OCTOBER 10, 2021

ESP3201

SEARCH MINI-ASSIGNMENT

GOH KHENG XI, JEVAN
A0199806L
NUS Engineering

Contents

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. A reinforcement learning agent consists of the following components:

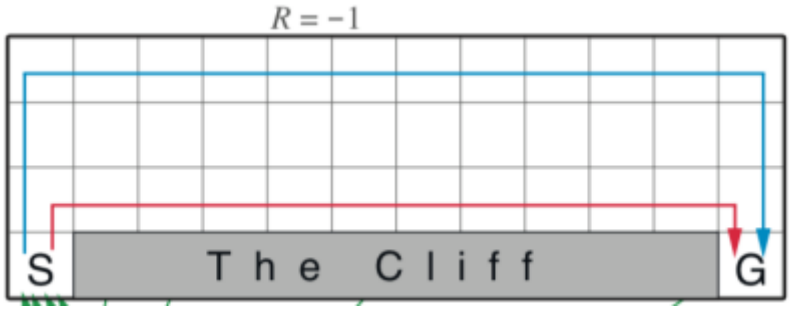
1. Policy (π): The policy determines the action for the agent to take at a given state

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.

1.1 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.



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).

2.1 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.

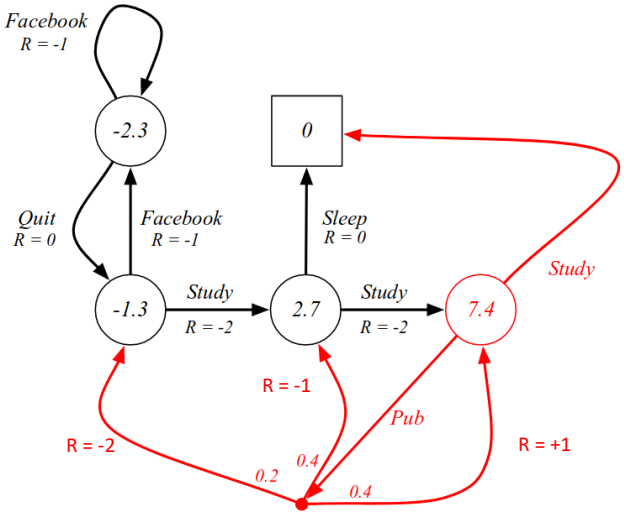


Figure 2 depicts an example of a MDP where the current state is at the red circle of 7.4 value. In the example, there are 2 possible actions of the current state which are to study and to go Pub.

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.

$$Q(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma v_k(s')] \qquad \text{(equation 1)}$$

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, v, of the transition state. The discount factor, γ, 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, $p(s',r|s,a)$, 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:

$$v_{k+1}(s) = \max_a Q(s,a) \qquad \text{(equation 2)}$$

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

$$Q(s, sleep) = 1 \cdot (10 + 0.9 \cdot 0) = 10$$

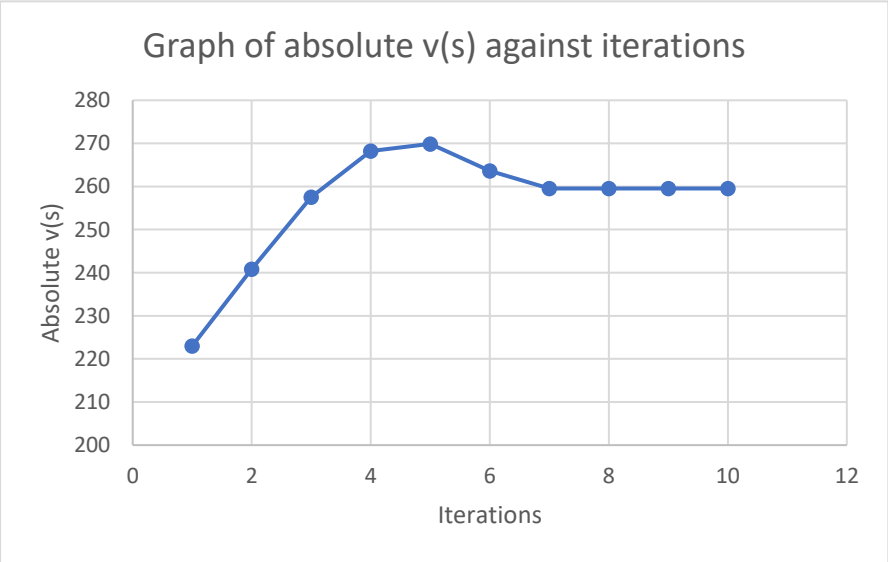
$$Q(s, pub) = 0.4 \cdot (1 + 0.9 \cdot 7.4) + 0.4 \cdot (-1 + 0.9 \cdot 2.7) + 0.2 \cdot (-2 + 0.9 \cdot -1.3) \approx 3$$

Since $Q(s, sleep) = 10 > Q(s, pub) = 3, \quad v_{k+1} = Q(s, sleep) = 10$

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 γ 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.

