The Introduction To Artificial Intelligence

Yuni Zeng yunizeng@zstu.edu.cn

2022-2023-1

The Introduction to Artificial Intelligence

- Part I Brief Introduction to AI & Different AI tribes
- Part II Knowledge Representation & Reasoning
- Part III AI GAMES and Searching
- Part IV Model Evaluation and Selection
- Part V Machine Learning

Machine Learning

Supervised learning

Unsupervised learning

Reinforcement learning

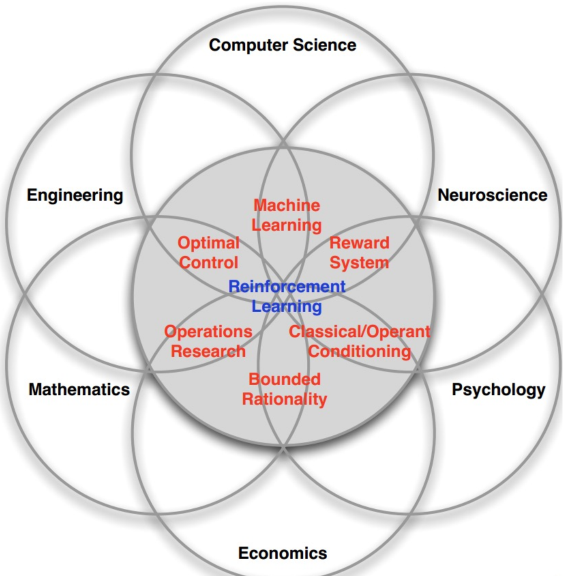
Introduction to Reinforcement learning

- Reinforcement learning
- Q-Learning

Reinforcement learning

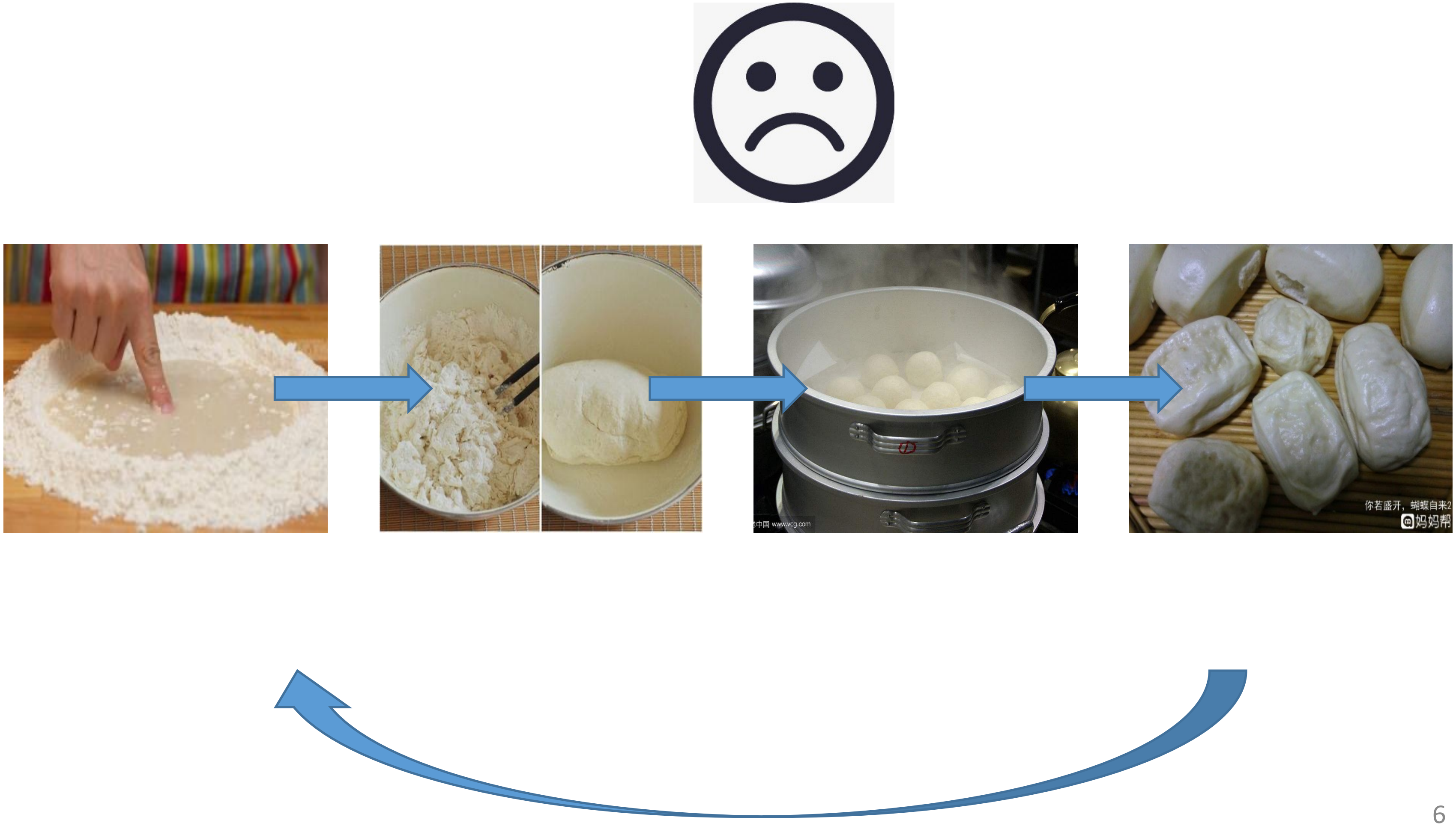
Reinforcement Learning

- "AI=RL" by David Silver
- Agent-oriented learning—learning by interacting with an environment to achieve a goal
- Learning by trial and error, with only delayed evaluative feedback (reward)



