SMG: A Shuffling Gradient-Based Method with Momentum

Trang H. Tran [1] **Lam M. Nguyen** [2] **Quoc Tran-Dinh** [3]

Abstract

We combine two advanced ideas widely used in optimization for machine learning: *shuffling* strategy and *momentum* technique to develop a novel shuffling gradient-based method with momentum, coined **S**huffling **M**omentum **G**radient (SMG), for non-convex finite-sum optimization problems. While our method is inspired by momentum techniques, its update is fundamentally different from existing momentum-based methods. We establish state-of-the-art convergence rates of SMG for any shuffling strategy using either constant or diminishing learning rate under standard assumptions (i.e. *L-smoothness* and *bounded variance*). When the shuffling strategy is fixed, we develop another new algorithm that is similar to existing momentum methods, and prove the same convergence rates for this algorithm under the L-smoothness and bounded gradient assumptions. We demonstrate our algorithms via numerical simulations on standard datasets and compare them with existing shuffling methods. Our tests have shown encouraging performance of the new algorithms.

1. Introduction

Most training tasks in supervised learning are boiled down to solving the following finite-sum minimization:

$$\min_{w\in\mathbb{R}^d}\Big\{F(w) := \frac{1}{n}\sum_{i=1}^{n} f(w;i)\Big\}, \tag{1}$$

where $f(\cdot;i) : \mathbb{R}^d \to \mathbb{R}$ is a given smooth and possibly nonconvex function for $i \in [n] := \{1, \cdots, n\}$.

Problem (1) looks simple, but covers various convex and nonconvex applications in machine learning and statistical learning, including, but not limited to, logistic regression, multi-kernel learning, conditional random fields, and neural networks. Especially, (1) covers the *empirical risk minimization* as a special case. Solution methods for approximately solving (1) have been widely studied in the literature under different sets of assumptions. The most common approach is perhaps stochastic gradient-type (SGD) methods (Robbins & Monro, 1951; Ghadimi & Lan, 2013; Bottou et al., 2018; Nguyen et al., 2018; 2019) and their variants.

Motivation. While SGD and its variants rely on randomized sampling strategies with replacement, gradient-based methods using without-replacement strategies are often easier and faster to implement. Moreover, practical evidence (Bottou, 2009) has shown that they usually produce a faster decrease of the training loss. Randomized shuffling strategies (also viewed as sampling without replacement) allow the algorithm to use exactly one function component $f(\cdot;i)$ at each epoch compared to SGD, which has only statistical convergence guarantees (e.g., in expectation or with high probability). However, very often, the analysis of shuffling methods is more challenging than SGD due to the lack of statistical independence.

In the deterministic case, single permutation (also called shuffle once, or single shuffling) and incremental gradient methods can be considered as special cases of the shuffling gradient-based methods we study in this paper. One special shuffling strategy is randomized reshuffling, which is broadly used in practice, where we use a different random permutation at each epoch. Alternatively, in recent years, it has been shown that many gradient-based methods with momentum update can notably boost the convergence speed both in theory and practice (Nesterov, 2004; Dozat, 2016; Wang et al., 2020). These methods have been widely used in both convex and nonconvex settings, especially, in deep learning community. Remarkably, Nesterov's accelerated method (Nesterov, 1983) has made a revolution in large-scale convex optimization in the last two decades, and has been largely exploited in nonconvex problems. The developments we have discussed here motivate us to raise the following research question:

Can we combine both shuffling strategy and momentum scheme to develop new provable gradient-based algorithms for handling (1)?

[1]School of Operations Research and Information Engineering, Cornell University, Ithaca, NY, USA. [2]IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY, USA. [3]Department of Statistics and Operations Research, The University of North Carolina at Chapel Hill, NC, USA. Correspondence to: Lam M. Nguyen <LamNguyen.MLTD@ibm.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

In this paper, we answer this question affirmatively by proposing a novel algorithm called Shuffling Momentum Gradient (SMG). We establish its convergence guarantees for different shuffling strategies, and in particular, randomized reshuffling strategy. We also investigate different variants of our method.

Our contribution. To this end, our contributions in this paper can be summarized as follows.

(a) We develop a novel shuffling gradient-based method with momentum (Algorithm 1 in Section 2) for approximating a stationary point of the nonconvex minimization problem (1). Our algorithm covers any shuffling strategy ranging from deterministic to randomized, including incremental, single shuffling, and randomized reshuffling variants.

(b) We establish the convergence of our method in the nonconvex setting and achieve the state-of-the-art $\mathcal{O}\left(1/T^{2/3}\right)$ convergence rate under standard assumptions (i.e. the L-smoothness and bounded variance conditions), where T is the number of epochs. For randomized reshuffling strategy, we can improve our convergence rate up to $\mathcal{O}\left(1/(n^{1/3}T^{2/3})\right)$.

(c) We study different strategies for selecting learning rates (LR), including constant, diminishing, exponential, and cosine scheduled learning rates. In all cases, we prove the same convergence rate of the corresponding variants without any additional assumption.

(d) When a single shuffling strategy is used, we show that a momentum strategy can be incorporated directly at each iteration of the shuffling gradient method to obtain a different variant as presented in Algorithm 2. We analyze the convergence of this algorithm and achieve the same $\mathcal{O}(1/T^{2/3})$ epoch-wise convergence rate, but under a bounded gradient assumption instead of the bounded variance as for the SMG algorithm.

Our $\mathcal{O}(1/T^{2/3})$ convergence rate is the best known so far for shuffling gradient-type methods in nonconvex optimization (Nguyen et al., 2020; Mishchenko et al., 2020). However, like (Mishchenko et al., 2020), our SMG method only requires a generalized bounded variance assumption (Assumption 1(c)), which is weaker and more standard than the bounded component gradient assumption used in existing works. Algorithm 2 uses the same set of assumptions as in (Nguyen et al., 2020) to achieve the same rate, but has a momentum update. For the randomized reshuffling strategy, our $\mathcal{O}\left(1/(n^{1/3}T^{2/3})\right)$ convergence rate also matches the rate of the without-momentum algorithm in (Mishchenko et al., 2020). It leads to the total of iterations $nT = \mathcal{O}(\sqrt{n}\varepsilon^{-3})$.

We emphasize that, in many existing momentum variants, the momentum $m_i^{(t)}$ is updated recursively at each iteration as $m_{i+1}^{(t)} := \beta m_i^{(t)} + (1-\beta)g_i^{(t)}$ for a given weight $\beta \in (0,1)$. This update shows that the momentum $m_{i+1}^{(t)}$ incorporates all the past gradient terms $g_i^{(t)}, g_{i-1}^{(t)}, g_{i-1}^{(t)}, \ldots, g_0^{(t)}$ with exponential decay weights $1, \beta, \beta^2, \ldots, \beta^i$, respectively. However, in shuffling methods, the convergence guarantee is often obtained in epoch. Based on this observation, we modify the classical momentum update in the shuffling method as shown in Algorithm 1. More specifically, the momentum term $m_0^{(t)}$ is fixed at the beginning of each epoch, and an auxiliary sequence $\{v_i^{(t)}\}$ is introduced to keep track of the gradient average to update the momentum term in the next epoch. This modification makes Algorithm 1 fundamentally different from existing momentum-based methods. This new algorithm still achieves $\mathcal{O}(1/T^{2/3})$ epoch-wise convergence rate under standard assumptions. To the best of our knowledge, our work is the first analyzing convergence rate guarantees of shuffling-type gradient methods with momentum under standard assumptions.

