Pattern Recognition Letters 128 (2019) 197–203

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Barzilai–Borwein-based adaptive learning rate for deep learning

Jinxiu Liang[a], Yong Xu[a,b], Chenglong Bao[c], Yuhui Quan[a,*], Hui Ji[d]

[a] South China University of Technology, Guangzhou 510006, China
[b] Peng Cheng Laboratory, Shenzhen 510852, China
[c] Tsinghua University, Beijing 100084, China
[d] National University of Singapore, Singapore 117543, Singapore

ARTICLE INFO

Article history:
Received 21 June 2019
Revised 28 August 2019
Accepted 29 August 2019
Available online 30 August 2019

Keywords:
Barzilai–Borwein method
Deep neural network
Stochastic gradient descent
Adaptive learning rate

ABSTRACT

Learning rate is arguably the most important hyper-parameter to tune when training a neural network. As manually setting right learning rate remains a cumbersome process, adaptive learning rate algorithms aim at automating such a process. Motivated by the success of the Barzilai–Borwein (BB) step-size method in many gradient descent methods for solving convex problems, this paper aims at investigating the potential of the BB method for training neural networks. With strong motivation from related convergence analysis, the BB method is generalized to adaptive learning rate of mini-batch gradient descent. The experiments showed that, in contrast to many existing methods, the proposed BB method is highly insensitive to initial learning rate, especially in terms of generalization performance. Also, the BB method showed its advantages on both learning speed and generalization performance over other available methods.

1. Introduction

1.1. Background

In the last decade, deep learning has emerged as one leading machine learning tool in computer vision. Particularly, deep neural network (DNN) based learning, including supervised approaches [1,2] and unsupervised approaches [3,4], has been used for solving many long-lasting problems in computer vision with remarkable success, e.g. image classification, action recognition and semantic segmentation. DNN-based learning enables an artificial neural network (NN) to capture intricate structures of visual data with multiple levels of abstraction.

In DNN, once the architecture of an NN has been designed for solving a specific problem, the remaining task is to learn or train the weights of the NN. In the so-called supervised approach to NN training, the weights of an NN are adjusted with respect to input data (i.e. training samples) such that the error between the output of the NN and the preferred output is minimized. More specifically, Let θ denote the set of the weights of an NN. Consider a training set $\{(x_i, y_i)\}_{i=1}^N$ containing N samples, where x_i denotes the input data and y_i denotes its preferred output. The learning process is then to estimate θ that minimizes the following cost function:

$$\min_{\theta} L(\theta) = \frac{1}{N}\sum_{i=1}^{N} L_i(\theta), \tag{1}$$

where $L_i(\theta) = L(x_i, y_i; \theta)$.

An efficient and effective method to find a good solution of the problem (1) is critical to the success of DNN. Unfortunately, the problem (1) is often a very large-scale non-smooth and non-convex problem. For such a large-scale problem, first-order methods such as gradient descent are usually preferred. Among them, the so-called *stochastic gradient descent* (SGD) method is dominant in NN training. Instead of using the batch gradient which leverages over the cost gradients of all training samples $\frac{1}{N}\sum_{i=1}^{N}\nabla_\theta L_i(\theta)$, classic SGD methods only call the cost gradient of one sample, which could be overly noisy. Therefore, a prominent approach is using the so-called *mini-batch gradient descent* method [5], which uses a small portion of training samples for gradient estimation. The update of a mini-batch gradient descent method reads as follows:

$$\theta_{t+1} = \theta_t - \frac{\eta_t}{|\mathcal{B}_t|}\sum_{i\in\mathcal{B}_t}\nabla_\theta L_i(\theta_t), \tag{2}$$

where θ_t denotes the estimate at Iteration t, $\mathcal{B}_t$ denotes the index set of the samples randomly chosen from the training set at Iteration t, $|\mathcal{B}_t|$ denotes the cardinality of the set $\mathcal{B}_t$, and the value $\eta_t > 0$ is called *learning rate*. Mini-batch gradient descent is of core practical importance to NN training. In practice, the gradient

* Corresponding author.
E-mail address: csyhquan@scut.edu.cn (Y. Quan).

https://doi.org/10.1016/j.patrec.2019.08.029
0167-8655/

$\nabla_\theta L_i(\theta)$ is calculated by using a technique called *back-propagation*. When training a DNN, the learning rate η_t is arguably the most important hyper-parameter for achieving good performance, which requires rigorous tune-up [6]. Learning rate has profound effect on the convergence of NN training, as well as generalization performance of the trained NN. In order to make the algorithm converge, one often seen practice is to set learning rate to be decreasing over time.

1.2. Challenges on learning rate setting

The sequence of learning rates has great impact on training efficiency and generalization performance. If learning rates are too large, the training process is not stable. In such a case, the estimate can either overshoot the desired minimum such that they just osculate around the minimum, or does not converge. If learning rates are too small, there are also undesired outcomes. One is related to training inefficiency. Small learning rates make the training process unnecessarily time-consuming, as it only slowly updates the estimates. Another is related to poor generalization performance [7–11]. Such an issue comes from the highly non-convexity of the problem. Although training an NN does not require the sequence converges to a global minimum, those local minima with good generalization performance are usually far away from a random initialization. When using a small learning rate, the sequence tends to be trapped into a local minimum close to the initial. In other words, small learning rates might make the sequence converge to some local minimum whose generalization performance is poor [10,11].

