Homework sheet 4 2023-05-23

Due date: 2023-06-15 17:59

The goal of this homework, is to create more gradient based iterative reconstruction techniques.

Delivery format

- Create a folder called "hw04" 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 "hw04" -m "Tag for homework 4"` 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: Even more gradient based methods

Here we cover some more advanced topics in the space of gradient based methods. Namely, we will talk about choosing the correct step length and ISTA.

i) Choosing the correct step length

For first-order methods, the choice of step length is crucial. However, choosing it correctly is difficult. The problem is often referred to as line search.

Line search methods solve:

$$\arg\min_{\alpha} f(x_k + \alpha \nabla f(x_k)) \tag{1}$$

However, it might not be necessary to find a global minimum to this function. Generally, it is sufficient to find an α such that $f(x_k) > f(x_k - \alpha \nabla f(x_k))$, i.e. they ensure that we actually decrease our cost function.

The most often implemented (inexact) line-search algorithms are:

- Backtracking
- The choices from Barzilai and Borwein (BB)

For notation, let's call the search direction: $p = \nabla f(x_k)$. And for a comprehensive introduction, you can check out chapter 9 of the reference by Boyd and Vandenberghe and chapter 3 of Nocedal and Wright.

For all the methods below: implement them, adapt your gradient descent to use it, test them on small problems you can visualize easily using contour plots. Further test it on the X-ray CT problems. Visualize the choice of parameters. For X-ray CT reconstructions, make a convergence analysis using the reconstruction error ($||x - x_k||_2^2$). How close do they get? How fast do they get closer?

Backtracking

The principle behind backtracking is quite simple. You decrease your step length until a certain condition is met. The condition we will use here is the one by Armijo.

Given two control parameters $\rho \in (0, 1)$, $c \in (0, 1)$, and some initial step length $\alpha_0 > 0$. Then you update with $\alpha_{j+1} = \rho \alpha_j$, until

$$f(x_k - \alpha_j p) \leq f(x_k) - c\alpha_j ||p||_2^2 \tag{2}$$

The final α_j is the step length the function returns. Also note, that x_k is constant here.

(Optional): Wolfe Conditions Without going into details here, the Wolfe conditions add further restrictions on the step length. You can take a look at the reference by Nocedal and Wright. However, it also increases the computational and implementation effort.

Instead of implementing backtracking with the Wolfe conditions, take a look at the SciPy reference for their line search method, which satisfy the strong Wolfe conditions.

Build an example and show how the SciPy method compares to your backtracking algorithm. How does the choices differ? If you can, adapt your gradient descent algorithm to incorporate the line search method from SciPy and perform a reconstruction. How well does it perform? Both from a runtime and convergence perspective.

Barzilai and Borwein

The BB step lengths can be computed by

$$\alpha_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} \quad \text{or} \quad \alpha_k = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}}$$

where $s_{k-1} = x_k - x_{k-1}$ and $y_k = p_k - p_{k-1}$, with p_k being the k-th step direction. The two different choices are known as BB1 and BB2. Implement both and compare them with backtracking. How is the performance characteristic of this method compared to backtracking?

ii) Iterative Shrinkage-Thresholding Algorithm

We will take a look at the Iterative Shrinkage-Thresholding Algorithm (ISTA). It is the first method to solve the LASSO problem. And in the coming lectures and homework, we will build on this foundation to build more and more powerful methods! ISTA is mostly known due to the paper by Beck and Teboulle, which you can find in the references.

Your homework is to implement ISTA in its basic form. Compare it to previously implemented algorithm. Specifically, compare it to gradient methods that approximate sparsity using the Huber functional or the Fair Potential. What role does the regularization parameter play in the reconstruction? Use it to solve X-ray and deblurring problems. For the latter, it should outperform the Tikhonov regularization, can you reproduce that?

