INSA Toulouse
GMM 5ème année, Image

Lesson 5 : Adjoint of a linear operator

Introduction

Let T be a linear continuous mapping from an Hilbert space $\mathscr{H}_1$ to an Hilbert space $\mathscr{H}_2$. From the Riesz Theorem it exists a unique linear operator denoted T^* from $\mathscr{H}_2$ to $\mathscr{H}_1$ named the adjoint operator of T and defined by

$$\forall (x,y) \in \mathscr{H}_1 \times \mathscr{H}_2 \quad \langle Tx, y\rangle_{\mathscr{H}_2} = \langle x, T^* y\rangle_{\mathscr{H}_1}$$

from now we won't precise which scalar product we use, it will be implicit depending on the vectors we are dealing with.

This notion of adjoint operator plays a key role in many practical optimization algorithms. As an example if we consider $\mathscr{H}_1 = \mathbb{R}^n$ and $\mathscr{H}_2 = \mathbb{R}^d$ and F the function defined from $\mathscr{H}_1$ to $\mathbb{R}$ by $F(x) := \|Tx - b\|_2^2$, the gradient of F at point x is given by

$$\nabla F(x) = T^*(Tx - b)$$

It follows that being able to compute T^* will be necessary (or at least useful) to compute any Gradient Descent (GD) algorithm, any Forward Backward (FB) algorithm or any Nesterov accelerations to solve any optimization problem involving the function F. We will see, that the adjoint operator is also useful to define a dual problem associated to an optimization problem and thus the ability to compute the adjoint will be a crucial point to apply the Chambolle-Pock Primal-Dual (PD) algorithm.

1 Simple adjoint operators

1.1 A matrix is available

If $\mathscr{H}_1$ and $\mathscr{H}_2$ are two real euclidian spaces and an orthonormal basis is given for both spaces, the linear operator T is associated to matrix A, the matrix of T^* is A^t the transposed matrix of T since

$$\langle Ax, y\rangle = (Ax)^t y = x^t A^t y = x^t (A^t y) = \langle x, A^t y\rangle$$

if $\mathscr{H}_1$ and $\mathscr{H}_2$ are complex euclidian spaces the matrix of T^* is $\bar{A}^t$ the hermitian conjugate of A.

Hence if the matrix of A is given, the computation of T^* is simple, it is a simple multiplication by A^t or $(\bar{A}^t)$. Unfortunately, in many situations, especially when the dimension of x is large, the matrix A of the operator T is not given and would be too expensive to compute. Actually, in signal and image processing, fast algorithms may exist to compute a linear operator T such like the Fast Fourier Transform (FFT) to compute de Discrete Fourier Transform (DFT), or any Wavelet Transform using the Mallat's Algorithm, any filtering (or blurring) operator, discrete gradient operator and many others. But for none of them the matrix A associated to the linear transform is computed.

1.2 Orthogonal transformation

When T is a change of bases, the inverse change of bases is often available, e.g. the Inverse Fourier Transform (IFT) is the inverse transform of the FFT. But in general the inverse transform T^{-1} is not the adjoint T^* of T. However, if T is an orthogonal transformation, that is a linear transform that compute the change of coefficients between two orthogonal bases, we have $T^{-1} = T^*$. Classical orthogonal wavelet transforms are orthogonal transformation. The DFT or the Discrete Cosinus Transform (DCT) may also

be an orthogonal transformation, depending on some renormalization choices that are made. For example if we define the Fourier Transform of a vector f in $\mathbb{R}^N$ by

$$\hat{f}[k] = \frac{1}{\sqrt{N}} \sum_{\ell=0}^{N-1} f[\ell] e^{-\frac{2\pi k\ell}{N}}$$

The linear operator defined by $Tf = \hat{f}$ is orthogonal and thus $T^* = T^{-1}$ and T^* may be computed by the IFT. If we omit the factor $\frac{1}{\sqrt{N}}$ we have $T^{-1} = \frac{1}{N}T^*$. You thus have to be careful when you compute the adjoint of any orthogonal transform using the inverse. An hidden factor N may appear and depends on the exact chosen formula of the transformation.

Take away message : If T is an orthogonal transformation $T^* = T^{-1}$. An orthogonal Transformation is a change of bases between two orthonormal bases (Orthogonal Wavelets Transform, DFT, DCT, local DCT).

