Non-convex inverse problems

Irène Waldspurger

waldspurger@ceremade.dauphine.fr

Initial version: January to March 2023
This version: January to February 2026

Contents

Acknowledgements

Many thanks to Claire Boyer and Vincent Duval for sharing their lecture notes with me.

Chapter 1

Introduction

What you should know / be able to do after this chapter

- Know the definition of "inverse problem", and a few examples.
- Understand what we call (in the context of this course) *theoretical aspects* and *algorithmic aspects* of an inverse problem. Know that the course will be about algorithmic aspects.
- Know the definition of "uniqueness" and "stability" in the context of inverse problems.
- For a linear problem, determine whether it is stable or not by looking at the singular values.
- With some guidance, be able to prove that a given inverse problem satisfies the uniqueness and stability properties (or not).
- Know our evaluation criteria for algorithms.
- Be able to determine whether a given problem is convex or not.
- Identify the main common points and differences between sparse and low-rank recovery.
- Understand the change of variable which turns phase retrieval into a low-rank matrix recovery problem.

1.1 Inverse problems

1.1.1 Definition

An *inverse problem* consists in identifying a (possibly complicated) object from a set of observations[1]. For instance, if we are given (two-dimensional) photographs of a building, viewed from different angles, reconstructing a three-dimensional model of the building is an inverse problem. Here, the "object" is the 3D shape of the building and the set of observations is the set of photographs.

Mathematically, these problems are formalized as follows. Let E be the set of possible *objects*, and F the set of possible *observations*. The *observation procedure* is described by a function $M : E \to F$. An inverse problem is, given some observation $y \in F$,

$$\boxed{\text{find } x \in E \text{ such that } M(x) = y.} \tag{Inverse}$$

Remark

The notion of *inverse problem* is often opposed to the notion of *direct problem*. A direct problem is the converse of an inverse problem: assuming the object and the observation procedure are known, compute the observations. For instance, if we are given a description of a fluid at some instant (viscosity, density, velocity at each point...), predicting how the fluid will be one minute later is a direct problem, which amounts to solving a specific partial differential equation. Here, the object is the fluid, and the observation procedure is "let it flow for one minute, then look at it".

1.1.2 Theoretical aspects

Problems of the form (Inverse) can be approached from two main angles.

- One can try to describe the properties of the solutions, without explicitly computing them. I will call this the *theoretical aspects*.

[1]Here, we will call *observation* any procedure which, from the object, produces an outcome.

- One can design algorithms to numerically solve the problem. I will call this the *algorithmic aspects.* [2]

This course is about algorithmic aspects. However, it is difficult to design a sensible algorithm if one has no idea at all of the properties of the solution. Therefore, in this section, we give a very brief overview of the theoretical aspects.

When given a specific instance of Problem (Inverse), a first question that arises is the *existence* of solutions: for an arbitrary y, does there always exist a solution x to Problem (Inverse)? If we restrict ourselves to vectors y which are the outcome of a real measurement process (that is, of the form $y = M(x)$ for some x), the answer is obviously yes. But if some errors have occured in the process, the answer may not be obvious anymore. For the problems we will consider in this course, existence will rarely be a problem, so we leave this question aside.

Assuming a solution exists, the other main two questions are *uniqueness* and *stability.*

- Uniqueness: Is the solution of Problem (Inverse) unique? This question is crucial, since, if the solution is not unique, it is impossible to recover the true object of interest with certainty.
  More formally, we say that Problem (Inverse) satisfies the uniqueness property if and only if
  $$\forall x_1, x_2 \in E \text{ such that } x_1 \neq x_2, M(x_1) \neq M(x_2).$$

- Stability: If y is not exactly known, but only available up to some error, what will the solution(s) of Problem (Inverse) look like? Will it be close to the "true" solution, the one we would have obtained if there had been no error on y? This is also crucial: in real life, exact measurements are never available.
  There are several sensible, but not equivalent, ways to translate this informal property to a formal one. A standard one is to say that Problem (Inverse) is stable if there exists constants $C_1, C_2 > 0$ such that

  $\forall x_1, x_2 \in E$ such that $x_1 \neq 0,$

[2]This choice of names does not mean that there is no "theory" behind algorithms. Actually, this course is about algorithmic aspects, but it will be mostly theoretical and rigorous.

$$C_1||x_1 - x_2||_F \leq ||M(x_1) - M(x_2)||_E \leq C_2||x_1 - x_2||_F, \quad (1.1)$$

and $\frac{C_2}{C_1}$ is "not too large" (say at most 10). Here, $||.||_E$ and $||.||_F$ are norms on E and F.[3]

Example 1.1 : finite-dimensional linear inverse problem

Let us assume that

- E, F are real finite-dimensional vector spaces: $E = \mathbb{R}^d$ and $F = \mathbb{R}^m$ for some $d, m \in \mathbb{N}^*$;
- $M : E \to F$ is linear, represented by some matrix $A \in \mathbb{R}^{m\times d}$.

Under these assumptions, Problem (Inverse) rewrites as

$$\boxed{\text{find } x \in \mathbb{R}^d \text{ such that } Ax = y.}$$

For a given y, assuming a solution x_* exists, it is *unique* if

$$\{x \in \mathbb{R}^d, Ax = y\} = \{x_*\},$$

that is if and only if $\mathrm{Ker}(A) = \{0\}$ (A is an injective matrix).
We now assume that the solution is unique. Is it *stable*? If the norms $||.||_E$ and $||.||_F$ in Equation (1.1) are the standard ℓ^2-norms, then it is possible to show that the problem is *stable* if and only if the smallest and largest singular values of A satisfy

$$\frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} \lesssim 10.$$

The ratio $\frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$ is called *condition number* of A.
For more details, see the exercises.

As said before, these questions will not be the subject of the course. For each newly encountered problem, we will try to give conditions under which

[3]These norms must in principle be carefully chosen according to the physical structure of the concrete underlying problem. Some choices may reflect better than others the desired properties of the solutions.

the solution is unique and stable but we will not spend much time on it. When these questions are not mentionned, the reader can simply assume that the considered problem satisfies uniqueness and stability properties. However, in principle, when facing a new problem, these questions must be the starting point, otherwise we are at risk of working towards the conception of algorithms for solving problems which can actually not be solved.

1.1.3 Our focus: algorithms

In this course, we will be interested in algorithms which allow to solve inverse problems. Cambridge dictionary defines the word *algorithm* as

"a set of mathematical instructions or rules that, especially if given to a computer, will help to calculate an answer to a problem."

Following this definition, an *algorithm* can take many forms. In particular, although the class of *iterative algorithms* (that is, those that repeat a set of instructions until some stopping criterion is met) will be of particular importance to us, one must not imagine that all algorithms are iterative.

In applications, a "good" algorithm is an algorithm which

- works: given a problem, it must output a correct solution; we can tolerate the algorithm failing once in a while, but the failure rate must be as small as possible;
- uses as few computational resources as possible: it must be fast (not too many operations) and have a moderate memory footprint.

Here, we will be interested in algorithms for which, moreover,

- these good properties (especially the first one) can be rigorously proved.

This additional requirement tends to be in contradiction with the computational efficiency, in the sense that, oftentimes, the algorithms which work best in practice are difficult to study rigorously. As a consequence, the algorithms we will present in this course will in most cases not be the best ones for real applications. They must be considered as toy models for "really usable" algorithms, should ideally retain as many specificities of their "really usable" counterparts as possible, but will inevitably miss some.

Similarly, the hypotheses under which we will establish correctness guarantees for the algorithms will often be much stronger than what holds in real applications. It is an important but difficult research direction to weaken these hypotheses.

1.2 Convex vs non-convex

All inverse problems can be reformulated as *optimization problems*, that is problems of the following form:

$$\begin{aligned}\text{minimize } & f(x)\\ \text{over all } & x \in H\\ \text{such that } & x \in C_1,\\ & \dots\\ & x \in C_S.\end{aligned} \tag{Opt}$$

Here, $f : H \to \mathbb{R} \cup \{+\infty\}$ can be any *objective* function, over a real or complex vector space H, and $C_1, \dots, C_S$ are subsets of H which model the constraints imposed on the unknown x.

An optimization problem is called *convex* if f is a convex function and $C_1, \dots, C_S$ are convex sets. By extension, we say that an inverse problem is convex if it can be reformulated as a convex optimization problem.

Definition 1.2 : convexity

A function $f : H \to \mathbb{R} \cup \{+\infty\}$ is *convex* if, for any $x_1, x_2 \in H$ and any $s \in [0; 1]$,

$$f((1-s)x_1 + sx_2) \leq (1-s)f(x_1) + sf(x_2). \tag{1.2}$$

A set $C \subset H$ is *convex* if, for any $x_1, x_2 \in C$ and any $s \in [0; 1]$, the vector

$$(1-s)x_1 + sx_2$$

is also an element of C.

Remark

You may also have encountered in previous courses the following characterizations of convex functions:

- if $f : \mathbb{R}^d \to \mathbb{R}$ is differentiable, it is convex if and only if, for any $x, y \in \mathbb{R}^d$,
$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle . \tag{1.3}$$
- if $f : \mathbb{R}^d \to \mathbb{R}$ is twice differentiable, it is convex if and only if, for any $x \in \mathbb{R}^d$,
$$\nabla^2 f(x) \succeq 0.$$

In first approximation, we can say that convex problems admit efficient algorithms. This is not an absolute rule, since some convex sets or functions are quite difficult to manipulate. However, it is true that many algorithms exist for convex problems, with a behavior which is quite well understood.

The situation is very different for the problems we will consider in this course, which are non-convex. For non-convex problems, the existence of algorithms both guaranteed to succeed and running in an reasonable amount of time is an exception. It is important to understand that this fact is due to the intrinsic difficulty of non-convex problems, and not to the fact that better algorithms exist, but we have not discovered them yet. In particular, many families of non-convex problems have been proved to be NP-difficult. This means that, unless P=NP, there exists no algorithm able to solve all problems in the family with a time complexity at most polynomial in their dimension. As a consequence, in this course, we will not try to propose algorithms able to solve all problems of a given non-convex family: this is hopeless. At best, our algorithms will be able to solve "a large part" of problems of the family.

1.3 Non-convex inverse problems: examples

Let us now present a few examples of non-convex inverse problems.

1.3.1 Sparse recovery - compressed sensing

Our first example is called *sparse recovery* or *compressed sensing*. It consists in recovering a vector $x \in \mathbb{R}^d$ from linear measurements
$$y \overset{def}{=} Ax \in \mathbb{R}^m,$$
where $A \in \mathbb{R}^{m \times d}$ is a known matrix, under the assumption that x is *sparse*. The word *sparse* means that x has a small number of non-zero coordinates:

for some $k \in \mathbb{N}^*$ much smaller than d,

$$||x||_0 \leq k,$$

where $||x||_0 = \text{Card}\{i \leq d, x_i \neq 0\}$. (This quantity is often called the ℓ^0-*norm*, although it is not a norm, since it is not homogeneous.)

Note that, if $m \geq d$ and A is injective, then this problem can be solved by inverting A; it is not necessary to use the sparsity assumption. This problem is only interesting when m is much smaller than d, in which case A is not injective and, if we were to ignore the sparsity assumption, y would not uniquely determine x.

