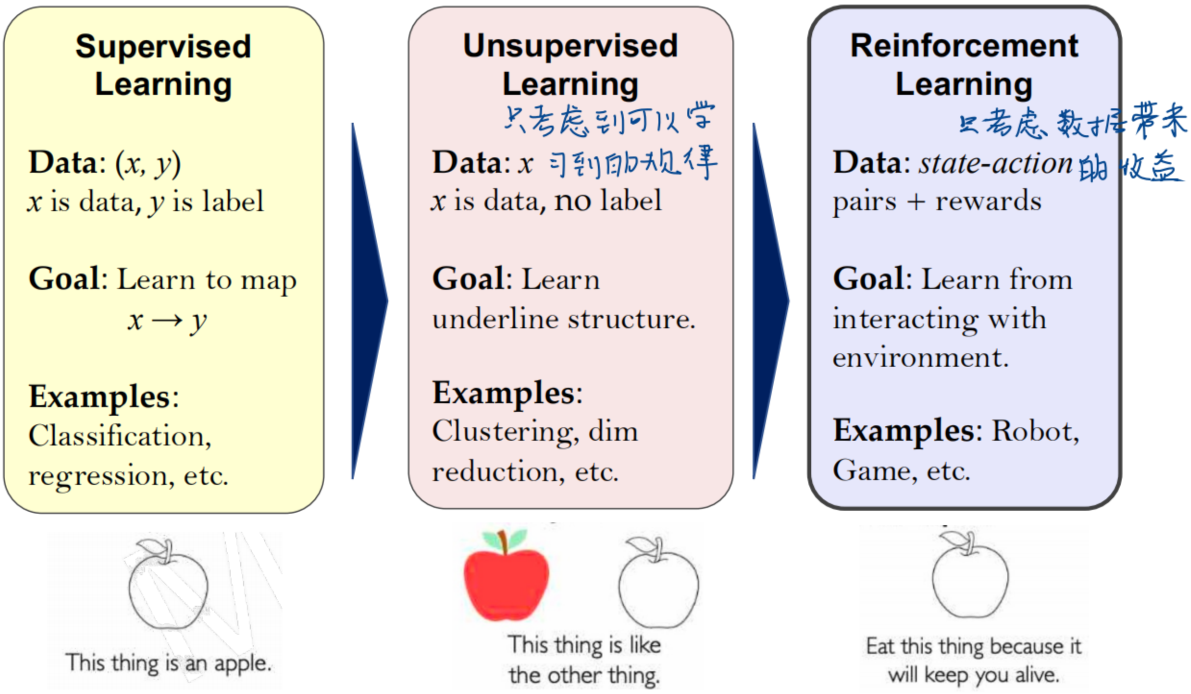
ch18: Reinforcement Learning

Introduce to RL

reinforcement learning 和其他的差异（P4）



Reinforcement Learning: Key Concepts（P9-12）

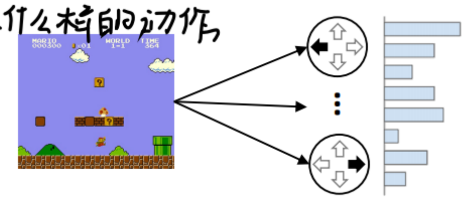
Agent: an actor which takes actions and learn knowledge
Environment: the world in which the agent exists and operates.
也会为agent提供feedback

