强化学习基础篇（二）马尔科夫决策过程（MDP）

上一篇中主要介绍了强化学习的一些主要组成要素（智能体，环境，奖励，状态以及动作等），以及介绍了强化学习的相关概念。本节主要介绍强化学习的基本数学形式，即马尔科夫决策过程（Markov Decision Processes,MDP）。MDP是序贯决策的经典表达形式，他是强化学习在数学上的理想化形式，因为在MDP这个框架之下，我们可以进行非常精确的理论推导。为了一步步引入MDP，我们将循序渐进地从马尔科夫性质（Markov Process），马尔科夫奖励过程（Markov Reward Process，MRP），再到马尔科夫决策过程（Markov Decision Processes,MDP）。在最后对MDP进行部分扩展，如有限与连续MDPs(Infinite and continuous MDPs)，部分观测MDP（Partially Observable MDPs，POMDP）以及无折扣或平均奖励MDPs(Undiscounted, average reward MDPs)。

马尔科夫决策过程形式上义了强化学习的环境，并且环境是完全可观测状态，即当前的环境状态可以完全表示过程。此外几乎所有的强化学习问题都可以转化为马尔科夫决策过程。比如在最优控制（Optimal Control）理论中主要处理连续马尔科夫决策过程，部分可观测问题也可以转换为马尔科夫决策过程。

1. 马尔科夫性质（Markov Property）

进一步了解马尔科夫决策过程之前，需要先了解马尔科夫性质(Markov Property)，马尔科夫性质即未来的状态只依赖于当前状态，与过去状态无关。

正式的定义如下：

> 马尔科夫性质（Markov Property）定义：
>
> 在时间步$t+1$时，环境的反馈仅取决于上一时间步t的状态s和动作a，与时间步$t-1$以及$t-1$步之前的时间步都没有关联性。
>
> $\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1,\dots,S_t]$

根据定义，当前状态捕捉了历史中所有相关的信息，所以知道了状态，历史(history)就可以完全丢弃。状态是未来的充分统计。然而在实际的环境中，智能体所需完成的任务不能够完全满足马尔科夫性质，即在时间步$t+1$的反馈不一定仅仅依赖于时间步t的状态和动作。但是为了简化问题的求解过程，仍然假设该任务满足马尔科夫属性（Markov Property），并通过约束环境的状态使得问题满足马尔科夫属性。

2.状态转移矩阵(State Transition Matrix)

对于马尔科夫状态s以及后续的状态s',状态转移概率可以定义为：

$$\mathbb{P}_{ss'} = \mathbb{P}[S_{s+1} = s'|S_t = s]$$

状态转移矩阵是马尔科夫过程中状态之间转移的概率所组成的矩阵，因此大小是状态数n的平方。他反映了所有当前状态s以及后续的状态s'的映射，所以他的每行概率之和必定为1。

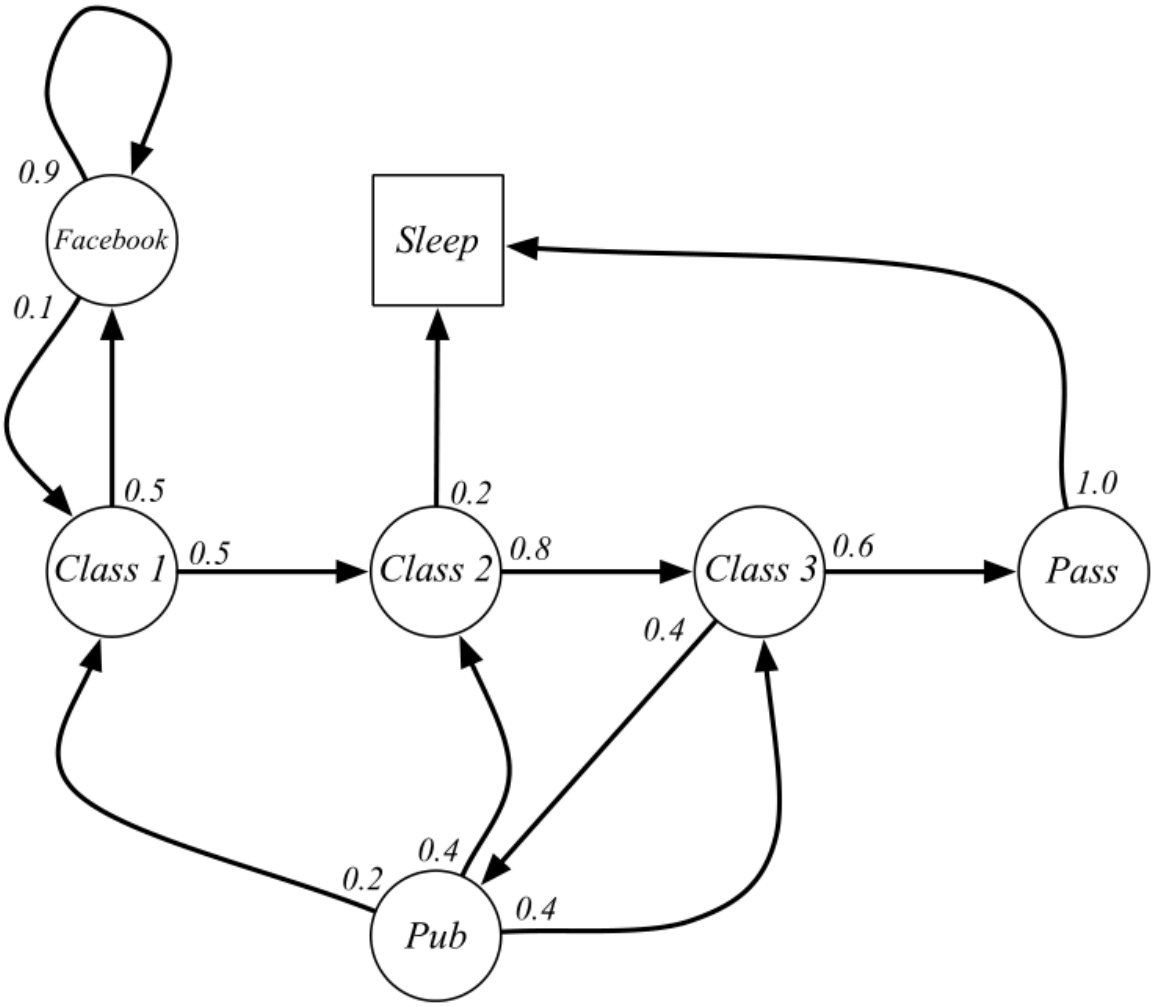
$$\mathbb{P} = \begin{vmatrix} P_{11} & \dots & P_{1n} \\ . & & . \\ P_{n1} & \dots & P_{nn} \end{vmatrix}$$

3.马尔科夫过程(Markov Process)

马尔科夫过程是一个无记忆的随机过程。例如满足马尔科夫性质的状态一系列序列$S1$，$S2$，$S3$，...。

> 马尔科夫过程(Markov Process)定义：
>
> 一个马尔科夫过程（Markov Process），或者马尔科夫链（Markov Chain）由一个二元组构成：(S, P)
>
> - S为有限的状态空间集，s_i表示时间步i的状态，其中$S = \{s_1, s_2, \dots s_n\}$。
> - P为状态转移矩阵，$\mathbb{P}_{ss'} = \mathbb{P}[S_{s+1} = s' | S_t = s]$

接下来举个例子，这里把上课学习过程当做是一个简单的马尔科夫过程中，包含了马尔科夫链的两个元素(S, P)。



这个任务中有7个状态(Facebook, Class 1, Class 2, Class 3, Pass, Pub, Sleep)。在每个状态对应着响应的转移概率，比如在Class1状态下，有0.5概率继续转移到Class2以及0.5的概率转移到Facebook，这里可以注意到每个状态转移概率之和为1。

