

# 强化学习 Reinforcement learning

## 概念

机器处于环境 $E$ 中，状态空间为 $X$ ，其中每个状态 $x \in X$ 是机器感知到的环境的描述，例如：种瓜任务中对西瓜长势的描述。机器能采取的动作构成了动作空间 $A$ 。例如：施肥，浇水，使用农药等。若某个动作 $a \in A$ 作用在当前状态 $x$ 上，则潜在的转移函数 $P$ 将使得环境从当前状态按某种概率转移到另一个状态。例如：瓜苗状态为缺水时候，执行浇水的操作，有一定概率会回复健康也可能不回复。在转移到另一个状态的同时，环境会根据潜在的“奖赏”函数 $R$ 反馈给机器一个奖赏。例如：保持健康奖赏+1，凋零则-10，奖赏可以简单地理解为一个抽象的指标。

综合来看，强化学习对应了四元组 $E = \langle X, A, P, R \rangle$ ，其中 $P: X \times A \times X \mapsto \mathbb{R}$ 指定了状态转移概率， $R: X \times A \times X \mapsto \mathbb{R}$ 指定了奖赏。在某些特殊情况，奖赏函数可能纸盒状态转移相关 $R: X \times X \mapsto \mathbb{R}$

目的：机器要做的是在这种环境下学习出一种策略 $\pi$ ，根据这个策略，在状态 $x$ 下就能得知要执行的动作 $a = \pi(x)$ ，例如：看到瓜苗状态为缺水，就能得知动作“浇水”。

## K-摇臂赌博机

### 探索和利用

K-摇臂赌博机是一种模型，有K个摇臂，赌徒再投入一个硬币后，可以选择一个摇臂，每个摇臂以一定概率吐出硬币，但这个概率赌徒并不知道，赌徒的目标是通过一定的策略最大化奖赏。

探索：估计摇臂优劣 利用：选择当前最优摇臂

仅探索法：将所有的尝试机会平均分给每个摇臂，最后以每个摇臂各自的平均吐币概率作为其奖赏期望的近似估计。目的是为了获取每个要比的期望奖赏

仅利用法：按下目前最优的(即到目前为止平均奖赏最大的)摇臂，若有多个则随机选取其中一个。目的是为了执行奖赏最大的动作

区别:仅探索法能很好估计每个摇臂的奖赏，但是会失去很多选择最优机会的摇臂，仅利用法刚好相反，他很难估计每个摇臂的期望奖赏。

为了能够比较好的对探索和利用进行折中，有两种算法优化， $\epsilon$ -贪心和softmax

### $\epsilon$ -贪心

$\epsilon$ -贪心法基于一个概率来对探索和利用进行这种：每次尝试时，以 $\epsilon$ 的概率进行探索，即以均匀概率随机选取一个摇臂；以 $1 - \epsilon$ 的概率进行利用，即选择当前平均奖赏最高的摇臂。

令 $Q(k)$ 记录摇臂 $k$ 的平均奖赏。若摇臂 $k$ 被尝试了 $n$ 次，得到的奖赏为 $v_1, v_2, \dots, v_n$ ，则平均奖赏为：

$$Q(k) = \frac{1}{n} \sum_{i=1}^n v_i$$

然而上式需要记录多个奖赏值，不够高效，更方便的方式是没尝试一次就更新，用下表来表示尝试的测试，则在经过 $n$ 此尝试后，平均奖赏应该更新为

$$\begin{aligned} Q_n(k) &= \frac{1}{n}((n-1) \times Q_{n-1}(k) + v_n) \\ &= Q_{n-1}(k) + \frac{1}{n}(v_n - Q_{n-1}(k)) \end{aligned}$$

这样只需要记录最近平均奖赏和已经尝试的次数 $n-1$ 。当摇臂奖赏的不确定性比较大时，需要更多探索，则需要较大的 $\epsilon$ 。当不确定较小时，则刚好相反。然而，若尝试次数非常大，则当一段时间后，奖赏可以比较好的近似出来就不需要探索，这个时候可以让 $\epsilon$ 随着次数增加而逐渐减小。

## Softmax

Softmax算法是基于当前已知的摇臂的平均奖赏来对探索和利用进行这种。若当前各摇臂的平均奖赏相当，则选取各摇臂的概率也相当，若某些摇臂的平均奖赏明显高于其他摇臂，则选取概率也更高

