CONVERGENCE OF VARIANCE-REDUCED LEARNING UNDER RANDOM RESHUFFLING

Bicheng Ying[⋆] *Kun Yuan*[⋆] *Ali H. Sayed*[†]

[⋆]Department of Electrical and Computer Engineering, University of California, Los Angeles
[†]School of Engineering, Ecole Polytechnique Federale de Lausanne, Switzerland

ABSTRACT

Several useful variance-reduced stochastic gradient algorithms, such as SVRG, SAGA, Finito, and SAG, have been proposed to minimize empirical risks with linear convergence properties to the exact minimizers. The existing convergence results assume uniform data sampling with replacement. However, it has been observed that random reshuffling can deliver superior performance and, yet, no formal proofs or guarantees of exact convergence exist for variance-reduced algorithms under random reshuffling. This paper makes two contributions. First, it resolves this open issue and provides the first theoretical guarantee of linear convergence under random reshuffling for SAGA; the argument is also adaptable to other variance-reduced algorithms. Second, under random reshuffling, the paper proposes a new amortized variance-reduced gradient (AVRG) algorithm with constant storage requirements compared to SAGA and with balanced gradient computations compared to SVRG. AVRG is also shown analytically to converge linearly.

Index Terms— Random reshuffling, variance-reduction, stochastic gradient descent, linear convergence.

1. INTRODUCTION AND MOTIVATION

In recent years, several variance-reduced stochastic gradient algorithms have been proposed, including SVRG [1], SAGA [2], Finito [3], and SAG [4], with the intent of reaching the exact minimizer of an empirical risk. Under constant step-sizes and strong-convexity assumptions on the loss functions, these methods have been shown to attain linear convergence towards the exact minimizer when the data are uniformly sampled *with* replacement.

However, it has been observed that implementations that rely instead on random reshuffling (RR) of the data (i.e., sampling *without* replacement) achieve better performance than implementations that rely on uniform sampling with replacement [5–7]. Under random reshuffling, the algorithm is run multiple times over the finite data set where each run is indexed by the integer $k \geq 1$ and is referred to as an epoch. For each epoch k, the original data is reshuffled so that the sample of index i becomes the sample of index $\boldsymbol{\sigma}^k(i)$, where the symbol $\boldsymbol{\sigma}$ represents a uniform random permutation of the indices.

It was shown in [7] that random reshuffling under *decaying* step-sizes can accelerate the convergence rate of stochastic-gradient learning from $O(1/i)$ to $O(1/i^2)$ [8, 9], where i is the iteration index. It is also possible to show [10] that in random reshuffling under small *constant* step-sizes, μ, can boost the steady-state performance of these algorithms from $O(\mu)$-suboptimal to $O(\mu^2)$-suboptimal around a small neighborhood of the exact minimizer [11]. A similar improvement in convergence rate and performance has been observed for the variance-reduced Finito algorithm [3]. However, no formal proofs or guarantees of *exact* convergence exist for the class of variance-reduced algorithms under random reshuffling, i.e., it is still not known whether these types of algorithms are always guaranteed to converge when RR is employed and under what conditions on the data. In [12], another variance-reduction algorithm is proposed under reshuffling; however, no proof of convergence is provided. The closest attempts at proof are the useful arguments given in [13, 14]. The work [13] deals with the case of incremental aggregated gradients, which corresponds to a deterministic version of RR for SAG, while the work [14] deals with SVRG in the context of ridge regression problems using regret analysis.

This paper makes two contributions. First, it resolves this open convergence issue and provides the first theoretical proof and guarantee of linear convergence to the exact minimizer under random reshuffling for SAGA. While the argument is easily adaptable to a wider class of variance-reduced implementations, we illustrate the technique in this work for the SAGA algorithm due to space limitations. A second contribution is that, under random reshuffling, we will propose a new amortized variance-reduced gradient (AVRG) algorithm with two clear benefits: it has constant storage requirements in comparison to SAGA, and it has balanced gradient computations in comparison to SVRG. The balancing in computations is attained by amortizing the full gradient calculation across all iterations. AVRG is also shown analytically to converge linearly.

