Compressive Sensing 2022
Claude Petit, courtesy of Aline Roumy

ComputerLab - Compressive sensing and application to MRI - Python Version

From Miki Lustig, translated to Python by Frank Ong and Jon Tamir

---

This computer lab addresses the implementation and analysis of reconstruction algorithms for compressive sensing. In particular, an application to medical resonance imaging (MRI) is addressed. The goal is to show the connection between compressive sensing and denoising. To get an intuition, this lab first explores synthetic 1D sparse signals.

This computer lab employs Python. You can also use jupyter notebook.

1 Denoising sparse 1D signals

Create a script file called tp1.py (or tp1.ipynb) that contains all instructions described below.

1.1. Generate a $1 \times n$ vector, x, with k non-zero ((1:k)/k) coefficients where $n = 128, k = 5$, and permute them randomly.
Create a graph reporting the signal as a function of the position in the vector. For each plot add labels, titles. You can use latex in the titles.

1.2. Add random Gaussian noise with standard deviation $\sigma = 0.05$ to the signal, $y = x + N$.
Create a graph showing the noisy signal.

1.3. One approach for denoising consists in using regularization with Tichonov penalty to estimate the signal from noisy data. More precisely, it solves for $\lambda \geq 0$:

$$\hat{x} = \arg\min_{z \in \mathbb{R}^n} \frac{1}{2}||z - y||_2^2 + \lambda \frac{1}{2}||z||_2^2 \tag{1}$$

This optimization trades the norm of the solution with data consistency. This approach has two advantages: (i) it has a closed form solution, (ii) the estimate is a linear function of the noisy data.
Show that the problem admits a closed form solution, and that this solution is

$$\hat{x} = \frac{1}{1+\lambda} y \tag{2}$$

1.4. Compute the estimate (2). Create a graph showing the reconstructed signal for $\lambda \in \{0.01, 0.05, 0.1, 0.2\}$. Why is the solution $\hat{x}$ not sparse?

1.5. Instead of Tichonov regularization, which penalizes the l_2 norm, we will use the l_1 norm penalized solution. More precisely, we will solve for $\lambda \geq 0$:

$$\hat{x} = \arg\min_{z \in \mathbb{R}^n} \left(\frac{1}{2}||z - y||_2^2 + \lambda ||z||_1 \right) \tag{3}$$

Show that the problem admits a closed form solution, and that this solution is

$$\hat{x} = \sigma_\lambda(y) = \begin{cases} y_l + \lambda & if \quad y_l \leq -\lambda \\ 0 & if \quad |y_l| < \lambda \\ y_l - \lambda & if \quad y_l \geq \lambda \end{cases} \qquad \forall l \in \{1, ..., n\} \tag{4}$$

In the course, the Basis Pursuit (BP) algorithm has been used instead of a closed form solution. What is the difference between (3) and the problem solved by BP?

1.6. The function (4) is often referred to as soft-thresholding or shrinkage. Write a function called SoftThresh, that implements (4).
Plot SoftThresh(u, λ) for $u \in [-10, 10]$ and $\lambda = 2$. (Note that the function is implemented for a vector of length n, here you have to plot the result for a vector of length 1).
Describe what happens when y_l is small compared to λ, and when y_l is large.

1.7. Apply SoftThresh to the noisy signal y for $\lambda \in \{0.01, 0.05, 0.1, 0.2\}$. Is the solution $\hat{x}$ sparse? Is the solution $\hat{x}$ the same as x? In particular, compare the position and the value of the non-zero coefficients.

2 Random Frequency Domain Sampling and Aliasing

We'll now explore the connection between compressive sensing and denoising and the importance of choosing a good measurement matrix. To do so, we'll observe the effect of regular and then random sampling of the signal in the frequency domain.

Create a script file called tp2 that contains all instructions described below.

