# Nostalgic Adam: Weighing more of the past gradients when designing the adaptive learning rate

**Haiwen Huang**
School of Mathematical Sciences
Peking University, Beijing, 100871
smshhw@pku.edu.cn

**Chang Wang**
School of Mathematical Sciences
Peking University, Beijing, 100871
hobbitish@pku.edu.cn

**Bin Dong**
Beijing International Center for Mathematical Research, Peking University
Center for Data Science, Peking University
Beijing Institute of Big Data Research
Beijing, China
dongbin@math.pku.edu.cn

## Abstract

First-order optimization methods have been playing a prominent role in deep learning. Algorithms such as RMSProp and Adam are rather popular in training deep neural networks on large datasets. Recently, Reddi et al. [2018] discovered a flaw in the proof of convergence of Adam, and the authors proposed an alternative algorithm, AMSGrad, which has guaranteed convergence under certain conditions. In this paper, we propose a new algorithm, called Nostalgic Adam (NosAdam), which places bigger weights on the past gradients than the recent gradients when designing the adaptive learning rate. This is a new observation made through mathematical analysis of the algorithm. We also show that the estimate of the second moment of the gradient in NosAdam vanishes slower than Adam, which may account for faster convergence of NosAdam. We analyze the convergence of NosAdam and discover a convergence rate that achieves the best known convergence rate $O(1/\sqrt{T})$ for general convex online learning problems. Empirically, we show that NosAdam outperforms AMSGrad and Adam in some common machine learning problems.

## 1 Introduction

First-order optimization algorithms have been playing a prominent role in deep learning due to their efficiency in solving large-scale optimization problems. These algorithms can be written in the following generic form [Reddi et al., 2018]:

$$x_{t+1} = x_t - \frac{\alpha_t}{\psi(g_1, ..., g_t)}\phi(g_1, ..., g_t), \tag{1}$$

where $g_i$ is the gradient of a loss function $f$ calculated at the $i$-th time step, $\alpha_t/\psi(g_1, ..., g_t)$ is the adaptive learning rate, and $\phi(g_1, ..., g_t)$ is the estimate of the first moment. There has been numerous work on the design of the term $\phi(g_1, ..., g_t)$ which can be traced back to the classical momentum methods [Polyak, 1964]. In this paper, however, we focus more on the design of the adaptive learning rate $\alpha_t/\psi(g_1, ..., g_t)$ to exploit the geometry of the loss function. This idea first appeared in Duchi et al. [2010], and was later adopted by most first-order optimization algorithms for neural network training. In particular, some most-widely used first-order algorithms, such as RMSProp [Tieleman and

Preprint. Work in progress.

Hinton, 2012], AdaDelta[Zeiler, 2012] and Adam[Kingma and Ba, 2014], use exponential moving average to estimate the square of the gradient magnitude (or the second moment of gradient) to adaptively change the learning rate. More specifically, they choose the following function $\psi$ for (1):

$$\psi(g_1, ..., g_t) = \sqrt{V_t},\ V_t = diag(v_t)^1 \text{ and } v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2.^2 \tag{2}$$

This running average, however, may lead to divergence of the Algorithm 1 as pointed out very recently in the inspiring work by Reddi et al. [2018]. This issue is closely related to the quantity

$$\Gamma_{t+1} = \frac{\sqrt{V_{t+1}}}{\alpha_{t+1}} - \frac{\sqrt{V_t}}{\alpha_t},$$

which essentially measures the change of the inverse of learning rate at each iteration. Algorithms that used exponential moving average to estimate the second moment of gradient cannot guarantee positive semi-definiteness of the quantity $\Gamma_t$, which can lead to divergence [Reddi et al., 2018].

Reddi et al. [2018] suggested that, to fix the aforementioned issue, the optimization algorithm should have "long-term memory", i.e. to incorporate more past gradients in the learning rate, and they proposed two new algorithms. The first algorithm, AMSGrad, adds a "max" operation to the running average, which essentially assigns bigger weights to the past large gradients. The other algorithm, AdamNC, puts various weights on past gradients, e.g. equal weights on all the past gradients. Both algorithms guarantee the positive semi-definiteness of $\Gamma_t$ and hence convergence under certain conditions.

In this paper, we address the same issue as in Reddi et al. [2018], and provide another approach to utilize the past gradients in the design of the adaptive learning rate. We propose a new family of algorithms called *Nostalgic Adam (NosAdam)*, which places bigger weights on the past gradients than recent gradients. We also propose a special class of NosAdam algorithm, called *NosAdam-HH*, which uses hyper-harmonic series to design the running average of the square of the gradient. We prove that NosAdam can achieve the best known convergence rate $O(1/\sqrt{T})$ for general convex online learning problems. Furthermore, NosAdam includes AdamNC as a special case. Unlike AdamNC, however, the assumption on the running average of NosAdam is independent of the loss function and much easier to check.

An interesting property of NosAdam is that the second moment estimate $v_t$ diminishes to zero noticeably slower than Adam. In [Kingma and Ba, 2014], the authors pointed out that "the second moment estimate $\hat{v}_t$ vanishes to zeros after a few epochs". When we use Adam in practice, we normally use $\psi(g_1, ..., g_t) = \sqrt{\hat{V}_t + \epsilon}$ with a manually chosen $\epsilon > 0$ to avoid the problem of dividing by zero. If $\hat{v}_t$ quickly vanishes to zero, then Adam is reduced to a momentum SGD algorithm. [3] In this paper, we empirically show that the second moment in NosAdam may provide a better approximation to the geometry of data due to the slow diminishing of the second moment estimate. This contributes to the faster convergence of NosAdam.

