Markov Decision Process in Reinforcement Learning

SRIROTH, POONYAPAT 1155205059

ZHAO, SHUWEI 1155210945

Introduction:

Reinforcement learning, widely used in game AI training, is the focus of our project due to our shared interest in developing game AI. This involves the Markov Decision Process, a generalized form of the Markov chain.

Methodology：

- ## Program Setting:

We developed a Python program to simulate a maze game. In a 2D grid maze with walls (black), bombs (red), a target point (green), and a starting point (robot's initial position), the robot is placed randomly and can move UP, DOWN, LEFT, or RIGHT without leaving the grid. The goal is to use MDP to train the robot autonomously to learn and navigate to the target point by assigning rewards after each move.

- ## Relationship with MDP:

1. We have an environment and an agent in the Markov Decision Process.

Environment	The maze with randomly generated bombs and walls
Agent	The robot

2. A Markov decision process (MDP) is defined by $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$

S (State Set) is all the cells in the maze.

A (Action Set) is all the moves the robots might do: {UP, DOWN, LEFT, RIGHT}.

P (Probability: P [S'=s' | S=s, A=a]) is the probability of changing to the next grid, with the current grid and specific action.

R (Reward):

Walls	-10000 and terminate the program (the robot dies)
Bombs	-20 (the robot doesn't die but receives a negative reward)
Target Point	+100 and terminate the program (the robot reaches the final point)
Goal Point	receive the maximum possible reward (which is affected by the size of map)

γ (Discount Factor): We use the discount factor in calculating the reward to control how much we focus on long-term benefit. Discount Factor is a number from 0 to 1. The larger it is, the more we focus on long-term benefit. We are changing it due to the size of the map and the policy.

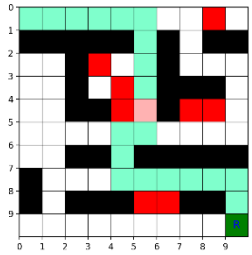
3. We have Action Value Function and Policy to determine what to do:

First, we calculate the Action Value Function $Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t|S_t = s, A_t = a]$, which is the expected reward when we take Action a to the current State s following the Policy (π). But this function result might not be the best in the end.

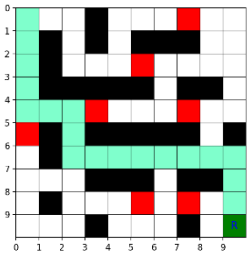
Our policy (π) consists of two parts: a probability of 1- ε to select the action with the best Action Value Function and a probability of ε to randomly explore. We call ε the Exploration Rate, which will be discussed later.

- **Result of the training:**

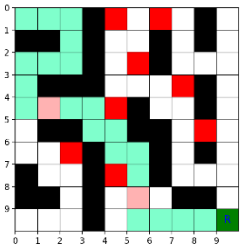
After training the agent on the Action Value Function using Markov Decision Process with 4000 episodes and 25 trainings per sample:



Sample 1



Sample 2



Sample 3

	The percentage of the trainings that the agent can reach the target point
Sample 1	88%
Sample 2	80%
Sample 3	84%

Challenges and Solutions:

During the implementation of our MDP-based reinforcement learning system, there are some noticeable issues that affected the performance of the agent. These challenges highlight the importance of the adjustment of the hyperparameters or hyperparameter tuning.

Challenges: Unexpected Behaviors in Trainings

1. **Cyclic State Transition**: The agent stuck at local maximum and can't reach target point.
2. **Suboptimal Path**: The agent did not choose the shortest path (the most optimal way).
3. **Going Through the Bombs**: The agent occasionally selected the path with the bombs (travel through the bombs) despite the availability of the path without the bombs.

Solution: Hyperparameter Analysis and Optimization

We have solved these problems by adjusting the two critical hyperparameters:

- **Exploration Rate (ε)**

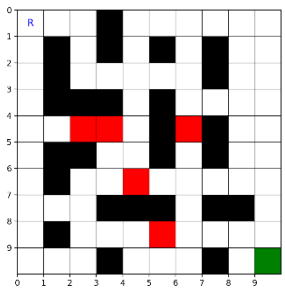
The Exploration Rate or ε determines the balance between the Exploration (discover potentially better strategies) and Exploitation (exploit) of the agent.

