Mathematical Programming
https://doi.org/10.1007/s10107-024-02078-z

FULL LENGTH PAPER

Series A

Sample complexity analysis for adaptive optimization algorithms with stochastic oracles

Billy Jin[1] · **Katya Scheinberg**[1] · **Miaolan Xie**[1]

Received: 20 March 2023 / Accepted: 26 February 2024

Abstract

Several classical adaptive optimization algorithms, such as line search and trust-region methods, have been recently extended to stochastic settings where function values, gradients, and Hessians in some cases, are estimated via stochastic oracles. Unlike the majority of stochastic methods, these methods do not use a pre-specified sequence of step size parameters, but adapt the step size parameter according to the estimated progress of the algorithm and use it to dictate the accuracy required from the stochastic oracles. The requirements on the stochastic oracles are, thus, also adaptive and the oracle costs can vary from iteration to iteration. The step size parameters in these methods can increase and decrease based on the perceived progress, but unlike the deterministic case they are not bounded away from zero due to possible oracle failures, and bounds on the step size parameter have not been previously derived. This creates obstacles in the total complexity analysis of such methods, because the oracle costs are typically decreasing in the step size parameter, and could be arbitrarily large as the step size parameter goes to 0. Thus, until now only the total iteration complexity of these methods has been analyzed. In this paper, we derive a lower bound on the step size parameter that holds with high probability for a large class of adaptive stochastic methods. We then use this lower bound to derive a framework for analyzing the expected and high probability total oracle complexity of any method in this class. Finally, we apply this framework to analyze the total sample complexity of two particular algorithms, STORM (Blanchet et al. in INFORMS J Optim 1(2):92–119, 2019) and SASS (Jin et al. in High probability complexity bounds for adaptive step search based on stochastic oracles, 2021. https://doi.org/10.48550/ARXIV.2106.06454), in the expected risk minimization problem.

✉ Miaolan Xie
mx229@cornell.edu

Billy Jin
bzj3@cornell.edu

Katya Scheinberg
ks2375@cornell.edu

[1] School of Operations Research and Information Engineering, Cornell University, Ithaca, NY, USA

Published online: 29 April 2024

Keywords Nonlinear optimization · Stochastic optimization · Adaptive algorithm · High probability · Sample complexity · Stochastic oracles

Mathematics Subject Classification 90C15 · 90C26 · 90C30

1 Introduction

The widespread use of stochastic optimization algorithms for problems arising in machine learning and signal processing has made the stochastic gradient method and its variants overwhelmingly popular despite their theoretical and practical shortcomings. *Adaptive stochastic optimization* algorithms, on the other hand, borrow from decades of advances in deterministic optimization research, and offer new paths forward for stochastic optimization to be more effective and even more applicable. Adaptive algorithms can avoid many of the practical deficiencies of contemporary methods (such as the tremendous costs of tuning the step sizes of an algorithm for each individual application) while possessing strong convergence and worst-case complexity guarantees in surprisingly diverse settings.

Adaptive optimization algorithms have a long and successful history in deterministic optimization and include line search, trust-region methods, cubic regularized Newton methods, etc. All these methods have a common iterative framework, where at each iteration a candidate step is computed by the algorithm based on a local model of the objective function and a step size parameter that controls the length of this candidate step. The candidate step is then evaluated in terms of the decrease it achieves in the objective function, with respect to the decrease that it was expected to achieve based on the model. Whenever the decrease is sufficient, the step is accepted and the step size parameter may be increased to allow the next iteration to be more aggressive. If the decrease is not sufficient (or not achieved at all) then the step is rejected and a new step is computed using a smaller step size parameter. The model itself remains unchanged if it is known that a sufficiently small step will always succeed, as is true, for example with first- and second-order Taylor models of smooth functions. The analysis of such methods relies on the key property that the step size parameter is bounded away from zero and that once it is small enough the step is always accepted and the objective function gets reduced.

When stochastic oracles are used to estimate the objective function and its derivatives, the models no longer have the same property as the Taylor models; steps that decrease the model might not decrease the function, no matter how small the step size is. Thus all stochastic variants of these methods recompute the model at each iteration. The requirement on the stochastic model is then that it achieves a Taylor-like approximation with sufficiently high probability. This also means that steps may get rejected even if the step size parameter is small, simply because the stochastic oracles fail to deliver desired accuracy. Despite this difficulty, one can develop and analyze stochastic variants of adaptive optimization methods.

Recently developed stochastic variants of line search (which we will call step search, since unlike the deterministic version the search direction has to change on each iteration, thus the algorithm is not searching along a line) include [1–3, 22]. Stochastic

trust-region methods have been analyzed in [4–6], and adaptive cubic regularized methods based on random models has been analyzed in [1, 7]. For all these methods, bounds on iteration complexity have been derived, either in expectation or in high probability, under the assumption that stochastic oracles involved in estimating the objective function deliver sufficiently high accuracy with sufficiently high probability. While this probability is usually fixed, the accuracy requirement of the oracles is adaptive and depends on the step size parameter. Specifically, the smaller that parameter is, the more accurate the oracles need to be, in order to maintain the Taylor-like behavior of the model used by the algorithm. In most applications, having more accurate stochastic oracles implies a higher per-iteration cost. Furthermore, unlike the deterministic case, the step sizes in the stochastic case are not bounded away from zero due to possible oracle failures, and bounds on the step size parameter have not been previously derived. This creates significant difficulty in the analysis of the total oracle complexity of such methods because the oracle costs could be arbitrarily large as the step size parameter goes to zero.

In this paper, we derive a lower bound on the step size parameter for a general class of stochastic adaptive methods that encompasses all the algorithms in the preceding paragraph. This enables us to derive a bound on the total oracle complexity for any algorithm within this class and specific stochastic oracles arising, for example, from expected risk minimization. Our key contributions are as follows:

- Provide a high probability lower bound on the step size parameter for a wide class of stochastic adaptive methods using a coupling argument between the stochastic process generated by the algorithm and a one-sided random walk.
- Derive a framework for analyzing expected and high probability total oracle complexity bounds for this general class of stochastic adaptive methods.
- Apply these bounds to STORM (Stochastic Trust-region Optimization with Random Models) [4] and SASS (Stochastic Adaptive Step Search) [22] to derive their total sample complexity for expected risk minimization, and show they essentially match the complexity lower bound of first-order algorithms for stochastic non-convex optimization [8].

We note that there is a plethora of other algorithms in the literature that propose different ways to adaptively vary the step sizes in stochastic optimization. Some of these methods, such as [9, 10], have step size dynamics that are very different from the framework studied in this paper, hence the analysis of this paper is not needed or applicable. Some other adaptive methods, like [11–13], have step size dynamics that are more similar. However, since these papers do not provide iteration complexity bounds, we cannot obtain their sample complexity by directly applying our framework. Nonetheless, it would be interesting to explore whether our step size lower bound can be applied to the algorithms in these papers when a certain stopping time is specified.

We consider a continuous optimization problem of the form

$$\min_{x \in \mathbb{R}^m} \ \phi(x), \tag{1}$$

where ϕ is possibly non-convex, (twice-)continuously differentiable with Lipschitz continuous derivatives. Neither function values $\phi(x)$, nor gradients $\nabla\phi(x)$ are assumed

to be directly computable. Instead, given any $x \in \mathbb{R}^m$, it is assumed that stochastic estimates of $\phi(x)$, $\nabla\phi(x)$, and possibly $\nabla^2\phi(x)$ can be computed, and these estimates may possess different levels of *accuracy* and *reliability* depending on the particular setting of interest.

The adaptive stochastic algorithms that have been developed recently [1–6, 14, 15, 22] use these stochastic estimates to compute an ε-optimal point x_ε, which means $\phi(x_\varepsilon) - \inf_x \phi(x) \leq \varepsilon$ if ϕ is convex or $\|\nabla\phi(x_\varepsilon)\| \leq \varepsilon$ if ϕ is non-convex. In the next section, we will introduce the general framework that encompasses these methods and discuss particular examples in more detail. In Sect. 2.6, we discuss the stochastic process generated by the algorithmic framework, including the stochastic step size parameter. In Sect. 3, we derive a lower bound on the step size parameter which holds in high probability. In Sect. 4, we use this lower bound to derive abstract expected and high probability total oracle complexity for any algorithm in this framework. In Sect. 5, we particularize these bounds to the specific examples of first-order STORM [4] and SASS [22] algorithms to bound their total sample complexity when applied to expected risk minimization. We conclude with some remarks in Sect. 6.

2 Algorithm framework and oracles

We now introduce and discuss an algorithmic framework for adaptive stochastic optimization in Algorithm 1. The framework is assumed to have access to stochastic oracles, that for any given point x can generate random quantities $f(x, \xi_0) \approx \phi(x)$ (via zeroth-order oracle), $g(x, \xi_1) \approx \nabla\phi(x)$ (first-order oracle) and, possibly, $H(x, \xi_2) \approx \nabla^2\phi(x)$ (second-order oracle). After we introduce the algorithm we will discuss the general definition of an oracle that we use in this paper.

2.1 General algorithm

At each iteration, given x_k, a stochastic local model $m_k(x_k+s) : \mathbb{R}^m \to \mathbb{R}$ of $\phi(x_k+s)$ is constructed using $g(x, \xi_1)$ and possibly $H(x, \xi_2)$. Using this model, a step $s_k(\alpha_k)$ is computed so that $m(x_k + s_k(\alpha_k))$ is a sufficient improvement over $m_k(x_k)$, where α_k is the step size parameter, which directly or indirectly controls the step length. The iterate x_k and the trial point $x_k^+ = x_k + s_k(\alpha_k)$ are evaluated using the zeroth-order oracle values $f(x_k, \xi_{0,k})$ and $f(x_k^+, \xi_{0,k}^+)$. The algorithm then checks whether these estimates suggest that sufficient improvement is attained. This is done by estimating whether the function value decrease is large enough (usually involving a user-chosen parameter $\theta \in (0, 1)$) relative to the model decrease. If yes, then the step is deemed *successful*, x_k^+ is accepted as the next iterate, and the parameter α_k is increased up to a multiplicative factor; otherwise, the step is deemed *unsuccessful*, the iterate does not change, and α_k is decreased by a multiplicative factor. Unlike in the deterministic case, new calls to all oracles are made at each iteration *even when the iterate does not change*.

For each method we give the form of $m_k(x_k + s)$, in terms of $g_k = g(x_k, \xi_{1,k})$ and $H_k = H(x_k, \xi_{2,k})$, $s_k(\alpha_k)$ and the sufficient reduction criterion in terms of $f_k^0 =$

$f(x_k, \xi_{0,k})$ and $f_k^+ = f(x_k^+, \xi_{0,k}^+)$. Note that each time the oracle is called on some point x, it generates a fresh estimate, even if the oracle has been called on the same point x before—hence the dependence of ξ on k.

Algorithm 1 Algorithmic Framework for Adaptive Stochastic Optimization

0. Initialization

Choose $\theta \in (0, 1)$, $\gamma \in (0, 1)$, $\alpha_{\max} \in (0, \infty)$, $x_0 \in \mathbb{R}^m$, and $\alpha_0 \in (0, \alpha_{\max}]$. Set $k \leftarrow 0$.

1. Determine model and compute step

Construct a stochastic model m_k of ϕ at x_k using $f(x_k, \xi_{0,k})$, $g(x_k, \xi_{1,k})$, and (optionally) $H(x_k, \xi_{2,k})$ from *probabilistic zeroth-, first-, and (optionally) second-order oracles*. Compute $s_k(\alpha_k)$ such that the model reduction $m_k(x_k) - m_k(x_k + s_k(\alpha_k)) \geq 0$ is sufficiently large.

