Unified Algorithm Framework for Nonconvex Stochastic Optimization in Deep Neural Networks

YINI ZHU¹, HIDEAKI IIDUKA²
¹Computer Science Course, Graduate School of Science and Technology, Meiji University, Kanagawa 214-8571, Japan (e-mail: yini@cs.meiji.ac.jp)
²Department of Computer Science, Meiji University, Kanagawa 214-8571, Japan (e-mail: iiduka@cs.meiji.ac.jp)

Corresponding author: Yini Zhu (e-mail: yini@cs.meiji.ac.jp).

This work was supported by JSPS KAKENHI Grant Number 21K11773.

I. INTRODUCTION

A USEFUL way to train deep neural networks is to solve a nonconvex optimization problem in terms of deep neural networks [1], [2], [3] and find suitable parameters for them. Many algorithms have been presented to solve nonconvex optimization problems. The simplest algorithm for the problem is stochastic gradient descent (SGD) (see, e.g., [4], [5] for recent studies for SGD). Adaptive-learning-rate optimization algorithms (see Subsection 8.5 in [6]) are powerful methods that adapt the learning rates of the model parameters to solve the problem quickly. Examples include Adaptive Gradient (AdaGrad) [7], Root Mean Square Propagation (RMSProp) [6, Algorithm 8.5], Adaptive Moment Estimation (Adam) [8], Adaptive Mean Square Gradient (AMSGGrad) [9], Adam with weighted gradient and dynamic bound of learning rate (GWDC) [2, Algorithm 2], AMSGGrad with weighted gradient and dynamic bound of learning rate (AMSGWDC) [2, Algorithm 3], and Adapting stepsizes by the belief in observed gradients (AdaBelief) [10]. Note that the existing adaptive-learning-rate optimization algorithms use the inverses of certain positive-definite matrices at each iteration; in other words, they are framed depending on the definitions of such positive-definite matrices.

In this paper, we first show that the positive-definite matrices used in the existing adaptive-learning-rate optimization algorithms assume common conditions (Assumption III.1). Next, we present an algorithm [11] (Algorithm 1) for which convergence is guaranteed under Assumption III.1. This implies that the algorithm is a unification of the existing ones (see also Example III.1). For the convergence analyses of the
TABLE 1. Convergence rates for SGD and adaptive-learning-rate optimization algorithms (GWDC, AMSGWDC, AdaBelief, and its unified algorithm) on recent papers published in 2020 and 2021

| Algorithm [Ref.] (Authors; year) | Nonconvex optimization | Convex optimization |
|---------------------------------|------------------------|---------------------|
|                                 | Constant learning rate | Diminishing learning rate | Constant learning rate | Diminishing learning rate |
| SGM [4] (Scaman et al.; 2020)   | $O\left(\frac{1}{n}\right) + C$ | $O\left(\frac{1}{\sqrt{n}}\right)$ | $O\left(\frac{1}{n}\right) + C$ | $O\left(\frac{1}{\sqrt{n}}\right)$ |
| SGM [5] (Loizou et al.; 2021)   | $O\left(\frac{1}{n}\right)$ | $O\left(\frac{1}{n}\right)$ | $O\left(\frac{1}{n}\right)$ | $O\left(\frac{1}{n}\right)$ |
| GWDC [2] (Liang et al.; 2020)   | $O\left(\frac{1}{n}\right) + C$ | $O\left(\frac{1}{\sqrt{n}}\right)$ | $O\left(\frac{1}{n}\right) + C$ | $O\left(\frac{1}{\sqrt{n}}\right)$ |
| AMSGWDC [2] (Liang et al.; 2020) | $O\left(\frac{1}{n}\right)$ | $O\left(\frac{1}{n}\right)$ | $O\left(\frac{1}{n}\right)$ | $O\left(\frac{1}{n}\right)$ |
| AdaBelief [10] (Zhuang et al.; 2020) | $O\left(\frac{1}{n}\right)$ | $O\left(\frac{1}{n}\right)$ | $O\left(\frac{1}{n}\right)$ | $O\left(\frac{1}{n}\right)$ |
| Algorithm 1 [11] (Idika; 2021) | $O\left(\frac{1}{n}\right)$ | $O\left(\frac{1}{n}\right)$ | $O\left(\frac{1}{n}\right)$ | $O\left(\frac{1}{n}\right)$ |

The convergence rate of nonconvex (resp. convex) optimization is measured by $\min_{k=1,2,\ldots,n} E[\|\nabla f(x_k)\|^2]$ (resp. $R(T)/T$), where $n$ denotes the number of iterations and $T$ denotes the number of training samples (see Table 2 for the definitions of the notation). $C$ and $C_i (i = 1, 2)$ are constants that are independent of $n$, $T$, and the constant learning rates $\alpha$ and $\beta$. A diminishing learning rate is $\alpha_n = 1/\sqrt{n}$.

algorithm, we use constant and diminishing learning rates. We show that, for a constant learning rate, the algorithm can approximate a stationary point of the nonconvex stochastic optimization problem (Theorem III.1), while for a diminishing rate, it converges to a stationary point of the problem (Theorem III.2). Table 1 summarizes convergence rate results of SGD and adaptive-learning-rate optimization algorithms for nonconvex and convex learning that were studied in 2020 and 2021.

