Proceedings of Machine Learning Research vol 145:1–22, 2021 2nd Annual Conference on Mathematical and Scientific Machine Learning

A Qualitative Study of the Dynamic Behavior for Adaptive Gradient Algorithms

Chao Ma CHAOMA@STANFORD.EDU
Department of Mathematics, Stanford University

Lei Wu LEIWU@PRINCETON.EDU*
Program in Applied and Computational Mathematics, Princeton University

Weinan E WEINAN@MATH.PRINCETON.EDU
Program in Applied and Computational Mathematics and Department of Mathematics, Princeton University

Editors: Joan Bruna, Jan S Hesthaven, Lenka Zdeborova

Abstract

The dynamic behavior of RMSprop and Adam algorithms is studied through a combination of careful numerical experiments and theoretical explanations. Three types of qualitative features are observed in the training loss curve: fast initial convergence, oscillations, and large spikes in the late phase. The sign gradient descent (signGD) flow, which is the limit of Adam when taking the learning rate to 0 while keeping the momentum parameters fixed, is used to explain the fast initial convergence. For the late phase of Adam, three different types of qualitative patterns are observed depending on the choice of the hyper-parameters: oscillations, spikes, and divergence. In particular, Adam converges much smoother and even faster when the values of the two momentum factors are close to each other. This observation is particularly important for scientific computing tasks, for which the training process usually proceeds into the high precision regime.

Keywords: Dynamical behavior; Adaptive gradient algorithm; Sign gradient descent; Adam optimizer.

1. Introduction

Adaptive gradient algorithms (Duchi et al., 2011; Tieleman and Hinton, 2012; Kingma and Ba, 2014), in particular RMSprop (Tieleman and Hinton, 2012) and Adam (Kingma and Ba, 2014), have demonstrated superior performance in training modern machine learning models, e.g. deep neural networks. Distinguished from the vanilla gradient descent (GD) or stochastic gradient descent (SGD), adaptive gradient algorithms use a coordinate-wise scaling of the update direction. The scaling factors are adaptively determined by using the history of past gradients (Duchi et al., 2011), which makes the understanding and analysis of these algorithms much more challenging.

Recent theoretical efforts (Reddi et al., 2018; Zhou et al., 2018; Xie et al., 2020; Li and Orabona, 2019; Chen et al., 2018) have focused on establishing the convergence of adaptive gradient algorithms. However, these results are still unsatisfactory, since they cannot explain any of the particular features of these adaptive gradient algorithms. Moreover, all these results require taking the limit that the learning rate η_t goes to zero, e.g. $\eta_t = 1/\sqrt{t}$. However, in practice, one usually starts with a large learning rate and only decays the learning rate several times during the training process. For

* The first two authors contributed equally.

© 2021 C. Ma, L. Wu & W. E.

the most of iterations, the learning rate is fixed. So it is interesting to see what happens when the learning rate is fixed. Figure 1 shows the dynamical behavior of full-batch Adam with a fixed learning rate for one classification problem and one regression problem, respectively. For both cases, one can see that the training curve does not behave monotonically even for this full batch setting. Large spikes keep repeatedly appearing in the late phase of the training.

These spikes may not cause serious problems for typical computer vision and NLP tasks. For these applications, the data are either highly noisy or very limited, the learning is mainly dominated by the generalization gap (as shown in the left panel of Figure 1). Typically, a training loss $10^{-2} \sim 10^{-3}$ is sufficient to select a good model. However, for many scientific computing tasks, such as fitting a target function and numerically solving PDEs, we are more interested in high-precision solutions. For these problems, the training data are relatively clean and easy to obtain, which reduces the risk of overfitting. Hence, lower training loss is always desired. The large spikes in the late phase make it difficult to pick a good stopping time, as shown in the right panel of Figure 1.

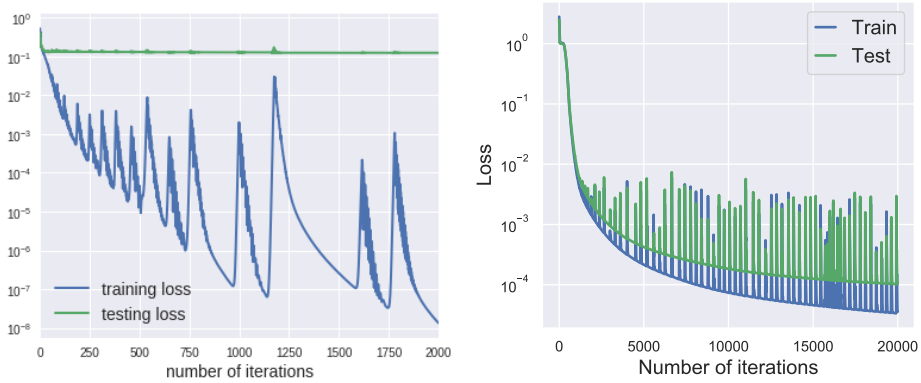


Figure 1: The training curves of full batch Adam. For both cases, the learning rate is fixed to be 0.001 and $(\beta_1, \beta_2) = (0.9, 0.999)$, the default values in PyTorch and TensorFlow. **Left:** Classify CIFAR-10 dataset with a fully-connected neural networks. The network has 3 hidden layers with widths 256-256-128. 2 classes are picked from CIFAR-10 with 1000 images in each class. Square loss function is used. **Right:** Fit the target function $f^*(\boldsymbol{x}) = \sum_{k=1}^{5} \sin(2\pi x_k)$ for $\boldsymbol{x} \in [0,1]^5$ with a fully connected network, whose architecture is 5-100-100-1. 2000 points are uniformly sampled from Unif$([0,1]^5)$ to form the training set

.

In addition to the learning rate, adaptive gradient algorithms also use extra hyper-parameters such as the second-order momentum factor for RMSprop and the first and second order momentum factors for Adam. Default values of these hyper-parameters are provided in mainstream packages (e.g. β_1=0.9, β_2=0.999 for Adam in PyTorch and TensorFlow), which are usually tuned on the classification problems with cross-entropy loss (Kingma and Ba, 2014). However, they are not necessarily optimal for regression and scientific computing problems, where the quadratic loss is used, e.g., the default values can cause the large spikes as shown in Figure 1. One objective of this paper is to carry out a comprehensive study of how the choice of these hyper-parameters affects the dynamical behavior.

Contributions In this paper, we provide well-designed experiments to demystify the dynamic behavior of adaptive gradient algorithms. Specifically, our contributions are summarized as follows.

1. We identify three types of typical phenomena in the training process of these adaptive algorithms: initial fast convergence (sometimes even super-linear), small oscillations, and large spikes in the late phase.

2. For RMSprop and Adam, if the learning rate decreases to zero while the momentum parameters are fixed, the algorithms converge to the signGD flow. For signGD flow, we prove the finite-time convergence for objective functions that satisfy the Polyak-Lojasiewicz (PL) (Polyak, 1963) condition. These arguments together provide a partial explanation of the fast initial convergence of RMSprop and Adam, which could be the one of reasons behind the popularity of these algorithms.

3. We show that the large spikes are caused by some instabilities of the algorithm at stationary points. For RMSprop on simple objective functions, we explicitly write down the limiting oscillating solution. For Adam, we classify the behavior into three different patterns in the space of the two momentum factors: the spike regime, the oscillation regime, and the divergence regime. Empirical results show that training is most stable in the "oscillation regime",in particular when $\beta_1 \approx \beta_2$.

Throughout this paper, all the activation functions are $\sigma(t) = \max(0, t)$, unless explicitly specified. The quadratic loss is used for all the experiments, including the classification problems. Other experimental details are described in the caption of each figure.

To make the notations more consistent, from now on we use α to denote the second-order momentum factor in both Adam and RMSprop, and use β to denote the first-order momentum in Adam. The conventional notations β_1 and β_2 for Adam will become β and α, respectively. For vectors $\boldsymbol{u}$ and $\mathbf{v}$, operations such as $\boldsymbol{u}^2$, $\sqrt{\boldsymbol{u}}$, $\boldsymbol{u}/\mathbf{v}$, and $|\boldsymbol{u}|$ are understood to be element-wise.

