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

COMP5400 Coursework 2

Name: Yuheng Huo

SID: 201199423

Introduction

Reinforcement Learning

Background

Reinforcement learning is inspired by the behaviourist theory in psychology of how an organism [1], stimulated by rewards or punishments given by the environment, gradually develops expectations of the stimulus and produces habitual behaviour that maximises the benefits. [2] The approach is so universal that it has been studied in many other fields, such as game theory, cybernetics, operations research, information theory, simulation optimization methods, multi-subject system learning, population intelligence, statistics, and genetic algorithms. [3]

Definition

Reinforcement learning is the process by which an intelligence learns how to adopt a set of actions in its environment in order to maximise cumulative returns. Reinforcement learning is the learning of a mapping from the state of the environment to the action. [2]

Characteristics

In supervised and unsupervised learning, the data is static and does not require interaction with the environment, e.g. for image recognition, the data is simply fed into a deep network with sufficient variance samples for training, whereas in reinforcement learning, the learning process is dynamic and constantly interactive, and the required data is generated through constant interaction with the environment. [1]Therefore, compared to supervised and unsupervised learning, reinforcement learning involves more objects, such as actions, environments, state transfer probabilities and reward functions. [2]

Trial-and-error search and delayed reward are two of the most important features of reinforcement learning. Trial-and-error search means that there is no direct instructional information, and the intelligence has to interact with the environment in a trial-and-error manner to obtain the best strategy. Delayed reward refers to the fact that an intelligence does not know whether it will receive a positive or negative reward until after it has performed an action. [3]

Process of Reinforcement Learning

Reinforcement learning is essentially the process of finding the optimal decision. Essentially every reinforcement learning experiment can be summarised in the following diagram.

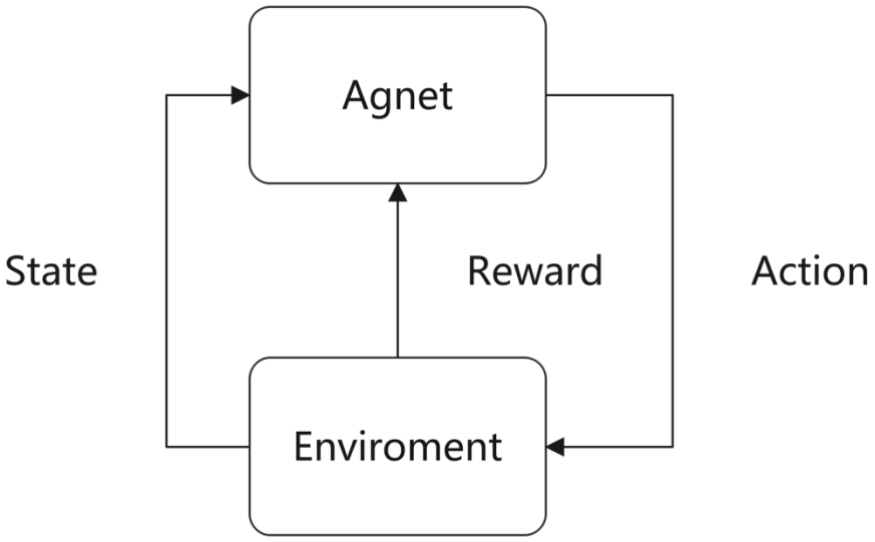

Figure 1. Process of Reinforcement Learning [3]

The intelligence interacts with the environment through the action. Under the action and the Environment, the intelligence will generate a new state and will also receive an immediate reward.

Q-Learning

Inspired by behaviourist learning theory, Q-learning was first prototyped by Watkins in 1989, [4] when he proposed that the memory matrix memorises the previous experiences of the 'intelligence' and that when a new external stimulus is presented, the intelligence generates a corresponding conditioned reflex, memory matrix is similar to the Q-table in Q-learning. [5]

Definition of Q-Learning

Q in the Q-Learning algorithm represents the Quality function, which maps each state action pair (s, a) to the total desired discounted future reward after observing state s to determine action a. [4]

Process of 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'
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
    $S \leftarrow S'$;
  until S is terminal

Figure 2. Process of Q-Learning [4]

Decision of Q-Learning

The algorithm is now in state s1, with two behaviours a1, a2, and as a rule of thumb, in this s1 state, a2 brings a higher potential reward than a1. Here the potential reward can be replaced by a Q-table with s and a. In the memory Q-table, Q(s1, a1) is smaller than Q(s1, a2), so the algorithm decides to choose a2 as the next behaviour. The algorithm now updates the state to s2 , and the algorithm still has two identical choices, repeating the process above, finding the values of Q(s2, a1) Q(s2, a2) in the Q table of behavioural criteria, and comparing their sizes, choosing the larger one. Then, based on a2 we reach s3 and repeat the above decision process here. [4]

