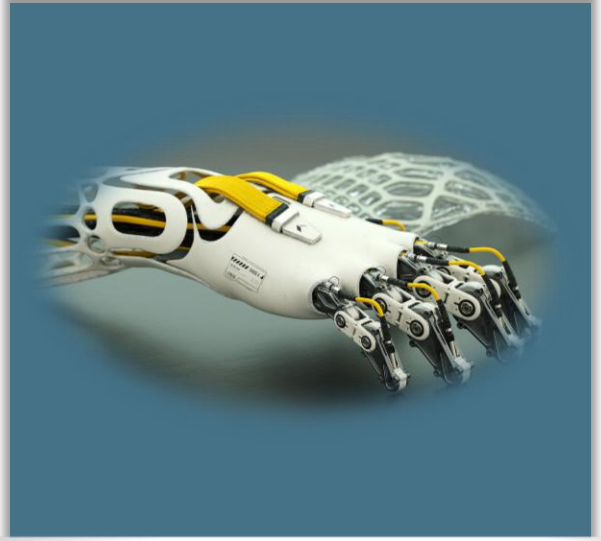
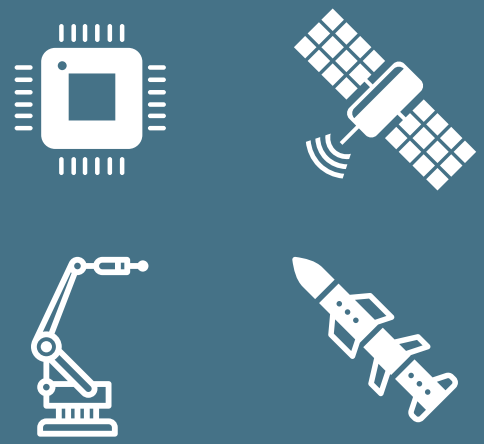
机器学习 第六讲

授课人：王闻博
Email: wenbo_wang@kust.edu.cn
昆明理工大学 机电工程学院

2026年04月17日



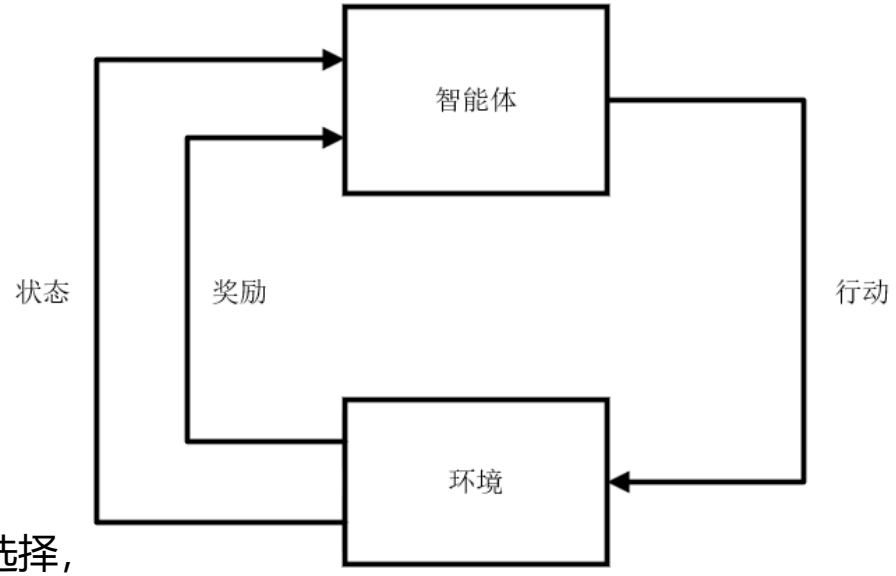
强化学习



1. 强化学习：概述
2. 强化学习的数学基础：马尔可夫决策过程（Markov Decision Process）
3. 无模型强化学习：时间差分算法和Q-Learning
4. 人工神经网络和强化学习：值函数近似
5. 函数近似下的策略梯度法

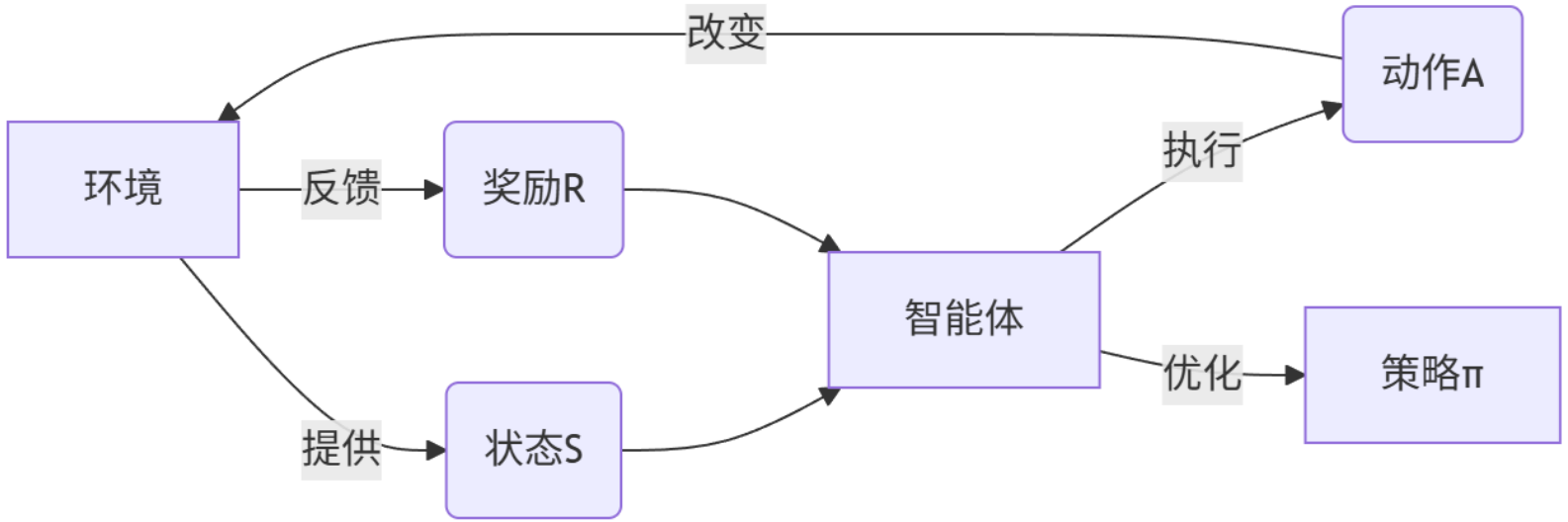
强化学习的一般表述

- **基本术语**
  - **智能体（Agent）**：负责决策的实体，也是算法的承载主体，智能体通过持续优化的策略与环境交互，目的是获得更多的“奖励”，而智能体基于从环境中观察到的状态和获取的奖励（反馈）不断改进策略。
  - **环境（Environment）和状态（State）**：环境是智能体执行任务时的交互对象，向智能体提供状态信息，以及奖励/惩罚的反馈。状态则是对环境的瞬时描述，表示给定时刻的环境特征。



  - **动作（Action）和策略（Policy）**：动作是智能体做出的与环境交互的动作选择，智能体的动作引发环境状态的转移。策略是智能体的“状态↦动作”的映射函数，决定在特定状态下智能体的动作选择一般用$\pi(s) \to a$表示。
  - **奖励（Reward）**：是环境到智能体的即时反馈信号，是评估智能体动作效果的核心指标。
  - **状态转移（State Transition）**：状态转移模型描述了环境动态变化，一般由历史环境状态和瞬时动作决定，一般用$\mathrm{P}(s'|s,a)$描述特定动作导致的状态迁移概率。
  - **回合（Episode）**：描述一个智能体执行任务的完整交互周期，包含一个智能体从初始状态到终止状态的“状态-动作-奖励-状态…”（即状态转移）的完整**轨迹（Trajectory）**。

