Thinking Inside the Ball: Near-Optimal Minimization of the Maximal Loss

Yair Carmon YCARMON@CS.TAU.AC.IL
Tel Aviv University
Arun Jambulapati JMBLPATI@STANFORD.EDU
Stanford University
Yujia Jin YUJIAJIN@STANFORD.EDU
Stanford University
Aaron Sidford SIDFORD@STANFORD.EDU
Stanford University

Editors: Mikhail Belkin and Samory Kpotufe

Abstract

We characterize the complexity of minimizing $\max_{i\in[N]} f_i(x)$ for convex, Lipschitz functions $f_1,\ldots,f_N$. For non-smooth functions, existing methods require $O(N\epsilon^{-2})$ queries to a first-order oracle to compute an ϵ-suboptimal point and $\widetilde{O}(N\epsilon^{-1})$ queries if the f_i are $O(1/\epsilon)$-smooth. We develop methods with improved complexity bounds of $\widetilde{O}(N\epsilon^{-2/3}+\epsilon^{-8/3})$ in the non-smooth case and $\widetilde{O}(N\epsilon^{-2/3}+\sqrt{N}\epsilon^{-1})$ in the $O(1/\epsilon)$-smooth case. Our methods consist of a recently proposed ball optimization oracle acceleration algorithm (which we refine) and a careful implementation of said oracle for the softmax function. We also prove an oracle complexity lower bound scaling as $\Omega(N\epsilon^{-2/3})$, showing that our dependence on N is optimal up to polylogarithmic factors.

Keywords: Convex optimization, Min-max problems, Monteiro-Svaiter acceleration, Ball optimization oracle, Stochastic first-order methods.

1. Introduction

Consider the problem of approximately minimizing the maximum of N convex functions: given $f_1,\ldots,f_N$ such that for every $i\in[N]$ the function $f_i:\mathbb{R}^d\to\mathbb{R}$ is convex, Lipschitz and possibly smooth, and a target accuracy ϵ,

$$\text{find a point } x \text{ such that } F_{\max}(x) - \inf_{x_\star\in\mathbb{R}^d} F_{\max}(x_\star) \le \epsilon \text{ where } F_{\max}(x) \coloneqq \max_{i\in[N]} f_i(x)\,. \quad (1)$$

Problems of this form play significant roles in optimization and machine learning. The maximum of N functions is a canonical example of structured non-smoothness and several works develop methods for exploiting it [31, 30, 36, 9, 12]. The special case where the f_i's are linear functions is particularly important for machine learning, since it is equivalent to hard-margin SVM training (with f_i representing the negative margin on the ith example) [38, 13, 21]. Going beyond the linear case, Shalev-Shwartz and Wexler [36] argue that minimizing the maximum classification loss can have advantageous effects on training speed and generalization in the presence of rare informative examples. Moreover, minimizing the worst-case objective is the basic paradigm of robust optimization [4, 27]. In particular, since $F_{\max}(x) = \max_{p\in\Delta^N}\sum_{i\in[N]} p_i f_i(x)$ the problem corresponds to an extreme case of distributionally robust optimization [5] with an uncertainty set that encompasses the entire probability simplex Δ^N.

© 2021 Y. Carmon, A. Jambulapati, Y. Jin & A. Sidford.

The goal of this paper is to characterize the complexity of this fundamental problem. We are particularly interested in the regime where the number of data points N and the problem dimension d are large compared to the desired level of accuracy $1/\epsilon$, as is common in modern machine learning. Consequently, we focus on dimension-independent first-order methods (i.e., methods which only rely on access to $f_i(x)$ and a (sub)gradient $\nabla f_i(x)$ as opposed to higher-order derivatives), and report complexity in terms of the number of function/gradient evaluations required to solve the problem.

1.1. Related work

To put our new complexity bounds in context, we first review the prior art in solving the problem (1) with first-order methods. For simplicity of presentation, throughout the introduction we assume each f_i is 1-Lipschitz and that $F_{\max}$ has a global minimizer $x_\star$ with (Euclidean) norm at most 1.

The simplest approach to solving the problem (1) is the subgradient method [33]. This method finds an ϵ-accurate solution in $O(\epsilon^{-2})$ iterations, with each step computing a subgradient of $F_{\max}$, which in turn requires evaluation of all N function values and a single gradient. Consequently, the complexity of this method is $O(N\epsilon^{-2})$. We are unaware of prior work obtaining improved complexity without further assumptions.[1]

However, even a weak bound on smoothness helps: if each f_i has $O(1/\epsilon)$-Lipschitz gradient, then it is possible to minimize $F_{\max}$ to accuracy ϵ with complexity $\widetilde{O}(N\epsilon^{-1})$ [31].[2] This result relies on the so-called "softmax" approximation of the maximum,

$$F_{\mathrm{smax},\epsilon}(x) := \epsilon' \log\left(\sum_{i\in[N]} e^{f_i(x)/\epsilon'}\right), \quad \text{where } \epsilon' = \frac{\epsilon}{2\log N}. \tag{2}$$

It is straightforward to show that $|F_{\mathrm{smax},\epsilon}(x) - F_{\max}(x)| \le \frac{\epsilon}{2}$ for all $x \in \mathbb{R}^d$, and that $\nabla F_{\mathrm{smax},\epsilon}$ is $\widetilde{O}(1/\epsilon)$-Lipschitz if ∇f_i is $O(1/\epsilon)$-Lipschitz for every i. Therefore, Nesterov's accelerated gradient descent [31] finds a minimizer of $F_{\mathrm{smax},\epsilon}$ to accuracy $\frac{\epsilon}{2}$ in $\widetilde{O}(\sqrt{1/\epsilon}/\sqrt{\epsilon})$ iterations, with each iteration requiring N evaluations of f_i and ∇f_i to compute $\nabla F_{\mathrm{smax},\epsilon}$, yielding the claimed bound. The assumption that ∇f_i is $O(1/\epsilon)$-Lipschitz is fairly weak; see Appendix A.1 for additional discussion.

Given more smoothness, further improvement is possible. Nesterov [33, Section 2.3.1] shows that it suffices to solve $O(\sqrt{L_g/\epsilon})$ linearized subproblems of the form $\min_{x\in\mathbb{R}^d} \max_{i\in[N]} \{f_i(y_t) + (\nabla f_i(y_t))^\top (x - y_t) + \frac{L_g}{2}\|x - y_t\|^2\}$. This yields a query complexity upper bound of $O(N\sqrt{L_g/\epsilon})$, Though the complexity of solving each subproblem is not immediately clear, in Appendix A.3 we explain how a first-order method [10] solves the subproblem to sufficient precision. Additional schemes for solving (1) in the special case of linear functions (i.e., $L_g = 0$) are discussed in Appendix A.2.

A powerful technique for solving optimization problems with a large number N of component functions is sampling components in order to compute cheap unbiased gradient estimates. However, both $F_{\max}$ and $F_{\mathrm{smax},\epsilon}$ are not given as linear combinations of the f_i's. Consequently, it is not clear how to efficiently compute unbiased estimators for their gradients. Several works address this by

1. The center of gravity method [24, 35] yields a query complexity $O(Nd\log(1/\epsilon))$ which is an improvement only for sufficiently small problem dimension d.
2. Throughout the paper, the $\widetilde{O}(\cdot)$ and $\widetilde{\Omega}(\cdot)$ hide polylogarithmic factors.

