Introduction to Optimization Theory

Lecture #1 - 9/15/20

MS&E 213 / CS 2690

Aaron Sidford
sidford@stanford.edu

Lecture Plan

Why?

- Prepare you for the quarter
- Help you choose which class to take

Part 1

Syllabus

- What course is and is not about
- Course setup, expectations, and plans

Part 2

Course Philosophy / Overview

- Overview of course approach
- Definitions we will use throughout quarter

Part 3

Brief Warmup Problem

- Time permitting

Thursday

Longer illustrative warmup problem.

Rest of Quarter

Build on foundations set this week.

What is this class?

Introduction to iterative algorithms

- Intro to theory of continuous optimization
- Provable guarantees for algorithm and methods solving continuous optimization problems
- Finite convergence rates of iterative methods
- Limits of efficient computation and optimization
- Structure of continuous optimization problems

What isn't this class?

- The most comprehensive optimization intro? (MS&E 211/x)
- The most focused introduction to convex analysis? (EE364 a/b)
- Source of immediate practical optimization experience

Why this class?

- Understand theory for why method work or don’t
- Guide design of optimization methods in practice
- Begin research on optimization & iterative methods
  - In some cases, the course will be at the cutting edge rather quickly.

Pre-requisites

- No optimization experience required

- Math (proofs, multivariable calculus, linear algebra, probability, etc.)

  - May re-introduce some concepts, provide references, and refresh material. However, these are not necessarily covered in class.

- **Note**: If you ever suspect that lectures are assuming more prior knowledge, please feel free to contact me. – sidford@stanford.edu

Course material

<u>Primary references</u>

- Lectures
  - Encourage to attend and participate
  - Will be recorded

<u>Additional References</u>

- Will be provided online
- Feel free to ask on Piazza

<u>Primary references</u>

- Lecture notes
  - Required reading
  - Work-in progress
  - Updating frequently
  - Feedback welcome
  - Typos / suggestions for participation credit

Expectations

Syllabus Questions?

Material and Presentations

- Stay up to date with lectures and assignments (material accumulates)
- Hope you can attend lecture and encourage participate
- Participation encouraged and possibly rewarded (in class, Piazza)
- Encourage to complete anonymous feedback

Assignments

- Psets 40% (Fridays at 5PM PST)
- Take-home midterm 25%
- Take-home final 35%

COVID and Virtual Classroom

- We are here to help you learn and succeed. Feel free to reach out.

Lecture Plan

Part 1 ✓

Syllabus

- What course is and is not about
- Course setup, expectations, and plans

Part 2

Course Philosophy / Overview

- Overview of course approach
- Definitions we will use throughout quarter

Part 3

Brief Warmup Problem

- Time permitting

Thursday

Longer illustrative warmup problem.

Rest of Quarter

Build on foundations set this week.

What's this course about?

Note

Feasible region, S, is often *infinite* in this course, this is in contrast to *discrete* or *combinatorial optimization* where S is finite.

<u>Function Minimization</u>

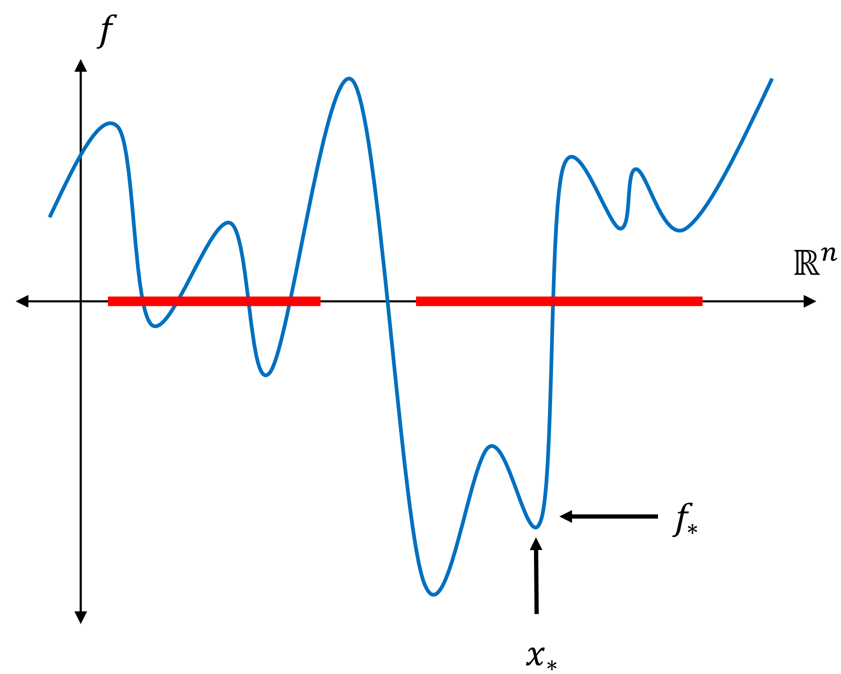
- *objective function*: $f: \mathbb{R}^n \to \mathbb{R}$
- *constraint set / feasible region*: $S \subseteq \mathbb{R}^n$

