强化学习引入

宋奕龙 北京大学 大数据中心

罗涂 北京大学 数学科学学院

目录

- 强化学习概述
- 多臂老虎机问题
- 从单一状态到多状态（马尔可夫过程）
- 贝尔曼方程
- 动态规划: 价值迭代与策略迭代

强化学习概述

强化学习应用广泛

- 围棋高手Alpha Go, Alpha Zero.
- 人类反馈的强化学习 RLHF

强化学习基本设定

- 智能体+环境
- 在状态S_t下，智能体选择执行动作A_t
- 在动作A_t下，环境返回奖励R_t以及新的状态S_{t+1}.

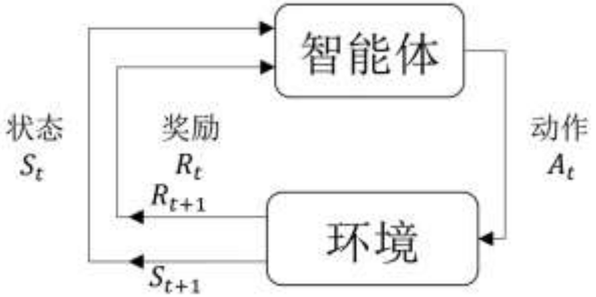
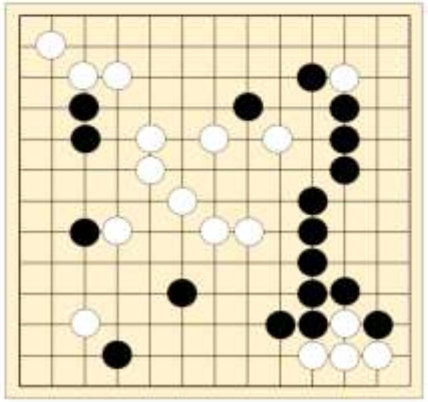
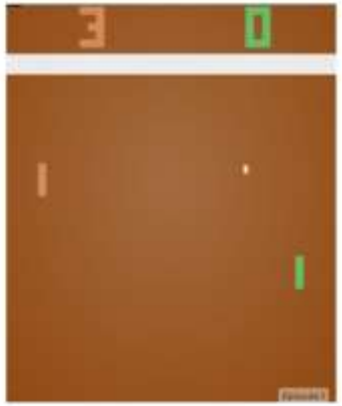


