

# 强化学习

# Reinforcement Learning

实验十一

2020/12/4

冯禹豪

## 有监督学习的一些问题

在之前的学习中，我们主要关注有监督学习，而有监督学习一般需要一定数量的带标签的数据。在很多的应用场景中，通过人工标注的方式来给数据打标签的方式往往行不通。比如我们通过有监督学习来训练一个模型可以自动下围棋，就需要将当前棋盘的状态作为输入数据，其对应的最佳落子位置（动作）作为标签。训练一个好的模型就需要收集大量的不同棋盘状态以及对应动作。这种做法实践起来比较困难，一是对于每一种棋盘状态，即使是专家也很难给出“正确”的动作，二是获取大量数据的成本往往比较高。对于下棋这类任务，虽然我们很难知道每一步的“正确”动作，但是其最后的结果（即赢输）却很容易判断。

# 什么是强化学习

强化学习（Reinforcement Learning, RL），也叫增强学习，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）。而强化学习的关键问题是在于每一个动作并不能直接得到监督信息，需要通过整个模型的最终监督信息（奖励）得到，并且有一定的延时性。所以我们要解决如何通过直接得到的监督信息，来获得每个状态中比较恰当的动作的问题。

# 强化学习定义

在强化学习中，有两个可以进行交互的对象：智能体和环境。智能体（Agent）可以感知外界环境的状态（State）和反馈的奖励（Reward），并进行学习和决策。智能体的决策功能是指根据外界环境的状态来做出不同的动作（Action），而学习功能是指根据外界环境的奖励来调整策略。环境（Environment）是智能体外部的所有事物，并受智能体动作的影响而改变其状态，并反馈给智能体相应的奖励。

强化学习的要素：

- 状态 $s$  是对环境的描述，可以是离散的或连续的，其状态空间为 $\mathcal{S}$ 。
- 动作 $a$  是对智能体行为的描述，可以是离散的或连续的，其动作空间为 $\mathcal{A}$ 。
- 策略 $\pi(a|s)$  是智能体根据环境状态 $s$  来决定下一步动作 $a$  的函数。
- 状态转移概率 $p(s'|s, a)$  是在智能体根据当前状态 $s$  做出一个动作 $a$  之后，环境在下一个时刻转变为状态 $s'$  的概率。
- 即时奖励 $r(s, a, s')$  是一个标量函数，即智能体根据当前状态 $s$  做出动作 $a$  之后，环境会反馈给智能体一个奖励，这个奖励也经常和下一个时刻的状态 $s'$  有关。

# 策略

智能体的策略（Policy）就是智能体如何根据环境状态 $s$ 来决定下一步的动作 $a$ ，通常可以分为确定性策略（Deterministic Policy）和随机性策略（Stochastic Policy）两种。确定性策略是从状态空间到动作空间的映射函数 $\pi : \mathcal{S} \rightarrow \mathcal{A}$ 。随机性策略表示在给定环境状态时，智能体选择某个动作的概率分布：

$$\pi(a|s) \triangleq p(a|s),$$
$$\sum_{a \in \mathcal{A}} \pi(a|s) = 1.$$

通常情况下，强化学习一般使用随机性策略。随机性策略可以有很多优点：1) 在学习时可以通过引入一定随机性更好地探索环境；2) 随机性策略的动作具有多样性，这一点在多个智能体博弈时也非常重要。采用确定性策略的智能体总是对同样的环境做出相同的动作，会导致它的策略很容易被对手预测。

# 马尔可夫决策过程

假设交互是一个离散序列： $s_0, a_0, s_1, r_1, a_1, \dots, s_{(t-1)}, r_{(t-1)}, a_{(t-1)}, s_t, r_t, \dots$ ,

