Adaptive Stochastic Variance Reduction for Non-convex Finite-Sum Minimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

We propose an adaptive variance-reduction method, called AdaSpider, for minimization of L-smooth, non-convex functions with a finite-sum structure. In essence, AdaSpider combines an AdaGrad-inspired [16], but a fairly distinct, adaptive step-size schedule with the recursive *stochastic path integrated estimator* proposed in Fang et al. [19]. To our knowledge, AdaSpider is the first parameter-free non-convex variance-reduction method in the sense that it does not require the knowledge of problem-dependent parameters, such as smoothness constant L, target accuracy ϵ or any bound on gradient norms. In doing so, we are able to compute an ϵ-stationary point with $\tilde{O}\left(n + \sqrt{n}/\epsilon^2\right)$ oracle-calls, which matches the respective lower bound up to logarithmic factors.

1 Introduction

This paper studies smooth, non-convex minimization problems with the following finite-sum structure:

$$\min_{x \in \mathbb{R}^d} f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x), \tag{Prob}$$

where each component function $f_i : \mathbb{R}^d \to \mathbb{R}$ is L-smooth and is possibly non-convex, and we further assume f is also non-convex. We seek to find an ϵ-approximate first-order stationary point $\hat{x}$ of f, such that $\|\nabla f(\hat{x})\| \leq \epsilon$, where $\epsilon > 0$ is the accuracy of the desired solution.

This structure captures many interesting learning problems from empirical risk minimization to training of neural networks. *First-order methods* have been the standard choice for solving (Prob), due to their efficiency and favorable practical behavior. In that regard, while gradient descent (GD) requires $O(n/\epsilon^2)$ gradient computations, stochastic gradient descent (SGD) requires $O(1/\epsilon^4)$ overall gradient computations. In many interesting machine learning applications n tends to be large, e.g., training a neural network for image classification with very big image datasets [14], hence SGD typically leads to better practical performance.

To leverage the best of both regimes, GD and SGD, the so-called variance reduction (VR) framework combines the *faster convergence rate* of GD with the *low per-iteration complexity* of SGD. Originally proposed for solving strongly-convex problems [23, 13, 37], variance reduction frameworks essentially generate low-variance gradient estimates by maintaining a balance between periodic full gradient computations and stochastic (mini-batch) gradients. VR methods and their theoretical behavior for *convex problems* have been well-studied under various problem setups and assumptions, including μ-strongly convex functions with $O(n + (L/\mu)\log(1/\epsilon))$ complexity [23, 37, 13]; μ-strongly convex functions with accelerated $O(n + \sqrt{L/\mu}\log(1/\epsilon))$ complexity [2, 27, 41] and smooth, convex functions with $\tilde{O}(n + 1/\epsilon)$ complexity [50, 41, 15]. For non-convex minimization, earlier attempts extended the existing VR frameworks, achieving the first rates of order $O(n + n^{2/3}/\epsilon^2)$ with sub-optimal

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

dependence on n [39, 47, 1, 32]. The most recent non-convex VR methods [18, 44, 33, 38, 34] close this gap and achieve the optimal gradient oracle complexity of $O(n + \sqrt{n}/\epsilon^2)$ [18].

Adaptivity and First-order Optimization

The selection of the step-size is of great importance in both the theoretical and practical performance of first-order methods, including the aforementioned VR methods. In the case of L-smooth minimization, first-order methods need the knowledge of L so as to adequately select their step-size [36], otherwise the method is not guaranteed to be convergent and might even diverge [15, 35]. To elucidate, classical analysis relies on the (expected) descent property and guarantees that the algorithm monotonically makes progress every iteration. To enforce this property everywhere on the optimization landscape, one needs to pick the step-size as $\gamma_t \leq O(1/L)$, which restricts the step length of the algorithm with respect to the worst-case constant L. On the other hand, estimating the smoothness constant for an objective of interest, such as neural networks, is a very hard task [20]. At the same time, using crude bounds on the smoothness constant leads to very small step-sizes and consequently to poorer convergence. In practice the step-size is tuned through an empirical search over a range of hand-picked values that adds a considerable computational overhead and burden. In order to alleviate the burden of tuning process, we need step-sizes that adjust in accordance with the optimization path.

A popular line of research studies first-order methods that *adaptively* select their step-size by taking advantage of the previously produced point. In many settings of interest, these *adaptive methods* are able to guarantee optimal convergence rates while they often admit superior empirical performance due to their ability to decrease the step-size according to the local geometry of the objective function. Inspired by the seminal work AdaGrad [16], several adaptive gradient methods have been recently proposed such that, given access to *noiseless gradient-estimates*, achieve accelerated rates in the case of L-smooth convex minimization [29, 25, 24, 17]. Similarly, Ene et al. [17] and Antonakopoulos et al. [5] propose adaptive methods with optimal convergence rates for monotone variational inequalities while Antonakopoulos et al. [6] improves upon the $O(1/\sqrt{T})$ rate when the noise has a multiplicative nature that vanishes around a solution. Hsieh et al. [22] and Vu et al. [43] study the convergence properties of adaptive first-order methods for routing and generic games. With a great relevance to our method, Levy et al. [30] propose an adaptive method with optimal convergence guarantees in the case of *expectation over smooth losses*[1], removing the requirement of the knowledge on problem parameters (e.g., smoothness constant, absolute bounds on gradient norms) by [12].

