Reinforcement Learning	April 03, 2019
Recitation 6	
Lecturer:Yishay Mansour	*TA: Lee Cohen*

Part 1: Q learning

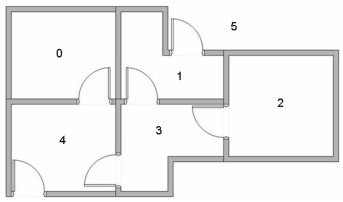
Algorithm

Initialize $Q(s,a)$, for all $s \in S$,$a \in A(s)$, arbitrarily, and $Q(terminal-state,*) = 0$.
Repeat (for each episode):

1. Initialize $s = s_0$.
2. Repeat (for every step of episode):
   (a) Choose $a \in A(s)$ using policy derived from Q (e.g., $\epsilon - greedy$).
   (b) Take action a, observe r, s'.
   (c) Update: $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_a Q(s',a) - Q(s,a)\right]$.
   (d) Update: $s \leftarrow s'$.

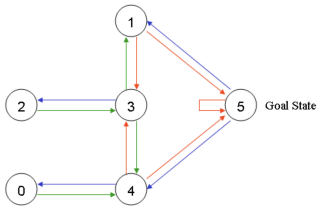
   End Episode when s is terminal

Exercise 1. Q learning The exercise describes an agent which uses unsupervised training to learn about an unknown environment. Suppose we have 5 rooms in a building connected by doors as shown in the figure below. We'll number each room 0 through 4. The outside of the building can be thought of as one big room (5). Notice that doors 1 and 4 lead into the building from room 5 (outside):

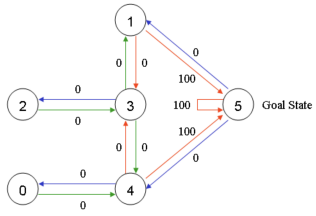


We can represent the rooms on a graph, each room as a node, and each door as a link:

In this exercise, we'd like to put an agent in any room, and from that room, go outside the building (this will be our target room). In other words, the goal room is number 5. To set



this room as a goal, we'll associate a reward value to each door (i.e. link between nodes). The doors that lead immediately to the goal have an instant reward of 100. Other doors not directly connected to the target room have zero reward. Because doors are two-way (0 leads to 4, and 4 leads back to 0), two arrows are assigned to each room. Each arrow contains an instant reward value, as shown below:



Train an agent that can exit the building with Q-Learning algorithm with discount factor of 0.8.

Solution 1. We'll call each room, including outside, a "state", and the agent's movement from one room to another will be an "action". Also, we will use $\alpha = 1$.

We can put the state diagram and the instant reward values into the following reward table:

$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

The -1's in the table represent null values (i.e.; where there isn't a link between nodes). For example, state 0 cannot go to state 1.

The agent starts out knowing nothing, the matrix Q is initialized to zero. In this example, for the simplicity of explanation, we assume the number of states is known (to be six). If we didn't know how many states were involved, the matrix Q could start out with only one element. It is a simple task to add more columns and rows in matrix Q if a new state is found.

Now let's imagine what would happen if our agent were in state 1 and by choosing randomly we move to state 5. Look at the sixth row of the reward matrix R (i.e. state 5). It has 3 possible actions: go to state 1, 4 or 5.

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$Q(1,5) = Q(1,5) + \alpha * (R(1,5) + \gamma * max(Q(5,1), Q(5,4), Q(5,5)) - Q(1,5)) = 0 + 1 * (100 + 0.8 * max(0,0,0) - 0) = 100$

Since matrix Q is still initialized to zero, Q(5, 1), Q(5, 4), Q(5, 5), are all zero. The result of this computation for Q(1, 5) is 100 because of the instant reward from R(5, 1).

The next state, 5, now becomes the current state. Because 5 is the goal state, we've finished one episode. Our agent's brain now contains an updated matrix Q as:

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

For the next episode, we start with a randomly chosen initial state. This time, we have state 3 as our initial state. Look at the fourth row of matrix R; it has 3 possible actions: go to state 1, 2 or 4. By random selection, we select to go to state 1 as our action.

