g_k\|^2\Big] \le \mathbb{E}[f(x_k) - f(x_{k+1})] + \frac{L\gamma_k^2}{2}\|\delta_k\|^2.$$

Summing this inequality for $k = 0, \ldots, K$ while cancelling common terms, we obtain

$$\sum_{k=0}^{K}\left(\gamma_k\epsilon - \frac{L\gamma_k^2}{2}\right)\mathbb{E}\left[\|g_k\|^2\right] \leq \mathbb{E}[f(x_0) - f(x_{K+1})] + \frac{L\delta^2}{2}\sum_{k=0}^{K}\gamma_k^2 \leq \mathbb{E}[f(x_0) - f(x^*)] + \frac{L\delta^2}{2}\sum_{k=0}^{K}\gamma_k^2.$$

The last inequality is based on $f(x^*) \leq f(x_{K+1})$. Note that

$$\min_{k=0,\ldots,K} \mathbb{E}\left[\|g_k\|^2\right]\sum_{k=0}^{K}\left(\gamma_k\epsilon - \frac{L\gamma_k^2}{2}\right) \leq \sum_{k=0}^{K}\left(\gamma_k\epsilon - \frac{L\gamma_k^2}{2}\right)\mathbb{E}\left[\|g_k\|^2\right].$$

Since $\gamma_k = \min\{\frac{\epsilon}{L}, \frac{C}{\delta\sqrt{K+1}}\}$ and $\sum_{k=0}^{K}(\gamma_k\epsilon - \frac{L\gamma_k^2}{2}) > 0$, it follows that

$$\min_{k=0,\ldots,K} \mathbb{E}\left[\|g_k\|^2\right] \leq \frac{D_f + (L/2)\delta^2\sum_{k=0}^{K}\gamma_k^2}{\sum_{k=0}^{K}\left(\gamma_k\epsilon - \frac{L\gamma_k^2}{2}\right)}, \tag{14}$$

where $D_f \triangleq \mathbb{E}[f(x_0) - f(x^*)]$. Note that $\gamma_0 = \gamma_k, \forall k$. Therefore,

$$\frac{D_f + (L/2)\delta^2\sum_{k=0}^{K}\gamma_k^2}{\sum_{k=0}^{t}\left(\gamma_k\epsilon - \frac{L\gamma_k^2}{2}\right)} = \frac{D_f + (L/2)\delta^2(K+1)\gamma_0^2}{(K+1)\gamma_0(\epsilon - \frac{L\gamma_0}{2})} \leq \frac{D_f + (L/2)\delta^2\gamma_0^2(K+1)}{\gamma_0\epsilon(K+1)} = \frac{D_f}{\gamma_0\epsilon(K+1)} + \frac{(L/2)\delta^2\gamma_0}{\epsilon}.$$

Since $\gamma_0 = \min\{\frac{\epsilon}{L}, \frac{C}{\delta\sqrt{K+1}}\}$, then $\frac{1}{\gamma_0} = \max\left\{\frac{L}{\epsilon}, \frac{\delta\sqrt{K+1}}{C}\right\}$. Thus,

$$\frac{D_f}{\gamma_0\epsilon(K+1)} + \frac{(L/2)\delta^2\gamma_0}{\epsilon} = \frac{D_f}{\epsilon(K+1)}\max\left\{\frac{L}{\epsilon}, \frac{\delta\sqrt{K+1}}{C}\right\} + \frac{(L/2)\delta^2}{\epsilon}\min\left\{\frac{\epsilon}{L}, \frac{C}{\delta\sqrt{K+1}}\right\}.$$

Using the above inequalities and (14) lead to (13). □

3.2.2. Non-smooth and strongly convex functions

$$x_{k+1} = \Pi_{\mathcal{X}}(x_k - \gamma_k\chi_k \odot \mathcal{G}_k), \tag{15}$$

where χ_k is defined in (3), $\Pi_{\mathcal{X}}$ denotes the projection on a convex domain $\mathcal{X}$, which is assumed to be a subset of some Hilbert space.

The following results show that MADAGRAD can be used to optimize any non-smooth strongly convex cost function over $\mathcal{X}$ given access to unbiased estimates of its subgradients.

Theorem 4. *Assume that $f(x)$ is a non-smooth μ-strongly convex function, $\mathbb{E}[\|\mathcal{G}_k\|^2] \leq G^2$ $\forall k$, and $f(x)$ attains a minimum at some $x^* \in \mathcal{X}$. By setting $\gamma_k = \frac{1}{\mu k}$ and $x_0 = 0$, then for any $T > 1$, the sequence $\{x_k\}$ generated by the algorithm described by (15), satisfies*

$$\mathbb{E}[f(x_T) - f(x^*)] \leq \frac{17G^2(1 + \log(T))}{\mu T}. \tag{16}$$

Theorem 4 provides a bound on $\mathbb{E}[f(x_T) - f(x^*)]$ for a bounded T. One limitation of Theorem 4 is the boundedness of the subgradient. The latter is used to facilitate the convergence bound proof in (16).

The proof of Theorem 4 is included in the Appendix since it follows similar steps as the proof of Theorem 1 in Shamir and Zhang (2013) for SGD.

4. Results and discussion

In this section we evaluate the performance of our proposed optimizer (2) with $\gamma_k = 1$ for all experiments. First, we show the superiority of the convergence rate of our proposed optimizer when compared to the optimal HB method, along with several other optimizers, on the strongly convex function given in Lessard, Recht, and Packard (2016) in Section 4.2. We further evaluate the convergence rate of our optimizer by comparing it with first-order optimizers on the (non-convex and non-quadratic) Beale function in Section 4.3. For the two aforementioned experiments, the lower bound ϵ of (3) is dismissed, meaning no tuning whatsoever is implemented on our method. Finally, we evaluate the optimizer's performance against popular optimizers on MNIST, QMNIST, CIFAR-10, and CIFAR-100, where we fix $\epsilon = 0.05$ of (3) as default value for image classification. It is important to note that in our toy examples (Lessard function, quadratic and Beale functions), we consistently run all optimizers under consideration through all the samples in our training set to do a single update for the weight, x_k, in every iteration, k. On the other hand, in our image classification examples, we also consistently run all optimizers under consideration while using only one minibatch or one subset of the training set to do the update for the weight in every iteration.

