增强学习

姓名：魏亚东
人工智能与机器人研究所
2017年7月

CONTENS

一、马尔可夫性质
二、马尔可夫决策过程(MDP)
三、值函数
四、增强学习(Reinforcement Learning and control)
五、tensorflow实现策略网络(Cartpole)
六、Q-Learning和DQN实现马里奥小游戏agent
七、总结

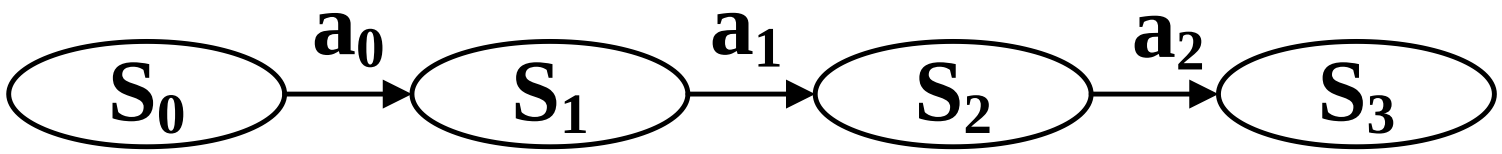
1.1马尔可夫子模型

- 当一个随机过程在给定现在状态及所有过去状态情况下，其未来状态的条件概率分布仅依赖于当前状态；换句话说，在给定现在状态时，它与过去状态（即该过程的历史路径）是条件独立的，那么此随机过程即具有马尔可夫性质。具有马尔可夫性质的过程通常称之为马尔可夫过程。
- 马尔科夫链(Markov Chain)：满足马尔科夫性质的随机变量序列(x1,x2,x3...),即给出当前状态，将来状态和过去状态中之间是相互独立的。
- 隐式马尔科夫模型(Hidden Markov Model):双重随机过程，不仅状态转移之间是一个随机事件，状态和输出之间也是一个随机事件
- 马尔可夫决策过程(Markov Decision Process, MDP)也具有马尔可夫性，与上面不同的是MDP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。

	不考虑动作	考虑动作
状态完全可见	马尔科夫链(MC)	马尔科夫决策过程(MDP)
状态不完全可见	隐马尔科夫模型(HMM)	不完全可观马尔科夫决策过程(POMDP)

2.1马尔可夫决策过程

- 马尔可夫决策过程由一个四元组构成M(S,A,Psa,R)。
- S:表示一个状态集合(States),Si表示第i步的状态。
- A：表示一组动作(actions)，Ai表示第i步的动作。
- Psa:表示状态转移概率，Psa表示的是在当前状态下，经过actions作用之后，会转移到其他状态的概率分布情况，比如在状态s下执行动作a，转移到s’的概率可以表示为(Ps’|s,a)。
- R是回报函数(reward function),回报函数可以表示为r(s,a)
- MDP 的动态过程如下：某个智能体(agent)的初始状态为s0，然后从 A 中挑选一个动作a0执行，执行后，agent 按Psa概率随机转移到了下一个s1状态。然后再执行一个动作a1，就转移到了s2，接下来再执行a2...，我们可以用下面的图表示状态转移的过程。



2.2马尔可夫决策过程

- 经过上面的转移路径后，得到的回报函数如下：

$$R(s_0,a_0)+\gamma R(s_1,a_1)+\gamma^2 R(s_2,a_2)+\cdots$$

- 如果R只和S有关，那么上式可以写作：

$$R(s_0)+\gamma R(s_1)+\gamma^2 R(s_2)+\cdots$$

- 我们的目标是选择一组最佳的action，使得全部的回报加权和期望最大。

$$\max E(R(s_0,a_0)+\gamma R(s_1,a_1)+\gamma^2 R(s_2,a_2)+\cdots)$$

- 从上面的式子可以看到，t时刻的回报值被打了γ^t的折扣，而且γ是代表的是一个衰减系数，越靠后的状态对回报和影响越小，最大期望值也就是将大的R(si)尽量放在前面，小的尽量放在后面。
- 已经处于某个状态s时，我们会以一定的策略π来选择下一个动作a执行，然后转换到下一个状态s’，我们将这一个动作选择的过程称为策略(policy)，每一个policy其实就是一个状态到动作的映射函数，给定π也就是给定了a=π(s)，也就是知道了每个状态下一步应该执行的动作。

