

# **Optimization - 3 CFU**

Francesca Maggioni

---

Academic Year 2023-2024

Department of Management, Information and Production Engineering  
University of Bergamo



## Francesca Maggioni

- Associate Professor of Operations Research (ssd MAT/09)

Department of Management, Information & Production Engineering -  
University of Bergamo

- E-mail: [francesca.maggioni@unibg.it](mailto:francesca.maggioni@unibg.it)
- Office hours: Monday 2:30-3:30 pm
- Home page: <http://www.francescamaggioni.it>



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Course Organization

## Lessons:

- Tuesday: 3:30 PM – 6:30 PM

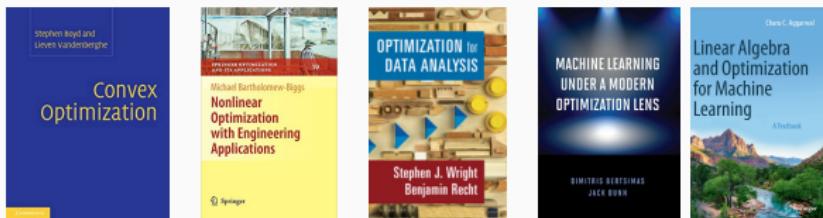
## Materials and Announcements:

- <https://elearning15.unibg.it/>



# References

- Stephen Boyd & Lieven Vandenberghe (2004). *Convex Optimization*, Cambridge University Press.
- Michael Bartholomew-Biggs (2008) *Nonlinear Optimization with Engineering Applications*, Springer
- Stephen J. Wright & Benjamin Recht (2022) *Optimization for Data Analysis*, Cambridge University Press.
- Dimitris Bertsimas and Jack Dunn (2019) *Machine Learning Under a Modern Optimization Lens*, Dynamic Ideas LLC
- Charu C. Aggarwal Linear Algebra and Optimization for Machine Learning



- Lecture notes and further reading will be posted to the e-learning platform.



# Assessment Method and Criteria

The **final exam** consists in two parts:

- Oral discussion about applied assignments and case studies (50% of the final grade). Students may work in small groups or individually.
- Final oral exam (50% of the final grade).



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

## Outline

- Introduction to non linear optimization (Constrained and unconstrained Optimization)
- Convex optimization
- Gradient Methods
- Subgradient Methods
- **Stochastic Gradient Descent**
- Coordinate Descent
- Primal-dual context
- Second-Order methods including Quasi-Newton
- Derivative-free optimization
- Support vector machine models
- decision tree via mixed integer linear programming.



# Motivation of the course

## Optimization is everywhere:

- Machine learning, big data, statistics, data analysis of all kinds, finance, logistics, planning, control theory, mathematics, search engines, simulations, and many other applications.
- **Mathematical Modeling**: defining & modeling the optimization problem.
- **Computational Optimization**: running an (appropriate) optimization algorithm.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

## Optimization for Machine Learning:

- **Mathematical Modeling**: defining & and measuring the machine learning model.
- **Computational Optimization**: learning the model parameters.
- Theory vs. practice:
  - libraries are available, algorithms treated as black box by most practitioners;
  - Not here: we look inside the algorithms and try to understand why and how fast they work!



- Main approaches:
  - Gradient Descent.
  - Stochastic Gradient Descent (SGD).
  - Coordinate Descent.
- History:
  - 1847: Cauchy proposes gradient descent.
  - 1950s: Linear Programs, soon followed by non-linear, SGD.
  - 1980s: General optimization, convergence theory.
  - 2005-today: Large scale optimization, convergence of SGD.



## Definition

A *Mathematical Programming* (or *Mathematical Optimization*) problem can be formulated as:

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in X \end{aligned}$$

where

- $X \subseteq \text{dom}(f)$  is the **set of feasible solutions**
- $f : \text{dom}(f) \rightarrow \mathbb{R}$  is the **objective function**.

**Convention:** formulate each problem as a minimization problem

$$\max \{f(\mathbf{x}) : \mathbf{x} \in X\} = - \min \{-f(\mathbf{x}) : \mathbf{x} \in X\}.$$



## Definition

- A problem for which there is no feasible solution ( $X = \emptyset$ ) is said to be **infeasible**; in this case we agree to write

$$\min \{f(\mathbf{x}) : \mathbf{x} \in X\} = +\infty.$$

- A problem for which  $f$  is not bounded from below in  $X$  is hence said to be **unbounded**; in this case we write

$$\min \{f(\mathbf{x}) : \mathbf{x} \in X\} = -\infty$$

$$\left[ \forall M > 0 \exists \mathbf{x} \in X \text{ such that } f(\mathbf{x}) < -M \right].$$



# Local and global optimal solutions

## Definition

Let  $f : \text{dom}(f) \rightarrow \mathbb{R}$  be and let  $X \subseteq \text{dom}(f)$ . A point  $\mathbf{x} \in X$  is a **minimizer** of  $f$  over  $X$  if

$$f(\mathbf{x}) \leq f(\mathbf{y}) \quad \forall \mathbf{y} \in X.$$

The point  $\mathbf{x}$  is said to be **optimal solution** and the corresponding value  $f(\mathbf{x})$  is said **optimal value**. This solution is not necessarily unique.

## Definition

A **local minimum** of  $f : \text{dom}(f) \rightarrow \mathbb{R}$  is a point  $\mathbf{x}$  such that there exists  $\varepsilon > 0$  with

$$f(\mathbf{x}) \leq f(\mathbf{y}) \quad \forall \mathbf{y} \in \text{dom}(f) \text{ satisfying } \|\mathbf{y} - \mathbf{x}\| < \varepsilon.$$



# Non Linear Optimization

- A **non linear programming problem (NLP)**, is a mathematical programming problem of the kind:

$$\min f(\mathbf{x})$$

$$s.t. \quad g_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, m \quad (\text{NLP})$$

$$\mathbf{x} \geq \mathbf{0}$$

where  $f : \text{dom}(f) \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$  and  $g_i : \text{dom}(g_i) \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$  are given functions and at least one of these functions (objective function or constraints) is non linear.

## Example

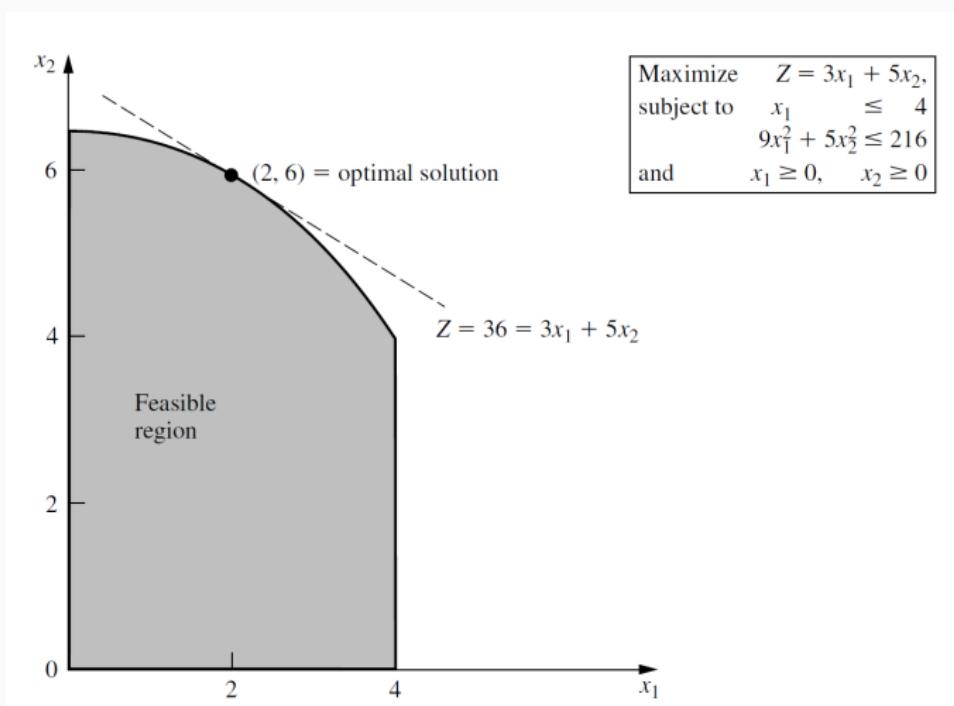
$$\max \left( x_1 - \frac{1}{2} \right)^2 + \left( x_2 - \frac{1}{2} \right)^2$$

$$s.t. \quad x_1 + x_2 \geq 1, \quad x_1 \leq 1$$

No algorithm that will solve *every* specific problem fitting this format is available. However, substantial progress has been made for some important special cases of this problem by making various assumptions about these functions.

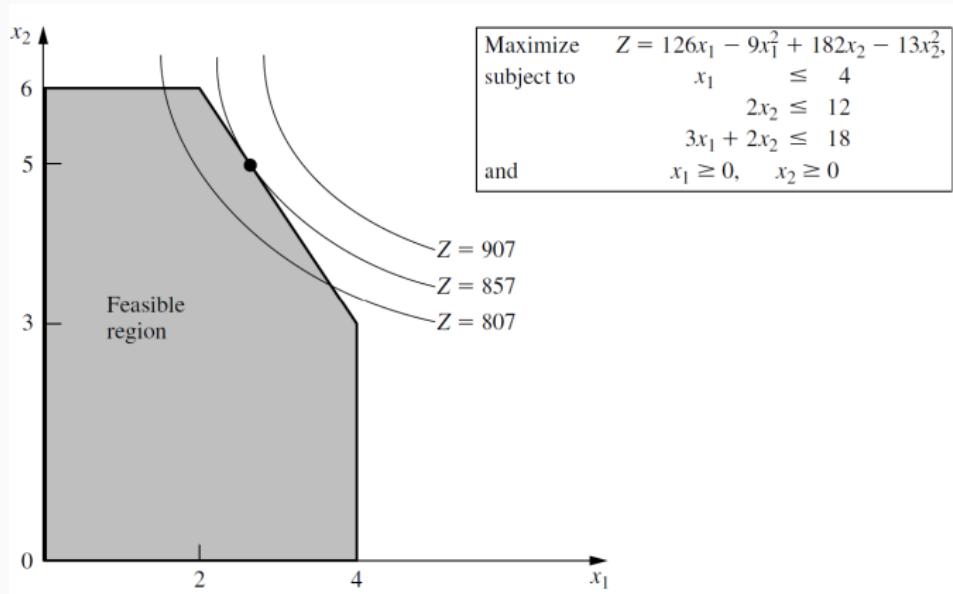
# Graphical illustration of non linear programming problems

- The optimal solution still lies on the boundary of the feasible region.  
However, it **is not a vertex** as in LP!



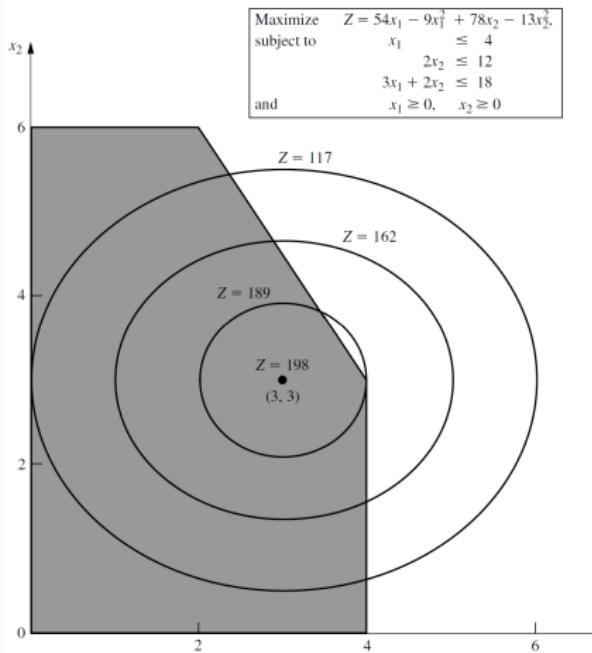
# Graphical illustration of non linear programming problems

- Non linear objective function subject to linear constraints.
- Optimal solution on the boundary:  $(\frac{8}{3}, 5)$



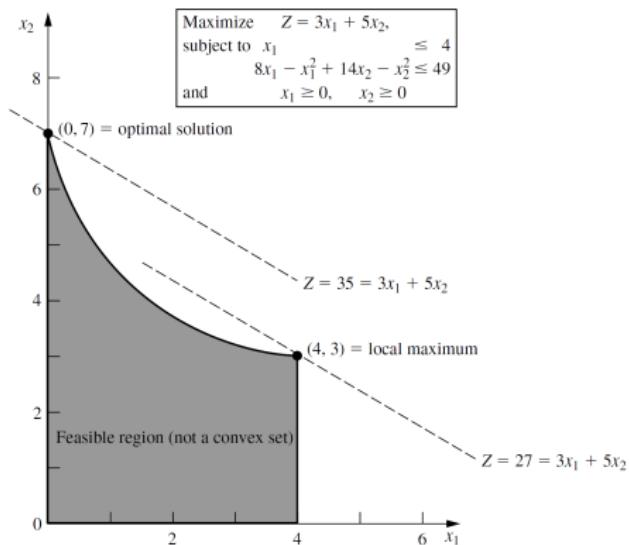
# Graphical illustration of non linear programming problems

- Optimal solution inside the feasible region: (3, 3).



A general algorithm for solving similar problems needs to consider all solutions in the feasible region, not just those on the boundary.

# Graphical illustration of non linear programming problems



- The feasible region is not a convex set. Consequently, we cannot guarantee that a local maximum is a global maximum. In fact, this example has two local maxima,  $(0, 7)$  and  $(4, 3)$ , but only  $(0, 7)$  is a global maximum. To guarantee that a local maximum is a global maximum for a nonlinear programming problem we need to consider a **convex programming problem**.

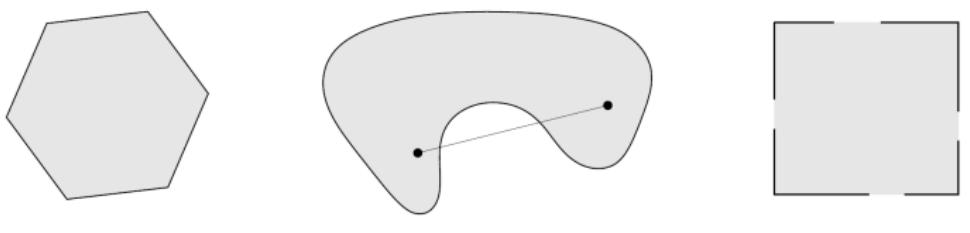
## **Convex Optimization**

---

# Convex Sets

A set  $C$  is **convex** if the line segment between any two points of  $C$  lies in  $C$ , i.e., if for any  $\mathbf{x}, \mathbf{y} \in C$  and any  $\lambda$  with  $0 \leq \lambda \leq 1$ , we have:

$$\lambda\mathbf{x} + (1 - \lambda)\mathbf{y} \in C.$$



- *Left:* Convex.
- *Middle:* Not convex, since line segment not in set.
- *Right:* Not convex, since some, but not all boundary points are contained in the set.



# Properties of Convex Sets

- Intersections of convex sets are convex.
  - ◊ Let  $C_i, i \in I$  be convex sets, where  $I$  is a (possibly infinite) index set. Then  $C = \cap_{i \in I} C_i$  is a convex set.
- Projections onto convex sets are unique, and often efficient to compute.



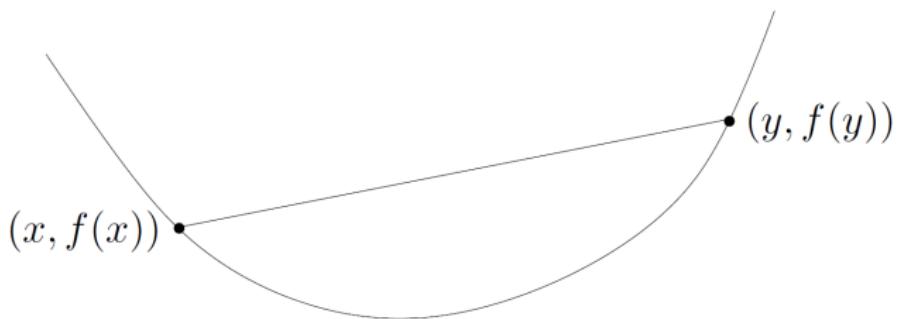
UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Convex Functions

A function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is **convex** if

1.  $\text{dom}(f)$  is a convex set;
2. for all  $\mathbf{x}, \mathbf{y} \in \text{dom}(f)$  and any  $\lambda$  with  $0 \leq \lambda \leq 1$ , we have:

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y}).$$



**Geometrically:** The line segment between  $(\mathbf{x}; f(\mathbf{x}))$  and  $(\mathbf{y}; f(\mathbf{y}))$  lies above the graph of  $f$ .

**Convex Optimization Problems** are of the form:

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ s.t. \quad & \mathbf{x} \in X \end{aligned}$$

where both:

- $f$  is a convex function;
- $X \subset \text{dom}(f)$  is a convex set (note:  $\mathbb{R}^d$  is convex).

## Crucial Property of Convex Optimization Problems

Every **local minimum** is a **global minimum**.



## Remark

*For convex optimization problems, all algorithms (Coordinate Descent, Gradient Descent, Stochastic Gradient Descent, Projected and Proximal Gradient Descent) do converge to the global optimum!*

## Example Theorem

The **convergence rate** is proportional to  $\frac{1}{t}$ , i.e.:

$$f(\mathbf{x}_t) - f(\mathbf{x}^*) \leq \frac{c}{t}$$

where  $\mathbf{x}^*$  is some optimal solution to the problem.

Meaning: Approximation error converges to 0 over time.



$f$	Algorithm	Rate	# Iter	Cost/iter
non-smooth	center of gravity	$\exp\left(-\frac{t}{n}\right)$	$n \log\left(\frac{1}{\varepsilon}\right)$	$1 \nabla,$ $1 n\text{-dim } f$
non-smooth	ellipsoid method	$\frac{R}{r} \exp\left(-\frac{t}{n^2}\right)$	$n^2 \log\left(\frac{R}{r\varepsilon}\right)$	$1 \nabla,$ mat-vec $\times$
non-smooth	Vaidya	$\frac{Rn}{r} \exp\left(-\frac{t}{n}\right)$	$n \log\left(\frac{Rn}{r\varepsilon}\right)$	$1 \nabla,$ mat-mat $\times$
quadratic	CG	$\frac{\text{exact}}{\exp\left(-\frac{t}{n}\right)}$	$\frac{n}{\kappa \log\left(\frac{1}{\varepsilon}\right)}$	$1 \nabla$
non-smooth, Lipschitz	PGD	$RL/\sqrt{t}$	$R^2 L^2/\varepsilon^2$	$1 \nabla,$ 1 proj.
smooth	PGD	$\beta R^2/t$	$\beta R^2/\varepsilon$	$1 \nabla,$ 1 proj.
smooth	AGD	$\beta R^2/t^2$	$R\sqrt{\beta/\varepsilon}$	$1 \nabla$
smooth (any norm) strong, conv., Lipschitz	FW	$\beta R^2/t$	$\beta R^2/\varepsilon$	$1 \nabla,$ 1 LP
strong, conv., smooth	PGD	$L^2/(\alpha t)$	$L^2/(\alpha\varepsilon)$	$1 \nabla,$ 1 proj.
strong, conv., smooth	PGD	$R^2 \exp\left(-\frac{t}{\kappa}\right)$	$\kappa \log\left(\frac{R^2}{\varepsilon}\right)$	$1 \nabla,$ 1 proj.
strong, conv., smooth	AGD	$R^2 \exp\left(-\frac{t}{\sqrt{\kappa}}\right)$	$\sqrt{\kappa} \log\left(\frac{R^2}{\varepsilon}\right)$	$1 \nabla$
$f + g,$ $f$ smooth, $g$ simple	FISTA	$\beta R^2/t^2$	$R\sqrt{\beta/\varepsilon}$	$1 \nabla$ of $f$ Prox of $g$
$\max_{y \in \mathcal{Y}} \varphi(x, y),$ $\varphi$ smooth	SP-MP	$\beta R^2/t$	$\beta R^2/\varepsilon$	MD on $\mathcal{X}$ MD on $\mathcal{Y}$
linear, $\mathcal{X}$ with $F$ $\nu$ -self-conc.	IPM	$\nu \exp\left(-\frac{t}{\sqrt{\nu}}\right)$	$\sqrt{\nu} \log\left(\frac{\nu}{\varepsilon}\right)$	Newton step on $F$
non-smooth	SGD	$BL/\sqrt{t}$	$B^2 L^2/\varepsilon^2$	$1 \text{ stoch. } \nabla,$ 1 proj.
non-smooth, strong, conv.	SGD	$B^2/(\alpha t)$	$B^2/(\alpha\varepsilon)$	$1 \text{ stoch. } \nabla,$ 1 proj.
$f = \frac{1}{m} \sum f_i$ $f_i$ smooth strong, conv.	SVRG	—	$(m + \kappa) \log\left(\frac{1}{\varepsilon}\right)$	$1 \text{ stoch. } \nabla$



## Definition

The **graph** of a function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is defined as:

$$\{(x, f(x)) \mid x \in \text{dom}(f)\}.$$

## Definition

The **epigraph** of a function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is defined as:

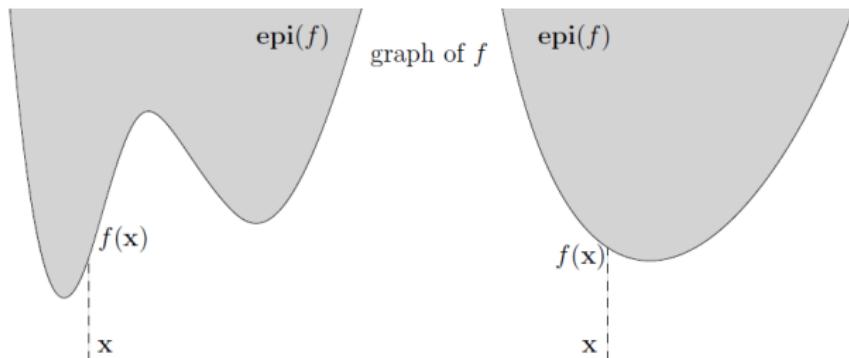
$$\text{epi}(f) := \{(x, \alpha) \in \mathbb{R}^{d+1} \mid x \in \text{dom}(f), \alpha \geq f(x)\}.$$



# Convex Functions and Sets

## Remark

A function is convex iff its epigraph is a convex set.



Examples of convex functions:

- Linear functions:  $f(\mathbf{x}) = \mathbf{a}^\top \mathbf{x}$ .
- Affine functions:  $f(\mathbf{x}) = \mathbf{a}^\top \mathbf{x} + b$ .
- Exponential:  $f(\mathbf{x}) = e^{\alpha \mathbf{x}}$ .
- Norms. Every norm on  $\mathbb{R}^d$  is convex.

## Convexity of a norm $\|\mathbf{x}\|$

By the triangle inequality:

$$\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$$

and homogeneity of a norm:

$$\|a\mathbf{x}\| = |a| \cdot \|\mathbf{x}\|$$

with  $a$  scalar, we have:

$$\|\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}\| \leq \|\lambda\mathbf{x}\| + \|(1 - \lambda)\mathbf{y}\| = \lambda\|\mathbf{x}\| + (1 - \lambda)\|\mathbf{y}\|.$$

We used the triangle inequality for the inequality and homogeneity for the equality.



## Recall: The Cauchy-Schwarz Inequality

Let  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$ . **Cauchy-Schwarz Inequality:**

$$|\mathbf{u}^\top \mathbf{v}| \leq \|\mathbf{u}\| \cdot \|\mathbf{v}\|.$$

Notation:

- $\mathbf{u} = (u_1, \dots, u_d)$ ,  $\mathbf{v} = (v_1, \dots, v_d)$   $d$ -dimensional column vectors with real entries;
- $\mathbf{u}^\top$  transpose of  $\mathbf{u}$ , a  $d$ -dimensional row vector;
- $\mathbf{u}^\top \mathbf{v} = \sum_{i=1}^d u_i v_i$  scalar (or inner) product of  $\mathbf{u}$  and  $\mathbf{v}$ ;
- $|\mathbf{u}^\top \mathbf{v}|$  absolute value of  $\mathbf{u}^\top \mathbf{v}$ ;
- $\|\mathbf{u}\| = \sqrt{\mathbf{u}^\top \mathbf{u}} = \sqrt{\sum_{i=1}^d u_i^2}$  Euclidean norm of  $\mathbf{u}$ .



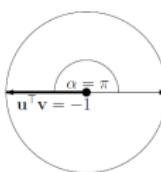
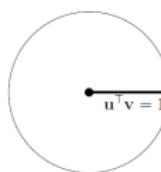
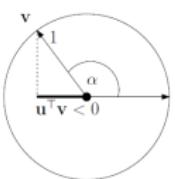
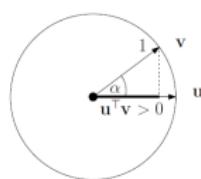
# Recall: The Cauchy-Schwarz Inequality

For non-zero elements this is equivalent to:

$$-1 \leq \frac{\mathbf{u}^\top \mathbf{v}}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|} \leq 1.$$

Fraction can be used to define the angle  $\alpha$  between  $\mathbf{u}$  and  $\mathbf{v}$ :

$$\cos(\alpha) = \frac{\mathbf{u}^\top \mathbf{v}}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|}.$$



Examples for unit vectors  
 $(\|\mathbf{u}\| = \|\mathbf{v}\| = 1)$

Equality in Cauchy-Schwarz if and only  
if  $\mathbf{u} = \mathbf{v}$  or  $\mathbf{u} = -\mathbf{v}$ .



# Jensen's Inequality

## Jensen's Inequality

Let  $f$  be convex,  $\mathbf{x}_1, \dots, \mathbf{x}_m \in \text{dom}(f)$ ,  $\lambda_1, \dots, \lambda_m \in \mathbb{R}_+$  such that  $\sum_{i=1}^m \lambda_i = 1$ . Then:

$$f\left(\sum_{i=1}^m \lambda_i \mathbf{x}_i\right) \leq \sum_{i=1}^m \lambda_i f(\mathbf{x}_i).$$

For  $m = 2$  this is convexity.

## Lemma

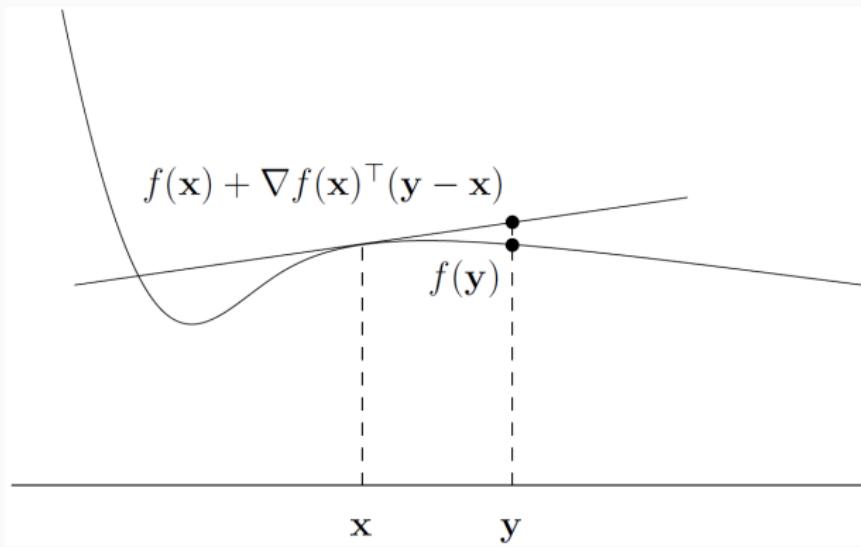
Let  $f$  be convex and suppose that  $\text{dom}(f)$  is open. Then  $f$  is continuous.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Differentiable Functions

Graph of the affine function  $f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x})$  is a **tangent hyperplane** to the graph of  $f$  at  $(\mathbf{x}, f(\mathbf{x}))$ .



# First-order Characterization of Convexity

## Lemma

Suppose that  $\text{dom}(f)$  is open and that  $f$  is differentiable; in particular, the **gradient** (vector of partial derivatives)

$$\nabla f(\mathbf{x}) := \left( \frac{\partial f}{\partial x_1}(\mathbf{x}), \dots, \frac{\partial f}{\partial x_d}(\mathbf{x}) \right)$$

exists at every point  $\mathbf{x} \in \text{dom}(f)$ . Then  $f$  is convex if and only if  $\text{dom}(f)$  is convex and

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x})$$

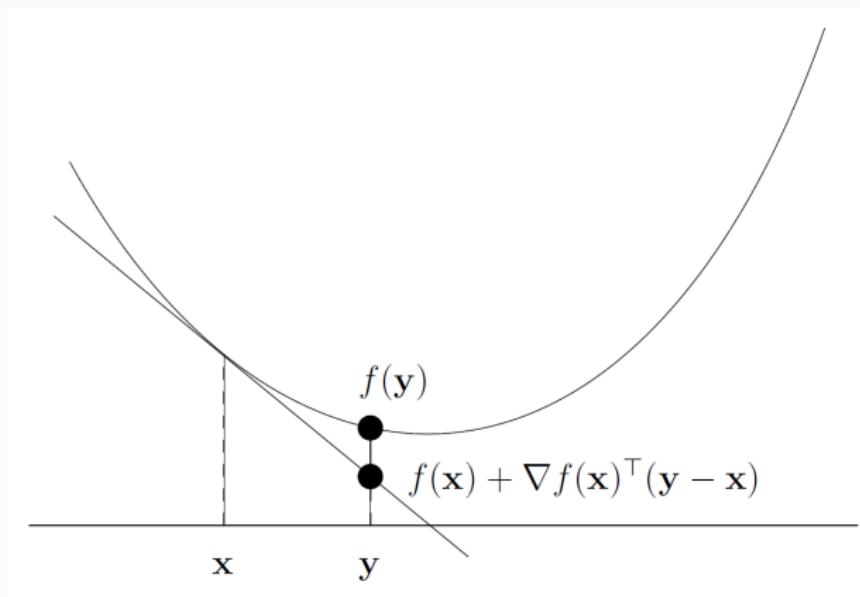
holds for all  $\mathbf{x}, \mathbf{y} \in \text{dom}(f)$ .



# First-order Characterization of Convexity

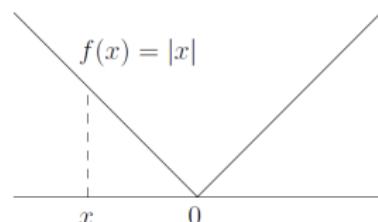
$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}), \quad \mathbf{x}, \mathbf{y} \in \text{dom}(f).$$

Graph of  $f$  is above all its tangent hyperplanes.

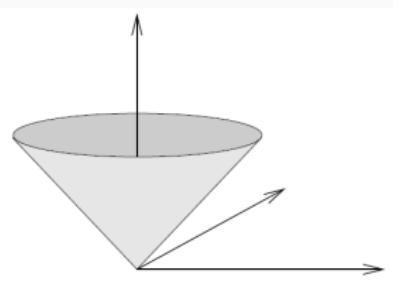


# Non-differentiable Functions

Non-differentiable functions are also relevant in practice:



More generally,  $f(\mathbf{x}) = \|\mathbf{x}\|$  (Euclidean norm). For  $d = 2$ , graph is the ice cream cone:



# Second-order Characterization of Convexity

## Lemma

Suppose that  $\text{dom}(f)$  is open and that  $f$  is twice differentiable; in particular, the **Hessian** (matrix of second partial derivatives)

$$\nabla^2 f(\mathbf{x}) := \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(\mathbf{x}) & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_d}(\mathbf{x}) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_2 \partial x_2}(\mathbf{x}) & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_d}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_d \partial x_2}(\mathbf{x}) & \dots & \frac{\partial^2 f}{\partial x_d \partial x_d}(\mathbf{x}) \end{bmatrix}$$

exists at every point  $\mathbf{x} \in \text{dom}(f)$  and is symmetric. Then  $f$  is convex if and only if  $\text{dom}(f)$  is convex, and for all  $\mathbf{x} \in \text{dom}(f)$ , we have

$$\nabla^2 f(\mathbf{x}) \succeq 0$$

i.e.,  $\nabla^2 f(\mathbf{x})$  is positive semidefinite.



# Second-order Characterization of Convexity

## Definition

A symmetric matrix  $M$  is positive semidefinite if  $\mathbf{x}^\top M \mathbf{x} \geq 0$  for all  $\mathbf{x}$ , and positive definite if  $\mathbf{x}^\top M \mathbf{x} > 0$  for all  $\mathbf{x} \neq 0$ .

## Example

Example:  $f(x_1, x_2) = x_1^2 + x_2^2$ .

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \succ 0.$$

## Theorem

- A symmetric matrix  $M$  is **positive semidefinite** if and only if the determinant of each principal submatrix (not only North-West) is non negative and the determinant of  $M$  is null.
- A symmetric matrix  $M$  is **positive definite** if and only if the determinant of all the North-West submatrices are  $> 0$ .

# Operations that Preserve Convexity

- Let  $f_1, \dots, f_m$  be convex functions,  $\lambda_1, \dots, \lambda_m \in \mathbb{R}_+$ . Then,  
 $f := \sum_{i=1}^m \lambda_i f_i$  is convex on  $\text{dom}(f) := \cap_{i=1}^m \text{dom}(f_i)$ .
- Let  $f$  be a convex function with  $\text{dom}(f) \subseteq \mathbb{R}^d$ ,  $g : \mathbb{R}^m \rightarrow \mathbb{R}^d$  an affine function, meaning that  $g(\mathbf{x}) = A\mathbf{x} + \mathbf{b}$ , for some matrix  $A \in \mathbb{R}^{d \times m}$  and some vector  $\mathbf{b} \in \mathbb{R}^d$ . Then the function  $f \circ g$  (that maps  $\mathbf{x}$  to  $f(A\mathbf{x} + \mathbf{b})$ ) is convex on  $\text{dom}(f \circ g) := \{\mathbf{x} \in \mathbb{R}^m : g(\mathbf{x}) \in \text{dom}(f)\}$ .



# Local Minima are Global Minima

## Lemma

Let  $\mathbf{x}^*$  be a local minimum of a convex function  $f : \text{dom}(f) \rightarrow \mathbb{R}$ . Then  $\mathbf{x}^*$  is a global minimum, meaning that  $f(\mathbf{x}^*) \leq f(\mathbf{y}) \forall \mathbf{y} \in \text{dom}(f)$ .

## Proof

Suppose there exists  $\mathbf{y} \in \text{dom}(f)$  such that  $f(\mathbf{y}) < f(\mathbf{x}^*)$ .

Define  $\mathbf{y}' := \lambda\mathbf{x}^* + (1 - \lambda)\mathbf{y}$  for  $\lambda \in (0, 1)$ . From convexity, we get that that  $f(\mathbf{y}') < f(\mathbf{x}^*)$ . Choosing  $\lambda$  so close to 1 that  $\|\mathbf{y}' - \mathbf{x}^*\| < \varepsilon$  yields a contradiction to  $\mathbf{x}^*$  being a local minimum.

## Lemma (Critical Points are Global Minima)

Suppose that  $f$  is convex and differentiable over an open domain  $\text{dom}(f)$ .

Let  $\mathbf{x} \in \text{dom}(f)$ . If  $\nabla f(\mathbf{x}) = 0$  (**critical point**), then  $\mathbf{x}$  is a global minimum.



# Strictly Convex Functions

## Definition

A function  $f : \text{dom}(f) \rightarrow \mathbb{R}$  is strictly convex if

- $\text{dom}(f)$  is convex;
- for all  $\mathbf{x} \neq \mathbf{y} \in \text{dom}(f)$  and all  $\lambda \in (0, 1)$ , we have:

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) < \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y}).$$



convex, but not strictly convex



strictly convex

## Lemma

Let  $f : \text{dom}(f) \rightarrow \mathbb{R}$  be strictly convex. Then  $f$  has at most one global minimum.

