**INM 426 - Software Agents**

**Lego Home Finding with Q-learning**

Harry Li & Xin Li

|  |  |
| --- | --- |
|  |  |
| Figure 1. London Underground lego map with Lego Store location Leicester Square highlighted | Figure 2. Graphic representation of the underground map on the left |

**Introduction**

This report shows an implement of Q-learning in a navigation task. The impact of different parameters (e.g. discount rate, learning rate, exploration rate) on the learning are explored.

Introduction of Q-learning (a breakthrough, off-policy control, a type of TD, 2 nice properties: use q and 1 step ahead full backup)

1. **Basic**
   1. **Domain and Task**

The domain is the central part of the London Underground with random starting station in each episode and a defined ending station of Leicester Square (station 7 in Figure 2) where the Lego store is based. It is an episodic finite navigation task. The agent is set to find the shortest path from the random starting station to the destination.

* 1. **State Transition Function**

The state provides the agent basis to make an action. It has the Markov property that every state include all the information in the past and enables the agent to make future interaction with the environment. (Sutton and Barto, 2018)

The state transition function represents the successor state an agent could be ended up after taking an action. It is part of the environment outside the control of the agent. In Dynamic Programming (DP), the state transition is controlled by the state transition probabilities. In Monte Carlo (MC) and Temporal Difference (TD) methods, the transition probabilities are generally unknown to the agent. Thus the agent has an equal probability to transit into any possible successor state including the current state. (Sutton and Barto, 2018)

Based on graphic representation in Figure 2, the state transition function can be expressed as:

|  |  |  |  |
| --- | --- | --- | --- |
| **State () → Next State ()** | | | |
| 1→ {1, 2, 5} | 4→ {3, 4, 7} | 7→ {3, 4, 6, 7, 8} | 10→ {5, 9, 10, 11} |
| 2→ {1, 2, 3, 5, 6} | 5→ {1, 2, 5, 6, 9, 10} | 8→ {6, 7, 11} | 11→ {8, 10, 11} |
| 3→ {2, 3, 4, 7} | 6→ {2, 5, 6, 7, 8} | 9→ {5, 9, 10} |  |
| Table 1. State transition function | | | |

* 1. **Reward Function**

The reward signal is the way to communicate to the agent in terms of what to achieve. It is one of the fundamental ideas of reinforcement learning. In general, the goal of the agent is to maximise the cumulative rewards receive in the long run. It can be expressed as , where is the cumulative return received at the terminal state T and R is the immediate reward after taking each action. (Sutton and Barto, 2018)

The reward function is defined as R-matrix in section 1.5 below. It represents the immediate reward an agent receives after transitioning from one state to the next. And it is the input to the Bellman optimality equation follows the Markov Decision Process (MDP). (Sutton and Barto, 2018)

* 1. **Policy**

Policy is a mapping from states to each possible action. The action made at state following policy can be expressed as , where for each . Policy is what the agent ultimately learn through iteratively evaluate and improve state-value function or action-value function (the process is also called generalised policy iteration). One way to improve the policy is to take a greedy approach in respect of or follow Bellman optimality equation ( in the case of Q-learning). (Sutton and Barto, 2018)

To balance the trade-off between exploration and exploitation, an exploration factor is used, where . With the probability of , the agent will randomly make an exploration instead of greedly move towards the state or state-action pair with the highest value function. This is called -greedy policy which is guaranteed to converge to the optimal policy . Further, a decay factor is introduced to reduce over the time to let the agent explore more at the beginning and less in the later stage, which is proven to be efficient. Decayed -greedy policy is employed in this report. (Sutton and Barto, 2018)

* 1. **Graph representation and R-matrix**

The R-matrix is demonstrated below. Rows show the state () and columns show the next state () after taking an action. Numbers represent the immediate reward an agent receives. 0 for every action unless arriving at the destination. - represents null value indicates no link between stations. 100 for reaching the destination, station 7.

|  |
| --- |
|  |
| Table 2. R-matrix |

The graphic representation is shown in Figure at the beginning of the report. The agent can take 1 action by leave and return to the same state, but this is not shown in the graph for better visualisation.

* 1. **Parameters**

The development of an off-policy TD control method called Q-learning was one of the breakthroughs in reinforcement learning (Watkins, 1989). The Q-learning equation is presented below. The new action-value function is equal to the old plus immediate reward by taking an action and the maximum one-step ahead action-value function. (Sutton and Barto, 2018) Learning rate () and discount rate () play their own role detailed below.