Our codes for all experiments will be made publicly available.

4.1. Convex smooth quadratic function

We use a positive-definite quadratic function to illustrate the linear convergence rate as reflected in Theorem 1. We consider the following cost function

$$f(x) = \frac{1}{2}x^TAx - b^Tx \tag{17}$$

where

$$A = \begin{bmatrix}100 & 1\\ 1 & 1\end{bmatrix}, b = \begin{bmatrix}1\\ 1\end{bmatrix}, x_{-1} = \begin{bmatrix}0\\ 0\end{bmatrix}, \text{ and } x_0 = \begin{bmatrix}2\\ 2\end{bmatrix}.$$

Instead of (2), we use $x_{k+1} = x_k - \phi_k \odot \nabla f(x_k)$, where $\phi_k \triangleq \left|\frac{(x_k - x_{k-1})_i}{\nabla f(x_k) - \nabla f(x_{k-1})_i}\right|$. The performance of our optimizer is illustrated in Fig. 1. Examining the left plot, it can be concluded that based on Eq. (6) in Theorem 1, $q < 0.1$. In addition, we consider the convergence rate of the optimal Polyak Heavy Ball, with corresponding convergence rate of $\theta^*_{HB} \triangleq \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$, which provides the fastest local convergence rate among all the first-order methods. The latter assumes that the hyper parameters are constants — unlike adaptive methods such as MADAGRAD. For this example, $L = 100$, $\mu = 0.99$, and the corresponding $\theta^*_{HB} = 0.819$. The right plot of Fig. 1 shows that $\theta_{\text{MADAGRAD}} < 0.3$, which is significantly smaller than θ^*_{HB}. Of note, the optimal Polyak Heavy Ball requires the exact knowledge of L and μ, whereas MADAGRAD is adaptive and basically requires no tuning. It is also worthwhile mentioning that similar results are obtained when we use $x_{k+1} = x_k - \chi_k \odot \nabla f(x_k)$, where χ_k is defined in (2) with $\epsilon = 0$.

4.2. Convex not continuously differentiable Lessard's function

In this section, we tackle the strongly convex function, $f(x)$, presented in Lessard et al. (2016), given by

$$\nabla f(x_k) = \begin{cases} 25x & \text{if } x < 1\\ x + 24 & \text{if } 1 \leq x < 2\\ 25x - 24 & \text{if } x \geq 2.\end{cases} \tag{18}$$

The Polyak HB method is not guaranteed to converge on this function. The function's gradient, $\nabla f(x)$, is continuous and

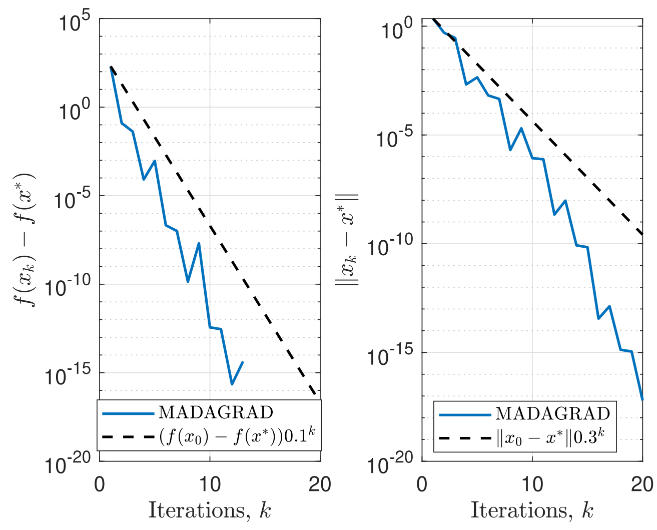

Fig. 1. Visualization of the convergence rate of MADAGRAD for a positive-definite quadratic cost function.

Table 1
The convergence results of the proposed optimizer in comparison to known optimizers when tested on Lessard's function.

Optimizer	Wins	Times converged	Average steps
Ours	**(1000; 1000)**	**(1000; 1000)**	**(1; 3)**
diffGrad	(0; 0)	(0; 0)	(50; 50)
SGD	(0; 0)	(1000; 1000)	(17.285; 16.977)
SGDm	(0; 0)	(0; 0)	(50; 50)
NAG	(0; 0)	(1; 1)	(49.998; 49.993)
RMSProp	(0; 0)	(0; 0)	(50; 50)
Adagrad	(0; 0)	(505; 448)	(35.097; 37.274)
Adam	(0; 0)	(0; 0)	(50; 50)

Table 2
The convergence rate of the proposed optimizer in comparison to known optimizers when tested on the Beale function.

Optimizer	Wins	Times converged	Average steps
Ours	**652**	**999**	**160.87**
diffGrad	10	539	619.70
SGD	0	27	992.94
SGDm	71	319	719.11
NAG	9	628	602.21
RMSProp	3	86	933.54
Adagrad	17	346	772.11
Adam	233	535	586.37

monotone but not continuously differentiable. We compare MADAGRAD with diffGrad (Dubey et al., 2019), SGD, SGD with Momentum (SGDm) (Qian, 1999), Nesterov's accelerated gradient (NAG) method (Nesterov, 1983), AdaGrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), and Adam (Kingma & Ba, 2014). The learning rates for all optimizers, excluding our proposed method, are chosen by conducting a random search over the values $\{0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001\}$, where we ran each optimizer over 1000 random initializations of the model parameters. The optimizer parameters that returned the fastest convergence rates were chosen.

