Instructions: (Please read carefully and follow them!)

Try to solve all problems on your own. If you have difficulties, ask the instructor or TAs.

In this session, we will continue with the implementation of gradient descent algorithm to solve problems of the form $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$. Recall that gradient descent takes a large number of iterations for some problems. In this lab, we will see some techniques to make gradient descent converge to the optimal point faster. We shall investigate the behavior of Newton's method on some problems and compare its performance against gradient descent algorithm. We will also discuss BFGS method to solve problems of the form $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$.

The implementation of the optimization algorithms in this lab will involve extensive use of the `numpy` Python package. It would be useful for you to get to know some of the functionalities of `numpy` package. For details on `numpy` Python package, please consult https://numpy.org/doc/stable/index.html

For plotting purposes, please use `matplotlib.pyplot` package. You can find examples in the site https://matplotlib.org/examples/.

Please follow the instructions given below to prepare your solution notebooks:

- Please use different notebooks for solving different Exercise problems.
- The notebook name for Exercise 1 should be `YOURROLLNUMBER_IE684_Lab04_Ex1.ipynb`.
- Similarly, the notebook name for Exercise 2 should be `YOURROLLNUMBER_IE684_Lab04_Ex2.ipynb`, etc and so on.

There are only 3 exercises in this lab. Try to solve all the problems on your own. If you have difficulties, ask the Instructors or TAs.

You can either print the answers using `print` command in your code or you can write the text in a separate text tab. To add text in your notebook, click `+Text`. Some questions require you to provide proper explanations; for such questions, write proper explanations in a text tab. Some questions require the answers to be written in LaTeX notation. (**Write the comments and observations with appropriate equations in LaTeX only.**) Some questions require plotting certain graphs. Please make sure that the plots are present in the submitted notebooks.

After completing this lab's exercises, click File → Download .ipynb and save your files to your local laptop/desktop. Create a folder with name `YOURROLLNUMBER_IE684_Lab04` and copy your `.ipynb` files to the folder. Then zip the folder to create `YOURROLLNUMBER_IE684_Lab04.zip`. Then upload only the `.zip` file to Moodle. **There will be some penalty for students who do not follow the proper naming conventions in their submissions**.

Please check the **submission deadline announced in moodle.**

The fourth Laboratory exercise aims to help you learn the **Quasi Newton (BFGS)** Application and **Gradient Descent with momentum, Proximal and Conjugate Gradient** methods.

Exercise 1 (20 marks) *In the last labs, when we tried to solve certain problems of the form* $\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$ *using gradient descent algorithm, we noticed that we are updating the point at all the coordinates and here we will see that only one coordinate of the point will be updated. We will also see how to apply gradient descent when we have constrained optimization problem to solve for.*

Algorithm 1 Coordinate Descent with Line Search

Require: Starting point x_0, Stopping tolerance τ

Initialize $k = 0$
while $\|\nabla f(x_k)\|_2 > \tau$ **do**
    Choose a random coordinate i
    Perform line search to find step size $\eta_k^{(i)}$
    Update $x_{k+1}^{(i)} = x_k^{(i)} - \eta_k^{(i)} \frac{\partial f}{\partial x^{(i)}}(x_k)$
    $k = k + 1$
Output: x_k

Algorithm 2 Proximal Gradient Descent

Require: Initial point $\mathbf{x}_0$, step size $\eta > 0$, regularization parameter $\lambda > 0$, maximum iterations T

Initialize $t = 0$
while $t < T$ **do**
    Compute gradient $\nabla f(\mathbf{x}_t)$
    Update $\mathbf{y}_t = \mathbf{x}_t - \eta \nabla f(\mathbf{x}_t)$
    Compute proximity operator: $\mathbf{x}_{t+1} = \text{prox}_{\lambda g}(\mathbf{y}_t)$ ▷ Proximity operator of g with parameter λ
    $t = t + 1$
Output: x_t

Proximity operator of g with parameter λ *can be defined as:*

$prox_{\lambda g}(\mathbf{y}) = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left(g(\mathbf{x}) + \frac{1}{2\lambda}\|\mathbf{y} - \mathbf{x}\|^2\right)$.

Consider the function $f(\mathbf{x}) = f(x_1, x_2) = x_1^2 + 4x_1x_2 + 4x_2^2$.

$$g(\mathbf{x}) = \mathbf{1}_L(\mathbf{x})$$
$$\mathbf{1}_L(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \in L \ , \\ \infty & \text{if } \mathbf{x} \notin L \ . \end{cases} \tag{1}$$

Where L is some closed convex set.