在这里我们如果对这个马尔科夫过程，从Class1状态开始采样，将会得到很多**幕（episode）**的采样数据。其中每一幕序列可能为：

- Class 1，Class 2, Class 3, Pass, Sleep
- Class 1，Facebook, Facebook, Class 1，Class 2, Sleep
- Class 1，Class 2, Class 3, Pub, Class 2, Class 3, Pass, Sleep

这里的每一幕数据$S_1, S_2, \dots S_T$都会有个终点状态（这里为Sleeep）。

此外在这个例子中，状态转移矩阵为：

$$\mathcal{P} = \begin{array}{c} \\ C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array} \begin{array}{c} \begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{array} \\ \begin{bmatrix} & 0.5 & & & & 0.5 & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{bmatrix} \end{array}$$

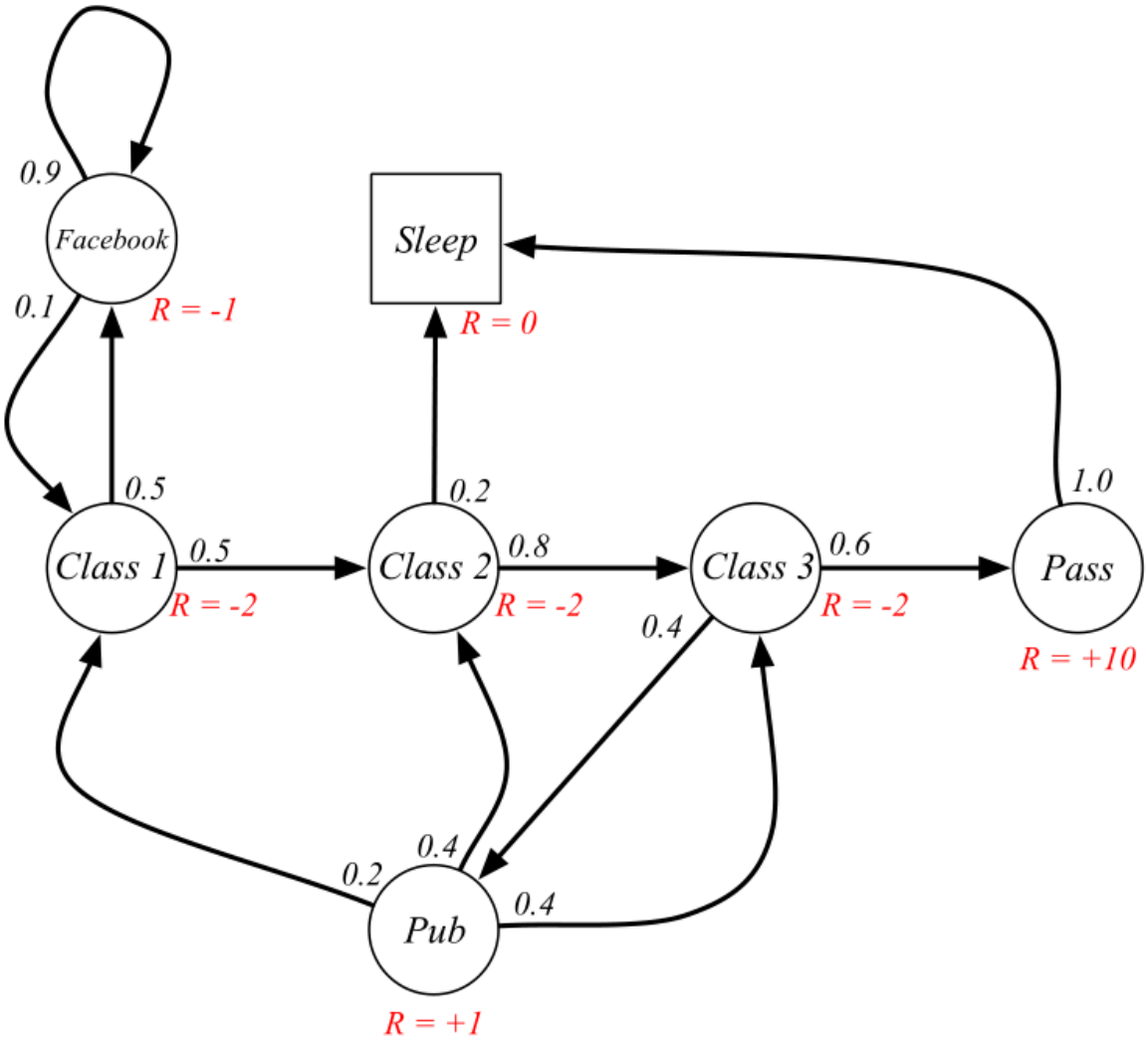
4.马尔科夫奖励过程(Markov Reward Process)

4.1 定义

原本的马尔科夫过程加上奖励值之后就变成了马尔科夫奖励过程，元素由$< S, P >$变为$< S, P, R, \gamma >$。其中R就是奖励，γ 是折扣因子。

> 马尔科夫奖励过程(Markov Reward Process)定义：
>
> 一个马尔科夫奖励过程(Markov Reward Process），由一个四元组构成：$< S, P, R, \gamma >$
>
> - S为有限的状态空间集，s_i表示时间步i的状态，其中$S = \{s_1, s_2, \dots s_n\}$。
> - P为状态转移矩阵，$\mathbb{P}_{ss'} = \mathbb{P}[S_{s+1} = s' | S_t = s]$
> - R为奖励函数，$R_s = \mathbb{E}[R_{t+1} | S_t = s]$
> - γ为折扣因子，$\gamma \in [0, 1]$

我们可以将之前例子中的马尔科夫过程相应的变成如下的马尔科夫奖励过程：



红色标注的值就是奖励，注意的是，这里的奖励是即时奖励，也就是脱离状态时立刻得到的奖励，这里没有考虑下个状态可能转移到哪里。

4.2 累积奖励（Return）与折扣因子γ

为了找到长期累积奖励，不仅要考虑当前时间步t的奖励，还需要考虑到未来的奖励。总奖励（Total Reward）R的计算公式如下：

$$R = r_1 + r_2 + r_3 \ldots + r_n$$

根据总奖励R的计算公式可知，长期累积奖励从当前时间步t开始，直到最终状态的奖励r_n，得到未来累积奖励(Future Cumulative Reward)R_t。

$$R = r_{t+1} + r_{t+2} \ldots + r_n$$

一般而言，环境是随机的或者未知的，这意味着下一个状态可能也是随机的。即由于所处的环境是随机的，所以无法确定下一次执行相同的动作，以及是否能够获得相同的奖励。向未来探索得越多，可能产生的分歧(不确定性)就越多。因此，在实际任务中，通常用折扣未来累积奖励（Discounted Future Cumulative Reward）G_t来代替未来累积奖励。

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^n R_n$$

如果考虑到无限时间的场景，更加通用的表示为（当前获得的奖励表示为R_{t+1}）：

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

其中γ为折扣因子（Discount Factor），是介于$[0,1]$的常数。对于距离当前时间步越远的奖励，其重要性就越低。假设折扣因子$\gamma=0$，可以认为该策略“目光短浅”，只考虑当前的及时奖励r_{t+1}。倘若想平衡当前时间的奖励与未来的奖励，可设置γ为一个较大的值，比如$\gamma=0.9$。如果环境是恒定的，或者说环境的所有状态是已知的（Model-based），那么未来累积奖励可以提前获得并不需要进行折扣计算，这时候可以简单的将折扣因子γ设置为1，看起来就像非常“有远见“。

