Introduction to Optimization

Romain Couillet and Ronald Phlypo

3rd November 2020

Contents

Chapter 1

Course

1.1 Motivation

The central objective of the class is to solve the following problem:

$$\text{Find } x^\star \in \operatorname{argmin}_{x\in\Omega\subset\mathscr{X}} f(x) \tag{1.1}$$

for some function $f : \mathscr{X} \to \mathbb{R} \cup \{+\infty\}$.

Remark 1. *Note that, in all generality,* $\operatorname{argmin}_{x\in\Omega\subset\mathscr{X}} f(x)$ *is a subset of* $\mathscr{X}$ *which may be empty, a singleton, a finite or countable discrete set or an uncountable set.*

Here f is usually referred to as the *cost*, *penalty*, *potential*, or *objective* function. The set Ω is generally referred to as the set of constraints.

1.1.1 What makes an optimization problem easy ?

Some optimization problems are easier to solve than others, it is thus of particular interest to know on what basis one can classify different optimization problems. This is also the strategy that will be used all along this course and should prepare the reader to answer a central question

"To what class of optimization problems does my particular problem belong?"

Once an answer to this question is found, all properties of this class become readily available and efficient and fast algorithms can be used or designed.

To illustrate, compare the following problems:

The knapsack problem: Among k objects, each of weight w_k, which is the maximal subset (in terms of weight) of objects S that can be taken into a knapsack that is known to support a maximal weight W, i.e., $\sum_{i\in S} w_i \leq W$ and $W - \sum_{i\in S} w_i$ is minimal? This is a problem found, e.g., when minimizing file fragmentation over (multiple) disks or in transportation of goods.

The travelling salesman problem: Let n cities be at pairwise distances $(d_{ij})_{i,j=1}^n$. What is the permutation of cities $(i_1, i_2, \ldots, i_n) = \pi(1, 2, \ldots, n)$ for which $\sum_{k=0}^{n-1} d_{i_k, i_{k+1}} + d_{i_n, i_0}$ is minimal?

Blind deconvolution: What is the function f such that for (h, g) known, f minimizes $\|g - h * f\|$? Typically observed in optics (astrophysics, microscopy, ...) or telecommunications when the channel h is known, but the signal f is not.

Inverse imaging: Suppose we have $y \approx \sum_{j=1}^p a_{i_j} x_{i_j} \in \mathbb{R}^m$ where $A = [a_1, \ldots, a_n] \in \mathbb{R}^{m\times n}$ is a fat matrix $(m \ll n)$. The factors that are deterministic for our observation may be found by minimizing p such that $\|y - \sum_{j=1}^p a_{i_j} x_{i_j}\| \leq \varepsilon$ for some ε. This problem is typical of inverse problem solving such as genomics, bio-electromagnetic imaging, etc.

Clearly, both the knapsack and travelling salesman problems are combinatorial in nature. One needs to go through all possible combinations or permutations to provide the ultimate answer, something that scales badly when the number of instances – objects or cities – increases. The problem of blind deconvolution is *easy* if the norm is smooth, but may be less so when the norm contains discontinuities. The inverse imaging problem is again a combinatorial problem (this time over the index set), subject to a constraint that is not necessarily differentiable.

In this course we mainly deal with functions over (a union of) continuous domains having a (piece-wise) continuous image through the objective function.

1.1.2 Description of the course

The objective of the course is to solve Problem (1.1) starting with elementary optimization tools and gradually shifting to more advanced tools. This will be done under various characterizations of f, the constraint set Ω, etc.

The first basic condition that shall be set throughout the course is that f is a *convex function* (that is – essentially – a U- or V-shaped function). The main advantage is that many algorithms exist with which the solution of (1.1) can be found. The most popular of these algorithms is the gradient descent method which consists in "descending" orthogonally to the level sets of the function (i.e., the sets having equal values), that is moving in the direction opposite to that of the gradient. One must understand here that the convexity assumption is key to ensure that the solution of (1.1) corresponds to a set of *global minima* for f; if not convex, f may contain local minima in which traditional descent methods will generally get stuck. This is a long standing problem for which, for high-dimensional $\mathscr{X}$, no satisfactory workaround exists. This shall not be discussed further in this class.

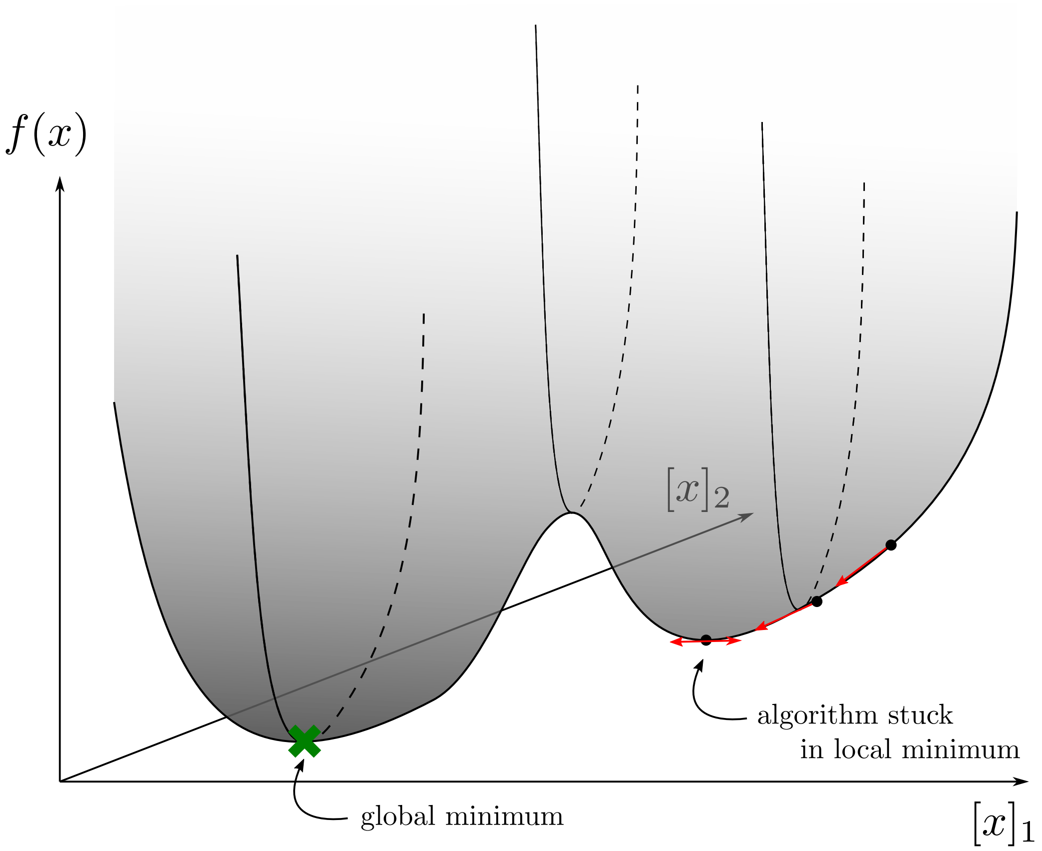


Figure 1.1: Example of non-convex function f and associated convex optimization scheme trapped in a local minimum.

This however implies that such a gradient be defined. As such, the first part of the course will assume that the function f, in addition to being convex, is everywhere differentiable. This will in fact not always be enough to ensure convergence of the developed algorithms as the function f may have sharp slopes away from the minima. In such a scenario, following a gradient descent iteration may project successive iterations further and further away from the minimum, if the *step sizes* are not appropriately set (resulting for instance in a back-and-forth loop trapping the algorithm); to this end, a control on a norm of the Hessian will be required. On the other hand, if the function f has a too flat region away from the minima, the algorithm may stop too early (by considering that the progression is slow and that the minima should be close); there, to avoid stopping too far from the minima, a control on the smallest eigenvalues of the Hessian is necessary.

Despite those essentially marginal problems, we shall see that the gradient descent algorithm is extremely simple, powerful, and fast. We will show that it decays in general exponentially fast to the minimum that is thus found in logarithmic time. As it sets a reference in the set of algorithms for convex optimization, somewhat paradoxically, we say that gradient descent has a *linear* convergence speed since the log of the objective function converges to the optimal value as $-ct$ (for a constant c and time step t). But there exist much faster algorithms. We shall notably introduce and discuss the Newton method. The main trick behind the Newton method is to refuse to descent along the gradient but rather to anticipate that, if f is smooth enough, it should locally look like a polynomial of order 2, and one should then descend in the direction of the minimum of this approximation of f rather than along the local gradient. By this trick, Newton's method can be shown to have a quadratically fast convergence speed, if initialized close enough to the minima (i.e., the log of the objective function converges as a quadratic function, $-ct^2$, of the time step t).

However, taking gradient descent steps involves being freely allowed to move in the space Ω. If the latter is bounded, it may occur that the descent algorithm desires to step *out of* Ω. If one just breaks progression by blocking the descent on the edges of Ω, the algorithm stops, often too early, not having reached the minimum. A first natural approach to avoid such a problem is referred to as interior point (or barrier) methods, which change the function f into the sum of f and a smooth function that induces a penalty close to the boundaries of Ω. Typically, instead of minimizing f, one then minimizes instead $f - \log(h)$ for some function $h \geq 0$ that equals zero on the edges of Ω (thereby inducing a close-to-infinite cost near the boundaries).

But interior point methods are difficult to work with. For once, since $f - \log(h)$ instead of just f is minimized, one needs subsequent iteration steps where $f - \log(h)$ is changed into $f - \mu_k \log(h)$ for some sequence of μ_k converging to 0. The speed of convergence of μ_k to zero naturally raises the same problems as described previously: too small μ_k induce sharp transitions for $f - \mu_k \log(h)$ near the edges of Ω, hence leading to a possible back-and-forth oscillation problem, while too large values of μ_k dramatically change the cost function f.

Even more importantly, interior point methods are of no use when Ω only spans a subspace of $\mathscr{X}$, so in particular when *equality constraints* are set (such as imposing that the sought-for x must satisfy $Ax = b$ for some matrix and vector A, b). This shall bring us to a drastically different way to look at convex optimization through the so-called *dual methods*, and notably the dual Lagrangian optimization. Dual methods consist in changing the optimization problem to another one, the cost of which ideally equals that of the primal problem at the solution. The advantage of the dual problem is that it naturally deals with both equality and inequality constraints.

Yet, all aforementioned approaches were assuming differentiable f. When f is simply convex but not necessarily differentiable, another set of tools is needed as the gradient itself (and thus all associated algorithms) is not defined. To circumvent this problem, we shall first define the notion of *subgradient* which extends that of gradient to non-differentiable functions. The resulting *subgradient algorithm* that naturally arises from the definition is, however, *no longer* a descent method, meaning that the subgradient steps may lead the algorithm to increase the cost of f and thus result in diverging algorithms. Nevertheless, a fallback solution exists, but it is rarely used in practice as it is known to converge slowly. In addition, the subgradient at point $x \in \mathscr{X}$ is defined as a *set* of possible directions, rather than as a unique vector (as in the case of the gradient); this means that there does not exist a deterministic manner to choose the next step direction which must thus be chosen at random.

Instead, we shall introduce more contemporary solutions to work around the differentiability issue, featuring the so-called *proximal* approach. The proximity operator, by means of a quadratic "relaxation" trick, first avoids the problem of multiple directions of exploration from a point x in the resulting algorithm. In addition, it is a powerful tool that does not require to know the global behavior of f (as was the case with the Hessian control). This comes in general at the price of a slower – though guaranteed – convergence. The course will notably conclude with examples of very modern algorithms in convex optimization, among which the forward-backward splitting method and the Douglas-Rachford splitting algorithms to solve generic convex optimization problems under convex set constraints.

We summarize in the table below the successive steps of the class:

Sec.	Covered Topic	Constraints on Optimization Problem (1.1)
2.1	Notions of convexity: what is a convex set?	-
2.2	From convex sets to convex functions	-
3.1	The gradient descent method	Differentiable f
3.2	Newton's method	Differentiable f, initialization close to minimum
3.3	Interior point methods	Ω is bounded, gradient descent can get trapped in an edge
4.1	Equality constrained optimization	Ω is a manifold of $\mathscr{X}$ of lower dimension, Gradient descent does not work, one moves to the dual space
4.2	Generic dual methods	Ω is the intersection of multiple manifolds of $\mathscr{X}$
5.1	Subgradients	f is not differentiable, the gradient is no longer defined
5.2	Proximal methods	non-differentiable f, convergence methods using subgradients
5.3	Sum of two functions	non-differentiable f and convex possibly bounded Ω.

1.1.3 Examples

Example 1 (Portfolio Optimization). *In statistical finance, one aims at distributing a unit (1) amount of money across n market assets, the objective being to have an ensured return gain. In order not to suffer from the so-called "volatility" of the market, the trader will seek to minimize the* variance *of the portfolio return. To this end, it is assumed that the returns at time t can be modelled with a random vector $x_t \in \mathbb{R}^n$ (with $[x_t]_i > 0$ if the return is favorable or < 0 otherwise) with mean μ and variance $C \in \mathbb{R}^{n\times n}$ that, in the short-term, do not depend on the time t. Minimizing the portfolio variance can be shown to correspond to solving the problem*

$$\operatorname{argmin}_{w\in\mathbb{R}^n} \mathrm{E}[|w^\top(x_t-\mu)|^2] = w^\top C w \text{ such that } \sum_{i=1}^n [w]_i = 1.$$

Here, $w \in \mathbb{R}^n$ is the so-called portfolio *and the constraint $\sum_{i=1}^n [w]_i = 1$ states that the total sum of money equals 1, as desired.*

This problem is a *quadratic optimisation problem under linear constraints.* Using the method of Lagrange multipliers, it can be shown to have an explicit solution

$$w^\star = \frac{C^{-1}1_n}{1_n^\top C^{-1} 1_n}$$

where $1_n \in \mathbb{R}^n$ is the vector full of ones.

However, the solution may have negative entries, which can be tolerated if one assumes the capability to *sell* rather than only *buy* assets. But in practice, it is rather demanded that $[w]_i \geq 0$ and the problem is now turned into

$$\operatorname{argmin}_{w\in\mathbb{R}^n} \mathrm{E}[|w^\top(x_t-\mu)|^2] = w^\top C w \text{ such that } \sum_{i=1}^n [w]_i = 1 \text{ and } [w]_i \geq 0.$$

In this case, the problem no longer has an explicit solution and can be seen as an optimization problem with *inequality constraints.* Methods based either on interior point techniques or proximal point algorithms are needed here.

Example 2 (Support Vector Machines). *Support vector machines is a very popular machine learning algorithm for data classification. The underlying idea is to design an automated method (the* machine*) that learns an implicit mapping between a set of vectors $x_1,\ldots,x_m \in \mathbb{R}^n$ and their respective association classes $y_1,\ldots,y_m \in \{-1,1\}$. The pairs (x_i,y_i), $i=1,\ldots,m$, are known and the objective is to discover the class of a new datum $x_0 \in \mathbb{R}^p$. To this end, we suppose that the data from class -1 can be isolated from those in class 1 by a hyperplane of $\mathbb{R}^p$ of the form*

$$x \mapsto x^\top w^\star + b^\star$$

for some $w^\star, b^\star \in \mathbb{R}^p$ that need be determined. Since such a hyperplane is in general not unique, we seek the one having maximal distance to the vectors $x_1, \ldots, x_m$, which can be shown to be equivalent to solving

$$(w^\star, b^\star) \in \operatorname{argmin}_{w\in\mathbb{R}^n, b\in\mathbb{R}} \|w\|^2 \text{ such that } y_i(w^\top x_i - b) \geq 1.$$

This corresponds again to a convex optimization problem with m inequality constraints and differentiable cost function and thus can be possibly solved by interior point methods or proximal methods.

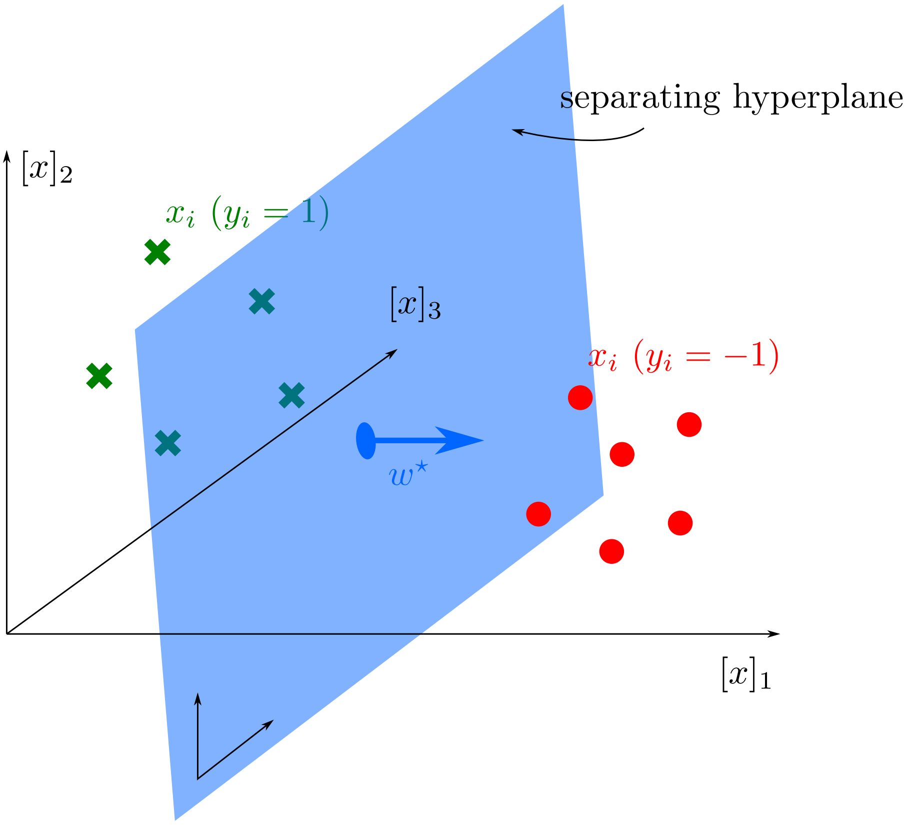


Figure 1.2: The separating hyperplane of a support vector machine.

However, the separating hyperplane may also not exist and the problem then has no solution. An alternative to achieve a reasonable solution is to relax the problem as

$$(w^\star, b^\star) \in \operatorname{argmin}_{w,b\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^{m} \max(0, 1 - y_i[w^\top x_i - b]) + \lambda\|w\|^2$$

for some $\lambda > 0$ that trades off the maximal hyperplane distance constraint to the cost of not meeting the $y_i[w^\top x_i - b] \geq 1$ inequality. This is referred to as a support vector machine with soft margins.

Such a problem consists in the minimization of the sum of two convex functions, one of which is differentiable ($\|w\|^2$) while the other one is convex but not everywhere differentiable. As we shall see in this course, this problem falls in line with the more elaborate proximal point approaches, and notably the *forward-backward splitting algorithm.*

Example 3 (Compressive Sensing)**.** *Compressive sensing is a wide field of research consisting in retrieving a large dimensional signal $x \in \mathbb{R}^n$ from a low dimensional linear observation of it $y = Ax \in \mathbb{R}^p$. In all genericity, this is an ill-posed problem as A cannot be inverted and many x's solving $y = Ax$ exist in general (if x_0 is a solution, then all $x_0 + z$ with $Az = 0$ are also solutions).*

The idea behind compressive sensing is to assume that x is a sparse *vector in the sense that the number of non-zero elements in x is small, according to Occam's razor (the simplest solution is the most plausible one). Finding the vector x with smallest non-zero entries satisfying $y = Ax$ consists in solving the problem*

$$x^\star \in \operatorname{argmin}_{x\in\mathbb{R}^n} \|x\|_0 \text{ such that } y = Ax$$

where $\|\cdot\|_0$ is the so-called l_0 pseudo-norm that counts the number of non-zero elements in x.

Unfortunately, this problem is *not convex* and can only be solved by an exhaustive search over the support (the entries of non-zero value) in x. This is prohibitively expensive. A whole theory has then been developed to show that, under some conditions on the matrix A, solving the l_0-norm problem is equivalent to solving an l_1-norm problem

$$x^\star \in \operatorname{argmin}_{x\in\mathbb{R}^n} \|x\|_1 \text{ such that } y = Ax$$

where now $\|x\|_1 = \sum_{i=1}^n |[x]_i|$ is a convex, yet, not-everywhere differentiable (because of the absolute value) function with linear equality constraints. Rewritting this problem as

$$x^\star \in \operatorname{argmin}_{x\in\mathbb{R}^n} \|x\|_1 + \imath_{\{x:\, y=Ax\}}(x)$$

with $\imath_\Omega(x) = 0$ if $x \in \Omega$ and $\imath_\Omega(x) = +\infty$ if $x \notin \Omega$, this can be seen as the sum of two non-differentiable, convex functions, which is then prone to be solved through the so-called *Douglas-Rachford splitting* method, discussed at the very end of this course.

1.2 Basics of Convex Optimization

Most of the content of the course is valid for ambient spaces $\mathscr{X}$ being a Hilbert space, i.e., a real or complex vector space on which a scalar product $\langle x, y\rangle$ is defined. This notably includes infinite dimension spaces, such as function spaces. Yet, for simplicity, we restrict ourselves all along to $\mathscr{X}$ being the finite dimensional Euclidean vector space $\mathbb{R}^n$, with scalar product $\langle x, y\rangle = x^\top y = \sum_{i=1}^n [x]_i [y]_i$.

1.2.1 Reminders of set theory in Euclidean spaces

Given a set $\mathscr{S} \subset \mathscr{X}$, where $\mathscr{X}$ is a Euclidean space, we have the following definitions:

Definition 1 (Subset). *$\mathscr{S}$ is called a subset of $\mathscr{X}$ if and only if we have*

$$\forall x, (x \in \mathscr{S}) \implies (x \in \mathscr{X})$$

Definition 2 (Complement of a set). *The set*

$$\complement_{\mathscr{X}}(\mathscr{S}) = \{x \in \mathscr{X}, x \notin \mathscr{S}\}$$

is the complement of the set $\mathscr{S}$ in $\mathscr{X}$.

Definition 3 (Open set). *If $\forall x \in \mathscr{S}$, $\exists \varepsilon > 0$ such that $\forall y$, $\|y - x\| < \varepsilon$ implies $y \in \mathscr{S}$, then the set $\mathscr{S}$ is called an open set.*

Definition 4 (Limit point). *Any $y \in \mathscr{X}$ for which*

$$\forall \varepsilon > 0, \exists x \in \mathscr{S} : \|x - y\| < \varepsilon$$

is called a limit point of the set $\mathscr{S}$.

Property 1 (Limit points of an open set $\mathscr{S}$). *$\mathscr{S}$ itself forms a subset of all the limit points of $\mathscr{S}$.*

Definition 5 (Closed set). *A set that contains all of its limit points is called a closed set.*

Property 2 (Finite unions and intersections).

- *a finite union of open (closed) sets is an open (closed) set*
- *a finite intersection of open (closed) sets is an open (closed) set*

Example 4 (Classical open and closed sets). *In $\mathscr{X} = \mathbb{R}$,*

- *segments $[a, b]$ are closed*
- *segments (a, b) and half-lines (a, ∞) are open*
- *half-open segments $[a, b)$ are neither open nor closed; however $[a, b)$ is an* open *set of $\mathscr{X} = [a, \infty)$ (prove it!)*

Definition 6 (Closure of a set). *The closure* $\operatorname{clos}\mathscr{S}$ *of a set $\mathscr{S}$ is the set of all limit points of $\mathscr{S}$. It is the smallest closed set that contains $\mathscr{S}$.*

Property 3 (Closure of a closed set). *$\mathscr{S}$ is a closed set $\iff \mathscr{S} = \operatorname{clos}\mathscr{S}$.*

Definition 7 (Interior of a set). *The set*

$$\operatorname{int}\mathscr{S} = \left\{x \middle| \exists \varepsilon > 0 : \|x - y\| < \varepsilon \implies y \in \mathscr{S}\right\}$$

is called the interior of the set $\mathscr{S}$. It is the largest open set contained in $\mathscr{S}$.

Property 4 (Interior of an open set). *$\mathscr{S}$ is an open set $\iff \mathscr{S} = \operatorname{int}\mathscr{S}$.*

Definition 8 (Exterior of a set). *The set*

$$\operatorname{ext}(\mathscr{S}) = \operatorname{int}\complement_{\mathscr{X}}(\mathscr{S})$$

is called the exterior of $\mathscr{S}$. It is the largest open set that does not intersect with $\mathscr{S}$.

Definition 9 (Boundary of a set). *The set*

$$\partial\mathscr{S} = \complement_{\operatorname{clos}\mathscr{S}}\operatorname{int}\mathscr{S} = \operatorname{clos}\mathscr{S} \cap \operatorname{clos}\complement_{\mathscr{X}}\mathscr{S}$$

is called the boundary of the set $\mathscr{S}$.