From the discussions above, it can be seen that choosing right learning rates is very important for training an NN with good performance. While there are some good guidelines for manually setting learning rates, it remains a very cumbersome process. In the past, there have been several adaptive learning rate methods proposed for automating such a process, e.g. the widely-used Adam [12] and its improved version AMSGrad [13]. These methods still have a lot of room for improvement when compared to the performance achieved by rigorous manual tune-up. It is empirically observed that these methods are sensitive to the initial value of learning rate. As a result, many trials on setting initial learning rate need to be done when training NNs, which is very time consuming. In short, there is certainly the need to have an automated way of setting good learning rates such that we can efficiently train an NN with good generalization performance.

1.3. Main ideas and contributions

Motivated by the importance of learning rate, this paper aims at developing an adaptive learning rate algorithm for significantly reducing computational effort to train an NN with good generalization performance. In this paper, we transferred the concept of the well-known Barzilai-Borwein (BB) technique [14] in convex optimization to the setting of adaptive learning rate for training NNs. The motivation comes from the fact that for the objective function with second-order continuous derivative, a good step size for the gradient descent is closely related to the eigenvalues of the Hessian matrix at current iteration. For training an NN, estimating these quantities are not only difficult but also expensive in terms of time and storage. The BB method is a well-known technique in optimization that approximates the secant equation by two consecutive points to find a nearly Newton step size. In the bi-variate case, the BB method estimates the curvature along the gradient orientation to obtain appropriate step sizes.

In this paper, the concept of BB method is generalized to the mini-batch gradient descent method for training NNs. There are several advantages of the proposed BB-based adaptive learning rate over the existing ones:

1. The proposed BB method is insensitive to initial learning rate, i.e., in a wide range, different initial learning rates do not impact generalization performance.
2. The proposed BB method has significant gain on the learning speed over other methods in most cases, as well as modest gain on generalization performance in some cases.

2. Related work

Learning rate is known as one of the most important hyper-parameters when training an NN. However, it is also a difficult and cumbersome process to find the right learning rate. Before proceeding, we first introduce some terms used in training NNs. One epoch refers to one forward pass and one backward pass of all the training examples. Batch size is the total number of training examples present in a one forward/backward pass. Iteration is the number of batches needed to complete one epoch.

In the past, several heuristics have been proposed to set learning rates, including how to initialize the learning rate and how to schedule it afterward. The common practice for the initialization of learning rate takes a trial-and-error strategy. One simple way for scheduling learning rate is decreasing it after a fixed period of epochs (step decay) or after the validation accuracy reaches the plateau for several epochs. Another commonly used trick is to change the learning rate with respect to the time t or the epoch k, which includes inverse linear decay over epoch.

There has been a enduring effort on developing the techniques that automate the process of setting learning rates when training DNNs [13]. Most existing adaptive learning rate methods, including the well-known AdaGrad [15], RMSProp [16], Adam [12] and AMSGrad [13], can be expressed in the following form:

$$\theta_{t+1} = \theta_t - \frac{\eta_t}{\sqrt{v_t}} m_t, \quad \text{for} \quad t = 1, 2, \ldots. \tag{3}$$

where m_t is a descent direction derived from the gradients at subsequent time-steps $\{g_1, \ldots, g_T\}$ for updating θ_t, and the value $\frac{\eta_t}{\sqrt{v_t}}$ is the learning rate at time-step t where v_t is derived from the corresponding squared gradients $\{g_1^2, g_2^2, \ldots, g_T^2\}$. However, it is observed that the performance of these adaptive learning rate methods are sensitive to initial learning rate. In other words, different initial learning rates lead to noticeable different training loss and test accuracy. See Fig 1 for an illustration. As a result, it requires many trials on setting right initial learning rates to have an NN trained with fast convergence and good generalization performance, which is a very time-consuming process.

Barzilai and Borwein [14] proposed a strategy to determine the step size for gradient descent methods, which is often called BB step size in the literature. It has been successfully used to solve various types of optimization problems arising from a wide range of applications, including sparse reconstruction [17] and coherence retrieval [18]. The Plain BB method also has been introduced for training NNs [19,20], which are restricted to the non-stochastic gradient descent method. The BB method is introduced in [21] to the SGD and its variance reduced variants SVRG [22] for solving convex problems. It is also extended in [23] for solving non-convex problems.

It is noted that all existing works on the application of the BB method for solving non-convex problems are based on the SVRG, which needs to perform a full gradient evaluation over the entire dataset per epoch. Such a practice is not used in practical deep learning for its computational cost, and it is indeed high inefficient when training a DNN [24]. Different from these existing works, this paper considers the mini-batch version of SGD, the most often

