第一章　强化学习-机器人导航

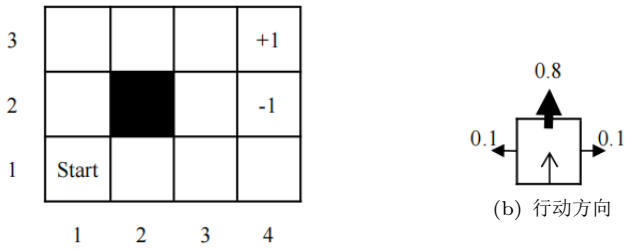
1.1　实验内容与任务

图1.1(a)是一个机器人导航问题的地图。机器人从起点Start出发，每一个时间点，它必须选择一个行动(上下左右)。在马尔可夫决策中实验中，机器人是根据环境模型中的转移矩阵 $P(x, a, x')$ 来进行价值函数和最优策略的计算，但是本次实验中，机器人并不知道这个转移矩阵。已知机器人行动之后，环境会告知机器人两件事情–新的实际位置以及到达新位置所得到的报酬。因此，如果机器人有一个策略$\pi(x)$，那么在与环境的交换中，机器人会具有这样一个数据序列：$(x_0, a_0, r_0, x_1, a_1, r_1, \ldots, a_{n-1}, r_{n-1}, x_n)$，其中$x_i, r_i$ 是环境告知的状态和该状态的报酬，$a_i = \pi(x_i)$是机器人自己的决策，x_0是Start, x_n 是一个终止状态。这个数据序列也称为一个样本路径。强化学习的任务是让机器人在环境中运行多次，得到多条样本路径，通过这些样本路径，来求解最优策略。

样本路径是与环境的交互中产生的，你先要实现一个环境模型。假设实际位置由环境按图1.1(b) 的方式决定：机器人每次移动的实际结果是机器人以0.8 的概率移向所选择方向，也可能是以0.1的概率移向垂直于所选方向。如果实际移动的方向上有障碍物，则机器人会停在原地。机器人移动到图中每个格子，会获得一个报酬, 图1.1(a) 中标有+1和-1 的格子中标记的就是该格子的报酬，其他格子的报酬是-0.04. 报酬会随着时间打折，假设折扣是1。

1.2　实验过程及要求

1. 实验环境要求：Windows/Linux操作系统，Python编译环境，numpy、scipy等程序库。



(a) 地图　(b) 行动方向

图 1.1: 机器人行动环境

2. 编写一个环境，它能跟机器人交互，主要提供行动的结果–下一步的状态及获得的报酬。

3. 已知机器人的策略$\pi(x)$, 通过与环境交互学习在该策略下的价值函数$U(x)$（或者叫效用函数）。

4. 机器的行动价值函数是$Q(x,a)$，且$\pi(x) = \arg\max_a Q(x,a)$是最优决策, 通过与环境交互学习这个行动价值函数$Q(x,a)$。

5. 已知机器人的策略$\pi(x)$, 用一个线性函数来逼近价值函数，通过与环境交互学习这个线性函数。

6. 撰写实验报告。

1.3　教学目标

1. 理解和掌握强化学习的原理。

2. 能够应用时序差分方法，计算状态的价值函数。

3. 能够应用时序差分方法，计算状态的行为价值函数，从而得到最优决策。

4. 能够提出一个新函数来逼近价值函数，并用梯度下降法来计算该新函数的参数。

5. 能够分析不同方案的优缺点，提高对复杂工程问题建模和分析的能力。

1.4　相关知识及背景

马尔可夫决策过程的实验中，通过状态转移矩阵和状态的报酬，用价值迭代法和策略迭代法来求解状态的效用以及最优决策。在没有状态转移矩阵和报酬定义的情况下，机器人可以通过反复与环境进行交互，获得状态变化数据以及报酬数据，并通过这些数据来学习状态效用和最优决策，这个过程称为强化学习。强化学习、监督学习、无监督学习构成了机器学习的几大类别。

