

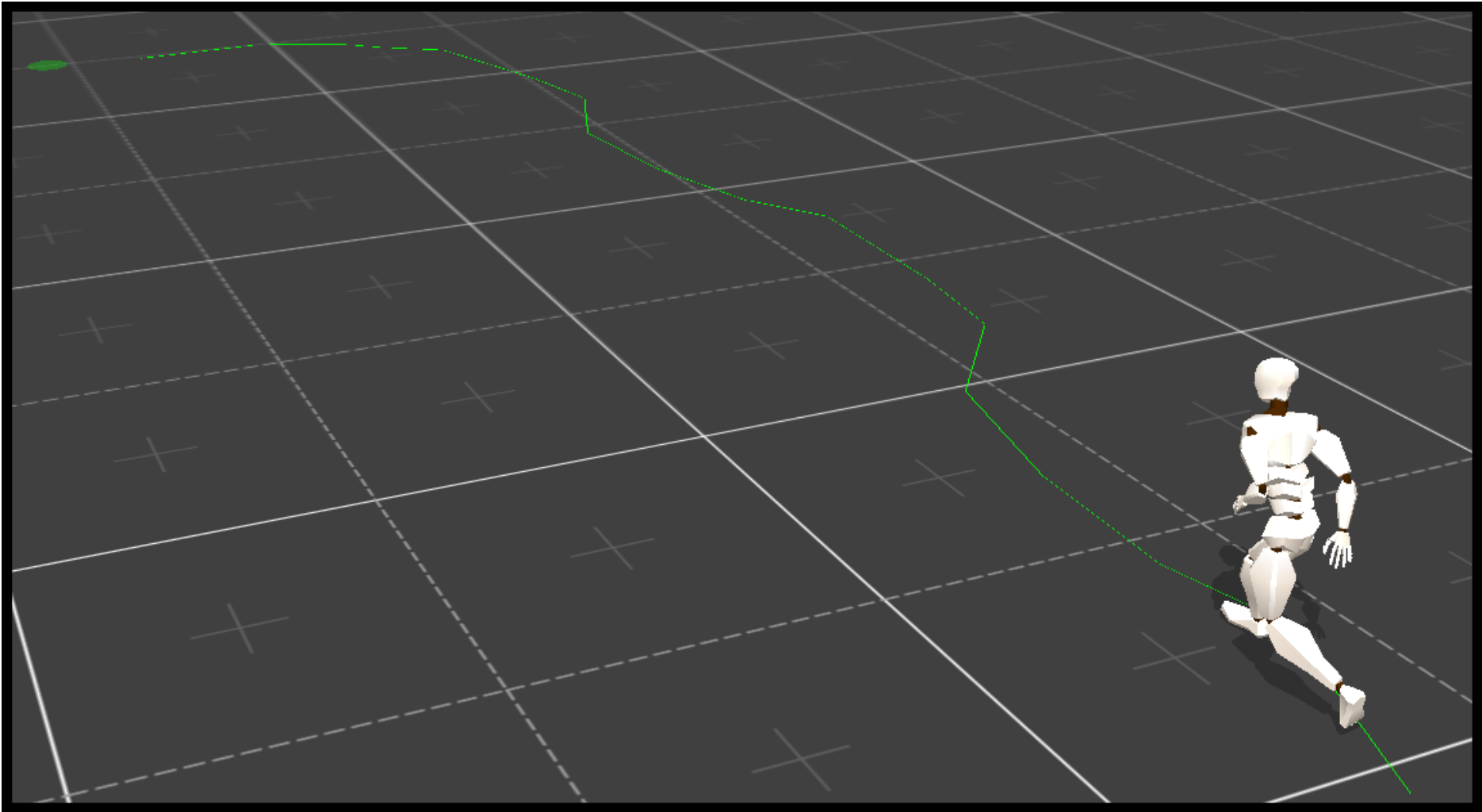
强化学习系列之一:马尔科夫决策过程

发表于2016年4月4日由lll

文章目录 [\[隐藏\]](#)