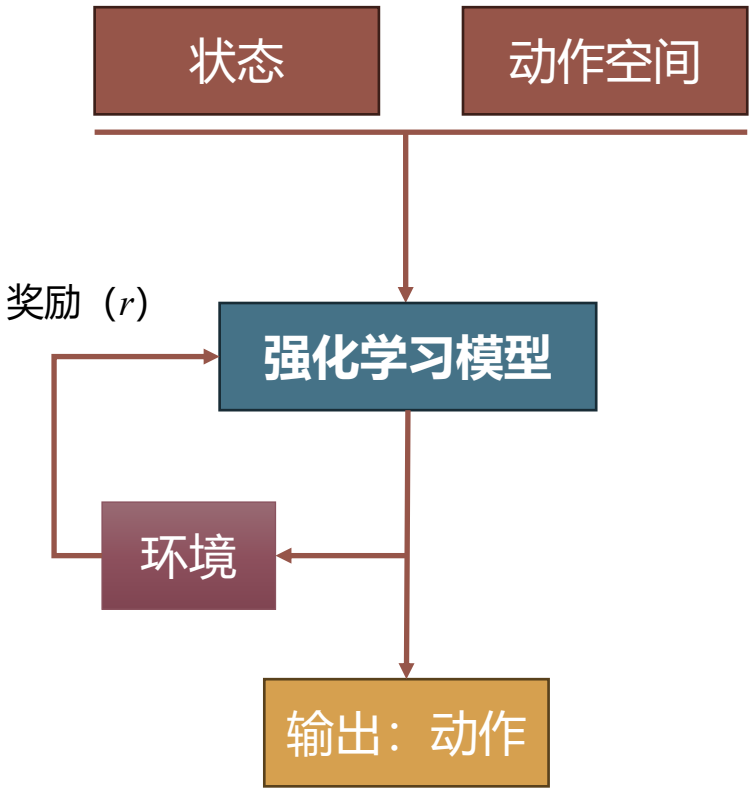
强化学习中的实体交互和术语关系

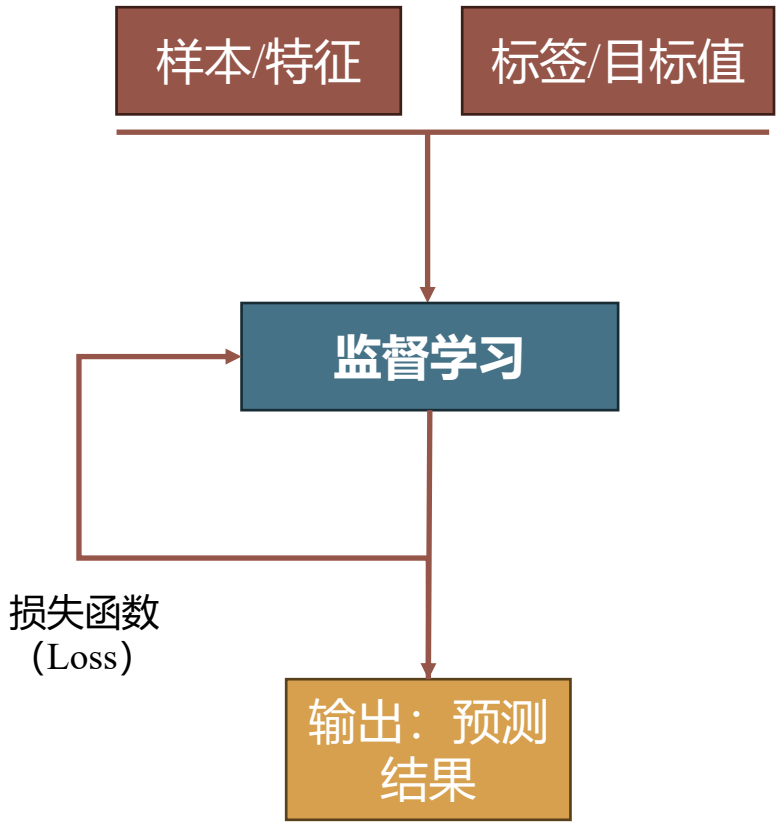


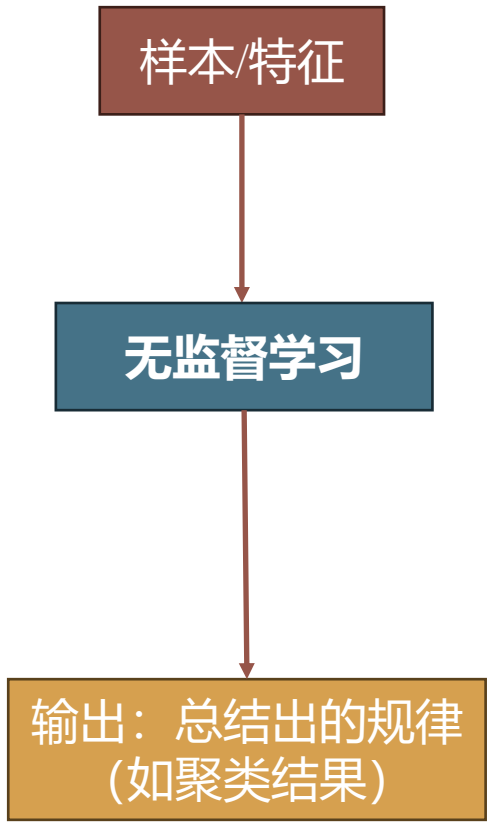
- **强化学习核心目标**
  - 目标本质：使智能体习得最优策略（Best Possible Policy）；
  - 在任意时刻，智能体能够通过观察知晓（或部分知晓）环境的实时状态。智能体同时知晓其自身采取的动作信息。智能体并不知晓如何针对不同状态选择动作，进而使其获得的回合内的奖励收益最大化。“如何选择动作以最大化回合奖励”，即最优策略映射关系，需通过自主学习获得。

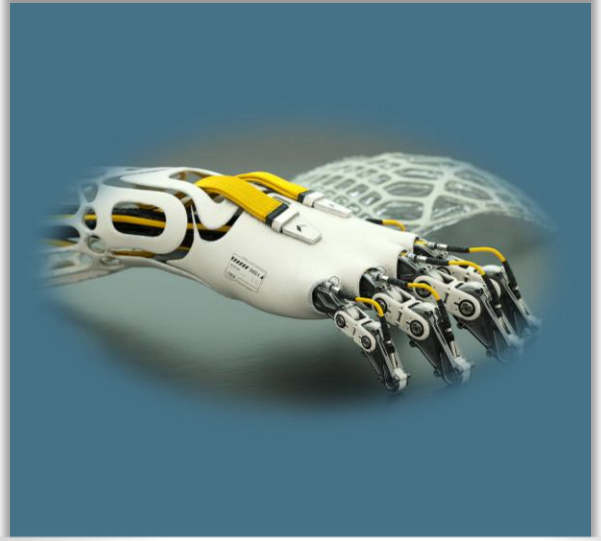
应用简例：无地图模式下基于深度强化学习的UAV自主避障与导航

强化学习范式和监督学习、无监督学习的对比

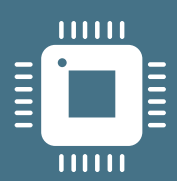








强化学习




1. 强化学习：概述
2. 强化学习的数学基础：马尔可夫决策过程（Markov Decision Process）
3. 无模型强化学习：时间差分算法和Q-Learning
4. 人工神经网络和强化学习：值函数近似
5. 函数近似下的策略梯度法

前置知识：马尔可夫过程（Markov Process）

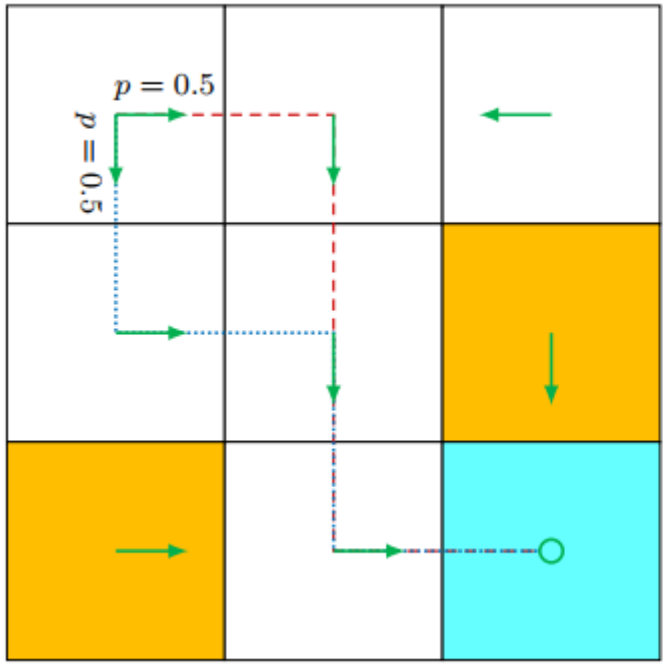
- **马尔可夫性**
  - 定义给定t时刻的状态为s_t，则马尔可夫性特指当前状态包含了对未来预测所需要的所有有用信息，只要当前状态可知，所有的历史信息都不再需要，当前状态就可以决定未来。通俗地讲：“未来状态只依赖现在，与过去无关”。
  - **使用状态转移概率描述：**

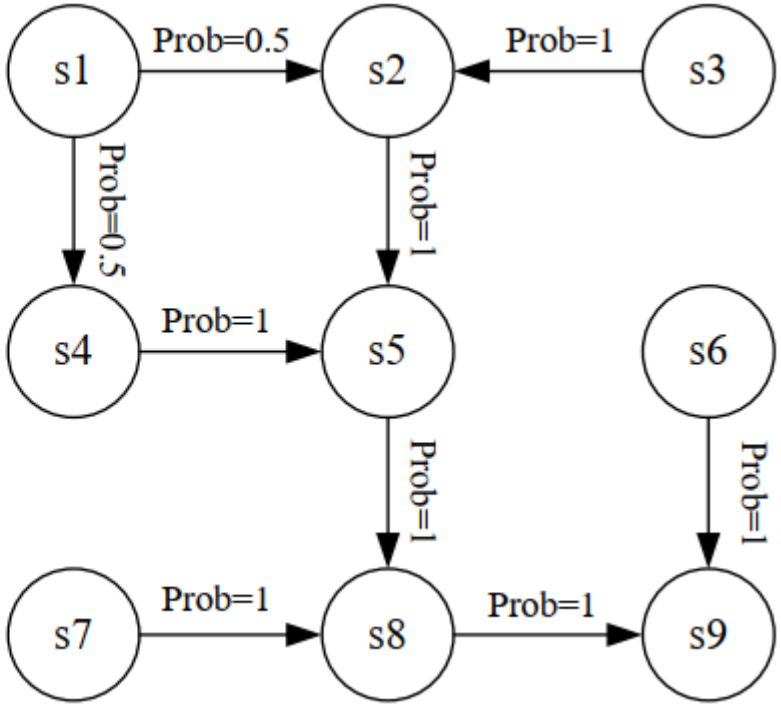
$$p(s_{t+1}|s_1, s_2, \cdots, s_t) = p(s_{t+1}|s_t)$$

- **马尔可夫过程（或称马尔可夫链，Markov chain）**
  - 是一个无记忆的随机过程$s_1, s_2, \cdots, s_t$；
  - 可以用一个二元组$\langle \mathcal{S}, \mathrm{Pr} \rangle$表示，其中
    - $\mathcal{S}$是有限数量的状态集，如$\mathcal{S} = \{s_1, s_2, s_3\}$，表示状态集中包含三个可能得状态；
    - Pr代表状态转移概率矩阵，其元素位置（$s \to s'$）对应状态转移概率$p(s_{t+1} = s'|s_t = s)$，其中s'表示下一时刻的状态，s表示当前状态。

