18 Lower Bound I

Goal

Black-box optimization, local oracle, performance estimation, lower bound for nonsmooth minimization, lower bound for smooth minimization

Alert 18.1: Convention

An overview of information-based complexity bound is Traub and Werschulz (2009), and a classic reference for this lecture is Nemirovski and Yudin (1983).

Gray boxes are not required hence can be omitted for unenthusiastic readers.

This note is likely to be updated again soon.

Traub, J. F. and A. G. Werschulz (2009). "Information-based complexity and information-based optimizationInformation-based Complexity and Information-based Optimization". In: *Encyclopedia of Optimization*. Ed. by C. A. Floudas and P. M. Pardalos, pp. 1603–1608.

Nemirovski, A. S. and D. B. Yudin (1983). "Problem complexity and method efficiency in optimization". Wiley.

Remark 18.2: Summary so far

The following table summarizes the algorithms we have learned so far:

☑: inherited by generality ✪: not fully discussed –: not discussed at all green: this lecture.

alg	code	obj	cons	step size				rate			opt?
				Cauchy	Amijo	const	diminish	func	grad	dist	
gd	1.4	smooth	✖	☑	☑	☑	–	–	$\frac{1}{\sqrt{t}}$	–	✖
projgrad	19.15	smooth	☑	–	☑	☑	–	$\frac{1}{t}$	–	–	✖
proxgrad	2.17	sum	☑	–	☑	☑	–	$\frac{1}{t}$	–	–	✖
condgrad	3.7	sum	☑	☑	☑	☑	☑	$\frac{1}{t}$	–	–	☑
subgrad	4.14	convex	✪	✖	–	✪	☑	$\frac{1}{\sqrt{t}}$	–	$\frac{1}{\sqrt{t}}$	☑
md	5.10	convex	✪	✖	–	✪	☑	$\frac{1}{t}$ or $\frac{1}{\sqrt{t}}$	–	$\frac{1}{\sqrt{t}}$	☑
metricgrad	6.8	smooth	✪	☑	–	☑	–	–	$\frac{1}{\sqrt{t}}$	–	?

An obvious question we have now is that, are any of these algorithms optimal? If yes, in what sense? If no, how do we find *provably* better ones? In this lecture we establish lower complexity bounds that any algorithm would have to respect, and starting from the next lecture we will develop faster algorithms that hopefully will close any gap between the lower bounds we prove below and the upper bounds shown in the above table.

We emphasize that we are mostly interested in dimension independent bounds, i.e., algorithms whose required number of iterations to reach a pre-defined tolerance does not depend on the dimension at all or very mildly (e.g. logarithmically). Of course, the per-step complexity of an algorithm may still be proportional to the dimension and vary from algorithm to algorithm. Needless to say, the actual running time of an algorithm is the product of the number of iterations and the per-step cost.

Definition 18.3: Function, oracle, algorithm and performance in black-box optimization

Formally, we are interested in minimizing a class of functions $\mathcal{F} \subseteq \mathbb{R}^{\mathsf{W}}$. We start from $\mathbf{w}_0 \in \bar{\mathsf{W}}$, a possible enlargement of W when it is difficult to maintain feasibility. We restrict our access to the function f through

an oracle

$$\mathcal{O} : \bar{\mathsf{W}}^* \times \mathcal{F} \to \mathsf{O}, \quad \text{where} \quad \bar{\mathsf{W}}^* := \bigcup_{t=1}^{\infty} \bar{\mathsf{W}}^t.$$

In other words, we do not inspect the underlying function f per se, but through querying an oracle to obtain information about it. Suboptimal at first glance, this black-box view turns out to be quite insightful when the underlying function is too complicated or when the chosen algorithm is limited either inherently or externally.

An algorithm is formally defined as a mapping

$$\mathcal{A} : \bar{\mathsf{W}} \times \mathsf{O}^* \to \bar{\mathcal{W}}$$

such that with the short-hand $\mathcal{O}_f := \mathcal{O}(\cdot, f)$, sequentially we have:

$$\begin{aligned}
\mathbf{w}_1 &= \mathcal{A}\big(\mathbf{w}_0, \mathcal{O}_f(\mathbf{w}_0)\big)\\
\mathbf{w}_2 &= \mathcal{A}\big(\mathbf{w}_0, \mathcal{O}_f(\mathbf{w}_0), \mathcal{O}_f(\mathbf{w}_0, \mathbf{w}_1)\big)\\
&\vdots\\
\mathbf{w}_t &= \mathcal{A}\big(\mathbf{w}_0, \mathcal{O}_f(\mathbf{w}_0), \mathcal{O}_f(\mathbf{w}_0, \mathbf{w}_1), \dots, \mathcal{O}_f(\mathbf{w}_0, \mathbf{w}_1, \dots, \mathbf{w}_{t-1})\big),
\end{aligned}$$

or collectively (and slightly abusing the notation) as

$$(\mathbf{w}_1, \dots, \mathbf{w}_t) := \mathcal{A}_t(\mathbf{w}_0, \mathcal{O}_f).$$

We evaluate an algorithm based on some performance metric

$$\mathcal{P}(\mathcal{O}_f, \mathbf{w}_0, \dots, \mathbf{w}_t, \mathbf{w}_\star), \quad \text{where} \quad \mathbf{w}_\star \in \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}} f(\mathbf{w})$$

is assumed to exist and supplied to the performance metric (which the algorithm cannot access). It will also be important to ensure

$$\|\mathbf{w}_0 - \mathbf{w}_\star\|_2 \le R$$

for some given constant R, for otherwise it may take forever for any algorithm to make any progress.