If the initial conditions are chosen within $3.07 \leq x_0 \leq 3.46$, then the optimal HB gets stuck in a limit cycle. We test our optimizer with initial conditions sampled from $x_0 < 1$, and $x_0 \geq 1$ while ensuring some samples are drawn from the interval $3.07 \leq x_0 \leq 3.46$. Our optimizer converges in one iteration for all $x_0 < 1$. This is expected as the gradient for $x < 1$ is $\nabla f(x) = 25x$, and the solution $x_* = 0$. Similarly to the steps taken in the discussion section when assuming a quadratic with positive diagonal quadratic matrix, we have $x_1 - x_0 = -\left((x_0) + \frac{b}{A}\right)$, where $b = 0$ and $A = 25$. Therefore $x_1 = 0$ which is the solution. However b is non-zero whenever $x \geq 1$, and thus $x_1 = -\frac{b}{A} \neq 0$. We run the optimizers over 1000 seeds using 50 iterations each for initial conditions sampled from uniform distributions between $[-5, 1)$ and $[1, 5]$. The convergence threshold is $|x_k - x_*| < 10^{-10}$. The results are summarized in Table 1. A *win* is recorded for the optimizers that converge to a solution *first* within the first 50 iterations.

By examining Table 1, we find that our method converges in one iteration for all initial conditions sampled from $[1, 5]$, and it also converges in 3 iterations for all the initial conditions sampled from $[-5, -1]$. The second best optimizer, SGD, also converges 100% of the time with average number of iterations about 17. We observe that all of those methods involving a momentum term such as Adam, RMSProp, NAG, SGDm and diffGrad fail to converge under the given convergence threshold and the initial conditions under consideration.

4.3. Non-convex smooth Beale function

The Beale function is listed as one of 175 benchmark test functions for optimization algorithms (Jamil & Yang, 2013). It is a 2-dimensional non-convex function with one global minimum $f(x_*) = 0$ at $x_* = (3, 0.5)$. The function is written as follows:

$$f(x, y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$$

We compare MADAGRAD with the same optimizers in the previous subsection, excluding the optimal HB, using the same tuning strategy. We run each optimizer over 1000 random initializations of the model parameters, which are sampled from a uniform distribution over the interval $[0, 4)$. The convergence rate is evaluated by recording the number of steps, k, for any optimizer to achieve a difference in error of $|f(x_*) - f(x_k)| \leq 10^{-5}$. The value of x_{k-1} at the first iteration is initialized to $(0, 0)$. The learning rates chosen for SGD, Momentum, NAG, Adagrad, and Adam are chosen using a similar protocol as in the previous subsection and are found to be 0.01, 0.01, 0.0005, 0.5, and 0.5, respectively. For Momentum the standard value of $\beta = 0.9$ is used, a common $\beta = \frac{k}{k+3}$ for NAG, and the standard $\beta_1 = 0.9$ and $\beta_2 = 0.99$ for Adam.

Clearly, from Table 2, our proposed optimizer outperforms the rest by converging to the global minimum the largest number of times and the fastest. The proposed optimizer converges 99.9% of all runs with average number of iterations ≈161, whereas the second best performing optimizer converges 62.8% with average number of iterations ≈620.

In order to justify the results in Theorem 2, we also inject noise in the non-convex Beale function. We add a uniformly distributed zero-mean random variable to its gradient with values $\in 5 \times 10^{-3}[-1, 1]$. We choose $\epsilon = 10^{-5}$ in (2) and

$$\gamma_k = \begin{cases} \frac{1}{\sqrt{k+1}}, & \text{if } k = \text{ mod } 100 = 0, \text{ set } k_0 = k \\ \frac{1}{\sqrt{k_0+1}}, & \text{otherwise.} \end{cases} \tag{19}$$

In Fig. 2, using MADAGRAD, we display the progress of $\|\nabla f(x_k)\|^2$, $\min_{k=0,\ldots,t} \|\nabla f(x_k)\|^2$, $\frac{1}{t+1}\sum_{k=0}^{t} \|\nabla f(x_k)\|^2$, and also $\|\nabla f(x_0)\|^2 / \sqrt{k+1}$ to show how the minimum and the average of $\|\nabla f(x_k)\|^2$ converges at a rate of $\mathcal{O}(1/\sqrt{k+1})$ over 10,000 iterations.

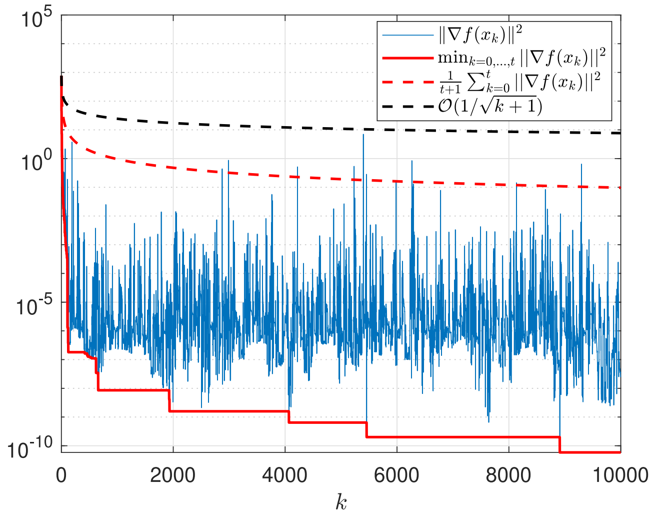

Fig. 2. Progress of $\|\nabla f(x_k)\|^2$ for the Beale function using noisy gradients with $x_0 = [1\ 1]^T$ and $x_{-1} = [0\ 0]^T$.

