1 Dynamming Programming

1.1 Method & Design

I use value iteration to solve the problem. As the maze is small, the computational cost for value iteration and policy iteration does not have much difference. Therefore, I chose value iteration because it is easier to implement. The only parameter $\gamma = 0.88$ (discount rate) is given by the instruction, thus no need for consideration. The assumption of DP is that the main problem can be divided into several sub-problems to solve, and unifying the sub-results gives the desired answer. This assumption is satisfied because the environment exhibits Markov property: reward only depends on the current state and the action taken at that state.

1.2 Output

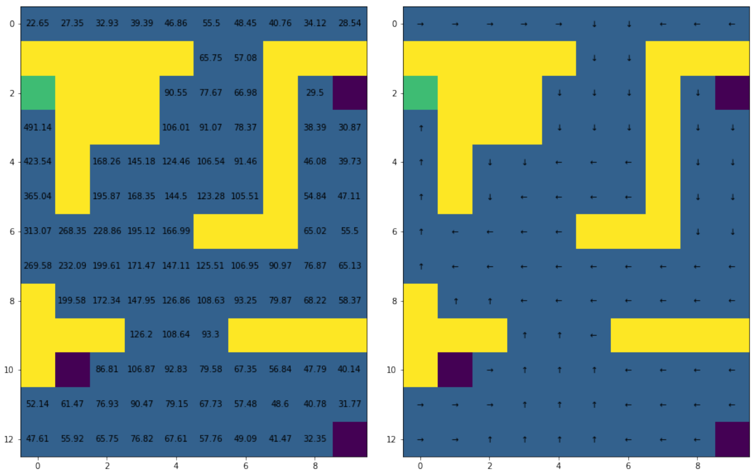

Figure 1: DP State value and policy ($\gamma = 0.88$, $p = 0.9$)

1.3 Parameter Investigation

The discount rate γ defines how DP values future rewards. I investigate cases where $\gamma = 0.2$ and $\gamma = 0.8$ with $p = 0.88$. For $\gamma = 0.2$, DP is short-sighted, only the four states closest to the reward state have positive values. The rest of the states have negative values around -1.25. The policy resulting from the low γ thus not sensible for states far away from the reward state. For $\gamma = 0.8$, the result is similar to Figure 1 with sensible values and policy.

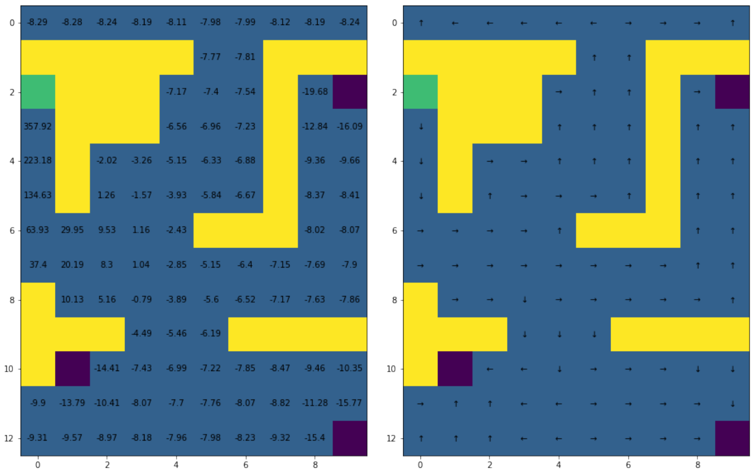

Figure 2: State value and policy output by DP ($\gamma = 0.88$, $p = 0.1$)

The success rate p defines the environment condition. I investigate the cases when $p = 0.1, 0.25,$ and 0.9 with $\gamma = 0.88$. For $p = 0.9$, the values and policy look exactly like Figure 1. For $p = 0.25$, all arrows in the optimal policy are pointing upwards as policy does not affect actions taken by the agent, and all states have negative values except for the six closest to the reward state. For $p = 0.1$, the resulting policy will always point to the wrong direction, but state values are sensible (see Figure 2).