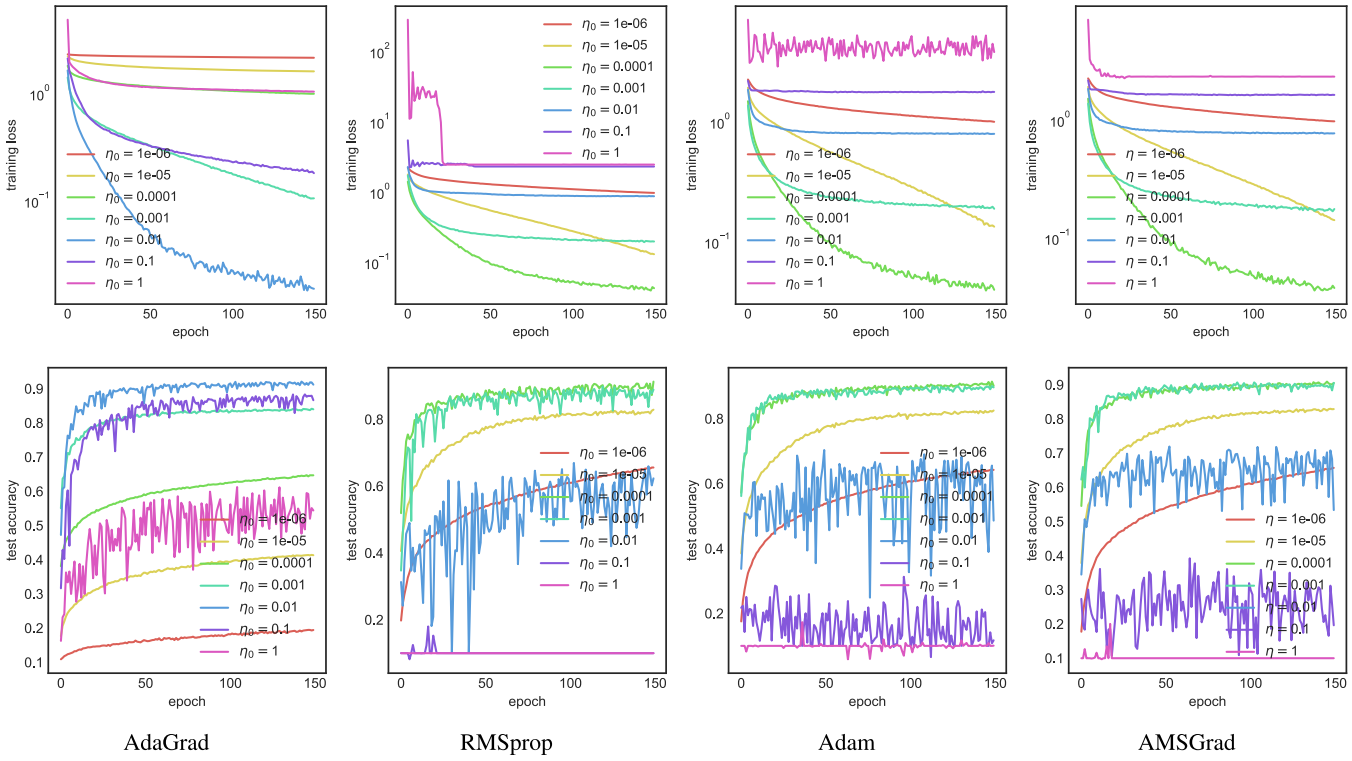

Fig. 1. Performance comparison of four existing methods when using different initial learning rates for training ResNet18 on CIFAR10. It is noted that for RMSProp, the accuracy curves of $\eta_0 = 1$ and $\eta_0 = 0.1$ have very close values that leads to occlusion.

seen one used in practical deep learning. To the best of our knowledge, this paper is the first one that studies the application of the BB method in practical DNN training, including the modifications for fitting practical training procedures in deep learning, convergence analysis and extensive experimental evaluations on several representative NN architectures.

3. BB-based adaptive learning rate

3.1. Preliminaries on BB method

Main idea of the BB method is to use information from the last iterations to determine the step size in the current iteration. Consider an unconstrained optimization problem:

$$\min_{x\in\mathbb{R}^n} f(x),$$

where $f:\mathbb{R}^n\to\mathbb{R}$ has second-order continuous derivatives. Let $\{x_1, x_2, \dots, \}$ denote the sequence generated by the method:

$$x_{t+1} = x_t - \eta_t \nabla f(x_t), \tag{4}$$

where $\eta_t > 0$ is the step size and the gradient $\nabla f(x_t)$ is the search direction. The gradient descent method is simple, but only has linear convergence rate when solving strongly convex problems. To achieve quadratic convergence rate, Newton's method uses the search direction $H^{-1}\nabla f$ where H denotes the Hessian matrix of f. As it is very computationally expensive to calculate the inverse of Hessian matrix of a large-scale problems, Quasi-Newton methods replace Hessian matrix by an approximate $B \approx H^{-1}$. In Quasi-Newton methods, the approximation matrix B satisfies the following secant equation:

$$B_t s_t = y_t,$$

where

$$\begin{cases} s_t = x_t - x_{t-1}, \\ y_t = \nabla f(x_t) - \nabla f(x_{t-1}). \end{cases}$$

The Barzilai–Borwein (BB) method [14] is motivated from the idea of Quasi-Newton methods, which replaces the approximate matrix B_t by an identity matrix multiplied by a scale, i.e. $\eta_t^{-1} I$. Then the optimal value of η_t minimizes the least square error of secant equation:

$$\left\| \eta_t^{-1} s_t - y_t \right\|_2^2,$$

and its explicit solution is

$$\eta_t = \frac{\|s_t\|_2^2}{s_t^\top y_t}.$$

3.2. The BB method for adaptive learning rate

3.2.1. Definition of BB-based learning rate

Let $\nabla L_{\mathcal{B}}(\theta)$ denote the mini-batch gradient used in training neural networks:

$$\nabla L_{\mathcal{B}}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i\in\mathcal{B}} \nabla_\theta L_i(\theta). \tag{5}$$

Let k denote the epoch index, and t denote the index of time-step inside each epoch starting from 1 to T. Then, the sequence $\{\theta_{k,t}\}_{k,t}$ generated from the training is

$$\{\underbrace{\theta_{0,1}, \theta_{0,2}, \dots, \theta_{0,T}}_{k=0}, \underbrace{\theta_{1,1}, \theta_{1,2}, \dots, \theta_{1,T}}_{k=1}, \cdots\}. \tag{6}$$

In the proposed BB method, the learning rate is only updated after finishing one epoch, i.e., the update rule is

$$\theta_{k,0} = \theta_{k-1,T}, \quad \theta_{k,t+1} = \theta_{k,t} - \eta_k \nabla L_{\mathcal{B}_{k,t}}(\theta_{k,t}), \tag{7}$$

for $t = 0, 1, \dots, T-1$ and $k = 0, 1, \dots$. There are two quantities in the BB-method for determining the value η_k: the difference of points s_k and the difference of gradients y_k.

