n 步自举法

强化学习讨论班

啥事儿啊
啥事儿啊系，啥事儿啊大学

2023 年 5 月

目录

基本定义回顾

1. **状态** s, s'，它们是对环境的描述，其状态空间我们记为 $\mathcal{S}$，在**时刻** t 的状态我们记为 S_t；
2. **动作** a 是对智能体行为的描述，其在状态 s 下所有可行动作的集合我们记为 $\mathcal{A}(s)$，在时刻 t 智能体采取的动作我们记为 A_t；
3. **策略** π 是智能体的决策规则，我们记 $\pi(s)$ 为在状态 s 根据确定性策略 π 选择的动作；而 $\pi(s|a)$ 代表在根据随机性策略 π 在状态 s 下选择动作 a 的概率；
4. **回报** G_t 指在时刻 t 智能体获得的收益，定义 $p(s', r|s, a)$ 为从状态 s 采取动作 a 转移至状态 s' 并获得回报 r 的概率；并记 $r(s, a)$ 为从状态 s 采取动作 a 得到的即时收益的期望。

预测 v.s. 控制

1. **预测** (Prediction) 指的是对固定策略 π，对在给定状态 s 下价值函数 $v_\pi(s)$ 与进行动作 a 后汇报函数 $q_\pi(s,a)$ 的计算。
2. **控制** (Control) 指的是去找到最优的策略 $\pi(s|a)$ 使得对于任意给定的状态都能最大化期望总收益。

Monte-Carlo 和时序差分方法

更新策略：考虑一个状态收益序列 $S_t, R_{t+1}, S_{t+1}, \ldots, R_T, S_T$

1. Monte-Carlo 方法（对于每次访问型、固定 α）：

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

Monte-Carlo 方法对 $V(S_t)$ 的估计值会沿着完整回报的方向进行更新，也即更新的目标是 $G_t := R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-t-1} R_T$

2. 时序差分方法（单步 TD）

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

单步 TD 的更新目标是 $R_{t+1} + \gamma V(S_{t+1})$，也被称为是**单步回报**，利用 $\gamma V(S_{t+1})$ 项代替了完整回报中的 $\gamma R_{t+2} + \ldots + \gamma^{T-t-1} R_T$ 项

TD(0) 的具体控制算法

1. Sarsa 法

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

2. Q-学习

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

3. 期望 Sarsa 法

$$Q(S_t, A_t) \leftarrow Q(S_t, S_t) + \alpha[R_{t+1} + \gamma \mathbb{E}[Q(S_{t+1}, A_{t+1})|S_{t+1}] - Q(S_t, A_t)]$$