Besides Algorithm 1, we also exploit recent advanced strategies for selecting learning rates, including exponential and cosine scheduled learning rates. These two strategies have shown state-of-the-art performance in practice (Smith, 2017; Loshchilov & Hutter, 2017; Li et al., 2020). Therefore, it is worth incorporating them in shuffling methods.

Related work. Let us briefly review the most related works to our methods studied in this paper.

Shuffling gradient-based methods. Shuffling gradient-type methods for solving (1) have been widely studied in the literature in recent years (Bottou, 2009; Gürbüzbalaban et al., 2019; Shamir, 2016; Haochen & Sra, 2019; Nguyen et al., 2020) for both convex and nonconvex settings. It was empirically investigated in a short note (Bottou, 2009) and also discussed in (Bottou, 2012). These methods have also been implemented in several software packages such as TensorFlow and PyTorch, broadly used in machine learning (Abadi et al., 2015; Paszke et al., 2019).

In the strongly convex case, shuffling methods have been extensively studied in (Ahn et al., 2020; Gürbüzbalaban et al., 2019; Haochen & Sra, 2019; Safran & Shamir, 2020; Nagaraj et al., 2019; Rajput et al., 2020; Nguyen et al., 2020; Mishchenko et al., 2020) under different assumptions. The best known convergence rate in this case is $\mathcal{O}(1/(nT)^2 + 1/(nT^3))$, which matches the lower bound rate studied in (Safran & Shamir, 2020) up to some constant factor. Most results in the convex case are for the incremental gradient variant, which are studied in (Nedic & Bertsekas, 2001; Nedić & Bertsekas, 2001). Convergence results of shuffling methods on the general convex case are investigated in (Shamir, 2016; Mishchenko et al., 2020), where (Mishchenko et al., 2020) provides a unified approach to cover different settings. The authors in (Ying et al., 2017) combine a randomized shuffling and a variance reduction

technique (e.g., SAGA (Defazio et al., 2014) and SVRG (Johnson & Zhang, 2013)) to develop a new variant. They show a linear convergence rate for strongly convex problems but using an energy function, which is unclear how to convert it into known convergence criteria.

In the nonconvex case, (Nguyen et al., 2020) first shows $\mathcal{O}(1/T^{2/3})$ convergence rate for a general class of shuffling gradient methods under the L-smoothness and bounded gradient assumptions on (1). This analysis is then extended in (Mishchenko et al., 2020) to a more relaxed assumption. The authors in (Meng et al., 2019) study different distributed SGD variants with shuffling for strongly convex, general convex, and nonconvex problems. An incremental gradient method for weakly convex problems is investigated in (Li et al., 2019), where the authors show $\mathcal{O}(1/T^{1/2})$ convergence rate as in standard SGD. To the best of our knowledge, the best known rate of shuffling gradient methods for the nonconvex case under standard assumptions is $\mathcal{O}(1/T^{2/3})$ as shown in (Nguyen et al., 2020; Mishchenko et al., 2020).

Our Algorithm 1 developed in this paper is a nontrivial momentum variant of the general shuffling method in (Nguyen et al., 2020) but our analysis uses a standard bounded variance assumption instead of bounded gradient one.

Momentum-based methods. Gradient methods with momentum were studied in early works for convex problems such as heavy-ball, inertial, and Nesterov's accelerated gradient methods (Polyak, 1964; Nesterov, 2004). Nesterov's accelerated method is the most influent scheme and achieves optimal convergence rate for convex problems. While momentum-based methods are not yet known to improve theoretical convergence rates in the nonconvex setting, they show significantly encouraging performance in practice (Dozat, 2016; Wang et al., 2020), especially in the deep learning community. However, the momentum strategy has not yet been exploited in shuffling methods.

Adaptive learning rate schemes. Gradient-type methods with adaptive learning rates such as AdaGrad (Duchi et al., 2011) and Adam (Kingma & Ba, 2014) have shown state-of-the-art performance in several optimization applications. Recently, many adaptive schemes for learning rates have been proposed such as diminishing (Nguyen et al., 2020), exponential scheduled (Li et al., 2020), and cosine scheduled (Smith, 2017; Loshchilov & Hutter, 2017). In (Li et al., 2020), the authors analyze convergence guarantees for the exponential and cosine learning rates in SGD. These adaptive learning rates have also been empirically studied in the literature, especially in machine learning tasks. Their convergence guarantees have also been investigated accordingly under certain assumptions.

Although the analyses for adaptive learning rates are generally non-trivial, our theoretical results in this paper are flexible enough to cover various different learning rates. However, we only exploit the diminishing, exponential scheduled, and cosine scheduled schemes for our shuffling methods with momentum. We establish that in the last two cases, our algorithm still achieves state-of-the-art $\mathcal{O}(T^{-2/3})$ (and possibly up to $\mathcal{O}(n^{-1/3}T^{-2/3})$) epoch-wise rates.

Content. The rest of this paper is organized as follows. Section 2 describes our novel method, Shuffling Momentum Gradient (Algorithm 1). Section 3 investigates its convergence rate under different shuffling-type strategies and different learning rates. Section 4 proposes an algorithm with traditional momentum update for single shuffling strategy. Section 5 presents our numerical simulations. Due to space limit, the convergence analysis of our methods, all technical proofs, and additional experiments are deferred to the Supplementary Document (Supp. Doc.).

2. Shuffling Gradient-Based Method with Momentum

In this section, we describe our new shuffling gradient algorithm with momentum in Algorithm 1. We also compare our algorithm with existing methods and discuss its per-iteration complexity and possible modifications, e.g., mini-batch.

Algorithm 1 Shuffling Momentum Gradient (SMG)

Initialization: Choose $\tilde{w}_0 \in \mathbb{R}^d$ and set $\tilde{m}_0 := \mathbf{0}$.
for $t := 1, 2, \cdots, T$ **do**
  Set $w_0^{(t)} := \tilde{w}_{t-1}$; $m_0^{(t)} := \tilde{m}_{t-1}$; and $v_0^{(t)} := \mathbf{0}$;
  Generate an arbitrarily deterministic or random permutation $\pi^{(t)}$ of $[n]$;
  for $i := 0, \cdots, n-1$ **do**
    Query $g_i^{(t)} := \nabla f(w_i^{(t)}; \pi^{(t)}(i+1))$;
    Choose $\eta_i^{(t)} := \frac{\eta_t}{n}$ and update

$$\left\{\begin{array}{lcl} m_{i+1}^{(t)} & := & \beta m_0^{(t)} + (1-\beta) g_i^{(t)} \\ v_{i+1}^{(t)} & := & v_i^{(t)} + \frac{1}{n} g_i^{(t)} \\ w_{i+1}^{(t)} & := & w_i^{(t)} - \eta_i^{(t)} m_{i+1}^{(t)}; \end{array}\right. \tag{2}$$

  end for
  Set $\tilde{w}_t := w_n^{(t)}$ and $\tilde{m}_t := v_n^{(t)}$;
end for
Output: Choose $\hat{w}_T \in \{\tilde{w}_0, \cdots, \tilde{w}_{T-1}\}$ at random with probability $\mathbb{P}[\hat{w}_T = \tilde{w}_{t-1}] = \frac{\eta_t}{\sum_{t=1}^T \eta_t}$.