2. Check for sufficient reduction

Set $x_k^+ \leftarrow x_k + s_k(\alpha_k)$ and compute $f(x_k^+, \xi_{0,k}^+)$ as a stochastic estimate of $\phi(x_k^+)$ using a *probabilistic zeroth-order oracle*. Check if $f(x_k, \xi_{0,k}) - f(x_k^+, \xi_{0,k}^+)$ is sufficiently large (e.g., relative to the model reduction $m_k(x_k) - m_k(x_k^+)$) using a condition parameterized by θ.

3. Successful iteration

If sufficient reduction has been attained (along with other potential requirements), then set $x_{k+1} \leftarrow x_k^+$ and $\alpha_{k+1} \leftarrow \min\{\gamma^{-1}\alpha_k, \alpha_{\max}\}$.

4. Unsuccessful iteration

Otherwise, set $x_{k+1} \leftarrow x_k$ and $\alpha_{k+1} \leftarrow \gamma\alpha_k$.

5. Next iteration

Set $k \leftarrow k + 1$ and go to Step 1.

2.2 General oracles

Typically in the optimization literature, an oracle is a computational procedure that provides the algorithm with some (estimate of) required information about the objective function. The oracle is endowed with some properties that the algorithm then utilizes. For example, an oracle may be assumed to produce $\nabla\phi(x)$ exactly—an assumption that a relevant optimization algorithm (and its analysis) will make use of and without which the algorithm may fail. Alternatively, an oracle may produce an estimate of $\nabla\phi(x)$, in which case a relevant algorithm will operate under some specific knowledge about the properties of the estimate; e.g. that it is an unbiased estimate with variance that is bounded as a function of $\|\nabla\phi(x)\|$, or that it is a deterministic estimator with an error bounded by some known quantity. Algorithms and their analyses differ depending on the properties of the oracles they utilize. The algorithms analyzed in [1–7, 14, 15, 22] all fit into the framework of Algorithm 1 but under a variety of specific assumptions on the oracles that generate function, gradient and and Hessian estimates. These assumptions are more complex than is typical in the literature, yet, they are shown to be applicable in many settings. For some extensive discussion on these properties, their comparison and specific examples we refer the reader, for example, to Cao et al. [6]. Here we summarize the main ideas.

As presented above, the algorithmic framework described in Algorithm 1 has access to oracles that generate random quantities $f(x, \xi_0) \approx \phi(x)$ (zeroth-order oracle),

$g(x, \xi_1) \approx \nabla\phi(x)$ (first-order oracle) and, possibly, $H(x, \xi_2) \approx \nabla^2\phi(x)$ (second-order oracle). Each of these oracles can have an input, an output and some intrinsic properties. The input is typically the current iterate x_k and the step size parameter α_k. These quantities, together with the intrinsic properties of the oracle, define the probability space (and thus the distribution) of the random variable ξ. For example, in the expected risk minimization setting, where the oracles are computed by averaging a minibatch of samples, ξ_i (for $i \in \{0, 1, 2\}$) is the random minibatch used for the ith-order oracle. The probability space and the distribution of ξ_i may depend on x and α, since (as we will see in Sect. 5) the size of the minibatch may depend on these quantities. As another example, the first-order oracles can be computed using (randomized) finite differences [16]. In this case, the probability space and the distribution of ξ_1 may depend on the current iterate, the step size parameter and the randomness in function value estimates. In this example, the distribution of ξ_1 may depend on the distribution of ξ_0. More examples of stochastic oracles can include robust gradient estimation [17, 18], SPSA [19], etc.

In summary, the oracles can be implemented in a variety of ways, depending on the application, while the algorithmic framework can be agnostic to how exactly the oracles are implemented. The algorithm operates under certain assumptions on the accuracy and reliability of the oracles. The cost of implementing an oracle to satisfy these requirements depends on the application, but it needs to be considered in an overall complexity analysis, which is what we address in this paper. The general oracles in this paper can be described at a high level as follows.

Definition 1 (*Stochastic jth-order oracle*) Given an input $x \in \mathbb{R}^n$ and the step size parameter α, the oracle computes $\varphi_j(x, \xi_j)$, an estimate of the jth-order derivative $\nabla^j\phi(x)$. In all the algorithms we consider, $\|\varphi_j(x, \xi_j) - \nabla^j\phi(x)\|$ is assumed to be bounded by some quantity (which will be a function of α), with probability at least $1-\delta_j$. Here, ξ_j is a random variable defined on probability space $(\Omega_j, \mathcal{F}_j, P_j)$ whose distribution depends on the input x and α, and δ_j is intrinsic to the oracle. The cost of the oracle is also a function of x, α and δ_j.

In this paper, we are interested in providing a simple way to upper bound the total oracle cost of an algorithm. The exact oracle cost can be lower. For example, we will bound the oracle cost by adding together the costs of the zeroth-, first-, (and optionally second-) order oracles. However, if the oracles can be implemented based on a shared sample set, our theory can be applied with small and simple adjustments to obtain a better bound on the oracle cost (for example, by not double counting the shared samples).

In the next subsection, we describe how various methods fit into the general framework, and what specific requirements they have on the oracles. In this paper we will use the specific notation where $f(x, \xi_0) = \varphi_0(x, \xi_0)$, $g(x, \xi_1) = \varphi_1(x, \xi_1)$ and $H(x, \xi_2) = \varphi_2(x, \xi_2)$.

2.3 Step search method

In the case of the step search (SS) methods in [1, 3, 22] the particulars are as follows. Quantities f_k^0 f_k^+ and g_k are random outputs of the stochastic oracles and H_k is some positive definite matrix (e.g., the identity).

- $m_k(x_k + s) = \phi(x_k) + g_k^T s + \frac{1}{2\alpha_k} s^T H_k s$,
- $s_k(\alpha_k) = -\alpha_k H_k^{-1} g_k$
- Sufficient reduction: $f_k^0 - f_k^+ \geq -\theta g_k^T s_k(\alpha_k) - r$

Note that although the true function value $\phi(x_k)$ appears in the definition of m_k, we do not need to know or query for it to run the algorithm because the function value is just a constant that makes no difference when we minimize the model. Here $\theta \in (0, 1)$, and r is a small positive number that compensates for the noise in the function estimates. We will discuss the choice of r after we introduce conditions on the oracle outputs $f(x, \xi_0)$ and $g(x, \xi_1)$.

- **SS.0** (Step search, zeroth-order oracle). Given a point x, the oracle computes a (random) function estimate $f(x, \xi_0)$ (where $\xi_0 = \xi_0(x)$ is the randomness of the oracle, which may depend on the current point x) such that

$$\mathbb{P}_{\xi_0}\left(|\phi(x) - f(x, \xi_0)| < \epsilon_f + t\right) \geq 1 - \delta_0(t),$$

  for some $\epsilon_f \geq 0$ and any $t > 0$.
- **SS.1** (Step search, first-order oracle). Given a point x and the current *step size parameter* α, the oracle computes a (random) gradient estimate $g(x, \xi_1)$ (where $\xi_1 = \xi_1(x, \alpha)$ is the randomness of the oracle, which may depend on x and α) such that

$$\mathbb{P}_{\xi_1}\left(\|g(x, \xi_1) - \nabla\phi(x)\| \leq \max\{\epsilon_g, \min\{\tau, \kappa\alpha\}\|g(x, \xi_1)\|\}\right) \geq 1 - \delta_1$$

  for some nonnegative constants ϵ_g, κ, τ and δ_1.

In [1], $\epsilon_f = 0$ and $\delta_0(t) \equiv 0$, which means that the zeroth-order oracle is exact, and $r = 0$. In [3], $\epsilon_f > 0$ and $\delta_0(t) \equiv 0$, which means that the zeroth-order oracle has a bounded error with probability one, and $r = 2\epsilon_f$. In [22], $\epsilon_f > 0$ and $\delta_0(t) = e^{-\lambda t}$ for some $\lambda > 0$. This means there is no restriction on the error if it is less than ϵ_f, and the tail of the error decays exponentially beyond ϵ_f. In [22], $r > 2 \sup_x \mathbb{E}_{\xi_0}[\,|\phi(x) - f(x, \xi_0)|\,]$.

In [1], $\epsilon_g = 0$ and $\delta_1 < \frac{1}{2}$. In [3, 22], $\epsilon_g > 0$ and δ_1 is sufficiently small with a more complicated upper bound. In [1, 3], $\alpha_{\max}$ is finite, thus τ is $\kappa\alpha_{\max}$. In [22], $\alpha_{\max}$ is infinity, and τ is simply assumed to be some constant intrinsic to the oracle.

2.4 Trust-region method

Stochastic trust-region (TR) methods that fall into the framework of Algorithm 1 have been developed and analyzed in [4–6]. In the case of TR algorithms, f_k^0, f_k^+, g_k, and (possibly) H_k are random outputs of the stochastic oracles, and

- $m_k(x_k + s) = \phi(x_k) + g_k^T s + \frac{1}{2} s^T H_k s$,
- $s_k(\alpha_k) = \arg\min_{s:\, \|s\| \le \alpha_k} m_k(x_k + s)$
- Sufficient reduction: $\frac{f_k^0 - f_k^+ + r}{m_k(x_k) - m_k(x_k + s_k(\alpha_k))} \ge \theta$
- Additional requirement for a successful iteration: $\|g_k\| \ge \theta_2 \alpha_k$, for some $\theta_2 > 0$.

The requirements for the oracles are as follows. In the case of first-order analysis in [4–6], the following first-order oracle is assumed to be available.

- **TR1.1** (First-order trust-region, first-order oracle). Given a point x and the current *trust-region radius* α, the oracle computes a gradient estimate $g(x, \xi_1)$ (where $\xi_1 = \xi_1(x, \alpha)$ is the randomness of the oracle, which can depend on x and α) such that

$$\mathbb{P}_{\xi_1} \left(\|g(x, \xi_1) - \nabla \phi(x)\| \le \epsilon_g + \kappa_{eg} \alpha \right) \ge 1 - \delta_1.$$

  Here, κ_{eg} and δ_1 are nonnegative constants.

In the second-order analysis, the following first- and second-order oracles are used:

- **TR2.1** (Second-order trust-region, first-order oracle). Given a point x and the current *trust-region radius* α, the oracle computes a gradient estimate $g(x, \xi_1)$ (where $\xi_1 = \xi_1(x, \alpha)$ is the randomness of the oracle, which can depend on x and α) such that

$$\mathbb{P}_{\xi_1} \left(\|g(x, \xi_1) - \nabla \phi(x)\| \le \epsilon_g + \kappa_{eg} \alpha^2 \right) \ge 1 - \delta_1.$$

  Here, κ_{eg} and δ_1 are nonnegative constants.
- **TR2.2** (Second-order trust-region, second-order oracle). Given a point x and the current *trust-region radius* α, the oracle computes a Hessian estimate $H(x, \xi_2)$ (where $\xi_2 = \xi_2(x, \alpha)$ is the randomness of the oracle, which can depend on x and α) such that

$$\mathbb{P}_{\xi_2} \left(\|H(x, \xi_2) - \nabla \phi(x)\| \le \epsilon_h + \kappa_{eh} \alpha \right) \ge 1 - \delta_2.$$

  Here, κ_{eh} and δ_2 are nonnegative constants.

ϵ_h and ϵ_g are assumed to equal 0 in [4, 5] but are allowed to be positive in [6].

In terms of the zeroth-order oracles, the three works make different assumptions. Specifically, in [5], as in [1], the zeroth-order oracle is assumed to be exact. In [6] the zeroth-order oracle is the same as in [22] (i.e. **SS.0**), and $r > 2\epsilon_f + \frac{2}{\lambda} \log 4$. For the first-order analysis in [4], however, the zeroth-order oracle is as follows (and $r = 0$).