Property 5 (Duality of closure and interior). *We have*

$$\operatorname{int}\mathscr{S} = \complement_{\mathscr{X}}\operatorname{clos}\left(\complement_{\mathscr{X}}\mathscr{S}\right)$$

and

$$\operatorname{clos}\mathscr{S} = \complement_{\mathscr{X}}\operatorname{int}\left(\complement_{\mathscr{X}}\mathscr{S}\right)$$

from which $\operatorname{int}\mathscr{S} \cup \operatorname{clos}\left(\complement_{\mathscr{X}}\mathscr{S}\right) = \operatorname{clos}\mathscr{S} \cup \operatorname{int}\left(\complement_{\mathscr{X}}\mathscr{S}\right) = \mathscr{X}$ *and* $\operatorname{int}\mathscr{S} \cap \operatorname{clos}\left(\complement_{\mathscr{X}}\mathscr{S}\right) = \operatorname{clos}\mathscr{S} \cap \operatorname{int}\left(\complement_{\mathscr{X}}\mathscr{S}\right) = \emptyset$.

Remark 2 (Partitioning of $\mathscr{X}$). *Given a set $\mathscr{S}$, the space $\mathscr{X}$ can be partitioned in three parts* $\operatorname{int}\mathscr{S}$, $\partial\mathscr{S}$, *and* $\operatorname{ext}\mathscr{S}$ *such that* $\operatorname{int}\mathscr{S} \cup \partial\mathscr{S} \cup \operatorname{ext}\mathscr{S} = \mathscr{X}$ *and* $\operatorname{int}\mathscr{S} \cap \partial\mathscr{S} = \partial\mathscr{S} \cap \operatorname{ext}\mathscr{S} = \operatorname{int}\mathscr{S} \cap \operatorname{ext}\mathscr{S} = \emptyset$.

1.2.2 Convex Sets

There are two fundamental reasons to naturally initiate a convex optimization study by defining convex sets. As shall appear subsequently, convex functions may be naturally defined by means of convex sets (in that their *epigraph* is a convex set). Besides, constrained optimization where the argmin in Equation 1.1 is searched in a set $\Omega \subset \mathscr{X}$ is conveniently solved only if Ω *is* convex. As such, convex optimization essentially revolves around finding a particular point in a convex set (or in the finite intersection of convex sets which, as we shall see, is a convex set).

Basic notions

Definition 10 (Convex set)**.** *A set $\mathscr{C} \subset \mathscr{X}$ is convex if and only if (iff) for all $x, y \in \mathscr{C}$ and $\lambda \in [0,1]$,*

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

In other words, if two points x, y belong to a convex $\mathscr{C}$, then so do all points on the connecting segment $[x, y]$. And if for all x, $y \in \mathscr{C}$ all points on the segments $[x, y]$ also belong to $\mathscr{C}$, the latter is convex.

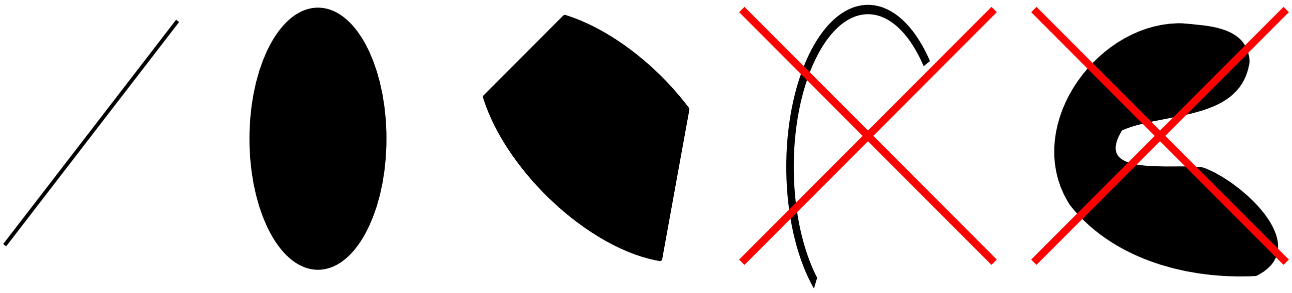


Figure 1.3: Convex sets and non-convex sets (crossed out).

Remark 3 (Ensemble manipulations on convex sets)**.** *For convex sets $\mathscr{C}_1$, $\mathscr{C}_2$,*

- *$\mathscr{C}_i$ can be open, closed, bounded, unbounded;*
- *$\mathscr{C}_1 \cap \mathscr{C}_2$ is convex;*
- *$\mathscr{C}_1 \cup \mathscr{C}_2$ is not necessarily convex.*

Remark 4 (List of convex sets)**.** *The following ensembles are convex:*

- *line, segment, half-line, $\mathbb{R}^n$;*
- *a vector subspace;*
- *hyperplanes $\{x,\ x^\mathsf{T} a = b\}$, half-spaces $\{x,\ x^\mathsf{T} a \le b\}$;*
- *balls $\mathscr{B}(x_c; r) \equiv \{x,\ \|x - x_c\| \le r\}$ (see proof below) and ellipsoids $\{x,\ (x-x_c)^\mathsf{T} P^{-1}(x-x_c) \le r\}$.*

Exercise 1 (Ball convexity)**.** *Show that the ball $\mathscr{B}(x_c; r) \equiv \{x,\ \|x - x_c\| \le r\}$ centered at x_c and of radius $r > 0$ is convex.*

Proof of ball convexity. Let $x, y \in \mathscr{B}(x_c; r)$. Then, $\forall \lambda \in [0, 1]$,

$$\begin{aligned} \|\lambda x + (1-\lambda)y - x_c\| &= \|\lambda(x-x_c) + (1-\lambda)(y-x_c)\| \\ &\le \lambda\|x-x_c\| + (1-\lambda)\|y-x_c\| \le r \end{aligned}$$

where we used the triangular inequality and linearity of norms. □

When linear equality and inequality constraints appear in an optimization problem (for instance positivity constraints on some variables), polyhedrons will naturally arise.

Exercise 2 (Polyhedron)**.** *For matrices $A \in \mathbb{R}^{l\times n}$, $B \in \mathbb{R}^{m\times n}$ and vectors $b \in \mathbb{R}^l$, $d \in \mathbb{R}^m$, show that the polyhedron*

$$\mathscr{P} = \{x,\ Ax \le b,\ Cx = d\}$$

is convex, where $Ax \le b$ and $Cx = d$ are understood entry-wise.

An ensemble of particular interest in optimization are cones and intersections of cones.

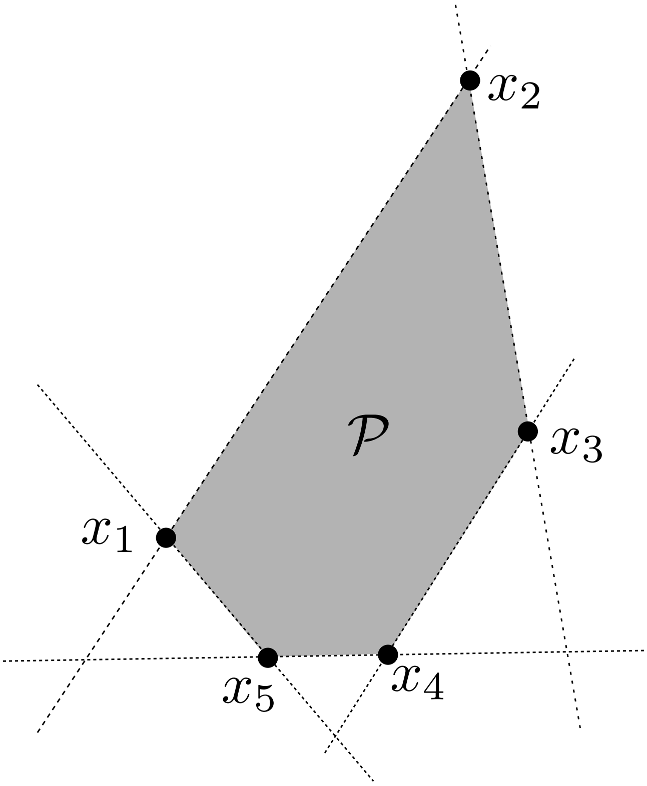


Figure 1.4: A polyhedron.

Definition 11 (Cone)**.** *A set $C \subset \mathcal{X}$ is a cone if, for all $x \in C$ and all $\theta > 0$, $\theta x \in C$.*

Definition 12 (Convex cone)**.** *A cone is a convex cone if for all $x, y \in C$ and all $\theta_1, \theta_2 > 0$, $\theta_1 x + \theta_2 y \in C$. In particular, a convex cone is convex.*

Proof of the convexity of convex cones. $\forall x, y \in C$, taking those θ_1, θ_2 for which $\theta_1 + \theta_2 = 1$ yields the desired result. □

Exercise 3 (Set of symmetric (semi) positive definite matrices)**.** *Show that the set of symmetric (semi) positive definite matrices is a convex cone.*

Proof. Let $\theta_A, \theta_B > 0$ and A, B symmetric (semi) positive definite. Then, $\forall x \in \mathbb{R}^p$

$$x^\top(\theta_A A + \theta_B B)x = \theta_A x^\top A x + \theta_B x^\top B x \geq 0$$

where equality may be obtained only if A, B are semi positive definite. □

An important notion in (convex) optimization is the notion of *convex hulls.*

Definition 13 (Convex combinations)**.** *Let $x_1, \ldots, x_k \in \mathcal{X}$. Then the set of convex combinations of $\{x_1, \ldots, x_k\}$ is the set*

$$\left\{\theta_1 x_1 + \ldots + \theta_k x_k \mid \sum_{i=1}^{k} \theta_i = 1,\ \theta_1, \ldots, \theta_k \geq 0\right\}.$$

In particular, the polyhedron in Figure 1.4 is the set of convex combinations of $x_1, \ldots, x_5$.

Exercise 4. *Prove that the set of convex combinations of $\{x_1, \ldots, x_k\}$ is a convex set.*
Hint. *Proceed by induction over $k \geq 2$.*

Definition 14 (Convex hull)**.** *The convex hull of a set $\mathcal{S} \subset \mathcal{X}$, denoted* $\operatorname{conv}(\mathcal{S})$*, is the set of all convex combinations of points in $\mathcal{S}$, i.e.,*

$$\operatorname{conv}(\mathcal{S}) = \left\{\theta_1 x_1 + \ldots + \theta_k x_k \mid \sum_{i=1}^{k} \theta_i = 1,\ \theta_1, \ldots, \theta_k \geq 0,\ x_1, \ldots, x_k \in \mathcal{S},\ k \geq 0\right\}.$$

Since convex combinations form convex sets, convex hulls are obviously convex and are in a way “minimally” convex in the following sense:

Property 6 (Convex sets and convex hulls)**.** *The set* $\operatorname{conv}(\mathcal{S})$ *is the smallest convex set containing $\mathcal{S}$. In particular, $\mathcal{S}$ is convex if and only if $\mathcal{S} = \operatorname{conv}(\mathcal{S})$.*

A duality perspective

An alternative approach to define convex sets lies in the remark that *no tangential hyperplane of the set can intersect a convex set.* This induces a so-called "dual" approach where, instead of considering relations for the points x of the convex set $\mathscr{C}$, we rather define $\mathscr{C}$ through the *directions* of their tangential hyperplanes, which surround its boundary $\partial\mathscr{C}$.

This is discussed in detail in the following definitions and remarks.

Definition 15 (Support function of a set)**.** *The support function of a set $\mathscr{S} \subset \mathscr{X}$ is the function*

$$\begin{aligned} h_{\mathscr{S}} : \mathscr{X}^\star &\to \mathbb{R} \\ y &\mapsto \sup_{x\in\mathscr{S}} \left\{ y^\mathsf{T} x \right\} \end{aligned}$$

This can be easily interpreted as that $x \in \mathscr{X}$ for which the projection on the half line $\{\alpha y \mid \alpha \in \mathbb{R}_+\}$ is maximal. Now, for any x we have $y^\mathsf{T} x = \|y\| \, \|x\| (\frac{y}{\|y\|})^\mathsf{T} \frac{x}{\|x\|} = \|y\| \, \|x\| \cos\theta$, with θ the angle formed by $\widehat{xOy}$. Hence, the orthogonal projection of x onto y is $\|x\| \cos(\theta) \frac{y}{\|y\|} = \frac{y^\mathsf{T} x}{\|y\|^2} y$.

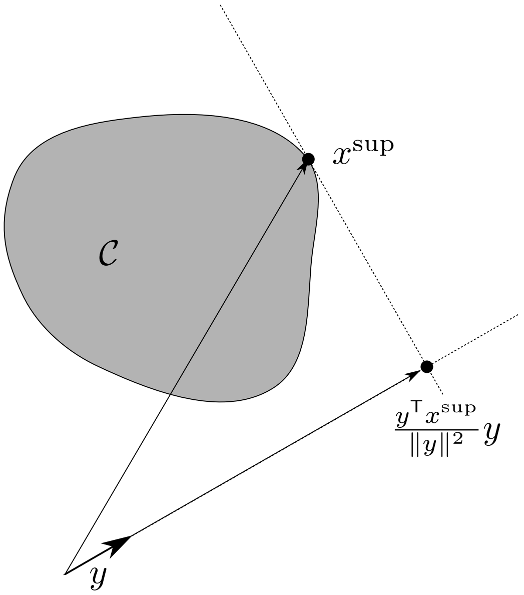


Figure 1.5: Support function of a convex set $\mathscr{C}$. $x^{\sup}$ is the vector for which the supremum is realized for a given y.

Definition 16 (Supporting hyperplanes of a set)**.** *The hyperplanes defined by*

$$\forall y \in \mathscr{X}^\star : \mathscr{H}_y = \left\{ \frac{h_{\mathscr{S}}(y)}{\|y\|^2} y + z \middle| z \in \mathscr{X}, y^\mathsf{T} z = 0 \right\}$$

are called the supporting hyperplanes of the set $\mathscr{S}$.

Remark 5 (Convex hull defined by a duality argument)**.** *The complement of the union of supporting hyperplanes of a set defines the interior of its convex hull. For a non-convex set, the set of all supporting hyperplanes define the complement of the interior of its convex hull.*

Property 7 (Boundary of a convex set)**.** *The set $\left\{ x^{\sup} \middle| x^{\sup} \in \arg\sup_{x\in\mathscr{C}} y^\mathsf{T} x, \forall y \in \mathbb{R}^n_* \right\}$ is the boundary $\partial\mathscr{C}$.*

Property 8 ("Infinite-dimensional" polyhedrons define convex sets)**.** *Any closed convex set $\mathscr{C}$ can be defined through an infinite-dimensional polyhedron, through the equations*

$$x \in \mathscr{C} \iff \forall y \in \mathscr{X}^\star : y^\mathsf{T} x \le h_{\mathscr{C}}(y)$$

The same holds for open convex sets with a strict inequality ('<' in place of '≤').

This property is the dual of the definition based on the relationship $x, y \in \mathscr{C} \implies (1-\lambda)x + \lambda y \in \mathscr{C}$. Indeed, one may define the convex set in the primal space through the membership definition (based on the convex combination), or one may define a convex set through the dual variable y using a system of inequalities.

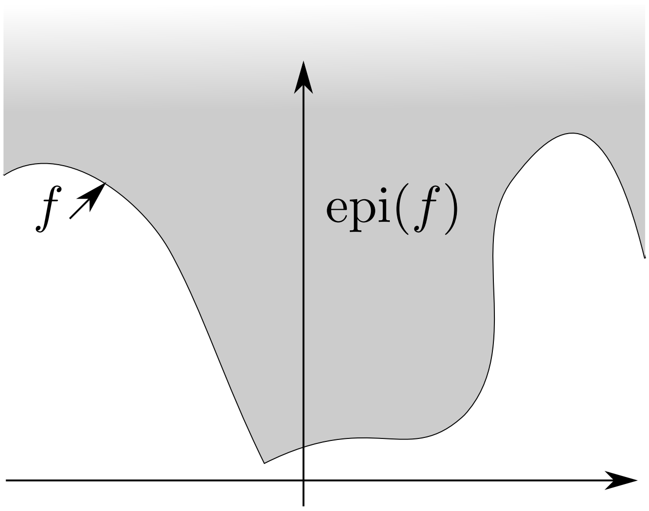


Figure 1.6: Epigraph of a function $f : \mathbb{R} \to \mathbb{R}$.

1.2.3 Convex Functions

Definition, first and second order conditions

Definition 17 (Epigraph of a function)**.** *The epigraph of a function $f : \mathcal{X} \to \mathbb{R}$ is the set*

$$\operatorname{epi}(f) = \left\{(x, c) \in \mathcal{X} \times \mathbb{R},\ f(x) \le c\right\}.$$

Definition 18 (Convex function)**.** *A function $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ is convex if the epigraph* epi(f) *is a convex set.*

Property 9 (Alternative definition of a convex function)**.** *A function $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ is convex if and only if, for all $x, y \in \mathcal{X}$ and $\lambda \in [0,1]$,*

$$f(\lambda x + (1-\lambda)y) \le \lambda f(x) + (1-\lambda) f(y). \tag{1.2}$$

Proof. $\Rightarrow$ Let $x, y \in \mathcal{X}$. Then $(x, f(x))$ and $(y, f(y))$ are in the epigraph of f. Thus so is $(\lambda x + (1-\lambda)y, \lambda f(x) + (1-\lambda) f(y))$. This by definition of the epigraph means that $\lambda f(x) + (1-\lambda) f(y) \ge f(\lambda x + (1-\lambda)y)$.
$\Leftarrow$ By the inequality, for $x, y \in \mathcal{X}$, $(\lambda x + (1-\lambda)y, \lambda f(x) + (1-\lambda) f(y))$ is in the epigraph of f, and so the epigraph is convex. □

To remember the direction of the inequality (1.2), the geometric interpretation of Figure 1.7 is convenient.

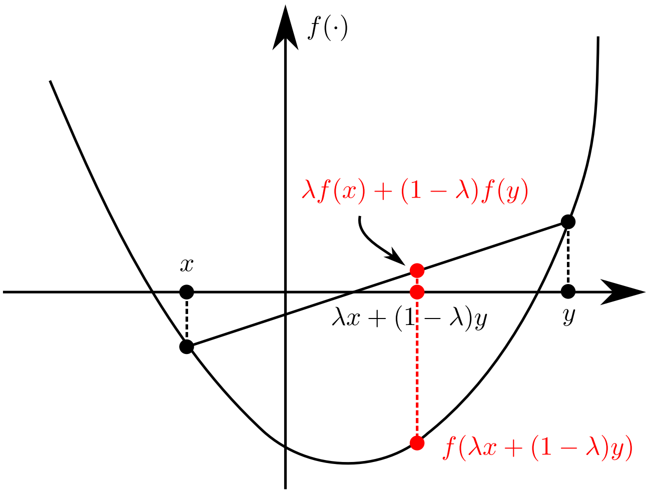

$f(\cdot)$
$\lambda f(x) + (1-\lambda) f(y)$
x
$\lambda x + (1-\lambda)y$
y
$f(\lambda x + (1-\lambda)y)$

Figure 1.7: Geometric interpretation of the convex function defining inequality (1.2).

From this (primal) inequality interpretation of convex sets, one naturally extends the notion to *strictly* convex functions.

Definition 19 (Strictly convex functions)**.** *A function f is said to be* strictly convex *if, for all $x, y \in \mathcal{X}$ and $\lambda \in [0,1]$,*

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y)$$

(which is the same as in (1.2) *but with a strict inequality sign).*

For *differentiable functions f*, recall that the gradient $\nabla f \in \mathbb{R}^n$ of f at x is defined as

$$\nabla f(x) = \left\{ \frac{\partial f}{\partial x_i}(x) \right\}_{i=1}^{n}.$$

Remark 6 (First intuitions on "gradient descent")**.** *For monovariate scenarios, where $n = 1$, $\nabla f(x) = f'(x)$, the derivative of f at $x \in \mathbb{R}$. Locally around x,*

$$f(x+\epsilon) = f(x) + \epsilon f'(x) + O(\epsilon^2)$$

and thus $f'(x)$ corresponds to the local increment of f at x. Similarly, for an arbitrary direction $h \in \mathbb{R}^n$, writing $g(\epsilon) = f(x+\epsilon h)$, we have

$$f(x+\epsilon h) = g(\epsilon) = g(0) + \epsilon g'(0) + O(\epsilon^2) = f(x) + \epsilon \nabla f(x)^{\mathsf{T}} h + O(\epsilon^2)$$

where we used the chain rule of derivation:

$$g'(t) = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \frac{dx_i}{dt} = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(x+th) h_i = \nabla f(x+th)^{\mathsf{T}} h.$$

As such, the projection of the gradient $\nabla f(x)$ onto h corresponds to the local evolution of f around x in the direction of *h.*

Note that, in particular, if $h = -\nabla f(x)$, then

$$f(x+\epsilon h) = f(x) - \epsilon \|\nabla f(x)\|^2 + O(\epsilon^2).$$

So, when $\|\nabla f(x)\| > 0$, for sufficiently small ϵ, $f(x+\epsilon h) < f(x)$ and thus we have found a direction *in $\mathcal{X}$ where f decreases. This is the root of the gradient descent algorithm.*

A first important condition for convexity of differentiable functions f is that all tangents of the epigraph of f fall below the epigraph (i.e., never intersect it). This follows after the duality approach to convex sets in the previous section

Definition 20 (Domain of a function)**.** *The domain of a function $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ is the set* $\mathrm{dom}(f) = \{x, f(x) < +\infty\}$.

Theorem 1 (First order conditions)**.** *Suppose $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ is differentiable in its domain. Then f is convex if and only if, for all $x, y \in$* $\mathrm{dom}(f)$*,*

$$f(y) \geq f(x) + \nabla f(x)^{\mathsf{T}} (y - x).$$

Proof. $\Rightarrow$ f is convex implies that, for $\lambda \in [0,1]$, $x, y \in \mathcal{X}$,

$$f((1-\lambda)x + \lambda y) = f(x + \lambda(y-x)) \leq (1-\lambda) f(x) + \lambda f(y) = f(x) + \lambda(f(y) - f(x))$$

or equivalently

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

Taking the limit $\lambda \to 0^+$ gives

$$\nabla f(x)^{\mathsf{T}} (y-x) \leq f(y) - f(x).$$

$\Leftarrow$ Denote $z = (1-\lambda)x + \lambda y$. Then we have both

$$(*)\ \ f(x) \geq f(z) + \nabla f(z)^{\mathsf{T}} (x - z)$$
$$(**)\ \ f(y) \geq f(z) + \nabla f(z)^{\mathsf{T}} (y - z).$$

Then $(1-\lambda)(*) + \lambda(**)$ gives

$$(1-\lambda) f(x) + \lambda f(y) \geq f(z) + \nabla f(z)^{\mathsf{T}} ((1-\lambda)x + \lambda y - z) = f((1-\lambda)x + \lambda y).$$

□

Geometrically, note that the hyperplane passing through $(x, f(x))$ is a supporting hyperplane whose equation is $(y, \nabla f(x)^\top(y-x)+C)$. The constant C can be found from the fact that $(x, f(x))$ is a point of the hyperplane (whence $C = f(x)$). Hence, falling below the epigraph is equivalent to saying that

$$f(x)+\nabla f(x)^\top(y-x) \leq f(y) \iff f(y)-f(x) \geq \nabla f(x)^\top(y-x).$$

See Figure 1.9 for a detailed proof for the monodimensional $n = 1$ case.

Exercise 5 (Geometric Interpretation for $n = 2$)**.** *Based on the geometric interpretation of Figure 1.9, retrieve the first order conditions for* $n = 2$.