1. *What is the minimizer and minimum function value of* $f(\mathbf{x})$ *?, Is the minimizer unique ?, Is it local or global minima ?, Are the functions* $f(\mathbf{x})$ *and* $g(\mathbf{x})$ *convex ?, explain each of them.*

2. *Can you implement* ***Algorithm 1*** *of this lab using* ***Algorithm 1*** *of Lab 03 ?, Can you come up with a useful choice for* $\mathbf{D}_k$ *?, Does* $\mathbf{D}_k$ *satisfies the condition of being PSD or PD ?, Implement all this for function* $f(\mathbf{x})$*, With starting point* $x_0 = (1, 40)$ *and* $\tau = 10^{-12}$*, we will now study the behavior of coordinate descent algorithm with backtracking line search, for different choices of* ρ*. Take* $\alpha = 1, \gamma = 0.5$*. Try* $\rho \in \{0.9, 0.8, 0.75, 0.6, 0.5, 0.4, 0.25, 0.1, 0.01\}$*.For each* ρ*, record the final minimizer, final objective function value and number of iterations to terminate, for the coordinate descent algorithm with backtracking line search. Prepare a plot where the number of iterations for the algorithm are plotted against* ρ *values. Use different colors and a legend to distinguish the plots. Comment on the observations. Comment about the minimizers*

*and objective function values obtained for different choices of the ρ values. Plot the level sets of the function $f(\mathbf{x})$ and also plot the trajectory of the optimization on the same plot and report your observations. Explain the differences (you may try to implement **Algorithm 1** of this lab as well !!) between **Algorithm 1** of this lab and **Algorithm 1** of Lab 03 and explain which is superior over other and when ?*

3. *Consider any general function $h(\mathbf{y})$ ($\mathbf{y} \in \mathbb{R}^d$), denote its linearization around a fixed point $\mathbf{x}$ by $t(\mathbf{y})$. Evaluate $prox_{\lambda t}(\mathbf{x})$. What is the result ?, Is the result that you got looks familiar to you ? Explain in detail.*

4. *Simplify $prox_{\lambda g}(\mathbf{y})$ with $g(\mathbf{x})$ as defined in equation (1) and use that in **Algorithm 2**. Take $L = \{\boldsymbol{x} \in \mathbb{R}^d; x_i \geq 0 \forall i = 1, ..., d\}$ and further simplify it. Now after all these simplifications implement **Algorithm 2** to solve $\min_{\mathbf{x}\in\mathbb{L}} f(\mathbf{x}) = \min_{\mathbf{x}\in\mathbb{L}} (x_1 - 1)^2 + x_2^2 + (x_3 + 1)^2$, take $\mathbf{x}_0 = (0, 0, 0), \eta = 0.3$ and,*
try $T \in \{10^2, 500, 10^3, 5000, 10^4, 50000, 10^5, 500000, 10^6, 5000000\}$. For each T, record the final minimizer, final objective function value and percentage error between practical and theoretical optimal objective function value at termination. Prepare a plot where the percentage error between practical and theoretical optimal objective function value from the algorithm are plotted against T values. Use different colors and a legend to distinguish the plots. Comment on the observations. Comment about the minimizers and objective function values obtained for different choices of the T values. Plot the level sets of the function $f(\mathbf{x})$ and also plot the trajectory of the optimization on the same plot and report your observations. Will gradient descent with $\eta = 0.7$ gives the same solution to the problem that we considered in this task ?

5. *Simplify $prox_{\lambda g}(\mathbf{y})$ with $g(\mathbf{x})$ as defined in equation (1) and use that in **Algorithm 2**. Take $L = \{\boldsymbol{x} \in \mathbb{R}^d; \|\boldsymbol{x}\|_2 \leq 1\}$ and further simplify it. Now after all these simplifications implement **Algorithm 2** to solve $\min_{\mathbf{x}\in\mathbb{L}} f(\mathbf{x}) = \min_{\mathbf{x}\in\mathbb{L}} 100(x_2 - x_1^2)^2 + (0.5 - x_1)^2$, take $\mathbf{x}_0 = (0, 0), \eta = 0.3$ and,*
try $T \in \{10^2, 500, 10^3, 5000, 10^4, 50000, 10^5, 500000, 10^6, 5000000\}$. For each T, record the final minimizer, final objective function value and percentage error between practical and theoretical optimal objective function value at termination. Prepare a plot where the percentage error between practical and theoretical optimal objective function value from the algorithm are plotted against T values. Use different colors and a legend to distinguish the plots. Comment on the observations. Comment about the minimizers and objective function values obtained for different choices of the T values. Plot the level sets of the function $f(\mathbf{x})$ and also plot the trajectory of the optimization on the same plot and report your observations. Will gradient descent with $\eta = 0.7$ gives the same solution to the problem that we considered in this task ?

