第4章：强化学习入门

周炜星 谢文杰

华东理工大学金融学系

2023年秋

纲要

纲要

强化学习简介

- 强化学习问题，是智能体学习问题，也是控制问题，更是优化问题。
- 随机优化源于各种各样的实际问题，如博弈游戏、供应链优化等。
- 金融资产或者证券的交易过程是一个典型的序贯决策过程，而强化学习算法擅长解决序贯决策问题，智能交易机器人最大化股票买卖交易的收益是一个典型的最优化问题。
- 金融数学中有一类非常特殊的随机优化问题：最佳止损问题，其中止损行为可能是出售金融资产或行使期权。

强化学习大师级人物

Andrew G. Barto教授和Richard S. Sutton 教授是著名强化学习大师，他们的经典教材《强化学习导论》（Reinforcement Learning: An Introduction）是入门深度强化学习的推荐书籍。2018年的第二版引入了很多新颖的应用案例分析，AlphaGo主要设计者David Silver对此做出了贡献。

纲要

马尔科夫决策过程定义

马尔科夫决策过程是强化学习模型的理论框架。在理解马尔科夫决策过程之前，我们需要了解什么是马尔科夫过程（Markov Process，MP），掌握马尔科夫过程的性质。

马尔科夫过程定义

如果离散随机过程满足：

- $\mathbb{P}(s_{t+1}|s_t) = \mathbb{P}(s_{t+1}|s_t, \ldots, s_0)$，

则该离散随机过程为马尔科夫过程。

在随机过程中，如果未来状态只与当前状态有关，而不受历史状态影响，则说明随机过程满足马尔科夫性，即为马尔科夫过程。

马尔科夫决策过程

$\mathbb{P}(s_{t+1}|s_t)$表示t时刻的状态s_t转移到$t+1$时刻的状态s_{t+1}的条件概率，$\mathbb{P}(s_{t+1}|s_t,\ldots,s_0)$表示在历史状态$s_t,\ldots,s_0$条件下$t+1$时刻转移到状态$s_{t+1}$的条件概率。两者相等说明转移概率不受历史状态信息影响，即与历史状态$s_{t-1},\ldots,s_0$无关，只与当前状态s_t有关。

随机过程满足马尔科夫性，离散状态之间**转移概率矩阵**P为：

$$
\begin{array}{cc}
 & \begin{array}{cccccc} s_1 & s_2 & s_3 & \cdots & s_{n-1} & s_n \end{array} \\
\begin{array}{c} s_1 \\ s_2 \\ s_3 \\ \vdots \end{array} &
\left(\begin{array}{cccccc}
p_{11} & p_{12} & p_{13} & \cdots & p_{1,n-1} & p_{1n} \\
p_{21} & p_{22} & p_{23} & \cdots & p_{2,n-1} & p_{2n} \\
p_{31} & p_{32} & p_{33} & \cdots & p_{3,n-1} & p_{3n} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots
\end{array}\right)
\end{array}
$$

状态转移概率矩阵

状态转移概率矩阵的元素p_{ij}表示从状态s_i转移到状态s_j的概率。状态转移概率矩阵有一些基本性质，如：

$$p_{ij} \geq 0. \tag{1}$$

$p_{ij} = 0$说明从状态s_i不能转移到状态s_j。状态转移概率矩阵的另一重要性质为：

$$\sum_{j=1}^{n} p_{ij} = 1. \tag{2}$$

状态转移概率矩阵在现实生活中随处可见，如城市之间人口流动概率矩阵等。

马尔科夫回报过程（Markov Reward Process，MRP）

在马尔科夫随机过程中，如果在状态转移时能够获得回报或收益，则此随机过程可以表示为马尔科夫回报过程（Markov Reward Process，MRP），定义如下：

马尔科夫回报过程

马尔科夫回报过程可以定义为一个四元组 $(\mathcal{S}, P, R, \gamma)$，其中:

- $\mathcal{S}$ 表示状态集合，
- $P : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$表示状态转移函数或状态转移矩阵，
- $R : \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{R}$表示回报函数，$\mathcal{R}$ 为连续区间，$R_{\max} \in \mathbb{R}^{+}$ (e.g., $[0, R_{\max}]$)，
- $\gamma \in [0, 1)$ 表示折扣系数。

马尔科夫决策过程（Markov Decision Process，MDP）

同样，在马尔科夫回报过程的基础上加入智能体行为$\mathcal{A}$，则可建模成离散**马尔科夫决策过程**，定义如下：

马尔科夫决策过程

马尔科夫决策过程可以表示为一个五元组 $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$，其中:

- $\mathcal{S}$ 表示状态集合，
- $\mathcal{A}$ 表示动作集合，
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$是状态转移函数，$P(s_t, a_t, s_{t+1})$是状态转移概率，
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$是回报函数，$\mathcal{R}$ 为连续区间，$R(s_t, a_t, s_{t+1}) \in \mathcal{R}$，$R_{\max} \in \mathbb{R}^+$ (e.g., $[0, R_{\max}]$),
- $\gamma \in [0, 1)$ 表示折扣系数。

纲要

动态规划

动态规划是求解马尔科夫决策过程的经典方法。马尔科夫决策过程五元组 $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$中，$P$表示状态转移概率。当状态转移概率和动作无关时，从当前t时刻状态s转移到$t+1$时刻状态s'，其状态转移概率可以写成

$$P_{ss'} = P(S_{t+1} = s' | S_t = s). \tag{3}$$

当状态转移概率与动作有关时，$P^a_{ss'}$定义为

$$P^a_{ss'} = P(S_{t+1} = s' | S_t = s, A = a), \tag{4}$$

其中，S_t 和 S_{t+1}表示相邻时间t和$t+1$时刻环境状态的随机变量，s为随机变量S_t的取值。$P^a_{ss'}$ 表示当前t时刻状态s下智能体选择动作a后（$a \in \mathcal{A}$），下一个时刻$t+1$转移到状态s'的概率。

策略函数

一般来说，复杂环境模型包括状态转移函数和回报函数。在问题求解之前，我们需要知道状态转移函数和回报函数，然后通过算法求解最优策略函数π。策略函数π可分成两种类型，一种是随机性策略，表示为：

$$\pi : \mathcal{S} \times \mathcal{A} \to [0, 1] \tag{5}$$

随机性策略函数π输出状态s下选择动作a的概率。智能体基于动作概率分布进行随机采样，得到最终动作a。另一种是确定性策略，表示为：

$$\pi : \mathcal{S} \to \mathcal{A}. \tag{6}$$

确定性策略函数π直接输出状态s下的动作a。从另一个角度而言，策略函数π也可以分成连续型策略和离散型策略。

状态转移函数

强化学习的目标是学习智能体的策略π，π可以建模成一个函数，将随机过程的状态空间映射到动作空间，表示为$\pi : \mathcal{S} \rightarrow \mathcal{A}$。动作影响状态转移和奖励回报。