- **TR1.0** (First-order trust-region, zeroth-order oracle). Given a point x and the current *trust-region radius* α, the oracle computes a function estimate $f(x, \xi_0)$ (where $\xi_0 = \xi_0(x, \alpha)$ is the randomness of the oracle, which can depend on x and α) such that

$$\mathbb{P}_{\xi_0} \left(|f(x, \xi_0) - \phi(x)| \le \kappa_{ef} \alpha^2 \right) \ge 1 - \delta_0,$$

  where κ_{ef} and δ_0 are some nonnegative constants.

For the second-order analysis in [4], the zeroth-order oracle requirements are tighter.

- **TR2.0** (Second-order trust-region, zeroth-order oracle). Given a point x and the current *trust-region radius* α, the oracle computes a function estimate $f(x, \xi_0)$ (where $\xi_0 = \xi_0(x, \alpha)$ is the randomness of the oracle, which can depend on x and α) such that

$$\mathbb{P}_{\xi_0}\left(|f(x, \xi_0) - \phi(x)| \leq \kappa_{ef1}\alpha^3\right) \geq 1 - \delta_0$$

and

$$\mathbb{E}_{\xi_0}\left[|f(x, \xi_0) - \phi(x)|\right] \leq \kappa_{ef2}\alpha^3$$

where κ_{ef1}, κ_{ef2} and δ_0 are some nonnegative constants.

2.5 Cubicly regularized Newton method

The cubicly regularized (CR) Newton methods in [1, 7] also fit the framework of Algorithm 1 with

$$\begin{aligned}
&\bullet\ m_k(x_k + s) = \phi(x_k) + g_k^T s + \tfrac{1}{2}s^T H_k s + \tfrac{1}{3\alpha_k}\|s\|^3,\\
&\bullet\ s_k(\alpha_k) = \arg\min_s m_k(x_k + s),\\
&\bullet\ \text{Sufficient reduction: } \tfrac{f_k^0 - f_k^+ + r}{m_k(x_k) - m_k(x_k + s_k(\alpha_k))} \geq \theta.
\end{aligned} \tag{2}$$

In [1], the zeroth-order oracle is assumed to be exact, that is $f_k^0 = \phi(x_k)$ and $f_k^+ = \phi(x_k + s_k(\alpha_k))$, and $r = 0$. In [7] the zeroth-order oracle and the choice of r are the same as in [22]. In [1, 7], a version of the following first- and second-order oracles are used. For specific implementation details, please refer to [1, 7].

- **CR.1** (Cubicly regularized Newton, first-order oracle). Given a point x and the current parameter α, the oracle computes a gradient estimate $g(x, \xi_1)$ (where $\xi_1 = \xi_1(x, \alpha)$ is the randomness of the oracle, which can depend on x and α) such that

$$\mathbb{P}_{\xi_1}\left(\|\nabla\phi(x) - g(x, \xi_1))\| \leq \kappa_{eg}\max\left\{\alpha, \|s\|^2\right\}\right) \geq 1 - \delta_1,$$

  where κ_{eg} and δ_1 are nonnegative constants, and s is defined in (2).
- **CR.2** (Cubicly regularized Newton, second-order oracle). Given a point x, and the current parameter α, the oracle computes a Hessian estimate $H(x, \xi_2)$ (where $\xi_2 = \xi_2(x, \alpha)$ is the randomness of the oracle, which can depend on x and α) such that

$$\mathbb{P}_{\xi_2}\left(\|(\nabla^2\phi(x) - H(x, \xi_2))s\| \leq \kappa_{eh}\max\left\{\alpha, \|s\|^2\right\}\right) \geq 1 - \delta_2,$$

  where κ_{eh} and δ_2 are nononegative constants, and s is defined in (2).

It is apparent that all of the algorithms that we discussed above rely on oracles whose accuracy requirements change adaptively with α. It is also clear that for many

settings, a higher accuracy requirement leads to a higher oracle complexity. For example, if a stochastic oracle is delivered via sample averaging, then more samples are needed to provide a higher accuracy. Therefore, to bound the total oracle complexity of the algorithm, we need to bound the accuracy requirement over the iterations, or equivalently provide a lower bound for the parameter α.

2.6 Notions of the stochastic process

When applied to problem (1), Algorithm 1 generates a stochastic process (with respect to the randomness underlying the stochastic oracles). Specifically, let $(X_k)_{k\geq 0}$ be the random iterates with realizations x_k, let $(G_k)_{k\geq 0}$ be the gradient estimates with realizations g_k, and let $(\mathcal{A}_k)_{k\geq 0}$ be the step size parameter values with realizations α_k. The prior works that analyze different methods belonging to the framework of Algorithm 1 define this stochastic process rigorously, with appropriate filtrations. Here for brevity, we will omit those details, as we do not use them in the analysis. We now define a stopping time for the process.

Definition 2 (*Stopping time*) For $\varepsilon > 0$, let T_ε be the first time such that a specified optimality condition is satisfied. For all the settings considered in this paper, $T_\varepsilon = \min\{k : \|\nabla\phi(x_k)\| \leq \varepsilon\}$ if ϕ is non-convex, and $T_\varepsilon = \min\{k : \phi(x_k) - \inf_x \phi(x) \leq \varepsilon\}$ if ϕ is strongly convex. We will refer to T_ε as the *stopping time* of the algorithm.

The following property is crucial in the analysis of algorithms in the framework of Algorithm 1.

Assumption 1 (*Properties of the stochastic process generated by the adaptive stochastic algorithm*) The random sequence of parameters $\mathcal{A}_k$ generated by the algorithm satisfies the following:

(i) For all k, $\mathcal{A}_k \in \{\gamma\mathcal{A}_{k-1}, \min\{\alpha_{\max}, \gamma^{-1}\mathcal{A}_{k-1}\}\}$, and
(ii) There exist constants $\bar{\alpha} > 0$, and $p > \frac{1}{2}$, such that for all iterations k,

$$\mathbb{P}(\mathcal{A}_{k+1} = \gamma^{-1}\mathcal{A}_k \mid \mathcal{F}_k,\ k < T_\varepsilon,\ \mathcal{A}_k \leq \bar{\alpha}) \geq p.$$

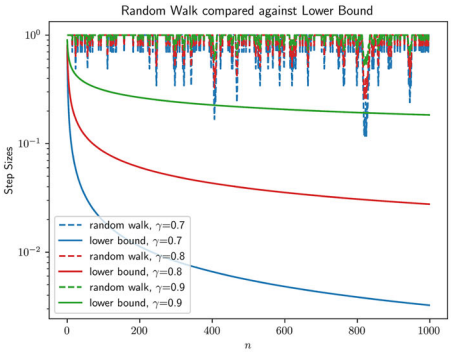
Here, $\mathcal{F}_k$ denotes the filtration generated by the algorithm up to iteration k.

The algorithms in [1–6, 22] all satisfy Assumption 1, under appropriate lower bounds on the oracle probabilities δ_0, δ_1 (and δ_2). Also, without loss of generality, we will assume that $\alpha_0 \geq \bar{\alpha}$, since if $\alpha_0 < \bar{\alpha}$ we can simply define $\bar{\alpha}$ to be α_0, as α_0 is a constant. In the next section, under Assumption 1, we derive a high probability lower bound on α_k as a function of the number of iterations n, $\bar{\alpha}$, p, and γ.

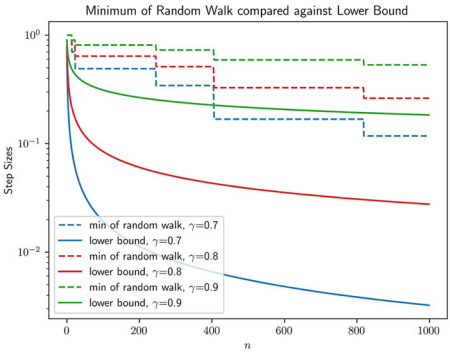
Throughout the remainder of this paper, we will use q to denote $1 - p$.

3 High probability lower bound for the step size parameter

The following theorem provides a high probability lower bound for α_k.



(a) Comparing the random walk trajectory with the theoretical lower bound for various values of γ.



(b) Comparing the *minimum value attained so far by the random walk curves in Figure 1a* with the theoretical lower bounds

Fig. 1 Illustration of Theorem 1

Theorem 1 *Suppose Assumption 1 holds for Algorithm 1. For any positive integer n, any $\omega > 0$, with probability at least $1 - n^{-\omega} - cn^{-(1+\omega)}$, we have*

$$\text{either } T_\varepsilon < n \text{ or } \min_{1 \le k \le n} \alpha_k \ge \alpha^*(n) := \bar{\alpha}\gamma\gamma^{(1+\omega)\log_{1/2q} n} = \bar{\alpha}\gamma n^{-(1+\omega)\log_{1/2q} 1/\gamma},$$

where $c = \frac{2\sqrt{pq}}{(1-2\sqrt{pq})^2}$ and $q = 1 - p$.

The proof of this theorem involves two steps. First, in Sect. 3.1, we show that for $n < T_\varepsilon$, the sequence of step size parameters $\mathcal{A}_k$ generated by the algorithm can be coupled with a random walk on the non-negative integers. This reduces the problem to that of bounding the maximum value of a one-sided random walk in the first n steps. We then derive a high probability upper bound on this maximum value in Sect. 3.2.

Before moving to its proof, we illustrate the theorem using some plots and comment on some implications of the theorem.

Illustration of Theorem 1. Figure 1a illustrates the high probability bound provided by Theorem 1. The solid curves depict the lower bounds given by the theorem for $\bar{\alpha} = 1, \omega = 1, p = 0.8$, and for varying values of γ. In comparison, the dotted lines correspond to one-sided random walks $\mathcal{Z}_k$ that start at $\bar{\alpha} = 1$. At each step, $\mathcal{Z}_{k+1} = \gamma\mathcal{Z}_k$ with probability $1 - p$, and $\mathcal{Z}_{k+1} = \min\{1, \gamma^{-1}\mathcal{Z}_k\}$ with probability p. The proof of Theorem 1 shown later implies that there is a coupling between the sequence of parameters $\mathcal{A}_k$ generated by the algorithm and $\mathcal{Z}_k$, such that $\mathcal{A}_k \ge \mathcal{Z}_k$, in other words, the sequence of parameters $\mathcal{A}_k$ generated by the algorithm *stochastically dominates* $\mathcal{Z}_k$.

Remarks on Theorem 1.

1. For fixed n, γ, and $\bar{\alpha}$, the lower bound is a function of p. It increases as p increases. Specifically, the exponent of n changes with p, and the exponent goes to 0 as p goes to 1. Hence as p goes to 1, this lower bound simplifies to $\bar{\alpha}\gamma$, which matches the lower bound in the deterministic case.

2. When p is close to 1 (i.e. when the stochastic oracles are highly reliable), this lower bound decreases slowly as a function of n, since the exponent of n is close to 0. Alternatively, when the stochastic oracles are not highly reliable, increasing the value of γ allows the algorithm to maintain a slow decrease of the step size.
3. Enlarging γ as p decreases makes intuitive sense for the algorithm. When p is large, an unsuccessful step is more likely to be caused by the step size being too large rather than the failure of the oracles to deliver the desired accuracy. On the other hand, when p is small, unsuccessful iterations are likely to occur even when the step size parameter is already small. Thus in the latter case, larger γ values help avoid an erroneous rapid decrease of the step size parameter.
4. If we choose $\gamma = \left(\frac{1}{2q}\right)^{-\frac{1}{4}}$ and $\omega = 1$, then the minimum step size is lower bounded by $\bar{\alpha}\gamma n^{-\frac{1}{2}}$ with high probability. This coincides with the typical choice of the step size decay schemes for the stochastic gradient method applied to non-convex functions.