Our first contribution is to provide convergence analyses of the existing adaptive-learning-rate optimization algorithms, i.e., analyses which could not be done using the methods in [2], [4], [5], [8], [9], [10]. The previously reported results (see, e.g., [2], [8], [9]) tried to minimize the regret (see Table 2 for the definition) for convex optimization. However, regret minimization does not always lead to solutions of optimization problems in deep learning (Subsection IV-E). In contrast to the previous results, this paper explicitly shows that the existing adaptive-learning-rate optimization algorithms can solve such problems (Subsections IV-A–IV-D). For nonconvex optimization, AdaBelief [10] has an $O(\log n/\sqrt{n})$ convergence rate, where $n$ denotes the number of iterations (see Table 1). Theorem III.2 ensures that using a diminishing learning rate $\alpha_n = 1/\sqrt{n}$ allows the proposed algorithm (Algorithm 1) including AdaBelief to have an $O(1/\sqrt{n})$ convergence rate (see also [11]). While GWDC and AMSGWDC [2] with diminishing learning rates can only be applied to convex optimization (see Table 1), the proposed algorithm (Algorithm 1) can be applied to both convex and nonconvex optimization (see Table 1).

In particular, we would like to emphasize that the existing algorithms with constant learning rates can be applied to the problem (Subsection IV-E and Table 1). The results for a constant learning rate are significant from the viewpoints of both theory and practice, since algorithms with a constant learning rate work well whereas the algorithms with a diminishing rate converging to zero do not work. Moreover, we also would like to emphasize that not only Adam and AMSGGrad but also GWDC and AMSGWDC can be applied to the problem. This is in contrast to [2], which presented a regret minimization only for GWDC and AMSGWDC with a diminishing learning rate, and [11], which presented convergence analyses only for Adam and AMSGGrad for constant and diminishing learning rates. Using constant learning rates allows Algorithm 1 for nonconvex optimization to have approximately an $O(1/n)$ convergence rate (see Table 1).

The second contribution of this paper is to show that the algorithm with a constant learning rate tends to be superior for training neural networks, while the one with a diminishing rate tends not to be good for training neural networks. Here, we focus on image classification using several neural networks and show the effectiveness of the algorithm with a constant learning rate (Subsections V-A–V-B). Moreover, we consider unconstrained optimization in [12] and provide numerical comparisons for Algorithm 1 with heuristic intelligent optimization algorithms, such as the genetic algorithm (GA), particle swarm optimization (PSO) biogeography-based optimization (BBO), an Atom search optimization (ASO), and a teaching-learning-based optimization (TLBO). The numerical comparisons show that, in particular, TLBO and Algorithm 1 perform well (Subsection V-C).

This paper is organized as follows. Section II states the main problem. Section III presents the proposed algorithm for solving the main problem and analyzes its convergence. Section IV compares the analyses in Section III with the ones in the previous reports. Section V numerically compares the behaviors of the proposed algorithm with those of the existing ones. Section VI concludes the paper with a brief summary.

II. OPTIMIZATION IN DEEP NEURAL NETWORKS

The notation used in this paper is summarized in Table 2.

In general, an optimization problem in a deep neural network can be expressed as the following nonconvex optimization problem:

Problem II.1 Assume that

(A1) $X \subset \mathbb{R}^d$ is a nonempty, closed convex set onto which
The optimal objective function value for the problem of minimizing

\( F(x) = \min_{x \in \mathbb{R}^d} f(x) \),

where \( f : \mathbb{R}^d \rightarrow \mathbb{R} \) is continuously differentiable for almost every \( x \in \mathbb{R}^d \), defines the metric projection onto a nonempty, closed convex set

\[ \text{argmin}_{x \in \mathbb{R}^d} f(x) \]

\( x^* \) is supported on a set \( \mathbb{R}^d \) where \( f \) is defined, where \( f(x) := \mathbb{E}[F(x, \xi)] \) for all \( x \in \mathbb{R}^d \).

\( \nabla f \) is the gradient of \( f \).

\( G(x, \xi) \) is the stochastic gradient for a given \( (x, \xi) \in \mathbb{R}^d \times \mathbb{R}^d \) which satisfies \( \mathbb{E}[G(x, \xi)] = \nabla f(x) \).

\( X^* \) is the set of stationary points of the problem of minimizing \( f \) over \( X \).

\( f^* \) is the optimal objective function value for the problem of minimizing \( f \) over \( X \).

We will consider Problem II.2 under the following conditions:

(C1) There is an independent and identically distributed sample \( \xi_0, \xi_1, \ldots \) of realizations of the random vector \( \xi \);

(C2) There is an oracle which, for a given input point \( (x, \xi) \in \mathbb{R}^d \times \mathbb{R}^d \), returns a stochastic gradient \( G(x, \xi) \) such that \( \mathbb{E}[G(x, \xi)] = \nabla f(x) \);

(C3) There exists a positive number \( M \) such that, for all \( x \in X \), \( \mathbb{E}[\|G(x, \xi)\|^2] \leq M^2 \).

### III. ADAPTIVE-LEARNING-RATE OPTIMIZATION ALGORITHM

Algorithm 1 [11] is a unified algorithm for the existing adaptive-learning-rate optimization algorithms to solve Problem II.2 under (C1)–(C3).