Clearly, if $\beta = 0$, then Algorithm 1 reduces to the standard shuffling gradient method as in (Nguyen et al., 2020; Mishchenko et al., 2020). Since each inner iteration of our method uses one component $f(\cdot, i)$, we use $\eta_i^{(t)} = \frac{\eta_t}{n}$ in Algorithm 1 to make it consistent with one full gradient, which consists of n gradient components. This form looks

different from the learning rates used in previous work for SGD (Shamir, 2016; Mishchenko et al., 2020), however, it does not necessary make our learning rate smaller. In the same order of training samples n, our learning rate matches the one in (Mishchenko et al., 2020) as well as the state-of-the-art complexity results. In fact, the detailed learning rate η_t used in Algorithm 1 will be specified in Section 3.

Comparison. Unlike existing momentum methods where $m_i^{(t)}$ in (2) is updated recursively as $m_{i+1}^{(t)} := \beta m_i^{(t)} + (1-\beta)g_i^{(t)}$ for $\beta \in (0,1)$, we instead fix the first term $m_0^{(i)}$ in (2) at each epoch. It is only updated at the end of each epoch by averaging all the gradient components $\{g_i^{(t)}\}_{i=0}^{n-1}$ evaluated in such an epoch. To avoid storing these gradients, we introduce an auxiliary variable $v_i^{(t)}$ to keep track of the gradient average. Here, we fix the weight β for the sake of our analysis, but it is possible to make β adaptive.

Our new method is based on the following observations. First, while SGD generally uses an unbiased estimator for the full gradient, shuffling gradient methods often do not have such a nice property, making them difficult to analyze. Due to this fact, updating the momentum at each inner iteration does not seem preferable since it could make the estimator deviate from the true gradient. Therefore, we consider updating the momentum after each epoch. Second, unlike the traditional momentum with exponential decay weights, our momentum term $m_0^{(t)}$ is an equal-weighted average of all the past gradients in an epoch, but evaluated at different points $w_i^{(t-1)}$ in the inner loop. Based on these observations, the momentum term should aid the full gradient ∇F instead of its component $\nabla f(\cdot; i)$, leading to the update at the end of each epoch.

It is also worth noting that our algorithm is fundamentally different from variance reduction methods. For instance, SVRG and SARAH variants require evaluating full gradient at each snapshot point while our method always uses single component gradient. Hence, our algorithm does not require a full gradient evaluation at each outer iteration, and our momentum term does not require full gradient of f.

When the learning rate $\eta_i^{(t)}$ is fixed at each epoch as $\eta_i^{(t)} := \frac{\eta_t}{n}$, where $\eta_t > 0$, we can derive from (2) that

$$w_{i+1}^{(t)} = w_i^{(t)} - \frac{(1-\beta)\eta_t}{n} g_i^{(t)} + \frac{\beta\eta_t}{n(1-\beta)\eta_{t-1}} e_t,$$

where $e_t := \tilde{w}_{t-1} - \tilde{w}_{t-2} + \beta\eta_{t-1}\tilde{m}_{t-2}$. Here, e_t plays a role as a momentum or an inertial term, but it is different from the usual momentum term. However, we still name Algorithm 1 the *Shuffling Momentum Gradient* since it is inspired by momentum methods.

Per-iteration complexity. The per-iteration complexity of Algorithm 1 is almost the same as in standard shuffling gradient schemes, see, e.g., (Shamir, 2016). It needs to store two additional vectors $m_{i+1}^{(t)}$ and $v_i^{(t)}$, and performs two vector additions and three scalar-vector multiplications. Such additional costs are very mild.

Shuffling strategies. Our convergence guarantee in Section 3 holds for any permutation $\pi^{(t)}$ of $\{1, 2, \cdots, n\}$, including deterministic and randomized ones. Therefore, our method covers any shuffling strategy, including incremental, single shuffling, and randomized reshuffling variants as special cases. We highlight that our convergence result for the randomized reshuffling variant is significantly better than the general ones as we will explain in detail in Section 3.

Mini-batch. Algorithm 1 also works with mini-batches, and our theory can be slightly adapted to establish the same convergence rate for mini-batch variants. However, it remains unclear to us how mini-batches can improve the convergence rate guarantee of Algorithm 1 in the general case where the shuffling strategy is not specified.

3. Convergence Analysis

We analyze the convergence of Algorithm 1 under standard assumptions, which is organized as follows.

3.1. Technical Assumptions

Our analysis relies on the following assumptions:

Assumption 1. *Problem (1) satisfies:*

(a) ***(Boundedness from below)*** $\mathrm{dom}(F) \neq \emptyset$ *and* F *is bounded from below, i.e.* $F_* := \inf_{w\in\mathbb{R}^d} F(w) > -\infty$.

(b) ***(L-Smoothness)*** $f(\cdot; i)$ *is* L*-smooth for all* $i \in [n]$*, i.e. there exists a universal constant* $L > 0$ *such that, for all* $w, w' \in \mathrm{dom}\,(F)$*, it holds that*

$$\|\nabla f(w; i) - \nabla f(w'; i)\| \leq L\|w - w'\|. \quad (3)$$

(c) ***(Generalized bounded variance)*** *There exist two non-negative and finite constants* Θ *and* σ *such that for any* $w \in \mathrm{dom}\,(F)$ *we have*

$$\frac{1}{n}\sum_{i=1}^{n} \|\nabla f(w; i) - \nabla F(w)\|^2 \leq \Theta\|\nabla F(w)\|^2 + \sigma^2. \quad (4)$$

Assumption 1(a) is required in any algorithm to guarantee the well-definedness of (1). The L-smoothness (3) is standard in gradient-type methods for both stochastic and deterministic algorithms. From this assumption, we have for any $w, w' \in \mathrm{dom}\,(F)$ (see (Nesterov, 2004)):

$$F(w) \leq F(w') + \langle \nabla F(w'), w - w'\rangle + \frac{L}{2}\|w - w'\|^2. \quad (5)$$

The condition (4) in Assumption 1(c) reduces to the standard bounded variance condition if $\Theta = 0$. Therefore, (4) is

more general than the bounded variance assumption, which is often used in stochastic optimization. Unlike recent existing works on momentum SGD and shuffling (Chen et al., 2018; Nguyen et al., 2020), we do not assume the bounded gradient assumption on each $f(\cdot;i)$ in Algorithm 1 (see Assumption 2). This condition is stronger than (4).

3.2. Main Result 1 and Its Consequences

Our first main result is the following convergence theorem for Algorithm 1 under any shuffling strategy.