Update of Q-Learning

According to the Q-table, since the value of a2 is larger in s1, the algorithm takes a2 at s1 and reaches s2 by the previous decision method, at which point it updates the Q-table used for the decision, and then, instead of taking any action in practice, imagines that it takes each action at s2, compares the magnitude of the actions Q separately, say Q(s2, a2) has the largest value, and multiplies Q(s2, a2) multiplied by a decay value gamma and added to it the reward R obtained when reaching s2 , which is my realistic value of Q(s1, a2), but we previously estimated the value of Q(s1, a2) from the Q table. So with the real and estimated values, we can update Q(s1, a2) by multiplying the difference between the estimated and real values by a learning efficiency alpha that accumulates the old Q(s1, a2) values to the new values. [4]

Parameters in Q-Learning

Epsilon Greedy

It is a strategy used in decision making, e.g. when epsilon = 0.9, it means that 90% of the time I choose the behaviour according to the optimal value of the Q table and 10% of the time I use a randomly chosen behaviour. [5]

Alpha

Alpha is the learning rate, which determines how much of the error is to be learned this time, alpha is a number less than 1. [5]

Gamma

Gamma is the decay value for future reward.

SARSA

The decision making part of State–action–reward–state–action (SARSA) is exactly the same as Q-learning, but SARSA does not update in the same way as Q-Learning. [6]

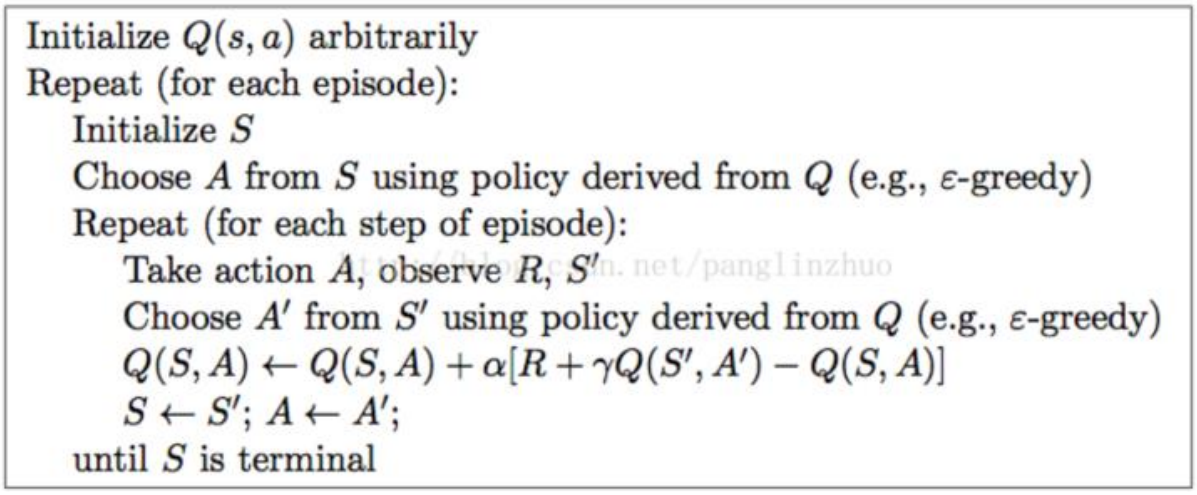

Figure 3. Process of SARSA

Update of SARSA

The algorithm is in state s1, and then picks the action that brings the greatest potential reward a2, thus reaching state s2, and at this step, if Q-learning is used, the algorithm looks at which action on s2 will bring the greatest reward, but when it comes to making a decision, it does not necessarily pick the action that brings the greatest reward, Q-learning In this step, the value of the next action is only estimated. SARSA, on the other hand, is a practical person who does what he says he will do, and the action he estimates at s2 is also the next action to be taken. So the realistic calculation of Q(s1, a2) is changed slightly by removing maxQ and replacing it with the actual Q value of a2 chosen at s2. Finally, as with Q learning, we find the difference between the real and estimated values and update Q(s1, a2) in the Q table. [6]

Q-Learning and SARSA Comparison in MATLAB

This model uses MATLAB code on how to implement exploration in two dimensions using Q-learning and SARSA, and trains the machine on how to implement planning bi to reach the goal automatically.

