强化学习基础篇（五）动态规划之策略迭代（1）

1、如何改善策略（How to improve a policy）

上节中我们讨论了如何使用贝尔曼期望方程进行策略估计,并没有对策略进行改进，而如果我们要解决控制问题，而不是预测问题的话，对策略进行改进是必要的，我们希望去找到某个问题的最优策略。其基本思想如下所示：

- 第一步：在一个给定的策略下迭代更新价值函数：

$$v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \ldots | S_t = s]$$

- 第二步：在当前策略基础上，根据v_π贪婪地选取行为，使得后继状态价值增加最多：

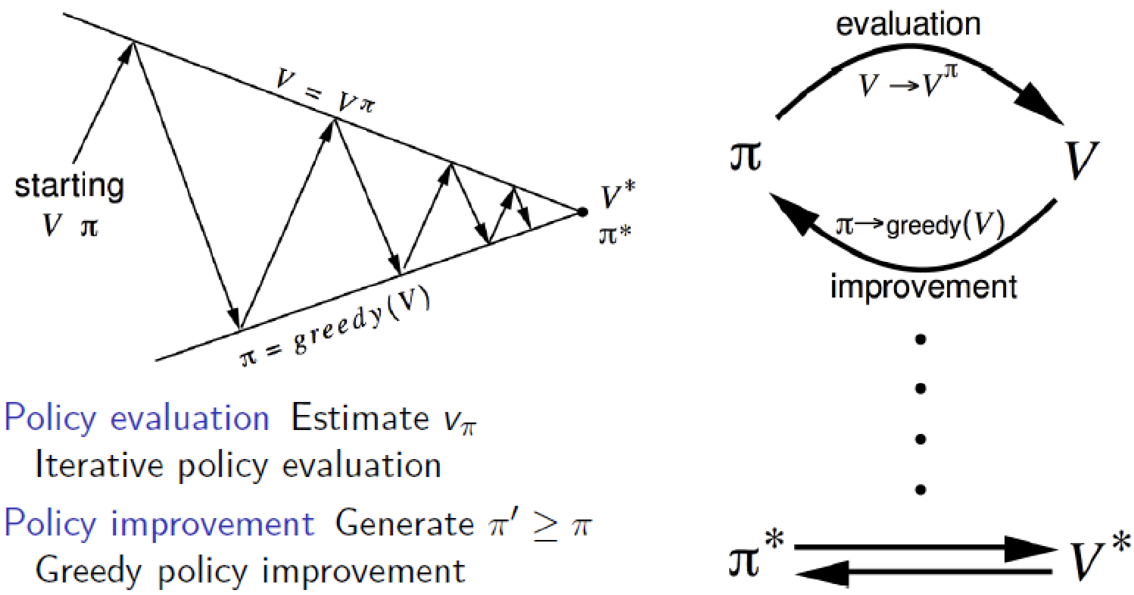
$$\pi' = greedy(v_\pi)$$

对于较小的格子世界（GridWorld）问题，基于给定策略的价值迭代最终收敛得到的策略就是最优策略，即 $\pi' = \pi^*$；

但是通常来说，我们需要更多的估计（evaluation/改进（improvement）迭代（即我们给出一个初始策略，估计其值函数至接近真实值，然后利用贪婪方法得到改进的策略，接着对改进后的策略进行估计，如此反复）。尽管如此，我们的策略迭代方法总能收敛到最优策略π^*。

2、策略迭代（Policy Iteration）

策略迭代的过程可以如下所示：



分为策略评估（Policy evaluation）和策略改进（Policy improvement）两个步骤。

- 策略评估（Policy evaluation）：根据策略π迭代式地计算值函数v_π。
- 策略改进（Policy improvement）：使用贪婪策略不断提升策略，使得$\pi' \geq \pi$。

细节描述如下：

a、我们随机初始化一个值V以及策略π

b、通过策略评估求得当前策略下的值函数$V = V^{\pi}$

c、使用贪婪策略提升策略，$\pi = greedy(v_{\pi})$

d、基于新的策略π计算值函数$V = V^{\pi}$，使得V函数与新策略一致

……

反复执行上面的过程，最终得到最优策略π^* 和最优状态价值函数V^*。

这个过程本质上就是使用当前策略产生新的样本，然后使用新的样本更好的估计策略的价值，然后利用策略的价值更新策略，然后不断反复。理论可以证明最终策略将收敛到最优。

3、策略迭代在杰克租车问题（Jack's Car Rental）中的示例

3.1、问题描述：

Jack有两个租车点（1号租车点和2号租车点）提供汽车租赁，由于不同的店车辆租赁的市场条件不一样，为了能够实现利润最大化，每天夜里，Jack可以在两个租车点间进行车辆调配，以便第二天能最大限度的满足两处汽车租赁服务。

问题即寻找最优的车辆调配策略。

3.2、已知条件：

状态空间：1号租车点和2号租车点，每个地点最多20辆车供租赁

行为空间：每天下班后最多转移5辆车从一个租车点到另一个租车点

即时奖励：Jack每租出去一辆车可以获利10美金，但必须是有车可租的情况，不考虑在两地转移车辆的支出

转移概率：租出去的车辆数量（n）和归还的车辆数量（n）是随机的，但是服从泊松分布$\frac{\lambda^n}{n!}e^{-\lambda}$。

- 对于1号租车点：向外出租服从$\lambda = 3$的泊松分布，回收也服从$\lambda = 3$的泊松分布
- 对于2号租车点：向外出租服从$\lambda = 4$的泊松分布，回收服从$\lambda = 2$的泊松分布

