Q-Learning Numerical Example

by Kardi Teknomo (../../index.html)

(purchase.html)

< Previous (Q-Learning-Algorithm.htm) | Next (Tower-of-Hanoi.htm) | Contents (index.html) >

Q-Learning Numerical Examples

To understand how the Q (Q-Learning-Algorithm.htm) learning algorithm (Q-Learning-Algorithm.htm) works, we will go through several steps of numerical examples. The rest of the steps can be can be confirm using the program that I made (the companion files of this tutorial can be purchasecan be purchase if you purchase here (purchase.html))

Let us set the value of learning parameter $\gamma = 0.8$ and initial state as room B.

First we set matrix Q as a zero matrix.

$$\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

I put again the instant reward matrix R that represents the environment in here for your convenience.

$$\mathbf{R} = \begin{array}{c} state \setminus action \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} - & - & - & - & 0 & - \\ - & - & - & 0 & - & 100 \\ - & - & - & 0 & - & - \\ - & 0 & 0 & - & 0 & - \\ 0 & - & - & 0 & - & 100 \\ - & 0 & - & - & 0 & 100 \end{bmatrix} \end{array}$$

Look at the second row (state B) of matrix R. There are two possible actions for the current state B, that is to go to state D, or go to

state F. By random selection, we select to go to F as our action.

Now we consider that suppose we are in state F. Look at the sixth row of reward matrix R (i.e. state F). It has 3 possible actions to go to state B, E or F.

$$\mathbf{Q}(state, action) = \mathbf{R}(state, action) + \gamma \cdot Max\left[\mathbf{Q}(next\ state, all\ actions)\right]$$
$$\mathbf{Q}(B,F) = \mathbf{R}(B,F) + 0.8 \cdot Max\left\{\mathbf{Q}(F,B), \mathbf{Q}(F,E), \mathbf{Q}(F,F)\right\} = 100 + 0.8 \cdot 0 = 100$$

Since matrix Q that is still zero, $\mathbf{Q}(F,B), \mathbf{Q}(F,E), \mathbf{Q}(F,F)$ are all zero. The result of computation $\mathbf{Q}(B,F)$ is also 100 because of the instant reward.

The next state is F, now become the current state. Because F is the goal state, we finish one episode. Our agent's brain now contain updated matrix Q as

$$\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

For the next episode, we start with initial random state. This time for instance we have state D as our initial state.

Look at the fourth row of matrix R; it has 3 possible actions, that is to go to state B, C and E. By random selection, we select to go to state B as our action.

Now we imagine that we are in state B. Look at the second row of reward matrix R (i.e. state B). It has 2 possible actions to go to state D or state F. Then, we compute the Q value

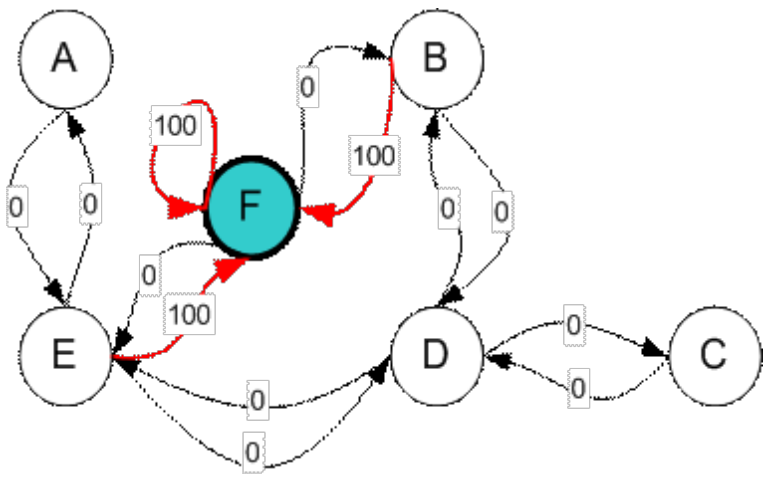
$$\mathbf{Q}(state, action) = \mathbf{R}(state, action) + \gamma \cdot Max\left[\mathbf{Q}(next\ state, all\ actions)\right]$$
$$\mathbf{Q}(D,B) = \mathbf{R}(D,B) + 0.8 \cdot Max\left\{\mathbf{Q}(B,D), \mathbf{Q}(B,F)\right\} = 0 + 0.8 \cdot Max\left\{0, 100\right\} = 80$$

We use the updated matrix Q from the last episode. $\mathbf{Q}(B,D) = 0$ and $\mathbf{Q}(B,F) = 100$. The result of computation $\mathbf{Q}(D,B) = 80$ because of the reward is zero. The Q matrix becomes

$$\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

The next state is B, now become the current state. We repeat the inner loop in Q learning algorithm because state B is not the goal state.