In preparation for the analysis, we review briefly some of the conditions and notation that are relevant. We consider a generic empirical risk function $J(w) : \mathbb{R}^M \to \mathbb{R}$, which is defined as a sample average of loss values over a possibly large but finite training set of size N:

$$w^\star \;\stackrel{\Delta}{=}\; \underset{w\in\mathbb{R}^M}{\arg\min}\; J(w) \;\stackrel{\Delta}{=}\; \frac{1}{N}\sum_{n=1}^{N} Q(w; x_n), \tag{1}$$

where the $\{x_n\}_{n=1}^N$ represent training data samples.

Assumption 1 (LOSS FUNCTION) *The loss function $Q(w; x_n)$ is convex, differentiable, and has a δ-Lipschitz continuous gradient, i.e., for every $n = 1, \ldots, N$ and any $w_1, w_2 \in \mathbb{R}^M$:*

$$\|\nabla_w Q(w_1; x_n) - \nabla_w Q(w_2; x_n)\| \leq \delta\|w_1 - w_2\| \tag{2}$$

where $\delta > 0$. We also assume that the empirical risk $J(w)$ is ν-strongly convex, namely,

$$\Big(\nabla_w J(w_1) - \nabla_w J(w_2)\Big)^{\mathsf{T}}(w_1 - w_2) \geq \nu\|w_1 - w_2\|^2 \tag{3}$$

$\square$

This work was supported in part by NSF grant CCF-1524250. Emails: {ybc, kunyuan}@ucla.edu, and ali.sayed@epfl.ch

2. SAGA WITH RANDOM RESHUFFLING

We consider the SAGA algorithm [2] in this work, while noting that the convergence analysis can be easily extended to other versions of variance-reduced algorithms; in particular, we shall illustrate how it applies to the new variant AVRG. We list the SAGA algorithm without the proximal step in the table below, and incorporate random reshuffling into the description of the algorithm. In the listing, random quantities are denoted in boldface notation.

SAGA with Random Reshuffling [2]

```
**Initialization**: $w_0^0 = 0, \nabla Q(\phi_{0,n}^0; x_n) = 0,\ n = 1, 2, \dots, N.$
**Repeat** $k = 0, 1, 2 \dots, K$ (epoch):
    generate a random permutation function $\boldsymbol{\sigma}^k(\cdot)$.
    **Repeat** $i = 0, 1, \dots N-1$ (iteration):
        $\boldsymbol{j} = \boldsymbol{\sigma}^k(i+1)$                                         (4)
        $\boldsymbol{w}_{i+1}^k = \boldsymbol{w}_i^k - \mu\Big[\nabla Q(\boldsymbol{w}_i^k; x_{\boldsymbol{j}}) - \nabla Q(\boldsymbol{\phi}_{i,\boldsymbol{j}}^k; x_{\boldsymbol{j}}) + \frac{1}{N}\sum_{n=1}^{N} \nabla Q(\boldsymbol{\phi}_{i,n}^k; x_n)\Big]$   (5)
        $\boldsymbol{\phi}_{i+1,\boldsymbol{j}}^k = \boldsymbol{w}_{i+1}^k$, and $\boldsymbol{\phi}_{i+1,n}^k = \boldsymbol{\phi}_{i,n}^k$, for $n \neq \boldsymbol{j}$   (6)
    **End**
    $\boldsymbol{w}_0^{k+1} = \boldsymbol{w}_N^k,\ \ \boldsymbol{\phi}_0^{k+1} = \boldsymbol{\phi}_N^k$                          (7)
**End**
```

It is seen from the above listing that, for each run k, the original data $\{x_n\}_{n=1}^N$ is randomly reshuffled so that the sample of index $i+1$ becomes the sample of index $\boldsymbol{j} = \boldsymbol{\sigma}^k(i+1)$. To facilitate the understanding of the algorithm, we associate a block matrix $\boldsymbol{\Phi}^k$ with each run. This matrix is only introduced for visualization purposes. We denote the block rows of $\boldsymbol{\Phi}^k$ by $\{\boldsymbol{\phi}_i^k\}$; one for each iteration i, as illustrated in Fig. 1. Each block row $\boldsymbol{\phi}_i^k$ has size $M \times N$. We can therefore view $\boldsymbol{\Phi}^k$ as consisting of cells $\{\boldsymbol{\phi}_{i,n}^k\}$, each having the same $M \times 1$ size as the minimizer $w^\star$. At every iteration i, one random cell in the $(i+1)-$th block row is populated by the iterate $\boldsymbol{w}_{i+1}^k$; the column location of this random cell is determined by the value of $\boldsymbol{j}$.

