Stochastic Euler Heavy Ball Method

Zhou Shao[†]; Junyan Liu[‡]; Tong Lin[§*]
[†]*Center for Data Science, Peking University*
[‡]*Department of Computer Science and Engineering, University California San Diego*
[§]*Key Lab. of Machine Perception (MOE), School of EECS, Peking University*
[§]*Peng Cheng Laboratory, Shenzhen, China*
{[†]shaozhou, [§]lintong}@pku.edu.cn; [‡]jul037@ucsd.edu

***Abstract*—Stochastic Heavy Ball method (SHB) has been widely used in various machine learning and deep learning tasks due to its superior generalization performance. However, a large number of effort need to be spent in tuning the learning rates of SHB, which is costly and inefficient in practical applications. Towards this end, this paper proposes the Stochastic Euler Heavy Ball method (SEHB), which simultaneously achieves good generalization like SHB and obtains rapid convergence. Our method adopts new adaptive learning rates which is different from classical adaptive methods like Adam. Convergence analysis is discussed in both convex and non-convex situations. Furthermore, we conduct numerical experiments and deep learning experiments to test the performance of SEHB. Empirical results demonstrate that our method shows better generalization performance than classical stochastic optimization methods such as SHB and Adam.**

***Index Terms*—Stochastic optimization, SGD, Adam, Heavy ball**

I. Introduction

Modern neural networks are typically trained with first-order gradient methods [1]–[3]. Remarkably, Stochastic Gradient Descend (SGD) is the main first-order methods. The method is often trained in the form of mini-batch SGD in order to meet the requirements of computing power, and achieve good generalization performance [4], [5]. However, SGD has the following main drawbacks. First, SGD chooses the negative gradients of loss functions as descent directions which would yield a slow convergence near the local minima. Second, SGD scales the gradients uniformly in all directions which may yield poor performance as well as limited training speed. Last but not least, when applied to machine learning and deep learning tasks, SGD is painstakingly hard to tune the learning rates decay scheduling manually. However, one has to decay learning rates as the algorithm proceeds in order to control the variances of stochastic gradients for achieving convergence due to the high-dimensional non-convexity of machine learning and deep learning optimization problems.

To tackle aforementioned issues, considerable efforts have been spent and several remarkable variants have been proposed recently. Accelerated schemes and adaptive methods are two categories of dominant variants. Accelerated schemes, such as stochastic heavy ball (SHB) [6] and stochastic Nesterov's accelerated gradient (SNAG) [7], employ momentum to adjust descent directions which achieve faster convergence and better generalization performance than other variants. However, they also suffer from the third drawback of SGD, and thus one need to spend many efforts on tuning and decaying learning rates manually. On the other hand, adaptive methods aim to alleviate this issue which automatically decay the learning rates and scale them non-uniformly. The first prominent algorithm in this line of research is AdaGrad [8], which divides element-wisely accumulative squared historical gradients. AdaGrad performs well when gradients are sparse, but its performance degrades in dense or non-convex settings, which is attributed to the rapid decay in learning rates. Towards this end, several methods, proposed by [9]–[11], scale gradients down by square roots of exponential moving averages of squared historical gradients (called EMA mechanism). This mechanism is very popular and some famous variants, including AdaDelta [9], RMSprop [10] and Adam [11], are based on it. Particularly, Adam is a combination of momentum and EMA mechanism which converges fast in the early training phases and is easier to tuning than SGD, becoming the default algorithm leveraged across various deep learning frameworks.

Despite Adam's popularity, there also have been concerns about their convergence and generalization performance. In particular, Adam and even most EMA based methods, may not converge to the optimal solution even in simple convex settings [12], which relies on the fact that effective learning rates of EMA based methods can potentially increase over time in a fairly quickly manner. For convergence, it is important to have the effective learning rates decrease over iterations, or at least have controlled increase [13]. This problem persists even if the learning rate scheduling (decay) is applied. Recently, considerable efforts have been spent on improving EMA based methods [12]–[17] to avoid this issue and narrow the generalization gap between EMA based methods with SGD. However, despite these efforts, one pursuing the best generalization ability of models has to choose SGD as the default optimizer rather than Adam based on the fact that there is not enough evidence to show that those Adam-type methods can get close to or even surpass SGD in general tasks. Therefore, a natural idea is whether it is possible to develop a new adaptive method different form EMA based methods, which can overcome aforementioned issues and obtain even better generalization than SGD. However, to the best of our knowledge, there exists few efforts on proposing new adaptive mechanisms whose starting points are different form EMA mechanism.

* Corresponding author

Contributions In the light of this background, we list the main contributions of our paper.

- We propose a new adaptive method, called Stochastic Euler Heavy Ball method (SEHB) which achieves new adaptivity different from classical adaptive methods. SEHB overcomes aforementioned drawbacks of SGD and achieve better generalization than SHB and other popular adaptive methods.
- We theoretically analyze the convergence of SEHB in both convex and non-convex settings. In Online convex optimization framework, SEHB shows a $O(\sqrt{T})$ regret bound; In stochastic non-convex optimization, our method shows a $O(\log T/\sqrt{T})$ convergence rate. The above bounds are at least comparable and not less than bounds of Adam-type methods.
- We conduct extensive empirical experiments for SEHB and compare with several representative methods. Empirical results show that SEHB achieves fast convergence as Adam-type methods and shows the best generalization performance in most tasks.

The rest of this paper is organized as follows. In Section 2, we list a few very closely related work about SGD and Adam-type methods. The main contribution SEHB is described in Section 3. Section 4 and Section 5 show the properties of SEHB from both theoretical and empirical perspectives separately.