3.1状态值函数(值函数只与状态有关)

- 从递推的角度上考虑，当期状态s的值函数V，其实可以看作是当前状态的回报R(s)和下一状态的值函数V’之和，也就是将折算累计回报公式变为：

$$V^{\pi}(s)=R(s_0)+\gamma(E[R(s_1)+\gamma R(s_2)+\gamma^2 R(s_3)+\cdots])=R(s_0)+\gamma V^{\pi}(s')$$

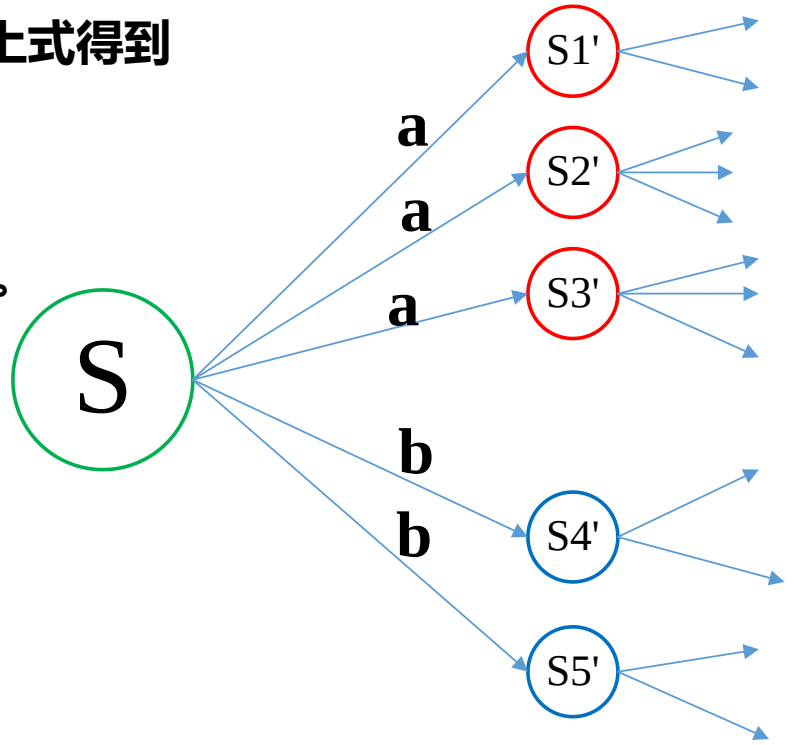
- 然而，我们我们需要注意的是虽然给定策略之后，在给定s状态的条件下，a是唯一的，但是比如，选择a为向前掷筛子，那么下一个状态可能有6种，再由Bellman等式，从上式得到

$$V^{\pi}(s)=R(s)+\gamma\sum_{s'\in S}P_{s\pi(s)}(s')V^{\pi}(s')$$

- 前面一项的R(s)称为立即回报，V(s’)为下一个状态值函数的期望值。
- 定义了状态V*，我们在定义最优的策略π*如下：

$$\pi^{*}(s)=\arg\max_{a\in A}\sum_{s'\in S}P_{sa}(s')V^{*}(s')$$

$$V^{*}(s)=V^{\pi^{*}}(s)\geq V^{\pi}(s)$$



3.2动作值函数

- 状态值函数值函数只与状态s有关，如果与状态s和动作a都有关，就是所谓的Q函数：

$$Q^{\pi}(s,a) = E(R(s_0,a_0) + \gamma R(s_1,a_1) + \gamma^2 R(s_2,a_2) + \cdots)$$
$$= R(s,a) + \gamma Q^{\pi}(s',a')$$