图 2.1 智能体与环境

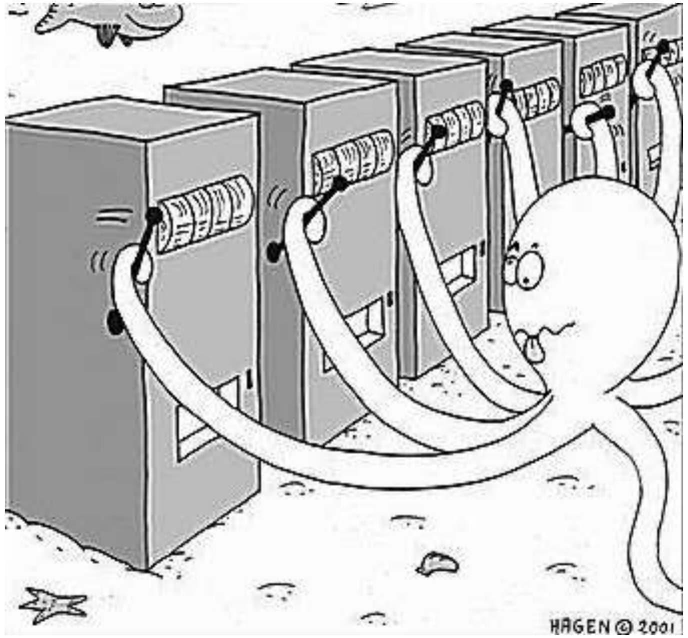




多臂老虎机问题

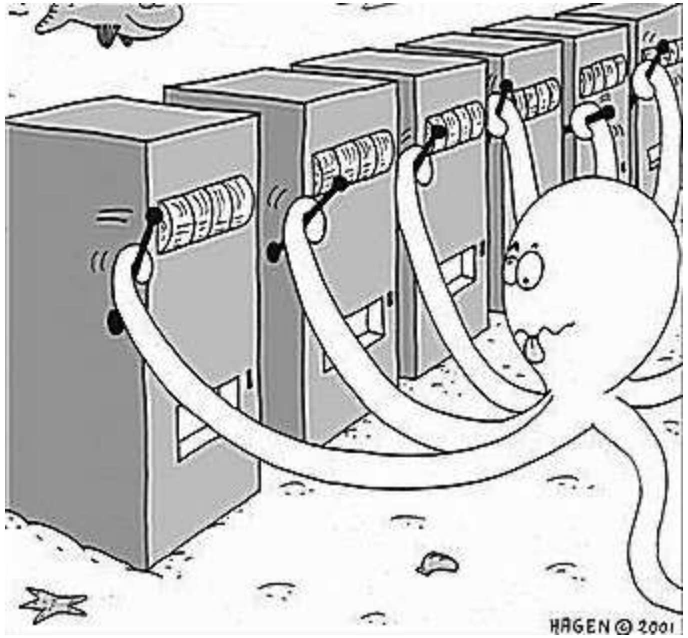
多臂老虎机——强化学习界的果蝇

- 问题设定：假设老虎机有n个臂，每个臂每次给出的奖励值可认为满足正态分布$N\left(\mu_i, \sigma_i^2\right)$.
- 每一轮游戏，玩家可根据先前的经验选择一个臂，同时获得相应奖励。
- 玩家的目标是尽可能在较少的轮次内，识别出奖励值较大的那些臂，并在之后的轮次内更多选择这些臂。



多臂老虎机——强化学习界的果蝇

- 智能体：玩家
- 环境：多臂老虎机
- 状态集：只有一个状态，即在玩状态。
- 动作集：选择臂的编号。
- 奖励值：每次摇臂获得的奖励。



一个简单的想法：

*统计每个臂的每次回报的***平均值***，作为对其期望奖励的估计。同时在每次需要抉择时，选择当前所有臂中***平均奖励最大***的一个臂来进行下一轮游戏。*

根据当前已有信息，始终选择最优的算法通常称为**贪心算法**。

- 贪心算法的弊端在于其十分容易陷入局部最优解中。譬如多臂老虎机问题中，玩家将所有臂尝试一次后，选出某个单次奖励最大的臂，如果该臂确实是一个较优的臂，玩家在很长的时间内可能都会选择该臂，而不再去探索其他潜在可能更好的臂。
- 与贪心算法相对的是**完全随机算法**，即玩家每次完全随机地从每个臂中选择一个臂。
- 实际中的算法通常是介于贪心算法与完全随机算法之间的算法，既考虑了充分**利用**现有的信息，又考虑了去**探索**潜在的可能更好的选择。

利用（exploitation）探索（exploration）

- **利用**与**探索**的平衡是强化学习乃至整个计算机科学的一对重要范畴。
- 例：当食堂上新时，我们倾向于选择**探索**，以期尝试到更美味的食物；而当食堂菜品趋于稳定时，我们倾向于选择**利用**，即经常光顾我们品鉴过的不错的窗口，以求每日尽可能达到最好的就餐体验。

