Complexity analysis in optimization (1/6)

Clément W. Royer

ED SDOSE Doctoral course

January 30, 2025

Warm-up

Resources

- Resources via my webpage:
  `https://www.lamsade.dauphine.fr/~croyer`
- Slides, virtual boards, etc.

Logistics

- 4 sessions Thursday 1.45-5pm Jan. 30, Feb. 06, Feb. 13, Feb. 20.
- 2 sessions Wednesday 1.45-5pm Feb. 12, Feb. 19.
- Will try to cover relatively independent subjects during each session!

Breaking the ice

My ID card

- I'm Clément!
- At Dauphine since 2019.
- I got here via RER C.

Breaking the ice

My ID card

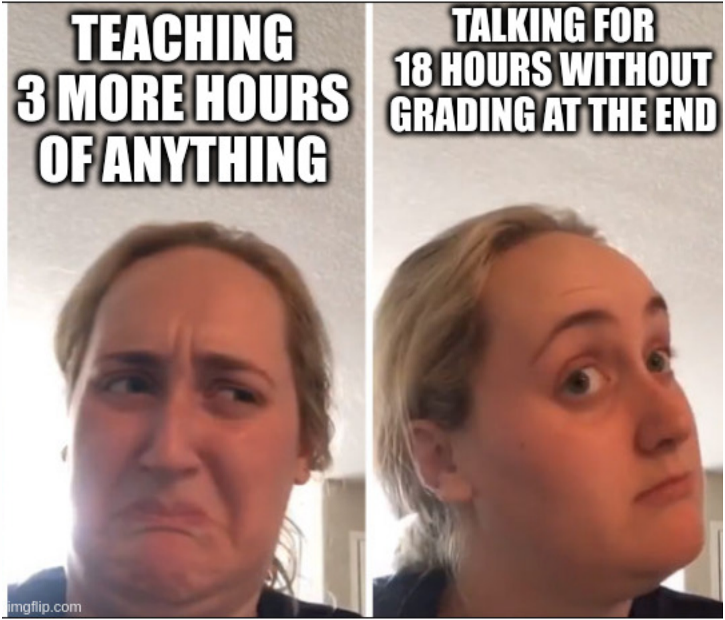
- I'm Clément!
- At Dauphine since 2019.
- I got here via RER C.

What about you?

- First name.
- Your first year at Dauphine.
- How did you get to Dauphine today?

Why did I agree to give this course?

Why did I agree to give this course?



Why did I agree to give this course?

- This is my field of study!
- I am preparing my HDR on this!
- I think this is important (and cool).

What will I talk about?

Detailed (tentative) syllabus:

- **Part 1** Basics of continuous optimization: Derivatives, optimality conditions, classes of functions and algorithms. Introduction to complexity in continuous optimization: convergence rates, worst-case complexity, connection to NP-hardness.
- **Part 2** Complexity in convex optimization: basic complexity results for convex and strongly convex functions, acceleration, upper/lower bounds. Complexity in nonconvex optimization: results for first-order methods and limitations, second-order methods and beyond, upper/lower bounds.
- **Part 3** Stochastic optimization: complexity results for stochastic gradient methods, lower/upper bounds. Games and saddle point problems: Complexity of gradient descent-ascent, open problems. Complexity in continuous VS discrete optimization: submodular optimization, recent advances in integer programming.

- Initial syllabus: Way too ambitious!
  Focus on (six) results.

What will I talk about?

Detailed (tentative) syllabus:

- **Part 1** Basics of continuous optimization: Derivatives, optimality conditions, classes of functions and algorithms. Introduction to complexity in continuous optimization: convergence rates, worst-case complexity, connection to NP-hardness.
- **Part 2** Complexity in convex optimization: basic complexity results for convex and strongly convex functions, acceleration, upper/lower bounds. Complexity in nonconvex optimization: results for first-order methods and limitations, second-order methods and beyond, upper/lower bounds.
- **Part 3** Stochastic optimization: complexity results for stochastic gradient methods, lower/upper bounds. Games and saddle point problems: Complexity of gradient descent-ascent, open problems. Complexity in continuous VS discrete optimization: submodular optimization, recent advances in integer programming.

