Machine Learning for Signal Processing

[5LSL0]

Rik Vullings
Ruud van Sloun
Nishith Chennakeshava
Hans van Gorp

Assignment: Deep Unfolding

April 2022

Deep Unfolding

In this assignment you will implement and analyze the Iterative Shrinkage and Thresholding Algorithm (ISTA), its learned counterpart the Learned ISTA (LISTA), and a Neural Proximal Gradient Descent algorithm.

This assignment is roughly divided into three weeks. This division is only meant as a guideline for you to follow. Note that we will not be checking your work in between weeks. This assignment is quite long and complex so make sure to start on time. Moreover, if you finish a weeks-worth of material early, do no hesitate to go along to the next week.

For questions where you need to code in Python, provide the most relevant lines of code and show the plots generated with the code in your report.

Week 1

Many practical problems can be posed as linear inverse problems of the form:

$$\mathrm{y} = \mathrm{A}\mathrm{x} + \mathrm{n}, \tag{.1}$$

where $\mathrm{y} \in \mathbb{R}^m$ is a measurement, $\mathrm{A} \in \mathbb{R}^{m \times n}$ the measurement matrix, $\mathrm{x} \in \mathbb{R}^n$ is the target, and $\mathrm{n} \in \mathbb{R}^m$ is measurement noise. Usually, A is known and y is a measurement, and we are interested in recovering x. Often, this problem is ill-posed and permits many different solutions for x. A way to resolve this, is by assuming x is sparse in some domain and then performing maximum a-posteriori (MAP) optimization. For example, if we assume x itself will be sparse and the measurement error, n, follows a Gaussian distribution, we get:

$$\hat{\mathrm{x}} = \min_{\mathrm{x}} ||\mathrm{A}\mathrm{x} - \mathrm{y}||_2^2 + \lambda ||\mathrm{x}||_1. \tag{.2}$$

A popular way find a solution to this optimization problem are proximal gradient methods such as ISTA, in which we alternate between gradient steps toward $||\mathrm{A}\mathrm{x} - \mathrm{y}||_2^2$ and proximal steps toward $\lambda ||\mathrm{x}||_1$. See the slides of the unfolding lecture for more details.

To become familiar with ISTA (and later on LISTA) we will be analyzing its performance on the MNIST denoising problem we have worked on in previous assignments. In the MNIST denoising problem the measurement matrix, A, is simply the identity matrix, and we only need to remove the noise.

To help you on your way we have provided some template code for week 1.

Exercise 1 - ISTA

(a) **[3 pt]**Create a Python function that implements ISTA for MNIST denoising. Make sure the function takes as inputs: step size μ, shrinkage parameter λ, number of iteration K, and measurements (batch of MNIST images) y. It should output the final batch of reconstructions x_K.

hint: the images are normalized between -1 and 1, which means the background of the images is not at zero. Think about how to still aply ISTA correctly.

(b) **[2 pt]** Use your ISTA algorithm on 10 example images. Show them in a figure consisting of 10 collumns (1 per digit) and 3 rows. Row 1 should contain the noisy measurement, row 2 the reconstruction, and row 3 the actual ground truth target. You will have to play with the values for μ, λ, and K. Explain what you see.

(c) **[1 pt]** To get a numerical estimate for the performance of ISTA, run it on the entire test set and report the resulting mean squared error (mse).

Exercise 2 - LISTA

We will now expand upon ISTA into its learned counterpart LISTA. In LISTA, we shall replace the data consistency step $((I - \mu \mathrm{A}^T \mathrm{A})\mathrm{x} + \mu \mathrm{A}\mathrm{y})$ with learned neural network layers. Specifically, we shall change $(I - \mu \mathrm{A}^T \mathrm{A})\mathrm{x}$ into a convolutional filter, and $\mu \mathrm{A}\mathrm{y}$ into another convolutional filter. Moreover, we will make the shrinkage parameter λ learned and layer dependent, so we shall have parameters λ_1, λ_2, λ_3, etc. Lastly, the standard soft thresholding algorithm is not very permissive of gradients, so we shall change it into a smoother counterpart as:

$$\tau(\mathrm{x}, \lambda) = \mathrm{x} + \frac{1}{2}\left(\sqrt{(x-\lambda)^2 + 1} - \sqrt{(x+\lambda)^2 + 1}\right) \tag{.3}$$

(a) **[4 pt]**Implement LISTA for 3 unfolded iterations. Train it for at least 10 epochs and plot the loss in terms of either epochs or number of batches seen. Make suitable choices for all the hyper-parameters.

