INM 426 – Software Agents

**Lego Home Finding with Q-learning**

Harry Li, Xin Li

# Introduction

Q-learning is one of the early breakthroughs in reinforcement learning developed by Chris Watkins in 1989 when he was a graduate student at Cambridge University. The algorithm is efficient and simple. It is still one of the most popular algorithms today. This report shows the implementation of Q-learning in a navigation task. The impact of different parameters (e.g. learning rate, discount rate, exploration factor, decay) are explored. A grid search is performed to study the optimal combination. Furthermore, Double Q-learning, invented to overcome overestimation of Q-learning, is implemented to understand the differences to Q-learning. In addition, the parallelism between Q-learning and psychological learning theories is discussed.

# Basic

## Domain and Task

The domain is the central part of 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.

|  |  |
| --- | --- |
| https://lh6.googleusercontent.com/9B1EbAu4kpB8Rxkv-xmpqXUnZu3lhjkPrqn_SwueRd0ayEAdam7D4M5RW4GDBSK8kPpf15b28ywNZhAaV7xEGfYb4DU53tnc4JZms6Gk24vCWvmTGuwGKyyIxGJI-EGWh2BBzhfc | https://lh3.googleusercontent.com/asbKjLmPk8e4h8HoE-DrdfkXfvPiPzSg36lX5YIhylegD7OW2xQpcpb_VMW9dQzu11pp1AWkFK0rkEANfAHph-LnFdgzoABfvqOHDM42slrpNgDe0panGKsMuJjjH46N4SP1xhP4 |
| Figure 1. London Underground Lego map | Figure 2. Graphic representation of the domain |

## State Transition Function

The state provides the agent basis to make an action. It has the Markov property that every state includes all the information in the past and enables the agent to make future interaction with the environment. 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 | | | |

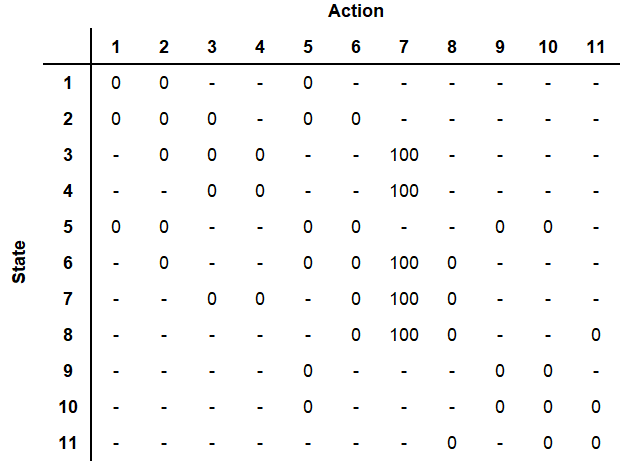
## Reward Function

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. 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)

## Policy

Policy () is a mapping from states to each possible action. Policy is what the agent ultimately learns through iteratively evaluate and improve state-value function or action-value function (this process is called Generalised Policy Iteration). One way to improve the policy is to take a greedy approach in respect of value function follow Bellman optimality equation. However, always take greedy approach would make agent leave many options unexplored. Therefore, to balance between exploration and exploitation, an exploration factor () is introduced. ε-greedy policy is one of the simplest ideas to balance between exploration and exploitation, but effective. With the probability of , the agent takes a random move instead of greedily. Furthermore, a decay factor for is also employed in this report. This allows the agent to explore more at the beginning and gradually less towards the end. This assumes that agent explored enough in early episodes and claimed to be efficient. (Sutton and Barto, 2018)

## Graphic Representation and R-matrix

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

The graphic representation of the domain is shown in Figure 2 above.

## Parameters

Q-learning is an off-policy TD control algorithm which is presented below. The new is equal to the old plus immediate reward after taking an action and maximum one-step ahead action-value function. is learning rate and is discount rate. (Sutton and Barto, 2018)

### Learning Rate

Bootstrapping is one property of Q-learning and TD model in general. It means the current update is based on previous estimates. Learning rate () is used to control how much of the agent takes into account the new update given it is an estimation. We might not want to have a full update but instead steady steps toward the right direction. When close to 0, the agent only takes a small step towards the updated value. When approaching 1, the agent update from the updated value. (Even-Dar and Mansour, 2003, Sutton and Barto, 2018) The learning rate is initially set as 0.9 for simplicity as Even-Dar and Mansour (2003) suggested 0.85 is a good start. Beleznay, Grobler and Szepesvari (1999) suggested to use fixed learning rate for faster convergence, thus the learning rate is not decayed.

### Discount Rate

Discount rate () is used to represents the uncertainty of the estimates and it determines the present value of it. It’s also a mathematical convenience to turn an infinite horizon problem to a finite one. 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) A discount rate of 0.8 is used as initial setting.

