ϵ-Approximation of Adaptive Leaning Rate Optimization Algorithms for Constrained Nonconvex Stochastic Optimization

Hideaki Iiduka

***Abstract*—This paper considers constrained nonconvex stochastic finite-sum and online optimization in deep neural networks. Adaptive-learning-rate optimization algorithms, such as Adam, AMSGrad, and their variants, have widely been used for these optimizations because they are powerful and useful in theory and practice. Here, it is shown that the adaptive-learning-rate optimization algorithms are ϵ-approximations for these optimizations. We provide the learning rates, mini-batch sizes, number of iterations, and stochastic gradient complexity with which to achieve ϵ-approximations of the algorithms.**

***Index Terms*—Adaptive-learning-rate optimization algorithm, deep neural network, ϵ-approximation, nonconvex stochastic optimization, stochastic gradient complexity.**

I. Introduction

ADAPTIVE-learning-rate optimization algorithms (ALROAs) have been used to train deep neural networks [1], [2], [3], [4], [5], [6], [7], [8]. These algorithms can find suitable parameters for deep neural network models by using nonconvex optimization since they can adapt the learning rates of all model parameters. They are based on the stochastic gradient descent (SGD) algorithm [9], [10], [11], which is the simplest algorithm for solving nonconvex optimization problems in deep neural networks.

The adaptive gradient (AdaGrad) algorithm [12] is based on SGD, and the root mean square propagation (RMSProp) algorithm [13, Chapter 8] is a modification of AdaGrad. The adaptive moment estimation (Adam) algorithm [14] is widely used to train deep neural networks. Adam is based on momentum [15] and RMSProp. The adaptive mean square gradient (AMSGrad) algorithm [16], [17] that is based on Adam is another useful algorithm for training deep neural networks. Variants of Adam and AMSGrad have been presented that adapt the step sizes; they include belief in observed gradients (AdaBelief) [18] and AMSGrad with weighted gradient and dynamic bound (AMSGWDC) [19]. The above algorithms are called first-order stochastic optimization algorithms and use the stochastic gradient of an observed differentiable function.

In this paper, we focus on nonconvex stochastic optimization in deep neural networks and consider algorithms for nonconvex stochastic optimization. Nonconvex stochastic optimization is divided into two classes: finite-sum optimization and online optimization. The objective function in finite-sum optimization is defined as the sum of all loss functions, while the objective function in online optimization is defined as an expectation of loss functions (see also Assumption II.1(A2) for the definitions of objective functions).

H. Iiduka is with the Department of Computer Science, Meiji University, Kanagawa 214-8571, Japan (e-mail: iiduka@cs.meiji.ac.jp). This work was supported by JSPS KAKENHI Grant Number 21K11773.

A. Existing results

The convergence rate analyses for SGD [10], Mini-batch SGD (MSGD) [11], AMSGrad [17], and AdaBelief [18] are summarized in Table I. The table shows that these algorithms with diminishing learning rates can only be applied to unconstrained finite-sum optimization.

Meanwhile, the stochastic path-integrated differential estimator (SPIDER) [20] can be applied to unconstrained online optimization as well as finite-sum optimization. SPIDER, which is a first-order stochastic optimization algorithm, is useful for nonconvex optimization and it can achieve an ϵ-approximation such that the mean of the expectation of the gradient norm is less than or equal to ϵ (see also Table I for the detailed definition of ϵ-approximation). Showing that an ϵ-approximation exists is important because it lets us know in advance the number of iterations and the stochastic gradient complexity (SGC), which is the stochastic gradient computation cost, to evaluate the performance of the optimization algorithms.

Recently, an algorithm [21] was presented to unify the existing ALROAs, including SGD, AMSGrad, AMSGWDC, and AdaBelief. The unified algorithm can be applied to not only unconstrained but also *constrained* finite-sum and online optimization, as indicated in the two "ALROAs [21]" rows in Table I. It was shown in [21] that the unified algorithm with diminishing learning rates has an $\mathcal{O}(1/\sqrt{k})$ convergence, where k is the number of iterations. This implies that the convergence rate of the existing ALROAs, such as AMSGrad and AdaBelief, is $\mathcal{O}(1/\sqrt{k})$, which is an improvement on the previous results [17], [18] in Table I.

B. Motivation

Subsection I-A indicated that the existing first-order stochastic optimization algorithms are useful for solving nonconvex optimization problems in deep neural networks. The previous results in [21] only gave convergence analyses of ALROAs for finite-sum and online optimization. It is important to clarify the mini-batch sizes and learning rates for ALROAs in order to achieve ϵ-approximations in both theory and practice.

Here, we have two motivations related to the previous results in [20], [21]. The first is to determine the mini-batch

TABLE I: Comparison of convergence rates, numbers of iterations, and stochastic gradient complexities (SGC) needed for ϵ-approximation of first-order stochastic optimization algorithms for unconstrained and constrained nonconvex optimization

		Unconstrained nonconvex optimization			Constrained nonconvex optimization		
		Convergence rate	ϵ-approximation (i) Iteration	ϵ-approximation (ii) SGC	Convergence rate	ϵ-approximation (i) Iteration	ϵ-approximation (ii) SGC
Finite -sum	SGD [10]	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	———	———	———	———	———
	MSGD [11]	$\mathcal{O}\left(\frac{1}{k}\right)+C(m)$	———	———	———	———	———
	AMSGrad [17]	$\mathcal{O}\left(\frac{\ln k}{\sqrt{k}}\right)$	———	———	———	———	———
	AdaBelief [18]	$\mathcal{O}\left(\frac{\ln k}{\sqrt{k}}\right)$	———	———	———	———	———
	SPIDER [20]	———	$\left\lfloor \frac{4L\Delta n_0}{\epsilon^2} \right\rfloor + 1$	$\mathcal{O}\left(n+\frac{\sqrt{n}}{\epsilon^2}\right)$	———	———	———
	ALROAs [21]	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	———	———	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	———	———
	this work	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor \frac{1}{\epsilon^4} \right\rfloor + 1$	$\mathcal{O}\left(\min\left\{n+\frac{n}{\epsilon^4}, \frac{1}{\epsilon^4}\right\}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor \frac{1}{\epsilon^4} \right\rfloor + 1$	$\mathcal{O}\left(\min\left\{n+\frac{n}{\epsilon^4}, \frac{1}{\epsilon^4}\right\}\right)$
Online	SPIDER [20]	———	$\left\lfloor \frac{4L\Delta n_0}{\epsilon^2} \right\rfloor + 1$	$\mathcal{O}\left(\frac{1}{\epsilon^3}\right)$	———	———	———
	ALROAs [21]	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	———	———	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	———	———
	this work	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor \frac{1}{\epsilon^4} \right\rfloor + 1$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor \frac{1}{\epsilon^4} \right\rfloor + 1$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$

Let n be the total number of samples, k the number of iterations, and $C(m)$ a positive constant dependent on the mini-batch size m. (See Section III-A for the definitions of the parameters L, Δ, and n_0.) For the constrained case, the convergence rate of an algorithm when the learning rate α_k is $\mathcal{O}(1/\sqrt{k})$ or $\mathcal{O}(1/k)$ is measured by the upper bound of $\min_{j\in[k]} \mathbb{E}[\langle \boldsymbol{x}_j - \boldsymbol{x}, \nabla f(\boldsymbol{x}_j)\rangle]$ ($\boldsymbol{x} \in X$), which is a generalization of $\min_{j\in[k]} \mathbb{E}[\|\nabla f(\boldsymbol{x}_j)\|^2]$, which is in turn a measure of the convergence rate for the unconstrained case, where $[k] = \{1, 2, \ldots, k\}$. For the constrained case, the number of iterations K needed for an ϵ-approximation of an algorithm is measured by $(1/K)\sum_{k=1}^{K} \mathbb{E}[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle] \leq \epsilon^2$ ($\boldsymbol{x} \in X$), while for the unconstrained case, it is measured by $(1/K)\sum_{k=1}^{K} \mathbb{E}[\|\nabla f(\boldsymbol{x}_k)\|] \leq \epsilon$.

size, learning rates, and number of iterations to achieve an ϵ-approximation of ALROAs. Motivated by the results in [20], we would like to devise an ϵ-approximation of ALROAs for constrained finite-sum optimization as well as for constrained online optimization. Additionally, we would like to know the SGC of such an ϵ-approximation.

