# Artificial Intelligence – Q Learning

## What is Q-Learning?

Q-Learning is a part of Artificial Intelligence, used to determine the state action values for each possible state and its action. Q-Learning starts with all the values as zeroes in the Environment. The Reward for each state action pair helps in determining the Q value for a state action. The agent uses these Q values to traverse to the Goal.

Goal here means the State action pair having the maximum value. It can be any value. We will have it as 100.

Q-Learning is applied to a STATIC environment where we know the total number of States and the possible actions, without this knowledge the agent will not be able to determine the correct path.

I shall try to cover the important topics related to Q-Learning. I will not get into much mathematics to make things simpler. Out of curiosity I have gathered information from various sources and consolidated them here.

A complete iteration in an environment is called an EPISODE.

Let us consider an example for easy understanding:

R - Reward matrix having state in Rows and Action in Columns.

Q - Q matrix having state in Rows and Action in Columns.

Let us take an environment of 5 States and 5 Action:

**Junction2**

**Junction 1**

**Junction 3**

Ice cream Parlour

Signal

**Junction 4**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| State | Action | | | | | |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | 0 | 0 | -1 | 0 | -1 |
| 2 | 0 | 0 | 0 | -1 | 100 |
| 3 | -1 | 0 | 0 | -0.5 | 100 |
| 4 | 0 | -1 | -0.5 | 0 | -1 |
| 5 | 0 | 0 | -1 | 0 | 100 |

R =

Here are 4 junctions, the agent wants to go the Ice cream parlour from various junctions, the junction 5 is the ice cream parlour, there is a signal between 4 and 3 which takes a long time. They are bidirectional roads connecting each other, if there is no connectivity we give a reward of -1, for the signal we give reward of -0.5, and the road leading to Ice cream parlour is given a reward of 100.

Similarly we have a Q table as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| State | Action | | | | | |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |

Q =

The above Q table is initialized with 0 , we will use the formula given below to calculate the value of the Q(state,action) using the below formula:

|  |
| --- |
| Q(state, action) = R(state, action) + Gamma \* Max[Q(next state, all actions)] |

Example:

Gamma – Learning rate, lets keep it 0.4

1. Lets think that the agent is in Junction 1, the possible places he can go is (1,1), (1,2), (1,4)

Lets calculate Q(1,1) = R(1,1) + 0.4 \* Max[Q(1,1), Q(1,2), Q(1,4)]

Q(1,1) = 0 + 0.4 \* 0

Q(1,1) = 0

1. Lets imagine he has taken route (1,2) from (1,1), the possible places he can go is (2,3), (2,5), (2,2)

Q(1,2) = R(1,2) + 0.4 \* Max[Q(2,3), Q(2,5), Q(2,2)]

Q(1,2) = 0 + 0.4 \* 0

Q(1,2) = 0

1. Lets imagine he has taken route (2,5) from (1,2), the possible places he can go is (5,2), (5,5), (5,3)

Q(2,5) = R(2,5) + 0.4 \* Max[Q(5,2), Q(5,5), Q(5,3)]

Q(2,5) = 100 + 0.4 \* Max[0,100,0]

Q(2,5) = 100 + 40

Q(2,5) = 140

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| State | Action | | | | | |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 140 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |

Q=

Similarly all the Q values are updated and the agent learns to move around the optimal path.

## Exploitation Vs Exploration:

**Exploitation** is a greedy methodology where the agent moves in an optimal path without exploring the other possibilities, for eg in our environment the agent will move from Junction 4 to Junction 3 since it is the optimal path when compared to the other route which is a longer route.

**Exploration** is a method where the agent moves in a stochastic path leading to new path avoiding the negative rewards or penalty. In our eg the agent can move from Junction 4 -> Junction 1 -> Junction 2 -> Ice cream parlour.

Q-Learning has a disadvantage that it follows Exploitation rather than exploration. To overcome this there is a new algorithm called the SARSA( State Action Reward State Action).

The difference between the 2 is that the normal Q-Learning calculates the Q values of the present state using the Reward of current state and the Maximum of the next Q (state,action). So it will always choose an optimal path since it accounts the Max Q value of next states.

In case of SARSA, the Q value is calculated after moving to the next Q state, i.e from Q(S,A) it will move to Q(SX,AX) , Where X is the next new state, using this it will calculate the Reward and update the current Q(S,A).

Formula is as follows:

|  |
| --- |
| Q(state, action) = R(state, action) + Gamma \* Q(SX,AX) |

So now when the agent iterates through all the States it comes to know that avoiding signal is the best path, instead of choosing the road with the signal.

## Difference between Temporal Difference and Q – Learning:

In Q-Learning the Environment is deterministic, which means that we know all the state and its actions where as TD is used for policy evaluation in a non deterministic environment where there is no transition model available , the samples are generated by executing the policy and performing the stochastic value updates based on the states being visited. TD computes the state value for a given policy. Policy here is nothing but random selection of new state. Temporal Difference requires only the experience and not the environment

The Temporal Difference along with Q – Learning provides a powerful algorithm which can be applied to a larger unknown environment.

## On Policy and Off Policy Learning:

**On Policy:**

On policy learning is a kind of learning where the agent sticks to the policy and follows the policy strictly based on rewards, here very rarely the exploration takes places, always the optimum path is selected. There will be more of exploitation rather than exploration. The agent learns by experience.

**Off Policy:**

As the name suggest it is in contrast to On Policy learning where the agent learns and obtains rewards by selecting different and random paths, by this the agent learns and acts intelligently, it can find new ways and strategies which where not introduced in the learning phase unlike On policy agent which will adhere to the experiences gained in the learning phase.

**Action Selection Policies:**

**-greedy** - most of the time the action with the highest estimated reward is chosen, called the greediest action. Every once in a while, say with a small probability , an action is selected at random. The action is selected uniformly, independant of the action-value estimates. This method ensures that if enough trials are done, each action will be tried an infinite number of times, thus ensuring optimal actions are discovered.

**-soft** - very similar to -greedy. The best action is selected with probability 1 -  and the rest of the time a random action is chosen uniformly.

**Softmax** - one drawback of -greedy and -soft is that they select random actions uniformly. The worst possible action is just as likely to be selected as the second best. Softmax remedies this by assigning a rank or weight to each of the actions, according to their action-value estimate. A random action is selected with regards to the weight associated with each action, meaning the worst actions are unlikely to be chosen. This is a good approach to take where the worst actions are very unfavourable.

There are two types of Temporal Differences:

1. **Bootstrapping:**

The values are updated using the estimate of the successor state. Eg: A mouse running for cheese in an environment with a trap(often used example).

1. **Sampling:**

The values are updated by looking ahead for a sample successor state. The difference is that Bootstrapping knows its successor states where as sampling does not know its successor states. (Self Driving Cars)

TD and Q- Learning can be related using the below formula:

Qt(S,A) = Q(t-1)(S,A) + Gamma \* TDt(A,S)

**References:**

1. <https://www.tu-chemnitz.de/informatik/KI/scripts/ws0910/ml09_6.pdf>
2. <https://studywolf.wordpress.com/2013/07/01/reinforcement-learning-sarsa-vs-q-learning/>
3. <http://mnemstudio.org/path-finding-q-learning-tutorial.htm>
4. http://www.cse.unsw.edu.au/~cs9417ml/RL1/tdlearning.html
5. Hadelin and Kirill Eremenko courses on AI.