# Constrained Minimization

## Definition

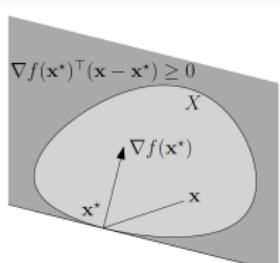
Let  $f : \text{dom}(f) \rightarrow \mathbb{R}$  be convex and let  $X \subseteq \text{dom}(f)$  be a convex set. A point  $\mathbf{x} \in X$  is a **minimizer** of  $f$  over  $X$  if

$$f(\mathbf{x}) \leq f(\mathbf{y}) \quad \forall \mathbf{y} \in X.$$

## Lemma

Suppose that  $f : \text{dom}(f) \rightarrow \mathbb{R}$  is convex and differentiable over an open domain  $\text{dom}(f) \subseteq \mathbb{R}^d$ , and let  $X \subseteq \text{dom}(f)$  be a convex set. Point  $\mathbf{x}^* \in X$  is a minimizer of  $f$  over  $X$  if and only if

$$\nabla f(\mathbf{x}^*)^\top (\mathbf{x} - \mathbf{x}^*) \geq 0 \quad \forall \mathbf{x} \in X.$$



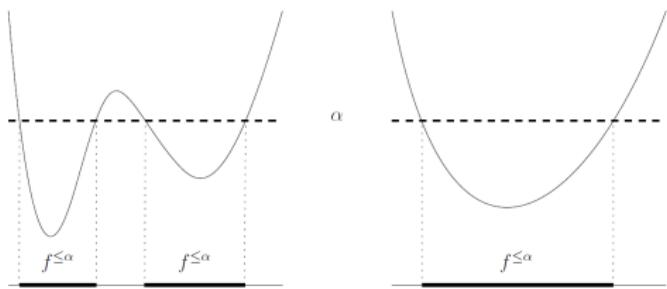
# Existence of a Minimizer

How do we know that a global minimum exists?

Not necessarily the case, even if  $f$  bounded from below ( $f(x) = e^x$ ).

## Definition

$f : \mathbb{R}^d \rightarrow \mathbb{R}, \alpha \in \mathbb{R}$ . The set  $f^{\leq \alpha} := \{\mathbf{x} \in \mathbb{R}^d : f(\mathbf{x}) \leq \alpha\}$  is the  $\alpha$ -sublevel set of  $f$ .



# The Weierstrass Theorem

## Theorem

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a convex function, and suppose there is a nonempty and bounded sublevel set  $f^{\leq \alpha}$ . Then  $f$  has a global minimum.

## Proof

We know that  $f$  (as a continuous function) attains a minimum over the closed and bounded (= compact) set  $f^{\leq \alpha}$  at some  $\mathbf{x}^*$ . This  $\mathbf{x}^*$  is also a global minimum as it has value  $f(\mathbf{x}^*) \leq \alpha$  while any  $\mathbf{x} \notin f^{\leq \alpha}$  has value  $f(\mathbf{x}) > \alpha \geq f(\mathbf{x}^*)$ .



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

- A **Quadratic Programming (QP)** problem, is a mathematical programming problem of the kind:

$$\begin{aligned} \min \quad & \frac{1}{2} \mathbf{x}^T Q \mathbf{x} + \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad & A \mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{QP}$$

where  $Q \in \mathbb{R}^{n \times n}$ .

## Remark

*The objective function of the problem is a convex function of  $\mathbf{x}$  when  $Q$  is a positive semidefinite matrix. When this condition is satisfied, the QP problem can be solved in polynomial time.*



# Example of a Quadratic programming problem in Finance

## Portfolio Selection and Asset Allocation

- Consider an investor who has a certain amount of money to be invested in a number of different securities (stocks, bonds, etc.) with random returns.
- For each security  $i = 1, \dots, n$ , estimates of its **expected return**  $\mu_i$  and **variance**  $\sigma_i^2$  are given.
- Furthermore, for any two securities  $i$  and  $j$ , their **correlation** coefficient  $\rho_{ij}$  is also assumed to be known.
- If we represent the **proportion** of the total funds **invested in security  $i$**  by  $x_i$ , one can compute the expected return and the variance of the resulting portfolio  $\mathbf{x} = (x_1, \dots, x_n)$  as follows:

$$\mathbb{E}[\mathbf{x}] = x_1\mu_1 + x_2\mu_2 + \dots + x_n\mu_n = \boldsymbol{\mu}^\top \mathbf{x}, \quad \text{expected return}$$

and

$$Var[\mathbf{x}] = \sum_{i,j} \rho_{ij}\sigma_i\sigma_j x_i x_j = \mathbf{x}^\top Q \mathbf{x}, \quad \text{variance}$$

where  $\rho_{ii} \equiv 1$ ,  $Q_{ij} = \rho_{ij}\sigma_i\sigma_j$  and  $\boldsymbol{\mu} = (\mu_1, \dots, \mu_n)$ .

# Example of a Quadratic programming problem in Finance

## Portfolio Selection and Asset Allocation

- **Markowitz' Portfolio Optimization** problem (also called the **Mean-Variance Optimization (MVO)** problem) can be formulated via the following convex QP problem:

$$\begin{aligned} \min \quad & \mathbf{x}^\top Q \mathbf{x} \\ s.t. \quad & \sum_i x_i = 1 \\ & \mu^\top \mathbf{x} \geq R \\ & \mathbf{x} \geq 0 \end{aligned} \tag{MVO}$$

- The objective function corresponds to the **total variance of the portfolio**.
- The first constraint indicates that the proportions  $x_i$  **should sum to 1**.
- The second constraint indicates that the expected return is no less than the **target value  $R$** .
- Nonnegativity constraints on  $x_i$  are introduced to rule out **short sales**<sup>1</sup>.

<sup>1</sup> selling a security that you do not have

# Karush-Kuhn-Tucker conditions (KKT conditions)

Consider the NLP:

$$\max f(\mathbf{x})$$

$$s.t. \quad g_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, m \quad (\text{NLP})$$

$$\mathbf{x} \geq \mathbf{0}$$

- How to recognize an optimal solution for a nonlinear programming problem (with differentiable functions). What are the necessary and (perhaps) sufficient conditions that such a solution must satisfy? See the **Karush-Kuhn-Tucker conditions** (below we assume a maximization problem).

Problem	Necessary Conditions for Optimality	Also Sufficient If:
One-variable unconstrained	$\frac{df}{dx} = 0$	$f(x)$ concave
Multivariable unconstrained	$\frac{\partial f}{\partial x_j} = 0 \quad (j = 1, 2, \dots, n)$	$f(\mathbf{x})$ concave
Constrained, nonnegativity constraints only	$\frac{\partial f}{\partial x_j} = 0 \quad (j = 1, 2, \dots, n)$ (or $\leq 0$ if $x_j = 0$ )	$f(\mathbf{x})$ concave
General constrained problem	Karush-Kuhn-Tucker conditions	$f(\mathbf{x})$ concave and $g_i(\mathbf{x})$ convex ( $i = 1, 2, \dots, m$ )

# Karush-Kuhn-Tucker conditions (KKT conditions)

## Theorem

If the vectors  $\nabla g_j(\mathbf{x}_0)$  for all  $j \in I(\mathbf{x}_0) = \{j | g_j(\mathbf{x}_0) = 0\}$  are linearly independent, then the constraints of the problem NLP are qualified.

## Theorem

Let  $\mathbf{x}_0$  be a local solution of the constrained problem NLP. If the active constraints in  $\mathbf{x}_0$  are qualified, there exists a vector  $\mathbf{u}^0 \in \mathbb{R}^m$  such that  $\mathbf{u}^0$  and  $\mathbf{x}_0$  are solution of the following system (KKT conditions):

$$\frac{\partial f(\mathbf{x}_0)}{\partial x_j} - \sum_{i=1}^m u_i^0 \frac{\partial g_i(\mathbf{x}_0)}{\partial x_j} = 0, \quad i = 1, \dots, n \quad (1)$$

$$g_j(\mathbf{x}_0) \leq 0 \quad j = 1, \dots, m \quad (2)$$

$$u_j^0 \geq 0 \quad j = 1, \dots, m \quad (3)$$

$$u_j^0 g_j(\mathbf{x}_0) = 0 \quad j = 1, \dots, m \quad (4)$$

- The  $u_i$  correspond to the dual variables of linear programming and they have a comparable economic interpretation. However, they arose in the mathematical derivation as Lagrange multipliers.



## Corollary

Assume that  $f(\mathbf{x})$  is a concave function and that  $g_1(\mathbf{x}), g_2(\mathbf{x}), \dots, g_m(\mathbf{x})$  are convex functions (i.e., this problem is a **convex programming problem**), where all these functions satisfy the regularity conditions. Then  $\mathbf{x}^0$  is an optimal solution **if and only if** all the conditions of the previous theorem are satisfied.

- From a practical computational viewpoint, the method of Lagrange multipliers and consequently KKT are not a particularly powerful procedure. It is often essentially impossible to solve the equations to obtain the critical points. Furthermore, even when the points can be obtained, the number of critical points may be so large (often infinite) that it is impractical to attempt to identify a global minimum or maximum. However, for certain types of small problems, this method can sometimes be used successfully.



# References

- Sébastien Bubeck. **Convex Optimization: Algorithms and Complexity.**  
Foundations and Trends in Machine Learning, 8(3-4):231 - 357, 2015.
- Stephen Boyd and Lieven Vandenberghe. **Convex Optimization.**  
Cambridge University Press, New York, NY, USA, 2004.

# OPTIMIZATION

---

*Francesca Maggioni*

Department of Management, Information and Production Engineering  
University of Bergamo

Lesson 2 versus the GD

Academic Year 2023-2024



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

---

*"If you optimize everything, you will always be unhappy."*

Donald Knuth

---



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Example of convex function minimization in machine learning

## Least squares

- Suppose we want to fit a hyperplane to a set of data points  $\mathbf{x}_1, \dots, \mathbf{x}_m$  in  $\mathbb{R}^d$ , based on the hypothesis that the points actually come (approximately) from a hyperplane. A classical method for this is least squares.
- Suppose that the data points are

$$(1, 10), (2, 11), (3, 11), (4, 10), (5, 9), (6, 10), (7, 9), (8, 10) \in \mathbb{R}^2$$

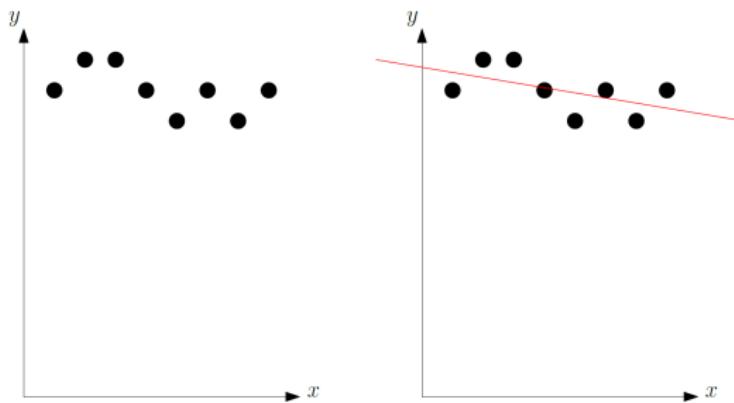


Figure 1: Data points in  $\mathbb{R}^2$  (left) and least-squares fit (right)

# Least squares

- For simplicity let us restrict to fitting a linear model of the form  
 $y = w_0 + w_1 x$ .
- If  $(x_i, y_i)$  is the  $i$ -th data point, the least squares fit chooses  $w_0, w_1$  s.t.  
the least squares objective

$$f(w_0, w_1) = \sum_{i=1}^8 (w_1 x_i + w_0 - y_i)^2$$

is minimized.

- $f(w_0, w_1) = 204w_1^2 + 72w_1w_0 - 706w_1 + 8w_0^2 - 160w_0 + 804$  which is **convex**.

- We can check the convexity using second order conditions:

**Gradient:**  $\nabla f(w_0, w_1) = (72w_1 + 16w_0 - 160, 408w_1 + 72w_0 - 760)$

**Hessian:**

$$\nabla^2 f(w_0, w_1) = \begin{bmatrix} 16 & 72 \\ 72 & 408 \end{bmatrix}$$

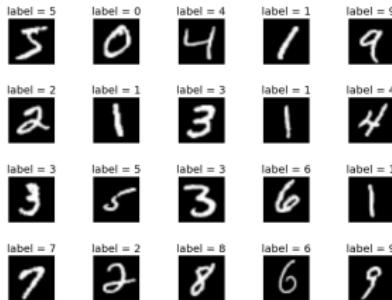
which is positive definite and  $f$  **strictly convex**.

- Putting  $\nabla f(w_0, w_1) = (0, 0)$  we obtain  $(w_0^*, w_1^*) = (43/4, -1/6)$ .  
The optimal line is  $y = -1/6x + 43/4$ .

# Example of convex function minimization in machine learning

## Handwritten digit recognition

- Suppose you want to write a program that **recognizes handwritten decimal digits**  $0, 1, \dots, 9$ .
- You have a set  $P$  of **grayscale images** ( $28 \times 28$  pixels, say) that represent handwritten decimal digits, and for each image  $x \in P$  you know the digit  $d(x) \in \{0, \dots, 9\}$  that it represents, (see Figure 2).
- You want to train your program with the set  $P$ , and after that, use it to **recognize handwritten digits** in arbitrary  $28 \times 28$  images.



**Figure 2:** Some training images from the MNIST data set (picture from <http://corochann.com/mnist-dataset-introduction-1138.html>)

# Handwritten digit recognition

- We represent an **image** as a **feature vector**  $\mathbf{x} \in \mathbb{R}^{784}$ , where  $x_i$  is the gray value of the  $i - th$  pixel (in some order).
- During the training phase, we compute a **matrix**  $W \in R^{10 \times 784}$  and then use the vector  $\mathbf{y} = W\mathbf{x} \in \mathbb{R}^{10}$  to predict the digit seen in an arbitrary image  $\mathbf{x}$ .
- The idea is that  $y_j$ ,  $j = 0, \dots, 9$  corresponds to the **probability of the digit being  $j$** . This does not work directly, since the entries of  $\mathbf{y}$  may be negative and generally do not sum up to 1.
- But we can convert  $\mathbf{y}$  to a vector  $\mathbf{z}$  of **actual probabilities**, such that a small  $y_j$  leads to a small probability  $z_j$  and a large  $y_j$  to a large probability  $z_j$ . A well-known formula that works is:

$$z_j = z_j(\mathbf{y}) = \frac{e^{y_j}}{\sum_{k=0}^9 e^{y_k}}.$$

The classification then simply outputs digit  $j$  with probability  $z_j$ .



# Handwritten digit recognition

- The matrix  $W$  is chosen such that it (approximately) **minimizes the classification error** on the training set  $P$ .
- We use the following **loss function** to evaluate the error induced by a given matrix  $W$ :

$$\ell(W) = - \sum_{\mathbf{x} \in P} \ln(z_{d(\mathbf{x})}(W\mathbf{x})) = \sum_{\mathbf{x} \in P} \left( \ln \left( \sum_{k=0}^9 e^{(W\mathbf{x})_k} \right) - (W\mathbf{x})_{d(\mathbf{x})} \right)$$

- This function "punishes" images for which the correct digit  $j$  has **low probability**  $z_j$  (corresponding to a significantly negative value of  $\log(z_j)$  ). In an ideal world, the correct digit would always have probability 1, resulting in  $\ell(W) = 0$ . But under  $z_j$  definition, probabilities are always strictly between 0 and 1, so we have  $\ell(W) > 0$  for all  $W$ .
- $\ell(W)$  is a **convex function**.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Why We Need Gradient Descent

- Solving the equation  $f'(x) = 0$  for  $x$  provides an **analytical solution for a critical point**.
- Unfortunately, it is not always possible to compute such analytical solutions in closed form.
- It is often difficult to exactly solve the equation  $f'(x) = 0$  because this derivative might itself be a complex function of  $x$ .
- In other words, **a closed form solution typically does not exist**.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Why We Need Gradient Descent

## Example

For example, consider the following function that needs to be minimized:

$$f(x) = x^2 \cdot \ln(x) - x \quad (1)$$

Setting the first derivative of this function to 0 yields the following condition:

$$f'(x) = 2x \cdot \ln(x) + x - 1 = 0.$$

This equation is hard to solve. By trial and error, one might get lucky and find out that  $x = 1$  is indeed a solution to the first-order optimality condition because it satisfies  $f'(1) = 2\ln(1) + 1 - 1 = 0$ .

Furthermore, the second derivative  $f''(x)$  can be shown to be positive at  $x = 1$ , so this point is at least a local minimum. However, solving an equation like this numerically causes all types of numerical and computational challenges; these types of challenges increase for multivariate optimization.



# Why We Need Gradient Descent

- A very popular approach for optimizing objective functions is to use the method of **gradient descent**.
- In gradient descent, one starts at an **initial point**  $x = x_0$  and successively updates  $x$  using the **steepest descent direction**.
- In the univariate case, the notion of *steepest* is hard to appreciate, as there are only two directions of movement (*i.e.*, increase  $x$  or decrease  $x$ ). One of these directions causes ascent, whereas the other causes descent.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Why We Need Gradient Descent

## Gradient Method (Cauchy, 1847)

Let  $x_0 \in \mathbb{R}$ . For  $t = 0, 1, \dots$

$$x_{t+1} := x_t - \gamma f'(x_t),$$

with  $\gamma \geq 0$  learning rate.

---

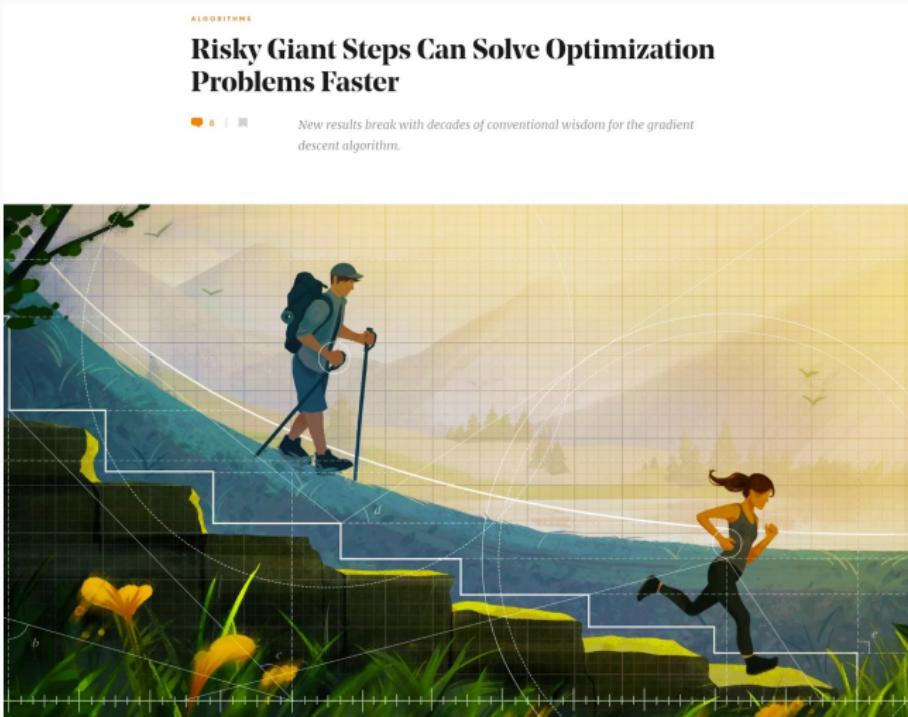
<sup>0</sup>Developed trying to solve the problem of determining the orbit of a celestial body from its equations of motion.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Why We Need Gradient Descent

# Fresh Printing Results



ALGORITHMS

## Risky Giant Steps Can Solve Optimization Problems Faster



New results break with decades of conventional wisdom for the gradient descent algorithm.

<sup>1</sup> <https://www.quantamagazine.org/risky-giant-steps-can-solve-optimization-problems-faster-20230811/>  
Benjamin Grimmer (2023) Provably Faster Gradient Descent via Long Steps, arXiv.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Why We Need Gradient Descent

Summarizing:

- In **gradient descent** one starts at an **initial point**  $x = x_0$  and successively updates  $x$  using the **steepest descent direction**:

$$x \leftarrow x - \alpha f'(x).$$

- Here,  $\alpha > 0$  regulates the **step size** (or **learning rate**).



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Why We Need Gradient Descent

- However, in **multivariate problems**, there can be an **infinite number of possible directions** of descent, and the generalization of the notion of univariate derivative leads to the steepest descent direction.
- The value of  $x$  changes in each iteration by  $\delta x = -\alpha f'(x)$ .
- Note that at infinitesimally small values of the learning rate  $\alpha > 0$ , the above updates will always reduce  $f(x)$ .
- This is because for **very small  $\alpha$** , we can use the **first-order Taylor expansion** to obtain the following:

$$f(x + \delta x) \approx f(x) + \delta x f'(x) = f(x) - \alpha [f'(x)]^2 < f(x). \quad (2)$$



# Why We Need Gradient Descent

- Using very **small values of  $\alpha > 0$**  is not advisable because it will take a **long time** for the algorithm to converge.
- On the other hand, using **large values of  $\alpha > 0$**  could make the effect of the update **unpredictable** with respect to the computed gradient (as the **first-order Taylor expansion is no longer a good approximation**).
- After all, the gradient is only an instantaneous rate of change, and it does not apply over larger ranges.
- Therefore, large step-sizes could cause the solution to overshoot an optimal value, if the sign of the gradient changes over the length of the step.
- At extremely large values of the learning rate, it is even possible for the solution to diverge, where it moves at an increasing speed towards large absolute values, and typically terminates with a numerical overflow.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

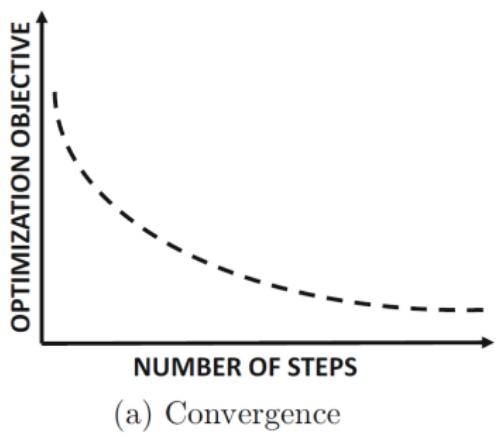
# Convergence of Gradient Descent

- The execution of gradient-descent updates will generally result in a **sequence of values**  $x_0, x_1, \dots, x_t$  of the optimization variable, which become **successively closer to an optimum solution**.
- As the value of  $x_t$  nears the optimum value, the derivative  $f'(x_t)$  also tends to be closer and closer to zero (thereby satisfying the first-order optimality conditions).
- In other words, **the absolute step size will tend to reduce over the execution of the algorithm**.
- As gradient descent nears an optimal solution, the objective function will also improve at a slower rate.
- This observation provides some natural ideas on making decisions regarding the **termination of the algorithm** (when the current solution is sufficiently close to an optimal value).



# Convergence of Gradient Descent

- The idea is to plot the current value of  $f(x_t)$  with iteration index  $t$  as the algorithm progresses.
- An example of good **progress during gradient descent** is shown hereby:



# Convergence of Gradient Descent

- The **objective function value need not be monotonically decreasing** over the course of the algorithm, but it will tend to show **small noisy changes** (without significant long-term direction) after some point.
- This situation can be treated as a good termination point for the algorithm.
- However, in some cases, the update steps can be shown to diverge from an optimal solution, if the step size is not chosen properly.



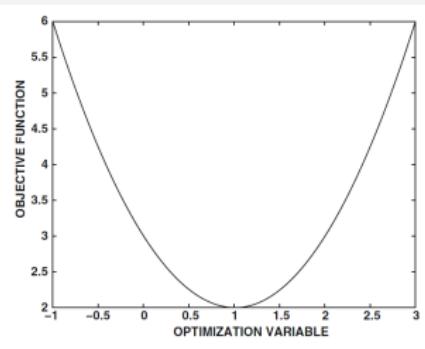
UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# The Divergence Problem

- Choosing a **very large learning rate  $\alpha$**  can cause overshooting from the optimal solution, and even **divergence**. Consider the following example.

## Example

Consider the quadratic function  $f(x)$  in the following figure, which takes on its optimal value at  $x = 1$ :



(a) Single global minimum  
 $f(x) = x^2 - 2x + 3$

# The Divergence Problem

## Example

- Now imagine a situation where the starting point is  $x_0 = 2$ , and one chooses a large learning rate  $\alpha = 10$ .
- The derivative of  $f(x) = 2x - 2$  evaluates to  $f'(x_0) = f'(2) = 2$ .
- Then, the update from the first step yields the following:

$$x_1 \leftarrow x_0 - 10 \cdot 2 = 2 - 20 = -18.$$

- Note that the new point  $x_1$  is much further away from the optimal value of  $x = 1$ , which is caused by the overshooting problem.
- Even worse, the absolute gradient is very large at this point, and it evaluates to  $f'(-18) = -38$ . If we keep the learning rate fixed, it will cause the solution to move at an even faster rate in the opposite direction:

$$x_2 \leftarrow x_1 - 10 \cdot (-38) = -18 + 380 = 362.$$



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# The Divergence Problem

## Example

- In this case, the solution has overshot back in the original direction but is even further away from the optimal solution.
- Further updates cause back-and-forth movements at increasingly large amplitudes:

$$x_3 \Leftarrow x_2 - 10 \cdot 722 = 362 - 7220 = -6858, \quad x_4 \Leftarrow x_3 + 10 \cdot 13718 = 130322.$$

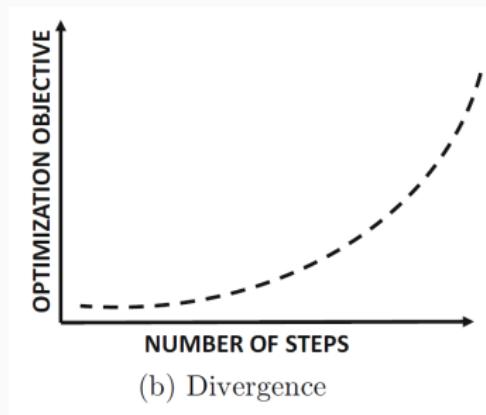
- Note that each iteration flips the sign of the current solution and increases its magnitude by a factor of about 20.
- In other words, the solution moves away faster and faster from an optimal solution until it leads to a numerical overflow.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# The Divergence Problem

- An example of the **behavior of the objective function during divergence** is shown hereby:



# The Divergence Problem

- It is common to **reduce the learning rate over the course of the algorithm**, and one reason to do that is to arrest divergence.
- However, in some cases, such an approach might not prevent divergence, especially if the initial learning rate is large.
- Therefore, when an analyst encounters a situation in gradient descent, where the size of the parameter vector seems to increase rapidly (and the optimization objective worsens), it is a tell-tale sign of divergence.
- The first adjustment should be to **experiment with a lower initial learning rate**.
- However, choosing a learning rate that is too small might lead to unnecessarily slow progress, causing the procedure to take too much time.
- There is a **considerable literature in finding the correct step size or adjusting it over the course of the algorithm**.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Gradient Descent: The Algorithm

Get near to a minimum  $\mathbf{x}^*$  / close to the optimal value  $f(\mathbf{x}^*)$ ?

*Assumption:*  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  convex, differentiable, has a global minimum  $\mathbf{x}^*$ .

## Goal

Find  $\mathbf{x}^* \in \mathbb{R}^d$  such that

$$f(\mathbf{x}) - f(\mathbf{x}^*) \leq \varepsilon.$$

Note that there can be several global minima  $\mathbf{x}_1^* \neq \mathbf{x}_2^*$  with  $f(\mathbf{x}_1^*) = f(\mathbf{x}_2^*)$ .

## Iterative Algorithm

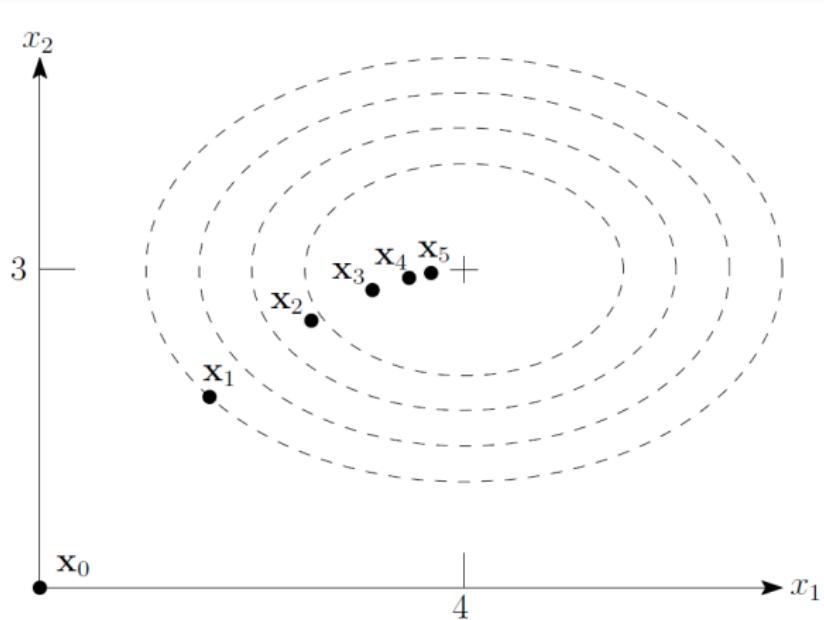
Choose  $\mathbf{x}_0 \in \mathbb{R}^d$ .

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma \nabla f(\mathbf{x}_t),$$

for timesteps  $t = 0, 1, \dots$ , and stepsize  $\gamma \geq 0$ .



## Example



$$f(x_1, x_2) := 2(x_1 - 4)^2 + 3(x_2 - 3)^2, \mathbf{x}_0 := (0, 0), \gamma := 0.1$$



# Vanilla Analysis

How to bound  $f(\mathbf{x}_t) - f(\mathbf{x}^*)$ ?

- Abbreviate  $\mathbf{g}_t := \nabla f(\mathbf{x}_t)$  (gradient descent:  $\mathbf{g}_t = (\mathbf{x}_t - \mathbf{x}_{t+1})/\gamma$ ).

$$\mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*) = \frac{1}{\gamma} (\mathbf{x}_t - \mathbf{x}_{t+1})^\top (\mathbf{x}_t - \mathbf{x}^*)$$

- Apply  $2\mathbf{v}^\top \mathbf{w} = \|\mathbf{v}\|^2 + \|\mathbf{w}\|^2 - \|\mathbf{v} - \mathbf{w}\|^2$  to re-write:

$$\begin{aligned}\mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*) &= \frac{1}{2\gamma} \left( \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 + \|\mathbf{x}_t - \mathbf{x}^*\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 \right) \\ &= \frac{\gamma}{2} \|\mathbf{g}_t\|^2 + \frac{1}{2\gamma} \left( \|\mathbf{x}_t - \mathbf{x}^*\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 \right).\end{aligned}$$

- Sum this up over the first  $T$  iterations:

$$\sum_{t=0}^{T-1} \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*) = \frac{\gamma}{2} \sum_{t=0}^{T-1} \|\mathbf{g}_t\|^2 + \frac{1}{2\gamma} \left( \|\mathbf{x}_0 - \mathbf{x}^*\|^2 - \|\mathbf{x}_T - \mathbf{x}^*\|^2 \right).$$



# Vanilla Analysis

Use first-order characterization of convexity:  $f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x})$  for all  $\mathbf{x}, \mathbf{y}$ .

- with  $\mathbf{x} = \mathbf{x}_t$ ,  $\mathbf{y} = \mathbf{x}^*$ :

$$f(\mathbf{x}_t) - f(\mathbf{x}^*) \leq \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*)$$

giving

$$\sum_{t=0}^{T-1} \left( f(\mathbf{x}_t) - f(\mathbf{x}^*) \right) \leq \frac{\gamma}{2} \sum_{t=0}^{T-1} \|\mathbf{g}_t\|^2 + \frac{1}{2\gamma} \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

an upper bound for the **average error**  $f(\mathbf{x}_t) - f(\mathbf{x}^*)$  over the steps.

- Last iterate is not necessarily the best one.
- Stepsize is crucial.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Optimization

Francesca Maggioni

---

Academic Year 2023-2024

Department of Management, Information and Production Engineering  
University of Bergamo



# Gradient Descent: The Algorithm

Get near to a minimum  $\mathbf{x}^*$  / close to the optimal value  $f(\mathbf{x}^*)$ ?

*Assumption:*  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  convex, differentiable, has a global minimum  $\mathbf{x}^*$ .

## Goal

Find  $\mathbf{x}^* \in \mathbb{R}^d$  such that

$$f(\mathbf{x}) - f(\mathbf{x}^*) \leq \varepsilon.$$

Note that there can be several global minima  $\mathbf{x}_1^* \neq \mathbf{x}_2^*$  with  $f(\mathbf{x}_1^*) = f(\mathbf{x}_2^*)$ .

## Iterative Algorithm

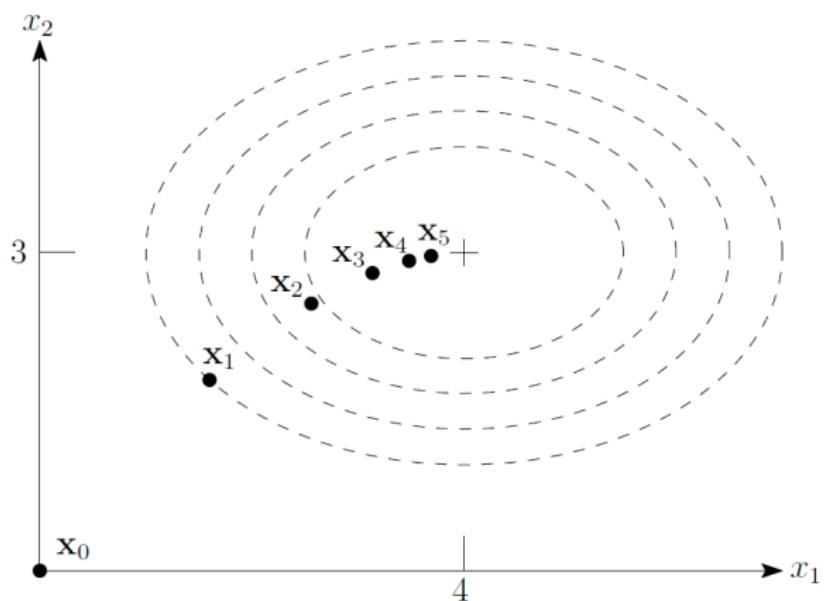
Choose  $\mathbf{x}_0 \in \mathbb{R}^d$ .

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma \nabla f(\mathbf{x}_t),$$

for timesteps  $t = 0, 1, \dots$ , and stepsize  $\gamma \geq 0$ .



## Example



$$f(x_1, x_2) := 2(x_1 - 4)^2 + 3(x_2 - 3)^2, \mathbf{x}_0 := (0, 0), \gamma := 0.1$$

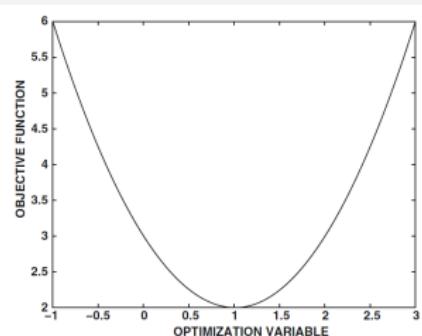


# The Divergence Problem

- Choosing a **very large learning rate  $\alpha$**  can cause overshooting from the optimal solution, and even **divergence**. Consider the following example.

