

# Homework 1: Gradient Descent

## Contents

- Exercise 1: GD on a 1D Function
- Exercise 2: Backtracking Line Search
- Exercise 3: GD in 2D
- Exercise 4: Exact Line Search vs Backtracking
- Exercise 5: Gradient Descent on the Rosenbrock Function

### Warning

The submission of the homeworks has **NO** deadline. You can submit them whenever you want, on Virtuale. You are only required to upload it on Virtuale **BEFORE** your exam session, since the Homeworks will be a central part of the oral exam.

You are asked to submit the homework as one of the two, following modalities:

- A PDF (or Word) document, containing screenshots of code snippets, screenshots of the results generated by your code, and a brief comment on the obtained results.
- A Python Notebook (i.e. a `.ipynb` file), with cells containing the code required to solve the indicated exercises, alternated with a brief comment on the obtained results in the form of a markdown cell. We remark that the code **SHOULD NOT** be runned during the exam, but the student is asked to enter the exam with all the programs **already executed**, with the results clearly visible on the screen.

Joining the oral exam with a non-executed code OR without a PDF file with the obtained results visible on that, will cause the student to be rejected.

## Exercise 1: GD on a 1D Function

Consider the 1-dimensional function

$$\mathcal{L}(\Theta) = (\Theta - 3)^2 + 1.$$

1. **Compute the Gradient** of  $\mathcal{L}(\Theta)$  explicitly.
2. **Implement the Gradient Descent** to optimize  $\mathcal{L}(\Theta)$  following what we introduced on the theoretical sections.

3. Test three different constant step sizes:

- $\eta = 0.05$
- $\eta = 0.2$
- $\eta = 1.0$

4. For each choice:

- Plot the sequence  $\Theta^{(k)}$  on the real line (i.e. draw a line and represent, on top of it, the position of all the successive elements  $\Theta^{(k)}$ ).
- Plot the function values  $\mathcal{L}(\Theta^{(k)})$  vs iteration.
- Comment on convergence, oscillations, and divergence.

5. Relate your observations to the discussion in class about:

- step-size being too small / too large,
- the role of convexity,
- how the “just right” step size leads to fast convergence.

*Hint:* This function is strictly convex with a unique minimizer at  $\Theta^* = 3$ .

## Exercise 2: Backtracking Line Search

Consider the non-convex function

$$\mathcal{L}(\Theta) = \Theta^4 - 3\Theta^2 + 2.$$

1. Implement **Gradient Descent with Backtracking**, using the Armijo condition, considering the `backtracking(...)` function from class.
2. Test different initial points:

- $\Theta_0 = -2$
- $\Theta_0 = 0.5$
- $\Theta_0 = 2$

3. For each starting point, plot:

- the function curve  $\mathcal{L}(\Theta)$  in 1D in the domain  $[-3, 3]$ ,
- the trajectory of the iterates  $\Theta^{(k)}$  overlaid on the curve.

4. Discuss:

- Why different initializations converge to different minima.
- How backtracking automatically chooses a suitable step size at each iteration.
- Situations where constant step size would fail.

## Exercise 3: GD in 2D

Consider the quadratic function:

$$\mathcal{L}(\Theta) = \frac{1}{2} \Theta^T A \Theta, \quad A = \begin{bmatrix} 1 & 0 \\ 0 & 25 \end{bmatrix}.$$

1. Implement Gradient Descent in 2D, as seen in class.
2. Plot the **level sets** of  $L$  (using the helper code provided in the lecture notes) and overlay the iterates  $\Theta^{(k)}$  for:
  - $\eta = 0.02$ ,
  - $\eta = 0.05$  (borderline),
  - $\eta = 0.1$  (diverging). To overlay the iterates on the plot, simply add a `plt.plot(...)` function taking as input the coordinates  $\Theta_1^{(k)}$  and  $\Theta_2^{(k)}$  for each  $k$ .

3. Comment on:

- the elongated ellipses produced by ill-conditioning,
- the gradient direction compared to the level set lines,
- zig-zag behaviour for large condition numbers,
- the relation to the lecture discussion on slow convergence in narrow valleys.

## Exercise 4: Exact Line Search vs Backtracking

Let

$$\mathcal{L}(\Theta) = \frac{1}{2} \Theta^T A \Theta, \quad A = \begin{bmatrix} 5 & 0 \\ 0 & 2 \end{bmatrix}.$$

1. For GD with **exact line search** i.e. selecting the **optimal** step, you get:

$$\eta_k^* = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T A \mathbf{g}_k}, \quad \mathbf{g}_k = A \Theta^{(k)}.$$

2. For GD with **backtracking**, use the implementation provided in your notes.

3. Starting from  $\Theta_0 = (3, 3)^T$ :

- Plot trajectories on the level-set map.
- Plot the loss  $\mathcal{L}(\Theta^{(k)})$  vs iteration for both methods.

4. Compare:

- speed of convergence,
- smoothness of step-sizes.

# Exercise 5: Gradient Descent on the Rosenbrock Function

The 2-dimensional Rosenbrock function is defined as:

$$f(\Theta) = (1 - \Theta_1)^2 + 100(\Theta_2 - \Theta_1^2)^2,$$

with a unique global minimum at:

$$\Theta^* = (1, 1), \quad \mathcal{L}(\Theta_1^*, \Theta_2^*) = 0.$$

Despite having only one minimum, its landscape is **extremely challenging**:

- The valley is long and curved.
- The condition number varies drastically across the region.
- Gradients can point almost orthogonally to the valley direction.
- Gradient Descent may zig-zag and make very slow progress toward the solution.

This makes Rosenbrock a perfect testbed to study the difficulties of pure Gradient Descent.

1. Implement the function and its gradient to run the Gradient Descent algorithm. Test its performance using both **constant step size** and the **backtracking algorithm**.

2. Plot the level sets of  $\mathcal{L}$ : use the standard 2D contour plotting approach practiced earlier.

Ensure that you plot:

- A wide range (e.g.  $\Theta_1 \in [-2, 2]$ ,  $\Theta_2 \in [-1, 3]$ )
- A reasonable number of contour lines to visualize the narrow valley.

1. Choose multiple initial points, testing at least the following four:

- $\Theta^{(0)} = (-1.5, 2)$
- $\Theta^{(0)} = (-1, 0)$
- $\Theta^{(0)} = (0, 2)$
- $\Theta^{(0)} = (1.5, 1.5)$

Run both:

- GD with constant step size (try  $\eta = 10^{-3}, 10^{-4}, 10^{-5}$ ),
- GD with backtracking line search.

2. Visualize and analyze the trajectories: For each initialization and each training method:

1. Plot the sequence of iterates  $(\Theta_1^{(k)}, \Theta_2^{(k)})$  as a path on the level-set map.
2. Plot the loss curve  $L(\Theta^{(k)})$  vs iteration.
3. Observe and explain:
  - whether the method enters the valley,
  - how long it takes to “turn” correctly along the valley direction,
  - whether the method zig-zags,
  - whether step sizes become too small (with constant  $\eta$ ) or too large (divergence).