Our second motivation is to determine whether ALROAs based on different learning rate rules can achieve ϵ-approximations. Here, there are two learning rate rules in the convergence analyses of first-order stochastic optimization algorithms. The constant learning rate rule is used for SGD [10], SPIDER [20], and ALROAs [21], while the diminishing learning rate rule is used for SGD [10], MSGD [11], AMSGrad [17], AdaBelief [18], and ALROAs [21].

C. Contribution

This paper shows that ALROAs with not only *constant* but also *diminishing* learning rates can be ϵ-approximations for *constrained* finite-sum and online optimization. The results of the finite-sum and online optimization for ALROAs with diminishing learning rates are summarized in the two "this work" rows in Table I (see Table III for the results of finite-sum and online optimization for ALROAs with constant learning rates).

First, we show that ALROAs with *constant* learning rates can achieve ϵ-approximations for both constrained finite-sum and online optimization (Section IV-A). Here, the number of iterations needed for an ϵ-approximation is $\mathcal{O}(\lfloor 1/\epsilon^4 \rfloor)+1$. The SGCs are $\mathcal{O}(\min\{n+n/\epsilon^4, 1/\epsilon^4\})$ for constrained finite-sum optimization and $\mathcal{O}(1/\epsilon^4)$ for online optimization (Theorem IV.1 ii) and Table III). Moreover, we show that, for ALROAs with constant learning rates, an upper bound of the expectation of the variational inequality is $\mathcal{O}(1/k) + \epsilon^2$ (Theorem IV.1 i) and Table III). While SPIDER [20] with constant learning rates can be applied to unconstrained finite-sum and online optimization, we should emphasize that our results apply to *constrained* finite-sum and online optimization. Obviously, our results also allow us to perform unconstrained optimization because constrained optimization reduces to unconstrained optimization.

The previous studies [10], [11], [17], [18], [21] showed that SGD, AMSGrad, and the variants of Adam and AMSGrad can perform nonconvex optimization in deep neural networks (see Table I). However, they only presented convergence rate analyses of ALROAs and did not show any ϵ-approximation of ALROAs with diminishing learning rates. The second contribution of this paper is to show that ALROAs with *diminishing* learning rates can achieve ϵ-approximations for both constrained finite-sum and online optimization (Section IV-B). Here, the number of iterations needed for an ϵ-approximation is $\lfloor 1/\epsilon^4 \rfloor + 1$. The SGCs of constrained finite-sum and online optimization are, respectively, $\mathcal{O}(\min\{n + n/\epsilon^4, 1/\epsilon^4\})$ and $\mathcal{O}(1/\epsilon^4)$ (Theorem IV.2 ii) and Tables I and III). Moreover, we show that the convergence rate of ALROAs with a diminishing learning rate $\alpha_k = 1/\sqrt{k}$ is $\mathcal{O}(1/\sqrt{k})$, which is the same result as in [21] (Theorem IV.2 i)).

D. Comparisons of our results with recent studies

This paper is related to recent papers [7], [8], [21]. In [8], it was studied how increasing the batch size affects the performance of SGD, SGD with momentum [15], [22], and Nesterov momentum [23], [24]. Adam and K-FAC (Kronecker-factored approximate curvature [25]) were studied in [7]. It was numerically shown that increasing the batch size tends to decrease the number of iterations K needed to achieve an ϵ-approximation. However, it was also shown that there are diminishing returns to increasing the batch size to yield a decrease in K [7, Figure 8], [8, Figure 4]. Moreover, for a fixed batch size, the smaller

ϵ is, the larger K becomes [8, Figure 2]. The numerical results in [7], [8] support our results showing that K is proportional to $1/\epsilon^4$ (see also Theorems IV.1 and IV.2 and Tables I and III).

In [21], it was shown that ALROAs with a diminishing learning rate $\alpha_k = 1/k^\eta$, where $\eta \in (1/2, 1]$, converge to stationary points of general constrained nonconvex stochastic optimization problems. However, it is not guaranteed that this result implies the existence of ϵ-approximations. This is because the previous study [21] did not consider how setting the mini-batch size affects the performance of ALROAs. Hence, further analyses are needed to guarantee that ALROAs can be ϵ-approximations. Thus, the novelty of this paper compared with [21] is to show that ALROAs using an appropriate mini-batch size are ϵ-approximations for constrained nonconvex stochastic optimization problems (see also the two "this work" rows in Table I and Section I-C). Section IV-C provides detailed comparisons with the previous results in [21].

This paper is organized as follows: Section II presents mathematical preliminaries, including the assumptions of a constrained set and on the objective function. Section III reviews the existing algorithms for deep neural networks. Section IV shows that ALROAs can achieve ϵ-approximations for constant learning rate and diminishing learning rate rules. It also compares the results of our method with those of previous methods. Section V concludes the paper with a brief summary.

II. Mathematical Preliminaries

A. Notation and definitions

$|\mathcal{S}|$ denotes the number of elements of a set $\mathcal{S}$. $\mathbb{N}$ denotes the set of all positive integers and zero. Let $n \in \mathbb{N}\backslash\{0\}$. We define $[n] := \{1, 2, \ldots, n\}$. $\mathbb{R}^d$ denotes a d-dimensional Euclidean space with inner product $\langle \cdot, \cdot \rangle$, which induces the norm $\| \cdot \|$. $\mathbb{S}^d$ denotes the set of $d \times d$ symmetric matrices, i.e., $\mathbb{S}^d = \{M \in \mathbb{R}^{d\times d} \colon M = M^\top\}$, where we use $M^\top$ to indicate its transpose. Let $\mathbb{S}^d_{++}$ be the set of $d \times d$ symmetric positive-definite matrices, i.e., $\mathbb{S}^d_{++} = \{M \in \mathbb{S}^d \colon M \succ O\}$, and let $\mathbb{D}^d$ be the set of $d \times d$ diagonal matrices denoted by $\mathbb{D}^d = \{M \in \mathbb{R}^{d\times d} \colon M = \mathsf{diag}(x_i),\ x_i \in \mathbb{R}\ (i \in [d])\}$.

We define $\boldsymbol{x} \odot \boldsymbol{x}$ for $\boldsymbol{x} := (x_i)_{i=1}^d \in \mathbb{R}^d$ by $\boldsymbol{x} \odot \boldsymbol{x} := (x_i^2)_{i=1}^d \in \mathbb{R}^d$. Suppose that $H \in \mathbb{S}^d_{++}$. The H-inner product of $\mathbb{R}^d$ is defined for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_H := \langle \boldsymbol{x}, H\boldsymbol{y} \rangle$ and the H-norm is defined by $\|\boldsymbol{x}\|_H := \sqrt{\langle \boldsymbol{x}, H\boldsymbol{x} \rangle}$. The metric projection onto a nonempty, closed convex set X ($\subset \mathbb{R}^d$) is denoted by $P_X \colon \mathbb{R}^d \to X$. It is defined for all $\boldsymbol{x} \in \mathbb{R}^d$ by $P_X(\boldsymbol{x}) \in X$ and $\|P_X(\boldsymbol{x}) - \boldsymbol{x}\| = \inf_{\boldsymbol{y}\in X} \|\boldsymbol{y} - \boldsymbol{x}\|$. We use $P_{X,H}$ for the metric projection onto X under the H-norm. For a random variable Z, we use $\mathbb{E}[Z]$ to indicate its expectation.

B. Assumptions of constrained set, objective function, and gradient estimation

This paper considers optimization problems under the following assumptions.

Assumption II.1

(A1) [Constrained set] *$X \subset \mathbb{R}^d$ is a nonempty, closed convex set onto which the projection can be easily computed.*

(A2) [Objective function] *$f_i \colon \mathbb{R}^d \to \mathbb{R}$ ($i \in [n]$) is differentiable and $f \colon \mathbb{R}^d \to \mathbb{R}$ is defined for all $\boldsymbol{x} \in \mathbb{R}^d$ by*

$$f(\boldsymbol{x}) := \begin{cases} \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{x}) & \textit{(finite-sum)}, \\ \mathbb{E}[f_\xi(\boldsymbol{x})] & \textit{(online)}. \end{cases}$$

The gradient vector of f at $\boldsymbol{x} \in \mathbb{R}^d$, denoted by $\nabla f(\boldsymbol{x})$, coincides with $\mathbb{E}[\nabla f_i(\boldsymbol{x})]$.

