Logo, company name

Description automatically generated

October 17, 2021

Goh Kheng Xi, Jevan

a0199806l

NUS Engineering

ESP3201

Search Mini-Assignment

Contents

[1. Introduction 2](#_Toc85395299)

[1.1 Aim 2](#_Toc85395300)

[2. Value iteration 2](#_Toc85395301)

[2.1 MDP 2](#_Toc85395302)

[2.2 Algorithm 3](#_Toc85395303)

[2.3 Testing and results 4](#_Toc85395304)

[2.4 Varying 5](#_Toc85395305)

[3. Q-learning 6](#_Toc85395306)

[3.1 Algorithm 6](#_Toc85395307)

[3.2 Testing and results 6](#_Toc85395308)

[3.3 Varying 8](#_Toc85395309)

[3.4 Varying 8](#_Toc85395310)

[3.5 Varying 9](#_Toc85395311)

[4. Conclusion 9](#_Toc85395312)

[5. References 10](#_Toc85395313)

# 1. Introduction

Reinforcement learning is a branch of machine learning methods that trains an agent through a reward and punishment mechanism as the agent interacts with the state space [1]. A reinforcement learning agent consists of the following components:

1. Policy (π): The policy determines the action for the agent to take at a given state
2. Rewards function (r): A numerical score that seeks to represents the objectives and subobjectives of the given tasks
3. Value function (v): Represents the expected future cumulative rewards of a given state and policy.

## Aim

The task given is to implement a value iteration and a Q learning agent in a similar environment shown in Figure 1 below where the agent starts from the bottom left corner and tries to get to the goal at the bottom right corner.

A picture containing table

Description automatically generated

Figure . Environment

If the agent reaches the cliff, it receives a reward of -100 and the run terminates and if it reaches the goal at the bottom right, it incurs a reward of 10 and the run terminates. Other transitions will incur a reward of -1.

# 2. Value iteration

Value iteration is a reinforcement learning algorithm that updates the estimated value function of every state iteratively based on a Markov Decision Process (MDP) [2].

## 2.1 MDP

MDP is a model representation of the given environment that defines the probability of every possible outcome after performing an action in a given state and its corresponding reward [2] [3].

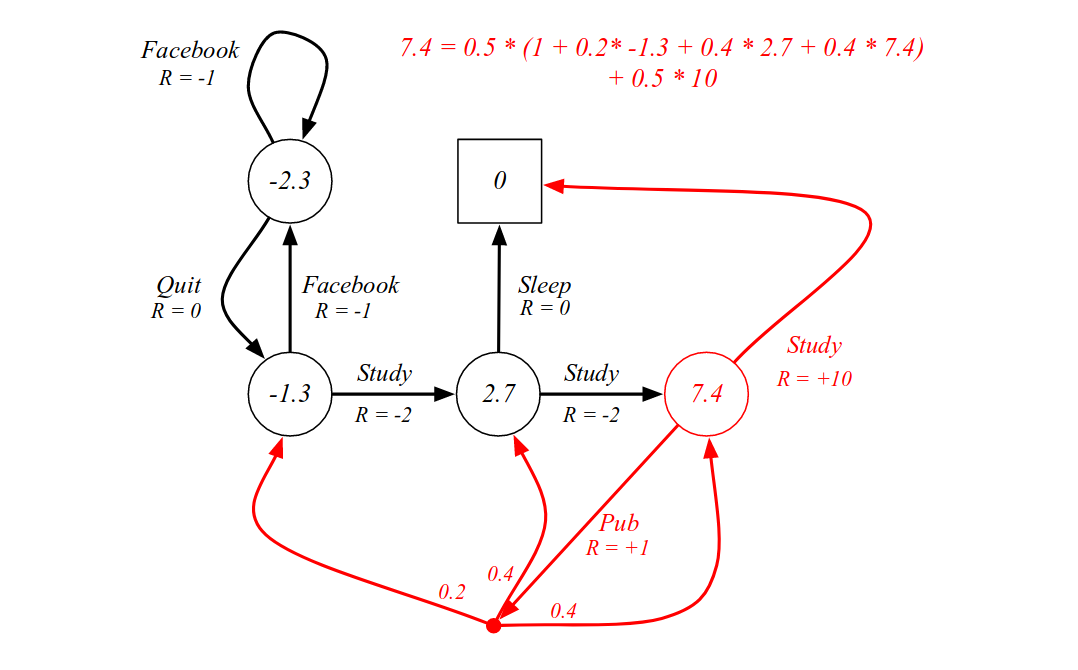


Figure . Example of a MDP

R = -1

R = +1

R = -2

Figure 2 depicts an example of a MDP where the current state is at the red circle of 7.4 value. In the example, there are 2 possible actions of the current state which are to study and to go Pub.