- **"Goal"**: "minimize" objective subject to constraint

$$\text{"solve"} \min_{x \in S} f(x)$$

minimum value: $f_* \stackrel{\text{def}}{=} \min_{x \in S} f(x)$

minimizer: $x_* \in \min_{x \in S} f(x)$



Common Theme: Reductions and Generality

Unconstrained Minimization

- $S = \mathbb{R}^n$ *(focus for first few weeks)*

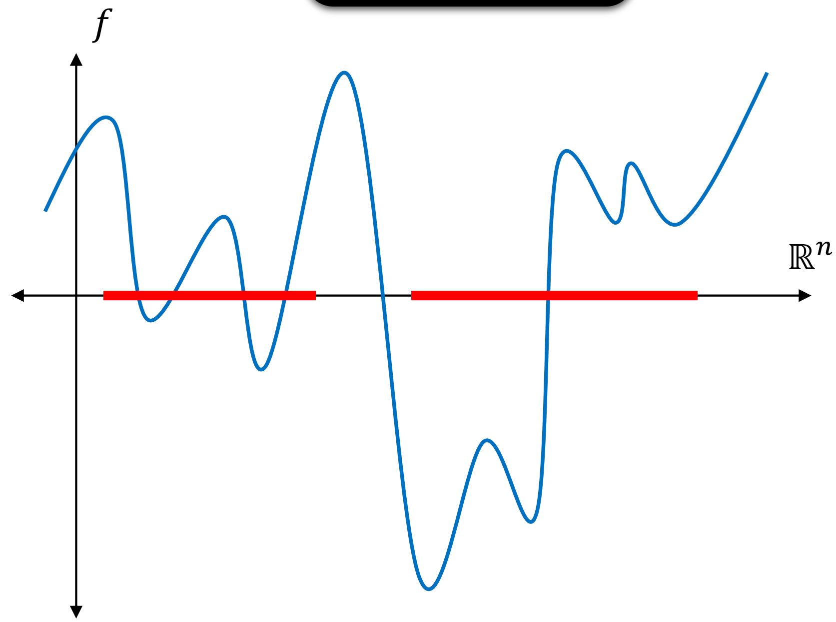
More or less difficult?

Same difficulty!

How to show problem a is no more difficult than problem b? "Reduce" b to a, i.e. use a to solve b.

Goal

$$\min_{x \in S \subseteq \mathbb{R}^n} f(x)$$



Common Theme: Reductions and Generality

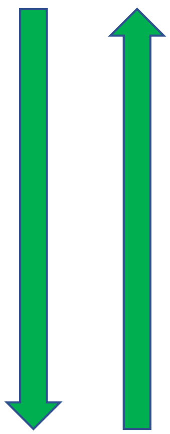
Goal

$$\min_{x \in S \subseteq \mathbb{R}^n} f(x)$$

Unconstrained Minimization (A)

- $S = \mathbb{R}^n$

Can reduce (A) to (B), i.e. (B) is as hard a (A), since (A) is a special case of (B).