**Algorithm 1** Adaptive-learning-rate optimization algorithm for Problem II.2

**Require:** \((\alpha_n)_{n \in \mathbb{N}} \subset (0, 1), (\beta_n)_{n \in \mathbb{N}} \subset [0, 1), \gamma \in [0, 1)\)

1. \( n \leftarrow 0, x_0, m_{-1} \in \mathbb{R}^d, H_0 \in S^d_+ \cap \mathbb{R}^d \)
2. **loop**
3. \( m_n := \beta_n m_{n-1} + (1 - \beta_n) G(x_n, \xi_n) \)
4. \( \bar{m}_n := \frac{1 - \gamma^{n+1}}{1 - \gamma} \)
5. \( H_n \in S^d_+ \cap \mathbb{R}^d \)
6. Find \( d_n \in \mathbb{R}^d \) that solves \( H_n d = -\bar{m}_n \)
7. \( x_{n+1} := P_{X, H_n}(x_n + \alpha_n d_n) \)
8. \( n \leftarrow n + 1 \)
9. **end loop**

We need the following conditions to analyze Algorithm 1.
Assumption III.1 The sequence \( (H_n)_{n \in \mathbb{N}} \subset \mathbb{S}_+^d \cap \mathbb{D}^d \), denoted by \( H_n := \text{diag}(h_{n,i}) \), in Algorithm 1 satisfies the following conditions:

(A3) \( h_{n+1,i} \geq h_{n,i} \) almost surely for all \( n \in \mathbb{N} \) and all \( i = 1, 2, \ldots, d \);

(A4) For all \( i = 1, 2, \ldots, d \), a positive number \( B_i \) exists such that sup\( \{E[h_{n,i}] : n \in \mathbb{N}\} \leq B_i \).

Moreover,

(A5) \( D := \max_{i=1,2,\ldots,d} \sup \{ (x_i - y_i)^2 : (x_i), (y_i) \in X \} < +\infty \).

Assumption (A5) holds under the boundedness condition of \( X \), which is assumed in Theorem 4.1 in [8], [9], (A5) in [11]. We call Algorithm 1 an adaptive-learning-rate optimization algorithm (see Subsection 8.5 in [6]) since Algorithm 1 under (A3) and (A4) reduces to the previous algorithms that assume (A5).

Example III.1

(i) Adam [8]: Let us consider \( H_n \) and \( v_n \) \((n \in \mathbb{N})\), which are defined for all \( n \in \mathbb{N} \) by

\[
\begin{align*}
&v_n := \delta v_{n-1} + (1 - \delta)G(x_n, \xi_n) \odot G(x_n, \xi_n), \\
&\hat{v}_n := \frac{v_n}{1 - \delta^{n+1}}, \\
&\check{v}_n := (\check{v}_{n,i}) := \left( \max\{\hat{v}_{n-1,i}, \hat{v}_{n,i}\} \right), \\
&H_n := \text{diag} \left( \sqrt{\check{v}_{n,i}} \right),
\end{align*}
\]

where \( v_{-1} = \hat{v}_{-1} = 0 \in \mathbb{R}^d \) and \( \delta \in [0, 1) \). From Example III.1(ii) and (iii), \( H_n \) and \( v_n \), which are defined by (1), satisfy (A3) and (A4) (see Section III in [11]). Hence, Algorithm 1 with (1) is based on the Adam algorithm.

(ii) AMSGrad [9]: Let us consider \( H_n \) and \( v_n \) \((n \in \mathbb{N})\), which is defined for all \( n \in \mathbb{N} \) by

\[
\begin{align*}
&v_n := \delta v_{n-1} + (1 - \delta)G(x_n, \xi_n) \odot G(x_n, \xi_n), \\
&\hat{v}_n := \frac{v_n}{1 - \delta^{n+1}}, \\
&\check{v}_n := (\check{v}_{n,i}) := \left( \max\{\hat{v}_{n-1,i}, \hat{v}_{n,i}\} \right), \\
&H_n := \text{diag} \left( \sqrt{\check{v}_{n,i}} \right),
\end{align*}
\]

where \( v_{-1} = \hat{v}_{-1} = 0 \in \mathbb{R}^d \) and \( \delta \in [0, 1) \). Algorithm 1 with (2) is the AMSGrad algorithm [9].

(iii) GWDC [2]: Suppose that \( (t_n)_{n \in \mathbb{N}} \subset \mathbb{R}^{++} \) is monotone increasing and \( (u_n)_{n \in \mathbb{N}} \subset \mathbb{R}^{++} \) is monotone decreasing with \( t_n \leq u_n \) for all \( n \in \mathbb{N} \). Let us consider \( H_n \) and \( v_n \) \((n \in \mathbb{N})\), which is defined for all \( n \in \mathbb{N} \) by

\[
\begin{align*}
&v_n := \delta v_{n-1} + (1 - \delta)G(x_n, \xi_n) \odot G(x_n, \xi_n), \\
&\hat{v}_n := \frac{v_n}{1 - \delta^{n+1}}, \\
&\check{v}_n := (\check{v}_{n,i}) := \left( \text{Clip} \left( \frac{1}{\sqrt{u_{n,i}}}, l_n, u_n \right)^{-1} \right),
\end{align*}
\]

where \( \gamma \) is defined as in (A5), \( B := \max \{\|m - 1\|_2, M^2\} \), and \( D \) is defined as in (A5). The following proposition (see also [11], Proposition 1) enables us to compare Algorithm 1 with Adam [8], AMSGrad [9], GWDC [2], and AMSSGDC [2].

### Proposition III.1

Suppose that (A1)–(A5) and (C1)–(C3) hold, \( F(\cdot, \xi) \) is convex for almost every \( \xi \in \Xi \), and \( (x_n)_{n \in \mathbb{N}} \)
is the sequence generated by Algorithm 1 with \( \alpha_n := \alpha \) and \( \beta_n := \beta \ (n \in \mathbb{N}) \). Then,
\[
\lim_{n \to \infty} \mathbb{E} \left[ f(x_n) - f^* \right] \leq \frac{\bar{B}^2 M^2}{\bar{b} \gamma^2} + \frac{M \sqrt{\frac{\bar{D} d}{b \gamma}}}{\beta},
\]
where \( f^* \) denotes the optimal value of Problem II.1 (see (F2)), and \( \gamma, \bar{b}, \bar{M}, D, \) and \( \bar{B} \) are defined as in Theorem III.1. Additionally, for the regret on a sequence of \( F(\cdot, t) := f_t(\cdot) \) \( (t = 1, 2, \ldots, T) \), Algorithm 1 satisfies
\[
\frac{R(T)}{T} \leq O \left( \frac{1}{T} \right) + C_1 \alpha + C_2 \beta.
\]

**B. CONVERGENCE ANALYSIS OF ALGORITHM 1 WITH A DIMINISHING LEARNING RATE**

The following is a convergence analysis of Algorithm 1 with a diminishing sub-learning rate. The proof of Theorem III.2 is given in the proof of Theorem 2 in [11].

