Journal of Optimization Theory and Applications
https://doi.org/10.1007/s10957-022-02132-w

Momentum-Based Variance-Reduced Proximal Stochastic Gradient Method for Composite Nonconvex Stochastic Optimization

Yangyang Xu[1] · **Yibo Xu**[2]

Received: 25 April 2021 / Accepted: 27 October 2022

Abstract
Stochastic gradient methods (SGMs) have been extensively used for solving stochastic problems or large-scale machine learning problems. Recent works employ various techniques to improve the convergence rate of SGMs for both convex and nonconvex cases. Most of them require a large number of samples in some or all iterations of the improved SGMs. In this paper, we propose a new SGM, named PStorm, for solving nonconvex nonsmooth stochastic problems. With a momentum-based variance reduction technique, PStorm can achieve the optimal complexity result $O(\varepsilon^{-3})$ to produce a stochastic ε-stationary solution, if a mean-squared smoothness condition holds. Different from existing optimal methods, PStorm can achieve the $O(\varepsilon^{-3})$ result by using only one or $O(1)$ samples in every update. With this property, PStorm can be applied to online learning problems that favor real-time decisions based on one or $O(1)$ new observations. In addition, for large-scale machine learning problems, PStorm can generalize better by small-batch training than other optimal methods that require large-batch training and the vanilla SGM, as we demonstrate on training a sparse fully-connected neural network and a sparse convolutional neural network.

Keywords Stochastic gradient method · Variance reduction · Momentum · Small-batch training

Mathematics Subject Classification 90C15 · 65K05 · 68Q25

Communicated by Amir Beck.

✉ Yangyang Xu
xuy21@rpi.edu

Yibo Xu
yibox@clemson.edu

[1] Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180, USA

[2] School of Mathematical and Statistical Sciences, Clemson University, Clemson, SC 29634, USA

Published online: 02 December 2022

1 Introduction

The stochastic approximation method first appears in [26] for solving a root-finding problem. Nowadays, its first-order version, or the stochastic gradient method (SGM), has been extensively used to solve machine learning problems that involve huge amounts of given data and also to stochastic problems that involve uncertain streaming data. Complexity results of SGMs have been well established for convex problems. Many recent research papers on SGMs focus on nonconvex cases.

In this paper, we consider the regularized nonconvex stochastic programming

$$\Phi^* = \min_{\mathbf{x}\in\mathbb{R}^n} \Phi(\mathbf{x}) := \left\{F(\mathbf{x}) \equiv \mathbb{E}_\xi[f(\mathbf{x};\xi)]\right\} + r(\mathbf{x}), \tag{1.1}$$

where $f(\cdot;\xi)$ is a smooth nonconvex function almost surely for ξ, and r is a closed convex function on $\mathbb{R}^n$. Examples of (1.1) include the sparse online matrix factorization [19], the online nonnegative matrix factorization [40], and the streaming PCA (by a unit-ball constraint) [21]. In addition, as ξ follows a uniform distribution on a finite set $\Xi = \{\xi_1, \ldots, \xi_N\}$, (1.1) recovers the so-called finite-sum structured problem. It includes most regularized machine learning problems such as the sparse bilinear logistic regression [28], the sparse convolutional neural network [18], and the group sparse regularized deep neural networks [27].

1.1 Background

When $r \equiv 0$, the recent work [3] gives an $O(\varepsilon^{-3})$ lower complexity bound of SGMs to produce a stochastic ε-stationary solution of (1.1) (see Definition 1.2), by assuming the so-called mean-squared smoothness condition (see Assumption 2). Several variance-reduced SGMs [6, 9, 32, 33] have achieved an $O(\varepsilon^{-3})$ or $\tilde{O}(\varepsilon^{-3})$ complexity result.[1] Among them, [6, 9] only consider smooth cases, i.e., $r \equiv 0$ in (1.1), and [32, 33] study nonsmooth problems in the form of (1.1). To reach an $O(\varepsilon^{-3})$ complexity result, the Hybrid-SGD method in [32] needs $O(\varepsilon^{-1})$ samples at the initial step and then at least two samples at each update, while [9, 33] require $O(\varepsilon^{-2})$ samples after every fixed number of updates. The STORM method in [6] requires one single sample of ξ at each update, but it only applies to smooth problems. Practically on training a (deep) machine learning model, small-batch training is often used to have better generalization [14, 20]. In addition, for certain applications such as reinforcement learning [30], one single sample can usually be obtained, depending on the stochastic environment and the current decision. Furthermore, regularization terms can improve generalization of a machine learning model, even for training a neural network [34]. We aim at designing a new SGM for solving the nonconvex nonsmooth problem (1.1) and achieving a (near)-optimal[2] complexity result by using $O(1)$ (that can be *one*) samples at each update.

[1] Throughout the paper, we use $\tilde{O}$ to suppress an additional polynomial term of $|\log\varepsilon|$.

[2] By "optimal," we mean that the complexity result can reach the lower bound result; a result is "near optimal," if it has an additional logarithmic term or a polynomial of logarithmic term than the lower bound.

1.2 Mirror-Prox Algorithm

Our algorithm is a mirror-prox SGM, and we adopt the momentum technique to reduce variance of the stochastic gradient in order to achieve a (near)-optimal complexity result.

Let w be a continuously differentiable and 1-strongly convex function on $\mathrm{dom}(r)$, i.e.,

$$w(\mathbf{y}) \geq w(\mathbf{x}) + \langle \nabla w(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{1}{2}\|\mathbf{y} - \mathbf{x}\|^2, \ \forall \mathbf{x}, \mathbf{y} \in \mathrm{dom}(r).$$

The Bregman divergence induced by w is defined as

$$V(\mathbf{x}, \mathbf{z}) = w(\mathbf{x}) - w(\mathbf{z}) - \langle \nabla w(\mathbf{z}), \mathbf{x} - \mathbf{z} \rangle. \tag{1.2}$$

At each iteration of our algorithm, we obtain one or a few samples of ξ, compute stochastic gradients at the previous and current iterates using the *same* samples, and then perform a mirror-prox momentum stochastic gradient update. The pseudocode is shown in Algorithm 1. We name it as PStorm as it can be viewed as a proximal version of the Storm method in [6]. Notice that when $\beta_k = 1, \forall k \geq 0$, the algorithm reduces to the non-accelerated stochastic proximal gradient method. However, our analysis does not apply to this case, for which an innovative analysis can be found in [7].

Algorithm 1: Momentum-based variance-reduced proximal stochastic gradient method for (1.1)

Input: max iteration numer K, minibatch size m, and positive sequences $\{\beta_k\} \subseteq (0, 1)$ and $\{\eta_k\}$.

Initialization: choose $\mathbf{x}^0 \in \mathrm{dom}(r)$ and let $\mathbf{d}^0 = \frac{1}{m_0} \sum_{\xi \in B_0} \nabla f(\mathbf{x}^0; \xi)$ with m_0 i.i.d. samples $B_0 = \{\xi_1^0, \ldots, \xi_{m_0}^0\}$

for $k = 0, 1, \ldots, K-1$ **do**

Update $\mathbf{x}$ by

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ \langle \mathbf{d}^k, \mathbf{x} \rangle + \frac{1}{\eta_k} V(\mathbf{x}, \mathbf{x}^k) + r(\mathbf{x}) \right\}. \tag{1.3}$$

Obtain m i.i.d. samples $B_{k+1} = \{\xi_1^{k+1}, \ldots, \xi_m^{k+1}\}$ and let

$$\mathbf{v}^{k+1} = \frac{1}{m} \sum_{\xi \in B_{k+1}} \nabla f(\mathbf{x}^{k+1}; \xi), \quad \mathbf{u}^{k+1} = \frac{1}{m} \sum_{\xi \in B_{k+1}} \nabla f(\mathbf{x}^k; \xi). \tag{1.4}$$

Let $\mathbf{d}^{k+1} = \mathbf{v}^{k+1} + (1 - \beta_k)(\mathbf{d}^k - \mathbf{u}^{k+1})$.

Return $\mathbf{x}^\tau$ with τ selected from $\{0, 1, \ldots, K-1\}$ uniformly at random or by the distribution

$$\mathrm{Prob}(\tau = k) = \frac{\frac{\eta_k}{4}(1 - \eta_k L) - \frac{\eta_k^2}{5m\eta_{k+1}}(1 - \beta_k)^2}{\sum_{j=0}^{K-1} \left(\frac{\eta_j}{4}(1 - \eta_j L) - \frac{\eta_j^2}{5m\eta_{j+1}}(1 - \beta_j)^2 \right)}, \ k = 0, 1, \ldots, K-1. \tag{1.5}$$

1.3 Related Works

Many efforts have been made on analyzing the convergence and complexity of SGMs for solving nonconvex stochastic problems, e.g., [1, 6–11, 32, 33, 37]. We list comparison results on the complexity in Table 1.

The work [10] appears to be the first one that conducts complexity analysis of SGM for nonconvex stochastic problems. It introduces a randomized SGM. For a smooth nonconvex problem, the randomized SGM can produce a stochastic ε-stationary solution within $O(\varepsilon^{-4})$ SG iterations. The same-order complexity result is then extended in [11] to nonsmooth nonconvex stochastic problems in the form of (1.1). To achieve an $O(\varepsilon^{-4})$ complexity result, the accelerated prox-SGM in [11] needs to take $\Theta(k)$ samples at the k-th update for each k. Assuming a weak-convexity condition and using the tool of Moreau envelope, [7] establishes an $O(\varepsilon^{-4})$ complexity result of stochastic subgradient method for solving more general nonsmooth nonconvex problems to produce a near-ε stochastic stationary solution (see [7] for the precise definition).

In general, the $O(\varepsilon^{-4})$ complexity result cannot be improved for smooth nonconvex stochastic problems, as [3] shows that for the problem $\min_{\mathbf{x}} F(\mathbf{x})$ where F is smooth, any SGM that can access unbiased SG with bounded variance needs $\Omega(\varepsilon^{-4})$ SGs to produce a solution $\bar{\mathbf{x}}$ such that $\mathbb{E}\big[\|\nabla F(\bar{\mathbf{x}})\|\big] \le \varepsilon$. However, with one additional mean-squared smoothness condition on each unbiased SG, the complexity can be reduced to $O(\varepsilon^{-3})$, which has been reached by a few variance-reduced SGMs [6, 9, 23, 32, 33]. These methods are closely related to ours. Below we briefly review them.

Spider. To find a stochastic ε-stationary solution of (1.1) with $r \equiv 0$, [9] proposes the Spider method with the update: $\mathbf{x}^{k+1} = \mathbf{x}^k - \eta_k \mathbf{v}^k$ for each $k \ge 0$. Here, $\mathbf{v}^k$ is set to

$$\mathbf{v}^k = \begin{cases} \frac{1}{|B_k|}\sum_{\xi\in B_k}\big(\nabla f(\mathbf{x}^k;\xi) - \nabla f(\mathbf{x}^{k-1};\xi)\big) + \mathbf{v}^{k-1}, & \text{if } \mathrm{mod}(k,q)\neq 0, \\ \frac{1}{|C_k|}\sum_{\xi\in C_k}\nabla f(\mathbf{x}^k;\xi), & \text{otherwise,} \end{cases} \tag{1.6}$$

where $|B_k| = \Theta(\frac{1}{q\varepsilon^2})$, $|C_k| = \Theta(\varepsilon^{-2})$, and $q = \Theta(\varepsilon^{-1})$ or $q = \Theta(\varepsilon^{-2})$. Under the mean-squared smoothness condition (see Assumption 2), the Spider method can produce a stochastic ε-stationary solution with $O(\varepsilon^{-3})$ sample gradients, by choosing appropriate learning rate η_k (roughly in the order of $\frac{1}{q\|\mathbf{v}^k\|}$).

Storm. [6] focuses on a smooth nonconvex stochastic problem, i.e., (1.1) with $r \equiv 0$. It proposes the Storm method, which can be viewed as a special case of Algorithm 1 with $m_0 = m = 1$ applied to the smooth problem. However, its analysis and also algorithm design rely on the knowledge of a uniform bound on $\{\|\nabla f(\mathbf{x};\xi)\|\}$ or on the bound of the variance of the stochastic gradient. In addition, because the learning rate of Storm is set dependent on the sampled stochastic gradient, its analysis needs almost-sure uniform smoothness of $f(\mathbf{x};\xi)$. This assumption is significantly stronger than the mean-squared smoothness condition, and also the uniform smoothness constant can be much larger than an averaged one.

