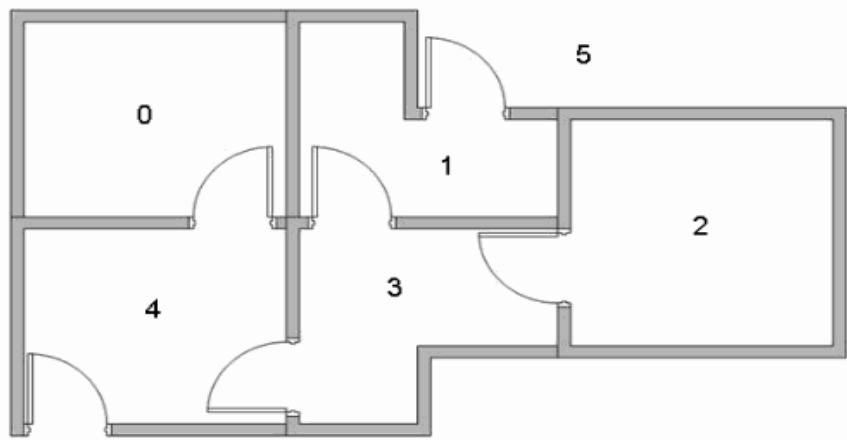
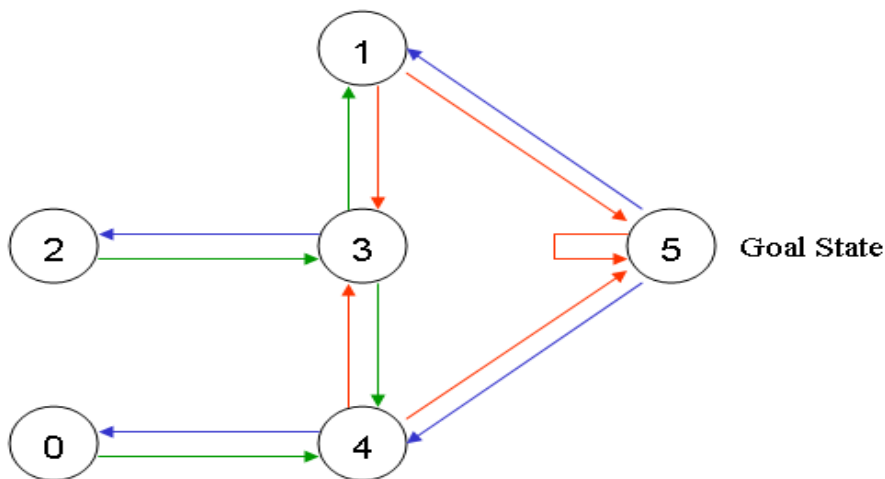


Q-Learning with a numerical example

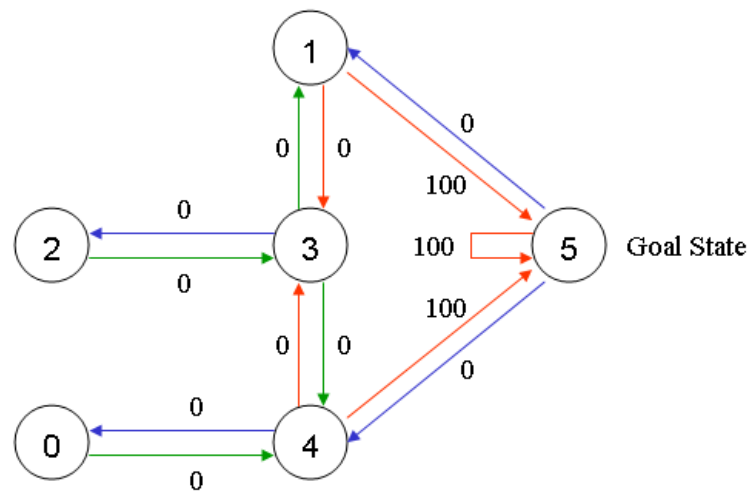
Suppose we have 5 rooms in a building connected by doors as shown in the figure below. We'll number each room 0 through 4. The outside of the building can be thought of as one big room (5). Notice that doors 1 and 4 lead into the building from room 5 (outside).



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



For this example, 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 rewards. Because doors are two-way (0 leads to 4, and 4 leads back to 0), two arrows are assigned to each room. We'll call each room, including outside, a "state", and the agent's movement from one room to another will be an "action". In our diagram, a "state" is depicted as a node, while "action" is represented by the arrows.



We can put the state diagram and the instant reward values into the following reward table, “matrix R”.

		Action					
State		0	1	2	3	4	5
0	$R =$	-1	-1	-1	-1	0	-1
1		-1	-1	-1	0	-1	100
2		-1	-1	-1	0	-1	-1
3		-1	0	0	-1	0	-1
4		0	-1	-1	0	-1	100
5		-1	0	-1	-1	0	100

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.

