Non-convex inverse problems

Irène Waldspurger

waldspurger@ceremade.dauphine.fr

Initial version: January to March 2023
This version: January to February 2025

Contents

Acknowledgements

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

Chapter 1

Introduction

Apporter son ordinateur la prochaine fois, en ayant installé Convex.jl et SCS.jl pour les utilisateur/trices de Julia, CVXPY pour les utilisateur/trices de Python.
Dans le courant du cours, se renseigner sur la familiarité des élèves avec le compressed sensing.

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 class 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.
- Identify the main differences between convex and non-convex inverse problems.

- 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.

Suggérer un autre exemple. Autres exemples que je peux donner :

- à partir de la consommation électrique d'un ordinateur au cours du temps, déterminer au mieux quelles ont été les activités de l'utilisatrice ;
- (échographie) donner une image d'un organe humain en envoyant des ondes sonores à l'intérieur et en enregistrant les ondes réfléchies.

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 $y \in F$,

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

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

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

This class 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 class, 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*.

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

- 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 a constant $C > 0$ "not too large" (say $C \leq 10$) such that

$$\forall x_1, x_2 \in E \text{ such that } x_1 \neq 0,$$
$$\frac{||x_1 - x_2||_E}{||x_1||_E} \leq C \frac{||M(x_1) - M(x_2)||_F}{||M(x_1)||_F}. \quad (1.1)$$

  Here, $||.||_E$ and $||.||_F$ are norms on E and F.[3]

Pour l'exemple ci-dessous : attention, tous les problèmes inverses ne sont pas linéaires !

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 $A \in \mathbb{R}^{m\times d}$.

[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.

Under these assumptions, Problem (Inverse) rewrites as

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

Questions 1 et 3 de l'exercice 1.
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 class. For each newly encountered problem, we will try to give conditions under which 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 class, 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 which what? Trois critères.

- 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 class 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:

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

$$x \in C_S.$$

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.

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 class, 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.

Intuitively, 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, Inequality (1.2) makes it possible to compute upper and lower bounds on f, and hence obtain an approximation of its minimum. One can then query the values at other points to refine the approximation. This is illustrated on Figures 1.1a and 1.1b. But if the function is not convex, the knowledge of its values at a few points provides no information about the values at other points and, in particular, provides no information on its minimum. This is illustrated on Figures 1.1c and 1.1d. This is what makes non-convex optimization much more difficult than convex optimization.