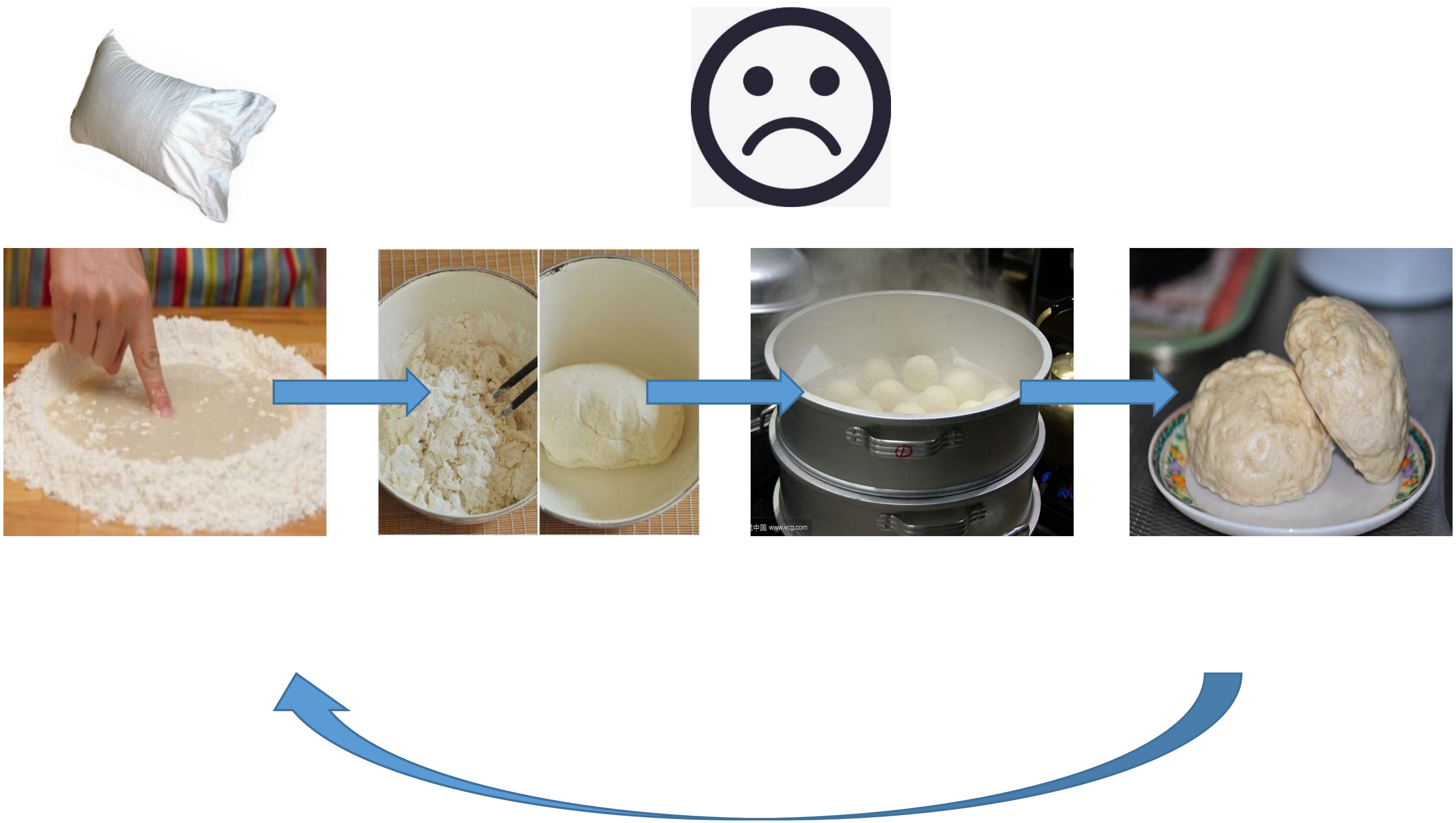
1. Different ML methods

- Reinforcement Learning -- example



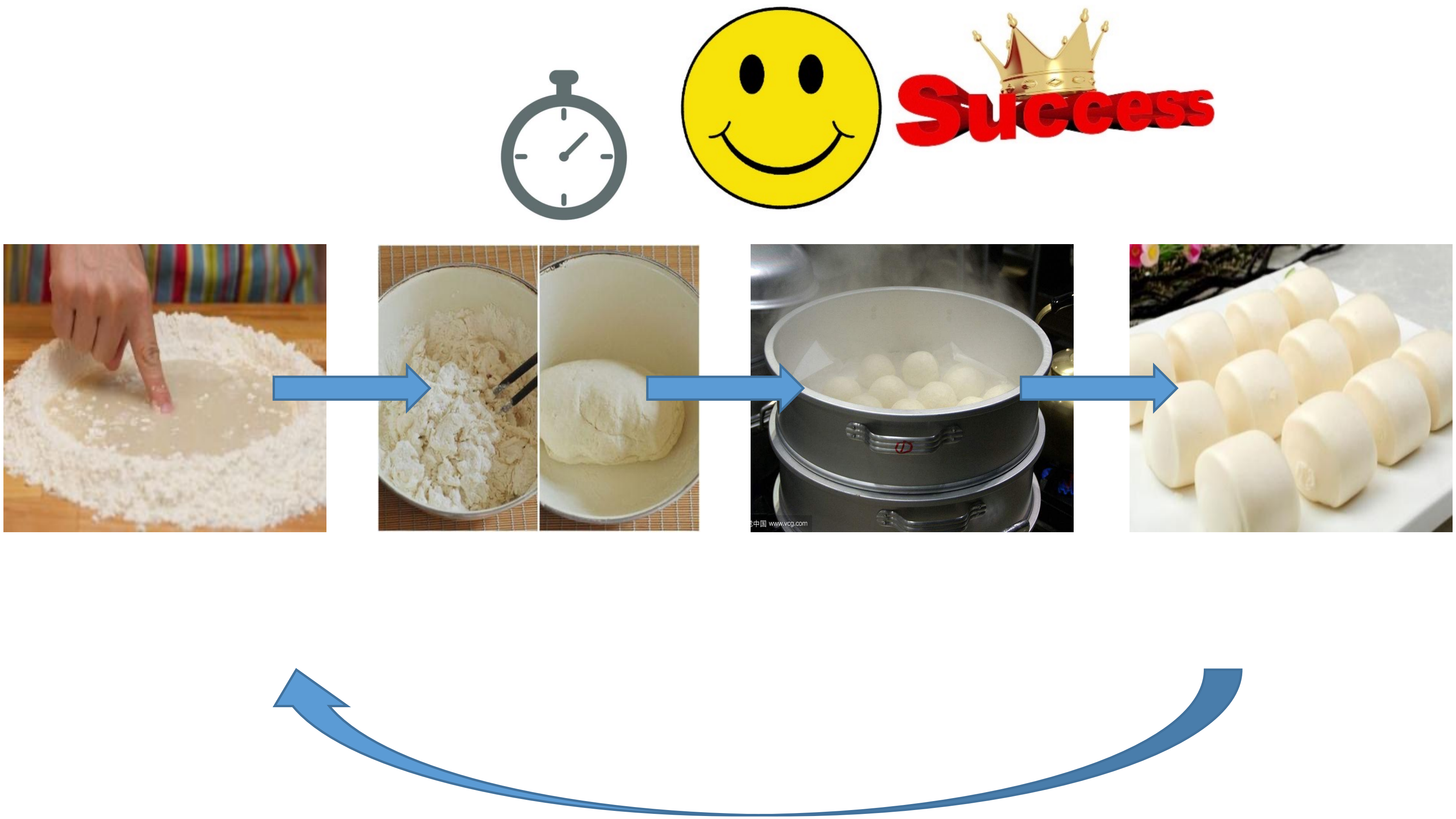
1. Different ML methods

Reinforcement Learning -- example



1. Different ML methods

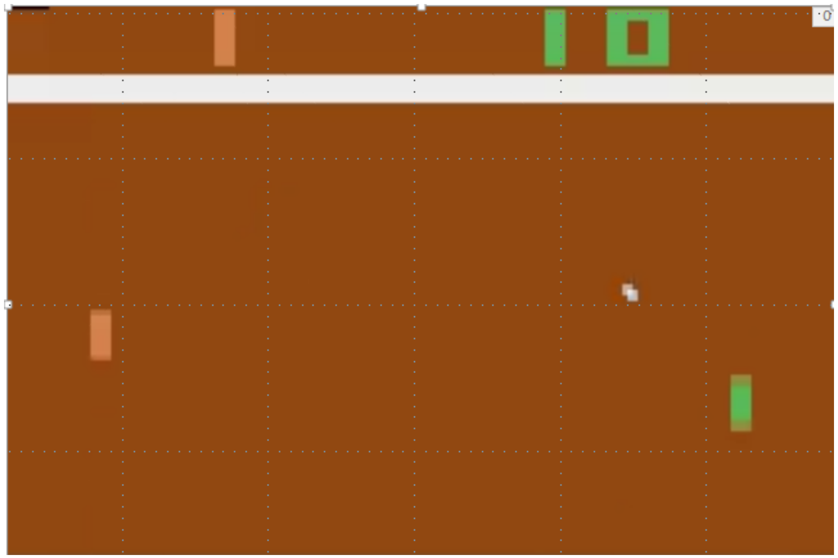