Spiderboost. [33] extends Spider into solving a nonsmooth nonconvex stochastic problem in the form of (1.1) by proposing a so-called Spiderboost method. Spiderboost

Table 1 Comparison of the complexity results of several methods in the literature to our method to produce a stochastic ε-stationary solution of a nonconvex stochastic optimization problem

Method	Problem	Key assumptions	#Samples at k-th iteration	Complexity
Accelerated prox-SGM [11]	$\min_{\mathbf{x}}\{\mathbb{E}_{\xi}[f(\mathbf{x};\xi)]+r(\mathbf{x})\}$	$\mathbb{E}_{\xi}[f(\mathbf{x};\xi)]$ is smooth r is convex	$\Theta(k)$	$O(\varepsilon^{-4})$
Stochastic subgradient [7]	$\min_{\mathbf{x}}\{\mathbb{E}_{\xi}[f(\mathbf{x};\xi)]+r(\mathbf{x})\}$	$\mathbb{E}_{\xi}[f(\mathbf{x};\xi)]$ is weakly-convex r is convex bounded stochastic subgrad.	$O(1)$	$O(\varepsilon^{-4})$
Spider [9]	$\min_{\mathbf{x}}\{\mathbb{E}_{\xi}[f(\mathbf{x};\xi)]\}$	mean-squared smoothness see Assumption 2	$\Theta(\varepsilon^{-2})$ or $\Theta(\varepsilon^{-1})$	$O(\varepsilon^{-3})$
Storm [6]	$\min_{\mathbf{x}}\{\mathbb{E}_{\xi}[f(\mathbf{x};\xi)]\}$	$f(\cdot;\xi)$ is smooth a.s. bounded stochastic grad.*	1	$\tilde{O}(\varepsilon^{-3})$
Spiderboost [33]	$\min_{\mathbf{x}}\{\mathbb{E}_{\xi}[f(\mathbf{x};\xi)]+r(\mathbf{x})\}$	mean-squared smoothness r is convex	$\Theta(\varepsilon^{-2})$ or $\Theta(\varepsilon^{-1})$	$O(\varepsilon^{-3})$
Hybrid-SGD [32]	$\min_{\mathbf{x}}\{\mathbb{E}_{\xi}[f(\mathbf{x};\xi)]+r(\mathbf{x})\}$	mean-squared smoothness r is convex	$\Theta(\varepsilon^{-1})$ if $k=0$ $O(1)$ but at least 2 if $k>0$	$O(\varepsilon^{-3})$
PStorm (**This paper**)	$\min_{\mathbf{x}}\{\mathbb{E}_{\xi}[f(\mathbf{x};\xi)]+r(\mathbf{x})\}$	mean-squared smoothness r is convex	$O(1)$ and can be 1 varying stepsize	$\tilde{O}(\varepsilon^{-3})$
			$O(1)$ and can be 1 constant stepsize	$O(\varepsilon^{-3})$

*: The boundedness assumption on stochastic gradient made by Storm [6] can be lifted if a bound σ on the variance of the stochastic gradient is known

To obtain the listed results, all the compared methods assume unbiasedness and variance boundedness of the stochastic (sub)gradients. The results only show the dependence on ε. All other constants (e.g., the smoothness constant L and the initial objective error) are hidden in the big-O. More complete results of the proposed method are given in Remarks 2.2 and 2.3

iteratively performs the update

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \langle \mathbf{v}^k, \mathbf{x} \rangle + \tfrac{1}{\eta} V(\mathbf{x}, \mathbf{x}^k) + r(\mathbf{x}), \tag{1.7}$$

where V denotes the Bregman divergence induced by a strongly-convex function, and $\mathbf{v}^k$ is set by (1.6) with $q = |B_k| = \Theta(\varepsilon^{-1})$ and $|C_k| = \Theta(\varepsilon^{-2})$. Under the mean-squared smoothness condition, Spiderboost reaches a complexity result of $O(\varepsilon^{-3})$ by choosing $\eta = \frac{1}{2L}$, where L is the smoothness constant.

Hybrid-SGD. [32] considers a nonsmooth nonconvex stochastic problem in the form of (1.1). It proposes a proximal stochastic method, called Hybrid-SGD, as a hybrid of SARAH [22] and an unbiased SGD. The Hybrid-SGD performs the update $\mathbf{x}^{k+1} = (1-\gamma_k)\mathbf{x}^k + \gamma_k \hat{\mathbf{x}}^{k+1}$ for each $k \geq 0$, where

$$\hat{\mathbf{x}}^{k+1} = \arg\min_{\mathbf{x}} \langle \mathbf{v}^k, \mathbf{x} \rangle + \tfrac{1}{2\eta_k} \|\mathbf{x} - \mathbf{x}^k\|^2 + r(\mathbf{x}).$$

Here, the sequence $\{\mathbf{v}^k\}$ is set by $\mathbf{v}^0 = \frac{1}{|B_0|} \sum_{\xi \in B_0} \nabla f(\mathbf{x}^0; \xi)$ with $|B_0| = \Theta(\varepsilon^{-1})$ for a given $\varepsilon > 0$ and

$$\mathbf{v}^k = \beta_{k-1}\mathbf{v}^{k-1} + \beta_{k-1}\big(\nabla f(\mathbf{x}^k; \xi_k) - \nabla f(\mathbf{x}^{k-1}; \xi_k)\big) + (1-\beta_{k-1})\nabla f(\mathbf{x}^k; \zeta_k), \tag{1.8}$$

where ξ_k and ζ_k are two independent samples of ξ. A mini-batch version of Hybrid-SGD is also given in [32]. By choosing appropriate constant parameters $\{(\beta_k, \gamma_k, \eta_k)\}$, Hybrid-SGD can reach an $O(\varepsilon^{-3})$ complexity result. Although the update of $\mathbf{v}^k$ requires only two or $O(1)$ samples, its initial setting needs $O(\varepsilon^{-1})$ samples. As explained in [32, Remark 3], if the initial minibatch size is $|B_0| = O(1)$, then the complexity result of Hybrid-SGD will be worsened to $O(\varepsilon^{-4})$. It is possible to reduce the $O(\varepsilon^{-4})$ complexity by using an adaptive β_k as mentioned in [32, Remark 3] to adopt the technique in [31]. This way, a near-optimal $\tilde{O}(\varepsilon^{-3})$ result may be shown for Hybrid-SGD without a large initial minibatch. Notice that with $\xi_k = \zeta_k, \forall\, k$, the stochastic gradient estimator by Hybrid-SGD will reduce to that by Storm, and further with $\gamma_k = 1, \forall\, k$, the update of Hybrid-SGD will recover ours. However, the analysis in [31, 32] relies on the independence of ξ_k and ζ_k and the condition $\gamma_k \in (0, 1)$, and thus it does not apply to our algorithm.

More. There are many other works analyzing complexity results of SGMs on solving nonconvex finite-sum structured problems, e.g., [2, 13, 17, 24]. These results often emphasize the dependence on the number of component functions and also the target error tolerance ε. In addition, several works have analyzed adaptive SGMs for nonconvex finite-sum or stochastic problems, e.g., [5, 36, 41]. Moreover, along the direction of accelerating SGMs, some works (e.g., [31, 35, 38, 39]) have considered minimax structured or compositional optimization problems. An exhaustive review of all these works is impossible and also beyond the scope of this paper. We refer interested readers to those papers and the references therein.

1.4 Contributions

Our main contributions are about the algorithm design and analysis.

- We design a momentum-based variance-reduced mirror-prox stochastic gradient method for solving nonconvex nonsmooth stochastic problems. The proposed method generalizes Storm in [6] from smooth cases to nonsmooth cases. In addition, with one single data sample per iteration, it achieves, by taking varying stepsizes, the same near-optimal complexity result $\tilde{O}(\varepsilon^{-3})$ under a mean-squared smooth condition, which is weaker than the almost-sure uniform smoothness condition assumed in [6].
- When constant stepsizes are adopted, the proposed method can achieve the optimal $O(\varepsilon^{-3})$ complexity result, by using one single or $O(1)$ data samples per iteration. While Spiderboost [33] can also achieve the optimal $O(\varepsilon^{-3})$ complexity result for stochastic nonconvex nonsmooth problems, it needs $\Theta(\varepsilon^{-2})$ data samples every $\Theta(\varepsilon^{-1})$ iterations and $\Theta(\varepsilon^{-1})$ samples for every other iteration. To achieve the optimal $O(\varepsilon^{-3})$ complexity result, Hybrid-SGD [32] needs $\Theta(\varepsilon^{-1})$ data samples for the first iteration and at least two samples for all other iterations. However, if only $O(1)$ samples can be obtained initially, the worst-case complexity result of Hybrid-SGD with constant stepsize will increase to $O(\varepsilon^{-4})$. Our proposed method is the first one that uses only one or $O(1)$ samples per iteration and can still reach the optimal complexity result, and thus it can be applied to online learning problems that need real-time decision based on possibly one or several new data samples.
- Furthermore, the proposed method only needs an estimate of the smoothness parameter and is easy to tune to have good performance. Empirically, we observe that it converges faster than a vanilla SGD and can give higher testing accuracy than Spiderboost and Hybrid-SGD on training sparse neural networks.

1.5 Notation, Definitions, and Outline

We use bold lowercase letters $\mathbf{x}, \mathbf{y}, \mathbf{g}, \ldots$ for vectors. $\mathbb{E}_{B_k}$ denotes the expectation about a mini-batch set B_k conditionally on the all previous history, and $\mathbb{E}$ denotes the full expectation. $|B_k|$ counts the number of elements in the set B_k. We use $\|\cdot\|$ for the Euclidean norm. A differentiable function F is called L-smooth, if $\|\nabla F(\mathbf{x}) - \nabla F(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$ for all $\mathbf{x}$ and $\mathbf{y}$.

Definition 1.1 (*proximal gradient mapping*) Given $\mathbf{d}$, $\mathbf{x} \in \mathrm{dom}(r)$, and $\eta > 0$, we define $P(\mathbf{x}, \mathbf{d}, \eta) = \frac{1}{\eta}(\mathbf{x} - \mathbf{x}^+)$, where $\mathbf{x}^+ = \arg\min_{\mathbf{y}} \left\{ \langle \mathbf{d}, \mathbf{y} \rangle + \frac{1}{\eta} V(\mathbf{y}, \mathbf{x}) + r(\mathbf{y}) \right\}$.

By the proximal gradient mapping, if a point $\bar{\mathbf{x}} \in \mathrm{dom}(r)$ is an optimal solution of (1.1), then it must satisfy $P(\bar{\mathbf{x}}, \nabla F(\bar{\mathbf{x}}), \eta) = \mathbf{0}$ for any $\eta > 0$. Based on this observation, we define a near-stationary solution as follows. This definition is standard and has been adopted in other papers, e.g., [33].

Definition 1.2 (*stochastic ε-stationary solution*) Given $\varepsilon > 0$, a random vector $\mathbf{x} \in \mathrm{dom}(r)$ is called a stochastic ε-stationary solution of (1.1) if for some $\eta > 0$, it holds $\mathbb{E}[\|P(\mathbf{x}, \nabla F(\mathbf{x}), \eta)\|^2] \leq \varepsilon^2$.

From [12, Lemma 1], it holds

$$\langle \mathbf{d},\, P(\mathbf{x},\mathbf{d},\eta)\rangle \geq \|P(\mathbf{x},\mathbf{d},\eta)\|^2 + \frac{1}{\eta}\big(r(\mathbf{x}^+) - r(\mathbf{x})\big). \tag{1.9}$$

In addition, the proximal gradient mapping is nonexpansive from [12, Proposition 1], i.e.,

$$\|P(\mathbf{x},\mathbf{d}_1,\eta) - P(\mathbf{x},\mathbf{d}_2,\eta)\| \leq \|\mathbf{d}_1 - \mathbf{d}_2\|,\ \forall\, \mathbf{d}_1,\mathbf{d}_2,\ \forall\, \mathbf{x}\in \mathrm{dom}(r),\ \forall\, \eta > 0. \tag{1.10}$$

For each $k \geq 0$, we denote

$$\mathbf{g}^k = P(\mathbf{x}^k,\mathbf{d}^k,\eta_k), \quad \bar{\mathbf{g}}^k = P(\mathbf{x}^k,\nabla F(\mathbf{x}^k),\eta_k). \tag{1.11}$$

Notice that $\|\bar{\mathbf{g}}^k\|$ measures the violation of stationarity of $\mathbf{x}^k$. The gradient error is represented by

$$\mathbf{e}^k = \mathbf{d}^k - \nabla F(\mathbf{x}^k). \tag{1.12}$$

Outline. The rest of the paper is organized as follows. In Sect. 2, we establish complexity results of Algorithm 1. Numerical experiments are conducted in Sect. 3, and we conclude the paper in Sect. 4.

2 Convergence Analysis

In this section, we analyze the complexity result of Algorithm 1. Part of our analysis is inspired from that in [6] and [33]. In addition, we give a novel analysis that enables us to obtain the optimal $O(\varepsilon^{-3})$ complexity result by using $O(1)$ samples every iteration. Throughout our analysis, we make the following assumptions.

Assumption 1 (*finite optimal objective*) The optimal objective value Φ^* of (1.1) is finite.

Assumption 2 (*mean-squared smoothness*) The function $f(\cdot\,;\xi)$ satisfies the mean-squared smoothness condition: $\mathbb{E}_\xi\big[\|\nabla f(\mathbf{x};\xi) - \nabla f(\mathbf{y};\xi)\|^2\big] \leq L^2\|\mathbf{x}-\mathbf{y}\|^2,\ \forall\, \mathbf{x},\mathbf{y} \in \mathrm{dom}(r)$.

Assumption 3 (*unbiasedness and variance boundedness*) There is $\sigma > 0$ such that

$$\mathbb{E}_\xi[\nabla f(\mathbf{x};\xi)] = \nabla F(\mathbf{x}), \quad \mathbb{E}[\|\nabla f(\mathbf{x};\xi) - \nabla F(\mathbf{x})\|^2] \leq \sigma^2,\ \forall\, \mathbf{x}\in \mathrm{dom}(r). \tag{2.1}$$

It is easy to show that under Assumptions 2 and 3, the function $F(\mathbf{x}) = \mathbb{E}_\xi[f(\mathbf{x};\xi)]$ is L-smooth; see the arguments at the end of Section 2.2 of [32]. We first show a few lemmas. The lemma below estimates one-iteration progress. Its proof follows from [33].

Lemma 2.1 (one-iteration progress) *Let* $\{\mathbf{x}^k\}$ *be generated from Algorithm* 1. *If* F *is* L*-smooth, then*

$$\Phi(\mathbf{x}^{k+1}) - \Phi(\mathbf{x}^k) \le \frac{\eta_k}{2}(2-\eta_k L)\|\mathbf{e}^k\|^2 - \frac{\eta_k}{4}(1-\eta_k L)\|\bar{\mathbf{g}}^k\|^2,\ \forall k \ge 1,$$

where $\bar{\mathbf{g}}^k$ *is defined in* (1.11).

