Reinforcement Learning Coursework 1

Jiqiu Hu
CID: 02429885

Question 1: Dynamic Programming

1. I choose **Policy Iteration** Algorithm to solve this Maze problem because I aim to acquire more accurate value functions.

1.1 Parameters setting

Table 1: Parameters setting in DP

Parameters	values
Probability (p)	0.82
Discount factor (γ)	0.96
Reward state (R_i)	R_1
Policy Evaluation threshold (θ)	0.0002
Iteration	100

a. **Iteration:** Given the knowledge about environment, DP can easily find the optimal value function and policy, so that I set the number of iterations a small value.
b. **Threshold:** I set the Policy Evaluation threshold to be as small as possible to compute value function more accurately.

1.2 Design and Assumptions

a. **Initialize** every state's policy and value function of all state to 0.
b. **Condition for finishing policy evaluation:** To conclude a policy evaluation, the maximum difference between the updated value function and the one from the previous iteration must be smaller than θ, considering all states.
c. **Condition for finishing policy iteration algorithm:** The policy iteration terminates when the current policy becomes stable, meaning that the updated policy matches non-updated ones.

2. **Report the graphical representation of optimal policy and optimal value function.**

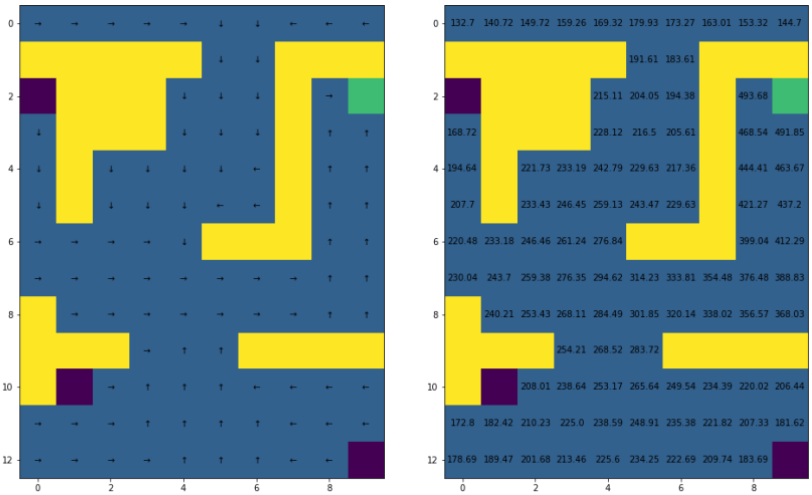


Figure 1: optimal policy and optimal value function in DP

3. **Discuss the effects of Probability (p) and Discount factor (γ) on optimal value function and value policy.**

3.1 Change the value of p

a. When $p = 0.10$

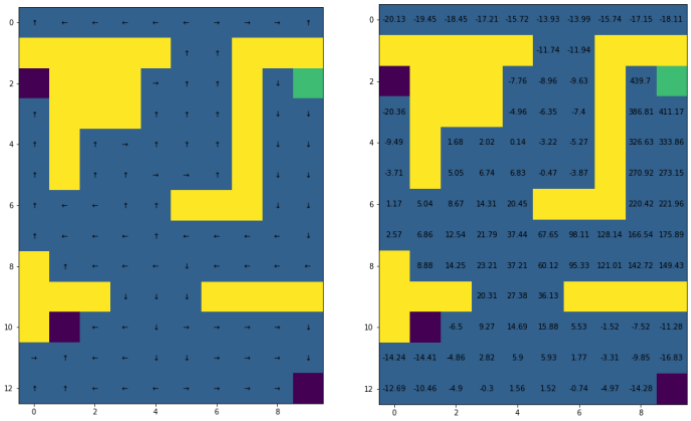


Figure 2: optimal policy and optimal value function in DP ($p = 0.10$)

b. When $p = 0.25$

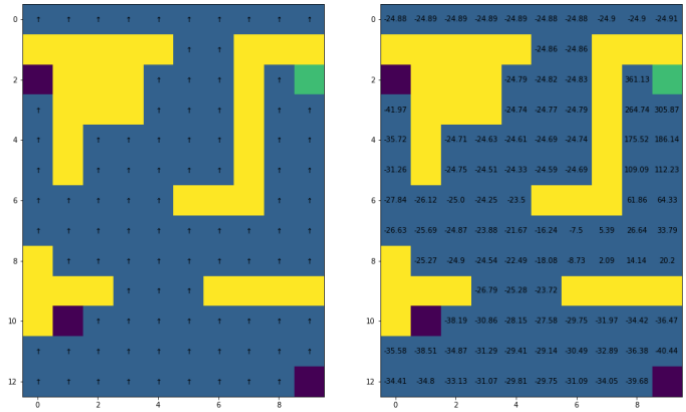


Figure 3: optimal policy and optimal value function in DP ($p = 0.25$)