Assuming that k is known, the problem can be written as

$$\boxed{\begin{aligned} &\text{recover } x \in \mathbb{R}^d \\ &\text{such that } Ax = y, \\ &\text{and } ||x||_0 \leq k. \end{aligned}} \tag{CS}$$

It is non-convex because the set $\{x, ||x||_0 \leq k\}$ is non-convex.

Sometimes, the unknown x is not directly sparse, but only sparse when represented in some adequate basis, or after some adequate linear transformation. In this case, the condition "$||x||_0 \leq k$" must be replaced with "$||\Phi x||_0 \leq k$", where Φ encodes the basis or linear transformation.

This problem is notably natural in image processing, since many natural images enjoy a sparsity structure. Photos, for instance, are well-known to be approximately sparse when represented in a *wavelet basis*.

For compressed sensing, uniqueness of the reconstruction can be guaranteed through a condition on the kernel of A.

Proposition 1.3 : unique recovery for compressed sensing

We assume that $\text{Ker}(A)$ does not contain a vector X such that $||X||_0 \leq 2k$. Then, if Problem (CS) has a solution, this solution is unique.

Proof. Let us assume, by contradiction, that Problem (CS) has two distinct solutions $X_1, X_2 \in \mathbb{R}^d$. Then

$$A(X_1 - X_2) = AX_1 - AX_2 = y - y = 0,$$

so $X_1 - X_2$ belongs to Ker(A). And

$$||X_1 - X_2||_0 \leq ||X_1||_0 + ||X_2||_0 \leq 2k,$$

which contradicts the assumption. $\square$

From this proposition, one can show that, if $m \geq 2k$, then almost all matrices A guarantee unique recovery of the underlying sparse vector. Under a stronger condition on A, one can also establish stability recovery guarantees (see for instance the introductory article [Candès and Wakin, 2008]).

1.3.2 Low rank matrix recovery

In low-rank matrix recovery, the goal is also to recover an object from linear measurements. This time, the "object" is a matrix $X \in \mathbb{R}^{d_1 \times d_2}$ (or $X \in \mathbb{C}^{d_1 \times d_2}$). As in the case of compressed sensing, there are not enough linear measurements to uniquely determine X without additinal information, but we do have some additional information on X: it is low-rank. This yields the problem

$$\boxed{\begin{aligned} &\text{recover } X \in \mathbb{R}^{d_1 \times d_2} \\ &\text{such that } \mathcal{L}(X) = y, \\ &\text{and } \operatorname{rank}(X) \leq r. \end{aligned}} \qquad \text{(Low rank)}$$

Here, $\mathcal{L} : \mathbb{R}^{d_1 \times d_2} \to \mathbb{R}^m$ is the linear measurement operator and r is a given upper bound on the rank of the matrix. Given that any $d_1 \times d_2$ matrix has rank at most $\min(d_1, d_2)$, the rank constraint is only useful if $r < \min(d_1, d_2)$. In some applications, it is relevant to assume that $d_1 = d_2$ and X is semidefinite positive: $X \succeq 0$.

This problem is sometimes called *matrix sensing*, especially when $\mathcal{L}$ is a random operator. A uniqueness result similar to Proposition (1.3) holds.

Proposition 1.4 : uniqueness for low-rank matrix recovery

We assume that Ker$(\mathcal{L})$ does not contain a matrix X such that

$$\operatorname{rank}(X) \leq 2r.$$

Then, if Problem (Low rank) has a solution, this solution is unique.

The proof of the proposition is identical to Proposition 1.3. From this proposition, one can show (but it is not easy) that the solution of Problem (Low rank), when it exists, is unique, for almost all operators $\mathcal{L}$, provided that

$$\begin{aligned} m &\geq 2r(d_1+d_2-2r) \qquad \text{if } 2r \leq \min(d_1,d_2), \\ &\geq d_1 d_2 \qquad \text{if } \min(d_1,d_2) < 2r < 2\min(d_1,d_2). \end{aligned}$$

When r is small (of order 1, for instance), this shows that we can hope to recover the "true" matrix X with a number of linear measurements much smaller than what we would need if we did not know X to be low-rank (in this case, we would need $m \geq \dim(\mathbb{R}^{d_1\times d_2}) = d_1 d_2$, which is much larger than $2r(d_1+d_2-2r)$ if $r \ll \min(d_1,d_2)$).

Matrix completion Several special cases of Problem (Low rank) are of particular interest, and form subfamilies of inverse problems with their own applications and theoretical characteristics. The first one is *matrix completion*. In this case, the linear measurements available on X are a subset of coefficients:

$$\boxed{\begin{array}{c} \text{recover } X \in \mathbb{R}^{d_1\times d_2} \\ \text{such that } X_{ij} = y_{ij}, \forall (i,j) \in \Omega \\ \text{and } \operatorname{rank}(X) \leq r. \end{array}} \qquad \text{(Matrix completion)}$$

Here, $\Omega \subset \{1,\dots,d_1\} \times \{1,\dots,d_2\}$ contains the indices of available coefficients.

The most popular application is the so-called "*Netflix* problem".[4] In this application, X represents the opinion of users on films: the coefficient X_{ij} is an "affinity score" between User i and Film j (it represents how much User i would like Film j). It is reasonable to assume that X is low-rank:[5] this models the similarities between the users, and between the films (e.g. if User 1 and 2 have the same opinion on Films $1,2,3,4$, it is plausible that they also have essentially the same opinion on Film 5). The available coefficients X_{ij}

[4]asked by Netflix in 2006, with a $1,000,000\$$ prize, and declared solved in 2009

[5]Keep however in mind that this assumption is only approximately satisfied by the "true" Netflix affinity scores matrix. On the other hand, the true matrix has additional structure that can be exploited to solve the problem.

correspond to pairs (i, j) for which User i has watched Film j and sent the corresponding score to the film distribution platform. The other coefficients are not available, but the platform would like to guess them, so as to be able to propose relevant film suggestions to their users. Guessing the non-available coefficients exactly amounts to solving Problem (Matrix completion).

Phase retrieval Another special case of Problem (Low rank) which we will discuss in length in this course is *phase retrieval*.

At first sight, phase retrieval problems have nothing to do with matrices and low-rankness. They are problems of the following general form

$$\boxed{\begin{aligned} &\text{recover } x \in \mathbb{C}^d \\ &\text{such that } |L_j(x)| = y_j, \forall j \leq m. \end{aligned}} \tag{Phase retrieval}$$

Here, $L_1, \dots, L_m : \mathbb{C}^d \to \mathbb{C}$ are known linear operators, the notation "$|.|$" stands for the usual complex modulus, and $y_1, \dots, y_m$ are given.

The main motivations for studying phase retrieval come from the field of imaging. Indeed, it is much easier to record the intensity (that is, the modulus, in an adequate mathematical model) of an electromagnetic wave than its phase. It is therefore frequent to have to recover an object from modulus-only measurements. Oftentimes, these measurements can specifically be described by a Fourier transform (because, under some assumptions, the diffraction pattern of an object is the Fourier transform of its characteristic function), but not always. Phase retrieval is also of interest for audio processing.

Remark

For any $x \in \mathbb{C}^d$ and $u \in \mathbb{C}$ such that $|u| = 1$, it holds

$$|L_j(ux)| = |uL_j(x)| = |u|\,|L_j(x)| = |L_j(x)|, \quad \forall j \leq m.$$

Therefore, the sole knowledge of $(y_j = |L_j(x)|)_{j \leq m}$ can never allow to exactly recover x. There is always a *global phase ambiguity*: x cannot be distinguished from ux.
This is in general not harmful in applications, and we will be satisfied if we can recover x up to a global phase.

Given specific linear forms L_j, it is in general difficult to determine if the (Phase retrieval) problem satisfies the uniqueness and stability properties. However, it is known that uniqueness holds "in principle" as soon as m is larger than (roughly) $4d$.

Proposition 1.5 : [Conca, Edidin, Hering, and Vinzant, 2015]

Let us assume that $m \geq 4d - 4$. Then, for almost all linear maps $L_1, \dots, L_m : \mathbb{C}^d \to \mathbb{C}$, it holds that, for all $x, x' \in \mathbb{C}^d$,

$$\big(|L_j(x)| = |L_j(x')|, \forall j \leq m\big) \quad \Rightarrow \quad \big(\exists u \in \mathbb{C}, |u| = 1, x = ux'\big).$$

With a slightly larger m, stability also "generically" holds.

Let us now explain why phase retrieval is a special case of low-rank matrix recovery. Readers which are not perfectly comfortable with the notions of Hermitian matrices and of semidefinite positive matrices should first read Appendix A.

The crucial ingredient is an adequate change of variable: instead of recovering $x \in \mathbb{C}^d$ up to a global phase, let us try to recover

$$X \overset{def}{=} xx^* = \begin{pmatrix} |x_1|^2 & x_1\overline{x}_2 & \dots & x_1\overline{x}_d \\ x_2\overline{x}_1 & |x_2|^2 & \dots & x_2\overline{x}_d \\ \vdots & & \ddots & \vdots \\ x_d\overline{x}_1 & & \dots & |x_d|^2 \end{pmatrix}.$$

Remark

A matrix $X \in \mathbb{C}^{d\times d}$ can be written as $X = xx^*$ for some $x \in \mathbb{C}^d$ if and only if

$$X \succeq 0 \quad \text{and} \quad \text{rank}(X) \leq 1.$$

When these conditions hold, x is equal, up to a global phase, to

$$\sqrt{\lambda_1} z_1,$$

where λ_1 is the largest eigenvalue of X, and z_1 any unit-normed eigenvector for this eigenvalue.

Proof. For any $x \in \mathbb{C}^d$, the matrix xx^* is Hermitian, and semidefinite positive:

$$\forall z \in \mathbb{C}^d, \quad \langle z, xx^* z\rangle = z^*(xx^*)z = |z^*x|^2 \geq 0.$$

It has rank at most 1 because $\text{Range}(xx^*) = \text{Vect}\{x\}$.

Conversely, if $X \succeq 0$ and $\text{rank}(X) \leq 1$, then X can be diagonalized in an orthogonal basis $(z_1, \dots, z_d)$:

$$X = \sum_{k=1}^{d} \lambda_k z_k z_k^* \quad \text{with } \lambda_1 \geq \dots \geq \lambda_d \text{ the eigenvalues.}$$

All the eigenvalues are nonnegative, since $X \succeq 0$. Since $\text{rank}(X) \leq 1$, they are all 0, except possibly the first one, so

$$X = \lambda_1 z_1 z_1^* = (\sqrt{\lambda_1} z_1)(\sqrt{\lambda_1} z_1)^*,$$

so it can be written as $X = xx^*$ with $x = \sqrt{\lambda_1} z_1$. This proves the first part of the remark.