在复杂环境模型已确定的情况下，智能体的策略输出的动作直接影响奖励回报，以随机策略函数举例分析，将状态转移函数重写为：

$$P_{ss'}^{\pi} = \sum_{a \in A} \pi(a|s) P_{ss'}^{a}, \tag{7}$$

其中，$\pi(a|s)$表示状态s下执行动作行为a的概率。

奖励函数

奖励函数或回报函数 R 决定了智能体在环境状态s下执行行动作a后得到的奖励值R_s^a，可以表示为：

$$R_s^a = \mathbb{E}[R_t | S_t = s, A_t = a]. \tag{8}$$

我们结合策略函数π，可以计算智能体在当前时刻状态s下选择不同动作后获得的期望奖励回报：

$$R_s^\pi = \sum_{a \in A} \pi(a|s) R_s^a. \tag{9}$$

累积回报

智能体从当前时刻状态s开始直至终止状态所获得的**累积奖励回报**可定义为：

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots + \gamma^T R_{t+T} = \sum_{k=0}^{T} \gamma^k R_{t+k}, \quad (10)$$

其中，γ是折扣系数，且$\gamma \in [0,1)$。强化学习的折扣系数与金融分析中的折扣因子类似。因为$\gamma < 1$，G_t不会出现无穷大。智能体在无限长时间的累积收益情况的具体分析如下：

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k} < R_{\max} \sum_{k=0}^{\infty} \gamma^k = R_{\max} \frac{1}{1-\gamma}, \quad (11)$$

其中，$R_{\max}$表示最大即时奖励值。

状态值函数

我们在累积回报G_t基础上，可以定义状态值函数$V_\pi(s)$。$V_\pi(s)$表示从状态s出发，智能体基于当前策略函数π获得的期望回报，具体数学表示为：

$$V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]. \tag{12}$$

状态值函数$V_\pi(s)$是智能体在状态s获得累积回报G_t的期望。对状态值函数$V_\pi(s)$进行简单推导，可以得到

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots |S_t = s] \\ &= \mathbb{E}_\pi[R_t + \gamma(R_{t+1} + \gamma^1 R_{t+2} + \gamma^2 R_{t+3} + \cdots)|S_t = s] \\ &= \mathbb{E}_\pi[R_t + \gamma G_{t+1}|S_t = s] \\ &= \mathbb{E}_\pi[R_t + \gamma V_\pi(S_{t+1})|S_t = s]. \end{aligned} \tag{13}$$

S_{t+1}为随机变量，G_{t+1}的期望值用状态值函数$V_\pi(S_{t+1})$替换。

状态-动作值函数

同样地定义状态-动作值函数$Q_\pi(s,a)$。$Q_\pi(s,a)$ 表示智能体从状态s出发，基于当前策略函数π执行动作a能够获得的期望累积回报，衡量了状态s下动作a的价值，具体数学表达式为：

$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]. \tag{14}$$

对状态-动作值函数$Q_\pi(s,a)$进行简单推导，可以得到：

$$\begin{aligned} Q_\pi(s) &= \mathbb{E}_\pi[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots |S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_t + \gamma(R_{t+1} + \gamma^1 R_{t+2} + \gamma^2 R_{t+3} + \cdots)|S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_t + \gamma G_{t+1}|S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_t + \gamma Q_\pi(S_{t+1})|S_t = s, A_t = a]. \end{aligned} \tag{15}$$

S_{t+1}为随机变量，G_{t+1}的期望值用动作-状态值函数$Q_\pi(S_{t+1})$替换。

状态-动作值函数与状态值函数的关系

基于状态值函数和状态-动作值函数的定义，可以发现两者之间具有紧密的联系：

$$V_\pi(s) = \sum_{a \in A} \pi(a|s) Q_\pi(s, a). \tag{16}$$

上式说明状态s的值函数$V_\pi(s)$是在策略函数π下执行动作a获得累积收益回报的期望值。$\pi(a|s)$表示状态s下动作a的概率，状态-动作值函数$Q_\pi(s, a)$表示状态s下动作a的期望累积收益，因此，智能体在状态s下遍历所有动作并累积期望收益$\pi(a|s)Q_\pi(s, a)$，得到了状态s的价值$V_\pi(s)$。

状态-动作值函数与状态值函数的关系

我们也可以将两者之间的紧密联系表示为:

$$Q_\pi(s,a) = R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a V_\pi(s'). \tag{17}$$

式（17）说明，状态-动作值函数等于动作a的即时奖励值加上下一个可能状态s'的值函数$V_\pi(s')$的加权和$\sum_{s'\in S} P_{ss'}^a V_\pi(s')$。由于$V_\pi(s')$是下一个时刻状态值，我们需要乘上折扣因子$\gamma$。

将式（17）代入式（16），可以得到状态值函数另外一种更加复杂的表示形式:

$$V_\pi(s) = \sum_{a\in A} \pi(a|s)\left(R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a V_\pi(s')\right). \tag{18}$$

上式只包含了状态值函数V_π，无状态-动作值函数Q_π，此方程是求解状态值函数$V_\pi(s)$的关键。

状态-动作值函数与状态值函数的关系

类似地，将式（16）代入式（17），可以得到状态-动作值函数另外一种更加复杂的表示形式：

$$Q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \left(\sum_{a' \in A} \pi(a'|s') Q_\pi(s',a') \right). \tag{19}$$

上式只包含了状态-动作值函数Q_π，无状态值函数V_π，此方程是求解动作-状态值函数$Q_\pi(s,a)$的关键。

Bellman方程

在马尔科夫回报过程中，关于状态值函数$V_\pi(s)$的Bellman方程可以表示为：

$$V_\pi(s) = \mathbb{E}_\pi[R_t + \gamma V_\pi(S_{t+1}) | S_t = s]. \tag{20}$$

状态值函数$V_\pi(s)$的Bellman方程不包含策略函数和动作。

Bellman方程

在马尔科夫决策过程中，关于状态值函数$V_\pi(s)$的Bellman方程可以表示为：

$$
\begin{aligned}
V_\pi(s) &= \sum_{a\in A} \pi(a|s)\left(R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a V_\pi(s')\right) \\
&= \sum_{a\in A} \pi(a|s) R_s^a + \gamma \sum_{a\in A} \pi(a|s)\left(\sum_{s'\in S} P_{ss'}^a V_\pi(s')\right).
\end{aligned} \tag{21}
$$