2 Monte Carlo Reinforcement Learning

2.1 Method & Design

I use first-visit evaluation to evaluate the state values and ϵ-greedy policy to choose the action at each state. I chose first-visit evaluation because it is an unbiased estimator of the true state values, as each state-action values sampled from different episodes are independently and identically distributed. The ϵ-greedy policy is applied because while we aim to always select optimal action in a given state, due to nature randomness in MC learning, the agent may not discover the real optimal policy if we keep acting the same. Therefore we must allocate some probability to act randomly for environment exploration.

I set $\epsilon = 0.4$ for ϵ-greedy policy to start, because it has a decent learning speed and outcome (see section 2.5). Let k be the number of episodes ran so far. For every new episode I discount ϵ by 0.9995^k, i.e. $\epsilon_k = \epsilon * 0.9995^k$ to reach a deterministic policy when the learning finishes. I stop the learning at $k = 7000$ because the learning converges roughly at $k = 2000$ for different parameter values (see Figure 4 and Section 2.5). There is no learning rate parameter α as I average state values in first-visit evaluation to compute state-value estimates. The discount rate $\gamma = 0.88$ is given in the instruction, thus no need for consideration.

The two assumptions of MC learning are satisfied: 1. MC agent can reach all states at any given starting point. 2. episodes are guaranteed to terminate.

2.2 Output

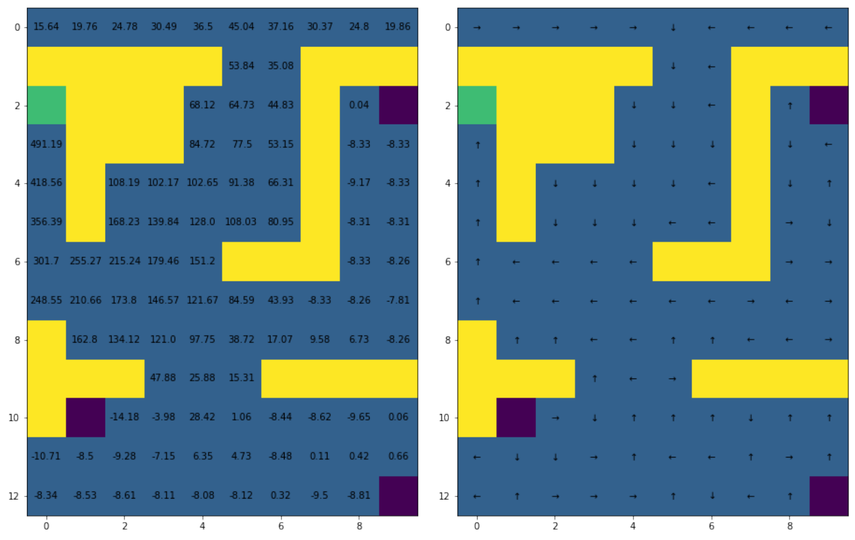

Figure 3: MC State value and policy ($\epsilon = 0.4$, $\gamma = 0.88$)

2.3 MC Variability

The variability results from the different traces that the agent samples each time. Different traces lead to different updated policies, state values, and learning curves. Therefore, the performance of MC has high variability.

By Central Limit Theorem, if we run the agent infinite times and average the total reward over each run, the distribution of the averaged total reward will converge to a normal distribution with mean centred at the reward corresponding to the agent's true performance. The sufficient number for convergence given by rule of thumb is

30. However, since we are not sure about the level of variability of the MC agent, I decide to go for 50 just to be safe. The other way to view this is that we have 10 states to start and we can take 4 actions at each starting state. There are 40 possible starting points, so we should replicate the learning process more than 32 times.