大多数的马尔科夫奖励过程（MRP）与马尔科夫决策过程（MDP）都会使用折扣因子γ。主要基于如下几点考虑：

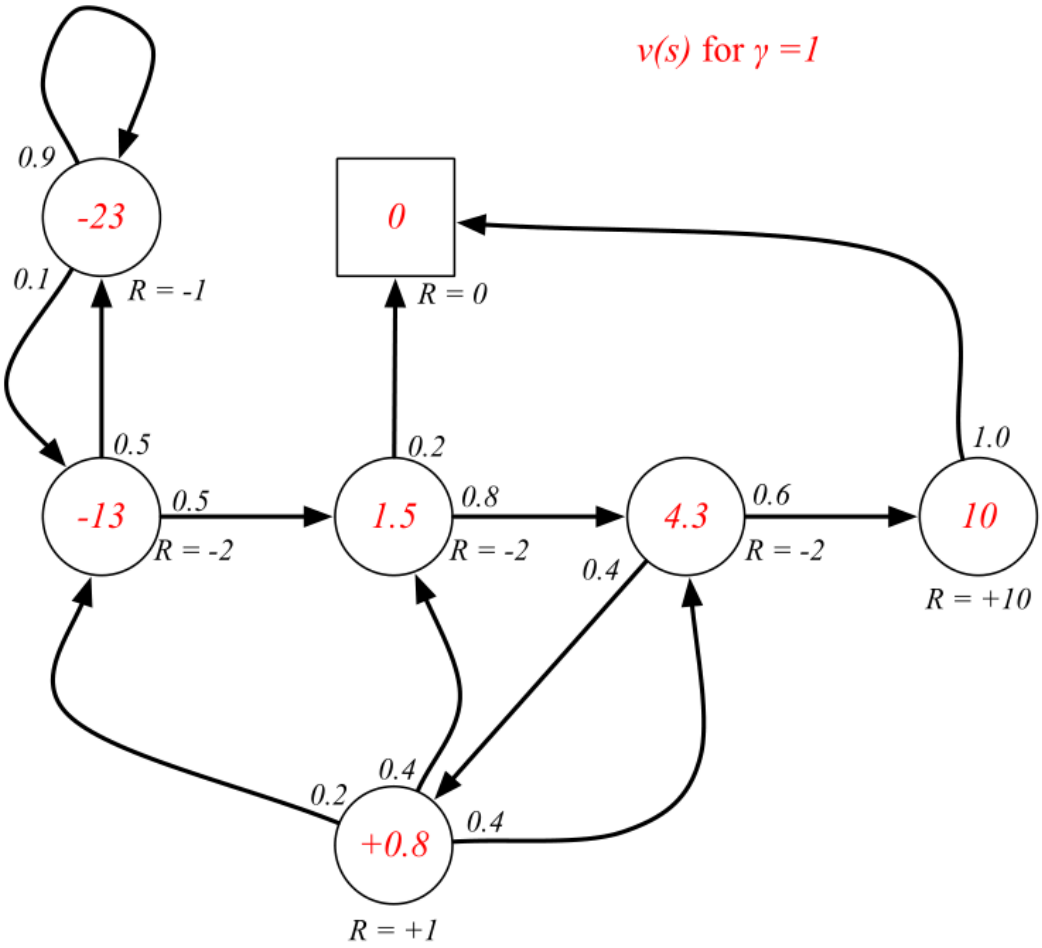
- 数学上计算的便利性
- 避免未来进入循环马尔科夫过程带来的无限大回报
- 未来的不可靠性
- 类比经济学中，眼前的利益比未来的利益更加有意义
- 人类的行为也更加倾向于眼前利益
- 退一步来说，令γ为1，也可以简单的转化成没有折扣因子的状态

4.3. 价值函数

价值函数是状态s的长期价值表示。它是上面说的回报的期望。

$$v(s)=\mathbb{E}[G_t|S_t=s]$$

在概率论和统计学中，数学期望是试验中每次可能的结果的概率乘以结果值得总和。这里回报的期望，即价值函数会会随着折扣因子的变化而取不同的值，因为回报会因为γ的值而改变，期望自然也就会改变。



上图是γ为1时，例子中的马尔科夫奖励过程，红色的就是状态的价值。暂时可以不必纠结上面的价值是怎么算出来的。通过价值函数的公式我们可以知道,要想求解一个状态的状态价值， 需要根据马尔科夫链把所有的可能列出来。每个可能都在终点终止，但是上面这个马尔科夫过程其实是有环的， 可能陷入无限循环的局面。因为折扣因子还是1，所以会导致回报G无限大，期望也就无限大，状态价值V也就无限大。下面引入贝尔曼方程就是为了更好的求解价值函数的。

4.4 马尔科夫奖励过程的贝尔曼方程（Bellman Equation）

贝尔曼方程(Bellman Equation）可以用来方便的表示和计算马尔科夫奖励过程，价值函数可以分为两个部分;

- 即时奖励R_{t+1}
- 下一状态的折扣状态价值$\gamma v(S_{t+1})$

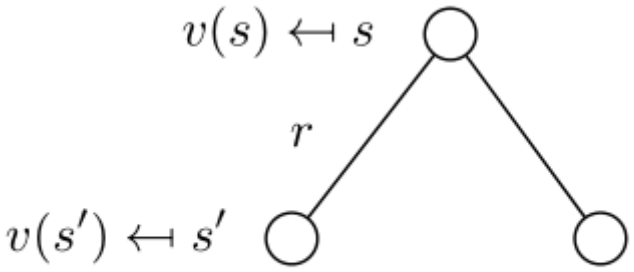
MRP的贝尔曼方程可以进行如下简单的推导:

$$
\begin{aligned}
v(s) &= \mathbb{E}[G_t|S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \ldots) | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s]
\end{aligned}
$$

贝尔曼方程的推导过程中只是简单得使用了累积回报G_t，以及状态价值函数$v(s)$的基本定义。最后即可得出当前状态价值函数$v(s)$的值为，在当前状态s下，即时奖励R_{t+1}与下一状态的折扣状态价值$\gamma v(S_{t+1})$的期望之和。

其核心即阻碍与表示当前状态价值函数$v(s)$与下一刻状态价值函数$v(s_{t+1})$之间的递归关系。

我们可以回溯树的方式更好的理解贝尔曼方程的计算。因回溯图中的关系是回溯运算的基础，也是强化学习的核心内容。通俗的讲，回溯操作就是讲后续状态的价值信息回传给当前时刻的状态。