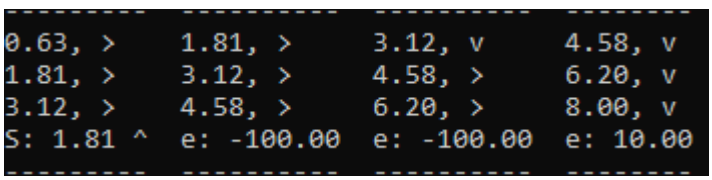
2.3 Testing and results

For testing the metrics used to observe the convergence of the algorithm is the sum of all the states' absolute value function, $\sum|v(s)|$. 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	$\sum\|v(s)\|$
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 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.

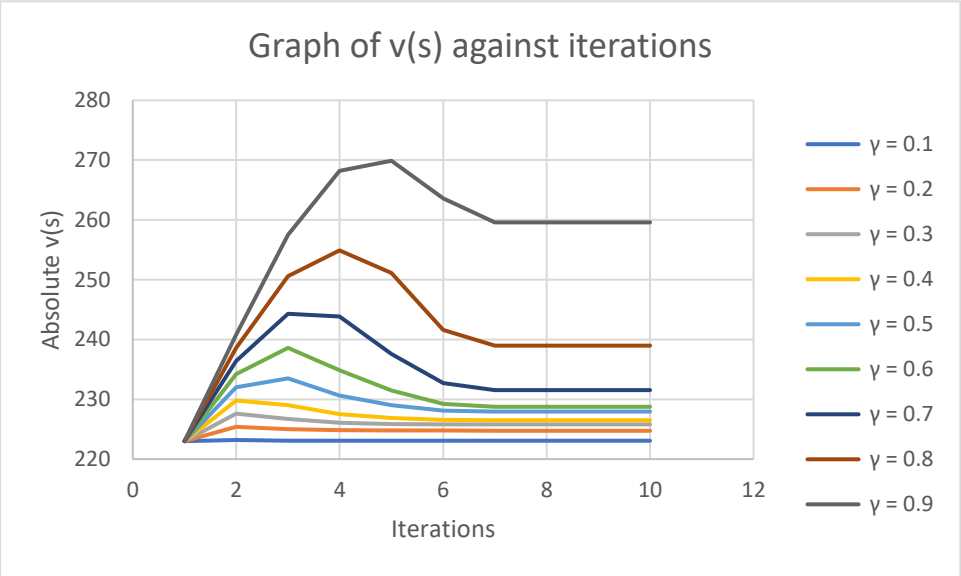


```
Q-VALUES AFTER 10 ITERATIONS
--------  ------  -------  --------  ----------  -------  -------  ----------  -------  -------  --------  -----
|         -0.43            |         0.63                |         1.81                |         3.12
|  -0.43          0.63  >  |  -0.43            1.81  >   |  0.63           3.12  >   |  1.81            3.12
|         \0.63/           |         \1.81/              |         \3.12/              |         \4.58/
-----     -----    -----     -----     -----     -----     -----     -----    -----     -----     -----     -----
|         -0.43            |         0.63                |         1.81                |         3.12
|  0.63           1.81  >  |  0.63             3.12  >   |  1.81           4.58  >   |  3.12            4.58
|         \1.81/           |         \3.12/              |         \4.58/              |         \6.20/
-----     -----    -----     -----     -----     -----     -----     -----    -----     -----     -----     -----
|         0.63             |         1.81                |         3.12                |         4.58
|  1.81           3.12  >  |  1.81             4.58  >   |  3.12           6.20  >   |  4.58            6.20
|         0.63             |         -91.00              |         -91.00              |         \8.00/
-----     -----    -----     -----     -----     -----     -----     -----    -----     -----     -----     -----
|         /1.81\           |                             |                             |
|  0.63   Start    -91.00  |         [ -100.00]          |         [ -100.00]          |         [ 10.00]
|         0.63             |                             |                             |
-----     -----    -----     -----     -----     -----     -----     -----    -----     -----     -----     -----
```

