Reinforcement Learning
Coursework 1: Maze Environment

Charlize Yang
CID: 01666113

November 8, 2021

1 Dynamic Programming

Q1.1

I used the Value Iteration algorithm in Dynamic Programming (DP) to find the optimal policy and value function, shown in Figures 1(a) and 1(b). The Value Iteration algorithm truncates the policy evaluation step after one sweep of the value function and ensures a faster convergence of policy iteration. I chose the threshold value to be $\theta = 0.001$ in policy evaluation, ensuring a relatively fast convergence of the algorithm while giving two-decimal-point accuracy of the value function. Such accuracy is sufficient given that all the absorbing states have large absolute rewards (500 and -50).

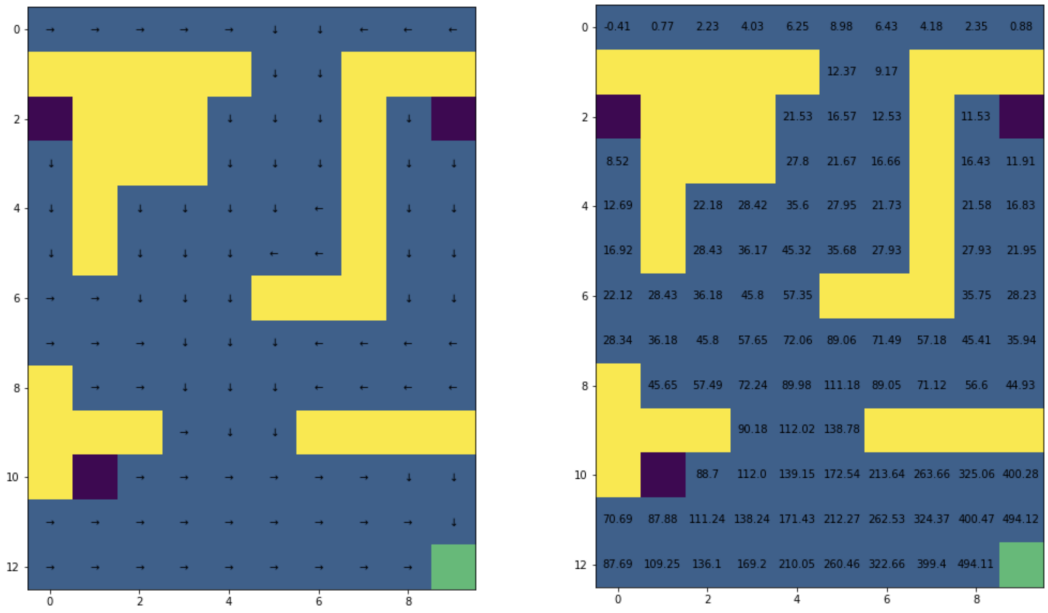
Q1.3

The parameter values p and γ impact the optimisation process. A larger p indicates a higher probability that the agent performs the chosen action in the algorithm, thus forming a shorter path towards the goal (i.e. a better policy) and better estimating the state value functions. A higher γ means the agent retains more future rewards, leading to more considerable differences between the iterative value function estimates in the algorithm. As a result, we will have slower convergence in the optimal value functions and a longer path to the goal.

2 Monte-Carlo Reinforcement Learning

Q2.1

I adopted the iterative implementation of the on-policy first-visit MC control algorithm to find the optimal policy and value function, shown in Figures 2(a) and 2(b). This method is computationally efficient compared to batch learning and every-visit MC, only appending the return from a state's first occurrence and throwing away



(a) DP Optimal Policy (b) DP Optimal Value Function

Figure 1: Optimal policy and value function computed using the Value Iteration algorithm in DP.