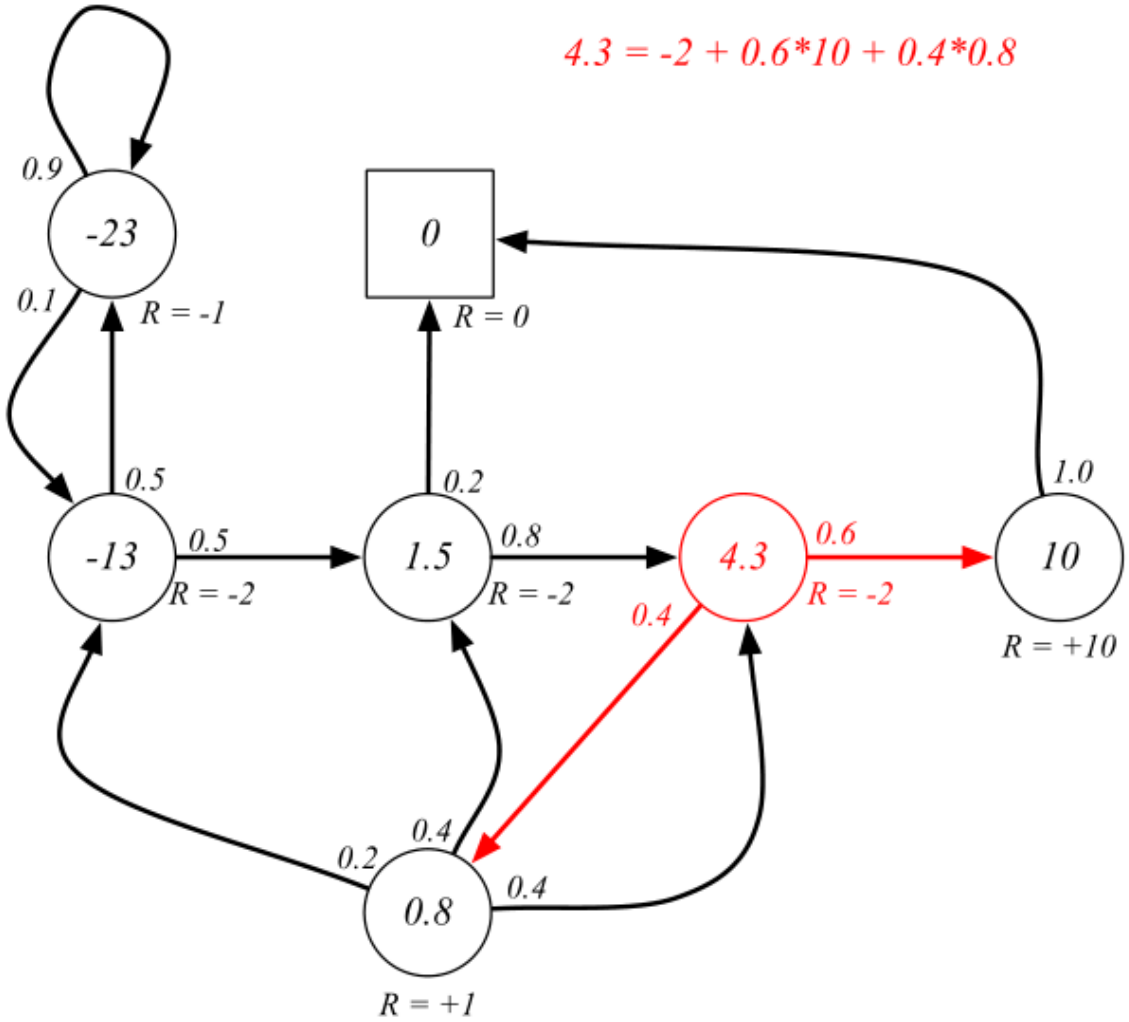


其中空心圆表示一个状态，以状态s作为根节点，智能体依据状态转移概率，可能会到达两个后续状态s‘，以及在离开状态s时获得的奖励r。贝尔曼方程对所有可能性采用其出现概率进行了加权平均。这也就说明了起始状态的价值一定等于后续状态的（折扣）期望值加上对应收益的期望值。

$$
v(s) = \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s]
$$

$$
v(s) = \mathbb{R}_s + \gamma \sum_{s' \in S} \mathbb{P}_{ss'} v(s')
$$

这里通过之前的实例可以进行直观的验证。这里主要观测图中Class3状态下的状态值函数4.3的计算过程。



Class3状态下的即时奖励为$R=-2$，并有两个后续状态（0.6的概率转移到pass状态，0.4的概率转移到pub状态），同时设定了折扣系数γ为1。所以可以计算得出:

$$v(Class3) = \text{即时奖励}R + P_{\text{转移到}pass} * v(pass) + P_{\text{转移到}pub} * v(pub)$$

$$= -2 + 0.6 * 10 + 0.4 * 0.8$$

贝尔曼方程的矩阵形式

贝尔曼方程也可以简单得表示为矩阵的形式:

$$v = R + \gamma P v$$

这里的价值函数v是关于每个状态的价值函数的列向量，矩阵形式展开后的表示如下:

$$\begin{vmatrix} v_1 \\ . \\ v_n \end{vmatrix} = \begin{vmatrix} R_1 \\ . \\ R_n \end{vmatrix} + \gamma \begin{vmatrix} P_{11} & \dots & P_{1n} \\ . & & . \\ P_{n1} & \dots & P_{nn} \end{vmatrix} \begin{vmatrix} v_1 \\ . \\ v_n \end{vmatrix}$$

贝尔曼方程的求解

既然已经能够将贝尔曼方程表示为矩阵形式，并且其为线性方程，那么就可以直接进行求解：

$$
\begin{aligned}
v &= R + \gamma P v \\
(1 - \gamma P)v &= R \\
v &= (1 - \gamma P)^{-1} R
\end{aligned}
$$

对于一些低复杂度的MRP问题，这样直接计算解析解是非常快的，但是由于要计算矩阵的逆，其计算复杂度为$O(n^3)$。所以并不适用与复杂的计算场景，针对复杂MRP问题，一般用迭代的方式求解贝尔曼方程，例如：

- 动态规划（Dynamic programming）
- 蒙特卡洛方法(Monte-Carlo evaluation)
- 差分学习法(Temporal-Difference learning)

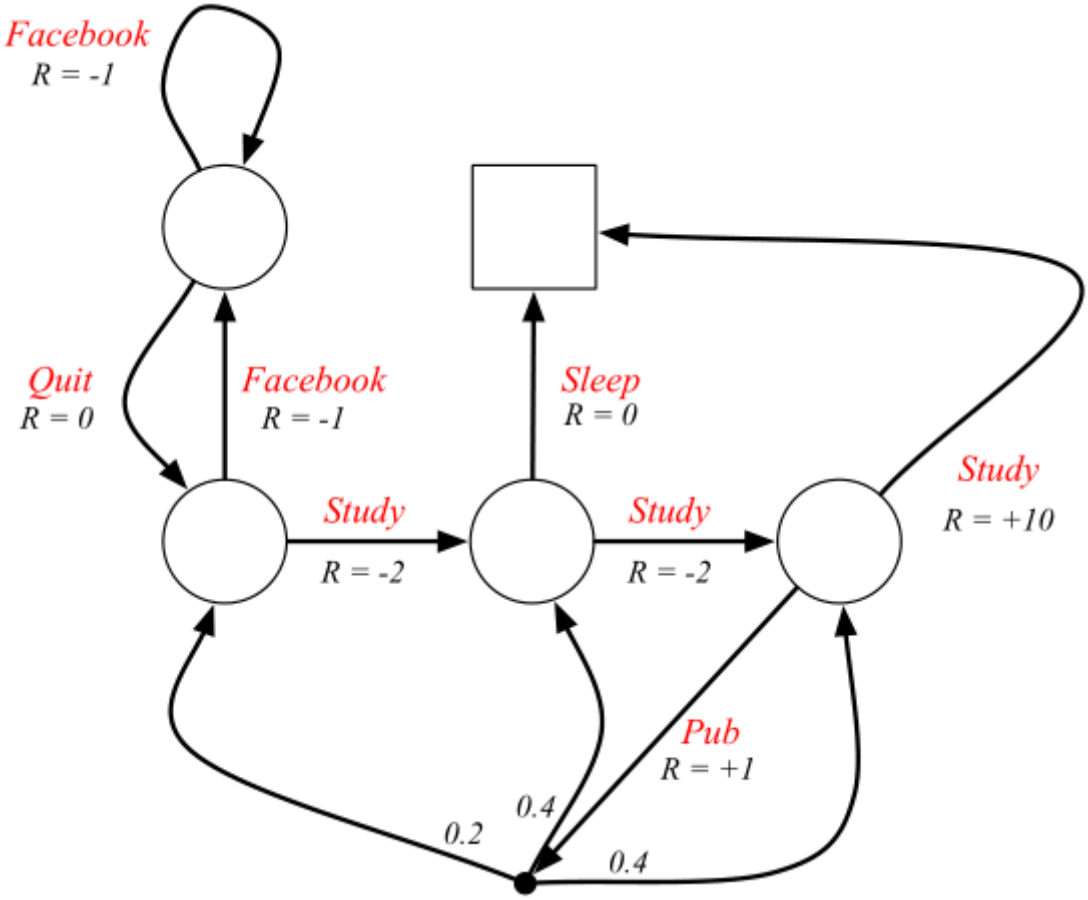
5.马尔科夫决策过程（Markov Decision Processes,MDP）

5.1 马尔科夫决策过程(Markov DecisionProcess)定义

到目前为止其实我们都没有讲到强化学习，因为我们虽然对原始的马尔科夫过程（Markov Process,MP）引入了奖励R而引入马尔科夫奖励过程（Markov Reward Process，MRP），可是我们并没有决策的部分，强化学习本身是一个决策的问题。所以现在我们再引入一个因子，就是动作（Action）。从而将MRP问题变成了.马尔科夫决策过程（Markov Decision Processes,MDP）。此时才能算得上是强化学习。MDP是一个环境，里面的每个状态都满足马尔科夫性质（Markov Property）。

