# 马尔科夫决策过程（MDP）

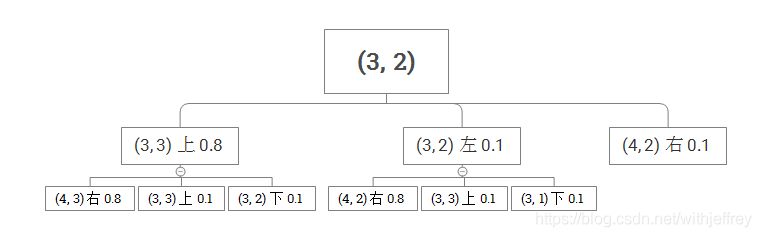
1. making complex decisions

## 一、马尔科夫过程（Markov Process）

## 现在设定一个如图所示的游戏，规则是智能体从Start位置开始，每步只能按照上下左右的动作走一格，当到达+1和-1时游戏结束，当智能体到达+1则表示赢得了游戏。如果智能体选择其中一种动作，而且该动作100%会执行或者说发生，那么很明显可以按照上上右右右或者右右上上上的顺序依次执行，达到+1。但是如果这是一个充满着“不确定”的世界，当智能体选择某一种动作时，只有80%的概率到达动作期望的状态，而20%的概率则是与所选择动作呈90°的方向到达另一个状态。比如说，从（1，1）开始，如果选择上，并且最终到达状态（1，2）的概率只有80%，而实际执行右并到达（2，1）有10%概率，实际执行左将碰到墙壁而保持不动有10%概率。我们用转换模型（Transition Model）来表示这样一组从一个状态因采取不同的动作到达另一个状态的概率模型。

## 

## 现在我们来定义效用函数。定义一个动作序列的效用函数就是最后状态的效用减去0.04\*动作序列的长度，也就是说，每一个除了+1和-1的状态，都有-0.04的“奖励”，而一系列的状态的奖励之和可以计算出其效用。比如说动作序列长度为6，那么一系列动作后到达+1，那么效用为0.76。在一个“确定”的世界，智能体在给定的动作下可以到达期望的状态，而在一个“不确定”的世界，智能体虽然给定了动作，但是有一定的概率达到期望的状态。比如说，智能体在（3，2），动作是（上，右），那么它到达（4，3）状态的概率是0.64。具体过程如下图所示：



## 它可能到达的状态有5个：

## 到达（4，3）的概率为0.8\*0.8=0.64

## 到达（3，3）的概率为0.8\*0.1+0.1\*0.1=0.09

## 到达（3，2）的概率为0.8\*0.1=0.08

## 到达（4，2）的概率为0.1\*0.8+0.1=0.18

## 到达（3，1）的概率为0.1\*0.1=0.01

## 正好所有的概率之和为0.64+0.09+0.08+0.18+0.01=1

## 一个完整的从状态到动作的映射称之为策略。给定一个策略，可以计算出该策略的期望效用。所以，我们要解决的问题也就是计算出最佳的策略，即可以最大化期望效用的策略。这样一个理想和随机的世界，通过转换模型寻找最佳策略的问题称之为马可夫决策问题（MDP），以俄罗斯的统计学家Andrei A. Markov命名。

## 二、马尔科夫奖励过程（Markov Reward Process）

马尔科夫奖励过程是在马尔科夫过程基础上增加了奖励函数(R)和衰减系数(γ)， 用 (<S,R,P,γ>)表示

(R) : 表示 (S) 状态下某一时刻的状态(S\_t) 在下一个时刻 ((t + 1)) 能获得的奖励的期望:R\_s = E[R\_{t+1}|S\_t=s]

γ: 折扣因子(Discount factor ),γ ∈ [0, 1]）

* 为了避免出现状态循环的情况
* 系统对于将来的预测并不一定都是准确的，所以要打折扣
* 很显然γ越靠近1，考虑的利益越长远。

(V(s)):状态价值函数（state value function）,表示从从该状态开始的马尔科夫链收获的效用

为了获得属性的简单表达式，我们需要做出某种偏好独立假设。最自然的假设是，主体在状态序列之间的偏好是平稳的。平稳性是一个看起来相当无害的假设，具有非常强的结果：事实证明，在平稳性条件下，只有两种连贯的方法可以将效用分配给序列：