Adaptivity and Finite-Sum minimization

In parallel with what we discussed earlier, existing variance-reduction methods (VR) crucially need to know the smoothness constant L to select their step-size appropriately to guarantee their convergence. To this end, the following natural question arises

*Can we design **adaptive** VR methods that achieve the optimal gradient computation complexity?*

Li et al. [31] and Tan et al. [42] were the first to propose adaptive variance-reduction methods by using the Barzilai-Borwein step-size [8]. Despite their promising empirical performance, these methods do not admit formal convergence guarantees. When the objective function f in (Prob) is convex, [15] recently proposed an adaptive VR method requiring $\tilde{O}(n + 1/\epsilon)$ gradient computation while, shortly after, [35] proposed an *accelerated* adaptive VR method requiring $\tilde{O}(n + \sqrt{n}/\sqrt{\epsilon})$ gradient computations.

To the best of our knowledge, there is no adaptive VR method in the case where f is *non-convex*. We remark that f being non-convex captures the most interesting settings such as minimizing the empirical loss of deep neural network where each f_i stands for the loss with respect to i-th data point and thus is a non-convex function in the parameters of the neural architecture. Through this particular example, we could motivate adaptive VR methods in two fronts: first, even estimating the smoothness constant L of a deep neural network is prohibitive [20], and at the same time, the parameter n in (Prob) equals the number of data samples, which can be very large in practice and is prohibitive for the use of deterministic methods.

[1] This is a slightly more general setting than finite-sum minimization at which the objective function $f(x) := \mathbb{E}_{\xi\sim\mathcal{D}} g(x, \xi)$ with $g(x, \xi)$ being L-smooth with respect to x for all ξ. This more general setting comes with worse lower bounds than the finite-sum setting. More precisely, in order to compute an ϵ-stationary point at least $\Omega(1/\epsilon^3)$ gradient-estimates are needed [7]. We remark that in the case of finite-sum minimization there exist variance reduction methods with $O(n + \sqrt{n}/\epsilon^2)$ gradient complexity [19, 44].

Table 1: In the following table we present the gradient computation complexity of the existing non-adaptive and adaptive variance reduction methods for both convex and non-convex finite-sum minimization. Since for there are multiple non-adaptive VR methods, we present the earliest-proposed method matching up to logarithmic factors the respective lower bounds.

$f(x)$	**Non-Adaptive VR**	**Adaptive VR**	**Complexity Lower Bound**
convex (ϵ-*opt. solution*)	$\tilde{O}\left(n+\sqrt{\frac{n}{\epsilon}}\right)$ [26]	$\tilde{O}\left(n+\sqrt{\frac{n}{\epsilon}}\right)$ [35]	$\Omega(n+\sqrt{\frac{n}{\epsilon}})$ [45]
convex (ϵ-*opt. solution*)	$\tilde{O}\left(n+\frac{1}{\epsilon}\right)$ [49]	$\tilde{O}\left(n+\frac{1}{\epsilon}\right)$ [15]	$\Omega(n+\sqrt{\frac{n}{\epsilon}})$ [45]
non-convex (ϵ-*stat. point*)	$O\left(n+\frac{\sqrt{n}}{\epsilon^2}\right)$ [19]	$\tilde{O}\left(n+\frac{\sqrt{n}}{\epsilon^2}\right)$ This work	$\Omega\left(n+\frac{\sqrt{n}}{\epsilon^2}\right)$ [19]

Contribution and Techniques In this work we present an adaptive VR method, called AdaSpider, that converges to an ϵ-*stationary point* for (Prob) by using $\tilde{O}(n+\sqrt{n}L^2/\epsilon^2)$ gradient computations. Our gradient complexity bound matches the existing lower bounds up to logarithmic factors and its dependence on L [19].

AdaSpider combines an adaptive step-size schedule in the lines proposed by AdaGrad [16] with the variance-reduction mechanism based on the *stochastic path integrated differential estimator* of the Spider algorithm [19]. More precisely, AdaSpider selects the step-size by aggregating the norm of its recursive estimator, while following a single-loop structure as in Fang et al. [19].

Our contributions and techniques can be summarized as follows:

- To our knowledge, AdaSpider is *the first* parameter-free method in the sense that it is both *accuracy-independent* and is oblivious to the knowledge of any problem parameters including L. Moreover, ϵ-independence enables us to provide *any-iterate* guarantees. While Spider needs both ϵ and L to set its step-size as $\min(\frac{\epsilon}{L\sqrt{n}\|\nabla_t\|}, \frac{1}{2\sqrt{n}L})$ to achieve optimal gradient complexity [19], all other existing non-convex methods must know at least the value of L in order to guarantee convergence [4, 44].
- We introduce a novel step-size schedule $\gamma_t := n^{-1/4}\left(\sqrt{n}+\sum_{s=0}^{t}\|\nabla_s\|^2\right)^{-1/2}$ where ∇_s is the recursive variance-reduced estimator at round s. By identifying a unique additive/multiplicative form for integrating n, we manage to achieve optimal dependence on the number of components.
- We show how to combine the above adaptive step-size schedule with the recursive Spider estimator in order to ensure that the *average variance* $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla_t-\nabla f(x_t)\|\right]$ decays at a rate $\tilde{O}\left(n^{1/4}/\sqrt{T}\right)$. This might be of independent interest for other variance reduction techniques.

We follow a novel technical path that uses the adaptivity of the step-size to bound the overall variance of the process. This fact differentiates our approach from the previous adaptive and non adaptive VR approaches (see Section 4 for further details) and provides us a surprisingly concise analysis.