> 马尔科夫决策过程(Markov DecisionProcess)定义：
>
> 一个马尔科夫决策过程(Markov DecisionProcess)，由一个五元组构成：$< S, A, P, R, \gamma >$
>
> - S为有限的状态空间集，s_i表示时间步i的状态，其中$S = \{s_1, s_2, \dots s_n\}$
> - A为动作空间集，a_i表示时间步t的动作，其中$A = \{a_1, a_2, \dots, a_n\}$
> - P为状态转移矩阵，表示在当前状态s下执行动作a后，转移到另一个状态s'的概率分布，可以记为 $\mathbb{P}^a_{ss'} = \mathbb{P}[S_{s+1} = s' | S_t = s, A_t = a]$
> - R为奖励函数，表示在状态s下执行动作a后转移到另一个状态s'获得的奖励，$R^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$
> - γ为折扣因子，$\gamma \in [0, 1]$

为了直观理解MDP过程，还是以之前的例子为例，图中红色的部分就是动作（Action）。



这里需要注意的是，因为有了动作的加入，奖励不再只和状态相关，还和动作有关。之前的奖励是离开状态就获取的即时奖励，现在是
在某个状态下采取特定动作后获取的即时奖励.

接下来我们将介绍在MDP的框架之下的一些基本要素。

5.1 策略（Policy）

策略是状态到动作的映射，在某个状态下采取什么样的动作，可以是确定的策略，也可以是一个随机策略(概率事件）。其定义如下：

$$\pi(a|s) = P[A_t = a|S_t = s]$$

- 策略完整定义了智能体的所有行为方式。
- MDP的策略只依赖于当前的状态，不依赖于历史状态。
- 策略是稳态的，不受时间约束。 即$A_t \sim \pi(.|S_t), \forall t > 0$。
- 在给定一个MDP，$M =< S, A, P, R, \gamma >$以及一个策略π，那么状态序列$S_1, S_2, \ldots$，可以表示的前面章节描述的马尔科夫过程(MP)为$< S, P^\pi >$。
- 在给定一个MDP，$M =< S, A, P, R, \gamma >$以及一个策略π，那么状态与奖励序列$S_1, R_2, S_2, R_3 \ldots$，可以表示前面章节描述的的马尔科夫奖励过程（MRP）为$< S, P^\pi, R^\pi, \gamma >$。

上述中的状态转移矩阵与奖励函数定义为：

$$P^\pi_{s,s'} = \sum_{a \in A} \pi(a|s) P^a_{ss'}$$

$$R^\pi_s = \sum_{a \in A} \pi(a|s) R^a_s$$

5.2 价值函数（Value Function）

MDP的价值函数和MRP的有一些不同，与策略相关。正因为有了策略，价值函数不再单纯的只和状态s相关了。采取不同的策略，价值函数也会不同。因为从贝尔曼方程中我们也能看出，价值的计算和动作相关，而动作的选择就是策略。但是这里不得不提一下，这里的价值函数只是策略的价值函数，它的好坏不一定代表真正的状态的好坏，它只是根据你提供的这个策略计算出来的，提供的这个策略不一定是一个好策略,那么自然计算出来的价值不一定具有很强的参考性。

当执行到某一步时，如果需要评估当前智能体在该时间步状态的好坏程度，主要由价值函数（Value Function）来完成。由于价值函数的输入分为状态s和<状态，价值>对$< s, a >$，所以通常，当输入状态时统称为状态值函数，输入<状态，价值>对$< s, a >$时统称为动作值函数，当不讨论其输入时，统称为价值函数。