- 1. 马尔科夫决策过程
 - 2. 策略和价值
 - 3. 最优策略存在性和贝尔曼等式
- [强化学习系列系列文章](#)

机器学习一共有三个分支，有监督学习、无监督学习和强化学习。强化学习是系统从环境学习以使得奖励最大的机器学习。强化学习和有监督学习不同在于教师信号。强化学习的教师信号是动作的奖励，有监督学习的教师信号是正确的动作。



我最近在学习强化学习。为了整理学习心得，开始写这个强化学习系列博客。由于学识浅薄，内容难免有些错误，还望各路大神不吝赐教。

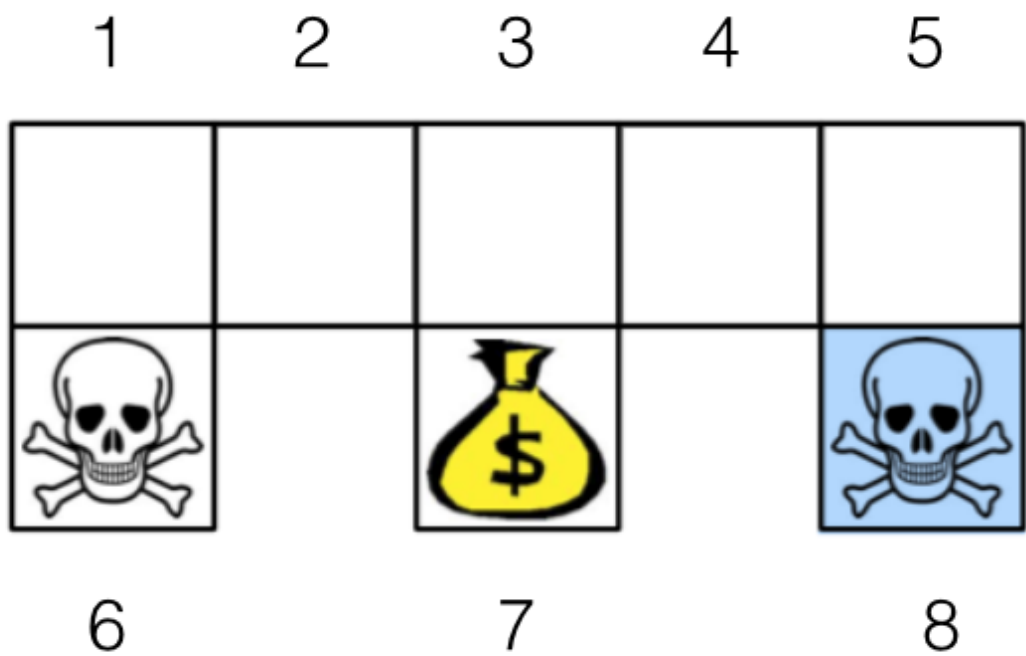
1. 马尔科夫决策过程

要说强化学习，就必须说说马尔科夫决策过程 (Markov Decision Processes, MDP)。马尔可夫决策过程是基于马尔可夫过程理论的随机动态系统的决策过程，其分五个部分：

- 1. S 表示状态集 (states);
- 2. A 表示动作集 (Action);
- 3. $P_{s,a}^{s'}$ 表示状态 s 下采取动作 a 之后转移到 s' 状态的概率;
- 4. $R_{s,a}$ 表示状态 s 下采取动作 a 获得的奖励;
- 5. γ 是衰减因子。

和一般的马尔科夫过程不同，马尔科夫决策过程考虑了动作，即系统下个状态不仅和当前的状态有关，也和当前采取的动作有关。还是举下棋的例子，当我们在某个局面（状态s）走了一步（动作a）。这时对手的选择（导致下个状态s'）我们是不能确定的，但是他的选择只和s和a有关，而不用考虑更早之前的状态和动作，即s'是根据s和a随机生成的。