For the second part, let us assume that $X = xx^*$ for some $x \in \mathbb{C}^d$. We have just seen that X is also equal to $\tilde{x}\tilde{x}^*$ for $\tilde{x} = \sqrt{\lambda_1} z_1$. We must simply show that x and $\tilde{x}$ are equal up to a global phase. As

$$\text{Vect}\{x\} = \text{Range}(X) = \text{Vect}\{\tilde{x}\},$$

it holds that x and $\tilde{x}$ are colinear: there exists $u \in \mathbb{C}$ such that $x = u\tilde{x}$. In addition,

$$||x||^2 = \text{Tr}(X) = ||\tilde{x}||^2,$$

hence x and $\tilde{x}$ have the same norm. As $||x|| = |u|\,||\tilde{x}||$, this implies that $|u| = 1$: x and $\tilde{x}$ are equal up to a global phase. □

From the previous remark, it is equivalent to recover x up to a global phase or X. Indeed, X can be computed from x (even up to a global phase: $(ux)(ux)^* = u\bar{u}xx^* = xx^*$ if $|u| = 1$) and x can be computed up to a global phase from X by extracting the only eigenvector of X with non-zero eigenvalue.

In addition, for any j, knowing $|L_j(x)|$ is equivalent to knowing $|L_j(x)|^2$. Denoting v_j the vector such that $L_j = \langle v_j, . \rangle$, we have

$$\begin{aligned} |L_j(x)|^2 &= \langle v_j, x \rangle \overline{\langle v_j, x \rangle} \\ &= (v_j^* x)(x^* v_j) \\ &= v_j^* X v_j. \end{aligned}$$

Consequently, Problem (Phase retrieval) is equivalent to

$$\boxed{\begin{aligned} &\text{recover } X \in \mathbb{C}^{d\times d} \\ &\text{such that } v_j^* X v_j = y_j^2, \forall j \le m, \\ &X \succeq 0, \\ &\operatorname{rank}(X) \le 1. \end{aligned}} \qquad \text{(Matrix PR)}$$

This is, as announced, a low rank matrix recovery problem.

1.3.3 Other examples

These examples will not be covered in class (except for super-resolution, but later); they are provided for curious readers only.

Machine learning In a machine learning task, the goal is to predict some output y given some input x. For instance, the input can be a photograph, and the output the name of the objects represented on the photograph, or the input can be a low-quality audio signal and the output the corresponding high-quality signal. We denote P the "perfect" prediction function, which to an input x maps the correct

$$y = P(x).$$

The predictor P is unknown and must be learned from the available input-output examples $(x_1, y_1), \dots, (x_n, y_n)$. This leads to the problem

$$\boxed{\begin{aligned} &\text{find } P \in \mathcal{H} \\ &\text{such that } P(x_k) = y_k, \forall k \le n, \end{aligned}} \qquad \text{(ML)}$$

where $\mathcal{H}$ is a well-chosen class of functions ($\mathcal{H}$ can for instance be the set of linear maps, or the set of neural networks with a given architecture).

The questions raised by Problem (ML) are quite different from the ones raised by the other inverse problems we have seen. Indeed, it often happens that the perfect predictor P is not in the chosen set $\mathcal{H}$, in which case the problem may not have an exact solution, only an approximate one. In addition, if $\mathcal{H}$ is a bit sophisticated, there are typically several (and even many) elements $P \in \mathcal{H}$ such that $P(x_k) = y_k$ for all k (in other words, the uniqueness property does not hold). All these elements P yield the same predictions for the available inputs $x_1, \dots, x_n$, but may differ significantly on unseen examples. It is therefore important to choose, among these P, the one which has the best chances to perform well on unseen examples (i.e. which *generalizes* best)

Dictionary learning In this problem, one is given a set of "interesting" signals $y_1, \dots, y_m \in \mathbb{R}^d$ (e.g. patches of natural photographs or of medical images), and the goal is to learn a good "representation" for them, under the form of a *dictionary.* A dictionary is a set of elements $a_1, \dots, a_M \in \mathbb{R}^d$, usually called *atoms*, such that any signal y_k can be written as a linear combination of a small number of atoms:

$$y_k = \sum_{l=1}^{M} \lambda_l^{(k)} a_l \quad \text{such that } ||\lambda^{(k)}||_0 \text{ is small.}$$

We write the dictionary in matricial form by concatenating the atoms into a single matrix:

$$A = \begin{pmatrix} a_1 & a_2 & \dots & a_M \end{pmatrix}$$

Finding the dictionary A consists in solving the following problem

$$\boxed{\begin{aligned} &\text{find } A \in \mathbb{R}^{d\times M}, \lambda^{(1)}, \dots, \lambda^{(m)} \in \mathbb{R}^M \\ &\text{such that } A\lambda^{(k)} = y_k, \forall k \leq m, \\ &||\lambda^{(k)}||_0 \leq S, \end{aligned}} \qquad \text{(Dictionary learning)}$$

where S is an a priori bound on the number of atoms involved in the decomposition of each signal y_k.

Super-resolution *Super-resolution* is a general term which covers all problems where one tries to recover a "sharp" signal from a "blurred" version. In this paragraph, we present the simplest possible model for such a problem.

The signal we aim at identifying is a collection of point masses in $[0;1[$. The positions of the masses are $\tau_1, \dots, \tau_S$ and their weights are $a_1, \dots, a_S$. This signal can be represented by a measure

$$\mu = \sum_{s=1}^{S} a_s \delta_{\tau_s} \in \mathcal{M}([0;1[),$$

where $\mathcal{M}([0;1[)$ is the set of signed (or even complex-valued, if $a_1,\dots,a_S$ are complex) finite Borel measures on $[0;1[$ and, for any s, δ_{τ_s} is the dirac at position τ_s.[6]

The "blurred" version of the signal is modelled as the set of low-frequency coefficients of the Fourier transform of μ: for all $k=-N,\dots,N$, we have access to

$$\hat{\mu}[k] = \int_0^1 e^{-2\pi ikt} d\mu(t) \left(= \sum_{s=1}^{S} a_s e^{-2\pi ik\tau_s} \right).$$

If we call $y_{-N},\dots,y_N$ the known Fourier coefficients, the problem can be written as

$$\boxed{\begin{gathered}\text{find } \mu \in \mathcal{M}([0;1[) \\ \text{such that } \hat{\mu}[k] = y_k, \forall k = -N,\dots,N, \\ \text{and } \mu \text{ is a sum of } S \text{ diracs.}\end{gathered}} \qquad \text{(Super-resolution)}$$

This problem can be seen as a continuous version of compressed sensing (Problem (CS)). The unknown, instead of a finite-dimensional vector, is a measure on $[0;1[$, but it must still be recovered from linear measurements, and satisfies a sparsity constaint (it is the sum of at most S diracs).

[6]that is to say, δ_{τ_s} is the measure such that, for any measurable $E\subset[0;1[$, $\mu(E)=1$ if $\tau_s\in E$ and $\mu(E)=0$ otherwise.

Chapter 2

General non-convex optimization

What you should know / be able to do after this chapter

- Remember that, while it is a reasonable goal to find an approximate minimizer of a convex function, the same objective is in general not reasonable for a non-convex function.
- Know the definition of first-order and second-order critical points.
- Know that a local minimizer is a second-order critical point, that the converse may not be true, but is true for so-called Morse functions.
- Know that approximate second-order critical points can be found using a second-order method (at least for smooth enough objective functions).
- Describe the Trust Regions algorithm (without the detailed radius update procedure).
- About gradient descent:
  - know a set of assumptions under which gradient descent iterates necessarily converge (Theorem 2.11);
  - know that, under these assumptions, the limit is a first-order critical point;
  - know that, under these assumptions, the limit is a second-order critical point for almost any initial point;
  - be able to sketch the proof in the case where the objective function is quadratic around any first-order critical point.

The aim of this chapter is to give an overview of general non-convex optimization algorithms.

To simplify the discussion, let us restrict ourselves to a finite-dimensional and *unconstrained* optimization problem. "Unconstrained" means that, in Problem (Opt), there are no constraint sets C_s. In other words, we consider a problem of the following form:

$$\text{minimize } f(x) \text{ over all } x \in \mathbb{R}^d.$$

For simplicity, we assume that a minimizer exists.

In Section 2.1, we explain why finding a minimizer of f is possible (at least approximately) when f is convex, but in general extremely difficult when f is non-convex. For non-convex f, optimization algorithms can at best find a so-called *critical point*. We define two versions of this notion: *first-order* and *second-order critical points.*

In Sections 2.2 and 2.3, we turn to concrete algorithms, and describe the convergence guarantees of standard optimization methods towards, respectively, first and second-order critical points.

2.1 Critical points versus minimizers

2.1.1 Finding minimizers of non-convex maps is difficult

Definition 2.1

A point $x_* \in \mathbb{R}^d$ is a *(global) minimizer* of f if

$$f(x) \geq f(x_*), \forall x \in \mathbb{R}^d.$$

It is a *local minimizer* of f if there exists $r > 0$ such that

$$f(x) \geq f(x_*), \forall x \in B(x_*, r).$$

(Here, $B(x_*, r)$ is the ball centered at x_* with radius r.)

You have seen in previous courses that, when f is convex, under reasonable assumptions (some form of smoothness of f, in particular), it is possible to numerically find an approximate minimizer, with an arbitrary level of precision. Intuitively, one reason for this is that convexity allows to deduce global information from local one. For instance, if one knows the values at a few points of a convex function f and its gradient, one has access (from Inequalities (1.2) and (1.3)) to upper and lower bounds on f. Hence, one can obtain an approximation of the minimum. It is then possible to query the values at other points to refine the approximation. This is illustrated on Figures 2.1a and 2.1b.

In addition, a local minimizer of a convex map is necessarily a global one.[1] Therefore, it seems reasonable that algorithms based on *local* update rules can find a *global* minimizer.

But if the function does not satisfy a property comparable to convexity, the knowledge of its values at a few points provides no information about the values at other points and, in particular, no information on its minimum. This is illustrated on Figures 2.1c and 2.1d. Moreover, non-global local minima may exist, and can trap iterative algorithms which use local information. This is what makes non-convex optimization much more difficult than convex optimization.

In the case where f is not convex, an intuitive strategy to find a global minimizer of f is to query information on f (its value, at least) at all points of a fine grid of $\mathbb{R}^d$ (more realistically, a bounded subset thereof), and approximate the global minimizer with the minimizer on this fine grid. This strategy indeed works, if f is somewhat smooth, in the sense that the output is an approximate minimizer, with a precision that can be made arbitrarily good if the grid is refined. However, it is extremely slow as soon as d is larger than 1 or 2: the number of points in a grid of (for instance) $[0;1]^d$ with spacing ϵ is $\left(\frac{1}{\epsilon}\right)^d$, which is prohibitive as soon as $d \geq 2$ or 3. Unfortunately, this simple intuitive strategy is essentially optimal, implying that *it is not possible to design an algorithm which globally minimizes any non-convex function in a reasonable amount of time*[2].

Thus, what can we expect from a good non-convex optimization algorithm? It won't be able to find global minimizers with certainty. Can it at least be guaranteed to find a *local* minimizer, if one exists? It turns out that this is also out of reach: there are functions, even polynomial ones, for which

[1] Proof: Let x_{loc} be a local minimizer. For any $x \in \mathbb{R}^d$, it holds that $f(x_{loc}) \leq f((1-s)x_{loc} + sx)$ for any $s \in [0;1]$ close enough to 0 (as x_{loc} is a local minimizer). From the definition of convexity, this implies that $f(x_{loc}) \leq (1-s)f(x_{loc}) + sf(x)$, hence $f(x_{loc}) \leq f(x)$. As this is true for any x, x_{loc} is a global minimizer.

