

Random Variable

- **Random variable:** a variable whose values depend on outcomes of a random event.
- Uppercase letter X for random variable.

<i>Random Variable</i>	<i>Possible Values</i>	<i>Random Events</i>	<i>Probabilities</i>
$X = \{$	0		$\mathbb{P}(X = 0) = 0.5$
	1		$\mathbb{P}(X = 1) = 0.5$

Random Variable

- **Random variable:** a variable whose values depend on outcomes of a random event.
- Uppercase letter X for random variable.
- Lowercase letter x for an observed value.
- For example, I flipped a coin 4 times and observed:
 - $x_1 = 1$,
 - $x_2 = 1$,
 - $x_3 = 0$,
 - $x_4 = 1$.

Probability Density Function (PDF)

- PDF provides a relative likelihood that the value of the random variable would equal that sample.

Example: Gaussian distribution

- It is a continuous distribution.
- PDF:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

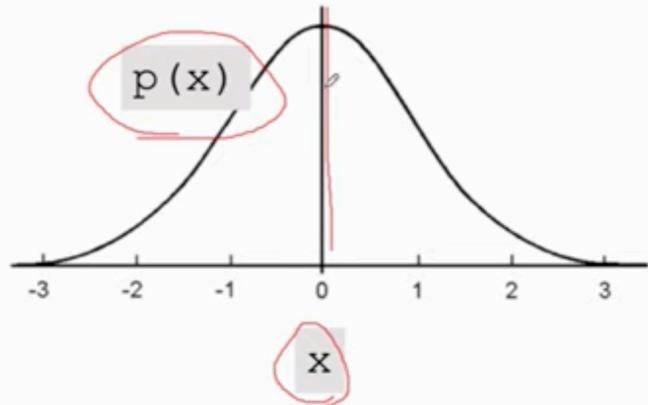
Probability Density Function (PDF)

- PDF provides a relative likelihood that the value of the random variable would equal that sample.

Example: Gaussian distribution

- It is a continuous distribution.
- PDF:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$



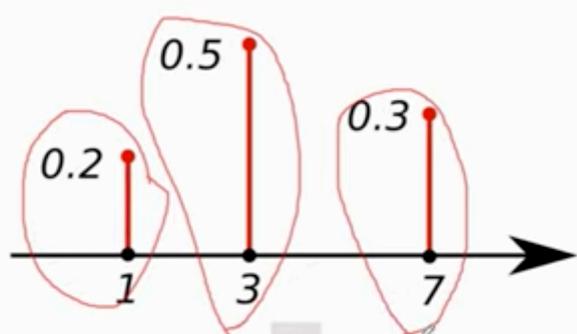
Probability Density Function (PDF)

- PDF provides a relative likelihood that the value of the random variable would equal that sample.

Example: Discrete distribution

- Discrete random variable: $X \in \{1, 3, 7\}$.
- PDF:

$$\begin{aligned}p(1) &= 0.2, \\p(3) &= 0.5, \\p(7) &= 0.3.\end{aligned}$$



Probability Density Function (PDF)

- Random variable X is in the domain $\underline{\mathcal{X}}$.

- For continuous distribution,

$$\int_{\underline{\mathcal{X}}} p(x) dx = \underline{1}.$$

- For discrete distribution,

$$\sum_{x \in \underline{\mathcal{X}}} p(x) = 1.$$

Expectation

- Random variable X is in the domain $\underline{\mathcal{X}}$.

- For continuous distribution, the expectation of $f(X)$ is:

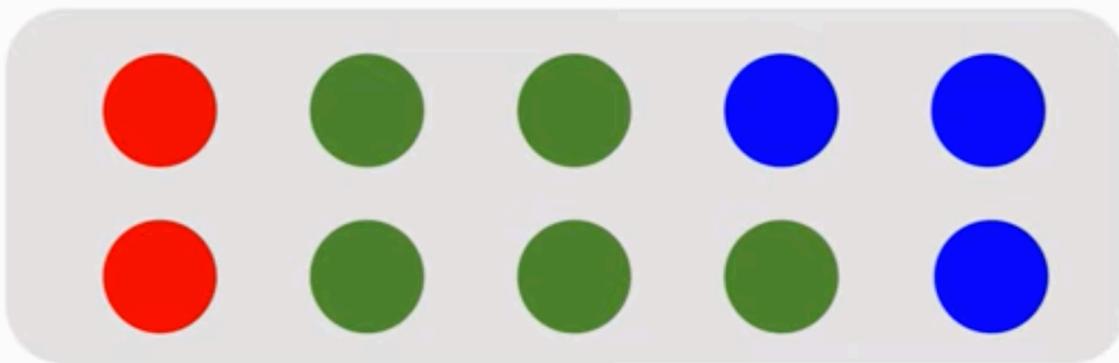
$$\mathbb{E}[f(X)] = \int_{\underline{\mathcal{X}}} p(x) \cdot f(x) dx.$$

- For discrete distribution, the expectation of $f(X)$ is:

$$\mathbb{E}[f(X)] = \sum_{x \in \underline{\mathcal{X}}} p(x) \cdot f(x).$$

Random Sampling

- There are 10 balls in a bin: 2 are red, 5 are green, and 3 are blue,
- Randomly sample a ball.
- What will be the outcome?



Random Sampling

- Sample **red ball w.p. 0.2**, **green ball w.p. 0.5**, and **blue ball w.p. 0.3**.
- Randomly sample a ball.
- What will be the outcome?

```
from numpy.random import choice
```

```
samples = choice(['R', 'G', 'B'], size=100, p=[0.2, 0.5, 0.3])  
print(samples)
```

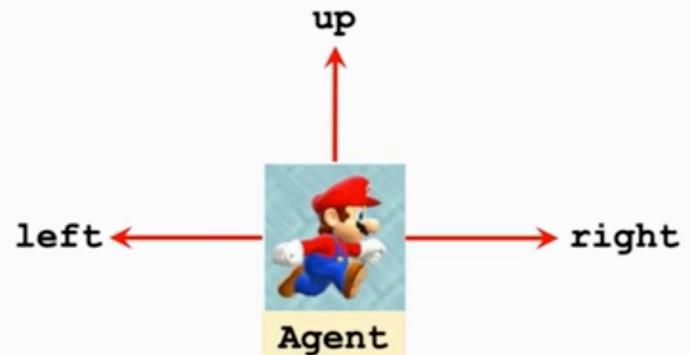
```
[ 'R' 'G' 'R' 'R' 'R' 'B' 'B' 'B' 'G' 'G' 'B' 'G' 'B' 'B' 'B' 'G' 'B'  
'B' 'B' 'G' 'B' 'G' 'B' 'G' 'B' 'B' 'G' 'B' 'G' 'B' 'G' 'B' 'G' 'B'  
'B' 'B' 'B' 'B' 'B' 'G' 'G' 'B' 'R' 'R' 'B' 'R' 'B' 'G' 'R' 'G' 'R'  
'R' 'R' 'B' 'G' 'G' 'B' 'R' 'G' 'B' 'G' 'R' 'G' 'G' 'G' 'B' 'B' 'R'  
'G' 'G' 'B' 'B' 'R' 'B' 'B' 'R' 'B' 'G' 'B' 'B' 'R' 'B' 'R' 'G' 'B'  
'B' 'B' 'G' 'G' 'R' 'R' 'B' 'R' 'G' 'B' 'R' 'B' 'G' 'B' 'R' 'B' 'R' ]
```

Terminology: state and action

state s (this frame)