- Reinforcement Learning -- example



1. Different ML methods

Reinforcement Learning

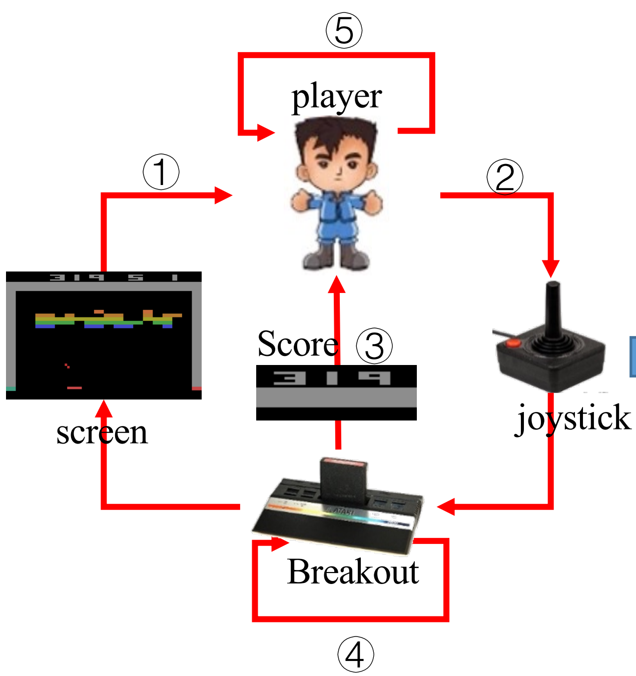
- **Game Pong**



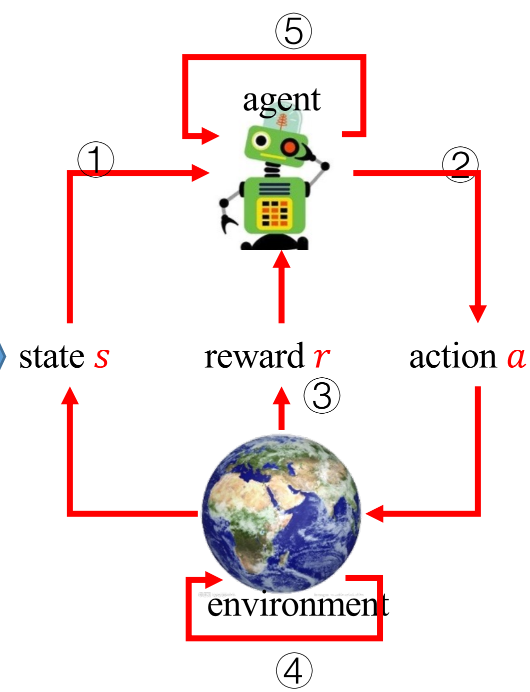
- **Game Breakout**

1. Different ML methods

Reinforcement Learning



Game Breakout



Reinforcement Learning

- Rules are unknown
- Learn directly from the interaction

At each time step t:

① Agent receives state $\boldsymbol{s}(\boldsymbol{t})$

② Agent executes an action $\boldsymbol{a}(\boldsymbol{t})$ by his action policy $\boldsymbol{\pi}(s(t))$

