Reinforcement Learning Coursework 1

Alexander Mikheev

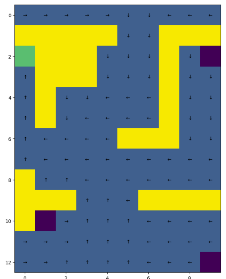
November 2022

1 Question 1 (Dynamic Programming)

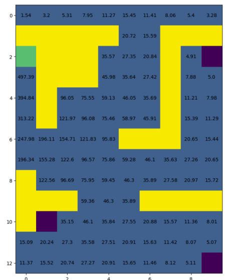
1.1 Method, parameters, assumptions

The chosen Dynamic Programming method is Value Iteration over Policy Iteration. Both methods were implemented however Value Iteration converged considerably faster and more efficiently; Value Iteration took 18 epochs to converge as opposed to Policy Iteration's 191 epochs. This is a consequence of the fact that Value Iteration combines the evaluation and improvement of the policy in one step, and runs faster because it does not need to wait for convergence of the policy evaluation algorithm. The assumption of the implementation is complete model knowledge (with access to all transition probabilities, state rewards, and absorbing states). The implementation of Value Iteration only takes only a threshold hyperparameter; when the value difference from two improvement iterations is less than the threshold parameter for all states, the algorithm terminates. This threshold is set to 0.0001.

1.2 Optimal Policy + Optimal Value Function



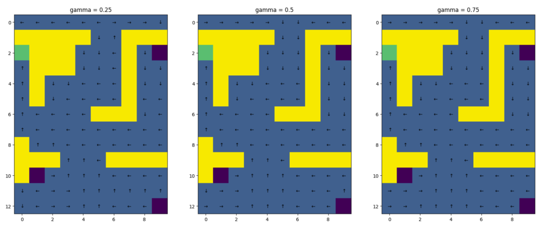
(a) DP Optimal Policy



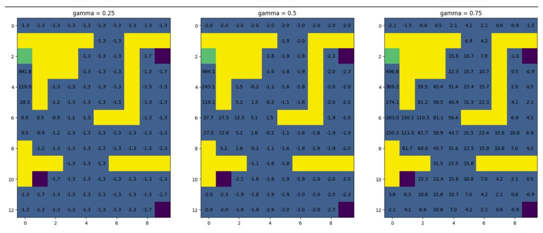
(b) DP Optimal Values

Figure 1

1.3 Varying γ and p

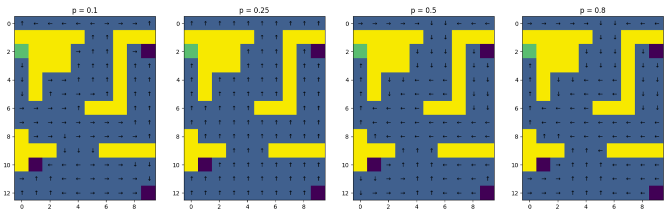


(a) Policies for varying γ

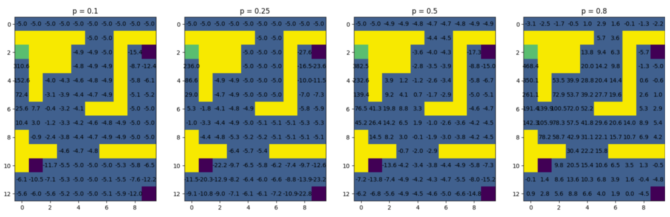


(b) Values for varying γ

Figure 2



(a) Polices for varying p



(b) Values for varying p

Figure 3

A γ value less than 0.5 discounts the future rewards to an extremely small value to the point where the future states have little-to-no effect on the value of the current state. At $\gamma = 0.25$ almost all state values are -1.3 with the exception of states that are within a few steps of an absorbing state. Increasing γ to 0.5 increases the look-ahead, and thus we can see varying state values with respect to the distance to an absorbing state. However this value of γ is still too small to vary the state values in the top row (the starting states) which are further away from the absorbing states, thus is still too small. Increasing γ above 0.5 rectifies this issue because there is an adequate look-ahead; all states in the grid have varying values with respect to the distance to the nearest absorbing state. With a probability = 0.25 every action is equally probable thus every possible policy has the same value (because no no policy can change the dynamics of the agent). With a probability > 0.25 the agent is now more likely to step into the desired direction, thus we can now measure the effect of taking different actions, thus the expected optimal policy is learned. With a probability < 0.25, there is a higher probability of not taking the desired action, thus the policy learns to pick the worst action because with this probability of success, we are most likely to avoid the worst action.

