

# Deep learning and continuous optimization

Spring semester 2023/24

---

**Kristóf Bérczi**

MTA-ELTE Matroid Optimization Research Group  
Department of Operations Research, ELTE

# General information

- **First half of the semester:** 5 lectures and practicals
- **Course requirement:** 50% exam (middle of semester, max 50pts), 50% homework (max 50pts)
- **Evaluation:** 0-39 – 1, 40-54 – 2, 55-69 – 3, 70-84 – 4, 85-100 – 5
- **Contact:** `kristof.berczi@ttk.elte.hu`, Room 3.502
- **Reading:**
  - D. Bertsimas, J.N. Tsitsiklis. Introduction to linear optimization.
  - N. Vishnoi. Algorithms for convex optimization.
  - L.C. Lau. Convexity and optimization.
  - S. Bubeck. Convex Optimization: Algorithms and Complexity.
  - S. Boyd, L. Vandenberghe. Convex Optimization.
  - 'I'm a bandit' blog by Sébastien Bubeck.
  - '3Blue1Brown' channel by Grant Sanderson.

# Linear programming

---

# Systems of linear equations

**Example:** A firm produces two different goods using two different raw materials. The available amounts of materials are 12 and 5, respectively. The goods require 2 and 3 units of the first material, and both require 1 unit of the second material. Find a production plan that uses all the raw materials.

**Idea:** Let  $x_1$  and  $x_2$  denote the amounts of the first and second goods produced, respectively. Then the constraints can be written as

$$2 \cdot x_1 + 3 \cdot x_2 = 12$$

$$x_1 + x_2 = 5$$

$$\begin{array}{|c|c|} \hline x_1 & x_2 \\ \hline \end{array} \begin{array}{|c|c|} \hline 2 & 3 \\ \hline 1 & 1 \\ \hline \end{array} = \begin{array}{|c|} \hline 12 \\ \hline 5 \\ \hline \end{array}$$

## Solution

**Step 1.**  $x_1 = 5 - x_2$

**Step 2.**  $10 - 2 \cdot x_2 + 3 \cdot x_2 = 12 \Rightarrow x_2 = 2$

**Step 3.**  $x_1 = 5 - 2 = 3$

# In general

In general: Gauss elimination

$$\begin{array}{|cccc|} \hline x_1 & x_2 & \dots & x_n \\ \hline a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \\ \hline \end{array} = \begin{array}{|c|} \hline b_1 \\ b_2 \\ \vdots \\ b_m \\ \hline \end{array} \Rightarrow \begin{array}{|cccc|} \hline x_1 & x_2 & \dots & x_n \\ \hline 1 & a'_{12} & \dots & a'_{1n} \\ 0 & 1 & \dots & a'_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a'_{mn} \\ \hline \end{array} = \begin{array}{|c|} \hline b'_1 \\ b'_2 \\ \vdots \\ b'_m \\ \hline \end{array}$$

Reduction of the matrix using **elementary row operations**, such as

- swapping two rows,
- multiplying a row by a nonzero number,
- adding a multiple of a row to another row.

**Remarks:**

- The set of solutions does not change.
- A final solution is 'easy' to read out.

# Existence of a solution

Assume that your boss gives you such a problem, that is, solve  $Ax = b$ .

How to prove that a solution exists?

- Just provide a solution  $x$ .

How to prove that there is **no** solution?

- Gauss elimination concludes whether there exists a solution or not.  
**BUT:** this requires the understanding of the algorithm (that you cannot necessarily assume about your boss...)
- Would it be possible to provide some 'shorter' proof?

# Fredholm alternative theorem

## Fredholm alternative theorem

There exists an  $x$  satisfying  $Ax = b$  if and only if there exists no  $y$  such that  $yA = 0$ ,  $yb \neq 0$ .

### Proof of 'only if' direction.

We show that at most one of  $x$  and  $y$  may exist. Suppose to the contrary that  $x$  and  $y$  are such that  $Ax = b$  and  $yA = 0$ ,  $yb \neq 0$ . Then

$$0 = (yA)x = y(Ax) = yb \neq 0,$$

a contradiction. □

**Conclusion:** The non-existence of a solution can be proved by providing  $y$ .

# Geometric interpretation

Naming convention:

**Primal problem**

$$Ax = b \quad (\mathbf{P})$$

**Dual problem**

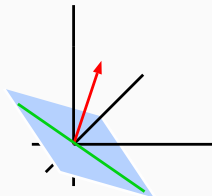
$$\begin{aligned} yA &= 0 \\ yb &\neq 0 \end{aligned} \quad (\mathbf{D})$$

⇒ Fredholm's theorem states that exactly one of **(P)** and **(D)** has a solution.

- The set  $H := \{x : ax = b\}$  is a **hyperplane**.
- $y$  is a **normal vector** of the hyperplane  $H = \{x : ax = b\}$  if  $yx = 0$  for every  $x \in H$ .

## Fredholm as separation theorem

Either  $b$  lies in the subspace generated by the columns of  $A$ , or it can be separated from it by a homogeneous hyperplane with normal vector  $y$ .





# The diet problem

What happens if, instead of equalities, a system of linear inequalities is given?

**Example:** A list of available foods is given together with the nutrient content. Furthermore, the requirement per day of each nutrient is also prescribed. For example, the data corresponding to two types of fruits (F1 and F2) and three types of nutrients (fats, proteins, vitamins) is as follows:

	Fats	Proteins	Vitamins	Available
F1	1	4	5	3
F2	0	2	9	5
Req.	1	5	14	

The problem is to find how much of each fruit to consume per day so as to get the required amount per day of each nutrient, if one can consume at most 2 kg of fruits per day.

## Modeling the problem

	Fats	Proteins	Vitamins	Available
F1	1	4	5	3
F2	0	2	9	5
Req.	1	5	14	

Let  $x_1$  and  $x_2$  denote the amounts of fruits F1 and F2 to be consumed per day.

$$x_1 \geq 1$$

$$4x_1 + 2x_2 \geq 5$$

$$5x_1 + 9x_2 \geq 14$$

$$x_1 + x_2 \leq 2$$

### Questions:

- How to decide feasibility?
- How to find a solution (if exists) algorithmically?
- How to verify that there is no solution?

# Different forms

## Observations:

- An equality  $ax = b$  can be represented as a pair of inequalities  $ax \leq b$  and  $-ax \leq -b$ .
- An inequality  $ax \leq b$  can be represented as the combination of an equality  $ax + s = b$  and a non-negativity constraint  $s \geq 0$ , where  $s$  is called a **slack** variable.
- A non-positivity constraint  $x \leq 0$  can be expressed as a non-negativity constraint  $-x \geq 0$ .
- A variable  $x$  unrestricted in sign can be replaced everywhere by  $x^+ - x^-$ , where  $x^+, x^- \geq 0$ .

### General form

$$Px_0 + Ax_1 = b_0$$

$$Qx_0 + Bx_1 \leq b_1$$

$$x_1 \geq 0$$

### Standard form

$$Ax = b$$

$$x \geq 0$$

### Canonical form

$$Qx \leq b$$

$$x \geq 0$$

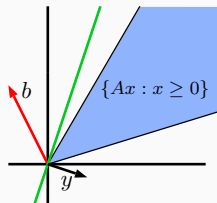
# Farkas' lemma

## Farkas' lemma, standard form

There exists an  $x$  satisfying **(P)**  $Ax = b, x \geq 0$  if and only if there exists no  $y$  such that **(D)**  $yA \geq 0, yb < 0$ .

## Farkas' lemma as separation theorem

Either  $b$  lies in the cone generated by the columns of  $A$ , or it can be separated from it by a homogeneous hyperplane with normal vector  $y$ .



## Proof of the 'only if' direction.

We show that at most one of  $x$  and  $y$  may exist. Suppose to the contrary that  $x$  and  $y$  are such that  $Ax = b, x \geq 0$  and  $yA \geq 0, yb < 0$ . Then

$$0 \leq (yA)x = y(Ax) = yb < 0,$$

a contradiction. □

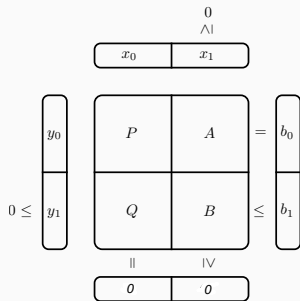
# Farkas' lemma in general

## Farkas' lemma, general form

There exists an  $x = (x_0, x_1)$  satisfying

**(P)**  $Px_0 + Ax_1 = b_0, Qx_0 + Bx_1 \leq b_1, x_1 \geq 0$   
if and only if there exists no  $y = (y_0, y_1)$  such that

**(D)**  $y_0P + y_1Q = 0, y_0A + y_1B \geq 0, y_1 \geq 0, y_0b_0 + y_1b_1 < 0.$



**Conclusion:** The feasibility/infeasibility of a system of linear inequalities can be proved by providing a solution to the primal/dual problem, respectively.

**Remaining question:** How to find such a solution?

$\Rightarrow$  We will answer this in a far more general setting!

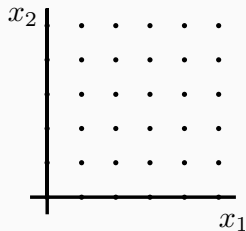
# Geometric background I

## Example

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$



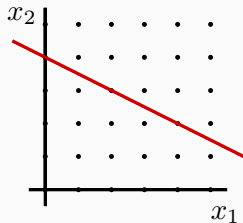
# Geometric background I

## Example

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$



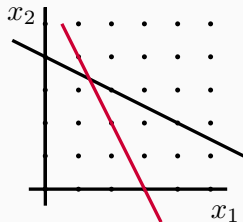
# Geometric background I

## Example

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$





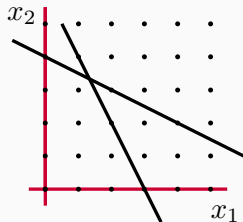
# Geometric background I

## Example

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$



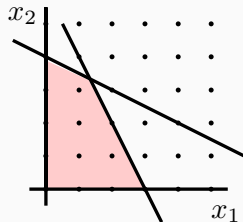
# Geometric background I

## Example

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$



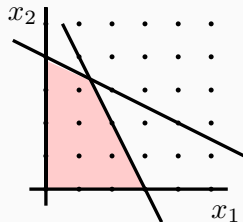
# Geometric background I

## Example

$$x_1 + 2 \cdot x_2 \leq 8$$

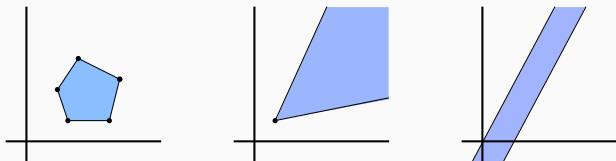
$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$



- An inequality  $ax \leq b$  defines a **half space**.
- The solution set is the **intersection** of a finite number of half spaces, called a **polyhedron**.

## Geometric background II



- Given a polyhedron  $P$ , a point  $x \in P$  is a **vertex** of  $P$  if there exists no  $y$  such that  $x + y, x - y \in P$ .
- A **polytope** is the convex hull of a finite number of points.

### Thm.

Every polytope is a polyhedron, and every bounded polyhedron is the convex hull of its vertices.

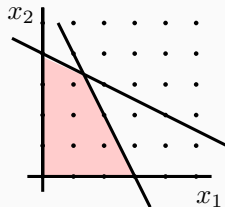
# Geometric background III

## Example

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$



**Goal:** Maximize/minimize a linear objective function over the set of solutions.

⇒ **Example:**  $\max\{x_1 + x_2\}$ .

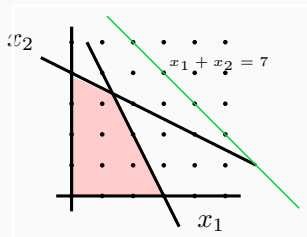
# Geometric background III

## Example

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$



**Goal:** Maximize/minimize a linear objective function over the set of solutions.

⇒ **Example:**  $\max\{x_1 + x_2\}$ .

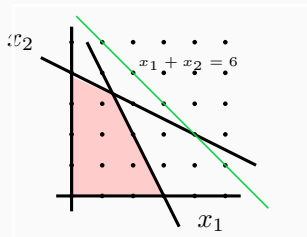
# Geometric background III

## Example

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$



**Goal:** Maximize/minimize a linear objective function over the set of solutions.

⇒ **Example:**  $\max\{x_1 + x_2\}$ .

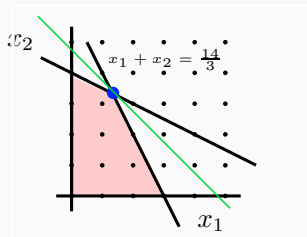
# Geometric background III

## Example

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$



**Goal:** Maximize/minimize a linear objective function over the set of solutions.

⇒ **Example:**  $\max\{x_1 + x_2\}$ .



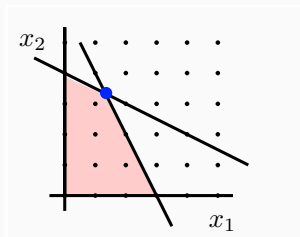
# Geometric background III

## Example

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$



**Goal:** Maximize/minimize a linear objective function over the set of solutions.

⇒ **Example:**  $\max\{x_1 + x_2\}$ .

**Idea:** Start from a vertex, and move to a neighboring vertex with improved objective value.

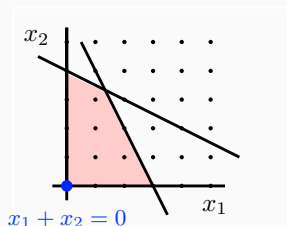
# Geometric background III

## Example

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$



**Goal:** Maximize/minimize a linear objective function over the set of solutions.

⇒ **Example:**  $\max\{x_1 + x_2\}$ .

**Idea:** Start from a vertex, and move to a neighboring vertex with improved objective value.

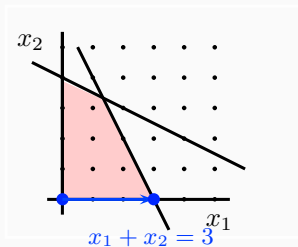
# Geometric background III

## Example

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$



**Goal:** Maximize/minimize a linear objective function over the set of solutions.

⇒ **Example:**  $\max\{x_1 + x_2\}$ .

**Idea:** Start from a vertex, and move to a neighboring vertex with improved objective value.

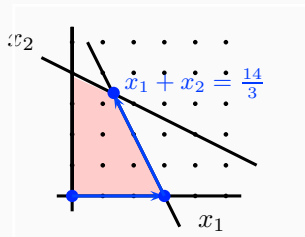
# Geometric background III

## Example

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$



**Goal:** Maximize/minimize a linear objective function over the set of solutions.

⇒ **Example:**  $\max\{x_1 + x_2\}$ .