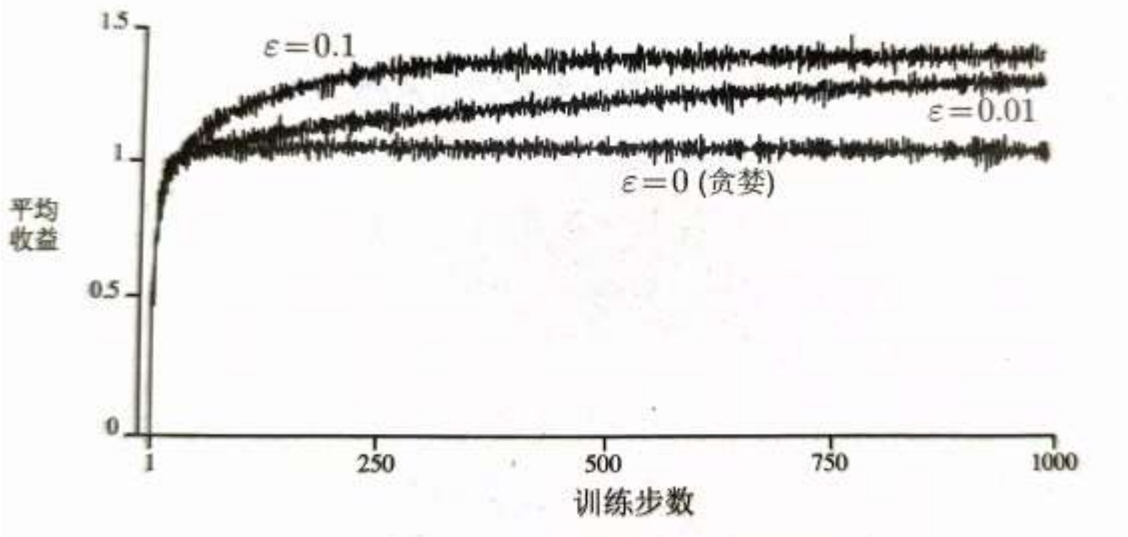
几种经典的动作选择策略

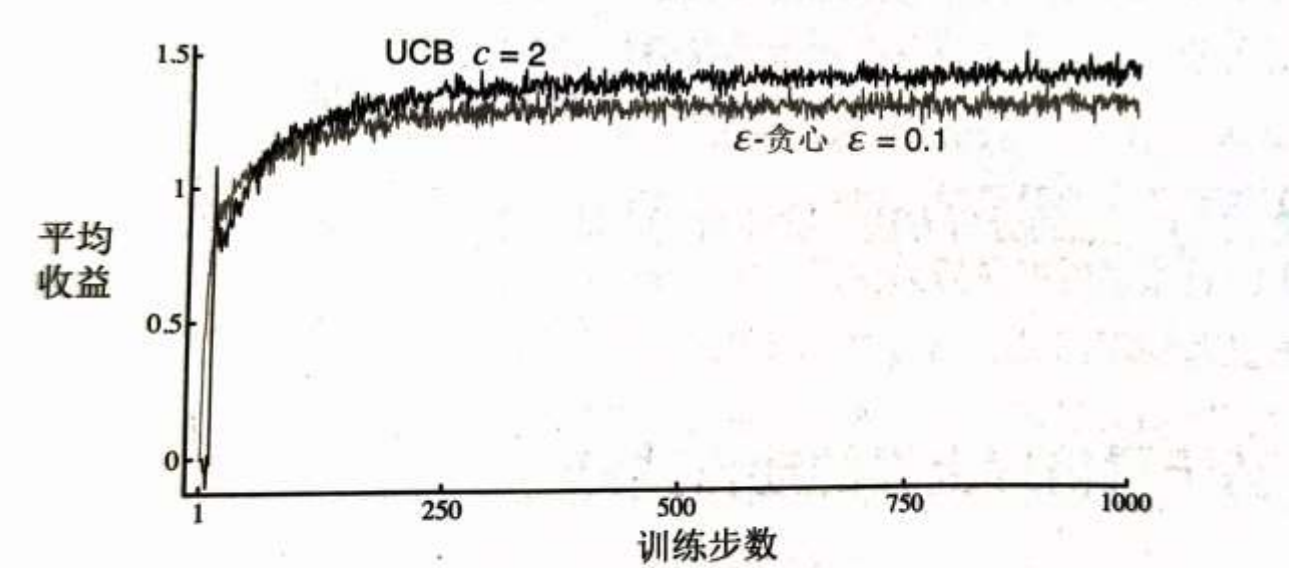
- 贪心策略：每次选择平均奖励最大的臂。
- ε贪心策略：以(1-ε)的概率选择平均奖励最大的臂；以ε的概率随机选择。
- 置信上界（UCB）方法：

$$A_t = \underset{a}{\arg\max}\left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}\right]$$

- 其中$Q_t(a)$即为该臂的平均奖励；c为一个给定的置信度系数，用于平衡探索与开发；$N_t(a)$表示在t时刻之前动作a被选择了几次。

实际效果


1.5
1
0.5
0
ε=0.1
ε=0.01
ε=0 (贪婪)
平均
收益
1
250
500
750
1000
训练步数


1.5
1
0.5
0
UCB c = 2
ε-贪心 ε = 0.1
平均
收益
1
250
500
750
1000
训练步数

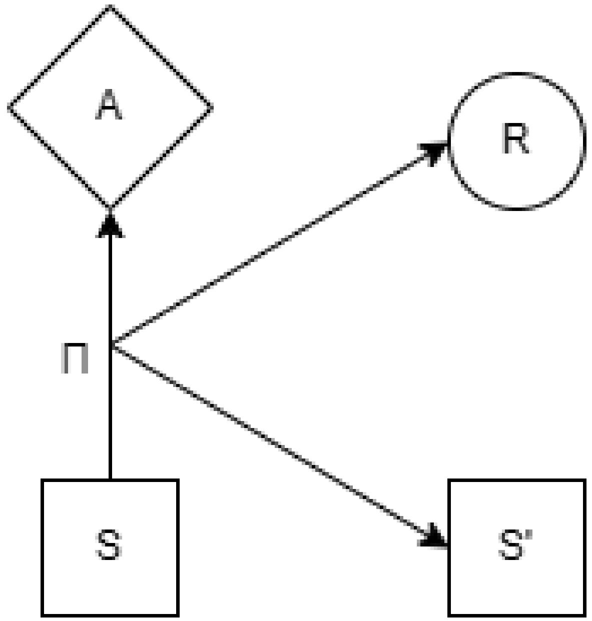
多臂老虎机 vs 机器学习

- 机器学习>多臂老虎机：多臂老虎机**仅有一个状态**，学习的目标是在**有限个**动作中选择一个对于该单一状态最优的动作；而机器学习中，每个学习样本可视为一个状态，同时包含潜在的未纳入学习样本的状态，一般来讲状态空间是不可数的，学习目标是寻找到一个泛化性强的映射，在大量状态下都能取得不错的效果，映射空间一般也是不可数的。
- 多臂老虎机>机器学习：机器学习的损失一般为显式定义，且不具有随机性，给定输入以及模型参数取值后便可求得确定的损失；多臂老虎机在确定动作后，奖励值需要由环境给出，且通常带有随机性。
- 联系：二者均体现学习过程，机器学习表现为学习输入到输出的映射、多臂老虎机表现为学习状态到动作的映射。

