DSA5103
Optimization algorithms for data modelling

National University of Singapore
Data Science and Machine Learning Department
March 12, 2023

Assignment 2

Wang Jiangyi, A0236307J
e0732857@u.nus.edu

1 Question 1, Slater + KKT

$$\begin{aligned} & min_x : -\sum_{i=1}^{n} log(1+x_i) \\ & s.t: \ x_i \geq 0, \quad i \in [n] \\ & \sum_{i=1}^{n} x_i = 1 \end{aligned} \tag{1}$$

1.1 Slater's condition

Firstly, recap that, Slater's condition is, find $\hat{x} \in \mathbb{R}^n$ such that,

$$\begin{cases} \sum_{i=1}^{n} \hat{x}_i = 1 \\ \hat{x}_i > 0, \ i \in [n] \end{cases}$$

This exactly holds since we can choose $\hat{x} = [\frac{1}{n}, ..., \frac{1}{n}]^T \in \mathbb{R}^n$, then we can check,

$$\begin{cases} \sum_{i=1}^{n} \frac{1}{n} = n \times \frac{1}{n} = 1 \\ \hat{x}_i = \frac{1}{n} > 0, \ i \in [n] \end{cases}$$

Therefore, Slater's condition holds.

[This part is not required for this problem] Moreover, Optimization problem (1) is indeed one **Convex Programming**. Based on that, i.e., **convex programming and slater's condition**, it implies that **Strong Duality holds** for Optimization problem (1). Suppose that we make the following notation,

$$\begin{cases} f(x) = -\sum_{i=1}^{n} log(1+x_i) \\ g(x) = \mathbb{I}^T x - 1 \\ h_i(x) = -x_i, \ i \in [n] \end{cases}$$

After simple calculation, we have,

$$\nabla^2 f(x) = Diag\left\{ \frac{1}{(1+x_1)^2}, ..., \frac{1}{(1+x_n)^2} \right\} \succ 0$$

Therefore, we have, $f(\cdot)$ is convex; $g(\cdot)$ is affine; $h_i(\cdot)$ is affine (thus convex). This shows Optimization problem (1) is exactly a **convex programming**, implying that **Strong Duality holds**.

1.2 KKT condition

Denote Lagrangian function as $L(x; u, v)$, then we have:

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

Therefore, KKT condition (at $x = x^*$) with respect to dual variable ($u = u^*, v = v^*$) is,

1. Stationarity Condition:

$$\nabla_{x_i} L(x^*; u^*, v^*) = 0 \implies -\frac{1}{1+x_i^*} - u_i^* - v^* = 0, \ i \in [n]$$

2. Primal Feasibility Condition:

$$\begin{cases} \sum_{i=1}^{n} x_i^* = 1 \\ x_i^* \geq 0, \ i \in [n] \end{cases}$$

3. Dual Feasibility Condition:
$$\begin{cases} u_i^* \geq 0,\ i \in [n] \\ v^* \in \mathbb{R} \end{cases}$$

4. Complementary Slackness Condition:
$$u_i^* x_i^* = 0,\ i \in [n]$$

All (1-4) conditions (Stationarity, Primal Feasibility, Dual Feasibility, Complementary Slackness) contribute to the whole KKT condition.

[This part is not required for this problem] In Section 1.1, we show that, **Strong Duality holds** and **Optimization problem (1) is Convex Programming**. Therefore, it is sufficient to show, $(x^*; u^*, v^*)$ is the optimal solution to $(Primal\ problem; Dual\ problem)$ if and only if, KKT condition is satisfied (at $x = x^*$) with respect to dual variable $(u = u^*, v = v^*)$.

This result builds the connection between 'dual optimal solution' and 'primal optimal solution' with the help of 'KKT condition'.

2 Question 2, Coordinate Descent

$$min_x : f(x_1, x_2) = 2x_1^2 - 6x_1x_2 + 5x_2^2 - 4x_1 - 3x_2 \tag{2}$$

To better apply Coordinate Descent Algorithm, let us find general exact solutions for the following coordinate-wise optimization problem:

$$\begin{cases} \hat{x}_2 = argmin_{x_2} : f(a, x_2) = argmin_{x_2} : 5x_2^2 - (3+6a)x_2 \\ \hat{x}_1 = argmin_{x_1} : f(x_1, b) = argmin_{x_1} : 2x_1^2 - (4+6b)x_1 \end{cases}$$

Based on Convexity of $f(\cdot)$ and Second-order Optimality Condition, we can easily achieve the following update formular of $(\hat{x}_1, \hat{x}_2)$,

$$\begin{cases} \hat{x}_2 = \frac{1}{10} \cdot (3 + 6x_1) \\ \hat{x}_1 = \frac{1}{2} \cdot (2 + 3x_2) \end{cases}$$

