强化学习: 强化学习是一种优化智能体在环境中行为的一种方法。根据环境反馈的奖励，调整智能体的行为策略，提升智能体实现目标的能力。

〇强化学习考虑的是**序贯决策过程**:

智能体处在特定的环境中产生一系列的动作，而环境能够根据这些动作改变智能体的当前状态。

举例

遥控直升飞机的特技表演；打败围棋世界冠军；管理股票证券；发电厂调控；控制人型机器人双足行走；视频游戏上超越人类

目标：选择一组动作使未来奖励和最大化

动作可能在未来很久才会产生影响；奖励可能是延时的；有可能会牺牲短期利益从而获得长期的回报

举例:

投资（要几个月才会收益）；直升机加油（防止几小时后没油坠机）

强化学习:

①产生的结果(动作) 能够改变数据的分布(状态)

②最终的目标可能要很长时间才能观察到(下棋)

③没有明确的标签(label) 数据

④根据当前的奖励，最终实现长远的目标

监督学习(Supervised Learning, SL)/非监督学习:

①产生的结果(输出) 不会改变数据的分布②结果是瞬时的③要么有明确的标签数据(SL)④要么完全没有任何标签数据(USL)

●马尔可夫性 Markov property

智能体未来的状态只与当前时刻的状态S_t有关，而与过去的状态$\{S_1,\dots,S_{t-1}\}$无关，那么称智能体的模型具有马尔可夫性。

$$P[S_{t+1}|S_t]=P[S_{t+1}|S_1,\dots,S_t]$$

未来只与当前有关，与历史无关$S_{1:t}\to S_t\to S_{t+1:\infty}$

一旦当前状态确定了，历史状态都可以丢弃了，也就是说当前状态足以决定未来状态是什么样的　全/部分可观测

强化学习主要研究的是具有马尔可夫性的问题.　s_t:环境/模型内在状态

状态转移矩阵:

对于一个马尔可夫状态s和后继状态s'，状态转移概率定义为

$$P_{ss'}=P(S_{t+1}=s'|S_t=s)$$　o_t:智能体观测量

状态转移矩阵P 定义从所有状态s到所有后继状态s'的转移概率

$$P=\text{from}\overset{\text{to}}{\begin{bmatrix}P_{11}&\cdots&P_{1n}\\\vdots&\ddots&\vdots\\P_{n1}&\cdots&P_{nn}\end{bmatrix}}$$ 其中矩阵的每行和为 1

●马尔可夫过程 Markov Process

一个马尔可夫过程是一个无记忆的随机过程，即一组具有马尔可夫性的随机状态序列 S1; S2;···

定义:

马尔可夫过程（马尔可夫链）可以用一组$\langle S,P\rangle$表示

S是（有限）状态集

P是状态转移概率矩阵$P_{ss'}=P(S_{t+1}=s'|S_t=s)$

●马尔可夫奖励过程 Markov Reward Process

一个马尔可夫奖励过程是一个马尔可夫链加上奖励

定义:

一个马尔可夫奖励过程由一组$\langle S,P,R,\gamma\rangle$构成

S是一组有限状态集

P是状态转移概率矩阵$P_{ss'}=P(S_{t+1}=s'|S_t=s)$

R是奖励函数，$R_s=E(R_{t+1}|S_t=s)$

γ是折扣因子，$\gamma\in[0,1]$

回报 Return:回报G_t代表从t时刻往后所有的折扣奖励:

$$\boldsymbol{G_t=R_{t+1}+\gamma R_{t+2}+\cdots=\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}}$$

折扣因子γ代表未来的奖励在当前时刻贡献的价值

$k+1$时刻后的奖励R对当前回报的贡献只有$\boldsymbol{\gamma^k R}$

0=目光短浅；1=目光长远　回报对应某一具体的轨迹

如果想要调整奖励的重要性，数学上很方便，在循环马尔可夫过程问题中能避免无穷回报。但是有可能会忽视未来奖励。如果奖励代表金钱，近期的奖励会比远期产生更多的收益。自然界的人类或动物行为模式更倾向于近期奖励。有时候也会使用无折扣的马尔可夫奖励过程（即 1）。例如所有的事件序列都有终止状态

价值函数$v(s)$ 代表智能体在状态s 下的长期价值

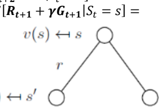
一个马尔可夫奖励过程的状态价值函数等于从状态$\boldsymbol{s}$出发的期望回报　价值代表所有轨迹的期望

$$\boldsymbol{v(s)=\mathbf{E}[G_t|S_t=s]}$$

MRPs 的贝尔曼方程 Bellman equation

价值函数拆分成两部分:瞬间奖励R_{t+1};后继状态的折扣价值$\gamma v(S_{t+1})$

$$v(s)=E[\boldsymbol{G_t}|S_t=s]=[\boldsymbol{R_{t+1}+\gamma R_{t+2}}+\cdots|S_t=s]=$$
$$E[\boldsymbol{R_{t+1}+\gamma(R_{t+2}}+\cdots)|S_t=s]=E[\boldsymbol{R_{t+1}+\gamma G_{t+1}}|S_t=s]=$$
$$E[\boldsymbol{R_{t+1}}+\gamma v(S_{t+1})|S_t=s]$$
$$v(s)=E[\boldsymbol{R_{t+1}}+\gamma v(S_{t+1})|S_t=s]$$
$$v(s)=R_s+\gamma\sum_{s'\in S}P_{ss'}v(s')$$



矩阵：$v=R+\gamma Pv$

$$\begin{bmatrix}v(1)\\\dots\\v(n)\end{bmatrix}=\begin{bmatrix}R(1)\\\dots\\R(n)\end{bmatrix}+\gamma\begin{bmatrix}P_{11}&\cdots&P_{1n}\\\vdots&\ddots&\vdots\\P_{n1}&\cdots&P_{nn}\end{bmatrix}\begin{bmatrix}v(1)\\\dots\\v(n)\end{bmatrix}$$

$v=(I-\gamma P)^{-1}R$: n个状态下的计算复杂度 O(n^3)

●马尔可夫决策过程 Markov Decision Process, MDP

马尔可夫决策过程是马尔可夫奖励过程加上决策. 问题的所有状态都具有马尔可夫性.

