WiDS Assignment: Week 3 (Reinforcement Learning)

Nihar Chouhan

December 2025

Part I: The Essentials

Question 1: The "Cliff Walker" (Manual Calculation)

1. Calculate Return (G_0)

The agent follows the path: $S_1 \xrightarrow{Right} S_2 \xrightarrow{Left} S_1 \xrightarrow{Right} S_2 \xrightarrow{Right} S_{Term}$.

- **Step 1** ($S_1 \to S_2$): Reward $R_1 = -1$.
- **Step 2** ($S_2 \to S_1$): Reward $R_2 = -1$.
- **Step 3** ($S_1 \to S_2$): Reward $R_3 = -1$.
- **Step 4** ($S_2 \to S_{Term}$): Reward $R_4 = +10$.

Using the discount factor $\gamma = 0.9$, the total discounted return G_0 is:

$$\begin{aligned} G_0 &= R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 \\ G_0 &= (-1) + 0.9(-1) + (0.9)^2(-1) + (0.9)^3(10) \\ G_0 &= -1 - 0.9 - 0.81 + (0.729)(10) \\ G_0 &= -2.71 + 7.29 \\ G_0 &= \mathbf{4.58} \end{aligned}$$

2. Value Function (v_π)

We must write the Bellman Expectation Equation for state S_2 under a "Random Drunk" policy where $\pi(\text{Left}|S_2) = 0.5$ and $\pi(\text{Right}|S_2) = 0.5$.
The general Bellman equation is:

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

For state S_2:

- **Action Left (0.5):** Transitions to S_1, Reward is -1. Value contribution: $v_\pi(S_1)$.

- **Action Right (0.5):** Transitions to S_{Term}, Reward is +10. Value contribution: $v_\pi(S_{Term})$.

Since S_{Term} is the end of the episode, $v_\pi(S_{Term}) = 0$. Substituting these into the equation:

$$v_\pi(S_2) = 0.5 \times [-1 + \gamma v_\pi(S_1)] + 0.5 \times [10 + \gamma v_\pi(S_{Term})]$$
$$v_\pi(S_2) = 0.5[-1 + 0.9 v_\pi(S_1)] + 0.5[10 + 0]$$

Question 2: The Philosophy of Reward (Design)

Behavior Learned:
The agent has likely learned a cycle of **sucking up dust and then ejecting it (or dumping it) back onto the floor** to suck it up again.
Explanation:
The reward function gives +1 every time dust is "Sucked Up". It does not penalize creating a mess or ejecting dust. Therefore, the optimal policy to maximize total reward over time is not to clean the room once and stop, but to clean a spot ($R = +1$), immediately make it dirty again, and clean it again ($R = +1$). This allows the agent to collect infinite rewards while the room remains dirty.

Question 3: The Discount Factor (Concept)

Part A: The Math

- **Why $\gamma < 1$?** If the task is continuous (infinite horizon) and rewards are non-zero (e.g., constant +1), the sum of rewards $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ would diverge to infinity if $\gamma = 1$. We need $\gamma < 1$ for the series to converge.
- **If $\gamma = 1$ with +1 rewards:** The value function $v_\pi(s)$ would be infinite (∞). The agent would be unable to distinguish between policies as all would appear to yield infinite return.

Part B: The Intuition

- **Case 1 ($\gamma = 0$):** This agent is **"Impulsive"**. It considers only the immediate reward (R_{t+1}) and ignores all future consequences.
- **Case 2 ($\gamma = 0.99$):** This agent is **"Strategic"**. It weighs future rewards almost as heavily as immediate ones, potentially sacrificing short-term gain for long-term value.

Question 4: The Brain Teaser (Nuance)

Does the optimal policy π_* change?
Yes. Description of new behavior:

- **Original Setup ($R = -1$):** The goal is to minimize penalty, so the agent seeks the shortest path to end the episode.
- **Modification ($R = +1$):** With the boost ($R_{new} = -1 + 2 = +1$), every step gives a positive reward. Since the goal state ends the episode (stopping reward accumulation), the agent now wants to **avoid the goal state forever**.
- **Result:** The optimal policy is to loop infinitely in the environment to accumulate infinite positive reward, rather than finishing the game.

Part II: Advanced Mechanics

Question 5: Rigorous Derivation (Proof)

We start with the definition of the value function:

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

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

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s]$$

The expectation depends on the action a chosen by the policy $\pi(a|s)$. We sum over all actions:

$$v_\pi(s) = \sum_a \pi(a|s)\mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a]$$

Given state s and action a, the environment transitions to s' with reward r via probability $p(s', r|s, a)$. We sum over all s', r:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma\mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']]$$

The term $\mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']$ is simply $v_\pi(s')$. Thus proving:

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

Question 6: The Linear Algebra of RL (Systems)

1. Matrix Form
The Bellman equation is $v_\pi = \mathcal{R}^\pi + \gamma\mathcal{P}^\pi v_\pi$. Solving for v_π:

$$\begin{aligned} v_\pi - \gamma\mathcal{P}^\pi v_\pi &= \mathcal{R}^\pi \\ (I - \gamma\mathcal{P}^\pi)v_\pi &= \mathcal{R}^\pi \\ v_\pi &= (I - \gamma\mathcal{P}^\pi)^{-1}\mathcal{R}^\pi \end{aligned}$$

2. The Computational Wall

- States (N) $\approx 10^{20}$.
- Matrix Inversion Complexity $\approx O(N^3) = (10^{20})^3 = 10^{60}$ operations.
- Supercomputer Speed = 10^{18} FLOPs/sec.

Time in seconds:

$$\text{Time} = \frac{10^{60}}{10^{18}} = 10^{42} \text{ seconds}$$

Converting to years:

$$\text{Years} \approx \frac{10^{42}}{3.15 \times 10^7} \approx 3.17 \times 10^{34} \text{ years}$$

3. Conclusion
Analytical solutions (matrix inversion) are computationally impossible for large state spaces like Backgammon. We must use approximation methods like Monte Carlo.

Question 7: The "Model-Free" Dilemma (Design)

1. Greedy Action using $v(s)$
To pick the best action using only state values, we must look ahead using the transition dynamics:

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

2. Greedy Action using $q(s, a)$
If we have action values, the look-ahead is already computed. We simply pick the max:

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

3. The Comparison
Using $v_*(s)$ requires knowing $p(s', r|s, a)$ (the environment model). In a "Model-Free" environment (like Blackjack or the Real World), these probabilities are unknown. However, using $q_*(s, a)$ does not require the model; we can simply act greedily based on the learned values. This is why Model-Free control relies on action values.