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,

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

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.

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

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

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

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

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 $\|\mathcal{O}_{\lambda,\delta}(\bar{x}) - \mathrm{bprox}^f_{\lambda,r}(\bar{x})\| \le \delta$.

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 $F_{\mathrm{smax},\epsilon}$, 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 $F_{\mathrm{smax},\epsilon}$ 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 $\epsilon$, BROO $\mathcal{O}_{\lambda,\delta}(\cdot)$
Output: $x_{\rm ret}$ such that $f(x_{\rm ret}) - \arg\min_{z \in \mathbb{B}_R(x_0)} f(z) \le \epsilon$
1  Let $v_0 = x_0$, $A_0 = 0$
2  for $t = 0, 1, 2, \dots$ do
3      $\lambda_{t+1} = \lambda$-BISECTION$(x_t, v_t, A_t, \lambda_{\max} = \frac{2L_f}{r}, \lambda_{\min} = \frac{\epsilon}{6rR})$
           ▷ Finds $\lambda_{t+1}$ such that $x_{t+1} \approx \mathrm{prox}^f_{\lambda_{t+1}}(y_t)$ and either $\|x_{t+1} - y_t\| \approx r$ or $x_{t+1}$ is $\epsilon$-optimal
4      $a_{t+1} = \frac{1}{2\lambda_{t+1}}(1 + \sqrt{1 + 4\lambda_{t+1}A_t})$ and $A_{t+1} = A_t + a_{t+1}$    ▷ $A_{t+1} = a_{t+1}^2 \lambda_{t+1}$
5      $y_t = \frac{A_t}{A_{t+1}} x_t + \frac{a_{t+1}}{A_{t+1}} v_t$
6      $x_{t+1} = \mathcal{O}_{\lambda_{t+1}, \delta_{t+1}}(y_t)$, where $\delta_{t+1} = \frac{\epsilon}{12\lambda_{t+1}R}$
7      $v_{t+1} = \arg\min_{v \in \mathbb{B}_R(x_0)} \left\{ a_{t+1}\lambda_{t+1} \langle y_t - x_{t+1}, v \rangle + \frac{1}{2}\|v - v_t\|^2 \right\}$
8      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
9          return $x_{\rm ret} \in \arg\min_{x \in \{x_0, x_1, \dots, x_{t+1}\}} f(x)$
10 function $\lambda$-BISECTION$(x, v, A, \lambda_{\max}, \lambda_{\min})$
11     For all $\lambda'$, let $y_{\lambda'} := \alpha_{2A\lambda'} \cdot x + (1 - \alpha_{2A\lambda'}) \cdot v$, where $\alpha_\tau := \frac{\tau}{1+\tau+\sqrt{1+2\tau}}$
12     Define $\Delta(\lambda) := \|\mathcal{O}_{\lambda, \frac{r}{17}}(y_\lambda) - y_\lambda\|$    ▷ approximation of $\widehat{\Delta}(\lambda) := \|\mathrm{bprox}^f_{\lambda,r}(y_\lambda) - y_\lambda\|$
13     Let $\lambda = \lambda_{\max}$
14     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
15     if $\lambda \le \lambda_{\min}$ then return $2\lambda$    ▷ happens only if $\mathrm{bprox}^f_{2\lambda,r}(y_{2\lambda})$ is $O(\epsilon)$-optimal for small $\lambda_{\min}$
16     Let $\lambda_u = 2\lambda$, $\lambda_\ell = \lambda$ and $\lambda_m = \sqrt{\lambda_u \lambda_\ell}$
17     if $\Delta(\lambda_\ell) \le \frac{15r}{16}$ then return $\lambda_\ell$    ▷ happens only if $\Delta(\lambda_\ell) \in [\frac{13r}{16}, \frac{15r}{16}]$
18     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
19         if $\Delta(\lambda_m) < \frac{13r}{16}$ then $\lambda_u = \lambda_m$ else $\lambda_\ell = \lambda_m$
20         $\lambda_m = \sqrt{\lambda_u \lambda_\ell}$
21     return $\lambda_m$    ▷ the while loop terminates in $O\big(\log(\frac{R}{r} + \frac{L_f}{\lambda_{\min} r})\big)$ 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 $F_{\mathrm{smax},\epsilon}(x) = \epsilon' \log\Big(\sum_{i\in[N]} e^{f_i(x)/\epsilon'}\Big)$ approximates $F_{\max}$ to within $\epsilon/2$ additive error. The key challenge in designing an efficient stochastic method for minimizing $F_{\mathrm{smax},\epsilon}$ is a lack of cheap unbiased gradient estimators. Specifically, we have $\nabla F_{\text{smax},\epsilon}(x) = \sum_{i\in[N]} p_i(x)\nabla f_i(x)$, 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 $\nabla F_{\text{smax},\epsilon}(x)$ 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 $\nabla F_{\text{smax},\epsilon}$ 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 $\nabla F_{\text{smax},\epsilon}(x)$. Instead, we define below a surrogate function "exponentiating the softmax" that closely approximates $F_{\text{smax},\epsilon}$ 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

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

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

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

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 $F^\lambda_{\text{smax},\epsilon}$ and is as regular as $F^\lambda_{\text{smax},\epsilon}$ 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. *$F^\lambda_{\text{smax},\epsilon}(x)$ and $\Gamma_{\epsilon,\lambda}$ have the same minimizer $x_\star$ in $\mathbb{B}_r(\bar{x})$. Moreover, for every $x \in \mathbb{B}_r(\bar{x})$,*

$$F^\lambda_{\text{smax},\epsilon}(x) - F^\lambda_{\text{smax},\epsilon}(x_\star) \le C(\Gamma_{\epsilon,\lambda}(x) - \Gamma_{\epsilon,\lambda}(x_\star)).$$

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 $F_{\text{smax},\epsilon}$ in the sense that $\mathbb{E}g_i(x) = Z(x)\nabla F_{\text{smax},\epsilon}(x)$ 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 $\lambda \le O(L_f/r_\epsilon)$, 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 $\lambda \le O(L_f/r_\epsilon)$, 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 $\|x_0 - x_\star\| \le R$ 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 $K := L_f R\epsilon^{-1}\log N$. 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 Kayushal 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 $\widetilde{O}(N\mathcal{T}_f + L_f^2R^2\epsilon^{-2}(\mathcal{T}_f+\mathcal{T}_g))$. 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 $t \ge 1$, we have

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

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 $x_1, \dots, x_t$ 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

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

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 $v_{[\le l]}$ denotes the vector whose first l coordinates are identical to those of v and the remainder are zero. Recall also that $\mathbb{B}^T_1(0)$ is the unit ball in $\mathbb{R}^T$.

Definition 7 *A sequence $f_1, \dots, f_N$ of functions $f_i : \mathbb{B}^T_1(0) \to \mathbb{R}$ is called an α-*robust N-element zero-chain *if for all $x \in \mathbb{B}^T_1(0)$, all y in a neighborhood of x, and all $i \in [N]$, we have*

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

To unpack this definition, consider any first-order algorithm with the following two simplifying properties: (1) the queries $i_1, i_2, \dots$ are drawn i.i.d. from $\mathrm{Uniform}([N])$ and (2) every query x_t lies in the span of previously observed gradients $\nabla f_{i_1}(x_1), \dots, \nabla f_{i_{t-1}}(x_{t-1})$ [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 $f_2, \dots, f_N$ 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 $f_1, \dots, f_N$. 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 $T \le N/2$. Let $(f_i)_{i \in [N]}$ be an α-robust N-element zero-chain with domain $\mathbb{B}^T_1(0)$. For $d \ge T + \frac{2}{\alpha^2}\log\frac{4NT^2}{\delta}$, draw U uniformly from the* *set of $d \times T$ orthogonal matrices, and draw Π uniformly from the set of permutations of $[N]$. Let $\tilde{f}_i(x) := f_{\Pi^{-1}(i)}(U^\top x)$. Let $\{(i_t, x_t)\}_{t\ge 1}$ 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*

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

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

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

Clearly, $\psi_{\alpha,\ell}$ is 1-Lipschitz, ℓ-smooth, and is identically zero for all $|t| \le \alpha$. 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

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

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 $\ell \ge 0$, such that $T \le N$, 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*

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

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 $\mathbb{B}^d_R(0)$ 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 $\Omega(N r_\epsilon^{-2/3})$. 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.

Appendix A. Additional discussion

Here we discuss three important points that exceeded the scope of our introduction. First, Appendix A.1 explains what makes L_f^2/ϵ a natural smoothness scale and when we can expect the f_i to be at least that smooth. Then, Appendix A.2 discusses the maximally smooth case of linear loss functions, and compare our results to guarantees of methods specialized to this setting. Finally, Appendix A.3 considers the computational complexity of implementing the steps in the accelerated iterative linearization scheme of [33, Section 2.3.1].

A.1. The generality of the smoothness assumption $L_g = O(L_f^2/\epsilon)$

Let f be a convex, L_f Lipschitz function. When f is not continuously differentiable, it is still possible to uniformly approximate it with a continuously differentiable function, that moreover has a Lipschitz gradient. More concretely, for every $\varepsilon > 0$, we may consider the infimal convolution of f with a quadratic regularizer (also known as its Moreau envelope):

$$\tilde{f}(x) = \min_{y\in\mathbb{R}^d}\left\{ f(y) + \frac{L_f^2}{2\varepsilon}\|x-y\|^2 \right\}. \tag{14}$$

It holds that $0 \le f(x) - \tilde{f}(x) \le \varepsilon/2$ for all $x \in \mathbb{R}^d$ and moreover that $\tilde{f}$ is L_f^2/ε smooth (i.e., with L_f^2/ε Lipschitz gradient) [see, e.g., 18]. Therefore, if we wish to minimize f to accuracy ϵ, we may choose $\varepsilon = \epsilon$ and replace f with the $O(L_f^2/\epsilon)$-smooth function $\tilde{f}$.

The computational cost of such replacement depends on the application. In certain cases, smoothed versions of the f_i's have closed-form expression, and we may simply define the problem with them instead. Moreover, in several machine learning applications each function f_i is "simple," and querying index i really corresponds to obtaining full access to this function, in which case directly computing (14) might be feasible.

However, when we are truly restricted to accessing f_i through a gradient and value black box, there will be some instances where computing (14) (and indeed any other smoothing) is substantially more expensive than a single oracle query. To see this, note that the worst-case oracle complexity of minimizing a single non-smooth convex function scales as $\Omega(L_f^2 R^2 \epsilon^{-2})$, while the complexity of minimizing an L_g-smooth function scales as $O(\sqrt{L_g R^2 \epsilon^{-1}})$. If T_ϵ is the worst number of oracle calls required to compute an $O(\epsilon)$-accurate $O(L_f^2/\epsilon)$-smooth approximation for any L_f-Lipschitz, we immediately have a general complexity upper bound of $O(T_\epsilon L_f R \epsilon^{-1})$. Comparing it to the lower bound $\Omega(L_f^2 R^2 \epsilon^{-2})$ immediately yields that $T_\epsilon = \Omega(L_f R \epsilon^{-1})$ in the worst case. Therefore, the fact that our algorithm maintains the same leading order dependence on N even for $L_g = \infty$ is nontrivial and potentially useful.

A.2. The special case of linear loss functions ($L_g = 0$)

When the losses are linear (i.e., $f_i(x) = a_i^\top x + b_i$) exactly N function value and gradient evaluations suffice to completely identify the problem instance, so the optimal oracle complexity should never be more than N. In this setting, then, it is more relevant to discuss the *computational* (runtime) complexity of solving the problem (1). To simplify the following discussion, we return to the setting in the introduction where each f_i is 1-Lipschitz (i.e., ∇f_i has norm at most 1) and we assume the existence of a minimizer of $F_{\max}$ with Euclidean norm at most 1.

In the linear setting, an equivalent form of the problem (1) is

$$\underset{x\in\mathbb{R}^d}{\text{minimize}}\ \max_{p\in\Delta^N} p^\top(Ax-b)$$

where A is a matrix whose ith row contains ∇f_i. Stochastic primal-dual methods are able to take advantage of this matrix structure to obtain cheap unbiased estimates for the gradients of $p^T Ax$ with respect to both x and p, sampling rows and columns of A respectively. Assuming that reading a row and a column of A takes $O(N+d)$ time, the stochastic primal-dual method has runtime complexity $\widetilde{O}((N+d)\epsilon^{-2})$ which, for sufficiently large ϵ, is sublinear in the problem size Nd [13]. For lower values of ϵ, a variance reduction technique [10] has preferable runtime complexity $\widetilde{O}(Nd+\sqrt{Nd(N+d)}\epsilon^{-1})$.

In comparison, in the linear case our method has runtime complexity $\widetilde{O}(Nd\epsilon^{-2/3}+\sqrt{N}d\epsilon^{-1})$ assuming that sampling a row takes $O(d)$ time. This improves on the variance reduction method in the somewhat narrow parameter regime $N=\widetilde{\Omega}(d)$ and $d=\widetilde{O}(\epsilon^{-2/3})$. However, we note that our method operates under a strictly weaker access assumption, since it only samples rows and not columns. In scenarios where accessing a column takes $\omega(N)$ time, the relative merit of the methods changes. In the most extreme case where reading a column of A is as expensive as reading the entire matrix (e.g., because the rows are scattered across many devices), the stochastic primal-dual methods become less efficient than exact-gradient counterparts with runtime $\widetilde{O}(Nd\epsilon^{-1})$ [31, 28, 32], while the runtime of our method is unchanged and always superior. To the best of our knowledge, this is the first guarantee for a stochastic gradient method that improves on exact gradient methods in a first-order oracle model that can only provides rows of A.

The literature also considers high-order methods for solving the linear case of problem (1) to better accuracy but with potentially worse dependence on problem dimension. Bullins [9] proposes a fourth-order accelerated regularization method that requires $\widetilde{O}(\epsilon^{-4/5})$ solutions of linear systems of the form $A^\top DA=b$ for a positive diagonal matrix D. Carmon et al. [12] use ball oracle acceleration to obtain an improved method requiring only $\widetilde{O}(\epsilon^{-2/3})$ linear system solutions. They also propose to solve these systems using an efficient first-order method, resulting in a runtime guarantee of $\widetilde{O}(Nd\epsilon^{-2/3}+d^{3/2}\epsilon^{-5/3})$. We believe more careful reasoning about the conditioning of each linear system to be solved (as we do in Section 4.3) would improve this guarantee to $\widetilde{O}(Nd\epsilon^{-2/3}+d^{3/2}\epsilon^{-1})$ under the assumption that individual entries of A take $O(1)$ time to read. In the linear case, this improves on our result when $d<N$, albeit with stronger matrix access assumptions.

For even higher accuracy, it is possible to express the linear case of problem (1) as a linear program and solve it using interior point methods. The best existing theoretical runtimes for these methods are $\widetilde{O}((Nd+(N\wedge d)^2)\sqrt{N\wedge d})$ [23] or $\widetilde{O}(Nd+d^{2.5})$ [6], both depending logarithmically on the desired accuracy $1/\epsilon$. When the problem dimensions N and d are sufficiently large compared to $1/\epsilon$, first-order methods are preferable.

A.3. The computational complexity of accelerated iterative linearization

This subsection uses our full notation defined in Section 2. We suggest considering this section after reading Sections 3 and 4 as well.

In [33, Section 2.3.1], Nesterov shows how solving $O(\sqrt{L_g R^2\epsilon^{-1}})$ subproblems of the form

$$\min_{x\in\mathbb{R}^d}\max_{i\in[N]}\left\{f_i(y_t)+(\nabla f_i(y_t))^\top(x-y_t)+\frac{L_g}{2}\|x-y_t\|^2\right\}$$

allows solving the problem (1) when the functions are L_g-smooth. We note that each subproblem is equivalent to

$$\min_{x\in\mathbb{R}^d}\max_{p\in\Delta^N}\left\{p^\top(Ax-b)+\frac{L_g}{2}\|x\|^2\right\}$$

for a matrix $A\in\mathbb{R}^{N\times d}$ whose rows have norm at most L_f. Consequently we may apply a variance-reduced bilinear saddle-point method to solve the subproblem to additive error ν in time

$$\widetilde{O}\left(Nd+\sqrt{Nd(N+d)}\frac{L_f}{\sqrt{L_g\nu}}\right),$$

see Proposition 6 and the subsequent discussion in the arXiv version of [10].

