IT 567 - Lab Exercise 2
Winter 2024-2025

Kewalramani, Tanay
202201362

24 January 2025

Problem Set 1: Exercises 3.7, 3.8, and 3.9

Exercise 3.7: Insufficient Reward Signal

The agent struggles to improve its ability to escape the maze due to an insufficient reward signal. A sparse reward system, where +1 is given for escaping and 0 otherwise, does not effectively guide the agent because it provides no intermediate feedback.

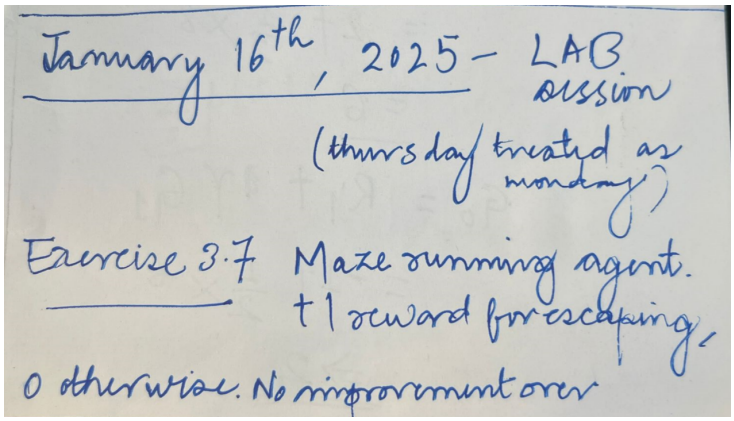

Figure 1: First part of solution for exercise 3.7.

[H]

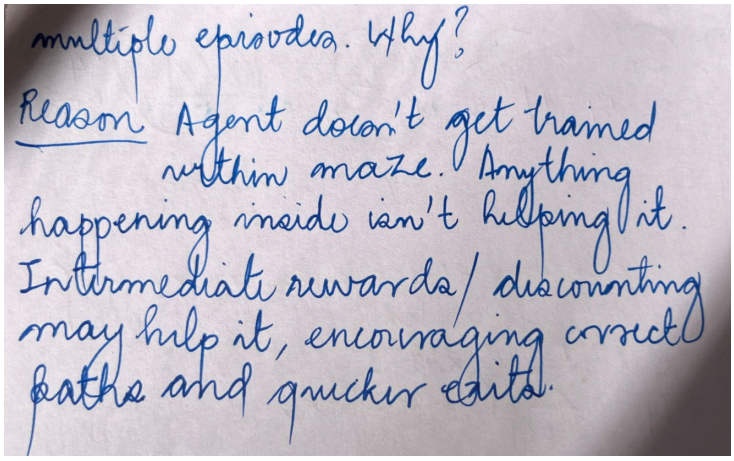

Figure 2: Second part of solution for exercise 3.7.

Exercise 3.8: Calculating Returns Backward

Given $\gamma = 0.5$ and rewards $R_1 = -1, R_2 = 2, R_3 = 6, R_4 = 3, R_5 = 2$, with $T = 5$, the returns G_t are computed as:

$$G_t = R_t + \gamma G_{t+1}.$$

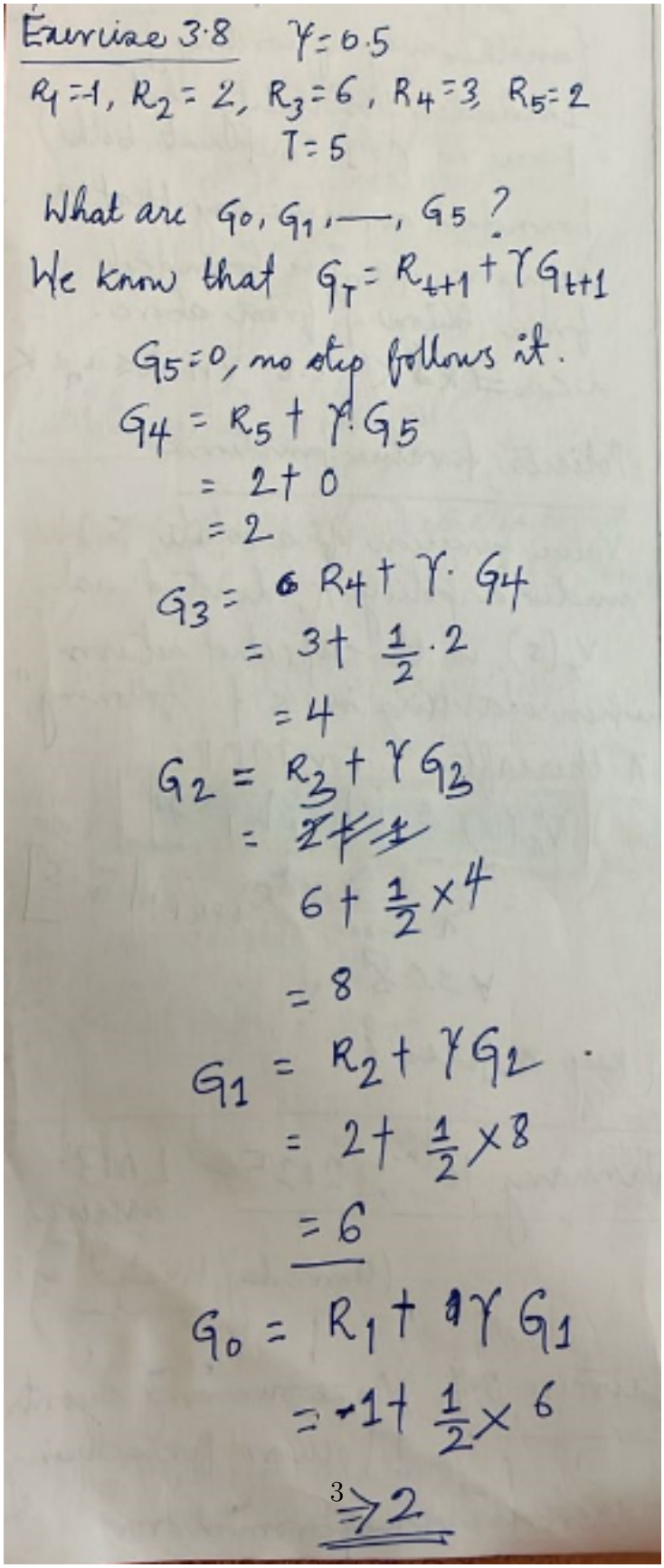

Figure 3: Solution for exercise 3.8.

Exercise 3.9: Infinite Rewards with Discount Factor

For $\gamma = 0.9$ and rewards $R_1 = 2$ followed by an infinite sequence of 7s, calculate G_t.

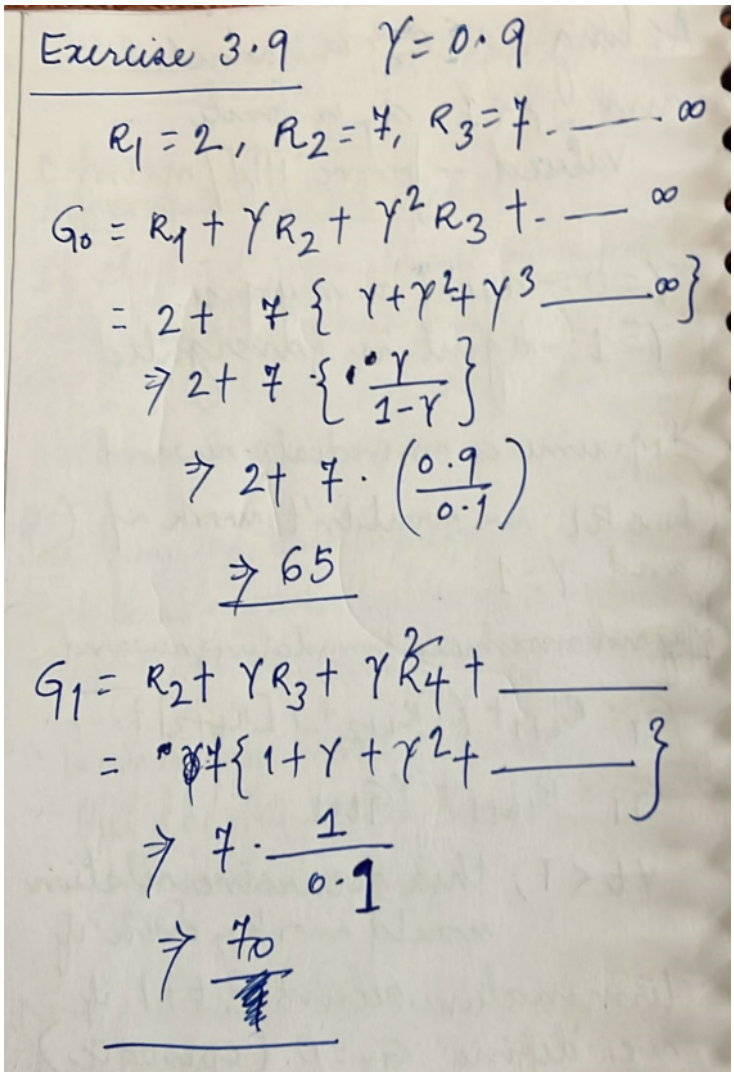

Figure 4: Second part of solution for exercise 3.7.