The contribution of this paper is threefold:

- We propose a new algorithm, NosAdam, which addresses the same convergence issue of Adam as Reddi et al. [2018]. We discovered that the conditions that guarantee convergence of NosAdam force our algorithm to weigh more of the past gradients in the weighted average estimate of the square of the gradient magnitude, which is a new observation.
- We raise the issue that the role of the second moment estimate in Adam needs to be further investigated. We empirically show that in NosAdam, the second moment estimate vanishes slower than Adam and hence plays a more important role in the algorithm.
- Our experiments show that NosAdam outperforms Adam and AMSGrad in some common machine learning problems.

[1] For simplicity, we ignore the bias-correction in Adam.

[2] In this paper, the $p$-th power of a vector $V = (v_1, \dots, v_n)$ is defined as $V^p = (v_1^p, \dots, v_n^p)$, i.e. element-wise power

[3] It is known to the community that $\epsilon$ seems to play an important role in Adam, which is mainly caused by the vanishing second moment estimate.

## 2 Related Work

Since the introduction of AdaGrad [Duchi et al., 2010], the adaptive learning rate has been extensively studied. There are various algorithms of this type. The most-widely used first-order methods may be RMSProp [Tieleman and Hinton, 2012] and Adam [Kingma and Ba, 2014]. Our algorithm bears a similar spirit as these algorithms, but has better convergence result and a different way of exploiting the past gradients.

**AdaGrad** AdaGrad is originally derived from the proximal gradient method in the context of online learning. The algorithm takes the form $x_{t+1} = x_t - \alpha \cdot g_t/(\sqrt{\sum_{i=1}^t g_i^2})$. The momentum version of AdaGrad is NosAdam-HH with $\gamma = 0$.

**RPROP and RMSProp** The idea of using the sign of the gradients to adaptively change the step size stems from RPORP [Riedmiller and Braun, 1993]. In the slides of Geoffrey Hinton [Tieleman and Hinton, 2012], RMSProp was introduced as the mini-batch version of RPROP, and was described as "dividing the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight". This update can be denoted as $x_{t+1} = x_t - \alpha_t \cdot g_t/\sqrt{(\mathbb{E}[g^2])}$. The exponential moving average is used for the estimation of $\mathbb{E}[g^2]$. However, as shown by Reddi et al. [2018], this may lead to divergence of the algorithm. Since the estimation only relies on very recent mini-batches, the variance among different mini-batches can also make the estimation unstable.

**Adam** Adam adopts a similar idea as previous algorithms by combining AdaGrad with RMSProp [Kingma and Ba, 2014]. The update rule of Adam can be written as $x_{t+1} = x_t - \alpha_t \cdot \mathbb{E}[g_t]/\sqrt{(\mathbb{E}[g^2])}$, where the first moment $\mathbb{E}[g_t]$ is estimated using momentum methods, and the second moment is the same as RMSProp. However, in our experiments, due to the fast decay of the estimate of second momentum of Adam, the major contribution of speed-up is in fact from the use of momentum methods rather than the second moment information. Reddi et al. [2018] recently pointed out a fundamental flaw of the convergence analysis of Adam, and fixed the convergence issue by proposing two new variant algorithms. Our work is most related to [Kingma and Ba, 2014, Reddi et al., 2018].

## 3 Algorithm

**Algorithm 1** Nostalgic Adam Algorithm (NosAdam)

**Input:** $x \in F$, $m_0 = 0$, $V_0 = 0$
**for** $t = 1$ **to** $T$ **do**
  $g_t = \nabla f_t(x_t)$
  $\beta_{2,t} = B_{t-1}/B_t$, where $B_t = \sum_{k=1}^t b_k$ with $b_k > 0$, $t \geq 1$ and $B_0 = 0$
  $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$
  $v_t = \beta_{2,t} v_{t-1} + (1-\beta_{2,t}) g_t^2$, and $V_t = diag(v_t)$
  $\hat{x}_{t+1} = x_t - \alpha_t m_t/\sqrt{V_t}$
  $x_{t+1} = \mathcal{P}_{\mathcal{F},\sqrt{V_t}}(\hat{x}_{t+1})$
**end for**

The proposed Nostalgic Adam (NosAdam) algorithm is presented in Algorithm 2. The difference of NosAdam from AMSGrad and Adam is marked in red in Algorithm 2. Our motivation comes from convergence analysis of NosAdam. As mentioned in the introduction, a sufficient condition for convergence of the algorithm is the positive semi-definiteness of the quantity $\Gamma_t$. We observe that the condition $\Gamma_t \succeq 0$ is almost tightly satisfied when $B_t/t$ is non-increasing (see the appendix), which naturally leads to the choice of $\beta_{2,t}$ of NosAdam. (See the remark following Theorem 2 for further discussions.)

Convergence results of NosAdam will be presented in the next section and the proof will be given in the appendix. Note that there are multiple choices of $B_t$ which can guarantee convergence of the algorithm. One example, which is also the algorithm we tested in our numerical experiments, is to choose $B_t$ using the hyper-harmonic series, i.e. $B_t = \sum_{k=1}^t k^{-\gamma}$. To differentiate this special case of NosAdam from the general NosAdam, we call the algorithm that uses hyper-harmonic series as NosAdam-HH. For NosAdam-HH, the conditions on the running average in Theorem 2 are readily satisfied for all $\gamma \geq 0$ and hence its convergence is automatically guaranteed (see Corollary 1).