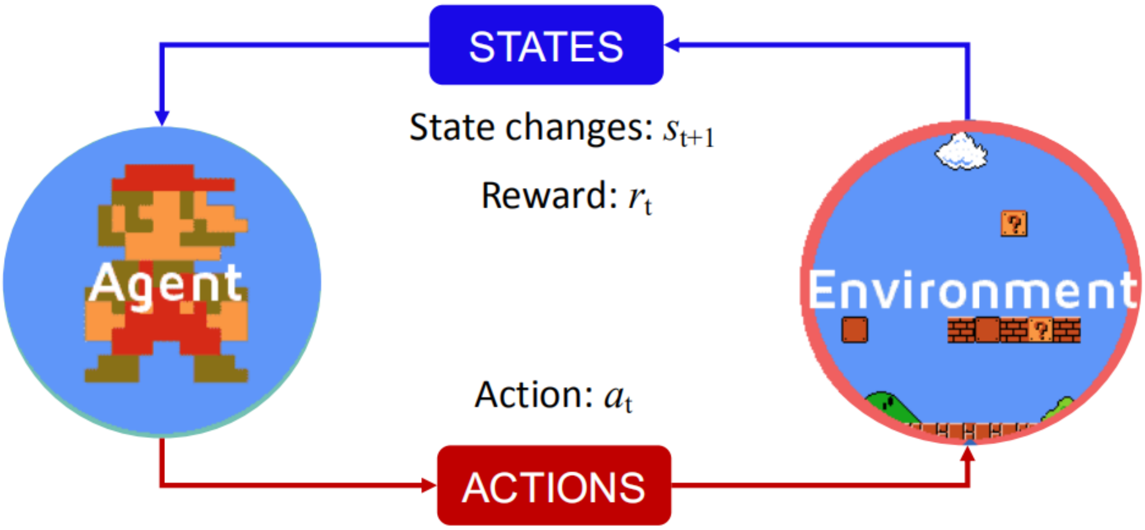
Action: a move the agent can make in the environment.
Action space A: the set of possible actions an agent can make in the environment.

State: observations agent观察到环境的状态
Reward: feedback that scores the agent's action.

Policy π: S $\rightarrow$ A: a function that maps from state to action

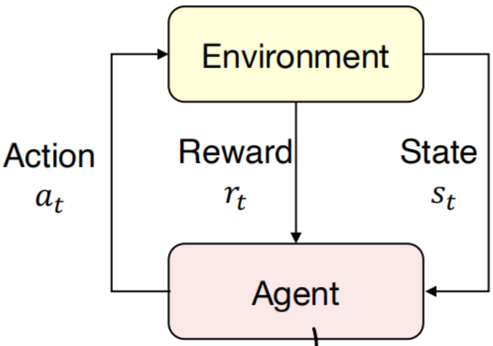
在一个state下面应该采取什么样的动作
是一个概率分布





Reinforcement Learning Problem Overview（P13）

- Learning from **interacting** with an **environment.**



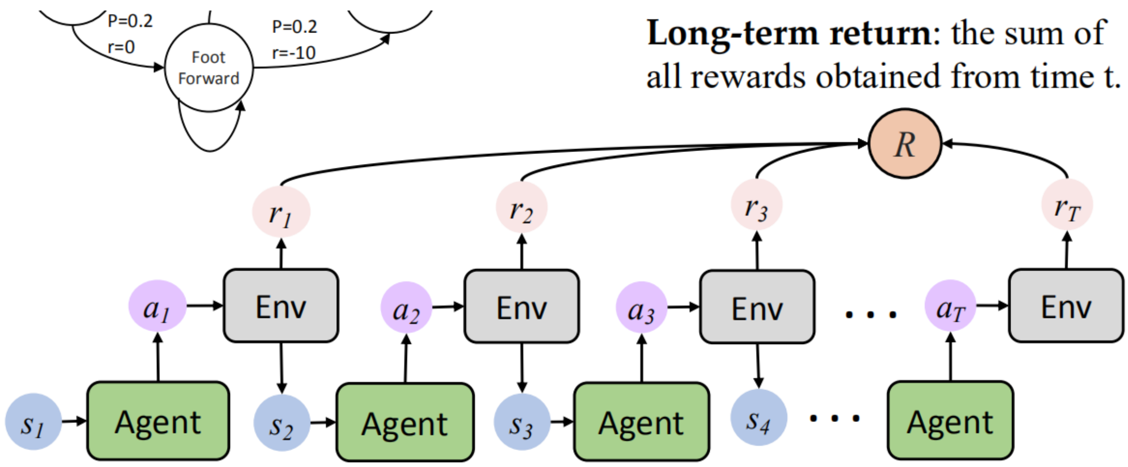
- a set of environment states S;
- a set of actions A;
- rules of transitioning between states; 获得reward的规则
- rules that determine the scalar immediate reward of a transition;
- rules that describe what the agent observes.

