Advanced Methods for Mathematical Image Analysis Winter School

Modern Techniques of Large Scale Optimization for Data Science

Exercise sheet 2

Problem 1: An interior point method for X-ray tomography

In this exercise, you will see an IPM in action on an X-ray tomography reconstruction problem. In particular, the problem consists in understanding the distribution of two materials inside a 2-dimensional object. The problem takes the following form

$$\hat{\mathbf{g}} = \begin{bmatrix} \hat{\mathbf{g}}^{(1)} \\ \hat{\mathbf{g}}^{(2)} \end{bmatrix} = \text{argmin}_{\mathbf{g}^{(j)} \geq 0} \left\{ \|\mathbf{m} - \mathcal{A}\mathbf{g}\|_2^2 + \alpha \mathcal{R}(\mathbf{g}) + \beta \mathcal{S}(\mathbf{g}) \right\},$$

where $\hat{\mathbf{g}}^{(j)}$ contains the distribution of material j, $\mathbf{m}$ is the measurement vector, $\mathcal{R}(\mathbf{g}) = \|\mathbf{g}\|^2$ is a standard Tikhonov regularizer, with parameter α, while $\mathcal{S}(\mathbf{g}) = 2\ (\hat{\mathbf{g}}^{(1)})^T \hat{\mathbf{g}}^{(2)}$ is a regularizer that promotes the separation between the two materials, with parameter β. $\mathcal{A}$ is the following operator

$$\mathcal{A} := \begin{bmatrix} c_{11}A & c_{12}A \\ c_{21}A & c_{22}A \end{bmatrix},$$

where $c_{11}, c_{12}, c_{21}, c_{22}$ are the attenuation coefficients of the two materials at two different X-ray energies and A is a matrix that depends on the geometry of the tomographic imaging process.

This problem can be rewritten as a standard quadratic program

$$\text{argmin}_{\mathbf{g}^{(j)} \geq 0} \ -\mathbf{m}^T \mathcal{A}\mathbf{g} + \frac{1}{2}\mathbf{g}^T Q \mathbf{g},$$

where

$$Q := Q_1 + Q_2, \quad Q_1 := \mathcal{A}^T\mathcal{A} = \begin{bmatrix} (c_{11}^2 + c_{21}^2)A^TA & (c_{11}c_{12} + c_{21}c_{22})A^TA \\ (c_{11}c_{12} + c_{21}c_{22})A^TA & (c_{12}^2 + c_{22}^2)A^TA \end{bmatrix}, \quad Q_2 := \begin{bmatrix} \alpha I & \beta I \\ \beta I & \alpha I \end{bmatrix}.$$

An IPM applied to this problem finds the Newton direction $(\Delta\mathbf{g}, \Delta\mathbf{s})$ in the following way

$$(Q + G^{-1}S)\Delta\mathbf{g} = \mathbf{r_1} + G^{-1}\mathbf{r_2}, \qquad \Delta\mathbf{s} = G^{-1}(\mathbf{r_2} - S\Delta\mathbf{g}), \tag{1}$$

where $G = diag(g)$, $S = diag(s)$ and $\mathbf{r}_1$ and $\mathbf{r}_2$ are vectors computed by the IPM algorithm.

Download the files for this session and open `EXERCISE_XRAY.m`. This script sets up the problem and calls an IPM to solve it (`ipm_xray.m`). You are given the functions `apply_Operator_A.m`, that applies $\mathcal{A}$ to a vector (i.e. performs the matrix vector product with $\mathcal{A}$), `apply_Operator_At.m`, that applies $\mathcal{A}^T$, `apply_Q2.m`, that applies Q_2, and `apply_prec.m`, that applies a preconditioner.

For the IPM to work, you need to complete the following tasks:

Task 1: Implement the function `apply_NE.m`, that applies the normal equations matrix to a vector, i.e. that performs the operation $(Q + G^{-1}S)\mathbf{x}$.

Task 2: Inside `ipm_xray.m`, call the PCG to solve the normal equations system (1); use `apply_NE.m` from the previous task and the provided preconditioner, set a tolerance of 10^{-6} and 500 as maximum number of iterations.

Now the script `EXAMPLE_XRAY.m` should run correctly. At the beginning of the script, you can choose the size of the problem; try to run it with $N = 8$, $N = 16$ or $N = 32$. You can also try to run the PCG with or without preconditioner and observe the behaviour of the number of PCG iterations. Try to change the value of the parameter β; what effect does it have on the number of PCG iterations and on the quality of the final images?

You can find more details about this problem and the specific interior point method in the paper:

J. Gondzio, M. Lassas, S. Latva-Aijo, S. Siltanen and F. Zanetti, *Material-separating regularizer for multi-energy X-ray tomography.* Inverse Problems *38*, 2, **2022**, 025013.

Extra Task (if you have time): Have a look at `apply_prec.m`; can you understand how the preconditioner works?

Problem 2: An interior point method to recover a partially known image

In this exercise, you will see an IPM in action on the problem of computing a low-rank approximation of an image that is only partially known because some pixels are missing. We analyze the case when the missing pixels are randomly distributed, see Figure 1 (downloaded from `http://www.imageprocessingplace.com`)

(a) True image (b) 50% of missing pixels

Figure 1: The Lake 512×512 original grayscale image and the inpainted version (50% of missing pixels).