2. Preliminaries

2.1. Adaptive gradient algorithms

Adaptive gradient algorithms are a family of optimization algorithms that use a coordinate-wise scaling of the update direction (gradient or gradient with momentum) according to the history of gradients. Many adaptive algorithms can be cast to the following form (da Silva and Gazeau, 2018),

$$\begin{aligned} \mathbf{m}_{t+1} &= h_t \nabla f(\boldsymbol{x}_t) + r_t \mathbf{m}_t \\ \mathbf{v}_{t+1} &= p_t (\nabla f(\boldsymbol{x}_t))^2 + q_t \mathbf{v}_t \\ \boldsymbol{x}_{t+1} &= \boldsymbol{x}_t - \eta_t \frac{\mathbf{m}_{t+1}}{\sqrt{\mathbf{v}_{t+1}} + \epsilon}, \end{aligned} \tag{1}$$

with different choice of h, r, p, q. In (1), h, r, p, q are scalar functions of t. For example, Adagrad (Duchi et al., 2011) is recovered when $h, p, q = 1$ and $r = 0$, and RMSprop corresponds to the case when $h = 1$, $r = 0$, $p = 1 - \alpha$ and $q = \alpha$ for some constant $\alpha \in (0, 1)$. Viewed from the dynamics of $\boldsymbol{x}_t$ alone, adaptive gradient algorithms usually have a "memory effect" due to the momentum terms. The strength of the memory depends on the momentum factors (h_t, r_t, p_t, q_t) and the learning rate η_t. Because of their efficiency in training neural network models, these algorithms are extensively used. We refer readers to (Ruder, 2016) for a more thorough review of existing adaptive algorithms.

In this paper, we focus on RMSprop and Adam — the two algorithms that are most widely used by practitioners. The discrete update rules of these algorithms are

- **RMSprop:**

$$\begin{aligned} \mathbf{v}_{t+1} &= \alpha \mathbf{v}_t + (1-\alpha)(\nabla f(\boldsymbol{x}_t))^2 \\ \boldsymbol{x}_{t+1} &= \boldsymbol{x}_t - \eta \frac{\nabla f(\boldsymbol{x}_t)}{\sqrt{\mathbf{v}_{t+1}} + \epsilon} \end{aligned} \tag{2}$$

- **Adam:**

$$\begin{aligned} \mathbf{v}_{t+1} &= \alpha \mathbf{v}_t + (1-\alpha)(\nabla f(\boldsymbol{x}_t))^2 \\ \mathbf{m}_{t+1} &= \beta \mathbf{m}_t + (1-\beta)\nabla f(\boldsymbol{x}_t) \\ \boldsymbol{x}_{t+1} &= \boldsymbol{x}_t - \eta \frac{\mathbf{m}_{t+1}/(1-\beta^{t+1})}{\sqrt{\mathbf{v}_{t+1}/(1-\alpha^{t+1})} + \epsilon} \end{aligned} \tag{3}$$

In (2) and (3), ϵ is a small constant used to avoid the division by 0. It is usually taken to be 10^{-8}.

In this paper, we mainly focus on the full batch setting, i.e., $\nabla f(\boldsymbol{x}_t)$ is the full gradient, for which the dynamical behavior is already rather complex. We also show that our observations of the full batch setting also apply to the stochastic setting if the batch size is relatively large. The systematic investigation of the influence of batch size is left to future work.

2.2. Continuous-time limits

RMSProp and Adam can be studied by considering the limiting ordinary differential equations (ODE) obtained by taking the learning rate η to 0. However, different limiting ODEs are obtained when the hyper-parameters are scaled differently.

If the momentum factors are kept fixed, then as $\eta \to 0$, the memory effect diminishes, because in each discrete iteration we lose the same amount of memory but one iteration occupies a shorter and shorter time. In this case, the continuous-time limit for both RMSprop and Adam are the following dynamics

$$\dot{\boldsymbol{x}} = -\frac{\nabla f(\boldsymbol{x})}{|\nabla f(\boldsymbol{x})| + \epsilon}. \tag{4}$$

Since ϵ is a small value, this dynamics is close to the signGD flow:

$$\dot{\boldsymbol{x}} = -\text{sign}(\nabla f(\boldsymbol{x})). \tag{5}$$

Proposition 1 *Assume that ∇f is bounded and Lipschitz continuous, i.e. there exists constants M and L such that $\|\nabla f(\boldsymbol{x}_1)\| \le M$ and $\|\nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2)\| \le L\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$ hold for any $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Let $\{\boldsymbol{x}_k^\eta\}$, $k = 0, 1, 2, \cdots$ be the solution given by algorithm (2) or (3) starting from $\boldsymbol{x}_0$, $\mathbf{m}_0$ and $\mathbf{v}_0 \ge 0$, with learning rate η and some fixed $\alpha, \beta \in (0, 1)$ and $\epsilon > 0$. Let $\boldsymbol{X}^\eta(\cdot)$ be a piece-wise constant function of $t \in [0, \infty)$ that satisfies*

$$\boldsymbol{X}^\eta(t) = \boldsymbol{x}_k^\eta, \quad for\, t \in [k\eta, (k+1)\eta).$$

In addition, let $\boldsymbol{x}(\cdot)$ be the solution of (4) initialized from $\boldsymbol{x}_0$. Then, for any $T > 0$, we have

$$\lim_{\eta \to 0} \sup_{t \in [0,T]} \|\boldsymbol{X}^\eta(t) - \boldsymbol{x}(t)\| = 0. \tag{6}$$

The proof of the proposition is given in the appendix. Figure 2 provides numerical evidence that RMSprop and Adam are close to signGD in a finite time interval when η is small while α and β are fixed. The closeness between signGD and RMSprop is also shown in Figure 5 for a synthetic objective function.

Note that using the signGD method to train neural networks can date back to (Riedmiller and Braun, 1992), termed as Rprop algorithm. RMSProp was initially proposed as a stochastic version of Rprop (Tieleman and Hinton, 2012). This type of connection was also investigated for Adam in (Balles and Hennig, 2018). In contrast, Proposition 1 shows another type of connection: RMSProp/Adam converges to signGD flow in the limit $\eta \to 0$. This connection does not rely on the stochastic approximation and has not been explored before.

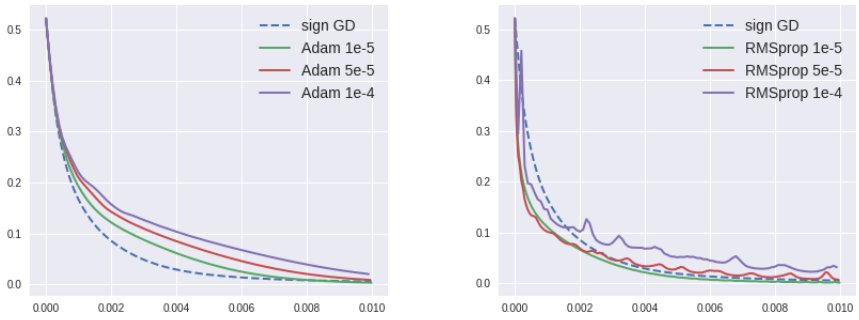


Figure 2: The comparison of the training loss curve between signGD and Adam/RMSprop with different learning rates for the early phase. The x-axis is the time (learning rate$\times$number of iterations) and y-axis denotes the training loss. For Adam, $\beta = 0.9$ and $\alpha = 0.999$; for RMSprop $\alpha = 0.99$. Learning rate of signGD is 10^{-5}. Experiments conducted on a fully-connected neural network with three hidden layers with widths be $256 - 128 - 64$. The training data is taken from 2 classes of CIFAR10 with 1000 samples per class.

On the other hand, if we want to keep the strength of the memory effect fixed, we have to let α and β go to 1 when η tends to 0. Specifically, let $\alpha = 1 - a\eta$ and $\beta = 1 - b\eta$, with a and b being positive constants. Then, it is easy to show that the trajectories of (2) and (3) converge to the following ODEs (7) and (8), respectively.

- **RMSprop flow:**

$$\begin{aligned} \dot{\mathbf{v}} &= a(\nabla f(\boldsymbol{x})^2 - \mathbf{v}) \\ \dot{\boldsymbol{x}} &= -\frac{\nabla f(\boldsymbol{x})}{\sqrt{\mathbf{v}} + \epsilon} \end{aligned} \tag{7}$$

- **Adam flow:**

$$\begin{aligned} \dot{\mathbf{v}} &= a(\nabla f(\boldsymbol{x})^2 - \mathbf{v}) \\ \dot{\mathbf{m}} &= b(\nabla f(\boldsymbol{x}) - \mathbf{m}) \\ \dot{\boldsymbol{x}} &= -\frac{(1 - e^{-bt})^{-1}\mathbf{m}}{\sqrt{(1 - e^{-at})^{-1}\mathbf{v}} + \epsilon} \end{aligned} \tag{8}$$

The following proposition is a simplification of Theorem 3.2 in (Barakat and Bianchi, 2018). Note that this result actually holds for stochastic RMSprop and Adam algorithms, but we will focus on the full-batch setting.