Applying the same arguments used to prove Theorem 2—but with $\lambda_t = L_g$ for all t—we have that the required subproblem solution accuracy is $O(\frac{\epsilon^2}{L_g R^2})$, and consequently we can solve each problem in time

$$\widetilde{O}\left(Nd+\sqrt{Nd(N+d)}\frac{L_f R}{\epsilon}\right).$$

Assuming $\mathcal{T}_f+\mathcal{T}_g=\Omega(d)$, the overall cost of the method is

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

A.4. Some necessary algorithmic structures

We now argue that several aspects of our method, namely functions value access, individual function queries and randomization are necessary in any method that achieves (or improves on) our complexity bounds.

Function value access. It is possible to minimize a convex function f by iterative (sub)gradient evaluations, without access to the value of f itself. In contrast, all algorithms for minimizing $F_{\max}=\max_{i\in[N]} f_i(x)$ *must* query the values of the f_i's in addition to their gradients. To see why this is so, consider the case where $f_i(x)=\Pi(i)-x_i$, where Π is a random permutation of $[N]$ and the domain is the unit Euclidean ball. The global minimum of $\max_{i\in[N]} f_i(x)$ is the $\Pi^{-1}(N)$-th standard basis vector. However, gradients provide no information about $\Pi^{-1}(N)$, since $\nabla f_i(x)=-e_i$ for all x, independent of Π.

Individual-function access. The algorithms from prior work in Table 1 (namely the subgradient method, AGD on softmax and AGD on linearization) are full-batch methods: they proceed by querying all N functions $f_1,\ldots,f_N$ at the same point x_t and using the result to generate the next query point x_{t+1}. In contrast, our BROO implementations proceed by sampling an index i_t, computing ∇f_i at x_t (and potentially another point), and generating the next query x_{t+1}. Full-batch methods are more amenable to parallelization, but for our problem have demonstrably worse oracle complexity. To see this, consider the case where all the f_i's are identical and equal to standard hard instance for convex optimization. For such input, any full-batch methods will have oracle complexity $\Omega(N\min\{\epsilon^{-2},\sqrt{L_g\epsilon^{-1}}\})$ [14], which is worse than our upper bounds for any $L_g\gg\epsilon^{-1/3}$ and sufficiently large N.

Randomization. Another contrast between the prior algorithms in Table 1 and our algorithm is that the former are deterministic while ours is randomized. Woodworth and Srebro [39] prove a lower bound of $\Omega(N \min\{\epsilon^{-2}, \sqrt{L_g \epsilon^{-1}}\})$ gradient queries for any deterministic method for minimizing the *average* of N functions. Observing that the maximum of the N functions in their construction has the same minimum value as their average (and that the maximum upper bounds the average in any other points), we conclude that this lower bound is also valid for any deterministic method for solving the problem (1). Therefore, randomization is necessary for obtaining our improved rates of convergence.

Appendix B. Proof of Theorem 2

In this section we give the analysis of our accelerated algorithm. Our analysis builds off of [12] and proceeds in several parts. We first prove several standard technical results Appendix B.1. Then, in Appendix B.2 we give the proof of Theorem 2 assuming the correctness of the λ-BISECTION subroutine. Finally in Appendix B.3 we prove this correctness.

B.1. Preliminary technical results

First, we observe that $\mathcal{O}_{\lambda,\delta}(y)$ returns a point which is close to $\mathrm{bprox}^f_{\lambda,r}(y)$, the true minimizer of the proximal objective.

Lemma 11 *Let f be a convex function and let $x = \mathrm{bprox}^f_{\lambda,r}(y)$. Then if $\mathcal{O}_{\lambda,\delta}(\cdot)$ is an r-BROO for f, the point $\widetilde{x} = \mathcal{O}_{\lambda,\delta}(y)$ satisfies $\|\widetilde{x} - x\| \le \delta$.*

Proof By Definition 1 of the BROO, we have

$$\nu = \left[f(\widetilde{x}) + \frac{\lambda}{2}\|\widetilde{x} - y\|^2\right] - \left[f(x) + \frac{\lambda}{2}\|x - y\|^2\right] \le \frac{\lambda\delta^2}{2}.$$

Since the function $f(x)+\frac{\lambda}{2}\|x-y\|^2$ is λ-strongly convex, we obtain $\nu \ge \frac{\lambda}{2}\|x-\widetilde{x}\|_2^2$, and substituting $\nu \le \lambda\delta^2/2$ gives the result. ■

Second, we provide standard facts regarding proximal mappings. Though this follows from standard facts regarding subgradients we provide a self-contained proof for completeness.

Lemma 12 *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function, $S \subseteq \mathbb{R}^d$ be a closed convex set, $\lambda \ge 0$, and $x_\lambda, x_0 \in S$ satisfy*

$$x_\lambda = \arg\min_{x \in S}\left\{f(x) + \frac{\lambda}{2}\|x - x_0\|^2\right\}$$

Then $g_\lambda := \lambda(x_0 - x_\lambda)$ is a subgradient of f, i.e.,

$$f(y) \ge f(x_\lambda) + \langle g_\lambda, y - x_\lambda\rangle \text{ for all } y \in S\,. \tag{15}$$

Further, we have

$$g_\lambda^\top(x_\lambda - z) = \frac{\lambda}{2}\|z - x_0\|^2 - \frac{\lambda}{2}\|z - x_\lambda\|^2 - \frac{\lambda}{2}\|x_\lambda - x_0\|^2 \text{ for all } z \in \mathbb{R}^d \tag{16}$$

and

$$f(x_\lambda) \le f(y) + \frac{\lambda}{2}\|y - x_0\|^2 - \frac{\lambda}{2}\|y - x_\lambda\|^2 - \frac{\lambda}{2}\|x_\lambda - x_0\|^2 \text{ for all } y \in S\,. \tag{17}$$

Proof Let $x^\alpha := \alpha \cdot y + (1-\alpha)\cdot x_\lambda$ for all $\alpha \in (0,1]$. Note that $x^\alpha \in S$ for all $\alpha \in (0,1]$ since S is convex and $x_\lambda, y \in S$. Consequently, (15) and convexity of f imply that for all $\alpha \in [0,1]$

$$\begin{aligned} f(x_\lambda) + \frac{\lambda}{2}\|x_\lambda - x_0\|^2 &\le f(x^\alpha) + \frac{\lambda}{2}\|x_0 - x^\alpha\|^2 \\ &\le \alpha f(y) + (1-\alpha) f(x_\lambda) + \frac{\lambda}{2}\|\alpha\,(x_0 - y) + (1-\alpha)(x_0 - x_\lambda)\|^2 . \end{aligned}$$

Rearranging yields that for all $\alpha > 0$ implies

$$f(x_\lambda) - f(y) \le \frac{\lambda}{2\alpha}\left[\alpha^2\|x_0 - y\| + 2\alpha(1-\alpha)\left\langle x_0 - y, x_0 - x_\lambda\right\rangle + (1-\alpha)^2\|x_\lambda - x_0\|^2 - \|x_\lambda - x_0\|^2\right]$$

Taking the limit as $\alpha \to 0$ yields that

$$f(x_\lambda) - f(y) \le \lambda(x_0 - y)^\top(x_0 - x_\lambda) - 2\lambda\|x_\lambda - x_0\|^2 = \lambda(x_\lambda - x_0)^\top(x_0 - x_\lambda)$$

The remaining claims (16) and (17) follow from direct algebraic manipulation of this inequality and the definition $g_\lambda = \lambda(x_0 - x_\lambda)$. ■

Third, we bound the function error induced by proximal mapping.

Lemma 13 *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function, $\lambda \ge 0$, and $x_\lambda, x_0 \in \mathbb{R}^d$ satisfy $x_\lambda = \mathrm{prox}^f_\lambda(x_0)$. If $y \in \mathbb{R}^d$ satisfies $\|y - x_0\| \le R$ and $\|x_\lambda - x_0\| \le \Theta$ then $f(x_\lambda) - f(y) \le \lambda\Theta R$.*

Proof Bound (15) in Lemma 12 yields

$$\begin{aligned} f(x_\lambda) - f(y) &\le -\lambda\left\langle x_0 - x_\lambda, y - x_\lambda\right\rangle = -\lambda\left\langle x_0 - x_\lambda, y - x_0\right\rangle - \lambda\|x_0 - x_\lambda\|^2 \\ &\le \lambda\left\langle x_\lambda - x_0, y - x_0\right\rangle \le \lambda\|x_0 - x_\lambda\| \cdot \|y - x_0\| \le \lambda\Theta R . \end{aligned}$$

■

Fourth, we prove that for any constrained minimizer of f (denoted $x_\star$), BROO calls either decrease the distance to $x_\star$ or have objective value not much worse than $x_\star$.

Lemma 14 *Let f be a convex function, $\epsilon, R \ge 0$, $x_0 \in \mathbb{R}^d$, $x_\star \in \mathbb{B}_R(x_0)$, $y \in \mathbb{B}_R(x_\star)$, and $x' = \mathcal{O}_{\lambda,\delta}(y)$ for $\delta \le \frac{\epsilon}{4\lambda R}$ and $\lambda \ge \frac{\epsilon}{3R^2}$. If $f(x') - f(x_\star) > \frac{\epsilon}{2}$ and $\mathrm{prox}^f_\lambda(y) = \mathrm{bprox}^f_{\lambda,r}(y)$ then $\|x' - x_\star\| < \|y - x_\star\|$.*

Proof Let $x_\lambda = \mathrm{prox}^f_\lambda(y)$. Since $\mathrm{prox}^f_\lambda(y) = \mathrm{bprox}^f_{\lambda,r}(y)$, Definition 1 of $\mathcal{O}_{\lambda,\delta}(\cdot)$ implies

$$f(x') \le f(x') + \frac{\lambda}{2}\|x' - y\|^2 \le f(x_\lambda) + \frac{\lambda}{2}\|x_\lambda - y\|^2 + \frac{\lambda\delta^2}{2}.$$

Further, Equation (17) implies that

$$f(x_\lambda) \le f(x_\star) + \frac{\lambda}{2}\|x_\star - y\|^2 - \frac{\lambda}{2}\|x_\star - x_\lambda\|^2 - \frac{\lambda}{2}\|x_\lambda - y\|^2.$$

Combining these inequalities, rearranging, and using $f(x') - f(x_\star) > \frac{\epsilon}{2}$ yields

$$\frac{\lambda}{2}\|x_\lambda - x_\star\|^2 \le \frac{\lambda}{2}\|y - x_\star\|^2 - \frac{\epsilon}{2} + \frac{\lambda\delta^2}{2}. \tag{18}$$

Since $\lambda\delta^2 \le \frac{\epsilon^2}{12\lambda R^2} \le \frac{\epsilon}{6} < \frac{\epsilon}{2}$, this implies $\|x_\lambda - x_\star\| \le \|y - x_\star\|$.

By the triangle inequality, we have $\|x' - x_\star\| \le \|x_\lambda - x_\star\| + \|x' - x_\lambda\|$. Thus,

$$\begin{aligned}\|x' - x_\star\|^2 &\le \|x_\lambda - x_\star\|^2 + 2\|x_\lambda - x_\star\| \cdot \|x' - x_\lambda\| + \|x' - x_\lambda\|^2 \\ &\le \|x_\lambda - x_\star\|^2 + 2\delta\|y - x_\star\| + \delta^2 \le \|x_\lambda - x_\star\|^2 + 2\delta R + \delta^2,\end{aligned}$$

where we have used $\|x_\lambda - x_\star\| \le \|y - x_\star\|$ (argued above) along with $\|x' - \hat{x}\| \le \delta$ (Lemma 11) and the assumption that $y \in \mathbb{B}_R(x_\star)$. Substituting into (18), we have

$$\lambda\|x' - x_\star\|^2 \le \lambda\|y - x_\star\|^2 - \epsilon + 2\lambda\delta R + 2\lambda\delta^2.$$

To conclude the proof we note that $-\epsilon + 2\lambda\delta R + 2\lambda\delta^2 < 0$ since $\lambda\delta \le \epsilon/(4R)$ and $\lambda\delta^2 \le \epsilon/6$. ■

Fifth, we bound the movement of the proximal operator by the Lipschitz continuity of the objective.

Lemma 15 *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and L_f-Lipschitz. Then for all $\lambda > 0$ and $y \in \mathbb{R}^d$,*

$$\|\mathrm{prox}_\lambda^f(y) - y\| \le \frac{L_f}{\lambda}.$$

Proof Let $x_\lambda \coloneqq \mathrm{prox}_\lambda^f(y) - y$. Equation (15) of Lemma 12 implies that

$$f(y) - f(x_\lambda) \ge \lambda \langle x_0 - x_\lambda, x_0 - x_\lambda\rangle = \lambda\|x_0 - x_\lambda\|^2.$$

Further, L_f-Lipschtiz continuity of f implies $f(y) \le f(x_\lambda) + L_f\|x_0 - x_\lambda\|^2$. Combining and noting that the claim is trivial when $\|x_0 - x_\lambda\| = 0$ yields the claim. ■

Finally, we mention a standard lemma about the relation between the sequences $\{A_t\}$ and $\{\lambda_t\}$ in accelerated proximal methods.

Lemma 16 ([cf. 12, Lemma 23]) *For any iteration t of Algorithm 1, we have $A_t = a_t^2\lambda_t$ and*

$$\sqrt{A_t} \ge \frac{1}{2}\sum_{i\in[t]} \frac{1}{\sqrt{\lambda_i}}.$$

B.2. Main algorithm analysis

In this section we give the analysis of our accelerated algorithm. Before going into the technical details, let us provide a brief overview of the algorithm and its analysis. At its core, our algorithm is an accelerated proximal point method [19, 25, 16]. These methods iteratively compute proximal points of the form $x_{t+1} \approx \mathrm{prox}_{\lambda_{t+1}}^f(y_t)$ and then use a momentum-like extrapolation scheme to compute y_{t+1}. Accelerated proximal points methods differ in the methods they employ to (approximately) compute that proximal points and the choices of $\{\lambda_t\}$. Our method approximates

$x_{t+1} \approx \mathrm{prox}^f_{\lambda_{t+1}}(y_t)$ using calls to a BROO, and employs a bisection procedure that finds values of λ_t for which such approximation is valid because the ball constraint is inactive, i.e., when $\|x_{t+1} - y_t\| < r$.

The crux of the analysis of our method is showing that the optimization error decreases roughly as $\exp\left(-\Omega(1) \sum_{i\in[t]}(\|x_{t+1} - y_t\|/R)^{2/3}\right)$. By requiring our bisection procedure to find values for which $\|x_{t+1} - y_t\| \in [r/2, r)$, we obtain the claimed $\widetilde{O}((R/r)^{2/3})$ complexity bound. Our analysis of our method closely follows the previous ball-oracle acceleration proof of [12], which itself draws from prior analyses of Monteiro-Svaiter-type algorithms [17, 7].

The differences between our algorithm and proof and those in [12] center around handling non-smoothness and less accurate ball oracle outputs. In particular, in Line 7 of Algorithm 1 we estimate $\nabla f(\mathrm{prox}^f_{\lambda_{t+1}}(y_t))$ as $\lambda_{t+1}(y_t - x_{t+1})$, which allows us to avoid smoothness assumptions but requires a somewhat different proof of the main potential bound. We also remove the assumption that x_0 is within distance R from a global minimizer of f, and instead compare the function value of our output to the minimizer of f in a ball of radius R. Our bisection subroutine and its analysis also differ from its counterpart in [12]; we explain these differences in the next subsection.

Our analysis of the Algorithm 1 outer loop relies on the following guarantee for our bisection subroutine, which we prove in Appendix B.3.

