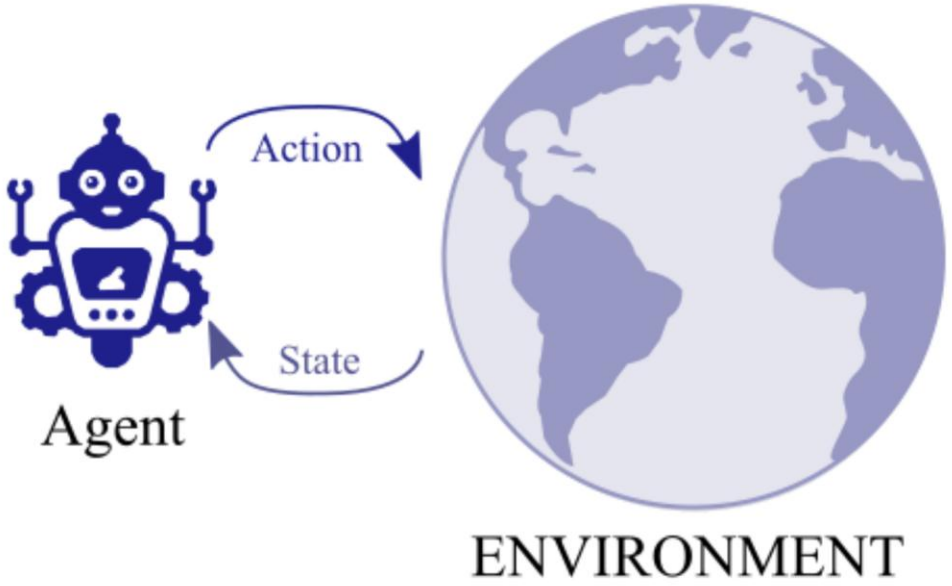
Reinforcement Learning
Q-Learning

- Instructor:
- Dr Yasser Alginahi

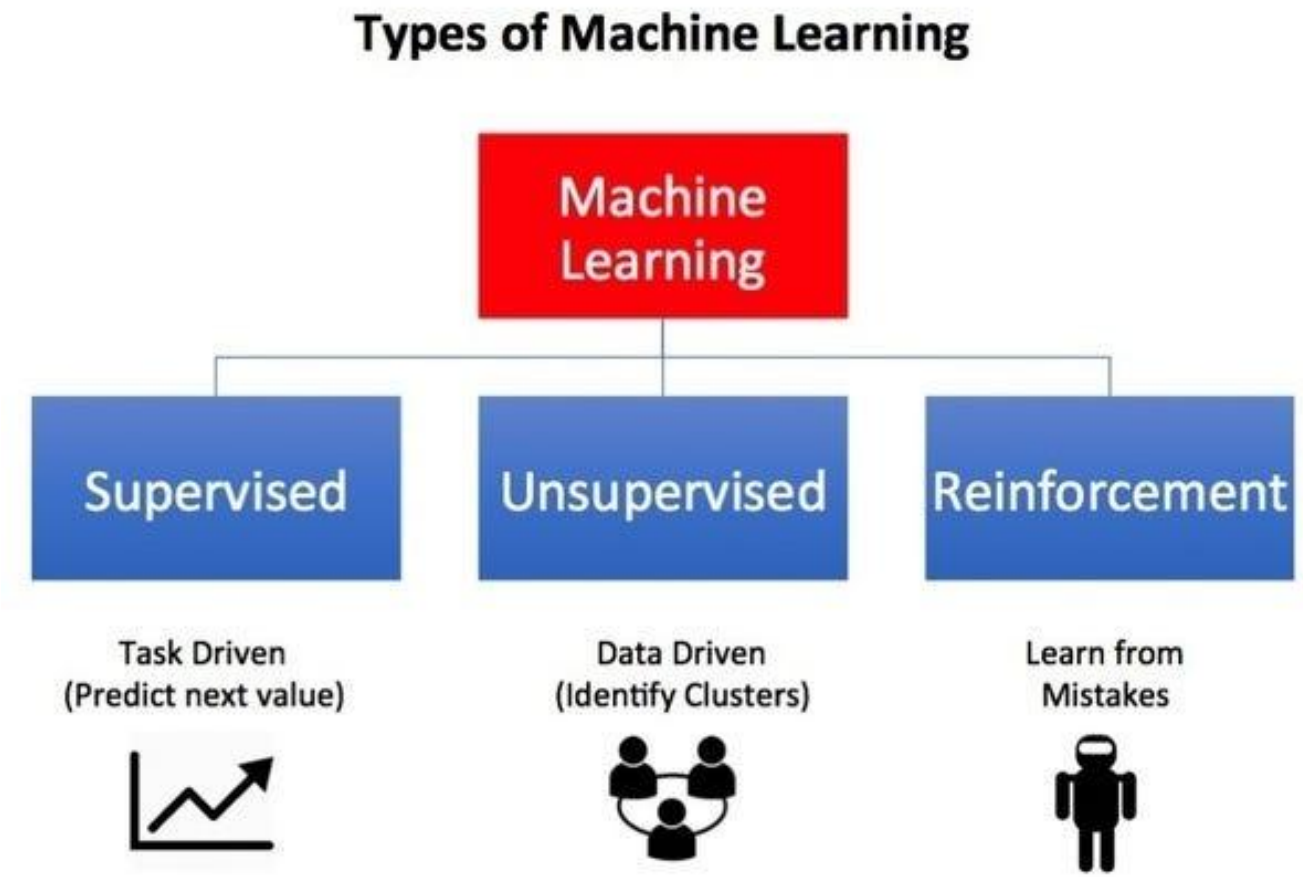
- Provided by:
- Somaiyeh Vaziri
- Atefeh Hafezizadeh