Theorem 1. *Suppose that Assumption 1 holds for (1). Let $\{w_i^{(t)}\}_{t=1}^T$ be generated by Algorithm 1 with a fixed momentum weight $0 \le \beta < 1$ and an epoch learning rate $\eta_i^{(t)} := \frac{\eta_t}{n}$ for every $t \ge 1$. Assume that $\eta_0 = \eta_1$, $\eta_t \ge \eta_{t+1}$, and $0 < \eta_t \le \frac{1}{L\sqrt{K}}$ for $t \ge 1$, where $K := \max\left\{\frac{5}{2}, \frac{9(5-3\beta)(\Theta+1)}{1-\beta}\right\}$. Then, it holds that*

$$\mathbb{E}\big[\|\nabla F(\hat{w}_T)\|^2\big] = \frac{1}{\sum_{t=1}^T \eta_t}\sum_{t=1}^T \eta_t \|\nabla F(\tilde{w}_{t-1})\|^2 \tag{6}$$
$$\le \frac{4[F(\tilde{w}_0) - F_*]}{(1-\beta)\sum_{t=1}^T \eta_t} + \frac{9\sigma^2 L^2(5-3\beta)}{(1-\beta)}\left(\frac{\sum_{t=1}^T \eta_{t-1}^3}{\sum_{t=1}^T \eta_t}\right).$$

Remark 1 (Convergence guarantee)**.** *When a deterministic permutation $\pi^{(t)}$ is used, our convergence rate can be achieved in a deterministic sense. However, to unify our analysis, we will express our convergence guarantees in expectation, where the expectation is taken over all the randomness generated by $\pi^{(t)}$ and $\hat{w}_T$ up to T iterations. Since we can choose permutations $\pi^{(t)}$ either deterministically or randomly, our bounds in the sequel will hold either deterministically or with probability 1 (w.p.1), respectively. Without loss of generality, we write these results in expectation.*

Next, we derive two direct consequences of Theorem 1 by choosing constant and diminishing learning rates.

Corollary 1 (**Constant learning rate**)**.** *Let us fix the number of epochs $T \ge 1$, and choose a constant learning rate $\eta_t := \frac{\gamma}{T^{1/3}}$ for some $\gamma > 0$ such that $\frac{\gamma}{T^{1/3}} \le \frac{1}{L\sqrt{K}}$ for $t \ge 1$ in Algorithm 1. Then, under the conditions of Theorem 1, $\mathbb{E}\big[\|\nabla F(\hat{w}_T)\|^2\big]$ is upper bounded by*

$$\frac{1}{T^{2/3}}\left(\frac{4[F(\tilde{w}_0) - F_*]}{(1-\beta)\gamma} + \frac{9\sigma^2(5-3\beta)L^2\gamma^2}{(1-\beta)}\right).$$

Consequently, the convergence rate of Algorithm 1 is $\mathcal{O}(T^{-2/3})$ in epoch.

With a constant LR as in Corollary 1, the convergence rate of Algorithm 1 is exactly expressed as

$$\mathcal{O}\left(\frac{[F(\tilde{w}_0) - F_*] + \sigma^2}{T^{2/3}}\right),$$

which matches the best known rate in the literature (Mishchenko et al., 2020; Nguyen et al., 2020) in term of T for general shuffling-type strategies.

Corollary 2 (**Diminishing learning rate**)**.** *Let us choose a diminishing learning rate $\eta_t := \frac{\gamma}{(t+\lambda)^{1/3}}$ for some $\gamma > 0$ and $\lambda \ge 0$ for $t \ge 1$ such that $\eta_1 := \frac{\gamma}{(1+\lambda)^{1/3}} \le \frac{1}{L\sqrt{K}}$ in Algorithm 1. Then, under the conditions of Theorem 1, after T epochs with $T \ge 2$, we have*

$$\mathbb{E}\big[\|\nabla F(\hat{w}_T)\|^2\big] \le \frac{C_1 + C_2\log(T-1+\lambda)}{(T+\lambda)^{2/3} - (1+\lambda)^{2/3}},$$

where C_1 and C_2 are respectively given by

$$C_1 := \frac{4\left[F(\tilde{w}_0) - F_*\right]}{(1-\beta)\gamma} + \frac{18\sigma^2L^2(5-3\beta)\gamma^2}{(1-\beta)(1+\lambda)} \quad \text{and}$$
$$C_2 := \frac{9\sigma^2L^2(5-3\beta)\gamma^2}{(1-\beta)}.$$

Consequently, the convergence rate of Algorithm 1 is $\mathcal{O}(T^{-2/3}\log(T))$ in epoch.

The diminishing LR $\eta_t := \frac{\gamma}{(t+\lambda)^{1/3}}$ allows us to use large learning rate values at early epochs compared to the constant case. However, we loose a $\log(T)$ factor in the second term of our worst-case convergence rate bound.

We also derive the following two consequences of Theorem 1 for exponential and cosine scheduled learning rates.

Exponential scheduled learning rate. Given an epoch budget $T \ge 1$, and two positive constants $\gamma > 0$ and $\rho > 0$, we consider the following exponential LR, see (Li et al., 2020):

$$\eta_t := \frac{\gamma\alpha^t}{T^{1/3}}, \quad \text{where } \alpha := \rho^{1/T} \in (0,1). \tag{7}$$

The following corollary shows the convergence of Algorithm 1 using this LR without any additional assumption.

Corollary 3. *Let $\{w_i^{(t)}\}_{t=1}^T$ be generated by Algorithm 1 with $\eta_i^{(t)} := \frac{\eta_t}{n}$, where η_t is in (7) such that $0 < \eta_t \le \frac{1}{L\sqrt{K}}$. Then, under Assumption 1, we have*

$$\mathbb{E}\big[\|\nabla F(\hat{w}_T)\|^2\big] \le \frac{4[F(\tilde{w}_0) - F_*]}{(1-\beta)\gamma\rho T^{2/3}} + \frac{9\sigma^2L^2(5-3\beta)\gamma^2}{(1-\beta)\rho T^{2/3}}.$$

Cosine annealing learning rate: Alternatively, given an epoch budget $T \ge 1$, and a positive constant $\gamma > 0$, we consider the following cosine LR for Algorithm 1:

$$\eta_t := \frac{\gamma}{T^{1/3}}\left(1 + \cos\frac{t\boldsymbol{\pi}}{T}\right), \quad t = 1, 2, \cdots, T. \tag{8}$$

This LR is adopted from (Loshchilov & Hutter, 2017; Smith, 2017). However, different from these works, we fix our learning rate at each epoch instead of updating it at every iteration as in (Loshchilov & Hutter, 2017; Smith, 2017).

Corollary 4. *Let $\{w_i^{(t)}\}_{t=1}^T$ be generated by Algorithm 1 with $\eta_i^{(t)} := \frac{\eta_t}{n}$, where η_t is given by (8) such that $0 < \eta_t \leq \frac{1}{L\sqrt{K}}$. Then, under Assumption 1, and for $T \geq 2$, $\mathbb{E}\left[\|\nabla F(\hat{w}_T)\|^2\right]$ is upper bounded by*

$$\frac{1}{T^{2/3}}\left(\frac{8[F(\tilde{w}_0)-F_*]}{(1-\beta)\gamma}+\frac{144\sigma^2(5-3\beta)L^2\gamma^2}{(1-\beta)}\right).$$

The scheduled LRs (7) and (8) still preserve our best known convergence rate $\mathcal{O}(T^{-2/3})$. Note that the exponential learning rates are available in both TensorFlow and PyTorch, while cosine learning rates are also used in PyTorch.

3.3. Main Result 2: Randomized Reshuffling Variant

A variant of Algorithm 1 is called a **randomized reshuffling variant** if at each iteration t, the generated permutation $\pi^{(t)} = \left(\pi^{(t)}(1), \cdots, \pi^{(t)}(n)\right)$ is uniformly sampled at random without replacement from $\{1, \cdots, n\}$. Since the randomized reshuffling strategy is extremely popular in practice, we analyze Algorithm 1 under this strategy.