Proof By the L-smoothness of F and the definition of $\mathbf{g}^k$ in (1.11), we have

$$\begin{aligned} F(\mathbf{x}^{k+1}) - F(\mathbf{x}^k) &\le \langle \nabla F(\mathbf{x}^k), \mathbf{x}^{k+1} - \mathbf{x}^k\rangle + \frac{L}{2}\|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2 \\ &= -\eta_k \langle \nabla F(\mathbf{x}^k), \mathbf{g}^k\rangle + \frac{\eta_k^2 L}{2}\|\mathbf{g}^k\|^2. \end{aligned} \tag{2.2}$$

Using the definition of $\mathbf{e}^k$ in (1.12) and the inequality in (1.9), we have

$$-\langle \nabla F(\mathbf{x}^k), \mathbf{g}^k\rangle = \langle \mathbf{e}^k, \mathbf{g}^k\rangle - \langle \mathbf{d}^k, \mathbf{g}^k\rangle \le \langle \mathbf{e}^k, \mathbf{g}^k\rangle - \|\mathbf{g}^k\|^2 + \frac{1}{\eta_k}\big(r(\mathbf{x}^k) - r(\mathbf{x}^{k+1})\big).$$

Plugging the above inequality into (2.2) and rearranging terms give

$$\Phi(\mathbf{x}^{k+1}) - \Phi(\mathbf{x}^k) \le \eta_k \langle \mathbf{e}^k, \mathbf{g}^k\rangle - \eta_k\|\mathbf{g}^k\|^2 + \frac{\eta_k^2 L}{2}\|\mathbf{g}^k\|^2.$$

By the Cauchy–Schwartz inequality, it holds $\eta_k\langle \mathbf{e}^k, \mathbf{g}^k\rangle \le \frac{\eta_k}{2}\|\mathbf{e}^k\|^2 + \frac{\eta_k}{2}\|\mathbf{g}^k\|^2$, which together with the above inequality implies

$$\Phi(\mathbf{x}^{k+1}) - \Phi(\mathbf{x}^k) \le \frac{\eta_k}{2}\|\mathbf{e}^k\|^2 - \frac{\eta_k}{2}(1-\eta_k L)\|\mathbf{g}^k\|^2. \tag{2.3}$$

From (1.10) and the definitions of $\mathbf{g}^k$ and $\bar{\mathbf{g}}^k$ in (1.11), it follows

$$\begin{aligned} -\|\mathbf{g}^k\|^2 &\le -\frac{1}{2}\|\bar{\mathbf{g}}^k\|^2 + \|\mathbf{g}^k - \bar{\mathbf{g}}^k\|^2 \le -\frac{1}{2}\|\bar{\mathbf{g}}^k\|^2 + \|\mathbf{d}^k - \nabla F(\mathbf{x}^k)\|^2 \\ &= -\frac{1}{2}\|\bar{\mathbf{g}}^k\|^2 + \|\mathbf{e}^k\|^2. \end{aligned} \tag{2.4}$$

Now plug the above inequality into (2.3) to give the desired result. □

The next lemma gives a recursive bound on the gradient error vector sequence $\{\mathbf{e}^k\}$. Its proof follows that of [6, Lemma 2].

Lemma 2.2 (recursive bound on gradient error) *Under Assumptions 2 and 3, it holds*

$$\mathbb{E}\big[\|\mathbf{e}^{k+1}\|^2\big] \le \frac{2\beta_k^2\sigma^2}{m} + \frac{4(1-\beta_k)^2\eta_k^2L^2}{m}\mathbb{E}\big[\|\bar{\mathbf{g}}^k\|^2\big] + (1-\beta_k)^2\left(1+\frac{4\eta_k^2L^2}{m}\right)\mathbb{E}\big[\|\mathbf{e}^k\|^2\big], \forall\, k \ge 0,$$

where $\bar{\mathbf{g}}^k$ *and* $\mathbf{e}^k$ *are defined in* (1.11) *and* (1.12).

Proof First, notice $\mathbb{E}_{B_{k+1}}[\langle \mathbf{v}^{k+1}, \mathbf{e}^k\rangle] = \langle \nabla F(\mathbf{x}^{k+1}), \mathbf{e}^k\rangle$ and $\mathbb{E}_{B_{k+1}}[\langle \mathbf{u}^{k+1}, \mathbf{e}^k\rangle] = \langle \nabla F(\mathbf{x}^k), \mathbf{e}^k\rangle$, and thus

$$\mathbb{E}_{B_{k+1}}[\langle \mathbf{v}^{k+1} - \nabla F(\mathbf{x}^{k+1}), \mathbf{e}^k\rangle] = 0, \quad \mathbb{E}_{B_{k+1}}[\langle \mathbf{u}^{k+1} - \nabla F(\mathbf{x}^k), \mathbf{e}^k\rangle] = 0. \tag{2.5}$$

Hence, by writing $\mathbf{e}^{k+1} = \mathbf{v}^{k+1} - \nabla F(\mathbf{x}^{k+1}) + (1-\beta_k)(\nabla F(\mathbf{x}^k) - \mathbf{u}^{k+1}) + (1-\beta_k)\mathbf{e}^k$, we have

$$\begin{aligned}\mathbb{E}_{B_{k+1}}\big[\|\mathbf{e}^{k+1}\|^2\big] = \mathbb{E}_{B_{k+1}}\big[\|\mathbf{v}^{k+1} - \nabla F(\mathbf{x}^{k+1}) + (1-\beta_k)(\nabla F(\mathbf{x}^k) - \mathbf{u}^{k+1})\|^2\big] \\ +(1-\beta_k)^2\|\mathbf{e}^k\|^2.\end{aligned} \tag{2.6}$$

By the Young's inequality, it holds

$$\begin{aligned}&\|\mathbf{v}^{k+1} - \nabla F(\mathbf{x}^{k+1}) + (1-\beta_k)(\nabla F(\mathbf{x}^k) - \mathbf{u}^{k+1})\|^2 \\ &= \big\|\beta_k\big(\mathbf{v}^{k+1} - \nabla F(\mathbf{x}^{k+1})\big) + (1-\beta_k)\big(\mathbf{v}^{k+1} - \nabla F(\mathbf{x}^{k+1}) + \nabla F(\mathbf{x}^k) - \mathbf{u}^{k+1}\big)\big\|^2 \\ &\le 2\beta_k^2\|\mathbf{v}^{k+1} - \nabla F(\mathbf{x}^{k+1})\|^2 + 2(1-\beta_k)^2\|\mathbf{v}^{k+1} - \nabla F(\mathbf{x}^{k+1}) \\ &\quad + \nabla F(\mathbf{x}^k) - \mathbf{u}^{k+1}\|^2.\end{aligned} \tag{2.7}$$

From the definition of $\mathbf{v}^{k+1}$ and $\mathbf{u}^{k+1}$ in (1.4), we have

$$\begin{aligned}&\mathbb{E}_{B_{k+1}}\big[\|\mathbf{v}^{k+1} - \nabla F(\mathbf{x}^{k+1}) + \nabla F(\mathbf{x}^k) - \mathbf{u}^{k+1}\|^2\big] \\ &= \frac{1}{m^2}\mathbb{E}_{B_{k+1}}\left\|\sum_{\xi\in B_{k+1}}\Big(\nabla f(\mathbf{x}^{k+1};\xi) - \nabla f(\mathbf{x}^k;\xi) - \nabla F(\mathbf{x}^{k+1}) + \nabla F(\mathbf{x}^k)\Big)\right\|^2 \\ &= \frac{1}{m^2}\sum_{j=1}^m\mathbb{E}_{\xi_j^{k+1}}\left\|\nabla f(\mathbf{x}^{k+1};\xi_j^{k+1}) - \nabla f(\mathbf{x}^k;\xi_j^{k+1}) - \nabla F(\mathbf{x}^{k+1}) + \nabla F(\mathbf{x}^k)\right\|^2 \\ &\le \frac{1}{m^2}\sum_{j=1}^m\mathbb{E}_{\xi_j^{k+1}}\left\|\nabla f(\mathbf{x}^{k+1};\xi_j^{k+1}) - \nabla f(\mathbf{x}^k;\xi_j^{k+1})\right\|^2 \\ &\le \frac{L^2}{m}\|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2,\end{aligned} \tag{2.8}$$

where the second equality holds because of the i.i.d. samples in B_{k+1} and the zero mean of the random vector $\nabla f(\mathbf{x}^{k+1};\xi_j^{k+1}) - \nabla f(\mathbf{x}^k;\xi_j^{k+1}) - \nabla F(\mathbf{x}^{k+1}) + \nabla F(\mathbf{x}^k)$

resulted from unbiasedness in Assumption 3, the first inequality is due to the fact that the variance of a random vector is upper bounded by its second moment, and the last inequality follows from Assumption 2.

Now, take conditional expectation on both sides of (2.7), use (2.8), and substitute it into (2.6). We have

$$\begin{aligned}\mathbb{E}_{B_{k+1}}\big[\|\mathbf{e}^{k+1}\|^2\big] \le (1-\beta_k)^2\|\mathbf{e}^k\|^2 + 2\beta_k^2\mathbb{E}_{B_{k+1}}\big[\|\mathbf{v}^{k+1} - \nabla F(\mathbf{x}^{k+1})\|^2\big] \\ + \frac{2(1-\beta_k)^2L^2}{m}\mathbb{E}_{B_{k+1}}\big[\|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2\big].\end{aligned}$$

Taking a full expectation over the above inequality and using Assumption 3, we have

$$\begin{aligned}\mathbb{E}\big[\|\mathbf{e}^{k+1}\|^2\big] &\le (1-\beta_k)^2\mathbb{E}\big[\|\mathbf{e}^k\|^2\big] + \frac{2\beta_k^2\sigma^2}{m} + \frac{2(1-\beta_k)^2L^2}{m}\mathbb{E}\big[\|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2\big] \\ &= (1-\beta_k)^2\mathbb{E}\big[\|\mathbf{e}^k\|^2\big] + \frac{2\beta_k^2\sigma^2}{m} + \frac{2(1-\beta_k)^2\eta_k^2L^2}{m}\mathbb{E}\big[\|\mathbf{g}^k\|^2\big], \quad (2.9)\end{aligned}$$

where we have used $\mathbf{x}^{k+1} - \mathbf{x}^k = -\eta_k\mathbf{g}^k$ in the equality.

By similar arguments as those in (2.4), it holds

$$\|\mathbf{g}^k\|^2 \le 2\|\bar{\mathbf{g}}^k\|^2 + 2\|\mathbf{g}^k - \bar{\mathbf{g}}^k\|^2 \le 2\|\bar{\mathbf{g}}^k\|^2 + 2\|\mathbf{e}^k\|^2.$$

Plugging the above inequality into (2.9), we obtain the desired result. □

2.1 Results with Varying Stepsize

In this subsection, we show the convergence results of Algorithm 1 by taking varying stepsizes. Using Lemmas 2.1 and 2.2, we first show a convergence rate result by choosing the parameters that satisfy a general condition. Then we specify the choice of the parameters.

Theorem 2.1 *Under Assumptions 1 through 3, let* $\{\mathbf{x}^k\}$ *be the iterate sequence from Algorithm 1, with the parameters* $\{\eta_k\}$ *and* $\{\beta_k\}$ *satisfying the condition:*

$$\begin{aligned}&\frac{1}{4}(1-\eta_kL) - \frac{\eta_k}{5m\eta_{k+1}}(1-\beta_k)^2 > 0, \text{ and} \\ &\frac{\eta_k}{2}(2-\eta_kL) - \frac{1}{20\eta_kL^2} + \frac{(1-\beta_k)^2(1+\frac{4\eta_k^2L^2}{m})}{20\eta_{k+1}L^2} \le 0, \ \forall k \ge 0. \quad (2.10)\end{aligned}$$

Let $\{\bar{\mathbf{g}}^k\}$ *be defined in* (1.11). *Then*

$$\sum_{k=0}^{K-1}\left(\frac{\eta_k}{4}(1-\eta_k L)-\frac{\eta_k^2}{5m\eta_{k+1}}(1-\beta_k)^2\right)\mathbb{E}[\|\bar{\mathbf{g}}^k\|^2]$$
$$\leq \Phi(\mathbf{x}^0)-\Phi^*+\frac{\sigma^2}{20m_0\eta_0 L^2}+\sum_{k=0}^{K-1}\frac{\beta_k^2\sigma^2}{10m\eta_{k+1}L^2}. \tag{2.11}$$

Proof From Lemmas 2.1 and 2.2, it follows that

$$\begin{aligned}
&\mathbb{E}\left[\Phi(\mathbf{x}^{k+1})+\frac{\|\mathbf{e}^{k+1}\|^2}{20\eta_{k+1}L^2}-\Phi(\mathbf{x}^k)-\frac{\|\mathbf{e}^k\|^2}{20\eta_k L^2}\right]\\
&\leq \mathbb{E}\left[\frac{\eta_k}{2}(2-\eta_k L)\|\mathbf{e}^k\|^2-\frac{\eta_k}{4}(1-\eta_k L)\|\bar{\mathbf{g}}^k\|^2-\frac{\|\mathbf{e}^k\|^2}{20\eta_k L^2}\right]\\
&\quad+\frac{1}{20\eta_{k+1}L^2}\mathbb{E}\left[\frac{2\beta_k^2\sigma^2}{m}+\frac{4(1-\beta_k)^2\eta_k^2L^2}{m}\|\bar{\mathbf{g}}^k\|^2+(1-\beta_k)^2\left(1+\frac{4\eta_k^2L^2}{m}\right)\|\mathbf{e}^k\|^2\right].
\end{aligned}\tag{2.12}$$

We have from the condition of $\{\beta_k\}$ that the coefficient of the term $\|\mathbf{e}^k\|^2$ on the right-hand side of (2.12) is nonpositive, and thus we obtain from (2.12) that

$$\begin{aligned}
&\mathbb{E}\left[\Phi(\mathbf{x}^{k+1})+\frac{\|\mathbf{e}^{k+1}\|^2}{20\eta_{k+1}L^2}-\Phi(\mathbf{x}^k)-\frac{\|\mathbf{e}^k\|^2}{20\eta_k L^2}\right]\\
&\leq \frac{\beta_k^2\sigma^2}{10m\eta_{k+1}L^2}-\left(\frac{\eta_k}{4}(1-\eta_k L)-\frac{\eta_k^2}{5m\eta_{k+1}}(1-\beta_k)^2\right)\mathbb{E}[\|\bar{\mathbf{g}}^k\|^2].
\end{aligned}$$

Summing up the above inequality from $k=0$ through $K-1$ gives

$$\begin{aligned}
&\mathbb{E}\left[\Phi(\mathbf{x}^K)+\frac{\|\mathbf{e}^K\|^2}{20\eta_K L^2}-\Phi(\mathbf{x}^0)-\frac{\|\mathbf{e}^0\|^2}{20\eta_0 L^2}\right]\\
&\leq \sum_{k=0}^{K-1}\frac{\beta_k^2\sigma^2}{10m\eta_{k+1}L^2}-\sum_{k=0}^{K-1}\left(\frac{\eta_k}{4}(1-\eta_k L)-\frac{\eta_k^2}{5m\eta_{k+1}}(1-\beta_k)^2\right)\mathbb{E}[\|\bar{\mathbf{g}}^k\|^2],
\end{aligned}$$

which implies the inequality in (2.11) by $\mathbb{E}[\|\mathbf{e}^0\|^2]\leq\frac{\sigma^2}{m_0}$. □

Below we specify the choice of parameters and establish complexity results of Algorithm 1.

