Applied Optimization Methods for Inverse Problems
Summer term 2023

Technical University of Munich
CIIP Group

Homework sheet 6 2023-06-20

Due date: 2023-06-29 17:59

Delivery format

- Create a folder called "hw06" under the "homework" folder in your git repository.
- Answer all questions to the related to the homework in a file called "README.md" in the above created folder. **The readme must be a complete and standalone solution**. You are highly encouraged to provide instructions on how to recreate your results (e.g. run this script with this dataset, or run through this Jupyter notebook).
- Put all scripts and images you've created to reproduce and comprehend your answers in the same folder. Examples for this, are scripts to create plots, visualization, certain reconstructions to compare different strengths and weaknesses and similar things.
- The concrete implementations of the homework, must be put into the "aomip" folder. You are free to choose the structure, layout and form of your implementation.
- Once you've finished your homework, commit your changes and create a git tag with `git tag -a "hw06" -m "Tag for homework 6"` and push your changes. This needs to be tagged and pushed to your "master" branch before the deadline ends. Please use this exact command.
- Finally, once you push the changes with the tag to your repository. Create a release in GitLab. You find it in your project, under the menu 'Deployments'.

Note

Generally, we expect and reward both explorativ work and good software engineering. Hence, we highly encourage you to look at the provided resources, do your own research and try different things and play around with the tools provided to you during the course.

Without explicitly permissions you are not allowed to pull any other dependencies. If you need dependencies to load or import a dataset, please get in touch with us and let us know the limitations. If you need to open `TIFF` files, use the already included dependency `tifffile`. Use it with `tifffile.imread`. See some examples at https://pypi.org/project/tifffile/#examples. This can also handle 3D TIFF-files, which makes this quite handy.

Homework 1: Proximal Optimized Gradient Method

Last exercise we have looked a proximal algorithms. You've implemented the proximal variants of FGM. And now you should implemented the proximal version of OGM. I.e. add a new algorithm that adds the second momentum term as with OGM.

The update step fo POGM is given as

$$\begin{aligned}
\theta_k &= \begin{cases} \frac{1}{2}(1 + \sqrt{4\theta_{k-1}^2 + 1}) & \text{if } 2 \leq k < N \\ \frac{1}{2}(1 + \sqrt{8\theta_{k-1}^2 + 1}) & \text{if } k = N \end{cases} \\
\gamma_k &= \frac{1}{L}\frac{2\theta_{k-1} + \theta_k - 1}{\theta_k} \\
\omega_k &= x_{k+1} - \frac{1}{L}\nabla f(x_{k-1}) \\
z_k &= \omega_k + \underbrace{\frac{\theta_{k-1} - 1}{\theta_k}(\omega_k + \omega_{k-1})}_{\textit{Nesterov}} + \underbrace{\frac{\theta_{k-1}}{\theta_k}(\omega_k - x_{k-1})}_{\textit{OGM}} + \underbrace{\frac{\theta_{k-1} - 1}{L\gamma_{k-1}\theta_k}(z_{k-1} - x_{k-1})}_{\textit{POGM}} \\
x_k &= \text{prox}_{\gamma_k g}(z_k)
\end{aligned}$$

With N being the number of iterations to run, $\omega_0 = z_0 = x_0$ and $\theta_0 = \gamma_k = 1$.

According to Fesslers notes, POGM has a $2\times$ better convergences rate for certain scenarios, as e.g. solving the LASSO problem. Can you confirm this finding for X-ray CT? Test is for the constrained optimization problem with non-negativity constraints as well.

Homework 2: Expectation Maximization

In the lecture, you've already seen one derivation of the Maximum Likelihood Expecetation Maximization (MLEM) algorithm. This was however for emission X-ray CT for e.g. Positron emission tomography (PET) or single photon emission computed tomography (SPECT). However, we will focus on Light-field microscopy as discussed in the last section. It has a sligthly different formulation, but the underlying idea is quite similar.

The update step for MLEM in the context of Light-field microscopy is given as:

$$x_{k+1} = x_k \cdot A^* \left(\frac{b}{Ax_k} \right) \tag{1}$$

Remember, the division and multiplications of vectors are coefficient wise, i.e.

$$\frac{b}{Ax_k} = \begin{bmatrix} \frac{(b)_1}{(Ax_k)_1} \\ \frac{(b)_2}{(Ax_k)_2} \\ \vdots \end{bmatrix} \tag{2}$$

The operator A is the convolution of the signal with the PSF of the microscope. A^* is the deconvolution of the signal with the PSF.