2.1. Same as Part 1. Generate a $1 \times n$ vector, x, with k non-zero ((1:k)/k) coefficients *where* $n = 128, k = 5$, and permute them randomly.
Create a graph reporting the signal as a function of the position in the vector.

2.2. Compute the Fourier transform of the sparse signal, $X = Fx$ where F is a Fourier transform operator. Create a graph reporting the signal X as a function of the position in the vector.

In the following, we will measure a subset of the Fourier transform, $X_u = F_u x$ where F_u is a Fourier transform evaluated only at a subset of frequency domain samples. This is an underdetermined data set for which there is an infinite number of possible solutions. However, we do know that the original signal is sparse, so there is hope we will be able to reconstruct it. This technique of subsampling the Fourier transform coefficients can be seen as a compressive sensing technique. Indeed, in compressive sensing, we take less measurements than dictated by the Shannon-Nyquist criterion and take linear combinations of the signal. More formally, we compute $y = Mx$, where M is a matrix with more columns than rows. F_u is an example of such a compressive sensing measurement matrix. The theory of compressive sensing suggests random undersampling. To see why, we will look at (non random) equispaced undersampling and compare it to random undersampling.

2.3. Undersample the Fourier transform coefficients by taking 32 equispaced samples.
Compute the inverse Fourier transform, filling the missing data (in the frequency domain) with zeroes, and multiply it by 4 to correct for the fact that we have only 1/4 of the samples.
Plot the real and imaginary parts of xu: real(xu) imag(xu). You should observe that xu is a periodic version of x. Why? Will we be able to reconstruct the original signal from the result?

2.4. Now, undersample X by taking 32 samples at random. Compute the zero-filled inverse Fourier transform and multiply by 4 again,
Plot the real and imaginary parts of xr: real(xr) imag(xr). (Use the Matlab functions stem, hold on/off).
Describe the result. Will we be able to reconstruct the signal from the result? How does this resemble a denoising problem? And how does this differ from a denoising problem?

3 Reconstruction from Randomly Sampled Frequency Domain Data

Create a new script file called tp3 that contains all instructions described below.

3.1. Same as Part. Generate a $1 \times n$ vector, x, with k non-zero ((1:k)/k) coefficients *where* $n = 128, k = 5$, and permute them randomly. Plot x.
Create a graph reporting the signal as a function of the position in the vector. Add the title "original

signal x".

Same as Part 2: compute the Fourier transform X and undersample X by taking 32 samples at random. This leads to y, the acquired signal. Create a graph reporting the acquired signal as a function of the position in the vector. Add the title "acquired signal y".

Note that here the acquired signal y is a n-length vector, containing "0" and the "true" acquired samples. This will ease the implementation.

3.2. Modify SoftThresh to handle complex signals

$$\hat{x} = \sigma_\lambda(y) = \begin{cases} 0 & if \quad |y_l| < \lambda \\ \frac{y_l}{|y_l|}(|y_l| - \lambda) & if \quad |y_l| \geq \lambda \end{cases} \qquad \forall l \in \{1, ..., n\} \tag{5}$$

and test this function.

3.3. We will now estimate x by taking into account the fact that x is sparse.

To do so, we will implement the Projection Over Convex Sets (POCS) algorithm described below. This algorithm consists in iteratively applying soft-thresholding and constraining data consistency. First, we initially set $\hat{X}_0 = y$. Then, we implement the following iteration

3.3.1. Compute the inverse Fourier transform to get an estimate of the signal:
$\hat{x}_i = F^{-1}\widehat{X}_i = F^*\widehat{X}_i$ (F is unitary).
3.3.2. Apply SoftThresh $\hat{x}_i = \sigma_\lambda(\hat{x}_i)$ in the signal domain (where the signal is sparse)
3.3.3. Compute the Fourier transform $\hat{X}_i = F\hat{x}_i$
3.3.4. Enforce data consistency in the frequency domain

$$\widehat{X}_{i+1}(j) = \begin{cases} \widehat{X}_i(j) & \text{if } y(j) = 0 \\ y(j) & \text{otherwise} \end{cases} \tag{6}$$

3.3.5. Stop after a fixed number of iteration. Another stopping criterion can be used: repeat until $||\hat{x}_{i+1} - \hat{x}_i|| < \epsilon$ (little changes of the estimate) .

