Reinforcement Learning
(RL)

一. What is RL ?

industrial revolution → repeated physical solutions —Digital revolution→ repeated mental solutions —AI revolution→ ?

RL: interacting the environment

※ Learning to make decisions from interactions

① active

② sequential interactions

Goal:

① find previously unknown solutions

② find solutions online
for unforeseen circumstances

RL vs ML

Supervised signal $\Longrightarrow$ reward

- label
- action

how good something is compared to something else

1. No supervision, only a reward signal
2. feedback can be delayed,
   not instantaneous
3. time matters
4. earlier decisions affect later interactions

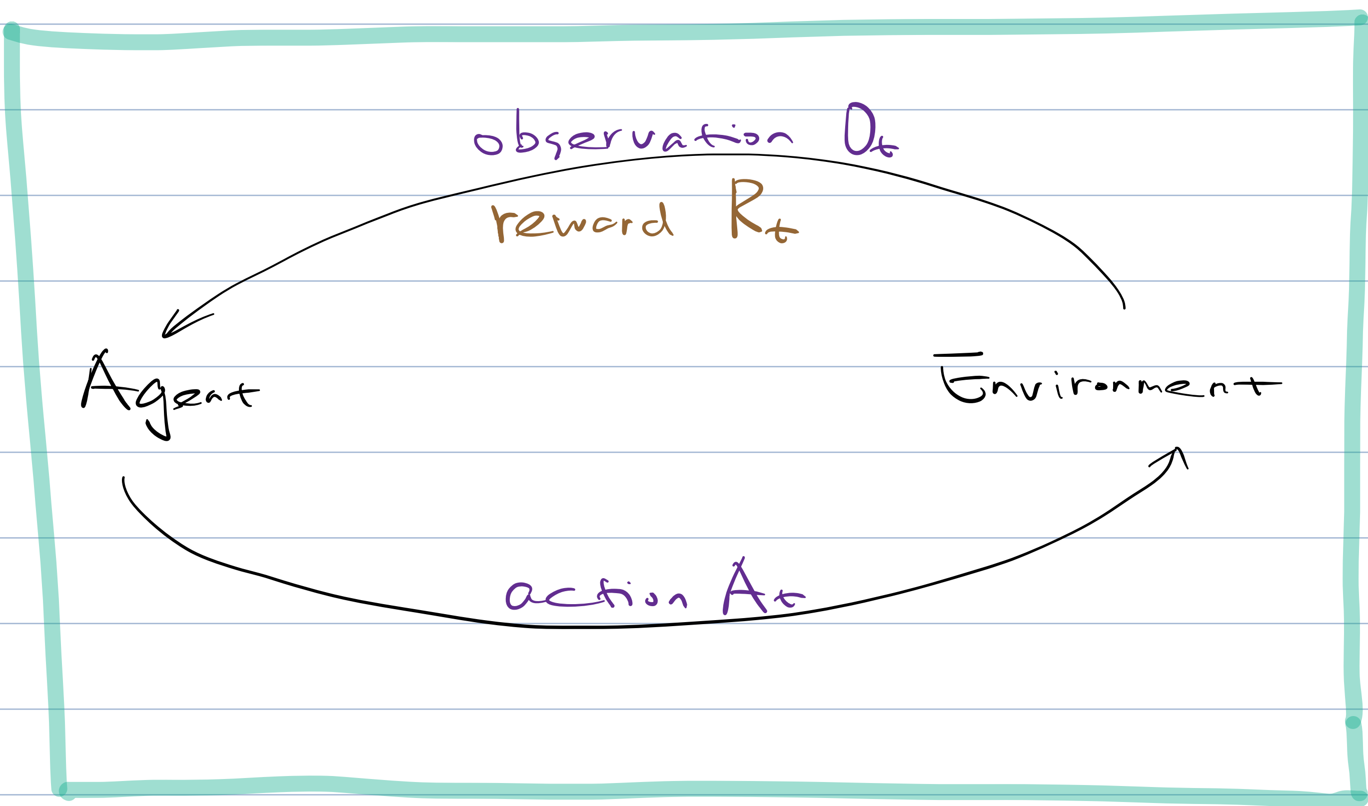
RL is learning what to do, how to map situations to actions, so as to maximize a numerical reward signals.

学习一个根据当前state选择一个能够最大化全局收益的action.

