Oracle Complexity of Second-Order Methods for Finite-Sum Problems

Yossi Arjevani [1] **Ohad Shamir** [1]

Abstract

Finite-sum optimization problems are ubiquitous in machine learning, and are commonly solved using first-order methods which rely on gradient computations. Recently, there has been growing interest in *second-order* methods, which rely on both gradients and Hessians. In principle, second-order methods can require much fewer iterations than first-order methods, and hold the promise for more efficient algorithms. Although computing and manipulating Hessians is prohibitive for high-dimensional problems in general, the Hessians of individual functions in finite-sum problems can often be efficiently computed, e.g. because they possess a low-rank structure. Can second-order information indeed be used to solve such problems more efficiently? In this paper, we provide evidence that the answer – perhaps surprisingly – is negative, at least in terms of worst-case guarantees. We also discuss what additional assumptions and algorithmic approaches might potentially circumvent this negative result.

1. Introduction

We consider finite-sum problems of the form

$$\min_{\mathbf{w}\in\mathcal{W}} F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{w}), \tag{1}$$

where $\mathcal{W}$ is a closed convex subset of some Euclidean or Hilbert space, each f_i is convex and μ-smooth, and F is λ-strongly convex[1]. Such problems are ubiquitous in machine learning, for example in order to perform empirical risk minimization using convex losses.

To study the complexity of this and other optimization problems, it is common to consider an oracle model, where the optimization algorithm has no a-priori information about the objective function, and obtains information from an oracle which provides values and derivatives of the function at various domain points (Nemirovsky and Yudin, 1983). The complexity of the algorithm is measured in terms of the number of oracle calls required to optimize the function to within some prescribed accuracy.

Existing lower bounds for finite-sum problems show that using a first-order oracle, which given a point $\mathbf{w}$ and index $i = 1,\ldots,n$ returns $f_i(\mathbf{w})$ and $\nabla f_i(\mathbf{w})$, the number of oracle queries required to find an ϵ-optimal solution is at least of order

$$\Omega\left(n + \sqrt{\frac{n\mu}{\lambda}}\log\left(\frac{1}{\epsilon}\right)\right),$$

either under algorithmic assumptions or assuming the dimension is sufficiently large[2] (Agarwal and Bottou, 2014; Lan, 2015; Woodworth and Srebro, 2016; Arjevani and Shamir, 2016a). This is matched (up to log factors) by existing approaches, and cannot be improved in general.

An alternative to first-order methods are *second-order* methods, which also utilize Hessian information. A prototypical example is the Newton method, which given a (single) function F, performs iterations of the form

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha_t \left(\nabla^2 F(\mathbf{w})\right)^{-1} \nabla F(\mathbf{w}), \tag{2}$$

where $\nabla F(\mathbf{w}), \nabla^2 F(\mathbf{w})$ are the gradient and the Hessian of F at $\mathbf{w}$, and α_t is a step size parameter. Second-order methods can have extremely fast convergence, better than those of first-order methods (i.e. quadratic instead of linear). Moreover, they can be invariant to affine transformations of the objective function, and provably independent of its strong convexity and smoothness parameters (assuming e.g. self-concordance) (Boyd and Vandenberghe, 2004). A drawback of these methods, however, is that they can be computationally prohibitive. In the context of machine

[1]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel. Correspondence to: Yossi Arjevani <yossi.arjevani@weizmann.ac.il>, Ohad Shamir <ohad.shamir@weizmann.ac.il>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

[1]For a twice-differentiable function f, it is μ-smooth and λ-strongly convex if $\lambda I \preceq \nabla^2 f(\mathbf{w}) \preceq \mu I$ for all $\mathbf{w} \in \mathcal{W}$.

[2]Depending on how ϵ-optimality is defined precisely, and where the algorithm is assumed to start, these bounds may have additional factors inside the log. For simplicity, we present the existing bounds assuming ϵ is sufficiently small, so that a $\log(1/\epsilon)$ term dominates.

learning, we are often interested in high-dimensional problems (where the dimension d is very large), and the Hessians are $d \times d$ matrices which in general may not even fit into computer memory. However, for optimization problems as in Eq. (1), the Hessians of individual f_i often have a special structure. For example, a very common special case of finite-sum problems in machine learning is empirical risk minimization for linear predictors, where

$$f_i(\mathbf{w}) = \ell_i(\langle \mathbf{w}, \mathbf{x}_i \rangle),$$

where $\mathbf{x}_i$ is a training instance and ℓ_i is some loss function. In that case, assuming ℓ_i is twice-differentiable, the Hessian has the rank-1 form $\ell_i''(\langle \mathbf{w}, \mathbf{x}_i \rangle)\mathbf{x}_i\mathbf{x}_i^\top$. Therefore, the memory and computational effort involved with storing and manipulating the Hessian of this function is merely linear (rather than quadratic) in d. Thus, it is tractable even for high-dimensional problems.

Building on this, several recent papers proposed and analyzed second-order methods for finite-sum problems, which utilize Hessians of the individual functions f_i (see for instance Erdogdu and Montanari (2015); Agarwal et al. (2016); Pilanci and Wainwright (2015); Roosta-Khorasani and Mahoney (2016a;b); Bollapragada et al. (2016); Xu et al. (2016) and references therein). These can all be viewed as approximate Newton methods, which replace the actual Hessian $\nabla^2 F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^n \nabla^2 f_i(\mathbf{w})$ in Eq. (2) by some approximation, based for instance on the Hessians of a few individual functions f_i sampled at random. One may hope that such methods can inherit the favorable properties of second-order methods, and improve on the performance of commonly used first-order methods.