II. Related Work

The literature in stochastic methods is vast and we review a few very closely related work on improving SGD or Adam. These proposed methods can simply be summarized into two families: EMA based methods and others. For EMA based methods, many efforts have been spent on closing the generalization gap between Adam and SGD family. AMSGrad [12] controls the increasing of effective learning rates over iterations, AdaBound [14] clips them, Yogi [13] considers the mini-batch size, PAdam [15] modifies the square root, RAdam [16] rectifies the variance of learning rates, AdamW [18] decouples weight decay from gradient descent, AdaBelief [17] centralizes the second order momentum in Adam. To our best knowledge, there exists several methods achieving different adaptivity from Adam-type methods in SGD. AdaGD [19] focuses on the local geometry and use the gradients of the latest two steps to adaptively adjust learning rates, which has a convergence only depending on the local smoothness in a neighborhood of the local minima. Besides, AEGD [20] is closest to our work which uses the same composite function of the loss with ours: $\sqrt{f(x)+c}$ where f is the loss and c is a constant s.t. $f(x)+c>0$ for all x in the feasible region. However, the intuition for AEGD which is far different from our method is to realize a gradient descent with the stable energy which is defined as the above composite function of the loss. Note that the energy of AEGD equals the definition just in the first step and updates in a monotonous way as the algorithm proceeds. Hence, the stable energy seems meaningless because that it unconditionally decrease over iterations.

III. Algorithm

In this section, we propose a new algorithm SEHB for achieving new adaptivity in SHB which is far different from EMA mechanism. Specifically, the motivation for SEHB is that the loss can help to adjust learning rates over iterations and we decompose the current gradient into a scaled gradient and a loss based vector which guides us to achieve adaptivity in SHB.

Notation For a vector $\theta \in \mathbb{R}^d$, we denote its i-th coordinate by θ_i; we use θ_t to denote θ in the t-th iteration and use $\theta_{t,i}$ for the i-th coordinate of θ in the t-th iteration. Furthermore, we use $||\cdot||$ to denote l_2-norm and use $||\cdot||_\infty$ to denote l_∞-norm. Given two vectors $v, w \in \mathbb{R}^d$, we use vw to denote element-wise product and use v^2 to denote element-wise square; we use $\frac{u}{v}$ to denote element-wise division.

A. Gradient Decomposition

The motivation for SEHB is that the loss can help to adjust learning rates over iterations. As the algorithm proceeds, typically one need to adjust the learning rates for convergence. Thus, the learning rate decay scheduling technique is often applied into the training process. Note that both momentum based methods and adaptive methods benefit from the combination with the learning rate decay scheduling technique. Hence, adjusting learning rates according to the loss information is feasible. SEHB employs a decomposition of gradients to access the loss information.

Consider a composite function of the loss $f(x)$:

$$g(x) = \sqrt{f(x)+c}, \tag{1}$$

where the objective loss $f(x)$ is a lower bounded function and $c>0$ is a constant s.t. $f(x)+c>0$, $\forall x \in \mathcal{X} \subseteq \mathbb{R}^d$. Take the derivative of $g(x)$ and we can decompose the gradient $\nabla f(x)$ into the product of two terms

$$\nabla f(x) = 2g(x)\nabla g(x), \tag{2}$$

where $\nabla f(x)$ and $\nabla g(x)$ are the gradient of $f(x)$ and $g(x)$ respectively. Note that $g(x)$ which has the same monotonicity with $f(x)$ includes the current information of the loss, and $\nabla g(x)$ is a scaled version of $\nabla f(x)$ with a factor $2g(x)$ which is a constant for a certain x. Thus, $-\nabla g(x)$ is also a descent direction because we have $-\nabla g(x)\nabla f(x) = -(\nabla f(x))^2/2g(x) < 0$ where $g(x) > 0$, $\forall x \in \mathcal{X}$. In conclusion, we decompose the gradient of the loss $f(x)$ into a surrogate gradient $\nabla g(x)$ which is a descent direction for optimizing $f(x)$ and a loss scalar $g(x)$. SEHB adjusts learning rates adaptively according to the loss scalar.

B. Our Method

Based on the above decomposition, we show the update rule of SEHB. To determine an optimizer, all we need are calculating the learning rate (step size) and searching the update direction. For example, the update scheme of vanilla SGD is:

$$x_{t+1} = x_t - \eta \nabla f(x_t), \tag{3}$$

where x_{t+1}, $x_t \in \mathbb{R}^d$, η is a constant learning rate, $\nabla f(x_t)$ is the gradient at the time step t and $-\nabla f(x_t)$ is the steepest descent direction.

As mentioned before, the gradient $\nabla f(x)$ is divided into a loss scalar and a scaled gradient. We introduce v to update the loss scalar and have

$$\begin{aligned} v_t &= v_{t-1} + \nabla g(x_t)(x_{t+1} - x_t), \\ x_{t+1} &= x_t - \eta v_t \nabla g(x_t), \end{aligned} \tag{4}$$

where the initial vector $v_{0,i} = g(x_0)$, $i = 1, 2, \cdots, d$. Note that applying the loss scalar $g(x)$, which is the second term of the gradient decomposition, directly results in two disadvantages: First, $g(x)$ is a scalar for a certain x so that we fail to achieve a non-uniform adaptive learning rate. Besides, if so, the update rule of x would exactly equal to SGD or SHB. Therefore, the first formula in Equation 4 employs the Euler's method to approximate $g(x)$ linearly to achieve the approximate loss. The aforementioned update rules directly yield

$$v_t = \frac{v_{t-1}}{1 + 2\eta\big(\nabla g(x_t)\big)^2}, \tag{5}$$

where we note that v_t decreases monotonically, similar to that of AEGD. Differently, we employ the heavy ball scheme of gradients to accelerate convergence and have the following update rule of the direction

$$\begin{aligned} m_t &= \gamma m_{t-1} + \nabla g(x_t), \\ x_{t+1} &= x_t - \eta v_t m_t, \end{aligned} \tag{6}$$

where $\gamma \in (0, 1)$ is a momentum constant.

Equation 5 and 6 are all update schemes of our Stochastic Euler Heavy Ball method (SEHB). Note that v and m, two terms from the decomposition of the gradient, help to achieve adaptive learning rates and accelerate the convergence SEHB. The pseudo code is showed in Algorithm 1.