Softmax算法中概率分配是基于Boltzmann分布

$$P(k) = \frac{e^{\frac{Q(k)}{\tau}}}{\sum_{i=1}^K e^{\frac{Q(i)}{\tau}}}$$

其中， $Q(i)$ 记录当前摇臂的平均奖赏； $\tau > 0$ 称为“温度”， $\tau$ 越小则平均奖赏高的摇臂被选取的概率越高。 $\tau$ 趋于0时，Softmax将趋于“仅利用”， $\tau$ 趋于无穷大时softmax则将趋于“仅探索”。

## 有模型学习

在已知模型的环境中，即马尔科夫决策过程中四元组 $E = \langle X, A, P, R \rangle$ 均为已知，学习的过程称为“有模型学习”。为方便讨论，先假设状态空间 $X$ 和动作空间 $A$ 均为有限。

### 策略评估

在模型已知时，对于任意策略 $\pi$ 能估计出该策略带来的期望累计奖赏。令函数 $V^\pi(x)$ 表示从状态 $x$ 出发，使用策略 $\pi$ 所带来的累计奖赏。函数 $Q^\pi(x, a)$ 表示从状态 $x$ 出发，执行动作 $a$ 以后在使用策略 $\pi$ 带来的累计奖赏。 $V(\cdot)$ 表示状态值函数。 $Q(\cdot)$ 表示状态-动作值函数。分别表示指定“状态”上以及指定“状态-动作”上的累计奖赏。

根据 $T$ 步累积奖赏和 $\gamma$ 折扣累积奖赏的定义，有状态值函数：

$$\begin{cases} V_T^\pi(x) = \mathbb{E}_\pi \left[ \frac{1}{T} \sum_{t=1}^T r_t | x_0 = x \right] \\ V_\gamma^\pi(x) = \mathbb{E}_\pi \left[ \sum_{t=0}^{+\infty} \gamma^t r_{t+1} | x_0 = x \right] \end{cases}$$

同理有状态-动作值函数：

$$\begin{cases} Q_T^\pi(x, a) = \mathbb{E}_\pi \left[ \frac{1}{T} \sum_{t=1}^T r_t | x_0 = x, a_0 = a \right] \\ Q_\gamma^\pi(x, a) = \mathbb{E}_\pi \left[ \sum_{t=0}^{+\infty} \gamma^t r_{t+1} | x_0 = x, a_0 = a \right] \end{cases}$$

由于MDP具有马尔科夫性质，即系统下一时刻的状态仅由当前时刻的状态决定，不依赖于以往任何状态。

对于T步累积奖赏有

$$\begin{aligned}
 V_T^\pi(x) &= \mathbb{E}_\pi \left[ \frac{1}{T} \sum_{t=1}^T r_t | x_0 = x \right] \\
 &= \mathbb{E}_\pi \left[ \frac{1}{T} r_1 + \frac{T-1}{T} \frac{1}{T-1} \sum_{t=2}^T r_t | x_0 = x \right] \\
 &= \sum_{a \in A} \pi(x, a) \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} \mathbb{E}_\pi \left[ \frac{1}{T-1} \sum_{t=1}^{T-1} r_t | x_0 = x' \right] \right) \\
 &= \sum_{a \in A} \pi(x, a) \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} V_{T-1}^\pi(x') \right)
 \end{aligned}$$

类似的对于 $\gamma$ 折扣累积奖赏有

$$V_\gamma^\pi(x) = \sum_{a \in A} \pi(x, a) \sum_{x' \in X} P_{x \rightarrow x'}^a (R_{x \rightarrow x'}^a + \gamma V_\gamma^\pi(x'))$$

## 策略改进

对某个策略的累积奖赏评估后，发现并不是最优的，理想的策略应该是最大化累积奖赏。

$$\pi^* = \arg \max_{\pi} \sum_{x \in X} V^\pi(x)$$

因此最优策略所对应的值函数 $V^*$ 称为最优值函数，即

$$\forall x \in X : V^*(x) = V^{\pi^*}(x)$$

注意，当策略空间无约束时，上式才是最优策略对应的值函数。

由于最优值函数的累积奖赏值已达最大，因此对前面的累积奖赏函数改动一下，就可以得到最优值函数。

$$\begin{cases} V_T^*(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} V_{T-1}^*(x') \right) \\ V_\gamma^*(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a \left( R_{x \rightarrow x'}^a + \gamma V_\gamma^*(x') \right) \end{cases}$$

