Introduction of Reinforcement Learning

KEISUKE FUKUTA

HARADA USHIKU LAB.

Outline

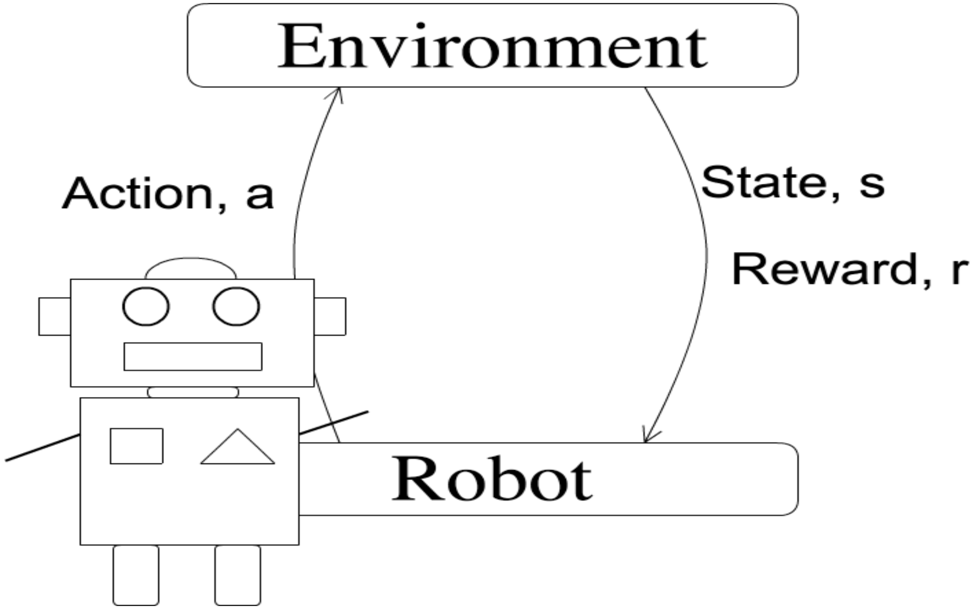
1. What is RL
2. Classical RL algorithm to DQN
3. Recent DRL