* + 1. Learning Rate ()

One property of TD learning, also Q-learning, is bootstrap. It means the current update is based on other estimates. Learning rate () is used to control how much of the agent takes into account the new update. Given the new update is an estimation, we might not want to have a full update but instead steady steps toward the right direction. (Even-Dar and Mansour, 2003, Sutton and Barto, 2018)

The Q value (action-value function) is updated at each episode towards the optimal action-value function under optimal policy . When close to 0, the agent only take a small step towards a larger . When approaching 1, the agent incorporates more update and finally the current value is replaced with new value if . (Sutton and Barto, 2018) For the illustration in the next two sections, a learning rate of 0.9 is used.

* + 1. Discount Rate ()

TD method uses multi-step bootstrap that the value-function is based on the estimates of the value of successor states. Discount rate () is used to represents the uncertainty of the estimates and it determines the present value of it. When close to 0, the agent tends to maximise the immediate and short-term reward. When approaching 1, the agent takes more consideration of future rewards. (Sutton and Barto, 2018) For the illustration in the next two section, a discount rate of 0.8 is used.

* + 1. Exploration factor ()

As noted in section 1.4, -greedy policy is employed in this report. With probability of , the agent takes random exploration. Otherwise, with probability of (), the agent takes greedy action. And also a decay factor () is also employed to let the agent to explore more at the beginning and gradually less after having better understanding of the environment. The exploration factor is set to 0.1 and decay factor is set to 0.95 for performance evaluation in section 1.8.

|  |  |
| --- | --- |
| **Summary of initial parameters setting** |  |
| Learning rate () | 0.9 |
| Discount rate () | 0.8 |
| Exploration factor () | 0.1 |
| Exploration decay factor () | 0.95 |
| Table 3. Summary of initial parameters setting | |

Are we able to find some reference for those parameters initial setting? - Ask Abi/Alex

* 1. **Q-matrix Update**

Q-learning is an asynchronous process, which means only visited state-action pair is updated each time (Even-Dar and Mansour, 2003). To demonstrate Q-learning, 2 episodes of Q-matrix asynchronous update is illustrated below with the parameters stated above (). Station 2 is randomly selected for the illustration. In addition, pseudocode of Q-learning is presented in the box below (Sutton and Barto, 2018).

|  |
| --- |
| **Asynchronous Q-learning (off-policy TD control) pseudocode** |
| Set algorithm parameters:  Initialise , arbitrarily set  Loop for each episode:  Initialise  Loop for each step of episode:  Choose from using -greedy policy  Take action , observe ,        until is terminal |
| Table 4. Q-learning pseudocode |

|  |  |
| --- | --- |
| **Episode 1:** Station 6 to Station 7 |  |
| **Step 1**: initialise Q-matrix |  |
| **Step 2:** choose to go to Station 7 follow policy  Station 7 becomes the current state. Since Station 7 is the terminal state, this episode ends here. |  |
| **Episode 2: Station 2 to Station 7** |  |
| **Step 1**: randomly choose to go to Station 6  Station 6 becomes the current state. |  |
| **Step 2**: randomly choose to go to Station 7  Station 7 becomes the current state. Since Station 7 is the terminal state, this episode ends here. |  |
| Table 4. Q-matrix update in two episodes |  |

* 1. **Performance vs Episodes**

Section 1.7 illustrated update on Q-matrix in two episodes with . To shows the performance with more episodes, we run the Q-learning algorithm for 1000 episodes to demonstrate the performance against the episodes. Since the domain is relatively small, we expect the agent to converge before 1000 episodes.

There are a number of ways to measure the agent’s performance. For the navigation task, the number of steps to the destination and average reward per step (i.e. total reward in one episode / total steps in the episode) are common metrics. The number of steps measures the efficiency of the agent. Average reward per step measures the quality of agent’s decision-making as well as efficiency. For better visualisation and comparability, we set the agent always start from Station 9 instead of random, and head for Station 7.

