SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2021, Vol. 64 132103:1–132103:13
https://doi.org/10.1007/s11432-020-3023-7

On the convergence and improvement of stochastic normalized gradient descent

Shen-Yi ZHAO, Yin-Peng XIE & Wu-Jun LI*

National Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210023*, China*

Received 20 March 2020/Revised 6 May 2020/Accepted 3 June 2020/Published online 8 February 2021

Abstract Non-convex models, like deep neural networks, have been widely used in machine learning applications. Training non-convex models is a difficult task owing to the saddle points of models. Recently, stochastic normalized gradient descent (SNGD), which updates the model parameter by a normalized gradient in each iteration, has attracted much attention. Existing results show that SNGD can achieve better performance on escaping saddle points than classical training methods like stochastic gradient descent (SGD). However, none of the existing studies has provided theoretical proof about the convergence of SNGD for non-convex problems. In this paper, we firstly prove the convergence of SNGD for non-convex problems. Particularly, we prove that SNGD can achieve the same computation complexity as SGD. In addition, based on our convergence proof of SNGD, we find that SNGD needs to adopt a small constant learning rate for convergence guarantee. This makes SNGD do not perform well on training large non-convex models in practice. Hence, we propose a new method, called stagewise SNGD (S-SNGD), to improve the performance of SNGD. Different from SNGD in which a small constant learning rate is necessary for convergence guarantee, S-SNGD can adopt a large initial learning rate and reduce the learning rate by stage. The convergence of S-SNGD can also be theoretically proved for non-convex problems. Empirical results on deep neural networks show that S-SNGD achieves better performance than SNGD in terms of both training loss and test accuracy.

Keywords non-convex problems, stochastic normalized gradient descent, computation complexity

Citation Zhao S-Y, Xie Y-P, Li W-J. On the convergence and improvement of stochastic normalized gradient descent. Sci China Inf Sci, 2021, 64(3): 132103, https://doi.org/10.1007/s11432-020-3023-7

1 Introduction

Many machine learning models can be formulated as the following optimization problem:

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} F(\boldsymbol{w}) := \mathrm{E}[f(\boldsymbol{w};\boldsymbol{a})], \tag{1}$$

where $\boldsymbol{w}$ denotes the model parameter, $\boldsymbol{a}$ denotes a stochastic sample, and $f(\boldsymbol{w};\boldsymbol{a})$ denotes the loss function. When $\boldsymbol{a}$ is uniformly sampled from a finite set $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n\}$, $F(\boldsymbol{w})$ can also be formulated as $1/n\sum_{i=1}^n f(\boldsymbol{w};\boldsymbol{a}_i)$, where n is the number of samples. The formulation in (1) contains a broad family of machine learning models, such as logistic regression and deep learning models.

Stochastic gradient descent (SGD) [1, 2] has been one of the most efficient optimization tools for solving (1). In the t-th iteration, SGD randomly selects some samples $\mathcal{I}_t$ and calculates a mini-batch gradient to update the model parameter, which is denoted by

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha \boldsymbol{g}_t, \tag{2}$$

where $\boldsymbol{g}_t = 1/|\mathcal{I}_t| \sum_{\boldsymbol{a}\in\mathcal{I}_t} \nabla f(\boldsymbol{w}_t;\boldsymbol{a})$ is a mini-batch gradient and $\alpha > 0$ is the learning rate. Compared to traditional batch training methods like gradient descent (GD), SGD does not need to calculate the full gradient $\nabla F(\boldsymbol{w}_t)$ which often costs much computation time. Hence, SGD is more popular and has been

* Corresponding author (email: liwujun@nju.edu.cn)

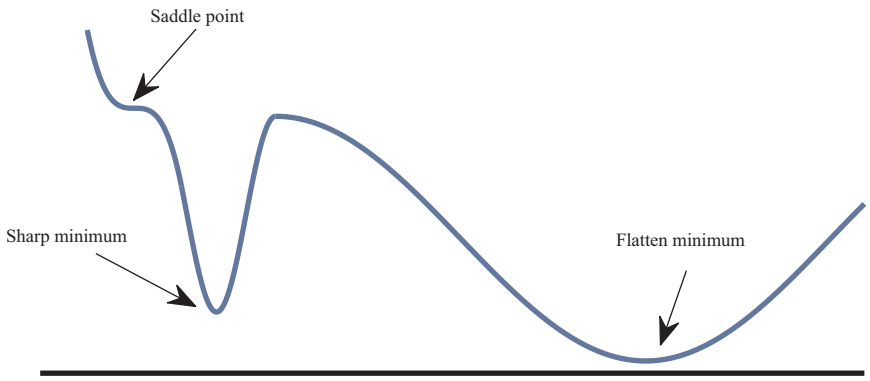

Figure 1 (Color online) Smooth loss curve. The flatten minimum usually leads to a small generalization error [11]. In the region of the saddle point, the gradient is small. In the region of the sharp minimum, the gradient is large. Intuitively, because $\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\| = \alpha$ in SNGD, with a suitable α, normalized gradient can yield a faster escape of saddle points and sharp minimum than unnormalized gradient.

widely used in machine learning applications. Many variants of SGD have been proposed [3–5] and there are also many studies that analyze the convergence of SGD [6–8].

With the rapid growth of data, more and more large non-convex models are adopted in machine learning applications for improving the generalization ability. One typical example is the deep neural network with multiple layers, which has achieved success in many areas like image classification [9]. However, training non-convex models is a difficult task, because there may be many saddle points and sharp local minimums in non-convex models [10, 11], especially in those large models. When $\boldsymbol{w}_t$ falls into the region of some saddle points, the gradient will be small. When $\boldsymbol{w}_t$ falls into the region of some sharp local minimums, the gradient will be relatively large. Both of these two cases can lead to the inefficiency of (2). When $\boldsymbol{w}_t$ falls into the region of some flatten minimums, it usually leads to a small generalization error [11]. Figure 1 shows examples about saddle point, sharp local minimum and flatten minimum. Usually, we need to carefully choose the initialization [12] to escape the saddle points and sharp local minimums.

Recently, stochastic normalized gradient descent (SNGD) [13–16] has attracted much attention for solving non-convex problems. SNGD is the stochastic version of normalized gradient descent (NGD) [17, 18]. In each iteration, SNGD adopts a normalized gradient to update the model parameter. In particular, SNGD can be written as

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha \frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|}, \tag{3}$$

where $\boldsymbol{g}_t = 1/|\mathcal{I}_t| \sum_{\boldsymbol{a}\in\mathcal{I}_t} \nabla f(\boldsymbol{w}_t; \boldsymbol{a})$, $\alpha > 0$ is the learning rate, and $\|\cdot\|$ is the Euclidean norm. It is easy to get that $\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\| = \alpha$, no matter whether the gradient is large or small. Hence, SNGD has better performance than SGD when $\boldsymbol{w}_t$ is around saddle points and sharp local minimums. Theoretical results show that normalized gradient can yield a faster escape of saddle points than SGD [14]. Although the convergence of SNGD in (3) for strictly-locally-quasi-convex problems has been theoretically proved in [13], none of the existing studies has provided theoretical proof about the convergence of SNGD in (3) for non-convex problems. Furthermore, we find that SNGD does not perform well on training large non-convex models in practice owing to the requirement of a small constant learning rate.

In this paper, we study on the convergence and improvement of SNGD. The main contributions of this paper are outlined as follows.

• We theoretically prove the convergence of SNGD for non-convex problems. In particular, we prove that SNGD can achieve the same computation complexity (total number of gradient computation) as SGD for non-convex problems.

• We propose a new method, called S-SNGD, to improve the performance of SNGD. Different from SNGD which necessarily adopts a small constant learning rate for convergence guarantee, S-SNGD can adopt a large initial learning rate and reduces the learning rate by stage. The convergence of S-SNGD can also be theoretically proved for non-convex problems.

• Empirical results on deep neural networks show that S-SNGD achieves better performance than SNGD in terms of both training loss and test accuracy.

2 Preliminary and related work

2.1 Preliminary

In this paper, we use $\|\cdot\|$ to denote the Euclidean norm. For a random variable X, $\mathrm{E}[X]$ denotes the expectation of X. For any two positive sequences $\{x_n\}$ and $\{y_n\}$, $y_n = \mathcal{O}(x_n)$ denotes that there exist two positive constants c and N such that $y_n \leqslant cx_n, \forall n \geqslant N$. For any two positive integers a and b, $\mathrm{mod}(a, b) = 0$ denotes that a is divisible by b. For a function $\phi(\boldsymbol{w})$, $\nabla\phi(\boldsymbol{w})$ denotes the gradient of $\phi(\boldsymbol{w})$ at $\boldsymbol{w}$ and $\nabla^2\phi(\boldsymbol{w})$ denotes the Hessian matrix of $\phi(\boldsymbol{w})$ at $\boldsymbol{w}$.

We also give the following two definitions.

