

Review of Reinforcement Learning

概述

强化学习：让智能体在**动态环境**中**学习动作**和**达成目标**。

智能：在**动态、开放环境**中**达到目标**的能力

智能体：有**目标**，**感知-动力**闭环的系统

学习：利用经验E在任务T上获得性能P的改善，则关于T和P，对E进行了学习

与监督学习的区别：监督学习使用带标注的训练集使智能体具备可泛化的决策能力。但不可能在所有情境下都获得既正确又有代表性的动作示例，智能体必须从自身经验中学习。

与无监督学习的区别：

无监督学习寻找未标注数据的隐含结构，强化学习最大化收益。

强化学习的问题：智能体如何高效地从与环境交互中学习，以获得自适应性？

特点：

1. 交互、试错学习
2. 无监督信号，只有奖励信号
3. 奖励不一定实时

从交互信息中发现因果关系、动作的后果，以及达到目标的方式。

环境的模型：智能体可以用来预测环境对其动作的反应的任何事物

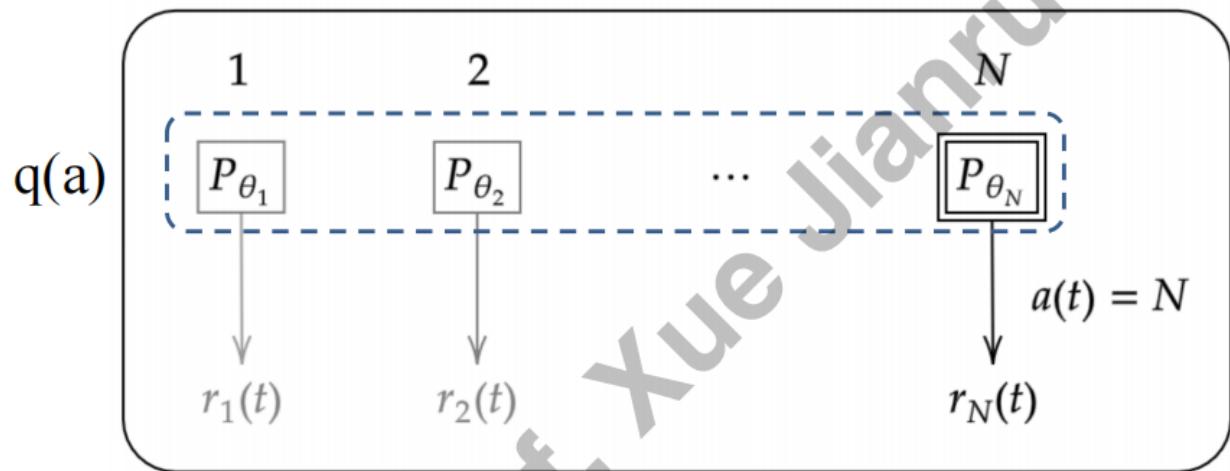
赌博机

探索/利用（开发）

问题描述

MAB (Multi-armed Bandit) 问题描述

For t in $\{1, \dots, T\}$



$$\text{Goal: Find } \arg \max_a E \sum_{t=1}^T (r_{a(t)}(t))$$

■ 推广—在线内容推荐、广告推送、选股

找到平均收益最大的臂

动作-价值方法

■ 动作的期望价值

$$q_*(a) \doteq \mathbb{E}[R_t | A_t = a], \quad \forall a \in \{1, \dots, k\}$$

动作选择: $A_t = \arg \max_a q_*(a)$

■ 动作-价值 $Q_t(a)$ 的估计: 采样平均法

$$Q_t(a) = \frac{\text{t时刻前通过执行动作a得到的收益总和}}{\text{t时刻前执行动作a的次数}}$$

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i \mathbf{I}(A_i=a)}{\sum_{i=1}^{t-1} \mathbf{I}(A_i=a)}$$

The number of times action a
has been taken by time t

$$\lim_{N_t(a) \rightarrow \infty} Q_t(a) = q_*(a)$$

$q_*(a)$ 是实际动作价值, $Q_t(a)$ 是t时刻对动作价值的估计。

采样平均法

用动作的平均收益估计期望收益

平衡探索与利用: ϵ -贪心

贪心策略一直在守成 (利用), 不断选择获益最高的动作

ϵ -贪心: 大概率贪心, 以小概率 ϵ 随机选择动作

ϵ -贪心策略

- 选择贪婪动作时，一直在守成
- 以 ϵ 概率不选择贪婪动作，大概率贪心，小概率随机选择动作

A simple bandit algorithm

Initialize, for $a = 1$ to k :

$$\begin{aligned} Q(a) &\leftarrow 0 \\ N(a) &\leftarrow 0 \end{aligned}$$

Repeat forever:

$$\begin{aligned} A &\leftarrow \begin{cases} \arg \max_a Q(a) & \text{with probability } 1 - \epsilon \\ \text{a random action} & \text{with probability } \epsilon \end{cases} \quad (\text{breaking ties randomly}) \\ R &\leftarrow \text{bandit}(A) \\ N(A) &\leftarrow N(A) + 1 \\ Q(A) &\leftarrow Q(A) + \frac{1}{N(A)} [R - Q(A)] \end{aligned}$$