6. *What type of convergence or divergence did you saw in Part 02 and Part 04 of Exercise 02 from the last lab i.e. Lab 03, when you implemented the Newton's method ? Give the set of initial points(x_0) for which we will see convergence in Exercise 02 from the last lab i.e. Lab 03 for Newton's method. Give the set of initial points(x_0) for which we will see divergence in Exercise 02 from the last lab i.e. Lab 03 for Newton's method.*
Redo Part 05 to solve-:
$\min_{\mathbf{x}\in\mathbb{L}} f(\mathbf{x}) = \min_{\mathbf{x}\in\mathbb{L}} \sin(5x_2 - 5) \exp\left((1 - \cos(5x_1 - 5))^2\right) + \cos(5x_1 - 5) \exp\left((1 - \sin(5x_2 - 5))^2\right) + 25(x_1 - x_2)^2$.

Exercise 2 (15 marks) *In the last labs we saw that Newton's method is computationally expensive due to Hessian matrix calculations, while basic gradient descent may converge slowly and requires manual tuning. We want to see how can we converge fast with having some momentum involved in our algorithm and also will see how to get rid of step size selection such that algorithm itself learn or takes optimal step size. We will also see how to solve the quadratic optimization problem for finding their minimum when we have large scale problem at hand, where variables involved are in some orders of the magnitude ($\approx 10^5$ and more). In this exercise we will see AdaGrad, NAGD and CG algorithms. Some of you asked for so called* ***Motivation and Applications*** *and hence Adaptive Gradient (AdaGrad) automates learning rates, Nesterov Accelerated Gradient Descent (NAG) with corrected momentum aids in convergence, and Conjugate Gradient (CG) with efficiency in large-scale problems addresses these drawbacks. These advanced algorithms offer tailored solutions, providing adaptability, improved convergence, and computational efficiency, making them preferable for* ***optimizing complex machine learning models****. Hopefully now after solving this exercise and other as well, you will be able to appreciate the contents that we are discussing.*

Algorithm 3 Nesterov Accelerated Gradient Descent (NAGD)

Input: Initial point x_0, step size α_k, momentum parameter β_k, tolerance τ
Initialize $x_{-1} = x_0$
Set iteration counter: $t = 0$
while $\|\nabla f(x_t)\|_2 > \tau$ **do**
    Compute gradient at perturbed point: $\nabla f_{\text{pert}}(x_k) = \nabla f(x_k + \beta_k(x_k - x_{k-1}))$
    Update parameters: $x_{k+1} = x_k - \alpha_k \nabla f_{\text{pert}}(x_k) + \beta_k(x_k - x_{k-1})$
    Increment iteration counter: $t = t + 1$
Output: x_t

Algorithm 4 AdaGrad (Adaptive Gradient Algorithm)

Input: Initial point x_0, Constrained set Ω, maximum iterations T
for $k = 1, 2, \ldots, T$ **do**
    Compute step size: $\alpha_k = \frac{R}{\sqrt{\sum_{s=1}^{k} \|\nabla f(x_s)\|_2^2}}$ ▷ $R = \max_{\mathbf{x},\mathbf{y}\in\Omega} \|\mathbf{x} - \mathbf{y}\|_2$
    Update: $x_{k+1} = \mathcal{P}_\Omega[x_k - \alpha_k \nabla f(x_k)]$ ▷ $\mathcal{P}_\Omega(x) = \frac{1}{2} \arg\min_{z\in\Omega} \|\mathbf{z} - \mathbf{x}\|_2^2$
Output: x_T

Algorithm 5 Conjugate Gradient (CG)