## Example

Consider the quadratic function  $f(x)$  in the following figure, which takes on its optimal value at  $x = 1$ :



(a) Single global minimum  
 $f(x) = x^2 - 2x + 3$



## Example

- Now imagine a situation where the starting point is  $x_0 = 2$ , and one chooses a large learning rate  $\alpha = 10$ .
- The derivative of  $f(x) = 2x - 2$  evaluates to  $f'(x_0) = f'(2) = 2$ .
- Then, the update from the first step yields the following:

$$x_1 \leftarrow x_0 - 10 \cdot 2 = 2 - 20 = -18.$$

- Note that the new point  $x_1$  is much further away from the optimal value of  $x = 1$ , which is caused by the overshooting problem.
- Even worse, the absolute gradient is very large at this point, and it evaluates to  $f'(-18) = -38$ . If we keep the learning rate fixed, it will cause the solution to move at an even faster rate in the opposite direction:

$$x_2 \leftarrow x_1 - 10 \cdot (-38) = -18 + 380 = 362.$$



# The Divergence Problem

## Example

- In this case, the solution has overshot back in the original direction but is even further away from the optimal solution.
- Further updates cause back-and-forth movements at increasingly large amplitudes:

$$x_3 \Leftarrow x_2 - 10 \cdot 722 = 362 - 7220 = -6858, \quad x_4 \Leftarrow x_3 + 10 \cdot 13718 = 130322.$$

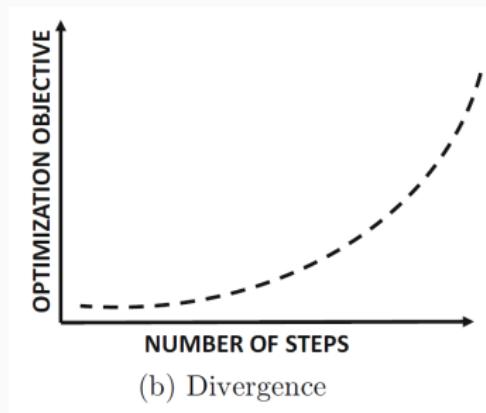
- Note that each iteration flips the sign of the current solution and increases its magnitude by a factor of about 20.
- In other words, the solution moves away faster and faster from an optimal solution until it leads to a numerical overflow.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# The Divergence Problem

- An example of the **behavior of the objective function during divergence** is shown hereby:



# The Divergence Problem

- Common approach: to arrest divergence **reduce the learning rate** over the course of the algorithm.
- However, if the **initial learning rate is large**, such an approach might not prevent divergence.
- First adjustment: to **experiment with a lower initial learning rate**.
- However, choosing a learning rate that is too small might lead to **slow progress**, causing the procedure to take too much time.
- There is a **considerable literature in finding the correct step size or adjusting it over the course of the algorithm**.



---

## The Minutiae of Gradient Descent

---

# Checking Gradient Correctness with Finite Differences

- Many **ML algorithms** use complex objective functions over **millions of parameters**.
- The gradients are computed either **analytically** and then hand-coded into the algorithm, or they are computed using **automatic differentiation methods** in applications like neural networks.
- In all these cases, **analytical or coding errors** remain a real possibility, which may or may not become obvious during execution.
- Knowing the reason for the **poor performance** of an algorithm is a critical step in deciding whether to simply debug the algorithm or to make fundamental design changes.



# Checking Gradient Correctness with Finite Differences

- Compute the gradient of the objective function  $J(\bar{w}) = J(w_1, \dots, w_d)$ .
- **Finite-difference approximation:** sample a few from  $w_1, \dots, w_d$  and check their partial derivatives.
- Basic idea: perturb  $w_i$  by a small amount  $\Delta$  and approximate the partial derivative with respect to  $w_i$  by using the difference between the perturbed value of the objective function and the original value:

$$\frac{\partial J(\bar{w})}{\partial w_i} \approx \frac{J(w_1, \dots, w_i + \Delta, \dots, w_d) - J(w_1, \dots, w_i, \dots, w_d)}{\Delta}.$$

- One would not obtain an exact value of the partial derivative in this way.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Learning Rate Decay

- Consider a **decaying learning rate**  $\alpha_t$  and the update is as follows:

$$\bar{w} \leftarrow \bar{w} - \alpha_t \nabla J.$$

- Typical decays:

$$\alpha_t = \alpha_0 \cdot \exp(-k \cdot t) \quad [\text{Exponential Decay}]$$

$$\alpha_t = \frac{\alpha_0}{1 + k \cdot t} \quad [\text{Inverse Decay}]$$

where:  $\alpha_0$  is the initial decay and  $k$  controls the rate of the decay.

- Another approach is to use **step decay** in which the learning rate is reduced by a particular factor every few steps of gradient descent.



## Bold Driver algorithm

- **Bold-driver algorithm:** popular approach for adjusting the learning rate depending on whether the objective function is **improving or worsening**.
- The learning rate is **increased** by factor of around **5%** in each iteration as long as the steps improve the objective function. As soon as the objective function worsens because of a step, the step is undone and an attempt is made again with the learning rate reduced by a factor of around **50%** (until convergence).
- BD does not work in **noisy settings** of gradient descent, where the objective function is approximated by using samples of the data (for example Stochastic Gradient Descent).
- In such cases, it is important to test the objective function and adjust the learning rate after  $m$  steps, rather than a single step.
- The change in objective function can be measured more robustly across multiple steps, and all  $m$  steps must be undone when the objective function worsens over these steps.



# Line Search

- **Line search:** uses the **optimum step size**
- Although it is **rarely used** in vanilla gradient descent (because it is computationally expensive), it is helpful in some specialized variations of gradient descent.
- Let  $J(\bar{w})$  be the objective function.
- Let  $\bar{g}_t$  be the descent direction at the beginning of the  $t$ -th step with  $\bar{w}_t$ .
- The parameter vector needs to be updated as follows:

$$\bar{w}_{t+1} \Leftarrow \bar{w}_t + \alpha_t \bar{g}_t.$$

- The step-size  $\alpha_t$  is computed as follows:

$$\alpha_t = \arg \min_{\alpha} J(\bar{w}_t + \alpha \bar{g}_t). \quad (1)$$

i.e. the learning rate  $\alpha_t$  is chosen in each step, so as to minimize the value of the objective function at  $\bar{w}_{t+1}$ .

- After performing the step, the gradient is computed at  $\bar{w}_{t+1}$  for the next step. The gradient at  $\bar{w}_{t+1}$  will be perpendicular to the search direction  $\bar{g}_t$  or else  $\alpha_t$  will not be optimal.
- Therefore, we obtain the following:

$$\bar{g}_t^\top [\nabla J(\bar{w}_t + \alpha_t \bar{g}_t)] = 0.$$

## Lemma

The gradient at the optimal point of a line search is **always orthogonal** to the current search direction.

- A natural question arises as to how the minimization of (1) is performed.
- Property of typical line-search settings: the objective function  $H(\alpha) = J(\bar{w}_t + \alpha \bar{g}_t)$ , when expressed in terms of  $\alpha$  is often a **unimodal function**.



# Line Search

- First step: to identify a range  $[0, \alpha_{\max}]$  in which to perform the search.
- This can be performed efficiently by evaluating the objective function value at geometrically increasing values of  $\alpha$  (increasing every time by a factor of 2).
- Subsequently, it is possible to use a variety of methods to narrow the interval such as the **binary-search method**, the **golden-section search method**, and the **Armijo rule**.
- The first two of these methods are **exact** methods, and they leverage the unimodality of the objective function in terms of the step-size  $\alpha$ .
- The Armijo rule is **inexact**, and it works even when  $H(\alpha) = J(\bar{w}_t + \alpha \bar{g}_t)$  is multimodal/nonconvex in  $\alpha$ .
- Therefore, the Armijo rule has broader use than exact line search methods, especially as far as simple forms of gradient descent are concerned.



# Binary Search

- We start by initializing the **binary search** interval for  $\alpha$  to  $[a, b] = [0, \alpha_{\max}]$ .
- $[a, b]$ , is narrowed by evaluating the objective function at two closely spaced points near  $(a + b)/2$ .
- We evaluate the objective function at  $(a + b)/2$  and  $(a + b)/2 + \varepsilon$ , where  $\varepsilon = 10^{-6}$ :  $H[(a + b)/2]$  and  $H[(a + b)/2 + \varepsilon]$ .
- If the function is increasing at  $(a + b)/2$ , the interval is narrowed to  $[a, (a + b)/2 + \varepsilon]$ .
- Otherwise, it is narrowed to  $[(a + b)/2, b]$ .
- This process is repeated, until an interval is reached with the required level of accuracy.



## Armijo Rule

- The basic idea behind the **Armijo rule** is that the descent direction  $\bar{g}_t$  at the starting point  $\bar{w}_t$  (*i.e.*, at  $\alpha = 0$ ) often deteriorates in terms of rate of improvement of objective function as one moves further along this direction.
- The rate of improvement of the objective function along the search direction at the starting point is  $|g_t^\top [\nabla F(\bar{w}_t)]|$ .
- Therefore, the (typical) improvement of the objective function at a particular value of  $\alpha$  can optimistically be expected to be  $\alpha |\bar{g}_t^\top [\nabla F(\bar{w}_t)]|$  for most real-world objective functions.
- The Armijo rule is satisfied with a **fraction**  $\mu \in (0, 0.5)$  of this improvement. A typical value of  $\mu$  is around 0.25.
- We want to **find the largest step-size**  $\alpha$  satisfying the following:

$$F(\bar{w}_t) - F(\bar{w}_t + \alpha \bar{g}_t) \geq \mu \alpha |\bar{g}_t^\top [\nabla F(\bar{w}_t)]|.$$



## Armijo Rule

- Note that for **small values of  $\alpha$** , the condition above will always be satisfied. In fact, one can show using the finite-difference approximation, that for infinitesimally small values of  $\alpha$ , the condition above is satisfied at  $\mu = 1$ .
- However, we want a larger step size to ensure faster progress. What is the largest step-size one can use?
- We test successively decreasing values of  $\alpha$  for the condition above, and stop the first time the condition above is satisfied.
- In backtracking line search, we start by testing  $H(\alpha_{\max}), H(\beta\alpha_{\max}), \dots, H(\beta^r\alpha_{\max})$ , until the condition above is satisfied. At that point we use  $\alpha = \beta^r\alpha_{\max}$ .
- Here,  $\beta$  is a parameter drawn from  $(0, 1)$ , and a typical value is 0.5.



# When to Use Line Search

- The line-search method, while guaranteed to converge to at least a local optimum, is considered **expensive**.
- Vanilla gradient descent rarely employs the line-search method due to its cost, but it finds application in specialized variations like Newton's method.
- Some variations of gradient descent, including vanilla gradient descent, may use fast, inexact methods like the Armijo rule instead of exact line search.
- In cases where exact line search is required, the relatively small number of steps often compensates for the expensive nature of individual steps, making it feasible.
- An important aspect of using the line-search method is that **convergence is guaranteed**, even if the resulting solution is a local optimum.



# Initialization

- The gradient-descent procedure begins at an initial point and iteratively improves the parameter vector at a specific learning rate.
- Choosing an appropriate initialization point is a crucial consideration. For simple ML problems, small random values from  $[-1, +1]$  or  $[0, 1]$  (for non-negative constraints) can be used.
- However, this straightforward initialization approach may pose challenges for more complex algorithms.
- In such cases, improper magnitudes of initial parameters may lead to numerical **overflows** or **underflows** during updates.
- **Heuristic optimization methods** for initialization, which pretrain the algorithm to an initialization near an optimum point, can be effective.



# Vanilla Analysis

How to bound  $f(\mathbf{x}_t) - f(\mathbf{x}^*)$ ?

- Abbreviate  $\mathbf{g}_t := \nabla f(\mathbf{x}_t)$  (gradient descent:  $\mathbf{g}_t = (\mathbf{x}_t - \mathbf{x}_{t+1})/\gamma$ ).

$$\mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*) = \frac{1}{\gamma} (\mathbf{x}_t - \mathbf{x}_{t+1})^\top (\mathbf{x}_t - \mathbf{x}^*)$$

- Apply  $2\mathbf{v}^\top \mathbf{w} = \|\mathbf{v}\|^2 + \|\mathbf{w}\|^2 - \|\mathbf{v} - \mathbf{w}\|^2$  to re-write:

$$\begin{aligned}\mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*) &= \frac{1}{2\gamma} \left( \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 + \|\mathbf{x}_t - \mathbf{x}^*\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 \right) \\ &= \frac{\gamma}{2} \|\mathbf{g}_t\|^2 + \frac{1}{2\gamma} \left( \|\mathbf{x}_t - \mathbf{x}^*\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 \right).\end{aligned}$$

- Sum this up over the first  $T$  iterations:

$$\sum_{t=0}^{T-1} \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*) = \frac{\gamma}{2} \sum_{t=0}^{T-1} \|\mathbf{g}_t\|^2 + \frac{1}{2\gamma} \left( \|\mathbf{x}_0 - \mathbf{x}^*\|^2 - \|\mathbf{x}_T - \mathbf{x}^*\|^2 \right).$$

# Vanilla Analysis

Use first-order characterization of convexity:  $f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x})$  for all  $\mathbf{x}, \mathbf{y}$ .

- with  $\mathbf{x} = \mathbf{x}_t$ ,  $\mathbf{y} = \mathbf{x}^*$ :

$$f(\mathbf{x}_t) - f(\mathbf{x}^*) \leq \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*)$$

giving

$$\sum_{t=0}^{T-1} (f(\mathbf{x}_t) - f(\mathbf{x}^*)) \leq \frac{\gamma}{2} \sum_{t=0}^{T-1} \|\mathbf{g}_t\|^2 + \frac{1}{2\gamma} \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

an upper bound for the **average error**  $f(\mathbf{x}_t) - f(\mathbf{x}^*)$  over the steps.

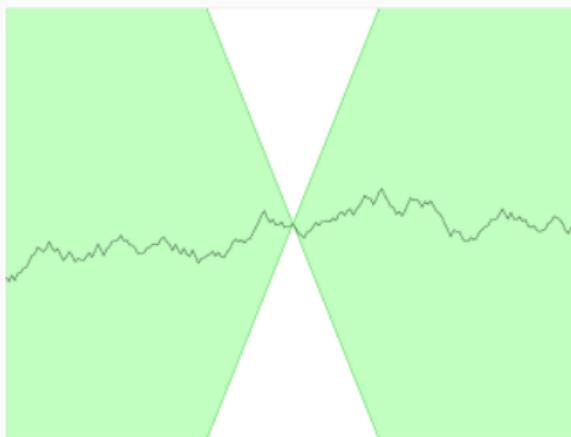
- Last iterate is not necessarily the best one.
- Stepsize is crucial.



# Lipschitz Convex Functions: $\mathcal{O}(1/\varepsilon^2)$ Steps

Assume that all gradients of  $f$  are bounded in norm.

- Equivalent to  $f$  being Lipschitz.
- Rules out many interesting functions (for example, the *supermodel*  $f(x) = x^2$ ).



The Lipschitz condition (or Lipschitz continuity) ensures that your function always remains entirely outside the white cone, so it cannot e.g. become infinitely steep at one point.

## Theorem

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be convex and differentiable with a global minimum  $\mathbf{x}^*$ ; furthermore, suppose that  $\|\mathbf{x}_0 - \mathbf{x}^*\| \leq R$  and  $\|\nabla f(\mathbf{x})\| \leq B$  for all  $\mathbf{x}$ .

Choosing the stepsize:

$$\gamma := \frac{R}{B\sqrt{T}}$$

gradient descent yields

$$\frac{1}{T} \sum_{t=0}^{T-1} f(\mathbf{x}_t) - f(\mathbf{x}^*) \leq \frac{RB}{\sqrt{T}}.$$



# Lipschitz Convex Functions: $\mathcal{O}(1/\varepsilon^2)$ Steps

## Proof

- Plug  $\|\mathbf{x}_0 - \mathbf{x}^*\| \leq R$  and  $\|\nabla f(\mathbf{x})\| \leq B$  into Vanilla Analysis:

$$\sum_{t=0}^{T-1} (f(\mathbf{x}_t) - f(\mathbf{x}^*)) \leq \frac{\gamma}{2} \sum_{t=0}^{T-1} \|\mathbf{g}_t\|^2 + \frac{1}{2\gamma} \|\mathbf{x}_0 - \mathbf{x}^*\|^2 \leq \frac{\gamma}{2} B^2 T + \frac{1}{2\gamma} R^2.$$

- Choose  $\gamma$  such that

$$q(\gamma) = \frac{\gamma}{2} B^2 T + \frac{R^2}{2\gamma}$$

is minimized.

- Solving  $q'(\gamma) = 0$  yields the minimum  $\gamma = \frac{R}{B\sqrt{T}}$  and  $q\left(\frac{R}{B\sqrt{T}}\right) = RB\sqrt{T}$ .
- Dividing by  $T$ , the result follows.



$$T \geq \frac{R^2 B^2}{\varepsilon^2} \Rightarrow \text{average error} \leq \frac{RB}{\sqrt{T}} \leq \varepsilon.$$

## Advantages:

- dimension-independent (no  $d$  in the bound)!
- holds for both average, or best iterate.

# Smooth Functions

## Definition

Let  $f : \text{dom}(f) \rightarrow \mathbb{R}$  be differentiable,  $X \subseteq \text{dom}(f)$ ,  $L \in \mathbb{R}_+$ . Function  $f$  is called **smooth** (with parameter  $L$ ) over  $X$  if

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{L}{2} \|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in X.$$

$f$  smooth :  $\iff$  smooth over  $\mathbb{R}^d$ .

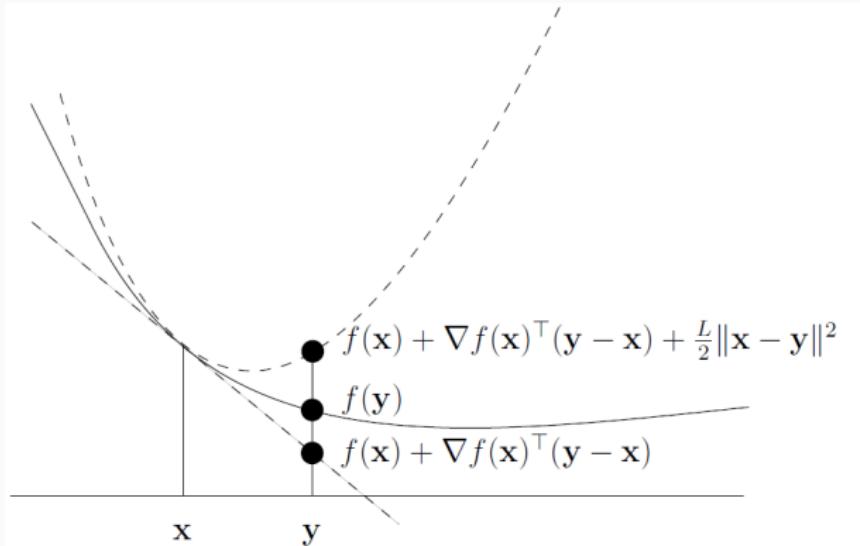
Definition does not require convexity.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Smooth Functions

**Smoothness:** For any  $\mathbf{x}$ , the graph of  $f$  is below a not too steep tangent paraboloid at  $(\mathbf{x}, f(\mathbf{x}))$ :



- In general: quadratic functions are smooth.
- Operations that preserve smoothness (the same that preserve convexity).

## Lemma

Let  $f_1, \dots, f_m$  be functions that are smooth with parameters  $L_1, \dots, L_m$  and let  $\lambda_1, \dots, \lambda_m \in \mathbb{R}_+$ . Then, the function  $f := \sum_{i=1}^m \lambda_i f_i$  is smooth with parameter  $\sum_{i=1}^m \lambda_i L_i$ .

## Lemma

Let  $f$  be smooth with parameter  $L$ , and let  $g(\mathbf{x}) = A\mathbf{x} + \mathbf{b}$ , for  $A \in \mathbb{R}^{d \times m}$  and  $\mathbf{b} \in \mathbb{R}^d$ . Then the function  $f \circ g$  is smooth with parameter  $L\|A\|^2$ , where  $\|A\|$  is the spectral norm of  $A$ .



- Bounded gradients  $\iff$  Lipschitz continuity of  $f$ .
- Smoothness  $\iff$  Lipschitz continuity of  $\nabla f$  (in the convex case).

## Lemma

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be convex and differentiable. The following two statements are equivalent.

1.  $f$  is smooth with parameter  $L$ .
2.  $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$  for all  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ .



# Sufficient Decrease

## Lemma

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be differentiable and smooth with parameter  $L$ . With stepsize

$$\gamma := \frac{1}{L},$$

gradient descent satisfies

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L} \|\nabla f(\mathbf{x}_t)\|^2, \quad t \geq 0.$$

## Remark

*More specifically, this already holds if  $f$  is smooth with parameter  $L$  over the line segment connecting  $\mathbf{x}_t$  and  $\mathbf{x}_{t+1}$ .*



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Sufficient Decrease

## Proof

Use smoothness and definition of gradient descents

$(\mathbf{x}_{t+1} - \mathbf{x}_t = -\nabla f(\mathbf{x}_t)/L)$ :

$$\begin{aligned}f(\mathbf{x}_{t+1}) &\leq f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^\top (\mathbf{x}_{t+1} - \mathbf{x}_t) + \frac{L}{2} \|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 \\&= f(\mathbf{x}_t) - \frac{1}{L} \|\nabla f(\mathbf{x}_t)\|^2 + \frac{1}{2L} \|\nabla f(\mathbf{x}_t)\|^2 \\&= f(\mathbf{x}_t) - \frac{1}{2L} \|\nabla f(\mathbf{x}_t)\|^2.\end{aligned}$$



## Theorem

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be convex and differentiable with a global minimum  $\mathbf{x}^*$ .

Furthermore, suppose that  $f$  is smooth with parameter  $L$ . Choosing stepsize

$$\gamma := \frac{1}{L}$$

gradient descent yields

$$f(\mathbf{x}_T) - f(\mathbf{x}^*) \leq \frac{L}{2T} \|\mathbf{x}_0 - \mathbf{x}^*\|^2, \quad T > 0.$$



## Proof

Vanilla Analysis:

$$\sum_{t=0}^{T-1} \left( f(\mathbf{x}_t) - f(\mathbf{x}^*) \right) \leq \frac{\gamma}{2} \sum_{t=0}^{T-1} \|\nabla f(\mathbf{x}_t)\|^2 + \frac{1}{2\gamma} \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

This time, we can bound the squared gradients by sufficient decrease:

$$\frac{1}{2L} \sum_{t=0}^{T-1} \|\nabla f(\mathbf{x}_t)\|^2 \leq \sum_{t=0}^{T-1} \left( f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}) \right) = f(\mathbf{x}_0) - f(\mathbf{x}_T).$$



# Smooth Convex Functions: $\mathcal{O}(1/\varepsilon^2)$ Steps

## Proof

Putting it together with  $\gamma = 1/L$ :

$$\begin{aligned}\sum_{t=0}^{T-1} (f(\mathbf{x}_t) - f(\mathbf{x}^*)) &\leq \frac{1}{2L} \sum_{t=0}^{T-1} \|\nabla f(\mathbf{x}_t)\|^2 + \frac{L}{2} \|\mathbf{x}_0 - \mathbf{x}^*\|^2 \\ &\leq f(\mathbf{x}_0) - f(\mathbf{x}_T) + \frac{L}{2} \|\mathbf{x}_0 - \mathbf{x}^*\|^2.\end{aligned}$$

Rewriting:

$$\sum_{t=1}^T (f(\mathbf{x}_t) - f(\mathbf{x}^*)) \leq \frac{L}{2} \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

As last iterate is the best (sufficient decrease!):

$$f(\mathbf{x}_T) - f(\mathbf{x}^*) \leq \frac{1}{T} \sum_{t=1}^T (f(\mathbf{x}_t) - f(\mathbf{x}^*)) \leq \frac{L}{2T} \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$



# Smooth Convex Functions: $\mathcal{O}(1/\varepsilon^2)$ Steps

$$R^2 := \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

$$T \geq \frac{R^2 L}{2\varepsilon} \Rightarrow \text{error} \leq \frac{L}{2T} R^2 \leq \varepsilon.$$

- $50 \cdot R^2 L$  iterations for error 0.01 ...
- ... as opposed to  $10,000 \cdot R^2 B^2$  in the Lipschitz case.

# Assignment (by Tuesday 7 March 2023)

1. Compute the gradient of the following functions:
  - $F = \mathbf{a}^\top \mathbf{x}$ ;
  - $F = \mathbf{x}^\top A \mathbf{x}$  with  $A$  symmetric;
  - $F = \mathbf{x}^\top A \mathbf{x}$  for a generic matrix  $A$ ;
  - $F = (\mathbf{a}^\top \mathbf{x})^2$ ;
  - $F = \frac{1}{2n} \|A\mathbf{x} - \mathbf{b}\|^2$ .
2. Prove that the quadratic function  $f(\mathbf{x}) = \mathbf{x}^\top Q\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ , with  $Q$  symmetric is smooth with parameter  $2\|Q\|$ .
3. Apply 2 iterations of the gradient descent method to the function  $F(x_1, x_2) = (x_1 - 1)^2 + x_2^3 - x_1 x_2$  with  $\mathbf{x}_0 = (1, 1)$ .

# Optimization

Francesca Maggioni

---

Lesson 4 - Academic Year 2023-2024

Department of Management, Information and Production Engineering  
University of Bergamo



# Can We Go Even Faster?

- So far: Error decreases with  $1/\sqrt{t}$ , or  $1/T$ ...
- Could it decrease exponentially in  $T$ ?
- On  $f(x) := x^2$ . Stepsize  $\gamma := 1/2$  ( $f$  is  $L = 2$ -smooth):

$$x_{t+1} := x_t - \frac{1}{2} \nabla f(x_t) = x_t - x_t = 0$$

converged in one step!

- Same for  $f(x) := x^2$ . Stepsize  $\gamma := 1/4$  ( $f$  is  $L = 4$ -smooth):

$$x_{t+1} := x_t - \frac{1}{4} \nabla f(x_t) = x_t - \frac{x_t}{2} = \frac{x_t}{2}$$

so  $f(x_t) = f(\frac{x_0}{2^t}) = \frac{1}{2^{2t}} x_0^2$ . Exponential in  $t$ .



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

## Definition

Let  $f : \text{dom}(f) \rightarrow \mathbb{R}$  be a differentiable function,  $X \subseteq \text{dom}(f)$  convex and  $\mu \in \mathbb{R}_+, \mu > 0$ . Function  $f$  is called **strongly convex** (with parameter  $\mu$ ) over  $X$  if:

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{\mu}{2} \|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in X.$$

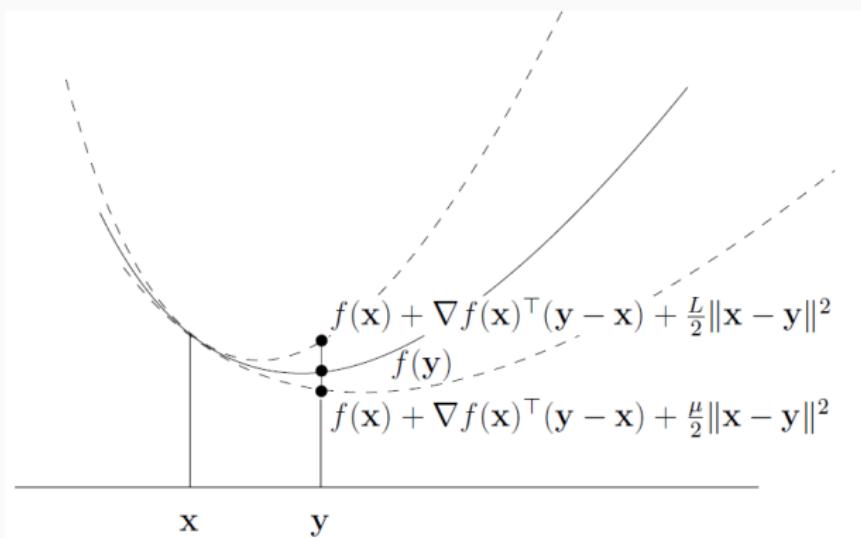
## Lemma

If  $f$  is strongly convex with parameter  $\mu > 0$ , then  $f$  is strictly convex and has a unique global minimum.



# Strongly Convex Functions

**Strong convexity:** For any  $x$ , the graph of  $f$  is above a not too flat tangent paraboloid at  $(x, f(x))$ :



## Smooth and Strongly Convex Functions: $\mathcal{O}(\log(1/\varepsilon))$ Steps

Want to show:  $\lim_{t \rightarrow \infty} \mathbf{x}_t = \mathbf{x}^*$ .

Vanilla Analysis:

$$\nabla f(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{x}^*) = \frac{\gamma}{2} \|\nabla f(\mathbf{x}_t)\|^2 + \frac{1}{2\gamma} (\|\mathbf{x}_t - \mathbf{x}^*\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2).$$

Now use stronger lower bound on left hand side, coming from strong convexity:

$$\nabla f(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{x}^*) \geq f(\mathbf{x}_t) - f(\mathbf{x}^*) + \frac{\mu}{2} \|\mathbf{x}_t - \mathbf{x}^*\|^2.$$

Putting it together:

$$f(\mathbf{x}_t) - f(\mathbf{x}^*) \leq \frac{1}{2\gamma} (\gamma^2 \|\nabla f(\mathbf{x}_t)\|^2 + \|\mathbf{x}_t - \mathbf{x}^*\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2) - \frac{\mu}{2} \|\mathbf{x}_t - \mathbf{x}^*\|^2.$$

Rewriting:

$$\|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 \leq 2\gamma (f(\mathbf{x}^*) - f(\mathbf{x}_t)) + \gamma^2 \|\nabla f(\mathbf{x}_t)\|^2 + (1 - \mu\gamma) \|\mathbf{x}_t - \mathbf{x}^*\|^2.$$



# Smooth and Strongly Convex Functions: $\mathcal{O}(\log(1/\varepsilon))$ Steps

Squared distance to  $\mathbf{x}^*$  goes down by a constant factor, up to some noise.

## Theorem

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be differentiable with a global minimum  $\mathbf{x}^*$ ; suppose that  $f$  is smooth with parameter  $L$  and strongly convex with parameter  $\mu > 0$ .

Choosing  $\gamma := \frac{1}{L}$ , gradient descent with arbitrary  $\mathbf{x}_0$  satisfies the following two properties.

1. Squared distances to  $\mathbf{x}^*$  are geometrically decreasing:

$$\|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 \leq \left(1 - \frac{\mu}{L}\right) \|\mathbf{x}_t - \mathbf{x}^*\|^2, \quad t \geq 0. \quad (\star)$$

2. The absolute error after  $T$  iterations is exponentially small in  $T$ :

$$f(\mathbf{x}_T) - f(\mathbf{x}^*) \leq \frac{L}{2} \left(1 - \frac{\mu}{L}\right)^T \|\mathbf{x}_0 - \mathbf{x}^*\|^2, \quad T > 0. \quad (\star\star)$$



# Smooth and Strongly Convex Functions: $\mathcal{O}(\log(1/\varepsilon))$ Steps

**Proof of  $(\star)$ .**

Bounding the noise:

$$\begin{aligned} 2\gamma(f(\mathbf{x}^*) - f(\mathbf{x}_t)) + \gamma^2 \|\nabla f(\mathbf{x}_t)\|^2 &= \frac{2}{L}(f(\mathbf{x}^*) - f(\mathbf{x}_t)) + \frac{1}{L^2} \|\nabla f(\mathbf{x}_t)\|^2 \\ &\leq \frac{2}{L}(f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t)) + \frac{1}{L^2} \|\nabla f(\mathbf{x}_t)\|^2 \\ &\leq -\frac{1}{L^2} \|\nabla f(\mathbf{x}_t)\|^2 + \frac{1}{L^2} \|\nabla f(\mathbf{x}_t)\|^2 = 0. \end{aligned}$$

Hence, the noise is non-positive, and we get  $(\star)$ :

$$\|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 \leq (1 - \mu\gamma) \|\mathbf{x}_t - \mathbf{x}^*\|^2 = \left(1 - \frac{\mu}{L}\right) \|\mathbf{x}_t - \mathbf{x}^*\|^2.$$



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Smooth and Strongly Convex Functions: $\mathcal{O}(\log(1/\varepsilon))$ Steps

## Proof of 2.

From  $(*)$ :

$$\|\mathbf{x}_T - \mathbf{x}^*\|^2 \leq \left(1 - \frac{\mu}{L}\right)^T \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

Smoothness together with  $\nabla f(\mathbf{x}^*) = \mathbf{0}$ :

$$f(\mathbf{x}_T) - f(\mathbf{x}^*) \leq \nabla f(\mathbf{x}^*)^\top (\mathbf{x}_T - \mathbf{x}^*) + \frac{L}{2} \|\mathbf{x}_T - \mathbf{x}^*\|^2.$$

Putting it together:

$$f(\mathbf{x}_T) - f(\mathbf{x}^*) \leq \frac{L}{2} \|\mathbf{x}_T - \mathbf{x}^*\|^2 \leq \frac{L}{2} \left(1 - \frac{\mu}{L}\right)^T \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

## Smooth and Strongly Convex Functions: $\mathcal{O}(\log(1/\varepsilon))$ Steps

$$R^2 := \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

$$T \geq \frac{L}{\mu} \ln \left( \frac{R^2 L}{2\varepsilon} \right) \Rightarrow \text{error} \leq \frac{L}{2} \left( 1 - \frac{\mu}{L} \right)^T R^2 \leq \varepsilon.$$

Conclusion: To reach absolute error at most  $\varepsilon$ , we only need  $\mathcal{O}(\log(1/\varepsilon))$  iterations, e.g.:

- $\frac{L}{\mu} \ln(50 \cdot R^2 L)$  iterations for error 0.01 ...
- ... as opposed to  $50 \cdot R^2 L$  in the smooth case.



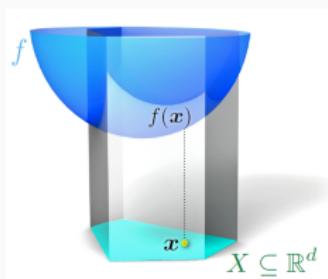
# Projected Gradient Descent: Constrained Optimization

## Constrained Optimization Problem

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in X. \end{aligned}$$

Solving Constrained Optimization Problems

- (A) Projected Gradient Descent.
- (B) Transform it into an unconstrained problem.



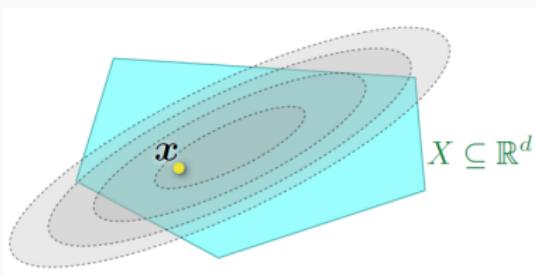
# Projected Gradient Descent: Constrained Optimization

## Constrained Optimization Problem

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in X. \end{aligned}$$

Solving Constrained Optimization Problems

- (A) **Projected Gradient Descent**  $\Leftarrow$  HERE!.
- (B) Transform it into an unconstrained problem.



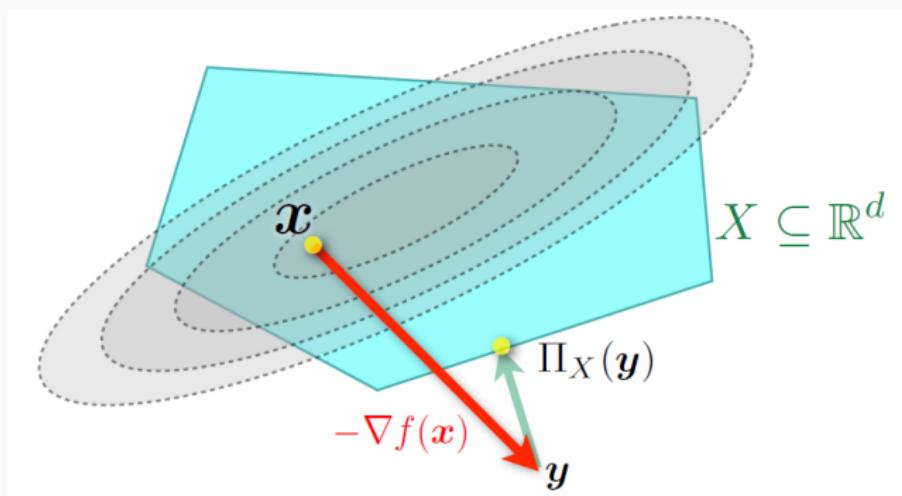
$$X \subseteq \mathbb{R}^d$$



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Projected Gradient Descent

**Idea:** project onto  $X$  after every step:  $\Pi_X(\mathbf{y}) := \operatorname{argmin}_{\mathbf{x} \in X} \|\mathbf{x} - \mathbf{y}\|$ .