Smoothness	Method	Upper bound	Lower bound
None ($L_g = \infty$)	Subgradient method	$N\epsilon^{-2}$	$N\epsilon^{-2/3} + \epsilon^{-2}$
	Ours	$N\epsilon^{-2/3} + \epsilon^{-8/3}$	
Weak ($L_g \approx 1/\epsilon$)	AGD on softmax	$N\epsilon^{-1}$	$N\epsilon^{-2/3} + \sqrt{N}\epsilon^{-1}$
	Ours	$N\epsilon^{-2/3} + \sqrt{N}\epsilon^{-1}$	
Strong ($L_g \ll 1/\epsilon$)	AGD on linearization*	$N\sqrt{L_g\epsilon^{-1}}$	$NL_g^{1/3}\epsilon^{-1/3} + \sqrt{NL_g\epsilon^{-1}}$

Table 1: The complexity of solving the problem (1) in terms of number of (i, x) queries for computing and $f_i(x)$ and $\nabla f_i(x)$. The tables assume each f_i is convex, 1-Lipschitz and (optionally) has L_g-Lipschitz gradient, and that $F_{\max}$ has a minimizer with norm at most 1. The stated rates omit constant and (in the upper bounds) polylogarithmic factors. *For this algorithm only, the computational complexity is not simply d times the query complexity; see Appendix A.3.

considering the saddle point problem

$$\min_{x\in\mathbb{R}^d}\max_{p\in\Delta^N} F_{\mathrm{pd}}(x;p) \coloneqq \sum_{i\in[N]} p_i f_i(x),$$

which is equivalent to minimizing to $F_{\max}$. One can obtain unbiased estimators for $\nabla F_{\mathrm{pd}}(x;p)$, and apply stochastic mirror descent to find its saddle-point [30, 36, 27]. However, all known estimators for $\nabla_p F_{\mathrm{pd}}$ have complexity-variance product $\Omega(N)$. Consequently, the best general guarantees known for such methods are $\widetilde{O}(N\epsilon^{-2})$ iterations and total complexity.[3] Shalev-Shwartz and Wexler [36] analyze a stochastic primal-dual method from an online learning perspective. They show that if the online method producing the primal updates admits a mistake bound (as is the case for learning halfspaces), then the complexity of the approach improves to $\widetilde{O}(N\epsilon^{-1})$. We show that adopting a primal-only perspective and iteratively restricting x to a small ball (i.e., "thinking inside the ball") allows us to make better use of the scalability of stochastic gradient methods.

1.2. Our contributions

To motivate our developments, note that the general complexity guarantees described above all scale linearly with the number of functions N. On the one hand, this is to be expected, as even evaluating the maximum of N numbers requires querying all of them. On the other hand, a linear scaling in N stands in sharp contrast to guarantees for minimizing the *average* of N functions, which are typically sublinear in N. Since good scaling with dataset size is crucial in machine learning, we wish to precisely characterize the number of dataset passes (that is, the coefficient of N) in the complexity of minimizing $F_{\max}$.

Towards that end, we prove an oracle complexity lower bound. The bound shows that any algorithm that operates by repeatedly querying i, x and observing $f_i(x), \nabla f_i(x)$, must make $\Omega(N\epsilon^{-2/3})$ queries in order to solve problem (1) for some convex, 1-Lipschitz problem instance $f_1, \ldots, f_N$ with domain in the unit ball. The same bound continues to hold even when constraining the f_i to have $O(1/\epsilon)$-Lipschitz gradient, and when using high-order derivative oracles. This result further sharpens the contrast to average risk minimization, as it implies $\Omega(\epsilon^{-2/3})$ dataset passes are required in

3. Exact-gradient primal-dual methods such as mirror-prox [28] and dual-extrapolation [32] have complexity guarantees scaling as $\widetilde{O}(N\epsilon^{-1})$ under the stronger smoothness assumption $L_g = O(1)$ [cf. 8, Section 5.2.4].

the worst case. However, it also suggests the potential for significant improvement over existing algorithms and their complexity bounds.

We realize this potential with new algorithms whose leading complexity term in N matches our lower bound up to polylogarithmic factors. In the non-smooth case, our approach solves (1) with complexity $\widetilde{O}(N\epsilon^{-2/3} + \epsilon^{-8/3})$, dominating prior guarantees for $N = \widetilde{\Omega}(\epsilon^{-2/3})$. For $O(1/\epsilon)$-Lipschitz gradient functions, we obtain the stronger rate $\widetilde{O}(N\epsilon^{-2/3} + \sqrt{N}\epsilon^{-1})$, which dominates prior guarantees for $N = \widetilde{\Omega}(1)$. At the core of these algorithms is a technique for accelerated optimization given a ball optimization oracle [12]; we make several improvements to this technique, which may be of independent interest.

Table 1 summarizes our results and their comparison to prior art. In addition to the results described above, the table also contains lower bounds on sublinear terms in N (that follow from standard arguments), as well as a lower bound for the smooth regime where $L_g = o(1/\epsilon)$. In this regime there exists a gap between the linear terms in the upper and lower bounds.

1.3. Overview of techniques

Our algorithms rely on a new technique introduced by Carmon et al. [12] for acceleration with a ball optimization oracle (BOO). For any $r > 0$ and $F : \mathbb{R}^d \to \mathbb{R}$, a BOO of radius r takes in a query point $\bar{x} \in \mathbb{R}^d$ and returns an (approximate) minimizer of F in a ball of radius r around $\bar{x}$. The technique, which is a variant of Monteiro-Svaiter acceleration [26, 17, 7, 9], minimizes F to ϵ accuracy using $\widetilde{O}((1/r)^{2/3})$ oracle calls (with $\mathsf{poly}(\log(1/\epsilon))$ factors hidden). Carmon et al. [11] apply their technique to the special case of (1) with linear losses (see also Appendix A.2), showing that the log-sum-exp function is quasi-self-concordant and implementing a BOO of radius $r = \widetilde{\Theta}(\epsilon)$ using $\widetilde{O}(1)$ linear system solves. However, this approach does not extend to general f_i because quasi-self-concordance no longer holds for $F_{\mathrm{smax},\epsilon}$, which might not even be differentiable.

The main technical insight of our paper is that it is possible to efficiently implement a BOO of radius $r_\epsilon = \widetilde{\Theta}(\epsilon)$ for $F_{\mathrm{smax},\epsilon}$ using stochastic first-order methods. More precisely, for any $\bar{x} \in \mathbb{R}^d$ we can minimize $F_{\mathrm{smax},\epsilon}$ in a ball of radius r_ϵ around $\bar{x}$ to any $\mathsf{poly}(\epsilon)$ accuracy with precisely N function evaluations and $\mathsf{poly}(1/\epsilon)$ (sub-)gradient evaluations. Using BOO acceleration, this immediately implies an $\widetilde{O}(N\epsilon^{-2/3} + \mathsf{poly}(1/\epsilon))$ complexity bound exhibiting optimal N dependence.

To implement the BOO for $F_{\mathrm{smax},\epsilon}$, we consider instead the "exponentiated softmax" function

$$\Gamma_\epsilon(x) = \epsilon' \cdot \exp\left(\frac{F_{\mathrm{smax},\epsilon}(x) - F_{\mathrm{smax},\epsilon}(\bar{x})}{\epsilon'}\right) = \sum_{i\in[N]} p_i \epsilon' \cdot e^{\frac{f_i(x) - f_i(\bar{x})}{\epsilon'}} \quad \text{where } p_i = \frac{e^{f_i(\bar{x})/\epsilon'}}{\sum_{j\in[N]} e^{f_i(\bar{x})/\epsilon'}},$$

and $\epsilon' = \epsilon/(2\log N)$ as in eq. (2). Note that Γ_ϵ is a monotonically increasing transformation of $F_{\mathrm{smax},\epsilon}$, and is therefore convex with the same minimizer as $F_{\mathrm{smax},\epsilon}$. Moreover, it is a (weighted) finite sum, and consequently amenable to stochastic gradient methods. It remains to verify that the functions $\xi_i(x) = \epsilon' \cdot e^{(f_i(x) - f_i(\bar{x}))/\epsilon'}$ are well-behaved, which might look difficult since exponentials are notoriously unstable. However, our choice of r and Lipschitz continuity of f_i implies that $e^{(f_i(x) - f_i(\bar{x}))/\epsilon} = \Theta(1)$ inside the ball, and consequently ξ_i is indeed well-behaved, with Lipschitz constant $O(1)$. We thus minimize Γ_ϵ (and hence $F_{\mathrm{smax},\epsilon}$) with stochastic gradient descent [20], sampling i from p. Moreover, if ∇f_i are Lipschitz, then $\nabla \xi_i$ are also Lipschitz, and we apply an accelerated variance reduction method [1] for better efficiency.

To complete the analysis of our methods it remains to determine how accurately we need to solve each ball subproblem. Unfortunately, the analysis of [12] makes fairly stringent accuracy

requirements, and also requires $\nabla F_{\mathrm{smax},\epsilon}$ to have a finite Lipschitz constant. To obtain tighter guarantees, we significantly rework the analysis in [12], modifying the algorithm to make it applicable without any differentiablility requirements. Our improved analysis takes into account the fact that the acceleration scheme only requires ball minimization with strong ℓ_2 regularization, which further improves the oracle implementation complexity.