**Idea:** Start from a vertex, and move to a neighboring vertex with improved objective value.

# History

1827, Fourier: Fourier-Motzkin elimination

1939, Kantorovich: reducing costs of army, general LP

1940's, Koopmans: economic problems as LPs

1941, Hitchcock: transportation problems as LPs

1946-47, Dantzig: general LP for planning problems in US Air Force (**simplex method**)

1979, Khachiyan: ellipsoid method, LP is solvable in linear time (more theoretical than practical)

1984, Karmakar: interior-point method (can be used in practice)

# Linear programs

We would like to solve problems of the form

## General form

$$\begin{aligned} \max \quad & c_0x_0 + c_1x_1 \\ \text{s.t.} \quad & Px_0 + Ax_1 = b_0 \\ & Qx_0 + Bx_1 \leq b_1 \\ & x_1 \geq 0 \end{aligned}$$

## Standard form

$$\begin{aligned} \max \quad & cx \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

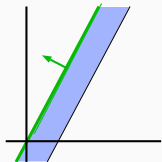
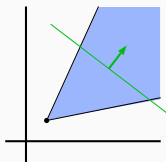
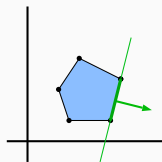
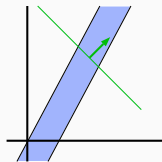
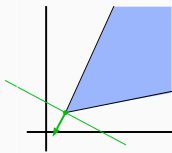
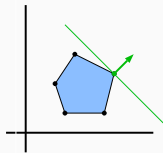
## Canonical form

$$\begin{aligned} \max \quad & cx \\ \text{s.t.} \quad & Qx \leq b \\ & x \geq 0 \end{aligned}$$

## Remarks:

- A minimization problem  $\min cx$  can be reformulated as a maximization problem  $\max (-c)x$  and vice versa.
- The optimal solution can be obtained by 'moving' a hyperplane with normal vector  $c$  towards the polyhedron, and finding the first point where they meet [**Be careful:** min or max?]  
⇒ **Intuition:** the optimum is always attained at a vertex.

## Geometric background IV



### Possible cases:

- single optimal solution,
- infinite number of optimal solutions, or
- no optimal solution (unbounded objective value).

# Geometric background V

## Thm.

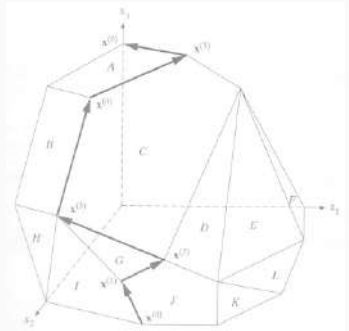
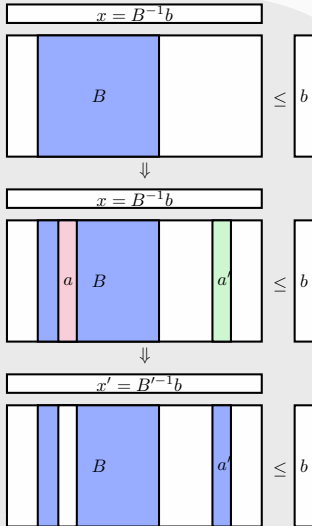
Let  $P = \{x : Qx \leq b\}$  where the columns of  $Q$  are linearly independent. Then  $x \in P$  is a vertex if and only if it can be obtained by taking a non-singular  $r(Q) \times r(Q)$  submatrix  $Q'$  of  $Q$  and the corresponding part  $b'$  of  $b$ , and solving the system  $Q'x = b'$ .

## Remarks:

- The number of such submatrices, and so the number of vertices is finite.  
     $\Rightarrow$  If each vertex is visited at most once, then the procedure terminates.
- When the columns are non-independent, then there is an infinite number of basic feasible solutions. However, there are only a finite number of so-called **strong basic feasible solutions**, and, if it exists, the optimum is attained in one of them.



# Simplex method



## Problems

- Running time?
- Optimal solution?

# Running time

**Problem 1:** The simplex algorithm might fail to terminate.

**Reason:** The algorithm can fall into cycles between bases associated with the same basic feasible solution and objective value.

**Solution:** Careful pivoting rule, e.g. **Bland's rule** prevents cycling.

**Problem 2:** Efficient in practice, but for almost every variant, there is a family of linear programs for which it performs **badly**.

**Reason:** The number of vertices of a polyhedron can be exponentially large.

**Solution:** Sub-exponential pivot rules are known.

**Major open problem:** Is there a variant with polynomial running time?

- **Hirsch's conjecture:** Let  $P$  be a  $d$ -dimensional convex polytope with  $n$  facets. Then the diameter of  $P$  is at most  $n - d$ .
- Counterexample by Francisco Santos, 2011 (86 facets, 43-dimensional).

# Duality theorem

**Problem 3:** Is the solution optimal?

## Duality theorem

Consider the problems

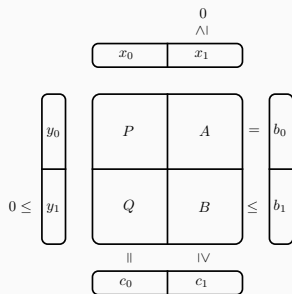
$$\text{(P)} \quad \max(c_0x_0 + c_1x_1) \quad \text{s.t.} \quad Px_0 + Ax_1 = b_0, Qx_0 + Bx_1 \leq b_1, x_1 \geq 0$$

and

$$\text{(D)} \quad \min(y_0b_0 + y_1b_1) \quad \text{s.t.} \quad y_0P + y_1Q = c_0, y_0A + y_1B \geq c_1, y_1 \geq 0.$$

Then exactly one of the followings hold:


- 1 both (P) and (D) are empty,
- 2 (D) is empty and (P) is unbounded,
- 3 (P) is empty and (D) is unbounded,
- 4 both (P) and (D) have a solution, and  $\max = \min$ .



# Reading assignment

 D. Bertsimas, J.N. Tsitsiklis. Introduction to linear optimization.

- Chapter 1, Sections 1.1, 1.2, 1.4, and 1.5
- Chapter 2, Sections 2.1 and 2.2
- Chapter 4, Sections 4.1-4.3, 4.6

 A. Frank, T. Király. Operációkutatás (in Hungarian).

- Chapter 2
- Chapter 3
- Chapter 4
- Chapter 6

# Exercises

- 1 Consider the problem  $x_2 \leq 4, x_1 + x_2 \leq 6, 2x_1 + x_2 \leq 10, x_1, x_2 \geq 0$ . Represent these constraints on the plane. Find a point that maximizes  $x_1 + 2x_2$ . (3pts)
- 2 Let  $A \in \mathbb{R}^{m \times n}$ ,  $b \in \mathbb{R}^m$ , and  $c_1, \dots, c_k \in \mathbb{R}^n$ . Formulate the following problem as an LP:  $Ax = b, x \geq 0, \min f(x)$ , where  $f(x) := \max\{c_1x, \dots, c_kx\}$ . (2pts)
- 3 Reduce the following systems of inequalities to each other (in the sense that if we can solve one of them, then we can solve any of them):

$$\begin{aligned} Ax &= b \\ x &\geq 0 \end{aligned}$$

$$\begin{aligned} Bx &\leq b \\ x &\geq 0 \end{aligned}$$

$$Qx \leq b$$

$$\begin{aligned} Px_0 &= b_0 \\ Qx &\leq b_1 \end{aligned}$$

(4pts)

- 4 Is it true, that a set  $K \subseteq \mathbb{R}^n$  is convex if and only if for any  $x, y \in K$  we have  $(x + y)/2 \in K$ ? (1pt)

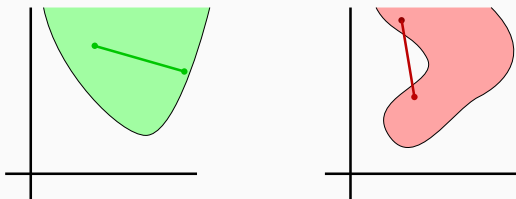
# Convexity

---

# Convex sets

A set  $K \subseteq \mathbb{R}^n$  is **convex** if for all  $x, y \in K$  and  $\theta \in [0, 1]$ , we have

$$\theta x + (1 - \theta)y \in K.$$



## Examples:

- **Polytopes:**  $K = \{x \in \mathbb{R}^n : \langle a_i, x \rangle \leq b_i \text{ for } i = 1, \dots, m\}$ , where  $a_i \in \mathbb{R}^n$  and  $b_i \in \mathbb{R}$  for  $i = 1, \dots, m$ .
- **Ellipsoids:**  $K = \{x \in \mathbb{R}^n : x^T A x \leq 1 \text{ where } A \in \mathbb{R}^{n \times n} \text{ is a positive definite matrix.}$
- **Balls (in  $\ell_p$  norms for  $p \geq 1$ ):**  $K = \{x \in \mathbb{R}^n : \sqrt[p]{\sum_{i=1}^n |x_i - a_i|^p} \leq 1\}$ , where  $a \in \mathbb{R}^n$  is a vector.

# Convex functions

A function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is **convex** if its domain is a convex set and for all  $x, y \in K$  and  $\theta \in [0, 1]$ , we have

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y).$$

If the inequality always holds as strict inequality, the function is **strictly convex**.

The function  $f$  is **concave** or **strictly concave** if  $-f$  is convex or strictly convex, respectively.

**Remark:** If  $f : K \rightarrow \mathbb{R}^n$  is a convex function, then setting  $f(x) = +\infty$  for  $x \notin K$  results in a convex function when the arithmetic operations on  $\mathbb{R} \cup \{+\infty\}$  are interpreted in the reasonable way.



# Semidefinite matrices

A matrix  $M \in \mathbb{R}^{n \times n}$  is **symmetric** if  $M^T = M$ .

The **identity matrix** of size  $n \times n$  is denoted by  $I_n$ .

A symmetric matrix  $M$  is **positive semidefinite (PSD)** if  $x^T M x \geq 0$  holds for all  $x \in \mathbb{R}^n$ , and this is denoted by  $M \succeq 0$ .

$M$  is **positive definite (PD)** if  $x^T M x > 0$  holds for all non-zero  $x \in \mathbb{R}^n$ , and this is denoted by  $M \succ 0$ .

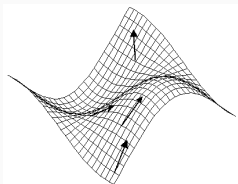
We define  $M \succeq N \Leftrightarrow M - N \succeq 0$  and  $M \succ N \Leftrightarrow M - N \succ 0$ .

# Calculus I

We are working with 'sufficiently smooth' functions  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ .

The derivative of  $f(x_1, \dots, x_n)$  is called the **gradient**, and is defined as

$$\nabla f(x) = \left[ \frac{\partial f}{\partial x_1}(x), \dots, \frac{\partial f}{\partial x_n}(x) \right]$$



The **directional derivative** of  $f$  in the direction  $d$  is  $\langle \nabla f(x), d \rangle$ .

The second derivatives of  $f$  can be summarized in the **Hessian** matrix

$$\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \cdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

**Remark:** The Hessian is symmetric if  $f$  is sufficiently differentiable.

# Calculus II

## Taylor expansion

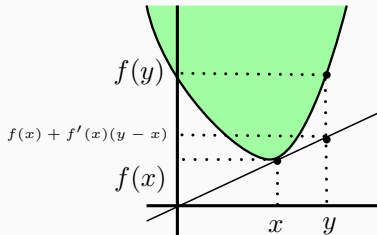
The Taylor series expansion of  $f$  around  $x = a$  is

$$f(x) = \underbrace{f(a) + \langle \nabla f(a), x - a \rangle}_{\text{first order approximation}} + \underbrace{\frac{1}{2}(x - a)^T \nabla^2 f(a)(x - a) + \dots}_{\text{second order approximation}}$$

Consider a function in one dimension, i.e.  $f : \mathbb{R} \rightarrow \mathbb{R}$ .

When  $f$  is convex, the tangent is 'below' the graph, i.e.

$$f(y) \geq f(x) + f'(x)(y - x).$$



# First order condition

## First order condition

Let  $f$  be a differentiable function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  over a convex set  $K$ . Then  $f$  is convex if and only if for all  $x, y \in K$

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle.$$

## Proof of the one-dimensional case.

$\Rightarrow$  For any  $\theta \in [0, 1]$ , we have

$$(1 - \theta)f(x) + \theta f(y) \geq f(\theta y + (1 - \theta)x) = f(x + \theta(y - x)).$$

Subtracting  $(1 - \theta)f(x)$  and dividing by  $\theta$  yields

$$f(y) \geq f(x) + \frac{f(x + \theta(y - x)) - f(x)}{\theta}.$$

Taking limit  $\theta \rightarrow 0$  gives  $f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle$ . □

# First order condition

## First order condition

Let  $f$  be a differentiable function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  over a convex set  $K$ . Then  $f$  is convex if and only if for all  $x, y \in K$

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle.$$

## Proof of the one-dimensional case.

$\Leftarrow$  Let  $z = \theta x + (1 - \theta)y$ . The first order approximation underestimates both  $f(x)$  and  $f(y)$ , hence

$$f(x) \geq f(z) + \nabla f(z)^T(x - z),$$

$$f(y) \geq f(z) + \nabla f(z)^T(y - z).$$

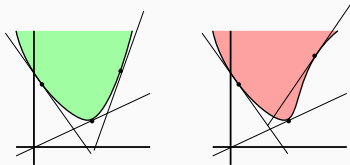
Therefore

$$(1 - \theta)f(x) + \theta f(y) \geq f(z) + \nabla f(z)^T(\theta x + (1 - \theta)y - z) = f(\theta(y) + (1 - \theta)x).$$

□

## Second order condition

In the one-dimensional case,  $f''(x) \geq 0$  when  $f$  is convex, that is, the slope of the tangent is non-decreasing, as otherwise when the slope decreases the function becomes non-convex.



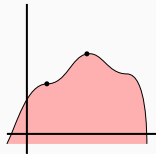
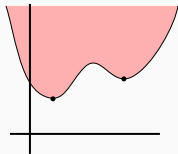
### Second order condition

Let  $f$  be twice differentiable such that  $\text{dom } f$  is open. Then  $f$  is convex if and only if  $\nabla^2 f(x) \succeq 0$  for all  $x \in \text{dom } f$ .

# Local vs. global optimum I

Convex optimization problem:

$$\inf_{x \in K} f(x) \xrightarrow{\text{usually}} \min_{x \in K} f(x)$$



**Intuition:**  $\nabla f(x) = 0$  when  $x$  is optimal.

**Problem:**  $\nabla f(x) = 0$  may correspond to a local optimum/maximum.

## Global optimum

If the domain of a convex differentiable function  $f$  is  $\mathbb{R}^n$ , then  $x$  is an optimal solution to  $\inf_{x \in \mathbb{R}^n} f(x)$  if and only if  $\nabla f(x) = 0$ .

