Imperial College London

COURSEWORK 1

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Reinforcement Learning

Author:
Konstantinos Mitsides (CID: 01857560)

Date: November 3, 2023

Question 1

1.

I have chosen to use the Value Iteration (VI) algorithm. Both Policy Iteration (PI) and VI algorithms have been tested and consistently converged to the same optimal value function and policy. VI consistently outperformed PI in terms of speed. The only parameter used for VI is a threshold value of 0.0001, which serves as both a convergence criterion and a stopping condition. The small threshold value ensures that the values have reached a sufficient level of convergence and automatically stops the algorithm when further iterations are unlikely to yield significantly different results. We assume that the environment can be modeled as an MDP.

2.

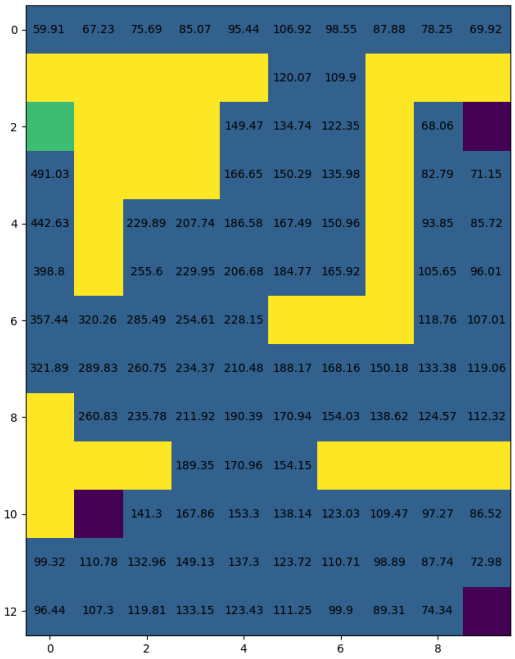

Figure 1: Optimal Value Function

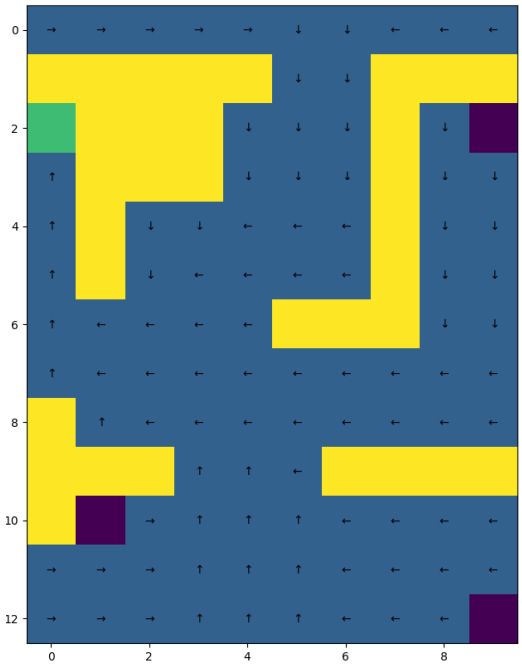

Figure 2: Optimal Policy

3.

Effect of p, with $\gamma = 0.92$

Recalls:
(A) The chosen action has a probability p to succeed and lead to the expected direction. If it fails, it has equal probability to lead to any other direction with probability of $\frac{1-p}{3}$.
(B) The optimal policy is deterministic. Therefore, the higher the probability we move towards the optimal direction, the higher the optimal value function.

For p = 0.25:
Here we have $p = \frac{1-p}{3}$, considering (A), this implies that the optimal policy can be randomly selected to be whatever. In fact, any policy is an optimal policy in this case. Considering (B), this implies that we should expect to see the lowest optimal value function in this case, as we cannot influence our path and thus cannot influence the optimal value function.
For p < 0.25:
In this case, $p < \frac{1-p}{3}$ and thus considering (A), we expect an optimal optimal policy that chooses actions with expected directions different to the optimal ones. Considering (B), it follows that the lower the value of p, the higher the possible optimal value function. However, we still cannot be expecting a very high optimal value function, as the highest probability $\frac{1-p}{3}$ can get, is $\frac{1}{3}$.
For p > 0.25:
Here we have $p > \frac{1-p}{3}$, and thus considering (A), the optimal policy chooses actions with expected direction equal to the optimal ones. The higher the value of p, the higher the optimal value function. Here, we can achieve maximum optimal value function by setting p=1.