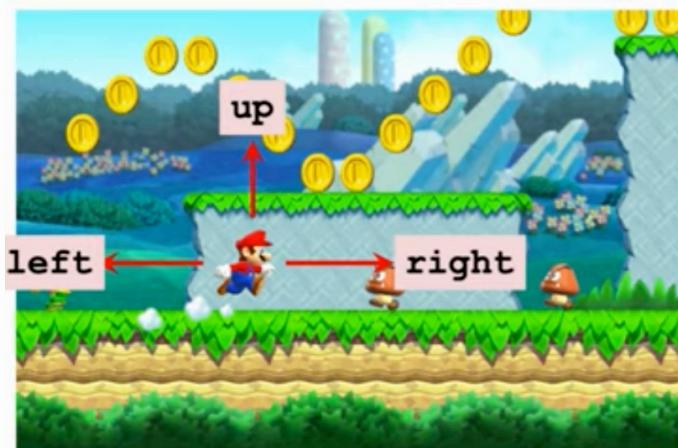


Action $a \in \{\text{left}, \text{right}, \text{up}\}$



Terminology: policy

policy π



- Policy function $\pi: (s, a) \mapsto [0,1]$:
 $\pi(a | s) = \mathbb{P}(A = a | S = s)$.
- It is the probability of taking action $A = a$ given state s , e.g.,
 - $\pi(\text{left} | s) = 0.2$,
 - $\pi(\text{right} | s) = 0.1$,
 - $\pi(\text{up} | s) = 0.7$,
- Upon observing state $S = s$, the agent's action A can be random.

Terminology: reward

reward R



- Collect a coin: $R = +1$
- Win the game: $R = +10000$
- Touch a Goomba: $R = -10000$ (game over).
- Nothing happens: $R = 0$

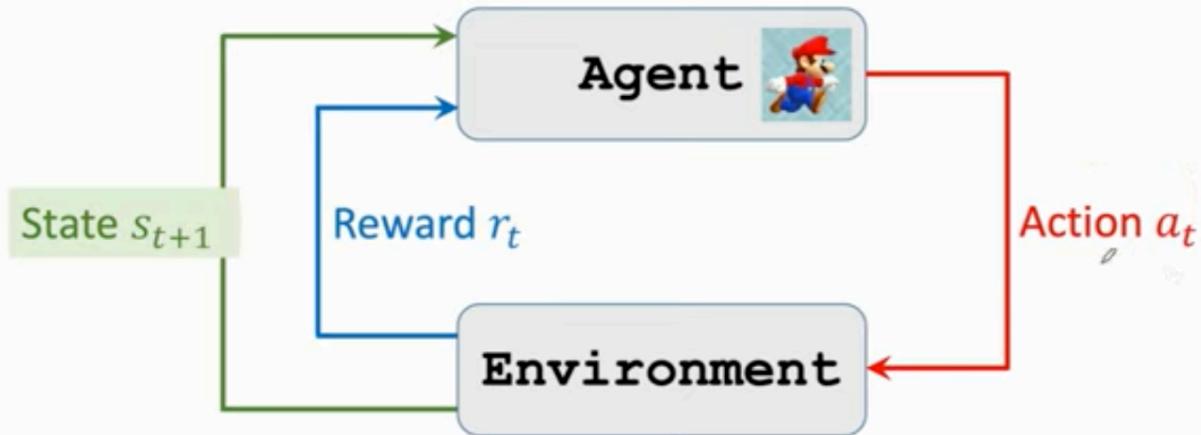
Terminology: state transition

state transition



- E.g., “up” action leads to a new state.
- State transition can be random.
- Randomness is from the environment.
- $\underline{p}(s'|s, a) = \mathbb{P}(S'_\theta = s' | S_\theta = s, A_\theta = a)$.

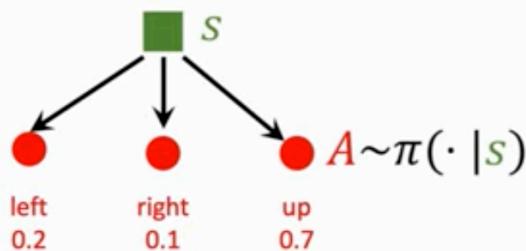
Terminology: agent environment interaction



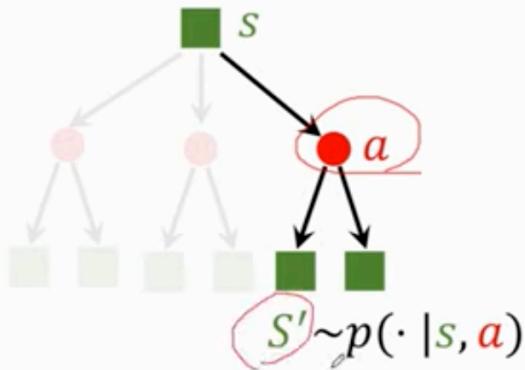
Randomness in Reinforcement Learning

Actions have randomness.