* 加性奖励

Uh([s0,s1,s2, ...]) = R(s0)+ R(s1)+ R(s2)+ ···

我们在启发式搜索算法中使用路径代价函数时隐式地使用了可加性

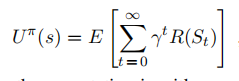
* 折扣奖励

Uh([s0 ,s1,s2,...]) = R(s0)+ γR(s1)+ γ2R(s2)+ ···

**三、马尔科夫决策过程（Markov Decision Process）**

马尔科夫决策过程是在马尔科夫奖励过程的基础上加Decision过程，相当于多了一个动作集合，可以用 (<S, A, P, R, γ>)表示。在确定了给定状态序列的效用是在序列中获得的折扣奖励的和之后，我们可以通过比较执行策略时获得的预期效用来比较策略。我们假设代理处于某个初始状态s，并定义了st（一个随机变量）是执行在特定策略π时代理在t时到达的状态。（显然，S0=s，即代理程序现在所处的状态。)在状态序列S上的概率分布1,S2, ...，由初始状态s、策略π和环境的转换概率决定。

通过以s开始执行π所得到的期望效用由



其中，期望是关于由s和π确定的状态序列上的概率分布。现在，在代理可以选择从s开始执行的所有策略中，有一个（或多个）将具有比其他所有策略更高的预期实用程序。

我们将使用π∗s 要表示以下策略之一：

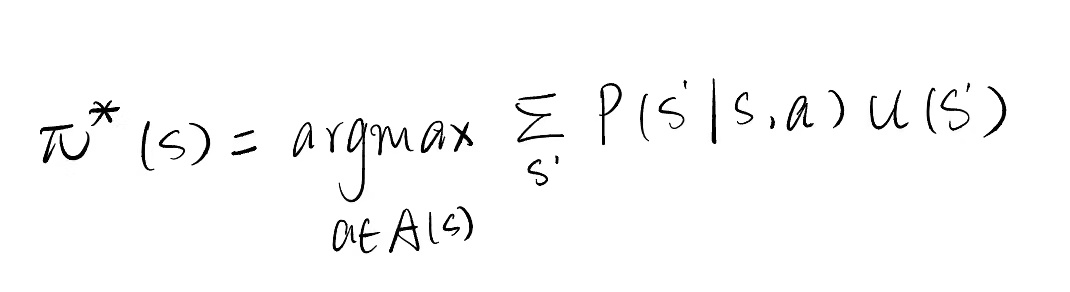
π\*s = argmax Uπ(s) .

π\*s是一个策略, 它与s的联系是，当s是起始状态时，它是一个最优策略。使用具有无限视界的折现效用的一个显著结果是，最优策略独 立于起始状态。

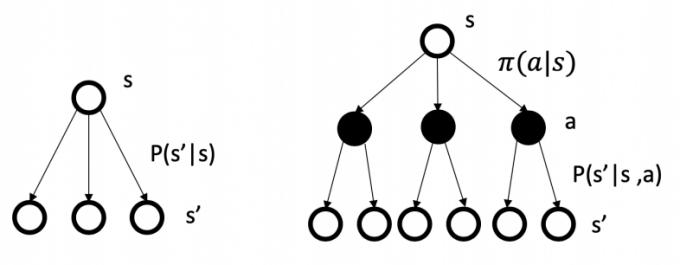
注意，U(s)和R(s)是完全不同的数量；R(s)是在s中的“短期”奖励，而U(s)是

从s以后的“长期”总奖励。

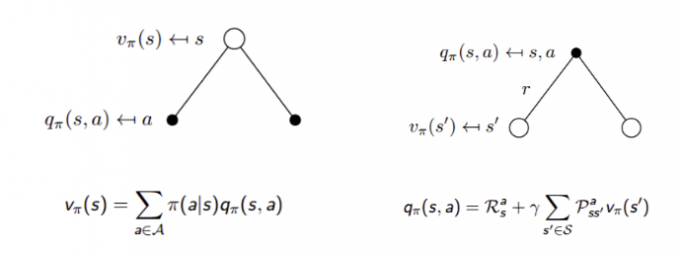
效用函数U(s)允许代理通过使用第16章中的最大预期效用原则来选择动作，即选择能够使后续状态的预期效用最大化的动作：



奖励函数可以描述为：在执行策略 (pi) 时获得的奖励等于执行该状态下所有行为的概率与对应行为产生的即时奖励的乘积的和。

[](https://imgchr.com/i/YW7tBD)

我们知道策略就是用来描述各个不同状态下执行各个不同行为的概率，而状态价值是遵循当前策略时所获得的收获的期望，即状态 s 的价值体现为在该状态下遵循某一策略而采取所有可能行为的价值按行为发生概率的乘积求和。可参照下图理解

[](https://imgchr.com/i/YfSnYT)