Proposition 17 (Bisection) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be L_f-Lipschitz and convex, and let $x, v \in \mathbb{R}^d$, $\epsilon, r, R \in \mathbb{R}_{>0}$ satisfy $\epsilon \le L_f R$, $r \le R$ and $\|x - v\| \le 2R$. Given $\lambda_{\max} \ge \frac{2L_f}{r}$ and $\lambda_{\min} \in (0, \lambda_{\max})$, λ-BISECTION(x, v, A) outputs $\lambda \in [\lambda_{\min}, \lambda_{\max}]$ such that*

$$\mathrm{prox}^f_\lambda(y_\lambda) = \mathrm{bprox}^f_{\lambda,r}(y_\lambda)$$

(i.e., $\|\mathrm{prox}^f_\lambda(y_\lambda) - y_\lambda\| \le r$). The subroutine uses $O(\log(\frac{\lambda_{\max}}{\lambda_{\min}}) + \log(\frac{R + L_f/\lambda_{\min}}{r}))$ calls to $\mathcal{O}_{\lambda', \frac{r}{17}}(\cdot)$ with $\lambda' \in [\frac{1}{2}\lambda, \lambda_{\max}]$. Moreover, for $\alpha_{2\lambda A} = \frac{2\lambda A}{1 + 2\lambda A + \sqrt{1 + 4\lambda A}}$ and $y_\lambda := \alpha_{2\lambda A} x + (1 - \alpha_{2\lambda A})v$ one of the following outcomes must occur:

(a) $\lambda \in [2\lambda_{\min}, \lambda_{\max}]$ *and* $\|\mathrm{prox}^f_\lambda(y_\lambda) - y_\lambda\| > \frac{3r}{4}$, *or*

(b) $\lambda < 2\lambda_{\min}$.

When taking $\lambda_{\max} = \frac{2L_f}{\epsilon}$, $\lambda_{\min} = \frac{\epsilon}{6rR}$, the number of calls to $\mathcal{O}_{\lambda', \frac{r}{17}}(\cdot)$ is bounded by $O(\log \frac{L_f R^2}{r\epsilon})$.

For the remainder of this section, we fix a parameter R and let

$$x_\star \in \arg\min_{x \in \mathbb{B}_R(x_0)} f(x)$$

denote a minimizer of f in a ball of radius R around the initial point x_0. (If it is not unique, we choose one arbitrarily). Based on the iterates $\{x_t, v_t, A_t\}$ generated by Algorithm 1, we define the following quantities:

$$E_t := f(x_t) - f(x_\star),\ \hat{E}_t = E_t - \frac{\epsilon}{4},\ D_t = \frac{1}{2}\|v_t - x_\star\|^2, \text{ and } P_t = A_t\hat{E}_t + D_t. \tag{19}$$

In the following lemma we prove our main potential decrease bound, under the conditions that the iterates x_t, v_t are within distance $2R$ of each other and (implicitly) that a suitably good solution has yet to be found. We establish these conditions inductively in subsequent lemmas and leverage this to lower bound the growth of A_t and prove Theorem 2.

Lemma 18 *Let f be a convex function, $x_0 \in \mathbb{R}^d$ and $\epsilon, R > 0$. If at iteration t of Algorithm 1 the following conditions hold,*

(a) $\|x_t - v_t\| \le 2R$

(b) $\lambda_{t+1} \ge \frac{\epsilon}{3rR}$

we have

$$P_{t+1} - P_t \le -\frac{A_{t+1}\lambda_{t+1}r^2}{12}.$$

Proof Let

$$\hat{x}_{t+1} \coloneqq \mathrm{bprox}^f_{r,\lambda_{t+1}}(y_t) = \operatorname*{arg\,min}_{x \in \mathbb{B}_r(y_t)} \left\{ f(x) + \frac{\lambda_{t+1}}{2}\|x - y_t\|^2 \right\}.$$

Condition (a) and Proposition 17 guarantee that the ball constraint is inactive, i.e.,

$$\hat{x}_{t+1} = \mathrm{bprox}^f_{r,\lambda_{t+1}}(y_t) = \mathrm{prox}^f_{\lambda_{t+1}}(y_t) = \operatorname*{arg\,min}_{x \in \mathbb{R}^d} \left\{ f(x) + \frac{\lambda_{t+1}}{2}\|x - y_t\|^2 \right\}.$$

(Note that y_t is precisely y_λ defined in Proposition 17). Consequently, by Lemma 12 we have that $g_{t+1} \coloneqq \lambda_{t+1}(y_t - \hat{x}_{t+1}) \in \partial f(\hat{x}_{t+1})$, i.e.,

$$f(u) \ge f(\hat{x}_{t+1}) + g_{t+1}^\top(u - \hat{x}_{t+1}) \text{ for all } u \in \mathbb{R}^d. \tag{20}$$

Further, since $v_{t+1} = \arg\min_{v \in \mathbb{B}_R(x_0)} \left\{ a_{t+1} \langle y_t - x_{t+1}, v \rangle + \frac{1}{2}\|v - v_t\|^2 \right\}$ and $x_\star \in \mathbb{B}_R(x_0)$ applying Lemma 12 again (with $u = x_\star \in \mathbb{B}_R(x_0)$) yields that

$$\begin{aligned} a_{t+1}\lambda_{t+1} \langle y_t - x_{t+1}, v_{t+1} - x_\star \rangle &\le \frac{1}{2}\|v_t - x_\star\|^2 - \frac{1}{2}\|v_{t+1} - x_\star\|^2 - \frac{1}{2}\|v_{t+1} - v_t\|^2 \\ &= D_t - D_{t+1} - \frac{1}{2}\|v_{t+1} - v_t\|^2. \end{aligned} \tag{21}$$

Our proof strategy is to upper and lower bound the inner product $\langle g_{t+1}, v_{t+1} - x_\star \rangle$. In particular, we will lower bound this inner product using (20) and upper bound it using (21). Towards this end, we define the point

$$\tilde{y}_t \coloneqq \frac{A_t}{A_{t+1}} x_t + \frac{a_{t+1}}{A_{t+1}} v_{t+1}.$$

We remark that the use of $\tilde{y}_t$ is inspired from the acceleration analysis of Allen-Zhu and Orecchia [2]. From the definition of y_t, we obtain

$$v_t = \frac{1}{a_{t+1}}(A_{t+1}y_t - A_t x_t) \quad \text{and} \quad v_t - v_{t+1} = \frac{A_{t+1}}{a_{t+1}}(y_t - \tilde{y}_t). \tag{22}$$

Recalling that $A_{t+1} = A_t + a_{t+1}$, we have

$$v_{t+1} = \frac{1}{a_{t+1}}(A_{t+1}\tilde{y}_t - A_t x_t) = \hat{x}_{t+1} + \frac{A_t}{a_{t+1}}(\hat{x}_{t+1} - x_t) - \frac{A_{t+1}}{a_{t+1}}(\hat{x}_{t+1} - \tilde{y}_t). \tag{23}$$

To begin our inner product lower bound, we note that

$$\begin{aligned}\langle g_{t+1}, v_{t+1} - x_\star\rangle &= \langle g_{t+1}, \hat{x}_{t+1} - x_\star\rangle + \frac{A_t}{a_{t+1}}\langle g_{t+1}, \hat{x}_{t+1} - x_t\rangle - \frac{A_{t+1}}{a_{t+1}}\langle g_{t+1}, \hat{x}_{t+1} - \tilde{y}_t\rangle \\ &\ge f(\hat{x}_{t+1}) - f(x_\star) + \frac{A_t}{a_{t+1}}\left[f(\hat{x}_{t+1}) - f(x_t)\right] - \frac{A_{t+1}}{a_{t+1}}\langle g_{t+1}, \hat{x}_{t+1} - \tilde{y}_t\rangle \\ &= \frac{A_{t+1}}{a_{t+1}}\left[f(\hat{x}_{t+1}) - f(x_\star)\right] - \frac{A_t}{a_{t+1}}\left[f(x_t) - f(x_\star)\right] - \frac{A_{t+1}}{a_{t+1}}\langle g_{t+1}, \hat{x}_{t+1} - \tilde{y}_t\rangle,\end{aligned} \tag{24}$$

where the inequality follows from (20). To relate $f(\hat{x}_{t+1})$ to $f(x_{t+1})$ in (24), we use the approximation guarantee (3) defining $\mathcal{O}_{\lambda,\delta}(\cdot)$ to obtain

$$\begin{aligned}f(\hat{x}_{t+1}) &\ge f(x_{t+1}) + \frac{\lambda_{t+1}}{2}\|x_{t+1} - y_t\|^2 - \frac{\lambda_{t+1}}{2}\|\hat{x}_{t+1} - y_t\|^2 - \frac{\lambda_{t+1}\delta_{t+1}^2}{2} \\ &\ge f(x_{t+1}) + \frac{\lambda_{t+1}}{2}\|x_{t+1} - y_t\|^2 + \langle g_{t+1}, \hat{x}_{t+1} - \tilde{y}_t\rangle - \frac{\lambda_{t+1}}{2}\|y_t - \tilde{y}_t\|^2 - \frac{\lambda_{t+1}\delta_{t+1}^2}{2}\end{aligned}$$

where we used $\langle g_{t+1}, \hat{x}_{t+1} - \tilde{y}_t\rangle = \frac{\lambda_{t+1}}{2}\|y_t - \tilde{y}_t\|^2 - \frac{\lambda_{t+1}}{2}\|\hat{x}_{t+1} - \tilde{y}_t\|^2 - \frac{\lambda_{t+1}}{2}\|\hat{x}_{t+1} - y_t\|^2$ (as in Lemma 12) to obtain the second inequality. Substituting into (24) and recalling that $\mathbb{E}_t = f(x_t) - f(x_\star)$ yields

$$\langle g_{t+1}, v_{t+1} - x_\star\rangle \ge \frac{A_{t+1}}{a_{t+1}}E_{t+1} - \frac{A_t}{a_{t+1}}E_t + \frac{A_{t+1}\lambda_{t+1}}{2a_{t+1}}\left[\|x_{t+1} - y_t\|^2 - \delta_{t+1}^2 - \|y_t - \tilde{y}_t\|^2\right].$$

The lower bound $\lambda_{t+1} \ge \frac{\epsilon}{3rR}$ implies that $\delta_{t+1} = \frac{\epsilon}{12\lambda_{t+1}R} \le \frac{r}{4}$. Moreover, by condition (b) ($\lambda_{t+1} \ge \frac{\epsilon}{3rR}$) and Proposition 17 we have that $\|\hat{x}_{t+1} - y_t\| \ge 3r/4$. Applying Lemma 11 we conclude that

$$\|x_{t+1} - y_t\|^2 - \delta_{t+1}^2 \ge (\|\hat{x}_{t+1} - y_t\| - \delta_{t+1})^2 - \delta_{t+1}^2 \ge r^2\left[(\tfrac{3}{4} - \tfrac{1}{4})^2 - (\tfrac{1}{4})^2\right] \ge \frac{r^2}{6}.$$

Substituting back, we have

$$\langle g_{t+1}, v_{t+1} - x_\star\rangle \ge \frac{A_{t+1}}{a_{t+1}}E_{t+1} + \frac{A_t}{a_{t+1}}E_t - \frac{A_{t+1}\lambda_{t+1}r^2}{12a_{t+1}} - \frac{A_{t+1}\lambda_{t+1}}{2a_{t+1}}\|y_t - \tilde{y}_t\|^2. \tag{25}$$

We now proceed to upper bound $\langle g_{t+1}, v_{t+1} - x_\star\rangle$. Recalling $g_{t+1} = \lambda_{t+1}(y_t - \hat{x}_{t+1})$ we obtain

$$\begin{aligned}\langle g_{t+1}, v_{t+1} - x_\star\rangle &= \lambda_{t+1}\langle y_t - x_{t+1}, v_{t+1} - x_\star\rangle + \lambda_{t+1}\langle x_{t+1} - \hat{x}_{t+1}, v_{t+1} - x_\star\rangle \\ &\le \lambda_{t+1}\langle y_t - x_{t+1}, v_{t+1} - x_\star\rangle + \lambda_{t+1}\|x_{t+1} - \hat{x}_{t+1}\|\|v_{t+1} - x_\star\|.\end{aligned} \tag{26}$$

To bound the term $\lambda_{t+1}\|x_{t+1} - \hat{x}_{t+1}\|\|v_{t+1} - x_\star\|$, note that $\|x_{t+1} - \hat{x}_{t+1}\| \le \delta_{t+1}$ by Lemma 11, $\|v_{t+1} - x_\star\| \le 2R$ since v_{t+1} and $x_\star$ are both in $\mathbb{B}_R(x_0)$ by assumption, and $\lambda_{t+1}\delta_{t+1}\cdot 2R = \frac{\epsilon}{6} \le \frac{\epsilon}{4}$. To bound the term $\lambda_{t+1}\langle y_t - x_{t+1}, v_{t+1} - x_\star\rangle$ we apply (21). Applying these bounds to (26) yields

$$\begin{aligned}\langle g_{t+1}, v_{t+1} - x_\star\rangle &\le \frac{1}{a_{t+1}}\left[D_t - D_{t+1} - \frac{1}{2}\|v_{t+1} - v_t\|^2 + \frac{a_{t+1}\epsilon}{4}\right] \\ &= \frac{1}{a_{t+1}}\left[D_t - D_{t+1} - \frac{A_{t+1}^2}{2a_{t+1}^2}\|y_t - \tilde{y}_t\|^2 + (A_{t+1} - A_t)\frac{\epsilon}{4}\right],\end{aligned} \tag{27}$$

where the equality is due to (22) and $a_{t+1} = A_{t+1} - A_t$.

Combining the lower and upper bound (25) and (27) and rearranging, we have

$$A_{t+1}(E_{t+1} - \tfrac{\epsilon}{4}) + D_{t+1} - A_t(E_{t+1} - \tfrac{\epsilon}{4}) - D_t \le -\frac{A_{t+1}\lambda_{t+1}r^2}{12} + \left(A_{t+1}\lambda_{t+1} - \frac{A_{t+1}^2}{a_{t+1}^2}\right)\frac{1}{2}\|y_t - \tilde{y}_t\|^2.$$

The proof is complete upon noticing that $A_{t+1}\lambda_{t+1} - \frac{A_{t+1}^2}{a_{t+1}^2} = 0$ (since $A_{t+1} = a_{t+1}^2\lambda_{t+1}$) and that $P_t = A_t\hat{E}_t + D_t = A_t(E_t - \frac{\epsilon}{4}) + D_t$. ■

Lemma 18 shows that the potential P_t decreases significantly whenever λ_{t+1} is not too small and $\|x_t - v_t\| \le 2R$ holds; we now the latter condition inductively.