Therefore, starting from $x^{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ we have:

$$\begin{cases} x_1^{(1)} = \frac{1}{2} \cdot (2 + 3x_2^{(0)}) = \frac{1}{2} \cdot 2 = 1 \\ x_2^{(1)} = \frac{1}{10} \cdot (3 + 6x_1^{(1)} = \frac{1}{10} \cdot 9 = \frac{9}{10}) \end{cases}$$

$$\begin{cases} x_1^{(2)} = \frac{1}{2} \cdot (2 + 3x_2^{(1)}) = \frac{1}{2} \cdot \frac{47}{10} = \frac{47}{20} \\ x_2^{(2)} = \frac{1}{10} \cdot (3 + 6x_1^{(2)} = \frac{1}{10} \cdot \frac{171}{10} = \frac{171}{100}) \end{cases}$$

To conclude, we attain: $x^{(1)} = \begin{bmatrix} 1 \\ \frac{9}{10} \end{bmatrix}$, $x^{(2)} = \begin{bmatrix} \frac{47}{20} \\ \frac{171}{100} \end{bmatrix}$

3 Question 3, LASSO

$$min_\beta : \frac{1}{2}||X\beta - Y|| + \lambda||\beta||_1 \tag{3}$$

Here, $X \in \mathbb{R}^{n \times p}$, $\beta \in \mathbb{R}^p$ and $n = 1000$, $p = 5000$. The ground truth label Y is generated by,

$$Y = X\beta^* + 0.01\epsilon$$

β^* is a sparse vector, with only 5% non-zero entries. $X_{ij} \sim N(0, 1)$ and $\epsilon \sim N(\mathbf{0}_n, \mathbb{I}_n)$.

3.1 Relative Residual Error v.s. Iteration Number

The result can be shown as follows:

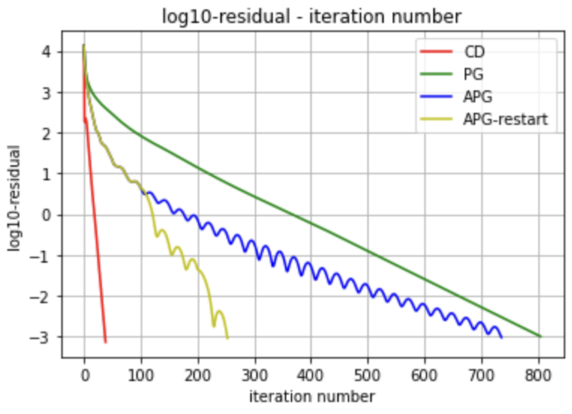


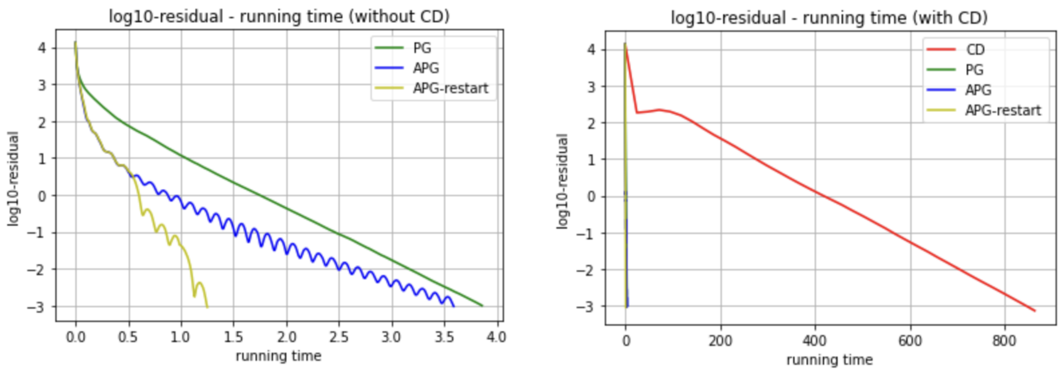
Figure 1: Relative Residual Error v.s. Iteration Number

Explanation:

1. CD method achieves '1e-3' relative residual error with the smallest iteration number. Possible reason is, CD method solves **exact** coordinate-wise minimization problem. From this perspective, CD method will not twist orginal objective too much. This also leads to the **strict decrease** for residual error.
2. As for PG method, the residual error also **decreases strictly**. That is because, PG method can be viewed as one variation of GD method. Suppose that the step length is small enough, then we can guarantee the strict decrease in GD framework. For APG and APG-restart method, **this never happen**s due to the momentum term. However, the momentum term can accelerates the algorithm convergence rate.
3. Compared with APG, APG-restart converges faster since the restart mechanism. As the training goes on, $t_k \to \infty$ and we utilzie too many $\Delta\beta$ information, which will slow down the convergence. This is why we need to restart the t_k to accelerate the convergence.
4. In the beginning stage of APG and APG-restart, it is almost the same with PG since the weight for $\Delta\beta$ term is extremely small.