下图是一个机器人从任意一个状态出发寻找金币的例子。找到金币则获得奖励1，碰到海盗则损失1。找到金币或者碰到海盗则机器人停止。



我们可以把这个问题建模成马尔科夫决策过程。图中不同位置为状态，因此 $S = \{1,...,8\}$ 。机器人采取动作是向东南西北四个方向走，因此 $A=\{'n','e','s','w'\}$ 。转移概率方面，当机器人碰到墙壁，则会停在原来的位置；当机器人找到金币时获得奖励1，当碰到海盗则损失1，其他情况不奖励也不惩罚。因此除了 $R_{1,s} = -1$, $R_{3,s} = 1$, $R_{5,s} = -1$ 之外，其他情况 $R_{*,*} = 0$ 。衰减因子 γ 在后面介绍。写成代码下面所示。

```
class Mdp:

    def __init__(self):

        self.states          = [1,2,3,4,5,6,7,8] # 0 indicates end
        self.terminal_states = dict()
        self.terminal_states[6] = 1
        self.terminal_states[7] = 1
        self.terminal_states[8] = 1

        self.actions         = ['n','e','s','w']

        self.rewards         = dict();
        self.rewards['1_s'] = -1.0
        self.rewards['3_s'] = 1.0
        self.rewards['5_s'] = -1.0

        self.t               = dict();
        self.t['1_s']        = 6
        self.t['1_e']         = 2
        self.t['2_w']         = 1
        self.t['2_e']         = 3
        self.t['3_s']         = 7
        self.t['3_w']         = 2
        self.t['3_e']         = 4
        self.t['4_w']         = 3
        self.t['4_e']         = 5
        self.t['5_s']         = 8
        self.t['5_w']         = 4
```

```

self.gamma = 0.8

def transform(self, state, action): ##return is_terminal, state, reward
    if state in self.terminal_states:
        return True, state, 0

    key = '%d_%s'%(state, action);
    if key in self.t:
        next_state = self.t[key];
    else:
        next_state = state

    is_terminal = False
    if next_state in self.terminal_states:
        is_terminal = True

    if key not in self.rewards:
        r = 0.0
    else:
        r = self.rewards[key];

    return is_terminal, next_state, r;

```

马尔科夫决策过程是强化学习的理论基础。不管我们是将强化学习应用于五子棋游戏、星际争霸还是机器人行走，我们都假设背后存在了一个马尔科夫决策过程。只不过有的时候我们知道马尔科夫决策过程所有信息(状态集合，动作集合，转移概率和奖励)，有的时候我们只知道部分信息(状态集合和动作集合)，还有些时候马尔科夫决策过程的信息太大无法全部存储(比如围棋的状态集合总数为 $3^{19 \times 19}$)。强化学习算法按照上述不同情况可以分为两种: 基于模型 (Model-based) 和非基于模型 (Model-free)。基于模型的强化学习算法是知道并可以存储所有马尔科夫决策过程信息，非基于模型的强化学习算法则需要自己探索未知的马尔科夫过程。

2. 策略和价值

强化学习技术是要学习一个策略 (Policy)。这个策略其实是一个函数，输入当前的状态 s ，输出采取动作 a 的概率 $\pi(s, a)$ 。有监督学习希望分类器正确地对实例分类，那么强化学习的目标是什么呢？强化学习希望把策略训练成什么样呢？假设我们的系统在一个状态 s 中，我们不会选择当前奖励 $R_{s,a}$ 最大的动作 a 。因此这个动作可能导致系统进入死胡同，即系统之后会受到很大的处罚。为了避免这种情况，策略要考虑到后续的影响。因此我们最大化递减奖励的期望




$$E_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k R_k \right] = E_{\pi} [R_0 + \gamma R_1 + \dots] \quad (1)$$

其中 γ 是马尔科夫决策过程的第五个部分：衰减因子。 γ 用于平衡当前奖励和远期奖励的重要性，也是用来避免计算结果无穷。 R_k 是系统在当前策略下第 k 步之后获得的奖励。这种目标既考虑了当前奖励又考虑了远期奖励，避免了下一个状态是死胡同的问题。