一个马尔可夫决策过程由一组$\langle S,A,P,R,\gamma\rangle$构成

S是一组有限状态集

A是有限动作集

P是状态转移概率矩阵

$$P^a_{ss'}=P(S_{t+1}=s'|S_t=s,A_t=a)$$

R是奖励函数，$R^a_s=E(R_{t+1}|S_t=s,A_t=a)$

γ是折扣因子，$\gamma\in[0,1]$

●强化学习的主要组成元素:

策略：智能体的行为

价值函数（值函数、性能指标函数）：智能体在某一状态和/或某一动作时是好还是坏

模型：智能体对真实环境的估计

策略：代表了智能体是如何行为的，是从状态到动作的映射，是状态到动作的一种分布

确定性策略：$a=\pi(s)$

随机性策略：$\pi(a|s)=P(A_t=a|S_t=s)$

一个策略定义了一个智能体的行为，MDP 问题的策略取决于当前时刻的状态(与历史无关)，即策略是静态的(时不变性)，迷宫中的箭头（上下左右）代表策略$\pi(s)$在状态s的动作

给定一个 MDP 的M $=\langle S,A,P,R,\gamma\rangle$和策略$\pi$:

状态序列$S_1,S_2,\dots$是一个马尔可夫过程$\langle S,P^\pi\rangle$

状态和奖励序列$S_1,R_1,S_2,\dots$是一个马尔可夫奖励过程$\langle S,P^\pi,R^\pi,\gamma\rangle$

$P^\pi_{ss'}=\sum_{a\in A}\pi(a|s)P^a_{ss'}$　　$R^\pi_s=\sum_{a\in A}\pi(a|s)R^a_s$

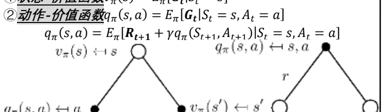
价值函数 Value Function

价值函数是对未来奖励的预测，评估智能体在某一状态下是好还是坏，因而可以用来选择对智能体最有利的动作

①***状态-价值函数***$v_\pi(s)=E_\pi[\boldsymbol{G_t}|S_t=s]$

②***动作-价值函数***$q_\pi(s,a)=E_\pi[\boldsymbol{G_t}|S_t=s,A_t=a]$

$$q_\pi(s,a)=E_\pi[\boldsymbol{R_{t+1}}+\gamma q_\pi(S_{t+1},A_{t+1})|S_t=s,A_t=a]$$



$$v_\pi(s)=\sum_{a\in A}\pi(a|s)\,q_\pi(s,a)$$
$$q_\pi(s,a)=R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}v_\pi(s')$$

$V_\pi(s)$ 的轨迹

$$s_t=s,\ a_t\sim\pi,\ r_{t+1}\sim\mathcal{R},\ s_{t+1}\sim\mathcal{P},\ a_{t+1}\sim\pi,\dots$$

$Q_\pi(s,a)$ 的轨迹

$$s_t=s,\ a_t=a,\ r_{t+1}\sim\mathcal{R},\ s_{t+1}\sim\mathcal{P},\ a_{t+1}\sim\pi,\dots$$

贝尔曼期望方程

$$v_\pi=\sum_{a\in A}\pi(a|s)\left(R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}v_\pi(s')\right)$$
$$q_\pi(s,a)=R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}\sum_{a'\in A}\pi(a'|s')\,q_\pi(s',a')$$
$$v_\pi=(I-\gamma P^\pi)^{-1}R^\pi$$

最优价值函数:

最优状态-价值函数: 在所有策略中价值函数最大的

$$v_*(s)=\max_\pi v_\pi(s)$$

最优动作-价值函数: 所有策略中动作价值函数最大

$$q_*(s,a)=\max_\pi q_\pi(s,a)$$

最优价值函数代表了智能体在该 MDP 问题下最好的性能；如果得到了最优值函数，那么 MDP 问题就已经求解了

最优策略: $\pi\geq\pi'\,if\ v_\pi(s)\geq v_{\pi'}(s),\forall s$

定理

对任意马尔可夫决策过程

总是存在一个最优策略π*比其它所有策略都不差

$$\pi_*\geq\pi\quad\forall\pi$$

所有最优策略的价值函数都相等，且等于最优价值函数，$V_{\pi_*}(s)=v_*(s)$

所有最优策略的动作-价值函数都相等，且等于最优动作-价值函数，$q_{\pi_*}(s,a)=q_*(s,a)$

寻找最优策略

一个最优策略可以通过最大化$q_*(s,a)$来确定

$$\pi_*(a|s)=\begin{cases}1 & if\ a=\arg\max_{a\in A}q_*(s,a)\\0 & otherwise\end{cases}$$

对任何 MDP 都存在一个确定性的最优策略

如果$q_*(s,a)$已知，即可得到最优策略

最优化原理:

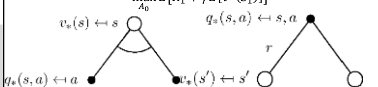
强化学习目标是找到一组时间序列的动作$\{A_0,A_1,\dots\}$使得智能体从S_0出发得到的期望累加奖励最大化

最优价值函数:

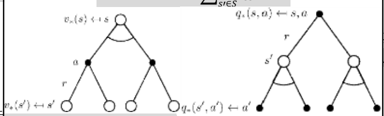
$$v^*(S_0)=E\left[\max_{A_0,A_1,\dots}(R_1+\gamma R_2+\gamma^2R_3+\cdots)\right]$$

最优策略具有如下性质：不论初始状态和初始决策(第一步决策)如何，以第一步决策所形成的阶段和状态作为初始条件来考虑时，余下的决策对余下的问题而言也必构成最优策略

$$v^*(S_0)=E\left[\max_{A_0,A_1,\dots}(R_1+\gamma R_2+\gamma^2R_3+\cdots)\right]$$
$$=\max_{A_0}E\left[R_1+\gamma\max_{A_1,A_2,\dots}(R_2+\gamma R_3+\cdots)\right]$$
$$=\max_{A_0}E\left[R_1+\gamma E[v^*(S_1)]\right]$$



$$v_*(s)=\max_a q_*(s,a)$$
$$q_*(s,a)=R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}v_*(s')$$



关于v_*的 ~~Bellman 最优方程~~:

$$v_*(s)=\max_a\left(R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}v_*(s')\right)$$

关于q_*的：$\boldsymbol{q_*(s,a)=R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}\max_{a'}q_*(s',a')}$

确定性的最优策略:

$$\pi_*(s)=\arg\max_{a\in A}q_*(s,a)$$

求解贝尔曼最优方程需要:

1 求解非线性算子 max

2 模型已知

3 足够的计算空间

$$\pi_*(s)=\arg\max_a\left(\mathcal{R}^a_s+\gamma\sum_{s'}\mathcal{P}^a_{ss'}V_*(s')\right)$$

动态规划 DP

通过把原问题分解为相对简单的子问题来求解复杂问题的方法，适用于:

最优子结构性质：问题的最优解所包含的子问题的解也是最优的，也就是满足最优化原理，将问题划分成子问题。

子问题重叠性质：使用递归算法自顶向下对问题进行求解，每次产生的子问题并不总是新问题。（子问题重复多次出现，保存第一次的解）

马尔可夫决策过程满足以上两个属性：贝尔曼方程具有递归性；价值函数保存和重复利用。

●价值迭代

$$V'(s)=\max_a\left(\mathcal{R}^a_s+\gamma\sum_{s'}\mathcal{P}^a_{ss'}V(s')\right)$$

贝尔曼最优方程的难点在于求解的v_*同时存在于等式左右两边

价值迭代的基本思路:

1 对v_*定义一个估计函数 v

2 将估计函数代入方程右边，等式左边得到一个新函数 v'

3 v'是对 v* 更为准确的估计

4 将 v' 代入右式继续上述过程

1: 初始化一个函数 V_1 (e.g. $V_1(s)=0,\forall s\in\mathcal{S}$)

2: **loop**

3: 根据已知的 V_k 计算一个新的函数

$$V_{k+1}(s)=\max_{a\in\mathcal{A}}\left(\mathcal{R}^a_s+\gamma\sum_{s'\in\mathcal{S}}\mathcal{P}^a_{ss'}V_k(s')\right),\forall s\in\mathcal{S}$$