1.3 Orthogonal projector

For any subspace E of R^N, the orthogonal projector P on E is self-adjoint, that is $P^* = P$.

In image and signal processing, the inpainting problem which consists in recovering a signal or an image from incomplete informations may be formulated as follow : We consider incomplete observations $y = Mx$ where x are the (unknown) source data. The goal of the inpainting is to recover x form y. The operator M is a linear operator setting to 0 a part of the components of x (part of the pixels in image inpaiting). The application M is a masking operator very simple to apply : some components of x are kept and some are set to 0. The operator M is also an orthogonal projector on the subspaces generated by "visible" pixels, hence the adjoint of the masking operator is ... the operator itself. That is why it is important to recognize an orthogonal projector when you have to deal with it. Be careful, if the projector is not orthogonal it is not self-adjoint.

2 Adjoint of classical operators

2.1 Filtering and blurring

2.1.1 1D Filtering

Convolutions are classical transformations that can be applied to 1D- signal $x \in \mathbb{R}^N$: the operator T, the convolution by the filter h is defined by

$$Tx[k] = \sum_{\ell=0}^{N-1} x[\ell] h[n-\ell]$$

And we denote $Tx = h \star x$.

The convolution may be circular or not. If it is circular we set $h[j] := h[N-j]$ when $j < 0$.

We denote $\bar{h}$ the reverse filter of h defined by $\forall k \in \mathbb{Z}$, $\bar{h}[k] = h[-k]$.

The adjoint of T is thus the convolution by $\bar{h}$ indeed, for all $(x,y) \in R^N$

$$\langle Tx, y\rangle = \sum_{k=0}^{N-1} Tx[k]y[k] = \sum_{k=0}^{N-1}\sum_{\ell=0}^{N-1} x[\ell]h[k-\ell]y[k] = \sum_{\ell=0}^{N-1} x[\ell] \sum_{k=0}^{N-1} h[k-\ell]y[k] = \sum_{\ell=0}^{N-1} x[\ell](\bar{h}\star y)[\ell] = \langle x, \bar{h}\star y\rangle$$

Hence $T^* y = \bar{h} \star y$.

If the convolution is circular the filter $\bar{h}$ is the filter defined by $\bar{h}[k] = h[N-k]$ for all $k \leqslant N$.

One can observe that if the convolution is circular, the operator T is diagonal in the Fourier Transform and the Fourier Transform of $\bar{h}$ is the complexe conjugate of the Fourier Transform of h : $\hat{\bar{h}}[k] = \overline{\hat{h}[k]}$

$$\langle h\star x, y\rangle = \sum_{k=0}^{N-1} \hat{x}[k]\hat{h}[k]\overline{\hat{y}[k]} = \sum_{k=0}^{N-1} \hat{x}[k]\overline{\overline{\hat{h}[k]}\hat{y}[k]}$$

Take away message : The adjoint of a convolution by a filter h is the convolution by the reverse filter $\bar{h}$. When the convolution is circular, the convolutions by h and $\bar{h}$ can be done in the Fourier domain.

2.1.2 2D Filtering

The 2D filtering is similar to the 1D filtering. The adjoint is also a convolution by the reverse filter. The computations are the same...

If the convolution is circular, the Fourier transform of the reverse filter $\bar{h}$ is also the complex conjugate of the Fourier transform $\hat{h}$ of h.

2.2 Differential operators, the example of discrete gradient

To avoid too large local variation of a signal or an image x, a simple way is to enforce the magnitude of gradient ∇x of x to be low. Indeed in several variational problems, we will minimize a convex function involving $\|\nabla x\|_1$ or $\|\nabla x\|_2^2$. We recall that if f is a function defined from $\mathbb{R}^2$ to $\mathbb{R}$, the gradient of f is defined by the vector ∇f whose components are

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix}$$

This vector is well defined if the partial derivative of f are well defined.