2 Setup and Preliminaries

During the whole of this manuscript, we consider that the non-convex objective function $f:\mathbb{R}^d\mapsto\mathbb{R}$ possesses a finite-sum structure

$$f(x)=\frac{1}{n}\sum_{i=1}^{n}f_i(x)$$

where each component function f_i is L-*smooth* (or alternatively has L-*Lipschitz gradient*) and (possibly) non-convex. To quantify the performance of our algorithm within the context of non-convex minimization, we want to find an ϵ-*first order stationary point* $\hat{x}\in\mathbb{R}^d$ such that

$$\|\nabla f(\hat{x})\|\leq\epsilon.$$

For notational simplicity we define $\|\cdot\|$ as the Euclidean norm. Then, we say that a continuously differentiable function f is L-*smooth* if

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \tag{2.1}$$

which admits the following equivalent form,

$$f(x) \leq f(y) + \nabla f(y)^\top (x - y) + \frac{L}{2}\|x - y\|^2 \quad \text{for all } x, y \in \mathbb{R}^d. \tag{2.2}$$

Observe that smoothness of each component immediately suggests that objective f is L-smooth itself. Since we are studying randomized algorithms for finite-sum minimization problems, we do not consider any variance bounds on the gradients of components. We only assume that we have access to an oracle which returns the gradient of individual components when queried.

We finally state two technical lemmas that have been extensively used in the analysis of adaptive methods, the proofs of which can be found in [29].

Lemma 1. *Let a sequence of non-negative real numbers* $\alpha_1, \ldots, \alpha_T \geq 0$ *then*

$$\sum_{t=1}^{T} \frac{\alpha_t}{\sqrt{\sum_{s=1}^{t} \alpha_s}} \leq 2\sqrt{\sum_{t=1}^{T} \alpha_t}$$

Lemma 2. *Let a sequence of non-negative real numbers* $\alpha_1, \ldots, \alpha_T \geq 0$ *then*

$$\sum_{t=1}^{T} \frac{\alpha_t}{1 + \sum_{s=1}^{t} \alpha_s} \leq \log\left(1 + \sum_{t=1}^{T} \alpha_t\right)$$

3 Adaptive SPIDER algorithm and convergence results

In this section, we present our adaptive variance reduction method, called AdaSpider (Algorithm 1) which exploits the variance reduction properties of the *stochastic path integrated differential estimator* proposed in [19] while combining it with an AdaGrad-type step-size construction [16]. Unlike the original Spider method [19], our algorithm admits anytime guarantees, i.e., we don't need to specify the accuracy ϵ a priori. Additionally, our algorithm does not need to know the smoothness parameter L and guarantees convergence without any tuning procedure.

Algorithm 1 Adaptive SPIDER (AdaSpider)

Input: $x_0 \in \mathbb{R}^d$

$G \leftarrow 0$
for $t = 0, ..., T-1$ **do**
  if $t \bmod n = 0$ **then**
    $\nabla_t \leftarrow \nabla f(x_t)$
  else
    pick $i_t \in \{1, \ldots, n\}$ uniformly at random
    $\nabla_t \leftarrow \nabla f_{i_t}(x_t) - \nabla f_{i_t}(x_{t-1}) + \nabla_{t-1}$
  end if
  $\gamma_t \leftarrow 1/\left(n^{1/4}\sqrt{n^{1/2} + \sum_{s=0}^{t}\|\nabla_s\|^2}\right)$
  $x_{t+1} \leftarrow x_t - \gamma_t \cdot \nabla_t$
end for
return uniformly at random $\{x_0, \ldots, x_{T-1}\}$.

As Algorithm 1 indicates, AdaSpider performs a full-gradient computation every n iterations while in the rest iterations updates the variance-reduced gradient estimator in a recursive manner, $\nabla_t \leftarrow \nabla f_{i_t}(x_t) - \nabla f_{i_t}(x_{t-1}) + \nabla_{t-1}$. The adaptive nature of AdaSpider comes from the selection

of the step-size at Step 9 that only depends on the norms of estimates produced by the algorithm in the previous steps.

Before presenting the formal convergence guarantees of AdaSpider (stated in Theorem 1), we present the cornerstone idea behind its design and motivate the analysis for controlling the overall variance of the process through the *adaptivity of the step-size*. This conceptual novelty differentiates our work form the previous adaptive VR methods [15, 35] at which the adaptivity of the step-size is only used for adapting to the smoothness constant L, and their constructions come with additional challenges in bounding the variance. As a result, the following challenge is the first to be tackled by the design of a VR method.

Challenge 1. *Does the average variance of the estimator,* $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla_t - \nabla f(x_t)\|\right]$*, diminishes at a sufficiently fast rate?*

Up next we explain why combining the variance-reduction estimator of Step 7 with the adaptive step-size of Step 9 provides a surprisingly concise answer to Challenge 1. We remark that Spider is able to control the variance at any iterations by choosing $\gamma_t := \min(\frac{\epsilon}{L\sqrt{n}\|\nabla_t\|}, \frac{1}{2\sqrt{n}L})$ as step-size. The latter enforces the method to make tiny steps, $\|x_t - x_{t-1}\| \leq \epsilon/L\sqrt{n}$ which results in ϵ-bounded variance at any iteration. The latter proposed SpiderBoost [44] provides the same gradient-complexity bounds with Spider but through the *accuracy-independent* step-size $\gamma = 1/L$. SpiderBoost handles Challenge 1 by using a dense gradient-computations schedule[2] combined with amortization arguments based on the descent inequality (this is why the knowledge of L is necessary in its analysis). We remark that AdaSpider, despite being oblivious to L and accuracy ϵ, admits a significantly simpler analysis by exploiting the adaptability of its step-size.

