强化学习2022
第3节

涉及知识点：
模型无关强化学习、蒙特卡洛方法、蒙特卡洛价值预测、时序差分学习、重要性采样

值函数估计

无模型的强化学习

无模型的强化学习（Model-free RL）

- 在现实问题中，通常没有明确地给出状态转移和奖励函数
  - 例如，我们仅能观察到部分片段（episodes）

Episode 1: $s_0^{(1)} \xrightarrow[R(s_0)^{(1)}]{a_0^{(1)}} s_1^{(1)} \xrightarrow[R(s_1)^{(1)}]{a_1^{(1)}} s_2^{(1)} \xrightarrow[R(s_2)^{(1)}]{a_2^{(1)}} s_3^{(1)} \ldots s_T^{(1)}$

Episode 2: $s_0^{(2)} \xrightarrow[R(s_0)^{(2)}]{a_0^{(2)}} s_1^{(2)} \xrightarrow[R(s_1)^{(2)}]{a_1^{(2)}} s_2^{(2)} \xrightarrow[R(s_2)^{(2)}]{a_2^{(2)}} s_3^{(2)} \ldots s_T^{(2)}$

- 模型无关的强化学习直接从经验中学习值（value）和策略（policy），而无需构建马尔可夫决策过程模型（MDP）
- 关键步骤：（1）估计值函数；（2）优化策略

Recall: 强化学习系统要素

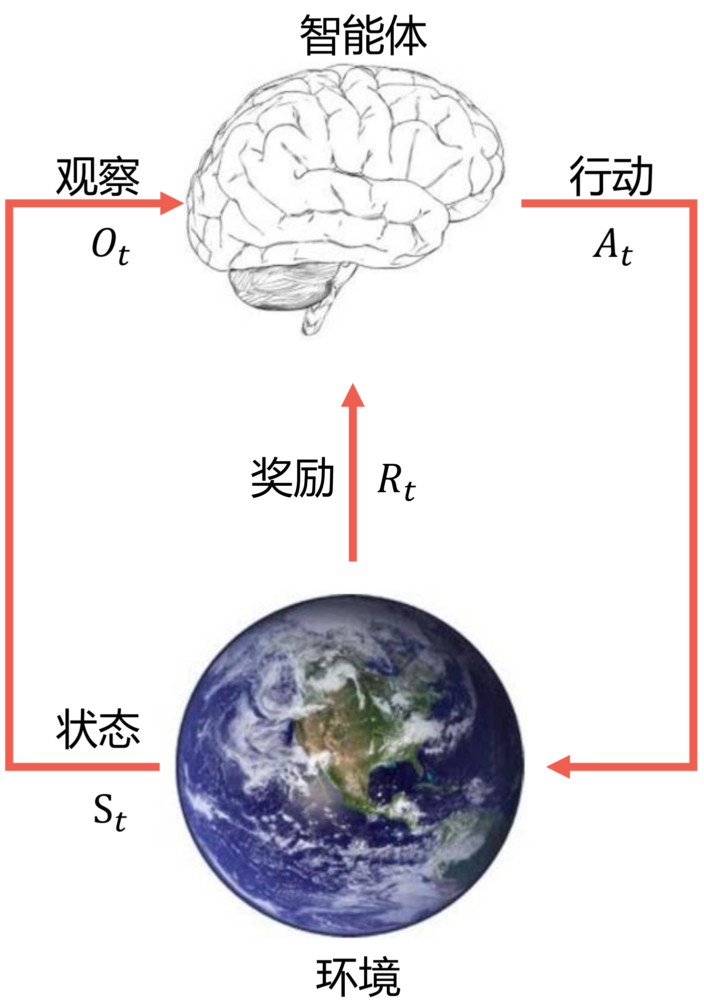
- 环境的模型（Model）用于模拟环境的行为

  - 预测下一个状态

    $$\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

  - 预测下一个（immediate 立即）奖励

    $$\mathcal{R}_{s}^{a} = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$



值函数估计

- 在基于模型的强化学习（MDP）中，值函数能够通过动态规划计算获得

$$
\begin{aligned}
V^{\pi}(s) &= \mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots | s_0 = s, \pi] \\
&= R(s) + \gamma \sum_{s' \in S} P_{s\,\pi(s)}(s') V^{\pi}(s')
\end{aligned}
$$

