
# Computational Mathematics For Learning and Data Analysis

## Optimization

Based on prof. Antonio Frangioni's lectures

Gemma Martini

August 27, 2021

A sincere thank you to Alessandro Cudazzo,
Donato Meoli, Giulia Volpi, Ivan Grujic and all those
who helped me improving these notes in style and contents.

# Contents

# Chapter 1

# Introduction

This course will deal with the optimization and numerical analysis of machine learning problems. We are not going to discuss difficult problems (e.g. NP-hard problems), besides we will present an efficient solution for simple ones (often **convex** ones), since we are dealing with huge amount of data.

Let us start with a warm up on machine learning problems.

## 1.1 Introduction to machine learning problems

Machine learning techniques are not as "young" as it might seem, the intuition has been there for ages, but we did not have enough calculus power. Machine learning algorithms have started working well recently, thanks to the many improvements in computer performances; for this reason, it is becoming a more and more popular subject to study.

The main idea behind machine learning is to take a huge amount of data (e.g. frames of a video for object-recognition) and squeeze them, in order to process them. This intuitive concept is translated into mathematical terms as "building a **model**" that fits our data. As in practical engineering problems, people want to construct a model (a small sized representation of the large thing we want to produce in the end) and try to understand its behaviour, before actually building such an object. Let us take as an example the problem of designing a jet. It is not clever to start building the plane before designing a cheap prototype to better study its behaviour in the atmosphere.

The kind of models we want to build are cheap to construct and as close as possible to the real problem we are studying. In physics, people try to find the best mathematical model to describe a real world phenomenon. We will see during this course that the core problem is computation, since the more accurate the model, the more costly the prediction phase. Therefore, a good model is a trade-off between accuracy and simplicity, namely it provides good prediction without incurring in slow computations.

The model, though, has to be parametric: we do not have only one model,

we have a "shape" of a model, which is fit to our problem through the tuning of some parameters.

**Example 1.1.1.** *As an example, we are given three couples:* $f(x_1) = y_1$, $f(x_2) = y_2$, $f(x_3) = y_3$, *as shown in Figure 1.1.*

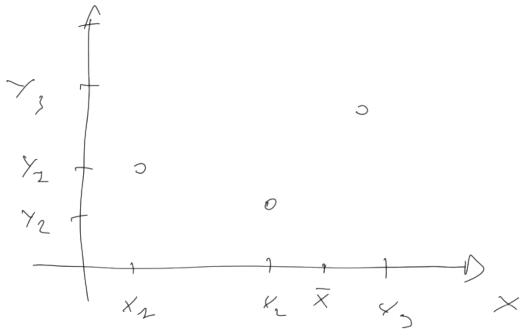


FIGURE 1.1: Geometric representation of the input. We are interested in finding a model that fits the input data and allows to predict $\bar{y}$ out of $\bar{x}$.

*We need to make some choices: first, we need to decide the kind of model we believe is a good approximation of the objective function, say a linear model* $f(x) = ax + b$. *After doing that, we are left with choosing its parameters (in order to pick a line among the whole family of linear functions), namely* $a$ *and* $b$.

We are interested in building a model that fits the data we are given and then tuning the parameters in order to achieve a good accuracy for a given application (recall that the model is parametric, hence the right values for the parameters have to be learned).

Another important characteristic of a good model is that it should not take too long to be built.

In this course we do not concentrate on the problem of finding the model that best fits our data. Our setting, instead, is such that we are given a problem and a model and we perform a study on its behaviour through its parameters. In other words, within the family of models with a given outline, we want to find the one that better represents the phenomena observed. This is called **fitting** and it is clearly some sort of optimization problem, where the fitting task is typically the computational bottleneck.
However, machine learning is more than simply fitting: fitting minimizes training error (or empirical risk), but ML aims at minimizing test error (i.e. generalization error).

At first a machine learning solution builds a model that fits the observed data and then performs an hyper-parameter tuning in order to achieve a good "predicting power" on unseen inputs. In machine learning, a model that fits the data while achieving good performances on new examples is said to be "non **overfitting**".

For machine learning purposes, a mathematical model should be:

- *accurate* (describing well the process under consideration)
- *computationally inexpensive* (providing answers rapidly)
- *general* (it can be applied to many different processes)

It goes without saying that achieving all these goals is practically impossible.

## 1.2 Optimization

In the rest of this lecture we are going to better understand what an optimization problem is, through some intuitive real world examples.

**Terminology**

In this course we will refer to vectors using the bold notation $\mathbf{v} \in \mathbb{R}^n$. The $i$-th entry of vector $\mathbf{v}$ is denoted as $v_i \in \mathbb{R}$, while the $j$-th pattern in the training set is a couple of vectors $(\mathbf{x^j}, \mathbf{y^j})$.
For more details on vectors and vector spaces and matrices see Section 2.4.

### 1.2.1 Linear estimation

Let us consider a phenomenon measured in terms of one real number $y \in \mathbb{R}$ that is believed to depend on a vector $\mathbf{x} = [x_1, \dots, x_n]$. We are provided a set of observations: $\{(y^1, \mathbf{x^1}), \dots, (y^m, \mathbf{x^m})\}$.

**Definition 1.2.1** (Linear model)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be the objective function. We call $\widetilde{f}(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i + w_0 = \mathbf{w_+}^T \mathbf{x} + w_0$ the **linear model** of $f$ for a given set of parameters, which is a vector $\mathbf{w} = (w_0, \mathbf{w_+}) = (w_0, w_1, \dots, w_n) \in \mathbb{R}^{n+1}$.*

How can we evaluate the "similarity" between our model and the objective function? Through computing the "error" or difference between the objective function value and the model prediction on each input. Under this assumption, the error function may be used to find the best parameters for our model, by solving a minimum problem:

**Definition 1.2.2** (Least squares problem)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be the objective function, such that $f(\mathbf{x}) = \mathbf{y}$ and let $X\mathbf{w}$ be our linear model. Then we can find the best values for vector $\mathbf{w} \in \mathbb{R}^{n+1}$ by solving the **least squares problem***

$$\min_{\mathbf{w}} \|\mathbf{y} - X\mathbf{w}\|$$

$$\text{where } Y = \begin{pmatrix} \mathbf{y^1} \\ \vdots \\ \mathbf{y^m} \end{pmatrix} \in M(m, n, \mathbb{R}) \text{ and } X = \begin{pmatrix} 1 & \mathbf{x^1} \\ \vdots & \vdots \\ 1 & \mathbf{x^m} \end{pmatrix} \in M(m, n+1, \mathbb{R}).$$

If the matrix $X$ is invertible then the simple solution is $\mathbf{w} = X^{-1}\mathbf{y}$. The point is that this operation is very costly when dealing with a huge number of entries (in the next paragraph we will see a way to manage it).

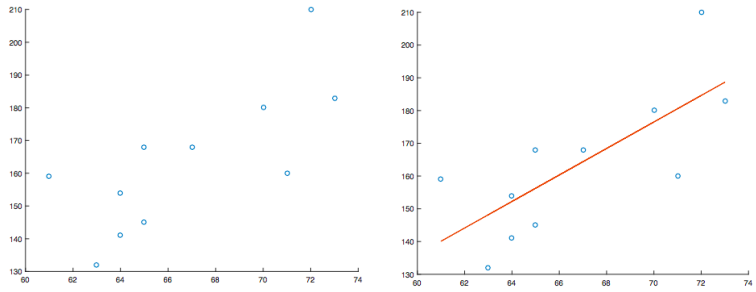


FIGURE 1.2: A linear estimation fitting example

### 1.2.2 Low-rank approximation

A (large, sparse) matrix $M \in M(n, m, \mathbb{R})$ describes a phenomenon depending on pairs (e.g., objects bought by customers) and we may want to approximate that matrix as the product between two smaller matrices (find a few features that describe most of users' choices): a "tall thin" matrix $A \in M(n, k, \mathbb{R})$ and a "fat large" $B \in M(k, m, \mathbb{R})$ (where $k \ll n, m$).

$$\boxed{M} \approx \boxed{A} \cdot \boxed{B}$$

This problem can be translated into a numerical analysis problem of the following shape

$$\min_{A,B} \|M - AB\|$$

The matrices $A$ and $B$ can be obtained from eigenvectors of $M^T M$ and $MM^T$, but matrix $M$ is a huge and possibly dense matrix. Therefore, a more efficient way should be used, which also avoids the explicit formation of $M^T M$ and $MM^T$ because that would need a lot of memory.
Efficiently solving this problem requires:

- low-complexity computation
- avoiding the explicit forming of $M^T M$ and $MM^T$
- exploiting the structure of $M$ (sparsity, similar columns, ...)
- ensuring that the solution is *numerically stable*

Mettere reference a numerically stable.

### 1.2.3 Support vector machines

Let us consider the so-called "decision problem": given a set of values of many parameters (variables) "label" a person $i$ as ill or healthy, formally $y^i \in \{1, -1\}$.

The geometric intuition in two dimensions is given by Figure 1.3. We would like to find the line that better splits the plane into two regions, because this could help to diagnose the next patient. The rationale here is to maximize the space between the line and the nearest points (called **margin**), in order to have a better accuracy.

It is known that the distance between the two hyperplanes $(w_0, \mathbf{w}_+)$ and $(w_0', \mathbf{w}_+)$ in Figure 1.3 is computed as $\frac{2}{\|\mathbf{w}_+\|}$, so in order to maximize the distance between the planes we aim at minimizing $\|\mathbf{w}_+\|$. After a normalization step, for each hyperplane that lies between $(w_0, \mathbf{w}_+)$ and $(w_0', \mathbf{w}_+)$ the following holds

$$\begin{cases} \mathbf{w}_+\mathbf{x}^i + w_0 \geq 1 & \text{if } \ y^i = 1 \\ \mathbf{w}_+\mathbf{x}^i + w_0 \leq -1 & \text{if } \ y^i = -1 \end{cases}$$

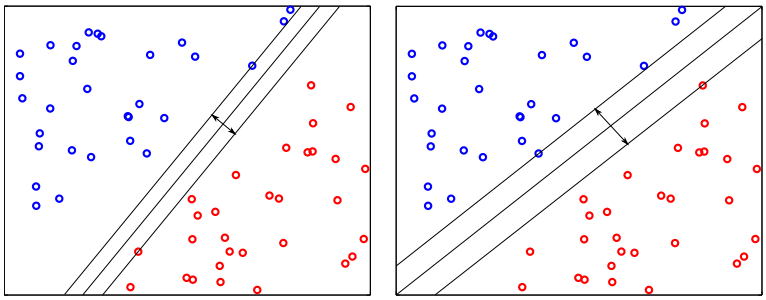


FIGURE 1.3: There are many possible boundaries that can be chosen as a model using many angular coefficients. Our best guess is the one that maximizes the distance between the line and the nearest points.

Under the hypotheses of this optimization problem, the maximum-margin separating hyperplane (assuming that any exists) is the solution of

$$\min_{\mathbf{w}} \{\|\mathbf{w}_+\|^2 \ : \ y^i(\mathbf{w}_+\mathbf{x}^\mathbf{i} + w_0) \geq 1, \ i = 1, \ldots, m\}$$

In real world cases, most of the times there is no such line that splits perfectly the positive examples from the negative ones. To overcome this issue we introduce the concept of "penalty" that accounts for the number of points that are misclassified, through the following

**Definition 1.2.3** (SVM Primal problem)**.** *We term Support Vector Machine primal problem (SVM-P) a convex constrained problem with complex constraints that is formalized as*

$$\begin{cases} \min\limits_{\mathbf{w},\xi} \|\mathbf{w}_+\|^2 + C \sum\limits_{i=1}^{m} \xi_i \\ y^i(\mathbf{w}_+ \mathbf{x}^i + w_0) \geq 1 - \xi_i \,,\, \xi_i \geq 0 \,,\, \forall i = 1, \ldots, m \end{cases} \tag{SVM-P}$$

*Where $C$ is an **hyper-parameter** that weights the violation of the separating margin.*

This definition formalizes the intuition that the approximated function may have a greater norm and lead to a very small misclassification error, or it could be the other way round. Both these solutions are acceptable and their performances depend only on the problem.

Whenever we are able to solve a multi-objective optimization problem we are also able to solve what is called the **dual problem**, which in our case has the following shape.

**Definition 1.2.4** (SVM Dual problem)**.** *We call Support Vector Machine dual problem (SVM-D) a convex constrained quadratic problem defined as*

$$\begin{cases} \max\limits_{\alpha} \sum\limits_{i=1}^{m} \alpha_i - \frac{1}{2} \sum\limits_{i=1}^{m} \sum_{j=1}^{m} \alpha_i < \mathbf{x^i}, \mathbf{x^j} > \alpha_j \\ \sum\limits_{i=1}^{m} y^i \alpha_i = 0 \\ 0 \leq \alpha_i \leq C \,,\, \forall i = 1, \ldots, m \end{cases} \tag{SVM-D}$$

The idea behind the theory of duality is to obtain the solution of a problem by solving an (apparently) different one.

**Lemma 1.2.1.** *Given an optimal solution $\alpha^*$ for the dual problem (SVM-D), then ${\mathbf{w}^*}_+ = \sum_{i=1}^{m} \alpha_i^* y^i \mathbf{x^i}$ is optimal for the primal problem (SVM-P).*

A reader who has a deeper background in the field of machine learning knows that the scalar product in the dual form allows the usage of the so-called "kernel trick". This mathematical transformation allows the mapping of the input space into a larger feature space where the points are more likely to be linearly separable.

Theoretically, the feature space can be infinite-dimensional, provided that the scalar product can be (efficiently) computed.

This whole course has the aim of presenting some techniques for solving efficiently **convex quadratic problems**, as the ones presented above.

# Chapter 2

# Mathematical background for optimization problems

As a warm-up, let us refer to univariate cases. We will move towards multivariate examples later in this lecture.

**Definition 2.0.1** (Minimum problem)**.** *Let $S \subset \mathbb{R}$ be a set, called **feasible region** and let $f : S \to \mathbb{R}$ be* any *function, called **objective function**. We term **minimum problem** the problem of finding the minimum value of such $f$. Formally,*

$$f_* = \min_x \{f(x) \ : x \in S\} \tag{P}$$

> ✪ **Mantra**
>
> In this course we will deal with minimum problems only. Luckily enough, switching from minimum to maximum problems requires only some sign adjustments.

**Definition 2.0.2** (Feasible solution)**.** *Let $F \subseteq \mathbb{R}$ be a proper superset of the feasible region $S \subset F$ and let $x \in F$ be a solution of the minimum problem (P). We say that $x$ is a **feasible solution** if $x \in S$. Conversely, $x$ is **unfeasible** if $x \in F \setminus S$.*

**Definition 2.0.3** (Optimal solution)**.** *Under the same hypothesis of the above definition, we call $x_*$ that realizes $f(x_*) = f_*$ an **optimal solution**, where $f_* \leq f(x) \, \forall x \in S$ and $\forall v > f_* \, \exists \, x \in S$ s.t. $f(x) < v$.*

It is possible to find problems where there is no optimal solution at all.

**Observation 2.0.1.** *There are two cases in which it is not possible to find an optimal solution of a minimum problem:*

1. *the domain is empty, which may be not trivial to prove, since it is an NP-hard problem sometimes;*

2. *the objective function is unbounded below ($\forall M \exists x_M \in S$ s.t. $f(x_M) \leq M$).*

**Example 2.0.1** (Bad optimization problems)**.** *The following practical cases are examples of problems that do not admit an optimal solution:*

- *empty case ($S = \emptyset$):* $\min\{f(x) = x \ : \ x \in \mathbb{R} \wedge x \leq -1 \wedge x \geq 1\}$
- *unbounded (below):* $\min\{f(x) = x \ : \ x \in \mathbb{R} \wedge x \leq 0\}$
- *bad $f$ and $S$:* $\min\{f(x) = x \ : \ x \in \mathbb{R} \wedge x > 0\}$
- *bad $f$: let us consider an iterative algorithm that moves towards the optimum. It may happen that the function decreases and increases along a certain direction, such function is not continuous so it is impossible to reach the optimum:*

$$\min \left\{ f(x) = \begin{cases} x & \text{if } x > 0 \\ 1 & \text{if } x = 0 \end{cases} \quad x \in [0,1] \right\}$$

Solving an optimization problem can be one of the following procedures:

1. Finding $x_*$ and proving it is optimal
2. Proving $S = \emptyset$
3. Constructively proving that the function is unbounded ($\forall M \exists x_M \in S$ s.t. $f(x_M) \leq M$).

In this course we are interested in applying the theoretical concept to the numerical world. Therefore, in our terms, "$x \in \mathbb{R}$" actually means "$x \in \mathbb{Q}$" with up to k digits precision and most of the times we consider optimal a solution which is close to the true optimal value, modulo some error (we call $\bar{x}$, the approximated optimal).

**Definition 2.0.4** (Absolute error)**.** *We call **absolute error** the gap between the real value and the one we obtained. Formally,*

$$f(\bar{x}) - f_* \leq \varepsilon$$

**Definition 2.0.5** (Relative error)**.** *We term **relative error** the absolute error, normalized by the true value of the function*

$$(\, f(\bar{x}) - f_* \,)/|\, f_* \,| \leq \varepsilon$$

## 2.1 Multi-objective Optimization

It may happen that there are more than one function that need to be minimized (maximized) and they could be contrasting or have incomparable units, for example buying a painting which is the most beautiful and the cheapest simultaneously.

In multi-objective optimization, typically there is not a feasible solution that minimizes all objective functions at the same time. It is in this context that *Pareto optimal solutions* are introduced: intuitively, such solutions cannot be improved in any of the objectives without degrading at least one of the other objectives.

**Definition 2.1.1** (Pareto dominance)**.** *Let $S \subseteq \mathbb{R}$ and $f_1, \ldots, f_k : S \to \mathbb{R}$ and let the minimum multi-objective optimization problem be formalized as*

$$\min_x \{[f_1(x), \ldots, f_k(x)] : x \in S\}$$

*A feasible solution $x^1 \in S$ is said to **(Pareto) dominate** another solution $x^2 \in S$, if $x^1$ "beats" $x^2$ in any of the objective functions. Formally,*

$$f_i(x^1) \leq f_i(x^2) \quad \forall i \in \{1, 2, \ldots, k\}$$

**Definition 2.1.2** (Pareto frontier)**.** *A solution $x^* \in S$ (and the corresponding outcome $f(x^*)$) is called **Pareto optimal**, if there does not exist another solution that dominates it. The set of Pareto optimal outcomes is often called the **Pareto frontier**.*

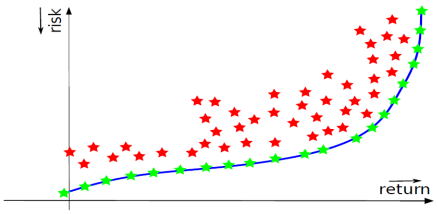



FIGURE 2.1: An example of Pareto frontier

**Example 2.1.1.** *Let us take two functions $f_1$ and $f_2$. We want to solve the minimization problem*

$$\min_x \{[f_1(x), f_2(x)] \ : x \in S\} \tag{P}$$

*where $f_1$ accounts for return of an investment and $f_2$ accounts for the risk. For solving this problem, we present two different approaches:*

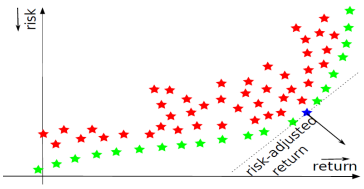
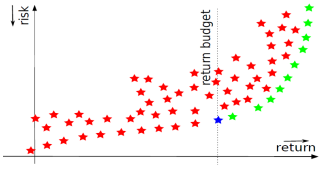


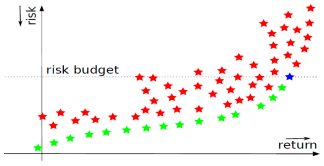

FIGURE 2.2: Maximize risk-adjusted return

SCALARIZATION. *Using a linear combination of the two functions, formally,* $f(x) = \alpha f_1(x) + \beta f_2(x)$. *An example, where* $\alpha = 1$ *is shown in Figure 2.2.*

BUDGETING. *Intuitively corresponds to taking into account only one objective function and add the others as constraints, provided that the values of the other functions are not too high. In formal terms,* $f(x) = f_1(x)$, *where we modify the admissible region* $S := S \cup \{x \in S \ : \ f_2(x) \leq b\}$. *As an example see Figure 2.3.*




(a) Maximize return with budget on maximum risk, $\min\{f_1(x) \ : \ f_2(x) \leq \beta \ : x \in S\}$




(b) Minimize risk with budget on minimum return, $\min\{f_2(x) \ : \ f_1(x) \leq \beta \ : x \in S\}$

FIGURE 2.3: Budgeting

## 2.2 Infima, suprema and extended reals

Let us introduce some mathematical background on minimization (maximization).

**Definition 2.2.1** (Totally ordered set)**.** *We say that a set* $S$ *is* ***totally ordered*** *if* $\forall\, x, y \in S$, *either* $f(x) \leq f(y)$ *or* $f(y) \leq f(x)$.

**Definition 2.2.2** (Infima and suprema)**.** *Let* $R$ *be a totally ordered set and let* $S$ *be one of its subsets* ($S \subseteq R$)*:*

$\underline{s}$ *is the* **infimum** *of* $S(\underline{s} = \inf S)$ *iff* $\underline{s} \leq s \ \ \forall s \in S \ \ \wedge \ \ \forall t > \underline{s} \ \exists\, s \in S \ s.t. \ s \leq t$

$\bar{s}$ *is the* **supremum** *of* $S(\bar{s} = \sup S)$ *iff* $\bar{s} \geq s \ \ \forall s \in S \ \wedge \ \forall t < \bar{s} \ \exists\, s \in S \ s.t. \ s \geq t$

An attentive reader may notice that not all subsets of $\mathbb{R}$ allow infima and suprema.

**Definition 2.2.3** (Extended real)**.** *In the case of unbounded functions the value of infima or suprema are $\infty$, and we call* ***extended reals*** $\overline{\mathbb{R}} = \{-\infty\} \cup \mathbb{R} \cup \{+\infty\}$.

**Property 2.2.1.** *The following holds for the extended reals:*

- *For all $S \subseteq \mathbb{R}, \sup/\inf S \in \overline{\mathbb{R}}$*
- $\inf S = -\infty$ *indicates that there is no (finite) infimum*
- $\inf \emptyset = \infty$, $\sup \emptyset = -\infty$

## 2.3 Sequences in $\mathbb{R}$ and optimization

We are interested in studying sequences, because iterative methods can be seen as sequences of points: we start from a certain point and move towards the optimum.

**Terminology**

We denote a sequence of iterates as $\{x_i\} \subset S$ and we denote the plugging a function $f$ into such a sequence as $v_i = f(x_i)$.

**Definition 2.3.1** (Limit)**.** *Given a sequence $\{x_i\}$ the* ***limit*** *for $i \to \infty$ is defined as*

$$\lim_{i \to \infty} v_i = v \iff \forall \varepsilon > 0 \; \exists \; h \; s.t. \; |v_i - v| \leq \varepsilon \; \forall i \geq h$$

It may happen that a sequence has or does not have a limit. For example $\{\frac{1}{n}\}$ has limit 0 for $n \to +\infty$, while $\{(-1)^n\}$ does not have any.

**Fact 2.3.1.** *Let us be given a monotone sequence, then the sequence* ***does*** *have a limit.*

Notice that a sequence either is monotone or it can be "split" into two monotone sequences.

**Example 2.3.1.** *Let us consider the sequence $\{(-1)^n\}$. It does not converge to any value, but it can be split into $\{(-1)^{2n}\}$ and $\{(-1)^{2n+1}\}$ and these two subsequences are both monotone.*

In general, monotonicity on sequences in $\mathbb{R}$ can be performed by splitting the sequence into a non-decreasing sequence and a non-increasing sequence. Formally, given a sequence $\{v_i\} \subset S$ we obtain $\{\underline{v}_i\} \subset \{v_i\} \subset S$ and $\{\bar{v}_i\} \subset \{v_i\} \subset S$ such that $\underline{v}_1 \leq \underline{v}_2 \leq \ldots$ and $\bar{v}_1 \geq \bar{v}_2 \geq \ldots$. It also holds that $\underline{v}_i^* = \inf\{v_h : h \leq i\}$ and $\bar{v}_i^* = \sup\{v_h : h \leq i\}$. $\{\underline{v}_i\}$ and $\{\bar{v}_i\}$ have a limit and they are formalized as $\liminf_{i \to \infty} v_i \lim_{i \to \infty} \underline{v_i}^* = v_\infty^* = f_*$ and $\limsup_{i \to \infty} v_i \lim_{i \to \infty} \bar{v}_i^* = v_\infty^* = f_*$.

**Fact 2.3.2.** *The following holds:*

- $\bar{v}_i \geq \underline{v}_i \Rightarrow \limsup_{i\to\infty} v_i \geq \liminf_{i\to\infty} v_i$
- $\lim_{i\to\infty} v_i = v \iff \limsup_{i\to\infty} v_i = v = \liminf_{i\to\infty} v_i$

## 2.4 Vector spaces and topology

In our setting, real numbers are used to describe each feature of a dataset and this has the mathematical equivalent in the term "vector".

**Definition 2.4.1** (Euclidean vector space)**.** *We call $n$-**dimensional Euclidean vector space** the set of columns made of $n$ real numbers. Formally,*

$$\mathbb{R}^n := \left\{ \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \ : \ x_i \in \mathbb{R},\ i = 1, \dots, n \right\}$$

*Equivalently, we can characterize the Euclidean space as Cartesian product of $\mathbb{R}$ $n$ times: $\mathbb{R}^n = \overbrace{\mathbb{R} \times \mathbb{R} \times \dots \mathbb{R}}^{n}$. Every vector space and in particular Euclidean spaces are closed under sum and scalar multiplication. The elements of a vector space are called **vectors**.*

The main operations on elements of the Euclidean space (vectors) are:

$$\text{Sum: } \mathbf{x} + \mathbf{y} := \begin{pmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{pmatrix}$$

$$\text{Scalar multiplication: } \alpha\mathbf{x} = \begin{pmatrix} \alpha x_1 \\ \alpha x_2 \\ \vdots \\ \alpha x_n \end{pmatrix}$$

**Terminology**

In linear algebra, $\mathbf{x} \in \mathbb{R}^n$ is a column vector. Whenever a row vector is needed for dimensionality issues it is denoted as $\mathbf{x}^T$.

**Definition 2.4.2** (Basis)**.** *Any vector $\mathbf{x} \in \mathbb{R}^n$ can be obtained from a linear combination of a set of vectors that is called "basis". In Euclidean spaces there exists a special basis that is called "canonical", where each vector $\mathbf{u^i}$ has a 1 in position $i$ and a 0 elsewhere.*

**Definition 2.4.3** (Finite vector space)**.** *Let $V(\mathbb{K}, n, m)$ (it is read "a vector space of dimensions $n$ and $m$ on a field $\mathbb{K}$") be a vector space. It is said to be **finite** if its bases have a finite cardinality.*

At this point, it is crucial to observe that not all vector spaces are finite nor they are a totally ordered set.
In this course we will deal with the concept of limit over a vector space very often and for this purpose we are going to introduce a topology on vector spaces (i.e. norm, scalar product, distance).

### 2.4.1 Topology in $\mathbb{R}^n$

**Definition 2.4.4** (Standard scalar product)**.** *Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ we define the **standard scalar product** between these two vectors as*

$$< \mathbf{x}, \mathbf{y} > := \mathbf{y}^T\mathbf{x} = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + \cdots + x_n y_n$$

**Fact 2.4.1.** *A scalar product has the following properties:*

1. $< \mathbf{x}, \mathbf{y} >=< \mathbf{y}, \mathbf{x} > \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ *(symmetry)*
2. $< \mathbf{x}, \mathbf{x} > \geq 0,\ \forall \mathbf{x} \in \mathbb{R}^n,\ < \mathbf{x}, \mathbf{x} >= 0 \iff \mathbf{x} = \mathbf{0}$;
3. $< \alpha\mathbf{x}, \mathbf{y} >= \alpha < \mathbf{x}, \mathbf{y} >,\ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \forall \alpha \in \mathbb{R}$;
4. $< \mathbf{x} + \mathbf{y}, \mathbf{z} >=< \mathbf{x}, \mathbf{z} > + < \mathbf{y}, \mathbf{z} >,\ \forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$.

An important geometric characterization of the scalar product is the one that uses angles:

$$< \mathbf{x}, \mathbf{y} >= \|\mathbf{x}\|\,\|\mathbf{y}\| \cos\theta$$

According to this new definition, we can observe that $\mathbf{x}$ and $\mathbf{y}$ are orthogonal ($\mathbf{x} \perp \mathbf{y}$) iff $< \mathbf{x}, \mathbf{y} >= 0$ and in all the other cases they somehow point in the same direction.

We could rewrite the scalar product between $\mathbf{x}$ and $\mathbf{y}$ as $\mathbf{y}^T I\mathbf{x}$, where $I \in D(\mathbb{R}, n)^*$ is the identity matrix. Thanks to this observation. we can state the more general form of the scalar product:

**Definition 2.4.5** (Scalar product)**.** *Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and let $M \in M(\mathbb{R}, n)$ be a positive definite matrix (for further details see). We call **scalar product** the following*

$$< \mathbf{x}, \mathbf{y} >_M := \mathbf{y}^T M\mathbf{x}$$

Aggiungere reference al punto in cui si definisce una matrice definita positiva

*We denote with $D(\mathbb{R}, n)$ the vector space of all $n \times n$ diagonal matrices on the Euclidean field $\mathbb{R}$.

**Definition 2.4.6** (Euclidean distance)**.** *The **Euclidean distance** between two points* $\mathbf{x}$ *and* $\mathbf{y}$ *in* $\mathbb{R}^n$ *is the length of the line segment connecting them. Formally, let us take two points in the* $n$*-dimensional Euclidean space* $\mathbf{x}^T = (x_1, x_2, \ldots, x_n)$ *and* $\mathbf{y}^T = (y_1, y_2, \ldots, y_n)$*, then the distance (*$d$*) from* $\mathbf{x}$ *to* $\mathbf{y}$*, or equivalently from* $\mathbf{y}$ *to* $\mathbf{x}$*, is given by*

$$d(\mathbf{x}, \mathbf{y}) := \|\mathbf{x} - \mathbf{y}\| = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}$$

**Fact 2.4.2.** *The Euclidean distance has the following properties:*

1. $d(\mathbf{x}, \mathbf{y}) \geq 0 \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n \ , \ d(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y}$
2. $d(\alpha\mathbf{x}, 0) = |\alpha| d(\mathbf{x}, 0) \ \forall \mathbf{x} \in \mathbb{R}^n, \ \forall \alpha \in \mathbb{R}$
3. $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y}) \ \forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$ *(triangle inequality)*

**Definition 2.4.7** (Norm)**.** *Given a vector space* $V$ *over a field* $\mathbb{K}$ *of the real or complex numbers, a **norm** on* $V$ *is a nonnegative-valued function* $p : V \to \mathbb{R}$ *with the following properties:*

1. *For all* $a \in \mathbb{K}$ *and for all* $\mathbf{u}, \mathbf{v} \in V$ $p(\mathbf{u}+\mathbf{v}) \leq p(\mathbf{u})+p(\mathbf{v})$ *(being subadditive or satisfying the triangle inequality);*
2. $p(a\mathbf{v}) = |a|p(\mathbf{v})$ *(being absolutely homogeneous or absolutely scalable);*
3. *If* $p(\mathbf{v}) = 0$ *then* $\mathbf{v} = \mathbf{0}$ *is the zero vector (being positive definite or being point-separating).*

**Definition 2.4.8** (Euclidean norm)**.** *Let* $\mathbf{x} \in \mathbb{R}^n$*, we define **euclidean norm** of* $\mathbf{x}$ *the square root of the squared sum of all its entries. Formally,*

$$\|\mathbf{x}\|_2 := \sqrt{\sum_{i=1}^{n} {x_i}^2} = \sqrt{< \mathbf{x}, \mathbf{x} >}$$

**Definition 2.4.9** (Frobenius' norm)**.** *Let* $A \in M(n, \mathbb{R})$*, we define **Frobenius' norm** of* $A$ *the square root of the squared sum of all its entries. Formally,*

$$\|A\|_F := \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} {a_{ij}}^2}$$

**Fact 2.4.3.** *Any norm on a vector space has the following properties:*

1. $\|\mathbf{x}\| \geq 0 \ \forall \mathbf{x} \in \mathbb{R}^n$ *and* $\|\mathbf{x}\| = 0 \iff \mathbf{x} = \mathbf{0}$*;*
2. $\|\alpha\mathbf{x}\| = |\alpha| \, \|\mathbf{x}\| \, , \ \forall \mathbf{x} \in \mathbb{R}^n, \ \forall \alpha \in \mathbb{R}$*;*
3. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\| \, , \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ *(triangle inequality).*

**Fact 2.4.4** (Cauchy-Schwartz's inequality)**.** *Let* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$*. The following holds:*

$< \mathbf{x}, \mathbf{y} >^2 \leq < \mathbf{x}, \mathbf{x} >< \mathbf{y}, \mathbf{y} >$ *or, equivalently* $|< \mathbf{x}, \mathbf{y} >| \leq \|\mathbf{x}\|_2 \|\mathbf{y}\|_2$, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

**Fact 2.4.5** (Parallelogram Law)**.**

$$2\|\mathbf{x}\|^2 + 2\|\mathbf{y}\|^2 = \|\mathbf{x}+\mathbf{y}\|^2 + \|\mathbf{x}-\mathbf{y}\|^2$$

**Corollary 2.4.6.**

$$\|\mathbf{x}+\mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 + 2 < \mathbf{x}, \mathbf{y} >$$

*Proof.*

$$2\|\mathbf{x}\|^2 + 2\|\mathbf{y}\|^2 = \|\mathbf{x}+\mathbf{y}\|^2 + \|\mathbf{x}-\mathbf{y}\|^2$$

iff

$$\begin{aligned}
\|\mathbf{x}+\mathbf{y}\|^2 &= 2\|\mathbf{x}\|^2 + 2\|\mathbf{y}\|^2 - \|\mathbf{x}-\mathbf{y}\|^2 \\
&= \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 + \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - \|\mathbf{x}-\mathbf{y}\|^2 \\
&= \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 + \sum_{i=1}^{n} {x_i}^2 + \sum_{i=1}^{n} {y_i}^2 - \sum_{i=1}^{n} (x_i - y_i)^2 \\
&= \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 + \sum_{i=1}^{n} {x_i}^2 + {y_i}^2 - {x_i}^2 - {y_i}^2 + 2x_i y_i \\
&= \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 + 2 \cdot \underbrace{\sum_{i=1}^{n} x_i y_i}_{<\mathbf{x},\mathbf{y}>}
\end{aligned} \tag{2.4.1}$$

□

The 2-norm is a special case of the more general $p$-norm.

**Definition 2.4.10** (p-norm)**.** *Let* $p \geq 1$ *be a real number, the p-norm of a vector* $\mathbf{x} \in \mathbb{R}^n$ *is defined as follows:*

$$\|\mathbf{x}\|_p := \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

**Observation 2.4.1.** *We require* $p \geq 1$ *for the general definition of the p-norm because the triangle inequality fails to hold if* $p < 1$*.*

**Fact 2.4.7.** *The p-norm is convex for* $p \geq 1$*.*

*Proof.* For this proof we need the definition of convexity, which is a concept that will be expanded in Section 3.4, more precisely in Definition 3.4.9. By properties 2 and 3 of Proposition 2.4.3, we get the following:

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

□

The most important $p$-norms are the following:

- $\|\mathbf{x}\|_1 := \sum_{i=1}^{n} |x_i|$
- $\|\mathbf{x}\|_\infty := \max\{|x_i| \ : \ i = 1, \ldots, n\}$
- $\|\mathbf{x}\|_i := |\{i \ : \ |x_i| > 0\}|$

**Fact 2.4.8.** *For any given finite-dimensional vector space $V(\mathbb{K}, n)$ (e.g. $\mathbb{R}^n$), all norms on $V$ are* equivalent *(therefore, convergence in one norm implies convergence in any other norm). Formally, $\forall \|\cdot\|_A$ , $\|\cdot\|_B$:*

$$\exists\, 0 < \alpha < \beta \ \ s.t \ \ \alpha \|\mathbf{x}\|_A \leq \|\mathbf{x}\|_B \leq \beta \|\mathbf{x}\|_A \qquad \forall \mathbf{x} \in V$$

This rule may not apply in infinite-dimensional vector spaces such as function spaces. For such cases we define the

**Fact 2.4.9** (Holder's inequality)**.** *Let $V(\mathbb{K}, n)$ be a (possibly infinite) vector space and let $p, q \in [1, \infty)$ with $1/p + 1/q = 1$. The* ***Holder's inequality*** *states that*

$$< \mathbf{x}, \mathbf{y} >^2 \leq \|\mathbf{x}\|_p \|\mathbf{y}\|_q$$

**Definition 2.4.11** (Ball)**.** *Let $\bar{\mathbf{x}} \in \mathbb{R}^n$. We term* ***ball*** *centered in $\bar{\mathbf{x}}$ and having $\varepsilon \in \mathbb{R}$ as radius as the set of points that are close enough to $\mathbf{x}$: $B(\bar{\mathbf{x}}, \varepsilon) = \{\mathbf{x} \in \mathbb{R}^n \ : \ \|\mathbf{x} - \bar{\mathbf{x}}\| \leq \varepsilon\}$.*

Moreover, we term *unit ball* a ball where $\varepsilon = 1$. An attentive reader may notice that a unit ball has different shapes for different real values of $p$ for the norm. In Figure 2.4 we may observe the different shapes of the same ball varying the value $p$ in the $p$-norm.

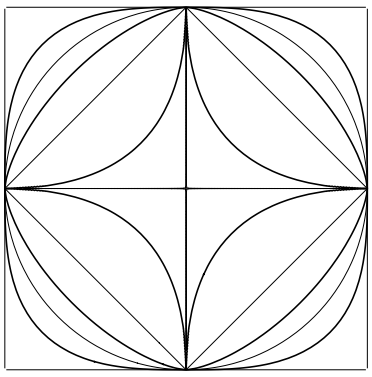


FIGURE 2.4: The shapes of balls centered in the origin of radius 1 varying the value of $p$-norm.

## 2.5 Limit of a sequence in $\mathbb{R}^n$

We have now all the tools to define the notion of limit of a sequence in $\mathbb{R}^n$.

**Definition 2.5.1** (Limit of a sequence in the Euclidean space)**.** *Let $\{\mathbf{x_i}\} \subset \mathbb{R}^n$ be a sequence in $\mathbb{R}^n$. We say that $\{\mathbf{x_i}\}$ converges to a **limit** $\mathbf{x}$ for $i \to +\infty$ and denote $\lim_{i\to\infty} \mathbf{x_i} = \mathbf{x}$ or $\{\mathbf{x_i}\} \to \mathbf{x}$ if $\forall \varepsilon > 0\ \exists h$ s.t. $d(\mathbf{x_i}, \mathbf{x}) \leq \varepsilon\ \forall i \geq h$*
*or, equivalently, $\forall \varepsilon > 0\ \exists h$ s.t. $\mathbf{x_i} \in \mathcal{B}(\mathbf{x}, \varepsilon)\ \forall i \geq h$*
*or, even $\lim_{i\to\infty} d(\mathbf{x_i}, \mathbf{x}) = 0$.*

The definition of limit formalizes the idea that the points of $\{\mathbf{x_i}\}$ eventually all come arbitrarily close to $\mathbf{x}$. Moreover, since $\mathbb{R}^n$ is not totally ordered, there is no obvious lim inf nor lim sup.

**Definition 2.5.2** (Minimizing sequence)**.** *Let $\{x_i\} \in S$ be a sequence and let $f : S \to \mathbb{R}^n$. We say that $\{x_i\}$ is a **minimizing sequence** if the sequence of function values $\{f(x_i)\}$ tends to the* greatest lower bound *of $f$ ($f_* = \min\{f(x)\ :\ x \in S\}$).*

**Example 2.5.1.** *As an example, consider the following minimum problems and sequences:*

$$\min\{f(x) = x\ :\ x \in \mathbb{R} \wedge x > 0\} \text{ and the sequence } \{x_i = 1/i\}$$

$$\min\{f(x) = 1/x\ :\ x \in \mathbb{R} \wedge x > 0\} \text{ and the sequence } \{x_i = i\}$$

*In both cases, $\{f(x_i)\} \to 0$, but it is not an optimum.*

We want *conditions* that ensure that if $\{f(x_i)\}$ converges to a number ($\{f(x_i)\} \to f_*$) then $\{x_i\} \to x_* \in S$ is an optimal solution.

**Definition 2.5.3** (Interior and border of a set)**.** *Given $S \subseteq \mathbb{R}^n$, we say that $\mathbf{x}$ is an **interior point** ($\mathbf{x} \in int(S)$) if it lies inside a ball contained in $S$. Formally, $\exists r > 0$ s.t. $\mathcal{B}(\mathbf{x},\ r) \subseteq S$.*

*Moreover, we term **border points** those $\mathbf{x} \in \partial(S)$ such that all the points in the ball centered in $\mathbf{x}$ lie for a half inside the set $S$ and for the other half outside. Formally, $\forall r > 0\ \exists \mathbf{y}, \mathbf{z} \in \mathcal{B}(\mathbf{x},\ r)$, where $\mathbf{y} \in S \wedge \mathbf{z} \notin S$.*

Notice that a point on the boundary is not necessarily inside the ball, in that case we talk about **open set** (a set which is identical to its interior: $S = int(S)$).

**Definition 2.5.4** (Closure of a set)**.** *Let $S \subseteq \mathbb{R}^n$ we term **closure** of $S$ the set $cl(S) = int(S) \cup \partial S$.*

**Definition 2.5.5** (Closed set)**.** *We say that a set $S \subseteq \mathbb{R}^n$ is **closed** if it coincides with its closure: $S = cl(S)$.*

*Equivalently, a set $S$ is termed **closed** if its complementary ($\mathbb{R}^n \backslash S$) is open.*

It is interesting to notice that all the functions that lead to minimizing sequences and do not converge to an optimum are all defined in open sets.

Notice that there are sets that are both open and closed, for example $\mathbb{R}^n$.

**Definition 2.5.6** (Bounded set)**.** *Let $S \subseteq \mathbb{R}^n$. We say that $S$ is **bounded** if $\exists r > 0$ such that $S \subseteq \mathcal{B}(\mathbf{0}, r)$.*

*Intuitively, a bounded set does not go to $\infty$.*

**Definition 2.5.7** (Compact set)**.** *Let $S \subseteq \mathbb{R}^n$. We term $S$ **compact** if it is both closed and bounded.*

**Definition 2.5.8** (Accumulation point)**.** *Let $\{x_i\} \subseteq S$ be a sequence. We say that $x$ is an **accumulation point** if $\exists$ $\{x_{n_i}\}$ subsequence of $\{x_i\}$ converging to $x$. Formally, $\{x_{n_i}\} \to x$ or $\liminf_{i\to\infty} d(x_{n_i},\ x) = 0$.*

**Theorem 2.5.1** (Bolzano-Weierstrass)**.** *Let $\{x_i\} \subseteq S$ be a* bounded, real *sequence. Then it has a converging subsequence.*

**Fact 2.5.2.** *Let $\{x_i\} \subseteq S$ be a bounded sequence of real numbers. Then $\{x_i\}$ has at least one accumulation point.*

*Proof.* Since $\{x_i\}$ is bounded, by the Bolzano-Weierstrass theorem, $\{x_i\}$ contains a convergent subsequence $\{x_{n_i}\}$. Suppose that $\{x_{n_i}\}$ converges to a values $x \in \mathbb{R}$. Then $x$ is an accumulation point for $\{x_i\}$. □

**Fact 2.5.3.** *Let $S$ be a compact set and let $\{x_i\} \subseteq S$ be a minimizing sequence for the objective function $f$. The limit of the sequence is a feasible solution.*

*Proof.* Thanks to Proposition 2.5.2 $\{x_i\}$ has an accumulation point $x \in S$. Since it is minimizing $f(x)$ is feasible. □

Why did we say *feasible* but not *optimal*? If the function is not continuous (cfr. Figure 2.5) it may happen that the sequence is minimizing, but the limit is not the optimum.

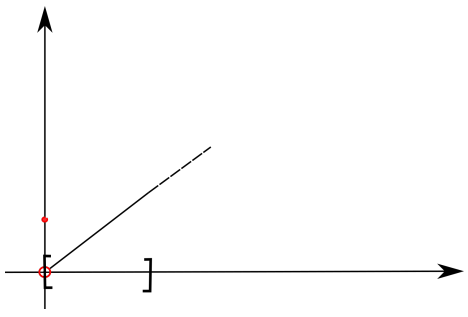


FIGURE 2.5: Case of non-continuity of the objective function in the border point $(0, 0)$.

**Definition 2.5.9** (Domain)**.** *Let $f : D \to \mathbb{R}$. We term $D$ **domain** of $f$ and denote $D = dom(f)$.*

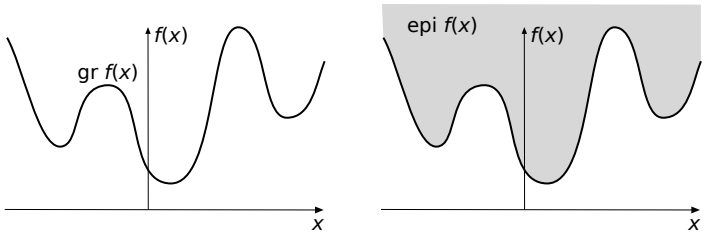
**Mantra**

In this course we do not take into account the domain of functions, because we can force all functions to be defined in the whole space $\mathbb{R}^n$ as follows:

$$f : \mathbb{R}^n \to \bar{\mathbb{R}}, \text{ where } f(x) = \begin{cases} \infty \text{ for } x \notin D \\ f(x) \text{ otherwise} \end{cases}$$

**Definition 2.5.10** (Graph and epigraph)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. We term **graph** the set of ordered couples representing the function value in a point and the point itself. Formally, $gr(f) = \{(f(\mathbf{x}),\ \mathbf{x}) \ :\ \mathbf{x} \in dom(f)\}$.*

*Conversely, we term **epigraph** the set of ordered couples representing all the points that are* above *the function value. Formally, $epi(f) = \{(v,\ \mathbf{x}) \ :\ \mathbf{x} \in dom(f) \ \wedge \ v \geq f(\mathbf{x})\}$, see Figure 2.6.*




(a) Graph (b) Epigraph

FIGURE 2.6: Graph and epigraph

When dealing with multidimensional inputs spaces it is crucial to have mathematical tools that make possible to somehow "see" the function's behaviour in a lower-dimensional space.

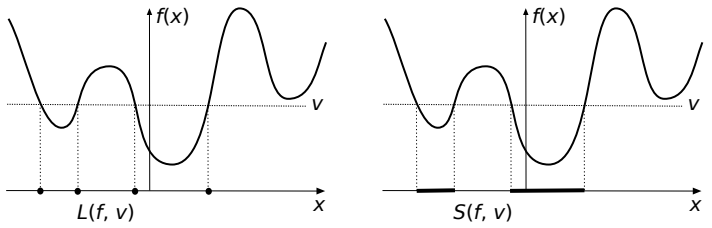
**Definition 2.5.11** (Level and sublevel set)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. We term **level set** the set of all the inputs that have the same output value. Formally, $L(f, v) = \{\mathbf{x} \in dom(f) \ :\ f(\mathbf{x}) = v\}$.*

*Conversely, we term **sublevel set** the set of all the inputs which image is smaller than a fixed value $v$: $S(f, v) = \{\mathbf{x} \in dom(f) \ :\ f(\mathbf{x}) \leq v\}$, see Figure 2.7.*

## 2.6 Continuity

**Definition 2.6.1** (Continuity)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ and let $\mathbf{x} \in \mathbb{R}^n$. We say that $f$ is **continuous** in $\mathbf{x}$ if $\forall \varepsilon > 0\ \exists\, \delta > 0$ such that $\forall \mathbf{y} \in \mathcal{B}(\mathbf{x},\ \delta)\ |f(\mathbf{y}) - f(\mathbf{x})| < \varepsilon$.*

**Property 2.6.1.** *Let $f, g : \mathbb{R}^n \to \mathbb{R}$ and let $\mathbf{x} \in \mathbb{R}^n$ such that $f$ and $g$ are continuous in $\mathbf{x}$. The following holds:*




(a) Level set (b) Sub-level set

FIGURE 2.7: Level and sub-level sets

1. $f + g$, $f \cdot g$ *continuous at* $\mathbf{x}$
2. $\max\{f,\ g\}$ *and* $\min\{f,\ g\}$ *continuous at* $\mathbf{x}$
3. $f \circ g \equiv f(g(\cdot))$ *continuous at* $\mathbf{x}$

**Theorem 2.6.2** (Intermediate value)**.** *Let* $f : \mathbb{R} \to \mathbb{R}$*.* $f$ *is continuous on* $[a,\ b]$ *if* $\forall v \in \mathbb{R}$ *s.t.* $\min\{f(a),\ f(b)\} \leq v \leq \max\{f(a),\ f(b)\}\ \exists\ c \in [a,\ b]$*:* $f(c) = v$*.*

**Theorem 2.6.3** (Weierstrass extreme value theorem)**.** *Let* $S \subseteq \mathbb{R}^n$ *be a compact set and let* $f$ *be continuous on* $S$*. Then* $f$ *must attain a maximum and a minimum, therefore* $(P)$ *has an optimal solution.*

*Equivalently, let* $S \subseteq \mathbb{R}^n$ *compact and let* $f$ *continuous on* $S$*. Then all accumulation points of any minimizing sequence are optima and* there is at least one*.*

**Definition 2.6.2** (Lipschitz continuity)**.** *Let* $f : \mathbb{R}^n \to \mathbb{R}$*. We term* $f$ ***Lipschitz continuous*** *(L.c.) on* $S \subseteq \mathbb{R}^n$ *if* $\exists L > 0$ *such that*

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L \|\mathbf{x} - \mathbf{y}\| \ \ \forall \mathbf{x}, \mathbf{y} \in S$$

*More generally, we say that* $f$ *is* ***globally Lipschitz continuous*** *when* $S = \mathbb{R}^n$ *and it is* ***locally Lipschitz continuous*** *at* $\mathbf{x}$ *if* $\exists\ \varepsilon > 0\ S = \mathcal{B}(\mathbf{x},\ \varepsilon)$*.*

Notice that Lipschitz continuity offers a stronger form of continuity and that the $L$ constant value depends on $S$. The wider the set the smaller $L$.

**Fact 2.6.4.** *Let* $f : \mathbb{R}^n \to \mathbb{R}$*. On a compact set* $S \subseteq \mathbb{R}^n$ *if* $f$ *is continuous then it is Lipschitz continuous.*

**Definition 2.6.3** (Lower(upper) semi-continuity)**.** *Let* $\{\mathbf{x_i}\} \subseteq \mathbb{R}^n$ *be a sequence with accumulation point in* $\mathbf{x}$ *and let* $f : \mathbb{R}^n \to \mathbb{R}$*.* $f$ *is* ***lower (upper) semi-continuous*** *(l.(u.)s.c.) at* $\mathbf{x}$ *if* $f(\mathbf{x}) \leq \liminf_{i \to \infty} f(\mathbf{x_i})$ *(*$f(\mathbf{x}) \geq \limsup_{i \to \infty} f(\mathbf{x_i})$*).*

*Equivalently,* $\liminf \mathbf{y} \to \mathbf{x} f(\mathbf{y}) \geq f(\mathbf{x})$*,* $\limsup_{\mathbf{y} \to \mathbf{x}} f(\mathbf{y}) \leq f(\mathbf{x})$*.*

## 2.7 Derivatives

In this section we address the problem of inferring information on a complicated function around a certain value $\bar{x}$ using very simple functions, that are able to provide reliable information around that $\bar{x}$. Those functions are called "derivatives".
Let us start with a brief recap on single-dimension differentiability.