(A3) [Gradient estimation] *For each iteration k, the optimization algorithms sample a mini-batch $\mathcal{S}_k \subset [n]$, of size $s := |\mathcal{S}_k|$ independently of k and estimate the full gradient ∇f as*

$$\nabla f_{\mathcal{S}_k} := \frac{1}{s}\sum_{i\in\mathcal{S}_k} \nabla f_i.$$

(A4) [Gradient boundedness] *There exists a positive number M such that, for all $\boldsymbol{x} \in X$, $\mathbb{E}[\|\nabla f_{\mathcal{S}_k}(\boldsymbol{x})\|^2] \le M^2$.*

Examples of X satisfying (A1) are the whole space $\mathbb{R}^d$, a closed ball, an affine subspace, a halfspace, and a hyperslab [26, Chapter 28]. Assumption (A2) is a standard one for nonconvex optimization in deep neural networks (see, e.g., [17, (2)] and [20, (1.1), (1.2), (B.5)]). Assumption (A3) is needed for the optimization algorithms to work (see, e.g., [17, Section 2] and [20, Notation section]), and Assumption (A4) is used to analyze the optimization algorithms (see, e.g., [17, A2]).

III. Nonconvex Optimization Problem in Deep Neural Networks

This paper deals with the following stationary point problem for nonconvex optimization to minimize an objective function f in (A2) over a constrained set X in (A1).

Problem III.1 *Under Assumption II.1, we would like to find a stationary point $\boldsymbol{x}^\star$ of a nonconvex optimization problem to minimize f over X, i.e.,*

$$\boldsymbol{x}^\star \in X^\star := \left\{\boldsymbol{x}^\star \in X \colon \langle \boldsymbol{x}^\star - \boldsymbol{x}, \nabla f(\boldsymbol{x}^\star)\rangle \le 0\ (\boldsymbol{x} \in X)\right\}.$$

Suppose that $X = \mathbb{R}^d$. Let $\boldsymbol{x}^\star \in X^\star$. Setting $\boldsymbol{x} := \boldsymbol{x}^\star - \nabla f(\boldsymbol{x}^\star) \in \mathbb{R}^d$ ensures that $0 \ge \langle \nabla f(\boldsymbol{x}^\star), \nabla f(\boldsymbol{x}^\star)\rangle = \|\nabla f(\boldsymbol{x}^\star)\|^2$, which implies that $\nabla f(\boldsymbol{x}^\star) = \boldsymbol{0}$. If $\boldsymbol{x}^\star$ satisfies $\nabla f(\boldsymbol{x}^\star) = \boldsymbol{0}$, then $\boldsymbol{x}^\star \in X^\star$. Hence, Problem III.1 with $X = \mathbb{R}^d$ can be expressed as the problem [17], [18] of finding a local minimizer of f over $\mathbb{R}^d$, i.e.,

$$X^\star = \left\{\boldsymbol{x}^\star \in \mathbb{R}^d \colon \nabla f(\boldsymbol{x}^\star) = \boldsymbol{0}\right\}. \tag{1}$$

Accordingly, Problem III.1 is a generalization of problem (1). Let $\epsilon > 0$. Then, the inequality $\langle \boldsymbol{x}^\star - \boldsymbol{x}, \nabla f(\boldsymbol{x}^\star)\rangle \le \epsilon^2$ ($\boldsymbol{x} \in X = \mathbb{R}^d$) implies that $\|\nabla f(\boldsymbol{x}^\star)\| \le \epsilon$ (see also Table I for the detailed definition of ϵ-approximation). If f is convex [14], [16], the solution to Problem III.1 is a global minimizer of f over X.

While problem (1) requires us to solve a nonlinear equation $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$, Problem III.1 requires us to solve a *variational inequality* [27], [28], [29] $\langle \boldsymbol{x} - \boldsymbol{y}, \nabla f(\boldsymbol{x})\rangle \le 0$ ($\boldsymbol{y} \in X$). For general constrained optimization, it is not guaranteed that a

solution of the variational inequality satisfies $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$.[1] Hence, in constrained optimization, we must consider more difficult cases, i.e., $\nabla f(\boldsymbol{x}) \neq \boldsymbol{0}$ and $\langle \boldsymbol{x} - \boldsymbol{y}, \nabla f(\boldsymbol{x}) \rangle \leq 0$ for *all* $\boldsymbol{y} \in X$ than $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$.

A. Related work

This section reviews the first-order stochastic optimization algorithms for nonconvex optimization that are listed in Table I and mentioned in Section I-A.

1) Algorithms for unconstrained finite-sum optimization: First, we consider an unconstrained finite-sum optimization, i.e., problem (1) when $f = (1/n)\sum_{i=1}^n f_i$. The simplest way of solving it is to use SGD as follows:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{g}_k, \tag{2}$$

where $(\alpha_k)_{k\in\mathbb{N}} \subset (0, +\infty)$ and $\boldsymbol{g}_k$ denotes the stochastic gradient. When the learning rate α_k is constant, i.e., $\alpha_k = 1/\beta$, where $\beta > 0$ is the Lipschitz constant of ∇f, Theorem 12 in [10] shows that SGD (2) satisfies the condition that there exist positive real numbers M_1 and M_2 such that, for all $k \geq 1$, almost surely

$$\frac{1}{k}\sum_{j=1}^k \|\nabla f(\boldsymbol{x}_j)\|^2 \leq \frac{M_1}{k} + M_2. \tag{3}$$

When the learning rate α_k is diminishing, i.e., $\alpha_k = \min\{\mathcal{O}(1/\sqrt{k}), 1/\beta\}$, Theorem 11 in [10] shows that SGD (2) satisfies the condition that there exist positive real numbers M_3 and M_4 such that, for all $k \geq 1$,

$$\frac{1}{k}\sum_{j=1}^k \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_j)\|^2\right] \leq \frac{M_3}{k} + \frac{M_4}{\sqrt{k}} \tag{4}$$

(See also Table I for the convergence rate of SGD [10]).

Convergence rate analyses of SGD (2) depending on the mini-batch size were presented in [11]. In particular, Theorem 3.2 in [11] indicated that, under certain assumptions, MSGD with $\alpha_k = \mathcal{O}(1/k)$ satisfies the condition that there exists a positive real number M_5 such that, with probability at least $1 - CKe^{-cm^{2\epsilon}}$,

$$\|\nabla f(\boldsymbol{x}_K)\|^2 \leq C\left(\frac{M_5}{K+1} + m^{-\frac{1}{2}+\epsilon}\right), \tag{5}$$

where $c, C > 0$, m ($> C$) is the mini-batch size, and $\epsilon \in (0, (C\ln\ln m)/\ln m)$ (See also Table I).

The following generalized adaptive moment estimation (GAdam) was presented in [17]:

$$\begin{aligned} &\boldsymbol{m}_k = \beta_k \boldsymbol{m}_{k-1} + (1-\beta_k)\boldsymbol{g}_k, \\ &\hat{\boldsymbol{v}}_k = h_k(\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_k), \\ &\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \frac{\boldsymbol{m}_k}{\sqrt{\boldsymbol{v}_k}}, \end{aligned} \tag{6}$$

where $(\alpha_k)_{k\in\mathbb{N}}, (\beta_k)_{k\in\mathbb{N}} \subset (0, +\infty)$, and $\boldsymbol{m}_k/\sqrt{\boldsymbol{v}_k}$ denotes element-wise division. GAdam is a generalization [17, Table 1] of popular algorithms such as SGD, AdaGrad [12], AMSGrad [16], and RMSProp [13, Algorithm 8.5]. For example, AMSGrad is GAdam when

$$\begin{aligned} &\boldsymbol{v}_k := \beta \boldsymbol{v}_{k-1} + (1-\beta)\boldsymbol{g}_k \odot \boldsymbol{g}_k \text{ and} \\ &\hat{\boldsymbol{v}}_k := (\max\{\hat{v}_{k-1,i}, v_{k,i}\})_{i=1}^d, \end{aligned} \tag{7}$$

where $\beta \in (0, 1)$. Corollaries 3.1 and 3.2 in [17] showed that, if the learning rate $\alpha_k = \mathcal{O}(1/\sqrt{k})$ is chosen, then AMSGrad and AdaGrad with first-order momentum (AdaFom), which is an example of GAdam, achieve the following convergence rate:

$$\min_{j\in[k]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_j)\|^2\right] \leq \frac{1}{\sqrt{k}}(Q_1 + Q_2 \ln k), \tag{8}$$

where Q_1 and Q_2 are two positive constants independent of k (See also Table I).