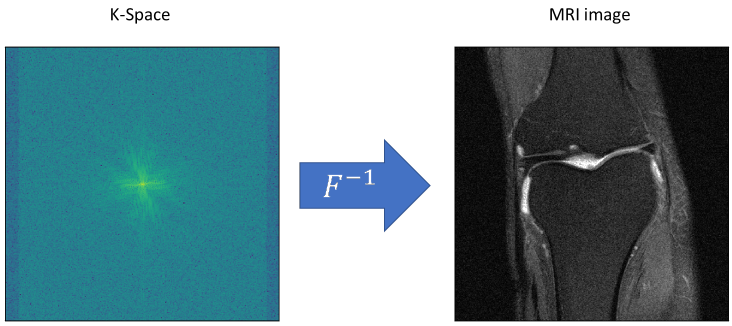
(b) **[2 pt]** Repeat exercise 1b using your trained LISTA.

(c) **[1 pt]** Repeat exercise 1c using your trained LISTA.

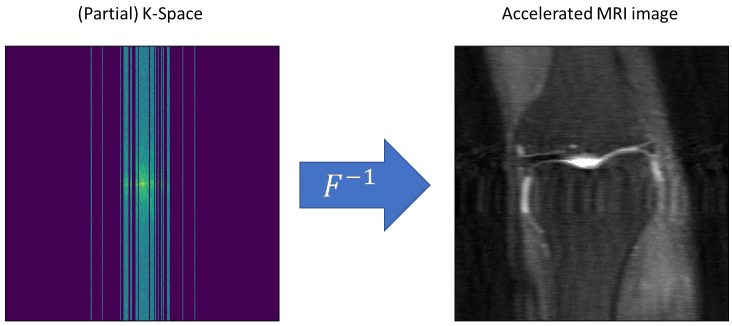
(d) **[1 pt]** As you know, neural networks need non-linear activation functions to learn a smarter mapping than just a linear one. Where is/are the non-linear activation functions in LISTA, if there even are any?

Week 2

Up till now we have mainly been looking at the denoising of MNIST digits. We shall now expand our view to more practical problems, in our case: Accelerated Magnetic Resonance Imaging (MRI). Due to the physics of how an MRI machine works (which we shall not delve into here), images of tissues inside the body can be made using what are known as k-space measurements. K-space can be though of as an analogy of the 2D-Fourier transform of an image, thus we can create an image from a k-space measurement using the 2D inverse Fourier transform:



One peculiar thing about MRI images is that we are acquiring a lot of data, the k-space above has a memory footprint of 800 KB. However, in jpeg form the image above only takes up 16 KB. Compressed sensing theory then states that we could have done better; if we can compress the image later on in the pipeline, we should be able to take fewer measurements. Accelerated MRI tries to make good on that promise by taking less K-space measurements. For example we could accelerate the acquisition process eight-fold:



Again due to the physics of an MRI machine we are acquiring entire lines in K-space. As you can see, by taking fewer measurements we still achieve an image that somewhat looks like the real image. However, we will need some type of solver to somehow recover the ground-truth image.

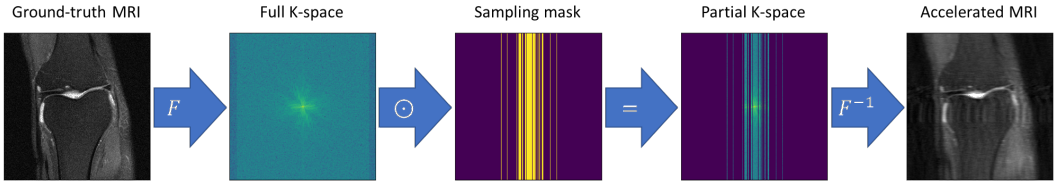
Our knowledge of linear inverse problems will now prove useful, as the Accelerated MRI recovery task can be viewed as one. Remember that the main equation for this type of problem is:

$$\mathrm{y} = \mathrm{Ax} + \mathrm{n}, \tag{.4}$$

However, the MRI image is not a vector, but an array. Furthermore, we are not measuring a linear combination of pixels, but rather lines in k-space. We will thus rewrite this equation into:

$$\mathrm{Y} = |\mathrm{F}^{-1}\left(\mathrm{M} \odot (\mathrm{FX} + \mathrm{N})\right)|, \tag{.5}$$

Where F signifies the 2D Fourier Transform and F^{-1} its inverse. X is the ground-truth fully acquired MRI image, while Y is the accelerated measurement. N is measurement noise, and M is the measurement mask, consisting of ones and zeros to under-sample k-space. Moreover, the operator, $\odot$, is element-wise multiplication. Lastly, notice how we take the absolute value, as the inverse Fourier transform will result in an imaginary signal, as we are using only partial k-space measurements. The entire pipeline this equation signifies can also be shown graphically as:



As a source of MRI data we shall be making use of the Facebook and NYU fastMRI dataset [1]. This dataset is rather large (88 GB), so we provide you with a down-sampled version consisting of 995 MRI images of knees. We have also provided you with a dataloader for said images. This dataloader will load in a partial k-space measurement, the accompanying measurement mask, and the ground truth image. Make sure to point the dataloader to the correct location. We have already created a train-test split (750 train, 245 test images, from 150 and 49 subjects, respectively).