]

Exercise 3.6: Return in Pole-Balancing with Failure Reward

Problem Statement

Suppose you treated pole-balancing as an episodic task but also used discounting, with all rewards zero except for +1 upon failure. What would the return be at each time step? How does this return differ from that in the discounted, continuing formulation of this task?

Solution

Episodic Task with Discounting: In the episodic formulation:

- The task ends upon failure at time step K.
- Rewards are zero for all $t < K$ and +1 at $t = K$.

The return G_t at time t is:

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

Since $R_{t+k+1} = 0$ for $t + k + 1 < K$ and $R_K = 1$, the return simplifies to:

$$G_t = \gamma^{K-t}.$$

Key Points:

- The return G_t depends on the discount factor γ and the time difference $K - t$.
- The return decreases exponentially as t increases.

—

Continuing Task with Discounting: In the discounted, continuing formulation:

- The task does not terminate after failure.
- Rewards are +1 for every failure, occurring periodically at intervals K.

The return G_t is:

$$G_t = \sum_{i=1}^{\infty} \gamma^{iK-t}.$$

This is a geometric series:

$$G_t = \gamma^{K-t} + \gamma^{2K-t} + \gamma^{3K-t} + \dots$$

The sum of this infinite geometric series is:

$$G_t = \frac{\gamma^{K-t}}{1-\gamma^K}.$$

Key Points:

- The return G_t accounts for multiple future failures, discounted by γ^K for each additional failure.
- The value is higher than the episodic case due to the contribution from repeated failures.

—

Comparison of Episodic and Continuing Returns

- Episodic Task:

  $$G_t = \gamma^{K-t}.$$

  The return depends only on the next failure and is finite, as the task terminates after failure.

- Continuing Task:

  $$G_t = \frac{\gamma^{K-t}}{1-\gamma^K}.$$

  The return accumulates rewards from multiple failures, resulting in a higher value.

Key Difference: - The episodic task focuses only on the immediate failure, while the continuing task reflects the long-term impact of periodic failures.

Problem Set 3: Exercises 3.14, 3.15, and 3.16

Before we begin with the three exercises in this set, we're asked to simulate the gridworld problem on python.

Following is the python code for the Gridworld problem:

Code:

```
import numpy as np

# Gridworld setup
grid_size = 5
gamma = 0.9  # Discount factor
actions = ['N', 'S', 'E', 'W']
action_prob = 1 / len(actions)  # Equiprobable policy
rewards = np.zeros((grid_size, grid_size))  # Rewards
    for each state
state_values = np.zeros((grid_size, grid_size))  #
    State values to compute

# Special states and their transitions
special_states = {
    (0, 1): {'reward': 10, 'to': (4, 1)},  # State A
    (0, 3): {'reward': 5, 'to': (2, 3)}    # State B
}

# Transition function
def get_next_state_and_reward(state, action):
    x, y = state
    if state in special_states:
        return special_states[state]['to'],
            special_states[state]['reward']

    if action == 'N':
        x = max(x - 1, 0)
    elif action == 'S':
        x = min(x + 1, grid_size - 1)
    elif action == 'E':
        y = min(y + 1, grid_size - 1)
    elif action == 'W':
        y = max(y - 1, 0)
```

```
    # Check for boundary conditions
    if state == (x, y):
        return state, -1  # Penalty for hitting the
            wall
    else:
        return (x, y), 0  # No penalty for valid moves

# Iterative Policy Evaluation
threshold = 1e-4  # Convergence threshold
delta = float('inf')  # Initialize max change

while delta > threshold:
    delta = 0
    for i in range(grid_size):
        for j in range(grid_size):
            old_value = state_values[i, j]
            new_value = 0
            for action in actions:
                next_state, reward =
                    get_next_state_and_reward((i, j),
                    action)
                nx, ny = next_state
                new_value += action_prob * (reward +
                    gamma * state_values[nx, ny])
            state_values[i, j] = new_value
            delta = max(delta, abs(old_value -
                new_value))

# Print the state values
print("State Values for Equiprobable Random Policy:")
print(np.round(state_values, 2))
```

