DSA5103 Assignment 2

Instructions While all languages are acceptable, it is recommended that you code using Python or MATLAB. You must write your own code. Due on March 12, 11:59pm. Please submit a pdf file including answers for Q1, Q2, Q3(c,d,e,f). Insert the figures for Q3(c,d) into the pdf file, report the results for Q3(e) in the form of Table 1 in the pdf file, and summarize your comparisons for Q3(f) in the pdf file. In addition, you should submit the codes for Q3 (.m file or .ipynb file) that I can run for generating the numerical results reported in the pdf file. No programming is required for Q1 and Q2.

1. **KKT**

   Consider the problem ($x \geq 0$ means $x_i \geq 0$, $i \in [n]$)

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & -\sum_{i=1}^n \log(1+x_i) \\ \text{s.t.} \quad & x \geq 0 \\ & x_1 + \cdots + x_n = 1. \end{aligned}$$

   (a) Check Slater's condition. [1.5 mark]

   (b) Write KKT conditions. [1.5 mark]

   Solution. (a) There exists a strictly feasible point $\hat{x} = (1/n, \ldots, 1/n)^T \in \mathbb{R}^n$ such that $x > 0$ and $x_1 + \cdots + x_n = 1$. Therefore, Slater's condition holds.

   (b) Let $u \in \mathbb{R}^n_+$ and $v \in \mathbb{R}$. The Lagrangian is

$$L(x,u,v) = -\sum_{i=1}^n \log(1+x_i) - \sum_{i=1}^n x_i u_i + (x_1 + \cdots + x_n - 1)v.$$

   The KKT conditions for (x,u,v) are

$$\begin{aligned} & \frac{\partial}{\partial x_i} L(x,u,v) = -\frac{1}{1+x_i} - u_i + v = 0,\ i \in [n], \\ & x \geq 0, \quad x_1 + \cdots + x_n = 1, \\ & u \geq 0, \\ & x_i u_i = 0,\ i \in [n]. \end{aligned}$$

2. **Coordinate descent**

Apply coordinate descent method for

$$\min_{x=(x_1;x_2)\in\mathbb{R}^2} \quad f(x_1,x_2) = 2x_1^2 - 6x_1x_2 + 5x_2^2 - 4x_1 - 3x_2$$

with initial point $x^{(0)} = (0;0)$. Find $x^{(1)}$ and $x^{(2)}$. [1.5 mark for $x^{(1)}$ + 1.5 mark for $x^{(2)}$]

Solution. We obtain that $x^{(1)} = (1;0.9)$ and $x^{(2)} = (2.35;1.71)$ via the computations

$$x_1^{(1)} = \arg\min_{x_1} 2x_1^2 - 4x_1 = 1, \quad x_2^{(1)} = \arg\min_{x_2} 5x_2^2 - 9x_2 = 0.9,$$
$$x_1^{(2)} = \arg\min_{x_1} 2x_1^2 - 9.4x_1 = 2.35, \quad x_2^{(2)} = \arg\min_{x_2} 5x_2^2 - 17.1x_2 = 1.71.$$

Alternative solution. (update x_2 first and then update x_1) We obtain that $x^{(1)} = (1.45;0.3)$ and $x^{(2)} = (2.755;1.17)$ via the computations

$$x_2^{(1)} = \arg\min_{x_2} 5x_2^2 - 3x_2 = 0.3, \quad x_1^{(1)} = \arg\min_{x_1} 2x_1^2 - 5.8x_1 = 1.45,$$
$$x_2^{(2)} = \arg\min_{x_2} 5x_2^2 - 11.7x_2 = 1.17, \quad x_1^{(2)} = \arg\min_{x_1} 2x_1^2 - 11.02x_1 = 2.755.$$

3. **Lasso**

For solving the Lasso problem

$$\min_{\beta\in\mathbb{R}^p} \frac{1}{2}\|X\beta - Y\|^2 + \lambda\|\beta\|_1,$$

we write our own codes to implement the methods: coordinate descent, proximal gradient, accelerated proximal gradient, and accelerated proximal gradient with restart techniques.