**Theorem III.2** Suppose that (A1)–(A5) and (C1)–(C3) hold and \((x_n)_{n \in \mathbb{N}}\) is the sequence generated by Algorithm 1 with \( \alpha_n \) and \( \beta_n \ (n \in \mathbb{N}) \) satisfying \( \sum_{n=0}^{t} \alpha_n = +\infty \), \( \sum_{n=0}^{t} \alpha_n^2 < +\infty \), and \( \sum_{n=0}^{t} \alpha_n \beta_n < +\infty \). Then, for all \( x \in \mathbb{X} \),
\[
\limsup_{n \to \infty} \mathbb{E} \left[ \langle x - x_n, \nabla f(x_n) \rangle \right] \geq 0.
\]
Moreover, if \( \alpha_n := 1/n^\eta \ (\eta \in [1/2, 1)) \) and if \( \beta_n := \lambda^\eta \ (\lambda \in (0, 1)) \), then Algorithm 1 achieves the following convergence rate:
\[
\max_{k=1,2,\ldots,n} \mathbb{E} \left[ \langle x - x_k, \nabla f(x_k) \rangle \right] \geq -O \left( \frac{1}{n^{1-\eta}} \right).
\]

The following proposition (see also [11, Proposition 2]) enables us to compare Algorithm 1 with Adam [8], AMSGrad [9], GWDC [2], and AMSGWDC [2].

**Proposition III.2** Suppose that (A1)–(A5) and (C1)–(C3) hold, \( F(\cdot, \xi) \) is convex for almost every \( \xi \in \Xi \), and \((x_n)_{n \in \mathbb{N}}\) is the sequence generated by Algorithm 1 with \( \alpha_n := 1/n^\eta \) and \( \beta_n := \lambda^\eta \ (n \in \mathbb{N}) \), where \( \eta \in [1/2, 1] \) and \( \lambda \in (0, 1) \). Then, under \( \eta \in (1/2, 1] \),
\[
\liminf_{t \to +\infty} \mathbb{E} \left[ f(x_n) - f^* \right] = 0,
\]
where \( f^* \) denotes the optimal value of the problem of minimizing \( f \) over \( X \). Moreover, under \( \eta \in [1/2, 1) \), any accumulation point of \((\tilde{x}_n)_{n \in \mathbb{N}}\) defined by \( \tilde{x}_n := (1/n) \sum_{k=1}^{n} x_k \) almost surely belongs to \( X^* := \arg \min_{x \in X} f(x) \), and Algorithm 1 achieves the following convergence rate:
\[
\mathbb{E} \left[ f(\tilde{x}_n) - f^* \right] = O \left( \frac{1}{n^{1-\eta}} \right).
\]
Additionally, for the regret on a sequence of \( F(\cdot, t) := f_t(\cdot) \) \( (t = 1, 2, \ldots, T) \), Algorithm 1 satisfies
\[
\frac{R(T)}{T} \leq O \left( \frac{1}{T^{1-\eta}} \right).
\]

## IV. COMPARISON OF OUR CONVERGENCE ANALYSES WITH CONVERGENCE ANALYSES ON EXISTING ALGORITHMS

To compare the previously reported results in [2], [8], [9] with the results in Section III, we will consider Problem II.1 when \( f_t(\cdot) := F(\cdot, t) \) \( (t = 1, 2, \ldots, T) \) is convex (see, e.g., [2, p.110932]), i.e.,
\[
\text{Find } x^* \in \arg \min_{x \in \mathbb{X}} \sum_{t=1}^{T} f_t(x).
\]

See [11, Table II] and Table 1 for comparisons of SGD [4], AMSGrad [13], AdaBelief [10], and Algorithm 1 for nonconvex optimization. Since \( f \) is convex, (F2) implies that \( X^* = \arg \min_{x \in \mathbb{X}} \sum_{t=1}^{T} f_t(x) \). The performance measure used for the previously reported results in [2], [8], [9] is the regret \( R(T) \).

**A. COMPARISON OF ADAM WITH ALGORITHM 1 IN THE CASE OF EXAMPLE III.1(I)**

Theorem 4.1 in [8] indicates that Adam with \( \alpha_n = O(1/\sqrt{t}) \) and \( \beta_n = \lambda^\eta \ (\lambda \in (0, 1)) \) ensures that there exists a positive real number \( D \) such that
\[
\frac{R(T)}{T} = \frac{1}{T} \sum_{t=1}^{T} f_t(x_t) - f^* \leq \frac{D}{\sqrt{T}}.
\]

Unfortunately, Theorem 1 in [9] shows that a counterexample to Theorem 4.1 in [8] exists. Meanwhile, Algorithm 1 with (1) resembles the Adam algorithm (see Footnote 1 for details). Proposition III.2 indicates that Algorithm 1 with (1), \( \eta = 1/2 \), and \( \beta_n = \lambda^\eta \) satisfies
\[
\mathbb{E} \left[ f \left( \frac{1}{T} \sum_{t=1}^{T} x_t \right) - f^* \right] = O \left( \frac{1}{\sqrt{T}} \right),
\]
\[
\frac{R(T)}{T} \leq O \left( \frac{1}{\sqrt{T}} \right).
\]

This implies that Algorithm 1 based on Adam achieves an \( O(1/\sqrt{T}) \) convergence rate, which could not be done using the analysis in [8].