Proposition 2 *Under the same condition of f in Proposition 1, let $\{\boldsymbol{x}_k^\eta\}$, $k = 0, 1, 2, \cdots$ be the solution given by algorithm (2) starting from $\boldsymbol{x}_0$ and $\mathbf{v}_0 = 0$, with learning rate η and $\alpha = 1 - a\eta$ for a fixed constant $a > 0$. Let $\boldsymbol{X}^\eta(\cdot)$ be a piece-wise constant vector function of $t \in [0, \infty)$ that satisfies*

$$\boldsymbol{X}^\eta(t) = \boldsymbol{x}_k^\eta, \quad \text{for } t \in [k\eta, (k+1)\eta).$$

In addition, let $\boldsymbol{x}(\cdot)$ be the solution of (7) initialized from $\boldsymbol{x}_0$ and $\mathbf{v}_0 \geq 0$. Then, for any $T > 0$, we have

$$\lim_{\eta \to 0} \sup_{t \in [0,T]} \|\boldsymbol{X}^\eta(t) - \boldsymbol{x}(t)\| = 0. \tag{9}$$

Similarly, if $\alpha = 1 - a\eta$ and $\beta = 1 - b\eta$ for some constants $a, b > 0$, then the same convergence statements hold for the solutions of (3) and (8).

As can be seen from (7) and (8), the smaller the value of a and b, the slower the dynamics of $\mathbf{v}$ (and $\mathbf{m}$), and consequently the slower the whole dynamics. Numerical results in Figure 3 confirm this. However, it is worth mentioning that this difference in convergence speed does not manifest at the very beginning of the training process. To understand this, consider the dynamics of Adam (8) with $\mathbf{v}_0 = \mathbf{m}_0 = 0$ and $\epsilon = 0$. When $t \ll 1$, we have

$$\begin{aligned} \mathbf{v}_t &\approx (1 - e^{-at}) \nabla f(\boldsymbol{x}_0)^2, \\ \mathbf{m}_t &\approx (1 - e^{-bt}) \nabla f(\boldsymbol{x}_0). \end{aligned}$$

Hence, we have

$$\dot{\boldsymbol{x}} \approx -\frac{(1 - e^{-bt})^{-1}(1 - e^{-bt}) \nabla f(\boldsymbol{x})}{\sqrt{(1 - e^{-at})^{-1}(1 - e^{-at}) \nabla f(\boldsymbol{x})^2}} \text{sign}(\nabla f(\boldsymbol{x})) \approx \text{sign}(\nabla f(\boldsymbol{x}_0)),$$

which shows that the initial speed of $\boldsymbol{x}$ does not depend on a and b. Rigorously, we have the following proposition, whose proof is deferred to the appendix.

Proposition 3 *Given the same conditions of f in Proposition 1. Let $(\boldsymbol{x}(t), \mathbf{m}(t), \mathbf{v}(t))$ be the solution of (8) starting from $(\boldsymbol{x}_0, 0, 0)$ with $\epsilon = 0$. Assume $|[\nabla f(\boldsymbol{x}_0)]_i| > c$ for some positive constant c, where $[\nabla f(\boldsymbol{x}_0)]_i$ means the i-th element of $\nabla f(\boldsymbol{x}_0)$. Then, for any τ that satisfies $\tau < \frac{c^3}{32M^2L\sqrt{d}}$, we have*

$$\|\dot{\boldsymbol{x}}(\tau) + sign(\nabla f(\boldsymbol{x}(\tau)))\| < \frac{54M^3L\sqrt{d}}{c^4}\tau. \tag{10}$$

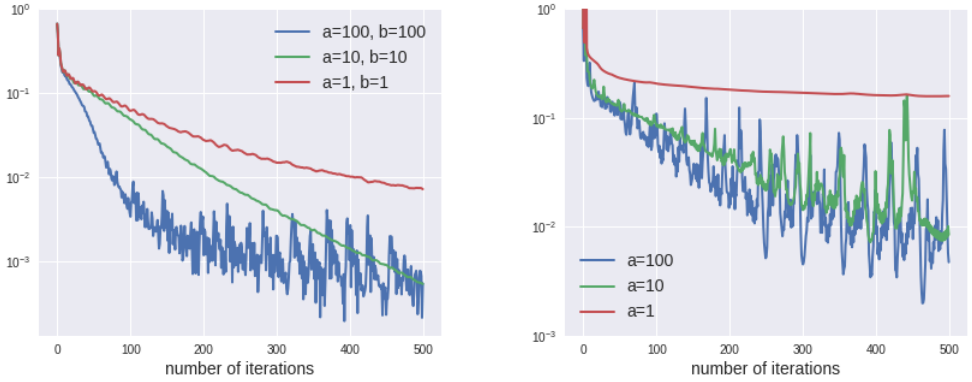


Figure 3: How the values of a and b affect the speed of dynamics. **Left:** Adam; **Right:** RMSprop. The learning rate is 0.001 for all the experiments. The model and training data are the same as Figure 2. One can see that at the early stage of the training (after a very short period from initialization), optimizers with larger a and b converge faster.

3. RMSprop and signGD: Fast convergence and oscillation

In this section, we focus on RMSprop. Figure 4 shows the loss curves and trajectories of RMSprop for a typical multi-layer neural network model. There are three obvious features:

1. **Fast initial convergence:** the loss curve decreases very fast, sometimes even super-linearly, at the early stage of the training.

2. **Small oscillations:** The fast initial convergence is followed by oscillations around the minimum.

3. **Large spikes:** Spikes are the sudden increase of the loss values, which are followed by an oscillating recovery. Different from small oscillations, spikes make the loss much larger and the interval between two spikes is also longer.

We relate the fast initial convergence with the closeness of the RMSprop trajectory to signGD. For the other two features, we attribute them to the instability at the stationary points.

Fast initial convergence As discussed in the last section, when η tends to 0 while α stays fixed, RMSprop tends to signGD. So the loss curve of RMSprop and signGD align well during the initial phase as shown in Figure 2. Figure 5 shows the loss curves of both signGD and RMSprop on a quadratic objective function. Their behaviors are similar — they both experience fast initial convergence and then the loss stops decreasing. For this reason, we will study the fast initial convergence of RMSprop with the help of signGD. Under the PL condition, the following proposition shows that signGD flow can reach the global minimum in finite time.

Proposition 4 *Assume that the objective function satisfies the Polyak-Lojasiewicz (PL) (Polyak, 1963) condition:* $\|\nabla f(\boldsymbol{x})\|_2^2 \geq \mu(f(\boldsymbol{x}) - f(\boldsymbol{x}^*))$ *for any* $\boldsymbol{x}$*. Here* $\boldsymbol{x}^* = argmin_{\boldsymbol{x}} f(\boldsymbol{x})$*. Let* $\boldsymbol{x}(t)$ *be*

Figure 4: The loss curves of training a neural network model on CIFAR-10 dataset with RMSprop. Model and data the same as Figure 1. The learning rate is 1e-3, and $\alpha = 0.99$. 2000 iterations are run. **Left:** The whole training loss curves **Right:** The training loss of the last 500 iterations.

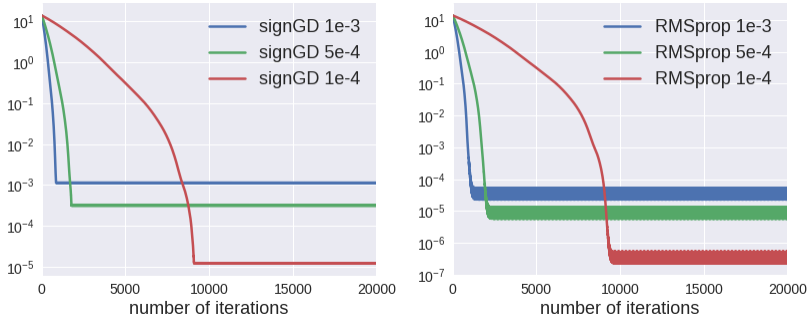


Figure 5: The loss curves of signGD (**Left**) and RMSprop (**Right**) with varying learning rates for a quadratic objective function f. Here, $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x}$. $A = UU^T$ with $U \in \mathbb{R}^{10\times 10}$ and $U_{i,j} \overset{iid}{\sim} \mathcal{N}(0,1)$. For RMSprop, α is fixed to be 0.9.

the solution of the signGD flow (5)*. Then, we have*

$$f(\boldsymbol{x}(t)) - f(\boldsymbol{x}^*) \leq \left(\sqrt{f(\boldsymbol{x}_0) - f(\boldsymbol{x}^*)} - \frac{\sqrt{\mu}}{2}t\right)^2.$$

The signGD flow will stop within $T \leq 2\sqrt{\frac{f(\boldsymbol{x}_0)-f(\boldsymbol{x}^*)}{\mu}}$.