换言之就是

$$V^*(x) = \max_{a \in A} Q^{\pi^*}(x, a)$$

带入状态动作值函数可得到

$$\begin{cases} Q_T^*(x, a) = \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} \max_{a' \in A} Q_{T-1}^*(x', a') \right) \\ Q_\gamma^*(x, a) = \sum_{x' \in X} P_{x \rightarrow x'}^a \left( R_{x \rightarrow x'}^a + \gamma \max_{a' \in A} Q_\gamma^*(x', a') \right) \end{cases}$$

上述等式称为最优Bellman等式，其唯一解是最优值函数。其解释了非最优策略的改进方式：将策略选择的动作改变为当前最优的动作。

$$\begin{aligned}
V^\pi(x) &\leq Q^\pi(x, \pi'(x)) \\
&= \sum_{x' \in X} P_{x \rightarrow x'}^{\pi'(x)} \left( R_{x \rightarrow x'}^{\pi'(x)} + \gamma V^\pi(x') \right) \\
&\leq \sum_{x' \in X} P_{x \rightarrow x'}^{\pi'(x)} \left( R_{x \rightarrow x'}^{\pi'(x)} + \gamma Q^\pi(x', \pi'(x')) \right) \\
&= \dots \\
&= V^{\pi'}(x)
\end{aligned}$$

这个公式我其实没推出来过。。。因此改进后的策略为

$$\pi'(x) = \arg \max_{a \in A} Q^\pi(x, a)$$

## 策略迭代与值迭代

知道如何评估一个策略和改进一个策略后，将两者结合起来就可以求解得到最优解的方法：从一个初始策略出发，先进性策略评估，然后改进策略，评估改进的，再进一步改进一次类推，这称为“策略迭代”

但是策略迭代算法每次改进之后都需要对策略评估，因此比较耗时，因此可以把策略改进视作值函数的改进

$$\begin{cases} V_T(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} V_{T-1}(x') \right) \\ V_\gamma(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a \left( R_{x \rightarrow x'}^a + \gamma V_\gamma(x') \right) \end{cases}$$

强化学习任务可以归结基于动态规划的寻有问题，但是和监督学习不同在于，他并未涉及到泛化能力，只是为每一个状态寻找到最好的动作。

## 免模型学习

与前面提到的有模型学习相反，不提前知道四元组的前提下，不依赖环境建模就是“免模型学习”

## 蒙特卡罗强化学习

从起始状态出发，使用某种策略进行采样，执行该策略 $T$ 步并获得轨迹

$$\langle x_0, a_0, r_1, x_1, a_1, r_2, \dots, x_{T-1}, a_{T-1}, r_T, x_T \rangle$$

然后更新Q，然后根据Q再生成新的策略，再利用新策略更新Q，如此迭代下去。

根据生成策略和原始策略是否一致分为同策略和异策略两种学习方法：

同策略：生成策略与更新策略都为原始策略 $\pi$ 对应的 $\epsilon$ -贪心策略 异策略：生成策略是原始策略的 $\pi$ 对应的贪心策略，更新策略是原始策略

## 时序差分学习

与蒙特卡洛强化学习不同的是，蒙特卡洛是在获得完整的轨迹后，进行策略的更新，而时序差分是走一步更新一步。通过下面这个公式，可以动态的更新状态动作对值函数

$$Q_{t+1}^{\pi}(x, a) = Q_t^{\pi}(x, a) + \frac{1}{t+1}(r_{t+1} - Q_t^{\pi}(x, a))$$

令  $\frac{1}{t+1}$  用系数  $\alpha_{t+1}$  替换，当系数越大时，则越靠后的累积奖赏越重要

以  $\gamma$  折扣累积奖赏为例有：

$$\begin{aligned} Q^{\pi}(x, a) &= \sum_{x' \in X} P_{x \rightarrow x'}^a (R_{x \rightarrow x'}^a + \gamma V^{\pi}(x')) \\ &= \sum_{x' \in X} P_{x \rightarrow x'}^a \left( R_{x \rightarrow x'}^a + \gamma \sum_{a' \in A} \pi(x', a') Q^{\pi}(x', a') \right) \end{aligned}$$

