Making Simple and Complex Decisions

Problem Set

Problem 1: Risk Preferences

Alex is given the choice between two games:

- **Game 1:** A fair coin is flipped. If it comes up heads, Alex receives \$100. If the coin comes up tails, Alex receives nothing.
- **Game 2:** A fair coin is flipped twice. Each time the coin comes up heads, Alex receives \$50. Alex receives nothing for each coin flip that comes up tails.

Definitions & Assumptions:

- Assume Alex has a monotonically increasing utility function for money, $U(x)$, in the range $[\$0, \$100]$.
- Recall that an agent is defined as **risk averse** if the utility of the expected monetary value is greater than the expected utility of the gamble itself. Mathematically: $U(E[X]) > E[U(X)]$.

Question: Show mathematically that if Alex prefers Game 2 to Game 1, then Alex satisfies the definition of risk aversion given above.

Problem 2: Discounted Rewards

Consider a sequential decision problem with a discount factor of $\gamma = 0.50$. The following sequence of rewards is received over time $T = 5$:

$$R_1 = -1, \quad R_2 = 2, \quad R_3 = 6, \quad R_4 = 3, \quad R_5 = 2$$

Question: Calculate the cumulative discounted rewards (returns) $G_0, G_1, \ldots, G_5$.

Hint: Work backwards using the recursive relationship $G_t = R_{t+1} + \gamma G_{t+1}$.

Problem 3: Continuing MDP Policies

Consider the continuing MDP shown in the diagram below. The only decision to be made is in the top state, where two actions are available: **left** and **right**. The numbers show the rewards that are received deterministically after each action.

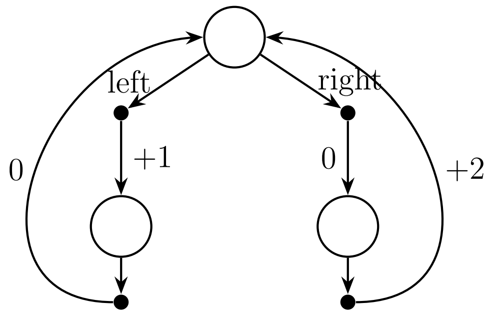


Figure 1: State transition diagram for Problem 3.

There are exactly two deterministic policies: π_{left} (always choose left) and π_{right} (always choose right).

Question: Which policy is optimal for the following discount factors?

(a) $\gamma = 0$

(b) $\gamma = 0.9$

(c) $\gamma = 0.5$

Problem 4: The Recycling Robot (Bellman Equations)

Consider a mobile robot that collects empty soda cans. It has a rechargeable battery with two states: **High** (h) and **Low** (l). The robot acts in an environment defined by the following dynamics:

Actions & Dynamics

In High state (h):

- **Search (s):** With probability α, the robot finds a can and stays in state h (reward r_s). With probability $1-\alpha$, it runs down the battery and transitions to state l (reward r_s).
- **Wait (w):** The robot stays in state h with probability 1 (reward r_w).

In Low state (l):

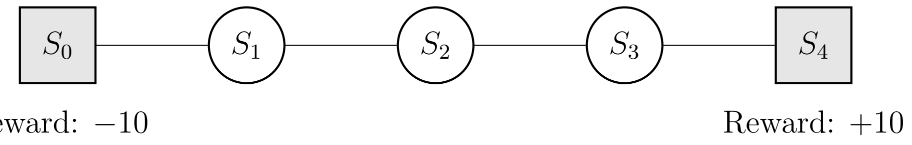
- **Search (s):** With probability β, the robot finds a can and stays in state l (reward r_s). With probability $1 - \beta$, it runs out of battery, has to be rescued, and is reset to state h (reward -3).
- **Wait (w):** The robot stays in state l with probability 1 (reward r_w).
- **Recharge (re):** The robot recharges and transitions to state h with probability 1 (reward 0).

Questions

1. Write the **Bellman Optimality Equations** for the state values $v_*(h)$ and $v_*(l)$.
2. Write the **Bellman Optimality Equations** for the action-values (Q-values) $q_*(s, a)$ for the recycling robot.