Moreover, our algorithm shows an important difference in the estimate of square of gradient magnitude from exponential moving average methods. Since $\beta_{2,t} = B_{t-1}/B_t$, when $B_t = \sum_{k=1}^{t} b_k$, we have

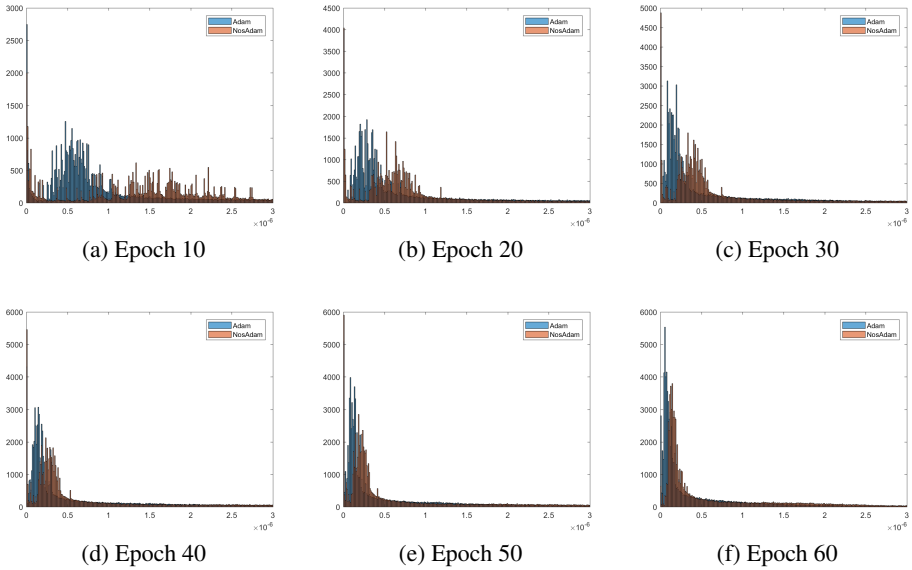
$$V_t = \sum_{k=1}^{t} \frac{b_k}{B_t} g_k^2. \tag{3}$$

A sufficient condition to enure $\Gamma_t \succeq 0$, which eventually leads to the convergence of the algorithm, is to require $B_t/t$ to be non-increasing (see Lemma 2 in appendix). This requirement implies $b_k$ to be non-increasing as well. Note from (3) that $b_k$ is the relative weight of $g_k^2$ in $V_t$. This indicates that NosAdam weighs more of the past gradients than the recent ones. Such weighing strategy is the opposite of exponential moving average which weighs more of the most present gradients.

We also provide an intuitive explanation of NosAdam's weighing strategy of the gradients. The update rule (1) can be rewritten as $x_{t+1} = x_t - \alpha_t \mathbb{E}[g]/\sqrt{\mathbb{E}[g^2]}$. The momentum methods not only can estimate $\mathbb{E}[g]$ but can also considerably speed up optimization. When it comes to the estimation of $\mathbb{E}[g^2]$, however, methods like exponential moving average may fail. This is because in stochastic setting, the gradient magnitude may vary a lot from batch to batch, making the estimate by exponential moving average unstable. In the appendix, we show a similar example as in Reddi et al. [2018]. This example shows that when the oscillation of the gradients is large enough, methods that use exponential moving average to estimate $\mathbb{E}[g^2]$ will diverge.

On the other hand, weighing more of the past gradients can make the estimate much more stable than Adam and AMSGrad. Even when the new gradients oscillate, the weighing strategy of NosAdam can still stabilize the entire estimate. For NosAdam-HH, the extent to which how much the present gradient weighs in the estimate can be easily controlled by $\gamma$. Furthermore, in contrast to AMSGrad, NosAdam still allows the estimate of $v_t$ to decrease or increase according to the magnitudes of the new gradients.

As mentioned in the introduction that the weighing strategy of NosAdam leads to a slower decay of $v_t$ than Adam. The fast decay of $v_t$ makes the algorithm almost equivalent to the momentum version of SGD, which may be the main reason why Adam sometimes performs poorly. To support this observation, we present the distribution of $v_t$ from Adam and NosAdam at epoch 10, 20, ..., 60 in Figure 1. This may indicate that NosAdam takes better advantage of the geometry of the loss function than Adam which leads to faster convergence.




(a) Epoch 10 (b) Epoch 20 (c) Epoch 30

(d) Epoch 40 (e) Epoch 50 (f) Epoch 60

Figure 1: Distribution of second moment estimate $v_t$ for all parameters trained with Adam and NosAdam during the training of a multilayer neural network on MNIST.

## 4 Convergence Analysis

We adopt the online learning setup which was first introduced by Zinkevich [2003]. At each time step $t$, the optimization algorithm picks a point $x_t \in \mathcal{F}$, where $\mathcal{F} \in \mathbb{R}^d$ is the feasible set of parameters. A loss function $f_t$ corresponding to the chosen instances in the next mini-batch is then revealed, and the algorithm incurs the loss function $f_t(x_t)$. We evaluate our algorithm using the following regret

$$R_T = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in \mathcal{F}} \sum_{t=1}^{T} f_t(x). \tag{4}$$

Let $\mathcal{S}_d^+$ be the set of all positive definite $d \times d$ matrices. The projection operator $\mathcal{P}_{\mathcal{F},A(y)} : \mathbb{R}^d \mapsto \mathbb{R}^d$ for a given $A \in \mathcal{S}_d^+$ is defined as $\operatorname{argmin}_{x \in \mathcal{F}} ||A^{1/2}(x-y)||$ with $y \in \mathbb{R}^d$. Finally, we say $\mathcal{F}$ has bounded diameter $D_\infty$, if $||x-y||_\infty \leq D_\infty$ for all $x, y \in \mathcal{F}$. The convergence result of NosAdam is given by the following theorem.