- 在模型无关的强化学习中
  - 我们无法直接获得 P_{sa} 和 R
  - 但是，我们拥有一系列可以用来估计值函数的经验

Episode 1: $s_0^{(1)} \xrightarrow[R(s_0)^{(1)}]{a_0^{(1)}} s_1^{(1)} \xrightarrow[R(s_1)^{(1)}]{a_1^{(1)}} s_2^{(1)} \xrightarrow[R(s_2)^{(1)}]{a_2^{(1)}} s_3^{(1)} \ldots s_T^{(1)}$

Episode 2: $s_0^{(2)} \xrightarrow[R(s_0)^{(2)}]{a_0^{(2)}} s_1^{(2)} \xrightarrow[R(s_1)^{(2)}]{a_1^{(2)}} s_2^{(2)} \xrightarrow[R(s_2)^{(2)}]{a_2^{(2)}} s_3^{(2)} \ldots s_T^{(2)}$

Analogy: Expected Age

Goal: Compute expected age of students

Known P(A)

$$E[A] = \sum_a P(a) \cdot a = 0.35 \times 20 + \ldots$$

Without P(A), instead collect samples $[a_1, a_2, \ldots a_N]$

Unknown P(A): "Model Based"

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$

$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

Why does this work? Because eventually you learn the right model.

Unknown P(A): "Model Free"

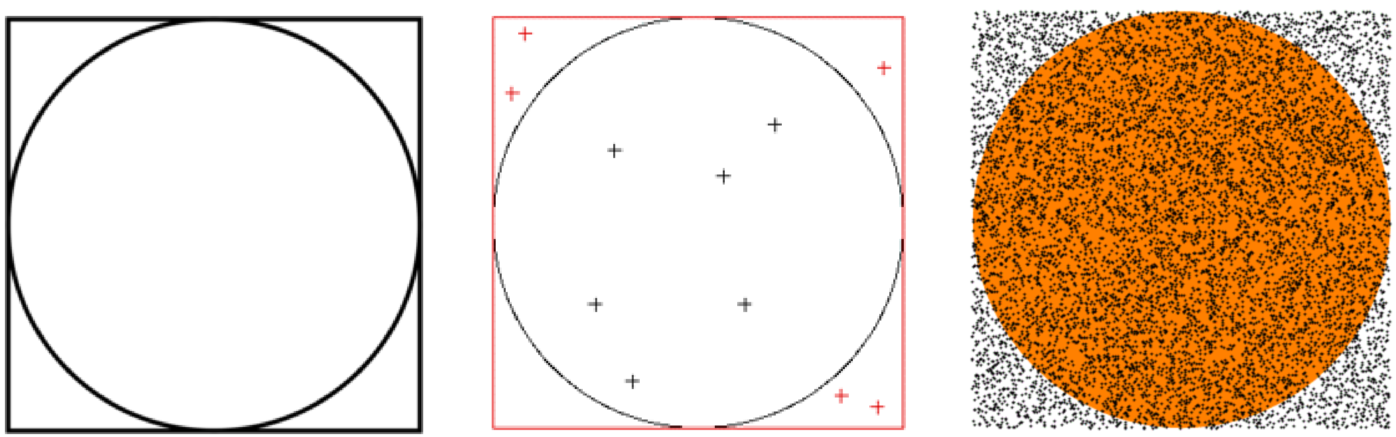
$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Why does this work? Because samples appear with the right frequencies.

