Homework 7 - ADMM algorithm to solve Robust PCA problem

Attention: Because math rendering of .Rmd is not ideal, please see the enclosed pdf for correct rendering of all equations (an exact copy of this file)

Introduction

In this homework, you will be implementing ADMM algorithm to solve the robust PCA problem. The idea behind robust PCA is as follows: given a matrix $M \in \mathbb{R}^{n \times p}$, you want to decompose this matrix as a sum of low-rank and sparse matrices. The standard PCA can be viewed as a low-rank matrix decomposition of given M (no sparse matrix). In contrast, the robust PCA uses an extra sparse matrix to allow for "few" outliers - "few" elements of M that deviate from low rank structure (hence extra sparse matrix, only "few" nonzero elements).

Given matrix M, Robust PCA is formulated as optimization problem

$$\text{minimize}_{L,S} \{\|L\|_* + \gamma \|S\|_1\} \quad \text{s.t.} \quad L + S = M.$$

Here $\|L\|_*$ is the nuclear norm of L (discussed in class), $\|S\|_1 = \sum_{i,j} |s_{i,j}|$ is the sum of absolute values of the elements of matrix S (ℓ_1 norm applied to vectorized S, i.e. lasso type penalty). The parameter $\gamma > 0$ controls the relative weights between the nuclear norm and the sparsity penalties, and is chosen by the user.

The Robust PCA problem is already in the form required by ADMM, so the implementation should consist of three updates: update of L, update of S and update of dual variable H (matrix η, we consider scaled ADMM for simplicity). You should discover that the first two updates correspond to proximal operators of nuclear norm and ℓ_1 norm, correspondingly, evaluated at specific points.

Remark: ADMM updates in vector form generalize easily to matrices by substituting squared Frobenius norm $\|\cdot\|_F^2$ instead of squared euclidean norm $\|\cdot\|_2^2$, and the matrix inner-product (trace operator $\text{tr}(A^\top B)$) instead of the vector inner-product $a^\top b$.

Derivation of ADMM updates

Your first task is to derive an explicit form of all three updates for ADMM for Robust PCA problem above. To submit your derivations, please **directly modify the README.Rmd** document in this folder (use R markdown), and **knit** the document to update the corresponding pdf accordingly.

Augmented Lagrangian: given $\tau > 0$ ($\tau = 1/\rho$)

$$L_\tau(L, S, \nu) = \{||L||_* + \gamma||S||_1 + tr(\nu^T(L + S - M)) + \frac{1}{2\tau}||L + S - M||_F^2\}$$

Using the scaled version of ADMM updates, let $\eta = \nu \cdot \tau$. Derive explicit updates below.

Update of L:

$$L^{t+1} = \arg\min_L \{\mathcal{L}_\tau(L, S_t, \frac{\eta_t}{\tau})\}, \text{ (Where } \nu_t = \frac{\eta_t}{\tau})$$

which is equivalent to (derive the explicit update and fill in below, use more than one line if necessary)

$$L^{t+1} = \arg\min_L \{||L||_* + tr((\frac{\eta_t}{\tau})^T(L + S_t - M) + \frac{1}{2\tau}tr((L + S_t - M)^T(L + S_t - M))\} \quad (1)$$

(Dropping $\gamma||S_t||_1$, since irrelevant to L)

$$= \arg\min_L \{||L||_* + \frac{1}{2\tau}tr(2\eta_t^T(L + S_t - M) + (L + S_t - M)^T(L + S_t - M))\} \quad (2)$$

$$= \arg\min_L \{||L||_* + \frac{1}{2\tau}(||(L + S_t - M) + \eta_t||_F^2 - ||\eta_t||_F^2)\} \quad (3)$$

$$= \arg\min_L \{||L||_* + \frac{1}{2\tau}||L - (M - S_t - \eta_t)||_F^2\} \quad (4)$$