### Exploration and Decay Factor

As noted in section 1.4 that ε-greedy policy with decay factor is implemented in this report. The larger the exploration factor and the smaller the decay factor, the more agent would explore instead of acting greedily towards the short-term maximum value. We initially set exploration factor to be 0.9 and decay to be 0.999 in a view to let the agent explore more at the early episodes. The table below summarised the initial configuration of parameters.

|  |  |  |  |
| --- | --- | --- | --- |
| **Summary of initial parameters configuration** | | | |
| Learning Rate () | 0.9 | Exploration Factor () | 0.9 |
| Discount Rate () | 0.8 | Decay factor () | 0.999 |
| Table 2. Summary of initial parameters configuration | | | |

## Q-matrix Update

To demonstrate Q-learning, 2 episodes of Q-matrix asynchronous update is illustrated below with the parameters stated above (). In addition, pseudocode of Q-learning algorithm is presented in the box below (Sutton and Barto, 2018).

|  |
| --- |
| **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 ε-greedy policy  Take action , observe      until is terminal |

|  |  |
| --- | --- |
| **Episode 1: Station 6 to 7** | |
| Step 1: initial Q-matrix | https://lh4.googleusercontent.com/Vkae9rccj5pTS7bR_AbtHIoydRd45unas4th_q38XnrzWP4DNJaGZQiOiGX47P1AIgQcpIIsWx48ImgUaTKMausHqcIKBJ-HRkAEjblxTB_uf8QtuSyeQu0WRyob5nK53_A46dIM |
| Step 2: choose to go to Station 7  Station 7 becomes the current state. Since Station 7 is the terminal state, this episode ends here. | https://lh5.googleusercontent.com/-DOJBhw_4QlEuSVQ4rNA2V1rTF7mmJx-lrKepuAgdrwrg9lpb4KPyuh-5U_KK22FYPsuIcweGjV-boAg6_IUMkAFQMvv6fqiWM_c7OfBEKue4VqWDGchEM01rvWKPGZ6azq0ZZpx |
|  | |
| **Episode 1: Station 2 to 7** | |
| Step 1: go to Station 6 follow policy  Station 6 becomes the current state. | https://lh3.googleusercontent.com/5pKDvd_Z-CcYr_YpHWf0PWGqc7hHLSmMsskvoog_cuIoW1xdOZXDkesrfTIMeE8umUl87wuNA3sTxUc5VcQz-euM1F8jjIs08wfuN7u3i9lGlGdqQfLPxx5d44g4eVfUElvUYgQl |
| Step 2: go to Station 7 follow policy  Station 7 becomes the current state. Since Station 7 is the terminal state, this episode ends here. | https://lh4.googleusercontent.com/xEwByaC1-7Nf_2qmf6L6vp5suxUIShQ20SckcX9snNWRRzrUn-9QDCZ14sdre3clud60sj9jNP0a9t_-tH5Edb0kT0g17PvQYD6Di_b5byTGJ65RpShypkQv3Gf3c0DqrtN9Fm4D |

## Performance vs Episodes

Section above illustrated two episodes of update in Q-matrix with and . To fully show the performance, we run the algorithm for 1000 episodes with parameters in Table 2. There are many ways to measure the performance. To make it comparable among all configurations. We evaluate the Q-matrix after each episode by set the agent navigate from Station 9 to Station 7 (destination) using the Q-matrix just updated. The idea is that if the Q-matrix is completely updated, the agent should able to navigate with the optimal path. The total number of steps, total reward, average reward per step, and mean of average reward per step in the last episodes are recorded. Only the relevant measurement supporting the analysis are presented in the following sections of this report. And all the Q-matrix in this report is normalised by dividing the largest value () for comparability.

|  |  |
| --- | --- |
| https://lh6.googleusercontent.com/_mLHzblZiUg7xT65-t9iFNPwYl9Lh9ZMvpjyIS56wIqTut-JvU-vcCC1ZLmf2inFXprAoSwGas1-yjmu0KMMQBi39blQEhl44RZg4p79lTX0ZyYeP7rChTg6ddJIRG3GnRJ2Rmhh | https://lh4.googleusercontent.com/dDKAnz8yK1uiCUG3_k1L42-nrVT68vMEJp7lWCP2RejNhdLUKwtDCv15-Ijw0qxOZ0D1LWYLW7y7NIIoi2Isu3SyjJ0Y8ENobp9FVSchxIaNSzeGUkEcIeaVtwIlm01uqcbNPsgH |
| Figure 3. Reward/Steps & Steps in 1000 episodes | Table 3. Statistics of Reward/Steps & Steps |

Figure 3 shows the performance evaluation in 1000 episodes together with descriptive statistics in Table 3. The convergence reached around 320th episode, taking 4 steps from Station 9 to 7. The reward/steps is around 60 after convergence. The table below shows the Q-matrix by 1000 episodes and the optimal path from all stations to Station 7.

