# Artificial Intelligence
# 人工智能实验

## 强化学习基础

中山大学计算机学院
2024年春季

# 目录

# 1.1 强化学习介绍

## 基础知识

- 强化学习（Reinforcement Learning，RL），是指一类从（与环境）交互中不断学习的问题以及解决这类问题的方法。强化学习问题可以描述为一个智能体从与环境的交互中不断学习以完成特定目标（比如取得最大奖励值）。
  - Reinforcement learning is learning what to do—how to map situations to actions——so as to maximize a numerical reward signal. ----- Richard S. Sutton and Andrew G. Barto 《Reinforcement Learning: An Introduction II》
- 形式化定义：由六元组构成的马尔可夫决策过程定义

Markov Decision Process(MDP)

- $S$ denotes the state space
- $A$ is the action space
- $R = R(s, a)$ is the reward function
- $T: S \times A \times S \to [0,1]$ is the state transition function
- $P_0$ is the distribution of the initial state
- $\gamma$ is a discount factor
- Goal: find the optimal policy that maximizes expected reward

# 1.1 强化学习介绍

## 基础知识

- 状态价值函数(State-value Function)：用来度量给定策略的情况下，当前状态的好坏程度

$$v_\pi(s) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} | S_t = s\right]$$

$$G_t = r_{t+1} + \gamma r_{t+2} + \cdots = \sum_{k=0}^{\infty}\gamma^k r_{t+k+1}$$

$$\begin{aligned} v_\pi(s) &= \mathbb{E}[G_t | S_t = s] \\ &= \mathbb{E}_\pi[r_{t+1} + \gamma r_{t+2} + \cdots | S_t = s] \\ &= \mathbb{E}_\pi[r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \cdots) | S_t = s] \\ &= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \mathbb{E}_\pi[r_{t+1} + \gamma v(S_{t+1}) | S_t = s] \end{aligned}$$

- 动作价值函数(Action-value Function)：用来度量给定状态和策略的情况下，采用动作的好坏程度

$$q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} | S_t = s, A_t = a\right]$$

$$q_\pi(s,a) = \mathbb{E}_\pi[r_{t+1} + \gamma q(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