**Definition 2.7.1** (Left and right derivative)**.** *Let $f : \mathbb{R} \to \mathbb{R}$. We term **left derivative** $f'_-(x) = \lim_{t \to 0_-} [f(x+t) - f(x)]/t$.*

*Conversely, **right derivative** $f'_+(x) = \lim_{t \to 0_+} [f(x+t) - f(x)]/t$.*

**Definition 2.7.2** (Differentiable)**.** *Let $f : \mathbb{R} \to \mathbb{R}$. We say that $f$ is **differentiable** at $x \in dom(f)$ if both left and right derivatives exist and coincide. Formally, $\exists f'_-, f'_+$ and $f'_-(x) = f'_+(x)$.*

**Fact 2.7.1.** *Let $f : \mathbb{R} \to \mathbb{R}$ be differentiable in $x \in dom(f)$ then $f$ is continuous in $x$.*

**Theorem 2.7.2** (Mean value theorem)**.** *Let $f : \mathbb{R} \to \mathbb{R}$ continuous on $[a, b]$ and differentiable on $(a, b)$ then $\exists\, c \in (a, b)$ s.t. $f'(c) = (\, f(b) - f(a)\,)/(b - a)$.*

**Theorem 2.7.3** (Rolle's theorem)**.** *Let $f : \mathbb{R} \to \mathbb{R}$. If $f(a) = f(b)$ then $\exists\ c \in (a, b)$ s.t. $f'(c) = 0$*

**Corollary 2.7.4.** *In the same hypothesis of Rolle's theorem, let $a$ and $b$ consecutive roots of $f$. Then $f'(a)$ and $f'(b)$ have opposite sign.*

### 2.7.1 Multivariate differentiability

**Definition 2.7.3** (Partial derivative)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. We term **partial derivative** of $f$ w.r.t. $x_i$ at $\mathbf{x} \in \mathbb{R}^n$ as*

$$\frac{\partial f}{\partial x_i}(\mathbf{x}) = \lim_{t \to 0} \frac{f(x_1,\ ,\ldots,\ x_{i-1},\ x_i + t,\ x_{i+1},\ \ldots,\ x_n) - f(\mathbf{x})}{t}$$

*In other words, it is just $f'(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n)$, considering each component $x_j$ where $j \neq i$ as constant.*

**Definition 2.7.4** (Gradient)**.** *Given $f : \mathbb{R}^n \to \mathbb{R}$. We term **gradient** of $f$ the vector of all the partial derivatives and we denote $\mathbb{R}^n \ni \nabla f = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n})^T$.*

The gradient is the direction on which the function increases more rapidly, while the opposite of the gradient suggests the direction on which the function decreases the most.

**Definition 2.7.5** (Directional derivative)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. The **directional derivative** represents the rate of change of $f$ from a point $\mathbf{x} \in \mathbb{R}^n$ along a specific direction $\mathbf{d} \in \mathbb{R}^n$. Formally,*

$$\frac{\partial f}{\partial \mathbf{d}}(x) := \lim_{t \to 0} \frac{f(\mathbf{x} + t\mathbf{d}) - f(\mathbf{x})}{t}$$

Notice that in a multivariate space it is not possible to assure differentiability checking if the directional derivatives are equal, because there is an infinite number of different directions.

Let us now introduce the notion of multivariate differentiability.

**Definition 2.7.6** (Differentiable)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. We say that $f$ is **differentiable** at $\mathbf{x} \in \mathbb{R}^n$ if $\exists$ a linear function $\phi(\mathbf{h}) =< \mathbf{c}, \mathbf{h} > +f(\mathbf{x})$ s.t.*

$$\lim_{\|h\| \to 0} \frac{\|f(\mathbf{x}+\mathbf{h}) - \phi(\mathbf{h})\|}{\|\mathbf{h}\|} = 0$$

*The intuition here is that the linear function $\phi$ should approximate $f$ pretty well around $\mathbf{x}$.*

**Fact 2.7.5.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable at $\mathbf{x}$. Then $f$ is locally Lipschitz continuous at $\mathbf{x}$.*

**Corollary 2.7.6.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable at $\mathbf{x}$. Then $f$ is continuous at $\mathbf{x}$.*

Notice that the converse does not hold.

**Fact 2.7.7.** *Let $f : \mathbb{R}^n \to \mathbb{R}$, let $\mathbf{x} \in \mathbb{R}^n$ and let us assume $\exists \delta > 0$ s.t. $\forall i$ $\frac{\partial f}{\partial x_i}(\mathbf{y})$ is continuous $\forall \mathbf{y} \in \mathcal{B}(\mathbf{x}, \delta)$.*

*Then $f$ differentiable at point $\mathbf{x}$.*

Notice that the converse of Proposition 2.7.7 does not hold either.

We are now ready to introduce a class of functions that allows good results for optimization: the $\mathcal{C}^1$ class.

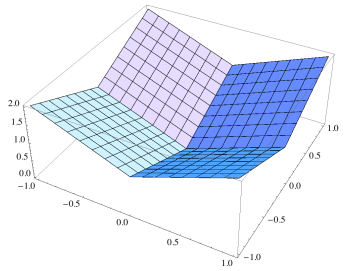
**Definition 2.7.7** ($\mathcal{C}^1$ class)**.** *We call **$\mathcal{C}^1$-class** the class of functions with continuous gradient.*

**Fact 2.7.8.** *Let $f \in \mathcal{C}^1$, then $f$ is differentiable everywhere and also continuous everywhere.*

**Definition 2.7.8** (First order model)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ and let $\mathbf{x} \in dom(f)$ We term **first order model** of $f$ at point $\mathbf{x} \in \mathbb{R}^n$*

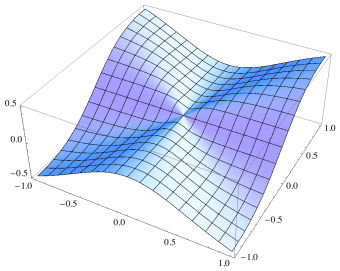
$$L_{\mathbf{x}}(\mathbf{y}) = \nabla f(\mathbf{x})(\mathbf{y} - \mathbf{x}) + f(\mathbf{x})$$

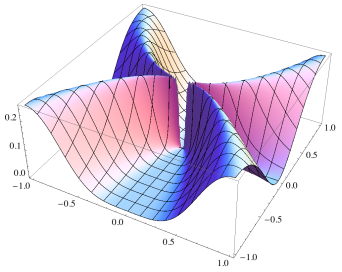
**Example 2.7.1.** *Let us consider some functions that are not differentiable in the minimum.*




*The function $f_1(x_1, x_2) = \|[x_1, x_2]\| = |x_1| + |x_2|$ has some kinks, that are points where the function is not smooth, hence points where the directional derivatives are not defined.*

*The function* $f_2(x_1, x_2) = \frac{{x_1}^2 x_2}{{x_1}^2+{x_2}^2}$ *may be put to* 0 *in* $(0,0)$ *for continuity, but it is still not differentiable in* $(0,0)$ *although the function admits directional derivatives and looks kind of smooth.*





*The function* $f_3(x_1, x_2) = \left(\frac{{x_1}^2 x_2}{{x_1}^4+{x_2}^2}\right)^2$ *is not continuous in* 0. *The directional derivatives exist and they are equal, but still* $f$ *is non differentiable because of the non continuity.*

In general, we look at the gradient in an $n+1$-dimensional space, where $n$ dimensions are needed for the input and the spare dimension is used for the function value (in Example 2.7.1 the input + output space was $\mathbb{R}^3$).

It is possible to get information about the gradient staying in a $n$-dimensional space, using the level sets: all the points that have the same output value (as defined in Definition 2.5.11).

**Terminology**

In $\mathbb{R}^n$ the concept of hyperplane that is tangent to the level sets of the function $f$ in a certain point $\mathbf{x}$ is denoted as $S(L_{\mathbf{x}}, f(\mathbf{x}))$ and it holds that is orthogonal to the gradient and also to the hyperplane tangent to the function.

**Fact 2.7.9.** *Let* $f : \mathbb{R}^n \to \mathbb{R}$ *and let* $\mathbf{x} \in dom(f)$ *such that* $f$ *is differentiable at* $\mathbf{x}$. *Then* $(L_{\mathbf{x}}, f(\mathbf{x})) \perp S(f, f(\mathbf{x})) \perp \nabla f(\mathbf{x})$.

Geometrically speaking, we can observe that a function is smooth whenever its level sets are smooth.

**Fact 2.7.10.** *Let* $f : \mathbb{R}^n \to \mathbb{R}$ *and let* $\mathbf{x} \in dom(f)$ *such that* $f$ *is differentiable at* $\mathbf{x}$. *The tangent to the level sets with respect to the variable* $\mathbf{x}$ *is smooth.*

Le us take $f_2$ of Example 2.7.1, where $\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{2x_1 {x_2}^3}{({x_1}^2+{x_2}^2)^2} \\ \frac{{x_1}^2({x_1}^2-{x_2}^2)}{({x_1}^2+{x_2}^2)^2} \end{pmatrix}$. In Figure 2.8 we can observe that whenever the function is non differentiable, say $\bar{\mathbf{x}}$, the hyperplane tangent to the level sets has some kinks and the surroundings of $\bar{\mathbf{x}}$ appear to be less and less smooth the closer we get ot $\bar{\mathbf{x}}$.

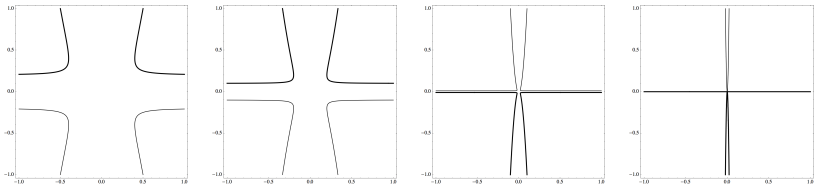


FIGURE 2.8: Shape of the level sets when $\mathbf{x} \to \bar{\mathbf{x}}$.

**Example 2.7.2** (On derivatives)**.** *Let $f : \mathbb{R}^2 \to \mathbb{R}$ such that $f(x, y) = x^2 e^y$. Compute the partial derivatives and the directional derivatives in the two directions $\mathbf{d_1} = (0, 1)^T$ and $\mathbf{d_2} = (1, 0)^T$.*

- $\frac{\partial f}{\partial x} = 2xe^y$
- $\frac{\partial f}{\partial y} = x^2 e^y$
- $\frac{\partial f}{\partial \mathbf{d_1}} = \lim_{t \to 0} \frac{f(x+t\cdot 0, y+t\cdot 1)}{t} = \lim_{t \to 0} \frac{f(x,y+t)}{t}$. *An attentive reader may notice that the directional derivative in direction $\mathbf{d_1}$ is equivalent to the derivative on the second component.*
- $\frac{\partial f}{\partial \mathbf{d_2}} = \lim_{t \to 0} \frac{f(x+t\cdot 1, y+t\cdot 0)}{t} = \lim_{t \to 0} \frac{f(x+t,y)}{t}$. *Conversely, with respect to what stated before, the directional derivative of $f$ along the direction $\mathbf{d_2}$ is equivalent to the partial derivative w.r.t the first component.*

*Let us compute the scalar product between the gradient and the direction $\mathbf{d_1}$:*

$$\frac{\partial f}{\partial \mathbf{d_1}} =< \begin{pmatrix} 2xe^y \\ x^2 e^y \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} >= \begin{pmatrix} 2xe^y \cdot 0 \\ x^2 e^y \cdot 1 \end{pmatrix} = \begin{pmatrix} 0 \\ x^2 e^y \end{pmatrix}$$

The intuition of this example is formalized in the following

**Fact 2.7.11.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. The directional derivative along a certain direction $\mathbf{d} \in \mathbb{R}^n$ can be computed as the scalar product between the gradient of the function and the direction, formally*

$$\frac{\partial f}{\partial \mathbf{d}} =< \nabla f, \mathbf{d} >$$

Until now we only considered real valued function, namely those that go from generic $\mathbb{R}^n$ to $\mathbb{R}$. We can generalize this definition and consider the following

**Definition 2.7.9** (Vector-valued function)**.** *A function which codomain is multi-dimensional is called **vector-valued function**. Formally, $f : \mathbb{R}^n \to \mathbb{R}^m$, where $f(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{pmatrix} \in \mathbb{R}^m$.*

For such functions the computation of the derivative requires to specify not only the component with respect to the one the derivation should be performed, but also the index of the function.

**Definition 2.7.10** (Partial derivative for vector-valued functions)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$, the partial derivative of the $j$-th function with respect to the $i$-th component is*

$$\frac{\partial f_j}{\partial x_i}(\mathbf{x}) = \lim_{t\to 0} \frac{f_j(x_1, x_2, \dots, x_{i-1}, \dots, x_i + t, x_{i+1}, \dots, x_n) - f_j(\mathbf{x})}{t}$$

*where $t \in \mathbb{R}$.*

**Definition 2.7.11** (Jacobian)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a vector-valued function. We call **Jacobian** the matrix of all its first-order partial derivatives.*

$$Jf(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \nabla f_1(\mathbf{x}) \\ \nabla f_2(\mathbf{x}) \\ \vdots \\ \nabla f_m(\mathbf{x}) \end{pmatrix} \in M(m, n, \mathbb{R})$$

The second order derivative of a scalar-valued function is a matrix, defined as

**Definition 2.7.12** (Hessian)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$, we call **Hessian** of $f$*

$$\nabla^2 f(\mathbf{x}) := J\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_2 \partial x_1}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1}(\mathbf{x}) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_2^2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_2}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_2 \partial x_n}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(\mathbf{x}) \end{pmatrix}$$

**Example 2.7.3.** *Let $f : \mathbb{R}^2 \to \mathbb{R}$ such that $f(x, y) = x^2 e^y$ as before, where the gradient was*

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} 2xe^y \\ x^2 e^y \end{pmatrix}$$

*Let us compute the second derivative of this function:*

$$\nabla^2 f(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x \partial x} & \frac{\partial^2 f}{\partial y \partial x} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y \partial y} \end{pmatrix} = \begin{pmatrix} 2e^y & 2xe^y \\ 2xe^y & x^2 e^y \end{pmatrix}$$

The size of the matrix grows very rapidly with the number of derivatives that we make (real valued function: 1 number, gradient: 2 numbers, Hessian 4 numbers).

In optimization, handling Hessians is a crucial task, because such matrices are very large hence unfeasible most of the times, we will see in this course how to gather information about the Hessian without actually computing it.

In most cases, the Hessian is a symmetric matrix, as stated in the following

**Theorem 2.7.12** (Schwartz's theorem)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ such that $\nabla^2 f(\mathbf{x})$ exists and it is continuous and let us take $\mathbf{x} \in \mathbb{R}^n$. $\exists \delta > 0$ s.t. $\forall \mathbf{x}' \in \mathcal{B}(\mathbf{x}, \delta)$ and $\forall i, j \in \{1, 2, \ldots, n\}$ $\frac{\partial^2 f}{\partial x_j \partial x_i}(\mathbf{x}')$ and $\frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}')$ exist and they are equal:*

$$\frac{\partial^2 f}{\partial x_j \partial x_i}(\mathbf{x}') = \frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}')$$

**Definition 2.7.13** ($\mathcal{C}^2$ functions)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$. We say that $f$* ***belongs to $\mathcal{C}^2$ class*** *iff $\nabla^2 f(x)$ is continuous.*

**Corollary 2.7.13.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ such that $f \in \mathcal{C}^2$, then the Hessian is* symmetric *and the gradient is* continuous*.*

Notice that the Hessian being symmetric means that its eigenvalues are real.

Aggiungere reference sulla parte di analisi numerica ed autovalori

We are now interested in providing some approximations to a generic function $f$. Such approximations are useful tools to have a hint on how to move along a decreasing direction, but the information is accurate only locally.

**Definition 2.7.14** (First order Taylor's model)**.** *Let $f : \mathbb{R}^n \to \mathbb{R} \in \mathcal{C}^1$, we define the* ***first-order Taylor's formula*** *as a* linear *approximation of the function $f$ in a neighborhood of $\mathbf{x}$.*
*Formally, for a given ball $\mathcal{B}(\mathbf{x}, \delta)$, $\forall \mathbf{x}' \in \mathcal{B}(\mathbf{x}, \delta)$ $\exists$ $\alpha \in (0, 1)$ s.t.*

$$f(\mathbf{x}') \approx L_{\mathbf{x}}(\mathbf{x}') =< \nabla f(\alpha \mathbf{x} + (1 - \alpha)\mathbf{x}'), \mathbf{x}' - \mathbf{x} > + f(\mathbf{x})$$

*Equivalently, we can write the so-called* ***remainder version of first-order Taylor formula*** *as*

$$f(\mathbf{x}') \approx L_{\mathbf{x}}(\mathbf{x}') =< \nabla f(\mathbf{x}), \mathbf{x}' - \mathbf{x} > + f(\mathbf{x}) + R_2(\mathbf{x}') \tag{2.7.1}$$

*where $R_2$ is called* ***quadratic remainder*** *that is such that $\lim_{h \to 0} \frac{R(h)}{\|h\|} = 0$.*

*Notice the general Lagrangian form of the remainder:*

$$R_k(\mathbf{x}') = \frac{f^{(k+1)}(\beta)}{(k+1)!} \cdot (\mathbf{x}' - \mathbf{x})^2$$

*where $\beta \in [\mathbf{x}, \mathbf{x}']$ and $f$ is $k+1$ times differentiable.*

Notice that the Taylor's approximation works *only locally*: the furthest we get from $\mathbf{x}$ the more distant the function value is from the model.

**Definition 2.7.15** (Second order Taylor's model)**.** *Let $f : \mathbb{R}^n \to \mathbb{R} \in \mathcal{C}^1$, we define the* ***second-order Taylor's formula*** *as a quadratic approximation of the function $f$ in a neighborhood of $\mathbf{x}$. Formally, given a ball $\mathcal{B}(\mathbf{x}, \delta)$, $\forall \mathbf{x}' \in \mathcal{B}(\mathbf{x}, \delta)$*

$$f(\mathbf{x}') \approx Q_{\mathbf{x}}(\mathbf{x}') = L_{\mathbf{x}}(\mathbf{x}') + \frac{1}{2}(\mathbf{x}' - \mathbf{x})^T \nabla^2 f(\mathbf{x})(\mathbf{x}' - \mathbf{x}) + R_3(\mathbf{x}' - \mathbf{x}) \tag{2.7.2}$$

*with* $\lim_{h\to 0} R(h)\,\|h\|^2 = 0$ *or, equivalently, the remainder vanishes at least cubically, or the error is* $O(\|\mathbf{x}' - \mathbf{x}\|^3)$ .

*An equivalent form of the second order Taylor's formula is the following*

$$\forall \mathbf{x}' \in B(\mathbf{x},\delta)\ \exists \alpha \in (0,1)\ s.t.\ f(\mathbf{x}') = L_{\mathbf{x}}(\mathbf{x}') + \frac{1}{2}(\mathbf{x}'-\mathbf{x})^T \cdot \nabla^2 f(\alpha\mathbf{x} + (1-\alpha)\mathbf{x}')(\mathbf{x}'-\mathbf{x}) \tag{2.7.3}$$

**Example 2.7.4.** *Let* $f : \mathbb{R}^2 \to \mathbb{R}$ *such that* $f(\mathbf{x}) = f\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = {x_1}^2 e^{x_2}$, *as above and* $\nabla f(\mathbf{x}) = \begin{pmatrix} 2x_1 e^{x_2} \\ {x_1}^2 e^{x_2} \end{pmatrix}$ *and* $\nabla^2 f(\mathbf{x}) = \begin{pmatrix} 2e^{x_2} & 2x_1 e^{x_2} \\ 2x_1 e^{x_2} & {x_1}^2 e^{x_2} \end{pmatrix}$. *In* $\mathbf{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \in \mathbb{R}^2$ *we have* $f(\mathbf{x}) = 1$ *and* $\nabla f(\mathbf{x}) = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$.

*Compute the first and second order Taylor's models for* $f$ *with* $\alpha = 1$.

*The linear model has the following shape*

$$L_{(1,0)}(x_1,x_2) = < \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} x_1 - 1 \\ x_2 - 0 \end{pmatrix} > +1 = 2x_1 - 2 + x_2 + 1 = 2x_1 + x_2 - 1$$

*And the quadratic model is*

$$f(\mathbf{x}') = < \nabla f\left(\alpha \cdot \begin{pmatrix} 2 \\ 1 \end{pmatrix} + (1-\alpha)\mathbf{x}'\right), \mathbf{x}' - \begin{pmatrix} 2 \\ 1 \end{pmatrix} > + f\begin{pmatrix} 2 \\ 1 \end{pmatrix} + \frac{1}{2} \cdot \nabla^2 f \begin{pmatrix} 2 \\ 1 \end{pmatrix} \cdot \left(\mathbf{x}' - \begin{pmatrix} 2 \\ 1 \end{pmatrix}\right)$$

$$\begin{aligned}
Q_{[1,0]}(x_1,x_2) &= 2x_1 + x_2 - 1 + \frac{1}{2} \cdot \begin{pmatrix} x_1 - 1, & x_2 - 0 \end{pmatrix} \cdot \begin{pmatrix} 2 & 2 \\ 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 - 1 \\ x_2 - 0 \end{pmatrix} \\
&= 2x_1 + x_2 - 1 + \frac{1}{2} \cdot \begin{pmatrix} 2x_1 - 2 + 2x_2, & 2x_1 - 2 + x_2 \end{pmatrix} \cdot \begin{pmatrix} x - 1 \\ x_2 - 0 \end{pmatrix} \\
&= 2x_1 + x_2 - 1 + \frac{1}{2} \cdot \Big( (2x_1 - 2 + 2x_2) \cdot (x_1 - 1) + (2x_1 - 2 + x_2) \cdot x_2 \Big) \\
&= 2x_1 + x_2 - 1 + \frac{1}{2} \cdot (2{x_1}^2 - 2x_1 + 2x_1x_2 - 2x_1 + 2 - 2x_2 + 2x_1x_2 - 2x_2 + {x_2}^2) \\
&= 2x_1 + x_2 - 1 + \frac{1}{2} \cdot (2{x_1}^2 - 4x_1 + 4x_1x_2 + 2 - 4x_2 + {x_2}^2) \\
&= 2x_1 + x_2 - 1 + {x_1}^2 - 2x_1 + 2x_1x_2 + 1 - 2x_2 + \frac{1}{2}{x_2}^2) \\
&= {x_1}^2 + 2x_1x_2 - x_2 + \frac{1}{2}{x_2}^2
\end{aligned} \tag{2.7.4}$$

The Taylor's model can be extended to the more general $k$-th order. Such an expansion requires the computation of the $k$-th order derivatives, but $\nabla^k f(\mathbf{x})$ is a tensor of order $k$, which means computing and storing $n^k$ numbers. For $k > 2$ this approach is practically unfeasible.

As it happens often in computer science, it is important to find a good trade-off between the complexity of the model and the accuracy of the approximation: if the model is too simple (but efficient) the approximation is not very good; conversely, complex and accurate approximations are computationally heavy.

**Fact 2.7.14.** *Let $f : \mathbb{R}^n \to \mathbb{R} \in \mathcal{C}^1$. If $f$ is Lipschitz continuous on $S \in \mathbb{R}^n$, then in such set the norm of the gradient is bounded by the Lipschitz constant.*

*Formally,* $\sup\{\|\nabla f(\mathbf{x})\| \ : \ \mathbf{x} \in S\} \leq L$. *If $S$ convex* $\sup\{\|\nabla f(\mathbf{x})\| \ : \ \mathbf{x} \in S\} = L$.

Moreover, we can prove

**Fact 2.7.15.** *Let $f : \mathbb{R}^n \to \mathbb{R} \in \mathcal{C}^1$. $f$ is Lipschitz continuous on $S \subset \mathbb{R}^n$ iff its Hessian is bounded on $S$. Formally,* $\sup\{\left\|\nabla^2 f(\mathbf{x})\right\| \ : \ \mathbf{x} \in S\} \leq L$.

**Fact 2.7.16.** *Let $f : \mathbb{R}^n \to \mathbb{R} \in \mathcal{C}^1$. If the gradient of $f$ is Lipschitz continuous in all points $\mathbf{x}' \in \mathbb{R}^n$ that are close to $\mathbf{x} \in \mathbb{R}^n$, then $f(\mathbf{x}') \leq L_{\mathbf{x}}(\mathbf{x}') + \frac{L}{2}\|\mathbf{x}' - \mathbf{x}\|^2$.*

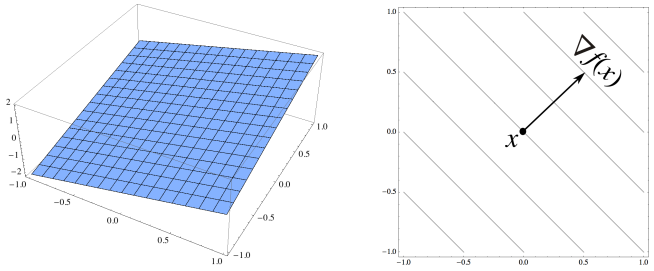
## 2.8 Simple functions

In the rest of the course we will deal mostly with some linear and quadratic approximations of the objective function. In this section, we introduce a couple of examples and considerations on such functions.

### 2.8.1 Linear functions

In this scenario, $f : \mathbb{R}^n \to \mathbb{R}$ has the following shape: $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$, for a fixed $\mathbf{c} \in \mathbb{R}^n$.

It holds that $\nabla f(\mathbf{x}) = \mathbf{c}$, $\nabla^2 f(\mathbf{x}) = 0$ and that level sets are parallel hyperplanes orthogonal to $\mathbf{c}$.




(a) Linear function (b) Level sets

FIGURE 2.9: Graphical example of linear function.

Such functions do not have any minima nor maxima, since they go to $-\infty$ and $+\infty$. Therefore, the question of where is a minimum on such functions is ill posed and this is why we do not stick with linear models only.

### 2.8.2 Quadratic functions

A quadratic function $f : \mathbb{R}^n \to \mathbb{R}$ is formalized as $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T\mathbf{x}$, for fixed $Q \in \mathcal{M}(n, \mathbb{R})$, $\mathbf{q} \in \mathbb{R}^n$. For such functions, the analytic expression for the gradient if $\nabla f = Qx + q$, while the Hessian $\nabla^2 f = Q$.

**Example 2.8.1.** *Let us take* $q = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ *and three different values for* $Q$*:*

- $Q_1 = \begin{pmatrix} 6 & -2 \\ -2 & 6 \end{pmatrix}$. *In this case the function has the shape of a bowl and its level sets are ellipsoids as shown in Figure 2.10(a).*
- $Q_2 = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix}$. *This* $Q$ *matrix leads to a shape that looks like a sheet held from to opposite corners and its level sets are degenerated ellipsoids, where one axis is elongated to* $+\infty$*, as shown in Figure 2.10(b).*
- $Q_3 = \begin{pmatrix} -2 & 6 \\ 6 & -2 \end{pmatrix}$. *This last* $Q$ *matrix induces a quadratic function which looks like a bowl in one direction, but in the other one the bowl is turned upside-down. The level sets are hyperboloids as shown in Figure 2.10(c).*

**Fact 2.8.1.** *Let* $f : \mathbb{R}^n \to \mathbb{R}$ *be a quadratic function with parameters* $Q \in \mathcal{M}(n, \mathbb{R})$ *and* $\mathbf{q}^T \in \mathbb{R}^n$*. If* $Q$ *is symmetric, then it has spectral decomposition. Formally, it can be written as* $H\Lambda H$*, where* $H$ *is the matrix which columns are the eigenvectors and* $\Lambda$ *is the diagonal matrix containing the eigenvalues.*

In the rest of the course we will assume for simplicity that $Q$ is symmetric, thanks to the following trick

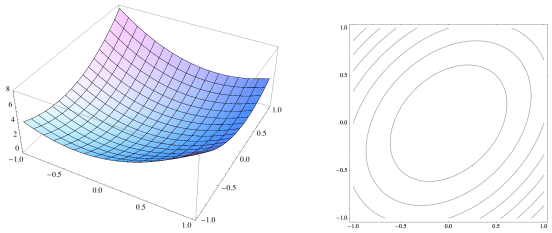
$$\mathbf{x}^T Q\mathbf{x} = \frac{[(\mathbf{x}^T Q\mathbf{x}) + (\mathbf{x}^T Q\mathbf{x})^T]}{2} = \mathbf{x}^T[\frac{(Q + Q^T)}{2}]\mathbf{x}$$

that allows us to use the matrix $\frac{Q+Q^T}{2}$ only during implementation.

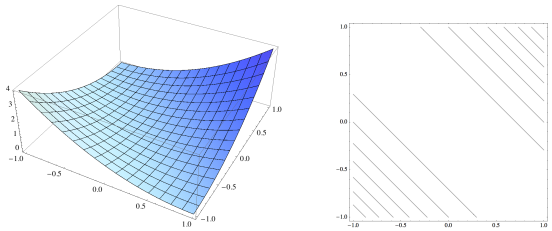
**Fact 2.8.2.** *Let us assume that the matrix* $Q$ *is non singular, or equivalently all its eigenvalues are non* $0$*. Then the function can be written in this form* $f(\mathbf{x}) = \frac{1}{2}\mathbf{x'}^T Q\mathbf{x'}$*, where* $\mathbf{x'} = \mathbf{x} - \bar{\mathbf{x}}$*, where* $\bar{\mathbf{x}}$ *denotes the optimum, that is where the gradient is* $0$*.*

*Proof.*

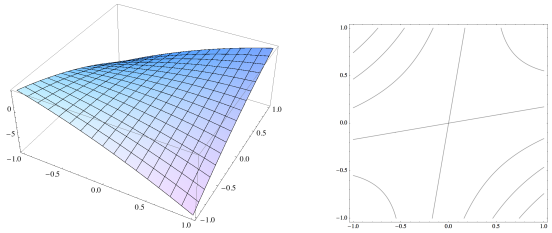
$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^T Q(\mathbf{x} - \bar{\mathbf{x}}) \\ &= \frac{1}{2}(\mathbf{x} + Q^{-1}\mathbf{q})^T Q(\mathbf{x} + Q^{-1}\mathbf{q}) \\ &\overset{(1)}{=} \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \frac{1}{2}\mathbf{q}^T \cancel{Q^{-T}}\cancel{Q}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \cancel{Q}\cancel{Q^{-T}}\mathbf{q} + \frac{1}{2}\mathbf{q}^T \cancel{Q^{-T}}\cancel{Q}Q^{-1}\mathbf{q} \\ &= \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}\mathbf{x} + \frac{1}{2}Q^{-1}\mathbf{q} \end{aligned}$$



(a) $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q_1 \mathbf{x} + q\mathbf{x}$



(b) $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q_2 \mathbf{x} + q\mathbf{x}$



(c) $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q_3 \mathbf{x} + q\mathbf{x}$

FIGURE 2.10: Shapes and level sets of quadratics functions with $Q_1, Q_2$ and $Q_3$ respectively.

where $\overset{(1)}{=}$ follows from the fact that $Q$ is symmetric. Notice that the constant factor $\frac{1}{2}Q^{-1}\mathbf{q}$ is not important for the purposes of minimizing. □

Proposition 2.8.2 allows us to look at the quadratic function without explicit linear terms. This is a translation that brings the origin in $\bar{\mathbf{x}}$ and looses the focus on the linear term.
Moreover, the size of the axes of the level curves is proportional to $\sqrt{1/\lambda_i}$, where $\lambda_i$ are the eigenvalues. If the eigenvalues are close to 0, then the axes get very long, while when they are both positive and negative, we no longer have axes.

This is formalized in

**Fact 2.8.3.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ such that $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}\mathbf{x}$. If the function is a bowl the unique minimum lies in the center. Formally, if $Q$ is positive definite $\bar{\mathbf{x}} = -Q^{-1}\mathbf{q}$ is the minimum of $f$, while if $Q$ is* indefinite *(i.e. $\exists \lambda_i < 0$) $f$ is*

*unbounded below.*

*Proof.* For the first part of the theorem, using the variable change of Proposition 2.8.2 $f(\mathbf{x}) = \mathbf{x}^T Q \mathbf{x}$ we get that ${\mathbf{x}'}^T Q \mathbf{x}' \geq 0$, because $Q$ is positive definite and the equality is matched only in $\mathbf{x} = \mathbf{0}$, i.e. $\mathbf{x} = \bar{\mathbf{x}}$. □

Notice that all we have said works symmetrically for quadratic functions such that ll the eigenvalues are negative. The only difference is that we look for the maximum in that case.

**Fact 2.8.4.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a quadratic function, let $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^n$ be orthonormal eigenvectors of $f$ relative to eigenvalues $\lambda_1, \lambda_2 \in \mathbb{R}$ respectively and let $\alpha \in \mathbb{R}^+$ be the step size. The following holds*

$$f(\alpha \mathbf{v}_i) = \lambda_i \alpha^2$$

*Proof.*

$$\begin{aligned} f(\alpha \mathbf{v}_i) &\overset{(1)}{=} (\alpha \mathbf{v}_i)^T Q (\alpha \mathbf{v}_i) \\ &= \alpha^2 {\mathbf{v}_i}^T \underbrace{Q \mathbf{v}_i}_{\lambda_i \mathbf{v}_i} \\ &= \alpha^2 \lambda_i {\mathbf{v}_i}^T \mathbf{v}_i \\ &= \alpha^2 \lambda_i \cdot \|\mathbf{v}_i\|^2 \\ &\overset{(2)}{=} \lambda_i \alpha^2 \end{aligned}$$

where $\overset{(1)}{=}$ follows from Proposition 2.8.2 and $\overset{(2)}{=}$ follows from the fact that the eigenvectors are orthonormal. □

**Do you recall?**

In Section 2.5 we introduced sequences, because the iterative procedure of starting from an initial point $\mathbf{x}_0$ and move towards the optimum gives birth to a hopefully minimizing sequence $\{\mathbf{x}_i\}$.

In the field of optimization it is crucial to decide at each iteration the moving direction as well as the step size along that particular direction.

We have already seen how a good guess for the direction $\mathbf{d}$ is the opposite of the gradient, where the function decreases very fast.

Let us now consider the step size, say $\alpha \in \mathbb{R}^+$. Intuitively, whenever we are moving along a direction in which the function value decreases very fast we are allowed to perform a reasonably big step, conversely $\alpha$ should be very small. In the simple case of quadratic functions, resorting Proposition 2.8.5, let us suppose we are sitting in a point such that the function value is 1, the step size is $\alpha = \sqrt{\frac{1}{\lambda_i}}$.

**Fact 2.8.5.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a quadratic function, such that $f(\mathbf{x}) = \mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T\mathbf{x}$, where $Q \in M(n, \mathbb{R})$ and it is positive semidefinite. Then $f$ is made of a purely quadratic part and a linear part, formally:*

$$f(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^T Q(\mathbf{x} - \bar{\mathbf{x}}) + \mathbf{q}_0\mathbf{x}$$

*where* $\ker(Q) =< \mathbf{q}_0 >$.

*Proof.* Thanks to the characterization theorem of $\mathbb{R}^n$ we know that the whole space $\mathbb{R}^n$ is generated by a linear combination between a basis of the image of matrix $Q$ (i.e. the column space of matrix $Q$ or, equivalently, the row space of $Q$) and a basis of the kernel of matrix $Q$. Formally, $\mathbb{R}^n = \text{row}(Q) + \ker(Q)$, where $\text{row}(Q) := \{\mathbf{x} = \mathbf{y}^T Q \text{ for some } \mathbf{y} \in \mathbb{R}^n\}$; moreover, the image of $Q$ and the kernel are orthogonal. Therefore, any vector $\mathbf{v} \in \mathbb{R}^n$ can be written as a linear combination of $\{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{n-1}, \mathbf{q}_0\}$, where $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{n-1}$ are a basis of the space spanned by the rows of $Q$ and $\mathbf{q}_0$ is the basis of the kernel of $Q$.

Without loss of generality, in order to ease notation, let us restrict to $\mathbb{R}^2$ and let us consider the row space generated by a vector called $\mathbf{q}_+$ and let $\mathbf{q} = \mathbf{q}_+ + \mathbf{q}_0$.

Let us perform some algebra and we get the thesis:

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T\mathbf{x} \\ &= \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + (\mathbf{q}_+ + \mathbf{q}_0)^T\mathbf{x} \\ &= \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + {\mathbf{q}_+}^T\mathbf{x} + {\mathbf{q}_0}^T\mathbf{x} \\ &\overset{(1)}{=} \frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^T Q(\mathbf{x} - \bar{\mathbf{x}}) + {\mathbf{q}_0}^T\mathbf{x} \end{aligned}$$

where $\overset{(1)}{=}$ follows from Proposition 2.8.2 □

In Proposition 2.8.5 we proved that a quadratic function $f$, which quadratic matrix $Q$ is singular (i.e., positive or negative semidefinite), has a quadratic shape along some directions and a linear shape along some (orthogonal) others. Let us consider the linear part: linear functions can be written as $\mathbf{a}^T\mathbf{x}$ and their gradient is $\nabla f(\mathbf{x}) = \mathbf{a}$. A linear function may increase, decrease or keep constant and in the first two cases it will diverge at the limit. If we sit in the point $\mathbf{x}_i$ and suppose that we move along a certain direction $\mathbf{d}$, our movement can be split into two different vectors, one along a level set (say $\mathbf{d}_{/\!/}$) and the other orthogonal to the level sets (say $\mathbf{p}_\perp$). It holds that if $\mathbf{d}_\perp \neq \mathbf{0}$ the linear function will go to $\infty$ either positive or negative. The situation explained above is represented in Figure 2.11.

It is known that in a quadratic function of the shape of $f$, the quadratic part dominates the linear one, but whenever the quadratic part ($\mathbf{x}^T Q\mathbf{x}$) is flat - and this happens for any $\mathbf{x} \in \ker(Q)$ - the linear part is responsible for the function's

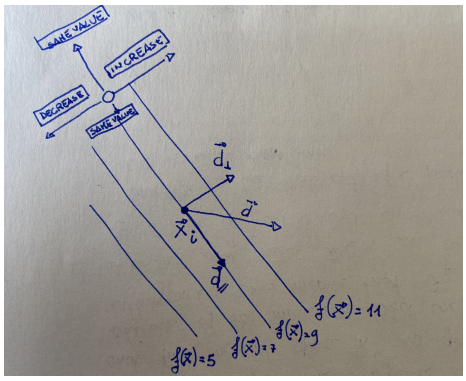



FIGURE 2.11: Level sets of a linear function.

behaviour. Formally,

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{x}^T \underbrace{Q\mathbf{x}}_{=0} + \mathbf{q}^T\mathbf{x} \\ &= 0 \cdot \mathbf{x}^T + \mathbf{q}^T\mathbf{x} \\ &= \mathbf{q}^T\mathbf{x} \end{aligned}$$

It is crucial to assess for which values of $\mathbf{q}$ the linear function does not diverge and this is formalized in the following

**Fact 2.8.6.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a quadratic function, such that $f(\mathbf{x}) = \mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T\mathbf{x}$, where $Q \in M(n, \mathbb{R})$ and it is positive semidefinite. Then $f$ has a minimum if and only if the vector $\mathbf{q}$ of the linear term of $f$ lies entirely in the row space of $Q$. Formally, $f$ has minimum $\iff$ $\mathbf{q}_0 = 0$, where $\ker(Q) =< \mathbf{q}_0 >$.*

*Proof.* The minimum of function $f$ corresponds to a point where the gradient vanishes, formally $\mathbf{x}^*$ such that $Q\mathbf{x}^* + \mathbf{q} = \mathbf{0}$ or, equivalently, $Q\mathbf{x}^* = -\mathbf{q}$. Here we have the thesis, thanks to the fact that a vector $(-1) \cdot \mathbf{q}$ can be written as $Q\mathbf{x}^*$ if and only if it belongs to the space spanned by the rows of matrix $Q$. □

# Chapter 3

# Unconstrained Optimality

## 3.1 Unconstrained Optimization

Until now we stated that the best conditions for finding the optimum are encountered when the domain is a compact set and we have many derivatives.

Now we need to consider when we can stop our algorithm.

**Definition 3.1.1** (Unconstrained optimization problem)**.** *We term* ***unconstrained optimization problem*** *the following*

$$(P)\ f_* = \min\{f(\mathbf{x})\ :\ \mathbf{x} \in \mathbb{R}^n\}$$

In unconstrained optimization, we deal with an unbounded set ($\mathbb{R}^n$), hence Weierstrass theorem does not apply. Because of this reason, we have no guarantee that a minimum $\mathbf{x}_*$ exists; moreover, provided its existence, finding it is an NP-hard problem. In order to make things easier, in practice we use a weaker condition: **local minimality**.

**Definition 3.1.2** (Local minimum)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. $\mathbf{x}_*$ is a* ***local minimum*** *if it is a global minimum in a little ball around $\mathbf{x}_*$. Formally,*

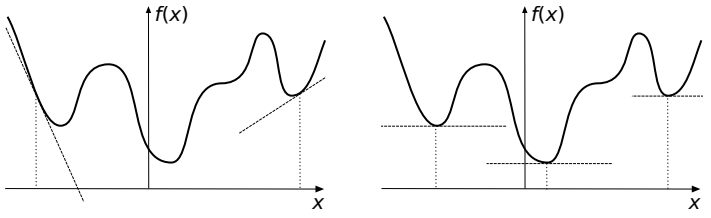
$$\min\{f(\mathbf{x})\ :\ \mathbf{x} \in \mathcal{B}(\mathbf{x}_*, \varepsilon)\}$$

*for some $\varepsilon > 0$.*

*Moreover, $\mathbf{x}_*$ is a* ***strict local minimum*** *if $f(\mathbf{x}_*) < f(\mathbf{x}')\ \forall \mathbf{x}' \in \mathcal{B}(\mathbf{x}_*,\ \varepsilon)$.*

A point where the gradient is smaller or bigger than 0 cannot be a local minimum, as shown in Figure 3.1.

Moreover, looking at the gradient we can easily tell if a point is a local minimum, but looking at the surroundings of a point we cannot tell if such point is a local or global minimum.




(a) Points where the derivative is not 0.

(b) Points where the derivative is 0.

FIGURE 3.1: Minima and non minima in one dimension.

## 3.2 First Order Model

**Do you recall?**

The first order model of $f : \mathbb{R}^n \to \mathbb{R} \in \mathcal{C}^1$ in some $\mathcal{B}(\mathbf{x}, \delta)$ is $L_{\mathbf{x}}(\mathbf{x}') = f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{x}' - \mathbf{x})$, such that $\forall \mathbf{x}' \in \mathbb{R}^n$ close to $\mathbf{x}$ $f(\mathbf{x}') = L_{\mathbf{x}}(\mathbf{x}') + R(\mathbf{x}' - \mathbf{x})$, where $R(\cdot)$ accounts for the distance between the model and the function, it is called **residual** and it has the property of more than linear convergence: $\lim_{\|\mathbf{h}\| \to 0} \frac{R(\mathbf{h})}{\|\mathbf{h}\|} = 0$.

In optimization, we are interested in moving towards a (local) minimum $\mathbf{x}_*$ as fast as possible.

**Fact 3.2.1.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be the objective function of the optimization problem (P) and let $\mathbf{x} \in \mathbb{R}^n$ be the current point at a generic iteration such that $f \in \mathcal{C}^1$ in some $\mathcal{B}(\mathbf{x}, \delta)$. In order to get closer to the optimum, we need to take a step along the* anti-gradient *direction. Formally, $\mathbf{x}(\alpha) = \mathbf{x} - \alpha \nabla f(\mathbf{x})$, where $\alpha \in \mathbb{R}$ is called* step size*.*

**Fact 3.2.2.** *Let $f$ be differentiable, if $\mathbf{x}$ is a local minimum, then $\nabla f(\mathbf{x}) = 0$.*

*Proof by contraddiction.* Let us assume that $\mathbf{x}$ is a local minimum but $\nabla f(\mathbf{x}) \neq 0$. We are interested in finding a scalar $\alpha$ such that $\mathbf{x}' = \mathbf{x}(\alpha) = \mathbf{x} - \alpha \nabla f(\mathbf{x})$ is such that $f(\mathbf{x}') < f(\mathbf{x})$.

In our case, $\mathbf{x}' = \mathbf{x} - \alpha \nabla f(\mathbf{x})$, so let us plug it into the *remainder version* of the Tailor's first-order model:

$$\begin{aligned} f(\mathbf{x}') &= f(\mathbf{x}) + < \nabla f(\mathbf{x}), \mathbf{x}' - \mathbf{x} > + R(\mathbf{x}' - \mathbf{x}) \\ &= f(\mathbf{x}) + < \nabla f(\mathbf{x}), \cancel{\mathbf{x}} - \alpha \nabla f(\mathbf{x}) - \cancel{\mathbf{x}} > + R(\cancel{\mathbf{x}} - \alpha \nabla f(\mathbf{x}) - \cancel{\mathbf{x}}) \\ &= f(\mathbf{x}) + < \nabla f(\mathbf{x}), -\alpha \nabla f(\mathbf{x}) > + R(-\alpha \nabla f(\mathbf{x})) \\ &= f(\mathbf{x}) \underbrace{-\alpha \|\nabla f(\mathbf{x})\|^2}_{<0} + R(-\alpha \nabla f(\mathbf{x})) \end{aligned}$$

At this point we need to find a small $\alpha$ that allows $R(-\alpha\nabla f(\mathbf{x})) < \alpha\|\nabla f(\mathbf{x})\|^2$. From the remainder version of Taylor's formula we know that $R(-\alpha\nabla f(\mathbf{x}))$ goes to 0 quadratically with respect to $\alpha$, while $\alpha\|\nabla f(\mathbf{x})\|^2$ goes to 0 linearly in $\alpha$, hence at some point we will get the inequality needed.

Once we fixed the moving direction, we can choose the step size $\alpha$, so it can be proved that $\lim_{\alpha\to 0} \frac{R(-\alpha\nabla f(\mathbf{x}))}{\|\alpha\nabla f(\mathbf{x})\|} = 0$, that is equivalent by definition to $\forall\varepsilon > 0\, \exists\bar{\alpha} > 0$ s.t. $\frac{R(-\alpha\nabla f(\mathbf{x}))}{\alpha\|\nabla f(\mathbf{x})\|} \leq \varepsilon\ \forall\ 0 \leq \alpha < \bar{\alpha}$.

If we take $\varepsilon < \|\nabla f(\mathbf{x})\|$, we get $R(-\alpha\nabla f(\mathbf{x})) < \alpha\,\|\nabla f(\mathbf{x})\|^2$, then

$$f(\mathbf{x}(\alpha)) = f(\mathbf{x}) - \alpha\,\|\nabla f(\mathbf{x})\|^2 + R(-\alpha\nabla f(\mathbf{x})) < f(\mathbf{x})$$

$\forall\alpha < \bar{\alpha}$ $\mathbf{x}$ cannot be a local minimum. □

Proposition 3.2.1 states that the first order model allows to find the decreasing direction, but if the gradient is 0 we do not know if we are in presence of a minimum, maximum or a saddle point (called stationary points). To discriminate among those, we exploit the information provided by the second derivative.

## 3.3 Second Order Model

**Do you recall?**

The second order model of $f : \mathbb{R}^n \to \mathbb{R} \in \mathcal{C}^2$ in some $\mathcal{B}(\mathbf{x}, \delta)$ is $Q_{\mathbf{x}}(\mathbf{x}') = L_{\mathbf{x}}(\mathbf{x}') + \frac{1}{2}(\mathbf{x}' - \mathbf{x})^T\nabla^2 f(\mathbf{x})(\mathbf{x}' - \mathbf{x}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{x}' - \mathbf{x}) + \frac{1}{2}(\mathbf{x}' - \mathbf{x})^T\nabla^2 f(\mathbf{x})(\mathbf{x}' - \mathbf{x})$, such that $\forall\mathbf{x}' \in \mathbb{R}^n$ close to $\mathbf{x}$ $f(\mathbf{x}') = Q_{\mathbf{x}}(\mathbf{x}') + R(\mathbf{x}' - \mathbf{x})$, where $R(\cdot)$ accounts for the distance between the model and the function, it is called **residual** and it has the property of more than quadratic convergence: $\lim_{\|\mathbf{h}\|\to 0} \frac{R(\mathbf{h})}{\|\mathbf{h}\|} = 0$.

**Property 3.3.1.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a quadratic function, let $\mathbf{x} \in \mathbb{R}^n$ be any stationary point and let the Hessian be an* invertible *matrix, then:*

- *If the Hessian is positive definite then $f$ attains a* local minimum *at $\mathbf{x}$.*
- *If the Hessian is negative definite then $f$ attains a* local maximum *at $\mathbf{x}$.*
- *If the Hessian has both positive and negative eigenvalues then $\mathbf{x}$ is a* saddle point *for $f$.*

Thanks to second order Taylor's model, we can approximate a function $f : \mathbb{R}^n \to \mathbb{R} \in \mathcal{C}^2$ using a quadratic model and so use Property 3.3.1 for checking when a stationary point is a minimum. In particular, the gradient of $Q_{\mathbf{x}}$ is computed as $\nabla Q_{\mathbf{x}} = \nabla f(\mathbf{x}) + \nabla^2 f(\mathbf{x})^T(\mathbf{x}' - \mathbf{x})$, then if $\mathbf{x} = \mathbf{x}'$ is a stationary point $\nabla Q_{\mathbf{x}}(\mathbf{x}) = \nabla f(\mathbf{x}) = \nabla L_{\mathbf{x}}(\mathbf{x}) = 0$.

Moreover, the Hessian of $Q_{\mathbf{x}}$ is computed as $Q_{\mathbf{x}} = \nabla^2 f(\mathbf{x})$, hence it has the same value of the Hessian of $f$.

**Fact 3.3.2.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ such that $f \in C^2$. If $\mathbf{x}$ is a local minimum then the Hessian is positive semidefinite. Formally, $\nabla^2 f(\mathbf{x}) \succeq 0$.*

*Proof by contraddiction.* Our contradictory hypothesis is that we are in a local minimum, but the Hessian is not positive semidefinite, hence there is a direction $\mathbf{d} \in \mathbb{R}^n$ of negative curvature. Formally, $\exists \mathbf{d}$ s.t. $\mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d} < 0$ or equivalently, the Hessian has a negative eigenvalue $\lambda_i < 0$. Let us now inspect how the function behaves moving along that direction and we will find out that the function decreases, meaning that $\mathbf{x}$ was not a local minimum.

