Optimisation Methods: Assignment 3

Deborah Sulem
deborah.sulem@usi.ch

Spring 2025

The purpose of this assignment is to familiarise yourself with Line Search algorithms.

It is due for **Wednesday 26th March at 2 pm**. Only the second part (Python implementation) will be graded but you can also submit your answers for the first part and the latter will also be corrected (though not graded). For the first part, you need to submit a Python notebook on iCorsi and follow these instructions:

- Put the answers for each part of the question into separate cells.
- Before each cell, put a markdown header that says which part of the question comes in the following cell.
- Good coding style is part of the grade: add clear comments throughout the code when it is necessary.
- Before you submit your notebook, make sure it runs.

Part 0: reading

Read Chapter 3 of the book "Numerical Optimisation" (Nocedal & Wright).

Part 1: written exercises

Exercise 1 (Properties of strongly convex functions)
Assume that f is μ-strongly convex.

1. *Prove that: for any $x \in \mathbb{R}^n$,*

$$2\mu(f(x) - f(x^*)) \leq \|\nabla f(x)\|^2,$$

   with x^* the global minimiser of f.

2. *Show that the function $g(x) = f(x) - \frac{\mu}{2}\|x\|_2^2$ is convex.*

3. *Show that $\nabla^2 f(x)$ is invertible for any x.*

4. *Prove that all eigenvalues of $[\nabla^2 f(x)]^{-1}$ are positive and at most $1/\mu$ and conclude that $\|[\nabla^2 f(x)]^{-1}\| \leq \mu^{-1}$.*

Exercise 2
For each of the following sequences,

1. $z^{(k)} = 1 + 2^{-2^k}$
2. $z^{(k)} = 1 - 3^{-k}$
3. $z^{(k)} = \frac{1}{2k+1}$
4. $z^{(k)} = (-1)^k \frac{1}{\sqrt{k!}}$,

prove or disprove whether it converges

1. *sublinearly*
2. *linearly*
3. *superlinearly*
4. *quadratically*

Check the definition of each of these options and write down an argument for each of them.

Exercise 3
Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable. Prove that if for any $x \in \mathbb{R}^n$ the eigenvalues of $\nabla^2 f(x)$ belong to $[m, M]$, with $M \geq m > 0$, then f is m-strongly convex and M-smooth. Hint: you may use the Cauchy-Schwarz inequality

$$x^T y \leq \|x\|\|y\|, \qquad \forall x, y \in \mathbb{R}^n$$

and the following property of the spectral norm:

$$\|Bx\| \leq |||B|||\|x\|$$

for any matrix B and vector x (since by definition $|||B||| = \max_{x \neq 0} \frac{\|Bx\|}{\|x\|}$).

Exercise 4 (Exact line search)
We consider the quadratic objective function defined for all $x \in \mathbb{R}^n$ as:

$$f(x) = \frac{1}{2} x^T Q x + g^T x$$

with parameters $Q \in \mathbb{R}^{n \times n}$, a symmetric and positive definite matrix, and $g \in \mathbb{R}^n$.

1. *Consider a starting point $x^{(0)}$ and a descent direction $d^{(0)}$. What are possible choices for $d^{(0)}$ (seen in class)?*
2. *We want to apply exact line search. What is it ?*
3. *Prove that the solution of exact line search at the first iteration can be computed analytically as:*

$$\alpha^{(0)} = \frac{-\nabla f(x^{(0)})^T d^{(0)}}{(d^{(0)})^T Q d^{(0)}}.$$

4. *Prove that from any starting point $x^{(0)}$, the standard Newton's method leads to $x^{(1)} = x^*$ where x^* is the unique global minimiser of f.*
5. *Prove that f is L-smooth with $L = |||Q|||$, the spectral norm of Q.*

Part 2: programming problems

Problem 1 (Bell curve fitting)
In this exercise we will use a data set of points with 2 coordinates: $(z^{(i)}, y^{(i)})$, $i = 1, \ldots, N$ *with* $z, y \in \mathbb{R}$. *This point cloud is plotted in Figure 1.*

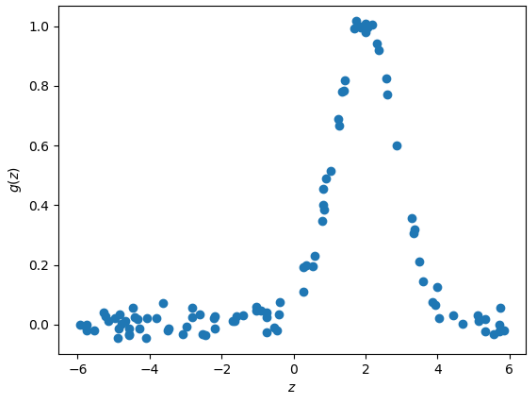

Figure 1: Data point cloud.