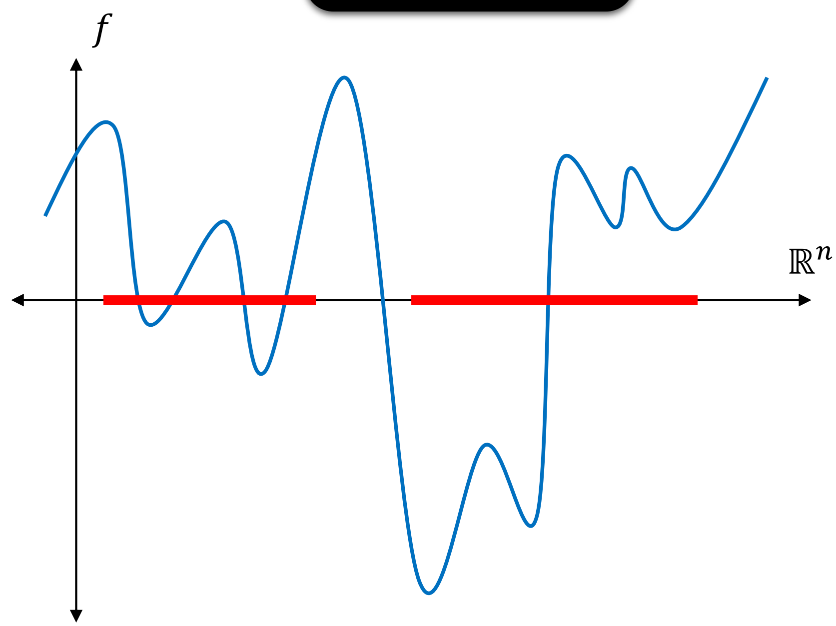
Given constrained problem f, S define penalty $\psi : \mathbb{R}^n \to \mathbb{R}$

$$\psi(x) = \begin{cases} 0 & x \in S \\ \infty & x \notin S \end{cases}$$

Obtain equivalent unconstrained minimization problem:

$$\min_{x \in S} f(x) = \min_{x \in \mathbb{R}^n} f(x) + \psi(x)$$

Constrained Minimization (B)



Common Theme: Reductions and Generality

Goal

$$\min_{x \in S \subseteq \mathbb{R}^n} f(x)$$

Function Maximization

$$\max_{x \in S \subseteq \mathbb{R}^n} f(x)$$

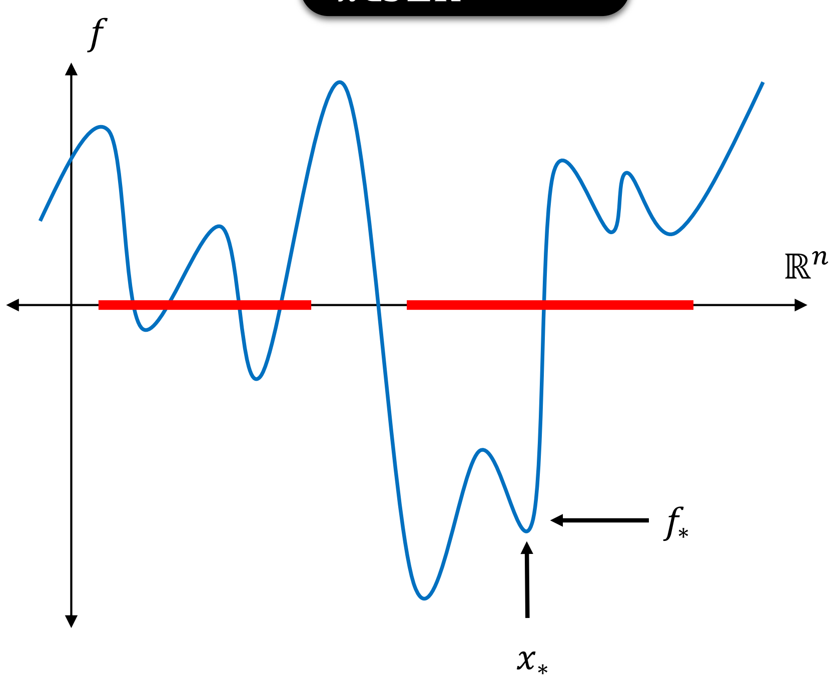
Equivalent to

$$\min_{x \in S \subseteq \mathbb{R}^n} -f(x)$$

Sums of Function

$$\max_{x \in S \subseteq \mathbb{R}^n} g(x) + h(x)$$

Define $f(x) \stackrel{\text{def}}{=} g(x) + h(x)$



Note

These reductions may be useful, but they also may lose problem structure that our methods depend on. *(We will see this later in the course).*

Goal of this Class

- Provably minimize f efficiently.
- Make minimal assumptions on f

Why?
Obtain general purpose algorithms and understand problem structure.

Questions

- How do we access f?
- What do we mean by minimize?
- What do we mean by efficiently?

How do we access f?

