Big Data Technology and Application Institute

REINFORCEMENT LEARNING

Speaker : XinLei Zhou

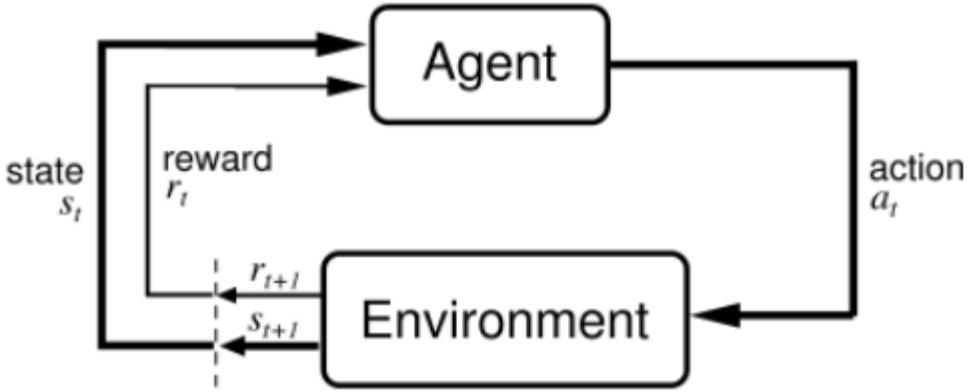
Outline

- Introduction to Reinforcement Learning
  - Important elements
  - Compared with familiar concepts
- How it work（ take Q-Learning as an example)