Taking a closer look at the action of going to the pub given the current state, there are three possible outcomes of the actions known as the transition states of the action. The probabilities of ending up in each transition state and their corresponding rewards are given which are required in the value iteration algorithm.

## 2.2 Algorithm

In value iteration, the value function of each state in the environment is initialised arbitrarily which is 0 in the codes implemented for this task. Next, the Q-value, which is the estimated reward of taking an action, of each state-action pair is calculated using equation 1 below.

(equation 1)

r represents the additional reward of ending up in the transition state, s’, after performing action a from current state, s. However, the total reward of ending up in s’ after performing a from s have to take into consideration the expected cumulative future rewards of s’ which is represented by the value function, , of the transition state. The discount factor, , is a ratio that lies between 0 and 1 that determines the importance of future rewards versus the current rewards in the calculation of the Q-value. The higher the discount factor, the greater the importance of future rewards. The weightage of the cumulative rewards of the transition state given a, s is determined by the probability, of ending up in s’ given s, a. Hence, the Q-value of a state-action pair is the sum of each transition states’ cumulative reward multiplied by their respective probabilities.

When updating the value function of a state, a greedy approach is implemented where the highest Q-value among the possible actions of the defined state is the new value function of the state. In mathematical notation, the equation is as follows:

(equation 2)

This update algorithm can be illustrated using the MDP example in Figure 2. Taking to be 0.9, at the current state in Figure 2,

Since

The iteration stops when the agent reaches a terminal state or when all the states are updated and the algorithm continues forever until a terminating condition is met. Since the update of the value function diminishes per iteration due to the discount factor (hence, a reason for to be between 0 and 1, else the value function will not converge), a common terminating factor is the minimum change in the updated value function.

## 2.3 Testing and results

For testing the metrics used to observe the convergence of the algorithm is the sum of all the states’ absolute value function,. This summation value is plotted against 10 iterations to observe the convergence of the algorithm where the change in this value diminishes. The initial value function of all the states is 0.

|  |  |
| --- | --- |
| Iteration |  |
| 1 | 223.0 |
| 2 | 240.8 |
| 3 | 257.5 |
| 4 | 268.21 |
| 5 | 269.89 |
| 6 | 263.62 |
| 7 | 259.56 |
| 8 | 259.56 |
| 9 | 259.56 |
| 10 | 259.56 |

Figure . Graph (left) and table (right) of against iterations

A screen shot of a computer

Description automatically generated with low confidenceFigure 3 shows the sum of all the value functions over 10 iterations. It can be observed that the change in the value diminishes very slightly over the number of iterations and eventually converged to a value of 259.56 after 7 iterations.

Figure . v(s) of every state after 10 iterations

Figure . Q(s, a) of every state-action pairsCalendar

Description automatically generated

Figure 4 shows the final v(s) of each state after 10 iterations and Figure 5 shows the final Q(s, a) of each state-action pairs.

## 2.4 Varying

The value was varied and the change in the convergence of the v(s) values were plotted and studied in Figure 5.

Figure . Graph of against iterations with varying

It can be observed from Figure 6 that the convergence becomes faster diminishingly as the discount factor decreases. However, as the increases, the agent tends to better maximise the cumulative future rewards.

The discount factor is then set to -1 and the algorithm wan ran. Figure 7 shows the results of the algorithm when the discount factor is set to -1. In addition, the discount factor is set to 2 and the program ran infinitely. From the results in Figure 6 and the infinite looping when discount factor is 2, it can be concluded that the algorithm does not converge when the discount factor is outside the range of 0 and 1.

Figure . Results when

# 3. Q-learning

Q-Learning is a value-based temporal difference (TD) reinforcement learning that learn from the experiences in the previous episodes [4]. Unlike value iteration, Q-learning does not require a MDP model of the environment [4] [5]. It is an off-policy TD algorithm which means that it does not trains from the experience in the current episodes [5].

## 3.1 Algorithm

The Q-values of each state-action paired are initialised to 0 initially and stored in a Q-table. From the starting state and for each of the subsequent states, the agent chooses to explore or to exploit based on a predefined epsilon value which determines the probability that the agent will explore. If the agent chooses to explore, it will pick a random action to perform an action the list of available actions in the current state. If the agent chooses to exploit, it will choose an action from the list of actions with the maximum Q-value to perform.

The agent will receive a reward, r, and transits to the next state, s’. The Q-value of the state-action pair is updated through equation 3 below

where is the discount factor with similar purpose as the discount factor in the value iteration algorithm. is the value function of the transition state which is the maximum Q(s’, a) of the transition state. is the learning rate that determines the amount of update to the Q(s, a) value; the higher the learning rate, the faster the convergence but the less accurate the convergence will be. The algorithm terminates once it reaches a terminal state which in the given problem statement, is the goal state and the cliff states, and this completes one episode.