Lemma 19 *Fix $t \ge 1$. In Algorithm 1 if $f(x_i) - f(x_\star) > \epsilon$ and $\lambda_i \ge \frac{\epsilon}{3rR}$ for all $0 \le i \le t$ then*

$$\|x_t - x_\star\| \le R, \ \|v_t - x_\star\| \le R, \ \textit{and} \ P_t - P_0 \le -\sum_{i\in[t-1]} \frac{A_{i+1}\lambda_{i+1}r^2}{12}.$$

Proof We proceed by induction on t. For the base case of $t = 0$, we note that $\|v_0 - x_\star\| = \|x_0 - x_\star\| \le R$ by assumption, and that $P_0 - P_0 \le 0$ trivially. Therefore we assume the inductive hypothesis that the lemma statement holds for iteration t, and show that it also holds for $t+1$. First, we note that $\|x_t - x_\star\| \le R$, $\|v_t - x_\star\| \le R$ and $\lambda_{t+1} \ge \frac{\epsilon}{3rR}$ satisfy the conditions of Lemma 18 and consequently $P_{t+1} \le P_t - \frac{1}{12}A_{t+1}\lambda_{t+1}r^2$; together with the inductive hypothesis this establishes

$$P_{t+1} - P_0 \le -\sum_{i=0}^{t} \frac{A_{i+1}r^2\lambda_{i+1}}{12}.$$

Next, we note that $f(x_{t+1}) - f(x_\star) > \epsilon$ implies that $\hat{E}_{t+1} = f(x_{t+1}) - f(x_\star) - \frac{\epsilon}{4} > 0$. Recalling the definition (19) and $A_0 = 0$, this implies

$$\frac{1}{2}\|v_t - x_\star\|^2 = D_t \le A_t\hat{E}_t + D_t = P_t \le P_0 = D_0 = \frac{1}{2}\|x_0 - x_\star\|^2 \le \frac{1}{2}R^2,$$

and consequently

$$\|v_{t+1} - x_\star\| \le R.$$

To complete the induction step we need to argue that $\|x_{t+1} - x_\star\| \le R$. To that end, we invoke Lemma 14 (recalling that $f(x_{t+1}) - f(x_\star) > \epsilon$ by assumption) which shows that $\|x_{t+1} - x_\star\| \le \|y_t - x_\star\|$ or $f(x_{t+1}) - f(x_\star) \le \frac{\epsilon}{2}$. The definition $y_t = \frac{A_t}{A_{t+1}}x_t + \frac{a_{t+1}}{A_{t+1}}v_t$ gives

$$\|x_{t+1} - x_\star\| \le \|y_t - x_\star\| \le \frac{A_t}{A_{t+1}}\|x_t - x_\star\| + \frac{a_{t+1}}{A_{t+1}}\|v_t - x_\star\| \le R,$$

where the final bounds holds since $A_{t+1} = A_t + a_{t+1}$ and $\|x_t - x_\star\|, \|v_t - x_\star\| \le R$ by the inductive assumption. ■

Lemma 20 *Fix $t \geq 1$. In Algorithm 1 if $f(x_i) - f(x_\star) > \epsilon$ and $\lambda_i \geq \frac{\epsilon}{3rR}$ for all $0 \leq i \leq t$ then*

$$A_t \geq \exp\left(\frac{r^{2/3}}{R^{2/3}}(t-1)\right) A_1 .$$

Proof Lemma 19 implies that

$$P_t - P_0 \leq -\sum_{i=0}^{t} \frac{A_{i+1} r^2 \lambda_{i+1}}{12} .$$

Further, that $f(x_i) - f(x_\star) \geq \epsilon$ for all $i \leq t$ implies $\hat{E}_t \geq 0$ and therefore $P_t \geq 0$. Combining these facts with the facts that $A_0 = 0$ and the A_i increase monotonically yields

$$\sum_{i=0}^{t-1} A_{i+1}\lambda_{i+1} \leq \frac{12}{r^2} P_0 = \frac{12}{r^2} D_0 \leq \frac{6R^2}{r^2}. \tag{28}$$

Next, note that the reverse Hölder inequality with $p = 2/3$ states that for any $u, v \in \mathbb{R}^d_{>0}$

$$\langle u, v\rangle \geq \left(\sum_{i\in[d]} u_i^{2/3}\right)^{3/2} \cdot \left(\sum_{i\in[d]} v_i^{-2}\right)^{-1/2} .$$

We therefore have

$$\begin{aligned}\sqrt{A_t} &\overset{(i)}{\geq} \frac{1}{2}\sum_{i\in[t]} \frac{1}{\sqrt{\lambda_i}} \overset{(ii)}{\geq} \frac{1}{2}\left(\sum_{i\in[t]}\left(\sqrt{A_i}\right)^{2/3}\right)^{3/2} \cdot \left(\sum_{i\in[t]}\left(\frac{1}{\sqrt{A_i\lambda_i}}\right)^{-2}\right)^{-1/2} \\ &\overset{(iii)}{\geq} \frac{1}{2}\left(\sum_{i\in[t]}\left(\sqrt{A_i}\right)^{2/3}\right)^{3/2} \cdot \frac{r}{\sqrt{6}R},\end{aligned}$$

where we used (i) Lemma 16, (ii) the reverse Hölder inequality with $u_i = \sqrt{A_i}$ and $v_i = 1/\sqrt{A_i\lambda_i}$, and (iii) the bound (28). Rearranging, we have

$$A_t^{1/3} \geq \frac{r^{2/3}}{3R^{2/3}}\left(\sum_{i\in[t]} A_i^{1/3}\right) . \tag{29}$$

Lemma 28 of [12] shows that for any sequence of A_t that satisfy (29) for all $t \in [T]$ also satisfies

$$A_T^{1/3} \geq \exp\left(\frac{r^{2/3}}{3R^{2/3}}(T-1)\right) A_1^{1/3}$$

and the result follows. ■

We are now ready to prove our main theorem.

Proof [Proof of Theorem 2] This proof proceeds in parts. First, we show that whenever the algorithm terminates, i.e. one of the conditions of Line 8 is met, then Algorithm 1 outputs a point x_{ret} with $f(x_{\text{ret}}) - f(x_\star) \leq \epsilon$ on Line 9 . Next, we bound T_o, the value of t at termination on Line 9, and show that $T_o = O\Big(\left(\frac{R}{r}\right)^{2/3} \log\Big(\frac{[f(x_0 - f(x_\star))]\cdot R^2}{\epsilon \cdot r^2}\Big)\Big)$. Leveraging these facts we complete the proof.

Termination due to small λ_{T_o+1}. Consider the case where the algorithm terminates because $\lambda_{T_o+1} < \frac{\epsilon}{3rR}$ on Line 8. Further, suppose that $f(x_i) - f(x_\star) > \epsilon$ for all $0 \le i \le T_o$ as otherwise $f(x_{\text{ret}}) - f(x_\star) \le \epsilon$ as desired. By definition of T_o, we have $\lambda_t \ge \frac{\epsilon}{3rR}$ for all $t \le T_o$ (otherwise we would have terminated earlier). Applying Lemma 19, we conclude it must be that $\|x_{T_o} - x_\star\| \le R$ and $\|v_{T_o} - x_\star\| \le R$ and therefore $\|y_{T_o} - x_\star\| \le R$. By Proposition 17 we have that $\hat{x}_{T_o+1} = \text{prox}^f_\lambda(y_{T_o}) = \text{bprox}^f_{\lambda,r}(y_{T_o})$ and thus $\|\hat{x}_{T_o+1} - y_{T_o}\| < r$. Lemma 13 with $x_0 = y_{T_o}$, $x_\lambda = \hat{x}_{T_o+1}$, $y = x_\star$ and $\Theta = r$ then yields

$$f(\hat{x}_{T_o+1}) - f(x_\star) \le \lambda_{t+1} rR < \frac{\epsilon}{3}.$$

Moreover, the BROO guarantee (3) gives

$$f(x_{T_o+1}) \le f(\hat{x}_{T_o+1}) + \frac{\lambda_{T_o+1}}{2}\left[\|\hat{x}_{T_o+1} - y_t\|^2 - \|x_{T_o+1} - y_t\|^2 + \delta^2_{T_o+1}\right] \le \frac{\lambda_{T_o+1}}{2}(r^2 + \delta^2_{T_o+1}).$$

Noting that $\lambda_{T_o+1} \in [\frac{\epsilon}{6rR}, \frac{\epsilon}{3rR}]$ and that $\delta_{T_o+1} = \frac{\epsilon}{12\lambda_{T_o+1}R}$, we have have $\lambda_{T_o+1}r^2 \le \epsilon \cdot \frac{r}{3R} \le \epsilon/3$ and $\lambda_{T_o+1}\delta^2_{T_o+1} \le \epsilon \cdot \frac{\epsilon}{12R} \cdot \frac{r}{2} \le \epsilon/24$. Substituting back, we have

$$f(x_{T_o+1}) - f(x_\star) \le f(\hat{x}_{T_o+1}) - f(x_\star) + \frac{9\epsilon}{49} \le \frac{\epsilon}{3} + \frac{9\epsilon}{49} \le \epsilon.$$

Termination due to large A_{T_o+1}. Next consider the case where $\lambda_{t+1} \ge \frac{\epsilon}{3rR}$ for all $t \le T_o$ but $A_{T_o+1} \ge \frac{R^2}{\epsilon}$. In this case, Lemma 19 implies that unless $f(x_{\text{ret}}) - f(x_\star) \le \epsilon$

$$A_{T_o+1}\left(f(x_{T_o+1}) - f(x_\star) - \frac{\epsilon}{4}\right) = A_{T_o+1}\hat{E}_{T_o+1} \le P_{T_o+1} \le P_0 = D_0 = \frac{1}{2}\|x_0 - x_\star\|^2 \le \frac{R^2}{2}.$$

Dividing by $A_{T_o+1} \ge \frac{R^2}{\epsilon}$, we obtain

$$f(x_{T_o+1}) - f(x_\star) \le \frac{\epsilon}{4} + \frac{\epsilon}{2} \le \epsilon.$$

Termination due to distant x_{T_o+1}, v_{T_o+1}. Next consider the case where $\lambda_{t+1} \ge \frac{\epsilon}{3rR}$ for all $t \le T_o$ but $\|x_{T_o+1} - v_{T_o+1}\| > 2R$. This implies that either $\|x_{T_o+1} - x_\star\| > R$ or $\|v_{T_o+1} - x_\star\| > R$. Consequently, Lemma 19 implies that $f(x_i) - f(x_\star) \le \epsilon$ for some $0 \le i \le T_o + 1$.

Termination due to slow A_t growth. Next consider the case where $\lambda_{t+1} \ge \frac{\epsilon}{3rR}$ for all $t \le T_o$ but $A_{T_o+1} < \exp\left(\frac{r^{2/3}}{R^{2/3}}(T_o - 1)\right) A_1$. In this case Lemma 20 implies that $f(x_i) - f(x_\star) \le \epsilon$ for some $0 \le i \le T_o + 1$ and therefore $f(x_{\text{ret}}) - f(x_\star) \le \epsilon$ as desired.

We conclude that in each of the four possible causes for the algorithm to terminate, i.e. one of the conditions in Line 8 to be true, the algorithm x_{ret} such that $f(x_{\text{ret}}) - f(x_\star) \le \epsilon$ as desired.

Iteration Bound. We now show that $T_o = O\left(\frac{R^{2/3}}{r^{2/3}} \log\left(\frac{[f(x_0) - f(x_\star)]R}{\epsilon r}\right)\right)$. Note that by definition of T_o as the iterate index t for which termination on Line 9 occurs, we have

$$\exp\left(\frac{r^{2/3}}{R^{2/3}} \cdot (T_o - 1)\right) A_1 \le A_{T_o} \le \frac{R^2}{\epsilon}$$

since otherwise we would have terminated on at $t = T_o - 1$. Consequently we have that

$$T_o = O\left(\frac{R^{2/3}}{r^{2/3}} \log\left(\frac{R^2}{A_1 \epsilon}\right)\right), \tag{30}$$

and it remains to lower bound $A_1 = a_1 = \frac{1}{2\lambda_1}$ (since $A_0 = 0$).

Note that since $x_0 = v_0$ it is the case that $y_0 = x_0$. Consequently, if $T_o > 0$, Line 8 implies that $\lambda_1 \geq \frac{\epsilon}{3rR}$, i.e., item (b) of Proposition 17 does not hold). Consequently, Proposition 17 implies that $\|\mathrm{prox}^f_{\lambda_1}(x_0) - x_0\| \in [\frac{3r}{4}, r]$ and therefore

$$f(x_0) \geq f\left(\mathrm{prox}^f_{\lambda_1}(x_0)\right) + \frac{\lambda}{2}\|\mathrm{prox}^f_{\lambda_1}(x_0) - x_0\|^2 \geq f\left(\mathrm{prox}^f_{\lambda_1}(x_0)\right) + \frac{9\lambda r^2}{32}.$$

Further, since $\|\mathrm{prox}^f_{\lambda_1}(x_0) - x_0\| \leq r \leq R$ we know that $f(\mathrm{prox}^f_{\lambda_1}(x_0)) \geq f(x_\star)$. Consequently,

$$\lambda_1 \leq \frac{32[f(x_0) - f(x_\star)]}{9r^2} \text{ and } A_1 = \frac{1}{\lambda_1} \geq \frac{9r^2}{32[f(x_0) - f(x_\star)]}.$$

Substituting back to (30) yields the claimed bound on T_o.

Remaining guarantees. To complete the proof, we observe that in each of the algorithm's

$$O\left(\left(\frac{R}{r}\right)^{2/3} \log\left(\frac{L_f R^2}{r\epsilon}\right)\right)$$

iterations, Proposition 17 guarantees that we perform at most $O(\log \frac{L_f R^2}{r\epsilon})$ queries to $\mathcal{O}_{\lambda,\delta}(\cdot)$. It also guarantees that the queried λ satisfies $\lambda \in [\frac{\epsilon}{24rR}, \frac{4L_f}{r}]$, yielding item 1 in Theorem 6. Finally, for the sequence $\lambda_{(1)}, \ldots, \lambda_{(T)}$ of λ values queried during the execution of Algorithm 1, we have

$$\sum_{i\in[T]} \frac{1}{\sqrt{\lambda_{(i)}}} = \sum_{t\in[T_o+1]} \sum_{\substack{i \text{ seen in line search} \\ \text{for iteration } t}} \frac{1}{\sqrt{\lambda_{(i)}}} \overset{(i)}{\leq} \sum_{i\in[T_o+1]} \frac{O(1)}{\sqrt{\lambda_t}} \log\left(\frac{L_f R^2}{r\epsilon}\right)$$
$$\overset{(ii)}{\leq} O\left(\sqrt{A_{T_o}} + \frac{1}{\sqrt{\lambda_{\min}}}\right) \log\left(\frac{L_f R^2}{r\epsilon}\right),$$

due to (i) Proposition 17 and (ii) Lemma 16 and $\lambda_{T_o+1} \geq \lambda_{\min} = \frac{\epsilon}{6rR}$ for all t. Finally, Line 8 guarantees $A_{T_o} < \frac{R^2}{\epsilon}$, and (since $r \leq R$) $\sqrt{A_{T_o}} + \frac{1}{\sqrt{\lambda_{\min}}} = O(\frac{R}{\sqrt{\epsilon}})$, giving item 2 in Theorem 6. ■

B.3. Bisection analysis

In this section we prove Proposition 17, a guarantee on our bisection subroutine, restated below.

Proposition 17 (Bisection) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be L_f-Lipschitz and convex, and let $x, v \in \mathbb{R}^d$, $\epsilon, r, R \in \mathbb{R}_{>0}$ satisfy $\epsilon \leq L_f R$, $r \leq R$ and $\|x - v\| \leq 2R$. Given $\lambda_{\max} \geq \frac{2L_f}{r}$ and $\lambda_{\min} \in (0, \lambda_{\max})$, λ-BISECTION(x, v, A) outputs $\lambda \in [\lambda_{\min}, \lambda_{\max}]$ such that*

$$\mathrm{prox}^f_\lambda(y_\lambda) = \mathrm{bprox}^f_{\lambda,r}(y_\lambda)$$

(i.e., $\|\mathrm{prox}_{\lambda}^{f}(y_\lambda)-y_\lambda\| \le r$). The subroutine uses $O(\log(\frac{\lambda_{\max}}{\lambda_{\min}})+\log(\frac{R+L_f/\lambda_{\min}}{r}))$ calls to $\mathcal{O}_{\lambda',\frac{r}{17}}(\cdot)$ with $\lambda' \in [\frac{1}{2}\lambda, \lambda_{\max}]$. Moreover, for $\alpha_{2\lambda A} = \frac{2\lambda A}{1+2\lambda A+\sqrt{1+4\lambda A}}$ and $y_\lambda \coloneqq \alpha_{2\lambda A}x + (1-\alpha_{2\lambda A})v$ one of the following outcomes must occur:

(a) $\lambda \in [2\lambda_{\min}, \lambda_{\max}]$ *and* $\|\mathrm{prox}_{\lambda}^{f}(y_\lambda) - y_\lambda\| > \frac{3r}{4}$, *or*

(b) $\lambda < 2\lambda_{\min}$.

When taking $\lambda_{\max} = \frac{2L_f}{\epsilon}$, $\lambda_{\min} = \frac{\epsilon}{6rR}$, the number of calls to $\mathcal{O}_{\lambda',\frac{r}{17}}(\cdot)$ is bounded by $O(\log\frac{L_f R^2}{r\epsilon})$.

B.3.1. PRELIMINARIES AND DISCUSSION

Notation. To analyze the bisection procedure, we use the following functions of $\lambda \ge 0$. Fixing $x, v \in \mathbb{R}^d$ and $A \ge 0$, we define

$$y_\lambda \coloneqq \alpha_{2A\lambda'} \cdot x + (1-\alpha_{2A\lambda'})\cdot, \quad \text{where} \quad \alpha_\tau \coloneqq \frac{\tau}{1+\tau+\sqrt{1+2\tau}}.$$

We also define

$$\hat{x}_\lambda \coloneqq \mathrm{prox}_{\lambda}^{f}(y_\lambda) = \operatorname*{arg\,min}_{x\in\mathbb{R}^d}\left\{ f(x) + \frac{\lambda}{2}\|x-y_\lambda\|^2 \right\}$$

and

$$\hat{\Delta}(\lambda) \coloneqq \|\hat{x}_\lambda - y_\lambda\|. \tag{31}$$

We approximate $\hat{x}_\lambda$ using the BROO output $\mathcal{O}_{\lambda,\delta}(y_\lambda)$ with $\delta = r/17$, and write

$$\Delta(\lambda) \coloneqq \|\mathcal{O}_{\lambda,\delta}(y_\lambda) - y_\lambda\|$$

for our approximation to $\hat{\Delta}(\lambda)$. Note that, depending on the BROO implementation, $\Delta(\lambda)$ need not be deterministic. Our bisection procedure implicitly assumes that the BROO is called only once per input x, v, A and distinct value of λ, and that subsequent references to $\Delta(\lambda)$ use cached values of $\mathcal{O}_{\lambda,\delta}(y_\lambda)$.