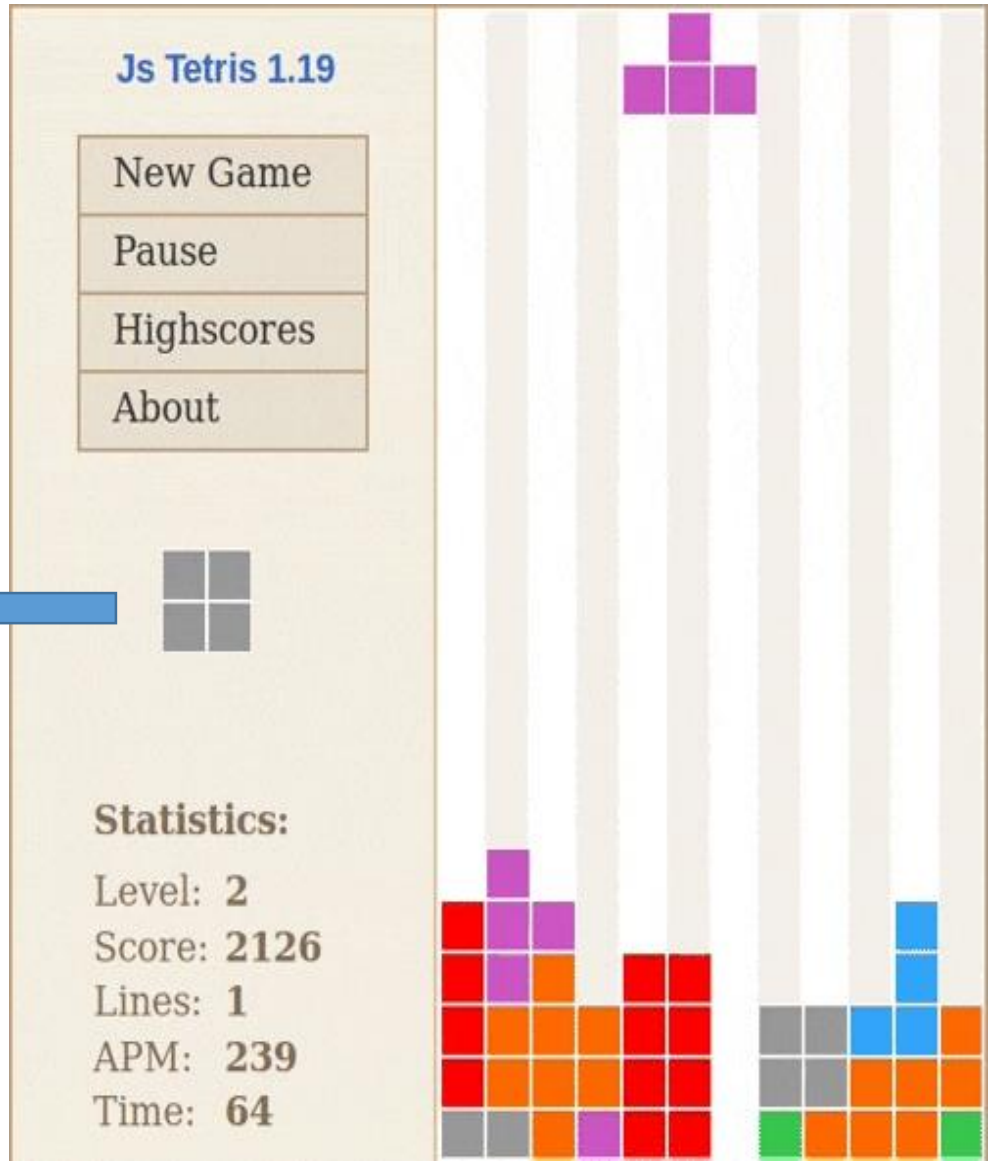
Important elements

"**Reinforcement learning (RL)** is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward"