In our MDP implementation, the ε-greedy policy determines the probability with which an agent chooses to explore random actions versus exploit its current knowledge. Specifically:

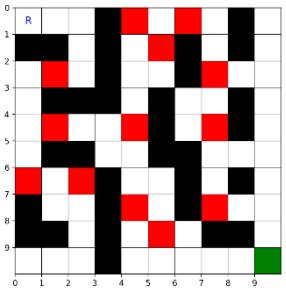
- With probability ε, the agent explores by selecting a random action
- With probability (1-ε), the agent exploits by selecting the action with the highest expected reward

Result of adjusting the Exploration Rate:

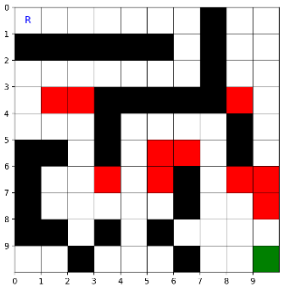
After adjusting the exploration rate, the results for the complex sample environment are shown in the table. It presents the percentage of successful attempts where the robot reaches the target via the shortest path across different exploration rates (4000 episodes, 20 trainings).



Sample 1



Sample 2



Sample 3

	High exploration rate (ε = 0.9)	Optimal range exploration rate (ε = 0.4)	Low exploration rate (ε = 0.2)
Sample 1	85%	90%	40%
Sample 2	85%	95%	45%
Sample 3	75%	90%	55%

Analysis of Exploration Rate:

A low Exploration Rate may prevent the robot from discovering the target point, leading to failure. Conversely, a high Exploration Rate can cause overly frequent exploration, resulting in inconsistent, random-like behavior. Even if the robot learns the optimal path, high Exploration Rate delays its ability to use what agent has learned before.

So, the optimal Exploration Rate results in the balance between Exploration and Exploitation, which makes the robot achieve the target point while using a small number of episodes. A balance is needed between exploration and exploitation. After searching for relevant details, we

learned to improve by gradually decreasing ε (typically exponentially) over time, achieving faster convergence and a better balance.

- **Discount Factor (γ)**

The Discount Factor γ (gamma) determines the agent's emphasis on future versus immediate rewards, shaping its decision-making and ability to learn long-term strategies. For distant target points, γ should be near 1 (e.g., 0.99) to prioritize long-term rewards. Conversely, a low γ focuses the robot on immediate rewards.

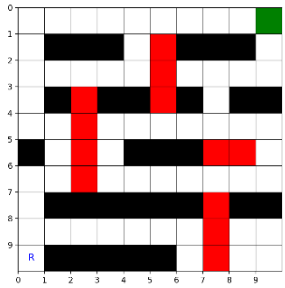
The relationship between the Reward and Discount Factor:

The Reward and Discount Factor are directly correlated, influencing the agent's decisions. Therefore, we must carefully balance them based on desired behavior (e.g. should the robot take a shorter path through a bomb or a longer, safer route?).

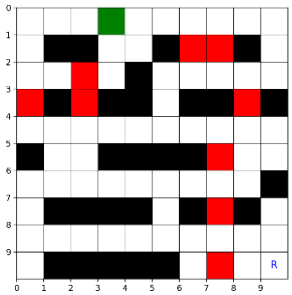
Result of adjusting the balance between the Reward and Discount Factor:

After adjusting the Reward and Discount Factor, the table shows the occurrences of each situation during training (labeled below), with 4000 episodes and 20 training times per sample.

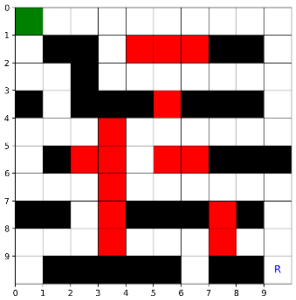
Three complicated sample environments:



Sample 1



Sample 2



Sample 3

Three labels of each situation:

A: the robot chooses to go through the bomb, and reach the target point

B: the robot chooses to avoid the bomb and take the longer path, but still reach the target point