强化学习特别适用于类似游戏智能，通过在游戏中获得奖励，引导Agent采取正确行动。2016年在围棋中击败顶级人类选手的人工智能AlphaGo，主要应用的是搜索技术、有监督学习技术和强化学习技术，通过大量的战例来训练模型。

本次实验通过奖励机制，应用时序差分方法来学习机器人的价值函数以及行动价值函数，并获得最优策略。

1.5　实验教学与指导

1.5.1　环境模型

需要模拟一个环境，由这个环境告知机器人的行动的结果。这个模型可以自定义一个转移矩阵P，并用它生成机器人的新状态。在这个模型中，除了P不能被学习算法使用外，其他的属性和方法可以被学习算法使用。

```
class Env():
    def __init__(self,name):
        self.Name=name
        self.N=11
        self.A=np.arange(4)
        self.X=np.arange(self.N)
        self.makeP() #定义转移矩阵
        self.makeR() #定义报酬向量
        self.Gamma=1            #折扣
        self.StartState=0
        self.EndStates=[6,10]
    def action(self,x,a):
    #环境模型通过action函数告知Agent报酬以及新状态
        x_=np.random.choice(self.N,p=self.P[x,a,:])
```

```
15          return x_
```

1.5.2　被动学习-时序差分方法TD Learning

给定一个策略$\pi(x)$，让机器人学习状态的价值函数，称为被动学习。通过与环境交互，机器人获得一个样本路径$sample = (x_0, r_0, a_0, x_1, r_1, a_1, \ldots, a_{n-1}, x_n, r_n)$。价值函数$U^{\pi}(x)$的定义是机器人从状态$x$出发，按照策略$\pi$连续移动所获得的报酬总和的期望值。在通过样本计算时，可以近似认为

$$\begin{aligned} U(x_i) &\approx \sum_{t=i}^{t=n} \gamma^{t-i} r_t \\ &\approx r_i + \gamma U(x_{i+1}) \end{aligned} \tag{1.1}$$

在进行迭代时，可以使用时序差分迭代公式：

$$\begin{aligned} U(x_i) &:= (1-\alpha)U(x_i) + \alpha(r_i + \gamma U(x_{i+1}) \\ &:= U(x_i) + \alpha(r_i + \gamma U(x_{i+1}) - U(x_i)) \end{aligned} \tag{1.2}$$

其中α是一个权重，称为学习率。

```
class TD():
    def __init__(self,E):
        self.E=E
        self.Alpha=0.5
        self.Pi=[3,2,2,2,3,3,0,0,0,0,0,0]
        self.U=[0,0,0,0,0,0,-1,0,0,0,1]

    def train(self):
        x=np.random.choice([0,1,2,3,4,5,7,8,9])
        while x not in self.E.EndStates:
            a=self.Pi[x]
            _x = self.E.action(x,a)
            r=self.E.R[x]
            self.U[x]=self.U[x]+self.Alpha*(
                r+self.E.Gamma*self.U[_x]-self.U[x])
            x=_x
```

1.5.3　主动学习-Q Learning

假设机器人由一个行为价值函数$Q(x,a)$，定义在状态x下采取行动a的价值，那么根据Q函数机器人可以按概率选取一个行动，并与环境进行交互，通过奖励来更新Q函数。如果学到了正确的Q 函数，则贪心策略就是最优策略，