Proof We have

$$\begin{aligned}\frac{d}{dt}f(\boldsymbol{x}(t)) &= -\langle \text{sign}(\nabla f(\boldsymbol{x}(t))), \nabla f(\boldsymbol{x}(t))\rangle = -\|\nabla f(\boldsymbol{x}(t))\|_1 \\ &\leq -\|\nabla f(\boldsymbol{x}(t))\|_2 \leq -\sqrt{\mu(f(\boldsymbol{x}(t)) - f(\boldsymbol{x}^*))}.\end{aligned}$$

Hence, we have $\frac{d}{dt}\sqrt{f(\boldsymbol{x}(t)) - f(\boldsymbol{x}^*)} \leq -\frac{\sqrt{\mu}}{2}$, which implies

$$f(\boldsymbol{x}(t)) - f(\boldsymbol{x}^*) \leq \left(\sqrt{f(\boldsymbol{x}_0) - f(\boldsymbol{x}^*)} - \frac{\sqrt{\mu}}{2}t\right)^2.$$

■

Consider an one-dimensional objective function $f(x) = \varepsilon x^2$ and the signGD flow starting from $x_0 > 0$. The signGD flow is given by $\dot{x}(t) = -1$, which will stop at $T_0^* = x_0$. For this example, the PL constant is $\mu = \|\nabla f(x)\|/f(x) = (2\varepsilon x)^2/(\varepsilon x^2) = 4\varepsilon$, which leads to the predicted stopping time $T_0 = 2\sqrt{x_0^2/(4\varepsilon)} = x_0\varepsilon^{-1/2}$. Consequently, the prediction of Proposition 4 is exact when $\varepsilon = 1$ but becomes very loose when $\varepsilon \ll 1$ or $\varepsilon \gg 1$. The intuition is that dynamics of signGD is actually independent of the curvature ε. This is quite different from gradient descent.

Instability at the stationary points Intuitively, both the small oscillations and the spikes are caused by the (near) singularity of the dynamics at the stationary points of the objective function. For adaptive gradient methods, at the stationary points, we have $\mathbf{v} = 0$. Hence, if the dynamics is linearized around the stationary point, the Jacobian will have very big eigenvalues (at the order of $O(\frac{1}{\epsilon})$). For example, linearizing continuous RMSprop (7) around some stationary point $(\boldsymbol{x}^*, 0)$ gives

$$\begin{aligned}\dot{\boldsymbol{x}} &= -\frac{\nabla^2 f(\boldsymbol{x}^*)}{\epsilon}(\boldsymbol{x} - \boldsymbol{x}^*),\\ \dot{\mathbf{v}} &= -a\mathbf{v}.\end{aligned}$$

The Jacobian of the above linearized dynamics at $(x^*, 0)$ is

$$\begin{bmatrix} -\frac{\nabla^2 f(\boldsymbol{x}^*)}{\epsilon} & 0 \\ 0 & -aI \end{bmatrix}, \tag{11}$$

whose eigenvalues are $-\lambda/\epsilon$ and $-a$, where λ is any eigenvalue of the Hessian $\nabla^2 f(\boldsymbol{x}^*)$. Therefore, this stationary point is nearly singular. This is not a problem for the continuous dynamics. However, for the discrete dynamics, the stationary point is unstable unless the learning rate is smaller than $\frac{2\epsilon}{\lambda_{\max}} \ll 1$. This implies that the discrete dynamics with a learning rate larger than $O(\epsilon)$ cannot converge to that stationary point, and instead, it may converge to some periodic trajectories or just oscillate around the stationary point. This analysis also holds for Adam.

Remark 5 *A rough way to understand the near singularity at the stationary points is to view the iterations as a GD with very large learning rate. Specifically, when $\boldsymbol{x}_t$ is close to a stationary point, $\mathbf{v}$ is very small, and we have*

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta\frac{\nabla f(\boldsymbol{x}_t)}{\sqrt{\mathbf{v}_{t+1}} + \epsilon} \approx \boldsymbol{x}_t - \frac{\eta}{\epsilon}\nabla f(\boldsymbol{x}_t). \tag{12}$$

Hence, $\boldsymbol{x}_t$ does not converge to the stationary point unless $\lambda_{\max}(\nabla^2 f(\boldsymbol{x}_t)) \leq O(\varepsilon)$, i.e., the landscape is extremely flat.

According the above analysis, oscillations may not happen if the landscape around the stationary point is sufficiently flat. This happens to be the case of the cross entropy loss function, for which $\lambda_{\max}(\nabla^2 f(\boldsymbol{x}_t)) \to 0$ as $\boldsymbol{x}_t$ approaches the minimum.

For low dimensional strongly convex objective functions, RMSprop can converge to a 2-periodic solution. For example, if the objective function is $f(x) = \frac{1}{2}x^2$, then the 2-periodic solution is an oscillation between $x = \frac{\eta}{2}$ and $x = -\frac{\eta}{2}$, where η is the learning rate. Figure 6 shows the convergence to this 2-periodic solution. This gives us a toy example of the small oscillations around the minimum.

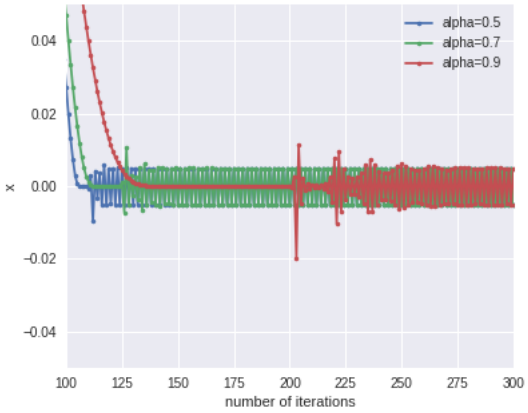


Figure 6: The trajectory of RMSprop for the 1-dimensional quadratic function $f(x) = \frac{x^2}{2}$ for different values of α. $\eta = 0.01$. One sees that all the trajectories eventually converge to the 2-periodic solution at $x = \frac{\eta}{2}$ and $x = -\frac{\eta}{2}$.

For more complicated objective functions, such as high-dimensional quadratic function, or the loss function of neural network models, the RMSprop trajectories show more complicated oscillation patterns, such as the spikes. We will take a closer look at the large spikes in the next section, where we see that Adam is also vulnerable to spikes.

4. Adam: performances for different values of a and b

The dynamic behavior of Adam is more complicated than RMSprop since it is influenced by 2 hyper-parameters. Different combinations of α and β (or a and b) can lead to different dynamic patterns. To rule out the influence of the learning rate, we will consider a and b instead of α and β. As is mentioned before, α and β are given by a and b through $\alpha = 1 - a\eta$ and $\beta = 1 - b\eta$. As we have seen in Proposition 1 , when a and b are sufficiently large compared to η, Adam behaves like signGD. For relatively small a and b, through extensive numerical experiments, we have found that there are roughly three different regimes of qualitative patterns in the parameter space (see Figure 7):

1. **The spike regime** happens when b is sufficiently larger than a. In this regime, large spikes appear in the loss curve, which makes the optimization process unstable. By observations,

