1 Grid world

1.1 Reward state, p and discount factor

For $x = 7$, $y = 7$ and $z = 5$, the reward state is as shown in Equation 1. The probability of success p and the discount factor γ are also shown in Equation 2 and Equation 3 respectively.

$$j = ((z + 1) \bmod 3) + 1 = 1, \quad \text{state } s_1 \tag{1}$$

$$p = 0.25 + 0.5\left(\frac{x+1}{10}\right) = 0.65 \tag{2}$$

$$\gamma = 0.2 + 0.5\left(\frac{y}{10}\right) = 0.55 \tag{3}$$

2 Dynamic Programming

2.1 Optimal value function and policy

A policy iteration algorithm was used to compute the optimal value function which alternates between policy evaluation and policy improvement. In the policy evaluation function, the Bellman equation was used to find the value at each state, and a low threshold of 0.0001 was set to allow the solution to converge close to the true value function.

In the policy iteration function, the initial policy was arbitrarily set to $[0.25, 0.25, 0.25, 0.25]$ for all states, where the elements corresponds to the movements $[\text{North}, \text{East}, \text{South}, \text{West}]$. A greedy policy was also implemented which chooses actions that lead to the highest Q value for each state.

The optimal value function is reported for each state in Figure 2.1. The optimal policy function is described for each state in Figure 2.2. An arrow pointing up refers to the movement North. The yellow box and purple box represents the two terminal states with rewards of +10 and -100 respectively.

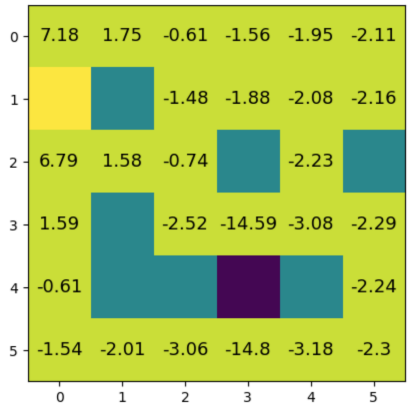

Figure 2.1: Optimal value function

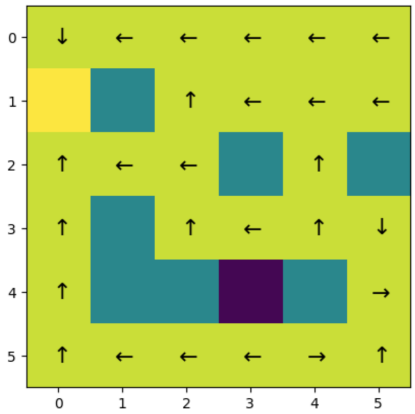

Figure 2.2: Optimal policy function

2.2 Effects of γ and p on the optimal value and policy functions

When $p < 0.25$, it is more likely to move in an unintended direction. What this implies is that although the agent knows which direction to move to in order to earn the best rewards, it is

more likely to take another direction instead. Thus, for low values of $p < 0.25$, the optimal policy steers the agent into the terminal state with -100 reward, and funnily enough, a higher $\gamma > 0.5$ value leads to a lower optimal value function. Overall, a low $p < 0.25$ value leads to a lower optimal value function, and gets lower with increasing γ.

When $p > 0.25$, it is more likely to move in the intended direction and thus, the optimal policy will steer the agent into the terminal state with +10 reward. When the value of γ is low, the agent is near-sighted and is less interested in maximising the long term rewards. This can lead to sub-optimal behaviours as seen from the bottom right corner of Figure 2.2 where the agent in state s_{24} would prefer to hit a wall, as opposed to moving to its neighbours which have lower value functions. However, as γ increases, the agent becomes more far-sighted and more interested in maximising the long term rewards which can resolve this problem. Overall, for $p > 0.25$, an increasing γ causes the value function of states near the -100 reward state to go down, and values of states near the +10 reward state to go up.

3 Monte Carlo and Temporal Difference RL

In the following exercises, $p = 0.65$ and $\gamma = 0.55$ values are used. It should be noted that since the optimal action is taken 65% of the time, even if the optimal policy was known, the agent might still end up in the -100 reward state, or take other sub-optimal actions.