Bellman方程

将式（9），即$R_s^\pi = \sum_{a\in A}\pi(a|s)R_s^a$，代入上式，可以得到：

$$\begin{aligned} V_\pi(s) &= R_s^\pi + \gamma\sum_{a\in A}\pi(a|s)\left(\sum_{s'\in S}P_{ss'}^a V_\pi(s')\right) \\ &= R_s^\pi + \gamma\sum_{s'\in S}\left(\sum_{a\in A}\pi(a|s)P_{ss'}^a\right)V_\pi(s'). \end{aligned} \tag{22}$$

将式（7），即$P_{ss'}^\pi = \sum_{a\in A}\pi(a|s)P_{ss'}^a$，代入上式，可以得到：

$$V_\pi(s) = R_s^\pi + \gamma\sum_{s'\in S}P_{ss'}^\pi V_\pi(s'). \tag{23}$$

Bellman方程

针对马尔科夫决策过程状态空间中的每一个状态$s_1, s_2, \cdots$，都可以写出类似的Bellman方程：

$$
\begin{aligned}
V_\pi(s_1) &= R_{s_1}^\pi + \gamma \sum_{s' \in S} P_{s_1 s'}^\pi V_\pi(s') \\
V_\pi(s_2) &= R_{s_2}^\pi + \gamma \sum_{s' \in S} P_{s_2 s'}^\pi V_\pi(s') \\
&\cdots \\
V_\pi(s_n) &= R_{s_n}^\pi + \gamma \sum_{s' \in S} P_{s_n s'}^\pi V_\pi(s').
\end{aligned}
\tag{24}
$$

Bellman方程

我们可以将上述方程组改写成矩阵形式：

$$\begin{bmatrix} V_\pi(s_1) \\ V_\pi(s_2) \\ V_\pi(s_3) \\ \vdots \\ V_\pi(s_n) \end{bmatrix} = \begin{bmatrix} R_\pi(s_1) \\ R_\pi(s_2) \\ R_\pi(s_3) \\ \vdots \\ R_\pi(s_n) \end{bmatrix} + \gamma \begin{bmatrix} P_{11}^\pi & P_{12}^\pi & \cdots & P_{1n}^\pi \\ P_{21}^\pi & P_{22}^\pi & \cdots & P_{2n}^\pi \\ P_{31}^\pi & P_{32}^\pi & \cdots & P_{3n}^\pi \\ \vdots & \vdots & \cdots & \vdots \\ P_{n1}^\pi & P_{n2}^\pi & \cdots & P_{nn}^\pi \end{bmatrix} \begin{bmatrix} V_\pi(s_1) \\ V_\pi(s_2) \\ V_\pi(s_3) \\ \vdots \\ V_\pi(s_n) \end{bmatrix}. \tag{25}$$

因此，我们可以进一步用矩阵符号表示：

$$\mathrm{V}_\pi = \mathrm{R}_\pi + \gamma \mathrm{P}_\pi \mathrm{V}_\pi, \tag{26}$$

其中，V_π和R_π为列向量，P_π为状态转移概率矩阵。

Bellman方程

求解状态值函数的Bellman方程组（26），可得：

$$\begin{aligned} V_\pi &= R_\pi + \gamma P_\pi V_\pi \\ (I - \gamma P_\pi) V_\pi &= R_\pi \\ V_\pi &= (I - \gamma P_\pi)^{-1} R_\pi . \end{aligned} \tag{27}$$

因此，状态值函数V_π可以基于状态转移函数$P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$和回报奖励函数$R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$直接求解。

策略迭代算法

在马尔科夫决策过程的状态值函数V_π解析公式

$$V_\pi = (I - \gamma P_\pi)^{-1} R_\pi$$

中：

- 解析解存在需要矩阵$(I - \gamma P_\pi)$可逆。
- 对于现实问题，复杂随机过程的状态转移概率矩阵不一定满足$(I - \gamma P_\pi)$可逆。
- 即使逆矩阵存在，由于马尔科夫决策过程的状态数量多，状态转移矩阵规模大。
- 矩阵求逆计算复杂度较高。
- 超大规模矩阵逆计算在有限时间和有限内存资源条件下也基本不可能完成。

策略迭代算法

因此，我们可以考虑采用数值方法中的迭代方法求解此类问题，迭代公式表示如下：

$$\mathrm{V}_{k+1} = \mathrm{R}_{\pi} + \gamma \mathrm{P}_{\pi} \mathrm{V}_{k}. \tag{28}$$

迭代公式为了求出状态值函数，用等式右边的状态值函数V_k计算出等式左边的状态值函数V_{k+1}后，继续将V_{k+1}代入等式右边，迭代计算状态值函数V_{k+2}，以此类推，循环迭代。

迭代公式的矩阵形式展开后可以得到每一个状态值函数迭代公式为

$$V_{k+1}(s) = \sum_{a \in A} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_k(s') \right). \tag{29}$$

状态值函数迭代过程为策略评估，即给定策略函数π，可以估计各个状态值函数。

策略迭代算法

通常地，状态值函数的初始值都设置成0，即$V_0 = 0$。我们通过迭代公式计算状态值函数V_k，直至收敛到V^*，则V^*为迭代公式的不动点，且不动点V^*必定满足：

$$V^* = R_\pi + \gamma P_\pi V^*. \tag{30}$$

将式(28)减去式(30)，可得：

$$V_{k+1} - V^* = \gamma P_\pi (V_k - V^*). \tag{31}$$

用e_k表示第k步数值误差，定义为数值解V_k与精确解V^*之差：

$$e_k = V_k - V^*, \tag{32}$$

策略迭代算法

同样，e_{k+1}表示第$k+1$步数值误差，定义为第$k+1$步数值解V_{k+1}与精确解V^*之差：

$$e_{k+1} = V_{k+1} - V^*, \tag{33}$$

我们将上述两个误差定义公式代入式(31)，可以得到迭代公式：

$$e_{k+1} = \gamma P_\pi e_k. \tag{34}$$

进一步迭代计算可得：

$$e_k = \gamma^k (P_\pi)^k e_0. \tag{35}$$

基于状态转移概率矩阵P_π的性质和折扣系数$\gamma < 1$，可以得到：

$$\lim_{k\to\infty} e_k = 0. \tag{36}$$

即当迭代次数足够多时，误差趋于0，此时数值解V_k收敛到精确解V^*。

策略改进

在定义模型值函数后，智能体通过值函数得到最优策略。策略改进的过程中可以采用贪心策略：

$$\pi_{k+1}(s) = \arg\max_a Q_{\pi_k}(s, a), \tag{37}$$

其中，$Q_{\pi_k}(s, a)$表示智能体在状态s下动作a的价值，状态-动作值函数的下标π_k表示当前策略函数：

$$Q_{\pi_k}(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_{\pi_k}(s'). \tag{38}$$