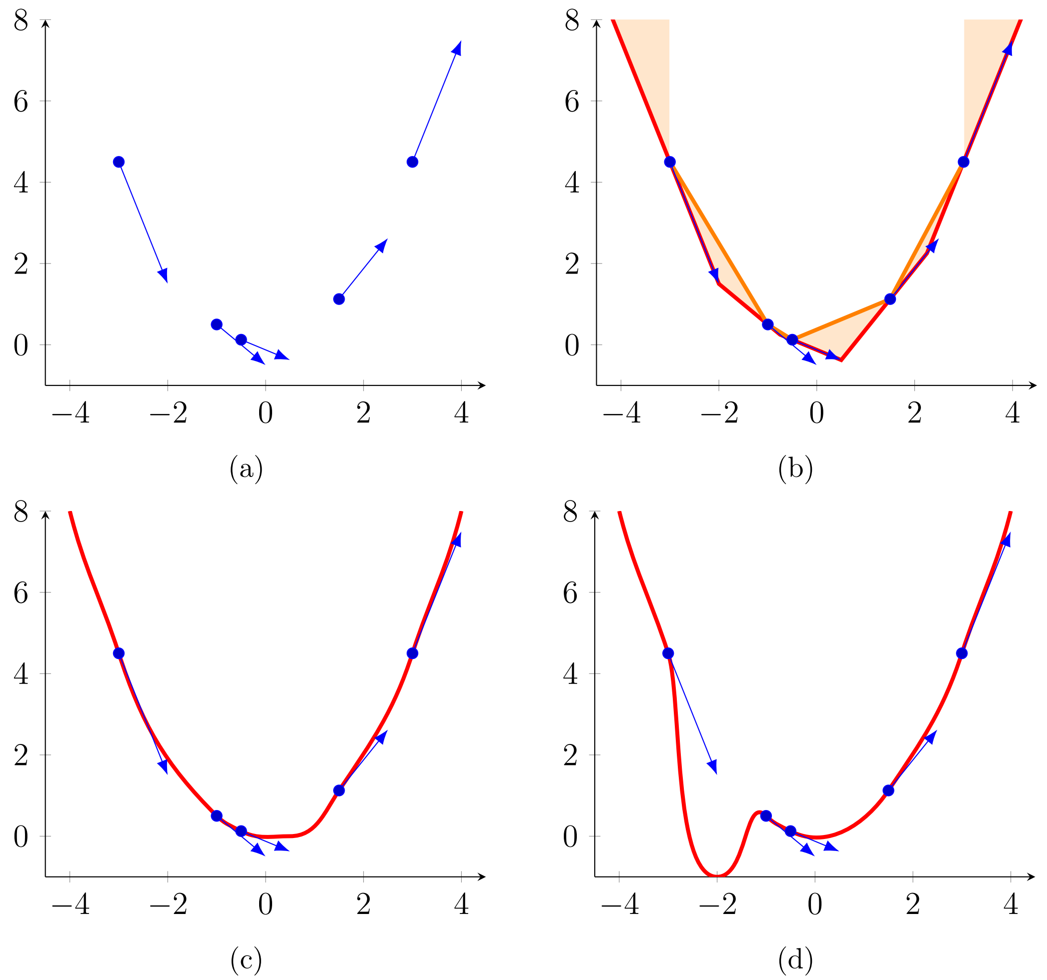

Figure 1.1: (a) Representation of the values and derivatives of a function $f : \mathbb{R} \to \mathbb{R}$ at a few points. (b) Upper and lower bounds on f (respectively orange and red lines) one can deduce from the knowledge of these values and derivatives if f is convex. Observe that it gives a reasonably 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 1.1c.

This difficulty is a fundamental property of non-convex problems: if we do not have good algorithms able to solve any non-convex problem, it is not because we have not discovered these good algorithms yet. It is because good algorithms do not exist.[4] As a consequence, in this class, 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.

Remplir le tableau ci-dessous. Je donne la première colonne.

	Convex problems	Non-convex problems
Definition	All functions and constraint sets in the optimization problem are convex.	The objective function or one of the constraint sets is not convex.
Theoretical aspects	The set of solutions is convex. Idem for *approximate* solutions. The solution is unique iff it is locally unique. Duality tools. ($\Rightarrow$ analysis less difficult.)	The set of solutions can be arbitrary. The solution can be non unique, but locally unique. Duality results are much weaker. ($\Rightarrow$ analysis more difficult.)
Algorithmic aspects	Many algorithms exist, for relatively large classes of convex problems. The goal is generally to solve the problem up to fixed arbitrary precision.	For most classes of problems, no algorithm exists, which is guaranteed to solve any problem in the class. When facing a family of problems, the goal is generally to solve as many instances as possible.

1.3 Non-convex inverse problems: examples

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

[4]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.

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,

$$||x||_0 \le k,$$

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

Note that, if $m \ge d$ 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.

Écrire ce problème sous la forme d'un problème d'optimisation (Opt). Convexe ou non-convexe ?

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

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

It is non-convex because the set $\{x, ||x||_0 \le k\}$ 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 "$||x||_0 \le k$" must be replaced with "$||\Phi x||_0 \le k$", 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 $\mathrm{Ker}(A)$ 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 $\mathrm{Ker}(A)$. And

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

which contradicts the assumption. □

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

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

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 $\operatorname{Ker}(\mathcal{L})$ 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 $r \ll \min(d_1, d_2)$).

Préparer (pour le cours suivant) un tableau de comparaison entre la reconstruction parcimonieuse et la reconstruction de matrices de rang faible. Je donne la première ligne comme exemple.

	Sparse recovery	Low rank matrix recovery
Unknown	vector	matrix
Measurements - convex part	linear	linear
Measurements - non-convex part	ℓ^0-norm $\leq k$	rank $\leq r$
Necessary condition for uniqueness	No element with ℓ^0-norm $\leq 2k$ in the kernel of the measurement operator	No element with rank $\leq 2r$ in the kernel of the measurement operator
Uniqueness for generic linear operators	if $m \geq 2k$ (ie twice the degrees of freedom)	if $m \geq 2r(d_1 + d_2 - 2r)$ (more or less twice the degrees of freedom)

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:

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

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".[5] 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:[6] 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}

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

[6] 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).

À votre avis, quelle doit être la taille de Ω pour qu'on ait unicité quel que soit y ?

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.

Ici, je rappelle la définition du produit hermitien, des matrices hermitiennes et des matrices semidéfinies positives.

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

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

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 $\mathrm{Range}(xx^*) = \mathrm{Vect}\{x\}$.

Conversely, if $X \succeq 0$ and $\mathrm{rank}(X) \leq 1$, 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 $\mathrm{rank}(X) \leq 1$, 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

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

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

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

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. $\square$

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.

Reformuler le problème de reconstruction de phase avec l'inconnue x en un problème avec l'inconnue X.

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

$$|L_j(x)|^2 = \langle v_j, x\rangle\,\overline{\langle v_j, x\rangle}$$

$$\begin{aligned} &= (v_j^* x)(x^* v_j) \\ &= v_j^* X v_j. \end{aligned}$$

Consequently, Problem (Phase retrieval) is equivalent to

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

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

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

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

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

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

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.