The gradient for estimating η_k is defined in the same way as the Adam method, i.e. the exponential moving average of the stochastic gradients over the epoch:

$$g_{k,t+1} = (1-\beta)g_{k,t} + \beta \nabla L_{\mathcal{B}_t}(\theta_{k,t}), \tag{8}$$

for $t = 0, 1, \ldots, T-1$ and $g_{k,0} = 0$, where β is a predefined constant smoothing factor within (0,1] that controls the degree of exponential decay. Then, we define the difference of gradients between two epochs by

$$y_k = g_{k,T} - g_{k-1,T}. \tag{9}$$

The difference of points over two epochs is defined as the difference of the last one of two epochs normalized by the iterations:

$$s_k = T^{-1}(\theta_{k,T} - \theta_{k-1,T}). \tag{10}$$

Now, we have the BB-based learning rate[1]:

$$\eta_{k+1} = \frac{\|s_k\|^2}{|s_k^\top y_k|}, \tag{11}$$

3.2.2. Related convergence analysis

The randomness of stochastic gradient could make BB step size sometimes too large or too small, and thus it is often used together with line search. It is suggested in [25] to restrict the BB step size when being applied in practice to guarantee the convergence of stochastic gradient descent based algorithms. Indeed, based on Bertsekas and Tsitsiklis [26, Proposition 3], we have the following convergence analysis of the mini-batch stochastic gradient descent method.

Proposition 1. *Let L_is defined in Eq. (1) be continuous and differentiable. Given θ_0, let $\{\theta_t\}$ be the sequence generated by the mini-batch stochastic gradient descent scheme, i.e.*

$$\theta_{t+1} = \theta_t - \eta_t \nabla L_{\mathcal{B}}(\theta_t),$$

where $\nabla L_{\mathcal{B}}(\theta_t)$ is defined in Eq. (5). Define $w_t = \nabla L(\theta_t) - \nabla L_{\mathcal{B}}(\theta_t)$. Assume $\mathbb{E}(\|w_t\|^2) \le A(1 + \|\nabla L(\theta_t)\|^2)$ for some constant A and the learning rate satisfies

$$\sum_{t=1}^{\infty} \eta_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} (\eta_t)^2 < \infty. \tag{12}$$

Then, we have $\lim_{t\to\infty} \nabla L(\theta_t) = 0$.

The proof is done by directly checking the conditions of Bertsekas and Tsitsiklis [26, Proposition 3]. In order to guarantee the convergence of the mini-batch stochastic gradient descent method, one sufficient condition is letting the sequence of learning rates satisfies the decay property (12).

Therefore, when we apply the BB method to generate learning rates, we impose a bound constraint on the BB-based learning rate η_k as follows.

$$\hat{\eta}_k = \begin{cases} \eta_k, & \text{if } \eta_k \in \frac{1}{k+1}[\tau_{\min}, \tau_{\max}], \\ \frac{1}{k+1}\tau_0, & \text{otherwise,} \end{cases} \tag{13}$$

where k denotes the index of epoch, $\tau_{\min}$, $\tau_{\max}$, τ_0 are predefined constants. Clearly, the learning rate $\{\hat{\eta}_k\}$ determined by Eq. (13) satisfies constraint (12) for convergence. See Algorithm 1 for the description of the BB method for adaptive learning rate.

Remark. In practice, it is very rare that the learning rates generated from the BB-method is out of the bound interval with default values. For instance, when training VGG on the dataset CIFAR10 with $\tau_{\max} = 10$ and $\tau_{\min} = 0.1$, only 1/150 of the BB-based learning rates are out of the bound interval.

[1] In stochastic setting, the calculated learning rate η might be negative. Therefore, we use the absolute value for the BB formula.

Algorithm 1 BB-based adaptive learning rate.

Require: max epochs K, steps per epoch T, batch size $|\mathcal{B}|$, weighting parameter $\beta \in (0, 1]$, initial point $\theta_{0,0}$, learning rate $\eta_0 = \eta_1$, τ_0, $\tau_{\min}$ and $\tau_{\max}$.

for $k \leftarrow 0$ to $K-1$ **do**
  if $k > 1$ **then**
    $y_{k-1} \leftarrow g_{k-1,T} - g_{k-2,T}$
    $s_{k-1} \leftarrow \frac{1}{T}(\theta_{k-1,T} - \theta_{k-2,T})$
    $\eta_k \leftarrow \frac{\|s_{k-1}\|^2}{|s_{k-1}^\top y_{k-1}|}$
    $\hat{\eta}_k = \begin{cases} \eta_k & \text{if } \eta_k \in \left[\frac{\tau_{\min}}{k+1}, \frac{\tau_{\max}}{k+1}\right], \\ \frac{\tau_0}{k+1}, & \text{otherwise.} \end{cases}$
  $g_{k,0} \leftarrow 0$
  for $t \leftarrow 0$ to $T-1$ **do**
    Randomly draw a batch $\mathcal{B}_{k,t}$
    $\theta_{k,t+1} \leftarrow \theta_{k,t} - \hat{\eta}_k \nabla L_{\mathcal{B}_{k,t}}(\theta_{k,t})$
    $g_{k,t+1} \leftarrow (1-\beta)g_{k,t} + \beta \nabla L_{\mathcal{B}_{k,t}}(\theta_{k,t})$
  $\theta_{k+1,0} \leftarrow \theta_{k,T}$

4. Experiments

This section is devoted to performance evaluation of the proposed BB method for adaptive learning rate on several representative NN architectures on the task of classification.

4.1. Configuration of experiments

All the experiments are conducted in PyTorch, using the default parameters if no specifications are provided.

