Imperial College
London

Reinforcement Learning

Imperial College London

Department of Computing

Reinforcement Learning Coursework 1

Author:
Megan Howard (CID: 01519881)

Date: November 2, 2023

1 Dynamic Programming Agent

1.1 Policy Iteration Approach

For this task I used Policy Iteration. I chose this approach because Policy Iteration can converge in fewer iterations than Value Iteration through actively improving the policy at each step. This was done in the 3 substeps of policy evaluation, policy improvement and policy iteration. For each state, the action that looks best according to the current value function estimate is selected (a greedy approach). In the policy iteration step, once the difference in the current and next step values being returned by my policy were over my delta threshold of 0.0001, the policy will no longer change. The improved policy has converged and the optimal policy is found. The small threshold of 0.0001 was chosen to ensure the optimal accuracy in the final policies and values found.

1.2 Policy and Value Functions

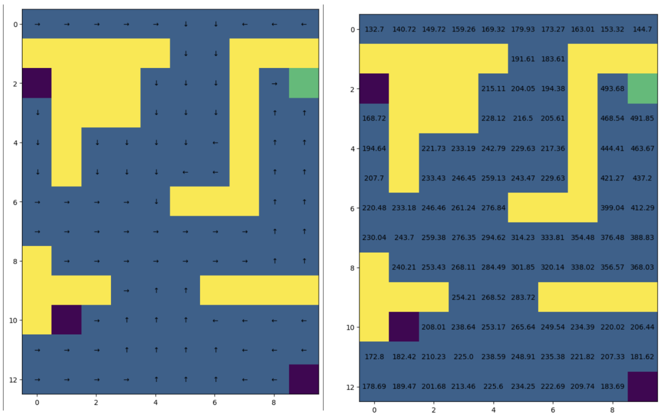


Figure 1: Optimal Policy

Figure 2: Optimal Value Function

Figure 3: Dynamic Programming Agent Policy and Value Function

In the 2 images attached above we can see that the DP agent has established a policy which moves all state squares towards those with the highest value functions, and thus in the direction of the absorbing state associated with the highest reward.

1.3 γ and P Variations

1.3.1 P variations

In the below figures, we can see how both the policy and the value functions have varied as the probability of success parameter (p) is increased from 0.1 up through 0.25, to 0.5. When the p-value is at 0.1, the policy actually moves away from the positive absorbing state - perhaps since it's harder to reach - and towards the more

