UNIVERSITY OF LEEDS

MATLAB Code Implementation of Reinforcement Learning in 2D Maze Exploration Q-LEARNING Versus SARSA.

Yuheng Huo[1]

COMP5400 April 2022

2D Maze Exploration Game

- **Agent** explores the maze to arrive the **Goal** without getting through the **Traps**.
- This is inspired by the Cliff Walking by Sutton and Barto in Lecture Notes in Computer Science.

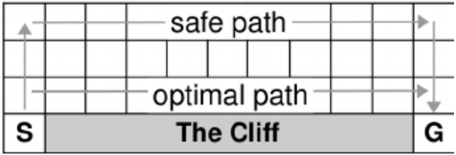


Figure 1: Map of Maze

What is Reinforcement Learning?

- **Reinforcement Learning (RL)** aims to use observed rewards to learn an optimal policy for the environment.
- **Interaction** with Environment.
- **Strategies** to accomplish a specific purpose or maximize benefits.

Is RL Bioinspired?

- **RL** was inspired by **Behaviourist Theories** in Psychology.
- Learning is the process of creating a direct link between **Stimulus** and **Response** through conditioning.

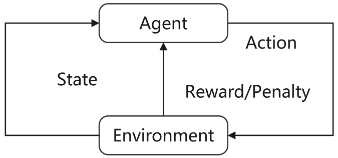


Figure 2: Process of Reinforcement Learning

Examples

Figure 3: Reinforcement Learning Applications in Games

Introduction

- Q-Table.
- Choose Action.
- Feedback from environment.
- Update Q-Table.

```
Initialize Q(s, a) arbitrarily
Repeat (for each episode):
    Initialize s
    Repeat (for each step of episode):
        Choose a from s using policy derived from Q (e.g., ε-greedy)
        Take action a, observe r, s'
        Q(s, a) ← Q(s, a) + α[r + γ max_a' Q(s', a') − Q(s, a)]
        s ← s';
    until s is terminal
```

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\big[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big]$
    $s \leftarrow s'$;
  until s is terminal

Figure 4: Process of Q-Learning

Q-Table

- Q-Table is the **Code of Conduct**.
- Q-Table is **Updated** in each episode.

	State 1	State 2
Action 1	(S_1, A_1)	(S_2, A_1)
Action 2	(S_1, A_2)	(S_2, A_2)
Action 3	(S_1, A_3)	(S_2, A_3)
Action 4	(S_1, A_4)	(S_2, A_4)

Table 1: Example of Q-Table.

Choose Action

- Q-Learning is **Value Base**.
- **Epsilon Greedy** ε is introduced to make random decisions.

	State 1	State 2
Action 1	1	4
Action 2	2	3
Action 3	3	2
Action 4	4	1

Table 2: Example of Q-Table.

Q-Table Update

$$Q_{(S,A)} = Q_{(S,A)} + \alpha * \left[\mathrm{Reward} + \gamma * \max_{a'} Q_{(S',A')} - Q_{(S,A)}\right]$$

- $\mathrm{NewQ} = \mathrm{OldQ} + \alpha * (\mathrm{Actual} - \mathrm{Estimation})$.
- A is the action chosen in state S.
- S' is the state after action A.
- α is the learning rate, deciding how much of the error is to be learned at this time.
- γ is the attenuation rate to future rewards.
- Reward is reward from environment after action A.
- $\max_{a'} Q_{(S',A')}$ is the maximum action value in state S'.

Introduction

- **SARSA** or State-Action-Reward-State'-Action' is also using **Q-Table** to store action values.
- The **Decision Making** or choosing action process is same as Q-Learning.

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\big[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big]$
    $s \leftarrow s'$;
  until s is terminal

Figure 5: Process of SARSA

Q-Table Update

$$Q_{(S,A)} = Q_{(S,A)} + \alpha * \left[\mathrm{Reward} + \gamma * Q_{(S',A')} - Q_{(S,A)}\right]$$

- SARSA chooses the corresponding Action in next state, but Q-Learning does not choose it at this state.
- $Q_{(S,A)}$ is updated based on the $Q_{(S',A')}$, but Q-Learning is based on $\max_{a'} Q_{(S',A')}$.