c. When $p = 0.82$

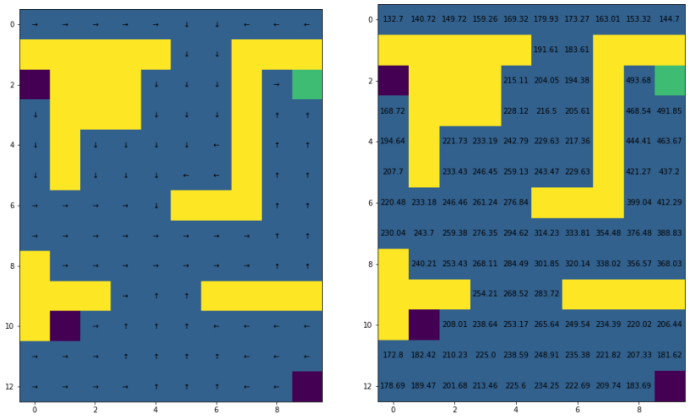


Figure 4: optimal policy and optimal value function in DP ($p = 0.82$)

Discussion: When $p < 0.25$, there is a less probability to succeed and lead to the expected direction, making it more probable to choose any of the other three directions. As shown in

Figure 2, the optimal policies for every state are reversed. When $\boldsymbol{p = 0.25}$, the probabilities of choosing every direction are equal, resulting in no policy improvement. When $\boldsymbol{p > 0.25}$, there is a higher probability to succeed and lead to the expected direction, allowing for identification of the optimal policy through policy improvement.

3.2 Change the value of γ

a. When γ**=0.3**

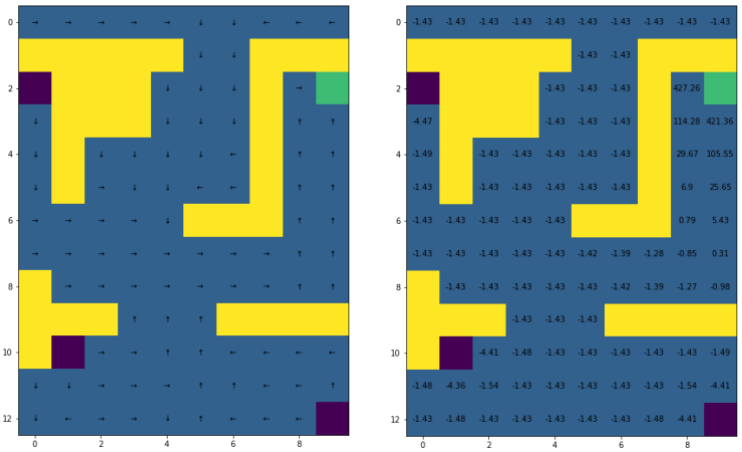


Figure 5: optimal policy and optimal value function in DP (γ=0.3)

b. When γ**=0.7**

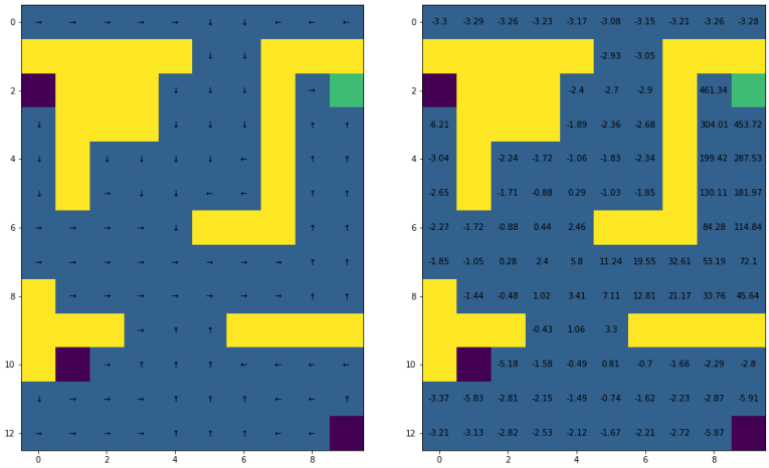


Figure 6: optimal policy and optimal value function in DP ($\gamma = 0.7$)

Discussion: Based on the figures above, there is no significant difference in the optimal policy when varying γ. However, when γ is close to 1 (γ=0.7), the algorithm will more focus on future potential rewards, which slightly affects the value function.