What is Reinforcement Learning

Learning paradigm that agent try to adapt to unknown environment

There is no superviser, only a reward signal

→ Learn through try-and-error



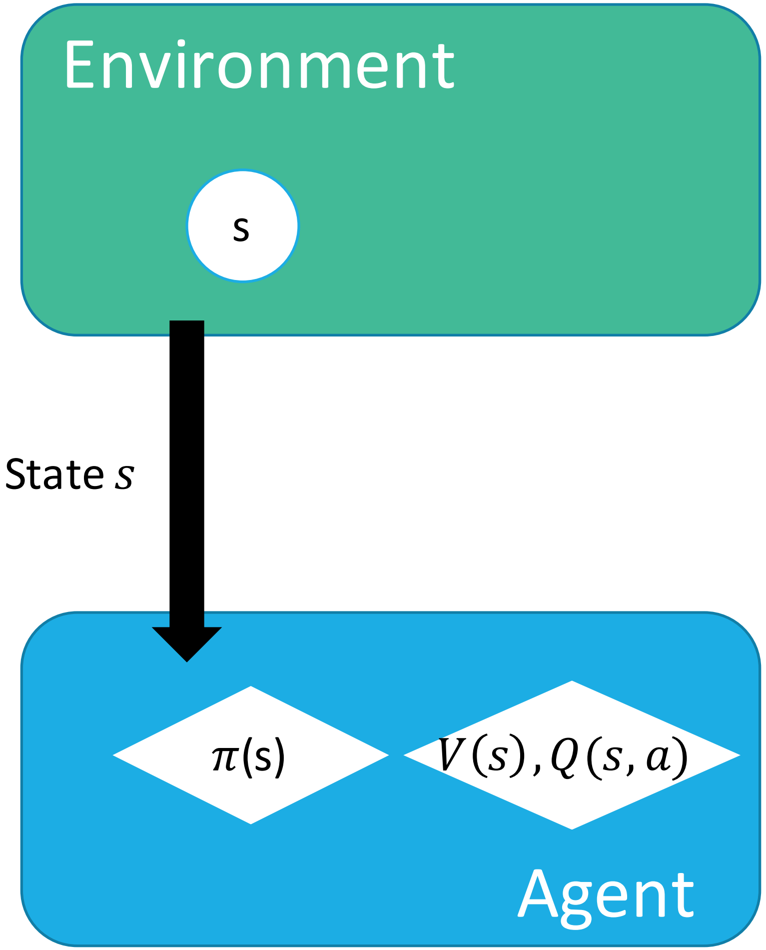
DQN

- https://www.youtube.com/watch?v=V1eYniJ0Rnk

What is RL

- At each Time Step $t = 0, 1, \ldots$

1. Agent observe $s_t \in S$



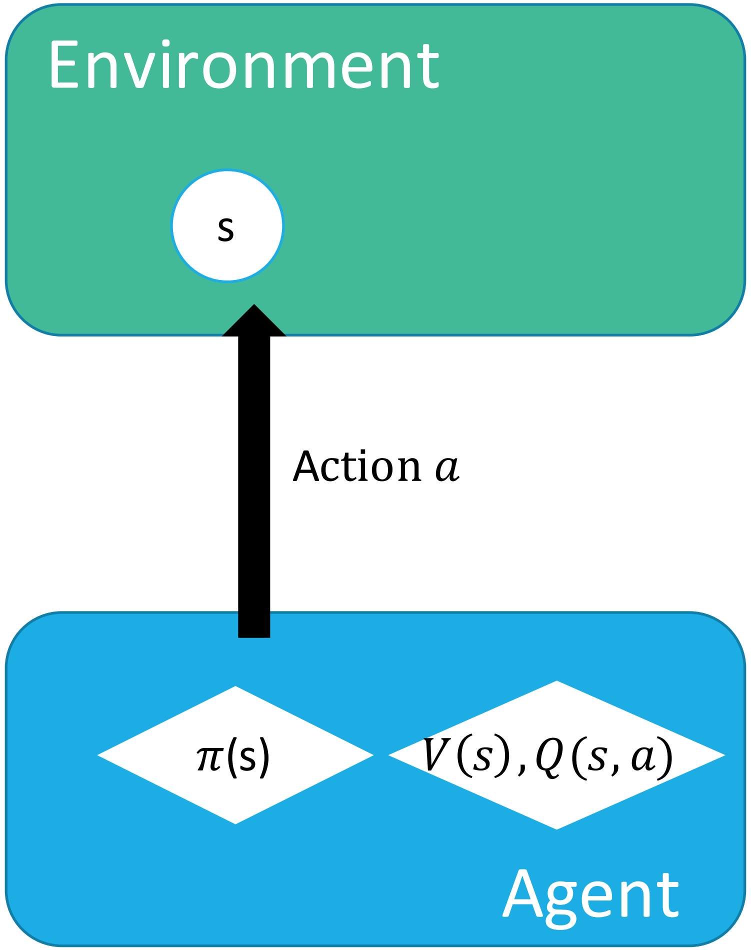
What is RL

- At each Time Step $t = 0, 1, \ldots$

1. Agent observe $s_t \in S$
2. Choose action $a_t \in A$ conditioned on s_t using $\pi(s)$

Policy $\pi(s)$

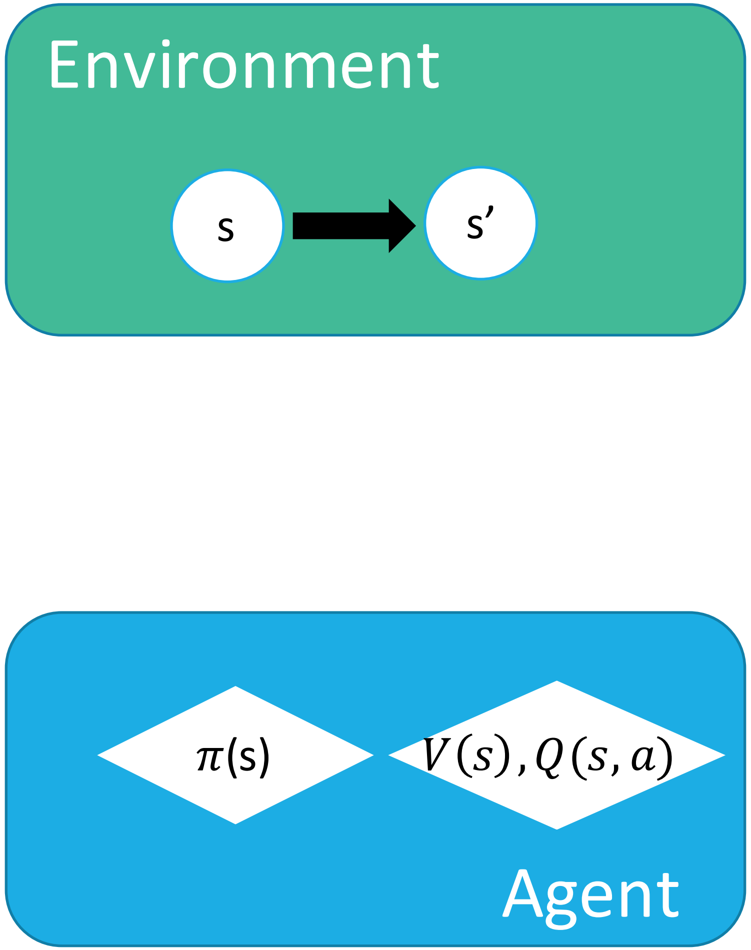
Agent's behaviour function
Mapping state to action $\pi : S \rightarrow A$