**Theorem 1** (Convergence of NosAdam)**.** *Let $B_t$ and $b_k$ be the sequences defined in Algorithm 2. Let $\alpha_t = \alpha/\sqrt{t}$, $\beta_1 = \beta_{1,1}, \beta_{1,t} \leq \beta_1$ for all t. Assume that $f_t$ is convex for each t, $\mathcal{F}$ has bounded diameter $D_\infty$ and $||\nabla f_t(x)||_\infty \leq G_\infty$ for all t and $x \in \mathcal{F}$. Furthermore, let $\beta_{2,t}$ be such that the following conditions are satisfied:*

$$1.\ \frac{B_t}{t b_t^2} \geq \frac{B_{t-1}}{(t-1) b_{t-1}^2}$$
$$2.\ \frac{B_t}{t} \leq \frac{B_{t-1}}{t-1}$$

*Then the regret of the sequence $\{x_t\}$ generated by NosAdam (Algorithm 2) satisfies the following bound*

$$R_T \leq \frac{D_\infty^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T} v_{T,i}^{\frac{1}{2}} + \frac{D_\infty^2}{2(1-\beta_1)} \sum_{t=1}^{T} \sum_{i=1}^{d} \frac{\beta_{1,t} v_{t,i}^{\frac{1}{2}}}{\alpha_t} + \frac{\alpha \beta_1}{(1-\beta_1)^3} \sum_{i=1}^{d} \sqrt{\frac{B_T}{T} \frac{\sum_{t=1}^{T} b_t g_{t,i}^2}{b_T^2}}.$$

**Remark**: The specific form of $\beta_{2,t} = B_{t-1}/B_t$ is in fact general as long as $\beta_{2,t}$ is data-independent and only relies on $t$. This can be easily seen from the following argument. For any $0 < \beta_{2,t} < 1$, it can be expressed as $B_{t-1}/B_t$ with $B_t > B_{t-1} > 0$. Since $b_t = B_t - B_{t-1}$, this is again equivalent to say $b_t > 0$. Therefore, the choice of $\beta_{2,t}$ only relies on the choice of $b_t$, e.g. the choice $b_t = t^{-\gamma}$ for NosAdam-HH.

This theorem analyzes a family of algorithms that uses a running average to estimate the square of the gradient magnitude. The data-independent constraints on $B_t$ and $b_t$ are easy-to check for algorithm designers. In particular, NosAdam-HH is just one special case where $B_t$ is a hyper-harmonic series. Different choice of $\gamma$ controls the the order of decreasing rate of $b_k$. The convergence of NosAdam-HH is a direct consequence of Theorem 2. We summarize the convergence result of NosAdam-HH in the following corollary.

**Corollary 1** (Convergence of NosAdam-HH)**.** *Suppose $\beta_{1,t} = \beta_1 \lambda^{t-1}$. The regret of the sequence $\{x_t\}$ generated by NosAdam-HH satisfies the following bound*

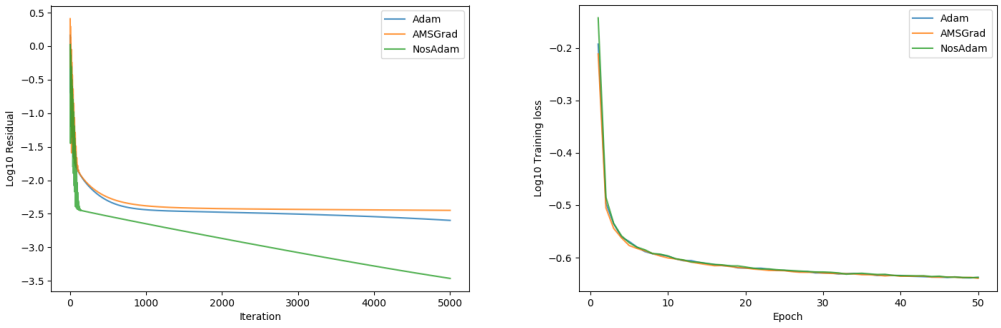
$$R_T \leq \frac{D_\infty^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T} v_{T,i}^{\frac{1}{2}} + \frac{D_\infty^2 G_\infty \beta_1}{2(1-\beta_1)} \frac{1}{(1-\lambda)^2} \cdot d + \frac{2\alpha\beta_1}{(1-\beta_1)^3} G_\infty \sqrt{T}.$$

These convergence results indicate that our algorithm achieves convergence rate of $O(1/\sqrt{T})$, which is by far the best convergence rate.[4]

## 5 Experiments

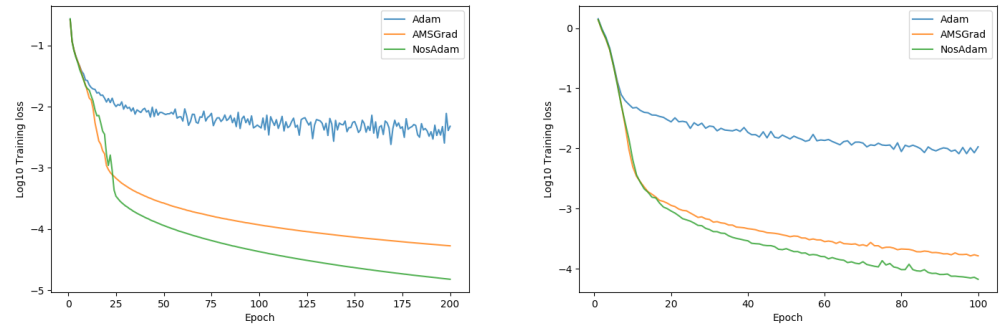
To evaluate the efficiency of the proposed algorithm, we study the problem of an ill-conditioned least squares problem and multiclass classification using logistic regression, multilayer fully connected

[4] In Reddi et al. [2018], the convergence rate of AMSGrad is in fact $O(\sqrt{\frac{\log(T)}{T}})$.