Question 2: Monte-Carlo Reinforcement learning

1. I choose **on-policy every-visit MC control (for ϵ-soft policies)** to solve this Maze problem.

1.1 Parameters setting

Table 2: Parameters setting in MC

Parameters	values
Probability (p)	0.82
Discount factor (γ)	0.96
Reward state (R_i)	R_1
Episodes	5000
Exploration (ϵ)	$\frac{1}{e}$ (e is the number of training episodes)
Learning rate(α)	0.25

a. Episode parameter: In order to enable MC agent explore as many states as possible and avoid getting trapped in a local optimum, I tried setting a relatively large number of episodes. However, if the number exceeds 7000, the learning curve becomes very volatile. Therefore, I finally chose 5000 as the number of episodes, under which the learning curve is optimal.

b. Exploration parameter: Because if ϵ reduces to zero with $\epsilon_k = \frac{1}{k}$, the ϵ-greedy operation is GLIE (Greedy in the Limit with Infinite Exploration). At the beginning, there is little information about the environment, and I want the agent to explore more actions. After many episodes, the policy becomes increasingly deterministic, selecting the action with the highest estimated Q-value in each state, ensuring comprehensive exploration.

c. Learning rate: I used incremental estimation updates in MC, where the learning rate controls the rate of forgetting old episodes. When $\alpha > 0.4$, the learning curve does not converge. When $\alpha = 0.3$, the agent does not learn and the learning curve remains flat. After many attempts, I found that $\alpha = 0.25$ ensures convergence.

1.2 Design and Assumptions

a. Initialize every state's policy and value function (V) to 0. Randomly initialize action-state function (Q).

b. Generate a sample trace: The algorithm uses ε-policy to choose actions, meaning that if $\pi(a|s) > t$ (where t is a random value), the agent will choose the optimal action 'a'; otherwise, it will randomly select an action from the action set A. Then, if the agent reaches terminal states or the maximum number of steps, the generation process will conclude.

c. Choose every-visit: I have opted for every visit instead of first visit in MC control, as first visit provides the agent with less experience, hindering effective learning and causing the total reward to diverge. By choosing very visit, the algorithm gains more experience, leading to better learning outcomes.

d. Incremental estimation updates: I choose this approach to update Q function without having to store sample traces.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_t - Q(S_t, A_t))$$

e. Policy improvement: After updating the Q function of a pair of (S_t, A_t), the algorithm also updates ϵ-policy for S_t. Subsequent episode will use updated policy to generate a new sample trace.