马尔可夫过程是指下一个时刻的状态只取决于当前状态：

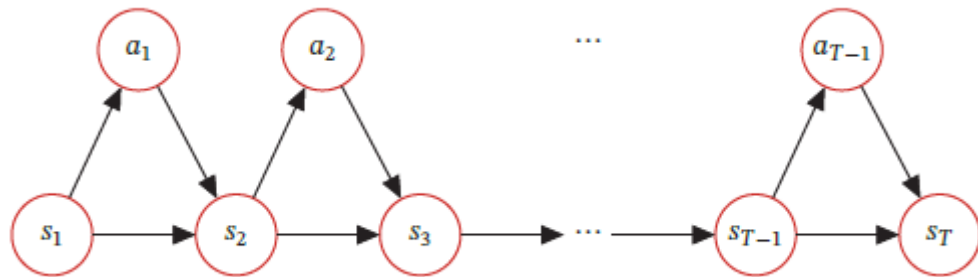
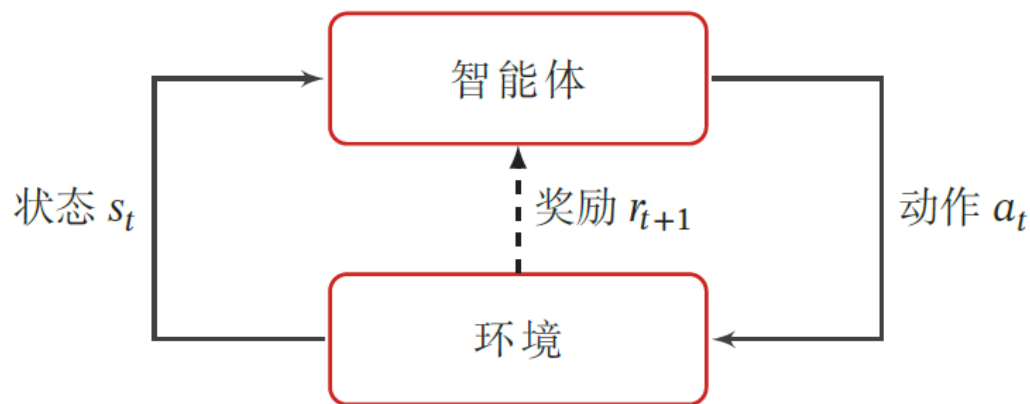
$$p(s_{t+1}|s_t, \dots, s_0) = p(s_{t+1}|s_t),$$

而马尔可夫决策过程是在马尔可夫过程中加入一个额外的变量——动作 $a$ ：

$$p(s_{t+1}|s_t, a_t, \dots, s_0, a_0) = p(s_{t+1}|s_t, a_t),$$

给定策略 $\pi(a|s)$ 和动作 $a$ ，马尔可夫决策过程的一个轨迹（Trajectory）的概率为：

$$\begin{aligned} \tau &= s_0, a_0, s_1, r_1, a_1, \dots, s_{T-1}, a_{T-1}, s_T, r_T \\ p(\tau) &= p(s_0, a_0, s_1, a_1, \dots) \\ &= p(s_0) \prod_{t=0}^{T-1} \pi(a_t|s_t) p(s_{t+1}|s_t, a_t). \end{aligned}$$



## 强化学习的目标函数

给定策略 $\pi(a|s)$ ，智能体和环境一次交互过程的轨迹 $\tau$  所收到的累积奖励为总回报（Return）：

$$G(\tau) = \sum_{t=0}^{T-1} r_{t+1} = \sum_{t=0}^{T-1} r(s_t, a_t, s_{t+1}).$$

引入一个折扣率来衡量长短期回报的权重。当 $\gamma$  接近于0 时，智能体更在意短期回报；而当 $\gamma$  接近于1 时，长期回报变得更重要：

$$G(\tau) = \sum_{t=0}^{T-1} \gamma^t r_{t+1}$$

强化学习为了学习到一个策略（ $\theta$ 是参数）来最大化期望回报。因此目标函数可以定义为：

$$\mathcal{J}(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)}[G(\tau)] = \mathbb{E}_{\tau \sim p_{\theta}(\tau)}\left[\sum_{t=0}^{T-1} \gamma^t r_{t+1}\right]$$