折扣因子γ：$\gamma = 0.9$

3.3、问题分析：

- 每个租车点最多20辆车，那么状态数量就是$21 * 21 = 441$个。
- 最多调配5辆车，即动作集合为:
  $A = \{(-5,5),(-4,4),(-3,3),(-2,2),(-1,1),(0,0),(1,-1),(2,-2),(3,-3),(4,-4),(5,-5)\}$
  其中么个动作元素表示为（1号租车点出入车辆，2号租车点出入车辆），正负号分别表示“入”和“出”。

3.4、求解过程：

“1号租车点有10辆车”收益分析：

考虑状态“1号租车点有10辆车”的未来可能获得收益，需要分析在保有10辆车的情况下的租车（Rent）与回收（Return）的行为。计算该状态收益的过程实际上是，另外一个动作策略符合泊松分布的马尔可夫决策过程。

将1天内可能发生的Rent与Return行为记录为[#Rent #Return]，其中“#Rent”表示一天内租出的车辆数，“#Return”表示一天内回收的车辆数，设定这两个指标皆不能超过20。

假设早上，1号租车点里有10辆车，那么在傍晚清点的时候，可能保有的车辆数为0~20辆。如果傍晚关门歇业时还剩0辆车，那么这一天的租收行为$A_{rent,return}$可以是：

$$A_{rent,return} = \begin{bmatrix} 10 & 0 \\ 11 & 1 \\ 12 & 2 \\ \dots & \dots \\ 20 & 10 \end{bmatrix}$$

Rent与Return是相互独立的事件且皆服从泊松分布，所以要计算某个行为出现的概率直接将$P(A_{rent})$与$P(A_{return})$相乘。

但这里要计算的是条件概率，即为$P(A_{rent,return}|S''=0)$，所以还需要再与傍晚清点时还剩0辆车的概率$P(S''=0)$相除。

各个租收行为所获得的收益是以租出去的车辆数为准，所以当傍晚还剩0辆车时，这一天的收益期望可以写为：

$$R(S'=10|S''=0) = 10 \begin{bmatrix} \frac{P(A_{rent}=10)P(A_{return}=0)}{P(S''=0)} \\ \frac{P(A_{rent}=11)P(A_{return}=1)}{P(S''=0)} \\ \dots \\ \frac{P(A_{rent}=20)P(A_{return}=10)}{P(S''=0)} \end{bmatrix}^T \begin{bmatrix} 10 \\ 11 \\ \dots \\ 20 \end{bmatrix}$$

其中，$P(S''=0)$也可以写为：

$$P(S''=0) = \sum P(A_{rent})P(A_{return})$$

在计算出矩阵$R(S'=10|S''=0,1,2,\dots 20)$后，再进行加权平均，即可得到状态“1号租车点有10辆车”的奖励期望$R(S'=10)$：

$$R(S'=10) = P(S''=0,1,2,\dots 20)R^T(S'=10|S''=0,1,2,\dots,20)$$

两个租车点，所有的状态按上述方法计算后，即可得出两个租车点的奖励矩阵$|R_1(S'),R_2(S')|$。

在计算出奖励矩阵后，这个问题就变成了bandit问题的变种，bandit问题是一个动作固定对应一个未来的状态，而这里虽然也是这样，不过所对应的状态却要以当前状态为基础进行计算得出，还是有些不同，所以称为bandit问题的一个变种。

3.5、算法流程：

策略改进（Policy improvement）是将已有的动作选择策略$\pi(S,A)$和值函数V矩阵带入与最优值进行比较，从而将 $\pi(S,A)$更新为最优。

策略改进（Policy improvement）：

a、初始化 Q 矩阵，将计算好的V 矩阵与策略$\pi(S,A)$带入状态循环中（每一个状态计算一遍）
b、将当前状态转变为1号与2号租车点的保有车辆数[#Car1 #Car2]
c、带入动作集合计算找出可能的未来状态S'与可执行的动作Possible Action
d、计算Q矩阵：

$$Q(S,PossibleAction) = R_1(S') + R_2(S') - 2Cost(PossibleAction) + \gamma V(s')$$

e、用策略 $\pi(S,A)$ 与Q_{max}所在的动作进行比较，若是不符则令一个flag：Policy_Stable = False

策略评估（Policy evaluation）+ 策略改进（Policy improvement）

a、计算奖励矩阵，初始化Q矩阵与V矩阵
b、判断Policy_Stable是否为False，如为True则输出结果$\pi(S,A)$，如为False则进入迭代循环过程。
c、令Policy_Stable = True
d、执行Policy-Evaluation算法
e、执行Policy-Improvement算法，得到Policy_Stable的结果，返回第b步

3.6、结果：

下面这幅图表示出了策略迭代的过程，直到第5次迭代，动作策略最终稳定为最优策略。横轴表示2号租车点的车辆保有量，纵轴表示1号租车点的车辆保有量，正负号分别表示从1号调出车辆到2号，从2号调出车辆到1号。