AdaBelief was presented in [18]; it is defined as

$$\begin{aligned} &\boldsymbol{m}_k = \beta_1 \boldsymbol{m}_{k-1} + (1-\beta_1)\boldsymbol{g}_k, \\ &\hat{\boldsymbol{m}}_k = \frac{\boldsymbol{m}_k}{1-\beta_1^k}, \\ &\boldsymbol{s}_k = \beta_2 \boldsymbol{s}_{k-1} + (1-\beta_2)(\boldsymbol{g}_k - \boldsymbol{m}_k) \odot (\boldsymbol{g}_k - \boldsymbol{m}_k), \\ &\hat{\boldsymbol{s}}_k = \frac{\boldsymbol{s}_k}{1-\beta_2^k}, \\ &\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \frac{\hat{\boldsymbol{m}}_k}{\sqrt{\hat{\boldsymbol{s}}_k}}, \end{aligned} \tag{9}$$

where $\beta_1, \beta_2 \in (0, 1)$. Theorem 2.2 in [18] shows that, if $s_{k,i} \leq s_{k+1,i}$ holds for all k and all i, then AdaBelief (9) with $\alpha_k = \mathcal{O}(1/\sqrt{k})$ achieves a (8) convergence rate (See also Table I).

SPIDER was presented in [20]: Let $q, \epsilon > 0$ and $\mathcal{S}_1, \mathcal{S}_2 \subset [n]$. For $k = 0, 1, \ldots, K$,

$$\begin{aligned} &\boldsymbol{v}_k = \begin{cases} \nabla f_{\mathcal{S}_1}(\boldsymbol{x}_k) & (\mathrm{mod}(k,q) = 0), \\ \nabla f_{\mathcal{S}_2}(\boldsymbol{x}_k) - \nabla f_{\mathcal{S}_2}(\boldsymbol{x}_{k-1}) + \boldsymbol{v}_{k-1} & (\mathrm{mod}(k,q) \neq 0), \end{cases} \\ &\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta \frac{\boldsymbol{v}_k}{\|\boldsymbol{v}_k\|}, \end{aligned} \tag{10}$$

where $\nabla f_{\mathcal{S}_i}$ $(i = 1, 2)$ is defined by (A3). We define

$$\begin{aligned} &\mathcal{S}_1 := [n],\ |\mathcal{S}_2| := \frac{\sqrt{n}}{n_0},\ \eta := \frac{\epsilon}{Ln_0},\ q := n_0\sqrt{n},\ \text{and} \\ &K := \left\lfloor \frac{4L\Delta n_0}{\epsilon^2} \right\rfloor + 1, \end{aligned} \tag{11}$$

where $n_0 \in [1, \sqrt{n}]$, $\Delta := f(\boldsymbol{x}_0) - \inf_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x})$, and $L > 0$ denotes the Lipschitz constant in the sense that $\mathbb{E}[\|\nabla f_i(\boldsymbol{x}) - f_i(\boldsymbol{y})\|^2] \leq L^2\|\boldsymbol{x} - \boldsymbol{y}\|^2$ $(\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d)$ [20, Assumption 1(ii)]. Accordingly, SPIDER (10) satisfies

$$\mathbb{E}\left[\|\nabla f(\tilde{\boldsymbol{x}})\|\right] := \frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|\right] \leq 5\epsilon, \tag{12}$$

which implies that SPIDER (10) finds an ϵ-approximation first-order stationary point in the sense of the mean of the expected gradient norms [20, Theorem 2]. Moreover, the gradient cost of SPIDER (10) is bounded by $n + 8L\Delta\sqrt{n}/\epsilon^2 + 2\sqrt{n}/n_0$ and the SGC is $\mathcal{O}(n + \sqrt{n}/\epsilon^2)$ (See also Table I).

[1] For example, a global minimizer of $f(x) = x^2$ over $X = \{x \in \mathbb{R} \colon x \geq 1\}$ is $x^\star = 1$, which satisfies $f'(x^\star) = 2 \neq 0$ and $\langle x^\star - y, f'(x^\star)\rangle \leq 0$ $(y \in X)$.

2) Algorithms for unconstrained online optimization: SPIDER (10) can be also applied to the unconstrained online optimization, i.e., problem (1) when $f = \mathbb{E}[f_\xi]$. Here, we define

$$|\mathcal{S}_1| := \frac{2\sigma^2}{\epsilon^2},\ |\mathcal{S}_2| := \frac{2\sigma}{\epsilon n_0},\ \eta := \frac{\epsilon}{Ln_0},\ q := \frac{\sigma n_0}{\epsilon}, \\ \text{and } K := \left\lfloor \frac{4L\Delta n_0}{\epsilon^2} \right\rfloor + 1, \tag{13}$$

and assume that the stochastic gradient has a finite variance bounded by σ^2 in the sense that $\mathbb{E}[\|\nabla f_i(\boldsymbol{x}) - \nabla f(\boldsymbol{x})\|^2] \leq \sigma^2$ ($\boldsymbol{x} \in \mathbb{R}^d$) [20, Assumption 1(iii)]. SPIDER (10) satisfies (12), and its SGC is $\mathcal{O}(1/\epsilon^3)$ [20, Theorem 1] (See also Table I).

3) Algorithms for constrained finite-sum and online optimization: The following algorithm (Algorithm 1) was presented in [21] to unify the various ALROAs for solving Problem III.1.

Algorithm 1 Unified algorithm of ALROAs for solving Problem III.1

Require: $(\alpha_k)_{k\in\mathbb{N}} \subset (0,1)$, $(\beta_k)_{k\in\mathbb{N}} \subset [0,b] \subset [0,1)$, $\gamma \in [0,1)$

$k \leftarrow 0$, $\boldsymbol{x}_0, \boldsymbol{m}_{-1} \in \mathbb{R}^d$, $\mathsf{H}_0 \in \mathbb{S}^d_{++} \cap \mathbb{D}^d$, $\mathcal{S}_0 \subset [n]$
loop
  $\boldsymbol{m}_k := \beta_k \boldsymbol{m}_{k-1} + (1-\beta_k)\nabla f_{\mathcal{S}_k}(\boldsymbol{x}_k)$
  $\hat{\boldsymbol{m}}_k := \dfrac{\boldsymbol{m}_n}{1-\gamma^{k+1}}$
  $\mathsf{H}_k \in \mathbb{S}^d_{++} \cap \mathbb{D}^d$ (see Table II for examples of H_k)
  Find $\mathbf{d}_k \in \mathbb{R}^d$ that solves $\mathsf{H}_k \mathbf{d} = -\hat{\boldsymbol{m}}_k$
  $\boldsymbol{x}_{k+1} := P_{X,\mathsf{H}_k}(\boldsymbol{x}_k + \alpha_k \mathbf{d}_k)$
  $k \leftarrow k+1$
end loop

Table II lists examples of H_k and shows that Algorithm 1 with $X = \mathbb{R}^d$ includes the existing ALROAs, such as Nesterov momentum [23], [24], AMSGrad [16], [17], and AMSGWDC [19],[2] AdaBelief [18], and modified Adam (MAdam) [21], for unconstrained nonconvex finite-sum optimization (see also (7) and (9) for the definitions of AMSGrad and AdaBelief).