The theorem implies that we can even bound α by a *constant* times $\bar{\alpha}$ with high probability, provided we set γ as a function of n.

Corollary 1 *Let Assumption 1 hold for Algorithm 1, then for any positive integer n, any $\omega > 0$, and any $\beta < \frac{1}{2}$, if*

$$\gamma \geq \max\left\{\frac{1}{2}, \left(\frac{1}{2q}\right)^{\frac{\log(2\beta)}{(1+\omega)\log n}}\right\},$$

then with probability at least $1 - n^{-\omega} - cn^{-(1+\omega)}$, where $c = \frac{2\sqrt{pq}}{(1-2\sqrt{pq})^2}$, we have

$$either T_\varepsilon < n or \min_{1\leq k\leq n} \alpha_k \geq \beta\bar{\alpha}.$$

Proof This follows from Theorem 1 by substituting in the specified value of γ. □

In the remainder of this section, we prove Theorem 1 in two steps.

3.1 Step 1: reduction to random walk

We will use a coupling argument to obtain the reduction to a random walk.

Let $\{\mathcal{A}_k\}_{k=0}^{\infty}$ denote the random sequence of parameter values (whose realization is $\{\alpha_k\}_{k=0}^{\infty}$), for Algorithm 1. Let us assume, WLOG, that $\mathcal{A}_0 = \gamma^j\bar{\alpha}$, for some integer $j \leq 0$. (Recall here that $0 < \gamma < 1$.) Then we observe that $\mathcal{A}_k = \gamma^{\bar{Y}_k}\bar{\alpha}$, where $\{\bar{Y}_k\}_{k=0}^{\infty}$ is a random sequence of integers, with $\bar{Y}_0 = j \leq 0$, which increases by one on every unsuccessful step, and decreases by one on every successful step. Moreover, by Assumption 1, whenever $k < T_\varepsilon$ and $\bar{Y}_k \geq 0$, the probability that it decreases by

one is at least p. Define Y_k as follows:

$$Y_k = \bar{Y}_k \quad \text{if } k \le T_\varepsilon, \qquad Y_k = \begin{cases} Y_{k-1} - 1 & \text{w.p. } p \\ Y_{k-1} + 1 & \text{w.p. } 1-p \end{cases} \quad \text{if } k > T_\varepsilon. \tag{3}$$

In other words, Y_k follows the algorithm until T_ε, and then behaves like a random walk with downward drift p after T_ε. We now couple $\{Y_k\}_{k=0}^{\infty}$ with a random walk $\{Z_k\}_{k=0}^{\infty}$ which stochastically dominates Y_k.

Consider the following one-sided random walk $\{Z_k\}_{k=0}^{\infty}$, defined on the non-negative integers.

$$Z_0 = 0, \qquad Z_{k+1} = \begin{cases} Z_k + 1, & \text{w.p. } 1-p, \\ Z_k - 1, & \text{w.p. } p, \text{ if } Z_k \ge 1, \\ 0, & \text{w.p. } p, \text{ if } Z_k = 0. \end{cases} \tag{4}$$

Lemma 1 *There exists a coupling between Z_k and Y_k, where Z_k stochastically dominates Y_k.*

Proof of Lemma 1 Initially, $Z_0 = 0$ and $Y_0 \le 0$. For each k, we show how to update Z_k to Z_{k+1} according to how Y_k changes to Y_{k+1}. We consider two cases depending on whether $k < T_\varepsilon$ or $k \ge T_\varepsilon$.

Case 1: $k < T_\varepsilon$. If $Y_k \le -1$, we update Z_{k+1} from Z_k according to Eq. 4, independently of how Y_k changes to Y_{k+1}. If $Y_k \ge 0$, then we first check if Y_k increased or decreased. Let p' be the probability that $Y_{k+1} = Y_k - 1$ on this sample path. Since $Y_k \ge 0$, we know by Assumption 1 that $p' \ge p$. Now, if $Y_{k+1} = Y_k + 1$, then we set $Z_{k+1} = Z_k + 1$. On the other hand, if $Y_{k+1} = Y_k - 1$, then we set $Z_{k+1} = Z_k + 1$ with probability $1 - \frac{p}{p'}$, and $Z_{k+1} = \max\{Z_k - 1, 0\}$ with probability $\frac{p}{p'}$. Note that these probabilities are well-defined because $p' \ge p$.

Case 2: $k \ge T_\varepsilon$. If $Y_{k+1} = Y_k + 1$, then set $Z_{k+1} = Z_k + 1$. Otherwise, if $Y_{k+1} = Y_k - 1$, then set $Z_{k+1} = \max\{Z_k - 1, 0\}$.

Observe that under this coupling, $Z_k \ge Y_k$ on every sample path. Moreover, $\{Z_k\}$ and $\{Y_k\}$ have the correct marginal distributions. For Y_k, this is easy to see, since it evolves according to its true distribution and we are constructing Z_k from it. For Z_k, on any step with $k \ge T_\varepsilon$, Z_{k+1} evolves from Z_k correctly according to Eq. 4 by construction. On a step with $k < T_\varepsilon$ there are two cases: (1) $Y_k \le -1$, and (2) $Y_k \ge 0$. In the first case, the update from Z_k to Z_{k+1} clearly follows Eq. 4. This is also true in the second case, since there the probability that Z_k increases is $(1 - p') + p'(1 - \frac{p}{p'}) = 1 - p$.

To summarize, we have exhibited a coupling between $\{Z_k\}$ and $\{Y_k\}$, under which $Z_k \ge Y_k$ on any sample path. □

3.2 Step 2: upper-bounding the maximum value of the random walk

We now derive a high probability upper bound on the maximum value reached by the random walk.

Definition 3 Let $N(l, n)$ be the random variable that denotes the number of times $Z_k = l$ in the first n steps of the random walk.

By definition of $N(l, n)$, we have $N(l, n) > 0$ if and only if state l is visited in the first n steps of the random walk. The next proposition upper bounds the probability that $N(l, n) > 0$.

Proposition 1 *Let $q = 1 - p$. We have*

$$\mathbb{P}(N(l,n) > 0) \leq (n - l + 1)\frac{1 - (q/p)}{1 - (q/p)^{l+1}}\left(\frac{q}{p}\right)^l + \frac{2\sqrt{pq}}{(1 - 2\sqrt{pq})^2}(2q)^l.$$

Proof First, observe that $\mathbb{P}(N(l, n) > 0)$ remains unchanged if we change the state space from $\{0, 1, 2, \ldots\}$ to $\{0, 1, 2, \ldots, l\}$ and modify the walk to hold in state l with probability q (instead of moving from l to $l + 1$ with that probability). This defines a Markov chain on $\{0, 1, 2, \ldots, l\}$, and let P be its transition matrix. Noting that $P^m_{0,l}$ is the probability that the Markov chain is in state l at time m, we see that

$$\mathbb{P}(N(l,n) > 0) \leq \sum_{m=l}^{n} P^m_{0,l}. \tag{5}$$

The matrix P is explicitly diagonalized in [20] (Section XVI.3). By (3.16) in that section,

$$P^m_{0,l} = \frac{1 - (q/p)}{1 - (q/p)^{l+1}}\left(\frac{q}{p}\right)^l - \frac{2q}{l+1}\left(\frac{q}{p}\right)^{\frac{l-1}{2}} \sum_{r=1}^{l} \frac{\left[\sin\frac{\pi r}{l+1}\right]\left[\sin\frac{\pi rl}{l+1}\right]\left[2\sqrt{pq}\cos\frac{\pi r}{l+1}\right]^m}{1 - 2\sqrt{pq}\cos\frac{\pi r}{l+1}}. \tag{6}$$

The absolute value of the sum appearing in (6) can of course be bounded above by

$$\sum_{r=1}^{l} \frac{(2\sqrt{pq})^m}{1 - 2\sqrt{pq}} = l\frac{(2\sqrt{pq})^m}{1 - 2\sqrt{pq}}$$

and this readily yields

$$P^m_{0,l} \leq \frac{1 - (q/p)}{1 - (q/p)^{l+1}}\left(\frac{q}{p}\right)^l + \frac{2p}{1 - 2\sqrt{pq}}\left(\frac{q}{p}\right)^{\frac{l+1}{2}}(2\sqrt{pq})^m. \tag{7}$$

Summing (7) over $m = l, \ldots, n$ and using (5), we obtain the bound on $\mathbb{P}(N(l, n) > 0)$ claimed in the proposition. □

Remark 1 The bound for Proposition 1 is essentially tight, as the decay of $\mathbb{P}(N(l, n) > 0)$ is not faster than geometric; q^l is a lower bound.

With the above proposition at hand, Theorem 1 is proved by choosing an appropriate level l, for which $\mathbb{P}(N(l, n) = 0)$ is high.

Proof of Theorem 1 Let $q = 1 - p$. By Proposition 1, we have:

$$\mathbb{P}(N(l,n) > 0) \leq (n-l+1)\frac{1-(q/p)}{1-(q/p)^{l+1}}\left(\frac{q}{p}\right)^{l} + \frac{2\sqrt{pq}}{(1-2\sqrt{pq})^2}(2q)^{l}.$$

In other words,

$$\begin{aligned}\mathbb{P}(N(l,n) = 0) &\geq 1 - (n-l+1)\frac{1-(q/p)}{1-(q/p)^{l+1}}\left(\frac{q}{p}\right)^{l} - \frac{2\sqrt{pq}}{(1-2\sqrt{pq})^2}(2q)^{l} \\ &\geq 1 - n\left(\frac{q}{p}\right)^{l} - \frac{2\sqrt{pq}}{(1-2\sqrt{pq})^2}(2q)^{l}.\end{aligned}$$

Let $a > 0$ be a parameter to be set later, and take $l = \lceil a\log(n)\rceil$. Then, the above inequality implies:

$$\begin{aligned}\mathbb{P}(N(l,n) = 0) &\geq 1 - n\left(\frac{q}{p}\right)^{a\log(n)} - \frac{2\sqrt{pq}}{(1-2\sqrt{pq})^2}(2q)^{a\log(n)} \\ &= 1 - n^{1-a\log(\frac{p}{q})} - \frac{2\sqrt{pq}}{(1-2\sqrt{pq})^2}n^{-a\log(1/(2q))}.\end{aligned}$$

In going from the first line to the second, we used the following fact about logarithms: $x^{a\log n} = n^{a\log x} = n^{-a\log(1/x)}$. Let $a = \frac{1+\omega}{\log(1/2q)}$. Then, $a\log(\frac{p}{q}) \geq a\log(\frac{1}{2q}) = 1+\omega$, so

$$\mathbb{P}(N(l,n) = 0) \geq 1 - n^{-\omega} - \frac{2\sqrt{pq}}{(1-2\sqrt{pq})^2}n^{-(1+\omega)}.$$

In other words, with probability at least $1 - n^{-\omega} - cn^{-(1+\omega)}$ with $c = \frac{2\sqrt{pq}}{(1-2\sqrt{pq})^2}$, the random walk will remain below $l = \lceil(1+\omega)\log_{1/2q} n\rceil$ in the first n steps. By construction of the coupling, with the above probability, we know the α parameter in the algorithm either remains above $\gamma^{\lceil a\log n\rceil}\bar{\alpha} = \gamma^{\lceil(1+\omega)\log_{1/2q} n\rceil}\bar{\alpha}$ throughout the first n steps, or the algorithm has reached its stopping time in n steps. □

It is natural to ask if the theory extends to the case where the factors for increasing and decreasing the step size are different. The main challenge in this case is that the stochastic process of the step sizes is now modeled by a random walk whose step length going up is different than the step length going down. If it turns out that the hitting probability $\mathbb{P}(N(l,n) > 0)$ can be bounded for the more general one-sided random walks, we believe the theory in our paper can be extended similarly. We will leave it as a subject for future research.

