MDPs

0.参考

https://www.youtube.com/watch?v=4LW3H_Jinr4 马尔可夫过程、价值迭代
https://www.youtube.com/watch?v=ZToWj64rxvQ&t=565s 马尔可夫过程，贝尔曼方程，收敛性，策略方法，策略评估，策略迭代
https://inst.eecs.berkeley.edu/~cs188/textbook/ notes

一、MDP基础知识

1.1 马尔可夫性质(MDP假设)

given the present state, the future and the past are independent
in MDPs means **action outcomes depend only on the current state
$P(S_{t+1}=s'|S_t=s,A_t=a,S_{t-1}=s_{t-1},\ldots,S_0=s_0)=P(S_{t+1}=s'|S_t=s,A_t=a)$

1.2 定义组成

$\mathcal{S}$ (状态集合) $\mathcal{A}$ (动作集合)
$T(s,a,s')$ 或 $P(s'|s,a)$ (状态转移函数，表示从 s 采取 a 到 s' 的概率)
这二者数值上相等，只是T 为函数，而P 表示一个概率数值

$R(s,a,s')$ (奖励函数，表示从 s 采取 a 到 s' 获得的奖励)

起始状态 s_{start}终止状态 s_{terminal} (可选)
$Policy$:
$\pi^*:S\to A$ 最佳策略π^* 为从S 映射到A
$Utility$
the sum of (discounted) rewards （折扣）奖励函数的和

奖励函数组成

折扣因子γ 原理：尽快获得奖励

1.3 有限视界与折扣 Finite Horizons and Discounting

为了解决：无限奖励问题
如果成本较低，可能会持续获得奖励，奖励函数没有上界，为了解决这个问题，可以采用三种方法

1. 有限视野：固定时间步上界T
   类似于深度有限搜索（depth-limited search），但是可能会导致**非平稳策略（nonstationary policies）**，意味着最优策略会依赖于“剩余时间（time left）”。比如，在游戏快结束时，你的策略可能和游戏刚开始时不同
2. 采用折扣奖励γ，其中$0 < |\gamma| < 1$

   $$U([r_0, r_1, \ldots, r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t$$

   如果 $r_t \leq R_{\max}$, 则有:

   $$\sum_{t=0}^{\infty} \gamma^t r_t \leq \frac{R_{\max}}{1-\gamma}$$

3. 人为保证每个策略都能达到终止状态$\mathbf{s}_{terminal}$
   这种方法与有限视野类似，都是通过明确的终止来避免无限奖励，但它不是基于固定步数，而是基于特定状态的达成

二、价值迭代解决MDP

1.最优价值相关

1.1 一些定义

最优量 (Optimal Quantities)：

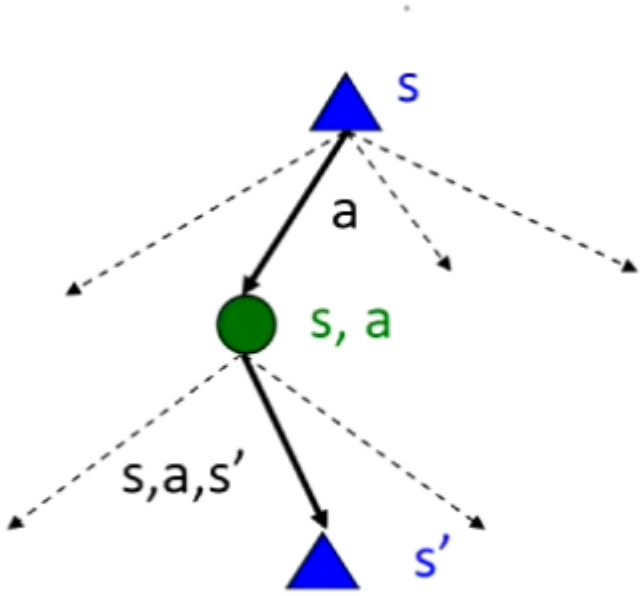
一个状态的价值 (Optimal Value Function):
$V^*(s)$:表示从状态 s 开始，并且从那时起**按照最优策略**（acting optimally）行动所能获得的**期望累积奖励（utility）。**
即：如果我从状态 s 开始，并且之后每一步都做出最好的选择，我最终能期望获得多少总奖励。它是衡量一个**状态“好坏”的指标