马尔可夫过程（Markov Process）

- **示例**
  - 迷宫中（左图），每个格子代表一个状态，共9个状态；
  - 状态之间（格子到邻接格子）的转移概率由人为定义，如右图所示。



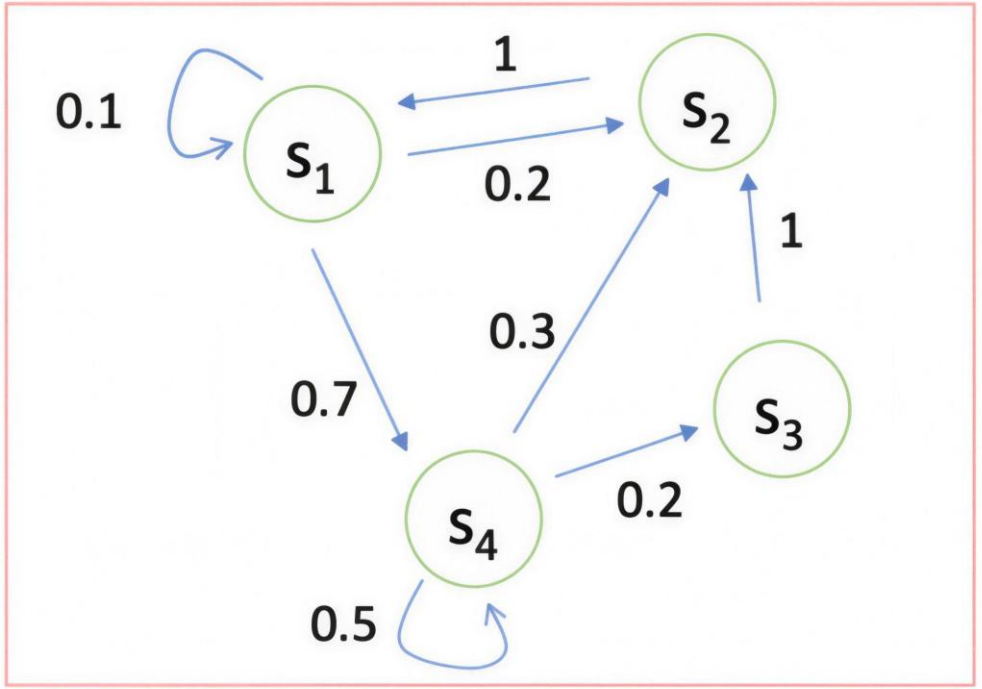


图来源：赵世钰《RL的数学原理》

马尔可夫过程（Markov Process，续）

- **另一个示例**
  - 如右图所示，对于状态s1来说，有0.1的概率保持不变，有0.2的概率转移到s2状态，有0.7的概率转移到s4状态。
  - 以下是状态转移矩阵的一般形式，可据此试写出对应右图的状态转移矩阵：

$$\Pr=\begin{pmatrix} P(s_1|s_1) & P(s_2|s_1) & \cdots & P(s_N|s_1) \\ P(s_1|s_2) & P(s_2|s_2) & \cdots & P(s_N|s_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(s_1|s_N) & P(s_2|s_N) & \cdots & P(s_N|s_N) \end{pmatrix}$$



从马尔可夫过程到马尔可夫决策过程（Markov Decision Process，MDP）

- **马尔可夫决策过程（MDP）：**
  - 在马尔可夫过程的基础上引入了动作和奖励的概念，状态转移不仅由前一时刻的状态决定，也由前一时刻所采取的动作决定。
- **马尔可夫决策过程由一个四元组构成：** $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \Pr \rangle$
  - $\mathcal{S}$:状态的集合；
  - $\mathcal{A}(s)$：动作的集合，与状态s相关联；
  - $\mathcal{R}(s,a)$：奖励的集合（实时奖励函数/映射），$\mathcal{R}(s,a)$与动作和状态相关；
  - 状态转移概率图$\Pr$：其中元素代表在状态s处，采取动作a，状态转移到新状态s'的概率为$p(s'|s,a)$；
  - 此外，有奖励概率$p(r|s,a)$：在状态s下，采取行动a，得到即时奖励r的概率。

马尔可夫决策过程的退化情况：马尔可夫奖励过程（Markov Reward Process，MRP）

- **马尔可夫奖励过程**
  - 是在马尔可夫过程基础上增加了奖励函数R和衰减系数γ，用 四元组$\langle \mathcal{S}, \mathcal{R}, \Pr, \gamma \rangle$ 表示；
  - 折扣因子(Discount factor)，$\gamma \in [0, 1]$：系统沿时间的累计奖励中对未来奖励的折扣系数，用以权衡即时奖励和未来奖励的重要性；γ越靠近1，考虑的利益越长远。
  - 对比Markov决策过程，MRP缺少了动作集，因此，其状态转移概率和奖励概率只同当前状态和前一步状态有关： $p(s'|s)$ ， $p(r|s)$ 。
  - 期望奖励r_s：表示某一时刻的状态$s_t = s$在下一个时刻$(t+1)$能获得的奖励的期望值：

$$r_s = \mathrm{E}[r_{t+1} | s_t = s]$$

  - 累积奖励G_t （即回报函数）：G_t为在一个马尔科夫奖励链上从t时刻开始往后所有的奖励的有衰减的收益总和： $G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T$

Markov决策过程转换为Markov奖励过程的转换公式：

$$P_\pi(s'|s) = \sum_{a \in \mathcal{A}} \pi(a|s) \cdot P(s'|s, a)$$

马尔可夫决策过程的扩展情况：部分可观马尔可夫决策过程（Partially Observable Markov Decision Process，POMDP）

- **部分可观马尔可夫决策过程**
  - 在POMDP中，智能体无法观测到完整的环境状态，只能通过部分观测结果（观测集）来推断；
  - POMDP由一个七元组组成：$\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \Pr, \Omega, O, \gamma \rangle$，其中
    - $\mathcal{S}$是状态集，表示环境可能处于的所有状态（同MDP）；
    - $\mathcal{A}$是动作集，表示智能体可以执行的所有动作（同MDP）；
    - $\mathcal{R}$是奖励集合（函数/映射），表示在某个状态下执行某个动作所获得的即时奖励（同MDP）；
    - Pr是状态转移概率，表示在某个状态和动作下，环境转移到另一状态（下个状态）的概率（同MDP）；
    - Ω是观测集，表示智能体能够观测到的所有可能的观测结果；
    - O是观测概率映射$p(\omega|s,a)$，表示特定状态s下采取动作a后，获得某一观测结果ω的概率；
    - γ是奖励折扣因子。

贝尔曼方程（Bellman Equation）：平稳策略下的价值函数

- **状态转移轨迹**
  - MDP中的状态转移映射由 $s_t \xrightarrow{a_t} s_{t+1}, r_{t+1}$ 决定，其中，s_t，a_t，s_{t+1}，r_{t+1}均为随机变量
  - 状态转移轨迹：$s_t \xrightarrow{a_t} s_{t+1}, r_{t+1} \xrightarrow{a_{t+1}} s_{t+2}, r_{t+2} \xrightarrow{a_{t+2}} s_{t+3}, r_{t+3} \cdots$
  - 累积奖励（根据定义式）：$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T$
- **状态价值函数（State Value）**
  - 给定一个平稳策略$\pi(s) \to a: \Pr(a|s)$，我们可定义状态价值函数为从状态s起始的累积奖励的期望值：

$$v_\pi(s) \triangleq \mathrm{E}_\pi[G_t | s_t = s]$$

其中$v_\pi(s)$ 不仅依赖状态s 也依赖策略π，但$v_\pi(s)$ 作为期望并不依赖时间t。

- **动作价值函数（Action Value）**

$$q_\pi(s, a) \triangleq \mathrm{E}_\pi[G_t | s_t = s, a_t = a]$$

  - 动作价值函数也是一个期望，它不仅依赖起始状态s 和策略π，也依赖起始动作a。它同样不依赖时间。

贝尔曼方程（Bellman Equation）：状态价值函数

- **状态价值函数的计算**

$$
\begin{aligned}
&v_\pi(s)\\
&= \mathrm{E}_\pi[G_t|s_t = s] = \mathrm{E}_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots |s_t = s] = \mathrm{E}_\pi[r_{t+1} + \gamma G_{t+1}|s_t = s]\\
&= \mathrm{E}_\pi[r_{t+1}|s_t = s] + \gamma \mathrm{E}_\pi[G_{t+1}|s_t = s]\\
&= \sum_{a\in\mathcal{A}} \pi(a|s) \sum_{r\in\mathcal{R}} p(r|s,a) r + \gamma \sum_{s'\in\mathcal{S}} \mathrm{E}_\pi[G_{t+1}|s_t = s, s_{t+1} = s'] p(s'|s)\\
&= \underbrace{\sum_{a\in\mathcal{A}} \pi(a|s) \sum_{r\in\mathcal{R}} p(r|s,a) r} + \gamma \underbrace{\sum_{s'\in\mathcal{S}} v_\pi(s') p(s'|s)}
\end{aligned}
$$

前一部分不变　　后一部分从$\mathrm{E}_\pi[G_{t+1}|s_{t+1} = s']$展开

- **（由上式展开得）状态价值函数的最终表达式**

$$
v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s) \underbrace{\left[\sum_{r\in\mathcal{R}} p(r|s,a) r \right]} + \gamma \underbrace{\sum_{s'\in\mathcal{S}} p(s'|s,a) v_\pi(s')}, \quad \text{for all } s \in \mathcal{S}
$$

即时奖励的期望　　未来奖励的期望

贝尔曼方程（Bellman Equation）：状态价值函数（续）

- **状态价值函数**

$$v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s) \left[\sum_{r\in\mathcal{R}} p(r|s,a)r + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a)v_\pi(s') \right], \quad \text{for all } s \in \mathcal{S}$$

即时奖励的期望　　未来奖励的期望

- $v_\pi(s)$和$v_\pi(s)'$是待定的状态价值；$v_\pi(s)$的计算依赖于$v_\pi(s)'$；当状态转移概率矩阵$p(s'|s,a)$、奖励生成函数$p(r|s,a)$和策略概率$\pi(a|s)$已知时，对有限状态的MDP而言，所有状态集里的状态依上式联立成一个可解线性方程组：

$$v_\pi = r_\pi + \gamma P_\pi v_\pi，\text{其中}v_\pi\text{是待求解向量，}P_\pi\text{、}r_\pi\text{是已知向量}$$