In this paper, we consider the opposite direction, and study *lower* bounds on the number of iterations required by algorithms using second-order (or possibly even higher-order) information, focusing on finite-sum problems which are strongly-convex and smooth. We make the following contributions:

- First, as a more minor contribution, we prove that in the standard setting of optimizing a *single* smooth and strongly convex function, second-order information cannot improve the oracle complexity compared to first-order methods (at least in high dimensions). Although this may seem unexpected at first, the reason is that the smoothness constraint must be extended to higher-order derivatives, in order for higher-order information to be useful. We note that this observation in itself is not new, and is briefly mentioned (without proof) in Nemirovsky and Yudin (1983, Section 7.2.6). Our contribution here is in providing a clean, explicit statement and proof of this result.
- We then turn to present our main results, which state (perhaps surprisingly) that under some mild algorithmic assumptions, and if the dimension is sufficiently large, the oracle complexity of second-order methods for finite-sum problems is no better than first-order methods, *even* if the finite-sum problem is composed of quadratics (which are trivially smooth to any order).
- Despite this pessimistic conclusion, our results also indicate what assumptions and algorithmic approaches might be helpful in circumventing it. In particular, it appears that better, dimension-dependent performance may be possible, if the dimension is moderate and the n individual functions in Eq. (1) are accessed adaptively, in a manner depending on the functions rather than fixed in advance (e.g. sampling them from a non-uniform distribution depending on their Hessians, as opposed to sampling them uniformly at random). This provides evidence to the necessity of adaptive sampling schemes, and a dimension-dependent analysis, which indeed accords with some recently proposed algorithms and derivations, e.g. (Agarwal et al., 2016; Xu et al., 2016). We note that the limitations arising from oblivious optimization schemes (in a somewhat stronger sense) was also explored in (Arjevani and Shamir, 2016a;b).

The paper is structured as follows: We begin in Sec. 2 with a lower bound for algorithms utilizing second-order information, in the simpler setting where there is a single function F to be optimized, rather than a finite-sum problem. We then turn to provide our main lower bounds in Sec. 3, and discuss their applicability to some existing approaches in Sec. 4. We conclude in Sec. 5, where we also discuss possible approaches to circumvent our lower bounds. The formal proofs of our results appear in Appendix A.

2. Strongly Convex and Smooth Optimization with a Second-Order Oracle

Before presenting our main results for finite-sum optimization problems, we consider the simpler problem of minimizing a single strongly-convex and smooth function F (or equivalently, Eq. (1) when $n = 1$), and prove a result which may be of independent interest.

To formalize the setting, we follow a standard oracle model, and assume that the algorithm does not have a-priori information on the objective function F, except the strong-convexity parameter λ and smoothness parameter μ. Instead, it has access to an oracle, which given a point $\mathbf{w} \in \mathcal{W}$, returns values and derivatives of F at $\mathbf{w}$ (either $\nabla F(\mathbf{w})$ for a first-order oracle, or $\nabla F(\mathbf{w}), \nabla^2 F(\mathbf{w})$ for a second-order oracle). The algorithm sequentially queries the oracle using $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{T-1}$, and returns the point $\mathbf{w}_T$. Our goal is to lower bound the number of oracle calls

T, required to ensure that $\mathbf{w}_T$ is an ϵ-suboptimal solution.

Given a first-order oracle and a strongly convex and smooth objective in sufficiently high dimensions, it is well-known that the worst-case oracle complexity is

$$\Omega(\sqrt{\mu/\lambda} \cdot \log(1/\epsilon))$$

(Nemirovsky and Yudin, 1983). What if we replace this by a second-order oracle, which returns both $\nabla^2 F(\mathbf{w})$ on top of $F(\mathbf{w}), \nabla F(\mathbf{w})$?

Perhaps unexpectedly, it turns out that this additional information does not substantially improve the worst-case oracle complexity bound, as evidenced by the following theorem:

Theorem 1. *For any μ, λ such that $\mu > 8\lambda > 0$, any $\epsilon \in (0,1)$, and any deterministic algorithm, there exists a μ-smooth, λ strongly-convex function F on $\mathbb{R}^d$ (for $d = \tilde{\mathcal{O}}(\sqrt{\mu/\lambda})$, hiding factors logarithmic in μ, λ, ϵ), such that the number of calls T to a second-order oracle, required to ensure that $F(\mathbf{w}_T) - \min_{\mathbf{w}\in\mathbb{R}^d} F(\mathbf{w}) \leq \epsilon \cdot (F(\mathbf{0}) - \min_{\mathbf{w}\in\mathbb{R}^d} F(\mathbf{w}))$, must be at least*

$$c\left(\sqrt{\frac{\mu}{8\lambda}} - 1\right) \cdot \log\left(\frac{(\lambda/\mu)^{3/2}}{c'\epsilon}\right),$$

where c, c' are positive universal constants.

For sufficiently large $\frac{\mu}{\lambda}$ and small ϵ, this complexity lower bound is $\Omega\left(\sqrt{\frac{\mu}{\lambda}} \cdot \log\left(\frac{1}{\epsilon}\right)\right)$, which matches existing lower and upper bounds for optimizing strongly-convex and smooth functions using first-order methods. As mentioned earlier, the observation that such first-order oracle bounds can be extended to higher-order oracles is also briefly mentioned (without proof) in Nemirovsky and Yudin (1983, Section 7.2.6). Also, the theorem considers deterministic algorithms (which includes standard second-order methods, such as the Newton method), but otherwise makes no assumption on the algorithm. Generalizing this result to randomized algorithms should be quite doable, based on the techniques developed in Woodworth and Srebro (2016). We leave a formal derivation to future work.