Input: Initial guess x_0, tolerance τ
Initialize variables: $r_0 = -\nabla f(x_0)$, $d_0 = r_0$, $i = 0$
while $\|r_i\| > \tau$ **do**
    Compute step size: $\alpha_i \leftarrow \frac{r_i^T r_i}{d_i^T \nabla^2 f(x_i) d_i}$
    Update parameters: $x_{i+1} \leftarrow x_i + \alpha_i d_i$
    Compute new gradient: $r_{i+1} \leftarrow -\nabla f(x_{i+1})$
    Compute beta value: $\beta_{i+1} \leftarrow -\frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}$
    Update search direction: $d_{i+1} \leftarrow r_{i+1} + \beta_{i+1} d_i$
    Increment iteration counter: $i \leftarrow i + 1$
Output: x_i

Consider the functions

$$q_1(\mathbf{x}) = q_1(x_1, x_2) = \frac{1}{2}\mathbf{x}^\top W \mathbf{x} - b_1^\top \mathbf{x}.$$

$$q_2(\mathbf{x}) = q_2(x_1, x_2) = \frac{1}{2}\mathbf{x}^\top A \mathbf{x} - b_2^\top \mathbf{x}.$$

$$W = \begin{bmatrix} t & \sqrt{t} \\ \sqrt{t} & 1+t \end{bmatrix}, \quad b_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, A = \begin{bmatrix} 4 & 1 \\ 1 & 3 \end{bmatrix}, \quad b_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

1. *What is the minimizer and minimum function value of the functions $q_1(\mathbf{x})$, $q_2(\mathbf{x})$?, Is the minimizer unique for both the functions ?, Is it local or global minima for both the functions ?, Are the function $q_1(\mathbf{x})$, $q_2(\mathbf{x})$ convex ?, explain each of them. (Assume $t \in (0, 1]$ throughout this exercise.)*

2. *Implement Nesterov's accelerated gradient descent algorithm on $q_1(\mathbf{x})$ with step sizes $\alpha_k = \frac{2}{3+\sqrt{9-4t^2}}, \beta_k = \frac{\sqrt{\mu_0}-1}{\sqrt{\mu_0}+1}, \mu_0 = \frac{3+\sqrt{9-4t^2}}{3-\sqrt{9-4t^2}}$. Report and analyze your observations clearly. Take $\tau = 10^{-8}, x_0 = (3, 5), t = 0.001$. Plot the log error of the functional values versus the number of iterations and compare it with the gradient descent algorithm with step-size $\frac{2}{3+\sqrt{9-4t^2}}$. Report and analyze your observations clearly.*

3. *Implement Nesterov's accelerated gradient descent algorithm on $q_2(\mathbf{x})$ with step sizes $\alpha_k = \frac{2}{7+\sqrt{5}}, \beta_k = \frac{\sqrt{\mu_0}-1}{\sqrt{\mu_0}+1}, \mu_0 = \frac{7+\sqrt{5}}{7-\sqrt{5}}$. Report and analyze your observations clearly. Take $\tau = 10^{-8}, x_0 = (3, 5)$. Plot the log error of the functional values versus the number of iterations and compare it with the gradient descent algorithm with step-size $\frac{2}{7+\sqrt{5}}$. Report and analyze your observations clearly.*

4. *Take $\Omega = \{\boldsymbol{x} \in \mathbb{R}^d; \|\boldsymbol{x}\|_2 \leq 1\}$. Implement **Algorithm 4** to solve $\min_{\mathbf{x}\in\Omega} f(\mathbf{x}) = \min_{\mathbf{x}\in\Omega} 100(x_2 - x_1^2)^2 + (0.5 - x_1)^2$, take $\mathbf{x}_0 = (0, 0)$ and, try $T \in \{10^2, 500, 10^3, 5000, 10^4, 50000, 10^5, 500000, 10^6, 5000000\}$. For each T, record the final minimizer, final objective function value and percentage error between practical and theoretical optimal objective function value at termination. Redo this part to solve-:*
$\min_{\mathbf{x}\in\Omega} f(\mathbf{x}) = \min_{\mathbf{x}\in\Omega} \sin(5x_2 - 5) \exp\left((1 - \cos(5x_1 - 5))^2\right) + \cos(5x_1 - 5) \exp\left((1 - \sin(5x_2 - 5))^2\right) + 25(x_1 - x_2)^2$.