Algorithm 1 SEHB (good initialization: $c = 1$, $\gamma = \{0.9, 0.99\}$, $\eta = 0.03$)

Input: $x_1 \in \mathcal{X} \subseteq \mathbb{R}^d$, step size η, c, momentum γ
Initialize $m_{0,i} = 0$, $v_{0,i} = \sqrt{f(x_1) + c}$, $i = 1, 2, \cdots, d$
for $t = 1$ **to** T **do**
  $g \leftarrow \nabla f(x_t) / 2\sqrt{f(x_t) + c}$
  $v_t \leftarrow v_{t-1} / (1 + 2\eta g^2)$
  $m_t \leftarrow \gamma m_{t-1} + g$
  $x_{t+1} \leftarrow x_t - 2\eta v_t m_t$
end for

IV. Convergence Analysis

We discuss the convergence of SEHB in both convex and non-convex situations. The convergence under the condition of convex objective functions is showed in the online convex optimization framework [8], [12], [21], [22] which is similar to Adam [11], AMSGrad [12], AdaBound [14] and AdaBelief [17]. Furthermore, we analyze the convergence in the stochastic non-convex optimization problem, which is similar to the previous work [17], [23]. This situation is more in line with actual scenarios of machine learning and deep learning tasks. To avoid the repetition of notation, we use l instead of g as the composite function of f.

A. Online Convex Optimization

In online optimization, we have a loss function $f_t : \mathcal{X} \to \mathbb{R}$. After a decision $x_t \in \mathcal{X}$ is picked by the algorithm, we have the following regret to minimize:

$$R(T) = \sum_{i=0}^{T} f_t(x_t) - \min_{x \in \mathcal{X}} \sum_{i=0}^{T} f_t(x). \tag{7}$$

The standard assumptions [8], [12], [21], [22] in the setting of online convex optimization are as follows:

Assumption 1. *(1) $\mathcal{X} \subseteq \mathbb{R}^d$ is a compact convex set; (2) f_t is a convex lower semi-continuous (lsc) function, $g_t \in \partial f_t(x_t)$; (3) $D = \max_{x,y \in \mathcal{X}} ||x - y||$, $G = \max_t ||g_t||$.*

We propose the following lemma:

Lemma 1. *f_t is a lower bounded function and $c > 0$ is a constant s.t. $f_t(x) + c > 0, x \in \mathcal{X}$. Let $l_t(x) = \sqrt{f_t(x) + c}$. If f_t has bounded gradients, then l_t has bounded gradients too and is bounded in the feasible regions.*

Remark 1. *The above lemma shows that two terms from the decomposition of the gradient $\nabla f_t(x)$ are both bounded. In particular, the assumption (3) in the standard assumptions 1 yields $||\nabla l_t(x_t)|| \le G$, $|l_t(x_t)| \le L$, $x_t \in \mathcal{X}$.*

Therefore, we can get the following assumptions for SEHB which are entirely yielded from the standard assumptions 1:

Assumption 2. *(1) $\mathcal{X} \subseteq \mathbb{R}^d$ is a compact convex set; (2) f_t is a convex lsc function, $l_t = \sqrt{f_t + c}$, $c > 0$; (3) $||x - y|| \le D$, $||\nabla l_t(x_t)|| \le G$, $|l_t(x_t)| \le L$, $x_t \in \mathcal{X}$.*

The key results are as follows:

Theorem 1. *Under the Assumption 2, let $\gamma \in (0, 1)$, $\gamma_t = \gamma \lambda^{t-1}$ and $\eta_t = \frac{\eta}{\sqrt{t}}$, $\eta > 0$, SEHB has the following bound on the regret:*

$$\begin{aligned} R(T) \le & \frac{LD^2\sqrt{T}}{4\eta} \sum_{i=1}^{d} v_{T,i}^{-1} + \frac{LD^2\gamma^2}{\eta(1-\lambda^2)^2} \\ & + \frac{\eta G^2 L(1+L)}{1-\gamma}(2\sqrt{T} - 1). \end{aligned} \tag{8}$$

Remark 2. *Theorem 1 implies the regret of SEHB is upper bounded by $O(\sqrt{T})$, similar to Adam [11], AMSGrad [12], AdaBound [14] and AdaBelief [17]. Besides, the condition of γ_t can be relaxed to $\gamma_t = \gamma/\sqrt{t}$ and still ensures a regret bound of $O(\sqrt{T})$.*

B. Stochastic Non-convex Optimization

We discuss the convergence in the stochastic non-convex learning which is more in line with actual scenarios of machine learning and deep learning tasks than the online convex optimization. The standard assumptions [17], [23] are as follows:

Assumption 3. *(1) f is lower bounded and differentiable, $||\nabla f(x) - \nabla f(y)|| \leq L||x-y||, \forall x, y$; (2) The noisy gradient is unbiased, and has independent noise, i.e. $g(t) = \nabla f(\theta_t) + \zeta_t$, $\mathbb{E}\zeta_t = 0$, $\zeta_t \perp \zeta_j$, $\forall t, j \in \mathbb{N}$, $t \neq j$; (3) At step t, the algorithm can access a bounded noisy gradient, and the true gradient is also bounded, i.e. $||\nabla f(\theta_t)|| \leq H$, $||g_t|| \leq H$, $\forall t > 1$.*

Similarly, the above assumptions yield the following assumptions for SEHB according to Lemma 1:

Assumption 4. *(1) f is lower bounded and differentiable, $||\nabla f(x) - \nabla f(y)|| \leq L||x-y||, \forall x, y$; (2) The noisy gradient is unbiased, and has independent noise, i.e. $g_t = \nabla f(\theta_t) + \zeta_t$, $\mathbb{E}\zeta_t = 0$, $\zeta_t \perp \zeta_j$, $\forall t, j \in \mathbb{N}$, $t \neq j$; (3) At step t, the algorithm can access a bounded noisy gradient, and the true gradient is also bounded, i.e. $||l_t|| \leq L$, $||\hat{g}_t|| \leq G$, $\hat{g}_t$ is the noisy gradient of $l_t = \sqrt{f_t + c}$, $\forall t > 1$.*

The key result is as follows:

Theorem 2. *Under the Assumption 4, let $\gamma_t < \gamma \leq 1$, $||v_T||_1 \geq \epsilon$ and $\eta_t = \frac{\eta}{\sqrt{t}}$, $\eta > 0$, SEHB satisfies*

$$\begin{aligned}\min_{t\in[T]} \mathbb{E}\Big(||\nabla f(\theta_t)||^2\Big) \leq & \frac{L^2}{\epsilon\eta\sqrt{T}}\Big(C_1 L^2\eta^2 G^2(1+\log T) \\ & + C_2 d\eta + C_3 d L^2\eta^2 + C_4\Big),\end{aligned} \tag{9}$$

where C_1, C_2, C_3 are constants independent of d and T, C_4 is a constant independent of T.