一个 Q-状态的价值 (Optimal Q-Value Function):
$Q^*(s,a) =$ 表示从状态 s 开始，**首先采取动作 a**，然后从下一个状态开始**按照最优策略**行动所能获得的**期望累积奖励。**
即：如果我目前在状态 s，并且我决定现在采取动作 a（即使它可能不是最优的动作），然后从下一个状态开始，我每一步都做最好的选择，我最终能期望获得多少总奖励。它是衡量**在特定状态下，某个特定动作“好坏”的指标

最优策略 (Optimal Policy):
$\pi^*(s) =$ 即 a^*
π 本质上是一个函数，它告诉代理在每个状态 s 应该选择哪个动作 a。最优策略 π^* 就是那个能够

最大化长期累积奖励的策略。*对于每个状态 s，$\pi^*(s)$ 都会给出那个能带来最高 $Q^(s, a)$ 值的动作 a^*

(strike!):



这里用图来辅助理解state 、Q-state 、s‘之间的关系
可以理解为状态 s 与 s’ 之间加入了一个新的q-state节点，这个新节点用两个东西来连接 s 与 s‘
从 s 出发，可以选择动作空间中的一个动作 a（根据Q-状态的价值比如max或者特定的Q-状态），达到一个q-state节点，在这个节点上，状态转移函数$T(s,a,s')$ 将节点指向新的 s’

1.2 最优状态价值 求解

状态价值 $V^*(s)$ 与 Q-价值 $Q^*(s,a)$ 的关系
$V^*(s) = \max_a Q^*(s,a)$
即：采取状态s下**最优**的动作a，是最优的“期望累计奖励”$V^*(s)$

Q-价值 $Q^*(s,a)$ 的定义
$Q^*(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^*(s')]$
即：在一个状态 s 采取动作 a 的 Q-价值。它是对**所有可能的下一个状态 s' 进行加权求和，权重是$P(s'|s,a)$ (等价于$T(s,a,s')$)。** 而奖励计算分为**即时奖励$R(s,a,s')$ 和未来折扣奖励$\gamma V^*(s')$**
(strike!) 这里$V^*(s)$ 都是相对于未来而言的，表示如果未来都按照最优策略能获得期望奖励，所以在这里乘上γ 折扣因子

贝尔曼最优方程 (Bellman Optimality Equation) :
$V^*(s)$的递归定义，将Q-价值带入$V^*(s)$ 与$Q^*(s,a)$ 关系中：

