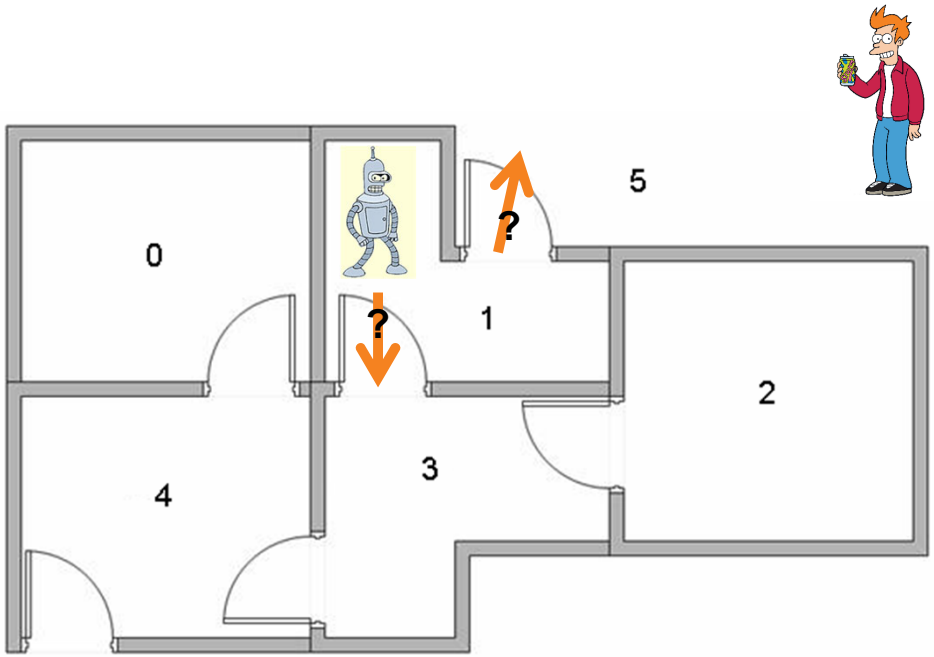
Q-Learning

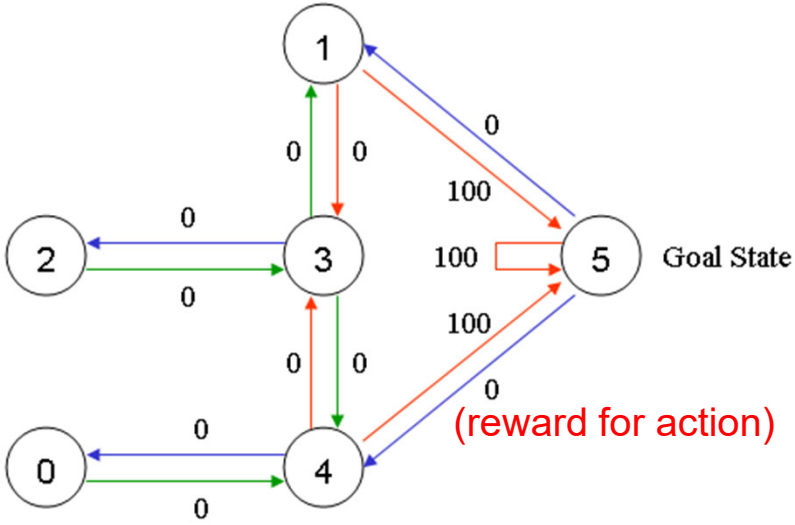
- Simple approach to solve Markov Decision Process

  $$Q(s, a) = R(s, a) + Gamma * Max[Q(next\ state, all\ actions)]$$

- Gamma: discount factor (between 0 and 1)
  - How much of future rewards to consider compared to present rewards
- “Learning from experience”

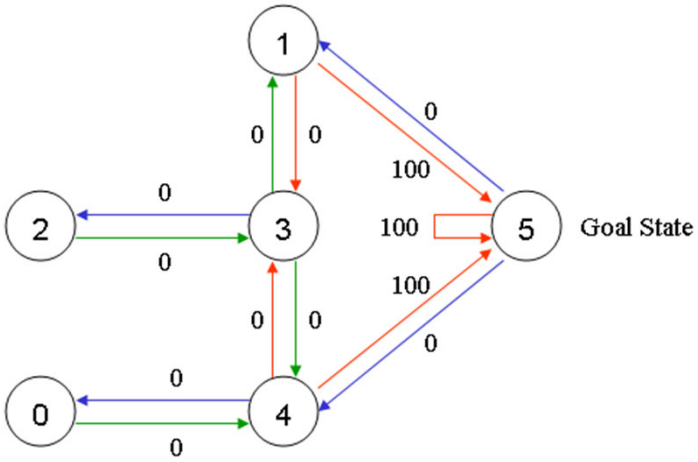
Q-Learning: Learn the Path through Rewards





Demo: https://github.com/lisaong/diec/blob/master/day4/rl/path_finding_demo.ipynb

Q-Learning: Path Finding



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

Q-Learning Task:

- Complete the Q-values table so that robot can find the **best path** to take at a **given state**
- -1 values indicate invalid directions

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

Q-Learning: Algorithm

Select parameter Gamma, set rewards in matrix R

Initialize matrix Q to zero

```
For each episode {
        Select a random initial state
        While (goal state not reached) {
                Select 1 possible action for current state
                Using the selected action, consider going to the next state

                Get maximum Q value for next state
                Set the next state as the current state
        }
}
```