Remark 7 (Strictly convex functions)**.** *For strictly convex functions the equality is met if and only if $x = y$. In that case, the couple $(y, \nabla f(x))$, where y is the intercept $f(x)-\nabla f(x)^\top x$, stands in a bijective relationship to $(x, f(x))$.*

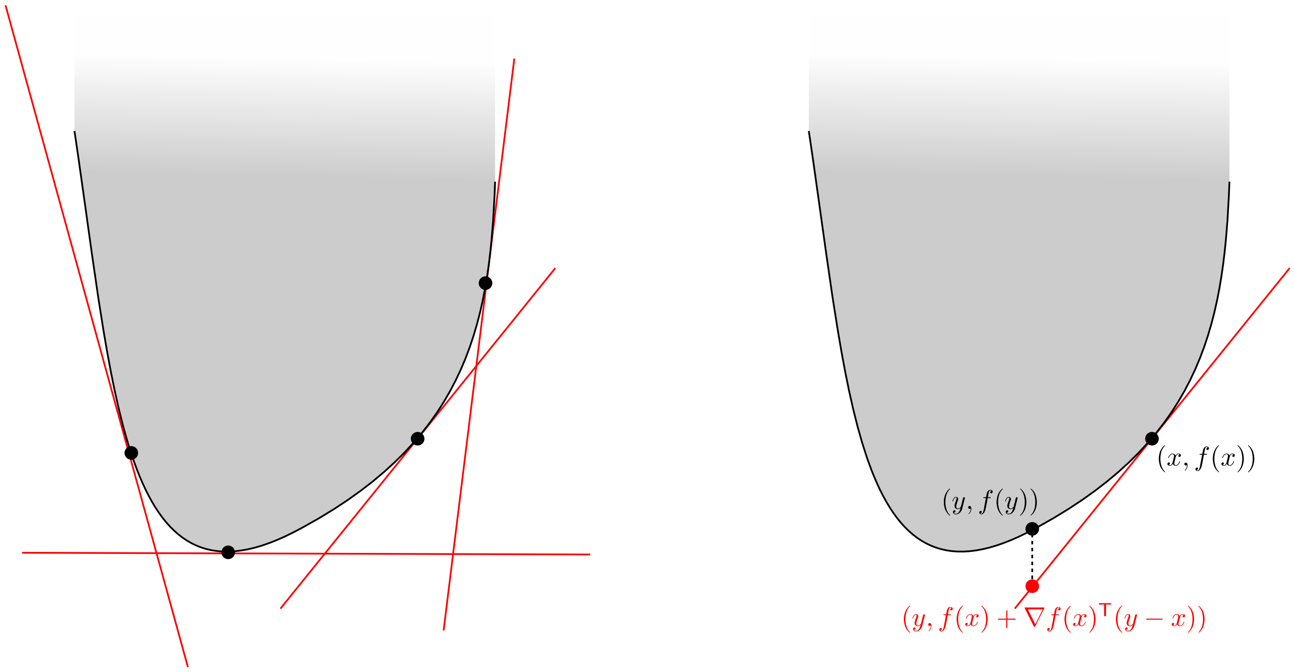


Figure 1.8: Geometric interpretation of the first order conditions for differentiable f: the supporting hyperplanes of the epigraph fall "below" the epigraph.

An important consequence of the theorem is the so-called *Fermat's rule*, given as follows:

Theorem 2 (Fermat's rule)**.** *For f a convex function, $x^\star \in \mathcal{X}$ minimizes $f : \mathcal{X} \to \mathbb{R}\cup\{+\infty\}$ if and only if* $\nabla f(x^\star) = 0$.

Proof. $\Rightarrow$ Assume $\nabla f(x^\star) \neq 0$. Then, for any $h \in \mathcal{X}$ and $\epsilon > 0$,

$$f(x^\star+\epsilon h) = f(x^\star)+\epsilon\nabla f(x^\star)^\top h + O(\epsilon^2)$$
$$f(x^\star-\epsilon h) = f(x^\star)-\epsilon\nabla f(x^\star)^\top h + O(\epsilon^2)$$

If $\nabla f(x^\star)^\top h \neq 0$, we have a contradiction in the small ϵ limit. So necessarily $\nabla f(x^\star)^\top h = 0$. But since this holds for all h, this implies $\nabla f(x^\star) = 0$.
$\Leftarrow$ If $\nabla f(x^\star) = 0$, then, since f is convex, for all $x \in \mathcal{X}$, $f(x) \geq f(x^\star)+\nabla f(x^\star)^\top(x-x^\star) = f(x^\star)$ so that $x^\star$ minimizes f. □

When $f : \mathcal{X} \to \mathbb{R}\cup\{+\infty\}$ is twice differentiable on its domain, second order conditions may also be considered as follows.

Theorem 3 (Second order conditions)**.** *For $f : \mathcal{X} \to \mathbb{R}\cup\{+\infty\}$ twice differentiable, define the Hessian $\nabla^2 f$ of f as the symmetric matrix*

$$\nabla^2 f(x) = \left\{\frac{\partial^2 f}{\partial x_i \partial x_j}\right\}_{i,j=1}^{n}.$$

Then, f is convex on its domain if and only if $\nabla^2 f(x)$ is semi-definite positive for all $x \in$ $\mathrm{dom}(f)$.

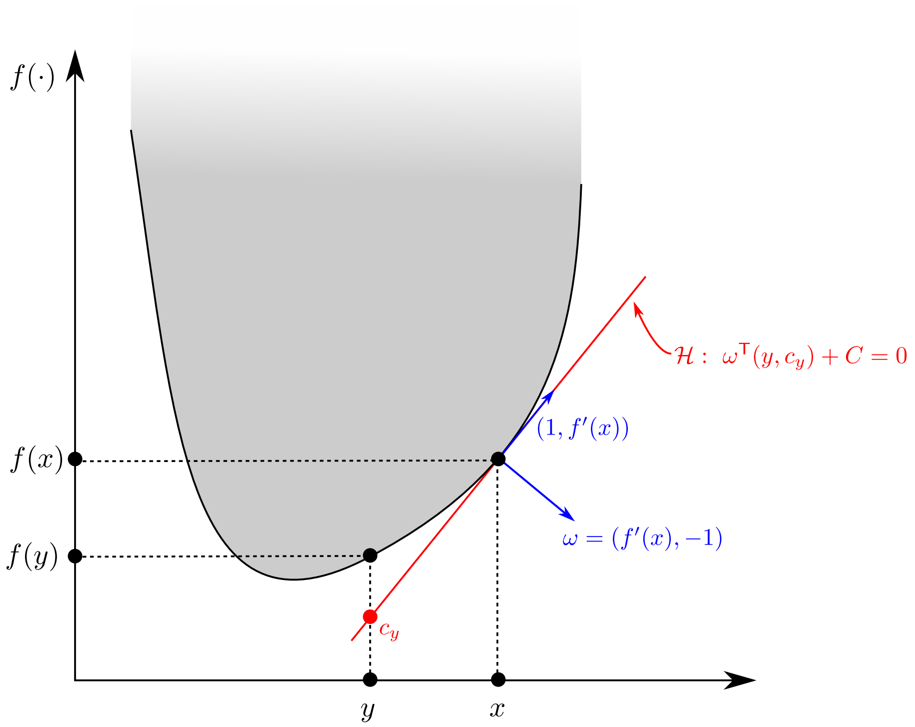


Figure 1.9: Detailed derivation of the first order conditions for $n = 1$: hyperplane $\mathcal{H}$ equation given by $\omega^\top(y,c_y)+C = 0$, with $(x, f(x)) \in \mathcal{H}$; hence $C = f'(x)x - f(x)$. Using $c_y \leq f(y)$, one retrieves the first order conditions.

Remark 8 (Case $n = 1$)**.** *For $n = 1$, $\nabla^2 f(x) = f''(x)$ and thus, f is convex if and only if $f''(x) > 0$, or equivalently if and only if $f'(x)$ is a non-decreasing function.*

Proof. $\Rightarrow$ By Taylor-Lagrange, for all $h \in \mathcal{X}$ and $\epsilon > 0$,

$$\exists\gamma \in (0,\epsilon),\ f(x+\epsilon h) = f(x)+\epsilon h^\top\nabla f(x)+\epsilon^2 h^\top\nabla^2 f(x+\gamma h)h$$

for some $\gamma \in [0,\epsilon]$ (this is readily obtained from Taylor-Lagrange for $\mathbb{R}\to\mathbb{R}$ functions by instead differentiating $g:\epsilon\mapsto f(x+\epsilon h)$). But, by convexity, we know that

$$f(x+\epsilon h) \geq f(x)+\epsilon\nabla f(x)^\top h$$

so that, since $\epsilon^2 > 0$, we find

$$\forall h \in \mathcal{X},\ h^\top\left[\nabla^2 f(x+\gamma h)\right]h \geq 0.$$

Since $\epsilon > 0$ can be chosen arbitrarily small, we obtain $\forall h \in \mathcal{X}$, $h^\top\left[\nabla^2 f(x)\right]h \geq 0$ and, finally, $\nabla^2 f$ is semi-definite positive.

$\Leftarrow$ For simplicity, it will be practical to work with the function $g:[0,1]\to\mathbb{R}\cup\{+\infty\}$

$$g(t) = f(tx+(1-t)y).$$

By the chain rule of derivation ($g'(t) = \sum_{i=1}^n \frac{\partial f}{\partial[z]_i}\frac{d[z]_i(t)}{dt}$ with $g(t) \equiv f(z(t))$, and similarly for $g''(t)$)

$$g''(t) = (x-y)^\top\left[\nabla^2 f(tx+(1-t)y)\right](x-y) \geq 0$$

since $\nabla^2 f$ is nonnegative definite. By a Taylor-Lagrange expansion, we then have, for some $\zeta_x,\zeta_y \in [0,1]$,

$$(*)\ f(y) = g(0) = g(t)+(0-t)g'(t)+\frac{1}{2}t^2 g''(\zeta_y) \geq g(t)-tg'(t)$$
$$(**)\ f(x) = g(1) = g(t)+(1-t)g'(t)+\frac{1}{2}t^2 g''(\zeta_x) \geq g(t)+(1-t)g'(t).$$

Using $(1-t)(*)+t(**)$ leads to

$$tf(x)+(1-t)f(y) \geq g(t) = f(tx+(1-t)y).$$

□

To go further: support function of epi_f and Fenchel conjugate

Since convex functions may be primarily defined through their convex epigraph, and that convex sets may themselves be defined through the set of their tangential hyperplanes, we have the following set of remarks and properties.

Definition 21 (Fenchel convex conjugate)**.** *For all vectors $(y,-1)$ and all functions f (not necessarily convex), we may evaluate the support function of the convex set* $\mathrm{epi} f$*, i.e.,* $h_{\mathrm{epi} f}((y,-1)) = \sup_x y^\top x - f(x) \equiv f^*(y)$. *We call f^* the Fenchel convex conjugate of the function f.*

Proof of convexity of the Fenchel convex conjugate. Let y, z be in $\mathbb{R}^n$ and f a function, then

$$\begin{aligned}\sup_x((1-\lambda)y+\lambda z)^\top x - f(x) &= \sup_x (1-\lambda)y^\top x + \lambda z^\top x - f(x)\\ &\le (1-\lambda)\left[\sup_u y^\top u - f(u)\right] + \lambda\left[\sup_x z^\top x - f(x)\right]\end{aligned}$$

where the inequality follows from a *triangular inequality* for the sup operator. □

The fact that only the tuples $(y, -1)$ are considered follows from the epigraph being bounded below but not above. Hence, we only consider the hyperplanes that support the function from below, which is exactly the first order condition in Theorem 1.

Property 10 (Biconjugate inequality)**.** *For all f, we have $f^{**} \le f$.*

Proof.

$$\begin{aligned}\left(f^*\right)^*(z) &= \sup_y z^\top y - \left(\sup_x y^\top x - f(x)\right)\\ &= \sup_y \inf_x z^\top y + f(x) - y^\top x\\ &\le \inf_x \sup_y f(x) + (z-x)^\top y\\ &= f(z)\end{aligned}$$

□

Theorem 4 (Fenchel-Moreau)**.** *Let $f : \mathbb{R} \to \mathbb{R} \cup \{\pm\infty\}$, then $f = f^{\star\star}$ if and only if f is a proper (i.e., $f(\mathcal{X})$ does not contain $-\infty$), lower semi-continuous (i.e., $\forall x_0 \in \mathcal{X}$, $\lim\inf_{x\to x_0} f(x) \ge f(x_0)$), convex function or $f \equiv +\infty$ or $f \equiv -\infty$.*

Example 5 (Fenchel conjugate of the indicator function)**.** *Let $f(x) = \imath_{\mathcal{B}_\infty}(x)$, where $\imath_{\mathcal{B}_\infty}$ is the indicator function of the unit ℓ_∞-ball, i.e., $\imath_{\mathcal{B}_\infty}(x) = \begin{cases} 0 & \text{if} \max_j |x_j| \le 1 \\ +\infty & \text{otherwise}\end{cases}$. Then, $f^*(y) = \sup_x y^\top x - \imath_{\mathcal{B}_1}(x) = \sup_{x\in\mathcal{B}_1} y^\top x = \sum_j |y_j| = \|y\|_1$ and $f^{**}(x) = \sup_y x^\top y - \|y\|_1$, which is a separable problem that can be written in each of its variables $\sup_{y_i} x_i y_i - |y_i|$. Now, if $|x_i| \le 1$ we have $f^{**}(x_i) = 0$, and for $|x_i| > 1$ we have $f^{**}(x_i) = +\infty$, hence we have the indicator function of the unit ℓ_∞-ball.*

Exercise 6 (Computing the Fenchel conjugate of norms)**.** *Show that for $p \ge 1$ and q such that $\frac{1}{p} + \frac{1}{q} = 1$, we have that $f_p(x) = \|x\|_p$ implies $f_p^*(y) = \imath_{\mathcal{B}_q}(y)$, and vice versa.*

If the function f is not convex, then the convex conjugate of its convex conjugate f^{**} corresponds to the convex hull of the function f (see also Remark 5). This may be used to find an approximation to a non-convex function to which convex algorithms could then be applied.

Exercise 7 (Convex hull of a non-convex function)**.** *Let $f(x) = |x^2 - 1|$. Compute its convex hull by using the Fenchel conjugate of its Fenchel conjugate.*

Property 11 (Fenchel convex conjugate and first order optimality)**.** *If $f(x^*)$ is a minimum of f, then $-f$ attains its maximum at $x^\star$, and hence $f^*(0) = \sup_x(0^\top x) - f(x)$. This means that the hyperplane parallel to the coordinate axes supports the epigraph in $x^\star$. In particular, for a differentiable function this implies that the derivative vanishes at $x^\star$.*

1.3 Basic Algorithms for Convex Optimization

Recall that our objective is to solve the problem

$$x^\star \in \operatorname{argmin}_{x\in\mathcal{X}} f(x).$$

We shall here concentrate on the simple case where $\mathcal{X} = \operatorname{dom}(f)$ is an unbounded open set (generally $\mathcal{X} = \mathbb{R}^n$) and f is convex and differentiable. This allows us to leverage two convenient properties:

- being differentiable, the gradient of f can be evaluated at any point of $\mathcal{X}$ and, as we shall see, this allows one to perform *gradient descent* steps;
- while performing gradient descent steps, no constraint will limit the progression of the descent.

(More interesting) cases when either one of these assumptions is not met will be covered in subsequent sections of the course.

We shall proceed by solving the optimization problem at hand by *iterative* algorithms. That is, algorithms that sequentially evaluate f at positions $x_1, x_2, \ldots$ with x_{k+1} being a function of x_k. The algorithm then terminates under one of the following conditions:

- either $\|x_{k+1} - x_k\| < \epsilon$ for some predetermined $\epsilon > 0$: this suggests that the algorithm no longer progresses in its exploration of $\mathcal{X}$;
- either $\left|f(x_{k+1}) - f(x_k)\right| < \epsilon$: this here suggests that the *value* associated to the sequence of points explored no longer progresses (which does not necessarily imply that x_k is close to x_{k+1}; see for instance the case of a convex function plateauing at its minimum);
- either $\left\|\nabla f(x_k)\right\| < \epsilon$: this suggests that the level reached is almost flat, thus one leverages the first order condition that $\nabla f(x^*) = 0$ under the hypothesis that the gradient changes smoothly.

1.3.1 Descent and Gradient Descent Algorithms

Descent algorithms are iterative methods which ensure that for a sequence $x_1, x_2, \ldots$, we have $f(x_{k+1}) \le f(x_k)$.

Definition 22 (Descent Method)**.** *A descent method is an algorithm producing a sequence $x_1, x_2, \ldots \in \mathcal{X}$ of the form*

$$x_{k+1} = x_k + t_k \Delta x_k$$

for $t_1, t_2, \ldots > 0$ in such a way that $f(x_{k+1}) < f(x_k)$ if $x_k \notin \operatorname{argmin} f$ and $f(x_{k+1}) = f(x_k)$ if $x_k \in \operatorname{argmin} f$. We call t_k the step size *at iteration k and Δx_k the* increment.

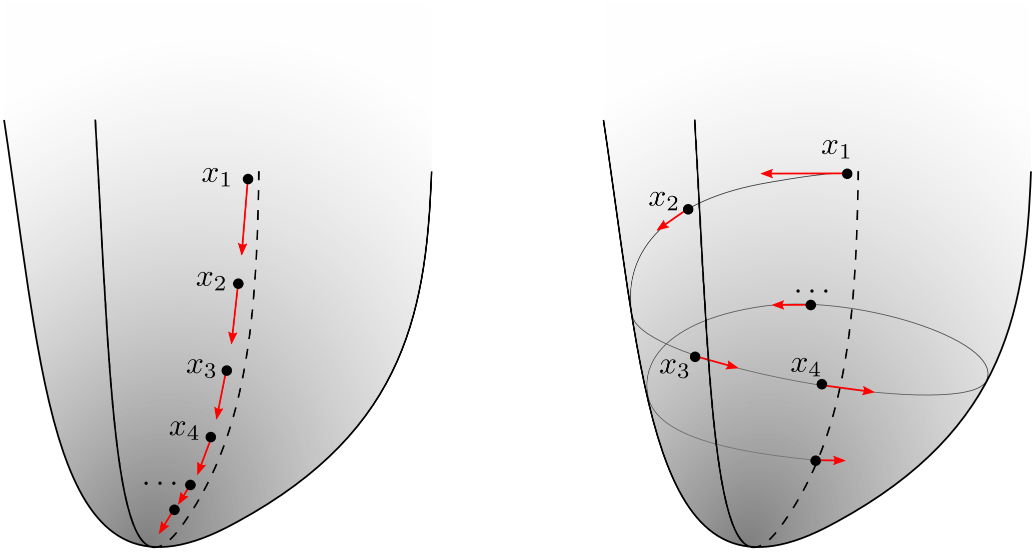


Figure 1.10: Examples of descent methods.

Remark 9 (Convergence of descent algorithms)**.** *For a function $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ having a non-empty set of minima, descent algorithms necessarily converge,* however *not necessarily to a local minimum. In particular,*

- *the descent may progress too slowly to reach a minimum, in which case* $\lim_k f(x_k) > f(x^\star)$ *for at least one local minimum* $x^\star$*;*
- *even if* $f(x_k) \to f(x^\star)$*, this does not imply that* x_k *converges at all (*x_k *may have a periodic behaviour).*

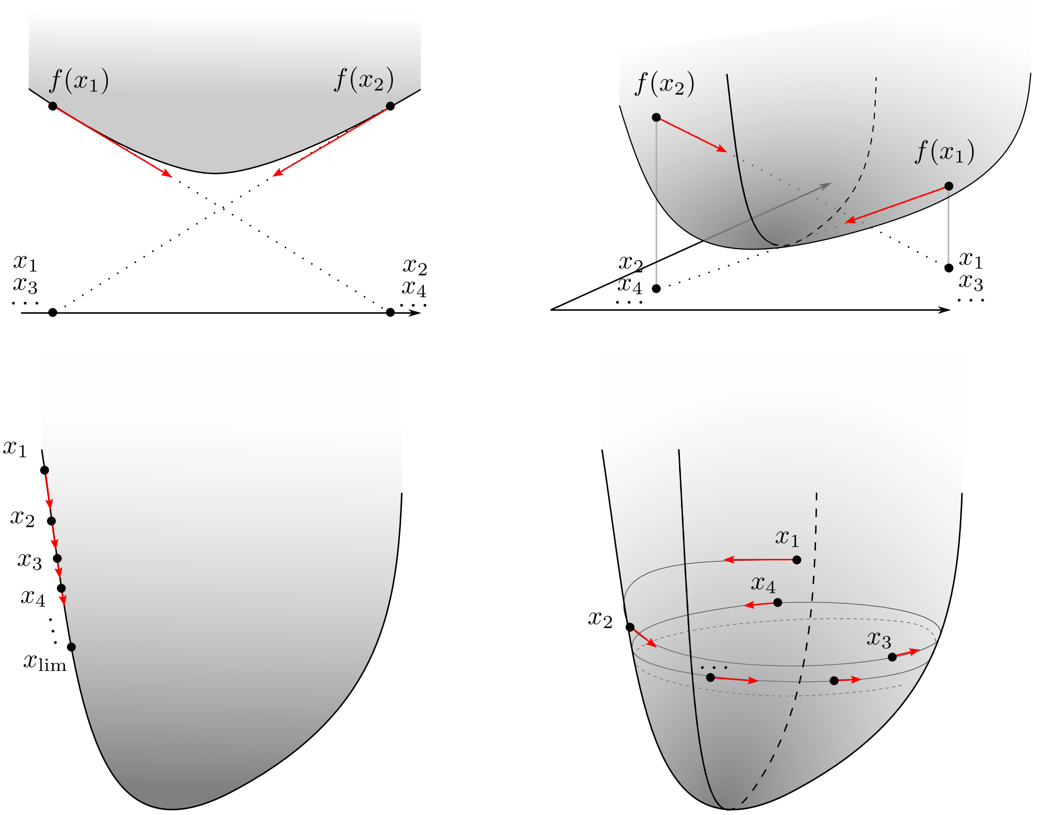


Figure 1.11: Descent sequences either not converging (top) or not reaching minimum (bottom).

An important property of the gradient of a convex function lies in the following remark. For two arbitrary points $x_k, x_{k+1} \in \mathscr{X}$, since f is convex and differentiable, we know from the first order condition that

$$f(x_k + t_k \Delta x_k) \geq f(x_k) + t_k \nabla f(x_k)^{\mathsf{T}} \Delta x_k.$$

As such, letting $x_1, x_2, \ldots$ be the sequence defined by

$$x_{k+1} = x_k + t_k \Delta x_k$$

for all $k \geq 1$, for some arbitrary x_1 and $t_1, t_2, \ldots > 0$, we have

$$f(x_{k+1}) \geq f(x_k) + \nabla f(x_k)^{\mathsf{T}} (x_{k+1} - x_k) = f(x_k) + t_k \nabla f(x_k)^{\mathsf{T}} \Delta x_k$$

and thus $x_1, x_2, \ldots$ cannot be a descent method sequence unless $\nabla f(x_k)^{\mathsf{T}} \Delta x_k \leq 0$.

Property 12 (Descent direction)**.** *A necessary condition for* $x_1, x_2, \ldots$ *to be a sequence generated by a descent method is that*

$$\nabla f(x_k)^{\mathsf{T}} \Delta x_k \leq 0$$

where $\Delta x_k = x_{k+1} - x_k$*, and equality is reached if and only if* $x_k \in \operatorname{argmin} f$*.*

Note that the condition is not sufficient! For instance, consider the function $f(x) = [x]_1^2 + [x_2]^2$. At point $x_1 = [1, 1]$, $f(x_1) = 2$ and $\nabla f(x_1) = [2, 2]$. One may consider next evaluating the point $x_2 = [0, 3/2]$ for which $\Delta x_1 = [-1, 1/2]$ and thus $\nabla f(x_1)^{\mathsf{T}} \Delta x_1 = -2 + 1 < 0$. But $f(x_2) = 9/4 > 2$. See Figure 1.12 for a visualization of this example.

Nonetheless, the condition is "locally sufficient" when small steps are taken *and* when f is twice differentiable in the neighborhood of the considered iteration; indeed, by a Taylor expansion

$$f(x_{k+1}) = f(x_k) + t_k \nabla f(x_k)^{\mathsf{T}} \Delta x_k + O(t_k^2 \|\Delta x_k\|^2)$$

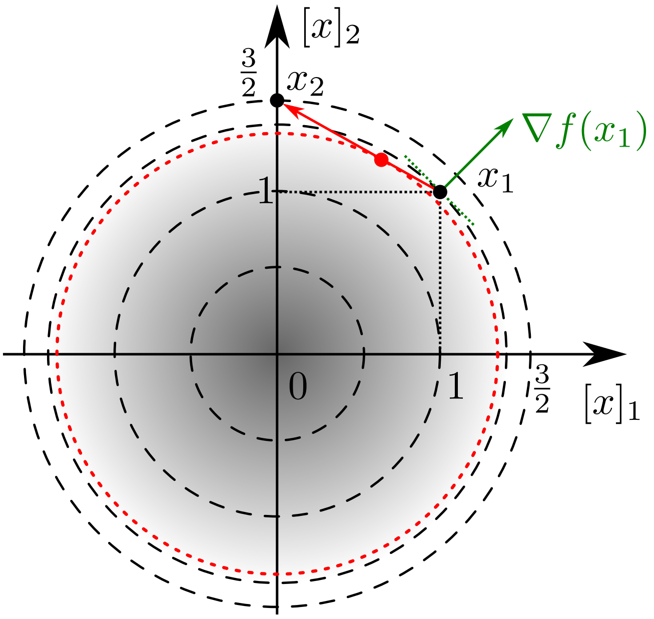


Figure 1.12: Function $f(x) = [x]_1^2 + [x]_2^2$ and optimization algorithm initialized at $x = [1,1]$. Although $\Delta x_1 = [-1, 1/2]$ has an acute angle with $-\nabla f(x_1)$, $x_2 = [0,3/2] = x_1 + \Delta x_1$ increases rather than decreases the value of f. Yet, for small enough t, $x_1 + t\Delta x_1$ provides a descent direction (see red circle mark for instance).

and thus, for unit norm Δx_k and all sufficiently small $t_k > 0$, $f(x_{k+1}) < f(x_k)$. It is thus important in what follows to *carefully control the step sizes.*

This being said, it also appears that, still in the limit of small step sizes, the iteration gain $|f(x_{k+1}) - f(x_k)|$ is maximal when $\nabla f(x_k)^{\mathsf{T}}\Delta x_k$ is both negative and of maximal absolute value. For unit norm Δx_k, this is obviously optimal when

$$\Delta x_k = -\frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}.$$

This leads to the popular *gradient descent* algorithm defined below.