$$V^*(s) = \max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^*(s')]$$

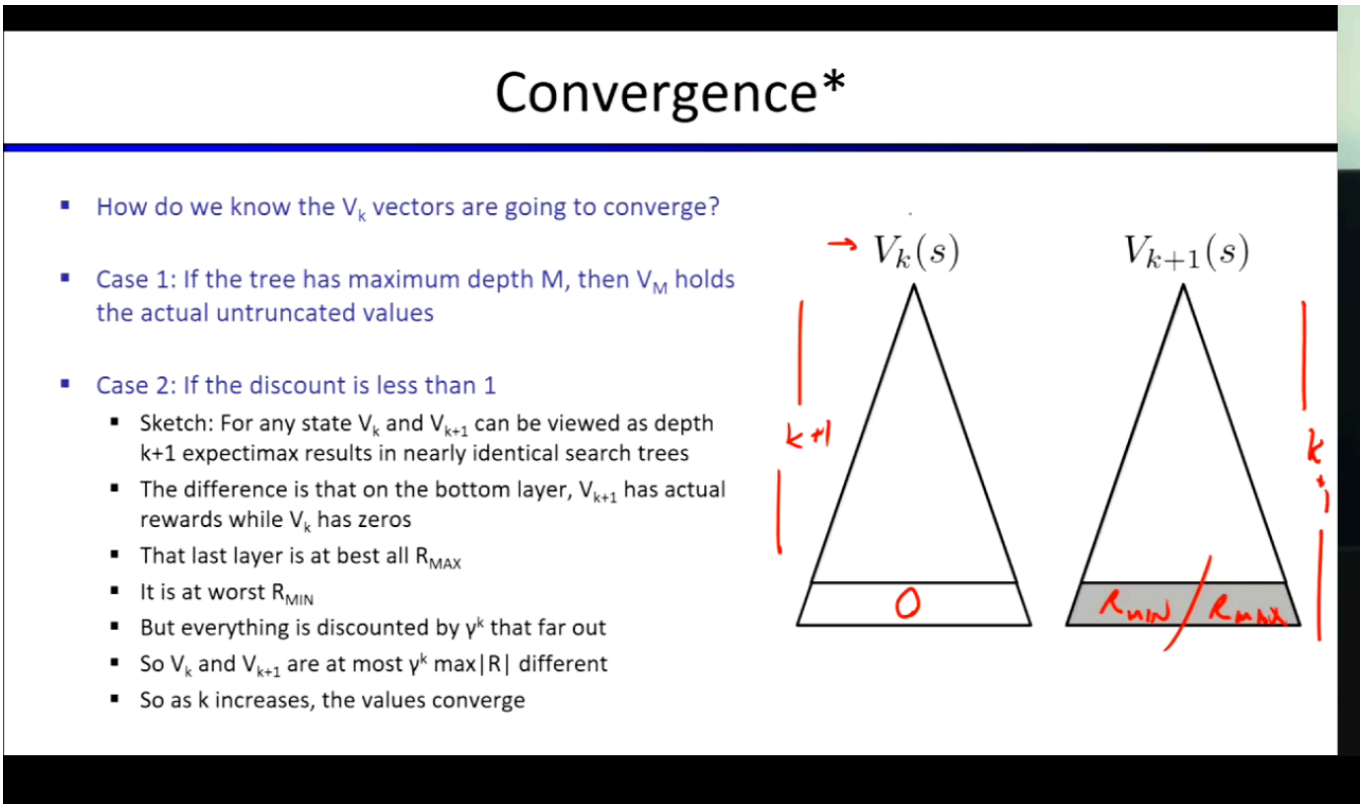
1.3 最优价值迭代算法

$V_k(s)$ 为在k个时间步内、s的最优状态价值
迭代公式：

$$V_0^*(s) = 0$$

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_k(s')]$$

proof. 该算法是收敛的，也就是说不会有无限的$V(s)$：

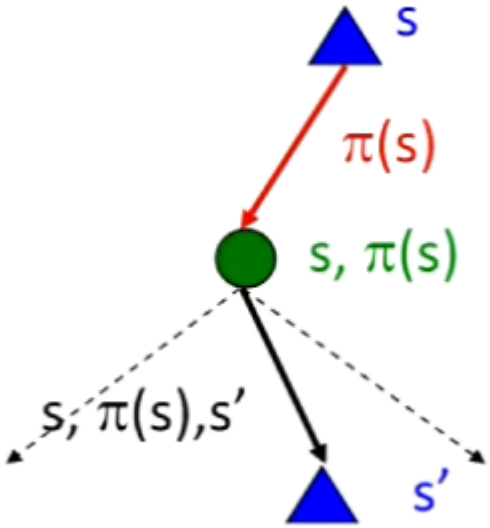


最后一步即当$k \to \infty$ 时，$V_k(s)$ 与 $V_{k+1}(s)$ 之间相差$\gamma^k \max|R| \to 0$
所以根据cauchy收敛定理，$V_k(s)$ 收敛

2.策略相关

2.1 固定策略价值 求解

$V^\pi(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V^\pi(s')]$



这里从 s 到 q-state 与价值迭代算法不同，只有一条边就是通过给出的策略$\pi(s)$ 到达

2.2 策略评估 Policy Evaluation

与价值迭代类似只不过最优策略变为固定策略π

$$V_0^{\pi}(s) = 0$$

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, a, s') + \gamma V_k^{\pi}(s')]$$

**这是一个线性方程组！！！ 没有max！
策略评估解决的是如何判断一个策略的好坏？由于我们无法求解$V^{\pi}(s)$ (不是k步以内的值，而是最终值，即一直采用策略π 我能获得的最终期望奖励U)，转而通过引入迭代的方式，计算$k+1$与k 的关系来逼近

3. 策略提取，在最优价值下获取策略

3.1 已知$V^*(s)$

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^*(s')\right]$$

3.2 已知$Q^*(s, a)$

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

所以想要策略或者动作时，Q-value更好！！！

三、策略迭代解决MDP

3.1 策略迭代过程

1. 策略评估 比价值迭代快$\mathcal{A}$ 倍
2. 策略改进 与价值迭代一样快

公式推导:

1. 策略评估

$$V_{k+1}^{\pi_i}(s) \leftarrow \sum_{s'} T(s, \pi_i(s), s') \left[R(s, \pi_i(s), s') + \gamma V_k^{\pi_i}(s') \right]$$

   说明：见上文策略评估
2. 策略改进

$$\pi_{i+1}(s) = \arg\max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V_k^{\pi_i}(s') \right]$$

   说明：右侧为如果按照旧策略π_i 继续执行的话，能达到的最好动作，有:

$$V^{\pi_{i+1}}(s) \geq V^{\pi_i}(s) \quad \text{对于所有状态 } s$$

   即策略得到了改进!