③ Environment emits a immediate reward $\boldsymbol{r}(\boldsymbol{t}+\boldsymbol{1})$ to agent

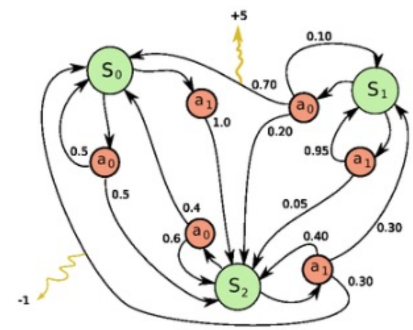
④ Environment changes its state to $\boldsymbol{s}(\boldsymbol{t}+\boldsymbol{1})$

⑤ Agent improves his policy $\boldsymbol{\pi}(\boldsymbol{s})$ according to the reward.

$$\begin{cases} < s, a, r, s' > \\ s \leftarrow s' \end{cases}$$

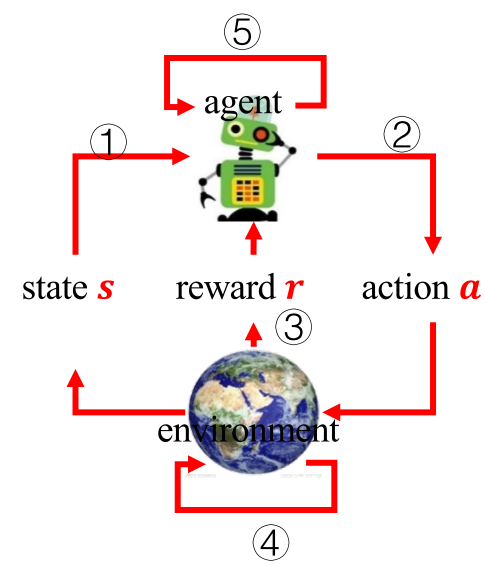
Reinforcement learning

- RL problem can be described as a Markov decision process
  - The future is independent of the past given the present
- One episode of this process forms a finite sequence :



$$s(0), a(0), r(1), s(1), a(1), r(2), \cdots\cdots, s(n-1), a(n-1), r(n), s(n)$$

$$\begin{cases} < s, a, r, s' > \\ s \leftarrow s' \end{cases}$$



- The agent are always trying to get the maximum rewards through policy $\pi(s)$

Question: How to define the maximum reward ?

Reinforcement learning

One episode of this process forms a finite sequence of states, actions, and rewards:

$$s(0), a(0), r(1), s(1), a(1), r(2), \cdots\cdots, s(n-1), a(n-1), r(n), s(n)$$

- **Total reward** of one episode:

$$R = r(1) + r(2) + r(3) + \cdots\cdots r(n-1) + r(n)$$

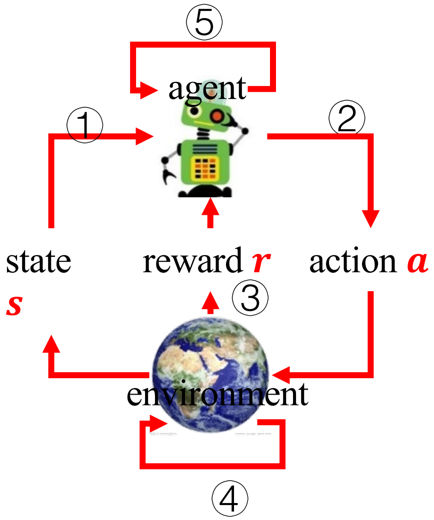
- **Total future reward** from time step t :

$$R(t)$$

$$= r(t) + r(t+1) + r(t+2) + \cdots\cdots r(n-1) + r(n)$$

- **Discounted future reward** reward from time step t :

$$R(t) = r(t) + \gamma r(t+1) + \gamma^2 r(t+2) + \cdots\cdots + \gamma^{n-t} r_{\mathrm{n}}$$



Question: How can agent get the maximum reward ?