采样平均法的增量形式

采样平均法的增量形式

平均动作-价值估计: $Q = (Q \times (N-1) + R) / N$, 把Q单独拿出来

$$Q = Q + (R - Q) / N$$

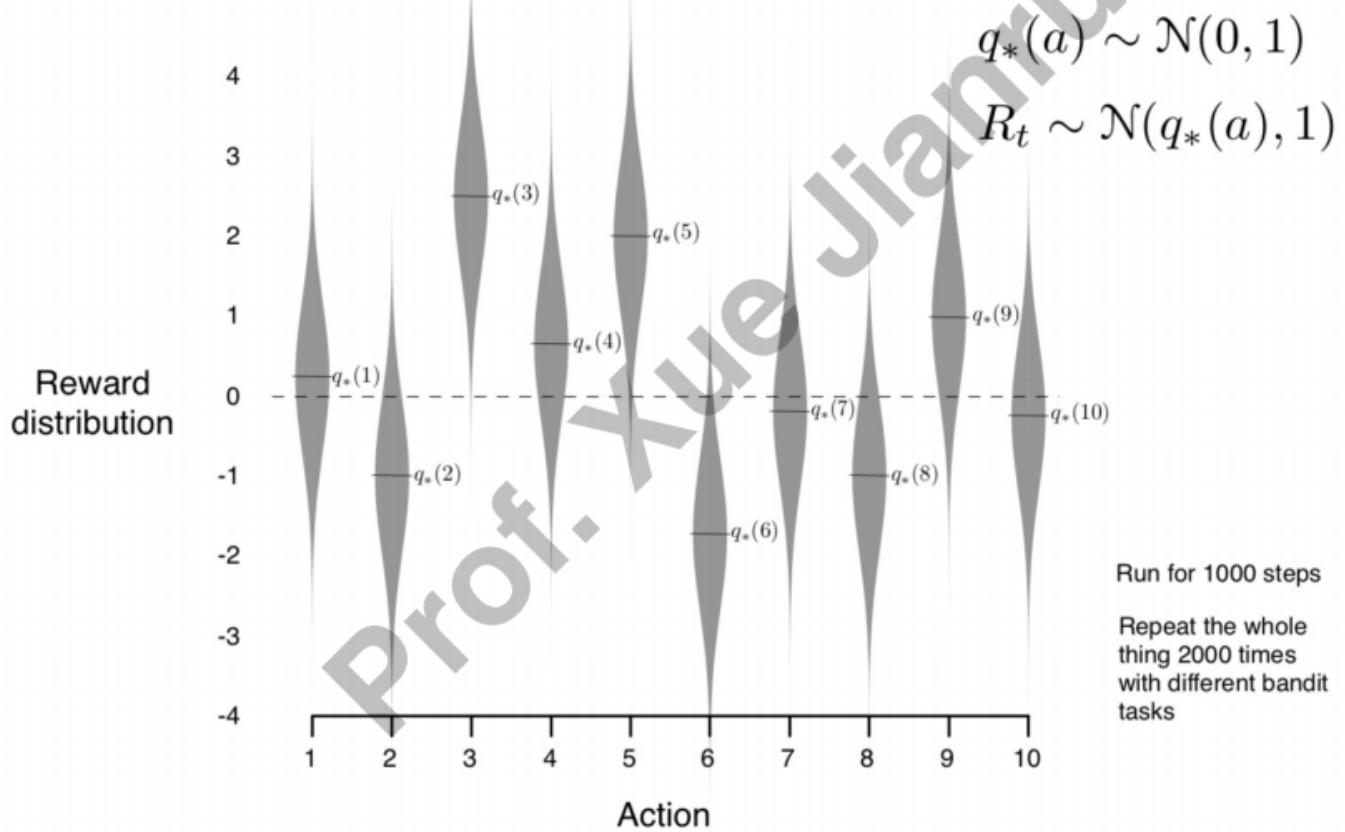
$$Q += (R - Q) / N$$

学习/更新的标准形式

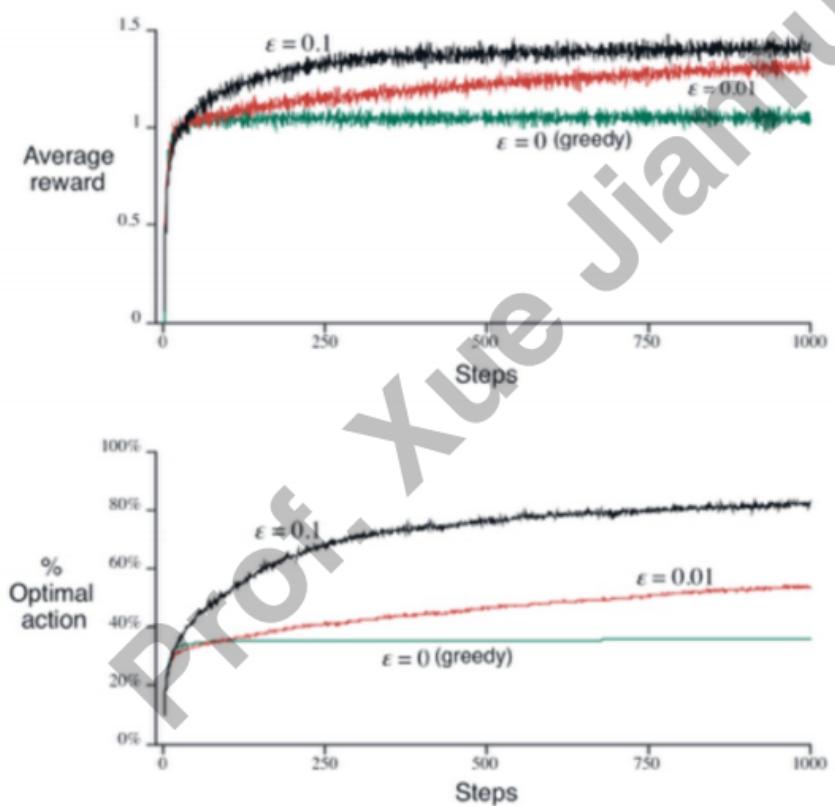
$$\text{NewEstimate} \leftarrow \text{OldEstimate} + \text{StepSize} [\text{Target} - \text{OldEstimate}]$$

