Computer Assignment 3

Note: Some extra information about different libraries in python and links to documentation are added in Appendix section.

1 ISTA using Python

In class we learned about the LASSO problem formulated as follows:

$$\min_x \left[J(x) = \|y - \mathbf{H}x\|_2^2 + \lambda \|x\|_1 \right] \tag{1}$$

One classical iterative algorithm that solves this problem is the ISTA algorithm formulated as follows:

$$x_{k+1} = \text{soft}\left(x_k + \frac{1}{\alpha}\mathbf{H}^T(y - \mathbf{H}x_k), \frac{\lambda}{2\alpha} \right) \qquad \text{where,} \qquad \alpha \geq \text{maxeig}(\mathbf{H}^T\mathbf{H}) \tag{2}$$

Write a function implementing ISTA algorithm for solving the LASSO problem using Python. Your program should take the dictionary matrix $\mathbf{H}$ and signal vector y as input, as well as the parameter λ, and return the solution x. Note that the parameter α should not be an input to your program and has to be determined automatically. Note that the reciprocal of α acts as a step-size parameter for the algorithm and has the effect on convergence of the algorithm. In order to satisfy the condition on α and obtain the largest possible step-size, you can set $\alpha = \text{maxeig}(\mathbf{H}^T\mathbf{H})$.
You can initialize the algorithm with any arbitrary vector x_0 (e.g. $x_0 = 0$). Set the convergence condition as $\mathit{error\_ratio} = (\mathit{old\_error} - \mathit{new\_error})/\mathit{old\_error} < T$, (set T in the order of 1e-7)where the error refers to the cost $J(x)$.
Hint: You can consult the MATLAB program in `http://eeweb.poly.edu/iselesni/lecture_notes/sparse_signal_restoration.pdf`, Professor Selesnick's lecture notes on "Sparse Signal Recovery". Note that in the Matlab program the number of iterations is specified but here you are required to check the convergence with another criterion known as the error ratio.

2 DCT transform and ISTA

Test your function for a dimension $N = 16$ vector with sparse representation using 1D-DCT transform. You can either generate a dictionary $\mathbf{H}$ that includes all the 1D-DCT basis vectors of a given dimension N or use the 1D-DCT transform pair as an operator. To generate the DCT basis, you should write your own function: `DCT_basis_gen(N)` which takes dimension N as an input and returns the 1D-DCT basis vectors. You can use the following equation to generate your DCT basis:

$$h_k(n) = \alpha(k) \cos\left[\frac{(2n+1)k\pi}{2N} \right]$$

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & k = 0 \\ \sqrt{\frac{2}{N}} & k = 1, \ldots, N-1 \end{cases}$$

Note that in the above notation, k is the index of the basis (the column index of the dictionary), and n is the index for the vector component. For example, to obtain the first DCT basis, you should set $k = 0$, and let $n = 0, 1, 2, \ldots, 15$ to obtain the 16 components of the basis vector.
Choose a coefficient vector x that is relatively sparse (i.e., only a few coefficients are non-zero), and generate a data vector y using $y = \mathbf{H}x + w$ where w is a vector containing i.i.d. Gaussian noise. You can set your noise range so that the signal to noise ratio (SNR) is 10 dB. Note that the regularization level λ depends on the noise level. Try different values of λ and comment on the effect of λ.
Optional: As an optional exercise, you could evaluate your results with several different noise levels, and for each noise level, you would need to find the appropriate λ. You could plot the error of the denoised y as a function of the noise level.

3 Wavelets and ISTA

In this exercise, you should take an image, and add noise to it to generate a noisy image (you can choose the noise level so that the SNR is 10dB). Then apply ISTA using the wavelet transform. In this case, instead of converting your entire image into a long vector, and forming a dictionary from the wavelet transform basis images, modify your ISTA function, to directly work with 2D images. Note that $\mathbf{H}x$ corresponds to performing inverse 2D transform, and $\mathbf{H}^T y$ corresponds to the forward transform. The remaining operations are element-wise operations. Therefore, you can rewrite your ISTA function to directly work with 2D images (i.e., y is an input noisy image, and x are the wavelet transform coefficients represented in whatever data structure that is returned by the 2D wavelet transform, and you do not need to explicitly save the dictionary matrix $\mathbf{H}$. Instead you can program $\mathbf{H}x$ as a function `inverse_transform(x)`, and $\mathbf{H}^T y$ as another function `forward_transform(y)`. The update of x can be done for each individual element in x. In Python, you can use `pywt.wavedec()` for forward transform, and `pywt.waverec()` for inverse transform. For this exercise, pick one noise level and tune the parameter λ such that you have good recovery. Show results using two different wavelets (Haar wavelet and the Daubechies 8/8 wavelet) and comment on the effect of different values of λ. Use 3 level decomposition. (You are encouraged to compare results obtained with different levels of decomposition. But this is not necessary)
Include plots of original image, noisy image, the wavelet transform of the noisy image, the wavelet transform of the final denoised image, and the final denoised image. Include a plot of the error vs. iteration numbers for the tuned λ. Comment on the pros and cons of using different wavelet filters.

Appendix: Basic Information

Norms

Mathematically, a function $f : \mathbb{R}^n \to \mathbb{R}$ is called a norm and noted as $f(x) = \|x\|$, if it is nonnegative, definite, homogeneous and satisfies the triangle inequality. For vector $x \in \mathbb{R}^n$ a family of norms parameterized by a constant traditionally denoted, p with $p \geq 1$ is defined as follows and called ℓ_p-norm:

$$\|x\|_p = (|x_1|^p + |x_2|^p + \ldots + |x_n|^p)^{\frac{1}{p}}$$

Note that in the family of ℓ_p-norms, ℓ_1 and ℓ_2 are the most commonly used and were discussed in class. In Python, in order to calculate the ℓ_p-norm of a 1D NumPy vector, you can use the `numpy.linalg.norm()` function. For more help you can refer to the following link:
https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.linalg.norm.html
Note that for $p < 1$ the above definition is not a mathematical norm since it does not satisfy the triangular inequality but it is still very useful in many applications and the `numpy.linalg.norm()` function still returns values for $p < 1$.

Wavelet Transform in Python

In class we learned about wavelet transforms as a set of orthonormal transform. The wavelet transform has been used as a computational tool in signal and image processing for many applications such as signal and image deblurring and denoising. As discussed in class different versions of wavelet transform can be formulated using different filters. Ingrid Daubechies, in 1992, introduced a set of filters that define a family of wavelet transforms that are very commonly used in signal processing applications.
In Python the `PyWavelets` library (also called `pywt`) has many different types of wavelet transforms implemented for 1D, 2D and nD-arrays. In order to install this library (it should already be installed with Anaconda package)you can use `pip install PyWavelets` or `conda install PyWavelets`. Also, for more documentation you can see the following link: http://pywavelets.readthedocs.io/en/latest/
Since most of the time the wavelet transform is followed by a thresholding operator for example to suppress the noise, the `PyWavelets` library has a build-in threshold function that can be used for *soft* and *hard* thresholding operations. See the following link on how to use this function: http://pywavelets.readthedocs.io/en/latest/ref/thresholding-functions.html

Eigenvalues of a Matrix in Python

You can use the `numpy.linalg.eig()` function to calculate the eigenvalues of a square matrix. Note that the values returned by this function are not ordered and for obtaining the maximum eigenvalue you can use the `max` function. See following link: https://docs.scipy.org/doc/numpy-1.10.1/reference/generated/numpy.linalg.eig.html