之前算法的总结

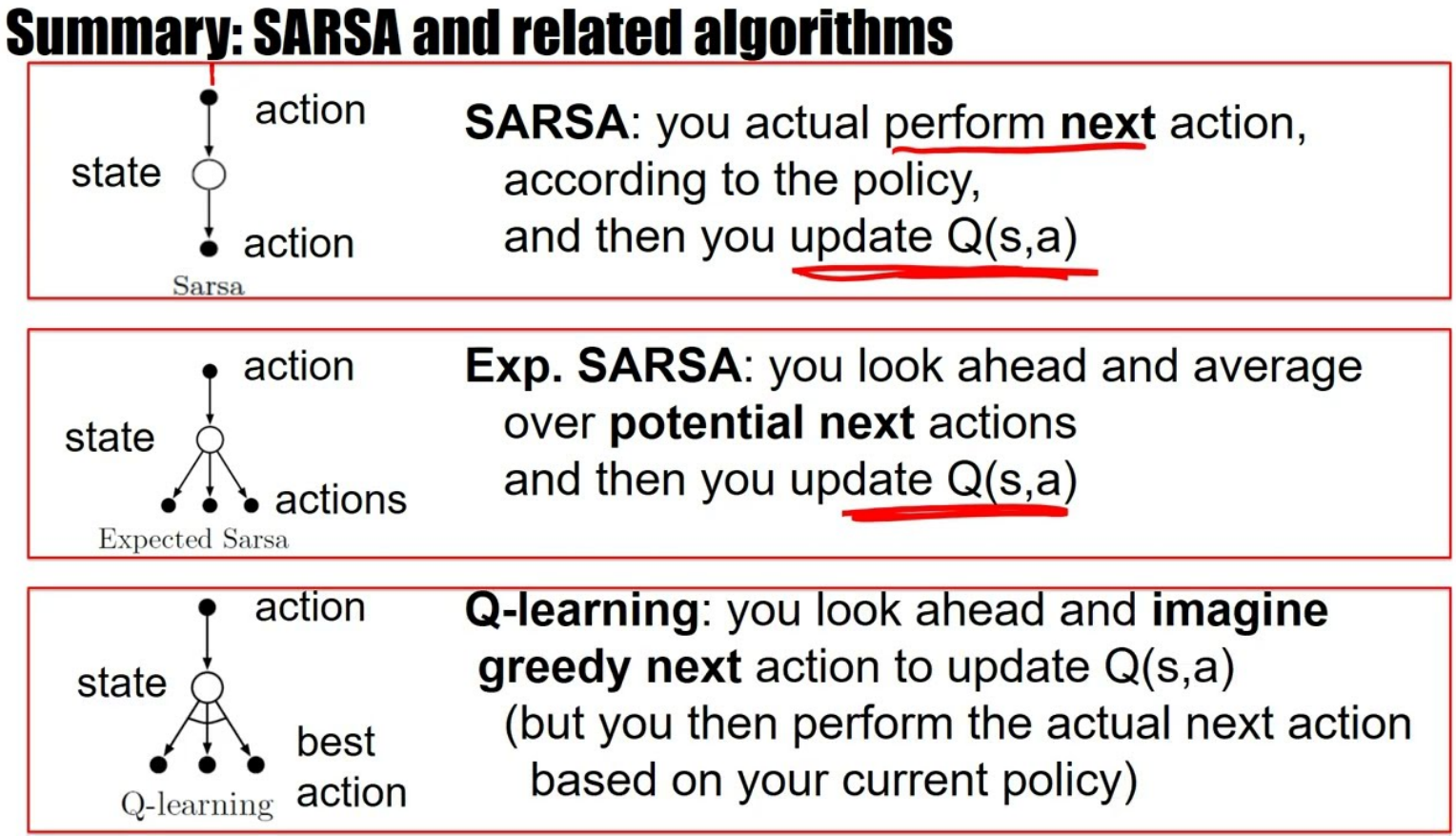


图: Sarsa 相关算法的总结，图片来自 Wulfram Gerstner

自举法是对已有数据的再采样

自举法 (Bootstrapping method) 是一种非参数统计估计方法，它的思想是通过对小样本数据估计的平均来估计群体的统计量。

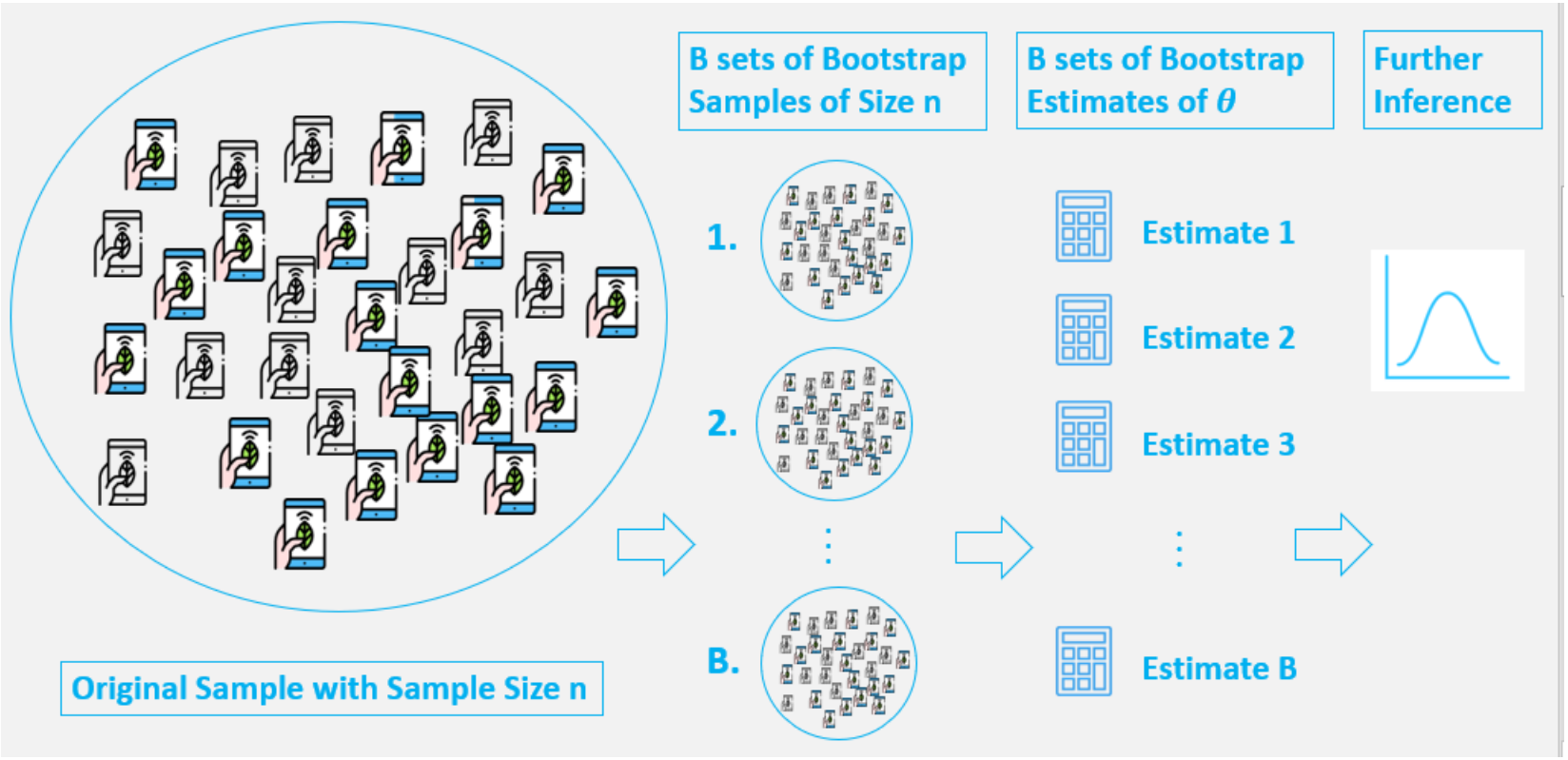


图: 自举法流程，图片来自 towards data science

目录

n 步方法的回溯图

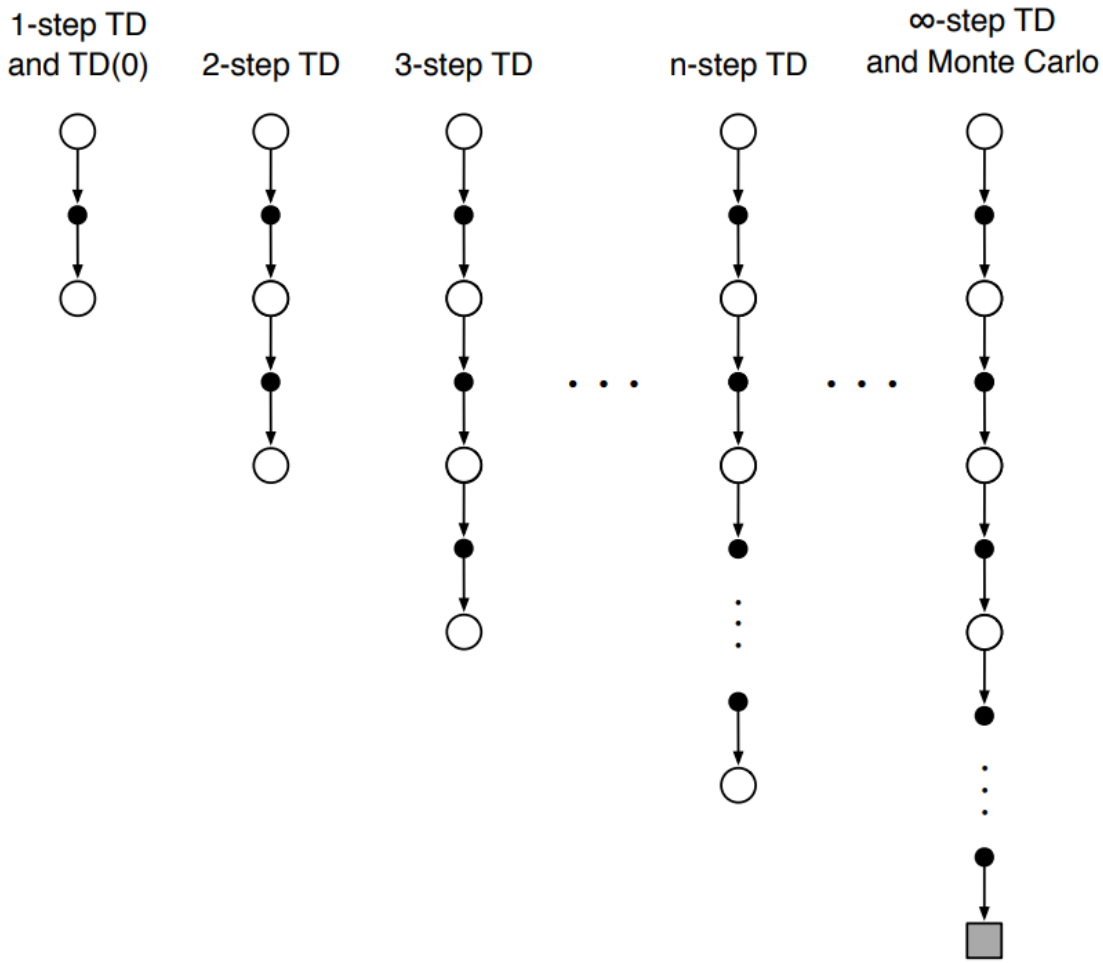


图: n 步时序差分方法是介于单步差分法和 Monte-Carlo 方法之间的算法

n 步方法的更新方式

在单步方法中，我们定义了单步回报：

$$G_{t:t+1} := R_{t+1} + \gamma V_t(S_{t+1})$$

其中 $G_{t:t+1}$ 的下标表示这是一种截断回报，它由时刻 t 到时刻 $t+1$ 的累积收益加上折后回报 $\gamma V_t(S_{t+1})$ 组成的。那对于 n 步更新的情况，我们也可以考虑将目标定义为 n 步回报上：

$$G_{t:t+n} := R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$$