4.1.1. Datasets

For evaluation, the proposed BB methods are used for training neural networks for the task of classification. Five benchmark datasets are used in the experiments, namely, MNIST[2] [27], CIFAR10 and CIFAR100[3] [28], ImageNet(a.k.a. ILSVRC-2012) [29] and its downsampled variant ImageNet-32-32[4] [30]. The configuration of these five datasets is summarized in Table 1. For all the five datasets, the input images are normalized by mean and standard deviation per channel. In the CIFAR10 and CIFAR100 datasets, we performed data augmentation with random horizontal flipping additionally. The training-test splitting in our experiment was based on official requirements.

4.1.2. NN Models

The methods are applied on several representative NN models. For classic and shallow NNs, the following models are tested: a multilayer perceptron (MLP) with two fully connected layers on MNIST, and a convolutional neural network (CNN) named LeNet [27] with ReLU activation function on CIFAR10. For deep NNs, the following models are tested[5]: VGG [31] and Residual Neural Networks (ResNet) [32] (We useResNet18 for CIFAR datasets and ResNet50 for ImageNet datasets). The configuration of the models used in the experiments is summarized in Table 2. For MLP and LeNet, we initialized the weights with the method proposed by [33]. As for VGG and ResNet, we follow the same weight initialization strategy as the original paper, except the weights of

[2] http://yann.lecun.com/exdb/mnist/.
[3] https://www.cs.toronto.edu/~kriz/cifar.html.
[4] http://image-net.org/download-images.
[5] The models used in this paper is a reduced version with less layers and less nodes for computational efficiency, which has been implemented in official release of PyTorch and are widely used in the study of deep learning.

Table 1
The characteristics of the datasets.

Dataset	MNIST	CIFAR-10	CIFAR-100	ImageNet-32-32	ImageNet
#Training set	60,000	50,000	50,000	1,281,167	1,281,167
#Test set	10,000	10,000	10,000	50,000	50,000
Resolution	28×28	32×32	32×32	32×32	original size varies; crop to 224×244
#Classes	10	10	100	1000	1000

Table 2
The characteristics of the models.

Model	MLP	LeNet	VGG	ResNet18	ResNet50 for ImageNet-32-32	ResNet50 for ImageNet
#Layer	2	5	11	18	50	50
#Convolutional layer	N/A	2	8	17	49	49
Convolutional kernel	N/A	5×5	3×3	1×1, 3×3	1×1, 3×3	1×1, 3×3, 7×7
#Fully connected layer	2	3	3	1	1	1
#Fully connected unit	1024,1024	400, 120, 84	512, 512, 512	512	512	512
Batch Norm	–	–	+	+	+	+
#Trainable parameters	1,863,690	62,006	9,756,426	11,173,962	25,549,352	25,557,032

convolutional layers, which use the normal initialization proposed by [34] instead.

4.1.3. Methods for comparison

The proposed adaptive learning rate method is compared to several adaptive learning rate methods that are widely used in practice on MNIST and CIFAR datasets: AdaGrad [15], RMSProp [16], Adam [12] and AMSGrad [13]. We also compared the proposed method with the Nesterov Accelerated Gradient (NAG, a.k.a. SGD with Nesterov momentum) method [35], as it has been used as the preferred optimization algorithm for training several representative deep neural networks [31,32,36].

The initialization of the learning rate is best-tuned by densely grid search in $\left[10^{-6}, 10^{0}\right]$. Following experimental protocols in prior works [12,13], fixed learning rates are adopted for training neural networks. All methods are implemented using weight decay regularization with the regularization parameter $\lambda = 5 \times 10^{-4}$. In addition, we verified the performance of the proposed method for training ResNet50 on the large-scale image dataset ImageNet. For ImageNet-32-32, the proposed method is compared with NAG with learning rate initialized with 0.01 and dropped by a factor of 0.5 every 10 epochs following [30]. For ImageNet, following [32] and [37], the proposed method is compared with NAG whose learning rate is initialized with 0.1 and dropped by a factor of 0.1 every 30 epochs. The weight decay regularization with the regularization parameter $\lambda = 1 \times 10^{-4}$ is used.

4.1.4. Configuration of the BB method

For the proposed BB method, the weighting parameter $\beta = 4/T$. The parameter $\tau_0 = 1$ for all tasks, while $\tau_{\max} = 3, \tau_{\min} = 0.33$ for shallow networks and $\tau_{\max} = 10, \tau_{\min} = 0.1$ for deep NNs. Considering that the learning rate to achieve good performance generally does not exceed 1 in practice, we could set $\eta_{\max} = k^* + 1$ to ensure that the maximum learning rate during training is as close as possible to 1 but not more than 1, where k^* is the index of the epoch when the learning rate exceeds 1 for the first time. For Example, $k^* = 6$ for training VGG in CIFAR10. For the lower bound, we simply set $\tau_{\min} = \frac{1}{\tau_{\max}}$. The size of mini-batch is 128 for MNIST and CIFAR datasets [13,38] and 256 for ImageNet datasets [30,32,37]. We conduct the experiments in the setting of supervised learning. In fact, the proposed method is also applicable to the related optimization problem arising from those unsupervised deep learning tasks such as [3,4].

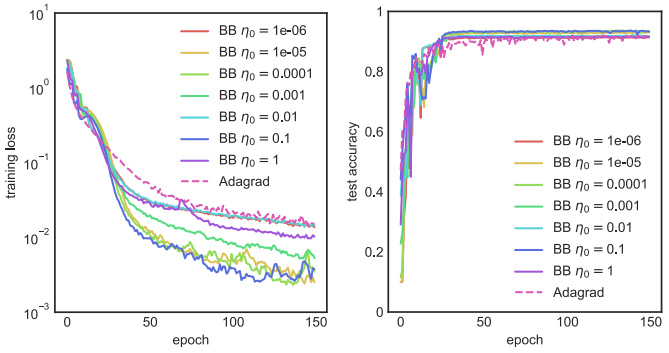

Fig. 2. Performance comparison of BB with different initial learning rates for training ResNet18 on CIFAR10. For comparison, the result from AdaGrad is shown in dashed lines.