Although this result may seem surprising at first, it has a simple explanation: In order for Hessian information, which is local in nature, to be useful, there should be some regularity constraint on the Hessian, which ensures that it cannot change arbitrarily quickly as we move around the domain. A typical choice for a constraint of this kind is Lipschitz continuity which dictates that

$$\|\nabla^2 F(\mathbf{w}) - \nabla^2 F(\mathbf{w}')\| \leq L\|\mathbf{w} - \mathbf{w}'\|,$$

for some constant L. Indeed, the construction relies on a function which does not have Lipschitz Hessians: It is based on a standard lower bound construction for first-order oracles, but the function is locally "flattened" in certain directions around points which are to be queried by the algorithm. This is done in such a way, that the Hessian observed by the algorithm does not provide more information than the gradient, and cannot be used to improve the algorithm's performance.

3. Second-Order Oracle Complexity Bounds for Finite-Sum Problems

We now turn to study finite-sum problems of the form given in Eq. (1), and provide lower bounds on the number of oracle calls required to solve them, assuming a second-order oracle. To adapt the setting to a finite-sum problem, we assume that the second-order oracle is given both a point $\mathbf{w}$ and an index $i \in \{1, \ldots, n\}$, and returns $\{f_i(\mathbf{w}), \nabla f_i(\mathbf{w}), \nabla^2 f_i(\mathbf{w})\}$. The algorithm iteratively produces and queries the oracle with point-index pairs $\{(\mathbf{w}_t, i_t)\}_{t=1}^T$, with the goal of making the suboptimality (or expected suboptimality, if the algorithm is randomized) smaller than ϵ using a minimal number of oracle calls T.

In fact, the lower bound construction we use is such that each function f_i is quadratic. Unlike the construction of the previous section, such functions have a constant (and hence trivially Lipschitz) Hessian. Moreover, since any p-order derivative of a quadratic for $p > 2$ is zero, this means that our lower bounds automatically hold even if the oracle provides p-th order derivatives at any $\mathbf{w}$, for arbitrarily large p.

However, in order to provide a lower bound using quadratic functions, it is necessary to pose additional assumptions on the structure of the algorithm (unlike Thm. 1 which is purely information-based). To see why, note that without computational constraints, the algorithm can simply query the Hessians and gradients of each $f_i(\mathbf{w})$ at $\mathbf{w} = \mathbf{0}$, take the average to get $\nabla F(\mathbf{0}) = \frac{1}{n}\sum_{i=1}^n \nabla f_i(\mathbf{0})$ and $\nabla^2 F(\mathbf{0}) = \frac{1}{n}\sum_{i=1}^n \nabla^2 f_i(\mathbf{0})$, and return the exact optimum, which for quadratics equals $-\nabla^2 F(\mathbf{0})^{-1}\nabla F(\mathbf{0})$. Therefore, with second-order information, the best possible information-based lower bound for quadratics is no better than $\Omega(n)$. This is not a satisfying bound, since in order to attain it we need to invert the (possibly high-rank) $d \times d$ matrix $\nabla^2 F(\mathbf{0})$. Therefore, if we are interested in bounds for computationally-efficient algorithms, we need to forbid such operations.

Specifically, we will consider two algorithmic assumptions, which are stated below (their applicability to existing algorithms is discussed in the next section). The first assumption constrains the algorithm to query and return points $\mathbf{w}$ which are computable using linear-algebraic manipula-

tions of previous points, gradients and Hessians. Moreover, these manipulations can only depend on (at most) the last $\lfloor n/2 \rfloor$ Hessians returned by the oracle. As discussed previously, this assumption is necessary to prevent the algorithm from computing and inverting the full Hessian of F, which is computationally prohibitive. Formally, the assumption is the following:

Assumption 1 (Linear-Algebraic Computations)**.** *$\mathbf{w}_t$ belongs to the set $\mathcal{W}_t \subseteq \mathbb{R}^d$, defined recursively as follows: $\mathcal{W}_1 = \{\mathbf{0}\}$, and $\mathcal{W}_{t+1}$ is the closure of the set of vectors derived from $\mathcal{W}_t \cup \{\nabla f_{i_t}(\mathbf{w}_t)\}$ by a finite number of operations of the following form:*

- *$\mathbf{w}, \mathbf{w}' \to \alpha\mathbf{w} + \alpha'\mathbf{w}'$, where α, α' are arbitrary scalars.*
- *$\mathbf{w} \to H\mathbf{w}$, where H is any $d \times d$ matrix which has the same block-diagonal structure as*

$$\sum_{\tau=\max\{1,t-\lfloor n/2 \rfloor+1\}}^{t} \alpha_\tau \nabla^2 f_{i_\tau}(\mathbf{w}_\tau), \tag{3}$$

for some arbitrary $\{\alpha_\tau\}$.

The first bullet allows to take arbitrary linear combinations of previous points and gradients, and already covers standard first-order methods and their variants. As to the second bullet, by "same block-diagonal structure", we mean that if the matrix in Eq. (3) can be decomposed to r diagonal blocks of size $d_1, \ldots, d_r$ in order, then H can also be decomposed into r blocks of size $d_1, \ldots, d_r$ in order (note that this does not exclude the possibility that each such block is composed of additional sub-blocks). To give a few examples, if we let H_t be the matrix in Eq. (3), then we may have:

- $H = H_t$,
- $H = H_t^{-1}$ if H_t is invertible, or its pseudoinverse,
- $H = (H_t + D)^{-1}$ (where D is some arbitrary diagonal matrix, possibly acting as a regularizer),
- H is a truncated SVD decomposition of H_t (or again, $H_t + D$ or $(H_t + D)^{-1}$ for some arbitrary diagonal matrix D) or its pseudoinverse.