因此我们考虑的一个更新方式：

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

n 步 TD 算法的伪代码实现

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

Input: a policy π
Algorithm parameters: step size $\alpha \in (0, 1]$, a positive integer n
Initialize $V(s)$ arbitrarily, for all $s \in \mathcal{S}$
All store and access operations (for S_t and R_t) can take their index mod $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
  | 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 $T \leftarrow t+1$
  | $\tau \leftarrow t - n + 1$ (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 $\tau = T - 1$

误差减少性质

n 步回报利用 V_{t+n-1} 用于校正 R_{t+n} 之后的所有剩余收益之和。使用 n 步回报的期望值对 v_π 的估计回比 V_{t+n-1} 更好，也即：

$$\max_s |\mathbb{E}_\pi[G_{t:t+n}|S_t = s] - v_\pi(s)| \leq \gamma^n \max_s |V_{t+n-1}(s) - v_\pi(s)|$$

具体证明见 Stackexchange，一个简要的思路就是考虑展开 $G_{t:t+n}$ 和 $v_\pi(s)$，然后代入不等式左边后利用 $|\mathbb{E}(X)| \leq \mathbb{E}(|X|)$ 计算即可。

随机游走任务

例

从中间结点 C 出发，每步有均等的概率向左或向右。若能达到右边的终止态，返回奖励值 1，其余情况均返回奖励值 0。

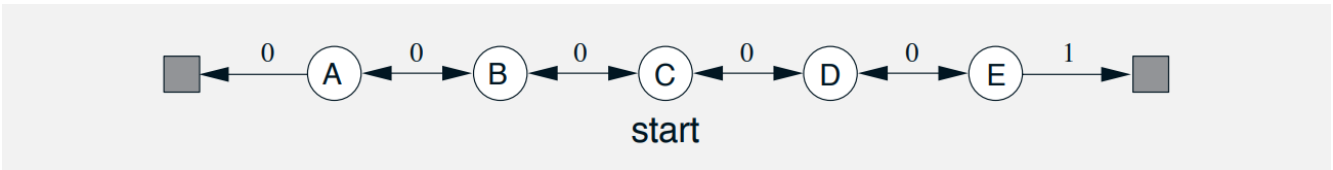


图: 随机游走任务，这是一个 Markov 收益过程

现在我们假设从中间节点出发，往右边走经过节点 D 和 E。最开始所有的状态价值都是 0.5，也即 $V(s) = 0.5$。单步差分只能更新最后一个状态的价值函数 $V(E)$，而对于二步或者任意 n 步方法 $(n > 2)$ 都可以向 1 的方向更新所有经过状态的价值函数。