2.4 Learning Curve

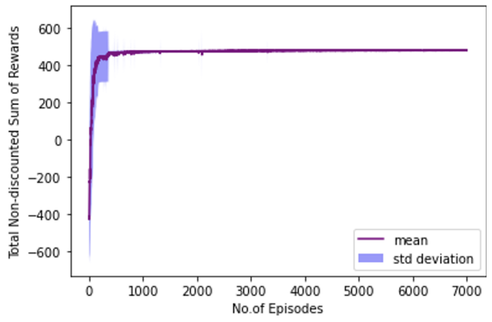

Figure 4: MC learning curve over 50 runs ($\epsilon = 0.4$, $\gamma = 0.88$)

2.5 Parameter Investigation

Exploration parameter ϵ affects how likely the agent will explore the environment at the beginning of the learning. Let Figure 4 be the baseline learning curve, which is generated with $\epsilon = 0.4$. The curve has high variation in the first 1000 episodes then stabilise afterwards. There is a clear cut-off in standard deviation at around 800-th episode.

The way to interpret Figure 4 is that in the first 800 episodes, ϵ is still high and the agent has not found the optimal policy yet, so there is large variability in the reward due to environment exploration. However, after 800 episodes, the agent has found optimal policy and ϵ becomes lower after discounting, so it starts to follow the optimal policy and take actions accordingly, thus variation drops.

Figure 5 shows learning curves with varying ϵ. If we start with $\epsilon = 1$, the curve shows a lower slope and a larger variability in the first 1500 episodes compared to the baseline curve. The higher ϵ prolongs the exploration period. Although the agent may have discovered the optimal policy before the 1500-th episode, higher ϵ encourages it to keep exploring the environment, leading to a slower convergence rate. Although the learning takes longer, it successfully converge to the maximum possible total reward as ϵ gradually decreases.

If we start with $\epsilon = 0.1$, we discourage the agent to explore from the beginning. Setting ϵ too low leads to high variability in performance across each run. As MC samples different paths each time, if we keep the policy too deterministic from the start, the agent may fail to find the path with the greatest reward. It is worth noting that the learning curve given by $\epsilon = 0.1$ converges to the lowest mean reward among the three learning curves.

If we start with $\epsilon = 0.7$, the result lies between $\epsilon = 1$ and $\epsilon = 0.4$. There is no variation drop in the graph but it converges faster than $\epsilon = 1$.

To conclude, setting ϵ too low may fail to find the optimal policy, while setting it too high slows down the convergence rate. Therefore I chose $\epsilon = 0.4$, which converges quickly but also gives correct policy and value estimates.

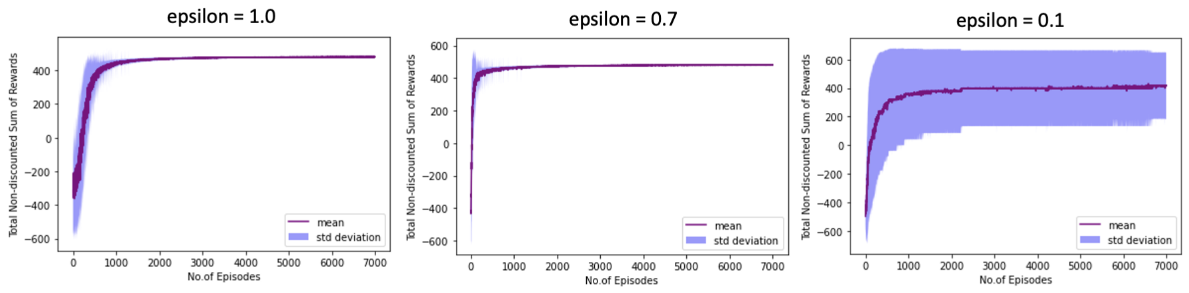

Figure 5: MC learning curve with varying ϵ

3 Temporal Difference Reinforcement Learning

3.1 Method & Design

I use Q-learning with ϵ-greedy policy. I choose Q-learning because it is the most reward-driven while risk-taking TD algorithm. It generates the policy according to the maximum expected reward in the one-step look ahead, thus gives a more optimistic output than Sarsa, which does not use maximum expected reward to derive its policy. Although Q-learning is riskier than Sarsa, it's fine to use it here as we do not incur serious loss if the agent fails to reach the reward state in the first few episodes. Moreover, in my given maze, there is no danger (absorbing states with negative rewards) on the way from any starting points to the reward state. Therefore applying Q-learning gives me decent outcomes without too much variability. Similar to the case in MC, the reason for applying ϵ-greedy policy is to act greedy while explore the environment at the same time.

