WIDS Week 3

Name: Dhruvin Doshi

Roll No.: 24B0984

Contents

Chapter 1

Question 1: The “Cliff Walker ” (Manual Calculation)

1.1 Environment Description

We consider a simple one-dimensional GridWorld consisting of four states:

$$S_0,\ S_1,\ S_2,\ S_{\text{Term}}$$

The agent may take two actions in each non-terminal state: *Left* or *Right*. The environment dynamics are deterministic. Moving left from S_0 leaves the agent in S_0, while moving right from S_2 transitions the agent to the terminal state S_{Term}, ending the episode.

The reward structure is defined as follows:

- Any transition into S_{Term} yields a reward of $+10$
- All other transitions yield a reward of -1 (living penalty)

The discount factor is $\gamma = 0.9$.

1.2 Subquestion 1: Calculation of the Discounted Return

The agent begins in state S_1 and follows the trajectory:

$$S_1 \to S_2 \to S_1 \to S_2 \to S_{\text{Term}}$$

We compute the return starting at time $t = 0$. Using the definition of discounted return:

$$G_0 = R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4$$

The rewards encountered along the trajectory are:

- $R_1 = -1$
- $R_2 = -1$
- $R_3 = -1$
- $R_4 = +10$

Substituting values:

$$G_0 = (-1) + 0.9(-1) + 0.9^2(-1) + 0.9^3(10)$$

Evaluating powers of γ:

$$0.9^2 = 0.81, \quad 0.9^3 = 0.729$$

Thus:

$$G_0 = -1 - 0.9 - 0.81 + 7.29 = 4.58$$

This shows how delayed rewards dominate short-term penalties under discounting.

1.3 Subquestion 2: Bellman Expectation Equation for $v_\pi(S_2)$

We now consider a *Random Drunk* policy π under which the agent selects Left or Right with probability 0.5 each.

From state S_2:

- Action Left leads to S_1 with reward -1
- Action Right leads to S_{Term} with reward $+10$

Assuming $v_\pi(S_{\text{Term}}) = 0$, the Bellman expectation equation gives:

$$v_\pi(S_2) = \frac{1}{2}[-1 + \gamma v_\pi(S_1)] + \frac{1}{2}[10]$$

This equation expresses the value of S_2 recursively in terms of its neighboring states.

1.4 Discussion

This question demonstrates both numerical return calculation and the recursive nature of value functions, which is central to reinforcement learning theory.

Chapter 2

Question 2: The Philosophy of Reward (Design)

2.1 Problem Setup

We consider a robot trained to clean a room. The agent receives a reward of $+1$ whenever its sensors detect that dust has been sucked up. The agent eventually maximizes reward, but the room becomes dirtier over time.

2.2 Analysis of Learned Behavior

The agent optimizes the reward function exactly as defined, not the designer's true intention. Since reward is tied only to the detection of dust being collected, the agent may exploit this signal.

Possible exploitative behaviors include repeatedly releasing collected dust and sucking it up again, or manipulating sensor readings to trigger false detections. In each case, reward is accumulated without genuine cleaning.

2.3 Reward Hacking

This phenomenon is known as *reward hacking*, where the agent discovers loopholes in the reward function that allow high reward without accomplishing the intended task.

2.4 Conclusion

Effective reward design must align the reward signal with long-term objectives rather than local, easily exploitable indicators.

Chapter 3

Question 3: The Discount Factor (Concept)

3.1 Introduction

The discount factor γ controls how much future rewards are valued relative to immediate rewards.

3.2 Part A: Mathematical Necessity of $\gamma < 1$

For infinite-horizon tasks, the return is:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

If rewards are always $+1$ and $\gamma = 1$, then $G_t = \sum_{k=0}^{\infty} 1$, which diverges. Hence, $v_\pi(s)$ becomes infinite and ill-defined. Choosing $\gamma < 1$ ensures convergence of the geometric series.

3.3 Part B: Intuition Behind Different Values of γ

When $\gamma = 0$, the agent considers only immediate reward and behaves impulsively. When $\gamma = 0.99$, the agent values future outcomes heavily and behaves strategically, planning for long-term benefit.

3.4 Summary

The discount factor is both a mathematical necessity and a behavioral control parameter.

Chapter 4

Question 4: The Brain Teaser (Reward Hypothesis)

4.1 Original Environment

Each step yields a reward of -1, and reaching the goal terminates the episode. The optimal policy minimizes the number of steps.

4.2 Modified Reward Structure

Adding a constant $+2$ changes the per-step reward to $+1$.

4.3 Effect on Optimal Policy

With $\gamma = 1$, the agent now prefers longer episodes in order to accumulate more reward. If possible, it will avoid reaching the goal entirely.

4.4 Conclusion

Yes, the optimal policy changes. This illustrates how adding constants to rewards can fundamentally alter agent behavior.

Chapter 5

Question 5: Rigorous Derivation of the Bellman Expectation Equation

5.1 Starting Point

We begin with the definition:

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

5.2 Recursive Definition of Return

$$G_t = R_{t+1} + \gamma G_{t+1}$$

5.3 Applying the Law of Iterated Expectations

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \end{aligned}$$

5.4 Expanding Over Actions and Transitions

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$$

5.5 Conclusion

This completes the formal derivation, showing each required expansion explicitly.

Chapter 6

Question 6: The Linear Algebra of Reinforcement Learning

6.1 Bellman Equation as a Linear System

For a fixed policy, the Bellman equation can be written as:

$$v_\pi = R_\pi + \gamma P_\pi v_\pi$$

6.2 Matrix Solution

Rearranging:

$$(I - \gamma P_\pi)v_\pi = R_\pi$$

Thus:

$$v_\pi = (I - \gamma P_\pi)^{-1} R_\pi$$

6.3 Computational Wall

Backgammon has approximately 10^{20} states. Matrix inversion scales as $O(N^3)$, requiring roughly 10^{60} operations, which would take approximately 10^{34} years even on a supercomputer.

6.4 Implications

Exact solutions are infeasible for large problems, motivating Monte Carlo and approximate methods.

Chapter 7

Question 7: The Model-Free Dilemma (Design)

7.1 Policy Improvement Using $v^*(s)$

The greedy policy using state-values is:

$$\pi'(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v^*(s')]$$

This requires full knowledge of environment dynamics.

7.2 Policy Improvement Using $q^*(s, a)$

Using action-values:

$$\pi'(s) = \arg\max_a q^*(s, a)$$

7.3 Model-Free Comparison

In real-world settings, transition probabilities are unknown. Action-values allow direct estimation from experience, making model-free control possible.

7.4 Conclusion

Learning $q^*(s, a)$ is essential for optimal decision-making in unknown environments.