Let us consider that $\mathbf{d}$ has norm 1 without loss of generality and let us be given a function of the step size step $\mathbf{x}(\alpha) = \mathbf{x} + \alpha \mathbf{d}$ and then take the second-order Taylor formula as follows (where there is no linear term involved because $\nabla f(\mathbf{x}) = 0$):

$$
\begin{aligned}
f(\mathbf{x}(\alpha)) &= f(\mathbf{x}) + \frac{1}{2}(\mathbf{x}(\alpha) - \mathbf{x})^T \nabla^2 f(\mathbf{x})(\mathbf{x}(\alpha) - \mathbf{x}) + R(\mathbf{x}(\alpha) - \mathbf{x}) \\
&= f(\mathbf{x}) + \frac{1}{2}(\cancel{\mathbf{x}} + \alpha \mathbf{d} - \cancel{\mathbf{x}})^T \nabla^2 f(\mathbf{x})(\cancel{\mathbf{x}} + \alpha \mathbf{d} - \cancel{\mathbf{x}}) + R(\cancel{\mathbf{x}} + \alpha \mathbf{d} - \cancel{\mathbf{x}}) \\
&= f(\mathbf{x}) + \frac{1}{2}(\alpha \mathbf{d})^T \nabla^2 f(\mathbf{x})(\alpha \mathbf{d}) + R(\alpha \mathbf{d}) \\
&= f(\mathbf{x}) + \frac{1}{2}\alpha^2 \mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d} + R(\alpha \mathbf{d})
\end{aligned}
$$

with $\lim_{\|h\| \to 0} \frac{R(h)}{\|h\|^2} = \lim_{\alpha \to 0} \frac{R(\alpha \mathbf{d})}{\alpha^2} = 0$ or equivalently $\forall \varepsilon > 0 \, \exists \bar{\alpha} > 0$ s.t. $R(\alpha \mathbf{d}) \leq \varepsilon \alpha^2 \; \forall 0 \leq \alpha < \bar{\alpha}$. Notice that the term $\frac{1}{2}\alpha^2 \mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d} < 0$ and $R(\alpha \mathbf{d}) < 0$ At this point, since this condition holds for each $\varepsilon$ we are allowed to take the most convenient: $\varepsilon < -\frac{1}{2}\mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d}$, so that we obtain this condition on the residual $R(\alpha \mathbf{d}) < -\frac{1}{2}\alpha^2 \mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d}$, hence

$$
f(\mathbf{x}(\alpha)) = f(\mathbf{x}) + \frac{1}{2}\alpha^2 \mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d} + R(\alpha \mathbf{d}) < f(\mathbf{x}) \forall \, 0 \leq \alpha < \bar{\alpha}
$$

The contradiction lies in the fact that $\mathbf{x}(\alpha)$ leads to a value of $f$ smaller than the one in $\mathbf{x}$. □

Proposition 3.3.2 states that a positive semidefinite Hessian is a necessary condition for a local minimum. Follows a proposition that provides a sufficient condition.

**Fact 3.3.3.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ s.t. $f \in C^2$ and let the Hessian be symmetric (hence real eigenvalues). If $\nabla f(\mathbf{x}) = 0$ and the Hessian is strictly positive definite ($\nabla^2 f(\mathbf{x}) \succ 0$) then $\mathbf{x}$ is a strict local minimum.*

**Do you recall?**

In the notes on Numerical Methods we stated the **Variational characterization**:
Let $Q \in S(n, \mathbb{R})$ and let $\mathbf{x} \in \mathbb{R}^n$. Then

$$\lambda_{\min}\|\mathbf{x}\|^2 \leq \mathbf{x}^T Q\mathbf{x} \leq \lambda_{\max}\|\mathbf{x}\|^2$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are respectively the eigenvalue of maximum value and the eigenvalue of minimum value.

*Proof.* Let $\mathbf{x} \in \mathbb{R}^n$ be a point where the gradient of the function is 0, then we get the following second order Taylor's approximation for $\mathbf{x}_{\text{new}} = \mathbf{x} + \mathbf{d}$ (unitary step size for simplicity):

$$f(\mathbf{x}+\mathbf{d}) = f(\mathbf{x}) + \frac{1}{2}\mathbf{d}^T\nabla^2 f(\mathbf{x})\mathbf{d} + R(\mathbf{d}) \text{ with } \lim_{h\to 0}\frac{R(\mathbf{d})}{\|\mathbf{d}\|^2} = 0$$

Hence, by definition of limit $\forall \varepsilon > 0\ \exists\ \delta > 0$ s.t. $R(\mathbf{d}) \geq -\varepsilon\|\mathbf{d}\|^2\ \ \forall \mathbf{d} \in \mathbb{R}^n$ s.t. $\|\mathbf{d}\| < \delta$.

Since the Hessian is strictly positive definite, the smallest eigenvalue $\lambda_{\min}$ is strictly greater than 0, hence the variational characterization of eigenvalues $\mathbf{d}^T\nabla^2 f(x)\mathbf{d} \geq \lambda_{\min}\|\mathbf{d}\|^2$.

We are now ready to pick the $\varepsilon$ we prefer (i.e. $\varepsilon < \frac{\lambda_{\min}}{2}$) to get $\forall \mathbf{d}$ s.t. $\|\mathbf{d}\| < \delta$ that the function value in any other point close to $\mathbf{x}$ is greater:

- $\mathbf{d}^T\nabla^2 f(x)\mathbf{d} \geq \lambda_{\min}\|\mathbf{d}\|^2$
- $R(\mathbf{d}) \geq -\varepsilon\|\mathbf{d}\|^2$

$$f(\mathbf{x}+\mathbf{d}) = f(\mathbf{x}) + \frac{1}{2}\mathbf{d}^T\nabla^2 f(\mathbf{x})\mathbf{d} + R(\mathbf{d}) \geq f(\mathbf{x}) + (\frac{\lambda_{\min}}{2} - \varepsilon)\|\mathbf{d}\|^2 > f(\mathbf{x})$$

Where the term $\lambda_{\min} - \varepsilon$ is strictly positive. □

**Definition 3.3.1** (Strong local optimality)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ and let $\mathbf{x} \in \mathbb{R}^n$ a solution of the minimum problem (P). We say that $\mathbf{x}$ is a **strong** local optimum if $f$ grows at least quadratically. Formally,*

$$\exists \delta > 0 \text{ and } \gamma > 0 \text{ s.t. } f(\mathbf{x}') \geq f(\mathbf{x}) + \gamma\|\mathbf{x}' - \mathbf{x}\|^2\ \forall \mathbf{x}' \in \mathcal{B}(\mathbf{x}, \delta)$$

There are some points, called saddle points (Figure 3.7), where both the first and the second order derivative do not give any information about the shape of the function and the third order derivatives are required to infer such information.
In the rest of this lecture, we will provide conditions that ensure that once a local minimum is found, it is also a global minimum, namely that the function does not look like the one in Figure 3.1.

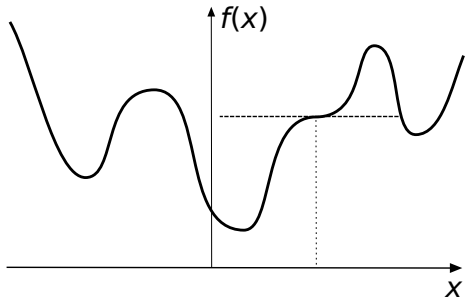



FIGURE 3.2: Saddle point.

So far, we said that the local minima are those points where the gradient is 0 and the Hessian is positive semidefinite. An easy way to ensure that the Hessian is positive semidefinite in a ball around $\mathbf{x}$ is to have that the Hessian is positive semidefinite everywhere ($\forall \mathbf{x} \in \mathbb{R}^n$), that actually means that there are no local maxima and this is guaranteed to be true whenever $f$ is a *convex* function, as the one displayed in Figure 3.3.

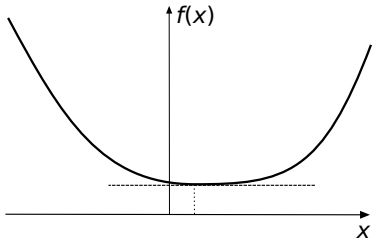



FIGURE 3.3: Convex function.

As the reader may know, in machine learning (which is one of the subjects to which the tools of optimization are applied) there is the need of choosing the class of functions the model belongs to. At this step, it is preferable to select a function which is convex, for the reasons explained above.

## 3.4 Convexity

Let us introduce convexity as far as both sets and functions are concerned.

### 3.4.1 Convex sets

**Definition 3.4.1** (Convex hull)**.** *Let* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ *we term* ***convex hull*** *and denote* $conv(\mathbf{x}, \mathbf{y}) = \{\mathbf{z} = \alpha\mathbf{x} + (1-\alpha)\mathbf{y} \ : \ \alpha \in [0,1]\}$ *the segment joining* $\mathbf{x}$ *and* $\mathbf{y}$*.*

**Definition 3.4.2** (Convex set)**.** *Intuitively, we say that* $C$ *is a* ***convex set*** *if for each couple in the set, the line joining such points belongs to the set. Formally,* $C \subset \mathbb{R}^n$ *is a* ***convex set*** *if* $\forall \ \mathbf{x}, \mathbf{y} \in C$ $conv(\mathbf{x}, \mathbf{y}) \subseteq C$*.*

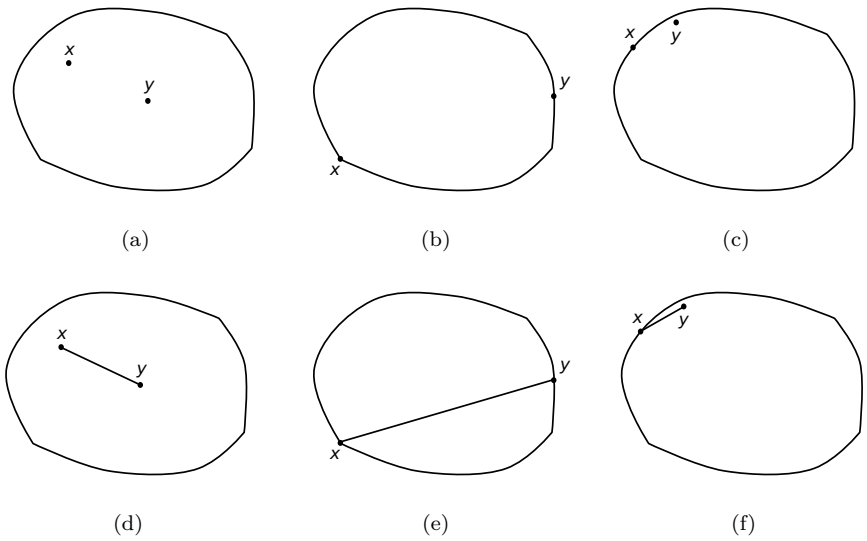



FIGURE 3.4: Some examples of convex sets.

Notice that "disconnected sets" cannot be convex sets, as an example see Figure 3.5.

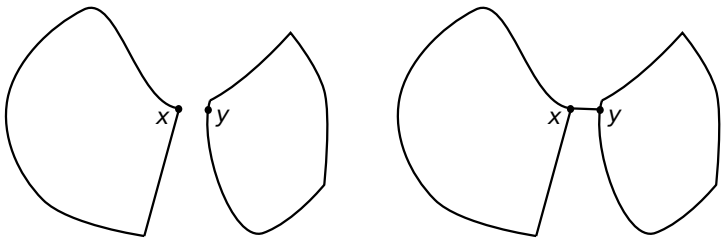



FIGURE 3.5: Non connected, hence non convex sets.

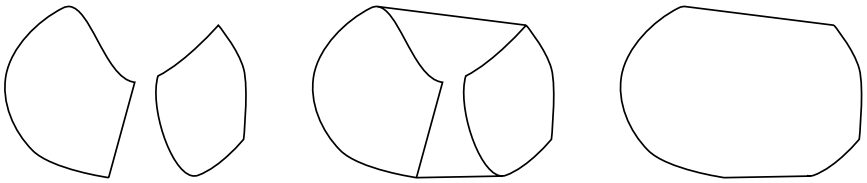
In general, it is more likely to deal with a set that is non convex instead of a convex one. In this case we can make use of a trick that somehow "completes"

the non convex set into a convex one, as shown in Figure 3.6 and formalized in Definition 3.4.3.

**Definition 3.4.3** (Convex hull of a set)**.** *Given a set $S \subseteq \mathbb{R}^n$, we can "complete" it to a convex set conv$(S)$, called **convex hull of** $S$, using two different approaches: one is to add all the segments joining any possible couple of points in $S$, the other one is to choose the smallest convex set that contains $S$. Formally,*

$$\begin{aligned} conv(S) &= \bigcup \{\, conv(\mathbf{x}, \mathbf{y}) \,:\, \mathbf{x}, \mathbf{y} \in S \,\} \\ &= \bigcap \{\, C \,:\, C \text{ is convex } \wedge C \supseteq S \,\} \end{aligned}$$

*Equivalently, the convex hull of $S$ can be defined iteratively as the convex hull of all the couples of points in $S$.*
*Equivalently, we can define the convex hull of a set $S$ as the smallest convex set containing $S$.*



(a) (b) (c)

FIGURE 3.6: How to merge two disjoint sets into a convex one.

**Note**

A more general definition of a **convex hull** is the following: $\text{conv}(\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}) = \left\{ \mathbf{x} = \sum_{i=1}^{k} \alpha_i \mathbf{x}_i \;:\; \sum_{i=1}^{k} \alpha_i = 1,\ \alpha_i \geq 0 \ \forall i \right\}$

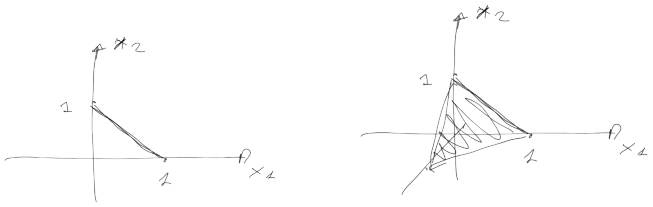
**Definition 3.4.4** (Unitary simplex)**.** *We term **unitary simplex** the set of $k$ non-negative scalars summing to 1, formally*

$$\Theta^k = \{ \sum_{i=1}^{k} \alpha_i \mathbf{x}_i \in \mathbb{R}^k \;:\; \sum_{i=1}^{k} \alpha_i = 1, \alpha_i \geq 0 \ \forall i \}$$

Our goal is to find the simplest possible convex set that approximates our set.

**Fact 3.4.1.** *A convex set is equal to its convex hull. Formally,*

$$C \text{ is convex} \iff C = conv(C)$$



(a) In $\mathbb{R}^2$ (b) In $\mathbb{R}^3$

FIGURE 3.7: Example of unitary simplexes.

**Fact 3.4.2.** *A set $C \subseteq \mathbb{R}^n$ is convex iff for any possible choice of its elements, their convex hull is contained in $C$. Formally,*

$$C \text{ is convex} \iff C \supseteq conv(\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k\}) \ \forall \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k \in C$$

We are interested in sufficient conditions for convexity. In the rest of this lecture we are going to define some basic sets that are convex and then we will see some operations that preserve convexity.

In this way we can try to reconstruct a set as originated from some convex set via some allowed manipulations.

**Definition 3.4.5** (Cone)**.** *We term **cone** the set $\mathcal{C} = \{\mathbf{x} \in \mathbb{R}^n \ : \ \alpha\mathbf{x} \in \mathcal{C} \forall \alpha \geq 0\}$.*

An attentive reader may notice that the definition of cone is a relaxation of the unitary simplex, where we do not require the unitary sum.

**Example 3.4.1.** *The following sets are convex:*

- *Convex polytope $conv(\{\mathbf{x}_1, \dots, \mathbf{x}_k\})$, unitary simplex $\Theta$*
- *Affine hyperplane: $\mathcal{H} := \{\mathbf{x} \in \mathbb{R}^n \ : \ \mathbf{a}^T\mathbf{x} = b\}$*
- *Affine close (open) subspace: $\mathcal{S} := \{\mathbf{x} \in \mathbb{R}^n \ : \ \mathbf{a}^T\mathbf{x} \leq (<) b\}$*
- *Close (open) ball of radius $r$ in $p$-norm, $p \geq 1$: $\mathcal{B}_p(\mathbf{x}, r) = \{\mathbf{y} \in \mathbb{R}^n \ : \ \|\mathbf{y} - \mathbf{x}\|_p \leq (<) r\}$*
- *Close (open) ellipsoid: $\mathcal{E}(Q, \mathbf{x}, r) := \{\mathbf{y} \in \mathbb{R}^n \ : \ (\mathbf{y} - \mathbf{x})^T Q(\mathbf{y} - \mathbf{x}) \leq (<) r\}$ with $Q \succeq 0$. Notice that ellipsoids are levelsets of quadratic functions.*
- *Cones*
- *Conical hull of a finite set of directions:*

$$cone(\{\mathbf{d_1}, \dots, \mathbf{d_k}\}) = \left\{\mathbf{d} = \sum_{i=1}^{k} \mu_i \mathbf{d_i} \ : \ \mu_i \geq 0 \ \forall i \right\}$$

- *Lorentz's cone (ice-cream cone):* $\mathbb{L} = \left\{ \mathbf{x} \in \mathbb{R}^n \,:\, x_n \geq \sqrt{\sum_{i=1}^{n-1} {x_i}^2} \right\}$
- *Cone of positive semidefinite matrices:* $\mathbb{S}_+ = \{ A \in \mathcal{M}(n, \mathbb{R}) \,:\, A \succeq 0 \}$

**Fact 3.4.3.** *The following operations preserve convexity.*

1. *Given a possibly infinite family of convex sets* ($\{C_i\}_{i \in I}$)*, the* intersection ($\bigcap_{i \in I} C_i$) *is convex;*
2. *If we have convex sets in different subspaces, their* Cartesian product *is a convex set* ($C_1, \ldots, C_k$ *convex* $\iff C_1 \times \cdots \times C_k$ *convex);*
3. *Given a convex set, its image under a* linear mapping *(i.e. scaling, translation, rotation) is a convex set. Formally, let $C$ be a convex set. Then $A(C) := \{\mathbf{x} = A\mathbf{y} + \mathbf{b} \,:\, \mathbf{y} \in C\}$ for a given $A \in \mathcal{M}(n, \mathbb{R})$ and $\mathbf{b} \in \mathbb{R}^n$ is convex;*
4. *If $C$ is a convex set its* inverse *image under a linear mapping is convex as well. Formally, $A^{-1}(C) := \{\mathbf{x} \,:\, A\mathbf{x} + \mathbf{b} \in C\}$ is convex;*
5. *A linear combination of convex sets is convex. Formally, let $C_1$ and $C_2$ be convex sets and let $\alpha_1, \alpha_2 \in \mathbb{R}$, then $\alpha_1 C_1 + \alpha_2 C_2 := \{\mathbf{x} = \alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2 : \mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2\}$ is convex;*
6. *Let $C \subseteq \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$ be a convex set. Slicing and projection lead to convex sets as well, as shown in Figure 3.8:*

   Slice: $C_s(\mathbf{y}) := \{\mathbf{x} \in \mathbb{R}^{n_1} \,:\, (\mathbf{x}, \mathbf{y}) \in C\}$ *is convex*

   Projection: $C_p := \{\mathbf{x} \in \mathbb{R}^{n_1} \,:\, \exists \mathbf{y} \in \mathbb{R}^{n_2} \text{ s.t. } (\mathbf{x}, \mathbf{y}) \in C\}$ *is convex*
7. *Given a convex set $C$, the interior of such set and its closure are convex.*


(a) Slice on $\bar{\mathbf{y}}$ (b) Project on the $x$ component

Figure 3.8: Pictorial examples of slicing and projecting.

**Fact 3.4.4.** *The union of convex sets is not convex.*

*Proof.* Let us provide a counter-example to this fact. Let us choose intervals in $\mathbb{R}$ for simplicity: $[0, 1]$ and $[2, 3]$ are both convex, because they satisfy the definition, but their union has a "gap", hence $t \cdot 1 + (1 - t) \cdot 2$ is not in $[0, 1] \cup [2, 3]$ for any $t \in (0, 1)$. □

**Definition 3.4.6** (Polyhedron)**.** *Let $A \in M(m,n,\mathbb{R})$ and let $\mathbf{b} \in \mathbb{R}^m$, we term* ***polyhedron*** *$P := \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} \leq \mathbf{b}\}$.*

**Fact 3.4.5.** *A polyhedron is convex.*

**Fact 3.4.6.** *Let $P \subseteq \mathbb{R}^n$ be a polyhedron. Its recession cone is a cone. Formally,*

$$R := \{\mathbf{d} : \mathbf{x} + \alpha\mathbf{d} \in P \ \forall \mathbf{x} \in P,\ \alpha \geq 0\}$$

*is a cone.*

*Proof.*

$$\begin{array}{c} \forall \mathbf{d} \in R\ \underbrace{\beta\mathbf{d} \in R}\ \forall \beta \geq 0 \\ \Updownarrow \\ \mathbf{x} + \alpha\beta\mathbf{d} \in P \\ \Updownarrow \\ A(\mathbf{x} + \alpha\beta\mathbf{d}) \leq \mathbf{b} \\ \Updownarrow \\ A\mathbf{x} + \alpha\beta A\mathbf{d} \leq \mathbf{b} \end{array}$$

If we pick $\alpha' = \alpha\beta$ we get the thesis. □

**Theorem 3.4.7.** *$\mathcal{P}$ is a polyhedron iff any of its points can be written as a combination of a point in a convex set and one in a cone. Formally, iff $\exists\{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \subseteq \mathbb{R}^n$ and $\{\mathbf{d}_1, \ldots, \mathbf{d}_h\} \subseteq \mathbb{R}^n$ s.t. $\mathcal{P} = conv(\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}) + cone(\{\mathbf{d}_1, \ldots, \mathbf{d}_h\})$.*

### 3.4.2 Convex functions

In the rest of this lecture we will introduce the class of convex functions, that have the characteristic of having local minima coinciding with global minima.

**Definition 3.4.7** (Graph)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$, we term* ***graph*** *of $f$ the set of ordered pairs $(\mathbf{x}, y)$ such that $f(\mathbf{x}) = y$.*

**Definition 3.4.8** (Epigraph)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$, we term* ***epigraph*** *or* ***supergraph*** *of $f$ the set of points lying on or above its graph. Formally,*

$$epi(f) = \{(\mathbf{x}, \mu)\ :\ \mathbf{x} \in \mathbb{R}^n,\ \mu \in \mathbb{R},\ \mu \geq f(\mathbf{x})\} \subseteq \mathbb{R}^{n+1}$$

**Definition 3.4.9** (Convex function)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function. We say that $f$ is* ***convex*** *if $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, the segment that joins $f(\mathbf{x})$ and $f(\mathbf{y})$ lies* above *the function.*
*In other words, $f$ is* ***convex*** *iff $epi(f)$ is convex.*
*Equivalently, we say that $f$ is* ***convex*** *if $\forall \mathbf{x}, \mathbf{y} \in dom(f)$ for any $\alpha \in [0,1]$, $\alpha f(\mathbf{x}) + (1-\alpha)f(\mathbf{y}) \geq f(\alpha\mathbf{x} + (1-\alpha)\mathbf{y})$, as depicted in Figure 3.9.*
*Equivalently, $\forall \mathbf{x}^1, \ldots, \mathbf{x}^k$, $\alpha \in \Theta^k$*

$$f\left(\sum_{i=1}^{k} \alpha_i \mathbf{x}^i\right) \leq \sum_{i=1}^{k} \alpha_i f(\mathbf{x}^i)$$

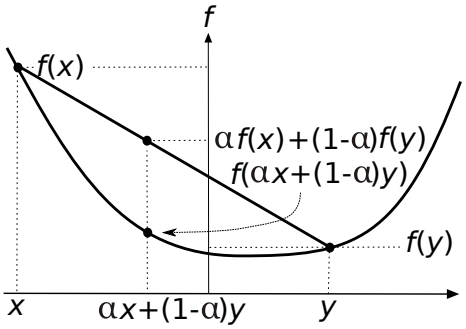



FIGURE 3.9: Given two points $x, y \in \mathbb{R}$ the function value of any of the points lying inside the interval $]x, y[$ is below the value of a linear function that passes by $(x, f(x)), (y, f(y))$, hence the function is strictly convex.

**Definition 3.4.10** (Strict convexity)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$. We term $f$* ***strictly convex*** *iff $\alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y}) > f(\alpha \mathbf{x} + (1-\alpha)\mathbf{y})$.*

Notice that a linear function is convex but not strictly convex.

**Definition 3.4.11** (Concave function)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function. We say that $f$ is* ***concave*** *if $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, the segment that joins $f(\mathbf{x})$ and $f(\mathbf{y})$ lies* below *the function. Formally, $\forall \mathbf{x}, \mathbf{y} \in dom(f)$ for any $\alpha \in [0, 1]$, $\alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y}) \leq f(\alpha \mathbf{x} + (1-\alpha)\mathbf{y})$.*

**Definition 3.4.12** (Sublevel graph)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a function. We term* ***sublevel graph*** *of $f$ on a scalar $v \in \mathbb{R}$ and denote $S(f, v)$ the projection on the $\mathbf{x}$ axis of the portions of the graph of $f$ which lie below the constant value $v$.*

**Fact 3.4.8.** *The following holds:*

- *Let $f$ convex. Then $S(f, v)$ is convex $\forall v \in \mathbb{R}$;*
- *$f$ is concave if $-f$ is convex.*

**Mantra**

*Convex analysis is a one-sided world*, that means that the arrows in propositions flow in one way only.

The second statement of Proposition 3.4.8 is useful to make a comparison between minimizing and maximizing. In particular, if our aim is to maximize the function, we can be sure to have found a global maximum if the function is concave.

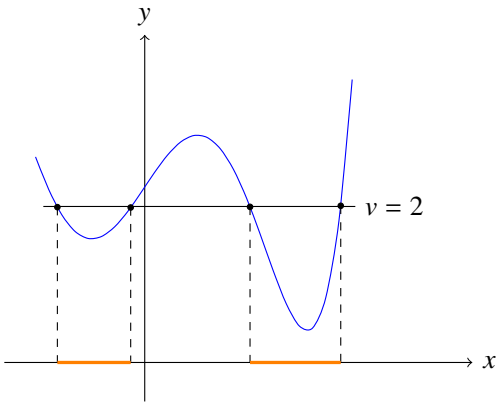



FIGURE 3.10: $f(x) = x^5 - 8x^3 + 10x + 15$ in blue and its sublevel graph in orange.

**Definition 3.4.13** (Strong convexity)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. We term $f$* ***strongly convex*** *if exists a quadratic function that lies below $f$, that means that the function grows quadratically. Formally, we say that $f$ is* ***strongly convex modulus*** *a scalar $\tau > 0$ iff $f(\mathbf{x}) - \frac{\tau}{2} \|\mathbf{x}\|^2$ is convex. Equivalently, $f$ is* ***strongly convex modulus*** *a scalar $\tau > 0$ iff*

$$\alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y}) \geq f(\alpha \mathbf{x} + (1-\alpha)\mathbf{y}) + \frac{\tau}{2} \alpha(1-\alpha) \|(\mathbf{y} - \mathbf{x})\|^2$$

In the next paragraphs we will provide techniques that allow to check if a function is convex in practice, hence where a local minimum is also a global minimum.

As we already stated for convex sets, we can prove that a function is convex if we are able to derive it from convex functions, through "convex friendly" operations.

**Note**

In order to have insights about convexity we can use the software called CVX, designed to model convex objects. A pretty easy way to check if an object is convex is to try to write it in CVX. If such an operation is possible, then the object is convex.

Let us enumerate some convex functions:

**Linear functions:** $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ (they are both convex and concave);

**Quadratic functions:** $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q \mathbf{x} + \mathbf{q}^T \mathbf{x}$ is convex iff $Q \succeq 0$;

**Exponential function:** $f(\mathbf{x}) = e^{a\mathbf{x}}$ for any $a \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^n$

**Anti-logaritmic function:** $f(x) = -\log(x)$ for $x > 0$

**Monomial:** $f(x) = x^a$ for $a \geq 1$ or $a \leq 0$ on $x \geq 0$;

**$p$-norm:** $f(\mathbf{x}) = \|\mathbf{x}\|_p$ for $p \geq 1$;

**Maximum:** $f(x) = \max\{x_1, \ldots, x_n\}$;

**Indicator function:** for any convex set $C$, its indicator function $\mathbb{1}_C(x)$ is convex:

$$\mathbb{1}_C(x) = \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{if } x \notin C \end{cases}$$

**Sum of $m$ largest eigenvalues:** Let $A \in S(n, \mathbb{R})$ such that its eigenvalues (sorted in increasing order) are $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_n$. $f_m(A) = \sum_{i=1}^{m} \lambda_i$ is convex.

**Fact 3.4.9.** *The following operations preserve convexity:*

**Linear non-negative combination:** *let $f, g : \mathbb{R}^n \to \mathbb{R}$ be convex functions and let $\alpha, \beta \in \mathbb{R}_+$, then $\alpha f + \beta g$ is convex;*

**Supremum:** *let $\{f_i\}_{i \in I}$ be a set of infinitely many convex functions, then $\sup_{i \in I} f_i(\mathbf{x})$ is convex, see Figure 3.11(a);*

**Pre-composition with linear function:** *let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function and let $A \in M(m, n, \mathbb{R}), \mathbf{b} \in \mathbb{R}^m$, then $f(A\mathbf{x} + \mathbf{b})$ is convex;*

**Post-composition with increasing convex function:** *let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function and let $g : \mathbb{R} \to \mathbb{R}$ be a convex increasing function. Then $g \circ f = g(f(\mathbf{x}))$ is convex;*

**Infimal convolution:** *let $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ be convex functions. Then $f(\mathbf{x}) = \inf\{f_1(\mathbf{x_1}) + f_2(\mathbf{x_2}) \ : \ \mathbf{x_1} + \mathbf{x_2} = \mathbf{x}\}$ is convex;*

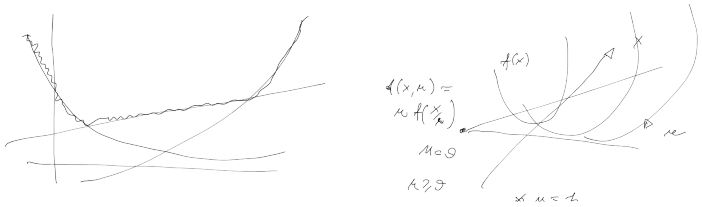
**Image under linear mapping:** *let $g : \mathbb{R}^n \to \mathbb{R}$ be a convex function. Then $f(\mathbf{x}) = \inf\{g(\mathbf{y}) \ : \ A\mathbf{y} = \mathbf{x}\}$ is convex;*

**Partial minimization:** *let $g \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} : \mathbb{R}^{n+m} \to \mathbb{R}$ be a convex function. Then $f(\mathbf{x}) = \inf \left\{ g \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \ : \ \mathbf{y} \in \mathbb{R}^m \right\}$ is convex;*

**Perspective or dilation:** *let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. Then*

$$\tilde{f}(\mathbf{x}, u) = \begin{cases} u \cdot f(\mathbf{x}/u) & \text{if } u > 0 \\ \infty & \text{otherwise} \end{cases}$$

*is convex, see Figure 3.11(b).*



(a) Supremum (b) Perspective

FIGURE 3.11: Examples of convexity-preserving operations.

**Fact 3.4.10.** *Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a convex function. If $\exists \bar{\mathbf{x}} \in dom(f)$ such that $f(\bar{\mathbf{x}}) = -\infty$, then $f \equiv -\infty$.*

**Note**

As we did in the case of functions which domain is not the whole $\mathbb{R}$, from now on we will solve the issue of functions with a non convex domain, saying that in those points where the function is not defined, we value is $+\infty$.

**Do you recall?**

Let us paste here the definition of **lower semi-continuity** presented in Definition 2.6.3.
Let $\{\mathbf{x_i}\} \subseteq \mathbb{R}^n$ be a sequence with accumulation point in $\mathbf{x}$ and let $f : \mathbb{R}^n \to \mathbb{R}$. $f$ is **lower semi-continuous** (l.s.c.) at $\mathbf{x}$ if $f(\mathbf{x}) \leq \liminf_{i \to \infty} f(\mathbf{x_i})$.
Equivalently, $\liminf_{\mathbf{y} \to \mathbf{x}} f(\mathbf{y}) \geq f(\mathbf{x})$, $\limsup_{\mathbf{y} \to \mathbf{x}} f(\mathbf{y}) \leq f(\mathbf{x})$.

**Example 3.4.2.** *Let us take a set $X \subseteq \mathbb{R}$ and its characteristic function $\mathbb{1}_X$. Such function is lower semi-continuous.*

**Fact 3.4.11.** *In minimization we have that any closed convex function $f$ is also at least lower semi-continuous and continuous in the interior of its domain (i.e. far from the points where the function value shuts up to $+\infty$).*

**Fact 3.4.12.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. Then $f$ is Lipschitz continuous on each bounded convex set $S \subseteq int(dom(f))$, but we have no information on the behaviour on the border of the domain.*

*Moreover, a function $f$, which is continuous but not Lipschitz continuous is not convex.*

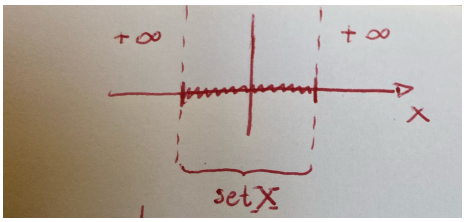
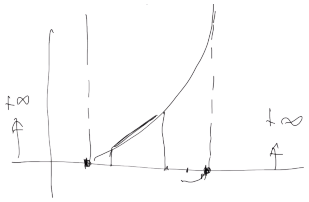
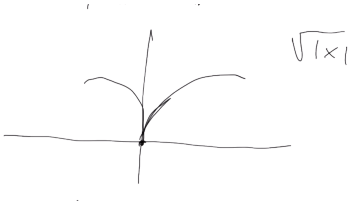



FIGURE 3.12: $\mathbb{1}_X$ on the codomain $\bar{\mathbb{R}}$.




(a) On a compact set in the interior of the domain (far from the boundaries) the function is Lipschitz continuous.



(b) If a function is not Lipschitz on a compact subset it is not convex. In this case $\lim_{x\to 0} f'(x) = \infty$.

FIGURE 3.13

A couple of examples of Proposition 3.4.12 can be found in Figure 3.13.

**Fact 3.4.13.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. Then it is Lipschitz continuous on any bounded set and continuous everywhere.*

It happens often that the set of points in which a function is non differentiable has measure 0.

## 3.5 Convexity and Higher Order Information

In this section we would like to provide equivalent definitions of convexity, provided that the function is continuously differentiable ($f \in \mathcal{C}^1$).

**Theorem 3.5.1** (Convexity characterization)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be in $\mathcal{C}^1$. It is convex on a convex set $C \subseteq \mathbb{R}^n$ iff the value of the function lies* above *the first order model (see Figure 3.14). Formally,*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) \ \forall \mathbf{x}, \mathbf{y} \in C$$

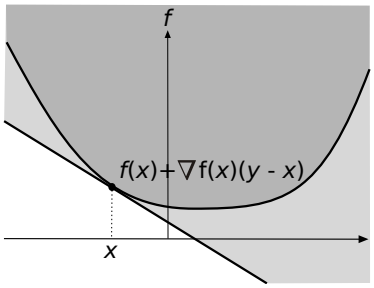



FIGURE 3.14: The epigraph is an half-space that contains that of $f$ $epi(L_x) \supseteq epi(f)$

**Do you recall?**

In the proof of the previous theorem we need the result presented in Section 2.7, in Proposition 2.7.11, where we derived the directional derivative from the gradient as $\frac{\partial f}{\partial \mathbf{d}}(\mathbf{x}) =< \nabla f(\mathbf{x}), \mathbf{d} >$.

*Proof.* $\Rightarrow$) Let us write the definition of convexity:

$$\begin{aligned}
\alpha f(\mathbf{y}) + (1-\alpha) f(\mathbf{x}) &\geq f(\alpha \mathbf{y} + (1-\alpha)\mathbf{x}) \\
\alpha f(\mathbf{y}) + f(\mathbf{x}) - \alpha f(\mathbf{x}) &\geq f(\alpha \mathbf{y} + \mathbf{x} - \alpha \mathbf{x}) \\
\alpha \Big( f(\mathbf{y}) - f(\mathbf{x}) \Big) + f(\mathbf{x}) &\geq f\Big( \alpha(\mathbf{y} - \mathbf{x}) + \mathbf{x} \Big) \\
\alpha \Big( f(\mathbf{y}) - f(\mathbf{x}) \Big) &\geq f\Big( \alpha(\mathbf{y} - \mathbf{x}) + \mathbf{x} \Big) - f(\mathbf{x}) \\
f(\mathbf{y}) - f(\mathbf{x}) &\geq \underbrace{\frac{f\Big( \alpha(\mathbf{y} - \mathbf{x}) + \mathbf{x} \Big) - f(\mathbf{x})}{\alpha}}_{(*)}
\end{aligned}$$

where $(*)$ is the incremental ratio of the directional derivative:

$$\lim_{\alpha \to 0} \frac{f\Big( \alpha(\mathbf{y} - \mathbf{x}) + \mathbf{x} \Big) - f(\mathbf{x})}{\alpha} = \frac{\partial f(\mathbf{x})}{\partial (\mathbf{y} - \mathbf{x})} \overset{(1)}{=} < \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} >$$

where $\overset{(1)}{=}$ holds because $f \in \mathcal{C}^1$.

The thesis follows from a couple of passages, for $\alpha \to 0$:

$$\begin{aligned}
f(\mathbf{y}) - f(\mathbf{x}) &\geq < \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} > \\
f(\mathbf{y}) - f(\mathbf{x}) &\geq \nabla f(\mathbf{x})^T (\mathbf{y} - \mathbf{x}) \\
f(\mathbf{y}) &\geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T (\mathbf{y} - \mathbf{x})
\end{aligned}$$

□

Notice that if $\mathbf{x} \in \mathbb{R}^n$ is a local minimum, then $\nabla f(\mathbf{x}) = 0$, hence $f(\mathbf{y}) \geq f(\mathbf{x})\ \forall \mathbf{y} \in \mathbb{R}^n$. Therefore, $\mathbf{x}$ is a global minimum and we proved the following

**Corollary 3.5.2.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ in $\mathcal{C}^1$ be a convex function. $\mathbf{x} \in \mathbb{R}^n$ is a stationary point for $f$ iff $\mathbf{x}$ is a global minimum.*

**Fact 3.5.3.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable ($f \in \mathcal{C}^2$). $f$ is convex on an open set $S \subseteq \mathbb{R}^n$ iff the Hessian is positive semidefinite.*

*Moreover, $f$ is strictly (strongly) convex iff $\nabla^2 f(\mathbf{x}) \succ 0$ ($\nabla^2 f(\mathbf{x}) \geq \tau I$).*

This proposition gives us an algorithm to checc k if a function is convex or not: we only need to compute the eigenvalues of the Hessian and check if they are positive.

There are some functions which do not have differentiability property and for those who are interested there is a section in **??**.

A way to work with functions which are not defined on all $\mathbb{R}^n$ is to solve the following problem:

$$(P) \equiv \inf\{f_X(\mathbf{x}) = f(\mathbf{x}) + \mathbb{1}_X(\mathbf{x}) \ : \ \mathbf{x} \in \mathbb{R}^n\}$$

thanks to the following

**Theorem 3.5.4** (Essential objective)**.** *$\mathbf{x}_*$ optimal for $(P)$ $\iff$ $\mathbf{x}_*$ local minimum of $f_X$.*

So far, we provided two different approaches for checking if a function is convex: one requires to derive the function from some basic, continuous functions using the operations allowed. Conversely, we can make use of higher order information when it is possible and check some properties, thanks to characterization theorems.

## 3.6 Subgradients and Subdifferentials

**Definition 3.6.1** (Subgradient)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. We say that $\mathbf{s} \in R^n$ is a **subgradient** of $f$ at point $\mathbf{x} \in \mathbb{R}^n$ if $\forall \mathbf{y} \in \mathbb{R}^n$ the following holds:*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + s(\mathbf{y} - \mathbf{x})$$

Let us assume that the minimum of the non differentiable function resides in one of its kinky points, then for $\mathbf{s} = \mathbf{0}$ we have a subgradient which is flat and this is a sufficient condition for minimality, for a pictorial example see Figure 3.15.

The issue here is that it is unfeasible to check if $\mathbf{s} = \mathbf{0}$ is a subgradient for $f$, since we should check all possible $\mathbf{y} \in \mathbb{R}^n$.

**Definition 3.6.2** (Subdifferential)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ and let $\mathbf{x} \in \mathbb{R}^n$. We call **subdifferential** the set of all possible subgradients at $\mathbf{x} \in \mathbb{R}^n$.*

*Formally,*

$$\partial f(\mathbf{x}) := \{\mathbf{s} \in \mathbb{R}^n \ : \ \mathbf{s} \text{ is a subgradient at } \mathbf{x}\}$$

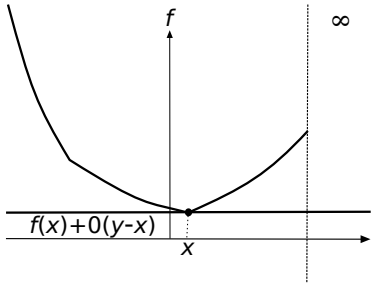



FIGURE 3.15: Pictorial example of subgradients of a non diffrentaible function.

**Definition 3.6.3** (Descent direction)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ and let $\mathbf{x} \in \mathbb{R}^n$. We say that $\mathbf{d} \in \mathbb{R}^n$ is a **descent direction** if $< \mathbf{s}, \mathbf{d} >< 0 \ \forall \ \mathbf{s} \in \partial f(\mathbf{x})$.*

**Theorem 3.6.1.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. $\mathbf{x}$ global minimum $\iff 0 \in \partial f(\mathbf{x})$.*

Notice that in general, when we are not in proximity of a border (where $f$ is unbounded above) we get that the subdifferential is a compact interval.

Formally,$\partial f(\mathbf{x})$ closed and convex, compact $\forall \mathbf{x} \in int\, dom(f)$.

Moreover, we can prove the following

**Fact 3.6.2.** *Let $f : \mathbb{R}^n \to \mathbb{R}$.*

$$\partial f(\mathbf{x}) = \{\nabla f(\mathbf{x})\} \iff f \text{ differentiable at } \mathbf{x}$$

The following fact tries to provide a similar characterization of the subdifferential.

**Fact 3.6.3.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ and let $\mathbf{x} \in \mathbb{R}^n$. If $\frac{\partial f}{\partial \mathbf{d}}(\mathbf{x}) = \sup\{< \mathbf{s}, \mathbf{d} > : \ \mathbf{s} \in \partial f(\mathbf{x})\}$ then $\mathbf{d}$ is a descent direction $\iff < \mathbf{s}, \mathbf{d} >< 0 \ \forall \ \mathbf{s} \in \partial f(\mathbf{x})$.*

As in the differentiable case, we are interested in moving in the steepest descent direction, formally $\mathbf{s}_* = -\text{argmin}\{\|\mathbf{s}\| \ : \ \mathbf{s} \in \partial f(\mathbf{x})\}$.

**Fact 3.6.4** (Linearity of subdifferential)**.** *Let $f, g : \mathbb{R}^n \to \mathbb{R}$ and take $\alpha, \beta \in \mathbb{R}_+$, then $\partial[\alpha f + \beta g](\mathbf{x}) = \alpha \partial f(\mathbf{x}) + \beta \partial g(\mathbf{x})$.*

**Fact 3.6.5** (Chain rule)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$.*

- *Let $A \in M(n, \mathbb{R})$ and $\mathbf{b} \in R^n$ then $\partial[f(A\mathbf{x} + \mathbf{b})] = A^T[\partial f](A\mathbf{x} + \mathbf{b})$;*
- *Let $g : \mathbb{R} \to \mathbb{R}$ be an increasing function. Then $\partial[g(f(\mathbf{x}))] = [\partial g](f(\mathbf{x}))[\partial f](\mathbf{x})$.*

**Definition 3.6.4** ($\varepsilon$-subgradient)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. We say that $\mathbf{s} \in \mathbb{R}^n$ is $\varepsilon$-**subgradient** at $\mathbf{x} \in \mathbb{R}^n$ if it defines a support hyperplane passing $\varepsilon$ below $epi(f)$. Formally,*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{s}(\mathbf{y} - \mathbf{x}) - \varepsilon \ \forall \mathbf{y} \in \mathbb{R}^n$$

**Fact 3.6.6.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ and let $\mathbf{x} \in \mathbb{R}^n$. Saying that $\mathbf{0} \in \mathbb{R}^n$ is a $\varepsilon$-subgradient for $f$ in $\mathbf{x}$ means that the function value $f(\mathbf{x})$ cannot be at distance grater than $\varepsilon$ from the minimum $f(\mathbf{x}_*)$.*

*Formally, $0 \in \partial_\varepsilon f(\mathbf{x}) \iff \mathbf{x}$ is $\varepsilon$-optimal.*

We are now allowed to compute $\mathbf{s}_* = \operatorname{argmin}\{\|\mathbf{s}\| \ : \ \mathbf{s} \in \partial_\varepsilon f(\mathbf{x})\}$.

If $\mathbf{s}_* = \mathbf{0}$ then $\mathbf{x}$ is $\varepsilon$ optimal. Otherwise, $\exists \alpha > 0$ s.t. $f(\mathbf{x} - \alpha \mathbf{s}_*) \leq f(\mathbf{x}) - \varepsilon$ ($-\mathbf{s}_*$ is of $\varepsilon$-descent).

The $\varepsilon$-subgradient is very powerful, but the issue is that is even more expensive to compute than the subgradient.

# Chapter 4

# Unconstrained Optimization

**Do you recall?**

We are interested in finding the minimum of a function through an iterative procedure, such that we start from an initial guess $\mathbf{x}^0$ and go on ($\mathbf{x^i} \rightsquigarrow \mathbf{x^{i+1}}$). We want to move towards the optimum.
Notice that with $\mathbf{x^i} \in \mathbb{R}^n$ we refer to the point $\mathbf{x}$ at $i$-th iteration.

In general, when we talk about optimization we need to carefully choose a starting point $\mathbf{x}^0$ that will determine the behaviour of the convergence. Moreover, we will stop whenever a stationary point is encountered, but such point is guaranteed to be a global minimum only if the objective function is convex. Notice that a minimizing sequence $\{\mathbf{x^i}\}$ may not lead to a minimum, because the codomain is *unbounded*, therefore we need to converge to an accumulation point.

How to be sure that we are in an optimum?

- (strong) $\{\mathbf{x^i}\} \to \mathbf{x}_*$: the whole sequence converges to an optimal solution;
- (weaker) all accumulation points of $\{\mathbf{x^i}\}$ are optimal solutions;
- (weakest) at least one accumulation point of $\{\mathbf{x^i}\}$ is optimal.

The iterative process of moving from a point $\mathbf{x^i}$ to $\mathbf{x^{i+1}}$, that is supposed to have a lower function value can be held in two different ways:

LINE SEARCH**:** first choose the direction of the movement $\mathbf{d^i} \in \mathbb{R}^n$, then choose $\alpha^i \in \mathbb{R}$ (that we term *stepsize* or equivalently "learning rate") s.t. $\mathbf{x^{i+1}} \leftarrow \mathbf{x^i} + \alpha^i \mathbf{d^i}$;

TRUST REGION: first choose $\alpha^i$ (that we call *trust radius* and somehow indicates the ball in which we think that the information that we have is accurate), then choose $\mathbf{d^i}$.

In both these alternatives, it is crucial to choose a proper model to approximate the objective function $f$. The simplest model we can build is a linear one, namely $L^i(\mathbf{x}) = L_{\mathbf{x}^i}(\mathbf{x}) = f(\mathbf{x^i}) + \nabla f(\mathbf{x^i})^T(\mathbf{x} - \mathbf{x^i})$ and find the direction along the one the model decreases the most, provided that we are reasonably sure that the model approximates well the function around the current point $\mathbf{x^i}$.

The first order information is accurate only around $\mathbf{x^i}$ and this is why we should not move too far from $\mathbf{x^i}$, so we want a step size $\alpha_i \to 0$.

In the following sections, we will dig into the so-called *gradient descent algorithms*, where we pick the steepest descent direction as

$$\mathbf{d^i} = \arg\min_{\mathbf{d}} \left\{ \lim_{t \to 0} \frac{f(\mathbf{x} + t\mathbf{d})}{t} \right\} = -\nabla f(\mathbf{x^i}) \qquad \text{(SDQ: direction)}$$

Notice that a too short step is bad either, because the gain in the value of the function is very little.
Alternatively, we can choose to pick a different step size at each iteration by solving a "small" minimization problem at each iterate:

$$\alpha^i = \arg\min_{\alpha \geq 0} f(\mathbf{x^i} + \alpha \mathbf{d^i}) \qquad \text{(SDQ: step size)}$$

Notice that choosing linear functions to approximate the objective function in a point $\mathbf{x}$ has the drawback that linear functions are unbounded below.

In the following section we present the *steepest descent algorithm* in the simple case of quadratic objective functions that are the easiest kind of objective functions that one can study.

## 4.1 Gradient Method for Quadratic Functions

In this section, we will deal with the following optimization problem (P), where $f : \mathbb{R}^n \to \mathbb{R}$ is a *quadratic function* and it is formalized as

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q \mathbf{x} + \mathbf{q}^T \mathbf{x}$$

where $Q \succeq 0$, otherwise $f$ could be unbounded below (as stated in Proposition 2.8.3).
The minimum is the point in which the gradient ($\nabla f(\mathbf{x}) = Q\mathbf{x} + \mathbf{q}$) is 0. Such condition, whenever $Q$ is non singular, is reached in $\mathbf{x} = Q^{-1}\mathbf{q}$, but solving a linear system requires $O(n^3)$ and this is why we introduce an iterative method for computing the optimum.
A formalization of the gradient descent algorithm for quadratic functions can be found in Algorithm 4.1.1.

ALGORITHM 4.1.1 Pseudocode for **S**teepest **D**escent method for detecting the minimum of **Q**uadratic functions.

```
1: procedure SDQ(f, x, ε)
2:     while (‖∇f(x)‖ > ε) do
3:         d ← −∇f(x);
4:         α ← ‖d‖² / (d^T Q d);
5:         x ← x + αd;
6:     end while
7: end procedure
```

As usual, we need to choose an error threshold $\varepsilon$ under the one we feel allowed to believe that the optimum has been reached and then we can stop; we set it as the norm of the gradient. The algorithmic parameter $\varepsilon$ could be provided by the stakeholder, because he knows how accurate the answer needs to be.

In Algorithm 4.1.1 we can see that the step size $\alpha$ is computed using a closed formula, instead of the minimum problem of Equation (SDQ: step size). How is that formula obtained? Let us define $\varphi : \mathbb{R} \to \mathbb{R}$ s.t. $\varphi(\alpha) = f(\mathbf{x^i} + \alpha\mathbf{d^i})$, we are looking for the minimum of that function $\varphi$:

$$
\begin{aligned}
\min_\alpha \varphi(\alpha) &= \min_\alpha f(\mathbf{x^i} + \alpha\mathbf{d^i}) \\
&= \min_\alpha \frac{1}{2}(\mathbf{x^i} + \alpha\mathbf{d^i})^T Q(\mathbf{x^i} + \alpha\mathbf{d^i}) + \mathbf{q}^T(\mathbf{x^i} + \alpha\mathbf{d^i}) \\
&= \min_\alpha \frac{1}{2}\mathbf{x^i}^T Q\mathbf{x^i} + \frac{1}{2}\mathbf{x^i}^T Q\alpha\mathbf{d^i} + \frac{1}{2}\alpha\mathbf{d^i}^T Q\mathbf{x^i} + \frac{1}{2}\alpha\mathbf{d^i}^T Q\alpha\mathbf{d^i} + \mathbf{q}^T\mathbf{x^i} + \alpha\mathbf{q}^T\mathbf{d^i} \\
&\overset{(1)}{=} \min_\alpha \frac{1}{2}\mathbf{x^i}^T Q\mathbf{x^i} + \alpha\mathbf{x^i}^T Q\mathbf{d^i} + \frac{1}{2}\alpha^2\mathbf{d^i}^T Q\mathbf{d^i} + \mathbf{q}^T\mathbf{x^i} + \alpha\mathbf{q}^T\mathbf{d^i} \\
&\overset{(2)}{=} \min_\alpha \alpha\mathbf{x^i}^T Q\mathbf{d^i} + \frac{1}{2}\alpha^2\mathbf{d^i}^T Q\mathbf{d^i} + \alpha\mathbf{q}^T\mathbf{d^i} \\
&= \frac{1}{2}\alpha^2\mathbf{d^i}^T Q\mathbf{d^i} + \min_\alpha \alpha\mathbf{x^i}^T Q\mathbf{d^i} + \alpha\mathbf{q}^T\mathbf{d^i} \\
&= \min_\alpha \left( \left( \frac{1}{2}\mathbf{d^i}^T Q\mathbf{d^i} \right)\alpha^2 + \alpha \cdot \left( \mathbf{x^i}^T Q + \mathbf{q}^T \right)\mathbf{d^i} \right)
\end{aligned}
$$

where $\overset{(1)}{=}$ is due to the fact that $Q$ is symmetric and $\overset{(2)}{=}$ follows from the fact that $\frac{1}{2}\mathbf{x^i}^T Q\mathbf{x^i}$ and $\mathbf{q}^T\mathbf{x^i}$ are constant with respect to $\alpha$. We need to minimize the quadratic function in one variable of the shape $\varphi(\alpha) = a\alpha^2 + b\alpha$, where $a = \frac{1}{2}\mathbf{d^i}^T Q\mathbf{d^i}$ and $b = \mathbf{d^i}^T(Q\mathbf{x^i} + \mathbf{q})$. We know that the minimum of $\varphi$ can be found imposing the derivative ($\varphi'(\alpha) = 2a\alpha + b$) to 0: $2a\alpha + b = 0 \iff \alpha = -\frac{b}{2a}$, hence:

$$\begin{aligned}\alpha &= \frac{-\mathbf{d^i}^T(Q\mathbf{x^i}+\mathbf{q})}{\not{2}\frac{1}{\not{2}}\mathbf{d^i}^T Q\mathbf{d^i}} \\ &\overset{(1)}{=} \frac{-\mathbf{d^i}^T(-\mathbf{d^i})}{\mathbf{d^i}^T Q\mathbf{d^i}} \\ &= \frac{\|\mathbf{d^i}\|^2}{\mathbf{d^i}^T Q\mathbf{d^i}}\end{aligned}$$

where $\overset{(1)}{=}$ follows from the fact that we chose $\mathbf{d^i}$ as the anti-gradient. Notice that the computational complexity of each step of Algorithm 4.1.1 is $O(n^2)$ for computing the descent direction $\mathbf{d^i}$ and $O(n^2)$ for choosing the step size $\alpha^i$.

