Week 3

Ajinkya Chandak

December 31, 2025

Question 1

Part 1:

The agent follows this path: $S_1 \xrightarrow{\text{Right}} S_2 \xrightarrow{\text{Left}} S_1 \xrightarrow{\text{Right}} S_2 \xrightarrow{\text{Right}} S_{\text{Term}}$
At each transition:

- $S_1 \to S_2$: $r_1 = -1$
- $S_2 \to S_1$: $r_2 = -1$
- $S_1 \to S_2$: $r_3 = -1$
- $S_2 \to S_{\text{Term}}$: $r_4 = +10$

Using $\gamma = 0.9$:

$$\begin{aligned} G_0 &= r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 \\ &= -1 + (0.9)(-1) + (0.9)^2(-1) + (0.9)^3(10) \\ &= -1 - 0.9 - 0.81 + 7.29 \\ &= 4.58 \end{aligned}$$

Part 2:

Under the random policy, the agent picks Left or Right with probability 0.5 each.
From S_2:

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

Since $v_\pi(S_{\text{Term}}) = 0$:

$$\begin{aligned} v_\pi(S_2) &= \sum_a \pi(a|S_2) \sum_{s',r} p(s', r|S_2, a)[r + \gamma v_\pi(s')] \\ &= 0.5[10 + 0.9(0)] + 0.5[-1 + 0.9 v_\pi(S_1)] \\ &= 5 + 0.5(-1 + 0.9 v_\pi(S_1)) \\ &= 5 - 0.5 + 0.45 v_\pi(S_1) \\ &= 4.5 + 0.45 v_\pi(S_1) \end{aligned}$$

Question 2

The robot discovers it can get +1 reward repeatedly by recycling the same dust.

When the robot sucks up dust, it goes into its internal bag. The robot figures out it can dump this dust back onto the floor, then immediately suck it up again for another +1 reward. It keeps doing this in a loop - suck, dump, suck, dump - collecting infinite rewards while the room never actually gets clean.

The problem is we're rewarding the action (sucking dust) rather than the outcome (clean room). A better design would reward based on how clean the room actually is.

Question 3

Part A:

Consider an infinite task where we get reward $r = 1$ at every timestep. If $\gamma = 1$:

$$v_\pi(s) = 1 + 1 + 1 + \cdots = \infty$$

The value function explodes. We can't compare states or find optimal policies when everything has infinite value.

With $\gamma < 1$, we get a geometric series:

$$v_\pi(s) = r + \gamma r + \gamma^2 r + \cdots = \frac{r}{1-\gamma}$$

This converges since $|\gamma| < 1$.

Part B:

When $\gamma = 0$, the agent only sees one step ahead: $v_\pi(s) = \mathbb{E}[R_{t+1}]$. This creates myopic behavior - grab immediate rewards without thinking about consequences.

When $\gamma = 0.99$, future rewards barely decay: $v_\pi(s) = \mathbb{E}[R_{t+1} + 0.99R_{t+2} + 0.99^2 R_{t+3} + \cdots]$. The agent plans ahead and makes strategic sacrifices, like studying for exams instead of playing games.

Question 4

The optimal policy stays the same.

Original rewards: $R = -1$ per step. A path with n steps gets $G = -n$. Maximizing this means minimizing n.

Modified rewards: $R = +1$ per step. A path with n steps gets $G = n$.

This seems backwards - longer paths give higher returns! But the episode ends when you reach the goal. You can't wander forever collecting rewards.

Both setups have the same optimal policy: reach the goal as fast as possible. In the first case, you minimize penalties. In the second case, you minimize opportunity cost. Adding a constant to all rewards doesn't change the relative preference between trajectories when episodes are finite and deterministic.

Question 5

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

The return satisfies $G_t = R_{t+1} + \gamma G_{t+1}$, so:

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

By linearity of expectation:

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

Using the tower property:

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

Expanding by summing over actions and next states:

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

The expectations become:

$$\mathbb{E}[R_{t+1}|S_t = s, A_t = a] = \sum_{s',r} p(s', r|s, a) \cdot r$$

$$\mathbb{E}[v_\pi(S_{t+1})|S_t = s, A_t = a] = \sum_{s',r} p(s', r|s, a) \cdot v_\pi(s')$$

Substituting and factoring:

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

Question 6

Part 1:

The Bellman equation: $\mathbf{v} = \mathbf{P}^\pi(\mathbf{R}^\pi + \gamma \mathbf{v})$

Expanding: $\mathbf{v} = \mathbf{P}^\pi \mathbf{R}^\pi + \gamma \mathbf{P}^\pi \mathbf{v}$

Rearranging:

$$\mathbf{v} - \gamma \mathbf{P}^\pi \mathbf{v} = \mathbf{P}^\pi \mathbf{R}^\pi$$

$$(\mathbf{I} - \gamma \mathbf{P}^\pi)\mathbf{v} = \mathbf{P}^\pi \mathbf{R}^\pi$$

Solution: $\mathbf{v} = (\mathbf{I} - \gamma \mathbf{P}^\pi)^{-1} \mathbf{P}^\pi \mathbf{R}^\pi$

Part 2:

Backgammon has $N = 10^{20}$ states. Matrix inversion takes $O(N^3)$ operations.

Total operations: $(10^{20})^3 = 10^{60}$

A supercomputer does about 10^{18} operations per second.

Time needed:

$$t = \frac{10^{60}}{10^{18}} = 10^{42} \text{ seconds} \approx 3 \times 10^{34} \text{ years}$$

The universe is only about 10^{10} years old. This would take trillions of times longer.

Part 3:

Direct analytical solutions don't scale. We need sampling-based methods like Monte Carlo or TD learning that learn from experience. These methods work with just a few samples and don't need to store values for all states. Function approximation with neural networks helps by generalizing across similar states.

Question 7

Part 1:

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

Part 2:

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

Part 3:

The formula in Part 1 needs $p(s', r|s, a)$ - the environment's transition probabilities. In model-free settings like Blackjack, we don't have access to this. We can't calculate which action is best using only $v_*(s)$ because we don't know where our actions will lead.

But $q_*(s, a)$ already bakes in the transition dynamics:

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

When we learn $q_*(s, a)$ from experience, we're implicitly learning what happens after each action through trial and error. After playing thousands of games, we learn that certain actions give bad results on average, without ever knowing the exact probabilities. In Part 2, we just pick $\arg\max_a q_*(s, a)$ directly.