Our lower bound follows from a variation on the classical "chain constructions" in optimization lower bounds [29, 39, 18, 14], where in order to make a unit of progress on our constructed function, any algorithm must (with constant probability) make $\Omega(N)$ queries in order to discover a single new link in the chain. We build a chain of length $\Omega(\epsilon^{-2/3})$ for which querying any ϵ minimizer of $F_{\max}$ requires discovering the entire chain, giving the $\Omega(N\epsilon^{-2/3})$ complexity lower bound. To prove this result for arbitrary randomized algorithms, we randomize both the order of the functions and the rotation of the domain.

Paper outline. Section 2 provides some additional preliminaries and notation. Section 3 gives our improved derivation of the BOO acceleration method of [12], and Section 4 develops a BOO for $F_{\mathrm{smax},\epsilon}$, culminating in our upper complexity bounds for the problem (1), stated in Theorem 6. Section 5 gives our lower bounds with the main result stated in Theorem 10. Section 6 includes some comments on our results and potential future work.

2. Preliminaries

General notation. Throughout, $\|\cdot\|$ denotes the Euclidean norm. We write $\mathbb{B}_r(z)$ for the Euclidean ball of radius r centered at z, and $\mathbb{B}_r^d(z)$ when emphasizing that the ball is d-dimensional. We use L_f to denote a function Lipschitz constant and L_g to denote a gradient Lipschitz constant; we say that f is L_g-smooth if it has L_g-Lipschitz gradient. To disambiguate between sequence and coordinate indices, in Section 5 we denote the former with normal subscript and the latter with bracketed subscript, i.e., $x_{[i]}$ is the ith coordinate of x and x_k is the kth element in the sequence $x_1, x_2, \ldots$. We also write $v_{[\le i]}$ to denote a copy of v with coordinate $i+1, i+2, \ldots$ set to zero. We use $a \wedge b \coloneqq \min\{a, b\}$ to abbreviate binary minimization. We write the binary indicator of event A as $\mathbb{I}\{A\}$.

Complexity model. We mainly measure complexity through the number individual function and gradient evaluations required to solve the problem (1). We write $\mathcal{T}_f$ for the cost of evaluating $f_i(x)$ for a single i and x, and similarly write $\mathcal{T}_g$ for the cost of evaluating $\nabla f_i(x)$. Assuming $\mathcal{T}_f, \mathcal{T}_g = \Omega(d)$, our evaluation complexity upper bounds translate directly to runtime upper bounds.

Proximal operators. For any function f and regularization parameter $\lambda \ge 0$, we define the standard proximal mapping $\mathrm{prox}_\lambda^f(\bar{x}) \coloneqq \arg\min_{x\in\mathbb{R}^d}\{f(x) + \frac{\lambda}{2}\|x-\bar{x}\|^2\}$. We also define the ball constrained proximal mapping $\mathrm{bprox}_{\lambda,r}^f(\bar{x}) \coloneqq \arg\min_{x\in\mathbb{B}_r(\bar{x})}\{f(x) + \frac{\lambda}{2}\|x-\bar{x}\|^2\}$. Finally, we define the notion of an approximate oracle for $\mathrm{bprox}_{\lambda,r}^f$, which plays a key role in our analysis.

Definition 1 (BROO) *We say that a mapping $\mathcal{O}_{\lambda,\delta}(\cdot)$ is a Ball Regularized Optimization Oracle of radius r (r-BROO) for f, if for every query point $\bar{x}$, regularization parameter λ and desired accuracy δ, it return $\tilde{x} = \mathcal{O}_{\lambda,\delta}(\bar{x})$ satisfying*

$$f(\tilde{x}) + \frac{\lambda}{2}\|\tilde{x}-\bar{x}\|^2 \le \min_{x\in\mathbb{B}_r(\bar{x})}\left\{f(x) + \frac{\lambda}{2}\|x-\bar{x}\|^2\right\} + \frac{\lambda}{2}\delta^2. \tag{3}$$

Note that when f is convex, the strong convexity of $f(x) + \frac{\lambda}{2}\|x - \bar{x}\|^2$ and the approximation requirement (3) guarantee that $\|\mathcal{O}_{\lambda,\delta}(\bar{x}) - \text{bprox}^f_{\lambda,r}(\bar{x})\| \le \delta$.

3. BROO acceleration

In this section, we describe a variant of the ball optimization acceleration scheme of Carmon et al. [12], given as Algorithm 1. Both methods follow the template of Monteiro-Svaiter acceleration [26], but our algorithm improves on [12] in two ways. First, it accesses the objective strictly through the ball oracle, while [12] also uses gradient computations. Second, our algorithm requires an oracle that solves *regularized* ball optimization problems, which are easier to implement.[4]

As a consequence of these differences, our accelerated algorithm's guarantee does not require any smoothness of the objective function. Moreover, our setup allows for far less accurate solutions to the ball optimization subproblems: Carmon et al. [12] require $\delta = O(\frac{\epsilon}{L_g R})$ while we only require $\delta = O(\frac{\epsilon}{\lambda R})$. While our requirement becomes stricter as the regularizer λ grows, it also becomes easier to fulfill since the ball optimization problem becomes more strongly convex and hence easier to solve. Our relaxed accuracy requirement ultimately translates to an improved ϵ^{-1} dependence in the sublinear-in-N term in our upper bound.

With the key innovations of Algorithm 1 explained, we now formally state its convergence guarantee; we defer the proof to Appendix B.

Theorem 2 *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and L_f-Lipschitz, and let $z \in \mathbb{R}^d$. For any domain bound $R > 0$, ball radius $r \in (0, R]$, accuracy level $\epsilon > 0$, and initial point $x_0 \in \mathbb{R}^d$, Algorithm 1 returns a point $x \in \mathbb{R}^d$ satisfying $f(x) - \min_{z \in \mathbb{B}_R(x_0)} f(z) \le \epsilon$ using at most*

$$T = O\left(\left(\frac{R}{r}\right)^{2/3} \log\left(\frac{[f(x_0) - \min_{z \in \mathbb{B}_R(x_0)} f(z)]R}{\epsilon r}\right) \log\left(\frac{L_f R^2}{\epsilon r}\right)\right)$$

queries to an r-BROO. Moreover, the BROO query parameters $(\lambda_{(1)}, \delta_{(1)}), \ldots, (\lambda_{(T)}, \delta_{(T)})$ satisfy

1. $\Omega(\frac{\epsilon}{rR}) \le \lambda_{(i)} \le O(\frac{L_f}{r})$ *and* $\delta_{(i)} \ge \Omega(\frac{\epsilon}{\lambda_{(i)} R})$ *for all* $i \in [T]$.
2. $\sum_{i \in [T]} \frac{1}{\sqrt{\lambda_{(i)}}} \le O(\frac{R}{\sqrt{\epsilon}} \log \frac{L_f R^2}{\epsilon r})$.

We remark that Theorem 2 requires a bound on the Lipschitz constant of f solely to bound the complexity of the bisection procedure for finding $\{\lambda_t\}$.