Algorithm overview. The goal of λ-BISECTION is to find a value of λ where $\hat{\Delta}(\lambda) < r$ so that $\hat{x}_\lambda = \mathrm{prox}_{\lambda}^{f}(y_\lambda) = \mathrm{bprox}_{\lambda,r}^{f}(y_\lambda)$ is well-approximated by the BROO output $\mathcal{O}_{\lambda,\delta}(y_\lambda)$. In addition, the bisection has to guarantee that either

- $\hat{\Delta}(\lambda) \ge 3r/4$, which implies that the outer loop of Algorithm 1 makes sufficient progress (see Lemma 18), or
- $\lambda < 2\lambda_{\min}$, which implies that $\hat{x}_\lambda$ is near-optimal (as we argue in the proof of Theorem 6).

Our procedure (given in Algorithm 1) starts with $\lambda = \lambda_{\max}$ sufficiently large to guarantee $\hat{\Delta}(\lambda) < r$. It then iteratively halves λ until finding λ_0 such that $\Delta(\lambda_0) > \frac{13r}{16}$ or $\lambda_0 < \lambda_{\min}$. In the latter case it returns $2\lambda_0 < 2\lambda_{\min}$ and we note that $\Delta(2\lambda_0) \le \frac{13r}{16}$ implies $\hat{\Delta}(2\lambda_0) < r$. In the former case we perform a binary search in the interval $[\lambda_0, 2\lambda_0]$ until we find λ_m such that $\Delta(\lambda_m) \in [\frac{13r}{16}, \frac{15r}{16}]$, which implies $\hat{\Delta}(\lambda_m) \in [3r/4, r)$.

Comparison to the bisection in [12]. Our bisection procedure essentially attempts to find λ such that $\hat{\Delta}(\lambda)$ is close to (but smaller than) r. In contrast, Carmon et al. [12] use the implicit relation $y_\lambda - \hat{x}_\lambda = \frac{1}{\lambda}\nabla f(\hat{x}_\lambda)$ and attempt to find λ such that $\|\nabla f(\hat{x}_\lambda)\|/\lambda$ is close to r. Consequently, the iteration count bounds on the bisection of [12] depend on the continuity ∇f, whereas our bisection succeeds even when f is non-smooth and hence ∇f is discontinuous. The key to this improvement is a careful analysis of the continuity of $\hat{\Delta}(\lambda)$ (see the following subsection). Another novel aspect of our procedure is the two-stage structure where we first iteratively halve λ and only then perform a binary search. This structure allows us to guarantee that we never query the BROO with λ' that is much smaller than the λ we eventually output. This guarantee is necessary for proving statement 2 in Theorem 6, which in turn is necessary for establishing an optimal complexity bound in the weakly-smooth regime $L_g = \Theta(L_f^2/\epsilon)$.

B.3.2. CONTINUITY ANALYSIS OF $\hat{\Delta}(\cdot)$

We begin by proving a bound on the Jacobian of $\delta_\lambda = \hat{x}_\lambda - y_\lambda$ with respect to λ (Lemma 21) under the assumption that f is twice differentiable. We subsequently remove this assumption via a smoothing argument (Corollary 22).

Lemma 21 *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and twice differentiable, let $y_\lambda \in \mathbb{R}^d$ be a differentiable function of $\lambda > 0$, $\hat{x}_\lambda = \mathrm{prox}_\lambda^f(y_\lambda)$, and $\delta_\lambda = \hat{x}_\lambda - y_\lambda$. Then for all $\lambda > 0$ we have*

$$\left\|\frac{d}{d\lambda}\delta_\lambda\right\| \le \left\|\frac{d}{d\lambda}y_\lambda\right\| + \frac{1}{\lambda}\left\|\delta_\lambda\right\|.$$

Proof Note that $\nabla f(\hat{x}_\lambda) + \lambda(\hat{x}_\lambda - y_\lambda) = 0$. Differentiating yields

$$\nabla^2 f(\hat{x}_\lambda) \cdot \frac{d}{d\lambda}\hat{x}_\lambda + (\hat{x}_\lambda - y_\lambda) + \lambda\left(\frac{d}{d\lambda}\hat{x}_\lambda - \frac{d}{d\lambda}y_\lambda\right) = 0\,.$$

Rearranging the terms, we obtain

$$\left(\nabla^2 f(\hat{x}_\lambda) + \lambda I\right) \cdot \left(\frac{d}{d\lambda}\hat{x}_\lambda - \frac{d}{d\lambda}y_\lambda\right) = y_\lambda - \hat{x}_\lambda - \nabla^2 f(\hat{x}_\lambda)\frac{d}{d\lambda}y_\lambda\,.$$

Since f is convex, $\nabla^2 f(\hat{x}_\lambda)$ is PSD and therefore

$$\left(\frac{d}{d\lambda}\hat{x}_\lambda - \frac{d}{d\lambda}y_\lambda\right) = \left(\nabla^2 f(\hat{x}_\lambda) + \lambda I\right)^{-1} \cdot \left(y_\lambda - \hat{x}_\lambda - \nabla^2 f(\hat{x}_\lambda) \cdot \frac{d}{d\lambda}y_\lambda\right)\,.$$

Note that $\left(\nabla^2 f(\hat{x}_\lambda) + \lambda I\right)^{-1}\nabla^2 f(\hat{x}_\lambda)$ is a symmetric PSD matrix with all eigenvalues $\in [0,1]$ and $\left(\nabla^2 f(\hat{x}_\lambda) + \lambda I\right)^{-1}$ is a symmetric PSD matrix with all eigenvalues at most $1/\lambda$. Consequently, $\|\left(\nabla^2 f(\hat{x}_\lambda) + \lambda I\right)^{-1}\nabla^2 f(x)\| \le 1$ and $\|\left(\nabla^2 f(\hat{x}_\lambda) + \lambda I\right)^{-1}\| \le \frac{1}{\lambda}$ yielding the claim. ■

We now relax the assumption of twice differentiability.

Corollary 22 *Let $f : \mathbb{R}^d \to \mathbb{R}$ be L_f-Lipschitz and convex, let $y_\lambda \in \mathbb{R}^d$ be a differentiable function of $\lambda > 0$, $\hat{x}_\lambda = \mathrm{prox}_\lambda^f(y_\lambda)$, and $\delta_\lambda = \hat{x}_\lambda - y_\lambda$. Then for all $\lambda_1, \lambda_2 > 0$ we have*

$$\left\|\delta_{\lambda_2} - \delta_{\lambda_1}\right\| \le \int_{\lambda=\lambda_1}^{\lambda_2}\left\|\frac{d}{d\lambda}y_\lambda\right\|d\lambda + \int_{\lambda=\lambda_1}^{\lambda_2}\frac{1}{\lambda}\left\|\delta_\lambda\right\|d\lambda\,.$$

Proof For $\sigma > 0$, let $\nu = \mathcal{N}(0; \sigma^2 I_{d\times d})$ and let $f_\sigma(x) = \mathbb{E}f(x+\nu)$. Then f_σ is convex, infinitely differentiable and satisfies $0 \le f_\sigma(x) - f(x) \le L_f \mathbb{E}\|\nu\| \le L_f\sqrt{d}\sigma$ for all $x \in \mathbb{R}^d$. Thus, applying Lemma 21 for f_σ and $\delta^\sigma_\lambda = \mathrm{prox}^{f_\sigma}_\lambda(y_\lambda) - y_\lambda$, we have

$$\left\|\delta^\sigma_{\lambda_2} - \delta^\sigma_{\lambda_1}\right\| \le \int_{\lambda=\lambda_1}^{\lambda_2} \left\|\frac{d}{d\lambda} y_\lambda\right\| d\lambda + \int_{\lambda=\lambda_1}^{\lambda_2} \frac{1}{\lambda}\left\|\delta^\sigma_\lambda\right\| d\lambda.$$

Noting that $\mathrm{prox}^{f_\sigma}_\lambda(y_\lambda)$ is at most $L_f\sqrt{d}\sigma$-suboptimal for $f(x) + \frac{\lambda}{2}\|x - y_\lambda\|^2$, we have

$$\|\delta_\lambda - \delta^\sigma_\lambda\| = \|\mathrm{prox}^f_\lambda(y_\lambda) - \mathrm{prox}^{f_\sigma}_\lambda(y_\lambda)\| \le \sqrt{\frac{2L_f\sqrt{d}\sigma}{\lambda}}$$

by λ-strong-convexity of $x \mapsto f(x) + \frac{\lambda}{2}\|x - y_\lambda\|^2$. Substituting back, we find that

$$\left\|\delta_{\lambda_2} - \delta_{\lambda_1}\right\| \le \int_{\lambda=\lambda_1}^{\lambda_2} \left\|\frac{d}{d\lambda} y_\lambda\right\| d\lambda + \int_{\lambda=\lambda_1}^{\lambda_2} \frac{1}{\lambda}\left\|\delta_\lambda\right\| d\lambda + \sqrt{\sigma}\cdot\frac{5\lambda_2 L_f^{1/2} d^{1/4}}{\lambda_1^{3/2}}.$$

For all $\sigma > 0$. Taking the limit $\sigma \to 0$ concludes the proof. ■

With this, we prove our desired bound on the continuity of $\hat{\Delta}(\lambda)$.

Lemma 23 *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and L_f-Lipschitz, and for $R > 0$ let $x, v \in \mathbb{R}^d$ such that $\|x - v\| \le 2R$. For any $0 < \lambda_1 \le \lambda_2$, the function $\hat{\Delta}(\lambda)$ defined in eq. (31) satisfies*

$$|\hat{\Delta}(\lambda_1) - \hat{\Delta}(\lambda_2)| \le \left(R + \frac{L_f}{\lambda_1}\right)\log\frac{\lambda_2}{\lambda_1}.$$

Proof Note that for all $t > 0$

$$\begin{aligned}\frac{d}{dt}\alpha_t &= \frac{1}{1+t+\sqrt{1+2t}} - \frac{t + t(1+2t)^{-1/2}}{(1+t+\sqrt{1+2t})^2} = \frac{(1+t)\sqrt{1+2t} + 1 + 2t - t\sqrt{1+2t} - t}{(1+t+\sqrt{1+2t})^2\sqrt{1+2t}} \\ &= \frac{1}{(1+t+\sqrt{1+2t})\sqrt{1+2t}} \in \left[0, \frac{1}{2t}\right]\end{aligned}$$

Consequently,

$$|\hat{\Delta}(\lambda_1) - \hat{\Delta}(\lambda_2)| \le \left\|\frac{d}{d\lambda}y_\lambda\right\| = \left\|(v - x)\frac{d}{d\lambda}\alpha_{2A\lambda}\right\| \le \frac{2A}{4A\lambda}\|v - x\| = \frac{1}{2\lambda}\|v - x\| \le \frac{R}{\lambda}.$$

By Lemma 21 we have

$$\left\|\delta_{\lambda_2} - \delta_{\lambda_1}\right\| \le \int_{\lambda=\lambda_1}^{\lambda_2} \left\|\frac{d}{d\lambda} y_\lambda\right\| d\lambda + \int_{\lambda=\lambda_1}^{\lambda_2} \frac{1}{\lambda}\left\|\delta_\lambda\right\| d\lambda.$$

We also have $\|\delta_\lambda\| \le \frac{L_f}{\lambda}$ from Lemma 15. Substituting back and using $\lambda \ge \lambda_1$, we obtain

$$\left\|\delta_{\lambda_2} - \delta_{\lambda_1}\right\| \le (R + \frac{L_f}{\lambda})\int_{\lambda_1}^{\lambda_2}\frac{d\lambda}{\lambda} \le \left(R + \frac{L_f}{\lambda_1}\right)\log\frac{\lambda_2}{\lambda_1}$$

as claimed. ■

Proof [Proof of Proposition 17] We first prove the correctness of our procedure, recalling the notation $\hat{x}_\lambda = \mathrm{prox}^f_\lambda(y_\lambda)$, $\hat{\Delta}(\lambda) = \|\hat{x}_\lambda - y_\lambda\|$ and $\Delta(\lambda) = \|\mathcal{O}_{\lambda,\delta}(y_\lambda) - y_\lambda\|$, where $\delta = \frac{r}{17}$ throughout. Our analysis will use the following fact: whenever $\Delta(\lambda) < r - \delta$ then (by Lemma 11) we have $\|\mathrm{bprox}^f_{\lambda,r}(y_\lambda) - y_\lambda\| < r$ and consequently $\mathrm{bprox}^f_{\lambda,r}(y_\lambda) = \mathrm{prox}^f_\lambda(y_\lambda) = \hat{x}_\lambda$ and $|\hat{\Delta}(\lambda) - \Delta(\lambda)| \le \delta$.

By $\lambda_{\max} \ge \frac{2L_f}{r}$ and Lemma 15, we have $\hat{\Delta}(\lambda_{\max}) \le r/2$ and $\mathrm{bprox}^f_{\lambda_{\max},r}(y_\lambda) = \mathrm{prox}^f_{\lambda_{\max}}(y_\lambda)$. By Lemma 11 we have $\|\mathcal{O}_{\lambda_{\max},\delta}(y_{\lambda_{\max}}) - \hat{x}_{\lambda_{\max}}\| \le \delta = r/17$, and consequently $\Delta(\lambda_{\max}) \le r/2 + r/16 \le 13r/16$. Therefore, the first while loop (Line 14) executes at least once.

Suppose the procedure terminates on Line 15. Denoting the return value as λ, by the condition on Line 14 we have $\Delta(\lambda) \le \frac{7r}{8}$, which by Lemma 11 implies $\hat{\Delta}(\lambda) < \frac{15r}{16} < r$. Consequently, outcome (b) in Proposition 17 occurs.

Next, consider the case where the procedure terminates due to the condition $\Delta(\lambda) \in [\frac{13r}{16}, \frac{15r}{16}]$ (either in line 17 or in line 18). Applying Lemma 11, we conclude that $\hat{\Delta}(\lambda) \in [\frac{3r}{4}, r)$. Consequently, outcome (a) in Proposition 17 occurs.

It remains to check the case where the procedure terminates due to the condition $\log\frac{\lambda_u}{\lambda_\ell} < \frac{r}{8(R+L_f/\lambda_\ell)}$ in line 18. Note that the binary search maintains the invariant $\Delta(\lambda_\ell) > \frac{15r}{16}$ and $\Delta(\lambda_u) < \frac{13r}{16}$. By Lemma 11, this implies

$$\hat{\Delta}(\lambda_\ell) > \frac{7r}{8} > \hat{\Delta}(\lambda_u).$$

By continuity of $\hat{\Delta}(\lambda)$, we therefore always have $\hat{\Delta}(\lambda') = \frac{7r}{8}$ for some $\lambda' \in (\lambda_\ell, \lambda_u)$. Therefore, if $\|x - v\| \le 2R$, Lemma 23 guarantees that

$$\left|\hat{\Delta}(\lambda_m) - \tfrac{7}{8}r\right| = \left|\hat{\Delta}(\lambda_m) - \hat{\Delta}(\lambda')\right| \le \left(R + \frac{L_f}{\min\{\lambda', \lambda_m\}}\right)\log\left|\frac{\lambda_m}{\lambda'}\right| \le \frac{1}{2}\left(R + \frac{L_f}{\lambda_\ell}\right)\log\frac{\lambda_u}{\lambda_\ell}$$

Consequently, when $\log\frac{\lambda_u}{\lambda_\ell} < \frac{r}{8(R+L_f/\lambda_\ell)}$ and $\|x - v\| \le 2R$, we are guaranteed that $\hat{\Delta}(\lambda_m) \in (3r/4, r)$, and outcome (a) in Proposition 17 occurs, concluding the proof of correctness.

Next, we briefly justify the bounds on the λ' values with which we query the BROO in the λ-BISECTION procedure. By construction, we have $\lambda' \in [\lambda_{\min}, \lambda_{\max}]$. Let λ_o and λ_s be the procedure's output and smallest queried λ values, respectively. When terminating on lines 15 or 17, we clearly have $\lambda_s = \lambda_o$. Furthermore, when terminating on line 18 λ_s equals λ_ℓ when entering line 18 for the first time, and consequently $\lambda_o \le 2\lambda_s$.