## 3.2 Testing and results

For the testing the values for the parameters are set to these values: and every value in the Q-table is initialised to 0 initially. Next, the metrics used to observe the convergence for the Q-learning algorithm is the sum of absolute Q(s, a) of each state action pairs as the Q(s, a) values are the value that is being updated directly. This summation value is plotted against 500 episodes and the results are shown in Figure 8 below.

Figure . Graph of against episodes

From Figure 8, it shows that the algorithm converges after approximately 200 episodes. Figure 9 shows the final Q-values of each state-action pairs after 500 episodes while Figure 10 shows the value function and optimal policy for each state after 500 episodes.

A computer screen capture

Description automatically generated with medium confidence

Figure . Q(s, a) of every state-action pairs after 500 episodes

A screenshot of a computer

Description automatically generated with medium confidence

Figure . v(s) = max(Q(s, a)) of every state after 500 episodes

## 3.3 Varying

is a numerical value representing the probability that the agent will explore unexplored states in the current state and lies between 0 and 1.

Figure . Graph of ∑|Q(s,a)| against episodes with varying

Figure 11 shows the results of the varying value and it can be observed that as the value increases, the faster the convergence. This is due to the higher value allows the agent greater opportunities to explore new actions and hence, it converges to a solution faster. However, if the value is too high, this might cause the agent to act randomly and not learn anything Likewise, if the value is too low, the agent will mostly be performing the greedy actions and might not learn other solutions that might be more rewarding.

## 3.4 Varying

, also known as the learning rate, determines the amount of new Q-Value to be updated to the current Q-value; the larger the , the larger the update. In this section, the learning rate is varied while keeping the other parameters constant. The results are shown in Figure 12 below.

Figure . Graph of ∑|Q(s,a)| against episodes with varying

It can be observed from Figure 12 that the larger the , the faster the convergence as the agent is performing larger updates to the Q-values. However, the larger the the less accurate the agent will be as the agent might “miss” the optimal solution if the learning rate is too high.

The value must be between 0 and 1, else the algorithm would not converge. This is evident when the code was ran with an value of 2 and -1. The code ran infinitely as the magnitude of the Q-values kept increasing.

## 3.5 Varying

, also known as the discount factor, determines the weightage of future rewards versus the immediate rewards; the higher the , the greater the importance of future rewards. In this section, was varied while keeping the other parameters constant.

Figure . Graph of ∑|Q(s,a)| against episodes with varying

From Figure 13, it can be observed that the smaller the , the faster the convergence as the importance of future rewards increases. However, as the increases the agent will tend to better maximise the cumulative rewards.

The discount factor should be set in between 0 and 1, else the algorithm will diverge which is evident in the infinite loop when the code was ran with values of -1 and 2.

# 4. Conclusion

Different machine learning methods are suitable for different task. If the MDP model of the environment is not given, value iteration might not be appropriate. Although Q-Learning is model-free, it does have its own limitations as well. Q-learning only works in an environment with a finite number of states and actions which is in contrast to real life problems where the number of states and actions can be infinite.

For simpler problems such as this assignment, simple reward structure and hyperparameters have proven to be relatively successful. However, for more complex problems, more complex configurations of reward structure and hyperparameters can be explored. For example, the value in Q-Learning can be configured to decrease over time as the agent should be exploring the most at the start of the algorithm. Nevertheless, it is without a doubt that whichever reinforcement is applied, the hyperparameters must be carefully tuned in order to achieve a good performance.

# 5. References

|  |  |
| --- | --- |
| [1] | B. Osiński and Konrad Budek, “What is reinforcement learning? The complete guide,” Deepsense, 5 July 2018. [Online]. Available: https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/. [Accessed 17 October 2021]. |
| [2] | P. Abbeel, “Markov Decision Processes,” 2012. [Online]. Available: https://people.eecs.berkeley.edu/~pabbeel/cs287-fa12/slides/mdps-exact-methods.pdf. [Accessed 17 October 2021]. |
| [3] | R. Jagtap, “Understanding Markov Decision Process (MDP),” Towards Data Science, 28 September 2020. [Online]. Available: https://towardsdatascience.com/understanding-the-markov-decision-process-mdp-8f838510f150. [Accessed 17 October 2021]. |
| [4] | A. Violante, “Simple Reinforcement Learning: Q-learning,” Towards Data Science, 19 March 2019. [Online]. Available: https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56. [Accessed 17 October 2021]. |
| [5] | C. Shyalika, “A Beginners Guide to Q-Learning,” Towards Data Science, 15 November 2019. [Online]. Available: https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c. [Accessed 17 October 2021]. |