(a) Ill-Conditioned Least Squares Problem

(b) MNIST Logistic Regression




(c) MNIST Multilayer Neural Network

(d) CIFAR-10 ResNet18

Figure 2: Performance comparison of Adam, AMSGrad and NosAdam for the ill-conditioned least squares problem, logistic regression, multilayer fully connected neural network and ResNet18. For clear comparison, the training loss is plotted in log scale with base 10.

neural networks and deep convolutional neural networks on MNIST [LeCun and Cortes, 2010] and CIFAR-10 [Krizhevsky et al.]. We demonstrate that NosAdam-HH can perform efficiently in both convex and nonconvex problems. Since our main modification to Adam is on the design of the adaptive learning rate by properly weighing the past gradients, our algorithm bears the same spirit as the recently proposed AMSGrad. Therefore, we shall compare NosAdam with Adam and AMSGrad on different problems.

Throughout our experiments, we fixed $\beta_1$ to be 0.9. We manually selected an optimal $\beta_2 \in \{0.99, 0.999\}$ for Adam and AMSGrad, and $\gamma \in \{0.05, 0.1\}$ for NosAdam-HH. During the experiments, we found that NosAdam-HH is relatively insensitive to the choice of $\gamma$. The learning rate of each algorithm is chosen by dense grid search and all the results are reported using the best set of hyper-parameters. Since the focus of this study is on optimization, we only report the training errors which is a common practice [Sutskever et al., 2013].

**Ill-Conditioned Least Squares Problem** The least squares problem $\min_x \frac{1}{2}||Ax - b||_2^2$ is often used to test optimization algorithms. In our experiment, we consider an ill-conditioned least squares problem. We first generate a diagonal matrix $D$ with its diagonal entries decaying at rate $O(k^{-5})$ with $k$ being its diagonal index. Then we generate a random orthogonal matrix $Q$, and set $A = QDQ^\top$. To be consistent with the theory, we set the step size $\alpha_t = \alpha/\sqrt{t}$. The results are shown in Figure 2a. We can see that NosAdam performs significantly better than Adam and AMSGrad.

**Logistic Regression** To investigate the performance of the algorithms on convex problems, we evaluate Adam, AMSGrad and NosAdam on multi-class logistic regression problem using the MNIST dataset [LeCun and Cortes, 2010]. Same as the previous problem, we set the step size $\alpha_t = \alpha/\sqrt{t}$. We set the mini-batch size to 128. According to Figure 2b, the three algorithms perform similarly.

**Multilayer Fully Connected Neural Networks** We first train a simple 1-hidden fully connected layer neural network for the multi-classification problem on MNIST. We use a fully connected 100 rectified linear units (ReLU) as the hidden layer. Furthermore, we use a constant step size $\alpha_t = \alpha$ throughout the experiments. This is consistent with the setting typically used in the deep learning community for training neural networks. From Figure 2c, we see that the performance of NosAdam-HH outperforms AMSGrad, while Adam is much worse than both algorithms with large oscillations. This is consistent with our previous intuitive explanation that, by properly using the past gradients, the second moment estimate can be more stable.

**Deep Convolutional Neural Networks** Finally, we train a deep convolutional neural network on CIFAR-10[Krizhevsky et al.]. In our experiment, we use ResNet18 as the deep model. The residual block is the same as in He et al. [2015], with two convolutional layers and batch normalization. The ResNet18 we use is constructed by piling up a $3 \times 3$ convolutional layer and nine residual blocks. Same as the previous experiment, we set the step size $\alpha$ to be constant and mini-batch size to 128. A grid search is used to determine the best hyper-parameters of the algorithms. The performance of the three algorithms is demonstrated in Figure 2d. Similar to the previous experiment, we see that NosAdam is more efficient than Adam and AMSGrad.

## 6 Conclusions

In this paper, we focus on the design of adaptive learning rate for first-order stochastic optimization algorithms. Specifically, we address the importance of paying more attention to the past gradients than the present ones when estimating the square of the gradients' magnitude. We proposed a new family of algorithms, NosAdam, and analyzed its convergence. Empirically, we show that NosAdam-HH, a specific class of algorithms of NosAdam, outperforms Adam and AMSGrad on some convex and nonconvex problems in machine learning.

## 7 Acknowledgements

This work would not have existed without the support of BICMR and School of Mathematical Sciences, Peking University. Bin Dong is supported in part by NSFC 11671022. The authors would also like to thank the mosquitos in Jingchun Garden whose presence significantly speeded up our work progress at the night of the submission.

# Appendix

## A Divergence issue of Adam: a quantitative example

Besides the two examples given in Reddi et al. [2018], here we provide a more quantitative example to illustrate the issue of divergence of Adam. This example suggests that the cause of divergence of Adam is due to the oscillation of gradients which in fact leads to the violation of $V_t/\alpha_t^2 - V_{t-1}/\alpha_{t-1}^2 \succeq 0$.