We formulate it as a **matrix completion problem** that takes the following form

$$\begin{array}{ll} \min & rank(\bar{X}) \\ \text{s.t.} & \bar{X}_\Omega = B_\Omega, \end{array} \tag{2}$$

where Ω is the set of locations corresponding to the observed pixels of original image B and the equality is meant element-wise, that is $\bar{X}_{s,t} = B_{s,t}$, for all $(s,t) \in \Omega$. Let m be the cardinality of Ω and r be the rank of B. We expect this rank to be **small**. Define

$$X = \begin{bmatrix} W_1 & \bar{X} \\ \bar{X}^T & W_2 \end{bmatrix},$$

where $\bar{X} \in \mathbb{R}^{n_1 \times n_2}$ is the matrix to be recovered and W_1, W_2 are symmetric matrices. Observe that the minimization of $trace(X) = trace(W_1) + trace(W_2)$ naturally promotes that $\bar{X}$ has small rank.

Then problem (2) can be formulated as a standard semidefinite programming (SDP) problem

$$\begin{array}{ll} \min & \frac{1}{2} I \bullet X \\ \text{s.t.} & \begin{bmatrix} 0 & \Theta_{st} \\ \Theta_{st}^T & 0 \end{bmatrix} \bullet X = B_{(s,t)}, \quad (s,t) \in \Omega \\ & X \succeq 0, \end{array} \tag{3}$$

where for each $(s,t) \in \Omega$ the matrix Θ_{st} is defined element-wise as $(\Theta_{st})_{kl} = \begin{cases} 1/2 & \text{if } (k,l) = (s,t) \\ 0 & \text{otherwise,} \end{cases}$.

We observe that the recovered image $\bar{X}$ corresponds to the (1,2)-block of the primal variable X, the symmetric matrix C in the objective of a standard SDP is a scaled identity matrix, the vector $b \in \mathbb{R}^m$ is defined by the known pixels of the image B and, for $i = 1, \ldots, m$, each constraint matrix A_i, corresponds to the known elements of the image B stored in b_i.

Since we are interested in finding a **low-rank solution** X of (3), a specialised relaxed version of the IPM will be used: the Relaxed Interior Point Algorithm for Low Rank SDP (IPLR) is an IPM that promotes the computation of low-rank solutions of SDPs.

Download the files for this session and open and read the documentation of `IPLR_BB.m`. This is an implementation of IPLR for general standard SDPs where a Barzilai-Borwein gradient method is used for computing the Newton steps. The function header is

```
function [X, y, S] = IPLR_BB(C,A,b,r,S,y)
```

where C, A, b are the SDP data, r is the sought rank and (S, y) is a dual feasible starting pair. The output is a primal-dual solution where X has rank r.

In order to recover the inpainted image in Figure 1, you need to complete the following tasks:

Task 1: Set up the SDP problem
Write the script `mc_image.m` which:

- reads the image 'lake.bmp' and shows it in Matlab
  ```
  B = imread('lake.bmp');
  figure(1)
  imshow(B);
  ```
- converts the image into a double variable and assigns the first slice to variable `B_true` (also recovers the image dimensions);
  ```
  B = double(B); B_true = B(:,:,1); [n1,n2] = size(B_true);
  ```
- produces an image with `p`% of known pixels by generating m random entries and scales it by its norm:
  ```
  p = 0.8; m = round(p*(n1*n2));
  Omega = randsample(n1*n2,m);
  ```

Implement the routine

```
function [A,b,C] = getInput(B_true, Omega)
```

to define the matrices A, C and the vector b of the SDP formulation (3) and call it in the script.

To produce the (normalized) vector b you need to:

- Define `b = B_true(Omega);`
- Set `nb = norm(b); b = b/nb`

Moreover, you can use the supplied function `A = generate_constraints(Omega,n1,n2)` to generate the matrix A of the constraints in (3). Such function is provided in the routine `getInput.m`

Task 2: Recover the image using IPLR
Inside `mc_image.m`, define a dual feasible starting point for S and y, set a value of the rank r, say $r = 80$, and call `IPLR_BB.m`.

Task 3: Analyze the results
Extract the recovered image `B_rec` from the output of `IPLR_BB.m`, rescale it back by `nB` and show it. Also, display the error in Frobenius norm wrt the true image both considering all the pixels (`B_true`) and only those in Ω (`B_part`), and finally display the value of the signal-to-noise ratio:

$$\texttt{PSNR} = 10 * \texttt{log10}(\texttt{n1} * \texttt{n2} * 255^2/(\texttt{norm}(\texttt{B\_true} - \texttt{B\_rec})^2)).$$

Try different values for p (percentage of known pixels) and for the rank r. What effect does it have on the number of IPM iterations and on the quality of the final images?

You can find more details about this problem and the specific interior point method in the paper:

S. Bellavia, J. Gondzio, and M. Porcelli, *A relaxed interior point method for low-rank semidefinite programming problems with applications to matrix completion*, Journal of Scientific Computing, 89 (2021), pp. 1-36.

Extra Task (if you have time): compute the best approximation of rank r of the original image `B_true` using the SVD; can you understand the pros and cons of using this technique compared to the SDP approach?