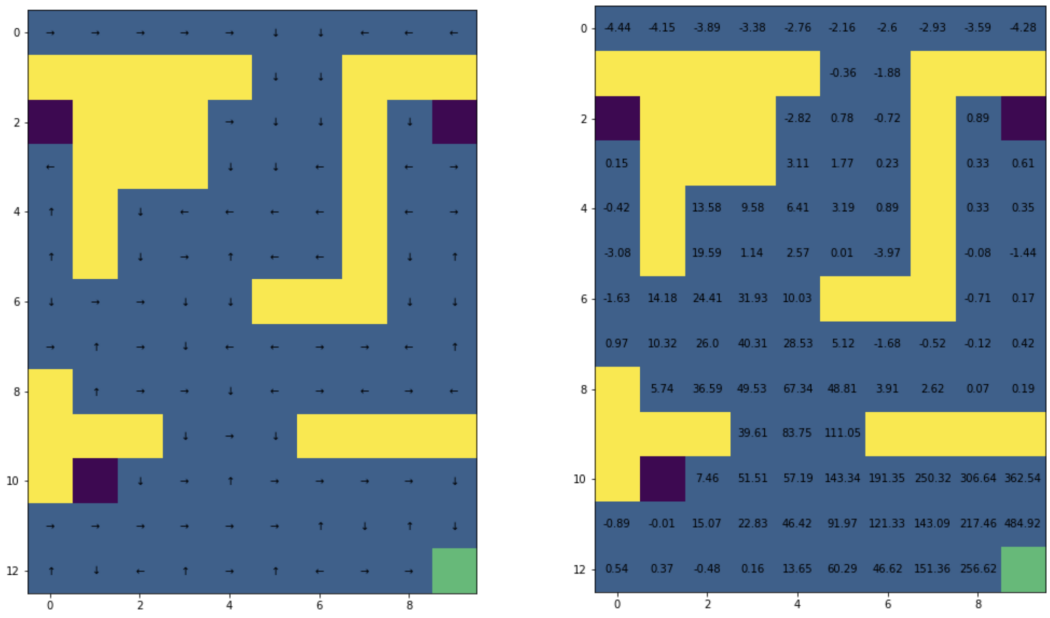
the transition after each iteration. I trained the MC agent over 1000 episodes using an ϵ-greedy policy, and set $\epsilon = 0.1$ to ensure a reasonable chance of exploring all other actions without immediately settling for a policy. In empirical testing, $\epsilon = 0.1$ also yields the highest total non-discounted sum of reward. The learning rate (i.e. the rate of forgetting old episodes) is set to be $\alpha = 0.05$ to ensure sufficient numbers of exploration to all states. It also produces the best results in testing.

Q2.3

The sources of variability in optimal policy and value function come from the randomly generated MCMC sample traces, which are only partially representative of the unknown Markov Decision Process (MDP). In each MC-learning run, only finitely many sample traces are generated, which are different from the last time, thus yielding different policy and value function estimates. A sufficient number of replications would be 100 times for 1000 episodes of traces. This was found as the best empirical number of repetitions to reduce noise in the averaged learning curves with a reasonable training time. In Figure 3, we see that the agent indeed learns over episodes where the learning curve becomes flat relatively quickly.

Q2.5

As ϵ increases, the learning curve shows slightly higher variance and lower final rewards. This makes sense as the agent now has more chances to explore other actions than the optimal policy with potentially much lower rewards. As α increases,



(a) MC Optimal Policy

(b) MC Optimal Value Function

Figure 2: Optimal policy and value function computed using the iterative implementation of the on-policy first-visit MC control algorithm.

the learning curves become flat at an earlier stage, implying a quicker learner. This is because, as we shift our focus to more recent episodes, we are pulling the estimated value function faster towards the optimal value. In contrast, a lower α gives learning curve with slower convergence, higher final rewards and lower variance, as we now have more information from the previous episodes for policy updates.

3 Temporal Difference Reinforcement Learning

Q3.1

I used the Q-learning algorithm, which is an off-policy Temporal Difference (TD) control algorithm to find the optimal policy and value function, shown in Figures 4(a) and 4(b). The off-policy algorithm assumes learning a policy given experience off that policy under the coverage assumption. Q-learning allows us to improve both the behaviour and target policies on each step without explicitly writing the policies. Again, I trained the TD agent for 100 repetitions over 1000 episodes using an ϵ-greedy policy. The learning curve of the TD agent is shown in Figure 5. I set $\epsilon = 0.1$ and the learning rate $\alpha = 0.05$, following a similar reasoning in Q2.1. These parameter values also yield the highest total non-discounted sum of reward in empirical testing, outperforming various choices of decaying ϵ and α.

