



DEPARTMENT OF COMPUTER SCIENCE ST. FRANCIS XAVIER UNIVERSITY

St. Francis Xavier University
Department of Computer Science

CSCI-531 - Reinforcement Learning
Practice Exercises: MDPs and Dynamic Programming

Part I: MDP Fundamentals

1. MDP Components Identification

Consider a smart thermostat that learns to control room temperature efficiently. The thermostat can set the temperature to Low, Medium, or High, and the room temperature depends on outside weather and current setting.

- (a) Identify and define the four MDP components for this scenario:

Solution:

- **States (S):** Current room temperature (e.g., Cold, Comfortable, Hot), outside temperature (Hot, Mild, Cold), time of day
- **Actions (A):** Set thermostat to {Low, Medium, High}
- **Transitions (T):** Probability of room temperature change given action and weather conditions
- **Rewards (R):** +reward for comfortable temperature, -penalty for energy usage, -penalty for uncomfortable temperature

- (b) Write the transition probability notation for: "Given the room is Cold and we set thermostat to High, there's a 70% chance the room becomes Comfortable."

Solution: $p(\text{Comfortable}|\text{Cold, High}) = 0.7$

Or more formally: $p(s' = \text{Comfortable}|s = \text{Cold}, a = \text{High}) = 0.7$

- (c) Design a reward function that encourages energy efficiency while maintaining comfort.

$$\text{Solution: } r(s, a) = \begin{cases} +10 & \text{if room is Comfortable} \\ -5 & \text{if room is Hot or Cold} \\ -2 & \text{if action is High (energy penalty)} \\ -1 & \text{if action is Medium} \\ 0 & \text{if action is Low} \end{cases}$$

Total reward combines comfort and energy terms.

2. Markov Property Analysis

- (a) For each scenario, determine if it satisfies the Markov Property. If not, suggest how to modify the state representation:
1. A chess AI where the state is the current board position
 2. A stock trading bot where the state is only today's stock price
 3. A robot navigating where the state includes position, velocity, and battery level
 4. A recommendation system where the state is the user's last clicked item

Solution:

1. **Markovian** - Current board position contains all information needed for optimal play
2. **Not Markovian** - Need price history, trends, volume. Modify state to include recent price history and market indicators
3. **Markovian** - Position, velocity, and battery contain sufficient information for navigation decisions
4. **Not Markovian** - Single item doesn't capture user preferences. Modify state to include user profile, recent history, session context

- (b) An autonomous vehicle's state is defined as: `(current_speed, position, destination)`. Explain why this might not be Markovian and propose a better state representation.

Solution: Why not Markovian: Missing critical information like: - Traffic conditions and other vehicles - Road conditions (weather, construction) - Recent driving history (for predicting behavior) - Vehicle dynamics (acceleration, steering angle)

Better state representation:

- position, velocity, acceleration, steering_angle
- nearby_vehicles, traffic_lights, weather_conditions
- road_type, destination, recent_actions

This captures all information needed to predict future states and make optimal driving decisions.

3. Return Calculations

An agent receives the following reward sequence: $r_0 = 5, r_1 = -2, r_2 = 8, r_3 = 1, r_4 = 3$

- (a) Calculate the undiscounted return G_0 .

Solution: $G_0 = r_0 + r_1 + r_2 + r_3 + r_4 = 5 + (-2) + 8 + 1 + 3 = 15$

- (b) Calculate the discounted return G_0 with $\gamma = 0.8$.

Solution: $G_0 = r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \gamma^4 r_4 = 5 + 0.8(-2) + 0.8^2(8) + 0.8^3(1) + 0.8^4(3)$
 $= 5 - 1.6 + 5.12 + 0.512 + 1.2288 = 10.26$

- (c) Calculate G_2 with $\gamma = 0.9$.

Solution: $G_2 = r_2 + \gamma r_3 + \gamma^2 r_4 = 8 + 0.9(1) + 0.9^2(3) = 8 + 0.9 + 2.43 = 11.33$

4. Simple MDP Analysis

Consider a 2-state MDP with states $S = \{s_1, s_2\}$ and actions $A = \{a_1, a_2\}$:

Transition probabilities:

- $p(s_1|s_1, a_1) = 0.7, p(s_2|s_1, a_1) = 0.3$
- $p(s_1|s_1, a_2) = 0.2, p(s_2|s_1, a_2) = 0.8$
- $p(s_1|s_2, a_1) = 0.4, p(s_2|s_2, a_1) = 0.6$
- $p(s_1|s_2, a_2) = 0.1, p(s_2|s_2, a_2) = 0.9$

Rewards: $r(s_1, a_1) = 2, r(s_1, a_2) = 1, r(s_2, a_1) = 0, r(s_2, a_2) = 3$

Discount factor: $\gamma = 0.9$

- (a) Consider the deterministic policy $\pi(s_1) = a_1, \pi(s_2) = a_2$. Write the system of Bellman equations for this policy.