## Local vs. global optimum II

### Proof of the 'if' direction.

Assume that  $\nabla f(x_0) = 0$ . Since  $f$  is convex, we know that for all  $y \in \mathbb{R}^n$  we have

$$\begin{aligned} f(y) &\geq f(x_0) + \langle \nabla f(x_0), y - x_0 \rangle \\ &= f(x_0) + \langle 0, y - x_0 \rangle \\ &= f(x_0). \end{aligned}$$

□

**Remark:** In the constrained setting, i.e. when  $K \neq \mathbb{R}^n$ , the following holds.

### Global optimum

If  $f$  is a convex differentiable function, then  $x$  is an optimal solution to  $\inf_{x \in \mathbb{R}^n} f(x)$  if and only if  $\langle \nabla f(x), y - x \rangle \geq 0$  for all  $y \in \mathbb{R}^n$ .



# Convex programs

A **convex program** can be written as follows.

## Convex program

$$\inf f_0(x)$$

$$\text{s.t. } f_i(x) \leq 0 \text{ for } 1 \leq i \leq m$$

$$h_j(x) = 0 \text{ for } 1 \leq j \leq p$$

- $f_i$  is convex for  $i = 0, \dots, m$
- $h_j$  is convex for  $j = 1, \dots, p$

**Remark:** The domain of the problem is  $D := \left(\bigcap_{i=0}^m \text{dom } f_i\right) \cap \left(\bigcap_{j=1}^p \text{dom } h_j\right)$ , which is a convex set  $\Rightarrow$  Roughly speaking, this makes the problem tractable.

**Question:** Can we define a dual program? How to give a lower bound?

# Dual programs I

**Idea:** “move the constraints into the objective function”

The **Lagrangian** associated with the problem is

$$L(x, \lambda, \mu) := f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{j=1}^p \mu_j h_j(x),$$

where the  $\lambda_i$ s and  $\mu_j$ s are called **Lagrangian multipliers**, and  $\lambda \in \mathbb{R}^m$  and  $\mu \in \mathbb{R}^p$  are called the **dual variables**.

The **Lagrangian dual function** is the min value of the Lagrangian over  $x$ ,

$$g(\lambda, \mu) := \inf_x L(x, \lambda, \mu).$$

## Dual programs II

Let  $OPT_P$  denote the optimum value of the primal problem, and let  $\hat{x}$  be an arbitrary feasible solution. Furthermore, assume that  $\lambda \geq 0$ . Then

$$\begin{aligned}g(\lambda, \mu) &\leq f_0(\hat{x}) + \sum_{i=1}^m \lambda_i f_i(\hat{x}) + \sum_{j=1}^p \mu_j h_j(\hat{x}) \\ &\leq f_0(\hat{x}),\end{aligned}$$

hence  $g(\lambda, \mu) \leq \inf_{x \text{ feasible}} f_0(x) = OPT_P$ .

### Conclusion:

- This gives a lower bound when  $\lambda \geq 0$  and  $g(\lambda, \mu) > -\infty \Rightarrow$  Such a pair  $\lambda, \mu$  is called **dual feasible**.

# Weak duality

The goal is to get the best lower bound on  $OPT_P$  using the Lagrangian dual.

The **dual program** is thus defined as

**Dual program**

$$\begin{aligned} \max \quad & g(\lambda, \mu) \\ \text{s.t.} \quad & \lambda \geq 0 \end{aligned}$$

Let  $OPT_D$  denote the optimal value of the dual. Then **weak duality** holds by construction, that is,  $OPT_D \leq OPT_P$ .

**Remarks:**

- The dual program is always convex, regardless of the primal.
- That is, for any primal program (even though non-convex), we can always write a convex program that gives a lower bound on the primal objective value.

# Strong duality

**Question:** Does  $OPT_D = OPT_P$  always holds?

**Answer:** Unfortunately **NOT**. But!

The **Slater's condition** requires that there is  $x \in \text{rel int}(D)$  such that  $f_i(x) < 0$  for  $1 \leq i \leq m$  and  $h_j(x) = 0$  for  $1 \leq j \leq p$ .

(That is, there exists an **interior** point in the domain, which is a feasible solution, and satisfies the non-affine inequality constraints **strictly**.)

## Strong duality

If Slater's condition holds, then  $OPT_D = OPT_P$ .

## Complementary slackness

Assume that  $OPT_D = OPT_P$ . Let  $x^*$  be a primal,  $\lambda^*, \mu^*$  be dual optimal solutions. Then

$$\begin{aligned}f_0(x^*) &= g(\lambda^*, \mu^*) \\&= \inf_x L(x, \lambda^*, \mu^*) \\&\leq L(x^*, \lambda^*, \mu^*) \\&= f_0(x^*) + \sum_{i=1}^m \lambda_i^* f_i(x^*) + \sum_{j=1}^p \mu_j^* h_j(x^*) \\&\leq f_0(x^*),\end{aligned}$$

as  $\lambda \geq 0$  (dual feasible) and  $f_i(x^*) \leq 0$ ,  $h_j(x^*) = 0$  (primal feasible).

Therefore

- $x^*$  is a minimizer of  $L(x, \lambda^*, \mu^*)$ , and
- $\lambda_i^* f_i(x^*) = 0$  for  $1 \leq i \leq m$ , called the **complementary slackness condition**, meaning that the non-zero pattern of  $\lambda_i^*$  and  $f_i(x^*)$  must be complementary.

# Karush-Kuhn-Tucker (KKT) conditions I

Assume that  $f_0, f_1, \dots, f_m, h_1, \dots, h_p$  are all differentiable.

Since  $x^*$  minimize  $L(x, \lambda^*, \mu^*)$  by the above, the gradient of  $L$  at  $x^*$  must be zero, that is,

$$\nabla f_0(x^*) + \sum_{i=1}^m \lambda_i^* \nabla f_i(x^*) + \sum_{j=1}^p \mu_j^* \nabla h_j(x^*) = 0.$$

To sum up, the following are some necessary conditions for any pair of primal and dual optimal solutions.

**Primal feasibility:**  $f_i(x^*) \leq 0$  for  $1 \leq i \leq m$ ,  $h_j(x^*) = 0$  for  $1 \leq j \leq p$ .

**Dual feasibility:**  $\lambda_i^* \geq 0$  for  $1 \leq i \leq m$ .

**Complementary slackness:**  $\lambda_i^* f_i(x^*) = 0$  for  $1 \leq i \leq m$ .

**Lagrangian optimality:**  $\nabla f_0(x^*) + \sum_{i=1}^m \lambda_i^* \nabla f_i(x^*) + \sum_{j=1}^p \mu_j^* \nabla h_j(x^*) = 0$ .

This set of conditions is called the **KKT conditions**.

## Karush-Kuhn-Tucker (KKT) conditions II

When the primal problem is convex, the KKT conditions are also **sufficient!**

⇒ Any  $x^*, \lambda^*, \mu^*$  satisfying KKT must be primal and dual optimal solutions.

**Reason:** If the primal is convex, then  $L(x, \lambda, \mu)$  is convex in  $x$  when  $\lambda, \mu$  are fixed. Hence a local optimal solution is also a global optimal solution. More precisely:

$$\begin{aligned}g(\lambda^*, \mu^*) &= \inf_x L(x, \lambda^*, \mu^*) \\&= L(x^*, \lambda^*, \mu^*) \\&= f_0(x^*) + \sum_{i=1}^m \lambda_i^* f_i(x^*) + \sum_{j=1}^p \mu_j^* h_j(x^*) \\&= f_0(x^*).\end{aligned}$$

**Summary:** For a convex problem with differentiable functions, if Slater's condition is satisfied, then the KKT conditions are **necessary** and **sufficient** for optimality.



# Reading assignment



N. Vishnoi. Algorithms for convex optimization.

- Chapter 1
- Chapter 2
- Chapter 3
- Chapter 4
- Chapter 5



L.C. Lau. Convexity and optimization.

- Lecture 1
- Lecture 2
- Lecture 3
- Lectures 4-5

# Exercises

① Verify the following statements. (5pts)

- a  $e^{ax}$  is convex on  $\mathbb{R}$  for any  $a \in \mathbb{R}$ .
- b  $x^a$  is convex on  $\mathbb{R}_{>0}$  when  $a \geq 1$  or  $a \leq 0$ , otherwise it is concave.
- c  $\log x$  is concave on  $\mathbb{R}_{>0}$ .
- d  $x \log x$  is convex on  $\mathbb{R}_{>0}$ .
- e  $\max\{x_1, \dots, x_n\}$  is convex on  $\mathbb{R}^n$ .

② Consider the optimization problem

$$\begin{aligned} \min \quad & x^2 + 2x + 1 \\ \text{s.t.} \quad & (x + 2)(x - 4) \leq 0 \end{aligned}$$

with variable  $x \in \mathbb{R}$ .

- a Give the feasible set, the optimal value, and the optimal solution. (1pt)
- b Plot the objective  $x^2 + 2x + 1$  versus  $x$ . On the same plot, show the feasible set, optimal point and value, and plot the Lagrangian  $L(x, \lambda)$  versus  $x$  for a few positive values of  $\lambda$ . Derive and sketch the Lagrange dual function  $g$ . (2pts)
- c State the dual problem, and verify that it is a concave maximization problem. Find the dual optimal value and dual optimal solution  $\lambda^*$ . (2pts)

# Gradient descent, Mirror descent, and Multiplicative Weights Update

---

# Setting

**Objective:**  $\min_{x \in \mathbb{R}^n} f(x)$  (**unconstrained setting**)

**Model:** 1st-order oracle is given, i.e., we can query the gradient at any point.

**Solution:** Given  $\varepsilon > 0$ , output a point  $x \in \mathbb{R}^n$  s.t.  $f(x) \leq y^* + \varepsilon$ , where  $y^*$  denotes the optimal value.

- The running time will be proportional to  $1/\varepsilon$ , hence it is not polynomial. However, we will see that in this setting one cannot obtain polynomial time algorithms.

**Remark:** As  $f$  is convex, a local minimum is a global minimum. So as long as we can find a point to decrease the objective value, we are making progress and we won't get stuck. But how to decrease the objective?

# Gradient descent

Not a single method, but a general framework.

## Scheme:

- ① Choose a starting point  $x_1 \in \mathbb{R}^n$ .
- ② Suppose  $x_1, \dots, x_t$  are computed. Choose  $x_{t+1}$  as a linear combination of  $x_t$  and  $\nabla f(x_t)$ .
- ③ Stop once a certain stopping criterion is met and output the last iterate.

If  $T$  is the total number of iterations, then the running time is  $O(T \cdot M(x))$ , where  $M(x)$  is the time of each update.

- The update time  $M(x)$  cannot be optimized below a certain level.
- The main goal is to **keep  $T$  as small as possible**.

# Why using the gradient? I

We only have local information about  $x \Rightarrow$  a reasonable idea is to pick a direction which locally provides the **largest drop** in the function value.

**Formally:** Pick a unit vector  $u$  for which a 'tiny' ( $\delta$ ) step in direction  $u$  maximizes

$$f(x) - f(x + \delta u).$$

This leads to the optimization problem

$$\max_{\|u\|=1} \left[ \lim_{\delta \rightarrow 0^+} \frac{f(x) - f(x + \delta u)}{\delta} \right].$$

By the Taylor approximation of  $f$ , the limit is simply the directional derivative of  $f$  at  $x$  in direction  $u$ , thus

$$\max_{\|u\|=1} [-\langle \nabla f(x), u \rangle].$$

# Cauchy-Schwarz inequality

## Cauchy-Schwarz inequality

For all  $x, y \in \mathbb{R}^n$ , we have  $\langle x, y \rangle \leq \|x\| \|y\|$ .

### Proof sketch.

Assuming  $x, y \in \mathbb{R}^2$ , we know that  $\langle x, y \rangle = \|x\| \|y\| \cos \theta$ , where  $\theta$  is the angle between  $x$  and  $y$ . In higher dimensions, intuitively, the two vectors  $x$  and  $y$  form together a subspace of dimension at most 2 that can be thought of as  $\mathbb{R}^2$ .  $\square$

# Why using the gradient? II

**Recall:**  $\max_{\|u\|=1} [-\langle \nabla f(x), u \rangle]$

From the Cauchy-Schwarz inequality, we get

$$-\langle \nabla f(x), u \rangle \leq \|\nabla f(x)\| \|u\| = \|\nabla f(x)\|,$$

and equality holds if  $u = -\frac{\nabla f(x)}{\|\nabla f(x)\|}$ .

⇒ Moving in the direction of the **negative gradient** is an instantaneously good strategy - called the **gradient flow**:

$$\frac{dx}{dt} = -\frac{\nabla f(x)}{\|\nabla f(x)\|}.$$

**Question:** How to implement the strategy on a computer?

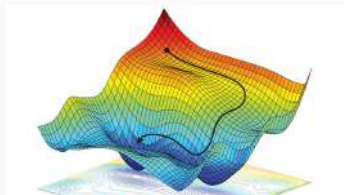
**Natural discretization:**

$$x_{t+1} = x_t - \alpha \frac{\nabla f(x_t)}{\|\nabla f(x_t)\|},$$

where  $\alpha > 0$  is the 'step length'. More generally,

$$x_{t+1} = x_t - \eta \nabla f(x_t),$$

where  $\eta > 0$  is a parameter.





# Assumptions

**Step length:** Ideally, we would like to take **big steps**. This results in **smaller number of iterations**, but **the function can change dramatically, leading to a large error**.

**Solution:** Assumptions on certain **regularity parameters**.

① **Lipschitz gradient.** For every  $x, y \in \mathbb{R}^n$  we have

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|.$$

This is also sometimes referred to as  **$L$ -smoothness** of  $f$ .

$\Rightarrow$  Around  $x$ , the gradient changes in a controlled manner; we can take larger step size.

② **Bounded gradient.** For every  $x \in \mathbb{R}^n$  we have

$$\|\nabla f(x)\| \leq G.$$

This implies that  $f$  is  $G$ -Lipschitz.

$\Rightarrow$  The function can go towards infinity in a controlled manner.

③ **Good initial point.** A point  $x_1$  is provided such that  $\|x_1 - x^*\| \leq D$ , where  $x^*$  is some optimal solution.

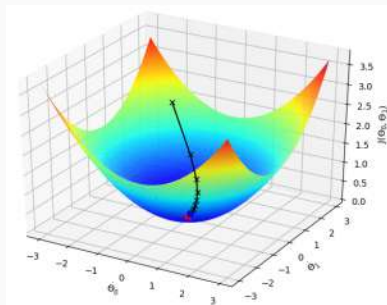
# Lipschitz gradient

## Thm.

Given a first-order oracle access to an  $L$ -Lipschitz convex function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , an initial point  $x_1 \in \mathbb{R}^n$  with  $\|x_1 - x^*\| \leq D$ , and  $\varepsilon > 0$ , there is an algorithm that outputs a point  $x \in \mathbb{R}^n$  such that  $f(x) \leq f(x^*) + \varepsilon$ . The algorithm makes  $T = O\left(\frac{LD^2}{\varepsilon}\right)$  queries to the oracle and performs  $O(nT)$  arithmetic operations.