- Initial syllabus: Way too ambitious!
  Focus on (six) results.
- You can learn about complexity by others.
  I'll give you pointers to the literature!

What will I talk about?

Detailed (tentative) syllabus:

- Part 1 Basics of continuous optimization: Derivatives, optimality conditions, classes of functions and algorithms. Introduction to complexity in continuous optimization: convergence rates, worst-case complexity, connection to NP-hardness.
- Part 2 Complexity in convex optimization: basic complexity results for convex and strongly convex functions, acceleration, upper/lower bounds. Complexity in nonconvex optimization: results for first-order methods and limitations, second-order methods and beyond, upper/lower bounds.
- Part 3 Stochastic optimization: complexity results for stochastic gradient methods, lower/upper bounds. Games and saddle point problems: Complexity of gradient descent-ascent, open problems. Complexity in continuous VS discrete optimization: submodular optimization, recent advances in integer programming.

- Initial syllabus: Way too ambitious!
  Focus on (six) results.
- You can learn about complexity by others.
  I'll give you pointers to the literature!
- It will probably never be useful to you.
  I'll tell you how it was useful to me.

What I intend to do

- Every session will have a key result.
- We'll prove it, possibly illustrate it, discuss what it means.
- I'll have a story to tell about this.

Today's roadmap

1. Optimization background
2. Complexity of gradient descent
3. Worst-case example
4. Newton's method and backstory

Roadmap

1. Optimization background
2. Complexity of gradient descent
3. Worst-case example
4. Newton's method and backstory

Basic optimization problem

Unconstrained smooth minimization:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}}\, f(x)$$

Basic optimization problem

Unconstrained smooth minimization:

$$\operatorname*{minimize}_{x \in \mathbb{R}^n} f(x)$$

Assumptions on f

- f bounded below: $f(x) \geq f_{\mathrm{low}}$.
- f $\mathcal{C}^1$ (continuously differentiable):
  At every $x \in \mathbb{R}^n$, the **gradient** $\nabla f(x) \in \mathbb{R}^n$ exists.

Basic optimization problem

Unconstrained smooth minimization:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}}\, f(x)$$

Assumptions on f

- f bounded below: $f(x) \geq f_{\mathrm{low}}$.
- f $\mathcal{C}^1$ (continuously differentiable):
  At every $x \in \mathbb{R}^n$, the **gradient** $\nabla f(x) \in \mathbb{R}^n$ exists.
- f $\mathcal{C}^{1,1}_L$ (aka L-smooth)

$$\forall (x,y) \in (\mathbb{R}^n)^2, \|\nabla f(y) - \nabla f(x)\| \leq L\|y-x\|.$$

  The gradient is L-Lipschitz continuous.

Our goal: Solving the problem?

$$\underset{x \in \mathbb{R}^n}{\text{minimize}}\ f(x)$$

Definition: $x^* \in \mathbb{R}^n$ is

- a global minimum of f if $f(x^*) \leq f(x)\ \forall x \in \mathbb{R}^n$.
- a local minimum of f if $f(x^*) \leq f(x)\ \forall x$ close enough to x^*.
- A first-order stationary point of f satisfies $\|\nabla f(x^*)\| = 0$.

Our goal: Solving the problem?

$$\underset{x \in \mathbb{R}^n}{\text{minimize}}\, f(x)$$

Definition: $x^* \in \mathbb{R}^n$ is

- a global minimum of f if $f(x^*) \le f(x)\ \forall x \in \mathbb{R}^n$.
- a local minimum of f if $f(x^*) \le f(x)\ \forall x$ close enough to x^*.
- A first-order stationary point of f satisfies $\|\nabla f(x^*)\| = 0$.

A first "complexity" result

Under our assumptions,

Our goal: Solving the problem?

$$\underset{x \in \mathbb{R}^n}{\text{minimize}}\ f(x)$$

Definition: $x^* \in \mathbb{R}^n$ is

- a global minimum of f if $f(x^*) \leq f(x)\ \forall x \in \mathbb{R}^n$.
- a local minimum of f if $f(x^*) \leq f(x)\ \forall x$ close enough to x^*.
- A first-order stationary point of f satisfies $\|\nabla f(x^*)\| = 0$.