Moreover, for quadratic functions, it is easily verified that the assumption also allows prox operations (i.e. returning $\arg\min_{\mathbf{w}} f_i(\mathbf{w}) + \frac{\rho}{2}\|\mathbf{w} - \mathbf{w}'\|^2$ for some ρ, i and previously computed point $\mathbf{w}'$). Also, note that the assumption places no limits on the number of such operations allowed between oracle calls. However, crucially, all these operations can be performed starting from a linear combination of at most $\lfloor n/2 \rfloor$ recent Hessians. As mentioned earlier, this is necessary, since if we could compute the average of all Hessians, then we could implement the Newton method. The assumption that the algorithm only "remembers" the last $\lfloor n/2 \rfloor$ Hessians is also realistic, as existing computationally-efficient methods seek to use much fewer than n individual Hessians at a time. We note that the choice of $\lfloor n/2 \rfloor$ is rather arbitrary, and can be replaced by αn for any constant $\alpha \in (0, 1)$. Also, the way the assumption is formulated, the algorithm is assumed to be initialized at the origin $\mathbf{0}$. However, this is merely for simplicity, and can be replaced by any other fixed vector (the lower bound will hold by shifting the constructed "hard" function appropriately).

The second (optional) assumption we will consider constrains the indices chosen by the algorithm to be oblivious, in the following sense:

Assumption 2 (Index Obliviousness)**.** *The indices $i_1, i_2, \ldots$ chosen by the algorithm are independent of $f_1, \ldots, f_n$.*

To put this assumption differently, the indices may just as well be chosen before the algorithm begins querying the oracle. This can include, for instance, sampling functions f_i uniformly at random from $f_1, \ldots, f_n$, and performing deterministic passes over $f_1, \ldots, f_n$ in order. As we will see later on, this assumption is not strictly necessary, and can be removed at the cost of a somewhat weaker result. Nevertheless, the assumption covers all optimal first-order algorithms, as well as most second-order methods we are aware of (see Sec. 4 for more details).

With these assumptions stated, we can finally turn to present the main result of this section:

Theorem 2. *For any $n > 1$, any $\mu > \lambda > 0$, any $\epsilon \in (0, c)$ (for some universal constant $c > 0$), and any (possibly randomized) algorithm satisfying Assumptions 1 and 2, there exists μ-smooth, λ-strongly convex quadratic functions $f_1, \ldots, f_n : \mathbb{R}^d \to \mathbb{R}$ (for $d = \tilde{\mathcal{O}}(1 + \sqrt{\mu/\lambda}n)$, hiding factors logarithmic in $n, \mu, \lambda, \epsilon$), such that the number of calls T to a second-order oracle, so that*

$$\mathbb{E}\left[F(\mathbf{w}_T) - \min_{\mathbf{w}\in\mathbb{R}^d} F(\mathbf{w})\right] \leq \epsilon \cdot \left(F(\mathbf{0}) - \min_{\mathbf{w}\in\mathbb{R}^d} F(\mathbf{w})\right),$$

must be at least

$$\Omega\left(n + \sqrt{\frac{n\mu}{\lambda}} \cdot \log\left(\frac{(\lambda/\mu)^{3/2}\sqrt{n}}{\epsilon}\right)\right).$$

Comparing this with the (tight) first-order oracle complexity bounds discussed in the introduction, we see that the

lower bound is the same up to log-factors, despite the availability of second-order information. In particular, the lower bound exhibits none of the favorable properties associated with full second-order methods, which can compute and invert Hessians of F: Whereas the full Newton method can attain $\mathcal{O}(\log\log(1/\epsilon))$ rates, and be independent of μ, λ if F satisfies a self-concordance property (Boyd and Vandenberghe, 2004), here we only get a linear $\mathcal{O}(\log(1/\epsilon))$ rate, and there is a strong dependence on μ, λ, even though the function is quadratic and hence self-concordant.

The proof of the theorem is based on a randomized construction, which can be sketched as follows: We choose indices $j_1, \ldots, j_{d-1} \in \{1, \ldots, n\}$ independently and uniformly at random, and define

$$f_i(\mathbf{w}) = a \cdot w_1^2 + \hat{a} \cdot \sum_{l=1}^{d-1} \mathbf{1}_{j_l=i}(w_l - w_{l+1})^2 + \bar{a} \cdot w_d^2 - \tilde{a} \cdot w_1 + \frac{\lambda}{2}\|\mathbf{w}\|^2,$$

where $\mathbf{1}_A$ is the indicator function of the event A, and $a, \hat{a}, \bar{a}, \tilde{a}$ are parameters chosen based on λ, μ, n. The average function $F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^n f_i(\mathbf{w})$ equals

$$F(\mathbf{w}) = a \cdot w_1^2 + \frac{\hat{a}}{n} \cdot \sum_{l=1}^{d-1}(w_l - w_{l+1})^2 + \bar{a} \cdot w_d^2 - \tilde{a} \cdot w_1 + \frac{\lambda}{2}\|\mathbf{w}\|^2.$$

By setting the parameters appropriately, it can be shown that F is λ-strongly convex and each f_i is μ-smooth. Moreover, the optimum of F has the form $(q, q^2, q^3, \ldots, q^d)$ for

$$q = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1},$$

where

$$\kappa = \frac{\frac{\mu}{\lambda} - 1}{n} + 1 \tag{4}$$

is the so-called condition number of F. The proof is based on arguing that after T oracle calls, the points computable by any algorithm satisfying Assumptions 2 and 1 must have 0 values at all coordinates larger than some l_T, hence the squared distance of $\mathbf{w}_T$ from the optimum must be at least $\sum_{i=l_T+1}^d q^{2i}$, which leads to our lower bound. Thus, the proof revolves around upper bounding l_T. We note that a similar construction of F was used in some previous first-order lower bounds under algorithmic assumptions (e.g. Nesterov (2013); Lan (2015), as well as Arjevani and Shamir (2015) in a somewhat different context). The main difference is in how we construct the individual functions f_i, and in analyzing the effect of second-order rather than just first-order information.