Example 18.4: First-order oracle in black-box optimization

For example, we may let $\mathcal{F} = \mathcal{F}_{\sigma, \mathsf{L}^{[1]}}$ be σ-strongly convex and $\mathsf{L}^{[1]}$-smooth functions (w.r.t. the ℓ_2 norm) on $\bar{\mathsf{W}} = \mathsf{W} = \mathbb{R}^d$. A first-order oracle returns the function value and gradient at the current iterate:

$$\mathcal{O}^1_f(\mathbf{w}_0, \dots, \mathbf{w}_{t-1}) = \mathcal{O}^1_f(\mathbf{w}_{t-1}) = \big(f(\mathbf{w}_{t-1}), \nabla f(\mathbf{w}_{t-1})\big).$$

All of our algorithms so far fall into this setting. With this oracle, we may choose the following performance metrics:

$$\begin{aligned}
\mathcal{P}(\mathcal{O}^1_f, \mathbf{w}_0, \dots, \mathbf{w}_t, \mathbf{w}_\star) &= f(\mathbf{w}_t) - f(\mathbf{w}_\star), \text{ or } \min_{0 \le s \le t} f(\mathbf{w}_s) - f(\mathbf{w}_\star)\\
\mathcal{P}(\mathcal{O}^1_f, \mathbf{w}_0, \dots, \mathbf{w}_t, \mathbf{w}_\star) &= \|\nabla f(\mathbf{w}_t)\|_2, \text{ or } \min_{0 \le s \le t} \|\nabla f(\mathbf{w}_s)\|_2\\
\mathcal{P}(\mathcal{O}^1_f, \mathbf{w}_0, \dots, \mathbf{w}_t, \mathbf{w}_\star) &= \|\mathbf{w}_t - \mathbf{w}_\star\|_2.
\end{aligned}$$

A local oracle is one that satisfies:

$$\forall t, \quad \mathcal{O}_f(\mathbf{w}_0, \dots, \mathbf{w}_{t-1}) = \mathcal{O}_g(\mathbf{w}_0, \dots, \mathbf{w}_{t-1})$$

whenever f and g agree on a neighborhood around $\mathbf{w}_0, \dots, \mathbf{w}_{t-1}$. Obviously, any-order oracle is local.

Definition 18.5: Performance estimation problem (PEP) (Drori and Teboulle, 2014)

Remarkably, the worst-case performance of an optimization algorithm can itself be formulated as an optimization problem, as noted by Drori and Teboulle (2014):

$$\begin{aligned}\mathfrak{p}(\mathcal{F},\mathcal{O},\mathcal{P},T,R,\mathcal{A}) = \sup_{f,\mathbf{w}_0,\dots,\mathbf{w}_T,\mathbf{w}_\star} \quad & \mathcal{P}(\mathcal{O}_f,\mathbf{w}_0,\dots,\mathbf{w}_T,\mathbf{w}_\star)\\ \text{s.t.} \quad & f\in\mathcal{F}\\ & \mathbf{w}_\star\in\operatorname{argmin} f\\ & (\mathbf{w}_1,\dots,\mathbf{w}_T)=\mathcal{A}_T(\mathbf{w}_0,\mathcal{O}_f)\\ & \|\mathbf{w}_0-\mathbf{w}_\star\|_2\le R,\end{aligned}$$

based on which we can also define the minimax optimal performance among all possible algorithms:

$$\underline{\mathfrak{p}}(\mathcal{F},\mathcal{O},\mathcal{P},T,R) := \left[\inf_{\mathcal{A}}\ \mathfrak{p}(\mathcal{F},\mathcal{O},\mathcal{P},T,R,\mathcal{A})\right] = \inf_{\mathcal{A}}\sup_{f\in\mathcal{F}}\ \mathcal{P}(\mathcal{O}_f,\mathbf{w}_0,\dots,\mathbf{w}_T,\mathbf{w}_\star), \tag{18.1}$$

subject to the constraints on $\mathbf{w}$'s above. The challenge is that solving $\mathfrak{p}$ is an infinite-dimensional optimization problem (due to the function class $\mathcal{F}$)! Fortunately, we only run the algorithm $\mathcal{A}$ for finitely many steps (i.e. T), which allows us to relativize to a finite dimensional problem.