2. Report the graphical representation of optimal policy and optimal value function.

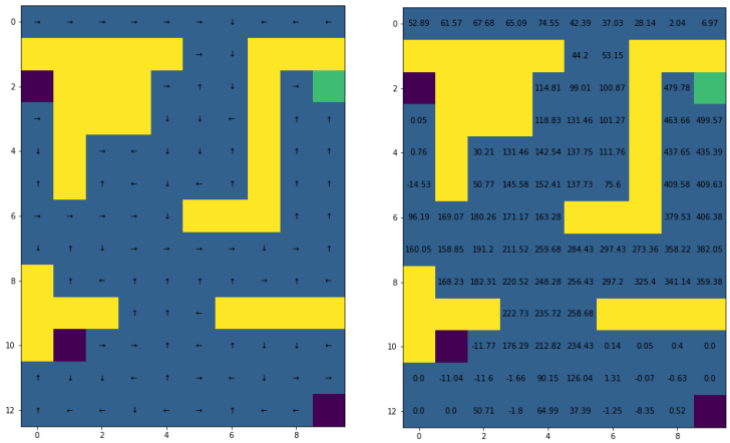


Figure 7: optimal policy and optimal value function in MC

3. Plot the learning curve

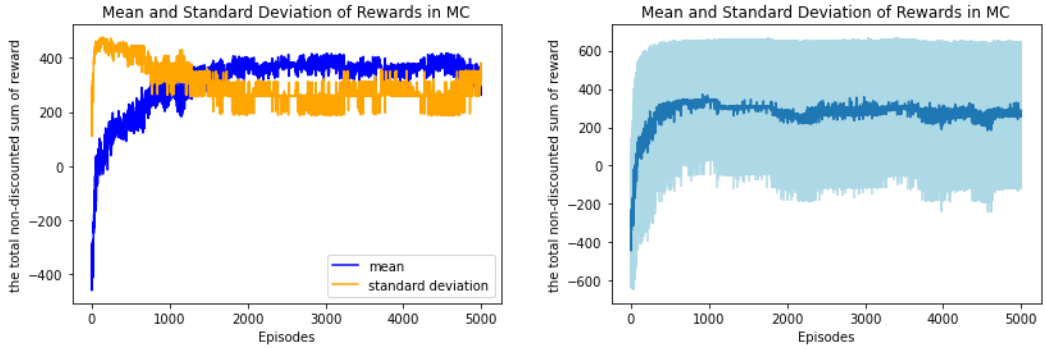


Figure 8: Learning curve in MC

Discussion: In the **Figure 8**, the left picture displays mean and standard deviation, distinguished by different colors. The right picture displays the distribution of total rewards using error bars. I believe that because MC learns from complete episodes, and each episode is likely to be different, it results in high standard deviations.

Question 3: Temporal Difference Reinforcement Learning

1. I choose **SARSA** to solve this Maze problem. In on-policy methods, we use the same policy to learn and improve, ensuring policy stability. Moreover, we do not require additional space or computational resources for another policy.

1.1 Parameters setting

Table 2: Parameters setting in TD

Parameters	values
Probability (p)	0.82
Discount factor (γ)	0.96
Reward state (R_i)	R_1
Episodes	500
Exploration (ϵ)	$\frac{1}{e}$ (e is the number of training episodes)
Learning rate(α)	0.5

a. Episode parameter: I experimented with different episodes parameters and observed that the learning curve generally converges and stabilizes after 200 episodes. Consequently, I chose 500 episodes to ensure comprehensive exploration and avoid the risk of the optimal policy being localized.

b. Exploration parameter: To ensure the convergence of the learning curve, I set $\epsilon = \frac{1}{e}$ (where e is the number of training episodes) following GLIE strategy.

c. Learning rate: According the convergence theorem of SARSA, we can choose reciprocals of powers of 2 of the steps. However, using this decaying parameter led to a divergent learning curve. After several attempts, I found that using certain constants as learning rate value yielded better results. Among these attempts, $\alpha = 0.5$ proved to be the most effective choice.

1.2 Design and Assumptions

a. Initialize every state's policy and value function (V) to 0. Randomly initialize action-state function (Q).

b. Choose actions: The algorithm uses ε-policy to choose actions, which means if $\pi(a|s) > t$ (where t is a random value), the agent will choose the optimal action 'a'; otherwise, it will randomly select an action from action set A.

2. Report the graphical representation of optimal policy and optimal value function.

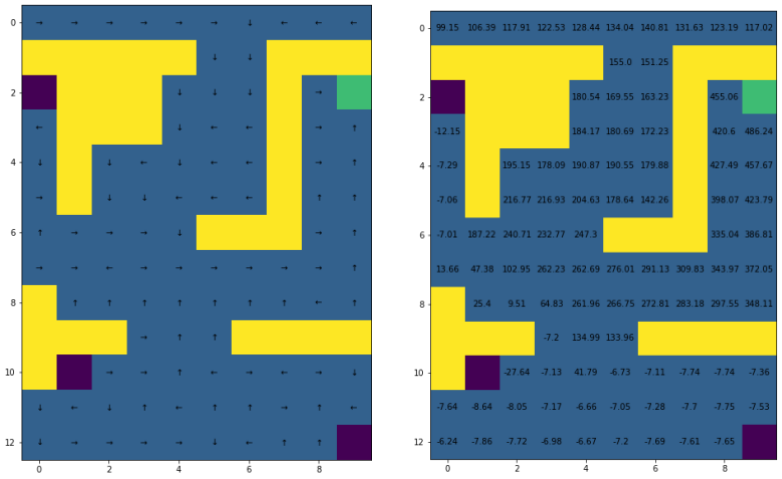
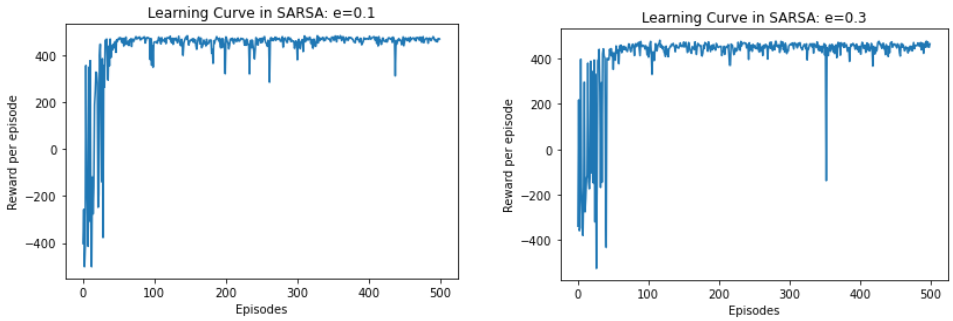