RL

一、RL基础综述

1.1 目标

存在MDP，但是不知道MDP具体参数，只能行动来获取信息
RL解决应该怎么做的问题

1.2 基础知识

在每个时间步(time step)内，agent从状态s开始，采取动作a，到达状态s'，获得奖励r
(s, a, s', r) 元组称为一个样本(sample)，agent会持续采取行动、持续收集样本(sample)直到终止状态，这种样本(sample)的集合称为**epsode**

1.3 Model-Based VS. Model-Free

基于模型的学习利用探索期间获得的样本来**估计**MDP问题中的转移函数(transition)和奖励函数(reward)，利用估计出来的T、R 来解决MDP(使用上文所说的价值迭代或者策略迭代)

无模型学习则是直接尝试估计状态的价值$V(s)$ 或 Q 值$Q(s, a)$ ，而不是尝试构建或者估计 MDP 的奖励和转移模型。

但是！实际上两者的数学原理一致

二、基于模型学习 Model-Based Leearning

通过记录进入每个Q-state后到达每个状态s‘的次数，对转移函数$T(s,a,s')$ 进行估计，再进行归一化转变为概率：
分子为从s状态转变为s’状态的总次数，分母为遍历所有能从s状态到达的状态s‘’的总数

$$\hat{T}(s,a,s') = \frac{N(s,a,s')}{\sum_{s''} N(s,a,s'')}$$

通过记录每次到达三元组(s,a,s') 时获得的奖励r 可以对奖励函数$R(s,a,s')$进行估计，得到$\hat{R}(s,a,s')$
基于模型学习原理直观，但是维护所有三元组(s,a,s') 代价高

三、无模型学习 Model-Free Learning

分为主动强化学习和被动强化学习，其中

1. 主动强化学习(Direct Evaluation、Temporal Difference Learning)：agent已经有一个策略(Polivy)来遵循，在进行每一个eipisode是学习该策略下状态价值，与 MDP 的策略评估在已知 T 和 R 时所做的事情完全相同
2. 被动强化学习(Q-Learning)：agent使用收到的犯规来更新迭代策略(Policy)，同时对Q值(或者价值函数)进行估计，在充分探索之后得到最佳策略

3.1 直接评估 Direct Evaluation

给定一个固定策略 π，智能体在环境中按照 π运行多个回合（episode），收集状态-回报样本。
对于每个状态 s，我们统计：

1. $N(s)$ 状态s被访问个数
2. $U(s)$ 在状态s时获得回报的累计和
   则：对状态s的价值进行估计：

$$V^{\pi}(s) \approx \frac{U(s)}{N(s)}$$

3.2 时序差分学习 Difference Learning

TD试图解决：在没有权重函数$T(s,a,s')$ 时如何计算这个加权平均值的问题

$$V^{\pi}(s) = \sum_{s'} T(s,\pi(s),s') \Big[R(s,\pi(s),s') + \gamma V^{\pi}(s') \Big]$$

首先初始化

$$\forall s, \quad V^{\pi}(s) = 0$$

规定样本价值

$$\text{sample} = R(s, \pi(s), s') + \gamma V^{\pi}(s')$$

对样本进行加权平均，其中α 称为学习率

$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha \cdot \text{sample} \qquad 0 \leq \alpha \leq 1$$

迭代

$$V_k^{\pi}(s) \leftarrow (1-\alpha)V_{k-1}^{\pi}(s) + \alpha \cdot \text{sample}_k$$

$$V_k^{\pi}(s) = \alpha \sum_{i=1}^{k} (1-\alpha)^{k-i} \cdot \text{sample}_i$$

这意味着旧sample($sample_i, i$较小时)在最后策略占比极小
与直接评估相比，收敛更快

3.3 Q学习 Q-Learning

直接评估和时序差分学习最终都会学习到它们所遵循策略下所有状态的真实价值。然而，它们都有一个主要的固有问题——我们希望为我们的智能体找到一个最优策略，这需要了解状态的 Q 值。而从状态价值中计算Q值需要转移函数T和奖励函数R，通过贝尔曼方程计算Q
Q迭代

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \Big[R(s,a,s') + \gamma \max_{a'} Q_k(s',a') \Big]$$

$$\text{sample} = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \cdot \text{sample}$$