In the rest of the section we present our approach to Challenge 1 and we conclude the section with Theorem 1 stating the formal convergence guarantees of AdaSpider.

Handling the variance with adaptive step-size We start with the following variance aggregation lemma that is folklore in (VR) literature (e.g. [48]).

Lemma 3. *Define the gradient estimator at point x as $\nabla_x := \nabla f_i(x) - \nabla f_i(y) + \nabla_y$ where i is sampled uniformly at random from $\{1, \ldots, n\}$. Then,*

$$\mathbb{E}\left[\|\nabla_x - \nabla f(x)\|^2\right] \leq L^2\|x - y\|^2 + \mathbb{E}\left[\|\nabla_y - \nabla f(y)\|^2\right]$$

Now, let us apply Lemma 3 on Spider estimator, $\nabla_t := \nabla f_{i_t}(x_t) - \nabla f_{i_{t-1}}(x_t) + \nabla_{t-1}$ to measure its variance at step x_t.

$$\begin{aligned}
\mathbb{E}\left[\|\nabla_t - \nabla f(x_t)\|^2\right] &\leq L^2\mathbb{E}\left[\|x_t - x_{t-1}\|^2\right] + \mathbb{E}\left[\|\nabla_{t-1} - \nabla f(x_{t-1})\|^2\right] \\
&\leq L^2\mathbb{E}\left[\gamma_{t-1}^2\|\nabla_{t-1}\|^2\right] + \mathbb{E}\left[\|\nabla_{t-1} - \nabla f(x_{t-1})\|^2\right] \\
&\leq L^2\mathbb{E}\left[\gamma_{t-1}^2\|\nabla_{t-1}\|^2\right] + \ldots + \mathbb{E}\left[\|\nabla_{t-(t \bmod n)} - \nabla f(x_{t-(t \bmod n)})\|^2\right] \\
&= \sum_{\tau=t-(t \bmod n)+1}^{t-1} L^2\mathbb{E}\left[\gamma_\tau^2 \cdot \|\nabla_\tau\|^2\right]
\end{aligned}$$

where the last equality follows by the fact $\mathbb{E}\left[\|\nabla_{t-(t \bmod n)} - \nabla f(x_{t-(t \bmod n)})\|^2\right] = 0$ since Algorithm 1 performs a full-gradient computations for every t with $t \bmod n = 0$ (Step 3 of Algorithm 1). By telescoping the summation we get,

$$\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla_t - \nabla f(x_t)\|^2\right] \leq \sum_{t=0}^{T-1}\sum_{\tau=t-(t \bmod n)+1}^{t-1} L^2\mathbb{E}\left[\gamma_\tau^2 \cdot \|\nabla_\tau\|^2\right] \leq L^2 n \cdot \sum_{t=0}^{T-1}\mathbb{E}\left[\gamma_t^2 \cdot \|\nabla_t\|^2\right]$$

where the n factor on the right-hand side is due to the fact that each term $\mathbb{E}\left[\gamma_t^2\|\nabla_t\|^2\right]$ appears at most n times in the total summation. To this end, using the structure of the *stochastic path integrated differential estimator* we have been able to bound the overall variance of the process as follows,

$$\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla_t - \nabla f(x_t)\|^2\right] \leq L^2 n \cdot \sum_{t=0}^{T-1}\mathbb{E}\left[\gamma_t^2 \cdot \|\nabla_t\|^2\right]$$

[2] AdaSpider computes a full-gradient every $\sqrt{n}$ steps and at the intermediate steps uses batches of size $\sqrt{n}$.

However it is not clear at all why the above bound is helpful. At this point the adaptive selection of the step-size (Step 9 in Algorithm 1) comes into play by providing the following surprisingly simple answer,

$$\sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla_t - \nabla f(x_t)\|^2\right] \leq L^2 n\, \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma_t^2 \cdot \|\nabla_t\|^2\right]$$
$$= \quad L^2\sqrt{n}\, \mathbb{E}\left[\sum_{t=0}^{T-1} \frac{\|\nabla_t\|^2}{\sqrt{n} + \sum_{s=0}^{t}\|\nabla_s\|^2}\right] \leq L^2\sqrt{n} \log\left(1 + \mathbb{E}\left[\sum_{t=0}^{T-1}\|\nabla_t\|^2\right]\right)$$

where the last inequality comes from Lemma 2. To finalize the bound, we require the following expression that follows by the fact that $\gamma_t \leq 1/\|\nabla_t\|$ and thus $\|x_t - x_{t-1}\| \leq 1$.

Lemma 4. *Let $x_0, x_1, \ldots, x_{T-1}$ be the sequence produced by Algorithm 1. Then,*

$$\sum_{t=0}^{T-1} \|\nabla_t\|^2 \leq 10 L^2 n^2 T^3 \cdot \|\nabla f(x_0)\|^2.$$

In simple terms, Lemma 4 helps us avoid the *bounded gradient norm* assumption that is common among adaptive non-convex methods. As a result, ADASPIDER admits the following cumulative variance bound,

$$\sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla_t - \nabla f(x_t)\|^2\right] \leq O\left(L^2\sqrt{n} \log\left(nLT \cdot \|\nabla f(x_0)\|\right)\right). \tag{3.1}$$

Remark 1. To this end one might notice that using a more aggressive n dependence on γ_t leads to smaller variance of the estimator which is obviously favorable. However more aggressive dependence on n leads to smaller step-sizes and thus to sub-optimal overall gradient-computation complexity. In Section 4, we explain why the optimal way to inject the n dependence into the step-size is through the simultaneous multiplicative/additive way described in Step 9 of ADASPIDER that may seem unintuitive on the first sight.