- November 2023
- University of Windsor



CONTENTS

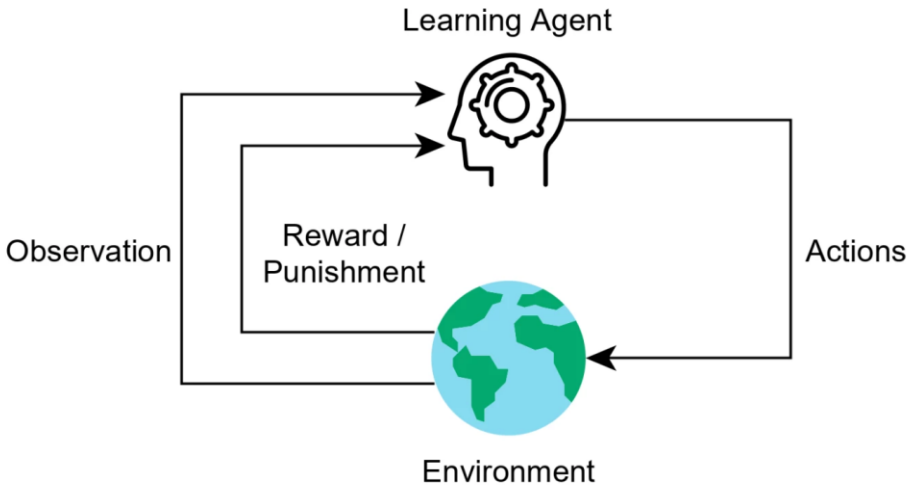
Introduction



[Source: https://tinyurl.com/mry75nzb]

Reinforcement Learning

- ***RL*** is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences.

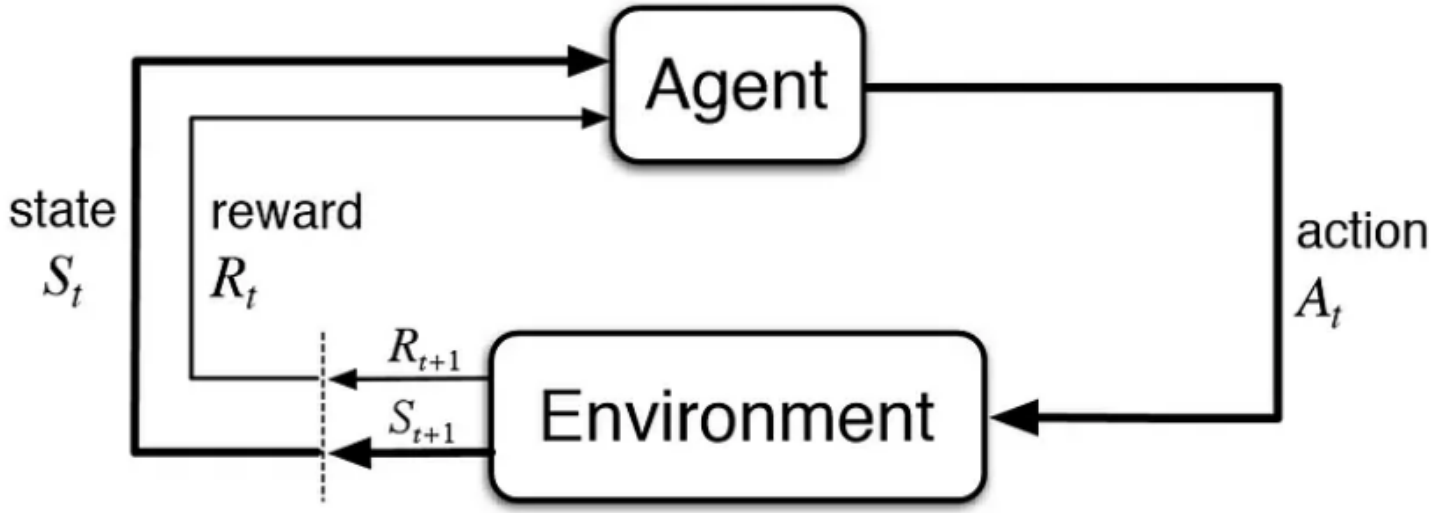


[Source: https://tinyurl.com/5n795dt9]

- Percepts received by an agent should be used not only for acting, but also for improving the agent's ability to behave optimally in the future to achieve the goal.

Reinforcement Learning:

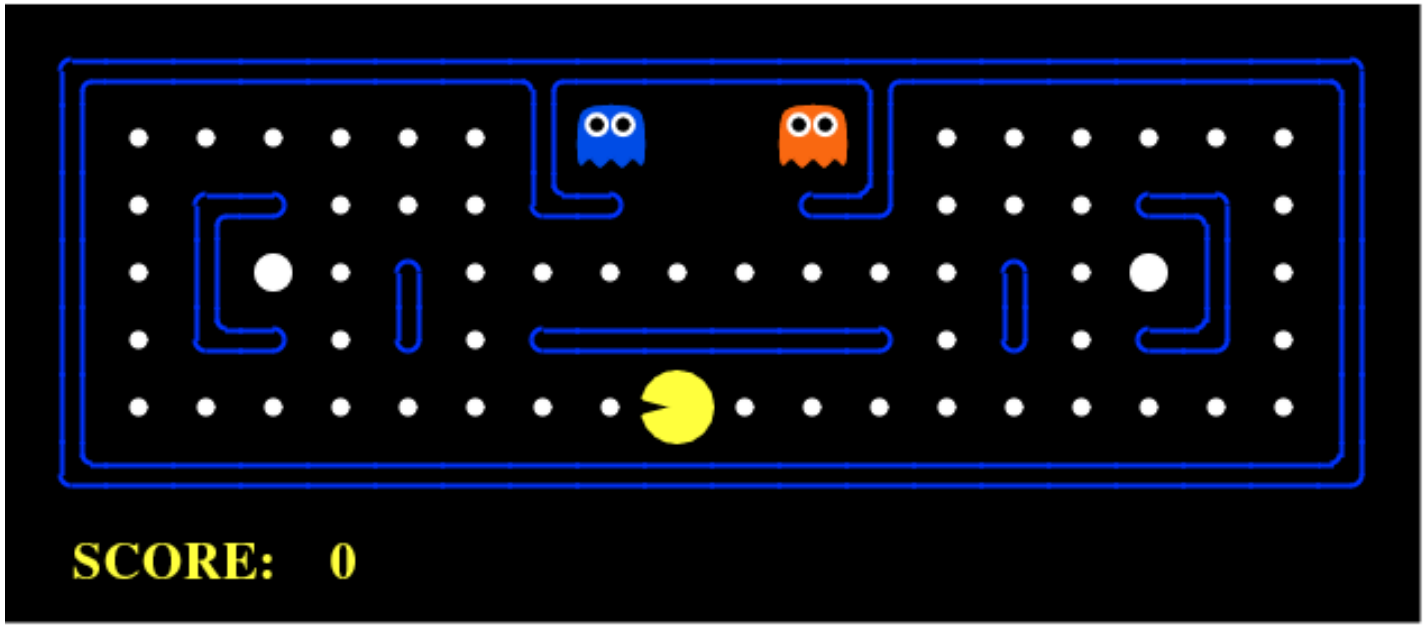
The action-reward feedback loop of a generic RL model



[Source: https://tinyurl.com/mry75nzb]

Reinforcement Learning:

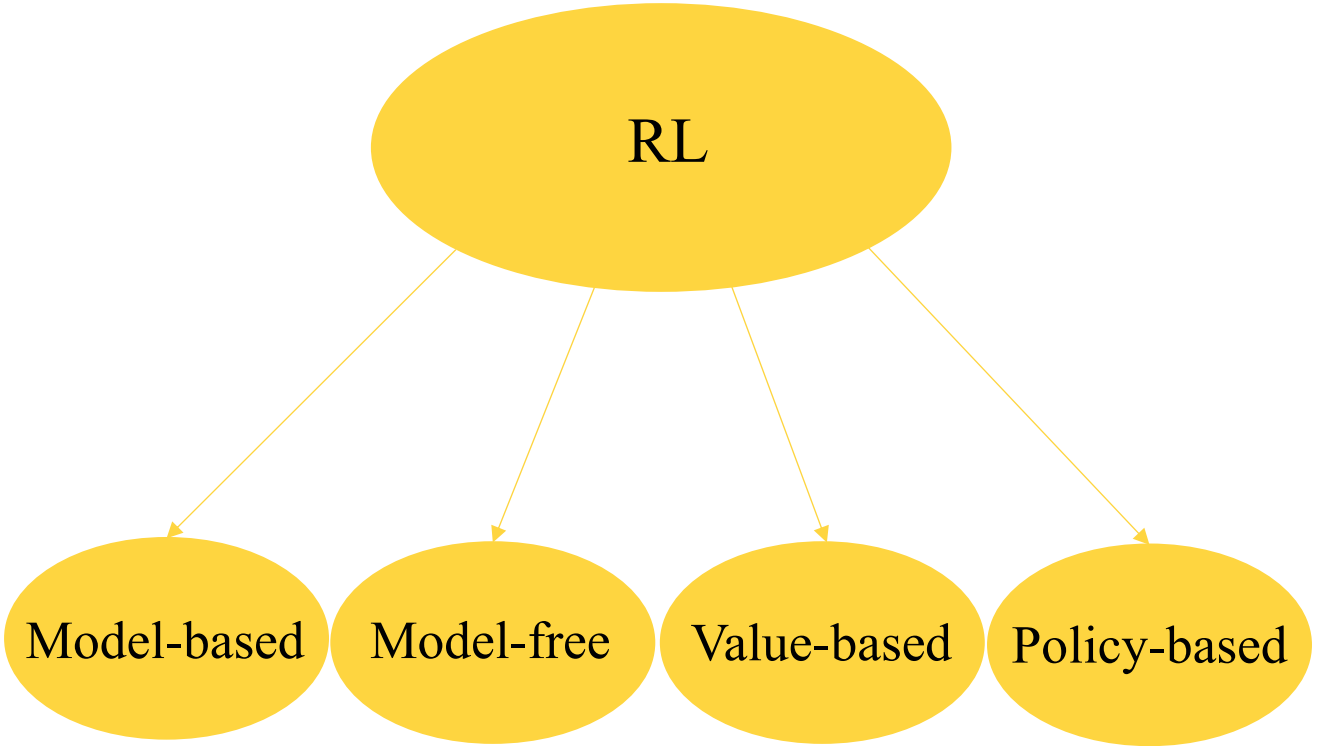
A clear example of RL



[Source: https://tinyurl.com/mry75nzb]

Reinforcement Learning:

Reinforcement learning categories



Q-Learning

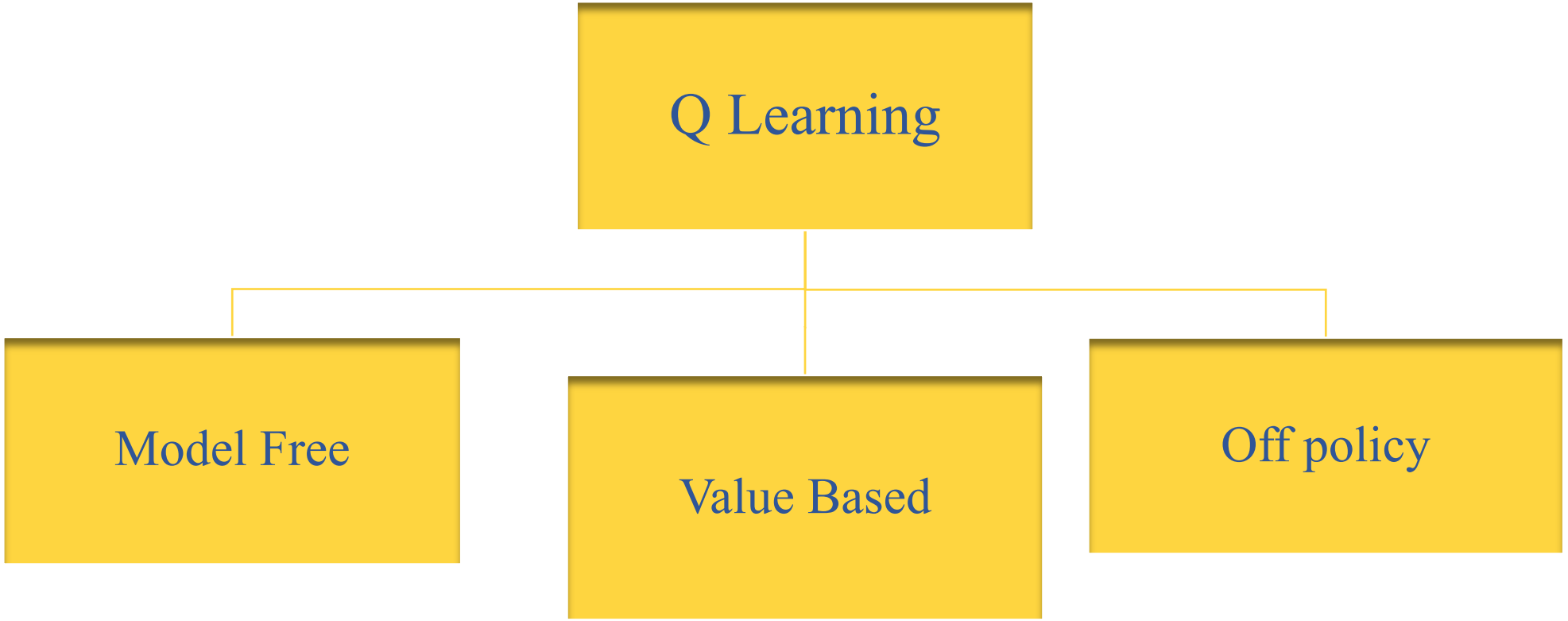
The “Q” stands for quality

Q-learning is an algorithm that will find the best series of ***actions*** based on the agent's ***current state***

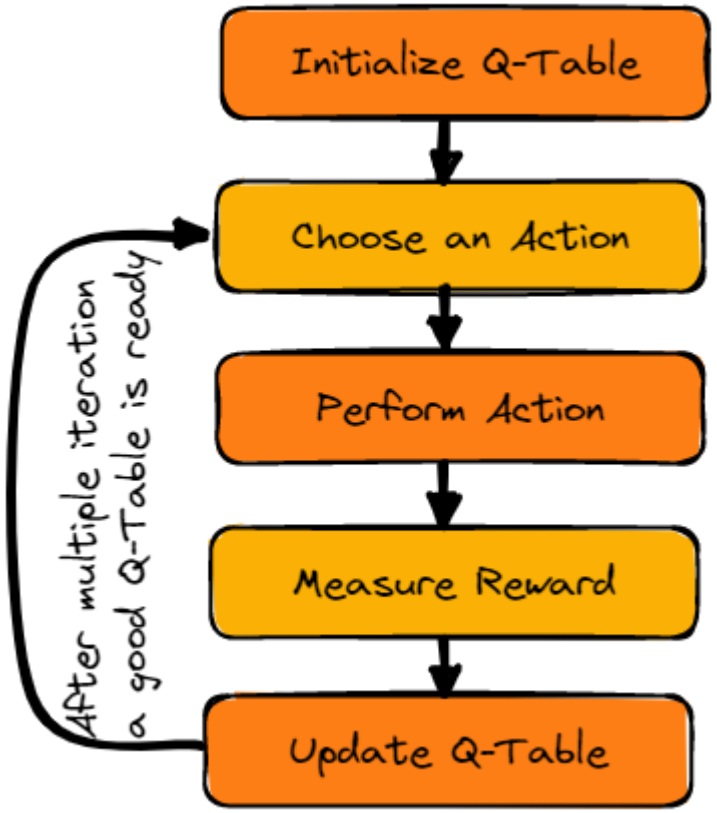
Quality of each state represents how valuable the action is in maximizing future rewards

Q-Learning:

Q-Learning categories



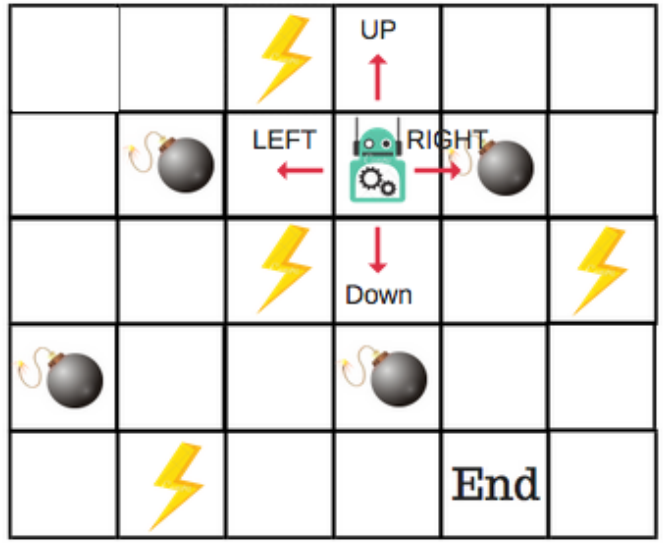
Q-Learning:
Q-learning algorithm



[Source: https://tinyurl.com/bdcd2mxt]

Q-Learning:

Key Terminologies in Q-learning



- *States(s):* the current position of the agent in the environment.
- *Action(a):* a step taken by the agent in a particular state.
- *Rewards:* for every action, the agent receives a reward and penalty. [Source: https://tinyurl.com/5eryh9xz]
- *Episodes:* the end of the stage, where agents can't take new action.

  - It happens when the agent has achieved the goal or failed.

Q-Learning:

Key Terminologies in Q-learning

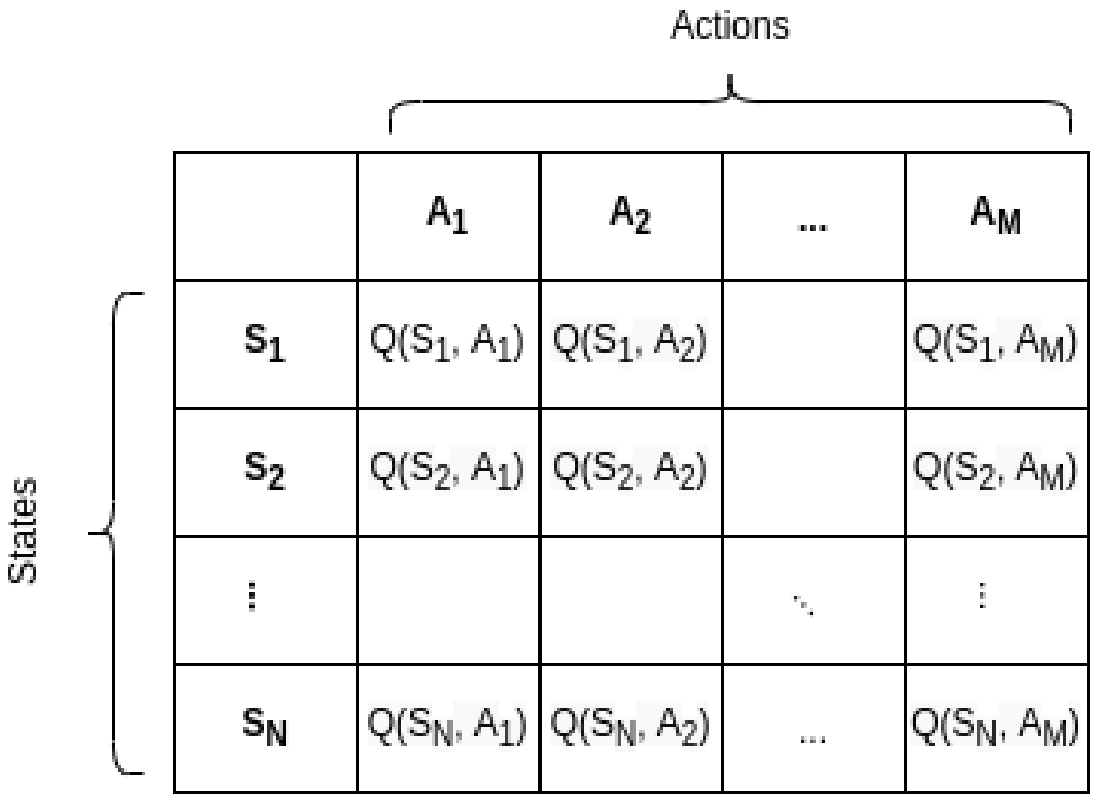
- *$Q(S_{t+1}, a)$:* expected optimal Q-value of doing the action in a particular state.
- *$Q(S_t, A_t)$:* it is the current estimation of $Q(S_{t+1}, a)$.
- *Q-Table:* the agent maintains the Q-table of sets of states and actions.
- *Q-function:* It estimates the expected cumulative reward of taking a particular action.
- *Temporal Differences(TD):* used to estimate the expected value of $Q(S_{t+1}, a)$ by using the current state and action and previous state and action.

Q-Learning:

Q-Table

- The agent will use a Q-table to take the best possible action based on the expected reward for each state in the environment.

- In simple words, a Q-table is a data structure of sets of actions and states, and we use the Q-learning algorithm to update the values in the table.



[Source: https://tinyurl.com/3x9vrnfr]

Q-Learning:

Q-Function

The Q-function uses the Bellman equation and takes state(s) and action(a) as input.