Remark 3. *Theorem 2 implies that SEHB has a $O(\log T/\sqrt{T})$ convergence rate in the stochastic non-convex situation which is similar to Adam-type methods [17], [23]. Besides, Theorem 3.1 in [23] needs to specify the bound of each update, but SEHB needs not. The proof follows the general framework in [23], and it's possible the above bound is loose. A sharper convergence analysis remains open.*

V. Experiments

In this section, we study the generalization performance of our methods and several representative optimization methods. Except SGD with momentum (SHB), we additionally test two families of optimizers including Adam-type methods and other adaptive methods. The former includes Adam, AMSGrad, AdaBound, AdaBelief and the latter includes AEGD and our method SEHB. We conduct experiments on the two classical high-dimensional functions and popular deep learning tasks for testing the performance in thestochastic situation. Particularly, several neural network structures will be chosen including multilayer perceptron, deep convolution neural network and deep recurrent neural network. Concretely, we focus on the following experiments: multilayer perceptron (MLP) on MNIST dataset [3]; ResNet-50 [24] and DenseNet-121 [25] on CIFAR-10 dataset [26]; LSTMs on Penn Treebank dataset [27].

A. Details

Hyperparameters For SHB and AEGD, we employ the grid search for learning rates in $\{1, 0.5, 0.3, 0.1, 0.01\}$. We set momentum γ in SHB to the default value 0.9. Note that reported in [17], the best learning rate for SHB is 30 for LSTMs on Penn Treebank dataset, and we follow this setting. For Adam, AMSGrad, AdaBound and AdaBelief, we employ the grid search for learning rates in $\{0.1, 5e{-}2, 1e{-}2, 5e{-}3, 1e{-}3\}$. We turn over β_1 values of $\{0.9, 0.99\}$ and β_2 values of $\{0.99, 0.999\}$. For other parameters in above Adam-type methods, we follow the setting reported in [14], [17] for achieving the best performance on CIFAR-10 and Penn Treebank dataset and use the default values for other experiments. For SEHB, we use the default value of hyperparameters, the default learning rate for CIFAR-10 and a warm-up learning rate for LSTMs.

Numerical Experiments We conduct numerical experiments on the two classical high-dimensional functions, Extended Powell Singular function and Extended Rosenbrock function, to test the performance of our method in a convex and non-convex function respectively. The former is a high-dimensional convex function, but the Hessian has a double singularity at the solution so that in the global optimization literature this function is stated as a difficult test case. Extended Powell Singular function is as follows:

$$\begin{aligned}f(\mathbf{x}) = \sum_{i=1}^{N/4} \Big(& (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 \\ & + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4\Big),\end{aligned}$$

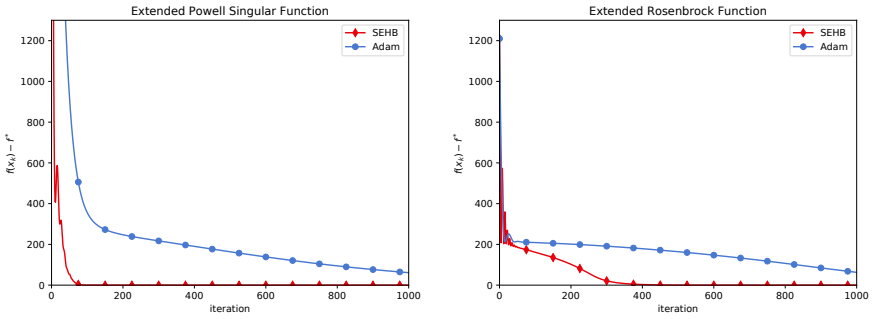
where we set $N = 100$ and the initial point is $(3, -1, 0, 1, 3, -1, 0, 1, \cdots, 3, -1, 0, 1)^T$. The exact optimal solution is $(0, 0, \cdots, 0)^T$ and the exact minimum value of Extended Powell Singular function is 0. The latter, Extended Rosenbrock function, is a famous high-dimensional non-convex function in optimization. It has the following form:

$$f(\mathbf{x}) = \sum_{i=1}^{N/2} \Big(100(x_{2i-1}^2 - x_{2i})^2 + (x_{2i} - 1)^2\Big)$$

where we set $N = 100$. We start with initial point $x_0 = (-1.2, 1, -1.2, 1, \cdots, -1.2, 1)^T$ to find the optimal solution. The exact optimal solution is $(1, 1, \cdots, 1)^T$ and the exact minimum value of Extended Rosenbrock function is 0. We test Adam and our method SEHB in these two functions and the learning rates are $1e-2$ for Adam and $1e-4$ for SEHB. The results are reported in Figure 1(a). We note that SEHB converge faster than Adam with even smaller learning rates.