We will conclude this discussion with a complementary remark on the interplay between our adaptive step-size and the convergence rate. As we demonstrated in Eq. (3.1), using a data-adaptive step-size leads to a decreasing variance bound *in an amortized sense* as opposed to *any iterate* variance bound of SPIDER. The trade-off in our favor is the parameter-free step-size that is independent of ϵ and L. For a fair exposition of our results, notice that the aforementioned advantages of an adaptive step-size comes at an additional $\log(T)$ term in our final bound due to Eq. (3.1). This has a negligible effect on the convergence as even in the large iteration regime when T is in the order billions, it amounts to a small constant factor.

We conclude the section with Theorem 1 that formally establishes the convergence rate of ADASPIDER. The proof of Theorem 1 is deferred for the next section.

Theorem 1. *Let $x_0, x_1, \ldots, x_{T-1}$ be the sequence produced by Algorithm 1 in case $f(\cdot)$ is L-smooth. Let us also define $\Delta_0 := f(x_0) - f^*$. Then,*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla f(x_t)\|\right] \leq O\left(n^{1/4} \cdot \frac{\Delta_0 + L^2 \cdot \log\left(nLT\|\nabla f(x_0)\|\right)}{\sqrt{T}}\right)$$

Overall, Algorithm 1 needs at most $\tilde{O}\left(n + \sqrt{n} \cdot \frac{\Delta_0^2 + L^4}{\epsilon^2}\right)$ oracle calls to reach an ϵ-stationary point.

4 Sketch of Proof of Theorem 1

In this section we present the key steps for proving Theorem 1. We first use the triangle inequality to derive,

$$\sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla f(x_t)\|\right] \leq \sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla_t - \nabla f(x_t)\|\right] + \sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla_t\|\right]$$

We have previously discussed how to bound the first term in Section 3. More precisely, by the Jensen's inequality and the arguments presented in Section 3, we obtain the following variance bound.

Lemma 5. *Let $x_0, x_1 \ldots, x_T$ the sequence of points produced by Algorithm 1. Then,*

$$\mathbb{E}\left[\sum_{t=0}^{T-1} \|\nabla_t - \nabla f(x_t)\|\right] \leq \tilde{O}\left(n^{1/4} \cdot \sqrt{T}\right).$$

We continue with presenting how to treat the term $\sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla_t\|\right]$. By the smoothness of the function and through a telescopic summation one can easily establish the following bound,

$$\mathbb{E}\left[\sum_{t=0}^{T-1} \gamma_t \cdot \|\nabla_t\|^2\right] \leq 2(f(x_0) - f^*) + L \cdot \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma_t^2 \cdot \|\nabla_t\|^2\right] + \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma_t \cdot \|\nabla f(x_t) - \nabla_t\|^2\right]$$

As we explained in Section 3, the term $\mathbb{E}\left[\sum_{t=0}^{T-1} \gamma_t^2 \cdot \|\nabla_t\|^2\right]$ can be upper bounded by the adaptability of the step-size γ_t. At the same time, using the adaptability of γ_t we are able to establish that $\sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla_t\|\right]$ is at most $n^{1/4}\sqrt{T} \cdot \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma_t \cdot \|\nabla_t\|^2\right]$. All the above are formally stated and established in Lemma 6 the proof of which is deferred to the appendix.

Lemma 6. *Let $x_0, x_1, \ldots$ the sequence of points produced by Algorithm 1 and $\Delta_0 := f(x_0) - f^*$. Then,*

$$\mathbb{E}\left[\sum_{t=0}^{T-1} \|\nabla_t\|\right] \leq \tilde{O}\left(\Delta_0 + L^2 + \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma_t \|\nabla f(x_t) - \nabla_t\|^2\right]\right) \cdot n^{1/4}\sqrt{T}.$$

Due to the use of the adaptive step-size γ_t, the *estimator's error* and the step-size itself are dependent random variables, meaning that the weighted variance term $\mathbb{E}\left[\sum_{t=0}^{T-1} \gamma_t \|\nabla f(x_t) - \nabla_t\|^2\right]$ cannot be handled by Lemma 3. To overcome the latter, we use the monotonic behavior of the step-size γ_t to establish the following refinement of Lemma 3.

Lemma 7. *Let $x_0, x_1, \ldots$ the sequence of points produced by Algorithm 1. Then,*

$$\mathbb{E}\left[\sum_{t=0}^{T-1} \gamma_t \cdot \|\nabla_t - \nabla f(x_t)\|^2\right] \leq L^2 n \cdot \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma_t^3 \cdot \|\nabla_t\|^2\right]$$

To this end we are ready to summarize the importance of simultaneous additive/multiplicative dependence of γ_t in Step 9 of Algorithm 1. This selection permits us to do achieve two *orthogonal goals* at the same time,

- Bounding the variance of the process, $\mathbb{E}\left[\sum_{t=0}^{T-1} \|\nabla_t - \nabla f(x_t)\|\right] \leq \tilde{O}\left(n^{1/4}\sqrt{T}\right)$ (see Section 3 and Lemma 5).
- Bounding the sum, $\mathbb{E}\left[\sum_{t=0}^{T-1} \|\nabla_t\|\right] \leq \tilde{O}\left(n^{5/4}\sqrt{T} \cdot \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma_t^3 \|\nabla_t\|^2\right]\right)$ (see Lemma 6 and Lemma 7).