然而这样的目标函数只能针对一个完整序列来求期望回报，那么在某一个状态下，我们并不能得知未来结果的序列是什么，所以也就无法指导下一个动作的产生。因此我们需要引入值函数来评估策略 $\pi$  在某个状态和动作下的期望回报，寻求当前状态下的期望总回报（状态值函数）和当前状态下执行某一个动作的期望总回报（状态-动作值函数）。



## 状态值函数

策略 $\pi$  的期望回报可以分解为：

$$\mathbb{E}_{\tau \sim p(\tau)}[G(\tau)] = \mathbb{E}_{s \sim p(s_0)} \left[ \mathbb{E}_{\tau \sim p(\tau)} \left[ \sum_{t=0}^{T-1} \gamma^t r_{t+1} | \tau_{s_0} = s \right] \right] = \mathbb{E}_{s \sim p(s_0)} [V^\pi(s)],$$

其中  $V^\pi(s)$  称为状态值函数（State Value Function），表示从状态 $s$  开始，执行策略 $\pi$  得到的期望总回报。为了方便起见，我们用  $\tau_{0:T}$  来表示轨迹  $s_0, a_0, s_1, \dots, s_T$ ，用  $\tau_{1:T}$  来表示轨迹  $s_1, a_1, \dots, s_T$ ，因此有  $\tau_{0:T} = s_0, a_0, \tau_{1:T}$ ，那么  $V^\pi(s)$  可以展开得到：

$$V^\pi(s) = \mathbb{E}_{\tau_{0:T} \sim p(\tau)} \left[ r_1 + \gamma \sum_{t=1}^{T-1} \gamma^{t-1} r_{t+1} | \tau_{s_0} = s \right] \quad (14.15)$$

$$= \mathbb{E}_{a \sim \pi(a|s)} \mathbb{E}_{s' \sim p(s'|s,a)} \mathbb{E}_{\tau_{1:T} \sim p(\tau)} \left[ r(s, a, s') + \gamma \sum_{t=1}^{T-1} \gamma^{t-1} r_{t+1} | \tau_{s_1} = s' \right] \quad (14.16)$$

$$= \mathbb{E}_{a \sim \pi(a|s)} \mathbb{E}_{s' \sim p(s'|s,a)} \left[ r(s, a, s') + \gamma \mathbb{E}_{\tau_{1:T} \sim p(\tau)} \left[ \sum_{t=1}^{T-1} \gamma^{t-1} r_{t+1} | \tau_{s_1} = s' \right] \right] \quad (14.17)$$

$$= \mathbb{E}_{a \sim \pi(a|s)} \mathbb{E}_{s' \sim p(s'|s,a)} [r(s, a, s') + \gamma V^\pi(s')]. \quad (14.18)$$

公式(14.18) 也称为贝尔曼方程（Bellman Equation），表示当前状态的值函数可以通过下个状态的值函数来计算。

## 状态-动作值函数

$$\mathbb{E}_{a \sim \pi(a|s)} \mathbb{E}_{s' \sim p(s'|s,a)} [r(s, a, s') + \gamma V^\pi(s')] \quad (14.18)$$

公式(14.18) 中的第二个期望是指初始状态为 $s$  并进行动作 $a$ ，然后执行策略 $\pi$  得到的期望总回报，称为状态-动作值函数（State-Action Value Function）：

$$Q^\pi(s, a) = \mathbb{E}_{s' \sim p(s'|s,a)} [r(s, a, s') + \gamma V^\pi(s')], \quad (14.19)$$

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(a|s)} [Q^\pi(s, a)]. \quad (14.20)$$

$$Q^\pi(s, a) = \mathbb{E}_{s' \sim p(s'|s,a)} \left[ r(s, a, s') + \gamma \mathbb{E}_{a' \sim \pi(a'|s')} [Q^\pi(s', a')] \right], \quad (14.21)$$

