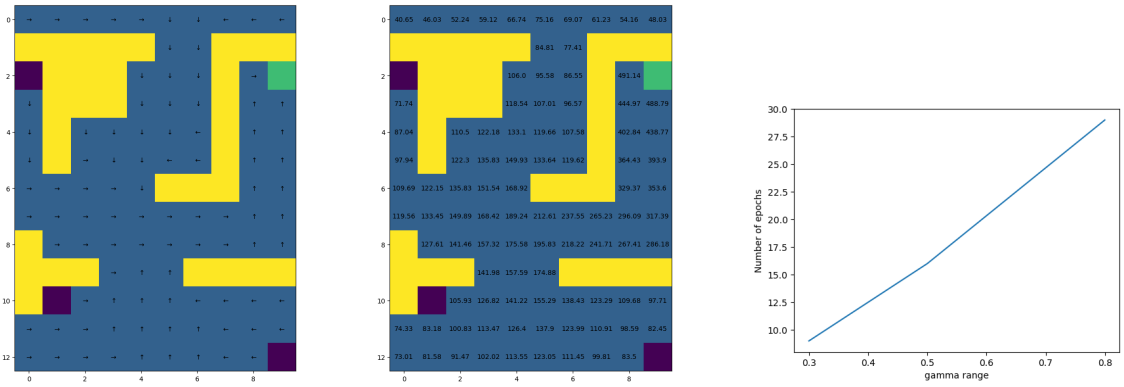
# 1 DP

1. The value iteration method was used to implement dynamic programming. According to my CID, $\gamma = 0.8 + 0.02 * 6 = 0.92$, and probability of success $0.8 + 0.02 * (9 - 6) = 0.86$, and the $p$ parameter for the envieonment becomes 0.14. For DP, we assume that we know all of the information about the environment, including the absorbing states, rewards, topology of the map, and the transition matrix. I chose *value iteration* as my algorithm. Compare to policy iteration, value iteration does not need a epoch-wise policy evaluation. By updating value function according to Bellman equation, the algorithm yield the policy at the end.