4.2. Experimental results

4.2.1. Robustness to initial learning rate

Most existing adaptive learning rate methods are sensitive to initial learning rate. This experiment is to illustrate the robustness of the proposed BB method to initial learning rate. From Fig. 2, It can be seen that the BB method is much less sensitive to the initial learning rate than other adaptive learning rate do in terms of learning. More importantly, the BB method is insensitive to initial learning rate in terms of generalization performance. This property enables the users to avoid unnecessary many trials on initial learning rate for achieving high testing accuracy. In the following experiments, we uniformly use $\eta_0 = 0.1$ throughout all experiments, thanks to its insensitivity to generalization performance.

4.2.2. Experiments on training shallow NNs

Fig. 3 shows the results of training a multilayer perceptron on MNIST. It can be seen that all optimization algorithms achieve close to 99% accuracy on MNIST. Among them, the BB method achieves the lowest training loss and the highest test accuracy. Fig. 4 shows the results of training shallow CNN (LeNet) on CIFAR10. The NAG, a first-order method, has the best performance in both training and testing for LeNet. The performance of the BB method is better than that of the AdaGrad, and is comparable to other adaptive learning rate algorithms. It is noted that the optimization problem for training a LeNet on MNIST is of quite small scale. It has only 0.06M parameters to train, which is 1/400 of that of ResNet50. For such a small-scale problem, it attenuate the advantage of the proposed method over step-size efficiency, since the

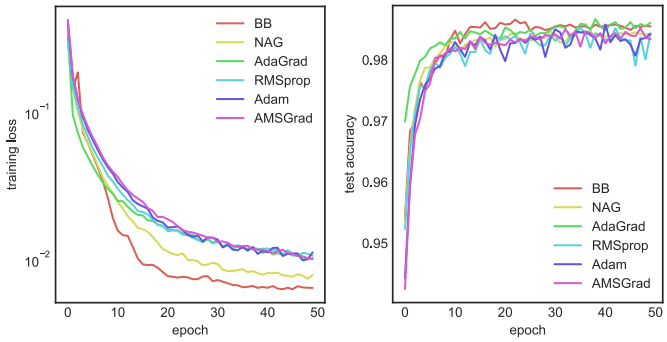

Fig. 3. Performance comparison of different methods for training MLP on MNIST.

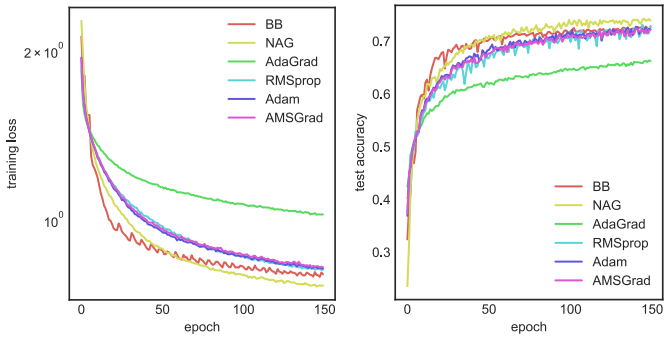

Fig. 4. Performance comparison of different methods for training LeNet on CIFAR10.

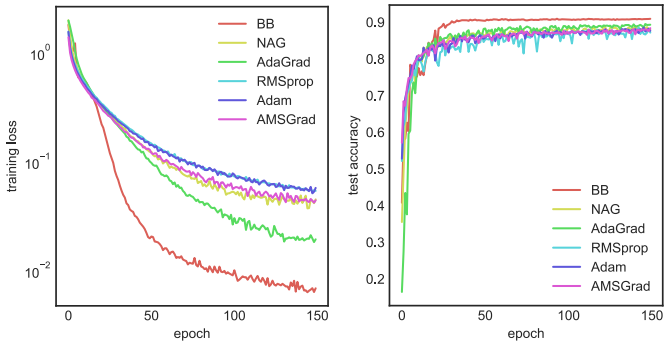

Fig. 5. Performance comparison of different methods for training VGG on CIFAR10.

BB method is for approximating second-order method. In addition, one weakness of second-order methods, such as Newton method, is that they sometimes attract to saddle points more often than first-order methods do [39].

4.2.3. Experiments on training DNNs

Figs. 5 and 6 show the results of training VGG on CIFAR10 and CIFAR100. On both datasets, the training error and testing accuracy obtained by VGG are generally worse than ResNet18. It can be seen that after 25 epochs, the BB still has the lowest training error and the highest test accuracy than other compared methods. For other methods, AdaGrad has the lowest training error on CIFAR10; on CIFAR100, NAG has the lowest training error. In terms of test accuracy, other methods are comparable.

Figs. 7 and 8 show the results of training ResNet18 on CIFAR10 and CIFAR100 respectively. It can be seen that on both datasets, the BB method is significantly better than other methods, as shown in both the final and the middle results. The results of AdaGrad are inferior to BB. Figs. 9 and 10 show the results of training ResNet50 on ImageNet-32-32 and ImageNet respectively. Classification task on ImageNet-32-32 is more difficult than on its original version

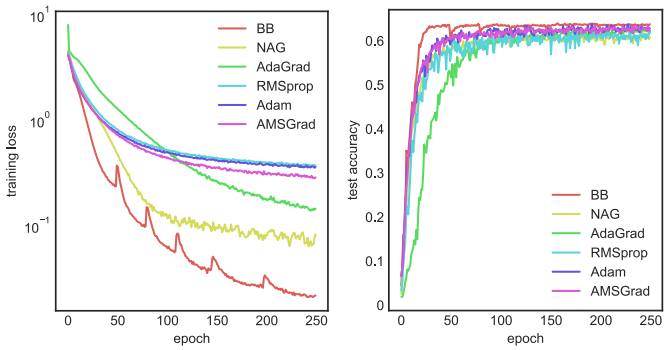

Fig. 6. Performance comparison of different methods for training VGG on CIFAR100.