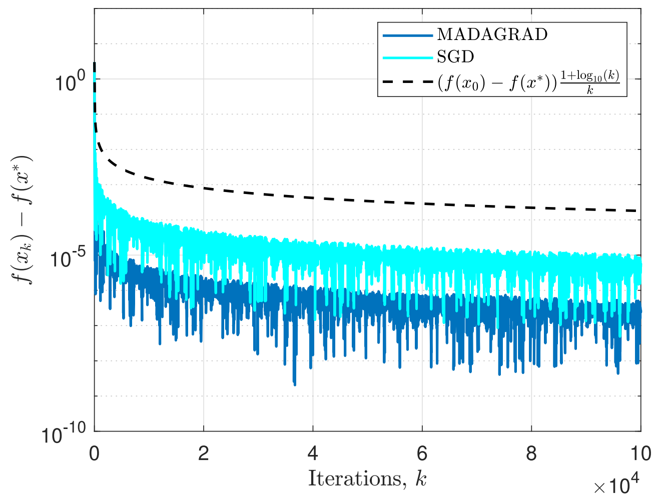

Fig. 3. Progress of $f(x_k) - f(x^*)$ for the L^1-norm function using noisy gradients with $x_{-1} = [2\ \ 2\ \ 2]^T$ and $x_0 = [1\ \ 1\ \ 1]^T$.

4.4. Convex non smooth L^1-norm function

We consider the L^1-norm as our cost function; that is, $f(x) = \|x\|_1$ with

$$\partial f(x) = \begin{cases} \frac{x}{\|x\|_1}, & \text{for } x \neq 0 \\ \{s \mid \|s\|_1 \leq 1\}, & \text{for } x = 0 \end{cases}$$

where the minimum is $x^* = 0$ and $0 \in \partial f(0)$. In this example, we compare the performance of MADAGRAD and SGD since the upper bounds of $\mathbb{E}[f(x_T) - f(x^*)]$ in Theorem 4 and Theorem 1 in Shamir and Zhang (2013) corresponding to SGD are the same. We use $x \in \mathbb{R}^3$ with $\gamma_k = \frac{1}{k}$, for both Madagrad and SGD, and for MADAGRAD, we set $\epsilon = 0.05$, which is the value used in all of our image classification examples. At each iteration, we add to the gradient a random vector, $v \in \mathbb{R}^3$, with $(v)_{1\leq i\leq 3} \in \mathcal{N}(0, 10^{-4})$. Examining Fig. 3, we note that $f(x_k)-f(x^*) \leq (f(x_0)-f(x^*))\frac{1+\log_{10}(k)}{k}$, which is consistent with the upper bound (16) of Theorem 4. In addition, the upper envelope of $f(x_k) - f(x^*)$ corresponding to MADAGRAD is about 17 times smaller than the one for SGD, and $\min_k(f(x_k) - f(x^*))$ is about 24 times smaller than the one for SGD.

4.5. Image classification

Before presenting the results and discussion, we provide a brief description of the popular adaptive methods. Root Mean Square Propagation (RMSProp) divides the learning rate by a running average of the magnitudes of recent gradients. Adaptive Moment Estimation (Adam), which is considered as an update to RMSProp, uses running averages of both the gradients and the second moments of the gradients. Adam with decoupled weight decay (AdamW) is a modification of Adam where the weight decay is decoupled from the optimization steps taken with respect to the loss function. Difference between the present and the immediate past gradient (diffGrad) builds on the proven Adam optimizer by developing an adaptive ‘friction clamp’ and monitoring the local change in gradients — not just by using the exponential moving average. The latter modification is aimed at automatically locking in optimal parameter values that Adam can skip over. All of these methods rely on the square roots of exponential moving averages of squared past gradients. On the other hand, by examining Algorithm 1, it can be readily concluded that MADAGRAD does not require more computational cost per iteration or more memory requirement than all the aforementioned popular adaptive methods.

Table 3
Comparison of tuning efforts required for our approach versus popular first-order adaptive and non-adaptive optimizers via number of tuning hyper-parameters.

Optimizer	Number of tuning hyper-parameters
Ours	2: γ, ϵ ($\gamma \equiv 1, \epsilon \equiv 0.05$)
SGDm	2: γ, β
NAG	2: γ, β
RMSProp	2: γ, β
Adam	3: γ, β_1, β_2
diffGrad	3: γ, β_1, β_2

We also reflect on the amount of tuning efforts in Table 3 by comparing the number of hyper-parameters that need to be tuned for each method For example, even though the number of hyper-parameters to tune for our method is 2, we use $\gamma \equiv 1, \epsilon \equiv 0.05$ for all datasets. This $\epsilon = 0.05$ is holistically selected so that the performance of any dataset is not substantially compromised. Of note, choosing a different value of ϵ for each dataset would lead to better performance. We empirically show that with these fixed default values, our method achieves comparable, if not superior, performance with respect to the remaining optimizers. Also, the adaptive version of the NAG method requires the tuning of only the learning rate γ, which we use in Sections 4.2 and 4.3, where the momentum term is automatically updated as a function of the iteration number.

We run four image classification tasks; namely, MNIST, QMNIST, CIFAR-10, and CIFAR-100. The MNIST dataset is a 10-class image classification dataset composed of 60,000 and 10,000 training and testing gray-scale images of hand-written digits, respectively. QMNIST extends MNIST’s testing set to 60,000 testing images. The CIFAR-10 and CIFAR-100 datasets consist of 50,000 training images and 10,000 testing images with dimensions 32 × 32 with 10 and 100 classes, respectively. In these tasks, we show that the proposed optimizer is very well-suited for deep neural networks. We compare our proposed optimizer with diffGrad, SGDm, NAG, Adam, and AdamW. The hyper-parameters for the optimizers tuned in-house are chosen by conducting a search over the values {1, 0.1, 0.5, 0.01, 0.05, 0.001, 0.005, 0.0001, 0.0005}, and choosing the value that returns the highest validation accuracy. We run all networks for 200 epochs.