关于 n 的选择：随机游走任务（续）

考虑一个包含 19 个状态、左边收益为 -1，且满足所有的初始状态价值为 0 的 Markov 收益过程，对于不同的 n，作出关于学习率 α 的函数如下，我们可以得到性能的比较图：

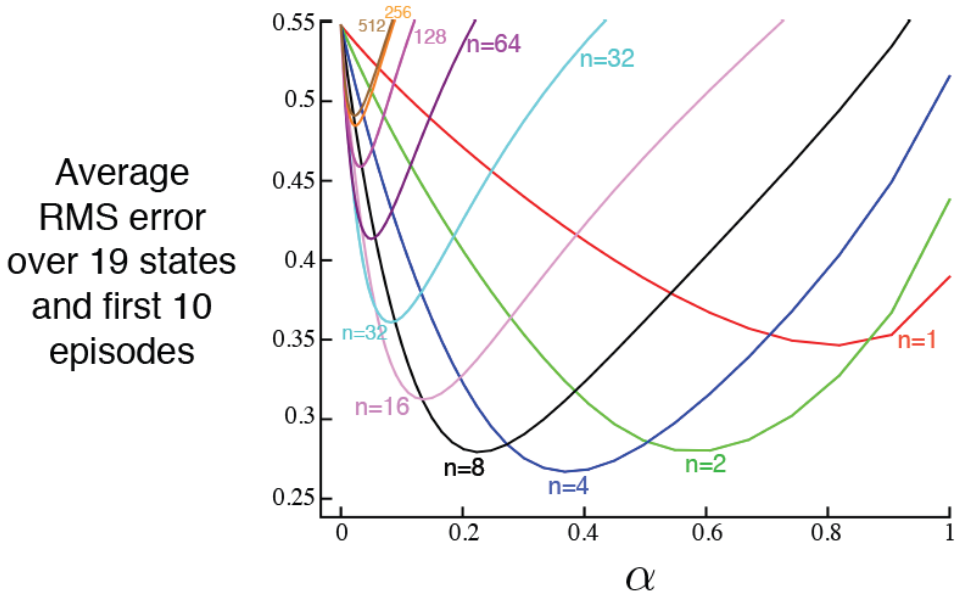
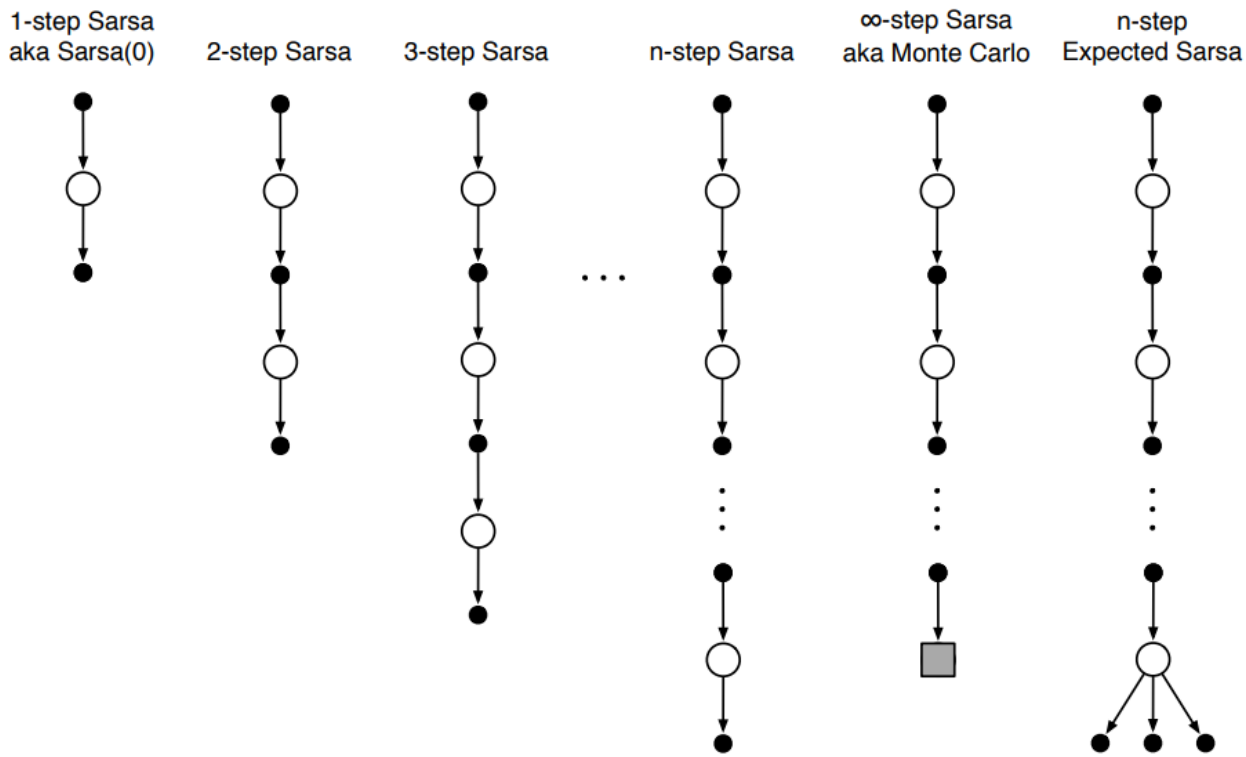