Definition 23 (Gradient Descent Algorithm). *The gradient descent algorithm is described by $x_1 \in \mathcal{X}$ and, for all $k \geq 1$,*

$$x_{k+1} = x_k - t_k \nabla f(x_k)$$

for some sequence $t_1, t_2, \ldots > 0$.

The descent algorithms are said to be *constant step* descents if $t_k = t$ is constant. Constant steps are practical as they do not request often complicated fine-tuning of the t_k step sizes. Yet, they are far from optimal. Next, we give a practical condition to test whether a step size is appropriate for a given descent direction. This will then in turn allow to design appropriate *adaptive step size* algorithms.

Definition 24 (Armijo-Goldstein condition for descent). *For a given Δx_k, and control parameter $\alpha \in (0,1)$ with $\|\Delta x_k\| = 1$ and $\nabla f(x_k)^{\mathsf{T}}\Delta x_k < 0$, we say that t_k satisfies the Armijo-Goldstein condition if*

$$f(x_k + t_k\Delta x_k) < f(x_k) + \alpha t_k \nabla f(x_k)^{\mathsf{T}}\Delta x_k$$

thus necessarily producing a descent sequence $x_1, x_2, \ldots$.

Remark 10 (On step size adaptation). *Several approaches lead to improved step size setting, among which:*

- **[Line search]** *t_k is chosen to be such that $f(x_k + t_k\Delta x_k)$ is minimized. This implies either being capable of solving the intermediary optimization problem*

  $$t_k \in \operatorname{argmin}_{t>0} f(x_k + t\Delta x_k)$$

  or proceeding through an exhaustive (and generally costly) line search (for instance by jumps and dichotomy);

- **[Backtracking]** *A simplified version of the line search consists in letting* $t^{(0)} = 1$ *and, for some* $0 < \beta < 1$*, iterating* $t^{(j+1)} = \beta t^{(j)}$ *until*

  $$f(x_k + t^{(j+1)}\Delta x_k) < f(x_k) + \alpha t^{(j+1)} \nabla f(x_k)^\mathsf{T} \Delta x_k$$

  for some $\alpha \in (0,1)$*, therefore meeting the Armijo-Goldstein condition of Definition 24. This is always achievable since, for sufficiently small* $t^{(j)}$*,* $f(x_k + t^{(j+1)}\Delta x_k) \simeq f(x_k) + t^{(j)}\nabla f(x_k)^\mathsf{T}\Delta x_k < f(x_k) + \alpha t^{(j+1)}\nabla f(x_k)^\mathsf{T}\Delta x_k$*. This is passing justifies the importance of* $\alpha < 1$ *(without which the latter inequality may never be met).*

The popularity of the gradient descent algorithm lies in its guaranteed and rather fast convergence under sufficient regularity settings.

Theorem 5 (Convergence of Gradient Descent with Constant Step Size)**.** *Let* $f : \mathscr{X} \to \mathbb{R} \cup \{+\infty\}$ *be convex, twice continuously differentiable and such that* ∇f *is L-Lipschitz, i.e.,*

$$\forall x, y \in \mathscr{X} \qquad \|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|.$$

Then the gradient descent algorithm with constant step size $t \le 1/L$ *convergences to a minimum of* f*, i.e.,*

$$x_k \to x^\star \in \operatorname{argmin}_x f(x).$$

Proof. Since f is twice continuously differentiable, first observe that the Lipschitz condition on ∇f implies that $\nabla^2 f(x) \preceq L I_n$ in the order of symmetric matrices (i.e., $\|\nabla^2 f(x)\| \le L$ in spectral norm). To see this, let $x \in \mathscr{X}$ and $u \in \mathscr{X}$, and write the two relations

$$f(x + \epsilon u) = f(x) + \epsilon \nabla f(x)^\mathsf{T} u + \frac{1}{2}\epsilon^2 u^\mathsf{T} \nabla^2 f(x) u + o(\epsilon^2)$$
$$f(x) = f(x+\epsilon u) - \epsilon \nabla f(x+\epsilon u)^\mathsf{T} u + \frac{1}{2}\epsilon^2 u^\mathsf{T}\nabla^2 f(x+\epsilon u) u + o(\epsilon^2).$$

Summing the two relations and dividing by ϵ^2 leads to

$$\frac{(\nabla f(x+\epsilon u) - \nabla f(x))^\mathsf{T} u}{\epsilon} = \frac{1}{2} u^\mathsf{T}(\nabla^2 f(x) + \nabla^2 f(x+\epsilon u))u + o(1).$$

By the Cauchy-Schwarz inequality and using the Lipschitz condition,

$$\frac{1}{2}u^\mathsf{T}(\nabla^2 f(x) + \nabla^2 f(x+\epsilon u))u + o(1) \le \frac{\|\nabla f(x+\epsilon u) - \nabla f(x)\|\|u\|}{\epsilon} \le L\|u\|^2.$$

By continuity of the second derivative, taking now the limit $\epsilon \to 0$, this leads to

$$u^\mathsf{T}\nabla^2 f(x) u \le L\|u\|^2$$

which, since $u \in \mathscr{X}$ is arbitrary, ensures that $\nabla^2 f(x) \preceq L I_n$.

We now get to the core of the proof. Since f is convex (first order condition $(*)$) and $\nabla^2 f(x) \preceq L I_n$ (Taylor-Lagrange $(**)$), we have, for arbitrary $x, y \in \mathscr{X}$, the following relations

$$\begin{aligned}
(*)\ f(y) &\ge f(x) + \nabla f(x)^\mathsf{T}(y-x) \\
(**)\ f(y) &= f(x) + \nabla f(x)^\mathsf{T}(y-x) + \frac{1}{2}(y-x)^\mathsf{T}\nabla^2 f(\zeta)(y-x) \\
&\le f(x) + \nabla f(x)^\mathsf{T}(y-x) + \frac{1}{2}L\|y-x\|^2
\end{aligned}$$

(where we took $\zeta \in \mathscr{X}$ of the form $\zeta = x + \lambda(y-x)$ for some $\lambda \in [0,1]$).

As a consequence of $(**)$,

$$\begin{aligned}
f(x_{k+1}) &\le f(x_k) + \nabla f(x_k)^\mathsf{T}(x_{k+1} - x_k) + \frac{1}{2}L\|x_{k+1} - x_k\|^2 \\
&= f(x_k) - t\|\nabla f(x_k)\|^2 + \frac{1}{2}Lt^2\|\nabla f(x_k)\|^2 \\
&= f(x_k) - \left(1 - \frac{1}{2}Lt\right) t\|\nabla f(x_k)\|^2.
\end{aligned}$$

This is where we exploit the fact that $t \le 1/L$, since then

$$f(x_{k+1}) \le f(x_k) - \frac{t}{2}\|\nabla f(x_k)\|^2 (\le f(x_k)) \tag{1.3}$$

the second inequality being an equality only if $\nabla f(x_k) = 0$. So the gradient descent algorithm *is* a descent algorithm.

Yet the convergence to the minimum of f is not yet guaranteed. To show this, use now $(*)$ to see that, for any $x^\star \in \operatorname{argmin} f$ and any $x \in \mathscr{X}$,

$$f(x^\star) \geq f(x) + \nabla f(x)^\mathsf{T}(x^\star - x)$$

or equivalently

$$f(x) \leq f(x^\star) + \nabla f(x)^\mathsf{T}(x - x^\star).$$

So in particular, from (1.3)

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) - \frac{t}{2}\|\nabla f(x_k)\|^2 \\ &\leq f(x^\star) + \nabla f(x_k)^\mathsf{T}(x_k - x^\star) - \frac{t}{2}\|\nabla f(x_k)\|^2. \end{aligned}$$

To pursue the inequality, we need to relate $\nabla f(x_k)^\mathsf{T}(x_k - x^\star)$ to $t\|\nabla f(x_k)\|^2$, which may be done using

$$\|x_k - x^\star - t\nabla f(x_k)\|^2 = \|x_k - x^\star\|^2 + t^2\|\nabla f(x_k)\|^2 - 2t\nabla f(x_k)^\mathsf{T}(x_k - x^\star)$$

which then yields

$$f(x_{k+1}) \leq f(x^\star) + \frac{1}{2t}\Big(\|x_k - x^\star\|^2 - \|\underbrace{x_k - t\nabla f(x_k)}_{x_{k+1}} - x^\star\|^2\Big).$$

Summing up over $k = 1, \ldots, K$, we have a telescoping sum on the right-hand side, leading up to

$$\underbrace{\sum_{k=1}^{K} f(x_{k+1}) - f(x^\star)}_{\geq K(f(x_K) - f(x^\star))} \leq \frac{1}{2t}\left(\|x_1 - x^\star\|^2 - \|x_K - x^\star\|^2\right) \leq \frac{1}{2t}\|x_1 - x^\star\|^2.$$

So finally

$$f(x_K) - f(x^\star) \leq \frac{1}{2Kt}\|x_1 - x^\star\|^2 \to 0$$

as $K \to \infty$. Thus, while x_K may not converge, at least $f(x_K)$ asymptotically reaches $f(x^\star)$. □

Remark 11 (Advantages and limitations of gradient descent)**.** *The following observations are important to understand the popularity of gradient descent while preserving a conservative approach on its actual performance:*

- *the algorithm is simple to implement, each step being computationally cheap as only a gradient needs be evaluated; for functions not easily differentiable, a finite element approach may be performed, leading up to a gradient approximation* $\{(f(x_k + \epsilon e_i) - f(x_k))/\epsilon\}_{i=1}^n$ *with* $[e_i]_j = \delta_i^j$ *the* i*-th canonical vector;*
- *the algorithm is also quite flexible and generalizes in a myriad of ways; for instance, when* f *itself is not perfectly known, stochastic versions of gradient descent allow for an approximation at each step of the gradient which, due to linearity, averages well on the long run; this is at the core in particular of neural network learning;*
- *convergence is ensured for fixed step sizes and thus no step size adaptation is required;*
- *on the downside, gradient descent* requires *a sufficient small step size* ($< 1/L$) *which, in most cases, is difficult to evaluate; one must therefore be rather conservative on the step size, and hence the convergence speed, to avoid divergence;*
- *besides, the constraints on* f *are rather strong: the fact that the Hessian* $\nabla^2 f$ *is bounded (as a consequence of the Lipschitz character of the gradient) means in particular that* f *does not have "super-quadratic" regions (i.e., places where* f *grows faster than* $\|x\|^2$*); when non anticipated, functions* f *with very steep regions on at least one coordinate risk a bouncing of the iterates not guaranteeing the descent character of the algorithm any longer;*
- *also, the function* f *needs be everywhere differentiable (or at least almost everywhere if it can be guaranteed that no zero measure regions can be reached by descent steps) for a gradient to be evaluated;*

- *finally, the fact that the optimization is unconstrained and that $\mathcal{X}$ is unbounded ensures that every $x_k + t\nabla f(x_k)$ remains within the domain of f; in constrained scenarios, the algorithm may demand to evaluate subsequent points outside of the domain. Natural solutions exist to solve this difficulty (such as reducing t to reach at most the boundary of the domain* $\mathrm{dom}(f)$ *of f) but these may hinder the convergence (in the aforementioned case, by leaving the iterations stuck on a boundary).*

Exercise 8 (Importance of the Lipschitz constant)**.** *In order to emphasize the importance of the Lipschitz constant, consider the following optimization problem*

$$\min_{x\in\mathbb{R}} f(x)$$

where

$$f(x) = \begin{cases} |x|, & |x| > \frac{1}{2} \\ x^2 + \frac{1}{4}, & |x| \leq \frac{1}{2}. \end{cases}$$

Confirm that f is everywhere differentiable. Then, start a gradient descent from $x_1 = \frac{1}{2}$ with step size $t = 1$; describe what happens and explain the reasons. Adapt the conditions to fit in the setting of Theorem 5 and show numerically that convergence to the minimum ($x^\star = 0$) then holds.

Note importantly from the proof of Theorem 5 that the convergence speed satisfies at least

$$f(x_k) - f(x^\star) \leq \frac{1}{2kt}\|x_1 - x^\star\|^2.$$

In the worst case, it thus takes 100 steps to fall within 1% of the initialization error level. In optimization, this is a typically slow convergence rate, which is qualified as *sublinear*.

But much faster convergence rates are usually ensured for the gradient descent algorithm, under the extra assumption that the landscape of f is never “too flat”, that is by demanding that $\nabla^2 f \succeq lI_n$ for some $l > 0$.

Theorem 6 (Linear Convergence of Gradient Descent)**.** *Let $f : \mathcal{X} \to \mathbb{R}\cup\{+\infty\}$ be convex, twice continuously differentiable, and such that, for all $x \in \mathcal{X}$,*

$$lI_n \preceq \nabla^2 f(x) \preceq LI_n$$

for some $L \geq l > 0$. Then, for the constant step size gradient descent algorithm with step size $t \leq \frac{1}{L}$,

$$f(x_k) - f(x^\star) \leq \alpha C^k$$

for some $C < 1$. We say here that the convergence is linear.

Proof. We have already seen in (1.3) that, since $t \leq \frac{1}{L}$,

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2}t\|\nabla f(x_k)\|^2$$

which implies

$$f(x_{k+1}) - f(x^\star) \leq f(x_k) - f(x^\star) - \frac{1}{2}t\|\nabla f(x_k)\|^2. \tag{1.4}$$

By a Taylor expansion, we also know that, for $x, y \in \mathcal{X}$,

$$\begin{aligned} f(y) &= f(x) + \nabla f(x)^\mathsf{T}(y-x) + \frac{1}{2}(y-x)^\mathsf{T}\nabla^2 f(\zeta)(y-x) \\ &\geq f(x) + \nabla f(x)^\mathsf{T}(y-x) + \frac{l}{2}\|y-x\|^2 \end{aligned}$$

for some $\zeta \in \mathcal{X}$, where in the inequality we used $\nabla^2 f \succeq lI_n$. Now, the right-hand side term is minimized for $y = x - \frac{1}{l}\nabla f(x)$ (it suffices to differentiate along y). And thus, for all $x, y \in \mathcal{X}$,

$$f(y) \geq f(x) - \frac{1}{2l}\|\nabla f(x)\|^2.$$

Applying this identity to $y = x^\star$ and $x = x_k$, we get

$$-\frac{t}{2}\|\nabla f(x_k)\|^2 \leq tl(f(x^\star) - f(x_k))$$

Getting back to (1.4), this implies

$$f(x_{k+1}) - f(x^\star) \leq (1 - tl)\,(f(x_k) - f(x^\star))$$

with $1 - tl = C < 1$ by assumption. And thus, applying the inequality subsequently for $k = 1, \ldots, K$, we find

$$f(x_{K+1}) - f(x^\star) \leq C^K (f(x_1) - f(x^\star))$$

which is the expected result (with $\alpha = f(x_1) - f(x^\star)$). □

Remark 12 (On convergence modes)**.** *Geometric convergence modes are often met in optimization. As such, the convergence speed is often evaluated in exponential terms. Specifically, the convergence is* linear *if* $f(x_k) - f(x^\star) \leq e^{-(ak+b)}$ *for some* $a, b > 0$*, or equivalently if* $\log(f(x_k) - f(x^\star)) \leq -(ak + b)$*. The convergence is* quadratic *if* $\log(f(x_k) - f(x^\star)) \leq -(ak^2 + bk + c)$ *for some* $a, b, c > 0$.

In particular, if the stopping criterion for the descent algorithm is $f(x_k) - f(x^\star) < \epsilon$ *for some small* $\epsilon > 0$*, then under linear convergence, an upperbound on the number of steps* k_ϵ *needed to achieve this accuracy is*

$$k_\epsilon > \frac{1}{a}\log\left(\frac{1}{\epsilon}\right) - \frac{b}{a}.$$

Remark 13 (Variational perspective on the gradient/steepest descent)**.** *Let*

$$h_{x_k}(y) = f(x_k) + \nabla f(x_k)^\mathsf{T}\left(y - x_k\right) + \frac{1}{2t}\|y - x_k\|^2 \approx f(y)$$

then we have

$$x_{k+1} = x_k - t\nabla f(x_k) = \arg\min_y h_{x_k}(y) \ .$$

1.3.2 Newton's Method

The gradient descent algorithm is sometimes referred to as the *steepest descent method* in the sense that it corresponds to the unit-norm Δx_k that maximizes the product $\nabla f(x_k)\Delta x_k$. Yet, one must be careful that *steepest descent* may not be synonymous for *fastest descent.* Fundamentally, the gradient descent approach assumes a *local linear behavior* of the function f and is "fastest" under validity of this assumption. In particular, gradient descent is indeed fastest (step size adaptation not accounted for) on hyperplane-shaped cost functions, as shown on the left-hand side of Figure 1.13.

When it comes to smoother cost functions, starting with quadratic f, it seems natural to further approximate f locally by a next-order (quadratic) Taylor expansion and follow the direction, not of the steepest descent, but of the local minimum of this quadratic approximation. This is the idea behind Newton's method. As displayed on the right-end side of Figure 1.13, Newton's method is understandably much faster than gradient descent in non-flat regions of f.

Yet, one must be very careful when exploiting the method which, although fast once in a neighbourhood of the optimum, may have a slow start if not well initialized.

The intuition behind Newton's method is to write the second-order Taylor expansion of f in the neighborhood of $x \in \mathscr{X}$ as

$$f(x+h) = \underbrace{f(x) + \nabla f(x)^\mathsf{T} h + \frac{1}{2}h^\mathsf{T}\nabla^2 f(x)h}_{\equiv \hat{f}(x+h)} + o(\|h\|^2).$$

The idea is to approximate $f(x+h)$ by $\hat{f}(x+h)$ in the neighborhood of every $x \in \mathscr{X}$ and to solve a local minimization of $f(x+h)$ through the minimization of $\hat{f}(x+h)$ with respect to h .

In particular, $\hat{f}(x+h)$ is minimized for

$$h = -[\nabla^2 f(x)]^{-1}\nabla f(x)$$

provided $\nabla^2 f$ is invertible.

Hence the Newton's method, as follows:

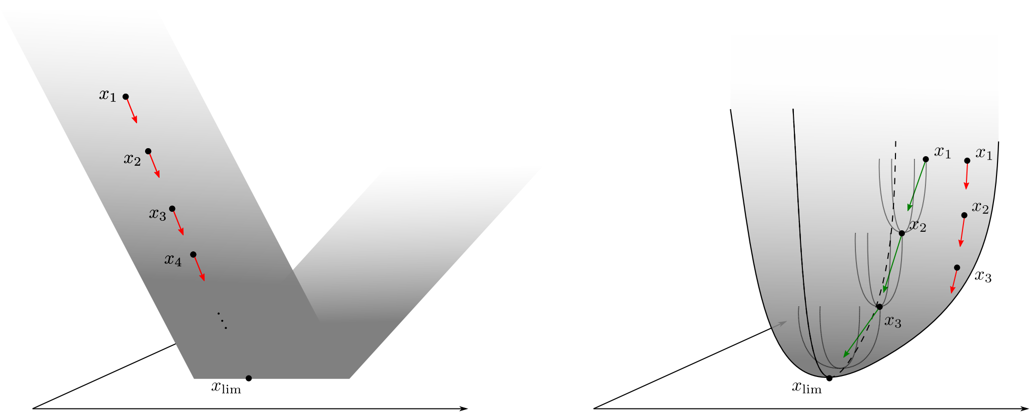


Figure 1.13: (left) Gradient descent is fast on hyperplane-shaped functions f; (right) Newton's method improves convergence speed while not following the *steepest descent.*

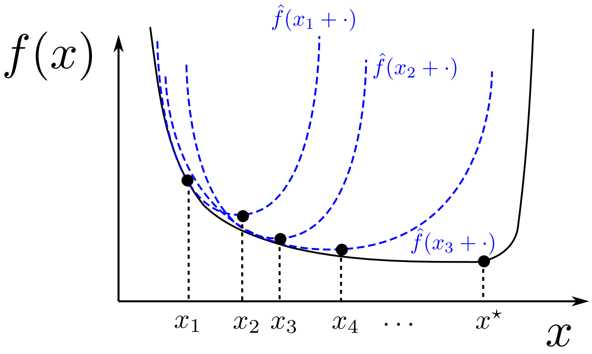


Figure 1.14: Newton's Method

Definition 25 (Newton's Method)**.** *Assume f twice differentiable and $\nabla^2 f(x) \succ 0$ for all $x \in \mathcal{X}$. Then Newton's method is an iteration method consisting in letting*

$$\begin{cases} \Delta x_k & = -\left[\nabla^2 f(x_k)\right] \nabla f(x_k). \\ t_k & = 1 \end{cases}$$

Property 13 (Newton's Method is a Descent Method)**.** *Since $\nabla^2 f(x) \succ 0$, $-\nabla f(x)^{\mathsf{T}} \left[\nabla^2 f(x_k)\right] \nabla f(x_k) \leq 0$ with equality only if $\nabla f(x_k) = 0$, so that Newton's method is a valid descent method.*

Remark 14. *A few remarks are in order.*

- *Note first that Newton's method has the advantage to be* linearly invariant *in the sense that, if $x = Ay$ and $g(y) = f(x) = f(Ay)$, then $y_{k+1} = Ax_{k+1}$ if $\{x_k\}$ is a series from Newton's method on f and $\{y_k\}$ a series from Newton's method on g. This is not the case of the gradient descent algorithm.*

- *If $\nabla^2 f$ is locally close to singular, Newton's method can be very slow and even diverge.*

- *Also, for large ambient space dimension n, Newton's method can be extremely costly as it requires the inversion of $\nabla^2 f(x_k)$ for every k (an operation of cost order $O(n^3)$).*

Exercise 9 (Affine invariance)**.** *Prove that Newton's method is affine invariant, i.e., if $x = Ay + b$ and $g(y) = f(x)$, then Newton's methods on f or g are mutually consistent. Show that this is not true in general of gradient descent.*

To avoid the aforementioned issues of divergence under singular $\nabla^2 f$, Newton's method is usually implemented with a step-size adaption as follows.

Definition 26 (Damped Newton's Method)**.** *The damped Newton's method produces the sequence*

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

with t_k obtained by backtracking line search.

Theorem 7 (Convergence of damped Newton's method)**.** *Assume $\nabla^2 f$ is M-Lipschitz, i.e.,*

$$\forall x, y,\ \left\|\nabla^2 f(y) - \nabla^2 f(x)\right\| \leq M\|y-x\|$$

and that $lI_n \preceq \nabla^2 f(x) \preceq LI_n$. Then the damped Newton's method converges sublinearly then quadratically as soon as $\|\nabla f(x_k)\| < \eta$ for some sufficiently small $\eta > 0$; besides, from this point on, $t_k = 1$.

Proof. We only show the second part of the proof and take $t_k = 1$. That is, we want to prove that Newton's method converges quadratically as soon as $\|\nabla f(x_k)\| < \eta$ for some $\eta > 0$ small.

To this end, write

$$\begin{aligned}\|\nabla f(x_{k+1})\| &= \|\nabla f(x_k + \Delta x_k) \underbrace{-\nabla f(x_k) - \nabla^2 f(x_k)\Delta x_k}_{=0}\| \\ &= \left\|\int_0^1 (\nabla^2 f(x_k + u\Delta x_k) - \nabla^2 f(x_k))\Delta x_k du\right\| \\ &\leq \frac{M}{2}\|\Delta x_k\|^2 \\ &= \frac{M}{2}\|[\nabla^2 f(x_k)]^{-1}\nabla f(x_k)\|^2 \\ &\leq \frac{M}{2l^2}\|\nabla f(x_k)\|^2.\end{aligned}$$

Now multiplying both sides by $M/(2l^2)$, leads to

$$\frac{M}{2l^2}\|\nabla f(x_K)\| \leq \left(\frac{M}{2l^2}\|\nabla f(x_{k_0})\|\right)^2$$

Iterated over $k = k_0, \ldots, K$, this eventually gives

$$\|\nabla f(x_K)\| \leq \alpha C^{2^{K-k_0}}$$

with $C = \frac{M}{2l^2}\|\nabla f(x_{k_0})\| < 1$ if $\|\nabla f(x_{k_0})\| < \eta = \frac{2l^2}{M}$. This concludes the proof. □

Remark 15 (Variational perspective of damped Newton's method)**.** *Let*

$$h_{x_k}(y) = f(x_k) + \nabla f(x_k)^\mathsf{T}(y - x_k) + \frac{1}{2t}(y - x_k)^\mathsf{T}\nabla^2 f(x_k)(y - x_k) \approx f(y)$$

then we have

$$x_{k+1} = x_k - t\left[\nabla^2 f(x_k)\right]^{-1}\nabla f(x_k) = \operatorname{argmin}_y h_{x_k}(y)\ .$$

1.3.3 Inequality Constraints and Barrier Methods

So far in the course, we have considered optimization over the unbounded space $\mathcal{X} = \mathbb{R}^n$. In practice though, we may demand that valid $x \in \mathcal{X}$ vectors be constrained. For instance, that $[x]_i > 0$ for all $i = 1, \ldots, n$. Since the convex optimization methods presented so far disregard these constraints, the presented algorithms may reach values in violation with those.

