

Example 3 – Reinforcement Learning:

Problem statement:

The problem statement is to program an agent that will get to a space in a video game map that punishes the agent for going down the middle of the map. This will force the agent to go the long way around the middle to get to the ideal position. The middle may be blocked off by walls or enemies and thus even though it is the fastest route will not produce the best outcome for the goal. The agent needs to learn to take the longer, safer route to get to the goal position. To visualise this problem statement the following map was created.

6						
5					+1	
4			-1			
3			-1			
2						
1						
	1	2	3	4	5	6

The values -1 simulate enemies and give the agent a negative reward if passed through them while the +1 position gives the agent a reward if the agent manages to get there. The grey spots signify walls that cannot be passed through. For the rest of the squares the reward function is set to -0.025 meaning to get the best reward the agent would have to go around the walls on either side of the map.

Description of the AI technique:

Reinforcement learning relies on having an agent perform a large number of trials to train the model and find the optimal policy for the agent to follow and maximise its rewards. For an agent to maximise its rewards it must maximise the utility of the state spaces so that the maximum of the expected immediate reward and the sum of the long-term rewards can be obtained. Once the utilisation of the spaces converges the agent will be able to follow the spaces with the most utilisation and thus maximise its rewards.

Reinforcement learning is great for games and for situations where the most optimal path or performance may not be known. For example, OpenAI made 5 agents that could outperform professional players in the game Dota 2. Dota 2 is an extremely complicated game that could not be represented in another technique such as a decision tree due to the inability to think of an optimised route for the game. So many things in a video game must be learned and cannot be prepared for thus using a technique like reinforcement learning that only cares about optimising its reward any means necessary is the perfect way to simulate a human learning to play a game.

To understand how the agent maximises its rewards, the method of maximising a state's utility must be understood. Essentially using the following equation:

$$U(s) = \max_a Q(s, a).$$

The maximum Q-value of a given state must be found which will provide the optimal utility value for that given space. The Q function which outputs a Q-value is the expected utility of taking a given action in a given state and can be found using the following formula:

$$\begin{aligned} Q(s, a) &= \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma U(s')] \\ &= \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma \max_{a'} Q(s', a')] \end{aligned}$$

Combining the two equations above the following equation for finding the optimal utility value is given:













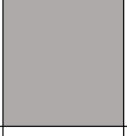
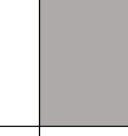


















$$U(s) = \max_{a \in A(s)} \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma U(s')].$$

This equation finds the optimal action “a” in a state “s”. To do this the equation sums all the possibilities “P” times the reward function “R” plus the discounted “γ” utility value “U” of the next state “s’ ”. Once the equation has been filled in and calculated the optimal utility for that particular state would have been found. To see the equation in action the code implementation will be discussed.

How the AI technique is used to solve the problem:

The first step is to define the map the agent will follow and the reward function for each space. As discussed earlier the reward function for each state is -0.025 and the goal state is +1 while the negative reward state is +1. The next step to solve the problem with reinforcement learning is to generate the probabilities for the different movements the agent can take in each space which is 0.84 for the intended action, 0.08 for turning 90 degrees to the left and 0.08 for turning 90 degrees to the right. The intended action is given by the following graph:

The states are defined by the following graph:

6						
5					+1	
4			-1			
3				-1		
2						
1	Start					
	1	2	3	4	5	6

This graph is called the policy and tells the agent the intended action for each state. The final value to generate is the discount value " γ " which was given a value of 0.9. Once all the values have been provided for the utility equation the next step is to train the model through a large number of trials. In the case of the code the number of trials were 500 and by the end of it the optimal utility values were calculated.

Solution Results:

The final calculations that the agent found to maximise its rewards were as followed:

```
(4, 4):1.0
(4, 0):0.8820137661027614
(0, 0):0.39537767922626266
(4, 3):0.9700801797210707
(2, 0):0.8099888363278516
(4, 2):0.9370766731151619
(3, 0):0.8507381380866944
(1, 0):0.5470057533031869
(4, 1):0.9116867587914881
(0, 1):-0.6311405684102491
(2, 4):-0.3193353559589077
(0, 4):-0.36940263262722584
(3, 4):0.974999582752025
(0, 3):-0.47910150300592813
(1, 4):-0.3443674471000174
(0, 2):-0.5637439770240278
(1, 2):-1.064067840619876
(1, 1):-1.0897720279054879
(2, 2):-1.0312500029918752
(3, 2):-1.0
(2, 1):-1.0632954243689103
(5, 3):0.9464416261529546
(5, 4):0.9733499685923144
(1, 5):0.87109375
(4, 5):0.96220703125
(2, 5):0.9115234375000001
(3, 5):0.9246826171875
(5, 1):0.8897779360221572
(5, 2):0.9175773224793359
(2, 3):-1.0
(5, 0):0.8609250779729336
(0, 5):0.846875
```

The co-ordinates on the left correspond to a state on the given map starting at 0 instead of 1 as seen above in the report and the number to the right of it shows the optimal utility values of each space calculated by the equation:

$$U(s) = \max_{a \in A(s)} \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma U(s')].$$

The solution shows that if the agent follows the states with the highest utility, then the agent will be able to maximise its rewards all the way to the goal state being 1. In order to improve the results techniques such as policy improvement could have been used which would have improved the Markov decision process (MDP) to produce a more optimal model for the agent to maximise its rewards.

References:

- Lu, H. 2021, 'Module 4 -- AI techniques for ability of learning -- Part III',
UTS Online Subject 41040 lecture notes, UTS, Sydney, viewed 20 September
2021, < <https://canvas.uts.edu.au/courses/18678/modules> >.
- Russell, S. J., Norvig, P. (2020). Artificial intelligence: A modern approach, 4th Edition.
Pearson.