Theorem 2. *Suppose that Assumption 1 holds for (1). Let $\{w_i^{(t)}\}_{t=1}^T$ be generated by Algorithm 1 under a **randomized reshuffling strategy**, a fixed momentum weight $0 \leq \beta < 1$, and an epoch learning rate $\eta_i^{(t)} := \frac{\eta_t}{n}$ for every $t \geq 1$. Assume that $\eta_t \geq \eta_{t+1}$ and $0 < \eta_t \leq \frac{1}{L\sqrt{D}}$ for $t \geq 1$, where $D = \max\left(\frac{5}{3}, \frac{6(5-3\beta)(\Theta+n)}{n(1-\beta)}\right)$ and $\eta_0 = \eta_1$. Then*

$$\mathbb{E}\left[\|\nabla F(\hat{w}_T)\|^2\right] = \frac{1}{\sum_{t=1}^T \eta_t}\sum_{t=1}^T \eta_t \mathbb{E}\left[\|\nabla F(\tilde{w}_{t-1})\|^2\right] \quad (9)$$
$$\leq \frac{4\left[F(\tilde{w}_0)-F_*\right]}{(1-\beta)\sum_{t=1}^T \eta_t}+\frac{6\sigma^2(5-3\beta)L^2}{n(1-\beta)}\left(\frac{\sum_{t=1}^T \eta_{t-1}^3}{\sum_{t=1}^T \eta_t}\right).$$

We can derive the following two consequences.

Corollary 5 (Constant learning rate). *Let us fix the number of epochs $T \geq 1$, and choose a constant learning rate $\eta_t := \frac{\gamma n^{1/3}}{T^{1/3}}$ for some $\gamma > 0$ such that $\frac{\gamma n^{1/3}}{T^{1/3}} \leq \frac{1}{L\sqrt{D}}$ for $t \geq 1$ in Algorithm 1. Then, under the conditions of Theorem 2, $\mathbb{E}\left[\|\nabla F(\hat{w}_T)\|^2\right]$ is upper bounded by*

$$\frac{1}{n^{1/3}T^{2/3}}\left(\frac{4\left[F(\tilde{w}_0)-F_*\right]}{(1-\beta)\gamma}+\frac{6\sigma^2(5-3\beta)L^2\gamma^2}{(1-\beta)}\right).$$

Consequently, the convergence rate of Algorithm 1 is $\mathcal{O}(n^{-1/3}T^{-2/3})$ in epoch.

With a constant LR as in Corollary 5, the convergence rate of Algorithm 1 is improved to

$$\mathcal{O}\left(\frac{[F(\tilde{w}_0)-F_*]+\sigma^2}{n^{1/3}T^{2/3}}\right),$$

which matches the best known rate as in the randomized reshuffling scheme, see, e.g., (Mishchenko et al., 2020).

In this case, the total number of iterations $\mathcal{T}_{\text{tol}} := nT$ is $\mathcal{T}_{\text{tol}} = \mathcal{O}\left(\sqrt{n}\varepsilon^{-3}\right)$ to obtain $\mathbb{E}\left[\|\nabla F(\hat{w}_T)\|^2\right] \leq \varepsilon^2$. Compared to the complexity $\mathcal{O}(\varepsilon^{-4})$ of SGD, our randomized reshuffling variant is better than SGD if $n \leq \mathcal{O}(\varepsilon^{-2})$.

Corollary 6 (Diminishing learning rate). *Let us choose a diminishing learning rate $\eta_t := \frac{\gamma n^{1/3}}{(t+\lambda)^{1/3}}$ for some $\gamma > 0$ and $\lambda \geq 0$ for $t \geq 1$ such that $\eta_1 := \frac{\gamma n^{1/3}}{(1+\lambda)^{1/3}} \leq \frac{1}{L\sqrt{D}}$ in Algorithm 1. Then, under the conditions of Theorem 2, after T epochs with $T \geq 2$, we have*

$$\mathbb{E}\left[\|\nabla F(\hat{w}_T)\|^2\right] \leq \frac{C_3 + C_4\log(T-1+\lambda)}{n^{1/3}\left[(T+\lambda)^{2/3}-(1+\lambda)^{2/3}\right]},$$

where C_3 and C_4 are respectively given by

$$C_3 := \frac{4\left[F(\tilde{w}_0)-F_*\right]}{(1-\beta)\gamma}+\frac{12\sigma^2(5-3\beta)L^2\gamma^2}{(1-\beta)(1+\lambda)} \quad \text{and}$$
$$C_4 := \frac{6\sigma^2(5-3\beta)L^2\gamma^2}{(1-\beta)}.$$

Consequently, the convergence rate of Algorithm 1 is $\mathcal{O}(n^{-1/3}T^{-2/3}\log(T))$ in epoch.

Remark 2. *Algorithm 1 under randomized reshuffling still works with exponential and cosine scheduled learning rates, and our analysis is similar to the one with general shuffling schemes. However, we omit their analysis here.*

4. Single Shuffling Variant

In this section, we modify the single-shuffling gradient method by directly incorporating a momentum term at each iteration. We prove for the first time that this variant still achieves state-of-the-art convergence rate guarantee under the smoothness and bounded gradient (i.e. Assumption 2) assumptions as in existing shuffling methods. This variant, though somewhat special, also covers an incremental gradient method with momentum as a special case.

Our new momentum algorithm using single shuffling strategy for solving (1) is presented in Algorithm 2.

Besides fixing the permutation π, Algorithm 2 is different from Algorithm 1 at the point of updating $m_i^{(t)}$. Here, $m_{i+1}^{(t)}$ is updated from $m_i^{(t)}$ instead of $m_0^{(t)}$ as in Algorithm 1. In addition, we update $\tilde{m}_t := m_n^{(t)}$ instead of the epoch gradient average. Note that we can write the main update of Algorithm 2 as

$$w_{i+1}^{(t)} := w_i^{(t)} - \frac{(1-\beta)\eta_t}{n}g_i^{(t)} + \beta(w_i^{(t)} - w_{i-1}^{(t)}),$$

which exactly reduces to existing momentum updates.

Algorithm 2 Single Shuffling Momentum Gradient

Initialization: Choose $\tilde{w}_0 \in \mathbb{R}^d$ and set $\tilde{m}_0 := \mathbf{0}$;
Generate a permutation π of $[n]$;
for $t := 1, 2, \cdots, T$ **do**
  Set $w_0^{(t)} := \tilde{w}_{t-1}$ and $m_0^{(t)} := \tilde{m}_{t-1}$;
  for $i =: 0, \cdots, n-1$ **do**
    Query $g_i^{(t)} := \nabla f(w_i^{(t)}; \pi(i+1))$;
    Choose $\eta_i^{(t)} := \frac{\eta_t}{n}$ and update

$$\begin{cases} m_{i+1}^{(t)} := \beta m_i^{(t)} + (1-\beta) g_i^{(t)} \\ w_{i+1}^{(t)} := w_i^{(t)} - \eta_i^{(t)} m_{i+1}^{(t)}; \end{cases}$$

  end for
  Set $\tilde{w}_t := w_n^{(t)}$ and $\tilde{m}_t := m_n^{(t)}$;
end for
Output: Choose $\hat{w}_T \in \{\tilde{w}_0, \cdots, \tilde{w}_{T-1}\}$ at random with probability $\mathbb{P}[\hat{w}_T = \tilde{w}_{t-1}] = \frac{\eta_t}{\sum_{t=1}^T \eta_t}$.