将其代入式（37）后，可得：

$$\pi_{k+1}(s) = \arg\max_a \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_{\pi_k}(s') \right). \tag{39}$$

此过程叫做策略改进，基于给定的值函数$V_{\pi_k}(s)$改进策略函数。

策略迭代算法的具体迭代过程伪代码

Algorithm 8: 策略迭代算法伪代码

Input: 马尔可夫决策过程五元组 $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$

Output: 最优策略 π^*

初始化状态值函数 $V(s) = 0$，初始化策略函数 π 为随机策略

for $k = 0, 1, 2, 3, \cdots$ **do**

    % 策略评估

    for $l = 0, 1, 2, 3, \cdots$ **do**

        for $s \in S$ **do**

            $V'(s) = \sum_{a\in A} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_k(s')\right)$

        if $V' == V$ **then**

            停止迭代;

        else

            $V = V'$

    % 策略改进

    for $s \in S$ **do**

        $\pi'(s) = \arg\max_{a\in A} \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_k(s')\right)$

    % 策略迭代终止判断

    if $\pi' == \pi$ **then**

        停止迭代

    else

        $\pi = \pi'$

$\pi^* = \pi'$

值函数迭代算法

值函数迭代算法直接求解马尔科夫决策过程中的值函数，智能体基于值函数选择最优动作。迭代算法直接进行值函数迭代，值函数迭代算法无策略改进过程，值函数迭代更新公式如下：

$$V_{k+1}(s) = \max_{a\in A}\left(R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a V_k(s')\right), \tag{40}$$

公式中的$\max_{a\in A}$操作是值函数迭代算法的关键。直接使用当前值函数的最大值来更新值函数，此过程类似如下操作：

$$V_{k+1}(s) = \max_{a\in A} Q_k(s,a), \tag{41}$$

其中，

$$Q_k(s,a) = R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a V_k(s'). \tag{42}$$

值函数迭代算法

换言之，值函数迭代算法将策略改进过程融入了值函数迭代过程，其关键操作为值函数更新中的$\max_{a\in A}$操作。迭代过程收敛后得到最终值函数，进而求得最优策略函数：

$$\pi(s) = \arg\max_{a\in A}\left(R_s^a + \gamma\sum_{s'\in S} P_{ss'}^a V_k(s')\right). \tag{43}$$

因此，值函数迭代算法中的关键步骤为值函数更新中的$\max_{a\in A}$操作，$\max_{a\in A}$操作需要遍历动作空间中的所有动作a，并选择期望累积收益最大的动作。值函数更新是基于期望收益最大的动作更新，具体迭代过程如值函数迭代算法伪代码：

值函数迭代算法伪代码

Algorithm 9: 值函数迭代算法伪代码

Input: 马尔可夫决策过程五元组 $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$

Output: 最优策略 π^*

```
初始化状态值函数 V(s) = 0，以及收敛阈值 Δ_V
for k = 0, 1, 2, 3, ··· do
    % 值函数迭代
    for l = 0, 1, 2, 3, ··· do
        for s ∈ S do
            V'(s) = max_{a∈A} R_s^a + γ Σ_{s'∈S} P_{ss'}^a V(s')
        if |V' − V| < Δ_V then
            停止迭代;
        else
            V = V'
% 计算最优策略
for s ∈ S do
    π'(s) = argmax_{a∈A} R_s^a + γ Σ_{s'∈S} P_{ss'}^a V(s')
```

纲要

蒙特卡罗估计

如果马尔科夫决策过程不包含模型状态转移函数或不存在显式的状态转移函数等，那么动态规划方法（策略迭代和值函数迭代算法）不具有可行性，只能基于状态值函数或者状态-动作值函数的初始定义分析。假设T步动作的状态值函数定义如下:

$$V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[R_t + \gamma R_{t+1} + \cdots + \gamma^{T-1} R_{t+T-1}|S_t = s]. \tag{44}$$

一般而言，无穷步动作的状态值函数可定义为:

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi[G_t|S_t = s] \\ &= \mathbb{E}_\pi[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots |S_t = s] \\ &= \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k}|S_t = s]. \end{aligned} \tag{45}$$

蒙特卡罗估计

同时，状态-动作值函数定义为:

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}_\pi[G_t|S_t=s,A_t=a] \\ &= \mathbb{E}_\pi[R_t+\gamma R_{t+1}+\gamma^2 R_{t+2}+\gamma^3 R_{t+3}+\cdots|S_t=s,A_t=a] \\ &= \mathbb{E}_\pi[\sum_{k=0}^{\infty}\gamma^k R_{t+k}|S_t=s,A_t=a]. \end{aligned} \tag{46}$$

上述公式基于数学期望定义了状态值函数和状态-动作值函数。在统计分析过程中，期望可以通过随机采样样本的均值进行估计和近似。在实际计算中，我们需要基于策略π采样得到一些样本，代入公式中估计出均值大小，近似（估计）值函数的期望值，此为蒙特卡罗方法的重要思想。

蒙特卡罗估计

因此，我们通过蒙特卡罗方法进行采样，得到一些**完整**的轨迹数据（Trajectory或Episode）：

$$
\begin{aligned}
&\text{Episode 1}: \langle s_0, a_{10}, r_{10}, s_{11}, a_{11}, r_{11}, .., s_{1T}, a_{1T}, r_{1T} \rangle \\
&\text{Episode 2}: \langle s_0, a_{20}, r_{20}, s_{21}, a_{21}, r_{21}, .., s_{2T}, a_{2T}, r_{2T} \rangle \\
&\text{Episode 3}: \langle s_0, a_{30}, r_{30}, s_{31}, a_{31}, r_{31}, .., s_{3T}, a_{3T}, r_{3T} \rangle \\
&\qquad\qquad \vdots \\
&\text{Episode } n: \langle s_0, a_{n0}, r_{n0}, s_{n1}, a_{n1}, r_{n1}, .., s_{nT}, a_{nT}, r_{nT} \rangle
\end{aligned}
\tag{47}
$$

蒙特卡罗估计

我们从状态s_0开始随机采样了n条完整的轨迹数据，运用蒙特卡罗方法估计状态s_0的状态值函数。经验轨迹i中状态s_0对应的累积奖励值为：

$$G_i = \sum_{t=0}^{T} r_{it}, \tag{48}$$

其中$i \in \{1, 2, \cdots, n\}$。我们计算$n$条完整轨迹的平均累积回报值作为状态$s_0$的状态值函数的估计：

$$V_\pi(s_0) = \frac{1}{n}\sum_{i=1}^{n} G_i. \tag{49}$$

上述$V_\pi(s)$的估计过程简化了对轨迹样本的统计分析。在实际计算过程中，可以进一步考虑**初访和每访**的区别。

蒙特卡罗估计

一般情况下，蒙特卡罗算法采用增量值更新状态值函数：

$$
\begin{aligned}
V_n(s) &= \frac{1}{n}\sum_{i=1}^{n} G_i \\
&= \frac{1}{n}\sum_{i=1}^{n-1} G_i + \frac{1}{n} G_n \\
&= \frac{1}{n}(n-1)\frac{1}{(n-1)}\sum_{i=1}^{n-1} G_i + \frac{1}{n} G_n \\
&= \frac{1}{n}(n-1)V_{n-1}(s) + \frac{1}{n} G_n \\
&= V_{n-1} + \frac{1}{n}(G_n - V_{n-1}(s)).
\end{aligned} \tag{50}
$$