马尔可夫过程

从单一状态到有限状态

- 强化学习面对的场景通常不止一种状态。在某种状态下选择一个动作，环境返回奖励值的同时，也会返回一个新的状态。
- 对于静态的机器学习，状态之间**不存在转移关系**，习得的策略只需要满足平均所有状态单次产生的奖励最大即可。
- 对于动态的强化学习，状态之间存在转移关系，因此习得的策略不仅要使得本次奖励值最大，还要确保**下一个状态**足够好。
- <u>状态转移关系的存在与否</u>是二者的第一个区别。



马尔可夫过程（环境侧）

- 环境对智能体的作用表现为如下的条件概率分布：

$$p_t(s', r|s, a),$$

  其含义为：在时间 t 时，状态 s 下，智能体在采取动作 a 后，环境返回奖励值 r 同时将状态转移至 s' 的概率。

- “马尔可夫”的含义即环境的转移概率仅与当前状态 s 以及智能体采取的动作 a 有关，而与更早的状态与动作无关。我们关心的强化学习问题通常都是马尔可夫的。进一步，如果该转移概率不随时间 t 发生变化，我们说该马尔可夫过程是**时齐**的，我们前期考虑的经典强化学习问题通常也都是时齐的。

- **状态与奖励值反馈的随机性**是机器学习与强化学习的第二个区别。

马尔可夫过程的简化版本

- 对于时齐的转移概率

$$p(s', r|s, a),$$

给定状态 s 及动作 a 后，环境返回的奖励值 r 可能是确定的，返回的新状态 s' 也可能是确定的，但它们都能囊括入上述框架中（即概率分布退化为单点分布）。即使二者都是随机的，它们之间也不一定独立，而且多数时候有很强的相关性。

- 当我们只关心某些变量时，上述转移概率可退化为诸如 $p(s'|s, a)$, $p(r|s, a)$, 它们本质上是联合概率分布的边缘分布。特别的，我们有时可能会用到 $p(s'|s)$, 它消除了动作 a 的影响，因此它不仅与环境有关，还与智能体决策 $\pi(a|s)$ 有关。

马尔可夫过程（智能体侧）

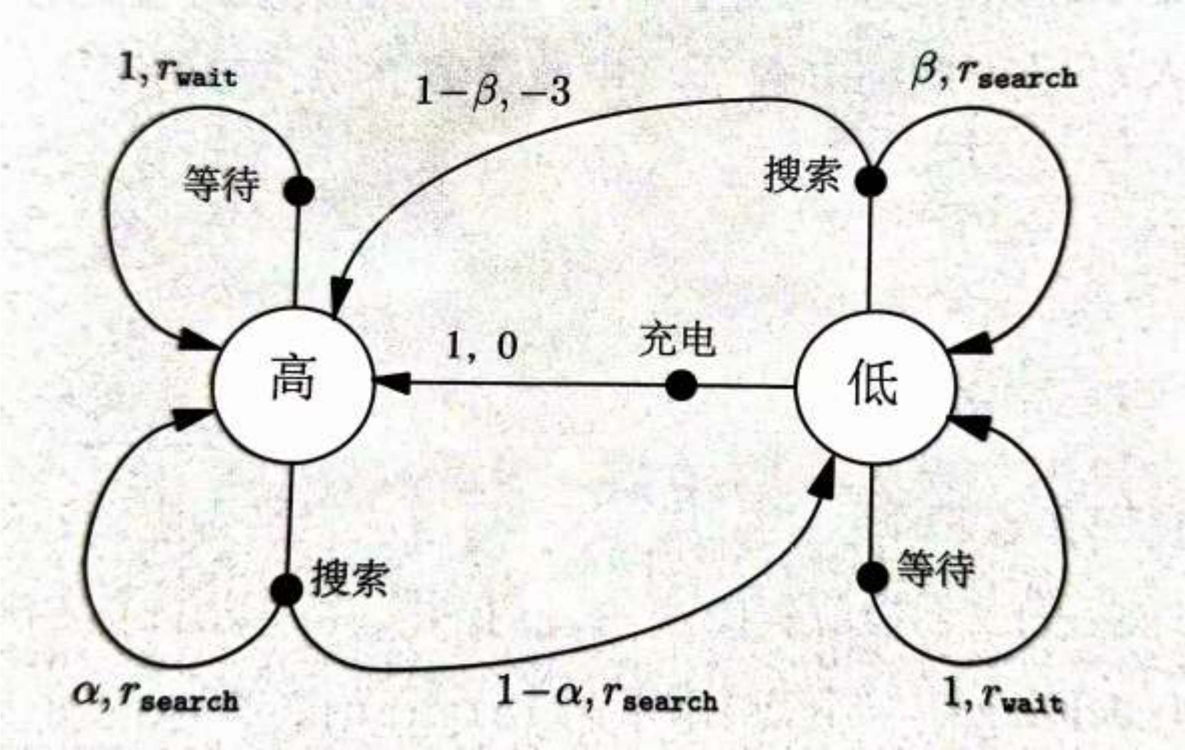
- 先前多臂老虎机的示例表明：智能体的任务是学习从当前状态 s 到最优决策 a 的映射，ϵ 贪心算法同时表明这一决策也有可能是随机的。因此，智能体决策可以表示为如下的条件概率分布：

