Homework Assignment 1:

Implementation of Basic Large and Huge Scale Optimization Algorithms

ELEC-E5431 – Large Scale Data Analysis (LSDA)

- Return your report preferably in pdf-format. Recommended deadline to return your report is by **Jan. 28, 2023** through MyCourse. (You will receive Homework Assignment 2 few days before Jan. 28, and will be focuses on the new assignment then. Thus, it is strongly recommended to return your homework solutions well before the recommended deadline and do not delay it after the recommended deadline. It is your interest to go smoothly with other homework assignments!)
- Enclose you codes as well!!!
- In subject line: write ELEC-E5431, your name and student ID.
- Show all the steps of your work. Your reasoning is the most important component for grading!
- It is preferable if you code in Python or MATLAB, but it is not a limitation.

Problem Setup:

The optimization problem to be addressed is a simple quadratic function minimization, that is,

$$\min_{\mathbf{x}} \; \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$$

where the matrix $\mathbf{A}$ and vector $\mathbf{b}$ are appropriately generated. Use the same $\mathbf{A}$ and $\mathbf{b}$ while comparing different methods.

Hints:

- $\mathbf{A}$ should be such that $\mathbf{x}^T\mathbf{A}\mathbf{x}$ is non-negative, that is, $\mathbf{A}$ is positive semi-definite, and $\mathbf{b}$ should be in the range of $\mathbf{A}$.

- In fact, by solving the above unconstrained minimization problem, you solve a system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{b}$. Indeed, the gradient of the objective function is $\mathbf{A}\mathbf{x} - \mathbf{b}$, and it should be equal to 0 at optimality. Thus, you can find optimal $\mathbf{x}^*$ using back-slash (or matrix inversion followed by computing the product $\mathbf{A}^{-1}\mathbf{b}$) operators in MATLAB. The optimal objective value can be then obtained by simply substituting such $\mathbf{x}$ into the objective of the above optimization problem. It is suitable for small and mid size problems, but the matrix inversion is prohibitively too expensive to be able to solve a system of linear equations for large scale problems. Thus, the only option for large scale problems is the use of algorithms that you implement in this assignment!

To be able to produce convergence figures for the algorithms that you test, let the dimension of $\mathbf{x}$ be 100 variables or few 100's (but after producing the figures also play with higher dimensions to see when the matrix inversion fails, but the large scale optimization methods still work fine and some also quite fast).

Set the tolerance parameter for the stopping criterion for checking the convergence to 10^{-5}. For example, check if $\|\nabla f(\mathbf{x})\| \leq 10^{-5}$, and limit the total number of iterations by 5000 if the predefined tolerance is still not achieved.

Task 1: *Gradient Descent Algorithm.*

Implement Gradient Descent Algorithm for solving the above optimization problem. Use correctly selected fixed step size α. Draw the experimental convergence rate, i.e., draw the plot $\log \|\mathbf{x}_k - \mathbf{x}^*\|_2$ versus iteration number k, for the algorithm and compare it with the theoretically predicted one.

Task 2: *Conjugate Gradient Algorithm.*

Implement Conjugate Gradient Algorithm for solving the above optimization problem. Use correctly selected fixed step size. Draw the experimental convergence rate, i.e., draw the plot $\log \|\mathbf{x}_k - \mathbf{x}^*\|_2$ versus iteration number k, for the algorithm and compare it with the theoretically predicted one.

Task 3: *FISTA.*

Implement FISTA for solving the above optimization problem. Use correctly selected fixed step size. Draw the experimental convergence rate, i.e., draw the plot $\log \|\mathbf{x}_k - \mathbf{x}^*\|_2$ versus iteration number k, for the algorithm and compare it with the theoretically predicted one.

Task 4: *Coordinate Descent.*

Implement the Coordinate Descent method for solving the above optimization problem. Use correctly selected fixed step size. Draw the experimental convergence rate, i.e., draw the plot $\log \|\mathbf{x}_k - \mathbf{x}^*\|_2$ versus iteration number k, for the algorithm.

Task 5: *Comparisons.*

Compare the results (in terms of the iterations required and the overall computation time) for different methods (including, for example, the standard MATLAB back-slash operator) and draw your overall conclusions. Observe up to which dimension Python or MATLAB still can invert a matrix, that is, define the dimension after which the problem turns to be large scale in the context of your implementation.