Reinforcement learning

Question: How can agent get the maximum reward ?

$$R = r(1) + r(2) + r(3) + \cdots\cdots r(n-1) + r(n)$$
$$= \underbrace{r(1) + r(2) + r(3) + \cdots r(t}_{\text{past reward}} - 1) + R(t)$$

future reward

At each time step, a good strategy for an agent would be to **always choose an action that maximizes the (discounted) future reward.**

$$R(t)$$
$$= r(t) + \gamma r(t+1) + \gamma^2 r(t+2) + \cdots\cdots + \gamma^{n-t} r_{\mathrm{n}}(t)$$
$$= r(t) + \gamma R(t+1)$$

Introduction to Reinforcement learning

- Reinforcement learning
- Q-Learning

Q-Learning

- Q function represents the "quality" of a certain action in a given state.
- It is a table of states and actions.

$$Q(s(t), a(t)) = maxR(t+1)$$

Q-table

$Q[s,a]$	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\cdots$	$\boldsymbol{a_m}$
$\boldsymbol{s_1}$				
$\boldsymbol{s_2}$				
$\boldsymbol{s_3}$				
$\vdots$				
$\boldsymbol{s_n}$				

$$\pi(s(t)) = \max_a Q(s(t), a)$$

choose an action that maximizes the future reward.

Q-Learning

■ Bellman equation :

$$< s(t), a(t), r(t+1), s(t+1) >$$

$$Q(s(t), a(t)) = \max R(t+1)$$

$$Q\big(s(t), a(t)\big) = r(t+1) + \gamma \max R(t+2)$$

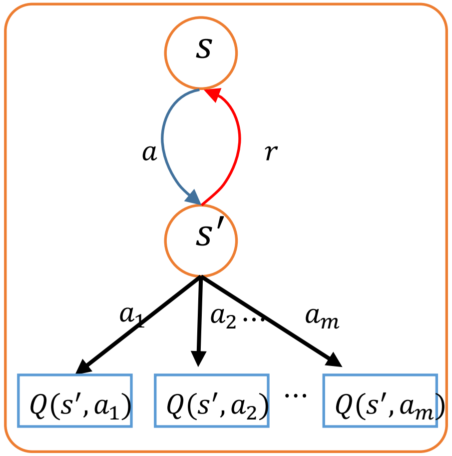
$$\boxed{R(t+1) = r(t+1) + \gamma R(t+2)}$$

$$Q\big(s(t), a(t)\big) = r(t+1) + \gamma \max_{a(t+1)} Q\big(s(t+1), a(t+1)\big)$$

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

current reward

maximum future reward from next state



Q-Learning

$$\begin{cases} < s, a, r, s' > \\ s \leftarrow s' \end{cases}$$

Q-table

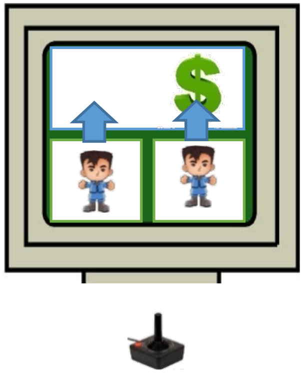
$Q[s,a]$	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\cdots$	$\boldsymbol{a_m}$
$\boldsymbol{s_1}$				
$\boldsymbol{s_2}$				
$\boldsymbol{s_3}$				
$\vdots$				
$\boldsymbol{s_n}$				

1. <u>Algorithm Q-Learning</u>
2. **Input:**
    1. S is a set of states
    2. A is a set of actions
    3. γ is the discount
3. initialize $Q[S, A]$ arbitrarily
4. observe initial state s
5. **Repeat:**
    1. select and carry out an action a, randomly
    2. receive reward r
    3. observe new state s'
    4. If s' is terminal state:
        1. $Q[s, a] = r$
    5. Else:
        1. $Q[s, a] = r + \gamma \max_{a'} Q[s', a']$
    6. $s \leftarrow s'$
6. **Until** terminated

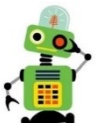
Q-Learning

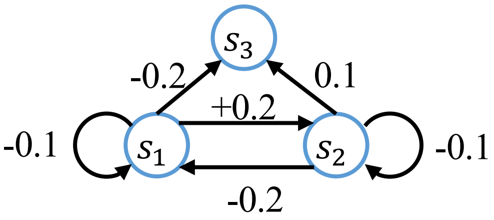
A tiny example:

Game description
States:
s_1, s_2, s_3, where s_3 is terminal state
Actions:
a_1 denotes *up*. The agent goes up and moves to terminal state.
a_2 denotes *left*. The agent moves to left in state s_2 with a reward -0.2, while stay still in state s_1 with a reward -0.1.
a_3 denotes *right*. The agent moves to right in state s_1 with a reward 0.2, while stay still in state s_2 with a reward -0.1.