[2] even if we restrict ourselves to non-convex functions satisfying standard additional smoothness assumptions

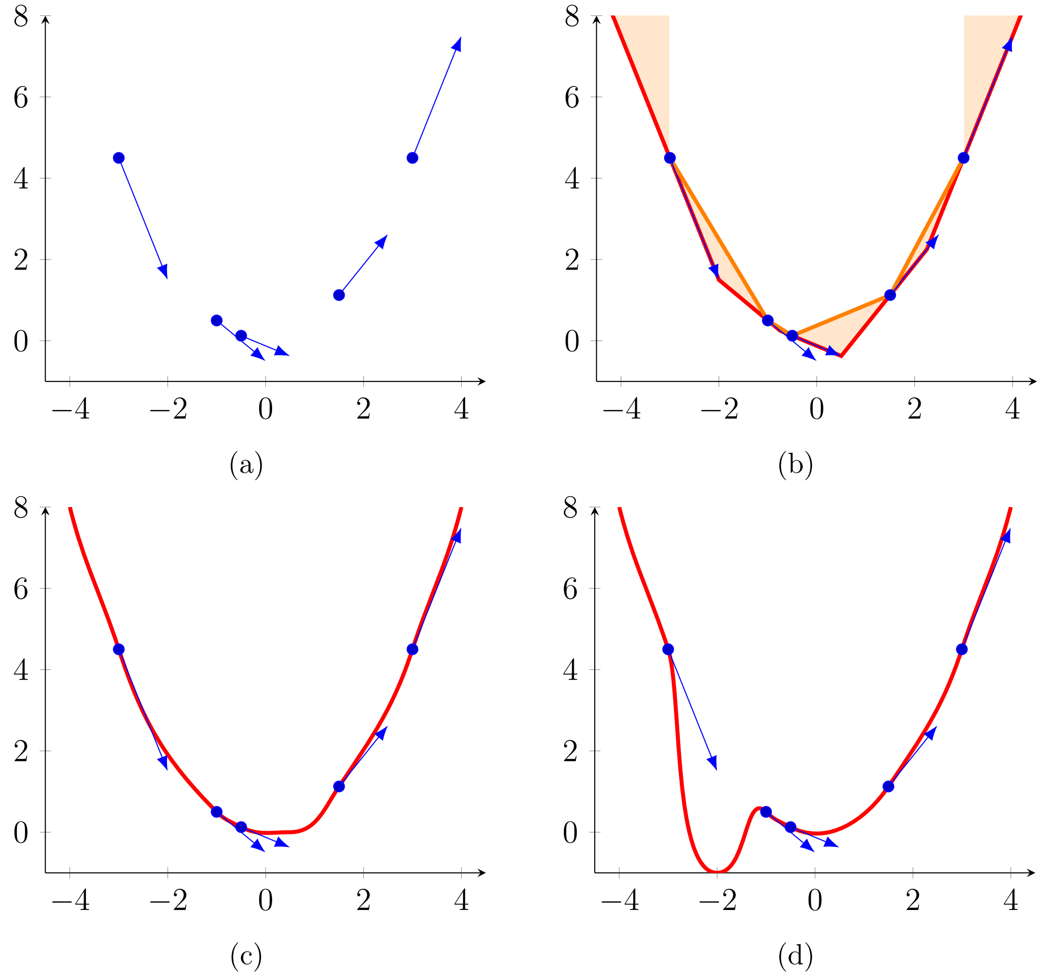

Figure 2.1: (a) Representation of the values and derivatives of a function $f : \mathbb{R} \to \mathbb{R}$ at a few points. (b) Upper and lower bounds on f (respectively orange and red lines) one can deduce from the knowledge of these values and derivatives if f is convex. Observe that it gives a reasonably tight approximation of f, its minimum and minimizer. (c) A non-convex function compatible with these values and derivatives. (d) Another non-convex function compatible with these values and derivatives. Observe that the minimum and minimizer are significantly different from 2.1c.

determining whether a point is a local minimum is already NP-difficult.[3] Therefore, we must lower our expectations again: if we try to solve an arbitrary non-convex problem, we cannot hope to find a local minimum; at best, we can find a *critical point*.

2.1.2 Critical points

Critical points are points at which "the derivatives of f satisfy the same properties as at a local minimizer".

Definition 2.2 : critical point

We say that an element x of $\mathbb{R}^d$ is

- a *first-order critical point* of f if $\nabla f(x) = 0$,
- a *second-order critical point* of f if $\nabla f(x) = 0$ and $\operatorname{Hess} f(x) \succeq 0$.

Of course, the first notion is well-defined only for differentiable functions f, and the second one only for twice-differentiable ones.

Remark

Local minimizers of f are necessarily second-order critical points, but the converse may not be true. For instance, the map $x \in \mathbb{R} \to x^3 \in \mathbb{R}$ has a second-order critical point at 0, but no local minimizer. However, if x is a second-order critical point such that $\operatorname{Hess} f(x) \succ 0$, then x is a local minimizer of f.

[3]Interested readers can refer to [Murty and Kabadi, 1987] for an example. In a nutshell, this article shows that it is NP-difficult to check whether an integer matrix $D \in \mathbb{Z}^{d\times d}$ is *copositive*, i.e. whether $x^T D x \geq 0$ for any $x \in (\mathbb{R}^+)^d$. And this is equivalent to checking whether 0 is a local minimum of

$$\begin{array}{rccc} f & : & \mathbb{R}^d & \to & \mathbb{R} \\ & & x & \to & \sum_{i,j=1}^d D_{ij} x_i^2 x_j^2. \end{array}$$

Example 2.3

Let us consider the map

$$\begin{array}{cccc} f: & \mathbb{R}^2 & \to & \mathbb{R} \\ & (x,y) & \to & \frac{x^4}{4} - \frac{x^3}{3} - x^2 + y^2. \end{array}$$

For any $(x,y) \in \mathbb{R}^2$,

$$\nabla f(x,y) = \begin{pmatrix} (x+1)x(x-2) \\ 2y \end{pmatrix}, \quad \mathrm{Hess} f(x,y) = \begin{pmatrix} 3x^2 - 2x - 2 & 0 \\ 0 & 2 \end{pmatrix}.$$

First-order critical points of f are the points (x,y) at which $\nabla f(x,y) = \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)$, i.e.

$$(-1,0), (0,0) \text{ and } (2,0).$$

The Hessian at these points is

$$\mathrm{Hess} f(-1,0) = \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix}, \qquad \mathrm{Hess} f(0,0) = \begin{pmatrix} -2 & 0 \\ 0 & 2 \end{pmatrix}$$
$$\text{and } \mathrm{Hess} f(2,0) = \begin{pmatrix} 6 & 0 \\ 0 & 2 \end{pmatrix}.$$

A diagonal matrix is semidefinite positive if and only if its diagonal coefficients are nonnegative. Consequently, $(-1,0)$ and $(2,0)$ are second-order critical points, but $(0,0)$ is not.
Actually, the Hessian at $(-1,0)$ and $(2,0)$ is *definite positive* (as the diagonal coefficients are strictly positive). Therefore, $(-1,0)$ and $(2,0)$ are local minimizers of f. More precisely, it can be shown that $(2,0)$ is a global minimizer while $(-1,0)$ is only a local one.

In the rest of the chapter, we will show that standard algorithms are able to find critical points of non-convex functions (first or second-order, depending on the assumptions). But the reader may wonder: why bother proving that a given non-convex algorithm always outputs a critical point of the objective function? What we really want are minimizers of f, not critical points! For us, the main reason is that knowing that an algorithm returns a critical point for sure is a first step towards analyzing its behavior. Indeed, it allows us to restrict our analysis of possible outputs to the set of critical points. In particular, if the objective function has few critical points, it already provides a lot of information on the output. Another motivation to have in mind is that, although it is not true that second-order critical points are always local minimizers (see the last remark), it is true for *generic* functions. This is explained in the following subsection.

2.1.3 When second-order critical points are local minimizers

In this subsection, we assume that f is C^2.

Definition 2.4

Let $x \in \mathbb{R}^d$ be a point. We say that the Hessian of f at x is *degenerate* if

$$\mathrm{Ker}\,(\mathrm{Hess}\, f(x)) \neq \{0\}.$$

Otherwise, it is called *non-degenerate*.

Definition 2.5

The map f is called a *Morse function* if, for any first-order critical point x of f, the Hessian of f at x is non-degenerate.

Proposition 2.6

If f is a Morse function, all its second-order critical points are local minimizers.

Proof. We assume that f is Morse.

Let $x \in \mathbb{R}^d$ be a second-order critical point of f. To show that it is a local minimizer, it suffices to show that $\mathrm{Hess}\, f(x)$ is positive definite.

From the definition of second-order criticality, we know that $\mathrm{Hess}\, f(x) \succeq 0$, i.e. all its eigenvalues are nonnegative. In addition, $\mathrm{Ker}\,(\mathrm{Hess}\, f(x)) = \{0\}$, as f is Morse, so $\mathrm{Hess}\, f(x)$ has no zero eigenvalue (otherwise, the corresponding eigenvectors would belong to the kernel). Therefore, all eigenvalues are strictly positive, which means that $\mathrm{Hess}\, f(x) \succ 0$ (see Proposition A.3). □

Theorem 2.7

For almost any $a \in \mathbb{R}^d$, the map

$$\begin{array}{rccc} f_a & : \mathbb{R}^d & \to & \mathbb{R} \\ & x & \to & f(x) + \langle a, x \rangle \end{array}$$

is a Morse function.

Proof. Let $a \in \mathbb{R}^d$. The first-order critical points of f_a are the points $x \in \mathbb{R}^d$ such that

$$\nabla f_a(x) = \nabla f(x) + a = 0,$$

that is,

$$a = -\nabla f(x).$$

Moreover, for every $x \in \mathbb{R}^d$, we have

$$\text{Hess}\, f_a(x) = \text{Hess}\, f(x),$$

so a first-order critical point x of f_a has a degenerate Hessian if and only if $\text{Hess}\, f(x)$ is degenerate, which is equivalent to $-\text{Hess}\, f(x)$ being degenerate.

Consider the map $F = -\nabla f : \mathbb{R}^d \to \mathbb{R}^d$. For any $a \in \mathbb{R}^d$, from the above, f_a is not Morse if and only if a is a *critical value* of F, i.e. there exists $x \in \mathbb{R}^d$ such that

$$a = -\nabla f(x) = F(x) \text{ and } -\text{Hess}\, f(x) = dF(x) \text{ is degenerate.}$$

By Sard's theorem, the set of critical values of F has Lebesgue measure zero in $\mathbb{R}^d$. In other words, the set of vectors a for which f_a is not Morse has Lebesgue measure zero. $\square$

This theorem shows, in some sense, that sufficiently generic objective functions are Morse.[4]

[4] This assertion can be formalized in different ways. For instance, it is possible to show that the set of Morse functions is a *Baire set* in $C^2(\mathbb{R}^d, \mathbb{R})$ for the most standard topology.