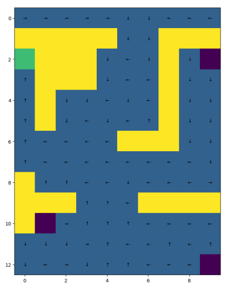
2 Question 2 (Monte Carlo)

2.1 Method, parameters, assumptions

The chosen Monte-Carlo (MC) method is first-visit online MC. Online MC was chosen over Batch MC; both were implemented however online MC converged faster, which makes sense because Q (the state-action value function) is updated incrementally, and we avoid having to wait for an entire batch of episodes to complete before updating Q.

There are 3 parameters to the MC learning algorithm; learning rate α, the soft policy probability ϵ, and the number of episodes, n. α dictates the rate of Q updates; higher α values generally converged faster but with more noise (see Figure 7). After tuning this hyperparameter, I found the optimal trade-off with a decaying α that starts high at 0.2, and reduces by a small decaying factor of 0.99 per episode (with a minimum of 0.05 after which there is no further decay). Thus, there are rapid improvements to the policy in the starting episodes, and reduced noise in the later episodes, and therefore the optimal noise/convergence speed trade-off. ϵ dictates the amount of exploration the agent takes in the environment; low ϵ values converge quickly but rarely reached the non-reward absorbing states, leading to poor values and policies around the non-reward absorbing states. A higher epsilon value yielded more accurate values, due to the increased exploration but at the expense of a higher variance, and less stable returns. The optimal trade-off was using a decaying ϵ that started relatively high at 0.7, and then gradually decreased as the number of episodes increased (as the certainty of the optimal policy increases). Although using a fixed ϵ converged faster than decaying, using decaying ϵ gave considerably better values with respect to the optimal Dynamic Programming values. The number of episodes n, was set to 5000. The justification is simply to observe the learning curve figure 6; at 5000 episodes there is a smooth convergence giving very stable results. With a fixed ϵ, 1000 episodes was sufficient for convergence, but at the cost of considerably less accurate value estimates. The assumption of this MC learning algorithm is that with an ϵ-greedy soft policy, every reachable state has an adequate number of visits, and thus can be accurately estimated.

2.2 Optimal Policy + Values


(a) MC Optimal Policy

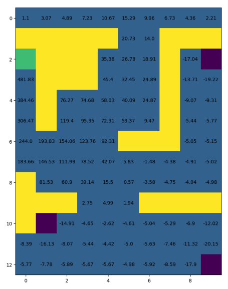

(b) MC Optimal Values

Figure 4

2.3 MC Variability + Determining Replications

There are two sources of variability with the MC learning algorithm; there is some randomness in the success of taking a desired action, and there is randomness in the generation of episodes because we are using a soft-policy (for exploration), thus different MC-learning runs will lead to different results. To determine a sufficient number of replications the L2 error is computed between the mean values per episode of the current repetition and the average of previous MC repetitions. Figure 5 illustrates this method; we can clearly see that by 30 repetitions the error has converged smoothly, thus 30 repetitions have been selected.

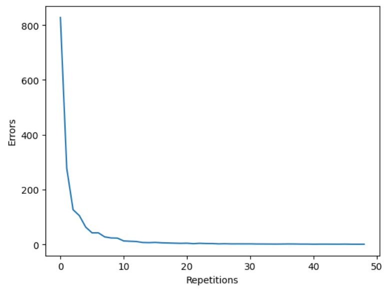


Figure 5: L2 error between current replication and the average of previous replications

2.4 Learning Curve of the Agent

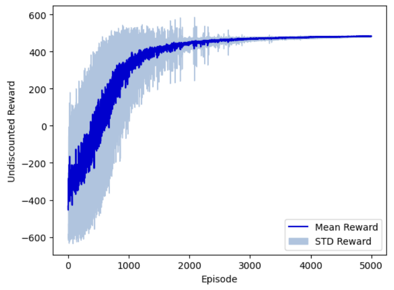
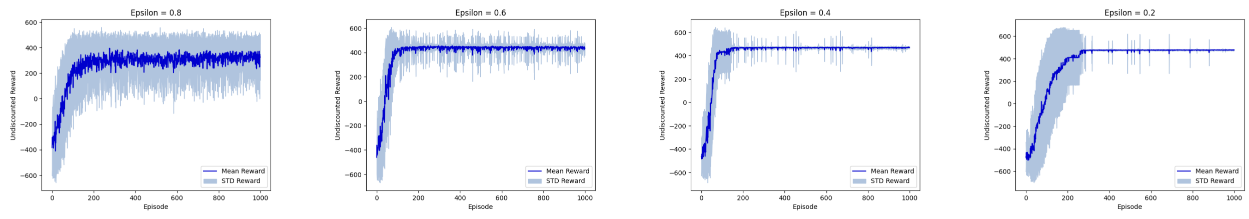