4. BROO implementation

In this section, we develop efficient BROO implementations for $F_{\text{smax},\epsilon}$, the softmax approximation of $F_{\max}$ (2). In Section 4.1 we develop our main analytical tool in the form of an "exponentiated softmax" function approximating $F_{\text{smax},\epsilon}$ and facilitating efficient stochastic gradient estimation. We then minimize the exponentiated softmax with standard tools from stochastic convex optimization. In Section 4.2 we give a BROO implementation for the non-smooth case using restarted

4. We note that λ in our notation corresponds to $1/\lambda$ in the notation of [12].

Algorithm 1: BROO acceleration

Input: Initial $x_0 \in \mathbb{R}^d$, Lipschitz and distance bounds L_f, R, r, accuracy ϵ, BROO $\mathcal{O}_{\lambda,\delta}(\cdot)$

Output: x_{ret} such that $f(x_{\text{ret}}) - \arg\min_{z \in \mathbb{B}_R(x_0)} f(z) \le \epsilon$

Let $v_0 = x_0$, $A_0 = 0$

for $t = 0, 1, 2, \ldots$ **do**

  $\lambda_{t+1} = \lambda$-BISECTION$(x_t, v_t, A_t, \lambda_{\max} = \frac{2L_f}{r}, \lambda_{\min} = \frac{\epsilon}{6rR})$

    ▷ Finds λ_{t+1} such that $x_{t+1} \approx \text{prox}^f_{\lambda_{t+1}}(y_t)$ and either $\|x_{t+1} - y_t\| \approx r$ or x_{t+1} is ϵ-optimal

  $a_{t+1} = \frac{1}{2\lambda_{t+1}}(1 + \sqrt{1 + 4\lambda_{t+1}A_t})$ and $A_{t+1} = A_t + a_{t+1}$ ▷ $A_{t+1} = a^2_{t+1}\lambda_{t+1}$

  $y_t = \frac{A_t}{A_{t+1}}x_t + \frac{a_{t+1}}{A_{t+1}}v_t$

  $x_{t+1} = \mathcal{O}_{\lambda_{t+1},\delta_{t+1}}(y_t)$, where $\delta_{t+1} = \frac{\epsilon}{12\lambda_{t+1}R}$

  $v_{t+1} = \arg\min_{v \in \mathbb{B}_R(x_0)} \left\{ a_{t+1}\lambda_{t+1} \langle y_t - x_{t+1}, v \rangle + \frac{1}{2}\|v - v_t\|^2 \right\}$

  if $A_{t+1} \ge \frac{R^2}{\epsilon}$, $\lambda_{t+1} \le \frac{\epsilon}{3rR}$, $\|x_{t+1} - v_{t+1}\| > 2R$, **or** $A_{t+1} < \exp\left(\frac{r^{2/3}}{R^{2/3}}(t-1)\right) A_1$ **then**

    return $x_{\text{ret}} \in \arg\min_{x \in \{x_0, x_1, \ldots, x_{t+1}\}} f(x)$

function λ-BISECTION$(x, v, A, \lambda_{\max}, \lambda_{\min})$

  For all λ', let $y_{\lambda'} \coloneqq \alpha_{2A\lambda'} \cdot x + (1 - \alpha_{2A\lambda'}) \cdot v$, where $\alpha_\tau \coloneqq \frac{\tau}{1+\tau+\sqrt{1+2\tau}}$

  Define $\Delta(\lambda) \coloneqq \|\mathcal{O}_{\lambda,\frac{r}{17}}(y_\lambda) - y_\lambda\|$ ▷ approximation of $\widehat{\Delta}(\lambda) \coloneqq \|\text{bprox}^f_{\lambda,r}(y_\lambda) - y_\lambda\|$

  Let $\lambda = \lambda_{\max}$

  while $\lambda \ge \lambda_{\min}$ **and** $\Delta(\lambda) \le \frac{13r}{16}$ **do** $\lambda \leftarrow \lambda/2$ ▷ terminates in $O(\log \frac{\lambda_{\max}}{\lambda_{\min}})$ steps

  if $\lambda \le \lambda_{\min}$ **then return** 2λ ▷ happens only if $\text{bprox}^f_{2\lambda,r}(y_{2\lambda})$ is $O(\epsilon)$-optimal for small $\lambda_{\min}$

  Let $\lambda_u = 2\lambda$, $\lambda_\ell = \lambda$ and $\lambda_m = \sqrt{\lambda_u\lambda_\ell}$

  if $\Delta(\lambda_\ell) \le \frac{15r}{16}$ **then return** λ_ℓ ▷ happens only if $\Delta(\lambda_\ell) \in [\frac{13r}{16}, \frac{15r}{16}]$

  while $\Delta(\lambda_m) \notin [\frac{13r}{16}, \frac{15r}{16}]$ **and** $\log_2 \frac{\lambda_u}{\lambda_\ell} \ge \frac{r}{8(R + L_f/\lambda_\ell)}$ **do**

    if $\Delta(\lambda_m) < \frac{13r}{16}$ **then** $\lambda_u = \lambda_m$ **else** $\lambda_\ell = \lambda_m$

    $\lambda_m = \sqrt{\lambda_u\lambda_\ell}$

  return λ_m ▷ the while loop terminates in $O\big(\log\big(\frac{R}{r} + \frac{L_f}{\lambda_{\min} r}\big)\big)$ steps

SGD [20]. In Section 4.3 we instead apply an accelerated variance reduction method (Katyusha [1]) that offers improved performance when the f_i are even slightly smooth. Finally, in Section 4.4 we combine our BROO implementations with Algorithm 1 and its guarantees to obtain our main results: new convergence guarantees for minimizing $F_{\max}$. We defer proofs to Appendix C.

4.1. Exponentiating a softmax

Recall that $\epsilon' = \epsilon/(2\log N)$ and that (for nominal accuracy ϵ) the softmax function $F_{\text{smax},\epsilon}(x) = \epsilon' \log\left(\sum_{i \in [N]} e^{f_i(x)/\epsilon'}\right)$ approximates $F_{\max}$ to within $\epsilon/2$ additive error. The key challenge in designing an efficient stochastic method for minimizing $F_{\text{smax},\epsilon}$ is a lack of cheap unbiased gradient

estimators. Specifically, we have $\nabla F_{\mathrm{smax},\epsilon}(x) = \sum_{i\in[N]} p_i(x)\nabla f_i(x)$, where

$$p_i(x) = \frac{e^{f_i(x)/\epsilon'}}{\sum_{j\in[N]} e^{f_j(x)/\epsilon'}}. \tag{4}$$

Given access to $p(x)$, we could easily obtain an unbiased estimator for $\nabla F_{\mathrm{smax},\epsilon}(x)$ by sampling $i \sim p(x)$ and outputting $\nabla f_i(x)$. However, computing $p(x)$ itself requires evaluating all N functions, making it basically as costly as computing $\nabla F_{\mathrm{smax},\epsilon}$ exactly.

This difficulty, however, is greatly relieved when we operate in a small ball of radius $r_\epsilon = \epsilon'/L_f$ centered at some point $\bar{x}$. To see why, note that for every i and every $x \in \mathbb{B}_{r_\epsilon}(\bar{x})$, Lipschitz continuity of f_i implies $|f_i(x)/\epsilon' - f_i(\bar{x})/\epsilon'| \le L_f r_\epsilon/\epsilon' = 1$. Consequently, $p(\bar{x})$ is a multiplicative approximation for $p(x)$ throughout the ball, satisfying $e^{-2}p_i(\bar{x}) \le p_i(x) \le e^2 p(\bar{x})$ for all $x \in \mathbb{B}_{r_\epsilon}(\bar{x})$. Our high-level strategy is thus: perform a full data pass *once* to compute $p(\bar{x})$, and then rely on the stability of $p(x)$ within $\mathbb{B}_{r_\epsilon}(\bar{x})$ to efficiently estimate gradients by sampling from $p(\bar{x})$. However, simply sampling $i \sim p(\bar{x})$ and returning $\nabla f_i(x)$ is not enough, because it leads to a biased estimator of $\nabla F_{\mathrm{smax},\epsilon}(x)$. Instead, we define below a surrogate function "exponentiating the softmax" that closely approximates $F_{\mathrm{smax},\epsilon}$ and for which $e^{(f_i(x)-f_i(\bar{x}))/\epsilon'}\nabla f_i(x)$ is an unbiased gradient estimator when $i \sim p(\bar{x})$.[5]

To precisely define the surrogate "exponentiated softmax" function, we require some additional notation. Fixing a ball center $\bar{x}$ and regularization parameter λ, let

$$f_i^\lambda(x) \coloneqq f_i(x) + \frac{\lambda}{2}\|x-\bar{x}\|^2 \text{ and } F_{\mathrm{smax},\epsilon}^\lambda(x) \coloneqq F_{\mathrm{smax},\epsilon}(x) + \frac{\lambda}{2}\|x-\bar{x}\|^2 = \epsilon' \log\left(\sum_{i\in[N]} e^{f_i^\lambda(x)/\epsilon'}\right)$$

be the regularized counterparts of f_i and $F_{\mathrm{smax},\epsilon}$, respectively. Then, we define the exponentiated softmax as

$$\Gamma_{\epsilon,\lambda}(x) = \epsilon' \cdot \exp\left(\frac{F_{\mathrm{smax},\epsilon}^\lambda(x) - F_{\mathrm{smax},\epsilon}^\lambda(\bar{x})}{\epsilon'}\right) = \sum_{i\in[N]} p_i(\bar{x})\gamma_i(x) \text{ where } \gamma_i(x) \coloneqq \epsilon' e^{\frac{f_i^\lambda(x)-f_i^\lambda(\bar{x})}{\epsilon'}}. \tag{5}$$

Clearly, $\Gamma_{\epsilon,\lambda}$ is a finite sum objective (weighted by $p(\bar{x})$), making stochastic first-order methods applicable. Moreover, as the following lemma shows, when the ball radius r and λ are not too large, $\Gamma_{\epsilon,\lambda}$ closely approximates $F_{\mathrm{smax},\epsilon}^\lambda$ and is as regular as $F_{\mathrm{smax},\epsilon}^\lambda$ up to a constant.