测试结果

The 10-armed Testbed



ϵ -Greedy Methods on the 10-Armed Testbed



非平稳问题

如果真实的动作价值随时间变化，采样平均法就不适用了（因为有时间的概念，近期的结果显然对估计更有帮助，久远的结果就不太有参考价值了）

引入指数因子

$$Q_{n+1} = Q_n + \alpha(R_n - Q_n)$$

对比之前的 $Q = Q + (R - Q) / N$

$$Q_{n+1} = Q_n + \alpha(R_n - Q_n)$$

$$= (1-\alpha)Q_n + \alpha R_n$$

即每次迭代时为之前的 Q 施加一个折扣权值 $(1-\alpha)$ ，并以一定的权重增加获益 R ，之前的估计获益就会随时间逐渐衰减。

■ 指数近因加固 exponential, recency-weighted average

$$\begin{aligned} Q_{n+1} &= Q_n + \alpha [R_n - Q_n] \\ &= (1 - \alpha)^n Q_1 + \sum_{i=1}^n \alpha(1 - \alpha)^{n-i} R_i \end{aligned}$$

where α is a constant, *step-size parameter*, $0 < \alpha \leq 1$

收敛性

收敛性的条件---随机逼近理论

- To assure convergence with probability 1:

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \quad \text{and} \quad \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty$$

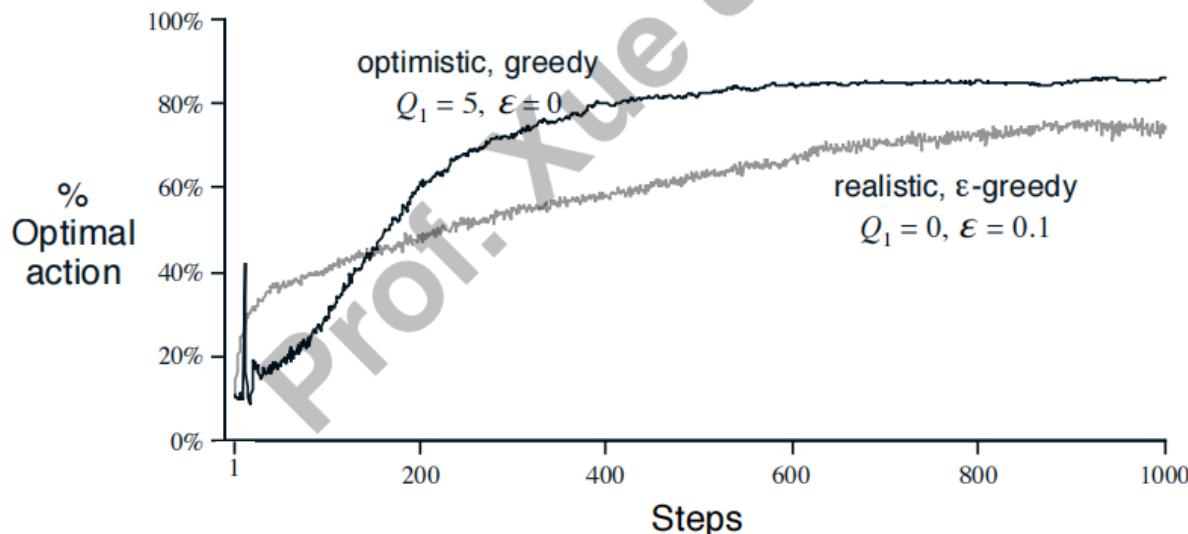
- e.g., $\alpha_n = \frac{1}{n}$
 if $\alpha_n = n^{-p}$, $p \in (0, 1)$
- not $\alpha_n = \frac{1}{n^2}$
 then convergence is at the optimal rate:
 $O(1/\sqrt{n})$

乐观初始值

在进行上述问题求解时需要设置初始值 Q_1 , 之前设置的是0, 但可以设置一个乐观初始值, 如5, 可以提供一种简单的试探方法。

乐观初始值

- 目前讨论的方法在一定程度上依赖Q1的值，常设置Q1为0。
- 动作初始值设置提供了一种简单的试探方法。在10臂测试平台上，设初始的动作价值为5



基于置信度上界UCB的动作选择

也是一种平衡探索-利用的策略，考虑动作估计值有多接近最大值和他们的不确定性（不确定性越高越需要尝试一下这个动作）

UCB估计其实就是在某个置信度下取置信区间的上界作为动作价值估计。

$$A_t \doteq \arg \max_a \left[Q_t(a) + c \sqrt{\frac{\log t}{N_t(a)}} \right]$$

$$\sqrt{\frac{\log t}{N_t(a)}}$$

对动作a的价值估计的不确定性的度量，
c: 置信水平

$\sqrt{\frac{\log t}{N_t(a)}}$ 是不确定性度量，值越大说明在较长的时间内对a的探索较少，不确定性就大。

选一个价值估计大且不确定的动作

效果比 ϵ -贪心要好点

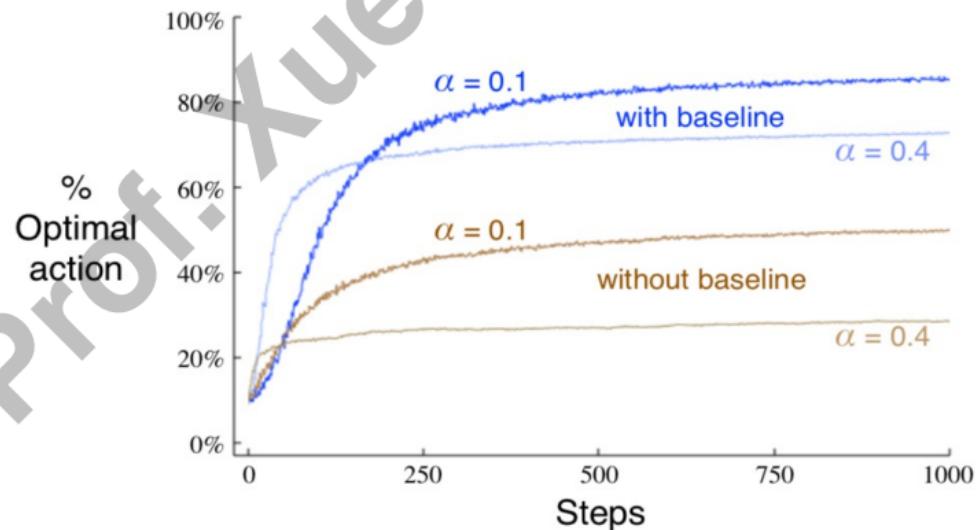
梯度赌博机算法

- Let $H_t(a)$ be a learned preference for taking action a

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^k e^{H_t(b)}} \doteq \pi_t(a) \quad \text{Softmax}$$