获得reward的时候还会发生状态的转移

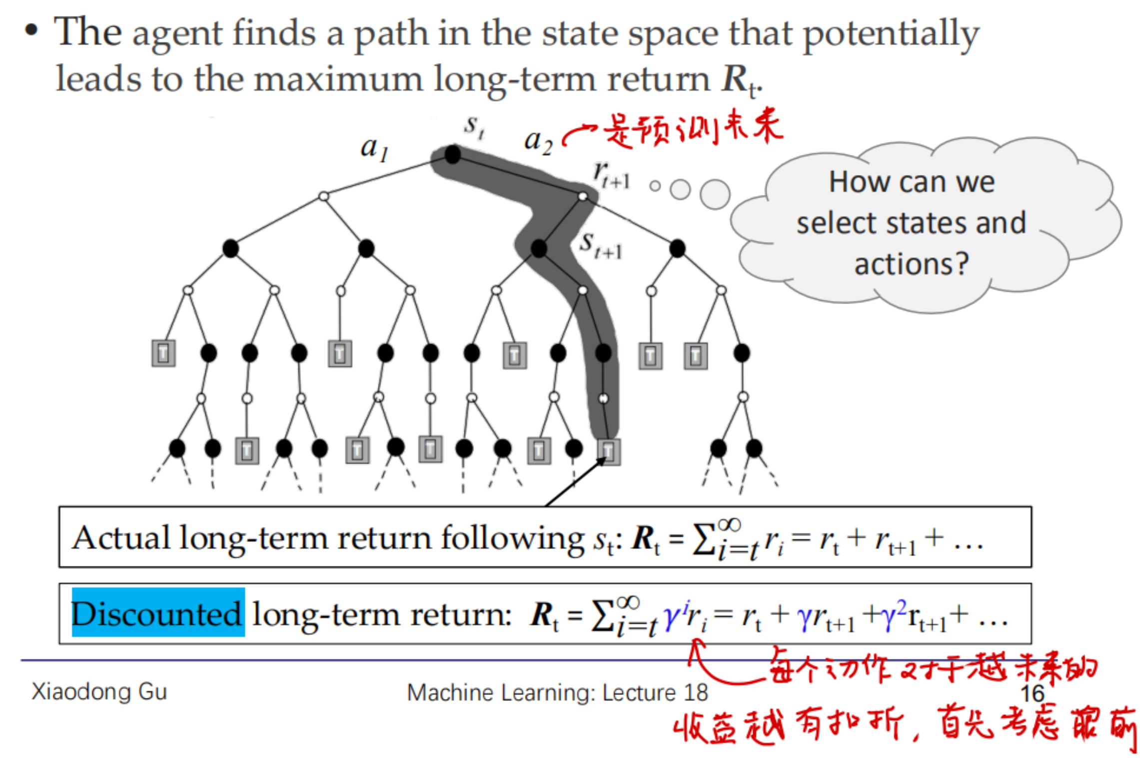
目标

Objective: learning a policy (i.e., state-action mapping) that maximizes the long-term payoff. Σr_t 最大

Reinforcement Learning Process（P15)



In the View of State Space（P16）



Scoring (s, a) pairs: Q-function（P17）

- The **Q-function** captures the **expected total future reward** an agent in state s can receive by executing a certain action a

$$Q(s_t, a_t) = \mathbb{E}[R_t \mid s_t, a_t]$$

score(state, action)

在 s_t 下采取 a_t 能够获得的收益 R_t 的平均值

$$\boldsymbol{R}_t = \sum_{i=t}^{\infty} \gamma^i r_i = r_t + \gamma r_{t+1} \dots + \gamma^n r_{t+n} + \dots$$

$$\begin{aligned} Q(s_t, a_t) &= \mathbb{E}[R_t \mid s_t, a_t] \\ &= r_t + \gamma r_{t+1} \dots + \gamma^n r_{t+n} + \dots \\ &= r_t + \gamma (r_{t+1} \dots + \gamma^{n-1} r_{t+n} + \dots) \\ &= r_t + \gamma\, Q(s_{t+1}, a_{t+1}) \end{aligned}$$

Recursion

根据Q-function确定policy(P18)

$$Q(s_t, a_t) = \mathbb{E}[R_t \mid s_t, a_t]$$

间接定义了策略

- Having obtained the Q-values, the agent derives the optimal **policy π(s)**, to infer the **best action to take** at state s

$$\pi^*(s) = \arg\max_a Q(s, a)$$

The policy should choose an action that maximizes future reward.

