Non-convex inverse problems: projects

Irène Waldspurger
waldspurger@ceremade.dauphine.fr

Due date: April 12, 2023

Common instructions for all projects

Each project is about a different non-convex inverse problem. The goal is to propose and implement an algorithm, then to numerically study its behavior.

- You must propose a sensible (i.e. better than a random guess) algorithm. However, the project is about studying the behavior of an algorithm, not about designing a highly efficient one, so do not worry if the recovery rate is low.
- If you can't find a sensible algorithm by yourself, you can look for an algorithm in the scientific literature. In this case, cite the article where you have found the algorithm.
- For the question "can you tell why the algorithm failed?", you are expected to identify - if possible - a general failure cause, for instance:
  - my convex relaxation was not tight;
  - my iterative non-convex algorithm failed to converge;
  - my gradient descent got stuck at a non-optimal critical point ...
- You should provide your code. You should notably include the instructions you used to generate the pictures, so that I can reproduce them.
- The subjects may contain errors, or be miscalibrated in terms of difficulty. If you are stuck, please send me an email describing your problem; I will try to help.
- You can write in either French or English.

Sparse phase retrieval

A general (real) sparse phase retrieval problem is a mixture of phase retrieval and sparse recovery:

$$\begin{aligned} &\text{find } x \in \mathbb{R}^d, \\ &\text{such that } |\langle v_j, x\rangle| = y_j, \quad \forall j = 1, \dots, m, \\ &\text{and } ||x||_0 \leq k. \end{aligned} \tag{Sparse PR}$$

Here, $v_1, \dots, v_m \in \mathbb{R}^d$ are known *measurement vectors*; $y_1, \dots, y_m$ are the given measurements. The integer k is known. The regime of interest is

$$k < m < d.$$

In this project, we consider random (Sparse PR) problems. The measurement vectors follow independent standard normal distributions:

$$v_1, \dots, v_m \sim \mathcal{N}(0, I_d).$$

We assume that the (unknown) solution x_{sol} is distributed according to the following law: a subset $S \subset \{1, \dots, d\}$ is chosen uniformly at random among all subsets with cardinality k; for each $s \in S$, $x_{sol,s}$ is chosen according to a normal law $\mathcal{N}(0, 1)$ and, for each $s \notin S$, $x_{sol,s} = 0$.

1. Propose and describe an algorithm for Problem (Sparse PR).
2. Implement your algorithm.
3. a) Test your algorithm on the random instances described above, for several values of k, m, d. Choose two runs, one where the algorithm succeeds in recovering x_{sol}, and the other one where it fails. Each time, indicate the values of k, m, d, plot x_{sol} and the recovered vector x. Provide the distance between $(|\langle v_j, x\rangle|)_{j=1,\dots,m}$ and y. You may include other plots if you find it interesting.
   b) For the second run, can you tell why the algorithm failed?
4. a) Choose an integer $d \geq 100$. Plot, as a function of k, the minimal number m of measurements necessary so that your algorithm succeeds with probability at least (roughly) 50%.
   [We say that the algorithm "succeeds" if it returns x such that $\min(||x - x_{sol}||_2, ||x + x_{sol}||_2) \leq 0.01||x_{sol}||_2$.]
   b) Comment the plot.

Robust PCA

In a Robust PCA problem, one observes a low-rank matrix where a few coefficients have been "corrupted": they have been replaced with arbitrary values. The goal is to recover the true low-rank matrix. This can be formulated as

$$\begin{aligned} &\text{find } M \in \mathbb{R}^{d\times d} \\ &\text{such that } \operatorname{rank}(M) \leq r \\ &\text{and } ||M - Y||_0 \leq k, \end{aligned} \tag{Robust PCA}$$

where $Y \in \mathbb{R}^{d\times d}$ is the observed corrupted matrix, and k, r are known.
In this project, we consider random (Robust PCA) problems. The (unknown) solution M_{sol} is generated as

$$M_{sol} = \frac{3}{\sqrt{r}} UV^T,$$

where $U, V \in \mathbb{R}^{d\times r}$ are matrices whose coefficients are realizations of independent uniform random variables over $[-1; 1]$.[1] A subset $S \subset \{1, \dots, d\}^2$ is then chosen at random among all subsets with cardinality k, and we choose

$$\begin{aligned} Y_{s,s'} &\sim \mathcal{N}(0,1), && \forall (s,s') \in S, \\ &= (M_{sol})_{s,s'}, && \forall (s,s') \in \{1,\dots,d\}^2 \setminus S. \end{aligned}$$