Q-matrix is a cumulative measure. The values in the matrix increase with the number of episode (Sutton and Barto, 2018). To ensure Q-matrix values stay relatively stable after the convergence, Q-matrix is normalised by dividing the largest value in the Q-matrix ().

|  |  |
| --- | --- |
|  |  |
| Figure 3. Q-learning algorithm in 1000 episodes | Table 5. Descriptive statistics of 1000 episodes |

Figure 3 shows the performance in 1000 episodes. From around 500th episode, it starts to converge that the agent takes 4 steps from Station 9 to Station 7. This agree with our expectation, which shows the efficiency of Q-learning in general. The tables below shows the final Q-matrix and the optimal path from a station to the Station 7.

|  |  |
| --- | --- |
| **Optimal paths:** | **Q-matrix after 1000 episodes:** |
| 1 to 7 (steps: 4):  2 to 7 (steps: 3):  3 to 7 (steps: 2):  4 to 7 (steps: 2):  5 to 7 (steps: 3):  6 to 7 (steps: 2):  7 to 7 (steps: 1):  8 to 7 (steps: 2):  9 to 7 (steps: 4):  10 to 7 (steps: 4):  11 to 7 (steps: 3): |  |
| Table 6. Normalised Q-matrix after 1000 episodes and optimal paths | |

1. **Advanced** 
   1. **Search of Learning Rate**

As stated in 1.6.1, the learning rate controls how much the agent takes the new estimates to update the Q matrix. Section 1.8 shows Q-learning with default parameters converge around 220 episodes. To understand the impact of the learning rate (), a wide range of values are explored: 0.01, 0.1, 0.5, 0.9, 1. The learning rate is not decayed over the episode as Beleznay, Grobler and Szepesvari (1999) promote a fixed learning rate for a quicker convergence.

Empirically, a larger learning rate leads to a quicker convergence however a too big learning rate with no exploration may lead to a suboptimal policy due to the noise in the reward process (Beleznay, Grobler and Szepesvari, 1999, Even-Dar and Mansour, 2003, Sutton and Barto, 2018). Even-Dar and Mansour (2003) suggested the best rate is around 0.85. Given the small size of our task and noiseless setup, suboptimal is not a concern. Therefore, we expect earlier convergence when having higher learning rate.

|  |
| --- |
| **Hypothesis Statement 1** |
| Higher learning rate leads to earlier convergence |

|  |  |
| --- | --- |
|  |  |
| Figure 4. Performance in different learning rates | Table 7. Descriptive stats for different learning rates |

Figure 4 and Table 7 above show the different performance of 5 learning rates. The maximum average reward per step around 60 reveals that convergence reached when learning rate is 0.9 or 1 after 1000 episodes. With learning rate equal 0.01, 0.1 and 0.5, convergence is not reached by 1000 episodes. When checking the Q-matrix with learning rate of 0.5 after 1000 episodes, it is noticed that the matrix is sufficient to guide the agent to follow the optimal path. However, it is still not converged that some of the second best Q values are not in line with the converged Q-matrix in Table 5. According to Figure 4, higher learning rate leads to quicker convergence. This is consistent with our hypothesis.

Point for future study: decreasing learning rate proposed by many other studies (Even-Dar and Mansour, 2003)

* 1. **Search of Discount Rate**

Section 1.6.2 mentioned that the discount rate () determines the present value of future rewards, and control how much the agent weight the future rewards. It is also a mathematical convenience that turning an infinite horizon problem to a finite one. High discount rate means less future reward is reflected, thus lower Q value (Sutton and Barto, 2018). To understand its impact, the following values are explored: 0.01, 0.1, 0.5, 0.8, 1. We expect lower discount rate would lead to slower convergence as fewer future reward is taken into account.

|  |
| --- |
| **Hypothesis Statement 2** |
| Higher discount rate (i.e. far-sighted) leads to earlier convergence |

|  |  |
| --- | --- |
|  |  |
| Figure 5. Performance in different discount rates | Figure 6. Steps with different discount rate |

Figure 5 shows that discount rate have limited impact on the speed of convergence. The convergence is reached before and around 200 episodes when discount rate is 0.01, 0.1 and 0.5. When discount rate approaching 1, Q values become unstable. The convergence is not reached by 1000 episodes when discount rate is 1. This is consistent with the findings from Even-Dar and Mansour (2003). Figure 6 also shows that there are still some large steps after 40 episodes when discount rate is 0.8 and 1. These prove that our hypothesis is not correct.

When discount rate approaching 1, Q-learning backup more future rewards to the current action-value function. Therefore, more actions at each state would lead to maximum rewards. In effect, the agent does not have enough incentive to find out the optimal next action (i.e. many options would eventually reach the destination thus maximum reward). When discount rate equal to 1, it becomes an infinite horizon problem that the reward at the terminal state is fully backup to the current Q value. Thus, the agent losses its direction for the optimal path. In this scenario, most of paths would give the maximum reward.