Given the shape of this point cloud, we will try to fit a bell curve to it, i.e., i.e., a model of the form:

$$m(z; x) = \exp\left(-\frac{(z - x_1)^2}{x_2}\right),$$

where $x = (x_1, x_2)$ *is a vector of 2 unknown parameters we will want to find. This model is called a bell curve because of its shape (see Figure 2 for an example):*

- x_1 *is the center of the bell curve*
- x_2 *is related to the width of the curve (in fact we need* $x_2 > 0$ *but for now we can ignore this constraint).*

To find the bell curve that bests fit the data, we will find the parameters x *that minimise the mean squared error :*

$$f(x) = \frac{1}{N}\sum_{i=1}^{N}(m(z^{(i)}; x) - y^{(i)})^2.$$

i.e., we will solve the problem:

$$\min_{x \in \mathbb{R}^2} f(x).$$

1. *Import the file 'dataset1.csv' (available under 'Assignment 3' on iCorsi) and store its content into an numpy nd.array. This file is a table containing 100 rows and 2 columns, representing 100 data points with 2 coordinates. Hint: import this table with numpy into an array with* `np.loadtxt` *with argument* `skiprows=2` *and* `delimiter=","`.

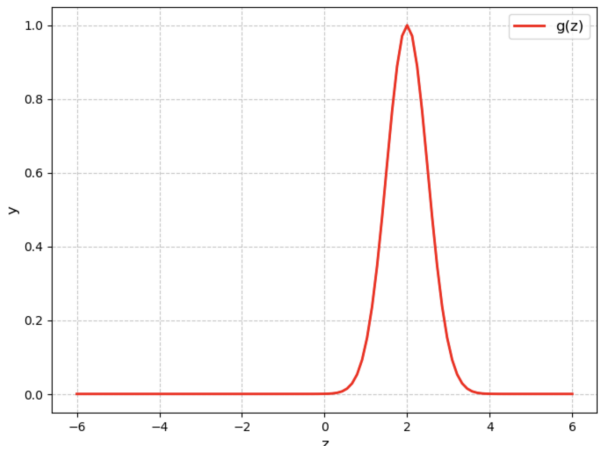

Figure 2: Graph of a bell curve.

2. *Plot this point cloud as in Figure 1, with the first column on the x-axis and the second column one the y-axis.*

3. *Write a function that computes $f(x)$ defined above for any given $x = (x_1, x_2)$. This function must also take as argument the array containing the data points. Evaluate f at $x = (1, 0.5)$ and $x = (0.5, 1)$.*

4. *Show that*

$$\nabla f(x) = \frac{2}{N} \sum_{i=1}^{N} \frac{z^{(i)} - x_1}{x_2} \exp\left(-\frac{(z^{(i)} - x_1)^2}{x_2}\right) (\exp\left(-\frac{(z^{(i)} - x_1)^2}{x_2}\right) - y^{(i)}) . \begin{pmatrix} 2 \\ \frac{z^{(i)} - x_1}{x_2} \end{pmatrix}$$

5. *Write a function that computes the gradient of f for any given x and evaluate ∇f at $x = (1, 0.5)$ and $x = (0.5, 1)$.*

6. *Write a function implementing the Gradient Descent algorithm. You can decide which arguments this function, but this function should also work the same on another data set (i.e., if someone provides another array containing another set of data points). Moreover, at each iteration k, you should print the value of $x^{(k)}, f(x^{(k)})$ and $\|\nabla f(x^{(k)})\|$.*

7. *Test this algorithm for a step size $\alpha \in \{0.1, 1, 10, 100\}$ starting from $x^{(0)} = (2, 5)$ and reports the results. Test also different starting points and comment (Remark: you need to choose $x_2^{(0)} > 0$).*

8. *Plot the bell curve with the optimal parameters on top of the point cloud.*

9. *Import the second file 'dataset2.csv' and store its content into another numpy nd.array.*

10. *Plot this point cloud, with the first column on the x-axis and the second column one the y-axis.*

11. *Run the gradient descent algorithm with step size $\alpha = 10$ until convergence starting from 3 different initialisation points:*

    - $x^{(0)} = (0, 5)$

- $x^{(0)} = (-1, 1)$
- $x^{(0)} = (1, 1)$.