4: $k\leftarrow k+1$

5: **end loop**

$$v_{k+1}(s)=\max_{a\in A}\left(R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}v_k(s')\right)$$
$$\boldsymbol{v_{k+1}=\max_{a\in A}(R^a+\gamma P^a v_k)}$$

- 价值迭代定义一个以函数作为输入的算子 $\mathcal{T}$, 对给定的函数 V_k 计算新的函数

$$V_{k+1}(s)=[\mathcal{T}(V_k)](s)=\max_a\left(\mathcal{R}^a_s+\gamma\sum_{s'}\mathcal{P}^a_{ss'}V_k(s')\right)$$

- $\mathcal{T}$ 称为价值迭代算子

贝尔曼最优方程:

$$v_*(s)=[T(v_*)](s)=\max_a\left(R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}v_*(s')\right)$$

价值迭代收敛性，$\gamma<1$(此时价值迭代算子是收缩算子)

使用∞-范数比较两个价值函数的不同,即状态集上两个函数最大的差值$\|u-v\|_\infty=\max_{s\in S}|u(s)-v(s)|$

收缩算子：对于任意两个函数f,g，如果两个函数的距离经过一个算子T后能够被缩小，那么这个算子被称为收缩算子

$$\|T(f-g)\|_\infty<\|f-g\|_\infty$$

是否需要证明过程？

$$|[\mathcal{T}(u)](s)-[\mathcal{T}(v)](s)|,\quad\forall s\in\mathcal{S}$$
$$=\left|\max_{a_1}\left(\mathcal{R}^{a_1}+\gamma\sum_{s'\in\mathcal{S}}\mathcal{P}^{a_1}_{ss'}u(s')\right)-\max_{a_2}\left(\mathcal{R}^{a_2}+\gamma\sum_{s'\in\mathcal{S}}\mathcal{P}^{a_2}_{ss'}v(s')\right)\right|$$
$$\leq\max_a\left|\left(\mathcal{R}^a+\gamma\sum_{s'\in\mathcal{S}}\mathcal{P}^a_{ss'}u(s')\right)-\left(\mathcal{R}^a+\gamma\sum_{s'\in\mathcal{S}}\mathcal{P}^a_{ss'}v(s')\right)\right|$$
$$=\gamma\max_a\left|\sum_{s'\in\mathcal{S}}\mathcal{P}^a_{ss'}(u(s')-v(s'))\right|$$
$$\leq\gamma\|u-v\|_\infty\qquad(\gamma<1)$$

$$\Rightarrow\|\mathcal{T}(u-v)\|_\infty<\|u-v\|_\infty$$

从任意初始价值函数 v_1 出发，经过价值迭代计算的新函数 v_{k+1} 与最优价值函数之间的距离

$$\|v_{k+1}-v_*\|_\infty=\|\mathcal{T}v_k-\mathcal{T}v_*\|_\infty$$
$$\leq\gamma\|v_k-v_*\|_\infty$$
$$\vdots$$
$$\leq\gamma^k\|v_1-v_*\|_\infty$$

当$k\to\infty,v_k\to v_*$

策略

智能体可根据给定的策略决定在当前状态下如何采取下一步动作

1 确定性策略：$a_t=\pi(s_t)$

2 随机性策略: $a_t\sim\pi(s_t)$

给定一个策略，就可以对它的好坏进行评估

智能体按照给定的策略执行动作，获得的期望回报称为该策略的价值函数$v_\pi(s)=E_\pi[R_{t+1}+\gamma R_{t+2}+\cdots|S_t=s,A_t\sim\pi(S_t)]$

价值函数越高的策略，性能越好

贝尔曼期望方程

价值函数也可以写成动态规划的形式

$$\boldsymbol{v_\pi=\sum_a\pi(a|s)\left(R^a_s+\gamma P^a_{ss'}v_\pi(s')\right)}$$

对于最优策略来讲

$$\boldsymbol{\pi_*(s)=\arg\max_{a\in A}\left(R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}v_*(s')\right)}$$

它的价值函数就是最优价值函数，满足贝尔曼最优方程

$$\boldsymbol{v_*(s)=\max_a\left(R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}v_*(s')\right)}$$

相比于其它策略，最优策略具有最大的价值函数

$$\boldsymbol{v_*(s)=v_{\pi^*}(s)\geq v_\pi(s),\forall s\epsilon S}$$

在给定策略π的情况下，智能体有限个状态之间的转移可以用矩阵形式表示

$$\boldsymbol{P^\pi=\begin{bmatrix}P^\pi_{S_1,S_1}&\cdots&P^\pi_{S_1,S_n}\\\vdots&\ddots&\vdots\\P^\pi_{S_n,S_1}&\cdots&P^\pi_{S_n,S_n}\end{bmatrix}}$$

其中$\boldsymbol{P^\pi_{S_i,S_j}}$代表了在策略$\pi$下，从状态$\boldsymbol{S_i}$转移到$\boldsymbol{S_j}$的概率

贝尔曼期望方程:$v_\pi=(I-\gamma P^\pi)^{-1}R^\pi$:矩阵很大,矩阵稀疏

$$\boldsymbol{v_\pi=\begin{bmatrix}v_\pi(S_1)\\\dots\\v_\pi(S_n)\end{bmatrix},\qquad R_\pi=\begin{bmatrix}\sum_a\pi(a|S_1)R^a_{S_1}\\\dots\\\sum_a\pi(a|S_n)R^a_{S_n}\end{bmatrix}}$$

如何提升策略————————————

给定一个策略π计算π的价值函数

$$v_\pi(s)=E_\pi[R_{t+1}+\gamma R_{t+2}+\cdots|S_t=s,A_t\sim\pi(S_t)]$$

从v_π提取出贪心策略

$$\pi'=greedy(v_\pi)$$
$$\pi'(s)=\arg\max_{a\in A}\left(R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}v_\pi(s')\right)$$

新的策略继续上述过程,经过多次策略迭代后，策略收敛到$\boldsymbol{\pi_*}$

策略迭代

1: 给定一个初始策略 π_1, $k=1$

2: **loop**

3: **策略评估**: 对当前策略 π_k 计算它的价值函数 v_{π_k}

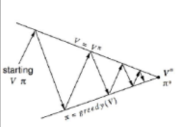
$$v_{\pi_k}(s)=\mathbb{E}[R_{t+1}+\gamma R_{t+2}+\dots|S_t=s,A_t\sim\pi_k(S_t)]$$
$$=\sum_a\pi_k(a|s)\left(\mathcal{R}^a_s+\gamma\mathcal{P}^a_{ss'}v_{\pi_k}(s')\right)$$

4: **策略提升**: 根据 v_{π_k} 提取出新的策略

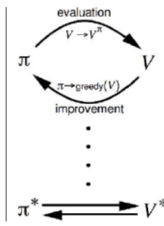
$$\pi_{k+1}(s)=\text{argmax}_{a\in\mathcal{A}}\left(\mathcal{R}^a_s+\gamma\sum_{s'\in\mathcal{S}}\mathcal{P}^a_{ss'}v_{\pi_k}(s')\right)$$

5: $k\leftarrow k+1$

6: **end loop**



策略提升产生的新策略 $\pi'\geq\pi$, 即 $v_{\pi'}(s)\geq v_\pi(s),\forall s\in\mathcal{S}$



考虑一个确定性的策略，$a=\pi(s)$

根据贪心策略对策略进行提升

$$\boldsymbol{\pi'(s)=\arg\max_{a\in A}\left(R^a_s+\gamma\sum_{s'}P^a_{ss'}v_\pi(s')\right)=\arg\max_{a\in A}q_\pi(s,a)}$$