The equation simplifies the state values and state-action value calculation.

$$Q^{\pi}(s_t, a_t) = \mathrm{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | s_t, a_t]$$

Q-values for the state given a particular state

Expected discounted cumulative reward

Given the state and action

Q-Learning:

An Introduction to Bellman Equation's Parameters

Alpha(α) = learning rate $0 < \alpha < 1$

- alpha defines the learning rate or step size. As we can see from the equation, the new Q-value for the state is calculated by incrementing the old Q-value by alpha multiplied by the selected action's Q-value.

γ= Gamma is the discount factor. $0 < \gamma < 1$

- In Q-learning, gamma is multiplied by the estimation of the optimal future value. The next reward's importance is defined by the gamma parameter.

Q-Learning:

Updating Q-values through Bellman equation

- We will update the function $Q(S_t, A_t)$ using the equation. It uses the previous episode's estimated Q-values, learning rate, and Temporal Differences error.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

New Q-value estimation

Former Q-value estimation

Learning Rate

Immediate Reward

Discounted Estimate Optimal Q-value Of next state

TD Target

Former Q-value estimation

TD Error

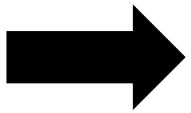
Q-Learning:

A sample of updating Q-table

$$Q(s,a)=(1-\alpha)Q(s,a)+(\alpha)[R(s,a,s')+\gamma \max Q(s',a')]$$

	A1	A2	A3	A4
S1	0	0	0	0
S2	0	0	0	0
S3	0	0	0	0

Updating

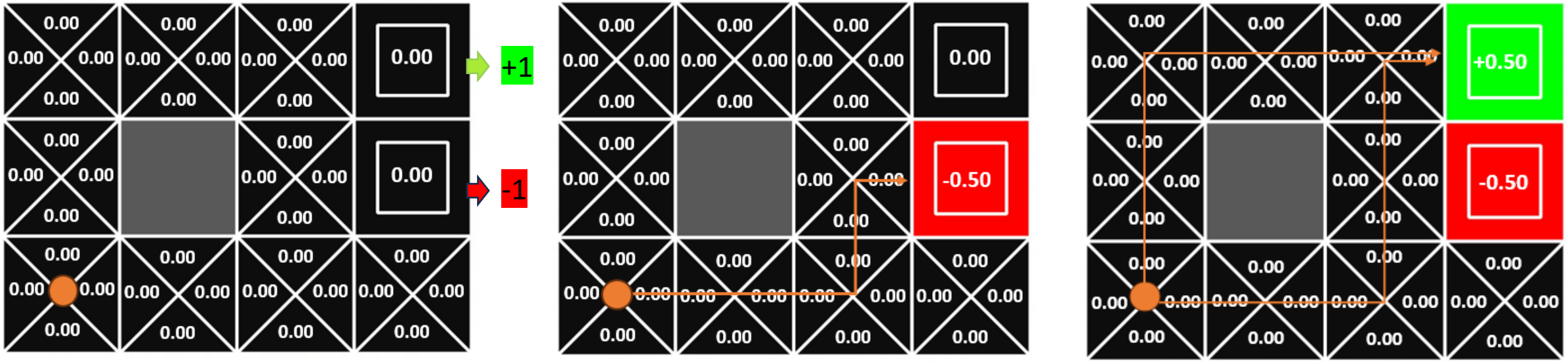


	A1	A2	A3	A4
S1	0.4	0.8	0.2	0.1
S2	0.7	0.3	0.6	0
S3	0.7	0.1	0.3	0.2

Q-Learning:

Updating Q-values

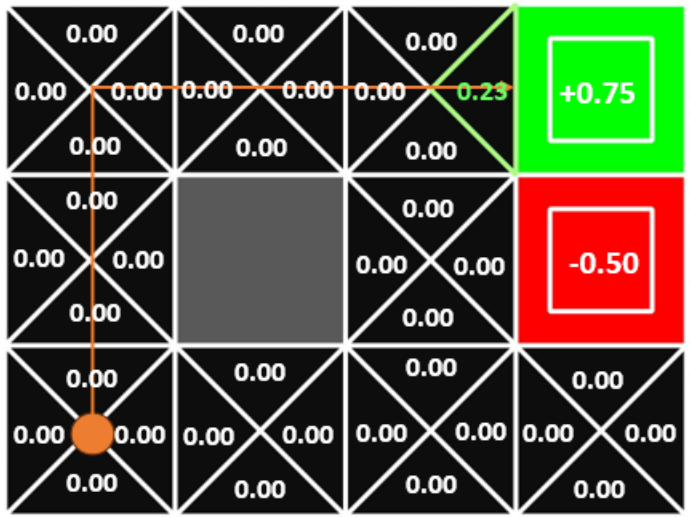
$\gamma = 0.9$, $\alpha = 0.5$, $R = +1, -1$