A first "complexity" result

Under our assumptions,

- Finding global minima of f is NP-hard in general.

Our goal: Solving the problem?

$$\operatorname*{minimize}_{x \in \mathbb{R}^n} f(x)$$

Definition: $x^* \in \mathbb{R}^n$ is

- a global minimum of f if $f(x^*) \le f(x)\ \forall x \in \mathbb{R}^n$.
- a local minimum of f if $f(x^*) \le f(x)\ \forall x$ close enough to x^*.
- A first-order stationary point of f satisfies $\|\nabla f(x^*)\| = 0$.

A first "complexity" result

Under our assumptions,

- Finding global minima of f is NP-hard in general.
- Finding local minima of f is NP-hard in general.

Our real goal: Approximately solving the problem with good complexity

For any $\epsilon > 0$, $\bar{x} \in \mathbb{R}^n$ is an ϵ-stationary point if

$$\|\nabla f(\bar{x})\| \leq \epsilon.$$

Our real goal: Approximately solving the problem with good complexity

For any $\epsilon > 0$, $\bar{x} \in \mathbb{R}^n$ is an ϵ-stationary point if

$$\|\nabla f(\bar{x})\| \leq \epsilon.$$

Take-away

We can design iterative algorithms that compute an ϵ-stationary point

- In a number of iterations that scales as a polynomial in ϵ^{-1}.
- Using a number of evaluations that also scales as a polynomial in ϵ^{-1}.

Our real goal: Approximately solving the problem with good complexity

For any $\epsilon > 0$, $\bar{x} \in \mathbb{R}^n$ is an ϵ-stationary point if

$$\|\nabla f(\bar{x})\| \le \epsilon.$$

Take-away

We can design iterative algorithms that compute an ϵ-stationary point

- In a number of iterations that scales as a polynomial in ϵ^{-1}.
- Using a number of evaluations that also scales as a polynomial in ϵ^{-1}.

This is what we call complexity analysis!

A bit of history

- When I learned optimization ($\approx$ L3 courses in Dauphine):
  - Typical theory: Show that $\|\nabla f(x)\| \rightarrow 0$, with possibly fast convergence near a solution.
  - Classical results in a 1980s-1990s paper.
  - Influence from applied maths/control people.

- When I learned optimization ($\approx$ L3 courses in Dauphine):
  - Typical theory: Show that $\|\nabla f(x)\| \to 0$, with possibly fast convergence near a solution.
  - Classical results in a 1980s-1990s paper.
  - Influence from applied maths/control people.
- How I teach it now ($\approx$ M2 courses in Dauphine)
  - Given $\epsilon > 0$, how fast can you find a point such that $\|\nabla f(x)\| \leq \epsilon$?
  - Finite-time guarantees: popular since late 2000s, now very standard in continuous optimization papers.
  - Blame it on ML/Theoretical CS people!

Roadmap

Using the gradient

Problem: minimize$_{x\in\mathbb{R}^n} f(x)$, f $\mathcal{C}^{1,1}_L$.

- At any point $x \in \mathbb{R}^n$, we have

$$f(y) \approx f(x) + \nabla f(x)^{\mathrm{T}}(y-x)$$

  when y is close to x.

Using the gradient

Problem: minimize$_{x \in \mathbb{R}^n} f(x)$, f $\mathcal{C}^{1,1}_L$.

- At any point $x \in \mathbb{R}^n$, we have

$$f(y) \approx f(x) + \nabla f(x)^{\mathrm{T}}(y-x)$$

when y is close to x.

- By Lipschitz-continuity of the gradient,

$$\forall x, y, \qquad f(y) \le f(x) + \nabla f(x)^{\mathrm{T}}(y-x) + \frac{L}{2}\|y-x\|^2.$$

Optimality conditions

- If a point x is a minimum of f, then $\|\nabla f(x)\| = 0$.
- If $\|\nabla f(x)\| > 0$ and $\alpha > 0$ is small enough, then $f(x - \alpha \nabla f(x)) < f(x)$ (basis of gradient descent).