一个马尔科夫决策过程的状态值函数$v_\pi(s)$是对未来奖励的预测，表示在状态s下，跟随给定的策略π会得到的奖励期望。

$$v_\pi(s) = E_\pi[G_t|S_t = s]$$

一个马尔科夫决策过程的动作值函数$q_\pi(s, a)$，表示在状态s下，执行动作a，并跟随给定的策略π会得到的奖励期望。

$$q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a]$$

5.3 马尔科夫决策过程的贝尔曼方程（Bellman Equation）

与在马尔科夫奖励过程的贝尔曼方程思想相同，状态值函数可以分解为即时奖励R_{t+1}与下一状态的折扣状态价值$\gamma v(S_{t+1})$的期望之和。

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]$$

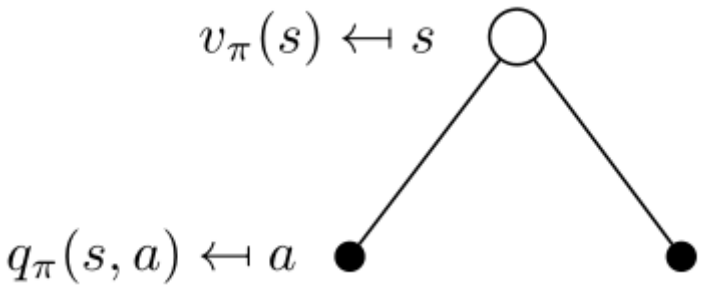
同时动作值函数也可以进行类似的分解：

$$q_\pi(s, a) = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a]$$

为了方便理解，可以画出响应的状态值函数与动作值函数的回溯图。

状态值函数V^π的回溯图

我们以根节点的空心圆表示当前的状态s，以实心圆表示在当前状态智能体根据策略π可能采取的动作a。

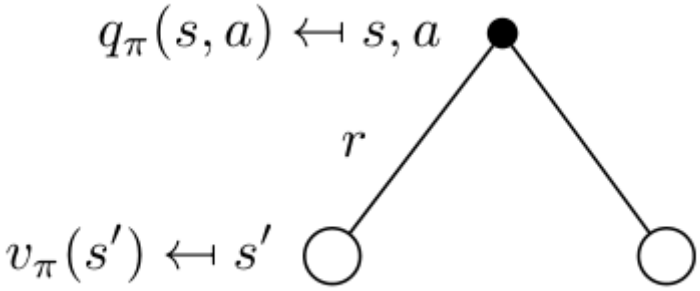


状态值函数即可表示为对所有可能的动作产生的动作值函数进行了加权平均。

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

动作值函数Q^{π}的回溯图

我们以根节点的实心圆表示当前智能体根据策略π已经选取的的动作a，以空心圆表示在当前动作之后，智能体可能的下一步状态s'。其中也包含了在动作执行后得到的奖励r。

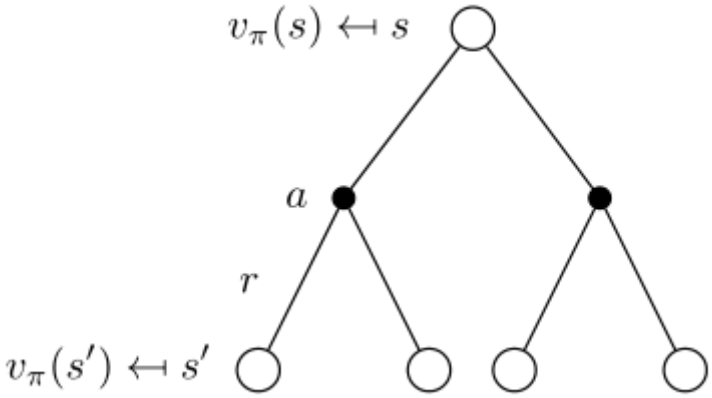


动作值函数即可表示为，执行当前动作获得的奖励R_s^a，加上对所有可能的未来状态的状态值函数进行了加权平均。

$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')$$

状态值函数V^{π}的二次展开的回溯图

根据前面状态值函数V^{π}的的溯图，进行下一个时间步的展开，考虑在动作后的状态。我们可以得到展开后的回溯图。

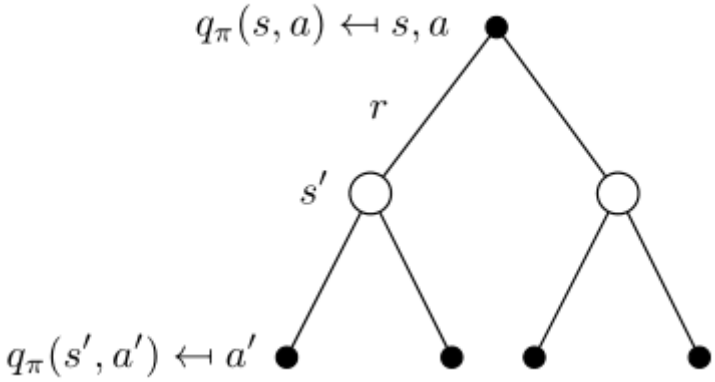


状态值函数即可表示为对所有可能的动作产生的动作值函数的详细二次分解进行了加权平均。

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

动作值函数Q^{π}的二次展开回溯图

根据前面动作值函数Q^{π}的回溯图，进行下一个时间步的展开，考虑在下一状态后，根据策略π可能采用的新的动作。我们可以得到展开后的回溯图。

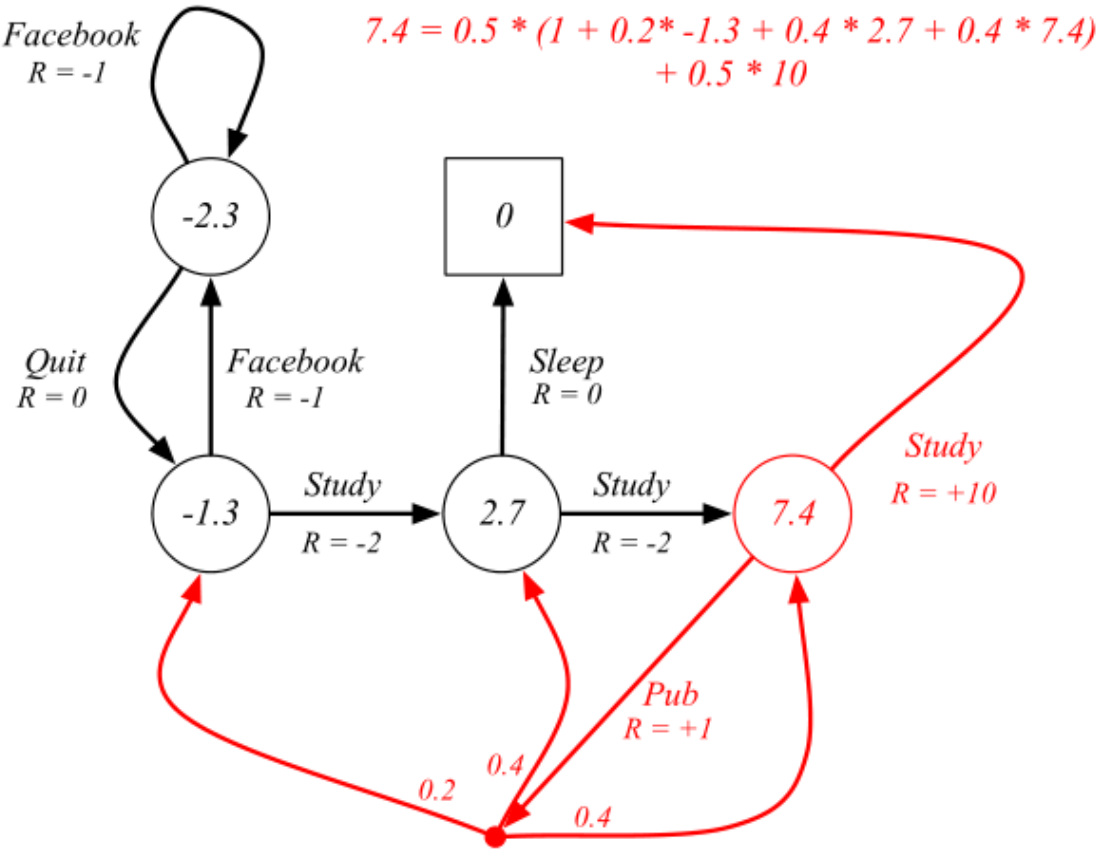


动作值函数即可表示为：

$$q_\pi(s,a) = R_s^a + \gamma \sum_{s‘ \in S} P_{ss'}^a \sum_{a' \in A} \pi(a‘|s') q_\pi(s‘,a')$$

贝尔曼方程求解示例

下面我们顺着之前的例子，看看贝尔曼方程的求解过程：



这里我们需要求解Class3状态下的状态值函数$v(class3)$。

$$\begin{aligned}
v(Class3) &= \sum_{a \in A} \pi(a|s) q_\pi(s,a) \\
&= P_{pub} * q_\pi(Class3, pub) + P_{study} * q_\pi(Class3, sleep) \\
&= P_{pub} * (R_{pub} + v(pub)) + P_{study} * (R_{study} + v(sleep)) \\
&= P_{pub} * (R_{pub} + P_{pub-class1} * v(class1) + P_{pub-class2} * v(class2) + P_{pub-class3} * v(class3)) + P_{study} * (R_{study} + v(sleep)) \\
&= 0.5 * (1 + 0.2 * (-1.3) + 0.4 * 2.7 + 0.4 * 7.4) + 0.5 * (10 + 0) \\
&= 7.4
\end{aligned}$$

MDP下贝尔曼方程的矩阵形式

在马尔科夫决策过程下的贝尔曼方程的矩阵形式与在马尔科夫奖励过程下的矩阵形式类似，只是考虑了策略π。

$$v_\pi = R^\pi + \gamma P^\pi v_\pi$$

其直接的解析解为：

$$v_\pi = (1 - \gamma P^\pi)^{-1} R^\pi$$

计算复杂度为$O(n^3)$。所以并不适用与复杂的计算场景，针对复杂MRP问题，一般用迭代的方式求解贝尔曼方程。

5.3 最优值函数（Optimal Value Function）

由之前的介绍可知，强化学习的目标就是求解马尔科夫决策过程的最优策略，而值函数是对最优策略π^*的表达。

最优值函数（Optimal Value Function）分为最优状态值函数（optimal state-value function)与最优动作值函数（optimal action_value function），他们的定义如下：

> 最优状态值函数（optimal state-value function）$v_*(s)$就是在所有策略π当中，最大的状态值函数：
>
> $$v_*(s) = \max_\pi v_\pi(s)$$
>
> 最优动作值函数（optimal action_value function）$q_*(s,a)$就是在所有策略π当中，最大的动作值函数：
>
> $$q_*(s,a) = \max_\pi q_\pi(s,a)$$

我们更希望的是最优值函数，最优值函数是与策略无关的，根据最优值函数可以直接得到最优策略。只需要沿着状态价值函数大的方向移动就行，这其实也就是强化学习中的一个流派，基于值学习的(相对于基于策略学习)。值比较大的状态就是好，比如终点前的一个状态值一般就比较大,因为下一刻就可以结束。此处用"一般"是因为考量的是状态值，如果这个状态不仅和终点相连并且还和几个失败点相连，状态值不一定大。参考贝尔曼方程计算公式，如果我们使用另一种动作值函数，代表状态s下采取特定动作a的价值。那么我们就可以说,终点前一个状态，采取动作a可以直接到终点，这个$q(s,a)$一定是一个大值。这也提示了使用$q(s,a)$值一般比使用状态价值$v(s)$更好，因为考虑了动作。

最优值函数确定了马尔科夫决策过程中智能体的最优的可能表现。获得了最优值函数，也就获得了么个状态的最有价值，那么此时马尔科夫决策过程的所有变量都为已知的，接下来便能够很好的求解马尔科夫决策过程的问题。