Definition 1. A function $\phi(\boldsymbol{w})$ is called an L-smooth function ($L > 0$) if $\forall \boldsymbol{u}, \boldsymbol{w}$,

$$\phi(\boldsymbol{u}) \leqslant \phi(\boldsymbol{w}) + \nabla\phi(\boldsymbol{w})^{\mathrm{T}}(\boldsymbol{u} - \boldsymbol{w}) + \frac{L}{2}\|\boldsymbol{u} - \boldsymbol{w}\|^2.$$

Definition 2. A function $\phi(\boldsymbol{w})$ is called a (L_0, L_1)-smooth function ($L_0 > 0, L_1 \geqslant 0$) if $\phi(\boldsymbol{w})$ is twice differentiable and $\forall \boldsymbol{w}$,

$$\|\nabla^2\phi(\boldsymbol{w})\| \leqslant L_0 + L_1\|\nabla\phi(\boldsymbol{w})\|,$$

where $\nabla^2\phi(\cdot)$ denotes the Hessian matrix of $\phi(\cdot)$ at $\boldsymbol{w}$.

It is easy to find that if a function $\phi(\boldsymbol{w})$ is $(L_0, 0)$-smooth, it is also L_0-smooth. If a function $\phi(\boldsymbol{w})$ is twice differentiable and L-smooth, it is also $(L, 0)$-smooth.

In this paper, we adopt the norm of gradient to measure the convergence. In particular, we call $\boldsymbol{w}$ an ϵ-stationary point if $\|\nabla F(\boldsymbol{w})\| \leqslant \epsilon$. The computation complexity of an algorithm is defined as the total number of gradient computation. The iteration complexity of an algorithm is defined as total number of model parameter updates.

2.2 Related work

NGD [17, 18] and its variants have attracted much attention for solving non-convex problems. In [14], the authors proposed Saddle-NGD which can be formulated as

$$\boldsymbol{w}_{t+1} = \begin{cases} \boldsymbol{w}_t - \alpha\dfrac{\nabla F(\boldsymbol{w}_t)}{\|\nabla F(\boldsymbol{w}_t)\|}, & \text{if } \mathrm{mod}(t, q) \neq 0, \\ \boldsymbol{w}_t - \alpha\dfrac{\nabla F(\boldsymbol{w}_t)}{\|\nabla F(\boldsymbol{w}_t)\|} + \xi_t, & \text{if } \mathrm{mod}(t, q) = 0, \end{cases} \tag{4}$$

where $q > 0$ is a constant integer, ξ_t is a Gaussian noise. Saddle-NGD is a variant of NGD which combines the noise perturbations. The work in [10] shows that Saddle-NGD achieves better iteration complexity (total number of model updates) than Noisy-GD [19].

The first work that analyzes the convergence of SNGD in (3) appears in [13]. In particular, it proves that for strictly-locally-quasi-convex problems, SNGD needs $\mathcal{O}(1/\epsilon^2)$ iterations to achieve the result of $F(\bar{\boldsymbol{w}}) - F(\boldsymbol{w}^*) \leqslant \epsilon$, where $\bar{\boldsymbol{w}}$ is the output of SNGD and $\boldsymbol{w}^*$ is the optimal solution of (1). Hence, the proved computation complexity to achieve $F(\bar{\boldsymbol{w}}) - F(\boldsymbol{w}^*) \leqslant \epsilon$ is $\mathcal{O}(1/\epsilon^4)$ owing to a batch size of $\mathcal{O}(1/\epsilon^2)$. Recently, in [15, 16], the authors proposed some mixture methods which combine SGD and SNGD. In particular, their formulations can universally be written as

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t\boldsymbol{g}_t, \tag{5}$$

where $\boldsymbol{g}_t$ is some stochastic gradients, $\eta_t = \min\{\alpha/\|\boldsymbol{g}_t\|, \beta\}, \alpha > 0, \beta > 0$. We can observe that when $\|\boldsymbol{g}_t\|$ is larger than α/β, the formulation in (5) is the same as that SNGD. Otherwise, the formulation in (5) is the same as SGD. In [15], the authors proposed a method called SIPDER-SFO. Benefiting from the stochastic path-integrated differential estimator which is defined as

$$\boldsymbol{g}_t = \begin{cases} \dfrac{1}{|\mathcal{I}_t|}\displaystyle\sum_{\boldsymbol{a}\in\mathcal{I}_t}[\nabla f(\boldsymbol{w}_t; \boldsymbol{a}) - \nabla f(\boldsymbol{w}_{t-1}; \boldsymbol{a})] + \boldsymbol{g}_{t-1}, \text{ if } \mathrm{mod}(t, q) \neq 0, \\ \dfrac{1}{|\mathcal{I}_t|}\displaystyle\sum_{\boldsymbol{a}\in\mathcal{I}_t}\nabla f(\boldsymbol{w}_t; \boldsymbol{a}), \text{ if } \mathrm{mod}(t, q) = 0. \end{cases}$$

SPIDER-SFO achieves a computation complexity of $\mathcal{O}(1/\epsilon^3)$ under the criterion of $\|\nabla f(\bar{\boldsymbol{w}})\| \leqslant \epsilon$, which is the best convergence result among existing first-order stochastic methods. However, whether $\boldsymbol{g}_t/\|\boldsymbol{g}_t\|$ can be adopted for small $\|\boldsymbol{g}_t\|$ and the corresponding computation complexity is unknown. Furthermore, owing to the requirement of a small constant learning rate α, we find that these mixture methods are not efficient in practice.

3 Convergence of SNGD

We briefly present SNGD [13] in Algorithm 1. We can observe that SNGD is as simple as SGD. In each iteration, SNGD only needs to calculate a normalized mini-batch gradient to update the model parameter with a constant learning rate α. Different from SGD in which the stochastic gradient is an unbiased estimation of $\nabla F(\boldsymbol{w})$, the stochastic gradient in SNGD is a biased one. That means

$$\mathrm{E}\left[\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|}\middle|\boldsymbol{w}_t\right] \neq \nabla F(\boldsymbol{w}_t).$$

Hence, the update direction is not necessarily the direction of $-\nabla F(\boldsymbol{w}_t)$ in expectation and this makes the convergence analysis difficult.

Algorithm 1 SNGD

1: Initialization: $\boldsymbol{w}_0, \alpha, b, T$;
2: **for** $t = 0, 1, \ldots, T-1$ **do**
3: Randomly choose b samples, denoted by $\mathcal{I}_t$;
4: Calculate a mini-batch gradient $\boldsymbol{g}_t = \frac{1}{b}\sum_{\boldsymbol{a}\in\mathcal{I}_t} f(\boldsymbol{w}_t; \boldsymbol{a})$;
5: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|}$;
6: **end for**
7: Return $\bar{\boldsymbol{w}}$, which is randomly chosen from $\{\boldsymbol{w}_0, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_T\}$.

First, we make the following assumption which is common in stochastic optimization.

Assumption 1. $F(\boldsymbol{w}) \geqslant 0$ and the stochastic gradient of $F(\boldsymbol{w})$ has a σ-bounded variance ($\sigma > 0$), i.e.,

$$\mathrm{E}\|\nabla f(\boldsymbol{w}; \boldsymbol{a}) - \nabla F(\boldsymbol{w})\|^2 \leqslant \sigma^2, \quad \forall \boldsymbol{w}.$$

If $F(\boldsymbol{w})$ is L-smooth, we have the following lemma which captures a general property in normalized gradient descent.

Lemma 1. Assume $F(\boldsymbol{w})$ is L-smooth ($L > 0$). For any $\boldsymbol{w}, \boldsymbol{g} \in \mathbb{R}^d$, we define $\boldsymbol{w}_+$ as follows:

$$\boldsymbol{w}_+ = \boldsymbol{w} - \alpha\frac{\boldsymbol{g}}{\|\boldsymbol{g}\|}, \tag{6}$$

where $\alpha > 0$. Then we have

$$\|\nabla F(\boldsymbol{w})\| \leqslant \frac{F(\boldsymbol{w}) - F(\boldsymbol{w}_+)}{\alpha} + \frac{L\alpha}{2} + 2\|\nabla F(\boldsymbol{w}) - \boldsymbol{g}\|.$$

Proof. Since $F(\boldsymbol{w})$ is L-smooth, we obtain

$$\begin{aligned} F(\boldsymbol{w}_+) &\leqslant F(\boldsymbol{w}) - \alpha\nabla F(\boldsymbol{w})^{\mathrm{T}}\boldsymbol{g}/\|\boldsymbol{g}\| + \frac{L\alpha^2}{2} \\ &= F(\boldsymbol{w}) - \alpha(\nabla F(\boldsymbol{w}) - \boldsymbol{g})^{\mathrm{T}}\boldsymbol{g}/\|\boldsymbol{g}\| - \alpha\|\boldsymbol{g}\| + \frac{L\alpha^2}{2} \\ &\leqslant F(\boldsymbol{w}) + \alpha\|\nabla F(\boldsymbol{w}) - \boldsymbol{g}\| - \alpha\|\boldsymbol{g}\| + \frac{L\alpha^2}{2}. \end{aligned}$$

Combining the fact that $\|\nabla F(\boldsymbol{w})\| \leqslant \|\boldsymbol{g}\| + \|\nabla F(\boldsymbol{w}) - \boldsymbol{g}\|$, we obtain

$$\|\nabla F(\boldsymbol{w})\| \leqslant \frac{F(\boldsymbol{w}) - F(\boldsymbol{w}_+)}{\alpha} + \frac{L\alpha}{2} + 2\|\nabla F(\boldsymbol{w}) - \boldsymbol{g}\|.$$

According to Lemma 1, we can set $\boldsymbol{w}, \boldsymbol{w}_+, \boldsymbol{g}$ in (6) as $\boldsymbol{w}_t, \boldsymbol{w}_{t+1}, \boldsymbol{g}_t$ respectively. Then we can obtain the following convergence result.