Incremental gradient method with momentum. If we choose $\pi := [n]$, then we obtain the well-known incremental gradient variant, but with momentum. Hence, Algorithm 2 is still new compared to the standard incremental gradient algorithm (Bertsekas, 2011). To prove convergence of Algorithm 2, we replace Assumption 1(c) by the following:

Assumption 2 (**Bounded gradient**). *There exists $G > 0$ such that $\|\nabla f(x; i)\| \le G$, $\forall x \in \mathrm{dom}\,(F)$ and $i \in [n]$.*

Now, we state the convergence of Algorithm 2 in the following theorem as our third main result.

Theorem 3. *Let $\{w_i^{(t)}\}_{t=1}^T$ be generated by Algorithm 2 with a LR $\eta_i^{(t)} := \frac{\eta_t}{n}$ and $0 < \eta_t \le \frac{1}{L}$ for $t \ge 1$. Then, under Assumption 1(a)-(b) and Assumption 2, we have*

$$\mathbb{E}\big[\|\nabla F(\hat{w}_T)\|^2\big] \le \frac{\Delta_1}{\big(\sum_{t=1}^T \eta_t\big)(1-\beta^n)} + L^2G^2\left(\frac{\sum_{t=1}^T \xi_t^3}{\sum_{t=1}^T \eta_t}\right) + \frac{4\beta^n G^2}{1-\beta^n}, \quad (10)$$

where $\xi_t := \max(\eta_t, \eta_{t-1})$ for $t \ge 2$, $\xi_1 = \eta_1$, and

$$\Delta_1 := 2[F(\tilde{w}_0) - F_*] + \left(\frac{1}{L} + \eta_1\right)\|\nabla F(\tilde{w}_0)\|^2 + 2L\eta_1^2 G^2.$$

Compared to (9) in Theorem 2, (10) strongly depends on the weight β as $\frac{4\beta^n G^2}{1-\beta^n}$ appears on the right-hand side of (10). Hence, β must be chosen in a specific form to obtain desired convergence rate.

Theorem 3 provides a key bound to derive concrete convergence rates in the following two corollaries.

Corollary 7 (**Constant learning rate**). *In Algorithm 2, let us fix $T \ge 1$ and choose the parameters:*

- $\beta := \big(\frac{\nu}{T^{2/3}}\big)^{1/n}$ *for some constant $\nu \ge 0$ such that $\beta \le \big(\frac{R-1}{R}\big)^{1/n}$ for some $R \ge 1$, and*
- $\eta_t := \frac{\gamma}{T^{1/3}}$ *for some $\gamma > 0$ such that $\frac{\gamma}{T^{1/3}} \le \frac{1}{L}$.*

Then, under the conditions of Theorem 3, we have

$$\mathbb{E}\big[\|\nabla F(\hat{w}_T)\|^2\big] \le \frac{D_0}{T^{2/3}} + \frac{R\|\nabla F(\tilde{w}_0)\|^2}{T} + \frac{2LG^2R\gamma}{T^{4/3}},$$

where

$$D_0 := \frac{2R}{\gamma}[F(\tilde{w}_0) - F_*] + \frac{R}{\gamma L}\|\nabla F(\tilde{w}_0)\|^2 + L^2G^2\gamma^2 + 4\nu G^2 R.$$

Thus the convergence rate of Algorithm 2 is $\mathcal{O}(T^{-2/3})$.

With a constant learning rate as in Corollary 7, the convergence rate of Algorithm 2 is

$$\mathcal{O}\left(\frac{L[F(\tilde{w}_0) - F_*] + \|\nabla F(\tilde{w}_0)\|^2 + G^2}{T^{2/3}}\right),$$

which depends on $L[F(\tilde{w}_0) - F_*]$, $\|\nabla F(\tilde{w}_0)\|^2$, and G^2, and is slightly different from Corollary 1.

Corollary 8 (**Diminishing learning rate**). *In Algorithm 2, let us choose the parameters:*

- $\beta := \big(\frac{\nu}{T^{2/3}}\big)^{1/n}$ *for some constant $\nu \ge 0$ such that $\beta \le \big(\frac{R-1}{R}\big)^{1/n}$ for some $R \ge 1$, and*
- *a diminishing learning rate $\eta_t := \frac{\gamma}{(t+\lambda)^{1/3}}$ for all $t \in [T]$ for some $\gamma > 0$ and $\lambda \ge 0$ such that $\eta_1 = \frac{\gamma}{(1+\lambda)^{1/3}} \le \frac{1}{L}$.*

Then, under the conditions of Theorem 3, we have

$$\mathbb{E}\big[\|\nabla F(\hat{w}_T)\|^2\big] \le \frac{D_1}{[(T+\lambda)^{2/3} - (1+\lambda)^{2/3}]} + \frac{4\nu G^2 R}{T^{2/3}} + \frac{L^2G^2\gamma^2\log(T-1+\lambda)}{[(T+\lambda)^{2/3} - (1+\lambda)^{2/3}]},$$

for $T \ge 2$, where

$$D_1 := \tfrac{2R}{\gamma}[F(\tilde{w}_0) - F_*] + \big[\tfrac{R}{\gamma L} + \tfrac{R}{(1+\lambda)^{1/3}}\big]\|\nabla F(\tilde{w}_0)\|^2 + \tfrac{2RL\gamma G^2}{(1+\lambda)^{2/3}} + \tfrac{2}{1+\lambda}L^2G^2\gamma^2.$$

Thus the convergence rate of Algorithm 2 is $\mathcal{O}(\frac{\log(T)}{T^{2/3}})$.

Remark 3. *Algorithm 2 still works with exponential and cosine scheduled LRs, and our analysis is similar to the one in Algorithm 1, which is deferred to the Supp. Doc.*

5. Numerical Experiments

In order to examine our algorithms, we present two numerical experiments for different nonconvex problems and compare them with some state-of-the-art SGD-type and shuffling gradient methods.

5.1. Models and Datasets

Neural Networks. We test our Shuffling Momentum Gradient (SMG) algorithm using two standard network architectures: fully connected network (FCN) and convolutional neural network (CNN). For the fully connected setting, we train the classic LeNet-300-100 model (LeCun et al., 1998) on the `Fashion-MNIST` dataset (Xiao et al., 2017) with $60,000$ images.

We also use the convolutional LeNet-5 (LeCun et al., 1998) architecture to train the well-known `CIFAR-10` dataset (Krizhevsky & Hinton, 2009) with $50,000$ samples. We repeatedly run the experiments for 10 random seeds and report the average results. All the algorithms are implemented and run in Python using the PyTorch package (Paszke et al., 2019).

Nonconvex Logistic Regression. Nonconvex regularizers are widely used in statistical learning such as approximating sparsity or gaining robustness. We consider the following nonconvex binary classification problem:

$$\min_{w\in\mathbb{R}^d}\Big\{F(w):=\frac{1}{n}\sum_{i=1}^{n}\log(1+\exp(-y_ix_i^\top w))+\lambda r(w)\Big\},$$

where $\{(x_i,y_i)\}_{i=1}^n$ is a set of training samples; and $r(w):=\frac{1}{2}\sum_{j=1}^d\frac{w_j^2}{1+w_j^2}$ is a nonconvex regularizer, and $\lambda:=0.01$ is a regularization parameter. This example was also previously used in (Wang et al., 2019; Tran-Dinh et al., 2019; Nguyen et al., 2020).