$$\pi(x) = \arg\max_a Q(x,a)$$

而且状态x的价值满足

$$U(x) = \max_a Q(x,a)$$

。

Q 函数的时序差分迭代公式为

$$Q(x_i,a_i) := Q(x_i,a_i) + \alpha(r_i + \gamma \max_{a'} Q(x_{i+1},a') - Q(s_i,a_i)) \tag{1.3}$$

```
class Q_Learning():
    def __init__(self, E):
        self.E=E
        self.Alpha=0.5
        self.Q=np.ones((11,4))/4
        self.Q[10,:]=1
        self.Q[6,:]=-1

    def train(self):
        x=np.random.choice([0,1,2,3,4,5,7,8,9])
        while x not in self.E.EndStates:
            P=normal(self.Q[x])
            a=np.random.choice(4,p=P)
            _x = self.E.action(x,a)
            r=self.E.R[x]
            self.Q[x,a]=self.Q[x,a]+self.Alpha*(
                r+self.E.Gamma*np.max(self.Q[_x])\
                -self.Q[x,a])
            x=_x
```

1.5.4　价值函数的线性逼近

当状态数过多而价值函数不宜用列表方式表达时，可以用函数逼近的方式。有些价值函数可以用线性函数做很好的逼近。在本实验中，状态x的特征为行列坐标(r,c)，如果令$U(x)=w_1+w_2r+w_3c$, 则可以定义平方损失函数$J=\frac{1}{2}(U(x)-S)^2$, 其中，S是从状态x开始的带折扣报酬序列总和。可以应用梯度下降法来求$U(x)$的参数$w=(w_1,w_2,w_3)$

$$\begin{aligned}\nabla J_w &= (U(x)-S)\nabla U_w \\ &= (U(x)-S)(1,r,c) \qquad (1.4)\\ w &:= w-\alpha(U(x)-S)(1,r,c) \qquad (1.5)\end{aligned}$$

```
class F_TD():
    def __init__(self,E):
        self.w=np.array([0.5,0.5,0.5])
        self.E=E
        self.Alpha=0.001
        self.Pi=[3,2,2,2,3,3,0,0,0,0,0]

    def U(self,x):
        if x==10:
            return 1
        if x==6:
            return -1
        (row,col)=self.E.X2RowCol[x]
        return np.dot( np.array([1,row,col]), self.w)

    def dU(self,x):
        (row,col)=self.E.X2RowCol[x]
        return np.array([1,row,col])

    def train(self):
        x0=np.random.choice([0,1,2,3,4,5,7,8,9])
        a0=self.Pi[x0]

        Rsum=self.E.R[x0]
        x=x0
        a=a0
```

```
27          gamma=self.E.Gamma
28          while x not in self.E.EndStates:
29              x=self.E.action(x,a)
30              Rsum += gamma*self.E.R[x]
31              a=self.Pi[x]
32              gamma *= self.E.Gamma
33
34          self.w= self.w + self.Alpha*(
35              Rsum-self.U(x0))*self.dU(x0)
```

1.6　实验报告要求

实验报告需包含实验任务、实验平台、实验原理、实验步骤、实验数据记录、实验结果分析和实验结论等部分，特别是以下重点内容：

1. 建立机器人导航问题的马尔可夫决策模型，实现Env模块。
2. 用时序差分方法计算价值函数和行动价值函数。
3. 用线性函数逼近价值函数。
4. 利用环境模型，应用马尔可夫决策方法得到价值函数和最优决策，检验强化学习的结果。

1.7　考核要求与方法

实验总分100分，通过实验报告进行考核，标准如下：

1. 报告的规范性10分。报告中的术语、格式、图表、数据、公式、标注及参考文献是否符合规范要求。
2. 报告的严谨性40分。结构是否严谨，论述的层次是否清晰，逻辑是否合理，语言是否准确。
3. 实验的充分性50分。实验是否包含“实验报告要求”部分的4个重点内容，数据是否合理，是否有创新性成果或独立见解。

1.8　案例特色或创新

本实验的特色在于：通过对机器人导航问题，培养学生应用强化学习技术，要求学生能够通过时序差分方法，利用样本数据来计算价值函数和行动价值函数，并计算优化决策。提供了状态数目较多时对价值函数的逼近方法。提高学生对复杂工程问题建模和分析的能力。