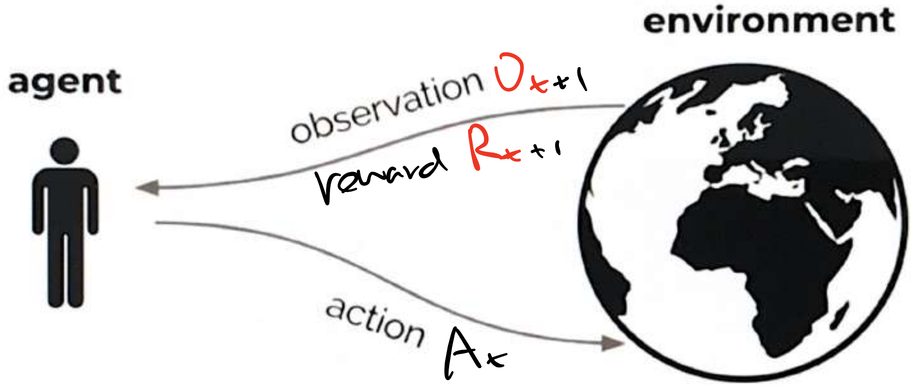
Core concepts of RL

- Environment
- Reward (signal)
- Agent
  - Agent state
  - policy
  - Value function (probably)
  - Model (optionality)

Agent 根据 state 选择 action 的方法叫做 policy



Agent and Environment



- At each step t the agent:
    - Receives observation O_t (and reward R_t)
    - Executes action A_t
- The environment:
    - Receives action A_t
    - Emits observation O_{t+1} (and reward R_{t+1})

Rewards R_t (reward hypothesis)

R_t : scalar feedback signal (reward / penalty)

indicates how well agent is doing at step t

(define its goal)

$$\boxed{G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots}$$ Maximize

Return (cumulative reward)

Reward Hypothesis:

Any goal can be formalized as the outcome of maximizing a cumulative reward.

Values : expected/mean cumulative reward from a state s

$$V(s) = \mathbb{E}[G_t \mid S_t = s]$$

$$= \mathbb{E}[R_{t+1} + R_{t+2} + R_{t+3} + \cdots \mid S_t = s]$$

$$G_t = R_{t+1} + G_{t+1}$$

Maximize value by picking suitable actions

Define Desirability of a state or action

(No supervised feedback)

Goal: Select actions to maximize value

Problem: actions $\rightarrow$ long-term consequences

reward $\rightarrow$ delayed

Solution: sacrifice immediate reward to gain long-term reward

policy

Mapping

states $\longrightarrow$ actions

Action values (condition the value on action):

$$q(s, a) = \mathbb{E}[G_t \mid S_t = s, A_t = a]$$

$$= \mathbb{E}[R_{t+1} + R_{t+2} + R_{t+3} + \cdots \mid S_t = s, A_t = a]$$

Agent components

Agent state : $S_{t+1} = f(S_t, A_t, R_{t+1}, O_{t+1})$

policy : $A = \pi(S)$; $\pi(A|S) = P(A|S)$

value function : $V(s) = \mathbb{E}[G_t \mid S_t = s]$

Model $= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s, \pi]$

P.S. Environment State : H_t

(Full observable / Partially observable)
Environment State

Markov decision process

(observation)
environment state : environment's internal state

history : a sequence of observations,
actions
rewards

$$H_t = O_0, A_0, R_0, O_1, \ldots, O_{t-1}, A_{t-1}, R_t, O_t$$

(used to construct an agent state S_t)

MDPs : Markov decision Processes

$$p(r, s \mid S_t, A_t) = p(r, s \mid H_t, A_t)$$

$$H_t \to S_t \to H_{t+1}$$

Markov

full observability

partial observability : agent gets partial information

(Partially observable MDPs)

Agent State

function of history

$$S_{t+1} = f(S_t, A_t, R_{t+1}, O_{t+1})$$

f: state update function

P.S. agent state $\ll$ environment state

Policy

define the agent's behaviour:

$$\text{agent state} \xrightarrow{\text{policy}} \text{action}$$

Deterministic policy: $A = \pi(s)$

Stochastic policy: $\pi(A|s) = p(A|s)$

Value Function

$$V_\pi(s) = \mathbb{E}[G(t) \mid S_t = s, \pi]$$

$$= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s, \pi]$$

discount factor $\gamma \in [0, 1]$ $\Big\{$ trades off importance of immediate vs long-term rewards

Bellman Equation (Bellman 1957)

recursive form:

$$G_t = R_{t+1} \gamma G_{t+1}$$

$$V_\pi(s) = \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t \sim \pi(s)]$$

$$= \mathbb{E}[R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s, A_t \sim \pi(s)]$$

$a \sim \pi(s)$: a is chosen by policy π in state s

(even if π is deterministic)

$$\boxed{V_*(s) = \max_a \mathbb{E}[R_{t+1} + \gamma V_*(S_{t+1}) \mid S_t = s, A_t = a]}$$

Hamilton-jacobi (bellman) equation

Value function approximations

① state space might be too big

② (sample) for expectations

Model

predict what the environment will do next.