上述过程提升了任意 s 一步的价值

$$\boldsymbol{q_\pi(s,\pi'(s))=\max_{a\in A}q_\pi(s,a)\geq q_\pi(s,\pi(s))=v_\pi(s)}$$

因此新策略的价值函数得到了提升，$\boldsymbol{v_{\pi'}(s)\geq v_\pi(s)}$

$$v_\pi(s)\leq q_\pi(s,\pi'(s))=\mathbb{E}_\pi[R_{t+1}+\gamma v_\pi(S_{t+1})|S_t=s,A_t=\pi'(S_t)]$$
$$\leq\mathbb{E}_\pi[R_{t+1}+\gamma q_\pi(S_{t+1},\pi'(S_{t+1}))|S_t=s,A_t=\pi'(S_t)]$$
$$\leq\mathbb{E}_\pi[R_{t+1}+\gamma R_{t+2}+\gamma^2q_\pi(S_{t+2},\pi'(S_{t+2}))|$$
$$S_t=s,A_t=\pi'(S_t),A_{t+1}=\pi'(S_{t+1})]$$
$$\leq\mathbb{E}_{\pi'}[R_{t+1}+\gamma R_{t+2}+\dots|S_t=s]=v_{\pi'}(s)$$

因为$\boldsymbol{v_{\pi1}\leq v_{\pi2}\leq\cdots\leq v_{\pi k}\leq v_*}$单调递增有上界，所以收敛

当策略不再变化时满足贝尔曼最优方程

价值迭代 VS 策略迭代

●价值迭代

$$\boldsymbol{v_{k+1}(s)=\max_a\left(R^a_s+\gamma P^a_{ss'}v_k(s')\right)}$$

只在收敛得到$\boldsymbol{v_*}$后计算$\boldsymbol{\pi_*}$，中间过程不产生策略

涉及赋值操作，计算量小,$O(|S|^2|A|)$，迭代次数多

●策略迭代

$$\boldsymbol{v_\pi(s)=\sum_a\pi(a|s)\left(R^a_s+\gamma P^a_{ss'}v_\pi(s')\right)}$$
$$\boldsymbol{\pi'(s)=\arg\max_{a\in A}\left(R^a_s+\gamma\sum_{s'}P^a_{ss'}v_\pi(s')\right)}$$

每次迭代开始时给定一个$\boldsymbol{\pi}$，结束时产生一个新$\boldsymbol{\pi'}$,迭代次数少

求解方程，计算量大，矩阵求逆$O(|S|^3)$，策略提升,$O(|S|^2|A|)$

动态规划方法

$$v_*(s)=\max_a\left(R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}v_*(s')\right)$$

动作规划: 价值迭代/策略迭代

通过迭代有效求解原始问题中非线性的 max 算子

依赖系统模型，R，P

足够的计算空间记录每个状态的函数值

价值迭代$\boldsymbol{v_*(s)}$

策略迭代$\boldsymbol{v_\pi(s),\pi(s)}$

状态-动作价值函数

$$q_*(s,a)=R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}\max_{a'}q_*(s',a')$$

●基于 Q 函数的价值迭代

初始化q_1, (e.g. q1(s; a) = 0; ∀s ∈ S; ∀a ∈ A)

迭代计算新的 Q 函数

$$\boldsymbol{q_{k+1}(s,a)=R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}\max_{a'}q_k(s',a')}$$

●基于 Q 函数的策略迭代

给定一个策略π_1

策略评估：计算 k 的状态-动作价值函数

$$q_{\pi k}(s,a)=R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}\sum_{a'\in A}\pi(a'|s')\,q_{\pi k}(s',a')$$

共同的不足：对模型依赖, R;P

要么是 MDP 问题的模型已知

要么智能体对环境建模

无模型的策略评估方法:

蒙特卡洛方法；时间差分方法

策略提升：根据$q_{\pi k}$提取出新的策略

$$\boldsymbol{\pi_{k+1}(s)=\arg\max_{a\epsilon A}}q_{\pi k}(s,a)$$

迭代策略评估:

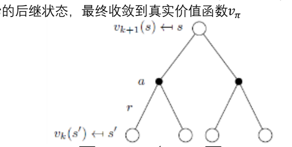
问题:对给定策略$\boldsymbol{\pi}$计算它的价值函数

方法:基于贝尔曼期望方程迭代更新价值函数

初始化一个价值函数v_1, $v_1\to v_2\to\cdots\to v_\pi$

同步更新:

在每次迭代中，对所有的状态$s\in S$，根据$v_k(s')$更新$v_{k+1}(s)$，其中s'是s的后继状态，最终收敛到真实价值函数v_π



$$v_{k+1}(s)=\sum_{a\in A}\pi(a|s)\left(R^a_s+\gamma\sum_{s'\in S}P^a_{ss'}v_k(s')\right)$$

贝尔曼期望更新算子T^π，$T^\pi(v)=R^\pi+\gamma P^\pi v$

该算子是γ-收缩的，即经过该算子两个价值函数的距离变为原来的γ倍

$$\|\mathcal{T}^\pi(u)-\mathcal{T}^\pi(v)\|_\infty=\|(\mathcal{R}^\pi+\gamma\mathcal{P}^\pi u)-(\mathcal{R}^\pi+\gamma\mathcal{P}^\pi v)\|_\infty$$
$$=\|\gamma\mathcal{P}^\pi(u-v)\|_\infty$$
$$\leq\|\gamma\mathcal{P}^\pi\|u-v\|_\infty\|_\infty$$
$$\leq\gamma\|u-v\|_\infty$$

所以迭代策略评估收敛到v_π

●异步动态规划:就地动态规划;优先动态规划;实时动态规划

蒙特卡洛方法 Monte-Carlo, MC

MC 方法直接从经历过的事件中学习

无模型方法: 不需要 MDP 的转移/奖励函数

从完整的事件中学习: 没有自举

基于最简单的思想: 价值= 平均回报

运行 MC 方法通常要求:

所有事件都到达终止状态$\{S_1,A_1,\dots,S_{terminal}\}$

或者事件的时序足够长$\{S_1,A_1,\dots,S_T\}$; T ≫ 1

蒙特卡洛策略评估

目标：从策略π产生的事件中学习v_π：$S_1,A_1,R_2,\dots,S_k\sim\pi$

回忆下回报的定义是所有折扣奖励和

$$\boldsymbol{G_t=R_{t+1}+\gamma R_{t+2}+\cdots\gamma^{T-1}R_t}$$

回忆下价值函数的定义是回报的期望$v_\pi(s)=E_\pi[\boldsymbol{G_t}|S_t=s]$

蒙特卡洛策略评估方法使用回报的经验均值作为回报的期望

基于样本更新:

接下来的课程将主要使用智能体从环境获得的样本进行更新，使用样本的奖励和状态转移$\langle S,A,,R,S'\rangle$而不是奖励函数 R 和转移函数 P。好处包括:

无模型：不需要预前知道 MDP 的模型信息

通过样本避免整个状态空间的维数灾问题

每次更新的计算量是固定，与后继的状态空间 n = |S| 无关

首次经过的 MC 策略评估

为了评估状态 s, 假设在一次事件中第一次经过状态 s 的时刻为 t, 计数加一 N(s) ← N(s) + 1，全部回报相加 S(s) ← S(s)+Gt，估计的价值等于回报均值 V(s) = S(s)/N(s)，根据大数定理，当 N(s) → ∞ 时, V(s) → $v_\pi(s)$

每次经过的 MC 策略评估

为了评估状态 s，对一次事件中每一次经过状态 s 的时刻 t，计数加一 N(s) ← N(s)+1，全部回报相加 S(s) ← S(s) +Gt，估计的价值等于回报均值 V(s) = S(s)/N(s)，同样当 N(s) → ∞ 时, V(s) → $v_\pi(s)$

增量计算均值

一个序列$x_1,x_2,\dots$的均值$\mu_1,\mu_2,\dots$可以通过增量的方式计算

$$\mu_k=\frac{1}{k}\sum_{j=1}^k x_j=\frac{1}{k}\left(x_k+\sum_{j=1}^{k-1}x_j\right)=\frac{1}{k}(x_k+(k-1)\mu_{k-1})$$
$$=\mu_{k-1}+\frac{1}{k}(x_k-\mu_{k-1})$$

增量式的 MC 更新

根据事件$S_1,A_1,R_1,\dots,S_T$对$V(s)$增量式地更新

对状态S_t，回报是G_t

$$N(S_t)\leftarrow N(S_t)+1$$
$$V(S_t)\leftarrow V(S_t)+\frac{1}{N(S_t)}(G_t-V(S_t))$$

如果环境是动态、不断变化的，我们更希望是能够跟踪当前不断变化的均值，遗忘掉很久之前的事件

$$V(S_t)\leftarrow V(S_t)+\alpha(G_t-V(S_t))$$

时间差分学习 Temporal Difference (TD) Learning

TD 方法直接从智能体经历的事件中学习

无模型的：不知道 MDP 问题的转移和奖励函数

可以从非完整的事件中学习，借助自举法

根据一个猜测值更新另一个猜测值

MC 和 TD

目标: 根据智能体在策略π作用下产生的经历在线学习v_π

增量式的 MC 方法:调整价值$V(S_t)$向真实的回报G_t逼近

$$V(S_t)\leftarrow V(S_t)+\alpha(G_t-V(S_t))$$

最简单形式的 TD 学习算法: TD(0)

调整价值$V(S_t)$向估计的回报$\boldsymbol{R_{t+1}}+\gamma V(S_{t+1})$逼近

$$V(S_t)\leftarrow V(S_t)+\alpha(\boldsymbol{R_{t+1}}+\gamma V(S_{t+1})-V(S_t))$$

TD 目标：$\boldsymbol{R_{t+1}}+\gamma V(S_{t+1})$

TD 误差：$\boldsymbol{\delta_t=R_{t+1}}+\gamma V(S_{t+1})-V(S_t)$

TD 学习算法:

1: 给定策略 π, 初始状态分布 D_0, $V(s)=0,\forall s\in\mathcal{S}$, 学习率 α

2: **for** $t=1,2,\dots$ **do**

3: **if** $S_t=S_{terminal}$ 或 S_t 未初始化 **then**

4: 初始化 $S_t\sim D_0$

5: **end if**

6: 根据 S_t 采样 $A_t\sim\pi(S_t)$, 智能体执行 A_t 后观察 $(S_t,A_t,R_{t+1},S_{t+1})$

7: 更新 $V(S_t)\leftarrow V(S_t)+\alpha(R_{t+1}+\gamma V(S_{t+1})-V(S_t))$

8: **end for**

MC 和 TD 对比

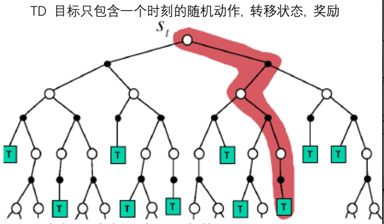
MC 是高方差，零偏差：

需要完整的事件序列计算出回报后学习
好的收敛性(即使是使用逼近器也能保证收敛性);
与价值函数初始值无关;
原理简单，使用方便;
要求事件达到终止状态或序列足够长
能够利用马尔可夫性，因此在马尔可夫环境下更有效

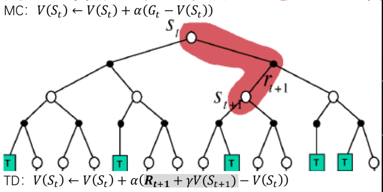
TD 是低方差，有偏差：

可以在智能体运行过程中的每一步在线学习
可以从不完整的事件序列中学习
通常情况是比 MC 更有效
TD(0) 能够收敛到$v_\pi(s)$
(但是与逼近器结合后并不一定收敛)
受价值函数初始值影响
不能利用马尔可夫性，因此在非马尔可夫环境下更有效

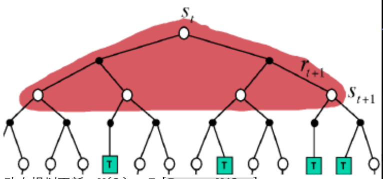
偏差/方差的权衡:
回报$G_t = R_{t+1} + \gamma R_{t+2} + \cdots \gamma^{T-1}R_t$是$v_\pi(S_t)$的无偏估计
真实的 TD 目标$R_{t+1} + \gamma v_\pi(S_{t+1})$是$v_\pi(S_t)$的无偏估计
TD 目标$R_{t+1} + \gamma V(S_{t+1})$是$v_\pi(S_t)$的有偏估计
TD 目标比回报的方差要小很多
回报的计算包含整个序列中的随机动作，转移状态，奖励
TD 目标只包含一个时刻的随机动作，转移状态，奖励



MC: $V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$



TD: $V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$



动态规划更新: $V(S_t) \leftarrow E_\pi[R_{t+1} + \gamma V(S_{t+1})]$
自举法: 更新时包含一个猜测量
MC 不使用自举　DP 使用自举　TD 使用自举
采样法: 使用采样的数据计算期望
MC 使用采样　DP 不使用采样　TD 使用采样
等价模型:
MC 收敛结果对应最小二乘误差，最佳匹配观测的回报

$$\sum_{k=1}^{K}\sum_{t=1}^{T_k}\left(G_t^k - V(s_t^k)\right)^2$$

TD(0) 收敛结果对应最大似然马尔可夫模型$\langle S, A, \hat{P}, \hat{R}, \gamma\rangle$

$$\hat{P}^a_{s,s'} = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k} 1(s_t^k, a_t^k, s_{t+1}^k = s, a, s')$$

$$\hat{R}^a_{s,s'} = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k} 1(s_t^k, a_t^k = s, a) r_t^k$$

$n=1$ (TD) $G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1})$
$n=2$ $G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})$
$\vdots$
$n=\infty$ (MC) $G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_T$

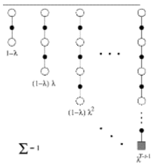
n-步的回报:

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

n-步时间差分学习

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t^{(n)} - V(S_t)\right)$$

TD(λ), λ-回报



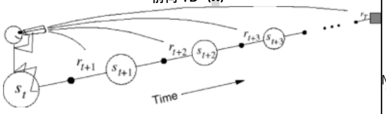
λ-回报G_t^λ将所有的 n-步回报$G_t^{(n)}$整合在一起
对每项使用权重$(1-\lambda)\lambda^{n-1}$