根据上面的目标，人们提出了价值的概念。一个策略下的一个状态的价值定义：这个状态下，按照这个策略，系统能够获得的递减奖励期望。

$$v(s) = E_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k R_k \right] \quad (2)$$

上面机器人找金币的例子中，设定衰减因子 γ 等于 0.5。如果策略是一直往西走碰壁之后立马向南，那么状态 2 的价值 $v(2) = 0 + 0.5 * -1.0 = -0.5$ 。如果策略是随机一个方向，那么所有状态的价值如下图所示。

-0.335	-0.008	0.283	-0.008	-0.335
				

后来人们进一步扩展了价值的概念，将价值扩展到状态-动作对上。一个状态-动作对的价值定义如下所示。

$$q(s, a) = R_{s,a} + E_{\pi} \left[\sum_{k=1}^{\infty} \gamma^k R_k \right] \quad (3)$$

3. 最优策略存在性和贝尔曼等式

根据上面的介绍，我们发现强化学习的目标是找到一个策略 π ，使得这个策略下每个状态的价值最大。但是这里有一个问题啊。对于两个策略，有可能出现：策略 π_1 状态 a 的价值大于策略 π_2 状态 b 的价值，但策略 π_2 状态 c 的价值大于策略 π_1 状态 d 的价值。因此我们不确定，是否存在一个策略 π^* 的所有状态价值大等于其他策略的状态价值。如果不存在这么一个策略，我们的目标是迷茫的。

但是万幸啊，下面的定理保证了这么一个策略存在。这么一个所有状态价值大等于其他所有的状态价值，我们可以称之为最优策略。强化学习算法的目标就是找到最优策略。

Define a partial ordering over policies

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

Theorem

For any Markov Decision Process

- There exists an optimal policy π_* that is better than or equal to all other policies, $\pi_* \geq \pi, \forall \pi$
- All optimal policies achieve the optimal value function, $v_{\pi_*}(s) = v_*(s)$
- All optimal policies achieve the optimal action-value function, $q_{\pi_*}(s, a) = q_*(s, a)$

另外一个重要的点是贝尔曼等式。贝尔曼等式表明了当前状态的值函数与下个状态的值函数的关系，具有很简明的形式。

$$\begin{aligned}
 & v(s) \\
 &= \sum_{a \in A} \pi(s, a) q(s, a) \\
 &= \sum_{a \in A} \pi(s, a) (R_{s,a} + \gamma \sum_{s' \in S} T_{s,a}^{s'} v(s')) \\
 & \quad q(s, a) \\
 &= R_{s,a} + \gamma \sum_{s' \in S} T_{s,a}^{s'} v(s') \\
 &= R_{s,a} + \gamma \sum_{s' \in S} T_{s,a}^{s'} \sum_{a' \in A} \pi(s', a') q(s', a') \quad (4)
 \end{aligned}$$

贝尔曼等式在强化学习中有很重要作用，这个我们后面会了解到。

文中所涉及代码可以在 [GitHub](#) 上找到。文章结尾欢迎关注我的公众号 AlgorithmDog，每周日的更新就会有提

醒哦~



欢迎关注
公众号讲述机器学习和系统研发的轶事，
希望讲得有趣，每周日更新~
扫描二维码即可关注。您，不关注下么？

强化学习系列文章

- 强化学习系列之一:马尔科夫决策过程
- [强化学习系列之二:模型相关的强化学习](#)
- [强化学习系列之三:模型无关的策略评价](#)
- [强化学习系列之四:模型无关的策略学习](#)
- [强化学习系列之五:价值函数近似](#)
- [强化学习系列之六:策略梯度](#)
- [强化学习系列之九:Deep Q Network \(DQN\)](#)