3.4. Apply the algorithm (at least 100 iterations) to the undersampled signal with $\lambda \in \{0.01, 0.05, 0.1\}$ and plot the reconstructed signal at each iteration.

3.5. Make a plot of error between the true x and the estimate at each iteration x_i as a function of the iteration number i, plotting the result for each of the λ's.Comment.

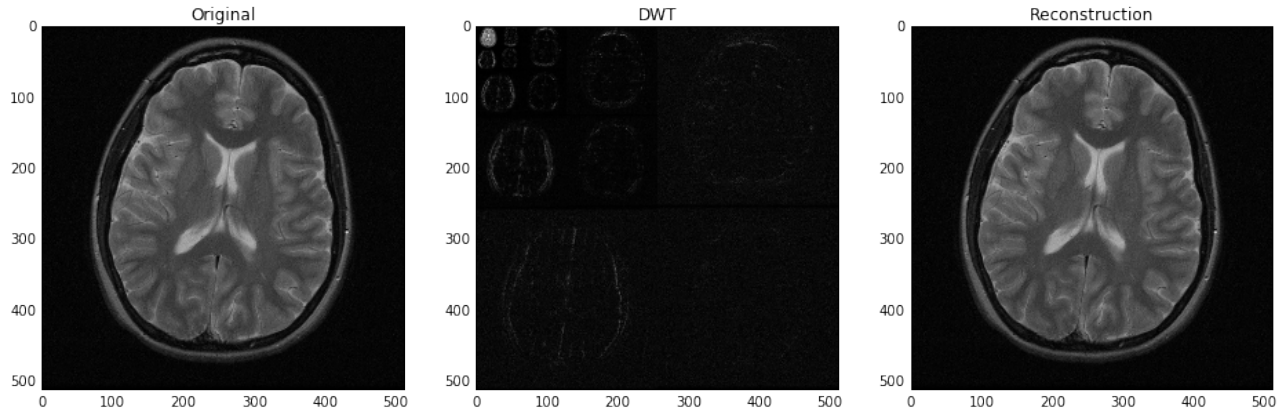
3.6. Now, repeat the iterative reconstruction for the equispaced undersampled signal. What's wrong?

4 Reconstruction of MRI acquired through compressive sensing

Download the script file called tp4.ipynb that contains all instructions and questions we briefly described below.

We now evaluate the sparse approximation of an axial T2 weighted FSE image. Even without noise, medical images are generally not sparse. However, like natural images, medical images have a sparse representation in a transform domain, such as the wavelet domain. Here, we will use the PyWavelets package to perform wavelet transforms. The PyWavelet package does not provide nice functions for visualizing the wavelet transforms. To do this, we need to define functions that stack and unstack the approximation and detail coefficients, as well as scale the different levels when displaying an image. These functions are given in the file.

The temporal MRI signal (i.e. the raw data) directly samples the spatial frequency domain of the MR image. In other words, the measurements are the 2D-Fourier transform of the MR image i.e. a linear combination of the MR signal. These raw data are usually referred to as k-space.



Compressive sensing MRI consists in sampling the k-space at a rate lower than the Shannon-Nyquist criterion. In other words, it consists in subsampling in the spatial-frequency domain. This is similar to what we studied in Part 2 and 3. The difference is that we studied 1D pure sparse signal and now we will study 2D not perfectly sparse signals.

To reconstruct the MR image, we will perform the reconstruction of the wavelet transform of the MR image, and then compute the MR image.

Open the tp4 file, read the instructions, execute the cells contents and complete it if necessary.