## Algorithm

- 1 Let  $T = O\left(\frac{LD^2}{\varepsilon}\right)$ .
- 2 Let  $\eta = \frac{1}{L}$ .
- 3 Repeat for  $t = 1, \dots, T - 1$ :
  - $x_{t+1} = x_t - \eta \nabla f(x_t)$ .
- 4 **Output**  $x_T$ .



# Lipschitz gradient

## Lower bound

Consider any algorithm for solving the convex unconstrained minimization problem  $\min_{x \in \mathbb{R}^n} f(x)$  in the first-order model, when  $f$  has Lipschitz gradient with constant  $L$  and the initial point  $x_1 \in \mathbb{R}^n$  satisfies  $\|x_1 - x^*\| \leq D$ . There is a function  $f$  such that

$$\min_{1 \leq i \leq T} f(x_i) - \min_{x \in \mathbb{R}^n} f(x) \geq \frac{LD^2}{T^2}.$$

⇒ The theorem translates to a lower bound of  $\Omega\left(\frac{1}{\sqrt{\varepsilon}}\right)$  iterations to reach an  $\varepsilon$ -optimal solution.

*Is there a method which matches the  $\frac{1}{\sqrt{\varepsilon}}$  iterations bound? Yes!*

## Nesterov's accelerated gradient descent algorithm

Under the same assumptions, there is an algorithm that outputs a point  $x \in \mathbb{R}^n$  such that  $f(x) \leq f(x^*) + \varepsilon$ , makes  $T = O\left(\frac{\sqrt{LD}}{\sqrt{\varepsilon}}\right)$  queries to the oracle, and performs  $O(nT)$  arithmetic operations.

# Constrained setting - projection

**Objective:**  $\min_{x \in K} f(x)$  (**constrained setting**)

$\Rightarrow$  The next iterate  $x_{t+1}$  might fall outside of  $K$ , hence we need to project it back onto  $K$ , that is,

$$x_{t+1} = \text{proj}_K(x_t - \eta_t \cdot \nabla f(x_t)).$$

**Difficulty:** The projection may or may not be computationally expensive to perform.

## Thm.

Given a first-order oracle access to a convex function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  with an  $L$ -Lipschitz gradient, oracle access to a projection operator  $\text{proj}_K$  onto a convex set  $K \subseteq \mathbb{R}^n$ , an initial point  $x_1 \in \mathbb{R}^n$  with  $\|x_1 - x^*\| \leq D$ , and  $\varepsilon > 0$ , there is an algorithm that outputs a point  $x \in \mathbb{R}^n$  such that  $f(x) \leq f(x^*) + \varepsilon$ . The algorithm makes  $T = O\left(\frac{LD^2}{\varepsilon}\right)$  queries to the first-order and the projection oracles and performs  $O(nT)$  arithmetic operations.

# Regularizers I

The Lipschitz gradient algorithm leaves out convex functions which are **non-differentiable**, such as  $f(x) = \sum_{i=1}^n |x_i|$  or  $f(x) = \max\{|x_1|, \dots, |x_n|\}$ .

*Let's reconsider how to choose the next point to converge quickly?*

**Obvious choice:**  $x^{t+1} = \arg \min_{x \in K} f(x)$

$\Rightarrow$  Converges quickly to  $x^*$  (in one step). Yet, it is not very helpful as  $x^{t+1}$  is **hard to compute**.

**Idea:** Construct a function  $f^t$  that **approximates**  $f$  in a certain sense and is **easy to minimize**. The update rule becomes

$$x^{t+1} = \arg \min_{x \in K} f^t(x).$$

$\Rightarrow$  Intuitively, if  $f^t$  becomes more and more accurate, the sequence of iterates should converge to  $x^*$ .

## Regularizers II

### Example

The Lipschitz gradient algorithm corresponds to the choice

$$f^t(x) = f(x^t) + \langle \nabla f(x^t), x - x^t \rangle + \frac{L}{2} \|x - x^t\|^2.$$

Indeed,  $\nabla f^t(x) = \nabla f(x^t) + L(x - x^t) = 0$  if and only if  $x = x^t - \frac{1}{L} \nabla f(x^t)$ .

In general, when the function is not differentiable, one can try to use the first order approximation of  $f$  at  $x^t$ , that is,

$$f^t(x) = f(x^t) + \langle \nabla f(x^t), x - x^t \rangle.$$

Then  $f^t(x) \leq f(x)$  and  $f^t$  gives a descent approximation of  $f$  in a small neighborhood  $x^t$ . The resulting updating rule will be

$$x^{t+1} = \arg \min_{x \in K} \{f(x^t) + \langle \nabla f(x^t), x - x^t \rangle\}.$$

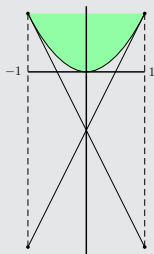
# Regularizers III

## Example

$K = [-1, 1]$  and  $f(x) = x^2$

⇒ The algorithm is way too aggressive as it jumps between  $-1$  and  $+1$  indefinitely.

[**Even worse:** if  $K$  is unbounded, then the minimum is not attained at any finite point!]



**Idea:** Add a term involving a distance function  $D : K \times K \rightarrow \mathbb{R}$  that does not allow  $x^{t+1}$  to land far away from  $x^t$ . More precisely,

$$\begin{aligned}x^{t+1} &= \arg \min_{x \in K} \{D(x, x^t) + \eta(f(x^t) + \langle \nabla f(x^t), x - x^t \rangle)\} \\ &= \arg \min_{x \in K} \{D(x, x^t) + \eta \langle \nabla f(x^t), x \rangle\}.\end{aligned}$$

**Remark:** By picking large  $\eta$ , the significance of the regularizer is reduced. By picking small  $\eta$ , we force  $x^{t+1}$  to stay close to  $x^t$ .

# Kullback-Leibler divergence

**Objective:**  $\min_{p \in \Delta_n} f(p)$ , where  $\Delta_n = \{p \in [0, 1]^n : \sum_{i=1}^n p_i = 1\}$  is the **probability simplex**.

Recall that

$$p^{t+1} = \arg \min_{p \in \Delta_n} \{D(p, p^t) + \eta \langle \nabla f(p^t), p \rangle\}.$$

For two probability distributions  $p, q \in \Delta_n$ , their **Kullback-Leibler divergence** is defined as

$$D_{KL}(p, q) = - \sum_{i=1}^n p_i \log \frac{q_i}{p_i}.$$

**Remarks:**

- $D_{KL}$  is **not** symmetric
- $D_{KL}(p, q) \geq 0$

## Lemma

Consider any vector  $q \in \mathbb{R}_{\geq 0}^n$  and a vector  $g \in \mathbb{R}^n$ . Define  $w_i^* = q_i e^{-\eta g_i}$  for  $i = 1, \dots, n$ . Then  $\arg \min_{p \in \Delta_n} \{D_{KL}(p, q) + \eta \langle g, p \rangle\} = \frac{w^*}{\|w^*\|_1}$ .



# Exponential gradient descent

## Algorithm

① Initialize  $p^1 = \frac{1}{n} \mathbb{1}$  (uniform distribution).

② Repeat for  $t = 1, \dots, T$ :

- Obtain  $g^t = \nabla f(p_t)$ .

- Let  $w^{t+1} \in \mathbb{R}^n$  and  $p^{t+1} \in \Delta_n$  be defined as

$$w_i^{t+1} = p_i^t e^{-\eta g_i^t} \text{ and } p_i^{t+1} = \frac{w_i^{t+1}}{\sum_{j=1}^n w_j^{t+1}}.$$

③ Output  $\bar{p} = \frac{1}{T} \sum_{t=1}^T p^t$ .

## Thm.

Suppose that  $f : \Delta_n \rightarrow \mathbb{R}$  is a convex function which satisfies  $\|\nabla f(p)\| \leq G$  for all  $p \in \Delta_n$ . If we set  $\eta = \Theta\left(\frac{\sqrt{\log n}}{\sqrt{T}G}\right)$ , then after  $T = \Omega\left(\frac{G^2 \log n}{\varepsilon^2}\right)$  iterations of the algorithm, the point  $\bar{p} = \frac{1}{T} \sum_{t=1}^T p^t$  satisfies  $f(\bar{p}) \leq f(p^*) + \varepsilon$ .

# Multiplicative weights update

The analysis of the exponential gradient descent algorithm reveals that one can work with arbitrary vectors  $g^t$  instead of the gradients of  $f$ .

## Algorithm

① Initialize  $p^1 = \frac{1}{n} \mathbb{1}$  (uniform distribution).

② Repeat for  $t = 1, \dots, T$ :

- Obtain  $g^t$  from the oracle.
- Let  $w^{t+1} \in \mathbb{R}^n$  and  $p^{t+1} \in \Delta_n$  be defined as

$$w_i^{t+1} = p_i^t e^{-\eta g_i^t} \text{ and } p_i^{t+1} = \frac{w_i^{t+1}}{\sum_{j=1}^n w_j^{t+1}}.$$

③ Output  $p^1, \dots, p^T \in \Delta_n$

## Thm.

Assume that  $\|g^t\| \leq G$  for  $t = 1, \dots, T$ . If we set  $\eta = \Theta\left(\frac{\sqrt{\log n}}{\sqrt{TG}}\right)$ , then after  $T = \Theta\left(\frac{G^2 \log n}{\varepsilon^2}\right)$  iterations we have  $\frac{1}{T} \sum_{i=1}^T \langle g^t, p^t \rangle \leq \min_{p \in \Delta_n} \frac{1}{T} \sum_{i=1}^T \langle g^t, p \rangle + \varepsilon$ .

## Regularizers revisited

**Update rule:**  $x^{t+1} = \arg \min_{x \in K} \{D(x, x^t) + \eta \langle \nabla f(x^t), x \rangle\}$ .

The **Bregman divergence** of a function  $f : K \rightarrow \mathbb{R}$  at  $u, w \in K$  is defined to be

$$D_f(u, w) = f(u) - (f(w) + \langle \nabla f(w), u - w \rangle).$$

**Remark:** The Kullback-Leibler divergence is the Bregman divergence corresponding to the function  $H(x) = \sum_{i=1}^n x_i \log x_i - x_i$ .

For any convex regularizer  $R : \mathbb{R}^n \rightarrow \mathbb{R}$ , by denoting the gradient at step  $t$  by  $g^t$ , we have

$$\begin{aligned}x^{t+1} &= \arg \min_{x \in K} \{D_R(x, x^t) + \eta \langle g^t, x \rangle\} \\&= \arg \min_{x \in K} \{\eta \langle g^t, x \rangle + R(x) - R(x^t) - \langle \nabla R(x^t), x - x^t \rangle\} \\&= \arg \min_{x \in K} \{R(x) - \langle \nabla R(x^t) - \eta g^t, x \rangle\}.\end{aligned}$$

Suppose that there exists  $w^{t+1}$  such that  $\nabla R(w^{t+1}) = \nabla R(x^t) - \eta g^t$ . Then

$$\begin{aligned}x^{t+1} &= \arg \min_{x \in K} \{R(x) - \langle \nabla R(x^t) - \eta g^t, x \rangle\} \\&= \arg \min_{x \in K} \{R(x) - R(w^{t+1}) + \langle \nabla R(w^{t+1}), x \rangle\} \\&= \arg \min_{x \in K} \{D_R(x, w^{t+1})\}. \quad (D_R\text{-projection of } w^{t+1} \text{ onto } K)\end{aligned}$$

# Mirror descent I

Assume that the regularizer  $R : \Omega \rightarrow \mathbb{R}^n$  has a domain  $\Omega$  which contains  $K$  as a subset. Furthermore, assume that  $\nabla R : \Omega \rightarrow \mathbb{R}^n$  is a bijection (**mirror map**).

## Algorithm

**Input:** 1st-order oracle access to convex  $f : K \rightarrow \mathbb{R}$ , oracle access to  $\nabla R$  and its inverse, projection operator w.r.t.  $D_R(\cdot, \cdot)$ , initial point  $x^1 \in K$ , parameter  $\eta > 0$ , integer  $T > 0$ .

- ① Repeat for  $t = 1, \dots, T$ :
  - Obtain  $g^t = \nabla f(p_t)$ .
  - Let  $w^{t+1}$  be such that  $\nabla R(w^{t+1}) = \nabla R(x^t) - \eta \nabla f(x^t)$ .
  - Set  $x^{t+1} = \arg \min_{x \in K} D_R(x, w^{t+1})$ .
- ② **Output**  $\bar{x} = \frac{1}{T} \sum_{t=1}^T x^t$ .

## Remarks:

- The mirror map  $\nabla R$  and its inverse should be efficiently computable.
- The projection step  $\arg \min_{x \in K} D_R(x, w^{t+1})$  should be computationally easy to perform.

### Thm.

Let  $f : K \rightarrow \mathbb{R}$  and  $R : \Omega \rightarrow \mathbb{R}$  be convex functions with  $K \subseteq \Omega \subseteq \mathbb{R}^n$ . Suppose that the gradient map  $\nabla R : \Omega \rightarrow \mathbb{R}^n$  is a bijection,  $\|\nabla f(x)\| \leq G$  for  $x \in K$  (**bounded gradient**), and that  $D_R(x, y) \geq \frac{\sigma}{2} \|x - y\|^{*2}$  for  $x \in \Omega$  ( $R$  is  $\sigma$ -**strongly convex** w.r.t. dual norm  $\|\cdot\|^*$ ).

If we set  $\eta = \Theta \left( \frac{\sqrt{\sigma D_R(x^*, x^1)}}{\sqrt{T}G} \right)$ , then after  $T = \Theta \left( \frac{G^2 D_R(x^*, x^1)}{\sigma \varepsilon^2} \right)$  iterations the point  $\bar{x}$  satisfies  $f(\bar{x}) \leq f(x^*) + \varepsilon$ .

# Reading assignment



N. Vishnoi. Algorithms for convex optimization.

- Chapter 6
- Chapter 7



L.C. Lau. Convexity and optimization.