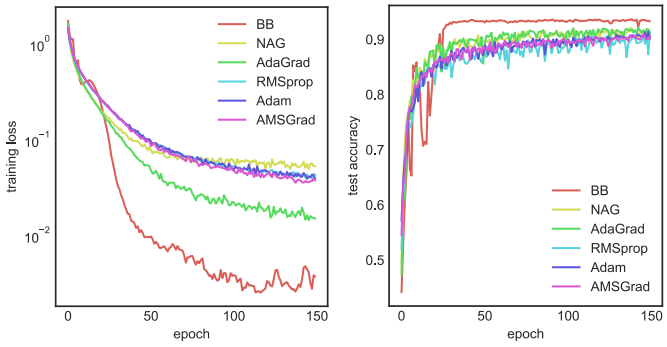

Fig. 7. Performance comparison of different methods for training ResNet18 on CIFAR10.

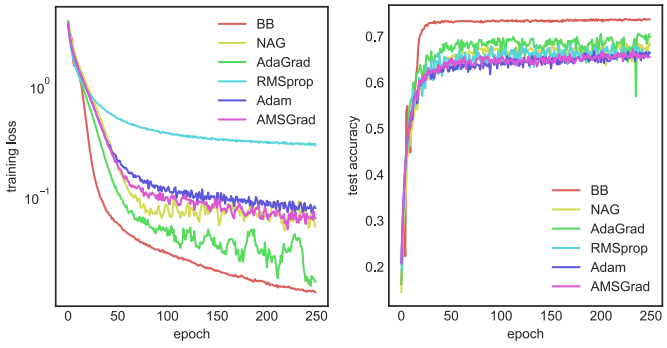

Fig. 8. Performance comparison of different methods for training ResNet18 on CIFAR100.

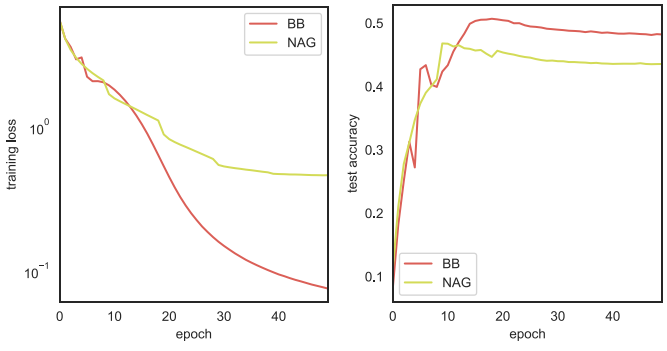

Fig. 9. Performance comparison of different methods for training ResNet50 on ImageNet-32-32.

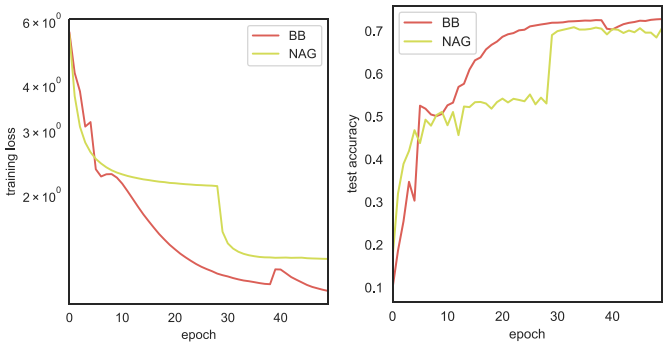

Fig. 10. Performance comparison of different methods for training ResNet50 on ImageNet.

due to the downsampling from 224 to 32. On both datasets, the proposed method could achieve lower training loss and higher test accuracy than NAG, the widely used method for training DNNs on ImageNet.

5. Conclusions

In this paper, we proposed an adaptive learning method for automating the training of NNs. The paper generalizes the well-established BB method in convex optimization to solve the training problem in neural network learning. It is shown that in contrast to many existing adaptive learning rate methods, the BB method is insensitive to initial learning rate, especially in terms of generalization performance, which significantly reduces computational effort to training an NN. The proposed BB method also achieves faster learning speed as well as better generalization performance in various scenarios. In conclusion, the BB method has the potential for automating the process of setting right learning rate when training NNs.

Declaration of Competing Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgments

This research was supported in part by the National Natural Science Foundation of China (grant numbers 61602184, 61872151, 61672241, U1611461), Natural Science Foundation of Guangdong Province (grant number 2016A030308013, 2017A030313376), Science and Technology Program of Guangzhou (grant numbers 201707010147, 201802010055), Science and Technology Planning Project of Guangdong Province (20140904-160), Fundamental Research Funds for Central Universities of China (x2js-D2181690), Beijing Academy of Artificial Intelligence (BAAI), and Singapore MOE AcRF (grant numbers R146000229114, MOE2017-T2-2-156).