2.2 Finding first-order critical points

Assuming f is differentiable, the most basic optimization algorithm is gradient descent. It defines a sequence of iterates $(x_t)_{t\in\mathbb{N}}$ by

$$x_{t+1} = x_t - \alpha_t \nabla f(x_t), \quad \forall t \in \mathbb{N}.$$

Here, the parameters $\alpha_t > 0$ are called the *stepsizes*.

In this subsection, we are going to see the following results:

- under very weak hypotheses, x_t is an approximate first-order critical point for t large enough (Corollary 2.9);
- under slightly stricter (but still weak) hypotheses, x_t actually converges to a first-order critical point when $t \to +\infty$ (Theorem 2.11).

We first need a proposition about the decay of f along the gradient descent trajectory.

Proposition 2.8

We assume that the gradient of f is L-Lipschitz[a] for some $L > 0$: for any $x, y \in \mathbb{R}^d$,

$$||\nabla f(x) - \nabla f(y)||_2 \le L||x-y||_2.$$

We consider gradient descent with stepsize $\alpha_t = \frac{1}{L}$.[b]
Then, for each $t \in \mathbb{N}$,

$$f(x_{t+1}) \le f(x_t) - \frac{1}{2L}||\nabla f(x_t)||_2^2.$$

[a]This assumption is often called *L-smoothness*.

[b]Other choices are possible. In practice, L is usually unknown and the stepsizes are chosen using linesearch.

Proof. For all $x, h \in \mathbb{R}^d$,

$$\begin{aligned} f(x+h) &= f(x) + \int_0^1 \langle \nabla f(x+th), h \rangle \, dt \\ &= f(x) + \int_0^1 \langle \nabla f(x) + (\nabla f(x+th) - \nabla f(x)), h \rangle \, dt \end{aligned}$$

$$
\begin{aligned}
&= f(x) + \langle \nabla f(x), h \rangle + \int_0^1 \langle \nabla f(x+th) - \nabla f(x), h \rangle \, dt \\
&\leq f(x) + \langle \nabla f(x), h \rangle + \int_0^1 ||\nabla f(x+th) - \nabla f(x)||_2 \, ||h||_2 dt \\
&\qquad\qquad\qquad\qquad \text{(by triangular inequality)} \\
&\leq f(x) + \langle \nabla f(x), h \rangle + L \int_0^1 ||h||_2^2 t dt \\
&\qquad\qquad\qquad\qquad \text{(as } \nabla f \text{ is } L\text{-Lipschitz)} \\
&= f(x) + \langle \nabla f(x), h \rangle + \frac{L}{2} ||h||_2^2.
\end{aligned}
$$

We apply this inequality to $x = x_t$ and $h = -\frac{1}{L}\nabla f(x_t)$:

$$\forall t \in \mathbb{N}, \quad f(x_{t+1}) \leq f(x_t) - \frac{1}{2L} ||\nabla f(x_t)||_2^2.$$

□

This property implies that the gradient descent iterates are "asymptotically first-order critical", in the sense that ∇f goes to zero along the sequence.

Corollary 2.9

Under the same assumptions as Proposition 2.8, and recalling that we assume the existence of at least one minimizer of f,

$$||\nabla f(x_t)||_2 \overset{t\to+\infty}{\longrightarrow} 0.$$

Proof. For any $T \in \mathbb{N}$, from Proposition 2.8,

$$
\begin{aligned}
\frac{1}{2L} \sum_{t=0}^{T} ||\nabla f(x_t)||_2^2 &\leq \sum_{t=0}^{T} [f(x_t) - f(x_{t+1})] \\
&= f(x_0) - f(x_{T+1}) \\
&\leq f(x_0) - \min f.
\end{aligned}
$$

Consequently, the sum $\sum_{t\in\mathbb{N}} ||\nabla f(x_t)||_2^2$ is convergent, so its terms go to zero. □

Refining the argument, we can moreover give an a priori estimate for the convergence rate of $||\nabla f(x_t)||_2$ towards zero.

Corollary 2.10

We keep the same assumptions as in Corollary 2.9
For any T, if we set $\tilde{x}_T = \text{argmin}\,\{||\nabla f(x)||_2, x \in \{x_0, \dots, x_T\}\}$, this point satisfies

$$||\nabla f(\tilde{x}_T)||_2 \leq \sqrt{\frac{2L(f(x_0) - \min f)}{T+1}}.$$

Proof. We have seen in the proof of Corollary 2.9 that, for any T,

$$\sum_{t=0}^{T} ||\nabla f(x_t)||_2^2 \leq 2L(f(x_0) - \min f).$$

Since $||\nabla f(\tilde{x}_T)||_2 \leq ||\nabla f(x_t)||_2$ for any $t \leq T$,

$$(T+1)||\nabla f(\tilde{x}_T)||_2^2 \leq 2L(f(x_0) - \min f),$$

which implies

$$||\nabla f(\tilde{x}_T)||_2 \leq \sqrt{\frac{2L(f(x_0) - \min f)}{T+1}}.$$

□

Another way of stating the above result is that, for fixed Lipschitz constant L and gap $(f(x_0) - \min(f))$, gradient descent needs at most $O\left(\frac{1}{\epsilon^2}\right)$ iterations to find an ϵ-approximate first-order critical point. Let us mention that this convergence rate is optimal: for any algorithm and any ϵ, there is at least one function with the given Lipschitz constant and gap such that the algorithm needs to query at least $O\left(\frac{1}{\epsilon^2}\right)$ values of the function or its derivatives to find an ϵ-approximate first-order critical point [Carmon, Duchi, Hinder, and Sidford, 2020].

We have seen that gradient descent iterates are asymptotically first-order critical. At this stage, a natural question is: do the iterates actually converge towards a first-order critical point? This may not be true for non-coercive[5] maps: the iterates may diverge to infinity. For coercive functions, it is also not always true. In particular, it is possible[6] that the iterates cycle around

[5]A function f is *coercive* if $f(x) \to +\infty$ when $||x||_2 \to +\infty$.

[6]in theory: in practice, this is very rare

a large set of critical points. Therefore, we need additional assumptions to obtain rigorous convergence guarantees.

Theorem 2.11 : convergence of gradient descent iterates

We still assume that the gradient of f is L-Lipschitz, for some $L > 0$. We also assume that f is coercive. In addition, we make either of the following two assumptions:

- the set of first-order critical points of f is discrete;[a]
- f is analytic.[b]

We still consider gradient descent with stepsize $\frac{1}{L}$.
The sequence of iterates $(x_t)_{t\in\mathbb{N}}$ converges towards a first-order critical point of f.

[a] A set E is discrete if, for all $x \in E$, there exists $\epsilon > 0$ such that $E \cap B(x, \epsilon) = \{x\}$.
[b] A function is analytic if it is C^∞ and agrees with its Taylor series in a neighborhood of every point.

Proof. We only prove the result for the first assumption. For the second one, the reader is referred to [Absil, Mahony, and Andrews, 2005, Thm 3.2].

From Proposition 2.8, the iterates satisfy

$$f(x_t) \le f(x_0), \quad \forall t \in \mathbb{N}.$$

We define $A = \left\{x \in \mathbb{R}^d, f(x) \le f(x_0)\right\}$. It is a closed and bounded set, which contains all points x_t. Consequently, $(x_t)_{t\in\mathbb{N}}$ is bounded, hence has at least one accumulation point.

From Corollary 2.9, and because ∇f is continuous, all accumulation points are first-order critical.

Let $x_{c,1}, \dots, x_{c,S}$ be the first-order critical points in A. There is only a finite number of them because the set of first-order critical points is discrete, hence has finite intersection with every bounded set.

Let us fix

$$\epsilon < \frac{1}{3} \min_{s \neq s'} ||x_{c,s} - x_{c,s'}||_2,$$

$$\mu \overset{def}{=} \min_{x \in A \setminus \left(\bigcup_{s \le S} B(x_{c,s}, \epsilon)\right)} ||\nabla f(x)||_2.$$

We observe that $\mu > 0$; otherwise, f would have a first-order critical point in A, different from all $x_{c,s}$, contradicting the definition of $x_{c,1}, \dots, x_{c,S}$.

From Corollary 2.9, for t large enough, $||\nabla f(x_t)||_2 < \mu$, hence

$$x_t \in \bigcup_{s \leq S} B(x_{c,s}, \epsilon).$$

Also for t large enough, $||x_{t+1} - x_t||_2 = \frac{1}{L}||\nabla f(x_t)||_2 < \epsilon$. Because all balls $B(x_{c,s}, \epsilon)$ are at distance at least ϵ one from each other (from the definition of ϵ), it is impossible that

$$x_t \in B(x_{c,s}, \epsilon) \quad \text{and} \quad x_{t+1} \in B(x_{c,s'}, \epsilon) \text{ for } s' \neq s.$$

Therefore, for t large enough, all iterates belong to the *same* ball $B(x_{c,s}, \epsilon)$. Let s be the index of this ball. All accumulation points of $(x_t)_{t\in\mathbb{N}}$ are first-order critical and the only first-order critical point in $\overline{B(x_{c,s}, \epsilon)}$ is $x_{c,s}$, so $(x_t)_{t\in\mathbb{N}}$ is a bounded sequence with a single accumulation point, which is $x_{c,s}$. Therefore,

$$x_t \overset{t\to+\infty}{\longrightarrow} x_{c,s}.$$

$\square$

Remark

If the gradient of f is not Lipschitz, but simply continuous, the theorem is still true, except for the fact that the stepsize of gradient descent cannot be chosen as $\frac{1}{L}$ (the Lipschitz constant L is not defined): it must be chosen by linesearch.

2.3 Finding second-order critical points

In this subsection, we assume that f is C^2 over $\mathbb{R}^d$.

2.3.1 Second-order algorithms

Since the definition of second-order critical points involves the Hessian of f, it seems reasonable that using Hess$f$ during the optimization procedure might help to find a second-order critical point. Such algorithms, which use second-order derivatives, are called *second-order algorithms*. In this paragraph, we present a simplified version of one of them, called *Trust-Region*.

The starting point of this algorithm is that for any $x \in \mathbb{R}^d$,

$$f(x+h) = f(x) + \langle h, \nabla f(x) \rangle + \frac{1}{2} \langle h, \mathrm{Hess} f(x) h \rangle + o(||h||^2). \tag{2.1}$$

In view of this equation, one might be tempted to define the iterates $(x_t)_{t \in \mathbb{N}}$ using a recurrence relation $x_{t+1} = x_t + h_t$, where

$$h_t \in \mathrm{argmin}_{h \in \mathbb{R}^d} \left(f(x_t) + \langle h, \nabla f(x_t) \rangle + \frac{1}{2} \langle h, \mathrm{Hess} f(x_t) h \rangle \right).$$

Unfortunately, this definition makes no sense: when Hess$f$ is not semidefinite positive, the above function is not lower bounded, hence has no minimizer. Even if a minimizer exists, it is only a sensible choice for x_{t+1} if it belongs to the neighborhood of x_t on which Approximation (2.1) is valid. Therefore, it is best to refine the previous definition as

$$x_{t+1} = x_t + h_t, \tag{2.2a}$$

$$h_t \in \underset{||h|| \le R_t}{\mathrm{argmin}} \left(f(x_t) + \langle h, \nabla f(x_t) \rangle + \frac{1}{2} \langle h, \mathrm{Hess}\, f(x_t) h \rangle \right). \tag{2.2b}$$

In this definition, R_t is a positive number, the *trust radius*, which serves as an estimation of the size of the region over which Equation (2.1) provides a good approximation of f.