图: 对于不同的 n，n 步时序差分方法的性能

代码实验可见 Random Walk Task。

n 步 Sarsa 的回溯图

核心思想：把状态 s 替换成状态和动作的二元组 (s, a)，并使用 ε-贪心策略



图：n 步 Sarsa 的回溯图

n 步 Sarsa 的更新方式

我们定义如下 n 步方法的回报：

$$G_{t:t+n} := R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, Q_{t+n})$$

当 $t+n \geq T$ 时，我们有 $G_{t:t+n} = G_t$，因此可以得到：

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

对于所有满足 $s \neq S_t$ 或 $a \neq A_t$ 的 s, a，我们都可以得到 $Q_{t+n}(s, a) = Q_{t+n-1}(s, a)$。

n 步 Sarsa 算法的伪代码实现

n-step Sarsa for estimating $Q \approx q_*$ or q_π

Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ε-greedy with respect to Q, or to a fixed given policy
Algorithm 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 $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
  | 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 $A_{t+1} \sim \pi(\cdot|S_{t+1})$
  | $\tau \leftarrow t-n+1$ (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 $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
  | | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\left[G - Q(S_\tau, A_\tau)\right]$
  | | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
  Until $\tau = T-1$

网格世界的例子

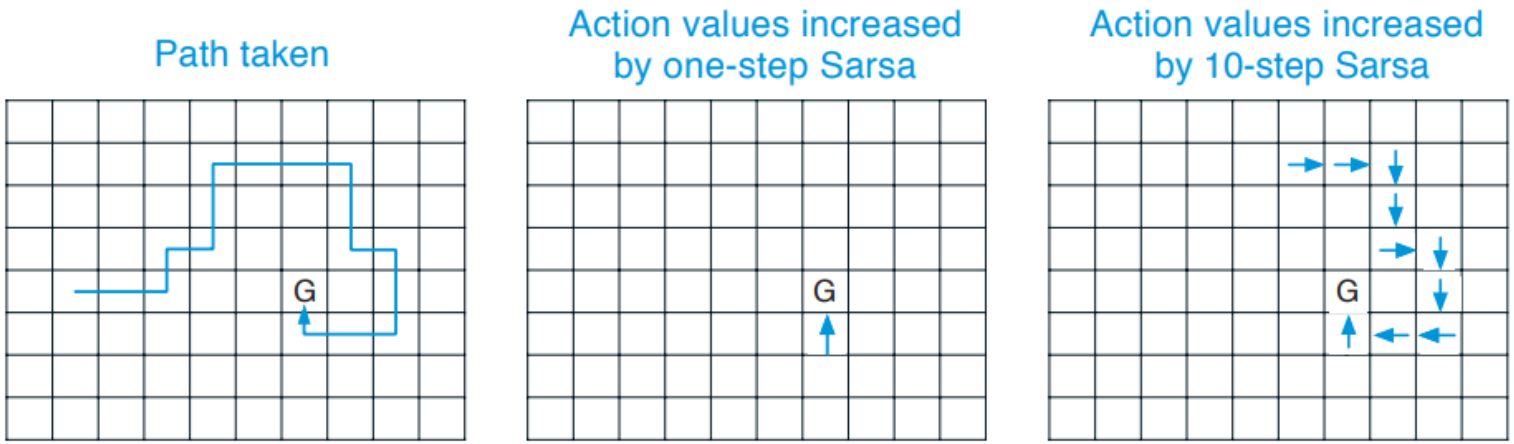


图: 单步 Sarsa 只增强了最后一步动作的价值，而 n 步 Sarsa 增强了最后 n 步动作的价值，因此获得了更多的知识

n 步期望 Sarsa 方法

与 n 步 Sarsa 方法的追溯图不同的是，n 步期望 Sarsa 方法的追溯图最后的节点对所有可能的动作采用策略 π 下的概率进行了加权，并需要对 n 步回报重新定义：

$$G_{t:t+n} := R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n \bar{V}_{t+n-1}(S_{t+n})$$

其中：

$$\bar{V}_t(s) := \sum_a \pi(a|s) Q_t(s,a), \forall s \in \mathcal{S}$$

重要性采样率

在 n 步方法里，回报根据 n 步建立，因此我们应该考虑在这 n 步里由于不同策略 b 对应动作的相对概率。最简单的，考虑使用 $\rho_{t:t+n-1}$ 对 t 时刻的更新进行加权：

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

其中 $\rho_{t:t+n-1}$ 被称为是**重要度采样率**，是两种策略采取 $A_t \sim A_{t+n}$ 这 n 个动作的相对概率，也即：

$$\rho_{t:h} := \prod_{k=t}^{\min(h,T-1)} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

离轨策略的 n 步 Sarsa 算法