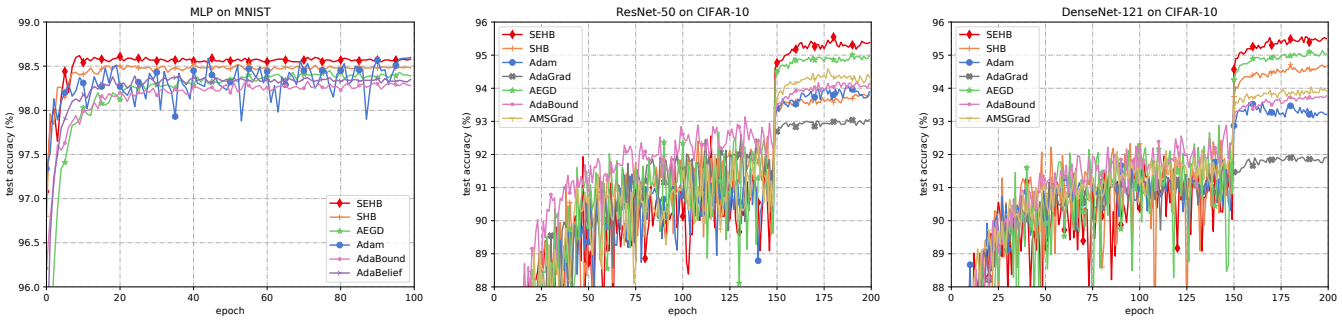
MLP on MNIST We conduct the experiment to test the performance of aforementioned optimizers with MLP on MNIST. We follow the experiment settings reported in AdaBound [14]. MLP is a fully connected neural network with only one hidden layer and total epoch is 100. Figure 2(a) shows the empirical result. Note that all optimization algorithms achieve a test error below 2% and our method SEHB and AMSGrad achieve slightly better performance than other methods on the test set.

ResNet-50 and DenseNet-121 on CIFAR-10 CIFAR-10 is a more complex dataset than MNIST. We use more advanced and powerful deep convolution neural networks, including ResNet-50 and DenseNet-121, to test various optimization



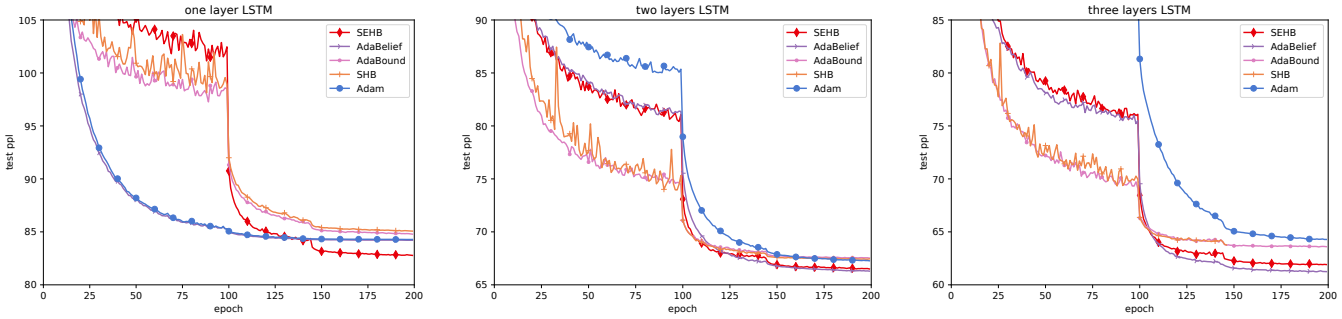
(a) Extended Powell Singular function (b) Extended Rosenbrock function

Fig. 1. Performance comparison on Extended Powell Singular function and Extended Rosenbrock function.



(a) MLP on MNIST (b) ResNet-50 on CIFAR-10 (c) DesNet-121 on CIFAR-10

Fig. 2. Performance of various optimizers in popular CV tasks. The higher is better.



(a) one layer LSTM on PTB (b) two layers LSTM on PTB (c) three layers LSTM on PTB

Fig. 3. Performance of various optimizers in popular NLP tasks. The lower is better.

methods in this classification task on CIFAR-10 dataset. We employ the fixed budget of 200 epochs, set the mini-batch size to 128. Figure 2(b) and 2(c) show the empirical results. Code is modified from the official implementation of AdaBound. As expected, the overall performances of each algorithm on ResNet-50 are similar to those on DenseNet-121. We note that SEHB shows the best generalization performance on DenseNet-121. For ResNet-50, the error of SEHB is higher than that of Adam with a margin 1.3%; For DenseNet-121, SEHB surpasses Adam with a margin 1.7%. We find that classical adaptive methods show rapid descent in the early period of training such as Adam. However, they show mediocre generalization ability on the test set. The empirical results show that our method overcomes the above drawback and achieves even better generalization performance than SHB.

LSTMs on Penn Treebank dataset We test our method on Penn Treebank dataset with one-layer LSTM, two-layers LSTM and three-layers LSTM respectively. We follow the setting of experiments in AdaBelief [17]. One difference is that AdaBelief improves these experiments by setting learning rate scheduling at epoch 100 and 145 in their official implementation. Except our methods, the results of other methods are reported in

AdaBelief. Code is modified from the official implementation of AdaBelief. The perplexities (ppl) of methods are reported in Figure 3(a), 3(b) and 3(c) except AEGD and AMSGrad due to their worse performances. To our knowledge, AdaBelief has been the best optimizer on Penn Treebank dataset. Note that our method SEHB achieves the similar performance to AdaBelief in all three experiments: For one-layer LSTM, SEHB surpasses AdaBelief and achieves the lowest perplexity; For two-layers LSTM, SEHB and AdaBelief show the best performance and SEHB is higher than AdaBelief by a margin 0.28; For three-layers LSTM, SEHB shows the lower perplexity than other methods except AdaBelief and is higher than AdaBelief by a margin 0.73. Considering perplexities equal to e^{loss}, the gap between AdaBelief and SEHB is very small.

VI. Conclusion

We have introduced SEHB, a simple and computationally efficient adaptive algorithm for non-convex stochastic optimization. This method is aimed towards large-scale optimization problems in the sense of large datasets and/or high-dimensional parameter spaces such as machine learning and deep neural networks. The practical intuition and excellent performance of SEHB show that our method is worth further research.