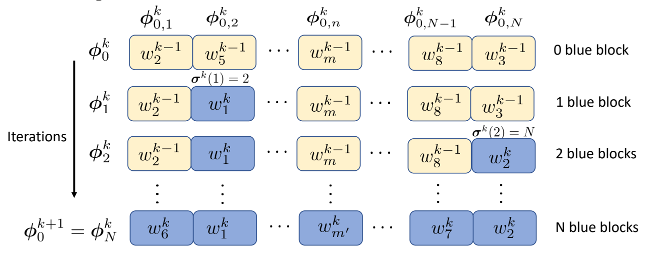

Fig. 1: An illustration of the evolution of $\{\boldsymbol{\phi}_{i,n}^k\}$.

2.1. Properties of the History Variables

Several useful observations can be drawn from Fig. 1.

Observation 1: At the start of each epoch k, the components $\{\boldsymbol{\phi}_{0,n}^k\}_{n=1}^N$ correspond to a permutation of the weight iterates from the previous run, $\{\boldsymbol{w}_i^{k-1}\}_{i=1}^N$.

Observation 2: At the beginning of the $i-$th iteration of an epoch k, all components of indices $\{\boldsymbol{\sigma}^k(m)\}_{m=1}^i$ will be set to weight iterates obtained during the $k-$th run, namely, $\{\boldsymbol{w}_m^k\}_{m=1}^i$, while the remaining $N-i$ history positions will have values from the previous run, namely, $\{\boldsymbol{w}_{t_n}^{k-1}\}_{n=1}^{N-i}$ for some values $t_n \in \{1, 2, \dots, N\}$.

Observation 3: At the beginning of the $i-$th iteration of an epoch k, it holds that

$$\boldsymbol{\phi}_{i,\boldsymbol{j}}^k = \boldsymbol{\phi}_{0,\boldsymbol{j}}^k, \qquad \text{where } \boldsymbol{j} \in \boldsymbol{\sigma}^k(i+1{:}N) \tag{8}$$

where $\boldsymbol{\sigma}^k(i+1{:}N)$ represents the selected indices for future iterations $i+1$ to N. The following result is now possible.

Lemma 1 (Second-order moment of history variable) *The second-order moment of each $\boldsymbol{\phi}_{i,n}^k$ satisfies:*

$$\mathbb{E}\left[\sum_{n=1}^{N} \|\boldsymbol{\phi}_{i,n}^k\|^2\right] = \sum_{n'=1}^{i} \mathbb{E}\,\|\boldsymbol{w}_{n'}^k\|^2 + \frac{N-i}{N}\sum_{n=1}^{N} \mathbb{E}\,\|\boldsymbol{w}_n^{k-1}\|^2 \tag{9}$$

■

For comparison purposes, this result and the previous properties do not hold for implementations that involve sampling the data *with* replacement. For example, property (9) would be replaced by [2]:

$$\mathbb{E}\left[\sum_{n=1}^{N} \|\boldsymbol{\phi}_{i,n}^k\|^2\right] = \mathbb{E}\,\|\boldsymbol{w}_i^k\|^2 + \frac{N-1}{N}\sum_{n=1}^{N} \mathbb{E}\,\|\boldsymbol{\phi}_{i-1,n}^k\|^2, \quad \forall i, k \tag{10}$$

This expression is similar to (9) only for $i = 1$. However, observe that (10) involves variables $\{\boldsymbol{\phi}_{i-1,n}^k\}$ on the right-hand side, instead of the variables $\{\boldsymbol{w}_n^{k-1}\}$ that appear in (9). This is because random reshuffling updates every history variable during each run, while uniform sampling may leave some variables $\boldsymbol{\phi}_{i-1,n}^k$ untouched. This fact helps explain why SAGA with RR tends to have faster convergence rate, as we will illustrate in a later experiment.