In a continuous setting, the adjoint of the operator gradient ∇ is the negative divergence. If A is 2D vector field defined in $\mathbb{R}^2$ $A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$ the divergence of is given by

$$divA = \frac{\partial A_1}{\partial x_1} + \frac{\partial A_2}{\partial x_2}$$

In numerical signal and image processing, we can only use notions of discrete gradient and discrete divergence. There is no really better choice than another to perform these discretisations but they must be chosen such that the discret divergence is the adjoint of the discret gradient.
We give now an example in 1D, we will see in a next section how to apply this approach in 2 or 3 dimensions.
Let's consider a vector $x \in \mathbb{R}^N$ and let's define $\nabla x \in \mathbb{R}^N$ the discret gradient of x by

$$\nabla x[k] = \begin{cases} x[k+1] - x[k] & \text{if } k \in (0, N-1) \\ 0 & \text{if } k = N-1 \end{cases} \tag{1}$$

Here we chose to consider a right-finite difference scheme and to end the vector by 0. That is not the only choice. Now if we want to define the adjoint of this discrete differential operator we have to be careful : If (x, y) are two vectors of $\mathbb{R}^N$ then

$$\langle \nabla x, y \rangle = \sum_{k=0}^{N-1} \nabla x[k] y[k] = \sum_{k=0}^{N-2} (x[k+1] - x[k]) y[k] = \sum_{k=0}^{N-2} x[k+1] y[k] - \sum_{k=0}^{N-2} x[k] y[k] = \sum_{\ell=1}^{N-1} x[\ell] y[\ell-1] - \sum_{k=0}^{N-2} x[k] y[k].$$

$$\langle \nabla x, y \rangle = -x[0] y[0] + \sum_{\ell=1}^{N-2} x[\ell] (y[\ell-1] - y[\ell]) + x[N-1] y[N-2].$$

If we define the linear operator $\tilde{T}$ by

$$\tilde{T} y[k] = \begin{cases} -y[0] & \text{if } k = 0 \\ y[k-1] - y[k] & \text{if } k \in (1, N-2) \\ y[N-2] & \text{if } k = N-1 \end{cases}$$

we get

$$\langle \nabla x, y \rangle = \sum_{k=0}^{N-1} x[k] \tilde{T} y[k] = \langle x, \tilde{T} y \rangle$$

and we deduce that if T is the discrete gradient previously defined then $T^* = \tilde{T}$ which is the discrete negative divergence which matches with this definition of the discrete gradient. If you want to change the definition of gradient, you have to change accordingly the definition of the negative divergence.

3 Adjoint of operators combining several ones

3.1 Concatenation of operators

1. Let $x \in \mathbb{R}^N$ and T_1 and T_2 be two operators defined respectively from $\mathbb{R}^N$ to $\mathbb{R}^{n_1}$ and to $\mathbb{R}^{n_2}$ and T defined from $\mathbb{R}^N$ to $\mathbb{R}^{n_1+n_2}$ by $Tx = \begin{pmatrix} T_1 x \\ T_2 x \end{pmatrix}$. T_1 and T_2 may be two orthogonal transform and Tx is thus the sequences of decompositions components in two different bases. T_1 and T_2 may also be the vertical and the horizontal gradient.

   The adjoint operator T is thus defined from $\mathbb{R}^{n_1+n_2}$ to R^N. Any $y \in \mathbb{R}^{n_1+n_2}$ can be wrote $y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$. where $y_1 \in \mathbb{R}^{n_1}$ and $y_2 \in \mathbb{R}^{n_2}$. and the adjoint operator is thus

$$T^* y = T_1^* y_1 + T_2^* y_2. \tag{2}$$

   Indeed, for all $(x, y) \in R^N \times \mathbb{R}^{n_1+n_2}$ we have

$$\langle Tx, y \rangle = \langle T_1 x, y_1 \rangle + \langle T_2 x, y_2 \rangle = \langle x, T_1^* y_1 \rangle + \langle x, T_2^* y_2 \rangle = \langle x, T_1^* y_1 + T_2^* y_2 \rangle$$

   In words if T can be seen as the concatenation of two operators T_1 and T_2, the adjoint of T is defined from the adjoints of T_1 and T_2 through a suitable sum.

2. In the converse way if $N = N_1 + N_2$ and $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$. where $x_1 \in \mathbb{R}^{N_1}$ and $x_2 \in \mathbb{R}^{N_2}$ and T is defined from $\mathbb{R}^N$ to $\mathbb{R}^n$ as a sum $Tx = T_1 x_1 + T_2 x_2$ where T_1 and T_2 are two linear operators respectively from $\mathbb{R}^{N_1}$ and $\mathbb{R}^{N_2}$ to $\mathbb{R}^n$ then with a similar proof, one can observe that for any $y \in \mathbb{R}^n$, $T^* y = \begin{pmatrix} T_1^* y \\ T_2^* y \end{pmatrix}$.