Theorem 2.12 : convergence of the trust-region method

Let $\epsilon > 0$ be fixed.
We assume that f has at least one minimizer x_* and Hess f is L_2-Lipschitz for some $L_2 > 0$:

$$\forall x, y, h \in \mathbb{R}^n, \quad ||(\mathrm{Hess}\, f(x) - \mathrm{Hess}\, f(y))h||_2 \le L_2 ||x-y||_2 \, ||h||_2.$$

Let $(x_t)_{t \in \mathbb{N}}$ be defined as in Equations (2.2a) and (2.2b), with $R_t = \frac{\sqrt{\epsilon}}{2L_2}$ for any t.
For any $x_0 \in \mathbb{R}^n$, the algorithm finds an ϵ-approximate second-order critical point in at most $O\left(\frac{L_2^2(f(x_0)-f(x_*))}{\epsilon^{3/2}}\right)$ iterations. More precisely, there exists

$$t \le c \frac{L_2^2(f(x_0) - f(x_*))}{\epsilon^{3/2}}$$

(for some explicit constant $c > 0$) such that

$$||\nabla f(x_t)||_2 \leq \frac{\epsilon}{L_2} \quad \text{and} \quad \mathrm{Hess} f(x_t) + \sqrt{\epsilon} I_d \succeq 0.$$

Sketch of proof, based on [Ye, 2015]. We admit the following statement: for each t, there exists $\sigma_t \geq 0$ such that

$$\left(\mathrm{Hess} f(x_t) + \sigma_t I_d\right) h_t = -\nabla f(x_t) \quad \text{and} \quad \mathrm{Hess} f(x_t) + \sigma_t I_d \succeq 0.$$

In addition, if $\sigma_t > 0$, then $||h_t||_2 = R_t$.

We first show that there exists $t \leq \frac{24L_2^2(f(x_0)-f(x_*))}{\epsilon^{3/2}} + 1$ such that

$$\sigma_t \leq \frac{\sqrt{\epsilon}}{2}.$$

By contradiction, let us assume that it is not true. Because the Hessian is L_2-Lipschitz, for all $t \leq \frac{24L_2^2(f(x_0)-f(x_*))}{\epsilon^{3/2}} + 1$,

$$\begin{aligned}
f(x_{t+1}) &= f(x_t + h_t) \\
&\leq f(x_t) + \langle h_t, \nabla f(x_t)\rangle + \frac{1}{2}\langle h_t, \mathrm{Hess}\, f(x_t) h_t\rangle + \frac{L_2}{6}||h_t||_2^3 \\
&= f(x_t) - \langle h_t, \mathrm{Hess} f(x_t) h_t + \sigma_t h_t\rangle + \frac{1}{2}\langle h_t, \mathrm{Hess}\, f(x_t) h_t\rangle + \frac{L_2}{6}||h_t||_2^3 \\
&= f(x_t) - \frac{1}{2}\langle h_t, \left(\mathrm{Hess} f(x_t) + \sigma_t I_d\right) h_t\rangle - \frac{\sigma_t}{2} R_t^2 + \frac{L_2}{6} R_t^3 \\
&\qquad\qquad (||h_t||_2 = R_t \text{ since } \sigma_t > 0) \\
&\leq f(x_t) - \frac{\sigma_t}{2} R_t^2 + \frac{L_2}{6} R_t^3 \\
&\qquad\qquad (\text{as } \mathrm{Hess} f(x_t) + \sigma_t I_d \succeq 0), \\
&< f(x_t) - \frac{\sqrt{\epsilon}}{4} R_t^2 + \frac{L_2}{6} R_t^3 \\
&= f(x_t) - \frac{\epsilon^{3/2}}{24 L_2^2}.
\end{aligned}$$

Therefore, for any $t \leq \frac{24L_2^2(f(x_0)-f(x_*))}{\epsilon^{3/2}} + 1$,

$$f(x_0) - f(x_*) \geq f(x_0) - f(x_{t+1})$$

$$> \frac{\epsilon^{3/2}}{24L_2^2} t,$$

which cannot be true for $t = \left\lceil \frac{24L_2^2(f(x_0)-f(x_*))}{\epsilon^{3/2}} \right\rceil$. This contradiction concludes the first part of the proof.

Let $t \leq \frac{24L_2^2(f(x_0)-f(x_*))}{\epsilon^{3/2}} + 1$ be such that $\sigma_t \leq \frac{\sqrt{\epsilon}}{2}$. We show that x_{t+1} is an approximate second-order critical point. First, it holds

$$\begin{aligned}
\text{Hess}f(x_{t+1}) &= \text{Hess}f(x_t + h_t) \\
&\succeq \text{Hess}f(x_t) - L_2||h_t||I_d \\
&= \text{Hess}f(x_t) + \sigma_t I_d - \sigma_t I_d - L_2||h_t||I_d \\
&\succeq -\sigma_t I_d - L_2||h_t||I_d \\
&\succeq -\sqrt{\epsilon} I_d.
\end{aligned}$$

Second, $||\nabla f(x_{t+1}) - \nabla f(x_t) - \text{Hess}f(x_t)h_t||_2 \leq \frac{L_2}{2}||h_t||_2^2$, hence

$$\begin{aligned}
||\nabla f(x_{t+1})||_2 &\leq ||\nabla f(x_t) + \text{Hess}f(x_t)h_t||_2 + \frac{L_2}{2}||h_t||_2^2 \\
&= || - \sigma_t h_t||_2 + \frac{L_2}{2}||h_t||_2^2 \\
&\leq \frac{\sqrt{\epsilon}}{2} R_t + \frac{L_2}{2} R_t^2 \\
&= \frac{3\epsilon}{8L_2}.
\end{aligned}$$

□

2.3.2 Gradient descent, again

We have seen in Subsection 2.2 that, under mild assumptions on f, gradient descent, starting at any point $x_0 \in \mathbb{R}^d$, allows to find an approximate first-order critical point. The same is not true for second-order critical points. For instance, if x_0 is a first-order critical point of f, but not a second-order critical point, then

$$x_0 = x_1 = x_2 = \dots,$$

because $\nabla f(x_0) = 0$, hence gradient descent stays stuck at x_0 and never gets close to a second-order critical point.

Nevertheless, this phenomenon is very rare: for “general” initializations, it does not happen, and gradient descent converges to a second-order critical point.

Theorem 2.13 : [Lee, Simchowitz, Jordan, and Recht, 2016], [Panageas and Piliouras, 2017]

Let f be a C^2 function which satisfies the same assumptions as in Theorem 2.11: the gradient of f is L-Lipschitz, for some $L > 0$; f is coercive and at least one of the following two assumptions holds:

- the set of first-order critical points of f is discrete;
- f is analytic.

We consider gradient descent with constant stepsize $\alpha \in]0; \frac{1}{L}[$.
For almost any x_0,[a] $(x_t)_{t\in\mathbb{N}}$ converges to a second-order critical point.

[a]that is, for all x_0 outside a zero-Lebesgue measure set

Remark

The theorem is still true even if ∇f is not Lipschitz, if we replace “with constant stepsize $\alpha \in]0; \frac{1}{L}[$” with “for a small enough stepsize α, possibly depending on x_0”. However, it is open whether the same holds true when the stepsize is chosen using the most standard linesearch procedure; see [Muşat and Boumal, 2025].

Intuition of proof. We consider the setting where the first hypothesis holds true: first-order critical points form a discrete set.

Theorem 2.11 shows that the gradient descent iterates $(x_t)_{t\in\mathbb{N}}$ converge to a first-order critical point whatever x_0.

We show that, if x_{crit} is a first-order but not a second-order critical point of f, then $(x_t)_{t\in\mathbb{N}}$ does not converge to x_{crit}, for almost any x_0. Since there are only countably many first-order critical points (a discrete set is countable), this is enough to establish the result: a countable union of zero Lebesgue measure sets has Lebesgue measure zero.

We consider an arbitrary first-order critical point which is not second-order critical. Up to translation, we can assume that it is 0.

We make the (very) simplifying hypothesis that f is quadratic in a ball centered at 0, with radius r_0:

$$\forall x \in B(0, r_0), \quad f(x) = \frac{1}{2} \langle x, Mx \rangle + \langle x, b \rangle ,$$

for some $n \times n$ symmetric matrix M.

For any $x \in B(0, r_0)$, $\nabla f(x) = Mx + b$. Since 0 is a first-order critical point, we necessarily have $b = 0$. In addition, $\operatorname{Hess} f(x) = M$ for any $x \in B(0, r_0)$. The assumption that 0 is not a second-order critical point is then equivalent to the fact that $M \not\succeq 0$.

The matrix M can be diagonalized in an orthonormal basis:

$$M = U^T \begin{pmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_d \end{pmatrix} U,$$

with $\lambda_1 \geq \dots \geq \lambda_d$ the eigenvalues of M and U an orthonomal matrix. Up to a change of coordinates, we can assume $U = \mathrm{Id}$. Since $M \not\succeq 0$, at least the smallest eigenvalue of M is negative: $\lambda_d < 0$.

If the sequence $(x_t)_{t \in \mathbb{N}}$ of gradient descent iterates converges to $x_{crit} = 0$, then x_t belongs to $B(0, r_0)$ for any t large enough, in which case

$$\begin{aligned} x_{t+1} &= x_t - \alpha \nabla f(x_t) \\ &= x_t - \alpha M x_t \\ &= \begin{pmatrix} (1-\alpha\lambda_1) x_{t,1} \\ \vdots \\ (1-\alpha\lambda_d) x_{t,d} \end{pmatrix}. \end{aligned}$$

We fix t_0 such that this relation holds for any $t \geq t_0$. Then, for any $s \in \mathbb{N}$,

$$x_{t_0+s} = \begin{pmatrix} (1-\alpha\lambda_1)^s x_{t_0,1} \\ \vdots \\ (1-\alpha\lambda_d)^s x_{t_0,d} \end{pmatrix}.$$

If the sequence converges to 0, all the coordinates of x_{t_0+s} must go to 0 when s goes to $+\infty$ (for any fixed t), which means that

$$\forall k \in \{1, \dots, d\}, \quad (1 - \alpha\lambda_k)^s x_{t_0,k} \overset{s \to +\infty}{\to} 0. \tag{2.3}$$

We have said that $\lambda_d < 0$, hence $1 < 1 - \alpha\lambda_d$ and $(1 - \alpha\lambda_d)^s \not\to 0$ when $s \to +\infty$. In order for Property (2.3) to hold, we must therefore have

$$x_{t_0,d} = 0.$$

To summarize, we have shown that, if $(x_t)_{t\in\mathbb{N}}$ converges to 0, then, for some t_0,

$$x_{t_0} \in \mathcal{E} \overset{def}{=} \{z \in B(0, r_0) \text{ such that } z_d = 0\}.$$