- Lecture 7

# Exercises

- ① Let  $G = (V, E)$  be an undirected graph and  $s, t \in V$ . Consider the following problem:

$$\min \sum_{uv \in E} |x_u - x_v|$$

$$\text{s.t. } x_s - x_t = 1$$

This is not a linear program in this form. Rewrite it as a linear program. (1pt)

- ② Let us consider the following functions:

$$f_1(w_1, w_2) = \frac{1}{2}w_1^2 + \frac{7}{2}w_2^2, \text{ and}$$

$$f_2(w_1, w_2) = 100(w_2 - w_1^2)^2 + (1 - w_1)^2 \quad (\text{Rosenbrock's function}).$$

- a Calculate the gradients of the functions. (2pts)
  - b Are these function convex? (2pts)
  - c Determine the global minimum of the functions. (2pts)
  - d Choose a starting point  $w = (w_1, w_2)$  within distance 5 from an optimal solution, and perform one step of the Gradient descent algorithm. (2pts)
- ③ Given a convex, differentiable function  $F : K \rightarrow \mathbb{R}$  over a convex subset  $K$  of  $\mathbb{R}^n$ , the Bergman divergence of  $x, y \in K$  is defined as

$$D_F(x, y) = F(x) - F(y) - \langle \nabla F(y), x - y \rangle.$$

Prove that  $D_F(x, y) \geq 0$ . (1pt)

# Newton's method

---

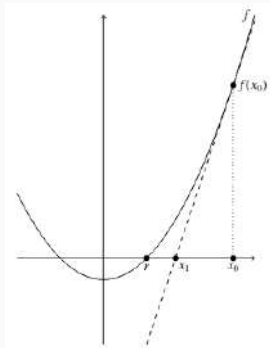


# Finding a root of a univariate function I

**Input:** A sufficiently differentiable function  $g : \mathbb{R} \rightarrow \mathbb{R}$ .

**Goal:** Find one of its roots, that is, a point  $r$  s.t.  $g(r) = 0$ .

**Setup:** Zeroth- and first-order access to  $g$  and point  $x_0$  that is sufficiently close to some root of  $g$ .



**Idea:** Given a point  $x$ , consider the tangent through  $(x, g(x))$ , and let  $x'$  be the intersection with the  $x$ -axis.

$$x' = x - \frac{g(x)}{g'(x)}$$

⇒ We hope to make progress in reaching a zero of  $g$ .

# Finding a root of a univariate function II

## Algorithm

- 1 Start with  $x_0 \in \mathbb{R}$ .
- 2 For  $t = 0, 1, \dots$ , let

$$x_{t+1} := x_t - \frac{g(x_t)}{g'(x_t)}$$

## Remarks:

- The method requires the differentiability of  $g$ .
- The convergence heavily depends on the starting point.

**Example:** Minimize  $f(x) := ax - \log x$  over all positive  $x > 0$ .

**Solution:** Take  $g(x) := f'(x)$ , and find a root of  $g$ . As  $f$  is convex, the root of  $g$  is an optimizer for  $f$ . We have

$$g(x) := f'(x) = a - \frac{1}{x}.$$

## Example 1

While the solution is obviously  $\frac{1}{a}$ , let us apply Newton's method.

**Reason 1:** To illustrate the method.

**Reason 2:** Early computers used Newton's method to compute the reciprocal as it only involved addition, subtraction, and multiplication.

We have

$$x_{t+1} = x_t - \frac{g(x_t)}{g'(x_t)} = 2x_t - ax_t^2.$$

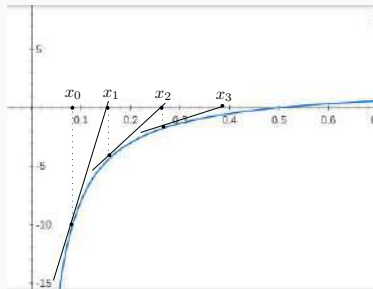
Define  $e_t := 1 - ax_t$ . Then

$$e_{t+1} = e_t^2.$$

- If  $|e_0| < 1$ , then  $e_t \rightarrow 0$ .
- If  $|e_0| = 1$ , then  $e_t = 1$  for  $t \geq 1$ .  $\Rightarrow$
- If  $|e_0| > 1$ , then  $e_t \rightarrow \infty$ .
- If  $0 < x_0 < \frac{2}{a}$ , then  $x_t \rightarrow \frac{1}{a}$ .
- If  $x_0 = \frac{2}{a}$ , then  $x_t = 0$  for  $t \geq 1$ .
- If  $x_0 > \frac{2}{a}$ , then  $x_t \rightarrow -\infty$ .

## Example II

- If  $0 < x_0 < \frac{2}{a}$ , then  $x_t \rightarrow \frac{1}{a}$ .
- If  $x_0 = \frac{2}{a}$ , then  $x_t = 0$  for  $t \geq 1$ .
- If  $x_0 > \frac{2}{a}$ , then  $x_t \rightarrow -\infty$ .



⇒ The right starting point has a crucial impact on whether the algorithm succeeds or fails.

**Note:** By modifying the function  $g$ , e.g.  $g(x) := x - \frac{1}{a}$ , we get a different algorithm to compute  $\frac{1}{a}$ . However, some of them might not make sense ( $x_{t+1} = x_t - \frac{g(x_t)}{g'(x_t)} = \frac{1}{a}$ ), or might not be efficient.

# Convergence I

**Recall:** The distance  $e_t = 1 - ax_t = a\left(\frac{1}{a} - x_t\right)$  was squared at every iteration in the example.

**Question:** Do we get quadratic convergence in general? **YES!**

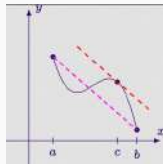
## Thm.

Suppose  $g : \mathbb{R} \rightarrow \mathbb{R}$  is twice differentiable and its second derivative is continuous,  $r \in \mathbb{R}$  is a root of  $g$ ,  $x_0 \in \mathbb{R}$  is a starting point, and  $x_1 = x_0 - \frac{g(x_0)}{g'(x_0)}$ .

Then  $|x_1 - r| \leq M|x_0 - r|^2$ , where  $M = \sup_{\xi \in (r, x_0)} \left| \frac{g''(\xi)}{2g'(x_0)} \right|$ .

The proof relies on the **Mean value theorem**, stating that if  $h : \mathbb{R} \rightarrow \mathbb{R}$  is a continuous function on the closed interval  $[a, b]$  and differentiable on the open interval  $(a, b)$ , then there exists  $c \in (a, b)$  s.t.

$$h'(c) = \frac{h(b) - h(a)}{b - a}.$$



## Convergence II

### Proof of the theorem.

By considering the second-order Taylor approximation of  $g$  around  $x_0$ , we have

$$g(r) = g(x_0) + (r - x_0)g'(x_0) + \frac{1}{2}(r - x_0)^2 g''(\xi)$$

for some  $\xi \in (r, x_0)$ .

From the definition of  $x_1$ , we know that  $g(x_0) = g'(x_0)(x_0 - x_1)$ . Furthermore,  $g(r) = 0$  as  $r$  is a root, hence

$$0 = g'(x_0)(x_0 - x_1) + (r - x_0)g'(x_0) + \frac{1}{2}(r - x_0)^2 g''(\xi).$$

This implies

$$g'(x_0)(x_1 - r) = \frac{1}{2}(r - x_0)^2 g''(\xi).$$

Therefore

$$|x_1 - r| = \left| \frac{g''(\xi)}{2g'(x_0)} \right| |x_0 - r|^2 \leq M|x_0 - r|^2,$$

where  $M$  is as stated in the theorem. □

## Convergence III

Assuming  $M$  is a small constant (say  $M \leq 1$ , and this holds throughout the procedure) and  $|x_0 - r| < \frac{1}{2}$ , the theorem implies quadratically fast convergence of  $x_t$  to  $r$ .

Indeed, after  $t$  steps we have

$$|x_t - r| \leq |x_0 - r|^{2^t} \leq 2^{-2^t}.$$

⇒ If  $t \approx \log \log \frac{1}{\varepsilon}$ , then  $|x_t - r| \leq \varepsilon$ .

**Summary:** Newton's method is very efficient!

See the Newton-Raphson method on GeoGebra!

# Multivariate functions

**Input:** A sufficiently differentiable function  $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$ .

**Goal:** Find one of its roots, that is, a point  $r$  s.t.  $g(r) = 0$ .

[**Be careful:** here 0 denotes the all-zero vector.]

**Setup:** Zeroth- and first-order access to  $g$  and point  $x_0$  that is sufficiently close to some root of  $g$ .

**Original idea:** Given a point  $x$ , define

$$x' = x - \frac{g(x)}{g'(x)}$$

$\Rightarrow$  Now  $g(x)$  is a vector while  $g'(x)$  is the **Jacobian** matrix of  $g$  at  $x$ , i.e.  $J_g(x)$  is the matrix of partial derivatives

$$\left[ \frac{\partial g_i}{\partial x_j}(x) \right]_{1 \leq i, j \leq n}$$

Hence the update rule becomes

$$x_{t+1} := x_t - J_g(x_t)^{-1} g(x_t).$$



# Newton's method for unconstrained optimization

What is the connection between convex programs and Newton's method?

**Key observation:** Minimizing a differentiable convex function in the unconstrained setting is equivalent to finding a root of its derivative.

**Input:** Sufficiently differentiable convex function  $f$ .

**Goal:** Find  $x^* := \arg \min_{x \in \mathbb{R}^n} f(x)$ .

**Recall:**

- $\nabla f$  is a function from  $\mathbb{R}^n$  to  $\mathbb{R}^n$ .
- The Jacobian  $J_{\nabla f}$  is the Hessian  $\nabla^2 f$ .

$\Rightarrow$  The update rule is

$$x_{t+1} := x_t - (\nabla^2 f(x_t))^{-1} \nabla f(x_t).$$

For ease of notation, we define the **Newton step** at  $x$  to be

$$n(x) := -(\nabla^2 f(x_t))^{-1} \nabla f(x_t).$$

Hence  $x_{t+1} := x_t + n(x_t)$ .

## Newton's method as a second-order method

Suppose we would like to find a global minimum of  $f$  and  $x_0$  is our current approximate solution. Let

$$\tilde{f}(x) := f(x_0) + \langle x - x_0, \nabla f(x_0) \rangle + \frac{1}{2}(x - x_0)^T \nabla^2 f(x_0)(x - x_0).$$

**Idea:** Set the next point to be the minimizer of  $\tilde{f}$ .

*[Roughly, we hope that  $\tilde{f}$  approximates  $f$  locally, and so the new point should be an even better approximation to  $x^*$ .]*

We have to find  $x_0 := \arg \min_{x \in \mathbb{R}^n} \tilde{f}(x)$ . Assuming that  $f$  is strictly convex (and so  $\nabla^2 f(x_0)$  is invertible), this is equivalent to solving  $\nabla \tilde{f}(x) = 0$ , that is,

$$\nabla f(x_0) + \nabla^2 f(x_0)(x - x_0) = 0,$$

leading to  $x_1 = x_0 - (\nabla^2 f(x_0))^{-1} \nabla f(x_0) = x_0 + n(x_0)$ .

$\Rightarrow$  We recovered Newton's method!

**Consequence:** When applied to strictly convex quadratic functions, i.e. of the form  $h(x) = x^T Mx + b^T x$  for  $M \succ 0$ , then after one iteration we land in the unique minimizer.

# Newton's method vs. gradient descent

Is Newton's method a "better" algorithm?

**Pros:** It uses the Hessian to perform the iterations, hence it is more powerful.

**Cons:** One iteration is now more costly, as a second-order oracle is needed.

More precisely, to compute  $x_{t+1}$ , we need to solve the following system:

$$(\nabla^2 f(x_t)) x = \nabla f(x_t).$$

- In general, this takes  $O(n^3)$  time using Gaussian elimination, or  $O(n^\omega)$  using fast matrix multiplication.
- If the Hessian has a special form, e.g. it is Laplacian, then there are nearly-linear time Laplacian solvers.

# Newton-Euclidian condition

## NE condition

Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be a function,  $x^*$  be one of its minimizers,  $x_0$  be arbitrary. We say that the **NE**( $M$ ) condition is satisfied for  $M > 0$  if there exists an Euclidean ball  $B(x^*, R)$  of radius  $R$  containing  $x_0$  and constants  $h, L > 0$  such that  $M \geq \frac{L}{2h}$  and

- for every  $x \in B(x^*, R)$ , we have  $\|\nabla^2 f(x)^{-1}\| \leq \frac{1}{h}$ ,
- for every  $x, y \in B(x^*, R)$ , we have  $\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L\|x - y\|_2$ .

Here the norm of a matrix is the so-called spectral norm, defined as

$$\|A\| := \sup_{x \in \mathbb{R}^n} \frac{\|Ax\|_2}{\|x\|_2}.$$

## Thm.

Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  and  $x^*$  be one of its minimizers. Let  $x_0$  be arbitrary and define  $x_1 := x_0 + n(x_0)$ . If the **NE**( $M$ ) condition is satisfied, then

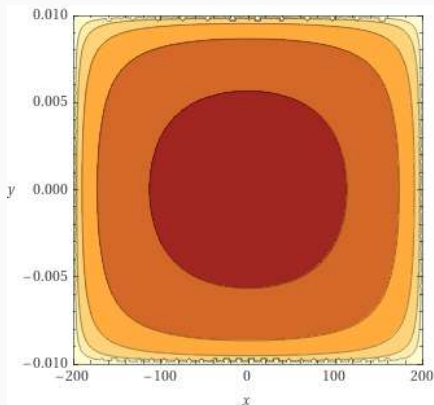
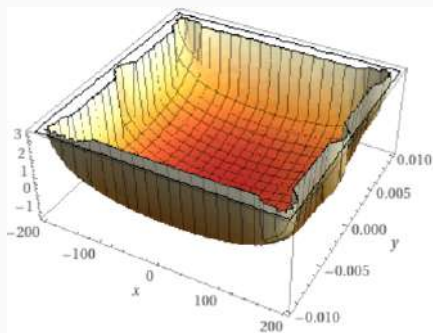
$$\|x_1 - x^*\|_2 \leq M\|x_0 - x^*\|_2^2.$$

# Problem with the convergence I

**Fact:** The theorem is stated with respect to quantities based on Euclidean norm  $\|\cdot\|_2$ , which makes it hard to apply in many cases.

**Example:** For  $K_1, K_2 > 0$  (large constants), consider

$$f(x_1, x_2) := -\log(K_1 - x_1) - \log(K_1 + x_1) - \log\left(\frac{1}{K_2} - x_2\right) - \log\left(\frac{1}{K_2} + x_2\right).$$



## Problem with the convergence II

Now the Hessian of  $f$  is

$$\nabla^2 f(x) = \begin{pmatrix} \frac{1}{(K_1 - x_1)^2} + \frac{1}{(K_1 + x_1)^2} & 0 \\ 0 & \frac{1}{(\frac{1}{K_2} - x_2)^2} + \frac{1}{(\frac{1}{K_2} + x_2)^2} \end{pmatrix}$$

$\Rightarrow$  It can be verified that  $M$ , which determines the quadratic convergence of Newton's method, is at least  $\Omega(K_1^2 K_2^2)$ . Therefore, even when the initial point is close to the optimal solution  $x^*$ , the guarantee in the theorem is too weak to imply that in one step the distance drops.