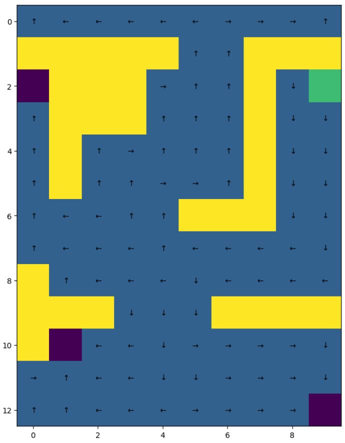


Figure 4: Policy where γ = 0.96 and p=0.1

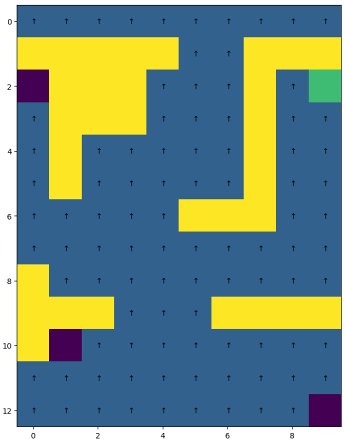


Figure 5: Policy where γ = 0.96 and p=0.25

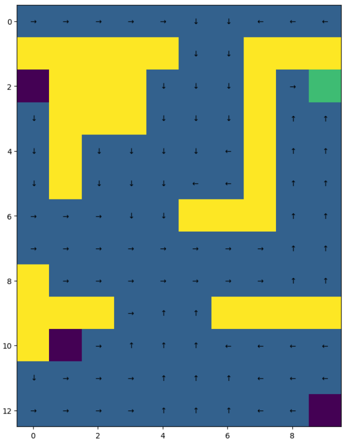


Figure 6: Policy where γ = 0.96 and p=0.5

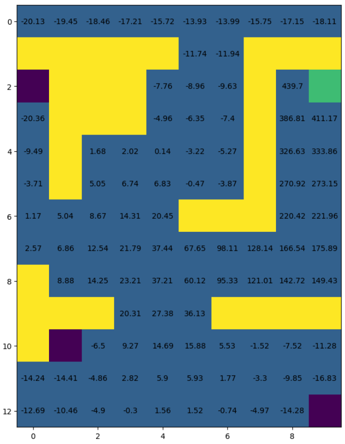


Figure 7: Value function where γ = 0.96 and p=0.1

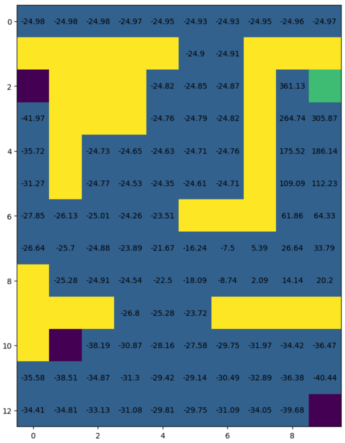


Figure 8: Value function where γ = 0.96 and p=0.25

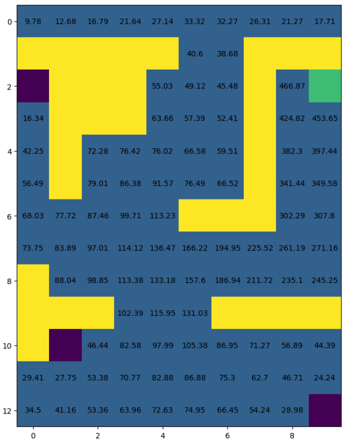


Figure 9: Value function where γ = 0.96 and p=0.5

accessible negative absorbing states. When the p-value is equal to 0.25, the probability of success increases, and the agent can be slightly more confident in its actions. Here we can see that all of the state's policies are pointing upwards. This may indicate that moving upwards is perceived as the least risky or most rewarding action given the current maze configuration. When the p-value is over 0.25 - a greater probability of success - the agent's actions become more reliable. We can see that our agent's actions are executed as anticipated and thus our policy converges to the optimal approach around the maze.

1.3.2 γ Variations

In the below figures, we can see how both the policy and the value functions have varied as the discount factor (γ) is increased from 0.25 up through 0.5, to 0.75. When the γ is at 0.25, the further rewards are factored less into the final policy

