HL2027 3D image reconstruction and analysis in medicine

Miniproject 1

In this mini-project you will implement and test a reconstruction method with a regularisation term for 3D CT data of realistic size (ca. 750MB). More formally: Let x be the image to be reconstructed, and y be the result of the application of the ray transform (forward) operator A to x with additive noise ε, that is $y = Ax + \varepsilon$. The goal is to estimate x as good as possible.

Since this is an ill-posed problem, one usual alternative is to add a regularisation term. In particular, you are going to use the so-called *Huber regularisation*. This is a *Total Variation*-type of regularization, where l_1-norm is replaced with the Huber norm for convenience. Therefore, instead of solving the least-squares problem

$$\min_x \frac{1}{2}\|Ax - b\|^2, \tag{1}$$

you have to choose a parameter $\lambda > 0$ and solve the *regularised problem*

$$\min_x \frac{1}{2}\|Ax - b\|^2 + \lambda \mathcal{H}_\gamma(\nabla x), \tag{2}$$

where $\|\cdot\|$ is the norm on the *data space* (`odl.solvers.L2Norm`), $\mathcal{H}_\gamma$ is the Huber norm, and $\nabla(\cdot)$ is an operator that computes gradient of a function (`odl.Gradient`). Since in practice images have discrete representation, the gradient is approximated by computing the difference between adjacent pixels/voxels. The Huber norm (`odl.solvers.Huber`) is defined as

$$\mathcal{H}_\gamma(z) = \sum_i h_\gamma(\|z_i\|) \tag{3}$$

where $h_\gamma(\cdot)$ is the Huber function

$$h_\gamma(t) = \begin{cases} \frac{t^2}{2\gamma}, & \text{if } |t| \le \gamma \\ |t| - \frac{\gamma}{2}. \end{cases} \tag{4}$$

The Huber function $h_\gamma(t)$ is a differentiable approximation of absolute value $|t|$. It is used instead of the absolute value because differentiable functions are easier to minimize. Usually γ is chosen very small.

Solving eq. (1) leads to noisy pixel values (because of ill-posedness). As shown in the course, we had to make use of early stopping in iterative methods. The idea of Huber regularisation is to force adjacent pixels to have similar values by penalizing large gradients. In addition, using l_1-norm or Huber norm promotes the situation, when most of the gradients are 0. Still, some gradient values are allowed to be large. Thus, this regularization promotes images with large constant regions and sharp borders, e.g. cartoon-like images. Solving eq. (2) amounts to a trade-off between trying to find x that fit the data (e.g., $\|Ax - b\|$ is small), but are not very noisy, that is, $\mathcal{H}_\gamma(\nabla x)$ cannot be too large.

For a suitable choice of the regularisation parameter λ, there is no need to use early stopping: the issue of semi-convergence is (at least partially) taken care of.

Tasks

- Write a code for solving the regularised problem. If you want, you can use any optimization method implemented in ODL (`odl.solvers.smooth`). In any case, you have to make sure that the method converges by providing appropriate hyper-parameters.
- Test the code with 2D phantoms with added noise. You can find some in ODL (under `odl.phantom`)
- Choose parameter γ for the Huber functional and motivate your choice.
- Now for the regularization parameter: we have to find some λ which gives a good compromise between data-fit and regularisation. You should implement one of the methods from the lecture to find a good choice of λ for a phantom with different levels of noise. For a few values of λ and levels of noise, compute the evolution of the error with the number of iterations by comparing the noiseless image to the reconstructed one. Make sure that you adapt the method to the problem in eq. (2) and describe in your own words what you are doing.
- ONLY when the code is working in 2D phantoms, test it in the realistic dataset. This is very important as the running time could be very long depending on the number of iterations you set. Describe the effect of using different λ and describe qualitatively the evolution of the reconstructed image. Show some slices of the result. Avoid using too many iterations as this will be very slow.
- Compute a reconstruction using Filtered Back-Projection (`odl.tomo.fbp_op`). Try changing `filter_type` and `frequency_scaling`. Frequency scaling is the relative frequency cut-off for the filter, a smoother image can be obtained by choosing smaller values. Compare the results to the method used in the previous tasks.
- Document your tests and the results of the problem.