4 Expected and high probability total oracle complexity

We now use the tools derived in the previous section to obtain abstract expected and high probability upper bounds on the total oracle complexity of Algorithm 1. In Sect. 5, we will derive concrete bounds for the total oracle complexity of two specific algorithms (STORM and SASS), and the specific oracles arising in expected risk minimization.

The cost of an oracle call may depend on the step size parameter α and the probability parameter $1-\delta$, thus we denote the cost by $\mathsf{oc}(\alpha, 1-\delta)$. We will use $\mathsf{oc}(\alpha)$ in the paper to simplify the notation because for all algorithms in the class, δ can be treated as a constant. Moreover, the cost of an oracle call is a non-increasing function of α for all algorithms developed so far that fit into the framework.

Assumption 2 $\mathsf{oc}(\alpha)$ is non-increasing in α.

Definition 4 (*Total oracle complexity*) For a positive integer n, let $\mathsf{TOC}(n)$ be the random variable which denotes the total oracle complexity of running the algorithm for n iterations. In other words,

$$\mathsf{TOC}(n) = \sum_{k=1}^{n} \mathsf{oc}(\mathcal{A}_k).$$

4.1 Abstract expected total oracle complexity

We now proceed to bound $\mathsf{TOC}(\min\{T_\epsilon, n\})$ in expectation, where n is an arbitrary positive integer.

Theorem 2 *Let Assumptions 1 and 2 hold in Algorithm 1. For any positive integer n, we have*

$$\mathbb{E}[\mathsf{TOC}(\min\{T_\epsilon, n\})] \leq n \sum_{l=1}^{n} \min\left\{1, n\left(\frac{q}{p}\right)^l + \frac{2\sqrt{pq}}{(1-2\sqrt{pq})^2}(2q)^l\right\} \cdot \mathsf{oc}(\bar{\alpha}\gamma^l) + n\,\mathsf{oc}(\bar{\alpha}).$$

Proof First, observe that if the α_k parameters are all above some value α^* in the first n steps, then by Assumption 2, $\mathsf{TOC}(n) \leq n \cdot \mathsf{oc}(\alpha^*)$. Therefore, for any integer $l \geq 0$, we have

$$\mathbb{P}(\mathsf{TOC}(\min\{T_\epsilon, n\}) > n \cdot \mathsf{oc}(\bar{\alpha}\gamma^l)) \leq \mathbb{P}(N(l+1, n) > 0). \tag{8}$$

By Proposition 1,

$$\mathbb{P}(N(l+1, n) > 0) \leq n\left(\frac{q}{p}\right)^{l+1} + \frac{2\sqrt{pq}}{(1-2\sqrt{pq})^2}(2q)^{l+1}.$$

This implies

$$\mathbb{P}(\mathsf{TOC}(\min\{T_\epsilon, n\}) > n \cdot \mathsf{oc}(\bar{\alpha}\gamma^l)) \le \min\left\{1, n\left(\frac{q}{p}\right)^{l+1} + \frac{2\sqrt{pq}}{(1-2\sqrt{pq})^2}(2q)^{l+1}\right\}.$$

By the definition of expectation,

$$\begin{aligned}
&\mathbb{E}[\mathsf{TOC}(\min\{T_\epsilon, n\})] \\
&= \sum_{i=0}^{n\cdot\mathsf{oc}(\bar{\alpha}\gamma^n)} i \cdot \mathbb{P}(\mathsf{TOC}(\min\{T_\epsilon, n\}) = i) \\
&\le \sum_{l=0}^{n-1} \mathbb{P}\left(\mathsf{TOC}(\min\{T_\epsilon, n\}) \in (n \cdot \mathsf{oc}(\bar{\alpha}\gamma^l), n \cdot \mathsf{oc}(\bar{\alpha}\gamma^{l+1})]\right) \cdot n\,\mathsf{oc}(\bar{\alpha}\gamma^{l+1}) \\
&\quad + \mathbb{P}\left(\mathsf{TOC}(\min\{T_\epsilon, n\}) \in [0, n\,\mathsf{oc}(\bar{\alpha})]\right) \cdot n\,\mathsf{oc}(\bar{\alpha}) \\
&\le \sum_{l=0}^{n-1} \mathbb{P}\left(\mathsf{TOC}(\min\{T_\epsilon, n\}) > n\,\mathsf{oc}(\bar{\alpha}\gamma^l)\right) \cdot n\,\mathsf{oc}(\bar{\alpha}\gamma^{l+1}) + n\,\mathsf{oc}(\bar{\alpha}) \\
&\le \sum_{l=0}^{n-1} \min\left\{1, n\left(\frac{q}{p}\right)^{l+1} + \frac{2\sqrt{pq}}{(1-2\sqrt{pq})^2}(2q)^{l+1}\right\} \cdot n\,\mathsf{oc}(\bar{\alpha}\gamma^{l+1}) + n\,\mathsf{oc}(\bar{\alpha}).
\end{aligned}$$

□

4.2 Abstract high probability total oracle complexity

We now proceed to bound $\mathsf{TOC}(T_\varepsilon)$ in high probability, using Theorem 1.

Theorem 3 *Let Assumptions 1 and 2 hold in Algorithm 1. For any $\omega > 0$ and positive integer n, with probability at least $1 - \mathbb{P}(T_\varepsilon > n) - n^{-\omega} - cn^{-(1+\omega)}$,*

$$\mathsf{TOC}(T_\varepsilon) \le n \cdot \mathsf{oc}(\alpha^*(n)),$$

where $\alpha^*(n) = \bar{\alpha}\gamma n^{-(1+\omega)\log_{1/2q} 1/\gamma}$, and c is as defined in Theorem 1.

If γ is chosen to be at least $\max\left\{\frac{1}{2}, \left(\frac{1}{2q}\right)^{\frac{\log(2\beta)}{(1+\omega)\log n}}\right\}$ for some $\beta < \frac{1}{2}$, then $\alpha^*(n) \ge \beta\bar{\alpha}$, thus with probability at least $1 - \mathbb{P}(T_\varepsilon > n) - n^{-\omega} - cn^{-(1+\omega)}$,

$$\mathsf{TOC}(T_\varepsilon) \le n \cdot \mathsf{oc}(\beta\bar{\alpha}).$$

Proof Let $\mathsf{TOC}_{\mathsf{rw}}(n)$ be the total oracle complexity of the first n iterations with the corresponding sequence of parameters α_k induced by the one-sided random walk (that is, the sequence defined by $\alpha_k = \bar{\alpha}\gamma^{Z_k}$, where Z_k is defined in Sect. 3.1). In other

words,

$$\mathsf{TOC}_{\mathsf{rw}}(n) = \sum_{k=1}^{n} \mathsf{oc}(\bar{\alpha}\gamma^{Z_k}).$$

With probability $1 - \mathbb{P}(T_\varepsilon > n)$, we have $T_\varepsilon \leq n$, which implies

$$\mathsf{TOC}(T_\varepsilon) \leq \mathsf{TOC}_{\mathsf{rw}}(T_\varepsilon) \leq \mathsf{TOC}_{\mathsf{rw}}(n).$$

Here, the first inequality is by Lemma 1 and Assumption 2, and the second inequality is by $T_\varepsilon \leq n$.

The same arguments used in the proof of Theorem 1 show that with probability at least $1 - n^{-\omega} - cn^{-(1+\omega)}$, we have $\min_{1\leq k\leq n} \bar{\alpha}\gamma^{Z_k} \geq \alpha^*(n)$. Thus, with at least this probability, $\mathsf{TOC}_{\mathsf{rw}}(n) \leq n \cdot \mathsf{oc}(\alpha^*(n))$.

Putting these together with a union bound, the result follows.

The second part of the theorem follows from substituting in the specific choice of γ. □

5 Applying to STORM and SASS

In this section, we demonstrate how the generic oracle complexity bounds in the previous section can be applied to concrete combinations of oracles and algorithms. We will consider the specific setting of expected risk minimization and two algorithms, first-order STORM and SASS, which are described earlier in the paper and fully analyzed in [4, 22], respectively. For each case, we will state the bounds on $\mathsf{oc}(\alpha)$ as a function of α, and use those bounds in conjunction with the known bounds on T_ε (that have been derived in previous papers), to obtain a bound on the total oracle complexity for each algorithm.

The results we obtain are the first ones that bound the total oracle complexity of STORM and SASS, and we show that both algorithms are essentially near optimal in terms of total gradient sample complexity. When deriving these results, for simplicity of presentation, we omit most of the constants involved in the specific bounds on T_ε and specific conditions on various algorithmic constants. For all such details, we refer the reader to [4, 22]. We will include short comments regarding these constants, but otherwise replace them with a $\mathcal{O}(\cdot)$ notation.

Problem Setting: Expected risk minimization (ERM) can be written as

$$\min_{x\in\mathbb{R}^m} \phi(x) = \mathbb{E}_{d\sim\mathcal{D}}[l(x, d)].$$

Here, x represents the vector of model parameters, d is a data sample following distribution $\mathcal{D}$, and $l(x, d)$ is the loss when the model parameterized by x is evaluated on data point d. This problem is central in supervised machine learning and other settings such as simulation optimization [21]. For this problem, it is common to assume the function ϕ is L-smooth and is bounded from below, and gradients of functions

$\nabla_x l(x, d)$ can be computed for any $d \sim \mathcal{D}$, so we will consider this setting in this section.

In this setting, the zeroth- and first-order oracles are usually computed as follows:

$$f(x, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{d \in \mathcal{S}} l(x, d), \quad g(x, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{d \in \mathcal{S}} \nabla_x l(x, d), \tag{9}$$

where $\mathcal{S}$ is the "minibatch"—that is a set of i.i.d samples from $\mathcal{D}$. Generally, $|\mathcal{S}|$ can be chosen to depend on x.

In what follows we will refer to the total number of times an algorithm computes $l(x, d)$ for a specific x and d as its *total function value sample complexity* and the number of times the algorithm computes $\nabla l(x, d)$ as its *total gradient sample complexity*. The total (oracle or sample) complexity of the algorithm is defined as the sum of these two quantities.

5.1 Total sample complexity of first-order STORM

We first consider the first-order stochastic trust-region method (STORM) as introduced and analyzed in [4]. The algorithm uses zeroth- and first-order oracles defined in TR1.0 and TR1.1 in Sect. 2. Trust-region algorithms are usually applied to nonconvex functions and the stopping time of STORM is defined as $T_\varepsilon = \min\{k : \|\nabla \phi(x_k)\| \leq \varepsilon\}$. In Section 3.3 of [4], it is shown that Assumption 1 is satisfied with $\bar{\alpha} = \frac{\varepsilon}{\zeta}$, where ζ is a moderate constant that depends on κ_{eg}, L and some constant chosen by the algorithm.

In [4], the oracle costs of STORM in the ERM setting are briefly discussed under the following **assumptions** on $l(x, d)$.

- Function value: It is assumed that there is some $\sigma_f \geq 0$ such that for all x, $\mathrm{Var}_{d \sim \mathcal{D}}[l(x, d)] \leq \sigma_f^2$.
- Gradient: It is assumed that $\mathbb{E}_{d \sim \mathcal{D}}[\nabla_x l(x, d)] = \nabla \phi(x)$, and that there is some $\sigma_g \geq 0$ such that for all x,

$$\mathbb{E}_{d \sim \mathcal{D}} \|\nabla_x l(x, d) - \nabla \phi(x)\|^2 \leq \sigma_g^2. \tag{10}$$

The cost of each oracle call is the number of samples in the associated minibatch $\mathcal{S}$. By applying Chebyshev's inequality it is easy to bound the oracle costs of TR1.0 and TR1.1.