**B. COMPARISON OF AMSGRAD WITH ALGORITHM 1 IN THE CASE OF EXAMPLE III.1(II)**

Theorem 4 in [9] indicates that AMSGrad with \( \alpha_n = O(1/\sqrt{t}) \) and \( \beta_n = \lambda^\eta \ (\lambda \in (0, 1)) \) ensures that there exists a positive real number \( D \) such that
\[
\frac{R(T)}{T} = \frac{1}{T} \sum_{t=1}^{T} f_t(x_t) - f^* \leq \frac{D \sqrt{1 + \ln T}}{T}.
\]
Example III.1(ii) shows that Algorithm 1 with (2) coincides with AMSGrad. Proposition III.2 indicates that Algorithm 1 with (2), \( \eta = 1/2 \) and \( \beta_n = \lambda^n \) (i.e., AMSGrad) satisfies
\[
\mathbb{E} \left[ f \left( \frac{1}{T} \sum_{t=1}^{T} x_t \right) - f^* \right] = \mathcal{O} \left( \frac{1}{\sqrt{T}} \right),
\]
\[
\frac{R(T)}{T} \leq \mathcal{O} \left( \frac{1}{\sqrt{T}} \right).
\]
This implies that AMSGrad can achieve an \( \mathcal{O}(1/\sqrt{T}) \) convergence rate, which is better than \( \mathcal{O}(\sqrt{(1 + \ln T)/T}) \) (see (7)).

**C. COMPARISON OF GWDC WITH ALGORITHM 1 IN THE CASE OF EXAMPLE III.1(iii)**

Inequality (15) in [2] indicates that GWDC with \( \alpha_n = O(1/\sqrt{n}) \) and \( \beta_n = \beta e^{-\beta n} (\beta \in \{0.9, 0.99\}, \beta \in (0, 1)) \) ensures that there exists a positive real number \( \bar{D} \) such that
\[
\frac{R(T)}{T} = \frac{1}{T} \sum_{t=1}^{T} f_t(x_t) - f^* \leq \frac{\bar{D}}{\sqrt{T}}.
\]

Example III.1(iii) shows that Algorithm 1 with (3) coincides with GWDC. Proposition III.2 indicates that Algorithm 1 with (3), \( \eta = 1/2 \), and \( \beta_n = \beta e^{-\beta n} \) (i.e., GWDC) satisfies
\[
\mathbb{E} \left[ f \left( \frac{1}{T} \sum_{t=1}^{T} x_t \right) - f^* \right] = \mathcal{O} \left( \frac{1}{\sqrt{T}} \right),
\]
\[
\frac{R(T)}{T} \leq \mathcal{O} \left( \frac{1}{\sqrt{T}} \right).
\]

**D. COMPARISON OF AMSGrad WITH ALGORITHM 1 IN THE CASE OF EXAMPLE III.1(iv)**

Appendix B in [2] indicates that AMSGrad with \( \alpha_n = O(1/\sqrt{n}) \) and \( \beta_n = \beta e^{-\beta n} (\beta \in \{0.9, 0.99\}, \beta \in (0, 1)) \) ensures that there exists a positive real number \( \bar{D} \) such that
\[
\frac{R(T)}{T} = \frac{1}{T} \sum_{t=1}^{T} f_t(x_t) - f^* \leq \frac{\bar{D}}{\sqrt{T}}.
\]

Example III.1(iv) shows that Algorithm 1 with (4) coincides with AMSGrad. Proposition III.2 indicates that Algorithm 1 with (4), \( \eta = 1/2 \), and \( \beta_n = \beta e^{-\beta n} \) (i.e., AMSGrad) satisfies
\[
\mathbb{E} \left[ f \left( \frac{1}{T} \sum_{t=1}^{T} x_t \right) - f^* \right] = \mathcal{O} \left( \frac{1}{\sqrt{T}} \right),
\]
\[
\frac{R(T)}{T} \leq \mathcal{O} \left( \frac{1}{\sqrt{T}} \right).
\]

**E. DISCUSSION**

First of all, we would like to emphasize that regret minimization does not always lead to solutions of problem (5) (see also [14]). This is because, even if \( (x_t)_{t=1}^{T} \) is satisfied, for a sufficiently large number \( T \),
\[
\frac{R(T)}{T} = \frac{1}{T} \sum_{t=1}^{T} f_t(x_t) - f^* \approx 0,
\]
we do not have that
\[
f(x_T) - f^* = \frac{1}{T} \sum_{t=1}^{T} f_t(x_T) - f^* \approx 0.
\]

Accordingly, from only (6), (7), (8), and (9), we cannot evaluate whether or not the output \( x_T \) generated by the existing adaptive-learning-rate optimization algorithms approximates the solution of problem (5). Meanwhile, Proposition III.2 and Subsections IV-A, IV-B, IV-C, and IV-D lead to the finding that Algorithm 1, including Adam, AMSGrad, GWDC, and AMSGWDC, ensures that, under \( \eta \in (1/2, 1] \),
\[
\liminf_{n \to +\infty} \mathbb{E} \left[ \sum_{t=1}^{T} f_t(x_n) - f^* \right] = 0 \tag{10}
\]
and that, under \( \eta \in (1/2, 1] \),
\[
\mathbb{E} \left[ \sum_{t=1}^{T} f_t \left( \frac{1}{T} \sum_{t=1}^{T} x_t \right) - f^* \right] = \mathcal{O} \left( \frac{1}{\sqrt{T}} \right). \tag{11}
\]

The results in (10) and (11) imply that Algorithm 1 can solve problem (5), in contrast to (6), (7), (8), and (9) that are results for regret minimization.