Theorem 2.2 (convergence rate with varying stepsizes) *Under Assumptions 1 through 3, let* $\{\mathbf{x}^k\}$ *be the iterate sequence from Algorithm 1, with* $m_0=m$ *and the parameters* $\{\eta_k\}$ *and* $\{\beta_k\}$ *set to*

$$\eta_k=\frac{\eta}{L(k+4)^{\frac{1}{3}}},\quad \beta_k=\frac{1+24\eta_k^2L^2-\frac{\eta_{k+1}}{\eta_k}}{1+4\eta_k^2L^2},\ \forall\, k\geq 0, \tag{2.13}$$

where $\eta \le \frac{\sqrt[3]{4}}{8}$ *is a positive number. If* τ *is selected according to* (1.5)*, then*

$$\mathbb{E}[\|\bar{\mathbf{g}}^\tau\|^2] \le \frac{2\left(L(\Phi(\mathbf{x}^0)-\Phi^*)+\frac{\sqrt[3]{4}\sigma^2}{20m\eta}+\frac{\sigma^2}{10m}\left(1152\eta^3(\frac{5}{4})^{\frac{1}{3}}(\log(K+3)-\log 3)+\frac{1}{3\sqrt[3]{9}\eta}\right)\right)}{3\left(\frac{7}{32}-\frac{1}{5}(\frac{5}{4})^{\frac{1}{3}}\right)\eta\left((K+4)^{\frac{2}{3}}-4^{\frac{2}{3}}\right)}. \tag{2.14}$$

Proof Since $\eta \le \frac{\sqrt[3]{4}}{8}$, it holds $\eta_k \le \frac{1}{8L}$. Also, notice $\frac{\eta_k}{\eta_{k+1}} \le (\frac{5}{4})^{\frac{1}{3}}$ or equivalently $\frac{\eta_{k+1}}{\eta_k} \ge (\frac{4}{5})^{\frac{1}{3}}$ for all $k \ge 0$. Hence, it is straightforward to have $\beta_k \in (0,1)$ and thus $(1-\beta_k)^2 \le 1-\beta_k$ for each $k \ge 0$. Now notice $\frac{5m\eta_{k+1}}{4\eta_k}(1-\eta_k L) \ge \frac{5}{4}(\frac{4}{5})^{\frac{1}{3}}\frac{7}{8} > 1 \ge (1-\beta_k)^2$, so the first inequality in (2.10) holds. In addition, to ensure the second inequality in (2.10), it suffices to have $(1-\beta_k)(1+\frac{4\eta_k^2L^2}{m}) \le \frac{\eta_{k+1}}{\eta_k} - 10\eta_k\eta_{k+1}L^2(2-\eta_k L)$. Because $20\eta_k^2L^2 \ge 10\eta_k\eta_{k+1}L^2(2-\eta_k L)$, this inequality is implied by $(1-\beta_k)(1+\frac{4\eta_k^2L^2}{m}) \le \frac{\eta_{k+1}}{\eta_k} - 20\eta_k^2L^2$, which is further implied by the choice of β_k in (2.13). Therefore, both conditions in (2.10) hold, and thus we have (2.11).

Next we bound the coefficients in (2.11). First, from $1-\eta_k L \ge \frac{7}{8}$ and $\frac{\eta_k}{\eta_{k+1}} \le (\frac{5}{4})^{\frac{1}{3}}$ for all k, we have

$$\begin{aligned}\sum_{k=0}^{K-1}\left(\frac{\eta_k}{4}(1-\eta_k L)-\frac{\eta_k^2}{5m\eta_{k+1}}(1-\beta_k)^2\right) &\ge c\sum_{k=0}^{K-1}\eta_k \ge \frac{c\eta}{L}\int_0^K (x+4)^{-\frac{1}{3}}dx \\ &= \frac{3c\eta}{2L}\left((K+4)^{\frac{2}{3}}-4^{\frac{2}{3}}\right),\end{aligned} \tag{2.15}$$

where $c = \frac{7}{32}-\frac{1}{5}(\frac{5}{4})^{\frac{1}{3}} > 0$. Second,

$$\begin{aligned}\sum_{k=0}^{K-1}\frac{\beta_k^2}{\eta_{k+1}} &\le \frac{L}{\eta}\sum_{k=0}^{K-1}(k+5)^{\frac{1}{3}}\left(1+24\eta_k^2L^2-\frac{\eta_{k+1}}{\eta_k}\right)^2 \\ &= \frac{L}{\eta}\sum_{k=0}^{K-1}(k+5)^{\frac{1}{3}}\left(1+24\eta_k^2L^2-\frac{(k+4)^{\frac{1}{3}}}{(k+5)^{\frac{1}{3}}}\right)^2.\end{aligned} \tag{2.16}$$

Note that

$$\begin{aligned}\sum_{k=0}^{K-1}(k+5)^{\frac{1}{3}}\eta_k^4 &= \frac{\eta^4}{L^4}\sum_{k=0}^{K-1}(k+5)^{\frac{1}{3}}(k+4)^{-\frac{4}{3}} \le \frac{\eta^4}{L^4}(\tfrac{5}{4})^{\frac{1}{3}}\sum_{k=0}^{K-1}(k+4)^{-1} \\ &\le \frac{\eta^4}{L^4}(\tfrac{5}{4})^{\frac{1}{3}}(\log(K+3)-\log 3).\end{aligned} \tag{2.17}$$

Furthermore, by $a^3 - b^3 = (a-b)(a^2+ab+b^2)$ for any $a, b \in \mathbb{R}$, we have

$$1 - \frac{(k+4)^{\frac{1}{3}}}{(k+5)^{\frac{1}{3}}} = (k+5)^{-\frac{1}{3}}\left((k+5)^{\frac{1}{3}} - (k+4)^{\frac{1}{3}}\right) = \frac{(k+5)^{-\frac{1}{3}}}{(k+5)^{\frac{2}{3}} + (k+5)^{\frac{1}{3}}(k+4)^{\frac{1}{3}} + (k+4)^{\frac{2}{3}}},$$

and thus

$$\sum_{k=0}^{K-1}(k+5)^{\frac{1}{3}}\left(1 - \frac{(k+4)^{\frac{1}{3}}}{(k+5)^{\frac{1}{3}}}\right)^2 = \sum_{k=0}^{K-1}\frac{(k+5)^{-\frac{1}{3}}}{\left((k+5)^{\frac{2}{3}} + (k+5)^{\frac{1}{3}}(k+4)^{\frac{1}{3}} + (k+4)^{\frac{2}{3}}\right)^2} \leq \frac{1}{9}\sum_{k=0}^{K-1}(k+4)^{-\frac{5}{3}} \leq \frac{1}{6\sqrt[3]{9}}. \tag{2.18}$$

Now applying the inequality $(a+b)^2 \leq 2a^2 + 2b^2$ to (2.16) and then using (2.17) and (2.18), we obtain

$$\sum_{k=0}^{K-1}\frac{\beta_k^2}{\eta_{k+1}} \leq 1152\eta^3 L(\tfrac{5}{4})^{\frac{1}{3}}(\log(K+3) - \log 3) + \frac{L}{3\sqrt[3]{9}\eta}. \tag{2.19}$$

Therefore, plugging (2.15) and (2.19) into (2.11) and by the selection of τ in (1.5), we obtain the desired result. □

Remark 2.1 The result in Theorem 2.2 does not include the noiseless case, i.e., $\sigma = 0$. Nevertheless, if in that case, we can simply choose $\eta_k = \Theta(\frac{1}{L})$ and $\beta_k = 1$ for all $k \geq 0$. This way, Algorithm 1 reduces to the deterministic mirror-prox method, and we can easily obtain $\min_{0\leq k<K}\|\bar{\mathbf{g}}^k\|^2 = O(\frac{1}{K})$ from (2.11).

By Theorem 2.2, we below estimate the complexity result of Algorithm 1 to produce a stochastic ε-stationary solution.

Corollary 2.1 (complexity result with varying stepsizes) *Let $\varepsilon > 0$ be given and suppose $\sigma > 0$. Then under the same conditions of Theorem 2.2, Algorithm 1 can produce a stochastic ε-stationary solution of (1.1) with a total complexity*

$$T_{\text{total}} = mK = O\left(\max\left\{m\varepsilon^{-3}\left(L(\Phi(\mathbf{x}^0) - \Phi^*)\right)^{\frac{3}{2}},\ \varepsilon^{-3}(|\log\varepsilon| + |\log\sigma|)^{\frac{3}{2}}\frac{\sigma^3}{\sqrt{m}}\right\}\right).$$

Proof By Theorem 2.2 with $\eta = \frac{\sqrt[3]{4}}{8}$, we have

$$\mathbb{E}[\|\bar{\mathbf{g}}^\tau\|^2] = O\left(K^{-\frac{2}{3}}\left(L(\Phi(\mathbf{x}^0) - \Phi^*) + \frac{\sigma^2\log K}{m}\right)\right). \tag{2.20}$$

Hence, it suffices to let $K = \Theta\Big(\max\Big\{\varepsilon^{-3}\big(L(\Phi(\mathbf{x}^0) - \Phi^*)\big)^{\frac{3}{2}},\ \varepsilon^{-3}(|\log\varepsilon| + |\log\sigma|)^{\frac{3}{2}}\frac{\sigma^3}{m^{\frac{3}{2}}}\Big\}\Big)$, to have $\mathbb{E}[\|\bar{\mathbf{g}}^\tau\|^2] \le \varepsilon^2$. This completes the proof. □

Remark 2.2 If $m = 1$ or $m = O(1)$ independent of σ, then the total complexity will be

$$T_{\text{total}} = O\left(\max\left\{\varepsilon^{-3}\big(L(\Phi(\mathbf{x}^0) - \Phi^*)\big)^{\frac{3}{2}},\ \varepsilon^{-3}\sigma^3(|\log\varepsilon| + |\log\sigma|)^{\frac{3}{2}}\right\}\right).$$

If $\sigma \ge 1$ is big and can be estimated, we can take $m = \Theta(\sigma^2)$. This way, we obtain the total complexity $O\left(\varepsilon^{-3}\sigma^2\big((|\log\varepsilon| + \log\sigma)^{\frac{3}{2}} + (L(\Phi(\mathbf{x}^0) - \Phi^*))^{\frac{3}{2}}\big)\right)$. This result is near-optimal in the sense that its dependence on ε has the additional logarithmic term $|\log\varepsilon|^{\frac{3}{2}}$ compared to the lower bound result in [3]. In the remaining part of this section, we show that with constant stepsizes, Algorithm 1 can achieve the optimal complexity result $O(\varepsilon^{-3})$.

2.2 Results with Constant Stepsize

In this subsection, we show convergence results of Algorithm 1 by taking constant stepsizes, i.e., $\eta_k = \eta_0, \forall\, k \ge 1$. In order to consider the dependence on the quantities L, $\Phi(\mathbf{x}^0) - \Phi^*$ and σ^2, we give two settings that yield two different results, but each result has the same dependence on the target accuracy ε. The first result is obtained from Theorem 2.1 by taking constant stepsizes.

Theorem 2.3 (convergence rate I with constant stepsizes) *Under Assumptions 1 through 3, let* $\{\mathbf{x}^k\}$ *be the iterate sequence from Algorithm 1, with the parameters* $\{\eta_k\}$ *and* $\{\beta_k\}$ *set to*

$$\eta_k = \frac{\eta}{L\sqrt[3]{K}}, \quad \beta_k = \beta = \frac{4\eta^2/m + 10\eta^2(2 - \eta/K^{\frac{1}{3}})}{K^{\frac{2}{3}} + 4\eta^2/m}, \ \forall\, k \ge 0, \tag{2.21}$$

where $\eta < \frac{\sqrt[3]{K}}{5}$ *is a positive number. If* τ *is selected from* $\{0, 1, \ldots, K-1\}$ *uniformly at random, then*

$$\mathbb{E}[\|\bar{\mathbf{g}}^\tau\|^2] \le \frac{1}{K^{\frac{2}{3}}\left(\frac{1}{4}(1 - \frac{\eta}{\sqrt[3]{K}}) - \frac{1}{5}\right)}\left(\frac{L\big(\Phi(\mathbf{x}^0) - \Phi^*\big)}{\eta} + \frac{\sigma^2\sqrt[3]{K}}{20m_0\eta^2} + \frac{24^2\sigma^2\eta^2}{10m}\right). \tag{2.22}$$

Proof First note $\frac{\eta}{\sqrt[3]{K}} < \frac{1}{5}$ and thus $\beta \in (0, 1)$. Now it is easy to verify by using $(1-\beta)^2 < 1 - \beta$ that the conditions in (2.10) are satisfied. Hence, the result in (2.11) holds.