2.2. Biased Nature of the Gradient Estimator

Before we examine convergence properties, it is useful to highlight that it is not necessary to insist on unbiased gradient estimators for proper operation of stochastic-gradient algorithms. To see this, let us examine first the SAGA implementation assuming uniform data sampling *with* replacement. In a manner similar to (5), the SAGA algorithm in this case will employ the following modified gradient direction:

$$\widehat{g}_{\boldsymbol{u}}(\boldsymbol{w}_i^k) \triangleq \nabla Q(\boldsymbol{w}_i^k; x_{\boldsymbol{u}}) - \nabla Q(\boldsymbol{\phi}_{i,\boldsymbol{u}}^k; x_{\boldsymbol{u}}) + \frac{1}{N}\sum_{n=1}^{N} \nabla Q(\boldsymbol{\phi}_{i,n}^k; x_n)$$

where the subscript $\boldsymbol{u}$ is used to denote a uniformly distributed random variable, $\boldsymbol{u} \sim \mathcal{U}[1, N]$. As a result, this modified gradient is *unbiased*, i.e., $\mathbb{E}_{\boldsymbol{u}}[\widehat{g}_{\boldsymbol{u}}(\boldsymbol{w}_i^k)|\boldsymbol{\mathcal{F}}_i^k] = \nabla J(\boldsymbol{w}_i^k)$, where $\boldsymbol{\mathcal{F}}_i^k$ denotes the collection of all available information before iteration i at epoch k. However, this property will not hold under random reshuffling! This is because data is now sampled *without* replacement and the selection of one index becomes dependent on the selections made prior to it. Specifically, it will instead hold that

$$\mathbb{E}_{\boldsymbol{j}}\left[\widehat{g}_{\boldsymbol{j}}(\boldsymbol{w}_i^k)|\boldsymbol{\mathcal{F}}_i^k\right] = \frac{1}{N-i}\sum_{n \notin \boldsymbol{\sigma}^k(1:i)} \Big(\nabla Q(\boldsymbol{w}_i^k; x_n) - \nabla Q(\boldsymbol{\phi}_{i,n}^k; x_n)\Big) + \frac{1}{N}\sum_{n=1}^{N} \nabla Q(\boldsymbol{\phi}_{i,n}^k; x_n) \tag{11}$$

where $\boldsymbol{j} = \boldsymbol{\sigma}^k(i+1)$. It is not hard to see that the expression on the right-hand side is generally different from $\nabla J(\boldsymbol{w}_i^k)$. Consequently, the gradient estimate that is employed by SAGA under RR is a *biased* estimator for the true gradient. Nevertheless, we will establish two useful facts in the following sections. First, this gradient estimate becomes asymptotically unbiased when the algorithm converges, as $k \to \infty$. Second, the biased gradient estimation does not harm the convergence rate because we will observe later that SAGA under RR actually converges faster than SAGA in the simulations.

2.3. Convergence Analysis

The following result can now be established, the proof of which is omitted due to space limitations (see [15] for the derivations). Let $\widetilde{\boldsymbol{w}}_0^k \triangleq w^\star - \boldsymbol{w}_0^k$ and introduce the energy function:

$$V_{k+1} \triangleq \mathbb{E}\,\|\widetilde{\boldsymbol{w}}_0^{k+1}\|^2 + \frac{11}{16}\gamma\left(\frac{1}{N}\sum_{i=1}^{N-1}\mathbb{E}\,\|\boldsymbol{w}_i^{k+1} - \boldsymbol{w}_0^{k+1}\|^2 + \frac{1}{N}\sum_{i=1}^{N-1}\mathbb{E}\,\|\boldsymbol{w}_N^k - \boldsymbol{w}_i^k\|^2\right) \tag{12}$$

where $\gamma = 9\mu\delta N$.

Theorem 1 (LINEAR CONVERGENCE OF SAGA) *For sufficiently small step-sizes, namely, for $\mu \le \frac{\nu}{11\delta^2 N}$, the quantity V_{k+1} converges linearly, i.e.*

$$V_{k+1} \le \alpha V_k \tag{13}$$

where

$$\alpha = \frac{1 - \mu\nu N/4}{1 - 27\delta^4\mu^3N^3/\nu} < 1 \tag{14}$$

It follows that $\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_0^k\|^2 \le \alpha^k V_0$. ■

It is worth noting from this result that to achieve an ϵ-optimal solution, the number of iterations required is close to $O(\delta^2/\nu^2)\log(1/\epsilon)$, which is slower than the theorem proved in [2]. The main reason is that the dependency between the samples makes it difficult to obtain a tight bound. As we will observe in the simulations later, in practice, the convergence can be faster than the original SAGA.