MNIST and QMNIST: The neural network used is the conventional convolutional neural network (CNN) as designed in

Table 4
Average test accuracy for MNIST, QMNIST, CIFAR-10, and CIFAR-100 over last 10 epochs.

Optimizer	MNIST	QMNIST	CIFAR-10	CIFAR-100
Ours	98.94% ±0.04%	98.72% ±0.02%	92.97% ±0.08%	**73.19% ±0.14%**
diffGrad	**99.17% ±0.02%**	**98.98% ±0.01%**	92.88% ±0.17%	70.69% ±0.43%
SGDm	99.03% ±0.02%	98.81% ±0.02%	84.16% ±1.44%	59.58% ±1.16%
NAG	99.13% ±0.01%	98.95% ±0.01%	**93.03% ±0.17%**	71.83% ±0.24%
Adam	99.17% ±0.02%	98.93% ±0.01%	92.38% ±0.24%	64.79% ±0.35%
AdamW	97.77% ±0.03%	97.61% ±0.04%	87.64% ±1.28%	63.95% ±0.78%

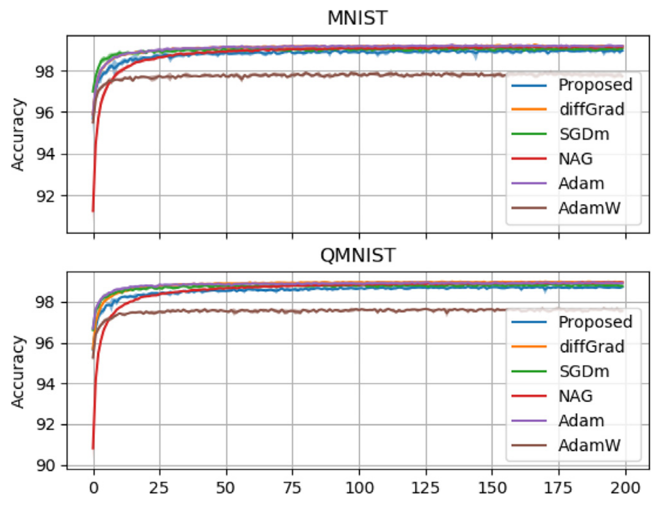

Fig. 4. Average test set accuracy over five independent runs of MADAGRAD and other popular first-order gradient descent optimizers on MNIST (top) and QMNIST (bottom).

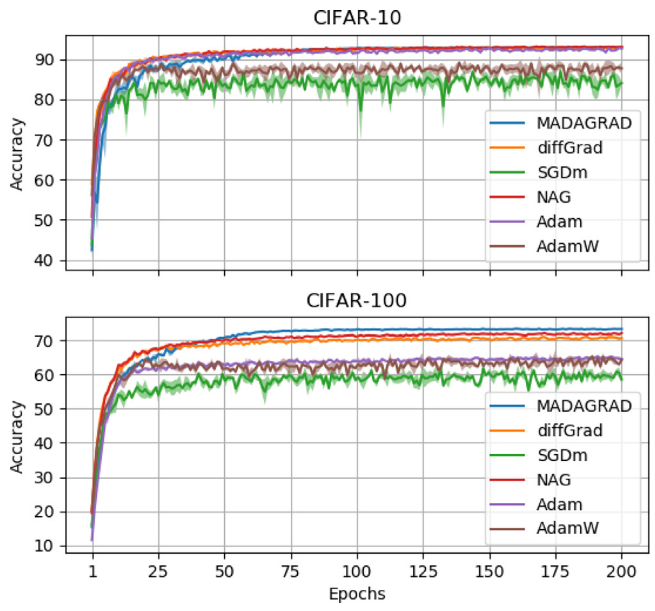

Fig. 5. Average test set accuracy over three independent runs of MADAGRAD and other popular first-order gradient descent optimizers on CIFAR-10 (top) and CIFAR-100 (bottom).

Koehler (2020), which includes two convolutional layers with kernel size 5, one fully-connected hidden layer, and a proceeding fully-connected classification layer of 50 neurons. The activation function chosen is the ReLU function. We run our networks over 5 random (seeds) initializations of the network parameters, and use a batch size of 64. The learning rate chosen for diffGrad is 0.001 with the standard values of $\beta_1 = 0.9$ and $\beta_2 = 0.99$. For SGDm, we choose a learning rate value of 0.01, with $\beta = 0.9$. For Adam, we choose a learning rate of 0.0005 with $\beta_1 = 0.9$ and $\beta_2 = 0.99$. For NAG, we use a learning rate of 0.01 and $\beta = 0.9$. Lastly, for AdamW, we choose a learning rate of 0.0005 with $\beta_1 = 0.9$ and $\beta_2 = 0.99$, and a weight decay value of 1.

CIFAR-10 and CIFAR-100: We adopt the tuning parameters set forth by Zhang, Lucas, Ba, and Hinton (2019) for SGDm and AdamW, and similarly run the proposed CIFAR experiments using a Resnet-18 (He, Zhang, Ren, & Sun, 2016) for three different seeds using a batch size of 128. Additionally, diffGrad and Adam are tuned in-house. We tune all optimizers on CIFAR-10, then use the same parameters on CIFAR-100. We use the same hyper-parameters for diffGrad from the MNIST experiments. We run SGDm with $\beta = 0.9$, learning rate of 0.05, and weight decay value of 0.001. AdamW has a learning rate of 0.0003 and weight decay value of 1. For NAG, we use a learning rate of 0.05 and $\beta = 0.9$. For Adam, we choose a learning rate value of 0.005.