Gradient descent

Algorithm($x_0 \in \mathbb{R}^n$, $\epsilon > 0$)
For $k = 0, 1, 2, \dots$

- If $\|\nabla f(x_k)\| \leq \epsilon$, stop and return x_k.
- Otherwise, compute $\alpha_k > 0$ and set $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$.

Gradient descent

Algorithm($x_0 \in \mathbb{R}^n$, $\epsilon > 0$)
For $k = 0, 1, 2, \dots$

- If $\|\nabla f(x_k)\| \leq \epsilon$, stop and return x_k.
- Otherwise, compute $\alpha_k > 0$ and set $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$.

- Cost: Evaluating $\nabla f(x_k)$+Computing α_k.

Iteration complexity

Key lemma

If $\alpha_k < \frac{2}{L}$, then

$$f(x_k - \alpha_k \nabla f(x_k)) \leq f(x_k) - (\alpha_k - \frac{L}{2}\alpha_k^2)\|\nabla f(x_k)\|^2 < f(x_k).$$

Theorem (Nesterov '04, **Gaviano & Lera '02**)

If $\alpha_k = \alpha = \frac{1}{L}$, then the method stops after at most

$$\left\lceil \frac{2L(f(x_0) - f_{\text{low}})}{\epsilon^2} \right\rceil = \mathcal{O}(\epsilon^{-2})$$

iterations/gradient evaluations.

Line-search variant

Armijo backtracking line search

Set α_k as the largest value $\alpha \in \{1, \theta, \theta^2, \dots\}$ such that

$$f(x_k - \alpha \nabla f(x_k)) < f(x_k) - c\alpha \|\nabla f(x_k)\|^2$$

for $\theta \in (0, 1)$ and $c \in (0, 1)$.

Line search termination

The line-search terminates with

$$\alpha_k \geq \frac{2\theta(1-c)}{L}.$$

Line-search variant

Iteration complexity

With line search, the method terminates after at most

$$\underbrace{\frac{L(f(x_0) - f_{\mathrm{low}})}{2\theta\, c(1-c)}}_{C_1} \epsilon^{-2} = \mathcal{O}(\epsilon^{-2})$$

iterations.
$\Rightarrow$ Same order without knowledge of L!

Line-search variant

Iteration complexity

With line search, the method terminates after at most

$$\underbrace{\frac{L(f(x_0) - f_{\text{low}})}{2\theta\, c(1-c)}}_{C_1} \epsilon^{-2} = \mathcal{O}(\epsilon^{-2})$$

iterations.

$\Rightarrow$ Same order without knowledge of L!

Evaluation complexity

With line search, the method terminates after at most

- $C_1 \epsilon^{-2}$ gradient evaluations.
- $\log_\theta \left(\frac{2(1-c)}{L}\right) \epsilon^{-2}$ **function evaluations**.

Roadmap

From upper to lower bounds

- We have shown that gradient descent takes **at most** $\mathcal{O}(\epsilon^{-2})$ iterations to get $\|\nabla f(x_k)\| \le \epsilon$...

From upper to lower bounds

- We have shown that gradient descent takes **at most** $\mathcal{O}(\epsilon^{-2})$ iterations to get $\|\nabla f(x_k)\| \leq \epsilon$...
- ...but that bound may be far from the best possible bound: $\mathcal{O}(\epsilon^{-143535})$ is also a valid upper bound!

From upper to lower bounds

- We have shown that gradient descent takes **at most** $\mathcal{O}(\epsilon^{-2})$ iterations to get $\|\nabla f(x_k)\| \le \epsilon$...
- ...but that bound may be far from the best possible bound: $\mathcal{O}(\epsilon^{-143535})$ is also a valid upper bound!
- A line of work has focused on finding **worst-case examples** for which the bound is attained.

