BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
WORK INTEGRATED LEARNING PROGRAMMES DIVISION

Deep Reinforcement Learning
Lab Assignment 1

Total Marks = 12 Marks (6M for the first part and 6M for the second part)

Intended Learning Outcome:

Students should be able to

- Understand the basic functionality of Multi-Armed Bandit and Dynamic Programming.
- Implement the concepts of Multi-Armed Bandit and Dynamic Programming.

Prerequisite:

(1) Students should go through the lectures CS1, CS2, CS3, CS4, CS5 ;
(2) Webinar demonstrations

Please note that this assignment will involve some amount of self-learning (on the part of modelling solutions appropriately + programming skills)

Submission Deadline: 22nd June, 2025

Instructions:

- Read the assignment proposal carefully.
- Solve both assignment problems. Submit two different solution files. (Team # - MAB, Team # - DP)
- It is mandatory to **submit** the assignment in **PDF format only** consisting of all the outcomes with each and every iteration. Any other format will not be accepted.
- Add comments and descriptions to every function you are creating or operation you are performing. If not found, then 1 mark will be deducted. There are many assignments that need to be evaluated. By providing the comments and description it will help the evaluator to understand your code quickly and clearly.

How to reach out for any clarifications:

This assignment is administered by

(1) Pooja Harde - pooja.harde@wilp.bits-pilani.ac.in
(2) Divya K - divyak@wilp.bits-pilani.ac.in
(3) Dincy R Arikkat - dincyrarikkat@wilp.bits-pilani.ac.in

Any request for clarification must be addressed through email (official email only) to all the three instructors listed.

Messaging in TEAMS is discouraged. This is to ensure we maintain track of all the transactions. If we find any clarifications to be shared across all the students, we will share this using discussion forums.

-

Part #1 - MAB
Total Marks = 6 Marks

Title: Adaptive Treatment Selection with Multi-Armed Bandits

Scenario: A pharmaceutical company is conducting clinical trials to evaluate the effectiveness of three antiretroviral drug combinations for treating HIV-positive patients. Due to the ethical and cost constraints of clinical trials, it is critical to identify the most effective treatment regimen using the least number of patients. Each treatment (or "arm") can lead to different outcomes depending on patient responses. The effectiveness of each treatment is evaluated using a reward function derived from the improvement in patients' immune system markers and survival status.

Objective: You are provided with a clinical dataset where each record corresponds to a patient, including the treatment they received and the resulting health outcomes. Your task is to simulate a clinical trial environment using various MAB strategies to sequentially recommend treatments and observe outcomes. The objective is to maximize the overall success rate across trials by identifying and favouring the most effective treatment.

Dataset Description: The dataset containing the following fields:

Age (age): Patient's age in years at baseline.

Weight (wtkg): Continuous feature representing weight in kilograms at baseline.

Gender (gender): Binary indicator of gender (0 = Female, 1 = Male).

CD4 Counts (cd40, cd420): Integer values representing CD4 counts at baseline and 20+/-5 weeks.

Treatment Indicator (trt): Categorical feature indicating the type of treatment received (0 = ZDV only, 1 = ZDV + ddI, 2 = ZDV + Zal, 3 = ddI only).

Censoring Indicator (label): Binary indicator (1 = failure, 0 = censoring) denoting patient status.

Link for accessing the dataset: Clinical_Trial.csv

Environment Details:

Arms (Actions): The treatment types

Arm 0: ZDV only

Arm 1: ZDV + ddI

Arm 2: ZDV + Zal

Arm 3: ddI only

Reward Function:

Reward r is defined as:

r = 1, if (label == 0) and (cd420 > cd40)

r = 0, otherwise

This reward represents a successful treatment outcome as an increase in CD4 count and survival.

Assumptions: Number of Iterations: Simulate at least 1000 trials (iterations). In each iteration, simulate one patient trial using one of the bandit policies.

Requirements and Deliverables:

1. Load the clinical treatment dataset. **(0.5 Mark)**
2. Define the bandit environment with treatment arms and compute the binary reward using CD4 count improvement and patient survival. **(0.5 Mark)**
3. Implement the Random policy for treatment selection. Run the simulation for at least 1000 iterations and print the treatment selected and reward at each iteration. **(0.5 Mark)**
4. Implement the Greedy policy that always selects the treatment with the highest average reward. Run the simulation and print each iteration's decision and reward. **(1 Mark)**
5. Implement the ε-Greedy policy with ε = 0.1, 0.2, 0.5. Report iteration-wise selections and rewards. Determine which ε yields the best result. **(1.5 Marks)**

6. Implement the UCB policy. Simulate and print each step's arm selection, and reward. **(1 Mark)**
7. Plot and compare cumulative rewards and arm selection frequency for all policies in a single graph to evaluate their relative performance. **(0.5 Mark)**
8. Based on the results, write a conclusion (approximately 250 words) summarizing which treatment policy was most effective. Discuss the balance between exploration and exploitation in your simulations. **(0.5 Mark)**

Colab Template: MAB_Clinical_Trial_Assignment.ipynb (*Make a copy of the template. Do not send the edit access request.*)

PART #2 - DP

Total Marks = 6 Marks

Title: The Smart Supplier: Optimizing Orders in a Fluctuating Market

Problem Statement: Develop a reinforcement learning agent using dynamic programming to help a Smart Supplier decide which products to manufacture and sell each day to maximize profit. The agent must learn the optimal policy for choosing daily production quantities, considering its limited raw materials and the unpredictable daily demand and selling prices for different products.