此条目发表在[强化学习](#), [算法荟萃](#)分类目录，贴了[强化学习](#)标签。将[固定链接](#)加入收藏夹。

《强化学习系列之一:马尔科夫决策过程》有 36 条评论



[Jeff](#) 说:
2016年4月11日下午7:59

先占个坑，最近导师每周都讲RL，所以也会持续关注的。加油，加油~
[回复](#)



[上微博的猫V](#) 说:
2016年4月11日下午10:28

感谢关注~__~。如发现哪里有错误，欢迎指出。
[回复](#)



[杨超](#) 说:
2016年5月11日下午2:49

博主在写Python 的时候习惯性加分号吗？？不是好习惯哟
[回复](#)



[上微博的猫V](#) 说:
2016年5月11日下午8:55

是的啊。因为我不知道除了不geek，加分号有什么坏处。~____~。
[回复](#)



[杨宇轩](#) 说:

2016年4月12日上午11:12

”如果策略是一直往东走碰壁之后立马向南，那么状态 2 的价值 $v(2)=0+0.5 * -1.0=-0.5$ “
这个的初始状态是机器人位于位置5吧？

回复



上微博的猫V说:

2016年4月12日下午7:15

一个状态的价值是值，机器人从这个位置出发能够获得奖励的期望。即使机器人从2出发，也可以执行“一直往东走碰壁之后立马向南”的策略~。

回复



杨宇轩 说:

2016年4月12日下午10:31

恩，但是若干从2出发话， $v(2)=0+0.5*0+0.5^2*0+0.5^3*0+0.5^4*(-1)$
应该是这样吧？

回复



上微博的猫V说:

2016年4月12日下午11:12

明白了。我写错了，策略应该是“一直往西走碰壁之后立马向南”。感谢指出。

回复



whatever lynn 说:

2016年5月17日下午4:13

还是错了吧， $v(2)=0+0.5 * 0+0.5^2* (-1.0) =-0.25$



上微博的猫V说:

2016年5月17日下午6:49

应该没错吧。 $v(2)=0(2->1的奖励)+0.5 * (-1.0)(1->6的奖励)=-0.5$



游 说:

2016年6月8日上午8:52

我有一點想不明白 為何 1->6的衰减因子 不是 0.5^2 而是 0.5呢？



QianBtr 说:

2016年8月10日下午4:24

1->6的时候K就为1了，当然是 0.5^1



海格力斯 说:

2016年10月26日上午11:49

应该是： $v(2)=0+0.5 * 0+0.5^2* (-1.0) =-0.25$
第一个0是状态2的即时回报
2->1,回报也是0，应该是 $0.5*0$
1->6,回报是-1，应该是 $0.5^2* (-1.0)$



海格力斯 说:

2016年10月26日上午11:51

在计算一个状态的价值时，自身的回报也是需要计算的



杨宇轩 说:
2016年4月29日下午9:37

有没有用强化学习搞机器人控制方面的人啊，求交流学习
[回复](#)



Luke 说:
2016年11月3日下午3:09

<https://research.googleblog.com/2016/10/how-robots-can-acquire-new-skills-from.html> 这个算是吧 参考论文里有详细的说如何用的深度强化学习
[回复](#)



Yi骑绝尘 说:
2016年5月2日上午7:26

值得学习哦
[回复](#)

Pingback引用通告：[强化学习系列之五:进入实际问题的关键——价值函数近似 | 神刀安全网](#)



肥魔仙 说:
2016年7月7日下午11:26

代码是什么语言？用 Matlab 或者干脆用伪代码吧。
[回复](#)



老董 说:
2016年7月10日下午1:03

应该是python
[回复](#)



上微博的猫V 说:
2016年7月16日下午8:52

确实是python
[回复](#)



老董 说:
2016年7月10日下午1:02

很不错的很不错的内容，最近正在看Reinforcement Learning: An Introduction,然后就发现您的博客了，不错，真不错。
[回复](#)



你这愚蠢的地球人 说:
2016年7月20日下午4:25

您也在看这个吗？我最近也在学这个，希望能和您多多交流
[回复](#)