We will start by presenting cases of inequality constraints, for the simple reason that they can be related to unconstrained optimization by a relaxation trick.

Example 6 (Linear Programming Problems)**.** *A classical example of inequality constrained optimization problems that is simple yet cannot be solved explicitly is called* linear programming, *that consists in the optimization*

$$\min_{x\in\mathbb{R}^n} c^\mathsf{T} x \text{ such that } Ax \leq b$$

where $Ax \leq b$ is understood entry-wise. This can be equivalently rewritten under the form

$$\min_{x\in\mathbb{R}^n,\ Ax\leq b} c^\mathsf{T} x$$

or, as we shall see subsequently in the course (on proximal approaches), under the alternative form

$$\min_{x\in\mathbb{R}^n} c^\mathsf{T} x + \imath_{\{Ax\leq b\}}(x)$$

with $\imath_S(x)$ the function equal to 0 *if $x \in S$ and $+\infty$ otherwise.*

It is not difficult to see that the solutions to a linear programming problem are found on one of the constraint corner points. The so-called "simplex algorithm" which consists in iterating from corner point to corner point in a descending manner solves the problem. However, it comes at a cost which may increase fast with the number of constraints and is often more suitably replaced by a regularization method.

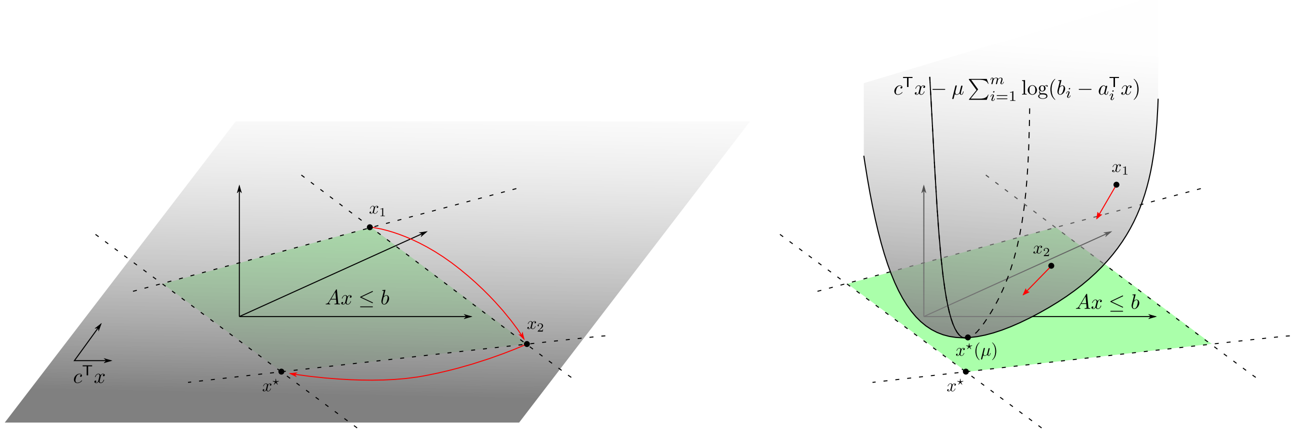


Figure 1.15: Linear Programming Problem. (left) The simplex method, iterating from edge to edge; (right) The relaxation logarithm barrier method that finds an approximation $x^\star(\mu)$ of $x^\star$.

For notational simplicity in the following, we will consider the following optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x) \text{ such that } c_i(x) \geq 0,\ i = 1, \ldots, m$$

where $c_i(x) = a_i^\top x - b_i$ for some $a_i, b_i \in \mathbb{R}^n$.

To avoid the problem of descent methods stuck in the boundaries of the constraint set (see left of Figure 1.16), the *interior point* (also called *barrier*) method consists in relaxing the function $f(x)$ via an additional cost resulting in infinity values outside the constraint set as follows.

Definition 27 (Barrier Method)**.** *The barrier method consists in the following iterative algorithm. Assuming f continuously differentiable, define for all $\mu > 0$, the function*

$$\phi(x; \mu) \equiv f(x) - \mu \sum_{i=1}^m \log(c_i(x)).$$

Then, starting with $x_0(\mu) \in \mathscr{X}$ such that $c_i(x_0(\mu)) > 0$ for all $i = 1, \ldots, m$, proceed to a descent algorithm to solve

$$\min_{x \in \mathbb{R}^n} \phi(x; \mu)$$

the solution of which is denoted $x^\star(\mu)$. Then decrease μ and, starting from the previous $x^\star(\mu)$, repeat (one must notably be careful to adapt the steps in such a way that $c_i(x_k(\mu)) > 0$ for all k).

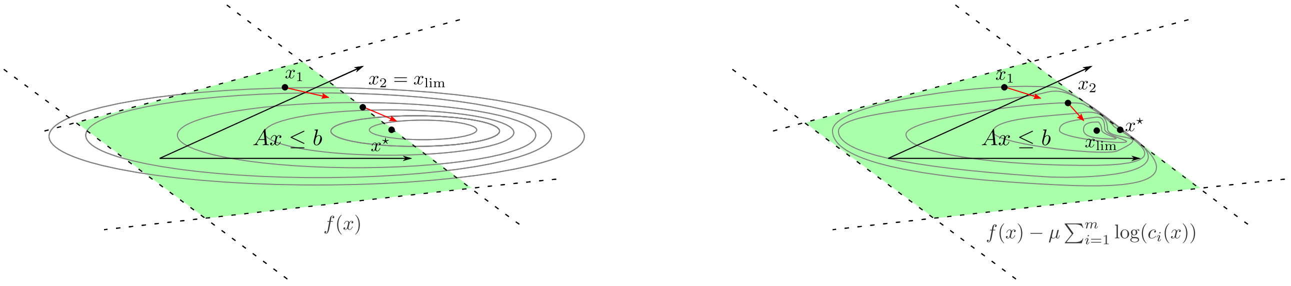


Figure 1.16: The Barrier (or interior point) Method. (left) Level sets of f and constraint set: here the algorithm is "stuck" in an invalid descent direction; (right) Level sets of $f - \mu \sum_{i=1}^m \log(c_i(x))$ and constraint set: here the algorithm finds an approximation for $x^\star$.

Exercise 10 (Numerical Exercise)**.** *Solve the problem*

$$\min_{x \in \mathbb{R}^2} ([x]_1)^2 + ([x]_2)^2 \text{ such that } [x]_1 + ([x]_2)^2 \geq 1$$

by iterating with $\mu = 10$, $\mu = 1$, $\mu = 0.1$. At each step, depict the level sets of $\phi(x; \mu)$ and $x^\star(\mu)$.

Remark 16 (On the Barrier Method)**.** *While avoiding the problem of stark boundaries where descent algorithms might get stuck, the barrier method is far from ideal for several reasons:*

- *the valid set of x is now a restriction (possibly bounded) of $\mathscr{X}$; as such, while all descent directions are valid, not all step sizes are and backtracking is in general necessary to avoid exiting the valid set;*
- *the algorithm involves a double-iteration with refinements on μ; in practice, this is often difficult to handle:*
  - *recall that the initialization point in a subsequent μ-step is the solution of the previous step; the latter must be sufficiently close to the next μ-step solution to avoid slow descents. Conversely, too small μ-updates slows the overall convergence.*
  - *when solutions are found near or at a constraint, the aforementioned limitation is exacerbated and one must a priori know where the solutions are expected to be found and how "stark" is the constraint likely to be to properly handle the convergence speed.*
  - *on a similar aspect, for stark barriers, step sizes must be very thinly adapted to avoid "jumping" over the minimum.*
- *the barrier method is only valid when the constraints are* inequality-based*; equality constraints require other techniques.*

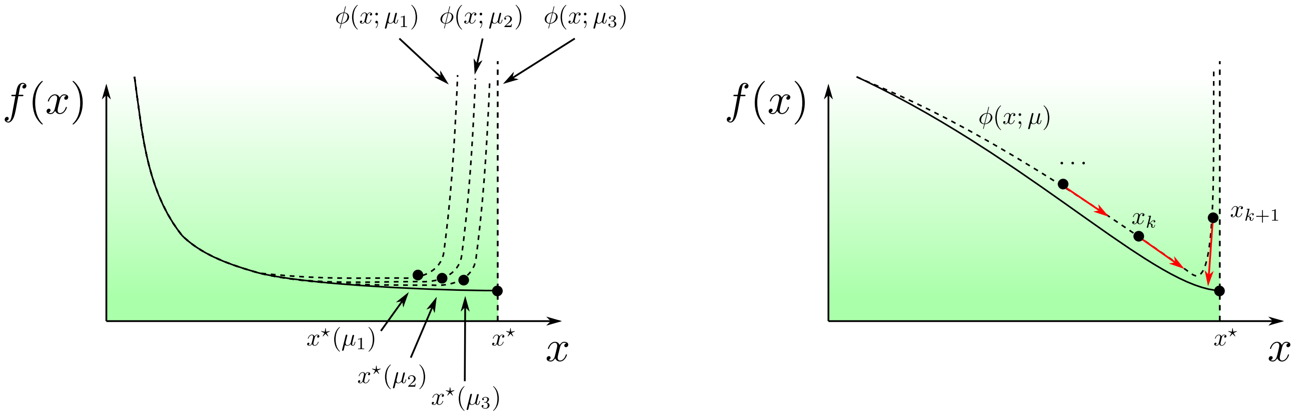


Figure 1.17: The Barrier (or interior point) Method. (left) Sequence of $\phi(x;\mu)$ approximations; (right) The difficulty raised by sharp minima and the possible "ping-pong" effect of iterates with not-sufficiently-small step sizes.

Due to these difficulties, the barrier method is in reality not often used. It is nonetheless a quick and simple regularization approach, flexible to non trivial functions f.

In the remainder of the course, we will introduce more advanced methods, based notably on proximal point approximations which are much more efficient and avoid most of the aforementioned problems, yet with additional, somewhat disadvantageous, difficulties. To this aim though, we first need to introduce the notion of duality in convex (and non-convex) optimization which primarily solves the question of equality-constrained optimization.

1.4 Constrained Optimization and Duality

1.4.1 Linearly Equality-Constrained Optimization

To introduce the notion of duality, it is convenient to start with the simple problem of equality constrained optimization.

Consider then the following optimization problem:

$$\min_{x\in\mathcal{X}} f(x) \text{ such that } h_i(x) = 0,\ i = 1,\dots,p. \tag{1.5}$$

Theorem 8. *If $x^\star$ is a solution to* (1.5)*, then there exists $\lambda_1,\dots,\lambda_p \in \mathbb{R}$ such that*

$$\nabla f(x^\star) = \sum_{i=1}^{p}(-\lambda_i)\nabla h_i(x^\star).$$

Geometric Proof. It is first important to note that the gradient of a differentiable function g is orthogonal to its level sets. Indeed, a level set is characterized by $\ell_c(g) = \{x\,|\,g(x) = c\}$ for some constant $c \in \mathbb{R}$. Now, for $h \in \mathcal{X}$ such that $g(x) = g(x+h) = c$ and $\|h\| \to 0$ along the level set, we have $0 = (g(x+h)-g(x))/\|h\| = \nabla g(x)^\mathsf{T}(h/\|h\|) + o(1)$ and thus $\nabla g(x)$ is orthogonal to the level set.

From Figure 1.18, we now see that every local minimum of f satisfying $h(x) = 0$ is such that the gradients of f and h are aligned. Indeed, if not, then one could keep descending to smaller level curves of f along some direction of $h(x) = 0$. This is in particular the case for $x^\star$ the solution to (1.5). Thus there must exist λ such that

$$\nabla f(x^\star) = \lambda\nabla h(x^\star).$$

Another case may nonetheless occur if $h(x) = 0$ passes by a local minimum of f, but then the formula above still holds with $\lambda = 0$.

The generalization to p constraints follows the same arguments.

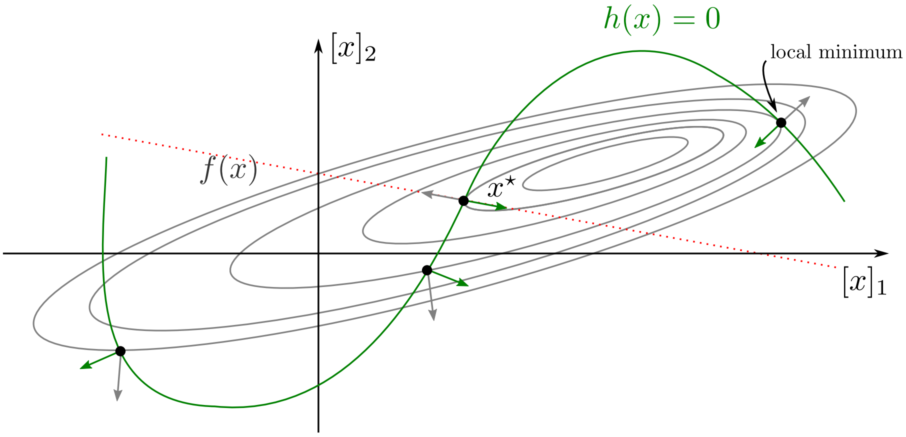


Figure 1.18: Geometric visualization of the level sets of f and the constraint $h(x) = 0$. Gradients of both f and h are orthogonal to the level sets.

□

Theorem 8 provides a necessary condition for finding an extremum for f under the linear constraints h_i, since the optimization problem then resorts to finding x such that $f(x) + \sum_i \lambda_i h_i(x)$ has a local zero derivative for some $\lambda_1,\dots,\lambda_p$.

This naturally leads one to introduce the so-called *Lagrange dual function* as follows.

Definition 28 (Lagrange dual function)**.** *For $\lambda \in \mathbb{R}^p$, the Lagrange dual function g of f is defined as*

$$g(\lambda) = \inf_{x\in\mathcal{X}} L(x;\lambda)$$

where

$$L(x;\lambda) \equiv f(x) + \sum_{i=1}^{p}\lambda_i h_i(x).$$

The coefficients $\lambda_1,\dots,\lambda_p$ are called the Lagrange multipliers.

Property 14 (Lagrange dual as a lower bound)**.** *For $x^\star$ a solution of the constrained optimization problem, since $h_i(x^\star) = 0$, it is immediate that, for all $\lambda \in \mathbb{R}^p$,*

$$g(\lambda) = \inf_{x\in\mathscr{X}} L(x;\lambda) \leq L(x^\star;\lambda) = f(x^\star)$$

and thus in particular

$$\sup_{\lambda\in\mathbb{R}^p} g(\lambda) \leq f(x^\star).$$

From this property, if $\sup_{\lambda\in\mathbb{R}^p} g(\lambda) = f(x^\star)$, then one has found a detoured manner to solve the constrained optimization problem by means of *two nested unconstrained optimization problems.*

This thus naturally brings us to the so-called dual Lagrangian problem definition.

Definition 29 (Lagrange dual problem)**.** *The Lagrange dual problem consists in solving*

$$\sup_{\lambda\in\mathbb{R}^n} g(\lambda) = \sup_{\lambda\in\mathbb{R}^p} \left\{ \inf_{x\in\mathscr{X}} L(x;\lambda) \right\}.$$

We shall generically denote $\lambda^\star \in \mathbb{R}^n$ any point of the set $\operatorname{argmax}_\lambda g(\lambda)$ *(which may be empty).*

For $\lambda^\star$ as above, we call $g(\lambda^\star) - f(x^\star) \geq 0$ the *duality gap* of the constrained optimization problem. In particular, if the duality gap is zero, then we can solve the original (so-called *primal*) problem through the Lagrange dual. This leads us to an important property of the Lagrangian dual problem.

Property 15. *The Lagrange dual function $\lambda \mapsto g(\lambda)$ is concave,* irrespective of the convexity of f.

Proof. Take $\lambda_1, \lambda_2 \in \mathbb{R}^p$, $\alpha \in [0,1]$ and observe that

$$\begin{aligned} g(\alpha\lambda_1 + (1-\alpha)\lambda_2) &= \inf_{x\in\mathscr{X}} \left\{ \alpha\left(f(x) + \sum_{i=1}^p \lambda_{1i} h_i(x)\right) + (1-\alpha)\left(f(x) + \sum_{i=1}^p \lambda_{2i} h_i(x)\right)\right\} \\ &\geq \alpha \inf_{x\in\mathscr{X}} \left\{ f(x) + \sum_{i=1}^p \lambda_{1i} h_i(x)\right\} + (1-\alpha) \inf_{x\in\mathscr{X}} \left\{ f(x) + \sum_{i=1}^p \lambda_{2i} h_i(x)\right\} \\ &= \alpha g(\lambda_1) + (1-\alpha) g(\lambda_2) \end{aligned}$$

where the inequality follows from the fact that

$$\inf_x \{f_1(x) + f_2(x)\} \geq \inf_x \{f_1(x)\} + \inf_x \{f_2(x)\}.$$

□

Since the dual problem $\sup_\lambda g(\lambda)$ is concave, this means that $\inf_\lambda -g(\lambda)$ is convex. Since both problems are one and the same, this means that the Lagrangian dual optimization can be solved by standard *unconstrained* convex optimization algorithms.

It is even more important to note that, if f is not convex (and thus $\min f$ cannot be solved by standard optimization), at least $\max g$ can be solved, leading up to a solution lower bounding $\min f$.

A fundamental property of the Lagrangian dual is that strong duality (and thus the possibility to solve the primal problem by solving the dual) holds for convex f and linear h_i. This sufficient condition is often referred to as Slater's condition.

Theorem 9 (Slater's condition)**.** *If there exists $x \in \mathscr{X}$ such that $h_i(x) = 0$ for $i = 1,\ldots,p$ (i.e., the feasibility set of x is non empty). If, furthermore, f is convex and h_i affine, i.e., $h_i(x) = a_i^\top x + b_i$ for some $a_i, b_i \in \mathbb{R}^p \times \mathbb{R}$, then the equality-constrained convex optimization problem is feasible and strong duality holds.*

Proof. Let $\bar\lambda \in \mathbb{R}^p$ be such that $\nabla f(x^\star) = \sum_{i=1}^p (-\bar\lambda_i)\nabla h_i(x^\star)$. Then

$$g(\bar\lambda) = \inf_{x\in\mathscr{X}} f(x) + \sum_{i=1}^p \bar\lambda_i h_i(x) = f(x^\star).$$

Indeed, $x \mapsto f(x) + \sum_{i=1}^p \bar\lambda_i h_i(x)$ is convex (since h_i is affine) and thus minimum when its gradient is zero which must hold for any point x having the same cost as $x^\star$. This cost equals $f(x^\star) + \sum_{i=1}^p \bar\lambda_i h_i(x^\star) = f(x^\star)$. As a consequence,

$$\begin{aligned} g(\lambda^\star) &= \max_{\lambda\in\mathbb{R}^p} g(\lambda) \geq g(\bar\lambda) = f(x^\star) \\ g(\lambda^\star) &\leq f(x^\star) \end{aligned}$$

and thus $g(\lambda^\star) = f(x^\star)$ and the duality gap is zero. □

1.4.2 Generalization to Equality and Inequality Constraints

The discussion above can be generalized to cases where both equality and inequality constraints are present. That is, we now consider the more general optimization problem:

$$\min_{x\in\mathscr{X}} f(x) \text{ such that } g_i(x)\leq 0,\ i=1,\ldots,m \text{ and } h_j(x)=0,\ j=1,\ldots,p. \tag{1.6}$$

For inequality constraints, additional Lagrange multipliers are added to the Lagrange dual function. The main difference, though, is that the associated multipliers are imposed to be *positive*. Essentially, two situations occur:

- if, at the minimum, the constraint is enforced (i.e., the optimum is found at an edge of the constraint set), then the inequality becomes an equality and the Lagrangian multiplier is therefore non zero. But then, the multiplier is necessarily *positive*; indeed, on the constraint subset $\{x,\ g(x)=0\}$, the gradient of g points away from the inequality constraint set $\{x,\ g(x)\leq 0\}$. Thus, as the gradient of f also points toward the increasing level sets, both gradients are opposed and the associated Lagrange multiplier is positive (remember that it is defined with a minus sign).
- if, instead, the constraint is not enforced (i.e., the optimum is found within the constraint set), then the associated Lagrange multiplier is zero.

See Figure 1.19 for a visual interpretation.

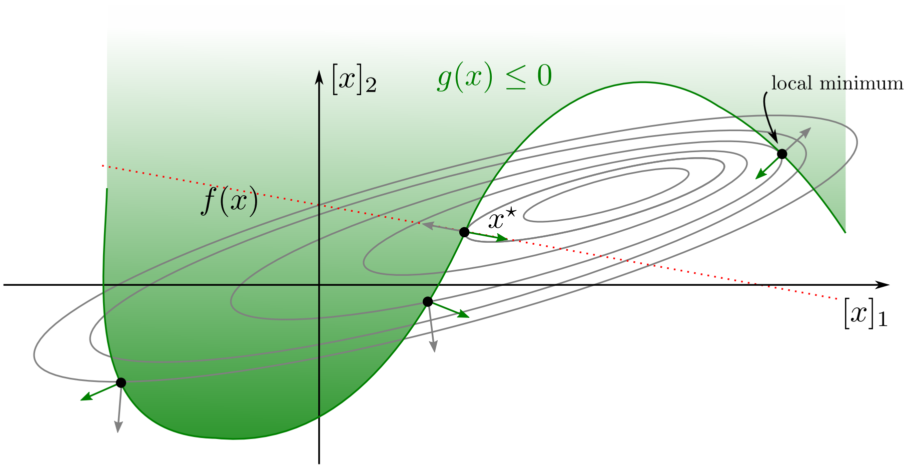


Figure 1.19: Optimization with Inequality Constraints

As in the case of equality constrained optimization, we define the Lagrange dual problem as follows.

Definition 30 (Lagrange Dual Problem)**.** *The Lagrange dual of the optimization problem* (1.6) *is given by*

$$\max_{\lambda\in\mathbb{R}^p,\ \nu\in\mathbb{R}^m_+} g(\lambda,\nu)$$

where

$$g(\lambda,\nu)\equiv\inf_{x\in\mathscr{X}} L(x;\lambda,\nu)$$

and $L(x;\lambda,\nu)$ *is the Lagrange function*

$$L(x;\lambda,\nu)\equiv f(x)+\sum_{i=1}^m \nu_i g_i(x)+\sum_{j=1}^p \lambda_j h_j(x).$$

In this case, we have the following updated version of Slater's condition.

Theorem 10 (Slater's Condition)**.** *Let f be convex, g_i be convex and h_j be affine. Then, if there exists $x\in\mathscr{X}$ such that $h_i(x)=0$ and $g_j(x)\leq 0$ for all i,j, the problem* (1.6) *is said to be feasible and strong duality holds.*

Remark that if g_j is a convex function, then $\mathscr{G}_j=\left\{x\middle|g_j(x)\leq 0\right\}$ defines a convex set (can you see why?). For an affine function h_i, we have that $\mathscr{H}_i=\left\{x\middle|h_i(x)=0\right\}$ also defines a convex set (why is that not so for the more general case where h_i is convex?). Hence, we are looking for a solution $x^\star=\operatorname{argmin}_{\mathscr{X}\cap(\bigcap_j\mathscr{G}_j)\cap(\bigcap_i\mathscr{H}_i)} f(x)$, i.e., minimising a function f over the convex set (as an intersection of convex sets) $\mathscr{X}\cap\left(\bigcap_j\mathscr{G}_j\right)\cap\left(\bigcap_i\mathscr{H}_i\right)$.

Related to the notion of duality is the Fenchel convex conjugate introduced in previous sections and which may be practical to emphasize the duality between two optimization problems.