Despite excellent performance of our method, there still remains several directions to explore in the future:

- First, we prove a $O(\log T/\sqrt{T})$ bound of our method SEHB in non-convex setting. However, empirical results show that the generalization performance of SEHB is better than many methods with a similar bound such as Adam. A tighter regret bound of SEHB needs to be explored in the future.
- Furthermore, as mentioned before, SEHB and Adam can be integrated with each other. This topic remains open.
- Finally, several works aim to find a new way to generate adaptive learning rates different from Adam-type methods. Thus, as this kind of works increase, how to measure the quality of adaptive learning rates is more and more important. However, there are few works on this topic.

Acknowledgment

This work was supported by NSFC Tianyuan Fund for Mathematics (No. 12026606), and National Key R&D Program of China (No. 2018AAA0100300).

Appendix

A. Proof of Lemma 1

Proof. Because $l_t(x_t) = \sqrt{f_t(x_t) + c}$, $x_t \in \mathcal{X}$, then

$$\nabla f_t(x_t) = 2l_t(x_t)\nabla l_t(x_t).$$

If $\nabla f_t(x_t)$ is bounded, $l_t(x_t)\nabla l_t(x_t)$ is bounded. Therefore, at least one of $l_t(x_t)$ and $\nabla l_t(x_t)$ is bounded.

First, consider $l_t(x_t)$ is bounded and $\nabla l_t(x_t)$ is unbounded, the only possible case is that

$$l_t(x_t) \to 0, \quad ||\nabla l_t(x_t)|| \to \infty.$$

Obviously, this case doesn't exist because if $l_t(x_t) \to 0$, l_t must have gradients with a limit of 0.

Second, consider $\nabla l_t(x_t)$ is bounded and $l_t(x_t)$ is unbounded, the only possible case is that

$$l_t(x_t) \to \infty, \quad ||\nabla l_t(x_t)|| \to 0.$$

However, if $l_t(x_t)$ has gradients with a limit of 0, $l_t(x_t)$ must be finite. Thus, this case doesn't exist too.

Finally, the only case is that $l_t(x_t)$ is bounded and $\nabla l_t(x_t)$ is bounded too. This proves that if f_t has bounded gradients, then l_t has bounded gradients too and is bounded in the feasible regions. □

B. Proof of Theorem 1

Proof. We first replace the element-wise product with the dialog matrix and obtain

$$x_{t+1} = x_t - 2\eta V_t m_t,$$

where $V_t = \mathrm{diag}\{v^*\}$ and V_t is monotonic decreasing. We aim to minimize the following regret:

$$R(T) = \sum_{i=1}^{T} f_t(x_t) - \min_{x\in\mathcal{X}} \sum_{i=1}^{T} f_t(x).$$

Let $x^* \in \mathcal{X}$ be the optimal solution, the above regret is

$$\begin{aligned} R(T) &= \sum_{i=1}^{T} f_t(x_t) - \sum_{i=1}^{T} f_t(x^*) \\ &\le \sum_{i=1}^{T} \big\langle \nabla f_t(x_t), x_t - x^* \big\rangle. \end{aligned}$$

Because

$$\begin{aligned} &||V_{t+1}^{-\frac{1}{2}}(x_{t+1} - x^*)||^2 \le ||V_t^{-\frac{1}{2}}(x_t - 2\eta_t V_t m_t - x^*)||^2 \\ =&||V_t^{-\frac{1}{2}}(x_t - x^*)||^2 + 4\eta_t^2||V_t^{\frac{1}{2}} m_t||^2 - 4\eta_t\gamma\big\langle m_{t-1}, x_t - x^*\big\rangle \\ &- 4\eta_t\big\langle \nabla l_t(x_t), x_t - x^*\big\rangle, \end{aligned}$$

we have

$$\begin{aligned} &\big\langle \nabla l_t(x_t), x_t - x^*\big\rangle \\ \le&\frac{1}{4\eta_t}\big(||V_t^{-\frac{1}{2}}(x_t - x^*)||^2 - ||V_t^{-\frac{1}{2}}(x_{t+1} - x^*)||^2\big) \\ &+ \eta_t||V_t^{\frac{1}{2}} m_t||^2 - \gamma\big\langle m_{t-1}, x_t - x^*\big\rangle \\ \le&\frac{1}{4\eta_t}(||V_t^{-\frac{1}{2}}(x_t - x^*)||^2 - ||V_t^{-\frac{1}{2}}(x_{t+1} - x^*)||^2) \\ &+ \eta_t||V_t^{\frac{1}{2}} m_t||^2 + \frac{\gamma^2}{\eta_t}||x_t - x^*||^2 + \eta_t||m_{t-1}||^2, \end{aligned}$$

where the last inequality follows Cauchy-Schwarz inequality and Young's inequality. Thus, we obtain

$$\begin{aligned} &\big\langle \nabla f_t(x_t), x_t - x^*\big\rangle \le \frac{l_t(x_t)}{4\eta_t}(||V_t^{-\frac{1}{2}}(x_t - x^*)||^2 \\ &- ||V_t^{-\frac{1}{2}}(x_{t+1} - x^*)||^2) + \eta_t l_t(x_t)||V_t^{\frac{1}{2}} m_t||^2 \\ &+ \frac{\gamma^2 l_t(x_t)}{\eta_t}||x_t - x^*||^2 + \eta_t l_t(x_t)||m_{t-1}||^2, \end{aligned}$$

and the regret is as follows:

$$\begin{aligned}R(T) \leq & \sum_{t=1}^{T} \frac{l_t(x_t)}{4\eta_t}(||V_t^{-\frac{1}{2}}(x_t - x^*)||^2 - ||V_t^{-\frac{1}{2}}(x_{t+1} - x^*)||^2) \\ & + \sum_{t=1}^{T} \eta_t l_t(x_t)||V_t^{\frac{1}{2}} m_t||^2 + \sum_{t=1}^{T} \frac{\gamma^2 l_t(x_t)}{\eta_t}||x_t - x^*||^2 \\ & + \sum_{t=1}^{T} \eta_t l_t(x_t)||m_{t-1}||^2.\end{aligned}$$