To upper bound l_T, we let l_t (where $t = 1, \ldots, T$) be the largest non-zero coordinate in $\mathbf{w}_t$, and track how l_t increases with t. The key insight is that if $\mathbf{w}_1, \ldots, \mathbf{w}_{t-1}$ are zero beyond some coordinate l, then any linear combinations of them, *as well as multiplying them by matrices based on second-order information*, as specified in Assumption 1, will still result in vectors with zeros beyond coordinate l. The only way to "advance" and increase the set of non-zero coordinates is by happening to query the function f_{j_l}. However, since the indices of the queried functions are chosen obliviously, whereas each j_l is chosen uniformly at random, the probability of this happening is quite small, of order $1/n$. Moreover, we show that even if this event occurs, we are unlikely to "advance" by more than $\mathcal{O}(1)$ coordinates at a time. Thus, the algorithm essentially needs to make $\Omega(n)$ oracle calls in expectation, in order to increase the number of non-zero coordinates by $\mathcal{O}(1)$. It can be shown that the number of coordinates needed to get an ϵ-optimal solution is $\tilde{\Omega}(\sqrt{\mu/n\lambda} \cdot \log(1/\epsilon))$ (hiding some log-factors). Therefore, the total number of oracle calls is about n times larger, namely $\tilde{\Omega}(\sqrt{n\mu/\lambda} \cdot \log(1/\epsilon))$. To complete the theorem, we also provide a simple and separate $\Omega(n)$ lower bound, which holds since each oracle call gives us information on just one of the n individual functions $f_1, \ldots, f_n$, and we need some information on most of them in order to get a close-to-optimal solution.

When considering non-oblivious (i.e., adaptive) algorithms, the construction used in Thm. 2 fails as soon as the algorithm obtains the Hessians of all the individual functions (potentially, after n oracle queries). Indeed, knowing the Hessians of f_i, one can devise an index-schedule which gains at least one coordinate at every iteration (by querying the function which holds the desired 2×2 block), as opposed to $O(1/n)$ on average in the oblivious case. Nevertheless, as mentioned before, we can still provide a result similar to Thm. 2 even if the indices are chosen adaptively, at the cost of a much larger dimension:

Theorem 3. *Thm. 2 still holds if one omits Assumption 2, and with probability 1 rather than in expectation, at the cost of requiring an exponentially larger dimensionality of* $d = n^{\tilde{\mathcal{O}}\left(1+\sqrt{\mu/\lambda n}\right)}$.

The proof is rather straightforward: Making the dependence on the random indices $j_1, \ldots, j_{d-1}$ explicit, the quadratic construction used in the previous theorem can be written as

$$F^{j_1,\ldots,j_{d-1}}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^n f_i^{j_1,\ldots,j_{d-1}}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^n \mathbf{w}^\top A_i^{j_1,\ldots,j_{d-1}}\mathbf{w} - \tilde{a}\langle \mathbf{e}_1, \mathbf{w}\rangle + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

for some $d \times d$ matrix $A_i^{j_1,\ldots,j_{d-1}}$ dependent on $j_1, \ldots, j_{d-1}$, and a fixed parameter $\tilde{a}$. Now, we create n huge block-diagonal matrices $A_1, \ldots, A_n$, where each A_i

contains $A_i^{j_1,\ldots,j_{d-1}}$ for each of the n^{d-1} possible choices of $j_1,\ldots,j_{d-1}$ along its diagonal (in some canonical order), and one huge vector

$$\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_{d+1} + \ldots + \mathbf{e}_{(n^{d-1}-1)d+1}.$$

We then let

$$\begin{aligned} F(\mathbf{w}) &= \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{w}) \\ &= \frac{1}{n}\sum_{i=1}^{n} \mathbf{w}^\top A_i \mathbf{w} - \tilde{a}\langle \mathbf{e}, \mathbf{w}\rangle + \frac{\lambda}{2}\|\mathbf{w}\|^2. \end{aligned}$$

This function essentially combines all n^{d-1} problems $F^{j_1,\ldots,j_{d-1}}$ simultaneously, where each $F^{j_1,\ldots,j_{d-1}}$ is embedded in a disjoint set of coordinates. Due to the block-diagonal structure of each A_i, this function inherits the strong-convexity and smoothness properties of the original construction. Moreover, to optimize this function, the algorithm needs to "solve" all n^{d-1} problems simultaneously, using the same choice of indices $i_1, i_2, \ldots$. Using a combinatorial argument which parallels the probabilistic argument in the proof of Thm. 2, we can show that no matter how these indices are chosen, the average number of non-zero coordinates of the iterates cannot grow too rapidly, and lead to the same bound as in Thm. 2. Since the construction is deterministic, and applies no matter how the indices are chosen, the lower bound holds deterministically, rather than in expectation as in Thm. 2.