$$H_{t+1}(a) \doteq H_t(a) + \alpha \left(R_t - \bar{R}_t \right) \left(\mathbb{1}\{A_t = a\} - \pi_t(a) \right), \quad \forall a$$

$$\bar{R}_t \doteq \frac{1}{t} \sum_{i=1}^t R_i$$



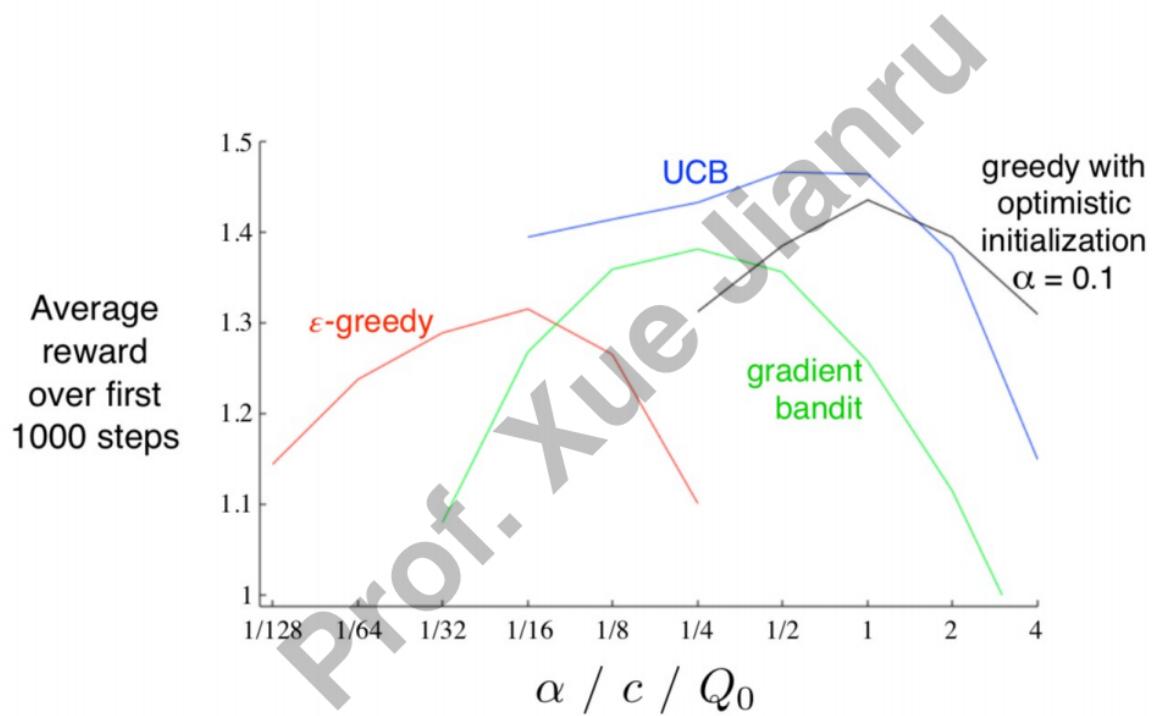
第t次迭代：

以一定概率选中动作a，概率通过 $H_t(a)$ 的softmax计算得到；

$H_{t+1}(a)$ 的更新公式如图， $\mathbb{1}\{A_t = a\}$ 是指示函数，如果第t次迭代选择的动作是a则函数值为1，反之为0；每次迭代都要对所有动作的 H 进行更新，更新公式由梯度下降推得；平均收益的计算如图。