Lemma 3 *Let $f_1, \cdots, f_N$ each be L_f-Lipschitz and L_g-smooth gradients. For any $c > 0$, $r \le c\epsilon'/L_f$, and $\lambda \le cL_f/r$ let $C = (1+c+c^2)e^{c+c^2/2}$. The exponentiated softmax $\Gamma_{\epsilon,\lambda}$ satisfies the following properties for any $\bar{x} \in \mathbb{R}^d$.*

1. *$F_{\mathrm{smax},\epsilon}^\lambda(x)$ and $\Gamma_{\epsilon,\lambda}$ have the same minimizer $x_\star$ in $\mathbb{B}_r(\bar{x})$. Moreover, for every $x \in \mathbb{B}_r(\bar{x})$,*

$$F_{\mathrm{smax},\epsilon}^\lambda(x) - F_{\mathrm{smax},\epsilon}^\lambda(x_\star) \le C(\Gamma_{\epsilon,\lambda}(x) - \Gamma_{\epsilon,\lambda}(x_\star)).$$

2. *Restricted to $\mathbb{B}_r(\bar{x})$, each function γ_i defined in (5) is CL_f-Lipschitz, $C^{-1}\lambda$ strongly convex, and $C(L_g + \lambda + L_f^2/\epsilon')$-smooth.*

The proof of Lemma 3 follows from a straightforward calculation, and we defer it to Appendix C.1.

5. We remark that $g_i(x) = e^{(f_i(x)-f_i(\bar{x}))/\epsilon'}\nabla f_i(x)$ is also nearly unbiased for $F_{\mathrm{smax},\epsilon}$ in the sense that $\mathbb{E}g_i(x) = Z(x)\nabla F_{\mathrm{smax},\epsilon}(x)$ for some $Z(x)$ that is close to 1 when inside $\mathbb{B}_{r_\epsilon}(\bar{x})$. Estimators of this form suffice for SGD, but are less amenable to variance reduction.

4.2. The non-smooth case: SGD implementation

To take advantage of the strong convexity of of $\Gamma_{\epsilon,\lambda}$ we use the restarted SGD variant of Hazan and Kale [20], which finds an ε-suboptimal point of a G-Lipschitz and μ-strongly convex function with $\widetilde{O}(G^2/(\mu\varepsilon))$ iterations (with high probability). To estimate the stochastic gradients, we sample $i \sim p(\bar{x})$ and output $\nabla\gamma_i(x)$; this takes $O(\mathcal{T}_g + \mathcal{T}_f)$ time per stochastic gradient, plus $O(N\mathcal{T}_f)$ preprocessing time to compute $p(\bar{x})$. We provide pseudocode for the algorithm in Appendix C.2, where we also prove the following complexity bound.

Corollary 4 *Let $f_1, f_2, \cdots, f_N$ be L_f Lipschitz, let $\sigma \in (0,1)$, $\epsilon, \delta > 0$ and $r_\epsilon = \epsilon/(2\log N \cdot L_f)$. For any $\bar{x} \in \mathbb{R}^d$ and $\lambda \leq O(L_f/r_\epsilon)$, with probability at least $1-\sigma$, Algorithm 2 outputs a valid r_ϵ-BROO response for $F_{\mathrm{smax},\epsilon}$ to query $\bar{x}$ with regularization λ and accuracy δ, and has cost*

$$O\left(\mathcal{T}_f N + (\mathcal{T}_g + \mathcal{T}_f)\frac{L_f^2}{\lambda^2\delta^2}\log\left(\frac{\log(L_f/\lambda\delta)}{\sigma}\right)\right). \tag{6}$$

4.3. The (slightly) smooth case: accelerated variance reduction implementation

If we further assume smoothness of $f_1, \ldots, f_N$, we can use stochastic variance reduction to obtain an improved runtime. With these methods, we estimate the gradient of $\Gamma_{\epsilon,\lambda}$ as $\nabla\Gamma_{\epsilon,\lambda}(x') + \nabla\gamma_i(x) - \nabla\gamma_i(x')$, where $i \sim p(\bar{x})$ and x' is a reference point which we recompute $\widetilde{O}(1)$ times. Here, the $O(N\mathcal{T}_f)$ cost of computing $p(\bar{x})$ is essentially free compared to the cost $\widetilde{O}(N\mathcal{T}_g)$ of computing the exact gradients of $\Gamma_{\epsilon,\lambda}$ at the reference point. We again take advantage of the regularization-induced λ-strong-convexity a variant of the Katyusha method of Allen-Zhu [1]. This results in the following complexity guarantee; see Appendix C.3 for a proof.

Corollary 5 *Let $f_1, \cdots, f_N$ be L_f-Lipschitz and L_g-smooth, let $\sigma \in (0,1)$, $\epsilon, \delta > 0$, $\epsilon' = \epsilon/(2\log N)$ and $r_\epsilon = \epsilon' / L_f$. For any $\bar{x} \in \mathbb{R}^d$ and $\lambda \leq O(L_f/r_\epsilon)$, with probability at least $1-\sigma$, Katyusha1 [1] outputs a valid r_ϵ-BROO response to query $\bar{x}$ with regularization λ and accuracy δ, and has computational cost*

$$O\left((\mathcal{T}_f + \mathcal{T}_g)\left(N + \frac{\sqrt{N}\left(L_f + \sqrt{\epsilon' L_g}\right)}{\sqrt{\lambda\epsilon'}}\right)\log\left(\frac{L_f r_\epsilon}{\lambda\delta^2\sigma}\right)\right). \tag{7}$$

4.4. Main result

With our oracle implementations in hand, we are ready to state our main result.

Theorem 6 *Let $f_1, f_2, \ldots, f_N$ be L_f-Lipschitz, let $x_\star$ be a minimizer of $F_{\max}(x) = \max_{i\in[N]} f_i(x)$ and assume $\|x_0 - x_\star\| \leq R$ for a given initial point x_0 and some $R > 0$. For any $\epsilon > 0$, Algorithm 1 with the BROO implementation for $F_{\mathrm{smax},\epsilon}$ in Algorithm 2 solves the problem (1) with probability at least $\frac{99}{100}$ and has computational cost*

$$O\left(\left(\frac{L_f R \log N}{\epsilon}\right)^{2/3}\left(\mathcal{T}_f N + \left(\frac{L_f R}{\epsilon}\right)^2 \cdot (\mathcal{T}_f + \mathcal{T}_g)\log K\right)\log^2 K\right), \tag{8}$$

where $K \coloneqq L_f R \epsilon^{-1} \log N$. *If moreover* $f_1, f_2, \ldots, f_N$ *are each* L_g*-smooth, then Algorithm 1 with a BROO implementation for* $F_{\mathrm{smax},\epsilon}$ *using Kayusha1 solves* (1) *with probability* $\geq \frac{99}{100}$ *and has cost*

$$O\left((\mathcal{T}_f + \mathcal{T}_g)\left(\left(\frac{L_f R \log N}{\epsilon}\right)^{2/3} N + \left(\frac{L_f R \sqrt{\log N}}{\epsilon} + \sqrt{\frac{L_g R^2}{\epsilon}}\right)\sqrt{N}\right)\log^3 K\right).$$

The proof of Theorem 6, which we provide in Appendix C.4, follows straightforwardly from Theorem 2 and Corollaries 4 and 5. When applying Corollary 4 with $\delta = \Omega(\frac{\epsilon}{\lambda R})$ the dependence of the complexity on λ cancels, and we get that each oracle call costs $\widetilde{O}(N\mathcal{T}_f + L_f^2 R^2 \epsilon^{-2}(\mathcal{T}_f + \mathcal{T}_g))$. The complexity bound then follows from multiplying the per-call cost with the bound $\widetilde{O}((R/r_\epsilon)^{-2/3})$ that Theorem 2 provides on the total number of oracle calls. When applying Corollary 5 we obtain an oracle implementation cost of $\widetilde{O}(N(\mathcal{T}_f + \mathcal{T}_g) + \lambda^{-1/2}\sqrt{N}\sqrt{L_f^2\epsilon^{-1} + L_g}(\mathcal{T}_f + \mathcal{T}_g))$. The complexity bound again follows by multiplying the per-call cost again with the total number of calls, except that to bound the contribution of $\sqrt{N}$ term we invoke the the guarantee $\sum_i \lambda_{(i)}^{-1/2} \leq \widetilde{O}(R\epsilon^{-1/2})$ in Theorem 2 to a tighter bound.