Moreover, we would like to emphasize that Algorithm 1 with a constant learning rate can approximate the solution of problem (5) in the sense of the result in Proposition III.1, i.e.,
\[
\liminf_{n \to +\infty} \mathbb{E} \left[ \sum_{t=1}^{T} f_t(x_n) - f^* \right] \leq T \left( \frac{\bar{D}^2 \bar{M}^2}{2b_\gamma^2} \alpha + \frac{\bar{M} \sqrt{D\bar{d}}}{b_\gamma} \beta \right).
\]

The above result indicates that using small constant learning rates would be a good way to solve problem (5). Proposition III.2 shows that any accumulation point of \( \bar{x}_n := (1/n) \sum_{k=1}^{n} x_k \) belongs to \( \arg\min_{x \in X} \sum_{t=1}^{T} f_t(x) \). According to, using a diminishing learning rate is a more robust way to solve problem (5) than using a constant learning rate in theory, as shown in [11], [14], [15]. However, Algorithm 1 with a diminishing learning rate might not work for a rather large number \( N \) of iterations, since step 7 in Algorithm 1 with \( \alpha N \approx 0 \) satisfies
\[
x_{N+1} = P_{X,H_N}(x_N + \alpha_N d_N) \approx P_{X,H_N}(x_N) = x_N.
\]
This implies that using a diminishing learning rate would not be good in practice. The next section shows that using a constant learning rate tends to be superior to using a diminishing one. This tendency was also observed in [11], [14], [15].

**V. NUMERICAL EXPERIMENTS**

We examined the behavior of Algorithm 1 with different learning rates. The adaptive-learning-rate optimization algorithms with \( \delta = 0.999 \) [8], [9] used in the experiments were as follows, where the initial points initialized automatically by PyTorch were used and \( l_n \) and \( u_n \) were based on [16, Section 4].

Algorithm 1 with constant learning rates:
ADAM-C1: Algorithm 1 with (1), $\gamma = 0.9, \alpha_n = 10^{-3}$, and $\beta_n = 0.9$
ADAM-C2: Algorithm 1 with (1), $\gamma = 0.9, \alpha_n = 10^{-3}$, and $\beta_n = 10^{-2}$
ADAM-C3: Algorithm 1 with (1), $\gamma = 0.9, \alpha_n = 10^{-2}$, and $\beta_n = 10^{-2}$
AMSG-C1: Algorithm 1 with (2), $\gamma = 0, \alpha_n = 10^{-3}$, and $\beta_n = 0.9$
AMSG-C2: Algorithm 1 with (2), $\gamma = 0, \alpha_n = 10^{-3}$, and $\beta_n = 10^{-3}$
AMSG-C3: Algorithm 1 with (2), $\gamma = 0, \alpha_n = 10^{-2}$, and $\beta_n = 10^{-2}$
GWDC-C1: Algorithm 1 with (3), $\gamma = 0.9, \alpha_n = 0.1 - 0.1/((1 - \delta)n + 1)$, $\beta_n = 0.1 + 0.1/(1 - \delta)n$, and $\beta_n = 10^{-3}$
GWDC-C2: Algorithm 1 with (3), $\gamma = 0.9, \alpha_n = 0.1 - 0.1/((1 - \delta)n + 1)$, $\beta_n = 0.1 + 0.1/(1 - \delta)n$, and $\beta_n = 10^{-3}$
GWDC-C3: Algorithm 1 with (3), $\gamma = 0.9, \alpha_n = 0.1 - 0.1/((1 - \delta)n + 1)$, $\beta_n = 0.1 + 0.1/(1 - \delta)n$, and $\beta_n = 10^{-2}$
AMSGGWDC-C1: Algorithm 1 with (4), $\gamma = 0, \alpha_n = 10^{-3}$, and $\beta_n = 0.9$
AMSGGWDC-C2: Algorithm 1 with (4), $\gamma = 0, \alpha_n = 10^{-3}$, and $\beta_n = 10^{-2}$
AMSGGWDC-C3: Algorithm 1 with (4), $\gamma = 0, \alpha_n = 10^{-2}$, and $\beta_n = 10^{-2}$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3120749, IEEE Access

A. FEEDFORWARD NEURAL NETWORK MODELS

First, we experimented with two feedforward neural network models, a single-layer perceptron and a double-layer perceptron, having different hidden layers for image classification. We used the MNIST dataset, which is a multi-class handwritten digits dataset (0–9), collected by the National Institute of Standards and Technology (NIST). The training data were gathered from Census Bureau employees and high-school students. The training data contains 60,000 grayscale images ($28 \times 28$), and the test data contains 10,000 grayscale images ($28 \times 28$).

1) Single-layer perceptron

Figures 1 and 2 indicate that Algorithm 1 with a constant learning rate tended to perform better than Algorithm 1 with a diminishing one in terms of training and test accuracy score. Moreover, Figure 2 shows that ADAM-C1 and ADAM-C2 minimized the training loss function faster than the other algorithms. Meanwhile, Figure 4 shows that GWDC-C1, AMSGWDC-C1, and AMSGWDC-C2 minimized the test loss function faster than the other algorithms. Figures 4 and 8 also show that the algorithm with a constant learning rate outperformed the one with a diminishing rate.