Notice that steepest descent method generates points that move along orthogonal directions, as depicted in Figure 4.1 and formalized in Proposition 4.1.1.

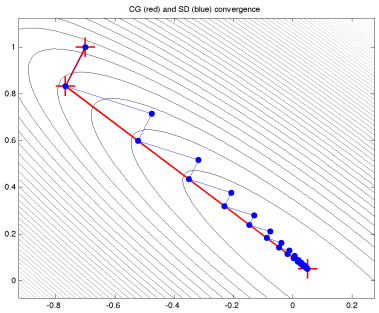



FIGURE 4.1: Some iterations of the gradient method

**Fact 4.1.1.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a quadratic function and let us consider the gradient descent algorithm for minimizing $f$, where at iteration $i$ the moving direction $\mathbf{d^i} \in \mathbb{R}^n$ and the step size $\alpha^i \in \mathbb{R}$ are computed. The following holds:* $< \mathbf{d^i}, \mathbf{d^{i+1}} > = 0$.

*Proof.* Let us rewrite $\mathbf{d^{i+1}}$ in terms of $\mathbf{d^i}$:

$$\begin{aligned}\mathbf{d^{i+1}} &:= -\nabla f(\mathbf{x^{i+1}}) \\ &= -Q\mathbf{x^{i+1}} - \mathbf{q} \\ &= -Q(\mathbf{x^i} + \alpha^i \mathbf{d^i}) - \mathbf{q} \\ &= -Q\mathbf{x^i} - Q\alpha^i \mathbf{d^i} - \mathbf{q} \\ &= \mathbf{d^i} - Q\alpha^i \mathbf{d^i}\end{aligned}$$

If we plug this into the scalar product $< \mathbf{d^i}, \mathbf{d^{i+1}} >$ we get

$$\begin{aligned} < \mathbf{d^i}, \mathbf{d^{i+1}} > &=< \mathbf{d^i}, \mathbf{d^i} - Q\alpha^i \mathbf{d^i} > \\ &= \left\|\mathbf{d^i}\right\|^2 - \mathbf{d^i}^T Q\alpha^i \mathbf{d^i} \\ &= \left\|\mathbf{d^i}\right\|^2 - \alpha^i \mathbf{d^i}^T Q\mathbf{d^i} \\ &\overset{*}{=} \left\|\mathbf{d^i}\right\|^2 - \frac{\left\|\mathbf{d^i}\right\|^2}{\cancel{\mathbf{d^i}^T Q\mathbf{d^i}}} \cdot \cancel{\mathbf{d^i}^T Q\mathbf{d^i}} \\ &= 0 \end{aligned}$$

where $\overset{*}{=}$ follows from $\alpha^i := \frac{\left\|\mathbf{d^i}\right\|^2}{\mathbf{d^i}^T Q\mathbf{d^i}}$. □

**Fact 4.1.2.** *Let us take a minimum problem (P) such that* $\{\mathbf{x^i}\}$ *converges to* $\mathbf{x}$. *Then* $\mathbf{x}$ *is a stationary point for the function* $f$.

*Proof.* We know that $\mathbf{d^i} \perp \mathbf{d^{i+1}}$, hence $< \nabla f(\mathbf{x^i}), \nabla f(\mathbf{x^{i+1}}) >= 0$. The thesis follows from

$$\lim_{i\to\infty} \underbrace{< \nabla f(\mathbf{x^i}), \nabla f(\mathbf{x^{i+1}}) >}_{0} =< \nabla f(\mathbf{x}), \nabla f(\mathbf{x}) >$$

□

Moreover, the following (that we won't prove) holds.

**Fact 4.1.3.** *Let us take a minimum problem (P) such that* $\{\mathbf{x^i}\}$ *has an accumulation point* $\mathbf{x}$. *Then* $\mathbf{x}$ *is also a stationary point for the function* $f$.

### Efficiency

In the rest of this lecture we are going to assess the convergence speed of this algorithm.
In general, showing how fast $\left\|\mathbf{x^i} - \mathbf{x}_*\right\|$ decreases is more involved that showing how fast $f(\mathbf{x^i}) - f_*$ decreases, but we do not know $f_*$. We concentrate on computing $\lim_{i\to\infty} \frac{f(\mathbf{x^{i+1}})-f_*}{f(\mathbf{x^i})-f_*{}^p} = R$. In principle an algorithm is fast whenever the gap between $f(^{i+1})$ and $f_*$ decreases rapidly. According to the values of $p$ and $R$ we get the following alternatives (displayed in Figure 4.2):

SUBLINEAR: $p = 1, R = 1$;

LINEAR: $p = 1, R < 1$;

SUPERLINEAR: $p = 1, R = 0$;

QUADRATIC: $p = 2, R > 0$.

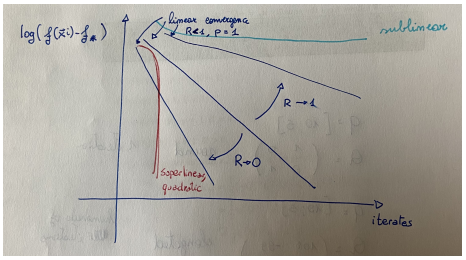



FIGURE 4.2: Some values of the convergence speed on a logarithmic scale.

**Fact 4.1.4.** *Suppose that we are using an algorithm that converges linearly. It converges at an error of* $\varepsilon > 0$ *in* $O(\log(1/\varepsilon))$.

Let us compute algebraically the function value in the optimum ($\mathbf{x}_* = -Q^{-1}\mathbf{q}$, when $Q$ is non singular):

$$\begin{aligned} f(\mathbf{x}_*) &= \frac{1}{2}(-Q^{-1}\mathbf{q})^T Q(-Q^{-1}\mathbf{q}) - \mathbf{q}^T Q^{-1}\mathbf{q} \\ &= \frac{1}{2}\mathbf{q}^T {Q^{-1}}^T \cancel{Q Q^{-1}}\mathbf{q} - \mathbf{q}^T Q^{-1}\mathbf{q} \\ &\overset{(*)}{=} \frac{1}{2}\mathbf{q}^T Q^{-1}\mathbf{q} - \mathbf{q}^T Q^{-1}\mathbf{q} \\ &= -\frac{1}{2}\mathbf{q}^T Q^{-1}\mathbf{q} \end{aligned}$$

where $\overset{*}{=}$ follows from the fact that $Q$ is symmetric.

At this point, we need to assess the convergence speed of the algorithm, thus we need to define an error function $\bar{f}$, such that "the error at $\mathbf{x}$ is the distance between $\mathbf{x}$ and $\mathbf{x}_*$ in $\|\cdot\|_Q$":

Rispetto alle slide, qui denoto con $\bar{f}(\cdot)$ la funzione che quantifica l'errore, anzichè chiamarla $f_*(\cdot)$, perchè questa seconda notazione si confonde con $f_*$ (senza argomento), che è il valore ottimo della funzione obiettivo.

$$\begin{aligned}
\bar{f}(\mathbf{x}) &\overset{(1)}{=} \frac{1}{2}(\mathbf{x}-\mathbf{x}_*)^T Q(\mathbf{x}-\mathbf{x}_*) \\
&= \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \frac{1}{2}\mathbf{x}_*{}^T Q\mathbf{x}_* - \mathbf{x}^T(Q\mathbf{x}_*) \\
&\overset{(2)}{=} \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \frac{1}{2}(Q^{-1}\mathbf{q})^T Q(Q^{-1}\mathbf{q}) - \mathbf{x}^T \cancel{Q}(\cancel{-Q^{-1}}\mathbf{q}) \\
&= \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \frac{1}{2}Q^{-1}\mathbf{q}^T Q(Q^{-1}\mathbf{q}) + \mathbf{q}^T\mathbf{x} \\
&= \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \frac{1}{2}\mathbf{q}^T \cancel{Q^{-1}}\cancel{Q}Q^{-1}\mathbf{q} + \mathbf{q}^T\mathbf{x} \\
&= \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \frac{1}{2}\mathbf{q}^T Q^{-1}\mathbf{q} + \mathbf{q}^T\mathbf{x} \\
&= \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T\mathbf{x} + \frac{1}{2}\mathbf{q}^T Q^{-1}\mathbf{q} \\
&= f(\mathbf{x}) - f(\mathbf{x}_*)
\end{aligned} \tag{4.1.1}$$

where $\overset{(1)}{=}$ follows from Proposition 2.8.2 because we translated the origin in the minimum and $\overset{(2)}{=}$ follows from the equality $\mathbf{x}_* = -Q^{-1}\mathbf{q}$.

**Do you recall?**

The family of quadratic functions is the simplest possible family of functions where a minimum exists. A quadratic function is defined as: $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T\mathbf{x}$, so its gradient is computed as $\nabla f(\mathbf{x}) = Q\mathbf{x} + \mathbf{q}$. We are interested in finding local minima of the function $f$.

Let us go back to the error function

$$\bar{f}(\mathbf{x^{i+1}}) = f(\mathbf{x^{i+1}}) - f_* = \frac{1}{2}(\mathbf{x^{i+1}} - \mathbf{x}_*)^T Q(\mathbf{x^{i+1}} - \mathbf{x}_*)$$

The $(i+1)$th iteration is computed as

$$\begin{aligned}
\mathbf{x^{i+1}} &= \mathbf{x^i} + \alpha^i \mathbf{d^i} \\
&\overset{(a)}{=} \mathbf{x^i} + \frac{\left\|\mathbf{d^i}\right\|^2}{\mathbf{d^i}{}^T Q\mathbf{d^i}}\mathbf{d^i} \\
&\overset{(b)}{=} \mathbf{x^i} + \frac{\left\|-Q\mathbf{x^i} - \mathbf{q}\right\|^2}{(-Q\mathbf{x^i} - \mathbf{q})^T Q(-Q\mathbf{x^i} - \mathbf{q})}
\end{aligned}$$

where $\overset{(a)}{=}$ follows from the definition $\alpha^i := \frac{\left\|\mathbf{d^i}\right\|^2}{\mathbf{d^i}{}^T Q\mathbf{d^i}}$ and $\overset{(b)}{=}$ from the fact that the moving direction is the opposite of the gradient (formally, $\mathbf{d^i} = -Q\mathbf{x^i} - q$).

This means not only **linear convergence**, but a bit more, because linear convergence only takes into consideration steps in proximity to the limit, while this formula holds at the beginning as well.

**Fact 4.1.5.** *Let $A \in M(n, \mathbb{R})$ be a positive semidefinite matrix. $\forall \mathbf{v} \in \mathbb{R}^n$ $\mathbf{v}^T A \mathbf{v} = \|\mathbf{v}\|_A^2$.*

**Fact 4.1.6.** *Let us consider a minimum problem (P) where the objective function is quadratic. In the setting of a gradient descent method if $Q$ is positive definite the error decreases by a constant factor at each iteration. Formally, if $Q \succ 0$ we can say that the error goes like $1 - \left( \frac{\left\|\mathbf{d^i}\right\|^4}{(\mathbf{d^i}^T Q \mathbf{d^i})(\mathbf{d^i}^T Q^{-1} \mathbf{d^i})} \right)$ or, equivalently, $1 - \frac{\left\|\mathbf{d^i}\right\|_I^2}{\|\mathbf{d^i}\|_Q{}^2} \cdot \frac{\left\|\mathbf{d^i}\right\|_I^2}{\|\mathbf{d^i}\|_{Q^{-1}}{}^2}$.*

Fare proof, $\bar{f}(\mathbf{x^{i+1}}) = 1 - \left( \frac{\left\|\mathbf{d^i}\right\|^4}{(\mathbf{d^i}^T Q \mathbf{d^i})(\mathbf{d^i}^T Q^{-1} \mathbf{d^i})} \right) \bar{f}(\mathbf{x^i})$(hint: for $y^i = x^i - x_*$, $d^i = Qy^i$)

We are measuring a vector with three different norms. We would like to estimate $\frac{<\mathbf{d_i}, \mathbf{d_i}>}{\mathbf{d_i}^T Q \mathbf{d_i}}$.

At this point we would like to find a formula for computing $f(\mathbf{x^{i+1}}) - f(\mathbf{x^i})$ that holds for all the iterates. In order to achieve such result we need to discuss a bit of linear algebra.

**Fact 4.1.7.** *Let $A \in M(n, \mathbb{R})$ be a positive semidefinite matrix. Given $\lambda_i$ the eigenvalues of $A$, $\frac{1}{\lambda_i}$ are the eigenvalues of the matrix $A^{-1}$.*

We can say that $\lambda_n \|\mathbf{x}\|^2 \leq \mathbf{x}^T Q \mathbf{x} \leq \lambda_1 \|\mathbf{x}\|^2$, where $\lambda_n$ is the smallest eigenvalue, while $\lambda_1$ is the largest.

We are looking for a close formula for calculating the convergence rate, since it depends recursively by the steps done. So we want to perform a worst case analysis in order to find a faster way to calculate the convergence rate.

We want to prove that the ratio $R$ is smaller than 1 so we are looking for an upper-bound.

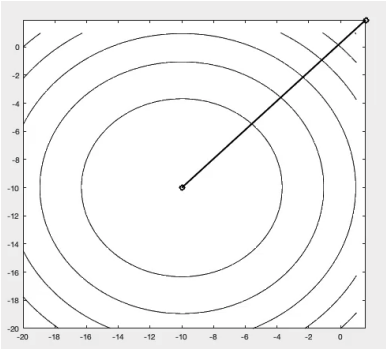
A coarse upper-bound is $(1 - \frac{\lambda_n}{\lambda_1})$, but we can prove more:

**Fact 4.1.8.** *Let $A \in M(n, \mathbb{R})$ be a positive definite matrix. Given $\lambda_i$ the eigenvalues of $A$,*

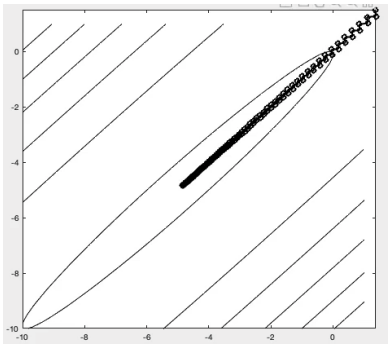
$$\forall\, \mathbf{v} \in \mathbb{R}^n \quad \frac{\|\mathbf{v}\|^4}{(\mathbf{v}^T A \mathbf{v})(\mathbf{v}^T A^{-1} \mathbf{v})} \geq \frac{4\lambda^1 \lambda^n}{(\lambda^1 + \lambda^n)^2}$$

We won't see the proof of this fact.

$R$ close to 0 means that the algorithm is converging fast, so when the largest eigenvalue ($\lambda_1$) and the smallest eigenvalue ($\lambda_n$) are very close to each-other the algorithm is converging fast. We can say that, since the eigenvalues represent the length of the axes of the ellipsoids (level sets), the algorithm is converging fast when the ellipsoids have a round shape. This fact is enforced by experimental results, shown in Figure 4.3.



(a) $Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $q^T = (10, 5)$. The convergence takes only one iteration



(b) $Q = \begin{pmatrix} 101 & -99 \\ -99 & 101 \end{pmatrix}$, $q^T = (10, 5)$. The convergence takes hundreds of iterations.

FIGURE 4.3: Convergence rates.

**Fact 4.1.9.** *Let us consider a minimum problem (P) where the objective function is quadratic. In the setting of a gradient descent method the convergence estimate is*

$$f(\mathbf{x^{i+1}}) - f(\mathbf{x}_*) \leq \left(\frac{\lambda^1 - \lambda^n}{\lambda^1 + \lambda^n}\right)^2 \cdot \Big(f(\mathbf{x^i}) - f(\mathbf{x}_*)\Big)$$

At this point, provided that we can estimate the number of iterations needed to reach convergence, how can we say that such a number is a good result? As usual, it depends on how that number is obtained. Since it is *dimensional independent* it is very good, because it scales well (when the size of the space - number of variables - increases). It only depends on the conditioning of the matrix $Q$.

Of course as $n$ grows $Q$ changes, so in practice it may happen that the conditioning of the problem is worsening as $n$ grows.

If the balls are very rounded the zig zags needed to start converging are very few.

**Note**

So far, we provided a bound for the convergence speed of the algorithm when $Q$ is positive definite. What can we say when $Q$ is positive semidefinite? The algorithm works, but we can't provide an upper-bound for the convergence rate. We are even more restrictive when dealing with machine precision, since if there is an eigenvalue which is bigger than zero, but very close to zero, it turns out to be 0 on the machine, so we cannot provide an upper-bound. We will see how to deal with this case. TODO: mettere reference

### Some experiments on SDQ implemented in Matlab

When we are trying to move from theory to practice we need to take into account some practical issues, such as picking a good value for $\varepsilon$. A good idea would be the norm of the gradient, but in order not to end up in a loop we need to provide also some performances bound, e.g. the maximum number of iterations or the maximum amount of time.

Let us suppose that we implemented the quadratic gradient descent algorithm in Matlab, with the following signature:

```
1 function[x, status] = SDQ(Q, q, x, fStar, eps, MaxIter)
```

where `fStar` is the optimal value, which is used to provide an estimate of the convergence.

**Note**

When implementing functions for optimization is important to:

- check the coherence of input values (namely, if the user passed allowed parameters);
- give all possible information about your result (for example if the algorithm stopped because the maximum number of iterations was reached or because the epsilon value was reached);
- check at any iteration the values of variables (e.g. before dividing by a quantity that may be smaller than the precision check it);
- design a good log, in order to understand what is going on at each step.

**Example 4.1.1.** *In the implementation of SDQ* provided by professor Frangioni, the execution keeps track of the actual ratio between the error at one step and the error at the next step. This information provides insights about how many orders of magnitude the error decreases. We can find starting points where the ratio is exactly $R$. We can observe that when the conditioning is quite good the error decreases faster than the $R$ limitation, but as soon as we change the values for $Q$ and $\mathbf{q}$ things may change completely.*

*If the reader wants to run some examples, he should notice that if the conditioning grows, the number of iterations needed to find the minimum increases as well.*

*Running the algorithm on some examples shows that the theoretical results are reflected well in the practical case.*

*https://elearning.di.unipi.it/mod/resource/view.php?id=5033

### Error

As mentioned above, if we picked as starting point for the algorithm a value that does not lead to a minimizing sequence, we need to stop iterating after a while, possibly when we are close enough to the solution, i.e. the **absolute error** $\varepsilon_A$ is small enough: $\varepsilon_A = f(\mathbf{x^i}) - f_* \leq \varepsilon$. It is in this context that we introduce the concept of **relative error**, that compares the error with the value of the function.

**Definition 4.1.1** (Relative error)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a quadratic function and let us consider the gradient descent algorithm for minimizing $f$, where at iteration $i$ the moving direction $\mathbf{d^i} \in \mathbb{R}^n$ and the step size $\alpha^i \in \mathbb{R}$ are computed. We term **relative error** at the $i$-th iteration with respect to the optimal value and denote $\varepsilon_R$*

$$\varepsilon_R = \frac{(f(\mathbf{x^i}) - f_*)}{|f_*|} = \frac{\varepsilon_A}{|f_*|} \leq \varepsilon$$

This relative error is invariant for scaling transformations. Notice that if we assume that $f^*$ might be zero the formula should be changed.

In practice, computing such error is unfeasible because we do not know $f^*$. In this very common case we can substitute $f^*$ with a good lower bound $\underline{f} \leq f^*$for $f^*$, but in this course we will not focus on finding $\underline{f}$, therefore we will use as stopping conditions the bounds on the norm of the gradient:

- $\left\|\nabla f(\mathbf{x^i})\right\| \leq \varepsilon$ ("absolute version")
- $\frac{\left\|\nabla f(\mathbf{x^i})\right\|}{\|\nabla f(\mathbf{x^0})\| \leq \varepsilon}$ ("relative version")

The second stopping condition is expressed in relation to the value of the norm of the gradient at the starting point.

We usually choose the norm of the gradient as a threshold for precision, but we do not know how this quantity relates to $\varepsilon_A$ or $\varepsilon_R$.

**Example 4.1.2.** *Let $f : \mathbb{R} \to \mathbb{R}$ a convex function and let $X = \mathcal{B}(\mathbf{0}, r)$. Estimate $\varepsilon_A$ when $\left\|\nabla f(\mathbf{x^i})\right\| \leq \varepsilon$. Provided that we are studying a convex function we can easily find the minimum in a ball. We can then minimize the linear function in the range of the ball and that minimum is surely a lower bound.*

*Let us assume that we are guaranteed that the optimum lies in the interval $[l, u]$. The first order model at a generic iterate $x^i$ lies below the function. The linear estimate is* bounded *below on a compact set, hence we know that $f_* \geq L_{x^i}(u)$, as displayed in Figure 4.4.*

$$L_{x^i}(y) = f(x^i) + \nabla f(x^i) \underbrace{(y - x^i)}_{(u-l)}$$

*hence in general the error goes like $\left\|\nabla f(x^i)\right\| (u-l)$, therefore whenever $\left\|\nabla f(x^i)\right\| \leq \varepsilon$ the actual error $\varepsilon_A$ is $\varepsilon(u - l)$.*

As we said so far, minimizing with quadratic functions is pretty straightforward. In this chapter and a few subsequent ones, we will explore functions that have different shapes.

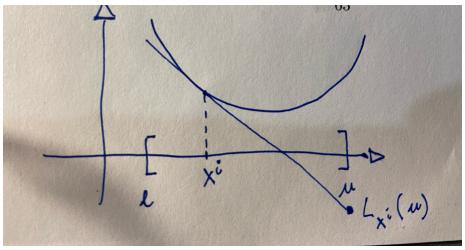



FIGURE 4.4

## 4.2 Gradient Method for Non Quadratic Functions

## 4.3 Gradient method for non quadratic functions

**Do you recall?**

Let us consider that we are sitting in a point $\mathbf{x^i}$ at the $i$-th iteration. In order to compute the next step, in the case of quadratic functions, we found a closed formula for the step size $\alpha^i$:

$$\alpha^i = \arg\min\{f(\mathbf{x} + \alpha\mathbf{d})\} = \frac{\|\mathbf{d^i}\|^2}{\mathbf{d^i}^T Q \mathbf{d^i}}$$

Such formula depends on the shape of the quadratic function $f$, but does not hold in general. Conversely, we may recall from last lecture that the proof of the orthogonality of the gradient (Proposition 4.1.1) does not depend on the quadratic nature of our functions, so it could work in the non-quadratic case as well, therefore Proposition 4.1.2 holds for non quadratic functions as well. Such proposition states that a sequence of iterates $\{\mathbf{x^i}\}$ converges to a value $\mathbf{x}$ iff $\mathbf{x}$ is a stationary point. Notice that the function $f$ must be at least $\mathcal{C}^1$.

The algorithm for finding local minima of non quadratic functions has the same structure of the one used for quadratic ones, i.e. first compute the direction of the step and then compute its size and it is formalized in Algorithm 4.3.1.

ALGORITHM 4.3.1 Pseudocode for **S**teepest **D**escent method for detecting the minimum of **N**on-**Q**uadratic functions.

```
1: procedure SNDQ(f, x, ε)
2:     while (‖∇f(x)‖ > ε) do
3:         d ← −∇f(x);
4:         α ← arg min f(x + αd);
5:         x ← x + αd;
6:     end while
7: end procedure
```

We will see that, differently from the quadratic case (where the gradient was $\nabla f(\mathbf{x}) = Q\mathbf{x} + \mathbf{q}$) computing the gradient in this more general case is not trivial.

The stepzise $\alpha^i$ should be computed finding the minimum of $\varphi(\alpha) = f(\mathbf{x^i} + \alpha\mathbf{d^i})$, but in the non-quadratic case $\varphi$ is non convex, hence finding the global minimum is an NP-hard problem. We will see that for our purposes finding a local minimum is enough, hence we *relax the complexity constraint.*

We are now interested in assessing the convergence speed of Algorithm 4.3.1; we can prove the following theorem that states that in the tail of the convergence process the function gets similar to a quadratic function.

**Theorem 4.3.1** (Convergence speed)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function in $\mathcal{C}^2$ and let $\mathbf{x}_*$ be a local minimum for $f$ such that the Hessian of $f$ is strictly positive definite ($\nabla^2 f(\mathbf{x}_*) \succ 0$). If $\{\mathbf{x^i}\}$ converges it is a minimizing sequence. Formally,*

$$\{\, \mathbf{x^i} \,\} \to \mathbf{x}_* \Longrightarrow \{\, f(\mathbf{x^i}) \,\} \to f(\mathbf{x}_*)$$

*linearly, with same $R$ as the quadratic case, depending on $\lambda_1$ and $\lambda_n$, respectively the biggest and smallest eigenvalues of the Hessian (affecting how much elongated the Hessian is).*

This theorem means that if the function is differentiable and the Hessian is strictly positive definite then when getting closer and closer to the minimum, the function is more and more similar to a quadratic function.

This similarity is a good new, since we can use the same methods of the quadratic case, but, as usual, we must pay attention to **conditioning**.

In the next section we will provide algorithms for computing the local minimum of the univariate function $\varphi(\alpha)$.

### 4.3.1 Finding the best step size

We need to perform a one-dimensional (or line) search, i.e. finding the local minimum of the one dimensional function $\varphi^i$, s.t.

$$\varphi^i(\alpha) = f(\mathbf{x^i} + \alpha\mathbf{d^i}),$$

where $\mathbf{d^i} = -\nabla f(\mathbf{x^i})$.

Let us omit the index $i$, since we are concentrating on a single iteration. We are interested in finding $\alpha^* \in \mathbb{R}$ such that $\varphi'(\alpha^*) = 0$, but finding the roots of a function is very expensive.

**Example 4.3.1.** *Let us take* $f : \mathbb{R}^2 \to \mathbb{R}$ *s.t.* $f(x_1, x_2) = x_1^2 e^{x_2}$. *We can differentiate* $f$ *and obtain* $\nabla f(x_1, x_2) = (2x_1 e^{x_2}, x_1^2 e^{x_2})^T$.
*Let us suppose that at the* $i$*-th iteration* $\mathbf{x} = (0,1)^T$, *so* $\nabla f(1,0) = (2,1)$. *Now* $\mathbf{x}(\alpha) = \mathbf{x} - \alpha \nabla f(\mathbf{x}) = (1,0)^T - \alpha(2,1)^T = (1-2\alpha, 0-\alpha)^T$.
*At this point we obtain* $\varphi(\alpha) = f(\mathbf{x}(\alpha)) = (1-2\alpha)^2 e^{-\alpha}$.

In general, we are not interested in finding those points where $\varphi'(\alpha) = 0$, instead we want to force the directional derivative to be small (by picking a threshold $\varepsilon'$ s.t. $|\varphi'(\alpha)| \leq \varepsilon'$).

**Fact 4.3.2.** *Let* $\varphi : \mathbb{R} \to \mathbb{R}$, *such that* $\varphi(\alpha) = f(\mathbf{x^i} + \alpha \mathbf{d^i})$. *The first order derivative of* $\varphi$ *is computed as the scalar product between the gradient and the moving direction. Formally,* $\varphi'(\alpha) =< \nabla f(\mathbf{x^i} + \alpha \mathbf{d^i}), \mathbf{d^i} >$.

*Proof.*

$$\frac{\partial f(\mathbf{x^i} + \alpha \mathbf{d^i})}{\partial \alpha} = \frac{\partial f(\mathbf{x^i} + \alpha \mathbf{d^i})}{\partial (\mathbf{x^i} + \alpha \mathbf{d^i})} \cdot \frac{\partial (\mathbf{x^i} + \alpha \mathbf{d^i})}{\partial \alpha} = \nabla f(\mathbf{x^i} + \alpha \mathbf{d^i}) \cdot \mathbf{d^i}$$

□

When we design optimization algorithms we require as a parameter $\varepsilon$, but we do not ask for $\varepsilon'$. We are interested in computing $\varepsilon'$ ourselves from $\varepsilon$.

**Fact 4.3.3.** *We claim that* $\varepsilon' = \varepsilon$.

*Proof.* Let us check what happens if $\varepsilon' = \varepsilon$.

**Key idea:** Normalization of the direction.

We may normalize the direction of movement $\mathbf{d^i}$ without perturbing the behaviour of the algorithm: $\mathbf{d^i} = -\frac{\nabla f(\mathbf{x^i})}{\|\nabla f(\mathbf{x^i})\|}$. Notice that dividing by the norm of the gradient is safe, since if it gets 0 we have already stopped the procedure.

In this new context $\left\|\mathbf{d^i}\right\| = 1$ and

$$\begin{aligned} \left|\varphi'(\alpha^i)\right| &\overset{\text{Proposition 4.3.2}}{=} \left|< \nabla f(\mathbf{x^i} + \alpha \mathbf{d^i}), \mathbf{d^i} >\right| \\ &= \left|< \nabla f(\mathbf{x^{i+1}}), \mathbf{d^i} >\right| \\ &= \left|< \nabla f(\mathbf{x^{i+1}}), -\frac{\nabla f(\mathbf{x^i})}{\|\nabla f(\mathbf{x^i})\|} >\right| \end{aligned}$$

If the iterations converge to the optimum point (formally, $\{\mathbf{x^i}\} \to \mathbf{x}$)

$$\begin{aligned}\lim_{i\to\infty} \left|\varphi'(\alpha^i)\right| &= \lim_{i\to\infty} \left| < \nabla f(\mathbf{x^{i+1}}), \frac{\nabla f(\mathbf{x^i})}{\|\nabla f(\mathbf{x^i})\|} > \right| \\ &= \left| < \nabla f(\mathbf{x}), \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|} > \right| \\ &= \left| \frac{< \nabla f(\mathbf{x}), \nabla f(\mathbf{x}) >}{\|\nabla f(\mathbf{x})\|} \right| \\ &= \|\nabla f(\mathbf{x})\| \leq \varepsilon' = \varepsilon\end{aligned}$$

An attentive reader may notice that the thesis is based on the fact that $\mathbf{d^i}$ is normalized. If we do not normalize $\mathbf{d^i}$, we get that $\varepsilon' = \varepsilon \left\|\nabla f(\mathbf{x^i})\right\|$.

This is not exactly good news, because $\varepsilon'$ becomes smaller and smaller when we get closer to the optimum. This implies that the iterations become heavier at each step, because $\alpha$ is chosen depending on the accuracy threshold $\varepsilon'$. □

If we can prove that the algorithm is converging we know when to stop.
This convergence is not the perfect mathematical convergence, since $\varepsilon \neq 0$, because the line search will never terminate.

In the rest of this subsection we will propose three different kinds of algorithms for performing the line search on $\varphi$:

- First order algorithms;
- Second order algorithms;
- Zero order algorithms;
- Inexact line search;

where the first three approaches provide an exact solution, while the fourth one is less accurate but faster.

### First order algorithms

This class of algorithms makes use of the first derivative of the function $\varphi$ for finding the exact minimum of $\varphi$, namely we want to find the minimum points of $\varphi$, which corresponds to points where the first order derivative is zero and it goes from negative to positive.

We will present three different algorithms for this class, two of them are iterative, while the last one is a direct method. For the first two methods, we would like to reduce the range in which performing the search, at each step.

**Doubling:** How can we be sure that in a given range there is a point where the derivative is 0? Rolle's theorem, as shown in Figure 4.5.

Since the gradient is continuous the directional derivative is continue, so $\varphi$ is continuous (the scalar product is continuous).

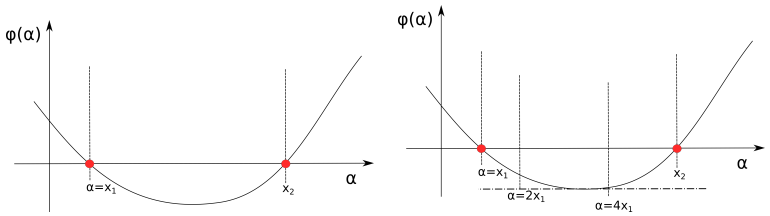



FIGURE 4.5: First, we restrict from $\mathbb{R}$ to $[x_1, x_2]$, then double $\alpha$ until the derivative is greater than 0

Actually, we only need to find where the derivative is positive, because the 0 of the derivative is between the previous value and this point. A formalization of the algorithm can be found in Algorithm 4.3.2.

ALGORITHM 4.3.2 First order method: **L**ine **S**earch bisection method

1: $\alpha \leftarrow x_1$; #or whatever value $> 0$
2: **while** $(\varphi'(\alpha) < 0)$ **do**
3: $\quad \alpha \leftarrow 2\alpha$; #or whatever factor $> 1$
4: **end while**

Since we are dealing with machine precision we stop when $\alpha < -10^{308}$, which is the smallest value for a double.

Works if $\varphi$ is **coercive**: $\lim_{\alpha \to \infty} \varphi(\alpha) = \infty$ (ex. $f$ strongly convex)

**Exercise 4.3.1.** *Build an example where $\bar{\alpha}$ exists but it is not found by this algorithm.*

**Solution:** The function changes its derivative in a range between $\alpha$ and $2\alpha$.

**Bisection:** we pick the middle point of the interval $[\alpha_-, \alpha_+]$, as shown in Algorithm 4.3.3.

ALGORITHM 4.3.3 First order method: Line Search Bisection Method

```
1: procedure LSBM((φ', α, ε))
2:     α_ ← 0;
3:     α+ ← α_;
4:     α ← α+;
5:     while (|φ'(α)| > ε) do
6:         α ← (α+ + α_)/2;
7:         if (φ'(α) < 0) then
8:             α_ ← α;
9:         else
10:            α+ ← α;
11:        end if
12:    end while
13: end procedure
```

**Quadratic approximation:** we use the first order information to build a quadratic model $m$ that approximates $\varphi$ in a ball centered in $\alpha$, as shown in Figure 4.6.

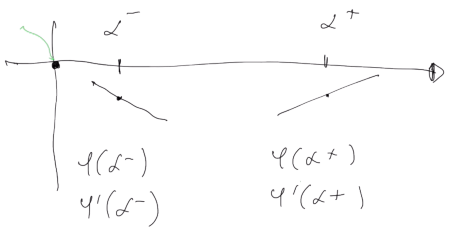



FIGURE 4.6: The information we have about function $\varphi$

We may use the information we have about the function, since we know $\varphi(\alpha_-)$, $\varphi'(\alpha_-)$, $\varphi(\alpha_+)$ and $\varphi'(\alpha_+)$.

At this point we can write a model ($m(\alpha) = a\alpha^2 + b\alpha + c$) and specialize it with the information we have, via computing a linear system:

$$\begin{cases} a(\alpha_-)^2 + b(\alpha_-) + c = \varphi(\alpha_-) \\ a(\alpha_+)^2 + b(\alpha_+) + c = \varphi(\alpha_+) \\ 2a\alpha_- + b = \varphi'(\alpha_-) \\ 2a\alpha_+ + b = \varphi'(\alpha_+) \end{cases}$$

that is solved by the closed formula (secant formula)

$$\alpha = \frac{\alpha - \varphi'(\alpha_+) - \alpha_+\varphi'(\alpha_)}{\varphi'(\alpha_+) - \varphi'(\alpha_-)}$$

This formula finds the intersection point between the segment linking $\alpha_-$ and $\alpha_+$ and the horizontal axis.

**Fact 4.3.4.** *Let $\varphi : \mathbb{R} \to \mathbb{R}$ such that $\varphi \in \mathcal{C}^3$, then quadratic interpolation has convergence of order $1 < p < 2$ (superlinear).*

In Figure 4.7 we can observe a situation in which the hypotheses of Proposition 4.3.4 are not satisfied.

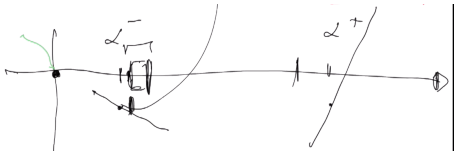


FIGURE 4.7: If the function is not $\mathcal{C}^3$ and one derivative is very big, then the range does not shrink much.

We would like to modify the formula to have at least linear convergence.

We can ensure to move not too close to one of the extremes, for example more than 10%.

It is possible to build a more complex model, i.e. using a cubic function. Although this is pretty involved, it allows the convergence to get quadratic ($p = 2$), which is better than super linear convergence of the quadratic interpolation.

**Do you recall?**

During last lecture we started dealing with algorithms for performing a line search for finding the minimum of $\varphi$, in particular we introduced first order algorithms.

### Second order algorithms

**Theorem 4.3.5.** *Given $f : \mathbb{R}^n \to \mathbb{R}$ such that $f \in \mathcal{C}^2$ and let $\varphi(\alpha) = \mathbf{x} + \alpha\mathbf{d}$, $\exists\varphi''(\alpha) = \mathbf{d}^T\nabla^2 f(\mathbf{x} + \alpha\mathbf{d})\mathbf{d}$ and it is continuous.*

*Proof.*

$$\begin{aligned}
\frac{\partial < \nabla f(\mathbf{x} + \alpha\mathbf{d}), \mathbf{d} >}{\partial\alpha} &= \frac{\partial < \mathbf{d}, \nabla f(\mathbf{x} + \alpha\mathbf{d}) >}{\partial\alpha} \\
&= \frac{\partial(\mathbf{d}^T \cdot \nabla f(\mathbf{x} + \alpha\mathbf{d}))}{\partial\alpha} \\
&= \mathbf{d}^T \cdot \left( \frac{\partial\nabla f(\mathbf{x} + \alpha\mathbf{d})}{\partial\mathbf{x} + \alpha\mathbf{d}} \cdot \frac{\partial(\mathbf{x} + \alpha\mathbf{d})}{\partial\alpha} \right) \\
&= \mathbf{d}^T\nabla^2 f(\mathbf{x} + \alpha\mathbf{d}) \cdot \mathbf{d}
\end{aligned}$$

□

Since we are looking for a point where the derivative $\varphi'(\alpha) = 0$, we may use the second order derivative to write a model and, assuming to trust the model, it can be studied.

**Definition 4.3.1** (Model−Newton's tangent method)**.** *Our model, in this case is the first order Newton's method applied to $\varphi'$, given a known point $\alpha^k$:*

$$\varphi'(\alpha) \approx \varphi'(\alpha^k) + \varphi''(\alpha^k)(\alpha - \alpha^k) = 0 \text{ iff } \alpha = \alpha^k - \varphi'(\alpha^k)/\varphi''(\alpha^k)$$

In this context, solving $\varphi'(\alpha) = 0$ implies finding those $\alpha$ such that $\alpha = \alpha^k - \varphi'(\alpha^k)/\varphi''(\alpha^k)$

Spiegazione: sono nel punto $\alpha^k$ e stimo $\varphi'(\alpha)$ come $\varphi'(\alpha^k)$ + $\varphi''(\alpha^k)$[coefficiente angolare] $\cdot (\alpha - \alpha^k)$

ALGORITHM 4.3.4 Second order method: Line Search with Newton's Method.

```
1: procedure LSNM(φ', φ'', α, ε)
2:     while (|φ'(α)| > ε) do
3:         α ← α − φ'(α)/φ''(α);
4:     end while
5: end procedure
```

We need to understand when and why $\varphi''(\alpha) \neq 0$ and when and why this method converges. The following theorem formalizes the fact that if we start from a point $\alpha^0$ which is close to the optimum we reach the optimum with quadratic speed.

**Theorem 4.3.6.** *Let $\varphi : \mathbb{R}^n \to \mathbb{R}$, $\varphi \in \mathcal{C}^3$ and take $\alpha_* \in \mathbb{R}$ such that $\varphi'(\alpha_*) = 0$ and $\varphi''(\alpha_*) \neq 0$. There is a neighborhood of the optimal solution such that the whole sequence of iterates converges quadratically, provided that the sequence starts in that neighborhood.*

*Formally, $\exists\ \delta > 0$ s.t. if $\alpha^0 \in [\alpha_* - \delta, \alpha_* + \delta]$ then $\{\alpha^k\} \to \alpha_*$, with $p = 2$.*

For the proof we need to reset the second form of the second order Taylor's model as defined in Definition 2.7.15

$$\forall \mathbf{x}' \in B(\mathbf{x}, \delta)\ \exists \alpha \in (0,1) \text{ s.t. } f(\mathbf{x}') = L_{\mathbf{x}}(\mathbf{x}') + \frac{1}{2}(\mathbf{x}' - \mathbf{x})^T \cdot \nabla^2 f(\alpha\mathbf{x} + (1-\alpha)\mathbf{x}')(\mathbf{x}' - \mathbf{x})$$

Equivalently,

$$\forall \mathbf{x}' \in B(\mathbf{x}, \delta)\ \exists \beta \in [\mathbf{x}, \mathbf{x}'] \text{ s.t } f(\mathbf{x}') = L_{\mathbf{x}}(\mathbf{x}') + \frac{1}{2}(\mathbf{x}' - \mathbf{x})^T \cdot \nabla^2 f(\beta)(\mathbf{x}' - \mathbf{x})$$

*Proof.* We are in the hypothesis that the function $\varphi$ is three times differentiable and we would like to prove that $\alpha^{k+1} - \alpha_* \to 0$

We want to compute how much the error is, if compared to the error at the previous iteration. Since (1)$\alpha^{k+1} := \alpha^k - \frac{\varphi'(\alpha^k)}{\varphi''(\alpha^k)}$ and (2)$\varphi'(\alpha_*) = 0$, we get

$$\begin{aligned}
\alpha^{k+1} - \alpha_* &\overset{(1)}{=} \alpha^k - \frac{(\varphi'(\alpha^k))}{\varphi''(\alpha^k)} - \alpha_* \\
&= \alpha^k - \alpha_* - \frac{(\varphi'(\alpha^k))}{\varphi''(\alpha^k)} \\
&\overset{(2)}{=} \alpha^k - \alpha_* - \frac{(\varphi'(\alpha^k) - \varphi'(\alpha_*))}{\varphi''(\alpha^k)} \\
&= \frac{-\varphi''(\alpha^k) \cdot (\alpha_* - \alpha_k) - \varphi'(\alpha^k) + \varphi'(\alpha_*)}{\varphi''(\alpha^k)}
\end{aligned}$$

Let us write the second form of the second order Taylor's model centered in $\alpha^k$ (Definition 2.7.15, Equation (2.7.3))

$$\exists\ \beta \in [\alpha^k, \alpha^*] \text{ s.t. } \varphi'(\alpha_*) = \varphi'(\alpha^k) + \varphi''(\alpha^k)(\alpha_* - \alpha^k) + \varphi'''(\beta)\frac{(\alpha_* - \alpha^k)^2}{2}$$

and plug it into the above equation:

$$\begin{aligned}
\alpha^{k+1} - \alpha_* &= \frac{-\varphi''(\alpha^k) \cdot (\alpha_* - \alpha_k) - \varphi'(\alpha^k) + \varphi'(\alpha_*)}{\varphi''(\alpha^k)} \\
&= \frac{\cancel{-\varphi''(\alpha^k) \cdot (\alpha_* - \alpha_k)} \cancel{- \varphi'(\alpha^k)} + \cancel{\varphi'(\alpha^k)} + \cancel{\varphi''(\alpha^k) \cdot (\alpha_* - \alpha_k)} + \frac{1}{2}\varphi'''(\beta) \cdot (\alpha_* - \alpha^k)^2}{\varphi''(\alpha^k)} \\
&= \frac{\frac{1}{2}\varphi'''(\beta) \cdot (\alpha_* - \alpha^k)^2}{\varphi''(\alpha^k)} \\
&= \frac{\varphi'''(\beta) \cdot (\alpha_* - \alpha^k)^2}{2 \cdot \varphi''(\alpha^k)}
\end{aligned}$$

We can say that the quantity $2\varphi''(\alpha^k)$ does not become too small and that the numerator $\varphi'''(\beta)$ does not become too big. This is proved since $\exists \delta > 0$ s.t. for $\alpha, \beta\ in [\alpha_* - \delta, \alpha_* + \delta]$:

- $\varphi''(\alpha) \geq k_2 > 0$, because $\varphi''(\alpha_*) \neq 0$ and $\varphi''$ is continuous
- $|\varphi'''(\beta)| \leq k_1 < \infty$, because it is computed in a finite interval, hence it cannot go to $\infty$.

We can go on bounding the difference between $\alpha^{k+1}$ and $\alpha_*$ as follows: for $\alpha$, $\beta \in [\,\alpha_* - \delta\,,\, \alpha_* + \delta]$

$$\begin{aligned} |\alpha^{k+1} - \alpha_*| &\overset{(3)}{=} \frac{\varphi'''(\beta)}{2\varphi''(\alpha^k)} \cdot (\alpha^k - \alpha_*)^2 \\ &\leq \frac{k_1}{2k_2}(\alpha^k - \alpha_*)^2 \end{aligned} \tag{4.3.1}$$

In general, $\frac{k_1}{2k_2}$ may be very large, but it is multiplied by $(\alpha^k - \alpha_*)^2$, which means that if we start close enough to $\alpha_*$ the upper-bound still makes sense.

$$|\alpha^{k+1} - \alpha_*| \leq \underbrace{[\frac{k_1}{2k_2}](\alpha^k - \alpha_*)}_{\leq 1} \cdot (\alpha^k - \alpha_*)$$

if $\frac{k_1}{2k_2} \cdot (\alpha^0 - \alpha_*) \leq 1$, it holds $\forall k > 1 \ \left|\alpha^{k+1} - \alpha_*\right| < \left|\alpha^k - \alpha_*\right|$ and this proves the theorem.

□

This approach is pretty accurate, but it is usually undesirable to devote substantial resources to finding a value of $\alpha$ to precisely minimize $f$. This is because the computing resources needed to find a more precise minimum along one particular direction could instead be employed to identify a better search direction.
The following class of algorithms circumvent the problem of the existence of derivatives, but in general allow to find a solution without computing derivatives at all.

**Mantra**

The more derivatives we have, the smallest number of points we need (second derivative → two points, third derivative → zero points). The opposite holds as well.

### Zero order algorithms

**Definition 4.3.2** (Unimodal function)**.** *Let $f : \mathbb{R} \to \mathbb{R}$. We say that $f$ is **unimodal** if for some value $m \in \mathbb{R}$ it is monotonically increasing for each $x \leq m$ and monotonically decreasing for any $x \geq m$ or vice-versa. Notice that any unimodal function admits* only one *maximum (minimum) value $f(m)$.*

Let us be given $\varphi$, we are looking for $\alpha_*$ within an error tolerance $\varepsilon$. Notice that if the function $\varphi$ is not unimodal we have no guarantee that the interval that we are discarding does not contain the "deepest" minimum. We know only the function values in given points and we would like to shrink a candidate interval in the best way possible.
We propose an elegant solution via golden ratio.

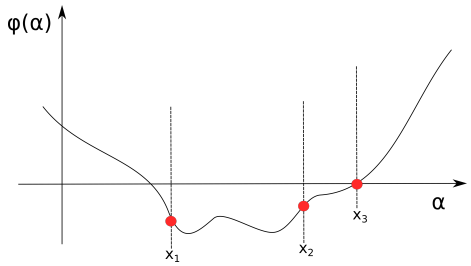



FIGURE 4.8: The candidate interval is $[x_1, x_2]$, since $\varphi(x_2) > \varphi(x_1)$ and we are allowed to exclude the interval $[x_3, +\infty)$ since the value in $x_3$ is bigger than $\varphi(x_2)$.

**Definition 4.3.3.** *In mathematics, two quantities are in the* ***golden ratio*** *if their ratio is the same as the ratio of their sum to the larger of the two quantities. Formally,* $\forall a, b \in \mathbb{R}$ *such that* $a > b > 0$,

$$a + b : a = a : b$$

*and we denote* $r_g$ *such ratio.*

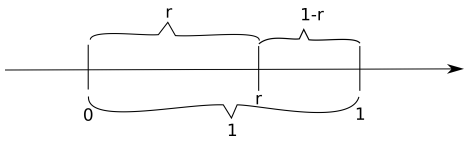



FIGURE 4.9: The relationship between $r$ and $1 - r$ is $r \ : \ 1 = (1 - r) \ : \ r$.

Let us start from an interval $[\alpha_-, \alpha_+] \subseteq \mathbb{R}$, and let us take $r$ as the inverse of the golden ratio $r = 1/\frac{\sqrt{5}-1}{2} \approx 0.618$.

ALGORITHM 4.3.5 Zero-th order method: Line Search Golden Ration Method.

**procedure LSGRM**$(\varphi, \alpha, \varepsilon)$
  $\alpha_- \leftarrow 0$;
  $\alpha_+ \leftarrow \alpha$;
  $\alpha'_- \leftarrow (1-r)\alpha$;
  $\alpha'_+ \leftarrow r\alpha$;
  **while** $(\alpha_+ - \alpha_- > \varepsilon)$ **do** ▷ not the same $\varepsilon$
    **if** $\varphi(v'_l) > \varphi(v'_r)$ **then**
      $\alpha_- \leftarrow \alpha'_-$;
      $\alpha'_- \leftarrow \alpha \leftarrow \alpha'_+$;
      $\alpha'_+ \leftarrow \alpha_+(1-r)(\alpha_+ - \alpha_-)$;
    **else**
      $\alpha_+ \leftarrow \alpha'_+$;
      $\alpha'_+ \leftarrow \alpha \leftarrow \alpha'_-$;
      $\alpha'_- \leftarrow \alpha_- + r(\alpha_+ - \alpha_-)$;
    **end if**
  **end while**
**end procedure**

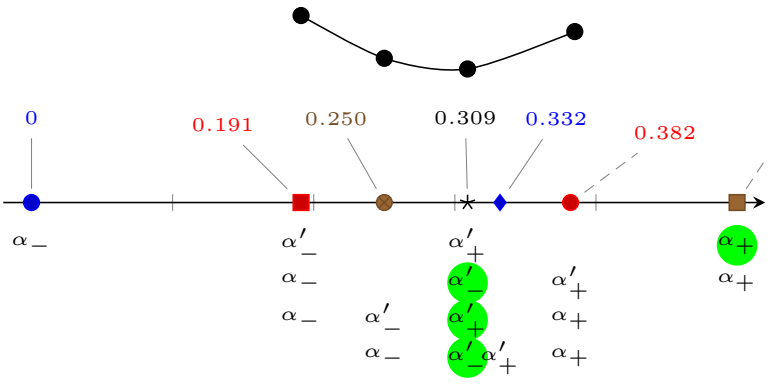



FIGURE 4.10: Plot of iterates of zero order algorithm for line search. The lines below the $x$ axis represent the different values of $\alpha_-, \alpha_+, \alpha'_-, \alpha'_+$ across different iterations, while the greed dot marks how the value $\alpha$ changes.