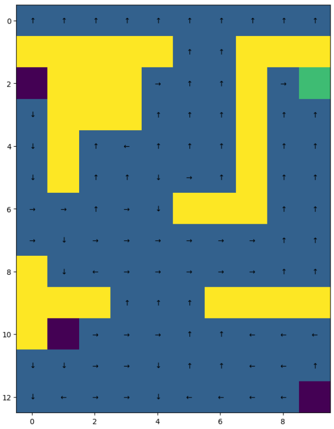


Figure 10: Policy where γ = 0.25 and p=82

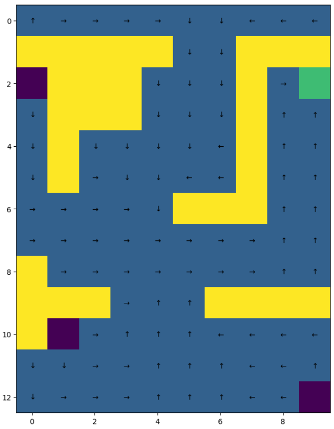


Figure 11: Policy where γ = 0.5 and p=0.82

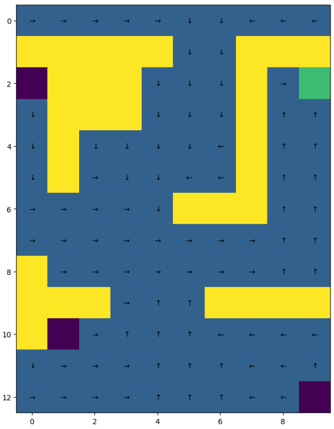


Figure 12: Policy where γ = 0.75 and p=0.82

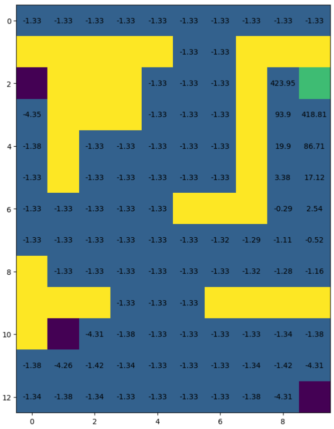


Figure 13: Value Function where γ = 0.25 and p=82

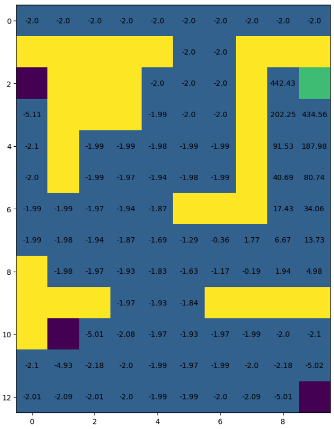


Figure 14: Value Function where γ = 0.5 and p=0.82

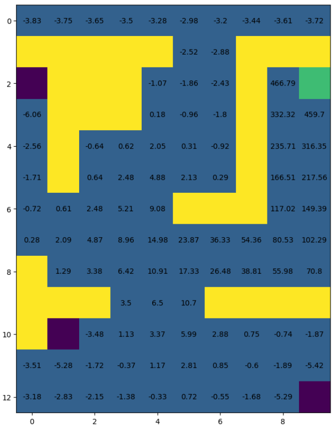


Figure 15: Value Function where γ = 0.75 and p=0.82

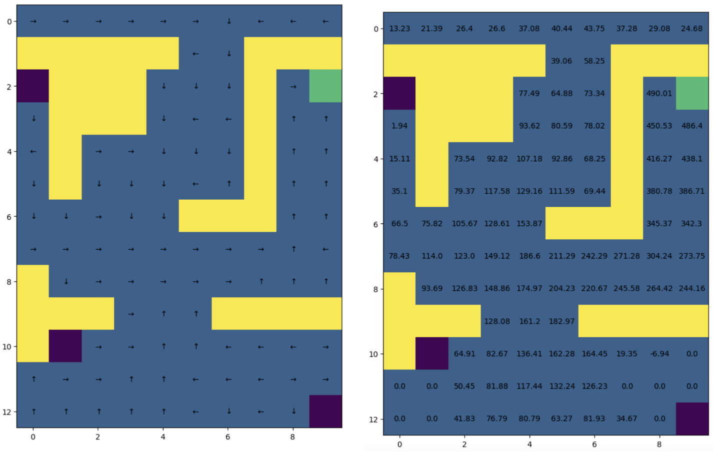
recommended than immediate rewards. As such, states further from the positive absorbing state have their policies more influenced by nearby rewards or penalties, rather than the long-term prospect of reaching the positive absorbing state. Given this, the policy figure is honestly still surprisingly decent. Though the policy points away from the positive absorbing state and towards the edge of the maze at the furthest states from the rewarding absorbing state, closer to this rewarding absorbing state the policy is correctly driving the agent towards this positive end state. When the γ is equal to 0.5, we can see that all of the state's action policies are progressively moving towards the optimal policy, especially for states that aren't too distant from the positive absorbing state. Only the most distant states are not being optimised, as the next state's values are increasingly considered. When the γ is 0.75, the agent strongly values future rewards. As a result, the policy converges to an optimal strategy where actions consistently lead toward the positive absorbing state, even from distant states. The state values also reflect this, with higher values for states leading to the positive reward due to the compounded effect of the discount factor over multiple steps.