$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^{(n)}$$

前向更新的 TD(λ)形式

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t^\lambda - V(S_t)\right)$$

前向 TD（λ）

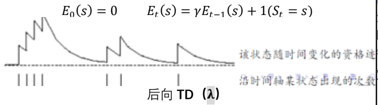


更新价值函数向**λ** -回报逼近，面向未来时刻计算G_t^λ，与 MC 一样要求完整的事件序列

资格迹 Eligibility Traces

可信度: 警铃和闪光是否和高压电直接相关?
受频率启发: 对发生次数最多的状态给予最高的可信度
受近因启发: 对时间上最近发生的状态给予最高的可信度
资格迹将两种方式结合

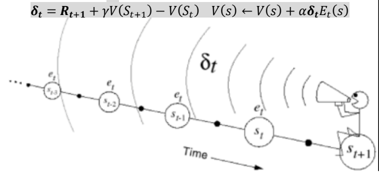
$$E_0(s) = 0 \qquad E_t(s) = \gamma E_{t-1}(s) + 1(S_t = s)$$



后向 TD（λ）

为每个状态s记录它的资格迹，更新s的价值$V(s)$
更新量与 TD 误差δ_t和资格迹$E_t(s)$呈正比

$$\boldsymbol{\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)} \quad V(s) \leftarrow V(s) + \alpha\boldsymbol{\delta_t}E_t(s)$$



TD(λ)和 TD(0)

当**λ**= 0 时，只有当前状态被更新

$$E_t(s) = 1(S_t = s) \qquad V(s) \leftarrow V(s) + \alpha\boldsymbol{\delta_t}E_t(s)$$

等同于 TD(0) 方法$V(s) \leftarrow V(s) + \alpha\boldsymbol{\delta_t}$

TD(λ)和 MC

当**λ**= 1 时，可信度被推迟到事件结束，考虑每个事件都到达终止状态的环境，使用离线更新，对每个事件序列，TD(1) 的所有更新和 MC 的所有更新是一致的
当使用离散更新时，前向和后向 **TD(λ)**所有的更新是相同的

$$\sum_{t=1}^{T}\alpha\delta_t E_t(s) = \sum_{t=1}^{T}\alpha\left(G_t^\lambda - V(S_t)\right)1(S_t = s)$$

MC 和 TD(1)

考虑在一次事件中 s 在时刻 k 时刻被唯一访问一次
该状态的 TD(1) 资格迹会在被访问之后随时间衰减

$$E_t(s) = \gamma E_{t-1}(s) + 1(S_t = s) = \begin{cases} 0 & if\ t < k \\ \gamma^{t-k} & if\ t \ge k \end{cases}$$

TD(1) 在更新时会在线累计误差

$$\sum_{t=1}^{T-1}\alpha\delta_t E_t(s) = \alpha\sum_{t=k}^{T-1}\gamma^{t-k}\delta_t = (G_k - V(S_k))$$

在事件结束时累计误差等于

$$\delta_k + \gamma\delta_{k+1} + \gamma^2\delta_{k+2} + \cdots + \gamma^{T-1-k}\delta_{T-1}$$

当**λ**= 1 时, TD 误差可以变换成 MC 误差

$$\boldsymbol{\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)}$$

$$\delta_k + \gamma\delta_{k+1} + \gamma^2\delta_{k+2} + \cdots + \gamma^{T-1-k}\delta_{T-1} = \boldsymbol{G_t} - V(S_t)$$

TD(λ) 和 TD(1)

TD(1) 大致等价于每次经过的 MC 方法，误差会随在线运行逐步累积，如果价值函数只在事件结束时离线更新，那么所有的更新量等同于 MC 方法

前向和后向 TD(λ)

考虑在一次事件中 s 在时刻 k 时刻被唯一访问一次
该状态的 **TD(λ)**资格迹会在被访问之后随时间衰减

$$E_t(s) = \gamma\lambda E_{t-1}(s) + 1(S_t = s) = \begin{cases} 0 & if\ t < k \\ (\lambda\gamma)^{t-k} & if\ t \ge k \end{cases}$$

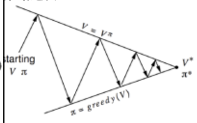
后向 TD(λ)在更新时会在线累计误差

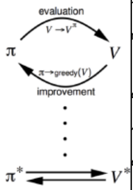
$$\sum_{t=1}^{T-1}\alpha\delta_t E_t(s) = \alpha\sum_{t=k}^{T-1}(\lambda\gamma)^{t-k}\delta_t = \left(\boldsymbol{G_k^\lambda} - V(S_k)\right)$$

在事件结束时总共累计误差等于**λ** -回报
如果 s 被多次访问，$E_t(s)$也会累积多次的误差
前向提供理论原理，后向给出算法实现: 在线学习; 每一时刻更新; 可以适用事件序列中的一小段，非完整序列
目标: 获得最优策略，基于模型: 动态规划、价值迭代、策略迭代
目标: 获得策略的价值函数，基于样本: 迭代策略评估.蒙特卡洛方法.时间差分学习.TD(λ) .
我们想获得最优策略，但是 MDP 的模型未知，或者问题空间太大只能考虑和智能体最相关的状态: 无模型方法; 使用智能体的经验样本; 最优化价值函数
上节课:无模型预测，估计一个模型未知 MDP 问题的价值函数
这节课:无模型控制，最优化一个模型未知 MDP 问题的价值函数

在策略(on-policy) 和离策略(off-policy) 学习

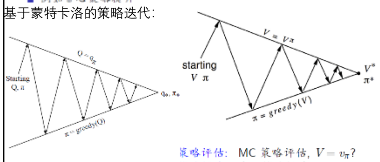
在策略学习: 根据策略π 产生的样本来学习关于π 的相关知识
离策略学习: 根据另一策略μ产生的样本来学习关于 π 的相关知识
策略迭代:





- 策略评估: 估计 v_π
  - 例如迭代策略评估
- 策略提升: 生成新的 $\pi' \ge \pi$
  - 例如贪心策略提升

基于蒙特卡洛的策略迭代:



策略评估: MC 策略评估, $V = v_\pi$?
策略提升: 贪心策略提升
策略评估: MC 策略评估, $Q = q_\pi$
策略提升: 贪心策略提升

动作-价值函数实现无模型的策略迭代:
基于 V(s) 进行贪心策略提升需要 MDP 的模型信息

$$\pi'(s) = \underset{a\in A}{\arg\max}\left(R_s^a + \gamma P^a_{ss'}V(s')\right)$$

基于 Q(s, a) 进行贪心策略提升是无模型的

$$\pi'(s) = \underset{a\in A}{\arg\max}\, Q(s,a)$$

ϵ-贪心策略

最简单实现连续探索的方法，所有的 m 个动作都有非零概率被选中执行，以 1−ϵ 概率选择贪心动作，以 ϵ 概率随机选择一个动作

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & if\ a^* = \underset{a\in A}{\arg\max}\, Q(s,a) \\ \epsilon/m & otherwise \end{cases}$$