值函数可以看作对策略 $\pi$  的评估，因此我们就可以根据值函数来优化策略。

## 基于值函数的学习方法

值函数是对策略 $\pi$  的评估。如果策略 $\pi$  有限（即状态数和动作数都有限），可以对所有的策略进行评估并选出最优策略 $\pi^*$ 。但这种方式在实践中很难实现，通过迭代的方法不断优化策略，直到选出最优策略。

针对如何学习一个最优的策略，我们可以这样做：先随机初始化一个策略，计算该策略的值函数，并根据值函数来设置新的策略，然后一直反复迭代直到收敛。基于值函数的策略学习方法中最关键的是如何计算策略 $\pi$  的值函数，一般有动态规划或蒙特卡罗两种计算方式。

# 动态规划

从贝尔曼方程可知，如果知道马尔可夫决策过程的状态转移概率 $p(s'|s, a)$ 和奖励 $r(s, a, s')$ ，我们直接可以通过贝尔曼方程来迭代计算其值函数。这种模型已知的强化学习算法也称为基于模型的强化学习（Model-Based Reinforcement Learning）算法。在模型已知时，可以通过动态规划的方法来计算。常用的方法主要有策略迭代和值迭代算法。

# 策略迭代

## 算法 14.1 策略迭代算法

输入: MDP 五元组:  $\mathcal{S}, \mathcal{A}, P, r, \gamma$ ;

1 初始化:  $\forall s, \forall a, \pi(a|s) = \frac{1}{|\mathcal{A}|}$ ;

2 **repeat**

    // 策略评估

3     **repeat**

4         根据贝尔曼方程 ( 公式(14.18) ), 计算  $V^\pi(s)$ ,  $\forall s$ ;

5     **until**  $\forall s, V^\pi(s)$  收敛;


    // 策略改进


6     根据公式(14.19), 计算  $Q(s, a)$ ;

7      $\forall s, \pi(s) = \arg \max_a Q(s, a)$ ;

8 **until**  $\forall s, \pi(s)$  收敛;

输出: 策略  $\pi$


$$\mathbb{E}_{a \sim \pi(a|s)} \mathbb{E}_{s' \sim p(s'|s,a)} [r(s, a, s') + \gamma V^\pi(s')]$$


$$Q^\pi(s, a) = \mathbb{E}_{s' \sim p(s'|s,a)} \left[ r(s, a, s') + \gamma \mathbb{E}_{a' \sim \pi(a'|s')} [Q^\pi(s', a')] \right]$$

收敛: 由于存在折扣率, 迭代一定步数后, 每个状态的值函数就会固定不变。

# 值迭代算法

---

## 算法 14.2 值迭代算法

---

输入: MDP 五元组:  $\mathcal{S}, \mathcal{A}, P, r, \gamma$ ;

1 初始化:  $\forall s \in \mathcal{S}, V(s) = 0$ ;

2 **repeat**

3      $\forall s, V(s) \leftarrow \max_a \mathbb{E}_{s' \sim p(s'|s,a)} \left[ r(s, a, s') + \gamma V(s') \right]$ ;

4 **until**  $\forall s, V(s)$  收敛;

5 根据公式(14.19)计算  $Q(s, a)$ ;

6  $\forall s, \pi(s) = \arg \max_a Q(s, a)$ ;

输出: 策略  $\pi$

---

$$V^*(s) = \max_a \mathbb{E}_{s' \sim p(s'|s,a)} \left[ r(s, a, s') + \gamma V^*(s') \right], \quad (14.27)$$

VS

$$\mathbb{E}_{a \sim \pi(a|s)} \mathbb{E}_{s' \sim p(s'|s,a)} \left[ r(s, a, s') + \gamma V^\pi(s') \right]$$

$$Q^\pi(s, a) = \mathbb{E}_{s' \sim p(s'|s,a)} \left[ r(s, a, s') + \gamma \mathbb{E}_{a' \sim \pi(a'|s')} [Q^\pi(s', a')] \right]$$