However, Newton's method does in fact converge rapidly to  $x^*$ !

## Local norm I

Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be a strictly convex function, i.e., the Hessian  $\nabla^2 f(x)$  is positive definite for every  $x \in \mathbb{R}^n$ . We define the **local inner product** at every point  $x$  as

$$\langle u, v \rangle_x := u^T \nabla^2 f(x) v \quad \text{for } u, v \in \mathbb{R}^n.$$

The corresponding **local norm** is

$$\|u\|_x := \sqrt{u^T \nabla^2 f(x) u} \quad \text{for } u \in \mathbb{R}^n.$$

**Recall:** When deriving the gradient descent algorithm, we picked the direction of steepest descent which is a solution to the following problem:

$$\arg \max_{\|u\|=1} (-\langle \nabla f(x), u \rangle).$$

The optimal direction w.r.t. the Euclidean norm  $\|\cdot\| = \|\cdot\|_2$  is in the direction  $-\nabla f(x)$ .

## Local norm II

**Idea:** What if instead maximize over all  $u$  of local norm 1? That is,

$$\arg \max_{\|u\|_x=1} (-\langle \nabla f(x), u \rangle) = \arg \max_{u^T \nabla^2 f(x) u = 1} (-\langle \nabla f(x), u \rangle).$$

*[We would like to capture the “shape” of  $f$  around  $x$  with our choice of the norm, and our best guess is the quadratic term given by the Hessian.]*

Using Cauchy-Schwarz, the optimal solution is in the direction

$$-\nabla^2 f(x)^{-1} \nabla f(x),$$

which is exactly the **Newton step!**

Indeed, set  $v := \nabla^2 f(x)^{-1} \nabla f(x)$ , and observe that

$$\begin{aligned} -\langle \nabla f(x), u \rangle &= -\left\langle \nabla^2 f(x)^{\frac{1}{2}} v, \nabla^2 f(x)^{\frac{1}{2}} u \right\rangle \\ &\leq \sqrt{v^T \nabla^2 f(x) v} \sqrt{u^T \nabla^2 f(x) u} \\ &= \|v\|_x \|u\|_x, \end{aligned}$$

and equality holds if and only if  $\nabla^2 f(x)^{\frac{1}{2}} u = -\nabla^2 f(x)^{\frac{1}{2}} v$ . This is the same as

$$u = -v = -\nabla^2 f(x)^{-1} \nabla f(x).$$



# Conclusion

Newton's method can be interpreted as a **steepest descent** algorithm, where the Newton step is the direction of steepest descent with respect to the local norm.

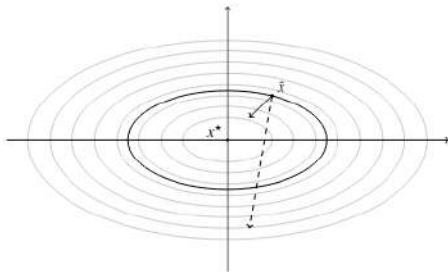


Figure 9.2 Illustration of the difference in directions suggested by gradient descent and Newton's method for the function  $x_1^2 + 4x_2^2$ .  $\bar{x} = (1, 1)$  and  $x^* = (0, 0)$ . The solid arrow is a step of length  $1/2$  in the direction  $(-1, -1)$  (Newton step) and the dashed arrow is a step of length  $1/2$  in the direction  $(-1, -4)$  (gradient step).

# Reading assignment



N. Vishnoi. Algorithms for convex optimization.

- Chapter 9



L.C. Lau. Convexity and optimization.

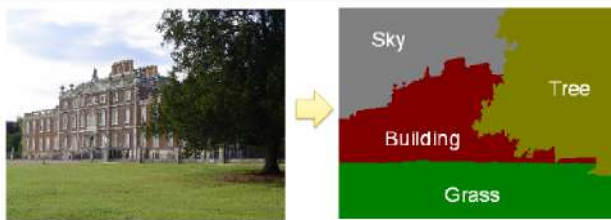
- Lecture 12

# Submodular functions

---

# Motivation

## Semantic segmentation:



**Question:** How can we map pixels to objects?

# Motivation

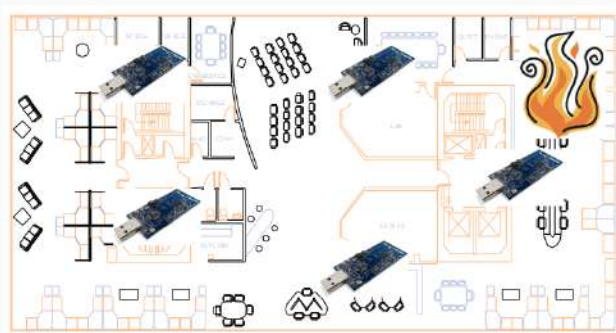
**Document summarization:**



**Question:** How can we select representative sentences?

# Motivation

Sensor placement:



**Question:** How to place the sensors optimally?

# Motivation

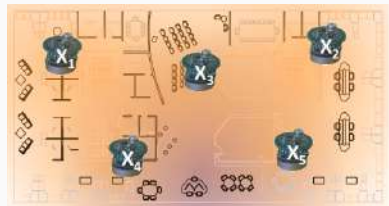
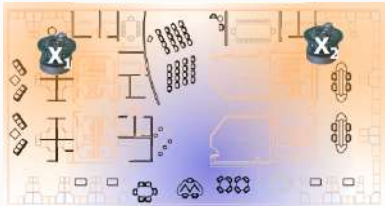
Sensor placement:



**Obs.** Some placements are more effective than others.

# Motivation

## Sensor placement:



**Obs.** Adding a new sensor has “more value” in the first case than in the second case.



# Discrete optimization

**Setup:** Given a set  $\mathcal{F}$  of feasible solutions and a function  $f : \mathcal{F} \rightarrow \mathbb{R}$ , solve

$$\max\{f(X) : X \in \mathcal{F}\}$$

$$\min\{f(X) : X \in \mathcal{F}\}$$

Arbitrary set functions are hopelessly difficult to optimize...for 100 items, we should check  $2^{100}$  sets!

**Goal:** Find sufficient conditions that make the problem tractable.

**Recall:** In the continuous case,  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  can be **minimized** if  $f$  is **convex**, and **maximized** if  $f$  is **concave**.

⇒ Is it possible to find discrete counterparts?

**Note:** Many problems in real life applications assume a discrete setting, therefore it would be crucial to provide efficient algorithms.

# Set functions

Let  $S$  be a set of size  $n$ . A **set function** is a function of the form  $f : 2^S \rightarrow \mathbb{R}$ , where  $2^S$  denotes the set of all subsets of  $S$ .

Given a set  $X \subseteq S$  and  $s \in S$ , we denote by

$$X + s := X \cup \{s\},$$

$$X - s := X \setminus \{s\}.$$

The **marginal value** of  $s$  w.r.t.  $X$  is

$$f(s|X) = f(X + s) - f(X).$$

Further properties:

- **Monotone:** if  $X \subseteq Y \subseteq S$ , then  $f(X) \leq f(Y)$ .
- **Nonnegative:**  $f(X) \geq 0$  for  $X \subseteq S$ .
- **Normalized:**  $f(\emptyset) = 0$  (we will usually assume this throughout).

# Modular functions

A set function  $f : 2^S \rightarrow \mathbb{R}$  is **modular** if for all  $X \subseteq S$  we have

$$f(X) = \sum_{s \in X} f(s).$$

**Intuitively:** Associate a weight  $w_s$  with each  $s \in S$ , and set  $f(X) = \sum_{s \in X} w_s$ .

⇒ Discrete analogue of linear functions.

# Submodularity

A set function  $f : 2^S \rightarrow \mathbb{R}$  is **submodular** if for all  $X \subseteq Y \subseteq S$  and  $s \in S \setminus Y$  we have

$$f(s|X) \geq f(s|Y).$$

**Intuitively:** The gain is more from a new element if we start with a smaller set.

**Example:**  $f(\text{new car}|\{\text{bike}\}) \geq f(\text{new car}|\{\text{bike, car, private jet}\})$

[The marginal value of an element exhibits *diminishing marginal returns*.]

**Remarks:**

- $f$  is **supermodular** if  $-f$  is submodular
- $f$  is **modular** if and only if it is both sub- and supermodular

## Equivalent definition I

A set function  $f : 2^S \rightarrow \mathbb{R}$  is submodular if and only if for all  $X, Y \subseteq S$  we have

$$f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y).$$

### Proof.

$\Leftarrow$  Let  $X \subseteq Y \subseteq S$  and  $s \in S \setminus Y$ . Then  $X \cup Y = Y$  and  $X \cap Y = X$ , hence

$$f(s|X) = f(X + s) - f(X) \geq f(Y + s) - f(Y) = f(s|Y).$$

$\Rightarrow$  Assume that  $f$  is submodular, and let  $X \setminus Y = \{x_1, \dots, x_k\}$ . Furthermore, let  $X_i := \{x_1, \dots, x_i\}$  for  $i = 1, \dots, k$ . Then

$$f((X \cap Y) \cup X_1) - f(X \cap Y) \geq f(Y \cup X_1) - f(Y)$$

$$f((X \cap Y) \cup X_2) - f((X \cap Y) \cup X_1) \geq f(Y \cup X_2) - f(Y \cup X_1)$$

$\vdots$

$$f((X \cap Y) \cup X_k) - f((X \cap Y) \cup X_{k-1}) \geq f(Y \cup X_k) - f(Y \cup X_{k-1})$$

---

$$f(X) - f(X \cap Y) \geq f(X \cup Y) - f(Y)$$



## Equivalent definition II

A set function  $f : 2^S \rightarrow \mathbb{R}$  is supermodular if and only if for all  $X, Y \subseteq S$  we have

$$f(X) + f(Y) \leq f(X \cap Y) + f(X \cup Y).$$

A set function  $f : 2^S \rightarrow \mathbb{R}$  is modular if and only if for all  $X, Y \subseteq S$  we have

$$f(X) + f(Y) = f(X \cap Y) + f(X \cup Y).$$

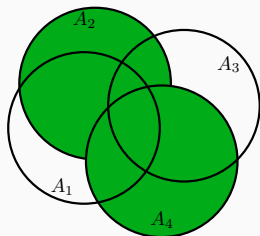
**Remark:** These functions play a crucial role in combinatorial optimization, and also in machine learning.

## Example I - Coverage

**Coverage function.** Assume that for  $s \in S$ , we are given a measurable set  $A_s$ .  
Then

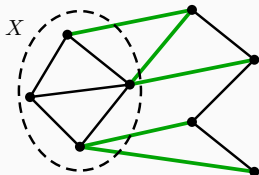
$$f(X) := \left| \bigcup_{s \in X} A_s \right|$$

is submodular.

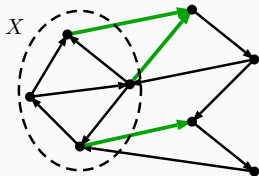


## Example II - Cuts in graphs

**Cut function.** Let  $G = (V, E)$  be an undirected graph. Then  $f(X) := d_G(X)$  is submodular.



**In- and out-degrees.** Let  $D = (V, A)$  be a directed graph. Then the out-degree  $f(X) := d_D^+(X)$  and the in-degree  $f(X) := d_D^-(X)$  functions are submodular.





## Example III - Entropy

**Entropy.** Let  $(\xi_s)_{s \in S}$  be random variables with finite number of values in  $(\mathcal{X}_s)_{s \in S}$ , respectively. For a set  $X = \{s_1, \dots, s_k\} \subseteq S$ , the joint entropy is

$$f(X) = - \sum_{x_{s_1} \in \mathcal{X}_{s_1}} \cdots \sum_{x_{s_k} \in \mathcal{X}_{s_k}} P(x_{s_1}, \dots, x_{s_k}) \log_2 P(x_{s_1}, \dots, x_{s_k}).$$

Then  $f$  is submodular.

**Mutual information.**  $i(X) := f(X) + f(S \setminus X) - f(S)$  is submodular.

# Properties

- ① **Positive linear combinations:** If  $f_1, \dots, f_k$  are submodular and  $\lambda_i \geq 0$  for  $i = 1, \dots, k$ , then  $\sum_{i=1}^k \lambda_i f_i$  is submodular.
- ② **Reflection:** If  $f$  is submodular, then  $g(X) := f(S \setminus X)$  is submodular.
- ③ **Restriction:** If  $X \subseteq S$  and  $f$  is submodular, then  $g(Y) := f(X \cap Y)$  is submodular.
- ④ **Conditioning:** If  $X \subseteq S$  and  $f$  is submodular, then  $g(Y) := f(X \cup Y)$  is submodular.
- ⑤ **Contraction:** If  $X \subseteq S$  and  $f$  is submodular, then  $g(Y) := f(X \cup Y) - f(X)$  is submodular.
- ⑥ **Maximum/minimum:** If  $f$  and  $g$  are submodular, then  $\max\{f, g\}$  and  $\min\{f, g\}$  are **not** necessarily submodular.

## Submodularity and concavity

Given a set  $X \subseteq S$ , let  $1_X$  denote its **characteristic vector**, that is,

$$(1_X)_s = \begin{cases} 1 & \text{if } s \in X, \\ 0 & \text{otherwise.} \end{cases}$$

A set function  $f : 2^S \rightarrow \mathbb{R}$  can be thought of as a function defined on  $\{0, 1\}^S$ .

**Recall:** A function  $f : \mathbb{R} \rightarrow \mathbb{R}$  is concave if  $f'(x)$  is non-increasing in  $x$ .

**Now:** A function  $f : \{0, 1\}^S \rightarrow \mathbb{R}$  is submodular if the “discrete derivative”

$$\partial_s f(x) = f(x + e_s) - f(x)$$

is non-increasing in  $x$ .

**Furthermore:** If a function  $g : \mathbb{R}_+ \rightarrow \mathbb{R}$  is concave, then  $f(X) := g(|X|)$  is submodular.

## Submodularity and convexity I

Let  $f : \{0, 1\}^S \rightarrow \mathbb{R}$  be a set function. For a vector  $c \in \mathbb{R}^S$ , let  $s_1, \dots, s_n$  be an ordering of the elements  $S$  such that  $c_{s_1} \geq \dots \geq c_{s_n}$ . Furthermore, let  $S_i := \{s_1, \dots, s_i\}$  for  $i = 1, \dots, n$ . The **Lovász-extension** of  $f$  on  $c$  is

$$\begin{aligned}\hat{f}(c) &:= c_{s_n} f(S_n) + \sum_{i=1}^{n-1} (c_{s_i} - c_{s_{i+1}}) f(S_i) \\ &= c_{s_1} f(S_1) + \sum_{i=2}^n c_{s_i} (f(S_i) - f(S_{i-1})) \\ &= c_{s_1} f(S_1) + \sum_{i=2}^n c_{s_i} f(s_i | S_{i-1}).\end{aligned}$$

$\Rightarrow$  The sum of the marginal gains weighted by the components of  $c$ .