Theorem 1. Assume $F(\boldsymbol{w})$ is L-smooth ($L>0$). Let $\{\boldsymbol{w}_t\}$ be the sequence produced by Algorithm 1, $b=\sigma^2/\epsilon^2$. Then we have

$$\frac{1}{T+1}\sum_{t=0}^{T}\mathrm{E}\|\nabla F(\boldsymbol{w}_t)\|\leqslant\frac{F(\boldsymbol{w}_1)}{\alpha(T+1)}+\frac{L\alpha}{2}+2\epsilon. \tag{7}$$

Furthermore, let $\bar{\boldsymbol{w}}$ denote the output of Algorithm 1. By setting $\alpha=2\epsilon/L$ and $T=LF(\boldsymbol{w}_1)/(2\epsilon^2)$, we obtain $\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}})\|\leqslant 4\epsilon$. Hence, the iteration complexity and computation complexity to achieve an ϵ-stationary point of SNGD are $\mathcal{O}(1/\epsilon^2)$ and $\mathcal{O}(1/\epsilon^4)$, respectively.

Proof. According to Lemma 1, we obtain

$$\begin{aligned}\mathrm{E}\|\nabla F(\boldsymbol{w}_t)\|\leqslant&\frac{\mathrm{E}[F(\boldsymbol{w}_t)-F(\boldsymbol{w}_{t+1})]}{\alpha}+\frac{L\alpha}{2}+2\mathrm{E}\|\nabla F(\boldsymbol{w}_t)-\boldsymbol{g}_t\|\\ \leqslant&\frac{\mathrm{E}[F(\boldsymbol{w}_t)-F(\boldsymbol{w}_{t+1})]}{\alpha}+\frac{L\alpha}{2}+2\sqrt{\mathrm{E}\|\nabla F(\boldsymbol{w}_t)-\boldsymbol{g}_t\|^2}\\ \leqslant&\frac{\mathrm{E}[F(\boldsymbol{w}_t)-F(\boldsymbol{w}_{t+1})]}{\alpha}+\frac{L\alpha}{2}+2\sqrt{\frac{\sigma^2}{b}}\\ =&\frac{\mathrm{E}[F(\boldsymbol{w}_t)-F(\boldsymbol{w}_{t+1})]}{\alpha}+\frac{L\alpha}{2}+2\epsilon,\end{aligned} \tag{8}$$

where the second inequality uses the fact that $(\mathrm{E}[x])^2\leqslant\mathrm{E}[x^2]$ and the third inequality uses the fact that $\mathrm{E}\|\nabla F(\boldsymbol{w}_t)-\boldsymbol{g}_t\|^2=\mathrm{E}\|\nabla F(\boldsymbol{w}_t)-\nabla f(\boldsymbol{w}_t;\boldsymbol{a})\|^2/b\leqslant\sigma^2/b$. Summing up (8) from $t=0$ to T, we obtain

$$\frac{1}{T+1}\sum_{t=0}^{\mathrm{T}}\mathrm{E}\|\nabla F(\boldsymbol{w}_t)\|\leqslant\frac{F(\boldsymbol{w}_1)}{\alpha(T+1)}+\frac{L\alpha}{2}+2\epsilon.$$

By setting $\alpha=2\epsilon/L$, $T=LF(\boldsymbol{w}_1)/(2\epsilon^2)$, we obtain

$$\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}})\|\leqslant 4\epsilon.$$

Hence, the iteration complexity is $T=\mathcal{O}(1/\epsilon^2)$ and the computation complexity is $Tb=\mathcal{O}(1/\epsilon^4)$.

The result in Theorem 1 implies that SNGD achieves the same computation complexity as SGD for smooth non-convex problems [6]. Furthermore, when $F(\boldsymbol{w})$ is L-smooth and $F(\boldsymbol{w})\geqslant 0$, we can obtain that $\forall\boldsymbol{w}$,

$$F(\boldsymbol{w})-F(\boldsymbol{w}^*)\geqslant\frac{1}{2L}\|\nabla F(\boldsymbol{w})\|^2,$$

where $\boldsymbol{w}^*\in\arg\min_{\boldsymbol{w}}F(\boldsymbol{w})$. If $F(\boldsymbol{w})$ further satisfies the strictly-locally-quasi-convex property [13], then the computation complexity to achieve an ϵ-stationary point in [13] is actually $\mathcal{O}(1/\epsilon^8)$, which is much higher than $\mathcal{O}(1/\epsilon^4)$ for small ϵ. Hence, our convergence result for SNGD is better than that in [13] for strictly-locally-quasi-convex problems.

If $F(\boldsymbol{w})$ is (L_0,L_1)-smooth, we can get the following result in normalized gradient descent.

Lemma 2. Assume $F(\boldsymbol{w})$ is (L_0,L_1)-smooth ($L_0>0,L_1\geqslant 0$). For any $\boldsymbol{w},\boldsymbol{g}\in\mathbb{R}^d$, we define $\boldsymbol{w}_+$ as follows:

$$\boldsymbol{w}_+=\boldsymbol{w}-\alpha\frac{\boldsymbol{g}}{\|\boldsymbol{g}\|}, \tag{9}$$

where $\alpha>0$. Then we have

$$\left(1-\frac{\alpha L_1}{2}\mathrm{e}^{\alpha L_1}\right)\|\nabla F(\boldsymbol{w})\|\leqslant\frac{F(\boldsymbol{w})-F(\boldsymbol{w}_+)}{\alpha}+\frac{(1+\alpha L_1\mathrm{e}^{\alpha L_1})L_0\alpha}{2}+2\|\nabla F(\boldsymbol{w})-\boldsymbol{g}\|.$$

Proof. Let $r(t)=t(\boldsymbol{w}_+-\boldsymbol{w})+\boldsymbol{w},t\in[0,1],p(t)=\|\nabla F(r(t))\|$. Since $\|\boldsymbol{w}_+-\boldsymbol{w}\|=\alpha$, we have

$$p(t)=\|\nabla F(r(t))\|=\left\|\int_0^t\nabla^2F(r(c))r'(c)\mathrm{d}c+\nabla F(r(0))\right\|$$

$$\begin{aligned}
&\leqslant \int_0^t \|\nabla^2 F(r(c))\|\|\boldsymbol{w}_+ - \boldsymbol{w}\|\mathrm{d}c + \|\nabla F(r(0))\| \\
&\leqslant \alpha \int_0^t L_0 + L_1\|\nabla F(r(c))\|\mathrm{d}c + \|\nabla F(\boldsymbol{w})\| \\
&\leqslant \alpha L_0 + \|\nabla F(\boldsymbol{w})\| + \alpha L_1 \int_0^t \|\nabla F(r(c))\|\mathrm{d}c,
\end{aligned}$$

i.e.,

$$p(t) \leqslant \alpha L_0 + \|\nabla F(\boldsymbol{w})\| + \alpha L_1 \int_0^t p(c)\mathrm{d}c.$$

According to Gronwall's inequality, we obtain $\forall t \in [0,1]$,

$$p(t) \leqslant (\alpha L_0 + \|\nabla F(\boldsymbol{w})\|)\mathrm{e}^{\alpha L_1}.$$

Since $F(\boldsymbol{w})$ is twice differentiable, we obtain that $\exists \xi \in [0,t]$ such that

$$\begin{aligned}
F(r(t)) &= F(r(0)) + t\nabla F(r(0))^{\mathrm{T}}(\boldsymbol{w}_+ - \boldsymbol{w}) + \frac{t^2}{2}(\boldsymbol{w}_+ - \boldsymbol{w}_0)^{\mathrm{T}}\nabla^2 F(r(\xi))(\boldsymbol{w}_+ - \boldsymbol{w}_0) \\
&\leqslant F(r(0)) + t\nabla F(r(0))^{\mathrm{T}}(\boldsymbol{w}_+ - \boldsymbol{w}) + \frac{\|\boldsymbol{w}_+ - \boldsymbol{w}\|^2 t^2}{2}(L_0 + L_1 p(\xi)) \\
&\leqslant F(r(0)) + t\nabla F(r(0))^{\mathrm{T}}(\boldsymbol{w}_+ - \boldsymbol{w}) + \frac{\|\boldsymbol{w}_+ - \boldsymbol{w}\|^2 t^2}{2}(L_0 + L_1((\alpha L_0 + \|\nabla F(\boldsymbol{w})\|)\mathrm{e}^{\alpha L_1})).
\end{aligned}$$