$$\pi_t(a|s),$$

- 这一表示隐含了马尔可夫性质，即智能体的决策仅与该轮的状态有关，而与更早的若干轮的状态无关。当决策不随时间 t 变化时，我们也称决策为时齐的。
- 因此，看似最符合马尔可夫性质的表达式 $p(s'|s)$ 实则掩盖了智能体与环境的交互关系，反而不利于我们理解。

马尔可夫过程示例

s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	$r_{\mathtt{search}}$
high	search	low	$1-\alpha$	$r_{\mathtt{search}}$
low	search	high	$1-\beta$	-3
low	search	low	β	$r_{\mathtt{search}}$
high	wait	high	1	$r_{\mathtt{wait}}$
high	wait	low	0	$r_{\mathtt{wait}}$
low	wait	high	0	$r_{\mathtt{wait}}$
low	wait	low	1	$r_{\mathtt{wait}}$
low	recharge	high	1	0
low	recharge	low	0	0



贝尔曼方程

累计回报

- 为了将后续状态的转移考虑在内，我们引入累计回报的概念，它是一个**随机变量**，定义为当前状态直至终止状态所有奖励值的和，即：

$$G_t := R_t + R_{t+1} + \cdots + R_T.$$

- 然而，运行至终止状态可能需要很长时间或是无穷轮次，这一定义可能**不收敛**。从实际上讲，智能体也不可能考虑之后所有步的决策，其对越靠后的决策理应关心得越少，因此我们定义累计折扣回报的概念：

$$G_t := R_t + \gamma R_{t+1} + \cdots + \gamma^{T-t} R_T$$

其中 $\gamma \in [0,1)$ 定义为折扣因子，当 $\gamma = 0$ 时即表示智能体只关心当前状态的最优性，即智能体是**短视**的。当每次的奖励值 R_t 均有界时，累计折扣回报可以保证收敛。

累计回报

- 累计折扣回报的性质为其具有递推形式：

$$G_t = R_t + \gamma G_{t+1},$$

且这一递推公式与时间变量 t 无关，这也是我们之后动态规划方法的基础。

价值函数

- 类比物理中势能的概念，我们在强化学习中定义（状态）价值函数，它可理解为在**给定策略** π 下，状态 S_t 后潜在能够给出的奖励的量。具体可以定义为累计折扣回报的期望，即：

$$v_\pi(s) := \mathbb{E}[G_t|S_t = s] = \mathbb{E}\Big[\sum_{k=0}^{\infty}\gamma^k R_{t+k}\Big|S_t = s\Big].$$

- 相应地也可以定义动作价值函数，它额外考虑了每个状态下所采取的每个动作的价值：

$$q_\pi(s,a) := \mathbb{E}[G_t|S_t = s, A_t = a] = \mathbb{E}\Big[\sum_{k=1}^{\infty}\gamma^k R_{t+k}\Big|S_t = s, A_t = a\Big]$$

- 状态价值函数更为简便，通常适用于状态转移完全被动作所决定的确定性情形（如迷宫问题）；动作价值函数考虑更精细，更适合动作执行后仍带有随机性的情形。

贝尔曼方程

- 利用累计折扣回报的递推公式及条件期望，我们可以得到关于状态价值函数的递推公式：

$$
\begin{aligned}
v_\pi(s) =& \mathbb{E}[G_t|S_t = s] \\
=& \mathbb{E}[R_t + \gamma G_{t+1}|S_t = s] \\
=& \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r + \gamma \mathbb{E}[G_{t+1}|S_{t+1} = s']\right] \\
=& \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]
\end{aligned}
$$