The third important thing that the selection of γ_t does is that it permits us to upper bound the term $\tilde{O}\left(n^{5/4}\, \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma_t^3 \|\nabla_t\|^2\right]\right)$ by $\tilde{O}\left(n^{1/4}\right)$. The proof of the latter upper bound can be found in the proof of Theorem 1 that due to lack of space is deferred to appendix. It is interesting that all the above three different purposes can be handled by selecting $\gamma_t = n^{-1/4}(\sqrt{n} + \sum_{s=0}^{t} \|\nabla_s\|^2)^{-1/2}$.

5 Experiments

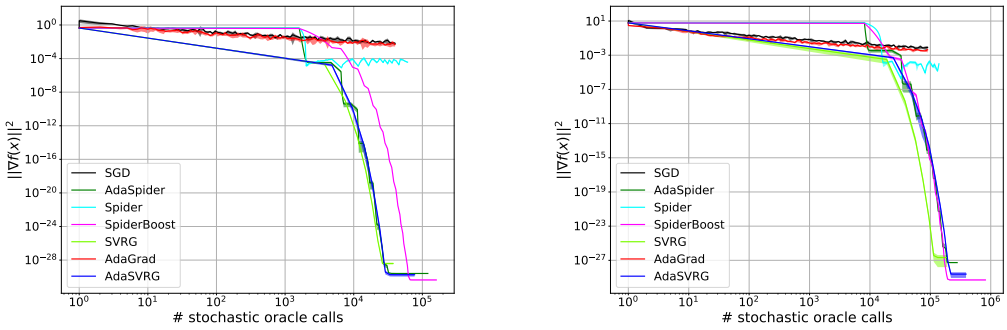
We complement our theoretical findings with an evaluation of the numerical performance of the algorithm under different experimental setups. We aim to highlight the sample complexity improvements over simple stochastic methods, while displaying the advantages of adaptive step-size strategies. For that purpose we design two setups; first, we consider the minimization of a convex loss with a non-convex regularizer in the sense of Wang et al. [44] and in a second part we consider an image classification task with neural networks.

5.1 Convex loss with a non-convex regularizer

We consider the following problem:

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(x, (a_i, b_i)) + \lambda g(x)$$

where $\ell(x, (a_i, b_i))$ is the loss with respect to the decision variable/weights x with (a_i, b_i) denoting the (feature vector, label) pair. We select $g = \sum_{i=1}^{d} \frac{x_i^2}{1+x_i^2}$, similar to Wang et al. [44], where the subscript denotes the corresponding dimension of x. We compare ADASPIDER against the original SPIDER, SPIDERBOOST, SVRG, ADASVRG and two non-VR methods, SGD and ADAGRAD. We made use of two datasets from LibSVM, namely `a1a, mushrooms`. We initialize each algorithm from the same point and repeat the experiments 5 times, then report the mean convergence with standard deviation as the shaded region around the mean curves. We tune the algorithms by executing a parameter sweep for their initial step-size over an interval of values which are exponentially scaled as $\{10^{-3}, 10^{-2}, ..., 10^{2}, 10^{3}\}$. After tuning the algorithms on one dataset, we run them with the same parameters for the others.



(a) a1a (b) mushrooms

Figure 1: Logistic regression with non-convex regularizer on LibSVM datasets

First, we clearly observe the difference between SGD & ADAGRAD, and the rest of the pack, which demonstrates the superior sample complexity of VR methods in general. Among VR algorithms, there does not seem to be any concrete differences with similar convergence, except for SPIDER. The performance of ADASPIDER is on par with other VR methods, and superior to SPIDER. The unexpected behavior of SPIDER algorithm has previously been documented in Wang et al. [44]. From a technical point of view, this behavior is predominantly due to the *accuracy dependence* in the step-size, making the step-size unusually small. We had to run SPIDER beyond its prescribed setting and tune the step-size with a large initial value to make sure the algorithm makes observable progress.

5.2 Experiments with neural networks

In our second setup, we train neural networks with our variance reduction scheme. Our focus is on standard image classification tasks trained with the cross entropy loss [9, 10]. Denoting by C the number of classes, the considered datasets in this section consist of n pairs (a_i, b_i) where a_i is a vectorized image and $b_i \in \mathbb{R}^C$ is a one-hot encoded class label. A neural network is parameterized with weights $x \in \mathbb{R}^d$ and its output on is denoted net(x, a) $\in \mathbb{R}^C$, where a is the input image. The training of the network consists of solving the following optimization problem:

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(-b_i^\top \text{net}(x, a_i) + \texttt{logsumexp}(\text{net}(x, a_i)) \right).$$

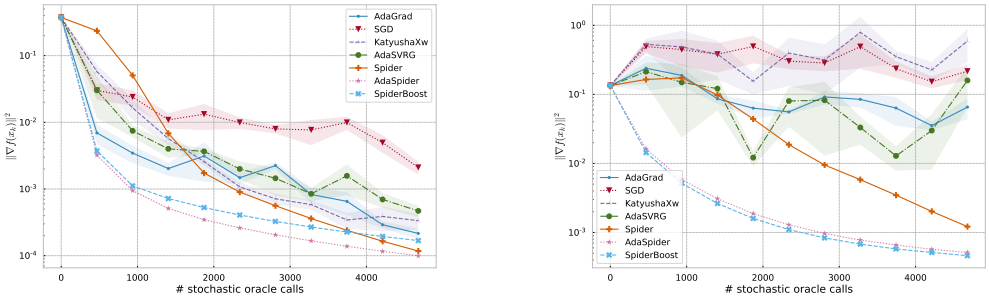
This is the default setup for doing image classification and we test our algorithm on two benchmark datasets : `MNIST`[28] and `FashionMNIST`[46].