- 从上式我们可以看出，我们不仅仅依赖状态s和策略π，并且还依赖于动作a。综上，我们可以将MDP的最优策略定义如下：

$$\pi^* = \arg\max V^{\pi}(s)$$

- 关于MDP的求解主要分为值迭代和策略迭代，分别站在不同的角度对MDP进行求解，这里我们不在赘述，网上有很多相关资料。下面我们简单阐述下动作值函数的值迭代求解方式，即所谓的Q-learning。

4.0 增强学习(Reinforcement Learning)

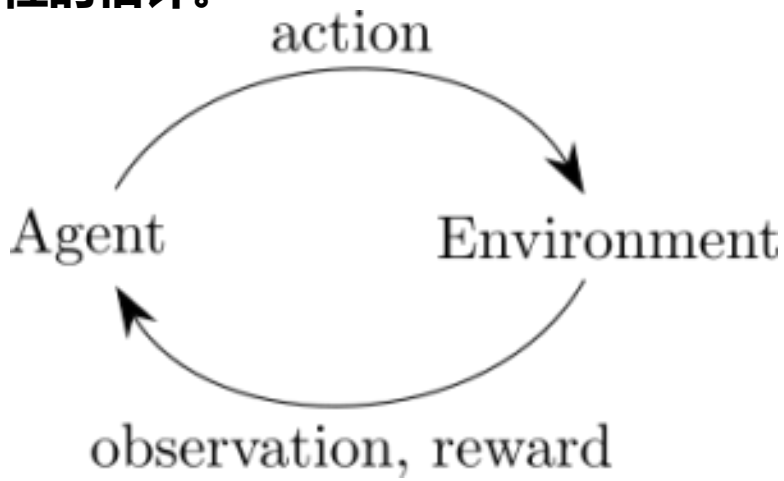
- 强化学习主要解决连续决策的问题，可以在复杂的，不确定的环境中学习如何实现我们的目标，应用场景非常广，几乎包括了所有需要做一系列决策的问题。
- 强化学习的三个主要的概念：环境，行动，奖励。强化学习的目标是获得最多的累计奖励。

强化学习方法：

- Policy-Based 和 Value-Based(Q-Learning)是强化学习中最重要的两类方法。主要区别是，Policy-Based 预测的是某个环境状态下应该采取的action，Value-Based的方法预测的是某个环境状态下所有action的期望价值。之后选择期望价值最高的action执行策略。结合了深度学习之后，Policy-Based变成了Policy策略网络(Policy Network)，Value-Based网络变成了估值网络(DQN)。
- Value-Based适合于只有少量离散action情形，Policy-Based则更加通用。
- 蒙特卡洛搜索树(Monte Carlo Tree Search)