I set $\epsilon = 0.4$ for ϵ-greedy policy to start, since $\epsilon = 0.4$ gives a nice learning curve (see Figure 7 and Section 3.4 for justification). Moreover, applying the same starting ϵ as MC allows us to compare the TD result with the MC result. For comparison purpose, I discount ϵ with the method mentioned in MC learning and stops the learning at 7000 episode, which is allowed as the learning converges well before 7000 episodes. I set $\alpha = 0.1$ (learning rate) because it gives me a smooth learning curve (see Section 3.4). The discount rate γ is given, thus no need for consideration.

The assumption to apply TD learning is that the environment exhibits Markov property, i.e. current reward does not depend on previous actions taken. The maze satisfies Markov property, so we can apply TD.

3.2 Output

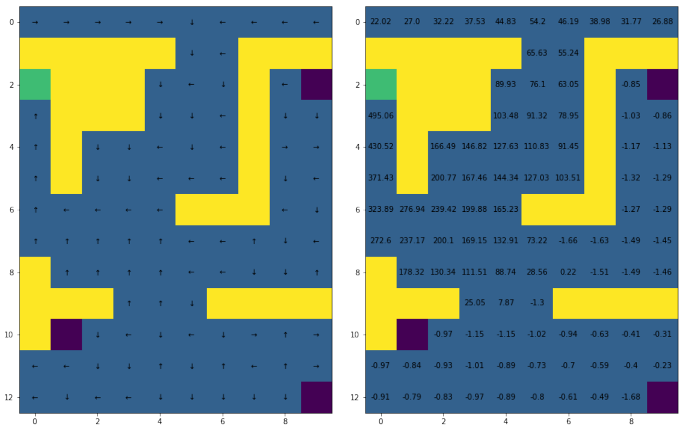

Figure 6: TD state value and policy ($\epsilon = 0.4$, $\alpha = 0.1$, $\gamma = 0.88$)

3.3 Learning Curve

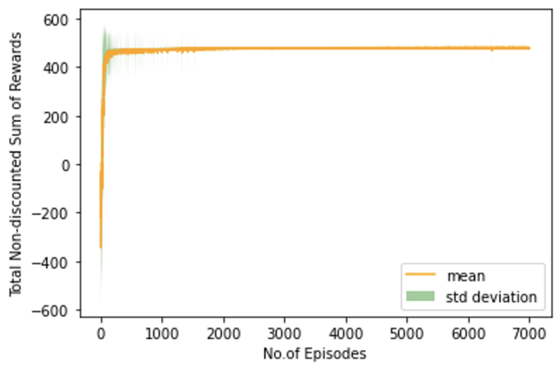

Figure 7: TD learning curve over 50 runs ($\epsilon = 0.4$, $\alpha = 0.1$, $\gamma = 0.88$)

3.4 Parameter Investigation

Let Figure 7 be the baseline learning curve, which is generated with $\epsilon = 0.4$ and $\alpha = 0.1$. The curve stabilises after 500 episodes and shows almost no variability after 2000 episodes.

Figure 8 shows varying ϵ with a fixed $\alpha = 0.1$. Setting $\epsilon = 1$ at the start encourage the agent to explore, leading to high variability in rewards in the first 1000 episodes and a slower learning rate. Setting $\epsilon = 0.1$ discourage the agent to explore from the beginning, so the mean reward converges quickly in 500 episodes. However, unlike the case in MC learning, TD shows low variability across each run even with a low starting ϵ. The low variability attributes to the nature of TD learning: it updates the policy every step, unlike MC which only updates every episode. For TD, although setting $\epsilon = 0.1$ in converges quicker than $\epsilon = 1$, the former shows a higher variation in later episodes. Setting $\epsilon = 0.7$ has the result between $\epsilon = 0.4$ and $\epsilon = 1$.