- Often in this class, we will not assume that we know f
- Typically we will assume a restricted *oracle model* for accessing f
- Assume a procedure that can run to get limited info regarding f

<u>Notation</u>:

"procedure" ↔ "oracle"
"run" ↔ "query"

<u>Setup</u>:

Example oracles

- “value oracle”, “evaluation oracle”, “0’th order oracle”

query $x \in \mathbb{R}^n$ → **oracle** → output $f(x) \in \mathbb{R}$

- “gradient oracle”

query $x \in \mathbb{R}^n$ → **oracle** → output $\nabla f(x) \in \mathbb{R}^n$

“gradient of f at x”

$[\nabla f(x)]_i = \frac{\partial}{\partial x_i} f(x)$ for all $i \in [n]$

- “first order oracle”

query $x \in \mathbb{R}^n$ → **oracle** → output $f(x), \nabla f(x)$

Many Oracles

- stochastic gradient oracle

query
$x \in \mathbb{R}^n$ → **oracle** → output
$g \in \mathbb{R}^n$ where $\mathbb{E}g = \nabla f(x)$

- noisy gradient oracle

query
$x \in \mathbb{R}^n$ → **oracle** → output
$g \in \mathbb{R}^n$ where $||g - \nabla f(x)|| \leq c \cdot ||g||$

We will see many throughout the class.

Why the oracle model?

Help understand the utility of problem structure and measurement.

Practical

- Sometimes don't know f and can only make observations.
- Sometimes can only make observations about f
- Evaluation may be expensive
- Maybe data is corrupted

Complexity Theory

- Can prove information theoretic lower bounds!

Theoretical

- Clarify what structure of f is being used in algorithm.

- E.g. regression

$$\min_x f(x) = \frac{1}{2}\|Ax - b\|_2^2$$

Is an algorithm using linear structure?

Is algorithm just using $\nabla f(x) = A^\top(Ax - b)$?

Goal of this Class 2.0

- **Given oracle access to f**
- Provably minimize f efficiently.
- Make minimal assumptions on f

Questions

- ~~How do we access f?~~
- What do we mean by minimize?
- What do we mean by efficiently?

Minimization? Optimization?

- For most of class, we will not exactly optimize.
- Instead, we will *approximately* optimize
- Consider different solution concepts.

ϵ-suboptimal point or a point with **ϵ-additive function error**:

- $x \in S$ s.t. $f(x) \leq f_* + \epsilon$ where $f_* = \min_{x \in S} f(x)$

ϵ-critical point:

- $x \in S$ s.t. $\|\nabla f(x)\|_2 \leq \epsilon$ where $\|y\|_2 \stackrel{\text{def}}{=} \sqrt{\sum_{i \in [n]} y_i^2}$

Efficiency?

Focus of this class

Oracle Complexity

- How many time query oracle

Happy to discuss

Runtime / Computational Complexity

- Total runtime / computational work done

Both are interesting to study and lens of #queries versus cost per query can be helpful in designing optimization algorithms (e.g. regression).

Have found very useful for research

So what do algorithms / methods look like?

Most of the oracles in this class yield local information regarding a queried point.

Idea: have algorithms iteratively repeatedly make local improvements.

This class is in part an introduction to such algorithms, often called iterative methods.

Iterative Methods (Rough Template)

- Start at initial point x_0
- For $t = 0, \dots, T-1$
  - Query oracle
  - Take "local step" to obtain x_{t+1}
  - Repeat
- Output aggregation of the x_t

e.g.

- **Last iterate**: x_{T-1}
- **Average iteration**: $\frac{1}{T}\sum_{k\in[T-1]} x_k$

How complicated can this be?

- Many possible local steps
- Many ways of measuring progress
- Many ways of using history

(Lots of progress over years and many uses in ML, TCS, OR, etc.)

Typical analysis?

- Bound the number of iterations
- ⇒ bound on oracle complexity (# queries)

This Class

- Motivate new oracle and assumptions on f
- Study structure and design new algorithms
- Prove upper bounds
- Discuss lower bounds
- Repeat ☺

Thursday

Longer illustrative warmup problem.

Plan Questions?

Lecture Plan

Part 1 ✓

<u>Syllabus</u>

- What course is and is not about
- Course setup, expectations, and plans

Part 2 ✓

<u>Course Philosophy / Overview</u>

- Overview of course approach
- Definitions we will use throughout quarter

Part 3

<u>Brief Warmup Problem</u>

- Time permitting

Thursday

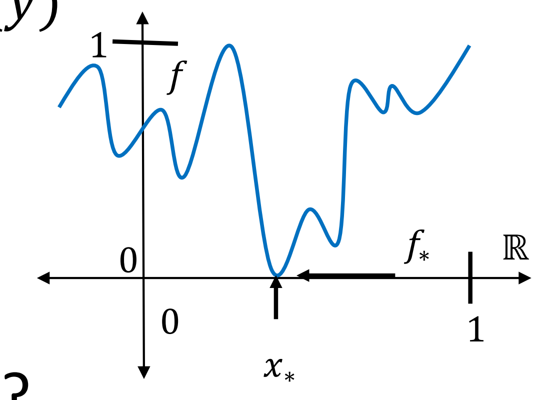
Longer illustrative warmup problem.