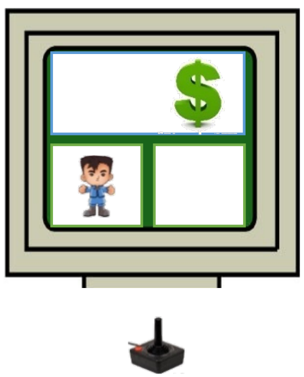


Move up/left/right





Q-Learning



Move up/left/right

$Q[s,a]$	$\boldsymbol{a_1}$ up	$\boldsymbol{a_2}$ left	$\boldsymbol{a_3}$ right
$\boldsymbol{s_1}$			
$\boldsymbol{s_2}$			
$\boldsymbol{s_3}$	-	-	-

<u>Algorithm Q-Learning</u>

Input:

S is a set of states

A is a set of actions

γ is the discount

initialize $Q[S,A]$ arbitrarily

observe initial state s

Repeat:

select and carry out an action a, randomly

receive reward r

observe new state s'

If s' is terminal state:

$Q[s,a] = r$

Else:

$Q[s,a] = r + \gamma \max_{a'} Q[s',a']$

$s \leftarrow s'$

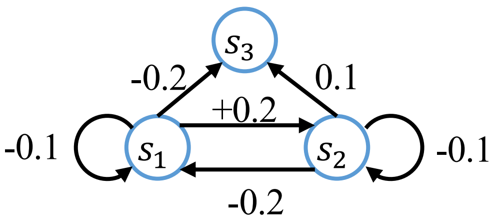
Until terminated

Q-Learning

- **Step 1:** initialize $Q[S, A]$

  $\gamma = 0.8$

$Q[s,a]$	$\boldsymbol{a_1}$ up	$\boldsymbol{a_2}$ left	$\boldsymbol{a_3}$ right
$\boldsymbol{s_1}$	0.60	0.74	0.94
$\boldsymbol{s_2}$	0.36	0.32	0.78
$\boldsymbol{s_3}$	-	-	-



- **Step 2:training loop**

1st episode:

$$\underbrace{s(0) = s_1, a(0) = a_3, r(1) = 0.2, s(1) = s_2}_{Q[s_1,a_3] = 0.2 + 0.8 * \max_{a_i}(Q[s_2,a_i]) = 0.2 + 0.8 * 0.78 = 0.82}, \underbrace{a(1) = a_3, r(2) = -0.1, s(2) = s_2}_{Q[s_2,a_3] = -0.1 + 0.8 * \max_{a_i}(Q[s_2,a_i]) = -0.1 + 0.8 * 0.78 = 0.52}, \underbrace{a(2) = a_1, r(3) = 0.1, s(3) = s_3}_{Q[s_2,a_1] = 0.1}$$

$Q[s_1, a_3] = 0.2 + 0.8 * \max_{a_i}(Q[s_2, a_i])$
$= 0.2 + 0.8 * 0.78$
$= 0.82$

$Q[s,a]$	$\boldsymbol{a_1}$ up	$\boldsymbol{a_2}$ left	$\boldsymbol{a_3}$ right
$\boldsymbol{s_1}$	0.60	0.74	0.82
$\boldsymbol{s_2}$	0.36	0.32	0.78
$\boldsymbol{s_3}$	-	-	-

$Q[s_2, a_3] = -0.1 + 0.8 * \max_{a_i}(Q[s_2, a_i])$
$= -0.1 + 0.8 * 0.78$
$= 0.52$

$Q[s,a]$	$\boldsymbol{a_1}$ up	$\boldsymbol{a_2}$ left	$\boldsymbol{a_3}$ right
$\boldsymbol{s_1}$	0.60	0.74	0.82
$\boldsymbol{s_2}$	0.36	0.32	0.52
$\boldsymbol{s_3}$	-	-	-

$Q[s_2, a_1] = 0.1$

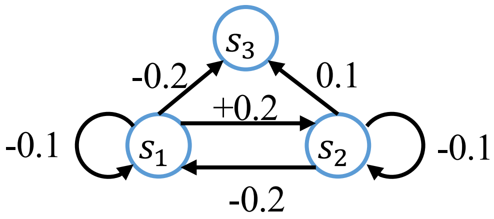
$Q[s,a]$	$\boldsymbol{a_1}$ up	$\boldsymbol{a_2}$ left	$\boldsymbol{a_3}$ right
$\boldsymbol{s_1}$	0.60	0.74	0.82
$\boldsymbol{s_2}$	0.1	0.32	0.52
$\boldsymbol{s_3}$	-	-	-

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

Q-Learning

After
11^{th} episode

$Q[s,a]$	a_1 up	a_2 left	a_3 right
s_1	-0.2	0.56	0.40
s_2	0.10	0.25	0.10
s_3	-	-	-



12^{st} episode:

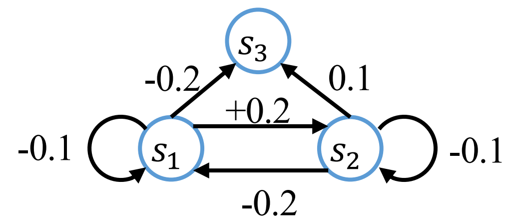
$$\underbrace{s(0) = s_1, a(0) = a_2, r(1) = -0.1, s(1) = s_1}_{Q[s_1,a_2] = -0.1 + 0.8 * \max_{a_i}(Q[s_1,a_i]) = -0.1 + 0.8 * 0.56 = 0.35} \quad \underbrace{, a(1) = a_1, r(2) = -0.2, s(2) = s_3}_{Q[s_2,a_1] = -0.2}$$

$Q[s,a]$	a_1 up	a_2 left	a_3 right
s_1	-0.20	0.35	0.40
s_2	0.10	0.25	0.10
s_3	-	-	-

$Q[s,a]$	a_1 up	a_2 left	a_3 right
s_1	-0.20	0.35	0.40
s_2	0.10	0.25	0.10
s_3	-	-	-

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

Q-Learning



After
15th episode

$Q[s,a]$	a_1 up	a_2 left	a_3 right
s_1	-0.20	0.18	0.30
s_2	0.10	0.08	-0.00
s_3	-	-	-

After
50th episode

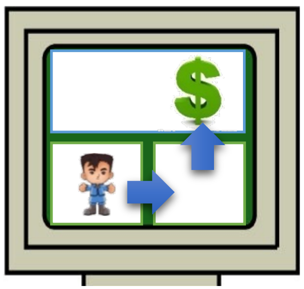
$Q[s,a]$	a_1 up	a_2 left	a_3 right
s_1	-0.20	0.12	0.28
s_2	0.10	0.02	-0.02
s_3	-	-	-

After
100th episode

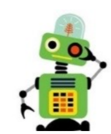
$Q[s,a]$	a_1 up	a_2 left	a_3 right
s_1	-0.20	0.12	0.28
s_2	0.10	0.02	-0.02
s_3	-	-	-

After
1000th episode

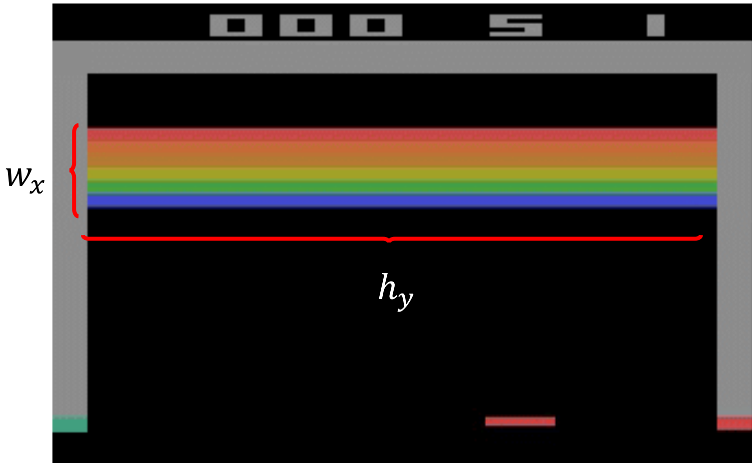
$Q[s,a]$	a_1 up	a_2 left	a_3 right
s_1	-0.20	0.12	0.28
s_2	0.10	0.02	-0.02
s_3	-	-	-



Move up/left/right



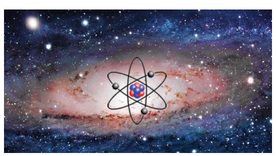
Q-Learning



Q-table

$Q[s,a]$	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\cdots$	$\boldsymbol{a_m}$
$\boldsymbol{s_1}$				
$\boldsymbol{s_2}$				
$\boldsymbol{s_3}$				
$\vdots$				
$\boldsymbol{s_n}$				

$$\begin{cases} < s, a, r, s' > \\ s \leftarrow s' \end{cases}$$



$< (3 * 256)^{w_x * h_y} < number\ of\ states$

Too huge states space to approximate Q-function iteratively by Q-table!!!

Conclusion – Machine Learning

1. Supervised Learning
   - Linear Regression
   - Logistic Regression
   - Classification
     - Distance-based algorithms
     - Linear classifiers
     - Other classifiers
2. Unsupervised Learning
   - Clustering
     - K-means method
     - Spectral clustering
   - Representation learning
3. Reinforcement Learning
   - Q-Learning, Q-table
   - Exploration & Exploitation