Table 2: Algorithm parameters and test accuracies (average of 5 runs, in %)

	MNIST Batch Size $= 32$, $c_{\text{init}} = 0.03$		FashionMNIST Batch Size $= 128$, $c_{\text{init}} = 0.01$	
Algorithm	**Parameters**	**Test Accuracy**	**Parameters**	**Test Accuracy**
AdaGrad[16]	$\eta = 0.01,\ \epsilon = 10^{-4}$	97.86	$\eta = 0.01,\ \epsilon = 10^{-4}$	86.19
SGD[40]	$\eta = 0.01$	98.11	$\eta = 0.01$	85.83
KatyushaXw[3]	$\eta = 0.005$	97.93	$\eta = 0.01$	86.27
AdaSVRG[15]	$\eta = 0.1$	98.03	$\eta = 0.1$	86.82
Spider[19]	$\epsilon = 0.01,\ L = 100.0,\ n_0 = 1$	97.53	$\epsilon = 0.01,\ L = 50.0,\ n_0 = 1$	82.22
SpiderBoost[44, 37]	$L = 200$	97.01	$L = 120$	84.42
AdaSpider	$n = 60000$	97.49	$n = 60000$	84.09

Architecture: The network is a 3-layer fully connected network with dimensions $[28 * 28, 512, 512, 10]$. The activation function is the ELU [11].

Initialization: The initialization of the network is a crucial component to guarantee good performance. We find that a slight modification of the Kaiming Uniform initialization [21] improves the stability of the tested variance reduction schemes. For each layer in the network with d_{in} inputs, the original method initializes the weights with independent uniform random variables with variance $\frac{1}{d_{in}}$. Our modification initializes with a smaller variance of $\frac{c_{\text{init}}}{d_{in}}$ with c_{init} in the order of 0.01. With this choice, we observed that fewer variance reductions schemes diverged, and standard algorithms like SGD and AdaGrad(for which the original method was tuned), were not penalized and performed well. This often overlooked initialization heuristic is the only "tuning" needed for AdaSpider.



(a) MNIST (b) FashionMNIST

Figure 2: Gradient norms throughout the epochs for image classification with neural networks (curves are averaged over 5 independent runs and the shaded region are the standard error).

Observations: We observe (Figure 2) that AdaSpider performs as well as other variance reduction methods in terms of minimizing the gradient norm. The key message here is that it does so without the need for extensive tuning. This diminished need for tuning is a welcome feature for deep learning optimization, but, often the true metric of interest is not the gradient norm, but the accuracy on unseen data, and on this metric variance reduction schemes are not yet competitive with simpler methods like SGD. With AdaSpider, the focus can go to finding the right initialization scheme and architecture to ensure good generalization without being distracted by other parameters like the step-size choice.

References

[1] Zeyuan Allen-Zhu. Natasha: Faster non-convex stochastic optimization via strongly non-convex parameter, 2017. URL https://arxiv.org/abs/1702.00763.

[2] Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, page 1200–1205. Association for Computing Machinery, 2017.

[3] Zeyuan Allen-Zhu. Katyusha X: Practical Momentum Method for Stochastic Sum-of-Nonconvex Optimization. In *Proceedings of the 35th International Conference on Machine Learning*, ICML '18, 2018. Full version available at http://arxiv.org/abs/1802.03866.

[4] Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 699–707. JMLR.org, 2016.

[5] Kimon Antonakopoulos, Elena Veronica Belmega, and Panayotis Mertikopoulos. Adaptive extra-gradient methods for min-max optimization and games. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[6] Kimon Antonakopoulos, Thomas Pethick, Ali Kavis, Panayotis Mertikopoulos, and Volkan Cevher. Sifting through the noise: Universal first-order methods for stochastic variational inequalities. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 13099–13111, 2021.

[7] Yossi Arjevani, Yair Carmon, John C. Duchi, Dylan J. Foster, Nathan Srebro, and Blake E. Woodworth. Lower bounds for non-convex stochastic optimization. *CoRR*, abs/1912.02365, 2019.

[8] Jonathan Barzilai and Jonathan M. Borwein. Two-Point Step Size Gradient Methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 01 1988. ISSN 0272-4979. doi: 10.1093/imanum/8.1.141. URL https://doi.org/10.1093/imanum/8.1.141.

[9] John Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. *Advances in neural information processing systems*, 2, 1989.

[10] John S Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer, 1990.

[11] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.07289.

[12] Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex sgd. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/b8002139cdde66b87638f7f91d169d96-Paper.pdf.

[13] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[15] Benjamin Dubois-Taine, Sharan Vaswani, Reza Babanezhad, Mark Schmidt, and Simon Lacoste-Julien. SVRG meets adagrad: Painless variance reduction. *CoRR*, abs/2102.09645, 2021. URL https://arxiv.org/abs/2102.09645.

[16] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[17] Alina Ene, Huy L. Nguyen, and Adrian Vladu. Adaptive gradient methods for constrained convex optimization and variational inequalities. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 7314–7321. AAAI Press, 2021.