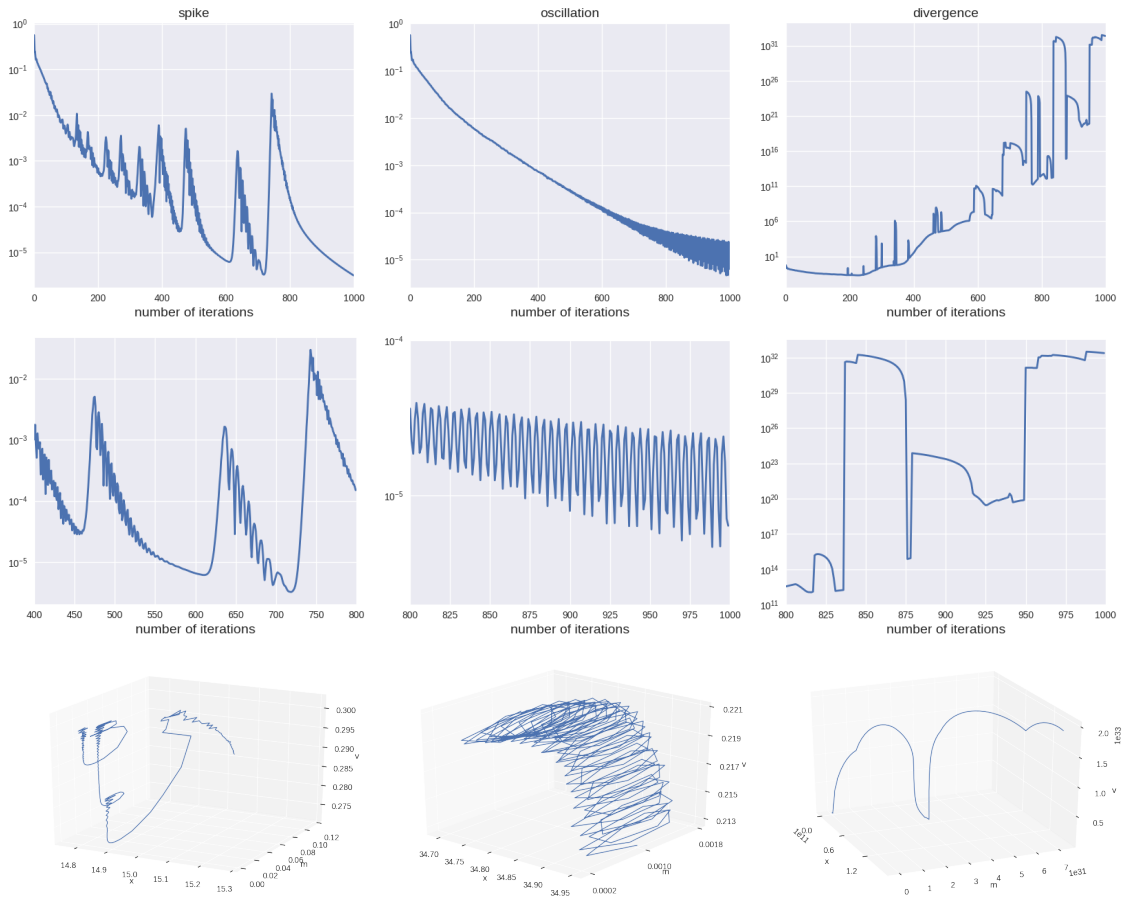


Figure 7: The three typical behavior patterns for Adam and the trajectories in the state space of $(\|\boldsymbol{x}\|, \|\mathbf{m}\|, \|\sqrt{\mathbf{v}}\|)$. $\eta = 0.001$. The model and the training data are the same as Figure 2. The first row shows the loss curve of totally 1000 iterations, the second row shows part of the loss curve (the last 200 iterations for oscillation and divergence regimes, and $400-800$ iterations for the spike regime), the bottom row shows the state space trajectory in the same period shown in the second row. **Left:** $a = 1$, $b = 100$, large spikes appear in the loss curve; **Middle:** $a = 10$, $b = 10$, the loss is small and oscillates very fast, and the amplitude of the oscillation is also small; **Right:** $a = 100$, $b = 1$, the loss is large and blows up.

spikes do not prevent the algorithm from achieving a small training loss, but they make the loss curve unstable by frequently driving the training loss to large values.

2. **The oscillation regime** happens when a and b have similar magnitude (or in the same order). In this regime, the loss curve exhibits fast and small oscillations. A small and stable loss curve can be achieved in this regime.

3. **The divergence regime** happens when a is sufficiently larger than b. In this regime, the loss curve is unstable and usually diverges after a period of training. This regime should be avoided in practice since the training loss stays large.

In Figure 7, we show one typical loss curve for each regime for a typical neural network model. We also show typical trajectories in the state space of $(\|\boldsymbol{x}\|, \|\mathbf{m}\|, \|\sqrt{\mathbf{v}}\|)$ for the three regimes. These trajectories are also qualitatively different for different regimes.

Next, we study the transition between the different regimes and the training loss behavior in different regimes. To this end, we carried out experiments for a multi-layer neural network model on the Fashion-MNIST dataset, with different values of a and b until the behavior of the training loss curve stabilizes. The left panel of Figure 8 shows the heatmap of the average loss value of the last 1000 iterations. The right panel of Figure 8 shows the classification of the behavior of the training curve into three different categories (oscillations, spikes, and divergence).

From these figures, we see that in the divergence regime, the training loss does not perform well (actually in some cases it may even blow up). Hence this regime should be avoided in practice. In the oscillation regime, the loss values are small and quite robust to the change of hyper-parameters. Therefore this is the regime that should be preferred in practice. This is the regime when $a \approx b$.

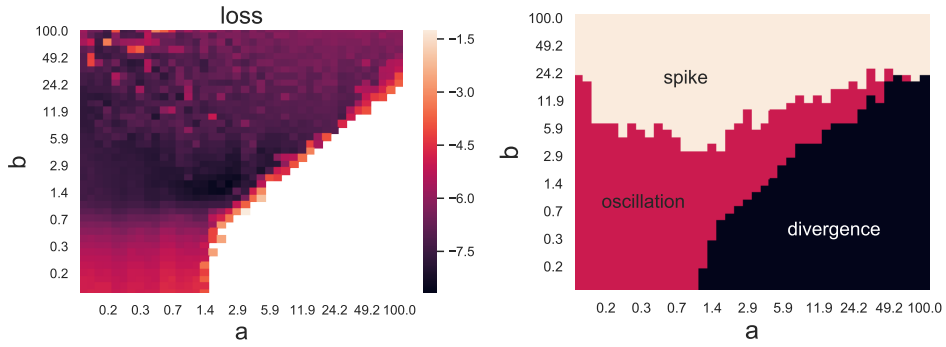


Figure 8: Train a neural network to fit the FashionMNIST dataset with Adam optimizers with varying momentum factors. The model is a fully connected network of 6 hidden layers and the width of each layer is 500. The learning rate of is 1e-3. **Left:** Heatmap of average training loss over the last 1000 iterations. The loss is shown in the logarithmic scale. a and b range from 0.1 to 100 and are also shown in the logarithmic scale. **Right:** The classification of the different dynamical behaviors of the loss curve.

4.1. Training ResNets on CIFAR10

The above investigation suggests that Adam performs better when $\alpha \approx \beta$. Here we provide further support by considering a more realistic problem: training a ResNet18 (He et al., 2016) on CIFAR10 using stochastic Adam with a large batch size. The results are shown in Figure 9. We see that with the default parameters ($\beta = 0.9, \alpha = 0.999$), there are are large spikes during the late phase of training. In contrast, when $a \approx b$, Adam converges very smoothly and is also faster than using the default parameters.

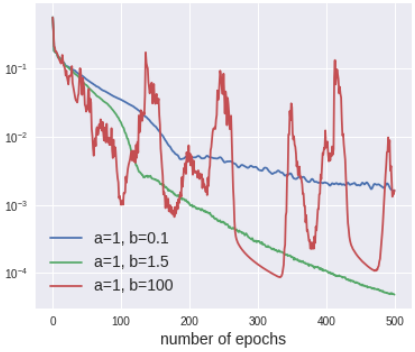


Figure 9: The training loss curve of stochastic Adam on a ResNet18 model and CIFAR-10 dataset. The learning rate is 1e-3. The red line shows the results of using the default hyper-parameters setting ($\beta = 0.9, \alpha = 0.999$). 1000 samples are taken from each class to form the training dataset. The network is a standard ResNet18 used in (He et al., 2016) but with number of channels halved. The batch size is 1000.

4.2. Solving Poisson equation

Consider the Poisson equation with Dirichlet boundary condition

$$-\Delta u = f \text{ in } \Omega, \quad u = g \text{ on } \partial\Omega.$$

Let $u(\cdot;\theta)$ denote the parameterized model. Deep Galerkin Method (DGM) (Sirignano and Spiliopoulos, 2018) looks for the solution that minimizes the following objective function

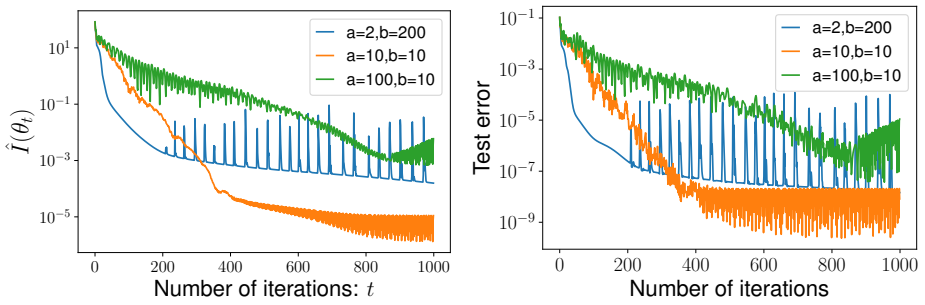
$$\hat{I}(\theta) = \frac{1}{n_d}\sum_{i=1}^{n_d}(\Delta u(\boldsymbol{x}_i;\theta) + f(\boldsymbol{x}_i))^2 + \frac{1}{n_b}\sum_{j=1}^{n_b}(u(\tilde{\boldsymbol{x}}_j;\theta) - g(\tilde{\boldsymbol{x}}_j))^2, \tag{13}$$

where $\{\boldsymbol{x}_i\}_{i=1}^{n_d}$ and $\{\tilde{\boldsymbol{x}}_j\}_{j=1}^{n_b}$ are samples uniformly drawn from Ω and $\partial\Omega$, respectively.