(a) Set $n = 1000$, $p = 5000$. Generate an $n \times p$ random matrix X where each entry X_{ij} is a random variable drawn from the standard normal distribution $N(0,1)$, and then standardize each column of X — subtracting its mean and dividing by its standard deviation. Generate a random sparse vector $\beta^* \in \mathbb{R}^p$ with approximately $0.05p$ nonzero entries (by "scipy.sparse.random" in Python or "sprandn" in MATLAB). Let $Y = X\beta^* + 0.01\epsilon$ where $\epsilon_i \sim N(0,1)$, $i \in [n]$ is the Gaussian noise.

Remark: Note that there is a factor 0.01 before ϵ.

(b) Set the penalty parameter be $\lambda = 0.1\|X^TY\|_\infty$. We always use the initial point $\beta^{(0)} = 0$. Let $L = \lambda_{\max}(X^TX)$ ($\lambda_{\max}(\cdot)$ denotes the largest eigenvalue of a matrix), step size $\alpha = 1/L$ for PG and APG. Implement the methods for solving the Lasso problem: coordinate descent (CD), proximal gradient (PG), accelerated proximal gradient (APG), and accelerated proximal gradient with restart techniques (APG-restart)

(one may choose to restart every 100 iterations). The codes should be terminated when it achieves the accuracy of tolerance $= 10^{-3}$ in the relative residual error for an approximate solution $\beta^{(k)}$, namely,

$$r(\beta^{(k)}) := \left\|\beta^{(k)} - S_\lambda\left(\beta^{(k)} - X^T(X\beta^{(k)} - Y)\right)\right\|_2 < \text{tolerance}.$$

Remark: In each iteration of CD, we have the division $\frac{1}{\|X_{\cdot i}\|^2}$. It is time-consuming to compute this value over and over again in each iteration. Instead we can compute and save the value at the very first beginning. In addition, we may compute X^TX and X^TY and store them.

(c) For CD, PG, APG, APG-restart, plot a figure of the relative residual error in base-10 logarithmic scale, i.e., $\log_{10}(r(\beta^{(k)}))$, against iterations k. [2 marks — 0.5 mark for one method achieving a relative residual error $\leq 10^{-3}$ within reasonable iterations]

(d) For CD, PG, APG, APG-restart, plot a figure of the relative residual error in base-10 logarithmic scale, i.e., $\log_{10}(r(\beta^{(k)}))$, against running time t_k, where t_k denotes the time taken for the first k iterations of a particular method. [2 marks — 0.5 mark for one method achieving a relative residual error $\leq 10^{-3}$ within a reasonable time period]

(e) Test your codes for APG and APG-restart to see whether it can achieve the accuracy of tolerance $= 10^{-10}$ in the relative residual error for an approximate solution $\beta^{(k)}$, namely,

$$r(\beta^{(k)}) := \left\|\beta^{(k)} - S_\lambda\left(\beta^{(k)} - X^T(X\beta^{(k)} - Y)\right)\right\|_2 < 10^{-10}$$

within 3000 iterations. Report the relative residual error, iterations, and time for APG and APG-restart in Table 1. [3 marks — 1.5 mark for a successful APG achieving a relative residual error $\leq 10^{-10}$ within 3000 iterations + 1.5 mark for a successful APG-restart achieving a relative residual error $\leq 10^{-10}$ within 1500 iterations]

	relative residual error $r(\beta^{(k)})$	iterations	time (sec)
APG			
APG-restart			

Table 1

(f) Set tolerance $= 10^{-6}$. If the step size $\alpha = 1/L = 1/\lambda_{\max}(X^TX)$ is increased slightly to $\alpha = 1.5/L = 1.5/\lambda_{\max}(X^TX)$. Do your codes for APG and APG-restart run faster, or do they diverge? Report the comparisons of numerical results with different step sizes ($\alpha = 1/L$ vs. $\alpha = 1.5/L$). [2 marks — 1 mark if the four cases all converge + 1 mark if the algorithm is observed to be faster with a larger step size]

Solution.