5. *Implement Conjugate gradient algorithm on $q_2(\mathbf{x})$. Report and analyze your observations clearly. Take $\tau = 10^{-8}, x_0 = (5, 3)$. Plot the log error of the functional values versus the number of iterations and compare it with the gradient descent algorithm with step-size $\frac{2}{7+\sqrt{5}}$.*

6. *Implement Conjugate gradient algorithm on $q_1(\mathbf{x})$. Report and analyze your observations clearly. Take $\tau = 10^{-8}, x_0 = (5, 3), t = 0.001$. Plot the log error of the functional values versus the number of iterations and compare it with the gradient descent algorithm with step-size $\frac{2}{3+\sqrt{9-4t^2}}$. Redo this part for $t \in \{10^{-1}, 10^{-2}, 10^{-4}, 10^{-5}\}$.*

Exercise 3 (15 marks) *Suppose that y is a noisy version of $A\bar{x}$. We will now try to estimate $\bar{x}$ assuming that we are given y and A. One possible approach is to solve the following problem:*

$$minimize_{\mathbf{x}} \quad f(\mathbf{x}) = \frac{1}{2}\|A\mathbf{x} - \mathbf{y}\|_2^2$$

The loss term $\|A\mathbf{x}-\mathbf{y}\|_2^2$ is called the ordinary least squares (OLS) loss, and the problem is called the OLS Regression problem. This is numbered as Problem (1). Please follow this Google Colab Link for data preparation and some preliminary exercise regarding this problem as a reference to move ahead with this exercise.

1. *Follow the link and solve it.*
2. *Follow the link and solve it.*
3. *With a starting point $\mathbf{x}_0 = \begin{bmatrix}0 & 0 & \dots & 0\end{bmatrix}^\top \in \mathbb{R}^{10}$, solve problem (1) using Newton's method implemented with backtracking line search (use $\alpha_0 = 0.99$, $\rho = 0.5$, $\gamma = 0.5$ for backtracking line search, and $\tau = 10^{-4}$). Comment on difficulties (if any) you face when computing the inverse of Hessian (recall that you need to use an appropriate Python function to compute the inverse of the Hessian). If you face difficulty in computing inverse of Hessian, try to think of some remedy so that you can avoid the issue.*
   - *Let $\mathbf{x}^*$ be the final optimal solution provided by your algorithm. Report the values of $\mathbf{x}^*$ and $\bar{\mathbf{x}}$, and discuss the observations.*
   - *Plot the values $\log(\|\mathbf{x}^k - \mathbf{x}^*\|_2)$ against iterations $k = 0, 1, 2, \dots$.*

- *Prepare a different plot for plotting* $\log(|f(\mathbf{x}^k) - f(\mathbf{x}^*)|)$ *obtained from Newton's method against the iterations.*
- *Comment on the convergence rates of the iterates and the objective function values.*

4. *With a starting point* $\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & \dots & 0 \end{bmatrix}^\top \in \mathbb{R}^{10}$*, solve problem (1) using BFGS method implemented with backtracking line search (use* $\alpha_0 = 0.99$*,* $\rho = 0.5$*,* $\gamma = 0.5$ *for backtracking line search, and* $\tau = 10^{-4}$*).*
   - *Let* $\mathbf{x}^*$ *be the final optimal solution provided by BFGS algorithm. Report the values of* $\mathbf{x}^*$ *and* $\bar{\mathbf{x}}$*, and discuss the observations.*
   - *Plot the values* $\log(\|\mathbf{x}^k - \mathbf{x}^*\|_2)$ *against iterations* $k = 0, 1, 2, \dots$.
   - *Prepare a different plot for plotting* $\log(|f(\mathbf{x}^k) - f(\mathbf{x}^*)|)$ *obtained from BFGS method against the iterations.*
   - *Comment on the convergence rates of the iterates and the objective function values.*
5. *Compare and contrast the results obtained by Newton's method and BFGS method and comment on the time taken by both the methods.*

Bibliography

1. *Y. Nesterov,* Introductory lectures on convex optimization: A basic course. *Springer Science & Business Media, 2003, vol. 87.*
2. *Jiantao Jiao, Course Materials,*
   https://people.eecs.berkeley.edu/~jiantao/227c2022spring/material.html
3. *Matthias L. R. Haußer, Lecture Slides,*
   https://people.maths.ox.ac.uk/hauser/hauser_lecture2.pdf
4. *Numerical Optimization for Machine Learning: Accelerated Gradient Methods,*
   https://www.cs.ubc.ca/~schmidtm/Courses/5XX-S22/S3.pdf
5. *Proximal Algorithms: A Joint Work by Neal Parikh and Stephen Boyd,*
   https://web.stanford.edu/%7Eboyd/papers/pdf/prox_algs.pdf#:~:text=Proximal%20methods%20sit%20at%20a%20higher%20level%20of,itself%20involves%20solving%20a%20small%20convex%20optimization%20problem.