3.0.1 MC optimal value function and policy

In general, the overall schema of generalised policy iteration was used. An every-visit, exploring starts batch MC algorithm was used to generate the optimal value function and policy. Each batch contains 50 episodes and between each batch, an on-policy ϵ-greedy GLIE algorithm was used to improve the policy where $\epsilon = 0.6$, and ϵ reduces to zero with $\epsilon_k = \frac{1}{k}$. The learning rate was set to $\alpha = 0.1$ so that the recent experience does not severely outweigh the past experiences. Finally, the algorithm was run 30 times for the Central Limit Theorem (CLT) to hold which means that the sample means and standard deviations are close to the population means and standard deviations.

The optimal value function and policy obtained using MC is shown in Figure 3.1. The learning curve is shown in Figure 3.2.

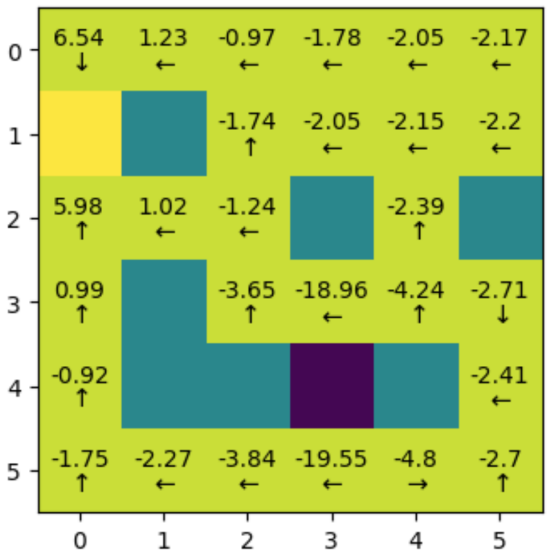

Figure 3.1: Optimal value function and policy for the Monte Carlo RL

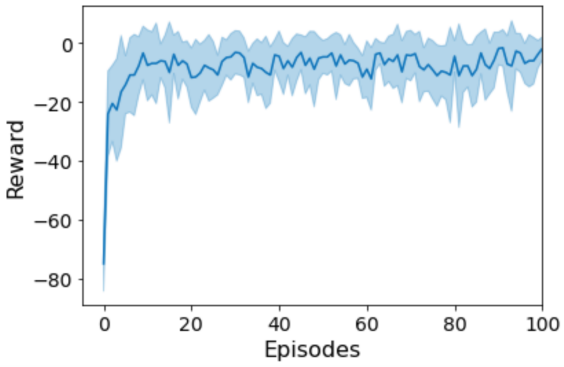

Figure 3.2: Learning curve for the Monte Carlo RL. The lightly shaded area represents the mean ± standard deviation. The algorithm was run 30 times for the CLT to hold

3.0.2 TD optimal value function and policy

For the TD value function estimation algorithm, a learning rate of $\alpha = 0.1$ was set. A SARSA on-policy ϵ-greedy GLIE TD control was also used, where $\epsilon = 0.6$, to update the Q function in order to explore states in the world, and allow it to converge to an optimal Q function. In addition, a Robbins-Monroe sequence of α_t was also set for convergence, where α is the reciprocals of powers of 2 of the steps. The optimal value function and policy obtained using TD is shown in Figure 3.3. The learning curve is shown in Figure 3.4.