Simulation Enviroment

- Explorer can move **Up**, **Right**, **Down** and **Left**.
- When explorer reaches Traps (**Grey Blocks**), the simulation in this episode will **end** immediately, and the Reward value is -1.
- When explorer reaches Goal (**Red Blocks**), the simulation in this episode will **end** immediately, and the Reward value is 1.

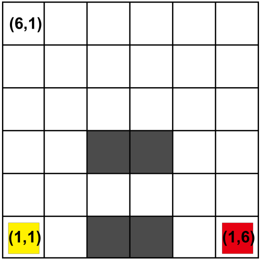
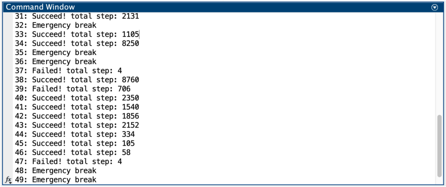


Figure 6: Map of Maze

Training Process

- **Emergency Break** means the number of steps in a single episode is greater than **10000**.
- In Q-Learning case, it has run for more than 50 times, the emergency break never happened.
- In SARSA, emergency break usually happened during the learning process. Once the Q-Table is trained good enough, the emergency break rarely happened.



```
Command Window
31: Succeed! total step: 2131
32: Emergency break
33: Succeed! total step: 1105
34: Succeed! total step: 8250
35: Emergency break
36: Emergency break
37: Failed! total step: 4
38: Succeed! total step: 8760
39: Failed! total step: 706
40: Succeed! total step: 2350
41: Succeed! total step: 1540
42: Succeed! total step: 1856
43: Succeed! total step: 2152
44: Succeed! total step: 334
45: Succeed! total step: 105
46: Succeed! total step: 58
47: Failed! total step: 4
48: Emergency break
fx 49: Emergency break
```

Figure 7: Convergence Speed

Convergence Speed

- The speed of convergence of Q-Learning is Faster than SARSA.

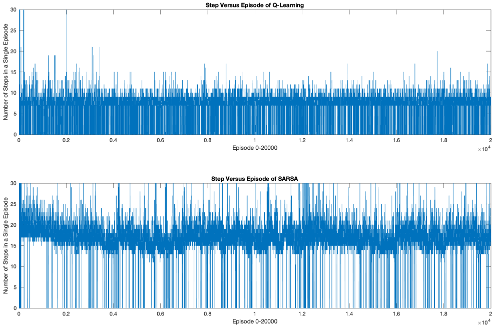


Figure 8: Emergency Break

Fail Rate and Number of Steps

- In a typical case, the **Fail Rate** of the Q-Learning after convergence is **23.84%**, and for SARSA this number is **2.22%**.
- After convergence, the **Average Number Steps** of Q-Learning is less than SARSA.

Routes

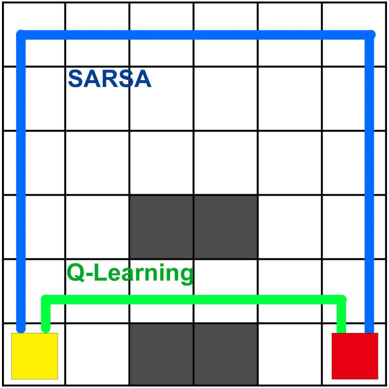


Figure 9: Routes Compare

Conclusion

- SARSA prefers the **Safer** path, Q-Learning prefers the **Optimal** path.
- SARSAR's **Convergence Speed** is slower than Q-Learning.
- SARSAR's **Fail Rate** is less than Q-Learning.

Conclusion and Discussion

- The table can store the value of sates in this case, but fail in more complex problems due to the limited storage and memory. The Q-Value can be generated from **Neural Network**, and the **Neural Networks** are updated in each episode.

Reinforcement Learning and Genetic Algorithm (GA)

- **Fitness Function** $\approx$ **Reward Function**?
- Agent in **GA** does not have a dynamic learning process during its own lifetime. Only problems where the strategy space is sufficiently small or can be easily structured are suitable for genetic algorithms.
- RL is more focused on the interaction with environment and sequence of strategies.
- From my own point of view, GA is like the **DNA** we born with, and RL like the **Knowledge** and **Moral Code** we acquire in our lifetime.