蒙特卡罗估计

因此，我们将公式中$\frac{1}{n}$替换成α，状态s在第n次采样后，更新状态值函数：

$$V_n(s) = V_{n-1}(s) + \alpha(G_n - V_{n-1}(s)), \tag{51}$$

其中α为机器学习中的学习率。

蒙特卡罗估计

同样，我们也可以估计状态-动作值函数$Q(s, a)$。我们从状态s_0开始选择动作a_0，然后随机采样了n条完整的轨迹数据：

$$
\begin{aligned}
\text{Episode 1} &: \langle s_0, a_0, r_{10}, s_{11}, a_{11}, r_{11}, .., s_{1T}, a_{1T}, r_{1T} \rangle \\
\text{Episode 2} &: \langle s_0, a_0, r_{20}, s_{21}, a_{21}, r_{21}, .., s_{2T}, a_{2T}, r_{2T} \rangle \\
\text{Episode 3} &: \langle s_0, a_0, r_{30}, s_{31}, a_{31}, r_{31}, .., s_{3T}, a_{3T}, r_{3T} \rangle \\
&\vdots \\
\text{Episode } n &: \langle s_0, a_0, r_{n0}, s_{n1}, a_{n1}, r_{n1}, .., s_{nT}, a_{nT}, r_{nT} \rangle
\end{aligned}
\tag{52}
$$

蒙特卡罗估计

运用蒙特卡罗方法估计状态s_0时选择动作a_0的状态-动作值函数，计算随机采样的n条完整的轨迹数据的平均累积回报：

$$Q_\pi(s_0, a_0) = \frac{1}{n}\sum_{i=1}^{n} G_i. \tag{53}$$

同样，蒙特卡罗算法采用增量值更新状态-动作值函数：

$$Q_n(s, a) = Q_{n-1}(s, a) + \alpha(G_n - Q_{n-1}(s, a)), \tag{54}$$

其中，α为学习率。获得状态-动作值函数$Q(s, a)$后，可以计算最优化策略函数：

$$\pi(s) = \arg\max_a Q(s, a). \tag{55}$$

蒙特卡罗估计

蒙特卡罗强化学习算法也是一个迭代算法，每次采样都使用最新的状态-动作值函数$Q(s,a)$来构建策略函数。在实际采样过程中，为了增加随机采样完整轨迹的多样性，一般并不完全按照最新的状态-动作值函数$Q(s,a)$进行采样，而是采用ϵ-贪心算法，具体采样策略为：

$$\pi(s,a)=\begin{cases}1-\epsilon+\frac{\epsilon}{|\mathcal{A}|}, & a=\arg\max_a Q(s,a)\\ \frac{\epsilon}{|\mathcal{A}|}, & a\neq \arg\max_a Q(s,a)\end{cases}, \tag{56}$$

其中，$|\mathcal{A}|$表示动作空间$\mathcal{A}$的大小，即动作数量；ϵ决定了智能体的探索能力，ϵ越大，智能体的行为随机性越大，探索能力越大。

蒙特卡罗强化学习算法伪代码

Algorithm 10: 在线策略蒙特卡洛强化学习算法伪代码

Input: 状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣系数 γ，以及环境 Env，初始化的状态-动作值函数 $Q(s,a)=0$，智能体采样策略为 ϵ-贪心策略

Output: 最优策略 π^*

for $k=0,1,2,3,\cdots$ **do**

  智能体采用 ϵ-贪心策略：

$$\pi(s,a)=\begin{cases} 1-\epsilon+\dfrac{\epsilon}{|\mathcal{A}|}, & a=\arg\max\limits_{a} Q(s,a) \\ \dfrac{\epsilon}{|\mathcal{A}|}, & a\neq \arg\max\limits_{a} Q(s,a) \end{cases} \tag{4.58}$$

  生成完整轨迹样本：$\langle s_0,a_0,r_0,s_1,a_1,r_1,\cdots,s_T,a_T,r_T\rangle$

  for $t=0,1,2,3,\cdots,T$ **do**

    $G_t=\sum_{k=t}^{T}\gamma^{k-t}r_i$

    $Q(s_t,a_t)\leftarrow Q(s_t,a_t)+\alpha(G_t-Q(s_t,a_t))$

% 计算最优策略

for $s\in S$ **do**

  $\pi^*(s)=\arg\max\limits_{a} Q(s,a)$

纲要

时序差分算法

我们简单比较蒙特卡罗强化学习算法和动态规划（Dynamic Programming，DP）算法的差异和实用场景。

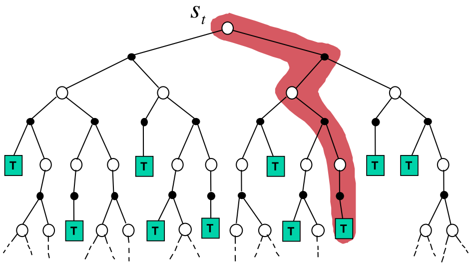


图 1: 图片来自David Silver 个人网站[1]

[1] https://www.davidsilver.uk/teaching/

时序差分算法

我们简单比较蒙特卡罗强化学习算法和动态规划（Dynamic Programming，DP）算法的差异和实用场景。

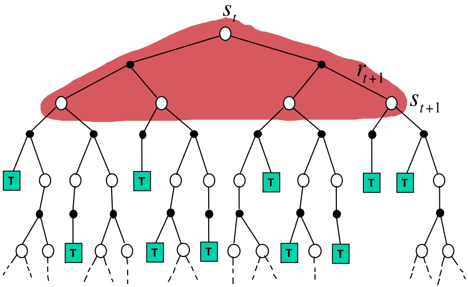


图 2: 图片来自David Silver 个人网站[2]

[2] https://www.davidsilver.uk/teaching/

时序差分算法

时序差分学习（Temporal-Difference Learning，TD Learning）算法是真正意义上的强化学习基础算法，而动态规划和蒙特卡罗算法都是经典的解决马尔科夫决策过程问题的方法。面对动态规划和蒙特卡罗方法的诸多问题，我们将介绍经典的强化学习的基础算法，由强化学习之父 Richard Sutton 教授提出的时序差分算法：