As a consequence,

$$x_0 \in (\mathrm{Id} - \alpha\nabla f)^{-t_0}(\mathcal{E}).$$

(For any map $g : \mathbb{R}^n \to \mathbb{R}^n$, we define $g^{-t_0}(\mathcal{E})$ as the set of points x such that $g^{t_0}(x) = g \circ \overset{t_0 \text{ times}}{\dots} \circ g(x) \in \mathcal{E}$.) Therefore, the set of initial points x_0 for which gradient descent iterates may converge to 0 is included in

$$\bigcup_{t\in\mathbb{N}} (\mathrm{Id} - \alpha\nabla f)^{-t}(\mathcal{E}).$$

The set $\mathcal{E}$ has zero Lebesgue measure and one can check that $\mathrm{Id} - \alpha\nabla f$ is a diffeomorphism, hence $(\mathrm{Id} - \alpha\nabla f)^{-t}(\mathcal{E})$ has zero Lebesgue measure for any $t \in \mathbb{N}$, and the set of “problematic” initial points also has zero Lebesgue measure. $\square$

Chapter 3

Convexification

What you should know / be able to do after this chapter

- Understand the general principle of convexification, and what "tightness" means.
- Be able to suggest convex relaxations of non-convex problems, based notably on the « convex hull » reasoning which provides intuition in the cases of compressed sensing and low-rank recovery.
- Using a 2-dimensional picture, explain why (Basis Pursuit) can be expected to be a tight relaxation of compressed sensing.
- Know the definition of « restricted isometry ».
- Know the proof technique for establishing tightness guarantees which relies on restricted isometry (in particular, know the statements of Theorems 3.4 and **??**).
- Know that restricted isometry holds true for the simplest cases of random linear operators.
- Explain the limitations of this technique: restricted isometry does *not* hold for some more "structured" operators.
- Understand the proof scheme from Figure **??** and be able to apply it in other (simpler) settings.

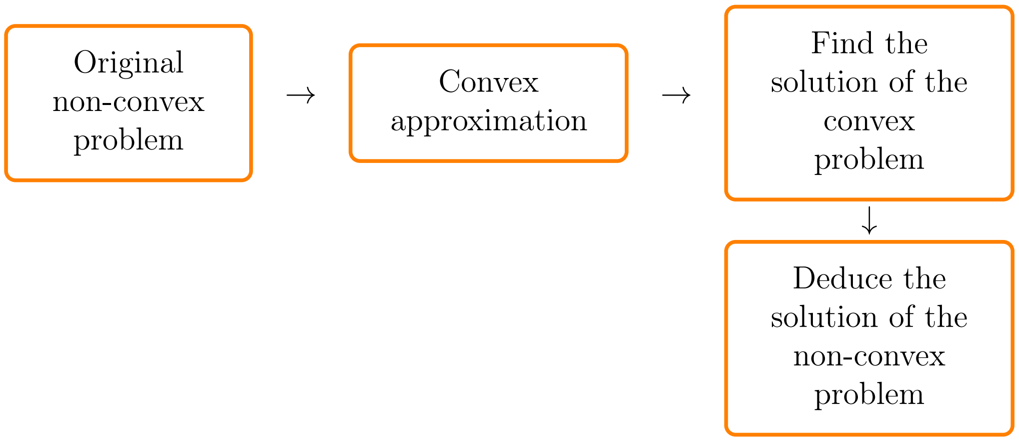
Original non-convex problem
Convex approximation
Find the solution of the convex problem
Deduce the solution of the non-convex problem

Figure 3.1: Principle of convexified algorithms, when relaxation is tight.

- Understand (i.e. be able to do it again alone, with minimal help) the derivation of the dual problem of TV minimization.

A possible strategy to overcome the difficulty of solving a non-convex inverse problem is to approximate the non-convex problem with a convex one. This convex approximation is called a *convex relaxation*. Since numerically solving a convex problem is in general doable, we can usually solve the approximation. At first sight, there is no reason why solving this approximation should provide useful information on the non-convex problem itself. But surprisingly, it turns out that, in many situations, the convex approximation has the same solution as the original non-convex problem! One then says that relaxation is *tight*. When this happens, it yields a convenient method for solving the non-convex problem. This general scheme is depicted on Figure 3.1.

3.1 The basis: compressed sensing

3.1.1 Convexification: principle

The model example for this chapter, which serves as a basis for other problems, is compressed sensing.

$$\boxed{\begin{aligned} &\text{recover } x \in \mathbb{R}^d \\ &\text{such that } Ax = y, \\ &\text{and } \|x\|_0 \leq k. \end{aligned}} \tag{CS}$$

When the problem has a unique solution, it is the vector with minimal ℓ^0-norm among all vectors x such that $Ax = y$. This allows to reformulate the problem as

$$\boxed{\begin{aligned} &\text{minimize } \|x\|_0 \\ &\text{for } x \in \mathbb{R}^d \\ &\text{such that } Ax = y. \end{aligned}} \tag{ℓ^0-min}$$

The set $\{x \in \mathbb{R}^d, Ax = y\}$ is convex. The non-convex part of the problem is the objective function $\|.\|_0$. To make the problem convex, we replace the ℓ^0-norm with the ℓ^1-norm:

$$\|x\|_1 = \sum_{i=1}^{d} |x_i|,$$

which leads to the following *convex* problem:

$$\boxed{\begin{aligned} &\text{minimize } \|x\|_1 \\ &\text{for } x \in \mathbb{R}^d \\ &\text{such that } Ax = y. \end{aligned}} \tag{Basis Pursuit}$$

3.1.2 Intuition

An intuitive reason for using the ℓ^1-norm as a convex approximation of the ℓ^0-norm is that the unit ℓ^1-ball is the smallest convex set which contains the "maximally sparse" vectors of norm 1.

Proposition 3.1 : ℓ^1-ball as a convex hull

Let $\mathcal{S}$ be the set of vectors with exactly one non-zero coordinate, equal to -1 or 1.
The unit ℓ^1-ball $\{x \in \mathbb{R}^d, ||x||_1 \leq 1\}$ is the convex hull of $\mathcal{S}$.

Proof. This proposition is a consequence of Proposition 3.2. Indeed, the unit ℓ^1-ball is a closed compact and convex subset of $\mathbb{R}^d$. It is therefore the convex hull of its extremal points (from the Krein-Milman theorem, see for instance [Barvinok, 2002, Chapter II, thm 3.3]), that it is the convex hull of $\mathcal{S}$. □

The next proposition states a stronger, but similar, result, which is crucial in explaining the success of (Basis Pursuit) (i.e. why it is oftentimes a *tight* convex relaxation).

Proposition 3.2 : extremal points of the ℓ^1-ball

The extremal points[a] of the unit ℓ^1-ball $\{x \in \mathbb{R}^d, ||x||_1 \leq 1\}$ are the vectors with exactly one non-zero coordinate, equal to -1 or 1.

[a] An *extremal point* of a convex set C is a point y which cannot be written as

$$y = (1-\theta)z_1 + \theta z_2$$

for $z_1, z_2 \in C$ different from y and $\theta \in [0;1]$.

Proof. Let $\mathcal{S}$ be the set of vectors with exactly one non-zero coordinate, equal to -1 or 1. Let B_{ℓ^1} be the unit ℓ^1-ball.

First, we show that the elements of $\mathcal{S}$ are extremal points of B_{ℓ^1}. Let $y \in \mathcal{S}$ be fixed. Let i be its unique non-zero coordinate. Let us assume for simplicity that $y_i = 1$ (the reasoning can be adapted for $y_i = -1$). Let $z_1, z_2 \in B_{\ell^1}, \theta \in [0;1]$ be such that

$$y = (1-\theta)z_1 + \theta z_2.$$

We must show that $z_1 = y$ or $z_2 = y$. If $\theta = 0$, then $z_1 = y$, and if $\theta = 1$, then $z_2 = y$, so we can assume $\theta \neq 0, 1$.

We have

$$1 = y_i = (1-\theta)(z_1)_i + \theta(z_2)_i.$$

Observe that $(z_1)_i \leq |(z_1)_i| \leq ||z_1||_1 \leq 1$ and, similarly, $(z_2)_i \leq 1$. These two inequalities must be equalities, otherwise $1 = (1-\theta)(z_1)_i + \theta(z_2)_i < (1-\theta)+\theta = 1$.

Now that we know that $(z_1)_i = 1$, we can say that

$$\sum_{j\neq i} |(z_1)_j| = ||z_1||_1 - |(z_1)_i| = ||z_1||_1 - 1 \leq 0,$$

hence $(z_1)_j = 0$ for all $j \neq i$. This shows $z_1 = y$, and concludes the proof that y is an extremal point of B_{ℓ^1}.

Conversely, we show that every extremal point of B_{ℓ^1} is in $\mathcal{S}$. Let $y \in B_{\ell^1}$ be extremal.

First, we note that $||y||_1 = 1$. Indeed, if $||y||_1 < 1$, we can write, for any vector $\epsilon \in \mathbb{R}^d \setminus \{0\}$,

$$y = \frac{1}{2}(y+\epsilon) + \frac{1}{2}(y-\epsilon).$$

When ϵ is close enough to zero, it holds $||y+\epsilon||_1, ||y-\epsilon||_1 \leq ||y||_1 + ||\epsilon||_1 \leq 1$, so $y+\epsilon, y-\epsilon$ belong to B_{ℓ^1} and are different from y, which contradicts the extremality of y.

Now, we show that y has only one non-zero coordinate. Let i be such that $y_i \neq 0$. By contradiction, we assume that not all other coordinates are zero. Let $\tilde{y}$ be the vector which is equal to y, except that the i-th coordinate y_i has been replaced with 0; it is not the null vector. Let $e \in \mathbb{R}^d$ be the vector such that

$$\begin{aligned} e_i &= \mathrm{sign}(y_i), \\ e_j &= 0, \quad \forall j \neq i. \end{aligned}$$

Then

$$y = |y_i| e + ||\tilde{y}||_1 \frac{\tilde{y}}{||\tilde{y}||_1} = |y_i| e + (1 - |y_i|)\frac{\tilde{y}}{||\tilde{y}||_1},$$

which contradicts the extremality. (The last equality is true because $||\tilde{y}||_1 = \sum_{j\neq i} |y_j| = ||y||_1 - |y_i| = 1 - |y_i|$.) □

Let us give an intuitive explanation, based on the previous proposition, of why we can expect (Basis Pursuit) to be a *tight* relaxation of Problem (CS), at least in some situations.