Here, we would like to emphasize that Algorithm 1 can be applied to not only finite-sum but also online constrained optimization. When the learning rates α_k and β_k are constant, i.e., $\alpha_k = \alpha$ and $\beta_k = \beta$, Algorithm 1 satisfies the following condition [21, Theorem 1]: there exist positive real numbers Q_3 and Q_4 such that, for all $\boldsymbol{x} \in X$ and all $K \geq 1$,

$$\liminf_{k\to+\infty} \mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq Q_3\alpha + Q_4\beta, \tag{14}$$

$$\frac{1}{K}\sum_{k=1}^{K} \mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq \mathcal{O}\left(\frac{1}{K}\right) + Q_3\alpha + Q_4\beta. \tag{15}$$

[2]AMSGWDC uses the following settings: $(l_k)_{k\in\mathbb{N}} \subset \mathbb{R}$ is monotone increasing, $(u_k)_{k\in\mathbb{N}} \subset \mathbb{R}$ is monotone decreasing with $0 < l_k \leq u_k$ for all $k \in \mathbb{N}$, and $\mathrm{Clip}(\cdot, l, u)\colon \mathbb{R} \to \mathbb{R}$ ($l, u \in \mathbb{R}$ with $l \leq u$ are given) is defined for all $x \in \mathbb{R}$ by

$$\mathrm{Clip}(x,l,u) := \begin{cases} l & \text{if } x < l, \\ x & \text{if } l \leq x \leq u, \\ u & \text{if } x > u. \end{cases}$$

TABLE II: Examples of $\mathsf{H}_k \in \mathbb{S}^d_{++} \cap \mathbb{D}^d$ (step 5) in Algorithm 1 ($\delta, \zeta \in [0,1)$)

	H_k
SGD ($\beta_k = \gamma = 0$)	H_k is the identity matrix.
Momentum [23] ($\gamma = 0$)	H_k is the identity matrix.
AMSGrad [17] ($\gamma = 0$)	$\boldsymbol{v}_k = \delta \boldsymbol{v}_{k-1} + (1-\delta)\nabla f_{\mathcal{S}_k}(\boldsymbol{x}_k) \odot \nabla f_{\mathcal{S}_k}(\boldsymbol{x}_k)$ $\hat{\boldsymbol{v}}_k = (\max\{\hat{v}_{k-1,i}, v_{k,i}\})_{i=1}^d$ $\mathsf{H}_k = \mathsf{diag}(\sqrt{\hat{v}_{k,i}})$
AMSGWDC [19] ($\gamma = 0$)	$\boldsymbol{v}_k = \delta \boldsymbol{v}_{k-1} + (1-\delta)\nabla f_{\mathcal{S}_k}(\boldsymbol{x}_k) \odot \nabla f_{\mathcal{S}_k}(\boldsymbol{x}_k)$ $\hat{\boldsymbol{v}}_k = (\max\{\hat{v}_{k-1,i}, v_{k,i}\})_{i=1}^d$ $\tilde{\boldsymbol{v}}_k = \left(\mathrm{Clip}\left(\frac{1}{\sqrt{\hat{v}_{k,i}}}, l_k, u_k\right)^{-1}\right)_{i=1}^d$ $\mathsf{H}_k = \mathsf{diag}(\sqrt{\tilde{v}_{k,i}})$
AdaBelief [18] ($s_{k,i} \leq s_{k+1,i}$ is needed)	$\tilde{\boldsymbol{s}}_k = (\nabla f_{\mathcal{S}_k}(\boldsymbol{x}_k) - \boldsymbol{m}_k) \odot (\nabla f_{\mathcal{S}_k}(\boldsymbol{x}_k) - \boldsymbol{m}_k)$ $\boldsymbol{s}_k = \delta \boldsymbol{v}_{k-1} + (1-\delta)\tilde{\boldsymbol{s}}_k$ $\hat{\boldsymbol{s}}_k = \frac{\boldsymbol{s}_k}{1-\zeta^k}$ $\mathsf{H}_k = \mathsf{diag}(\sqrt{\hat{s}_{k,i}})$
MAdam [21]	$\boldsymbol{v}_k = \delta \boldsymbol{v}_{k-1} + (1-\delta)\nabla f_{\mathcal{S}_k}(\boldsymbol{x}_k) \odot \nabla f_{\mathcal{S}_k}(\boldsymbol{x}_k)$ $\bar{\boldsymbol{v}}_k = \frac{\boldsymbol{v}_k}{1-\delta^{k+1}}$ $\hat{\boldsymbol{v}}_k = (\max\{\hat{v}_{k-1,i}, \bar{v}_{k,i}\})_{i=1}^d$ $\mathsf{H}_k = \mathsf{diag}(\sqrt{\hat{v}_{k,i}})$

When α_k and β_k are diminishing, i.e., $\alpha_k = \mathcal{O}(1/k^\eta)$ and $\beta_k = \lambda^k$, where $\eta \in [1/2, 1]$ and $\lambda \in (0,1)$, Algorithm 1 satisfies the following condition [21, Theorem 2]: for all $\boldsymbol{x} \in X$ and all $K \geq 1$,

$$\liminf_{k\to+\infty} \mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq 0, \tag{16}$$

$$\frac{1}{K}\sum_{k=1}^{K} \mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq \mathcal{O}\left(\frac{1}{K^{1-\eta}}\right), \tag{17}$$

where (16) holds for $\eta \in (1/2, 1]$ and (17) holds for $\eta \in [1/2, 1)$. Inequality (16) ensures that there exists a subsequence $(\boldsymbol{x}_{k_i})_{i\in\mathbb{N}}$ of $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$ such that $(\boldsymbol{x}_{k_i})_{i\in\mathbb{N}}$ converges to a solution of Problem III.1. Moreover, from (17) under $\eta = 1/2$, Algorithm 1 satisfies that, for all $\boldsymbol{x} \in X$ and all $K \geq 1$,

$$\min_{k\in[K]} \mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq \mathcal{O}\left(\frac{1}{\sqrt{K}}\right), \tag{18}$$

which, together with $X = \mathbb{R}^d$, implies that

$$\min_{k\in[K]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] = \mathcal{O}\left(\frac{1}{\sqrt{K}}\right) \tag{19}$$

(See also Table I for the convergence rate of ALROAs [21]).

IV. ϵ-APPROXIMATION OF ALGORITHM 1

To analyze Algorithm 1, we assume the following conditions [21].

Assumption IV.1 *The sequence $(\mathsf{H}_k)_{k\in\mathbb{N}} \subset \mathbb{S}^d_{++} \cap \mathbb{D}^d$, denoted by $\mathsf{H}_k := \mathsf{diag}(h_{k,i})$, in Algorithm 1 satisfies the following conditions:*

(A5) *$h_{k+1,i} \geq h_{k,i}$ almost surely for all $k \in \mathbb{N}$ and all $i \in [d]$;*
(A6) *For all $i \in [d]$, a positive number B_i exists such that $\sup\{\mathbb{E}[h_{k,i}]\colon k \in \mathbb{N}\} \leq B_i$.*

Moreover,

(A7) $D := \max_{i\in[d]} \sup\{(x_{k,i}-x_i)^2 \colon k \in \mathbb{N}\} < +\infty$*, where* $\boldsymbol{x} := (x_i) \in X$ *and* $(\boldsymbol{x}_k)_{k\in\mathbb{N}} := ((x_{k,i}))_{k\in\mathbb{N}}$ *is the sequence generated by Algorithm 1.*

Assumption (A7) holds under the boundedness of X, which is assumed in [18, Theorem 2.1], [14, Theorem 4.1], [16, p.2], and [30, p.1574].

Obviously, $(\mathsf{H}_k)_{k\in\mathbb{N}}$ in Table II satisfies (A5). To justify that (A5) is needed to analyze Algorithm 1, we consider Adam [14] defined by

$$\begin{aligned} &\boldsymbol{v}_k = \delta \boldsymbol{v}_{k-1} + (1-\delta)\nabla f_{\mathcal{S}_k}(\boldsymbol{x}_k) \odot \nabla f_{\mathcal{S}_k}(\boldsymbol{x}_k),\\ &\bar{\boldsymbol{v}}_k = \frac{\boldsymbol{v}_k}{1-\delta^{k+1}},\\ &\mathsf{H}_k = \mathsf{diag}(\sqrt{\bar{v}_{k,i}}),\\ &\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \mathsf{H}_k^{-1} \hat{\boldsymbol{m}}_k, \end{aligned} \tag{20}$$

where $\hat{\boldsymbol{m}}_k$ is defined as in steps 3 and 4 of Algorithm 1. Adam (20) is one example of Algorithm 1 that does not satisfy (A5). Theorem 3 in [16] shows that there is a stochastic optimization problem for which Adam (20) does not converge to the optimal solution since Adam (20) does not satisfy

$$\frac{h_{k+1,i}}{\alpha_{k+1}} := \frac{\sqrt{\bar{v}_{k+1,i}}}{\alpha_{k+1}} \geq \frac{\sqrt{\bar{v}_{k,i}}}{\alpha_k} =: \frac{h_{k,i}}{\alpha_k}. \tag{21}$$

Assumption (A5) under $\alpha_{k+1} \leq \alpha_k$ ($k \in \mathbb{N}$), which is satisfied for constant and diminishing learning rates, implies (21). Therefore, (A5) is needed to guarantee convergence of Algorithm 1. In particular, Adam is modified with MAdam (Table II), which satisfies (21) and (A5), to ensure its convergence.