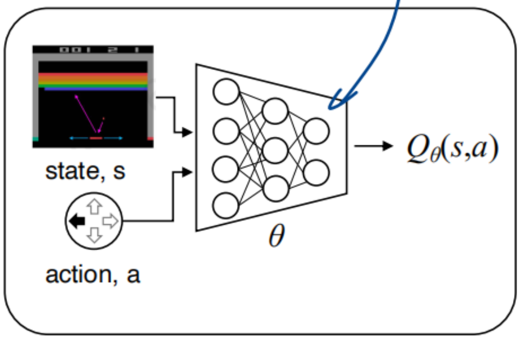
Reinforcement Learning Algorithms

Value Learning	Policy Learning
Find $Q(s, a)$ $a = \text{argmax}_a\, Q(s, a)$	Find $\pi(s)$ Sample $a \sim \pi(s)$

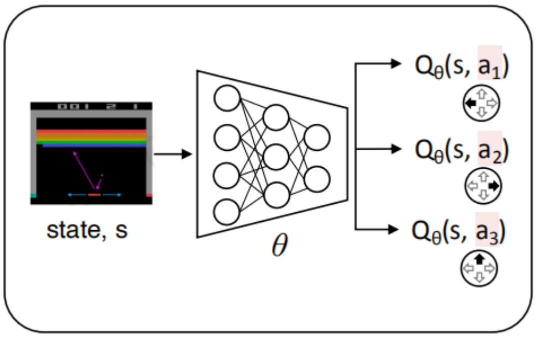
PART1

Deep-Q Networks (DQN)(P22-27)

- **Idea:** use deep neural networks to model the Q-function



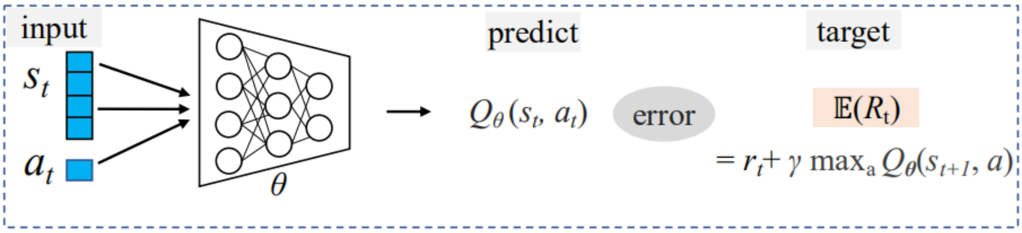
Action + State → Expected return



State → Expected return for each action

What happens if we take all the best actions?
Maximize the target return → Train the agent

Use deep neural networks (e.g., CNN) to model the Q-function.

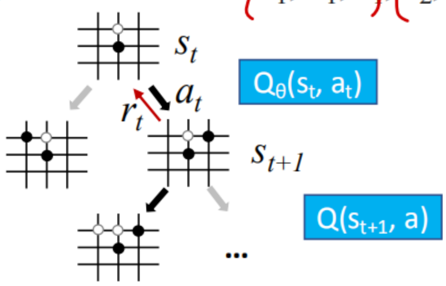


- Train:
  - estimate the parameters $\boldsymbol{\theta}$ in the $Q_\theta(s, a)$ network using agent history.
- Test:
  - calculate $Q_\theta(s, a)$ for all a's under s and choose $a = \operatorname{argmax}_a Q_\theta(s, a)$.

Training(P25-26)

- **Episode**: run agent and obtain $(s_1, a_1, r_1), (s_2, a_2, r_2), \ldots s_T, a_T, r_T$.