Drori, Y. and M. Teboulle (2014). "Performance of first-order methods for smooth convex minimization: a novel approach". *Mathematical Programming*, vol. 145, pp. 451–482.

Alert 18.6: Ordering matters!

We emphasize that the maximin complexity

$$\overline{\mathfrak{p}}(\mathcal{F},\mathcal{O},\mathcal{P},T,R) := \sup_{f\in\mathcal{F}}\inf_{\mathcal{A}}\ \mathcal{P}(\mathcal{O}_f,\mathbf{w}_0,\dots,\mathbf{w}_T,\mathbf{w}_\star)=0,$$

which amounts to swapping the inf and sup in (18.1), is trivial. The reason is that for any given function f, the "trivial" algorithm that simply outputs $\mathbf{w}_t\equiv\mathbf{w}_\star$ cannot be beaten. (If one feels like we are cheating by allowing an algorithm to access $\mathbf{w}_\star$, consider the uncountable set of algorithms that output each $\mathbf{w}\in\mathbb{R}^d$ all the time. One of them would have guessed $\mathbf{w}_\star$ correctly, no matter what.)

Whence it is also clear that even in the minimax complexity (18.1), the function class $\mathcal{F}$ must be sufficiently rich, otherwise an algorithm could just be lucky in "guessing" the minimizer.

Example 18.7: PEP with first-order oracle

Indeed, with the first-order oracle $\mathcal{O}_f^1$ in Example 18.4, we obtain the following equivalent problem:

$$\begin{aligned}\mathfrak{p}(\mathcal{F},\mathcal{O}^1,\mathcal{P},T,R,\mathcal{A}) = \sup_{\{\mathbf{w}_t,f_t,\mathbf{g}_t\},\ f} \quad & \mathcal{P}(\{\mathbf{w}_t,f_t,\mathbf{g}_t\})\\ \text{s.t.} \quad & \exists f\in\mathcal{F}\text{ with }\mathcal{O}_f^1(\mathbf{w}_t)=(f_t,\mathbf{g}_t),\quad \forall t\in\{0,\dots,T,\star\}\\ & \mathbf{g}_\star=\mathbf{0}\\ & (\mathbf{w}_1,\dots,\mathbf{w}_T)=\mathcal{A}_t\big(\mathbf{w}_0,(f_0,\mathbf{g}_0),\dots,(f_{T-1},\mathbf{g}_{T-1})\big)\\ & \|\mathbf{w}_0-\mathbf{w}_\star\|_2\le R.\end{aligned} \tag{18.2}$$

The performance metrics mentioned in Example 18.4 can be explicitly written as:

$$f_T-f_\star \text{ or } \min_{0\le t\le T} f_t-f_\star,\quad \|\mathbf{g}_T\|_2 \text{ or } \min_{0\le t\le T}\|\mathbf{g}_t\|_2,\quad \|\mathbf{w}_T-\mathbf{w}_\star\|_2.$$

The algorithm constraint can often be simplified as a linear system for many existing first-order algorithms (examples will follow). The functional constraint basically amounts to the following interpolation problem:

find function $f \in \mathcal{F}$ with given function value f_t and gradient $\mathbf{g}_t$ at point $\mathbf{w}_t$ for all $t \in \{0, \dots, T, \star\}$,

whose solution is well-known for certain function classes!

Theorem 18.8: Interpolating $\mathsf{L}^{[1]}$-smooth convex function (Taylor et al., 2017)

There exists an $\mathsf{L} = \mathsf{L}^{[1]}$-smooth convex function f with given function value f_t and gradient $\mathbf{g}_t$ at point $\mathbf{w}_t$ for all $t \in \mathsf{T}$ iff

$$\forall s, t \in \mathsf{T}, \quad f_s \geq f_t + \langle \mathbf{w}_s - \mathbf{w}_t, \mathbf{g}_t \rangle + \tfrac{1}{2\mathsf{L}} \|\mathbf{g}_s - \mathbf{g}_t\|_2^2. \tag{18.3}$$

We allow $\mathsf{L} = \infty$ above in which case L-smoothness is vacuous.

Proof: This result is well-known for $\mathsf{L} = \infty$. For $\mathsf{L} < \infty$, a duality argument based on Alert 3.25 suffices for the reduction to $\mathsf{L} = \infty$. ■