Basic Introduction

The relevant code has been uploaded in GitHub.
(https://github.com/Neowless/COMP5400_Courework2)

The code can run in MATLAB_R2021b without any additional pack. Please put all of the files in the same file folder.

Q_Learning.m and *SARSA.m* can be directly run, the result can be check in the Command Window and the Workspace. The code is still divided into four pieces, the main programs *Q-Learning.m* and *SARSA.m* and the action function *choose_action.m* and the environment function *get_env_feedback.m*.

I am extremely sorry that I don't have any experience in GUI software developing, apologize for any inconvenience.

Problem Model

The two-dimensional environment targeted by this paper is shown in the figure 4.

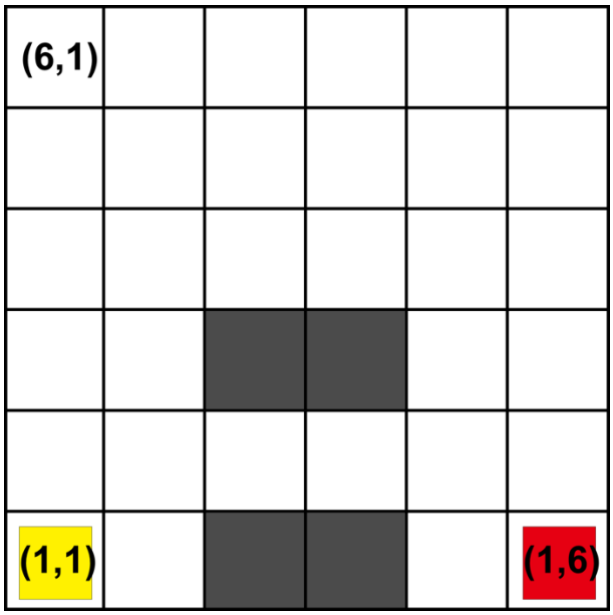

Figure 4. Maze Map

This is a two-dimensional grid model, which I describe as a maze, because it can be extended to any two-dimensional grid model on this basis. The grid is divided into 6 rows and 6 columns with a total of 36 grids, where the blue grid in the bottom left corner is the starting point of the quest, the red grid is our destination, i.e. we want the machine to eventually reach this exit, and the black grid indicates a trap, as soon as it falls into the black grid the round ends in failure. In each grid the machine can move in four directions: up, right, down and left, with the corresponding movements being A=1,2,3,4. The movement of the grid at the edge does not produce any movement beyond the border.

Problem Definition

In order to describe the environment in a form that can be easily expressed by the program, we number each grid from 1 to 36, with the first row being 1,2,3,4,5,6 from left to right, the second row being 7,8,9,10,11,12 from left to right, and so on, numbering all the grids, and this numbering also facilitates the conversion to the coordinates corresponding to each grid.

To enable the machine to learn autonomously, the reward of the environment is defined, with the reward at the exit set to 1, the reward at the trap set to -1, and the reward of any other grid set to 0. This completes the definition of the model

Result Analysis

Learning Process

During the learning 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.

Convergence Speed

In the case of fail and emergency break, the value of steps is set as 0. The speed of convergence is not of certain, but the graphs shown are typical cases.

According to the graphs, it is easy to indicate that the convergence speed of Q-Learning is faster than SARSA in this case.