[18] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. SPIDER: near-optimal non-convex optimization via stochastic path-integrated differential estimator. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 687–697, 2018.

[19] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/1543843a4723ed2ab08e18053ae6dc5b-Paper.pdf.

[20] Fabian Latorre Gómez, Paul Rolland, and Volkan Cevher. Lipschitz constant estimation of neural networks via sparse polynomial optimization. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[22] Yu-Guan Hsieh, Kimon Antonakopoulos, and Panayotis Mertikopoulos. Adaptive learning in continuous games: Optimal regret bounds and convergence to nash equilibrium. In Mikhail Belkin and Samory Kpotufe, editors, *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134 of *Proceedings of Machine Learning Research*, pages 2388–2422. PMLR, 2021.

[23] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, page 315–323. Curran Associates Inc., 2013.

[24] Pooria Joulani, Anant Raj, Andras Gyorgy, and Csaba Szepesvari. A simpler approach to accelerated optimization: iterative averaging meets optimism. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4984–4993. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/joulani20a.html.

[25] Ali Kavis, Kfir Y. Levy, Francis Bach, and Volkan Cevher. Unixgrad: A universal, adaptive algorithm with optimal guarantees for constrained optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 6260–6269. Curran Associates, Inc., 2019.

[26] Guanghui Lan, Zhize Li, and Yi Zhou. A unified variance-reduced accelerated gradient method for convex optimization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 10462–10472, 2019.

[27] Guanghui Lan, Zhize Li, and Yi Zhou. A unified variance-reduced accelerated gradient method for convex optimization, 2019.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[29] Kfir Y Levy, Alp Yurtsever, and Volkan Cevher. Online adaptive methods, universality and acceleration. In *Neural and Information Processing Systems (NeurIPS)*, December 2018.

[30] Kfir Yehuda Levy, Ali Kavis, and Volkan Cevher. STORM+: Fully adaptive SGD with recursive momentum for nonconvex optimization. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=ytke6qKpxtr.

[31] Bingcong Li, Lingda Wang, and Georgios B. Giannakis. Almost tune-free variance reduction. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5969–5978. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/li20n.html.

[32] Zhize Li and Jian Li. A simple proximal stochastic gradient method for nonsmooth nonconvex optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 5569–5579, Red Hook, NY, USA, 2018. Curran Associates Inc.

[33] Zhize Li, Hongyan Bao, Xiangliang Zhang, and Peter Richtarik. Page: A simple and optimal probabilistic gradient estimator for nonconvex optimization. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6286–6295. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/li21a.html.

[34] Zhize Li, Slavomír Hanzely, and Peter Richtárik. Zerosarah: Efficient nonconvex finite-sum optimization with zero full gradient computation, 2021. URL https://arxiv.org/abs/2103.01447.

[35] Zijian Liu, Ta Duy Nguyen, Alina Ene, and Huy L. Nguyen. Adaptive accelerated (extra-)gradient methods with variance reduction. *CoRR*, abs/2201.12302, 2022. URL https://arxiv.org/abs/2201.12302.

[36] Yurii Nesterov. Introductory lectures on convex optimization. 2004, 2003.

[37] Lam M. Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2613–2621. PMLR, 06–11 Aug 2017.

[38] Nhan H. Pham, Lam M. Nguyen, Dzung T. Phan, and Quoc Tran-Dinh. Proxsarah: An efficient algorithmic framework for stochastic composite nonconvex optimization, 2019. URL https://arxiv.org/abs/1902.05679.

[39] Sashank J. Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 314–323. PMLR, 2016.

[40] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951. doi: 10.1214/aoms/1177729586. URL https://doi.org/10.1214/aoms/1177729586.

[41] Chaobing Song, Yong Jiang, and Yi Ma. Variance reduction via accelerated dual averaging for finite-sum optimization. In *Advances in Neural Information Processing Systems*, volume 33, pages 833–844. Curran Associates, Inc., 2020.

[42] Conghui Tan, Shiqian Ma, Yu-Hong Dai, and Yuqiu Qian. Barzilai-borwein step size for stochastic gradient descent. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 685–693, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

[43] Dong Quan Vu, Kimon Antonakopoulos, and Panayotis Mertikopoulos. Fast routing under uncertainty: Adaptive learning in congestion games via exponential weights. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 14708–14720, 2021.

[44] Zhe Wang, Kaiyi Ji, Yi Zhou, Yingbin Liang, and Vahid Tarokh. Spiderboost and momentum: Faster variance reduction algorithms. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 2403–2413, 2019.

[45] Blake E. Woodworth and Nati Srebro. Tight complexity bounds for optimizing composite objectives. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3639–3647, 2016.

[46] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[47] Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic nested variance reduction for nonconvex optimization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/136f951362dab62e64eb8e841183c2a9-Paper.pdf.

[48] Zeyuan Allen Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 699–707. JMLR.org, 2016.

[49] Zeyuan Allen Zhu and Yang Yuan. Improved SVRG for non-strongly-convex or sum-of-non-convex objectives. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1080–1089. JMLR.org, 2016.

[50] Zeyuan Allen Zhu and Yang Yuan. Improved SVRG for non-strongly-convex or sum-of-non-convex objectives. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1080–1089. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/allen-zhub16.html.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See, e.g, discussion below Remark 1, contributions section of the Introduction
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 2, Theorem 1
   (b) Did you include complete proofs of all theoretical results? [Yes] See the Appendix

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We provide mean and standard deviation on the curves over multiple runs
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]