Homework #3

ELEC/IEDA3180 - Data Driven Portfolio Optimization Spring 2022/2023

Prof. Daniel Palomar, TAs: Amir Javaheri, Yifan Yu

2023-04-03

The HKUST Academic Honor Code applies.
This assignment is to be done individually.
Cheating **won't** be tolerated. Total marks:
130 points.

Problem 1 (70 points)

Consider the following quadratic problem:

$$\min_{x_1, x_2} \quad f(x_1, x_2) \triangleq 2x_1^2 + 3x_2^2 - 2x_1x_2 - 6x_1 - 12x_2 \quad (1)$$

(a) Let $\mathbf{x} = (x_1, x_2)^\top$. Then find $\mathbf{A} \in \mathbb{R}^{2\times 2}$ and $\mathbf{b} \in \mathbb{R}^2$ such that the problem can be restated as follows

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) \triangleq \frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x} - \mathbf{b}^\top \mathbf{x} \quad (2)$$

(b) Use the Gradient Descent method to solve problem (2) for the following choices of the step size parameter:

1. Constant step size $t^k = 0.1$;

2. Exact line search;

3. Backtracking line search
   $(\alpha = 0.2, \beta = 0.8)$.

Choose $\mathbf{x}^0 = (0, 0)$ as the initial point.

(c) Use the Newton's method to solve problem (2) with a constant step size $t^k = 1$ (with the same initial point).

(d) The Conjugate Gradient method is an acceleration method based on conjugate descent directions (denoted with $\mathbf{p}^k$ in the following algorithm). This method uses an adaptive step-size t^k for the updates via the following formulation:

Conjugate Gradient Algorithm:

Initialization:

i. Choose $\mathbf{x}^0$ and $\epsilon \ll 1$, and let $k = 0$.

ii. Let $\mathbf{p}^0 = \mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$

Repeat:

1. Let $t^k = \dfrac{\langle \mathbf{r}^k, \mathbf{r}^k \rangle}{\langle \mathbf{p}^k, \mathbf{A}\mathbf{p}^k \rangle}$
2. Update $\mathbf{x}^{k+1} = \mathbf{x}^k + t^k \mathbf{p}^k$

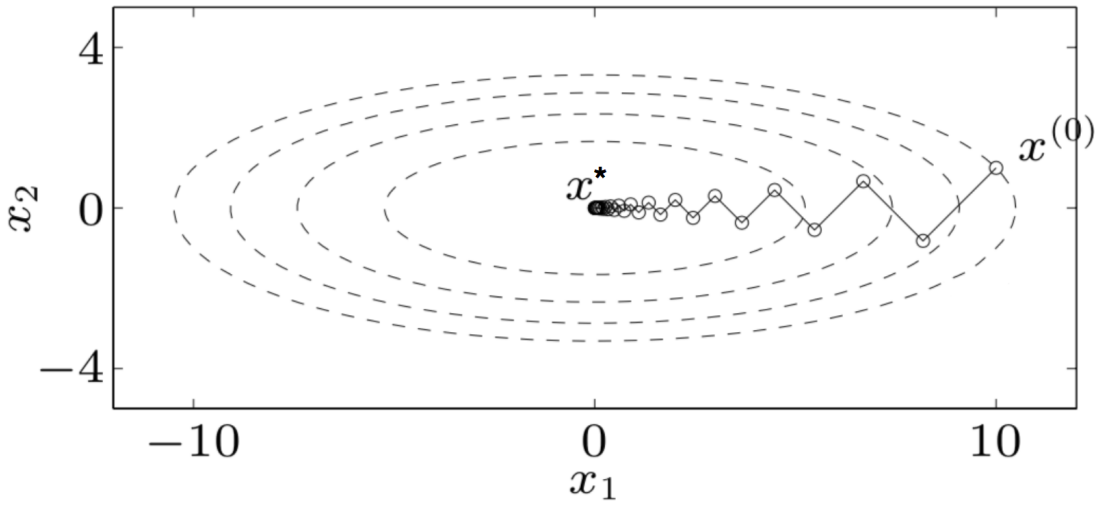
3. Update $\mathbf{r}^{k+1} = \mathbf{r}^k - t^k \mathbf{A}\mathbf{p}^k$
4. If $\left\|\mathbf{r}^{k+1}\right\| < \epsilon$ exit the loop
5. Let $\gamma^k = \dfrac{\left\langle \mathbf{r}^{k+1}, \mathbf{r}^{k+1} \right\rangle}{\left\langle \mathbf{r}^k, \mathbf{r}^k \right\rangle}$
6. Update $\mathbf{p}^{k+1} = \mathbf{r}^{k+1} + \gamma^k \mathbf{p}^k$
7. Update $k = k + 1$

Return $\mathbf{x}^{k+1}$.