- 在给定策略概率$\pi(a|s)$的情况下，上式的解被称为策略评估过程，上述方程成为贝尔曼策略方程（Bellman Policy Equation）。
- 状态转移概率矩阵$p(s'|s,a)$、奖励生成函数$p(r|s,a)$决定了系统模型；但在大多数任务场景中，我们并不知道系统模型是什么。

贝尔曼方程（Bellman Equation）：动作价值函数

- **动作价值函数的计算**
  - （回顾）累积回报函数：$\underbrace{\mathbb{E}[G_t|s_t=s]}_{v_\pi(s)}=\sum_{a\in\mathcal{A}}\underbrace{\mathbb{E}[G_t|s_t=s,a_t=a]}_{q_\pi(s,a)}\pi(a|s).$
  - 上式重写成如下形式：$v_\pi(s)=\sum_{a\in\mathcal{A}}\pi(a|s)q_\pi(s,a).$

即可代入如下状态价值函数（同上页）左端，

$$v_\pi(s)=\sum_{a\in\mathcal{A}}\pi(a|s)\left[\sum_{r\in\mathcal{R}}p(r|s,a)r+\gamma\sum_{s'\in\mathcal{S}}p(s'|s,a)v_\pi(s')\right],\quad \text{for all } s\in\mathcal{S}$$

$$\longrightarrow \sum_{a\in\mathcal{A}}\pi(a|s)q_\pi(s,a)=\sum_{a\in\mathcal{A}}\pi(a|s)\left[\sum_{r\in\mathcal{R}}p(r|s,a)r+\gamma\sum_{s'\in\mathcal{S}}p(s'|s,a)v_\pi(s')\right]$$

$$v_\pi(s)=\sum_{a\in\mathcal{A}}\pi(a'|s')q_\pi(s',a')$$

- **动作价值函数的最终表达式（用动作价值完全取代上式中的策略价值）**

$$q_\pi(s,a)=\sum_{r\in\mathcal{R}}p(r|s,a)r+\gamma\sum_{s'\in\mathcal{S}}p(s'|s,a)\sum_{a'\in\mathcal{A}(s')}\pi(a'|s')q_\pi(s',a'),$$

贝尔曼策略方程：动作价值和策略价值的相互转换

- **基于对同一回报函数期望的计算**
  - 由动作价值表示状态价值：

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \cdot q_\pi(s,a)$$

  - 由状态价值表示动作价值：

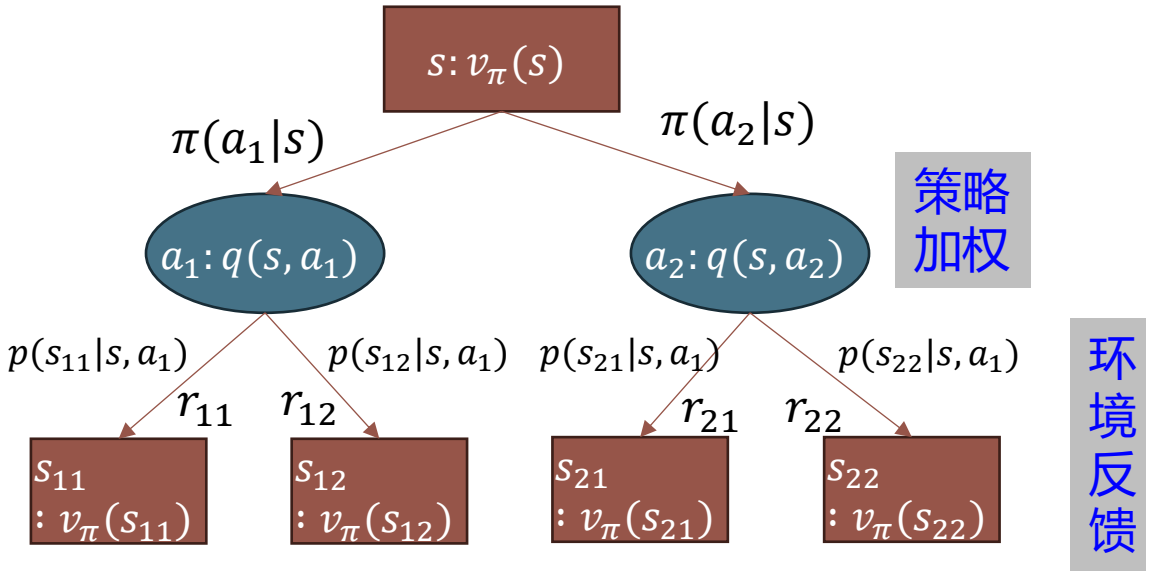
$$q_\pi(s,a) = \sum_{r,s'} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

- **上两式整理后，可得如下价值函数递归关系（Bellman方程）**

$$\begin{cases} v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \cdot \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')] \\ q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \sum_{a'} \pi(a'|s') \cdot q_\pi(s',a')\right] \end{cases}$$

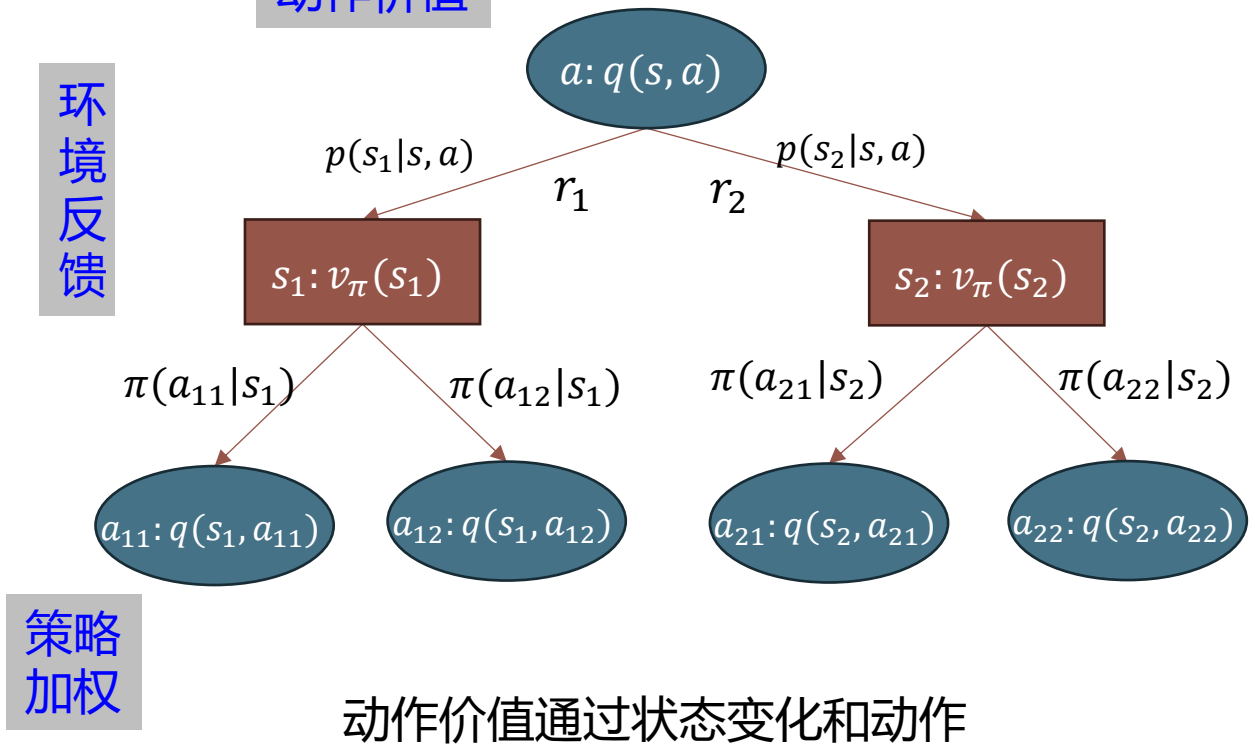
（续）相邻时刻的“状态-动作”间价值转换（Backup Diagram）

状态价值



状态价值通过动作选择和状态变化的递归依赖（转换关系）

动作价值



动作价值通过状态变化和动作选择的递归依赖（转换关系）

贝尔曼最优方程（Bellman Optimality Equation）

- **当策略生成函数$\pi(a|s)$不再固定时，我们希望求解出一个最优策略π^*，使得状态价值函数最大化。**
- **为此，构建一个以$\pi(a|s)$为决策变量的优化问题：**

$$
\begin{aligned}
v(s) &= \max_{\pi(s)\in\Pi(s)} \sum_{a\in\mathcal{A}} \pi(a|s) \left(\sum_{r\in\mathcal{R}} p(r|s,a)r + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a)v(s') \right) \\
&= \max_{\pi(s)\in\Pi(s)} \sum_{a\in\mathcal{A}} \pi(a|s)q(s,a),
\end{aligned}
$$

  - 此时$v(s)$和$v'(s)$都是未知（待求解）的中间参数；
  - $\pi(\cdot|s)$是最终待求解变量；
  - $q(s,a)$ 有如下形式：$q(s,a) \doteq \sum_{r\in\mathcal{R}} p(r|s,a)r + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a)v(s').$

在已知MDP模型（状态转移概率）的情况下求解贝尔曼最优方程

- **预备定理：压缩映射定理**
  - 一个点x称为不动点（fixed point），当$x = f(x)$;
  - 对于任何形式为$x = f(x)$的方程，如果f是一个压缩映射，那么存在一个$\gamma \in (0,1)$，使得

$$\|f(x_1) - f(x_2)\| \leq \gamma \|x_1 - x_2\|$$

  - 压缩映射满足不动点的存在性：x^*满足$f(x^*) = x^*$;
  - 压缩映射满足不动点的唯一性：不动点x^*是唯一的;

