Numerical Project in Python n.2

Image deblurring

Andrea Simonetto

version: February 27, 2023

We will consider the signal processing problem of deblurring an image. This will be formulated as an optimization problem with an ℓ_1 regularization penalty.

The goal of this project is for you to code a forward-backward method with Nesterov's acceleration and experience what it means to deal with the $f + g$ setting.

Some instructions.

- You can do the project in pairs, as long as everybody does a comparable amount of work. Since you had a first numerical project, keep the same partner in both projects and send the files at the same time.
- You need to submit a single .pdf file, of no more than 6 pages, that is readable (for example, label all your graphs in a readable manner, with a reasonable fontsize), and it has inside all the elements to assess your work. Name the file: `LastName1_LastName2_Project2_Final.pdf`
- You also need to submit a jupyter notebook that we can run to assess the correctness of your plots and graphs. The notebook will have to generate the results that you have in the report. And it has to be readable. Name the file: `LastName1_LastName2_Project2_Final.ipynb`
- The deadline for sending us the **TWO** files is **Wednesday March the 22nd, at 12H00 Paris time**, via the Moodle page of the course.
- As you figured already, you will use python, and in particular you will need the following packages (please refrain from using any other packages that are not strictly needed).

```
import numpy as np
import matplotlib.pyplot as plt
import pywt
import pylops
```

- Do as much as you can during the TP session (3 hours) at ENSTA, but it is possible that you will need to do extra work in addition to it.
- Don't wait the last day to put together the project.

Getting started

You have access to a python notebook `project-deblur.ipynb` who teaches you how to get the image and blur it, as below

Figure 1: The original and blurred image.

An image can be thought of a series of pixel. Each pixel has three dimension, but here we consider only one channel, so each pixel is just a number between 0 and 255 (RBG scale). Let $z \in \mathbf{R}^n$ be the n pixels mapped into a column vector.

Blurring causes a linear transformation of the pixels into other pixels, and with the same notation of the notebook, you have

$$z' = Cz, \qquad z = W^H x.$$

Here it is not important what these matrices are, the only important information is that C has more column than rows (i.e., we are only observing a small portion of the image).

Reconstructing the true image implies solving an inverse problem. Let b be the pixels that we observe of the blurred image, and $CW^H x$ our model of how an healthy image is distorted to account for blur.

Ideally, one would solve the convex least-squares problem

$$x^* \in \arg\min_{x \in \mathbf{R}^n} \frac{1}{2}\|CW^H x - b\|_2^2, \qquad z^* = W^H x^*.$$

However, this problem is ill-conditioned, since C may have very few rows. As alternative, one can add regularization terms to the cost as penalties to favor special properties of the solution. One such property is a minimal energy solution (for which you would add $+\epsilon\|x\|_2^2$). Another is sparsity, for which you would add $+\epsilon\|x\|_1$. We look at the second choice, which is very important in signal processing.

The problem to solve

Consider then the ℓ_1 regularized problem,

$$x^* \in \arg\min_{x \in \mathbf{R}^n} \frac{1}{2}\|CW^H x - b\|_2^2 + \epsilon\|x\|_1, \qquad z^* = W^H x^*.$$

In this project, you will code a forward-backward algorithm to solve it. Let $A = CW^H$ for simplicity, and label $f_1(x) = \|CW^H x - b\|_2^2$, $f_2(x) = \epsilon\|x\|_1$.

1. **[2 points]** Is the problem convex? Prove that $f_1 \in \mathcal{S}_{0,L}^{1,1}$. Determine an expression for the Lipschitz constant L.
2. **[2 points]** Write the proximal gradient method applied to this problem and prove that it is equivalent to ($k \in \mathbf{N}$)

$$v_k = x_k - \alpha \nabla f_1(x_k) \qquad [x_{k+1}]_i = \text{sign}([v_k]_i)(|[v_k]_i| - \alpha\epsilon)_+ \tag{1}$$

   Compute the gradient of f_1.

When the proximal gradient method is applied to an ℓ_1 regularized least-squares problem, it yields (1) and these iterations are traditionally called the ISTA (i.e., iterative soft-thresholding algorithm).