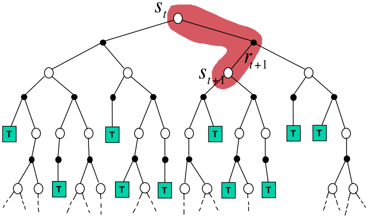


图 3: 图片来自David Silver 个人网站[3]

时序差分算法与动态规划和蒙特卡罗算法比较

我们简单比较时序差分算法、动态规划和蒙特卡罗算法这三个方法的状态值函数估计过程。在动态规划方法中，我们有：

$$V_\pi(S_t) = \mathbb{E}_\pi[R_t + \gamma V_\pi(S_{t+1}) | S_t = s]. \tag{57}$$

在蒙特卡罗方法中，我们有：

$$V_\pi(S_t) = \mathbb{E}_\pi[G_t | S_t = s]. \tag{58}$$

在时序差分方法中，我们有：

$$V_\pi(S_t) \approx [R_t + \gamma V_\pi(S_{t+1}) | S_t = s]. \tag{59}$$

时序差分算法与动态规划和蒙特卡罗算法比较

表 1: 时序差分（TD）、动态规划（DP）和蒙特卡罗（MC算法）的比较

对比项	动态规划	蒙特卡罗	时序差分
是否采样	无须采样	完整轨迹	不完整轨迹
是否自举	自举	不自举	自举
偏差	无偏差	无偏差	预估有偏，真实无偏
方差	无方差	高方差	低方差
是否依赖Markov	是	否	是

Q-learning

强化学习算法Q-learning的状态-动作值函数更新公式为：

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)). \tag{60}$$

其中，r表示智能体在状态s下动作a的即时回报，$\max_{a'} Q(s',a')$表示智能体跳转至下一个状态s'能获得的最大累积回报，此过程需要遍历所有的动作。状态-动作值函数的更新公式$r + \gamma \max_{a'} Q(s',a')$被称作**时序差分（TD）目标值**。式（60）与采用增量值更新状态-动作值函数的蒙特卡罗算法类似：

$$Q_n(s,a) = Q_{n-1}(s,a) + \alpha(G_n - Q_{n-1}(s,a)). \tag{61}$$

一条完整轨迹的累积回报G_n替换成 $r + \gamma \max_{a'} Q(s',a')$。

Q-learning算法伪代码

Algorithm 11: 时序差分 Q-learning 算法伪代码

Input: 状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣系数 γ 以及环境 Env，初始化状态-动作值函数 $Q(s,a)=0$，初始化采样策略为随机策略 $\pi(a|s)=\frac{1}{|\mathcal{A}|}$

Output: 最优策略 π^*

for $k=0,1,2,3,\cdots$ **do**

    % 每次循环针对一条轨迹

    初始化状态 s

    for $t=0,1,2,3,\cdots,T$ **do**

        % 采用 ϵ-贪心策略：

$$\pi(s,a)=\begin{cases}1-\epsilon+\frac{\epsilon}{|\mathcal{A}|}, & a=\arg\max_a Q(s,a)\\ \frac{\epsilon}{|\mathcal{A}|}, & a\neq\arg\max_a Q(s,a)\end{cases}\tag{4.64}$$

        产生一步轨迹 $\langle s,a,r,s'\rangle$，其中，a 和 r 是基于 ϵ-贪心策略产生的动作和即时奖励，s' 是智能体下一个状态。

        更新状态-动作值函数：$Q(s,a)\leftarrow Q(s,a)+\alpha(r+\gamma\max_{a'}Q(s',a')-Q(s,a))$

        智能体进入下一个状态 $s=s'$

        if s 为终止状态 **then**

            开始下一条轨迹采样

% 计算最优策略

for $s\in\mathcal{S}$ **do**

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

SARSA算法

Q-learning算法是强化学习中最负盛名的算法之一，但Q-learning算法也存在很多问题和不足，如过估计（Overestimation）问题。时序差分SARSA算法是另一个经典强化学习算法。SARSA算法与Q-learning算法的主要区别是状态-动作值函数更新公式：

$$\begin{aligned} \text{SARSA}: \quad & Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma Q(s',a') - Q(s,a)\right), \\ \text{Q-learning}: \quad & Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right). \end{aligned} \tag{62}$$

SARSA算法伪代码

Algorithm 12: 时序差分 SARSA 算法伪代码

Input: 状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣系数 γ 以及环境 Env，初始化状态-动作值函数 $Q(s,a)=0$，初始化智能体采样策略 $\pi(a|s)=\dfrac{1}{|\mathcal{A}|}$

Output: 最优策略 π^*

for $k=0,1,2,3,\cdots$ **do**

  % 每次循环针对一条轨迹

  初始化状态 s

  for $t=0,1,2,3,\cdots,T$ **do**

    % 采用 ϵ-贪心策略生成轨迹中的一步：

$$\pi(s,a)=\begin{cases}1-\epsilon+\dfrac{\epsilon}{|\mathcal{A}|}, & a=\arg\max\limits_{a} Q(s,a)\\ \dfrac{\epsilon}{|\mathcal{A}|}, & a\neq\arg\max\limits_{a} Q(s,a)\end{cases} \tag{4.68}$$

    产生轨迹 $\langle s,a,r,s',a'\rangle$，其中，$a$ 是基于 ϵ-贪心策略产生的动作，r 是智能体获得的即时奖励，s' 是智能体下一个状态，a' 是智能体在状态 s' 下基于 ϵ-贪心策略产生的下一个动作。

    更新动作值函数：

$$Q(s,a)\leftarrow Q(s,a)+\alpha(r+\gamma Q(s',a')-Q(s,a)) \tag{4.69}$$

    智能体进入下一个状态 $s=s'$ 以及 $a=a'$

    if s 为终止状态 **then**

      开始下一条轨迹采样

% 计算最优策略

for $s\in\mathcal{S}$ **do**

  $\pi^*(s)=\arg\max\limits_{a} Q(s,a)$

纲要

策略梯度方法

策略梯度方法同样通过智能体与环境交互，获得轨迹数据并学习和更新策略函数。轨迹τ可以表示为：

$$\tau = \{s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_T, a_T, r_T\}. \tag{63}$$

基于轨迹τ，可计算智能体的累积回报：

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t, \tag{64}$$

轨迹τ发生的概率：

$$\begin{aligned} p_\theta(\tau) &= p(s_0)\pi_\theta(a_0|s_0)p(s_1|s_0, a_0)\pi_\theta(a_1|s_1)p(s_2|s_1, a_1)\cdots \\ &= p(s_0)\prod_{t=0}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t), \end{aligned} \tag{65}$$