例子：

- x=0.5x，其中$f(x) = 0.5x, x \in \mathbb{R}, x^* = 0$是唯一的不动点，它可以通过以下方法迭代求解

$$x_{k+1} = 0.5x_k$$

- x=Ax，其中$f(x) = \mathrm{Ax}$，$x \in \mathbb{R}^n, A \in \mathbb{R}^{n \times n}$并且$\|\mathrm{A}\| < 1$，$x^* = 0$是唯一的固定点，它可以通过以下方法迭代求解

$$x_{k+1} = \mathrm{A}x_k$$

使用压缩映射求解Bellman最优方程（BOE）

- **压缩映射满足如下算法**
  - （压缩映射定理的一部分）考虑一个序列$\{x_k\}$，其中$x_{k+1} = f(x_k)$， f是一个压缩映射。当$k \to \infty$，$x_k \to x^*$ 对所有初始x_0成立，且压缩映射收敛速度是指数速度的。
- **求解贝尔曼最优方程**
  - 使用迭代方法：

$$v_{k+1} = f(v_k) = \max_{\pi}(r_\pi + \gamma P_\pi v_k)$$

求解Bellman最优公式，$f(v)$ 满足压缩映射$\| f(v_1) - f(v_2) \| \leq \gamma \| v_1 - v_2 \|$， γ是折扣系数。

  - 因此满足存在性，唯一性和算法，所以对于Bellman最优方程：

$$v = f(v) = \max_{\pi}(r_\pi + \gamma P_\pi v)$$

总存在一个最优方案v^*并且方案是唯一的。这个问题可以通过上述迭代求解：

$$v_{k+1} = f(v_k) = \max_{\pi}(r_\pi + \gamma P_\pi v_k)，\text{即}\pi^* = \arg\max_{\pi}(r_\pi + \gamma P_\pi v^*)$$

序列v_k在任意初始猜测值v_0下以指数方式快速的逼近于v^*。

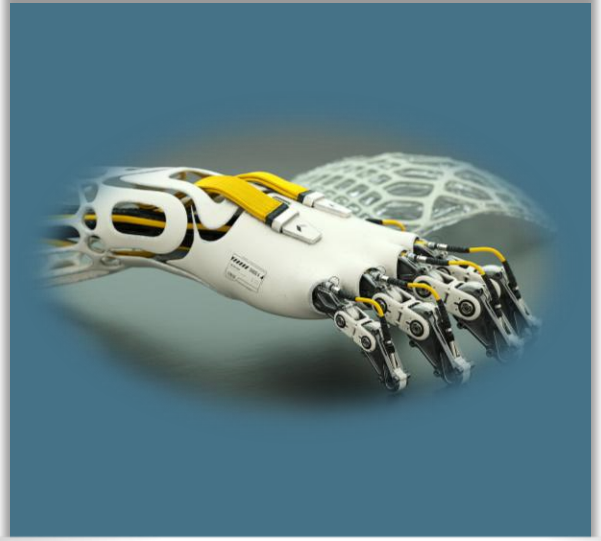
Bellman最优方程的解：值迭代算法

- **顺序迭代算法**
  - 根据值迭代公式： $v_{k+1} = f(v_k) = \max_{\pi}(r_\pi + \gamma P_\pi v_k)$
  - **步骤1**：策略更新 $\pi_{k+1} \leftarrow \arg\max_{\pi}(r_\pi + \gamma P_\pi v_k)$
  - **步骤3**：价值更新 $v_{k+1} \leftarrow r_{\pi_{k+1}} + \gamma P_{\pi_{k+1}} v_k$
  - 策略更新在展开后的状态价值函数上求解，有：

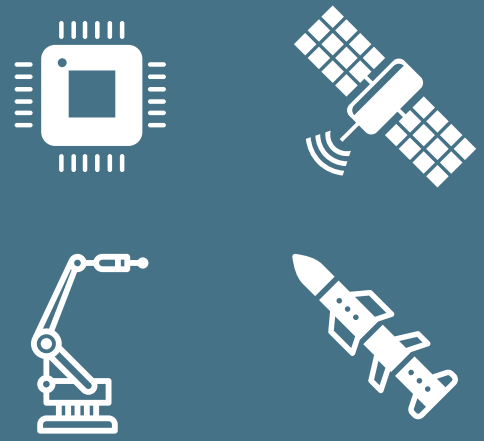
$$\pi_{k+1}(s) = \arg\max_{\pi} \sum_a \pi(a|s) \underbrace{\left(\sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v_k(s') \right)}_{q_k(s,a)}, \quad s \in \mathcal{S}$$

  - 上述优化问题的确定解（贪婪解）形式为

$$\pi^*(a|s) = \begin{cases} 1 & a = a_k^*(s) \\ 0 & a \neq a_k^*(s) \end{cases}$$ 其中，$a_k^*(s)$ 解为 $a_k^*(s) = \arg\max_a q_k(a,s)$



强化学习



1. 强化学习：概述
2. 强化学习的数学基础：马尔可夫决策过程（Markov Decision Process）
3. 无模型强化学习：时间差分算法和Q-Learning
4. 人工神经网络和强化学习：值函数近似
5. 函数近似下的策略梯度法

时序差分（Temporal Difference）

- **核心思想：在不知道模型信息的情况下，利用当前经验下的状态价值的估值，更新状态价值函数**
  - 给定策略π和当前经验：$\{s_0, a_1, s_1, r_1, \dots, s_t, a_t, s_{t+1}, r_{t+1}, \dots\}$；
  - 价值更新规则（其中t为学习步长）：

$$v_{t+1}(s_t) = v_t(s_t) - \alpha_t(s_t)\left[v_t(s_t) - \left(r_{t+1} + \gamma v_t(s_{t+1})\right)\right], \qquad \text{(a)}$$

$$v_{t+1}(s) = v_t(s), \quad \text{for all } s \neq s_t, \quad \text{只有当前访问的状态s的价值函数估计被更新}$$

其中：
$\alpha_t(s_t) \in (0,1]$是学习率；
r_{t+1}是即时奖励；
γ是折扣因子。

  - （特点）自举学习（Bootstrapping）：重复通过自身的估计值$v_t(s_{t+1})$和部分信息r_{t+1}，更新模型参数。
- **TD目标和TD误差**
  - 对本页顶部的（a）式而言：

$$\underbrace{v_{t+1}(s_t)}_{\text{新状态价值估计}} = \underbrace{v_t(s_t)}_{\text{当前状态价值估计}} - \alpha_t(s_t)\overbrace{[v_t(s_t) - \underbrace{(r_{t+1} + \gamma v_t(s_{t+1}))}_{\text{TD目标值}\bar{v}_t}]}^{\text{TD误差值}\delta_t}$$

时序差分和Bellman方程的关系

- **Bellman方程定义下的状态价值函数：**

$$v_\pi(s) = \mathbb{E}_\pi\left[r_{t+1} + \gamma v_\pi(s_{t+1}) \mid s_t = s\right]$$

  - 移项后可得：

$$\mathbb{E}_\pi\left[r_{t+1} + \gamma v_\pi(s_{t+1}) - v_\pi(s) \mid s_t = s\right] = 0 \qquad (a)$$

  即**TD 误差的期望为零**。

- **基于残差函数采样的“残差方程解”**
  - 根据（a）定义残差函数：$g(v_\pi(s)) \triangleq v_\pi(s) - \mathbb{E}[r_{t+1} + \gamma v_\pi(s_{t+1})|s_t = s]$　　它的期望=0
  - 在实际环境中，我们无法直接计算期望 $\mathbb{E}[\cdot]$，但可以获得采样样本s_{t+1}, r_{t+1}，用来替代期望值。
  - 展开$g(v_\pi(s))$，可得对其采样估计值$\tilde{g}(v_\pi(s))$有：

$$\tilde{g}(v_\pi(s)) = \underbrace{v_\pi(s) - \mathbb{E}[r_{t+1} + \gamma v_\pi(s_{t+1})|s_t = s]}_{\text{这是残差函数}} + \underbrace{\left(\mathbb{E}[r_{t+1} + \gamma v_\pi(s_{t+1})|s_t = s] - [r_{t+1} + \gamma v_\pi(s_{t+1})]\right)}_{\text{观测噪声}\eta}$$

根据这个带观测噪声的采样函数，我们可以用随机逼近的方式解出残差函数方程$g(v_\pi(s)) = 0$

时序差分和Bellman方程的关系（续）

- **残差函数方程$g(v_\pi(s))=0$的左端项采样估计：**

$$\tilde{g}(v_\pi(s)) = \underbrace{v_\pi(s) - \mathbb{E}[r_{t+1} + \gamma v_\pi(s_{t+1})|s_t = s]}_{\text{这是残差函数}} + \underbrace{(\mathbb{E}[r_{t+1} + \gamma v_\pi(s_{t+1})|s_t = s] - [r_{t+1} + \gamma v_\pi(s_{t+1})])}_{\text{观测噪声}\eta} \quad (a)$$

- **引入Robbins-Monro迭代方法求解**
  - Robbins-Monro求一个方程$h(\theta)=0$（w_t是观测噪声）的一般迭代方法：

$$\theta_{t+1} = \theta_t - \alpha_t\left(h(\theta_t) + w_t\right) \quad (b)$$

  - 将 (a) 代入 (b) 得到，$v_\pi(s_t) \leftarrow v_\pi(s_t) - \alpha_t \cdot \tilde{g}(v_\pi(s_t))$

$$\Rightarrow v(s_t) \leftarrow v(s_t) + \alpha \cdot \underbrace{(r_{t+1} + \gamma v(s_{t+1}) - v(s_t))}_{\text{TD误差}}$$

时序差分（TD）算法：状态价值更新

- **TD价值更新公式：**

$$v_{t+1}(s_t) = v_t(s_t) - \alpha_t(s_t)\left[v_t(s_t) - \left(r_{t+1} + \gamma v_t(s_{t+1})\right)\right]$$