Conclusion:
Our p value has been set equal to 0.86. Based on the above, the optimal policy should aim to move the agent towards the positive rewarding absorbing state. This implies that my optimal value function should be sufficiently high but not necessarily the highest possible, as there is still some uncertainty in the outcome of agent's actions.

Effect of γ, with p = 0.86

Investigating the function within the Policy Improvement algorithm used to calculate the optimal action given a particular current state,

$$\pi(\mathbf{s}) = \text{argmax}_{\text{a}} \sum_{s'} \mathcal{P}^{\text{a}}_{\text{ss}'} \left[\mathcal{R}^{\text{a}}_{\text{ss}'} + \gamma \text{V}(\text{s}') \right]$$

we see that γ is multiplied by the value function of the next state, which is considered fixed in the algorithm, and not dependent on the action value a. This means that γ does not affect the output of the argmax function in each Policy Improvement iteration and, consequently, the choice of the optimal policy in each iteration. γ however, affects the output of the Bellman's equation used to find the optimal function given a particular current state and policy within the Policy Evaluation algorithm,

$$\text{V}(\text{s}) = \sum_{s'} \mathcal{P}^{\pi(\text{s})}_{\text{ss}'} \left[\mathcal{R}^{\pi(\text{s})}_{\text{ss}'} + \gamma \text{V}(\text{s}') \right].$$

Since in the Policy Iteration algorithm we keep iterating through the Policy Evaluation and Policy Improvement algorithm, we expect changing the value of γ to have some effect (indirectly) on the optimal policy, however, not to a significant extent. On the other hand, considering the Bellman's equation, we see that the value of γ has a direct effect on the calculation of the optimal value function. In fact, γ's essence lies in discounting the present value of future rewards. In our problem, this translates to the following relationship: the lower the value of γ, then the further away the agent is from the positive rewarding absorbing state, the lower the optimal

value function of its current state, and vice versa.

Conclusion: Our γ has been set equal to 0.92. It follows from the above that states further away from the positively rewarding absorbing state will have a lower optimal value function than those closer to it. However, this difference is to an extent such that if the agent is able to follow the path towards the positive rewarding absorbing state from its current state, then its current state will have a positive optimal value function.

Question 2

1.

I have chosen to use the "on-policy ϵ-greedy first-visit MC control" algorithm. I preferred on-policy over off-policy to avoid the off-policy's potential problem of learning only from the tails of episodes, and thus not utilizing the potentially greedy actions. The possible frequent occurrence of non-greedy actions, leads to a very slow learning, especially for states appearing early in the episodes. Additionally, I have used first-visit MC, as the settings behind the generation of episodes guarantees us that each episode will terminate at most 500 steps, and therefore since 500 is relatively small, but also the fact that we run 10,000 episodes, the marginal benefit of using additional data is outweighed by the decrease in computational efficiency. We assume that the environment can modelled as an MDP, and that is stationary.

Parameters

Exploration parameter: ϵ

ϵ has been set to 1 for the first 1000 episodes, so that our agent takes actions purely randomly and thus have the chance to experience some states that otherwise it wouldn't. Although, the value function and policy will be a very bad estimate to the optimal ones within these first 100 episodes, at least the agent can compare which states are better when it ever appears in a state further away from the positive rewarding absorbing state. By experimenting, I found that adding this step in my algorithm resulted to better optimal policies in states further away from the positive rewarding absorbing state. Following the initial 1000 episodes, ϵ is updated by multiplying itself by 0.999 after each subsequent episode, but only if the total reward from that episode is positive. This ensures that the agent's probability of taking a particular action, as dictated by its policy as optimal, increases, only if it actually effectively learns and returns positive rewards. The multiplication of 0.999 is to ensure that ϵ approaches zero after the rest of the episodes, and thus the policy converges to the optimal one.

