Matlab Project: solving Lasso problem by ADMM and Proximal Gradient

(CUHK-SZ, S2020)

Lasso Model

We consider recovering a sparse signal $\hat{x} \in \mathbb{R}^n$ that approximately satisfies an under-determined linear system $Ax = b \in \mathbb{R}^m$, where $m < n$, with the help of ℓ_1-regularization by solving the Lasso problem

$$\min_{x \in \mathbb{R}^n} \ \rho\|x\|_1 + \frac{1}{2}\|Ax - b\|_2^2, \tag{1}$$

where the parameter $\rho > 0$ and the matrix A may or may not satisfy $AA^T = I$.

Task: Write 4 Solvers

To solve the above Lasso problem, write 4 Matlab functions implementing 4 algorithms (see below):

```
[x,myout] = myADMMp(A, b, myinp);     % primal ADMM
[x,myout] = myADMMd(A, b, myinp);     % dual ADMM
[x,myout] = myProxG(A, b, myinp);     % proximal gradient
[x,myout] = myFISTA(A, b, myinp);     % fast proximal gradient
```

where inputs A, b are problem data, "myinp" is a structure carrying necessary input quantities, while "myout" carries desired output information besides the computed solution x. *The above interface specifications must be strictly followed.*

Structure "myinp" should have at least the following 3 fields as inputs:

1. myinp.rho that holds the model parameter ρ value,
2. myinp.tol for a tolerance value to be used in a termination condition,
3. myinp.nonorth that holds a logical value: myinp.nonorth = 1 (true) if $AA^T \neq I$ (non-orthogonal) and myinp.nonorth = 0 (false) if $AA^T = I$ (orthogonal). Whenever possible, your solvers should take advantage of this structural information to enhance efficiency.

The output "myout" should include "myout.iter" and other relevant quantities of iteration history, such as function value at each iteration, to help evaluate the algorithm's progress throughout iterations.

You are allowed to develop your solvers by adopting the codes available from the following Stanford University websites:

- `https://web.stanford.edu/~boyd/papers/prox_algs/lasso.html`
- `https://web.stanford.edu/~boyd/papers/admm/lasso/lasso.html`

that contain implementations of a primal ADMM, a proximal gradient method and an accelerated (fast) version. These implementations are not necessarily efficient (for example, they do not take advantage of the structure $AA^T = I$ when it is known to exist). If you do decide to use the Stanford codes (none of other sources for existing codes is allowable), *you must acknowledge their usage in your written report* and describe what you have done to improve them.

Evaluations and Comparisons

A part of the evaluation will be done with the help of a Matlab package YALL1 (version 1.4) and other codes from the instructor. Download the YALL1 package. Read the manual to familiarize its basic usage. Run the YALL1 solver, with relevant inputs, side by side with your solvers to evaluate their correctness and efficiency. Of course your codes should first pass a set of correctness tests before their efficiency can be evaluated. It is your task to design and conduct a reasonably comprehensive numerical study to evaluate the correctness and efficiency of your solvers in comparison with the provided codes YALL1, yzProxG and yzFISTA, as well as in comparison between your solvers themselves.

Your numerical study should utilize multiple types of problems. For example, you should have test instances with A matrices representing the two cases: $AA^T \neq I$ and $AA^T = I$. The size of the test problems should be pushed to a sufficiently large scale to see how the solvers handle the "stress".

A simple test script *test0_lasso.m* is handed out to help you get started. Your should write more comprehensive test scripts to generate visually informative figures and other forms of evidence for your conclusions. Your evaluation results should be well summarized and nicely presented in your project report, with figures and possibly tables as well.

There are two must-run tests: that is, to run the two test scripts: *test1d_lasso.m* and *test2d_lasso.m*, and submit outputs. To compare efficiency fairly, make sure that the solvers involved in any experiment achieve *a similar level of solution accuracy* on the tested problems. In each test, you only need to run two solvers of yours: one of ADMM type and one of proximal gradient type. You can select the code by specifying the appropriate solver names in the two test script.

Submission Items

- **4 Source Codes as a ZIP file**

  Your 4 Matlab solvers: myADMMp.m, myADMMd.m, myProxG.m and myFISTA.m. Please submit them as a single zip file named: *solvers-yourname.zip*, where you substitute in your actual name.

- **Report and Results as a PDF file**

  - A typed written report of no more than 2 pages of text plus figures/tables (not counting towards the page limit). See more details below.
  - Outputs from the handout test scripts: *test1d_lasso.m* and *test2d_lasso.m* (including Matlab printput and the figures generated).
  - (PDF) Copies of the tested 2 solvers that you have selected and used in the test script to generate your test results.

- **Guidelines for the report**

  This report, based on your evaluations and comparisons (see above), will be *an important item* of this project. What will be expected is a level of depth and a degree of sophistication (or even better, a touch of creativity). It should reflect your experience, your assessment, your understanding and your insight about the involved algorithms/codes in the context of solving Lasso. Do not repeat or recite known materials from either this assignment or other sources. Do not describe routine steps that everyone goes through. Report things that are interesting and meaningful. Present claims/conclusions/speculations with well-organized observations or evidence, while avoid over-extrapolating.