Scenario: A small Smart Supplier manufactures two simple products: Product A and Product B. Each day, the supplier has a limited amount of raw material. The challenge is that the market demand and selling price for Product A and Product B change randomly each day, making some products more profitable than others at different times. The supplier needs to decide how much of each product to produce to maximize profit while managing their limited raw material.

State:

- Raw Material (RM): The supplier starts each day with a fixed amount of raw material (10 units).
    - o Producing 1 unit of Product A costs 2 RM.
    - o Producing 1 unit of Product B costs 1 RM.
- Products:
    - o Product A: High value, but higher raw material cost.
    - o Product B: Lower value, but lower raw material cost.
- Market State: Each day, the market is in one of two states:

- Market State 1 (High Demand for A):
  - Product A sells for +$8 per unit.
  - Product B sells for +$2 per unit.
- Market State 2 (High Demand for B):
  - Product A sells for +$3 per unit.
  - Product B sells for +$5 per unit.

- Day Limit: The problem runs for a fixed number of days (5 days). The goal is to maximize total profit over these days.
- Daily Market Shift: At the start of each new day, the market randomly shifts to either Market State 1 or Market State 2 (50% probability for each)
- Daily Reset: At the end of each day (after the production decision), the raw material is reset to the initial amount (i.e., 10 units) for the next day.

Rewards:

- Selling a unit of Product A: +$8 (if Market State 1) or +$3 (if Market State 2).
- Selling a unit of Product B: +$2 (if Market State 1) or +$5 (if Market State 2).
- Any raw material not used by the end of the day is lost (no penalty, just no gain).
- If the supplier tries to produce more than their available raw material, that production attempt fails (no units produced for that specific action, no penalty beyond wasted action).

Objective: The Smart Supplier's agent must learn the optimal policy π∗ using dynamic programming (Value Iteration or Policy Iteration) to decide how many units of Product A and Product B to produce each day to maximize the total profit over the fixed number of days, given the daily changing market conditions and limited raw material.

State Space:

- **Current Day:** An integer representing the current day (1 to 5).
- **Current Raw Material:** An integer representing the remaining raw material for the current day (0 to 10).
- **Current Market State:** An integer representing the current market condition (1 for "High Demand A", 2 for "High Demand B").

Action Space:

For each state, the agent needs to decide on a production combination. Below are the set of discrete actions:

- Produce_2A_0B: Produce 2 units of Product A, 0 units of Product B.
- Produce_1A_2B: Produce 1 unit of Product A, 2 units of Product B.
- Produce_0A_5B: Produce 0 units of Product A, 5 units of Product B.
- Produce_3A_0B: Produce 3 units of Product A, 0 units of Product B.
- Do_Nothing: Produce 0 units of both products.

Environment Setup:

- This would be a custom Python environment.
- Define the market states and their associated selling prices.
- Implement the raw material consumption for each action.
- Implement the daily market state transition.
- Implement the "day" counter and episode termination.

Requirements and Deliverables:

1. Custom Environment Creation: Design and implement the "Smart Supplier" environment, defining the product costs, daily market shifts, raw material limits, and rewards. **(1 Mark)**
2. Dynamic Programming Implementation: Implement dynamic programming (Value Iteration or Policy Iteration) to find the optimal policy. Crucially, the policy will be a function of the current day, raw material, and market state. **(2 Marks)**
3. Optimal Policy Analysis: Analyze the learned optimal policy. Discuss how the policy changes based upon: **(1 Mark)**
    - The current market state (like does it always favor Product A in Market State 1).
    - The remaining raw material (does it produce more of the cheaper product if raw material is low).
    - The remaining days (does it become more aggressive on the last day).
4. Performance Evaluation: **(1 Mark)**
    - Calculate the state-value function (V∗) for key states (e.g., start of Day 1, Market State 1, 10 RM).
    - Simulate the learned policy over multiple runs (e.g., 1000 runs of 5 days each) and calculate the average total profit achieved.
5. Impact of Dynamics: Compare the optimal policy learned in this dynamic environment to what you might expect if the market prices for Product A and Product B were always fixed (e.g., if it was always Market State 1 every day). How

does the agent's strategy adapt or change when the market can shift unexpectedly, versus if it were always the same? **(1 Mark)**

Colab Template: Dynamic_Programming_code_template.ipynb (*Make a copy of the template. Do not send the edit access request.*)