3. AMORTIZED VARIANCE-REDUCED GRADIENT (AVRG) LEARNING

One inconvenience of the SAGA implementation is its high storage requirement, which refers to the need to track history variables $\{\boldsymbol{\phi}_{i,n}^k\}$ or gradients for use in (5). There is a need to store $O(N)$ variables. In big data applications, the size of N can be prohibitive. An alternative method is the stochastic variance-reduced gradient (SVRG) algorithm [1]. This method replaces the history variables $\{\boldsymbol{\phi}_{i,\boldsymbol{j}}^k\}$ of SAGA by a fixed initial condition $\boldsymbol{w}_0^k$ for each epoch. This simplification greatly reduces the storage requirement. However, each epoch in SVRG is preceded by an aggregation step to compute a gradient estimate, which is time-consuming for large data sets. It also causes the operation of SVRG to become *unbalanced*, with a larger time interval needed before each epoch, and shorter time intervals needed within the epoch. Motivated by these two important considerations, we propose a new *amortized* implementation, referred to as AVRG. This new algorithm removes the initial aggregation step from SVRG and replaces it by an estimate $\boldsymbol{g}^{k+1}$. This estimate is computed iteratively within the inner loop by re-using the gradient, $\nabla Q(\boldsymbol{w}_i^k; x_{\boldsymbol{j}})$, to reduce complexity. Notice that $\boldsymbol{g}^{k+1}$ is actually calculated in epoch k in order to avoid delaying the computation.

3.1. Useful Properties

Several properties stand out when we compare the proposed AVRG implementation with the previous algorithms. First, observe that the storage requirement for AVRG in each epoch is just the variables $\boldsymbol{g}^k$, $\boldsymbol{g}^{k+1}$, and $\boldsymbol{w}_0^k$, which is similar to SVRG and considerably less than SAGA. A variance-reduced algorithm based on reshuffling is proposed in [12]; however, it still requires extra storage as SAGA.

Second, since the gradient vector $Q(\boldsymbol{w}_i^k; x_{\boldsymbol{j}})$ used in (17) has already been computed in (16), every iteration i will only require two gradients to be evaluated. Thus, the effective computation of gradients per epoch is smaller in AVRG than in SVRG.

AVRG with Random Reshuffling

```
Initialization: $\boldsymbol{w}_0^0 = 0$,  $\boldsymbol{g}^0 = 0$,  $\nabla Q(\boldsymbol{w}_0^0; x_n) \leftarrow 0$, $n = 1, 2, \ldots, N$
Repeat $k = 0, 1, 2 \ldots, K$ (epoch):
    generate a random permutation function $\boldsymbol{\sigma}^k(\cdot)$,
    set $\boldsymbol{g}^{k+1} = 0$
    Repeat $i = 0, 1, \ldots N-1$ (iteration):
        $\boldsymbol{j} = \boldsymbol{\sigma}^k(i+1)$                                        (15)
        $\boldsymbol{w}_{i+1}^k = \boldsymbol{w}_i^k - \mu\left[\nabla Q(\boldsymbol{w}_i^k; x_{\boldsymbol{j}}) - \nabla Q(\boldsymbol{w}_0^k; x_{\boldsymbol{j}}) + \boldsymbol{g}^k\right]$    (16)
        $\boldsymbol{g}^{k+1} \leftarrow \boldsymbol{g}^{k+1} + \frac{1}{N}\nabla Q(\boldsymbol{w}_i^k; x_{\boldsymbol{j}})$                    (17)
    End
    $\boldsymbol{w}_0^{k+1} = \boldsymbol{w}_N^k$                                        (18)
End
```

Third, observe from Eq. (17) how the estimated $\boldsymbol{g}^{k+1}$ is computed by averaging the loss values at successive iterates. This construction is feasible because of the use of random reshuffling. Under random reshuffling, the collection of gradients $\{Q(\boldsymbol{w}_i^k; x_{\boldsymbol{j}})\}$ that are used in (17) during each epoch will end up covering the entire set of data, $\{x_n\}_{n=1}^N$. This is not necessarily the case for operation under uniform sampling with replacement. Therefore, the AVRG procedure assumes the use of random reshuffling. We will simply refer to it as AVRG, rather than AVRG under RR.