What is RL

- At each Time Step $t = 0, 1, \ldots$

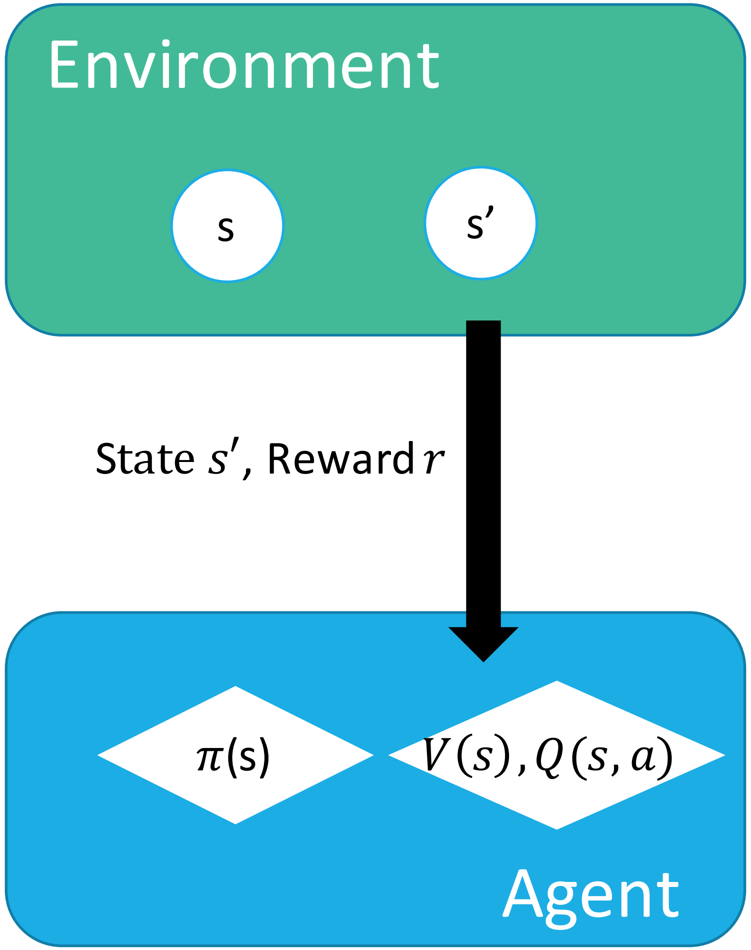
1. Agent observe $s_t \in S$
2. Choose action $a_t \in A$ conditioned on s_t using $\pi(s)$
3. Environment change its state $s_t \rightarrow s_{t+1}$



What is RL

- At each Time Step $t = 0, 1, \ldots$

1. Agent observe $s_t \in S$
2. Choose action $a_t \in A$ conditioned on s_t using $\pi(s)$
3. Environment change its state $s_t \rightarrow s_{t+1}$
4. Agent get reward $r_t \in R$ and observe s_{t+1}