3.2 Relative Residual Error v.s. Running Time

The result is shown as follows:



(a) without CD method (b) with CD method

Figure 2: Relative Residual Error v.s. Running Time

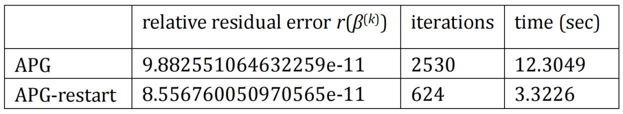
Observation:

1. For PG, APG and APG-restart algorithm, the running time is approximately propotional to the iteration

number. Moreover, compared with CD method, these 3 algorithms are extremely fast, at the cost of more iteration number.
2. For CD method, it is extremely slow. Reason is, although it requires small number of iteration to achieve '1e-3' residual error, within each iteration, it requires to update each entry of β sequentially (cannot parallelize).

3.3 APG, APG-restart Examination

The report of relative residual error, iteration and time of APG, APG-restart is shown as follows,



	relative residual error $r(\beta^{(k)})$	iterations	time (sec)
APG	9.882551064632259e-11	2530	12.3049
APG-restart	8.556760050970565e-11	624	3.3226

Figure 3: Relative Residual Error, Iteration, Time for APG, APG-restart

To conclude, for both APG and APG-restart, we are able to achieve '1e-10' relative residual error within 3000 iteration. However, APG-restart is faster than APG.

3.4 Choices of step length α

The result is, for both $\alpha = \frac{1}{L}$ and $\alpha = \frac{1.5}{L}$, APG and APG-restart will converge. Moreover, for $\alpha = \frac{1.5}{L}$, algorithms (both APG and APG-restart) will converge faster! The diagram can be shown as follows,

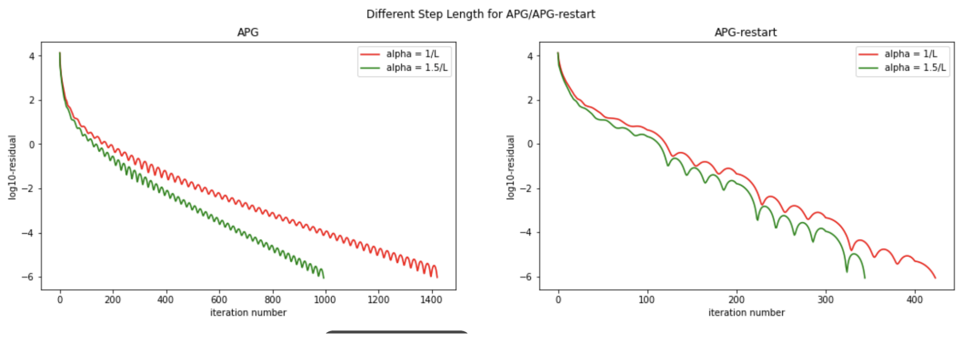
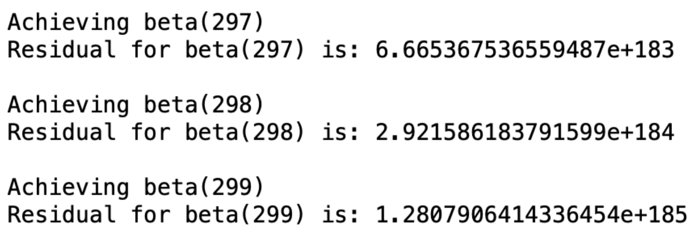


Figure 4: Different Step Length α for APG/APG-restart

From Figure 4, it can be observed that, after increasing α from $\frac{1}{L}$ to $\frac{1.5}{L}$ (1.5 times), we only require $\frac{1}{1.5} = \frac{2}{3}$ times iteration.

However, this does not mean we can arbitrarily enlarge the α, which will make the algorithm diverge. Here, we make one experiment, setting $\alpha = \frac{3}{L}$. Then the residual error starts to diverge, which can be shown as follow,



```
Achieving beta(297)
Residual for beta(297) is: 6.665367536559487e+183

Achieving beta(298)
Residual for beta(298) is: 2.921586183791599e+184

Achieving beta(299)
Residual for beta(299) is: 1.2807906414336454e+185
```

Figure 5: $\alpha = \frac{3}{L}$ for APG-restart Method

This illustrates that, when setting $\alpha = \frac{3}{L}$, after 300 iteration, the Relative Residual Error grows to 1e185. **That is to say, the algorithm will diverge when step length is large enough.**