Projected gradient descent:  $\mathbf{x}_{t+1} := \Pi_X [\mathbf{x}_t - \gamma \nabla f(\mathbf{x}_t)]$ .

# The Algorithm

## Projected gradient descent:

$$\mathbf{y}_{t+1} := \mathbf{x}_t - \gamma \nabla f(\mathbf{x}_t),$$

$$\mathbf{x}_{t+1} := \Pi_X(\mathbf{y}_{t+1}) := \operatorname{argmin}_{\mathbf{x} \in X} \|\mathbf{x} - \mathbf{y}_{t+1}\|^2.$$

for **timesteps**  $t = 0, 1, \dots$ , and **stepsize**  $\gamma \geq 0$ .

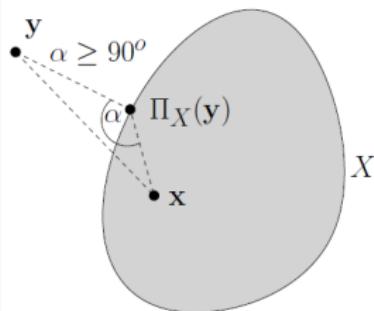


# Properties of Projection

## Fact

Let  $X \subseteq \mathbb{R}^d$  be closed and convex,  $\mathbf{x} \in X$ ,  $\mathbf{y} \in \mathbb{R}^d$ . Then:

- (a)  $(\mathbf{x} - \Pi_X(\mathbf{y}))^\top (\mathbf{y} - \Pi_X(\mathbf{y})) \leq 0$ .
- (b)  $\|\mathbf{x} - \Pi_X(\mathbf{y})\|^2 + \|\mathbf{y} - \Pi_X(\mathbf{y})\|^2 \leq \|\mathbf{x} - \mathbf{y}\|^2$ .



# Properties of Projection

## Proof of (a)

$\Pi_X(\mathbf{y})$  is minimizer of (differentiable) convex function  $d_{\mathbf{y}}(\mathbf{x}) = \|\mathbf{x} - \mathbf{y}\|^2$  over  $X$ . By first-order characterization of optimality

$$\begin{aligned} 0 &\leq \nabla d_{\mathbf{y}}(\Pi_X(\mathbf{y}))^\top (\mathbf{x} - \Pi_X(\mathbf{y})) \\ &= 2(\Pi_X(\mathbf{y}) - \mathbf{y})^\top (\mathbf{x} - \Pi_X(\mathbf{y})) \\ \iff 0 &\geq 2(\mathbf{y} - \Pi_X(\mathbf{y}))^\top (\mathbf{x} - \Pi_X(\mathbf{y})) \\ \iff 0 &\geq (\mathbf{x} - \Pi_X(\mathbf{y}))^\top (\mathbf{y} - \Pi_X(\mathbf{y})). \end{aligned}$$



# Properties of Projection

## Proof of (b)

$$\mathbf{v} := (\mathbf{x} - \Pi_X(\mathbf{y})), \quad \mathbf{w} := (\mathbf{y} - \Pi_X(\mathbf{y})).$$

By (a):

$$\begin{aligned} 0 \geq 2\mathbf{v}^\top \mathbf{w} &= ||\mathbf{v}||^2 + ||\mathbf{w}||^2 - ||\mathbf{v} - \mathbf{w}||^2 \\ &= ||\mathbf{x} - \Pi_X(\mathbf{y})||^2 + ||\mathbf{y} - \Pi_X(\mathbf{y})||^2 - ||\mathbf{x} - \mathbf{y}||^2. \end{aligned}$$



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Results for Projected Gradient Descent over Closed and Convex $X$

The **same** number of steps as gradient over  $\mathbb{R}^d$ !

- Lipschitz convex functions over  $X$ :  $\mathcal{O}(1/\varepsilon^2)$  steps.
- Smooth convex functions over  $X$ :  $\mathcal{O}(1/\varepsilon)$  steps.
- Smooth and strongly convex functions over  $X$ :  $\mathcal{O}(\log(1/\varepsilon))$  steps.

We will adapt the previous proofs for gradient descent. BUT:

- Each step involves a projection onto  $X$
- may or may not be efficient (in relevant cases, it is)...



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Lipschitz Convex Functions over $X$ : $\mathcal{O}(1/\varepsilon^2)$ Steps

Assume that all gradients of  $f$  are bounded in norm over closed and convex  $X$ .

- Equivalent to  $f$  being Lipschitz over  $X$ .
- Many interesting functions are Lipschitz over bounded sets  $X$ .

## Theorem (Same as the unconstrained one, but more useful)

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be convex and differentiable,  $X \subseteq \mathbb{R}^d$  closed and convex,  $\mathbf{x}^*$  a minimizer of  $f$  over  $X$ ; furthermore, suppose that  $\|\mathbf{x}_0 - \mathbf{x}^*\| \leq R$  with  $\mathbf{x}_0 \in X$  and  $\|\nabla f(\mathbf{x})\| \leq B$  for all  $\mathbf{x} \in X$ . Choosing the constant stepsize:

$$\gamma := \frac{R}{B\sqrt{T}}$$

**projected** gradient descent yields

$$\frac{1}{T} \sum_{t=0}^{T-1} f(\mathbf{x}_t) - f(\mathbf{x}^*) \leq \frac{RB}{\sqrt{T}}.$$



## Proof

- Replace  $\mathbf{x}_{t+1}$  in the vanilla analysis with  $\mathbf{y}_{t+1}$  (the unprojected gradient step):

$$\mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*) = \frac{1}{2\gamma} (\gamma^2 \|\mathbf{g}_t\|^2 + \|\mathbf{x}_t - \mathbf{x}^*\|^2 - \|\mathbf{y}_{t+1} - \mathbf{x}^*\|^2).$$

- Use Fact (b):  $\|\mathbf{x} - \Pi_X(\mathbf{y})\|^2 + \|\mathbf{y} - \Pi_X(\mathbf{y})\|^2 \leq \|\mathbf{x} - \mathbf{y}\|^2$ .
- With  $\mathbf{x} = \mathbf{x}^*$ ,  $\mathbf{y} = \mathbf{y}_{t+1}$  we have  $\Pi_X(\mathbf{y}) = \mathbf{x}_{t+1}$ , and hence:

$$\|\mathbf{x}^* - \mathbf{x}_{t+1}\|^2 \leq \|\mathbf{x}^* - \mathbf{y}_{t+1}\|^2.$$

- We go back to the original vanilla analysis and continue from there as before:

$$\mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*) \leq \frac{1}{2\gamma} (\gamma^2 \|\mathbf{g}_t\|^2 + \|\mathbf{x}_t - \mathbf{x}^*\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2).$$



# Smooth Functions over $X$

Recall:

$f$  is called smooth (with parameter  $L$ ) over  $X$  if:

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{L}{2} \|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in X.$$

# Sufficient Decrease

## Lemma

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be differentiable and smooth with parameter  $L$  over  $X$ .

Choosing stepsize

$$\gamma := \frac{1}{L},$$

projected gradient descent with arbitrary  $\mathbf{x}_0 \in X$  satisfies

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L} \|\nabla f(\mathbf{x}_t)\|^2 + \frac{L}{2} \|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2, \quad t \geq 0.$$



# Assignments due by April 16, 2024

- Prove that the function  $d_y : \mathbb{R}^d \rightarrow \mathbb{R}$ ,  $x \mapsto \|x - y\|^2$  is strictly convex for any  $y \in \mathbb{R}^d$ .
- Prove that  $f(x) = x^4$  is smooth over  $X = (-a; a)$  and determine a concrete smoothness parameter  $L$ .



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# **Optimization - 3 CFU**

Lesson 5

Instructor: Francesca Maggioni

---

Academic Year 2023-2024

Department of Management, Information and Production Engineering  
University of Bergamo



# Composite Optimization Problems

Consider objective functions composed as:

$$f(\mathbf{x}) := g(\mathbf{x}) + h(\mathbf{x})$$

where  $g$  is a **nice** function, whereas  $h$  is a **simple** additional term, which however doesn't satisfy the assumptions of niceness which we used in the convergence analysis so far. In particular, an important case is when  $h$  is not differentiable.



The classical gradient step for minimizing  $g$ :

$$\mathbf{x}_{t+1} = \arg \min_{\mathbf{y}} g(\mathbf{x}_t) + \nabla g(\mathbf{x}_t)^T (\mathbf{y} - \mathbf{x}_t) + \frac{1}{2\gamma} \|\mathbf{y} - \mathbf{x}_t\|^2.$$

For the stepsize  $\gamma = 1/L$  it exactly minimizes the local quadratic model of  $g$  at our current iterate  $\mathbf{x}_t$ , formed by the smoothness property with parameter  $L$ .

Now for  $f = g + h$ , keep the same for  $g$ , and add  $h$  unmodified.

$$\begin{aligned}\mathbf{x}_{t+1} &:= \arg \min_{\mathbf{y}} g(\mathbf{x}_t) + \nabla g(\mathbf{x}_t)^T (\mathbf{y} - \mathbf{x}_t) + \frac{1}{2\gamma} \|\mathbf{y} - \mathbf{x}_t\|^2 + h(\mathbf{y}) \\ &= \arg \min_{\mathbf{y}} \frac{1}{2\gamma} \|\mathbf{y} - (\mathbf{x}_t - \gamma \nabla g(\mathbf{x}_t))\|^2 + h(\mathbf{y}),\end{aligned}$$

the **proximal gradient descent** update.



# The Proximal Gradient Descent Algorithm

An iteration of proximal gradient descent is defined as:

$$\mathbf{x}_{t+1} = \text{prox}_{h,\gamma}(\mathbf{x}_t - \gamma \nabla g(\mathbf{x}_t))$$

where the **proximal mapping** for a given function  $h$ , and parameter  $\gamma > 0$  is defined as

$$\text{prox}_{h,\gamma}(\mathbf{z}) := \arg \min_{\mathbf{y}} \left\{ \frac{1}{2\gamma} \|\mathbf{y} - \mathbf{z}\|^2 + h(\mathbf{y}) \right\}.$$

The update step can be equivalently written as

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma G_\gamma(\mathbf{x}_t)$$

for

$$G_{h,\gamma}(\mathbf{x}) := \frac{1}{\gamma} (\mathbf{x} - \text{prox}_{h,\gamma}(\mathbf{x} - \gamma \nabla g(\mathbf{x})))$$

being the so called **generalized gradient** of  $f$ .



# A Generalization of Gradient Descent?

- $h \equiv 0$  recover gradient descent
- $h \equiv \iota_X$ : recover projected gradient descent!

Given a closed convex set  $X$ , the indicator function of the set  $X$  is given as the convex function

$$\iota_X : \mathbb{R}^d \rightarrow \mathbb{R} \cup +\infty$$

$$\mathbf{x} \mapsto \iota_X(\mathbf{x}) := \begin{cases} 0 & \text{if } \mathbf{x} \in X \\ +\infty & \text{otherwise.} \end{cases}$$

Proximal mapping becomes

$$\text{prox}_{h,\gamma}(\mathbf{z}) := \arg \min_{\mathbf{y}} \left\{ \frac{1}{2\gamma} \|\mathbf{y} - \mathbf{z}\|^2 + \iota_X(\mathbf{y}) \right\} = \arg \min_{\mathbf{y} \in \mathbf{X}} \|\mathbf{y} - \mathbf{z}\|^2.$$



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

## Convergence in $\mathcal{O}(1/\varepsilon)$ Steps

Same as vanilla case for smooth functions, but now for any  $h$  for which we can compute the proximal mapping.

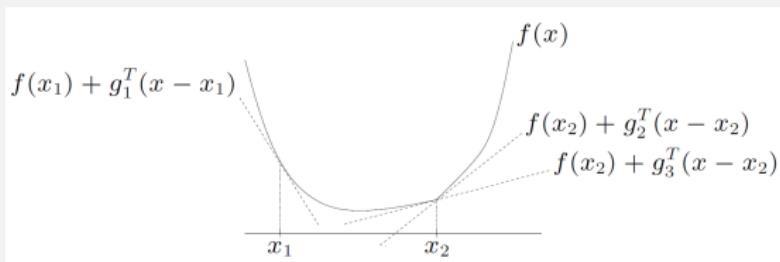
# Subgradients

What if  $f$  is not differentiable?

## Definition

$\mathbf{g} \in \mathbb{R}^d$  is a **subgradient** of  $f$  at  $\mathbf{x}$  if:

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{g}^\top (\mathbf{y} - \mathbf{x}) \quad \forall \mathbf{y} \in \text{dom}(f).$$

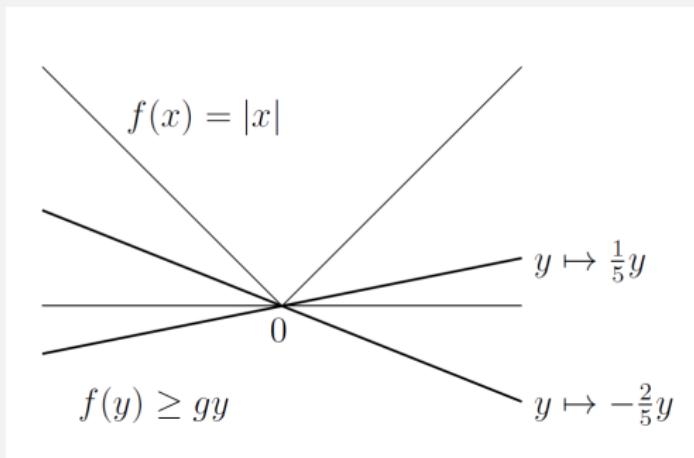


$\partial f(\mathbf{x}) \subseteq \mathbb{R}^d$  is the **subdifferential**, the set of subgradients of  $f$  at  $\mathbf{x}$ .



# Subgradients

## Example



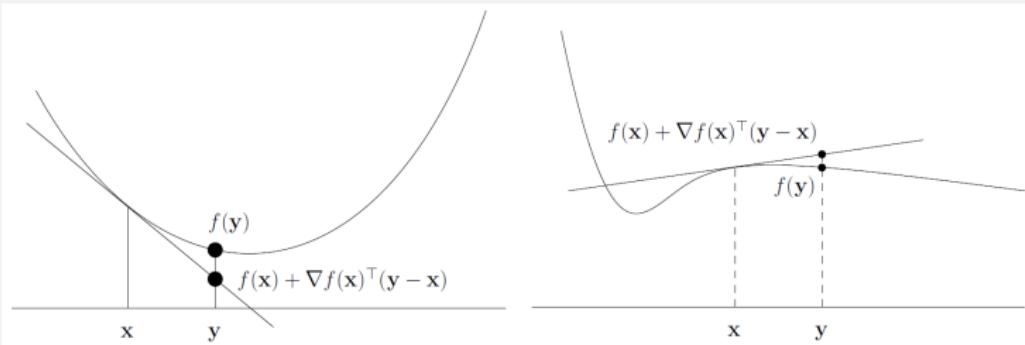
**Subgradient condition at  $x = 0$ :**  $f(y) \geq f(0) + g(y - 0) = gy$ .  
 $\partial f(0) = [-1, 1]$ .



# Subgradients

## Lemma

If  $f : \text{dom}(f) \rightarrow \mathbb{R}$  is differentiable at  $x \in \text{dom}(f)$ , then  $\partial f(x) \subseteq \{\nabla f(x)\}$ .  
Either exactly one subgradient  $\nabla f(x)$  ... ... or no subgradient at all.

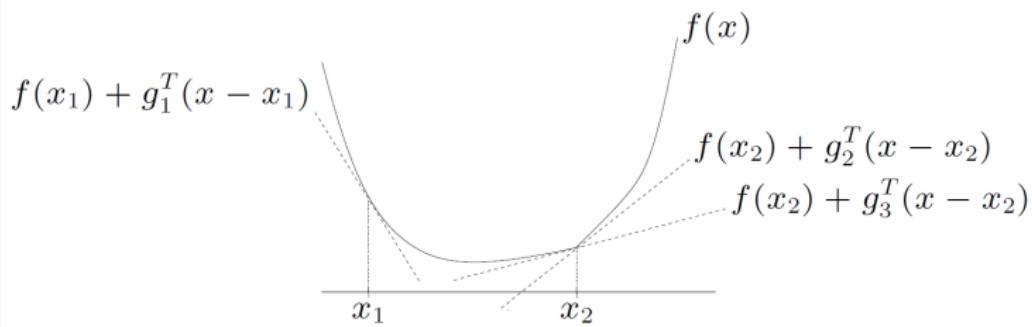


# Subgradient Characterization of Convexity

convex = subgradients everywhere.

## Lemma

A function  $f : \text{dom}(f) \rightarrow \mathbb{R}$  is convex if and only if  $\text{dom}(f)$  is convex and  $\partial f(\mathbf{x}) \neq \emptyset$  for all  $\mathbf{x} \in \text{dom}(f)$ .



## Lemma

Let  $f : \text{dom}(f) \rightarrow \mathbb{R}$  be convex,  $\text{dom}(f)$  open,  $B \in \mathbb{R}_+$ . Then the following two statements are equivalent.

1.  $\|\mathbf{g}\| \leq B$  for all  $\mathbf{x} \in \text{dom}(f)$  and all  $\mathbf{g} \in \partial f(\mathbf{x})$ .
2.  $|f(\mathbf{x}) - f(\mathbf{y})| \leq B\|\mathbf{x} - \mathbf{y}\|$  for all  $\mathbf{x}, \mathbf{y} \in \text{dom}(f)$ .



# Subgradient Optimality Condition

## Lemma

Suppose that  $f : \text{dom}(f) \rightarrow \mathbb{R}$  and  $\mathbf{x} \in \text{dom}(f)$ . If  $\mathbf{0} \in \partial f(\mathbf{x})$ , then  $\mathbf{x}$  is a global minimum.

## Proof

By definition of subgradients,  $\mathbf{g} = \mathbf{0} \in \partial f(\mathbf{x})$  gives

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{g}^\top (\mathbf{y} - \mathbf{x}) = f(\mathbf{x})$$

for all  $\mathbf{y} \in \text{dom}(f)$ , so  $\mathbf{x}$  is a global minimum.

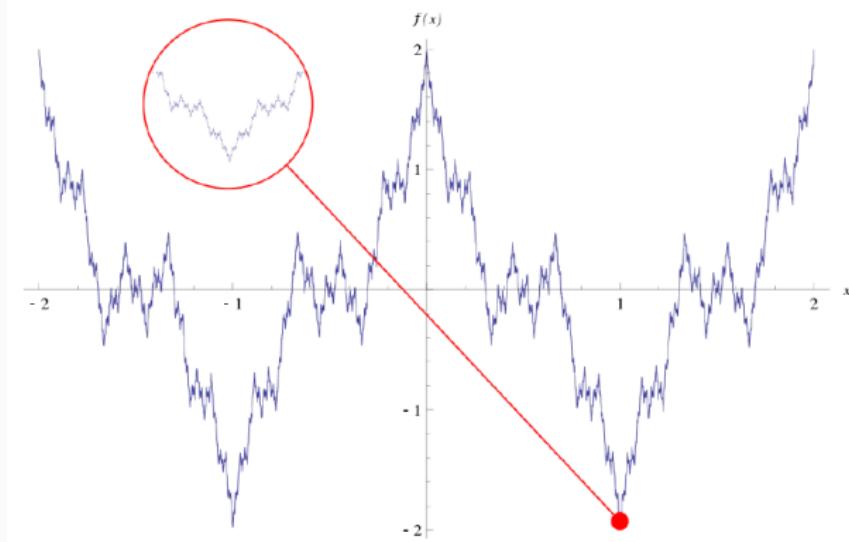


UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Differentiability of Convex Functions

How *wild* can a non-differentiable convex function be?

**Weierstrass function:** a function that is continuous everywhere but differentiable nowhere.



## Theorem

A convex function  $f : \text{dom}(f) \rightarrow \mathbb{R}$  is differentiable almost everywhere. In other words:

- Set of points where  $f$  is non-differentiable has measure 0 (no volume).
- For all  $\mathbf{x} \in \text{dom}(f)$  and all  $\varepsilon > 0$ , there is a point  $\mathbf{x}'$  such that  $\|\mathbf{x} - \mathbf{x}'\| \leq \varepsilon$  and  $f$  is differentiable at  $\mathbf{x}'$ .



# The Subgradient Descent Algorithm

**Subgradient descent:** choose  $\mathbf{x}_0 \in \mathbb{R}^d$ .

Let  $\mathbf{g}_t \in \partial f(\mathbf{x}_t)$

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma_t \mathbf{g}_t$$

for times  $t = 0, 1, \dots$ , and stepsizes  $\gamma_t \geq 0$ .

Stepsize can vary with time!

This is possible in (projected) gradient descent as well, but so far, we didn't need it.

# Lipschitz Convex Functions: $\mathcal{O}(1/\varepsilon^2)$ Steps

## Theorem

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be convex and  $B$ -Lipschitz continuous with a global minimum  $\mathbf{x}^*$ ; furthermore, suppose that  $\|\mathbf{x}_0 - \mathbf{x}^*\| \leq R$ . Choosing the constant stepsize:

$$\gamma := \frac{R}{B\sqrt{T}}$$

subgradient descent yields:  $\frac{1}{T} \sum_{t=0}^{T-1} f(\mathbf{x}_t) - f(\mathbf{x}^*) \leq \frac{RB}{\sqrt{T}}$ .

## Proof

Proof is identical to the one proposed in the theorem for convex and differentiable  $f$  with a global minimum  $\mathbf{x}^*$ ,  $\|\mathbf{x}_0 - \mathbf{x}^*\| \leq R$  and  $\|\nabla f(\mathbf{x})\| \leq B$  for all  $\mathbf{x}$ . The only differences are:

- In vanilla analysis, now use  $\mathbf{g}_t \in \partial f(\mathbf{x}_t)$  instead of  $\mathbf{g}_t = \nabla f(\mathbf{x}_t)$ .
- Inequality  $f(\mathbf{x}_t) - f(\mathbf{x}^*) \leq \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*)$  now follows from subgradient property instead of first-order characterization of convexity.

# Optimality of First-Order Methods

With all the convergence rates we have seen so far, a very natural question to ask is if these rates are best possible or not. Surprisingly, the rate can indeed not be improved in general.

## Theorem (Nesterov)

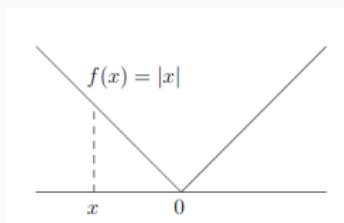
*For any  $T \leq d - 1$  and starting point  $\mathbf{x}_0$ , there is a function  $f$  in the problem class of  $B$ -Lipschitz functions over  $\mathbb{R}^d$ , such that any (sub)gradient method has an objective error at least*

$$f(\mathbf{x}_T) - f(\mathbf{x}^*) \geq \frac{RB}{2(1 + \sqrt{T + 1})}.$$



# Smooth (Non-Differentiable) Functions?

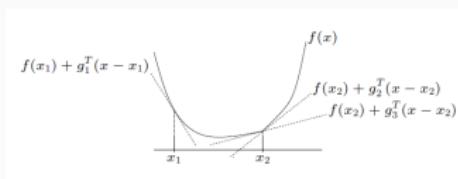
They don't exist!



At 0, graph can't be below a tangent paraboloid.

Can we still improve over  $\mathcal{O}(1/\varepsilon^2)$  steps for Lipschitz functions?

Yes, if we also require strong convexity (graph is above not too at tangent paraboloids).



# Strongly Convex Functions

Straightforward generalization to the non-differentiable case:

## Definition

Let  $f : \text{dom}(f) \rightarrow \mathbb{R}$  be convex,  $\mu \in \mathbb{R}_+, \mu > 0$ . Function  $f$  is called **strongly convex** (with parameter  $\mu$ ) if:

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{g}^\top (\mathbf{y} - \mathbf{x}) + \frac{\mu}{2} \|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in X, \forall \mathbf{g} \in \partial f(\mathbf{x}).$$

## Lemma

Let  $f : \text{dom}(f) \rightarrow \mathbb{R}$  be convex,  $\text{dom}(f)$  open,  $\mu \in \mathbb{R}_+, \mu > 0$ . Function  $f$  is strongly convex with parameter  $\mu$  if and only if  $f_\mu : \text{dom}(f) \rightarrow \mathbb{R}$  defined by:

$$f_\mu(\mathbf{x}) = f(\mathbf{x}) - \frac{\mu}{2} \|\mathbf{x}\|^2, \quad \mathbf{x} \in \text{dom}(f)$$

is convex.



# Strong Convexity: $\mathcal{O}(1/\varepsilon)$ Steps

## Theorem

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be strongly convex with parameter  $\mu > 0$  and let  $\mathbf{x}^*$  be the unique global minimum of  $f$ . With decreasing stepsize

$$\gamma_t := \frac{2}{\mu(t+1)}, \quad t > 0$$

subgradient descent yields:

$$f\left(\frac{2}{T(T+1)} \sum_{t=1}^T t \cdot \mathbf{x}_t\right) - f(\mathbf{x}^*) \leq \frac{2B^2}{\mu(T+1)}$$

where  $B = \max_{t=1}^T \|\mathbf{g}_t\|$ .



## Strong Convexity: Discussion

$$f\left(\frac{2}{T(T+1)} \sum_{t=1}^T t \cdot \mathbf{x}_t\right) - f(\mathbf{x}^*) \leq \frac{2B^2}{\mu(T+1)}.$$

Weighted average of iterates achieves the bound (later iterates have more weight).

Bound is independent of initial distance  $\|\mathbf{x}_0 - \mathbf{x}^*\| \dots$

...but not really:  $B$  typically depends on  $\|\mathbf{x}_0 - \mathbf{x}^*\|$  (for example,  $B = \mathcal{O}\|\mathbf{x}_0 - \mathbf{x}^*\|$  for quadratic functions).

Recall: we can only hope that  $B$  is small (can be checked while running the algorithm).

What if we don't know the parameter  $\mu$  of strong convexity?

→ In practice, try some  $\mu$ 's, pick best solution obtained.

# References

- R. Tyrrell Rockafellar. **Convex Analysis**. Princeton Landmarks in Mathematics. Princeton University Press, 1997.

## Assignments (due by April 30, 2024)

Consider the quadratic function  $f(x) = \frac{1}{2}x^\top Ax + \langle b; x \rangle + c$ , where  $A$  is a  $d \times d$  symmetric matrix,  $b \in \mathbb{R}^d$  and  $c \in \mathbb{R}$ .

1. What are the minimal conditions on  $A$ ,  $b$  and  $c$  that ensure that  $f$  is strictly convex? For the rest of then exercise we assume that these conditions are fulfilled.
2. Is  $f$  strongly convex?
3. Prove that  $f$  has a unique minimum  $x^*$  and give its closed form expression.
4. Show that  $f$  can be rewritten as

$$f(x) = \frac{1}{2}(x - x^*)^\top A(x - x^*) + f(x^*).$$

5. From an initial point  $x_0 \in \mathbb{R}^d$ , assume we run gradient descent with step-size  $\gamma > 0$  on the function  $f$ . Show that the  $n$ -th iterate  $x_n$  satisfies

$$x_n = x^* + (I_d - \gamma A)^n (x_0 - x^*),$$

where  $I_d$  is the  $d \times d$  identity matrix.

6. In which range must the step-size be so that the iterates converge towards  $x^*$ ?



# **Optimization - 3 CFU**

Francesca Maggioni

---

Academic Year 2023-2024

Department of Management, Information and Production Engineering  
University of Bergamo



# Stochastic Gradient Descent

Many objective functions are **sum structured**:

$$f(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n f_i(\mathbf{x})$$

*Example:*  $f_i$  is the cost function of the  $i$ -th observation, taken from a training set of  $n$  observations.

Evaluating  $\nabla f(\mathbf{x})$  of a sum-structured function is expensive (sum of  $n$  gradients).



# Stochastic Gradient Descent: the Algorithm

Choose  $\mathbf{x}_0$ .

sample  $i \in [n]$  uniformly at random

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma_t \nabla f_i(\mathbf{x}_t).$$

for **times**  $t = 0, 1, \dots$ , and **stepsizes**  $\gamma_t \geq 0$ .

Only update with the gradient of  $f_i$  instead of the full gradient!

Iteration is  $n$  times cheaper than in full gradient descent.

The vector  $\mathbf{g}_t := \nabla f_i(\mathbf{x}_t)$  is called a **stochastic gradient**.

$\mathbf{g}_t$  is a vector of  $d$  random variables, but we will also simply call this a random variable.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Unbiasedness

Can't use convexity

$$f(\mathbf{x}_t) - f(\mathbf{x}^*) \leq \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*)$$

on top of the vanilla analysis, as this may hold or not hold, depending on how the stochastic gradient  $\mathbf{g}_t$  turns out.

We will show (and exploit): the inequality holds **in expectation**.

For this, we use that by definition,  $\mathbf{g}_t$  is an **unbiased estimate** of  $\nabla f(\mathbf{x}_t)$ :

$$\mathbb{E} [\mathbf{g}_t | \mathbf{x}_t = \mathbf{x}] = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\mathbf{x}) = \nabla f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d.$$



# The Inequality $f(\mathbf{x}_t) - f(\mathbf{x}^*) \leq \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*)$ holds in Expectation

For any fixed  $\mathbf{x}$ , linearity of conditional expectations yields

$$\mathbb{E} \left[ \mathbf{g}_t^\top (\mathbf{x} - \mathbf{x}^*) | \mathbf{x}_t = \mathbf{x} \right] = \mathbb{E} [\mathbf{g}_t | \mathbf{x}_t = \mathbf{x}]^\top (\mathbf{x} - \mathbf{x}^*) = \nabla f(\mathbf{x})^\top (\mathbf{x} - \mathbf{x}^*)$$

Event  $\{\mathbf{x}_t = \mathbf{x}\}$  can occur only for  $\mathbf{x}$  in some finite set  $X$  ( $\mathbf{x}_t$  is determined by the choices of indices in all iterations so far).

## Theorem (Partition Theorem)

$$\begin{aligned} \mathbb{E} \left[ \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*) \right] &= \sum_{\mathbf{x} \in X} \mathbb{E} \left[ \mathbf{g}_t^\top (\mathbf{x} - \mathbf{x}^*) | \mathbf{x}_t = \mathbf{x} \right] \text{prob}(\mathbf{x}_t = \mathbf{x}) \\ &= \sum_{\mathbf{x} \in X} \nabla f(\mathbf{x})^\top (\mathbf{x} - \mathbf{x}^*) \text{prob}(\mathbf{x}_t = \mathbf{x}) \\ &= \mathbb{E} \left[ \nabla f(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{x}^*) \right] \end{aligned}$$

Hence,

$$\mathbb{E} \left[ \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*) \right] = \mathbb{E} \left[ \nabla f(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{x}^*) \right] \geq \mathbb{E} [f(\mathbf{x}_t) - f(\mathbf{x}^*)].$$

# Bounded Stochastic Gradients: $\mathcal{O}(1/\varepsilon^2)$ Steps

## Theorem

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be convex and differentiable,  $\mathbf{x}^*$  a global minimum; furthermore, suppose that  $\|\mathbf{x}_0 - \mathbf{x}^*\| \leq R$  and that  $\mathbb{E}[\|\mathbf{g}_t\|^2] \leq B^2$  for all  $t$ . Choosing the constant stepsize:

$$\gamma := \frac{R}{B\sqrt{T}}$$

stochastic gradient descent yields:

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[f(\mathbf{x}_t)] - f(\mathbf{x}^*) \leq \frac{RB}{\sqrt{T}}.$$



# Bounded Stochastic Gradients: $\mathcal{O}(1/\varepsilon^2)$ Steps

## Proof

Vanilla analysis (this time,  $\mathbf{g}_t$  is the stochastic gradient):

$$\sum_{t=0}^{T-1} \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*) \leq \frac{\gamma}{2} \sum_{t=0}^{T-1} \|\mathbf{g}_t\|^2 + \frac{1}{2\gamma} \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

Taking expectations and using convexity in expectation:

$$\begin{aligned} \sum_{t=0}^{T-1} \mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{x}^*)] &\leq \sum_{t=0}^{T-1} \mathbb{E}\left[\mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*)\right] \\ &\leq \frac{\gamma}{2} \sum_{t=0}^{T-1} \mathbb{E}[\|\mathbf{g}_t\|^2] + \frac{1}{2\gamma} \|\mathbf{x}_0 - \mathbf{x}^*\|^2 \\ &\leq \frac{\gamma}{2} B^2 T + \frac{1}{2\gamma} R^2. \end{aligned}$$

Result follows as every time (optimize  $\gamma$ )...



## Convergence Rate Comparison: SGD vs GD

**Classic GD:** For vanilla analysis, we assumed that  $\|\nabla f(\mathbf{x})\|^2 \leq B_{GD}^2$  for all  $\mathbf{x} \in \mathbb{R}^d$  where  $B_{GD}$  was a constant. So for sum-objective:

$$\left\| \frac{1}{n} \sum_i \nabla f_i(\mathbf{x}) \right\|^2 \leq B_{GD}^2 \quad \forall \mathbf{x}.$$

**SGD:** Assuming same for the **expected** squared norms of our stochastic gradients, now called  $B_{SGD}^2$ .

$$\frac{1}{n} \sum_i \|\nabla f_i(\mathbf{x})\|^2 \leq B_{SGD}^2 \quad \forall \mathbf{x}.$$

So by Jensen's inequality for  $\|\cdot\|^2$ .

- $B_{GD}^2 \approx \left\| \frac{1}{n} \sum_i \nabla f_i(\mathbf{x}) \right\|^2 \leq \frac{1}{n} \sum_i \|\nabla f_i(\mathbf{x})\|^2 \approx B_{SGD}^2$ .
- $B_{GD}^2$  can be smaller than  $B_{SGD}^2$ , but often comparable.

Very similar if larger mini-batches are used.

# Tame Strong Convexity: $\mathcal{O}(1/\varepsilon)$ Steps

## Theorem

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be differentiable and strongly convex with parameter  $\mu > 0$ ; let  $\mathbf{x}^*$  be the unique global minimum of  $f$ . With decreasing stepsize

$$\gamma_t := \frac{2}{\mu(t+1)}$$

**stochastic** gradient descent yields:

$$\mathbb{E} \left[ f \left( \frac{2}{T(T+1)} \sum_{t=1}^T t \cdot \mathbf{x}_t \right) - f(\mathbf{x}^*) \right] \leq \frac{2B^2}{\mu(T+1)}$$

where  $B^2 := \max_{t=1}^T \mathbb{E} [\|\mathbf{g}_t\|^2]$ .

