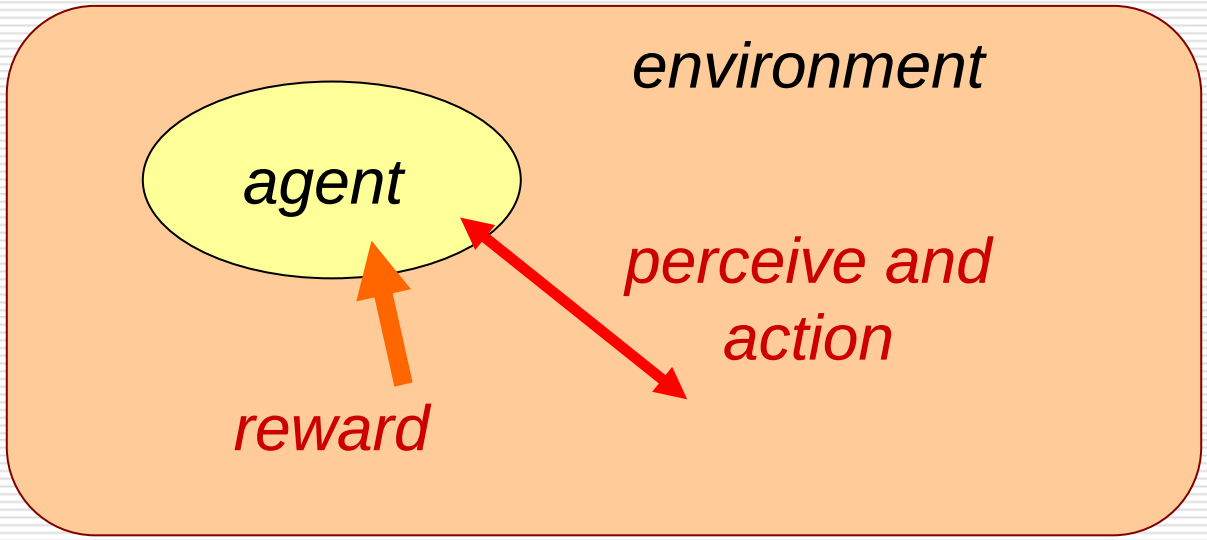
Evolutionary Optimization

Reinforcement Learning

Framework

Reinforcement learning

is one of the unsupervised learning methods.



It learns based only on reward from the environment

Agent and Environment

An agent acts in an environment

state transition

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{u-1}} s_u \xrightarrow{a_u} \cdots$$

$s_t \in S$: state of agent at t

$a_t \in A$: action of agent at t

$$env : S^* \times A \rightarrow P(S)$$

$P(S)$: power set of S

$$action : S^* \rightarrow A$$

Agent and Environment

Markov property

If an environment has markov property,
the current environment has the information on
all of the past histories: $s_t, a_t \rightarrow s_{t+1}$

$$env : S \times A \rightarrow P(S)$$

$$action : S \rightarrow A$$

Policy

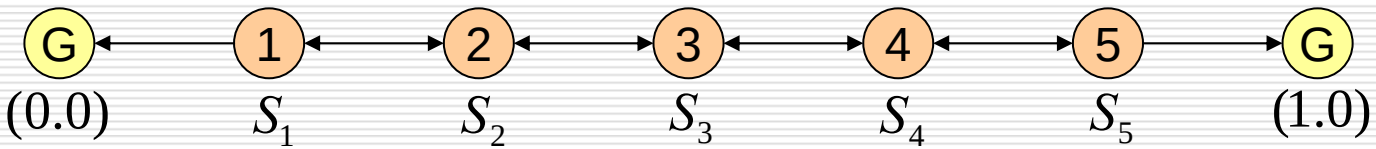
policy … A mapping from state to probability of selecting an action.