We divide the right formula into three parts:

$$\begin{aligned}P_1 &= \sum_{t=1}^{T} \frac{l_t(x_t)}{4\eta_t}(||V_t^{-\frac{1}{2}}(x_t - x^*)||^2 - ||V_t^{-\frac{1}{2}}(x_{t+1} - x^*)||^2), \\ P_2 &= \sum_{t=1}^{T} \frac{\gamma^2 l_t(x_t)}{\eta_t}||x_t - x^*||^2, \\ P_3 &= \sum_{t=1}^{T} \eta_t l_t(x_t)||V_t^{\frac{1}{2}} m_t||^2 + \sum_{t=1}^{T} \eta_t l_t(x_t)||m_{t-1}||^2.\end{aligned}$$

Consider the part 1 and apply Assumption 2:

$$\begin{aligned}P_1 =& \sum_{t=1}^{T} \frac{l_t(x_t)}{4\eta_t}(||V_t^{-\frac{1}{2}}(x_t - x^*)||^2 - ||V_t^{-\frac{1}{2}}(x_{t+1} - x^*)||^2) \\ \leq & \frac{L}{4\eta}\Big(||V_1^{-\frac{1}{2}}(x_1 - x^*)||^2 + \sum_{t=2}^{T}(\sqrt{t}||V_t^{-\frac{1}{2}}(x_t - x^*)||^2 \\ & - \sqrt{t-1}||V_{t-1}^{-\frac{1}{2}}(x_t - x^*)||^2)\Big) \\ \leq & \frac{LD^2}{4\eta}\Big(\sum_{i=1}^{d} v_{1,i}^{-1} + \sum_{t=2}^{T}\sum_{i=1}^{d}(\sqrt{t}v_{t,i}^{-1} - \sqrt{t-1}v_{t-1,i}^{-1})\Big) \\ = & \frac{LD^2\sqrt{T}}{4\eta}\sum_{i=1}^{d} v_{T,i}^{-1}.\end{aligned}$$

Then, the part 2 is as follows:

$$P_2 = \sum_{t=1}^{T} \frac{\gamma^2 l_t(x_t)}{\eta_t}||x_t - x^*||^2 \leq \frac{LD^2\gamma^2}{\eta(1-\lambda^2)^2}.$$

Finally, we give the upper bound of the part 3 by applying Assumption 2:

$$\begin{aligned}P_3 =& \sum_{t=1}^{T} \eta_t l_t(x_t)||V_t^{\frac{1}{2}} m_t||^2 + \sum_{t=1}^{T} \eta_t l_t(x_t)||m_{t-1}||^2 \\ \leq & \eta L \sum_{t=1}^{T} \frac{1}{\sqrt{t}}(||V_t^{\frac{1}{2}} m_t||^2 + ||m_{t-1}||^2) \\ \leq & \frac{\eta L G^2}{1-\gamma}\sum_{t=1}^{T} \frac{1}{\sqrt{t}}(v_t + 1) \\ \leq & \frac{\eta L G^2(1+v_0)}{1-\gamma}\sum_{t=1}^{T} \frac{1}{\sqrt{t}} \\ \leq & \frac{\eta G^2 L(1+L)}{1-\gamma}(2\sqrt{T} - 1)\end{aligned}$$

Hence, we get the final regret bound:

$$\begin{aligned}R(T) \leq & \frac{LD^2\sqrt{T}}{4\eta}\sum_{i=1}^{d} v_{T,i}^{-1} + \frac{LD^2\gamma^2}{\eta(1-\lambda^2)^2} \\ & + \frac{\eta G^2 L(1+L)}{1-\gamma}(2\sqrt{T} - 1).\end{aligned}$$

□

C. Proof of Theorem 2

Proof. The proof follows the general framework in [23]. In particular, we follow the proof in [23] and use the Theorem 3.1 in [23]. We note that Theorem 3.1 in [23] gives the convergence bound for generalize Adam [23]. However, this general framework needs no EMA mechanism, i.e. squared gradients and represents more general adaptivity which uses first order gradients. SEHB belongs to this general framework with $1/\sqrt{v_t}$ in this general framework corresponding to v_t in SEHB. Therefore, we can apply the aforementioned theorem to our proof. According to the above theorem and Assumption 4, we obtain

$$\begin{aligned}& \mathbb{E}\Big(\sum_{t=1}^{T} \eta_t \langle \nabla l(\theta_t), V_t \nabla l(\theta_t)\rangle\Big) \\ \leq & \mathbb{E}\Big(C_1 \sum_{t=1}^{T} \Big|\Big|\eta_t V_t \hat{g}_t\Big|\Big|^2 + C_2 \sum_{t=2}^{T} \Big|\Big|v_t\eta_t - v_{t-1}\eta_{t-1}\Big|\Big|_1 \\ & + C_3 \sum_{t=2}^{T-1} \Big|\Big|v_t\eta_t - v_{t-1}\eta_{t-1}\Big|\Big|^2\Big) + C_4,\end{aligned}$$

where C_1, C_2, C_3 are constants independent of d and T, C_4 is a constant independent of T. We divide the right formula to three parts:

$$\begin{aligned}P_1 &= \mathbb{E}\sum_{t=1}^{T} \Big|\Big|\eta_t V_t \hat{g}_t\Big|\Big|^2, P_2 = \mathbb{E}\sum_{t=2}^{T} \Big|\Big|v_t\eta_t - v_{t-1}\eta_{t-1}\Big|\Big|_1, \\ P_3 &= \mathbb{E}\sum_{t=2}^{T-1} \Big|\Big|v_t\eta_t - v_{t-1}\eta_{t-1}\Big|\Big|^2.\end{aligned}$$