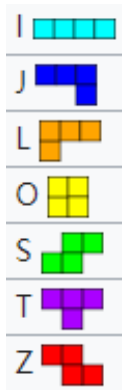
—— *Wikipedia*



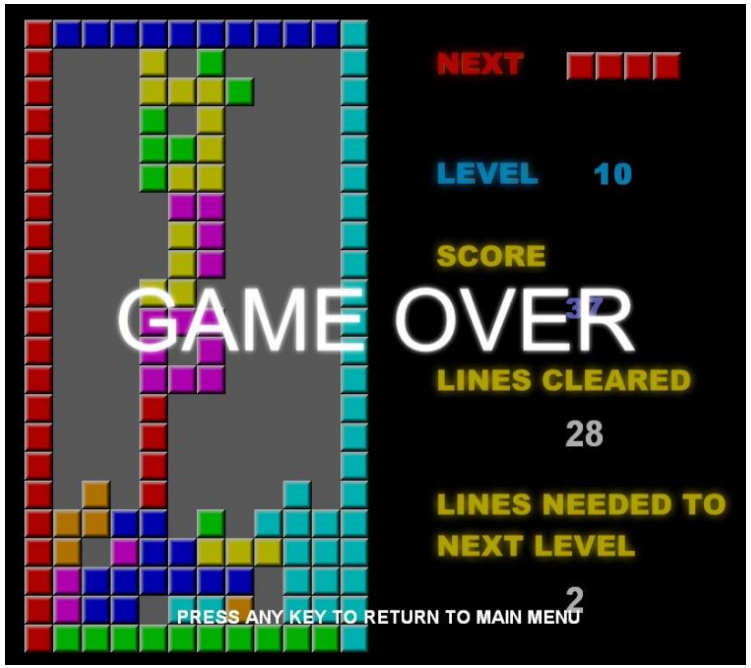
Tetris



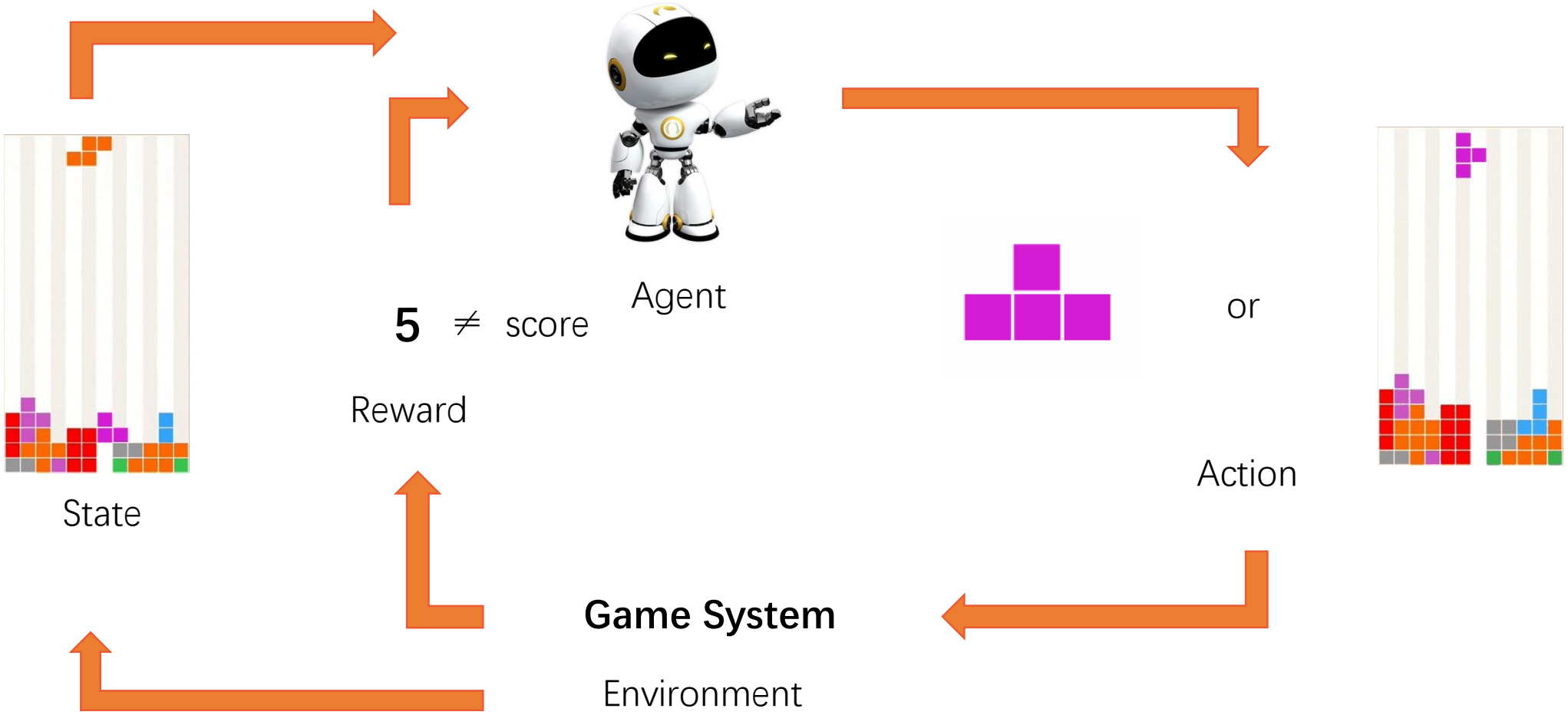
Tetromino (四格拼板)



The objective of the game is to manipulate these Tetriminos, by moving each one sideways (if the player feels the need) and rotating it by 90 degree units, with the aim of creating a horizontal line of ten units without gaps. When such a line is created, it gets cleared.



Important elements



The aim of RL is to learn a series decision which will get the best reward in the long run

Compared with familiar concepts

"**Reinforcement learning (RL)** is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward"

—— *Wikipedia*

Machine Learning :

- Supervised Learning
- Unsupervised Learning
- Reinforcement learning
- ...

	A	B	C	D	E
1	X1	X2	X2	X4	Label
2	5.1	3.5	1.4	0.2	1
3	4.9	3	1.4	0.2	0
4	4.7	3.2	1.3	0.2	0
5	4.6	3.1	1.5	0.2	1
6	5	3.6	1.4	0.2	0
7	5.4	3.9	1.7	0.4	2
8	4.6	3.4	1.4	0.3	2
9	5	3.4	1.5	0.2	0
10	4.4	2.9	1.4	0.2	1

The inputs and desired outputs of each instance should be given, and the goal is to learn a general rule that maps inputs to outputs.