$\mathcal{P}$ predicts the next state:

$$\mathcal{P}(s, a, s') \approx p(S_{t+1} = s' \mid S_t = s, A_t = a)$$

$\mathcal{R}$ predicts the next (immediate) reward

$$\mathcal{R}(s, a) \approx \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

$\boxed{\text{stochastic / generative models}}$

Agents → Value based (compare actions)

Agents → Policy based (compare policy)

Agents → Actor Critic (explicit action and policy)

Value-based 单步迭代，不是GD.（考虑拟合误差）

policy-based 是GD

需要大量样本去学习

学习成本大

不需对环境进行建模 → 最优策略

Environment $\xrightarrow{action}$ Model free → $[Q(s,a)]$ maximize Policy and/or value function

环境对输入的响应为一个映射

$s' = f(s,a)$

是否引入对环境的建模

Environment $\xrightarrow{action}$ Model based → Optionally policy and/or value function

环境对输入的响应为统计概率分布 $P(S_{t+1}|S_t, a_t)$

$P(s'|s,a)$

对环境进行建模 从而做出动作规划

$$\boxed{S_1 \xrightarrow{a_1} S_2, r}$$

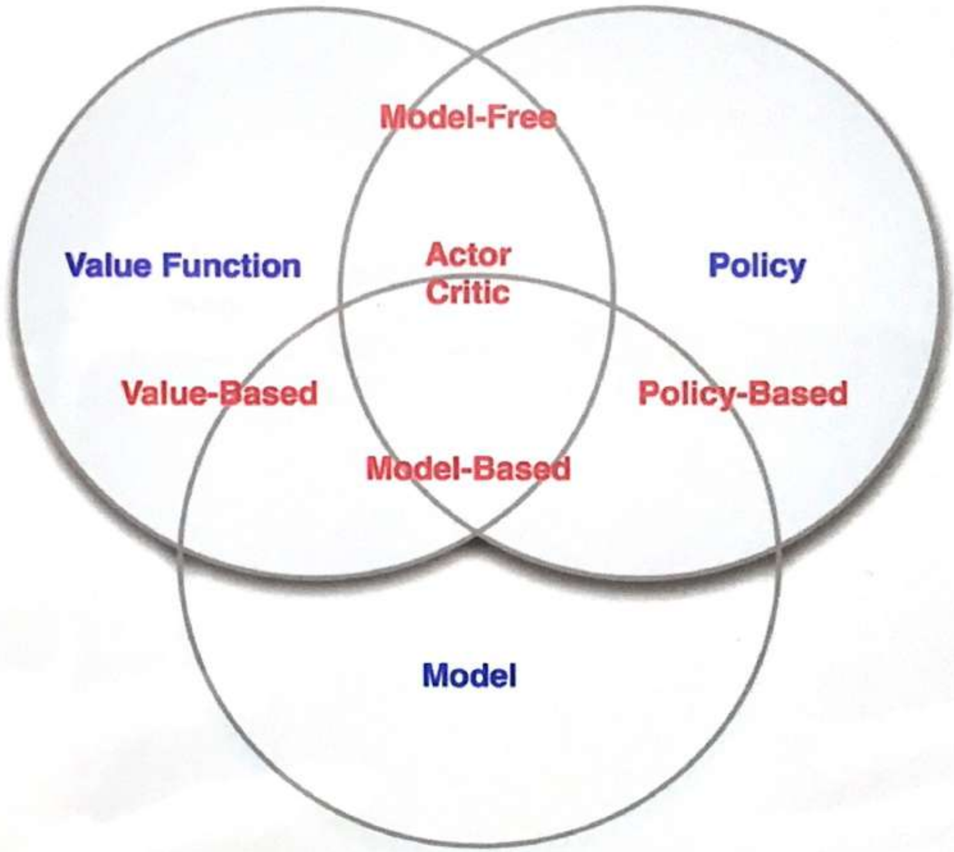
对下一步状态

和回报做出预测

状态转移概率: $T(S_2|S_1, a_1)$

奖励函数: $R(S_1, a_1)$

Agent Taxonomy



based extra critic you could have a
model based value based Asians and the

MacBook Pro

Challenges

1. Learning:

   ① environment initially unknown

   ② agent $\xleftrightarrow{\text{interact}}$ environment

2. planning:

   ① A model of the environment is given

   ② The agent plans in the model

   (without external interaction)

predicting

prediction: evaluate the future
(for a given policy)

control: optimize the future
(for a given action)

optimizing

$$\pi_*(s) = \underset{\pi}{\arg\max}\ V_\pi(s)$$

RL → trial-and-error search

RL → delayed reward

Learning the components of an agent

- All components are functions
  - Policies map states to actions
  - Value functions map states to values
  - Models map states to states and/or rewards
  - State updates map states and observations to new states
- We could represent these functions as neural networks, then use deep learning methods to optimize these
- Take care: we often violate assumptions from supervised learning (iid, stationarity)
- Deep reinforcement learning is a rich and active research field
- (Current) neural networks are not always the best tool (but they often work well)

any subset of those or superjet and
state updates which

MacBook Pro

IID : identically and independently distributed