**Lemma 1.** *Since $\sqrt{V_{t+1}}/\alpha_{t+1} - \sqrt{V_t}/\alpha_t$ may not be positive semi-definite, the exponential moving average algorithms may not converge. Define $f_t(x)$ as*

$$f_t(x) = \begin{cases} Cx & t \ mod \ 3{=}1 \\ -Bx & otherwise \end{cases},$$

*where $\mathscr{F} = [-1, 1]$, B,C >0. When $4 < C^2/B^2 < 4 + 2(1-\beta)$ with $0 < \beta < 1$ being the exponential decay parameter of $V_t$, Adam fails to converge.*

*Proof.* It is immediate from the definition of $f_t(x)$ that

$$x_{3t+4} = x_{3t+1} - \frac{\alpha C}{\sqrt{(3t+1)(V_{3t+1}+\epsilon)}} + \frac{\alpha B}{\sqrt{(3t+2)(V_{3t+2}+\epsilon)}} + \frac{\alpha B}{\sqrt{(3t+3)(V_{3t+3}+\epsilon)}},$$
$$V_{3t+1} > V_{3t+2} < V_{3t+3} < V_{3t+4},$$

and

$$x_{3t+1} > x_{3t+2} < x_{3t+3} < x_{3t+4}.$$

When $t \to \infty$, we have

$$\begin{aligned} V_{3t} &= [\beta^2(1-\beta)C^2 + (1+\beta)(1-\beta)B^2](1 + \beta^3 + \beta^6 + ... + \beta^{3t-3}) \\ &= \frac{\beta^2 C^2 + (1+\beta)B^2}{1+\beta+\beta^2}(1-\beta^{3t}) \to \frac{\beta^2 C^2 + (1+\beta)B^2}{1+\beta+\beta^2} \end{aligned}$$

And we will regard $V_{3t}$ as $(\beta^2C^2 + (1+\beta)B^2)/(1+\beta+\beta^2)$ when $t \to \infty$ in the following proof.

Therefore, when $4 < C^2/B^2 < 4 + 2(1-\beta)$, $\epsilon \to 0$ and $t \to \infty$, we have

$$\begin{aligned} &\frac{C^2}{4B^2} < 1 + \frac{1-\beta}{2} < 1 + \frac{3(1-\beta)}{1+4\beta+\beta^2} = \frac{4+\beta+\beta^2}{1+4\beta+\beta^2} \\ \Rightarrow &\frac{\beta C^2}{4B^2} < 1 - \frac{1-\beta^3}{1+\beta^2+4\beta} \\ \Rightarrow &1 > \frac{1-\beta^3}{1+\beta^2+\frac{\beta C^2}{B^2}} + \frac{\beta C^2}{4B^2} \\ \Rightarrow &\sqrt{\frac{1}{\beta} - \frac{(B^2+\epsilon)(1-\beta)/\beta}{\beta(\beta V_{3t} + (1-\beta)C^2 + \epsilon) + (1-\beta)(B^2+\epsilon)}} > \frac{C}{2B} \\ \Rightarrow &\sqrt{\frac{\beta V_{3t} + (1-\beta)C^2 + \epsilon}{\beta^2(V_{3t+1}+\epsilon) + (1-\beta^2)(B^2+\epsilon)}} > \sqrt{\frac{\beta V_{3t} + (1-\beta)C^2 + \epsilon}{\beta(V_{3t+1}+\epsilon) + (1-\beta)(B^2+\epsilon)}} > \frac{C}{2B} \\ \Rightarrow &\sqrt{\frac{\beta V_{3t} + (1-\beta)C^2 + \epsilon}{\beta(V_{3t+1}+\epsilon) + (1-\beta)(B^2+\epsilon)}} + \sqrt{\frac{\beta V_{3t} + (1-\beta)C^2 + \epsilon}{\beta^2(V_{3t+1}+\epsilon) + (1-\beta^2)(B^2+\epsilon)}} > \frac{C}{B} \\ \Rightarrow &\frac{\alpha B}{\sqrt{(3t+2)(V_{3t+2}+\epsilon)}} + \frac{\alpha B}{\sqrt{(3t+3)(V_{3t+3}+\epsilon)}} > \frac{\alpha C}{\sqrt{(3t+1)(V_{3t+1}+\epsilon)}} \\ \Rightarrow &x_{3t+4} > x_{3t+1}. \end{aligned}$$

Therefore, there exists $T_0$ and $x_{T_0}$, such that $x_{3t+1} > x_{T_0} > -1$ and $x_{3t+2}, x_{3t+3} < x_{3t+1}$ for $t > T_0$. This implies that

$$\begin{aligned} &f_{3t+1}(x_{3t+1}) + f_{3t+2}(x_{3t+2}) + f_{3t+3}(x_{3t+3}) - (f_{3t+1}(x^*) + f_{3t+2}(x^*) + f_{3t+3}(x^*)) \\ =&Cx_{3t+1} - Bx_{3t+2} - Bx_{3t+3} + C - 2B \\ >&(C-2B)(1+x_{3t+1}) > (C-2B)(1+x_{T_0}) > 0, \end{aligned}$$

which leads to $\sum_{i=1}^{T} f_i(x_i) - f_i(x^*) = O(T)$. □

**Remark:** Note that when $C > 2B$, $x = -1$ provides the minimal regret. Thus, if we fix $C$ in our lemma, $B$ essentially quantifies the oscillation towards the wrong direction. This lemma provides us with two insights: 1) when the gradients have a severe oscillation, the exponential moving average algorithms can diverge; 2) the lemma also reveals how much oscillation of gradients can cause divergence. The numerical result can be seen in Figure 3.