2 Monte Carlo Agent

2.1 Every Visit Approach

I adopted an 'every-visit', epsilon-greedy, on-policy Monte Carlo algorithm where the agent follows an epsilon-greedy policy during its interactions with the environment. Specifically, with a probability of ϵ, the agent explores a random action, and otherwise exploits the best-known action derived from Q-values. This approach was chosen for its direct learning capability, consequent more intuitive design /implementation, and stability. The exploration rate (ϵ) was set to 0.4 after experimentation indicated it achieved the best convergence rate. While convergence was typically observed around the 150th episode, I chose 1000 episodes to run my agent, as at numbers lower than this, there were significant portions of the maze that were underexplored - i.e. didn't show optimal policy.

2.2 Policy and Value Function



0	→	→	→	→	→	→	↓	←	←	←
						←	↓			
2					↓	↓	↓		→	
	↓				↓	←	←		↑	↑
4	←		→	→	↓	↓	↓		↑	↑
	↓		↓	↓	↓	←	↑		↑	↑
6	↓	↓	→	↓	↓				↑	↑
	→	→	→	→	→	→	→	→	↑	←
8		↓	→	→	→	→	→	↑	↑	↑
				→	↑	↑				
10			→	→	↑	↑	←	←	←	→
	↑	→	→	↑	↑	←	←	←	→	→
12	↑	↑	↑	↑	↑	←	↓	←	↓	
	0		2		4		6		8	

Figure 16: Optimal Policy

0	13.23	21.39	26.4	26.6	37.08	40.44	43.75	37.28	29.08	24.68
						39.06	58.25			
2					77.49	64.88	73.34		490.01	
	1.94				93.62	80.59	78.02		450.53	486.4
4	15.11		73.54	92.82	107.18	92.86	68.25		416.27	438.1
	35.1		79.37	117.58	129.16	111.59	69.44		380.78	386.71
6	66.5	75.82	105.67	128.61	153.87				345.37	342.3
	78.43	114.0	123.0	149.12	186.6	211.29	242.29	271.28	304.24	273.75
8		93.69	126.83	148.86	174.97	204.23	220.67	245.58	264.42	244.16
				128.08	161.2	182.97				
10			64.91	82.67	136.41	162.28	164.45	19.35	-6.94	0.0
	0.0	0.0	50.45	81.88	117.44	132.24	126.23	0.0	0.0	0.0
12	0.0	0.0	41.83	76.79	80.79	63.27	81.93	34.67	0.0	
	0		2		4		6		8	

Figure 17: Optimal Value Function

Figure 18: Monte Carlo Agent Policy and Value Function

In the 2 images attached above we can see the MC agent establishes a policy which moves nearly all state squares towards those with the most immediate highest value functions, and thus in the overall direction of the absorbing state associated with the highest reward. However, some of the furthest squares don't seem to have been explored enough to be considered, and this remained true even as the exploration rate was increased quite heavily.

2.3 MC Learning Curve

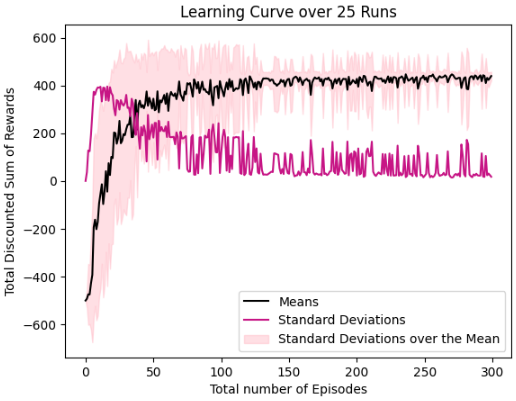


Figure 19: MC Learning Curve

The learning curve depicts the agent rapidly learning to optimise rewards within the initial 50 episodes and stabilizing around the 150th episode. A broader standard deviation in this learning phase is seen due to the 0.4 exploration rate. This ϵ rate encourages exploration and allows the agent to not only exploit known routes but also discover potentially better paths. As the algorithm approaches convergence, the standard deviation unsurprisingly narrows, reflecting a more consistent policy application and reduced exploration.