Almost same result as for subgradient descent, but in **expectation**.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Tame Strong Convexity: $\mathcal{O}(1/\varepsilon)$ Steps

## Proof

Take expectations over vanilla analysis, **before** summing up (with varying stepsize  $\gamma_t$ ):

$$\mathbb{E} \left[ \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*) \right] = \frac{\gamma_t}{2} \mathbb{E} [ \| \mathbf{g}_t \|^2 ] + \frac{1}{2\gamma_t} \left( \mathbb{E} [ \| \mathbf{x}_t - \mathbf{x}^* \|^2 ] - \mathbb{E} [ \| \mathbf{x}_{t+1} - \mathbf{x}^* \|^2 ] \right).$$

Strong convexity in expectation:

$$\begin{aligned} \mathbb{E} \left[ \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^*) \right] &= \mathbb{E} \left[ \nabla f(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{x}^*) \right] \\ &\geq \mathbb{E} [f(\mathbf{x}_t) - f(\mathbf{x}^*)] + \frac{\mu}{2} \mathbb{E} [ \| \mathbf{x}_t - \mathbf{x}^* \|^2 ] \end{aligned}$$

Putting it together (with  $\mathbb{E}[\| \mathbf{g}_t \|^2] \leq B^2$ ):

$$\mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{x}^*)] \leq \frac{B^2 \gamma_t}{2} + \frac{\gamma_t^{-1} - \mu}{2} \mathbb{E} [ \| \mathbf{x}_t - \mathbf{x}^* \|^2 ] - \frac{\gamma_t^{-1}}{2} \mathbb{E} [ \| \mathbf{x}_{t+1} - \mathbf{x}^* \|^2 ].$$

Proof continues as for subgradient descent, this time with expectations.



Instead of using a single element  $f_i$ , use an average of several of them:

$$\tilde{\mathbf{g}}_t := \frac{1}{m} \sum_{j=1}^m \mathbf{g}_t^j.$$

## Extreme cases:

- $m = 1 \iff$  SGD as originally defined;
- $m = n \iff$  full gradient descent.

**Benefit:** Gradient computation can be naively parallelized.



# Mini-Batch SGD

**Variance intuition:** Taking an average of many independent random variables reduces the variance. So for larger size of the mini-batch  $m$ ,  $\tilde{\mathbf{g}}_t$  will be closer to the true gradient, in expectation:

$$\begin{aligned}\mathbb{E} [\|\tilde{\mathbf{g}}_t - \nabla f(\mathbf{x}_t)\|^2] &:= \mathbb{E} \left[ \left\| \frac{1}{m} \sum_{j=1}^m \mathbf{g}_t^j - \nabla f(\mathbf{x}_t) \right\|^2 \right] \\ &= \frac{1}{m} \mathbb{E} [\|\mathbf{g}_t^1 - \nabla f(\mathbf{x}_t)\|^2] \\ &= \frac{1}{m} \mathbb{E} [\|\mathbf{g}_t^1\|^2] - \frac{1}{m} \|\nabla f(\mathbf{x}_t)\|^2 \leq \frac{B^2}{m}.\end{aligned}$$

Using a modification of the SGD analysis, can use this quantity to relate convergence rate to the rate of full gradient descent.

# To sum up

## Batch gradient descent

Choose  $\mathbf{x}_0 \in \mathbb{R}^d$ .

For  $t = 0, 1, \dots$

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma \nabla f(\mathbf{x}_t) = \mathbf{x}_t - \gamma \frac{1}{N} \sum_{i=1}^N \nabla f_i(\mathbf{x}_t),$$

with learning rate  $\gamma \geq 0$ .



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

## To sum up

### Stochastic gradient descent

Choose  $\mathbf{x}_0 \in \mathbb{R}^d$ .

For  $t = 0, 1, \dots$

sample  $i \in 1, \dots, N$  uniformly at random

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma_t \nabla f_i(\mathbf{x}_t),$$

with learning rate  $\gamma_t \geq 0$ .

### Stochastic gradient descent with mini-batch

Choose  $\mathbf{x}_0 \in \mathbb{R}^d$ .

For  $t = 0, 1, \dots$

Choose a random subset  $\{h_1, \dots, h_m\} \subseteq \{1, \dots, N\}$

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma_t \frac{1}{m} \sum_{j=1}^m \nabla f_{h_j}(\mathbf{x}_t),$$

with learning rate  $\gamma_t \geq 0$ .

# Stochastic Subgradient Descent

For problems which are not necessarily differentiable, we modify SGD to use a subgradient of  $f_i$  in each iteration. The update of stochastic subgradient descent is given by

sample  $i \in [n]$  uniformly at random

let  $\mathbf{g}_t \in \partial f_i(\mathbf{x}_t)$

$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma_t \mathbf{g}_t.$

In other words, we are using an **unbiased estimate of a subgradient** at each step,  $\mathbb{E}[\mathbf{g}_t | \mathbf{x}_t] \in \partial f(\mathbf{x}_t)$ .

Convergence in  $\mathcal{O}(1/\varepsilon^2)$ , by using the **subgradient property** at the beginning of the proof, where convexity was applied.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Constrained Optimization

For constrained optimization, our theorem for the SGD convergence in  $\mathcal{O}(1/\varepsilon^2)$  steps directly extends to constrained problems as well.

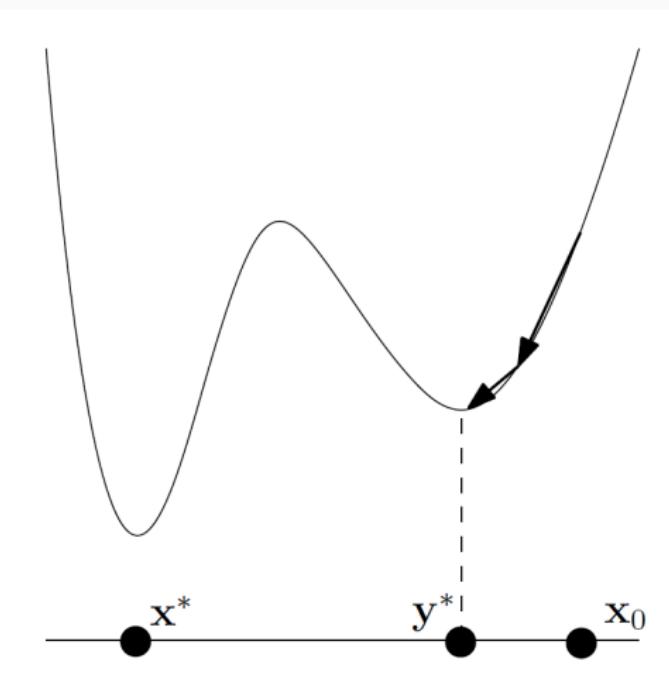
After every step of SGD, projection back to  $X$  is applied as usual. The resulting algorithm is called **projected SGD**.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Gradient Descent in the Non-Convex World

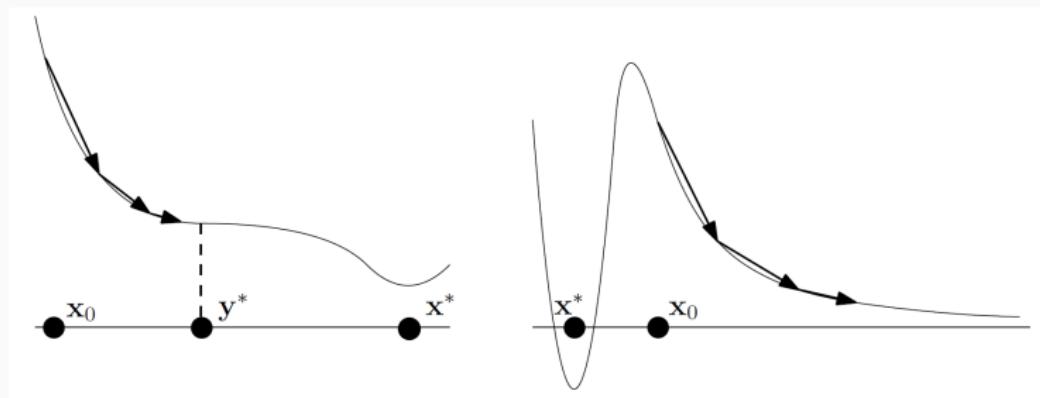
May get stuck in a **local** minimum and miss the global minimum.



# Gradient Descent in the Non-Convex World

Even if there is a **unique** local minimum (equal to the global minimum), we

- may get stuck in a **saddle point**;
- run off to infinity;
- possibly encounter other bad behaviors.



# Gradient Descent in the Non-Convex World

Often, we observe good behavior in practice.

Theoretical explanations mostly missing.

This lecture: under favorable conditions, we sometimes can say something useful about the behavior of gradient descent, even on non-convex functions.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Smooth (but Not Necessarily Convex) Functions

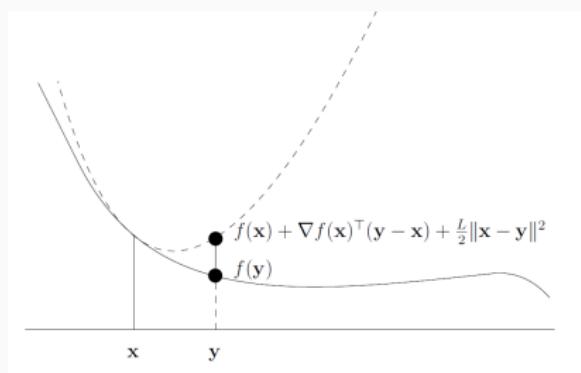
Recall:

## Definition

A differentiable  $f : \text{dom}(f) \rightarrow \mathbb{R}$  is smooth with parameter  $L \in \mathbb{R}_+$  over a convex set  $X \subseteq \text{dom}(f)$  if

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{L}{2} \|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in X.$$

Definition does not require convexity.

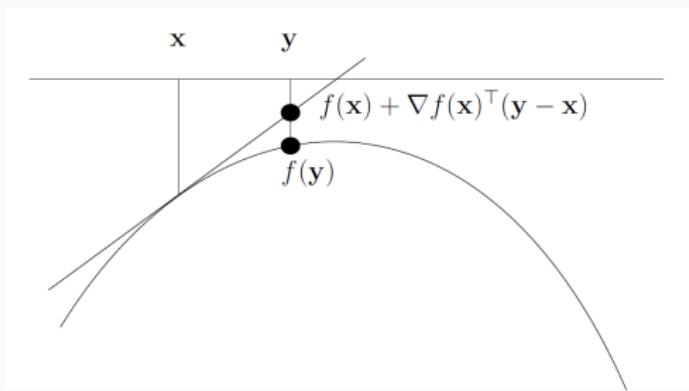


UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Concave Functions

$f$  is called **concave** if  $-f$  is convex.

For all  $x$ , the graph of a differentiable concave function is **below** the tangent hyperplane at  $x$ .



⇒ concave functions are smooth with  $L = 0 \dots$  but boring from an optimization point of view (no global minimum), gradient descent runs off to infinity.

## Lemma

Let  $f : \text{dom}(f) \rightarrow \mathbb{R}$  be twice differentiable, with  $X \subseteq \text{dom}(f)$  a convex set, and  $\|\nabla^2 f(\mathbf{x})\| \leq L$  for all  $\mathbf{x} \in X$ , where  $\|\cdot\|$  is spectral norm. Then  $f$  is smooth with parameter  $L$  over  $X$ .

Examples:

- all quadratic functions  $f(\mathbf{x}) = \mathbf{x}^\top A\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ .
- $f(x) = \sin(x)$  (many global minima).



# Bounded Hessians $\Rightarrow$ Smooth

## Proof (not required)

Bounded Hessians imply Lipschitz continuity of the gradient:

$$||\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})|| \leq L||\mathbf{x} - \mathbf{y}||, \quad \mathbf{x}, \mathbf{y} \in X.$$

To show that this implies smoothness, we use  $h(1) - h(0) = \int_0^1 h'(t)dt$  with

$$h(t) := f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})), \quad t \in [0, 1].$$

Chain rule:

$$h'(t) := \nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))^\top (\mathbf{y} - \mathbf{x}).$$



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Bounded Hessians $\Rightarrow$ Smooth

## Proof (not required)

For  $\mathbf{x}, \mathbf{y} \in X$ :

$$\begin{aligned} & f(\mathbf{x}) - f(\mathbf{y}) - \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) \\ &= h(1) - h(0) - \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) \\ &= \int_0^1 h'(t)dt - \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) \\ &= \int_0^1 \nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))^\top (\mathbf{y} - \mathbf{x})dt - \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) \\ &= \int_0^1 \left( \nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))^\top (\mathbf{y} - \mathbf{x}) - \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) \right) dt \\ &= \int_0^1 \left( \nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - \nabla f(\mathbf{x}) \right)^\top (\mathbf{y} - \mathbf{x}) dt \end{aligned}$$



# Bounded Hessians $\Rightarrow$ Smooth

## Proof (not required)

$$\begin{aligned} &= \int_0^1 \left( \nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - \nabla f(\mathbf{x}) \right)^\top (\mathbf{y} - \mathbf{x}) dt \\ &\leq \int_0^1 \left| \left( \nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - \nabla f(\mathbf{x}) \right)^\top (\mathbf{y} - \mathbf{x}) \right| dt \\ &\leq \int_0^1 \left\| \left( \nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - \nabla f(\mathbf{x}) \right) \right\| \|(\mathbf{y} - \mathbf{x})\| dt \quad (\text{Cauchy-Schwarz}) \\ &\leq \int_0^1 L \|t(\mathbf{y} - \mathbf{x})\| \|(\mathbf{y} - \mathbf{x})\| dt \quad (\text{Lipschitz continuous gradients}) \\ &= \int_0^1 Lt \|\mathbf{x} - \mathbf{y}\|^2 = \frac{L}{2} \|\mathbf{x} - \mathbf{y}\|^2. \end{aligned}$$



# Smooth $\Rightarrow$ Bounded Hessians?

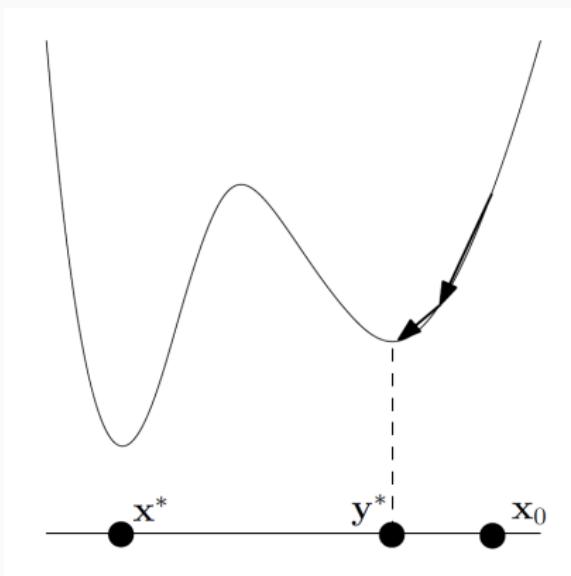
Yes, over any **open** convex set  $X$ .

# Gradient Descent on Smooth Functions

Will prove:  $\|\nabla f(\mathbf{x}_t)\|^2 \rightarrow 0$  for  $t \rightarrow \infty$ ...

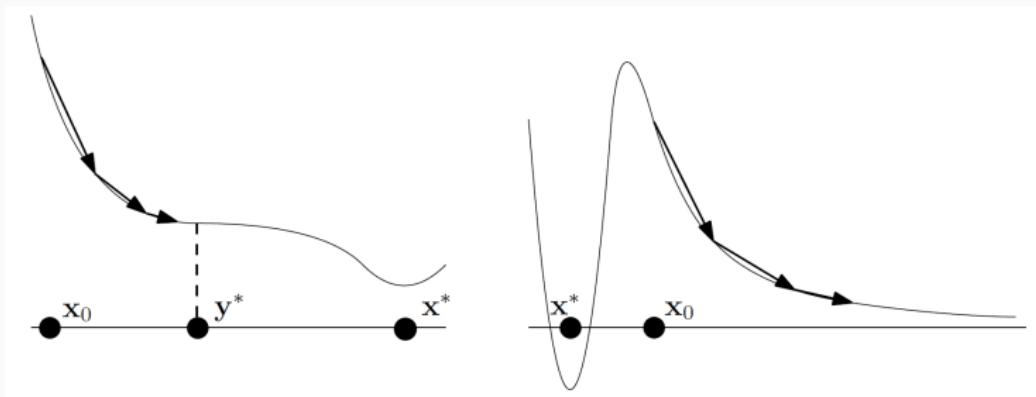
...at the same rate as  $f(\mathbf{x}_t) - f(\mathbf{x}^*) \rightarrow 0$  in the convex case.

$f(\mathbf{x}_t) - f(\mathbf{x}^*)$  itself may **not** converge to 0 in the non-convex case:



# What Does $\|\nabla f(\mathbf{x}_t)\|^2 \rightarrow 0$ Mean?

It may or may not mean that we converge to a **critical point** ( $\nabla f(\mathbf{y}^*) = \mathbf{0}$ ).



# Gradient Descent on Smooth (Not Necessarily Convex) Functions

## Theorem

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be differentiable with a global minimum  $\mathbf{x}^*$ ; furthermore, suppose that  $f$  is smooth with parameter  $L$ . Choosing stepsize

$$\gamma := \frac{1}{L},$$

gradient descent yields

$$\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla f(\mathbf{x}_t)\|^2 \leq \frac{2L}{T} (f(\mathbf{x}_0) - f(\mathbf{x}^*)), \quad T > 0.$$

In particular,  $\|\nabla f(\mathbf{x}_t)\|^2 \leq \frac{2L}{T} (f(\mathbf{x}_0) - f(\mathbf{x}^*))$  for some  $t \in \{0, \dots, T-1\}$ .

And also,  $\lim_{t \rightarrow \infty} \|\nabla f(\mathbf{x}_t)\|^2 = 0$ .



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Gradient Descent on Smooth (Not Necessarily Convex) Functions

## Proof

Sufficient decrease does not require convexity:

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L} \|\nabla f(\mathbf{x}_t)\|^2, \quad t \geq 0.$$

Rewriting:

$$\|\nabla f(\mathbf{x}_t)\|^2 \leq 2L(f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}))$$

Telescoping sum:

$$\sum_{t=0}^{T-1} \|\nabla f(\mathbf{x}_t)\|^2 \leq 2L(f(\mathbf{x}_0) - f(\mathbf{x}_T)) \leq 2L(f(\mathbf{x}_0) - f(\mathbf{x}^*))$$

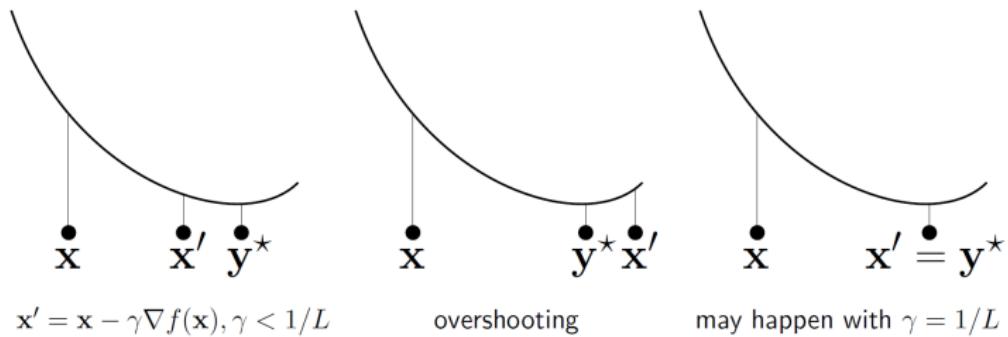
The statement follows (divide by  $T$ ).



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# No Overshooting

In the smooth setting, and with stepsize  $1/L$ , gradient descent cannot **overshoot**, i.e., pass a critical point.



# Optimization - Lesson 7

Francesca Maggioni

---

Academic Year 2023-2024

Department of Management, Information and Production Engineering  
University of Bergamo



# Accelerated Gradient Descent

# First-Order Methods: Less Than $\mathcal{O}(1/\varepsilon)$ Steps?

Maybe gradient descent is not the best possible algorithm?

After all, it is just **some** algorithm that uses gradient information.

## First-order method:

- An algorithm that gains access to  $f$  only via an **oracle** that is able to return values of  $f$  and  $\nabla f$  at arbitrary points.
- Gradient descent is a specific first-order method.

What is the **best** first-order method for smooth convex functions, the one with the smallest upper bound on the number of oracle calls in the worst case?

**Nemirovski and Yudin (1979): every** first-order method needs in the worst case  $\Omega(1/\sqrt{\varepsilon})$  steps (gradient evaluations) in order to achieve an additive error of  $\varepsilon$  on smooth functions.

There is a gap between  $O(1/\varepsilon)$  (gradient descent) and the lower bound!

# Acceleration for Smooth Convex Functions: $\mathcal{O}(1/\sqrt{\varepsilon})$ Steps

**Nesterov (1983):** There is a first-order method that needs only  $\mathcal{O}(1/\sqrt{\varepsilon})$  steps on smooth convex functions, and by the lower bound of Nemirovski and Yudin, this is a best possible algorithm!

The algorithm is known as (Nesterov's) accelerated gradient descent.

A number of (similar) optimal algorithms with other proofs of the  $\mathcal{O}(1/\sqrt{\varepsilon})$  upper bound are known, but there is no well-established 'simplest proof'.

Here: a recent proof based on **potential functions**. Proof is simple but not very instructive.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Nesterov's Accelerated Gradient Descent

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be convex, differentiable, and smooth with parameter  $L$ .

Choose  $\mathbf{z}_0 = \mathbf{y}_0 = \mathbf{x}_0$  arbitrary. For  $t \geq 0$ , set

$$\begin{aligned}\mathbf{y}_{t+1} &:= \mathbf{x}_t - \frac{1}{L} \nabla f(\mathbf{x}_t) \\ \mathbf{z}_{t+1} &:= \mathbf{z}_t - \frac{t+1}{2L} \nabla f(\mathbf{x}_t) \\ \mathbf{x}_{t+1} &:= \frac{t+1}{t+3} \mathbf{y}_{t+1} + \frac{2}{t+3} \mathbf{z}_{t+1}.\end{aligned}$$

- Perform a 'smooth step' from  $\mathbf{x}_t$  to  $\mathbf{y}_{t+1}$ .
- Perform a more aggressive step from  $\mathbf{z}_t$  to  $\mathbf{z}_{t+1}$ .
- Next iterate  $\mathbf{x}_{t+1}$  is a weighted average of  $\mathbf{y}_{t+1}$  and  $\mathbf{z}_{t+1}$ , where we compensate for the more aggressive step by giving  $\mathbf{z}_{t+1}$  a relatively low weight.



# Nesterov's Accelerated Gradient Descent: Error Bound

## Theorem

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be convex, differentiable with a global minimum  $\mathbf{x}^*$ ; furthermore, suppose that  $f$  is **smooth** with parameter  $L$ .

Accelerated gradient descent yields

$$f(\mathbf{y}_T) - f(\mathbf{x}^*) \leq \frac{2L\|\mathbf{z}_0 - \mathbf{x}^*\|^2}{T(T+1)}, \quad T > 0.$$

To reach error at most  $\varepsilon$ , accelerated gradient descent therefore only needs  $O(1/\sqrt{\varepsilon})$  steps instead of  $O(1/\varepsilon)$ .

Recall the bound for **gradient descent**:

$$f(\mathbf{x}_T) - f(\mathbf{x}^*) \leq \frac{L}{2T}\|\mathbf{x}_0 - \mathbf{x}^*\|^2, \quad T > 0.$$



# Nesterov's Accelerated Gradient Descent: The Potential Function

## Proof (Not required)

Idea: assign a **potential**  $\Phi(t)$  to each time  $t$  and show that  $\Phi(t+1) \leq \Phi(t)$ .

Out of the blue: let's define the potential as

$$\Phi(t) := t(t+1) \left( f(\mathbf{y}_t) - f(\mathbf{x}^*) \right) + 2L \|\mathbf{z}_t - \mathbf{x}^*\|^2.$$

If we can show that the potential always decreases, we get

$$\begin{aligned} T(T+1) \left( f(\mathbf{y}_T) - f(\mathbf{x}^*) \right) + 2L \|\mathbf{z}_T - \mathbf{x}^*\|^2 &\leq 2L \|\mathbf{z}_0 - \mathbf{x}^*\|^2 \\ \Phi(T) &\leq \Phi(0). \end{aligned}$$

Rewriting this, we get the claimed error bound.



# Potential Function Decrease: Three Ingredients

## Proof (Not required)

Sufficient decrease for the smooth step from  $\mathbf{x}_t$  to  $\mathbf{y}_{t+1}$ :

$$f(\mathbf{y}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L} \|\nabla f(\mathbf{x}_t)\|^2. \quad (1)$$

Vanilla analysis for the more aggressive step from  $\mathbf{z}_t$  to  $\mathbf{z}_{t+1}$  ( $\gamma = \frac{t+1}{2L}$ ,  $\mathbf{g}_t = \nabla f(\mathbf{x}_t)$ ):

$$\mathbf{g}_t^\top (\mathbf{z}_t - \mathbf{x}^*) = \frac{t+1}{4L} \|\mathbf{g}_t\|^2 + \frac{L}{t+1} \left( \|\mathbf{z}_t - \mathbf{x}^*\|^2 - \|\mathbf{z}_{t+1} - \mathbf{x}^*\|^2 \right). \quad (2)$$

Convexity (graph of  $f$  is above the tangent hyperplane at  $\mathbf{x}_t$ ):

$$f(\mathbf{x}_t) - f(\mathbf{w}) \leq \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{w}), \quad \mathbf{w} \in \mathbb{R}^d. \quad (3)$$



# Potential Function Decrease: Proof

## Proof (Not required)

By definition of potential,

$$\Phi(t+1) := t(t+1)\left(f(\mathbf{y}_{t+1}) - f(\mathbf{x}^*)\right) + 2(t+1)\left(f(\mathbf{y}_{t+1}) - f(\mathbf{x}^*)\right) + 2L\|\mathbf{z}_{t+1} - \mathbf{x}^*\|^2$$
$$\Phi(t) := t(t+1)\left(f(\mathbf{y}_t) - f(\mathbf{x}^*)\right) + 2L\|\mathbf{z}_t - \mathbf{x}^*\|^2.$$

Now, prove that  $\Delta := (\Phi(t+1) - \Phi(t))/(t+1) \leq 0$ :

$$\begin{aligned}\Delta &= t\left(f(\mathbf{y}_{t+1}) - f(\mathbf{y}_t)\right) + 2\left(f(\mathbf{y}_{t+1}) - f(\mathbf{x}^*)\right) + \frac{2L}{t+1} \left(\|\mathbf{z}_{t+1} - \mathbf{x}^*\|^2 - \|\mathbf{z}_t - \mathbf{x}^*\|^2\right) \\ &\stackrel{(2)}{=} t\left(f(\mathbf{y}_{t+1}) - f(\mathbf{y}_t)\right) + 2\left(f(\mathbf{y}_{t+1}) - f(\mathbf{x}^*)\right) + \frac{t+1}{2L}\|\mathbf{g}_t\|^2 - 2\mathbf{g}_t^\top(\mathbf{z}_t - \mathbf{x}^*) \\ &\stackrel{(4)}{\leq} t\left(f(\mathbf{x}_t) - f(\mathbf{y}_t)\right) + 2\left(f(\mathbf{x}_t) - f(\mathbf{x}^*)\right) - \frac{1}{2L}\|\mathbf{g}_t\|^2 - 2\mathbf{g}_t^\top(\mathbf{z}_t - \mathbf{x}^*) \\ &\leq t\left(f(\mathbf{x}_t) - f(\mathbf{y}_t)\right) + 2\left(f(\mathbf{x}_t) - f(\mathbf{x}^*)\right) - 2\mathbf{g}_t^\top(\mathbf{z}_t - \mathbf{x}^*) \\ &\stackrel{(3)}{\leq} t\mathbf{g}_t^\top(\mathbf{x}_t - \mathbf{y}_t) + 2\mathbf{g}_t^\top(\mathbf{x}_t - \mathbf{x}^*) - 2\mathbf{g}_t^\top(\mathbf{z}_t - \mathbf{x}^*) \\ &= \mathbf{g}_t^\top((t+2)\mathbf{x}_t - t\mathbf{y}_t - 2\mathbf{z}_t) \stackrel{(\text{algo})}{=} \mathbf{g}_t^\top \mathbf{0} = 0.\end{aligned}$$



## References

- Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. **A convergence analysis of gradient descent for deep linear neural networks.**
- Nikhil Bansal and Anupam Gupta. **Potential-function proofs for first-order methods.**
- Ching-Pei Lee and Stephen Wright. **First-order algorithms converge faster than  $o(1/k)$  on convex problems.**
- Yurii Nesterov. **A method of solving a convex programming problem with convergence rate  $o(1/k^2)$ .**



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# 1-Dimensional Case: Newton-Raphson Method

Newton-Raphson method is a root-finding algorithm which produces successively better approximations to the roots (or zeroes) of a real-valued function.

**Goal:** find a zero of differentiable  $f : \mathbb{R} \rightarrow \mathbb{R}$ :  $f(x) = 0$ .

**Method:**

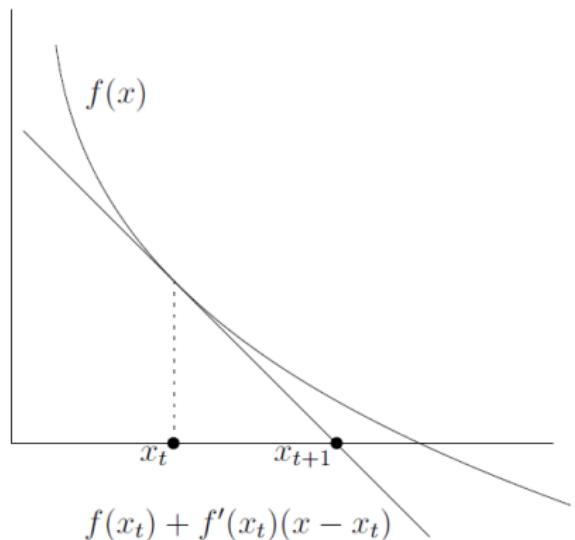
$$f'(x_t) = \frac{f(x_{t+1}) - f(x_t)}{x_{t+1} - x_t}$$

$$f(x_{t+1}) = 0$$

$$x_{t+1} := x_t - \frac{f(x_t)}{f'(x_t)}, \quad t \geq 0.$$

$x_{t+1}$  solves:

$$f(x_t) + f'(x_t)(x - x_t) = 0.$$



# Newton's Method for Optimization

**1-dimensional case:** Find a global minimum  $x^*$  of a differentiable convex function  $f : \mathbb{R} \rightarrow \mathbb{R}$ .

Can equivalently search for a zero of the derivative  $f'$ : Apply the Newton-Raphson method to  $f'$ .

Update step:

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

(needs  $f$  **twice** differentiable).

**d-dimensional case:** Newton's method for minimizing a convex function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$ :

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \nabla^2 f(\mathbf{x}_t)^{-1} \nabla f(\mathbf{x}_t).$$



# Newton's Method = Adaptive Gradient Descent

General update scheme:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - H(\mathbf{x}_t) \nabla f(\mathbf{x}_t),$$

where  $H(\mathbf{x}) \in \mathbb{R}^{d \times d}$  is some matrix.

Newton's method:  $H = \nabla^2 f(\mathbf{x}_t)^{-1}$ .

Gradient descent:  $H = \gamma I$ .

Newton's method: 'adaptive gradient descent', adaptation is w.r.t. the local geometry of the function at  $\mathbf{x}_t$ .



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Convergence in One Step on Quadratic Functions

A **nondegenerate** quadratic function is a function of the form

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top M\mathbf{x} - \mathbf{q}^\top \mathbf{x} + c,$$

where  $M \in \mathbb{R}^{d \times d}$  is an invertible symmetric matrix,  $\mathbf{q} \in \mathbb{R}^d$ ,  $c \in \mathbb{R}$ . Let  $\mathbf{x}^* = M^{-1}\mathbf{q}$  be the unique solution of  $\nabla f(\mathbf{x}) = \mathbf{0}$ .

- $\mathbf{x}^*$  is the unique global minimum if  $f$  is convex.

## Lemma

On nondegenerate quadratic functions, with any starting point  $\mathbf{x}_0 \in \mathbb{R}^d$ , Newton's method yields  $\mathbf{x}_1 = \mathbf{x}^*$ .

## Proof

We have  $\nabla f(\mathbf{x}) = M\mathbf{x} - \mathbf{q}$  (this implies  $\mathbf{x}^* = M^{-1}\mathbf{q}$ ) and  $\nabla^2 f(\mathbf{x}) = M$ . Hence:

$$\mathbf{x}_1 = \mathbf{x}_0 - \nabla^2 f(\mathbf{x}_0)^{-1} \nabla f(\mathbf{x}_0) = \mathbf{x}_0 - M^{-1} (M\mathbf{x}_0 - \mathbf{q}) = M^{-1} \mathbf{q} = \mathbf{x}^*.$$

# Minimizing the Second-Order Taylor Approximation

Alternative interpretation of Newton's method:

Each step minimizes the local **second-order Taylor approximation**.

## Lemma

Let  $f$  be convex and twice differentiable at  $\mathbf{x}_t \in \text{dom}(f)$ , with  $\nabla^2 f(\mathbf{x}_t) \succ 0$  being invertible. The vector  $\mathbf{x}_{t+1}$  resulting from the Netwon step satisfies:

$$\mathbf{x}_{t+1} = \arg \min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^\top (\mathbf{x} - \mathbf{x}_t) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_t)^\top \nabla^2 f(\mathbf{x}_t) (\mathbf{x} - \mathbf{x}_t).$$



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Local Convergence

We will prove: under suitable conditions, and starting close to the global minimum, Newton's method will reach distance at most  $\varepsilon$  to the minimum within  $\log \log(1/\varepsilon)$  steps.

- much faster than anything we have seen so far...
- ...but we need to start close to the minimum already.

This is a **local convergence** result.

**Global convergence** results that hold for every starting point were unknown for Newton's method until very recently.



# Once you're close, you're there...

## Theorem

Let  $f : \text{dom}(f) \rightarrow \mathbb{R}$  be **convex** with a unique global minimum  $\mathbf{x}^*$ . Suppose there is a ball  $X \subseteq \text{dom}(f)$  with center  $\mathbf{x}^*$ , s.t.

1. **Bounded inverse Hessians:** There exists a real number  $\mu > 0$  such that:

$$\|\nabla^2 f(\mathbf{x})^{-1}\| \leq \frac{1}{\mu}, \quad \forall \mathbf{x} \in X.$$

2. **Lipschitz continuous Hessians:** There exists a real number  $B > 0$  such that

$$\|\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{y})\| \leq B\|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in X.$$

Then, for  $\mathbf{x}_t \in X$  and  $\mathbf{x}_{t+1}$  resulting from the Newton step, we have:

$$\|\mathbf{x}_{t+1} - \mathbf{x}^*\| \leq \frac{B}{2\mu} \|\mathbf{x}_t - \mathbf{x}^*\|^2.$$



## Corollary

With the assumptions and terminology of the convergence theorem, and if

$$\|\mathbf{x}_0 - \mathbf{x}^*\| \leq \frac{\mu}{B},$$

then Newton's method yields

$$\|\mathbf{x}_T - \mathbf{x}^*\| \leq \frac{\mu}{B} \left(\frac{1}{2}\right)^{2^T - 1}, \quad T \geq 0.$$

Starting close to the global minimum, we will reach distance at most  $\varepsilon$  to the minimum within  $\mathcal{O}(\log \log(1/\varepsilon))$  steps.

Almost constant Hessians close to optimality...

...so  $f$  behaves almost like a quadratic function which has truly constant Hessians and allows Newton's method to converge in one step.