2. Graphical representation of policy and values show as 1




(a) policy (b) values (c) gamma-epoch number

Figure 1: DP output

3. Impacts of $p$ and $\gamma$: see fig(c) above for convergence epoch

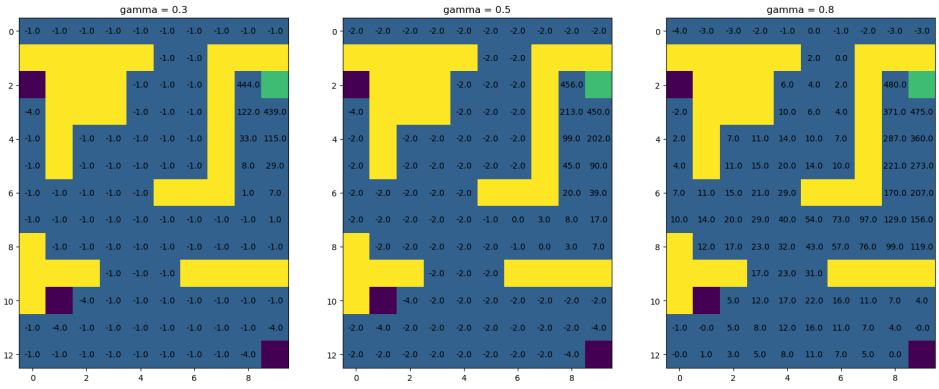



Figure 2: Graphical representation of the value function for each gamma

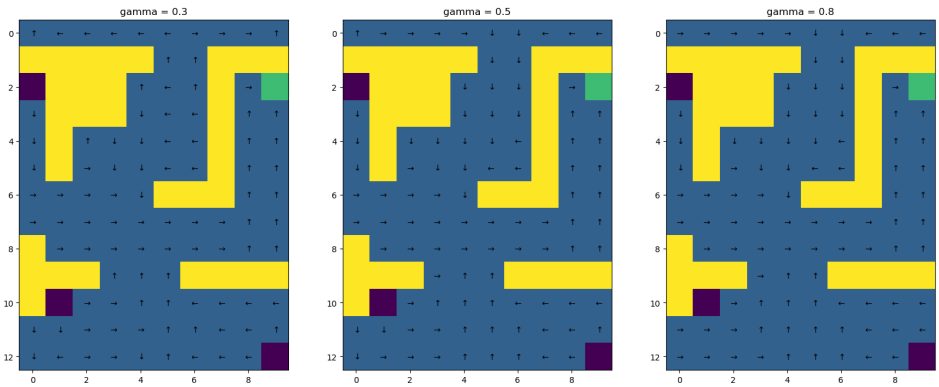



Figure 3: Graphical representation of the policy for each gamma

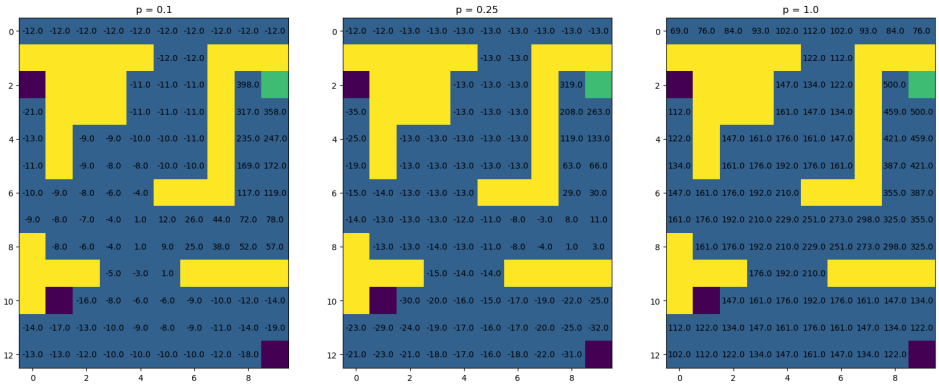



Figure 4: Graphical representation of the value function for each p

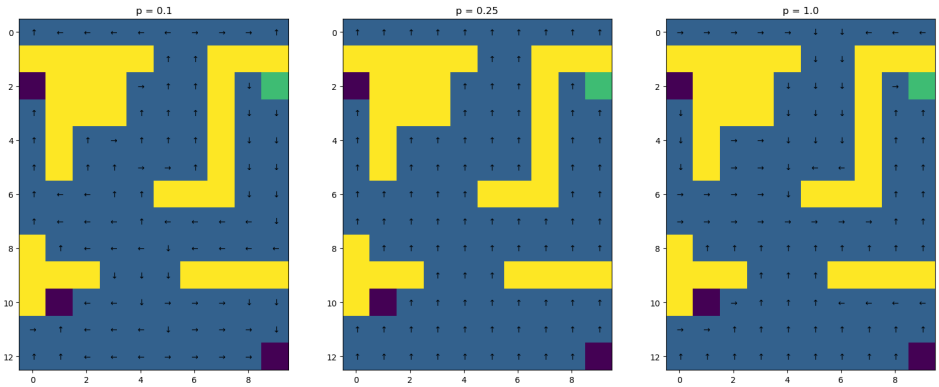



Figure 5: Graphical representation of the policy for each p

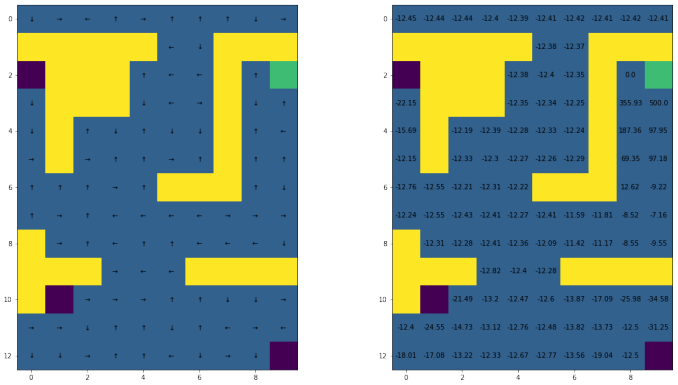
For $p$ values, $p < 0.25$ agent makes *inverse* actions, because it is more likely to go the other way of the action. $p > 0.25$ is the normal case. $p = 0.25$ makes all action having the same effects(random walk), so learning become meaningless. But all three cases generates reasonable value functions, because I used value iteration method. For $\gamma$ values, only slight difference incurred. Take those arrows in bottom left in 3 for example, shorted sighted agent are *escape* from the negative absorbing state. Because

for them, escaping from a nearby negative state is more important than pursuing a far rewarding state, the value function shows the same, negative values appear for short sighted agent near negative absorbing states, but rare for $\gamma = 0.8$.