赌博机算法比较

赌博机算法比较



问题描述

非关联任务

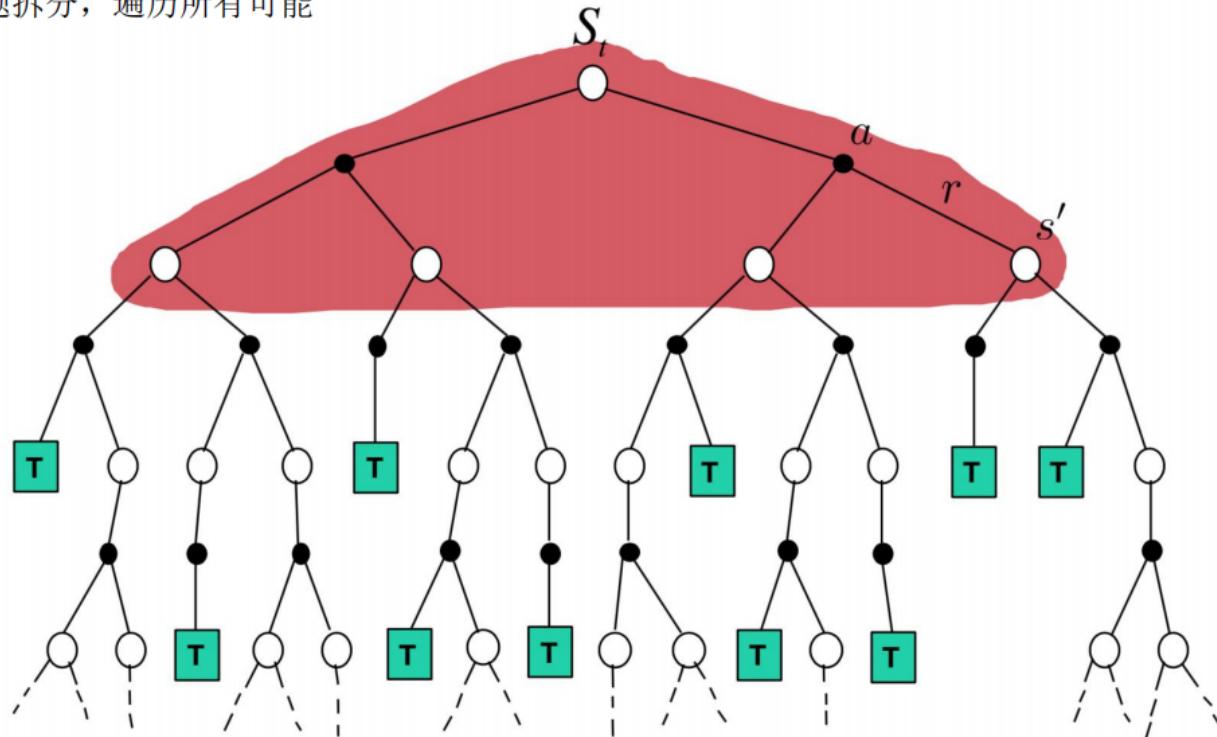
关联任务

DP与MC的对比

DP：计算价值函数

$$V(S_t) \leftarrow E_{\pi} [R_{t+1} + \gamma V(S_{t+1})] = \sum_a \pi(a|S_t) \sum_{s',r} p(s',r|S_t,a) [r + \gamma V(s')]$$