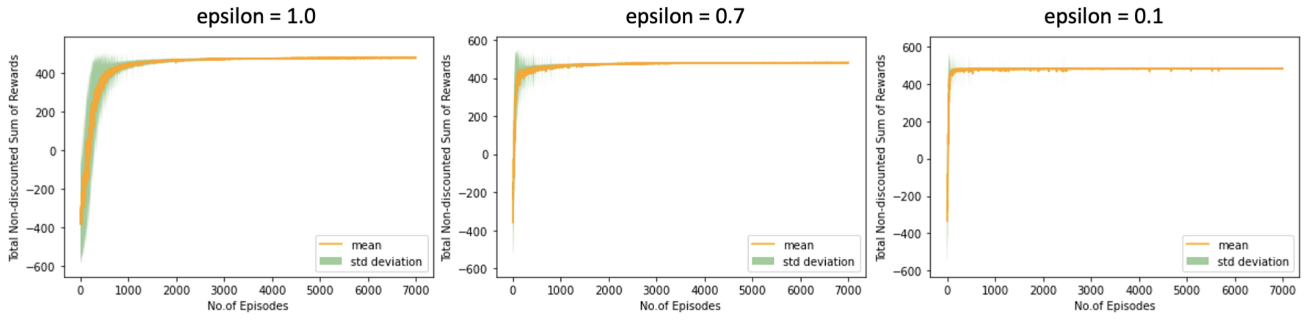

Figure 8: TD learning curve with varying ϵ

Figure 9 shows varying learning rate α with a fixed $\epsilon = 0.4$. The learning rate controls how much the agent learns from the prediction error. If $\alpha = 0.9$, the agent adjust heavily according to the error, thus there are more fluctuations in the learning curve. If $\alpha = 0.1$ as shown in Figure 7, the agent adjust slowly, thus giving a smoother learning curve. Setting $\alpha = 0.5$ gives results between $\alpha = 0.1$ and $\alpha = 0.9$.

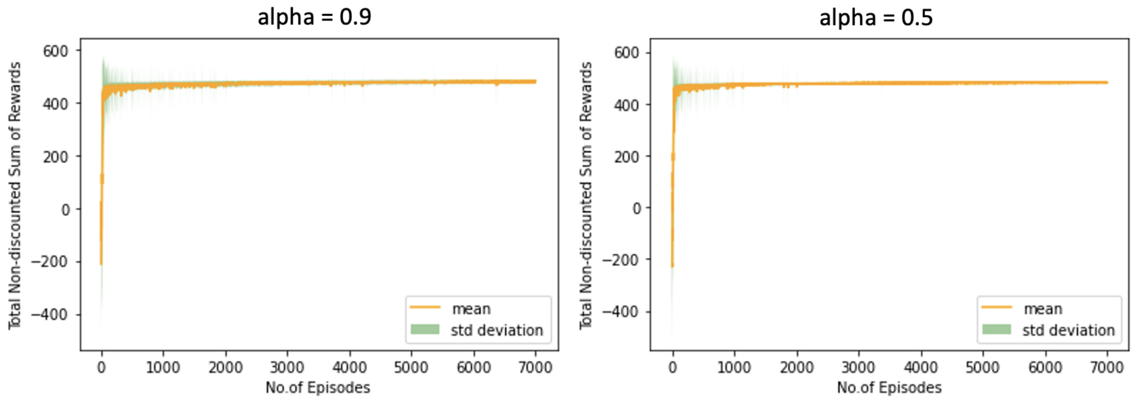

Figure 9: TD learning curve with varying α

After parameter investigation, I choose $\epsilon = 0.4$ and $\alpha = 0.1$ for TD learning. Setting $\epsilon = 0.4$ allows the algorithm converges quicker than $\epsilon = 0.7$ but also avoids fluctuations in later episodes as in the case when $\epsilon = 0.1$. Setting $\alpha = 0.1$ gives me a smoother learning curve, i.e. the reward in later episodes stabilises. Theoretically low α may take longer time to converge, but since the maze is small, the convergence rate does not vary much with larger α. As the result, I use a low α to generate more stable outcomes.