对于任何MDP，下面几点成立：

1.存在一个最优策略，比任何其他策略更好或至少相等；

2.所有的最优策略有相同的最优价值函数；

3.所有的最优策略具有相同的行为价值函数。

根据以上几点，我们可以最大化最优价值函数找到最优策略，对于状态比较多的环境，计算量太大，所以更加高效的方法是策略迭代（policy iteration）和值迭代（value iteration）

**四、值迭代算法（value iteration algorithm）**

通过值迭代的算法来计算最佳策略。值迭代的基本思路是先计算每个状态的效用U(state)，然后根据效用选择每个状态的最佳行动。

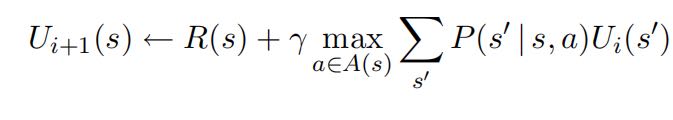
**4.1 Bellman Equation 贝尔曼方程**



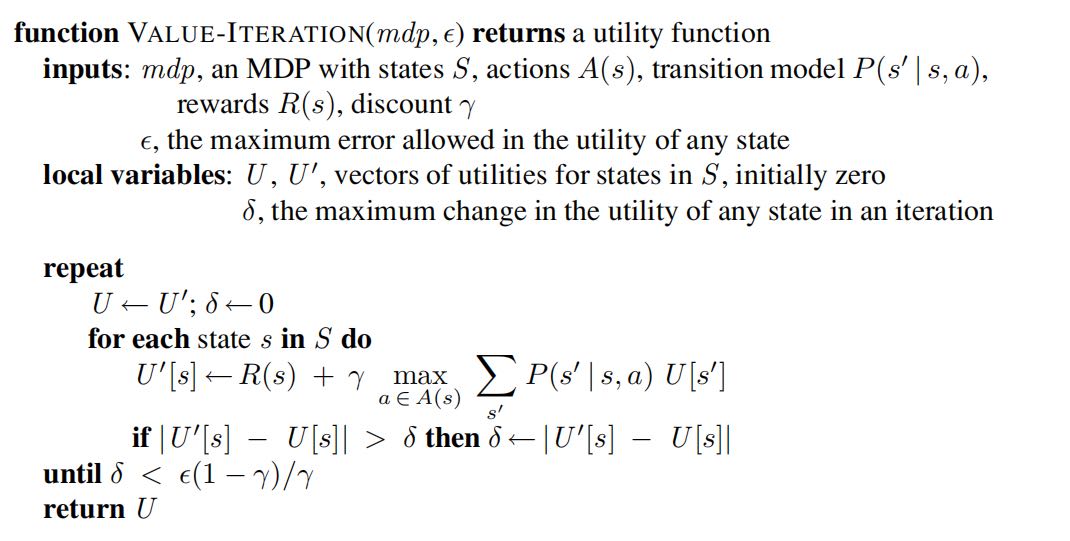
后续状态序列的期望效用是贝尔曼方程集的解。事实上，它们都是唯一的解决方案

**4.2 值迭代算法**

贝尔曼方程是求解MDPs的值迭代算法的基础。如果有n个可能的状态，那么就有n个贝尔曼方程，每个状态都有一个。那么方程式中包含n个未知数状态的效用。所以我们想求解这些联立方程来找到效用。而其中有一个问题：方程是非线性的，因为“最大”这个算子不是一个线性算子。线性方程组可以用线性代数技术快速求解，而非线性方程组的问题更严重。可以尝试的是迭代方法。我们从效用方程的任意初始值开始，计算方程的右侧，并将其插入左侧，从而从其邻居的实用程序中更新每个状态的效用方程。我们重复这个过程，直到我们达到一个平衡点。其中i(s)是第i次迭代时状态s的效用值。这个迭代步骤被称为贝尔曼更新。



其算法伪代码如下图所示：



值迭代最终收敛到贝尔曼方程的一组唯一的解。这是因为贝尔曼更新是在效用向量空间上的一个γ因子的收缩。（练习17.6为证明这一主张提供了一些指导。）因此，从一般的收缩性质来看，当γ<1时，值迭代总是收敛于贝尔曼方程的唯一解。