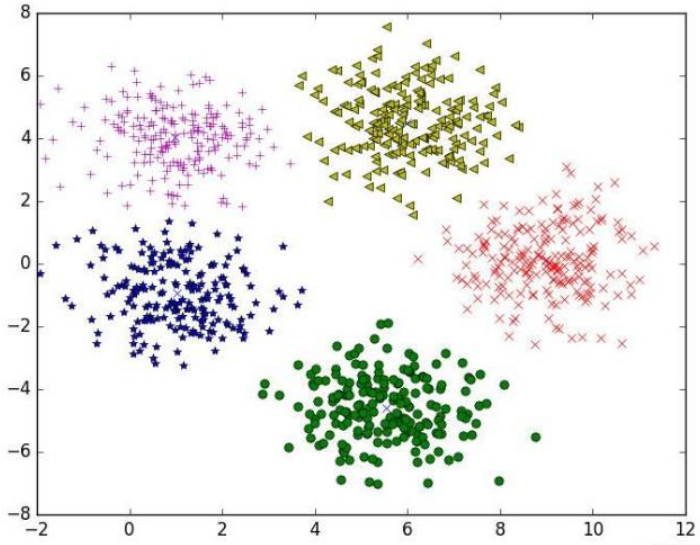
Compared with familiar concepts

"**Reinforcement learning (RL)** is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward"

—— *Wikipedia*

Machine learning :

- Supervised Learning
- Unsupervised Learning
- Reinforcement learning
- ...



No labels are given to the learning algorithm, leaving it on its own to find structure in its input.