Rest of Quarter

Build on foundations set this week.

Setting #1

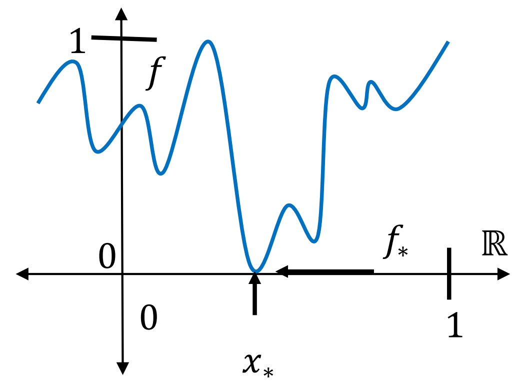
- $f: \mathbb{R} \to \mathbb{R}$ *(one dimensional)*
- Have evaluation oracle *(can compute $f(x)$ with 1 query)*
- Promised $\exists x_* \in [0,1]$ such that $f(x) = f_* = \inf_{y \in \mathbb{R}} f(y)$
- Promised $f(x) \in [0,1]$ for all $x \in \mathbb{R}$
- Goal: compute 1/2-optimal point
  - i.e. compute x with $f(x) \leq f(x_*) + 1/2$



- **Question**: what oracle complexity achievable?
- **Answer**: ∞ is optimal

Proof

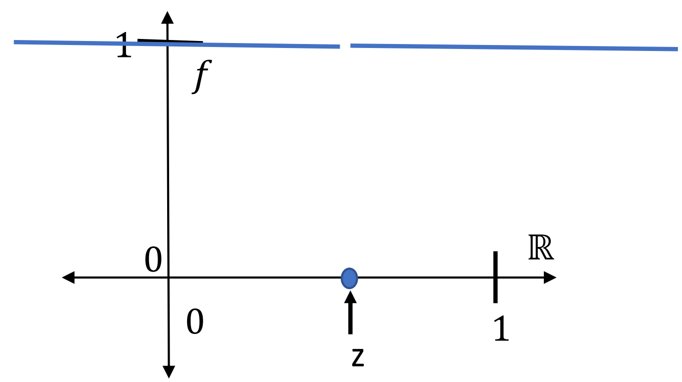
Claim: there is no algorithm and finite number t such that the algorithm always outputs the correct answer in t queries.



- For all $z \in [0,1]$ let

$$f_z(x) = \begin{cases} 1 & x \neq z \\ 0 & x = z \end{cases}$$

Note: the only ½-optimal point is z

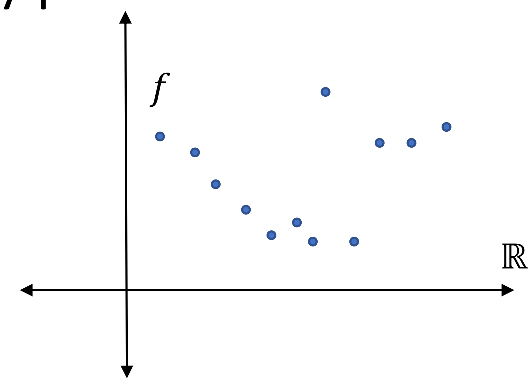


- Suppose oracle always returns 1. This is consistent with $f = f_z$ for all z that is not one of the queried points.

- Since there are two different f_z with disjoint ½-optimal points consistent with oracle, the algorithm will output the incorrect answer when one of these is the input

What went wrong?

Problem: oracle gives only pointwise information, no local information.



Solution:

- This is a class on *continuous* optimization
- Our problems will be continuous or have more structure
- Will see examples next class and the rest of the quarter!

Lecture Plan

See you Thursday!

Part 1 ✔

Syllabus
- What course is and is not about
- Course setup, expectations, and plans

Part 2 ✔

Course Philosophy / Overview
- Overview of course approach
- Definitions we will use throughout quarter

Part 3 ✔

Brief Warmup Problem
- Time permitting

Thursday

Longer illustrative warmup problem.

Rest of Quarter

Build on foundations set this week.