# 1.1 强化学习介绍

## 基础知识

- 最优动作价值函数：

$$q^*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} q^*(s',a')$$

  - 其中$P_{ss'}^a$是状态转移函数，即在状态$s$选择动作$a$转移到状态$s'$的概率

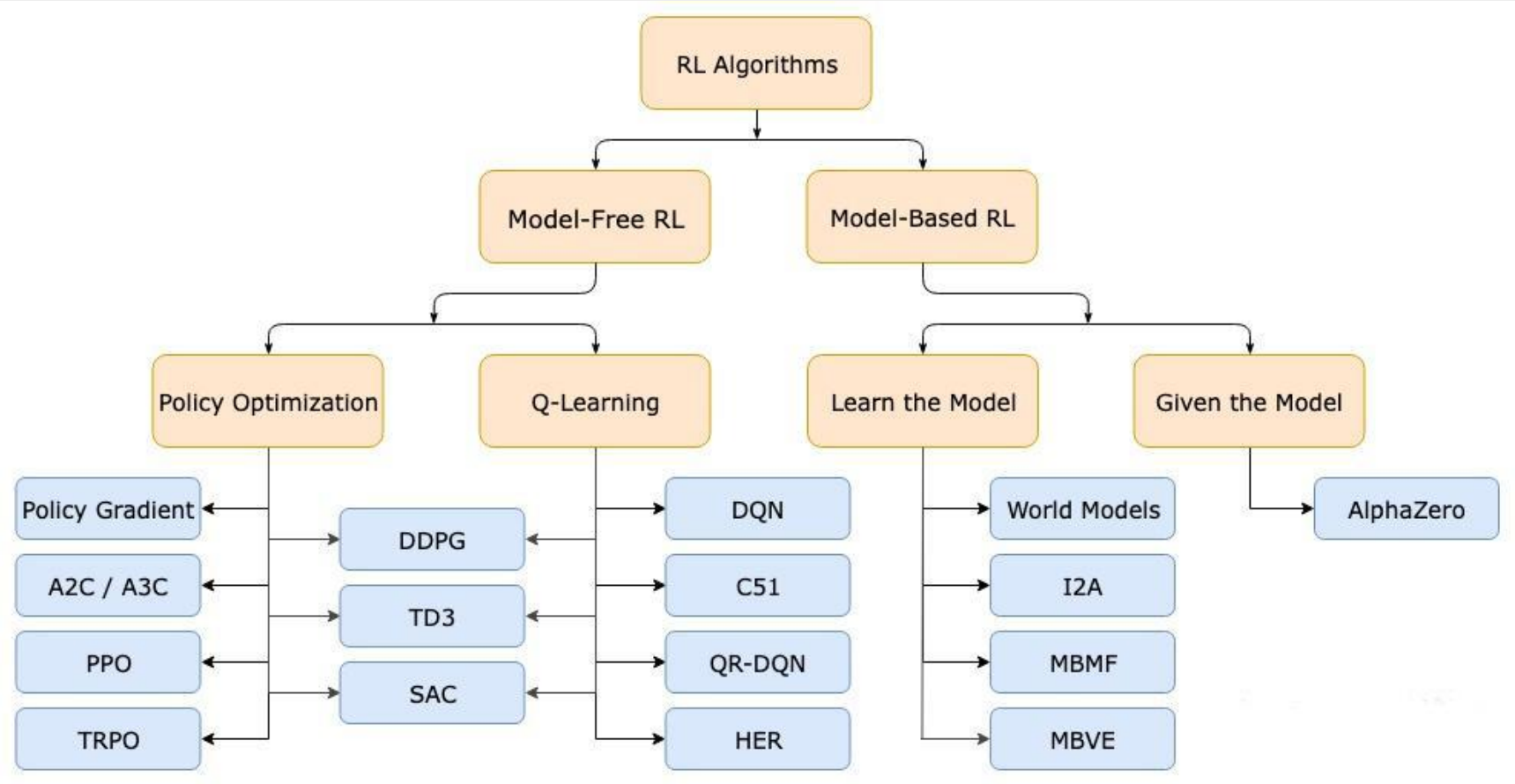
- 最优策略

$$\pi^*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_{a \in A} q^*(s,a) \\ 0, & \text{otherwise} \end{cases}$$

# 1.1 强化学习介绍

## 基础知识

- 强化学习算法分类



# 1.2 Q-learning

## Q-learning算法介绍

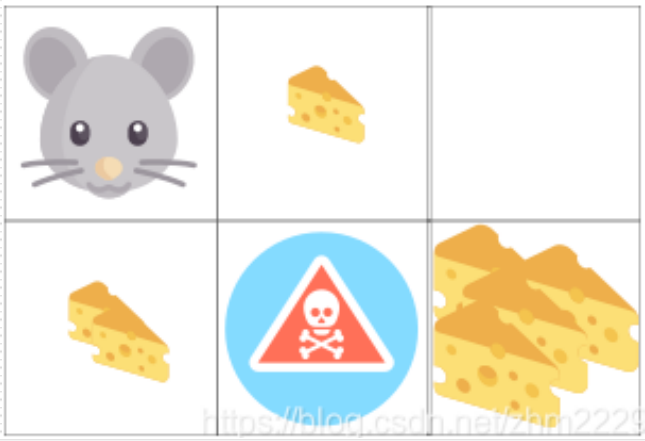
- 是一种 value-based 的强化学习算法，Q即为Q(s,a)，在某一时刻的state状态下，采取动作action能够获得收益的期望。
- 主要思想是将state和action构建一张Q-table表存储Q值，然后根据Q值选取能够获得最大收益的动作。
- 基于off-policy时序差分法，且使用贝尔曼方程可以对马尔科夫过程求解最优策略。
- 伪码：

1. Initialize Q-values ($Q(s, a)$) arbitrarily for all state-action pairs.
2. For life or until learning is stopped...
3. Choose an action ($a$) in the current world state ($s$) based on current Q-value estimates ($Q(s, \cdot)$).
4. Take the action ($a$) and observe the the outcome state ($s'$) and reward ($r$).
5. Update $Q(s, a) := Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right]$

# 1.2 Q-learning

## Q-learning算法实例

- 如左图是一个迷宫游戏，小老鼠（Agent）最开始在(0,0)位置，分别在(0,1),(0,2),(1,0),(1,1),(1,2)处可获得+1,0,+2,-10,+10的奖励值。当agent位于(1,1),(1,2)时，游戏结束。Agent的动作有四个，分别是上下左右。
- 右图是初始化的Q-table，其中行表示状态，列表示动作。



| | ← | → | ↑ | ↓ |
|---|---|---|---|---|
| Start | 0 | 0 | 0 | 0 |
| Small cheese | 0 | 0 | 0 | 0 |
| Nothing | 0 | 0 | 0 | 0 |
| 2 small cheese | 0 | 0 | 0 | 0 |
| Death | 0 | 0 | 0 | 0 |
| Big cheese | 0 | 0 | 0 | 0 |