It then follows immediately that the iff condition for interpolating a σ-weakly convex and L-smooth function is:

$$\forall s, t \in \mathsf{T}, \quad f_s - \sigma q_s \geq f_t - \sigma q_t + \langle \mathbf{w}_s - \mathbf{w}_t, \mathbf{g}_t - \sigma \mathbf{w}_t \rangle + \tfrac{1}{2(\mathsf{L}-\sigma)} \|\mathbf{g}_s - \sigma \mathbf{w}_s - \mathbf{g}_t + \sigma \mathbf{w}_t\|_2^2,$$

where $q_t = q(\mathbf{w}_t) = \frac{1}{2}\|\mathbf{w}_t\|_2^2$. Introducing

$$\begin{aligned} \mathsf{D}_\sigma(s,t) &:= f_s - f_t - \langle \mathbf{w}_s - \mathbf{w}_t, \mathbf{g}_t \rangle - \tfrac{\sigma}{2}\|\mathbf{w}_s - \mathbf{w}_t\|_2^2 \\ \mathsf{D}^*_{1/\sigma}(s,t) &:= f_t - f_s - \langle \mathbf{w}_t - \mathbf{w}_s, \mathbf{g}_s \rangle - \tfrac{1}{2\sigma}\|\mathbf{g}_s - \mathbf{g}_t\|_2^2, \end{aligned}$$

we may simplify the iff condition as:

$$\forall s, t \in \mathsf{T}, \quad \mathsf{D}_\sigma(s,t) + \tfrac{\sigma}{\mathsf{L}} \mathsf{D}^*_{1/\sigma}(s,t) \geq 0. \tag{18.4}$$

We note that condition (18.3) is clearly necessary, and quite pleasantly it turns out to be also sufficient.

Taylor, A. B., J. M. Hendrickx, and F. Glineur (2017). "Smooth strongly convex interpolation and exact worst-case performance of first-order methods". *Mathematical Programming*, vol. 161, pp. 307–345.

Example 18.9: PEP for function class $\mathcal{F}_{\sigma,\mathsf{L}^{[1]}}$ with first-order oracle

With (18.4), we can now further simplify (18.2):

$$\begin{aligned} \mathfrak{p}(\mathcal{F}_{\sigma,L}, \mathcal{O}^1, \mathcal{P}, T, R, \mathcal{A}) = \sup_{\{\mathbf{w}_t, f_t, \mathbf{g}_t\}} \quad & \mathcal{P}(\{\mathbf{w}_t, f_t, \mathbf{g}_t\}) \\ \text{s.t.} \quad & \mathsf{D}_\sigma(s,t) + \tfrac{\sigma}{\mathsf{L}} \mathsf{D}^*_{1/\sigma}(s,t) \geq 0, \quad \forall s, t \in \{0, \dots, T, \star\} \\ & \mathbf{g}_\star = \mathbf{0} \\ & (\mathbf{w}_1, \dots, \mathbf{w}_T) = \mathcal{A}_t\big(\mathbf{w}_0, (f_0, \mathbf{g}_0), \dots, (f_{T-1}, \mathbf{g}_{T-1})\big) \\ & \|\mathbf{w}_0 - \mathbf{w}_\star\|_2 \leq R, \end{aligned}$$

which is completely finite dimensional now! The only thing left to specify is the algorithm.

Remark 18.10: Simplification

We note that if the algorithm is translation equivariant w.r.t. the initializer $\mathbf{w}_0$ and the function class $\mathcal{F}$ is translation invariant, i.e.

$$\forall \mathbf{z}, f \in \mathcal{F},\ \mathcal{A}(\mathbf{w}_0 + \mathbf{z}, \mathcal{O}_f) = \mathcal{A}(\mathbf{w}_0, \mathcal{O}_f) + \mathbf{z},\quad f(\cdot + \mathbf{z}) \in \mathcal{F},$$

then we may simply take $\mathbf{w}_\star = \mathbf{0}$. If the algorithm and function class are shift invariant, i.e., $\mathcal{A}(\mathbf{w}_0, \mathcal{O}_f) = \mathcal{A}(\mathbf{w}_0, \mathcal{O}_{f+c})$ and $f + c \in \mathcal{F}$ whenever $f \in \mathcal{F}$, then we may also assume $f_\star = 0$. Of course, we implicitly assumed that translation or shifting does not affect the performance measure.

The above conditions are satisfied by the fixed step-size first-order methods:

$$\mathbf{w}_t = \mathbf{w}_0 - \sum_{s=1}^{t} h_{t,s} \mathbf{g}_{s-1}, \tag{18.5}$$

where the matrix $H \in \mathbb{R}^{T\times T}$ is *fixed* and lower-triangular (so that an update will not depend on future information).

Example 18.11: PEP for $\mathcal{F}_{\sigma,\mathsf{L}^{[1]}}$ with first-order oracle and fixed step-size

Putting everything above together we have:

$$\begin{aligned}
\mathfrak{p}(\mathcal{F}_{\sigma,\mathsf{L}^{[1]}}, \mathcal{O}^1, \mathcal{P}, T, R, H) = \sup_{\{\mathbf{w}_t, f_t, \mathbf{g}_t\}}\ & \mathcal{P}(\{\mathbf{w}_t, f_t, \mathbf{g}_t\}) \\
\text{s.t.}\quad & \mathsf{D}_\sigma(s,t) + \tfrac{\sigma}{\mathsf{L}}\mathsf{D}^*_{1/\sigma}(s,t) \geq 0,\quad s,t \in \{0,\dots,T,\star\} \\
& \mathbf{w}_\star = \mathbf{g}_\star = \mathbf{0}, f_\star = 0 \\
& \mathbf{w}_t = \mathbf{w}_0 - \sum_{s=1}^{t} h_{t,s}\mathbf{g}_{s-1}, \qquad t \in \{1,\dots,T\} \\
& \|\mathbf{w}_0\|_2 \leq R.
\end{aligned}$$