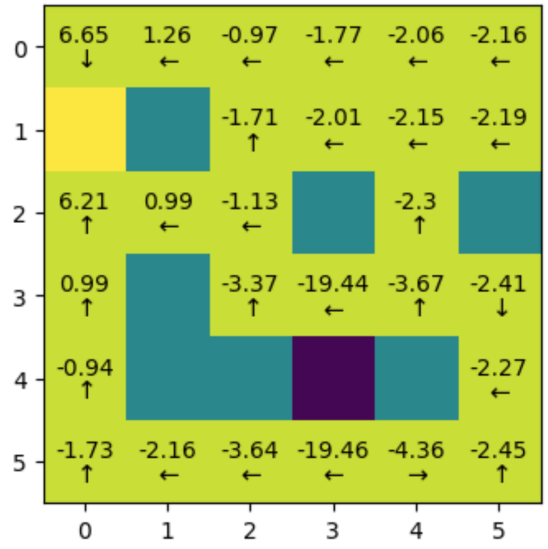

Figure 3.3: Optimal value function and policy for the Temporal Difference RL

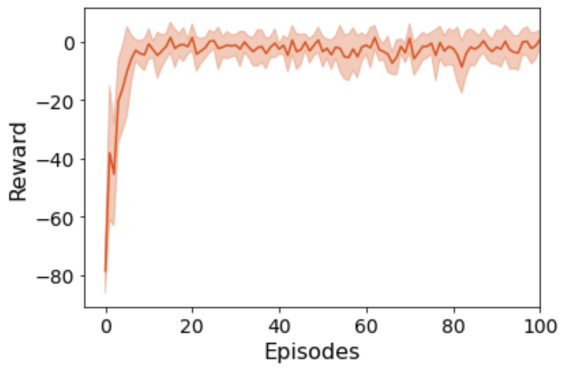

Figure 3.4: Learning curve for the Temporal Difference RL. The lightly shaded area represents the mean ± the standard deviation. The algorithm was run 30 times for the CLT to hold

3.0.3 Effects of ϵ and α

An increase in ϵ, and a decrease in α, both cause the MC and TD learning curves to have a longer rise time, and a longer time to convergence. If GLIE was not implemented, and a large

$\epsilon > 0.67$ value was chosen, the policy will not converge to the optimal policy, as the agent is equally or more likely to choose a sub-optimal action.

The exploration parameter ϵ gives an indicator of how much the agent is allowed to explore, by not choosing the greedy action all the time. Such exploration may not lead to the discovery of a better state-action pair and thus, it slows down the convergence. On the other hand, the learning rate α describes how much weight is given to recent experiences and thus, for a lower α value, updates in the Q function are slower, and the time taken for convergence increases.

4 Comparison of learners

4.0.1 Estimation error against episodes

The plot of the estimation error against episodes for both MC and TD is shown in Figure 4.1. comparison.

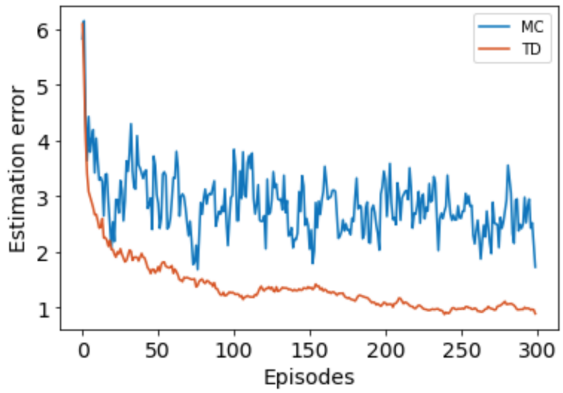

Figure 4.1: Plot of value function estimation error against episodes

In the estimation error against episodes plot, MC shows greater variance than TD and furthermore, the error for TD decreases faster than for MC. This relates back to the theory that since MC learns from complete episodes, it has a higher variance (lower precision). On the other hand, TD uses bootstrapping which is possible because of the optimal sub-problem structure. Due to this, not only is there lower variance, there is also more efficient use of data and thus, the error for TD decreases faster than for MC. However, since TD uses bootstrapping, it has a higher bias than MC and eventually, with long run times, MC will result in a more accurate value function than TD.

4.0.2 Estimation error against rewards

The plot of the estimation error against reward is shown in Figure 4.2. The axis for both plots are the same to allow for easier comparison.