The results are reported in Table 4 and illustrated in Fig. 4 and Fig. 5. Our optimizer achieves comparable results to the popular optimizers on MNIST, QMNIST, and CIFAR-10. It achieves the highest average accuracy of 73.19% over the last 10 epochs for CIFAR-100 whereas the second best average accuracy is 71.83%. In addition, whenever we take the average of the test accuracy over the four datasets for each optimizer, we find that our optimizer achieves the highest average of 90.995%, the second best is NAG with an average of 90.735%, the third is diffGrad with an average of 90.43%, and the rest is below 89%. We reiterate that no tuning was implemented on the two hyper-parameters of MADAGRAD. The learning rate $\gamma = 1$ and lower bound limiter of (3) $\epsilon = 0.05$ were fixed for all image classification tasks rather than fine tuning these two hyper-parameters for each dataset.

Remark. We point out that learning-rate scheduling may yield better performance, such as 95% test accuracy on CIFAR-10 using SGDm (Lang, Zhang, & Xiao, 2019). We reiterate that our main objective is significant tuning reduction while achieving comparable performance with the state-of-the-art optimizers. Thus, for fair comparison, we compare our method with other optimizers while only tuning their hyper-parameters.

5. Conclusion

This paper has proposed a novel computationally efficient multivariate adaptive gradient-descent method for large-scale systems such as deep neural networks. In the deterministic setting, the proposed method ensures global linear convergence to the solution by assuming a strongly convex cost function that is continuously differentiable with Lipschitz gradients. Under a stochastic setting, the proposed method converges in expectation at the order of $\mathcal{O}(1/\sqrt{k})$ for a non-convex cost function with Lipschitz continuous gradient. It is noted that the gradient is *not* assumed to be bounded under both settings. However, it has been shown that after T iterates, the cost function of the last iterate scales as $\mathcal{O}(\log(T)/T)$ for non-smooth strongly convex cost functions while assuming bounded subgradients.

The proposed method has been empirically validated on smooth and strongly convex function, convex with non-twice-differentiable gradients, a smooth non-convex function, non-smooth convex function, and four image classification tasks using deep neural networks. The optimizer has displayed superior convergence properties on all toy examples. Finally, it is shown that the algorithm achieves comparable results, without tuning, to the state-of-the-art on MNIST, QMNIST, and CIFAR-10, and better performance on CIFAR-100. In fact, the proposed multivariable adaptive gradient descent method with reduced tuning efforts yielded the best overall performance results for deep neural networks when comparing it to the state-of-the-art tuned optimizers such as NAG, diffGrad, Adam and SGDm.

CRediT authorship contribution statement

Samer Saab Jr: Conceptualization of this study, Methodology, theory, Software, Writing – original draft. **Khaled Saab:** Conceptualization of this study, Methodology, Data curation, Software, Theory. **Shashi Phoha:** Coordinator. **Minghui Zhu:** Supervised and organized the course of the article. **Asok Ray:** Supervision and organization the course of the article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

The work has been supported by the Walker Fellowship from the Applied Research Laboratory at the Pennsylvania State University and Wu Tsai Neuroscience Stanford Interdisciplinary Graduate Fellowship (SIGF). The work reported here has been supported in part by NSF CAREER award ECCS-1846706, and by the U.S. Air Force Office of Scientific Research (AFOSR) under Grant No. FA9550-15-1-040 in the area of dynamic data-driven application systems (DDDAS). Any opinions, findings and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

Ethics approval

The study does not involve human subjects and/or animals.

Code availability

The codes will be made available upon acceptance of the manuscript.

Data availability

The data is based on open-source repository.

Appendix

Theorem 4. Assume that $f(x)$ is a non-smooth μ-strongly convex function, $\mathbb{E}[\|\mathscr{G}_k\|^2] \leq G^2\ \forall k$, and $f(x)$ attains a minimum at some $x^* \in \mathcal{X}$. By setting $\gamma_k = \frac{1}{\mu k}$ and $x_0 = 0$, then for any $T > 1$, the sequence $\{x_k\}$ generated by the algorithm described by (15), satisfies

$$\mathbb{E}[f(x_T) - f(x^*)] \leq \frac{17G^2(1+\log(T))}{\mu T}.$$

Proof. The proof of Theorem 4 follows similar steps as the proof of Theorem 1 in Shamir and Zhang (2013) for SGD.

Since the elements of the subgradient are implicitly assumed to be unbiased (in expectation) and the elements of χ_k are positive and bounded by 1, then the elements of $\chi_k \odot \mathscr{G}_k$ are unbiased. In addition, since $x_0 = 0$, hence the elements of $x_k - x_{k-1}$ (15) are also unbiased. At a later stage of the proof x is substituted with x_{k-m}. Therefore, we have

$$\mathbb{E}[\langle \chi_k \odot \mathscr{G}_k, x_k - x\rangle] = \mathbb{E}[\langle \mathscr{G}_k, x_k - x\rangle].$$

In addition, since $\gamma_k > 0$, then

$$-2\gamma_k\mathbb{E}[\langle \chi_k \odot \mathscr{G}_k, x_k - x\rangle] = -2\gamma_k\mathbb{E}[\langle \mathscr{G}_k, x_k - x\rangle].$$

By convexity of $\mathcal{X}$, we have

$$\begin{aligned}\mathbb{E}[||x_{k+1} - x||^2] &= \mathbb{E}[||\Pi_{\mathcal{X}}(x_k - \gamma_k\chi_k \odot \mathscr{G}_k) - x||^2] \\ &\leq \mathbb{E}[||x_k - x||^2] - 2\gamma_k\mathbb{E}[\langle g_k, x_k - x\rangle] + \gamma_k^2 G^2,\end{aligned}$$

where $g_k \triangleq \mathbb{E}[\mathscr{G}_k]$ is a subgradient of f at x_k. The last term on the right-hand side of the above equation uses the fact that $|(\chi_k)_i| \leq 1$. Consequently,

$$\mathbb{E}[\langle g_k, x_k - x\rangle] \leq \frac{1}{2\gamma_k}\Big(\mathbb{E}[\|x_k - x\|^2] - \mathbb{E}[\|x_{k+1} - x\|^2]\Big) + \frac{\gamma_k G^2}{2}.$$