TD误差值δ_t

新状态价值估计 $v_{t+1}(s_t)$

当前状态价值估计 $v_t(s_t)$

TD目标值$\bar{v}_t$

- TD目标$\bar{v}_t$：包含了新的环境反馈r_{t+1}，因此被认为是更接近$v(s_t)$真实值的估计，算法希望估计值$v_t(s_t)$能够向$\bar{v}_t$修正。
- TD误差：是当前价值估计与TD目标之间的差异，反映了当前估计（第t步）与最新信息（t+1步）之间的不一致程度——

$$\delta_t = v_t(s_t) - \left(r_{t+1} + \gamma v_t(s_{t+1})\right)$$

- 一步更新后的价值估计$v_{t+1}(s_t)$向TD目标值$\bar{v}_t$的逼近：

$$|v_{t+1}(s_t) - \bar{v}_t| = \underbrace{(1 - \alpha_t(s_t))}_{\text{缩放系数}} \cdot |v_t(s_t) - \bar{v}_t| < |v_t(s_t) - \bar{v}_t|$$

时序差分（TD）算法：状态价值更新（续）

- **单步TD更新：只依赖未来一步的奖励r_{t+1}与下一个状态s_{t+1}的价值函数估计更新对$v(s)$的估计**

$$v_{t+1}(s_t) = v_t(s_t) - \alpha_t(s_t)[v_t(s_t) - (r_{t+1} + \gamma v_t(s_{t+1}))]$$

- **多步TD更新：依赖未来n步的奖励值和状态s_{t+n}的价值估计进行更新**
  - 由单步TD扩展至n步：

$$\begin{aligned} v_{t+n}(s_t) &= v_t(s_t) - \alpha_t(s_t)[v_t(s_t) - G_{t:t+n}] \\ &= v_t(s_t) - \alpha_t(s_t)\left[v_t(s_t) - \left(r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n v_t(s_{t+n})\right)\right] \end{aligned}$$

- **一般化扩展：TD(λ)算法——引入 λ回报（λ-return）机制**
  - λ回报（λ-return）： $G_t^\lambda = (1-\lambda)\sum_{n=1}^{T-t-1} G_{t:t+n} + \lambda G_t$ （T-t+1表示从当前步数t开始，到终止步数T之间的步数）；
  - 将λ回报（λ-return）作为TD目标函数：

$$v_{t+n}(s_t) = v_t(s_t) - \alpha_t(s_t)\left[v_t(s_t) - G_t^\lambda\right]$$

时序差分（TD）：面向动作价值估计的迭代更新

- **SARSA算法：给定策略π，对动作价值函数进行估计的迭代算法**
  - 每次迭代要求使用$\{s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}\}$；
  - 上述序列对应“State-Action-Reward-State-Action”，也就是SARSA名称的来源；
  - SARSA算法的一步更新公式：将TD算法中的状态价值替换成动作价值：

$$\begin{aligned} v_{t+1}(s_t) &= v_t(s_t) - \alpha_t(s_t)\Big[v_t(s_t) - \big(r_{t+1} + \gamma v_t(s_{t+1})\big)\Big], \\ v_{t+1}(s) &= v_t(s), \quad \text{for all } s \neq s_t, \end{aligned}$$

原始TD算法

价值函数替换：动作价值函数

$$\begin{aligned} q_{t+1}(s_t, a_t) &= q_t(s_t, a_t) - \alpha_t(s_t, a_t)\Big[q_t(s_t, a_t) - \big(r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1})\big)\Big], \\ q_{t+1}(s, a) &= q_t(s, a), \quad \text{for all } (s, a) \neq (s_t, a_t), \end{aligned}$$

SARSA算法

时序差分（TD）：面向最优策略的SARSA方法

- **基于SARSA的最优策略更新算法**
  - 初始化：对所有状态-动作对 (s,a) 和所有时间步 t，设置学习率$\alpha_t(s_t,a_t)$和随机探索率$\epsilon \in (0,1)$，初始化动作价值函数$q_0(s_0,a_0)$；
  - 每回合（Episode）的迭代流程：
    - 根据$\pi_0(s_0)$生成动作a_0；
    - 在s_t,a_t下通过与环境的交互，收集经验样本$\{r_{t+1},s_{t+1},a_{t+1}\}$；
    - 更新q值：$q_{t+1}(s_t,a_t)=q_t(s_t,a_t)-\alpha_t(s_t,a_t)\left[q_t(s_t,a_t)-(r_{t+1}+\gamma q_t(s_{t+1},a_{t+1}))\right]$
    - 更新策略（ϵ-贪婪策略法）：

$$\begin{cases}\pi_{t+1}(a|s_t)=1-\dfrac{\epsilon}{|\mathcal{A}(s_t)|}(\mathcal{A}(s_t)-1) & \text{如果} \quad a=\arg\max_a q_{t+1}(s_t,a)\\ \pi_{t+1}(a|s_t)=\dfrac{\epsilon}{|\mathcal{A}(s_t)|} & \text{其他情况}\end{cases}$$

    - 更新状态与动作：$s_t \leftarrow s_{t+1}, a_t \leftarrow a_{t+1}$

时序差分（TD）：面向最优策略的Q-Learning方法

- **Q-learning算法**
  - Q-learning使用不同于SARSA的TD目标函数：

SARSA的TD目标函数

$(r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1}))$

对比

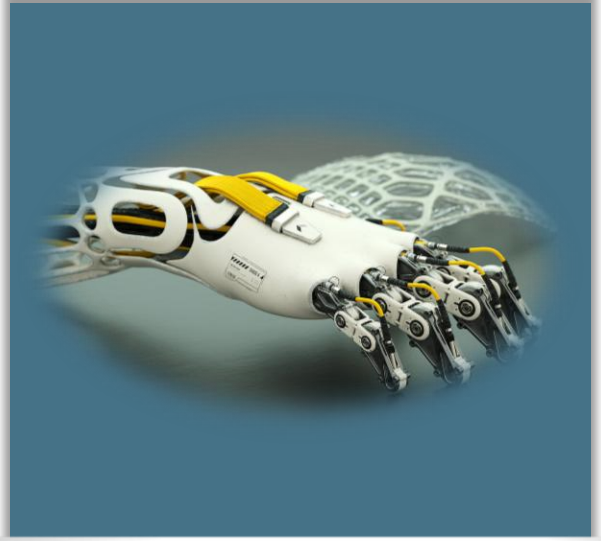
Q-learning的TD目标函数

$(r_{t+1} + \gamma \max_a q(s_{t+1}, a))$

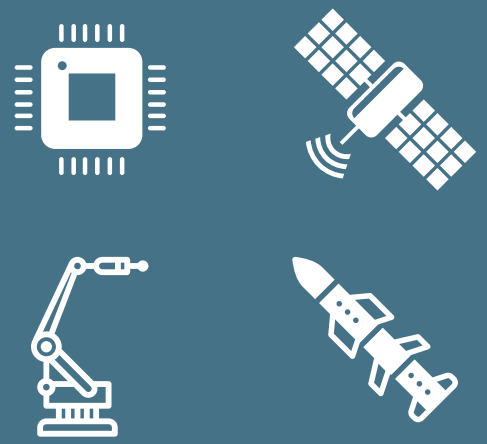
  - Q-learning的一步更新公式：

$$\begin{cases} q_{t+1}(s_t, a_t) = q_t(s_t, a_t) - \alpha_t(s_t, a_t)\left[q_t(s_t, a_t) - \left(r_{t+1} + \gamma \max\limits_{a \in \mathcal{A}(s_{t+1})} q_t(s_{t+1}, a)\right)\right] \\ q_{t+1}(s, a) = q_t(s, a), \quad \text{for all } (s, a) \neq (s_t, a_t), \end{cases}$$

- **Q-learning中的行为策略（behavior policy）和目标策略（target policy）**
  - 行为策略： 用于生成与环境交互的经验样本（轨迹）的策略；
  - 目标策略：不断向最优策略逼近的待更新策略；
  - SARSA使用行为策略更新TD-target，称为“在策略（On-policy）” 算法；
  - Q-learning使用目标策略（或非行为策略其他策略）更新TD-target，称为“离策略（Off-policy）” 算法。



强化学习



1. 强化学习：概述
2. 强化学习的数学基础：马尔可夫决策过程（Markov Decision Process）
3. 无模型强化学习：时间差分算法和Q-Learning
4. 人工神经网络和强化学习：值函数近似
5. 函数近似下的策略梯度法

Q-learning的表更新（Tabular Update）方法

- **离散状态-动作对下的Q-learning算法**
  - （回顾）在当前“状态-动作对“ (s_t, a_t) 下，对动作价值估计的更新：

$$q(s_t,a_t) \leftarrow q(s_t,a_t) - \alpha \left[\underbrace{q(s_t,a_t)}_{\text{预测值（predict\_Q）}} - \underbrace{(r_{t+1} + \gamma \cdot \max_a q(s_{t+1},a))}_{\text{目标值（target\_q）}} \right]$$

  - 表更新法：q-值表可以看做是一个二维矩阵，其行表示状态（State），列表示动作（Action），元素值 $q(s,a)$ 表示在状态 s 下选择动作 a 的预期累积奖励；
  - 表更新方式图示（t时刻下的Q表）：

	a_1	a_2	a_3	a_4
s_1	$q(s_t = s_1, a_t = a_1)$			
s_2			$q^*(s_{t+1} = s_1, a^* = a_3)$	
s_3				

更新q表单元值

目标值

值函数表达方式：从表更新到函数近似

- **离散状态-动作对下Q-learning表更新方法的局限性**
  - 存储开销：“状态-动作对“ (s_t, a_t) 对应的矩阵的维度随s_t和a_t 的维度呈指数增长；
  - 更新效率：每次计算 $\max_a q(s_{t+1}, a)$ 时，要遍历所有动作，在高维动作空间中计算开销大；
  - 表更新方式无法拓展到连续状态-动作空间。
- **广义线性函数拟合下的状态价值函数**
  - （回顾广义线性回归方法）引入广义线性函数拟合 $\hat{v}(s, \mathbf{w}) = \mathbf{w}^\top \mathbf{\Phi}(s)$
    - 其中，$\mathbf{\Phi}(s)$ 是基函数分量组成的特征向量；
    - w是可学习的参数向量。
  - 假设已知$\{v_\pi(s_i)\}_{i=1}^n$，则可以使用平方和误差（SSE）作为损失函数，构建一个典型的最小二乘问题：

$$J(\mathbf{w}) = \sum_{i=1}^{n} \left(v_\pi(s_i) - \hat{v}(s_i, \mathbf{w})\right)^2 \xrightarrow{\text{更一般地，有}} J(\mathbf{w}) = \mathbb{E}_\pi\left[\left(v_\pi(s) - \hat{v}(s, \mathbf{w})\right)^2\right]$$