The previous results in [17, p.29], [18, p.18], and [21] show that $(\mathsf{H}_k)_{k\in\mathbb{N}}$ in Table II satisfies (A6). There is a relationship between Assumptions (A4) and (A6) in this paper and Assumption A.2 in [17]. Reference [17] assumes that $(\nabla f(\boldsymbol{x}_k))_{k\in\mathbb{N}}$ and $(\alpha_k \boldsymbol{m}_k / h_{k,i})_{k\in\mathbb{N}}$ are bounded [17, Assumption A.2, Theorem 3.1]. Meanwhile, the proof of [21, Lemma 2] ensures that, if (A4) holds, then $(\boldsymbol{m}_k)_{k\in\mathbb{N}}$ in Algorithm 1 is bounded (see also Lemma IV.1). Accordingly, (A4) implies that $(\alpha_k \boldsymbol{m}_k / h_{k,i})_{k\in\mathbb{N}}$ in Algorithm 1 is bounded. This paper does not assume the boundedness of $(\nabla f(\boldsymbol{x}_k))_{k\in\mathbb{N}}$, in contrast to [17]. Meanwhile, this paper assumes (A6) in place of the boundedness of $(\nabla f(\boldsymbol{x}_k))_{k\in\mathbb{N}}$ [17, Assumption A.2].

Algorithm 1 satisfies the following conditions.

Lemma IV.1 [21, Lemma A.2] *Under (A4), we have that, for all* $k \in \mathbb{N}$*,*

$$\mathbb{E}\left[\|\boldsymbol{m}_k\|^2\right] \leq \tilde{M}^2 := \max\left\{\|\boldsymbol{m}_{-1}\|^2, M^2\right\}.$$

In addition, under (A5), we have that, for all $k \in \mathbb{N}$*,*

$$\mathbb{E}\left[\|\mathbf{d}_k\|_{\mathsf{H}_k}^2\right] \leq \frac{\tilde{B}^2 \tilde{M}^2}{(1-\gamma)^2},$$

where

$$\tilde{B} := \sup\left\{\max_{i\in[d]} \frac{1}{\sqrt{h_{k,i}}} \colon k \in \mathbb{N}\right\} \leq \max_{i\in[d]} \frac{1}{\sqrt{h_{0,i}}} < +\infty.$$

A. Constant learning rate rule

We set

$$\Gamma := (1-b)^2(1-\gamma)^2 \text{ and } \tilde{L} := cdBD,$$

where B_i is defined as in (A6), $B := \max_{i\in[d]} B_i < +\infty$, D is defined as in (A7), d is the number of dimensions, $c > 0$ is chosen arbitrarily, and $\gamma, b \in [0,1)$ are used in Algorithm 1. Let $\epsilon > 0$. We define

$$\begin{aligned} s = |\mathcal{S}_k| &:= \begin{cases} \min\left\{n, \frac{c}{\tilde{B}^2\tilde{M}^2}, \frac{c}{\sqrt{dD}\tilde{M}}, \frac{4\Gamma}{3\tilde{L}}\right\} & \text{(finite-sum)},\\ \min\left\{\frac{c}{\tilde{B}^2\tilde{M}^2}, \frac{c}{\sqrt{dD}\tilde{M}}, \frac{4\Gamma}{3\tilde{L}}\right\} & \text{(online)}, \end{cases}\\ \alpha_k &:= \frac{2(1-b)(1-\gamma)^2}{3c}\epsilon^2,\\ \beta_k &:= \min\left\{\frac{1-b}{3c}\epsilon^2, b\right\}, \end{aligned} \tag{22}$$

where n is the number of samples and $\tilde{B}, \tilde{M} > 0$ are defined in Lemma IV.1.

The following is an ϵ-approximation analysis of Algorithm 1 with constant learning rates for constrained finite-sum and online optimization. The proof of Theorem IV.1 is given in the Supplementary Material.

Theorem IV.1 *Suppose that (A1)–(A7) hold and consider the sequence* $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$ *generated by Algorithm 1 with* (22) *for Problem III.1. Then, the following hold:*

i) *For all* $\boldsymbol{x} \in X$ *and all* $K \geq 1$*,*

$$\frac{1}{K}\sum_{k=1}^{K} \mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq \frac{1}{\epsilon^2 K} + \frac{2}{3}\epsilon^2.$$

ii) *To ensure that Algorithm 1 is an* ϵ*-approximation, i.e., for all* $\boldsymbol{x} \in X$*,*

$$\frac{1}{K}\sum_{k=1}^{K} \mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq \epsilon^2,$$

Algorithm 1 terminates in at most K *iterations, defined by*

$$K := \left\lfloor \frac{3}{\epsilon^4} \right\rfloor + 1.$$

The SGC of Algorithm 1 for constrained finite-sum optimization is

$$\mathcal{O}\left(\min\left\{n + \frac{n}{\epsilon^4}, \frac{1}{\epsilon^4}\right\}\right),$$

and the SGC of Algorithm 1 for constrained online optimization is

$$\mathcal{O}\left(\frac{1}{\epsilon^4}\right).$$

B. Diminishing learning rate rule

Let $\epsilon > 0$, $\lambda \in (0,1)$, and $\eta \in (0,1)$. We define

$$s = |\mathcal{S}_k| := \begin{cases} \min\left\{n, \frac{c}{\tilde{B}^2\tilde{M}^2}, \frac{c}{\sqrt{d\tilde{D}}\tilde{M}}, \frac{4(1-\eta)\Gamma}{9\tilde{L}}\right\} & \text{(finite-sum)}, \\ \min\left\{\frac{c}{\tilde{B}^2\tilde{M}^2}, \frac{c}{\sqrt{d\tilde{D}}\tilde{M}}, \frac{4(1-\eta)\Gamma}{9\tilde{L}}\right\} & \text{(online)}, \end{cases}$$

$$\alpha_k := \frac{2(1-b)(1-\gamma)^2(1-\eta)}{3ck^\eta},$$

$$\beta_k := \min\left\{\frac{(1-b)(1-\lambda)\lambda^{k-1}}{3c}, b\right\}, \tag{23}$$

where the parameters are as in Subsection IV-A.

The following is the ϵ-approximation analysis of Algorithm 1 with diminishing learning rates for constrained finite-sum and online optimization. The proof of Theorem IV.2 is given in the Supplementary Material.

Theorem IV.2 *Suppose that (A1)–(A7) hold and consider the sequence $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$ generated by Algorithm 1 with* (23) *for Problem III.1. Then, the following hold:*

i) *For all $\boldsymbol{x} \in X$ and all $K \geq 1$,*

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq \frac{1}{K^\theta},$$

where $\theta := \min\{\eta, 1-\eta\}$.

ii) *To ensure that Algorithm 1 is an ϵ-approximation, i.e., for all $\boldsymbol{x} \in X$,*

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq \epsilon^2,$$

Algorithm 1 terminates in at most K iterations, defined by

$$K := \left\lfloor \frac{1}{\epsilon^{\frac{2}{\theta}}} \right\rfloor + 1.$$

The SGC of Algorithm 1 for constrained finite-sum optimization is

$$\mathcal{O}\left(\min\left\{n + \frac{n}{\epsilon^{\frac{2}{\theta}}}, \frac{1}{\epsilon^{\frac{2}{\theta}}}\right\}\right),$$

and the SGC of Algorithm 1 for constrained online optimization is

$$\mathcal{O}\left(\frac{1}{\epsilon^{\frac{2}{\theta}}}\right).$$

C. Comparison of our results with previous studies

Theorems IV.1 and IV.2 lead to Table III showing the convergence rate and ϵ-approximation of Algorithm 1 (SGD, Momentum, AMSGrad, AMSGWDC, AdaBelief, and MAdam) for constrained finite-sum and online optimization.