1. Propose and describe an algorithm for Problem (Robust PCA).
2. Implement your algorithm.
3. a) Test your algorithm on the random instances described above, for several values of k, r, d. Choose two runs, one where the algorithm succeeds in recovering M_{sol}, and the other one where it fails. Each time, indicate the values of k, r, d, display M_{sol}, the recovered matrix M, $M_{sol} - Y$ and $M_{sol} - M$ as images. Plot the singular values of M_{sol} and M. You may include other plots if you find it interesting.
   b) For the second run, can you tell why the algorithm failed?
4. a) Choose an integer $d \geq 50$. Plot, as a function of r, the maximal number k of corrupted entries below which your algorithm succeeds with probability at least (roughly) 50%.
   [We say that the algorithm "succeeds" if it returns M such that $||M - M_{sol}||_F \leq 0.01||M_{sol}||_F$.]

[1] The normalization $\frac{3}{\sqrt{r}}$ ensures that the coefficients of M_{sol} have zero-mean and unit variance.

b) Comment the plot.

Short-and-sparse deconvolution

Given two vectors $a \in \mathbb{R}^d, x \in \mathbb{R}^n$, with $d \leq n$, we define their convolution $a \star x \in \mathbb{R}^n$ by

$$(a \star x)_k = \sum_{s=0}^{d-1} a_s x_{k-s}, \quad \forall k = 0, \ldots, n-1.$$

(Here, the coordinates of vectors are indexed by integers between 0 and $n-1$ (or $d-1$), and the indices of x are considered modulo n: $x_{-1} = x_{n-1}$.)
A short-and-sparse deconvolution problem consists in recovering a and x from $a \star x$, under the assumption that a is "short" ($d \ll n$) and x is sparse:

$$\begin{aligned} &\text{find } a \in \mathbb{R}^d, x \in \mathbb{R}^n, \\ &\text{such that } a \star x = y, \\ &\text{and } ||x||_0 \leq k. \end{aligned} \tag{SaS}$$

Here, y is given and k is known. The regime of interest is

$$k, d \ll n.$$

1. Show that, for any $a \in \mathbb{R}^d, x \in \mathbb{R}^n, \lambda \in \mathbb{R}^*$,

$$a \star x = (\lambda a) \star \left(\frac{x}{\lambda}\right).$$

Because of this scaling ambiguity, we normalize all pairs (a, x) in such a way that

$$||a||_2 = 1,$$

that is, we replace each pair (a, x) with $\left(\frac{1}{||a||_2} a, ||a||_2 x\right)$. This leaves only a sign ambiguity on the solution.
In this project, we consider random problems of the form (SaS). We assume that the unknown solution (a_{sol}, x_{sol}) is distributed according to the following law: a_{sol} is a realization of a standard normal distribution $\mathcal{N}(0, I_d)$. For x_{sol}, a subset $S \subset \{1, \ldots, n\}$ is chosen uniformly at random among all subsets with cardinality k. For each $s \in S$, $x_{sol,s}$ is chosen according to a normal law $\mathcal{N}(0, 1)$ and, for each $s \notin S$, $x_{sol,s} = 0$.