蒙特卡洛方法

蒙特卡洛方法

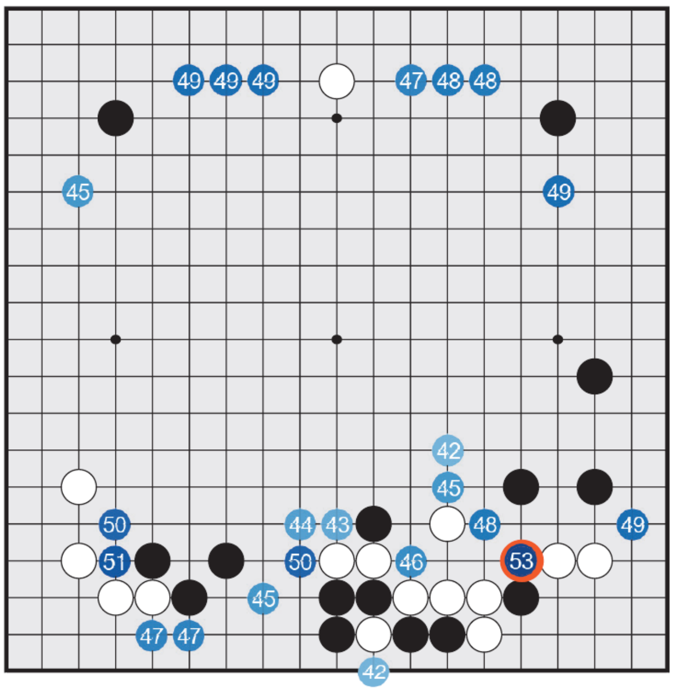
- 蒙特卡洛方法（Monte-Carlo methods）是一类广泛的计算算法。生活中处处都是MC方法。
  - 依赖于重复随机抽样来获得数值结果
- 例如，计算圆的面积

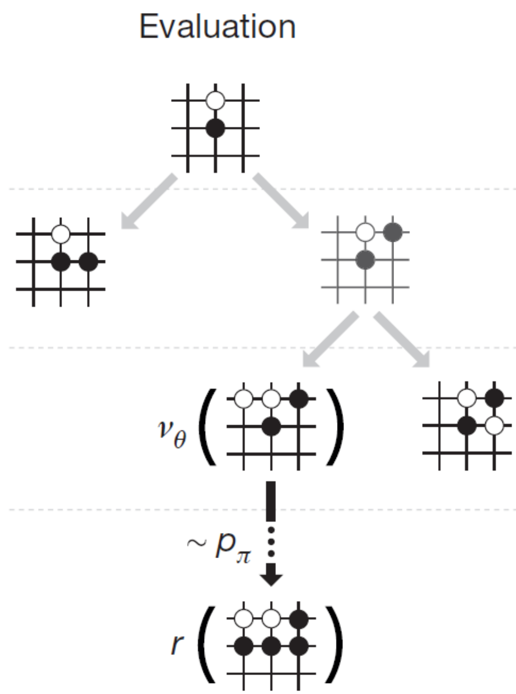


$$\text{Circle Surface} = \text{Square Surface} \times \frac{\#\text{points in circle}}{\#\text{points in total}}$$

蒙特卡洛方法

- 围棋对弈：估计当前状态下的胜率





$$\text{Win Rate}(s) = \frac{\#\text{win simulation cases started from } s}{\#\text{simulation cases started from } s \text{ in total}}$$

蒙特卡洛价值估计

- 目标：从策略 π 下的经验片段学习 V^{π}

$$s_0^{(i)} \xrightarrow[R_1^{(i)}]{a_0^{(i)}} s_1^{(i)} \xrightarrow[R_2^{(i)}]{a_1^{(i)}} s_2^{(i)} \xrightarrow[R_3^{(i)}]{a_2^{(i)}} s_3^{(i)} \dots s_T^{(i)} \sim \pi$$

- 回顾：累计奖励（return）是总折扣奖励

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

- 回顾：值函数（value function）是期望累计奖励

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots | s_0 = s, \pi] \\ &= \mathbb{E}[G_t | s_t = s, \pi] \\ &\simeq \frac{1}{N} \sum_{i=1}^{N} G_t^{(i)} \end{aligned}$$

- 使用策略π从状态s采样N个片段
- 计算平均累计奖励

- 蒙特卡洛策略评估使用经验均值累计奖励而不是期望累计奖励

蒙特卡洛价值估计

实现

- 使用策略π采样片段

$$S_0^{(i)} \xrightarrow[R_1^{(i)}]{a_0^{(i)}} S_1^{(i)} \xrightarrow[R_2^{(i)}]{a_1^{(i)}} S_2^{(i)} \xrightarrow[R_3^{(i)}]{a_2^{(i)}} S_3^{(i)} \ldots S_T^{(i)} \sim \pi$$