Problem 5: The "Slippery Chain" (Policy vs. Value Iteration)

Consider a world consisting of 3 non-terminal states (S_1, S_2, S_3) arranged in a chain, bounded by a **Cliff** (State S_0) on the left and a **Goal** (State S_4) on the right.



Dynamics & Rewards:

- **Actions:** In each state S_1, S_2, S_3, the agent can choose to move **Left (L)** or **Right (R)**.
- **Transition Model (P):** The floor is slippery.
  - If the agent chooses an action (e.g., R), it succeeds and moves to the adjacent neighbor (S_{i+1}) with probability **0.8**.
  - With probability **0.2**, it slips and moves in the *opposite* direction (S_{i-1}).
- **Rewards (R):**
  - Entering S_0 gives a reward of -10 (terminal).

  - Entering S_4 gives a reward of $+10$ (terminal).
  - Transitions between non-terminal states (S_1, S_2, S_3) have a step cost (reward) of -1.

- **Discount Factor (γ):** 0.5.

Part 1: Policy Iteration
We start with an initial policy π_0 where the agent always chooses to go **Left**: $\pi_0(S_1) = L, \pi_0(S_2) = L, \pi_0(S_3) = L$.

(a) **Policy Evaluation:** Write down the **system of 3 linear equations** that must be solved to find the utilities $U^{\pi_0}(S_1), U^{\pi_0}(S_2)$, and $U^{\pi_0}(S_3)$. *(Note: You do not need to solve the system for the final numbers, just structure the equations explicitly with all values plugged in.)*

(b) **Policy Improvement:** Suppose you are given the solution to the system above (hypothetically): $U^{\pi_0}(S_1) \approx -6.6$, $U^{\pi_0}(S_2) \approx -5.8$, $U^{\pi_0}(S_3) \approx -4.2$.
Perform the **Policy Improvement** step for state S_2. Calculate the Q-value for taking action **Right** in S_2 and determine if the policy $\pi(S_2)$ should change.

Part 2: Value Iteration
Instead of solving linear equations, we use Value Iteration to approximate the optimal utilities. Initialize $U_0(S) = 0$ for all non-terminal states.

(a) Calculate the utility values for the first iteration, $U_1(S_1)$, $U_1(S_2)$, and $U_1(S_3)$.

(b) Calculate the utility value $U_2(S_3)$ (the utility of state S_3 in the second iteration). Show your work clearly.

Problem 6: Reward Transformation Analysis

In Reinforcement Learning, designing the reward function is critical. A common question is whether we can shift all rewards by a constant amount without changing the optimal behavior of the agent.

Suppose we transform an MDP's reward function such that every received reward R is shifted by a constant scalar c. That is, the new reward is $R' = R + c$.

(a) **Continuing Tasks:**
Consider a general continuing (infinite-horizon) MDP with discount factor $\gamma < 1$.

- Prove mathematically that adding a constant c to all rewards adds a constant value, v_c, to the value of every state.
- Express v_c in terms of c and γ.
- Does this transformation affect the relative values of states or change the optimal policy? Explain why or why not.

(b) **Episodic Tasks:**
Now consider a general episodic MDP (where tasks end in a terminal state after a finite number of steps).

- Would adding a constant c to all rewards leave the optimal policy unchanged, as in the continuing case?
- Explain your reasoning. If the policy might change, provide a simple conceptual example (e.g., in a shortest-path problem) illustrating how the sign of c (positive vs. negative) influences the agent's behavior.

Problem 7: The Maze Robot

Imagine that you are designing a robot to run a maze. You decide to give it a reward of **+1** for escaping from the maze (reaching the terminal goal state) and a reward of **zero** at all other times.

The task breaks down naturally into episodes (successive runs through the maze), so you treat it as an episodic task where the goal is to maximize the expected total reward. After training the agent for a while, you observe that it successfully escapes the maze eventually, but it shows **no improvement** in the speed or efficiency of its path. It often wanders in loops before finally escaping.

Questions:

1. What is going wrong with the reward signal? Why does the current setup fail to motivate the agent to find the shortest path?
2. Have you effectively communicated to the agent what you want it to achieve?
3. Suggest a mathematical modification to the problem formulation (e.g., changing the rewards or adding a parameter) that would force the agent to prefer shorter paths to the exit.