2) Double-layer perceptron

Figures 9,11 and Figures 13,15 indicate that Algorithm 1 with a constant learning rate tended to perform better than Algorithm 1 with a diminishing one in terms of training and test accuracy score. Figure 10 shows that AMSG-C1 and AMSG-C2 minimized the training loss function faster than the other...
algorithms. On the other hand, Figure 12 shows that GWDC-C1, GWDC-C2, and AMSGWDC-C2 performed well. We can see that GWDC and AMGWDC with constant learning rates were superior at training the neural network. Figures 12 and 16 also show that the algorithm with a constant learning rate outperformed the one with a diminishing rate.

**B. CONVOLUTIONAL NEURAL NETWORK MODELS**

Next, we experimented with a dense convolutional network (DenseNet) and residual network (ResNet); both are relatively deep models based on convolution neural networks (CNN) for image classification. We used the CIFAR-10 dataset, which is a benchmark for image classification. The dataset is a collection of color image data collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. It includes ten categories of images, each of which has 6,000 color images ($32 \times 32$). There are 50,000 training images and 10,000 test images.

1) DenseNet

Figures 17, 19 and Figures 21, 23 indicate that Algorithm 1 with a constant learning rate outperformed Algorithm 1 with a diminishing one in terms of the training and test accuracy score. In particular, Algorithm 1 with a constant learning rate had high accuracies. Figures 18 and 22 show that AMSG-C1 and AMSG-D2 minimized the training loss function faster than the other algorithms, and Figures 20 and 24 show that all algorithms except for GWDC-C2, ADAM-D1, and AMSG-D1 minimized the test loss function.
2) ResNet

Figures 25, 27 and Figures 29, 31 indicate that Algorithm 1 with a constant learning rate outperformed Algorithm 1 with a diminishing one in terms of the training and test accuracy score. In particular, ADAM-D1 and AMSG-D1 did not work and had low accuracies. Figure 28 shows that AMSG-C2, AMSG-C3, and GWDC-C1 minimized the training loss function, while Figure 30 shows that ADAM-D2, ADAM-D3, AMSG-D2, and AMSG-D3 minimized the test loss function faster than other algorithms, such as GWDC-Di and AMSGWDC-Di (i = 1, 2, 3). Meanwhile, Figure 32 shows that all algorithms except for ADAM-D1 and AMSG-D1 minimized the test loss function, and Figure 28 and Figure 32 show that the algorithm with a constant learning rate performed better than the one with a diminishing rate.

C. UNCONSTRAINED OPTIMIZATION USING BENCHMARK FUNCTIONS

We performed optimizations for the seven unimodal benchmark functions in Table 3 [12, Table 2] and compared Algorithm 1 with heuristic intelligent optimization methods\(^2\), as follows:

- **GA\(^3\)**: Genetic algorithm, where the population size was 100, the probability of performing crossover was 0.95, the probability of mutation was 0.025, and the maximum iteration number was 500.

- **PSO [17]**: Particle swarm optimization algorithm, where the population size was 100, the acceleration constant

\(^2\)https://github.com/thieu1995/mealpy
\(^3\)https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_quick_guide.htm
was 1.2, and the maximum iteration number was 500.

- BBO [18]: Biogeography-based optimization algorithm, where the population size was 100, the mutation probability was 0.01, the number of elites was 2, and the maximum iteration number was 500.
- ASO [19]: Atom search optimization algorithm, where the population size was 100, the depth weight was 50, the multiplier weight was 0.2, and the maximum iteration number was 500.
- TLBO [20]: Teaching-learning-based optimization algorithm, where the population size was 100 and the maximum iteration number was 500.

The stopping condition for all the algorithms was \( n = 500 \).

Figures 33–39 plots the values of the Sphere, Schwefel2.22, Schwefel1.2, Schwefel2.21, Resonbrock, Step, and Quartic benchmark functions for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time. These figures show that all the algorithms minimized the seven benchmark functions. For example, TLBO, AMSGWDC-C3, and GWDC-C3 performed well for Sphere and Schwefel1.2, while TLBO converged to a solution faster than Algorithm 1 for Schwefel2.22. In contrast, in the case of the Resonbrock benchmark function, TLBO still had not converged to the solution when the stopping condition was reached, but AMSGWDC-C3 and GWDC-C3 had converged to a solution.

Figures 40–46 plots the values of Sphere, Schwefel2.22, Schwefel1.2, Schwefel2.21, Resonbrock, Step, and Quartic benchmark functions for Algorithm 1 using diminishing...
learning rates with GA, PSO, BBO, ASO, and TLBO versus elapsed time. These figures also show that all the algorithms minimized the seven benchmark functions. For example, for Schwefel2.22, TLBO converged to the solution faster than Algorithm 1. For Step, Algorithm 1 performed better than TLBO.

VI. CONCLUSION

This paper presented a unification of the existing adaptive-learning-rate optimization algorithms for nonconvex stochastic optimization in deep neural networks. It also presented two convergence analyses of the algorithm. The first analysis showed that the algorithm approximates a stationary point of the problem when it uses a constant learning rate. The second analysis showed that the algorithm converges to a stationary point of the problem when it uses a diminishing learning rate. The advantage of the proposed convergence analyses over the existing ones is that they show that the existing adaptive-learning-rate optimization algorithms can be applied to stochastic optimization in deep neural networks. The experiments provided support for the convergence analyses. In particular, the numerical results for training neural networks showed that the algorithm with a constant learning rate performed better than the one with a diminishing rate, as promised by the convergence analyses. Moreover, the numerical results for minimizing benchmark functions showed that the proposed algorithm and TLBO performed well.

VII. ACKNOWLEDGMENT

We are sincerely grateful to the Associate Editor, Nishant Unnikrishnan, and the two anonymous reviewers for helping us improve the original manuscript.