给定任意 ϵ-贪心策略 π，根据q_π构造出的新的 ϵ-贪心策略π'具有更好的性能，即$v_{\pi'}(s) \ge v_\pi(s)$

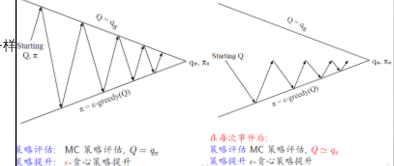
$$q_\pi(s, \pi'(s)) = \sum_{a\in A}\pi'(a|s)q_\pi(s,a)$$

$$= \epsilon/m\sum_{a\in A}q_\pi(s,a) + (1-\epsilon)\underset{a\in A}{\arg\max}\, q_\pi(s,a)$$

$$\ge \epsilon/m\sum_{a\in A}q_\pi(s,a) + (1-\epsilon)\sum_{a\in A}\frac{\pi(a|s) - \epsilon/m}{1-\epsilon}q_\pi(s,a)$$

$$= \sum_{a\in A}\pi(a|s)q_\pi(s,a) = v_\pi(s) \to v_{\pi'}(s) \ge v_\pi(s)$$

MC 策略迭代:　　MC 控制



策略评估: MC 策略评估, $Q = q_\pi$
策略提升: ϵ-贪心策略提升
在每次事件后:
策略评估 MC 策略评估, $Q \approx q_\pi$
策略提升: ϵ-贪心策略提升

无限探索，无穷时刻收敛为贪心策略 (Greedy in the Limit with Infinite Exploration, GLIE)
智能体能无限次数地探索所有的状态-动作对: $\lim_{k\to\infty}N_k(s,a) = \infty$
策略在无穷时刻收敛到贪心策略

$$\lim_{k\to\infty}\pi_k(a|s) = 1\left(a = \underset{a\in A}{\arg\max}\, Q_k(s,a')\right)$$

ϵ-贪心策略的探索率ϵ能随时间衰减到零$\epsilon_k = \frac{1}{k}$，则它就满足 GLIE

GLIE 蒙特卡洛控制

使用策略π采集第k次事件$\{S_1, A_1, R_2, \dots, S_T\} \sim \pi$
对事件中的每个状态S_t和动作A_t

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}\left(G_t - Q(S_t, A_t)\right)$$

基于新得到的动作-价值函数对策略进行提升

$$\epsilon \leftarrow \frac{1}{k} \qquad \pi \leftarrow greedy(Q)$$

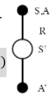
GLIE 蒙特卡洛控制能够收敛到最优动作-价值函数,

$$Q(s,a) \to q_*(s,a)$$

MC vs TD 控制

TD 学习和 MC 方法相比有如下优势:
低方差，在线学习，无完整事件序列
在控制方法中使用 TD 替代 MC:
基于 TD 方法训练$Q(S,A)$，使用 ϵ-贪心策略进行策略提升，每个时刻都对函数更新

基于 Sarsa 方法更新动作-价值函数



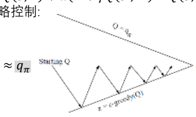
动作-价值函数的贝尔曼期望方程:

$$q_\pi(s,a) = R_s^a + \gamma\sum_{s'\in S}P^a_{ss'}\sum_{a'\in A}\pi(a'|s')q_\pi(s',a')$$

在某一时刻智能体观察到的一段事件(S, A, R, S', A')
基于样本的 TD 更新:

$$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma Q(S',A') - Q(S,A))$$

基于 Sarsa 的在策略控制:
在每一时刻:
策略评估:
使用 Sarsa 方法，$Q \approx q_\pi$
策略提升:
ϵ-贪心策略提升



算法:

任意初始化 $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$，令 $Q(S_{terminal}, \cdot) = 0$
repeat {在每次事件中:}
　初始化 S
　根据从 Q 提取的策略 (例如 ϵ-贪心策略) 对 S 选择动作 A
　repeat {对事件中的每一时刻}
　　执行动作 A，观察 R, S'
　　根据从 Q 提取的策略 (例如 ϵ-贪心策略) 对 S' 选择动作 A'
　　$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$
　　$S \leftarrow S'$, $A \leftarrow A'$
　until S 是终止状态
until

Sarsa 的收敛性:
当如下条件满足时 Sarsa 算法是收敛到最优动作-价值函数:
策略序列$\pi_t(a|s)$是 GLIE 的
更新步长α_t满足 Robbins-Monro 序列要求

$$\sum_{t=1}^{\infty}\alpha_t = \infty \qquad \sum_{t=1}^{\infty}{\alpha_t}^2 < \infty)$$

Sarsa vs MC

注意: 蒙特卡洛方法并不能直接应用在这个问题上
并不是所有的策略都能到达终止状态，如果学到的某一策略使智能体待在原地不动，MC 在基于该策略对事件采样时永远都无法结束
Sarsa 算法没有这样的问题
每一步都在学习，很快就会发现这样的策略性能很差，然后转向其它策略

n-步 Sarsa

考虑 n-步的回报，n = 1, 2,∞:
$n=1$ (Sarsa) $q_t^{(1)} = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$
$n=2$ $q_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}, A_{t+2})$
$\vdots$
$n=\infty$ (MC) $q_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1}R_T$

定义 n-步的 Q-回报

$$q_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1}R_{t+n} + \gamma^n Q(S_{t+n}, A_{t+n})$$

n-步 Sarsa 算法更新 Q(s, a)，向 n-步 Q-回报逼近

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(q_t^{(n)} - Q(S_t, A_t)\right)$$

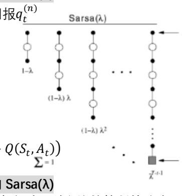
前向 Sarsa(λ)

q^λ回报包含了所有的n-步Q-回报$q_t^{(n)}$
使用$(1-\lambda)\lambda^{n-1}$对回报加权

$$q_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}q_t^{(n)}$$

前向 Sarsa(λ) 算法

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(q_t^\lambda - Q(S_t, A_t)\right)$$



后向 Sarsa(λ)

和 TD(λ) 算法一样，我们将资格迹引入到在线的控制算法中
但是 Sarsa(λ) 算法不同的地方在于为每个状态- 动作对都记录资格迹(之前是只对状态记录)

$$E_0(s,a) = 0 \qquad E_t(s,a) = \gamma\lambda E_{t-1}(s,a) + 1(S_t = s, A_t = a)$$

在每一时刻会对每个状态s和动作a的$Q(s,a)$更新
更新量与 TD 误差$\boldsymbol{\delta_t}$和资格迹$E_t(s,a)$呈正比

$$\boldsymbol{\delta_t = R_{t+1}} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha\boldsymbol{\delta_t}E_t(s,a)$$

任意初始化 $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$
repeat {在每次事件中:}
　令 $E(s,a) = 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$
　初始化 S, A
　repeat {对事件中的每一步}
　　执行动作 A，观察 R, S'
　　根据从 Q 提取的策略 (例如 ϵ-贪心策略) 对 S' 选择动作 A'
　　$\delta \leftarrow R + \gamma Q(S',A') - Q(S,A)$
　　$E(S,A) \leftarrow E(S,A) + 1$
　　for all $s \in \mathcal{S}, a \in \mathcal{A}$ **do**
　　　$Q(S,A) \leftarrow Q(S,A) + \alpha\delta E(s,a)$
　　　$E(s,a) \leftarrow \gamma\lambda E(s,a)$
　　end for
　　$S \leftarrow S'$, $A \leftarrow A'$
　until S 是终止状态
until

