

# Homework 2

PENG Qiheng

Student ID 225040065

November 11, 2025

Problem 1.

1.

state space  $\mathcal{S} = \{0, 1, \dots, n\}$

action space  $\mathcal{A} = \{'c', 'r'\}$

transition function  $P(s'|s, a) = \begin{cases} \frac{n-i}{n}, & a = 'c', s' = s + 1 \\ \frac{i}{n}, & a = 'c', s' = s - 1 \\ \frac{1}{n}, & a = 'r', \forall s' \in \mathcal{S} \\ 0, & \text{otherwise} \end{cases}$

reward function  $R(s, a, s') = \begin{cases} 1, & s' = n \\ 0, & \text{otherwise} \end{cases}$

optimal value function  $V^*(s) =$

$$\max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s, a) (R(s, a, s') + \gamma V^*(s'))$$

2. When  $n = 5, \gamma = 1$ , the optimal value function and policy are:

$$V^* = [0.0, 1.2857, 1.3929, 1.4643, 1.5714, 1.0]$$

$$\pi^* = [-', 'r', 'c', 'c', 'c', '-']$$

Problem 2.

$$V^* = R + \gamma P V^*$$

$$(I - \gamma P) V^* = R$$

$$V^* = (I - \gamma P)^{-1} R = \hat{V} = \Phi w$$

$$R = (I - \gamma P) \Phi w$$

So we need  $\Phi$  to satisfy the following equation:

$$R = (I - \gamma P) \Phi w$$

Thus we can guarantees that  $V^* = \Phi w$ .

Problem 3.

1. Iteration 1:

$$V_1(s_1) = \max\{R(s_1, a) + \gamma \sum_{s' \in \mathcal{S}} P(s_1, a, s') V_0(s')\}$$

$$= \max\{8 + 2.6, 10 + 1.2\} = 11.2$$

$$V_1(s_2) = \max\{R(s_2, a) + \gamma \sum_{s' \in \mathcal{S}} P(s_2, a, s') V_0(s')\}$$

$$= \max\{1 + 3.3, -1 + 5.3\} = 4.3$$

$$V_1(s_3) = \max\{R(s_3, a) + \gamma \sum_{s' \in \mathcal{S}} P(s_3, a, s') V_0(s')\}$$

$$= \max\{0, 0\} = 0$$

Iteration 2:

$$V_2(s_1) = \max\{R(s_1, a) + \gamma \sum_{s' \in \mathcal{S}} P(s_1, a, s') V_1(s')\}$$

$$= \max\{8 + 4.82, 10 + 1.55\} = 12.82$$

$$V_2(s_2) = \max\{R(s_2, a) + \gamma \sum_{s' \in \mathcal{S}} P(s_2, a, s') V_1(s')\}$$

$$= \max\{1 + 4.65, -1 + 6.89\} = 5.89$$

$$V_2(s_3) = \max\{R(s_3, a) + \gamma \sum_{s' \in \mathcal{S}} P(s_3, a, s') V_1(s')\}$$

$$= \max\{0, 0\} = 0$$

Thus the value function after two iterations is:

$$V_2(s_1) = 12.82$$

$$V_2(s_2) = 5.89$$

$$V_2(s_3) = 0$$

2. According to the calculation in part (a), we have:

$$\begin{aligned} \forall k \geq 2, \quad V_k(s_i) &\geq V_2(s_i) \\ \Rightarrow V^*(s_1) &\geq 12.82, \quad V^*(s_2) \geq 5.89, \quad V^*(s_3) = 0 \end{aligned}$$

Thus we can get:

$$\begin{aligned} Q_{k+1}(s_1, a_1) - Q_{k+1}(s_1, a_2) &= -2 + 0.1V_k(s_1) - 0.4V_k(s_2) \geq 0 \\ Q_{k+1}(s_2, a_1) - Q_{k+1}(s_2, a_2) &= 2 - 0.2V_k(s_1) \leq 0 \\ Q_{k+1}(s_3, a_1) = Q_{k+1}(s_3, a_2) &= 0 \\ \forall k \geq 2 \end{aligned}$$

And we let  $\pi_k(s_3) = a_1$ , then the policy is:

$$\forall k \geq 2, \quad \pi_k(s_1) = a_1, \quad \pi_k(s_2) = a_2, \quad \pi_k(s_3) = a_1$$

Hence the policy will not change anymore after iteration 2, i.e.  $\pi_k(s) = \pi_2(s), \forall k \geq 2$ .

### 3. Key difference:

Value iteration directly iterate the value function, update the value through the Bellman optimal equation until convergence. Then extract the optimal strategy from the final value function. It does not explicitly maintain policies, and every update requires traversal of all actions.

Policy iteration alternates between policy evaluation and policy improvement. The policy evaluation stage calculates the value function of the current strategy (possibly iterating to convergence), and the policy improvement stage updates the strategy based on the current value function. It explicitly maintains the strategy and usually converges faster.

### **Applicable scenarios:**

Value iteration is more suitable for situations with large state spaces and limited computing resources, as it is simple to implement but may converge slowly (especially when  $\gamma$  approaches 1). When no intermediate strategy is needed, value iteration is more appropriate.

Policy iteration is more suitable for situations where the state space is medium and requires rapid convergence, as the policy improvement steps may reach the optimal strategy faster. When intermediate strategies such as online learning are needed, policy iteration is more optimal.

#### 4. Unsure.

Usually, it cannot be guaranteed to obtain the optimal strategy. Due to insufficient policy evaluation, value function errors may propagate to policy improvements, which may not satisfy the Bellman optimal equation. However, in some cases (such as MDP having a simple structure and a good initial strategy), it may accidentally converge.

Problem 4. Here is the code link: [https://colab.research.google.com/drive/174K8FCc1RViczj1ufL5V8MxkhW-\\_WVwS?usp=sharing](https://colab.research.google.com/drive/174K8FCc1RViczj1ufL5V8MxkhW-_WVwS?usp=sharing)

With using 5 different random seeds, both Value Iteration and Policy Iteration methods achieve similar average score  $1.0000 \pm 0.0000$ .

The output of the code is as follows:

```
--- Experiment with seed 666 ---
Converged after 1360 iterations
Score obtained 1.0
Best score = 1.00.
Policy converged after 6 iterations
Best score = 1.00. Time taken = 0.0122 seconds

--- Experiment with seed 667 ---
Converged after 1360 iterations
Score obtained 1.0
Best score = 1.00.
Policy converged after 4 iterations
Best score = 1.00. Time taken = 0.0183 seconds

--- Experiment with seed 668 ---
Converged after 1360 iterations
Score obtained 1.0
Best score = 1.00.
Policy converged after 6 iterations
Best score = 1.00. Time taken = 0.0115 seconds

--- Experiment with seed 669 ---
```

```
Converged after 1360 iterations
Score obtained 1.0
Best score = 1.00.
Policy converged after 4 iterations
Best score = 1.00. Time taken = 0.0116 seconds
```

```
--- Experiment with seed 670 ---
Converged after 1360 iterations
Score obtained 1.0
Best score = 1.00.
Policy converged after 7 iterations
Best score = 1.00. Time taken = 0.0115 seconds
```

```
==== Comparison Results ====
Value Iteration - Mean: 1.0000, Std: 0.0000
Policy Iteration - Mean: 1.0000, Std: 0.0000
Both methods achieved similar average scores.
Both methods showed similar consistency.
```