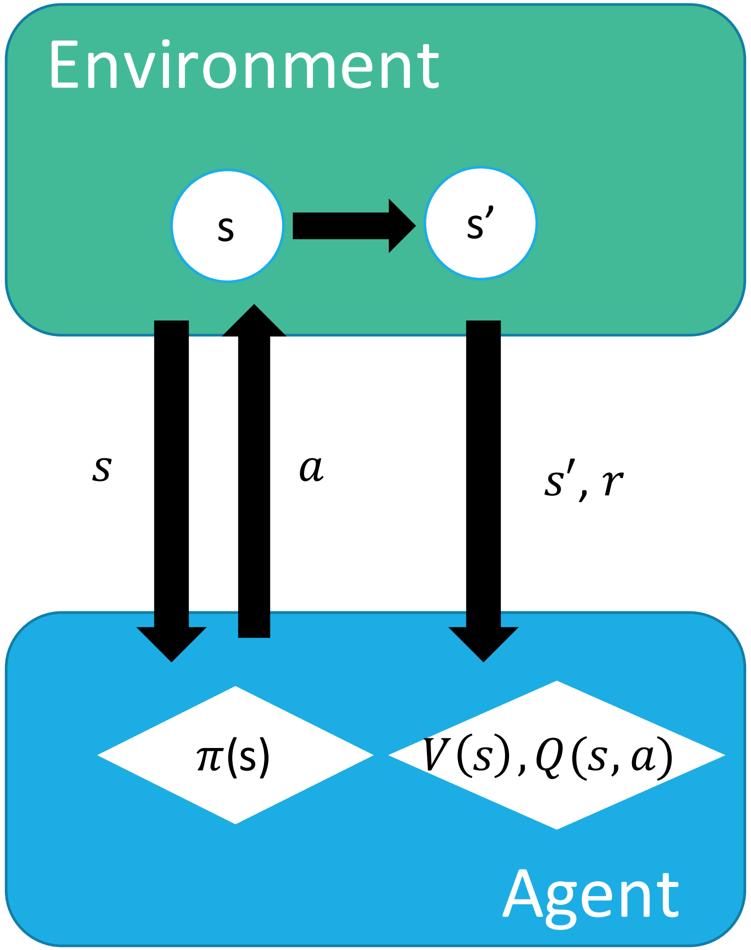
What is RL

- At each Time Step $t = 0, 1, \ldots$

1. Agent observe $s_t \in S$
2. Choose action $a_t \in A$ conditioned on s_t using $\pi(s)$
3. Environment change its state $s_t \rightarrow s_{t+1}$
4. Agent get reward $r_t \in R$ and observe s_{t+1}

Goal: To achieve policy that maximize cumulative rewards $\sum_t r_t$

(Usually introduce reward decay γ, then rewards become $\sum_t \gamma^{t-1} r_t$)



Applications of RL

- Play game (ex. Atari, Backgammon, Go)
  - reward for winning/losing a game or get point in game
- Fly stunt manoeuvers in a helicopter
  - positive reward for following desired trajectory, negative reward for crashing
- Manage an investment portfolio
  - positive reward for each $ in bank
- Control a power station
  - positive reward for producing power negative reward for exceeding safety thresholds
- Make a humanoid robot walk
  - positive reward for forward motion negative reward for falling over
- Dialogue system
  - rewards for good conversation

Characteristic of RL

- There is no supervisor, only a reward signal
- The environment is initially unknown
  - Different from Planning (often combine them)
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)
- Agent's actions affect the subsequent data it receives
- Markov decision Process
  - $P(s_{t+1}, r_t \mid s_t, a_t, s_{t-1}, a_{t-1} \ldots s_0, a_0) = P(s_{t+1}, r_t \mid s_t, a_t)$

Components of RL

An RL agent may include one or more of these components

- Policy:
  - agent’s behaviour function
- Value function:
  - how good is each state and/or action

Value Function

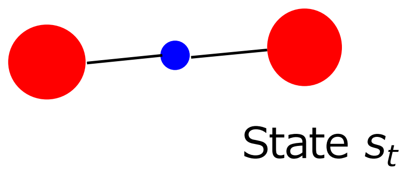
- Prediction of future reward
- Used to evaluate the goodness/badness of states

State-Value function$V^{\pi}(s)$

Function that predict expected future rewards after s

$$V^{\pi}(s) = \mathbb{E}_{s\sim P, r\sim R, a\sim \pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s\right]$$

Value of state s_t

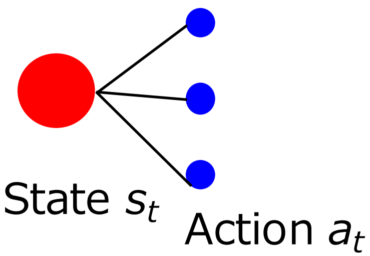


Action-Value function $Q^{\pi}(s,a)$

Function that predict expected future rewards after s if agent choose action a

$$Q^{\pi}(s,a) = \mathbb{E}_{s\sim P, r\sim R, a\sim \pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, a_t = a\right]$$

Value of action a_t at s_t



Classification of RL

- Model free or Model based
  - Whether agent use the model of environment
- On-policy or Off-policy
  - Whether directly improve policy used for decision making or not
- Value based or Policy based
  - Whether learn Value function or policy or both

Value based or Policy based

- Value-based
  - Learn Value-function, Implicit policy ex. TD-learning
- Policy-based
  - No Value-function, Learn policy directly
- Actor-Critic
  - Learn both Value-function and Policy

Value based or Policy based

- Value-based
  - Learn Value-function, Implicit policy ex. TD-learning
- Policy-based
  - No Value-function, Learn policy directly
- Actor-Critic
  - Learn both Value-function and Policy

TD Learning (Temporal-Difference Learning)

Bellman Equation

Cumulative Rewards $=$ Instance reward + expected future reward

$$\mathrm{V}^{\pi}(s) = \mathbb{E}_{\pi,s}[\, r_t + \gamma\, V^{\pi}(s')]$$

$$\mathrm{Q}^{\pi}(s,a) = \mathbb{E}_{\pi,s}[\, r_t + \gamma\, Q^{\pi}(s',\pi(s')]$$