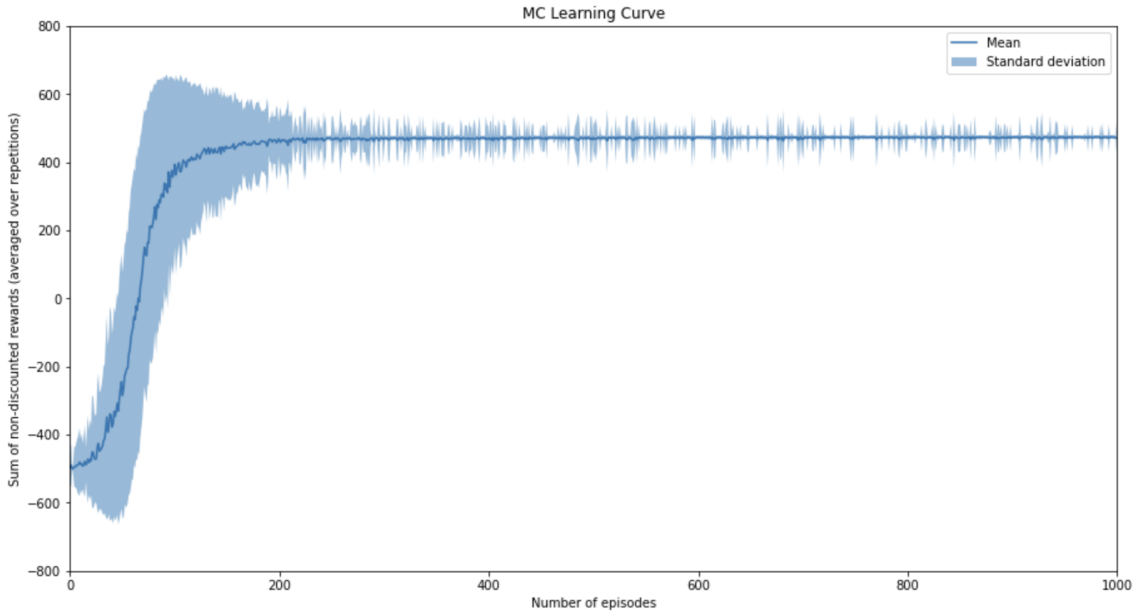


Figure 3: Learning curve for the MC agent.