一步 Sarsa vs Sarsa(λ)

一步 Sarsa 只对最终导致高奖励的最后一步动作强化它的价值
资格迹方法能够对事件中的多个动作强化它们的价值
强化的幅度(箭头大小) 随着离奖励步数的增加而衰减
衰减率等于γλ
这里γ = 1, λ = 0.9

离策略学习

想要对目标策略$\pi(a|s)$进行评估，计算$v_\pi(s)$或$q_\pi(s,a)$，但是智能体实际的行为策略是$\mu(a|s)$: $\{S_1, A_1, R_2, \dots, S_T\} \sim \mu$
为什么要考虑这种情况?
有时智能体需要观察人类或别的智能体的行为去学习
有时需要重复利用旧策略$\pi_1, \pi_2, \dots, \pi_{t-1}$产生的经验去学习
执行探索性的策略去学习最优策略
执行单一的策略去学习多个策略

Q-学习

考虑基于动作-价值$Q(s,a)$的离策略学习，不再使用重要性采样
智能体下一时刻执行的动作是由行为策略产生$A_{t+1} \sim \mu(\cdot|S_t)$
但是学习算法考虑的是由另一个目标策略产生的后继动作

$$A' \sim \pi(\cdot|S_t)$$

更新$Q(S_t, A_t)$向另一个后继动作的价值逼近

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t))$$

Q-学习实现离策略控制:
现在允许行为策略和目标策略都能提升
目标策略π是由$Q(s,a)$生成的贪心策略

$$\pi(S_{t+1}) = \underset{a'}{\arg\max}\, Q(S_{t+1}, a')$$

行为策略μ是由$Q(s,a)$生成的例如 ϵ-贪心策略
那么 Q-学习的目标就是

$$R_{t+1} + \gamma Q(S_{t+1}, A') = R_{t+1} + \gamma \underset{a'}{\arg\max}\, Q(S_{t+1}, a')$$

$$= R_{t+1} + \gamma\max_{a'}Q(S_{t+1}, a')$$

Q-学习控制算法

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R + \gamma\max_{a'}Q(S', a') - Q(S, A)\right)$$

Q 学习控制能够收敛到最优动作-价值函数,

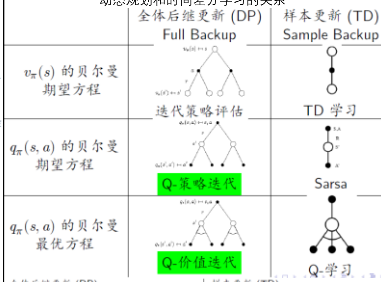
Q-学习算法实现离策略控制:

任意初始化 $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$，令 $Q(S_{terminal}, \cdot) = 0$
repeat {在每次事件中:}
　初始化 S
　repeat {对事件中的每一时刻}
　　根据从 Q 提取的策略 (例如 ϵ-贪心策略) 对 S 选择动作 A
　　执行动作 A，观察 R, S'
　　$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma\max_a Q(S',a) - Q(S,A)]$
　　$S \leftarrow S'$
　until S 是终止状态
until

Sarsa vs Q-学习

Q-学习正确地学到了最优路径，即沿着悬崖的边缘行走，但是由于使用的 ϵ-贪心策略会采取一定概率的随机动作，有时智能体跌入悬崖
Sarsa 学到的是安全路径,在学习过程中考虑了随机探索动作的影响，即使 Sarsa 学到的安全路径比 Q-学习的最优路径行走步数要长，但是每次获得的奖励和却比 Q-学习的高

动态规划和时间差分学习的关系



全体后继更新 (DP)	样本更新 (TD)
迭代策略评估 $V(s) \leftarrow \mathbb{E}[R + \gamma V(s')\|s]$	TD 学习 $V(S) \xleftarrow{\alpha} R + \gamma V(S')$
Q-策略迭代 $Q(s,a) \leftarrow \mathbb{E}[R + \gamma Q(s',a')\|s,a]$	Sarsa $Q(S,A) \xleftarrow{\alpha} R + \gamma Q(S',A')$
Q-价值迭代 $Q(s,a) \leftarrow \mathbb{E}[R + \gamma\max_{a'\in\mathcal{A}}Q(s',a')\|s,a]$	Q- 学习 $Q(S,A) \xleftarrow{\alpha} R + \gamma\max_{a'\in\mathcal{A}}Q(S',a')$

其中，$x \xleftarrow{\alpha} y \equiv x \leftarrow x + \alpha(y - x)$

探索(exploration) 与利用(exploitation)

在线决策过程存在的一个关键问题是:
利用: 根据当前的信息做出最佳的决策
探索: 采样更多的信息
想要做出长期的最佳决策有时需要牺牲当前短期的部分利益
收集更多的信息以而做出整体最佳的决策

Q-学习

如果 Q-学习没有探索，即 ϵ = 0，学习结果容易陷入局部最优解，无法找到真正最优策略，如果ϵ 很大也不好
1 状态，动作空间很大时，想要探索整个空间是费时费力的，对在线学习过程不利
2 随机的动作会降低智能体的实际回报

非均匀探索概率

ϵ-贪心策略除了最优动作，其它动作以等概率随机选择
假如: 有两个动作看起来不错，而其它的动作完全不可行
那么比较合理的探索方式是选择那些看起来不错的动作，从不错的动作中选出最优的动作，而不是将精力浪费在看起来不可行的动作，**根据动作-价值决定动作被选中的概率**

波尔兹曼探索, Boltzmann exploration

$$\pi(a|s) = \frac{e^{Q(s,a)/T}}{\sum_{a'\in A}e^{Q(s,a')/T}}$$

a被选中的概率与$e^{Q(s,a)/T}$呈正比($Q(s,a) \ge 0$, ∀s, a)
温度系数 T > 0 决定了策略的随机程度
如果 T 很大，所有动作几乎以等概率被选择(探索)
如果 T 很小，高价值的动作更容易被选中(利用)
极限情况 T → 0，只选择最优动作

求解非线性算子 max
动态规划: 价值迭代/策略迭代
模型已知
预测(prediction): 蒙特卡洛方法，时间差分学习, TD(λ)
控制(control): Sarsa, Sarsa(λ), Q-学习

算法	查表法	线性逼近	非线性逼近
MC 控制	✓	(✓)	×
Sarsa	✓	(✓)	×
Q 学习	✓	×	×
LSPI	✓	(✓)	-

(√)代表算法会在最优价值函数附近振荡

最小二乘策略迭代:

策略评估: 使用最小二乘学习策略的 Q 函数
策略提升: 贪心策略提升

在/离策略	算法	查表法	线性逼近	非线性逼近
在策略	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	✓	×
	LSTD	✓	✓	-
离策略	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	×	×
	LSTD	✓	✓	-

批处理强化学习, Batch RL：

梯度下降法简单，易实现，但是对样本的利用率不足
批处理的方法能够基于智能体的经验("训练数据")，试着找到最合适的价值函数

使用逼近器的好处

使用较少数量的参数就能表达复杂的函数(计算复杂度)
对一个权重的调整就可以影响到很多的点(泛化能力)
多种特征表示和逼近结构(多样性)
可微的函数逼近器，例如: 特征函数的线性组合，神经网络，决策树，最近邻，傅立叶/小波基函数