Let $t = 1$, we obtain

$$\begin{aligned}
F(\boldsymbol{w}+) &\leqslant F(\boldsymbol{w}) - \alpha\nabla F(\boldsymbol{w})^{\mathrm{T}}\frac{\boldsymbol{g}}{\|\boldsymbol{g}_t\|} + \frac{\alpha^2}{2}(L_0 + L_1((\alpha L_0 + \|\nabla F(\boldsymbol{w})\|)\mathrm{e}^{\alpha L_1})) \\
&\leqslant F(\boldsymbol{w}) - \alpha(\nabla F(\boldsymbol{w}) - \boldsymbol{g})^{\mathrm{T}}\boldsymbol{g}/\|\boldsymbol{g}\| - \alpha\|\boldsymbol{g}\| + \frac{\alpha^2}{2}(L_0 + L_1((\alpha L_0 + \|\nabla F(\boldsymbol{w})\|)\mathrm{e}^{\alpha L_1})) \\
&\leqslant F(\boldsymbol{w}) - \alpha\|\nabla F(\boldsymbol{w}) - \boldsymbol{g}\| - \alpha\|\boldsymbol{g}\| + \frac{\alpha^2}{2}(L_0 + \alpha L_0 L_1 \mathrm{e}^{\alpha L_1} + L_1\mathrm{e}^{\alpha L_1}\|\nabla F(\boldsymbol{w})\|).
\end{aligned}$$

Combining the fact that $\|\nabla F(\boldsymbol{w})\| \leqslant \|\boldsymbol{g}\| + \|\nabla F(\boldsymbol{w}) - \boldsymbol{g}\|$, we obtain

$$\left(1 - \frac{\alpha L_1}{2}\mathrm{e}^{\alpha L_1}\right)\|\nabla F(\boldsymbol{w})\| \leqslant \frac{F(\boldsymbol{w}) - F(\boldsymbol{w}_+)}{\alpha} + \frac{(1 + \alpha L_1 \mathrm{e}^{\alpha L_1})L_0\alpha}{2} + 2\|\nabla F(\boldsymbol{w}) - \boldsymbol{g}\|.$$

According to Lemma 2, we can set $\boldsymbol{w}, \boldsymbol{w}_+, \boldsymbol{g}$ in (9) as $\boldsymbol{w}_t, \boldsymbol{w}_{t+1}, \boldsymbol{g}_t$ respectively. Then we can obtain the following convergence result.

Theorem 2. Assume $F(\boldsymbol{w})$ is (L_0, L_1)-smooth ($L_0 > 0, L_1 \geqslant 0$). Let $\{\boldsymbol{w}_t\}$ be the sequence produced by Algorithm 1, $\alpha L_1 \leqslant 1/2$, $b = \sigma^2/\epsilon^2$. Then we have

$$\frac{1}{T+1}\sum_{t=0}^{\mathrm{T}} \mathrm{E}\|\nabla F(\boldsymbol{w}_t)\| \leqslant \frac{2F(\boldsymbol{w}_1)}{\alpha(T+1)} + 2L_0\alpha + 4\epsilon. \tag{10}$$

Furthermore, let $\bar{\boldsymbol{w}}$ denote the output of Algorithm 1. By setting $\alpha = \epsilon/L_0$, $T = L_0 F(\boldsymbol{w}_1)/\epsilon^2$, we obtain $\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}})\| \leqslant 8\epsilon$. Hence, the iteration complexity and computation complexity to achieve an ϵ-stationary point of SNGD are $\mathcal{O}(1/\epsilon^2)$ and $\mathcal{O}(1/\epsilon^4)$, respectively.

Proof. According to Lemma 2 and $\alpha L_1 \leqslant 1/2$, we obtain

$$\|\nabla F(\boldsymbol{w}_t)\| \leqslant \frac{2(F(\boldsymbol{w}_t) - F(\boldsymbol{w}_{t+1}))}{\alpha} + 2L_0\alpha + 4\|\nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t\|.$$

Similar to the proof of Theorem 1, we obtain

$$\mathrm{E}\|\nabla F(\boldsymbol{w}_t)\| \leqslant \frac{2\mathrm{E}[F(\boldsymbol{w}_t) - F(\boldsymbol{w}_{t+1})]}{\alpha} + 2L_0\alpha + 4\epsilon.$$

Summing up the above inequality from $t = 0$ to T, we obtain

$$\frac{1}{T+1}\sum_{t=0}^{\mathrm{T}}\mathrm{E}\|\nabla F(\boldsymbol{w}_t)\| \leqslant \frac{2F(\boldsymbol{w}_1)}{\alpha(T+1)} + 2L_0\alpha + 4\epsilon.$$

By setting $\alpha = \epsilon/L_0$, $T = L_0F(\boldsymbol{w}_1)/\epsilon^2$, we obtain

$$\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}})\| \leqslant 8\epsilon.$$

Hence, the iteration complexity is $T = \mathcal{O}(1/\epsilon^2)$ and the computation complexity is $Tb = \mathcal{O}(1/\epsilon^4)$.

Compared to the L-smooth assumption in Theorem 1, the (L_0, L_1)-smooth assumption is more general. In particular, if $F(\boldsymbol{w})$ is twice differentiable and L-smooth, then it must be $(L, 0)$-smooth. Furthermore, the (L_0, L_1)-smooth assumption implies that when $L_1 \neq 0$, the gradient of $F(\boldsymbol{w})$ may change drastically, which is a common phenomenon in machine learning models [16]. When we apply SGD on these models in practice, we usually need to clip gradients to avoid gradient explosion. Existing theoretical results about the convergence of SGD need the gradient norm to be bounded if $F(\boldsymbol{w})$ does not satisfy the L-smooth property in Definition 1 [20]. On the contrary, the result in Theorem 4 shows that SNGD can deal with the (L_0, L_1)-smooth problems without bounded gradient assumption and the corresponding computation complexity to achieve an ϵ-stationary point is $\mathcal{O}(1/\epsilon^4)$.

4 Stagewise stochastic normalized gradient descent

In both Theorems 1 and 2, we observe that SNGD needs a small constant learning rate $\alpha = \mathcal{O}(\epsilon)$ to achieve an ϵ-stationary point. The small constant learning rate may slow down the convergence rate and make SNGD inefficient in practice. Here, we propose a new method, called S-SNGD, to improve the performance of SNGD. The details of S-SNGD are presented in Algorithm 2.

Algorithm 2 S-SNGD

Initialization: $\tilde{\boldsymbol{w}}_0, b, S, T, \{\alpha_t\}, \{p_t\}$. Here, $\{p_t\}$ is a positive increasing sequence.
for $t = 0, 1, \ldots, T-1$ **do**
  $\boldsymbol{w}_{t,0} = \tilde{\boldsymbol{w}}_t$;
  $M_t = S/\alpha_t$;
  for $m = 0, 1, \ldots, M_t - 1$ **do**
    Randomly choose b samples, denoted by $\mathcal{I}_t$;
    $\boldsymbol{g}_{t,m} = \frac{1}{b}\sum_{\boldsymbol{a}\in\mathcal{I}_{t,m}} \nabla f(\boldsymbol{w}_{t,m}; \boldsymbol{a})$;
    $\boldsymbol{w}_{t,m+1} = \boldsymbol{w}_{t,m} - \alpha_t \frac{\boldsymbol{g}_{t,m}}{\|\boldsymbol{g}_{t,m}\|}$;
  end for
  Choose $\bar{\boldsymbol{w}}_{t+1}$ randomly from $\{\boldsymbol{w}_{t,0}, \ldots, \boldsymbol{w}_{t,M_t-1}\}$;
  $\tilde{\boldsymbol{w}}_{t+1} = \boldsymbol{w}_{t,M_t}$;
end for
Return $\bar{\boldsymbol{w}}$, which equals to $\bar{\boldsymbol{w}}_i$ with probability $p_i/\sum_{t=1}^{T} p_t, i = 1, \ldots, T$.

S-SNGD consists of T stages. In the t-th stage, S-SNGD runs SNGD with M_t iterations and the learning rate α_t. After each stage, S-SNGD randomly chooses a vector from $\{\boldsymbol{w}_{t,0}, \boldsymbol{w}_{t,1}, \ldots, \boldsymbol{w}_{t,M_t-1}\}$ to be $\bar{\boldsymbol{w}}_{t+1}$ and sets $\tilde{\boldsymbol{w}}_{t+1} = \boldsymbol{w}_{t,M_t}$. Different from SNGD in which a small constant learning rate is necessary for convergence guarantee, S-SNGD allows using a large initial learning rate and then reduces the learning rate by stage. Intuitively, according to (7), a larger initial learning rate can reduce the value of $F(\boldsymbol{w}_1)/\alpha T$ more fastly. After several model parameter updates, S-SNGD reduces the learning rate so that it can reduce the value of $L\alpha/2$. In particular, for L-smooth loss function, we have the following convergence result for S-SNGD.