|  |  |
| --- | --- |
| **Optimal paths to Station 7** | **Normalised Q-matrix by 1000 episodes** |
| 1 to 7 (steps: 4): 1→5→6→7  2 to 7 (steps: 3): 2→6→7  3 to 7 (steps: 2): 3→7  4 to 7 (steps: 2): 4→7  5 to 7 (steps: 3): 5→6→7  6 to 7 (steps: 2): 6→7  7 to 7 (steps: 1): 7→7  8 to 7 (steps: 2): 8→7  9 to 7 (steps: 4): 9→5→6→7  10 to 7 (steps: 4): 10→11→8→7  11 to 7 (steps: 3): 11→8→7 | https://lh4.googleusercontent.com/GdP0c_HsnxxJUK7ju1ruQ6XCLH5v-BakMsfhsTn8g3ppWkVVyvGZ_bpy673AuUUApMgACCAvAbwV4Ttu80y0JSCJSpceXCCSQE5SRTuXKxTjsVSgHRtr4J6c4rc2HMSVMqOiUMt2 |
| Table 4. Normalised Q-matrix by 1000 episodes & optimal paths to destination | |

# Advanced

## Search of Learning Rate

To understand the impact of the learning rate, a wide range of values are explored: 0.01, 0.1, 0.5, 0.9, 1. Empirically, large learning rate leads faster convergence. However, too large learning rate without exploration may lead to a suboptimal policy (Beleznay, Grobler and Szepesvari, 1999, Even-Dar and Mansour, 2003). Thus we expect a faster convergence when the learning rate is large. *[Hypothesis 1: larger learning rate leads to faster convergence]*

|  |  |
| --- | --- |
| https://lh5.googleusercontent.com/ZlbTdnqHtnaPZCKzmiVkSf_ZQk31KZNpyueFhb1m2uYZMHstXz-iQ_LSWMC_AOLRbe38LGhtdMrPDAN21-RwJth_JP8lWVDbWtTy6j_pNoZd3v7g0ljSFPkOGBb2qKz5u6XYi4Pv | https://lh4.googleusercontent.com/21G5grGpD0OfnJjm4QfR6ltha2ECIx17PMCTtFxkjxEXOf6Q8BXaOB4arAw9dl_TeFSeUbBdRtlrm8Zp33fd2Dnzegc-cpSJzoz4pBZBKCJanIHsJUwjmDMNRwVSYn24vbmKy138 |
| Figure 4. Average reward per step in different α | Table 5. Statistics of reward/steps in different α |

Figure 4 shows that the convergence reached by 1000 episodes when is 0.5, 0.9 or 1.0. Table 5 also shows that the maximum reward/steps is around 60 for those 3 values. When is 0.01 or 0.1, the convergence is not arrived by 1000 episodes. In terms of the speed of convergence, the large learning rate does converge faster. Therefore, it is consistent with our hypothesis and previous studies.

## Search of Discount Rate

The following discount rates are explored: 0.01, 0.1, 0.5, 0.8, 1. High discount rate (i.e. large value) means more future rewards is reflected, thus the agent is far-sighted. Therefore, we expect faster convergence can be brought by using a high discount rate. *[Hypothesis 2: larger discount rate leads to faster convergence]*

|  |  |
| --- | --- |
| https://lh3.googleusercontent.com/zZJm4sOYcSL-fNUI9MphvfID7VRPuvakL9t-QoSYYbxMqG4C4J2OKDcJF1xQ4ZTV1CXQbb2e5bOwepwrh9MKT-n_gXr8rLdOEmeGgpuh2EddQ7GsJuZSSyWpwJykpg1p2r1H2lV- | https://lh5.googleusercontent.com/UgkfTVvg1OACu-GAakBIAmMh8D3TWbdTUYYclQ7AkOdUaVqfvNQSDHE4K-KYB8OaxOrQJ9cXC0ftcxigfJjHcog_kJeQu_rUL28AKaZJPmyyEGBjAn79imBPEDrRYWwHgGRb3KOv |
| Figure 5. Average reward per step in different γ | Figure 6. Total steps in different γ |

Figure 5 shows opposite to our hypothesis. Agent converge earlier with smaller discount rate. And interestingly, average reward per step is very unstable when =1. It is not even converged by 1000 episodes. This is consistent with findings from Even-Dar and Mansour (2003) that Q value becomes unstable when discount rate is high. When =1, a complete reward estimation backup gives the agent not enough incentive to find out the optimal path. Since more options than actually have would give the agent the same maximum reward. Figure 6 also illustrated the unstableness that there are still large steps after 60 episodes when is1.0.