REFERENCES

[1] M. A. Hanif, A. Manglik, and M. Shaﬁque, “Resistive crossbar-aware neural network design and optimization,” IEEE Access, vol. 8, pp. 229066–229085, 2020.

TABLE 3. Unimodal benchmark functions [12, Table 2]

| Name        | Benchmark Function | Dimension | Range        | \( f_{\text{min}} \) |
|-------------|--------------------|-----------|--------------|----------------------|
| Sphere      | \( f_1(x) = \sum_{i=1}^{n} x_i^2 \) | 50        | \( x_i \in [-100, 100] \) | 0                     |
| Schwefel22.22| \( f_2(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i| \) | 50        | \( x_i \in [-10, 10] \) | 0                     |
| Schwefel1.2 | \( f_3(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2 \) | 50        | \( x_i \in [-100, 100] \) | 0                     |
| Schwefel21.2| \( f_4(x) = \max_{1 \leq i \leq D} \{ |x_i| \} \) | 50        | \( x_i \in [-100, 100] \) | 0                     |
| Resonbrock  | \( f_5(x) = \sum_{i=1}^{D} |100(x_{i-1} + 0.5) + (x_i - 1)|^2 \) | 50        | \( x_i \in [-30, 30] \) | 0                     |
| Step        | \( f_6(x) = \sum_{i=1}^{n} (|x_i + 0.5|)^2 \) | 50        | \( x_i \in [-100, 100] \) | 0                     |
| Quartic     | \( f_7(x) = \sum_{i=1}^{n} x_i^2 + \text{random}(0, 1) \) | 50        | \( x_i \in [-1.28, 1.28] \) | 0                     |

FIGURE 13. Training classification accuracy score for Algorithm 1 with a diminishing learning rate versus the number of epochs on the MNIST dataset using a double-layer perceptron

FIGURE 14. Training loss function value for Algorithm 1 with a diminishing learning rate versus the number of epochs on the MNIST dataset using a double-layer perceptron
FIGURE 15. Test classification accuracy score for Algorithm 1 with a diminishing learning rate versus the number of epochs on the MNIST dataset using a double-layer perceptron.

FIGURE 16. Test loss function value for Algorithm 1 with a diminishing learning rate versus the number of epochs on the MNIST dataset using a double-layer perceptron.

FIGURE 17. Training classification accuracy score for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet.

FIGURE 18. Training loss function value for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet.
FIGURE 19. Test classification accuracy score for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet

FIGURE 20. Test loss function value for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet

FIGURE 21. Training classification accuracy score for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet

FIGURE 22. Training loss function value for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet

[15] H. Iiduka, “Stochastic fixed point optimization algorithm for classifier ensemble,” IEEE Transactions on Cybernetics, vol. 50, no. 10, pp. 4370–4380, 2020.
[16] L. Luo, Y. Xiong, Y. Liu, and X. Sun, “Adaptive gradient methods with dynamic bound of learning rate,” Proceedings of The International Conference on Learning Representations, pp. 1–21, 2019.
[17] F. van den Bergh, An Analysis of Particle Swarm Optimizers. PhD thesis, The Faculty of Natural and Agricultural Science, University of Pretoria, 2001.
[18] D. Simon, “Biogeography-based optimization,” IEEE Transactions on Evolutionary Computation, vol. 12, no. 6, pp. 702–713, 2008.
[19] W. Zhao, L. Wang, and Z. Zhang, “Atom search optimization and its application to solve a hydrogeologic parameter estimation problem,” Knowledge-Based Systems, vol. 163, pp. 283–304, 2019.
[20] R. Venkata Rao and V. Patel, “An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems,” International Journal of Industrial Engineering Computations, vol. 3, no. 4, pp. 535–560, 2012.

YINI ZHU has been a masters student in Computer Science Course, Graduate School of Science and Technology, Meiji University, Kanagawa, Japan, since April 2020. Her research field is optimization theory and its applications.
FIGURE 23. Test classification accuracy score for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet

FIGURE 24. Test loss function value for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using DenseNet

FIGURE 25. Training classification accuracy score for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet

FIGURE 26. Training loss function value for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet

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.
FIGURE 27. Test classification accuracy score for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet.

FIGURE 28. Test loss function value for Algorithm 1 with a constant learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet.

FIGURE 29. Training classification accuracy score for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet.

FIGURE 30. Training loss function value for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet.
FIGURE 31. Test classification accuracy score for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet.

FIGURE 32. Test loss function value for Algorithm 1 with a diminishing learning rate versus the number of epochs on the CIFAR-10 dataset using ResNet.

FIGURE 33. Sphere values for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time.

FIGURE 34. Schwefel2.22 values for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time.
FIGURE 35. Schwefel1.2 values for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

FIGURE 36. Schwefel2.21 values for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

FIGURE 37. Resonbrock values for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

FIGURE 38. Step Function values for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time
FIGURE 39. Quartic function values for Algorithm 1 using constant learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

FIGURE 40. Sphere values for Algorithm 1 using diminishing learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

FIGURE 41. Schwefel2.22 values for Algorithm 1 using diminishing learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time

FIGURE 42. Schwefel1.2 values for Algorithm 1 using diminishing learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time
FIGURE 43. Schwefel2.21 values for Algorithm 1 using diminishing learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time.

FIGURE 44. Resonbrock values for Algorithm 1 using diminishing learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time.

FIGURE 45. Step Function values for Algorithm 1 using diminishing learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time.

FIGURE 46. Quartic function values for Algorithm 1 using diminishing learning rates with heuristic intelligent optimization methods (GA, PSO, BBO, ASO, and TLBO) versus elapsed time.