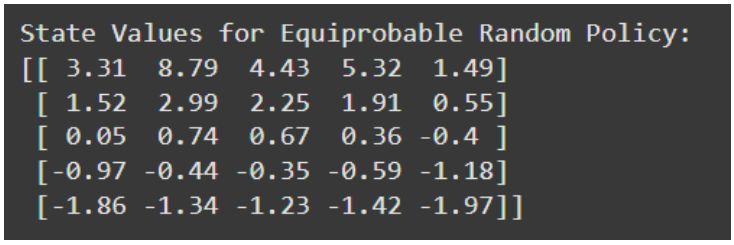

Figure 5: First part of solution for exercise 3.7.

Problem Set 3: Exercises 3.14, 3.15, and 3.16

Exercise 3.14: Bellman Equation

The Bellman equation for a state s is:

$$v^{\pi}(s) = \sum_{s'} P(s' \mid s, a) \left[R(s, a, s') + \gamma v^{\pi}(s') \right].$$

For the center state, $v(s) = 0.7$, and its four neighboring states have values $2.3, 0.4, -0.4, 0.7$, with equal transition probabilities to each neighboring state.

Solution: Using the Bellman equation for equiprobable transitions:

$$v(s) = \frac{1}{4} \left[R + \gamma(2.3) + R + \gamma(0.4) + R + \gamma(-0.4) + R + \gamma(0.7) \right].$$

Simplifying:

$$v(s) = R + \frac{\gamma}{4} \left(2.3 + 0.4 - 0.4 + 0.7 \right).$$

Substitute $\gamma = 0.9$ and $v(s) = 0.7$:

$$0.7 = R + \frac{0.9}{4} \cdot 3.0.$$

$$0.7 = R + 0.675.$$

Solve for R:

$$R = 0.7 - 0.675 = 0.025.$$

Answer: The reward $R = 0.025$ ensures the Bellman equation holds for $v(s) = 0.7$.

—

Exercise 3.15: Impact of Reward Signs

In the gridworld example: - Positive Rewards: Represent desirable states (e.g., goal states). - Negative Rewards: Represent undesirable states (e.g., falling off the grid). - Zero Rewards: Represent neutral states.

Solution: The agent optimizes its policy based on the relative differences between rewards, not their absolute values. To illustrate this, consider adding a constant c to all rewards. Let $v^\pi(s)$ be the value function for state s, defined as:

$$v^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid s_t = s \right].$$

Adding c to all rewards:

$$R'_{t+1} = R_{t+1} + c.$$

The new value function $v'^\pi(s)$ becomes:

$$v'^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t (R_{t+1} + c) \mid s_t = s \right].$$

Expanding:

$$v'^\pi(s) = v^\pi(s) + \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t c \mid s_t = s \right].$$

Since c is constant:

$$\sum_{t=0}^{\infty} \gamma^t = \frac{1}{1-\gamma}.$$

Thus:

$$v'^\pi(s) = v^\pi(s) + \frac{c}{1-\gamma}.$$

Conclusion: Adding c shifts all values by $\frac{c}{1-\gamma}$, but it does not affect the relative differences between states. Therefore, the agent's policy remains unchanged.

—

Exercise 3.16: Modified Bellman Equation

Adding a constant c to all rewards modifies the Bellman equation. Let the original Bellman equation be:

$$v^{\pi}(s) = \sum_{s'} P(s' \mid s, a) \left[R(s, a, s') + \gamma v^{\pi}(s') \right].$$

When c is added to all rewards:

$$R'(s, a, s') = R(s, a, s') + c.$$

The modified Bellman equation becomes:

$$v'^{\pi}(s) = \sum_{s'} P(s' \mid s, a) \left[(R(s, a, s') + c) + \gamma v'^{\pi}(s') \right].$$

Expanding:

$$v'^{\pi}(s) = \sum_{s'} P(s' \mid s, a) \left[R(s, a, s') + \gamma v'^{\pi}(s') \right] + \sum_{s'} P(s' \mid s, a) c.$$

Since $\sum_{s'} P(s' \mid s, a) = 1$:

$$v'^{\pi}(s) = v^{\pi}(s) + c.$$

Conclusion: Adding c shifts the value function by a constant c. The policy remains unaffected because the relative values of states are preserved.

—