值迭代算法直接计算最优的  $V(s)$ , 可以对比公式14.27和14.18。14.18是与当前的策略相关的, 而14.27是与策略无关的, 直接通过状态转移概率和奖励函数来计算最优的值函数  $V$ , 然后通过状态-动态值函数来求得最优的策略。

## 动态规划存在的问题

- 1、要求模型已知，即要给出马尔可夫决策过程的状态转移概率 $p(s'|s, a)$ 和奖励函数 $r(s, a, s')$ 。但实际应用中这个要求很难满足。
- 2、效率问题，即当状态数量较多时，算法效率比较低。但在实际应用中，很多问题的状态数量和动作数量非常多。比如，围棋有 $19 \times 19 = 361$  个位置，每个位置有黑子、白子或无子三种状态，整个棋局有 $3^{361} \approx 10^{170}$  种状态。动作（即落子位置）数量为361。一种有效的方法是通过一个函数（比如神经网络）来近似计算值函数，以减少复杂度，并提高泛化能力。

## 蒙特卡洛方法

在很多应用场景中，马尔可夫决策过程的状态转移概率 $p(s'|s, a)$ 和奖励函数 $r(s, a, s')$ 都是未知的。在这种情况下，我们一般需要智能体和环境进行交互，并收集一些样本，然后再根据这些样本来求解马尔可夫决策过程最优策略。这种模型未知，基于采样的学习算法也称为模型无关的强化学习（Model-Free Reinforcement Learning）算法。

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim p(\tau)}[G(\tau_{s_0=s, a_0=a})], \quad (14.29)$$

如果模型未知，Q 函数可以通过采样来进行计算，这就是蒙特卡罗方法。对于一个策略 $\pi$ ，智能体从状态 $s$ ，执行动作 $a$ 开始，然后通过随机游走的方法来探索环境，并计算其得到的总回报。假设我们进行 $N$ 次试验，得到 $N$ 个轨迹，Q 函数可以近似为：

$$Q^\pi(s, a) \approx \hat{Q}^\pi(s, a) = \frac{1}{N} \sum_{n=1}^N G(\tau_{s_0=s, a_0=a}^{(n)}).$$



## $\epsilon$ -贪心法策略

但在蒙特卡罗方法中，如果采用确定性策略 $\pi$ ，每次试验得到的轨迹是一样的，只能计算出 $Q\pi(s, \pi(s))$ ，而无法计算其他动作 $a'$ 的Q函数，因此也无法进一步改进策略。这样情况仅仅是对当前策略的利用（exploitation），而缺失了对环境的探索（exploration），即试验的轨迹应该尽可能覆盖所有的状态和动作，以找到更好的策略。

$$\pi^\epsilon(s) = \begin{cases} \pi(s), & \text{按概率 } 1 - \epsilon, \\ \text{随机选择 } \mathcal{A} \text{ 中的动作,} & \text{按概率 } \epsilon. \end{cases}$$

- 同策略（On-Policy）：在蒙特卡罗方法中，如果采样策略是 $\pi^\epsilon(s)$ ，不断改进策略也是 $\pi^\epsilon(s)$ 而不是目标策略 $\pi(s)$ 。
- 异策略（Off-Policy）：如果采样策略是 $\pi^\epsilon(s)$ ，而优化目标是策略 $\pi(s)$ ，可以通过重要性采样，引入重要性权重来实现对目标策略 $\pi(s)$ 的优化。

# 黑白棋

<http://www.4399.com/flash/66483.htm>

思考一下状态、动作有多少种，即时奖励可以如何设置（例如做了这个动作之后有多少个自己的棋子、这个动作之后并且对方在所有可以下的位置下完之后会有多少自己的棋子，也就是综合了两步得到的奖励值等），可以尝试通过采样得到一些样本，然后思考怎么去构建一个强化学习驱动的黑白棋AI。