Theorem 3. Assume $0 < F(\boldsymbol{w}) \leqslant \delta$ is L-smooth, and $0 < p_t \leqslant p_{t+1}, \forall t \geqslant 1$. $\bar{\boldsymbol{w}}$ denotes the output of Algorithm 2. Let $S > 0$, $M_t = S/\alpha_t$, $b = \sigma^2/\epsilon^2$. Then we have

$$\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}})\| \leqslant \frac{\delta p_T}{S\sum_{t=1}^{\mathrm{T}} p_t} + \frac{L\sum_{t=1}^{\mathrm{T}} \alpha_{t-1}p_t}{2\sum_{t=1}^{\mathrm{T}} p_t} + 2\epsilon. \tag{11}$$

Proof. According to Lemma 1, we obtain

$$\|\nabla F(\boldsymbol{w}_{t,m})\| \leqslant \frac{F(\boldsymbol{w}_{t,m}) - F(\boldsymbol{w}_{t,m+1})}{\alpha_t} + \frac{L\alpha_t}{2} + 2\|\nabla F(\boldsymbol{w}_{t,m}) - \boldsymbol{g}_{t,m}\|.$$

Summing up the above inequality from $m=0$ to M_t-1, we obtain

$$\frac{1}{M_t}\sum_{m=0}^{M_t-1}\mathrm{E}\|\nabla F(\boldsymbol{w}_{t,m})\| \leqslant \frac{\mathrm{E}[F(\boldsymbol{w}_{t,0})-F(\boldsymbol{w}_{t,M_t})]}{M_t\alpha_t}+\frac{L\alpha_t}{2}+2\epsilon.$$

Since $S=M_t\alpha_t$, we obtain

$$\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}}_{t+1})\| \leqslant \frac{\mathrm{E}[F(\tilde{\boldsymbol{w}}_t)-F(\tilde{\boldsymbol{w}}_{t+1})]}{S}+\frac{L\alpha_t}{2}+2\epsilon. \tag{12}$$

Then we obtain

$$\begin{aligned}\sum_{t=1}^{\mathrm{T}} p_t\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}}_t)\| &\leqslant \frac{1}{S}\sum_{t=0}^{T-1} p_{t+1}\mathrm{E}[F(\tilde{\boldsymbol{w}}_t)-F(\tilde{\boldsymbol{w}}_{t+1})]+\sum_{t=0}^{T-1}\left(\frac{L\alpha_t}{2}+2\sqrt{\frac{\sigma^2}{b}}\right)p_{t+1}\\ &\leqslant \frac{1}{S}\left[p_1F(\tilde{\boldsymbol{w}}_0)+\sum_{t=1}^{T-1}(p_{t+1}-p_t)F(\tilde{\boldsymbol{w}}_t)\right]+\sum_{t=0}^{T-1}\left(\frac{L\alpha_t}{2}+2\sqrt{\frac{\sigma^2}{b}}\right)p_{t+1}\\ &\leqslant \frac{\delta p_T}{S}+\sum_{t=0}^{T-1}\left(\frac{L\alpha_t}{2}+2\sqrt{\frac{\sigma^2}{b}}\right)p_{t+1}.\end{aligned}$$

Since $\bar{\boldsymbol{w}}=\bar{\boldsymbol{w}}_i$ with probability $p_i/\sum_{t=1}^{\mathrm{T}}p_t$, we obtain

$$\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}})\| \leqslant \frac{\delta p_T}{S\sum_{t=1}^{\mathrm{T}}p_t}+\frac{L\sum_{t=1}^{\mathrm{T}}\alpha_{t-1}p_t}{2\sum_{t=1}^{\mathrm{T}}p_t}+2\epsilon.$$

If $F(\boldsymbol{w})$ is (L_0,L_1)-smooth, we have the following convergence result for S-SNGD.

Theorem 4. Assume $0<F(\boldsymbol{w})\leqslant\delta$ is (L_0,L_1)-smooth ($L_0>0, L_1\geqslant 0$), and $0<p_t\leqslant p_{t+1}, \forall t\geqslant 1$. $\bar{\boldsymbol{w}}$ denotes the output of Algorithm 2. Let $\alpha_tL_1\leqslant 1/2$, $S>0$, $M_t=S/\alpha_t$, $b=\sigma^2/\epsilon^2$. Then we have

$$\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}})\| \leqslant \frac{2\delta p_T}{S\sum_{t=1}^{\mathrm{T}}p_t}+\frac{2L_0\sum_{t=1}^{\mathrm{T}}\alpha_{t-1}p_t}{\sum_{t=1}^{\mathrm{T}}p_t}+4\epsilon. \tag{13}$$

Proof. The proof is similar to that of Theorem 3. According to Lemma 2, we obtain

$$\|\nabla F(\boldsymbol{w}_{t,m})\| \leqslant \frac{2(F(\boldsymbol{w}_{t,m})-F(\boldsymbol{w}_{t,m+1}))}{\alpha_t}+2L_0\alpha_t+4\|\nabla F(\boldsymbol{w}_{t,m})-\boldsymbol{g}_{t,m}\|.$$

Summing up the above inequality from $m=0$ to M_t-1, we obtain

$$\frac{1}{M_t}\sum_{m=0}^{M_t-1}\mathrm{E}\|\nabla F(\boldsymbol{w}_{t,m})\| \leqslant \frac{2\mathrm{E}[F(\boldsymbol{w}_{t,0})-F(\boldsymbol{w}_{t,M_t})]}{M_t\alpha_t}+2L_0\alpha_t+4\sqrt{\frac{\sigma^2}{b}}.$$

Since $S=M_t\alpha_t$, we obtain

$$\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}}_{t+1})\| \leqslant \frac{2\mathrm{E}[F(\tilde{\boldsymbol{w}}_t)-F(\tilde{\boldsymbol{w}}_{t+1})]}{S}+2L_0\alpha_t+4\sqrt{\frac{\sigma^2}{b}}. \tag{14}$$

Then we obtain

$$\begin{aligned}\sum_{t=1}^{\mathrm{T}} p_t\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}}_t)\| &\leqslant \frac{2}{S}\sum_{t=0}^{T-1} p_{t+1}\mathrm{E}[F(\tilde{\boldsymbol{w}}_t)-F(\tilde{\boldsymbol{w}}_{t+1})]+\sum_{t=0}^{T-1}\left(2L_0\alpha_t+4\sqrt{\frac{\sigma^2}{b}}\right)p_{t+1}\\ &\leqslant \frac{2}{S}\left[p_1F(\tilde{\boldsymbol{w}}_0)+\sum_{t=1}^{T-1}(p_{t+1}-p_t)F(\tilde{\boldsymbol{w}}_t)\right]+\sum_{t=0}^{T-1}\left(2L_0\alpha_t+4\sqrt{\frac{\sigma^2}{b}}\right)p_{t+1}\\ &\leqslant \frac{2\delta p_T}{S}+\sum_{t=0}^{T-1}\left(2L_0\alpha_t+4\sqrt{\frac{\sigma^2}{b}}\right)p_{t+1}.\end{aligned}$$

Since $\bar{\boldsymbol{w}} = \bar{\boldsymbol{w}}_i$ with probability $p_i / \sum_{t=1}^{\mathrm{T}} p_t$, we obtain

$$\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}})\| \leqslant \frac{2\delta p_T}{S\sum_{t=1}^{\mathrm{T}} p_t} + \frac{2L_0 \sum_{t=1}^{\mathrm{T}} \alpha_{t-1} p_t}{\sum_{t=1}^{\mathrm{T}} p_t} + 4\epsilon.$$

According to the second term in the right side of (11) or (13), i.e., $\mathcal{O}(\sum_{t=1}^{\mathrm{T}} p_t \alpha_t / \sum_{t=1}^{\mathrm{T}} p_t)$, we observe that reducing the learning rate is necessary for the convergence guarantee. According to the first term in the right side of (11) or (13), i.e., $\mathcal{O}(\delta p_T / (S \sum_{t=1}^{\mathrm{T}} p_t))$, we can set a relatively large S, which means S-SNGD can update the model parameter with a large learning rate for many iterations. Below, we give some specific $\{\alpha_t\}$ and $\{p_t\}$ to show the convergence of S-SNGD.

Corollary 1. With the assumptions and settings in Theorem 3 or Theorem 4, and for any $\epsilon > 0$, let $T = \mathcal{O}(1/\epsilon)$, $\alpha_t = \rho/(t+1)$, $p_t = 1/\alpha_{t-1}$, where $\rho > 0$ is a constant. Then we have $\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}})\| \leqslant \mathcal{O}(\epsilon)$.