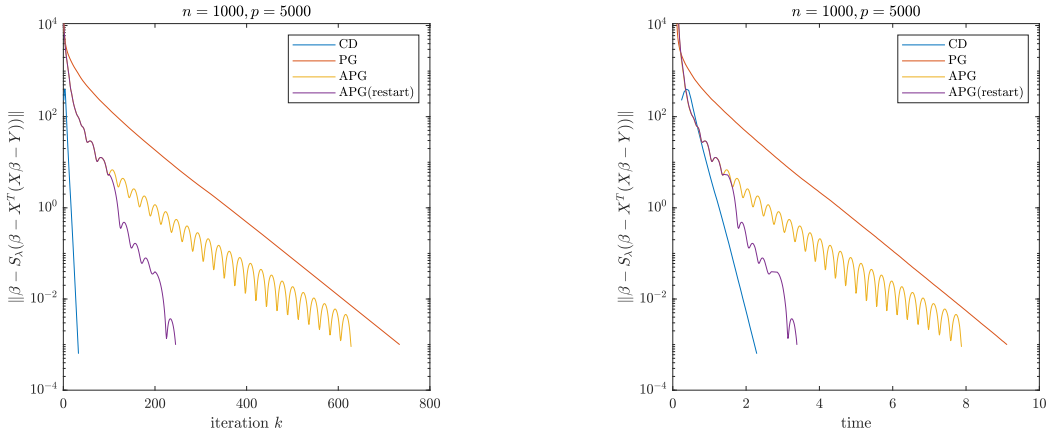

Figure 1: Left: for Q3(c). Right: for Q3(d). APG-restart restarts every 100 iterations.

Q3(e) My codes for APG and APG-restart can achieve the accuracy of tolerance = 10^{-10}. The results are reported in the following table. In addition, see Figure 2 for the progresses of the two methods.

	relative residual error $r(\beta^{(k)})$	iterations	time (sec)
APG	9.12×10^{-11}	1991	25.0
APG-restart	8.36×10^{-11}	590	8.6

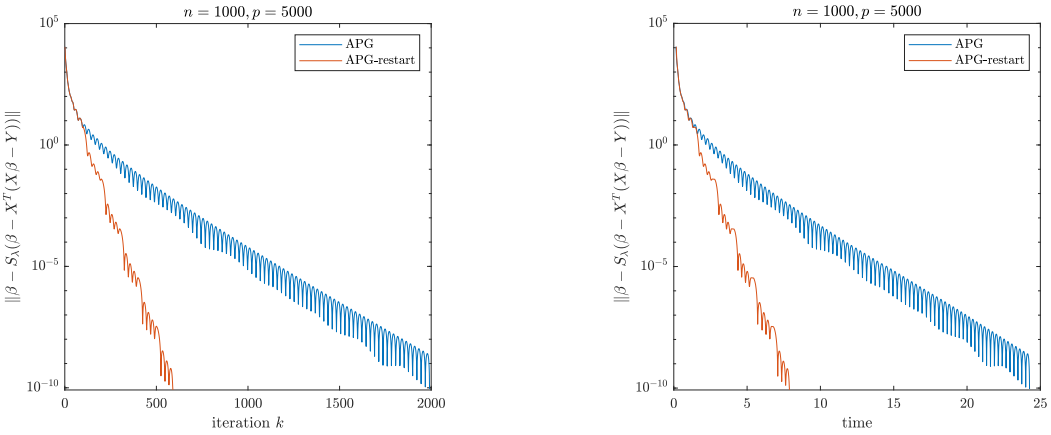

Figure 2

Q3(f) From the following table, we can see that my algorithms are faster if the step size is increased.

	relative residual error $r(\beta^{(k)})$	iterations	time (sec)
APG, $\alpha = 1/L$	8.12×10^{-7}	1183	14.6
APG-restart, $\alpha = 1/L$	8.50×10^{-7}	418	5.8
APG, $\alpha = 1.5/L$	8.98×10^{-7}	813	10.3
APG-restart, $\alpha = 1.5/L$	5.26×10^{-7}	320	4.4