4 Comparison between Learners

4.1 Mean Square Error along Episodes

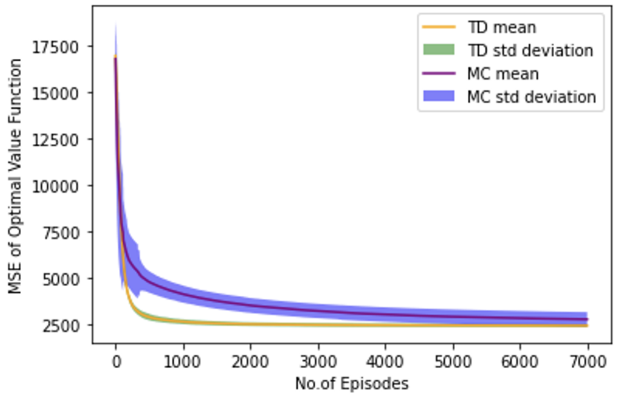

Figure 10: MSE of MC and TD over 50 runs

4.2 Analysis of MSE

The mean square error (MSE) of the state values for both learners converges from 16000 to roughly 2500, indicating that the learners are able to learn correct state values in 7000 episodes. The success in the learning attributes to the satisfication of learning assumptions. For TD, the environment exhibits Markov property, so TD are able to learn. For MC, the agent can visit every state from any starting points and the episodes are guaranteed to terminate.

In terms of the convergence rate, TD converges much faster compared to MC. The reason is that TD updates the policy more frequently: TD updates every step in an episode while MC only updates when an episode finishes. The higher frequency in updating leads to a faster convergence rate.

For variance, TD shows much lower variability in MSE than MC. The reason is similar to the one mentioned above. As TD updates more frequently, it reaches the optimal policy quickly so there is not much variability due to exploration. On the other hand, since MC updates after an episode finishes, the estimated state values will largely depend on the path taken by the agent in each episode, resulting in larger variance.

Theoretically if we run infinite episodes, the MC will converge to lower MSE than TD. Since MC only uses empirical reward, i.e. does not bootstrap, the state values estimated by MC are unbiased. On the other hand, TD applies bootstraping so the estimated state values are biased. However, in 7000 episodes the MSE of the two learners converge to similar values. If longer episodes are run, MC is highly likely to outperform TD as at the end of the graph, some of the MC have already outperformed TD.

4.3 Scatter plot of MSE vs Total Reward

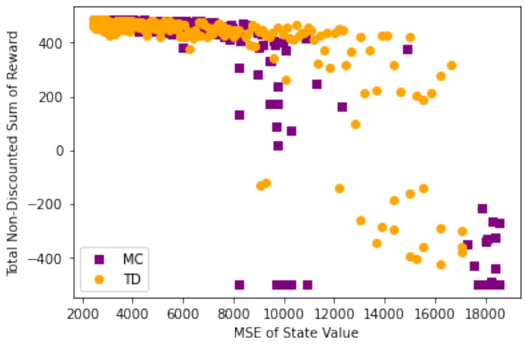

Figure 11: Relationship between MSE and total reward in one run

4.4 Analysis of MSE vs Total Reward

Figure 11 shows the relationship between MSE in state values and the total reward that the agents received in one episode. In general, lower MSE associates with higher total reward. Therefore it is important to have good state value estimates to obtain a good reward. Although the general trend is similar for the two agents, there are some difference between them.

TD shows a stronger monotonic relationship between MSE and total reward than MC. Whenever the MSE is lower, TD is almost guarantee to have a better reward. For MC this is not obvious. For example, given MSE=10000, the reward that MC received vary from -400 to 400. This may due to the higher variability in the MC learning.

TD shows a more even spread in MSE values. For example, MC only has two points between MSE=16000 and MSE=12000. The gap in MSE values in MC may be due to that MC updates its policy off-line. My guess is that in the first few episodes, MC does not have enough data to learn correct state values, so it has high MSE and low rewards (the lower right corner of Figure 11). However, after few episodes, it collects enough experience and produce state value estimates which are much more accurate than the previous ones, so the MSE drops. On the other hand, since TD updates its policy onine, so there is a gentle decrease in MSE as the number of episode increases, unlike MC which has an obvious drop.