Constantin Pape Manuel Haussmann
mlhd1516@gmail.com

Exercise 5

Deadline: 04.12.2015 9:30 am
Please use the following subject line in the email when handing in the exercise: [EX05] followed by the names of every group member, *e.g. [EX05] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome*. Please also add the names of each group member at the beginning of any .py/.pdf/.ipynb file you hand in.
In general, hand in your code and results as a single .ipynb or .zip file (!).

This exercise is dedicated to linear regression. We will test this method on the relevant application of Tomography.

Setting

You find yourself in a two dimensional world in the role of a medical doctor. A patient (lets use only his initials H.S. here to keep his anonymity) walks into your hospital and complains about a headache. A close examination of the skull does not give you any clues about a possible cause of the pain and you realize soon that you will have to make a computer tomography. Unfortunately the computer processing the recorded data is broken. You nevertheless let H.S. enter the tomograph and his scull gets x-rayed from several angles. You send H.S. out in the waiting room and begin to think on how you could process the scans with your private laptop.

The Scans

During the scanning, H.S.'s two dimensional head was placed between the x-ray source and a one dimensional sensor. The sensor is able to record a accumulated signal in the ray direction. To reconstruct all the internal structures the measurement needs to be repeated from different angles α. Given a flattened image $\mathbf{x}$ (a vector of the size of the number of pixel N^2; to flatten the image you can use the numpy function `flatten()`) we can describe the process of one scan resulting in a sensor response $\mathbf{y}$ (a vector of the size of the number of pixels K in the sensor) by:

$$A \cdot \mathbf{x} = \mathbf{y}. \tag{1}$$

Here A is a $K \times N^2$ matrix that describes the projection from the (flattened) 2D image ($\mathbf{x}$) to the 1D sensor($\mathbf{y}$).

The ratio of absorbed rays is determined by the material in between the source and the sensor:

$$I_{sensor} = I_{source}\text{exp}\left(-\int_{ray} \mu(x,y)\;dx\;dy\right) \tag{2}$$

$$\Rightarrow -\ln\left(\frac{I_{sensor}}{I_{source}}\right) = \int_{ray} \mu(x,y)\;dx\;dy. \tag{3}$$

In a discrete case we can write:

$$\mu(x,y) = \sum_{ij} K((x,y) - ij) \tag{4}$$

Where K is a Kernel that determines how the influence of the continuous position (x, y) is mapped on the discrete positions (the centers of the pixels) ij. We assume that a ray hitting the sensor in between two center points of sensorpixels distributes its intensity only between the two neighboring

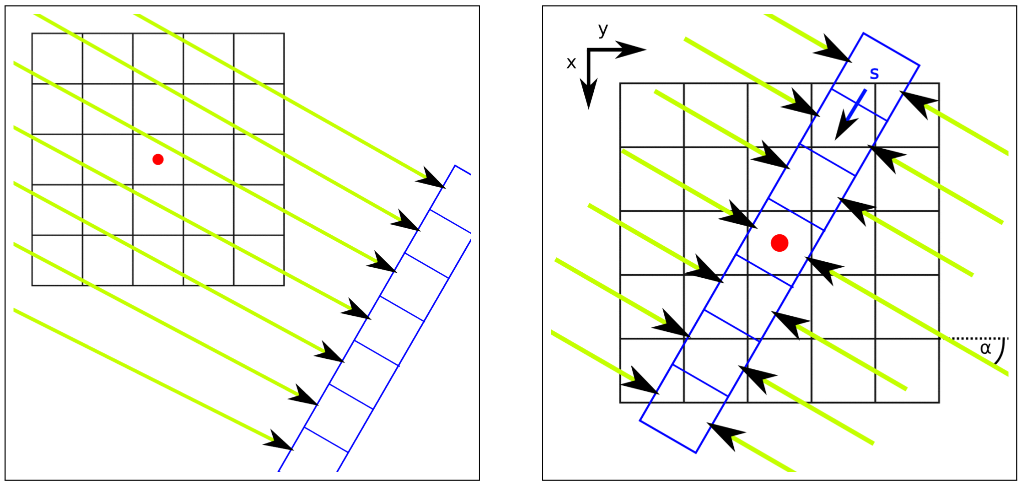

Figure 1: A sketch of a 5×5 image represented as the black grid in the background and the 7 pixel long sensor in blue that is rotating around the center of the image. Naturally the x-ray source and the sensor are both outside the tissue (left). For the one dimensional projection onto the sensor with parallel rays it would not make a difference if we would assume two opposing x-ray sources and a sensor centered in the image/tissue (right). Since this equivalent representation simplifies the construction of the matrix A you might want to adopt this view. Let's use also the convention that the counting of the sensorpixels s always starts at the sensorpixel with the lowest x value.

pixels. The ratio of the distances to the two centers of the sensorpixels is supposed to be the inverse ratio of the intensities captured by them.
Consider the following exmple: A ray hits the sensor at position 4.3. The left sensor center is at position 4 and the right sensor center at 5. Then the left sensorpixel gets 0.7 of the intensity, the one to the right 0.3. We do not have to make the ln explicitly when we assume that the sensor does not store I_{sensor} but $-\ln\left(\frac{I_{sensor}}{I_{source}}\right)$. This way we can just sum up all contributions of all image pixels.