# 2 MC

1. I used $On - policy\ \epsilon - greedy\ first\ visit\ MC\ control\ algorithm$ to implement the MC method. I my code, I set $\epsilon = 0.1$ for the greedy policies. Soft policy gives the agent opportunities to explore better path, rather than stuck in the local minimum after finding a correct by not good path. This hyper-parameter is first set to be $= 0.1$, to be tuned. A very low $\epsilon$ leads the agent stuck in local minimum, instead of converging to global optimal. Very high $\epsilon$ leads the optimization too hard to converge. Compare to previous DP method, MC method assumes we know nothing about the environment at the beginning, for example what do walls look like, or which absorbing states are rewarding state.

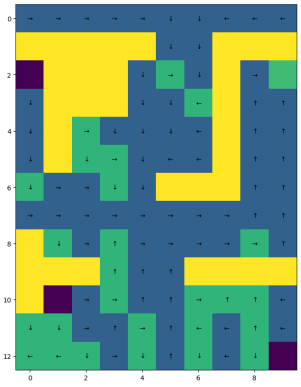
2. Output of MC method



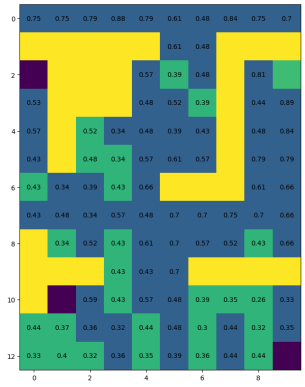
(a) policy (b) values

Figure 6: MC output

3. I tried to run the MC method for 20 times and do statistics for their outcomes. I plot the statistics to see the variability of the method. For presentation, I labeled the comparisons with the DP method in order to check the variability. The light green blocks are those states that MC method gives a answer different from previous DP method, where blue blocks agree to DP.



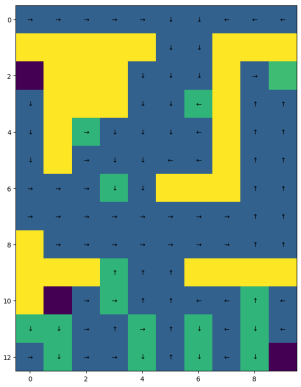
(a) max voted policy



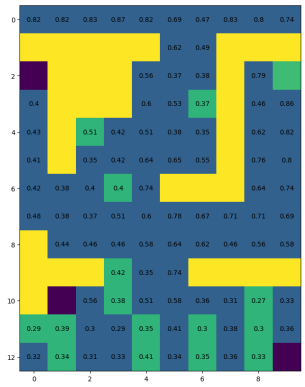
(b) ratio of the max voted policy

Figure 7: MC variability output, 20 replicates

For max voted policy, I used the statistic for each state: $Y = argmax(X_1, X_2, X_3, X_4)$, where $X_i = \sum X_{ij}$, and $X_{ij}$ is the probability of taking action $i$ during the $j$th replication. This statistic shows a general preference of the MC method under this environment. For ratio of the max voted policy, for each state, $Y = \frac{max(X_1,X_2,X_3,X_4)}{sum(X_1,X_2,X_3,X_4)}$, the value represent how the *majority policy* given by the left figure dominants all possible policies. To be clearer, all state will have a value between 0.25 and 0.925(this number comes from $1 - \frac{3}{4} \times \epsilon$, where $\epsilon = 0.1$.
The source of the variability of MC method comes from the random walk. As we use the $\epsilon$ *greedy* policy. In general the green blocks are quite many, means the policies are not that stable after 20 replicates.
We continue run 30 replicates, increase the replicates to 50, and do the same plot again.



(a) max voted policy



(b) ratio of the max voted policy

Figure 8: MC variability output, 50 replicates

We have seen a significant drop in number of light green blocks on the new plot. I tried 80 replicates after this, but the improvement is not significant, that makes 50 a $sufficient$ number of replicates.