We recall the definition of the Fenchel convex conjugate:

Definition 31 (Fenchel Convex Conjugate)**.** *For $f : \mathscr{X} \to \mathbb{R}$, we define the convex conjugate of f as the function $f^* : \mathscr{X} \to \mathbb{R}$*

$$\begin{aligned} f^*(y) &= \sup_{x\in\mathscr{X}} \left\{ y^\mathsf{T} x - f(x) \right\} \\ &= -\inf_{x\in\mathscr{X}} \left\{ f(x) - y^\mathsf{T} x \right\}. \end{aligned}$$

In particular, for the linearly-constrained optimization problem

$$\min_{x\in\mathscr{X}} f(x) \text{ such that } Ax \leq b \text{ and } Cx = d$$

(with (in)equalities understood entry-wise) we have a Lagrangian dual

$$\begin{aligned} &\sup_{v\geq 0,\lambda} \left\{ \inf_{x\in\mathscr{X}} f(x) + v^\mathsf{T}(Ax-b) + \lambda^\mathsf{T}(Cx-d) \right\} \\ &= \sup_{v\geq 0,\lambda} \left\{ -v^\mathsf{T} b - \lambda^\mathsf{T} d + \inf_{x\in\mathscr{X}} f(x) + [v^\mathsf{T} A + \lambda^\mathsf{T} C]x \right\} \\ &= \sup_{v\geq 0,\lambda} \left\{ -v^\mathsf{T} b - \lambda^\mathsf{T} d - f^*(-A^\mathsf{T} v - C^\mathsf{T}\lambda) \right\} \\ &= -\inf_{P_v q\geq 0,(I-P_v)q=0} \left\{ u^\mathsf{T} q + f^*\left(-Q^\mathsf{T} q\right) \right\}. \end{aligned}$$

where $u^\mathsf{T} = (b^\mathsf{T}, d^\mathsf{T})$; $Q^\mathsf{T} = \left(A^\mathsf{T}, C^\mathsf{T}\right)$; and $q^\mathsf{T} = (v^\mathsf{T}, \lambda^\mathsf{T})$. P_v is the projection of q onto the coordinates corresponding to v. Thus, if f^* assumes an explicit expression, solving the primal problem is equivalent to solving a (possibly simpler) inequality-constrained convex optimization problem. Indeed, since f^* is convex, no matter the convexity of f. This affirms that the Lagrangian dual under inequality constraints is a convex function in q, hence jointly in v and λ.

Exercise 11 (Convexity of the composition of functions)**.** *Proof the following two statements:*

1. *For f a convex function and g an affine function, $f+g$ is a convex function;*
2. *For f convex and g affine, the function $f \circ g$ is convex.*

1.5 Advanced Methods

1.5.1 Non-Differentiable Convex Functions

In this section, we consider a broader class of functions, that of convex, not everywhere differentiable functions. We will often use *convex non-differentiable functions* for short.

Examples of non differentiable functions in one or two dimensions are given in Figure 1.20.

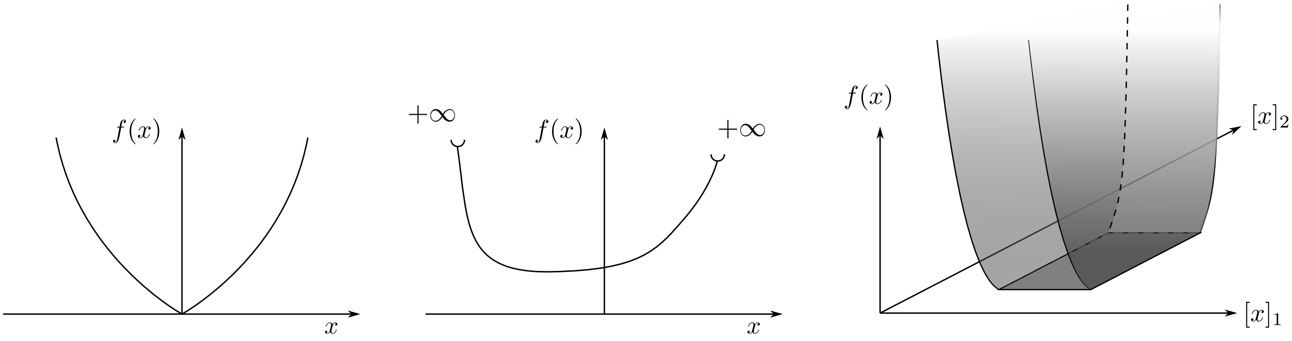


Figure 1.20: Examples of not-everywhere differentiable convex functions

Recall theorem 1 of first order conditions for convex functions : *Suppose $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ is differentiable in its domain. Then f is convex if and only if, for all $x, z \in \mathrm{dom}(f)$,*

$$f(z) \geq f(x) + \nabla f(x)^\top (z - x).$$

This theorem can also be used as an implicit definition of the gradient ∇f for convex functions, since it is the only linear application that satisfies the inequality of the theorem.

The gradient of a non-differentiable function f is not defined in the non-smooth regions. Yet, another notion of differentiation still exists, which is referred to as the *subdifferential*, and is defined as follows:

Definition 32 (Subdifferential)**.** *Let $f : \mathcal{X} \to \mathbb{R}$. We call* subdifferential *of f, denoted ∂f, the function*

$$\begin{aligned} \partial f : \mathcal{X} &\to 2^{\mathcal{X}} \\ x &\mapsto \left\{ u \in \mathcal{X} \mid \forall z \in \mathcal{X},\ f(x) \leq f(z) + u^\top (x - z) \right\}. \end{aligned}$$

As such, the subdifferential $\partial f(x)$ is a *set-valued* function where the members of the set are called *subderivatives.*

Property 16 (Subdifferential as a convex set)**.** *The subdifferential is always a nonempty convex compact set.*

Note that the subdifferential can also be obtained by looking at the set of supporting hyperplanes, and hence the Fenchel conjugate of f. Recall the support function of a convex set, here epi(f):

$$h_{\mathrm{epi}(f)}((u, -1)) = \sup_{p \in \mathrm{epi}(f)} p^\top (u, -1) = \sup_{x \in \mathcal{X}} x^\top u - f(x) \overset{\mathrm{def}}{=} f^*(u)$$

where f^* is the Fenchel conjugate of f, and $p = (x, f(x))$ because the supremum of a linear application over a convex set is always attained on its boundary, the boundary of epi(f) being the points $\big(x, f(x)\big)$.

Now, let x be such that $f^*(u) = \max_{z \in \mathcal{X}} u^\top z - f(z) = u^\top x - f(x)$, i.e., $x \in \arg\min_{z \in \mathcal{X}} f(z) - u^\top z$ and consider

$$\mathcal{S}(x) = \left\{ u \in \mathcal{X} \mid x \in \underset{z \in \mathcal{X}}{\arg\min} f(z) - u^\top z \right\} = \left\{ u \in \mathcal{X} \mid \forall z \in \mathcal{X}, f(x) \leq f(z) + u^\top (x - z) \right\} ,$$

but this is exactly the definition of the subdifferential $\partial f(x)$.

Property 17 (The subdifferential is the set of supporting hyperplanes)**.** *The subdifferential $\partial f(x)$ consists of the supporting hyperplanes (defined through their normal vector u) that support* epi(f) *at $\big(x, f(x)\big)$. See Figure 1.21 (top).*

Property 18. *For convex f, the subdifferential $\partial f(x)$ at those x where f is differentiable is a singleton*

$$\partial f(x) = \left\{ \nabla f(x) \right\}.$$

Proof. Let $u \in \partial f(x)$, hence $x \in \arg\min_{z \in \mathcal{X}} f(z) - u^\top z$. Since f is differentiable at x, the first order condition of this minimisation problem yields $\nabla f(x) = u$. □

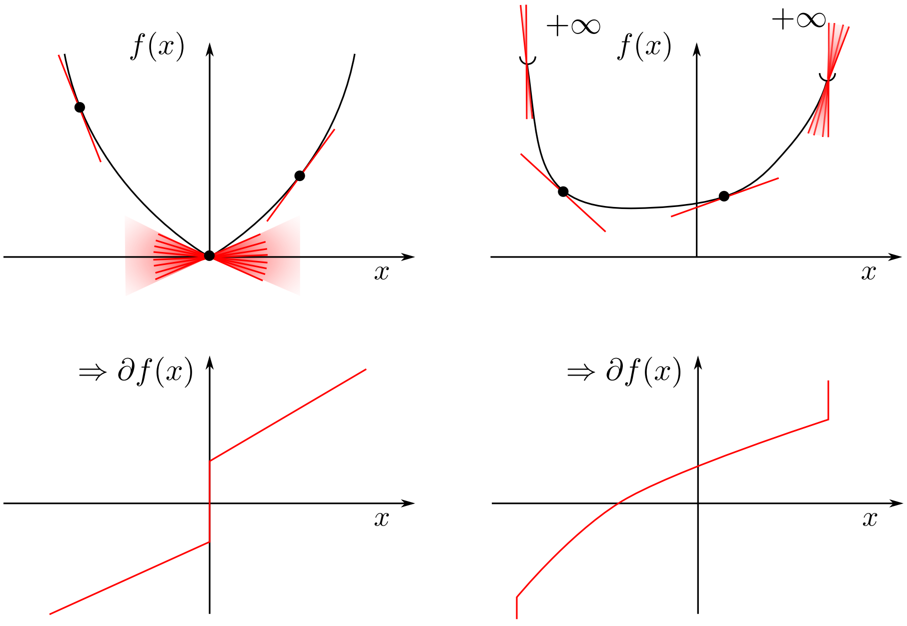


Figure 1.21: Subdifferential of non differentiable functions.

Exercise 12. *Determine the subdifferentials of the following functions* $f : \mathbb{R} \to \mathbb{R}$

1. $f(x) = \text{abs}(x)$
2. $f(x) = \frac{1}{2}x^2 1_{x \le 0} + x 1_{x>0}$.

The subdifferential enjoys fundamental properties that extend the notion of differentials. In particular, Fermat's rule for the optimization of differentiable convex functions extends as follows.

Theorem 11 (Fermat's Rule)**.** *Let* $f : \mathscr{X} \to \mathbb{R}$ *be convex. Then*

$$x^\star \in \text{argmin}_{x\in\mathscr{X}} f(x) \Leftrightarrow 0 \in \partial f(x^\star).$$

Proof. ($\Rightarrow$) It follows that the set of subdifferentials at $x^\star$ must at least contain 0, since $x^\star \in \text{argmin}_{x\in\mathscr{X}} f(x) + 0^\mathsf{T} x$; ($\Leftarrow$) $0 \in \partial f(x) \implies x \in \text{arg}\min_{z\in\mathscr{X}} f(z)$, but then x must be in the set of optimal solutions. □

As such, looking for a minimum of f is equivalent to finding a 0 in one of the sets $\partial f(x)$, $x \in \mathscr{X}$. Note that this is different from looking for the singleton $\{0\}$ among the sets $\partial f(x)$, $x \in \mathscr{X}$ (which would correspond to looking for minima where f is differentiable).

From the aforementioned important observations, a natural extension of the gradient descent algorithm is the so-called *subgradient method* that consists in the following algorithm.

Definition 33 (Subgradient algorithm)**.** *Under the conditions of Theorem 5, where the Lipschitz character is extended to every value of the subgradients of* f*, the subgradient algorithm consists in the following steps:*

1. $x_{k+1} = x_k - t_k u_k$*, for any* $u_k \in \partial f(x_k)$
2. $f_{\text{best}}^{k+1} = \min\{f_{\text{best}}^k, f(x_{k+1})\}$.

The second step in the algorithm is crucial and underlies a major weakness of the method (which is rarely used in practice): the subgradient algorithm is *not* a descent method.

Exercise 13 (Example of "bad" subgradient)**.** *Consider the function*

$$\begin{aligned} f : \mathbb{R}^2 &\to \mathbb{R} \\ x &\mapsto |[x]_1| + 3|[x]_2|. \end{aligned}$$

Evaluate the subgradient at $x = (0,0)$*. Show that* $u_1 = (1,0)$ *and* $u_2 = (-1,3)$ *belong to* $\partial f(x)$*. Which of* $-u_1$ *and* $-u_2$ *are descent directions?*

Nonetheless, it can be shown that the subgradient algorithm does converge in constant step sizes, provided that Lipschitz conditions on the subgradient are in place. However, convergence can be very slow and the algorithm is therefore generally not used in practice.

This is what leads us to the more powerful *proximal point* methods [1]. Before moving to proximal methods though, we set as two exercises (1) an interpretation of the proximal operator and (2) a natural transition between subdifferentials and proximal operators.

Exercise 14 (The Projection Operator)**.** *Let Ω be a convex set and the indicator function ι_Ω defined as $\iota_\Omega(x) = 0$ if $x \in \Omega$ and $\iota_\Omega(x) = +\infty$ if $x \notin \Omega$. Now define*

$$\min_{x \in \mathcal{X}} \frac{1}{2}\|x - y\|^2 + \iota_\Omega(x)$$

Show that the optimum solution $x^\star$ is simply the (Euclidean) projection of y onto $\Omega \cap \mathcal{X}$. If we suppose that $\Omega \subset \mathcal{X}$, this is simply the projection onto Ω.

Proximal has its roots in "proximity". Indeed, a proximal point of y tends to stay close to x—through the $\|\cdot - y\|^2$ term,—and simultaneously (approximately) minimises the objective function, here ι_Ω.

In many applications of compressive sensing (inverse problems with sparsity constraints), an important function is the so-called *soft-thresholding operator* $\mathrm{soft}(\cdot)$. This operator is related to the following optimization problem

$$\mathrm{soft}(y) = \arg\min_{x \in \mathcal{X}} \frac{1}{2}\|x - y\|^2 + \lambda\|x\|_1$$

where $\|x\|_1 = \sum_{i=1}^n |[x]_i|$ and $\lambda > 0$ is arbitrary.

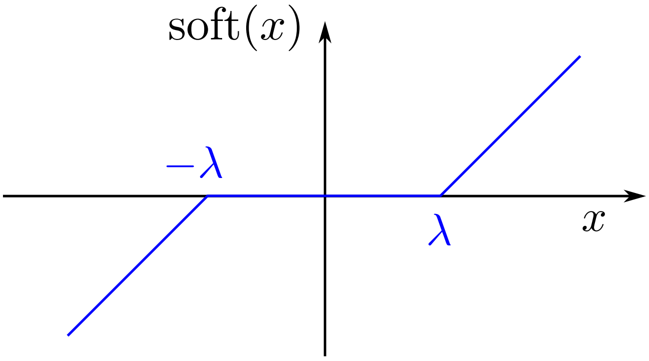
Exercise 15 (The Soft-Thresholding Operator)**.** *Show that*

$$x^\star = \mathrm{soft}(y)$$

where

$$\mathrm{soft}(y) = \begin{cases} [y]_i + \lambda, & [y]_i < -\lambda \\ [y]_i - \lambda, & [y]_i > \lambda \\ 0, & |[y]_i| \le \lambda. \end{cases}$$

The function $\mathrm{soft}(\cdot)$ *happens to be the* proximal operator *for the function $x \mapsto \lambda\|x\|_1$.*



1.5.2 The Proximal Operator Approach

The proximal operator method consists in a relaxation approach which works around the differentiability problems. To understand this advanced notion, we start with the definition of the proximal operator which—to some extent—can be related to Fenchel's convex conjugate and to the notion of duality.

Definition 34 (The Proximal (Point) Operator)**.** *Let $f : \mathcal{X} \to \mathbb{R}$ be convex. Then the proximal operator* prox_f *of f is defined by*

$$\begin{aligned} \mathrm{prox}_f : \mathcal{X} &\to \mathcal{X} \\ x &\mapsto \operatorname*{argmin}_{y \in \mathcal{X}} \left\{ f(y) + \frac{1}{2}\|y - x\|^2 \right\}. \end{aligned}$$

As the definition suggests, the proximal point operator is *single-valued.* This is an important, though far from obvious, property (recall that f itself may have multiple minimizers!) that needs to be properly shown. But first we need to introduce another definition

Definition 35 ((Strictly) Monotone operator)**.** *An operator $D : \mathscr{X} \to 2^{\mathscr{X}}$ is a (strictly) monotone operator if*

$$\forall x, y \in \mathscr{X}, D : d_x \in D(x), d_y \in D(y) \implies \left(d_y - d_x\right)^\top \left(y - x\right) \geq 0$$

where for strictly monotone operators the equality is only obtained for $x = y$.

Property 19 (inverse of a strictly monotone operator)**.** *The inverse of the strictly monotone operator is single-valued.*

Uniqueness of the inverse of a strictly monotone operator. The proof goes by contradiction. Let $x \in \mathscr{X}$ such that $\delta_x \in D(x)$ and suppose $\exists x'$ such that $\delta_x \in D(x')$ which means that δ_x has at least two preimages. But, by strict monotonicity, $0 < (\delta_x - \delta_x)^\top (x - x') = 0$, hence,—by contradiction—the inverse of D must be single-valued. □

Property 20. *The operator* prox_f *as introduced in Definition 34 is single-valued and thus well-defined.*

Proof. The property relies on the fact that ∂f is a *monotone operator* in the sense that, for all $d_x \in \partial f(x)$ and $d_y \in \partial f(y)$,

$$(d_y - d_x)^\top (y - x) \geq 0 \ .$$

This follows from summing up the two first order relations $f(x) \geq f(y) + d_y^\top (x - y)$ and $f(y) \geq f(x) + d_x^\top (y - x)$. This implies in turn that $I + \partial f$ is a *strictly monotone* operator, since

$$((y + d_y) - (x + d_x))^\top (y - x) = (d_y - d_x)^\top (y - x) + \|y - x\|^2 > 0 \ .$$

Now, let $y \in \text{prox}_f(x) = \text{argmin}_z\, f(z) + \frac{1}{2}\|z - x\|^2$. The first order optimality condition yields $0 \in \partial f(y) + y - x = (I + \partial f)(y) - x$ or equivalently $y \in (I + \partial f)^{-1}(x)$. But the inverse of strictly monotone operators are single-valued. □

The uniqueness of the proximal operator makes the optimization analysis simpler. In particular, note that, while f may have multiple minima, the proximal point of f at x is always unique.

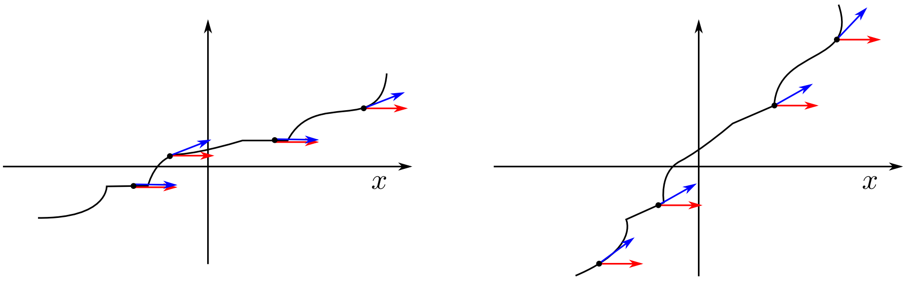


Figure 1.22: Monotone (left) and strictly monotone (right) operators.

A few basic properties and remarks are needed before fully exploiting the power of the proximal point operator.

Remark 17. *Note first that, for $\lambda > 0$,*

$$\text{prox}_{\lambda f}(x) = \underset{y \in \mathscr{X}}{\text{argmin}} \left\{ f(y) + \frac{1}{2\lambda}\|x - y\|^2 \right\}.$$

In particular, for differentiable f,

$$y = \text{prox}_{\lambda f}(x) = x - \lambda \nabla f(y) \iff y + \nabla f(y) = x.$$

As such, performing iterative steps (from x to y) resembles a backward gradient ascent algorithm, i.e., if we would have started with y, a step along the gradient of the destination point *would bring us to our* starting point *as such increasing the cost. The parameter λ here plays the role of the step size.*

Also remark that, still for differentiable f,

$$\nabla\left(f(y) + \frac{1}{2\lambda}\|y - x\|^2\right) = \nabla f(y) + \frac{1}{\lambda}(x - y)$$

and thus, at $y = x$, both f and $f + \frac{1}{2\lambda}\|x - \cdot\|^2$ have the same value and gradient. The proximal operator thus aims at minimizing a local approximation of f.

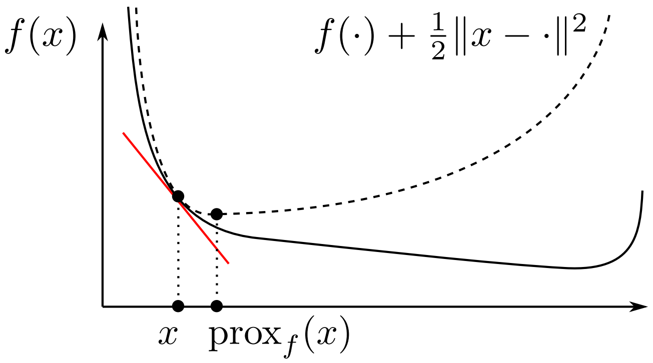


Figure 1.23: The proximal operator.

Remark 18 (Fenchel's convex conjugate and proximal operators). *Let $g(x) = f(x) - \frac{1}{2}\|x\|^2$, then*

$$y = \mathrm{prox}_g(u) = \mathrm{prox}_{f-\frac{1}{2}\|\cdot\|^2}(u) = \operatorname*{argmin}_{x\in\mathcal{X}} -u^\top x + f(x) = f^*(u)$$

Inversely,

$$y = \mathrm{prox}_f(u) = \operatorname*{argmin}_{x\in\mathcal{X}} -u^\top x + \frac{1}{2}\|x\|^2 + f(x) = \left(f + \frac{1}{2}\|\cdot\|^2\right)^*(u)$$

The key property of proximal operators lies in the fundamental property that the fixed-points of the proximal operator of f are exactly the minimisers of f. To see this, first remark that, if $\tilde{x} = \mathrm{prox}_f(x)$, then by definition

$$0 \in \partial f(\tilde{x}) + \tilde{x} - x. \tag{1.7}$$

Property 21 (Proximal fixed-points and minimisers). *The minimizers of f correspond to the fixed-point of the proximal operator, i.e.,*

$$x^\star \in \operatorname*{argmin}_{x\in\mathcal{X}} f(x) \Leftrightarrow 0 \in \partial f(x^\star) \Leftrightarrow x^\star = \mathrm{prox}_f(x^\star).$$

Proof. Observe simply that

$$\begin{aligned} x^\star \in \operatorname*{argmin}_{x\in\mathcal{X}} f(x) &\Leftrightarrow 0 \in \partial f(x^\star) \\ &\Leftrightarrow 0 \in \partial f(x^\star) + (x^\star - x^\star) \\ &\Leftrightarrow x^\star = \mathrm{prox}_f(x^\star) \end{aligned}$$

where the last equivalence is based on Equation (1.7) and $x^\star = \mathrm{prox}_f(x^\star) \implies x^\star \in \mathrm{argmin}_x f(x)$. □

This fundamental property suggests that the classical *fixed-point algorithm*, consisting in iterating $x_{k+1} = \mathrm{prox}_f(x_k)$ until convergence should lead to finding a minimum of f, *without resorting to differentiation.* However, for this to hold, one must show that these iterations do indeed converge.

To this end, a first idea is to show that prox_f is a *contractive mapping* (a function g is a contraction if g is α-Lipschitz for some $\alpha \in]0,1[$, in which case, for $x^\star = g(x^\star)$ a fixed point of g, $\|x_{k+1} - x^\star\| \le \alpha\|x_k - x^\star\|$ where $x_{k+1} = g(x_k)$ and thus $x_k \to x^\star$).

This is however not the case. The proximity operator is *not* contractive. Yet, it has a weaker but sufficient property, called *firm non-expansiveness*, which implies the even weaker property of *non-expansiveness*.

Definition 36 (Non-expansiveness). *An application $g : \mathcal{X} \to \mathcal{X}$ is* non-expansive *if, for all $x, y \in \mathcal{X}$,*

$$\|g(x) - g(y)\| \le \|x - y\|.$$

In other words, g is 1-Lipschitz. A proof of the non-expansiveness of the prox-operator follows from its firm non-expansiveness and the Cauchy-Schwarz inequality.

