LAB1 Optimisation 2

Proximal Gradient Descent

Teacher: Ahmad KARFOUL
Assistants: Augustin GODINOT [1]
Annaëlle JACOB
Maël PEREIRA

Consider a problem where you are interested in computing the value of a vector $x \in \mathbb{R}^d$. The issue is that you cannot direclty measure or observe x, you can only observe a filtered version of x: Ax with $A \in \mathbb{R}^{m \times N}$. Even worse, because no measurement device is perfect, you realize that it introduces a small noise $b \in \mathbb{R}^m$ in your measurement. At the end of the day, the value y you mesure is a noisy and filtered version of the value x you would like to compute (see (1)).

$$\underbrace{y}_{\in \mathbb{R}^m} = \underbrace{A}_{\in \mathbb{R}^{m \times N}} \underbrace{x}_{\in \mathbb{R}^N} + \underbrace{b}_{\in \mathbb{R}^m} \tag{1}$$

For example, x can be a waveform sent by Alice, which is sent through a linear channel A and gets measured by Bob with noise b picked up by his antenna.

In this lab we will explore three methods to solve the problem:
- Iterative Soft Thresholding Algorithm (ISTA), also called Proximal Gradient Descent (PGD)
- *Fast* Iterative Soft Thresholding Algorithm (FISTA), also called Accelerated Proximal Gradient Descent (APGD)
- Alternating Direction Method of Multipliers (ADMM)

A nice text book on proximal methods written by N. Parikh and S. Boyd, [1] can be found online[2]

Q1 Considering the brief introduction, what is the variable of the problem and what are we trying to solve? Explain with your own words.

1. Generating simulation data

As in the previous labs, we need to simulate data in order to benchmark the algorithms you will implement later. In the following, we set $m = 1000$ and $N = 50$.

Q2 With these values of m and N, is the problem over or under parametrized?

PY1 Create a function to generate the vector x as follows. Visualize its coefficients using the function `stem` from matplotlib.

$$\forall i \in [N], x_i = (-1)^i \exp\left(-\frac{i}{10}\right) \tag{2}$$

We generate the matrix A as follows. We set the default value of `corr` to 0.5

```
from numpy.random import multivariate_normal
from scipy.linalg import toeplitz

cov = toeplitz(corr ** np.arange(0, m))
A = multivariate_normal(np.zeros(m), cov, size=N).T
```

[1] augustin.godinot@univ-rennes.fr
[2] https://stanford.edu/~boyd/papers/pdf/prox_algs.pdf

We assume the noise to be Gaussian with mean 0 and variance σ. For now we take $\sigma = .5$.

PY2 Create a function that takes x as an input and returns the generated matrix A and the simulated measurement y. Visualize y using `stem`.

2. The proximal operator

As seen in **Q1**, we want to solve the optimization problem $\operatorname{argmin}_x \|y - Ax\|$. As we saw in the first labs, it is always a good idea to introduce a regularization term. In this lab we will consider two options for the regularization term:

- $R_1(x) = \|x\|_1$
- $R_2(x) = \|x\|_2$

Define $f(x) = \|y - Ax\|_2^2$. The problem we are interested in is

$$\operatorname*{argmin}_{x \in \mathbb{R}^N} f(x) + \lambda R(x) \tag{3}$$

Q3 Why not just use the vanilla gradient descend algorithm or the Newton algorithm we have seen in previous labs?

Recall the definition of the proximal operator of a function $R : \mathbb{R}^N \to \mathbb{R}$

$$\forall x \in \mathbb{R}^N, \quad \operatorname{prox}_{\lambda R}(x) = \operatorname*{argmin}_{z \in \mathbb{R}^N} \frac{1}{2\lambda}\|x - z\|_2^2 + R(z) \tag{4}$$

Q4 Express the proximal operator of R_1 (also called *soft-thresholding operator*). Give and explain the steps of the proof.

PY3 Create a function that takes x as argument and returns the value of $R_1(x)$. Create a function that takes λ and x as arguments and returns the value of $\operatorname{prox}_{\lambda R_1}(x)$.

It is important to create intuitive visualization of the code you implement to check that it is working properly.

PY4 Using the `stem` function, vizualize the effect of $\operatorname{prox}_{\lambda R_1}(x)$ on the coefficients of the vector x you generated in **PY1**.

Define $(a)_+ = \max(0, a)$. The proximal operator of R_2 (also called the *block soft-thresholding operator*) is

$$\operatorname{prox}_{\lambda R_2}(x) = \left(1 - \frac{\lambda}{\|x\|_2}\right)_+ x \tag{5}$$

Q5 How different is the effect of $\operatorname{prox}_{\lambda R_2}$ on a vector x compared to $\operatorname{prox}_{\lambda R_1}$?

PY5 Create a function that take x as argument and returns the value of $R_2(x)$. Create a function that takes λ and x as arguments and returns the value of $\operatorname{prox}_{\lambda R_2}(x)$.

2.1. Objective function and gradient

Recall that the objective function is

$$f(x) = \|y - Ax\|_2^2 \tag{6}$$

Q6 Compute the gradient of f with respect to x.

PY6 Create a function that takes a point x and returns the value of the loss $f(x)$ at this point. Create a function that takes a point x and returns the value of the gradient of $f(x)$ at this point.

PY7 Use the function `check_grad` to check your implementation of the gradient.

Q7 What is the definition of the Lipschitz constant of a function? Explain it in you own words. Compute the Lipschitz constant $\mathcal{L}$ of the gradient of the loss $\nabla_x f(x)$.

PY8 Create a function that takes A as a parameter and returns the Lipschitz constant of $\nabla_x f(x)$.