We have conducted the experiments on two classification datasets `w8a` ($49,749$ samples) and `ijcnn1` ($91,701$ samples) from `LIBSVM` (Chang & Lin, 2011). The experiment is repeated with random seeds 10 times and the average result is reported.

5.2. Comparing SMG with Other Methods

We compare our SMG algorithm with Stochastic Gradient Descent (SGD) and two other methods: SGD with Momentum (SGD-M) (Polyak, 1964) and Adam (Kingma & Ba, 2014). For the latter two algorithms, we use the hyper-parameter settings recommended and widely used in practice (i.e. momentum: 0.9 for SGD-M, and two hyper-parameters $\beta_1:=0.9$, $\beta_2:=0.999$ for Adam). For our new SMG algorithm, we fixed the parameter $\beta:=0.5$ since it usually performs the best in our experiments.

To have a fair comparison, we apply the randomized reshuffling scheme to all methods. Note that shuffling strategies are broadly used in practice and have been implemented in TensorFlow, PyTorch, and Keras (Chollet et al., 2015). We tune each algorithm using constant learning rate and report the best result obtained.

Results. Our first experiment is presented in Figure 1, where we depict the value of "train loss" (i.e. $F(w)$ in (1)) on the y-axis and the "number of effective passes" (i.e. the number of epochs) on the x-axis. It was observed that SGD-M and Adam work well for machine learning tasks (see, e.g., (Ruder, 2017)). Align with this fact, from Figure 1, we also observe that our SMG algorithm and SGD is slightly worse than SGD-M and Adam at the initial stage when training a neural network. However, SMG quickly catches up to Adam and demonstrates a good performance at the end of the training process.

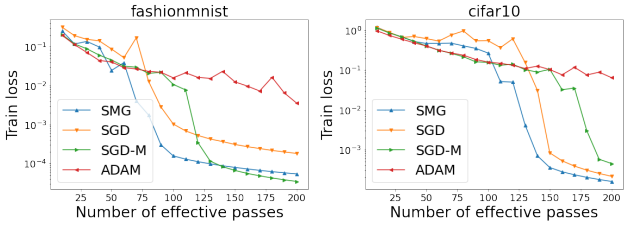

Figure 1. The train loss produced by SMG, SGD, SGD-M, and Adam for `Fashion-MNIST` and `CIFAR-10`, respectively.

For the nonconvex logistic regression problem, our result is reported in Figure 2. For two small datasets tested in our experiments, our algorithm performs significantly better than SGD and Adam, and slightly better than SGD with momentum.

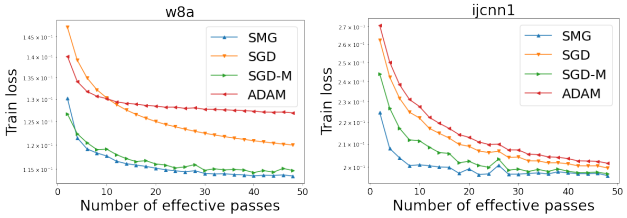

Figure 2. The train loss produced by SMG, SGD, SGD-M, and Adam for the `w8a` and `ijcnn1` datasets, respectively.

5.3. The Choice of Hyper-parameter β

Since the hyper-parameter β plays a critical role in the proposed SMG method, our next experiment is to investigate how this algorithm depends on β, while using the same constant learning rate.

Results. Our result is presented in Figure 3. We can observe from this figure that in the early stage of the training process, the choice $\beta:=0.5$ gives comparably good performance comparing to other smaller values. This choice also results in the best train loss in the end of the training process. However, the difference is not really significant, showing that SMG seems robust to the choice of the momentum weight β in the range of $[0.1,0.5]$.

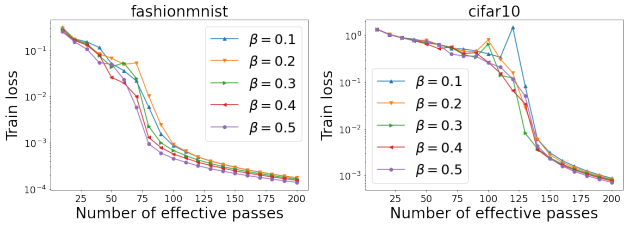

Figure 3. The train loss reported by SMG with different β on the `Fashion-MNIST` and `CIFAR-10` datasets, respectively.

In the nonconvex logistic regression problem, the same choice of β also yields similar outcome for two datasets: `w8a` and `ijcnn1`, as shown in Figure 4. We have also experimented with different choices of $\beta \in \{0.6, 0.7, 0.8, 0.9\}$. However, these choices of β do not lead to good performance, and, therefore, we omit to report them here. This finding raises an open question on the optimal choice of β. Our empirical study here shows that the choice of β in $[0.1, 0.5]$ works reasonably well, and $\beta := 0.5$ seems to be the best in our test.

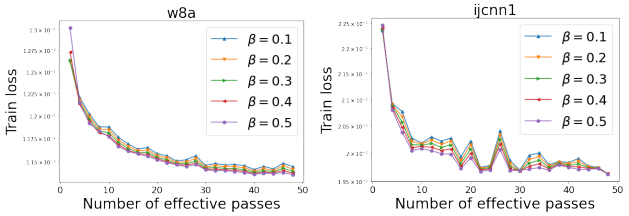

Figure 4. The train loss produced by SMG under different values of β on the `w8a` and `ijcnn1` datasets, respectively.

5.4. Different Learning Rate Schemes

Our last experiment is to examine the effect of different learning rate variants on the performance of our SMG method, i.e. Algorithm 1.

Results. We conduct this test using four different learning rate variants: constant, diminishing, exponential decay, and cosine annealing learning rates. Our results are reported in Figure 5 and Figure 6. From Figure 5, we can observe that the cosine scheduled and the diminishing learning rates converge relatively fast at the early stage. However, the exponential decay and the constant learning rates make faster progress in the last epochs and tend to give the best result at the end of the training process in our neural network training experiments.

For the non-convex logistic regression, Figure 6 shows that the cosine learning rate has certain advantages compared to other choices after a few dozen numbers of epochs.

We note that the detailed settings and additional experiments in this section can be found in the Supp. Doc.

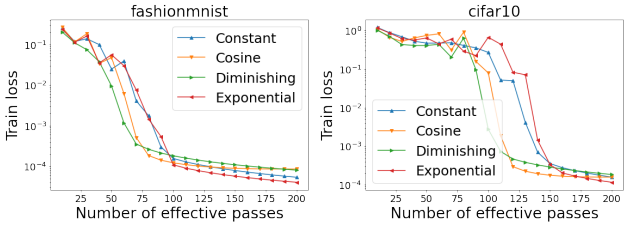

Figure 5. The train loss produced by SMG under four different learning rate schemes on the `Fashion-MNIST` and `CIFAR-10` datasets, respectively.

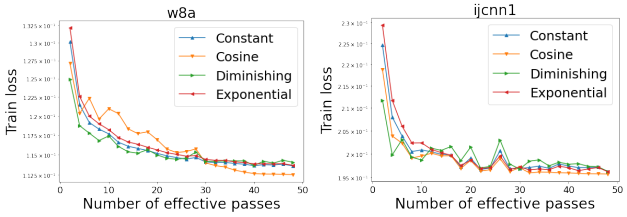

Figure 6. The train loss produced by SMG using four different learning rates on the `w8a` and `ijcnn1` datasets, respectively.