Proof. Since $\alpha_t = \rho/(t+1)$, $p_t = 1/\alpha_{t-1}$, where $\rho > 0$ is a constant, we obtain that $\sum_{t=1}^{T} p_t = T(T+1)/(2\rho)$ and $\sum_{t=1}^{T} \alpha_{t-1} p_t = \rho T$. By plugging the two summation terms into (11) or (13), we obtain that

$$\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}})\| \leqslant \mathcal{O}\left(\frac{1}{T} + \epsilon\right) = \mathcal{O}(\epsilon),$$

where the inequality uses the fact that $T = \mathcal{O}(1/\epsilon)$.

Corollary 2. With the assumptions and settings in Theorem 3 or Theorem 4, and for any $\epsilon > 0$, let $T = \mathcal{O}(\log(1/\epsilon))$, $S = \mathcal{O}(1/\epsilon)$, $\alpha_t = \rho^t$, $p_t = 1/(2\rho)^t$, where $\rho \in (0, 0.5)$ is a constant. Then we have $\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}})\| \leqslant \mathcal{O}(\epsilon)$.

Proof. Since $\alpha_t = \rho^{t+1}$, $p_t = 1/(2\rho)^t$, where $\rho \in (0, 0.5)$ is a constant, we obtain that $\sum_{t=1}^{T} p_t = (1/(2\rho)^{\mathrm{T}} - 1)/(2\rho - 1)$ and $\sum_{t=1}^{T} \alpha_{t-1} p_t \leqslant 1$. Similar to the proof of Corollary 1, we plug the two summation terms into (11) or (13), and obtain that $\mathrm{E}\|\nabla F(\bar{\boldsymbol{w}})\| \leqslant \mathcal{O}(\epsilon)$.

Remark 1. In Theorems 3 and 4, we assume that the objective value $F(\boldsymbol{w})$ is bounded. In practice, we find that $F(\boldsymbol{w}_{t,m})$ does be bounded. Furthermore, if we do not make this assumption, the convergence can also be guaranteed. In particular, we can directly sum up (12) or (14) (the two equations do not need the bounded value assumption), and set $\alpha_t = \mathcal{O}(1/t)$. Then we obtain

$$\frac{1}{T}\sum_{t=1}^{T} \mathrm{E}\|\nabla F(\bar{\boldsymbol{w}}_t)\| \leqslant \frac{2F(\tilde{\boldsymbol{w}}_0)}{TS} + \mathcal{O}\left(\frac{\log T}{T}\right) + \mathcal{O}(\epsilon).$$

Although the convergence rate is slightly worse than that in Corollary 1 owing to the logarithm term $\log T$, we can observe that the convergence rate does not rely on the upper bound constant δ of the objective function defined in Theorems 3 and 4.

5 Comparison with SPIDER-SFO

We briefly present SPIDER-SFO [15] in Algorithm 3. SPIDER-SFO also consists of several stages and it adopts the constant learning rates α and β throughout the whole algorithm. SPIDER-SFO updates the model parameter by the normalized gradient when $\|\boldsymbol{\mu}_{t,m}\|$ is larger than α/β. Otherwise, it updates the model parameter by the unnormalized gradient. Hence, SPIDER-SFO keeps the relation $\|\boldsymbol{w}_{t,m} - \boldsymbol{w}_{t,m-1}\| \leqslant \alpha$. In [15], the authors theoretically proved that for L-smooth loss functions, SPIDER-SFO can achieve a computation complexity $\mathcal{O}(1/\epsilon^3)$ by setting $\alpha = \mathcal{O}(\epsilon/L)$, $\beta = \mathcal{O}(1/L)$, $B = \mathcal{O}(1/\epsilon^2)$, $b = \mathcal{O}(1/\epsilon)$, $M = \mathcal{O}(1/\epsilon)$, $T = \mathcal{O}(1/\epsilon)$. Owing to the technique of the stochastic path-integrated differential estimator, SPIDER-SFO achieves faster convergence rate than other variance reduction methods like SVRG [21], SCSG [22] and Natasha2 [23]. The work in [24] also points out that those variance reduction methods are ineffective for some non-convex optimization problems like deep learning. In fact, SPIDER-SFO theoretically achieves the best computation complexity among all existing first-order stochastic methods [15].

Comparing the convergence conditions of SPIDER-SFO and SNGD, we can find that both of them require a small constant learning rate $\alpha = \mathcal{O}(\epsilon/L)$. However, when training small non-convex models (e.g., neural networks with one hidden layer), we empirically find that SNGD can adopt a relatively large

Algorithm 3 SPIDER-SFO [15]

Initialization: $\tilde{\boldsymbol{w}}_0, \alpha, \beta, B, b$.
for $t = 0, 1, \ldots, T-1$ **do**
  Randomly choose B samples, denoted by $\mathcal{I}_t$;
  Calculate a mini-batch gradient $\tilde{\boldsymbol{\mu}}_t = \frac{1}{B}\sum_{\boldsymbol{a}\in\mathcal{I}_t}\nabla f(\tilde{\boldsymbol{w}}_t;\boldsymbol{a})$;
  $\boldsymbol{w}_{t,0} = \boldsymbol{w}_{t,1} = \tilde{\boldsymbol{w}}_t$;
  $\boldsymbol{\mu}_{t,0} = \tilde{\boldsymbol{\mu}}_t$;
  for $m = 1, 2, \ldots, M$ **do**
    Randomly choose b samples, denoted by $\mathcal{I}_{t,m}$;
    $\boldsymbol{\mu}_{t,m} = \frac{1}{b}\sum_{\boldsymbol{a}\in\mathcal{I}_{t,m}}(\nabla f(\boldsymbol{w}_{t,m};\boldsymbol{a}) - \nabla f(\boldsymbol{w}_{t,m-1};\boldsymbol{a})) + \boldsymbol{\mu}_{t,m-1}$;
    if $\|\boldsymbol{\mu}_{t,m}\| \leqslant \alpha/\beta$ **then**
      $\boldsymbol{w}_{t,m+1} = \boldsymbol{w}_{t,m} - \beta\boldsymbol{\mu}_{t,m}$;
    else
      $\boldsymbol{w}_{t,m+1} = \boldsymbol{w}_{t,m} - \alpha\frac{\boldsymbol{\mu}_{t,m}}{\|\boldsymbol{\mu}_{t,m}\|}$;
    end if
  end for
  $\tilde{\boldsymbol{w}}_{t+1} = \boldsymbol{w}_{M+1}$;
end for
Return $\bar{\boldsymbol{w}}$, which is randomly chosen from $\{\boldsymbol{w}_{t,m}\}_{t=0,1,\ldots,T;m=1,2,\ldots,M}$.

constant learning rate with good convergence property, and SPIDER-SFO with a large constant learning rate does not perform well. This interesting phenomenon can be explained as follows. According to the proofs in [15], we can obtain that the $\boldsymbol{\mu}_{t,m}$ in Algorithm 3 is an unbiased estimation of $\nabla F(\boldsymbol{w}_{t,m})$, i.e., $\mathrm{E}[\boldsymbol{\mu}_{t,m}|\boldsymbol{w}_{t,m}] = \nabla F(\boldsymbol{w}_{t,m})$. Furthermore, if $f(\boldsymbol{w};\boldsymbol{a})$ is L-smooth, then the variance of $\boldsymbol{\mu}_{t,m}$ is bounded as follows: $\forall t, 1 \leqslant m \leqslant M$,

$$\mathrm{E}\|\boldsymbol{\mu}_{t,m} - \nabla F(\boldsymbol{w}_{t,m})\|^2 \leqslant \frac{M\alpha^2L^2}{b} + \frac{\sigma^2}{B} = \mathcal{O}\left(\alpha^2 + \frac{1}{B}\right), \tag{15}$$

where $M = \Theta(b)$ for $\mathcal{O}(1/\epsilon^3)$ computation complexity guarantee. According to (15), we can observe that the variance of the stochastic gradient $\boldsymbol{\mu}_{t,m}$ is related to four hyper-parameters B, b, α, M, and a large learning rate α may yield a large variance in the learning process. Hence, a small learning rate is necessary for SPIDER-SFO in real applications. On the contrary, the variance of $\boldsymbol{g}_t$ in Algorithm 1 is only related to the hyper-parameter b and the learning rate will not affect the variance of $\boldsymbol{g}_t$. Hence, SNGD can empirically adopt a relatively large constant learning rate. Owing to the same reason that the learning rate does not affect the variance of $\boldsymbol{g}_{t,m}$ in Algorithm 2, S-SNGD can successfully adopt stagewise learning rate reducing strategies with a large initial learning rate.

6 Experiments

We conduct experiments on PyTorch platform (version 1.4.0) with an NVIDIA V100 GPU (cuda version 10.0). As stated in Section 5, the work in [15] has verified that SPIDER-SFO is better than existing stochastic optimization methods including some variance reduction methods like Natash2. Hence, we mainly adopt SPIDER-SFO as the baseline for comparison. In addition, we also compare our method with SGD to show that SNGD can more easily escape the saddle points or sharp minimums of the non-convex problems.