**Example 4.3.2.** *Let us try Algorithm 4.3.5 on a toy example, displayed in Figure 4.10. Let* $\alpha = 0$ *and let* $\varepsilon = 0.1$.

$$\alpha_- = 0$$
$$\alpha_+ = 0.5$$
$$\alpha'_- = r\alpha = (1 - 0.618) \cdot 0.5 = 0.191$$
$$\alpha'_+ = r \cdot \alpha = 0.618 \cdot 0.5 = 0.309$$

*Since* $0.5 - 0 < 0.1$ *we enter the loop. According to the values of function* $\varphi$ *in* $\alpha_-$ *and* $\alpha_+$ *we enter the* if *branch:*

$$\alpha_- = \alpha'_- = 0.191$$
$$\alpha'_- = \alpha'_+ = 0.309$$
$$\alpha = \alpha'_+ = 0.309$$
$$\alpha'_+ = \alpha_- + r \cdot (\alpha_+ - \alpha_-) = 0.191 + 0.618 \cdot (0.5 - 0.191) = 0.382$$

*Provided that* $0.5 - 0.191 < 0.1$ *we enter the loop. According to the values of function* $\varphi$ *in* $\alpha_-$ *and* $\alpha_+$ *we enter the* else *branch:*

$$\alpha_+ = \alpha'_+ = 0.382$$
$$\alpha'_+ = \alpha'_- = 0.309$$
$$\alpha = \alpha'_- = 0.309$$
$$\alpha'_- = \alpha_- + (1 - r) \cdot (\alpha_+ - \alpha_-) = 0.191 + 0.309 \cdot (0.382 - 0.191) = 0.25$$

*Provided that* $0.382 - 0.191 < 0.1$ *we enter the loop. According to the values of function* $\varphi$ *in* $\alpha_-$ *and* $\alpha_+$ *we enter the* if *branch:*

$$\alpha_- = \alpha'_- = 0.25$$
$$\alpha'_- = \alpha'_+ = 0.309$$
$$\alpha = \alpha'_+ = 0.309$$
$$\alpha'_+ = \alpha_- + r \cdot (\alpha_+ - \alpha_-) = 0.25 + 0.618 \cdot (0.382 - 0.25) = 0.332$$

### Inexact line search

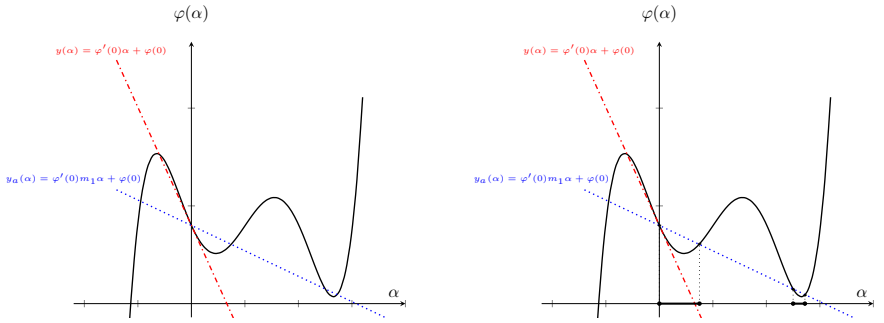
In the rest of this lecture we will present a line search method to determine the maximum amount of movement to perform along a given search direction, starting with a relatively large estimate of the step size for movement along the search direction, and iteratively shrinking the step size ("backtracking") until a decrease of the objective function is observed that adequately corresponds to the decrease that is expected, based on the local gradient of the objective function.

**Definition 4.3.4** (Armijo's condition)**.** *Let* $\varphi : \mathbb{R}^+ \cup \{0\} \to \mathbb{R}$ *such that* $\varphi(\alpha) = f(\mathbf{x} + \alpha\mathbf{d})$*. The* ***Armijo's condition*** *selects those real values* $\alpha$ *such that the function value in those points* $(\varphi(\alpha))$ *is smaller than the value of a line which passes through* $\varphi(0)$ *and is less steep than the tangent to the curve in such point.*

*Formally,* $\alpha \in \mathbb{R}$ *satisfies* ***Armijo's condition*** *if for some* control parameter $0 < m_1 < (\ll)1$ *the following holds*

$$\varphi(\alpha) \leq y_a(\alpha) = \varphi(0) + m_1 \alpha \varphi'(0) \ (A)$$

*where we call* $y_a(\alpha)$ *the* ***Armijo's line****.*




(a) Armijo's condition leads to a line which is less steep than the tangent in $\alpha = 0$.

(b) Armijo's condition selects those $\alpha \in \mathbb{R}$ such that $\varphi(\alpha) < y_a(\alpha)$.

FIGURE 4.11: Graphical hint of the functioning of Armijo's condition.

Intuitively, provided that we are minimizing $\varphi$, $\varphi'(0) < 0$. This means that the slope of the line $y_a$ $(m_1\varphi'(0))$ is smaller in absolute value (hence bigger) than the slope of the tangent to the curve $\varphi$ in 0, therefore the line $y_a$ is less steep then $y$.

An attentive reader may notice that $\alpha \approx 0$ lead to values of $\varphi(\alpha) < y(\alpha)$, hence smaller than $y_a(\alpha)$, but this would lead to a very slow convergence.
In order to overcome this issue one can decide to start from a large $\alpha$ and reduce it until it satisfies Armijo's condition (as shown in Algorithm 4.3.6) or introduce another algebraic constraint:

ALGORITHM 4.3.6 Inexact method: **B**acktracking **L**ine **S**earch.

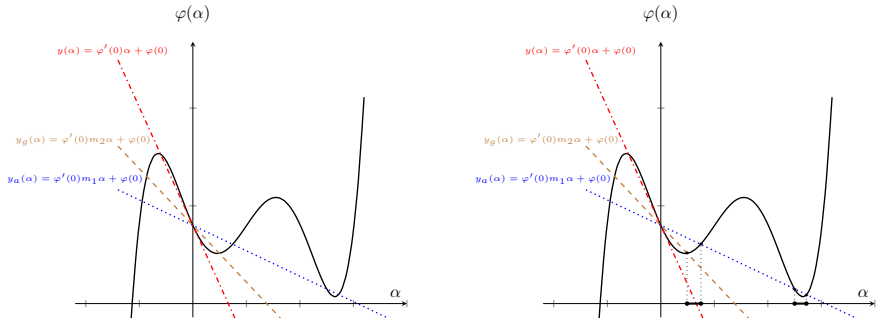
```
1: procedure BLS(φ, φ', α, m1, τ)
2:     while (φ(α) > φ(0) + m1 α φ'(0)) do
3:         α ← τα;                          ▷ With τ < 1
4:     end while
5: end procedure
```

**Definition 4.3.5** (Goldstein's condition)**.** *Let* $\varphi : \mathbb{R}^+ \cup \{0\} \to \mathbb{R}$ *such that* $\varphi(\alpha) = f(\mathbf{x} + \alpha\mathbf{d})$ *and let* $m_1 \in \mathbb{R}$ *be the Armijo's constant. The* ***Goldstein's condition*** *selects those real values* $\alpha$ *that satisfy Armijo's condition but also such that the function value in those points* $(\varphi(\alpha))$ *is* larger *than the value of a*

*line which passes through* 0 *and is less steep than the tangent to the curve in such point.*

*Formally,* $\alpha \in \mathbb{R}$ *satisfies the* ***Goldstein's condition*** *if for come* control parameter $m_1 < m_2 < 1$ *the following holds*

$$\varphi(\alpha) \geq y_g(\alpha) = \varphi(0) + m_2\alpha\varphi'(0) \ (G)$$




(a) Goldstein's condition leads to a line which is less steep than the tangent in $\alpha = 0$ but steeper than Armijo's line.

(b) Armijo's and Goldstein's conditions select those $\alpha \in \mathbb{R}$ such that $\varphi(\alpha) < y_a(\alpha)$ and $\varphi(\alpha) > y_g(\alpha)$.

FIGURE 4.12: Conjunction of Armijo's and Goldstein's conditions.

It may happen that the points that satisfy both Goldstein's and Armijo's conditions do not contain a local minimum.
To circumvent this problem another condition comes to help us.

**Definition 4.3.6** (Wolfe's condition)**.** *Let* $\varphi : \mathbb{R}^+ \cup \{0\} \to \mathbb{R}$ *such that* $\varphi(\alpha) = f(\mathbf{x} + \alpha\mathbf{d})$ *and let* $m_1 \in \mathbb{R}$ *be the Armijo's constant. The* ***Wolfe's condition*** *selects those real values* $\alpha$ *that allow for a decrease in the slope (*curvature condition*), following the reasoning that close to the optimum the tangent line is almost horizontal.*

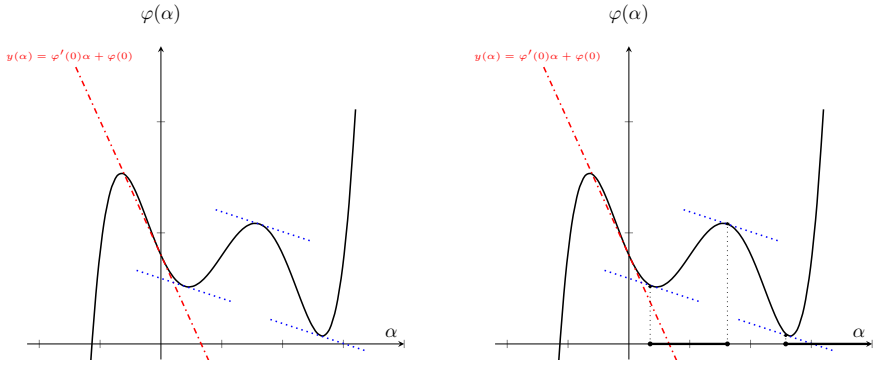
*Formally, let* $m_1 < m_3 < 1$

$$\varphi'(\alpha) \geq m_3\varphi'(0) \ (W)$$

Figure 4.13(b) shows the intervals selected by Wolfe's rule. Provided that we are moving along a direction of negative curvature for $f$, we have that $\varphi'(\alpha) < 0$, therefore Wolfe's condition is satisfied also by those points $\alpha$ where the derivative is positive.

Sometimes, the curvature condition is modified to force the step length to stay in at least a broad neighborhood of a local minimizer or stationary point of the univariate function $\varphi$.

Provided that $\varphi'(0) < 0$, $|\varphi'(0)| = -\varphi'(0)$; follows a stronger variant of Wolfe's condition.




(a) Wolfe's condition fixes a threshold for the slope of the line tangent to the curve $\varphi$.

(b) Wolfe's rule selects those points where $\varphi'(\alpha) > 0^-$.

FIGURE 4.13: Wolfe's condition.

**Definition 4.3.7** (Strong Wolfe's condition)**.** *Let $\varphi : \mathbb{R}^+ \cup \{0\} \to \mathbb{R}$ such that $\varphi(\alpha) = f(\mathbf{x} + \alpha\mathbf{d})$ and let $m_1 \in \mathbb{R}$ be the Armijo's constant. The **strong Wolfe's condition** selects those real values $\alpha$ that allow for a decrease in the slope but prevent it to become big in absolute value. Formally, let $m_1 < m_3 < 1$*

$$|\varphi'(\alpha)| \leq m_3|\varphi'(0)| = -m_3\varphi'(0) \ (W')$$

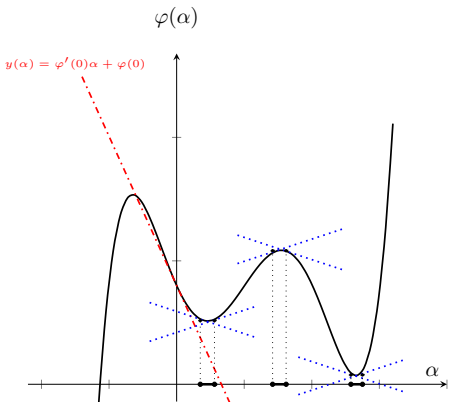



FIGURE 4.14: Strong Wolfe's condition.

**Fact 4.3.7.** *Let $\varphi : \mathbb{R}^+ \cup \{0\} \to \mathbb{R}$ such that $\varphi(\alpha) = f(\mathbf{x} + \alpha\mathbf{d})$. If $\varphi'(\alpha) \not\gg 0$ and both Armijo's and Wolfe's (or Armijo's and strong Wolfe's) conditions hold then all local minima (maxima) are captured unless $m_1$ is too close to 1.*

For a graphical hint of Proposition 4.3.7, see Figure 4.15, where we can observe that if $m_1 \approx 1$, the line is pretty steep, therefore the intersection between Armijo's and Wolfe's regions is very small. In practice it is common to choose $m_1 \approx 0.0001$.

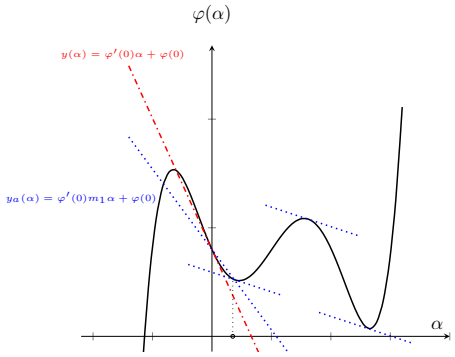



FIGURE 4.15: $m_1 \approx 1$ and the intersection between Armijo's and Wolfe's ranges.

The $m_i$ are like the hyperparameters of machine learning. Less formally, if we choose an $m_1$ far enough from 1 everything works fine.

**Theorem 4.3.8.** *Let $\varphi \in \mathcal{C}^1$ and $\varphi(\alpha)$ bounded below for $\alpha \geq 0$ then $\exists\ \alpha$ s.t. (A) $\cap$ (W') holds.*

*Proof.* Let us write Armijo's line: $y_a(\alpha) = \varphi(0) + m_1\varphi'(0)\alpha$. Let us define $d(\alpha) = y_a(\alpha) - \varphi(\alpha) = \varphi(0) + m_1\varphi'(0)\alpha - \varphi(\alpha)$, such that $d'(\alpha) = y_a'(\alpha) - \varphi'(\alpha) = m_1\varphi'(0) - \varphi'(\alpha)$. It goes without saying that $d(0) = 0$ and $d'(0) = m_1\varphi'(0) - \varphi'(0) = \underbrace{(m_1 - 1)}_{<1}\underbrace{\varphi'(0)}_{<1} > 0$.

In $\alpha = 0$ the derivative of $\varphi$ is negative, hence the function $\varphi$ is decreasing. From the hypothesis we know that $\varphi$ is bounded below, hence at a certain point it will start increasing its value until it will intersect with Armijo's line eventually. Let us call $\bar{\alpha} > 0$ the smallest value such that $d(\bar{\alpha}) = 0$. All the points $\alpha \in ]0, \bar{\alpha}[$ satisfy Armijo's condition.
We are left with the task of proving that also strong Wolfe's condition holds for those $\alpha \in ]0, \bar{\alpha}[$. 0 and $\bar{\alpha}$ are the two roots of the function $d$, so we can use Rolle's theorem, in order to prove that the function $d$ has a stationary point in the interval $]0, \bar{\alpha}[$. Let us call the stationary point $\alpha^* \in ]0, \bar{\alpha}[$, $d'(\alpha^*) = 0$ iif $\varphi'(\alpha^*) = -m_1\varphi'(0)$, therefore $|\varphi'(\alpha^*)| = m_1 \cdot |\varphi'(0)|$ and this proves Wolfe's rule, because $m_1 < m_3 < 1$, hence $|\varphi'(\alpha^*)| \leq m_3|\varphi'(0)|$. □

How can we find such a point?

**Fact 4.3.9.** *Let $\varphi : \mathbb{R}^+ \cup \{0\} \to \mathbb{R}$ such that $\varphi(\alpha) = f(\mathbf{x} + \alpha\mathbf{d})$ and let $\nabla f$ be L-Lipschitz. Then $\varphi'$ is L-Lipschitz as well and L does not depend on the iterate point $\mathbf{x}$.*

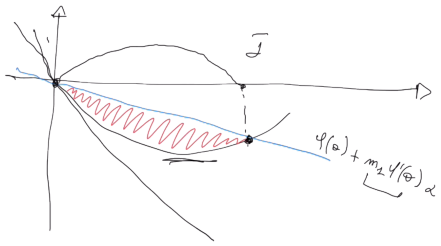


FIGURE 4.16: If $\varphi$ is not going to $-\infty$ the Armijo's line (blue) and the function will meet in $\bar{\alpha} > 0$.

*Proof.*

$$\begin{aligned}
|\varphi'(\alpha_1) - \varphi'(\alpha_2)| &= \left|\mathbf{d}^T \nabla f(\mathbf{x} + \alpha_1 \mathbf{d}) - \mathbf{d}^T \nabla f(\mathbf{x} + \alpha_2 \mathbf{d})\right| \\
&= \mathbf{d}^T \underbrace{|\nabla f(\mathbf{x} + \alpha_1 \mathbf{d}) - \nabla f(\mathbf{x} + \alpha_2 \mathbf{d})|}_{\leq L \cdot |\mathbf{x} + \alpha_1 \mathbf{d} - (\mathbf{x} + \alpha_2 \mathbf{d})|} \\
&\leq \mathbf{d}^T L \cdot |\cancel{\mathbf{x}} + \alpha_1 \mathbf{d} - (\cancel{\mathbf{x}} + \alpha_2 \mathbf{d})| \\
&= \mathbf{d}^T L |\alpha_1 - \alpha_2| \mathbf{d} \\
&\overset{*}{=} L \cdot |\alpha_1 - \alpha_2|
\end{aligned}$$

where $\overset{*}{=}$ follows from the fact that $\mathbf{d}$ has norm 1. □

**Fact 4.3.10.** *Let $\varphi : \mathbb{R}^+ \cup \{0\} \to \mathbb{R}$ such that $\varphi(\alpha) = f(\mathbf{x} + \alpha \mathbf{d})$, $\varphi$ bounded below and $\nabla f$ L-Lipschitz. Let $\bar{\alpha} \in \mathbb{R}$ s.t. $\varphi(\alpha) = \varphi(0) + m_1 \varphi'(0)\bar{\alpha}$, then $\bar{\alpha}$ has the same behaviour of $\left\|\nabla f(\mathbf{x^i})\right\|$. Formally,*

$$\bar{\alpha} > (1 - m_1) \frac{\left\|\nabla f(\mathbf{x^i})\right\|}{L}$$

*Proof.* According to Proposition 4.3.9, $\varphi'$ is $L$-Lipschitz hence

$$L(\bar{\alpha} - 0) \geq \varphi'(\bar{\alpha}) - \varphi'(0)$$

Thanks to the choice of $\bar{\alpha}$ we have that $\varphi'(\bar{\alpha})$, therefore

$$\varphi'(\bar{\alpha}) - \varphi'(0) > (1 - m_1)(-\varphi'(0)) \overset{*}{=} (1 - m_1)(\left\|\nabla f(\mathbf{x^i})\right\|)$$

where $\overset{*}{=}$ follows from

$$\varphi'(\alpha) = < \nabla f(\mathbf{x} + \alpha \mathbf{d}), \overbrace{\mathbf{d}}^{-\nabla f(\mathbf{x} + \alpha \mathbf{d}) / \|\nabla f(\mathbf{x} + \alpha \mathbf{d})\|} > = \frac{\|\nabla f(\mathbf{x} + \alpha \mathbf{d})\|^2}{\|\nabla f(\mathbf{x} + \alpha \mathbf{d})\|}$$

therefore we have the thesis $\bar{\alpha} > (1 - m_1) \frac{\left\|\nabla f(\mathbf{x^i})\right\|}{L}$. □

Now we can prove the following

**Theorem 4.3.11.** *If (A) ∩ (W) holds ∀i then either* $\{f(\mathbf{x^i})\} \to -\infty$ *or* $\{\|\nabla f(\mathbf{x^i})\|\} \to 0$.

*Proof.* **By contradiction**. Let us assume that $\|\nabla f(\mathbf{x^i})\|$ diverges. Formally, $-\varphi'(0) = \|\nabla f(\mathbf{x^i})\| \geq \varepsilon > 0 \ \forall i$. Then

**Wolfe.** Wolfe's condition ensures that the step $\bar{\alpha}$ does not go to 0. Formally,
$\alpha^i \geq \bar{\alpha} > (1-m_1)\frac{\|\nabla f(\mathbf{x^i})\|}{L} \geq (1-m_1)\frac{\varepsilon}{L}$.
Let $\delta := \frac{(1-m_1)\cdot\varepsilon}{L}$, then $\alpha^i \geq \delta > 0$;

**Armijo.** Armijo's condition ensures that the function value decreases at each iteration. Formally, $f(\mathbf{x^{i+1}}) \leq f(\mathbf{x^i}) - m_1\alpha^i \|\nabla f(\mathbf{x^i})\| \leq f(\mathbf{x^i}) - \underbrace{m_1\delta\varepsilon}_{\neq 0}$,
because both $\delta, \varepsilon > 0$;

Therefore, if the norm of the gradient is not 0, $f$ decreases at each iteration, it will eventually go to $-\infty$. □

**Fact 4.3.12.** *If (A) ∩ (W) holds ∀i then Backtracking Line Search algorithm converges.*

*Proof.* **By contradiction**. Let us assume that $\|\nabla f(\mathbf{x^i})\|$ diverges. Formally, $\|\nabla f(\mathbf{x^i})\| \geq \varepsilon > 0 \ \forall i$. Thanks to Theorem 4.3.11, we have $\bar{\alpha} > \delta > 0 \forall i$.

Let us suppose to start the BLS procedure with a stepsize $\alpha^0 = 1$, the generic step of BLS algorithm is $\alpha \leftarrow \tau\alpha$. We denote with $h \in \mathbb{N}$ the smallest integer such that $\alpha^h = \tau^h \leq \delta$. Formally, $h = \min\{k : \tau^k \leq \delta\}$. It holds that $\forall i \ \alpha^i \geq \tau^{-h} > 0$ As proved by Theorem 4.3.11 $f(\mathbf{x^{i+1}}) \leq f(\mathbf{x^i}) - m_1\tau^{-h}\varepsilon$ then the function $f$ eventually goes to $-\infty$. □

## 4.4 Good Practices for the Design of the Project

Follow some good suggestions for implementing the gradient method in Matlab:

- When we have to implement a function the first question we need to ask ourselves is: "how many parameters should the function take?";
- The interface needs to be designed, i.e. in the case of the gradient method we would like the user to be able to run it although he might not know what a good starting point could be;
- There might be some parameters with the same semantic of hyper-parameters, so they need to be adjusted in order to specify the algorithm;
- Another thing that should be taken into consideration is that sometimes, when measuring the number of times the code of the function has been executed we need to count also the calls to functions inside a single step;

- It is important to check the conditions that should be satisfied by the input data for the theoretical coherence. In case of not valid input an error message should be returned;
- Another important thing to be done is building a set of test functions, possibly edge cases for the problem, to see how the algorithm works;
- `diff(f, x)` calculates the gradient of the function $f$ in the variable $x$. If we want to get a simplified version, we may run `simplify(diff(f,x))`;
- A very nice tool to compute derivatives for arbitrary functions is **Adigator**[†].

## 4.5 General Descent Methods

**Do you recall?**

So far, we chose as direction for the step the direction where the decrease of the function is maximum ($\mathbf{d^i} = -\nabla f(\mathbf{x^i})$) and for that descent direction we applied a line search for finding the optimal step size $\alpha$. This was possible thanks to the **convergence argument** which says that in proximity of the stationary point of the linear model the norm of the gradient goes to 0. Formally, $\|\nabla f(\mathbf{x})\| \to 0$ when $\varphi'(0) \to 0$, where $\varphi'(\alpha) =< \nabla f(\mathbf{x^i} + \alpha^i \mathbf{d^i}), \mathbf{d^i} >$.

In this lecture we will propose a different choice for the moving direction $\mathbf{d^i}$, that keeps satisfying the convergence argument.

For example, let us take as direction a rotation of the opposite of the gradient, the value of $\varphi'$ is then the cosine of the angle of the rotation times the opposite of the norm of the gradient. We have an infinite number of angles that we can choose, so we have a lot of flexibility.

Let us define the angle $\theta^i$ between the search direction $\mathbf{d^i}$ and the steepest descent direction $-\nabla f(\mathbf{x^i})$ as

$$\cos\theta^i = \frac{< \nabla f(\mathbf{x}), \mathbf{d^i} >}{\|\nabla f(\mathbf{x^i})\| \cdot \|\mathbf{d^i}\|}$$

Note that the angle between $\mathbf{d^i}$ and the gradient should not be too close to 90°, otherwise the cosine would get approximately 0..

**Theorem 4.5.1** (Zoutendijk's theorem)**.** *Let us consider minimum problem (P) where the objective function $f \in \mathcal{C}^1$, $\nabla f$ is L-Lipschitz and $f$ bounded below. Let*

[†]https://sourceforge.net/projects/adigator

*us consider an iteration* $\mathbf{x^{i+1}} = \mathbf{x^i} + \alpha^i \mathbf{d^i}$*, where* $\mathbf{d^i}$ *is a descent direction and* $\alpha^i \in \mathbb{R}$ *satisfies both Armijo's and Wolfe's conditions. Then*

$$\sum_{i=1}^{\infty} \cos^2(\theta^i) \left\| \nabla f(\mathbf{x^i}) \right\|^2 < \infty$$

**Do you recall?**

A positive infinite sequence, whose corresponding series converges to a number, converges to 0 reasonably fast.

Therefore, if we choose an angle $\theta^i$ which cosine is bounded below then the norm of the gradient goes to 0 quickly. More formally,

**Fact 4.5.2.** *Under the same hypotheses of Theorem 4.5.1, given* $\theta^i$ *angle between* $\mathbf{d^i}$ *and the gradient such that* $\cos(\theta^i) \geq \varepsilon > 0$ *then* $\left\| \nabla f(\mathbf{x^i}) \right\| \to 0$*.*

*Proof.* As stated in the recap, the following holds:

$$\cos^2(\theta^i) \left\| \nabla f(\mathbf{x^i}) \right\|^2 \to 0$$

therefore, one between $\cos(\theta^i)$ and $\left\| \nabla f(\mathbf{x^i}) \right\|^2$ should converge to zero, but it is no the cosine, because it is bounded below by $\varepsilon > 0$. □

We found a converging algorithm that allows much more freedom by allowing infinite choices for the descent direction.

## 4.6 Newton's Method

**Mantra**

If you want better direction, use a better model.

Newton's method attempts to solve the minimum problem (P) by constructing a sequence $\{\mathbf{x^i}\}$ from an initial guess (starting point) $\mathbf{x^0} \in \mathbb{R}^n$ that converges towards a minimizer $\mathbf{x}_*$ of $f$ by using a sequence of second-order Taylor approximations of $f$ around the iterates.

Newton's method is the father of all algorithms that pick a direction which is different from the opposite of the gradient. So far, we used a linear model to approximate the objective function in a ball around the current point. Using a linear model we did not have any information on the curvature.

### 4.6.1 Convex case

In the next line we will explore Newton's method in the simplest case, when the function is perfectly convex (i.e. the Hessian is strictly positive definite).

**Theorem 4.6.1.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function such that $\nabla^2 f(\mathbf{x^i}) \succ 0$. Then the second order model in proximity of point $\mathbf{x}$ ($Q_{\mathbf{x^i}}(\mathbf{y})$ for $\mathbf{y} \in \mathcal{B}(\mathbf{x}, \delta)$) admits a minimum.*

*Proof.* Let us write $\mathbf{x}$ instead of $\mathbf{x^i}$ to ease notation. The second order Taylor's expansion of $f$ around the point $\mathbf{x}$ (recall Definition 2.7.15) is:

$$\begin{aligned} Q_{\mathbf{x}}(\mathbf{y}) &= f(\mathbf{x}) + < \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} > + \frac{1}{2}(\mathbf{y} - \mathbf{x})^T \nabla^2 f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \\ &= f(\mathbf{x}) + < \nabla f(\mathbf{x}), \mathbf{y} > - < \nabla f(\mathbf{x}), \mathbf{x} > + \\ &+ \frac{1}{2}\left(\mathbf{y}^T \nabla^2 f(\mathbf{x})\mathbf{y} - 2\mathbf{x}^T \nabla^2 f(\mathbf{x})\mathbf{y} + \mathbf{x}^T \nabla^2 f(\mathbf{x})\mathbf{x}\right) \\ &= f(\mathbf{x}) + < \nabla f(\mathbf{x}), \mathbf{y} > - < \nabla f(\mathbf{x}), \mathbf{x} > + \\ &+ \frac{1}{2}\mathbf{y}^T \nabla^2 f(\mathbf{x})\mathbf{y} - \mathbf{x}^T \nabla^2 f(\mathbf{x})\mathbf{y} + \frac{1}{2}\mathbf{x}^T \nabla^2 f(\mathbf{x})\mathbf{x} \end{aligned}$$

We are looking for the minimum of $Q_{\mathbf{x}}$, that it is equivalent to searching for a stationary point. Let us differentiate $Q_{\mathbf{x}}$:

$$\begin{aligned} \frac{\partial Q_{\mathbf{x}}(\mathbf{y})}{\partial \mathbf{y}} &\overset{*}{=} \frac{\partial(< \nabla f(\mathbf{x}), \mathbf{y} > + \frac{1}{2}\mathbf{y}^T \nabla^2 f(\mathbf{x})\mathbf{y} - \mathbf{x}^T \nabla^2 f(\mathbf{x})\mathbf{y}}{\partial \mathbf{y}} \\ &= \frac{\partial(< \nabla f(\mathbf{x}) + \frac{1}{2}\mathbf{y}^T \nabla^2 f(\mathbf{x}) - \mathbf{x}^T \nabla^2 f(\mathbf{x}), \mathbf{y} >}{\partial \mathbf{y}} \\ &= \nabla f(\mathbf{x}) + \frac{1}{2}\nabla^2 f(\mathbf{x})\mathbf{y} - \mathbf{x}^T \nabla^2 f(\mathbf{x}) \end{aligned}$$

where $\overset{*}{=}$ is given by the fact that $f(\mathbf{x})$, $- < \nabla f(\mathbf{x}), \mathbf{x} >$ and $\frac{1}{2}\mathbf{x}^T \nabla^2 f(\mathbf{x})\mathbf{x}$ are constant terms with respect to $\mathbf{y}$. The equality obtained above allows to say that $\mathbf{y}$ is a stationary point iff

$$\begin{aligned} \frac{1}{2}\nabla^2 f(\mathbf{x})\mathbf{y} &= \mathbf{x}^T \nabla^2 f(\mathbf{x}) - \nabla f(\mathbf{x}) \\ &\overset{(1)}{=} \nabla^2 f(\mathbf{x})\mathbf{x} - \nabla f(\mathbf{x}) \\ \mathbf{y} &= 2 \cdot \left(\mathbf{x} - [\nabla^2 f(\mathbf{x})]^{-1} \nabla f(\mathbf{x})\right) \end{aligned}$$

Where $\overset{(1)}{=}$ follows from Corollary 2.7.13 ($f \in \mathcal{C}^2 \Rightarrow \nabla^2 f \in S(n, \mathbb{R})$). □

The moving direction of the gradient descent algorithm using Newton's method is obtained taking the opposite of the inverse of the Hessian times the gradient of the function. Formally, $\mathbf{d^i} = -[\nabla^2 f(\mathbf{x^i})]^{-1} \nabla f(\mathbf{x^i})$.

In Newton's method $\alpha^i = 1$ will always work.

An attentive reader may notice that Theorem 4.6.1 holds only if the Hessian is invertible.

Moreover, even if the Hessian is invertible, the computation of the descent direction requires to solve a linear system. A way to circumvent this problem is putting the gradient to 0, so we can write the Taylor form of the gradient and solve a linear equation which is $\nabla f(\mathbf{x}) \approx \nabla f(\mathbf{x^i}) + \nabla^2 f(\mathbf{x^i})(\mathbf{x} - \mathbf{x^i})$.

**Theorem 4.6.2.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ s.t. $f \in \mathcal{C}^3$ and let $\mathbf{x}_*$ be a saddle point such that $\nabla f(\mathbf{x}_*) = 0$ and $\nabla^2 f(\mathbf{x}_*) \succ 0$. Then $\exists\, \mathcal{B}(\mathbf{x}_*, r)$ s.t. $\mathbf{x^1} \in \mathcal{B}$ and $\{\mathbf{x^i}\} \to \mathbf{x}_*$ quadratically.*

Notice that Theorem 4.6.2 states that the descent direction computed through Newton's method is correct both close and far from the minimum point.

An attentive reader may notice that the scalar product between the function value at the $i$-th iterate $f(\mathbf{x^i})$ and the descent direction $\mathbf{d^i}$ should be negative but also not too close to 0. Such condition is ensured when the function $f$ is such that $uI \preceq \nabla^2 f \preceq LI$, which implies that the function is strongly convex, in other words that the eigenvalues of the Hessian do not get too close to zero (in terms of machine precision).

**Theorem 4.6.3** (Global convergence of Newton's method)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ s.t. $f \in \mathcal{C}^3$ and such that satisfies $uI \preceq \nabla^2 f \preceq LI$ and $\cos\theta^i = \frac{<\nabla f(\mathbf{x}), \mathbf{d^i}>}{\|\nabla f(\mathbf{x^i})\| \cdot \|\mathbf{d^i}\|}$. Then Newton's method converges globally.*

**Do you recall?**

In the notes on Numerical Methods we stated the **Variational characterization**:
Let $Q \in S(n, \mathbb{R})$ and let $\mathbf{x} \in \mathbb{R}^n$. Then

$$\lambda_{\min}\|\mathbf{x}\|^2 \leq \mathbf{x}^T Q \mathbf{x} \leq \lambda_{\max}\|\mathbf{x}\|^2$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are respectively the eigenvalue of maximum value and the eigenvalue of minimum value.

*Proof.* The variational characterization of the eigenvalues and the definition of $\mathbf{d^i}$ ($\mathbf{d^i} := -[\nabla^2 f(\mathbf{x^i})]^{-1} \nabla f(\mathbf{x^i}) \iff \nabla^2 f(\mathbf{x^i})\mathbf{d^i} = -\nabla f(\mathbf{x^i})$) leads to the following upper-bounds:

1. $$\mathbf{d^i}^T \nabla f(\mathbf{x^i}) = -(\mathbf{d^i})^T \nabla^2 f(\mathbf{x^i}) \mathbf{d^i} \leq -\lambda_{\min} \left\|\mathbf{d^i}\right\|^2$$

2. $$\left\|\nabla f(\mathbf{x^i})\right\| = \left\|\nabla^2 f(\mathbf{x^i})\mathbf{d^i}\right\| \leq \left\|\nabla^2 f(\mathbf{x^i})\right\| \cdot \left\|\mathbf{d^i}\right\| = \lambda_{\max} \left\|\mathbf{d^i}\right\|$$

From the definition of $\cos\theta^i = \frac{<\nabla f(\mathbf{x}), \mathbf{d^i}>}{\|\nabla f(\mathbf{x^i})\| \cdot \|\mathbf{d^i}\|}$ we get

$$\cos(\theta^i) \leq -\lambda_{\min}/\lambda_{\max} \leq -u/L$$

□

**Theorem 4.6.4.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ s.t. $f \in \mathcal{C}^3$ and such that satisfies $uI \preceq \nabla^2 f \preceq LI$ and $\cos(\theta^i) \leq -\lambda_{min}/\lambda_{max} \leq -u/L$. Then the method not only converges, but we also have that from some iteration onward, $\alpha^i = 1$ always satisfies Armijo's condition.*

*Proof.*

$$\begin{aligned}
\varphi(\alpha = 1) = f(\mathbf{x^i} + \mathbf{d^i}) &= f(\mathbf{x^i}) + < \nabla f(\mathbf{x^i}), \mathbf{d^i} > + \frac{1}{2} \cdot (\mathbf{d^i})^T [\nabla^2 f(\mathbf{x^i})] \mathbf{d^i} + R_3(\left\|\mathbf{d^i}\right\|) \\
&= f(\mathbf{x^i}) + (\mathbf{d^i})^T \nabla f(\mathbf{x^i}) + \frac{1}{2} \cdot (\mathbf{d^i})^T [\nabla^2 f(\mathbf{x^i})] \mathbf{d^i} + R_3(\left\|\mathbf{d^i}\right\|) \\
&= f(\mathbf{x^i}) - \nabla f(\mathbf{x^i})^T \left( [\nabla^2 f(\mathbf{x^i})]^{-1} \right)^T \nabla f(\mathbf{x^i}) + \\
&+ \frac{1}{2} \nabla f(\mathbf{x^i})^T \left( [\nabla^2 f(\mathbf{x^i})]^{-1} \right)^T \cancel{\nabla^2 f(\mathbf{x^i})} \cancel{[\nabla^2 f(\mathbf{x^i})]^{-1}} \nabla f(\mathbf{x^i}) + R_3(\left\|\mathbf{d^i}\right\|) \\
&\overset{*}{=} f(\mathbf{x^i}) - \nabla f(\mathbf{x^i})^T [\nabla^2 f(\mathbf{x^i})]^{-1} \nabla f(\mathbf{x^i}) + \\
&+ \frac{1}{2} \nabla f(\mathbf{x^i})^T [\nabla^2 f(\mathbf{x^i})]^{-1} \nabla f(\mathbf{x^i}) + R_3(\left\|\mathbf{d^i}\right\|) \\
&= f(\mathbf{x^i}) - \frac{1}{2} \cdot \nabla f(\mathbf{x^i})^T \cdot \underbrace{[\nabla^2 f(\mathbf{x^i})]^{-1} \nabla f(\mathbf{x^i})}_{\mathbf{d^i}} + R_3(\left\|\mathbf{d^i}\right\|) \\
&= \underbrace{f(\mathbf{x^i})}_{\varphi(0)} \underbrace{- \frac{1}{2}}_{m_1} \underbrace{< \nabla f(\mathbf{x^i}), \mathbf{d^i} >}_{\varphi'(0)} + R_3(\left\|\mathbf{d^i}\right\|)
\end{aligned} \tag{4.6.1}$$

where $\overset{*}{=}$ follows from Corollary 2.7.13 ($f \in \mathcal{C}^2 \Rightarrow \nabla^2 f \in S(n, \mathbb{R})$) and the fact that the inverse of a symmetric matrix (if it exists) is a symmetric matrix as well. □

It can be proved that the convergence is superlinear.

If we start with a step size of 1, we end up in a situation in which the line search is not computed when we are close to the minimum.

This works under the assumption that the eigenvalues are bounded both above and below (deriving from the bounds on the Hessian).

### 4.6.2 Interpretation of Newton's method

**Mantra**

Newton's method is exactly gradient method on a dilated space. Or equivalently, Newton's method is the gradient method with some preconditioning.

Let us take $Q \in M(n, \mathbb{R})$ such that $Q \succeq 0$, then $Q = R \cdot R$ for some $R \in M(n, \mathbb{R})$. Let us perform a variable change ($\mathbf{y} = R\mathbf{x}$) — which is possible given that $R$ is non singular — and we get $f_{\mathbf{y}}(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T I\mathbf{y} + \mathbf{q}R^{-1}\mathbf{y}$, which has as an Hessian the identity matrix, which is the optimal matrix for convergence in Newton's method.

Formally, the descending direction $\mathbf{d_y}$ is computed as follows: $\mathbf{d_y} = -\nabla f_{\mathbf{y}}(\mathbf{y}) = -\mathbf{y} - R^{-1}\mathbf{q}$. Since we chose 1 as step size we obtain that $\nabla f_{\mathbf{y}}(\mathbf{y} + \mathbf{d_y}) = \nabla f_{\mathbf{y}}(\cancel{\mathbf{y}} - \cancel{\mathbf{y}} - R^{-1}\mathbf{q}) = \nabla f_{\mathbf{y}}(-R^{-1}\mathbf{q}) = 0$.

It takes only one iteration, because all the eigenvalues are 1 so the the ratio between the greatest and the smallest is 1 and the subtraction is 0.

If we do the inverse operation ($\mathbf{x} = R^{-1}\mathbf{y}$) to the direction we get $\mathbf{d_x}$, the direction in $\mathbf{x}$ variable.

**Problem:**

We made a lot of assumptions on the Hessian, without the ones there's no guarantee that we are moving on a descending direction. How can we relax these constraints?

### 4.6.3 Non convex case

Saying that the function $f$ is not convex is equivalent to saying that the Hessian is indefinite, therefore not invertible. In such case we propose to compute the descent direction (previously defined as $\mathbf{d^i} = -[\nabla^2 f(\mathbf{x^i})]^{-1}\nabla f(\mathbf{x^i})$) substituting the inverse of the Hessian with a matrix $H'$ which is strictly positive definite ad has more or less the same properties of the Hessian.

For example, we can build $H^i$ summing to the Hessian a multiple of the identity matrix: $H^i = \nabla^2 f(\mathbf{x^i}) + \varepsilon^i I \succ 0$. The sequence $\{H^i\}$ is supposed to converge to a matrix $H'$ that is strictly positive definite.
A proper choice for $\varepsilon$ should ensure that all the eigenvalues are at least $\delta > 0$, for an "appropriately chosen smallish $\delta$" such that $uI \preceq H' \preceq LI$. Formally,

$$\varepsilon = \max\{0,\ \delta - \lambda_{\min}\}$$

The real value $\delta$ should not be too small for both numerical (any double $\leq$ `1e-16` is treated as 0 by computers) and algorithmic (the smaller $\lambda_{\min}(\nabla^2 f(\mathbf{x^i}) + \varepsilon I)$ the more elongated the axes of $S(Q_{\mathbf{x^i}}, \cdot)$) reasons.

It can be proved that the $\varepsilon$ we chose is the solution to an optimization problem: $\min\{\left\|H - \nabla^2 f(\mathbf{x^i})\right\| \ H \succeq \delta I\}$. The choice for $\delta$ in the Matlab code is $10^{-6}$.

**Observation 4.6.1.** *Note that these constraints are important and we will get back to them later on in the course.*

**Do you recall?**

Let us rewrite here matrix norm from Definition 2.4.9: Let $A \in M(n, \mathbb{R})$, we define **Frobenius' norm** of $A$ the square root of the squared sum of all its entries. Formally,

$$\|A\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} {a_{ij}}^2}$$

Could we use a different norm instead of the 2-norm? Yes, for example we can use Frobenius's norm, changing a bit the algorithm, but we still get convergence. We would need to solve $\min\{\left\|H - \nabla^2 f(\mathbf{x^i})\right\|_F \ H \succeq \delta I\}$, which is performed in two steps:

1. compute spectral decomposition $\nabla^2 f(\mathbf{x^i}) = H\Lambda H^T$
2. $H^i = H\bar{\Lambda}H^T$ with $\bar{\gamma}^i = \max\{\lambda^i, \delta\}$

In both cases, $\{\mathbf{x^i}\} \to \mathbf{x}_*$ with $\nabla^2 f(\mathbf{x}_*) \succeq \delta I$, which implies $\varepsilon^i = 0$ and $H^i = \nabla^2 f(\mathbf{x^i})$ eventually.

**Fact 4.6.5.** *If "$H^i$ looks like $\nabla^2 f(\mathbf{x^i})$ along $\mathbf{d^i}$", formally* $\lim_{i\to\infty} \left\|(H^i - \nabla^2 f(\mathbf{x^i}))\mathbf{d^i}\right\| \left\|\mathbf{d^i}\right\| = 0$*, the convergence is* superlinear.

This proposition is important because it stresses that matrix $H$ does not have to be close to the Hessian except for a particular direction.

**Computational complexity**

We still need to compute eigenvalues, which takes $O(n^3)$, which is too much if we are in multidimensional spaces.

As a closing observation we want to stress that Newton's method is very fast to converge to a local minimum, but this behaviour may represent a problem, because the algorithm may miss global minima.

**Key observation**

When we look at a quadratic model approximating $f$ around a certain point $\mathbf{x}$ we could not take negative curvature directions, although those are the *most promising* directions because there the function value decreases. In quasi-Newton methods we will try to look at those directions, removing the constraint of second order differentiability and then we will try to avoid inverting matrices.

When the Hessian is not strictly positive definite, it is not invertible, hence Newton's direction $\mathbf{d^i} = -\nabla^2 f(\mathbf{x^i})^{-1}\nabla f(\mathbf{x^i})$ is not well defined.
Moreover, when the Hessian is not even *positive semidefinite* it means that there are some directions of negative curvature, hence the function is unbounded below along some directions.
The approach we presented in last lecture ($H^i = \nabla^2 f(\mathbf{x^i}) + \varepsilon I$ or $H^i = \lambda_{\min} \cdot (\nabla^2 f(\mathbf{x^i}) + \varepsilon I)$) consists in modifying the Hessian in a way in which we do not consider such direction.

In general, we know that a model is reliable only around a certain point $\mathbf{x^i}$, so we can find the minimum of a model in a ball around $\mathbf{x^i}$ (*trust region*).
Formally, we aim to solve $\mathbf{x^{i+1}} \in \arg\min\{Q_{\mathbf{x^i}}(\mathbf{y}) \ : \ \mathbf{y} \in \mathcal{T}^i\}$, where $\mathcal{T}^i \subseteq \mathbb{R}^n$ and $\mathcal{T}^i$ is a compact set where the quadratic model around $\mathbf{x^i}$ ($Q_{\mathbf{x^i}}$) can be trusted.
In our case, the Hessian is not positive semidefinite, hence the model has no minimum, unless we restrict to a **compact set**, in which we can trust our model (**trust region**).

**Observation 4.6.2.** *Finding the minimum of $f$ in such a region is a* NP-hard problem*, if we do not restrict to a rounded ball.*

Let us consider to pick $r$ as the radius of the ball defining the trust region, then the scalar $\lambda \in \mathbb{R}$ that satisfies the Karush-Khun-Tucker conditions is fully determined.

**Definition 4.6.1** (Karush-Khun-Tucker conditions)**.** *Any optimal solution of the problem $\mathbf{x^{i+1}}$ must satisfy that $\equiv \exists\, \lambda \geq 0$ s.t.*

KARUSH: $[H^i + \lambda I]\mathbf{x^{i+1}} = -\nabla f(\mathbf{x^i})$ *[linear];*

KUHN: $H^i + \lambda I \ \succeq \ 0$ *[semidefinite];*

TUCKER: $\lambda(r - \left\|\mathbf{x^{i+1}}\right\|) = 0$ *[nonlinear].*

*Where the last condition means that either $\lambda = 0$ or the solution lies on the border of the ball.*

As an example, let us take a big ball (large $r$) and let $\mathbf{x^{i+1}}$ *not* on the border, the *complementary slackness* is satisfied iff $\lambda = 0$.

What's the difference between this approach and what we used to do before? We have two different cases:

- If $\left\|\mathbf{x^{i+1}}\right\| < r$ the the *Complementary slackness* condition is satisfied only if $\lambda = 0$. In this scenario, the Hessian is not amended, which implies that we perform exaclty the Newton's method, without taking into account $\mathcal{T}$;
- Conversely, we need to find $\lambda > 0$ and we need to solve a linear system (Karush condition) and this means solving a linear system (complexity $O(n^3)$). Notice that this is the same of performing a line search using $\varepsilon^i = \lambda$.

Notice that the *trust region method* works selecting first the step size (the radius of the ball) and then the direction.

**Key idea**

We don't need to compute the Hessian, we can use the first order information to infer things on the second order matrix.

## 4.7 Quasi-Newton's Methods

*Quasi Newton* methods exploit first order information (computing gradients) to approximate $H^i$, using an iterative process.

From the $i$-th iteration to the $(i+1)$-th we compute two different Hessians ($H^i$ and $H^{i+1}$) that can be chosen custom(ish).

At the $i$-th step we have the $i$-th model defined as

$$m^i(\mathbf{x}) = \nabla f(\mathbf{x^i})(\mathbf{x} - \mathbf{x^i}) + \frac{1}{2}(\mathbf{x} - \mathbf{x^i})^T H^i(\mathbf{x} - \mathbf{x^i})$$

The next step $\mathbf{x^{i+1}}$ is computed as $\mathbf{x^{i+1}} = \mathbf{x^i} + \alpha^i \mathbf{d^i}$.

At the next step we need to recompute the gradient in $\mathbf{x^{i+1}}$ and the matrix $H^{i+1}$ and then build the model as

$$m^{i+1}(\mathbf{x}) = \nabla f(\mathbf{x^{i+1}})(\mathbf{x} - \mathbf{x^{i+1}}) + \frac{1}{2}(\mathbf{x} - \mathbf{x^{i+1}})^T H^{i+1}(\mathbf{x} - \mathbf{x^{i+1}})$$

**Property 4.7.1.** *$H^i$ should:*

1. *positive definite ($H^{i+1} \succ 0$);*
2. *allow the following equality: $\nabla m^{i+1}(\mathbf{x^i}) = \nabla f(\mathbf{x^i})$, where we know the gradient in the previous point and the gradient in the current point. Equivalently, $H^{i+1}(\mathbf{x^{i+1}} - \mathbf{x^i}) = \nabla f(\mathbf{x^{i+1}}) - \nabla f(\mathbf{x^i})$, which we call **secant equation** and we denote (S).*

   *Come mai sono equivalenti? Non manca un $(\mathbf{x} - \mathbf{x^{i+1}})$?*

3. *be such that $\left\|H^{i+1} - H^i\right\|$ is "small".*

Let us define $\mathbf{s^i}$ such that: $\mathbf{s^i} = \mathbf{x^{i+1}} - \mathbf{x^i} = \alpha^i \mathbf{d^i}$ and $\mathbf{y^i} = \nabla f(\mathbf{x^{i+1}}) - \nabla f(\mathbf{x^i})$. Using this new notation, we can rewrite the second condition that should be satisfied by $H^i$ as $H^{i+1}\mathbf{s^i} = \mathbf{y^i} \Rightarrow \mathbf{s^i}^T\mathbf{y^i} = \mathbf{s^i}^T H^{i+1}\mathbf{s^i}$. Provided that $H^{i+1} \succ 0$ we have that $\mathbf{s^i}^T\mathbf{y^i} > 0$. Moreover, $\mathbf{s^i}$ depends the stepsize that should be properly chosen in order to satisfy the curvature condition of the subsequent step.

$s^i$ is chosen, while $y^i$ is decided by the function.

In order to have a matrix $H^i$ that satisfies the first two conditions we could check that $H^{i+1}\mathbf{s^i} = \mathbf{y^i}$, because this implies $\mathbf{s^i}\mathbf{y^i} = (\mathbf{s^i})^T H^{i+1}\mathbf{s^i}$ and this implies 1. and 2., hence we obtain the **curvature condition** (C) $\mathbf{s^i}^T\mathbf{y^i} > 0$. This also implies that if a couple $\mathbf{y^i}, \mathbf{s^i}$ does not have positive scalar product then the corresponding $H^i$ does not respect all the three conditions stated above.

In we choose a step that satisfies the Wolfe's condition, automatically the curvature condition is satisfied.