Second, by the choice of η_k and β_k, we have

$$\sum_{k=0}^{K-1}\left(\tfrac{\eta_k}{4}(1-\eta_k L)-\tfrac{\eta_k^2}{5m\eta_{k+1}}(1-\beta_k)^2\right)\geq\sum_{k=0}^{K-1}\left(\tfrac{\eta}{4L\sqrt[3]{K}}\left(1-\tfrac{\eta}{\sqrt[3]{K}}\right)-\tfrac{\eta}{5L\sqrt[3]{K}}\right)$$
$$=\tfrac{\eta}{L}K^{\frac{2}{3}}\left(\tfrac{1}{4}\left(1-\tfrac{\eta}{\sqrt[3]{K}}\right)-\tfrac{1}{5}\right),\qquad(2.23)$$

and

$$\sum_{k=0}^{K-1}\frac{\beta_k^2\sigma^2}{10m\eta_{k+1}L^2}\leq\sum_{k=0}^{K-1}\frac{\sigma^2\sqrt[3]{K}}{10m\eta L}\left(\frac{4\eta^2+20\eta^2}{K^{\frac{2}{3}}}\right)^2=\frac{24^2\sigma^2\eta^3}{10mL}.\qquad(2.24)$$

Plugging (2.23) and (2.24) into (2.11), we obtain the desired result by the selection of τ in (1.5). □

From (2.22), we see that in order to have the $O(K^{-\frac{2}{3}})$ convergence rate, we need to set $m_0=\Theta(\sqrt[3]{K})$. Next we set m_0 in this way and estimate the complexity result of Algorithm 1 with the constant stepsize.

Corollary 2.2 (complexity result I with constant stepsizes) *Let $\varepsilon>0$ be given. Under Assumptions 1 through 3, let $\{\mathbf{x}^k\}$ be the iterate sequence from Algorithm 1 with $m_0\geq c_0\sqrt[3]{K}$ and the parameters $\{\eta_k\}$ and $\{\beta_k\}$ set to those in (2.21) where $\eta\leq\frac{\sqrt[3]{K}}{10}$. Let τ be selected from $\{0,1,\ldots,K-1\}$ uniformly at random. Then $\mathbf{x}^\tau$ is a stochastic ε-stationary solution of (1.1) if*

$$K=\left\lceil\frac{40^{\frac{3}{2}}\left(\frac{L\left(\Phi(\mathbf{x}^0)-\Phi^*\right)}{\eta}+\frac{\sigma^2}{20c_0\eta^2}+\frac{24^2\sigma^2\eta^2}{10m}\right)^{\frac{3}{2}}}{\varepsilon^3}\right\rceil.\qquad(2.25)$$

Proof When $\eta\leq\frac{\sqrt[3]{K}}{10}$, it holds $\frac{1}{4}\left(1-\frac{\eta}{\sqrt[3]{K}}\right)-\frac{1}{5}\geq\frac{1}{40}$. Hence, (2.22) with $m_0\geq c_0\sqrt[3]{K}$ implies

$$\mathbb{E}[\|\bar{\mathbf{g}}^\tau\|^2]\leq\frac{40}{K^{\frac{2}{3}}}\left(\frac{L\left(\Phi(\mathbf{x}^0)-\Phi^*\right)}{\eta}+\frac{\sigma^2}{20c_0\eta^2}+\frac{24^2\sigma^2\eta^2}{10m}\right),$$

which together with the condition of K in (2.25) gives $\mathbb{E}[\|\bar{\mathbf{g}}^\tau\|^2]\leq\varepsilon^2$. This completes the proof. □

Remark 2.3 Suppose that $\sigma\geq 1$ and can be estimated. Also, assume $L=\Omega(1)$ and $\Phi(\mathbf{x}^0)-\Phi^*=\Omega(1)$. In this case, we let $\eta=\Theta(\sigma^{-\frac{2}{3}}\left(L(\Phi(\mathbf{x}^0)-\Phi^*)\right)^{\frac{1}{3}})$, $c_0=\Theta(\sigma^{\frac{8}{3}})$, and $m=O(1)$ independent of σ. Then from (2.25), we have $K=$

$O\left(\varepsilon^{-3}\sigma L(\Phi(\mathbf{x}^0)-\Phi^*)\right)$. With this choice, the total number of sample gradients will be

$$\begin{aligned} T_{\text{total}} &= m_0 + m(K-1) \\ &= O\left(\varepsilon^{-1}\sigma^3\left(L(\Phi(\mathbf{x}^0)-\Phi^*)\right)^{\frac{1}{3}} + \varepsilon^{-3}\sigma L(\Phi(\mathbf{x}^0)-\Phi^*)\right). \end{aligned} \tag{2.26}$$

The dependence on the pair (ε, σ) matches with the result in [32].

The complexity result given in (2.26) has a low dependence on $(\varepsilon, \sigma, L(\Phi(\mathbf{x}^0) - \Phi^*))$ in the sense that ε^{-3} only multiplies with $\sigma L(\Phi(\mathbf{x}^0)-\Phi^*)$ but not a higher order. However, the drawback is that the initial batch m_0 must be in the order of ε^{-1} to obtain the complexity result $O(\varepsilon^{-3})$. Our second result with constant stepsizes will relax the requirement. We utilize the momentum accumulation in the parameter of (2.9) and give our novel convergence analysis, by introducing the following quantity

$$\Gamma_k = \begin{cases} \prod_{i=0}^{k-1}(1-\beta_i)^2, & \text{if } k \geq 1, \\ 1, & \text{if } k = 0. \end{cases} \tag{2.27}$$

We first give a generic result below under certain conditions on the parameters. Then, we will specify the choice of parameters to satisfy the conditions.

Theorem 2.4 *Under Assumptions 1 through 3, let $\{\mathbf{x}^k\}$ be the iterate sequence from Algorithm 1. Suppose there are constants A and B such that the parameters $\{\eta_k\}$ and $\{\beta_k\}$ satisfying the conditions:*

$$2\eta_k L + \frac{4L^2}{m}\frac{\eta_k}{\Gamma_k}\sum_{j=k+1}^{K-1}\eta_j\Gamma_j \leq 1, \ \forall k = 0, \ldots, K-1, \tag{2.28}$$

$$\sum_{k=1}^{K-1}\eta_k\Gamma_k \leq A, \ \text{and} \ \sum_{k=1}^{K-1}\eta_k\Gamma_k\sum_{j=0}^{k-1}\frac{\beta_j^2}{\Gamma_{j+1}} \leq B, \tag{2.29}$$

where K is the maximum number of iterations in Algorithm 1. Let $\{\bar{\mathbf{g}}^k\}$ be defined in (1.11). Then

$$\sum_{k=0}^{K-1}\eta_k\mathbb{E}\left[\|\bar{\mathbf{g}}^k\|^2\right] \leq 12\left[\Phi(\mathbf{x}^0)-\Phi^*\right] + (8A+6\eta_0)\frac{\sigma^2}{m_0} + 16B\frac{\sigma^2}{m}. \tag{2.30}$$

Proof We begin by taking the total expectation and telescoping the inequality in (2.3) over $k = 0, \ldots, K-1$ to obtain

$$\begin{aligned}\mathbb{E}\big[\Phi(\mathbf{x}^K)\big] - \Phi(\mathbf{x}^0) &\le \sum_{k=0}^{K-1} \frac{\eta_k}{2}\mathbb{E}\big[\|\mathbf{e}^k\|^2\big] - \sum_{k=0}^{K-1} \frac{\eta_k}{2}(1-\eta_k L)\mathbb{E}\big[\|\mathbf{g}^k\|^2\big] \\ &\le \frac{\eta_0}{2}\cdot\frac{\sigma^2}{m_0} + \sum_{k=1}^{K-1} \frac{\eta_k}{2}\mathbb{E}\big[\|\mathbf{e}^k\|^2\big] \\ &\quad - \sum_{k=0}^{K-1} \frac{\eta_k}{2}(1-\eta_k L)\mathbb{E}\big[\|\mathbf{g}^k\|^2\big],\end{aligned}$$

where we have used $\mathbb{E}\big[\|\mathbf{e}^0\|^2\big] \le \frac{\sigma^2}{m_0}$ by Assumption 3. Since $\Phi(\mathbf{x}^K) \ge \Phi^*$ from Assumption 1, the above inequality implies

$$\sum_{k=0}^{K-1} \frac{\eta_k}{2}(1-\eta_k L)\mathbb{E}\big[\|\mathbf{g}^k\|^2\big] \le \Phi(\mathbf{x}^0) - \Phi^* + \frac{\eta_0}{2}\cdot\frac{\sigma^2}{m_0} + \sum_{k=1}^{K-1} \frac{\eta_k}{2}\mathbb{E}\big[\|\mathbf{e}^k\|^2\big]. \tag{2.31}$$

In addition, we divide both sides of (2.9) by Γ_{k+1} and obtain from the definition of Γ_{k+1} in (2.27) that

$$\frac{1}{\Gamma_{k+1}}\mathbb{E}\big[\|\mathbf{e}^{k+1}\|^2\big] \le \frac{1}{\Gamma_k}\mathbb{E}\big[\|\mathbf{e}^k\|^2\big] + \frac{1}{\Gamma_{k+1}}\frac{2\beta_k^2\sigma^2}{m} + \frac{1}{\Gamma_k}\frac{2\eta_k^2 L^2}{m}\mathbb{E}\big[\|\mathbf{g}^k\|^2\big],\ \forall\, k \ge 0.$$

Let $j = 0, \ldots, k-1$ be another index on which the above inequality is telescoped. We obtain

$$\frac{1}{\Gamma_k}\mathbb{E}\big[\|\mathbf{e}^k\|^2\big] \le \mathbb{E}\big[\|\mathbf{e}^0\|^2\big] + \sum_{j=0}^{k-1}\frac{1}{\Gamma_{j+1}}\frac{2\beta_j^2\sigma^2}{m} + \sum_{j=0}^{k-1}\frac{1}{\Gamma_j}\frac{2\eta_j^2 L^2}{m}\mathbb{E}\big[\|\mathbf{g}^j\|^2\big],\ \forall\, k \ge 1.$$

Multiplying Γ_k to both sides of the above inequality and rearranging it gives

$$\mathbb{E}\big[\|\mathbf{e}^k\|^2\big] \le \Gamma_k\left(\frac{\sigma^2}{m_0} + \frac{2\sigma^2}{m}\sum_{j=0}^{k-1}\frac{\beta_j^2}{\Gamma_{j+1}}\right) + \frac{2L^2}{m}\sum_{j=0}^{k-1}\frac{\Gamma_k}{\Gamma_j}\eta_j^2\mathbb{E}\big[\|\mathbf{g}^j\|^2\big],\ \forall\, k \ge 1,$$

where we have used $\mathbb{E}\big[\|\mathbf{e}^0\|^2\big] \le \frac{\sigma^2}{m_0}$ again. Now multiply η_k to the above inequality and sum it up over $k = 1, \ldots, K-1$ to have

$$\begin{aligned}&\sum_{k=1}^{K-1}\eta_k\mathbb{E}\big[\|\mathbf{e}^k\|^2\big] \\ &\quad\le \sigma^2\sum_{k=1}^{K-1}\eta_k\Gamma_k\left(\frac{1}{m_0} + \frac{2}{m}\sum_{j=0}^{k-1}\frac{\beta_j^2}{\Gamma_{j+1}}\right) + \frac{2L^2}{m}\sum_{k=1}^{K-1}\sum_{j=0}^{k-1}\frac{\eta_k\Gamma_k}{\Gamma_j}\eta_j^2\mathbb{E}\big[\|\mathbf{g}^j\|^2\big]\end{aligned}$$

$$= \sigma^2 \sum_{k=1}^{K-1} \eta_k \Gamma_k \left(\frac{1}{m_0} + \frac{2}{m} \sum_{j=0}^{k-1} \frac{\beta_j^2}{\Gamma_{j+1}} \right) + \frac{2L^2}{m} \sum_{j=0}^{K-2} \frac{\eta_j^2}{\Gamma_j} \left(\sum_{k=j+1}^{K-1} \eta_k \Gamma_k \right) \mathbb{E}\big[\|\mathbf{g}^j\|^2\big]$$
$$= \sigma^2 \sum_{k=1}^{K-1} \eta_k \Gamma_k \left(\frac{1}{m_0} + \frac{2}{m} \sum_{j=0}^{k-1} \frac{\beta_j^2}{\Gamma_{j+1}} \right) + \frac{2L^2}{m} \sum_{k=0}^{K-1} \frac{\eta_k^2}{\Gamma_k} \left(\sum_{j=k+1}^{K-1} \eta_j \Gamma_j \right) \mathbb{E}\big[\|\mathbf{g}^k\|^2\big], \tag{2.32}$$

where the first equality follows by swapping summation, and the second equality is obtained by swapping indices and realizing that the coefficient for $\mathbb{E}\big[\|\mathbf{g}^{K-1}\|^2\big]$ is null.

Now we have by substituting (2.32) into (2.31) and rearranging terms that

$$\sum_{k=0}^{K-1} \frac{\eta_k}{2} \left(1 - \eta_k L - \frac{2L^2}{m} \frac{\eta_k}{\Gamma_k} \sum_{j=k+1}^{K-1} \eta_j \Gamma_j \right) \mathbb{E}\big[\|\mathbf{g}^k\|^2\big]$$
$$\leq \Phi(\mathbf{x}^0) - \Phi^* + \frac{\eta_0}{2} \cdot \frac{\sigma^2}{m_0} + \frac{\sigma^2}{2} \sum_{k=1}^{K-1} \eta_k \Gamma_k \left(\frac{1}{m_0} + \frac{2}{m} \sum_{j=0}^{k-1} \frac{\beta_j^2}{\Gamma_{j+1}} \right),$$

which together with the conditions in (2.28) and (2.29) gives the bound for $\mathbf{g}^k$:

$$\sum_{k=0}^{K-1} \eta_k \mathbb{E}\big[\|\mathbf{g}^k\|^2\big] \leq 4\big[\Phi(\mathbf{x}^0) - \Phi^*\big] + 2(A + \eta_0) \frac{\sigma^2}{m_0} + 4B \frac{\sigma^2}{m}. \tag{2.33}$$

Use (2.28) again and substitute (2.33) into (2.32). We obtain the bound for $\mathbf{e}^k$:

$$\sum_{k=0}^{K-1} \eta_k \mathbb{E}\big[\|\mathbf{e}^k\|^2\big] \leq A \frac{\sigma^2}{m_0} + 2B \frac{\sigma^2}{m} + \sum_{k=0}^{K-1} \frac{\eta_k}{2} \mathbb{E}\big[\|\mathbf{g}^k\|^2\big]$$
$$\leq 2\big[\Phi(\mathbf{x}^0) - \Phi^*\big] + (2A + \eta_0) \frac{\sigma^2}{m_0} + 4B \frac{\sigma^2}{m}. \tag{2.34}$$

Finally, we have from (2.4) that $\|\bar{\mathbf{g}}^k\|^2 \leq 2\|\mathbf{g}^k\|^2 + 2\|\mathbf{e}^k\|^2$. Sum up this inequality over $k = 0, \ldots, K-1$ and substitute (2.33) and (2.34) into the summation. We obtain the result in (2.30). □

Below we specify the choice of parameters and establish complexity results of Algorithm 1. The following lemma will be used to show the conditions in (2.28) and (2.29).