通过增量求和得到

$$Q_{t+1}^{\pi}(x, a) = Q_t^{\pi}(x, a) + \alpha (R_{x \rightarrow x'}^a + \gamma Q_t^{\pi}(x', a') - Q_t^{\pi}(x, a))$$

时序差分也分为同策略和异策略

Sarsa: 同策略，生成策略与更新策略都是  $\epsilon$ -贪心策略

Q-学习: 异策略，生成策略为  $\epsilon$ -贪心策略，更新策略为原始策略  $\pi$

## 值函数近似

在前面的所有强化学习中，状态空间都是离散空间。如果状态空间是连续空间，则比较好的方式是值函数近似。例如，把状态-动作值函数假设为线性函数：

$$Q_{\theta}(x; a) = \theta^T(x; a)$$

$(x; a)$  表示把状态  $x$  与  $a$  连起来组成一个大的向量。

利用梯度下降的方式，更新规则有：

$$\theta = \theta + \alpha (Q(x; a) - Q_{\theta}(x; a)) (x; a)$$

利用时序差分学习，用当前估计值替换真实值：

$$Q(x; a) = R_{x \rightarrow x'}^a + \gamma Q^{\pi}(x'; a')$$

## 模仿学习

在现实任务中，可能有人类专家的决策过程范例。从范例中学习称为模仿学习。

### 直接模仿学习

假设有一批人类专家的决策轨迹数据  $\{\tau_1, \tau_2, \dots, \tau_m\}$ ，每条轨迹包含状态和动作序列

$$\tau_i = \langle s_1^i, a_1^i, s_2^i, a_2^i, \dots, s_{n_i+1}^i \rangle$$

其中 $n_i$ 为第 $i$ 条轨迹中的转移次数。

将轨迹中所有的“状态-动作对”抽取出来，构造出一个新的数据集

$$D = \left\{ (s_1, a_1), (s_2, a_2), \dots, \left( s_{\sum_{i=1}^m n_i}, a_{\sum_{i=1}^m n_i} \right) \right\}$$

在这个数据集上采用分类或回归算法学得策略模型，可作为进行强化学习的初始策略，然后通过以上方法学得更好的策。

## 逆强化学习

从人类专家提供的范例数据中反推出奖赏函数，就是逆强化函数。

假设奖赏函数能表达为状态特征的线性函数 $R(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$  于是策略的累积奖赏可以写为：

$$\begin{aligned} \rho^\pi &= \mathbb{E} \left[ \sum_{t=0}^{+\infty} \gamma^t R(x_t) | \pi \right] \\ &= \mathbb{E} \left[ \sum_{t=0}^{+\infty} \gamma^t \mathbf{w}^T \mathbf{x}_t | \pi \right] \\ &= \mathbf{w}^T \mathbb{E} \left[ \sum_{t=0}^{+\infty} \gamma^t \mathbf{x}_t | \pi \right] \end{aligned}$$

把状态向量的期望 $\mathbb{E} \left[ \sum_{t=0}^{+\infty} \gamma^t \mathbf{x}_t | \pi \right]$ 简写成 $\bar{\mathbf{x}}^\pi$ ，范例记为 $\bar{\mathbf{x}}^*$ 。那么对于任意策略产生的 $\bar{\mathbf{x}}^\pi$ ，有

$$\mathbf{w}^{*T} \bar{\mathbf{x}}^* - \mathbf{w}^{*T} \bar{\mathbf{x}}^\pi = \mathbf{w}^{*T} (\bar{\mathbf{x}}^* - \bar{\mathbf{x}}^\pi) \geq 0$$

$$\mathbf{w}^* = \operatorname{argmax}_w \min_{\pi} w^T (\bar{\mathbf{x}}^* - \bar{\mathbf{x}}^\pi)$$

这样可以得到一个更好的 $w$ (对应更好的策略)，然后再用新得到的 $w$ 去产生新的轨迹，这样迭代直到收敛。收敛得到的策略就是范例下的最优策略。