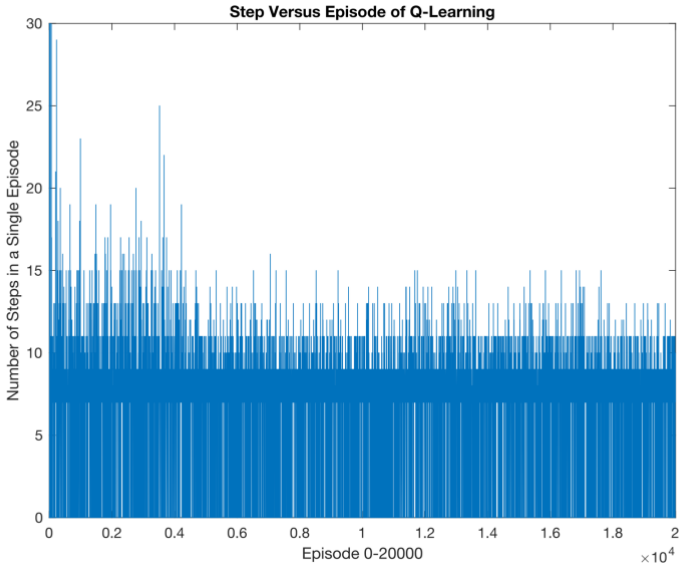

Figure 5. Q-Learning

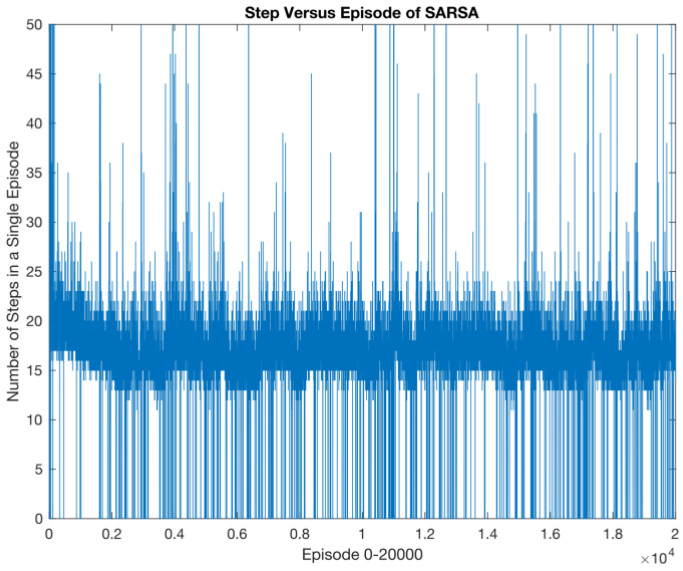

Figure 6. SARSA

Final Performance

According to Figure 2 and Figure 3, when the algorithms are converged, the average value of steps SARSA is much higher than Q-Learning.

However, the fail rate of Q-Learning after convergence is much higher than SARSA. As in these typical cases, the fail rate of the Q-Learning after convergence is 23.84%, and for ARSA this number is 2.22%.

Conclusion

SARSA converges more slowly, and the optimal solution achieved is nothing more than suboptimal. In addition, during the simulation process, SARSA can easily fail in training and fail to achieve the goal. This is because once it falls into the trap, the machine becomes too cautious and stops from there, especially in harsher environments.

The optimal path for SARSA and Q-Learning according to the Q-Table shown is figure.

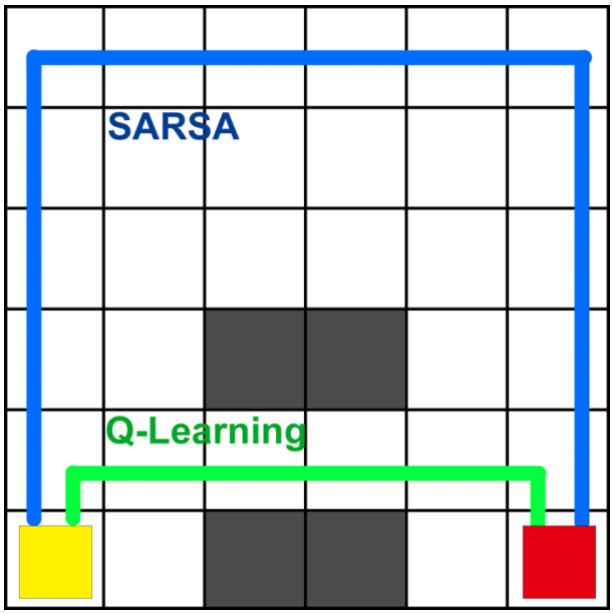

Figure 7. Routes

SARSA is an on-policy, learning what it is doing. Q learning is imaginary learning, so it is also called Off-policy, off-line learning. The comparison revealed that the Q-learning algorithm was more daring and used for experimentation compared to SARSA, which appeared to be more cautious.

References

[1] L. L. M. a. M. A. Kaelbling, “Reinforcement learning: A survey.,” *Journal of artificial intelligence research,* vol. 4, pp. 237-285, 1996.

[2] M. A. &. V. O. M. Wiering, “Reinforcement learning.,” *Adaptation, learning, and optimization,* vol. 12, no. 3, p. 729, 2012.

[3] R. S. a. A. G. B. Sutton, Reinforcement learning: An introduction., MIT press, 2018.

[4] C. J. a. P. D. Watkins, “"Q-learning.",” *Machine learning,* vol. 8, no. 3, pp. 279-292, 1992.

[5] J. a. E. L. Clifton, “Q-learning: theory and applications.,” *Annual Review of Statistics and Its Application,* vol. 7, pp. 279-301, 2020.

[6] D. e. a. Zhao, “Deep reinforcement learning with experience replay based on SARSA.,” *2016 IEEE Symposium Series on Computational Intelligence (SSCI).,* pp. 1-6, 2016.