Lemma 2.3 *Let*

$$\beta_k = 3\big[(k+3)^{1/3} - (k+2)^{1/3}\big], \quad k \geq 0. \tag{2.35}$$

Then we have

$$\sum_{j=k+1}^{K-1} \frac{\Gamma_j}{\Gamma_k} \leq \frac{1}{2}(k+2)^{2/3} + \frac{1}{6}(k+2)^{1/3} + \frac{1}{36}. \tag{2.36}$$

Proof By the fact $a^3 - b^3 = (a-b)(a^2+ab+b^2)$, we have

$$\beta_k = 3\left[(k+3)^{1/3} - (k+2)^{1/3}\right] = \frac{3}{(k+3)^{2/3} + (k+3)^{1/3}(k+2)^{1/3} + (k+2)^{2/3}}. \tag{2.37}$$

Hence, $\beta_k \in \left[(k+3)^{-2/3}, (k+2)^{-2/3}\right]$ for all $k \geq 0$, and it is a decreasing sequence. In addition, by the definition of Γ_k and β_k, it holds for all $j > k \geq 0$ that

$$\frac{\Gamma_j}{\Gamma_k} = \frac{\prod_{l=0}^{j-1}(1-\beta_l)^2}{\prod_{l=0}^{k-1}(1-\beta_l)^2} = \prod_{l=k}^{j-1}(1-\beta_l)^2 \leq e^{-2\sum_{l=k}^{j-1}\beta_l} = e^{-6\left[(j+2)^{1/3}-(k+2)^{1/3}\right]}, \tag{2.38}$$

where the inequality holds because $0 \leq 1+x \leq e^x, \forall\, x \geq -1$. Therefore, we have that for any $k \geq 0$,

$$\sum_{j=k+1}^{K-1} \frac{\Gamma_j}{\Gamma_k} \leq \sum_{j=k+1}^{K-1} e^{-6\left[(j+2)^{1/3}-(k+2)^{1/3}\right]} = e^{6(k+2)^{1/3}} \sum_{j=k+1}^{K-1} e^{-6(j+2)^{1/3}}. \tag{2.39}$$

Since $e^{-6x^{1/3}}$ is a decreasing function and has an anti-derivative $-\frac{1}{36}e^{-6x^{1/3}}(18x^{2/3} + 6x^{1/3} + 1)$, we have

$$\begin{aligned} \sum_{j=k+1}^{K-1} e^{-6(j+2)^{1/3}} &\leq \int_{k+2}^{K+1} e^{-6x^{1/3}} dx \\ &\leq \frac{1}{36} e^{-6(k+2)^{1/3}} (18(k+2)^{2/3} + 6(k+2)^{1/3} + 1). \end{aligned} \tag{2.40}$$

Substituting (2.40) into (2.39) gives (2.36) and completes the proof. □

Now we are ready to show the second convergence rate result with constant stepsizes.

Theorem 2.5 (convergence rate II with constant stepsizes) *Under Assumptions 1 through 3, let* $\{\mathbf{x}^k\}$ *be the iterate sequence from Algorithm 1 with* $\eta_k = \frac{\eta}{L\sqrt[3]{K}}$ *and* $\{\beta_k\}$ *set by (2.35), where* $\eta \leq \min\{\frac{\sqrt[3]{K}}{4}, \sqrt{\frac{m}{8}}\}$ *is a positive number. If* τ *is selected from* $\{0, 1, \ldots, K-1\}$ *uniformly at random, then*

$$\mathbb{E}[\|\bar{\mathbf{g}}^{\tau}\|^2] \le \frac{1}{K^{\frac{2}{3}}}\left(\frac{12L}{\eta}\big[\Phi(\mathbf{x}^0)-\Phi^*\big]+\Big(2^{-1/3}+\frac{1}{6}2^{1/3}+\frac{7}{9}\Big)\frac{8}{\sqrt[3]{K}}\frac{\sigma^2}{m_0}\right.$$
$$\left.+\frac{32}{(1-2^{-2/3})^2}\frac{\sigma^2}{m}\right). \tag{2.41}$$

Proof We show the desired result by verifying the conditions in Theorem 2.4. First, with $\eta_k = \frac{\eta}{L\sqrt[3]{K}}$, the condition in (2.28) becomes

$$\frac{2\eta}{\sqrt[3]{K}}+\frac{4}{m}\frac{\eta^2}{K^{2/3}}\sum_{j=k+1}^{K-1}\frac{\Gamma_j}{\Gamma_k}\le 1,\quad k=0,\ldots,K-1.$$

Notice that when $k = K-1$ the summation above is null. Hence, by (2.36), it suffices to require

$$\frac{2\eta}{\sqrt[3]{K}}+\frac{4}{m}\frac{\eta^2}{K^{2/3}}\left(\frac{1}{2}K^{2/3}+\frac{1}{6}K^{1/3}+\frac{1}{36}\right)\le 1,$$

which is guaranteed when $\eta \le \min\{\frac{\sqrt[3]{K}}{4}, \sqrt{\frac{m}{8}}\}$ and $K \ge 1$. Therefore, the condition in (2.28) holds.

Secondly, by letting $k = 0$ in (2.36) and recalling $\Gamma_0 = 1$, we have $\sum_{k=1}^{K-1}\eta_k\Gamma_k \le \big(\frac{1}{2}2^{2/3}+\frac{1}{6}2^{1/3}+\frac{1}{36}\big)\frac{\eta}{L\sqrt[3]{K}}$. Hence, the first condition in (2.29) holds with $A = \big(2^{-1/3}+\frac{1}{6}2^{1/3}+\frac{1}{36}\big)\frac{\eta}{L\sqrt[3]{K}}$. Finally, notice

$$\begin{aligned}
&\sum_{k=1}^{K-1}\eta_k\Gamma_k\sum_{j=0}^{k-1}\frac{\beta_j^2}{\Gamma_{j+1}}\\
&=\sum_{j=0}^{K-2}\frac{\beta_j^2}{(1-\beta_j)^2}\sum_{k=j+1}^{K-1}\eta_k\frac{\Gamma_k}{\Gamma_j}\\
&\le\frac{\eta}{L\sqrt[3]{K}}\sum_{j=0}^{K-2}\frac{\beta_j^2}{(1-\beta_0)^2}\left(\frac{1}{2}(j+2)^{2/3}+\frac{1}{6}(j+2)^{1/3}+\frac{1}{36}\right)\\
&\le\frac{\eta}{(1-\beta_0)^2L\sqrt[3]{K}}\sum_{j=0}^{K-2}\left(\frac{1}{2}(j+2)^{-2/3}+\frac{1}{6}(j+2)^{-1}+\frac{1}{36}(j+2)^{-4/3}\right)\\
&\le\frac{\eta}{(1-\beta_0)^2L\sqrt[3]{K}}\left(\frac{3}{2}(K^{1/3}-1)+\frac{1}{6}\log K+\frac{1}{12}(1-K^{-1/3})\right)\\
&\le\frac{2\eta}{(1-2^{-2/3})^2L},
\end{aligned}\tag{2.42}$$

where the first inequality follows from (2.36), the decreasing monotonicity of β_k, and the setting of η_k, the second inequality holds by $\beta_j \le (j+2)^{-2/3}$, and the last

inequality is obtained by $\beta_0 \leq 2^{-2/3}$ and using the fact $3x^{1/3} > \log x, \forall\, x > 0$. Thus, the second condition in (2.29) holds with $B = \frac{2\eta}{(1-2^{-2/3})^2 L}$. Therefore, (2.41) follows from (2.30) and the choice of τ by uniformly random selection. □

From Theorem 2.5, we can immediately obtain the next complexity result of Algorithm 1 with the constant stepsize.

Corollary 2.3 (complexity result II with constant stepsizes) *Let $\varepsilon > 0$ be given. Under Assumptions 1 through 3, let $\{\mathbf{x}^k\}$ be the iterate sequence from Algorithm 1 with $\eta_k = \frac{\eta}{L\sqrt[3]{K}}$ and $\{\beta_k\}$ set by (2.35), where $\eta \leq \min\{\frac{\sqrt[3]{K}}{4}, \sqrt{\frac{m}{8}}\}$ is a positive number. Let τ be selected from $\{0, 1, \ldots, K-1\}$ uniformly at random. Then $\mathbf{x}^\tau$ is a stochastic ε-stationary solution of (1.1) if*

$$K = \left\lceil \frac{\left(\frac{12L}{\eta}\left[\Phi(\mathbf{x}^0) - \Phi^*\right] + \left(2^{-1/3} + \frac{1}{6}2^{1/3} + \frac{7}{9}\right)\frac{8\sigma^2}{m_0} + \frac{32}{(1-2^{-2/3})^2}\frac{\sigma^2}{m}\right)^{3/2}}{\varepsilon^3} \right\rceil. \tag{2.43}$$

Remark 2.4 We need $\eta \leq \min\{\frac{\sqrt[3]{K}}{4}, \sqrt{\frac{m}{8}}\}$, which is true as long as $\eta = \sqrt{\frac{m}{8}}$ and $m \leq \frac{K^{2/3}}{2}$. Then we have from (2.43) that $K = O(\varepsilon^{-3}(\frac{L}{m^{1/2}}[\Phi(\mathbf{x}^0) - \Phi^*] + \frac{\sigma^2}{m_0} + \frac{\sigma^2}{m})^{3/2})$ by ignoring the dependence on absolute constants; the total sample complexity is $m_0 + m(K-1) = O(\varepsilon^{-3}(L[\Phi(\mathbf{x}^0) - \Phi^*]m^{1/6} + \frac{\sigma^2}{m^{1/3}})^{3/2})$ if we let $m_0 = m$. Let $m_0 = m = O(1)$, the total sample complexity $m_0 + m(K-1) = O(\varepsilon^{-3})$ matches with the lower bound in [3]. However, the dependence on $(L(\Phi(\mathbf{x}^0) - \Phi^*))^{3/2}$ will be not as good as the result in (2.26). Let $m_0 = m = \Theta(\frac{\sigma^4}{L^2(\Phi(\mathbf{x}^0)-\Phi^*)^2})$, total sample complexity is $m_0 + m(K-1) = O(\varepsilon^{-3}\sigma L(\Phi(\mathbf{x}^0) - \Phi^*))$.

3 Numerical Experiments

In this section, we test Algorithm 1, named as PStorm, on solving three problems. The first problem is the nonnegative principal component analysis (NPCA) [25], and the other two are on training neural networks. We compare PStorm to the vanilla proximal SGD, Spiderboost [33], and Hybrid-SGD [32]. Spiderboost and Hybrid-SGD both achieve optimal complexity results, and the vanilla proximal SGD is used as a baseline for the comparison. For NPCA, all methods were implemented in MATLAB 2021a on a quad-core iMAC with 40 GB memory, and for training neural networks, all methods were implemented by using PyTorch on a Dell workstation with 32 CPU cores, 2 GPUs, and 64 GB memory.

3.1 Nonnegative Principal Component Analysis (NPCA)

In this subsection, we compare the four methods on solving the NPCA problem:

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\text{Max}}\ \frac{1}{2}\mathbb{E}_{\mathbf{z}}[\mathbf{x}^\top(\mathbf{z}\mathbf{z}^\top)\mathbf{x}],\ \text{s.t.}\ \|\mathbf{x}\| \leq 1, \mathbf{x} \geq \mathbf{0}, \tag{3.1}$$

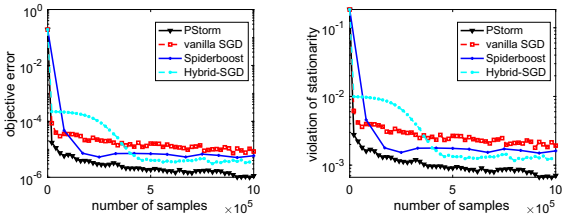

Fig. 1 Objective error and the violation of stationarity by PStorm, the vanilla SGD, Spiderboost, and Hybrid-SGD on solving (3.1) with randomly generated dataset

where $\mathbf{z} \in \mathbb{R}^n$ represents a random data point following a certain distribution, and $\mathbb{E}_{\mathbf{z}}$ takes expectation about $\mathbf{z}$. The problem (3.1) can be formulated into the form of (1.1), by negating the objective and adding an indicator function of the constraint. Two datasets were used in this test. The first one takes $\mathbf{z} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$ where $\mathbf{w} \sim \mathcal{N}(\mathbf{1}, \mathbf{I})$, and we solved a stochastic problem; for the second one, we used the normalized training and testing datasets of `realsim` from LIBSVM [4], and we solved a deterministic finite-sum problem. For both datasets, each sample function in the objective of (3.1) is 1-smooth, and thus we used the Lipschitz constant $L = 1$ for all methods.

