NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

ACADEMIC YEAR 2025-2026
SEMESTER 1

EE2213: Introduction to AI

ASSIGNMENT 3: L5-L6

Assignment 3 (5% of total grade)
Submission Deadline: 3 October 2025 (Friday), 23:59

Problem Statement:

In this assignment, you will solve two independent problems using Python:

Problem 1: Logistics optimization:

MobiTech, a global smartphone manufacturer, needs to plan its quarterly distribution. The company operates 3 factories and supplies 4 major markets. The goal is to minimize the total transportation cost while meeting all market demands and not exceeding any factory's production capacity.

Input Data:

- **Factory Capacities (Supply):** China: 50 units, India: 30 units, Brazil: 40 units
- **Market Demands:** Singapore: 20 units, US: 45 units, Germany: 25 units, Japan: 30 units
- **Transportation Cost Matrix ($ per unit):**
    - China to [Singapore, US, Germany, Japan]: [10, 25, 30, 20]
    - India to [Singapore, US, Germany, Japan]: [12, 32, 25, 22]
    - Brazil to [Singapore, US, Germany, Japan]: [35, 20, 15, 40]

Constraints:

- Total units shipped from a factory cannot exceed its capacity.
- Total units received by a market must meet its demand.
- Shipments must be in whole units (no fractional smartphones).

Your Mission:

Formulate this as an Integer Linear Program (ILP) and implement it in Python:

- Minimize the total transportation cost.
- Determine the optimal number of units to ship from each factory to each market.

Your solution should calculate and return two results:

1. minimal_cost: The total minimized transportation cost.

2. shipment_matrix: A 3x4 matrix where shipment_matrix[i, j] is the number of units to ship from Factory i to Market j. The rows correspond to factories in the order: China, India, Brazil. The columns correspond to markets in the order: Singapore, US, Germany, Japan.

Hint: You may find the PuLP or CVXPY libraries helpful for this task.

Problem 2: Gradient descent (convex optimization):

Minimize the function $f(w) = 1 + (w - 5)^2$. Start with an initial value of $w = 3.5$, and perform gradient descent.

Instructions:
Submit a single file named A3_StudentMatriculationNumber.py (replace "StudentMatriculationNumber" with your own your student matriculation number).
Your file must define two functions: optimize_shipments(supply, demand, cost_matrix) and gradient_descent(learning_rate, num_iters)
Do not change the above function names. Do not include any test print statements or I/O code. We will call your functions directly during grading.

Function Details:

optimize_shipments(supply, demand, cost_matrix)

- **Input arguments:**
    - supply (list): A list of factory capacities [capacity_china, capacity_india, capacity_brazil]
    - demand (list): A list of market demands [demand_singapore, demand_us, demand_germany, demand_japan]
    - cost_matrix (2D list): A 3x4 matrix where cost_matrix[i][j] is the cost to ship one unit from factory i to market j. The rows correspond to factories in the order: China, India, Brazil. The columns correspond to markets in the order: Singapore, US, Germany, Japan.
- **Must return two outputs:**
    - minimal_cost (float): The total minimized transportation cost.
    - shipment_matrix (numpy.ndarray): A 3x4 NumPy array of integers (dtype=int) where shipment_matrix[i, j] is the number of units to ship from factory i to market j. The rows correspond to factories in the order: China, India, Brazil. The columns correspond to markets in the order: Singapore, US, Germany, Japan.

gradient_descent(learning_rate, num_iters):

- **Inputs:**
    - learning_rate: A float between 0 and 0.2 (e.g., 0.001)

- num_iters: A positive integer specifying the number of gradient descent iterations for the minimization task.

- **Outputs:**
  - w_out: NumPy array of length num_iters with updated values of w (e.g., w_out[0] is the value of w after the first iteration of gradient descent (NOT the initialized value of w)).
  - f_out: NumPy array of length num_iters with the updated values of $f(w)$. For example, f_out[0] is the value of $f(w)$ after the first iteration of gradient descent, i.e., the value of $f(w)$ given w_out[0].

Do not redefine learning_rate or num_iters inside the function. We will do it in the main calling part.

Rename your file using your student matriculation number.
For example, if your matriculation ID is **A1234567R**, then:

- Filename should be **"A3_A1234567R.py"**

Please use the provided template (A3_StudentMatriculationNumber.py) and do not comment out any lines. You **must not** change the function names provided in the template. Submissions with renamed functions will **lose marks**.

Please do NOT zip/compress your file.

Please ensure you test your code before submission. You can design your own test file to confirm your code produces the correct output.

You must follow instructions strictly; marks will be deducted otherwise due to large class size.

Marks Allocation:

- Correct minimal_cost for problem 1: 1.5%
- Correct shipment_matrix for problem 1: 1.5%
- Correct w_out for problem 2: 1%
- Correct f_out for problem 2: 1%

The way we would run your code might be something like this (main calling part):

>> from A3_A1234567R import optimize_shipments, gradient_descent

>> #code to define supply, demand, cost_matrix, learning_rate and num_iters --- you do not need to redefine them inside your function

>> minimal_cost, shipment_matrix = optimize_shipments(supply, demand, cost_matrix)

>> w_out, f_out = gradient_descent (learning_rate, num_iters)

Please ensure your filenames and function names follow the specified format exactly.

Important Notes:

Please **replace** "StudentMatriculationNumber" in your **filename** with your **actual matriculation number**.

Submission folder: Canvas→EE2213→Assignments→A3

Deadline:
23:59, 3 October 2025 (Friday)
No extensions will be granted.

Late submission policy:

- Up to 12 hours late: 80% of your grade
- 1 day late: 50% of your grade
- 2 days late: 30% of your grade
- More than 2 days late: No marks (0%)