The exercise notebook is available under . In the notebook complete the function 'compute_volume_update_MLEM' to update the volume after one iteration. The main idea is that you compute the error in image space, backpropagate it to volume space and update the volume. Return the updated volume from the function.

Homework 3: Low Dose X-ray CT

We have discussed multiple challenging scenarios for X-ray CT such as sparse view and limited angle tomography. Another major challenge is the reconstruction of low dose data. As you know, X-ray radiation is in large doses harmful for the human body, and all of the three problems are ways to reduce the the requiered dosage.

You will find data on the server from the 2016 Low Dose CT Grand Challenge. There you will find (a preprocessed version of) the training data. This training data provides both full and quarter dose projection data. For more background information checkout:

1. Low-dose CT for the detection and classification of metastatic liver lesions: Results of the 2016 Low Dose CT Grand Challenge, by McCollough et. al.
2. Low-dose CT image and projection dataset, by Moen et. al.

To load the data use the following snippet

```python
from skimage.io import imread, imsave
import json

def load_tiff_stack_with_metadata(file):
    if not (file.name.endswith('.tif') or file.name.endswith('.tiff')):
        raise FileNotFoundError('File has to be tif.')
    with tifffile.TiffFile(file) as tif:
        data = tif.asarray()
        metadata = tif.pages[0].tags["ImageDescription"].value
    metadata = metadata.replace("'", "\"")
    try:
        metadata = json.loads(metadata)
    except:
        print('The tiff file you try to open does not seem to have metadata attached.')
        metadata = None
    return data, metadata
```

The metadata contains information about source and similar. You might need to study the metadata a little more, but this snippet should help you extract the most important aspects of the metadata:

```python
# extract angles in degree
angles = np.degrees(np.array(metadata["angles"])[: metadata["rotview"]])

# setup some spacing and sizes
voxel_size = 0.7 # can be adjusted
vox_scaling = 1 / voxel_size
vol_spacing = vox_scaling
vol_size = image_size

# size of detector
det_count = sino_data.shape[:-1]
det_spacing = vox_scaling * metadata["du"]

# Distances from source to center, and center to detector
ds2c = vox_scaling * metadata["dso"]
dc2d = vox_scaling * metadata["ddo"]
```

First, play around with the dataset and make yourself familiar with it. Have a look at both sinograms for full and low dose. Then perform any reconstruction of a full dose and low dose projection data. How do they compare? Show both images side by side and discuss the kind of visible artifacts in the low dose reconstruction.

In the low-dose setting, the assumption of Gaussian noise is wrong. A better suited model is the transmission log likelihood, which is given by:

$$\sum_{i=1}^{n}(b_i e^{-[Ax]_i} + r_i) - y_i \log(b_i e^{-[Ax]_i} + r_i) \tag{3}$$

Here, A is the X-ray transform, x is the current guess, $[Ax]_i$ is the i-th entry of the forward projection, b are blank scans, and r are the number of background events. If you have no information about the background events, then $r = \mathbf{0}$. And if you have no blank scans provided by the dataset, then $b = I_0$. Note, that usually, you only have a couple of blank scans, however the size of b is equal to the size of the sinogram. For practical purposes, it is sufficient to have a single blank scan of the size of the of a projection, then NumPy can broadcast it in the multiplication.

The gradient of the transmission log likelihood functional is:

$$A'\left(be^{-Ax}\left[\frac{y}{be^{-Ax} + r} - 1\right]\right) \tag{4}$$

with $r = \mathbf{0}$ it simplifies to:

$$A'(y - be^{-Ax}) \tag{5}$$

Now, from here you can reconstruct transmission data. You can use any gradient based method both proximal and others. Try this with a couple of datasets, and specifically with the low-dose clinical dataset.

I suggest you start with a dataset dataset, for which the raw transmission images are provided (for which this should be easier). Next, try to transform the clinical data from the Mayo clinic to transmission data and recosntruct it using the transmission log likelihood. Can you reduce the artifacts in the low dose case using the transmission log likelihood compared to the Gaussian least squares model?

A couple of important practical aspects. For the transformation from absorption to transmission data, you must normalize the sinogram to the range between 0 and 1 first. Then transform it to transmission data.

Further, you would normally need the initial intensity I_0. However, to the best of my knowledge it is not clearly provided by the Mayo dataset. The best I can provide you with is, that it should be around $1e^8$.