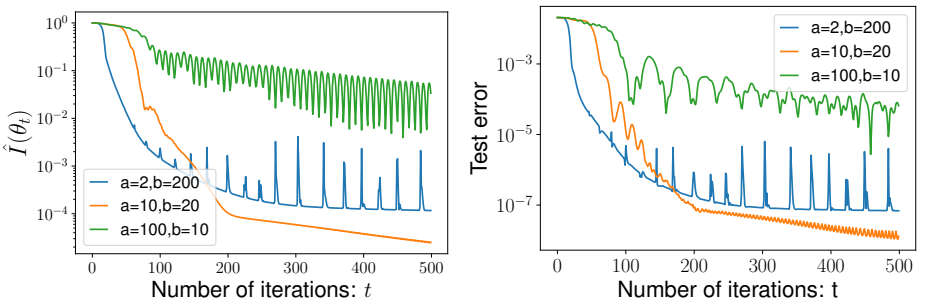
Here, we consider two examples:

- $\Omega = (0,1)^4, f(\boldsymbol{x}) = 0, g(\boldsymbol{x}) = x_1x_2 + x_3x_4$. In this case, the solution is $u^*(\boldsymbol{x}) = x_1x_2 + x_3x_4$. $u(\cdot;\theta)$ is parameterized using a 3-layer fully-connected networks with the architecture being 4-200-200-1 and the Tanh activation function is applied.
- $\Omega = \{(x_1, x_2) : x_1^2 + x_2^2 < 1\}, f(x_1, x_2) = 1, g(x_1, x_2) = 0$. The solution in this case is $u^*(\boldsymbol{x}) = \frac{1}{4}(x_1^2 + x_2^2 - 1)$. $u(\cdot;\theta)$ is parameterized using a 5-layer fully connected network, whose architecture is 2-10-10-10-10-1. The GELU activation function (Hendrycks and Gimpel, 2016) is applied.

For each example, we uniformly sample 2000 points in Ω and extra 2000 points on $\partial\Omega$ to form the training set. Figure 10 shows the training curves of Adams with various a's and b's. The learning rate is fixed to be 5e-4. One can see that with the default hyperparameters, Adam inevitably endures large spikes during the late phase of training. In contrast, when $a \approx b$, the training curve becomes much more stable and faster, although there still exist very small oscillations during the very late phase of training. It is also expected that Adam performs the worst when $a > b$.



$(a)\ \Omega = (0,1)^4, f = 0, g(\boldsymbol{x}) = x_1x_2 + x_3x_4.$



$(b)\ \Omega = \{(x_1, x_2) : x_1 + x_2 < 1\}, f = 1, g = 0$

Figure 10: Solving two Poisson equations with the deep Galerkin method (DGM). Adam optimizers with various a's and b's are used to optimize the objective functions. The learning rate is fixed to be 5e-4. Note that the case of $a = 2, b = 200$ corresponds to the default setting, i.e. $\beta = 0.9, \alpha = 0.999$. **Left:** The dynamics of the objective function of DGM. **Right:** The dynamics of the test error. Here the test error is $\mathbb{E}_{\boldsymbol{x}}[(u(\boldsymbol{x};\theta) - u^*(\boldsymbol{x}))]$, which is estimated by the emprical mean over 10000 extra samples, independently drawn from Unif(Ω).

5. Discussion

In this paper, we reported the results of some systematical investigation on the dynamic behavior of adaptive gradient algorithms, particularly RMSprop and Adam. Three typical phenomena—fast initial convergence, small oscillation, and large spikes—are observed and analyzed. The influence of the choice of the hyper-parameters on the dominant training behavior is also investigated.

It is worth noting that the investigation in this paper focuses on the full-batch setting. However, the result in Figure 9 provides some evidence to show that the phenomena revealed here should also be of relevance for the stochastic setting when the batch size is relatively large. The systematic study of the influence of batch size, especially in the small-batch regime, is left to future work.

There are still many other important open questions. For example, learning rate decay is a common practice used in training large neural networks. When performing learning rate decay, usually one does not change the values of α and β. This makes the effective a and b larger, pushing the optimizer to the signGD-like regime. Another choice is to adaptively tune α, β such that a and b are kept fixed. It is interesting to see the comparison of the two strategies.

This paper focuses on optimization. For machine learning problems, another important consideration when implementing optimization algorithms is the generalization performance. It has been reported that the solutions found by adaptive gradient algorithms usually perform a bit worse than

those found by SGD in terms of generalization (see (Wilson et al., 2017)). The study of generalization performance of adaptive gradient algorithms is left to future work.

References

Lukas Balles and Philipp Hennig. Dissecting Adam: The sign, magnitude and variance of stochastic gradients. In *International Conference on Machine Learning*, pages 404–413, 2018.

Anas Barakat and Pascal Bianchi. Convergence and dynamical behavior of the ADAM algorithm for non convex stochastic optimization. *arXiv preprint arXiv:1810.02263*, 2018.

Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of Adam-type algorithms for non-convex optimization. In *International Conference on Learning Representations*, 2018.

André Belotto da Silva and Maxime Gazeau. A general system of differential equations to model first order adaptive algorithms. *arXiv preprint arXiv:1810.13108*, 2018.

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

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

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Boris Teodorovich Polyak. Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 3(4):643–653, 1963.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. In *International Conference on Learning Representations*, 2018.

Martin Riedmiller and Heinrich Braun. Rprop-a fast adaptive learning algorithm. In *Proc. of ISCIS VII), Universitat*. Citeseer, 1992.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in neural information processing systems*, pages 4148–4158, 2017.

Yuege Xie, Xiaoxia Wu, and Rachel Ward. Linear convergence of adaptive stochastic gradient descent. In *International Conference on Artificial Intelligence and Statistics*, pages 1475–1485. PMLR, 2020.

Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao, and Quanquan Gu. On the convergence of adaptive gradient methods for nonconvex optimization. *arXiv preprint arXiv:1808.05671*, 2018.

Appendix A. Proof of Proposition 1

Here we prove Proposition 1. For that purpose, we show that for any $T > 0$ and $\tau > 0$, there exists an $\eta_{T,\tau}$, such that as long as $\eta < \eta_{T,\tau}$ we have

$$\sup_{t\in[0,T]} \|\boldsymbol{X}^\eta(t) - \boldsymbol{x}(t)\| < \tau. \tag{14}$$

In the following we focus on RMSprop. The proof for Adam is similar.

First, let K be a positive integer whose value will be specified later, and let $\tilde{\boldsymbol{x}}_k^\eta = \boldsymbol{x}(k\eta)$. Then, for $\boldsymbol{x}_K^\eta$ and $\tilde{\boldsymbol{x}}_K^\eta$ we have

$$\|\boldsymbol{x}_K^\eta - \boldsymbol{x}_0\| \le \eta \sum_{i=0}^{K-1} \left\| \frac{\nabla f(\boldsymbol{x}_i)}{\sqrt{\mathbf{v}_{i+1}} + \epsilon} \right\| \le \eta \sum_{i=0}^{K-1} \frac{M}{\epsilon} = \frac{\eta K M}{\epsilon},$$

and

$$\|\tilde{\boldsymbol{x}}_K^\eta - \boldsymbol{x}_0\| \le \int_0^{K\eta} \left\| \frac{\nabla f(\boldsymbol{x}(t))}{|\nabla f(\boldsymbol{x}(t))| + \epsilon} \right\| dt \le \frac{\eta K M}{\epsilon}$$

Therefore,

$$\|\boldsymbol{x}_K^\eta - \tilde{\boldsymbol{x}}_K^\eta\| \le \frac{2\eta K M}{\epsilon}. \tag{15}$$

Next, for $k \ge K$, we have

$$\boldsymbol{x}_{k+1}^\eta - \tilde{\boldsymbol{x}}_{k+1}^\eta = (\boldsymbol{x}_k^\eta - \tilde{\boldsymbol{x}}_k^\eta) + \left(\int_{k\eta}^{(k+1)\eta} \frac{\nabla f(\boldsymbol{x}(t))}{|\nabla f(\boldsymbol{x}(t))| + \epsilon} dt - \eta \frac{\nabla f(\boldsymbol{x}_k^\eta)}{\sqrt{\mathbf{v}_{k+1}} + \epsilon} \right).$$

Let

$$\Delta = \int_{k\eta}^{(k+1)\eta} \frac{\nabla f(\boldsymbol{x}(t))}{|\nabla f(\boldsymbol{x}(t))| + \epsilon} dt - \eta \frac{\nabla f(\boldsymbol{x}_k^\eta)}{\sqrt{\mathbf{v}_{k+1}} + \epsilon},$$

then

$$\|\boldsymbol{x}_{k+1}^\eta - \tilde{\boldsymbol{x}}_{k+1}^\eta\| \le \|\boldsymbol{x}_k^\eta - \tilde{\boldsymbol{x}}_k^\eta\| + \|\Delta\|. \tag{16}$$

Next we estimate $\|\Delta\|$. First we have

$$\begin{aligned}
\Delta &= \int_{k\eta}^{(k+1)\eta} \frac{\nabla f(\boldsymbol{x}(t))}{|\nabla f(\boldsymbol{x}(t))| + \epsilon} - \frac{\nabla f(\boldsymbol{x}_k^\eta)}{\sqrt{\mathbf{v}_{k+1}} + \epsilon} dt \\
&= \int_{k\eta}^{(k+1)\eta} \nabla f(\boldsymbol{x}(t)) \left(\frac{1}{|\nabla f(\boldsymbol{x}(t))| + \epsilon} - \frac{1}{\sqrt{\mathbf{v}_{k+1}} + \epsilon} \right) dt + \int_{k\eta}^{(k+1)\eta} \frac{\nabla f(\boldsymbol{x}(t)) - \nabla f(\boldsymbol{x}_k^\eta)}{\sqrt{\mathbf{v}_{k+1}} + \epsilon} dt \\
&:= I + J.
\end{aligned}$$