1 Construction of A (13 points)

In order to be able to solve the inverse problem to eq. (7) and thereby reconstruct $\mathbf{x}$, we first of all need more than one projection of the tissue on the sensor (multiple alphas). So now we want A to be a $N^2 \times (MK)$ matrix where M denotes the total number of all αs used. Note that A is not dependent on the image/tissue $\mathbf{x}$. It is purely describing the setup of the tomograph. Your task is to write a function `A = makeA(shape, alphas)`. Here `shape =` (N, N) is a tuple holding the shape of the square image/tissue and `alphas = [-90, ..., ]` holds the different angles of the sensor. The concrete definition of α can be seen in Fig. 1. The size of the sensor was determined from the `shape`. It has to be long enough to cover the diagonal and if this would result in a even number of sensor-pixel then take the next highest number. A odd number of sensorpixel simplifies the problem. This way you can rotate around the central sensorpixel. If a ray passing the center of an imagepixel meets the sensor in between the centers of two sensorpixel we assume that its intensity is divided between them. As we see in Fig. 2 each imagepixel maximally influences two sensorpixel per rotation. To test your function please plot your matrix for `makeA((10, 10), [-33, 1, 42])` in a similar fashion than Fig. 2. For comparison, you can also find this matrix saved as A_example.npy ind the uploaded files.

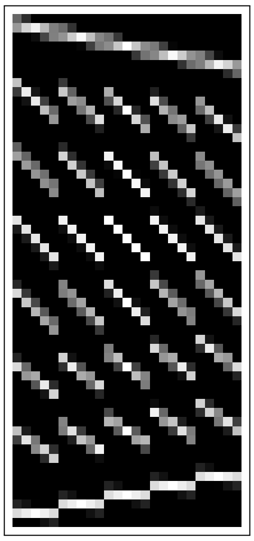

Figure 2: This shows a exemplary matrix A with $5 * 5$ imagepixel and eight projections whereas the sensor consists of 7 pixels. The angles $[-77, -33, -12, 3, 21, 42, 50, 86]$ are used. You can test your function for creating A by comparing against this result.

2 Reconstruction of the Image (4 points)

You can find the recording s of the sensors **y** here: `https://www.dropbox.com/s/j8p3qgoybwiabjg/hs_tomography.zip?dl=0`. There are projections of images of different resolution and different noise level available. With the ability to create the matrix A and given projections y we are now able to reconstruct the original image from `x` (`numpy.reshape(x, shape)` can unflatten the vector):

$$
\begin{aligned}
A \cdot \mathbf{x} &= \mathbf{y} && (5)\\
\Rightarrow A^T A \cdot \mathbf{x} &= A^T \mathbf{y} && (6)\\
\Rightarrow \mathbf{x} &= (A^T A)^{-1} A^T \mathbf{y} && (7)
\end{aligned}
$$

Doing the inverse explicitly will take forever. Luckily there is a collection of algorithms for sparse matrices in scipy (scipy.sparse.linalg). You can construct A by filling a `scipy.sparse.dok_matrix` via the normal bracket operation: eg. `dok_instance[3847, 2345] = 0.21`. You can then convert the filled Matrix via `dok_instance.tocsc()` into a Compressed Sparse Column matrix. Now you can use `lsqr` (`from scipy.sparse.linalg import lsqr`) to find the least-squares solution to a large, sparse, linear system of equations, such as the one we are dealing with here. Reconstruct the original image and plot it. Make a diagnosis on what causes H.S.'s headache and propose a treatment.

3 Minimization of the radiation dose (3 points)

Obviously one does not want to expose patients to unnecessary radiation. Try to reduce the number of projections in a sensible way and observe how this changes the quality of the reconstruction. In the case of H.S., what would you say is the minimal number of projections that still allows us to resolve the cause of his headache?

Regulations

Please hand in the python code, figures and explanations (describing clearly which belongs to which). Non-trivial sections of your code should be explained with short comments, and variables should have self-explanatory names. Plots should have informative axis labels, legends and captions. Please enclose all results into a single .pdf document and hand in the .py files that created the results. Please email the solutions to `mlhd1516@gmail.com` before the deadline. You may hand in the exercises in teams of maximally three people, which must be clearly named on the solution sheet (one email is sufficient). Discussions between different teams about the exercises are encouraged, but the code must not be copied verbatim (the same holds for any implementations which may be available on the WWW). Please respect particularly this rule, otherwise we cannot give you a passing grade. Solutions are due by email at the beginning of the next exercise. For each exercise there will be maximally 20 points assigned. If you have 50% or more points in the end of the semester you will be allowed to take the exam.