C: the robot cannot reach the target point

	Bomb's reward = -20 Discount factor = 0.95			Bomb's reward = -20 Discount factor = 0.70			Bomb's reward = -1 Discount factor = 0.95		
	A	B	C	A	B	C	A	B	C
Sample1	16	2	2	12	5	3	7	10	3
Sample2	15	1	4	5	11	4	5	13	2
Sample3	11	4	5	2	13	5	2	14	4

Analysis of the result:

In the baseline case, where the bomb's Reward is -20 and the Discount Factor is 0.95, the robot avoids the bomb. In the second case, with a very low Discount Factor, the robot prioritizes immediate rewards and chooses to go through the bomb for a negative reward instead of taking a longer, bomb-free path. Similarly, in the third case, if the bomb's Reward is only slightly negative, the robot will take the shorter path through the bomb to receive the small negative reward.

However, none of these scenarios is wrong, as the behavior depends on the desired outcome. If we want the robot to go through the bomb and accept the negative reward, we set the parameters like in the second and third cases. If avoiding bombs is the priority, we use settings from the first scenario.

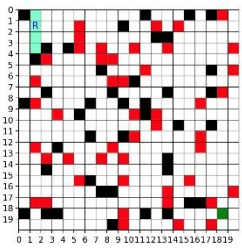
Experiments and Results:

In the process of searching for the effect of different exploration rates, we started to think **if the exploration rate has to do with the minimum training times.** Then we made enough tests to explore. We tested in 6 circumstances; each circumstance is tested 50 times. And we found there are three situations (ε for Exploration Rate, T for Training Times):

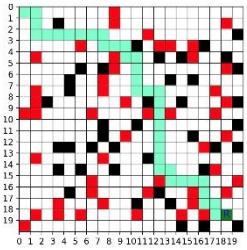
A: Stuck around the starting point

B: Get to the target point but not in the shortest way

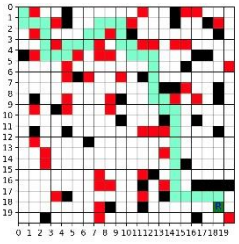
C: Get to the target point in the shortest way



A



B

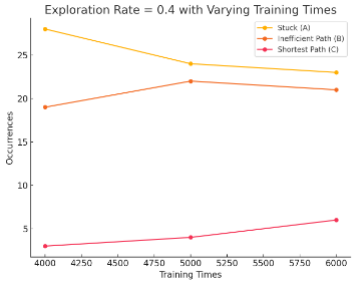


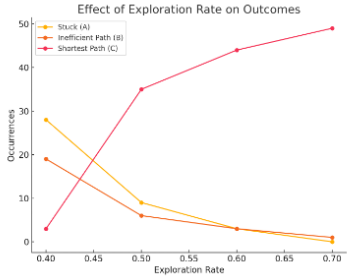
C

Then the results are:

Conditions	Number of A out of 50	Number of B out of 50	Number of C out of 50
ε =0.4, T=4000	28	19	3
ε =0.4, T=5000	24	22	4
ε =0.4, T=6000	23	21	6
ε =0.5, T=4000	9	6	35

ε =0.6, T=4000	3	3	44
ε =0.7, T=4000	0	1	49





Results show that increasing training times from (1) to (3) improves performance, though not significantly. In contrast, higher exploration rates (from (1) and (4) to (6)) reduce training times significantly and enhance performance.

In summary: **Higher Exploration Rates significantly reduce Training times.**

Conclusion:

This project demonstrates the application of the Markov Decision Process in training an agent for autonomous navigation in a complex maze. By changing key parameters like Exploration Rate and Discount Factor, we balanced Exploration and Exploitation, achieving efficient learning and desired behaviors. The findings highlight the importance of parameter tuning in improving agent performance and decision-making.

Reference:

Mignon, A. dos S., & da Rocha, R. L. de A. (2017). An adaptive implementation of ε-greedy in reinforcement learning. *Procedia Computer Science*, *109*, 1146–1151. https://doi.org/10.1016/j.procs.2017.05.431

Puterman, M. L. (1990). Chapter 8: Markov decision processes. *Handbooks in Operations Research and Management Science*, *2*, 331–434. Elsevier. https://doi.org/10.1016/S0927-0507(05)80172-0