Let m be an arbitrary element in $\{1, \ldots, [T/2]\}$. Summing the above inequality over all $k = T-m, \ldots, T$, and cancelling common terms, we obtain

$$\begin{aligned}\sum_{k=T-m}^{T} \mathbb{E}[\langle g_k, x_k - x\rangle] \leq{}& \frac{1}{2\gamma_{T-m}}\mathbb{E}[\|x_{T-m} - x\|^2] \\ &+ \sum_{k=T-m+1}^{T} \frac{\mathbb{E}[\|x_{T-m} - x\|^2]}{2}\Big(\frac{1}{\gamma_k} - \frac{1}{\gamma_{k-1}}\Big) \\ &+ \frac{G^2}{2}\sum_{k=T-m}^{T}\gamma_k.\end{aligned} \tag{20}$$

Since $f(x)$ is μ-strongly convex, then from Definition 2, we get

$$\mathbb{E}[f(x) - f(x_k)] \geq \mathbb{E}[\langle x - x_k, g_k\rangle] + \frac{\mu}{2}\mathbb{E}[\|x - x_k\|^2],$$

or

$$\mathbb{E}[f(x_k) - f(x)] \leq \mathbb{E}[\langle x_k - x, g_k\rangle] - \frac{\mu}{2}\mathbb{E}[\|x_k - x\|^2]. \tag{21}$$

Plugging the above inequality into (20) while substituting $\gamma_k = \frac{1}{\mu k}$, we obtain

$$\begin{aligned}\sum_{k=T-m}^{T} \mathbb{E}[f(x_k) - f(x)] \leq{}& \frac{\mu(T-m)}{2}\mathbb{E}[\|x_{T-m} - x\|^2] \\ &- \sum_{k=T-m+1}^{T} \frac{\mathbb{E}[\|x_{T-m} - x\|^2]}{2}\Big(\frac{1}{\mu(k-1)}\Big) \\ &+ \frac{G^2}{2\mu}\sum_{k=T-m}^{T}\frac{1}{k} + \frac{\mu}{2}\sum_{k=T-m+1}^{T}\mathbb{E}[\|x_k - x\|^2].\end{aligned}$$

Since the second term on the right-hand side is negative, then we drop it and obtain

$$\begin{aligned}\sum_{k=T-m}^{T} \mathbb{E}[f(x_k) - f(x)] \leq{}& \frac{\mu(T-m)}{2}\mathbb{E}[\|x_{T-m} - x\|^2] \\ &+ \frac{\mu}{2}\sum_{k=T-m+1}^{T}\mathbb{E}[\|x_k - x\|^2] + \frac{G^2}{2\mu}\sum_{k=T-m}^{T}\frac{1}{k}.\end{aligned} \tag{22}$$

Note that using (21), we have $\mathbb{E}[\langle x_k - x^*, g_k\rangle] \geq \frac{\mu}{2}\mathbb{E}[\|x_k - x^*\|^2]$. Using the latter inequality, Lemma 1 in Shamir (2011) shows that $\mathbb{E}[\|x_k - x^*\|^2] \leq \frac{4G^2}{\mu^2 k}$. This implies that for $k \geq T - m$,

$$\begin{aligned}\mathbb{E}[\|x_k - x_{T-m}\|^2] &\leq 2\mathbb{E}[\|x_k - x^*\|^2 + \|x_{T-m} - x^*\|^2] \\ &\leq \frac{8G^2}{\mu^2}\Big(\frac{1}{k} + \frac{1}{T-m}\Big) \leq \frac{16G^2}{\mu^2(T-m)} \\ &\leq \frac{32G^2}{\mu^2 T}.\end{aligned}$$

The last inequality is due to the fact that $1 \leq m \leq [T/2]$. Next, we use the above inequality and set $x = x_{T-m}$ in (22) to obtain

$$\begin{aligned}\mathbb{E}\Big[\sum_{k=T-m}^{T}(f(x_k) - f(x_{T-m}))\Big] &\leq \frac{\mu}{2}\sum_{k=T-m+1}^{T}\mathbb{E}[\|x_k - x_{T-m}\|^2] \\ &+ \frac{G^2}{2\mu}\sum_{k=T-m}^{T}\frac{1}{k} \leq \frac{16G^2 m}{\mu T} + \frac{G^2}{2\mu}\sum_{k=T-m}^{T}\frac{1}{k} \\ &= \frac{G^2}{2\mu}\Big(\frac{32m}{T} + \sum_{k=T-m}^{T}\frac{1}{k}\Big).\end{aligned}$$

We omit the rest of the proof since it follows the same steps as in the proof of Theorem 1 in Shamir and Zhang (2013).

References

Alecsa, Cristian Daniel, Pinţa, Titus, & Boros, Imre (2020). New optimization algorithms for neural network training using operator splitting techniques. *Neural Networks*, *126*, 178–190.

Bach, Francis, & Moulines, Eric (2011). Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Neural information processing systems (NIPS)*.

Bertsekas, Dimitri P. (1997). A new class of incremental gradient methods for least squares problems. *SIAM Journal on Optimization*, *7*(4), 913–926.

Cohen, Jeremy M., Kaur, Simran, Li, Yuanzhi, Kolter, J. Zico, & Talwalkar, Ameet (2021). Gradient descent on neural networks typically occurs at the edge of stability. arXiv preprint arXiv:2103.00065.

Dubey, Shiv Ram, Chakraborty, Soumendu, Roy, Swalpa Kumar, Mukherjee, Snehasis, Singh, Satish Kumar, & Chaudhuri, Bidyut Baran (2019). diffGrad: an optimization method for convolutional neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(11), 4500–4511.

Duchi, John, Hazan, Elad, & Singer, Yoram (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*(7).