# Proof of Convergence Theorem

## Proof

We abbreviate  $H := \nabla^2 f$ ,  $\mathbf{x} = \mathbf{x}_t$ ,  $\mathbf{x}' = \mathbf{x}_{t+1}$ . Subtracting  $\mathbf{x}^*$  from both sides of the Newton step definition:

$$\begin{aligned}\mathbf{x}' - \mathbf{x}^* &= \mathbf{x} - \mathbf{x}^* - H(\mathbf{x})^{-1} \nabla f(\mathbf{x}) \\ &= \mathbf{x} - \mathbf{x}^* + H(\mathbf{x})^{-1} (\nabla f(\mathbf{x}^*) - \nabla f(\mathbf{x})) \\ &= \mathbf{x} - \mathbf{x}^* + H(\mathbf{x})^{-1} \int_0^1 H(\mathbf{x} + t(\mathbf{x}^* - \mathbf{x}))(\mathbf{x}^* - \mathbf{x}) dt\end{aligned}$$

using the fundamental theorem of calculus

$$\int_a^b h'(t) dt = h(b) - h(a)$$

with

$$\begin{aligned}h(t) &= \nabla f(\mathbf{x} + t(\mathbf{x}^* - \mathbf{x})) \\ h'(t) &= \nabla^2 f(\mathbf{x} + t(\mathbf{x}^* - \mathbf{x}))(\mathbf{x}^* - \mathbf{x}).\end{aligned}$$

# Proof of Convergence Theorem

## Proof

We so far have:

$$\mathbf{x}' - \mathbf{x}^* = \mathbf{x} - \mathbf{x}^* + H(\mathbf{x})^{-1} \int_0^1 H(\mathbf{x} + t(\mathbf{x}^* - \mathbf{x}))(\mathbf{x}^* - \mathbf{x}) dt.$$

With

$$\mathbf{x} - \mathbf{x}^* = H(\mathbf{x})^{-1} H(\mathbf{x})(\mathbf{x} - \mathbf{x}^*) = H(\mathbf{x})^{-1} \int_0^1 -H(\mathbf{x})(\mathbf{x}^* - \mathbf{x}) dt,$$

we further get

$$\mathbf{x}' - \mathbf{x}^* = H(\mathbf{x})^{-1} \int_0^1 (H(\mathbf{x} + t(\mathbf{x}^* - \mathbf{x})) - H(\mathbf{x})) (\mathbf{x}^* - \mathbf{x}) dt.$$

Taking norms, we have

$$\|\mathbf{x}' - \mathbf{x}^*\| \leq \|H(\mathbf{x})^{-1}\| \cdot \left\| \int_0^1 (H(\mathbf{x} + t(\mathbf{x}^* - \mathbf{x})) - H(\mathbf{x})) (\mathbf{x}^* - \mathbf{x}) dt \right\|$$

because  $\|Ay\| \leq \|A\| \cdot \|y\|$  for any  $A, y$  (by definition of spectral norm).

# Proof of Convergence Theorem

## Proof

We so far have:

$$\begin{aligned} \|\mathbf{x}' - \mathbf{x}^*\| &\leq \|H(\mathbf{x})^{-1}\| \cdot \left\| \int_0^1 (H(\mathbf{x} + t(\mathbf{x}^* - \mathbf{x})) - H(\mathbf{x})) (\mathbf{x}^* - \mathbf{x}) dt \right\| \\ &\leq \|H(\mathbf{x})^{-1}\| \cdot \int_0^1 \|(H(\mathbf{x} + t(\mathbf{x}^* - \mathbf{x})) - H(\mathbf{x})) (\mathbf{x}^* - \mathbf{x})\| dt \\ &\leq \|H(\mathbf{x})^{-1}\| \cdot \int_0^1 \|H(\mathbf{x} + t(\mathbf{x}^* - \mathbf{x})) - H(\mathbf{x})\| \cdot \|\mathbf{x}^* - \mathbf{x}\| dt \\ &= \|H(\mathbf{x})^{-1}\| \cdot \|\mathbf{x}^* - \mathbf{x}\| \int_0^1 \|H(\mathbf{x} + t(\mathbf{x}^* - \mathbf{x})) - H(\mathbf{x})\| dt. \end{aligned}$$

We can now use the properties (1) and (2) (bounded inverse Hessians, Lipschitz continuous Hessians) to conclude that:

$$\begin{aligned} \|\mathbf{x}' - \mathbf{x}^*\| &\leq \frac{1}{\mu} \|\mathbf{x}^* - \mathbf{x}\| \int_0^1 B \|t(\mathbf{x}^* - \mathbf{x})\| dt \\ &= \frac{B}{\mu} \|\mathbf{x}^* - \mathbf{x}\|^2 \int_0^1 t dt = \frac{B}{2\mu} \|\mathbf{x}^* - \mathbf{x}\|^2. \end{aligned}$$

## Strong Convexity $\Rightarrow$ Bounded inverse Hessians

One way to ensure bounded inverse Hessians is to require strong convexity over  $X$ .

### Lemma

Let  $f : \text{dom}(f) \rightarrow \mathbb{R}$  be twice differentiable and strongly convex with parameter  $\mu$  over an open convex subset  $X \subseteq \text{dom}(f)$  meaning that

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{\mu}{2} \|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in X.$$

Then  $\nabla^2 f(\mathbf{x})$  is invertible and  $\|\nabla^2 f(\mathbf{x})^{-1}\| \leq \frac{1}{\mu}$  for all  $\mathbf{x} \in X$ , where  $\|\cdot\|$  is the spectral norm.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Downside of Newton's Method

**Computational bottleneck** in each step:

- compute and invert the **Hessian matrix**
- or solve the linear system  $\nabla^2 f(\mathbf{x}_t) \Delta \mathbf{x} = -\nabla f(\mathbf{x}_t)$  for the next step  $\Delta \mathbf{x}$ .

Matrix/system has size  $d \times d$ , taking up to  $\mathcal{O}(d^3)$  time to invert / solve.

In many applications,  $d$  is large...



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# The Secant Method

Another iterative method for finding zeros in dimension 1.

Start from Newton-Raphson step:

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

Use **finite difference approximation**<sup>1</sup> of  $f'(x_t)$ :

$$f'(x_t) \approx \frac{f(x_t) - f(x_{t-1})}{x_t - x_{t-1}}.$$

Obtain the **secant method**:

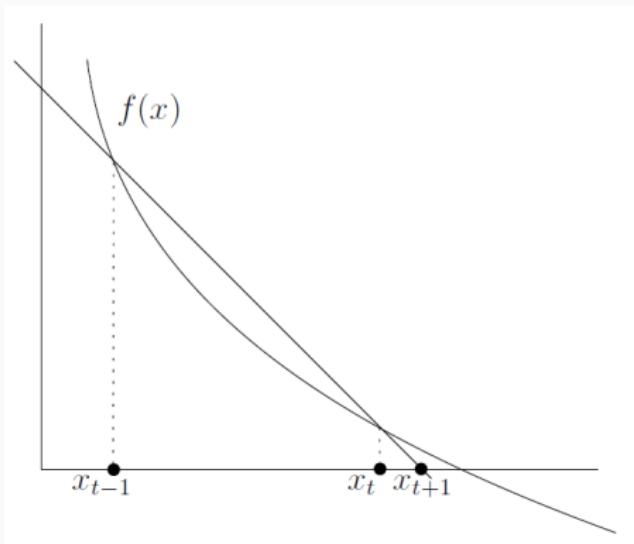
$$x_{t+1} := x_t - f(x_t) \frac{x_t - x_{t-1}}{f(x_t) - f(x_{t-1})}.$$

---

<sup>1</sup>For small  $|x_t - x_{t-1}|$ .



# The Secant Method



- construct the line through the two points  $(x_{t-1}, f(x_{t-1}))$  and  $(x_t, f(x_t))$ ;
- next iterate  $x_{t+1}$  is where this line intersects the  $x$ -axis.

# The Secant Method

We now have a derivative-free version of the Newton-Raphson method.

**Secant method for optimization:** Can we also **optimize** a differentiable univariate function  $f$ ? Yes, apply the secant method to  $f'$ :

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

→ a **second-derivative-free** version of Newton's method for optimization.

Can we generalize this to higher dimensions to obtain a **Hessian-free** version of Newton's method on  $\mathbb{R}^d$ ?



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# The Secant Condition

Apply finite difference approximation to  $f''$  (still 1-dim):

$$H_t := \frac{f'(x_t) - f'(x_{t-1})}{x_t - x_{t-1}} \approx f''(x_t) \iff f'(x_t) - f'(x_{t-1}) = H_t(x_t - x_{t-1}),$$

the **secant condition**.

- Newton's method:  $x_{t+1} := x_t - f''(x_t)^{-1} f'(x_t)$ .
- Secant method:  $x_{t+1} := x_t - H_t^{-1} f'(x_t)$ .

In higher dimensions: Let  $H_t \in \mathbb{R}^{d \times d}$  be a symmetric matrix satisfying the  **$d$ -dimensional secant condition**

$$\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}) = H_t(\mathbf{x}_t - \mathbf{x}_{t-1}).$$

The secant method step then becomes

$$\mathbf{x}_{t+1} := \mathbf{x}_t - H_t^{-1} \nabla f(\mathbf{x}_t). \tag{4}$$



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Quasi-Newton Methods

- Newton:  $\mathbf{x}_{t+1} := \mathbf{x}_t - \nabla^2 f(\mathbf{x}_t)^{-1} \nabla f(\mathbf{x}_t)$ .
- Secant:  $\mathbf{x}_{t+1} := \mathbf{x}_t - H_t^{-1} \nabla f(\mathbf{x}_t)$ , where  
 $\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}) = H_t(\mathbf{x}_t - \mathbf{x}_{t-1})$ .

If  $f$  is twice differentiable, secant condition and first-order approximation of  $\nabla f(\mathbf{x})$  at  $\mathbf{x}_t$  yield:

$$\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}) = H_t(\mathbf{x}_t - \mathbf{x}_{t-1}) \approx \nabla^2 f(\mathbf{x}_t)(\mathbf{x}_t - \mathbf{x}_{t-1}).$$

Might therefore hope that  $H_t \approx \nabla^2 f(\mathbf{x}_t) \dots$

...meaning that the secant method approximates Newton's method.

- $d = 1$ : unique number  $H_t$  satisfying the secant condition.
- $d > 1$ : Secant condition  $\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}) = H_t(\mathbf{x}_t - \mathbf{x}_{t-1})$  has infinitely many symmetric solutions  $H_t$  (underdetermined linear system).

Any scheme of choosing in each step of the secant method a **symmetric**  $H_t$  that satisfies the secant condition defines a **Quasi-Newton method**.

# Quasi-Newton Methods

- Newton's method is a Quasi-Newton method if and only if  $f$  is a nondegenerate quadratic function.
- Hence, Quasi-Newton methods do not generalize Newton's method but form a family of related algorithms.
- The first Quasi-Newton method was developed by William C. Davidon in 1956; he desperately needed iterations that were faster than those of Newton's method in order obtain results in the short time spans between expected failures of the room-sized computer that he used to run his computations on.
- But the paper he wrote about his new method got rejected for lacking a convergence analysis, and for allegedly dubious notation. It became a very influential Technical Report in 1959 and was finally officially published in 1991, with a foreword giving the historical context. Ironically, Quasi-Newton methods are today the methods of choice in a number of relevant machine learning applications.
- Here: no convergence analysis (for a change), we focus on development of algorithms from first principles.

# Developing a Quasi-Newton Method

For efficiency reasons (want to avoid matrix inversions!), directly deal with the inverse matrices  $H_t^{-1}$ .

Given: iterates  $\mathbf{x}_{t-1}$ ,  $\mathbf{x}_t$  as well as the matrix  $H_{t-1}^{-1}$ .

Wanted: next matrix  $H_t^{-1}$  needed in next Quasi-Newton step

$$\mathbf{x}_{t+1} := \mathbf{x}_t - H_t^{-1} \nabla f(\mathbf{x}_t).$$

How should we choose  $H_t^{-1}$ .

Newton's method:  $\nabla^2 f(\mathbf{x}_t)$  fluctuates only very little in the region of extremely fast convergence.

Hence, in a Quasi-Newton method, it also makes sense to have that

$H_t \approx H_{t-1}$ , or  $H_t^{-1} \approx H_{t-1}^{-1}$ .



# Greenstadt's Family of Quasi-Newton Methods

Given: iterates  $\mathbf{x}_{t-1}$ ,  $\mathbf{x}_t$  as well as the matrix  $H_{t-1}^{-1}$ .

Wanted: next matrix  $H_t^{-1}$  needed in next Quasi-Newton step

$$\mathbf{x}_{t+1} := \mathbf{x}_t - H_t^{-1} \nabla f(\mathbf{x}_t).$$

Greenstadt: Update

$$H_t^{-1} := H_{t-1}^{-1} + E_t,$$

$E_t$  an error matrix.

Try to minimize the error subject to  $H_t$  satisfying the secant condition!

Simple error measure: Frobenius norm

$$\|E\|_F^2 := \sum_{i=1}^d \sum_{j=1}^d E_{ij}^2.$$



# Greenstadt's Family of Quasi-Newton Methods

Greenstadt: minimizing  $\|E\|_F$  gives just one method, this is 'too specialized'.

Greenstadt searched for a compromise between variability in the method and simplicity of the resulting formulas.

More general error measure

$$\|AEA^\top\|_F^2,$$

where  $A \in \mathbb{R}^{d \times d}$  is some fixed invertible transformation matrix.

$A = I$ : squared Frobenius norm of  $E$ , the 'specialized' method.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# The Greenstadt Update: $H_{t-1}^{-1} \rightarrow H_t^{-1}$

Secant condition in terms of  $H_t^{-1}$ :

$$H_t^{-1} \left( \nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}) \right) = (\mathbf{x}_t - \mathbf{x}_{t-1}).$$

Fix  $t$  and simplify notation:

$H$	$:=$	$H_{t-1}^{-1}$	(old inverse)
$H'$	$:=$	$H_t^{-1}$	(new inverse)
$E$	$:=$	$E_t$	(error matrix)
$\sigma$	$:=$	$\mathbf{x}_t - \mathbf{x}_{t-1}$	(step in solutions)
$\mathbf{y}$	$=$	$\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1})$	(step in gradients)
$\mathbf{r}$	$=$	$\sigma - H\mathbf{y}$	(error of old inverse in secant condition)

The update formula is:

$$H' = H + E.$$

Secant condition becomes:

$$H'\mathbf{y} = \sigma \quad (\iff E\mathbf{y} = \mathbf{r}).$$

## The Greenstadt Update: $H_{t-1}^{-1} \rightarrow H_t^{-1}$

Minimizing the error becomes a convex constrained minimization problem in the  $d^2$  variables  $E_{ij}$ :

$$\begin{aligned} \min \quad & \frac{1}{2} \|AEA^\top\|_F^2 \quad (\text{error function}) \\ \text{s.t.} \quad & E\mathbf{y} = \mathbf{r} \quad (\text{secant condition}) \\ & E^\top - E = 0 \quad (\text{symmetry}) \end{aligned}$$

Don't need to solve it computationally (for numbers  $E_{ij}$ )...

...but mathematically (formula for  $E$ ).

Minimize **convex quadratic** function subject to **linear equations**  $\rightarrow$  analytic formula for the minimizer from the **method of Lagrange multipliers**.



# References

- William C. Davidon. **Variable metric method for minimization..**
- J. Greenstadt. **Variations on variable-metric methods..**
- Sai Praneeth Karimireddy, Sebastian U Stich, and Martin Jaggi. **Global linear convergence of Newton's method without strong-convexity or Lipschitz gradients..**



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# **Optimization - Lesson 8**

Francesca Maggioni

---

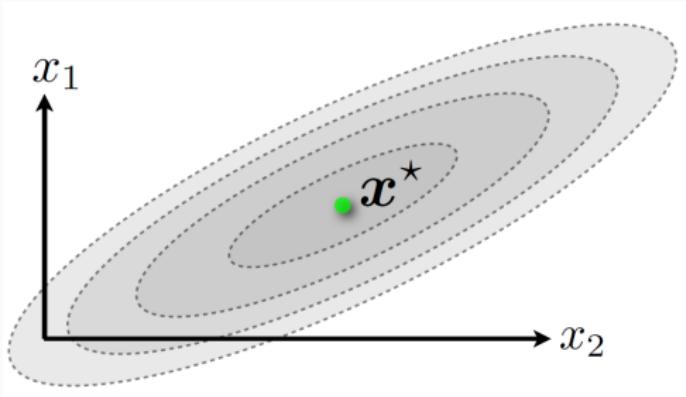
Academic Year 2023-2024

Department of Management, Information and Production Engineering  
University of Bergamo



# Coordinate Descent

**Goal:** Find  $\mathbf{x}^* \in \mathbf{R}^d$  minimizing  $f(\mathbf{x})$ .

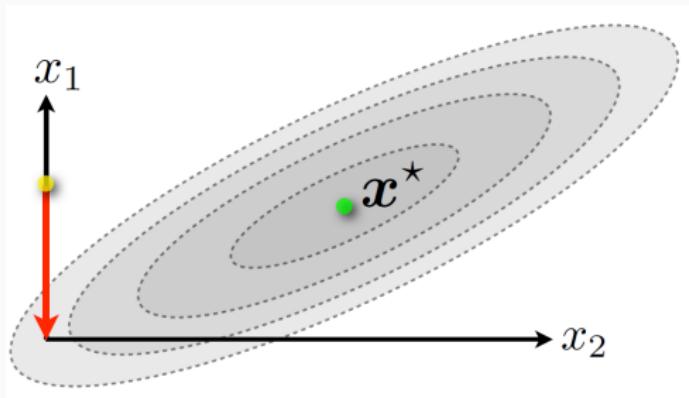


Example:  $d = 2$ .

**Idea:** Update one coordinate at a time, while keeping others fixed.

# Coordinate Descent

**Goal:** Find  $\mathbf{x}^* \in \mathbf{R}^d$  minimizing  $f(\mathbf{x})$ .

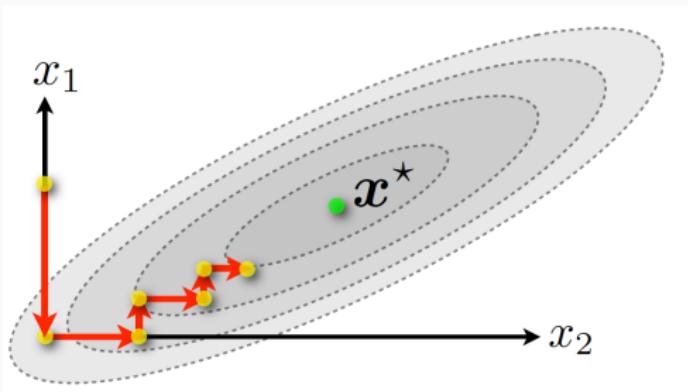


Example:  $d = 2$ .

**Idea:** Update one coordinate at a time, while keeping others fixed.

# Coordinate Descent

**Goal:** Find  $\mathbf{x}^* \in \mathbf{R}^d$  minimizing  $f(\mathbf{x})$ .



Example:  $d = 2$ .

**Idea:** Update one coordinate at a time, while keeping others fixed.

# Coordinate Descent

Modify only one coordinate per step:

select  $i_t \in \{1, \dots, d\}$

$\mathbf{x}_{t+1} := \mathbf{x}_t + \gamma \mathbf{e}_{i_t}.$

Two main variants:

1. Gradient-based step-size:

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \frac{1}{L} \nabla_{i_t} f(\mathbf{x}_t) \mathbf{e}_{i_t}.$$

2. Exact coordinate minimization: solve the single-variable minimization

$$\arg \min_{\gamma \in \mathbb{R}} f(\mathbf{x}_t + \gamma \mathbf{e}_{i_t})$$

in closed form.

# Randomized Coordinate Descent

select  $i_t \in \{1, \dots, d\}$  uniformly at random

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \frac{1}{L} \nabla_{i_t} f(\mathbf{x}_t) \mathbf{e}_{i_t}.$$

1. Faster convergence than gradient descent  
(if coordinate step is significantly cheaper than full gradient step)

# Convergence Analysis

Assume coordinate-wise smoothness:

$$f(\mathbf{x} + \gamma \mathbf{e}_i) \leq f(\mathbf{x}) + \gamma \nabla_i f(\mathbf{x}) + \frac{L}{2} \gamma^2 \quad \forall \mathbf{x} \in \mathbb{R}^d, \forall \gamma \in \mathbb{R}, \forall i.$$

Is equivalent to coordinate-wise Lipschitz gradient:

$$|\nabla_i f(\mathbf{x} + \gamma \mathbf{e}_i) - \nabla_i f(\mathbf{x})| \leq L|\gamma| \quad \forall \mathbf{x} \in \mathbb{R}^d, \forall \gamma \in \mathbb{R}, \forall i$$

→ Additionally assume **strong convexity**.

## Theorem

Let  $f$  be coordinate-wise smooth with constant  $L$ , and strongly convex with parameter  $\mu > 0$ . Then, coordinate descent with a step-size of  $1/L$ ,

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \frac{1}{L} \nabla_{i_t} f(\mathbf{x}_t) \mathbf{e}_{i_t}$$

when choosing the active coordinate  $i_t$  uniformly at random, has an expected linear convergence rate of

$$\mathbb{E} [f(\mathbf{x}_t) - f^*] \leq \left(1 - \frac{\mu}{dL}\right)^t [f(\mathbf{x}_0) - f^*].$$



# Convergence Proof

## Proof

Plugging the update rule, into the smoothness condition, we have

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L} |\nabla_{i_t} f(\mathbf{x}_t)|^2.$$

Take expectation with respect to  $i_t$ :

$$\begin{aligned}\mathbb{E}[f(\mathbf{x}_{t+1})] &\leq f(\mathbf{x}_t) - \frac{1}{2L} \mathbb{E}[|\nabla_{i_t} f(\mathbf{x}_t)|^2] \\ &= f(\mathbf{x}_t) - \frac{1}{2L} \frac{1}{d} \sum_i |\nabla_i f(\mathbf{x}_t)|^2 \\ &= f(\mathbf{x}_t) - \frac{1}{2dL} \|\nabla f(\mathbf{x}_t)\|^2.\end{aligned}$$

[**Lemma:** strongly convex  $f$  satisfy **PL:**  $\frac{1}{2} \|\nabla f(\mathbf{x})\|^2 \geq \mu(f(\mathbf{x}) - f^*) \quad \forall \mathbf{x}.$ ]

Subtracting  $f^*$  from both sides, we therefore obtain

$$\mathbb{E}[f(\mathbf{x}_{t+1}) - f^*] \leq \left(1 - \frac{\mu}{dL}\right) [f(\mathbf{x}_t) - f^*].$$

# The Polyak-Lojasiewicz Condition

## Definition

$f$  satisfies the *Polyak-Lojasiewicz Inequality* (PL) if the following holds for some  $\mu > 0$

$$\frac{1}{2} \|\nabla f(\mathbf{x})\|^2 \geq \mu(f(\mathbf{x}) - f^*), \quad \forall \mathbf{x}.$$

## Lemma (Strong Convexity $\Rightarrow$ PL)

Let  $f$  be strongly convex with parameter  $\mu > 0$ . Then  $f$  satisfies PL for the same  $\mu$ .

## Proof

For all  $\mathbf{x}$  and  $\mathbf{y}$  we have

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{\mu}{2} \|\mathbf{y} - \mathbf{x}\|^2.$$

Minimizing each side of the inequality with respect to  $\mathbf{y}$  we obtain

$$f(\mathbf{x}^*) \geq f(\mathbf{x}) - \frac{1}{2\mu} \|\nabla f(\mathbf{x})\|^2.$$

# Linear Convergence without Strong Convexity

## Examples satisfying PL:

- $f(\mathbf{x}) := g(A\mathbf{x})$  for strongly convex  $g$  and arbitrary matrix  $A$ , including least squares regression and many other applications in machine learning.

Linear convergence for all  $f$  satisfying the PL condition:

### Corollary

For minimization of a function  $f$  which is coordinate-wise smooth with constant  $L$ , satisfies the PL inequality, and has a non-empty solution set  $\mathcal{X}^*$ , random coordinate descent with a step-size of  $1/L$  has the expected linear convergence rate of

$$\mathbb{E}[f(\mathbf{x}_t) - f^*] \leq \left(1 - \frac{\mu}{dL}\right)^t [f(\mathbf{x}_0) - f^*].$$



# Importance Sampling

Uniformly random selection is not always best!

- ▷ individual smoothness constants  $L_i$  for each coordinate  $i$

$$f(\mathbf{x} + \gamma \mathbf{e}_i) \leq f(\mathbf{x}) + \gamma \nabla_i f(\mathbf{x}) + \frac{L_i}{2} \gamma^2.$$

Coordinate descent using this modified selection probabilities

$$P[i_t = i] = \frac{L_i}{\sum_i L_i}$$

and using a step-size of  $1/L_{i_t}$  converges with the faster rate of

$$\mathbb{E} [f(\mathbf{x}_t) - f^*] \leq \left(1 - \frac{\mu}{d\bar{L}}\right)^t [f(\mathbf{x}_0) - f^*],$$

where  $\bar{L} = \frac{1}{d} \sum_{i=1}^d L_i$ .

Often:  $\bar{L} \ll L = \max_i L_i$ !

# Steepest Coordinate Descent

## Coordinate Selection Rule

$$i_t := \arg \max_{i \in [d]} |\nabla_i f(\mathbf{x}_t)|.$$

“Greedy” or steepest coordinate descent.

Deterministic vs random.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Convergence of Steepest Coordinate Descent

Has same convergence rate as for random coordinate descent!

Use

$$\max_i |\nabla_i f(\mathbf{x})|^2 \geq \frac{1}{d} \sum_i |\nabla_i f(\mathbf{x})|^2.$$

(And: algorithm is deterministic, so no need to take expectations in the proof.)

## Corollary

Steepest coordinate descent with a step-size of  $1/L$  has the linear convergence rate of

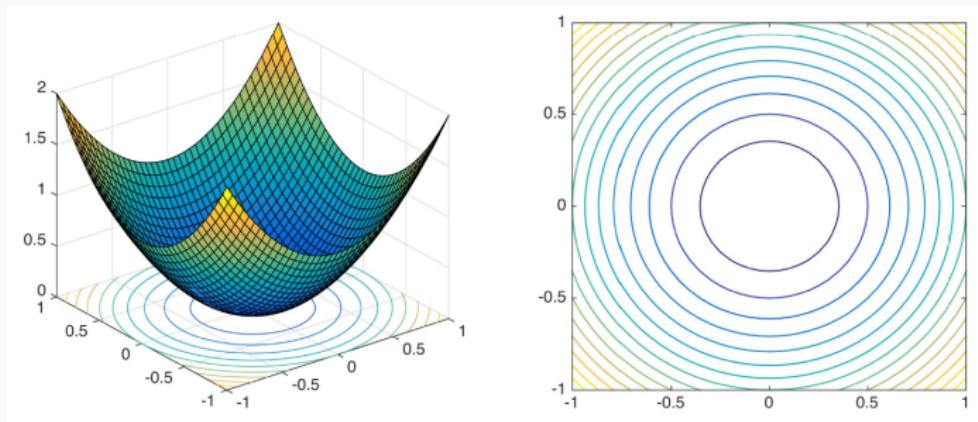
$$\mathbb{E}[f(\mathbf{x}_t) - f^*] \leq \left(1 - \frac{\mu}{dL}\right)^t [f(\mathbf{x}_0) - f^*].$$



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Non-Smooth Objectives

Have proved everything for smooth  $f$ . What about **non-smooth**?

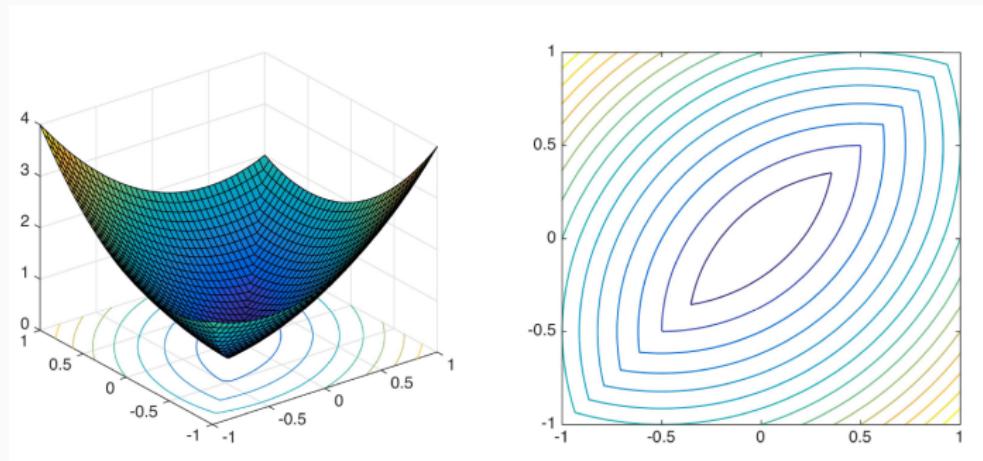


A smooth function:  $f(\mathbf{x}) := \|\mathbf{x}\|^2$ .



# Non-Smooth Objectives

For general non-smooth  $f$ , coordinate descent **fails**: gets permanently stuck:



A non-smooth function:  $f(\mathbf{x}) := \|\mathbf{x}\|^2 + |x_1 - x_2|$ .

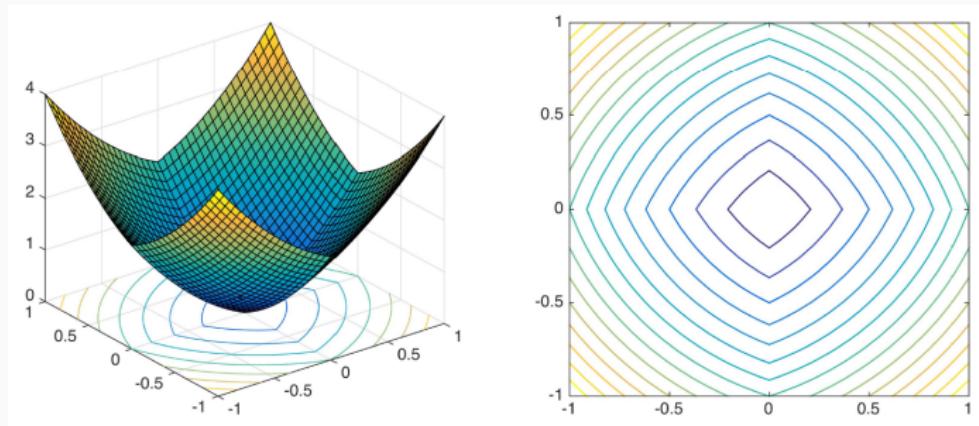


# Non-Smooth Objectives

What if the non-smooth part is separable over the coordinates?

$$f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x}) \quad \text{with} \quad h(\mathbf{x}) = \sum_i h_i(x_i),$$

→ global convergence!



A non-smooth but separable function:  $f(\mathbf{x}) := \|\mathbf{x}\|^2 + \|\mathbf{x}\|_1$ .

## 1. Random coordinate descent

- ◊ is state-of-the-art for generalized linear models  $f(\mathbf{x}) := g(A\mathbf{x}) + \sum_i h_i(x_i)$ .  
Regression, classification (with different regularizers)

## 2. Steepest coordinate descent

- ◊ Training with the help of GPUs  
(or other hardware of limited memory):  
Use steepest coordinates to decide which subset of the data A to put onto the GPU.  
→ DuHL algorithm used by IBM & NVIDIA. *link1*, *link2*.



# Optimization - Lesson 9

Francesca Maggioni

---

Academic Year 2023-2024

Department of Management, Information and Production Engineering  
University of Bergamo



# No gradients

Can we optimize  $\min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x})$  if without access to gradients?

## Random search

pick a random direction  $\mathbf{d}_t \in \mathbb{R}^d$

$$\gamma := \arg \min_{\gamma \in \mathbb{R}} f(\mathbf{x}_t + \gamma \mathbf{d}_t) \quad (\text{line-search})$$

$$\mathbf{x}_{t+1} := \mathbf{x}_t + \gamma \mathbf{d}_t.$$



# Convergence rate for derivative-free random search

Converges same as gradient descent - up to a slow-down factor  $d$ .

## Proof

Assume that  $f$  is a  $L$ -smooth convex, differentiable function. For any  $\gamma$ , by smoothness, we have:

$$f(\mathbf{x}_t + \gamma \mathbf{d}_t) \leq f(\mathbf{x}_t) + \gamma \langle \mathbf{d}_t, \nabla f(\mathbf{x}_t) \rangle + \frac{\gamma^2 L}{2} \|\mathbf{d}_t\|^2.$$

Minimizing the upper bound, there is a step size  $\bar{\gamma}$  for which

$$f(\mathbf{x}_t + \bar{\gamma} \mathbf{d}_t) \leq f(\mathbf{x}_t) - \frac{1}{L} \left\langle \frac{\mathbf{d}_t}{\|\mathbf{d}_t\|^2}, \nabla f(\mathbf{x}_t) \right\rangle^2.$$

The step size we actually took (based on  $f$  directly) can only be better:

$$f(\mathbf{x}_t + \gamma \mathbf{d}_t) \leq f(\mathbf{x}_t + \bar{\gamma} \mathbf{d}_t).$$

Taking expectations<sup>1</sup>:  $\mathbb{E}[f(\mathbf{x}_t + \gamma \mathbf{d}_t)] \leq \mathbb{E}[f(\mathbf{x}_t)] - \frac{1}{Ld} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2]$ .

<sup>1</sup>and using the Lemma  $\mathbb{E}_{\mathbf{r}}(\mathbf{r}^\top \mathbf{g})^2 = \frac{1}{d} \|\mathbf{g}\|^2$  for  $\mathbf{r} \sim \text{sphere} \subseteq \mathbb{R}^d$ .



## Convergence rate for derivative-free random search

Same as what we obtained for **gradient descent**,  
now with an **extra factor of  $d$** .  $d$  can be huge!!!

Can do the same for different function classes, as before

- For convex functions, we get a rate of  $\mathcal{O}(dL/\varepsilon)$ .
- For strongly convex, we get  $\mathcal{O}(dL \log(1/\varepsilon))$ .

Always  $d$  times the complexity of gradient descent on the function class.

# Applications for derivative-free random search

## Applications:

- Competitive method for **Reinforcement Learning**.
- Memory and communication advantages: never need to store a gradient.
- Hyperparameter optimization, and other difficult e.g. discrete optimization problems.
- Can be improved to learn a second-order model of the function, during optimization.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Reinforcement Learning

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t, \mathbf{e}_t)$$

where  $\mathbf{s}_t$  is the **state** of the system,  $\mathbf{a}_t$  is the control **action**, and  $\mathbf{e}_t$  is some random **noise**. We assume that  $f$  is fixed, but unknown.

We search for a control “policy”.

$$\mathbf{a}_t := \pi(\mathbf{a}_1, \dots, \mathbf{a}_{t-1}, \mathbf{s}_0, \dots, \mathbf{s}_t).$$

which takes a trajectory of the dynamical system and outputs a new control action.

Want to maximize overall **reward**

$$\begin{aligned} \max_{\mathbf{a}_t} \quad & \mathbb{E}_{\mathbf{e}_t} \left[ \sum_{t=0}^N R_t(\mathbf{s}_t, \mathbf{a}_t) \right] \\ \text{s.t.} \quad & \mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t, \mathbf{e}_t) \\ & (\mathbf{s}_0 \text{ given}). \end{aligned}$$



# Adagrad

Adagrad is an adaptive variant of SGD

pick a stochastic gradient  $\mathbf{g}_t$

$$\text{update } [G_t]_i := \sum_{s=0}^t ([\mathbf{g}_s]_i)^2 \quad \forall i$$

$$[\mathbf{x}_{t+1}]_i := [\mathbf{x}_t]_i - \frac{\gamma}{\sqrt{[G_t]_i}} [\mathbf{g}_t]_i \quad \forall i$$

(standard choice of  $\mathbf{g}_t := \nabla f_j(\mathbf{x}_t)$  for sum-structured objective functions  $f = \sum_j f_j$ ).

- chooses an adaptive, coordinate-wise learning rate;
- strong performance in practice;
- Variants: Adadelta, Adam, RMSprop.



**Adam** is a momentum variant of Adagrad

pick a stochastic gradient  $\mathbf{g}_t$

$$\mathbf{m}_t := \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t \quad (\text{momentum term})$$

$$[\mathbf{v}_t]_i := \beta_2 [\mathbf{v}_{t-1}]_i + (1 - \beta_2) ([\mathbf{g}_s]_i)^2 \quad \forall i \quad (\text{2nd-order statistics})$$

$$[\mathbf{x}_{t+1}]_i := [\mathbf{x}_t]_i - \frac{\gamma}{\sqrt{[\mathbf{v}_t]_i}} [\mathbf{m}_t]_i \quad \forall i.$$

- faster forgetting of older weights
- momentum from previous gradients (see acceleration)
- (simplified version, without correction for initialization of  $\mathbf{m}_0, \mathbf{v}_0$ )
- strong performance in practice, e.g., for self-attention networks.