Lastly, it is useful to consider how the bounds stated in Thm. 2 and Thm. 3 differ when the dimension d is fixed and finite. Inspecting the proofs of both theorems reveals that in both cases the suboptimality, as a function of the iteration number T, has a linear convegence rate bounded from below by

$$\mathbb{E}\left[\frac{F(\mathbf{w}_T) - F(\mathbf{w}^\star)}{F(\mathbf{0}) - F(\mathbf{w}^\star)}\right] \geq \Omega(1)\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{\mathcal{O}\left(\frac{T}{n}\right)} \qquad (5)$$

(where κ is as defined in Eq. (4), and $\Omega(1)$ hides dependencies on the problem parameters, but is independent of T). However, whereas the bound established in Thm. 2 is valid for $\mathcal{O}(d)$ number of iterations, Thm. 3 applies to a much restricted range of roughly $\log(d)/\log(n)$ iterations. This indicate that adaptive optimization algorithms might be able to gain a super-linear convergence rate after a significantly smaller number of iterations in comparison to oblivious algorithms (see (Arjevani and Shamir, 2016a) for a similar discussion regarding first-order methods). That being said, trading obliviousness for adaptivity may increase the per-iteration cost and reduce numerical stability.

4. Comparison to Existing Approaches

As discussed in the introduction, there has been a recent burst of activity involving second-order methods for solving finite-sum problems, relying on Hessians of individual functions f_i. In this section, we review the main algorithmic approaches and compare them to our results. The bottom line is that most existing approaches satisfy the assumptions stated in Sec. 3, and therefore our lower bounds will apply, at least in a worst-case sense. A possible exception to this is the Newton sketch algorithm (Pilanci and Wainwright, 2015), which relies on random projections, but on the flip side is computationally expensive.

Turning to the details, existing approaches are based on taking the standard Newton iteration for such problems,

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t - \alpha_t \left(\nabla^2 F(\mathbf{w}_t)\right)^{-1} \nabla F(\mathbf{w}_t) \\ &= \mathbf{w}_t - \alpha_t \left(\frac{1}{n}\sum_{i=1}^{n} \nabla^2 f_i(\mathbf{w}_t)\right)^{-1} \left(\frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{w}_t)\right), \end{aligned}$$

and replacing the inverse Hessian term $\left(\frac{1}{n}\sum_{i=1}^{n} \nabla^2 f_i(\mathbf{w})\right)^{-1}$ (and sometimes the vector term $\frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{w})$ as well) by some approximation which is computationally cheaper to compute. One standard and well-known approach is to use only gradient information to construct such an approximation, leading to the family of quasi-Newton methods (Nocedal and Wright, 2006). However, as they rely on first-order rather than second-order information, they are orthogonal to the topic of our work, and are already covered by existing complexity lower bounds for first-order oracles.

Turning to consider Hessian approximation techniques using second-order information, perhaps the simplest and most intuitive approach is sampling: Since the Hessian equals the average of many individual Hessians,

$$\nabla^2 F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \nabla^2 f_i(\mathbf{w}),$$

we can approximate it by taking a sample S of indices in $\{1,\ldots,n\}$ uniformly at random, compute the Hessians of the corresponding individual functions, and use the approximation

$$\nabla^2 F(\mathbf{w}) \approx \frac{1}{|S|}\sum_{i\in S} \nabla^2 f_i(\mathbf{w}).$$

If $|S|$ is large enough, then by concentration of measure arguments, this sample average should be close to the actual Hessian $\nabla^2 F(\mathbf{w})$. On the other hand, if $|S|$ is not too large, then the resulting matrix is easier to invert (e.g. because it has a rank of only $\mathcal{O}(|S|)$, if each individual Hessian has rank $\mathcal{O}(1)$, as in the case of linear predictors). Thus, one can hope that the right sample size will lead

to computational savings. There have been several rigorous studies of such "subsampled Newton" methods, such as Erdogdu and Montanari (2015); Roosta-Khorasani and Mahoney (2016a;b); Bollapragada et al. (2016) and references therein. However, our lower bound in Thm. 2 holds for such an approach, since it satisfies both Assumption 2 and 1. As expected, the existing worst-case complexity upper bounds are no better than our lower bound.

(Xu et al., 2016) recently proposed a subsampled Newton method, together with *non-uniform* sampling, which assigns more weight to individual functions which are deemed more "important". This is measured via properties of the Hessians of the functions, such as their norms or via leverage scores. This approach breaks Assumption 2, as the sampled indices are now chosen in a way dependent on the individual functions. However, our lower bound in Thm. 3, which does not require this assumption, still applies to such a method.

A variant of the subsampled Newton approach, studied in Erdogdu and Montanari (2015), uses a low-rank approximation of the sample Hessian (attained by truncated SVD), in lieu of the sample Hessian itself. However, this still falls in the framework of Assumption 1, and our lower bound still applies.

A different approach to approximate the full Hessian is via randomized sketching techniques, which replace the Hessian $\nabla^2 F(\mathbf{w})$ by a low-rank approximation of the form

$$(\nabla^2 F(\mathbf{w}))^{1/2} S S^\top (\nabla^2 F(\mathbf{w}))^{1/2},$$

where $S \in \mathbb{R}^{d\times m}, m \ll d$ is a random sketching matrix, and $(\nabla^2 F(\mathbf{w}))^{1/2}$ is the matrix square root of $\nabla^2 F(\mathbf{w})$. This approach forms the basis of the Newton sketch algorithm proposed in Pilanci and Wainwright (2015). This approach currently escapes our lower bound, since it violates Assumption 1. That being said, this approach is inherently expensive in terms of computational resources, as it requires us to compute the square root of the full Hessian matrix. Even under favorable conditions, this requires us to perform a full pass over all functions $f_1, \ldots, f_n$ at every iteration. Moreover, existing iteration complexity upper bounds have a strong dependence on both μ/λ as well as the dimension d, and are considerably worse than the lower bound of Thm. 2. Therefore, we conjecture that this approach cannot lead to better worst-case results.