**Theorem 4.7.2.** *Wolf condition implies $\mathbf{s^i}^T\mathbf{y^i} > 0$, using the notation we introduced: (W) $\Longrightarrow$ (C).*

*Proof.*

$$\varphi'(\alpha^i) = \nabla f(\mathbf{x^{i+1}})d^i \geq m_3\varphi'(0) = m_3\nabla f(\mathbf{x^i})d^i$$

$$\Downarrow$$

$$(\nabla f(\mathbf{x^{i+1}}) - \nabla f(\mathbf{x^i}))d^i \geq (m_3 - 1)\varphi'(0) > 0$$

□

**Observation 4.7.1.** *We may observe that this theorem implies that if we perform Armijo Wolf exact line search condition (C) can always be satisfied.*

### 4.7.1 Davidson-Fletcher-Powell

Let us rewrite the three conditions above as $H^{i+1} = \arg\min\{\|H - H^i\| : \text{(S) holds and } H \succeq 0\}$.

This minimum problem can be solved using a closed formula when the norm has certain characteristics.

**Theorem 4.7.3** (Davidson-Fletcher-Powell)**.** *The new matrix is obtained at each step constructing a rank two matrix, obtained from $H^i$ as a rank two correction, as follows:* $H^{i+1} = \underbrace{(I - \rho^i y^i (s^i)^T)}_{rank1} H^i \underbrace{(I - \rho^i \mathbf{s^i}(\mathbf{y^i})^T)}_{rank2} + \underbrace{\rho^i \mathbf{y^i}(\mathbf{y^i})^T}_{rank2}$

*where* $\rho^i = 1/(\mathbf{y^i}^T \cdot \mathbf{s^i})$

Let us denote $B^i = H^{i^{-1}}$. At any step we need to compute $B^{i+1} = (H^{i+1})^{-1}$, because we need to solve the system. We have some formulas that give us a way to compute $(H^{i+1})^{-1}$ from $H^{i^{-1}}$.

**Theorem 4.7.4** (Sherman-Morrison-Woodbury)**.** *Let $A \in GL(n, \mathbb{R})$. The inverse of a rank-one correction of $A$ (which has the form $A+\mathbf{ab}^T$, where $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$) is also a matrix with a rank one correction. Formally,*

$$[A + \mathbf{ab}^T]^{-1} = \frac{A^{-1} - A^{-1}\mathbf{ab}^T A^{-1}}{1 - \mathbf{b}^T A^{-1}\mathbf{a}}$$

**Corollary 4.7.5.** *From Theorem 4.7.4 we can conclude that* $B^{i+1} = \frac{B^i + \rho^i \mathbf{s^i}(\mathbf{s^i})^T - B^i \mathbf{y^i}(\mathbf{y^i})^T B^i}{(\mathbf{y^i})^T B^i \mathbf{y^i}}$.

Thanks to Corollary 4.7.5, we get that computing $B^{i+1}$ once we have $B^i$ takes $O(n^2)$.
We can do better, in terms of computational complexity.

### 4.7.2 Broyden-Fletcher-Goldfarb-Shanno

We can use directly $B^i$, defined as the inverse of $H^i$. Write (S) for $B^{i+1}$: $\mathbf{s^i} = B^{i+1}\mathbf{y^i} \implies B^{i+1} = \arg\min\{\|B - B^i\| \ : \ \dots\}$.

$$H^{i+1} = H^i + \rho^i \mathbf{y^i}\mathbf{y^i}^T - \frac{H^i \mathbf{s^i}\mathbf{s^i}^T H}{\mathbf{s^i}^T H^i \mathbf{s^i}}$$

$$\begin{aligned} B^{i+1} &= (I = \rho^i \mathbf{s^i}\mathbf{y^i}^T) B^i (I - \rho^i \mathbf{y^i}\mathbf{s^i}^T) + \rho^i \mathbf{s^i}\mathbf{s^i}^T \\ &= B^i + \rho^i [(1 + \rho^i (\mathbf{y^i})^T B^i \mathbf{y^i}) \mathbf{s^i}(\mathbf{s^i})^T - (B^i \mathbf{y^i}(\mathbf{s^i})^T + \mathbf{s^i}(\mathbf{y^i})^T B^i)] \end{aligned} \quad \text{(BFGS)}$$

This formula proves to be more stable than the other one.
This method takes $O(n^2)$.
The two $B^i$s, obtained from DFP and BFGS, are *different* although both sensible. It turns out that the BFGS formula works better than the others, but we can use a convex combination of the two.

**Observation 4.7.2.** *How can we choose $H^1$? The value at the first step will make a difference n the results, at least for the first steps.*

Let us see a couple of choices for $B^1$:

- Scalar multiples of identity, but how to choose the scalar?
- Approximate the Hessian as a finite difference:
- Compute the gradient in $n$ directions and approximate $H$. This will cost $O(n^3)$, but it should be done only once.

Let us compute the space needed to store the $B^i$s: order of $n^2$ is still a lot. What happens if we restrict to working with information of the last $k$ operations?

### 4.7.3 Poorman's approach - limited memory BFGS

At each step we only consider $B^{i-k}$ and $k$ rank one operations using an "unfolding" approach. This operations cost $n$ each, and we have $k$ lines. Formally, given $V^i = I - \rho^i \mathbf{y^i s^i}^T$

$$B^{i+1} = (V^i \cdot V^{i-1} \cdot \ldots \cdot V^{i-k})^T B^{i-k} (V^i \cdot V^{i-1} \cdot \ldots \cdot V^{i-k}) +$$
$$+ \rho^{i-k+1} (V^i \cdot \ldots \cdot V^{i-k+1})^T \mathbf{s^{i-k+1}} \cdot (\mathbf{s^{i-k+1}})^T \cdot (V^i \cdot \ldots \cdot V^{i-k+1}) +$$
$$+ \rho^{i-k+2} (V^i \cdot \ldots \cdot V^{i-k+2})^T \mathbf{s^{i-k+2}} (\mathbf{s^{i-k+2}})^T \cdot (V^i \cdot \ldots \cdot V^{i-k+2}) + \ldots + \rho^i \mathbf{s^i s^i}^T$$

The problem is that $B^{i-k}$ takes $O(n^2)$ space. We can optimize if we choose $B^{i-k}$ to be simpler, say a multiple of the identity, or finite difference of the gradient. Then the space and time complexity is $O(kn)$.
I need to tune the algorithm to find the right $k$ which gives me enough precision and also keeps the computational cost low.

**Final observation of quasi Newton methods**

We may notice that this variation of Newton method doesn't get trapped in local minima, as Newton method did. In the end, the fact that quasi Newton isn't that precise at the beginning may be a good feature.

## 4.8 Conjugate Gradient Method

**Do you recall?**

In the gradient method, the angle between two consecutive directions is exactly 90°, as can be seen in Figure 4.17. This behaviour leads to a pretty slow convergence because the algorithm does not move smoothly, but it kind of zig-zags.

We would like to take into account not only the subspace spanned by $d^{i+1}$ but we would like to optimize over larger and larger subspaces (spanned by $d^i$ and $d^{i+1}$).

**Definition 4.8.1** (Q-conjugate)**.** *Let $v$ and $w$ be vectors in $\mathbb{R}^n$. We say that* $\mathbf{v}$ *ad* $\mathbf{w}$ *are* ***$Q$-conjugate*** *if* $(\mathbf{v})^T Q \mathbf{w} = 0$.

We would like pick a direction to be $Q$-conjugate with all the previous iterations. The point is that we can't take into account all the previous directions, but we will see that we only need the previous direction to obtain all the information we need.

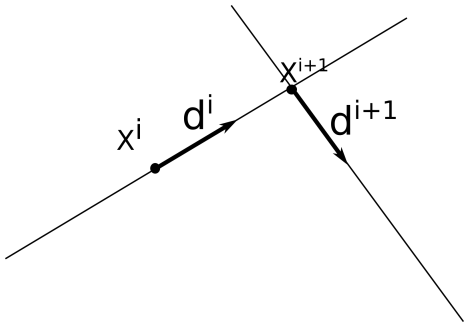



FIGURE 4.17: Geometric idea on how the new direction is chosen.

ALGORITHM 4.8.1 Pseudocode for **C**onjugate **G**radient method for **Q**uadratic functions.

```
1:  procedure CGQ(Q, q, x, ε)
2:      d⁻ ← 0;
3:      while (‖∇f(x)‖ > ε) do
4:          if (d⁻ = 0) then
5:              d ← −∇f(x);
6:          else
7:              β = (∇f(x)^T Q d⁻)/((d⁻)^T Q d⁻);
8:              d ← −∇f(x) + β d⁻;
9:          end if
10:         α ← (∇f(x)^T d)/(d^T Q d);
11:         x ← x + α d;
12:         d⁻ ← d;
13:     end while
14: end procedure
```

The number of iterations needed to converge is proportional to the clusterization of the eigenvalues of the matrix $Q$.

The algorithm that was presented is for quadratic functions, but the same algorithm works for non quadratic function as well, as long as we change the formula for $\beta$.

The pseudocode of Algorithm 4.8.2 is referred to Fletcher-Reeves' definition of $\beta^i$ $\beta^i = \left\|\nabla f(\mathbf{x^i})\right\| / \left\|\nabla f(\mathbf{x^{i-1}})\right\|^2$.

This algorithm converges in at most $n$ iterations.

ALGORITHM 4.8.2 Pseudocode for **C**onjugate **G**radient method with **A**rmijo-Wolfe's update

```
1: procedure CGA(Q, q, x, ε)
2:     ∇f⁻ = 0;
3:     while (‖∇f(x)‖ > ε) do
4:         if (∇f⁻ = 0) then
5:             d ← −∇f(x);
6:         else
7:             β = ‖∇f(x^i)‖² / ‖∇f⁻‖²;
8:             d ← −∇f(x) + βd⁻;
9:         end if
10:        α ← AWLS(f(x + αd));
11:        x ← x + αd;
12:        d⁻ ← d;
13:        ∇f⁻ ← ∇f(x);
14:    end while
15: end procedure
```

We have three different formulas for $\beta_i$, which coincide in the quadratic case.

1. Polak-Ribière: $\beta^i = \frac{[\nabla f(\mathbf{x^i})^T(\nabla f(\mathbf{x^i}) - \nabla f(x^{i-1}))]}{\|\nabla f(\mathbf{x^{i-1}})\|^2}$
2. Hestenes-Stiefel: $\beta^i = \frac{\nabla f(\mathbf{x^i})^T(\nabla f(\mathbf{x^i}) - \nabla f(\mathbf{x^{i-1}}))}{(\nabla f(\mathbf{x^i}) - \nabla f(\mathbf{x^{i-1}}))^T \mathbf{d^{i-1}}}$
3. Dai-Yuan: $\beta^i = \frac{\left\|\nabla f(\mathbf{x^i})\right\|^2}{(\nabla f(\mathbf{x^i}) - \nabla f(\mathbf{x^{i-1}}))^T \mathbf{d^{i-1}}}$

Some of these algorithms require some hypotheses on the function in order for the conjugate method to converge.

1. Fletcher-Reeves requires $m_1 < m_2 < \frac{1}{2}$ for (A) $\cap$ (W') to work;
2. (A) $\cap$ (W') $\not\Longrightarrow d^i$ of Polak-Ribière is of descent, unless $\beta^i_{PR} = \max\{\beta^i, 0\}$.

Sometimes it happens that the algorithm gets stuck because of a bad direction. In those cases, it is crucial to restart from scratches.

The idea of taking the gradient and modify it instead of multiplying by a factor, adding the previous direction.

It's possible to design hybrids between quasi-Newton and conjugate method.

## 4.9 Deflected Gradient Methods

The idea behind this family of algorithms, is to determine the next position $\mathbf{x^{i+1}}$ using the gradient and summing to it something else that gives us more information.

This kind of algorithms work also in cases in which the gradient isn't continuous.

This methods use the information about the previous iterations without exploiting properties about the second order derivative.

### 4.9.1 Heavy ball gradient method

The intuition behind this algorithm may be expressed through the following metaphore: an object is moving in the space and it's subject to a force. We can observe that the heavier the object, the stronger should be the force imposed in order to make it describe a certain trajectory.

In this interpretation, we may define the $(i+1)$-th iteration as

$$\mathbf{x^{i+1}} \leftarrow \mathbf{x^i} - \alpha^i \nabla f(\mathbf{x^i}) + \beta^i(\mathbf{x^i} - \mathbf{x^{i-1}}),$$

where $\boldsymbol{\beta^i}$ is called **momentum**, $\mathbf{x^i}$ **heavy** and $\nabla\mathbf{f}(\mathbf{x^i})$ **force**.

This is not a descent algorithm: we are not choosing a moving direction and then performing line search for picking a proper step size. We have no guarantee that the value of the function after one iteration will be smaller than the previous one.

The first thing to do is to choose $\alpha^i$ and $\beta^i$ properly.

We can prove for some cases that these methods are better than gradient method, although they aren't as good as Newton or quasi Newton. Their strength resides in their simplicity though.

Notice that if the smaller eigenvalue is not zero (i.e. quadratic case) we have a close formula to choose $\alpha$ and $\beta$ independently from the iteration:

$$\alpha = \frac{4}{\left(\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}\right)^2}, \qquad \beta = \max\left\{\left|1 - \sqrt{\alpha\lambda_{\min}}\right|, \left|1 - \sqrt{\alpha\lambda_{\max}}\right|\right\}^2$$

We may observe that the step we take is something that goes like $\frac{1}{L}$, where $L$ is the Lipschitz constant, since $\lambda_{\min}$ is very small. With these choices the rate is the following. We observe that in the gradient the rate is the same, although there aren't the square roots

$$\left\|\mathbf{x^{i+1}} - \mathbf{x}_*\right\| \leq \left(\frac{\sqrt{\lambda_{\max}} - \sqrt{\lambda_{\min}}}{\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}}\right) \cdot \left\|\mathbf{x^i} - \mathbf{x}_*\right\|$$

An alternative idea could be choosing $\beta^i$ and finding $\alpha^i$ using line search. A possible issue is that we don not know if we are moving along a descending direction, but in this method it is perfectly acceptable not to make any movement at a single step (notice that in gradient method if one step has size 0 then we will not move anymore).

$\beta^i$ is seen as an hyper-parameter, hence its value is tuned running the logarithm several times.

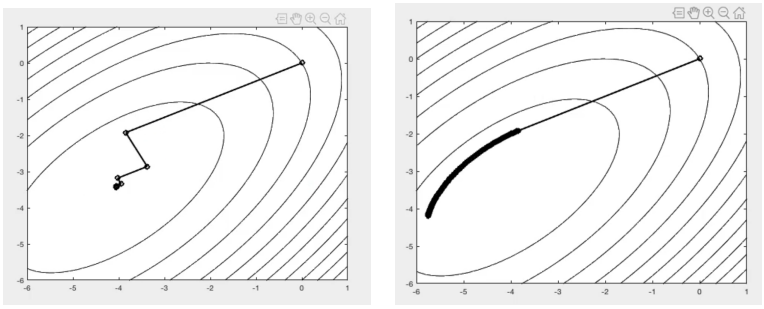


FIGURE 4.18: On the left side Newton method and on the right side the heavy ball method.

The plot of the convergence of the heavy ball method gives a graphical idea of the fact that the direction isn't orthogonal to the one at the previous iteration, since we have the gradient plus some quantity.

In particular, the bigger $\beta^i$ the "less orthogonal" the steps are. This feature allows the algorithm to have good performances on elongated functions.

### 4.9.2 Accelerated gradient

This method works only on convex functions, it has some similarities with heavy ball, but it's slightly different.

ALGORITHM 4.9.1 Pseudocode for accelerated gradient method.

**procedure ACCG**$(f, \nabla f, \mathbf{x}, \varepsilon)$
  $\mathbf{x}_- \leftarrow \mathbf{x}$;
  $\gamma \leftarrow 1$;
  **repeat**
    $\gamma_- \leftarrow \gamma$;
    $\gamma \leftarrow (\sqrt{4\gamma^2 + \gamma^4} - \gamma^2)/2$;
    $\beta \leftarrow \gamma(1/\gamma_- - 1)$;
    $\mathbf{y} \leftarrow \mathbf{x} + \beta(\mathbf{x} - \mathbf{x}_-)$;
    $\mathbf{g} \leftarrow \nabla f(\mathbf{y})$;
    $\mathbf{x}_- \leftarrow \mathbf{x}$;
    $\mathbf{x} \leftarrow \mathbf{y} - (1/L)\mathbf{g}$;
  **until** $(\|\mathbf{g}\| > \varepsilon)$
**end procedure**

The rational behind this algorithm is the following: When we are in a certain point at a certain iteration, we go on a little bit $\beta^i$ and we end up in a point $\mathbf{y}$. The gradient is computed in that point and used to choose the next point.

If we choose $\gamma$ optimally then the quadratic approximation is very close to the function. We want to find the value for $\gamma$s that gives best results in the worst case.

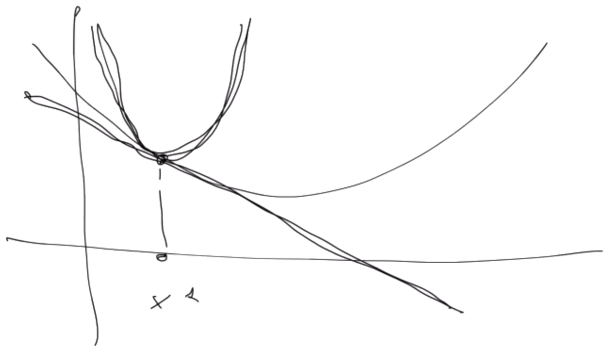


FIGURE 4.19: We build a linear model, which is a lowerbound for the function, then we build a quadratic model, which is above our function.

We can prove that the error $\left\|f(\mathbf{x^i}) - f_*\right\| \leq \sigma^i (f(\mathbf{x^i}) - f_*) \searrow 0$ is multiplied by this factor $\delta_i$, that goes like the inverse of $i^2$.

Notice that if we choose $\alpha$ small, it will always be small.

The convergence is sublinear.

**Theorem 4.9.1** (Optimality of accelerated gradient)**.** *If the function isn't strongly convex no algorithm has better convergence than* $\left|f(\mathbf{x^i}) - f_*\right| = 3L\frac{\left\|\mathbf{x^1}-\mathbf{x}_*\right\|^2}{32(i+1)^2}$.

**Observation 4.9.1.** *This theorem tells us that this algorithm never gets worse than* $\left|f(\mathbf{x^i}) - f_*\right| = 3L\frac{\left\|\mathbf{x^1}-\mathbf{x}_*\right\|^2}{32(i+1)^2}$*, but this doesn't imply that this method is fast on average. The state of the art provides a lot of different formulas for* $\beta$*, which of the ones leads to some theoretical results.*

From now on we will move towards a different family of functions, that aren't even differentiable, hence we can't compute the gradient.

## 4.10 Incremental Gradient Methods

This method has good performances in real world machine learning cases, where the function is differentiable but we do not want to compute the gradient.

Let $I = \{1, \dots, m\}$ be the set of observations, let $X = [\mathbf{x^i} \subset \mathcal{X}]_{i\in I}$ be the set of inputs and let $\mathbf{y} = [\mathbf{y^i}]_{i\in I}$ be the set of outputs. Our goal is to explain $\mathbf{y}$ from $X$.

Since these vectors are uniformly distributed over the space (at least this is our hypothesis) when they get summed we expect some of them to cancel out.

The idea is to rewrite the problem as learning a linear function in the feature space, formally $\min\left\{\sum_{i\in I} l(\mathbf{y^i}, <\Phi(\mathbf{x^i}), \mathbf{w}>) \ : \ \mathbf{w} \in \mathbb{R}^n\right\}$, where $l(\,\cdot\,,\,\cdot\,) =$ is

called **loss function** and $\Phi : \mathbb{R}^n \to M(n, m, \mathbb{R})$ is a map from the input space to the feature space.

We are now interested in computing the gradient, which is not hard to compute since the function is linear: $\nabla f(\mathbf{w}) = \sum_{i \in I} \nabla f^i(\mathbf{w}) = \sum_{i \in I} -A^i(\mathbf{y^i} - A^i\mathbf{w})$.

The issue here is that computing the gradient, although it's the gradient of a very simple function, takes too long (since there are too many vectors in machine learning datasets).

To overcome this problem we choose to restrict to only a subset of observations. How can we choose such set? Randomly, of course.

At this point the algorithm is not deterministic anymore, but it is completely **stochastic**.

The intuition behind this algorithm is to take only one observation, compute what is needed on this observation and make a small step.

An online application may be a sensor that produces hundreds of outputs per second; Storing each of them is unfeasible. They should be used to infer some information and then thrown away.

We study the converge of this kind of algorithms from a stochastic point of view.

In machine learning we always need some regularization, because the tuning of hyper-parameters clearly takes into account only the error in the samples that have been seen. Let us regularize the model as follows

$$\min \left\{ \sum_{i \in I} l(\mathbf{y^i}, < \Phi(\mathbf{x^i}), \mathbf{w} >) + \mu\Omega(\mathbf{w}) : \mathbf{w} \in \mathbb{R}^n \right\}$$

The usage of regularization may be useful, since we want to keep close to the minimum, without reaching it (because it would mean overfitting). It is enough to change slightly the problem and then solve it.

The regularization hyper-parameter $\Omega(\mathbf{w})$ may be chosen as follows:

1. Lasso regularizer (best known): $\Omega(\mathbf{w}) = \|\mathbf{w}\|_1$;
2. In order to increase sparsity: $\Omega(\mathbf{w}) = \|\mathbf{w}\|^2{}_2$;
3. Leading to feature selection: many $w_j = 0$ as possible.

$\Omega$ function is not differentiable, so the function gets non differentiable, as an example look at Figure 4.20, which represents the plot of $f(w_1, w_2) = (3w_1 + 2w_2 - 2)^2 + 10(|w_1| + |w_2|)$.

If we sit in a kinky point it might be the case that we choose a descent direction or we do not, hence we move away from the minimum or if it is reached the non-differentiability does not allow to identify that we are in the minimum. In detail, gradient descent methods work because at a certain point the movement, obtained as product between the gradient and the step size, is 0. But this cannot happen if the gradient does not vanish, because it is not defined.

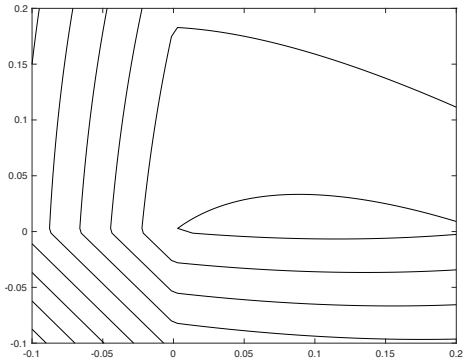



FIGURE 4.20: This function has a lot of kinky points.

## 4.11 Subgradient Methods

It is often the case that the objective function under consideration is **not differentiable** and in such situation we need to use different tools to be sure to move towards the optimum.

**Do you recall?**

Let us rewrite here the definitions introduced in Section 3.6, namely *subgradient* and *subdifferential*.
Let $f : \mathbb{R}^n \to \mathbb{R}$. We say that $\mathbf{s} \in R^n$ is a **subgradient** of $f$ at point $\mathbf{x} \in \mathbb{R}^n$ if $\forall \mathbf{y} \in \mathbb{R}^n$ the following holds:

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{s}^T(\mathbf{y} - \mathbf{x})$$

Under the same hypotheses we call **subdifferential** the set of all possible subgradients at $\mathbf{x} \in \mathbb{R}^n$.
Formally,

$$\partial f(\mathbf{x}) := \{\mathbf{s} \in \mathbb{R}^n \ : \ \mathbf{s} \text{ is a subgradient at } \mathbf{x}\}$$

**Example 4.11.1.** *Let us assume that we know the minimum point $\mathbf{x}_* \in \mathbb{R}^n$ for $f : \mathbb{R}^n \to \mathbb{R}$ and we are in $\mathbf{x}$ at the current iteration. We would like to move towards $\mathbf{x}_*$ using only the information provided by the subdifferential. $(-)\partial f(\mathbf{x})$ is convex and compact and all its elements $(-)\mathbf{s} \in \partial f(\mathbf{x})$ (subgradients) "point towards $\mathbf{x}_*$". Formally, $< \mathbf{s}, \mathbf{x}_* - \mathbf{x} >< 0$.*

**Subgradient methods** are thought for **convex functions** that are **not differentiable** in the whole domain.
The intuition behind **subgradient methods** is to move in the direction of a specific subgradient $(-\mathbf{s})$, but with a small step-size $\alpha^i$, because if the step is

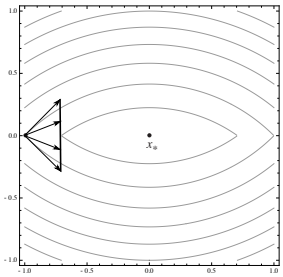


FIGURE 4.21: There are many different subgradients in $\mathbf{x}$. We pick the one with minimum norm among the ones which have a negative scalar product with $\mathbf{x} - \mathbf{x}_*$.

too large we may end up in a point which is actually further from $\mathbf{x}^*$ than the previous step. In that new point we may find a direction and perform line search, because that point is not kinky. In this context we will not try to minimize $\|f(\mathbf{x}) - f(\mathbf{x}^*)\|$, but we will minimize $\|\mathbf{x} - \mathbf{x}^*\|$, because the function may zigzag near to that point. It goes without saying that choosing a too small value for $\alpha$ leads to a too slow convergence speed.

**Do you recall?**

**??** of last lecture: let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function, $\forall \mathbf{s} \in \partial f(\mathbf{x}), \forall\, \mathbf{y} \in \mathbb{R}^n$ lis equivalent to say $f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{s}^T(\mathbf{y} - \mathbf{x})$. This is a characterization of the function with respect to the model.

Let us assume we know the minimum point $\mathbf{x}_*$. We observe that the scalar product between the subgradient and the direction we should choose is negative. Formally, $f(\mathbf{x}_*) \geq f(\mathbf{x}) + < \mathbf{d}, \mathbf{x}_* - \mathbf{x} >$, hence $< \mathbf{d}, \mathbf{x}_* - \mathbf{x} > \leq f(\mathbf{x}_*) - f(\mathbf{x}) \leq 0$, where $\mathbf{d} \in \partial f(\mathbf{s})$.

We want to bound the distance between the point at the "next step" and the optimum:

$$\begin{aligned}
\left\|\mathbf{x^{i+1}} - \mathbf{x}_*\right\|^2 &\overset{(1)}{=} \left\|\mathbf{x^i} - \alpha^i \mathbf{d^i} - \mathbf{x}_*\right\|^2 \\
&\overset{(2)}{=} \left\|\mathbf{x^i} - \mathbf{x}_*\right\|^2 + \left\|\alpha^i \mathbf{d^i}\right\|^2 + 2 < \mathbf{x^i} - \mathbf{x}_*, -\alpha^i \mathbf{d^i} > \\
&= \left\|\mathbf{x^i} - \mathbf{x}_*\right\|^2 + (\alpha^i)^2 \left\|\mathbf{d^i}\right\|^2 + 2\alpha^i \mathbf{d^i}^T (\mathbf{x}_* - \mathbf{x^i}) \\
&\overset{(3)}{\leq} \underbrace{\left\|\mathbf{x^i} - \mathbf{x}_*\right\|^2}_{\geq 0} + \underbrace{(\alpha^i)^2 \left\|\mathbf{d^i}\right\|^2}_{>0} \underbrace{-2\alpha^i \mathbf{d^i}^T (f(\mathbf{x^i}) - f(\mathbf{x}_*))}_{<0}
\end{aligned} \tag{4.11.1}$$

Where $\overset{(1)}{=}$ follows from the definition of the step: $\mathbf{x^{i+1}} = \mathbf{x^i} - \alpha^i \mathbf{d^i}$, $\overset{(2)}{=}$ follows from Corollary 2.4.6 and $\overset{(3)}{\leq}$ follows from the inequality we stated above.
The algorithm converges only if $2\alpha^i \mathbf{d^i}^T (f(\mathbf{x^i}) - f(\mathbf{x}_*)) > (\alpha^i)^2 \left\|\mathbf{d^i}\right\|^2$.

This happens when $\alpha^i$ is small enough, so that the linear part dominates the quadratic part, but $\alpha^i$ should not be too small either, according to Armijo's conditions.

**Observation 4.11.1.** *The distance from the optimum is bounded by*

$$\begin{aligned}\left\|\mathbf{x^{i+1}} - \mathbf{x}_*\right\|^2 &\leq \left\|\mathbf{x^i} - \mathbf{x}_*\right\|^2 + (\alpha^i)^2 \left\|\mathbf{d^i}\right\|^2 - 2\alpha^i \mathbf{d^i}^T (f(\mathbf{x^i}) - f(\mathbf{x}_*)) \\ &\overset{(1)}{=} \left\|\mathbf{x^i} - \mathbf{x}_*\right\|^2 + (\alpha^i)^2 - 2\alpha^i \frac{(f(\mathbf{x^i}) - f(\mathbf{x}_*))}{\left\|\mathbf{d^i}\right\|^2} \\ &\overset{(2)}{\leq} \left\|\mathbf{x^1} - \mathbf{x}_*\right\|^2 + \sum_{k=1}^{i} (\alpha^k)^2\end{aligned}$$

*Where* $\overset{(1)}{=}$ *is obtained normalizing the step direction and* $\overset{(2)}{\leq}$ *comes from adding all the steps from the first iteration.*

If the series of the squares of step sizes does not diverge, then the sequence does not diverge as well, hence it converges somewhere, say $\bar{\mathbf{x}}$.

The convergence of the series of the squares may be obtained using the following

**Definition 4.11.1** (Diminishing-Square Summable)**.** *We term a series that diverges, while the series of the squares of the terms converges, as* ***diminishing-square summable****.*

*Formally,*

*(DSS)* $\sum_{i=1}^{\infty} \alpha^i = \infty \ \wedge \ \sum_{i=1}^{\infty} (\alpha^i)^2 < \infty.$

**Fact 4.11.1.** *Assuming the function* convex*, definite in all its domain, and bounded (hence,* $\mathbf{d^i} \leq L$*) we claim that the sequence of the stepsizes is a diminishing-square sequence.*

*Proof by contradiction.* Let us assume $f(\mathbf{x^i}) - f_* \geq \varepsilon > 0, \ \forall i$. Then,

$$\left\|\mathbf{x^{i+1}} - \mathbf{x}_*\right\|^2 \leq \left\|\mathbf{x^1} - \mathbf{x}_*\right\|^2 - \delta \cdot \sum_{k=1}^{i} \alpha^k + \sum_{k=1}^{i} (\alpha^k)^2$$

The contradiction is due to the fact that as $i \to \infty$ the right-hand side goes to $-\infty$. □

This family of algorithms is clearly incredibly robust in theory, but in practice it does not work very well.

Let us make an experiment: let us suppose we know the minimum of the function $f$.

**Definition 4.11.2** (Polyak's stepsize)**.** *Let* $f$ *be our convex objective function and let* $\mathbf{x}_*$ *be the optimum for* $f$*. We term* ***Polyak's stepsize****:*

$$\textit{(PSS)}\ \alpha^i = \beta^i \frac{(f(\mathbf{x^i}) - f(\mathbf{x}_*))}{\left\|\mathbf{d^i}\right\|}$$

*where* $\beta \in (0, 2)$.

**Fact 4.11.2.** *Let* $f : \mathbb{R}^n \to \mathbb{R}$ *be a convex, bounded function and let* $\beta_i = 1$. *Then*

$$\frac{\left(f(\mathbf{x^i}) - f(\mathbf{x}_*)\right)^2}{\left\|\mathbf{d^i}\right\|^2} \leq \left\|\mathbf{x^i} - \mathbf{x}_*\right\|^2 - \left\|\mathbf{x^{i+1}} - \mathbf{x}_*\right\|^2$$

*Proof.* From Equation (4.11.1) and by substituting $\alpha^i$ we get

$$\begin{aligned}\left\|\mathbf{x^{i+1}} - \mathbf{x}_*\right\|^2 &\leq \left\|\mathbf{x^i} - \mathbf{x}_*\right\|^2 + (\alpha^i)^2 - 2\alpha^i \frac{(f(\mathbf{x^i}) - f(\mathbf{x}_*))}{\left\|\mathbf{d^i}\right\|^2} \\ &= \left\|\mathbf{x^i} - \mathbf{x}_*\right\|^2 + \frac{\left(f(\mathbf{x^i}) - f(\mathbf{x}_*)\right)^2}{\left\|\mathbf{d^i}\right\|^2} - 2\frac{(f(\mathbf{x^i}) - f(\mathbf{x}_*))}{\left\|\mathbf{d^i}\right\|} \frac{(f(\mathbf{x^i}) - f(\mathbf{x}_*))}{\left\|\mathbf{d^i}\right\|^2}\end{aligned}$$

Da finire

□

Although we do not know the optimum, let us assume that we know it and compute the efficiency.
Since we know that the sequence is bounded, we know that the objective function is globally Lipschitz (the norm of the gradient is bounded above).
The point is that the sequence $\{\mathbf{x_1}\}$ is not necessarily monotone, so we pick the so-called record value of best value ($\underline{f}^i = \min\{f(\mathbf{x^h}) \ : \ h = 1, \dots, i\}$)

$$\frac{\left(\underline{f}^i - f(\mathbf{x}_*)\right)^2}{L^2} \leq \frac{\left(f(\mathbf{x^i}) - f(\mathbf{x}_*)\right)^2}{\left\|\mathbf{d_i}\right\|^2} \leq \left\|\mathbf{x^i} - \mathbf{x}_*\right\|^2 - \left\|\mathbf{x^{i+1}} - \mathbf{x}_*\right\|^2$$

Summing for $i = 1, \dots, k$ we obtain a telescopic series:

$$\left\|\mathbf{x^1} - \mathbf{x}_*\right\| - \cancel{\left\|\mathbf{x^2} - \mathbf{x}_*\right\|} + \cancel{\left\|\mathbf{x^2} - \mathbf{x}_*\right\|} - \cancel{\left\|\mathbf{x^3} - \mathbf{x}_*\right\|} + \cdots + \cancel{\left\|\mathbf{x^k} - \mathbf{x}_*\right\|} - \left\|\mathbf{x^{k+1}} - \mathbf{x}_*\right\|$$

Hence resulting in

$$k \cdot \frac{\left(\underline{f}^k - f(\mathbf{x}_*)\right)^2}{L^2} \leq \left\|\mathbf{x^1} - \mathbf{x}_*\right\|^2 - \left\|\mathbf{x^{k+1}} - \mathbf{x}_*\right\|^2 \leq \left\|\mathbf{x^1} - \mathbf{x}_*\right\|^2 = R$$

which is equivalent to $\left(\underline{f}^k - f(\mathbf{x}_*)\right)^2 \leq \frac{R^2 L^2}{k}$, which is again equivalent to $\underline{f}^k - f(\mathbf{x}_*) \leq \sqrt{\frac{RL}{k}}$, where $L$ is the Lipschitz constant.
The issue here is that the convergence is sublinear: $k \geq \frac{1}{\varepsilon^2}$.

**Theorem 4.11.3.** *Take an algorithm that uses only the subgradient. It's possible to construct a function that makes the algorithm converge with sublinear speed. Hence, we cannot do better.*

It comes without saying that although this algorithm is not very good it is the "less bad" it can be.
There are some lucky cases in which we do know the optimal value, but this is not the case usually.

### 4.11.1 Target level stepsize

Let us assume $f(\mathbf{x}_*)$ is unknown. The **target level stepsize** algorithm is used to approximate it iteratively The only information available is that this value is below the function value at any possible iteration.

The rationale behind this algorithm is to assume to know the optimal value and as soon as we realize it is not correct we change it.

Let us first give an informal description of the algorithm:

- $\delta$ is the displacement: how much below the function is with respect to the best value obtained so far;
- reference value $f_{rec} = \underline{f}$. At the beginning is the value at the first iterate and then we define the target value as the difference between the reference value and some $\delta$ (at the beginning $\delta_0$).

ALGORITHM 4.11.1 Pseudocode for target level stepsize.

```
1:  procedure SGPTL(f, g, x, i_max, β, δ_0, R, ρ)
2:      r ← 0;
3:      δ ← δ_0;
4:      f_ref ← f_rec ← f(x);
5:      i ← 1;
6:      while (i < i_max) do
7:          d = g(x);
8:          α = β(f(x) − (f_ref − δ))/‖d‖^2;
9:          x ← x − αd;
10:         if (f(x) ≤ f_ref − δ/2) then
11:             f_ref ← f_rec;
12:             r ← 0;
13:         else
14:             if ( r > R ) then
15:                 δ ← δρ;
16:                 r ← 0;
17:             else
18:                 r ← r + α ‖d‖;
19:             end if
20:         end if
21:     end while
22:     f_rec ← min{ f_rec , f(x) };
23:     i ← i + 1;
24: end procedure
```

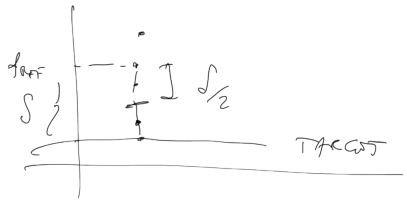


FIGURE 4.22: There are two cases: either the function value is significantly below the reference (for example $\frac{\delta}{2}$ below the reference) or it's not . If we are in the "happy case" (we just move the reference to the best value). For what concerns the "unhappy case", if we are unhappy for 1 iteration, no problem. Two iterations? no problem. Many iterations? Problem: after some iterations in which we are not improving it means that we have to decrease the reference values. How? $r$ is updated at each bad step and reset when a good step occurs. When $r$ gets too large we decrease $\delta$ and reset everything.

At this point we defined the algorithm, but for implementing it we need to choose of a lot of parameters. A way to choose them is to use grid search or any other kind of hyper-parameter tuning.

Two big issues of this algorithm are that it does not provide a good stopping criterion and it is very sensitive to many parameters.

## 4.12 Deflected Subgradient Methods

**Deflected Subgradient methods** are also thought for **convex functions** that are **not differentiable** in the whole domain. The idea is to use the same trick of ball-step (also called primal-dual). We are lying in $\mathbf{x^i}$ and we would like to move toward $\mathbf{x^{i+1}}$ without getting out of the ball with a certain radius around $\mathbf{x}_*$.

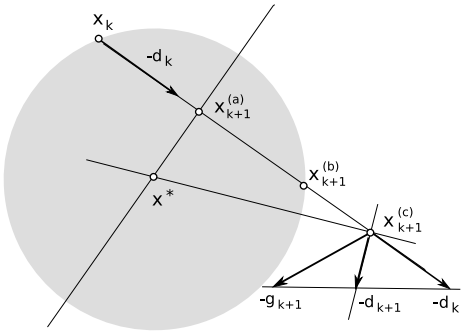



FIGURE 4.23: Representation of deflected subgradient method.

Let us assume that our function was differentiable. The subgradient method

collapses to the gradient method and we know that the gradient method does not provide a good convergence. Yet, deflection is possible: $\mathbf{d^i} = \gamma^i g^i + (1 - \gamma^i)\mathbf{d^{i-1}}$, $\mathbf{x^{i+1}} = \mathbf{x^i} - \alpha^i \mathbf{d^i}$. We can prove that $\mathbf{d^i}$ approximates the subgradient. We can also prove that the algorithm converges in the end. The parameters of this algorithm are two: $\beta$ (stepsize) and $\gamma$ (deflection). In order to choose them we have two different approaches:

STEPSIZE-RESCRICTED $\equiv$ deflection-first. We first choose $\beta$ and when choosing $\alpha$ we need to take into account $\beta$: $\alpha^i = \frac{\beta^i (f(\mathbf{x^i}) - f_*)}{\|\mathbf{d^i}\|} \wedge \beta^i \leq \gamma^i$ "as deflection $\nearrow$, stepsize has to $\searrow$";

DEFLECTION-RESTRICTED $\equiv$ stepsize-first. We first choose $\gamma$, then we pick a step size that depends on $\gamma$:

$$\text{(DSS)} \ \wedge \ \frac{\alpha^{i-1} \left\|\mathbf{d^{i-1}}\right\|}{(f(\mathbf{x^i}) - f_*) + \alpha^{i-1}} \left\|\mathbf{d^{i-1}}\right\| \leq \gamma^i$$

"as $f(\mathbf{x^i}) \to f_*$, deflection $\searrow$"

This algorithm gets the optimal $O(1/\varepsilon^2)$ on average, sadly not worst case.

## 4.13 Smoothed Gradient Methods

**Smoothed gradient methods** are also thought for **convex functions** that are **not differentiable** in the whole domain.

Intuitively, **smoothed gradient methods** "fix" the kinky points by making the function smooth in those points.

**Definition 4.13.1** (Lagrangian function)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function of the following shape:*

$$f(\mathbf{x}) = \max\{\mathbf{x}^T A \mathbf{z} \ : \ \mathbf{z} \in Z\}$$

*where $Z$ is convex and bounded.*

Let us assume that $Z$ is also compact (which is equivalent to saying that it is closed and bounded).

**Example 4.13.1.** *A graphical example in the case of $f : \mathbb{R} \to \mathbb{R}$ s.t. $f(x) = |x| = \max\{x, -x\} = \max\{zx : z \in [-1, 1]\}$ is shown in Figure 4.24.*

*In the case of the absolute value, the nifty trick is "to make it have only one optimal solution" in the point* 0*.*

This result is obtained by adding a small quadratic term: $f(\mathbf{x}) = max\{\mathbf{x}^T A \mathbf{z} - \mu \|\mathbf{z}\|^2 \ : \ \mathbf{z} \in Z\}$. This trick is depicted in two dimension in Figure 4.25. At this point the new function $f_\mu$ is not the original function anymore, but it is very close to it whenever the constant $\mu$ is small.
Notice that this new function is smooth (differentiable) and it might be not very easy to compute once the value for $\mu$ was chosen.

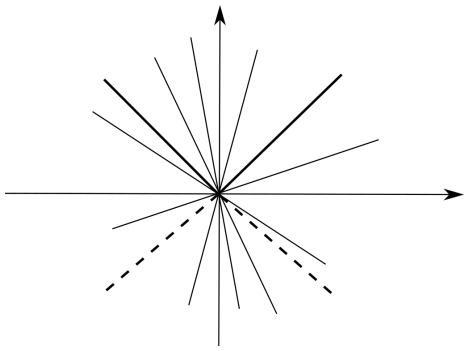


FIGURE 4.24: Let us take the absolute value function $f(x) = |x|$. This function would be differentiable, if it wasn't for some nasty points (in this case 0, were the optimization problem has many optimal solutions). In 0 there are many subgradients, so it has many optima.

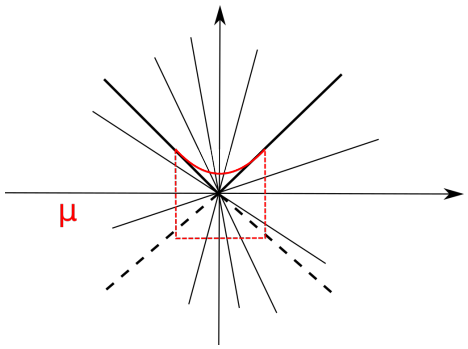



FIGURE 4.25: Geometric intuition of the usage of variable $\mu$.

In general, we need $\mu$ to be close to 0 but not too close, because for $\mu = 0$ the function is not differentiable.

At this point we are in the situation of $f_\mu(\mathbf{x}) \leq f(\mathbf{x}) \leq f_\mu(\mathbf{x}) + \mu R$, such that as $\mu \searrow 0$, "$\arg\min\{ f_\mu(\mathbf{x}) \} \to \mathbf{x}_*$".

The new function is not only convex, but it is also Lipschitz continuous and its gradient $\nabla f_\mu$ is $L$-Lipschitz with $L = \|A\|^2/\mu$, but it is "less and less Lipschitz" as $\mu \searrow 0$.

**Fact 4.13.1.** *If $f_* > -\infty$ and picking a very special value of $\mu$ ($\mu = \varepsilon/(2R)$), then an appropriate ACCG obtains $f(\mathbf{x}^\mathbf{i}) - f_* \leq \varepsilon$ for $i \geq 4 \cdot \|A\| \cdot \|\mathbf{x}_*\| \cdot \sqrt{R}/\varepsilon$.*

We observe that the convergence is much better, because it depends on $O(1/\varepsilon)$ instead of $O(1/\varepsilon^2)$.

FIGURE 4.26: At the beginning we will make a lot of bad steps (the upper gray line). We can improve (pick the black line) changing the $\varepsilon$ value and we obtain something that looks more stable. The more precision we want, the smaller the step we make. At the ending it pays, but at the beginning it is not so.

## 4.14 Bundle Methods

### 4.14.1 Cutting-plane algorithm

We cheated to get first order information, we want to do more. We want to cheat and have also the second order information.

We want to use the same idea of limited memory of quasi-Newton methods. Using some limited memory of the Hessian, in order to understand th curvature.

The point is that the directional derivatives are defined and (if computed massively) give me a hint of the curvature of the function (cfr. Figure 4.27). Notice that it is not possible to build a matrix, because it is not defined.

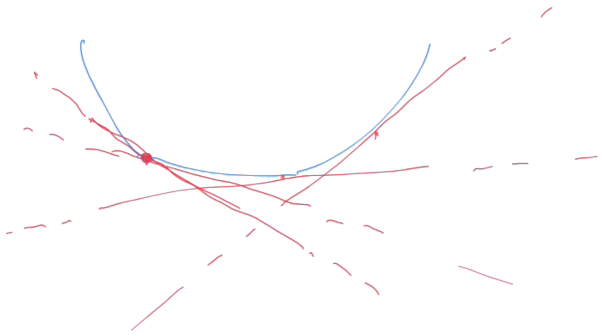


FIGURE 4.27: The idea is that we do not compute a subgradient and discard it. We store it, thus resulting in a model

Let us say we performed $i$ iterates, hence we computed $i$ subgradients and function values. Let us store them: $\mathscr{B} = \{(\mathbf{x^i},\ f^i = f(\mathbf{x^i}),\ \mathbf{s^i} \in \partial f(\mathbf{x^i}))\} \equiv$ bundle of first-order information.

We can now define a piece wise linear function defined as the maximum of the first order model:

$$f_{\mathscr{B}}(\mathbf{x}) = \max\{f^i + \mathbf{s^i}^T(\mathbf{x} - \mathbf{x^i}) \ :\ (\mathbf{x^i}, f^i, \mathbf{s^i}) \in \mathscr{B}\}$$

At this point we can apply Newton's method: first we minimize the model and then use the minimum as next point. We collect information in that specific point and then we repeat.

Notice that the model is always below the objective function ($f_{\mathscr{B}}(\mathbf{x}) \leq f(\mathbf{x})\ \forall \mathbf{x}$), hence $\min\{f_{\mathscr{B}}(\mathbf{x})\} \leq f_*$, so $\mathbf{x}^*{}_{\mathscr{B}} \in \operatorname{argmin}\{f_{\mathscr{B}}(\mathbf{x})\} \approx \mathbf{x}_*$.

This function has many kinky points, so how to minimize it? Add linear constraints, so that methods like simplex can be used.

$$\min\{\, f_{\mathscr{B}}(\mathbf{x})\,\} = \min\{\, v \,:\, v \geq f^i + \mathbf{s^i}^T(\mathbf{x} - \mathbf{x^i}),\ \text{where } (\mathbf{x^i}, f^i, \mathbf{s^i}) \in \mathscr{B}\}$$

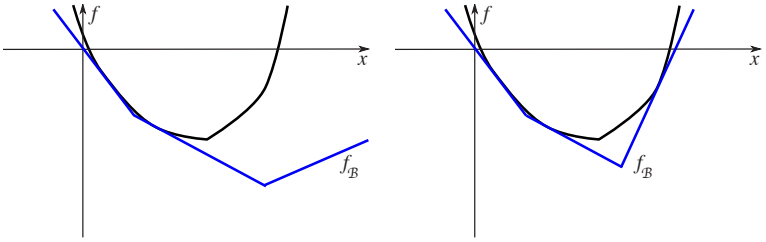



FIGURE 4.28: A geometric representation of how the model (blue line) changes after one iteration.

At this point we have obtained both an upper-bound and a lower-bound, which go one towards the other.
The problem is that at each step we need to solve a minimization problem and the convergence is very slow.

It's also possible that the blue function (the model) does not have a minimum (unbounded below).

How to overcome this problem? Use so-called bundle methods.

### 4.14.2 Bundle methods

The intuition behind **bundle methods** is to keep close to the point where the model corresponds to the function, because the further we get the more distant from the function.

A way to express this is to add a quadratic quantity to the function that grows when we move outside the current point.

**Definition 4.14.1** (Stabilized master problem)**.** *We term* ***stabilized master problem*** *the following*

$$\min\left\{ f_{\mathscr{B}}(\mathbf{x}) + \frac{\mu \|\mathbf{x} - \bar{\mathbf{x}}\|^2}{2} \right\}$$

This improved model cannot be unbounded below because the function is quadratic, but we need to choose $\mu$ and $\bar{\mathbf{x}}$ wisely. Notice that $\mu$ accounts for the curvature of the parabola, but for wrong choices of mu the interception with the linear model is not placed properly.

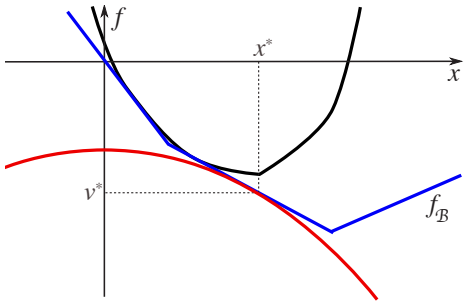



FIGURE 4.29: The interception between the blue line and the quadratic red line is the point where the minimum is placed

A possible way is to move the **stability center** whenever the current value is better the the best encountered so far.

ALGORITHM 4.14.1 Pseudocode for Bundle Method.

```
1:  procedure PBM(f, g, x̄, m_1, ε)
2:      choose μ;
3:      B ← {(x̄, f(x̄), g(x̄))};
4:      while ( true ) do
5:          x* ← arg min{f_B(x) + μ‖x − x̄‖^2/2};
6:          if (μ‖x* − x̄‖_2 ≤ ε) then
7:              break;
8:          end if
9:          if (f(x*) ≤ f(x̄) + m_1(f_B(x*) − f(x̄))) then
10:             x̄ ← x*;                          ▷ Serious Step (SS)
11:             possibly decrease μ;
12:         else
13:             possibly increase μ;              ▷ Null Step (NS)
14:         end if
15:         B ← B ∪ ( x* , f(x*) , g(x*) );
16:     end while
17: end procedure
```

Notice that this algorithm allows not to move from an iteration to the other, but the information computed is used to mode towards a better point.
The bundle method converges in few steps, although each step is quite costly.

We reached a point where for solving an *unconstrained* problem we need to solve a *constrained* one. For this reason, we will introduce in next chapter constrained optimization problems.

# Chapter 5

# Constrained Optimality

## 5.1 Constrained Optimization

In this lecture we address the problem of finding the **optimum** of a function in a subset of its domain, called $X$. The term optimum differs from the minimum, because the optimum in that subset may not be a minimum of the whole function.

$$f_* = \min\{\, f(\mathbf{x}) \;:\; \mathbf{x} \in X \,\}$$

**Definition 5.1.1** (Local optimum)**.** *Given a function $f$ and a constraint set $X$, we denote **local optimum** the point where the function assumes the minimum value inside the set $X$. Formally,* $\min\{f(\mathbf{x}) \;:\; \mathbf{x} \in \mathcal{B}(\mathbf{x}_*, \varepsilon) \cap X\}$ *for some* $\varepsilon > 0$.

Notice that the only points in which the constraint adds some information are the ones on the boundary, as shown in Figure 5.1

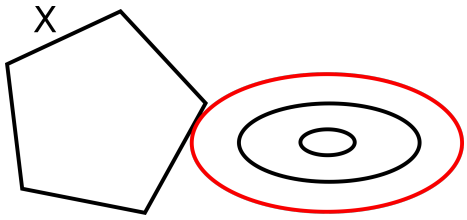



FIGURE 5.1: The red line is level set of the function corresponding to the smallest value that touches the set $X$. The point in the intersection is not a saddle point of the function $f$, although it is the minimum.