We note that by the transformation

$$f \leftarrow \tfrac{1}{R^2} f(R\cdot),\quad \mathbf{w}_t \leftarrow \mathbf{w}_t/R,\quad \mathbf{g}_t \leftarrow \mathbf{g}_t/R \quad \text{and change } \mathcal{P} \text{ accordingly},$$

we may assume $R = 1$. Moreover, using the next transformation

$$f \leftarrow f/\mathsf{L},\quad H \leftarrow \mathsf{L}H,\quad \kappa \leftarrow \sigma \leftarrow \sigma/\mathsf{L} \quad \text{and change } \mathcal{P} \text{ accordingly},$$

we may also assume $\mathsf{L} = 1$. We have thus arrived at a further simplification:

$$\begin{aligned}
\mathfrak{p}_1(\kappa, \mathcal{P}, T, H) = \sup_{\{\mathbf{w}_t, f_t, \mathbf{g}_t\}}\ & \mathcal{P}(\{\mathbf{w}_t, f_t, \mathbf{g}_t\}) \\
\text{s.t.}\quad & \mathsf{D}_\kappa(s,t) + \kappa\mathsf{D}^*_{1/\kappa}(s,t) \geq 0,\quad s,t \in \{0,\dots,T,\star\} \\
& \mathbf{w}_\star = \mathbf{g}_\star = \mathbf{0},\quad f_\star = 0 \\
& \mathbf{w}_t = \mathbf{w}_0 - \sum_{s=1}^{t} h_{t,s}\mathbf{g}_{s-1}, \qquad t \in \{1,\dots,T\} \\
& \|\mathbf{w}_0\|_2 \leq 1,
\end{aligned}$$

which is almost convex (assuming $\mathcal{P}$ is concave), except the bilinear terms $\langle \mathbf{w}, \mathbf{g}\rangle$ in D_κ and $\mathsf{D}^*_{1/\kappa}$!

Definition 18.12: PEP final formulation

We now substitute $\mathbf{w}_t$ out using (18.5) and introduce the matrices

$$P = [\mathbf{w}_0, \mathbf{g}_0, \dots, \mathbf{g}_T] \in \mathbb{R}^{d\times(T+2)}, \quad Q = P^\top P \in \mathbb{S}_+^{T+2}.$$

We note that the algorithm constraint can be written as a linear constraint on f and Q (we omit the tedious formula), leading to a final rank constrained semidefinite program:

$$\mathfrak{p}_1(\kappa, \mathcal{P}, T, H) = \sup_{\mathbf{f}\in\mathbb{R}^{T+1}, Q\in\mathbb{S}_+^{T+2}} \mathcal{P}(\mathbf{f}, Q) \tag{18.6}$$

$$\text{s.t.} \quad \mathbb{L}(\mathbf{f}, Q; \kappa, H) \le \mathbf{0}, \tag{18.7}$$
$$Q_{11} \le 1,$$
$$\operatorname{rank}(Q) \le d.$$

Dropping the last rank constraint gives us an upper bound on the worst-case performance of a fixed step-size first-order algorithm (determined by H). Of course, the upper bound is attained when $d \ge T + 2$.

Definition 18.13: Finding the optimal step-size (Drori and Teboulle, 2014)

We note that the linear constraint (18.7) is also convex quadratic in H, hence we may try to find the optimal step-size by minimizing the worst-case complexity:

$$\inf_{H\in\mathbb{R}^{T\times T}} \mathfrak{p}_1(\kappa, \mathcal{P}, T, H), \quad H \text{ is lower triangular},$$

which is a tractable convex problem! This idea has led to some shocking analyses and algorithms.

Drori, Y. and M. Teboulle (2014). "Performance of first-order methods for smooth convex minimization: a novel approach". *Mathematical Programming*, vol. 145, pp. 451–482.

Remark 18.14: Extensions

The PEP formulation (18.6) has been extended to line search (De Klerk et al., 2017), composite problems (Taylor et al., 2017, 2018), Newton's algorithm (De Klerk et al., 2020) and more (Drori and Taylor, 2020).

De Klerk, E., F. Glineur, and A. B. Taylor (2017). "On the worst-case complexity of the gradient method with exact line search for smooth strongly convex functions". *Optimization Letters*, vol. 11, pp. 1185–1199.

Taylor, A. B., J. M. Hendrickx, and F. Glineur (2017). "Exact Worst-Case Performance of First-Order Methods for Composite Convex Optimization". *SIAM Journal on Optimization*, vol. 27, no. 3, pp. 1283–1313.