5. Lower bounds

In this section, we prove oracle complexity lower bounds showing that the results of the previous section are order optimal for sufficiently large N and L_g. While our algorithms are first-order methods, our lower bounds remain valid even for other algorithms that use high order derivatives, as is typical for our proof technique.

We begin by providing a formal definition of the oracle-based optimization model we consider (Section 5.1). In Section 5.2, we define an N-element variant for the zero-chain concept, and prove that it allows us to control the progress of any (possibly randomized) algorithm. Then, in Section 5.3 we construct a particular N-element zero-chain for which slow progress implies a large optimality gap. Finally, Section 5.4 ties these results together, giving our lower bound and providing some discussion.

5.1. Optimization protocol

Consider problem instances of the form $(f_i)_{i\in[N]}$, where $f_i : D \to \mathbb{R}$ for some common domain D and all $i \in [N]$. We say that an algorithm operating on $(f_i)_{i\in[N]}$ is an *N-element algorithm* if it uses the following iterative protocol. At iteration t, the algorithm produces a query i_t, x_t, with $i_t \in [N]$ and $x_t \in D$. It then observes the output of a *local oracle* for f_{i_t} at the point x_t, which we denote by $\mathcal{O}^{\mathrm{loc}}_{f_{i_t}}(x_t)$.

Formally, $\mathcal{O}^{\mathrm{loc}}$ can be any mapping that satisfies $\mathcal{O}^{\mathrm{loc}}_f(x) = \mathcal{O}^{\mathrm{loc}}_{\tilde{f}}(x)$ whenever $f(y) = \tilde{f}(y)$ for all y in some open set containing x (subsequently referred to as a "neighborhood" of x). In particular, the first-order oracle used for our upper bounds corresponds to $\mathcal{O}^{\mathrm{loc}}_f(x) = (f(x), \nabla f(x))$ and is valid local oracle. The pth order derivative oracle $\mathcal{O}^{\mathrm{loc}}_f(x) = (f(x), \nabla f(x), \ldots, \nabla^p f)$ is also a valid local oracle. The notion of local oracles is classical in the literature on information-based complexity [29, 18].

The algorithms we consider may be randomized, and we use ζ to denote the algorithm's randomness. Beyond ζ, the query of the algorithm at iteration t may only depend on the information it

observes from the oracle. That is, for any $t \geq 1$, we have

$$i_t, x_t = Q_t\Big(\zeta, \mathcal{O}^{\mathrm{loc}}_{f_{i_1}}(x_1), \ldots, \mathcal{O}^{\mathrm{loc}}_{f_{i_{t-1}}}(x_{t-1})\Big) \tag{9}$$

for some measurable function Q_t.

5.2. Progress control argument

Following well-established methodology [33, 18, 11], instead of directly bounding the sub-optimality of the queries $x_1, \ldots, x_t$ we first bound a surrogate quantity we call *progress*. Informally, the progress is the highest coordinate index that the algorithm managed to "discover" using the oracle responses. Formally, we define the progress of a point x as

$$\mathrm{prog}_\alpha(x) \coloneqq \max\big\{i \geq 1 \;\big|\; |x_{[i]}| > \alpha\big\} \;\; (\text{where } \max \emptyset \coloneqq 0). \tag{10}$$

The parameter α is a significance threshold for declaring a coordinate "discovered;" it allows us to prevent algorithms from trivially discovering coordinates by querying directions at random.

We next define a structural property that facilitates controlling the rate with which $\mathrm{prog}_\alpha(x_t)$ increases. For this definition, we recall that $v_{[\leq l]}$ denotes the vector whose first l coordinates are identical to those of v and the remainder are zero. Recall also that $\mathbb{B}_1^T(0)$ is the unit ball in $\mathbb{R}^T$.

Definition 7 *A sequence $f_1, \ldots, f_N$ of functions $f_i : \mathbb{B}_1^T(0) \to \mathbb{R}$ is called an* α-robust N-element zero-chain *if for all $x \in \mathbb{B}_1^T(0)$, all y in a neighborhood of x, and all $i \in [N]$, we have*

$$\mathrm{prog}_\alpha(x) \leq p \implies f_i(y) = \begin{cases} f_i(y_{[\leq p]}) & i < p+1 \\ f_i(y_{[\leq p+1]}) & i = p+1 \\ f_N(y_{[\leq p]}) & i > p+1. \end{cases} \tag{11}$$

To unpack this definition, consider any first-order algorithm with the following two simplifying properties: (1) the queries $i_1, i_2, \ldots$ are drawn i.i.d. from $\mathrm{Uniform}([N])$ and (2) every query x_t lies in the span of previously observed gradients $\nabla f_{i_1}(x_1), \ldots, \nabla f_{i_{t-1}}(x_{t-1})$ [cf. 33]. The first query of the algorithm must be $x_1 = 0$, and consequently $\mathrm{prog}_\alpha(x_1) = 0$. Definition 7 then implies that $f_2, \ldots, f_N$ are all constant in a neighborhood of x_1, while f_1 depends only on the first coordinate. Therefore, the span of the gradients (and the next query's progress) can only increase to 1 after the algorithm queries $i = 1$ for the first time. With uniformly random index queries, that takes $\Omega(N)$ queries with constant probability. Repeating this argument, we see that every increase of the gradient span (and hence query progress) takes $\Omega(N)$ queries with constant probability, and therefore reaching progress T takes $\widetilde{\Omega}(NT)$ queries with high probability.

To extend this conclusion to general algorithms of the form (9), we perform two types of randomization. First, to handle arbitrary strategies for choosing i_t (as opposed to uniform sampling), we apply a random permutation to $f_1, \ldots, f_N$. Second, to handle arbitrary queries x_t (as opposed to queries in the span of observed gradients), we randomly rotate the coordinate system. This randomization scheme guarantees that no algorithm can materially improve on uniform sampling and span-preserving, as we formally state in the following.

Proposition 8 *Let $\delta, \alpha \in (0,1)$ and let $N, T \in \mathbb{N}$ with $T \leq N/2$. Let $(f_i)_{i\in[N]}$ be an α-robust N-element zero-chain with domain $\mathbb{B}_1^T(0)$. For $d \geq T + \frac{2}{\alpha^2}\log\frac{4NT^2}{\delta}$, draw U uniformly from the*

set of $d \times T$ orthogonal matrices, and draw Π uniformly from the set of permutations of $[N]$. Let $\tilde{f}_i(x) \coloneqq f_{\Pi^{-1}(i)}(U^\top x)$. Let $\{(i_t, x_t)\}_{t\geq 1}$ be the queries of any N-element algorithm operating on $\tilde{f}_1, \ldots, \tilde{f}_N$. Then with probability at least $1-\delta$ we have

$$\mathrm{prog}_\alpha(U^\top x_t) < T \;\; \textit{for all} \;\; t \leq \tfrac{1}{16} N\big(T - \log \tfrac{2}{\delta}\big).$$

See Appendix D.1 for a proof. Our definition of N-element zero-chains and our proof of their progress control property builds on the notion of (single element) zero-chain functions [11]. It is also closely related to probability-p zero-chains [3]; Proposition 8 essentially shows that N-element algorithms interacting with an N-element zero-chain make progress about as slowly as stochastic algorithms interacting with with a probability-N^{-1} zero-chain.