There are two kinds of constraints:

FAKE ONES: this first kind is such that the minimum of the function lies in the *interior* of the set $X$, hence there is no need to use the constraints at all and we can impose $\nabla f(\mathbf{x}) = 0$;

REAL ONES: when the optimal is on the boundary. This is the case of linear functions, because the gradient is constant $\nabla f(\mathbf{x}) = c$.

At this point we want to decide if a point on the boundary is an optimum. In this context it is important how the boundary is defined.

### 5.1.1 Linear equality constraints

Le us define a minimum problem subject to linear equality constraints:

$$\min\{f(\mathbf{x}) \ : \ A\mathbf{x} = \mathbf{b}\}$$

where $rk(A)$ accounts for the number of linearly independent rows.
Let us assume that there are not linearly independent rows in $A$, then or this behaviour is reflected in $B$ or the system does not have any solution.
In the case of presence of linearly dependent columns such columns may be eliminated to ease the computation without loss of information.

A linear equality constraint is an hyperplane, as shown in Figure 5.2.

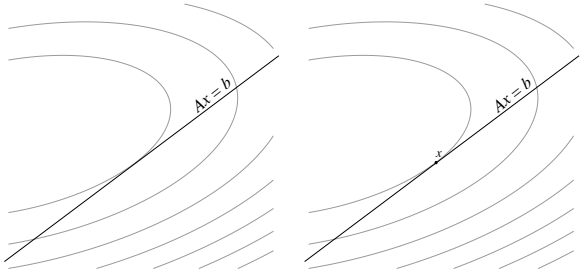



FIGURE 5.2: Linear constraint and a point on the boundary.

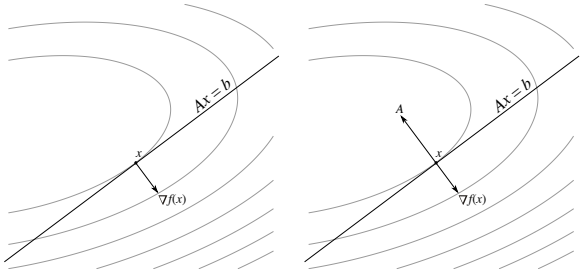



FIGURE 5.3: The gradient is orthogonal to the level set in that point, when the function is smooth. The same holds for matrix $A$.

**Fact 5.1.1.** *Let $A \in M(m, n, \mathbb{R})$ and let $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$, such that $rk(A) = m < n$. Then the $m$ constraints kill exactly $m$ degrees of freedom.*

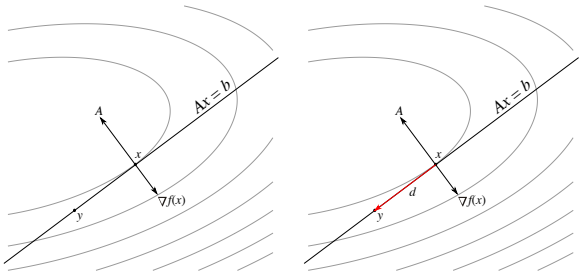



FIGURE 5.4: If we take any other point in the space it has to be orthogonal to $A$. If we take any other point $\mathbf{y}$ along the linear constraint, such point is not better than $\mathbf{x}$, because it is orthogonal to the gradient, hence the directional derivative is 0.

*Proof.* Let us "split" matrix $A$ and vector $\mathbf{x}$ into two parts, as follows:

$$A = \begin{pmatrix} A_B & A_N \end{pmatrix} \ , \ \mathbf{x} = \begin{pmatrix} \mathbf{x_B} \\ \mathbf{x_N} \end{pmatrix}$$

where $A_B \in M(m, \mathbb{R})$, $A_N \in M(m, n-m, \mathbb{R})$, $\mathbf{x_B} \in \mathbb{R}^m$, $\mathbf{x_N} \in \mathbb{R}^{n-m}$ and $det(A_B) \neq 0$. The system becomes

$$\begin{aligned} A_B\mathbf{x_B} + A_N\mathbf{x_N} &= \mathbf{b} \\ (A_B \text{ non singular}) &\Downarrow \\ \mathbf{x_B} + {A_B}^{-1}A_N\mathbf{x_N} &= {A_B}^{-1}\mathbf{b} \\ &\Downarrow \\ \mathbf{x_B} &= {A_B}^{-1}(\mathbf{b} - A_N\mathbf{x_N}) \end{aligned}$$

□

Thanks to Proposition 5.1.1, the original optimization problem can be expressed through an **optimization problem** on a **reduced space**, because $\mathbf{x_B}$ is fully determined once computed $\mathbf{x_N}$. Formally,

$$\min\{r(\mathbf{w}) = f(D\mathbf{w} + \mathbf{d}) : \mathbf{w} \in \mathbb{R}^{n-m}\} \qquad ((\mathrm{R}))$$

where $D = \begin{pmatrix} -A_B^{-1}A_N \\ I_{n-m} \end{pmatrix} \in M(n, n-m, \mathbb{R})$ and $\mathbf{d} = \begin{pmatrix} {A_B}^{-1}\mathbf{b} \\ 0_{n-m} \end{pmatrix}$.

**Fact 5.1.2.** *The gradient of the reduced function $r : \mathbb{R}^{n-m} \to \mathbb{R}^n$ is computed as $\nabla\ r(\mathbf{w}) = D^T \nabla\ f(D\mathbf{w} + \mathbf{d})$.*

*Proof.*

$$\begin{aligned} \frac{\partial r(\mathbf{w})}{\partial \mathbf{w}} &= \frac{\partial f(D\mathbf{w} + \mathbf{d})}{\partial \mathbf{w}} \\ &= \frac{\partial f(D\mathbf{w} + \mathbf{d})}{\partial D\mathbf{w} + \mathbf{d}} \cdot \frac{\partial D\mathbf{w} + \mathbf{d}}{\partial \mathbf{w}} \\ &= D^T \nabla\ f(D\mathbf{w} + \mathbf{d}) \end{aligned}$$

□

The fact that $\mathbf{w}^*$ is an optimum implies that $\nabla\ r(\mathbf{w}^*) = 0$.

$AD = \begin{pmatrix} A_B & A_N \end{pmatrix} \cdot \begin{pmatrix} -A_B^{-1} A_N \\ I_{n-m} \end{pmatrix} = \cancel{A_B} \cdot \cancel{A_B^{-1}} \cdot A_N + A_N = 0.$

Now the point is taking a multiple of matrix $A$, finding a feasible $\mathbf{x}$ and corresponding $\mathbf{w}$ (because there is a bijection), then finding the value $\mu$ that allows the equality.

**Theorem 5.1.3.** *Let $A\mathbf{x} = \mathbf{b}$ be a linear system, where $A \in M(m, n, \mathbb{R})$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$ and let $\mathbf{w} \in \mathbb{R}^{n-m}$ such that $\mathbf{x} = \begin{pmatrix} {A_B}^{-1}(\mathbf{b} - A_N\mathbf{w}) \\ \mathbf{w} \end{pmatrix}$. It is equivalent to find a stationary point $\mathbf{x}$ for the original problem $(P)$ or finding the stationary point $\mathbf{w}$ for $(R)$. Formally, if $\exists\ \mu\ \in\ \mathbb{R}^m$ s.t. $\mu^T A = \nabla\ f(\mathbf{x})$ then $\nabla r(\mathbf{w}) = D^T \nabla\ f(\mathbf{x}) = 0$, see Figure 5.5.*

**Definition 5.1.2** (Poorman's Karush Kuhn-Tucker conditions)**.** *A point is a good candidate for being a minimum of the constrained problem if and only if it satisfies **Poorman's KKT conditions**, namely the problem is feasible and $\exists \mu \in \mathbb{R}^m$ s.t. $\mu^T A = \nabla f(\mathbf{x})$.*

**Theorem 5.1.4.** *Let $f$ be a convex function, then KKT conditions are enough for optimality.*

A very naive explanation of the theorem is that if the function is convex also the restriction is convex and a stationary point of a convex function is a minimum.

Our idea is to characterize the directions we can move along in order to find new points that satisfy the constraint. Formally, $A\mathbf{x} = \mathbf{b}$ is our constraint and we want to move towards $\mathbf{x} + \mathbf{d}$ and stay in the feasible region. How? $A(\mathbf{x} + \mathbf{d}) = \mathbf{b} \Leftrightarrow A\mathbf{x} + A\mathbf{d} = \mathbf{b} \Leftrightarrow A\mathbf{d} = 0$, since $A\mathbf{x} = \mathbf{b}$. The only way to move along the constraint is choosing a direction which scalar product with $A$ is 0, hence 0 scalar product with the gradient.

From now on we would like to study the behaviour on constrained problems where the constraints are not equalities, but inequalities.

In order to do this we need some mathematical background.

### 5.1.2 Background for linear inequality constraints

**Definition 5.1.3** (Tangent cone)**.** *We call **tangent cone** of $X$ at $\mathbf{x}$ $\mathbf{T}_X(\mathbf{x}) = t$.*

$$\left\{ \mathbf{d} \in \mathbb{R}^n\ :\ \exists\ \{\mathbf{z_i} \in X\} \to \mathbf{x}\ \wedge\ \{t_i \geq 0\} \to 0\ s.t.\ \mathbf{d} = \lim_{i \to \infty} \frac{z_i - x}{t_i} \right\}$$

**Theorem 5.1.5.** *Let $\mathcal{C}$ be a cone, $\forall \mathbf{x} \in \mathcal{C}\ \alpha\mathbf{x} \in \mathcal{C},\ \forall \alpha > 0$.*

**Theorem 5.1.6.** *Given a function $f$, where $\mathbf{x}$ is a **local** optimum $< \nabla f(\mathbf{x}), \mathbf{d} > \geq 0\ \forall \mathbf{d} \in T_X(\mathbf{x})$.*

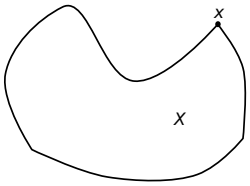



FIGURE 5.5: When the boundary has this shape we can move along directions that point inside the constraints. Tangent directions are not allowed.

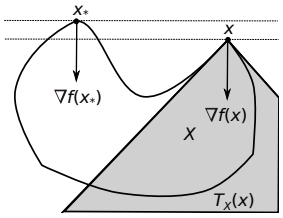



FIGURE 5.6: Geometric representation of tangent cone, which is the region of the space where we can pick the directions. The intuition is to zoom in $x$ and the result of this zooming is a cone. It is clear from this picture that we can only find a *local* optimum.

*Proof.* **Proof by contradiction:** Assume $\exists \mathbf{d} \in T_X(\mathbf{x})$ such that $< \nabla f(\mathbf{x}), \mathbf{d} >< 0$, but $\mathbf{x}$ is a local optimum.

By definition $\exists X \supset \{\mathbf{z_i}\} \to \mathbf{x}$ and $\{t_i\} \to 0$ such that $\mathbf{d} \overset{*}{=} \lim_{i\to\infty} \frac{\mathbf{z_i}-\mathbf{x}}{t_i}$.

First order Taylor $f(\mathbf{z_i}) - f(\mathbf{x}) =< \nabla f(\mathbf{x}), (\mathbf{z_i} - \mathbf{x}) > + R(\mathbf{z_i} - \mathbf{x})$.

$$\begin{aligned}
\lim_{i\to\infty} \frac{f(\mathbf{z_i}-\mathbf{x})}{t_i} &= \lim_{i\to\infty} < \nabla f(\mathbf{x}), \frac{\mathbf{z_i}-\mathbf{x}}{t_i} > + \frac{R(\mathbf{z_i}-\mathbf{x})}{t_i} \\
&= \lim_{i\to\infty} < \nabla f(\mathbf{x}), \frac{\mathbf{z_i}-\mathbf{x}}{t_i} > + \lim_{i\to\infty} \frac{R(\mathbf{z_i}-\mathbf{x})}{t_i} \\
&\overset{*}{=} < \nabla f(\mathbf{x}), \mathbf{d} > + \lim_{i\to\infty} \frac{R(\mathbf{z_i}-\mathbf{x})}{t_i} \\
&\overset{(1)}{=} < \nabla f(\mathbf{x}), \mathbf{d} > \\
&< 0
\end{aligned} \tag{5.1.1}$$

Where, $\overset{(1)}{=}$ follows from the fact that the residual of the Taylor's model goes to 0: $\lim_{i\to\infty} \frac{R(\mathbf{z_i}-\mathbf{x})}{t_i} = 0$. □

**Observation 5.1.1.** *The optimum of Theorem 5.1.6 is global when the function is convex, because in that case $X \subseteq \mathbf{x} + T_X(\mathbf{x})$. For a geometric idea see Figure 5.7.*

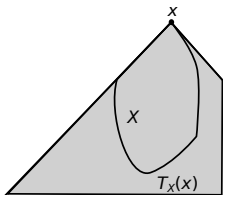



FIGURE 5.7: Convex function.

**Observation 5.1.2.** *Notice that the converse of* **??** *(*$< \nabla f(\mathbf{x}), \mathbf{d} >\geq 0 \ \forall \mathbf{d} \in T_X(\mathbf{x}) \Rightarrow \mathbf{x}$*) does not hold.*

*Let us take* $f : \mathbb{R}^2 \to \mathbb{R}$ *s.t.* $f(\mathbf{x}) = x_2$ *and the minimum problem* $\min\{f(\mathbf{x}) \ : \ x_2 \geq x_1^3\}$. *The constraint looks like Figure 5.8.*

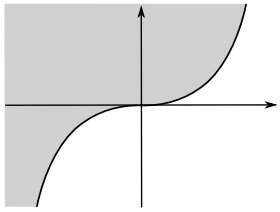


FIGURE 5.8: Shape of the constraint.

*Assume that we are sitting in* $\mathbf{x} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ *such that* $\nabla f(\mathbf{x}) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

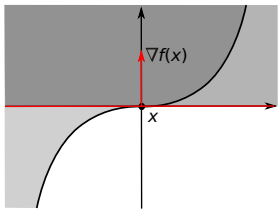



FIGURE 5.9: The gradient and the tangent cone.

*We are lying in a saddle point, so* $\mathbf{x}$ *is not the optimum even if* $< \nabla f(\mathbf{x}), \mathbf{d} >\geq 0, \ \forall \mathbf{d} \in T_X(\mathbf{x})$.

Now we need a more manageable object for $T_X(\mathbf{x})$.

**Definition 5.1.4** (Cone of feasible directions)**.**
*Let* $X$ *be a set, we term* ***feasible cone*** *in a point* $\mathbf{x}$ *the set of all directions such that there exist small but not* 0 *steps such that all the points on this direction are*

*feasible. Formally, a* ***feasible cone*** *is* $F_X(\mathbf{x}) = \{\mathbf{d} \in \mathbb{R}^n \ : \ \exists \bar{\varepsilon} > 0 \ s.t. \ \mathbf{x} + \varepsilon \mathbf{d} \in X, \ \forall \varepsilon \in [0, \bar{\varepsilon}]\}$.

**Fact 5.1.7.** *The properties of such cone are:*

1. $T_X$ *closed,* $F_X$ *in general not (hence the cone of feasible directions is the tangent cone minus the tangent directions);*
2. $cl(F_X) \subseteq T_X$*, where* $cl(F_X)$ *is the closure of the cone of feasible directions;*
3. *If* $X$ *convex then the cones coincide. Formally,* $T_X$ *and* $F_X$ *convex and* $cl(F_X) = T_X$.

**Fact 5.1.8.** *The feasible cone is the of the set of all possible feasible directions.*

We are now interested in finding a better characterization of the cone of feasible directions $F_X$ and we will do so by working on the tangent cone $T_X \subseteq F_X$.

Let us enumerate 4 representations of the feasible region $X$:

First representation:

$$X = \{\mathbf{x} \in \mathbb{R}^n \ : \ g_i(\mathbf{x}) \leq 0 \ \ i \in \mathcal{I}, \ h_j(\mathbf{x}) = 0 \ \ j \in \mathcal{J} \}$$

where $\mathcal{I}$ is the set of inequality constraints and $\mathcal{J}$ is the set of equality constraints;

Second representation:

$$X = \{ \mathbf{x} \in \mathbb{R}^n \ : \ G(\mathbf{x}) \leq 0 \, , \, H(\mathbf{x}) = 0\}$$

where $G = [g_i(\mathbf{x})]_{i \in \mathcal{I}} : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{I}|}$ and $H = [h_i(\mathbf{x})]_{i \in \mathcal{J}} : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{J}|}$;

Third representation: (hiding equalities)

$$X = \{\mathbf{x} \in \mathbb{R}^n : g_i(\mathbf{x}) \leq 0 \ i \in \mathcal{I}, \ h_j(\mathbf{x}) \leq 0 \ \wedge \ h_j(\mathbf{x}) \geq 0 \ j \in \mathcal{J}\}$$

Fourth representation: (hiding inequalities into a single function)

$$X = \{\mathbf{x} \in \mathbb{R}^n \ : \ g(\mathbf{x}) = \max\{ g_i(\mathbf{x}) \ : \ i \in \mathcal{I} \} \leq 0 \ i \in \mathcal{I}, \ h_j(\mathbf{x}) = 0 \ \in \mathcal{J}\}$$

**Definition 5.1.5** (Active constraints)**.** *We term* ***active constraints*** *at* $\mathbf{x} \in X$ *the inequality constraints which are satisfied with the equality. Formally,*

$$\mathcal{A}(\mathbf{x}) = \{i \in \mathcal{I} \ : \ g_i(\mathbf{x}) = 0\} \subseteq \mathcal{I}$$

Let us introduce some useful notation on the subject: let $\mathcal{B} \subseteq \mathcal{I}$ a subset of indices of inequality constraints.
We denote $G_{\mathcal{B}} = [g_i(\mathbf{x})]_{i \in \mathcal{B}} : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{B}|}$ the corresponding set of inequalities.
Notice that when choosing a moving direction we need to pay attention only to active constraints and inequality constraints.

**Definition 5.1.6** (First-order feasible direction cone)**.** *We term* ***first-order feasible direction cone at*** $\mathbf{x} \in X$ *the set of all directions that satisfy equality constraints and active constraints. Formally,*

$$D_X(\mathbf{x}) = \{\mathbf{d} \in \mathbb{R}^n \ : < \nabla g_i(\mathbf{x}), \mathbf{d} >\leq 0 \ i \in \mathcal{A}(\mathbf{x}) \ < \nabla h_j(\mathbf{x}), \mathbf{d} >= 0 \ j \in \mathcal{J}\}$$

*Or equivalently, written in the form of the Jacobian,*

$$D_X(\mathbf{x}) = \{\mathbf{d} \in \mathbb{R}^n : (JG_{\mathcal{A}(\mathbf{x})}(\mathbf{x}))\mathbf{d} \leq 0, \ (JH(\mathbf{x}))\mathbf{d} = 0\}$$

**Example 5.1.1.** *Let us examine the meaning of Definition 5.1.6 through an example. Let us assume that the inequality constraints are the functions* $g_1, g_2$*, shown in Figure 5.10(a) and in Figure 5.10(b) respectively. The directions allowed for satisfying each of the constraints should point towards the gray part, hence have a negative scalar product with both* $\nabla g_1(\mathbf{x})$ *and* $\nabla g_2(\mathbf{x})$*, hence the first-order feasible cone is the one shown in Figure 5.10(d).*

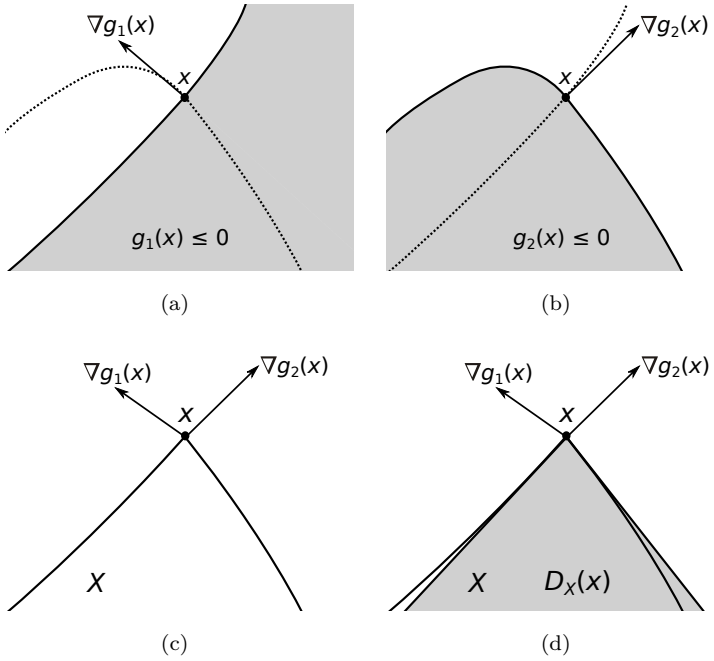



FIGURE 5.10: Constraints and first order feasible direction cone.

**Fact 5.1.9.** *The tangent cone is a subset of the first-order feasible direction cone. Formally,* $T_X(\mathbf{x}) \subseteq D_X(\mathbf{x})$*.*

We would like the first-order feasible direction cone to be exactly equal to the tangent cone and this is almost always true, except for some pathological cases.

**Example 5.1.2.** *Let us have a function $f$ to minimize subject to a linear and a quadratic constraint:*

$$\min\{f(\mathbf{x}) \ : \ {x_1}^2 + (x_2 - 1)^2 - 1 \leq 0, \ x_2 \leq 0\}$$

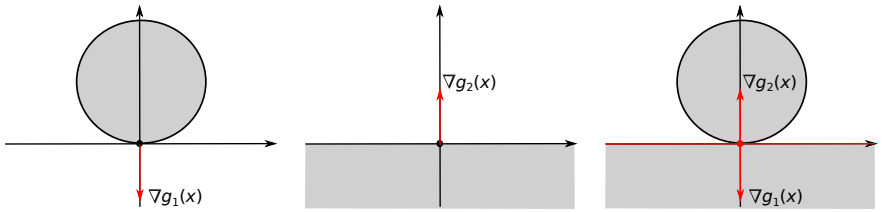



FIGURE 5.11: First-order feasible cone not coinciding with the tangent cone.

*From Figure 5.11 we can see that there are no directions that at the same time point towards the second order constraint and the first order constraint, except from* $\mathbf{0}$*, therefore* $T_X(\mathbf{x}) = \{\mathbf{0}\}$.

*Conversely, all the vectors with* 0 *as second component are included in the first-order feasible direction cone, hence* $D_X(\mathbf{x}) = \left\{ \begin{pmatrix} x_1 \\ 0 \end{pmatrix} \ s.t. \ x_1 \in \mathbb{R} \right\}$.

We would like to ensure we are not in one of these pathological cases and to do this we introduce some conditions.

**Fact 5.1.10.** *The following holds:*

AFFINE CONSTRAINTS (AFFC): *Let $g_i$ and $h_j$ be affine constraints.*

$$\forall \ i \in \mathcal{I} \ \forall \ j \in \mathcal{J} \ T_X(\mathbf{x}) = D_X(\mathbf{x}) \ \forall \mathbf{x} \in X$$

SLATER'S CONDITION (SLAC): *Let $g_i$ convex $\forall \ i \in \mathcal{I}$ and let $h_j$ affine $\forall \ j \in \mathcal{J}$ if it is possible to find a point $\mathbf{x} \in X$ that satisfies "inactively" each inequality constraint, then the first-order feasible cone coincides with the tangent cone. Formally if $\exists \ \bar{x} \in X$ s.t. $g_i(\bar{x}) < 0 \ \forall \ i \in \mathcal{I}$, then $T_X(x) = D_X(x) \ \forall x \in X$;*

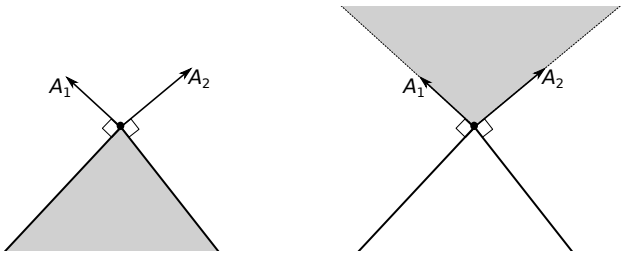
LINEAR INDEPENDENCE (LINI): *A certain point $\bar{x} \in X \ \wedge$ allows the equality between the first-order feasible cone and the tangent cone if the gradient of both the equality and inequality constraints are linearly independent. Formally, if $\{\nabla g_i(\bar{x}) \ : \ i \in \mathcal{A}(\bar{x})\} \cup \{\nabla h_j(\bar{x}) \ : \ j \in \mathcal{J}\}$ are linearly independent, then $T_X(\bar{x}) = D_X(\bar{x})$*

It goes without saying that we cannot check all the directions in order to exclude the nasty pathological cases.

**Definition 5.1.7** (Dual cone)**.** *Let $\mathcal{C}$ be a* ***polyhedral cone*** $\mathcal{C} = \{\mathbf{d} \in \mathbb{R}^n \ : \ A\mathbf{d} \leq 0\}$*, for some* $A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_k \end{pmatrix} \in M(k, n, \mathbb{R})$.

*We term* ***dual cone*** $\mathcal{C}^* = \{\mathbf{c} = \sum_{i=1}^{k} \lambda_i A_i \ : \ \lambda \geq 0\}$.

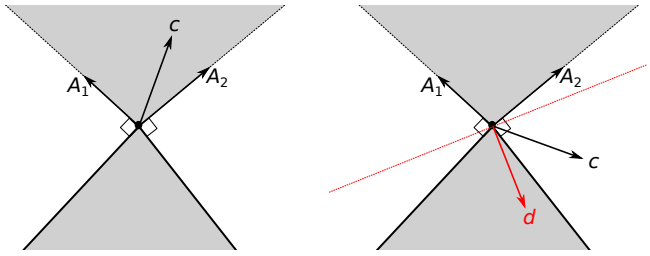
A bi-dimensional example of primal and dual cone is shown in Figure 5.12.




(a) (b)

FIGURE 5.12: Primal and dual cone.

**Lemma 5.1.11** (Farka's lemma)**.** *Any vector $\mathbf{x} \in \mathbb{R}^n$ either belongs to the dual cone or there exists a vector $\mathbf{d}$ in the polyhedral cone which has a negative scalar product with it, see Figure 5.13. Formally, either $\exists\, \lambda \geq 0$ s.t. $\mathbf{c} = \sum_{i=1}^{k} \lambda_i A_i$ or $\exists\, \mathbf{d}$ s.t. $A\mathbf{d} \leq 0 \wedge < \mathbf{c}, \mathbf{d} >> 0$.*




(a) (b)

FIGURE 5.13: Graphical example of Farka's lemma.

**Theorem 5.1.12** (Karush-Kuhn-Tucker Theorem)**.** *Let us assume that we found an optimal solution $\mathbf{x}_*$. Then the anti-gradient in $\mathbf{x}_*$ can be obtained as the linear combination of the gradients of the active inequality constraints and the*

*equality constraints. Formally,* $\exists \lambda \in \mathbb{R}_+^{|\mathcal{I}|}$ *and* $\mu \in \mathbb{R}^{|\mathcal{I}|}$ *such that*

$$\nabla f(\mathbf{x}_*) + \sum_{i \in \mathcal{A}(\mathbf{x}_*)} \lambda_i \nabla g_i(\mathbf{x}_*) + \sum_{j \in \mathcal{J}} \mu_j \nabla h_j(\mathbf{x}_*) = 0$$

Notice that we did not impose $\mu \geq 0$. Let us take an equality constraint $h_j(\mathbf{x}) = 0$. This is equivalent to write $h_j(\mathbf{x}) \leq 0 \ \wedge h_j(\mathbf{x}) \geq 0$, thus leading to two different multipliers, say $\lambda_j^+$ and $\lambda_j^-$. The term of the sum concerning $h_j$ looks like this $\lambda_j^+ \nabla h_j(x_*) - \mu_j^- \nabla h_j(x_*) = (\mu_j^+ - \mu_j^-) \cdot \nabla h_j(x_*)$, where both $\mu_j^+$ and $\mu_j^-$ are $\geq 0$, hence their difference (denoted by $\mu_j$) may be either positive or negative.

**Fact 5.1.13.** *The Karush-Kuhn-Tucker conditions are also written as:*

FEASIBILITY: $\mathbf{x} \in X$ *is equivalent to saying* $g_i(\mathbf{x}) \leq 0 \ \ i \in \mathcal{I}, \ h_j(\mathbf{x}) = 0 \ \ j \in \mathcal{J}$

KKT-G: $\nabla f(\mathbf{x}) + \sum_{i \in \mathcal{I}} \lambda_i \nabla g_i(\mathbf{x}) + \sum_{j \in \mathcal{J}} \mu_j \nabla h_j(\mathbf{x}) = 0$

COMPLEMENTARITY SLACKNESS: $\sum_{i \in \mathcal{I}} \lambda_i g_i(\mathbf{x}) = 0$

Notice that the third condition expresses the fact that the constraint should be active.

**Fact 5.1.14.** *Let (P) be a convex problem such that the inequality and equality constraint satisfy* constraint qualification *(they are affine functions). In this case, if the Karush-Kuhn-Tucker conditions hold and then* $\mathbf{x}$ *is a global optimum.*

At this point, we are left with the task of checking optimality in the case of a **non-convex** function and this is obtained through the theory of *duality*.

## 5.2 Duality

**Definition 5.2.1** (Lagrangian function)**.** *Let (P) be our constrained minimum problem over* $f : \mathbb{R}^n \to \mathbb{R}$*, where* $\mathcal{I}$ *is the set of the inequality constraints and* $\mathcal{J}$ *is the set of the equality constraints. We term* ***Lagrangian function*** *the following:*

$$L(\mathbf{x}, \ \lambda, \ \mu) = f(\mathbf{x}) + \sum_{i \in \mathcal{I}} \lambda_i g_i(\mathbf{x}) + \sum_{j \in \mathcal{I}} \mu_j h_j(\mathbf{x})$$

*where* $\mathbf{x}$ *is the variable and* $\lambda, \mu$ *are parameters.*

**Fact 5.2.1.** *According to KKT-G condition, a necessary condition for optimality is that the gradient of the Lagrangian function is* 0*. Formally,* $\nabla L(\mathbf{x}, \ \lambda, \mu) = 0$.

So far, our algorithm had the following behaviour: suppose to be sitting in a point $\mathbf{x}$, check if the optimality condition holds (from Proposition 5.2.1 the gradient of the Lagrangian should be 0). If $\mathbf{x}$ is not optimal, we should move toward a new and hopefully better point $\mathbf{x}'$.

boh

A first approach to use the Lagrangian function would be taking $\lambda$ and $\mu$ such that the Lagrangian is positive semidefinite.

But this is a too strict requirement, in fact we impose:

**Definition 5.2.2** (Critical cone)**.** *Let us assume $(\mathbf{x}, \lambda, \mu)$ satisfies (KKT). We define the **critical cone** as*

$$C(\mathbf{x},\ \lambda,\ \mu) = \left\{ \mathbf{d} \in \mathbb{R}^n\ :\ \begin{array}{ll} < \nabla g_i(\mathbf{x}), \mathbf{d} >= 0 & i \in \mathcal{A}(\mathbf{x})\ s.t.\ \lambda_i^* > 0 \\ < \nabla g_i(\mathbf{x}), \mathbf{d} >\leq 0 & i \in \mathcal{A}(\mathbf{x})\ s.t.\ \lambda_i^* = 0 \\ < \nabla h_j(\mathbf{x}), \mathbf{d} >= 0 & j \in \mathcal{J} \end{array} \right\}$$

**Theorem 5.2.2.** *Let us assume we have a point $(\mathbf{x},\ \lambda,\ \mu)$ that satisfies the Karush-Kuhn-Tucker conditions and satisfies the linear independence of the constraints. If $\mathbf{x}$ is local optimum then $\mathbf{d}^T \nabla^2_{\mathbf{xx}} L(\mathbf{x},\ \lambda,\ \mu)\mathbf{d} \geq 0\ \forall\ \mathbf{d} \in C(\mathbf{x},\ \lambda,\ \mu)$.*

*Informally, if the hypothesis holds, then the Hessian of the Lagrangian function is $\succeq 0$ on the critical cone.*

**Observation 5.2.1.** *$(x,\ \lambda,\ \mu)$ satisfies (KKT) $\wedge$ $\nabla^2_{\mathbf{xx}} L(\mathbf{x},\ \lambda,\ \mu) \succ 0$ on $C(\mathbf{x},\ \lambda,\ \mu)$ then $\mathbf{x}$ local optimum.*

We would like to say something more about $\lambda$ and $\mu$. Until now we considered the Lagrangian as a function of $x$, but what if we consider the Lagrangian in terms of $\lambda$ and $\mu$?

## 5.3 Lagrangian Duality

**Definition 5.3.1** (Lagrangian relaxation)**.** *Let (P) be our constrained minimum problem over $f : \mathbb{R}^n \to \mathbb{R}$, where $\mathcal{I}$ is the set of the inequality constraints and $\mathcal{J}$ is the set of the equality constraints. Let us consider the Lagrangian function on such problem. We term **Lagrangian relaxation** the function where we fixed $\lambda$ and $\mu$ and minimize on $\mathbf{x}$:*

$$\psi(\lambda,\ \mu) = \min\{L(\mathbf{x},\ \lambda,\ \mu)\ :\ \mathbf{x} \in \mathbb{R}^n\}$$

The relaxation leads to an unconstrained problem and we learnt how to solve one of those.

This Lagrangian relaxation leads to the definition of **lagrangian dual** $\psi$.

**Property 5.3.1.** *The dual function has the following properties:*

1. *$\psi$ is concave, but $\psi(\lambda, \mu) = -\infty$;*
2. *Let $\bar{\mathbf{x}}$ be optimal in $(R_{\lambda,\mu})$, then $[G(\bar{\mathbf{x}}), H(\bar{\mathbf{x}})] \in \partial\psi(\lambda, \mu)$*
3. *$\psi$ is non differentiable, although $f$, $g_i$ and $h_j$ are, but if $\bar{\mathbf{x}}$ is unique then $\psi$ is differentiable and $\nabla\psi(\lambda, \mu) = [G(\bar{\mathbf{x}}), H(\bar{\mathbf{x}})]$*

*4.* $\forall$ *fixed* $\lambda \geq 0, \mu, \bar{x} \in X$ $\psi(\lambda, \mu) = \min_x L(x, \lambda, \mu) \leq L(\bar{x}, \lambda, \mu) \leq f(\bar{x})$

$$\psi(\lambda,\mu) = \begin{pmatrix} \lambda_1 g_1(\bar{x}) \\ \vdots \\ \lambda_k g_k(\bar{x}) \\ \lambda_1 \mu_1(\bar{x}) \\ \vdots \\ \lambda_p \mu_p(\bar{x}) \end{pmatrix} \textit{ is such that } \begin{pmatrix} g_1(\bar{x}) \\ \vdots \\ g_k(\bar{x}) \\ \mu_1(\bar{x}) \\ \vdots \\ \mu_p(\bar{x}) \end{pmatrix} \textit{ belongs to the supergradient.}$$

**Theorem 5.3.2** (Weak duality)**.** $\forall$ *fixed* $\lambda \in \mathbb{R}^+ \cup \{0\}, \mu \in \mathbb{R}$ *such that* $\psi(\lambda,\ \mu) \leq v(P)$ *and let us take any feasible* $\bar{\mathbf{x}} \in X$ *and* $g(\bar{\mathbf{x}}) \leq 0$*,* $h(\bar{\mathbf{x}}) = 0$*.*

$$\psi(\lambda, \mu) = \min_{\mathbf{x}} L(\mathbf{x}, \lambda, \mu) \leq L(\bar{\mathbf{x}}, \lambda, \mu) \leq f(\bar{\mathbf{x}})$$

*Proof.*

$$\begin{aligned} \psi(\bar{\lambda}, \bar{\mu}) &= \min_{\mathbf{x}} L(\mathbf{x}, \lambda, \mu) \\ &= \min_{\mathbf{x}} \Big( f(\mathbf{x}) + \sum_i \lambda_i g_i(\mathbf{x}) + \sum_j \mu_j h_j(\mathbf{x}) \Big) \\ &\overset{(*)}{\leq} f(\bar{\mathbf{x}}) + \underbrace{\sum_i \bar{\lambda}_i g_i(\bar{\mathbf{x}})}_{\leq 0} + \underbrace{\sum_j \bar{\mu_j} h_j(\bar{\mathbf{x}})}_{0} \\ &\leq f(\bar{\mathbf{x}}) \end{aligned}$$

where $\overset{(*)}{\leq}$ holds because $\bar{\mathbf{x}}$ is feasible. □

Thanks to Theorem 5.3.2, we obtained a lower bound for the optimum value of the function, which means that we can use it as a stopping criterion. Formally, let us assume we are sitting in a feasible point $\bar{\lambda}$, $\bar{\mu}$ and $\bar{\mathbf{x}}$ then $\psi(\bar{\lambda}, \bar{\mu}) \leq f(\bar{\mathbf{x}}) \leq \psi(\bar{\lambda}, \bar{\mu}) + \varepsilon$, hence we are far from the optimum of a factor $\varepsilon$.

In general, when dealing with lower-bounds, we are interested in maximizing those, so that we have a more precise estimate of the real minimum, hence the need of maximizing the Lagrangian relaxation, which is concave (although it is not smooth). Formally,

$$(D) \quad \max\{\, \psi(\, \lambda\, ,\, \mu\,) \ :\ \lambda \in \mathbb{R}_+^{|\mathcal{I}|} \ ,\ \ \mu \in \mathbb{R}^{|\mathcal{J}|} \,\}$$

Notice that the constraints on $\lambda$ are very easy, so the problem may be considered somehow unconstrained.
We can use local methods to compute the maximum of this function.
If we are able to compute the gradient of $\psi$ then we have the super-gradients of the function $f$.

Is (P) equal to (D)? Yes, provided that everything is convex. In general the Lagrangian gives a lower-bound, hence the maximum of the dual problem is a lower-bound of the minimum of the primal problem. Formally,

$$v(D) \leq v(P)$$

**Example 5.3.1.** *Let us take a concave objective function* $\min\{-x^2 \ : \ 0 \leq x \leq 1\}$*, such that its Lagrangian function is* $L(x, \lambda) = -\mathbf{x}^2 + \lambda_1(x-1) - \lambda_2 x$*. It goes without saying that the Lagrangian function is unbounded below (upside-down parabola).*

*Formally,* $\psi(\lambda) = \min_{x \in \mathbb{R}} L(x, \lambda) = -\infty, \ \forall \lambda \in \mathbb{R}^2$*, which means that the Lagrangian dual is* $v(D) = -\infty < v(P) = -1$*.*

**Theorem 5.3.3.** *Let us assume that* $f$*,* $g$ *and* $h$ *are convex, and constraints qualification holds. If the problem has an optimum in* $\mathbf{x}_*$ *then the value of the primal and the value of the dual are the same. Formally,*

$$T_X(\mathbf{x}_*)D_X(\mathbf{x}_*) \Rightarrow v(D) = v(P)$$

*Proof.* Since $\mathbf{x}_*$ is an optimum, the necessary KKT conditions hold, then we can find $(\lambda^*, \mu^*)$ that satisfy the KKT with $\mathbf{x}$.

Then our claim is that $(\lambda^*, \mu^*)$ is an optimal solution of the dual (D) and the value of the dual is equal to the value of the primal.

$\mathbf{x}_*$ is a stationary point for the Lagrangian function, since it satisfies the KKT conditions then $\mathbf{x}_*$ is exactly the minimum, since the Lagrangian function is convex.

Hence, the value of the dual $v(D) \geq \psi(\lambda - *, \mu_*) = L(\mathbf{x}_*, \lambda_*, \mu_*) = f(\mathbf{x}_*) = v(P) \geq v(D)$. □

How many solution may $\psi(\lambda, \mu)$ have? In principle many, but if $f$ is strongly convex, than everything in the Lagrangian is strongly convex, then he solution of $\psi$ is unique and it is also differentiable, but typically not twice differentiable. At this point, the Lagrangian dual has a single solution and the optimum of the Lagrangian dual corresponds to an optimum for $f$.

We only translated a minimum problem on convex functions to another minimum problem on other convex functions. We will see soon that in some cases this approach is advantageous in others it is not.

## 5.4 Specialized Dual

### 5.4.1 Linear programs

Let us define a *linear* constrained program as

$$(P) \quad \min\{\, \mathbf{c}^T\mathbf{x} \ : \ A\mathbf{x} \geq b \,\}$$

For this problem, the Lagrangian function is defined as $L(\mathbf{x}, \lambda) = \mathbf{c}^T\mathbf{x} + \lambda(\mathbf{b} - A\mathbf{x}) = \lambda\mathbf{b} + (\mathbf{c} - \lambda A)\mathbf{x}$, while the dual function is

$$\psi(\,\lambda\,) = \min_{x \in \mathbb{R}^n} L(\,x\,,\,\lambda\,) = \begin{cases} -\infty & \text{if } c - \lambda A \neq 0 \\ \lambda b & \text{if } c - \lambda A = 0 \end{cases}$$

hence the dual problem is

$$(D) \quad \max \{ \psi(\lambda) : \lambda \geq 0 \} \equiv \max \{ \lambda \mathbf{b} : \lambda A = \mathbf{c}, \lambda \geq 0 \}$$

Since the Lagrangian is far from having an unique optimal solution the dual does not have a unique maximum so $v(D) \neq v(P)$.

### 5.4.2 Quadratic programs

Let us define a *quadratic* constrained program as

$$(P) \quad \min \{ \frac{1}{2}\|\mathbf{x}\|_2^2 : A\mathbf{x} = \mathbf{b} \}$$

expressed in the form of linear least-norm. For this problem, the Lagrangian function is defined as

$$L(\mathbf{x}, \mu) = \frac{1}{2}\|\mathbf{x}\|_2^2 + \mu(A\mathbf{x} - \mathbf{b})$$

which is strictly convex and the stationary point is $\mathbf{x} = -\mu A$, such that $\nabla_{\mathbf{x}} L = \mathbf{x} + \mu A = 0$.

The dual function is

$$\psi(\mu) = \min_{\mathbf{x} \in \mathbb{R}^n} L(\mathbf{x}, \mu) = L(-\mu A, \mu) = -\frac{1}{2}\mu^T (AA^T)\mu - \mu \mathbf{b}$$

and the Lagrangian dual

$$(D) \quad \max\{-\frac{1}{2}\mu^T (AA^T)\mu - \mu b : \mu \in \mathbb{R}^m\}$$

A more general form of a quadratic problem is the following

$$(P) \quad \min \{ \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T \mathbf{x} : A\mathbf{x} \geq \mathbf{b} \}$$

where $Q \succ 0$, otherwise it is likely not to have a minimum. the dual problem is

$$(D) \quad \max \{ \lambda \mathbf{b} - \frac{1}{2}\mathbf{v}^T Q^{-1} \mathbf{v} : \lambda A - \mathbf{v} = \mathbf{q}, \lambda \geq 0 \}$$

In the case of strong strong duality $\equiv v(P) = v(D)$ (almost) always holds.

Solving the KKT system means solving the following

$$\begin{pmatrix} Q & A^T \\ A & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x} \\ \mu \end{pmatrix} = \begin{pmatrix} -\mathbf{q} \\ \mathbf{b} \end{pmatrix}$$

where the matrix is symmetric, but indefinite and its structure allows to employ linear algebra techniques such as *indefinite factorization* and *GMRES* (or other Krylov-type iterative methods). The details on how to solve a quadratic problem with linear equality constraints is discussed in Section 6.1.

### 5.4.3 Conic programs

We can do duals of things that are not quadratic programs. Sometimes we want to have non linear things to be able to draw different shapes.

**Definition 5.4.1** (Conic program)**.** *We term* ***conic program*** $\min\{cx : Ax \geq_K b\}$, *where* $x \geq_K y \equiv x - y \in K$, *where* $K$ *is a convex cone.*

**Example 5.4.1.** *It is easy to see that the* $\geq$ *constraints we saw before are a special case of conic program, where the conic is* $\mathbb{R}^n_+$.

$-x_1 - 2x_2 \leq 2$

$x_1 + x_2 \geq 1$

$2x_1 - x_2 \geq 0$

*Let us write* $-x_1 - 2x_2 - 2 = s_0$, $x_1 + x_2 - 1 = s_1$ *and* $2x_1 - x_2 = s_2$, *such that* $s_0, s_1, s_2 \geq 0$.

*We have a mapping from* $R^n$ *(where the variables live) to* $\mathbb{R}^m$ *(where the constraints live) and* $Ax - b \in K$.

*Rather than writing* $s_0, s_1, s_2 \geq 0$ *we could write* $s_0 \geq \sqrt{s_1^2 + s_2^2}$ *and this would still be a conic program.*

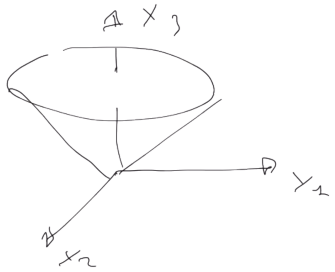


FIGURE 5.14: $x^3 \geq \sqrt{x_1^2 + x_2^2}$ is a convex cone.

The idea is to hide the non linear part in the cone, in particular in $\geq_K$. Let us see what happens to cones depending on the function.

There are three interesting cones:

- $K = \mathbb{R}^n_+ \equiv$ sign constraints $\equiv$ Linear Program;
- $K = \mathbb{L} = \{ x \in \mathbb{R}^n : x_n \geq \sqrt{\sum_{i=1}^{n-1} x_i^2} \} \equiv$ Second-Order Cone (or Lorentz cone) Program, that generalizes linear programs;
- $K = \mathbb{S}_+ = \{ A \succeq 0 \} \equiv$ "$\succeq$" constraints $\equiv$ SemiDefinite Program.

Given the problem that looks like linear except for the cone.

**Definition 5.4.2** (Conic dual)**.** *Conic dual: (D)* $\max\{ yb : yA = c\,,\, y \geq_{K^D} 0 \}$, *where* $K^D = \{ z :< z, x >\geq 0 \ \ \forall\, x \in K \}$ *is a dual cone.*

Sometimes constraints qualification does not hold, since the conic program is not linear sometimes.

Explicit form of second order cone program (SOCP) ("explicit data" $D_i, d_i, p_i, q_i$) $\min\{cx \ : \ \|D_i x - d_i\|_2 \leq p_i x - q_i \ i = 1, \ldots, m\}$.

It collapses to a linear program for $D_i = 0$ and $d_i = 0$.

It turns out that the dual has just this explicit form:

$$\max\{\sum_{i=1}^{m} \lambda_i d_i + \nu_i q_i \ : \ \sum_{i=1}^{m} \lambda_i D_i + \nu_i p_i = c, \ \|\lambda_i\|_2 \leq \nu_i, \ i = 1, \ldots, m\}$$

where the $nu_i$ are the dual variables of the rightmost part, while the $\lambda_i$ are the dual variables of the leftmost part.

We can write the explicit form of semidefinite problems as $\min\{cx \ : \ \sum_{i=1}^{n} x_i A^i \succeq B\}$, where $A^i, B \in M(k, \mathbb{R})$.

There is an even more general form of duality, that we are going to introduce next.

## 5.5 Fenchel's Duality

**Definition 5.5.1** (Fenchel's conjugate)**.** *We denote* ***Fenchel's conjugate*** *of* $f$ $f^*(z) = sup_x\{zx - f(x)\}$.

We can observe that $f^*$ is always convex, even if $f$ is not and it is closed if $f$ is.

The following functions are such that $f^*$ can be computed easily:

1. $f(x) = \frac{1}{2}\|x\|_2^2 \Longrightarrow f^*(z) = \frac{1}{2}\|z\|_2^2$ (only function s.t. $f^* = f$);
2. $(\|\cdot\|_1)^*(z) = ß_{\mathcal{B}_\infty(0,1)}(z)$, $(\|\cdot\|_\infty)^*(z) = ß_{\mathcal{B}_1(0,1)}(z)$;
3. $f(x) = \max\{g_i x - \alpha_i \ i \in I\} \Longrightarrow f^*(z) = \min\{\sum_{i\in I} \alpha_i\theta_i \ : \ \sum_{i\in I} g_i\theta_i = z, \ \sum_{i\in I}\theta_i = 1 \ , \ \theta_i \geq 0 \ i \in I\}$.

**Fact 5.5.1.** *If we are minimizing the sum of two convex functions (P)* $\min\{f(x)+g(x)\}$*, this problem is the same of maximizing the sum of "almost" the two conjugates: (D)* $-\min\{f^*(z) + g^*(-z)\}$.

# Chapter 6

# Constrained Optimization

## 6.1 Quadratic Problem with Linear Equality Constraints

Let $A$ be a matrix in $M(m, n, \mathbb{R})$, where $m < n$ (otherwise the system is either fully determined or impossible) and $rk(A) = m$. The constrained quadratic problem may be written as

$$\min \left\{ \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T\mathbf{x} \ : \ A\mathbf{x} = \mathbf{b} \right\}$$

where $Q \succeq 0$.

A way to solve this problem is through Karush-Kuhn-Tucker system:

$$\begin{matrix} \text{(a)} \\ \text{(b)} \end{matrix} \begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\mu} \end{bmatrix} = \begin{bmatrix} -\mathbf{q} \\ \mathbf{b} \end{bmatrix} \tag{6.1.1}$$

where the first row (a) says that the gradient is a linear combination of the normals of the gradient of the constraints and the second one is just feasibility.

Everything is linear here. This system is symmetric, although indefinite, because it contains many 0s.

We are left with solving the KKT system:

REDUCED KKT (**Good for few constraints** - $m$ small) in this method we first add the hypothesis of non-singularity to the matrix $Q$ so we get the following

$$\begin{cases} Q\mathbf{x} + A^T\boldsymbol{\mu} = -\mathbf{q} & \text{(a)} \\ A\mathbf{x} = \mathbf{b} & \text{(b)} \end{cases}$$

Multiply (a) by $AQ^{-1}$

$$\begin{cases} \cancel{AQ^{-1}Q}\mathbf{x} + AQ^{-1}A^T\boldsymbol{\mu} = -AQ^{-1}\mathbf{q} & \text{(a)} \\ A\mathbf{x} = \mathbf{b} & \text{(b)} \end{cases}$$

Multiply (b) by $-1$ and add to it (a)

$$\begin{cases} A\mathbf{x} + AQ^{-1}A^T\boldsymbol{\mu} = -AQ^{-1}\mathbf{q} & \text{(a)} \\ \cancel{A\mathbf{x}} + AQ^{-1}A^T\boldsymbol{\mu} - \cancel{A\mathbf{x}} = -AQ^{-1}\mathbf{q} - \mathbf{b} & \text{(b)} \end{cases}$$

Multiply (a) by $A^{-1}$

$$\begin{cases} \mathbf{x} + Q^{-1}A^T\boldsymbol{\mu} = -Q^{-1}\mathbf{q} & \text{(a)} \\ AQ^{-1}A^T\boldsymbol{\mu} = -AQ^{-1}\mathbf{q} - \mathbf{b} & \text{(b)} \end{cases}$$

Isolate $\mathbf{x}$

$$\begin{cases} \mathbf{x} = -Q^{-1}(A^T\boldsymbol{\mu} + \mathbf{q}) & \text{(a)} \\ AQ^{-1}A^T\boldsymbol{\mu} = -AQ^{-1}\mathbf{q} - \mathbf{b} & \text{(b)} \end{cases}$$

Notice that $0 \preceq AQ^{-1}A^T = M \in M(m, \mathbb{R})$ and may be much smaller than the original one, since its size depends on the number of constraints. The issue here is that the matrix $M$ is less sparse than both $A$ and $Q$.