- Cost of TR1.0 with parameter α: $\mathsf{oc}_0(\alpha) = \frac{\sigma_f^2}{\delta_0 \kappa_{ef}^2 \alpha^4}$,
- Cost of TR1.1 with parameter α: $\mathsf{oc}_1(\alpha) = \frac{\sigma_g^2}{\delta_1 \kappa_{eg}^2 \alpha^2}$.

Below we substitute the specific oracle costs into Theorem 2 to obtain the expected total sample complexity for the first-order STORM algorithm. Specifically, we will bound the total sample complexity of STORM $\mathbb{E}[\mathsf{TOC}(\min\{T_\varepsilon, n\})]$ by deriving bounds

on the expectation of the total function value sample complexity TOC_0 and the total gradient sample complexity TOC_1, where

$$\mathsf{TOC}_0(n) = \sum_{k=1}^{n} \mathsf{oc}_0(\mathcal{A}_k),\ \mathsf{TOC}_1(n) = \sum_{k=1}^{n} \mathsf{oc}_1(\mathcal{A}_k) \text{ and } \mathsf{TOC}(n) = \mathsf{TOC}_0(n) + \mathsf{TOC}_1(n).$$

Theorem 4 (Expected total sample complexity bound of first-order STORM) *Let $p = 1 - \delta_0 - \delta_1$ and $q = 1 - p$. For the first-order STORM algorithm, for any iteration number $n \in \mathbb{Z}^+$, and $\gamma > (2q)^{\frac{1}{4}}$, we have*

$$\mathbb{E}[\mathsf{TOC}(\min\{T_\varepsilon, n\})] \leq \mathcal{O}\left(n \log_{p/q}(n) \left(\frac{\sigma_f^2}{\varepsilon^4} n^{4\log_{q/p}\gamma} + \frac{\sigma_g^2}{\varepsilon^2} n^{2\log_{q/p}\gamma}\right)\right).$$

If $\gamma \geq \left(\frac{q}{p}\right)^{\frac{\log c}{\log n}}$ for some constant $c > 1$ (so that $n^{\log_{q/p}\gamma} \leq c$), the above simplifies to be

$$\mathbb{E}[\mathsf{TOC}(\min\{T_\varepsilon, n\})] \leq n \log(n) \cdot \mathcal{O}\left(\frac{\sigma_f^2}{\varepsilon^4} + \frac{\sigma_g^2}{\varepsilon^2}\right). \tag{11}$$

Proof By Theorem 2, the total expected cost of the zeroth-order oracle over n iterations is bounded above by:

$$\begin{aligned}
\mathbb{E}[\mathsf{TOC}_0(\min\{T_\varepsilon, n\})] &\leq n \sum_{l=1}^{n} \min\left\{1, n\left(\frac{q}{p}\right)^l + \frac{2\sqrt{pq}}{(1-2\sqrt{pq})^2}(2q)^l\right\} \\
&\quad \cdot \mathsf{oc}_0(\bar{\alpha}\gamma^l) + n\,\mathsf{oc}_0(\bar{\alpha}) \\
&\leq \underbrace{n \sum_{l=1}^{n} \min\left\{1, n\left(\frac{q}{p}\right)^l\right\} \cdot \mathsf{oc}_0(\bar{\alpha}\gamma^l)}_{=:A} \\
&\quad + \frac{2n\sqrt{pq}}{(1-2\sqrt{pq})^2} \underbrace{\sum_{l=1}^{n} (2q)^l \cdot \mathsf{oc}_0(\bar{\alpha}\gamma^l)}_{=:B} + n\,\mathsf{oc}_0(\bar{\alpha}).
\end{aligned}$$

For the zeroth-order oracle, $\mathsf{oc}_0(\alpha) = \frac{\sigma_f^2}{\delta_0 \kappa_{ef}^2 \alpha^4} = \mathcal{O}(\frac{\sigma_f^2}{\alpha^4})$. We use this to calculate upper bounds for A and B. First, we consider A. Note that $\min\{1, n(\frac{q}{p})^l\} = 1$, if and only if $l \leq \log_{p/q}(n)$. Therefore,

$$
\begin{aligned}
A &\leq \sum_{l=1}^{\log_{\frac{p}{q}}(n)} \mathsf{oc}_0(\bar{\alpha}\gamma^l) + n \sum_{l \geq \log_{\frac{p}{q}}(n)} \left(\frac{q}{p}\right)^l \mathsf{oc}_0(\bar{\alpha}\gamma^l) \\
&\leq \sum_{l=1}^{\log_{\frac{p}{q}}(n)} \mathcal{O}\left(\frac{\sigma_f^2}{\bar{\alpha}^4\gamma^{4l}}\right) + n \sum_{l \geq \log_{\frac{p}{q}}(n)} \left(\frac{q}{p}\right)^l \mathcal{O}\left(\frac{\sigma_f^2}{\bar{\alpha}^4\gamma^{4l}}\right) \\
&\leq \mathcal{O}\left(\frac{\sigma_f^2}{\bar{\alpha}^4}\right)\left(\sum_{l=1}^{\log_{\frac{p}{q}}(n)} \frac{1}{\gamma^{4l}} + n \sum_{l \geq \log_{\frac{p}{q}}(n)} \left(\frac{q}{p}\right)^l \frac{1}{\gamma^{4l}}\right) \\
&\leq \mathcal{O}\left(\frac{\sigma_f^2}{\varepsilon^4}\right)\left(\log_{\frac{p}{q}}(n) \cdot \left(\frac{1}{\gamma^4}\right)^{\log_{\frac{p}{q}}(n)} + n\left(\frac{q}{p\gamma^4}\right)^{\log_{\frac{p}{q}}(n)} \frac{1}{1-\frac{q}{p\gamma^4}}\right) \\
&= \mathcal{O}\left(\frac{\sigma_f^2}{\varepsilon^4} \log_{\frac{p}{q}}(n)\, n^{\log_{\frac{p}{q}}\left(\frac{1}{\gamma^4}\right)}\right) \\
&= \mathcal{O}\left(\frac{\sigma_f^2}{\varepsilon^4} \log_{\frac{p}{q}}(n)\, n^{4\log_{\frac{q}{p}}\gamma}\right).
\end{aligned}
$$

Next we bound B. We have

$$
B = \sum_{l=1}^{n} (2q)^l \cdot \mathsf{oc}_0(\bar{\alpha}\gamma^l) \leq \sum_{l=0}^{\infty} (2q)^l\, \mathcal{O}\left(\frac{\sigma_f^2}{\bar{\alpha}^4\gamma^{4l}}\right) \leq \mathcal{O}\left(\frac{\sigma_f^2}{\varepsilon^4}\right)\left(\frac{1}{1-2q\cdot\frac{1}{\gamma^4}}\right) = \mathcal{O}\left(\frac{\sigma_f^2}{\varepsilon^4}\right).
$$

Using these bounds on A and B in the expression for $\mathbb{E}[\mathsf{TOC}_0(\min\{T_\epsilon, n\})]$, we obtain the bound on the total function value sample complexity as $\mathcal{O}\left(\frac{\sigma_f^2}{\varepsilon^4}\, n \log_{\frac{p}{q}}(n)\, n^{4\log_{\frac{q}{p}}\gamma}\right)$.

A similar calculation using the cost of the first-order oracle yields the bound $\mathcal{O}\left(\frac{\sigma_g^2}{\varepsilon^2}\, n \log_{\frac{p}{q}}(n)\, n^{2\log_{\frac{q}{p}}\gamma}\right)$ for $\mathbb{E}[\mathsf{TOC}_1(\min\{T_\epsilon, n\})]$. Since $\mathsf{TOC}(\min\{T_\epsilon, n\}) = \mathsf{TOC}_0(\min\{T_\epsilon, n\}) + \mathsf{TOC}_1(\min\{T_\epsilon, n\})$ by definition, the result follows. □

Let us discuss the implications of Theorem 4. In [8], a lower bound on the total gradient sample complexity for stochastic optimization of non-convex, smooth functions is derived and shown to be, in the worst case, $\frac{C}{\varepsilon^4}$, for some positive constant C. This complexity lower bound holds even when exact function values $\phi(x)$ are also available. We note that the definition of complexity in [8] is the smallest number of sample gradient evaluations required to return a point x with $\mathbb{E}[\|\nabla\phi(x)\|] \leq \varepsilon$, which is different from $\mathsf{TOC}(T_\epsilon)$ which we are aiming to bound here. We believe that the lower bound in [8] applies to our definition as well, but this is a subject of a separate study.

In [4], it is shown that $\mathbb{E}[T_\varepsilon] \leq \frac{C_1}{\varepsilon^2}$ for some C_1 sufficiently large that depends on δ_1, δ_0, κ_{eg}, L and some algorithmic constants. Thus, if $n = \frac{C_1}{\varepsilon^2}$

in inequality (11) of Theorem 4 , as long as γ is sufficiently large, we obtain $\mathbb{E}[\mathsf{TOC}(\min\{T_\varepsilon, n\})] \leq \mathcal{O}\left(\left(\frac{\sigma_f^2}{\varepsilon^6} + \frac{\sigma_g^2}{\varepsilon^4}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$. In particular, the total gradient sample complexity is $\mathcal{O}\left(\frac{\sigma_g^2}{\varepsilon^4}\log\left(\frac{1}{\varepsilon}\right)\right)$, which essentially matches the complexity lower bound as described in [8] up to a logarithmic factor. The total function value sample complexity is worse than that of the gradient if σ_f^2 is large. However, if $\sigma_f^2 \leq \mathcal{O}(\sigma_g^2\varepsilon^2)$ (which often happens in practice since σ_g usually scales with the dimension of the problem, and tends to be much larger than σ_f), the total sample complexity bound of STORM matches the lower bound up to a logarithmic factor.

We note now that choosing $n = \frac{C_1}{\varepsilon^2}$ in Theorem 4 does not in fact guarantee that $T_\varepsilon < n$, since for STORM, only a bound on $\mathbb{E}[T_\varepsilon]$ has been derived. However, this statement can be made true in probability, thanks to Theorem 3, by simply applying Markov inequality for $n = C_2\frac{C_1}{\varepsilon^2}$ (where $C_2 > 1$).

Theorem 5 (High probability total sample complexity bound of first-order STORM) *For the first-order STORM algorithm applied to expected risk minimization, let n be chosen such that* $n \geq C_2\frac{C_1}{\varepsilon^2}$ *(for some* C_1 *sufficiently large so that* $\frac{C_1}{\varepsilon^2} \geq \mathbb{E}[T_\varepsilon]$*, and any* $C_2 > 1$*), and* γ *be chosen so that* $\gamma \geq \max\left\{\frac{1}{2}, \left(\frac{1}{2q}\right)^{\frac{\log(2\beta)}{(1+\omega)\log n}}\right\}$ *(for some* $\beta \leq \frac{1}{2}$*, and any* $\omega > 0$*). Then, with probability at least* $1 - \frac{1}{C_2} - \mathcal{O}(n^{-\omega})$*,*

$$\mathsf{TOC}(T_\varepsilon) \leq \mathcal{O}\left(\frac{\sigma_f^2}{\varepsilon^6} + \frac{\sigma_g^2}{\varepsilon^4}\right). \tag{12}$$

Proof The theorem is a simple application of Theorem 3 to the specific setting. □

Remark 2 1. Compared to the expected total sample complexity bound, this high probability bound is smaller by a log factor.