5.3 最优策略（Optimal Policy）

最优策略（Optimal Policy）的定义为：

$$\pi \geq \pi' \quad if \quad v_\pi(s) \geq v_{\pi'}(s), \forall s$$

在状态s下，当策略π的价值函数优于任何其他策略π'的价值函数时，策略π即为状态s下的最优策略。关于马尔科夫决策过程的最优策略，有如下3个定理：

> （1）对于任何马尔科夫决策过程问题，存在一个最优策略π^*，其优于（至少等于）任何其他策略，即$\pi^* \geq \pi$。
>
> （2）所有最优策略下都有最优状态值函数，即$v_{\pi^*}(s) = v^*(s)$
>
> （3）所有最优策略下都有最优动作值函数，即$q_{\pi^*}(s,a) = q^*(s,a)$

基于上述三个定理，寻找最优策略可以通过最优状态值函数$v_{\pi^*}(s)$或者最优动作值函数$q_{\pi^*}(s,a)$来得到。也就是说如果最优值函数已知，则可以获得马尔科夫决策过程的最优策略。

因此，可以通过最大化$q^*(s,a)$得到最优策略π^*，具体的定义如下：

$$\pi_*(a|s)=\begin{cases}1, & if\ \ a=\max_{a\in A}\ q(s,a)\\ \\ 0, & \text{其他}\end{cases}$$

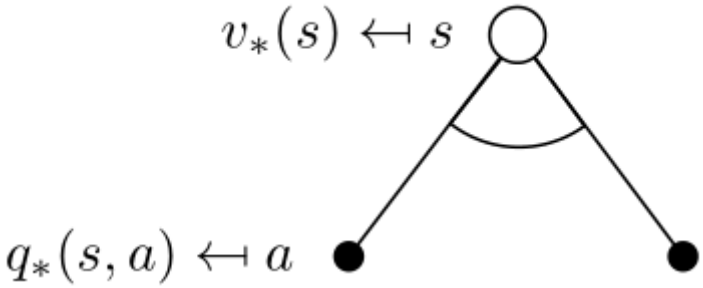
上式中，$a=\max_{a\in A}\ q(s,a)$时，$\pi(a|s)$为1，表明如果动作值函数的最大值为最优策略所选择的动作，那么智能体已经找到最优策略π^*。只要最优动作值函数$q^*(s,a)$已知，就可以立即获得最优策略。综上所述，最优策略π^*对于任何马尔科夫决策过程都会有一个对应的确定性最优策略$\pi_*(a|s)$。

到目前为止，最优策略的求解问题事实上已经转换成为了最优值函数的求解问题。如果已经求出最优值函数，那么最优策略是非常容易得到的，反之同理。通过最优策略求解问题的转换，可以将鼓励的最优策略π^*、最优值函数$v^*(s)$、最优动作值函数$q^*(s,a)$连为一体。需要注意的是，在实际工作中，也可以不求最优值函数，而使用其他方法直接求解最优策略。

最优值函数的贝尔曼方程与回溯图

我们若以回溯图的形式展示v_*和q_*的贝尔曼方程中所进行的后续状态和动作的扩展过程。除了在智能体的选择节点处加入了弧线表示应该在给定策略下取最大值而不是期望值之外，其他都和之前介绍的v_π以及q_π的回溯图相同。

最优状态值函数$v_*(s)$表示如下：

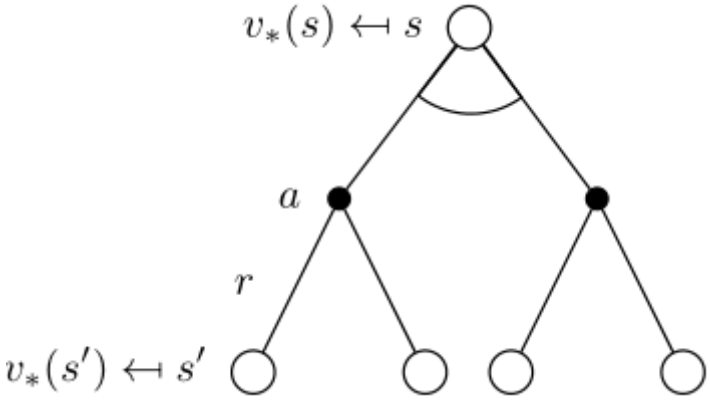


$$v_*(s)=\max_a q_*(s,a)$$

最优动作值函数$q_*(s,a)$表示如下：

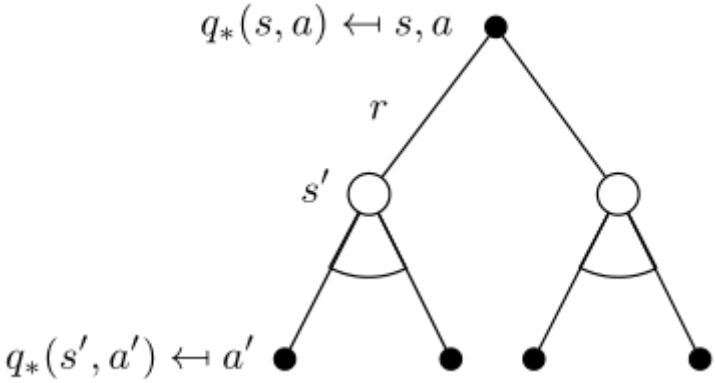
$$q_*(s,a)=R_s^a+\gamma\sum_{s`\in S}P_{ss'}^a v_*(s')$$

最优状态值函数$v_*(s)$的二次展开的回溯图表示如下：



$$v_*(s)=\max_a\ (R_s^a+\gamma\sum_{s`\in S}P_{ss'}^a v_*(s'))$$

最优动作值函数$q_*(s,a)$的二次展开的回溯图表示如下：



$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} q_*(s', a')$$

5.4 求解贝尔曼最优方程

上面描述的回溯图过程中已经给出了贝尔曼最优方程的基本形式，其阐述的一个事实是：最优策略下各个状态的价值一定等于这个状态下最优动作的期望回报。我们已经知道，

- 求解强化学习问题实际上是求解最优策略。
- 最优策略可以通过求解最优值函数得到。
- 最优值函数的求解就是优化贝尔曼方程。

简而言之，强化学习的求解最后演化成了优化贝尔曼方程。贝尔曼最优方程实际上是一个方程组，每个状态对应一个方程等式。也就是说，如果有n个状态，那么有n个含有n个未知量的方程。如果环境的动态变化特性p是已知的，那么原则上可以用解非线性方程组的求方法来求解v_*方程组。类似地，我们也可以求得q_*的一组解。对于小规模的马尔科夫决策过程，可以直接求解价值函数，对于大规模的马尔科夫决策过程，通常非常难以获得解析解，必须采用迭代的方法优化贝尔曼方程。比如：

- Value Iteration算法
- Policy Iteration算法
- Q-learning算法
- Sarsa算法

6.马尔科夫决策过程的扩展

除了上述的标准的MDP，还有很多MDP的拓展情形，比如无限的和连续的MDP(infinite and continuous MDP)，部分可观测的MDP(partially observable MDPs)，非折扣的、平均奖励的MDP等。

6.1 无穷或连续 MDPs (Infinite and continuous MDPs)

如果考虑到动作空间或者状态空间的的连续性，MDP会有如下几种扩展：

- 动作空间或状态空间的大小为无穷可数。
- 动作空间或状态空无限不可数（连续），这种情况在线性最小二乘（Linear quadratic，LQR）模式中有闭式解。
- 时间连续的MDP，可以参考《Adaptive optimal control for continuous-time linear systems based on policy iteration》

6.2 部分可观测 MDPs（Partially observable MDPs, POMDPs)

我们已经知道MDP模型是根据系统当前的实际状态做出决策的，MDP的一个重要前提是智能体对环境的观察室完备的，而现实世界中智能体只能观察到部分信息，比如很多情况下，难以获得系统的精确状态。尤其对复杂的机械系统来说，测量系统状态的传感器信号通常会收到噪声污染，难以获得系统的精确状态。针对这样的问题，一个更接近现实世界的模型被提出，即部分可观测马尔科夫决策过程（POMDP）。它比MDP更加接近一般的决策过程，除了他的前提是部分可观测之外，他也基于MDP类似的假设，采用的是最大化期望奖励的方法。因此POMDP可以被看做是MDP的扩展，他的状态空间包括所有定义在对应的MDP的状态集合上的概率分布，这些概率分布表现了信念状态。