First, we give the upper bound of the part 1 according to Assumption 4:

$$P_1 \leq L^2\eta^2\mathbb{E}\sum_{t=1}^{T} \Big|\Big|\frac{\hat{g}_t}{\sqrt{t}}\Big|\Big|^2 \leq L^2\eta^2G^2(1 + \log T),$$

where the last inequality is due to $\sum_{t=1}^{T} 1/t \leq 1 + \log T$. Then, consider the part 2:

$$\begin{aligned}P_2 =& \mathbb{E}\sum_{t=2}^{T} v_{t-1}\eta_{t-1} - v_t\eta_t = \mathbb{E}\sum_{j=1}^{d} \eta_1 v_{1,j} - \eta_T v_{T,j} \\ \leq & \mathbb{E}\sum_{j=1}^{d} \eta_1 v_{1,j} \leq dL\eta.\end{aligned}$$

Finally, the part 3 is as follows:

$$\begin{aligned}P_3 =&\mathbb{E}\sum_{t=2}^{T-1}||\eta_t v_t-\eta_{t-1}v_{t-1}||_1||\eta_t v_t-\eta_{t-1}v_{t-1}||_1\\ \leq& L\eta\mathbb{E}\sum_{t=2}^{T-1}||\eta_t v_t-\eta_{t-1}v_{t-1}||_1\leq dL^2\eta^2.\end{aligned}$$

Hence, we obtain

$$\begin{aligned}&\mathbb{E}\Big(C_1\sum_{t=1}^{T}\Big|\Big|\eta_t V_t\hat{g}_t\Big|\Big|^2+C_2\sum_{t=2}^{T}\Big|\Big|v_t\eta_t-v_{t-1}\eta_{t-1}\Big|\Big|_1\\&+C_3\sum_{t=2}^{T-1}\Big|\Big|v_t\eta_t-v_{t-1}\eta_{t-1}\Big|\Big|^2\Big)+C_4\end{aligned}$$

Now we lower bound the LHS. With the assumption $||v_T||_1\geq c$, we have

$$(\eta_t V_t)_j=\frac{\eta(V_t)_j}{\sqrt{t}}\geq\frac{\eta c}{\sqrt{t}},$$

and thus

$$\begin{aligned}&\mathbb{E}\Big(\sum_{t=1}^{T}\eta_t\big\langle\nabla l(\theta_t),V_t\nabla l(\theta_t)\big\rangle\Big)\\&\geq\mathbb{E}\Big(\sum_{t=1}^{T}\frac{\eta c}{\sqrt{t}}||\nabla l(\theta_t)||^2\Big)\geq\frac{\eta c}{L^2}\mathbb{E}\sum_{t=1}^{T}||\nabla f(\theta_t)||^2\\&\geq\frac{\eta c\sqrt{T}}{L^2}\min_{t\in[T]}\mathbb{E}||\nabla f(\theta_t)||^2.\end{aligned}$$

We finally obtain:

$$\begin{aligned}&\min_{t\in[T]}\mathbb{E}\Big(||\nabla f(\theta_t)||^2\Big)\\\leq&\frac{L^2}{c\eta\sqrt{T}}\Big(C_1L^2\eta^2G^2(1+\log T)+C_2dL\eta+C_3dL^2\eta^2+C_4\Big)\end{aligned}$$

This proves Theorem 2. □

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems (NIPS)*, 2012, p. 1097–1105.

[2] A. Graves, A.-r. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, p. 6645–6649.

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, p. 2278–2324.

[4] T. M. Heskes and B. Kappen, "On-line learning processes in artificial neural networks," ser. North-Holland Mathematical Library, J. Taylor, Ed. Elsevier, 1993, vol. 51, pp. 199 – 233.

[5] Y. LeCun, L. Bottou, G. Orr, and K. Müller, *Efficient backprop*, ser. Lecture Notes in Computer Science, 2012, pp. 9–48.

[6] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," in *USSR Computational Mathematics and Mathematical Physics*, 1964, pp. 4:791–803.

[7] Y. Nesterov, "A method of solving a convex programming problem with convergence rate O(1/sqrt(k))," in *Soviet Mathematics Doklady*, 1983, pp. 27:372–376.

[8] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," in *Journal of Machine Learning Research (JMLR)*, 2011.

[9] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," in *CoRR*, 2012.

[10] T. Tieleman and G. Hinton, "RMSprop: Divide the gradient by a running average of its recent magnitude," in *COURSERA: Neural networks for machine learning*, 2012.

[11] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[12] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," in *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[13] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar, "Adaptive methods for nonconvex optimization," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018.

[14] L. Luo, Y. Xiong, Y. Liu, and X. Sun, "Adaptive gradient methods with dynamic bound of learning rate," in *Proceedings of International Conference on Learning Representations (ICLR)*, 2019.

[15] J. Chen, D. Zhou, Y. Tang, Z. Yang, Y. Cao, and Q. Gu, "Closing the generalization gap of adaptive gradient methods in training deep neural networks," in *International Joint Conferences on Artificial Intelligence*, 2020.

[16] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," in *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)*, April 2020.

[17] J. Zhuang, T. Tang, Y. Ding, S. Tatikonda, N. Dvornek, X. Papademetris, and J. Duncan, "Adabelief optimizer: Adapting stepsizes by the belief in observed gradients," *Conference on Neural Information Processing Systems*, 2020.

[18] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019.

[19] Y. Malitsky and K. Mishchenko, "Adaptive gradient descent without descent," in *CoRR*, 2017.

[20] H. Liu and X. Tian, "AEGD: Adaptive gradient decent with energy," 2020.

[21] E. Hazan, "Introduction to online convex optimization," *CoRR*, vol. abs/1909.05207, 2019. [Online]. Available: http://arxiv.org/abs/1909.05207

[22] A. Alacaoglu, Y. Malitsky, P. Mertikopoulos, and V. Cevher, "A new regret analysis for adam-type algorithms," in *International Conference on Machine Learning*, 2020, pp. 202–210.

[23] X. Chen, S. Liu, R. Sun, and M. Hong, "On the convergence of a class of adam-type algorithms for non-convex optimization," 2019.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[25] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[26] A. Krizhevsky and G. E. Hinton, "Learning multiple layers of features from tiny images," in *Technical report*, 2009.

[27] M. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The penn treebank," 1993.