$\pi(s,a)$: probability of selecting action ***a*** at state ***s***

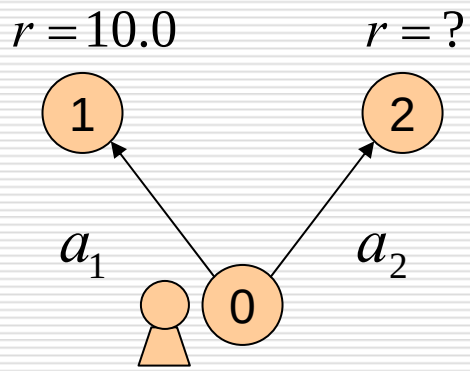
Example.

$a = \text{move right, move left}$

$\pi(s,a) = 1/2 \; (\forall s \in S, \forall a \in A)$ ⟶ random policy



Exploitation and Exploration



2-armed bandit problem

- exploitation

  To select a1, because the expected reward is known.

- exploration

  To select a2 in order to obtain new information.

exploitation ⟷ exploration

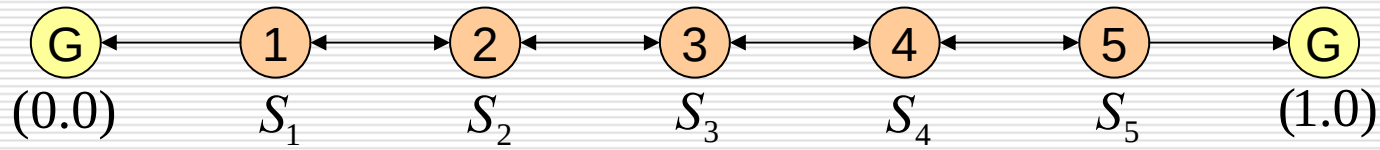
trade-off

State-Value Function

State-value function

… function representing the desirability of the state ; defined as an expected reward obtained from the state.

Example.



π_1 : move random

$$V^{\pi_1}(s_1) = 1/6, V^{\pi_1}(s_2) = 2/6, \ldots, V^{\pi_1}(s_5) = 5/6$$

π_2 : move right only

$$V^{\pi_2}(s_1) = V^{\pi_2}(s_2) = \cdots = V^{\pi_2}(s_5) = 1.0$$

Discounting Expected Reward

Discounting expected reward

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-1} r_T = \sum_{k=1}^{T-t} \gamma^{k-1} r_{t+k}$$

r_t : reward from s_t

T : step of episode

γ : discount rate $\quad 0 \le \gamma \le 1$

This allows the agent to learn and obtain a lot of reward with minimal steps.

TD Learning

Temporal difference learning

TD learning can obtain a precise state-value function

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha(R_t - V^{\pi}(s_t)) \qquad \alpha : \text{learning rate}$$

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha(r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t))$$

$$V^{\pi}(s_t) = E_{\pi}\{R_t \mid s_t\}$$

$$= E_{\pi}\left\{\sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k} \mid s_t\right\}$$

$$= E_{\pi}\left\{r_{t+1} + \gamma \sum_{k=1}^{\infty} \gamma^{k-1} r_{r+k+1} \mid s_t\right\}$$

$$= E_{\pi}\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t\}$$

Action-Value Function and Q-learning

Action-value function

$Q^{\pi}(s_t, a_t)$: value of action ***a*** at state ***s*** with policy ***π***

Optimal action-value function

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a) \quad (\forall s \in S, \forall a \in A)$$

$$Q^{*}(s_t, a_t) = r_{t+1} + \gamma \max_{a_{t+1} \in A} Q^{*}(s_{t+1}, a_{t+1})$$

Action-Value Function and Q-learning

Q-learning

It updates *Q(s,a)* to obtain an optimal *Q*(s,a)* using the following equation.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

When all *Q* values are converged, the agent has an optimal policy.

Action-Value Function and Q-learning

Policies used for Q-learning

- greedy policy

$$a = \arg\max_{a \in A} Q(s,a)$$

- **ε** greedy policy

probability **ε** → random policy

probability 1- **ε** → greedy policy

- Boltzman distribution

$$\pi(s,a) = \frac{\exp(Q(s,a)/T)}{\sum_{b \in A} \exp(Q(s,b)/T)}$$