— (2018). "Exact Worst-Case Convergence Rates of the Proximal Gradient Method for Composite Convex Minimization". *Journal of Optimization Theory and Applications*, vol. 178, pp. 455–476.

De Klerk, E., F. Glineur, and A. B. Taylor (2020). "Worst-Case Convergence Analysis of Inexact Gradient and Newton Methods Through Semidefinite Programming Performance Estimation". *SIAM Journal on Optimization*, vol. 30, no. 3, pp. 2053–2082.

Drori, Y. and A. B. Taylor (2020). "Efficient first-order methods for convex minimization: A constructive approach". *Mathematical Programming*, vol. 184, pp. 183–220.

Alert 18.15: Algorithms live in the linear span

We will make the following simplifying assumption (Nesterov, 2018):

$$\forall t, \quad \mathbf{w}_{t+1} \in \operatorname{span}\{\mathbf{w}_0, \nabla f(\mathbf{w}_0), \dots, \nabla f(\mathbf{w}_t)\}. \tag{18.8}$$

This assumption is not necessary but it greatly simplifies our proof and most algorithms satisfy it anyway.

Nesterov, Y. (2018). "Lectures on Convex Optimization". 2nd. Springer.

Theorem 18.16: Lower bound for nonsmooth minimization (Drori and Teboulle, 2016)

For any $\mathsf{L} > 0$, $T \le d-1$, initializer $\mathbf{w}_0 \in \mathbb{R}^d$, and any first-order algorithm $\mathcal{A}$ ~~that satisfies the linear span assumption (18.8)~~, there exists an $\mathsf{L} = \mathsf{L}^{[0]}$-Lipschitz continuous (w.r.t. $\|\cdot\|_2$) and convex function f, a first-order oracle $\mathcal{O}_f$ and minimizer $\mathbf{w}_\star$, such that the sequence $\{\mathbf{w}_t\}$ generated by algorithm $\mathcal{A}$ suffers:

$$f(\mathbf{w}_T) \ge f_\star + \frac{\mathsf{L}\cdot\|\mathbf{w}_\star - \mathbf{w}_0\|_2}{\sqrt{T+1}}.$$

Proof: Let us consider the following functions on our universe $\mathbb{R}^d$, with $T+1 \le d$:

$$\begin{aligned}\ell_1(\mathbf{w}) &= \max_{j=1,\dots,T+1} w_j\\ \ell_0(\mathbf{w}) &= \|\mathbf{w}\|_2 - 1 - \tfrac{1}{\sqrt{T+1}}\\ f_{T+1}(\mathbf{w}) &= \max\{\ell_1(\mathbf{w}), \ell_0(\mathbf{w})\}.\end{aligned}$$

We verify that f_{T+1} is convex and 1-Lipschitz continuous w.r.t. the ℓ_2 norm.

To minimize f_{T+1}, we find that ℓ_1 only depends on the indices j such that $w_j = \ell_1(\mathbf{w})$, while ℓ_0 motivates us to set any other w_j to 0. Thus, at minimum $f_{T+1}(\mathbf{w}_\star) = \max\{\alpha, |\alpha|\sqrt{t} - 1 - \frac{1}{\sqrt{T+1}}\}$, where α is the maximum value in $\mathbf{w}_\star$ and $t \le T+1$ is the number of ties. It is obvious then that we should set $t = T+1$ and $\alpha = -\frac{1}{\sqrt{T+1}}$, i.e.

$$\mathbf{w}_\star = -\tfrac{1}{\sqrt{T+1}}(\underbrace{1,\dots,1}_{T+1},0,\dots,0), \quad f_\star = -\tfrac{1}{\sqrt{T+1}}.$$

Consider now starting with $\mathbf{w}_0 = \mathbf{0}$ and run algorithm $\mathcal{A}$ to minimize f_{T+1}. Clearly, we have

$$\|\mathbf{w}_\star - \mathbf{w}_0\|_2 = 1 \le 1.$$

At each $\mathbf{w}$, a "resisting oracle" $\mathcal{O}_f$ supplies the subgradient $\mathbf{g} \in \partial f(\mathbf{w})$:

$$\mathbf{g} = \begin{cases}\mathbf{e}_k, \quad \text{where} \quad k = \min\{1 \le j \le T+1 : w_j = \ell_1(\mathbf{w}_t)\}, & \text{if } \ell_1(\mathbf{w}) \ge \ell_0(\mathbf{w})\\ \mathbf{w}/\|\mathbf{w}\|, & \text{if } \ell_1(\mathbf{w}) < \ell_0(\mathbf{w})\end{cases},$$

which has the following consequence:

$$\forall j \ge t+1,\ w_j = 0 \implies \forall j \ge t+2,\ g_j = 0.$$

It then follows from our linear span assumption (18.8) that after t iterations, only the first t entries in $\mathbf{w}_t$ can be nonzero. Therefore,

$$f_{T+1}(\mathbf{w}_T) - f_\star \ge \ell_1(\mathbf{w}_T) + \tfrac{1}{\sqrt{T+1}} \ge \tfrac{1}{\sqrt{T+1}},$$

since the last entry in $\mathbf{w}_T$ is 0.