  这便是**贝尔曼方程**。

- 类似地我们也可以得到关于动作价值函数的递推公式。

贝尔曼方程显式求解

- 当智能体的策略 $\pi(a|s)$ 确定，环境的转移概率 $p(s',r|s,a)$ 也已知时，贝尔曼方程

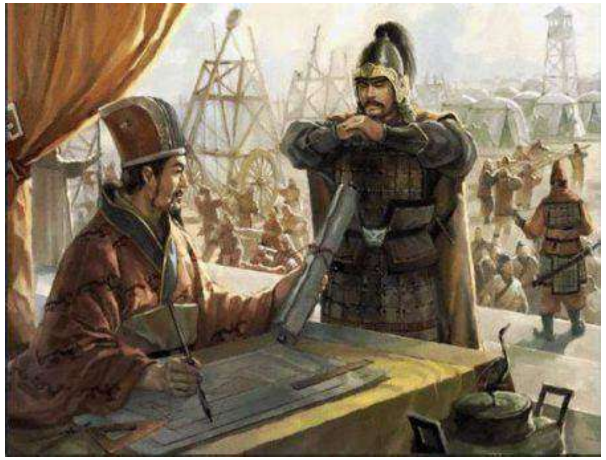
$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')], \quad \forall s \in \mathbb{S}$$

  为一个线性方程，可直接求得解析解。

- 而事实上，多数强化学习问题的环境为一个黑厢模型，即 $p(s',r|s,a)$ 可被采样但是其分布未知。多臂老虎机问题即是如此。

基于模型与去模型

- 根据环境的 $p(s', r|s, a)$ 已知与否，我们可区分出强化学习中**基于模型**与**去模型**两类场景。
- 基于模型场景下，智能体可以“运筹帷幄”，无需与真实环境交互，借助计算机便可求得最优策略。常用方法即动态规划。
- 去模型场景下，智能体需要“摸着石头过河”，在与真实环境进行交互的过程中形成对环境的认识。常用方法为蒙特卡洛方法、时序差分方法等。



价值迭代与策略迭代

最优价值函数

- 最优价值函数指在给定状态 s 时，价值函数在所有可能的策略下的最优取值，即：

  $$v_*(s) := \max_{\pi} v_\pi(s), \quad \forall s \in \mathbb{S}.$$

  相应地也可以定义最优的动作价值函数。

- 强化学习的目标即为找出使得价值函数达到最优时的策略 $\pi_*(a|s)$.

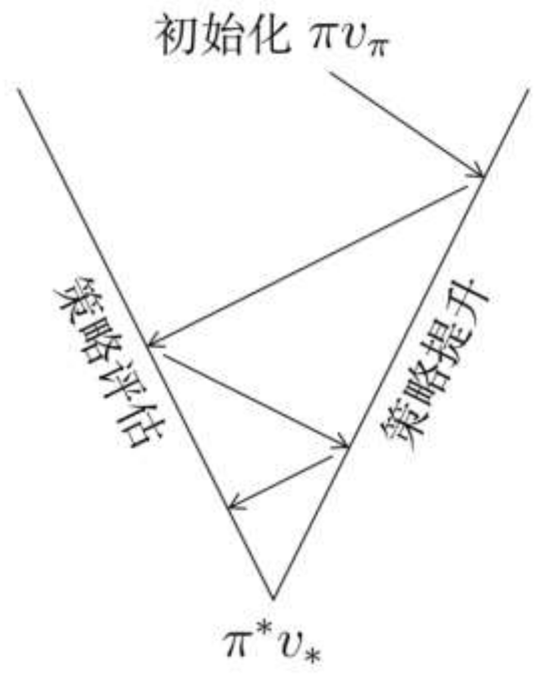
价值迭代与策略迭代

- 价值迭代是指当**策略给定**时，对价值函数进行迭代更新，使其更接近理论值。
- 价值迭代一般有基于模型的动态规划方法与去模型的蒙特卡洛、时序差分方法。

- 策略迭代是指当**价值函数给定**时，对策略进行迭代更新，使其更优。
- 策略迭代的一般方法即贪婪决策法、ε贪婪决策法、UCB方法等。

价值迭代与策略迭代交替进行

- 价值迭代使得智能体对当前策略的评估越来越准确；策略迭代使得智能体逐渐改进自己的策略。二者交替进行，在适宜的条件下，可以保证智能体习得最优策略，并求出最优的价值函数。
- 二者交替的过程可以形象理解为：考试测验→学习进步→考试测验→学习进步……的过程。



动态规划

动态规划(DP)的特点

- 需要知道求解的问题的全部信息，例如奖励函数和状态转移方程。但是在实际问题(例如COT搜索, 机器人控制等)上述的信息很难获取
- 在此基础上, DP求解大部分时候是多项式时间复杂度(例如背包问题,最长公共子序列)
- 因此, 面临信息的不完全和高昂的计算代价, 实际使用的算法都可以被看作是取得与DP算法相同的效果的尝试

应用DP的条件

- 需要**最优子结构**和**重叠子问题**
- 最优子结构是指一个给定问题的最优解可以分解成它的子问题的解
- 重叠子问题是指子问题的数是有限的，以及子问题递归地出现，使其可以被存储和重用
- **有限动作和状态空间的 MDP** 满足以上两个性质，贝尔曼方程实现了递归式的分解，价值函数存储了子问题的最优解