# 1.2 Q-learning

## Q-learning算法实例

- Q-learning根据$\epsilon-greedy$选择动作：以$\epsilon$的概率随机选择动作，以$1-\epsilon$的概率贪心的（greedy）选择动作。
- 如下图所示，从start状态随机选择向右的动作。

| | ← | → | ↑ | ↓ |
|---|---|---|---|---|
| Start | 0 | 0 | 0 | 0 |
| Small cheese | 0 | 0 | 0 | 0 |
| Nothing | 0 | 0 | 0 | 0 |
| 2 small cheese | 0 | 0 | 0 | 0 |
| Death | 0 | 0 | 0 | 0 |
| Big cheese | 0 | 0 | 0 | 0 |

We move at random (for instance, right)

# 1.2 Q-learning

## Q-learning算法实例

- Q-learning使用贝尔曼方程更新：

$$NewQ(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma max Q'(s',a') - Q(s,a)]$$

New Q value for that state and that action — 0.1

Current Q value — 0

Learning Rate — 0.1

Reward for taking that action at that state — 1.0

Discount rate — 0.9

Maximum expected future reward **given the new s' and a' possible actions at that new state** — 0

0

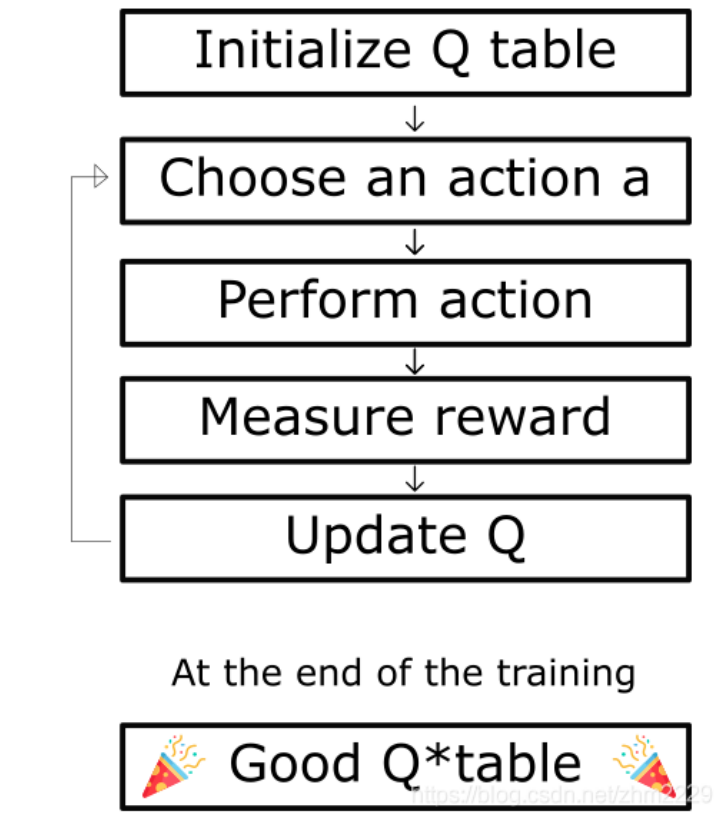
$\gamma = 0.9, \alpha = 0.1$

| | ← | → | ↑ | ↓ |
|---|---|---|---|---|
| Start | 0 | 0.1 | 0 | 0 |
| Small cheese | 0 | 0 | 0 | 0 |
| Nothing | 0 | 0 | 0 | 0 |
| 2 small cheese | 0 | 0 | 0 | 0 |
| Death | 0 | 0 | 0 | 0 |
| Big cheese | 0 | 0 | 0 | 0 |

# 1.2 Q-learning

## Q-learning算法实例

- Agent在每个step的时候都会用上述方法迭代更新一次Q-table，直到Q-table不再更新，或者到达游戏设置的结束局数。



# 1.3 SARSA

## SARSA算法介绍

- 和Q-learning类似，两者的区别在于：更新Q表的时候，选择的策略不同。Sarsa更新Q表的策略与选择动作策略一致，均采用$\epsilon-greedy$。而Q-learning更新Q表采用$greedy$ 策略，选择动作采用$\epsilon-greedy$ 。

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize $S$
  Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
  Repeat (for each step of episode):
    Take action $A$, observe $R$, $S'$
    Choose $A'$ from $S'$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until $S$ is terminal

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize $S$
  Repeat (for each step of episode):
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    Take action $A$, observe $R$, $S'$
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
    $S \leftarrow S'$;
  until $S$ is terminal

# 2. 实验任务

- 迷宫任务（无需提交）
  - 在给定迷宫环境中实现Q-learning和Sarsa算法。
  - 要求：
    - 在给定的代码框架下补充代码。
    - 最终智能体学习的策略能完成迷宫任务。