Exercise 3 - MRI helper functions

(a) **[2pt]**Create a Python function that, given an MRI image, calculates the K-space. Make sure to only use PyTorch functions and that the function can work on multiple images in parallel. Visualize a ground truth image with its accompanying full k-space measurement.

hint: make sure that the DC (i.e., 0 Hz component) is at the center of the image. Moreover, plotting the log of the absolute value of the k-space provides better contrast.

(b) **[1pt]**Create a Python function that, given a full k-space and measurement matrix, calculates the partial k-space.

(c) **[2pt]**Create a Python function that, given a partial k-space, calculates the accelerated measurement image. Visualize a partial k-space with its accompanying accelerated MRI.

(d) **[1pt]** Recreate the pipeline image shown above for a different MRI image.

hint: for the best result, use the same 'vmin' and 'vmax' for the full and partial k-space images.

(e) **[1pt]** In the accelerated MRI image it looks as though the same image is overlayed several times, what is this effect called, and why does it only happen in the horizontal direction?

[1] https://fastmri.org/

Exercise 4 - ISTA for MRI

We will now combine our knowledge of ISTA and MRI to try to solve the accelerated MRI reconstruction challenge.

(a) **[1pt]** Central to ISTA is the data consistency step: $((I-\mu \mathrm{A}^T \mathrm{A})\mathrm{x}+\mu \mathrm{A}\mathrm{y})$, where y is the measurement, x the current guess, and A the measurement matrix. Write down what this data consistency step should look like for accelerated MRI, given measurement Y and sampling mask M.

(b) **[3 pt]**Create a Python function that implements ISTA for accelerated MRI. Make sure the function takes as inputs: step size μ, shrinkage parameter λ, number of iteration K, a batch of k-space measurements and the accompanying batch of measurement masks M. It should output the final batch of reconstructions x_K.

(c) **[2 pt]** Use your ISTA algorithm on 5 images from the test set. Show them in a figure consisting of 5 collumns (1 per image) and 3 rows. Row 1 should contain the initial reconstruction from partial k-space, row 2 the reconstruction after ISTA, and row 3 the actual ground truth target. You will have to play with the values for μ, λ, and K. Explain what you see.

(d) **[1 pt]** To get a numerical estimate for the performance of ISTA, run it on the entire test set and report the resulting mean squared error (mse).

Week 3

We will now be putting on our deep learning hats and try to come up with better alternatives to ISTA.

Exercise 5 - ConvNet

As a starting point for a deep learning solution, design an end-to-end convolutional neural network. As input it should take an initial reconstruction from a partial k-space measurement and output a guess for the final reconstruction. The net is not (yet) allowed to make use of knowledge about the sampling mask.

(a) **[1pt]** What loss function would be most suitable to train this network, and what underlying assumption have you made by choosing this loss function?

(b) **[3pt]** Design a convolutional neural network of no more than 4 layers. Train it for at least 10 epochs and plot both the training loss and testing loss in terms of epochs or batches seen. Make suitable design choices.

(c) **[2pt]** Repeat exercise 4c using your trained ConvNet.

(d) **[1pt]** Repeat exercise 4d using your trained ConvNet.

Exercise 6 - Neural proximal gradient descent

We shall now combine the strenghts of ISTA with that of our deep learning ConvNet to arrive ate some state-of-the-art results in reconstruction of accelerated MRI. To that end, we shall be employing neural proximal gradient descent. This method is simmilar to LISTA, but with some slight changes.

(a) **[1pt]** Explain what the difference is between LISTA and neural proximal gradient descent. Why would the latter be a more suitable choice for accelerated MRI?

(b) **[4pt]** Implement neural proximal gradient descent for 5 unfolded iterations. As proximal operator, make use of the ConvNet you designed in exercise 5. Train it for at least 10 epochs and plot both the training loss and testing loss in terms of epochs or batches seen.

hint: μ should remain between 0 and 1. Moreover, it helps to set a higher learning rate for the μs compared to the other trainable parameters

(c) **[2pt]** Repeat exercise 4c using your trained ProxNet.

(d) **[1pt]** Repeat exercise 4d using your trained ProxNet.

(e) **[2pt]** A direct comparison between your ProxNet and ConvNet might not be fair, why? Can you think of a way to make the comparison fairer? Implement this and compare mse again.