Random dataset: For the randomly generated dataset, we set the dimension $n = 100$ and the minibatch size to $m = 10$ for PStorm, the vanilla proximal SGD, and the Hybrid-SGD. For the Spiderboost, we set $\varepsilon = 5 \times 10^{-3}$, and for each iteration k, it accessed $q = \varepsilon^{-1}$ data samples if $\mathrm{mod}(k, q) \neq 0$ and ε^{-2} data samples otherwise. Each method could access at most 10^6 data samples. The stepsize of PStorm was set according to (2.13) with η tuned from $\{0.1, 0.2, 0.5, 1\}$, out of which $\eta = 0.1$ turned out the best. The stepsize of the vanilla proximal SGD was set to $\frac{\eta}{\sqrt{k+1}}$ for each iteration $k \geq 0$ with η tuned from $\{0.1, 0.2, 0.5, 1\}$, out of which $\eta = 0.5$ turned out the best. The stepsize of Spiderboost was set to $\eta = 0.5$. The Hybrid-SGD has a few more parameters to tune. As suggested by [32, Theorem 4] and also its numerical experiments, we set γ_k, β_k, η_k, and the initial batch size to

$$\gamma_k \equiv \gamma = \frac{3c_0 m^{\frac{3}{4}}}{\sqrt{13 m_0 (K+1)^{\frac{1}{4}}}}, \ \beta_k \equiv \beta = 1 - \frac{\sqrt{m}}{\sqrt{m_0 K}}, \ \eta_k \equiv \eta = \frac{2}{L(3+\gamma)},$$
$$m_0 = \frac{c_1^2}{\lceil m(K+1)^{\frac{1}{3}} \rceil}, \tag{3.2}$$

where K is the maximum number of iterations. We tuned c_0 to 10 and c_1 to 5.

To evaluate the performance of the tested methods, we randomly generated 10^7 data samples following the same distribution as we described above, and at the iterates of the methods, we computed their violation of stationarity of the sample-approximation problem. Since the compared methods have different learning rate, to make a fair comparison, we measured the *violation of stationarity* at $\mathbf{x}$ by $\|P(\mathbf{x}, \nabla F, 1)\|$, where P is the proximal mapping defined in Definition 1.1, and F is the sample-approximated objective. Also, to obtain the "optimal" objective value, we ran the projected gradient method to 1,000 iterations on the deterministic sample-approximation problem. The results in terms of the number of samples are plotted in Fig. 1, which clearly shows the superiority of PStorm over all the other three methods.

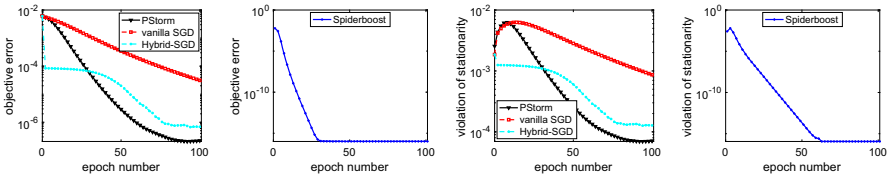

Fig. 2 Objective error and the violation of stationarity by PStorm, the vanilla SGD, Spiderboost, and Hybrid-SGD on solving (3.1) with realsim dataset

realsim dataset: The `realsim` dataset has $N = 72,309$ samples in total. We set the minibatch size to $m = 64$ for PStorm, the vanilla proximal SGD, and the Hybrid-SGD. For each iteration k of the Spiderboost, we set $|B_k| = q = \lceil \sqrt{N} \rceil = 269$ in (1.6), as suggested by [33, Theorem 3]. The stepsizes of PStorm and the vanilla proximal SGD were tuned in the same way as above, and the best η was 0.2 for the former and 0.5 for the latter. The stepsize for Spiderboost was still set to 0.5 as the smoothness constant is $L = 1$. For Hybrid-SGD, we set its parameters to

$$\gamma_k \equiv \gamma = 0.95,\ \beta_k \equiv \beta = 1 - \frac{\sqrt{m}}{\sqrt{m_0 K}},\ \eta_k \equiv \eta = \frac{2}{L(3+\gamma)},$$
$$m_0 = \max\left\{N, \frac{c_1^2}{\lceil m(K+1)^{\frac{1}{3}} \rceil}\right\},$$

where K is the maximum number of iterations and c_1 was tuned to 15. Notice that different from (3.2), here we simply fix $\gamma = 0.95$. This choice of γ was also adopted in [32], and it turned out that this setting resulted in the best performance of Hybrid-SGD for this test.

We ran each method to 100 epochs, where one epoch is equivalent to one pass of all data samples. The results in terms of epoch number are shown in Fig. 2, where the violation of stationary was again measured by $\|P(\mathbf{x}, \nabla F, 1)\|$ and the "optimal" objective value was given by running the projected gradient method to 1,000 iterations. For this test, we found that Spiderboost converges extremely fast and gave much smaller errors than those by other methods, and thus we plot the results by Spiderboost in separate figures. PStorm performed better than the vanilla proximal SGD and the Hybrid-SGD. We also tested the methods on the datasets `w8a` and `gisette` from LIBSVM. Their comparison performance was similar to that on `realsim`.

3.2 Regularized Feedforward Fully-connected Neural Network

In this subsection, we compare different methods on solving an ℓ_1-regularized 3-layer feedforward fully-connected neural network, formulated as

$$\min_{\boldsymbol{\theta}} \frac{1}{N}\sum_{i=1}^{N} \ell\Big(\text{softmax}\big(\mathbf{W}_3\sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}_i))\big), y_i\Big) + \lambda\big(\|\mathbf{W}_1\|_1 + \|\mathbf{W}_2\|_1 + \|\mathbf{W}_3\|_1\big). \tag{3.3}$$

Here $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ is a c-class training data set with $y_i \in \{1, \ldots, c\}$ for each i, $\boldsymbol{\theta} := (\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3)$ contains the parameters of the neural network, $\sigma(\cdot)$ is an activation function, ℓ denotes a loss function, $\text{softmax}(\mathbf{z}) := \frac{1}{\sum_{j=1}^c e^{z_j}}[e^{z_1}; \ldots; e^{z_c}] \in \mathbb{R}^c$, $\forall\, \mathbf{z} \in \mathbb{R}^c$, and $\lambda \geq 0$ is a regularization parameter to trade off the loss and sparsity.

In the test, we used the MNIST dataset [16] of hand-written-digit images. The training set has 60,000 images, and the testing set has 10,000 images. Each image was originally 28×28 and vectorized into a vector of dimension 784. We set $\mathbf{W}_1 \in \mathbb{R}^{784\times 120}$, $\mathbf{W}_2 \in \mathbb{R}^{120\times 84}$, and $\mathbf{W}_3 \in \mathbb{R}^{84\times 10}$, whose initial values were set to the default ones in `libtorch`, a C++ distribution of PyTorch. We used the hyperbolic tangent activation function $\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ and the cross-entropy $\ell(\mathbf{q}, y_i) = -\log q_{y_i}$ for any distribution $\mathbf{q} \in \mathbb{R}^c$.

The parameters of PStorm were set according to (2.13) with $L = 1$ and $\eta = \frac{\sqrt[3]{4}}{8} \approx 0.198$. Notice that the gradient of the loss function in (3.3) is not uniformly Lipschitz continuous, and its Lipschitz constant depends on $\boldsymbol{\theta}$. More specifically, the gradient is Lipschitz continuous over any bounded set of $\boldsymbol{\theta}$. Nevertheless, PStorm with this parameter setting performed well. The learning rate of the vanilla SGD was set to $\eta_k = \frac{\eta}{\sqrt{k+1}}, \forall\, k \geq 0$ with $\eta = \frac{\sqrt[3]{4}}{8}$. We also tried $\eta = 0.5$, and it turned out that the performance of the vanilla SGD was not as well as that with $\eta = \frac{\sqrt[3]{4}}{8}$ when $\lambda > 0$ in (3.3). For Spiderboost, we set $q = \lceil\sqrt{60000}\rceil = 245$ in (1.6) as specified by [33, Theorem 2] and its learning rate $\eta = 0.02$ in (1.7). We also tried $\eta = 0.1$ and $\eta = 0.01$. It turned out that Spiderboost could diverge with $\eta = 0.1$ and converged too slowly with $\eta = 0.01$. For Hybrid-SGD, we fixed its parameter $\gamma = 0.95$ as suggested in the numerical experiments of [32], and we set $\beta_k = \beta = 1 - \frac{1}{\sqrt{K+1}}, \forall k \geq 0$ in (1.8), where K is the maximum number of iterations. Its learning rate was set to $\eta = \frac{2}{4+L\gamma}$. Then we chose the initial mini-batch size m_0 from $\{256, 2560, 30000, 60000\}$ and L from $\{5, 10, 50, 100\}$. The best results were reported.

We ran each method to 100 epochs. Mini-batch size was set to 32 for PStorm, the vanilla SGD, and Hybrid-SGD. Again, to make a fair comparison, we measured the *violation of stationarity* at $\boldsymbol{\theta}$ by $\|P(\boldsymbol{\theta}, \nabla F, 1)\|$, where P is the proximal mapping defined in Definition 1.1, and F is the smooth term in the objective of (3.3). Table 2 and Fig. 3 show the results by the compared methods. Each result in the table is the average of those at the last five epochs. For Hybrid-SGD, the best results were obtained with $(m_0, L) = (60000, 50)$ when $\lambda = 0$ and with $(m_0, L) = (60000, 100)$ when $\lambda > 0$. From the results, we see that PStorm and Hybrid-SGD give similar training loss and testing accuracies while the vanilla SGD and Spiderboost yield higher loss and lower accuracies. The lower accuracies by Spiderboost may be caused by its larger batch size that is required in [33], and the lower accuracies by the vanilla SGD are because of its slower convergence. In addition, PStorm produced sparser solutions than those by other methods in all regularized cases. In terms of the violation of stationarity, the solutions by PStorm have better quality than those by other methods. Furthermore, we notice that the model (3.3) trained by PStorm with $\lambda = 5 \times 10^{-4}$ is much sparser than that without the ℓ_1 regularizer, but the sparse model gives just slightly lower testing

Table 2 Results by the proposed method PStorm, the vanilla SGD, Hybrid-SGD, and Spiderboost on training the model (3.3)

Method	PStorm				Vanilla SGD				Spiderboost				Hybrid-SGD			
λ	Train	Test	Grad	Density	Train	Test	Grad	Density	Train	Test	Grad	Density	Train	Test	Grad	Density
0.00	3.61e–3	**98.01**	**3.45e–3**	100	6.91e–2	97.09	3.42e–2	100	4.24e–2	97.41	1.57e–2	100	**1.50e–3**	97.11	3.64e–3	100
2e–4	4.38e–2	97.60	**1.60e–2**	**14.06**	1.08e–1	96.62	5.77e–2	99.47	8.70e–2	97.24	1.87e–2	27.17	**4.08e–2**	**97.78**	9.53e–2	28.29
5e–4	8.86e–2	**97.12**	**1.94e–2**	**6.16**	1.69e–1	95.54	5.96e–2	92.86	1.41e–1	96.16	2.18e–2	10.62	**8.34e–2**	**97.12**	1.11e–1	12.69

The first three methods use mini-batch $m = 32$. Each method runs to 100 epochs. "train" is for training loss; "test" is for testing accuracy; "grad" is for the violation of stationarity; "density" is for the percentage of nonzeros in the solution. The best results for "test," "grad," and "density" are highlighted in bold

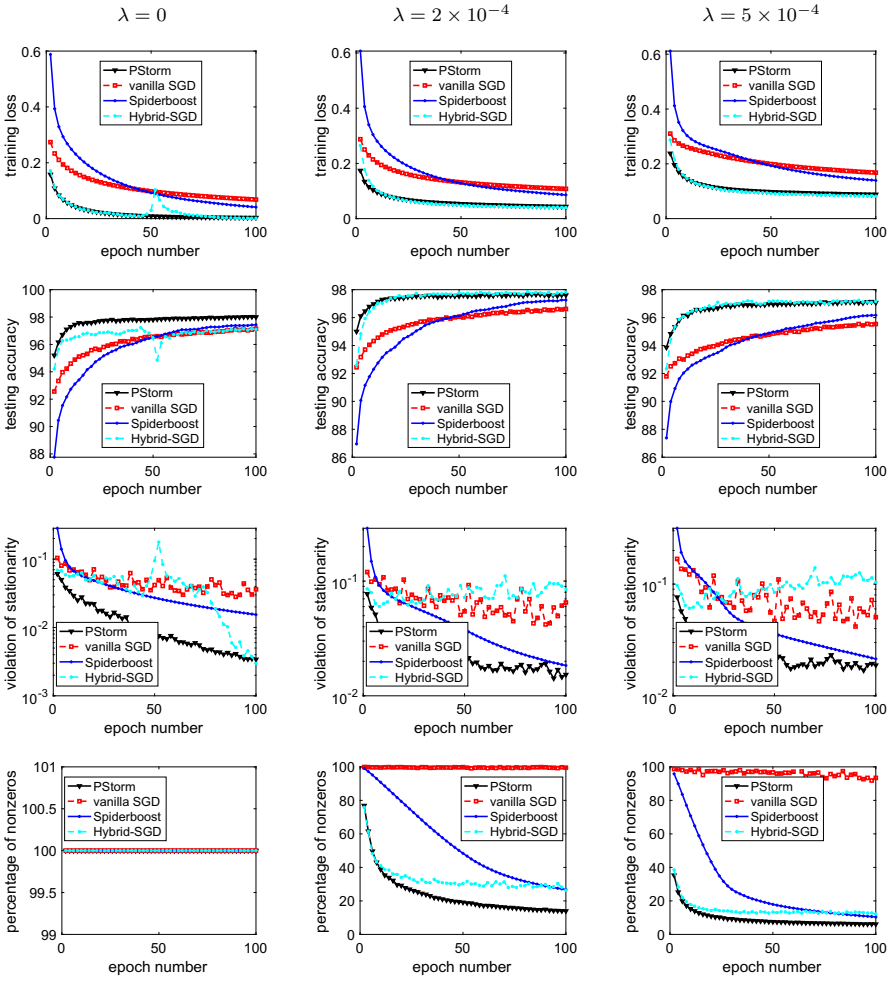

Fig. 3 Results in terms of epoch by the proposed method PStorm, the vanilla SGD, Hybrid-SGD, and Spiderboost on training the model (3.3). The first three methods use mini-batch $m = 32$

accuracy than the dense one. This is important because a sparser model would reduce the inference time when the model is deployed to predict new data.