Now we'll add a similar matrix, “Q”, to the brain of our agent, representing the memory of what the agent has learned through experience. The rows of matrix Q represent the current state of the agent, and the columns represent the possible actions leading to the next state (the links between the nodes).

The agent starts out knowing nothing, the matrix Q is initialized to zero.

The transition rule of Q learning is a very simple formula:

$$Q(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \text{Gamma} * \text{Max}[Q(\text{next state}, \text{all actions})]$$

According to this formula, a value assigned to a specific element of matrix Q is equal to the sum of the corresponding value in matrix R and the learning parameter Gamma, multiplied by the maximum value of Q for all possible actions in the next state.

Our virtual agent will learn through experience, without a teacher (this is called unsupervised learning). The agent will explore from state to state until it reaches the goal. We'll call each exploration an episode. Each episode consists of the agent moving from the initial state to the goal state. Each time the agent arrives at the goal state, the program goes to the next episode.

The Q-Learning algorithm goes as follows:

1. Set the gamma parameter and environment rewards in matrix R.
2. Initialize matrix Q to zero.
3. For each episode:

Select a random initial state.

Do While the goal state hasn't been reached.

Select one among all possible actions for the current state.

Using this possible action, consider going to the next state.

Get maximum Q value for this next state based on all possible actions.

Compute:

$$Q(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \text{Gamma} * \text{Max}[Q(\text{next state}, \text{all actions})]$$

Set the next state as the current state.

The Gamma parameter has a range of 0 to 1 ($0 \leq \text{Gamma} \leq 1$). If Gamma is closer to zero, the agent will tend to consider only immediate rewards. If Gamma is closer to one, the agent will consider future rewards with greater weight, willing to delay the reward.

To use the matrix Q, the agent simply traces the sequence of states, from the initial state to the goal state. The algorithm finds the actions with the highest reward values recorded in matrix Q for the current state:

Algorithm to utilize the Q matrix:

1. Set current state = initial state.
2. From the current state, find the action with the highest Q value.
3. Set current state = next state.
4. Repeat Steps 2 and 3 until current state = goal state.

The algorithm above will return the sequence of states from the initial state to the goal state.

Q-Learning Example By Hand

We'll start by setting the value of the learning parameter $\gamma = 0.8$, and the initial state as Room 1.

Initialize matrix Q as a zero matrix:

[illegible]

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

Look at the second row (state 1) of matrix R. There are two possible actions for the current state 1: go to state 3, or go to state 5. By random selection, we select to go to 5 as our action.

Now let's imagine what would happen if our agent were in state

5. Look at the sixth row of the reward matrix R (i.e. state 5). It has 3 possible actions: go to states 1, 4, or 5.

$Q(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \text{Gamma} * \text{Max}[Q(\text{next state}, \text{all actions})]$

$$\begin{aligned} Q(1, 5) &= R(1, 5) + 0.8 * \text{Max}[Q(5, 1), Q(5, 4), Q(5, 5)] \\ &= 100 + 0.8 * 0 = 100 \end{aligned}$$

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:

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

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 states 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:

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:

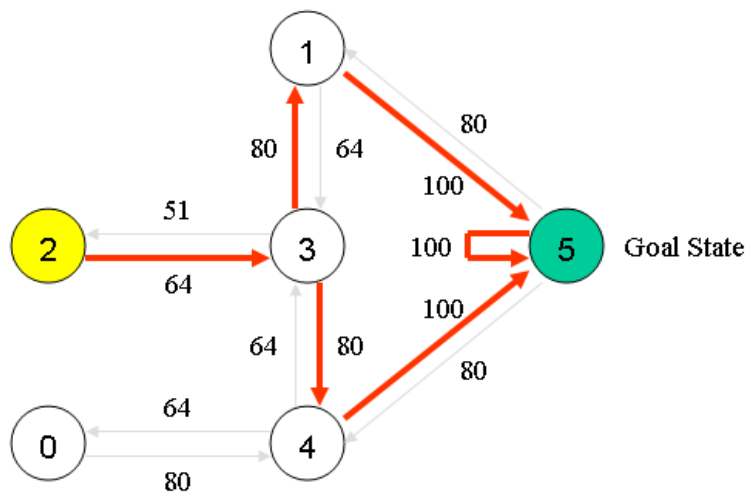
$$Q = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}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

Our agent's brain now contains updated matrix Q as:

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

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

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



For example, from initial State 2, the agent can use the matrix Q as a guide: From

State 2 the maximum Q values suggest the action to go to state 3.

From State 3 the maximum Q values suggest two alternatives: go to state 1 or 4.

Suppose we arbitrarily choose to go to 1.

From State 1 the maximum Q values suggest the action to go to state 5. Thus the sequence is 2 – 3 – 1 – 5.