Agarwal et al. (2016) develop another line of stochastic second-order methods, which are based on the observation that the Newton step $(\nabla^2 F(\mathbf{w}))^{-1}\nabla F(\mathbf{w})$ is the solution of the system of linear equations

$$\nabla^2 F(\mathbf{w})\mathbf{x} = \nabla F(\mathbf{w}).$$

Thus, one can reduce the optimization problem to solving this system as efficiently as possible. The basic variant of their algorithm (denoted as LiSSA) relies on operations of the form

$$\mathbf{w} \mapsto (I - \nabla^2 f_i(\mathbf{w}))\mathbf{w}$$

(for i sampled uniformly at random), as well as linear combinations of such vectors, which satisfy our assumptions. A second variant, LiSSA-Quad, re-phrases this linear system as the finite-sum optimization problem

$$\min_{\mathbf{x}} \ \mathbf{x}^\top \nabla^2 F(\mathbf{w})\mathbf{x} + \nabla F(\mathbf{w})^\top \mathbf{x} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}^\top \nabla^2 f_i(\mathbf{w})\mathbf{x} + \nabla f_i(\mathbf{w})^\top \mathbf{x},$$

and uses some first-order method for finite-sum problems in order to solve it. Since individual gradients of this objective are of the form $\nabla^2 f_i(\mathbf{w})\mathbf{x} + \nabla f_i(\mathbf{w})$, and most state-of-the-art first-order methods pick indices i obliviously, this approach also satisfies our assumptions, and our lower bounds apply. Yet another proposed algorithm, LiSSA-Sample, is based on replacing the optimization problem above by

$$\min_{\mathbf{x}} \ \mathbf{x}^\top \nabla^2 F(\mathbf{w}) B^{-1}\mathbf{x} + \nabla F(\mathbf{w})^\top \mathbf{x}, \tag{6}$$

where B is some invertible matrix, solving it (with the optimum being equal to $B(\nabla^2 F(\mathbf{w}))^{-1}\nabla F(\mathbf{w})$), and multiplying the solution by B^{-1} to recover the solution $(\nabla^2 F(\mathbf{w}))^{-1}\nabla F(\mathbf{w})$ to the original problem. In order to get computational savings, B is chosen to be a linear combination of $\mathcal{O}(d\log(d))$ sampled individual hessians $\nabla^2 f_i(\mathbf{w})$, where it is assumed that $d\log(d) \ll n$, and the sampling and weighting is carefully chosen (based on the Hessians) so that Eq. (6) has strong convexity and smoothness parameters within a constant of each other. As a result, Eq. (6) can be solved fast using standard gradient descent, taking steps along the gradient, which equals $\nabla^2 F(\mathbf{w})B^{-1}\mathbf{x} + \nabla F(\mathbf{w})$ at any point $\mathbf{x}$. This gradient is again computable under 1 (using $\mathcal{O}(n)$ oracle calls), since B is a linear combination of $d\log(d) \ll n$ sampled individual Hessians. Thus, our lower bound (in the form of Thm. 3) still applies to such methods.

That being said, it is important to note that the complexity upper bound attained in Agarwal et al. (2016) for LiSSA-Sample is on the order of

$$\tilde{\mathcal{O}}((n + \sqrt{d\mu/\lambda}) \cdot \mathrm{polylog}(1/\epsilon))$$

(at least asymptotically as $\epsilon \to 0$), which can be better than our lower bound if $d \ll n$. There is no contradiction, since the lower bound in Thm. 3 only applies for a dimension d much larger than n. Interestingly, our results also indicate that an adaptive index sampling scheme is *necessary* to get this kind of improved performance when $d \ll n$: Otherwise, it could violate Thm. 2, which establishes a lower

bound of $\tilde{\mathcal{O}}(n + \sqrt{n\mu/\lambda})$ even if the dimension is quite moderate ($d = \tilde{\mathcal{O}}(1 + \sqrt{\mu/\lambda n})$, which is $\ll n$ under the mild assumption that $\mu/\lambda \ll n^3$).

The observation that an adaptive scheme (breaking assumption 2) can help performance when $d \ll n$ is also seen in the lower bound construction used to prove Thm. 2: If μ, λ, n are such that the required dimension d is $\ll n$, then it means that only the functions $f_{j_1}, \ldots, f_{j_{d-1}}$, which are a small fraction of all n individual functions, are informative and help us reduce the objective value. Thus, sampling these functions in an adaptive manner is imperative to get better complexity than the bound in Thm. 2. Based on the fact that only at most $d-1$ out of n functions are relevant in the construction, we conjecture that the possible improvement in the worst-case oracle complexity of such schemes may amount to replacing dependencies on n with dependencies on d, which is indeed the type of improvement attained (for small enough ϵ) in Agarwal et al. (2016).

Finally, we note that Agarwal et al. (2016) proposes another algorithm tailored to self-concordant functions, with runtime independent of the smoothness and strong convexity parameters of the problem. However, it requires performing ≥ 1 full Newton steps, so the runtime is prohibitive for large-scale problems (indeed, for quadratics as used in our lower bounds, even a single Newton step suffices to compute an exact solution).