回顾: 贝尔曼方程

- 策略函数(policy function): a $= \pi(s)$, 根据状态s决定动作a. 在强化学习中是优化对象
- 价值函数(value function): 更精确地, 称状态价值函数$V^{\pi}(s)$.
  - 评价policy function的优劣的标准之一, 在状态s下, 可以有多个动作a, 每执行一个动作, 系统就会转移到另一个状态(可能是一个在多个状态上的概率分布)
  - 定义: 从当前的状态开始, 到最终状态时系统所获得的**累积回报的期望**. 依赖于状态s和策略π.
  - Formulation: $V^{\pi}(s) = E_{\pi}[R(t)|s_t = s], R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$

回顾: 贝尔曼方程

- 把贝尔曼方程拆分称递归的形式, 有

$$
\begin{aligned}
V^{\pi}(s) &= E_{\pi}\left[R_t \middle| s_t = s\right] \\
&= E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right\} \\
&= E_{\pi}\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_t = s\right\} \\
&= \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_{t+1} = s'\right)\right] \\
&= \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V^{\pi}\left(s'\right)\right]
\end{aligned}
$$

- $\pi(s,a)$是在状态s下选择动作a的概率, $P_{ss'}^a$是选择动作a时从s到s'的概率

策略估计

- 第一步：策略评估

$$v_\pi(s) = \mathbb{E}_\pi[R_t + \gamma v_\pi(S_{t+1}) \,|S_t = s]$$

- 策略评估使用贝尔曼方程来计算状态价值
- 下面是一个经典的例子计算

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

策略估计

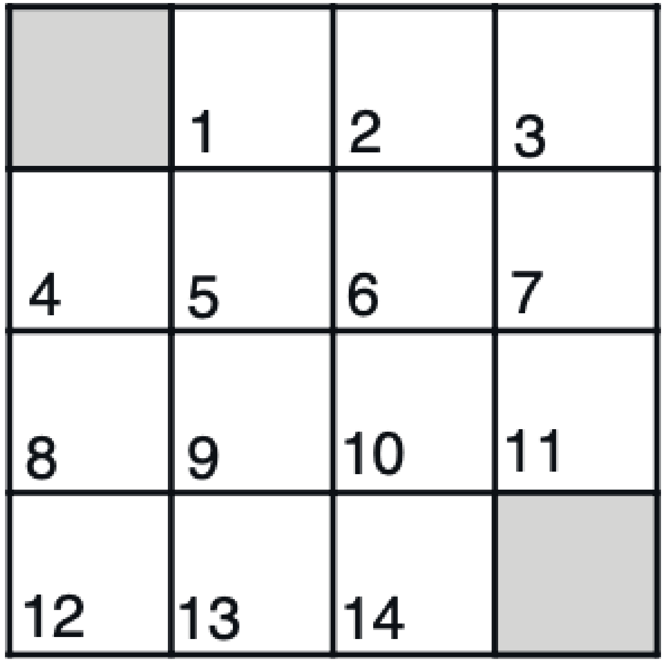
- 终止状态：0 和 15。
- 内部状态：5, 6, 9, 10（动作后确定转移）。
- 边缘状态：如 1, 2, 3, 4, 7, 8, 11, 12, 13, 14（可能因边界限制状态不变）。

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

- 动作：
- 东（+1），南（+4），西（-1），北（-4）。
- 转移规则：
- 内部状态：直接按动作转移。
- 边缘状态：如果超出边界，保持当前状态。
- 终止状态：无动作，价值固定为 0。

策略估计

- 任何在非终止状态间的转移得到的即时奖励均为-1，进入终止状态即时奖励为0。
- 策略：每个动作概率为 0.25



策略估计，例子

```python
def policy_evaluation(policy, gamma=1.0, theta=1e-6):
    V = np.zeros(num_states)  # 初始化价值为 0
    while True:
        delta = 0  # 记录最大变化
        for s in range(num_states):
            if s in terminal_states:
                V[s] = 0  # 终止状态价值固定为 0
                continue
            v = V[s]  # 当前价值
            new_v = 0  # 新价值
            for a in range(num_actions):
                next_state = get_next_state(s, a)
                reward = get_reward(s, next_state)
                new_v += policy[s, a] * (reward + gamma * V[next_state])
            V[s] = new_v
            delta = max(delta, abs(v - new_v))
        if delta < theta:  # 收敛条件
            break
    return V
```

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

策略改进

- 策略改进基于当前评估得到的价值函数
- 如何改进？分析状态-动作价值函数，也就是，在当前状态s，执行动作a之后的期望收益
- 而状态-动作价值函数的计算依赖于之前“估计”的价值函数

