Homework sheet 5 2023-06-13

Due date: 2023-06-22 17:59

Delivery format

- Create a folder called "hw05" 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 "hw05" -m "Tag for homework 5"` 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 Operators

In the lecture you've looked at proximal operators. One note to connect some different aspects together. Given a closed convex set $\mathcal{C}$, the indicator function $\chi_{\mathcal{C}}$ for that set is defined as:

$$\chi_{\mathcal{C}}(x) = \begin{cases} 0 & x \in \mathcal{C} \\ \infty & x \notin \mathcal{C} \end{cases} \tag{1}$$

This definition can be used for e.g. the box set you have dealt with last week together with the projected gradient descent. Now, the proximal operator of the indicator function is simply the Euclidean

projection onto the set $\mathcal{C}$. Hence, proximal operators can be viewed as a generalization of the projection. This also leads to the generalization of the projected gradient descent, the proximal gradient descent (more on that later).

So, you've implemented the proximal operator for the indicator function already. But you also have implemented the proximal operator of the ℓ^1 norm. This is simply the soft threshold operator. So now we still want some more. Please implement the proximal operators of the functions given below.

All proximal operators depend on a parameter σ, which is usually passed as an argument. They will be denoted as $\text{prox}_{\sigma f}$.

Constant functional Let $f(x) = c$ for some $c \in \mathbb{R}$ be the constant functional. The proximal operator of the constant function is simply the identity function

$$\text{prox}_{\sigma f}(v) = v \tag{2}$$

The proximal operator is independent of σ.

Why is that useful? We will use it down below for some experiment, but for even more advanced algorithms it can be useful to set some part of the problem formulation to the zero function. We will cover that, when it's useful.

Translation Given the function $f(x) = g(x - y)$ for some function g, you can compute the proximal operator of an arbitrary translation, the following way:

$$\text{prox}_{\sigma f}(v) = y + \text{prox}_{\sigma g}(x - y) \tag{3}$$

Assuming you can evaluate the proximal of g, you can evaluate the proximal of a translated function.

ℓ^2-Norm Squared Given the function $f = \frac{\lambda}{2}||x||_2^2$, i.e. squared ℓ^2-Norm with a regularization parameter. The proximal operator is given as:

$$\text{prox}_{\sigma f}(v) = \frac{x}{1 + \sigma\lambda} \tag{4}$$

Huber Let f being the Huber functional for some δ:

$$f(x) = \begin{cases} \frac{x_i^2}{2\delta} & \text{if } |x_i| < \delta \\ |x_i|\frac{\delta}{\delta} & \text{else} \end{cases} \qquad \forall i \tag{5}$$

Then the proximal operator is defined as:

$$\text{prox}_{\sigma f}(v) = \left(1 - \frac{\sigma}{\max\{||x||_2, \sigma\} + \delta}\right) v \tag{6}$$

Homework 2: Proximal Gradient Method

In this, we will cover two methods explained and discussed by Fessler regarding the proximal gradient method (PGM).

In general we will be talking about a function f, which can be written as:

$$f(x) = g(x) + h(x) \tag{7}$$

where g is (L-Lipschitz) differentiable and h is prox friendly.

i) Proximal Gradient Method

In the last exercise, you implemented ISTA and the projected gradient descent. Both of them are a special case of the PGM. So you can generalize them in some way to, instead of taking a projection or being fixed with the soft threshold, accept a proximal operator.

ii) Fast proximal gradient method

In the first couple of exercises, we have spent quite some time to look at different acceleration methods regarding gradient descent. The same ideas can be translated to the PGM methods. Using Nesterov's momentum, you get to an algorithm called fast proximal gradient method (FPGM).

The update step is given as:

$$\begin{aligned} z_k &= \text{prox}_{\frac{1}{L}g}\left(x_{k-1} - \frac{1}{L}\nabla f(x_{k-1})\right) \\ x_k &= z_k + \alpha_k(z_k - z_{k-1}) \end{aligned}$$

Popular choices for momentum parameters $\{\alpha_k\}$:

- $a_k = \frac{k-1}{k+2}$
- $a_k = \frac{t_{k-1}-1}{t_k}$, where $t_0 = 1$ and $t_k = \frac{1}{2}\left(1 + \sqrt{1 + 4t_{k+1}^2}\right)$

Experiment with both choices, can you see that one is better or worse? If can you spot a pattern for certain problems or choices of problem formulation?

The worst-case convergence rate is $\mathcal{O}(\frac{1}{N^2})$. Can you verify the improved convergence rate over the simple PGM algorithm?

iii) Uniqueness of formulation

Imagine you want to solve the Tikhonov regularization:

$$f(x) = \frac{1}{2}||Ax - b||_2^2 + \frac{\beta}{2}||x||_2^2 \tag{8}$$

Then you have two choices for PGM (an it's variants):

1. $g(x) = \frac{1}{2}||Ax - b||_2^2 + \frac{\beta}{2}||x||_2^2$ and $h(x) = 0$
2. $g(x) = \frac{1}{2}||Ax - b||_2^2$ and $h(x) = \frac{\beta}{2}||x||_2^2$

Test both formulations and check the convergence, is one converging faster than the other? Why may that be so, can you find an explanation?

iv) Elastic Net Formulation

We can now also solve the elastic-net problem given in one of the early lectures:

$$f(x) = \frac{1}{2}||Ax - b||_2^2 + \beta||x||_1 + \frac{\tau}{2}||x||_2^2 \tag{9}$$

Solve this problem for X-ray CT and the challenge data set. Can you improve your reconstructions for low angle tomography?

Homework 3: Restart conditions

Algorithms are complex and strange, and sometimes they get stuck, overshoot in the wrong direction and waste time. One method to deal with that is to restart the algorithm. Basically, you set x_0 to your current iterate and reset all variables. You can either restart every k iterations, or more adaptively.

In this exercise, you should explore a fixed restart every k iterations. First, explore it with a known phantom, forward project the phantom, add noise and do the reconstruction (with any problem formulation you choose) and gather the reconstruction error and the objective function value for each iteration. For this synthetic test, perform it with a larger variety of algorithms both proximal and first-order methods. Visualize the reconstruction error and objective function value for some algorithm with and without restart. Choose good candidates to show if restarting is beneficial in your experiment.

As a second step, do the experiment again for real data or the challenge data. Perform this step with a smaller selection of algorithms.

Can you improve the convergence of the algorithm by using a fixed restart version? How often does a restart benefit you?