2.4 Convergence rates - 10 Points

In this problem, you will study the convergence of gradient descent as well as gradient descent with strong convexity. Recall that gradient descent is expected to converge in the following way under various assumptions.

Theorem 1. *Convergence rate of gradient descent (Lecture 2, slide 33)*
Let f be a twice-differentiable convex function, if

$$\begin{array}{llcll}
f \text{ is } L\text{-smooth}, & \alpha = \frac{1}{L}: & f(\mathbf{x}^k) - f(\mathbf{x}^\star) & \leq & \frac{2L}{k+4} \|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2 \\
f \text{ is } L\text{-smooth and } \mu\text{-strongly convex}, & \alpha = \frac{2}{L+\mu}: & \|\mathbf{x}^k - \mathbf{x}^\star\|_2 & \leq & \left(\frac{L-\mu}{L+\mu}\right)^k \|\mathbf{x}^0 - \mathbf{x}^\star\|_2 \\
f \text{ is } L\text{-smooth and } \mu\text{-strongly convex}, & \alpha = \frac{1}{L}: & \|\mathbf{x}^k - \mathbf{x}^\star\|_2 & \leq & \left(\frac{L-\mu}{L+\mu}\right)^{\frac{k}{2}} \|\mathbf{x}^0 - \mathbf{x}^\star\|_2
\end{array}$$

In the file `exercise_2_4.py`, you can find the incomplete code that you need to fill.

a) (2 points) Fill the lines `theoretical_rate_gd` and `theoretical_rate_gd_str` with their corresponding theoretical convergence rate. Fill the blanks in the code to have the rates plotted as a straight line. Are the observed convergence rates consistent with the theoretical ones? Why?

   Remark. Pay particular attention to the hypothesis on the function that holds in this problem.

b) (1 point) Are the observed convergence rates of GD and GDstr linear, sublinear or quadratic? Explain how you can come to this conclusion.

c) (5 points) We will try to estimate the rate of convergence of the GD and GDstr in the specific instance our problem. We assume that the observed rate will be of the form $\|\mathbf{x}^k - \mathbf{x}^\star\|_2 = a \cdot b^k$. Show that a and b can be estimated by doing a linear regression on the graph $(k, \log(\|\mathbf{x}^k - \mathbf{x}^\star\|_2)$. Fill the corresponding code.

   Remark. To closely observe the algorithms reaching the theoretical convergence rates, it would require running them for $\sim$ 100'000 iterations in this problem. However, you are only be asked to run them for 4000 iterations in consideration of your time. While you will see a trend of convergence, you will not observe the convergence at the asymptotic rate.

d) (2 points) Having obtained the linear regression coefficients $(a_{\text{GD}}, b_{\text{GD}})$ and $(a_{\text{GD\_str}}, b_{\text{GD\_str}})$, fill the blanks in the code to plot the result. Compare the empirical speed of convergence $a_{\text{GD(\_str)}}$ with the theoretical one.

Attach the linear regression coefficients, their transformed form as well as the resulting plot to your report.

3 Image reconstruction - 15 Points

Associated code folder: `exercise3/`

In this exercise an *image* should be understood as the matrix equivalent of a digital grayscale image, where an entry represents the intensity of the corresponding pixel.

3.1 Wavelets

A widely used multi-scale localized representation in signal and image processing is the wavelet transform.[2] This representation is constructed so that piecewise polynomial signals have sparse wavelet expansions [4]. Since many real-world signals can be composed by piecewise smooth parts, it follows that they have sparse or compressible wavelet expansions.

Scale. In wavelet analysis, we frequently refer to a particular scale of analysis for a signal. In particular, we consider dyadic scaling, such that the supports from one scale to the next are halved along each dimension. For images, which can be represented with matrices, we can imagine the highest scale as consisting of each pixel. At other scales, each region correspond to the union of four neighboring regions at the next higher scale, as illustrated in Figure 1.
The Wavelet transform. The wavelet transform offers a multi-scale decomposition of an image into coefficients related to different dyadic regions and orientations. In essence, each wavelet coefficient is given by the scalar product of the image with a wavelet function which is concentrated approximately on some dyadic square and has a given orientation (vertical, horizontal or diagonal). Wavelets are essentially bandpass functions that detect abrupt changes in a signal. The scale of a wavelet, which controls its support both in time and in frequency, also controls its sensitivity to changes in the signal.

[2]This section is an adaption from *Signal Dictionaries and Representations* by M. Watkin (see `http://bit.ly/1wXDDbG`).

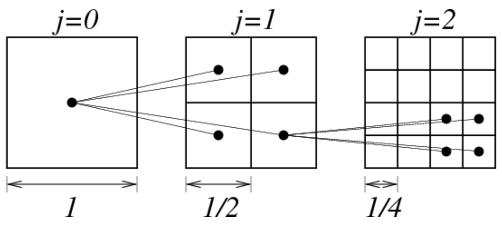

Figure 1: Dyadic partitioning of the unit square at scales $j = 0, 1, 2$. The partitioning induces a coarse-to-fine parent/child relationship that can be modeled using a tree structure.