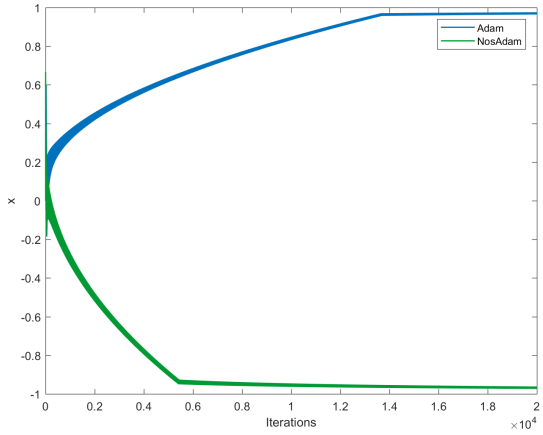



Figure 3: Numerical result of the divergence example

## B Convergence of Nostalgic Adam

**Algorithm 2** Nostalgic Adam Algorithm

**Input:** $x \in F$, $m_0 = 0$, $V_0 = 0$

**for** $t = 1$ **to** $T$ **do**
  $g_t = \nabla f_t(x_t)$
  $\beta_{2,t} = B_{t-1}/B_t$, where $B_t = \sum_{k=1}^{t} b_k$ for $t \geq 1$, and $B_0 = 0$
  $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$
  $v_t = \beta_{2,t} v_{t-1} + (1-\beta_{2,t}) g_t^2$, and $V_t = diag(v_t)$
  $\hat{x}_{t+1} = x_t - \alpha_t m_t / \sqrt{V_t}$
  $x_{t+1} = \mathcal{P}_{\mathcal{F}, \sqrt{V_t}}(\hat{x}_{t+1})$
**end for**

NosAdam is recalled in Algorithm 2. The convergence of NosAdam is presented in the following theorem.

**Theorem 2** (Convergence of NosAdam)**.** *Let $B_t$ and $b_k$ be the sequences defined in Algorithm 2. Let $\alpha_t = \alpha/\sqrt{t}$, $\beta_1 = \beta_{1,1}, \beta_{1,t} \leq \beta_1$ for all t. Assume that $f_t$ is convex for each t, $\mathcal{F}$ has bounded diameter $D_\infty$ and $||\nabla f_t(x)||_\infty \leq G_\infty$ for all t and $x \in \mathcal{F}$. Furthermore, let $\beta_{2,t}$ be such that the following conditions are satisfied:*

$$1.\ \frac{B_t}{t b_t^2} \geq \frac{B_{t-1}}{(t-1) b_{t-1}^2}$$

$$2.\ \frac{B_t}{t} \leq \frac{B_{t-1}}{t-1}$$

*Then the regret of the sequence $\{x_t\}$ generated by NosAdam (Algorithm 2) satisfies the following bound*

$$R_T \leq \frac{D_\infty^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T} v_{T,i}^{\frac{1}{2}} + \frac{D_\infty^2}{2(1-\beta_1)} \sum_{t=1}^{T} \sum_{i=1}^{d} \frac{\beta_{1,t} v_{t,i}^{\frac{1}{2}}}{\alpha_t} + \frac{\alpha \beta_1}{(1-\beta_1)^3} \sum_{i=1}^{d} \sqrt{\frac{B_T}{T} \frac{\sum_{t=1}^{T} b_t g_{t,i}^2}{b_T^2}}.$$

We first prove a few lemmas before proving the theorem.

**Lemma 2.** *The positive semi-definiteness of $V_t/\alpha_t^2 - V_{t-1}/\alpha_{t-1}^2$ is satisfied when $B_t/t$ is non-increasing.*

*Proof.*

$$\begin{aligned}\frac{V_t}{\alpha_t^2} =& \frac{t}{\alpha^2}\sum_{j=1}^{t}\Pi_{k=1}^{t-j}\beta_{2,t-k+1}(1-\beta_{2,j})g_j^2\\ =& \frac{t}{\alpha^2}\sum_{j=1}^{t}\frac{B_{t-1}}{B_t}\cdots\frac{B_j}{B_{j+1}}\frac{B_j-B_{j-1}}{B_j}g_j^2\\ =& \frac{t}{B_t\alpha^2}\sum_{j=1}^{t}b_jg_j^2 \geq \frac{t-1}{B_{t-1}\alpha^2}\sum_{j=1}^{t-1}b_jg_j^2\\ =& \frac{V_{t-1}}{\alpha_{t-1}^2}\end{aligned}$$

□

**Remark:** This lemma shows that in NosAdam, $V_t/\alpha_t^2 - V_{t-1}/\alpha_{t-1}^2$ is positive semi-definite. Since $g_j$ can be infinitesimal, this suggests that when $b_t$ is monotonous, the condition non-increasing of $B_t/t$ is almost necessary for the positive semi-definiteness of $V_t/\alpha_t^2 - V_{t-1}/\alpha_{t-1}^2$. Note that $B_t/t$ being non-increasing is equivalent to that $b_t$ is non-increasing (which can be derived by mathematical induction). Since $V_t = \sum_{k=1}^{t} g_k^2 b_k/B_t$, it is also equivalent to having non-increasing weights on $g_k^2$ when calculating $V_t$.

**Lemma 3.** *When $\frac{V_t}{\alpha_t} - \frac{V_{t-1}}{\alpha_{t-1}}$ is positive semi-definite, we have*

$$\begin{aligned}R_T \leq& \frac{D_\infty^2}{2\alpha(1-\beta_1)}\sum_{i=1}^{d}\sqrt{T}v_{T,i}^{\frac{1}{2}} + \frac{D_\infty^2}{2(1-\beta_1)}\sum_{t=1}^{T}\sum_{i=1}^{d}\frac{\beta_{1,t}v_{t,i}^{\frac{1}{2}}}{\alpha_t}\\ &+\frac{\beta_1}{2(1-\beta_1)}\sum_{t=1}^{T}\frac{\alpha_t}{1-\beta_1}\frac{\sum_{j=1}^{t}\beta_1^{t-j}g_j^2}{\sqrt{\sum_{j=1}^{t}\Pi_{k=1}^{t-j}\beta_{2,t-k+1}(1-\beta_{2,j})g_j^2}}\end{aligned}$$

The proof of this lemma is the same as the proof of [Reddi et al., 2018, Theorem 5]. Now, we provide a bound of the last term of the bound in Lemma 3.