Learning rate: $\alpha = 0.05$

$\alpha = \frac{1}{N_{S_t}}$ gives an equal weight on a state update for each episode, i.e. never forgets old episodes, whereas $\alpha = 0.05$ starts giving more weight on recent episodes, forgetting older ones. Considering the latter, and our exploration parameter settings, I chose to go with $\alpha = 0.05$ so that my agent forgets older episodes, especially the first 1000 episodes, and emphasizes more on the recent ones, where the exploration rate decreases, and the agent becomes more experienced and makes better estimations for the optimal value function and policy. $\alpha = 0.05$ has indeed constantly performed better than the average ($\alpha = \frac{1}{N_{S_t}}$), improving the estimated optimal val-

ues for states further away from the positive rewarding absorbing state, especially the starting states. Note though, that if we set the number of episodes parameter higher, the variable α might have outperformed the constant (the explanation for this, is similar to the one I use at Q.3.3). The value 0.05 was chosen because it consistently outperformed various constant alpha values in terms of average performance.

Number of episodes:

The number of episodes has been set to 10,000 so that the agent has the chance to do some excessive exploration at the start without affecting the convergence of the algorithm to the optimal value function and policy. We assume that this number is sufficient for convergence.

2.

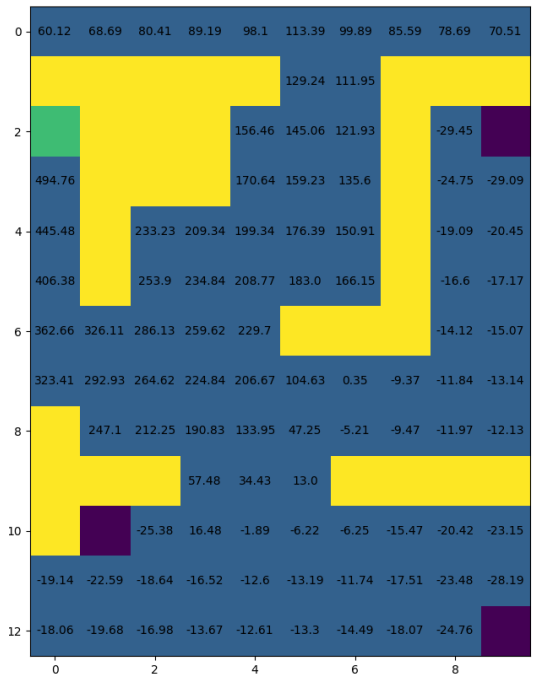

Figure 3: Optimal Value Function

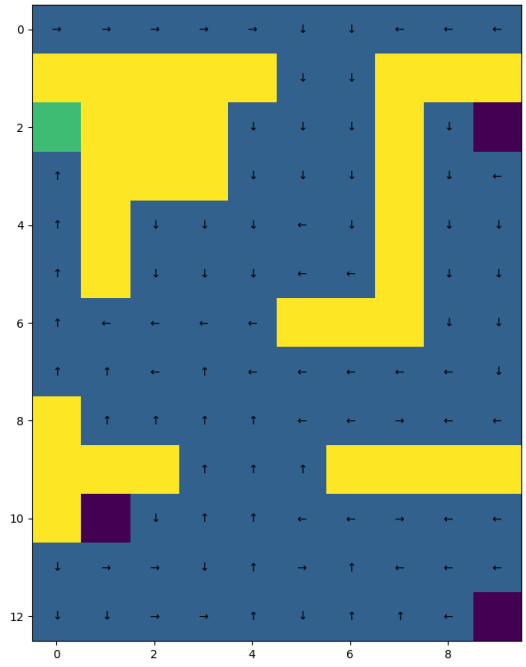

Figure 4: Optimal Policy

3.

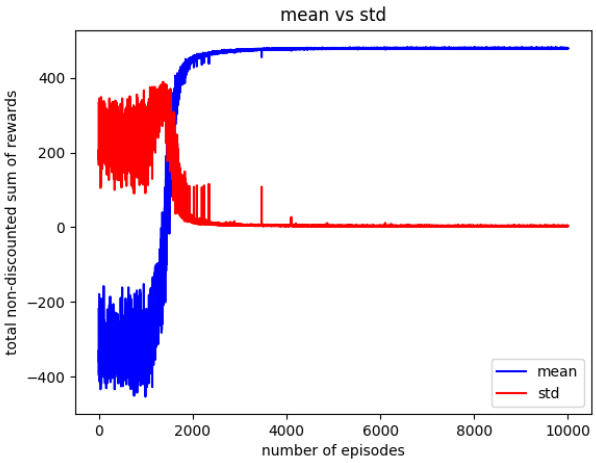

Figure 5: Mean and standard deviation of the total non-discounted sum of rewards at each episode

Question 3

1.