回合



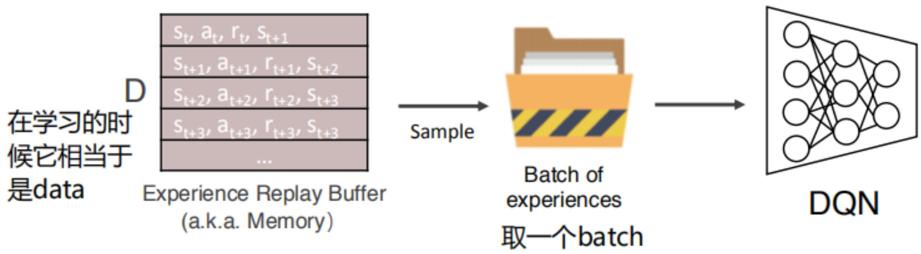
long term

对于未来reward的估计

- For any given input s_t and a_t, the model estimates a score $Q_\theta(s_t, a_t)$. 预测 Let r_t be the reward from the environment and s_{t+1} be the next state. The expected long-term reward is: $r_t + \gamma \max_{a \in A} Q(s_{t+1}, a)$.
- Our goal is to minimize their difference, i.e., the MSE loss:

$$\ell(\boldsymbol{\theta} \mid s_t, a_t, r_t) = \| \underbrace{(r_t + \gamma \max_a Q_\theta(s_{t+1}, a)}_{\text{target}} - \underbrace{Q_\theta(s_t, a_t)}_{\text{predicted}} \|^2$$

- **Experience Replay**: run agent multiple episodes and store the transitions in a **replay memory**: $D = \{(s^{(1)}, a^{(1)}, r^{(1)}, s'^{(1)}), \ldots, (s^{(N)}, a^{(N)}, r^{(N)}, s'^{(N)})\}$



- For multiple $(s, a, r, s') \in D$:

$$L(\boldsymbol{\theta} \mid D) = \underbrace{\mathbb{E}_{(s,a,r,s')\sim D}}_{\text{mini-batch experience replay}}[\| \, (\underbrace{r + \gamma \max_{a'} Q_\theta(s', a')}_{\text{Maximum possible Q-value for the next state (=Q\_target)}} - \underbrace{Q_\theta(s, a)}_{\text{Current predicted Q-value}} \|^2]$$

- Gradients:

两个Q都在learning

$$\boxed{\nabla_\theta L = \underbrace{E_{(s,a,r,s')\sim D}}_{\text{mini-batch experience replay}} \left(\underbrace{r + \gamma \max_{a'} Q(s', a'; \theta)}_{\text{Q target}} - Q(s, a; \theta) \right) \nabla_\theta Q(s, a; \theta)}$$

varying target?

- **Solution**: use some old, fixed parameters θ_{old} as a fixed Q-target (update every C steps)

$$\boxed{\nabla_{\theta t} \boldsymbol{L} = E_{(s,a,r,s')\sim D} (\underbrace{r + \gamma \max_{a'} \boldsymbol{Q}(s', a'; \theta_{old})}_{\text{fixed, old Q-target}} - \boldsymbol{Q}(s, a; \theta t)) \nabla_{\theta_t} \boldsymbol{Q}(s, a; \theta_t)}$$

不更新, C步一替换

The Deep Q-Learning Algorithm(P27)

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N 记忆体
Initialize action-value function Q with random weights θ
for episode $= 1, M$ **do**
    Initialise state s_t
    for $t = 1, T$ **do**
        加一定的bias来探索，避免局部最优解
        With probability ϵ select a random action a_t — Sampling
        otherwise select $a_t = \max_a Q^*(s_t, a; \theta)$
        Execute action a_t and observe reward r_t and state s_{t+1}
        Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
        Set $s_{t+1} = s_t$
        Sample random minibatch of transitions (s_t, a_t, r_t, s_{t+1}) from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } s_{t+1} \\ r_j + \gamma \max_{a'} Q(s_{t+1}, a'; \theta) & \text{for non-terminal } s_{t+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(s_t, a_j; \theta))^2$ — Training
    end for
end for

PART2

Policy Gradient: Key Idea(P31)

Policy Gradient: directly optimize the policy $\pi(s)$