Fourth, unlike the SVRG algorithm, which requires a step to compute the full gradient, the AVRG implementation is amenable to decentralized implementations (i.e., to fully-decentralized implementations with no master nodes), and also to asynchronous operation [16]. The unbalanced gradient computation in SVRG poses difficulties for decentralized solutions and introduces idle times when multiple devices/agents with different amounts of data cooperate to solve an optimization problem. We will address these challenges in another work [17].

Finally, the modified gradient direction that is employed in (16) by AVRG has distinctive properties in relation to the modified gradient direction (5) in SAGA. To see this, we note that the gradient direction in (16) can be written as

$$\widehat{g}_{\boldsymbol{j}}(\boldsymbol{w}_i^k) \triangleq \nabla Q(\boldsymbol{w}_i^k; x_{\boldsymbol{j}}) - \nabla Q(\boldsymbol{w}_0^k; x_{\boldsymbol{j}}) + \frac{1}{N}\sum_{n=0}^{N-1}\nabla Q(\boldsymbol{w}_n^{k-1}; x_{\boldsymbol{\sigma}^{k-1}(n+1)}) \tag{19}$$

It is clear that even when the index $\boldsymbol{j}$ is chosen uniformly, the above vector cannot be an unbiased estimator for the true gradient in general. What is more critical for convergence is that the modified gradient direction of an algorithm should satisfy the useful property that, as the weight iterate gets closer to the optimal value, i.e., as $\|w^\star - \boldsymbol{w}_i^k\| \le \epsilon$ for arbitrary small ϵ and large enough k, the modified and true gradients will also get arbitrarily close to each. This property holds for (19) since

$$\begin{aligned}&\|\widehat{g}_{\boldsymbol{j}}(\boldsymbol{w}_i^k) - \nabla J(w^\star)\|\\ &\le \delta\|\boldsymbol{w}_i^k - w^\star\| + \delta\|\boldsymbol{w}_0^k - w^\star\| + \frac{\delta}{N}\sum_{n=1}^{N-1}\left\|\boldsymbol{w}_{n-1}^{k-1} - w^\star\right\|\\ &\le 3\delta\epsilon\end{aligned} \tag{20}$$