## Submodularity and convexity II

- $\hat{f}$  is an extension of  $f$  in the sense that  $\hat{f}(1_X) = f(X)$  for  $X \subseteq S$ .
- $\hat{f}$  is piecewise affine.
- $\hat{f}$  is **convex** if and only if  $f$  is **submodular**.
- When restricted to  $[0, 1]^S$ ,  $\hat{f}$  attains its minimum at one of the vertices, that is,

$$\min_{c \in [0, 1]^S} \hat{f}(c) = \min_{X \subseteq S} f(X).$$

**Conclusion:** Submodular functions share properties in common with both convex and concave functions. So, can we minimize/maximize them?

# Submodular minimization I

**Input:** A submodular function  $f : 2^S \rightarrow \mathbb{R}$ .

**Goal:** Find  $\arg \min_{X \subseteq S} f(X)$ .

By the properties of the Lovász extension, this is equivalent to finding

$$\arg \min_{x \in [0,1]^n} \hat{f}(x).$$

## Thm.

The Lovász extension  $\hat{f}$  can be minimized using the Ellipsoid method in  $O(n^8 \log^2 n)$  time.

## Remarks:

- $O(n^6)$  algorithm (Schrijver (2000), Iwata et al. (2001), Orlin (2009)).
- Faster algorithms in special cases (cuts, flows).

# Submodular minimization II

- ① **Symmetric submodular functions.** The function  $f$  is symmetric if  $f(X) = f(S \setminus X)$ . In this case

$$2f(X) = f(X) + f(S \setminus X) \geq f(\emptyset) + f(S) = 2f(\emptyset) = 0,$$

hence the minimum is trivially attained at  $S$ .

⇒ Usually, we are interested in  $\arg \min_{\emptyset \neq X \subset S} f(X)$ .

## Queyranne, 1998

If  $f$  is symmetric, then there is a fully combinatorial algorithm for solving  $\arg \min_{\emptyset \neq X \subset S} f(X)$  in  $O(n^3)$  time.

- ② **Constrained setting.** A simple constraint can make submodular minimization hard, e.g.,  $n^{1/2}$ -hardness for  $\min_{X \subseteq S, |X| \geq k} f(S)$ .  
⇒ In such cases, one might be interested in finding approximate solutions.

## Example - Clustering

**Input:** A set  $S$ .

**Goal:** Find a partition into  $k$  clusters  $S_1, \dots, S_k$  such that

$$g(S_1, \dots, S_k) = \sum_{i=1}^k f(S_i)$$

is minimized, where  $f$  is a submodular function (e.g. entropy or cut function).

**Observation:** For  $k = 2$ , the function  $g(X) = f(X) + f(S \setminus X)$  is symmetric and submodular, thus Queyranne's algorithm applies.

- ① Let  $\mathcal{P}_1 = \{S\}$ .
- ② For  $i = 1, \dots, k - 1$ :
  - a For each  $S_j \in \mathcal{P}_i$ , let  $\mathcal{P}_i^j$  be a partition obtained by splitting  $S_j$  using Queyranne's algorithm.
  - b Set  $\mathcal{P}_{i+1} = \arg \min f(\mathcal{P}_i^j)$ .

### Thm.

If  $\mathcal{P}$  is the partition provided by the greedy splitting algorithm, then

$$f(\mathcal{P}) \leq \left(2 - \frac{2}{k}\right) f(\mathcal{P}_{opt}).$$



# Submodular maximization

The maximization of submodular functions naturally comes up in applications.

The function is often assumed to be monotone, that is,  $f(X) \leq f(Y)$  for  $X \subseteq Y \subseteq S$ .

$\Rightarrow$  When  $f$  is monotone, then the maximum is clearly attained on  $S$ .

**Hence:**

- Non-monotone submodular maximization (e.g. Max Cut).
- Monotone submodular maximization with constraints (e.g.  $\max_{X \subseteq S, |X| \leq k} f(X)$ ).

# Monotone submodular maximization

## Greedy algorithm

- 1 Set  $S_0 := \emptyset$ .
- 2 For  $i = 1, 2, \dots, k$ :
  - Pick an element  $s$  maximizing  $f(s|S_{i-1})$ .
  - If the marginal value is non-negative, set  $S_i := S_{i-1} + s$ .
  - Otherwise stop.

## Nemhauser, Wolsey, Fisher

The greedy algorithm gives a  $(1 - \frac{1}{e})$ -approximation for the problem  $\max_{X \subseteq S, |X| \leq k} f(X)$ , where  $f$  is monotone submodular.

### Remark:

- When instead of  $|X| \leq k$  a matroid constraint  $X \in \mathcal{I}$  is given, then the greedy algorithm gives a  $\frac{1}{2}$ -approximation.

## Further approaches

- ① **Partial enumeration:** Guess the first few elements, then run the greedy algorithm.
- ② **Local search:** Switch up to  $t$  elements if the function value is decreased.
  - 1/3-approximation for unconstrained (non-monotone) maximization
  - Further results for matroid constraints.

# Reading assignment



submodularity.org

# Exercises

- 1 Verify that the in-degree function of a directed graph is submodular. (2pts)
- 2 Prove the following statements. (4pts)
  - a The non-negative linear combination of submodular functions is submodular.
  - b The reflection of a submodular function is submodular.
  - c The restriction of a submodular function is submodular.
  - d The contraction of a submodular function is submodular.
- 3 Provide examples showing that the maximum/minimum of two submodular functions are not necessarily submodular. (2pts)
- 4 Give a 2-approximation for the Max Cut problem in undirected graphs, where the goal is to find a set  $X$  with maximum degree. (2pts) [Hint: try to find a greedy approach.]

# Integer programming

---

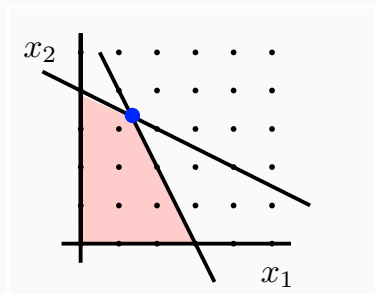
# Example revisited

Example:

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$



# Example revisited

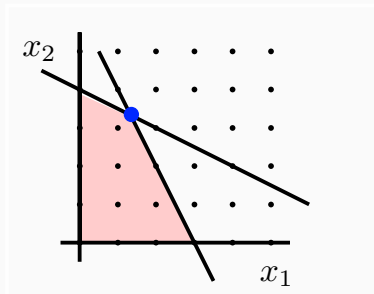
Example:

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$

$$x_1, x_2 \in \mathbb{Z}$$





# Example revisited

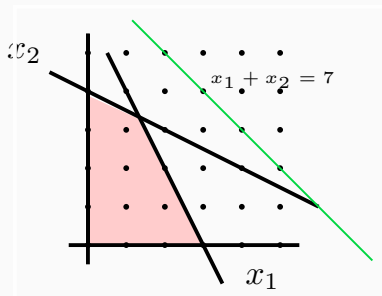
Example:

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$

$$x_1, x_2 \in \mathbb{Z}$$



# Example revisited

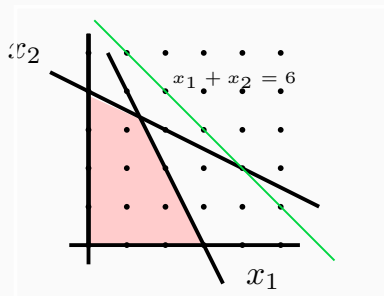
Example:

$$x_1 + 2 \cdot x_2 \leq 8$$

$$2 \cdot x_1 + x_2 \leq 6$$

$$x_1, x_2 \geq 0$$

$$x_1, x_2 \in \mathbb{Z}$$



## Another example

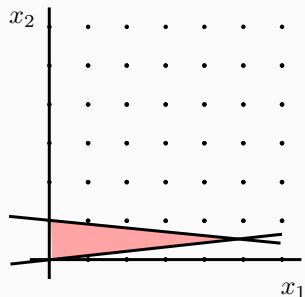
Example:

$$x_1 + 10 \cdot x_2 \leq 10$$

$$x_1 - 10 \cdot x_2 \leq 0$$

$$x_1, x_2 \geq 0$$

$$\max\{x_1\}$$



## Another example

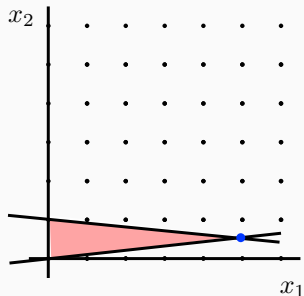
Example:

$$x_1 + 10 \cdot x_2 \leq 10$$

$$x_1 - 10 \cdot x_2 \leq 0$$

$$x_1, x_2 \geq 0$$

$$\max\{x_1\}$$



# Another example

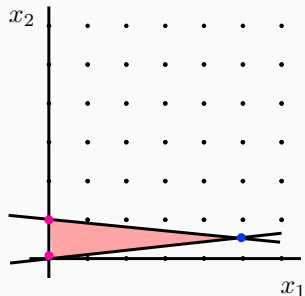
Example:

$$x_1 + 10 \cdot x_2 \leq 10$$

$$x_1 - 10 \cdot x_2 \leq 0$$

$$x_1, x_2 \geq 0$$

$$\max\{x_1\}$$



## Another example

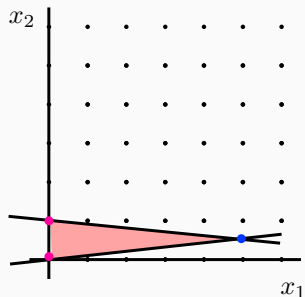
Example:

$$x_1 + 10 \cdot x_2 \leq 10$$

$$x_1 - 10 \cdot x_2 \leq 0$$

$$x_1, x_2 \geq 0$$

$$\max\{x_1\}$$



The **fractional** optimum can be far from the **integer** one.

# Approaches

**Bad news:** integer programming is NP-complete

# Approaches

**Bad news:** integer programming is **NP-complete**

**Good news:** there exist efficient algorithms

- **totally unimodular** matrices
  - every square submatrix has determinant 0, +1 or -1
- **cutting plane** methods
  - adding further inequalities that separate the actual optimum from the convex hull of the true feasible set
- **branch and bound** methods
  - systematically enumerating the candidate solutions, forming a rooted tree
- **rounding** methods (threshold rounding, iterative rounding)
  - rounding the coordinates of an optimal fractional solution
- **heuristic** methods (tabu search, hill climbing, simulated annealing, ant colony optimization, etc)
  - some would call these 'voodoo'...



# Branch and bound I

$$\begin{array}{ll} \min & c(x) \\ \text{s.t.} & x \in F \end{array}$$

Here  $F$  is the set of integer feasible solutions to the problem.

## Ideas:

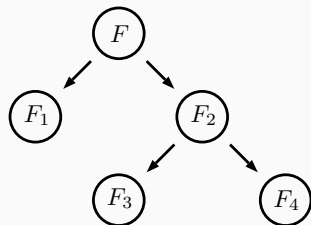
- Partition  $F$  into subsets  $F_1, \dots, F_k$ , and solve the subproblems  $\min c(x)$  s.t.  $x \in F_i$ . [May be as difficult as the original one, hence split into further subproblems - **branching part.**]
- Compute lower bounds  $b(F_i)$  for the subproblems. [A lower bound might be easy to obtain, e.g. LP relaxation - **bounding part.**]
- Maintain an upper bound  $U$  on the optimal cost. [E.g. the cost of the best feasible solution thus far.]

**Key observation:** If  $b(F_i) \geq U$ , then the subproblem need not be considered further.

# Branch and bound II

## Algorithm (general step):

- 1 Select an active subproblem  $F_i$ .
- 2 If the subproblem is infeasible, delete it; otherwise compute  $b(F_i)$ .
  - If  $b(F_i) \geq U$ , delete the subproblem.
  - If  $b(F_i) < U$ , either determine an optimal solution for  $F_i$ , or break it into further (active) subproblems.



## Remarks:

- Choosing the subproblem, e.g. BFS or DFS.
- Computing the lower bounds, e.g. LP relaxation.
- Breaking into subproblems.

# Rounding methods

Given a minimization problem, an  $\alpha$ -approximation algorithm provides a solution of value at most  $\alpha \cdot OPT$ .

## Integer program

$$\min c^T \cdot x$$

$$A \cdot x \leq b$$

$$x \in \mathbb{Z}^n$$

Naiv approach:

1. remove the integrality constraint,
2. solve the corresponding LP, and
3. round the entries of the solution to get an integer solution.

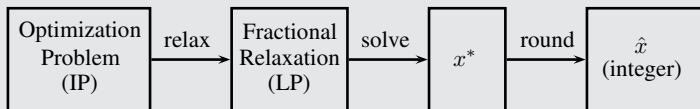
Problems:

- the solution may not be feasible
- the solution may not be optimal

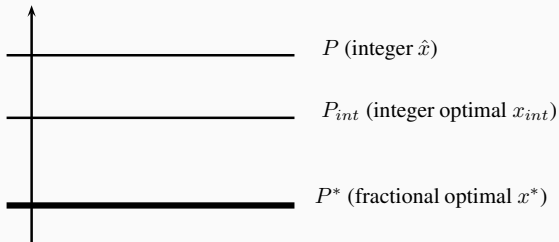
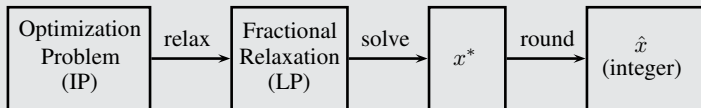
Maintain feasibility.

Approximation?

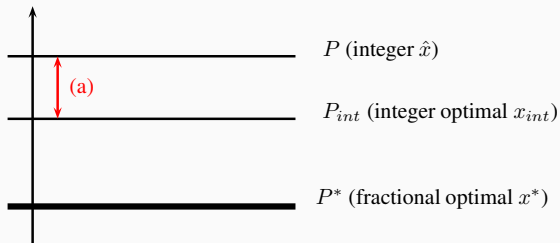
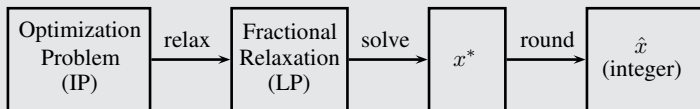
# Analysing the solution



# Analysing the solution

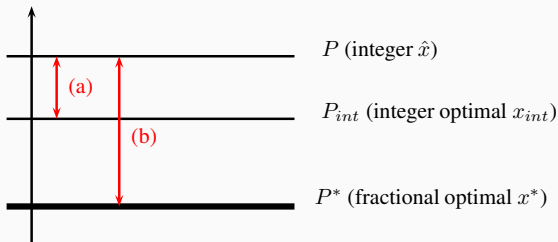
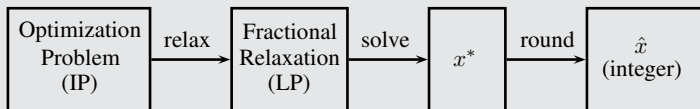


# Analysing the solution



(a) = Approximation ratio between  $\hat{x}$  and  $x_{int}$ .