Future study: change R-matrix to -1 instead of 0

* 1. **Search of Exploration Factor**

The balancing between exploration and exploitation is key in reinforcement learning. The agent needs to try a variety of actions and progressively choose the best. Exploration is one of the few conditions for Q-learning to reach convergence as we need to ensure the agent visits every possible state-action paris. (Even-Dar and Mansour, 2003, Sutton and Barto, 2018)

Section 1.6.3 explains that the exploration factor () determines how much to explore in every episode. It has a direct impact on the speed of convergence, since more exploration leads to less following the optimal action to update Q matrix. of 0.001, 0.01, 0.1, 0.5, 0.9 are explored and the hypothesis station is below:

|  |
| --- |
| **Hypothesis Statement 3** |
| Higher exploration factor leads to slower convergence |

|  |  |
| --- | --- |
|  |  |
| Figure 7. Performance in different value in | Table 8. Descriptive stats for different value in |

Figure 7 shows all 5 cases are converged by 1000 episodes. The decay factor of 0.95 on the exploration factor partially contributed to this. As we can see the average reward per step is getting stable from 800 episodes. However, at the early steps, the agent with a smaller exploration factor getting close to the convergence value of 60 sooner.

Table 8 demonstrated that the value of epsilon is negatively correlated with mean of the average reward per step, but positively correlated with standard deviation. It tells that the agent with the smaller epsilon value explores less and more greedy towards the best action. It is difficult to say if our hypothesis stand in this small task, however the trend is clear.

* 1. **Search of Decay Factor**

The decay factor () controls the speed of the decrease in exploration factor () in episodic learning. As we can see in Figure 7 that the agent explores less in later episodes. This is proved to be efficient and a good practice, as the agent should have explored enough in the early episodes (Sutton and Barto, 2018). The following values of decay factor are explored: 0.1, 0.5, 0.7, 0.9, 0.99. Slow decay (i.e. large decay factor value) makes the agent still explore at the later episodes thus might converge late. High decay (i.e. small decay factor value) makes the agent stop exploring at the early episodes thus might learn a suboptimal policy. In our small task, suboptimal is not a concern, therefore we expect earlier convergence with small decay factor value.

|  |
| --- |
| **Hypothesis Statement 4** |
| Higher decay factor leads to slower convergence |

|  |  |
| --- | --- |
|  |  |
| Figure 8. Performance in different value in decay | Table 9. Descriptive stats for different value in decay |

It is clear in Figure 8 that the agent explore more when decay factor value is large (i.e. slow decrease in exploration factor). For example, the agent stopped random walk actions just before 180 episodes and still exploring at episode 500 when decay is 0.99. Whereas when decay is 0.1, the agent reached a relatively stable status around 400 episodes.

Table 9 tells a same story that the decay factor value is positively correlated with standard deviation of average reward per step, but negatively correlated with the mean value. When decay factor is large, the agent explore more thus higher standard deviation and lower mean value of average reward per step. Therefore, our hypothesis is correct in our task.

* 1. **Search for an Optimal Combination**

The sections above demonstrated the impact on Q-learning by varying a single parameter in each section. The Table 10 below summarises the findings above.

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Direction** | **Impact on convergence** | **Optimal Value Range** |
| Learning rate | Increase | Faster convergence | Between 0.5 and 0.9 |
| Discount rate | Increase | Limited impact but close to 1 is problematic | Between 0.5 and 0.8 |
| Exploration | Increase | Slower convergence | Between 0.01 and 0.1 |
| Decay | Increase | Faster convergence | Between 0.7 and 0.9 |
| Table 10. Findings from section 2.1 to 2.4 | | | |

To find the optimal parameters for our task, a Grid Search on learning rate (0.8, 0.9), exploration factor (0.05, 0.1), and decay factor (0.8, 0.9) is performed. The discount rate is fixed at 0.7. The discount rate has a direct impact on the final Q value, thus varying the discount rate would make the comparison challenging.

|  |  |
| --- | --- |
|  |  |
| Figure 9. Grid Search on 8 combinations | Table 11. Average reward/steps in last 500 episodes over 10 runs |