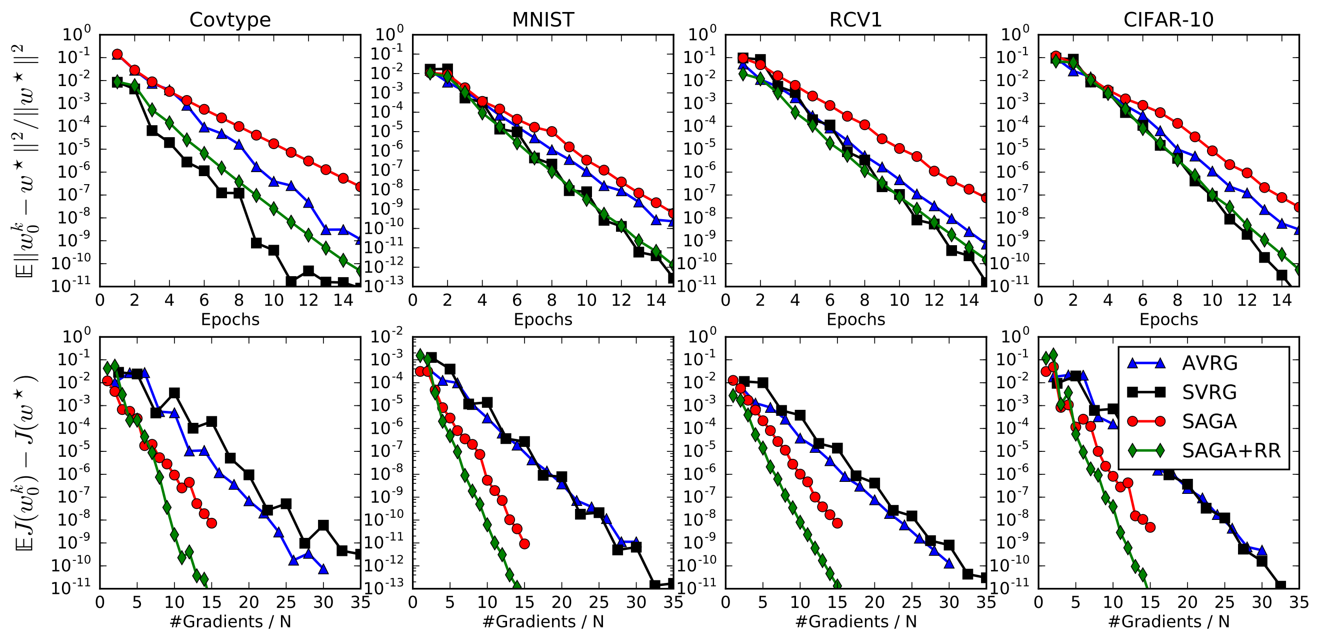

Fig. 2: Comparison of various variance-reduced algorithms over four datasets: Covtype, MNIST, RCV1, and CIFAR-10. The top four plots compare the relative mean-square-error performance versus the epoch index, k, while the bottom four plots compare the excess risk values versus the number of gradients computed.

Table 1: Comparison of the variance-reduced implementations.

	SVRG	AVRG	SAGA	SAGA+RR
grad. comp. per epoch	$2.5N$	$2N$	N	N
extra storage req.	$O(1)$	$O(1)$	$O(N)$	$O(N)$
balanced grad. comp.	No	Yes	Yes	Yes
unbiased grad. est.	Yes	No	Yes	No

where in the second inequality we exploited Jensen's inequality, the triangle inequality, Lipschitz assumption, and the fact that $\boldsymbol{\sigma}^{k-1}(n)$ corresponds to sampling without replacement. Because ϵ can be chosen arbitrary small, then $\widehat{g}_{\boldsymbol{j}}(\boldsymbol{w}_i^k)$ must approach the true gradient at $w^\star$. This result also implies the aforementioned asymptotic unbiasedness property of the gradient estimate. For ease of reference, Table 1 compares the trade-offs between storage and computational complexity of different variance-reduced algorithms. The same approach used to establish the convergence of SAGA under RR is also suitable for AVRG.

Theorem 2 (Linear convergence of AVRG) *For the step-sizes satisfying $\mu \le \frac{\nu}{9\delta^2 N}$, the quantity V_{k+1} converges linearly:*

$$V_{k+1} \le \alpha V_k \tag{21}$$

where

$$\alpha = \frac{1-\mu\nu N/4}{1-18\delta^3\mu^3N^3/\nu} < 1 \tag{22}$$

and

$$V_{k+1} \triangleq \mathbb{E}\,\|\widetilde{\boldsymbol{w}}_0^{k+1}\|^2 + \tag{23}$$
$$\frac{13}{16}\gamma\left(\frac{1}{N}\sum_{i=1}^{N-1}\mathbb{E}\,\|\boldsymbol{w}_i^{k+1}-\boldsymbol{w}_0^{k+1}\|^2+\frac{1}{N}\sum_{i=1}^{N-1}\mathbb{E}\,\|\boldsymbol{w}_N^{k}-\boldsymbol{w}_i^{k}\|^2\right)$$

■

This is similar to the theorem for SAGA under RR. However, in practice, AVRG will perform differently from SAGA under RR.

4. SIMULATION RESULTS

In this section, we illustrate the convergence performance of various algorithms by numerical simulations. We consider the following regularized logistic regression problem:

$$\min_w \quad J(w) = \frac{1}{N}\sum_{n=1}^{N} Q(w; h_n, \gamma(n)) \tag{24}$$
$$\triangleq \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\rho}{2}\|w\|^2 + \ln\left(1+\exp(-\gamma(n)h_n^{\mathsf T}w)\right)\right)$$