Gaivoronski, Alexei A. (1994). Convergence properties of backpropagation for neural nets via theory of stochastic gradient methods. Part 1. *Optimization Methods & Software*, *4*(2), 117–134.

Ghadimi, Euhanna, Feyzmahdavian, Hamid Reza, & Johansson, Mikael (2015). Global convergence of the heavy-ball method for convex optimization. In *2015 European control conference (ECC)* (pp. 310–315). IEEE.

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

Gower, Robert Mansel, Loizou, Nicolas, Qian, Xun, Sailanbayev, Alibek, Shulgin, Egor, & Richtárik, Peter (2019). SGD: General analysis and improved rates. In *International conference on machine learning* (pp. 5200–5209). PMLR.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, & Sun, Jian (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Jamil, Momin, & Yang, Xin-She (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, *4*(2), 150–194.

Karimi, Hamed, Nutini, Julie, & Schmidt, Mark (2016). Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 795–811). Springer.

Khan, Mohammad, Nielsen, Didrik, Tangkaratt, Voot, Lin, Wu, Gal, Yarin, & Srivastava, Akash (2018). Fast and scalable bayesian deep learning by weight-perturbation in adam. In *International conference on machine learning* (pp. 2611–2620). PMLR.

Kingma, Diederik P., & Ba, Jimmy (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Koehler, Gregor (2020). Mnist handwritten digit recognition in pytorch. URL: https://nextjournal.com/gkoehler/pytorch-mnist.

Lang, Hunter, Zhang, Pengchuan, & Xiao, Lin (2019). Using statistics to automate stochastic optimization. arXiv preprint arXiv:1909.09785.

Lei, Yunwen, Hu, Ting, Li, Guiying, & Tang, Ke (2019). Stochastic gradient descent for nonconvex learning without bounded gradient assumptions. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(10), 4394–4400.

Lessard, Laurent, Recht, Benjamin, & Packard, Andrew (2016). Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, *26*(1), 57–95.

Li, Wenjing, & Bian, Wei (2021). Smoothing neural network for L0 regularized optimization problem with general convex constraints. *Neural Networks*, *143*, 678–689.

Li, Xiaoyu, & Orabona, Francesco (2019). On the convergence of stochastic gradient descent with adaptive stepsizes. In *The 22nd international conference on artificial intelligence and statistics* (pp. 983–992). PMLR.

Lin, Junhong, & Zhou, Ding-Xuan (2017). Online learning algorithms can converge comparably fast as batch learning. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(6), 2367–2378.

Loizou, Nicolas, Vaswani, Sharan, Laradji, Issam, & Lacoste-Julien, Simon (2020). Stochastic polyak step-size for SGD: An adaptive learning rate for fast convergence. arXiv preprint arXiv:2002.10542.

Luo, Liangchen, Xiong, Yuanhao, Liu, Yan, & Sun, Xu (2019). Adaptive gradient methods with dynamic bound of learning rate. arXiv preprint arXiv:1902.09843.

Needell, Deanna, Srebro, Nathan, & Ward, Rachel (2016). Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. *Mathematical Programming*, *155*(1–2), 549–573.

Needell, Deanna, & Ward, Rachel (2016). Batched stochastic gradient descent with weighted sampling. In *International conference approximation theory* (pp. 279–306). Springer.

Nesterov, Yu (1983). A method of solving a convex programming problem with convergence rate o (1/kˆ 2) o (1/k2). In *Sov. math. dokl., Vol. 27* (p. 2).

Nesterov, Yurii (2003). *Introductory lectures on convex optimization: A basic course, Vol. 87*. Springer Science & Business Media.

Probst, Philipp, Boulesteix, Anne-Laure, & Bischl, Bernd (2019). Tunability: Importance of hyperparameters of machine learning algorithms. *Journal of Machine Learning Research*, *20*(53), 1–32.

Qian, Ning (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, *12*(1), 145–151.

Robbins, Herbert, & Monro, Sutton (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 400–407.

Saab, Samer S., & Shen, Dong (2019). Multidimensional gains for stochastic approximation. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(5), 1602–1615.

Shamir, Ohad (2011). Making gradient descent optimal for strongly convex stochastic optimization. CoRR, abs/1109.5647.

Shamir, Ohad, & Zhang, Tong (2013). Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International conference on machine learning* (pp. 71–79). PMLR.

Tao, Wei, Long, Sheng, Wu, Gaowei, & Tao, Qing (2021). The role of momentum parameters in the optimal convergence of adaptive polyak's heavy-ball methods. arXiv preprint arXiv:2102.07314.

Vaswani, Sharan, Mishkin, Aaron, Laradji, Issam, Schmidt, Mark, Gidel, Gauthier, & Lacoste-Julien, Simon (2019). Painless stochastic gradient: Interpolation, line-search, and convergence rates. arXiv preprint arXiv:1905.09997.

Wilson, Ashia C., Roelofs, Rebecca, Stern, Mitchell, Srebro, Nathan, & Recht, Benjamin (2017). The marginal value of adaptive gradient methods in machine learning. arXiv preprint arXiv:1705.08292.

Xu, Dongpo, Zhang, Shengdong, Zhang, Huisheng, & Mandic, Danilo P. (2021). Convergence of the rmsprop deep learning method with penalty for nonconvex optimization. *Neural Networks*, *139*, 17–23.

Yang, Tianbao, Lin, Qihang, & Li, Zhe (2016). Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. arXiv preprint arXiv:1604.03257.

Zeiler, Matthew D. (2012). Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.

Zhang, Michael, Lucas, James, Ba, Jimmy, & Hinton, Geoffrey E. (2019). Lookahead optimizer: k steps forward, 1 step back. In *Advances in neural information processing systems* (pp. 9597–9608).

Zou, Fangyu, Shen, Li, Jie, Zequn, Zhang, Weizhong, & Liu, Wei (2019). A sufficient condition for convergences of adam and rmsprop. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11127–11135).