References

[1] J. Lu, G. Wang, J. Zhou, Simultaneous feature and dictionary learning for image set based face recognition, IEEE Trans. Image Process. 26 (8) (2017) 4042–4054.
[2] J. Lu, J. Hu, Y. Tan, Discriminative deep metric learning for face and kinship verification, IEEE Trans. Image Process. 26 (9) (2017) 4269–4282.
[3] R. Zhang, P. Isola, A.A. Efros, Split-Brain Autoencoders: unsupervised learning by cross-channel prediction, in: Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR), 2017, pp. 1058–1067.
[4] Y. Duan, J. Lu, Z. Wang, J. Feng, J. Zhou, Learning deep binary descriptor with multi-quantization, IEEE Trans. Pattern Anal. Mach. Intell. 41 (8) (2019) 1924–1938.
[5] L. Bottou, F. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, SIAM Rev. 60 (2) (2018) 223–311.
[6] Y. Bengio, Practical recommendations for gradient-based training of deep architectures, in: G. Montavon, G.B. Orr, K.-R. Müller (Eds.), Neural Networks: Tricks of the Trade, second ed., Springer, 2012, pp. 437–478.
[7] S. Haykin, Neural Networks: a Comprehensive Foundation, second ed., Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.
[8] D. Wilson, T. Martinez, The need for small learning rates on large problems, in: Proc. Int. Joint Conf. Neural Netw. (IJCNN), 2001, pp. 115–119.
[9] Q. Abbas, W.H. Bangyal, J. Ahmad, Analysis of learning rate using BP algorithm for hand written digit recognition application, in: Int. Conf. Inf. Emerging Technol. (ICIET), 2010, pp. 1–5.
[10] S.L. Smith, P.-J. Kindermans, Q.V. Le, Don't decay the learning rate, increase the batch size, in: Proc. Int. Conf. Learn. Represent. (ICLR), 2018.
[11] C. Xing, D. Arpit, C. Tsirigotis, Y. Bengio, A walk with SGD, arXiv:1802.08770 (2018).
[12] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: Proc. Int. Conf. Learn. Represent. (ICLR), 2015.
[13] S.K. Sashank J. Reddi, S. Kale, On the convergence of Adam and Beyond, in: Proc. Int. Conf. Learn. Represent. (ICLR), 2018.
[14] J. Barzilai, J.M. Borwein, Two-point step size gradient methods, IMA J. Numer. Anal. 8 (1) (1988) 141–148.
[15] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (Jul) (2011) 2121–2159.
[16] T. Tieleman, G. Hinton, Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude, COURSERA Neural Netw. Mach. Learn. 4 (2) (2012) 26–31.
[17] S.J. Wright, R.D. Nowak, M.A.T. Figueiredo, Sparse reconstruction by separable approximation, IEEE Trans. Signal Process. 57 (7) (2009) 2479–2493.
[18] C. Bao, G. Barbastathis, H. Ji, Z. Shen, Z. Zhang, Coherence retrieval using trace regularization, SIAM J. Imaging Sci. 11 (1) (2018) 679–706.
[19] V.P. Plagianakos, G.D. Magoulas, M.N. Vrahatis, Deterministic nonmonotone strategies for effective training of multilayer perceptrons, IEEE Trans. Neural Netw. 13 (6) (2002) 1268–1284.
[20] V.P. Plagianakos, D.G. Sotiropoulos, M.N. Vrahatis, An improved backpropagation method with adaptive learning rate, in: N. Mastorakis (Ed.), Recent Advances in Circuits and Systems, World Scientific, 1998, p. 5.
[21] C. Tan, S. Ma, Y.-H. Dai, Y. Qian, Barzilai-Borwein step size for stochastic gradient descent, in: Proc. Annu. Conf. Neural Inf. Process. Syst. (NIPS), 2016, pp. 685–693.
[22] R. Johnson, T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, in: Proc. Annu. Conf. Neural Inf. Process. Syst. (NIPS), 2013, pp. 315–323.
[23] K. Ma, J. Zeng, J. Xiong, Q. Xu, X. Cao, W. Liu, Y. Yao, Stochastic non-convex ordinal embedding with stabilized Barzilai-Borwein step size, in: Proc. AAAI Conf. Artif. Intell. (AAAI), 2018.
[24] A. Defazio, L. Bottou, On the ineffectiveness of variance reduced optimization for deep learning, arXiv:1812.04529 (2018).
[25] X. Wang, S. Ma, W. Liu, Stochastic Quasi-Newton methods for nonconvex stochastic optimization, arXiv:1412.1196 (2014).
[26] D. Bertsekas, J. Tsitsiklis, Gradient convergence in gradient methods with errors, SIAM J. Optim. 10 (3) (2000) 627–642.
[27] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[28] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Technical report, University of Toronto, 2009. 02438.
[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.
[30] P. Chrabaszcz, I. Loshchilov, F. Hutter, A downsampled variant of ImageNet as an alternative to the CIFAR datasets, arXiv:1707.08819 (2017).
[31] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556 (2014).
[32] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 770–778.
[33] Y.A. LeCun, L. Bottou, G.B. Orr, K.-R. Müller, Efficient BackProp, in: G. Montavon, G.B. Orr, K.-R. Müller (Eds.), Neural Networks: Tricks of the Trade, second ed., Springer, 2012, pp. 9–48.
[34] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on ImageNet classification, in: Proc. IEEE Int. Conf. Comput. Vis. (ICCV), 2015, pp. 1026–1034.
[35] I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: Proc. Int. Conf. Mac. Learn. (ICML), 2013, pp. 1139–1147.
[36] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR), 2017, pp. 2261–2269.
[37] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, K. He, Accurate, large minibatch SGD: training imagenet in 1 hour, arXiv:1706.02677 (2017).
[38] I. Loshchilov, F. Hutter, SGDR: stochastic gradient descent with warm restarts, in: Proc. Int. Conf. Learn. Represent. (ICLR), 2017.
[39] Y.N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, Y. Bengio, Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, in: Proc. Annu. Conf. Neural Inf. Process. Syst. (NIPS), 2014, pp. 2933–2941.