I have chosen to use the Q-learning algorithm to solve this problem. I preferred the Q-learning over the SARSA algorithm because Q-learning is an off-policy and can benefit from both exploitation and exploration. It is ϵ-greedy in finding next actions while in episode (updating behaviour policy), but at the same time being greedy in implicitly improving the policy, by updating the optimal Q value using the next state's optimal Q value(updating target policy). SARSA on the other hand, is an on-policy, where it uses the same (ϵ-greedy) policy to both generate the episodes and improve the policy. This prevents it from exploiting the environment and thus having a slower convergence rate but also less optimal results than Q-learning. SARSA is better suited to a "dangerous" environment where one wrong action can dramatically decrease the return. Given that our environment is not highly dangerous, Q-learning, which tends to yield better optimal results, is the preferred choice. This is a kind of cheating, as the environment is "unknown", so I shouldn't be able to tell if it's dangerous or not. I will set it as an assumption. We also assume that the environment can be modelled as a MPD, and is stationary.

Parameters
Exploration parameter: ϵ
During the first 3750 episodes, the ϵ gradually decreases from 1 to 0.3. This decrease happens in steps based on some conditions on the episode number set in my algorithm. After, the 3751^{th}, ϵ is multiplied by 0.99 after each episode, only if the total reward from the corresponding episode is positive. The idea behind these choices is similar to the one in my MC algorithm. The agent tries to utilize exploration as much as it can in the initial episodes, and then focuses in optimizing the value function and policy (by a decaying ϵ), becoming greedier only when it performs well.

Learning rate: $\alpha = 0.05$
The α parameter has been set to 0.05. I chose this for similar reasons stated in the above question. Again, the value 0.05 was chosen because it consistently outperformed various constant alpha values in terms of average performance.
Number of episodes:
The number of episodes has been set to 10,000 so that the agent has the chance to do some excessive exploration at the start without affecting the convergence of the algorithm to the optimal value function and policy. We assume that this number is sufficient for convergence.

2.

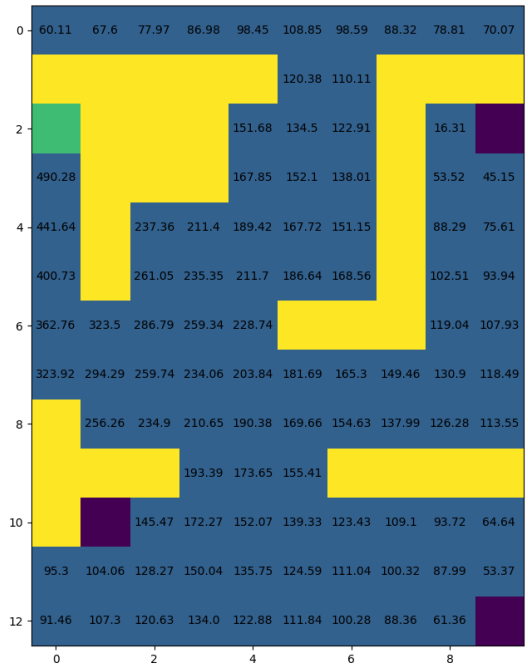

Figure 6: Optimal Value Function

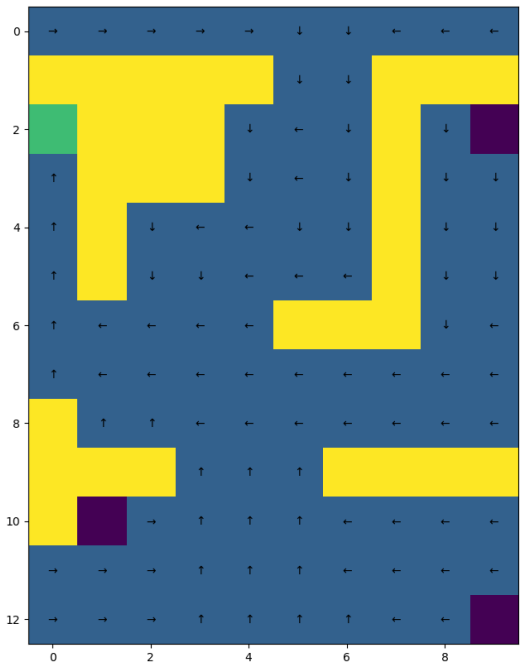

Figure 7: Optimal Policy

3.