Solution: For policy π : $v_\pi(s_1) = r(s_1, a_1) + \gamma \sum_{s'} p(s'|s_1, a_1) v_\pi(s')$
 $v_\pi(s_1) = 2 + 0.9[0.7 \cdot v_\pi(s_1) + 0.3 \cdot v_\pi(s_2)]$
 $v_\pi(s_2) = r(s_2, a_2) + \gamma \sum_{s'} p(s'|s_2, a_2) v_\pi(s')$
 $v_\pi(s_2) = 3 + 0.9[0.1 \cdot v_\pi(s_1) + 0.9 \cdot v_\pi(s_2)]$
System: $v_\pi(s_1) = 2 + 0.63v_\pi(s_1) + 0.27v_\pi(s_2)$
 $v_\pi(s_2) = 3 + 0.09v_\pi(s_1) + 0.81v_\pi(s_2)$

- (b) Solve the system to find $v_\pi(s_1)$ and $v_\pi(s_2)$.

Solution: Rearranging:

$$0.37v_\pi(s_1) - 0.27v_\pi(s_2) = 2$$

$$-0.09v_\pi(s_1) + 0.19v_\pi(s_2) = 3$$

$$\text{From second equation: } v_\pi(s_1) = \frac{0.19v_\pi(s_2) - 3}{0.09}$$

$$\text{Substituting: } 0.37 \cdot \frac{0.19v_\pi(s_2) - 3}{0.09} - 0.27v_\pi(s_2) = 2$$

$$\text{Solving: } v_\pi(s_2) \approx 18.9 \text{ and } v_\pi(s_1) \approx 6.2$$

- (c) Write the Bellman optimality equations for both states.

$$\text{Solution: } v^*(s_1) = \max_{a \in \{a_1, a_2\}} \{r(s_1, a) + \gamma \sum_{s'} p(s'|s_1, a)v^*(s')\}$$

$$v^*(s_1) = \max \begin{cases} 2 + 0.9[0.7v^*(s_1) + 0.3v^*(s_2)] & (\text{action } a_1) \\ 1 + 0.9[0.2v^*(s_1) + 0.8v^*(s_2)] & (\text{action } a_2) \end{cases}$$

$$v^*(s_2) = \max \begin{cases} 0 + 0.9[0.4v^*(s_1) + 0.6v^*(s_2)] & (\text{action } a_1) \\ 3 + 0.9[0.1v^*(s_1) + 0.9v^*(s_2)] & (\text{action } a_2) \end{cases}$$

Part II: Dynamic Programming

5. Policy Evaluation Algorithm

Consider a 3×3 gridworld where an agent starts at position $(0,0)$ and wants to reach the goal at $(2,2)$. The agent can move Up, Down, Left, Right, but there's a 10% chance of staying in place on each move.

Actions that would move outside the grid result in staying in the current position. Reward: -1 per step, +10 for reaching goal. Discount factor: $\gamma = 0.9$

- (a) Define the policy evaluation update equation for this problem.

$$\text{Solution: For a policy } \pi: V_{k+1}(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s, a)[r(s, a) + \gamma V_k(s')]$$

For our gridworld: - $r(s, a) = -1$ for all non-goal states, $r(\text{goal}, a) = 10$ - $p(s'|s, a) = 0.9$ if s' is the intended next state, 0.1 if $s' = s$ (stay in place) - If action would go outside grid, $p(s|s, a) = 1.0$

- (b) Consider a simple policy: "always move right if possible, otherwise move up". Perform 2 iterations of policy evaluation starting with $V_0(s) = 0$ for all states. Show calculations for at least 3 states.