(Dropping $||\eta_t||_F^2$, since irrelevant to L)

$$= prox_{\tau||\cdot||_*}(M - S_t - \eta_t) = US_\tau(D)V^T \quad (5)$$

(Where UDV^T is SVD of $M - S_t - \eta_t$)

Update of S:

$$S^{t+1} = \arg\min_S \{\mathcal{L}_\tau(L_{t+1}, S, \frac{\eta_t}{\tau})\}, \text{ (Where } \nu_t = \frac{\eta_t}{\tau})$$

which is equivalent to (derive the explicit update and fill in below)

$$S^{t+1} = \arg\min_S \{\gamma||S||_1 + tr((\frac{\eta_t}{\tau})^T(L_{t+1} + S - M) + \frac{1}{2\tau}tr((L_{t+1} + S - M)^T(L_{t+1} + S - M))\} \quad (6)$$

(Dropping $||L_{t+1}||_*$, since irrelevant to S)

$$= \arg\min_S \{\gamma||S||_1 + \frac{1}{2\tau}(||(L_{t+1} + S - M) + \eta_t||_F^2 - ||\eta_t||_F^2)\} \quad (7)$$

(Using the same technique in above)

$$= \arg\min_S \{\gamma||S||_1 + \frac{1}{2\tau}||S - (M - L_{t+1} - \eta_t)||_F^2\} \quad (8)$$

(Dropping $||\eta_t||_F^2$, since irrelevant to S)

$$= \arg\min_S \{||S||_1 + \frac{1}{2\tau\gamma}||S - (M - L_{t+1} - \eta_t)||_F^2\} \quad (9)$$

$$= prox_{\tau\gamma||\cdot||_1}(M - L_{t+1} - \eta_t) = Soft_{\gamma\tau}(M - L_{t+1} - \eta_t) \quad (10)$$

Update of H: (fill in, note that here H is a matrix version of η)

$$H^{t+1} = H^t + \frac{1}{\tau}(L_{t+1} + S_{t+1} - M)$$

Starter code

The starter code for four functions with detailed description is provided in **ADMMfunctions.R**. The first three are helper functions, whereas the last function is the full ADMM algorithm. Feel free to create additional functions if needed, however please adhere to specified format of the 4 provided functions.

- There are two different soft-thresholding functions that you need to implement: 1) `soft` is a standard soft-thresholding but now generalized to work on matrix-valued input; 2) `soft_nuclear` is the soft-thresholding of singular values of a matrix that arises in the context of nuclear norm minimization.

The script **ApplyADMM.R** has a code to apply your function on a simple synthetic example. The provided example is an illustration of how the sparsity part allows to extract low-rank component even when the original matrix is far from low-rank. You don't need to add any additional code to this script.

The script **TestADMM.R** is a place holder for your own tests. As **ApplyADMM.R** is for exploratory purposes only, you still need to extensively test your functions to make sure they are running correctly.

Grading criteria

Your assignment will be graded based on

- ADMM derivations *(10% of the grade)*

Filling out the ADMM updates in this .Rmd correctly.

- correctness *(40% of the grade)*

Take advantage of objective function values over iterations as a way to indirectly check the correctness of your function.

- speed *(30% of the grade)*

There is not a lot of degrees of freedom in this assignment, but you will get +5 bonus points if your **completely correct** code on 1st submission is faster than mine (median your time/median mine time $<$ 0.9). My function in ApplyADMM.R script takes as median 37 sec to run on my laptop for that specific example with that specific seed and parameter selection.

- code style/documentation *(10% of the grade)*

You need to comment different parts of the code so it's clear what they do, have good indentation, readable code with names that make sense. See guidelines on R style, and posted grading rubric.

- version control/commit practices *(10% of the grade)*

I expect you to start early on this assignment, and work gradually. You want to commit often, have logically organized commits with short description that makes sense. See guidelines on good commit practices, and posted grading rubric.