如果我们考虑 $\pi = b$，也即两个策略相同，则我们可以得到重要度采样率恒为 1，也即

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

实际上是对我们上面 n 步方法更新的推广。因此，我们能将之前的 n 步 Sarsa 更新方法用下面的离轨策略法代替：

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

离轨策略的 n 步 Sarsa 伪代码实现

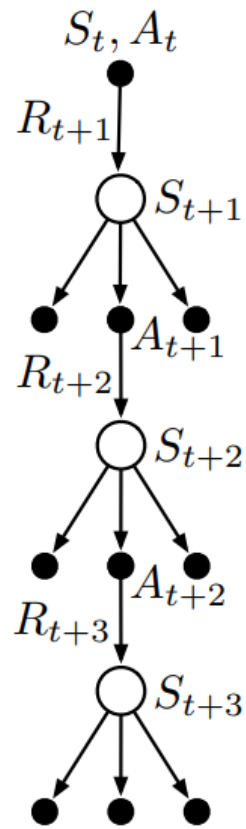
Off-policy n-step Sarsa for estimating $Q \approx q_*$ or q_π

Input: an arbitrary behavior policy b such that $b(a|s) > 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be greedy with respect to Q, or as a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, a positive integer n
All store and access operations (for S_t, A_t, and R_t) can take their index mod $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim b(\cdot|S_0)$
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
    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 $\tau \geq 0$:
      $\rho \leftarrow \prod_{i=\tau+1}^{\min(\tau+n-1,T-1)} \frac{\pi(A_i|S_i)}{b(A_i|S_i)}$ $(\rho_{\tau+1:t+n-1})$
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
      If $\tau+n < T$, then: $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
      $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\rho\left[G - Q(S_\tau, A_\tau)\right]$
      If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is greedy wrt Q
  Until $\tau = T-1$

n 步树回溯算法的动机

不使用重要度采样的离轨学习在单步的情况下我们讨论了单步 Q 学习和期望 Sarsa 方法，下面我们介绍多步的方法，也被称为是树回溯算法。考虑左边的三步树回溯图：



the 3-step tree-backup update

树回溯算法的更新策略

单步回报与期望 Sarsa 相同，对于 $t < T-1$ 我们有：

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

而对于 $t < T-2$，我们有两步树回溯的回报：

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

树回溯算法的更新策略（续）

在上一面 slides 我们看到了递归定义的一般形式，因此我们写出 n 步树回溯的回报为：

$$\begin{aligned} G_{t:t+n} :=& R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi\left(a \mid S_{t+1}\right) Q_{t+n-1}\left(S_{t+1}, a\right) \\ &+ \gamma \pi\left(A_{t+1} \mid S_{t+1}\right) G_{t+1:t+n} \end{aligned}$$