问题拆分，遍历所有可能

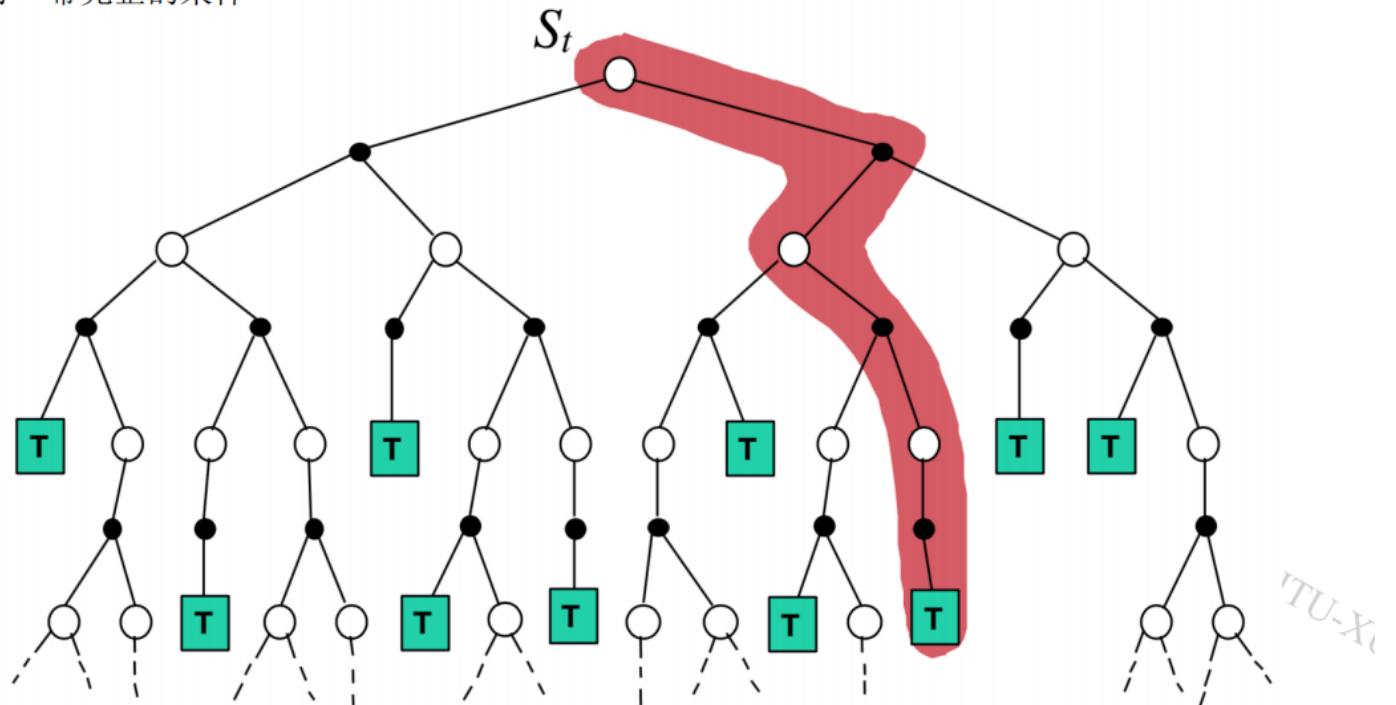


MC：估计价值函数

与自然计算

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

进行一幕完整的采样

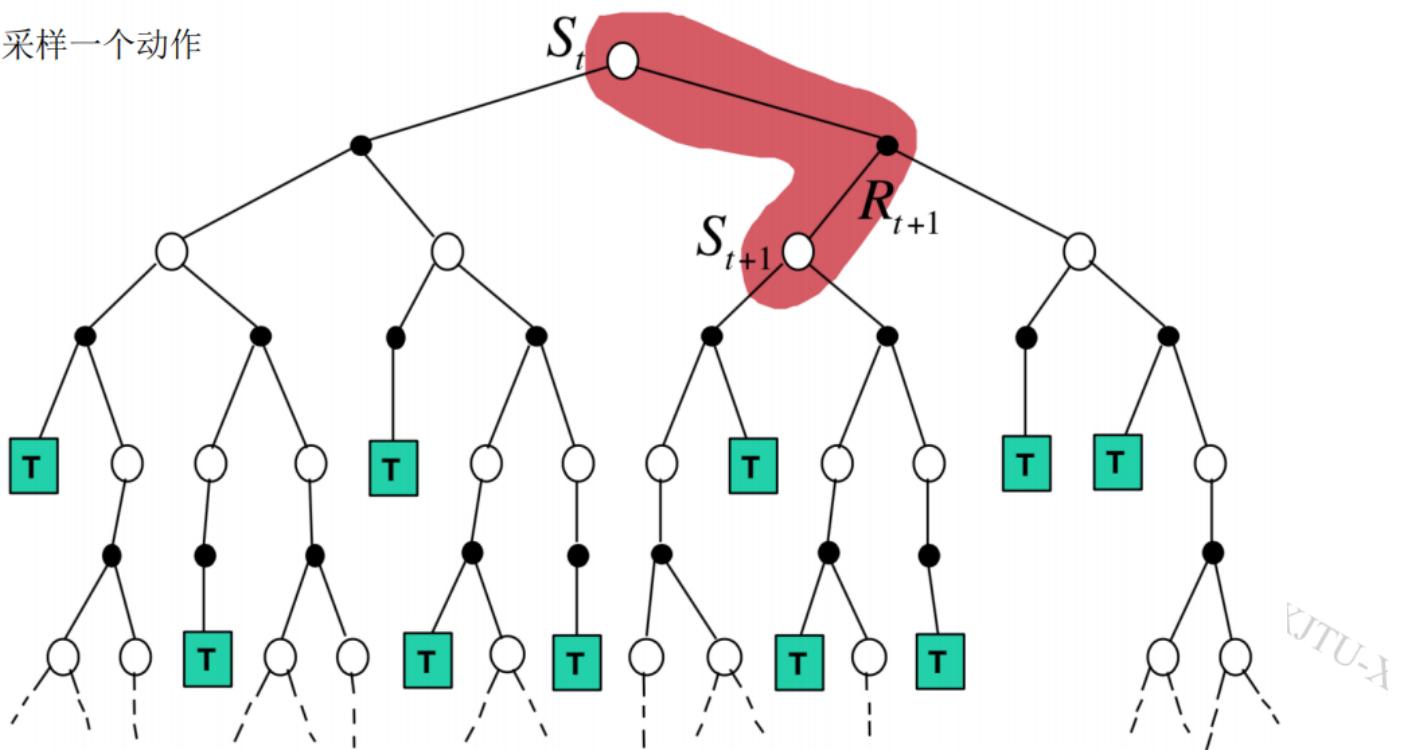


TD: 估计价值函数

与自然计算

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

只采样一个动作



DP

策略迭代算法（使用迭代策略评估，输出最优策略）

1. Initialization

$V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation

Repeat

$\Delta \leftarrow 0$

For each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_{s',r} p(s', r | s, \pi(s)) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number)

3. Policy Improvement

$policy-stable \leftarrow true$

For each $s \in \mathcal{S}$:

$a \leftarrow \pi(s)$

$\pi(s) \leftarrow \arg \max_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$

If $a \neq \pi(s)$, then $policy-stable \leftarrow false$

If $policy-stable$, then stop and return V and π ; else go to 2

动态规划的策略评估，然后贪心进行策略优化，这个是标准的DP

四、价值迭代

确定性价值迭代

若知道子问题 $v_*(s')$ 的解，那么 $v_*(s)$ 可以通过向前看一步得到

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma v_k(s')] \quad \forall s \in \mathcal{S}$$

结合了策略改进和截断策略评估

用 \max 代替策略的求和， v 不再与策略有关

$$v_{k+1}(s) = \max_a \sum_{s',r} p(s',r|s,a) [r + \gamma v_k(s')] \quad \forall s \in \mathcal{S}$$

直觉：从最后的奖励开始往后递推

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat

$$\Delta \leftarrow 0$$

For each $s \in \mathcal{S}$:

$$v \leftarrow V(s)$$

$$V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$$

$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π , such that

$$\pi(s) = \arg \max_a \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$$

价值迭代，直接取动作的最大值作为动作选择（合并了策略评估和策略改进）

异步DP：前面都是同步DP，即价值的更新是同步的，需要先用一个数组 v 存储 V ，再对 V 的所有状态同时进行更新。现采用异步DP，就是每个状态直接就地更新，不需要同时对整个 V 进行更新

MC

策略评估

对各状态价值进行评估

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$: 要找状态 s 未来的收益，倒着遍历 t

$G \leftarrow \gamma G + R_{t+1}$ 感觉这里应该是not appear

Unless S_t appears in S_0, S_1, \dots, S_{t-1} :

Append G to $Returns(S_t)$ if S_t not in $[S_0, S_1, \dots, S_{t-1}]$:

$V(S_t) \leftarrow \text{average}(Returns(S_t))$ $Returns(S_t).append(G)$

$V(S_t) = \text{average}(Returns(S_t))$

策略控制

还要对策略进行优化改进，所以需要计算状态-动作价值即 Q

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:

$Q(s, a) \leftarrow$ arbitrary

$\pi(s) \leftarrow$ arbitrary

$Returns(s, a) \leftarrow$ empty list

Repeat forever:

Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0

Generate an episode starting from S_0, A_0 , following π

For each pair s, a appearing in the episode:

$G \leftarrow$ return following the first occurrence of s, a

Append G to $Returns(s, a)$

$Q(s, a) \leftarrow \text{average}(Returns(s, a))$ 有一个完整的episode， s, a, r 序列就可以算 G

For each s in the episode:

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

存在问题：不能保证对每个动作都探索到（因为一直贪心选择，只有利用没有探索）
需要软策略

引入软策略

同轨

待学习策略 π 本身就是一个软策略，同轨是指行动策略和需要学习的策略一致，怎么学就怎么动。
这里以 ϵ -贪心策略为软策略。

On-policy first-visit MC control (for ϵ -soft policies), estimates $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$$Q(s, a) \leftarrow \text{arbitrary}$$

$$\text{Returns}(s, a) \leftarrow \text{empty list}$$

$$\pi(a|s) \leftarrow \text{an arbitrary } \epsilon\text{-soft policy}$$

Repeat forever:

(a) Generate an episode using π

(b) For each pair s, a appearing in the episode:

$$G \leftarrow \text{return following the first occurrence of } s, a$$

$$\text{Append } G \text{ to } \text{Returns}(s, a)$$

$$Q(s, a) \leftarrow \text{average}(\text{Returns}(s, a))$$

(c) For each s in the episode:

$$A^* \leftarrow \arg \max_a Q(s, a)$$

For all $a \in \mathcal{A}(s)$:

$$\pi(a|s) \leftarrow \begin{cases} 1 - \epsilon + \epsilon/|\mathcal{A}(s)| & \text{if } a = A^* \\ \epsilon/|\mathcal{A}(s)| & \text{if } a \neq A^* \end{cases}$$

离轨

从别人的行动经验中学习，待学习策略和行动（采样）策略不一致。可以从一个软策略中学习，最终得到一个贪心策略。

离轨评估

Off-policy MC prediction, for estimating $Q \approx q_\pi$

Input: an arbitrary target policy π

Initialize, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:

$$Q(s, a) \leftarrow \text{arbitrary}$$

$$C(s, a) \leftarrow 0$$

Repeat forever:

$b \leftarrow$ any policy with coverage of π

Generate an episode using b :

$$S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T, S_T$$

$$G \leftarrow 0$$

$$W \leftarrow 1$$

For $t = T - 1, T - 2, \dots$ downto 0:

$$G \leftarrow \gamma G + R_{t+1}$$

$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$ 动作-状态价值更新为加权和

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$$

$W \leftarrow W \frac{\pi(A_t | S_t)}{b(A_t | S_t)}$ 重要性采样，评估策略与采样策略之比

If $W = 0$ then ExitForLoop

离轨控制

使用贪心策略为待学习策略 π ，因为是贪心策略，所以 W 更新时分子就直接是 1 了（只会采用最优的动作）

Off-policy MC control, for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:

$$Q(s, a) \leftarrow \text{arbitrary}$$

$$C(s, a) \leftarrow 0$$

$$\pi(s) \leftarrow \operatorname{argmax}_a Q(S_t, a) \quad (\text{with ties broken consistently})$$

Repeat forever:

$b \leftarrow$ any soft policy

Generate an episode using b :

$$S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T, S_T$$

$$G \leftarrow 0$$

$$W \leftarrow 1$$

For $t = T - 1, T - 2, \dots$ downto 0:

$$G \leftarrow \gamma G + R_{t+1}$$

前面同上

$$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$$

$$\pi(S_t) \leftarrow \operatorname{argmax}_a Q(S_t, a) \quad (\text{with ties broken consistently})$$

If $A_t \neq \pi(S_t)$ then ExitForLoop

$W \leftarrow W \frac{1}{b(A_t | S_t)}$ 待评估策略和采样策略得到的动作不一致，退出循环（再算下去也没意义了）

TD(0)

Sarsa

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$

Repeat (for each episode):

 Initialize S

 Choose A from S using policy derived from Q (e.g., ϵ -greedy)

 Repeat (for each step of episode):

 Take action A , observe R, S'

 Choose A' from S' using policy derived from Q (e.g., ϵ -greedy)

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

$S \leftarrow S'; A \leftarrow A'$;

 until S is terminal

Q-Learning

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$

Repeat (for each episode):

 Initialize S

 Repeat (for each step of episode):

 Choose A from S using policy derived from Q (e.g., ϵ -greedy)

 Take action A , observe 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 is terminal

对比

在状态-价值更新时不同；并且Sarsa的下一个A承接A'，因为是同轨的，待评估策略与行动策略一致。但Q-Learning就不是了，A是待学习策略决定的，而后继动作A'是贪心策略选择的，所以不能直接挪用。

TD(n)

以n步采样得到的总收益G作为预计收益

策略评估

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha [G_{t:t+n} - V_{t+n-1}(S_t)], \quad 0 \leq t < T$$

n-step TD for estimating $V \approx v_\pi$

Initialize $V(s)$ arbitrarily, $s \in \mathcal{S}$

Parameters: step size $\alpha \in (0, 1]$, a positive integer n

All store and access operations (for S_t and R_t) can take their index mod $n+1$

Repeat (for each episode):

 Initialize and store $S_0 \neq$ terminal

$T \leftarrow \infty$

 For $t = 0, 1, 2, \dots :$

 If $t < T$, then:

 Take an action according to $\pi(\cdot | S_t)$

 Observe and store the next reward as R_{t+1} and the next state as S_{t+1}

 If S_{t+1} is terminal, then $T \leftarrow t+1$ 一暮走到头了就停, 不再继续选动作, 逐步更新先前的V就行了

$\tau \leftarrow t - n + 1$ (τ is the time whose state's estimate is being updated)

 If $\tau \geq 0$:

$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$

 If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$

$V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$

 Until $\tau = T - 1$ 前n-1步不更新, 走完第n步时更新第1步, 最后的n-1步可以直接算出G

策略控制

n步SARSA

进行n步采样, 用n步采样的总收益G代替SARSA中一步采样的收益R

n -step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given π

Initialize $Q(s, a)$ arbitrarily, $\forall s \in \mathcal{S}, a \in \mathcal{A}$

Initialize π to be ε -greedy with respect to Q , or to a fixed given policy

Parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$, a positive integer n

All store and access operations (for S_t , A_t , and R_t) can take their index mod $n+1$

$Q \rightarrow \pi$

Repeat (for each episode):

 Initialize and store $S_0 \neq$ terminal

 Select and store an action $A_0 \sim \pi(\cdot | S_0)$

$T \leftarrow \infty$

 For $t = 0, 1, 2, \dots :$

 If $t < T$, then:

 Take action A_t

 Observe and store the next reward as R_{t+1} and the next state as S_{t+1}

 If S_{t+1} is terminal, then:

$T \leftarrow t + 1$

 同上，不断选择动作直到到达终点，到达终点之后就只更新，不

 再选择

 Select and store an action $A_{t+1} \sim \pi(\cdot | S_{t+1})$

$\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)