Finally, look at the choice of step length. For the least squares term you can compute the Lipschitz constant using the power-iterations from the last exercise, can you find a decent default that works as a step size? Further, incorporate your previously implemented line search methods and see how they perform. Can they beat your default choice? As we heard from the lecture, there is still ongoing research, but just try it out!

Algorithm ISTA solves the LASSO problem:

$$\min_x g(x) + \beta||x||_1 \tag{3}$$

where g is differentiable (i.e. least squares). The update step is then given as:

$$x_{k+1} = \text{soft}(x_k - \alpha_k \nabla g(x_k), \beta\alpha_k) \tag{4}$$

soft is the soft-thresholding operations as defined below.

Soft-Thresholding The soft-thresholding operator is simply given by

$$\text{soft}(v, \rho) = \text{sign}(v)\max(|v| - \rho, 0) \tag{5}$$

All operations are performed coefficient wise.

iii) Projected Gradient Descent

A method very closely connected to ISTA is the projected gradient method. Next lesson, we will tie them together and find a common theoretical ground.

The update step for projected gradient method is very similar to the one by ISTA, only it doesn't use the soft-thresholding operation, but rather a projection onto the set $\mathcal{C}$. The update step is given as:

$$x_{k+1} = \mathcal{P}_{\mathcal{C}}(x_k - \alpha_k \nabla g(x_k)) \tag{6}$$

Note, that projection must be an orthogonal projection. If the set is closed and convex, the orthogonal projection minimized the steps stated there. But otherwise, the algorithm might not converge.

In the exercise, you already implemented the non-negativity projection, adopt it here and use it with projected gradient method and compare it to ISTA. Compare the convergence, image quality and general performance.

What kind of values make sense for the lower and upper bound for the projection for X-ray CT? Reason for some and try them out, do the results yield the results you expected?

References:

- Convex Optimization, Stephen Boyd and Lieven Vandenberghe. Available at https://web.stanford.edu/~boyd/cvxbook/
- Numerical Optimization, J. Nocedal, S. J. Wright (2006)
- Minimization of functions having Lipschitz continuous first partial derivatives, L. Armijo (1966)
- Two-point step size gradient methods, J. Barzilai and J. M. Borwein (1988)
- A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems, A. Beck and M. Teboulle (2009)

Homework 2: Semi-Convergence

Some algorithms have a certain in some sense favorable property, called semi-convergence. Iterative solvers that exhibit semi-convergence start by converging to a regularized solution - often in the first few iterations - but then in later iterations, the solution is deteriorated by noise.

Conduct the following experiment: Set up a known phantom, forward project it, add noise to the sinogram, and then run some iterative reconstruction algorithm, to solve the least squares problem (e.g. gradient descent or Landweber iterations) For every couple of iterations, compute and store the reconstruction error ($||x - x_k||_2^2$ where x is your phantom, and x_k is your current estimate). Finally, plot the gathered data.

What do you see? What do you expect the result to look like from the above description of semi-convergence? What algorithm, which you've implemented, exhibit semi-convergence? Show examples of the best reconstruction according to the reconstruction error, and the final image. Is the final image deteriorated by noise?

And finally, what is your takeaway from this experiment? You might want to repeat the experiment with different levels of noise. Also compare it to a reconstruction, which explicitly uses regularization.

Homework 3: Challenge Dataset

Use the newly developed algorithms and methods and try to explore more limited-angle problems (e.g. 90° or 60°) with the challenge dataset. Can you achieve better scores with the newly developed methods? If so, why do you think so? If not, why?

Submit some of your trials to the leaderboard.

(Optional) Homework 4: Catch up

If you haven't finished some of the work of the previous homework sheets, it's recommended to go back and implement them now. The most important thing to implement, if you haven't, is finite differences. It will be requiered and very important in one of the next homeworks.

Further, if you haven't implemented deblurring or deconvolution, we highly recommend to play around with it further, as we will put more time into in the near future as well!

Else, spend time improving code quality or certain things that come up during the time you might want to change now. We also recommend looking into nice visualization of e.g. geometries.