Solution: Policy: Right if possible, else Up

$$\text{Iteration 1: } V_1((0, 0)) = -1 + 0.9[0.9 \cdot V_0((1, 0)) + 0.1 \cdot V_0((0, 0))] = -1 + 0.9[0] = -1$$

$$V_1((1, 1)) = -1 + 0.9[0.9 \cdot V_0((2, 1)) + 0.1 \cdot V_0((1, 1))] = -1 + 0.9[0] = -1$$

$$V_1((2, 2)) = 10 + 0.9[1.0 \cdot V_0((2, 2))] = 10 \text{ (goal state)}$$

$$\text{Iteration 2: } V_2((0, 0)) = -1 + 0.9[0.9 \cdot (-1) + 0.1 \cdot (-1)] = -1 + 0.9(-1) = -1.9$$

$$V_2((1, 1)) = -1 + 0.9[0.9 \cdot (-1) + 0.1 \cdot (-1)] = -1.9$$

$$V_2((2, 1)) = -1 + 0.9[0.9 \cdot 10 + 0.1 \cdot (-1)] = -1 + 0.9(8.9) = 7.01$$

6. Value Iteration vs Policy Iteration

- (a) Compare Value Iteration and Policy Iteration algorithms by filling in the table:

Aspect	Policy Iteration	Value Iteration
Update Equation		
Convergence		
Per Iteration Cost		
When to Use		

Solution:

Update Equation	$V(s) = \max_a \sum_{s'} p(s' s, a)[r + \gamma V(s')]$
Convergence	Finite steps (for finite MDPs)
Per Iteration Cost	Higher (full policy evaluation)
When to Use	Small action spaces, want explicit policy

- (b) For the 2-state MDP from Question 4, perform one iteration of Value Iteration starting with $V_0(s_1) = 0, V_0(s_2) = 0$. Show all calculations.

Solution: Value Iteration Update: $V_1(s_1) = \max \begin{cases} 2 + 0.9[0.7 \cdot 0 + 0.3 \cdot 0] = 2 & (a_1) \\ 1 + 0.9[0.2 \cdot 0 + 0.8 \cdot 0] = 1 & (a_2) \end{cases} = 2$

$$V_1(s_2) = \max \begin{cases} 0 + 0.9[0.4 \cdot 0 + 0.6 \cdot 0] = 0 & (a_1) \\ 3 + 0.9[0.1 \cdot 0 + 0.9 \cdot 0] = 3 & (a_2) \end{cases} = 3$$

Therefore: $V_1(s_1) = 2, V_1(s_2) = 3$

Optimal actions: $\pi^*(s_1) = a_1, \pi^*(s_2) = a_2$

7. Policy Improvement

Given the value function $v_\pi(s)$ for some policy π : - $v_\pi(s_1) = 5.2$ - $v_\pi(s_2) = 8.7$ - $v_\pi(s_3) = 3.1$

And the MDP parameters from a 3-state system: - $\gamma = 0.8$ - Rewards: $r(s_1, a_1) = 2, r(s_1, a_2) = 1$ - Transitions from s_1 : $p(s_2|s_1, a_1) = 0.6, p(s_3|s_1, a_1) = 0.4$ - Transitions from s_1 : $p(s_1|s_1, a_2) = 0.3, p(s_2|s_1, a_2) = 0.7$

- (a) Calculate $q_\pi(s_1, a_1)$ and $q_\pi(s_1, a_2)$.

Solution: $q_\pi(s_1, a_1) = r(s_1, a_1) + \gamma \sum_{s'} p(s'|s_1, a_1)v_\pi(s') = 2 + 0.8[0.6 \cdot 8.7 + 0.4 \cdot 3.1] = 2 + 0.8[5.22 + 1.24] = 2 + 0.8(6.46) = 2 + 5.168 = 7.168$

$$q_\pi(s_1, a_2) = r(s_1, a_2) + \gamma \sum_{s'} p(s'|s_1, a_2)v_\pi(s') = 1 + 0.8[0.3 \cdot 5.2 + 0.7 \cdot 8.7] = 1 + 0.8[1.56 + 6.09] = 1 + 0.8(7.65) = 1 + 6.12 = 7.12$$

- (b) What action should the improved policy π' take in state s_1 ?

Solution: Since $q_\pi(s_1, a_1) = 7.168 > q_\pi(s_1, a_2) = 7.12$, the improved policy should select: $\pi'(s_1) = a_1$

- (c) (3 points) Is the current policy π optimal in state s_1 ? Justify your answer.

Solution: The policy is not optimal in s_1 because $q_\pi(s_1, a_1) = 7.168 > v_\pi(s_1) = 5.2$.

This means there exists an action (a_1) that gives higher value than the current policy's expected value, so the policy can be improved.