NULL SPACE METHOD: (**Good for number of constraints close to the number of variables** - $m \approx n$) In this method we need no assumption on $Q$. Let us resort the method we used in Proposition 5.1.1 on constrained optimization. We can "split" matrix $A$ and vector $\mathbf{x}$ into two parts, as follows:

$$A = \begin{pmatrix} A_B & A_N \end{pmatrix}, \ \mathbf{x} = \begin{pmatrix} \mathbf{x_B} \\ \mathbf{x_N} \end{pmatrix}$$

where $A_B \in M(m, \mathbb{R})$, $A_N \in M(m, n-m, \mathbb{R})$, $\mathbf{x_B} \in \mathbb{R}^m$, $\mathbf{x_N} \in \mathbb{R}^{n-m}$ and $det(A_B) \neq 0$.

By replacing $A$ and $\mathbf{x}$ in the equality constraints definition, we get $A_B\mathbf{x_B} + A_N\mathbf{x_N} = \mathbf{b} \iff \mathbf{x_B} = {A_B}^{-1} \cdot (\mathbf{b} - A_N\mathbf{x_N})$, hence resulting in $\mathbf{x} = D\mathbf{x_N} + \mathbf{d}$, where

$$\mathbf{d} = \begin{pmatrix} \mathbf{b} \\ \mathbf{0}_{n-m} \end{pmatrix}, \ D = \begin{pmatrix} -{A_B}^{-1}A_N \\ I_{n-m} \end{pmatrix} \in M(m, n-m)$$

Notice that $D$ is a basis of the **null space** (or kernel) of $A$ and it is not mandatory to build it like we did above, it is only important to obtain a basis of the kernel.

Let us multiply (a) by $D^T$ and obtain

$$\begin{aligned} D^T(Q\mathbf{x} + A^T\boldsymbol{\mu}) &= -D^T\mathbf{q} \\ D^TQ\mathbf{x} + D^TA^T\boldsymbol{\mu} &= -D^T\mathbf{q} \\ D^TQ\mathbf{x} + \cancel{(AD)^T\boldsymbol{\mu}} &= -D^T\mathbf{q} \\ D^TQ(D\mathbf{x_N} + \mathbf{d}) &= -D^T\mathbf{q} \end{aligned} \tag{6.1.2}$$

Where in the last step we applied the definition $\mathbf{x} = D\mathbf{x_N} + \mathbf{d}$ and hence, $(D^TQD)\mathbf{x_N} = -D^T(Q\mathbf{d} + \mathbf{q})$.

We term $H = D^TQD \in M(n-m, \mathbb{R})$ the reduced Hessian of the problem and notice that whenever the number of constraints is close to the number of variables this matrix is very small.

It is important to note that in order to solve an equality constrained problem we can choose either the reduced KKT method or the null space method, depending on the structure of our problem.

## 6.2 Quadratic Problem with Linear Inequality Constraints

Let $A$ be a matrix in $M(m, n, \mathbb{R})$, where $m < n$ (otherwise the system is either fully determined or impossible) and $rk(A) = m$. The inequality-constrained problem may be written as

$$(P)\min\{f(\mathbf{x}) \ : \ A\mathbf{x} \leq \mathbf{b}\}$$

In general, it is possible that there is no solution for $A\mathbf{x} \leq \mathbf{b}$, hence the problem is empty. In the rest of the course we will consider only problems that have a solution.

### 6.2.1 Projected gradient method

Let us suppose that we are sitting in a point $\mathbf{x}$ such that there are two active constraints $A_1$ and $A_2$, as displayed in Figure 6.1(a) and the anti-gradient points inside the feasible cone $D_X$. In this case, it is irrelevant to take into account constraints $A_1$ and $A_2$.

A completely different scenario would be the one in which the anti-gradient points outside of the cone of feasible directions given by the two constraints $A_1$ and $A_2$, as displayed in Figure 6.1(b). It is in this case that we choose as moving direction the projection of the anti-gradient onto one of the constraints.

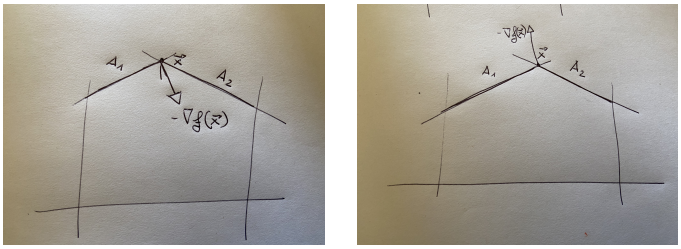
Among the multiple admissible directions, shown in Figure 6.2, we pick the closest to the anti-gradient, which is the green arrow pointing to the right.

The rationale is to pick the direction that minimizes the norm of the difference with the gradient. Formally

$$\mathbf{d} = argmin\{\|\nabla f(\mathbf{x}) - \mathbf{d}\|^2 : \mathbf{d} \in \mathcal{D}_X(\mathbf{x})\}$$

where $\mathcal{D}_X(\mathbf{x})$ is the cone of feasible directions, defined as $\mathcal{D}_X(\mathbf{x}) = \{\mathbf{d} \in \mathbb{R}^n \ : \ A_{\mathcal{A}(\mathbf{x})}\mathbf{d} \leq 0\}$, where $\mathcal{A}(\mathbf{x})$ is the set of all the active constraints. Notice that if the difference between the gradient and the direction is 0 it means that we can stop, since the descent direction would bring us outside the feasible set.

From now on, for sake of clarity, let us denote $\bar{A} = A_{\mathcal{A}(\mathbf{x})}$.




(a) (b)

FIGURE 6.1: Anti-gradient pointing inside and outside the cone of feasible directions.

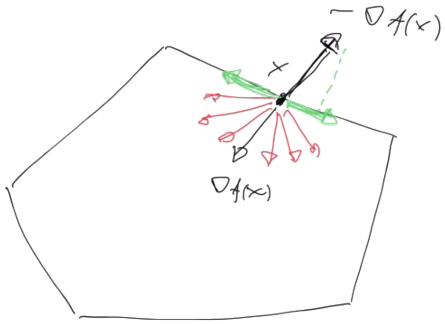


FIGURE 6.2: How to choose the direction in case of $\mathbf{x}$ lying on the boundary and the anti-gradient of $f$ pointing outside the cone of feasible directions.

At this point we are ready to project the problem in such a way that inequality constraints become equality constraints:

$$\min \left\{ \frac{1}{2} \|\nabla f(\mathbf{x}) + \mathbf{d}\|^2 = \frac{1}{2} \mathbf{d}^T I \mathbf{d} + \nabla f(\mathbf{x}) \mathbf{d} \ : \ \bar{A}\mathbf{d} = 0 \right\}$$

We discussed how to solve a quadratic problem with equality constraints in the previous section, in this case the Hessian is the identity matrix $I \succ 0$, therefore we can use the *reduced KKT method*:

$$\begin{cases} Q\mathbf{x} + \bar{A}^T \boldsymbol{\mu} = -\mathbf{q} \\ \bar{A}\mathbf{d} = 0 \end{cases}$$

$(\mathbf{x} = \mathbf{d},\ Q = I,\ \mathbf{b} = \nabla f(\mathbf{x})\mathbf{d})$

$$\begin{cases} \mathbf{d} + \bar{A}^T \boldsymbol{\mu} = -\nabla f(\mathbf{x}) \\ \bar{A}\mathbf{d} = 0 \end{cases}$$

$$\begin{cases} \mathbf{d} = -\nabla f(\mathbf{x}) - \bar{A}^T \boldsymbol{\mu} \\ \bar{A}\mathbf{d} = 0 \end{cases}$$

$$\begin{cases} \cancel{\bar{A}}\mathbf{d} = -\bar{A}\nabla f(\mathbf{x}) - \bar{A}\bar{A}^T \boldsymbol{\mu} \\ \bar{A}\mathbf{d} = 0 \end{cases}$$

$$\begin{cases} \bar{A}\bar{A}^T \boldsymbol{\mu} = -\bar{A}\nabla f(\mathbf{x}) \\ \bar{A}\mathbf{d} = 0 \end{cases}$$

Therefore, we need to solve a system in $\boldsymbol{\mu}$ and then we have the direction. If the number of active constraints is small, solving the system is very fast.

**Fact 6.2.1.** *We can restrict to solve*

$$\begin{cases} \boldsymbol{\mu} = -(\bar{A}\bar{A}^T)^{-1}\bar{A}\nabla f(\mathbf{x}) \\ \mathbf{d} = (I - \bar{A}^T(\bar{A}\bar{A}^T)^{-1}\bar{A})(-\nabla f(\mathbf{x})) \end{cases}$$

*Proof.* $\bar{A}$ has full row rank, then is non singular and may be inverted.

$$\begin{cases} \boldsymbol{\mu} = -(\bar{A}\bar{A}^T)^{-1}\bar{A}\nabla f(\mathbf{x}) \\ \mathbf{d} = -\nabla f(\mathbf{x}) + \bar{A}^T(\bar{A}\bar{A}^T)^{-1}\bar{A}\nabla f(\mathbf{x}) \end{cases} \iff \begin{cases} \boldsymbol{\mu} = -(\bar{A}\bar{A}^T)^{-1}\bar{A}\nabla f(\mathbf{x}) \\ \mathbf{d} = (I - \bar{A}^T(\bar{A}\bar{A}^T)^{-1}\bar{A})(-\nabla f(\mathbf{x})) \end{cases}$$

□

The rationale behind the algorithm is that if the matrix of active constraints $\bar{A}$ contains some linearly dependent rows; such rows are dropped in order to obtain an invertible matrix $A_B$, where $B \subseteq \mathcal{A}(\mathbf{x})$ such that $A_B$ is invertible.

Moreover, we need a vector $\boldsymbol{\mu}_B$ such that all of its components are greater or equal than zero; if there is a component (say $\mu_i \in \mathbb{R}$) that is smaller than zero the set $B$ of linearly independent active constraints is changed accordingly.

This procedure is formalized in Algorithm 6.2.1, where at line 14 the step-size is taken as the minimum step size among the ones that satisfy some subsets of the constraints.

ALGORITHM 6.2.1 Pseudocode for **P**rojected **G**radient **M**ethod for quadratic functions.

```
1: procedure PGM(f, A, b, x, ε)
2:     for (; ;) do
3:         B ← maximal ⊆ A(x) s.t. rank(A_B) = |B|;
4:         for (; ;) do
5:             d ← (I − A_B^T (A_B A_B^T)^{-1} A_B)(−∇f(x));
6:             if < ∇f(x), d > ≤ ε then
7:                 μ_B ← −(A_B A_B^T)^{-1} A_B ∇f(x);
8:                 μ_i ← 0 ∀ i ∉ B;
9:                 if μ_B ≥ 0 then return
10:                end if
11:                h ← min{i ∈ B : μ_i < 0};
12:                B ← B\{ h };
13:                continue;
14:            end if
15:            ᾱ ← min{α_i = (b_i − A_i x)/A_i d : A_i d > 0, i ∉ B};
16:            x ← x + αd;
17:            if ᾱ > 0 then
18:                break;
19:            end if
20:            k ← min{i ∉ B : A_i d > 0 : α_i = 0};
21:            B ← B ∪ { k };
22:        end for
23:        α ← Line_Search(f, x, d, ᾱ);
24:        x ← x + αd;
25:    end for
26: end procedure
```

**Fact 6.2.2.** *The following holds:*

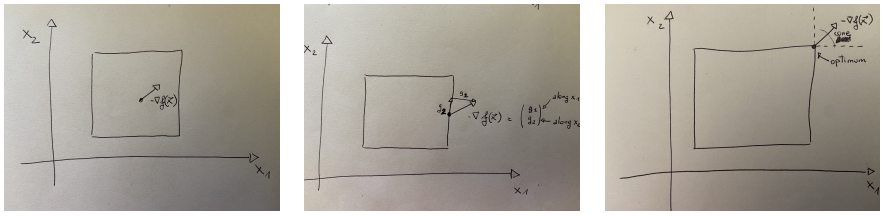
1. *If* $\mathbf{d} = \mathbf{0}$ *and* $\boldsymbol{\mu} \geq \mathbf{0}$ *then* $\mathbf{x}$ *is optimal (from KKT);*

2. *If* $\mathbf{d} = \mathbf{0} \wedge \exists h \in B$ *s.t.* $\mu_h < 0$ *then* $\exists \mathbf{x}' \in \{ \mathbf{x} \in \mathbb{R}^n : A_{B\setminus\{h\}}\mathbf{x} = \mathbf{b}_{B\setminus\{h\}}, A_h\mathbf{x} \leq \mathbf{b}_{B\setminus\{h\}} \}$ *s.t.* $f(\mathbf{x}') < f(\mathbf{x})$. *In other words, if we remove* $h$ *from* $B$*, next time we will have a descent direction;*

3. *Let* $\mathbf{d} \neq \mathbf{0}$ *be a descent direction and let* $H := I - A_B{}^T[A_B A_B{}^T]^{-1}A_B$ *be symmetric and idempotent* ($HH = H^T H = H$)*, then* $< \mathbf{d}, \nabla f(\mathbf{x}) >< 0$.

*Proof.*

trivial;



(a) (b) (c)

FIGURE 6.3: Anti-gradient pointing inside and outside the cone of feasible directions.

2. $\exists \mathbf{d}$ s.t. $A_{B\setminus\{h\}}\mathbf{d} = 0 \wedge A_h\mathbf{d} < 0 \implies < \nabla f(\mathbf{x}), \mathbf{d} >=< -\boldsymbol{\mu} A_B, \mathbf{d} >= -\boldsymbol{\mu}_h A_h \mathbf{d} < 0$;

3.
$$\begin{aligned} < \mathbf{d}, \nabla f(\mathbf{x}) > &=< -H\nabla f(\mathbf{x}), \nabla f(\mathbf{x}) > \\ &= -(H\nabla f(\mathbf{x}))^T \nabla f(\mathbf{x}) \\ &= -\nabla f(\mathbf{x})^T H^T \nabla f(\mathbf{x}) \\ &= -\nabla f(\mathbf{x})^T H^T H \nabla f(\mathbf{x}) \\ &= -(H\nabla f(\mathbf{x}))^T H \nabla f(\mathbf{x}) \\ &< 0 \end{aligned}$$

□

Once we found the optimal face we move inside that face and we are dealing with a steepest descent, which is slow.
At a certain point of the execution the set $B$ of the active constraints will stabilize and become the one of the optimal solution.
At this point the smart thing to do is to use the KKT conditions, since we know the set of active constraints. This idea is pursued in Section 6.2.3.

### 6.2.2 Projected gradient method with box constraints

Let us assume that we have a minimum problem expressed as

$$\text{(P)} \ \min \left\{ f(\mathbf{x}) \ : \ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \right\}$$

where the constraints form a box. If we lie inside the box, any descent direction $-\nabla f(\mathbf{x})$ is acceptable. Conversely, if we are on the boundary, we could move along the direction in which the constraint is satisfied, as shown in **??**.

Algorithm 6.2.2 starts by picking a feasible solution $\mathbf{x}$ between the lower and the upper bound (say the point in the middle) and then proceed by setting to 0 the component of the direction which would bring us to go outside the feasible region.

ALGORITHM 6.2.2 Pseudocode for **P**rojected **G**radient **M**ethod for quadratic functions in the case of **B**ox **C**onstraints.

```
1: procedure PGMBC(f, l, u, x, ε)
2:     d = −∇f(x);
3:     ᾱ = ∞;
4:     for (i = 1 . . . n s.t. d_i ≠ 0) do
5:         if (d_i < 0) then
6:             if (x_i = l_i) then
7:                 d_i ← 0;
8:             else
9:                 ᾱ ← min{ᾱ(x_i − l_i)/d_i};
10:            end if
11:        else
12:            if (x_i = u_i) then
13:                d_i ← 0;
14:            else
15:                ᾱ ← min{ᾱ(u_i − x_i)/d_i};
16:            end if
17:        end if
18:        if (< ∇f(x), d >≤ ε) then return
19:        end if
20:        α ← Line_Search(f, x, d, ᾱ);
21:        x ← x + αd;
22:    end for
23: end procedure
```

Notice that we can assume that $l_i < u_i \ \forall i$, because otherwise that component would be fixed.
This algorithm is a modification of the gradient method, hence it could be very small. For example, if the box is very large and the optimum lies in the middle of it this algorithm is exactly the same of the gradient method.

### 6.2.3 Active-set method for quadratic programs

Let us be given the following quadratic minimum problem

$$\min \left\{ \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^\mathbf{T}\mathbf{x} \ : \ A\mathbf{x} \leq \mathbf{b} \right\}$$

where we know $\mathcal{A}(\mathbf{x}_*)$.

The **active set method** works starting from a certain point $\mathbf{x}$, such that the constraints are satisfied as equalities. According to KKT conditions, a solution $\bar{\mathbf{x}}$ is optimal if $\bar{\mathbf{x}}$ is feasible and $\boldsymbol{\mu} \geq 0$. Otherwise, if $\boldsymbol{\mu}$ is not positive we eliminate the corresponding constraint from the active set and start again. In the case of $\mathbf{x}$ unfeasible, we know the descent direction, we only need to revise the step size.

All this machinery is formalized in Algorithm 6.2.3.

ALGORITHM 6.2.3 Pseudocode for **A**ctive **S**et **M**ethod for **Q**uadratic **P**rograms.

```
1: procedure ASMQP(Q, q, A, b, x, ε)
2:     for (B ← A(x); ;) do
3:         solve (P_B) min{½ x^T Q x + q^T x : A_B x = b_B} for
   (x̄, μ̄_B);
4:         if (A_i x̄ ≤ b_i ∀ i ∉ B) then
5:             if (μ_B ≥ 0) then return
6:             end if
7:             h ← min{i ∈ B : μ_i < 0};
8:             B ← B\{h};
9:             continue;
10:        end if
11:        d ← x̄ − x;
12:        ᾱ ← min{α_i = (b_i − A_i x)/A_i d : A_i d > 0, i ∉ B};
13:        x ← x + ᾱ d;
14:        B ← A(x);
15:    end for
16: end procedure
```

By means of Algorithm 6.2.3 we are allowed to solve the same problem under box constraints, because we end up having three sets: $L$, indexes of variables fixed to the lower bound, $U$, indexes of variables fixed to the upper bound, and $F = \{1, \ldots, n\} \backslash (L \cup U)$, indexes of free variables.

$$\min \left\{ \frac{1}{2} \mathbf{x}^T Q \mathbf{x} + \mathbf{q}^\mathbf{T} \mathbf{x} \ : \ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \right\}$$

Let us suppose that $\mathbf{l} = \mathbf{0}$ without loss of generality, then the only constraints that we need to take into account are those in $F$ set, hence we can rewrite $\mathbf{x}$ as

$$\mathbf{x} = \begin{pmatrix} \mathbf{0}_{|L|} \\ \mathbf{x}_{|F|} \\ \mathbf{u}_{|U|} \end{pmatrix}$$

Thanks to this consideration the problem to solve becomes

$$\min_{\mathbf{x}_F} \left\{ \frac{1}{2} {\mathbf{x}_F}^T Q_{FF} \mathbf{x}_F + (\mathbf{q}_F + {\mathbf{u}_U}^T Q_{UF}) \mathbf{x}_F \right\} \ [+ \frac{1}{2} {\mathbf{u}_U}^T Q_{UU} \mathbf{u}_U + \mathbf{q}_U \mathbf{u}_U]$$

where the quantity in square brackets is constant in $\mathbf{x}_F$ hence irrelevant for minimization purposes.

### 6.2.4 Frank-Wolfe's method

Let us take a non-linear function and the following problem

$$\min\{f(\mathbf{x}) \ : \ A\mathbf{x} \leq \mathbf{b}\}$$

At any step, if the gradient of $f$ is not 0, then we approximate $f$ with a linear model and use it to pick a descent direction and this is formalized in Algorithm 6.2.4. The linear model is below the function in the quadratic case, hence we get a lower bound.

Supposing $f$ is convex we can make a first order model and minimize it over our constraints.

In this case we use the right constraints and an approximation of the function.

ALGORITHM 6.2.4 Pseudocode for **F**rank-**W**olfe's **M**ethod for non linear functions.

```
1: procedure FWM(f, A, b, x, ε)
2:     while (‖∇f(x)‖ > ε) do
3:         x̄ ← arg min{< ∇f(x), y > : Ay ≤ b};
4:         d ← x̄ − x;
5:         α ← Line_Search(f, x, d, 1);
6:         x ← x + αd;
7:     end while
8: end procedure
```

**Fact 6.2.3.** *Let $(P)$ be a non-linear minimum problem and let $\mathbf{d}$ be the descent direction chosen according to the first order model. Then either $< \nabla f(\mathbf{x}), \mathbf{d} >= 0$ and we are in the optimum or $< \nabla f(\mathbf{x}), \mathbf{d} >> 0$ and we are moving toward the optimum.*

Notice that the step-size $\alpha$ is bounded below by 0 and above by 1, because the direction $\mathbf{d}$ has as modulo the maximum movement that is possible without violating the consecutive constraint, as shown in Figure 6.4.

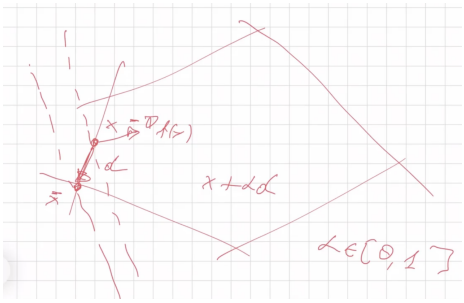



FIGURE 6.4: Bounds on the step-size.

**Fact 6.2.4.** *Let $(P)$ be a non-linear minimum problem and let $\mathbf{d}$ be the descent direction chosen according to the first order model. If $f$ is convex the value of*

*the first order model is below the true value of the function and a point that has* 0 *scalar product between the gradient and the descent direction is a global optimum. Formally,*

$$f(\mathbf{x})+ < \nabla f(), \mathbf{d} >\leq v(P)$$

*and if* $< \nabla f(), \mathbf{d} >= 0$*, then* $\mathbf{x}$ *is a global optimum.*

In general, we trust the linear model only around a certain point $\mathbf{x}$, but the algorithm might move far from it. For this reason, we introduce another constraint that bounds the movement around $\mathbf{x}$:

**Box Constraint:** $\|\mathbf{y}-\mathbf{x}\|_\infty \leq \tau$

**Separable penalty:** $\mu\|\mathbf{y}-\mathbf{x}\|_2^2$

As usual, in order to have a better direction, we need to choose a better model, meaning that we can use second order information as follows:

$$\bar{\mathbf{x}} = \mathbf{x} + \arg\min \left\{ \frac{1}{2}\mathbf{d}^T\nabla^2 f(\mathbf{x})\mathbf{d} + \nabla f(\mathbf{x})\mathbf{d} \ : \ A(\mathbf{x}+\mathbf{d}) \leq \mathbf{b} \right\}$$

Notice that if we do not want to compute the Hessian we can always approximate it using quasi-Newton formulae.

In the previous sections we addressed the problem of linear constrained optimization. Our first approach was to deal very little with constraints (projected gradient method), after a few improvements we took all of them and modify the function (Frank-Wolfe's method).

## 6.3 Dual methods

### 6.3.1 Dual methods for linear constrained optimization

Let us consider the quadratic problem with linear inequality constraints, formalized as

$$\min \left\{ \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T\mathbf{x} \ : \ A\mathbf{x} \leq \mathbf{b} \right\}$$

We can omit the constraints, by adding a penalization term and move to the dual problem:

$\forall$ fixed $\boldsymbol{\lambda} \geq \mathbf{0}$, the Lagrangian problem is

$$\psi(\boldsymbol{\lambda}) = \min_{\mathbf{x}} \left\{ \underbrace{\frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T\mathbf{x} + \boldsymbol{\lambda}^T(\mathbf{b}-A\mathbf{x})}_{f'(\mathbf{x})} \right\} \leq v(P)$$

**Fact 6.3.1.** *Let* $(P)$ *be the quadratic problem stated above and let* $\psi$ *be concave and* $Q \succ 0$*. The optimal solution is* $\mathbf{x}(\boldsymbol{\lambda}) = Q^{-1}(\boldsymbol{\lambda}^T A - \mathbf{q}^T)$.

*Proof.*

$$\begin{aligned}\frac{\partial f'(\mathbf{x})}{\partial \mathbf{x}} &= \frac{\partial(\frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T\mathbf{x} + \boldsymbol{\lambda}^T(\mathbf{b} - A\mathbf{x}))}{\partial \mathbf{x}} \\ &= \frac{1}{2}Q\mathbf{x} + \mathbf{q}^T - \boldsymbol{\lambda}^T A\end{aligned}$$

The gradient is 0 iff $\mathbf{x} = Q^{-1}(\boldsymbol{\lambda}^T A - \mathbf{q}^T)$. □

**Fact 6.3.2.** *Let $(P)$ be the quadratic problem stated above and let $\psi$ be differentiable. The gradient of $\psi$ is $\nabla\psi(\boldsymbol{\lambda}) = \mathbf{b} - A\mathbf{x}(\boldsymbol{\lambda})$.*

*Proof.*

$$\begin{aligned}\frac{\partial \psi(\mathbf{x})}{\partial \boldsymbol{\lambda}} &= \frac{\partial\Big(\min\Big\{\frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T\mathbf{x} + \boldsymbol{\lambda}^T(\mathbf{b} - A\mathbf{x})\Big\}\Big)}{\partial \boldsymbol{\lambda}} \\ &= \mathbf{b} - A\mathbf{x}\end{aligned}$$

□

**Fact 6.3.3.** *Let $(P)$ be the quadratic problem stated above, the Lagrangian dual is the following*

$$(D)\ \max\{\psi(\boldsymbol{\lambda})\ :\ \boldsymbol{\lambda}^T \geq 0\}$$

*and the optimum value of the dual problem is not only a lower bound, but it is exactly the optimum of the primal problem. Formally, $v(P) = v(D)$.*

At this point we only need to solve the dual problem, which has the shape of a box constrained optimization, where the box has only one boundary and this can be solved via quasi-Newton methods.
Notice that $\psi \notin \mathcal{C}^2$ so the Hessian is not defined.
This dual approach is advantageous in the case of a small number of constraints, because the size of the problem decreases. For example, in the case of one constraint, the Lagrangian dual becomes a problem in one variable, hence solvable through a line search.
In this method the degenerate case (more than one constraint active at a time) is not an issue.
If the quadratic function is convex we may use quasi-Newton methods (e.g. projected gradient). Otherwise the global optimality is not guaranteed and may be used if we accept not to be able to solve the original problem, but only to find a lower bound.

**Do you recall?**

In Frank-Wolfe's method we have naturally the upper-bound and the theory of duality allows us to compute a lower-bound for better convergence.

In non-linear constrained optimization the Lagrangian dual allows to have an upper-bound for the method, hence we have better convergence with respect to the convergence that we have with only the natural lower-bound.

### 6.3.2 Separable problems and partial dual

Let us assume that our constraints are separable, which means that it is not mandatory to work with all of them, but they can be split into constraints of groups of variables.

$$\min\{f(\mathbf{x}) \ : \ A\mathbf{x} \leq \mathbf{b}, \ E\mathbf{x} \leq \mathbf{d}\}$$

We can decide to use a **partial dual**, writing the Lagrangian problem picking only some constraints that we chose:

$$\psi(\boldsymbol{\lambda}) = \min_{\mathbf{x}}\{f(\mathbf{x}) + \boldsymbol{\lambda}^T(\mathbf{b} - A\mathbf{x}) \ : \ E\mathbf{x} \leq \mathbf{d}\}$$

The Lagrangian dual method may be better than projected gradient or worse and it depends on the instance.

In the dual approaches we can't move inside the feasible solution. We find an optimum for the dual, which surely breaks feasibility. Then, if the variable is above the upper-bound it gets decreased to the upper bound, otherwise if it is below the lower bound it takes the value of the lower bound.

This way we get an upper-bound for the function to be minimized.

## 6.4 Barrier Methods

**Do you recall?**

In dual programming we transform a (possibly) complicated constrained problem into a problem with simple constraints (or no constraints at all in the case of linear equality constrained primal problem).

The family of **barrier methods** is the best class of methods for solving **non-linear constrained** optimization problems. This kind of methods are designed to overcome the cons of dual approaches, namely the fact that $\psi$ does not have the Hessian (and this creates problems to quasi-Newton method) and the fact that $\mathbf{x}$ is not feasible until the end.

**Barrier methods** work applying a slight modification to the objective function $\psi$ in order to make it twice differentiable.

At the same time this methods keep the unconstrained property of the Lagrangian dual.

### 6.4.1 Barrier function and central path

Let us take the original problem

$$(P) \quad \min_{\mathbf{x}}\{f(\mathbf{x}) \ : \ A\mathbf{x} \leq \mathbf{b}\}$$

we modify it by adding a penalty to the objective function $f$, which depends on the scalar $\mu \in \mathbb{R}_+$ and it is called **logarithmic barrier**

$$(P_\mu) \ \min\{f_\mu(\mathbf{x}) = f(\mathbf{x}) - \mu \underbrace{\sum_{i=1}^{m} \log(b_i - A_i\mathbf{x})}_{\text{logarithmic barrier}}\}$$

The rationale behind this algorithm is to minimize a modification of the objective function $f$ which penalizes the value of the original function when the solution gets closer and closer to the boundaries of the feasible set, as displayed in two dimensions in Figure 6.5. The parameter $\mu$ is there to weight the proximity to the boundary.

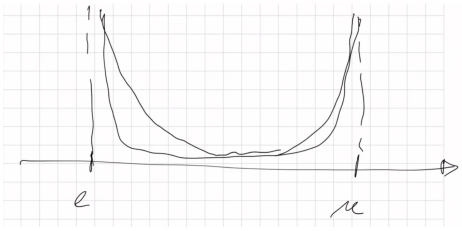


FIGURE 6.5: Let the function be in two dimensions and hence the barrier is $-\mu(\log(u - x) + \log(x - l)$. It is clear that the points close to the boundary are penalized.

Minimizing the logarithmic barrier would bring to the middle between the lower and the upper-bound and that point is called **analytic center** of the polyhedron

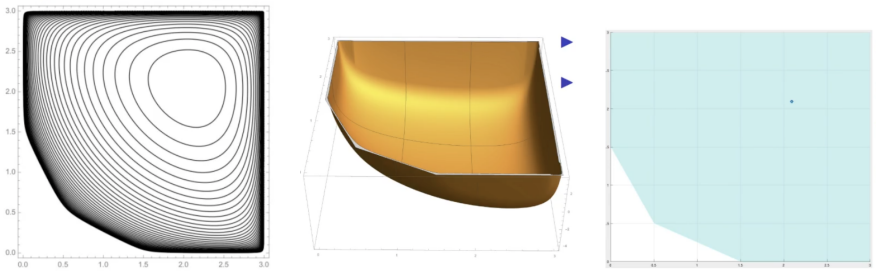
**Property 6.4.1.**

- *If $f$ is convex, $f_\mu$ is strictly convex;*
- *If $f \in \mathcal{C}^2$ then $f_\mu \in \mathcal{C}^2$, since $\log \in \mathcal{C}^\infty$;*
- *$\forall \mu \ \exists! \ \mathbf{x}_\mu$ optimal of $(P_\mu)$, since $\mu \sum_{i=1}^{m} \log(b_i - A_i\mathbf{x})$ is strictly convex;*
- *As $\mu \to 0$ $\mathbf{x}_\mu$ converges to the analytic center of the optimal face. An example of this behaviour may be seen in Example 6.4.1.*

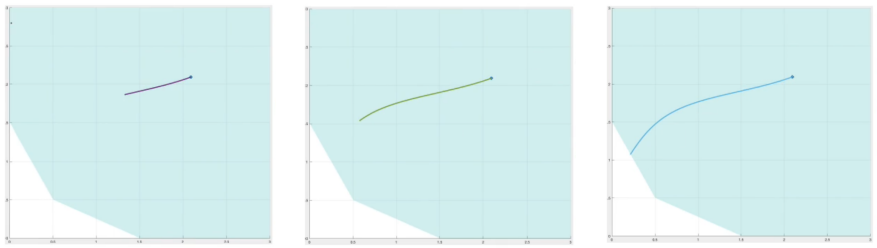
**Definition 6.4.1** (Central path)**.** *We term **central path** the sequence of points for the values of $\mu$ from $\infty$ to $0$. Formally,*

$$\mathcal{C} = \{\mathbf{x}_\mu \ : \ \mu \in (0, \infty)\}$$

**Example 6.4.1.** *In Figure 6.6(a) we can see the level-sets of our modified objective function $f_\mu$. We can see that the function goes to $-\infty$ the closest we get to the border. Intuitively, if $\mu \approx +\infty$, $f(\mathbf{x})$ is not relevant, hence we lay in the analytic center of the function $f_\mu$ (Figure 6.6(c)). As $\mu$ goes to $0$ the solution $\mathbf{x}$ moves along a smooth curve (from Figure 6.6(d) to Figure 6.6(f) until*



(a) (b) (c) $\mathbf{x}_\infty = \lim_{\mu \to \infty} \mathbf{x}_\mu$



(d) (e) (f) $\mathbf{x}_0 = \lim_{\mu \to 0} \mathbf{x}_\mu$

FIGURE 6.6: A convex $f_\mu$, the trajectory converges to the optimal solution of the problem, when $\mu \to 0$.

*it gets to an optimal solution of the original problem (center of optimal face of the polyhedron of constraints).*

Notice that the barrier function is chosen to satisfy some properties, such as being **self concordant**, that means that provided to be starting from a neighborhood $\mathcal{N}$ of $\mathcal{C}$, $\mathbf{x^i}$ gets "close" to $\mathbf{x}(\mu^i)$ in very few Newton's steps. Let us pick a point $\mathbf{x^i}$ and let us pick the closest vector $\mathbf{x}_{\mu^i}$. If we perform one step of Newton's method, we get $\mathbf{x^{i+1}}$, that is "much closer" to $\mathbf{x}_{\mu^i}$ than $\mathbf{x^i}$. Moreover, $\mathbf{x^{i+1}}$ is "close" to $\mathbf{x}_{\mu^{i+1}}$, with $\mu^{i+1} \ll \mu^i$ (more formally, $\mu^{i+1} = \tau\mu^i$, $\tau < 1$).

This behaviour may be observed in Figure 6.7

The convergence is linear in the number of variables ($O(\log n \log(1/\varepsilon))$) exponential if $\tau$ is very small, but these iterations are very costly, because the Hessian changes at each step, hence it needs to be recomputed.

### Computing Newton's step

.

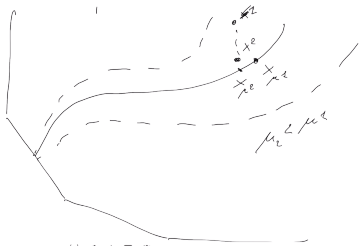


FIGURE 6.7: The dotted line represents a region where the Newton method is very efficient. We are starting from a point $x_1$, which belongs to that region and we want to move towards $x_{\mu^1}$. At next iterate $x_2$ is closer to $x_{\mu^2}$ than the current iterate.

**Do you recall?**

We are dealing with the following minimum problem

$$(P) \quad \min_{\mathbf{x}}\{f(\mathbf{x}) \ : \ A\mathbf{x} \leq \mathbf{b}\}$$

which dual is written as

$$\psi(\boldsymbol{\lambda}) = \min_{\mathbf{x}}\{f(\mathbf{x}) + \boldsymbol{\lambda}^T(\mathbf{b} - A\mathbf{x}) \ : \ E\mathbf{x} \leq \mathbf{d}\}$$

The Karush-Kuhn-Tucker conditions follow:

PRIMAL FEASIBILITY: $A\mathbf{x} + \mathbf{s} = \mathbf{b}, \ \mathbf{s} \geq \mathbf{0}$;

DUAL FEASIBILITY: $\mathbf{x}^T Q + \boldsymbol{\lambda}^T A = -\mathbf{q}^T, \ \boldsymbol{\lambda} \geq \mathbf{0}$;

COMPLEMENTARY SLACKNESS: $\lambda_i s_i = 0, \ \forall \ i = 1, \ldots, m$.

Notice that this is a **primal-dual** method, because we solve simultaneously the primal and the dual problem.

We can write a slackened version of KKT conditions in order to characterize $\mathbf{x}_\mu$, by imposing $\lambda_i s_i = \mu, \ \forall \ i = 1, \ldots, m$, where $\mu \in \mathbb{R}$ and should be decreased over iterations until it gets closer enough to 0.
Let us construct $\Lambda, \ S \in D(m, \mathbb{R})$ such that the diagonal is made of $\lambda_i$ and $s_i$ respectively:

$$\Lambda = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_m \end{pmatrix}, \ S = \begin{pmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_m \end{pmatrix}$$

At this point, we rewrite the problem in terms of the displacement from the fixed current point we are in:

- $\mathbf{x} \to \mathbf{x} + \boldsymbol{\Delta}\mathbf{x}$
- $\mathbf{s} \to \mathbf{s} + \boldsymbol{\Delta}\mathbf{s}$
- $\boldsymbol{\lambda} \to \boldsymbol{\lambda} + \boldsymbol{\Delta}\boldsymbol{\lambda}$

Taking into account the displacement required by the step we get that the KKT system becomes:

PRIMAL FEASIBILITY: $A\mathbf{x} + A\boldsymbol{\Delta}\mathbf{x} + \mathbf{s} + \boldsymbol{\Delta}\mathbf{s} = \mathbf{b},\ \mathbf{s} \geq \mathbf{0}$;

DUAL FEASIBILITY: $\mathbf{x}^T Q + \boldsymbol{\Delta}\mathbf{x}^T Q + \boldsymbol{\lambda}^T A + \boldsymbol{\Delta}\boldsymbol{\lambda}^T A = -\mathbf{q}^T,\ \boldsymbol{\lambda} \geq \mathbf{0}$;

COMPLEMENTARY SLACKNESS: $\lambda_i s_i + \lambda_i \Delta s_i + s_i \Delta\lambda_i + \Delta\lambda_i \Delta s_i = \mu,\ \forall\ i = 1, \ldots, m$.

In this new system of coordinates the first two KKT conditions are linear, while the third one is no longer linear ($\Delta\lambda_i \Delta s_i$). Let us rewrite the KKT system in a matrix form

$$\begin{bmatrix} Q & A^T & 0 \\ A & 0 & I \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta}\mathbf{x} \\ \boldsymbol{\Delta}\boldsymbol{\lambda} \\ \boldsymbol{\Delta}\mathbf{s} \end{bmatrix} \overset{(1)}{=} \begin{bmatrix} -(Q\mathbf{x} + \mathbf{q}) - \boldsymbol{\lambda}A \\ \mathbf{b} - A\mathbf{x} - \mathbf{s} \\ \mu\mathbf{u} - \Lambda S\mathbf{u} - \Delta\Lambda\Delta S\mathbf{u} \end{bmatrix} \approx \begin{bmatrix} 0 \\ 0 \\ \mu\mathbf{u} - \Lambda S\mathbf{u} \end{bmatrix} \tag{6.4.1}$$

Where (1) holds since $\boldsymbol{\Delta}\boldsymbol{\lambda}A = A^T\boldsymbol{\Delta}\boldsymbol{\lambda}^T$, although $\boldsymbol{\Delta}\boldsymbol{\lambda}$ is written without the "transpose" syntax to ease notation.

Notice that $\mathbf{u} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^m$ and has the purpose of adjusting dimension:

$$S\boldsymbol{\Delta}\boldsymbol{\lambda} = \begin{bmatrix} s_1\boldsymbol{\Delta}\lambda_1 \\ \vdots \\ s_m\boldsymbol{\Delta}\lambda_m \end{bmatrix} \in \mathbb{R}^m;\ \Lambda\boldsymbol{\Delta}\mathbf{s} = \begin{bmatrix} \lambda_1\boldsymbol{\Delta}s_1 \\ \vdots \\ \lambda_m\boldsymbol{\Delta}s_m \end{bmatrix} \in \mathbb{R}^m;\ \mu\mathbf{u} = \begin{bmatrix} \mu \\ \vdots \\ \mu \end{bmatrix} \in \mathbb{R}^m;$$

$$\Lambda S\mathbf{u} = \begin{bmatrix} -s_1\lambda_1 \\ \vdots \\ -s_m\lambda_m \end{bmatrix} \in \mathbb{R}^m;\ \Delta\Lambda\Delta S\mathbf{u} = \begin{bmatrix} -\Delta\lambda_1\Delta s_1 \\ \vdots \\ -\Delta\lambda_m\Delta s_m \in \mathbb{R}^m \end{bmatrix} \in \mathbb{R}^m;$$

The algorithm works by starting from a point that is strictly feasible ($\boldsymbol{\lambda}, \mathbf{s} > \mathbf{0}$), solve the Newton's system without the non-linear term for finding the direction and then find the best step-size ($\alpha < 1$) that allows to move to a feasible point.

## 6.5 Primal-dual interior point method

This method is based on the observation that we can solve the dual problem

$$(D) \quad \max\{-\boldsymbol{\lambda}^T\mathbf{b} - \frac{1}{2}\mathbf{x}^T Q\mathbf{x} \ : \ Q\mathbf{x} + \boldsymbol{\lambda}A = -\mathbf{q}, \ \boldsymbol{\lambda} \geq \mathbf{0}\}$$

thus obtaining both a lower and upper bound for the solution $\mathbf{x}$:

$$-\boldsymbol{\lambda}^T\mathbf{b} - \frac{1}{2}\mathbf{x}^T Q\mathbf{x} \leq v(D) \leq v(P) \leq \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T\mathbf{x}$$

We term **complementarity gap** the quantity $(\frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T\mathbf{x}) - (-\boldsymbol{\lambda}\mathbf{b} - \frac{1}{2}\mathbf{x}^T Q\mathbf{x}) = \boldsymbol{\lambda}^T\mathbf{s} = \mu$.
Once we found a solution for Equation (6.4.1), we perform a step and compute a new couple of primal and dual solutions and reduce the gap $\mu$.

**Fact 6.5.1.** *The normal equations (or KKT system) written in Equation* (6.4.1) *can be expressed as*

$$\begin{bmatrix} Q & A^T \\ A & -\Lambda^{-1}S \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta}\mathbf{x} \\ \boldsymbol{\Delta}\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{s} - \mu\Lambda^{-1}\mathbf{u} \end{bmatrix} \tag{6.5.1}$$

*Proof.* 1. express $\boldsymbol{\Delta}\mathbf{s}$ as a linear combination of $\boldsymbol{\Delta}\boldsymbol{\lambda}$. Let us work on the third line of Equation (6.4.1):

$$\begin{aligned} 0\boldsymbol{\Delta}\mathbf{x} + S\boldsymbol{\Delta}\boldsymbol{\lambda} + \Lambda\boldsymbol{\Delta}\mathbf{s} &= \mu\mathbf{u} - \Lambda S\mathbf{u} \\ \Lambda\boldsymbol{\Delta}\mathbf{s} &= \mu\mathbf{u} - \Lambda S\mathbf{u} - S\boldsymbol{\Delta}\boldsymbol{\lambda} \\ \boldsymbol{\Delta}\mathbf{s} &= \Lambda^{-1}\mu\mathbf{u} - \cancel{\Lambda^{-1}\Lambda}S\mathbf{u} - \Lambda^{-1}S\boldsymbol{\Delta}\boldsymbol{\lambda} \\ \boldsymbol{\Delta}\mathbf{s} &= \Lambda^{-1}\mu\mathbf{u} - S\mathbf{u} - \Lambda^{-1}S\boldsymbol{\Delta}\boldsymbol{\lambda} \\ \boldsymbol{\Delta}\mathbf{s} &= \Lambda^{-1} \cdot (\mu\mathbf{u} - S\boldsymbol{\Delta}\boldsymbol{\lambda}) - S\mathbf{u} \\ \boldsymbol{\Delta}\mathbf{s} &= \Lambda^{-1} \cdot (\mu\mathbf{u} - S\boldsymbol{\Delta}\boldsymbol{\lambda}) - \mathbf{s} \end{aligned} \tag{6.5.2}$$

2. let us work on the second row of Equation (6.4.1):

$$A\boldsymbol{\Delta}\mathbf{x} + 0\boldsymbol{\Delta}\boldsymbol{\lambda} + I\boldsymbol{\Delta}\mathbf{s} = 0 \Leftrightarrow A\boldsymbol{\Delta}\mathbf{x} = -\boldsymbol{\Delta}\mathbf{s} \tag{6.5.3}$$

3. substitute Equation (6.5.2) in Equation (6.5.3):

$$\begin{aligned} A\boldsymbol{\Delta}\mathbf{x} &= -\boldsymbol{\Delta}\mathbf{s} \\ A\boldsymbol{\Delta}\mathbf{x} &= -\Lambda^{-1} \cdot (\mu\mathbf{u} - S\boldsymbol{\Delta}\boldsymbol{\lambda}) + \mathbf{s} \\ A\boldsymbol{\Delta}\mathbf{x} &= -\mu\Lambda^{-1}\mathbf{u} + \Lambda^{-1}S\boldsymbol{\Delta}\boldsymbol{\lambda} + \mathbf{s} \\ A\boldsymbol{\Delta}\mathbf{x} - \Lambda^{-1}S\boldsymbol{\Delta}\boldsymbol{\lambda} &= -\mu\Lambda^{-1}\mathbf{u} + \mathbf{s} \end{aligned} \tag{6.5.4}$$

4. plug Equation (6.5.4) into the second row of the KKT conditions:

$$\begin{bmatrix} Q & A^T \\ A & -\Lambda^{-1}S \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta}\mathbf{x} \\ \boldsymbol{\Delta}\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{s} - \mu\Lambda^{-1}\mathbf{u} \end{bmatrix} \tag{6.5.5}$$

□

Notice that with respect to normal equations of **??**, we have in position $(2,2)$ a quantity $(-\Lambda^{-1}S)$, which is not 0, but it is the opposite of a strictly positive definite matrix.

**Fact 6.5.2.** *We can rewrite the system as something of the shape of reduced KKT (see Section 6.1).*

$$\begin{cases} Q\boldsymbol{\Delta}\mathbf{x} + A^T\boldsymbol{\Delta}\boldsymbol{\lambda} = 0 \\ (Q + A^T\Lambda S^{-1}A)\boldsymbol{\Delta}\mathbf{x} = A^T(\boldsymbol{\lambda} - \mu S^{-1}\mathbf{u}) \end{cases} \tag{6.5.6}$$

*Proof.* The first line follows from the expansion of the first row of the KKT system (Equation (6.5.5)) and the second one is obtained as follows:

1. isolate $\boldsymbol{\Delta}\boldsymbol{\lambda}$ from the second row of the KKT system (Equation (6.5.5)):

$$\begin{aligned} A\boldsymbol{\Delta}\mathbf{x} - \Lambda^{-1}S\boldsymbol{\Delta}\boldsymbol{\lambda} &= \mathbf{s} - \mu\Lambda^{-1}\mathbf{u} \\ \Lambda^{-1}S\boldsymbol{\Delta}\boldsymbol{\lambda} &= A\boldsymbol{\Delta}\mathbf{x} - \mathbf{s} + \mu\Lambda^{-1}\mathbf{u} \\ \boldsymbol{\Delta}\boldsymbol{\lambda} &= (\Lambda^{-1}S)^{-1}A\boldsymbol{\Delta}\mathbf{x} - (\Lambda^{-1}S)^{-1}\mathbf{s} + (\Lambda^{-1}S)^{-1}\mu\Lambda^{-1}\mathbf{u} \\ \boldsymbol{\Delta}\boldsymbol{\lambda} &= S^{-1}\Lambda A\boldsymbol{\Delta}\mathbf{x} - S^{-1}\Lambda\mathbf{s} + \mu S^{-1}\cancel{\Lambda}\cancel{\Lambda^{-1}}\mathbf{u} \\ \boldsymbol{\Delta}\boldsymbol{\lambda} &= S^{-1}\Lambda A\boldsymbol{\Delta}\mathbf{x} - S^{-1}\Lambda\mathbf{s} + \mu S^{-1}\mathbf{u} \\ \boldsymbol{\Delta}\boldsymbol{\lambda} &= \mu S^{-1}\mathbf{u} + \Lambda S^{-1}A\boldsymbol{\Delta}\mathbf{x} - \Lambda S^{-1}\mathbf{s} \\ \boldsymbol{\Delta}\boldsymbol{\lambda} &= \mu S^{-1}\mathbf{u} + \Lambda S^{-1}A\boldsymbol{\Delta}\mathbf{x} - \Lambda\mathbf{u} \\ \boldsymbol{\Delta}\lambda &= \mu S^{-1}\mathbf{u} + \Lambda S^{-1}A\boldsymbol{\Delta}\mathbf{x} - \boldsymbol{\lambda} \end{aligned} \tag{6.5.7}$$

2. substitute Equation (6.5.7) into the first row of the KKT system (Equation (6.5.5)):

$$\begin{aligned} Q\boldsymbol{\Delta}\mathbf{x} + A^T\boldsymbol{\Delta}\boldsymbol{\lambda} &= 0 \\ Q\boldsymbol{\Delta}\mathbf{x} + A^T\cdot(\mu S^{-1}\mathbf{u} + \Lambda S^{-1}A\boldsymbol{\Delta}\mathbf{x} - \boldsymbol{\lambda}) &= 0 \\ Q\boldsymbol{\Delta}\mathbf{x} + A^T\mu S^{-1}\mathbf{u} + A^T\Lambda S^{-1}A\boldsymbol{\Delta}\mathbf{x} - A^T\lambda &= 0 \\ Q\boldsymbol{\Delta}\mathbf{x} + A^T\Lambda S^{-1}A\boldsymbol{\Delta}\mathbf{x} &= -A^T\mu S^{-1}\mathbf{u} + A^T\boldsymbol{\lambda} \\ (Q + A^T\Lambda S^{-1}A)\boldsymbol{\Delta}\mathbf{x} &= A^T(\boldsymbol{\lambda} - \mu S^{-1}\mathbf{u}) \end{aligned} \tag{6.5.8}$$

□

Let us term $M = Q + A^T\Lambda S^{-1}A$ and the following holds.

**Fact 6.5.3.** *If $A$ has full column rank (aka it is invertible) then $M$ is positive definite ($M \succ 0$).*

At this point we need to factorize the matrix $M$, that changes at each iteration (since $\Lambda S^{-1}$ does) and this is the bottleneck.
Cholesky factorization may be used, although its complexity is cube. Another downside of this approach is that the matrix $M$ is much denser than $A$, $\Lambda$ and $S^{-1}$.
An orthogonal approach to the reduced KKT is called **predictor-corrector** and it works computing a solution without taking into account the non linear term $\boldsymbol{\Delta}\Lambda\boldsymbol{\Delta}S\mathbf{u}$, then computing it according to the approximated solution and repeat until convergence.
The bottleneck again is solving the system in Equation (6.4.1).
For what concerns implementation, we should start from a triplet $(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s})$, that could be not feasible and then compute the residuals and iterate until feasibility is reached.

$$\mathbf{r^D} = -(Q\mathbf{x} + \mathbf{q}) - \boldsymbol{\lambda}A$$
$$\mathbf{r^P} = \mathbf{b} - A\mathbf{x} - \mathbf{s}$$

When dealing with the step size we need to highlight the fact that $\lambda + \boldsymbol{\Delta\lambda} \geq \mathbf{0}$ and $\mathbf{s} + \boldsymbol{\Delta s} \geq \mathbf{0}$ should hold.

In order to achieve this we find the maximum $\alpha$ that satisfies the equality and then multiply it by a constant $\bar{\alpha} = 0.995$ (or 0.9995), in order to get closer.

Let us assume that we also have a bunch of box constraints, hence our problem becomes

$$(P) \qquad \min\left\{\frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T\mathbf{x} \;:\; A\mathbf{x} = \mathbf{b},\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}\right\}$$

In this special case, things simplify a lot.