Finally, we bound the total number of BROO queries. The first while loop requires at most $\log_2\frac{\lambda_{\max}}{\lambda_{\min}}$ queries. In the second while loop, initially we have $\log_2\frac{\lambda_u}{\lambda_\ell} = 1$ and each query decreases $\log_2\frac{\lambda_u}{\lambda_\ell}$ by a factor of 2. Therefore, using $\lambda_\ell \ge \lambda_{\min}$, the stopping condition $\log\frac{\lambda_u}{\lambda_\ell} < \frac{r}{8(R+L_f/\lambda_\ell)}$ must hold after $O\left(\log\left(\frac{R+L_f/\lambda_{\min}}{r}\right)\right)$ queries. Given $\lambda_{\max} = \frac{2L_f}{r}$, $\lambda_{\min} = \frac{\epsilon}{6rR}$ the total number of BROO queries is bounded as

$$\log\left(\frac{\lambda_{\max}}{\lambda_{\min}}\right) + \log\left(\frac{R + L_f/\lambda_{\min}}{r}\right) = O\left(\log\frac{L_f R^2}{r\epsilon} + \log\frac{L_f R^2}{r\epsilon}\right) = O\left(\log\frac{L_f R^2}{r\epsilon}\right)$$

queries, giving the claimed complexity bound. ■

Appendix C. BROO implementation

C.1. Proof of Lemma 3

Recall that the "eponentiated softmax" function is defined as follows

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

and note that $\Gamma_{\epsilon,\lambda}(\bar{x}) = \epsilon'$.

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

Proof For the first statement, we note that $\Gamma_{\epsilon,\lambda}$ is a monotonic increasing transformation of $F^{\lambda}_{\mathrm{smax},\epsilon}$ and consequently they have the same minimizer $x_\star$ in $\mathbb{B}_r(\bar{x})$. Further

$$\begin{aligned} F^{\lambda}_{\mathrm{smax},\epsilon}(x) - F^{\lambda}_{\mathrm{smax},\epsilon}(x_\star) &= \epsilon' \log\left(\frac{\Gamma_{\epsilon,\lambda}(x)}{\Gamma_{\epsilon,\lambda}(x_\star)}\right) = \Gamma_{\epsilon,\lambda}(\bar{x}) \log\left(1 + \frac{\Gamma_{\epsilon,\lambda}(x) - \Gamma_{\epsilon,\lambda}(x_\star)}{\Gamma_{\epsilon,\lambda}(x_\star)}\right) \\ &\le \frac{\Gamma_{\epsilon,\lambda}(\bar{x})}{\Gamma_{\epsilon,\lambda}(x_\star)}(\Gamma_{\epsilon,\lambda}(x) - \Gamma_{\epsilon,\lambda}(x_\star)), \end{aligned}$$

where the final inequality uses $\log(1 + x) \le x$. Next, note that Lipschitz continuity of each f_i implies

$$f_i^{\lambda}(x_\star) \ge f_i(x_\star) \ge f_i(\bar{x}) - L_f\|x_\star - \bar{x}\| = f_i^{\lambda}(\bar{x}) - L_f\|x_\star - \bar{x}\|.$$

Substituting $\|x_\star - \bar{x}\| \le r \le c\epsilon'/L_f$ and $C \ge e^c$, we have that

$$e^{f_i^{\lambda}(x_\star)/\epsilon'} \ge e^{f_i^{\lambda}(\bar{x})/\epsilon' - c} \ge \frac{1}{C} e^{f_i^{\lambda}(\bar{x})}.$$

Consequently, we have

$$\frac{\Gamma_{\epsilon,\lambda}(\bar{x})}{\Gamma_{\epsilon,\lambda}(x_\star)} = \frac{\sum_{i\in[N]} e^{f_i^{\lambda}(\bar{x})/\epsilon'}}{\sum_{i\in[N]} e^{f_i^{\lambda}(x_\star)/\epsilon'}} \le C$$

which proves the first statement.

For the second statement, we first compute the gradient and Hessian of the function $\gamma_i(x)$ as

$$\nabla\gamma_i(x) = \exp\left(\frac{\lambda}{2\epsilon'}\|x - \bar{x}\|^2\right) \exp\left(\frac{f_i(x) - f_i(\bar{x})}{\epsilon'}\right) \left[\nabla f_i(x) + \lambda(x - \bar{x})\right], \text{ and}$$

$$\nabla^2\gamma_i(x) = \exp\left(\frac{f_i(x) - f_i(\bar{x})}{\epsilon'}\right) H_i, \text{ where}$$

$$H_i = \nabla^2 f_i(x) + \lambda I + \frac{1}{\epsilon'}\nabla f_i(x)\nabla f_i(x)^\top + \frac{\lambda^2}{\epsilon'}(x - \bar{x})(x - \bar{x})^\top.$$

(Note that the expression for $\nabla^2\gamma_i$ assumes that f_i is twice differentiable almost everywhere; we only rely on it for Section 4.3 where this holds.)

Now we note that for any $i \in [N]$

$$\|\nabla\gamma_i(x)\| \le \exp(\lambda r^2/2\epsilon' + L_f r/\epsilon')(\lambda r + L_f),$$

and

$$\lambda e^{-\lambda r^2/2\epsilon' - L_f r/\epsilon'} I \preceq \nabla^2\gamma_i(x) \preceq e^{\lambda r^2/2\epsilon' + L_f r/\epsilon'}\left(L_g + \lambda + L_f^2/\epsilon' + \lambda^2 r^2/\epsilon'\right) I,$$

where for the inequality we use the fact that f_i is L_f-Lipschitz, L_g-smooth, and that $x \in \mathbb{B}_r(\bar{x})$.

Now by plugging in the assumption that $r \le c\epsilon'/L_f$ and $\lambda \le cL_f/r$ we have $\lambda r \le cL_f$, $\lambda^2 r^2/\epsilon' \le c^2\lambda$, and $\exp(\lambda r^2/2\epsilon' + L_f r/\epsilon') \le \exp(c + c^2/2)$. Thus we can further simplify the bounds on the gradient and Hessian of $\gamma_i(x)$ by definition of C as

$$\|\nabla\gamma_i(x)\| \le CL_f \quad \text{and} \quad C^{-1}\lambda I \preceq \nabla^2\gamma_i(x) \preceq C\left(L_g + \lambda + L_f^2/\epsilon'\right) I,$$

which completes the proof. ■

C.2. SGD implementation

We first cite the following stochastic gradient method with restarts that obtains an $\widetilde{O}(1/\mu T)$ bound for μ-strongly convex function.

Lemma 24 (Theorem 11 in Hazan and Kale [20]) *Given a μ-strongly-convex objective function $f : \mathcal{X} \to \mathbb{R}$ with minimizer $x_\star$ with an unbiased stochastic estimator with norm at most G in the convex compact set $\mathcal{X}$, Epoch-SGD-Proj algrotihm finds an approximate minimizer $\tilde{x}$ satisfying with probability $1-\sigma$*

$$f(\tilde{x}) - f(x_\star) \le O\left(\frac{G^2\log(\log(T)/\sigma)}{\mu T}\right),$$

using T stochastic gradient queries.

Applying the lemma immediately gives the following guarantee for minimizing $\Gamma_{\epsilon,\lambda}$ inside a ball of radius r_ϵ and hence implementing an r_ϵ-BROO for $F_{\mathrm{smax},\epsilon}$.

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 $\lambda \le O(L_f/r_\epsilon)$, 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*

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

Proof We first note that by choice of r_ϵ and bounds on λ, $\Gamma_{\epsilon,\lambda}$ is $\Omega(\lambda)$-strongly convex according to Lemma 3. At each iteration, we sample $i \in [N]$ with probability p_i and compute stochastic gradient

$$\nabla\gamma_i(x) = \gamma_i(x)\left(\frac{\lambda}{\epsilon'}(x-\bar{x}) + \nabla f_i(x)/\epsilon'\right) \quad \text{bounded by} \quad G = O(L_f)$$

Algorithm 2: Epoch-SGD-Proj on the exponentiated softmax

```
Input: Functions $f_1, \dots, f_N$, ball center $\bar{x}$, ball radius $r_\epsilon$, regularization strength $\lambda$, smoothing
       parameter $\epsilon'$, failure probability $\sigma$
Parameters: Step size $\eta_1 = 1/(3\lambda)$, domain size $D_1 = \Theta(G\sqrt{\log(\log(T)/\delta)}/\lambda)$, $T_1 = 450$
            and total iteration budget $T$
Output: Approximate minimizer of $\Gamma_{\epsilon,\lambda}$ (and hence $F^{\lambda}_{\mathrm{smax},\epsilon}$) in $\mathbb{B}_{r_\epsilon}(\bar{x})$)
1  Precompute sampling probabilities $p_i = e^{f_i(\bar{x})/\epsilon'} / \sum_{i\in[N]} e^{f_i(\bar{x})/\epsilon'}$ for all $i \in [N]$
2  Initialize $x_1^1 \in \mathbb{B}_{r_\epsilon}(\bar{x})$ arbitrarily, set $k = 1$
3  while $\sum_{i\in[k]} T_i \le T$ do
4      for $t = 1, \dots, T_k$ do
5          Sample $i \in [N]$ with probability $p_i$
6          Query stochastic gradient $\hat{g}_t = e^{(f_i^\lambda(x) - f_i^\lambda(\bar{x}))/\epsilon'} \nabla f_i^\lambda(x)$
7          Update $x_{t+1}^k \leftarrow \Pi_{\mathbb{B}_{r_\epsilon}(\bar{x}) \cap \mathbb{B}_{D_k}(x_1^k)}(x_t^k - \eta_k \hat{g}_t)$
8      Let $x_1^{k+1} \leftarrow \frac{1}{T_k} \sum_{t\in[T_k]} x_t^k$
9      Update parameters $T_{k+1} \leftarrow 2T_k$, $\eta_{k+1} \leftarrow \eta_k/2$, $D_{k+1} \leftarrow D_k/\sqrt{2}$, $k \leftarrow k+1$
10 return $x_1^k$
```

following from the second statement of Lemma 3. Thus by directly applying Lemma 24 with $T = \Theta(L_f^2 \lambda^{-2} \delta^{-2} \log(\log(L_f/\lambda\delta)/\sigma))$ the algorithm outputs an approximate minimizer $\tilde{x}$ satisfying

$$\Gamma_{\epsilon,\lambda}(\tilde{x}) - \min_{x \in \mathbb{B}_{r_\epsilon}(\bar{x})} \Gamma_{\epsilon,\lambda}(x) \le O\left(\frac{L_f^2}{\lambda T} \log(\log(T)/\sigma)\right) \le \frac{\lambda\delta^2}{6e^2}.$$

By the first property of Lemma 3, bound above implies that for $x_\star = \mathrm{bprox}_{\lambda,r}^{F_{\mathrm{smax},\epsilon}}$

$$F^{\lambda}_{\mathrm{smax},\epsilon}(x) - F^{\lambda}_{\mathrm{smax},\epsilon}(x_\star) \le 3e^2(\Gamma_{\epsilon,\lambda}(x) - \Gamma_{\epsilon,\lambda}(x_\star)) \le \frac{\lambda\delta^2}{2},$$

i.e., algorithm outputs a valid r_ϵ-BROO response for $F_{\mathrm{smax},\epsilon}$.

The bound on the total computational cost follows from noticing that the initialization cost is dominated by N function value queries, and that the cost of each step in the stochastic gradient descent is dominated by a function and a gradient query at the current iteration. ■

C.3. Accelerated variance reduction implementation

We first cite the following accelerated variance reduction guarantee.

Lemma 25 (Theorem 5.4 in Allen-Zhu [1]) *Let $f_1, \dots, f_N$ be L-smooth and μ-strongly-convex, let $F(x) = \sum_{i\in[n]} w_i f_i(x)$ with $w_i \ge 0$ and $\sum_{i\in[N]} w_i = 1$, and let $x_\star \in \arg\min F(x)$. For any $s \in \mathbb{N}$, Katyusha1 with batch size $b = 1$ and initial point $\bar{x}$ finds an approximate solution $\tilde{x}_s$ satisfying*

$$\mathbb{E}[F(\tilde{x}_s) - F(x_\star)] \le \frac{1}{2^s}\left[F(\bar{x}) - F(x_\star)\right]$$

using

$$O\left(s\cdot\left(N+\sqrt{\frac{N\cdot L}{\mu}}\right)\right) \text{ evaluations of } \nabla f_i(x).$$

An immediate corollary of Lemma 25 provides a high-probability guarantee.

Corollary 26 *Under the same assumptions as Lemma 25, for any $\epsilon > 0$ and $\sigma \in (0,1)$ Katyusha1 with batch size $b = 1$ and initial point $\bar{x}$ finds an approximate solution $\tilde{x}$ satisfying*

$$F(\tilde{x}) - F(x_\star) \le \epsilon \text{ with probability at least } 1-\sigma$$

using

$$O\left(\left(N+\sqrt{\frac{N\cdot L}{\mu}}\right)\log\left(\frac{F(\bar{x})-F(x_\star)}{\varepsilon\sigma}\right)\right) \text{ evaluations of } \nabla f_i(x).$$

Proof Since $F(\tilde{x}_s) - F(x_\star) \ge 0$, Markov's inequality and Lemma 25 imply that for any $\epsilon > 0$,

$$\mathbb{P}(F(\tilde{x}_s) - F(x_\star) \ge \epsilon) \le \frac{\mathbb{E}\left[F(\tilde{x}_s) - F(x_\star)\right]}{\epsilon} \le \frac{(1/2)^s\left(f(\bar{x}) - f(x_\star)\right)}{\epsilon},$$

and thus the high-probability bound follows . ■

Specializing Corollary 26 for $\Gamma_{\epsilon,\lambda}(x) = \sum_{i\in[N]} p_i(\bar{x})\gamma_i(x)$, we obtain the following guarantee.

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 $\lambda \le O(L_f/r_\epsilon)$, 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*

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

Proof

By the choice of r_ϵ and the bound $\lambda = O(L_f^2/\epsilon')$, Lemma 3 guarantees that the γ_i have strong convexity $\mu = \Omega(\lambda)$, and smoothness $L = O(L_g + \lambda + L_f^2/\epsilon' + \lambda^2 r_\epsilon^2/\epsilon') = O(L_g + L_f^2/\epsilon')$. In addition, for $x_\star := \arg\min_{x\in\mathbb{B}_{r_\epsilon}(\bar{x})} \Gamma_{\epsilon,\lambda}(x)$ one has

$$\begin{aligned}\Gamma_{\epsilon,\lambda}(\bar{x}) - \Gamma_{\epsilon,\lambda}(x_\star) &\le \langle \nabla_x \Gamma_{\epsilon,\lambda}(x_\star), \bar{x} - x_\star\rangle \\ &\le \|\nabla_x \Gamma_{\epsilon,\lambda}(x_\star)\|\|\bar{x} - x_\star\| = O(L_f r_\epsilon),\end{aligned}$$

where we use the second property for bounding $\nabla_x \Gamma_{\epsilon,\lambda}(\bar{x})$ from Lemma 3. Plugging these into the complexity of Lemma 25 and noticing that each evaluation of $\nabla\gamma_i$ requires evaluation of f_i and ∇f_i (assuming $f_i(\bar{x})$ is pre-stored) gives the stated complexity bound. ■

C.4. Proof of Theorem 6

With Corollaries 4 and 5 established, we prove Theorem 6.

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 $\|x_0 - x_\star\| \le R$ 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 $K := L_f R\epsilon^{-1}\log N$. 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 $\ge \frac{99}{100}$ and has cost

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

Proof We use guarantees of Theorem 2 and Corollaries 4 and 5 on the problem $\min_x F_{\mathrm{smax},\epsilon}(x)$, $\|x - x_0\| \le O(R)$ to prove the correctness and bound the complexity of the algorithm.