- Update Value-function using Bellman Equation

$$\mathrm{V}^{\pi}(s) \leftarrow \alpha\, \mathrm{V}^{\pi}(s) + (1-\alpha)\,\{\, r_t + \gamma\, V^{\pi}(s')\}$$

$$\mathrm{Q}^{\pi}(s,a) \leftarrow \alpha \mathrm{Q}^{\pi}(s,a) + (1-\alpha)\{r_t + \gamma\, Q^{\pi}(s',\pi(s')\}$$

- Can update immediately without waiting episode end
- Some research said that Out brains are doing TD-learning

Q-learning

One of most classical algorithms

- Optimize Action-Value function
- Using Implicit Policy $\pi(s) = \arg\max_{a} Q(s, a)$
- Update

$$\mathrm{Q}^{\pi}(s, a) \leftarrow \alpha \mathrm{Q}^{\pi}(s, a) + (1 - \alpha)\{r_t + \gamma \max_{a'} Q(s', a')\}$$

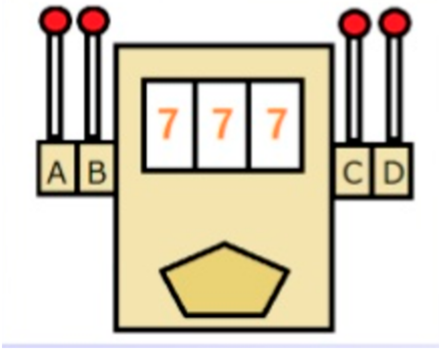
- Always choose action of highest Q-value → Update become efficient

→ Is it good to always choose best action for that time ?? It will cause local optima??

Exploitation-Exploration Trade-off

Multi-armed bandit

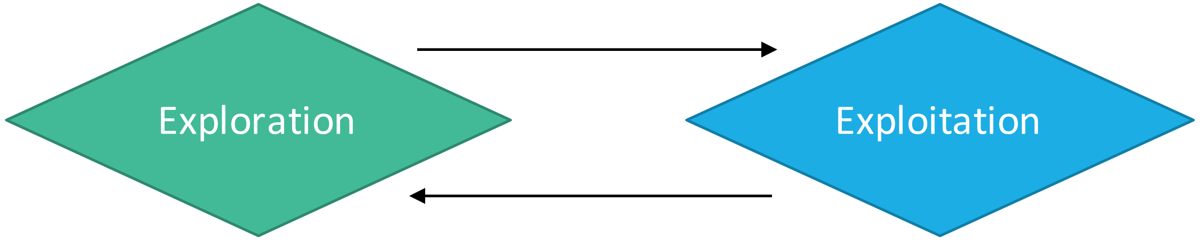
- Slot-machine that has multiple arms
- Each arm has different probability and prize
- Maximize the prize in limited trial