Only use the sign (one bit) of each gradient entry:  
SignSGD is a communication efficient variant of SGD.

pick a stochastic gradient  $\mathbf{g}_t$

$$[\mathbf{x}_{t+1}]_i := [\mathbf{x}_t]_i - \gamma_t \text{sign}([\mathbf{g}_t]_i) \quad \forall i.$$

(with possible rescaling of  $\gamma$  with  $\|\mathbf{g}_t\|_1$ ).

- communication efficient for distributed training;
- convergence issues.

# Non Linear Programming (Recall)

## Definition

Consider a very general optimization problem of the form:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_j(\boldsymbol{x}) = b_j \quad j = 1, \dots, m \\ & h_i(\boldsymbol{x}) \leq d_i \quad i = 1, \dots, p \end{aligned}$$

or the equivalent more concise form:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{b} \\ & \boldsymbol{h}(\boldsymbol{x}) \leq \boldsymbol{d} \end{aligned} \tag{1}$$

where  $f, g_j, h_i: \mathbb{R}^n \rightarrow \mathbb{R}$ . In the special case when all functions  $f, g_j, h_i$  are linear, problem (1) is a linear program. When some of the functions  $f, g_j, h_i$  are nonlinear, problem (1) is a **Non Linear Program (NLP)**.



## Numerical Nonlinear Programming Solvers

There are numerous software packages for solving nonlinear programs. The following are some popular ones. We list them according to the class of algorithms they are based on:

1. CONOPT, GRG2, ExcelSOLVER. These solvers are based on the *generalized reduced-gradient method*.
2. MATLAB optimization toolbox, SNOPT, NLPQL. These solvers are based on *sequential quadratic programming*.

In particular fmincon under Matlab has five algorithm options:

- 'interior-point' (default)
- 'trust-region-reflective'
- 'sqp'
- 'sqp-legacy'
- 'active-set'

3. MINOS, LANCELOT. These solvers are based on an augmented *Lagrangian approach*.
4. MOSEK, LOQO, IPOPT. These solvers are based on *interior-point methods*.

## Assumption

We consider the class of NLP where the functions  $f, g_i, h_j$  are once or twice **continuously differentiable**.

## Remark

The optimality conditions for linear and convex quadratic programs extend to this more general context, albeit some new technicalities arise. In particular, for a general NLP the theory described below applies to *local minima*.



## Optimality Conditions: Constrained Case (Recall)

### Theorem: First-order necessary conditions

Suppose  $f, g_i, h_j$  are continuously differentiable. If a point  $\mathbf{x}^* \in \mathcal{X}$  is a local minimum of (1) and satisfies the linear independence constraint qualification, then there exist some Lagrange multipliers  $\mathbf{y} \in \mathbb{R}^m$  and  $\mathbf{s} \in \mathbb{R}^p$  such that:

$$\begin{aligned}\nabla f(\mathbf{x}^*) + \sum_{j=1}^m y_j g_j(\mathbf{x}^*) + \sum_{i=1}^p s_i \nabla h_i(\mathbf{x}^*) &= \mathbf{0} \\ \mathbf{s} &\geq \mathbf{0} \\ s_i(h_i(\mathbf{x}^*) - d_i) &= 0 \quad i = 1, \dots, p.\end{aligned}\tag{2}$$



We next describe three main algorithmic approaches to solving a **constrained optimization problem**, namely:

1. the **Generalized Reduced Gradient**;
2. the **Sequential Quadratic Programming**;
3. the **Interior-Point Methods**.

For the sake of time, we will not address their convergence properties as discussed for all the algorithms presented in the previous lessons, but we will analyze their general idea.



# Generalized Reduced Gradient

- The main idea behind the generalized reduced gradient method is to **reduce a constrained problem to a sequence of unconstrained problems** in a space of lower dimension.
- To illustrate this procedure, consider the special case when the equality constraints are linear:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned} \tag{3}$$

for some  $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ .

- Without loss of generality we may assume that  $\boldsymbol{A}$  has **full row rank** as otherwise either some constraints are redundant or the problem is infeasible.
- Since  $\boldsymbol{A}$  has full rank, we can partition both  $\boldsymbol{A}$  and  $\boldsymbol{x}$  as follows:  $\boldsymbol{A} = [\boldsymbol{A}_B \ \boldsymbol{A}_N]$  and  $\boldsymbol{x} = [\boldsymbol{x}_B \ \boldsymbol{x}_N]^\top$  for some subset  $B \subseteq \{1, \dots, n\}$  such that  $\boldsymbol{A}_B$  is non-singular.



# Generalized Reduced Gradient

- Therefore

$$\mathbf{A}\mathbf{x} = \mathbf{b} \Leftrightarrow \mathbf{A}_B\mathbf{x}_B + \mathbf{A}_N\mathbf{x}_N = \mathbf{b} \Leftrightarrow \mathbf{x}_B = \mathbf{A}_B^{-1}(\mathbf{b} - \mathbf{A}_N\mathbf{x}_N).$$

- Consequently, problem (3) is equivalent to the following reduced space unconstrained minimization problem:

$$\min_{\mathbf{x}_N} \hat{f}(\mathbf{x}_N)$$

where  $\hat{f}(\mathbf{x}_N) = f(\mathbf{A}_B^{-1}(\mathbf{b} - \mathbf{A}_N\mathbf{x}_N), \mathbf{x}_N)$ .

- Consider a more general program with nonlinear equality constraints:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) = \mathbf{b} \end{aligned} \tag{4}$$

- We can extend the above approach by approximating the nonlinear equality constraints with their first-order Taylor approximation.

# Generalized Reduced Gradient

More precisely, suppose the current point is  $\mathbf{x}_k$ . Consider the modification of (4) obtained by replacing  $\mathbf{g}(\mathbf{x}) = \mathbf{b}$  with its first-order Taylor approximation:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}^k) + \nabla \mathbf{g}(\mathbf{x}^k)^\top (\mathbf{x} - \mathbf{x}^k) = \mathbf{b}. \end{aligned} \tag{G}$$

Observe that (G) is of the form (3) and is thus amenable to the type of reduced space approach described above. The following algorithm describes a template for a generalized reduced gradient approach to problem (4).

---

## Algorithm Generalized reduced gradient

---

- 1: choose  $\mathbf{x}^0$
  - 2: **for**  $k = 0, 1, \dots$  **do**
  - 3:     solve the linearized constraints problem (G) to find a search direction  $\Delta \mathbf{x}^k$
  - 4:     choose a step length  $\alpha > 0$  and set  $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha \Delta \mathbf{x}^k$
  - 5: **end for**
- 



# Sequential Quadratic Programming

- The central idea of sequential quadratic programming is to capitalize on algorithms for quadratic programming to solve more general nonlinear programming problems of the form (1).
- Given a current iterate  $\mathbf{x}_k$ , problem (1) can be approximated with the following quadratic program::

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^{\top} (\mathbf{x} - \mathbf{x}^k) + \frac{1}{2} (\mathbf{x} - \mathbf{x}^k)^{\top} \mathbf{B}_k (\mathbf{x} - \mathbf{x}^k) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}^k) + \nabla \mathbf{g}(\mathbf{x}^k)^{\top} (\mathbf{x} - \mathbf{x}^k) = \mathbf{b} \\ & \mathbf{h}(\mathbf{x}^k) + \nabla \mathbf{h}(\mathbf{x}^k)^{\top} (\mathbf{x} - \mathbf{x}^k) \leq \mathbf{d} \end{aligned} \tag{S}$$

where  $\mathbf{B}_k = \nabla_{\mathbf{x}\mathbf{x}}^2 L(\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k)$  is the Hessian of the Lagrangian function with respect to the  $\mathbf{x}$  variables, and  $(\mathbf{y}^k, \mathbf{s}^k)$  is the current estimate of the vector of Lagrange multipliers.



# Sequential Quadratic Programming

The following algorithm describes a template for a sequential quadratic programming approach to problem (4). Once again, the step length  $\alpha$  at each iteration is typically chosen to balance both goals of **objective function reduction** and **constraint satisfaction**.

---

## Algorithm Sequential quadratic programming

---

- 1: choose  $\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0$
  - 2: **for**  $k = 0, 1, \dots$  **do**
  - 3:     solve the quadratic program (S) to find a search direction  
        $(\Delta \mathbf{x}^k, \Delta \mathbf{y}^k, \Delta \mathbf{s}^k)$
  - 4:     choose a step length  $\alpha > 0$  and set  $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}) = (\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k) + \alpha(\Delta \mathbf{x}^k, \Delta \mathbf{y}^k, \Delta \mathbf{s}^k)$
  - 5: **end for**
- 



# Interior-Point Methods

- Interior-point methods generate a **sequence of iterates** that satisfy some inequalities strictly and each iteration of the algorithm aims to make progress towards satisfying the optimality conditions (2).
- The algorithm inevitably becomes a bit more elaborate for nonlinear programs because of the nonlinearities in the constraints.
- Given a vector  $s \in \mathbb{R}^p$ , let  $S \in \mathbb{R}^{p \times p}$  denote the diagonal matrix defined by  $S_{ii} = s_i$ , for  $i = 1, \dots, n$ , and let  $\mathbf{1} \in \mathbb{R}^p$  denote the vector whose components are all 1s.
- The optimality conditions (2) can be restated as:

$$\begin{bmatrix} \nabla f(x) + \nabla g(x)y + \nabla h(x)s \\ g(x) - b \\ h(x) + z - d \\ Sz\mathbf{1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad s, z \geq 0.$$



# Interior-Point Methods

- Given  $\mu > 0$ , let  $(\mathbf{x}(\mu), \mathbf{y}(\mu), \mathbf{z}(\mu), \mathbf{s}(\mu))$  be the solution to the following perturbed version of the above optimality conditions:

$$\begin{bmatrix} \nabla f(\mathbf{x}) + \nabla g(\mathbf{x})\mathbf{y} + \nabla h(\mathbf{x})\mathbf{s} \\ g(\mathbf{x}) - \mathbf{b} \\ h(\mathbf{x}) + \mathbf{z} - \mathbf{d} \\ S\mathbf{Z}\mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mu\mathbf{1} \end{bmatrix} \quad \mathbf{s}, \mathbf{z} \geq \mathbf{0}.$$

- The first condition above can be written as  $\mathbf{r}_\mu(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{s}) = \mathbf{0}$  for the **residual vector**:

$$\mathbf{r}_\mu(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{s}) = \begin{bmatrix} \nabla f(\mathbf{x}) + \nabla g(\mathbf{x})\mathbf{y} + \nabla h(\mathbf{x})\mathbf{s} \\ g(\mathbf{x}) - \mathbf{b} \\ h(\mathbf{x}) + \mathbf{z} - \mathbf{d} \\ S\mathbf{Z}\mathbf{1} - \mu\mathbf{1} \end{bmatrix}.$$



# Interior-Point Methods

- The **central path** is the set  $\{(x(\mu), y(\mu), z(\mu), s(\mu)) : \mu > 0\}$ .
- Under suitable assumptions  $(x(\mu), y(\mu), z(\mu), s(\mu))$  converges to a **local optimal solution** to (2).
- This suggests the following algorithmic strategy: suppose  $(x, y, z, s)$  is *near*  $(x(\mu), y(\mu), z(\mu), s(\mu))$  for some  $\mu > 0$ .
- Use  $(x, y, z, s)$  to move to a better point  $(x^+, y^+, z^+, s^+)$  *near*  $(x(\mu^+), y(\mu^+), z(\mu^+), s(\mu^+))$  for some  $\mu^+ < \mu$ .



# Interior-Point Methods

- It can be shown that if a point  $(x, y, z, s)$  is on the central path, then the corresponding value of  $\mu$  satisfies  $z^\top s = p\mu$ .
- Likewise, given  $z, s > \mathbf{0}$  define:

$$\mu(z, s) := \frac{z^\top s}{p}.$$

- To move from a current point  $(x, y, z, s)$  to a new point, we use the Newton step for the nonlinear system of equations  $r_\mu(x, y, z, s) = \mathbf{0}$ :

$$(\Delta x, \Delta y, \Delta z, \Delta s) = -r'_\mu(x, y, z, s)^{-1} r_\mu(x, y, z, s). \quad (\text{N})$$



The following algorithm presents a template for an interior-point method.

---

**Algorithm** Interior-point method for nonlinear programming

---

- 1: choose  $\mathbf{x}^0, \mathbf{y}^0$  and  $\mathbf{z}^0, \mathbf{s}^0 > 0$
  - 2: **for**  $k = 0, 1, \dots$  **do**
  - 3:     solve the Newton system (N) for  $(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{s}) = (\mathbf{x}^k, \mathbf{y}^k, \mathbf{z}^k, \mathbf{s}^k)$  and  $\mu := 0.1\mu(\mathbf{z}^k, \mathbf{s}^k)$
  - 4:     choose a step length  $\alpha \in (0, 1]$  and set  $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{z}^{k+1}, \mathbf{s}^{k+1}) = (\mathbf{x}^k, \mathbf{y}^k, \mathbf{z}^k, \mathbf{s}^k) + \alpha(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{z}, \Delta\mathbf{s})$
  - 5: **end for**
- 

The step length  $\alpha$  in step 4 should be chosen via a backtracking procedure so that  $\mathbf{z}^{k+1}, \mathbf{s}^{k+1} > 0$  and the size of  $r_\mu(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{z}^{k+1}, \mathbf{s}^{k+1})$  is sufficiently smaller than  $r_\mu(\mathbf{x}^k, \mathbf{y}^k, \mathbf{z}^k, \mathbf{s}^k)$ .



# Optimization - 3 CFU

## Lesson 10

---

*Francesca Maggioni*

Department of Management, Information and Production Engineering  
University of Bergamo

Academic Year 2023-2024

Department of Management, Information and Production Engineering  
University of Bergamo



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

## Objective

**Support Vector Machine** (SVM) is a recent and very effective class of supervised *Machine Learning* (ML) models and algorithms for **classification** and regression.

## Origins

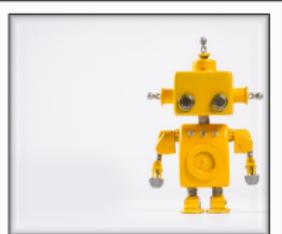
SVM were first introduced in the 70s:

- Vapnik, V., Chervonenkis, A., 1974. **Theory of Pattern Recognition**.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Applications of Support Vector Machine



Robotics



Medical Diagnosis



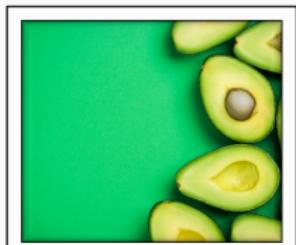
Speech Recognition



Character Recognition



Spam Filtering



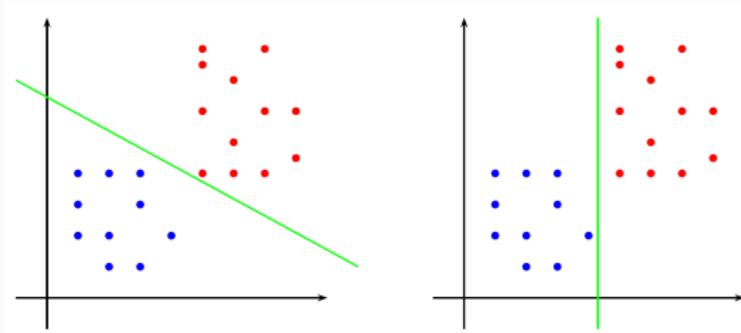
Nutrition



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

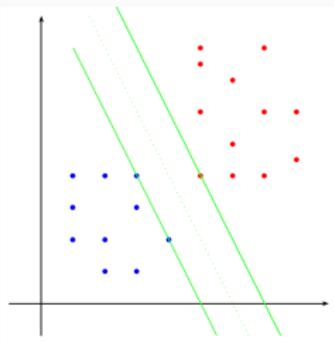
# Support Vector Machine

- The figure below shows two different hyperplanes (in this case, two lines) that both separate the red and blue points.
- In terms of training set the two hyperplanes are equivalent but the classification of new points is highly different.



# Support Vector Machine

- For classification problems, the fundamental idea in SVM is to determine, instead of a single separating hyperplane, **a pair of parallel hyperplanes for which the margin is maximum.**



# Linear Separable SVM Model

- Let  $\{(x^p, y_p)\}$  with  $p = 1, \dots, P$  be the set of training data (i.e., the **training set**) where:
  - $x^i \in \mathbb{R}^N$  represents an input observation,
  - and  $y_p \in \{-1, +1\}$  its associated label.
- The training set is **linearly separable** if and only there exists an hyperplane  $H_{w,\theta}$  where  $w \in \mathbb{R}^N$  and  $\theta \in \mathbb{R}$  such that:

$$\begin{cases} w^\top x^p + \theta > 0 & \text{if } y_p = +1 \\ w^\top x^p + \theta < 0 & \text{if } y_p = -1 \end{cases}$$

for all  $p = 1, \dots, P$ .



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Linear Separable SVM Model

- For a linearly separable training set there always exists a separating hyperplane  $H_{\bar{w}, \bar{\theta}}$  such that for all  $p = 1, \dots, P$ :

$$\begin{cases} \bar{w}^\top x^p + \bar{\theta} \geq +1 & \text{if } y_p = +1 \\ \bar{w}^\top x^p + \bar{\theta} \leq -1 & \text{if } y_p = -1. \end{cases}$$

- In fact, let  $H_{w, \theta}$  be a separating hyperplane and let:

$$\alpha = \min_{p=1, \dots, P, y_p = +1} w^\top x^p + \theta$$

$$\beta = \max_{p=1, \dots, P, y_p = -1} w^\top x^p + \theta.$$

Note that  $\alpha > 0$  and  $\beta < 0$ .



# Linear Separable SVM Model

- Moreover, let  $\gamma = \min\{\alpha, -\beta\}$ ,  $\bar{w} = \frac{w}{\gamma}$  and  $\bar{\theta} = \frac{\theta}{\gamma}$ .
- Now, for each  $p = 1, \dots, P$  if  $y_p = +1$  then:

$$w^\top x^p + \theta \geq \gamma \quad \text{and hence} \quad \bar{w}^\top x^p + \bar{\theta} \geq 1$$

while if  $y_p = -1$  then:

$$w^\top x^p + \theta \leq -\gamma \quad \text{and hence} \quad \bar{w}^\top x^p + \bar{\theta} \leq -1.$$

- Therefore, a training set is linearly separable if there exist a separating hyperplane  $H_{w,\theta}$  such that for all  $p = 1, \dots, P$ :

$$\begin{cases} w^\top x^p + \theta \geq +1 & \text{if } y_p = +1 \\ w^\top x^p + \theta \leq -1 & \text{if } y_p = -1. \end{cases}$$

- Note that conditions above can be rewritten as:

$$y_p(w^\top x^p + \theta) \geq 1 \quad \forall p = 1, \dots, P.$$

# Linear Separable SVM Model

## Definition (Hyperplane Margin)

Let the training set be linearly separable. The **margin**  $\rho_{w,\theta}$  of  $H_{w,\theta}$  is the minimum distance between points in the training set and the hyperplane:

$$\rho_{w,\theta} = \min_{p=1, \dots, P} \frac{|w^\top x^p + \theta|}{\|w\|}$$

where  $\|\cdot\|$  is the Euclidean norm.

- SVM will choose as separating hyperplane  $H_{w^*,\theta^*}$  the one for which the margin is maximum.
- That can be achieved by solving the following optimization problem:

$$\begin{aligned} \max \quad & \rho_{w,\theta} \\ \text{s.t.} \quad & y_p(w^\top x^p + \theta) \geq 1 \quad p = 1, \dots, P. \end{aligned}$$



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Linear Separable SVM Model

- Notice that since for all  $p = 1, \dots, P$ :

$$y_p(w^\top x^p + \theta) \geq 1 \iff |w^\top x^p + \theta| \geq 1$$

then from the definition of  $\rho_{w,\theta}$  we have:

$$\rho_{w,\theta} = \min_{p=1, \dots, P} \frac{|w^\top x^p + \theta|}{\|w\|} \geq \frac{1}{\|w\|}.$$

- Moreover, for each separating hyperplane  $H_{w,\theta}$  there exists a separating hyperplane  $H_{\bar{w},\bar{\theta}}$  such that:

$$\rho_{w,\theta} \leq \rho_{\bar{w},\bar{\theta}} = \frac{1}{\|w\|}$$

and  $p', p'' \in \{1, \dots, P\}$  such that

$$y^{p'} = +1 \quad \text{and} \quad \bar{w}^\top x^{p'} + \bar{\theta} = +1$$

$$y^{p''} = -1 \quad \text{and} \quad \bar{w}^\top x^{p''} + \bar{\theta} = -1.$$

# Linear Separable SVM Model

- In fact, for the separating hyperplane  $H_{w,\theta}$  let  $p', p'' \in \{1, \dots, P\}$  such that  $y_{p'} = +1$ ,  $y_{p''} = -1$  and:

$$\alpha = \min_{p=1, \dots, P, y_p=+1} \frac{|w^\top x^p + \theta|}{\|w\|} = \frac{|w^\top x^{p'} + \theta|}{\|w\|}$$
$$\beta = \min_{p=1, \dots, P, y_p=-1} \frac{|w^\top x^p + \theta|}{\|w\|} = \frac{|w^\top x^{p''} + \theta|}{\|w\|}.$$

- Note that:

$$\rho_{w,\theta} \leq \min\{\alpha, \beta\} \leq \frac{\alpha + \beta}{2} = w^\top (x^{p'} - x^{p''}).$$



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Linear Separable SVM Model (further calculations)

- Let now  $\gamma, \delta \in \mathbb{R}$  such that:

$$\gamma w^\top x^{p'} + \delta = +1$$

$$\gamma w^\top x^{p''} + \delta = -1$$

that is:

$$\gamma = \frac{2}{w^\top (x^{p'} - x^{p''})} \quad \text{and} \quad \delta = \frac{w^\top (x^{p'} - x^{p''})}{w^\top (x^{p'} + x^{p''})}.$$

- Now:

$$w^\top x^{p'} + \theta \geq 1 \geq 0 \text{ and } w^\top x^{p''} + \theta \leq -1, \text{ hence } -(w^\top x^{p''} + \theta) \geq 1 \geq 0.$$

- Therefore:

$$w^\top x^{p'} + \theta - (w^\top x^{p''} + \theta) = w^\top (x^{p'} - x^{p''}) \geq 2 > 0.$$

- So,  $0 < \gamma \leq 2$ .

# Linear Separable SVM Model (further calculations)

- Let  $\bar{w} = \gamma w$  and  $\bar{\theta} = \delta$ . The Hyperplane  $H_{\bar{w}, \bar{\theta}}$  is a separating hyperplane.  
In fact, since:

$$w^\top x^p \geq w^\top x^{p'} \quad \forall p \in \{1, \dots, P : y_p = +1\}$$

$$w^\top x^p \leq w^\top x^{p''} \quad \forall p \in \{1, \dots, P : y_p = -1\}$$

then

$$\gamma w^\top x^p + \bar{\theta} \geq \gamma w^\top x^{p'} + \bar{\theta} = +1 \quad \forall p \in \{1, \dots, P : y_p = +1\}$$

$$\gamma w^\top x^p + \bar{\theta} \leq \gamma w^\top x^{p''} + \bar{\theta} = -1 \quad \forall p \in \{1, \dots, P : y_p = -1\}.$$

- Hence:

$$\bar{w}^\top x^p + \bar{\theta} \geq +1 \quad \forall p \in \{1, \dots, P : y_p = +1\}$$

$$\bar{w}^\top x^p + \bar{\theta} \leq -1 \quad \forall p \in \{1, \dots, P : y_p = -1\}.$$



# Linear Separable SVM Model (further calculations)

- Moreover:

$$\begin{aligned}\rho_{\bar{w}, \bar{\theta}} &= \min_{p=1, \dots, P} \frac{|\bar{w}^\top x^p + \bar{\theta}|}{\|\bar{w}\|} \\ &= \frac{1}{\|\bar{w}\|} \\ &= \frac{1}{\|\gamma w\|} \\ &= w^\top (x^{p'} - x^{p''}) \\ &= \rho_{w, \theta}\end{aligned}$$

and

$$\begin{aligned}\bar{w}^\top x^{p'} + \bar{\theta} &= +1 \\ \bar{w}^\top x^{p''} + \bar{\theta} &= -1.\end{aligned}$$



# Linear Separable SVM Model

Therefore the problem:

$$\begin{aligned} \max \quad & \rho_{w,\theta} \\ \text{s.t.} \quad & y_p(w^\top x^p + \theta) \geq 1 \quad p = 1, \dots, P \end{aligned}$$

is equivalent to solve the following convex quadratic optimization problem:

## Hard Margin SVM

$$\begin{aligned} \min \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y_p(w^\top x^p + \theta) \geq 1 \quad p = 1, \dots, P. \end{aligned} \tag{*}$$

Notice that even though the objective function of problem (\*) is not strictly convex (in  $w$  and  $\theta$ ), it admits a unique solution.



# Optimality Conditions and Dual Problem

The **Lagrangian function** associated to problem (\*) is:

$$\mathcal{L}(w, \theta, \alpha) = \frac{1}{2} w^\top w - \sum_{p=1}^P \alpha_p (y_p(w^\top x^p + \theta) - 1)$$

and the Karush–Kuhn–Tucker optimality conditions are:

## KKT Conditions

$$w = \sum_{p=1}^P \alpha_p y_p x^p$$

$$\sum_{p=1}^P \alpha_p y_p = 0$$

$$\alpha_p \geq 0 \quad p = 1, \dots, P$$

$$y_p(w^\top x^p + \theta) - 1 \geq 0 \quad p = 1, \dots, P$$

$$\alpha_p [y_p(w^\top x^p + \theta) - 1] = 0 \quad p = 1, \dots, P.$$



## Remark

- The first KKT condition indicates that the optimal  $w$  is a linear combination (with coefficients  $y_p \alpha_p$ ) of the training set elements  $x^p$ .
- The last set of complementarity conditions implies that the only vectors entering in the linear combination are those for which

$$y_p(w^\top x^p + \theta) = 1.$$

These vectors are called support vectors.



# Optimality Conditions and Dual Problem

The **Wolfe dual** is given by:

$$\max_{w, \theta, \alpha} \quad \mathcal{L}(w, \theta, \alpha)$$

$$s.t. \quad \nabla_w \mathcal{L}(w, \theta, \alpha) = 0$$

$$\nabla_\theta \mathcal{L}(w, \theta, \alpha) = 0$$

$$\alpha \geq 0.$$

Now:

$$\begin{aligned}\mathcal{L}(w, \theta, \alpha) &= \frac{1}{2} w^\top w - \sum_{p=1}^P \alpha_p (y_p (w^\top x^p + \theta) - 1) \\ &= -\frac{1}{2} w^\top w + w^\top \left( w - \sum_{p=1}^P \alpha_p y_p x^p \right) - \theta \sum_{p=1}^P \alpha_p y_p + \sum_{p=1}^P \alpha_p.\end{aligned}$$



# Optimality Conditions and Dual Problem

Therefore, after substituting for  $w$ , the dual problem can be written as:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \left( \sum_{p=1}^P \alpha_p y_p x^p \right)^\top \left( \sum_{q=1}^P \alpha_q y_q x^q \right) - \sum_{p=1}^P \alpha_p \\ \text{s.t.} \quad & \sum_{p=1}^P \alpha_p y_p = 0 \\ & \alpha_p \geq 0 \quad p = 1, \dots, P. \end{aligned}$$

Let now  $K : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$  with  $K(\bar{x}, \hat{x}) = \bar{x}^\top \hat{x}$ . Then, the minimization problem above can be re-written as:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \alpha^\top Q \alpha - e^\top \alpha \\ \text{s.t.} \quad & y^\top \alpha = 0 \\ & \alpha \geq 0 \end{aligned}$$

where  $e$  is a vector with all components equal to 1,  $Q \in \mathbb{R}^{P \times P}$  and:

$$Q_{pq} = y_p y_q K(x^p, x^q).$$

# Optimality Conditions and Dual Problem

Once the dual problem is solved (let  $\alpha^*$  be the optimal solution), the separating hyperplane

$$w^{*\top} x + \theta^* = 0$$

is obtained as follows:

- $w^* = \sum_{p=1}^P \alpha_p^* y_p x^p$  (see first KKT condition);
- if  $p'$  is a support vector (that is,  $\alpha_{p'}^* > 0$ ), then:

$$y_{p'}(w^{*\top} x^p + \theta^*) - 1 = 0$$

(see last KKT conditions) from which  $\theta^*$  can be easily computed.

Then, the decision function  $h(\cdot)$  for linear SVM is:

## Decision Function

$$h(x) = \text{sign}(w^{*\top} x + \theta^*) = \text{sign}\left(\sum_{p=1}^P \alpha_p^* y_p K(x^p, x) + \theta^*\right) \quad (\diamond)$$

where  $x \in \mathbb{R}^N$ .

# Linear Non-Separable SVM Model

When the **training set** is **not linearly separable** the problem

$$\begin{aligned} \min \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y_p(w^\top x^p + \theta) \geq 1 \quad p = 1, \dots, P. \end{aligned}$$

has **empty feasible region**. In this case **slack variables**  $\xi_p$  must be introduced and the new optimization problem is:

## Soft Margin SVM

$$\begin{aligned} \min \quad & \frac{1}{2} \|w\|^2 + C \sum_{p=1}^P \xi_p \\ \text{s.t.} \quad & y_p(w^\top x^p + \theta) \geq 1 - \xi_p \quad p = 1, \dots, P \\ & \xi_p \geq 0 \quad p = 1, \dots, P. \end{aligned} \tag{**}$$



# Linear Non-Separable SVM Model

- For the hyperplane  $H_{w,\theta}$  if a point  $x^p$  is not correctly classified then the corresponding quantity  $\xi_p$  is bigger than 1. In fact:

if  $y_p = +1$  and  $w^\top x^p - \theta \leq 0$  then  $\xi_p = 1 - w^\top x^p + \theta \geq 1$ ,

if  $y_p = -1$  and  $w^\top x^p + \theta \geq 0$  then  $\xi_p = 1 + w^\top x^p + \theta \geq 1$ .

- Moreover, values of  $\xi_p$  in the interval  $(0, 1)$  correspond to points correctly classified by the hyperplane but inside the separation zone.
- The objective function includes two terms:
  - the first term corresponds to margin maximization:  $\frac{1}{2}||w||^2$ ;
  - the second term is an upper bound on the number of incorrectly classified training set points:  $\sum \xi_p$ .



# Optimality Conditions and Dual Problem

The **Lagrangian function** associated to problem (\*\*) is:

$$\mathcal{L}(w, \theta, \xi, \alpha) = \frac{1}{2} w^\top w + C \sum_{p=1}^P \xi_p - \sum_{p=1}^P \alpha_p \left( y_p (w^\top x^p + \theta) - 1 + \xi_p \right)$$

and the Karush–Kuhn–Tucker optimality conditions are:

$$w = \sum_{p=1}^P \alpha_p y_p x^p$$

$$\sum_{p=1}^P \alpha_p y_p = 0$$

$$0 \leq \alpha_p \leq C \quad p = 1, \dots, P$$

$$\xi_p \geq 0 \quad p = 1, \dots, P$$

$$\xi_p (C - \alpha_p) = 0 \quad p = 1, \dots, P$$

$$y_p (w^\top x^p + \theta) - 1 + \xi_p \geq 0 \quad p = 1, \dots, P$$

$$\alpha_p [y_p (w^\top x^p + \theta) - 1 + \xi_p] = 0 \quad p = 1, \dots, P.$$

# Optimality Conditions and Dual Problem

The **dual problem** is now:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \left( \sum_{p=1}^P \alpha_p y_p x^p \right)^\top \left( \sum_{q=1}^P \alpha_q y_q x^q \right) - \sum_{p=1}^P \alpha_p \\ \text{s.t.} \quad & \sum_{p=1}^P \alpha_p y_p = 0 \\ & 0 \leq \alpha_p \leq C \quad p = 1, \dots, P \end{aligned}$$

or, in more compact form:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \alpha^\top Q \alpha - e^\top \alpha \\ \text{s.t.} \quad & y^\top \alpha = 0 \\ & 0 \leq \alpha \leq Ce \end{aligned}$$

where  $e$  is a vector with all components equal to 1,  $Q \in \mathbb{R}^{P \times P}$  and:

$$Q_{pq} = y_p y_q K(x^p, x^q).$$

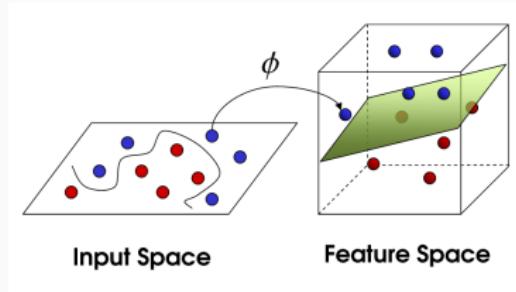
## Remark

- Here the support vectors are either free support vectors corresponding to  $0 < \alpha_p^* < C$  or bounded support vectors, corresponding to  $\alpha_p^* = C$ .
- Again, the vector  $w^*$  is a linear combination of support vectors while the scalar  $\theta^*$  can be calculated using any free support vector.



# Non-Linear SVM Model: Learning in the Feature Space

- The idea behind non-linear SVM is to map the input space into a **feature space** where they are linearly separable.
- It must be recalled that in linear SVM models the dual problem can be easily constructed using the function  $K(\cdot, \cdot)$ . Moreover, the decision function ( $\diamond$ ) only utilizes the function  $K(\cdot, \cdot)$ .
- Let  $\mathcal{F}$  be an Hilbert space (the feature space) whose dimension is bigger than  $n$  and possibly infinite, and let  $\phi : \mathbb{R}^n \mapsto \mathcal{F}$  be a mapping from  $\mathbb{R}^n$  to the feature space.

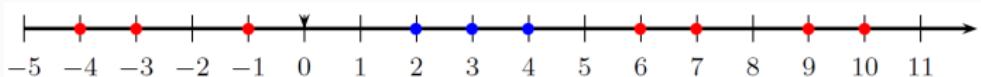


## Example

As an example, consider the following 1-dimensional training set with  $P = 10$ :

$p$	1	2	3	4	5	6	7	8	9	10
$x_p$	-4	-3	-1	2	3	4	6	7	9	10
$y_p$	-1	-1	-1	+1	+1	+1	-1	-1	-1	-1

represented in the figure below:



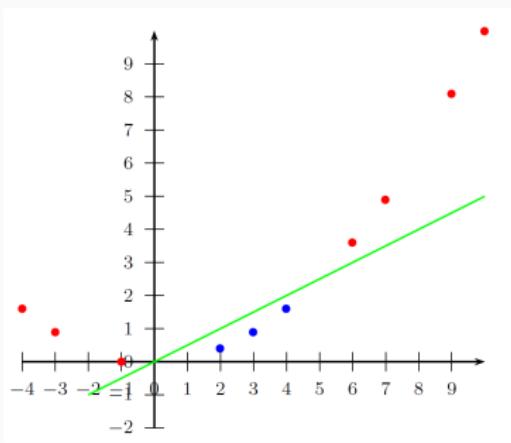
This training set is not linearly separable. A possible classification function would be  $h(x) = \text{sign}(x(x - 5))$ .