Lastly, applying the transformation

$$f_{T+1}(\mathbf{w}) \leftarrow (\mathsf{L}f_{T+1}R)(\mathbf{w} - \mathbf{w}_0) := \mathsf{L}Rf_{T+1}(\tfrac{\mathbf{w}-\mathbf{w}_0}{R}), \quad \text{where} \quad R = \|\mathbf{w}_0 - \mathbf{w}_\star\|_2,$$

completes our proof. ■

Drori, Y. and M. Teboulle (2016). "An optimal variant of Kelley's cutting-plane method". *Mathematical Programming*, vol. 160, pp. 321–351.

Remark 18.17: Immediate consequences

A few immediate remarks with regard to Theorem 18.16:

- Quite remarkably, the subgradient Algorithm 4.14, along with some other variants (e.g. Drori and Teboulle, 2016), achieves the above lower bound even with matching constants! (Simply choose $\eta_t \equiv \frac{\mathsf{L}\|\mathbf{w}_0-\mathbf{w}_\star\|_2}{\sqrt{T+1}}$ in Theorem 4.17.)
- The condition $T \le d-1$ is applicable when the dimension is huge or when we are interested in the initial phase of our algorithm. When $T \ge d-1$, dimension-dependent lower bounds can also be proved in a similar way.
- Not surprisingly, the worst-case function f_{T+1} depends on T, which needs to be fixed beforehand.
- The proof leaves open the possibility when an algorithm chooses its subgradient from the subdifferential carefully, i.e. with a "friendly" oracle instead of a resisting one. Note however that choosing the minimum-norm subgradient does not help. On the other hand, choosing a subgradient that is "maximal" in some sense can also be problematic, since these subgradients are troublesome even when we are at the minimizer!

Drori, Y. and M. Teboulle (2016). "An optimal variant of Kelley's cutting-plane method". *Mathematical Programming*, vol. 160, pp. 321–351.

Theorem 18.18: Lower complexity bound for smooth minimization (Nesterov, 2018)

For any $\mathsf{L} > 0$, $T \le \frac{d-1}{2}$, initializer $\mathbf{w}_0 \in \mathbb{R}^d$, and any first-order algorithm $\mathcal{A}$ ~~that satisfies the linear span assumption (18.8)~~, there exists an $\mathsf{L} = \mathsf{L}^{[1]}$-smooth (w.r.t. $\|\cdot\|_2$) and convex function f with minimizer $\mathbf{w}_\star$, such that the sequence $\{\mathbf{w}_t\}$ generated by algorithm $\mathcal{A}$ suffers:

$$f(\mathbf{w}_T) \ge f_\star + \frac{3\mathsf{L}\|\mathbf{w}_0-\mathbf{w}_\star\|_2^2}{32(T+1)^2}$$
$$\|\mathbf{w}_T-\mathbf{w}_\star\|_2^2 \ge \tfrac{1}{8}\|\mathbf{w}_0-\mathbf{w}_\star\|_2^2.$$

Proof: The proof is similar to that of Theorem 18.16. We consider the quadratic function

$$f_t(\mathbf{w}) = \frac{1}{8}\left[w_1^2 + w_t^2 + \sum_{j=1}^{t-1}(w_j - w_{j+1})^2\right] - \frac{1}{4}w_1.$$

It is easy to verify that

$$0 \le \frac{1}{4}\left[z_1^2 + z_t^2 + \sum_{j=1}^{t-1}(z_j - z_{j+1})^2\right] = \langle \nabla^2 f_t(\mathbf{w})\mathbf{z}, \mathbf{z}\rangle \le \frac{1}{4}\left[z_1^2 + z_t^2 + \sum_{j=1}^{t-1} 2z_j^2 + 2z_{j+1}^2\right] \le \|\mathbf{z}\|_2^2,$$

i.e. f_t is convex and 1-smooth. We set the derivative to 0 to compute a minimizer and minimum value of f_t:

$$\left.\begin{aligned} &w_1 + w_1 - w_2 - 1 && = 0\\ &\forall j = 2,\dots,t-1,\quad w_j - w_{j-1} + w_j - w_{j+1} && = 0\\ &w_t + w_t - w_{t-1} && = 0 \end{aligned}\right\} \implies (w_\star)_j = \begin{cases} 1 - \frac{j}{t+1}, & \text{if } j \le t\\ 0, & \text{otherwise}\end{cases},$$

$$\|\mathbf{w}\|_2^2 = \sum_{j=1}^{t}(1-\tfrac{j}{t+1})^2 = t - t + \tfrac{t(2t+1)}{6(t+1)} \le \tfrac{t-1/4}{3},$$

$$(f_t)_\star = \frac{1}{8}\left[\frac{t^2}{(t+1)^2} + \frac{1}{(t+1)^2} + (t-1)\frac{1}{(t+1)^2}\right] - \frac{1}{4}\frac{t}{t+1} = -\frac{1}{8}\frac{t}{t+1}.$$