4. Learning curve plot

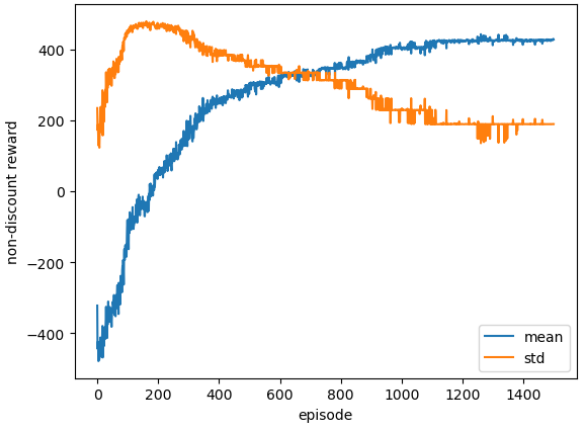
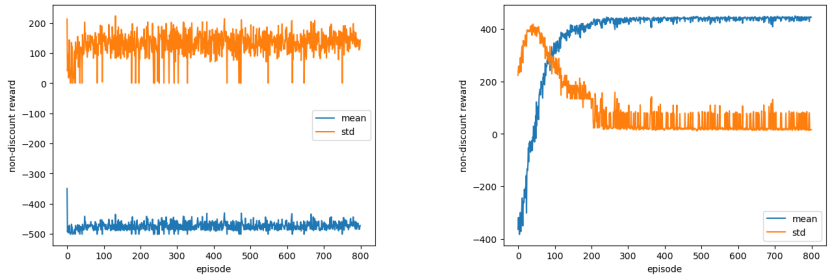



Figure 9: Learning curve of MC

5. Change $\epsilon = 0.0, 0.5$.




(a) $\epsilon = 0.0$ (b) $\epsilon = 0.5$

Figure 10: Changing $\epsilon$

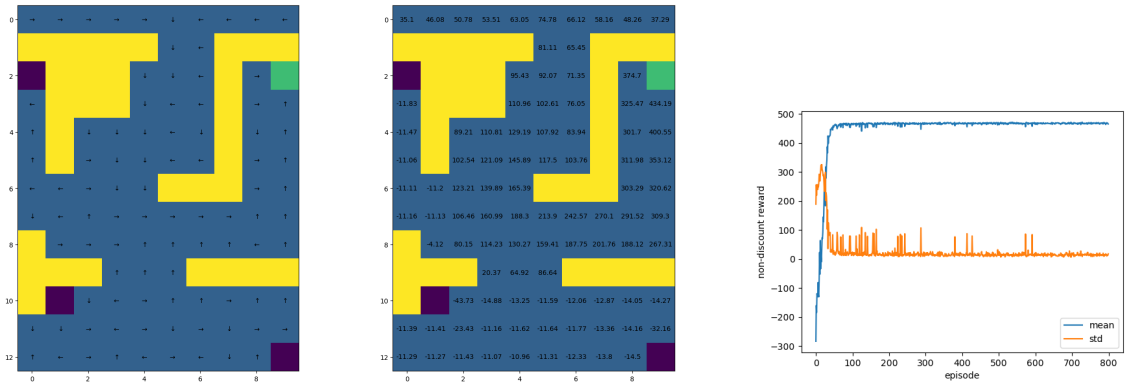
From the plot, it is clear we cannot use $\epsilon = 0.0$, the agent simply cannot learn. The agent keeps going to nowhere and was killed by the program after too many moves. Because the random walk will not work in this case, after a iteration, and all policy on the path become deterministic. As it is unlikely that the first random trace hit the reward, the agent stuck in this wrong policy and cannot get out. Compare $\epsilon = 0.1$ in previous question and $\epsilon = 0.5$, we see a faster learning curve for 0.5. For large $\epsilon$, the agent prefer exploring rather than follow its old experience, so it is more likely yo find a good path early, see from 9, the std curve for $\epsilon = 0.1$ has not step down to zero after 1500 episodes, implies the softness is not enough. After all, I will set $\epsilon = 0.3$ in later comparisons.

# 3 TD

1. I choose SARSA to implement TD method. Compare with Q-learning, SARSA are more likely to perform a safer strategy. In our environment, we have absorbing state that have negative rewards. Our goal is to solve the maze, and the size of

maze is small compare to the rewards in the absorbing state, so some detour from SARSA will not be that harmful.
For parameters, we have $\epsilon$ and $\alpha$ for greedy policy and learning rate respectively. I choose $\epsilon = 0.3$ and $\alpha = 0.5$ for TD.