- 在一个片段中的每个时间步长t的状态s都被访问
  - 增量计数器 $N(s) \leftarrow N(s) + 1$
  - 增量总累计奖励 $S(s) \leftarrow S(s) + G_t$
  - 价值被估计为累计奖励的均值 $V(s) = S(s)/N(s)$
  - 由大数定率有

$$V(s) \rightarrow V^{\pi}(s) \ \text{as} \ N(s) \rightarrow \infty$$

增量蒙特卡洛更新

- 每个片段结束后逐步更新$V(s)$
- 对于每个状态S_t和对应累计奖励G_t

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

- 对于非稳定的问题（即，环境会随时间发生变化），我们可以跟踪一个现阶段的平均值（即，不考虑很久之前的片段）

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

α is the learning rate to allow more flexible trade-off between past and future ($\alpha > 1/N$?)

蒙特卡洛值估计

思路： $$V(S_t) \simeq \frac{1}{N}\sum_{i=1}^{N} G_t^{(i)}$$

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

- 蒙特卡洛方法：直接从经验片段进行学习
- 蒙特卡洛是模型无关的：未知马尔可夫决策过程的状态转移/奖励
- 蒙特卡洛从完整的片段中进行学习
- 蒙特卡洛采用最简单的思想：值（value）= 平均累计奖励（mean return）
- 注意：只能将蒙特卡洛方法应用于有限长度的马尔可夫决策过程中
  - 即，所有的片段都有终止状态

时序差分学习

时序差分学习（Temporal Difference Learning）

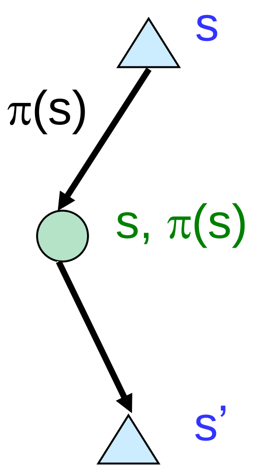
$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \simeq R_{t+1} + \gamma V(S_{t+1})$$

$$V(S_t) \leftarrow V(S_t) + \alpha\big(\underbrace{R_{t+1}}_{\text{观测值}} + \underbrace{\gamma V(S_{t+1})}_{\text{对未来的猜测}} - V(S_t)\big)$$

- 时序差分方法直接从经验片段中进行学习
- 时序差分是模型无关的
  - 不需要预先获取马尔可夫决策过程的状态转移/奖励
- 通过bootstrapping，时序差分从不完整的片段中学习
- 时序差分更新当前预测值使之接近估计累计奖励（非真实值）

Temporal Difference Value Learning

- Big idea: learn from every experience!
  - Update V(s) each time we experience a transition (s, a, s', r)
  - Likely outcomes s' will contribute updates more often



- Temporal difference learning of values
  - Policy fixed, doing evaluation!
  - Move values toward value of whatever successor occurs: running average

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

Update to V(s): $V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$

Same update: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

Gradient Descent View

$$f(x) = \frac{1}{2}(y - x)^2$$

$$\frac{df}{dx} = -(y - x)$$

- Goal: find x that minimizes $f(x)$

1. Start with initial guess, x_0
2. Update x by taking a step in the direction that $f(x)$ is changing fastest (in the negative direction) with respect to x:

   $x \leftarrow x - \alpha \nabla_x f$, where α is the step size or learning rate
3. Repeat until convergence

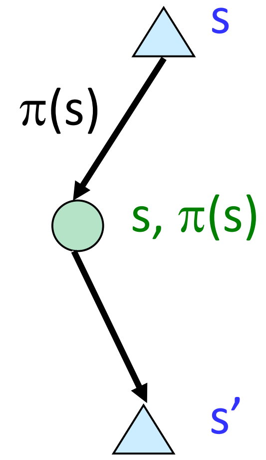
- TD goal: find value(s), V, that minimizes difference between sample(s) and V

$$V \leftarrow V - \alpha \nabla_V Error$$

$$Error(V) = \frac{1}{2}(sample - V)^2$$

Gradient Descent View 2

- Big idea: learn from every experience!
  - Update V(s) each time we experience a transition (s, a, s', r)
  - Likely outcomes s' will contribute updates more often