6. Conclusions and Future Work

We have developed two new shuffling gradient algorithms with momentum for solving nonconvex finite-sum minimization problems. Our Algorithm 1 is novel and can work with any shuffling strategy, while Algorithm 2 is similar to existing momentum methods using single shuffling strategy. Our methods achieve the state-of-the-art $\mathcal{O}(1/T^{2/3})$ convergence rate under standard assumptions using different learning rates and shuffling strategies. When a randomized reshuffling strategy is exploited for Algorithm 1, we can further improve our rates by a fraction $n^{1/3}$ of the data size n, matching the best known results in the non-momentum shuffling method. Our numerical results also show encouraging performance of the new methods.

We believe that our analysis framework can be extended to study non-asymptotic rates for some recent adaptive SGD schemes such as Adam (Kingma & Ba, 2014) and AdaGrad (Duchi et al., 2011) as well as variance-reduced methods such as SARAH (Nguyen et al., 2017) under shuffling strategies. It is also interesting to extend our framework to other problems such as minimax and federated learning. We will further investigate these opportunities in the future.

Acknowledgements

The authors would like to thank the reviewers for their useful comments and suggestions which helped to improve the exposition in this paper. The work of Q. Tran-Dinh has partly been supported by the Office of Naval Research under Grant No. ONR-N00014-20-1-2088 (2020-2023).

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Ahn, K., Yun, C., and Sra, S. Sgd with shuffling: optimal rates without component convexity and large epoch requirements. *arXiv preprint arXiv:2006.06946*, 2020.

Bertsekas, D. Incremental proximal methods for large scale convex optimization. *Math. Program.*, 129(2):163–195, 2011.

Bottou, L. Curiously fast convergence of some stochastic gradient descent algorithms. In *Proceedings of the symposium on learning and data science, Paris*, volume 8, pp. 2624–2633, 2009.

Bottou, L. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pp. 421–436. Springer, 2012.

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization Methods for Large-Scale Machine Learning. *SIAM Rev.*, 60(2):223–311, 2018.

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chen, X., Liu, S., Sun, R., and Hong, M. On the convergence of a class of adam-type algorithms for non-convex optimization. *ICLR*, 2018.

Chollet, F. et al. Keras, 2015. URL https://github.com/fchollet/keras.

Defazio, A., Bach, F., and Lacoste-Julien, S. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pp. 1646–1654, 2014.

Dozat, T. Incorporating nesterov momentum into ADAM. *ICLR Workshop*, 1:2013—-2016, 2016.

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

Ghadimi, S. and Lan, G. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM J. Optim.*, 23(4):2341–2368, 2013.

Gürbüzbalaban, M., Ozdaglar, A., and Parrilo, P. A. Why random reshuffling beats stochastic gradient descent. *Mathematical Programming*, Oct 2019. ISSN 1436-4646. doi: 10.1007/s10107-019-01440-w. URL http://dx.doi.org/10.1007/s10107-019-01440-w.

Haochen, J. and Sra, S. Random shuffling beats sgd after finite epochs. In *International Conference on Machine Learning*, pp. 2624–2633. PMLR, 2019.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, pp. 315–323, 2013.

Kingma, D. P. and Ba, J. ADAM: A Method for Stochastic Optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, abs/1412.6980, 2014.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, L., Zhuang, Z., and Orabona, F. Exponential step sizes for non-convex optimization. *arXiv preprint arXiv:2002.05273*, 2020.

Li, X., Zhu, Z., So, A., and Lee, J. D. Incremental methods for weakly convex optimization. *arXiv preprint arXiv:1907.11687*, 2019.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts, 2017.

Meng, Q., Chen, W., Wang, Y., Ma, Z.-M., and Liu, T.-Y. Convergence analysis of distributed stochastic gradient descent with shuffling. *Neurocomputing*, 337:46–57, 2019.

Mishchenko, K., Khaled Ragab Bayoumi, A., and Richtárik, P. Random reshuffling: Simple analysis with vast improvements. *Advances in Neural Information Processing Systems*, 33, 2020.

Nagaraj, D., Jain, P., and Netrapalli, P. Sgd without replacement: Sharper rates for general smooth convex functions. In *International Conference on Machine Learning*, pp. 4703–4711, 2019.

Nedić, A. and Bertsekas, D. Convergence rate of incremental subgradient algorithms. In *Stochastic optimization: algorithms and applications*, pp. 223–264. Springer, 2001.

Nedic, A. and Bertsekas, D. P. Incremental subgradient methods for nondifferentiable optimization. *SIAM J. on Optim.*, 12(1):109–138, 2001.

Nesterov, Y. A method for unconstrained convex minimization problem with the rate of convergence $\mathcal{O}(1/k^2)$. *Doklady AN SSSR*, 269:543–547, 1983. Translated as Soviet Math. Dokl.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87 of *Applied Optimization*. Kluwer Academic Publishers, 2004.

Nguyen, L., Nguyen, P. H., van Dijk, M., Richtarik, P., Scheinberg, K., and Takac, M. SGD and Hogwild! convergence without the bounded gradients assumption. In *Proceedings of the 35th International Conference on Machine Learning-Volume 80*, pp. 3747–3755, 2018.

Nguyen, L. M., Liu, J., Scheinberg, K., and Takáč, M. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2613–2621. JMLR. org, 2017.

Nguyen, L. M., Nguyen, P. H., Richtárik, P., Scheinberg, K., Takáč, M., and van Dijk, M. New convergence aspects of stochastic gradient algorithms. *Journal of Machine Learning Research*, 20(176):1–49, 2019. URL http://jmlr.org/papers/v20/18-759.html.

Nguyen, L. M., Tran-Dinh, Q., Phan, D. T., Nguyen, P. H., and van Dijk, M. A unified convergence analysis for shuffling-type gradient methods. *arXiv preprint arXiv:2002.08246*, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Rajput, S., Gupta, A., and Papailiopoulos, D. Closing the convergence gap of sgd without replacement. In *International Conference on Machine Learning*, pp. 7964–7973. PMLR, 2020.

Robbins, H. and Monro, S. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3): 400–407, 1951.

Ruder, S. An overview of gradient descent optimization algorithms, 2017.

Safran, I. and Shamir, O. How good is sgd with random shuffling? In *Conference on Learning Theory*, pp. 3250–3284. PMLR, 2020.

Shamir, O. Without-replacement sampling for stochastic gradient methods. In *Advances in neural information processing systems*, pp. 46–54, 2016.

Smith, L. N. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472. IEEE, 2017.

Tran-Dinh, Q., Pham, N. H., Phan, D. T., and Nguyen, L. M. Hybrid stochastic gradient descent algorithms for stochastic nonconvex optimization. *arXiv preprint arXiv:1905.05920*, 2019.

Wang, B., Nguyen, T. M., Bertozzi, A. L., Baraniuk, R. G., and Osher, S. J. Scheduled restart momentum for accelerated stochastic gradient descent. *arXiv preprint arXiv:2002.10583*, 2020.

Wang, Z., Ji, K., Zhou, Y., Liang, Y., and Tarokh, V. Spiderboost and momentum: Faster variance reduction algorithms. *Advances in Neural Information Processing Systems*, 32:2406–2416, 2019.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Ying, B., Yuan, K., and Sayed, A. H. Convergence of variance-reduced stochastic learning under random reshuffling. *arXiv preprint arXiv:1708.01383*, 2(3):6, 2017.