- Given state s , the action can be random, e.g.,
 - $\pi(\text{"left"}|s) = 0.2$,
 - $\pi(\text{"right"}|s) = 0.1$,
 - $\pi(\text{"up"}|s) = 0.7$.



Randomness in Reinforcement Learning



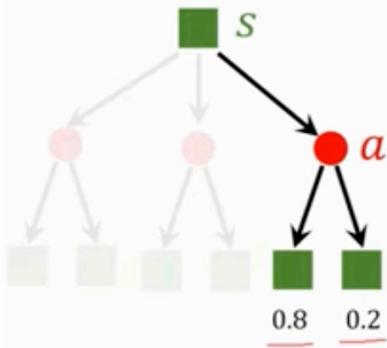
Actions have randomness.

- Given state s , the action can be random, e.g., .
 - $\pi(\text{"left"}|s) = 0.2$,
 - $\pi(\text{"right"}|s) = 0.1$,
 - $\pi(\text{"up"}|s) = 0.7$.

State transitions have randomness.

- Given state $S = s$ and action $A = a$, the environment randomly generates a new state S' .

Randomness in Reinforcement Learning



Actions have randomness.

- Given state s , the action can be random, e.g., .
 - $\pi(\text{"left"}|s) = 0.2$,
 - $\pi(\text{"right"}|s) = 0.1$,
 - $\pi(\text{"up"}|s) = 0.7$.

State transitions have randomness.

- Given state $S = s$ and action $A = a$, the environment randomly generates a new state S' .

Play the game using AI



- Observe a frame (state s_1)
- → Make action a_1 (left, right, or up)
- → Observe a new frame (state s_2) and reward r_1
- → Make action a_2
- → ...



- Observe a frame (state s_1)
 - → Make action a_1 (left, right, or up)
 - → Observe a new frame (state s_2) and reward r_1
 - → Make action a_2
 - → ...
- (state, action, reward) trajectory:
- $$s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_T, a_T, r_T.$$

Return

Definition: Return (aka cumulative future reward).

- $U_t = R_t + R_{t+1} + R_{t+2} + R_{t+3} + \dots$

Question: Are R_t and R_{t+1} equally important?

- Which of the followings do you prefer?
 - I give you \$100 right now.
 - I will give you \$100 one year later.
- Future reward is less valuable than present reward.
- $\underline{R_{t+1}}$ should be given less weight than R_t .

Return

Definition: Return (aka cumulative future reward).

- $U_t = R_t + R_{t+1} + R_{t+2} + R_{t+3} + \dots$

Definition: Discounted return (aka cumulative discounted future reward).

- $\underline{\gamma}$: discount rate (tuning hyper-parameter).
- $U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \dots$

Randomness in Returns

Definition: Discounted return (at time step t).

- $\underline{U_t} = \underline{R_t} + \gamma \underline{R_{t+1}} + \gamma^2 \underline{R_{t+2}} + \gamma^3 \underline{R_{t+3}} + \dots$

At time step t , the return U_t is **random**.

- Two sources of randomness:

1. Action can be random: $\mathbb{P}[A = a | S = s] = \pi(a|s)$.
2. New state can be random: $\mathbb{P}[S' = s' | S = s, A = a] = p(s'|s, a)$.

Randomness in Returns

Definition: Discounted return (at time step t).

- $\underline{U_t} = \underline{R_t} + \gamma \underline{R_{t+1}} + \gamma^2 \underline{R_{t+2}} + \gamma^3 \underline{R_{t+3}} + \dots$

At time step t , the return U_t is **random**.

- Two sources of randomness:

1. Action can be random: $\mathbb{P}[A = a | S = s] = \pi(a|s)$.
2. New state can be random: $\mathbb{P}[S' = s' | S = s, A = a] = p(s'|s, a)$.

- For any $i \geq t$, the reward R_i depends on S_i and A_i .