2. TD output

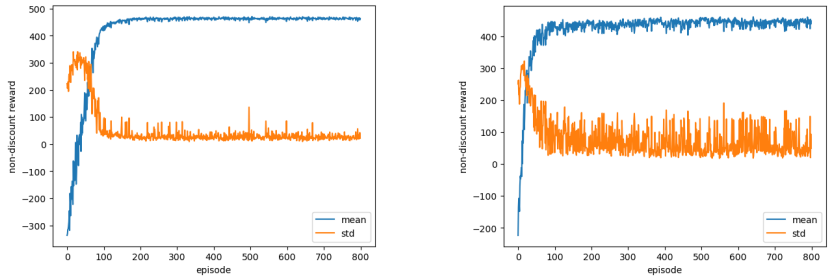



(a) policy (b) value (c) learning curve

Figure 11: TD output

3. Learning curve: (c) above

4. Try $\alpha = 0.1$ and 1.0.



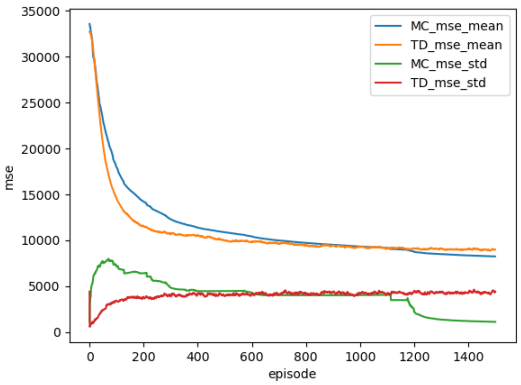

(a) $\alpha = 0.1$ (b) $\alpha = 1.0$

Figure 12: TD change $\alpha$

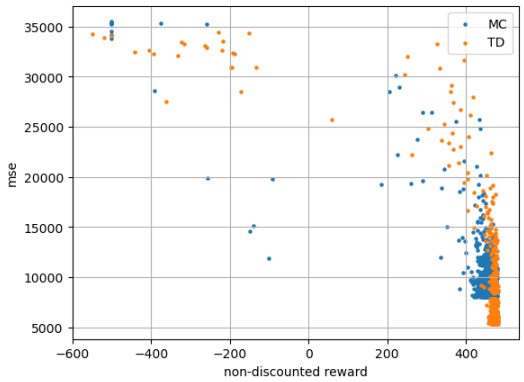
The $\alpha$ learning rate affect how fast the agent learn. Compare the two plot above, large $\alpha$ invoke steep learning curve, but the std keep fluctuating, while low learning rate show otherwise. For $\epsilon$, we observe a small impact to TD when changing it between 0.1 and 0.5. This is different from MC case. As MC rely on a well tuned $\epsilon$ value, or its trace generated will not be reliable. As MC agent will not learn online during the episode, an unexpected detour can ruin entire trace. TD on the other hand, are able to learn from a incomplete trace.

# 4 Comparisons

1. MSE to DP




(a) MSE, STD




(b) reward-mse scatter

Figure 13: Compare MC and TD

2. After taking average between all 50 replicates, both MC and TD have a steady decrease in mse, which TD decreases faster. According to lecture notes, MC agent learn after a trace finish, while TD learning during the episode. So it some wrong traces generated in early stage of learning will not be so harmful. Standard deviation of MSE provides a indicator that how replicated agents different from each other. Both MC and TD have a increase in STD of MSE at early stage, because agents are initialised in the same way, but by randomness leads different learning curves respectively. MC has a higher peak of Std at around episode 100, implies some MC agent had found a fairly good path while other *unlucky* agents were still struggling in some random traces. TD agents do not show this, the performance are more stable.

3. Scatter plot see (b) above

4. A lower variability of TD(orange points) observed. And for same mse value, TD has a higher average of total reward in general. Also, TD agents are more likely achieve a lower mse of value function. As mse is composed by variance and square of bias. It is within expectation from theory that TD usually have a lower variability than MC but still have a relatively small bias.
   In conclusion, in the problem TD performs better than MC. The maze travel is a Markov environment, MC method may better in some non-Markov environment.