 If $\tau \geq 0$:

$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$

 If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$

($G_{\tau:\tau+n}$)

$Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$

If π is being learned, then ensure that $\pi(\cdot | S_\tau)$ is ε -greedy wrt Q

Until $\tau = T - 1$

相对于单步SARSA加快了策略学习

离轨

利用重要性采样

Off-policy n -step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given π

Input: an arbitrary behavior policy b such that $b(a|s) > 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$

Initialize $Q(s, a)$ arbitrarily, $\forall s \in \mathcal{S}, a \in \mathcal{A}$

Initialize π to be ε -greedy with respect to Q , or as a fixed given policy

Parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$, a positive integer n

All store and access operations (for S_t , A_t , and R_t) can take their index mod $n+1$

Repeat (for each episode):

 Initialize and store $S_0 \neq$ terminal

 Select and store an action $A_0 \sim b(\cdot|S_0)$ *首个动作随机选择*

$T \leftarrow \infty$

 For $t = 0, 1, 2, \dots$:

 If $t < T$, then:

 Take action A_t

 Observe and store the next reward as R_{t+1} and the next state as S_{t+1}

 If S_{t+1} is terminal, then:

$T \leftarrow t + 1$

 else:

 Select and store an action $A_{t+1} \sim b(\cdot|S_{t+1})$

$\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)

 If $\tau \geq 0$:

$\rho \leftarrow \prod_{i=\tau+1}^{\min(\tau+n-1, T-1)} \frac{\pi(A_i|S_i)}{b(A_i|S_i)}$ *广义策略评估* $(\rho_{\tau+1:t+n-1})$

$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$ *重要性采样的校正也叠加*

 If $\tau + n < T$, then: $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$

$Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha \rho [G - Q(S_\tau, A_\tau)]$

 If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε -greedy wrt Q

 Until $\tau = T - 1$

n 步回溯树：不用重要性采样

直接用回溯树算出每步期望获益，就不需要再从 b 中去采样 π 了

单步树回溯的回报(与期望Sarsa相同)

$$G_{t:t+1} \doteq R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q_t(S_{t+1}, a)$$

a

对于 $t < T-2$, 两步树回溯的回报

$$\begin{aligned} G_{t:t+2} &\doteq R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+1}(S_{t+1}, a) \\ &\quad + \gamma \pi(A_{t+1}|S_{t+1}) \left(R_{t+2} + \gamma \sum_a \pi(a|S_{t+2}) Q_{t+1}(S_{t+2}, a) \right) \\ &= R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+1}(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1}) G_{t+1:t+2}, \end{aligned}$$

通用的n步树回溯回报的递归形式

$$G_{t:t+n} \doteq R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+n-1}(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1}) G_{t+1:t+n},$$

动作价值更新

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha [G_{t:t+n} - Q_{t+n-1}(S_t, A_t)]$$

n -step Tree Backup for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given π

Initialize $Q(s, a)$ arbitrarily, $\forall s \in \mathcal{S}, a \in \mathcal{A}$

Initialize π to be ε -greedy with respect to Q , or as a fixed given policy

Parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$, a positive integer n

All store and access operations can take their index mod $n+1$

Repeat (for each episode):

 Initialize and store $S_0 \neq$ terminal

 Select and store an action $A_0 \sim \pi(\cdot | S_0)$

 Store $Q(S_0, A_0)$ as Q_0

$T \leftarrow \infty$

 For $t = 0, 1, 2, \dots :$

 If $t < T$:

 Take action A_t

 Observe the next reward R ; observe and store the next state as S_{t+1}

 If S_{t+1} is terminal:

$T \leftarrow t + 1$

 Store $R - Q_t$ as δ_t

-Q_t 是因为上面的那个递归式的求和是 $a \neq A_{t+1}$

 else:

 Store $R + \gamma \sum_a \pi(a | S_{t+1}) Q(S_{t+1}, a) - Q_t$ as δ_t

 Select arbitrarily and store an action as A_{t+1}

 Store $Q(S_{t+1}, A_{t+1})$ as Q_{t+1}

 Store $\pi(A_{t+1} | S_{t+1})$ as π_{t+1}

$\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)

 If $\tau \geq 0$:

$e \leftarrow 1$

$G \leftarrow Q_\tau$

 For $k = \tau, \dots, \min(\tau + n - 1, T - 1)$:

$G \leftarrow G + e \delta_k$

$e \leftarrow \gamma e \pi_{k+1}$

$Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$

 If π is being learned, then ensure that $\pi(a | S_\tau)$ is ε -greedy wrt $Q(S_\tau, \cdot)$

 Until $\tau = T - 1$

策略梯度方法

动作价值方法：基于动作价值函数选择策略

策略梯度方法：把策略用参数进行表示， $\pi(s, \theta)$ ，策略改进就是对 θ 做关于性能 $J(\theta)$ 的梯度上升。

$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta)$ 。直接学习策略（动作选择）不依赖于动作价值估计。带有动作-状态价值的策略梯度方法成为“行动器-评判器”

相较于动作价值的好处：以随机概率选择动作（softmax），可以学到一个随机的策略而不是确定的动作选择。好处是可以接近于一个确定的策略，同时以一个任意的概率来选择动作（因为有些时候最佳策略可能是概率的，如扑克游戏）。另一个优势是动作概率平滑变化，基于动作价值估计的方法可能因为动作价值函数的微小变化导致动作选择概率的突变（ ϵ -贪心），但策略梯度方法参数是连续变化的，策略也是平滑变化。

策略梯度定理

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s, a) \nabla \pi(a|s, \theta),$$

其中 μ 是分布，用这个定理得到 $J(\theta)$ 正比于某个式子，系数可以被梯度上升法的 α 包括。因此可以用后边那个式子作为梯度上升法的梯度。

REINFORCE

REINFORCE 算法就是基于策略梯度定理的一种策略梯度算法，也是一种蒙特卡洛算法。