强化学习2-policy iteration

前边的笔记介绍了Bellman方程，提到求解Bellman方程一般采用policy iteration，今天就介绍怎么用迭代法求解Bellman方程的最优解。

Bellman Optimal Equation,BOE

最优策略，顾名思义，就是在任何state下的state value都大于其他策略的state value。

> **Definition**
> A policy π^* is optimal if $v_{\pi^*}(s) \geq v_\pi(s)$ for all s and for any other policy π.

最优策略条件下的bellman方程：

$$v(s) = \max_\pi \sum_{a \in A} \pi(a|s)[\sum_{s'} \sum_r p(s', r|s, a)[r + \gamma v(s')]] = \max_\pi \sum_{a \in A} \pi(a|s) q(s, a), s \in \mathcal{S}$$

上式存在两个未知数v和 π 。考虑到有 $\sum_a \pi(a|s) = 1$ ，可以先固定v，先对 π 求max，接下来就只用求解v了：

$$v(s) = \max_{a \in A} \ q(s, a)$$

接下来的对v的求解和证明参考论文：**Mathematical Foundations of Reinforcement Learning**

首先定义压缩映射的概念，

> 如果存在$\alpha \in (0, 1)$，使得函数g对$\forall x_1, x_2 \in \mathrm{dom}\ g$满足
>
> $$\|g(x_1) - g(x_2)\| \leq \alpha \|x_1 - x_2\|$$
>
> 则我们称g为一个压缩映射，称常数α为压缩常数.

而BOE的矩阵形式 $v = \max_\pi (r_\pi + \gamma P_\pi v)$ 就是一个压缩映射，对于压缩映射，我们就有：

设 g是 $[a,b]$ 上的一个压缩映射，则

1. g在$[a,b]$ 中**存在唯一**的不动点 $\xi=g(\xi)$；
2. 由**任何初始值** $x_0 \in [a,b]$ 和递推公式

$$x_{n+1}=g(x_n), \quad n\in N^*$$

生成的数列 $\{x_n\}$ 一定收敛于 ξ.

也就是说对于BOE，存在且唯一存在一个最优的state value v^* ，可以通过迭代的方式 $v_{k+1}=\max_\pi(r_\pi+\gamma P_\pi v_k)$ 收敛到 v^* ，无论给定任何的初始状态 v_0 ，其收敛速度由 γ 决定。

在每一次迭代过程中（下一章详细介绍具体的迭代过程），采用上边提到的 $v_{k+1}(s)=\max_{a\in A} q_k(s,a)$ 。该公式右边也就是采用贪心策略，对每个状态采取能获得最大state value的行动

$$\pi(a|s)=\begin{cases}1 & a=a^*,\\ 0 & a\neq a^*.\end{cases}$$

其中 $a^*=\arg\max_a q_k(s,a)$ 。求解出 v^* 之后自然的就能得到最优的 π^* ,可以证明这样得到的 π^* 就是最优策略。

$$\pi^*=\arg\max_\pi(r_\pi+\gamma P_\pi v^*)$$

Policy iteration

给定一个初始策略 π_0 , 分为两个步骤：

1. 策略评估：通过迭代法不断更新当前策略下的state value，直到state value的前后差值小于一个参数（收敛到一个定值）：

$$v_\pi=r_\pi+\gamma P_\pi v_\pi$$

> 注意这里的策略是固定的，由于采用贪心策略，所以在每个状态下采取的动作也是固定的，即 $\pi(a^*|s)=1$,而其他的动作都为0。因此在下面算法中省略了 $\sum_{a\in A}\pi(a|s)$ 。

2. 策略优化：基于更新后的 v_π 更新策略：

$$\pi=\arg\max_\pi(r_\pi+\gamma P_\pi v_\pi)$$

具体到每一个状态：

$$\pi(s)=\arg\max_\pi\sum_a\pi(a|s)\cdot\sum_{s'}\sum_r p(s',r|s,a)[r+\gamma v_\pi(s')]=\arg\max_\pi\sum_a\pi(a|s)\cdot q_\pi(s,a)$$