6.1 Small non-convex model

First, we use the handwritten digits dataset MNIST[1)] and a fully connected 3-layer network to compare SNGD and SPIDER-SFO. In MNIST, the training set includes 60000 samples and the testing set includes 10000 samples. Since each sample in MNIST is a 28×28 gray picture, we scale the pixels in each picture to $[0, 1]$ as that in the website[2)]. The dimensions of input and output layers of this network are 784 and 10, respectively. The hidden layer of this network contains 100 units. We use the exponential linear unit (ELU) as activation function. In SPIDER-SFO, we set $\beta = 1$, $B = 1024$, $M = B/(2b)$. For both SNGD and SPIDER-SFO, we set $\alpha \in \{0.1, 0.001\}$, $b \in \{32, 128\}$. The results are presented in Figure 2.

Comparing the blue dotted line to the yellow solid line in Figures 2(a) and (c), we can observe that when a small learning rate is adopted for these two methods, SPIDER-SFO converges faster than SNGD in

1) http://yann.lecun.com/exdb/mnist/.
2) https://github.com/pytorch/examples/tree/master/mnist.

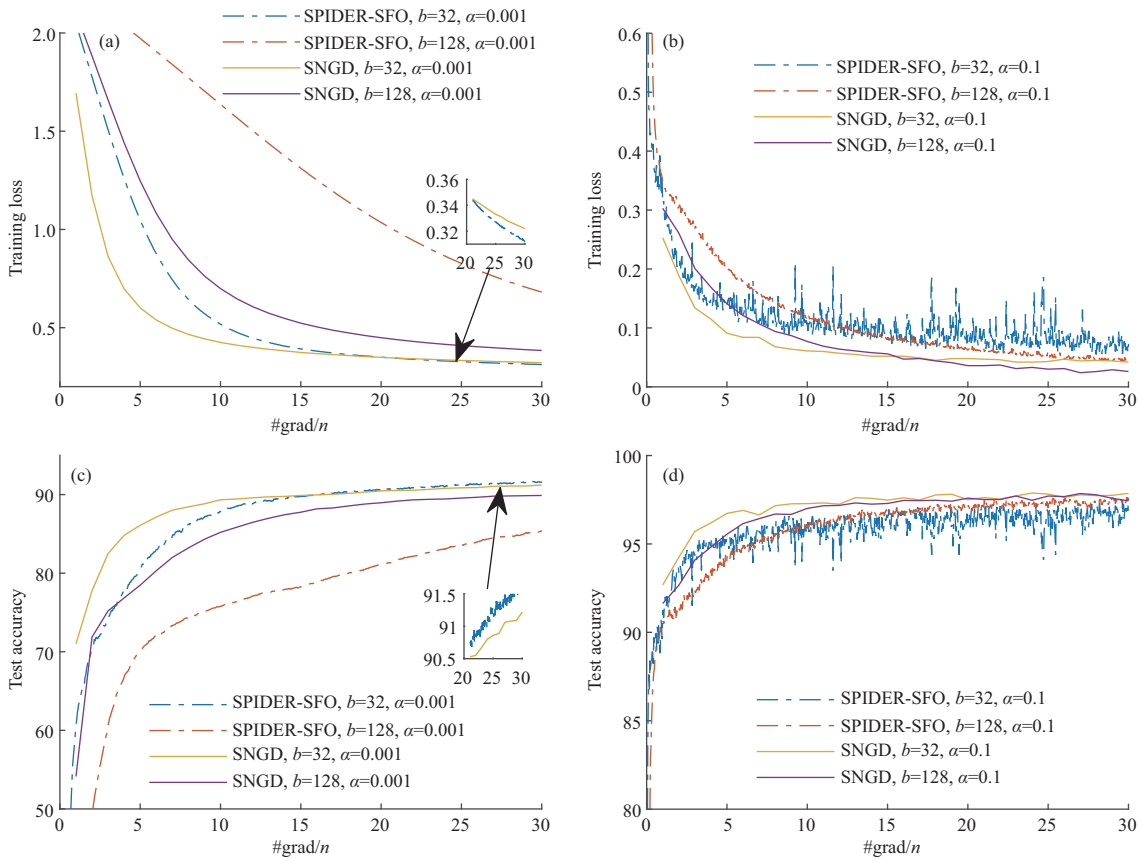

Figure 2 (Color online) Training loss and test accuracy of a small non-convex model.

terms of both training loss and test accuracy after 20 epochs. However, by comparing the four subfigures in Figure 2, we can observe that SNGD with the relatively large learning rate $\alpha = 0.1$ achieves better performance than SPIDER-SFO with $\alpha = 0.1$ or 0.001. Furthermore, according to the learning curves in Figures 2(b) and (d), we can observe that SPIDER-SFO suffers large variance if we adopt a large learning rate for it, while the learning curves of SNGD are smooth. This empirical phenomenon is consistent with the theoretical analysis in Section 5. We also try some other B and M for SPIDER-SFO and similar phenomenons are observed.

6.2 Large non-convex model

Next, we evaluate S-SNGD by training ResNet20 and ResNet56 [25] on the dataset CIFAR10 [26]. In CIFRA10, the training set includes 50000 samples and the testing set includes 10000 samples. Since each sample in CIFAR10 is a 32×32 RGB picture, we scale the pixels in each picture to $[0, 1]$ as that in the website[3]. These two models contain more parameters than the previous fully-connected network. We set the weight decay of the two models as 0.0001. In both SPIDER-SFO and SNGD, we set the constant learning rate $\alpha \in \{0.5, 0.1\}$. In SPIDER-SFO, we still set $\beta = 1$, $B = 1024$, $M = B/(2b)$. In S-SNGD, we set two different kinds of learning rate strategies: (1) poly power learning rate, i.e., $\alpha_t = \alpha(1 - t/T)^r$, $\alpha = 0.5$ and $r = 1.1$; (2) multi-step learning rate, i.e., the initial $\alpha = 0.5$ and α is divided by 10 at the 80th and 120th epochs. S-SNGD with the poly power learning rate strategy is denoted by S-SNGD(v1), and S-SNGD with the multi-step learning rate strategy is denoted by S-SNGD(v2). We set the batch size to be 128 for these methods. The results are shown in Figure 3. We can observe that both S-SNGD(v1) and S-SNGD(v2) achieve better performance than SNGD and SPIDER-SFO with different learning rates. Since S-SNGD(v2) and SNGD with $\alpha = 0.5$ are almost the same before 80 epochs, by comparing the learning curves of S-SNGD(v2) and SNGD with $\alpha = 0.5$, we get that if we adopt a large initial learning rate in S-SNGD, then reducing learning rate is necessary for S-SNGD to achieve better performance.

3) https://github.com/kuangliu/pytorch-cifar/blob/master/main.py.

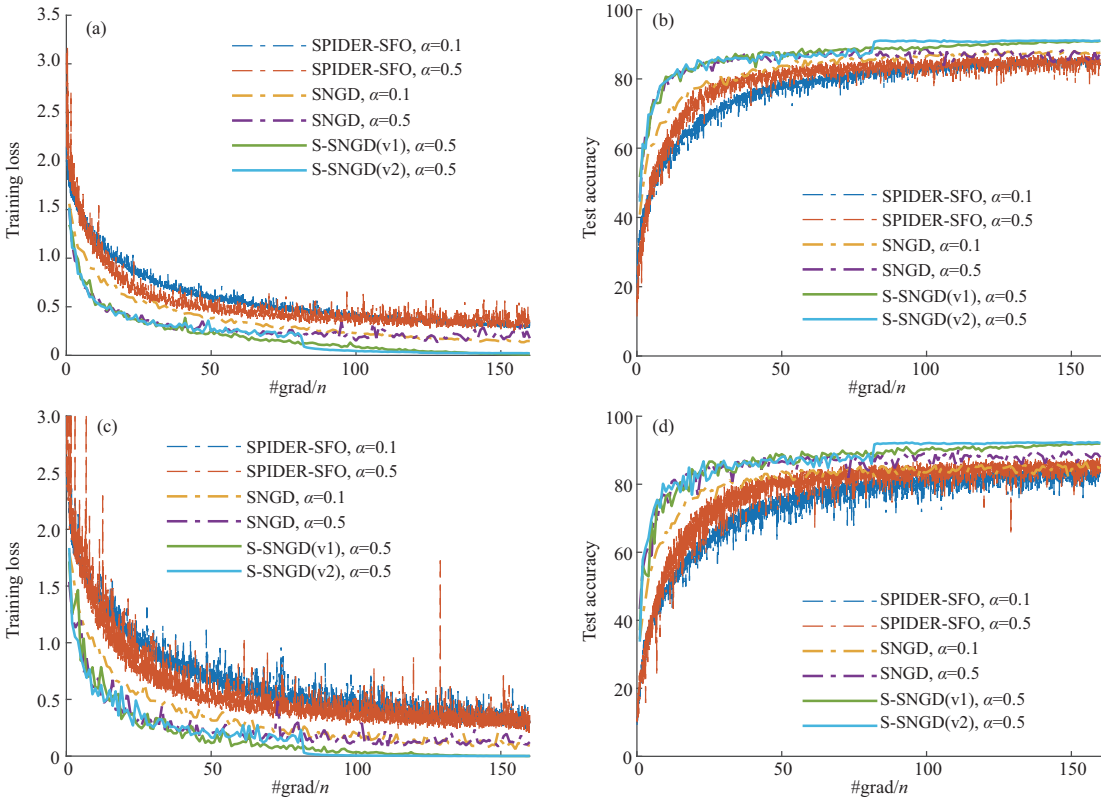

Figure 3 (Color online) Training loss and test accuracy of two large non-convex models. (a) Training loss of ResNet20; (b) test accuracy of ResNet20; (c) training loss of ResNet56; (d) test accuracy of ResNet56.