Impact of α, with $\epsilon = 0.1$
In figure 8, we investigate how varying the learning rate, α, impacts my learning curve, I use a plot showing the Interim and the Asymptotic performance of my Q-learning algorithm with an ϵ-greedy behaviour policy of ϵ=0.1. The motivation comes from the Figure 6.3 on page 133 of the "Introduction to Reinforcement Learning" book. The asymptotic performance here is calculated by taking the mean of the first 10,000 episodes, whereas the interim performance is calculated by taking the mean of the first 100 episodes. These data are averages of 500 runs and 5 runs for the interim and asymptotic cases respectively. It is evident that as α increases the interim performance of the algorithm increases, maximizing at around α=0.6, and for values larger than that, there is a slight decline. On the contrary, we can see that the asymptotic performance increases with lower α value. This shows that lower α values converge slower but more accurately to the optimal values in the long-run. This makes sense in theory, as higher α values

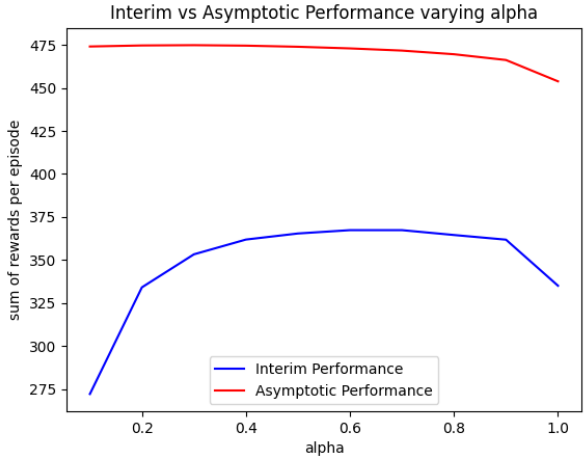

Figure 8: Mean and standard deviation of the total non-discounted sum of rewards at each episode

give more weight to recent episodes which provide more accurate information about the optimal values in the short-run, however, due to their tendency to forget older episodes, they might be losing some information that could contribute to a more accurate estimation of optimal values in the long-run.

Impact of ϵ, with $\alpha = 0.05$

With a fixed α value of 0.05, Figure 9 illustrates that all values of ϵ result in the maximum total reward at approximately the same stage of training, but lower ϵ values tend to yield slightly higher maximum rewards. Moreover, it is evident that larger values of ϵ lead to more frequent and larger variations in the mean reward. This suggests that if we chose a constant exploration rate, a lower ϵ value would lead to more stable convergence. These observations align with the principles of Q-learning, where higher values of ϵ increase the likelihood of taking suboptimal actions and receiving lower rewards. The only way to consistently select the best actions with minimal variance is to set ϵ to 0. Therefore, an ϵ-greedy policy with ϵ gradually converging to 0 can address the issues observed in the graphs. Additionally, the ability of high ϵ values to achieve high rewards is sensible, because Q-learning updates Q-values using a greedy policy, ensuring that the value function remains reasonably close to the optimal.

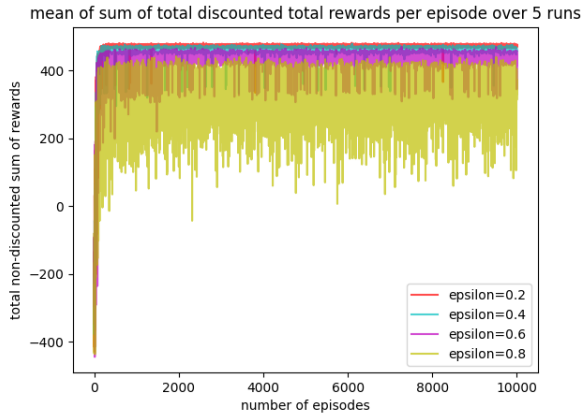

Figure 9: Mean of the total non-discounted sum of rewards at each episode