Now let us check the performance of Algorithm 1 with a constant learning rate rule in detail. Theorem IV.1 i) shows that Algorithm 1 including SGD, Momentum, AMSGrad, AMSGWDC, AdaBelief, and MAdam satisfy that, for all $\boldsymbol{x} \in X$ and all $K \geq 1$,

$$\min_{k\in[K]}\mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq \mathcal{O}\left(\frac{1}{K}\right) + \epsilon^2. \tag{24}$$

Accordingly, there is a possibility that using a small parameter ϵ speeds up convergence of Algorithm 1. However, Theorem IV.1 ii) indicates that the smaller ϵ is, the larger the required iterations K and SGC become. A similar trend was observed in SPIDER (10) with a constant learning rate $\eta = \epsilon/Ln_0$ for unconstrained finite-sum and online optimization (see also Table I, (11), and (13)). The previous study [21] reported convergence analyses (14) and (15) for Algorithm 1 with constant learning rates, i.e.,

$$\liminf_{k\to+\infty}\mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq Q_3\alpha + Q_4\beta,$$

$$\min_{k\in[K]}\mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq \mathcal{O}\left(\frac{1}{K}\right) + Q_3\alpha + Q_4\beta,$$

where Q_3 and Q_4 are positive constants. If $\alpha = \beta = \mathcal{O}(\epsilon^2)$ (see (22)), then the previous result (15) coincides with (24). However, the previous study [21] did not consider how setting the mini-batch size affects the performance of Algorithm 1. In contrast to the previous study [21], Theorem IV.1 shows that Algorithm 1 with the constant learning rates and a mini-batch size defined by (22) can achieve ϵ-approximations. This implies that using an appropriate mini-batch size allows Algorithm 1 to be an ϵ-approximation.

Next, let us consider Algorithm 1 with diminishing learning rates. Theorem IV.2 i) guarantees that, for all $\boldsymbol{x} \in X$ and all $K \geq 1$,

$$\min_{k\in[K]}\mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq \frac{1}{K^\theta}, \tag{25}$$

which, together with $X = \mathbb{R}^d$, implies that

$$\min_{k\in[K]}\mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \leq \frac{1}{K^\theta}, \tag{26}$$

where $\theta = \min\{\eta, 1-\eta\}$. The previous study [21] reported convergence analyses (16), (17), (18), and (19) for Algorithm 1 with diminishing learning rates. In particular, Algorithm 1 using $\alpha_k = \mathcal{O}(1/\sqrt{k})$ (i.e., $\eta = 1/2$) and $\beta_k = \lambda^k$ satisfies that, for all $\boldsymbol{x} \in X$ and all $K \geq 1$,

$$\min_{k\in[K]}\mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq \mathcal{O}\left(\frac{1}{\sqrt{K}}\right),$$

$$\min_{k\in[K]}\mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] = \mathcal{O}\left(\frac{1}{\sqrt{K}}\right).$$

Accordingly, the previous results in [21] are the same as (25) and (26). As mentioned in the above paragraph, the previous study [21] did not consider how the mini-batch size affects the performance of Algorithm 1. As a result, the results in [21] could not show that Algorithm 1 is an ϵ-approximation. We would like to emphasize that Theorem IV.2 shows that Algorithm 1 with the diminishing learning rates and the mini-batch size defined by (23) can achieve ϵ-approximations, which never be shown by the previous study [21]. Here, our (25) result shows that the larger θ is, the quicker the convergence $\mathcal{O}(1/K^\theta)$ becomes. Moreover, Theorem IV.2 ii) indicates that, the larger θ is, the smaller the required iterations K and SGC become. Since $\theta = \min\{\eta, 1-\eta\}$ attains the maximum value, $1/2$, when $\eta = 1/2$, $\eta = 1/2$ is the best choice from the viewpoints of convergence speed, number of iterations, and

TABLE III: Our results for Algorithm 1 (SGD, Momentum, AMSGrad, AMSGWDC, AdaBelief, and MAdam) for constrained finite-sum and online optimization (Note that we give ϵ-approximations for the previous studies shown in Table I.)

		Constant learning rate rule			Diminishing learning rate rule		
		Upper bound	ϵ-approximation		Convergence	ϵ-approximation	
			(i) Iteration	(ii) SGC	rate	(i) Iteration	(ii) SGC
Finite	SGD	$\mathcal{O}\left(\frac{1}{k}\right)+\epsilon^2$	$\left\lfloor\frac{3}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\min\left\{n+\frac{n}{\epsilon^4},\frac{1}{\epsilon^4}\right\}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor\frac{1}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\min\left\{n+\frac{n}{\epsilon^4},\frac{1}{\epsilon^4}\right\}\right)$
-sum	Momentum	$\mathcal{O}\left(\frac{1}{k}\right)+\epsilon^2$	$\left\lfloor\frac{3}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\min\left\{n+\frac{n}{\epsilon^4},\frac{1}{\epsilon^4}\right\}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor\frac{1}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\min\left\{n+\frac{n}{\epsilon^4},\frac{1}{\epsilon^4}\right\}\right)$
	AMSGrad	$\mathcal{O}\left(\frac{1}{k}\right)+\epsilon^2$	$\left\lfloor\frac{3}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\min\left\{n+\frac{n}{\epsilon^4},\frac{1}{\epsilon^4}\right\}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor\frac{1}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\min\left\{n+\frac{n}{\epsilon^4},\frac{1}{\epsilon^4}\right\}\right)$
	AMSGWDC	$\mathcal{O}\left(\frac{1}{k}\right)+\epsilon^2$	$\left\lfloor\frac{3}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\min\left\{n+\frac{n}{\epsilon^4},\frac{1}{\epsilon^4}\right\}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor\frac{1}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\min\left\{n+\frac{n}{\epsilon^4},\frac{1}{\epsilon^4}\right\}\right)$
	AdaBelief	$\mathcal{O}\left(\frac{1}{k}\right)+\epsilon^2$	$\left\lfloor\frac{3}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\min\left\{n+\frac{n}{\epsilon^4},\frac{1}{\epsilon^4}\right\}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor\frac{1}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\min\left\{n+\frac{n}{\epsilon^4},\frac{1}{\epsilon^4}\right\}\right)$
	MAdam	$\mathcal{O}\left(\frac{1}{k}\right)+\epsilon^2$	$\left\lfloor\frac{3}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\min\left\{n+\frac{n}{\epsilon^4},\frac{1}{\epsilon^4}\right\}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor\frac{1}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\min\left\{n+\frac{n}{\epsilon^4},\frac{1}{\epsilon^4}\right\}\right)$
Online	SGD	$\mathcal{O}\left(\frac{1}{k}\right)+\epsilon^2$	$\left\lfloor\frac{3}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor\frac{1}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$
	Momentum	$\mathcal{O}\left(\frac{1}{k}\right)+\epsilon^2$	$\left\lfloor\frac{3}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor\frac{1}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$
	AMSGrad	$\mathcal{O}\left(\frac{1}{k}\right)+\epsilon^2$	$\left\lfloor\frac{3}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor\frac{1}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$
	AMSGWDC	$\mathcal{O}\left(\frac{1}{k}\right)+\epsilon^2$	$\left\lfloor\frac{3}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor\frac{1}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$
	AdaBelief	$\mathcal{O}\left(\frac{1}{k}\right)+\epsilon^2$	$\left\lfloor\frac{3}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor\frac{1}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$
	MAdam	$\mathcal{O}\left(\frac{1}{k}\right)+\epsilon^2$	$\left\lfloor\frac{3}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\left\lfloor\frac{1}{\epsilon^4}\right\rfloor+1$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$

Let n be the total number of samples and $\epsilon>0$. The upper bound and convergence rate are measured by $\min_{j\in[k]}\mathbb{E}[\langle \boldsymbol{x}_j-\boldsymbol{x},\nabla f(\boldsymbol{x}_j)\rangle]$ ($\boldsymbol{x}\in X$), and the ϵ-approximation is measured by $(1/K)\sum_{k=1}^{K}\mathbb{E}[\langle \boldsymbol{x}_k-\boldsymbol{x},\nabla f(\boldsymbol{x}_k)\rangle]\leq\epsilon^2$ ($\boldsymbol{x}\in X$).

SGC to achieve an ϵ-approximation of Algorithm 1. See Table III for the results of Algorithm 1 with $\eta=1/2$.