An important property of the wavelet functions is that they form an orthonormal basis $\mathbf{W}$. Formally, this means $\mathbf{W}\mathbf{W}^T = \mathbf{W}^T\mathbf{W} = \mathbf{I}$, where $\mathbf{I}$ is the identity operator. In the discrete case, $\mathbf{W}$ is just an orthonormal matrix.

Implementation. We provide the codes that efficiently compute the 2D Wavelet transform $\mathbf{W}$ and its inverse $\mathbf{W}^{-1} = \mathbf{W}^T$. Operators $\mathbf{W}$ and $\mathbf{W}^T$ receive a vectorized image as input and outputs a vectorized transform. We base our implementation on the `PyWavelets` library[3]. You can find a usage example in the file `examples/example_wavelet.py`. The interaction with the `pywt` library happens through the class `RepresentationOperator` available in the `common.operators` file.

- The method `W(im)` computes the transformation from an image `im` to its wavelet counterpart `im_wav`.
- The method `WT(im_wav)` computes the transformation from wavelet coefficients `im_wav` back to the image domain `im`.

Remark 1. Should the provided codes not work after you installed the required packages, directly contact one of the teaching assistants.

3.2 Total Variation

The Total Variation (TV) was proposed by Rudin, Osher and Fatemi [7] as a regularizer for solving inverse problems. There is compelling empirical evidence showing that this regularization does not blur the sharp edges of images and as a consequence, significant research efforts went into developing efficient algorithms for computing the TV proximal operator. In this exercise we use the approach developed by Chambolle [1].

Computing the TV of an image relies on the discrete gradient operator $\nabla\mathbf{x} \in \mathbb{R}^{(m\times m)\times 2}$, formally given by $(\nabla\mathbf{x})_{i,j} = \left((\nabla\mathbf{x})^1_{i,j}, (\nabla\mathbf{x})^2_{i,j}\right)$ with

$$(\nabla\mathbf{x})^1_{i,j} = \begin{cases} \mathbf{x}_{i+1,j} - \mathbf{x}_{i,j} & \text{if } i < m, \\ 0 & \text{if } i = m, \end{cases}$$

$$(\nabla\mathbf{x})^2_{i,j} = \begin{cases} \mathbf{x}_{i,j+1} - \mathbf{x}_{i,j} & \text{if } j < m, \\ 0 & \text{if } i = m. \end{cases}$$

Then, the discrete Total Variation (TV) is defined as

$$\|\mathbf{x}\|_{\mathrm{TV}} := \begin{cases} \sum_{1\leq i,j\leq m} \|(\nabla\mathbf{x})_{i,j}\|_2 & \text{called } \textit{isotropic} \\ \sum_{1\leq i,j\leq m} |(\nabla\mathbf{x})^1_{i,j}| + |(\nabla\mathbf{x})^2_{i,j}| & \text{called } \textit{anisotropic} \end{cases}$$

Even though the isotropic case presents a slightly more difficult computational challenge, it has the advantage of being unaffected by the direction of the edges in images.

The TV-norm can be seen as the ℓ_1-norm of the absolute values of the gradients evaluated at each pixel. Analogous to the case of ℓ_1-norm, minimizing the TV-norm subject to data fidelity leads to sparse gradients, which implies an image with many flat regions and few sharp transitions.

[3]`https://pywavelets.readthedocs.io/en/latest/index.html`

3.3 Image in-painting

Image in-painting consists in reconstructing the missing parts of an image. By exploiting the structures such as sparsity over the wavelet coefficients, it is possible to in-paint images with a large portion of their pixels missing. In this part of the homework, we are going to study the convergence of different methods related to FISTA, as well as different restart strategies. We consider a subsampled image $\mathbf{b} = \mathbf{P}_\Omega \mathbf{x}$, where $\mathbf{P}_\Omega \in \mathbb{R}^{n\times p}$ is an operator that selects only few, $n \ll p := m^2$, pixels from the vectorized image $\mathbf{x} \in \mathbb{R}^p$. We can try to reconstruct the original image $\mathbf{x}$ by solving the following problems.

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \underbrace{\frac{1}{2}\|\mathbf{b} - \mathbf{P}_\Omega \mathbf{W}^T \boldsymbol{\alpha}\|_2^2}_{f_{\ell_1}(\boldsymbol{\alpha})} + \underbrace{\lambda_{\ell_1}\|\boldsymbol{\alpha}\|_1}_{g_{\ell_1}(\boldsymbol{\alpha})}, \tag{8}$$

$$\min_{\mathbf{x}\in\mathbb{R}^p} \underbrace{(1/2)\|\mathbf{b} - \mathbf{P}_\Omega \mathbf{x}\|_2^2}_{f_{\mathrm{TV}}(\mathbf{x})} + \underbrace{\lambda_{\mathrm{TV}}\|\mathbf{x}\|_{\mathrm{TV}}}_{g_{\mathrm{TV}}(\mathbf{x})}. \tag{9}$$

Here, the operator $\mathbf{W}^T$ is vectorized to be dimension compatible with $\mathbf{P}_\Omega$. An example of image in-painting with this model is given in Figure 2.