策略梯度方法

智能体与环境交互得到的所有轨迹τ的期望收益为：

$$\bar{R}_\theta = \sum_\tau R(\tau) p_\theta(\tau). \tag{66}$$

基于策略梯度的强化学习的目标是找到最优化的策略函数参数θ，即最优策略π_θ，使得期望累积收益$\bar{R}_\theta$ 最大，用数学语言描述为：

$$\pi_\theta = \arg\max_{\pi_\theta} \bar{R}_\theta = \arg\max_{\pi_\theta} \sum_\tau R(\tau) p_\theta(\tau). \tag{67}$$

显然，$\bar{R}_\theta = \sum_\tau R(\tau) p_\theta(\tau)$是策略梯度方法的目标函数，可采用梯度上升法求解最优参数θ。

策略梯度方法

在分析策略梯度算法的过程中，需要用到对数函数求导公式：

$$\nabla \log f(x) = \frac{\nabla f(x)}{f(x)}. \tag{68}$$

此公式在策略函数推导过程中具有重要的作用，可以改写成：

$$\nabla f(x) = f(x) \nabla \log f(x). \tag{69}$$

将函数$f(x)$替换成轨迹τ发生的概率$p_\theta(\tau)$，可得：

$$\nabla p_\theta(\tau) = p_\theta(\tau) \nabla \log p_\theta(\tau), \tag{70}$$

即

$$\frac{\nabla p_\theta(\tau)}{p_\theta(\tau)} = \nabla \log p_\theta(\tau). \tag{71}$$

策略梯度方法

针对优化问题$\pi_\theta = \arg\max_{\pi_\theta} \sum_\tau R(\tau)p_\theta(\tau)$，我们采用策略梯度方法直接对目标函数$\bar{R}_\theta = \sum_\tau R(\tau)p_\theta(\tau)$求梯度，可得：

$$
\begin{aligned}
\nabla \bar{R}_\theta &= \sum_\tau R(\tau)\nabla p_\theta(\tau) \\
&= \sum_\tau R(\tau)p_\theta(\tau)\frac{\nabla p_\theta(\tau)}{p_\theta(\tau)} \\
&= \sum_\tau R(\tau)p_\theta(\tau)\nabla \log p_\theta(\tau) \\
&= \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[R(\tau)\nabla \log p_\theta(\tau)\right].
\end{aligned} \tag{72}
$$

策略梯度方法

公式$\nabla \bar{R}_\theta = \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[R(\tau)\nabla \log p_\theta(\tau)\right]$为策略梯度方法中更新策略函数参数的核心公式，如此转化公式的作用是将目标函数梯度的计算问题，转化成通过蒙特卡罗采样完成梯度估计：

$$\nabla \bar{R}_\theta = \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[R(\tau)\nabla \log p_\theta(\tau)\right] \approx \frac{1}{n}\sum_{i=1}^{n} R\left(\tau_i\right)\nabla \log p_\theta\left(\tau_i\right). \tag{73}$$

因此，策略梯度方法的策略函数参数更新公式可写作：

$$\theta \leftarrow \theta + \alpha \nabla \bar{R}_\theta, \tag{74}$$

其中α为学习率，决定了策略函数参数更新的步长大小。

策略梯度方法

策略梯度方法的核心公式需要计算 $\nabla \log p_\theta(\tau)$，对式(65)求梯度可得：

$$
\begin{aligned}
\nabla \log p_\theta(\tau) &= \nabla \left(\log p(s_0) + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \sum_{t=1}^{T} \log p(s_{t+1}|s_t, a_t) \right) \\
&= \nabla \log p(s_0) + \nabla \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \nabla \sum_{t=1}^{T} \log p(s_{t+1}|s_t, a_t) \\
&= \nabla \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) \\
&= \sum_{t=1}^{T} \nabla \log \pi_\theta(a_t|s_t)
\end{aligned} \tag{75}
$$

策略梯度方法

在公式推导过程中有两个关键之处，分别为：

$$\nabla \log p(s_0) = 0 \tag{76}$$

以及

$$\nabla \sum_{t=1}^{T} \log p(s_{t+1}|s_t, a_t) = 0. \tag{77}$$

状态稳定分布函数$p(s_0)$和环境状态转移函数 $p(s_{t+1}|s_t, a_t)$由环境模型决定，不会因为策略函数的变化而变化，因而与策略函数参数θ无关，其梯度均为0。在公式推导中，只有策略函数$\pi_\theta(a_t|s_t)$包含了参数θ。

策略梯度方法

将式（75）代入目标函数梯度公式（73），可得：

$$\begin{aligned}
\nabla \bar{R}_\theta &= \sum_\tau R(\tau) \nabla p_\theta(\tau) \\
&= \mathbb{E}_{\tau \sim p_\theta(\tau)} [R(\tau) \nabla \log p_\theta(\tau)] \\
&\approx \frac{1}{n} \sum_{i=1}^{n} R\left(\tau^i\right) \nabla \log p_\theta\left(\tau^i\right) \\
&= \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{T_i} R\left(\tau^i\right) \nabla \log \pi_\theta\left(a_t^i | s_t^i\right),
\end{aligned} \tag{78}$$

其中，n为随机采样的完整轨迹数量，变量上标i为轨迹编号，T_i为第i条轨迹的长度。

蒙特卡罗策略梯度算法伪代码

Algorithm 13: 蒙特卡洛策略梯度（REINFORCE）算法伪代码

Input: 状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣系数 γ 以及环境 Env，可微分策略函数 $\pi_{\boldsymbol{\theta}}(a|s)$，学习率 α

Output: 最优策略 π^*

初始化策略函数的参数 $\boldsymbol{\theta}$

for $n = 0, 1, 2, 3, \cdots$ **do**

  % 每次循环针对一条轨迹

  初始化状态 s_0，生成一条轨迹

$$\tau = \{s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_T, a_T, r_T\} \tag{4.92}$$

  for $t = 0, 1, 2, 3, \cdots, T$ **do**

    计算当前时间步开始到轨迹结束的累积回报 G_t

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha G_t \nabla \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \tag{4.93}$$