  - 最小二乘解：$\mathbf{w} = \left(\mathbf{\Phi}^{\mathrm{T}}(s)\mathbf{\Phi}(s)\right)^{-1}\mathbf{\Phi}(s)\boldsymbol{v}_\pi$

状态价值未知情况下的函数近似方法

- **状态价值函数v_π已知的情况下**
  - 对损失函数 $J(\mathbf{w}) = \mathbb{E}_\pi\left[(v_\pi(s) - \hat{v}(s,\mathbf{w}))^2\right]$可以采取迭代更新的方法（梯度下降）：

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t \delta \nabla_{\mathbf{w}} \hat{v}(s_t, \mathbf{w}_t)$$

  - 其中，$\delta = v_\pi(s_t) - \hat{v}(s_t, \mathbf{w}_t)$，是价值函数估计的误差。
- **状态价值函数v_π未知的情况下，可以采取不同的价值函数估计形式代替误差函数中的$v_\pi(s)$**
  - 蒙特卡洛（Monte Carlo）法
    - 假设有一个完整回合的采样序列$\{s_1, r_1, s_2, r_2 \cdots\}$，则可使用累积回报$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots$作为$\boldsymbol{v}_\pi(s_t)$的近似：

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t (G_t - \hat{v}(s_t, \mathbf{w}_t)) \nabla_{\mathbf{w}} \hat{v}(s_t, \mathbf{w}_t)$$

  - 时间差分（Temporal Difference，TD）法
    - 通过自举学习（Bootstrapping），使用自身的估计值$\hat{v}(s_{t+1}, \mathbf{w}_t)$和及时奖励信息信息$r_{t+1}$作为$\boldsymbol{v}_\pi(s_t)$的近似：

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t \left(r_{t+1} + \gamma \hat{v}(s_{t+1}, \mathbf{w}_t) - \hat{v}(s_t, \mathbf{w}_t)\right) \nabla_{\mathbf{w}} \hat{v}(s_t, \mathbf{w}_t)$$

动作价值的函数近似方法下的TD-learning

- **动作价值函数近似 + SARSA算法**
  - 每次迭代要求使用序列$\{s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}\}$;
  - （回顾）SARSA算法中的基于Q表的动作价值更新方程：

$$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) - \alpha_t(s_t, a_t)\left[q_t(s_t, a_t) - (r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1}))\right],$$

$$q_{t+1}(s, a) = q_t(s, a), \quad \text{for all } (s, a) \neq (s_t, a_t),$$

  - 使用值函数近似的方式，用$\hat{q}(s_t, a_t, \mathbf{w})$代替表元素值$q(s_t, a_t)$，依旧采用梯度下降法更新$\mathbf{w}$：

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t \underbrace{\left(r_{t+1} + \gamma \hat{q}(s_{t+1}, a_{t+1}, \mathbf{w}_t)\right)}_{\text{TD目标函数}} - \hat{q}(s_t, a_t, \mathbf{w}_t)) \nabla_{\mathbf{w}} \hat{q}(s_t, a_t, \mathbf{w}_t)$$

  - 使用ϵ-贪婪策略法：

$$\begin{cases} \pi_{t+1}(a|s_t) = 1 - \dfrac{\epsilon}{|\mathcal{A}(s_t)|}(\mathcal{A}(s_t) - 1) \\ \pi_{t+1}(a|s_t) = \dfrac{\epsilon}{|\mathcal{A}(s_t)|} \end{cases}$$

基于近似函数得到贪婪动作

$$a = \arg\max_{a \in \mathcal{A}(s_t)} [\hat{q}_{t+1}(s_t, a, \mathbf{w}_t)]$$

动作价值的函数近似方法下的Q-learning

- **动作价值函数近似 + Q-learning算法**
  - （回顾）表更新的形式下，Q-learning使用不同于SARSA的TD目标函数：

SARSA的TD目标函数

$$(r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1}))$$

对比

Q-learning的TD目标函数

$$(r_{t+1} + \gamma \max_a q(s_{t+1}, a))$$

  - 使用值函数近似的方式，用$\max_a \hat{q}(s_{t+1}, a, \mathbf{w})$代替SARSA算法中的$\hat{q}(s_{t+1}, a_{t+1}, \mathbf{w})$，依旧采用梯度下降法更新$\mathbf{w}$：

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t \left(r_{t+1} + \gamma \max_{a \in \mathcal{A}(s_{t+1})} \hat{q}(s_{t+1}, a, \mathbf{w}_t) - \hat{q}(s_t, a_t, \mathbf{w}_t) \right) \nabla_{\mathbf{w}} \hat{q}(s_t, a_t, \mathbf{w}_t)$$

  - 可以和SARSA方法一样，使用ϵ-贪婪策略法平衡未知策略探索和最优策略的使用频率（公式略）。

使用人工神经网络：深度Q-learning算法

- **在前述的讨论中，我们并未指定各算法中，$\hat{q}(s_t, a_t, \mathbf{w})$或$\hat{v}(s_t, \mathbf{w})$的形式**
  - 显然，我们可以使用前馈神经网络等形式实现值函数的近似；
  - 回顾最早的基于已知策略、已知状态价值函数的近似函数参数最优化问题的损失函数设计：

$$J(\mathbf{w}) = \mathbb{E}_\pi \left[\left(v_\pi(s) - \hat{v}(s, \mathbf{w}) \right)^2 \right]$$

  - 在动作价值函数的体系下，使用如下Q-Learning式的参数更新（见前述讨论）

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t \left(r_{t+1} + \gamma \max_{a \in \mathcal{A}(s_{t+1})} \hat{q}(s_{t+1}, a, \mathbf{w}_t) - \hat{q}(s_t, a_t, \mathbf{w}_t) \right) \nabla_{\mathbf{w}} \hat{q}(s_t, a_t, \mathbf{w}_t)$$

相当于使用如下的损失函数设计形式，其中（$\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{s}'$）是一个有序对：

$$J(\mathbf{w}) = \mathbb{E} \left[\left(r + \gamma \max_{a \in \mathcal{A}(s')} \hat{q}(s', a, \mathbf{w}) - \hat{q}(s, a, \mathbf{w}) \right)^2 \right]$$

  - 深度Q-learning引入两个神经网络：用于值函数近似的主神经网络$\hat{q}(s, a, \mathbf{w})$，和更新频率较低的目标网络$\hat{q}(s, a, \mathbf{w}_T)$。

深度Q-learning的网络参数更新

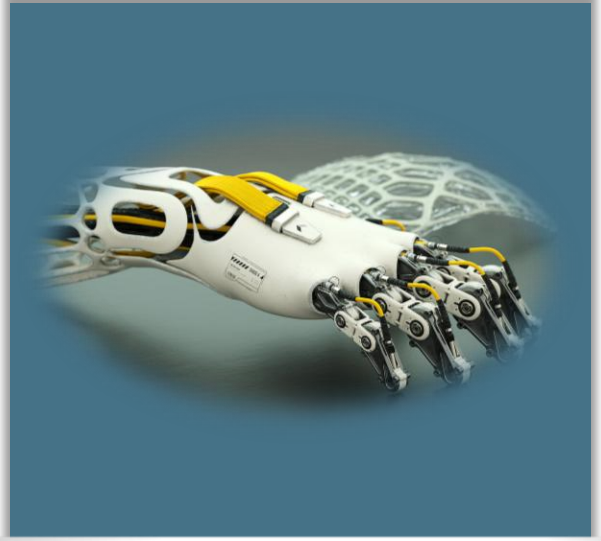
- **主神经网络（Main Network）和目标网络（Target Network）的参数更新**
  - 固定目标网络的参数$\mathbf{w}_T$，主神经网络梯度形式如下：

$$\nabla_{\mathbf{w}} J = -\mathbb{E}\left[\left(r + \gamma \max_{a \in \mathcal{A}(s')} \hat{q}(s', a, \mathbf{w}_T) - \hat{q}(s, a, \mathbf{w})\right) \nabla_{\mathbf{w}} \hat{q}(s, a, \mathbf{w})\right]$$

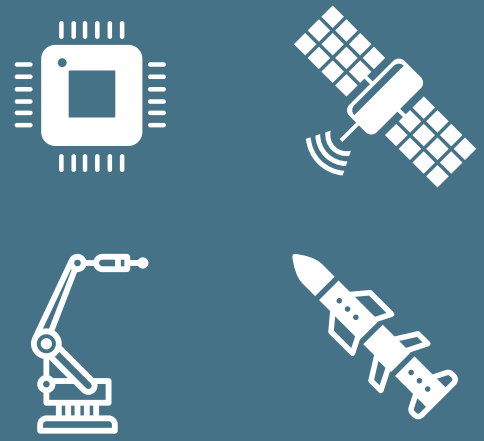
    - 更新梯度时，一般使用缓存mini-batch序列$\{(s_t, a_t, r_t, s_{t+1})\}$的形式计算$J$。
  - 主神经网络的输入为“状态-特征对”(s_t, a_t)，输出为动作价值的估计$\mathrm{y} = \hat{q}(s, a, \mathbf{w})$；
  - TD目标函数定义为：

$$y_T = r + \gamma \max_{a \in \mathcal{A}(s')} \hat{q}(s', a, \mathbf{w}_T)$$

  - 注意：目标网络$\hat{q}(s, a, \mathbf{w}_T)$本身不需要通过反向传播进行训练，而是每隔一定的步长，使用主干网络更新后的$\mathbf{w}$权重进行更新。
  - 新特性：经验回放池（Experience Replay Buffer）
    - 经验回放池中的数据由经验数据对$\{(s_t, a_t, r_t, s_{t+1})\}$有序排列组成；
    - 主干网络训练时的mini-batch由回放池中的数据对通过均匀分布随机抽取生成，即认为(s, a)符合均匀分布。