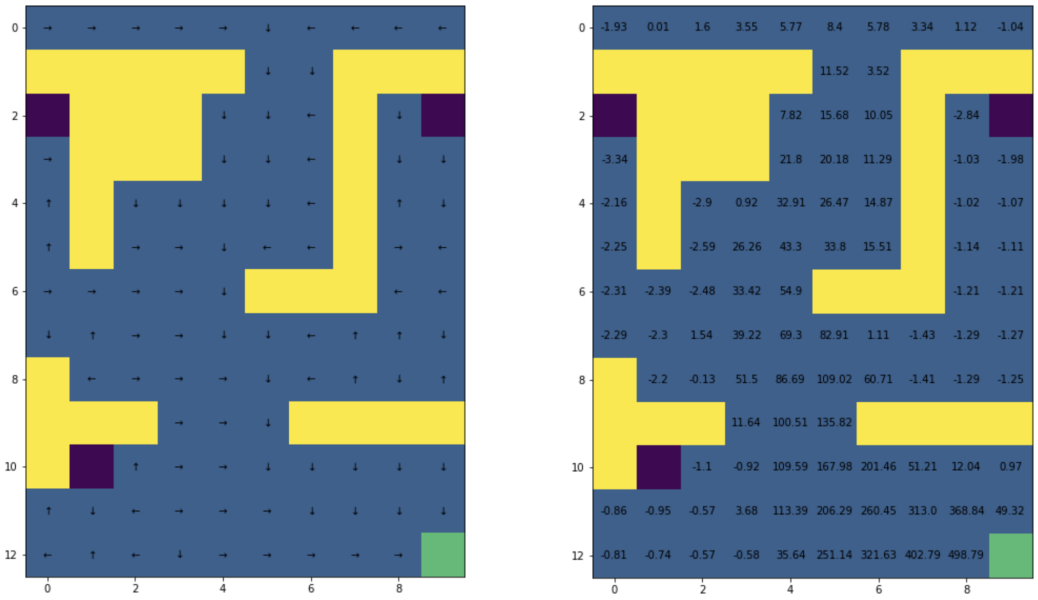
Q3.4

The effects of varying ϵ and α on the learning curves are similar as described in Q2.5. As ϵ increases, the learning curves show higher variance and lower final rewards, as the agent has more chances to explore other actions with potentially lower rewards. A lower α gives a learning curve with slower convergence, lower variance and higher final rewards, as we collect more information from the previous episodes. In this case, choosing a decaying ϵ with $\epsilon_k = \frac{1}{k}$ yields comparable rewards to my choice of $\epsilon = 0.1$. A possible explanation is that the agent becomes more confident in choosing a greedy policy as it learns more from experience.

4 Comparison of Learners

Q4.2

The mean square error for TD agent decreases quicker with more episodes, meaning that when estimating the optimal value function, the TD agent learns more quickly than the MC agent. This makes sense, as TD learner improves the estimates using only estimates from the next states, thus is more efficient than MC learner which needs to learn from complete returns. We expect the MC learner to have lower bias than the TD learner, so with a larger number of episodes, I expect the eventual mean square error for the MC learner will be smaller than TD. Although in this case, the high bias of TD learner may be reduced as we randomly initiate the starting state for each episode.



(a) TD Optimal Policy (b) TD Optimal Value Function

Figure 4: Optimal policy and value function computed using the Q-learning: off-policy TD control algorithm.

Q4.4

As the estimation error for value function decreases, the agent obtain more rewards, as they are choosing the most optimal action in each state which maximises the value function. This shows that in order to gain higher rewards, it is vitally important to have a good estimation of the optimal value function.

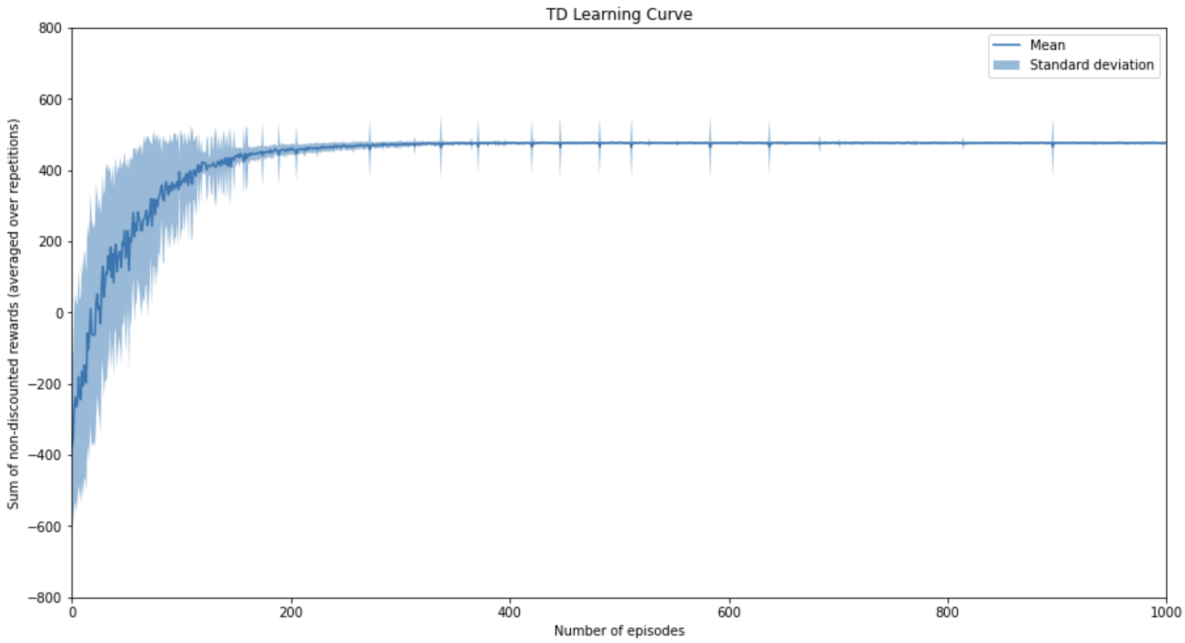


Figure 5: Learning curve for the TD agent.