Definition 37 (Firm non-expansiveness). *An application $g : \mathcal{X} \to \mathcal{X}$ is* firmly non-expansive *if there exists a non-expansive mapping $G : \mathcal{X} \to \mathcal{X}$ such that $g = \frac{1}{2}(I + G)$, with $I : \mathcal{X} \to \mathcal{X}$, $x \mapsto I(x) = x$.*

Theorem 12. *For a convex function f, the proximal operator* $\mathrm{prox}_f : \mathcal{X} \to \mathcal{X}$, $x \mapsto \mathrm{argmin}_y f(y) + \frac{1}{2}\|x - y\|^2$ *is firmly non-expansive.*

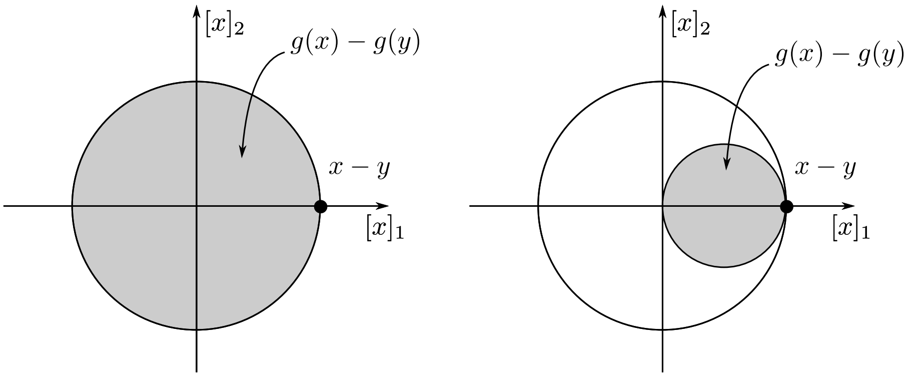


Figure 1.24: Non-expansive operator g (left) and firmly non-expansive operator g (right).

Proof. It suffices to prove that the mapping $2\text{prox}_f - I$ is non-expansive, i.e., for all $x, y \in \mathcal{X}$,

$$\left\|(2\text{prox}_f(x) - x) - (2\text{prox}_f(y) - y)\right\|^2 \leq \|x-y\|^2$$

or equivalently, expanding the left-hand side,

$$\left\|\text{prox}_f(x) - \text{prox}_f(y)\right\|^2 - (\text{prox}_f(x) - \text{prox}_f(y))^\mathsf{T}(x-y) \leq 0. \tag{1.8}$$

To this end, first recall that ∂f is a monotone operator. Hence for $a = \text{prox}_f(x)$ and $b = \text{prox}_f(y)$, by definition of the proximal operator, we have $x - a \in \partial f(a)$ and $y - b \in \partial f(b)$. Thus

$$((x - \text{prox}_f(x)) - (y - \text{prox}_f(y)))^\mathsf{T}(\text{prox}_f(x) - \text{prox}_f(y)) \geq 0$$

which implies $(\text{prox}_f(x) - \text{prox}_f(y))^\mathsf{T}(x-y) \geq \|\text{prox}_f(x) - \text{prox}_f(y)\|^2 \geq 0$. □

The main property of firmly non-expansive operators lies in the convergence of the fixed-point algorithm to a fixed-point of the operator. We will particularize this proof to the proximal operator, of interest here.

Theorem 13 (The Proximal Point Algorithm). *Let $f : \mathcal{X} \to \mathbb{R}$ be some convex function and define the sequence $x_1, x_2, \ldots$ through $x_1 \in \mathcal{X}$ and, for $k \geq 1$,*

$$x_{k+1} = \text{prox}_f(x_k).$$

Then $x_k \to x^\star$ for some $x^\star \in \text{argmin}_{x\in\mathcal{X}} f(x)$.

Proof. First write

$$\begin{aligned}
\|x_{k+1} - x_k\|^2 &= \left\|\text{prox}_f(x_k) - x_k\right\|^2 \\
&= \left\|(\text{prox}_f(x_k) - x_k) - (\text{prox}_f(x^\star) - x^\star)\right\|^2 \\
&= \left\|\text{prox}_f(x_k) - \text{prox}_f(x^\star)\right\|^2 + \|x_k - x^\star\|^2 - 2\left(\text{prox}_f(x_k) - \text{prox}_f(x^\star)\right)^\mathsf{T}\left(x_k - x^\star\right) \\
&\leq \|x_k - x^\star\|^2 - \|\text{prox}_f(x_k) - \text{prox}_f(x^\star)\|^2
\end{aligned}$$

where the last inequality follows directly from the firm non-expansiveness of prox_f, in particular:

$$(\text{prox}_f(x_k) - \text{prox}_f(x^\star))^\mathsf{T}(x_k - x^\star) \geq \left\|\text{prox}_f(x_k) - \text{prox}_f(x^\star)\right\|^2 \geq 0 \ .$$

This result is also geometrically interpreted in Figure 1.25.

From Equation (1.8), we obtain

$$\begin{aligned}
&\left\|\text{prox}_f(x) - \text{prox}_f(y)\right\|^2 - (\text{prox}_f(x) - \text{prox}_f(y))^\mathsf{T}(x-y) \leq 0 \\
\Longleftrightarrow\ & 2\left\|\text{prox}_f(x) - \text{prox}_f(y)\right\|^2 - 2(\text{prox}_f(x) - \text{prox}_f(y))^\mathsf{T}(x-y) + \left\|x-y\right\|^2 \leq \left\|x-y\right\|^2 \\
\Longleftrightarrow\ & \left\|\text{prox}_f(x) - \text{prox}_f(y)\right\|^2 + \left\|\left(I - \text{prox}_f\right)(y) - \left(I - \text{prox}_f\right)(x)\right\|^2 \leq \left\|x-y\right\|^2
\end{aligned} \tag{1.9}$$

and in particular $\left\|x_{k+1}-x^\star\right\|^2+\|x_{k+1}-x_k\|^2 \leq \left\|x_k-x^\star\right\|^2$ and $\left\|\mathrm{prox}_f(x)-\mathrm{prox}_f(y)\right\|^2 \leq \left\|x-y\right\|^2$. And now summing over $k=1,\dots,K$ finally yields

$$K\|x_{K+1}-x_K\|^2 \leq \|x_1-x^\star\|^2-\|x_{K+1}-x^\star\|^2 \leq \|x_1-x^\star\|^2$$

and thus

$$\|x_{K+1}-x_K\| \leq \frac{1}{\sqrt{K}}\|x_1-x^\star\|$$

which tends to 0 as $K\to\infty$. As a consequence, $\|\mathrm{prox}_f(x_k)-x_k\|\to 0$, hence x_k converges to a fixed-point of prox_f, so to a point in $\mathrm{argmin}_x f(x)$.

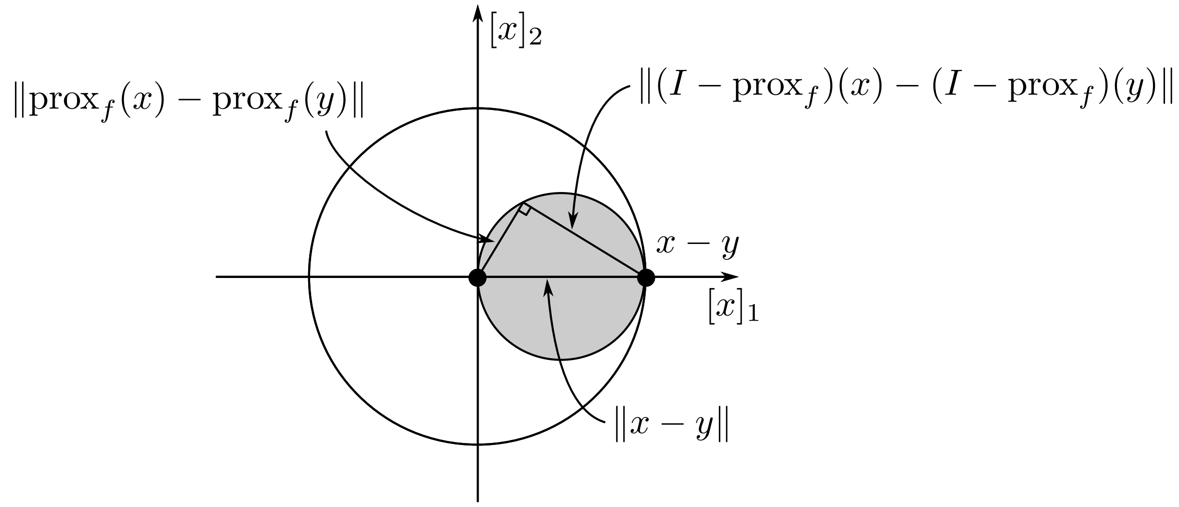

$[x]_2$
$\|\mathrm{prox}_f(x)-\mathrm{prox}_f(y)\|$
$\|(I-\mathrm{prox}_f)(x)-(I-\mathrm{prox}_f)(y)\|$
$x-y$
$[x]_1$
$\|x-y\|$

Figure 1.25: A geometric view of proximal inequality (1.9) by the Pythagorean theorem.

□

A few comments are of importance to fully grasp the importance of the proximal point algorithms.

Remark 19 (On the proximal point algorithm)**.** *The following comments are in order:*

- *first note that the proximal point algorithm necessitates neither f to be differentiable, nor any step size constraint; this is extremely advantageous when compared to the gradient-based approaches for which second-order controls are needed;*
- *one can freely change the cost function from f to λf for any $\lambda > 0$ without affecting the algorithm, however possibly gaining in performance; as a matter of fact, methods relying on introducing a step-size λ in the algorithm exist with λ possibly updated at each step to improve convergence;*
- *on the other hand, two main difficulties may arise: (i) the proximal operator itself may be difficult to evaluate and may itself require an independent optimization, (ii) in the worst case, the convergence rate is sublinear.*

In order to show that in many standard cases, the limitation (i) above is easily tackled, we subsequently introduce a list of proximal operators, set as an exercise.

Exercise 16 (Some proximal operators)**.** *Demonstrate the following proximal operator and gradient values*

f	$\mathrm{prox}_f(x)$	$\nabla f(x)$ -
0	x	0
$\imath_\Omega(x)$	$P_\Omega(x)$	-
$\imath_{\mathbb{R}^n_+}(x)$	$\{\max([x]_i,0)\}_{i=1}^N$	-
$\lambda\|x\|_1$	$\{\mathrm{sgn}([x]_i)\max(\lvert[x]_i\rvert-\lambda,0)\}_{i=1}^n$	-
$\imath_{\{\bar{x},A\bar{x}=y\}}(x)$	$x+A^\top(AA^\top)^{-1}(y-Ax)$	-
$\frac{1}{2}\|Ax-y\|^2$	$(I_n+A^\top A)^{-1}(x+A^\top y)$	$A^\top(Ax-y)$
$x^\top A^\top y$	$x-A^\top y$	$A^\top y$
$\frac{1}{2}x^\top Ax$	$(I_n+A)^{-1}x$	Ax

where P_Ω is the Euclidean projector on the set Ω.

1.5.3 Minimization of the Sum of Two Functions

Of utmost interest in proximal point algorithms is when used to write the optimization constraints as a convex but often non-differentiable additive cost. That is, when solving

$$\min_{x\in\mathscr{X}} f_1(x) + f_2(x)$$

where, often, f_2 is an indicator function on the constraint set, e.g., $f_2(x) = \iota_\Omega(x)$ for some (convex) validity set Ω.

In all generality though, we may be interested in scenarios where one aims at minimising the sum of two convex functions (which we recall is a convex function).

We start by considering the scenario where f_2 is *convex differentiable* with L-Lipschitz gradient ∇f_2, while f_1 is *only convex.*

Exploiting the set-operations defined by the proximity operator, solving the optimization problem can be easily written through the following sequence of equivalence relations:

$$\begin{aligned}
x^\star \in \operatorname{argmin}_{x\in\mathscr{X}} \{f_1(x) + f_2(x)\} &\Leftrightarrow 0 \in \partial f_1(x^\star) + \nabla f_2(x^\star)\\
&\Leftrightarrow 0 \in \gamma\partial f_1(x^\star) + \gamma\nabla f_2(x^\star)\\
&\Leftrightarrow x^\star \in x^\star + \gamma\partial f_1(x^\star) + \gamma\nabla f_2(x^\star)\\
&\Leftrightarrow x^\star - \gamma\nabla f_2(x^\star) \in x^\star + \gamma\partial f_1(x^\star)\\
&\Leftrightarrow x^\star = \operatorname{prox}_{\gamma f_1}((I-\gamma\nabla f_2)(x^\star))\\
&\Leftrightarrow x^\star = \left(\operatorname{prox}_{\gamma f_1} \circ (I-\gamma\nabla f_2)\right)(x^\star).
\end{aligned}$$

Thus here we are looking for a fixed-point of the operator

$$\operatorname{prox}_{\gamma f_1} \circ (I-\gamma\nabla f_2).$$

Note the seemingly arbitrary introduction of the parameter γ in this sequence of equations. As a matter of fact, γ is important here. Indeed, to ensure that the fixed-point algorithm will converge at all, one first needs to ensure that the operator $\operatorname{prox}_{\gamma f_1} \circ (I-\gamma\nabla f_2)$ is firmly non-expansive. This can be shown to be the case if $I-\gamma\nabla f_2$ is non-expansive, i.e., if $\gamma < \frac{1}{L}$.

This gives rise to the so-called *forward-backward splitting* algorithm defined in the following definition-theorem.

Theorem 14 (Forward-Backward Splitting)**.** *Let $f_1, f_2 : \mathscr{X} \to \mathbb{R}$ be two convex functions with f_2 differentiable and with L-Lipschitz gradient. For $x_1 \in \mathscr{X}$ and $k \geq 1$, define the sequence*

$$x_{k+1} = \operatorname{prox}_{\gamma f_1}\left(x_k - \gamma\nabla f_2(x_k)\right).$$

Then, as $k \to \infty$, $x_k \to x^\star$ for some

$$x^\star \in \operatorname{argmin}_{x\in\mathscr{X}} f_1(x) + f_2(x).$$

Note that the name *forward-backward splitting* refers to the fact that one step of the algorithm can be divided in two:

1. a first step to move from x_k to $\tilde{x}_k \equiv x_k - \gamma\nabla f_2(x_k)$, which is a gradient descent step on f_2, so a *forward* progression in the problem of minimizing f_2;
2. a second step to move from $\tilde{x}_k$ to $x_{k+1} = \operatorname{prox}_{\gamma f_1}(\tilde{x}_k)$, that is to move "*backward*" from $\tilde{x}_k$ to $x_{k+1} = (I+\partial f_1)^{-1}(\tilde{x}_k)$.

Remark 20 (Forward-Backward Splitting in Practice)**.** *The forward-backward splitting method is particularly convenient in practice when one aims at minimizing a convex differentiable function f_2 under convex constraints encapsulated in f_1. For instance, consider the problem*

$$\min_{x\in\Omega} f_2(x)$$

for $\Omega \subset \mathscr{X}$ a convex set. Then this can be rewritten under the form

$$\min_{x\in\mathscr{X}} f_1(x) + f_2(x)$$

where $f_1(x) = \iota_\Omega(x)$. The main advantage of this approach is that, as long as the proximal operation can be efficiently performed (here a projection on the set Ω), the constrained minimization problem has been turned into a much simpler unconstrained minimization of two functions.

Exercise 17. *Consider the following equality-constrained problem*

$$\min_{x,\ Ax=y} f(x)$$

for f_2 convex differentiable with L-Lipschitz gradient. Express the forward-backward splitting steps and provide a numerical algorithm. Compare with the Lagrange multipliers approach.

It should be now noted, however, that the forward-backward splitting method is constrained by the need for f_2 to be both differentiable and of Lipschitz gradient. Let us now try to relax this constraint.

Proceeding as before, it may seem convenient to propose an algorithm that iterates $(2\text{prox}_{\gamma f_2} - I) \circ (2\text{prox}_{\gamma f_1} - I)$. Indeed, we have the following equivalence steps

$$x = (2\text{prox}_{\gamma f_2} - I) \circ (2\text{prox}_{\gamma f_1} - I)(x) \Leftrightarrow x = 2\text{prox}_{\gamma f_2}(2\tilde{x} - x) - (2\tilde{x} - x)$$

where $\tilde{x} \equiv \text{prox}_{\gamma f_1}(x)$, i.e., $x - \tilde{x} \in \gamma \partial f_1(\tilde{x})$, which is further equivalent to

$$\begin{aligned}
&\Leftrightarrow 0 = \text{prox}_{\gamma f_2}(2\tilde{x} - x) - \tilde{x} \\
&\Leftrightarrow 2\tilde{x} - x \in (\gamma \partial f_2 + I)(\tilde{x}) \\
&\Leftrightarrow \tilde{x} - x \in \gamma \partial f_2(\tilde{x}) \\
&\Leftrightarrow 0 \in \gamma \partial f_1(x) + \gamma \partial f_2(x)
\end{aligned}$$

where in the last line we used the aforementioned relation $x - \tilde{x} \in \gamma \partial f_1(\tilde{x})$.

However, this method has the major issue of only providing *non-expansive* iterations, which does not guarantee convergence (recall indeed that prox_f being firmly non-expansive is equivalent to $2\text{prox}_f - I$ being non-expansive).

As this is not enough to ensure convergence, one needs to add an extra $\rho \in (0,1)$ in the resulting algorithm steps as follows.

Theorem 15 (Douglas-Rachford Splitting)**.** *Let $f_1, f_2 : \mathcal{X} \to \mathbb{R}$ be two convex functions. For $x_0 \in \mathcal{X}$, $\lambda > 0$, $\rho \in (0,1)$, and $k \geq 1$, define the following sequence*

$$\begin{aligned}
\tilde{x}_k &= \text{prox}_{\gamma f_1}(x_k) \\
x_{k+1} &= x_k + 2\rho \left(\text{prox}_{\gamma f_2}(2\tilde{x}_k - x_k) - \tilde{x}_k\right).
\end{aligned}$$

Then, as $k \to \infty$, $x_k \to\ x^\star$ for some

$$x^\star \in \text{argmin}_{x \in \mathcal{X}}\, f_1(x) + f_2(x).$$

Exercise 18. *Show that, for f_1, f_2 convex, $\lambda > 0$ and $\rho \in (0,1)$, both $(2\rho \cdot \text{prox}_{f_1} - I)$ and $(2\rho \cdot \text{prox}_{\gamma f_2} - I) \circ (2\text{prox}_{\gamma f_1} - I)$ are firmly non-expansive operators.*

Chapter 2

Lab Sessions

Generic Instructions:
The following lab sessions are described in an "open" manner, leaving a lot of room for self-conducting the session (as opposed to a list of precise instructions). The implementation may be performed in any code language, but preferably under Matlab (or Octave, SciLab) or Python.

For each lab session, an ***individual report*** *(jupyter notebook, matlab documented script, ..., converted to a 4–6 page report) containing (i) codes, (ii) results, and, most importantly, (iii) detailed comments (analysis of the results), is expected. These reports will be evaluated mostly on the capacity of the experimenter to properly explain his/her choices and methodology. Particular care should be taken to:*

- *justify the method being used (in particular the choice of dimensioning of the hyperparameters)*
- *explain the reasons why the proposed approaches should work and their limitations*
- *discuss and interpret the obtained results: what do they say? are they consistent with the expected/theoretical outputs? what is the dependence on the choice of hyperparameters ? what is the influence of initialisation? How do multiple methods compare (if applicable)?* Remember that a figure often says more than a thousand words!

2.1 Lab Session 1: Portfolio Optimization

We consider the portfolio optimization problem of Example 1.

2.1.1 Uniform Portfolio.

Under Matlab (or Python), import the datasets from the SP, NYSE and HSI stock market indexes (provided in **log returns** and not linear returns). We shall denote n the number of assets in the selected stock market.

For

$$w = w_{\text{uniform}} \equiv \frac{1}{n} 1_n$$

plot the values of the returns $x_t^{\mathsf{T}} w_{\text{uniform}}$ for the remaining 30 days. Then estimate, based on these 30 days, the obtained empirical variance (called *volatility* in finance).

2.1.2 Unconstrained Optimization.

We assume the stock market returns form a stationary (n-variate) time series. Estimate in particular the covariance matrix C by means of the sample covariance matrix of the n stock market (linear) returns $x_1, x_2, \ldots \in \mathbb{R}^n$ from 'day 1' to '30 days before the end of the recording' (be careful to use linear and not log returns).

Using the method of Lagrange multipliers, show that the problem

$$\operatorname{argmin}_{w \in \mathbb{R}^n} w^{\mathsf{T}} C w \text{ such that } \sum_{i=1}^{n} [w]_i = 1$$

has for unique solution

$$w_{\text{unconstrained}} = \frac{C^{-1} 1_n}{1_n^{\mathsf{T}} C^{-1} 1_n}.$$

Implement the formula and compare the values of the returns and the empirical variance now achieved to the uniform case. Comment also on the value of the entries of $w_{\text{unconstrained}}$.

2.1.3 Constrained Optimization.

The problem of the previous optimization is that it may return negative values for the entries of $w_{\text{unconstrained}}$, forcing "short selling". We now assume short selling is impossible (one can only buy and not sell on the market). We thus impose that $[w]_i \geq 0$ for all $i = 1, \ldots, n$.

As such, we now consider the constrained optimization problem with positivity constraints on the entries of w, that is

$$\operatorname{argmin}_{w \in \mathbb{R}^n} w^{\mathsf{T}} C w \text{ such that } \sum_{i=1}^{n} [w]_i = 1 \text{ and } [w]_i \geq 0.$$

Naive approach

We first propose to solve the problem by means of a simple 'naive' method consisting in proceeding to a gradient descent method on the unconstrained optimization problem mixed with projections to the constraint set.

Propose and implement such a method (or several approaches) and discuss their associated advantages and limitations.

Advanced optimization

We now proceed to a careful simplification of the problem by successively (i) *eliminating* the equality constraint (i.e., turning the problem into one with only inequality constraints) and (ii) solving the resulting inequality constrained optimization with a barrier method.

Explain why eliminating the equality constraint is convenient for the proposed barrier optimization approach. Could we have solved the problem differently? (think for instance of dual approaches)

We thus first proceed to the elimination of the equality constraint. To this end, show that we can recast the problem as

$$\operatorname{argmin}_{\tilde{w}\in\mathbb{R}^n} w^\mathsf{T} C w \text{ such that } w = P\tilde{w} + \frac{1}{n}1_n \text{ and } [w]_i \geq 0$$

for $P = I_n - \frac{1}{n}1_n 1_n^\mathsf{T}$, the projector operator on the orthogonal complement of span (1_n).

Show however that the resulting cost function has a vanishing gradient for all vectors $\tilde{w}$ on a non-trivial subspace. What practical implementation problem can that lead to?

To avoid the problem, we now impose $[\tilde{w}]_1 = 1$. Show that the solution to the previous problem is equivalent to that of

$$\operatorname{argmin}_{\tilde{w}\in\mathbb{R}^{n-1}} w^\mathsf{T} C w \text{ such that } w = P\left[\begin{smallmatrix}1\\ \tilde{w}\end{smallmatrix}\right] + \frac{1}{n}1_n \text{ and } [w]_i \geq 0$$

and that the resulting cost function no longer has a zero-gradient subspace.

As such, the problem can now be solved by a barrier method by relaxing the constraint $[w]_i \geq 0$ for all $i = 1,\ldots,n$, by a log-barrier constraint $-\mu\sum_{i=1}^n \log([w]_i)$. Start by computing the gradient of the new cost function, that is, the gradient with respect to $\tilde{w} \in \mathbb{R}^{n-1}$ of

$$\left(P\left[\begin{smallmatrix}\frac{1}{n}\\ \tilde{w}\end{smallmatrix}\right] + \frac{1}{n}1_n\right)^\mathsf{T} C\left(P\left[\begin{smallmatrix}\frac{1}{n}\\ \tilde{w}\end{smallmatrix}\right] + \frac{1}{n}1_n\right) - \mu\sum_{i=1}^n \log\left(\left[P\left[\begin{smallmatrix}\frac{1}{n}\\ \tilde{w}\end{smallmatrix}\right] + \frac{1}{n}1_n\right]_i\right).$$

Implement the barrier method to solve the problem. The solution will be denoted $w_{\text{constrained}}$. Be especially careful to ensure that each gradient descent step does not break a barrier constraint (backtracking with decreasing step sizes could be used here). A thin control on the progression of the algorithm is also recommended to avoid early stopping.

Finally evaluate the returns $x_t^\mathsf{T} w_{\text{constrained}}$ achieved on the 30 last days, plot $w_{\text{constrained}}$ and compute the variance of the returns. Compare to the uniform and constrained solutions.