For each initialisation point, plot the bell curve with the parameters found by GD, on top of the data points. Comment the results.

12. *Plot in 3D the surface of the objective function for $x \in [-10, 10] \times [0.2, 10]$, for the first and the second data set.*

Problem 2 (Newton's algorithm)
We consider the same bivariate function as in Problem 2 of Assignment 2:

$$f(x) = 100(y - x^2)^2 + (x - 1)^2$$

Find the minimum of f using Newton's Method and compare with the results obtained with Gradient Descent, when the algorithms are initialised at $x^{(0)} = (2, 2)$. How many iterations does each algorithm need until convergence?

Problem 3 (Newton and Line Search)
We consider the bivariate function

$$f(x) = 4x_1^2 - 3x_1 + x_2^2 + 2x_2.$$

1. *Define a function that computes $f(x)$ for any $x = (x_1, x_2)$ and evaluate it at $x^{(0)} = (0, 0)$.*

2. *Define a function that computes the gradient of f, $\nabla f(x)$ for any $x = (x_1, x_2)$ and evaluate it at $x^{(0)} = (0, 0)$.*

3. *Define a function that computes the Hessian of f, $\nabla^2 f(x)$ for any $x = (x_1, x_2)$ and evaluate it at $x^{(0)} = (0, 0)$.*

4. *Compute the Newton's direction d_N at the point $x^{(0)} = (0, 0)$ and check if it is descent direction.*

5. *Define a function* `line_search` *that computes the value of the line search objective function, in the Newton's direction for a given (anchor) point x and any step size α:*

$$h(\alpha) = f(x + \alpha d_N).$$

6. *Plot the graph of the function $h(\alpha)$ for $\alpha \in [0, 2.5]$ and $x = x^{(0)} = (0, 0)$. What would be the value of α in exact line search ? (give an approximate value using the previous plot).*

7. *We now want to find a step size α that verifies the Wolfe conditions. Define a function* `first_wolfe` *that compute $\ell(\alpha)$ defined as*

$$\ell(\alpha) = f(x^{(0)}) + \alpha\eta\nabla f(x^{(0)})^T d_N$$

with relaxation parameter $\eta = 0.2$.

8. *Plot the graph of $\ell(\alpha)$ for $\alpha \in [0, 2.5]$ on top of the graph of $h(\alpha)$. What would be acceptable values of α according to the first Wolfe conditions? You can give an approximate value given the plot or, optionally, find the value α^* at which $h(\alpha)$ intersects $l(\alpha)$. Hint: you can use the function* `fsolve` *from the package* `scipy.optimize` *(doc here).*

9. *We now consider the second Wolfe condition:*

$$\nabla f(x^{(0)} + \alpha d_N)^T d_N \geq \bar{\eta} \nabla f(x^{(0)})^T d_N$$

with $\bar{\eta} = 0.7$. First explain why this condition is equivalent to:

$$\frac{\nabla f(x^{(0)} + \alpha d_N)^T d_N}{\nabla f(x^{(0)})^T d_N} \leq \bar{\eta}.$$

(Write your answer in a text cell). Second, define a function `second_wolfe` *that compute the ratio $r(\alpha) = \frac{\nabla f(x^{(0)} + \alpha d_N)^T d_N}{\nabla f(x^{(0)})^T d_N}$ for any α.*

10. *Plot the graph of $r(\alpha)$ for $\alpha \in [0, 2.5]$ on top of the graph of $h(\alpha)$ and $\ell(\alpha)$. What would be the acceptable values of α according to both Wolfe conditions? You can give an approximate value given the plot or, optionally, find the value α^* at which $r(\alpha) = \bar{\eta}$. Hint: you can use the function* `fsolve` *from the package* `scipy.optimize` *(doc here).*