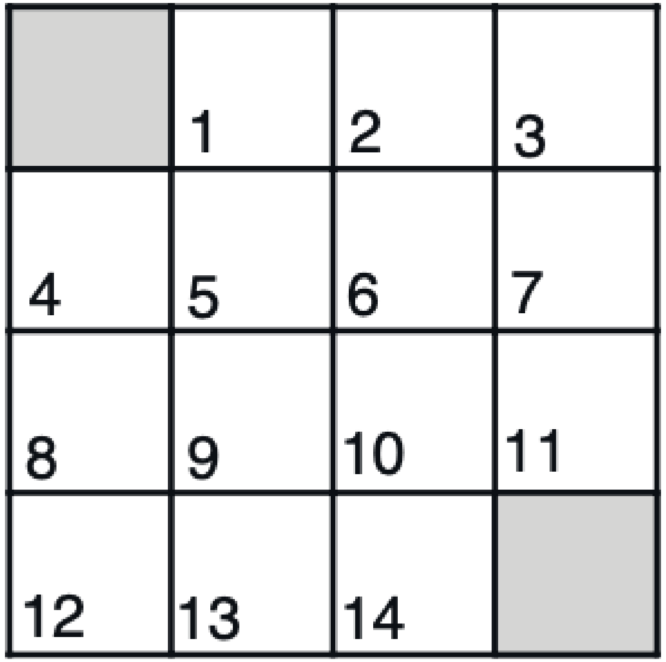
策略改进

- 策略改进基于当前评估得到的价值函数
- 如何改进？分析状态-动作价值函数，也就是，在当前状态s，执行动作a之后的期望收益（下面是之前的例子中的状态-动作价值函数, 借助Grok）
- $$Q(s,a) = R(s,a,s') + \gamma V(s')$$

  - s'：执行动作 a 后确定的下一状态。
  - $R(s,a,s')$：从状态 s 执行动作 a 到达 s' 的即时奖励。
  - γ：折扣因子，在您的问题中 $\gamma = 1$。
  - $V(s')$：下一状态 s' 的价值，由策略评估得到。

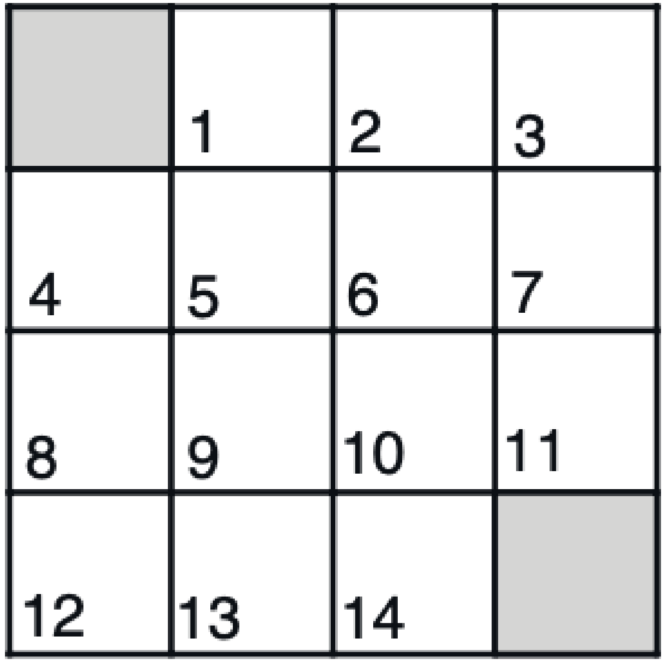
策略改进

- 策略改进基于当前评估得到的价值函数
- 对于状态-动作价值函数Q(s,a)，我们用贪心算法计算对每一个状态s使得Q(s,a)最大的动作a'，作为新的策略



策略改进，例子

```python
def policy_improvement(V, gamma=1.0):
    policy = np.zeros((num_states, num_actions))  # 初始化新策略
    for s in range(num_states):
        if s in terminal_states:
            continue  # 终止状态无动作
        # 计算每个动作的 Q(s, a)
        Q = np.zeros(num_actions)
        for a in range(num_actions):
            next_state = get_next_state(s, a)
            reward = get_reward(s, next_state)
            Q[a] = reward + gamma * V[next_state]
        # 找到最优动作（贪心选择）
        best_action = np.argmax(Q)
        policy[s, best_action] = 1.0  # 将概率设为 1
    return policy
```



小结

- 强化学习中智能体与环境是一对核心范畴，它们的作用分别体现在策略分布与状态转移及奖励的联合分布中。
- 多臂老虎机问题提供了策略迭代的基本方式，同时带领我们初识强化学习与机器学习的异同。强化学习与机器学习的主要差异为：强化学习需要考虑状态间的动态转移，强化学习的奖励反馈与状态转移通常具有随机性且分布无法得知。
- 价值函数提供了一种评估策略优劣的方式，价值迭代与策略迭代的配合，可以使得智能体逐步习得最优策略及其相应的价值函数。

谢谢！