Now starting with $\mathbf{w}_0 = \mathbf{0}$ and apply algorithm $\mathcal{A}$ to minimize f_{2T+1}. Clearly, we have

$$\|\mathbf{w}_\star - \mathbf{w}_0\|_2 \leq \tfrac{2T+1-1/4}{3} \leq \tfrac{8T+3}{12}.$$

We verify the following consequence:

$$\forall j \geq t+1,\ w_j = 0 \implies \forall j \geq t+2,\ \nabla_j f_{2T+1}(\mathbf{w}) = 0.$$

It then follows from our linear span assumption (18.8) that after t iterations, only the first t entries in $\mathbf{w}_t$ can be nonzero. Therefore,

$$\frac{f_{2T+1}(\mathbf{w}_T) - (f_{2T+1})_\star}{\|\mathbf{w}_\star - \mathbf{w}_0\|_2^2} = \frac{f_T(\mathbf{w}_T) - (f_{2T+1})_\star}{\|\mathbf{w}_\star - \mathbf{w}_0\|_2^2} \geq \frac{-\frac{T}{8(T+1)} + \frac{2T+1}{8(2T+2)}}{\frac{8T+3}{12}} \geq \frac{3}{4(T+1)(8T+3)} \geq \frac{3}{32(T+1)^2}.$$

Lastly, tedious calculation verifies:

$$\|\mathbf{w}_T - \mathbf{w}_\star\|_2^2 \geq \sum_{j=T+1}^{2T+1} (1 - \tfrac{j}{2T+2})^2 \geq \frac{1}{8}\frac{8T+3}{12} \geq \frac{1}{8}\|\mathbf{w}_0 - \mathbf{w}_\star\|_2^2,$$

which completes our proof. ■

Nesterov, Y. (2018). "Lectures on Convex Optimization". 2nd. Springer.

Remark 18.19: Immediate consequences

We make the following immediate remarks with regard to Theorem 18.18:

- Drori (2017) improved Theorem 18.18 to: for all $T \leq d-1$, for all algorithm $\mathcal{A}$ (w/o (18.8)),

$$f(\mathbf{w}_T) - f_\star \geq \frac{\mathsf{L}^{[1]}\|\mathbf{w}_0 - \mathbf{w}_\star\|_2^2}{2\theta_T^2} \approx \frac{\mathsf{L}^{[1]}\|\mathbf{w}_0 - \mathbf{w}_\star\|_2^2}{T^2}, \quad \text{where} \quad \theta_t = \begin{cases} 1, & \text{if } t = 0 \\ \frac{1+\sqrt{1+4\theta_{t-1}^2}}{2}, & \text{if } 1 \leq t \leq T-1 \\ \frac{1+\sqrt{1+8\theta_{t-1}^2}}{2}, & \text{if } t = T \end{cases},$$

  which is tight even with matching constants!

- There is a significant gap between the lower bound and our existing upper bound: recall from Theorem 2.21 the convergence rate is upper bounded by $O(1/T)$, which is an order of magnitude worse than what the lower bound indicates! This gap has directly inspired one of the most profound results in gradient algorithms, as we will see in the next lecture.

Drori, Y. (2017). "The exact information-based complexity of smooth convex minimization". *Journal of Complexity*, vol. 39, pp. 1–16.

Alert 18.20: How to interpret lower bounds?

Lower complexity bounds as above reveal the worst-case behaviour of any algorithm. An algorithm that achieves the lower bound is optimal in the worst-case sense. However, it does not mean an optimal algorithm will necessarily converge faster than a suboptimal one on any particular problem instance. In fact, since our two lower bounds are constructive, they do not tell us anything when the particular function that we aim to minimize is not the ones constructed in the proof. In contrast, an upper bound on the convergence rate would always hold regardless of what function we minimize (as long as it falls into the class). In short:

- An upper bound is for a particular algorithm and applicable to all functions (within a class).

- A lower bound is for a particular function and applicable to all algorithms (within a class).

We have also assumed an oracle access to the objective function. While this is reasonable in certain settings (e.g. when the function is private or overly complicated), it is perhaps too strong in most settings. As such, a natural question arises:

> Can we *strictly* improve the convergence rate if we know more about the function class we aim to minimize? If so, how?

The answer is surprisingly (or not surprisingly) yes, and we will see examples in later lectures.

Remark 18.21: Lower bound through smoothing

Guzmán and Nemirovski (2015) provided lower bounds for function class of milder smoothness, based on the idea of smoothing, which we will discuss in a later lecture.

Guzmán, C. and A. Nemirovski (2015). "On lower complexity bounds for large-scale smooth convex optimization". *Journal of Complexity*, vol. 31, pp. 1–14.