For the new loop, the current state is state B. I copy again the state diagram that represent instant reward matrix R for your convenient.



There are two possible actions from the current state B, that is to go to state D, or go to state F. By lucky draw, our action selected is state F.

Now we think of state F that has 3 possible actions to go to state B, E or F. We compute the Q value using the maximum value of these possible actions.

$$\mathbf{Q}(state, action) = \mathbf{R}(state, action) + \gamma \cdot Max\left[\mathbf{Q}(next\ state, all\ actions)\right]$$

$$\mathbf{Q}(B,F) = \mathbf{R}(B,F) + 0.8 \cdot Max\{\mathbf{Q}(F,B), \mathbf{Q}(F,E), \mathbf{Q}(F,F)\}$$
$$= 100 + 0.8 \cdot Max\{0,0,0\} = 100$$

The entries of updated Q matrix contain $\mathbf{Q}(F,B), \mathbf{Q}(F,E), \mathbf{Q}(F,F)$ are all zero. The result of computation $\mathbf{Q}(B,F)$ is also 100 because of the instant reward. This result does not change the Q matrix.

Because F is the goal state, we finish this episode. Our agent's brain now contain updated matrix Q as

$$\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

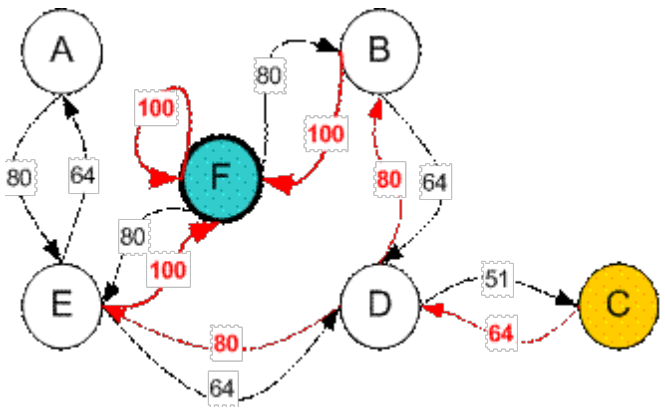
If our agent learns more and more experience through many episodes, it will finally reach convergence values of Q matrix as

$$\mathbf{Q} = \begin{array}{c} state \setminus action \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} - & - & - & - & 400 & - \\ - & - & - & 320 & - & 500 \\ - & - & - & 320 & - & - \\ - & 400 & 256 & - & 400 & - \\ 320 & - & - & 320 & - & 500 \\ - & 400 & - & - & 400 & 500 \end{bmatrix} \end{array}$$

This Q matrix, then can be normalized into a percentage by dividing all valid entries with the highest number (divided by 500 in this case) becomes

$$\hat{\mathbf{Q}} = \begin{array}{c} state \setminus action \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} - & - & - & - & 80 & - \\ - & - & - & 64 & - & 100 \\ - & - & - & 64 & - & - \\ - & 80 & 51 & - & 80 & - \\ 64 & - & - & 64 & - & 100 \\ - & 80 & - & - & 80 & 100 \end{bmatrix} \end{array}$$

Once the Q matrix reaches almost the convergence value, our agent can reach the goal in an optimum way. To trace the sequence of states, it can easily compute by finding action that makes maximum Q for this state.



For example from initial State C, it can use the Q matrix as follow:

From State C the maximum Q produces action to go to state D

From State D the maximum Q has two alternatives to go to state B or E. Suppose we choose arbitrary to go to B

From State B the maximum value produces action to go to state F

Thus the sequence is C -> D -> B -> F

< Previous (Q-Learning-Algorithm.htm) | Next (Tower-of-Hanoi.htm) | Contents (index.html) >

Preferable reference for this tutorial is

Teknomo, Kardi. 2005. Q-Learning by Examples. http://people.revoledu.com/kardi/tutorial/ReinforcementLearning/index.html