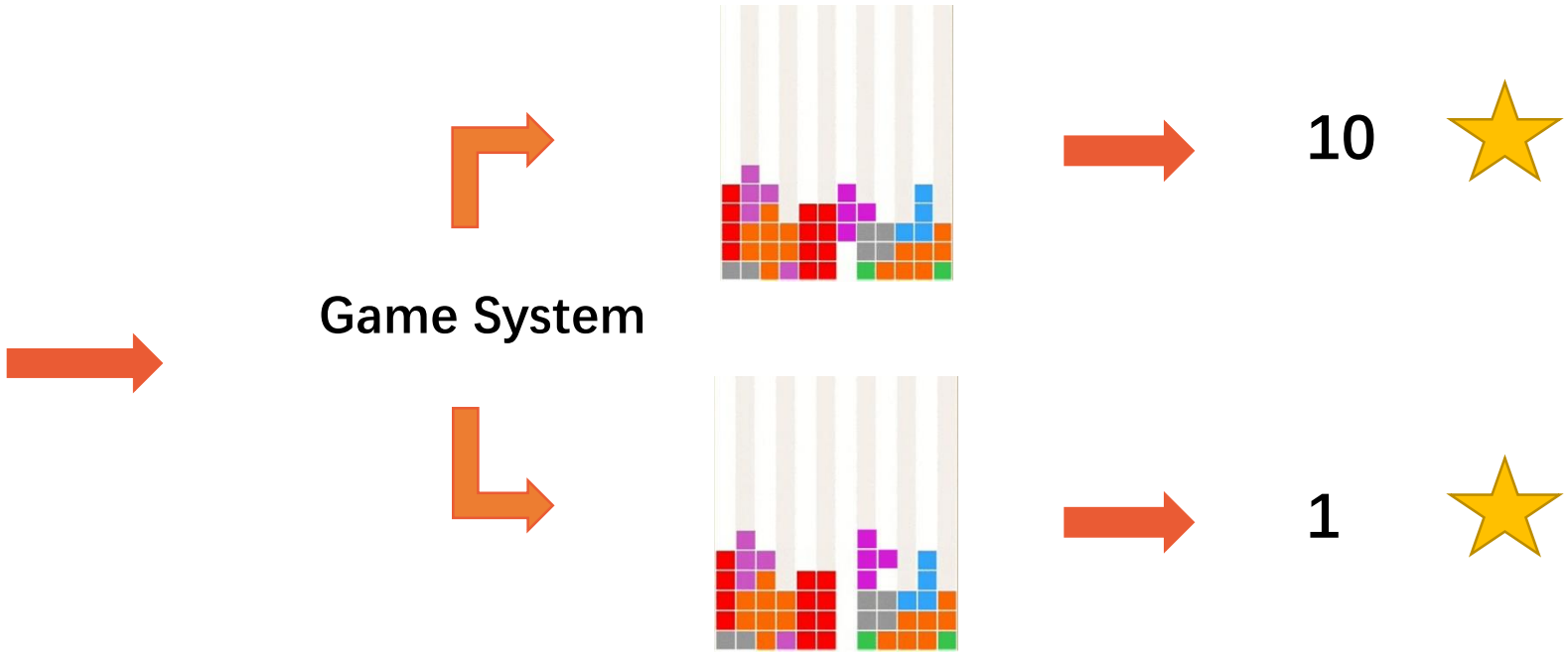
Compared with familiar concepts

"**Reinforcement learning (RL)** is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward"

—— *Wikipedia*

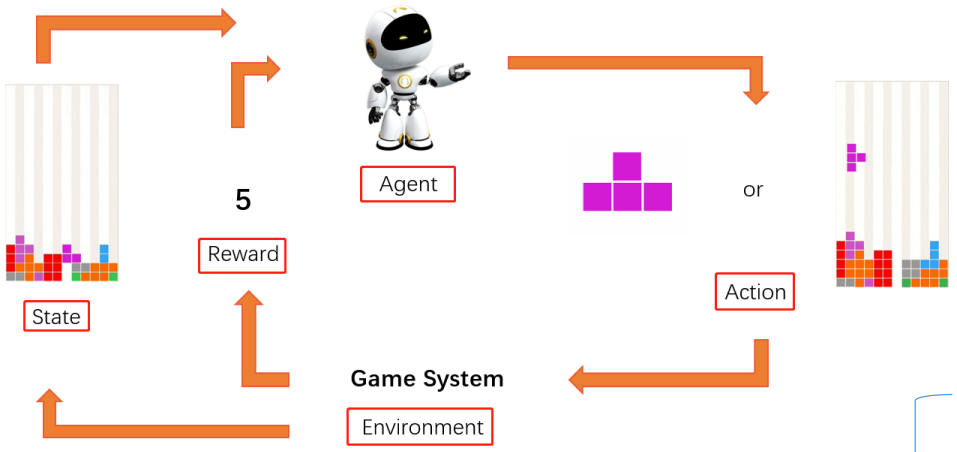
Machine learning :

- Supervised Learning
- Unsupervised Learning
- Reinforcement learning
- ...



The environment will give award to guide the change of model without the detail about how to do

Q-Learning



Input

- **Agent** → The goal of RL is to train the Agent to have the ability to play Tetris by itself.
- **Environment** → The game system which we need to set up includes interface building, formulating game rules, etc., it's generally built based on OpenAI's Gym.
- **Action** → The Agent must be told how many actions are available to choose from.
- **State** → Only when the Agent have known what's the current state, can it select the right action
- **Reward** → Is the only guiding information for the Agent to update.

The key point is how to give the most appropriate reward for each action that made by the agent under each state

Q-Learning

$$Q = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} \multicolumn{6}{c}{\text{Action}} \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$

The main idea of Q-learning is to learn this Q-table

Q-Learning

Q-table (State, Action)

Initialize $Q(s, a)$ arbitrarily

Repeat (for each episode):

Finish a game (loss or win)

Initialize s

Repeat (for each step of episode):

Choose a from s using policy derived from Q (e.g., ε-greedy)

Jump out of the best experience with a certain probability ε

Take action a, observe r, s'

r: Real rewards based on specified rules

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

s': The next state

$s \leftarrow s'$;

until s is terminal

Q-Learning

Learning rate

Discount factor

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

Real reward

The highest reward which corresponds to the action a' in the next state s'

Updat this Q-table:

Q(s,a)

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

Q(s',a')

$\gamma \max_{a'} Q(s',a')$

The bigger **γ**, the more emphasis on **past experience**

$r + \gamma \max_{a'} Q(s',a')$

Now we are in the state s , so this is the **imaginary Q value** when I take the acion a' in **the next state s'**

$\alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$

The larger **α** is, the larger the difference between the imaginary Q value in next state and Q(s,a)