Since $f_1 \in \mathcal{S}_{0,L}^{1,1}$, we can also think to apply a Nesterov's type acceleration, leading to FISTA (i.e., fast ISTA), as follows.

$$\lambda_0 = 0, \quad \lambda_{k+1} = \frac{1 + \sqrt{1 + 4\lambda_k^2}}{2}, \quad \gamma_k = \frac{1 - \lambda_k}{\lambda_{k+1}} \tag{2}$$

$$v_k = x_k - \alpha \nabla f_1(x_k) \quad [y_{k+1}]_i = \text{sign}([v_k]_i)(|[v_k]_i| - \alpha\epsilon)_+ \quad x_{k+1} = \gamma_k y_k + (1 - \gamma_k) y_{k+1} \tag{3}$$

We have the following results,

ISTA (for $\alpha < 2/L$):

$$f_1(x_k) + f_2(x_k) - (f_1(x^*) + f_2(x^*)) \leq O(1/k).$$

FISTA (for $\alpha \leq 1/L$):

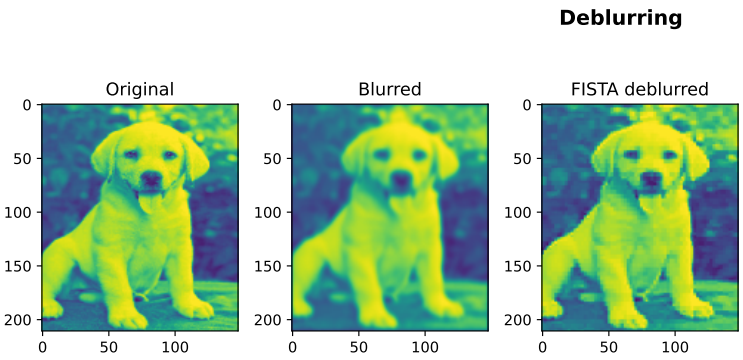
$$f_1(x_k) + f_2(x_k) - (f_1(x^*) + f_2(x^*)) \leq O(1/k^2).$$

ISTA and FISTA are two cornerstones of model signal processing and machine learning.

3. **[5 points]** Use `project-deblur.ipynb` and code your own ISTA and FISTA algorithm. Select $\alpha = 1/L$, and use the optimal solution of the already implemented `pylops.optimization.sparsity.fista` as optimal value (as shown in the code).

   Plot objective convergence, and the resulting images. You should obtain something similar to below.

4. **[3 points]** Experiment with different ϵ values (now fixed at $\epsilon = 0.1$), different step sizes, and different blurring (i.e., augment -0.1 and -0.3 in the second box $\times 10$ and diminish them $/10$).



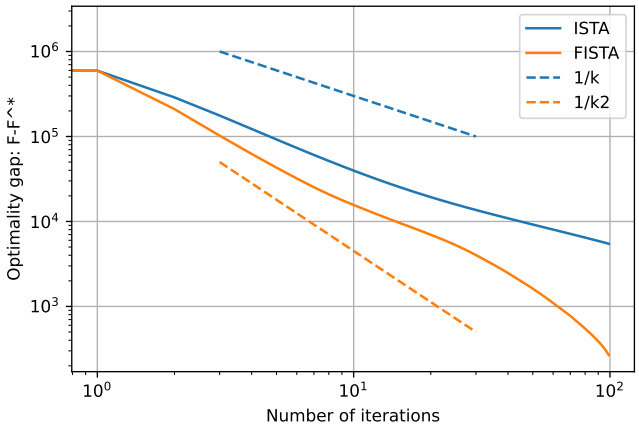


Figure 2: Sample of the results.

Going beyond ISTA: Research questions

Choose one of the following research questions.

5. **[3 points]** In the second box, diminish the sampling to $4, 3, 2, 1$, and comment on what happens. In fact, the dimension of your problem augments: describe how the computational time augments with the dimension of the problem (via a graph). Could we solve the iterates (3) in parallel on multiple machines? How?

6. **[3 points]** Instead of proximal gradient, implement a Douglas-Rachford algorithm (See TD3) and comment on what happens. Now that we can select the step size freely, can we converge faster?

Total: /15 **Points.**