If the vector x_* we are trying to recover through Problem (CS) is sparse, then it is a linear combination of a small number of "maximally sparse" non-zero vectors. From Proposition 3.2, it is therefore a linear combination of a

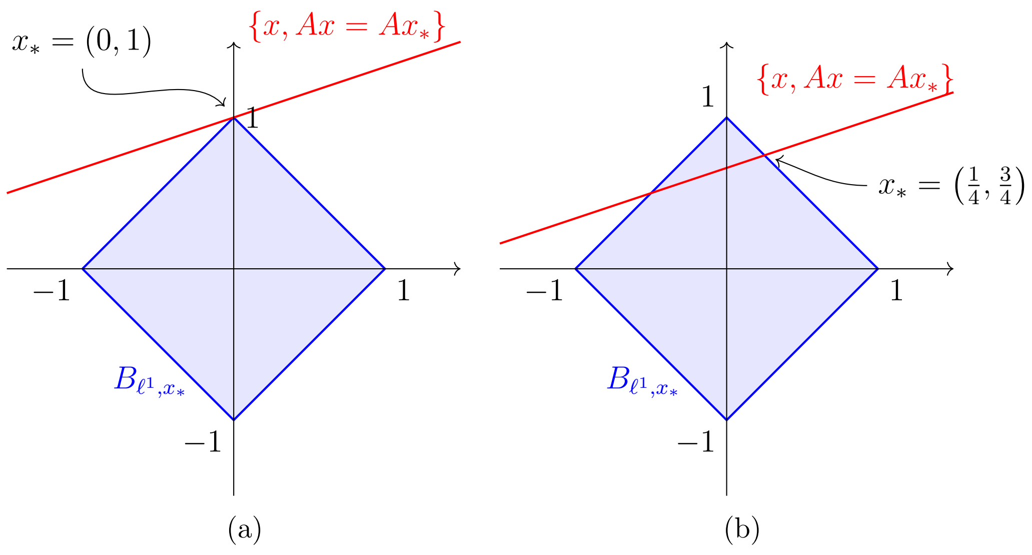

Figure 3.2: Representation of B_{ℓ^1,x_*} and $\{x \in \mathbb{R}^2, Ax = Ax_*\}$ for $A = \begin{pmatrix} 1 & -3 \end{pmatrix}$ in two situations: (a) when $x_* = (0,1)$ is sparse ; (b) when $x_* = \left(\frac{1}{4}, \frac{3}{4}\right)$ is not sparse. Observe that $B_{\ell^1,x_*} \cap \{x, Ax = Ax_*\}$ is a singleton in the first case, but not in the second one.

small number of extremal points of the ℓ^1-ball. This can be geometrically interpreted as the fact that x_* belongs to a "corner" of the ℓ^1-ball

$$B_{\ell^1,x_*} \overset{def}{=} \{x \in \mathbb{R}^d, ||x||_1 \leq ||x_*||_1\}.$$

The convex approximation (Basis Pursuit) has a unique minimizer equal to x_* if and only if

$$\begin{aligned} &\nexists x \in \mathbb{R}^d \text{ such that } Ax = y = Ax_* \text{ and } ||x||_1 \leq ||x_*||_1 \\ \iff &\quad B_{\ell^1,x_*} \cap \{x \in \mathbb{R}^d, Ax = Ax_*\} = \{x_*\} \\ \iff &\quad B_{\ell^1,x_*} \cap (\{x_*\} + \mathrm{Ker}(A)) = \{x_*\}. \end{aligned}$$

And the intersection of B_{ℓ^1,x_*} and an affine space containing x_* has much more chances to be the singleton $\{x_*\}$ if x_* is in a "corner" of B_{ℓ^1,x_*} (very crudely, if x_* is in a "corner", then, in the neighborhood of x_*, B_{ℓ^1,x_*} occupies only a small fraction of the space; it is therefore easier not to intersect it when considering an affine space going through x_*). This is depicted on Figure 3.2.

3.1.3 Tightness guarantees under restricted isometry

The convex problem (Basis Pursuit) can be traced back to at least the 70's. Since then, many researchers have proposed conditions on x_* and A under which the relaxation is tight (that is, the solutions of (Basis Pursuit) and (CS) are the same). A major progress (due notably to Candès, Donoho, Romberg and Tao) on this subject was, around twenty years ago, the introduction of the so-called *Restricted Isometry Property*, which is a simple assumption on A under which it is possible to guarantee tightness without imposing stringent conditions on x_*.

Definition 3.3 : restricted isometry

Let $A \in \mathbb{R}^{m\times d}$ be a matrix. For any $k \in \{1,\dots,d\}$, we define the k-restricted isometry constant δ_k of A as the smallest real number such that

$$(1-\delta_k)||z||_2 \le ||Az||_2 \le (1+\delta_k)||z||_2$$

for all vectors $z \in \mathbb{R}^d$ with at most k non-zero coordinates.

Tightness of the convex relaxation (Basis Pursuit) under a restricted isometry condition is guaranteed by the following theorem.

Theorem 3.4

Let $A \in \mathbb{R}^{m\times d}$ be a matrix. For some $k \in \{1,\dots,d\}$, we assume that its $4k$-restricted isometry constant satisfies

$$\delta_{4k} < \frac{1}{4}. \tag{3.4}$$

For any $x_* \in \mathbb{R}^d$ with at most k non-zero coordinates, Problem (Basis Pursuit) with $y = Ax_*$ has a unique solution, which is x_*.

Under the same condition, it is moreover possible to prove a stability result for the convex relaxation: if y is "close" to Ax_*, then the solution of a slight modification of (Basis Pursuit) is "close" to x_*. The proof of Theorem 3.4 is the subject of an exercise, which follows [Candès, Romberg, and Tao, 2006].

Let us keep in mind that the restricted isometry property is a *sufficient* but not *necessary* condition for the correctness of the basis pursuit approach:

there are matrices A for which condition (3.4) does *not* hold and, nevertheless, Problems (CS) and (Basis Pursuit) have the same solution. However, it turns out that many natural matrices A satisfy the condition, hence Theorem 3.4 explains the success of the basis pursuit approximation in several interesting situations. The following theorem provides the simplest example of matrices with the restricted isometry property: matrices chosen at random according to a normal distribution (with high probability).

Theorem 3.5 : [Candès and Tao, 2005]

Let $c > 0$ be some explicit constant, whose value we will not give here. We assume that $A \in \mathbb{R}^{m\times d}$ is generated at random according to a normal distribution[a]. If

$$ck\log(d/k) \leq m,$$

Condition (3.4) holds with high probability.[b]

[a]that is, each coefficient of A is chosen independently at random according to a normal law $\mathcal{N}(0, 1/m)$.

[b]*With high probability* means that it holds with probability at least $1 - e^{-\alpha m}$ for some constant $\alpha > 0$.

This theorem, combined with Theorem 3.4, shows that convexification allows to recovery k-sparse vectors from $O(k\log(d/k))$ linear measurements. This is surprisingly few. Indeed, Problem (CS) is only interesting when the number of measurements is at least $O(k)$ (otherwise, the solution is not unique). At this threshold, solving this problem is a priori impossible with a polynomial time algorithm, but we see that it suffices to increase the number of measurements by a logarithmic factor so that polynomial time recovery becomes possible, through convexification.

Appendix A

Reminders on symmetric and Hermitian matrices

Let $d \in \mathbb{N}^*$ be fixed.

For any matrix $M \in \mathbb{C}^{d\times d}$, we denote M^* the transpose conjugate of M, that is the $d \times d$ matrix such that, for all $i, j \leq d$,

$$M^*_{ij} = \overline{M_{ji}}.$$

The notation "$\langle .,. \rangle$" stands for the standard dot product when applied to real vectors: for all $a, b \in \mathbb{R}^d$,

$$\langle a, b\rangle = \sum_{i=1}^{d} a_i b_i.$$

When applied to complex vectors, it denotes the standard Hermitian product: for all $a, b \in \mathbb{C}^d$,

$$\langle a, b\rangle = \sum_{i=1}^{d} \overline{a_i} b_i = a^* b.$$

Definition A.1 : Hermitian matrices

A matrix $M \in \mathbb{C}^{d\times d}$ is *Hermitian* if $M = M^*$, that is if, for all $i, j \leq d$,

$$M_{ij} = \overline{M_{ji}}.$$

Equivalently, M is Hermitian if and only if, for all $x, y \in \mathbb{C}^d$,

$$\langle Mx, y\rangle = \langle x, My\rangle .$$

Definition A.2 : semidefinite positive matrices

Let $\mathbb{K}$ be $\mathbb{R}$ or $\mathbb{C}$.
A matrix $M \in \mathbb{K}^{d\times d}$ is *semidefinite positive* if and only if it is symmetric (if $\mathbb{K} = \mathbb{R}$) or Hermitian (if $\mathbb{K} = \mathbb{C}$) and, for all $x \in \mathbb{K}^d$,

$$\langle x, Mx\rangle \in \mathbb{R}^+.$$

It is denoted "$M \succeq 0$".
It is *definite positive* if and only if it is semidefinite positive and, for all $x \in \mathbb{K}^d \setminus \{0\}$,

$$\langle x, Mx\rangle > 0.$$

It is denoted "$M \succ 0$".

Proposition A.3 : diagonalization of symmetric / Hermitian matrices

Let $\mathbb{K}$ be $\mathbb{R}$ or $\mathbb{C}$. Let $M \in \mathbb{K}^{d\times d}$ be a symmetric or Hermitian matrix. It can be diagonalized in an orthogonal basis, with real eigenvalues: there exist $\lambda_1, \dots, \lambda_d$ in $\mathbb{R}$ and $(z_1, \dots, z_d)$ an orthonormal basis of $\mathbb{K}^d$ such that

$$X = \begin{pmatrix} z_1 & \dots & z_d \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & & \vdots \\ \vdots & \ddots & \ddots & \\ & & & \lambda_d \end{pmatrix} \begin{pmatrix} z_1 & \dots & z_d \end{pmatrix}^* = \sum_{k=1}^{d} \lambda_k z_k z_k^*.$$

Matrix M is semidefinite positive if and only if $\lambda_k \geq 0$ for all $k \leq d$. It is definite positive if and only if $\lambda_k > 0$ for all $k \leq d$.

Bibliography

P.-A. Absil, R. Mahony, and B. Andrews. Convergence of the iterates of descent methods for analytic cost functions. *SIAM Journal on Optimization*, 16(2):531–547, 2005.

A. Barvinok. *A course in convexity*, volume 54. American Mathematical Society, 2002.

E. J. Candès and T. Tao. Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215, 2005.

E. J. Candès and M. B. Wakin. An introduction to compressive sampling. *IEEE signal processing magazine*, 25(2):21–30, 2008.

E. J. Candès, J. K. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.

Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford. Lower bounds for finding stationary points i. *Mathematical Programming*, 184(1-2):71–120, 2020.

A. Conca, D. Edidin, M. Hering, and C. Vinzant. Algebraic characterization of injectivity in phase retrieval. *Applied and Computational Harmonic Analysis*, 32(2):346–356, 2015.

J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht. Gradient descent converges to minimizers. In *Proceedings of the Conference on Computational Learning Theory*, 2016.

K. G. Murty and S. N. Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical Programming: Series A and B*, 39(2):117–129, 1987.

A.-A. Muşat and N. Boumal. Gradient descent avoids strict saddles with a simple line-search method too. *arXiv preprint arXiv:2507.13804*, 2025.

I. Panageas and G. Piliouras. Gradient descent only converges to minimizers: Non-isolated critical points and invariant regions. In *Innovations in Theoretical Computer Science*, 2017.

Y. Ye. Second order optimization algorithms i, 2015. http://web.stanford.edu/class/msande311/2017lecture13.pdf.