## Search of Exploration Factor

Exploration is one of the few conditions for Q-learning to learn optimal policy (Even-Dar and Mansour, 2003). of 0.001, 0.01, 0.1, 0.5, 0.9 are explored. We expect high would lead to a slow convergence since the agent spend longer time to explore. *[Hypothesis 3: larger exploration factor leads to slower convergence]*

|  |  |
| --- | --- |
| https://lh5.googleusercontent.com/RZrXoAoYxNeq-0-nQz36NeIywrQyOZ2s_oUmnqaTGPPJhgA2JivWAc8z3ehw91580j7yoVG_A-rmWMx-wJK-rckvxQicOo9dr4dZjtmm9tAMLvoJzDbW7ISWw22JatrgmU5h_Dkz | https://lh3.googleusercontent.com/bJjMXwp7EdMHOYMZHiGuwuvV143davd-6J4Ewfr5K4XCYDURDJ6KQUynK1cpFWNU8FPhbBQ0UAbYIeDFSMFmHHFKxWtSCz0KzHwy5LeB6B_AemFcwFV03ZRALZ-memnOP3OTgIPW |
| Figure 7. Average reward per step in different ε | Table 6. Q-matrix by 1000 episodes with ε=0.001 |

Figure 7 shows all 5 cases are converged by 1000 episodes. And seems only have marginal impact on the speed of convergence. However, only looking at performance evaluation in average reward per step is misleading in this case. Table 6 shows the normalised Q-matrix by 1000 episodes when is 0.001. It appears the matrix is sufficient to guide the agent navigate from Station 9 to 7 in the optimal path. But, many actions at each state are left without explored. This is a good example to show the effect when is too small and the necessity of exploration. In our case, the Q-matrix got completely updated when is 0.5 or 0.9.

## Search of Decay Factor

The following values of decay factor are explored: 0.5, 0.7, 0.9, 0.99, 0.999. We expect a slow convergence when decay factor is large, which the agent explores more. This might be an over simplified hypothesis, but we would expect this trend. *[Hypothesis 4: larger decay factor leads to slower convergence]*

|  |  |
| --- | --- |
| https://lh3.googleusercontent.com/9FHgWY4_IxF3mqQTXh5RlXsJijcUfc1640SBxXzrJPskededN06KeIDQek2SiVMAlShicAgIFWWYFdfwP97AOCtS9bt_GWZKInUFaMzflZav5bK_MAHPN8pRMrIOcgXb_32zn8r- | https://lh3.googleusercontent.com/c0u4VLnwXfdLfGdt2HuizEfzQTHXjjFl0TV1PLGp6qAoyGBRCzRzcveJj5FQkUautoLQdlwXSFmNWz7w63xLs5oR7ceibn76dS3yLmwrKEs1dp3Tcn1MMXs-qfvALl_Q8HkJxTj8 |
| Figure 8. Average reward per step in different decay | Table 7. Statistics of total steps in different decay |

Figure 8 doesn’t show any clear trend in terms of the speed of convergence with different decay factor, and little conclusion can be draw from it. By contrast, Table 7 reveals some signs of impact from the different decay factors. The decay factor is negatively correlated with average total steps. For example, average 4.4 steps when decay is 0.999 but 5.9 steps when decay is 0.5. Also, the 3rd quartile total steps is 4 steps only when decay is 0.999. It means slower decay (i.e. more exploration) is beneficial. When checking Q-matrix with different decay value, it shows that the Q-matrix is completely updated when decay value is 0.99 or 0.999. This is another good example to prove the value of exploration especially in early episodes.

## Search for an Optimal Combination

From error correction models to TD model

In Rescorla-Wagner model, if we have a compound CS A and X. The animal may already experienced stimulus A, and X might be new. Let V denote the associative strengths of stimulus. Suppose a trial involve a compound CS AX followed by US (denote as Y). R is the level of associative strength that US Y can support. Then the change in associate strengths are expressed as:

,

and are step-size parameters depended on CS and US. The formulas state that and will keep increase until the compound reach the same level of . To gradually transit to TD model, we treat the the conditioning process above as the predicting the magnitude of US. And states need to be introduced to convert Rescorla-Wagner model to a real-time model. Assume a state is described a vector of features . If the d-dimensional vector of associative strengths is , the aggregated associative strength is: . To update on trail : . is step-size parameter and is prediction error: . is the prediction of magnitude of US on trial . (Sutton and Barto, 2018)

To fully arrive at TD model, we now change to denote time step instead of a complete trial as above. And a short-term memory vector, eligibility trace is also introduced to only update eligible states or actions. Now, the associative strength update becomes: . The full state vector is replaced with eligibility trace vector. And prediction error now is: , where is the discount factor, is the immediate reward after take an action, and and are aggregated associative strength at and which can be seen as value function (action value function in the case of Q-learning). (Sutton and Barto, 2018)