# Analysing the solution



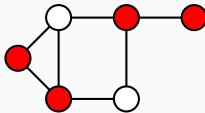
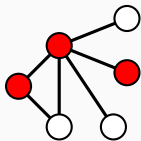
(a) = Approximation ratio between  $\hat{x}$  and  $x_{int}$ .

(b) = Approximation ratio between  $\hat{x}$  and  $x^*$ .

# Vertex cover I

## Problem

Find a minimum number of vertices covering every edge of a graph.

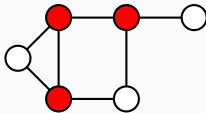
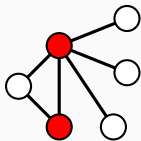




# Vertex cover I

## Problem

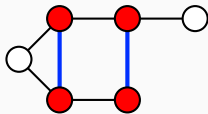
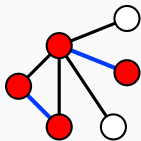
Find a minimum number of vertices covering every edge of a graph.



# Vertex cover I

## Problem

Find a minimum number of vertices covering every edge of a graph.



## Simple algorithm:

**Step 1.** Take an inclusionwise maximal matching  $M$ .

**Step 2.** Consider the end vertices of the matching edges.

## Observation

This gives a 2-approximation.

- One of Karp's 21 NP-complete problems.
- Moreover, it is APX-complete.
  - No better than 1.3606-approx. unless  $P = NP$ .
  - No better than 2-approx. assuming UGC.

## Vertex cover II

### IP formulation

$$\min \sum_{v \in V} x_v$$

$$x_u + x_v \geq 1 \quad \text{for } uv \in E$$

$$x_v \in \{0, 1\} \quad \text{for } v \in V$$

# Vertex cover II

## IP formulation

$$\min \sum_{v \in V} x_v$$

$$x_u + x_v \geq 1 \quad \text{for } uv \in E$$

$$x_v \in \{0, 1\} \quad \text{for } v \in V$$

## LP relaxation

$$\min \sum_{v \in V} x_v$$

$$x_u + x_v \geq 1 \quad \text{for } uv \in E$$

$$x_v \geq 0 \quad \text{for } v \in V$$

# Vertex cover II

## IP formulation

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ x_u + x_v & \geq 1 \quad \text{for } uv \in E \\ x_v & \in \{0, 1\} \quad \text{for } v \in V \end{aligned}$$

### Step 1.

Take a fractional solution  $x^*$ .

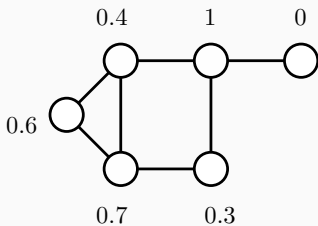
### Step 2.

Define

$$\hat{x}_v = \begin{cases} 1 & \text{if } x_v^* \geq 1/2, \\ 0 & \text{otherwise.} \end{cases}$$

## LP relaxation

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ x_u + x_v & \geq 1 \quad \text{for } uv \in E \\ x_v & \geq 0 \quad \text{for } v \in V \end{aligned}$$



# Vertex cover II

## IP formulation

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ x_u + x_v & \geq 1 \quad \text{for } uv \in E \\ x_v & \in \{0, 1\} \quad \text{for } v \in V \end{aligned}$$

## LP relaxation

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ x_u + x_v & \geq 1 \quad \text{for } uv \in E \\ x_v & \geq 0 \quad \text{for } v \in V \end{aligned}$$

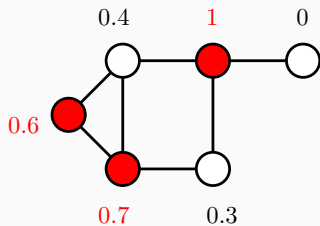
### Step 1.

Take a fractional solution  $x^*$ .

### Step 2.

Define

$$\hat{x}_v = \begin{cases} 1 & \text{if } x_v^* \geq 1/2, \\ 0 & \text{otherwise.} \end{cases}$$



# Vertex cover II

## IP formulation

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ x_u + x_v & \geq 1 \quad \text{for } uv \in E \\ x_v & \in \{0, 1\} \quad \text{for } v \in V \end{aligned}$$

## LP relaxation

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ x_u + x_v & \geq 1 \quad \text{for } uv \in E \\ x_v & \geq 0 \quad \text{for } v \in V \end{aligned}$$

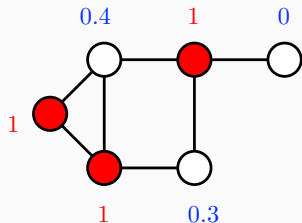
### Step 1.

Take a fractional solution  $x^*$ .

### Step 2.

Define

$$\hat{x}_v = \begin{cases} 1 & \text{if } x_v^* \geq 1/2, \\ 0 & \text{otherwise.} \end{cases}$$



# Vertex cover II

## IP formulation

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ x_u + x_v & \geq 1 \quad \text{for } uv \in E \\ x_v & \in \{0, 1\} \quad \text{for } v \in V \end{aligned}$$

## LP relaxation

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ x_u + x_v & \geq 1 \quad \text{for } uv \in E \\ x_v & \geq 0 \quad \text{for } v \in V \end{aligned}$$

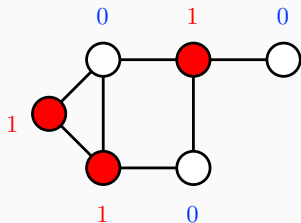
### Step 1.

Take a fractional solution  $x^*$ .

### Step 2.

Define

$$\hat{x}_v = \begin{cases} 1 & \text{if } x_v^* \geq 1/2, \\ 0 & \text{otherwise.} \end{cases}$$





# Vertex cover II

## IP formulation

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ x_u + x_v & \geq 1 \quad \text{for } uv \in E \\ x_v & \in \{0, 1\} \quad \text{for } v \in V \end{aligned}$$

## LP relaxation

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ x_u + x_v & \geq 1 \quad \text{for } uv \in E \\ x_v & \geq 0 \quad \text{for } v \in V \end{aligned}$$

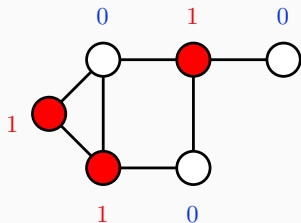
### Step 1.

Take a fractional solution  $x^*$ .

### Step 2.

Define

$$\hat{x}_v = \begin{cases} 1 & \text{if } x_v^* \geq 1/2, \\ 0 & \text{otherwise.} \end{cases}$$



# Vertex cover II

## IP formulation

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ x_u + x_v & \geq 1 \quad \text{for } uv \in E \\ x_v & \in \{0, 1\} \quad \text{for } v \in V \end{aligned}$$

## LP relaxation

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ x_u + x_v & \geq 1 \quad \text{for } uv \in E \\ x_v & \geq 0 \quad \text{for } v \in V \end{aligned}$$

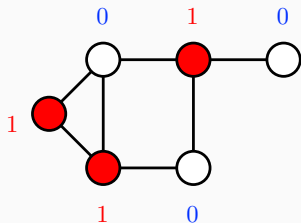
### Step 1.

Take a fractional solution  $x^*$ .

### Step 2.

Define

$$\hat{x}_v = \begin{cases} 1 & \text{if } x_v^* \geq 1/2, \\ 0 & \text{otherwise.} \end{cases}$$



# Vertex cover II

## IP formulation

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ x_u + x_v & \geq 1 \quad \text{for } uv \in E \\ x_v & \in \{0, 1\} \quad \text{for } v \in V \end{aligned}$$

## LP relaxation

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ x_u + x_v & \geq 1 \quad \text{for } uv \in E \\ x_v & \geq 0 \quad \text{for } v \in V \end{aligned}$$

### Step 1.

Take a fractional solution  $x^*$ .

### Step 2.

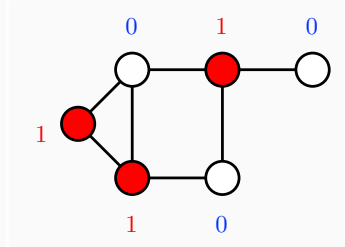
Define

$$\hat{x}_v = \begin{cases} 1 & \text{if } x_v^* \geq 1/2, \\ 0 & \text{otherwise.} \end{cases}$$

### Proof.

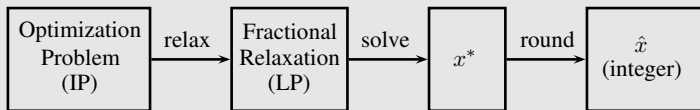
Note that  $\hat{x}$  is integral, feasible, and  $\hat{x}_v \leq 2 \cdot x_v^*$ . Hence

$$\sum_{v \in V} \hat{x}_v \leq 2 \cdot \sum_{v \in V} x_v^* \leq 2 \cdot \text{OPT}.$$

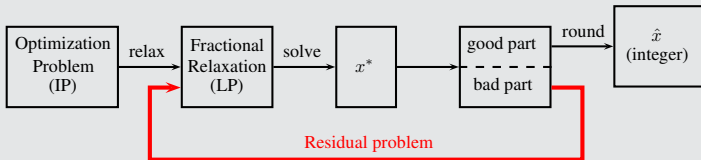


# Threshold vs. iterative rounding

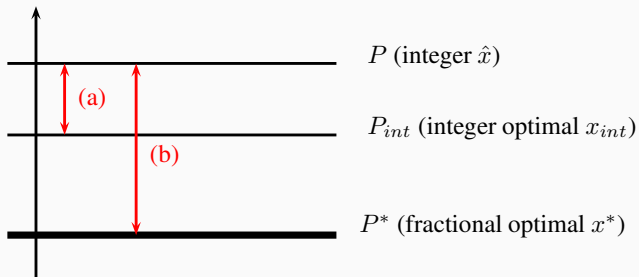
## Threshold rounding



## Iterative rounding



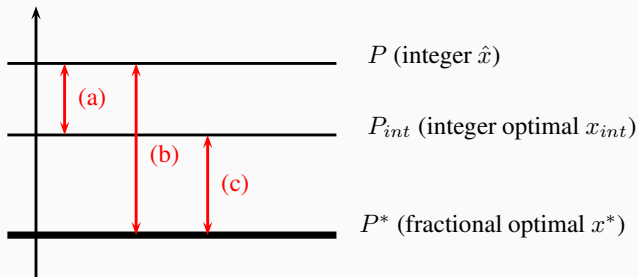
# Integrality gap



(a) = Approximation ratio between  $\hat{x}$  and  $x_{int}$ .

(b) = Approximation ratio between  $\hat{x}$  and  $x^*$ .

# Integrality gap



(a) = Approximation ratio between  $\hat{x}$  and  $x_{int}$ .

(b) = Approximation ratio between  $\hat{x}$  and  $x^*$ .

(c) = Integrality gap.

# Heuristics - Local search

$$\begin{array}{ll} \min & c(x) \\ \text{s.t.} & x \in F \end{array}$$

## Algorithm:

- 1 Start at some  $x \in F$ .
- 2 Evaluate  $c(x)$ , and evaluate  $c(y)$  for “neighbors”  $y \in F$  of  $x$ .
  - If  $c(y) < c(x)$ , the move to  $y$  and repeat.
  - Otherwise stop: *local optimum* has been found.

## Remarks:

- Specifics are determined once “neighbors” are defined.
- Simplex method can be viewed as a special case.
- In practice: run repeatedly starting from different initial solutions.
- Tradeoff: **better solution** is likely to be obtained when considering **larger neighborhood**, but this results in **slower running time**.

# Heuristics - Simulated annealing I

**Main drawback of local search:** Only finds local minimum.

**Idea:** Allow occasional moves to feasible solutions with higher costs.

**Algorithm:** For every state  $x \in F$ , a set  $N(x) \subseteq F$  of neighbors is given ( $y \in N(x) \Leftrightarrow x \in N(y)$ ).

- 1 Start from state  $x \in F$ .
- 2 Select a random neighbor  $y$  of  $x$  with probability  $q_{xy}$ .  
[Here  $q_{xy} \geq 0$  and  $\sum_{y \in N(x)} q_{xy} = 1$ .]
- 3 Compute the difference  $c(y) - c(x)$ .
  - If  $c(y) \leq c(x)$ , then move to state  $y$ .
  - If  $c(y) > c(x)$ , then move to state  $y$  with probability  $e^{-(c(y)-c(x))/T}$ .

**Remarks:**

- When the **temperature**  $T$  is small - cost increases are unlikely.
- When  $T$  is large - the value of  $c(y) - c(x)$  has insignificant effect.



## Heuristics - Simulated annealing II

The procedure evolves as a Markov chain. Let  $A = \sum_{z \in F} e^{-c(z)/T}$ .

**Steady-state distribution:**

$$\pi(x) = \frac{e^{-c(x)/T}}{A},$$

$\Rightarrow \pi(x)$  falls exponentially with  $c(x)$ . Hence if  $T$  is small, then almost all of the steady-state probability is concentrated on states minimizing  $c(x)$  **globally**. Should we set  $T$  to some very small constant then?

**Drawback:** the lower the value of  $T$ , the harder it is to escape from a local minimum and the longer it takes to reach steady-state.


**Instead:** Let the temperature vary with time:

$$T(t) = \frac{C}{\log t}.$$

**Thm.**

If  $C$  is sufficiently large, then  $\lim_{t \rightarrow \infty} P(x(t) \text{ is optimal}) = 1$ .

# Reading assignment

-  D. Bertsimas, J.N. Tsitsiklis. Introduction to linear optimization.
  - Chapter 11, Sections 11.2, 11.6, and 11.7

# Exercises

- ① Consider the following integer programming problem.

$$\begin{array}{ll} \text{maximize} & x_1 + 2x_2 \\ \text{subject to} & -3x_1 + 4x_2 \leq 4 \\ & 3x_1 + 2x_2 \leq 11 \\ & 2x_1 - x_2 \leq 5 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \quad \text{integer} \end{array}$$

Use a figure to answer the following questions.

- a What is the optimal cost of the linear programming relaxation? What is the optimal cost of the integer programming problem? (2pts)
- b What is the convex hull of the set of all solutions to the integer programming problem? (3pts)

# Exercises

- ② A company is manufacturing  $k$  different products using  $m$  resources. The amounts of available resources are given, together with the requirement of each of them for the different products. The selling price of the products are also known.
- Ⓐ Write up an IP model that aims at maximizing the total profit. (3pts)
  - Ⓑ Adjust the model if starting the production of product  $i$  requires a cost of  $s_i$ . (2pts)