2. In [5], a first-order trust-region algorithm similar to STORM with the same first-order oracle (i.e. TR1.1 with $\epsilon_g = 0$), but with an exact zeroth-order oracle (i.e. TR1.0 with $\kappa_{ef} = \delta_0 = 0$) is analyzed. In this case, it is shown that $\mathbb{P}(T_\varepsilon > n) \leq \exp(-C_1 n)$ (for some constant C_1 that depends on δ_1), for any $n \geq \frac{C_2}{\varepsilon^2}$ (with some sufficiently large C_2). Using a similar application of Theorem 3, we can show that as long as γ is sufficiently large, the total gradient sample complexity of that trust-region algorithm is bounded above by $\mathcal{O}(\frac{\sigma_g^2}{\varepsilon^4})$ with probability at least $1-\exp(-C_1 n)-\mathcal{O}(n^{-\omega})$ (which is a significant improvement over the probability in Theorem 5).
3. Another first-order trust-region algorithm, with weaker oracle assumptions than those in [5] is introduced and analyzed in [6]. This algorithm relies on the first-order oracle as described in TR1.1 and the zeroth-order oracle as described in SS.0. For this algorithm, it is shown that $\mathbb{P}(T_\varepsilon > n) \leq 2\exp(-C_1 n) + \exp(-C_2)$ (C_2 being any positive constant), where $n = C_3\frac{C_2}{\varepsilon^2}$ for some sufficiently large C_3 and some positive C_1 that depends on δ_0 and δ_1. Thus, again, using Theorem 3

we can show that as long as γ is sufficiently large, the total sample complexity of the first-order trust-region algorithm in [6] is bounded above by $\mathcal{O}(\frac{\sigma_f^2}{\varepsilon^6} + \frac{\sigma_g^2}{\varepsilon^4})$ with probability at least $1 - 2\exp(-C_1 n) - \exp(-C_2) - \mathcal{O}(n^{-\omega})$.

4. In the event where σ_g and σ_f are not known and need to be estimated, the implications are as follows. Consider the first-order oracle for STORM (TR1.1)—there is a constant κ_{eg} which in general can be arbitrarily large. Its value affects the value of $\bar{\alpha}$, which in turn affects the total complexity bound, but theory still applies. Earlier in this section, we saw that the sample cost of TR1.1 depends on $\frac{\sigma_g}{\kappa_{eg}}$, which means that underestimating σ_g (and using a smaller sample size) is equivalent to using a correct value for σ_g and obtaining an oracle with a larger κ_{eg} than what one would have had with the correct σ_g. In this case, the price of underestimating σ_g is directly reflected in the complexity bounds. On the other hand, the same situation does not quite apply to σ_f because the corresponding κ_{ef} cannot be arbitrarily large (at least not in the STORM analysis).

5.2 Total sample complexity of SASS

We now consider the SASS algorithm,[1] analyzed in [3, 22] and described in Sect. 2.3. By Proposition 1, 2 and 4 of [22], Assumption 1 is satisfied, with $\bar{\alpha}$ as given in the propositions.

In the empirical risk minimization setting, the following **assumptions** on $l(x, d)$ are made in [22].

- Function value: It is assumed that $|l(x, d) - \phi(x)|$ is a subexponential random variable and that there is some $\sigma_f \geq 0$ such that for all x, $\mathrm{Var}_{d\sim\mathcal{D}}[l(x, d)] \leq \sigma_f^2$. For example, if $l(x, d)$ is uniformly bounded, then $|l(x, d) - \phi(x)|$ is subexponential.
- Gradient: It is assumed that $\mathbb{E}_{d\sim\mathcal{D}}[\nabla_x l(x, d)] = \nabla\phi(x)$, and that for some $M_c, M_v \geq 0$ and for all x,

$$\mathbb{E}_{d\sim\mathcal{D}} \|\nabla_x l(x, d) - \nabla\phi(x)\|^2 \leq M_c + M_v \|\nabla\phi(x)\|^2 . \tag{13}$$

This assumption is fairly general and is studied in the literature [23].

For non-convex functions, the stopping time is defined as $T_\varepsilon = \min\{k : \|\nabla\phi(x_k)\| \leq \varepsilon\}$, same as in the case of STORM. For strongly convex functions, the stopping time is defined as $T_\varepsilon = \min\{k : \phi(x_k) - \inf_x \phi(x) \leq \varepsilon\}$. To achieve the desired accuracy, oracles SS.0 and SS.1 have to be sufficiently accurate in the sense that ϵ_f and ϵ_g have to be sufficiently small with respect to ε. In the case of expected risk minimization, the oracles can be implemented for any ϵ_f and ϵ_g by choosing an appropriate mini-batch size. Thus, here we will first fix ε and then discuss the oracles that deliver sufficient accuracy for such ε, for the theory in [22] to apply.

Oracle costs per iteration In [22], it is shown that given the desired convergence tolerance ε, sufficiently accurate oracles SS.0 and SS.1 can be implemented for any step size parameter α as follows:

[1] This algorithm was also referred to as ALOE in [22]. Its name has been changed to SASS since [22].

- Zeroth-order oracle: Proposition 5 of [22] shows that a sufficiently accurate zeroth-order oracle can be obtained by using a minibatch of size

$$\mathsf{oc}_0(\alpha) = \begin{cases} \mathcal{O}(\sigma_f^2/\varepsilon^4), & \text{for the non-convex case,} \\ \mathcal{O}(\sigma_f^2/\varepsilon^2), & \text{for the strongly convex case.} \end{cases} \tag{14}$$

  Note that the cost of the zeroth-order oracle is independent of α.
- First-order oracle: Proposition 6 of [22] implies that a sufficiently accurate first-order oracle can be obtained by using a minibatch of size

$$\mathsf{oc}_1(\alpha) = \begin{cases} \mathcal{O}\left(\frac{M_c}{\varepsilon^2} + \frac{M_v}{\min\{\tau, \kappa\alpha\}^2}\right), & \text{for the non-convex case,} \\ \mathcal{O}\left(\frac{M_c}{\varepsilon} + \frac{M_v}{\min\{\tau, \kappa\alpha\}^2}\right), & \text{for the strongly convex case.} \end{cases} \tag{15}$$

  The cost of the first-order oracle is indeed non-increasing in α, so Assumption 2 is satisfied. For simplicity of the presentation and essentially without loss of generality, we will assume $\tau \geq \kappa\bar{\alpha}$.

Substituting these bounds into Theorem 2, we obtain the following expected total sample complexity.

Theorem 6 (Expected total sample complexity of SASS) *For the SASS algorithm applied to expected risk minimization, for any iteration number $n \in \mathbb{Z}^+$, and any $\gamma > (2q)^{\frac{1}{2}}$, we have*

- *Non-convex case:*

$$\mathbb{E}[\mathsf{TOC}(\min\{T_\varepsilon, n\})] \leq \mathcal{O}\left(\frac{\sigma_f^2}{\varepsilon^4} \cdot n + \frac{M_c}{\varepsilon^2} \cdot n + M_v \cdot n \left(n^{\log_{\frac{p}{q}}\left(\frac{1}{\gamma^2}\right)} \log_{\frac{p}{q}}(n)\right)\right).$$

- *Strongly convex case:*

$$\mathbb{E}[\mathsf{TOC}(\min\{T_\varepsilon, n\})] \leq \mathcal{O}\left(\frac{\sigma_f^2}{\varepsilon^2} \cdot n + \frac{M_c}{\varepsilon} \cdot n + M_v \cdot n \left(n^{\log_{\frac{p}{q}}\left(\frac{1}{\gamma^2}\right)} \log_{\frac{p}{q}}(n)\right)\right).$$

Moreover, if $\gamma \geq \left(\frac{q}{p}\right)^{\frac{\log c}{2\log n}}$ for some constant $c > 1$ (so that $n^{\log_{\frac{p}{q}}(\frac{1}{\gamma^2})} \leq c$), the above simplifies to

- *Non-convex case:*

$$\mathbb{E}[\mathsf{TOC}(\min\{T_\epsilon, n\})] \leq \mathcal{O}\left(n\left(\frac{\sigma_f^2}{\varepsilon^4} + \frac{M_c}{\varepsilon^2} + M_v \log(n)\right)\right). \tag{16}$$

- *Strongly convex case:*

$$\mathbb{E}[\mathsf{TOC}(\min\{T_\epsilon, n\})] \leq \mathcal{O}\left(n\left(\frac{\sigma_f^2}{\varepsilon^2} + \frac{M_c}{\varepsilon} + M_v \log(n)\right)\right). \tag{17}$$

Proof Since the cost of each call to the zeroth-order oracle (14) is independent of α, the total function value sample complexity over n iterations is simply obtained by multiplying (14) by n.

The cost of the first-order oracle (15) consists of two parts, $\mathsf{oc}_1(\alpha) = \mathsf{oc}_{1,1}(\alpha) + \mathsf{oc}_{1,2}(\alpha)$. The first part, $\mathsf{oc}_{1,1}(\alpha)$, is $\mathcal{O}(\frac{M_c}{\varepsilon^2})$. Since this is *independent* of α, the total contribution of this part to the total gradient sample complexity over n iterations is $n\,\mathsf{oc}_{1,1}(\alpha)$, which is bounded by $\mathcal{O}(\frac{M_c}{\varepsilon^2}\,n)$.

The second part of the cost of the first-order oracle is $\mathsf{oc}_{1,2}(\alpha) := \mathcal{O}(\frac{M_v}{\min\{\tau,\kappa\alpha\}^2})$. By Theorem 2, the total expected cost over n iterations from this part is bounded above by:

$$\begin{aligned}
&n\sum_{l=1}^{n}\min\left\{1, n\left(\frac{q}{p}\right)^l + \frac{2\sqrt{pq}}{(1-2\sqrt{pq})^2}(2q)^l\right\}\cdot \mathsf{oc}_{1,2}(\bar{\alpha}\gamma^l) + n\,\mathsf{oc}_{1,2}(\bar{\alpha}) \\
&\quad\leq \underbrace{n\sum_{l=1}^{n}\min\left\{1, n\left(\frac{q}{p}\right)^l\right\}\cdot \mathsf{oc}_{1,2}(\bar{\alpha}\gamma^l)}_{=:A} \\
&\quad\quad + \frac{2n\sqrt{pq}}{(1-2\sqrt{pq})^2}\underbrace{\sum_{l=1}^{n}(2q)^l\cdot \mathsf{oc}_{1,2}(\bar{\alpha}\gamma^l)}_{=:B} + n\mathsf{oc}_{1,2}(\bar{\alpha}).
\end{aligned}$$

Note that the expression above only involves $\mathsf{oc}_{1,2}(\alpha)$ for $\alpha \leq \bar{\alpha}$. Therefore, by our earlier assumption that $\tau \geq \kappa\bar{\alpha}$, we have $\mathsf{oc}_{1,2}(\alpha) = \mathcal{O}(\frac{M_v}{\kappa^2\alpha^2}) = \mathcal{O}(\frac{M_v}{\alpha^2})$ (since κ is a constant). We now use this to calculate upper bounds for A and B. Using similar arguments as the proof of Theorem 4, we have

$$A = \mathcal{O}\left(\frac{M_v}{\bar{\alpha}^2}\,\log_{\frac{p}{q}}(n)\, n^{\log_{\frac{p}{q}}\left(\frac{1}{\gamma^2}\right)}\right) \text{ and } B = \mathcal{O}\left(\frac{M_v}{\bar{\alpha}^2}\right).$$

Together with the previous arguments, the result follows. □

In [22], the iteration bound on T_ϵ is shown to be $\frac{C_1}{\varepsilon^2} + \log_{1/\gamma}\frac{\alpha_0}{\bar{\alpha}}$ in the non-convex case, and $C_2\log\frac{1}{\varepsilon} + \log_{1/\gamma}\frac{\alpha_0}{\bar{\alpha}}$ in the strongly convex case (for some C_1 and C_2 sufficiently large) in high probability. Thus, we can select n appropriately and derive the high probability bound on the total sample complexity of SASS using Theorem 3.