Things you might consider

- The following are useful dependencies for your code, in particular the last entry provides you with a specific geometry for the data.

```
import numpy as np
import matplotlib.pyplot as plt
import odl
import odl.contrib.tomo
```

- The ray transform is obtained as follows in ODL:

```
# The geometry is already included in ODL.
geometry = odl.contrib.tomo.elekta_icon_geometry()
# The reconstruction space
reco_space = odl.uniform_discr([-112, -112, 0],
                               [112, 112, 224],
                               (448, 448, 448),
                               dtype='float32')
# With these data, we can compute the ray transform
# (which is the operator A in the text above).
# This operator can be applied to a reco_space.element()
# and returns a data_space.element()
ray_trafo = odl.tomo.RayTransform(reco_space, geometry)
# The data space is the range of the ray transform.
data_space = ray_trafo.range
```

- The most simple way to minimize the objective function $Q(x)$ in eq. (2) is to use the steepest descent (gradient descent) method:

1: x_0 arbitrary, $r_0 := -\nabla Q(x_0)$, t - step size.
2: **for** $k := 0, 1, \ldots$ **do**
3: $\quad x_{k+1} := x_k + t r_k$
4: $\quad r_{k+1} := -\nabla Q(x_{k+1})$
5: **end for**

Note: the Landweber's method is a special case of the steepest descent.

- Optionally, you may improve the steepest descent by implementing a line search:

1: x_0 arbitrary, $r_0 := -\nabla Q(x_0)$, t - step size.
2: **for** $k := 0, 1, \ldots$ **do**
3: $\quad x_{k+1} :=$ minima of Q on half-line $t \mapsto x_k + t r_k$
4: $\quad r_{k+1} := -\nabla Q(x_{k+1})$
5: **end for**

- The steepest descent is known to converge if the step size $0 < t < 2 * \|\nabla Q\|$.
- In ODL multiplication of operators $f * g$ implements composition $f(g(\cdot))$. For example, the operator $\|Ax - b\|^2$ can be defined in ODL as follows:

  ```
  Q = odl.solvers.L2NormSquared(data_space).translated(b) * ray_trafo
  ```

  Also you can easily sum operators and multiply by a scalar. For details address documentation `https://odlgroup.github.io/odl/guide/operator_guide.html`

  Use this to define a regularization term as in eq. (2).
- The operator ∇Q can be obtained in ODL by using `.gradient` property of an operator.
- The norm of an operator can be estimated using the property `.norm(estimate=True)`. If it you have a complex operator you might want to use properties (`https://en.wikipedia.org/wiki/Operator_norm`) to estimate it's norm by parts.
- Document your code by describing what it does. Structure the functions you use so that they accept general input. Give a meaning to each parameter of your function.
- The realistic data is available in the file `noisy_data.npy` where you find some data which fits the `data_space` from above. Use np.load('/hl2027/noisy_data.npy') to get the data and data_space.element to go from a numpy array to an ODL element. The file will be also available in Canvas for those who prefer not to use the server. Compute a reconstruction of the original data. Show some slices of the result. Avoid using too many iterations as this will be very slow.

The project work can be done in groups of **2 people**. Sign up for one in Canvas in the link People → Project 1. Remember that you have to choose another partner for the next project.

We provide access to one of our servers. The number of your group in Canvas will be the same for your account in our server. You will get the login information by e-mail once you have chosen your partner.

Each group has to hand in a report of maximum **4 pages** that includes (the relative weight of each section is given in parenthesis):

1. Description of the **problems** (10%)
2. Description of the used **methods** (10%)
3. Description of the **experiments** (35%)
4. Discussion of the main findings from the **experiments** (35%)

5. A final **conclusion** (10%)

Make sure to **include contents for each** of the given tasks. The highlighted parts may help to put emphasis on certain aspects.

Each group will do a **demo and present** its results to the rest of class the **17th of March**. Each group will have **7 minutes** for this. The report and code must be submitted not later than the **15th of March**. The grades of the mini-exams of the modules *Reconstruction and Restoration* are part of the final assessment for this project:

Part	weight
Report	40 points
Demo & presentation	20 points
Code	20 points
Mini exams	20 points

We will organize an optional Q&A session the 26th of February. We strongly recommend you to start with the project soon to bring interesting questions to the session.

Good luck!!