The Cartis-Gould-Toint example

SIAM J. OPTIM.
Vol. 20, No. 6, pp. 2833–2852

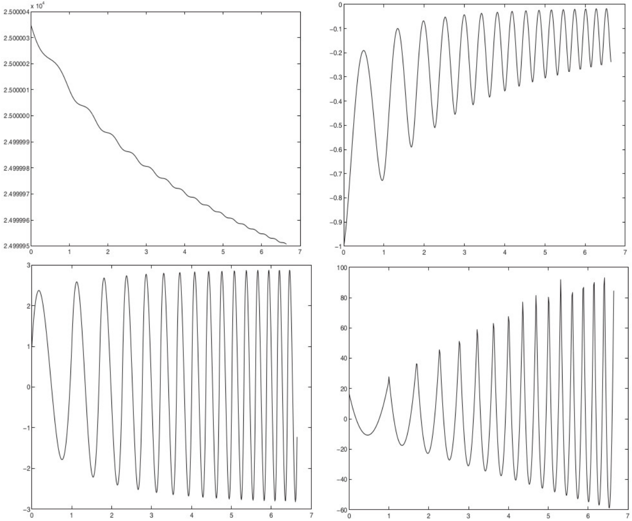
ON THE COMPLEXITY OF STEEPEST DESCENT, NEWTON'S AND REGULARIZED NEWTON'S METHODS FOR NONCONVEX UNCONSTRAINED OPTIMIZATION PROBLEMS*

C. CARTIS†, N. I. M. GOULD‡, AND PH. L. TOINT§

Abstract. It is shown that the steepest-descent and Newton's methods for unconstrained nonconvex optimization under standard assumptions may both require a number of iterations and function evaluations arbitrarily close to $O(\epsilon^{-2})$ to drive the norm of the gradient below ϵ. This shows that the upper bound of $O(\epsilon^{-2})$ evaluations known for the steepest descent is tight and that Newton's method may be as slow as the steepest-descent method in the worst case. The improved evaluation complexity bound of $O(\epsilon^{-3/2})$ evaluations known for cubically regularized Newton's methods is also shown to be tight.

- A pathological, 1-dimensional function.
- Gradient descent does take $\mathcal{O}(\epsilon^{-2})$ iterations!

How bad-looking is this function?



- "Some steps on a sandy dune" (Ph. Toint).
- Built by Hermite interpolation.

Roadmap

How did I came across these results?

- **2012**: First optimization course (as Master student).
  - Learn about gradient descent and Newton's method.
  - No complexity but local convergence/practical results.
  - Newton outperforms gradient descent!

How did I came across these results?

- **2012**: First optimization course (as Master student).
  - Learn about gradient descent and Newton's method.
  - No complexity but local convergence/practical results.
  - Newton outperforms gradient descent!
- **2013**: Master thesis on optimization
  - Start reading about complexity (but not the focus);
  - Discover the Cartis/Gould/Toint paper.
  - Learn that Newton can be as bad as gradient descent!

My history with complexity (Pt. 1)

- Semi-plenary: Coralia Cartis.
- Started reading a lot more complexity papers of hers afterwards.
- Not all were useful to me during the PhD.

Striking result: Newton's method

Newton's method

While $\|\nabla f(x_k)\| \leq \epsilon$,

$$x_{k+1} = x_k - \alpha_k \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$$

- Not always well-defined.
- Not always convergent.
- Works often well in practice.

Striking result: Newton's method

Newton's method

While $\|\nabla f(x_k)\| \le \epsilon$,

$$x_{k+1} = x_k - \alpha_k \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$$

- Not always well-defined.
- Not always convergent.
- Works often well in practice.

And yet...

- Newton's method terminates in at most $\mathcal{O}(\epsilon^{-2})$ iterations.
- There exists a function on which Newton's method takes at least $\mathcal{O}(\epsilon^{-2})$ iterations!

Summary

- **Complexity:** Certificate that you reach a given criterion in finite time.
- **Complexity analysis** Drives the modern theory of optimization algorithms.
- **Worst-case examples** Give lower bounds that match the upper bounds.

Summary

- **Complexity:** Certificate that you reach a given criterion in finite time.
- **Complexity analysis** Drives the modern theory of optimization algorithms.
- **Worst-case examples** Give lower bounds that match the upper bounds.

Next up

- This was all in the *nonconvex* setting.
- We will look (back in time) at the *convex* setting.

That's all for now!

Thank you!