上微博的猫V 说:
2016年7月23日下午5:17

感谢认可～ ^ _____ ^
[回复](#)



匿名 说:
2016年10月13日上午11:14

博主您好，我是科赛网（kesci.com）的编辑,科赛是一个专业的大数据竞赛平台。您的强化学习系列写得很棒，我们想转载到科赛的公众平台上，不知是否方便授权转载呢～谢谢！

[回复](#)



lili 说:
2016年10月13日下午11:00

恩恩，请注明链接哈～

[回复](#)



海格力斯 说:
2016年10月25日上午11:09

好文呀，但有个地方不明白：策略是随机一个方向时的状态价值是怎么计算的？比如状态1的-0.335

[回复](#)



海格力斯 说:
2016年10月25日下午4:44

看程序是每次随机选择方向前进，完成后计算这个路径对应的状态值函数，重复若干次，然后取平均值。在大量重复的情况下各路径的选择概率基本是均匀的，是不是可以用概率直接进行计算呢？比如状态3，有3个选择：左，右，下，概率都是1/3,然后乘以三条路径的状态值，最后求合？试算如下， $r=0.5$ 。向下： $\frac{1}{3} \times 1 = 0.333$ ，向左： $\frac{1}{3} \times (0 + 0.5 \times 0 + 0.5 \times 0.5 \times -1) = -0.083$ ，向右： $\frac{1}{3} \times (0 + 0.5 \times 0 + 0.5 \times 0.5 \times -1) = -0.083$ ，最后状态3的状态值为： $0.333 - 0.083 - 0.083 = 0.167$ ，不知道这样对不对？

[回复](#)



海格力斯 说:
2016年10月25日下午5:23

另外mdp_value.py中是不是不应当考虑不可能的方向？比如状态3是不可能往上走的。程序是否应当修改如下：

```
#!/bin/python
import numpy;
import random as ran;
ran.seed(0)
from mdp import *;

def random_pi():
    actions = ['n','w','e','s']
    r = int(ran.random() * 4)
    return actions[r]

def compute_random_pi_state_value():
    value = [ 0.0 for r in xrange(9)]
    num = 1000000

    for k in xrange(1,num):
        for i in xrange(1,6):
            mdp = Mdp();
            s = i;
            is_terminal = False
            gamma = 1.0
            v = 0.0
            while False == is_terminal:
                a = random_pi()
                is_terminal, stmp, r = mdp.transform(s, a)
```



```
if stmp!=s:
    v += gamma * r;
    gamma *= 0.5
    s = stmp

value[i] = (value[i] * (k-1) + v) / k

if k % 10000 == 0:
    print value

print value

compute_random_pi_state_value()

回复
```



海格力斯 说:
2016年10月25日下午5:42

哦，我说的这种应该不是随机策略，而是不碰壁并且不走回头路的策略，策略不同所以状态值也不同，是这样吗?
[回复](#)



lili 说:
2016年11月1日上午1:21

恩恩。状态价值是依赖于策略的，并不独立存在。这点可以用状态价值的定义中得到。状态价值应该用概率算的，我的程序只是用大量模拟得到概率而已～。
[回复](#)



lili 说:
2016年10月30日下午10:20

不好意思哈。平时没有登录博客后台，没有看到您的评论。您的评论的信息量有点多，等我详细看了之后再和您讨论哈～。感谢您的关注。
[回复](#)



lili 说:
2016年10月31日下午11:46

恩恩。状态价值是依赖于策略的，并不独立存在。这点可以用状态价值的定义中得到。状态价值应该用概率算的，我的程序只是用大量模拟得到概率而已～。
[回复](#)

Pingback引用通告：[新!!!\[强化学习系列之一:马尔科夫决策过程\] – Yizhen's blog](#)

Pingback引用通告：[\[强化学习系列之一:马尔科夫决策过程\] – Yizhen's blog](#)

Pingback引用通告：[机器学习 – 深度全栈](#)