For J, we have

$$\begin{aligned}\|J\| &\le \frac{1}{\epsilon}\int_{k\eta}^{(k+1)\eta} \|\nabla f(\boldsymbol{x}(t)) - \nabla f(\boldsymbol{x}_k^\eta)\| dt \\ &\le \frac{L}{\epsilon}\int_{k\eta}^{(k+1)\eta} \|\boldsymbol{x}(t) - \boldsymbol{x}_k^\eta\| dt \\ &\le \frac{L}{\epsilon}\int_{k\eta}^{(k+1)\eta} (\|\boldsymbol{x}(t) - \tilde{\boldsymbol{x}}_k^\eta\| + \|\tilde{\boldsymbol{x}}_k^\eta - \boldsymbol{x}_k^\eta\|) dt \\ &\le \frac{L}{\epsilon}\left(\frac{\eta^2 M}{\epsilon} + \eta\|\tilde{\boldsymbol{x}}_k^\eta - \boldsymbol{x}_k^\eta\|\right) \\ &= \frac{\eta L}{\epsilon}\|\tilde{\boldsymbol{x}}_k^\eta - \boldsymbol{x}_k^\eta\| + \frac{\eta^2 LM}{\epsilon^2}. \end{aligned} \tag{17}$$

For I, we have

$$\begin{aligned}\|I\| &\le \int_{k\eta}^{(k+1)\eta} \left\| \frac{|\nabla f(\boldsymbol{x}(t))|}{|\nabla f(\boldsymbol{x}(t))| + \epsilon} \frac{\sqrt{\mathbf{v}_{k+1}} - |\nabla f(\boldsymbol{x}(t))|}{\sqrt{\mathbf{v}_{k+1}} + \epsilon} \right\| dt \\ &\le \int_{k\eta}^{(k+1)\eta} \left\| \frac{\sqrt{\mathbf{v}_{k+1}} - |\nabla f(\boldsymbol{x}(t))|}{\sqrt{\mathbf{v}_{k+1}} + \epsilon} \right\| dt \\ &\le \frac{1}{\epsilon}\int_{k\eta}^{(k+1)\eta} \|\sqrt{\mathbf{v}_{k+1}} - |\nabla f(\boldsymbol{x}_k^\eta)|\| dt + \frac{1}{\epsilon}\int_{k\eta}^{(k+1)\eta} \||\nabla f(\boldsymbol{x}_k^\eta)| - |\nabla f(\boldsymbol{x}(t))|\| dt \end{aligned} \tag{18}$$

The second term in (18) can be estimated in a similar way as $\|J\|$, and it can be bounded by

$$\frac{\eta L}{\epsilon}\|\tilde{\boldsymbol{x}}_k^\eta - \boldsymbol{x}_k^\eta\| + \frac{\eta^2 LM}{\epsilon^2}.$$

For the first term of (18), use the fact that $(a-b)^2 \le a^2 - b^2$ for any $a \ge b \ge 0$, we have

$$\begin{aligned}&\frac{1}{\epsilon}\int_{k\eta}^{(k+1)\eta} \|\sqrt{\mathbf{v}_{k+1}} - |\nabla f(\boldsymbol{x}_k^\eta)|\| dt \\ &\le \frac{\eta}{\epsilon}\|\mathbf{v}_{k+1} - \nabla f^2(\boldsymbol{x}_k^\eta)\|^{\frac{1}{2}} \\ &= \frac{\eta}{\epsilon}\left\|(1-\alpha)\nabla f^2(\boldsymbol{x}_k^\eta) + \alpha(1-\alpha)\nabla f^2(\boldsymbol{x}_{k-1}^\eta) + \cdots + \alpha^k(1-\alpha)\nabla f^2(\boldsymbol{x}_0^\eta) - \nabla f^2(\boldsymbol{x}_k^\eta)\right\|^{\frac{1}{2}} \\ &\le \frac{\eta}{\epsilon}\left(\left\|(1-\alpha)\sum_{i=0}^{K-1}\alpha^i(\nabla f^2(\boldsymbol{x}_{k-i}^\eta) - \nabla f^2(\boldsymbol{x}_k^\eta))\right\| + 2\alpha^K M\right)^{\frac{1}{2}} \\ &\le \frac{2\eta^{3/2}MK^{1/2}}{\epsilon^{3/2}} + \frac{2\eta M\alpha^{K/2}}{\epsilon}. \end{aligned} \tag{19}$$

Hence we have

$$\|I\| \le \frac{\eta L}{\epsilon}\|\tilde{\boldsymbol{x}}_k^\eta - \boldsymbol{x}_k^\eta\| + \frac{\eta^2 LM}{\epsilon^2} + \frac{2\eta^{3/2}MK^{1/2}}{\epsilon^{3/2}} + \frac{2\eta M\alpha^{K/2}}{\epsilon}. \tag{20}$$

Combining (20) with (17) we get the estimate of Δ:

$$\|\Delta\| \leq \frac{2\eta L}{\epsilon}\|\tilde{\boldsymbol{x}}_k^\eta - \boldsymbol{x}_k^\eta\| + \frac{2\eta^2 LM}{\epsilon^2} + \frac{2\eta^{3/2}MK^{1/2}}{\epsilon^{3/2}} + \frac{2\eta M\alpha^{K/2}}{\epsilon}. \tag{21}$$

Hence

$$\|\boldsymbol{x}_{k+1}^\eta - \tilde{\boldsymbol{x}}_{k+1}^\eta\| \leq (1+\frac{2\eta L}{\epsilon})\|\boldsymbol{x}_k^\eta - \tilde{\boldsymbol{x}}_k^\eta\| + \frac{2\eta^2 LM}{\epsilon^2} + \frac{2\eta^{3/2}MK^{1/2}}{\epsilon^{3/2}} + \frac{2\eta M\alpha^{K/2}}{\epsilon}. \tag{22}$$

Finally, by Gronwall's inequality, we have

$$\begin{aligned}\|\boldsymbol{x}_k^\eta - \tilde{\boldsymbol{x}}_k^\eta\| &\leq \left(1+\frac{2\eta L}{\epsilon}\right)^{k-K}\|\boldsymbol{x}_K^\eta - \tilde{\boldsymbol{x}}_K^\eta\| + \left(1+\frac{2\eta L}{\epsilon}\right)^{k-K}\left(\frac{\eta M}{\epsilon} + \frac{\eta^{1/2}MK^{1/2}}{\epsilon^{1/2}L} + \frac{M\alpha^{K/2}}{L}\right)\\ &\leq \left(1+\frac{2\eta L}{\epsilon}\right)^{k}\left(\frac{2\eta KM}{\epsilon} + \frac{\eta^{1/2}MK^{1/2}}{\epsilon^{1/2}L} + \frac{M\alpha^{K/2}}{L}\right).\end{aligned} \tag{23}$$

We want (23) to hold for $t \leq T$, which means for all $k \leq \frac{T}{\eta}$. For these values of k, we have

$$\|\boldsymbol{x}_k^\eta - \tilde{\boldsymbol{x}}_k^\eta\| \leq e^{\frac{LT}{\epsilon}}\left(\frac{2\eta KM}{\epsilon} + \frac{\eta^{1/2}MK^{1/2}}{\epsilon^{1/2}L} + \frac{M\alpha^{K/2}}{L}\right). \tag{24}$$

Therefore, for any fixed small value $\tau > 0$, by taking sufficiently large K and sufficiently small η, we can achieve

$$\|\boldsymbol{x}_k^\eta - \tilde{\boldsymbol{x}}_k^\eta\| \leq \frac{\tau}{2}, \tag{25}$$

for any $0 \leq k \leq \lfloor\frac{T}{\eta}\rfloor + 1$. Then, if we further let

$$\eta < \frac{\tau\epsilon}{4M},$$

for any $t \in [0, T]$, let k satisfy $t \in [k\eta, (k+1)\eta)$, we have

$$\begin{aligned}\|\boldsymbol{X}^\eta(t) - \boldsymbol{x}(t)\| &\leq \|\boldsymbol{x}_k^\eta - \tilde{\boldsymbol{x}}_k^\eta\| + \|\boldsymbol{X}^\eta(t) - \tilde{\boldsymbol{x}}_k^\eta\| + \|\boldsymbol{x}(t) - \boldsymbol{x}_k^\eta\|\\ &\leq \frac{\tau}{2} + \frac{2\eta M}{\epsilon}\\ &\leq \tau.\end{aligned} \tag{26}$$

This completes the proof.