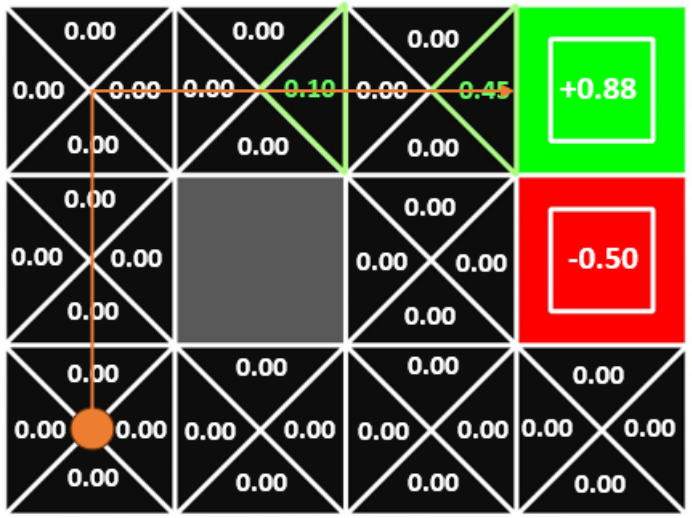


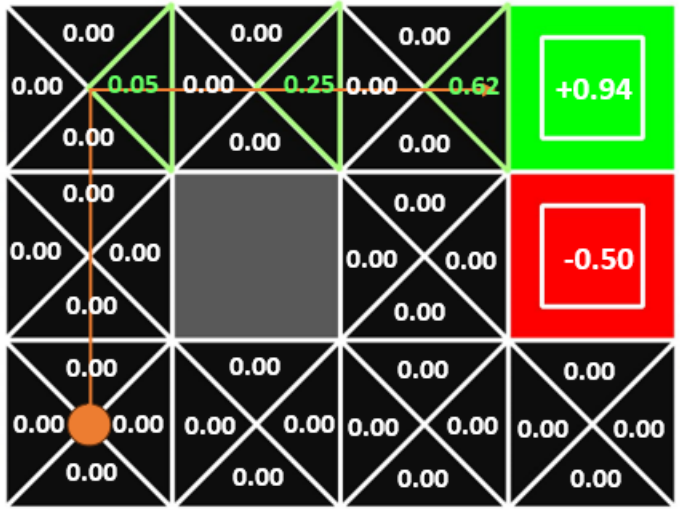
$$Q(s,a)=(1-\alpha)Q(s,a)+(\alpha)[R(s,a,s')+\gamma \max Q(s',a')]$$

Q-Learning:

Updating Q-values



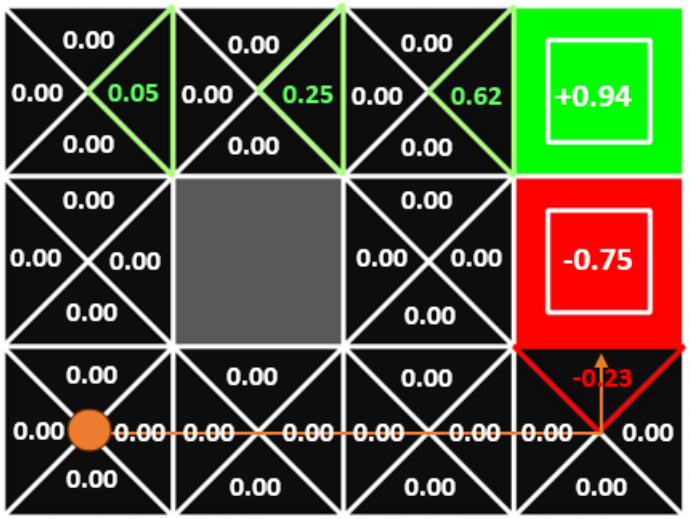




$$Q(s,a)=(1-\alpha)Q(s,a)+(\alpha)[R(s,a,s')+\gamma \max Q(s',a')]$$

Q-Learning:

Updating Q-values

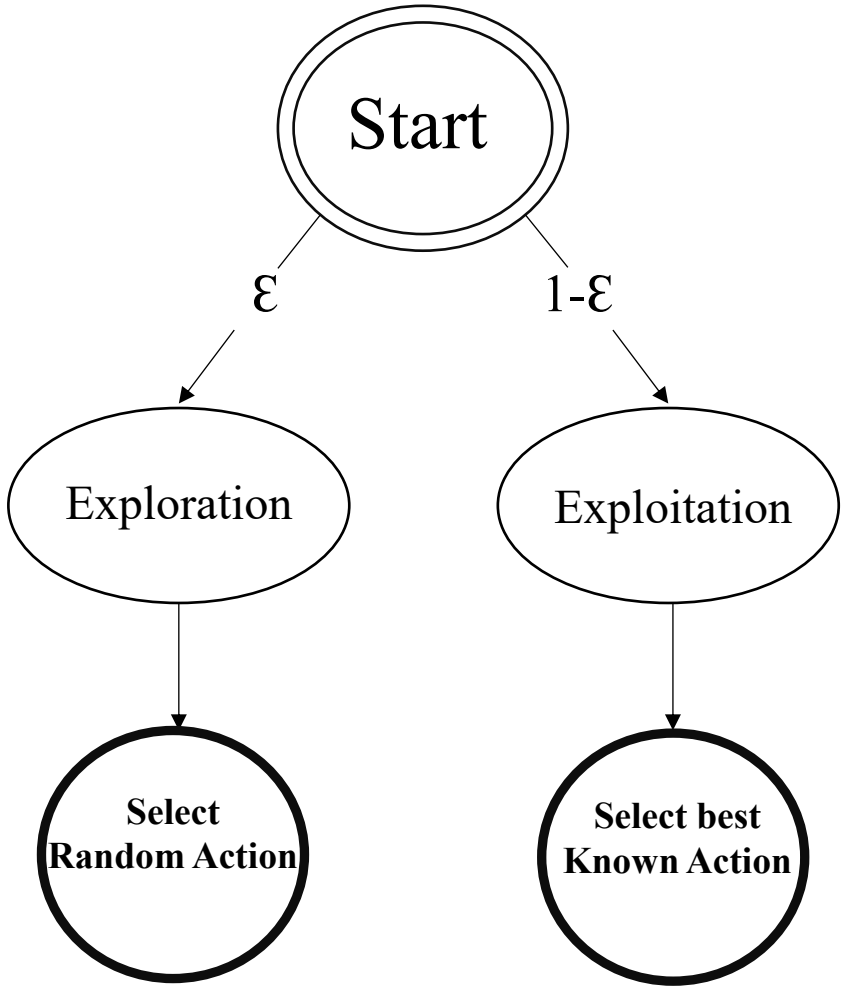


The agent continue learning to …??

$$Q(s,a)=(1-\alpha)Q(s,a)+(\alpha)[R(s,a,s')+\gamma \max Q(s',a')]$$

Exploration vs. Exploitation

- Epsilon is related to the epsilon-greedy action selection procedure in the Q-learning algorithm.
- In epsilon-greedy action selection, the agent uses exploitations to take advantage of prior knowledge and exploration to look for new options.