3.2 Compositions of operators

If T_1 is a linear operator from $\mathbb{R}^{N_1}$ to $\mathbb{R}^{N_2}$ anf T_2 a linear operator from $\mathbb{R}^{N_2}$ to $\mathbb{R}^{N_3}$ then if $T = T_2 \circ T_1$ then using the definition of the adjoint it follows that $T^* = T_1^* \circ T_2^*$. This property may be interesting when the operator T may be broken into two or more simple operators for which we know the adjoint.

4 Examples

1. Sources Separation.

   Let $y \in \mathbb{R}^N$ be a signal which is the sum of two sources y_1 and y_2, $y = y_1 + y_2$. We want to recover y_1 and y_2 from y using the a priori information that y_1 and y_2 have sparse representations in different bases. To perform the source separation we want to write $y = Tx$ where x is a sparse vector. This vector x contains the coefficients of y_1 and y_2 in suitable bases. Most algorithms built to recover y_1 and y_2, such as Matching Pursuit or Forward-Backward need to apply T and T^*.

Each source is sparse in an orthonormal basis, i.e $x_1 = F_1 y_1$ is sparse and $x_2 = F_2 y_2$ is sparse. It follows that $y = F_1^{-1}x_1 + F_2^{-1}x_2$. If we denote $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ and $T_1 = F_1^{-1} = F_1^*$ and $T_2 = F_2^{-1} = F_2^*$ we can apply (2) with $Tx = F_1^{-1}x_1 + F_2^{-1}x_2$ and thus $T^*y = \begin{pmatrix} F_1 y \\ F_2 y \end{pmatrix}$.

Hence to perform the source separation we consider $y \in \mathbb{R}^N$ and we look for a vector $x \in \mathbb{R}^{2N}$ which contains two vectors x_1 and x_2 which are the coefficients of two signals y_1 and y_2 through two orthonormal transforms F_1 and F_2. To apply T to x, we break x in two parts x_1 and x_2 and apply F_1^{-1} to x_1 and F_2^{-1} to x_2. To apply T^* to a vector $y \in \mathbb{R}^N$ we compute the vector in $\mathbb{R}^{2N}$ whose the N first components are $F_1 y$ and whose the N last components are $F_2 y$.

2. Inpainting
   Let $x \in \mathbb{R}^n$ be a signal and $y = Mx$ some partial information on x where M is a masking operator keeping some components of x and setting others to 0. If we have some informations on x, for example we know that for a given orthogonal transformation F, $z = Fx$ is sparse (if x is regular and F is a cosinus or a Fourier Transform, if x is piecewise regular and F is a wavelet transform) we can rewrite the problem
   $$y = MF^{-1}z := Tz$$
   where y are the data and z a sparse vector.
   Here again we want to recover z which is sparse from y. Once we get an estimation of z we recover x using the fact that $x = F^{-1}z$.
   To apply T we first apply the inverse transform F^{-1} ans may use a fast algorithm (FFT or wavelet transform) and then we mask the result. We don't need any matrix. To compute T^* we observe that T^* is a composition of two simple operators : $T^* = F^{*-1}M^* = FM$, since F is an orthogonal transformation and M is an orthogonal projector. Thus to apply T^* to a vector u we first apply the mask to u and then we apply the transformation F.

3. 2D Gradient and negative Divergence.
   In a previous section, we explain how to compute a 1D discrete gradient of a vector x and the associated discrete adjoint the discrete negative divergence. If x is an image or a 2D object, the gradient of x at each position (i, j) is a 2 components vector. The first one is a discrete horizontal derivative and the second one a discrete vertical derivative. To compute the vertical derivative $\nabla_v x = T_1 x$, we apply the 1D discrete derivative defined in (1) on each column of x, and to get the horizontal derivative $\nabla_h x = T_2 x$ we apply (1) on each line of x. Hence the gradient of x is a linear operator :
   $$Tx := \nabla x = \begin{pmatrix} \nabla_h x \\ \nabla_v x \end{pmatrix} = \begin{pmatrix} T_1 x \\ T_2 x \end{pmatrix} \tag{3}$$
   If $x \in \mathbb{R}^N$ is an image, $Tx = \nabla x \in \mathbb{R}^N \times \mathbb{R}^N$, hence the adjoint T^* of T is defined from $\mathbb{R}^N \times \mathbb{R}^N$ to $\mathbb{R}^N$. From the previous sections, we deduce that the adjoint of T can be computed using the adjoint of T_1 and T_2 using a sum. More precisely if $y = (y_1, y_2) \in \mathbb{R}^N \times \mathbb{R}^N$
   $$T^*y = T_1^* y_1 + T_2^* y_2$$
   where T_1^* is the horizontal discrete negative divergence and T_2^* is the vertical negative divergence.