Correctness. We first note that Theorem 2 guarantees that we only make BROO calls with $\lambda \le O(L_f/r_\epsilon)$, making Corollaries 4 and 5 applicable. Let

$$T = O\left(\left(\frac{R}{r_\epsilon}\right)^{2/3}\log^2\left(\frac{L_f R^2}{r_\epsilon \epsilon}\right)\right) \tag{32}$$

be the upper bound on the total number of oracle calls guaranteed in Theorem 2. Taking $\sigma = \frac{1}{100T}$ in Corollaries 4 and 5 and applying a union bound, we see that with probability at least $99/100$, the outputs of the corresponding BROO implementations are valid throughout the execution of Algorithm 1. Consequently by Theorem 2 we have that Algorithm 1 with accuracy $\epsilon/2$ outputs x_o such that $F_{\mathrm{smax},\epsilon}(x_o) - \min_{x:\|x-x_0\|\le R} F_{\mathrm{smax},\epsilon}(x) \le \epsilon/2$. Using the fact that $0 \le F_{\mathrm{smax},\epsilon}(x) - F_{\max}(x) \le \epsilon/2$ for all $x \in \mathbb{R}^d$ [see, e.g., 12, Lemma 45], we conclude that $F_{\max}(x_o) - \min_x F_{\max}(x) = F_{\max}(x_o) - \min_{\{x:\|x-x_0\|\le R\}} F_{\max}(x) \le F_{\mathrm{smax},\epsilon}(x_o) - \min_{x:\|x-x_0\|\le R} F_{\mathrm{smax},\epsilon}(x) + \epsilon/2 \le \epsilon$, establishing correctness.

Complexity. Substituting $r_\epsilon = \epsilon/(2L_f\log N)$ into (32), we see that the total number of oracle calls is

$$T = O\left(\left(\frac{L_f R\log N}{r_\epsilon}\right)^{2/3}\log^2\left(\frac{L_f R\log N}{\epsilon}\right)\right).$$

To bound to complexity of the SGD implementation, we simply multiply T by the per-call complexity bound (6) where we substitute $\delta = \Omega(\epsilon/(\lambda R))$ as guaranteed by Theorem 2 and $\sigma = 1/(100T)$.

To bound the complexity of the accelerated variance reduction implementation, we similarly substitute $\delta = \Omega(\epsilon/(\lambda R))$, $\lambda = O(L_f/r_\epsilon)$ and $\sigma = 1/(100T)$ into (7). Summing the result over all oracle calls yields the complexity bound

$$O\left((\mathcal{T}_f + \mathcal{T}_g)\log\left(\frac{L_f R\log N}{\epsilon}\right)\left(NT + \sqrt{N}\left(\sqrt{L_g} + L_f\sqrt{\frac{\log N}{\epsilon}}\right)\sum_{i\in[T]}\frac{1}{\sqrt{\lambda_{(i)}}}\right)\right),$$

where $\{\lambda_{(i)}\}$ is the sequence of λ values with which Algorithm 1 calls the BROO implementation. Theorem 2 guarantees that $\sum_{i\in[T]} \frac{1}{\sqrt{\lambda_{(i)}}} \le O\big(\frac{R}{\sqrt{\epsilon}} \log \frac{L_f R^2}{r\epsilon}\big) = O\big(\frac{R}{\sqrt{\epsilon}} \log \frac{L_f R \log N}{\epsilon}\big)$, completing the proof. ■

Appendix D. Lower bound proofs

D.1. Proof of Proposition 8

In this section, we make frequent use of the indication notation $\mathbb{I}\{\cdot\}$, where $\mathbb{I}\{A\} = 1$ if event A holds and $\mathbb{I}\{A\} = 0$ otherwise.

Proposition 8 *Let $\delta, \alpha \in (0,1)$ and let $N, T \in \mathbb{N}$ with $T \le N/2$. Let $(f_i)_{i\in[N]}$ be an α-robust N-element zero-chain with domain $\mathbb{B}_1^T(0)$. For $d \ge T + \frac{2}{\alpha^2}\log\frac{4NT^2}{\delta}$, draw U uniformly from the set of $d \times T$ orthogonal matrices, and draw Π uniformly from the set of permutations of $[N]$. Let $\tilde{f}_i(x) := f_{\Pi^{-1}(i)}(U^\top x)$. Let $\{(i_t, x_t)\}_{t\ge1}$ 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*

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

Proof Let us define several quantities that are important for our proof. First, we track the maximum progress attained by the algorithm queries.

$$p_t := \mathrm{prog}_\alpha(U^\top x_t) \;\text{ and }\; \bar{p}_t := \max_{s\le t} p_s.$$

Next, we recursively define a sequence that tracks the algorithm's progress in "unlocking" the relevant elements of the finite sum,

$$B_t := \mathbb{I}\{\Pi^{-1}(i_t) = C_t + 1\} \;\text{ where }\; C_t := \min\left\{\sum_{s<t} B_s, T\right\}.$$

To understand these definitions, note that C_t is the largest number k such that $1, 2, \ldots, k$ is a subsequence of $\Pi^{-1}(i_1), \Pi^{-1}(i_2), \ldots, \Pi^{-1}(i_t)$. Therefore, a "zero-respecting" algorithm (satisfying $p_t \le \max_{s<t}\{\mathrm{prog}_\alpha(\nabla f_{\Pi^{-1}(i)}(U^\top x_s))\}$) can only query points with progress at most C_t. We define the event that the general algorithm under consideration behaves as though it was zero-respecting for the first t iterations as

$$\mathfrak{Z}\mathfrak{R}_t := \{\bar{p}_s \le C_s \;\text{ for all }\; s \le t\}.$$

We also define the stopping time

$$\Theta_k := \min\{t \mid C_t = k\}$$

and the difference sequence

$$\Delta_k := \Theta_k - \Theta_{k-1}.$$

Let

$$\tau := \frac{1}{16}N\left(T - \log\frac{2}{\delta}\right)$$

so that our goal is to prove that $\mathbb{P}\big(\bar{p}_{\lfloor\tau\rfloor} < T\big) > 1-\delta$. Note that the intersection of the events $C_{\lfloor\tau\rfloor} < T$ and $\mathfrak{Z}\mathfrak{R}_{NT} \subset \mathfrak{Z}\mathfrak{R}_{\lfloor\tau\rfloor}$ imply the desired event $\bar{p}_{\lfloor\tau\rfloor} < T$. Moreover, $C_{\lfloor\tau\rfloor} < T$ is equivalent to $\Theta_T > \tau$. Consequently, we can upper bound $\mathbb{P}\big(\bar{p}_{\lfloor\tau\rfloor} \ge T\big)$ by the probability that $\mathfrak{Z}\mathfrak{R}_{NT}$ does not occur plus the probability that both $\Theta_T \le \tau$ and $\mathfrak{Z}\mathfrak{R}_{NT}$ occur, i.e.,

$$\mathbb{P}\big(\bar{p}_{\lfloor\tau\rfloor} \ge T\big) \le \mathbb{P}(\mathfrak{Z}\mathfrak{R}_{NT}^c) + \mathbb{P}(\Theta_T \le \tau, \mathfrak{Z}\mathfrak{R}_{NT}).$$

Our strategy is to bound each of $\mathbb{P}(\mathfrak{Z}\mathfrak{R}_{NT}^c)$ and $\mathbb{P}(\Theta_T \le \tau, \mathfrak{Z}\mathfrak{R}_{NT})$ by $\delta/2$.

As one final piece of of notation, we write $U_{\le k}$ for the first k columns of U, and $\Pi|_k$ as shorthand for $\Pi(1), \ldots, \Pi(k)$.

Lemma 27 *For any $t \ge 1$ and $k \le T$, if the events $\mathfrak{Z}\mathfrak{R}_{t-1}$ and $C_t \le k$ hold, the oracle responses to queries $(i_s, x_s)_{s<t}$, as well as the queries $(i_s, x_s)_{s\le t}$, are deterministic (measurable) functions of ζ, $\Pi|_k$ and $U_{\le k}$. Moreover,*

(a) The random variable $\Theta_k \mathbb{I}\{\mathfrak{Z}\mathfrak{R}_{\Theta_k}\}$ is measurable w.r.t $\zeta, \Pi|_k$ and U.

(b) When $\mathfrak{Z}\mathfrak{R}_{t-1}$ holds, $\mathbb{I}\{C_t = k\}$ is measurable w.r.t. ζ, Π and $U_{\le k}$.

Proof Consider the query i_s, x_s for $s < t$; we first show that under $\mathfrak{Z}\mathfrak{R}_{t-1}$ and $C_t \le k$ we can compute the oracle response to this query using only $\Pi|_k$ and $U_{\le k}$. Since $\mathfrak{Z}\mathfrak{R}_{t-1}$ holds, we have that $\mathrm{prog}_\alpha(U^\top x_s) = p_s \le C_s$. Invoking Definition 7 of the N-element zero chain, there exists a neighborhood of x_s such that for all y in that neighborhood we have

$$\tilde{f}_{i_s}(y) = f_{\Pi^{-1}(i_s)}(U^\top y) = \begin{cases} f_{\Pi^{-1}(i_s)}(U_{\le C_s}^\top y) & \Pi^{-1}(i_s) < C_s + 1 \\ f_{\Pi^{-1}(i_s)}(U_{\le C_s+1}^\top y) & \Pi^{-1}(i_s) = C_s + 1 \\ f_N(U_{\le C_s}^\top y) & \Pi^{-1}(i_s) > C_s + 1. \end{cases} \tag{33}$$

Recall that $B_s = \mathbb{I}\{\Pi^{-1}(i_s) = C_s + 1\}$ and that $C_{s+1} = \min\{C_s + B_s, T\} \le C_t \le k$ and let

$$\Pi^{-1}|_k(j) = \begin{cases} \Pi^{-1}(j) & j \in \Pi([k]) \\ N & \text{otherwise.} \end{cases}$$

The relationship (33) implies that

$$\tilde{f}_{i_s}(y) = f_{\Pi^{-1}|_k(i_s)}(U_{\le k}^\top y)$$

for all y in the neighborhood of x_s.

The above discussion shows that (under $\mathfrak{Z}\mathfrak{R}_{t-1}$) the oracle response to (i_s, x_s) is measurable with respect to $i_s, x_s, \Pi|_k$ and $U_{\le k}$, for every $s < t$. Moreover, (i_1, x_1) is measurable with respect to ζ, and consequently the first oracle response and the second query (i_2, x_2) are measurable with respect to $\zeta, \Pi|_k$ and $U_{\le k}$. Repeating this argument inductively shows that (under $\mathfrak{Z}\mathfrak{R}_{t-1}$ and $C_t \le k$) we can generate the entire query sequence $(i_s, x_s)_{s\le t}$, as well as the oracle responses to all but the last query, from $\zeta, \Pi|_k$ and $U_{\le k}$, as claimed.

To show part (a) of the lemma, note that with access to the entire matrix U, after generating queries $x_1, \ldots, x_s$ we can test whether $\mathfrak{Z}\mathfrak{R}_s$ holds. Moreover, the knowledge of $\Pi|_k$ suffices to test (for any query sequence $i_1, i_2, \ldots$) whether $C_t \ge k$ for every t. Therefore, using $\zeta, \Pi|_k$ and $U_{\le k}$ we can iteratively compute $(i_s, x_s)_{s\le t}$ until arriving at an iterate t where either (a) $C_t < k$ but $\mathfrak{Z}\mathfrak{R}_t$

does not hold (which implies that $\mathfrak{Z}\mathfrak{R}_{\Theta_k}$ fails as well), or (b) $C_t = k$ (in which case we have found Θ_k). In either case, we know the value of $\Theta_k \mathbb{I}\{\mathfrak{Z}\mathfrak{R}_{\Theta_k}\}$.

Finally, to show part (b) of the lemma, note that with full knowledge of Π we can track the values of $C_1, \ldots, C_t$ for any sequence of queries $i_1, \ldots, i_{t-1}$. Therefore, given $\mathfrak{Z}\mathfrak{R}_{t-1}$, we may use $\zeta, \Pi|_k$ and $U_{\le k}$ to iteratively generate queries until either (a) we arrive at an iteration $s < t$ where $C_s > k$, in which case $\mathbb{I}\{C_t = k\} = 0$, or (b) we successfully generate iteration $t-1$ in which case we can compute C_t. In either case, we know the value of $\mathbb{I}\{C_t = k\}$. ■

Lemma 28 *For every $k \le T$, and $j \ge 0$, we have*

$$\mathbb{P}(\Delta_k \le j, \mathfrak{Z}\mathfrak{R}_{\Theta_k} \mid \zeta, \Pi|_{k-1}, U) \le \frac{j}{N-k+1}.$$

Proof Suppose the algorithm could access the random permutation Π via an alternative oracle that, when queried at index $i \in [N]$ returns the number $\Pi^{-1}(i)$. We say that the algorithm succeeds if one of the first j queries is $\Pi(k)$ (so that the oracle returns k). Given $\Pi|_{k-1}$, the random variable $\Pi(k)$ is uniformly distributed over $\mathcal{I} = [N] \setminus \Pi([k-1])$. Therefore, for any set $\mathcal{J}$ of j queries, we have

$$\mathbb{P}(\Pi(k) \in \mathcal{J} \mid \Pi|_{k-1}) = \frac{\mathbb{E}|\mathcal{J} \cap \mathcal{I}|}{|\mathcal{I}|} \le \frac{\mathbb{E}|\mathcal{J}|}{|\mathcal{I}|} = \frac{j}{N-k+1}$$

and so for every algorithm the probability of success is at most $j/(N-k+1)$.

Now return to the original problem and the original oracle, and note that for $\Delta_k \le j$ to hold, we must have $\Pi^{-1}(i_t) = k$ for some t in $\{\Theta_{k-1}, \ldots, \Theta_{k-1} + j - 1\}$, corresponding to "success" in the problem described above. Moreover, when the event $\mathfrak{Z}\mathfrak{R}_{\Theta_k}$ holds, Lemma 27 guarantees that the oracle responses to the queries at iteration $1, \ldots, \Theta_{k-1} - 1$ are deterministic functions of ζ, U and $\Pi|_{k-1}$ (since $C_{\Theta_{k-1}} = k-1$ by definition). Consequently, when $\mathfrak{Z}\mathfrak{R}_{\Theta_{k-1}}$ holds, the true optimization algorithm operates with no more information that the alternative oracle setting described above, giving the claimed probability bound. ■

Lemma 29 *We have*

$$\mathbb{P}\big(\Theta_T \le \tfrac{1}{16} N (T - \log \tfrac{2}{\delta}), \mathfrak{Z}\mathfrak{R}_{NT}\big) \le \frac{\delta}{2}.$$

Proof Let $T_{\pm\delta} \coloneqq \frac{1}{2}\big(T \pm \log \frac{2}{\delta}\big)$. We begin with a sequence of straightforward inequalities:

$$\begin{aligned}
\mathbb{P}(\Theta_T \le \tfrac{1}{8} N T_{-\delta}, \mathfrak{Z}\mathfrak{R}_{NT}) &\le \mathbb{P}(\Theta_T \le \tfrac{1}{8} N T_{-\delta}, \mathfrak{Z}\mathfrak{R}_{\Theta_T}) \\
&\le \mathbb{P}\left(\sum_{k \in [T]} \mathbb{I}\{\Delta_k > \tfrac{1}{8} N\} \le T_{-\delta}, \mathfrak{Z}\mathfrak{R}_{\Theta_T}\right) \\
&= \mathbb{P}\left(\sum_{k \in [T]} \mathbb{I}\{\Delta_k \le \tfrac{1}{8} N\} > T_{+\delta}, \mathfrak{Z}\mathfrak{R}_{\Theta_T}\right) \\
&\le \mathbb{P}\left(\sum_{k \in [T]} \mathbb{I}\{\Delta_k \le \tfrac{1}{8} N, \mathfrak{Z}\mathfrak{R}_{\Theta_k}\} > T_{+\delta}\right) \\
&\le e^{-2T_{+\delta}} \mathbb{E}\Big[e^{2 \sum_{k \in [T]} \mathbb{I}\{\Delta_k \le \frac{1}{8} N, \mathfrak{Z}\mathfrak{R}_{\Theta_k}\}}\Big], \qquad (34)
\end{aligned}$$

where the last transition is an application of the Chernoff bound (with parameter $\lambda = 2$).