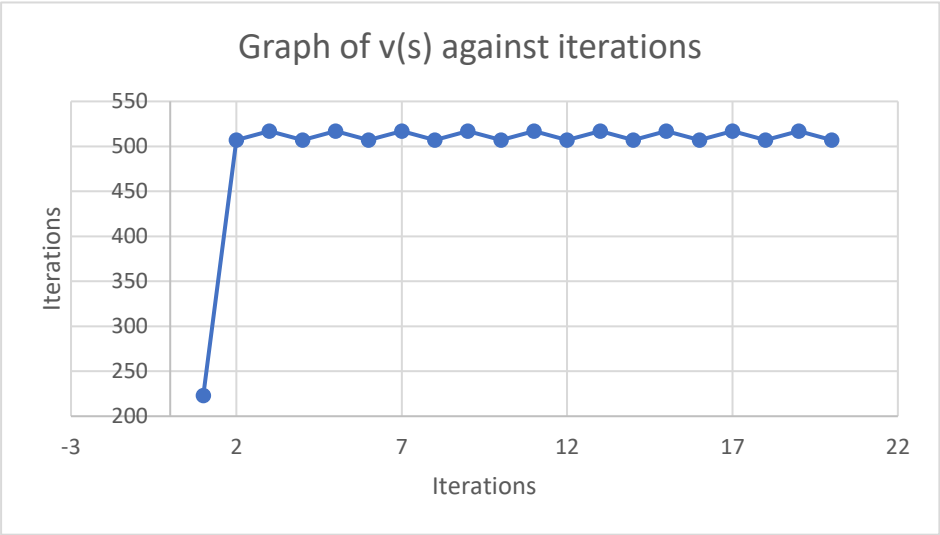
Figure 4 shows the final v(s) of each state after 10 iterations and Figure 5 shows the final Q(s, a) of each state-action pairs.

2.4 Varying γ

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



It can be observed from Figure 5 that the convergence becomes faster diminishingly as the discount factor decreases. However, the accuracy of the convergency becomes poorer as well with decreasing discount factor.



The discount factor is then set to -1 and the algorithm wan ran. Figure 6 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.

3. Q-learning

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

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 ϵ 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

$$Q(s,a)_{new} = Q(s,a)_{old} + \alpha[r + \gamma v(s') - Q(s,a)_{old}]$$

where γ is the discount factor with similar purpose as the discount factor in the value iteration algorithm. $v(s')$ is the value function of the transition state which is the maximum Q(s', a) of the transition state. α 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.

7. References

[1] Tutorials Point, “AI - Popular Search Algorithms,” Tutorials Point, [Online]. Available: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_popular_search_algorithms.htm. [Accessed 10 October 2021].

[2] S. J, “Analytics Vidhya,” Data Science Blogathon, 7 February 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/02/uninformed-search-algorithms-in-ai/. [Accessed 10 October 2021].

[3] Simplilearn, “The Ultimate Guide to Stacks And Queues Data Structures,” Simplilearn, 15 September 2021. [Online]. Available: https://www.simplilearn.com/tutorials/data-structure-tutorial/stacks-and-queues. [Accessed 10 October 2021].

[4] Geeks for geeks, “Priority Queue | Set 1 (Introduction),” Geeks for geeks, 28 June 2021. [Online]. Available: https://www.geeksforgeeks.org/priority-queue-set-1-introduction/. [Accessed 10 October 2021].

[5] B. Roy, “A-Star (A*) Search Algorithm,” Towards Data Science, 29 September 2019. [Online]. Available: https://towardsdatascience.com/a-star-a-search-algorithm-eb495fb156bb. [Accessed 10 October 2021].

[6] A. Patel, “Heuristics,” Standford Theory, [Online]. Available: http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html. [Accessed 10 October 2021].

[7] National Institute of Standards and Technology, “MINKOWSKI DISTANCE,” National Institute of Standards and Technology, 31 August 2017. [Online]. Available: https://www.itl.nist.gov/div898/software/dataplot/refman2/auxillar/minkdist.htm. [Accessed 10 October 2021].