Appendix B. Proof of Proposition 3

By (Barakat and Bianchi, 2018) as well as Proposition 2, the solution of (8) exists, and this solution is the limit trajectory of discrete Adam algorithm (3) with $\eta \to 0$ and $\alpha = 1 - a\eta$, $\beta = 1 - b\eta$. Hence, initially we have

$$\frac{d\boldsymbol{x}(0)}{dt} = -\text{sign}(\nabla f(\boldsymbol{x}_0)).$$

By (8), we can solve $\mathbf{v}$ and $\mathbf{m}$ involving $\boldsymbol{x}$:

$$\begin{aligned}\mathbf{v}(t) &= a\int_0^t e^{a(s-t)}(\nabla f(\boldsymbol{x}(t)))^2 ds,\\ \mathbf{m}(t) &= b\int_0^t e^{b(s-t)}\nabla f(\boldsymbol{x}(t)) ds,\end{aligned} \tag{27}$$

and hence for the equation of $\boldsymbol{x}$ we have

$$\dot{\boldsymbol{x}}(t) = -\sqrt{\frac{\int_0^t e^{-as}ds}{\int_0^t e^{a(s-t)}(\nabla f(\boldsymbol{x}(t)))^2 ds}}\left(\frac{\int_0^t e^{b(s-t)}\nabla f(\boldsymbol{x}(t))ds}{\int_0^t e^{-bs}ds}\right). \tag{28}$$

Let t^* be the first time when some element of $\nabla f(\boldsymbol{x}(t))$ becomes smaller than $c/2$, i.e.

$$t^* = \inf_t \left\{[\nabla f(\boldsymbol{x}(t))]_i \le \frac{c}{2} \text{ for some } i\right\},$$

then for any $t \in [0, t^*]$ we have $[\nabla f(\boldsymbol{x}(t))]_i \ge c/2$ for any $i = 1, 2, ..., d$. This together with (28) implies

$$|[\dot{\boldsymbol{x}}]_i| \le \frac{2M}{c} \tag{29}$$

for any $i = 1, 2, ..., d$. Then, assume $t^* < \frac{c^2}{4ML}$, we have

$$\begin{aligned}|[\nabla f(\boldsymbol{x}(t^*))]_i| &\ge |[\nabla f(\boldsymbol{x}(0))]_i| - L\|\boldsymbol{x}(t^*) - \boldsymbol{x}(0)\|\\ &> |[\nabla f(\boldsymbol{x}(0))]_i| - L\frac{c^2}{4ML}\frac{2M}{c}\\ &\ge c - \frac{c}{2}\\ &= \frac{c}{2},\end{aligned}$$

which is contradictory to the definition of t^*. Therefore, $t^* \ge \frac{c^2}{4ML}$. Considering $c < M$, we obtain $\tau < t^*$.

Next, let

$$\begin{aligned}\boldsymbol{r}_1(\tau) &= \frac{\int_0^\tau e^{a(s-\tau)}\left((\nabla f(\boldsymbol{x}(s)))^2 - (\nabla f(\boldsymbol{x}(\tau)))^2\right)ds}{\int_0^\tau e^{-as}ds},\\ \boldsymbol{r}_2(\tau) &= \frac{\int_0^\tau e^{b(s-\tau)}\left(\nabla f(\boldsymbol{x}(s)) - \nabla f(\boldsymbol{x}(\tau))\right)ds}{\int_0^\tau e^{-bs}ds}.\end{aligned}$$

Then, we have

$$\frac{\int_0^t e^{a(s-t)}(\nabla f(\boldsymbol{x}(t)))^2 ds}{\int_0^t e^{-as}ds} = (\nabla f(\boldsymbol{x}(\tau)))^2 + \boldsymbol{r}_1(\tau),$$

and

$$\frac{\int_0^t e^{b(s-t)}\nabla f(\boldsymbol{x}(t))ds}{\int_0^t e^{-bs}ds} = \nabla f(\boldsymbol{x}(\tau)) + \boldsymbol{r}_2(\tau).$$

Hence, combining (28), we have

$$\dot{\boldsymbol{x}}(\tau) = -\sqrt{\frac{1}{(\nabla f(\boldsymbol{x}(\tau)))^2 + \boldsymbol{r}_1(\tau)}}(\nabla f(\boldsymbol{x}(\tau)) + \boldsymbol{r}_2(\tau)),$$

and then at τ

$$\begin{aligned}\|\dot{\boldsymbol{x}} + \operatorname{sign}(\nabla f(\boldsymbol{x}))\| &= \left\| \frac{\nabla f(\boldsymbol{x})}{\sqrt{(\nabla f(\boldsymbol{x}))^2}} - \sqrt{\frac{1}{(\nabla f(\boldsymbol{x}))^2 + \boldsymbol{r}_1}}(\nabla f(\boldsymbol{x}) + \boldsymbol{r}_2) \right\| \\ &\leq \left\| \nabla f(\boldsymbol{x}) \left(\sqrt{\frac{1}{(\nabla f(\boldsymbol{x}))^2}} - \sqrt{\frac{1}{(\nabla f(\boldsymbol{x}))^2 + \boldsymbol{r}_1}} \right) \right\| \\ &\quad + \left\| \sqrt{\frac{1}{(\nabla f(\boldsymbol{x}))^2 + \boldsymbol{r}_1}}(\nabla f(\boldsymbol{x}) - (\nabla f(\boldsymbol{x}) + \boldsymbol{r}_2)) \right\| \end{aligned} \tag{30}$$

To estimate the above terms, we first estimate $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$. For $\boldsymbol{r}_2$, by the Lipschitz property of the gradient and (29), we have

$$\|\nabla f(\boldsymbol{x}(s)) - \nabla f(\boldsymbol{x}(\tau))\| \leq L\|\boldsymbol{x}(s) - \boldsymbol{x}(\tau)\| \leq \frac{2ML\sqrt{d}}{c}|\tau - s| \leq \frac{2ML\sqrt{d}}{c}\tau.$$

Hence,

$$\|\boldsymbol{r}_2(\tau)\| \leq \frac{2ML\sqrt{d}}{c}\tau. \tag{31}$$

For $\boldsymbol{r}_2$, considering

$$(\nabla f(\boldsymbol{x}(s)))^2 - (\nabla f(\boldsymbol{x}(\tau)))^2 = (\nabla f(\boldsymbol{x}(s)) - (\nabla f(\boldsymbol{x}(\tau)))(\nabla f(\boldsymbol{x}(s)) + (\nabla f(\boldsymbol{x}(\tau)))$$

and the upper bound for $\|\nabla f(\boldsymbol{x})\|$, similar to the estimation of $\boldsymbol{r}_2$ we have

$$\|\boldsymbol{r}_1(\tau)\| \leq \frac{4M^2L\sqrt{d}}{c}\tau. \tag{32}$$

By (31) and (32), we have

$$\begin{aligned} &\left\| \nabla f(\boldsymbol{x}) \left(\sqrt{\frac{1}{(\nabla f(\boldsymbol{x}))^2}} - \sqrt{\frac{1}{(\nabla f(\boldsymbol{x}))^2 + \boldsymbol{r}_1}} \right) \right\| \\ &\leq M \left\| \frac{\sqrt{(\nabla f(\boldsymbol{x}))^2 + \boldsymbol{r}_1} - \sqrt{(\nabla f(\boldsymbol{x}))^2}}{\sqrt{(\nabla f(\boldsymbol{x}))^2((\nabla f(\boldsymbol{x}))^2 + \boldsymbol{r}_1)}} \right\| \\ &= M \left\| \frac{\boldsymbol{r}_1}{\sqrt{(\nabla f(\boldsymbol{x}))^2((\nabla f(\boldsymbol{x}))^2 + \boldsymbol{r}_1)}(\sqrt{(\nabla f(\boldsymbol{x}))^2 + \boldsymbol{r}_1} + \sqrt{(\nabla f(\boldsymbol{x}))^2})} \right\| \\ &\leq \frac{48M^3L\sqrt{d}}{c^4}\tau, \end{aligned} \tag{33}$$

where the last line is derived by $|[\nabla f(\boldsymbol{x})]_i| \geq \frac{c}{2}$, and $|[\boldsymbol{r}_1]_i| < \frac{c^2}{8}$ which comes from $\tau < \frac{c^3}{32M^2L\sqrt{d}}$. On the other hand, we have

$$\left\| \sqrt{\frac{1}{(\nabla f(\boldsymbol{x}))^2 + \boldsymbol{r}_1}} (\nabla f(\boldsymbol{x}) - (\nabla f(\boldsymbol{x}) + \boldsymbol{r}_2)) \right\| \leq \frac{6ML\sqrt{d}}{c^2}\tau. \tag{34}$$

Combining (33) and (34), we have

$$\begin{aligned} \|\dot{\boldsymbol{x}} + \text{sign}(\nabla f(\boldsymbol{x}))\| &\leq \left(\frac{48M^3L\sqrt{d}}{c^4} + \frac{6ML\sqrt{d}}{c^2} \right) \tau \\ &\leq \frac{54M^3L\sqrt{d}}{c^4}\tau, \end{aligned} \tag{35}$$

where the second inequality comes from $c \leq M$. This completes the proof.