Numerical evaluations [7], [8] have precisely clarified the relationship between the batch size and the number of iterations needed for ϵ-approximations of ALROAs (Algorithm 1) and the relationship between ϵ and the number of iterations needed for ϵ-approximations of ALROAs. Table III shows that Algorithm 1 has the property that the smaller ϵ is, the larger K becomes. The useful numerical results in [8, Figure 2] support this theoretical result. Moreover, the numerical results in [7], [8] show that increasing the batch size tends to decrease the number of iterations K needed to achieve an ϵ-approximation, but there are diminishing returns whereby increasing the batch size beyond a certain point does not change K [7, Figure 8], [8, Figure 4]. As can be seen from Theorems IV.1 and IV.2 (see also Table III), the number of iterations needed for ϵ-approximation for finite-sum optimization is

$$K=\begin{cases}\text{[Constant learning rate rule]}\\ \left\lfloor\frac{3}{\epsilon^4}\right\rfloor+1\\ \text{[Diminishing learning rate rule]}\\ \left\lfloor\frac{1}{\epsilon^4}\right\rfloor+1\end{cases}$$

and the mini-batch size is

$$s=\begin{cases}\text{[Constant learning rate rule]}\\ \min\left\{n,\frac{c}{\tilde{B}^2\tilde{M}^2},\frac{c}{\sqrt{dD}\tilde{M}},\frac{4(1-b)^2(1-\gamma)^2}{3cdBD}\right\}\\ \text{[Diminishing learning rate rule]}\\ \min\left\{n,\frac{c}{\tilde{B}^2\tilde{M}^2},\frac{c}{\sqrt{dD}\tilde{M}},\frac{4(1-\eta)(1-b)^2(1-\gamma)^2}{9cdBD}\right\}.\end{cases}$$

For both the constant and diminishing learning rate rules, K does not depend on any parameters of the mini-batch size. Hence, we cannot conclude that increasing the batch size decreases K, as shown in [7, Figure 8], [8, Figure 4].

V. Conclusion

We studied ϵ-approximation of ALROAs (Algorithm 1) for constrained finite-sum and online optimization in deep neural networks. Using both constant and diminishing learning rates enables Algorithm 1 to achieve ϵ-approximations. In particular, Algorithm 1 with constant learning rates has an upper bound $\mathcal{O}(1/k)+\epsilon^2$ of the expectation of the variational inequality, while Algorithm 1 with diminishing learning rates has an $\mathcal{O}(1/\sqrt{k})$ convergence rate for the expectation of the variational inequality. The number of iterations needed for an ϵ-approximation of Algorithm 1 with constant learning rates was $\mathcal{O}(\lfloor 1/\epsilon^4\rfloor)+1$, while the number of iterations needed for an ϵ-approximation of Algorithm 1 with diminishing learning rates was $\lfloor 1/\epsilon^4\rfloor+1$. The SGC of Algorithm 1 for finite-sum optimization was $\mathcal{O}(\min\{n+n/\epsilon^4,1/\epsilon^4\})$. Meanwhile, the SGC of Algorithm 1 for online optimization was $\mathcal{O}(1/\epsilon^4)$. The numerical results in [7], [8] are helpful to support our ϵ-approximation analyses for ALROAs.

VI. Acknowledgment

I am sincerely grateful to Editor-in-Chief Yongduan Song, the Associate Editor, and the three anonymous reviewers for helping me improve the original manuscript.

References

[1] L. Jiao, R. Zhang, F. Liu, S. Yang, B. Hou, L. Li, and X. Tang, "New generation deep learning for video object detection: A survey," *IEEE Transactions on Neural Netwroks and Leaning Systems*, 2021.

[2] Y. Yin, D. Xu, X. Wang, and L. Zhang, "Directional deep embedding and appearance learning for fast video object segmentation," *IEEE Transactions on Neural Netwroks and Leaning Systems*, 2021.

[3] K. Muhammad, S. Khan, J. Del Ser, and V. Hugo C. de Albuquerque, "Deep learning for multigrade brain tumor classification in smart healthcare systems: A prospective survey," *IEEE Transactions on Neural Netwroks and Leaning Systems*, 2021.

[4] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of Machine Learning Research*, vol. 37, pp. 2048–2057, 2015.

[5] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *preprint*, arXiv:1701.07875 (2017).

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998–6008, 2017.

[7] G. Zhang, L. Li, Z. Nado, J. Martens, S. Sachdeva, G. E. Dahl, C. J. Shallue, and R. Grosse, "Which algorithmic choices matter at which batch sizes? Insights from a noisy quadratic model," in *Proceedings of The International Conference on Neural Information Processing Systems*, pp. 1–12, 2019.

[8] C. J. Shallue, J. Lee, J. Antognini, J. Sohl-Dickstein, R. Frostig, and G. E. Dahl, "Measuring the effects of data parallelism on neural network training," *Journal of Machine Learning Research*, vol. 20, pp. 1–49, 2019.

[9] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 1951.

[10] K. Scaman and C. Malherbe, "Robustness analysis of non-convex stochastic gradient descent using biased expectations," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 1–11, 2020.

[11] H. Chen, L. Zheng, R. A. Kontar, and G. Raskutti, "Stochastic gradient descent in correlated settings: A study on Gaussian processes," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 1–12, 2020.

[12] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, Cambridge, 2016.

[14] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proceedings of The International Conference on Learning Representations*, pp. 1–15, 2015.

[15] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, pp. 1–17, 1964.

[16] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," in *Proceedings of The International Conference on Learning Representations*, pp. 1–23, 2018.

[17] X. Chen, S. Liu, R. Sun, and M. Hong, "On the convergence of a class of Adam-type algorithms for non-convex optimization," in *Proceedings of The International Conference on Learning Representations*, pp. 1–30, 2019.

[18] J. Zhuang, T. Tang, Y. Ding, S. Tatikonda, N. Dvornek, X. Papademetris, and J. S. Duncan, "AdaBelief optimizer: Adapting stepsizes by the belief in observed gradients," in *Proceedings of The International Conference on Neural Information Processing Systems*, pp. 1–29, 2020.

[19] D. Liang, F. Ma, and W. Li, "New gradient-weighted adaptive gradient methods with dynamic constraints," *IEEE Access*, vol. 8, pp. 110929–110942, 2020.

[20] C. Fang, C.-J. Li, Z. Lin, and T. Zhang, "SPIDER: Near-optimal non-convex optimization via stochastic path-integrated differential estimator," in *Advances in Neural Information Processing Systems*, vol. 31, pp. 1–11, 2018.

[21] H. Iiduka, "Appropriate learning rates of adaptive learning rate optimization algorithms for training deep neural networks," *IEEE Transactions on Cybernetics*, 2021.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[23] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$," *Doklady AN USSR*, vol. 269, pp. 543–547, 1983.

[24] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *International Conference on Machine Learning*, pp. 1139–1147, 2013.

[25] J. Martens and R. Grosse, "Optimizing neural networks with Kronecker-factored approximate curvature," in *Proceedings of Machine Learning Research*, vol. 37, pp. 2408–2417, 2015.

[26] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. New York: Springer, 2011.

[27] D. Kinderlehrer and G. Stampacchia, *An Introduction to Variational Inequalities and Their Applications*. Classics Appl. Math. 31, SIAM, 2000.

[28] F. Facchinei and J.-S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems I*. Springer, New York, 2003.

[29] F. Facchinei and J.-S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems II*. Springer, New York, 2003.

[30] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on Optimization*, vol. 19, pp. 1574–1609, 2009.

Hideaki Iiduka received the Ph.D. degree in mathematical and computing science from Tokyo Institute of Technology, Tokyo, Japan, in 2005. From 2005 to 2007, he was a Research Assistant in the Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Tokyo, Japan. From 2007 to 2008, he was a Research Fellow (PD) of the Japan Society for the Promotion of Science. From October 2008 to March 2013, he was an Associate Professor in the Network Design Research Center, Kyushu Institute of Technology, Tokyo, Japan. From April 2013 to March 2019, he was an Associate Professor in the Department of Computer Science, School of Science and Technology, Meiji University, Kanagawa, Japan. Since April 2019, he has been a Professor in the same department. He was awarded the 4th Research Encourage Award of ORSJ in August 2014 and the 9th Research Award of ORSJ in September 2019. His research field is optimization theory and its applications to mathematical information science. He is a Fellow of ORSJ and a member of MOS and SIAM.