所以我们就可以通过反复迭代贝尔曼方程找到所需要的最优策略解。

**五、策略迭代（Policy Iteration）**

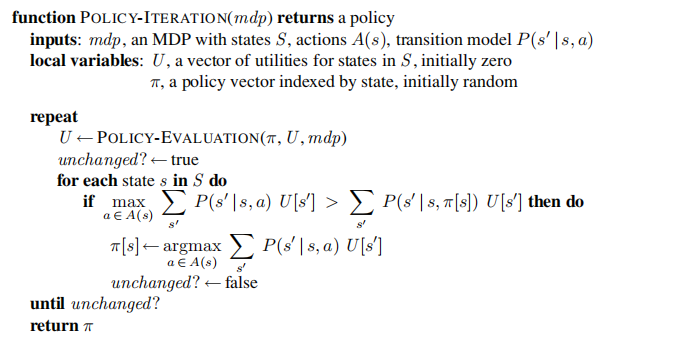
在值迭代中，即使效用函数估计不准确，也有可能得到一个最优策略。如果一个行动明显比其他行动好，那么所涉及国家的公用事业的确切规模就不一定精确。这样我们提出了一种寻找最佳政策的另一种方法-策略迭代。

策略迭代算法是选择一个初始策略，然后计算在该策略下每一个状态的效用。然后通过下一个状态的效用来更新在每一个状态下的策略，直到策略达到稳定。在给定策略下决定效用的步骤被称之为value determination，主要包括两个步骤：

* Evaluate（政策评估）：通过给定的策略π计算效用Ui = Uπi
* Improve（策略改进）：基于上一步的效用计算下一轮的政策，即贪心策略

如果我们有一个策略π，我们可以用策略估计出它的状态价值函数Uπi, 然后根据策略改进提取出更好的策略π’，接着再计算Uπ’i， 然后再获得更好的策略π''，直到相关满足相关终止条件。

策略迭代算法伪代码如下：

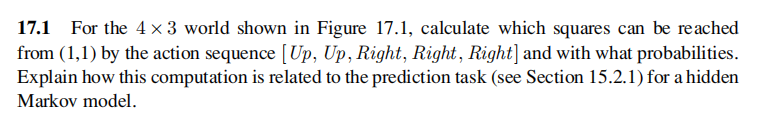


**六、总结**

到目前为止，上述两种算法需要一次更新所有状态的效用或策略。事实证明，这并不是严格必要的。事实上，在每次迭代中，我们可以选择状态的任何子集，并对该子集应用任何更新（策略改进或简化值迭代）。这种非常通用的算法被称为异步算法

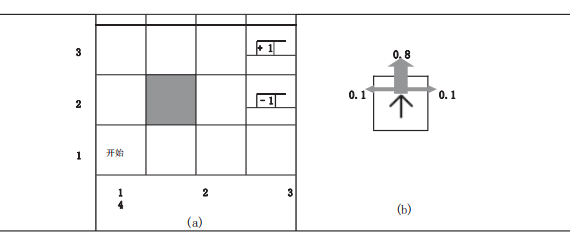
策略迭代在初始策略和初始效用函数的一定条件下，保证异步策略迭代收敛到最优策略。选择任何工作状态的自由意味着我们可以设计出更有效的启发式算法——例如，专注于更新一个好的策略可能会达到的状态值的算法。这在现实生活中很有意义：如果一个 人没有打算跌落悬崖，那么他就不应该花时间担心结果状态的确切价值。

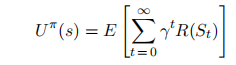
七、习题解答



解：

Figure1如下图：



由公式，其中γ=1

Uh([s0, s1, s2,...]) = R(s0) + γR(s1) + γ2R(s2) + ··· ,

可得：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | up | up | right | right | right |
| （1,1） | 1 | 0.1 | 0.02 | 0.026 | 0.0282 | 0.02462 |
| （1,2） |  | 0.8 | 0.24 | 0.258 | 0.2178 | 0.18054 |
| （1,3） |  |  | 0.64 | 0.088 | 0.0346 | 0.02524 |
| （2,1） |  | 0.1 | 0.09 | 0.034 | 0.0276 | 0.02824 |
| （2,3） |  |  |  | 0.512 | 0.1728 | 0.06224 |
| （3,1） |  |  | 0.01 | 0.073 | 0.0346 | 0.02627 |
| （3,2） |  |  |  | 0.001 | 0.0073 | 0.04443 |
| （3,3） |  |  |  |  | 0.4097 | 0.17994 |
| （4,1） |  |  |  | 0.008 | 0.0656 | 0.08672 |
| （4,2） |  |  |  |  | 0.0016 | 0.01400 |
| （4,3） |  |  |  |  |  | 0.32776 |

隐马尔可夫模型中的投影包括将占用概率向量乘以转移矩阵。在这个章节里，唯一的区别是，对于每个动作都有一个不同的转移矩阵。