3 Temporal Difference Agent

3.1 SARSA Approach

For this question, my agent uses a SARSA approach, with a 'Greedy in the Limit with Infinite Exploration' (GLIE) Policy such that exploration decreases over time, but encourages all state-action pairs to be visited initially. With a probability of ϵ, my agent selects a random action (exploration), and with probability 1 - ϵ, it selects the best-known action (exploitation) based on the current Q-value. For this, I chose an ϵ value of 0.5 to balance the agent's exploration of unknown areas and iteratively decreased this ϵ value by 0.1% for each action taken in an episode to encourage eventual exploitation only of known reward areas. I chose an α value of 0.5 as this learning rate allows the agent to learn efficiently without being too cautious or rash. Again I chose 1000 episodes to run my agent, to ensure all portions of the maze were adequately explored and the overall policy was correctly optimised.

3.2 Policy and Value Function

While in some corners the policy structure is still a little less understood, resulting in slightly more confused policy at times, the overall policy is optimised moving the agent in the direction of the absorbing state associated with the highest reward. The same is true for the value functions of each state.

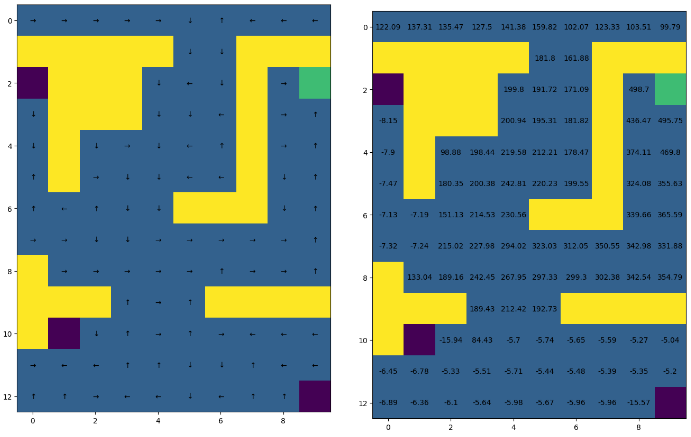


Figure 20: Optimal Policy

Figure 21: Optimal Value Function

Figure 22: Temporal Difference Agent Policy and Value Function

3.3 Varying α and ϵ

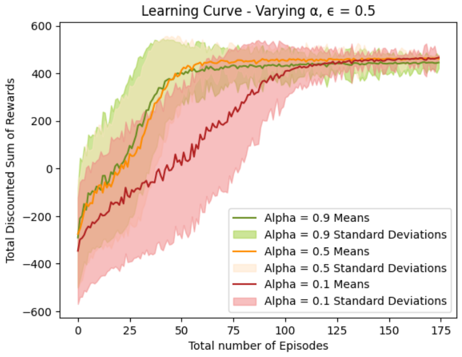


Figure 23: Learning Curve where α is varied, ϵ = 0.5

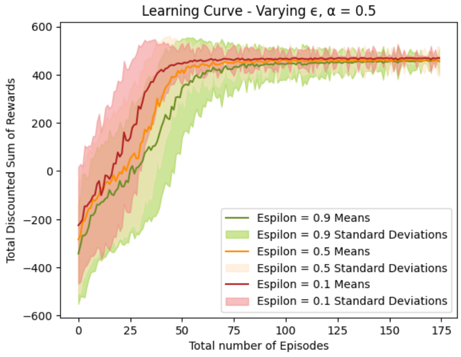


Figure 24: Learning Curve where α =0.5, ϵ is varied

As α is varied, we can see lower α values make the agent averse to changes; relying more on its prior knowledge in this way causes the learning to converge only after a lot more episodes. While a high α prioritising recent experiences in this scenario favours the learning converging to the optimal values far quicker, there are 2 disadvantages with such an approach. One is that the route can become narrow-sighted - i.e. with only the optimal path being followed. Additionally, the standard deviation also shows a bit more noise after this convergence due to the inherent instability of such a high learning rate. Varying the ϵ exploration rate in this scenario had a surprisingly minimal impact on the learning curve of my agent. I would've expected the lower epsilon to lead to premature convergence (potentially at a suboptimal policy if exploration hasn't been sufficient), which can be marginally seen in Figure 24. Additionally a higher ϵ leads to longer convergence times, but in this example it can be seen to most heavily contribute to a larger range of standard deviations as the agent keeps exploring even when it has learned a near-optimal policy.