POMDP模型

> 部分可观测马尔科夫决策过程(Partially observable MDPs, POMDPs)定义：
>
> 一个部分可观测马尔科夫决策过程(Partially observable MDPs, POMDPs)，由一个七元组构成：$< S, A, O, P, R, Z, \gamma >$
>
> - S为有限的状态空间集，s_i表示时间步i的状态，其中$S = \{s_1, s_2, \dots s_n\}$
> - A为动作空间集，a_i表示时间步t的动作，其中$A = \{a_1, a_2, \dots, a_n\}$
> - O为有限的观察结果空间集。
> - P为状态转移矩阵，表示在当前状态s下执行动作a后，转移到另一个状态s'的概率分布，可以记为 $\mathbb{P}^a_{ss'} = \mathbb{P}[S_{s+1} = s' | S_t = s, A_t = a]$
> - R为奖励函数，表示在状态s下执行动作a后转移到另一个状态s'获得的奖励，$R^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$
> - Z为观察函数，表明系统状态和观察值之间的关系，即，智能体在执行动作a进入环境状态s'后得到的观察值的概率。$Z^a_{s'o} = \mathbb{P}[O_{t+1} = o | S_{t+1} = s', A_t = a]$。
> - γ为折扣因子，$\gamma \in [0, 1]$

注意，O为有限的观察结果空间集，比如机器人的传感器获得的环境数据。在MDP中，由于智能体完全了解系统状态，因此$O \equiv S$，而在部分可观测环境中，观察仅在概率上取决于潜在的环境状态。因为在不同的环境状态中可以得到同样的观察，因此确定智能体所处的状态变得很困难。

假设在时刻t，此时的环境处于状态$s \in S$。智能体采取动作$a \in A$，根据状态转移矩阵$P(s'|s, a)$进入环境状态s'，同时智能体获得环境的观察值$o \in O$，这取决于环境的新状态，概率为$Z(o|s', a)$。最后智能体得到奖励值R。重复上述步骤。整个过程的目的是让智能体在每个时间步选择的动作可以最大化未来期望的折扣奖励:

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$$

其中r_t是在时间步t获得的奖励，折扣因子的大小决定了最近的奖励和未来的奖励哪个对智能体影响更大。当$\gamma = 0$时表示智能体只关心下一次动作选择哪个动作可以最大化奖励，因为之后的收益为0；当$\gamma = 1$时表示智能体更关心接下来所有动作的奖励之和。

信念状态（Belief States)

在POMDP中，智能体不能确信自己处于哪个状态，因此对于下一步动作的选择的决策基础是当前所处状态的概率，也就是说最有可能处于哪个状态。所以，智能体需要通过传感器收集环境信息，也就是得到观察值，来更新自己对当前所处状态的可信度。“信息收集”的动作并不是直接把智能体导向目标点，而是作为一个缓冲，让智能体先运动到邻近位置，在这个邻近位置上收集到的环境信息加大了智能体对自己所处状态的可信度。在非常确信自己所处的状态之后，智能体做出的动作决策才是更加有效的。因此，在每个时间周期，智能体虽然无法准确得知其所处的环境状态，但是它可以通过观察得到当前状态的不完整信息。通过观察和动作的历史来做决策，我们把t时刻观察和行为的历史定义为:

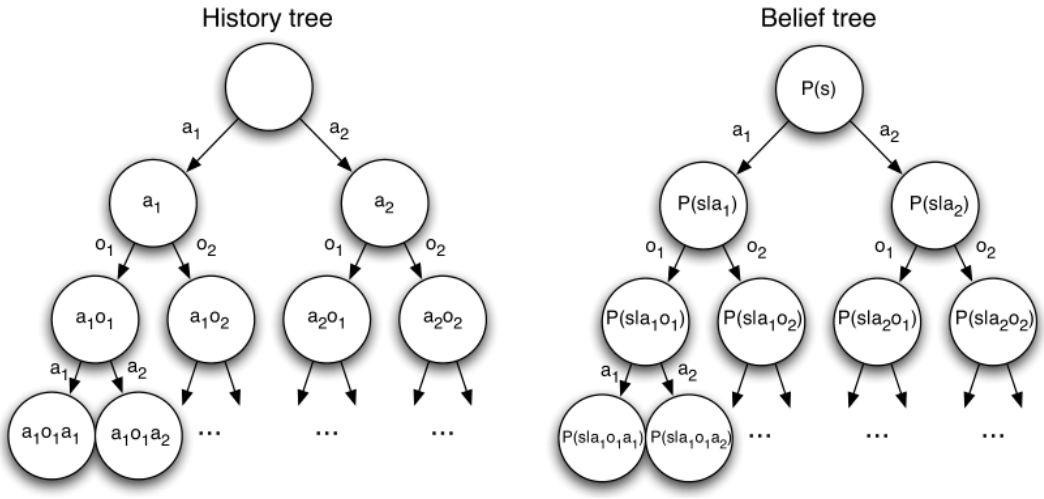
$$h_t = (a_0, o_1, \ldots, o_{t-1}, a_{t-1}, o_t)$$

这样的描述会消耗很大的存储空间，为了解决这个问题，一些学者研究了如何对过去的历史进行压缩表示，即采用较短的历史代替所有的观察和行为，Astrom提出用状态上的概率，这样就引入了信念状态$b(s)$的概念，表示智能体对自己当前所处状态的可信度。

$$b(h) = (\mathbb{P}[S_t = s^1 | H_t = h], \ldots, \mathbb{P}[S_t = s^n | H_t = h])$$

Sondik证明信念状态$b(s)$是对历史h_t的充分估计，所有状态上维护一个概率分布可以与维护一个完整历史提供同样的信息。以前解决POMDP问题时，需要知道历史动作才能决定当前的操作，这种解决方案是非马尔科夫链，然而引入信念状态后，POMDP问题就可以转化为基于信念空间的马尔科夫链来求解。因此简单的说，我们把POMDP转换为求解信念状态函数和策略问题。

他们之间满足的马尔可夫性可以表示为如下的树形结构：



6.3 无折扣，平均奖励MDPs(Undiscounted，Average reward MDPs)

遍历马尔科夫过程（Ergodic Markov Process)

马尔可夫链的遍历性通俗的理解就是任意取一个时间段，所有状态都有出现的可能。遍历链是非周期的平稳马尔可夫链，有长时间尺度下的稳态行为，因此是被广泛研究和应用的马尔可夫链。

- 周期性(Recurrent)：每个状态都被访问了无数次
- Aperiodic(非定期的)：访问每个状态都没有任何系统的周期

遍历马尔可夫过程具有以下性质的极限平稳分布：

$$d^{\pi}(s) = \sum_{s' \in S} d^{\pi} P_{s's}$$

> 遍历马尔科夫决策过程定义：
>
> 如果任何策略引起的马尔可夫链都是遍历的，则MDP是遍历的。
>
> An MDP is ergodic if the Markov chain induced by any policy is ergodic.

对于任何策略π，遍历MDP的每个时间步长ρ^{π}的平均奖励与开始状态无关。

$$\rho^{\pi} = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}[\sum_{t=1}^{T} R_t]$$

平均奖励的值函数（Average Reward Value Function）

- undiscounted, ergodic MDP(未打折的遍历MDP)的价值函数可以用平均奖励来表示。
- $\hat{v}_\pi$是从状态s开始的额外奖励。

$$\hat{v}_\pi = \mathbb{E}[\sum_{k=1}^{\infty}(R_{t+k} - \rho^\pi)|S_t = s]$$

有着相应的平均奖励贝尔曼方程：

$$\hat{v}_\pi = \mathbb{E}[(R_{t+1} - \rho^\pi) + \sum_{k=1}^{\infty}(R_{t+k+1} - \rho^\pi)|S_t = s]$$

$$= \mathbb{E}[(R_{t+1} - \rho^\pi) + \hat{v}_\pi(S_{t+1})|S_t = s]$$