where $h_n \in \mathbb{R}^M$ is the feature vector, $\gamma(n)\in\{\pm 1\}$ is the class label. In all our experiments, we set $\rho = 1/N$. The optimal $w^\star$ and the corresponding risk value are calculated by means of the Scikit-Learn package. We run simulations over four datasets: covtype.binary[1], rcv1.binary[1], MNIST[2], and CIFAR-10[3]. The last two datasets have been transformed into binary classification problems by considering data with labels 0 and 1, i.e., digital zero and one classes for MNIST and airplane and automobile classes for CIFAR-10. All features have been preprocessed and normalized to the unit vector [18]. The results are exhibited in Fig. 2. To enable fair comparisons, we tune the step-size parameter of each algorithm for fastest convergence in each case. The plots are based on measuring the relative mean-square-error, $\mathbb{E}\,\|\boldsymbol{w}_0^k - w^\star\|^2/\|w^\star\|^2$, and the excess risk value, $\mathbb{E}\,J(\boldsymbol{w}_0^k) - J(w^\star)$. Two key facts to observe from these simulations are that 1) SAGA with RR is consistently faster than SAGA, and 2) without the high memory cost of SAGA and without the unbalanced structure of SVRG, the proposed AVRG technique is able to match their performance reasonably well.

[1]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
[2]http://yann.lecun.com/exdb/mnist/
[3]http://www.cs.toronto.edu/~kriz/cifar.html

5. REFERENCES

[1] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proc. Advances in Neural Information Processing Systems* (NIPS), Lake Tahoe, Navada, 2013, pp. 315–323.

[2] A. Defazio, F. Bach, and S. Lacoste-Julien, "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Proc. Advances in Neural Information Processing Systems* (NIPS), Montreal, Canada, 2014, pp. 1646–1654.

[3] A. Defazio, J. Domke, and T. S. Caetano, "Finito: A faster, permutable incremental gradient method for big data problems.," in *Proc. International Conference of Machine Learning* (ICML), Beijing, China, 2014, pp. 1125–1133.

[4] N. L. Roux, M. Schmidt, and F. R. Bach, "A stochastic gradient method with an exponential convergence rate for finite training sets," in *Proc. Advances in Neural Information Processing Systems* (NIPS), Lake Tahoe, Navada, 2012, pp. 2663–2671.

[5] L. Bottou, "Curiously fast convergence of some stochastic gradient descent algorithms," in *Proc. Symposium on Learning and Data Science*, Paris, 2009, pp. 1–5.

[6] B. Recht and C. Ré, "Toward a noncommutative arithmetic-geometric mean inequality: Conjectures, case-studies, and consequences," in *Proc. Conference on Learning Theory* (COLT), Edinburgh, Scotland, 2012, pp. 1–11.

[7] M. Gürbüzbalaban, A. Ozdaglar, and P. Parrilo, "Why random reshuffling beats stochastic gradient descent," *arXiv:1510.08560*, Oct. 2015.

[8] Y. Nesterov, *Introductory Lectures on Convex Optimization: A basic course*, vol. 87, Springer, 2013.

[9] B. T. Polyak, *Introduction to Optimization*, Optimization Software, NY, 1987.

[10] B. Ying, B. Yuan, S. Vlaski, and A. H. Sayed, "Stochastic learning under random reshuffling," *submitted for publication*, 2018.

[11] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. 4–5, pp. 311–801, 2014.

[12] S. De and T. Goldstein, "Efficient distributed SGD with variance reduction," in *Proc. IEEE International Conference on Data Mining* (ICDM), Barcelona, Spain, 2016, pp. 111–120.

[13] M. Gürbüzbalaban, A. Ozdaglar, and P. Parrilo, "On the convergence rate of incremental aggregated gradient algorithms," *SIAM Journal on Optimization*, vol. 27, pp. 1035–1048, Jun. 2017.

[14] O. Shamir, "Without-replacement sampling for stochastic gradient methods: Convergence results and application to distributed optimization," *arXiv:1603.00570*, Mar. 2016.

[15] B. Ying, K. Yuan, and A. H. Sayed, "Variance-reduced stochastic learning under random reshuffling," *available on arXiv:1708.01383*, August 2017.

[16] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. J. Smola, "On variance reduction in stochastic gradient descent and its asynchronous variants," in *Advances in Neural Information Processing* (NIPS), pp. 2647–2655. Montréal, Canada, 2015.

[17] K. Yuan, B. Ying, and A. H. Sayed, "Variance-reduced stochastic learning by networked agents under random reshuffling," *available on arXiv:1708.01384*, August 2017.

[18] L. Xiao and T. Zhang, "A proximal stochastic gradient method with progressive variance reduction," *SIAM Journal on Optimization*, vol. 24, no. 4, pp. 2057–2075, 2014.