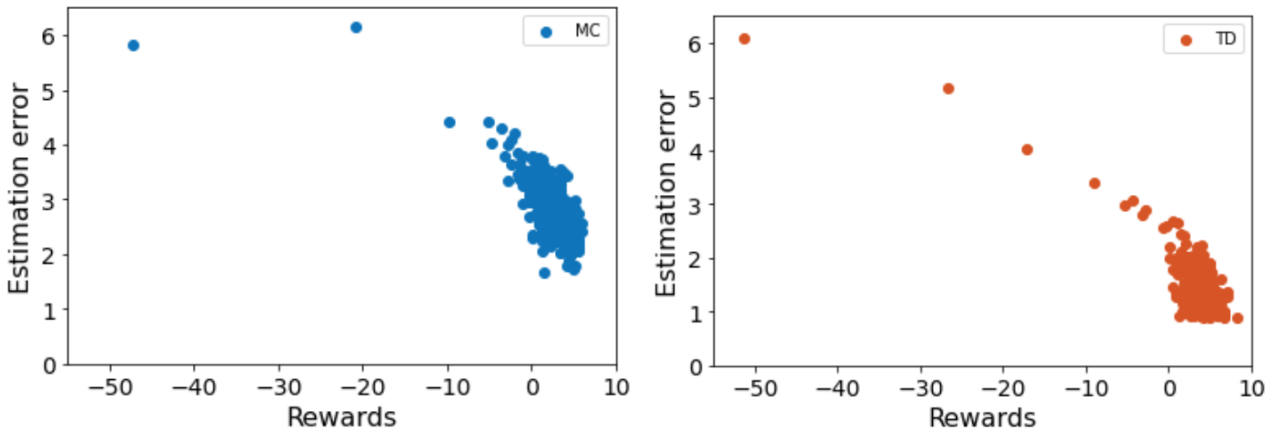

Figure 4.2: Plot of value function estimation error against rewards. Left: MC, Right: TD

Figure 4.2 shows the relationship between the estimation error and the rewards earned for an episode. In general, as episodes increase, not only does the agent learn the optimal policy which leads to higher rewards, the value function also gets updated and thus, the estimation error decreases. What this results in is a negative slope in the plot which reflects the improvement in rewards earned and value function estimate (decrease in error).

The plot shows that TD has a lower mean estimation error, which ultimately allows it to reach higher rewards as opposed to MC. This highlights the importance of an accurate value function estimate to obtain good rewards.

The scatter plot also helps visualise the variance in value function estimates for MC and TD. The plots show that MC has a larger variance than TD due to learning from complete traces. Furthermore, the plots also show that for the same number of episodes, TD decreases the estimation error faster due to bootstrapping. As more episodes are run, the estimation error of MC will decrease and its value function estimate will eventually become more accurate than TD.

4.0.3 Further comparisons

The effect of different p and γ values on the estimation error against rewards plot are shown below in Figure 4.3 and Figure 4.4 respectively.

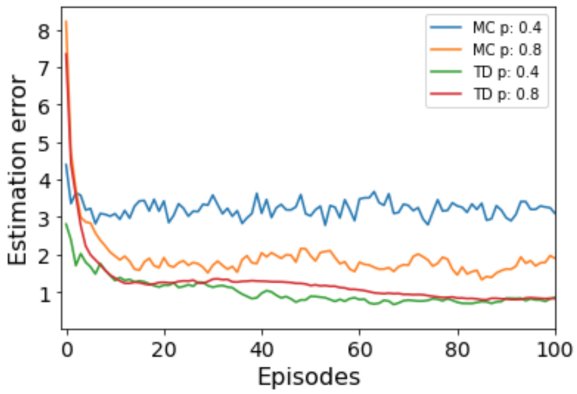

Figure 4.3: Effect of p on the value function estimation error of MC and TD. $\gamma = 0.55$, $\epsilon = 0.4$ and $\alpha = 0.05$

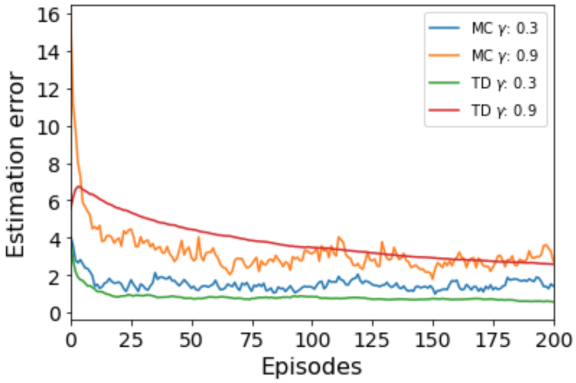

Figure 4.4: Effect of γ on the value function estimation error of MC and TD. $p = 0.7$, $\epsilon = 0.4$ and $\alpha = 0.05$

The plot for varying p shows that TD learns faster than MC for different p values and is able to decrease the value function estimation error at a faster rate. This implies that TD is "better" in terms of learning the policy quickly, however, TD is still subject to bias due to bootstrapping. As the number of episodes increases, MC will have a more accurate value function estimate than TD.

On the other hand, for large $\gamma = 0.9$ and $p = 0.7$ values, MC learns faster than TD as shown in Figure 4.4. This happens because when γ is high, the estimated value function, $\gamma V(s')$, in TD has a greater weightage. If this estimate is not accurate, it can lead to a bad update and cause a larger error in the value function estimate. Since MC relies on complete traces to calculate the value function, it is unaffected by this and remains unbiased.