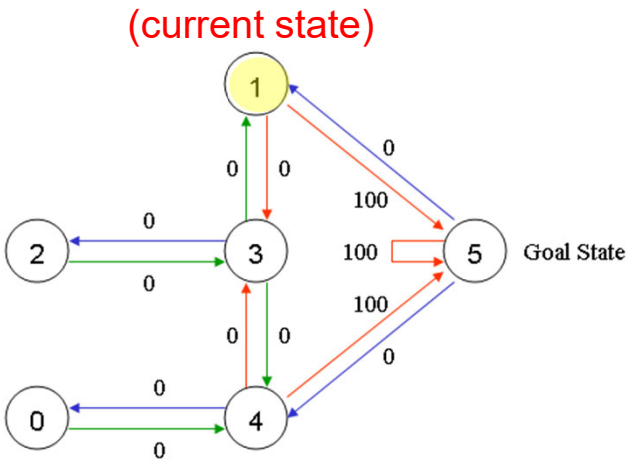
Q-Learning: Episode 1

Gamma = 0.8

State = 1 (random)

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

(Action / State)



Possible action: 5 (selected randomly from 3 and 5)

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

Q(1, 5) = R(1, 5) + 0.8 * Max[Q(5, 1), Q(5, 4), Q(5, 5)]

= 100 + 0.8 * 0 = **100**

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

(Action / State)

Q-Learning: Episode 2

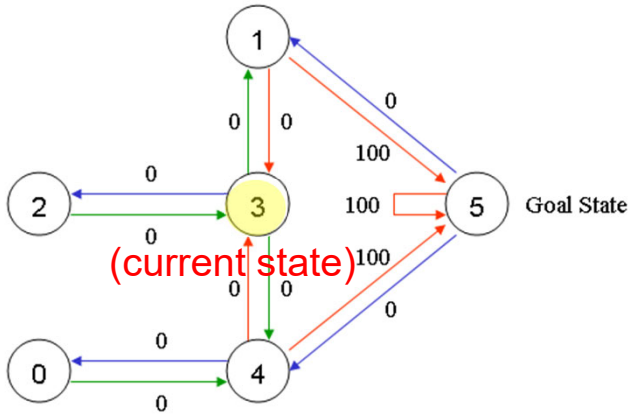
Gamma = 0.8

State = 3 (random)

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

Action

State



Possible action: 1 (selected randomly from 1, 2, and 4)

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

Q(3, 1) = R(3, 1) + 0.8 * Max[Q(1, 2), Q(1, 5)]

= 0 + 0.8 * Max(0, 100) = **80**

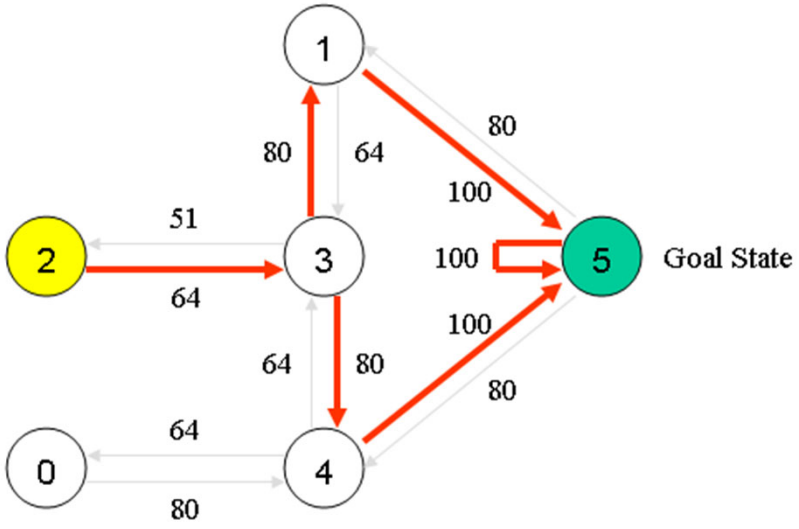
$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 &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

Action

State

Q-Learning: Convergence

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

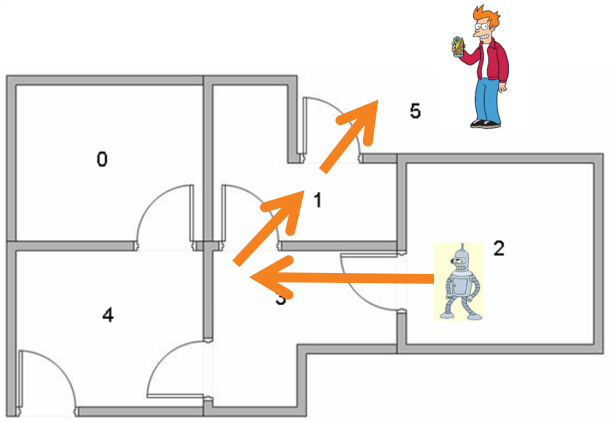


What path should the Robot use if starting from 2?

State 2: Maximum Q-value is for state 3
State 3: Maximum Q-value is same for state 1 or 4 => random choice

State 1: Maximum Q-value is state 5. Path: 2-3-1-5
State 4: Maximum Q-value is state 5. Path: 2-3-4-5

Q: 2-3-4-5 is a longer path, how do you take distance into account?



Q-Learning Enhancements

- Q-Learning when rewards for all state, action combinations *not fully known*
  - In real-world or complex environments, not all possible actions and states can be enumerated (e.g. autonomous driving, Starcraft)
- Strategies:
  - **Monte Carlo:** take average of the Q-values observed so far
  - **Temporal Differencing**: use the difference with previous step's Q-value to estimate the next Q-value
  - **Deep Q-Learning**: use a deep neural network to estimate the Q-values