Figure 6: MC Learning Curve across 30 repetitions

2.5 Varying α and ϵ

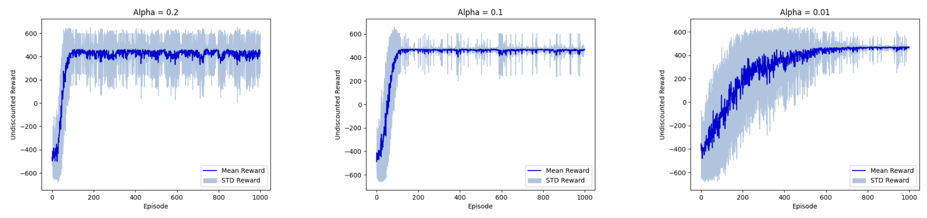
When ϵ is increased this means that the agent explores the environment more, and therefore we can see a general trend in figure 7 of higher ϵ values resulting in a higher variance, but slower convergence. This is the outcome we would intuitively expect because more exploration leads to a wider range of returns across the episodes. When epsilon is set to 0.2 there is a high amount of variance before it converges, but after it converges this ϵ value gives the least variance.



(a) (b) (c) (d)

Figure 7: MC error plots with varying ϵ

When α is increased, Figure 8 depicts a general trend of faster convergence and higher variance. Intuitively this outcome makes sense because a higher α leads to more extreme updates of the state-action value function, thus leads to taking bigger strides that converge faster but may also overshoot. When α is set to 0.01 we see a higher variance before convergence, simply because the learning rate is so small that the policy improves too slowly, resulting in varied rewards before it eventually converges. All learning rates eventually converge to similar values.



(a) (b) (c)

Figure 8: MC error plots with varying α

3 Question 3 (Temporal Difference)

3.1 Methods, parameters, assumptions

The chosen Temporal Difference (TD) learning method is Q-learning. An alternative is to use the SARSA algorithm; both were implemented however Q-learning outperformed SARSA both in terms of speed, and in terms of values (significantly closer to DP values). The advantage of Q-learning is that it directly learns the current optimal policy whilst using an ϵ-greedy policy to keep exploring, whereas SARSA learns the near-optimal ϵ-greedy policy.

There are 3 parameters to the TD learning algorithm; learning rate α, the soft policy probability ϵ, and the number of episodes, n. Figure 12 shows that a variety of α values converge to similar values, however increasing the α parameter significantly increases the speed of convergence, and thus is set to 0.5. In my environment configuration, the probability of a successful action is 0.98, therefore I need a high ϵ value to compensate for the lack of exploration. ϵ values less than 0.8 achieved poor value estimates around the negative-reward absorbing states, simply from a lack of exploration. The learning algorithm achieved the best estimates at an ϵ set to 0.8, this was a sufficient quantity for exploration giving very accurate estimates (with respect to DP). The number of episodes parameter n is set to 1000. Although the learning curve (Figure 10) shows that the

TD learning curve converges extremely quickly at around 200 episodes, smaller values of n yielded less stability in the estimated values; setting n to 1000 gave more consistent and stable values. The assumption of this TD learning algorithm is that with an ϵ-greedy soft policy, every reachable state has an adequate number of visits, to be accurately estimated.

3.2 Optimal Policy, Optimal Values

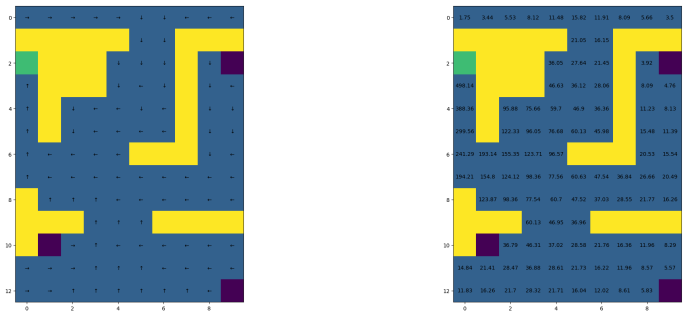

(a) (b)