- Thus, given s_t , the return $\underline{U_t}$ depends on the random variables:
 - $A_t, A_{t+1}, A_{t+2}, \dots$ and S_{t+1}, S_{t+2}, \dots

Action-Value Function $Q(s, a)$

Definition: Discounted return (aka cumulative discounted future reward).

- $U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \dots$

Definition: Action-value function for policy π .

- $Q_\pi(s_t, a_t) = \mathbb{E}[U_t | S_t = s_t, A_t = a_t].$
- Return U_t depends on actions $A_t, A_{t+1}, A_{t+2}, \dots$ and states $S_t, S_{t+1}, S_{t+2}, \dots$

Action-Value Function $Q(s, a)$

Definition: Discounted return (aka cumulative discounted future reward).

- $U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \dots$

Definition: Action-value function for policy π .

- $Q_\pi(s_t, a_t) = \mathbb{E}[U_t | S_t = s_t, A_t = a_t].$
- Return U_t depends on actions $A_t, A_{t+1}, A_{t+2}, \dots$ and states $S_t, S_{t+1}, S_{t+2}, \dots$
- Actions are random: $\mathbb{P}[A = a | S = s] = \pi(a|s)$. (Policy function.)
- States are random: $\mathbb{P}[S' = s' | S = s, A = a] = p(s'|s, a)$. (State transition.)

Action-Value Function $Q(s, a)$

Definition: Discounted return (aka cumulative discounted future reward).

- $U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \dots$

Definition: Action-value function for policy π .

- $Q_{\underline{\pi}}(s_t, a_t) = \mathbb{E}[U_t | S_t = s_t, A_t = a_t]$.

Definition: Optimal action-value function.

- $Q^*(s_t, a_t) = \max_{\pi} Q_{\pi}(s_t, a_t)$.

State-Value Function $V(s)$

Definition: Discounted return (aka cumulative discounted future reward).

- $U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \dots$

Definition: Action-value function for policy π .

- $Q_{\pi}(s_t, a_t) = \mathbb{E}[U_t | S_t = s_t, A_t = a_t]$.

Definition: State-value function.

- $\underline{V}_{\pi}(s_t) = \mathbb{E}_{\underline{A}}[Q_{\pi}(s_t, \underline{A})]$

State-Value Function $V(s)$

Definition: Discounted return (aka cumulative discounted future reward).

- $U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \dots$

Definition: Action-value function for policy π .

- $Q_\pi(s_t, a_t) = \mathbb{E}[U_t | S_t = s_t, A_t = a_t].$

Definition: State-value function.

- $V_\pi(s_t) = \mathbb{E}_A [Q_\pi(s_t, A)] = \sum_a \pi(a|s_t) \cdot Q_\pi(s_t, a).$ (Actions are discrete.)



Taken w.r.t. the action $A \sim \pi(\cdot | s_t).$

State-Value Function $V(s)$

Definition: Discounted return (aka cumulative discounted future reward).

- $U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \dots$

Definition: Action-value function for policy π .

- $Q_\pi(s_t, a_t) = \mathbb{E}[U_t | S_t = s_t, A_t = a_t].$

Definition: State-value function.

- $V_\pi(s_t) = \mathbb{E}_A [Q_\pi(s_t, A)] = \sum_a \pi(a|s_t) \cdot Q_\pi(s_t, a).$ (Actions are discrete.)

- $V_\pi(s_t) = \mathbb{E}_A [Q_\pi(s_t, A)] = \int \pi(a|s_t) \cdot Q_\pi(s_t, a) da.$ (Actions are continuous.)

Understanding the Value Functions

- Action-value function: $Q_\pi(s_t, a_t) = \mathbb{E}[U_t | S_t = s_t, A_t = a_t]$.
- For policy π , $Q_\pi(s, a)$ evaluates how good it is for an agent to pick action a while being in state s .
- State-value function: $V_\pi(s) = \mathbb{E}_{\underline{A}}[Q_\pi(s, A)]$
- For fixed policy π , $V_\pi(s)$ evaluates how good the situation is in state s .
- $\mathbb{E}_S[V_\pi(s)]$ evaluates how good the policy π is.

How does AI control the agent?

Suppose we have a good policy $\pi(a|s)$.