3.3 Regularized Convolutional Neural Network

In this subsection, we compare different methods on solving an ℓ_1-regularized convolutional neural network, formulated as

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \ell\Big(\text{softmax}(\phi_{\boldsymbol{\theta}}(\mathbf{x}_i)),\, y_i\Big) + \lambda \|\boldsymbol{\theta}\|_1. \tag{3.4}$$

Table 3 Results in terms of epoch by the proposed method PStorm, the vanilla SGD, Hybrid-SGD, and Spiderboost on training the model (3.4)

Method	PStorm				vanilla SGD				Spiderboost				Hybrid-SGD			
λ	train	test	grad	density	train	test	grad	density	train	test	grad	density	train	test	grad	density
0.0	**2.30e–2**	**89.74**	**0.10**	100	2.45e–1	85.61	0.76	100	1.86	36.63	0.12	100	5.26e–2	88.17	0.19	100
2e–4	**7.61e–1**	**89.40**	4.17	**44.91**	9.42e–1	88.76	2.78	89.64	2.93	20.43	**0.89**	53.79	8.15e–1	88.03	1.68	72.78
5e–4	**1.15**	**88.53**	5.94	**19.87**	2.15	86.62	5.55	40.64	4.69	18.62	**0.81**	32.21	1.75	86.71	6.09	60.78

The first three methods use mini-batch $m = 100$. Each method runs to 200 epochs. "train" is for training loss; "test" is for testing accuracy; "grad" is for the violation of stationarity; "density" is for the percentage of nonzeros in the solution. The best results for "test," "grad," and "density" are highlighted in bold

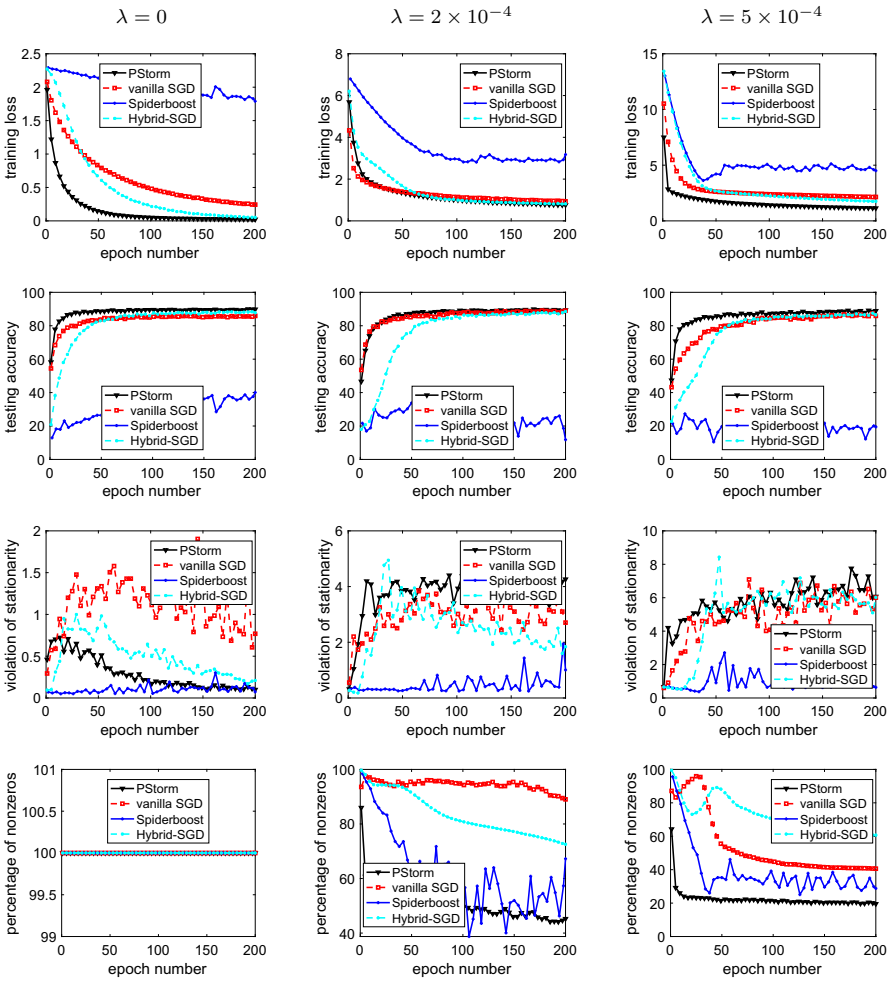

Fig. 4 Results in terms of epoch by the proposed method PStorm, the vanilla SGD, Hybrid-SGD, and Spiderboost on training the model (3.4). The first three methods use mini-batch $m = 100$

Similar to (3.3), $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ is a c-class training data set with $y_i \in \{1, \ldots, c\}$ for each i, $\boldsymbol{\theta}$ contains all parameters of the neural network, ℓ denotes a loss function, $\phi_{\boldsymbol{\theta}}$ represents the nonlinear transformation via the neural network parameterized by $\boldsymbol{\theta}$, and $\lambda \geq 0$ is a regularization parameter to trade off the loss and sparsity. In the test, we used the Cifar10 dataset [15] that has 50,000 training images and 10,000 testing images. In addition, we set ℓ to the cross-entropy loss and $\phi_{\boldsymbol{\theta}}$ to the all convolutional neural network (AllCNN) in [29] without data augmentation. The AllCNN has nine convolutional layers with ReLU activation.

We ran each method to 200 epochs. Mini-batch size was set to 100 for PStorm, the vanilla SGD, and Hybrid-SGD. The stepsizes of PStorm and the vanilla proximal SGD were tuned in the same way as in Sect. 3.1. For Spiderboost, we set $q = \lceil\sqrt{50000}\rceil =$

224 in (1.6), and its learning rate η in (1.7) was tuned by picking the best one from $\{0.01, 0.1, 0.5\}$. For Hybrid-SGD, we set its parameters in a way similar to that in Sect. 3.2 but chose the best pair of (L, m_0) from $\{1, 10, 100\} \times \{10^2, 10^3, 10^4\}$. Because the loss for AllCNN is not differentiable, we did not adopt the variance reduction for PStorm, Hybrid-SGD, and Spiderboost, i.e., we changed $\mathbf{u}^{k+1} = \mathbf{v}^{k+1}, \forall\, k$ in (1.4) and $\mathbf{x}^{k-1} = \mathbf{x}^k$ in (1.6) and (1.8). Results produced by the four methods are shown in Table 3 and Fig. 4. Again, each result in the table is the average of those at the last five epochs. From the results, we see that PStorm and Hybrid-SGD give similar training loss and testing accuracies. PStorm is slightly better than Hybrid-SGD, and the advantage of the former is more significant when $\lambda = 5 \times 10^{-4}$. Spiderboost can give small violation of stationarity, but it tended to have significantly higher loss and lower accuracies. This is possibly because Spiderboost used larger batch size.

4 Conclusions

We have presented a momentum-based variance-reduced mirror-prox stochastic gradient method for solving nonconvex nonsmooth problems, where the nonsmooth term is assumed to be closed convex. The method, named PStorm, requires only one data sample for each update. It is the first $O(1)$-sample-based method that achieves the optimal complexity result $O(\varepsilon^{-3})$ under a mean-squared smoothness condition for solving nonconvex nonsmooth problems. The $O(1)$-sample update is important in machine learning because small-batch training can lead to good generalization. On training sparse regularized neural networks, PStorm can perform better than two other optimal stochastic methods and consistently better than the vanilla stochastic gradient method.

Acknowledgements We thank two anonymous referees for their constructive comments and suggestions to improve the quality and contributions of the paper. This work is partly supported by NSF grants DMS-2053493 and DMS-2208394 and RPI-IBM AIRC.

References

1. Allen-Zhu, Z.: Natasha 2: Faster non-convex optimization than SGD. In: Advances in Neural Information Processing Systems, pp. 2675–2686 (2018)
2. Allen-Zhu, Z., Hazan, E.: Variance reduction for faster non-convex optimization. In: International Conference on Machine Learning, pp. 699–707 (2016)
3. Arjevani, Y., Carmon, Y., Duchi, J.C., Foster, D.J., Srebro, N., Woodworth, B.: Lower bounds for non-convex stochastic optimization. arXiv:1912.02365 (2019)
4. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. (TIST) **2**(3), 1–27 (2011)
5. Chen, X., Liu, S., Sun, R., Hong, M.: On the convergence of a class of adam-type algorithms for non-convex optimization. In: International Conference on Learning Representations (2018)
6. Cutkosky, A., Orabona, F.: Momentum-based variance reduction in non-convex SGD. In: Advances in Neural Information Processing Systems, pp. 32 (2019)
7. Davis, D., Drusvyatskiy, D.: Stochastic model-based minimization of weakly convex functions. SIAM J. Optim. **29**(1), 207–239 (2019)
8. Davis, D., Drusvyatskiy, D., Kakade, S., Lee, J.D.: Stochastic subgradient method converges on tame functions. Found. Comput. Math. **20**(1), 119–154 (2020)

9. Fang, C., Li, C.J., Lin, Z., Zhang, T.: Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In: Advances in Neural Information Processing Systems, pp. 689–699 (2018)
10. Ghadimi, S., Lan, G.: Stochastic first and zeroth-order methods for nonconvex stochastic programming. SIAM J. Optim. **23**(4), 2341–2368 (2013)
11. Ghadimi, S., Lan, G.: Accelerated gradient methods for nonconvex nonlinear and stochastic programming. Math. Program. **156**(1–2), 59–99 (2016)
12. Ghadimi, S., Lan, G., Zhang, H.: Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. Math. Program. **155**(1–2), 267–305 (2016)
13. Huo, Z., Huang, H.: Asynchronous stochastic gradient descent with variance reduction for non-convex optimization. arXiv:1604.03584 (2016)
14. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: generalization gap and sharp minima. arXiv:1609.04836 (2016)
15. Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical Report, University of Toronto, Toronto, ON (2009)
16. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
17. Lei, L., Ju, C., Chen, J., Jordan, M.I.: Non-convex finite-sum optimization via scsg methods. In: Advances in Neural Information Processing Systems, pp. 2348–2358 (2017)
18. Liu, B., Wang, M., Foroosh, H., Tappen, M., Pensky, M.: Sparse convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 806–814 (2015)
19. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online learning for matrix factorization and sparse coding. J. Mach. Learn. Res. **11**(Jan), 19–60 (2010)
20. Masters, D., Luschi, C.: Revisiting small batch training for deep neural networks. arXiv:1804.07612 (2018)
21. Mitliagkas, I., Caramanis, C., Jain, P.: Memory limited, streaming PCA. In: Advances in Neural Information Processing Systems, pp. 2886–2894 (2013)
22. Nguyen, L.M., Liu, J., Scheinberg, K., Takáč, M.: Sarah: a novel method for machine learning problems using stochastic recursive gradient. In: Proceedings of the 34th International Conference on Machine Learning, Vol. 70, pp. 2613–2621. JMLR. org (2017)
23. Pham, N.H., Nguyen, L.M., Phan, D.T., Tran-Dinh, Q.: ProxSARAH: an efficient algorithmic framework for stochastic composite nonconvex optimization. J. Mach. Learn. Res. **21**(110), 1–48 (2020)
24. Reddi, S.J., Hefny, A., Sra, S., Póczos, B., Smola, A.: Stochastic variance reduction for nonconvex optimization. In: International Conference on Machine Learning, pp. 314–323 (2016)
25. Reddi, S.J., Sra, S., Poczos, B., Smola, A.J.: Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In: Proceedings of the 30th International Conference on Neural Information Processing Systems, pp. 1153–1161 (2016)
26. Robbins, H., Monro, S.: A stochastic approximation method. Ann. Math. Stat. **22**, 400–407 (1951)
27. Scardapane, S., Comminiello, D., Hussain, A., Uncini, A.: Group sparse regularization for deep neural networks. Neurocomputing **241**, 81–89 (2017)
28. Shi, J.V., Xu, Y., Baraniuk, R.G.: Sparse bilinear logistic regression. arXiv:1404.4104 (2014)
29. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: The all convolutional net. arXiv:1412.6806 (2014)
30. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, New York (2018)
31. Tran Dinh, Q., Liu, D., Nguyen, L.: Hybrid variance-reduced SGD algorithms for minimax problems with nonconvex-linear function. Adv. Neural. Inf. Process. Syst. **33**, 11096–11107 (2020)
32. Tran-Dinh, Q., Pham, N.H., Phan, D.T., Nguyen, L.M.: A hybrid stochastic optimization framework for composite nonconvex optimization. Math. Program. **191**(2), 1005–1071 (2022)
33. Wang, Z., Ji, K., Zhou, Y., Liang, Y., Tarokh, V.: Spiderboost and momentum: faster variance reduction algorithms. In: Advances in Neural Information Processing Systems, pp. 32 (2019)
34. Wei, C., Lee, J.D., Liu, Q., Ma, T.: Regularization matters: generalization and optimization of neural nets vs their induced kernel. In: Advances in Neural Information Processing Systems, pp. 9709–9721 (2019)
35. Xu, Y., Xu, Y.: Katyusha acceleration for convex finite-sum compositional optimization. Informs J. Optim. **3**(4), 418–443 (2021)
36. Xu, Y., Xu, Y., Yan, Y., Sutcher-Shepard, C., Grinberg, L., Chen, J.: Parallel and distributed asynchronous adaptive stochastic gradient methods. arXiv:2002.09095 (2020)

37. Xu, Y., Yin, W.: Block stochastic gradient iteration for convex and nonconvex optimization. SIAM J. Optim. **25**(3), 1686–1716 (2015)
38. Zhang, J., Xiao, L.: A stochastic composite gradient method with incremental variance reduction. In: Advances in Neural Information Processing Systems, pp. 32 (2019)
39. Zhang, J., Xiao, L.: Stochastic variance-reduced prox-linear algorithms for nonconvex composite optimization. Math. Program. **195**, 1–43 (2021)
40. Zhao, R., Tan, V.Y.: Online nonnegative matrix factorization with outliers. IEEE Trans. Signal Process. **65**(3), 555–570 (2016)
41. Zhou, D., Tang, Y., Yang, Z., Cao, Y., Gu, Q.: On the convergence of adaptive gradient methods for nonconvex optimization. arXiv:1808.05671 (2018)