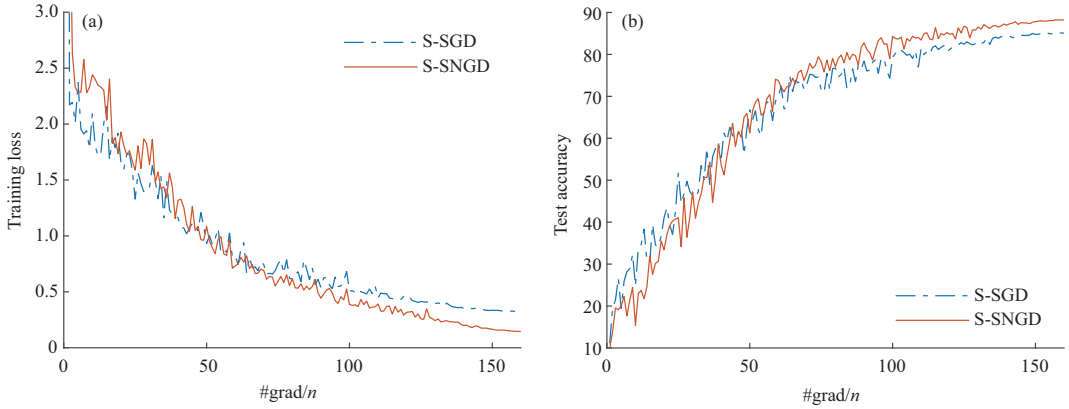

Figure 4 (Color online) Comparison between S-SNGD and S-SGD.

Furthermore, we can observe the large variance in the learning curves of SPIDER-SFO, which is similar to that in the experiments on the small non-convex model.

We also compare S-SNGD with stagewise SGD (S-SGD) [27] by training ResNet20 on CIFAR10. Here, we adopt S-SGD rather than standard SGD for comparison because existing studies [9,25,27] have shown that S-SGD can achieve better performance than standard SGD. In each stage, S-SGD runs SGD with a fixed learning rate and it decreases the learning rate by stage as that in S-SNGD. S-SGD has been widely used in many deep learning applications [9, 25]. The work in [11] empirically shows that when the batch size is large, S-SGD will make $\boldsymbol{w}$ fall into the region of saddle points or sharp minimums and lead to a worse test accuracy. Hence, we set the batch size as 4096 for both S-SGD and S-SNGD to show that S-SNGD can more easily escape the saddle points or sharp local minimums than S-SGD. We adopt the poly power learning rate strategy for both S-SNGD and S-SGD, i.e., $\alpha_t = \alpha(1 - t/T)^r$, where $r = 1.1$. We respectively tune the initial learning rate α for S-SNGD and S-SGD to guarantee that both of them achieve the best training loss. The weight decay is 0.0001. The results are presented in Figure 4.

We can observe that S-SNGD achieves better performance than S-SGD on both training loss and test accuracy, which verifies that S-SNGD can more easily escape the saddle points or sharp local minimums than S-SGD.

7 Conclusion

In this paper, we theoretically prove the convergence of SNGD for non-convex problems. Furthermore, we propose a new method, called S-SNGD, to improve the performance of SNGD. Different from SNGD in which a small constant learning rate is necessary to guarantee the convergence, S-SNGD can adopt a large initial learning rate and reduce the learning rate by stage. The convergence of S-SNGD can also be guaranteed for non-convex problems. Empirical results on deep neural networks show that S-SNGD achieves better performance than SNGD in terms of both training loss and test accuracy.

Acknowledgements This work was supported by Science and Technology Project of State Grid Corporation of China (Grant No. SGGR0000XTJS1900448).

References

1 Bottou L. Online learning and stochastic approximations. On-line Learn Neural Netw, 1998, 17: 142
2 Robbins H, Monro S. A stochastic approximation method. Ann Math Statist, 1951, 22: 400–407
3 Chen C Y, Wang W L, Zhang Y Z, et al. A convergence analysis for a class of practical variance-reduction stochastic gradient MCMC. Sci China Inf Sci, 2019, 62: 012101
4 Tseng P. An incremental gradient(-projection) method with momentum term and adaptive stepsize rule. SIAM J Optim, 1998, 8: 506–531
5 Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res, 2011, 12: 2121–2159
6 Ghadimi S, Lan G. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. Math Program, 2016, 156: 59–99
7 Ding F, Yang H Z, Liu F. Performance analysis of stochastic gradient algorithms under weak conditions. Sci China Ser F-Inf Sci, 2008, 51: 1269–1280
8 Nemirovski A, Juditsky A, Lan G, et al. Robust stochastic approximation approach to stochastic programming. SIAM J Optim, 2009, 19: 1574–1609
9 Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks. In: Proceedings of Advances in Neural Information Processing Systems, Lake Tahoe, 2012. 1097–1105
10 LeCun Y A, Bottou L, Orr G B, et al. Neural Networks: Tricks of the Trade. Berlin: Springer Science & Business Media, 2012. 9–48
11 Keskar N S, Mudigere D, Nocedal J, et al. On large-batch training for deep learning: generalization gap and sharp minima. In: Proceedings of International Conference on Learning Representations, Toulon, 2017
12 Sutskever I, Martens J, Dahl G E, et al. On the importance of initialization and momentum in deep learning. In: Proceedings of International Conference on Machine Learning, Atlanta, 2013. 1139–1147
13 Hazan E, Levy K, Shalev-Shwartz S. Beyond convexity: stochastic quasi-convex optimization. In: Proceedings of Advances in Neural Information Processing Systems, Montréal, 2015. 1594–1602
14 Levy K Y. The power of normalization: faster evasion of saddle points. 2016. ArXiv:1611.04831
15 Fang C, Li C J, Lin Z, et al. Spider: near-optimal non-convex optimization via stochastic path-integrated differential estimator. In: Proceedings of Advances in Neural Information Processing Systems, Montréal, 2018. 689–699
16 Zhang J, He T, Sra S, et al. Why gradient clipping accelerates training: a theoretical justification for adaptivity. In: Proceedings of International Conference on Learning Representations, Addis Ababa, 2020
17 Nesterov Y E. Introductory Lectures on Convex Optimization: A Basic Course. Berlin: Springer Science & Business Media, 2004
18 Wilson A C, Mackey L, Wibisono A. Accelerating rescaled gradient descent: fast optimization of smooth functions. In: Proceedings of Advances in Neural Information Processing Systems, Vancouver, 2019. 13533–13543
19 Ge R, Huang F, Jin C, et al. Escaping from saddle points-online stochastic gradient for tensor decomposition. In: Proceedings of Conference on Learning Theory, Paris, 2015. 797–842
20 Zinkevich M. Online convex programming and generalized infinitesimal gradient ascent. In: Proceedings of International Conference on Machine learning, Washington, 2003. 928–936
21 Johnson R, Zhang T. Accelerating stochastic gradient descent using predictive variance reduction. In: Proceedings of Advances in Neural Information Processing Systems, Lake Tahoe, 2013. 315–323
22 Lei L, Ju C, Chen J, et al. Non-convex finite-sum optimization via SCSG methods. In: Proceedings of Advances in Neural Information Processing Systems, Long Beach, 2017. 2348–2358
23 Allen Z, Bengio S, Wallach H, et al. Natasha2-faster non-convex optimization than SGD. In: Proceedings of Advances in Neural Information Processing Systems, Montréal, 2018. 2680–2691
24 Defazio A, Bottou L. On the ineffectiveness of variance reduced optimization for deep learning. In: Proceedings of Advances in Neural Information Processing Systems, Vancouver, 2019. 1753–1763
25 He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, 2016. 770–778
26 Krizhevsky A, Hinton G E. Learning Multiple Layers of Features From Tiny Images. Technical Report TR-2009. 2009
27 Yuan Z, Yan Y, Jin R, et al. Stagewise training accelerates convergence of testing error over SGD. In: Proceedings of Advances in Neural Information Processing Systems, Vancouver, 2019. 2604–2614