5. Summary and Discussion

In this paper, we studied the oracle complexity for optimization problems, assuming availability of a second-order oracle. This is in contrast to most existing oracle complexity results, which focus on a first-order oracle. First, we formally proved that in the standard setting of strongly-convex and smooth optimization problems, second-order information does not significantly improve the oracle complexity, and further assumptions (i.e. Lipschitzness of the Hessians) are in fact necessary. We then presented our main lower bounds, which show that for finite-sum problems with a second-order oracle, under some reasonable algorithmic assumptions, the resulting oracle complexity is – again – not significantly better than what can be obtained using a first-order oracle. Moreover, this is shown using quadratic functions, which have 0 derivatives of order larger than 2. Hence, our lower bounds apply even if we have access to an oracle returning derivatives of order p for all $p \geq 0$, and the function is smooth to any order. In Sec. 4, we studied how our framework and lower bounds are applicable to most existing approaches.

Although this conclusion may appear very pessimistic, they are actually useful in pinpointing potential assumptions and approaches which may circumvent these lower bounds. In particular:

- Our lower bound for algorithms employing adaptive index sampling schemes (Thm. 3) only hold when the dimension d is very large. This leaves open the possibility of better (non index-oblivious) algorithms when d is moderate, as was recently demonstrated in the context of the LiSSA-Sample algorithm of Agarwal et al. (2016) (at least for small enough ϵ). As discussed in the previous section, we conjecture that the possible improvement in the worst-case oracle complexity of such schemes may amount to replacing dependencies on n with dependencies on d.

- It might be possible to construct algorithms breaking Assumption 1, e.g. by using operations which are not linear-algebraic. That being said, we currently conjecture that this assumptions can be significantly relaxed, and similar results would hold for any algorithm which has "significantly" cheaper iterations (in terms of runtime) compared to the Newton method.

- Our lower bounds are worst-case over smooth and strongly-convex individual functions f_i. It could be that by assuming more structure, better bounds can be obtained. For example, as discussed in the introduction, an important special case is when $f_i(\mathbf{w}) = \ell_i(\mathbf{x}_i^\top \mathbf{w})$ for some scalar function ℓ_i and vector $\mathbf{x}_i$. Our construction in Thm. 2 does not quite fit this structure, although it is easy to show that we still get functions of the form $f_i(\mathbf{w}) = \ell_i(X_i^\top \mathbf{w})$, where X_i has $\mathcal{O}(1 + d/n) = \tilde{\mathcal{O}}(1 + \sqrt{\mu/\lambda n^3})$ rows in expectation, which is $\tilde{\mathcal{O}}(1)$ under a broad parameter regime. We believe that the difference between $\tilde{\mathcal{O}}(1)$ rows and 1 row is not significant in terms of the attainable oracle complexity, but we may be wrong. Another possibility is to provide results depending on more delicate spectral properties of the function, beyond its strong convexity and smoothness, which may lead to better results and algorithms under favorable assumptions.

- Our lower bounds in Sec. 3, which establish a linear convergence rate (logarithmic dependence on $\log(1/\epsilon)$), are non-trivial only if the optimization error ϵ is sufficiently small. This does not preclude the possibility of attaining better initial performance when ϵ is relatively large.

In any case, we believe that our work lays the foundation for a more comprehensive study of the complexity of efficient second-order methods, for finite-sum and related optimization and learning problems.

ACKNOWLEDGMENTS

This research is supported in part by an FP7 Marie Curie CIG grant and an Israel Science Foundation grant 425/13.

References

Alekh Agarwal and Leon Bottou. A lower bound for the optimization of finite sums. *arXiv preprint arXiv:1410.0723*, 2014.

Naman Agarwal, Brian Bullins, and Elad Hazan. Second order stochastic optimization in linear time. *arXiv preprint arXiv:1602.03943*, 2016.

Yossi Arjevani and Ohad Shamir. Communication complexity of distributed convex learning and optimization. In *Advances in Neural Information Processing Systems*, pages 1756–1764, 2015.

Yossi Arjevani and Ohad Shamir. Dimension-free iteration complexity of finite sum optimization problems. In *Advances in Neural Information Processing Systems*, pages 3540–3548, 2016a.

Yossi Arjevani and Ohad Shamir. On the iteration complexity of oblivious first-order optimization algorithms. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 908–916, 2016b.

Raghu Bollapragada, Richard Byrd, and Jorge Nocedal. Exact and inexact subsampled newton methods for optimization. *arXiv preprint arXiv:1609.08502*, 2016.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Murat A Erdogdu and Andrea Montanari. Convergence rates of sub-sampled newton methods. In *Advances in Neural Information Processing Systems*, pages 3052–3060, 2015.

Guanghui Lan. An optimal randomized incremental gradient method. *arXiv preprint arXiv:1507.02000*, 2015.

A. Nemirovsky and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley-Interscience, 1983.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

Mert Pilanci and Martin J Wainwright. Newton sketch: A linear-time optimization algorithm with linear-quadratic convergence. *arXiv preprint arXiv:1505.02250*, 2015.

Farbod Roosta-Khorasani and Michael W Mahoney. Sub-sampled newton methods i: globally convergent algorithms. *arXiv preprint arXiv:1601.04737*, 2016a.

Farbod Roosta-Khorasani and Michael W Mahoney. Sub-sampled newton methods ii: Local convergence rates. *arXiv preprint arXiv:1601.04738*, 2016b.

Blake Woodworth and Nathan Srebro. Tight complexity bounds for optimizing composite objectives. *arXiv preprint arXiv:1605.08003*, 2016.

Peng Xu, Jiyan Yang, Farbod Roosta-Khorasani, Christopher Ré, and Michael W Mahoney. Sub-sampled newton methods with non-uniform sampling. *arXiv preprint arXiv:1607.00559*, 2016.