Idiot ： Arm B omit 10$ once → Play only Arm B

Smart ： Indeed Arm B is not bad, but.. The rest might be better arm

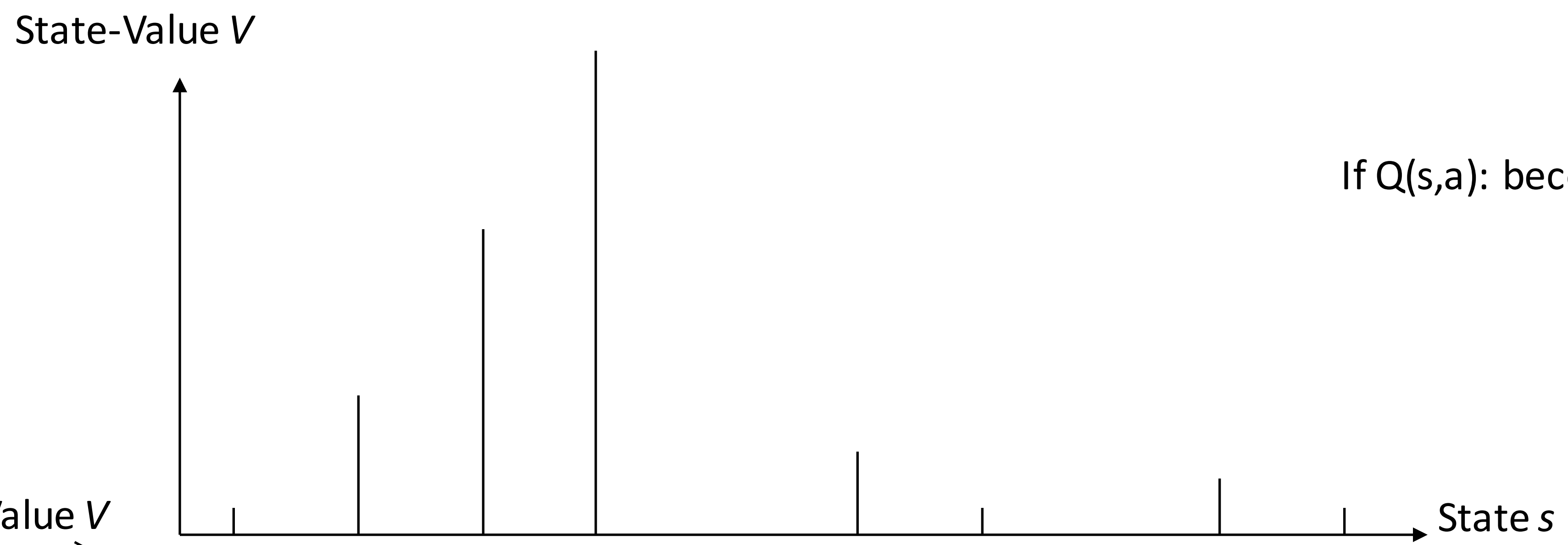
→ Exploitation-Exploration Trade off



Q-learning

- Greedy method
  - $\pi(s) = \arg\max_a Q(s,a)$
- ϵ - Greedy method
  - Probability ϵ : choose random action a → Exploration
  - Probability 1 - ϵ : $\arg\max_a Q(s,a)$ → Exploitation
  - In most case , ϵ aneal to small value with learning proceed (balance is important not to get local optima)
- Boltzman policy
  - $X_i = Q(s, a_i)$、$p(a_i) = \frac{e^{X_i/T}}{\sum_i e^{X_i/T}}$
  - Probability change depends on its Q-value