    if s 为终止状态 **then**

      开始下一条轨迹采样

  if $\boldsymbol{\theta}$ 收敛 **then**

    停止迭代

纲要

强化学习智能交易系统框架

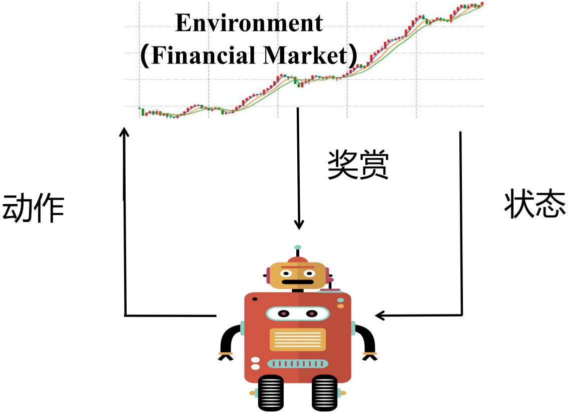


图 4: 基于强化学习的智能交易系统框架示意图

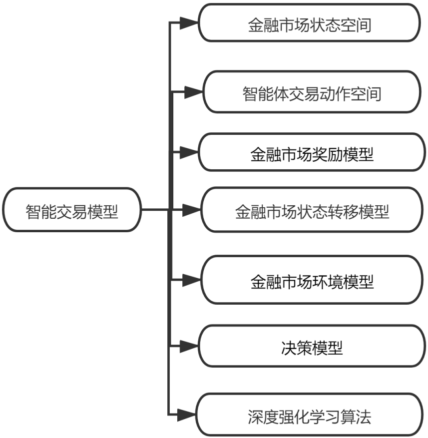
智能交易环境模型编程

智能交易模型一直是量化金融的热门方向。相较于游戏环境的强化学习智能体建模，金融市场智能体建模更具挑战性。金融市场环境的复杂度远远高于一般的游戏环境系统。
基于离散马尔科夫决策过程（Markov Decision Process，MDP)的金融市场环境模型可以表示为六元组 $(\mathcal{S}, \mathcal{A}, P, R, \gamma, H)$，其中:

- $\mathcal{S}$ 表示金融市场环境状态集合，
- $\mathcal{A}$ 表示智能体动作集合，
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$是金融市场环境状态转移函数，
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$是金融市场环境回报函数， $\mathcal{R}$ 为连续的区间， $R(s_t, a_t, s_{t+1}) \in \mathcal{R}$， $R_{\max} \in \mathbb{R}^+$ (e.g., $[0, R_{\max}]$)，
- $\gamma \in [0, 1)$ 是折扣系数，
- H 是投资期限。

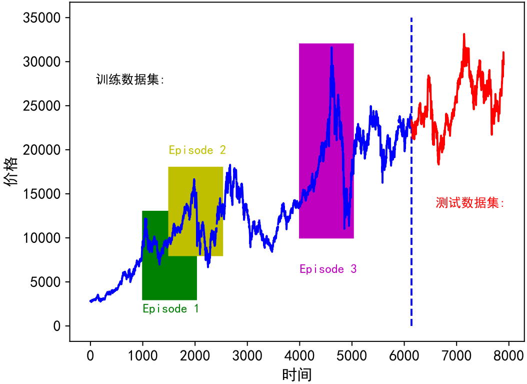
智能交易环境模型编程

在金融市场环境中，基于强化学习的智能交易系统模块：



金融市场状态空间

金融市场数据形式千变万化，如价格时间序列、交易网络、非结构化的新颖数据等，都能为个体和机构的投资决策提供市场环境信息和市场状态信息，可以作为投资智能体的决策变量。



智能体动作空间

在金融市场环境模型六元组中，$\mathcal{A}$表示智能体的动作空间，即智能体金融交易动作。智能体基于金融市场环境变量和当前策略函数给出交易动作。一般来说，智能体动作可以分成两类，一类是连续型，一类是离散型。如交易智能体的离散型动作$a \in \mathcal{A}$可表示为:

$$a = \begin{cases} 1, & \text{buy} \\ 0, & \text{hold} \\ -1, & \text{sell} \end{cases}, \tag{79}$$

离散型数值1、0和-1 分别表示买入、持有和卖出金融资产。

金融市场奖励模型

奖励函数 $R(s,a,s')$可以定义为投资组合总市值的变化量：

$$R(s,a,s') = v' - v, \tag{80}$$

其中，奖励函数R表示智能体在金融市场状态s下执行动作a并转化到下一个状态 s'后获得的即时奖励值，v' 和v 分别表示智能体在状态 s' 和s时资产市值。奖励函数 $R(s,a,s')$也可以定义为智能体动作前后投资组合市值的对数收益：

$$R(s,a,s') = \log(v') - \log(v). \tag{81}$$

在金融理论界和金融实务界，对数收益率更为常用。

金融市场状态转移模型

- 在金融市场环境模型六元组中，$P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ 表示金融市场环境的转移函数。
- 在金融市场环境模型中，状态转移的时间粒度可以重新设计。t时刻的金融市场状态可以转移到下一个状态，即$t + \Delta t$时刻的状态，Δt可以是1天、3天、5天、1周等，粒度可以更粗或更细。Δt决定了智能体进行决策的间隔时间或交易频率，也是智能体调仓的最短间隔时间。我们可以设置较大的Δt值，以减少金融交易智能体调仓的次数。
- 在金融市场环境模型六元组中，$\gamma \in [0, 1)$ 是折扣因子，与金融中折现因子具有类似的含义。
- 交易成本可设置为固定费用，即每笔交易扣除固定金额。一般而言，金融市场中按照百分比扣除交易费用，例如$1/1000$是每笔交易最常用的交易成本率。

决策模型

智能体的策略函数的功能是处理和分析金融市场环境信息，建模策略函数的主要工具是深度神经网络模型，如前馈神经网络、卷积神经网络、循环神经网络、图神经网络等。

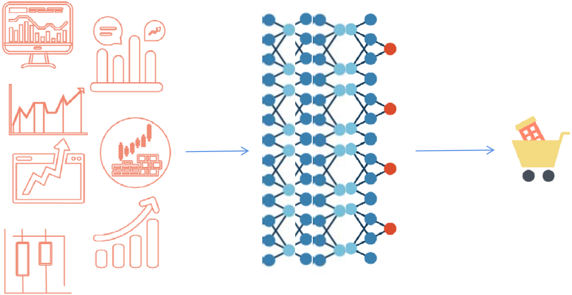


图 7: 智能体投资决策过程示意图

决策模型和强化学习算法

深度强化学习算法在强化学习理论和算法基础上融合了深度神经网络模型（DNN、CNN、RNN等）。

- Deep Q Network（DQN）
- 置信阈策略优化（Trust Region Policy Optimization，TRPO）
- 近端策略优化（Proximal Policy Optimization，PPO）
- 深度确定性策略梯度方法（Deep Deterministic Policy Gradient, DDPG）
- Twin Delayed DDPG （TD3）
- Actor-Critic算法等。

掌握的问题

1. 什么是马尔科夫过程？
2. 什么是马尔科夫回报过程？
3. 什么是马尔科夫决策过程？
4. 强化学习与动态规划的区别有哪些？
5. 强化学习与蒙特卡罗方法的区别有哪些？
6. Q-learning和SARSA的区别？
7. 基于值函数的强化学习算法有哪些优点？
8. 基于值函数的和基于策略的强化学习的区别？