2. Propose and describe an algorithm for Problem (SaS).
3. Implement your algorithm.
4. a) Test your algorithm on the random instances described above, for several values of k, d, n. Choose two runs, one where the algorithm succeeds in recovering (a_{sol}, x_{sol}) (up to sign), and the other one where it fails. Each time, indicate the values of k, d, n, plot $a_{sol}, x_{sol}, a_{sol} \star x_{sol}$ and $a, x, a \star x$. You may include other plots if you find it interesting.
   b) For the second run, can you tell why the algorithm failed?
5. a) Choose an integer $n \geq 200$. Plot, as a function of d, the maximal sparsity k below which your algorithm succeeds with probability at least (roughly) 50%.
   [We say that the algorithm "succeeds" if it returns (a, x) such that $\min(||(a,x)-(a_{sol},x_{sol})||_2, ||(a,x)+(a_{sol},x_{sol})||_2) \leq 0.01||(a_{sol},x_{sol})||_2$.]
   b) Comment the plot.

Gaussian mixture clustering

A clustering problem consists in grouping data points into classes based on their proximity. Here, we consider problems with only two classes, and assume that the distribution of data points in each class follows a Gaussian distribution, with identity covariance, and mean depending on the class:

$$\begin{aligned} &\text{recover } \epsilon \in \{0,1\}^m \\ &\text{knowing that } x_k \sim \mathcal{N}(v_{\epsilon_k}, \Delta I_d), \text{for all } k = 1, \ldots, m, \end{aligned} \tag{Clustering}$$

where

- $v_0, v_1 \in \mathbb{R}^d$ are unknown unit-normed vectors (the means of the two classes, called *centroids*);
- $x_1, \ldots, x_m \in \mathbb{R}^d$ are the given data points;
- $\Delta > 0$ is a known parameter (which quantifies the closeness between the data points and the centroid of their class).

We assume that each coordinate of the (unknown) solution ϵ_{sol} is 0 or 1 with probability $\frac{1}{2}$, independently from the other coordinates, and that the unknown v_0, v_1 are uniformly distributed over the unit sphere of $\mathbb{R}^d$.

We consider the regime

$$m = d \quad \text{and} \quad \frac{1}{d} < \Delta \leq 1.$$

Observe that, even if we perfectly recover the clusters, we cannot say which one has label 0 and which has label 1: we cannot distinguish ϵ_{sol} and $1 - \epsilon_{sol}$. Even up to this simple ambiguity the sole knowledge of $x_1, \ldots, x_m$ is not enough to perfectly recover all labels; at best, we can recover only a "significant" fraction of them. Therefore, we evaluate the quality of a candidate solution ϵ to Problem (Clustering) through the following quantity:

$$\text{sym overlap}(\epsilon, \epsilon_{sol}) = \max(\text{overlap}(\epsilon, \epsilon_{sol}), \text{overlap}(1 - \epsilon, \epsilon_{sol})),$$
$$\text{where overlap}(\epsilon, \epsilon_{sol}) = \frac{\langle \epsilon, \epsilon_{sol} \rangle}{\max\left(||\epsilon||_1, ||\epsilon_{sol}||_1\right)}.$$

1. Show that $0 \leq \text{sym overlap}(\epsilon, \epsilon_{sol}) \leq 1$, and $\text{sym overlap}(\epsilon, \epsilon_{sol}) = 1$ if and only if $\epsilon = \epsilon_{sol}$ or $\epsilon = 1 - \epsilon_{sol}$.

Note that, except for very small values of m, if the candidate solution ϵ is simply chosen at random, we expect

$$\text{sym overlap}(\epsilon, \epsilon_{sol}) \approx \frac{1}{2}.$$

2. Propose and describe an algorithm for Problem (Clustering).
3. Implement your algorithm.
4. a) Test your algorithm on the random instances described above, for several values of m, Δ. Choose two runs, one where the algorithm succeeds in recovering a reasonable approximation of ϵ_{sol}, and the other one where it fails. Each time, indicate the values of m, Δ, plot the points $x_1, \ldots, x_m$ (projected onto the plane $\text{Vect}\{v_0, v_1\}$, to make them two-dimensional), with different colors and marker shapes to indicate the true and recovered classes. You may include other plots if you find it interesting.
   b) For the second run, can you tell why the algorithm failed?
5. a) Choose an integer $d \geq 100$. Plot, as a function of Δ, the average $\text{sym overlap}(\epsilon, \epsilon_{sol})$ obtained with your algorithm.
   b) Comment the plot.