Figure 9 shows the average reward per step of 8 combination of parameters. It is not easy to make judgement based on it even with the descriptive statistics. Then we started to explore other metrics. The difference to the reward from the optimal path is considered. However, the optimal path does necessarily achieve the maximum reward in early episodes. Thus not informative in early episodes. Then, we explore the average reward/steps in the last few episodes. We can see that the convergence start from around 500 episodes. Therefore, we run the model 10 times and calculated the average reward/steps in the last 500 episodes over 10 runs as shown in Table 11. It revealed that arrive at the highest reward/steps of 53.98 in the last 500 episodes over the 10 run. And this combination ranked top 4 out of 10 times in Figure 10.

|  |
| --- |
|  |
| Figure 10. Ranking of reward/steps from 8 combinations over 10 runs |

We used the average over 10 runs to help us to determine the optimal parameter combination above. However, the optimal combination above does not guarantee the highest reward/steps in every run. Figure 10 illustrated the ranking of reward/steps of each combination in each run. No one combination ranked top in every run. It shows the randomness of how agent explores the environment, which is one of the fundamental of reinforcement learning and is the the way agent to learn. Despite the randomness, Q-learning is guaranteed to converge given enough exploration and iteration. (Sutton and Barto, 2018)

Future study: run all sections above multiple times to get more stable results

* 1. **Monte Carlo**

We also implement Monte Carlo learning and compare its performance with Q-learning. In contrast with Q-learning, Monte Carlo does not assume prior knowledge of the environment. (Sutton and Barto, 2018) Monte Carlo learning takes the combined the quality of each step towards reaching an end goal and requires that, in order to assess the quality of any step, we must wait and see the outcome of the whole combination.

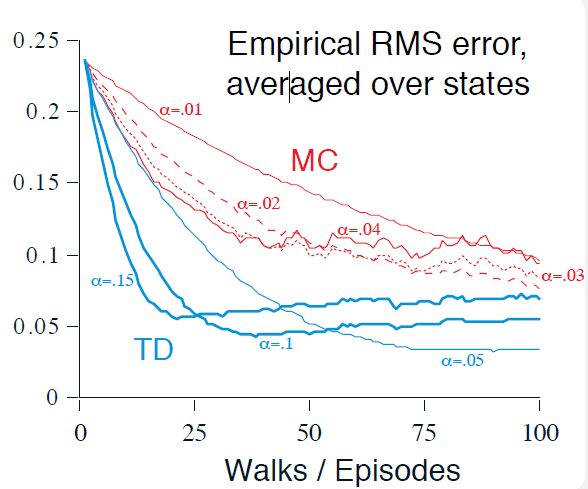
2.6.1. Q update

2.6.2 G value

In particular, suppose we wish to estimate v π (s), the value of a state s under policy π, given a set of episodes obtained by following π and passing through s. Each occurrence of state s in an episode is called a visit to s. Of course, s may be visited multiple times in the same episode; let us call the ﬁrst time it is visited in an episode the ﬁrst visit to s. The ﬁrst-visit MC method estimates v π (s) as the average of the returns following ﬁrst visits to s, whereas the every-visit MC method averages the returns following all visits to s. These two Monte Carlo (MC) methods are very similar but have slightly diﬀerent theoretical properties. First-visit MC has been most widely studied, dating back to the 1940s, and is the one we focus on in this chapter. Every-visit MC extends more naturally to function approximation and eligibility traces, as discussed in Chapters 9 and 12. First-visit MC is shown in procedural form in the box.

Double Q Learning

Can use Sum or Max or Average



An example of MC and TD comparison on page 125

1. **Essay Question** 
   1. **Q-learning and error correction models in psychology**
      1. Classical conditioning & Blocking
      2. Error correction models
      3. Instrumental conditioning
   2. **Error correction models and reinforcement learning**
      1. Essense of reinforcement learning

Selection & association (search & memory)

Law of Effect (try and error)

* + 1. From error correction models to Temporal Difference model

Reference:

[1] Beleznay, F., Grobler, T. and Szepesvari, C., 1999. Comparing value-function estimation algorithms in undiscounted problems. *Technical Report TR-99-02, MindMaker Ltd*.

[2] Even-Dar, E. and Mansour, Y., 2003. Learning rates for Q-learning. *Journal of Machine Learning Research*, *5*(Dec), pp.1-25.

[3] Sutton, R.S. and Barto, A.G., 2018. *Reinforcement learning: An introduction*. MIT press.

[4] Watkins, C.J.C.H., 1989. *Learning from delayed rewards* (Doctoral dissertation, King's College, Cambridge).

<https://www.youtube.com/watch?v=Q99bEPStnxk&list=PL-9x0_FO_lglnlYextpvu39E7vWzHhtNO&index=5>