Representation of Value Function



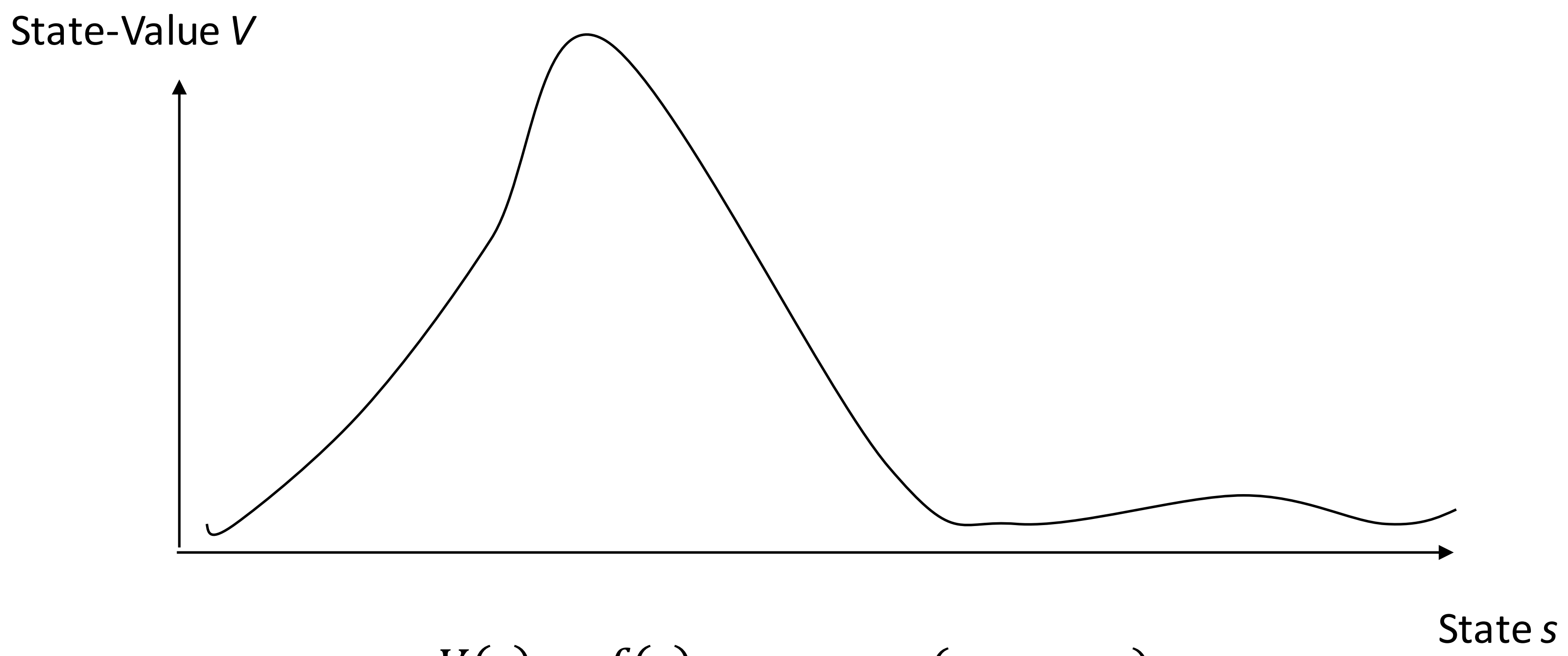
If Q(s,a): become table

State-Value V

0.1	0.5	1.0	1.5	0.0	0.3	0.1	…	0.2	0.1

If state has high dimension or continuous space, become intractable

Function Approximation of Value Function



$$V(s) = f(s)\ , \qquad s = (x_0, x_1, \dots)$$

Linear model or general linear model may be chosen for f

DQN

- Use CNN for function approximation of Action-Value function in Q-learning
- Using NN for function approximation of Value function is not new idea
  - TD – Gammon [Tesauro, 1994]

→ Then, what is NEW?

- Q-learning is theoritically proven that it converge with linear function approximation, proper learning rate and sufficient exploration

→ If using complicated non-linear model for function approximation, no guarantee

→ But… They did it !! (with several technique)

Experience Replay

Difficulty of DNN × RL

- learning NN requires data sample to be i.i.d.

→ But!! In RL…

  - Time-sequential input (correlation in samples)
  - Distribution of samples change with proceeding
    - It may forget previous states

→ Unstable

- It need enormous data samples
  - It need much interaction with environment

→ Too much time to get samples

→ Save experience to somewhere (Replay buffer)
And in update step, sample stochastically from buffer

Q-learning with Experience Replay

- Update formula in Q-learning （Again）

$$Q^{\pi}(s,a) \leftarrow \alpha Q^{\pi}(s,a) + (1-\alpha)\{r_t + \gamma \max_{a'} Q(s', a')\}$$

- Characterstics of Q-learning
  - Only using $(s, a, r. s')$ → not depend on old policy (off-policy sample)
  - (s, a, r, s’) can fed arbitrary order if sample can be given infinitely

→ Experience Replay can be applied!!

  - Save experience sample $(s, a, r. s')$ in Replay buffer
  - Update with mini-batch sampled stochastically from buffer

→ It reduce temporal correlation in mini-batch and sample-efficient

Classification of RL

- Value based or Policy based
  - Value-based
    - Learn Value-function, Implicit policy ex. TD-learning, etc
  - Policy-based
    - No Value-function, Learn policy directly ex. Policy gradient method, guided policy search,.. etc
  - Actor-Critic
    - Learn both Value-function and Policy

Policy based

- Value based
  - Policy is implicit ex. Q-learning: $\pi(s) = \arg\max_{a} Q(s, a)$

→ If action space A is too large, take too much to compute argmax

→ Some time stochastic policy is desired

→ In robot control, action space should be continuous

➡ Learn explicit policy $\pi(s)$

Policy gradient method
Guided policy search
.. Etc.

Tang-san will explain in next rinko!

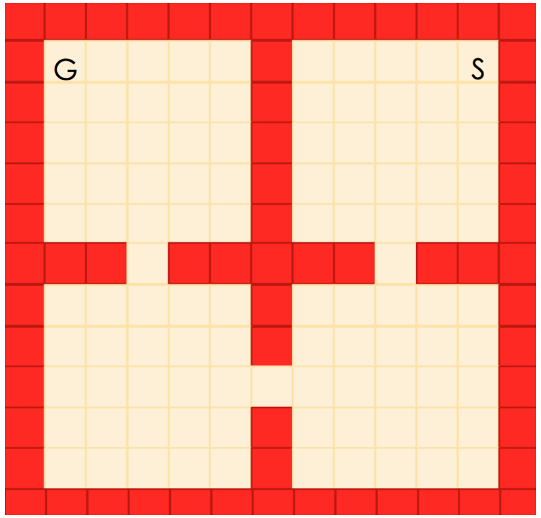
Actor-Critic

- Learn both policy and Value-function
- It means.. "Value function (Critic) criticize its policy (Actor) "
- Critic update
  - TD-learning, TD (λ)
- Actor update
  - Update using TD-error
  - Policy gradient method

Design of Reward

- Task that reach goal
  - Ex 1） get close to goal (+1), reach goal (+100)

    → Take so much time to reach goal
  - Ex 2) reward design depends on potential
  - → Learning proceed quickly, but it need experience and additional parameter