2.2. ISTA and FISTA algorithms

The Iterative Soft Tresholding Algorithm (ISTA) and its accelerated counterpart Fast-ISTA (FISTA) can be viewed as an extension of the classical gradient algorithm. The FISTA method was introduced by A. Beck and M. Teboulle in [2].

ISTA$(x^0, \mathcal{L}, \text{prox}_{\lambda R}, T)$:
1 **for** t **in** $\{0, ..., T-1\}$
2 $\quad x^{t+1} = \text{prox}_{\frac{R}{\mathcal{L}}}\left(x^t - \frac{1}{\mathcal{L}}\nabla f(x^t)\right)$
3 **return** $x^0, ..., x^{T-1}$

FISTA$(x^0, \mathcal{L}, \text{prox}_{\lambda R}, T)$:
1 set $z^0 = x^0, \beta^0 = 1$
2 **for** t **in** $\{0, ..., T-1\}$
3 $\quad x^{t+1} = \text{prox}_{\lambda\frac{R}{\mathcal{L}}}\left(z^t - \frac{1}{\mathcal{L}}\nabla f(z^t)\right)$
4 $\quad \beta^{t+1} = \frac{1+\sqrt{1+4(\beta^t)^2}}{2}$
5 $\quad z^{t+1} = x^{t+1} + \frac{\beta^t - 1}{\beta^{t+1}}(x^{t+1} - x^t)$
6 **return** $x^0, ..., x^{T-1}$

PY9 Implement the ISTA algorithm as a function that takes in all the relevant parameters, a number of iterations, and returns the iterates computed during the optimization.

To compare the performance of ISTA with FISTA and other proximal algorithms, we will evaluate their speed of convergence. This is done by creating what is called a convergence plot: the distance to the optimal value with respect to the number of iterations.

PY10 Using the iterates returned by your FISTA function and the true value of x you know, draw the convergence plot of the FISTA method.

PY11 How do ISTA and fista compare on your synthetic data? Compare their convergence speed (in number of iterations), the runtime (in seconds) and their computational complexity.

3. ADMM algorithm

Sometimes, optimizing the objective function f and the regularization λR in a single (proximal) gradient step can be difficult. The Alternating Direction Method of Multipliers (ADMM) [1, Section 4.4][3] provides a way to separate the optimization of the two objectives in multiple gradient steps while maintaining good convergence properties. To use the ADMM algorithm, we need to represent our problem as follows.

$$\begin{aligned} &\underset{u,v}{\text{argmin}}\; g(u) + h(v) \\ &\text{s.t. } Cu + Dv = 0 \end{aligned} \tag{7}$$

[3] Available at https://web.stanford.edu/~boyd/papers/pdf/prox_algs.pdf

Q8 Identify the variables and functions of the ADMM problem (u, v, g, h, C and D) with the variables and functions we defined in (3) (f, x, λ and R). Prove that (3) is equivalent to

$$\begin{aligned} &\underset{x,z\in\mathbb{R}^N}{\operatorname{argmin}} f(x)+\lambda R(z)+\frac{\rho}{2}\|x-z\|_2^2 \\ &\text{s.t. } z=x \end{aligned} \tag{8}$$

Q9 Write the langrangian $L(x,z,\nu)$ (with ν the Lagrange multiplier associated to the equality constraint) of (8) and find the expression of $x^* = \operatorname{argmin}_x L(x,z,\nu)$.

ADMM$(x^0, \text{prox}_{\lambda R}, T, \rho)$:
1 set $z^0 = x^0, \nu^0 = 0_{\mathbb{R}^N}$
2 **for** t **in** $\{0,...,T-1\}$
3 $\quad x^{t+1} = \operatorname{argmin}_x L(x,z,\nu)$ // descent on x
4 $\quad z^{t+1} = \text{prox}_{\lambda\frac{R}{\rho}}\left(x^{t+1}+\frac{\nu^t}{\rho}\right)$ // descent on z
5 $\quad \nu^{t+1} = \nu^t + \rho(x^{t+1}-z^{t+1})$ // multiplier update
6 **return** $x^0,...,x^{T-1}$

PY12 Implement the ADMM algorithm as a function that takes in all the relevant parameters, a number of iterations, and returns the iterates computed during the optimization.

PY13 Using the iterates returned by your ADMM function and the true value of x you know, draw the convergence plot of the ADMM method. Compare with the ISTA and FISTA methods.

4. Trying the method on real data

We will now apply the ISTA and FISTA to a very classical problem in computer vision: image denoising. For this part, you will need a image $I \in \mathbb{R}^{n\times m\times 3}$. You can choose the image you want, make sure it is not too large (max 1000×1000). To load your image, use the `Pillow` library. Here is a code example to load an image.

```
with Image.open("image.jpg") as im:
  # Provide the target width and height of the image
  (width, height) = (1_000, 1_000)
  # Convert the image to grayscale
  im = im.convert("L")
  # Resize the image to the target size
  im_resized = im.resize((width, height))
  # Convert the image into a Numpy array
  image = np.asarray(im)
```

If you prefer, you can directly use the function `scipy.datasets.face` to get an example image.

PY14 Implement a function to load you image, convert it to greyscale and normalize it. (Normalization consists in mapping the grayscale values to $[0,1]$)

4.1. Image denoising

Q10 In this image denoising problem, which variable of the problem do A, b, x and y represent? What are the expressions of A and b?

PY15 Implement a function to apply gaussian noise with mean 0 and variance σ to the image. Visualize the original image and the noised version with σ from 0.1 to 1.

PY16 Create two functions that denoises the image using the ISTA or FISTA method. Visualize the results on your image with varying levels of noise.

Bibliography

[1] N. Parikh and S. Boyd, "Proximal Algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014, doi: 10.1561/2400000003.

[2] A. Beck and M. Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009, doi: 10.1137/080716542.