Figure 9: TD Optimal Policy + Optimal Values

3.3 Learning Curve of the Agent

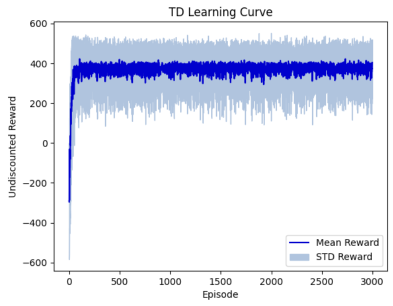
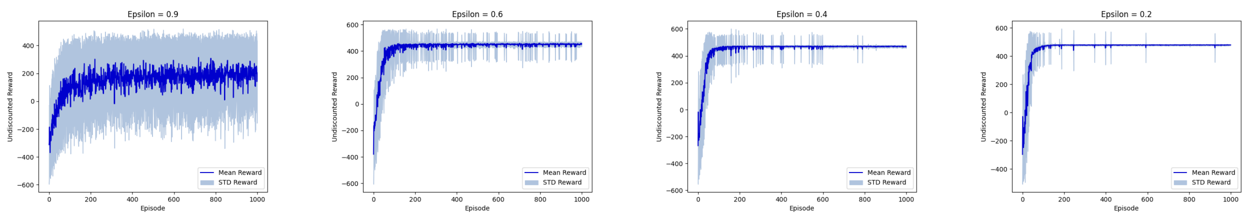


Figure 10: Learning Curve of TD agent across 30 repetitions

3.4 Varying α and ϵ

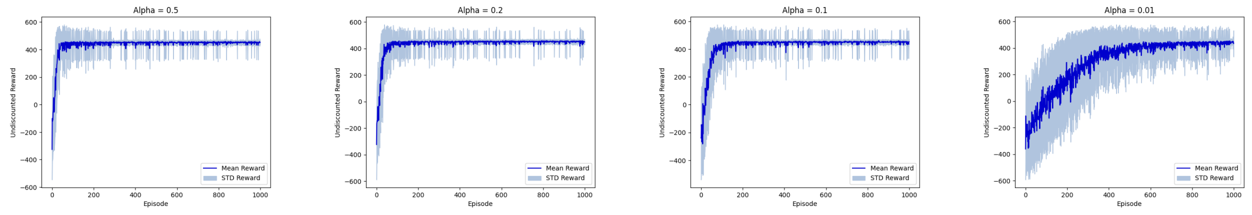
Figure 11 shows that increasing ϵ increases the variance. This result makes sense from a theoretical standpoint, because more exploration will lead to more varied returns, and thus a higher variance.



(a) (b) (c) (d)

Figure 11: TD error plots with varying ϵ

Figure 12 shows that increasing α values leads to faster convergence, but generally all α values converge to the same outcome. We see significantly more noise when α is set to 0.01, simply because this value is too small and the Q function is updated too slowly, leading to more varied results and significantly slower convergence. Similarly, this is consistent from a theoretical standpoint, a higher learning rate 'learns' faster, leading to faster convergence.



(a) (b) (c) (d)

Figure 12: TD error plots with varying α

4 Question 4 (Comparison of Learners)

4.1 Value Function Estimation Error

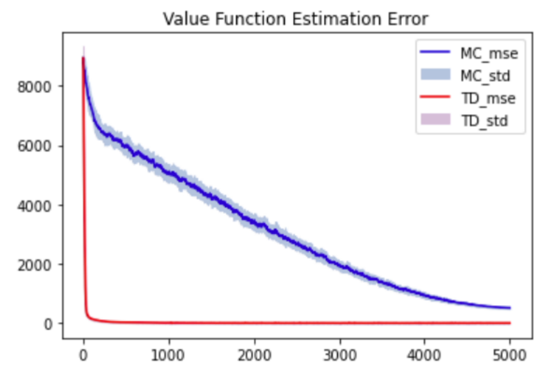


Figure 13: Value Function Estimation Error and Standard Deviation of TD and MC learners

4.2 Analysis of Value Function Estimation Error

Figure 13 clearly show that TD converges extremely rapidly to very optimal values, whereas MC converges considerably slower, to less-optimal values. The speed of convergence is the expected outcome; TD tends to converge faster than MC because TD uses bootstrapping to update the values at each step it explores, whereas MC only updates the values after computing the entire episode. Figure 13 also shows that TD converges to more accurate value estimates (smaller MSE errors) which makes sense from a theoretical standpoint because TD exploits the Markov property by bootstrapping, therefore tends to perform better in Markov environments. For TD, I have set the number of episodes n to 5000, in order to plot TD and MC on the same figure (usually TD is set to 1000 episodes).

4.3 Scatter plot

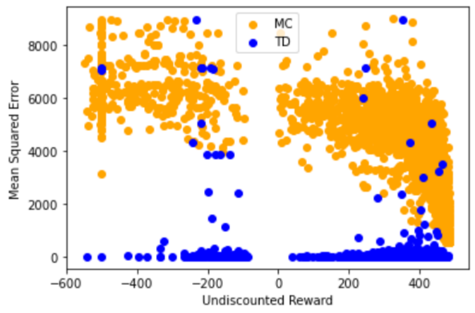


Figure 14: Scatter plot of Mean Squared Error against Undiscounted Reward for a single run

4.4 Analysis of scatter plot