Theorem 7 (High probability total sample complexity of SASS) *For the SASS algorithm applied to expected risk minimization, let $n \geq \frac{C_1}{\varepsilon^2} + \log_{1/\gamma}\frac{\alpha_0}{\bar{\alpha}}$ in the non-convex case, and $n \geq C_2\log\frac{1}{\varepsilon} + \log_{1/\gamma}\frac{\alpha_0}{\bar{\alpha}}$ in the strongly convex case.*

For any $\omega > 0$, with probability at least $1 - 2\exp(-C_3 n) - \mathcal{O}(n^{-\omega})$ (for some $C_3 > 0$), we have

- *Non-convex case:*

$$\mathsf{TOC}(T_\varepsilon) \leq \mathcal{O}\left(\left(\frac{1}{\varepsilon^2} + \log_{1/\gamma}\frac{\alpha_0}{\bar{\alpha}}\right)\cdot\left(\frac{\sigma_f^2}{\varepsilon^4} + \frac{M_c}{\varepsilon^2} + \frac{M_v}{\varepsilon^{4(1+\omega)\log_{2q}(\gamma)}}\right)\right). \tag{18}$$

- *Strongly convex case:*

$$\mathsf{TOC}(T_\varepsilon) \leq \mathcal{O}\left(\left(\log\frac{1}{\varepsilon} + \log_{1/\gamma}\frac{\alpha_0}{\bar{\alpha}}\right)\cdot\left(\frac{\sigma_f^2}{\varepsilon^2} + \frac{M_c}{\varepsilon} + M_v\left(\log\frac{1}{\varepsilon}\right)^{2(1+\omega)\log_{2q}(\gamma)}\right)\right). \tag{19}$$

If $\gamma \in \left[\max\left\{\frac{1}{2}, \left(\frac{1}{2q}\right)^{\frac{\log(2\beta)}{(1+\omega)\log n}}\right\}, \left(\frac{\bar{\alpha}}{\alpha_0}\right)^{\frac{1}{cn}}\right]$, where β is any constant smaller than $\frac{1}{2}$, and c is any constant in $(0, 1)$, the above simplifies to

- *Non-convex case:*

$$\mathsf{TOC}(T_\varepsilon) \leq \mathcal{O}\left(\frac{1}{\varepsilon^2}\cdot\left(\frac{\sigma_f^2}{\varepsilon^4} + \frac{M_c}{\varepsilon^2} + \frac{M_v}{\beta^2}\right)\right). \tag{20}$$

- *Strongly convex case:*

$$\mathsf{TOC}(T_\varepsilon) \leq \mathcal{O}\left(\log\frac{1}{\varepsilon}\cdot\left(\frac{\sigma_f^2}{\varepsilon^2} + \frac{M_c}{\varepsilon} + \frac{M_v}{\beta^2}\right)\right). \tag{21}$$

Proof The bounds (18) and (19) follow by using (14) and (15) in Theorem 3, and Theorem 3.8 of [22].

The bounds (20) and (21) follow from (18) and (19), respectively, by using the fact that γ lies in the appropriate range and $\log_{1/\gamma}\frac{\alpha_0}{\bar{\alpha}} \leq cn$ with $c \in (0, 1)$.

In the non-convex case, we have $\frac{1}{\varepsilon^2} + \log_{1/\gamma}\frac{\alpha_0}{\bar{\alpha}} = \mathcal{O}(\frac{1}{\varepsilon^2})$ and $\frac{M_v}{\varepsilon^{4(1+\omega)\log_{2q}(\gamma)}} = \mathcal{O}(\frac{M_v}{\beta^2})$ when γ lies in the specified range. It is worth noting that $c \in (0, 1)$ implies there is some $n = \mathcal{O}(\frac{1}{\varepsilon^2})$ that satisfies $n \geq \frac{C_1}{\varepsilon^2} + \log_{1/\gamma}\frac{\alpha_0}{\bar{\alpha}}$. The result for the strongly convex case follows similarly. □

Remark 3 1. We consider some of the implications of Theorem 7 below. Similar implications hold for the expected total sample complexity.

- In the non-convex case, from (20), the total function value sample complexity is $\mathcal{O}\left(\frac{\sigma_f^2}{\varepsilon^6}\right)$ and the total gradient sample complexity is $\mathcal{O}\left(\frac{M_c}{\varepsilon^4} + \frac{M_v}{\varepsilon^2}\right)$. In particular, the total gradient sample complexity matches that of SGD, and it essentially matches the complexity lower bound as described in [8] (for a different definition of complexity). Specifically, if $\sigma_f = 0$ (i.e., function values

are exact), the lower bound in [8] applies and the total sample complexity of SASS matches it.
- If $M_c = 0$ (sometimes referred to as the interpolation case), then the total gradient sample complexity reduces to $\mathcal{O}\left(\frac{M_v}{\varepsilon^2}\right)$. Hence, the total gradient sample complexity matches that of SGD under interpolation [23, 24].
- In the strongly convex case, the total function value sample complexity is $\mathcal{O}(\frac{\sigma_f^2}{\varepsilon^2}\log\frac{1}{\varepsilon})$ and the total gradient sample complexity is $\mathcal{O}((\frac{M_c}{\varepsilon}+M_v)\log\frac{1}{\varepsilon})$. In particular, the total gradient sample complexity matches that of SGD up to a logarithmic term.

2. Our framework can be applied to the convex setting as well. We focus on the non-convex and strongly convex settings in this paper for brevity and to cleanly illustrate how our framework can be applied, since the convex case requires some additional complications in the presentation. These complications are present in the previous papers that analyze iteration complexity bounds, and are not specific to this paper. For example, the stopping time in the convex setting for SASS is defined in terms of two parameters $\varepsilon = (\varepsilon_0, \varepsilon_1)$ as follows: $T_\varepsilon = \min\{k : \phi(X_k) - \phi(x^*) \leq \varepsilon_0 \text{ or } \|\nabla\phi(X_k)\| \leq \varepsilon_1\}$.

6 Conclusion

We analyzed the behavior of the step size parameter in Algorithm 1, an adaptive stochastic optimization framework that encompasses a wide class of algorithms analyzed in recent literature. We derived a high probability lower bound for this parameter, and as a result, developed a simple strategy for controlling this lower bound.

For many settings, having a fixed lower bound on the step size parameter implies an upper bound on the cost of the oracles that compute the gradient and function estimates. We developed a framework to analyze the expected and high probability total oracle complexity for this general class of algorithms, and illustrated the use of it by deriving total sample complexity bounds for two specific algorithms—the first-order stochastic trust-region (STORM) algorithm [4] and a stochastic step search (SASS) algorithm [22] in the expected risk minimization setting. We showed that the sample complexity of both these algorithms essentially matches the complexity lower bound of first-order algorithms for stochastic non-convex optimization [8], which was not known before.

Acknowledgements The authors wish to thank James Allen Fill for pointing out useful references and the proof of Proposition 1. This work was partially supported by NSF Grants TRIPODS 17-40796, CCF 2008434, CCF 2140057 and ONR Grant N00014-22-1-2154. Billy Jin was partially supported by NSERC fellowship PGSD3-532673-2019.

References

1. Cartis, C., Scheinberg, K.: Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. Math. Program. **169**(2), 337–375 (2017). https://doi.org/10.1007/s10107-017-1137-4
2. Paquette, C., Scheinberg, K.: A stochastic line search method with expected complexity analysis. SIAM J. Optim. **30**(1), 349–376 (2020)
3. Berahas, A.S., Cao, L., Scheinberg, K.: Global convergence rate analysis of a generic line search algorithm with noise. SIAM J. Optim. 1489–1518 (2019)
4. Blanchet, J., Cartis, C., Menickelly, M., Scheinberg, K.: Convergence rate analysis of a stochastic trust-region method via supermartingales. INFORMS J. Optim. **1**(2), 92–119 (2019)
5. Gratton, S., Royer, C.W., Vicente, L.N., Zhang, Z.: Complexity and global rates of trust-region methods based on probabilistic models. IMA J. Numer. Anal. **38**(3), 1579–1597 (2018)
6. Cao, L., Berahas, A.S., Scheinberg, K.: First-and Second-Order High Probability Complexity Bounds for Trust-Region Methods with Noisy Oracles (2022). arXiv:2205.03667
7. Scheinberg, K., Xie, M.: Stochastic adaptive regularization method with cubics: a high probability complexity bound. In: Winter Simulation Conference (2023)
8. Arjevani, Y., Carmon, Y., Duchi, J.C., Foster, D.J., Srebro, N., Woodworth, B.: Lower Bounds for Non-convex Stochastic Optimization (2019). arXiv:1912.02365
9. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization (2014). arXiv:1412.6980
10. Tan, C., Ma, S., Dai, Y.-H., Qian, Y.: Barzilai–Borwein step size for stochastic gradient descent. In: Advances in Neural Information Processing Systems, vol. 29, pp. 685–693. Curran Associates Inc., NY (2016)
11. Rinaldi, F., Vicente, L., Zeffiro, D.: Stochastic trust-region and direct-search methods: a weak tail bound condition and reduced sample sizing (2023). arXiv:2202.11074
12. Shashaani, S., Hashemi, F.S., Pasupathy, R.: Astro-df: a class of adaptive sampling trust-region algorithms for derivative-free stochastic optimization. SIAM J. Optim. **28**(4), 3145–3176 (2018)
13. Regier, J., Jordan, M.I., McAuliffe, J.: Fast black-box variational inference through stochastic trust-region optimization. Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 2399–2408. Curran Associates Inc., NY (2017)
14. Berahas, A.S., Xie, M., Zhou, B.: A Sequential Quadratic Programming Method with High Probability Complexity Bounds for Nonlinear Equality Constrained Stochastic Optimization (2023). https://doi.org/10.48550/ARXIV.2301.00477
15. Menickelly, M., Wild, S.M., Xie, M.: A Stochastic Quasi-Newton Method in the Absence of Common Random Numbers (2023). https://doi.org/10.48550/ARXIV.2302.09128
16. Berahas, A.S., Cao, L., Choromanski, K., Scheinberg, K.: A theoretical and empirical comparison of gradient approximations in derivative-free optimization. Found. Comput. Math. **22**, 507–560 (2021)
17. Anscombe, F.J.: Rejection of outliers. Technometrics **2**, 123–146 (1960)
18. Yin, D., Chen, Y., Kannan, R., Bartlett, P.: Byzantine-robust distributed learning: towards optimal statistical rates. In: International Conference on Machine Learning, pp. 5650–5659. PMLR (2018)
19. Spall, J.C.: Stochastic optimization and the simultaneous perturbation method. In: Proceedings of the 31st Conference on Winter Simulation: Simulation—A Bridge to the Future—Volume 1. WSC '99, pp. 101–109. Association for Computing Machinery, New York, NY, USA (1999). https://doi.org/10.1145/324138.324170
20. Feller, W.: An Introduction to Probability Theory and Its Applications, vol. 1. Wiley (1968). http://www.amazon.ca/exec/obidos/redirect?tag=citeulike04-20&path=ASIN/0471257087
21. Kim, S., Pasupathy, R., Henderson, S.G.: A guide to sample average approximation. In: Handbook of simulation optimization, pp. 207–243. Springer (2015)
22. Jin, B., Scheinberg, K., Xie, M.: High probability complexity bounds for line search based on stochastic oracles. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P.S., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems, vol. 34, pp. 9193–9203 (2021). https://proceedings.neurips.cc/paper/2021/file/4cb811134b9d39fc3104bd06ce75abad-Paper.pdf
23. Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. SIAM Rev. **60**(2), 223–311 (2018)
24. Bach, F., Moulines, E.: Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In: Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a Meeting Held 12–14 December 2011,

Granada, Spain, pp. 451–459 (2011). http://papers.nips.cc/paper/4316-non-asymptotic-analysis-of-stochastic-approximation-algorithms-for-machine-learning