Now we imagine that we are in state 1. Look at the second row of reward matrix R (i.e. state 1). It has 2 possible actions: go to state 3 or state 5. Then, we compute the Q value: $Q(3,1) = Q(3,1) + \alpha(R(3,1) + \gamma * max[Q(1,3), Q(1,5)] - Q(3,1)) = 0 + 1 * (0 + 0.8 * max(0,100) - 0) = 80$

We use the updated matrix Q from the last episode. Q(1,3)=0 and Q(1,5)=100. The result of the computation is Q(3,1)=80 because the reward is zero. The matrix Q becomes:

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

The next state, 1, now becomes the current state. We repeat the inner loop of the Q learning algorithm because state 1 is not the goal state.

Now, imaging we're in state 5, there are three possible actions: go to state 1, 4 or 5. We compute the Q value using the maximum value of these possible actions.

$Q(1,5) = Q(1,5) + \alpha * (R(1,5) + \gamma * max[Q(5,1), Q(5,4), Q(5,5)] - Q(1,5)) = 100 + 1 * (100 + 0.8 * max(0,0,0) - 100) = 100$

The updated entries of matrix Q, Q(5, 1), Q(5, 4), Q(5, 5), are all zero. The result of this computation for Q(1, 5) is 100 because of the instant reward from R(5, 1). This result does not change the Q matrix.

Because 5 is the goal state, we finish this episode. Our agent's brain now contain updated matrix Q as:

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

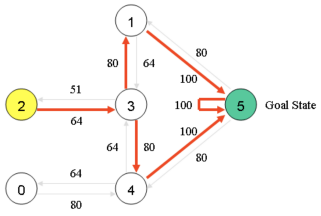
If our agent learns more through further episodes, it will finally reach convergence values in matrix Q like:

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$

This matrix Q, can then be normalized (i.e.; converted to percentage) by dividing all non-zero entries by the highest number (500 in this case):

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$

Once the matrix Q gets close enough to a state of convergence, we know our agent has learned the most optimal paths to the goal state. Tracing the best sequences of states is as simple as following the links with the highest values at each state:



Part 2: Monte Carlo

Algorithm

First-Visit Monte-Carlo Policy Evaluation.
In order to evaluate state s, in the first time-step t that state s is visited in an episode:

1. Increment counter $N(s) \leftarrow N(s) + 1$
2. Increment total return $\Gamma(s) \leftarrow \Gamma(s) + G_t$
3. Value is estimated by mean return $V(s) = \Gamma(s)/N(s)$
4. By law of large numbers, $V(s) \rightarrow V^{\pi}(s)$ as $N(s) \rightarrow \infty$

Every-Visit Monte-Carlo Policy Evaluation.
To evaluate state s, every time-step t that state s is visited in an episode:

1. Increment counter $N(s) \leftarrow N(s) + 1$
2. Increment total return $\Gamma(s) \leftarrow \Gamma(s) + G_t$
3. Value is estimated by mean return $V(s) = \Gamma(s)/N(s)$
4. By law of large numbers, $V(s) \rightarrow V^{\pi}(s)$ as $N(s) \rightarrow \infty$

Exercise 2. Monte Carlo
Consider an undiscounted Markov Reward Process with two states A and B. The transition matrix and reward function are unknown, but you have observed two sample episodes:
$A + 3 \rightarrow A + 2 \rightarrow B - 4 \rightarrow A + 4 \rightarrow B - 3 \rightarrow terminate$
$B - 2 \rightarrow A + 3 \rightarrow B - 3 \rightarrow terminate$

In the above episodes, sample state transitions and sample rewards are shown at each step, e.g. $A + 3 \rightarrow A$ indicates a transition from state A to state A, with a reward of $+3$.

(a) Using first-visit Monte-Carlo evaluation, estimate the state-value function $V(A)$, $V(B)$

(b) Using every-visit Monte-Carlo evaluation, estimate the state-value function $V(A)$, $V(B)$

Solution 2. state-value function estimation

Part (a) - first-visit Monte-Carlo evaluation:

1. $V(A) = (1/2) * (2 + 0) = 1$
2. $V(B) = (1/2) * (-3 - 2) = -5/2$

Part (b) - every-visit Monte-Carlo evaluation:

1. $V(A) = (1/4) * (2 - 1 + 1 + 0) = 1/2$
2. $V(B) = (1/4) * (-3 - 3 - 2 - 3) = -11/4$

References:

1. RL course, UCL, David Silver, Lecture 4-5
2. http://mnemstudio.org/path-finding-q-learning-tutorial.htm
3. "Learning and Planning in Dynamical Systems" by Shie Manor