Use the Conjugate Gradient method to solve problem (2) starting from $\mathbf{x}^0 = (0, 0)^\top$.

(e) Calculate the optimal solution $\mathbf{x}^\star$ of problem (2) and plot the convergence figures (the value of $f(\mathbf{x}^k) - f(\mathbf{x}^\star)$ vs. the iteration number k) for all of the algorithms above (a)-(d). Which algorithm has the fastest rate of convergence and why?

(f) (Bonus) Plot the paths of all the algorithm in (a)-(d). You should plot a figure like the one below showing the contours of the function $f(\mathbf{x})$ and the path of each algorithm. Use different colors and labels for readability.



A sample figure showing the covergence path

Problem 2 (60 points)

As you saw in your midterm exam, in the risk parity portfolio (RPP), you have to solve the following convex optimization problem

$$\min_{\mathbf{x}\geq\mathbf{0}} \quad g(\mathbf{x}) \triangleq \frac{1}{2}\mathbf{x}^\top\boldsymbol{\Sigma}\mathbf{x} - \mathbf{b}^\top \log(\mathbf{x}) \quad (3)$$

where $\log(\mathbf{x}) = (\log(x_1), \log(x_2), \ldots, \log(x_N))^\top$, $\mathbf{b}$ is the risk budget vector and $\boldsymbol{\Sigma}$ is the covariance matrix of the log-returns.

In this problem we assume $N = 3$ and

$$\boldsymbol{\Sigma} = \begin{bmatrix} 1.0 & 0.02 & -0.04 \\ 0.02 & 1.0 & 0.02 \\ -0.04 & 0.02 & 1.0 \end{bmatrix}$$

$$\mathbf{b} = \frac{\mathbf{1}}{3}$$

(a) Use the Jacobi algorithm to solve problem (3) (derive the update formula for each element x_i of $\mathbf{x}$, assuming the other elements to be constant).

Note: In Jacobi algorithm, the update $\mathbf{x}^{k+1}$ in $k+1$-th iteration is obtained after computing all x_i^{k+1} updates in parallel.

(b) (Bonus) Find θ such that $\theta\mathbf{I} - \mathbf{\Sigma} \succeq 0$ (positive semi-definite). Then prove that $u(\mathbf{x}, \mathbf{y})$ defined below, is a surrogate majorization function for $g(\mathbf{x})$.

$$u(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\mathbf{x}^\top \mathbf{\Sigma}\mathbf{x} + \frac{1}{2}(\mathbf{x}-\mathbf{y})^\top(\theta\mathbf{I} - \mathbf{\Sigma})(\mathbf{x}-\mathbf{y}) - \mathbf{b}^\top \log(\mathbf{x})$$

(c) Use the Majorization Minimization algorithm (MM) to solve the problem via the following iterations (choose $\theta = 1.2$)

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}\geq \mathbf{0}}{\mathsf{argmin}} \;\; u(\mathbf{x}, \mathbf{x}^k)$$

Hint: Rewrite $u(\mathbf{x}, \mathbf{x}^k)$ as follows

$$u(\mathbf{x}, \mathbf{x}^k) = \frac{\theta}{2}\left\|\mathbf{x} - \mathbf{x}^k + \frac{1}{\theta}\mathbf{\Sigma}\mathbf{x}^k\right\|^2 - \mathbf{b}^\top \log(\mathbf{x}) + h(\mathbf{x}^k)$$

(d) Plot the convergence figures of the above methods versus time. The vertical axis should be $g(\mathbf{x}^k) - g(\mathbf{x}^\star)$ and the horizontal axis should be t_k (the time taken to compute $\mathbf{x}^k$ in seconds, starting from zero at $k=0$). You may obtain $\mathbf{x}^\star$ using `cvxpy`.

Note #1: Make sure to write your code in a modular, readable way. Code organization will be taken into account for the grading. Use meaningful names for variables, create functions to organize your code as much as possible. In case of doubt on coding best practices, take a look at the Google's style guide. Note that you don't have to strictly follow that style, you can develop your own, but make sure it is understandable and you use it consistently.

Note #2: Submit your code via canvas in zipped file (`.zip`) containing a Jupyter notebook (`.ipynb`) and also its exported version in the HTML (`.html`) format. Submissions with missing `.ipynb` file won't be graded.