树回溯算法的伪代码实现

n-step Tree Backup for estimating $Q \approx q_*$ or q_π

Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be greedy with respect to Q, or as a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, a positive integer n
All store and access operations can take their index mod $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  Choose an action A_0 arbitrarily as a function of S_0; Store A_0
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
  | If $t < T$:
  | Take action A_t; observe and store the next reward and state as R_{t+1}, S_{t+1}
  | If S_{t+1} is terminal:
  | $T \leftarrow t+1$
  | else:
  | Choose an action A_{t+1} arbitrarily as a function of S_{t+1}; Store A_{t+1}
  | $\tau \leftarrow t+1-n$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | If $t+1 \geq T$:
  | $G \leftarrow R_T$
  | else
  | $G \leftarrow R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$
  | Loop for $k = \min(t, T-1)$ down through $\tau+1$:
  | $G \leftarrow R_k + \gamma \sum_{a \neq A_k} \pi(a|S_k) Q(S_k, a) + \gamma \pi(A_k|S_k) G$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is greedy wrt Q
  Until $\tau = T-1$

目录

自举法和算法选择

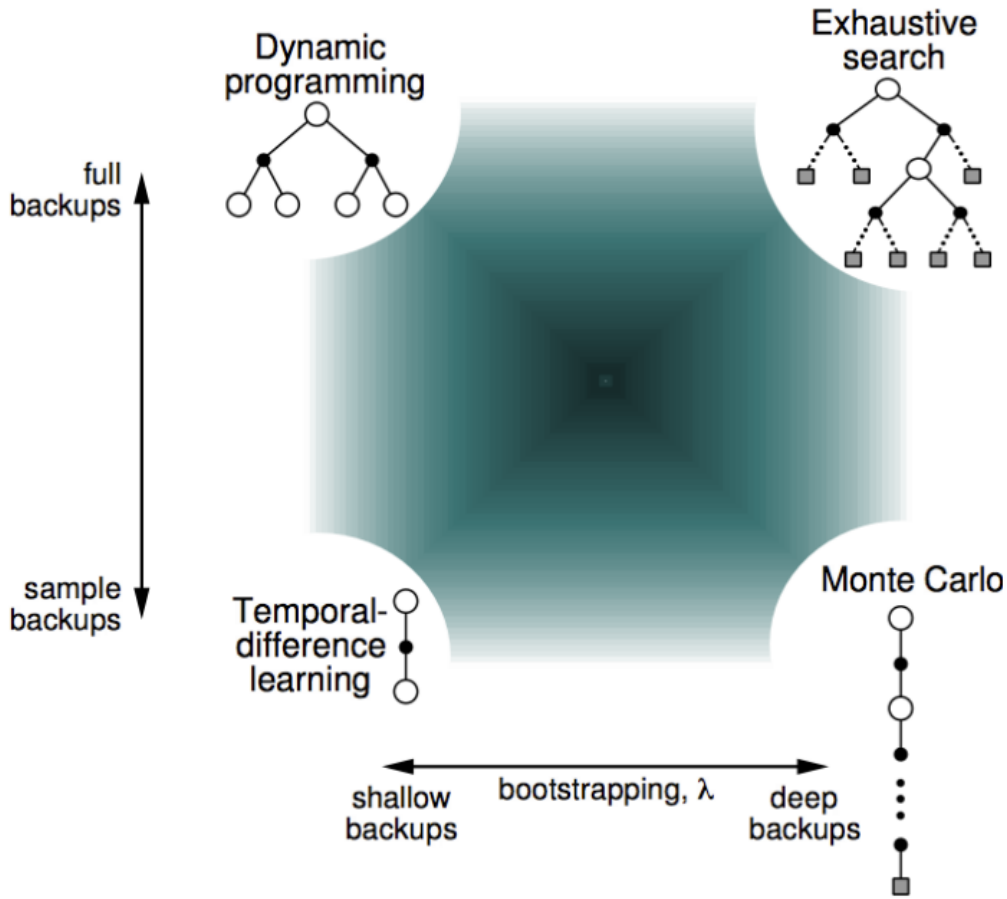


图: 图片来自 Richard Warren 的博客