5.3. Hard instance construction

With the progress-control machinery in hand, we proceed to constructing a specific N-element zero-chain that also guarantees a large optimality gap for points with progress smaller than T. Toward that end, we first define the "link function" $\psi_{\alpha,\ell} : \mathbb{R} \to \mathbb{R}_+$ as

$$\psi_{\alpha,\ell}(t) \coloneqq \begin{cases} 0 & |t| \leq \alpha \\ \frac{\ell}{2}(t-\alpha)^2 & \alpha \leq |t| \leq \ell^{-1} + \alpha \\ |t| - \alpha - \frac{1}{2\ell} & \text{otherwise.} \end{cases}$$

Clearly, $\psi_{\alpha,\ell}$ is 1-Lipschitz, ℓ-smooth, and is identically zero for all $|t| \leq \alpha$. We note that $\psi_{\alpha,\ell}$ is the composition of the Huber function [22] with $\max\{0, |t| - \alpha\}$.

Chain constructions of the form $\sum_{i\in[N]} \psi_{\alpha_T,\ell}(x_{[i]} - x_{[i-1]})$ are common in lower bounds for convex optimization [cf. 33, 39]. For our construction, we instead spread the link components across the different elements. Formally, for $i \in [N]$, we define the ith function in the our hard instance as

$$\hat{f}_i^{\{T,N,\ell\}}(x) \coloneqq \begin{cases} \psi_{\alpha_T,\ell}\Big(\frac{x_{[i]} - x_{[i-1]}}{2}\Big) & i \leq T \\ 0 & \text{otherwise} \end{cases} \quad \text{where} \;\; \alpha_T \coloneqq \frac{1}{4T^{3/2}} \text{ and } x_{[0]} \coloneqq \frac{1}{\sqrt{T}}. \tag{12}$$

The following lemma summarizes the properties of our construction. The proof of the lemma is straightforward and we provide it in Appendix D.2

Lemma 9 *For every $T, N \in \mathbb{N}$ and $\ell \geq 0$, such that $T \leq N$, we have that*

1. *The hard instance $(\hat{f}_i^{\{T,N,\ell\}})_{i\in N}$ is an α_T-robust N-element zero-chain.*
2. *The function $\hat{f}_i^{\{T,N,\ell\}}$ is 1-Lipschitz and ℓ-smooth for every $i \in [N]$.*
3. *For $x \in \mathbb{R}^d$ with $\mathrm{prog}_{\alpha_T}(x) < T$, the objective $\hat{F}_{\max}^{\{T,N,\ell\}}(x) = \max_{i\in[N]} \hat{f}_i^{\{T,N,\ell\}}(x)$ satisfies*

$$\hat{F}_{\max}^{\{T,N,\ell\}}(x) - \min_{x_\star \in \mathbb{B}_1(0)} \hat{F}_{\max}^{\{T,N,\ell\}}(x_\star) \geq \psi_{\alpha_T,\ell}\left(\frac{3}{8T^{3/2}}\right) \geq \min\left\{\frac{1}{8T^{3/2}}, \frac{\ell}{32T^3}\right\}.$$

5.4. Lower bound statement

Finally, we combine the results of the previous sections to state our lower bound. In the statement, we use $a \wedge b := \min\{a, b\}$ to abbreviate binary minimization.

Theorem 10 *Let $L_f, L_g, R > 0$, $\epsilon < L_f R \wedge L_g R^2$, $N \in \mathbb{N}$ and $\delta \in (0,1)$. Then, for any (possibly randomized) algorithm there exists an L_f-Lipschitz and L_g-smooth functions $(f_i)_{i\in[n]}$ with domain $\mathbb{B}_R^d(0)$ for $d = O\Big(\Big[\big(\frac{L_f R}{\epsilon}\big)^2 \wedge \big(\frac{L_g R^2}{\epsilon}\big)\Big] \log \frac{N(L_f R \wedge L_g R^2)}{\epsilon}\Big)$ such that with probability at least $\frac{1}{2}$ over the randomness of the algorithm, the first*

$$\Omega\left(N\left[\left(\frac{L_f R}{\epsilon}\right)^{2/3} \wedge \left(\frac{L_g R^2}{\epsilon}\right)^{1/3}\right] + \left[\left(\frac{L_f R}{\epsilon}\right)^{2} \wedge \left(\frac{N L_g R^2}{\epsilon}\right)^{1/2}\right]\right) \tag{13}$$

queries of the algorithm are all ϵ-suboptimal for $F_{\max}(x) = \max_{i\in[N]} f_i(x)$.

See Appendix D.3 for a proof of this result. The first (linear-in-N) term in the lower bound follows from Proposition 8 and Lemma 9 via a re-scaling argument. The second (sublinear-in-N) lower bound term is a direct consequence of existing lower bounds [14, 39, 15].

We remark that our lower bound is stated for optimization constrained to a ball of radius R, while our upper bounds assume unconstrained optimization given a minimizer of norm at most R. These two settings are essentially equivalent; in Appendix D.4 we sketch a general technique for transferring lower bounds to the unconstrained setting.

In Table 1 we specify our lower bound in the special cases $L_g = \infty$ and $L_g = \Theta(L_f^2/\epsilon)$, showing that they match our upper bounds (up to polylogarithmic factors) for $N = \Omega((L_f R/\epsilon)^2)$ in the former case and for any N in the latter. More broadly, when $L_g = \Theta(L_f^{2+q} R^q / \epsilon^{1+q})$ our lower and upper bounds match for any N and $q \in [0, 2/3]$. For $L_g = o(L_f^2/\epsilon)$ and $L_g = \omega(L_f^{8/3} R^{2/3}/\epsilon^{5/3})$, however, there remain gaps between our upper and lower bounds. We discuss these gaps in the following section.

6. Discussion

To conclude the paper, we provide some commentary on our results and the possibilities of improving them. For simplicity, in this section we revert to the setting $L_f = R = 1$ used in the introduction. We also use $a \ll b$ as a shorthand for $a = O(b)$, and ignore constant and logarithmic factors throughout.

6.1. Gaps between the upper and lower bounds

Regimes where a gap exists. Comparing our upper bound in Theorem 6 to our lower bound in Theorem 10, we identify two regimes where our upper and lower bounds disagree by more than polylogarithmic factors. The first is the *smooth regime* $L_g \ll \epsilon^{-1}$, the lower bound is $\Omega(N L_g^{1/3} \epsilon^{-1/3} + \sqrt{N L_g \epsilon^{-1}})$ while our upper bound is $\widetilde{O}(N\epsilon^{-2/3} + \sqrt{N}\epsilon^{-1})$, and a different algorithm gives a better oracle complexity $O(N\sqrt{L_g \epsilon^{-1}})$ (see Appendix A.3) which still falls short of the lower bound.

The second regime is the *non-smooth* regime $L_g \gg \epsilon^{-1}$, where both the upper and lower bounds share the term $N\epsilon^{-2/3}$. Comparing the lower bound to the variance reduced upper bound (6), we see that they disagree if and only if $N\epsilon^{-2/3} + \epsilon^{-2} \ll \sqrt{N L_g \epsilon^{-1}}$ which is equivalent to $N \ll L_g \epsilon^{1/3}$

and $N \gg \epsilon^{-3}/L_g$. Clearly, this is only possible only when $L_g \gg \epsilon^{-5/3}$, and so we conclude that the rate (6) is in fact optimal whenever $\epsilon^{-1} \ll L_g \ll \epsilon^{-5/3}$. Moreover, the upper bound (8) matches the lower bound whenever $N \gg \epsilon^{-2}$ for any $L_g \gg \epsilon^{-1}$. We conclude that gaps in the non-smooth regime exist only for $L_g \gg \epsilon^{-5/3}$ and $\epsilon^{-3}/L_g \ll N \ll \min\{\epsilon^{-2}, \epsilon^{1/3}L_g\}$.

Closing the gap in the non-smooth regime. Improving the bound (8) from $\widetilde{O}(N\epsilon^{-2/3} + \epsilon^{-8/3})$ to $\widetilde{O}(N\epsilon^{-2/3} + \epsilon^{-2})$ would imply that (13) gives the optimal rate for any $L_g \gg \epsilon^{-1}$. The main barrier for obtaining such improvement is our accuracy requirements $\delta_t = O(\epsilon/\lambda_t)$ in Algorithm 1. Meeting this requirement with SGD means that each oracle implementation costs $\widetilde{O}(N + \epsilon^{-2})$ function/gradient evaluations, and multiplying this cost by the number of rounds $\widetilde{O}(\epsilon^{-2/3})$ yields the exponent $8/3$. A variant of Algorithm 1 which can handle less accurate BROO outputs could close this gap by allowing a more efficient SGD-based implementation.