4. Integral operator, the example of Radon Transform.
   The Radon Transform is an integral transform that is used to model the tomography acquisition process. Computing the Radon Transform of a function f defined from $\mathbb{R}^2$ to $\mathbb{R}$ consists in computing the integrals of f along lines of various positions s and directions or angles θ.
   $$R(f, \theta, s) = \int_{-\infty}^{+\infty} f(z\sin\theta + s\cos\theta, -z\cos\theta + s\sin\theta)dz.$$

As an example $R(f,0,s)$ contains the integrals of f along vertical lines.

$$R(f,0,s) = \int_{-\infty}^{+\infty} f(s,-z)dz.$$

and $R(f,\frac{\pi}{2},s)$ contains the integrals of f along horizontal lines.

To compute a discrete Radon transform of a discrete image x, we first define a list of angles Θ, then for each $\theta \in \Theta$ we rotate the image x with the angle θ and for each rotation we sum the values of each pixels on each line. Hence to each direction or angle θ, $R_\theta(x)$ is a 1D signal containing the sums of values of pixels along the direction θ. The sinogram Rx of x, that is the Radon Transform applied to x for all $\theta \in \Theta$ is a 2D table where each column R_θ is indexed by a $\theta \in \Theta$.

Hence the discrete Radon Transform Rx is a concatenation of operator $R_\theta x$ which are themselves the composition of a rotation operator and an integration operation. To compute the adjoint operator of R, one thus be able to compute the adjoint of a rotation and the adjoint of an integration operator and then apply the rules described in previous sections.

— Adjoint of a rotation operator.
  In a continuous setting, the answer is simple, a rotation is an orthogonal transformation and the adjoint operator is the inverse rotation.
  The rotation of a discrete image is a non trivial problem since the original and the transformed image lives on a cartesian grid. Rotation, except for angles that are multiples of $\frac{\pi}{2}$ needs interpolations. When you compute a rotation, you must be aware of the choice that is made for the interpolation. In most librairies (Pillow in python) and matlab three interpolations are possible : nearest neighbor, bilinear and bicubic. Most of the time the default choice is the worst one : nearest neighbor. You may prefer bilinear which is more precise and no so time-consuming. Whatever your choice, the image by a rotation of the original square grid is not necessarily a square whose sides are aligned with axes. The rotation function may enlarge the original square to ensure that all points of the original image are included in the rotated image, or the rotated image may be cropped to fit the original size image. If the rotation is correctly cropped, the discrete adjoint of the rotation is very close to the inverse rotation with a crop. If it is not, the image by the rotation is larger than the original one. Then the actual rotation is a composition between a real rotation and an "extension" of the image by some zeros. Hence the adjoint is a rotation and a crop. If you compute it yourself, you must be careful.
— Adjoint of an integral operator
  Let T the linear application associating to any image x, the 1D vector of the sum along each line of x. A quick computation shows that T^* is the linear operator that is contant over each line and is equal to the value of the sum of the line on each line.

Hence the adjoint of the partial Radon Transform R_θ is an operator which associates to a 1D signal y, an image which is constant on lines in the direction θ. The values of the lines of $R_\theta^* y$ are the ones of y.

Using the formula (2) we deduce that the global adjoint of the discrete Radon transform R applied to a sinogram S is a sum

$$R^*S = \sum_i R_{\theta_i}^* S_i$$

Where S_i are the partial sinogram associated to angle θ_i. The operator R^* is also called a back-projection. The image R^*S is a sum of images constants along lines defined by angles θ_i.

A classical method to inverse the Radon Transform is the Filtered Back-Projection, actually it can be shown that R^*R is a convolution. Hence the inverse Radon Transform R can be computed applying the Back-Projection and a convolution by a ramp filter.