$$a^*=\arg\max_a q_k(s,a)$$

$$\pi(a|s) = \begin{cases} 1 & a = a^*, \\ 0 & a \neq a^*. \end{cases}$$

整个流程直到策略不再改变停止，伪代码如下：

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

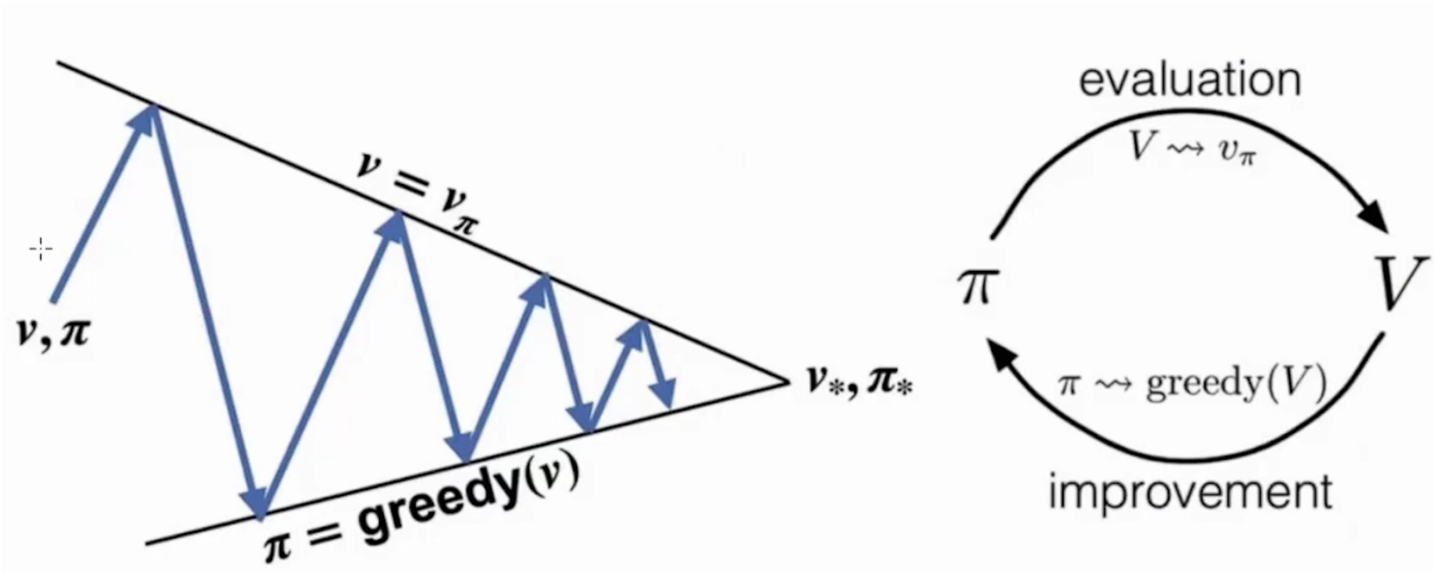
Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

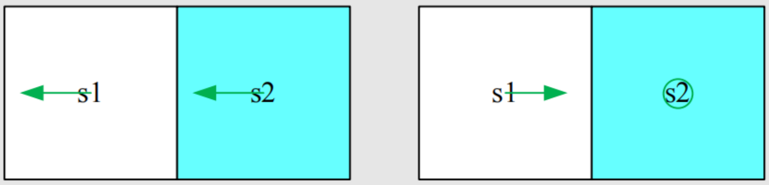
2. Policy Evaluation
   Loop:
       $\Delta \leftarrow 0$
       Loop for each $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\left[r + \gamma V(s')\right]$
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
       old-action $\leftarrow \pi(s)$
       $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
       If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

v和 π 的更新是交替进行的，其中v的更新由于要进行多次迭代，速度会比较慢，可以通过适当放宽收敛标准，减少迭代次数；而 π 的更新直接贪心，速度比较快。



举一个栗子，引用自**西湖大学赵世钰老师的【强化学习的数学原理】：**



(a) initial policy (b) optimal policy

▷ The reward setting is $r_{\text{boundary}} = -1$ and $r_{\text{target}} = 1$. The discount rate is $\gamma = 0.9$.

▷ Actions: a_ℓ, a_0, a_r represent go left, stay unchanged, and go right.

▷ Aim: use policy iteration to find out the optimal policy.

▷ Iteration $k = 0$: Step 1: policy evaluation

π_0 is selected as the policy in Figure (a). The Bellman equation is

$$v_{\pi_0}(s_1) = -1 + \gamma v_{\pi_0}(s_1),$$

$$v_{\pi_0}(s_2) = 0 + \gamma v_{\pi_0}(s_1).$$

- Solve the equations directly:

$$v_{\pi_0}(s_1) = -10, \quad v_{\pi_0}(s_2) = -9.$$

- Solve the equations iteratively. Select the initial guess as $v_{\pi_0}^{(0)}(s_1) = v_{\pi_0}^{(0)}(s_2) = 0$:

$$\begin{cases} v_{\pi_0}^{(1)}(s_1) = -1 + \gamma v_{\pi_0}^{(0)}(s_1) = -1, \\ v_{\pi_0}^{(1)}(s_2) = 0 + \gamma v_{\pi_0}^{(0)}(s_1) = 0, \end{cases}$$

$$\begin{cases} v_{\pi_0}^{(2)}(s_1) = -1 + \gamma v_{\pi_0}^{(1)}(s_1) = -1.9, \\ v_{\pi_0}^{(2)}(s_2) = 0 + \gamma v_{\pi_0}^{(1)}(s_1) = -0.9, \end{cases}$$

$$\begin{cases} v_{\pi_0}^{(3)}(s_1) = -1 + \gamma v_{\pi_0}^{(2)}(s_1) = -2.71, \\ v_{\pi_0}^{(3)}(s_2) = 0 + \gamma v_{\pi_0}^{(2)}(s_1) = -1.71, \end{cases}$$

▷ Iteration $k = 0$: Step 2: policy improvement

The expression of $q_{\pi_k}(s, a)$:

$q_{\pi_k}(s,a)$	a_ℓ	a_0	a_r
s_1	$-1 + \gamma v_{\pi_k}(s_1)$	$0 + \gamma v_{\pi_k}(s_1)$	$1 + \gamma v_{\pi_k}(s_2)$
s_2	$0 + \gamma v_{\pi_k}(s_1)$	$1 + \gamma v_{\pi_k}(s_2)$	$-1 + \gamma v_{\pi_k}(s_2)$

Substituting $v_{\pi_0}(s_1) = -10, v_{\pi_0}(s_2) = -9$ and $\gamma = 0.9$ gives

$q_{\pi_0}(s,a)$	a_ℓ	a_0	a_r
s_1	-10	-9	-7.1
s_2	-9	-7.1	-9.1

By seeking the greatest value of q_{π_0}, the improved policy is:

$$\pi_1(a_r|s_1) = 1, \quad \pi_1(a_0|s_2) = 1.$$

This policy is optimal after one iteration! In your programming, should continue until the stopping criterion is satisfied.