Closing the gap in the smooth regime. The gap between our upper and lower bounds when $L_g \ll \epsilon^{-1}$ is more fundamental than the one arising for $L_g \gg \epsilon^{-5/3}$, because it affects the term linear in N. The barrier for improving the linear term in our algorithm is the ball radius. Any r_ϵ-BROO implementation with $\Omega(N)$ cost will have overall complexity $\Omega(Nr_\epsilon^{-2/3})$. The techniques we develop in Section 4 only allow us to support $r_\epsilon = \widetilde{O}(\epsilon)$, because this is the largest radius where the exponentiated softmax is stable (see Lemma 3).

Conjectures and future work. We conjecture that our lower bound is in fact optimal in both smoothness regimes. In future work we will attempt to close the remaining complexity gaps described above.

6.2. Some necessary algorithmic structures

Several aspects of our method, namely functions value access, individual function queries and randomization are necessary for any method that achieves (or improves on) our complexity bounds. See Appendix A.4 for detailed discussion.

6.3. Practical considerations

The main purpose of the algorithms we develop in this paper is to clarify the complexity of the fundamental optimization (1). Nevertheless, since this problem formulation is relevant for a number of machine learning tasks [13, 21, 36], it is interesting to try and develop a more practical variant of algorithms. Two aspects of our method which we believe will be particularly useful in practice are the gradient estimation scheme we use in Algorithm 2 and the momentum scheme in Algorithm 1.

However, a number of aspects of our method seem rather impractical. First, the theory instructs us to constrain subproblem solutions to a very small ball of radius r_ϵ of roughly ϵ/L_f. Since usually neither ϵ or L_f are known in advance, the parameter r_ϵ must be tuned. Moreover, choosing r_ϵ to be small in keeping with the theory would likely mean very slow progress in the early stages of the algorithm. A second impractical aspect is the bisection stage in Algorithm 1: while in theory the bisection only increases complexity by a logarithmic factor, in practice it entails solving a considerable number of sub-problems without making progress. This bisection overhead is an issue with Monteiro-Svaiter acceleration more broadly and a topic of active research [37, 34].

Acknowledgments

YC was supported in part by Len Blavatnik and the Blavatnik Family foundation, and the Yandex Machine Learning Initiative for Machine Learning. YJ was supported by Stanford Graduate Fellowship. AS was supported in part by a Microsoft Research Faculty Fellowship, NSF CAREER Award CCF-1844855, NSF Grant CCF-1955039, a PayPal research award, and a Sloan Research Fellowship.

References

[1] Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *arXiv preprint arXiv:1603.05953*, 2016.

[2] Zeyuan Allen-Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. In *Innovations in Theoretical Computer Science*, 2017.

[3] Yossi Arjevani, Yair Carmon, John C Duchi, Dylan J Foster, Nathan Srebro, and Blake Woodworth. Lower bounds for non-convex stochastic optimization. *arXiv preprint arXiv:1912.02365*, 2019.

[4] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Princeton University Press, 2009.

[5] Aharon Ben-Tal, Dick Den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2):341–357, 2013.

[6] Jan van den Brand, Yin Tat Lee, Yang P Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, MDPs, and ℓ_1-regression in nearly linear time for dense instances. *arXiv preprint arXiv:2101.05719*, 2021.

[7] Sebastien Bubeck, Qijia Jiang, Yin Tat Lee, Yuanzhi Li, and Aaron Sidford. Complexity of highly parallel non-smooth convex optimization. In *Advances in Neural Information Processing Systems*, 2019.

[8] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 2015.

[9] Brian Bullins. Highly smooth minimization of non-smooth problems. In *Conference on Learning Theory*, pages 988–1030, 2020.

[10] Yair Carmon, Yujia Jin, Aaron Sidford, and Kevin Tian. Variance reduction for matrix games. In *Advances in Neural Information Processing Systems*, 2019.

[11] Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points I. *Mathematical Programming*, 184(1):71 – 120, 2020.

[12] Yair Carmon, Arun Jambulapati, Qijia Jiang, Yujia Jin, Yin Tat Lee, Aaron Sidford, and Kevin Tian. Acceleration with a ball optimization oracle. In *Advances in Neural Information Processing Systems*, 2020.

[13] Kenneth L Clarkson, Elad Hazan, and David P Woodruff. Sublinear optimization for machine learning. *Journal of the ACM (JACM)*, 59(5):1–49, 2012.

[14] Jelena Diakonikolas and Cristóbal Guzmán. Lower bounds for parallel and randomized convex optimization. *Journal of Machine Learning Research*, 21(5):1–31, 2020.

[15] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Near-optimal non-convex optimization via stochastic path integrated differential estimator. *Advances in Neural Information Processing Systems*, 31:689, 2018.

[16] Roy Frostig, Rong Ge, Sham Kakade, and Aaron Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *International Conference on Machine Learning*, 2015.

[17] Alexander Gasnikov, Pavel Dvurechensky, Eduard Gorbunov, Evgeniya Vorontsova, Daniil Selikhanovych, César A. Uribe, Bo Jiang, Haoyue Wang, Shuzhong Zhang, Sébastien Bubeck, Qijia Jiang, Yin Tat Lee, Yuanzhi Li, and Aaron Sidford. Near optimal methods for minimizing convex functions with Lipschitz p-th derivatives. In *Conference on Learning Theory*, 2019.

[18] Cristóbal Guzmán and Arkadi Nemirovski. On lower complexity bounds for large-scale smooth convex optimization. *Journal of Complexity*, 31(1):1–14, 2015.

[19] Osman Güler. New proximal point algorithms for convex minimization. *SIAM Journal on Optimization*, 2(4):649–664, 1992.

[20] Elad Hazan and Satyen Kale. Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization. *The Journal of Machine Learning Research*, 15(1): 2489–2512, 2014.

[21] Elad Hazan, Tomer Koren, and Nati Srebro. Beating SGD: Learning SVMs in sublinear time. In *Advances in Neural Information Processing Systems*, pages 1233–1241, 2011.

[22] Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992.

[23] Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 230–249. IEEE, 2015.

[24] Anatoly Yur'evich Levin. An algorithm for minimizing convex functions. *Doklady Akademii Nauk SSSR*, 160(6):1244–1247, 1965.

[25] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, 2015.

[26] Renato DC Monteiro and Benar Fux Svaiter. An accelerated hybrid proximal extragradient method for convex optimization and its implications to second-order methods. *SIAM Journal on Optimization*, 23(2):1092–1125, 2013.

[27] Hongseok Namkoong and John C Duchi. Stochastic gradient methods for distributionally robust optimization with f-divergences. In *Advances in Neural Information Processing Systems*, volume 29, pages 2208–2216, 2016.

[28] Arkadi Nemirovski. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

[29] Arkadi Nemirovski and David Yudin. *Problem complexity and method efficiency in optimization*. Wiley-Interscience, 1983.

[30] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4): 1574–1609, 2009.

[31] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[32] Yurii Nesterov. Dual extrapolation and its applications to solving variational inequalities and related problems. *Mathematical Programming*, 109(2):319–344, 2007.

[33] Yurii Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.

[34] Yurii Nesterov. Implementable tensor methods in unconstrained convex optimization. *Mathematical Programming*, pages 1–27, 2019.

[35] D. J. Newman. Location of the maximum on unimodal surfaces. *J. ACM*, 12(3):395–398, 1965.

[36] Shai Shalev-Shwartz and Yonatan Wexler. Minimizing the maximal loss: How and why. In *International Conference on Machine Learning*, pages 793–801, 2016.

[37] Chaobing Song, Yong Jiang, and Yi Ma. Unified acceleration of high-order algorithms under Hölder continuity and uniform convexity. *arXiv preprint arXiv:1906.00582*, 2019.

[38] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

[39] Blake Woodworth and Nathan Srebro. Tight complexity bounds for optimizing composite objectives. In *Advances in Neural Information Processing Systems*, pages 3646–3654, 2016.