By Lemma 27, the random variable $\mathbb{I}\{\Delta_k \le \frac{1}{8}N, \mathfrak{Z}\mathfrak{R}_{\Theta_k}\}$ is measurable with respect to $\zeta, \Pi|_k$ and U since it is a function of $\Theta_k \mathbb{I}\{\mathfrak{Z}\mathfrak{R}_{\Theta_k}\}$ and $\Theta_{k-1}\mathbb{I}\{\mathfrak{Z}\mathfrak{R}_{\Theta_{k-1}}\}$. Therefore,

$$
\begin{aligned}
&\mathbb{E}\left[e^{2\sum_{k\in[T]}\mathbb{I}\{\Delta_k \le \frac{1}{8}N, \mathfrak{Z}\mathfrak{R}_{\Theta_k}\}}\right] \\
&\quad= \mathbb{E}\left[\left(1 + (e^2-1)\mathbb{P}\left(\Delta_T \le \tfrac{1}{8}, \mathfrak{Z}\mathfrak{R}_{\Theta_T} \mid \zeta, \Pi|_{T-1}, U\right)\right)e^{2\sum_{k\in[T-1]}\mathbb{I}\{\Delta_k \le \frac{1}{8}N, \mathfrak{Z}\mathfrak{R}_{\Theta_k}\}}\right] \\
&\quad\le \left(1 + \frac{1}{4}(e^2-1)\right)\mathbb{E}\left[e^{2\sum_{k\in[T-1]}\mathbb{I}\{\Delta_k \le \frac{1}{8}N, \mathfrak{Z}\mathfrak{R}_{\Theta_k}\}}\right],
\end{aligned}
$$

where the inequality follows from Lemma 28 with $j = N/4$ and $k = T \le N/2$. Noting that $1 + \frac{1}{4}(e^2-1) \le e$ and iterating this argument, we conclude that

$$
\mathbb{E}\left[e^{2\sum_{k\in[T]}\mathbb{I}\{\Delta_k \le \frac{1}{8}N, \mathfrak{Z}\mathfrak{R}_{\Theta_k}\}}\right] \le e^T.
$$

Substituting this bound into eq. (34) and recalling that $e^{-2T+\delta} = \frac{\delta}{2}e^{-T}$ concludes the proof. ■

Lemma 30 *We have*

$$
\mathbb{P}(\mathfrak{Z}\mathfrak{R}^c_{NT}) \le \frac{\delta}{2}.
$$

Proof By definition, we have

$$
\mathbb{P}(\mathfrak{Z}\mathfrak{R}^c_{NT}) = \sum_{t\in[NT]} \mathbb{P}(p_t > C_t, \mathfrak{Z}\mathfrak{R}_{t-1}). \tag{35}
$$

We fix $t \le NT$ and argue that $\mathbb{P}(p_t > C_t, \mathfrak{Z}\mathfrak{R}_{t-1}) \le \frac{\delta}{2NT}$. Let $k \le T$; by Lemma 27 events $\mathfrak{Z}\mathfrak{R}_{t-1}$ and $C_t = k$ hold, we have $x_t = h(\zeta, \Pi, U_{\le k})$ for some measurable functions h. Therefore, we have

$$
\begin{aligned}
&\mathbb{P}(p_t > C_t, \mathfrak{Z}\mathfrak{R}_{t-1} \mid C_t = k, \zeta, \Pi, U_{\le k}) \overset{(i)}{=} \mathbb{P}(p_t > k, \mathfrak{Z}\mathfrak{R}_{t-1} \mid \zeta, \Pi, U_{\le k}) \\
=&\mathbb{P}\left(\mathrm{prog}_\alpha(U^\top x_t) > k, \mathfrak{Z}\mathfrak{R}_{t-1} \;\middle|\; \zeta, \Pi, U_{\le k}\right) \\
=&\mathbb{P}\left(\mathrm{prog}_\alpha(U^\top h(\zeta, \Pi, U_{\le k})) > k, \mathfrak{Z}\mathfrak{R}_{t-1} \;\middle|\; \zeta, \Pi, U_{\le k}\right) \\
\le&\mathbb{P}\left(\mathrm{prog}_\alpha(U^\top h(\zeta, \Pi, U_{\le k})) > k \;\middle|\; \zeta, \Pi, U_{\le k}\right) \overset{(ii)}{\le} \sum_{j=k+1}^{T} \mathbb{P}\left(\left|u_j^\top h(\zeta, \Pi, U_{\le k})\right| > \alpha \;\middle|\; \zeta, \Pi, U_{\le k}\right),
\end{aligned}
$$

where (i) follows from part (b) of Lemma 27, and (ii) follows from the definition of prog_α and a union bound, where we write u_j for the jth column of U.

Conditionally on $\zeta, \Pi, U_{\le k}$, u_j is uniform over the unit ball in the subspace of $\mathbb{R}^d$ orthogonal to $U_{\le k}$. Moreover, the magnitude of the projection of $h(\zeta, \Pi, U_{\le k})$ to that subspace can be at most 1, since x_t is a unit vector. Therefore, the probability that $\left|u_j^\top h(\zeta, \Pi, U_{\le k})\right| > \alpha$ holds is at most the probability that a coordinate of a uniform unit vector in $\mathbb{R}^{d-k}$ has magnitude greater than α. By standard concentration of measure arguments, this probability is at most $2\exp(-\frac{d-k+1}{2\alpha^2})$. By our choice of d (and recalling $k \le T$), we have that

$$
\mathbb{P}(p_t > C_t, \mathfrak{Z}\mathfrak{R}_{t-1} \mid C_t = k, \zeta, \Pi, U_{\le k}) \le (T-k) \cdot \frac{\delta}{2NT^2} \le \frac{\delta}{2NT},
$$

and substituting back into eq. (35) concludes the proof. ■

■

D.2. Proof of Lemma 9

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

Proof To see why part 1 holds, fix $x \in \mathbb{R}^T$ and let $p = \mathrm{prog}_{\alpha_T}(x)$. First, we have $\hat{f}_i^{\{T,N,\ell\}}(x) = \hat{f}_i^{\{T,N,\ell\}}(x_{[\leq i]})$ for every i and x, which immediately gives the first two cases of eq. (11). Second, when $i > p+1$, we have $|x_{[i]} - x_{[i-1]}| < 2\alpha_T$ and therefore $\hat{f}_i^{\{T,N,\ell\}}(x) = \psi_{\alpha_T,\ell}(t)\mathbb{I}\{i \leq N\}$ for some $|t| < \alpha$. Consequently $\hat{f}_i^{\{T,N,\ell\}}$ is identically zero in a neighborhood of x, giving the third and final case in eq. (11).

Part 2 is immediate because $\hat{f}_i^{\{T,N,\ell\}}$ is a composition of a 1-Lipschitz and ℓ-smooth function with the linear transformation $(x_{[i]} - x_{[i-1]})/2$ which has operator norm smaller than 1.

Finally, to see part 3, first note that the global minimum of $\hat{F}_{\max}^{\{T,N,\ell\}}$ satisfies $\hat{F}_{\max}^{\{T,N,\ell\}}(x_\star) = 0$ and $(x_\star)_{[i]} = \frac{1}{\sqrt{T}}$ for every $i \leq T$ (and therefore has unit norm). Consider any x for which $\mathrm{prog}_{\alpha_T}(x) < T$, so that $x_{[T]} \leq \alpha_T \leq \frac{1}{\sqrt{T}}$. We have
$$\max_{i\leq T}\left|x_{[i-1]} - x_{[i]}\right| \geq \frac{1}{T}\sum_{i\in[T]}\left|x_{[i-1]} - x_{[i]}\right| \geq \frac{1}{T}\left|x_{[0]} - x_{[T]}\right| \geq \frac{1}{T}\left(\frac{1}{\sqrt{T}} - \alpha_T\right) \geq \frac{3}{4T^{3/2}}.$$

Since $\psi_{\alpha_T,\ell}(t)$ is non-decreasing in $|t|$, we have $\hat{F}_{\max}^{\{T,N,\ell\}}(x) = \psi_{\alpha_T,\ell}\big(\frac{1}{2}\max_{i\leq T}\left|x_{[i-1]} - x_{[i]}\right|\big)$, and consequently $\hat{F}_{\max}^{\{T,N,\ell\}}(x) \geq \psi_{\alpha_T,\ell}\left(\frac{3}{8T^{3/2}}\right)$. To obtain the final bound, we observe that $\psi_{\alpha,\ell}(t) \geq \min\left\{\frac{1}{2}(t-\alpha), \frac{\ell}{2}(t-\alpha)^2\right\}$ for any $t \geq \alpha$. ■

D.3. Proof of Theorem 10

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 $d = O\left(\left[\left(\frac{L_f R}{\epsilon}\right)^2 \wedge \left(\frac{L_g R^2}{\epsilon}\right)\right] \log \frac{N(L_f R \wedge L_g R^2)}{\epsilon}\right)$ 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 $F_{\max}(x) = \max_{i\in[N]} f_i(x)$.

Proof We first show that an $\Omega\big(N\big[(\frac{L_f R}{\epsilon})^{2/3} \wedge (\frac{L_g R^2}{\epsilon})^{1/3}\big]\big)$ lower bound follows from our construction in the previous section. Fix any $T > 1$ and $\ell \geq 0$ and let $d = \lceil T + 4\alpha_T^{-2} \log 4NT \rceil$ with $\alpha_T = 1/(4T^{3/2})$. Let Π be random permutation of $[N]$ and let U be drawn uniformly from the set of $d \times T$ orthogonal matrices. For $i \in [N]$, let

$$\tilde{f}_i(x) = \hat{f}^{\{T,N,\ell\}}_{\Pi^{-1}(i)}(U^\top x).$$

Then, by Proposition 8 and Lemma 9.1, any optimization algorithm interacting with $(\tilde{f}_i)_{i \leq N}$ satisfies with probability as least $1/2$ that $\mathrm{prog}_{\alpha_T}(U^\top x_i) < T$ for every $i \leq \frac{1}{64}NT$.

Set

$$T = \left\lfloor \frac{1}{5}\left[\Big(\frac{L_f R}{\epsilon}\Big)^{2/3} \wedge \Big(\frac{L_g R^2}{\epsilon}\Big)^{1/3}\right]\right\rfloor \text{ and } \ell = \frac{L_g R}{L_f}.$$

We may assume $T \leq N/2$ without loss of generality, since otherwise the second term in the lower bound dominates. Let

$$f_i(x) = L_f R \tilde{f}_i(x/R).$$

With these settings, we have that f_i is both L_f-Lipschitz and L_g-smooth for every i due to Lemma 9.2, the choice of ℓ, and the fact that U is orthogonal. Moreover, if $x_1, x_2, \ldots$ are the iterates of an algorithm interacting with a finite sum oracle for $(f_i)_{i\in[N]}$ then $x_1/R, x_2/R, \ldots$ are the iterates of an algorithm interacting with $(\tilde{f}_i)_{i\leq[N]}$. Therefore, by the above discussion, with probability at least $1/2$ the firstp

$$\frac{1}{64}NT = \Omega\left(N\left[\Big(\frac{L_f R}{\epsilon}\Big)^{2/3} \wedge \Big(\frac{L_g R^2}{\epsilon}\Big)^{1/3}\right]\right)$$

iterates of the algorithm satisfy $\mathrm{prog}_{\alpha_T}(U^\top x) < T$, and consequently, by Lemma 9.3, they are

$$L_f R \min\left\{\frac{1}{8T^{3/2}}, \frac{\ell}{32T^3}\right\} = \min\left\{\frac{L_f R}{8T^{3/2}}, \frac{L_g R^2}{32T^3}\right\} > \epsilon$$

suboptimal for $\max_{i\in[N]} f_i$. To conclude the linear in N lower bound, we note that since the suboptimality bound holds with probability at least $1/2$ over Π, U and the algorithm randomness, for every algorithm there must exist fixed Π and U for which the bound holds with probability $1/2$ over the randomness of the algorithm alone.

To show the remaining $\Omega\big(\big[(\frac{L_f R}{\epsilon})^2 \wedge (\frac{N L_g R^2}{\epsilon})^{1/2}\big]\big)$ term in the lower bound, we recall the following classical result. For every $R' > 0$ there exists a distribution over functions $F : \mathbb{R}_d \to \mathbb{R}$ with $d' = O\Big(\Big[(\frac{L_f R'}{\epsilon})^2 \wedge (\frac{L_g R'^2}{\epsilon})^{1/2}\Big] \log \frac{N L_f R'}{\epsilon}\Big)$ that are L_g-Lipschitz, L_f-smooth and has a global minimizer with norm at most R, such that for any algorithm interacting with F, with probability at least $1 - \frac{1}{2N}$, the first

$$T_{R'} = \Omega\left(\left[\Big(\frac{L_f R'}{\epsilon}\Big)^2 \wedge \Big(\frac{Lg R'^2}{\epsilon}\Big)^{1/2}\right]\right)$$

iterations are ϵ suboptimal for F. This result follows from smoothing Nemirovski's function; see Diakonikolas and Guzmán [14, Theorem 14 with $p = 2$ and $\kappa = 0, 1$]. To strengthen the smooth term in the lower bound, we consider N copies of this construction with $R' = R/\sqrt{N}$ that operate on distinct coordinates, i.e., we let $f_i(x) = F(x_{[1+(i-1)d']}, \ldots, x_{[id']})$, so that the global minimizer of

$\max_i f_i(x)$, which consists of N copies of the global minimizer of F, has norm at most $R'\sqrt{N} = R$, and consequently we may constrain the domain to $\mathbb{B}_N^{d'N}$ without decreasing the optimality gap of any point in the ball. For any algorithm interacting with $(f_i)_{i\in[N]}$, the first

$$NT_{R/\sqrt{N}} = \Omega\left(\left[\left(\frac{L_f R}{\epsilon}\right)^2 \wedge \left(\frac{NLgR^2}{\epsilon}\right)^{1/2}\right]\right)$$

queries of the algorithm must select one of the N components at most $T_{R/\sqrt{N}}$ times and therefore (with probability at least $\frac{1}{2}$) be ϵ suboptimal for at least one component, and hence for their maximum. This gives the sublinear in N term of the lower bound. We remark that the "hard instance duplication" argument is at the core of existing lower bounds for finite sum optimization [39, 15]; our argument for proving the linear in N lower bound is inherently different. ■

D.4. Extension to unconstrained setting

While we state and prove our lower bound for optimization problems whose domain is a ball of radius R, it also extends to the case of unconstrained setting of our upper bounds. That is, when the domain is $\mathbb{R}^d$ and we are guaranteed a local minimizer exists in a radius of R from the initial point. One way to show this extension is the technique of [14] where we replace $\hat{f}_i^{\{T,N,\ell\}}$ with

$$\bar{f}_i^{\{N,T,\ell\}}(x) := \min_{y\in\mathbb{R}^d}\left\{\max\left\{\hat{f}_i^{\{T,N,\ell\}}(y), \|y\| - 1 - \ell^{-1}\right\} + \frac{\ell}{2}\|y-x\|^2\right\}. \quad (36)$$

The definition above consists of two modification: pairwise maximum with $\|\cdot\| - 1$ and infimal convolution with $\frac{\ell}{2}\|\cdot\|^2$. The pairwise maximum guarantees that large norm queries cannot break the zero-chain progress control mechanism: responses to query points with norm $\Omega(1)$ will not depend on the random transformation U at all, while for responses with norm $O(1)$ we can control the progress using the zero-chain structure of $\hat{f}^{\{T,N,\ell\}}$ as before. The infimal convolution guarantees the function remains ℓ smooth (and also does not increase the Lipschitz constant).

Finally, it remains to check that the new construction still satisfies property 3 of Lemma 9 up to a constant. To see that it does, first note that the global minimizer of $\bar{F}_{\max}^{\{N,T,\ell\}}(x) := \max_{i\in[N]} \bar{f}_i^{\{N,T,\ell\}}(x)$ still satisfies $x_{\star[i]} = 1/\sqrt{T}$ for all $i \le T$, and that $\bar{F}_{\max}^{\{N,T,\ell\}}(x_\star) = 0$. Moreover, we clearly have

$$\bar{f}_i^{\{N,T,\ell\}}(x) \ge \min_{y\in\mathbb{R}^d}\left\{\psi_{\alpha_T,\ell}(y/2) + \frac{\ell}{2}\left\|y - \frac{x_{[i]} - x_{[i-1]}}{2}\right\|^2\right\} := \tilde{\psi}\left(\frac{x_{[i]} - x_{[i-1]}}{2}\right),$$

and it is not hard to verify that $\tilde{\psi}(t) \ge \psi_{\alpha_T,\ell}(ct)$ for some constant $c > 0$ (in fact, $\tilde{\psi}(t) = \psi_{\alpha_T,c\ell}(t)$).