Sample code

Using python and with assuming the below parameters, we wrote a code for updating Q-table.

➢Parameters

alpha = 0.1, gamma = 0.6, epsilon = 0.1,

num_episodes=10000,
max_steps_per_episode=100

https://colab.research.google.com/drive/1OfwQQqhmK8aSaM0AS6xc1OFIfbl46W42?usp=sharing

Sample code

```
import numpy as np
import random

# Parameters
alpha = 0.1  # Learning rate
gamma = 0.6  # Discount factor
epsilon = 0.1  # Exploration rate
num_episodes = 10000  # Number of episodes to train on
max_steps_per_episode = 100  # Limit the number of steps per episode

# Environment setup
num_states = 10
num_actions = 3
Q = np.zeros((num_states, num_actions))  # Q-table initially filled with zeros
```

Sample code

```
# Dummy environment response (to be replaced with a real environment)
def step(state, action):
    # This function should be replaced with the actual dynamics of the environment
    # For the sake of example, we assume every action leads to a new state
    # randomly and gives a random reward.
    if num_states == 9:
        return random.randint(0, num_states - 1), reward==+1
    elif num_states==8:
        return random.randint(0, num_states - 1), reward==-1
    else:
        return random.randint(0, num_states - 1), random.uniform(-1, 1)
```

Sample code

```
# Training loop
for episode in range(num_episodes):
    state = random.randint(0, num_states - 1)  # Start at a random state

    for _ in range(max_steps_per_episode):  # Changed 'step' to '_' to avoid conflict
        # Exploration-exploitation trade-off
        if random.uniform(0, 1) < epsilon:
            action = random.randint(0, num_actions - 1)  # Explore action space
        else:
            action = np.argmax(Q[state])  # Exploit learned values

        # Take action and observe the outcome state and reward
        new_state, reward = step(state, action)  # 'step' here is the function defined
earlier
```

Sample code

```
 # Q-learning algorithm
        Q[state, action] = Q[state, action] +
alpha * (reward + gamma * np.max(Q[new_state]) -
Q[state, action])

        # Transition to the next state
        state = new_state

# Display the learned Q-table
print("Q-table:")
print(Q)
```

```
Q-table:
[[-0.02776233 -0.01735738 0.04502264]
[ 0.08347004 -0.13304128 0.15789628]
[-0.02942451 -0.13063879 0.13146398]
[-0.00323057 0.0427315 -0.0603509 ]
 [ 0.12286208 -0.08334864 -0.03393845]
[-0.06184578 -0.29502079 0.13205288]
 [ 0.22461559 -0.03299951 -0.08343481]
[-0.14660276 -0.00654734 0.09716222]
[-0.13534533 -0.02063201 0.15570354]
[-0.08623073 0.11169402 -0.08088196]]
```

Applications

- Adaptive traffic signal control based on delay minimization strategy
- Playing the 2600 Atari game using Deep Q-learning
- Human-level control through deep Q-learning
- Hybrid Control for Robot Navigation
- Multi-agent cooperation for Robot Soccer

Summary

- ✓ Q-Learning is one of the common methods available in Reinforcement Learning.
- ✓ Trial-and-error method is used for the exploratory agent to explore in a complex and nondeterministic environment, and then execute (exploitation) the best action based on experience.
- ✓ The experience is based on the reward or penalty that received from the previous trial-and-error action.
- ✓ Reinforcement learning algorithm promised to improve the performance of an agent with the agent's experience.

References

[1]. W. by: baeldung, “Epsilon-Greedy Q-Learning,” Baeldung on Computer Science, https://www.baeldung.com/cs/epsilon-greedy-q-learning (accessed Nov. 15, 2023)

[2]. B. Jang, M. Kim, G. Harerimana, and J. W. Kim, “Q-Learning Algorithms: A comprehensive classification and applications,” *IEEE Access*, vol. 7, pp. 133653–133667, 2019. doi:10.1109/access.2019.29412299

[3]. A. A. Awan, “An introduction to Q-learning: A tutorial for Beginners,” DataCamp, https://www.datacamp.com/tutorial/introduction-q-learning-beginner-tutorial (accessed Nov. 15, 2023).

[4]. S. Bhatt, “Reinforcement learning 101,” Medium, https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292 (accessed Nov. 15, 2023)

[5]. freeCodeCamp.org, “An introduction to Q-learning: Reinforcement learning,” freeCodeCamp.org, https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/ (accessed Nov. 15, 2023).

[6]. Y. K. Chin, L. K. Lee, N. Bolong, S. S. Yang, and K. T. Teo, “Exploring Q-learning optimization in Traffic Signal Timing Plan Management,” *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks*, 2011. doi:10.1109/cicsyn.2011.64

[7]. L. Canese *et al.*, “Multi-agent Reinforcement Learning: A review of challenges and applications,” *Applied Sciences*, vol. 11, no. 11, p. 4948, 2021. doi:10.3390/app11114948