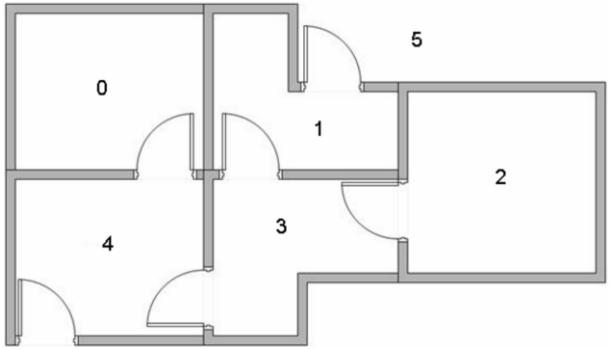
5.0策略网络实现

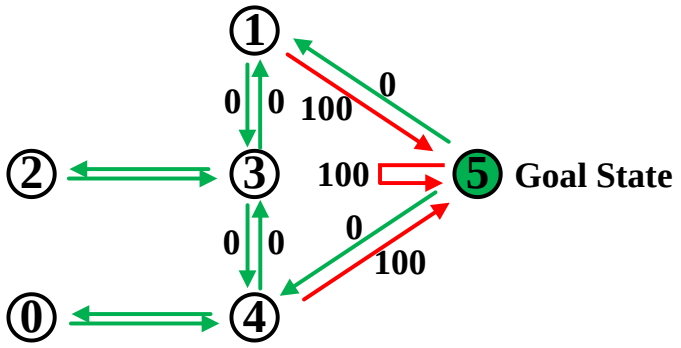
- 策略网络，即建立一个神经网络模型，它可以通过观察环境状态，**直接预测出目前最应该执行的策略**（policy），执行这个策略可以获得最大的期望收益（包括现在的和未来的reward）。和之前的任务不同，在强化学习中可能没有绝对正确的学习目标，样本的feature和label也不在一一对应。我们的学习目标是期望价值，即当前获得的reward和未来潜在的可获取的reward。所以在策略网络中不只是使用当前的reward作为label，而是使用Discounted Future Reward，即把所有未来奖励一次乘以衰减系数γ。这里的衰减系数是一个略小于但接近1的数，防止没有损耗地积累导致Reward目标发散，同时也代表了对未来奖励的不确定性的估计。



- Gym是OpenAI推出的开源的强化学习的环境生成工具。在Gym中有两个核心的概念，一个是Environment，指我们的任务或者问题，另一个就是Agent，即我们编写的策略或者算法。Agent会将执行的Action传给Environment，Environment接受某个Action后，再将结果Observation(即环境状态)和Reward返回给Agent。

6.1Q-Learning的例子





$$MatrixR = \begin{array}{c} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

- 问题描述：agent对于环境一无所知,不知道自己的开始的时候在那个房间，agent知道的信息只是自己的目的地是去5号区域。
- 每一个小房子之间的是否想通，通向哪里是已知信息。
- Reward矩阵，两个门之间没有连接的时候设置reward为-1，到达5的reward设置为100，有连接但是没有直接连接设置reward为0。
- Select action：

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

- Q函数计算公式：

$$Q(s,a) = r(s,a) + \gamma \max_{a'}(Q(s',a'))$$

$$r(s,a) = immediate\_reward$$

$$\gamma = relative\_value\_of\_dalay$$

$$s' = the\_new\_state$$

6.2Q-Learning例子

$$Q = \begin{array}{c} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

1--5

$$Q(1,5) = R(1,5) + \gamma Max(Q(5,1), Q(5,4), Q(5,5))$$
$$= 100$$

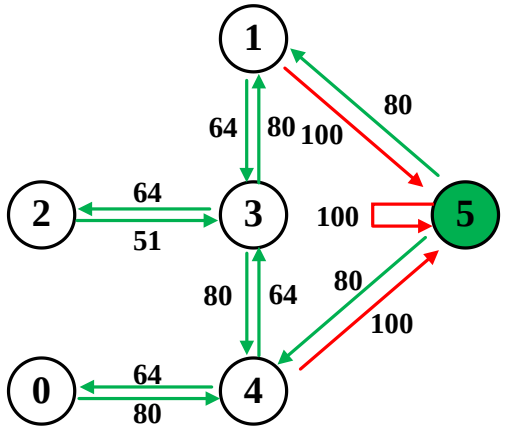
$$Q = \begin{array}{c} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

3--1

$$Q(3,1) = R(3,1) + \gamma Max(Q(1,3), Q(1,5))$$
$$= 80$$

$$Q = \begin{array}{c} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

无限次迭代



归一化

$$Q = \begin{array}{c} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$

$$Q = \begin{array}{c} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$

6.2DQN训练超级玛丽agent

- 利用CNN来识别游戏总马里奥的状态，并利用增强学习算法做出动作选择，然后根据新的返回状态和历史状态来计算reward函数从而反馈给Q函数进行迭代，不断的训练直到游戏能够通关。研究人员在训练了一个游戏后，将相同的参数用在别的游戏中发现也是适用的。

谢谢！

人工智能与机器人研究所

2017年6月