2.1.4 Visualization of the Solution

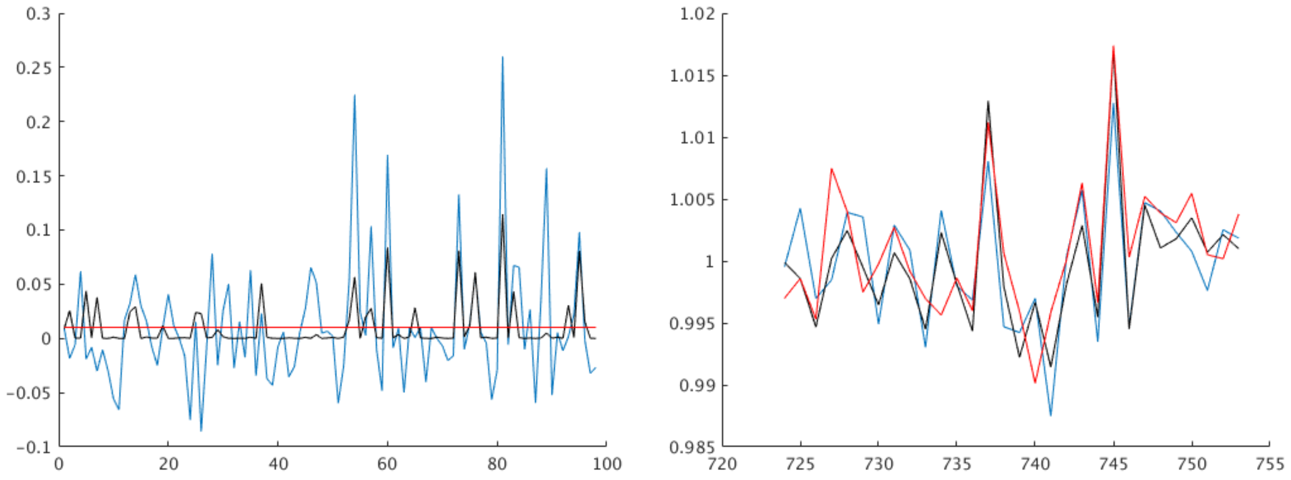


Figure 2.1: Portfolios allocated (left) and associated future returns (right) for the unconstrained optimization (blue), the constrained optimization (black) and the uniform allocation (red).

2.2 Lab Session 2: Support Vector Machines

We consider here the support vector machine application described in Example 2, where we impose $m > n$ (more observations than variables per vector observation). We will mostly deal with the "hard" thresholding case corresponding to the optimization problem:

$$(w^\star, b^\star) \in \operatorname{argmin}_{w\in\mathbb{R}^n, b\in\mathbb{R}} \frac{1}{2}\|w\|^2 \text{ such that } \forall i,\ y_i(w^\mathsf{T} x_i - b) \geq 1.$$

In what follows, we will write

$$\begin{cases} \tilde{w} = (w^\mathsf{T}, -b)^\mathsf{T} \\ P_w = \operatorname{diag}(1,1,\ldots,1,0) \in \mathbb{R}^{(n+1)\times(n+1)} \\ \tilde{x}_i = (x_i^\mathsf{T}, 1)^\mathsf{T} \end{cases} \implies \begin{cases} w = P_w \tilde{w} \\ x_i = P_w \tilde{x}_i \end{cases}$$

hence, the objective can be reformulated as

$$\tilde{w}^\star \in \operatorname*{argmin}_{\tilde{w}\in\mathbb{R}^{n+1}} \frac{1}{2}\|P_w \tilde{w}\|^2 \text{ such that } \forall i,\ y_i \tilde{w}^\mathsf{T} \tilde{x}_i \geq 1$$

In order to exploit proximal point methods, we will cast the problem as follows

$$\tilde{w}^\star \in \operatorname*{argmin}_{\tilde{w}\in\mathbb{R}^{n+1}} \frac{1}{2}\|P_w \tilde{w}\|^2 + \imath_{\{D_y \tilde{X}^\mathsf{T} \tilde{w} - 1_m \geq 0\}}(\tilde{w})$$

where $D_y = \operatorname{diag}(y_1,\ldots,y_m)$, $\tilde{X} = (\tilde{x}_1,\ldots,\tilde{x}_m)$ and we recall that $\imath_S(x) = 0$ if $x \in S$ while $\imath_S(x) = \infty$ is $x \notin S$.

We want to make use of the forward-backward algorithm and thus to take $f_1 = \imath_{\{D_y \tilde{X}^\mathsf{T} \tilde{w} - 1_m \geq 0\}}$. However, the proximal function for f_1 is not easily found and we shall then resort to the introduction of "slack variables" $\zeta \in \mathbb{R}^m$ to turn f_1 into a simpler function.

2.2.1 Slack Variable Method

The idea behind the slack variable approach is to note that the constraint

$$D_y \tilde{X}^\mathsf{T} \tilde{w} - 1_m \geq 0$$

can be equally written under the form

$$D_y \tilde{X}^\mathsf{T} \tilde{w} = \zeta, \quad \zeta \geq 1$$

for $\zeta \in \mathbb{R}^m$. With this simple tool, we have turned the inequality constraint into an equality constraint and a mere coordinate-wise inequality. We shall then proceed to (i) *eliminating* the equality constraint and (ii) exploiting a proximal approach on the simpler function $f_1 = \imath_{\{\zeta \geq 1\}}$.

Using the aforementioned slack variable approach, first show that we can cast equivalently the problem above as

$$\tilde{w} = \left(\tilde{X}\tilde{X}^\mathsf{T}\right)^{-1} \tilde{X} D_y \zeta$$

and

$$\zeta^\star \in \operatorname{argmin}_{\zeta\in\mathbb{R}^n} \frac{1}{2}\zeta^\mathsf{T} D_y \tilde{X}^\mathsf{T} \left(\tilde{X}\tilde{X}^\mathsf{T}\right)^{-1} P_w \left(\tilde{X}\tilde{X}^\mathsf{T}\right)^{-1} \tilde{X} D_y \zeta + \imath_{\{\zeta \geq 1\}}.$$

Next, we define

$$f_2(\zeta) \equiv \frac{1}{2}\zeta^\mathsf{T} D_y \tilde{X}^\mathsf{T} \left(\tilde{X}\tilde{X}^\mathsf{T}\right)^{-1} P_w \left(\tilde{X}\tilde{X}^\mathsf{T}\right)^{-1} \tilde{X} D_y \zeta \ .$$

Evaluate the gradient ∇f_2 of the function f_2 at position ζ and L the Lipschitz constant for the gradient of f_2.

Similarly, letting

$$f_1(\zeta) \equiv \imath_{\{\zeta \geq 1\}}$$

compute the proximal function $\operatorname{prox}_{\gamma f_1}(\zeta)$ of γf_1 at position ζ, for an arbitrary $\gamma > 0$.

Finally, propose a method based on the forward-backward algorithm to solve the original optimization problem in (w, b). For an unknown, new data point $x \in \mathbb{R}^n$, what is the class attributed by the algorithm solution to x?

2.2.2 Implementation

We shall now implement the proposed method on either *synthetic* or *real image* examples. To avoid loosing time on real image extraction, it is advised to start coding on the synthetic data only at first.

The real images will be extracted from the freely available MNIST database of handwritten digits

`http://yann.lecun.com/exdb/mnist/`

Matlab image extraction tools are available at

`http://ufldl.stanford.edu/wiki/index.php/Using_the_MNIST_Dataset`

Similar resources can be found for Python.

Start your code by creating a data extraction environment consisting in:

- **[Synthetic Data Choice]** Under this option, create both a *training* and a *test* sets, each made of $m = 50$ two-dimensional ($n = 2$) Gaussian random vectors $x_i \sim \mathcal{N}(0, I_2)$, half of which (25) have positive $[x_i]_2$ and half have negative $[x_i]_2$. These define the classes to be learned by the SVM.
- **[MNIST Data Choice]** Under this option, create similarly two sets of *training* and *test* data, each of size $m = 50$, subdivided into 25 images from a particular digit (from 0 to 9) and 25 images from another different digit. It is advised to start with easily differentiable digits, such as 0 and 1. To avoid too complex operations and to help visualization, we shall "compress" the images as follows: letting $Y = [y_1, \ldots, y_m] \in \mathbb{R}^n$ be the full set of training images, we let

  $$x_i \equiv U_2^\top y_i$$

  where $U_2 \in \mathbb{R}^{n\times 2}$ are the two dominant left singular vectors in the singular value decomposition of Y

  $$Y = USV^\top.$$

  To test the validity of the compression, once the operation performed, visualize an arbitrary compressed datum

  $$\hat{y}_i \equiv U_2 x_i = U_2 U_2^\top y_i$$

  (that is, $\hat{y}_i$ is the projection of y_i on the space of the two dominant singular vectors).

With the *training* data at hand, define $X = [y_1 x_1, \ldots, y_m x_m]$ where $y_i \in \{\pm 1\}$ according to the data classes. Then implement the forward-backward splitting algorithm to solve the SVM problem. The stopping criterion may be based on the iterated evolution in the vector ζ. Discuss the practical implementation "convenience" when opposed to the barrier method used in the previous lab-work.

Keep in memory the iterated values of the cost function

$$f_2(\zeta) = \|w\|^2$$

and display the graph of the cost function evolution with step iterations. Change the value of γ above and compare the evolutions. Comment and explain what you observe.

In a 2D-graph, display in different colors the points x_i associated to each class along with the separating hyperplane found by the algorithm. Comment on what you observe.

We now exploit the evaluated hyperplane to classify unseen data from the *test* sets. Display in the 2D-graph above the points of the test set and compute the proportion of misclassified data.

2.2.3 Soft Thresholding Case

Recall that the "hard" thresholding case considered so far is theoretically only valid (i.e., has a well defined solution) as long as the data are already separable in ambient space, i.e., there does exist a separating hyperplane for the data. In the case of the realistic MNIST dataset, this may not hold, especially for hard-to-discriminate digits. We thus instead consider the relaxed "soft" thresholding problem

$$(w^\star, b^\star) \in \operatorname{argmin}_{w,b\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^m \max(0, 1 - y_i[w^\top x_i - b]) + \lambda\|w\|^2$$

for some $\lambda > 0$. Using the same variable changes as above, determine the equivalent optimization problem over $\zeta \in \mathbb{R}^m$ and implement the solution using a forward-backward approach. It will be in particular necessary to determine $\operatorname{prox}_{\gamma f_1}$ for $f_1(\zeta) = \sum_{i=1}^m \max(0, -[\zeta]_i)$.

Play in particular on the parameter λ and discuss your findings. Compare with the hard thresholding case, particularly on hard-to-discriminate or even non-linearly separable data.

Visualization of the Solution

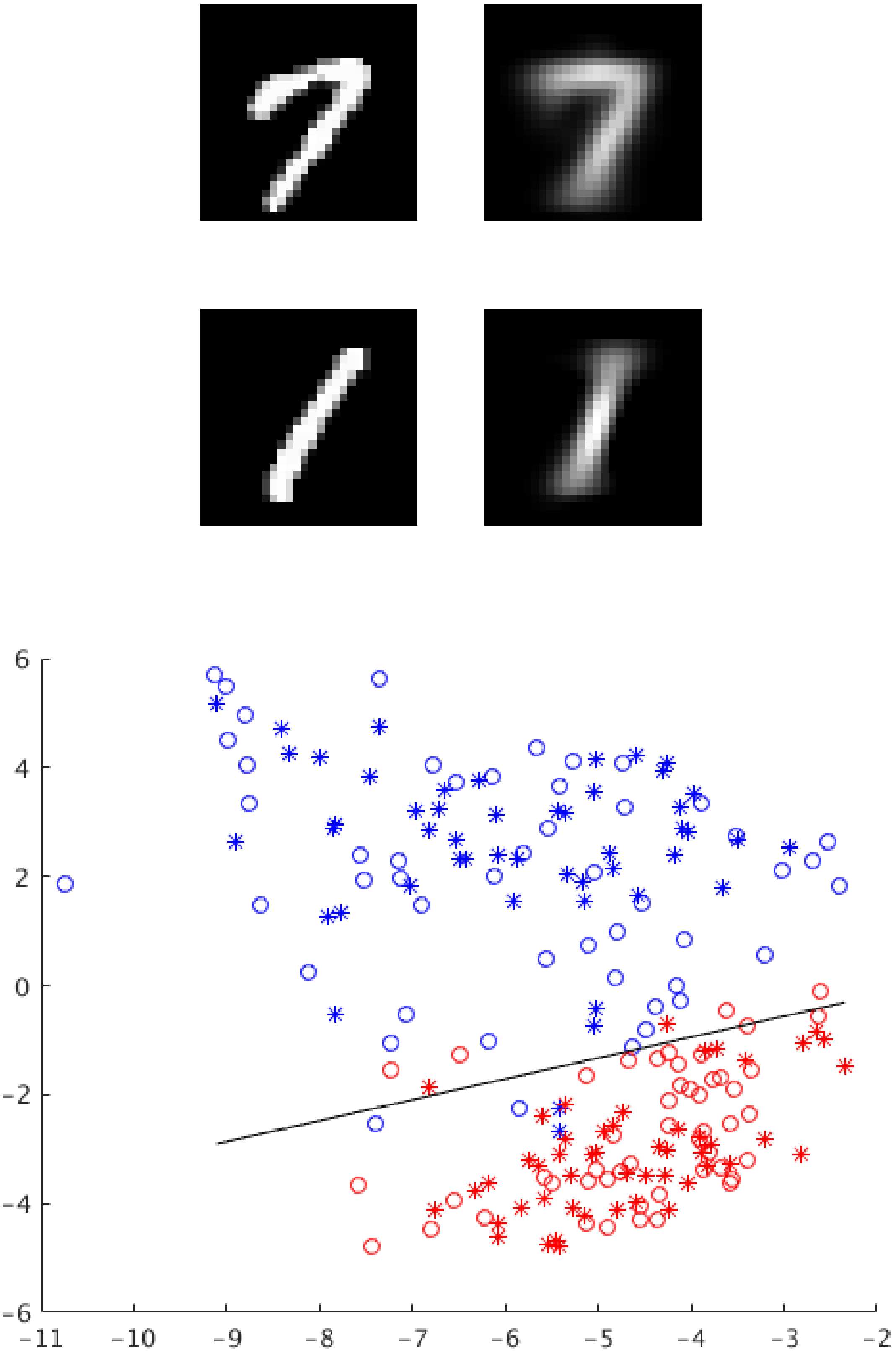


Figure 2.2: MNIST original and PCA-compressed ($n = 2$) images (top) and result from the soft-SVM implementation (bottom): training samples in blue versus red '$*$' and test samples in blue versus red '$\circ$'.

2.3 Lab Session 3: Compressive Sensing and Image Denoising

In this lab session, the objective is to perform image denoising exploiting the fact that images are "sparse" when seen in the Fourier domain. We shall rely on the compressive sensing setting of Example 3 to solve the problem

$$\mathscr{X}^\star \in \operatorname{argmin}_{\mathscr{X}\in\mathbb{R}^n} \|\mathscr{X}\|_1 + \iota_{\{y=\mathscr{A}\mathscr{X}\}} \tag{2.1}$$

where

- $\mathscr{X} \in \mathbb{C}^n$ is the 2D-Fourier transform of an original image $x \in \mathbb{R}^n$ (that is, composed of n pixels, for instance a $\sqrt{n} \times \sqrt{n}$ image) ;
- $y \in \mathbb{R}^s$, $s < n$ is a randomly sampled version of the pixels of x ;
- $\mathscr{A}$ is the linear operator $A \cdot \mathscr{F}^{-1}$ with $A \in \mathbb{R}^{s\times n}$ a random sampling matrix ($A_{ij} = 1$ if j is the i-th sampled element, and $A_{ij} = 0$ otherwise) and $\mathscr{F}$ the 2D-Fourier transform operator. An elementary sampling operator ;
- The ℓ_1-norms is to be understood as $\sum_{ij} \sqrt{|[\mathscr{X}]_{i,j}|^2}$.

We start the lab session by retrieving a greyscale (not too large) image, say of size 128 × 128, and set n as the total number of pixels in the image (that is, $n = 128^2$ in the aforementioned example). One may for instance take the popular `baboon` image from

`https://people.sc.fsu.edu/ jburkardt/data/png/baboon.png`

turn it into greyscale and scale the resolution by a factor $\frac{1}{4}$.

Extract such an image and create a vector $x \in \mathbb{R}^n$ of the pixel values. Then create a random mask $A \in \mathbb{R}^{s\times n}$ that induces a random sample vector

$$y = Ax$$

for s not too small compared to n (say $s > \frac{n}{4}$). The sampled image is then retrieved and can be observed from $A^\mathsf{T} y \in \mathbb{R}^n$.

For the problem at hand, evaluate the proximal operators of

$$\begin{aligned}\gamma f_1(\mathscr{X}) &\equiv \gamma\|\mathscr{X}\|_1\\ \gamma f_2(\mathscr{X}) &\equiv \iota_{\{y=\mathscr{A}\mathscr{X}\}}\end{aligned}$$

and implement these proximal operators using the fast 2D-Fourier transform and inverse transform routines (`fft2` and `ifft2` in Matlab for instance).

Next implement the Douglas-Rachford splitting method, with various values of γ and $\rho \in (0,1)$. Follow the evolution of the normalized mean-square error

$$\text{NMSE} \equiv \frac{\|x - x_k\|^2}{\|x\|^2}$$

for $x_k = \mathscr{F}^{-1}\mathscr{X}_k$ where $\mathscr{X}_k$ is the k-th iteration of the Douglas-Rachford splitting algorithm.

For different values of s/n, provide a comparative visual representation of (i) the original image x, (ii) the sampled image $A^\mathsf{T} y$, (iii) the solution image $x^\star = \mathscr{F}^{-1}\mathscr{X}^\star$ retrieved by the algorithm. A typical solution image is displayed in Figure 2.3.

As a comparison, you may implement a greedy approach that consists in retrieving one by one the dominant entries of $\mathscr{F} \cdot A^\mathsf{T} y$ until a certain percentage of the total "energy" is mined, and compare the resulting approximation to the original x obtained via this method or via the convex relaxation (2.1).

You may notice that the implementation of the products by $A \in \mathbb{R}^{s\times n}$ and A^T is computationally expensive, especially for large s. Improve the proposed method by means of faster computation methods. You may now rerun the developed approach on larger size images.

Visualization of the Solution

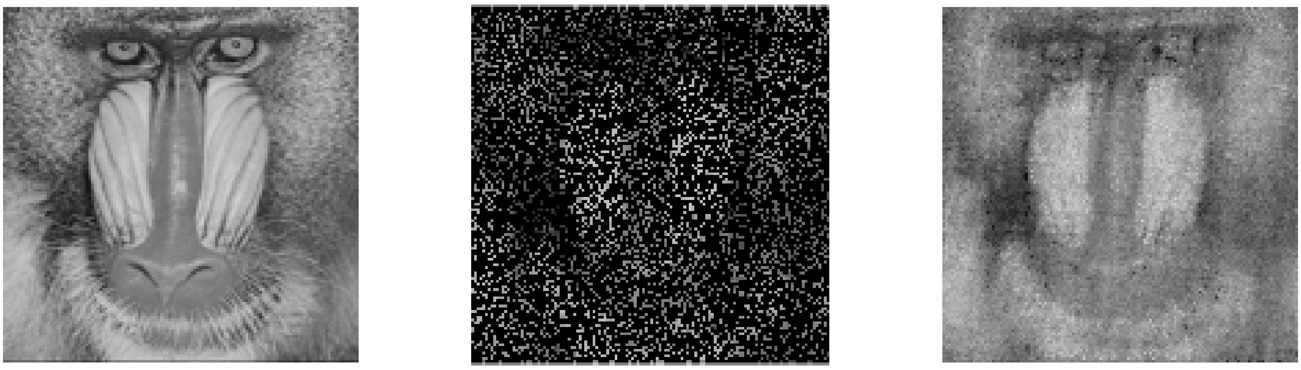


Figure 2.3: Original image (left), $s = \frac{n}{4}$ sampled version (center) and solution to the optimization problem (right).

Appendix A

Some useful inequalities

Property 22 (absolute value of sum)**.** $\forall x, y \in \mathbb{R} : \left||x| - |y|\right| \leq |x - y| \leq |x| + |y|$

absolute value of sum. Given that the inequalities are invariant to renaming of x and y, It is sufficient to convince oneself that the inequalities hold for the following three different cases $0 \leq x \leq y$, $x < 0 \leq y$, and $x \leq y < 0$. □

Property 23 (Cauchy–Bunyakovsky–Schwarz (inner product) inequality)**.** *Let X be a space endowed with an inner product. Then, $\forall x, y \in \mathcal{X}$ we have*

$$\left|\langle x, y\rangle\right| \leq \|x\|^2 \cdot \|y\|^2$$

Inner product inequality. Observe that if $y = 0$, the inequality holds trivially true. Now consider $y \neq 0$ and choose $\lambda = \frac{\langle x,y\rangle}{\|y^2}$, then

$$\begin{aligned} 0 \leq \|x - \lambda y\|^2 &= \|x\|^2 + |\lambda|^2 \|y\|^2 - \overline{\lambda}\langle x, y\rangle - \lambda\langle y, x\rangle \\ &= \|x\|^2 - \frac{\left|\langle x, y\rangle\right|^2}{\|y\|^2} \end{aligned}$$

□

Property 24 (Jensen's inequality)**.** *For any convex function f, probability measure μ ($\int_{\mathcal{X}} \mathrm{d}\mu = 1$), and μ-measurable function $g : \mathcal{X} \to \mathcal{X}$, we have*

$$f\left(\int g(x)\,\mathrm{d}\mu(x)\right) \leq \int f \circ g(x)\,\mathrm{d}\mu(x) \quad .$$

In the specific case where $\mu = \sum_{i=0}^{n-1} m_i \delta_{x_i}$ is an arbitrary convex combination of Dirac deltas, this brings us to

$$f\left(\sum_{i=0}^{n-1} m_i g(x_i)\right) \leq \sum_{i=1}^{n-1} m_i f \circ g(x_i) \quad .$$

Jensen's inequality. Define $x_0 = \int_{\mathcal{X}} g(x)\,\mathrm{d}\mu(x) \in \mathcal{X}$. Since f is a convex function, at each point x_0 of its domain $\mathcal{X}$, we have a non-empty set of subderivatives ∂f for which

$$\forall a \in \partial f(x_0), \forall x \in \mathcal{X} : f(x_0) + a(x - x_0) \leq f(x) \quad ,$$

where equality holds for $x = x_0$. We thus have

$$\int_{\mathcal{X}} f \circ g(x)\,\mathrm{d}\mu(x) \geq \int_{\mathcal{X}} f(x_0) + a\big(g(x) - x_0\big)\,\mathrm{d}\mu(x) = f(x_0) - a x_0 + a\int_{\mathcal{X}} g(x)\,\mathrm{d}\mu(x) = f\left(\int_{\mathcal{X}} g(x)\,\mathrm{d}\mu(x)\right) \quad .$$

□

Property 25 (Young's (product) inequality)**.** *Let $a, b \in \mathbb{R}_{\geq 0}$ and $p, q \in [1, +\infty]$ such that $\frac{1}{p} + \frac{1}{q} = 1$. Then,*

$$ab \leq \frac{a^p}{p} + \frac{b^q}{q} \quad ,$$

where equality holds of and only if $a^p = b^q$.

product inequality. Let $\lambda = \frac{1}{p}$, then $\frac{1}{q} = 1 - \lambda$ and we have

$$\log\left(\lambda a^p + (1-\lambda)\, b^q\right) \geq \lambda \log\left(a^p\right) + (1-\lambda) \log\left(b^q\right) = \log(ab) \quad ,$$

where we have used the concavity of log and Jensen's inequality. Since log is a strictly increasing function, the inequality follows. □

Property 26 (Hölder's (norm) inequality)**.** *Let $\left(\mathscr{X}, \Sigma, \mu\right)$ be a measure space and let $p, q \in [1, +\infty]$ with $\frac{1}{p} + \frac{1}{q} = 1$. Then, for all μ-measurable functions f and g on $\mathscr{X}$,*

$$\left\|fg\right\|_1 \leq \left\|f\right\|_p \left\|g\right\|_q$$

This property can be extended to spaces with the counting measure (vector norms ℓ_p and ℓ_q).

norm inequality. Use Young's product inequality pointwise and integrate. □

Property 27 (Minkowsky (norm) inequality)**.** *Let $\left(\mathscr{X}, \Sigma, \mu\right)$ be a measure space and let $1 \leq p \leq +\infty$. Let f and g be elements of $L^p(\mathscr{X})$. Then $f + g \in L^p(\mathscr{X})$ and the triangle inequality holds:*

$$\left\|f+g\right\|_p \leq \left\|f\right\|_p + \left\|g\right\|_p$$

with equality if and only if $\exists \lambda \in \mathbb{R}_{\geq 0}$ such that $f = \lambda g$ or $g = 0$. The norm is given by $\left\|f\right\|_p = \left(\int_{\mathscr{X}} \left\|f(x)\right\|^p \mathrm{d}\mu(x)\right)^{\frac{1}{p}}$.

One may extend the latter to spaces with the counting measure (vector norm ℓ_p).

norm inequality. First of all, show that the left-hand side exists, which follows from $\left|f+g\right|^p \leq 2^{p-1}\left(\left|f\right|^p + \left|g\right|^p\right)$. Write $p = 1 + (p-1)$, then $\left\|f+g\right\|^p = \int_{\mathscr{X}} \left|f+g\right| \cdot \left|f+g\right|^{p-1} \mathrm{d}\mu$. Using the triangle inequality on the first term under the integral sign and then Hölder's inequality to the two resulting integrals concludes the proof. □

Bibliography

[1] Moreau Jean J. Proximité et dualité dans un espace Hilbertien. *Bulletin de la société mathématique de France*, 93:273–299, 1965.