## Example

Now, consider the map  $\phi : \mathbb{R} \rightarrow \mathbb{R}^2$  given by:

$$\phi(x) := \begin{bmatrix} x \\ x^2 \end{bmatrix} \in \mathbb{R}^2.$$

In  $\mathbb{R}^2$  the mapped values are linearly separable as clearly showed in the figure:



# Non-Linear SVM Model: Learning in the Feature Space

For the non-linear SVM model the optimization problem is now:

## Non Linear SVM

$$\begin{aligned} \min_{w, \theta, \xi} \quad & \frac{1}{2} \|w\|^2 + C \sum_{p=1}^P \xi_p \\ \text{s.t.} \quad & y_p (w^\top \phi(x^p) + \theta) \geq 1 - \xi_p \quad p = 1, \dots, P \quad (***) \\ & \xi_p \geq 0 \quad p = 1, \dots, P. \end{aligned}$$



# Optimality Conditions and Dual Problem

For problem (\*\*\*), the Karush–Kuhn–Tucker optimality conditions are:

## KKT Conditions Non Linear SVM

$$w = \sum_{p=1}^P \alpha_p y_p \phi(x^p)$$

$$\sum_{p=1}^P \alpha_p y_p = 0$$

$$0 \leq \alpha_p \leq C \quad p = 1, \dots, P$$

$$\xi_p \geq 0 \quad p = 1, \dots, P$$

$$\xi_p(C - \alpha_p) = 0 \quad p = 1, \dots, P$$

$$y_p(w^\top \phi(x^p) + \theta) - 1 + \xi_p \geq 0 \quad p = 1, \dots, P$$

$$\alpha_p [y_p(w^\top \phi(x^p) + \theta) - 1 + \xi_p] = 0 \quad p = 1, \dots, P.$$



# Optimality Conditions and Dual Problem

The **dual problem** is now:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \alpha^\top Q \alpha - e^\top \alpha \\ s.t. \quad & y^\top \alpha = 0 \\ & 0 \leq \alpha \leq C e \end{aligned}$$

where  $e$  is a vector with all components equal to 1,  $Q \in \mathbb{R}^{P \times P}$  and:

$$Q_{pq} = y_p y_q K(x^p, x^q) \quad \text{with} \quad K(x^p, x^q) = \phi(x^p)^\top \phi(x^q).$$

## Remark

The function  $K : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$  is called the **Kernel function**.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Optimality Conditions and Dual Problem

Once again, when the dual problem has been solved (let  $\alpha^*$  be its optimal solution) the optimal hyperplane  $H_{w^*, \theta^*}$  can be obtained by:

- $w^* = \sum_{p=1}^P \alpha_p^* y_p \phi(x^p)$  (see first KKT condition);
- if  $p'$  is a support vector (that is,  $0 < \alpha_{p'}^* < C$ ), then:

$$y_{p'}(w^\top \phi(x^p) + \theta^*) - 1 = y_{p'} \left( \sum_{p=1}^P \alpha_p^* y_p K(x^p, x^{p'}) + \theta^* \right) - 1 = 0$$

from which  $\theta^*$  can be easily computed.

Then, the decision function  $h(\cdot)$  for non-linear SVM is:

$$h(x) = \text{sign} \left( w^{*\top} \phi(x) + \theta^* \right) = \text{sign} \left( \sum_{p=1}^P \alpha_p^* y_p K(x^p, x) + \theta^* \right). \quad (\diamond\diamond)$$



## Remark

- *The difficulty in an explicit use of the mapping  $\phi(\cdot)$  to construct the Feature Space is that the resulting space can be extremely high-dimensional.*
- *However, it must be noticed that both in calculating  $\theta^*$  and in evaluating the decision function on a new point, the actual form of the function  $\phi(\cdot)$  is not necessary but it is only necessary to calculate the Kernel function  $K(\cdot, \cdot)$ .*



## Example

Consider, for example the following function:

$$\phi : x \in \mathbb{R}^2 \mapsto \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}.$$

Then:

$$\phi(\bar{x})^\top \phi(\hat{x}) = \begin{bmatrix} \bar{x}_1^2 & \sqrt{2}\bar{x}_1\bar{x}_2 & \bar{x}_2^2 \end{bmatrix} \begin{bmatrix} \hat{x}_1^2 \\ \sqrt{2}\hat{x}_1\hat{x}_2 \\ \hat{x}_2^2 \end{bmatrix} = (\bar{x}^\top \hat{x})^2$$

and

$$K(\bar{x}, \hat{x}) = (\bar{x}^\top \hat{x})^2.$$



# Kernels

Therefore, a fundamental question is under which hypothesis a function

$$K : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$$

is a Kernel function, that is

$$K(\bar{x}, \hat{x}) = \phi(\bar{x})^\top \phi(\hat{x})$$

for some function  $\phi : \mathbb{R}^N \rightarrow \mathcal{F}$ .

A related question is for which mappings  $\phi(\cdot)$  is easy to evaluate the scalar product  $\phi(\bar{x})^\top \phi(\hat{x})$ ?

Clearly, due to the properties of the inner product, the matrix  $K \in \mathbb{R}^{P \times P}$  with

$$K_{pq} = K(x^p, x^q)$$

must be **symmetric**.

The answer is provided by the **Mercer Theorem**.

## Theorem

Let  $\mathcal{X}$  be a measure space and let  $L^2(\mathcal{X})$  be the Hilbert space of square-integrable functions  $f : \mathcal{X} \rightarrow \mathbb{R}$ . Let  $K \in L^2(\mathcal{X} \times \mathcal{X})$ .  $K$  is a kernel if and only if there exists some feature map  $\phi : \mathcal{X} \rightarrow \mathcal{H}$  where  $\mathcal{H}$  is a Hilbert space such that

$$K(\bar{x}, \hat{x}) = \phi(\bar{x})^\top \phi(\hat{x}).$$

A symmetric  $K \in L^2(\mathcal{X} \times \mathcal{X})$  is a kernel if and only if:

$$\forall f \in L^2(\mathcal{X}) \quad \int_{\mathcal{X} \times \mathcal{X}} K(\bar{x}, \hat{x}) f(\bar{x}) f(\hat{x}) d\bar{x} d\hat{x} \geq 0.$$



In the finite dimensional case the above results can be simplified as follows.

## Theorem

Let  $K : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$  be a symmetric function. The function  $K(\cdot, \cdot)$  is a **Kernel function if and only if** the matrix  $K \in \mathbb{R}^{P \times P}$  with

$$K_{pq} = K(x^p, x^q)$$

is **positive semidefinite** for each training set composed of  $P$  vectors.



# Kernels

The most common Kernels are:

- Linear Kernel:  $K(\bar{x}, \hat{x}) = \bar{x}^\top \hat{x}$ .
- Polynomial Kernel:  $K(\bar{x}, \hat{x}) = (\bar{x}^\top \hat{x})^d$ .
- Gaussian<sup>1</sup> Kernel (Radial Basis Function):  $K(\bar{x}, \hat{x}) = e^{-\frac{\|\bar{x} - \hat{x}\|^2}{2\sigma^2}}$ .
- Sigmoidal Kernel:  $K(\bar{x}, \hat{x}) = \tanh(\rho_1 \bar{x}^\top \hat{x} + \rho_2)$ .



---

<sup>1</sup>The Gaussian Kernel is an inner product in an infinite dimensional space.

# Kernels

Moreover, new Kernels can be constructed based on existing Kernels via the following operations<sup>2</sup>:

- Scalar multiplication, where  $\rho \geq 0$ :  $K(\bar{x}, \hat{x}) = \rho K_1(\bar{x}, \hat{x})$ .
- Adding a positive constant, where  $\rho \geq 0$ :  $K(\bar{x}, \hat{x}) = K_1(\bar{x}, \hat{x}) + \rho$ .
- Linear combination with non-negative weights:  
$$K(\bar{x}, \hat{x}) = \sum_{j=1}^J \rho_j K_j(\bar{x}, \hat{x}), \text{ where } \rho_j \geq 0 \text{ for } j = 1, \dots, J.$$
- Product of two Kernels:  $K(\bar{x}, \hat{x}) = K_1(\bar{x}, \hat{x}) \cdot K_2(\bar{x}, \hat{x})$ .
- Polynomial functions of a Kernel:  $K(\bar{x}, \hat{x}) = P(K_1(\bar{x}, \hat{x}))$ , where  $P(\cdot)$  is a polynomial with non-negative coefficients.
- Exponential function of a Kernel:  $K(\bar{x}, \hat{x}) = \exp(K_1(\bar{x}, \hat{x}))$ .

---

<sup>2</sup>Here  $K_1(\cdot, \cdot), K_2(\cdot, \cdot), \dots, K_J(\cdot, \cdot)$  are valid Kernels.

# Optimization - 3 CFU

## Lesson 11

---

*Francesca Maggioni*

Department of Management, Information and Production Engineering  
University of Bergamo

Academic Year 2023-2024

Department of Management, Information and Production Engineering  
University of Bergamo



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

We are given the training data  $(\mathbf{X}, \mathbf{y})$ , containing  $n$  observations  $(\mathbf{x}_i, y_i)$ ,  $i = 1, \dots, n$ , each with  $p$  features  $\mathbf{x}_i \in \mathbb{R}^p$  and a class label  $y_i \in \{1, \dots, K\}$  indicating which of  $K$  possible labels is assigned to this point.

We assume without loss of generality that the values for each dimension across the training data are normalized to the  $0 - 1$  interval, meaning each  $\mathbf{x}_i \in [0, 1]^p$ .

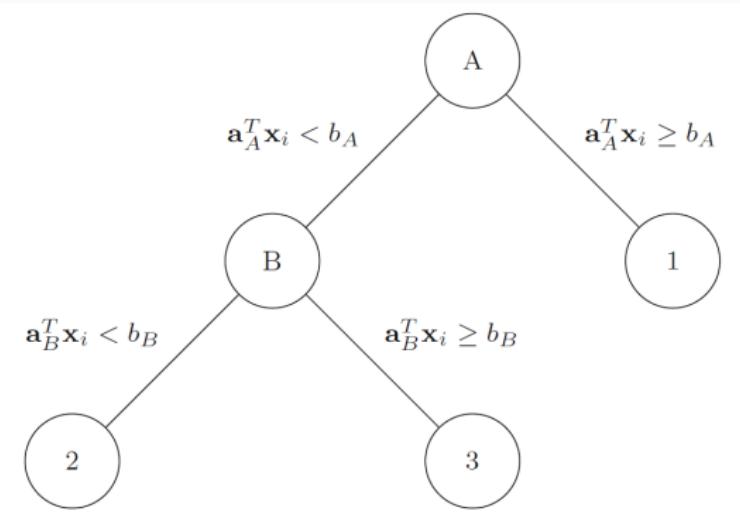
## Objective

**Decision Tree (DT)** method seeks to recursively partition  $[0, 1]^p$  to yield a number of hierarchical, disjoint regions that represent a classification tree.



# Decision Trees

An example of a DT is shown below.



The DT has two branch nodes  $A$  and  $B$  and three leaf nodes 1, 2, and 3.

# Decision Trees

The final tree is comprised of branch nodes and leaf nodes:

- Branch nodes apply a split with parameters  $\mathbf{a}$  and  $b$ . For a given point  $i$ , if  $\mathbf{a}^\top \mathbf{x}_i < b$  the point will follow the left branch from the node, otherwise it takes the right branch. A subset of methods, including CART, produce univariate or axis-aligned decision trees which restrict the split to a single dimension, *i.e.*, a single component of  $\mathbf{a}$  will be 1 and all others will be 0.
- Leaf nodes are assigned a class that will determine the prediction for all data points that fall into the leaf node. The assigned class is almost always taken to be the class that occurs most often among points contained in the leaf node.



## Remark

*Classical DT methods (e.g., CART, ID3, and C4.5) take a top-down approach to building the tree.*

*At each step of the partitioning process, they seek to find a split that will partition the current region in such a way to maximize a so-called splitting criterion.*

*This criterion is often based on the label impurity of the data points contained in the resulting regions instead of minimizing the resulting misclassification error.*

*The algorithm proceeds to recursively partition the two new regions that are created by the hyperplane split. The partitioning terminates once any one of a number of stopping criteria are met.*



The criteria for CART<sup>1</sup> are as follows:

- It is not possible to create a split where each side of the partition has at least a certain number of nodes,  $N_{\min}$ .
- All points in the candidate node share the same class.

Once the splitting process is complete, a class label  $1, \dots, K$  is assigned to each region. This class will be used to predict the class of any points contained inside the region. As mentioned earlier, this assigned class will typically be the most common class among the points in the region.



---

<sup>1</sup>Classification and Regression Trees

The final step in the process is pruning the tree in an attempt to avoid overfitting.

The pruning process works upwards through the partition nodes from the bottom of the tree.

The decision of whether to prune a node is controlled by the so-called **complexity parameter  $\alpha$** , which balances the additional complexity of adding the split at the node against the increase in predictive accuracy that it offers.

## Remark

*A higher complexity parameter leads to more and more nodes being pruned off, resulting in smaller trees.*



Using the details of the CART procedure, we can state the problem that CART attempts to solve as a formal **optimization problem**.

There are two parameters in this problem:

1. The trade-off between accuracy and complexity of the tree:  $\alpha$ ;
2. The minimum number of points we require in any leaf node:  $N_{\min}$ .

## Optimal Tree Problem

Given these parameters and the training data  $(\mathbf{x}_i, y_i)$ ,  $i = 1, \dots, n$ , we seek a tree  $\mathbb{T}$  that solves the problem:

$$\begin{aligned} \min \quad & R(\mathbb{T}) + \alpha |\mathbb{T}| \\ \text{s.t.} \quad & N(l) \geq N_{\min} \quad \forall l \in \text{leaves}(\mathbb{T}) \end{aligned} \tag{P}$$

where  $R(\mathbb{T})$  is the misclassification error of the tree  $\mathbb{T}$  on the training data,  $|\mathbb{T}|$  is the number of branch nodes in tree  $\mathbb{T}$ , and  $N(l)$  is the number of training points contained in leaf node  $l$ .

## Remark

*Notice that if we can solve this problem in a single step we obviate the need to use an impurity measure when growing the tree, and also remove the need to prune the tree after creation, as we have already accounted for the complexity penalty while growing the tree.*

## Remark

*We briefly note that our choice to use CART to define the optimal tree problem was arbitrary, and one could similarly define this problem based on another method like C4.5; we simply use this problem to demonstrate the advantages of taking a problem that is traditionally solved by a heuristic and instead solving it to optimality.*

## Remark

*Additionally, we note that the literature found that CART and C4.5 did not differ significantly in any measure of tree quality, including out-of-sample accuracy, and so we do not believe our choice of CART over C4.5 to be an important one.*

# Decision Trees: A Mixed Integer Optimization Approach

As mentioned previously, the top-down, greedy nature of state-of-the-art DT creation algorithms can lead to solutions that are only **locally optimal**.

The natural way to pose the task of creating the **globally optimal DT** is as a *Mixed Integer Optimization* (MIO) problem.

Indeed, at every step in tree creation, we are required to make a number of discrete decisions:

1. At every new node, we must choose to either branch or stop.
2. After choosing to stop branching at a node, we must choose a label to assign to this new leaf node.
3. After choosing to branch, we must choose which of the variables to branch on.
4. When classifying the training points according to the tree under construction, we must choose to which leaf node a point will be assigned such that the structure of the tree is respected.



# Decision Trees: A Mixed Integer Optimization Approach

Formulating this problem using MIO allows us to model all of these discrete decisions in a single problem.

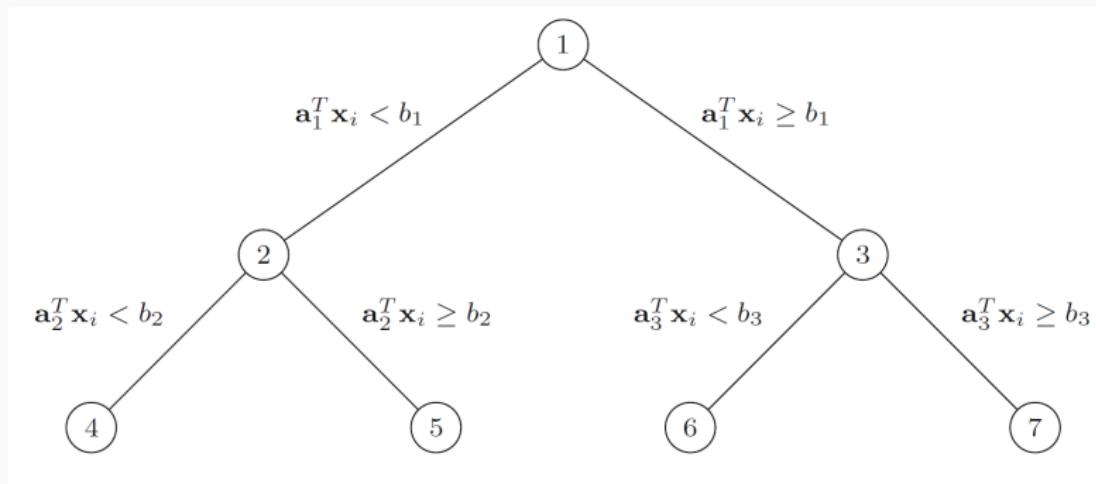
Modeling the construction process in this way allows us to consider the full impact of the decisions being made at the top of the tree, rather than simply making a series of locally optimal decisions, also avoiding the need for pruning and impurity measures.

We will next formulate the optimal tree creation problem (P) as a MIO problem. Consider the problem of trying to construct an optimal DT with a maximum depth of  $D$ . Given this depth, we can construct the maximal tree of this depth, which has  $T = 2^{(D+1)} - 1$  nodes, which we index by  $t = 1, \dots, T$ .



# Decision Trees: A Mixed Integer Optimization Approach

The following figure shows the maximal tree of depth  $D = 2$ .



- Branch nodes: 1, 2 and 3.
- Leaf nodes: 4, 5, 6 and 7.

# Decision Trees: A Mixed Integer Optimization Approach

We use the following notation:

- $p(t)$  refers to the parent node of node  $t$ ;
- $\mathcal{A}(t)$  denotes the set of ancestors of node  $t$ ;
- $\mathcal{L}(t)$  is the set of ancestors of  $t$  whose left branch has been followed on the path from the root node to  $t$ ;
- $\mathcal{R}(t)$  is the set of right-branch ancestors, such that  $\mathcal{A}(t) = \mathcal{L}(t) \cup \mathcal{R}(t)$ .

For example, in the tree of the previous slide:

- $\mathcal{L}(5) = \{1\}$ ,  $\mathcal{R}(5) = \{2\}$ ,
- $\mathcal{A}(5) = \{1, 2\}$ .



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Decision Trees: A Mixed Integer Optimization Approach

We divide the nodes in the tree into two sets:

1. **Branch nodes**: nodes  $t \in \mathcal{T}_B = \{1, \dots, [T/2]\}$  apply a split of the form  $\mathbf{a}^\top \mathbf{x} < b$ .
  - Points that **satisfy** this split follow the **left** branch in the tree...
  - ...and those that **do not** follow the **right** branch.
2. **Leaf nodes**: nodes  $t \in \mathcal{T}_L = \{[T/2] + 1, \dots, T\}$  make a class prediction for each point that falls into the leaf node.

We track the split applied at node  $t \in \mathcal{T}_B$  with variables  $\mathbf{a}_t \in \mathbb{R}^p$  and  $b_t \in \mathbb{R}$ .



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Decision Trees: A Mixed Integer Optimization Approach

## Remark

We restrict our model to **univariate DT** (like CART), and so the hyperplane split at each node should only involve a single variable. This is enforced by setting the elements of  $\mathbf{a}_t$  to be binary variables that sum to 1.

We want to allow the option of not splitting at a branch node. We use the binary variables  $d_t$  to track which branch nodes apply splits:

$$d_t = \begin{cases} 1 & \text{if node } t \text{ applies a split} \\ 0 & \text{otherwise.} \end{cases}$$

If a branch node does not apply a split, then we model this by setting  $\mathbf{a}_t = \mathbf{0}$  and  $b_t = 0$ . This has the effect of forcing all points to follow the right split at this node, since the condition for the left split is  $0 < 0$  which is never satisfied.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Decision Trees: A Mixed Integer Optimization Approach

We enforce this with the following constraints:

$$\sum_{j=1}^p a_{jt} = d_t \quad \forall t \in \mathcal{T}_B \quad (1)$$

$$0 \leq b_t \leq d_t \quad \forall t \in \mathcal{T}_B \quad (2)$$

$$a_{jt} \in \{0, 1\} \quad \forall j = 1, \dots, p, \quad t \in \mathcal{T}_B \quad (3)$$

where the second inequality is valid for  $b_t$ , since we have assumed that each  $\mathbf{x}_i \in [0, 1]^p$ , and we know that at has one element that is 1 if and only if  $d_t = 1$ , with the remainder being 0. Therefore we it is always true that

$$0 \leq \mathbf{a}_t^\top \mathbf{x}_i \leq d_t$$

for any  $i$  and  $t$ , and we need only consider values for  $b_t$  in this same range.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Decision Trees: A Mixed Integer Optimization Approach

Next, we will enforce the hierarchical structure of the tree. We restrict a branch node from applying a split if its parent does not also apply a split.

$$d_t \leq d_{p(t)} \quad \forall t \in \mathcal{T}_B \setminus \{1\}, \quad (4)$$

where no such constraint is required for the root node.

We have constructed the variables that allow us to model the tree structure using MIO; we now need to track the allocation of points to leaves and the associated errors that are induced by this structure.

We introduce the binary variables  $z_{it}$  to track the points assigned to each leaf node, where:

$$z_{it} = \begin{cases} 1 & \text{if point } i \text{ is in node } t \\ 0 & \text{otherwise.} \end{cases}$$



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Decision Trees: A Mixed Integer Optimization Approach

We also introduce binary variables  $l_t$ , where:

$$l_t = \begin{cases} 1 & \text{if leaf } t \text{ contains any point} \\ 0 & \text{otherwise.} \end{cases}$$

We use these binary variables together to enforce a minimum number of points at each leaf, given by  $N_{\min}$ :

$$z_{it} \leq l_t \quad \forall t \in \mathcal{T}_L \quad (5)$$

$$\sum_{i=1}^n z_{it} \geq N_{\min} l_t \quad \forall t \in \mathcal{T}_L. \quad (6)$$

We also force each point to be assigned to exactly one leaf:

$$\sum_{t \in \mathcal{T}_L} z_{it} = 1 \quad \forall i = 1, \dots, n. \quad (7)$$



# Decision Trees: A Mixed Integer Optimization Approach

Finally, we apply constraints enforcing the splits that are required by the structure of the tree when assigning points to leaves:

$$\mathbf{a}_m^\top \mathbf{x}_i < b_m + M_1(1 - z_{it}) \quad \forall i = 1, \dots, n, t \in \mathcal{T}_L, m \in \mathcal{L}(t) \quad (*)$$

$$\mathbf{a}_m^\top \mathbf{x}_i \geq b_m - M_2(1 - z_{it}) \quad \forall i = 1, \dots, n, t \in \mathcal{T}_L, m \in \mathcal{R}(t)$$

where  $M_1, M_2$  are sufficiently large constants such that the constraints are always satisfied when  $z_{it} = 0$ .



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Decision Trees: A Mixed Integer Optimization Approach

Note that the constraints  $(\star)$  use a strict inequality which is not supported by MIO solvers, so this must be converted into a form that does not use a strict inequality. To do this we can add a small constant  $\epsilon$  to the left-hand-side of  $(\star)$  and change the inequality to be non-strict:

$$\mathbf{a}_m^\top \mathbf{x}_i + \epsilon \leq b_m + M_1(1 - z_{it}) \quad \forall i = 1, \dots, n, t \in \mathcal{T}_L, m \in \mathcal{L}(t).$$

However, if  $\epsilon$  is too small, this could cause numerical instabilities in the MIO solver, so we seek to make  $\epsilon$  as big as possible without affecting the feasibility of any valid solution to the problem.

We can achieve this by specifying a different  $\epsilon_j$  for each feature  $j$ . The largest valid value is the smallest non-zero distance between adjacent values of this feature.



# Decision Trees: A Mixed Integer Optimization Approach

To find this, we sort the values of the  $j$ -th feature and take:

$$\epsilon_j = \min \left\{ x_j^{(i+1)} - x_j^{(i)} \mid x_j^{(i+1)} \neq x_j^{(i)}, i = 1, \dots, n-1 \right\}$$

where  $x_j^{(i)}$  is the  $i$ -th largest value in the  $j$ -th feature. We can then use these values for  $\epsilon$  in the constraint, where the value of  $\epsilon_j$  that is used is selected according to the feature we are using for this split:

$$\mathbf{a}_m^\top (\mathbf{x}_i + \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\min}) + \boldsymbol{\epsilon}_{\min} \leq b_m + M_1(1 - z_{it})$$

for all  $i = 1, \dots, n$ ,  $t \in \mathcal{T}_L$ ,  $m \in \mathcal{L}(t)$  and where  $\boldsymbol{\epsilon}_{\min} = \min_j \{\epsilon_j\}$ .



# Decision Trees: A Mixed Integer Optimization Approach

We must also specify values for the big- $M$  constants  $M_1$  and  $M_2$ . As mentioned previously, we know that both  $\mathbf{a}_t^\top \mathbf{x}_i \in [0, 1]$  and  $b_t \in [0, 1]$ , and so the largest possible value of  $\mathbf{a}_t^\top (\mathbf{x}_i + \epsilon) - b_t$  is  $1 + \epsilon_{\max}$ , where  $\epsilon_{\max} = \max_j \{\epsilon_j\}$ . We can therefore set:

$$M_1 = 1 + \epsilon_{\max}.$$

Similarly, we have the largest possible value of  $b_t - \mathbf{a}_t^\top \mathbf{x}_i$  is 1, so we can set:

$$M_2 = 1.$$

This gives the following final constraints that will enforce the splits in the tree:

$$\begin{aligned} \mathbf{a}_m^\top (\mathbf{x}_i + \epsilon - \epsilon_{\min}) + \epsilon_{\min} &\leq \\ b_m + (1 + \epsilon_{\max})(1 - z_{it}) & \quad \forall i = 1, \dots, n, t \in \mathcal{T}_L, m \in \mathcal{L}(t) \end{aligned} \quad (8)$$

$$\mathbf{a}_m^\top \mathbf{x}_i \geq b_m - (1 - z_{it}) \quad \forall i = 1, \dots, n, t \in \mathcal{T}_L, m \in \mathcal{R}(t). \quad (9)$$



# Decision Trees: A Mixed Integer Optimization Approach

The **objective** is to **minimize the misclassification error**, so an incorrect label prediction has cost 1, and a correct label prediction has cost 0.

We set  $N_{kt}$  to be the number of points of label  $k$  in node  $t$ , and  $N_t$  to be the total number of points in node  $t$ :

$$N_{kt} = \sum_{i:y_i=k} z_{it} \quad \forall t \in \mathcal{T}_L, k = 1, \dots, K \quad (10)$$

$$N_t = \sum_{i=1}^n z_{it} \quad \forall t \in \mathcal{T}_L. \quad (11)$$

We need to assign a label to each leaf node  $t$  in the tree, which we denote with  $c_t \in \{1, \dots, K\}$ . It is clear that the optimal label to predict is the most common of the labels among all points assigned to the node:

$$c_t = \arg \max_{k=1, \dots, K} \{N_{kt}\}. \quad (\diamond)$$



# Decision Trees: A Mixed Integer Optimization Approach

We will use binary variables  $c_{kt}$  to track the prediction of each node, where:

$$c_{kt} = \begin{cases} 1 & \text{if } c_t = k \\ 0 & \text{otherwise.} \end{cases}$$

We must make a single class prediction at each leaf node that contains points:

$$\sum_{k=1}^K c_{kt} = l_t \quad \forall t \in \mathcal{T}_L. \quad (12)$$

# Decision Trees: A Mixed Integer Optimization Approach

Since we know how to make the optimal prediction at each leaf  $t$  using  $(\diamond)$ , the optimal misclassification loss in each node, denoted  $L_t$ , is going to be equal to the number of points in the node less the number of points of the most common label:

$$L_t = N_t - \max_{k=1,\dots,K} \{N_{kt}\} = \min_{k=1,\dots,K} \{N_t - N_{kt}\},$$

which can be linearized to give:

$$L_t \geq N_t - N_{kt} - M(1 - c_{kt}) \quad \forall t \in \mathcal{T}_L, k = 1, \dots, K \quad (13)$$

$$L_t \leq N_t - N_{kt} + Mc_{kt} \quad \forall t \in \mathcal{T}_L, k = 1, \dots, K \quad (14)$$

$$L_t \geq 0 \quad \forall t \in \mathcal{T}_L \quad (15)$$

where again  $M$  is a sufficiently large constant that makes the constraint inactive depending on the value of  $c_{kt}$ . Here, we can take  $M = n$  as a valid value.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Decision Trees: A Mixed Integer Optimization Approach

The total misclassification cost is therefore

$$\sum_{t \in \mathcal{T}_L} L_t$$

and the complexity  $C$  of the tree is the number of splits included in the tree, given by:

$$C = \sum_{t \in \mathcal{T}_B} d_t. \quad (16)$$

Following CART, we normalize the misclassification against the baseline accuracy,  $\hat{L}$ , obtained by simply predicting the most popular class for the entire dataset. This makes the effect of  $\alpha$  independent of the dataset size. This means the objective from problem (P) can be written:

$$\min_{\hat{L}} \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_L} L_t + \alpha \cdot C.$$



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

# Decision Trees: A Mixed Integer Optimization Approach

Putting all of this together gives the following MIO formulation for (P).

## The Optimal Classification Trees (OCT) Model

$$\min \quad \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_L} L_t + \alpha \cdot C$$

s.t. (1) to (16)

$$z_{it}, l_t, c_{kt} \in \{0, 1\} \quad \forall i = 1, \dots, n, k = 1, \dots, K, t \in \mathcal{T}_L$$

$$d_t \in \{0, 1\} \quad \forall j = 1, \dots, p, t \in \mathcal{T}_B.$$



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

## Remark

*This model as presented is in a form that can be directly solved by any MIO solver. The difficulty of the model is primarily determined by the number of binary variables  $z_{it}$ , which is  $n \cdot 2^D$ .*

Empirically:

- we observe that we can find high-quality solutions in minutes for depths up to 4 for datasets with thousands of points;
- beyond this depth or dataset size, the rate of finding solutions is slower, and more time is required.



# OCT with Hyperplane Splits

- So far, we have only considered decision trees that use a single variable in their splits at each node, known as univariate decision trees.
- Now, we show that it is simple to extend our MIO formulation for univariate trees to yield a problem for determining the **optimal multivariate DT**.
- This shows the flexibility and power of modeling the problem using MIO.
- In a multivariate DT, we are no longer restricted to choosing a single variable upon which to split, and instead can choose a general **hyperplane split at each node**.



# OCT with Hyperplane Splits

The variables  $\mathbf{a}_t$  will be used to model the split at each node as before, except we relax (3) and instead choose  $\mathbf{a}_t \in [-1, 1]^p$  at each branch node  $t$ .

We must modify (1) to account for the possibility these elements are negative by dealing with the absolute values instead:

$$\sum_{j=1}^p |a_{jt}| \leq d_t \quad \forall t \in \mathcal{T}_B.$$

which can be linearized using auxiliary variables to track the value of  $|a_{jt}|$ :

$$\sum_{j=1}^p \hat{a}_{jt} \leq d_t \quad \forall t \in \mathcal{T}_B \quad (17)$$

$$\hat{a}_{jt} \geq a_{jt} \quad \forall t \in \mathcal{T}_B, j = 1, \dots, p \quad (18)$$

$$\hat{a}_{jt} \geq -a_{jt} \quad \forall t \in \mathcal{T}_B. \quad (19)$$

These constraints force the split to be all zeros if  $d_t = 0$  and no split is applied, otherwise imposing no restriction on  $\mathbf{a}_t$ .

# OCT with Hyperplane Splits

We now have that  $\mathbf{a}_t^\top \mathbf{x}_i \in [-1, 1]$ , so we replace (2) with:

$$-d_t \leq b_t \leq d_t \quad \forall t \in \mathcal{T}_B. \quad (20)$$

Now we consider the split constraints  $(\star)$  and (9). Previously we had that the range of  $(\mathbf{a}_t^\top \mathbf{x}_i - b_t)$  was  $[-1, 1]$ , whereas it is now  $[-2, 2]$ . This means we need  $M = 2$  to ensure that the constraint is trivially satisfied when  $z_{it} = 0$ .

The constraints therefore become:

$$\mathbf{a}_m^\top \mathbf{x}_i < b_m + 2(1 - z_{it}) \quad \forall i = 1, \dots, n, t \in \mathcal{T}_L, m \in \mathcal{L}(t) \quad (*)$$

$$\mathbf{a}_m^\top \mathbf{x}_i \geq b_m - 2(1 - z_{it}) \quad \forall i = 1, \dots, n, t \in \mathcal{T}_L, m \in \mathcal{R}(t). \quad (21)$$



# OCT with Hyperplane Splits

As before, as before we need to convert the strict inequality in (\*) to a non-strict version. We do this by introducing a sufficiently small constant  $\mu$ :

$$\mathbf{a}_m^\top \mathbf{x}_i + \mu \leq b_m + (2 + \mu)(1 - z_{it}) \quad \forall i = 1, \dots, n, t \in \mathcal{T}_L, m \in \mathcal{L}(t). \quad (22)$$

Note that we need to include  $\mu$  in the rightmost term to ensure the constraint is always satisfied when  $z_{ik} = 0$ .

## Remark

- Unlike in the univariate case, we cannot choose a value for  $\mu$  in an intelligent manner, and instead need to choose a small constant.
- We must take care when choosing the value of  $\mu$ . A value too small can lead to numerical issues in the MIO solver, while too large reduces the size of the feasible region, potentially reducing the quality of the optimal solution. We take  $\mu = 0.005$  as a compromise between these extremes.



# OCT with Hyperplane Splits

- In the univariate case, we controlled the complexity of the tree by penalizing the number of splits.
- In the multivariate regime, a single split may use multiple variables, and it seems natural to treat splits with greater number of variables as more complex than those with fewer.
- To achieve this, we can instead penalize the total number of variables used in the splits of the tree, which we note in the univariate case is exactly the number of splits in the tree.

To achieve this, we introduce binary variables  $s_{jt}$  to track if the  $j$ -th feature is used in the  $t$ -th split:

$$-s_{jt} \leq a_{jt} \leq s_{jt} \quad \forall t \in \mathcal{T}_B, j = 1, \dots, p. \quad (23)$$



## OCT with Hyperplane Splits

We must also make sure that the values of  $s_{jt}$  and  $d_t$  are compatible. The following constraints ensure that  $d_t = 1$  if and only if any variable is used in the split:

$$s_{jt} \leq d_t \quad \forall t \in \mathcal{T}_B, j = 1, \dots, p \quad (24)$$

$$\sum_{j=1}^p s_{jt} \geq d_t \quad \forall t \in \mathcal{T}_B. \quad (25)$$

Finally, we modify the definition of complexity  $C$  to penalize the number of variables used across the splits in the tree rather than simply the number of splits:

$$C = \sum_{t \in \mathcal{T}_B} \sum_{j=1}^p s_{jt}. \quad (26)$$



# OCT with Hyperplane Splits

Combining all of these changes yields to:

## The Optimal Classification Trees with Hyperplanes (OCT-H) Model

$$\min_{\hat{L}} \quad \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_L} L_t + \alpha \cdot C$$

$$s.t. \quad (4) \text{ to } (7)$$

$$(10) \text{ to } (15)$$

$$(17) \text{ to } (26)$$

$$z_{it}, l_t, c_{kt} \in \{0, 1\} \quad \forall i = 1, \dots, n, k = 1, \dots, K, t \in \mathcal{T}_L$$

$$d_t, s_{jt} \in \{0, 1\} \quad \forall j = 1, \dots, p, t \in \mathcal{T}_B.$$

### Remark

From this formulation we can easily obtain the OCT problem as a special case by restoring the integrality constraints on  $a_t$ . The close relationship between the univariate and multivariate problems reinforces the notion that the MIO formulation is the natural way to view the decision tree problem.