- Upon observe the state s_t ,
- random sampling: $a_t \sim \pi(\cdot | s_t)$.

Suppose we know the optimal action-value function $\underline{Q^*(s, a)}$.

- Upon observe the state s_t ,
- choose the action that maximizes the value: $a_t = \operatorname{argmax}_a Q^*(s_t, a)$.

OpenAI Gym

- Gym is a toolkit for developing and comparing reinforcement learning algorithms.
- <https://gym.openai.com/>

Classical control problems



Cart Pole



Pendulum

OpenAI Gym

- Gym is a toolkit for developing and comparing reinforcement learning algorithms.
- <https://gym.openai.com/>

Atari Games



Pong



Space Invader

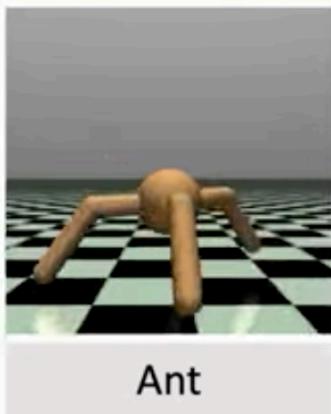


Breakout

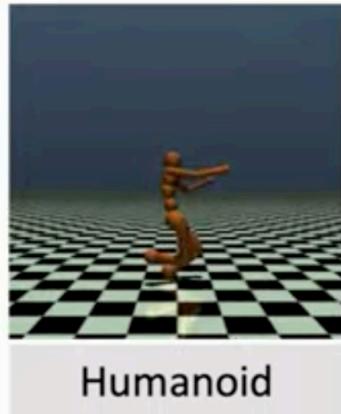
OpenAI Gym

- Gym is a toolkit for developing and comparing reinforcement learning algorithms.
- <https://gym.openai.com/>

MuJoCo (Continuous control tasks.)



Ant



Humanoid



Half Cheetah

Play CartPole Game

```
import gym  
env = gym.make('CartPole-v0')
```

- Get the environment of CartPole from Gym.
- “env” provides states and reward.



Play CartPole Game

```
state = env.reset()  
  
for t in range(100):  
    env.render() → A window pops up rendering CartPole.  
    print(state) → A random action.  
  
    action = env.action_space.sample()  
    state, reward, done, info = env.step(action)  
  
    if done: "done=1" means finished (win or lose the game)  
        print('Finished')  
        break  
  
env.close()
```

Summary

Terminologies

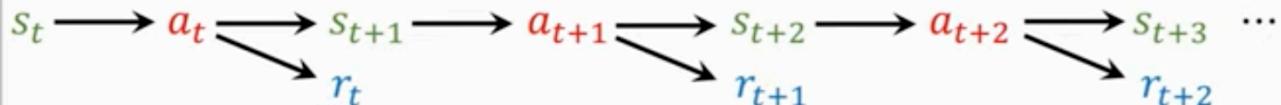
- Agent 
- Environment
- State s .
- Action a .
- Reward r .
- Policy $\pi(a|s)$
- State transition $p(s'|s, a)$.

Return and Value

- Return:
$$U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots$$
- Action-value function:
$$Q_\pi(s_t, a_t) = \mathbb{E}[U_t | s_t, a_t].$$
- Optimal action-value function:
$$Q^\star(s_t, a_t) = \max_\pi Q_\pi(s_t, a_t).$$
- State-value function:
$$V_\pi(s_t) = \mathbb{E}_A [Q_\pi(s_t, A)].$$

Play game using reinforcement learning

- Observe state s_t , make action a_t , environment gives s_{t+1} and reward r_t .



- The agent can be controlled by either $\underline{\pi(a|s)}$ or $\underline{\underline{Q^*(s,a)}}$.

We are going to study...

2. Value-based learning.

- Deep Q network (DQN) for approximating $Q^*(s, a)$.
- Learn the network parameters using temporal difference (TD).

3. Policy-based learning.

- Policy network for approximating $\pi(a|s)$.
- Learn the network parameters using policy gradient.

4. Actor-critic method. (Policy network + value network.)

5. Example: AlphaGo