**Lemma 4.** *Under the conditions in Theorem 2, we have*

$$\sum_{t=1}^{T}\frac{\alpha_t}{1-\beta_1}\frac{\sum_{j=1}^{t}\beta_1^{t-j}g_j^2}{\sqrt{\sum_{j=1}^{t}\Pi_{k=1}^{t-j}\beta_{2,t-k+1}(1-\beta_{2,j})g_j^2}} \leq \frac{2\alpha}{(1-\beta_1)^2}\sqrt{\frac{B_T}{T}\frac{\sum_{t=1}^{T}b_tg_{t,i}^2}{b_T^2}}$$

*Proof.*

$$\begin{aligned}&\sum_{t=1}^{T}\frac{\alpha_t}{1-\beta_1}\frac{\sum_{j=1}^{t}\beta_1^{t-j}g_j^2}{\sqrt{\sum_{j=1}^{t}\Pi_{k=1}^{t-j}\beta_{2,t-k+1}(1-\beta_{2,j})g_j^2}} = \sum_{t=1}^{T}\frac{\alpha}{1-\beta_1}\frac{\sum_{j=1}^{t}\beta_1^{t-j}g_j^2}{\sqrt{\sum_{j=1}^{t}\frac{t}{B_t}b_jg_j^2}}\\ \leq&\sum_{t=1}^{T}\frac{\alpha}{1-\beta_1}\sqrt{\frac{B_t}{t}}\sum_{j=1}^{t}\frac{\beta_1^{t-j}g_j^2}{\sqrt{\sum_{k=1}^{j}b_kg_k^2}} = \sum_{j=1}^{T}\sum_{t=j}^{T}\frac{\beta_1^{t-j}g_j^2}{\sqrt{\sum_{k=1}^{j}b_kg_k^2}}\sqrt{\frac{B_t}{t}}\frac{\alpha}{1-\beta_1}\\ \leq&\sum_{j=1}^{T}\sum_{t=j}^{T}\frac{\beta_1^{t-j}g_j^2}{\sqrt{\sum_{k=1}^{j}b_kg_k^2}}\sqrt{\frac{B_j}{j}}\frac{\alpha}{1-\beta_1}\\ \leq&\frac{\alpha}{(1-\beta_1)^2}\sum_{j=1}^{T}\frac{g_j^2}{\sqrt{\sum_{k=1}^{j}b_kg_k^2}}\sqrt{\frac{B_j}{j}}\end{aligned}$$

The first inequality comes from $\sum_{k=1}^{j} b_k g_k^2 \leq \sum_{k=1}^{t} b_k g_k^2$. The second inequality comes from the second constraint in Theorem 2, which tells us that $B_t/t$ is non-increasing with respect to $t$. The last inequality follows from $\sum_{t=j}^{T} \beta_1^{t-j} \leq 1/(1-\beta_1)$.

Let $S_j = \sum_{k=1}^{j} b_k g_k^2$. We have

$$\begin{aligned}
&\sum_{j=1}^{T} \frac{g_j^2}{\sqrt{\sum_{j=1}^{t} b_j g_j^2}} \sqrt{\frac{B_j}{j}} = \sum_{j=1}^{T} \frac{g_j^2}{\sqrt{S_j}} \sqrt{\frac{B_j}{j}} \\
\leq &\sum_{j=1}^{T} \frac{2g_j^2}{\sqrt{S_j} + \sqrt{S_{j-1}}} \sqrt{\frac{B_j}{j}} = \sum_{j=1}^{T} 2\left(\sqrt{S_j} - \sqrt{S_{j-1}}\right) \sqrt{\frac{B_j}{jb_j^2}} \\
= &2\sqrt{\frac{S_T B_T}{T b_T^2}} + 2\sum_{j=1}^{T-1} \left(\sqrt{\frac{B_j}{jb_j^2}} - \sqrt{\frac{B_{j+1}}{(j+1)b_{j+1}^2}}\right) \sqrt{S_j} \\
\leq &2\sqrt{\frac{S_T B_T}{T b_T^2}}.
\end{aligned}$$

The last inequality comes from the first constraint in Theorem 2, which tells us that $B_j/(jb_j^2)$ is non-decreasing with respect to $j$. This completes the proof of the lemma. □

*Proof.* (**Proof of Theorem 2**) Since the second constraint is satisfied, we know from Lemma 2 that $V_t/\alpha_t^2 - V_{t-1}/\alpha_{t-1}^2$ is positive semi-definite. Then combining Lemma 3 and Lemma 4 completes the proof of our theorem. □

## References


John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. Technical Report UCB/EECS-2010-24, EECS Department, University of California, Berkeley, Mar 2010. URL `http://www2.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-24.html`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Boris Polyak. Some methods of speeding up the convergence of iteration methods. 4:1–17, 12 1964.

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

M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. in ieee international conference on neural networks. 1993.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1139–III–1147. JMLR.org, 2013. URL `http://dl.acm.org/citation.cfm?id=3042817.3043064`.

T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL http://arxiv.org/abs/1212.5701.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 928–935. AAAI Press, 2003. ISBN 1-57735-189-4. URL http://dl.acm.org/citation.cfm?id=3041838.3041955.