EXERCISE 3.3.1 - (5 POINTS)

a) *Write the gradient of* $f_{\ell_1}(\boldsymbol{\alpha})$ *in* (8) *and* $f_{TV}(\mathbf{x})$ *in* (9).

b) *Find the Lipschitz constant of* $\nabla_\alpha f_{\ell_1}(\boldsymbol{\alpha})$ *for* (8) *and* $\nabla_{\mathbf{x}} f_{TV}(\mathbf{x})$ *in* (9) *analytically.*

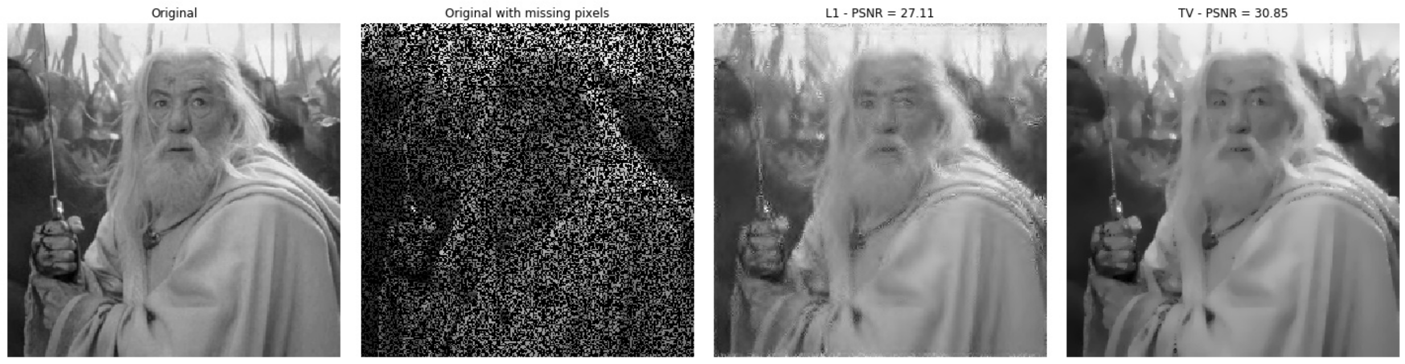

Figure 2: An example of image in-painting with the three models described in this homework. The regularization parameter has been set to 0.01 for all methods.

EXERCISE 3.3.2 - (10 POINTS)
Adapt the FISTA algorithm from Exercise 2.3 to solve the problems (8) *and* (9).

a) *Take a* 256×256 *natural image or better, a picture of you, and subsample* 40% *of its pixels uniformly at random (i.e. remove* 60% *of the pixels).*

b) *Perform a parameter sweep over* λ_1 *and* λ_{TV}. *You should choose a meaningful range for the parameters, one that allows you to observe the behavior of the PSNR as a function of the regularization parameters. Use* 200 *iterations for your algorithm.*

c) *Plot the obtained PSNR against the regularization parameter values.*

d) *Provide your codes with comments, and include the reconstructed images.*

4 Guidelines for the preparation and the submission of the homework

Work on your own. Do not copy or distribute your codes to other students in the class. Do not reuse any other code related to this homework. Here are few warnings and suggestions for you to prepare and submit your homework.

- This homework is due at 23:59, 31st of October 2021.
- Submit your work before the due date. Late submissions are not allowed and you will get 0 point from this homework if you submit it after the deadline.
- Your final report should include detailed answers and it needs to be submitted in PDF format. The PDF file can be a scan or a photo. Make sure that it is readable. The results of your simulations should also be presented in the final report with clear explanations.
- You can Python for the coding exercises. We provide some incomplete Python scripts. You need to complete the missing parts to implement the algorithms. Depending on your implementation, you might also want to change some written parts and parameters in the provided codes. In this case, indicate clearly your modifications on the original code, both inside the codes with comments, and inside your written report. Note that you are responsible for the entire code you submit.
- Your codes should be well-documented and they should work properly. Make sure that your code runs without errors. If the code you submit does not run, you will not be able to get any credits from the related exercises.
- Compress your codes and your final report into a single ZIP file, name it as `mod_2021_hw1_NameSurname.zip`, and submit it through the moodle page of the course.
- Discussing concepts with other students is OK; however, each homework exercise should be attempted and completed individually. You should write your report on your own, with your own words and understanding. Your reports and your codes will be checked thoroughly. Reports with overly similar unjustified statements will be considered as copying, and copying and cheating will not be tolerated. The first time results in zero point for the corresponding homework, and the second time results in zero point for the whole course.

References

[1] Antonin Chambolle. An algorithm for total variation minimization and applications. *J. Math. Imaging Vis.*, 20:89–97, 2004.

[2] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[3] Pontus Giselsson and Stephen Boyd. Monotonicity and restart in fast gradient methods. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 5058–5063. IEEE, 2014.

[4] S. Mallat. *A wavelet tour of signal processing*. Academic press, 1999.

[5] Brendan O'donoghue and Emmanuel Candes. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3):715–732, 2015.

[6] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[7] L.I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.