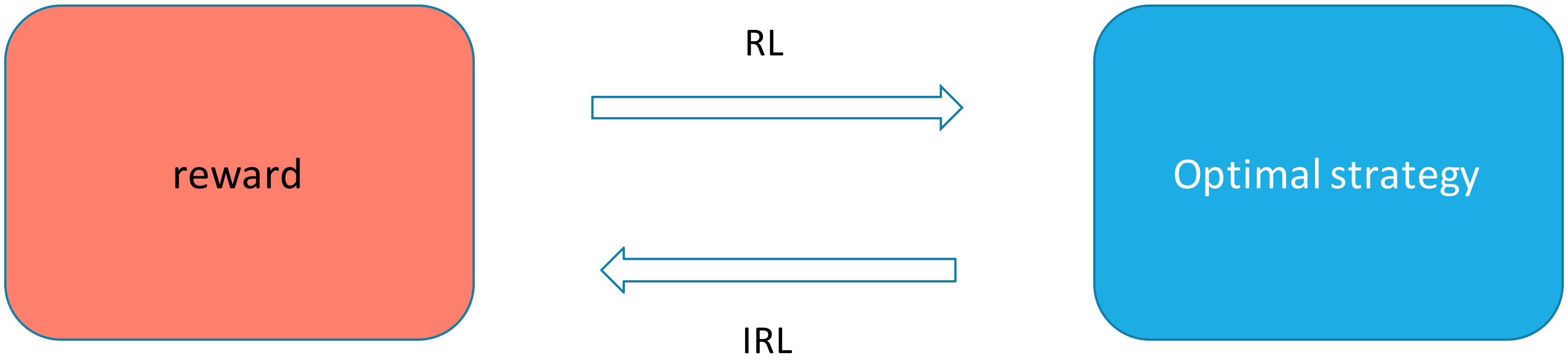


➡ Research about design of reward

- Reward shaping (how to design potential reward)
- Inverse reinforcement learning (apprentice learning)

Inverse Reinforcement Learning

- In RL, learn optimal policy or strategy from reward = what is good
- In IRL, learn reward function == what is good from optimal policy or strategy



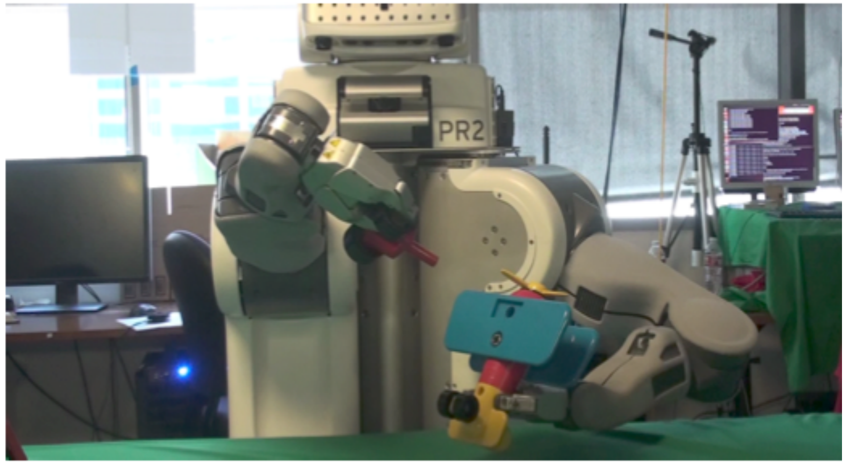
おまけ①

Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection
[Sergey Levine et al., 2016]

- Googleがロボットアーム14台並列に動かして物体のグラスピングを学習させてたやつ
- 概要
  1. 様々な状況でモーターを動かしてみて、成功したか失敗したかのサンプルを保存
  2. 保存したサンプルを利用して、Prediction Network $g(I_t, v_t)$ { I_t: 視覚情報, v_t: サーボへの命令 }をsupervised-learning
  3. サーボへの命令は、サンプリングベースで$g(I_t, v_t)$が高くなるのを選ぶ
- 強化学習とよく言われるが、 self-supervised learningと呼ばれる自分でデータサンプルを集めて学習する枠組み

おまけ②

End-to-End Training of Deep Visuomotor Policies
[Sergey Levine, 2015]



- UCバークレーのロボットが最初はランダムだが、だんだん鍵穴に鍵させるようになるやつ
- Guided Policy Searchと呼ばれる枠組み
  - Policy based
  - TD学習とかとは全く異なるアプローチ
  - ガチガチの制御理論（理解するの諦めたのでよくわかってないです）
  - 強化学習自体広い意味を持つのでこれも含まれるらしい

おまけ③

Asynchronous Methods for Deep Reinforcement Learning
[Mnih et al, 2016]

- 非同期に多数のエージェントを走らせてパラメータを同時に更新することでサンプル数を確保すると同時に入力の相関をなくすことができる

→ Experience Replayを使う必要がない

→ on-policyなRLアルゴリズムが使用可能!!

- Advantage functionを用いたActor-Criticを非同期で走らせた結果、CPUで１日たった時点で他手法を大きく上回る
(A3C : Asynchronous Advantage Actor Critic)