强化学习



1. 强化学习：概述
2. 强化学习的数学基础：马尔可夫决策过程（Markov Decision Process）
3. 无模型强化学习：时间差分算法和Q-Learning
4. 人工神经网络和强化学习：值函数近似
5. 函数近似下的策略梯度法

策略梯度法：使用带参函数π (a|s, w)估计策略本身

- **给定策略π下的平均状态价值函数**
  - d(s)：每个状态的价值权重（在给定策略**π**下的状态分布函数）
  - 状态的平稳分布函数d_π，对于给定概率转移图P_π而言： $d_\pi^T P_\pi = d_\pi^T$

$$\bar{v} = \sum_{s\in\mathcal{S}} d(s) v_\pi(s) = \mathbb{E}_{s\sim d}[v_\pi(s)]$$

对于带折扣奖励，有：

$$d_\pi(s) = \sum_{s'\in\mathcal{S}} d_0(s') \Pr_\pi(s|s')$$
$$= \sum_{s'\in\mathcal{S}} d_0(s') \sum_{k=0}^{\infty} \gamma^k [\Pr_\pi^k]_{s\to s'}$$

- **平均价值函数和累计奖励函数之间的关系**
  - 给定策略**π**(**w**)下沿时间收集奖励$\{r_{t+1}{}_{t=0}^{\infty}\}$，得到待优化目标函数

$$J(\mathbf{w}) = \lim_{n\to\infty} \mathbb{E}\left[\sum_{t=0}^{n} \gamma^t r_{t+1}\right] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1}\right]$$

  - 它和平均价值函数等价

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1}\right] = \sum_{s\in\mathcal{S}} d(s)\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1}|s_0 = s\right] = \sum_{s\in\mathcal{S}} d(s) v_\pi(s) = \bar{v}$$

最后一个等号写为矩阵形式：

$$\bar{\mathrm{v}}_\pi = d^T v_\pi$$

累计期望奖励函数（目标函数）的梯度

- **优化目标：最大化目标函数J(w)即期望累积奖励)**
- **目标函数J(w)对可学习参数w的梯度**

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \nabla_{\mathbf{w}} \pi(a|s, \mathbf{w}) q_\pi(s, a)$$

  - d(s) （状态分布）：表示在策略**π**下，智能体处于状态 s 的概率。这说明梯度不仅取决于动作的好坏，还取决于该状态出现的频率。
  - $\nabla_{\mathbf{w}} \pi(a|s, \mathbf{w})$ 策略梯度）：表示改变参数**w**会如何影响在状态 s 下选择动作 a 的概率。
- **梯度基于期望的紧凑形式**

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \mathbb{E}_{s \sim \eta, a \sim \pi(s, \mathbf{w})} \left[\nabla_{\mathbf{w}} \ln \pi(a|s, \mathbf{w}) q_\pi(s, a) \right]$$

  - 对数导数技巧：$\nabla_{\mathbf{w}} \ln \pi(a|s, \mathbf{w})$ 是 $\dfrac{\nabla_{\mathbf{w}} \pi(a|s, \mathbf{w})}{\pi(a|s, \mathbf{w})}$ 的简写，二者数学上等价。

累计期望奖励函数的梯度（续）

- **回顾：目标函数J(w)对可学习参数w的梯度**

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \nabla_{\mathbf{w}} \pi(a|s, \mathbf{w}) q_\pi(s, a) \qquad (a)$$

- 利用期望值的定义，上述公式（a）可写为

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \mathbb{E}_{S \sim d} \left[\sum_{a \in \mathcal{A}} \nabla_{\mathbf{w}} \pi(a|s, \mathbf{w}) q_\pi(s, a) \right]$$

- 将 $\nabla_{\mathbf{w}} \pi(a|s, \mathbf{w}) = \pi(a|s, \mathbf{w}) \nabla_{\mathbf{w}} \ln \pi(a|s, \theta)$ 代入（a），得到

$$\begin{aligned} \nabla_{\mathbf{w}} J(\mathbf{w}) &= \mathbb{E}_{S \sim d} \left[\sum_{a \in \mathcal{A}} \pi(a|s, \mathbf{w}) \nabla_{\mathbf{w}} \ln \pi(a|s, \mathbf{w}) q_\pi(s, a) \right] \\ &= \mathbb{E}_{s \sim d, a \sim \pi(s, \mathbf{w})} \left[\nabla_{\mathbf{w}} \ln \pi(a|s, \mathbf{w}) q_\pi(s, a) \right] \end{aligned}$$

蒙特卡洛策略梯度算法（REINFORCE）

- **根据目标函数J(w)对可学习参数w的梯度，构造梯度上升方法：**

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \nabla_{\mathbf{w}} J(\mathbf{w}_t)$$

- **根据策略梯度公式（展开），有**

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \mathbb{E}\left[\nabla_{\mathbf{w}} \ln \pi(a|s, \mathbf{w}_t) q_\pi(s, a)\right]$$

  - 由于真实期望未知，上述公式可由随机梯度（采样梯度）替代：

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \nabla_{\mathbf{w}} \ln \pi(a_t|s_t, \mathbf{w}_t) q_t(s_t, a_t)$$

    - 其中，$\nabla_{\mathbf{w}} \ln \pi(a_t|s_t, \mathbf{w}_t)$称为得分函数，它衡量了改变参数对选择动作概率的影响程度（对数概率）。
    - $q_t(s_t, a_t)$是动作价值估计函数，在蒙特卡洛框架下，可以用实际观测到的一整个轨迹（Episode）的累积回报估计。
    - $q_t(s_t, a_t)$可以拓展为优势估计（Advantage Estimate），它告诉估计函数（神经网络）：“在状态 s_t 选动作 a_t到底有多好？”，例如：$A = q_t(s_t, a_t) - b(s)$，其中$b(s)$ 称为baseline。

蒙特卡洛策略梯度算法（REINFORCE，续）

- **随机策略梯度公式**

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \nabla_{\mathbf{w}} \ln \pi(a_t|s_t, \mathbf{w}_t) q_t(s_t, a_t)$$

  - 可以进一步展开为：

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \beta_t \nabla_{\mathbf{w}} \pi(a_t|s_t, \mathbf{w}_t)$$

    - 其中，引入中间变量 $\beta_t = \dfrac{q_t(s_t, a_t)}{\pi(a_t|s_t, \mathbf{w}_t)}$ ，将更新项转化为了策略概率本身的梯度$\nabla_{\mathbf{w}} \pi(a_t|s_t, \mathbf{w}_t)$

    - $q_t(s_t, a_t)$是动作价值估计函数，在蒙特卡洛框架下，可以用实际观测到的一整个轨迹（Episode）的累积回报估计。

- **REINFORCE框架下，神经网络输出对连续动作空间和离散动作空间的不同处理**
  - 离散动作空间中，神经网络通常输出一个**概率分布向量**（Probabilities），其维度等于动作的总数（Softmax激活函数）。
  - 连续动作空间，策略通常用高斯分布表示，神经网络估计的往往是动作的均值估计。

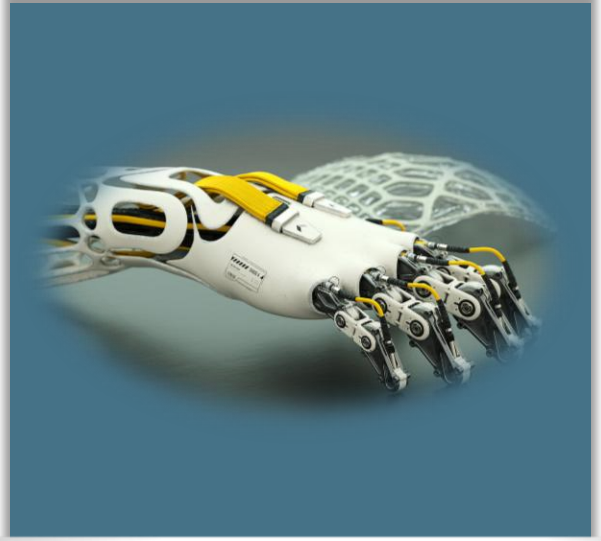
Actor-Critic（演员-评论家）算法

- **核心思想：结合“策略梯度”与“价值函数估计”**
  - REINFORCE算法：使用 蒙特卡洛 （Monte Carlo）学习来估计q值。需要等待回合结束，利用实际的累积回报G_t作为q的估计。
  - Actor-Critic算法：使用时序差分（Temporal-Difference, TD）学习估计q值。通过自举（Bootstrapping）的方式，利用下一步的价值估计来更新当前的价值，因此不需要等待回合结束。
- **极简Actor-Critic算法**
  - Critic：使用Sarsa/Q-learning 算法更新动作价值估计。通过参数w维护一个动作价值函数$q_t(s_t, a_t, \mathbf{w})$，根据环境的反馈（TD误差）来更新这个函数，例如：

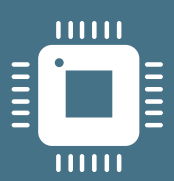
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \left(q(s_t, a_t, \mathbf{w}_t) - r_{t+1} + \gamma \max_{a \in \mathcal{A}(s_{t+1})} q(s_{t+1}, a, \mathbf{w}_t) \right) \nabla_{\mathbf{w}} q(s_t, a_t, \mathbf{w}_t)$$

  - Actor：使用策略梯度更新，通过参数θ，维护一个策略估计函数$\pi(s_t, a_t, \theta)$

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \ln \pi(a_t | s_t, \theta_t) q_t(s_t, a_t)$$



强化学习



参考文献：

[1] 赵世钰，强化学习的数学原理（英文版），清华大学出版社，2024.

[2] [美]劳拉·格雷泽，龚辉伦， 深度强化学习——基于Python的理论及实践，机械工业出版社，2020.