Figure 9: optimal policy and optimal value function in TD

3. Discuss the effect of the Exploration parameter ϵ and the Learning rate α on learning curves.

3.1 Change the value of ϵ



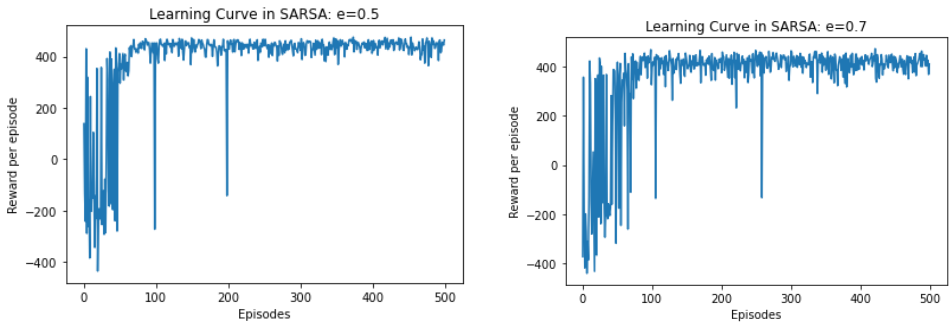


Figure 10: Learning Curves using different Exploration parameter (ϵ)

Discussion: In ϵ-greedy policies, a high value of ϵ indicates that the agent is more likely to explore the environment or try different actions. In contrast, a low value of ϵ means the agent is more likely to follow the optimal policy. As depicted in **Figure 10**, it is increasingly difficult to converge within the first 100 episodes as ϵ increases.

3.2 Change the value of α

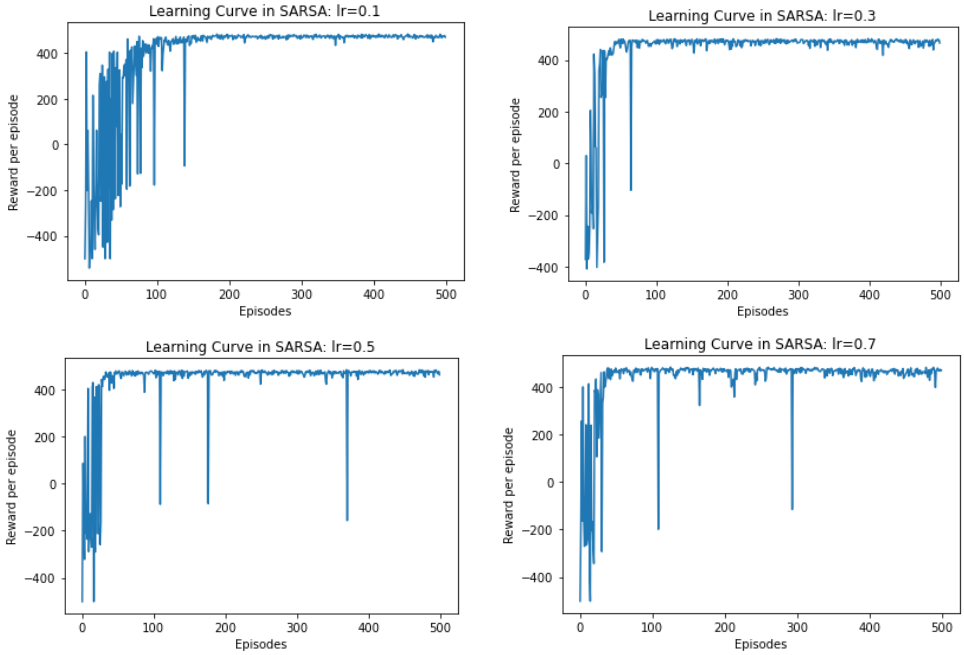


Figure 11: Learning Curves using different Learning rate (α)

Discussion: the parameter α controls the rate of forgetting old episodes. A high value of α means the agent gives more value to current data. In contrast, A low value of α means the agent relies on more past experiences, hindering the ability to find optimal policies. As shown in **Figure 11**, it is increasingly hard to converge within the first 100 episodes as ϵ decreases.