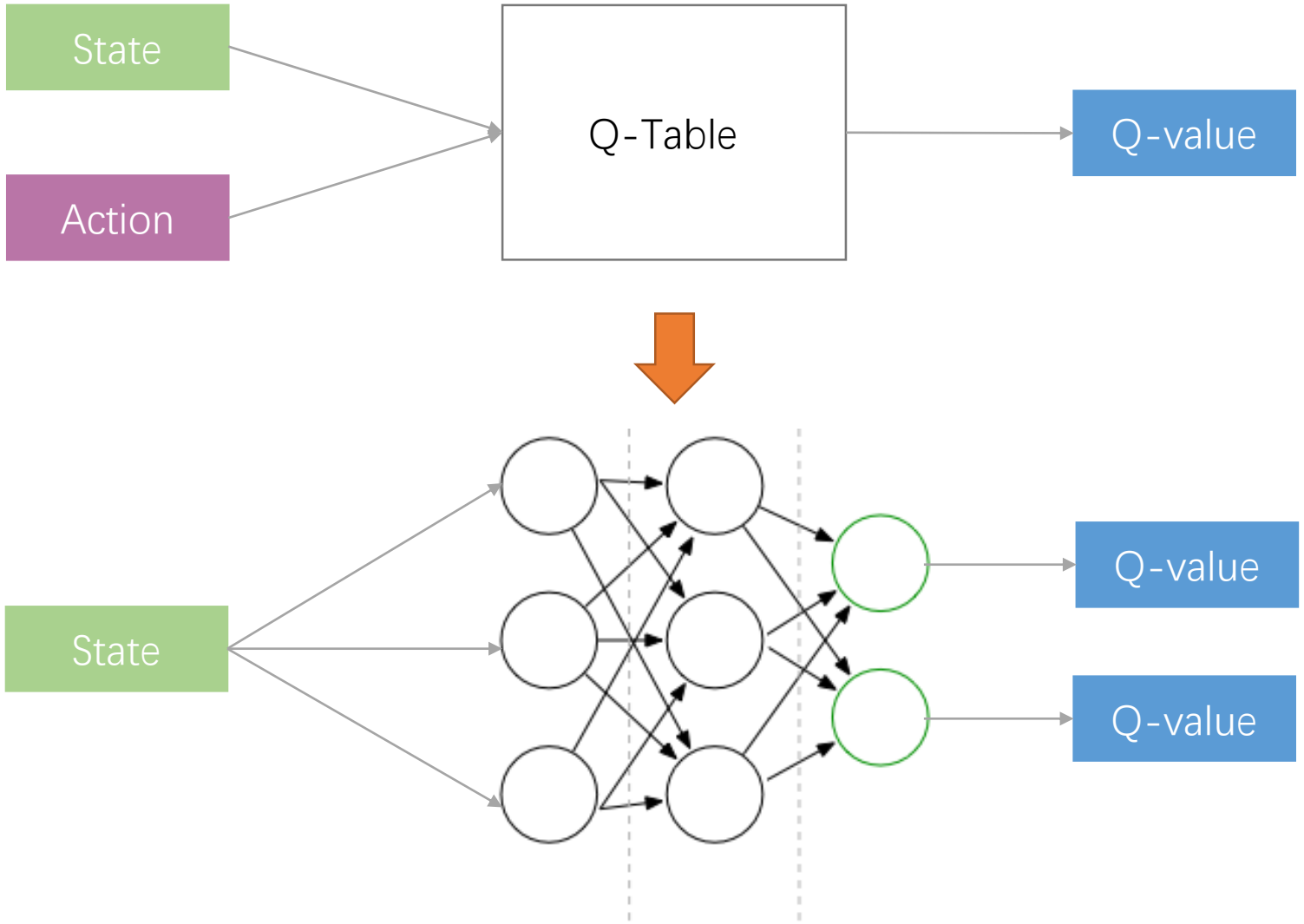
Q-Learning

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

...

The number of **state** is huge, which will cause the Q-table to be very large and bring the challenges of the storage and search.

Deep Q Network(DQN)



THANKS FOR

YOUR ATTENTION