- Temporal difference learning of values
  - Policy fixed, doing evaluation!
  - Move values toward value of whatever successor occurs: running average

Sample of V(s): $sample = r + \gamma V^{\pi}(s')$

Update to V(s): $V^{\pi}(s) \leftarrow (1-\alpha) V^{\pi}(s) + (\alpha)\, sample$

Same update: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha\, [sample - V^{\pi}(s)]$

Same update: $V^{\pi}(s) \leftarrow V^{\pi}(s) - \alpha \nabla Error$

$$Error = \frac{1}{2}\left(sample - V^{\pi}(s)\right)^2$$

Exponential Moving Average

- Exponential moving average
  - The running interpolation update: $V_n = (1-\alpha)V_{n-1} + \alpha x_n$ with $V_1 = x_1$
  - Makes recent samples more important

$$V_n = \alpha x_n + \alpha(1-\alpha)x_{n-1} + \cdots + \alpha(1-\alpha)^{n-2}x_2 + (1-\alpha)^{n-1}x_1$$

  - Forgets about the past (distant past values were wrong anyway)

- Decreasing learning rate (alpha) can give converging averages
  - Note $V_n = \alpha_n x_n + (1-\alpha_n)\alpha_{n-1}x_{n-1} + \cdots$

$$+(1-\alpha_n)(1-\alpha_{n-1})\cdot\cdots\cdot(1-\alpha_3)\alpha_2 x_2$$
$$+(1-\alpha_n)(1-\alpha_{n-1})\cdot\cdots\cdot(1-\alpha_3)(1-\alpha_2)x_1$$

蒙特卡洛 vs. 时序差分（MC vs. TD）

相同的目标：从策略π下的经验片段学习V^{π}

- 增量地进行每次蒙特卡洛过程（MC）
  - 更新值函数$V(S_t)$使之接近样本累计奖励G_t

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

- 最简单的时序差分学习算法（TD）：
  - 更新$V(S_t)$使之接近估计累计奖励$R_{t+1} + \gamma V(S_{t+1})$

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

  - 时序差分目标：$R_{t+1} + \gamma V(S_{t+1})$
  - 时序差分误差：$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

蒙特卡洛（MC）和时序差分（TD）的优缺点

- 时序差分：能够在知道最后结果之前进行学习
  - 时序差分能够在每一步之后进行在线学习
  - 蒙特卡洛必须等待片段结束，直到累计奖励已知

- 时序差分：能够无需最后结果地进行学习
  - 时序差分能够从不完整的序列中学习
  - 蒙特卡洛只能从完整序列中学习
  - 时序差分可以在连续（无终止的）环境下工作
  - 蒙特卡洛只能在片段化的（有终止的）环境下工作

偏差（Bias）/方差（Variance）的权衡

- 累计奖励$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$是$V^\pi(S_t)$的无偏估计
- 时序差分真实目标$R_{t+1} + \gamma V^\pi(S_{t+1})$是$V^\pi(S_t)$的无偏估计
- 时序差分目标$R_{t+1} + \gamma \underbrace{V(S_{t+1})}_{\text{当前估计}}$是$V^\pi(S_t)$的有偏估计
- 时序差分目标具有比累计奖励更低的方差
  - 累计奖励——取决于多步随机动作，多步状态转移和多步奖励
  - 时序差分目标——取决于单步随机动作，单步状态转移和单步奖励

蒙特卡洛（MC）和时序差分（TD）的优缺点（2）

MC:

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

蒙特卡洛高方差，无偏

- 良好的收敛性质
  - 使用函数近似时依然如此
- 对初始值不敏感
- 易于理解和使用

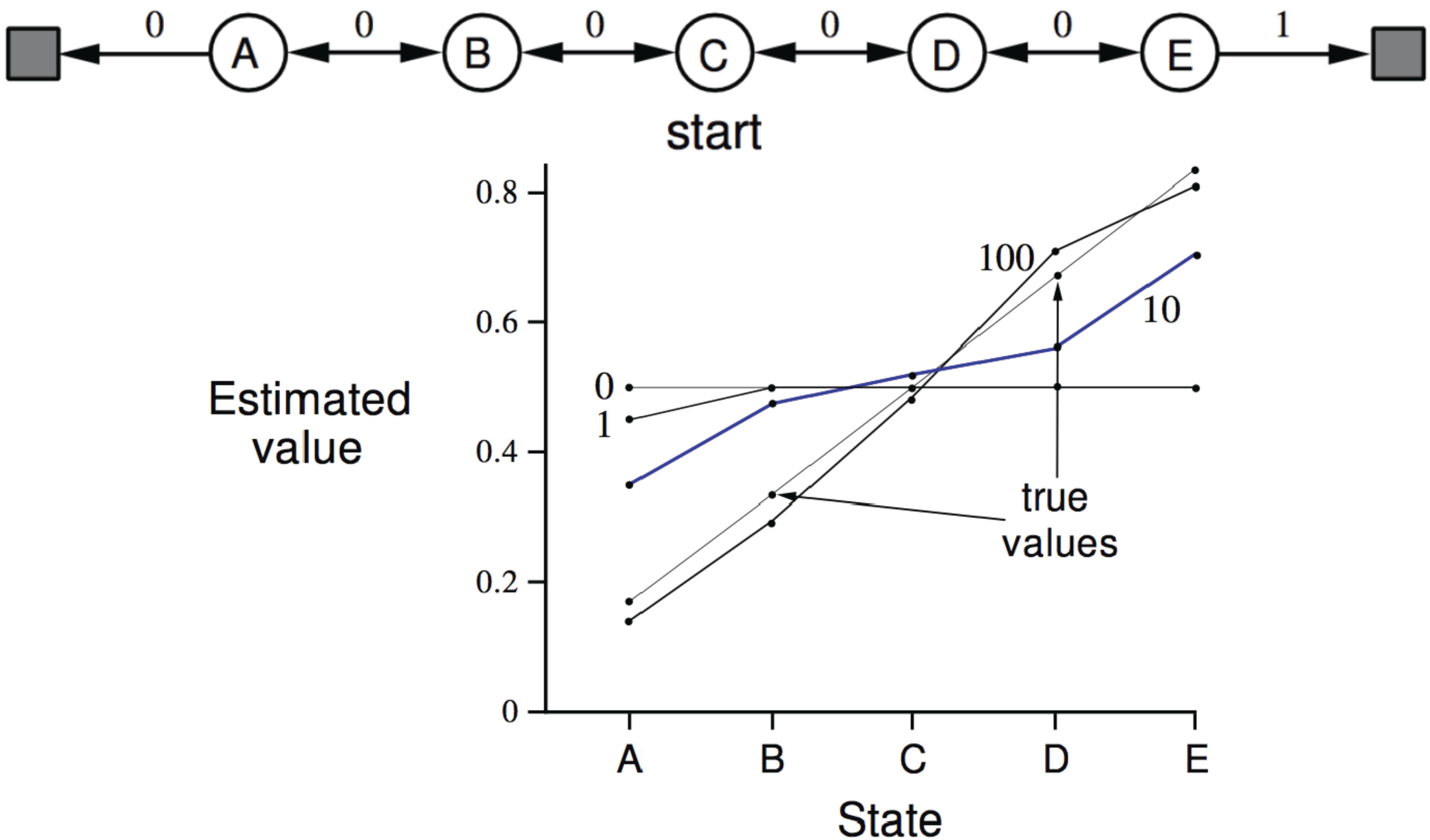
TD:

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

时序差分低方差，有偏

- 通常比蒙特卡洛更加高效
- 时序差分最终收敛到$V^{\pi}(S_t)$
  - 但使用函数近似并不总是如此
- 比蒙特卡洛对初始值更加敏感

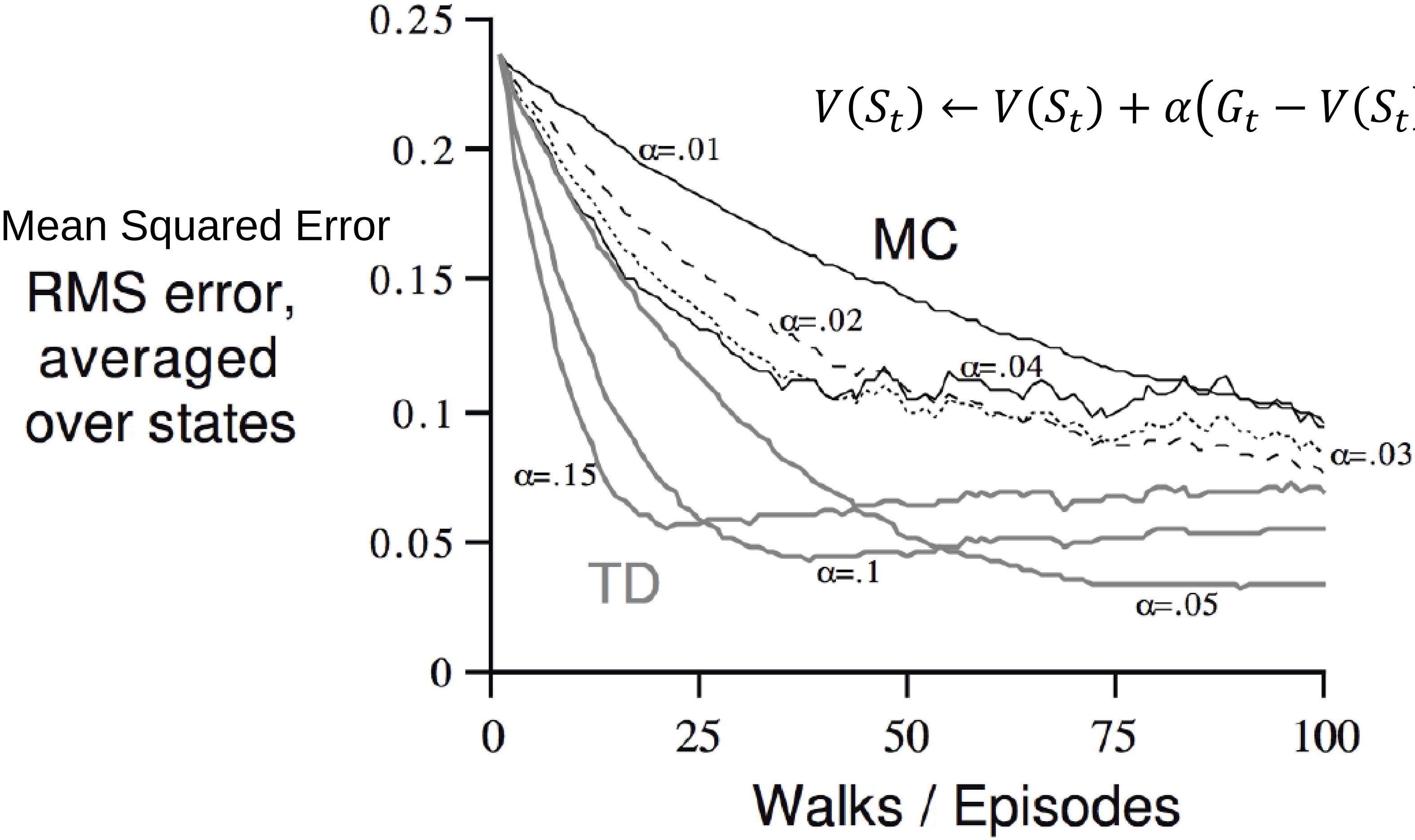
随机游走的例子



0
A
0
B
0
C
0
D
0
E
1
start
Estimated value
0.8
0.6
0.4
0.2
0
100
10
0
1
true values
A
B
C
D
E
State

随机游走的例子

Root Mean Squared Error

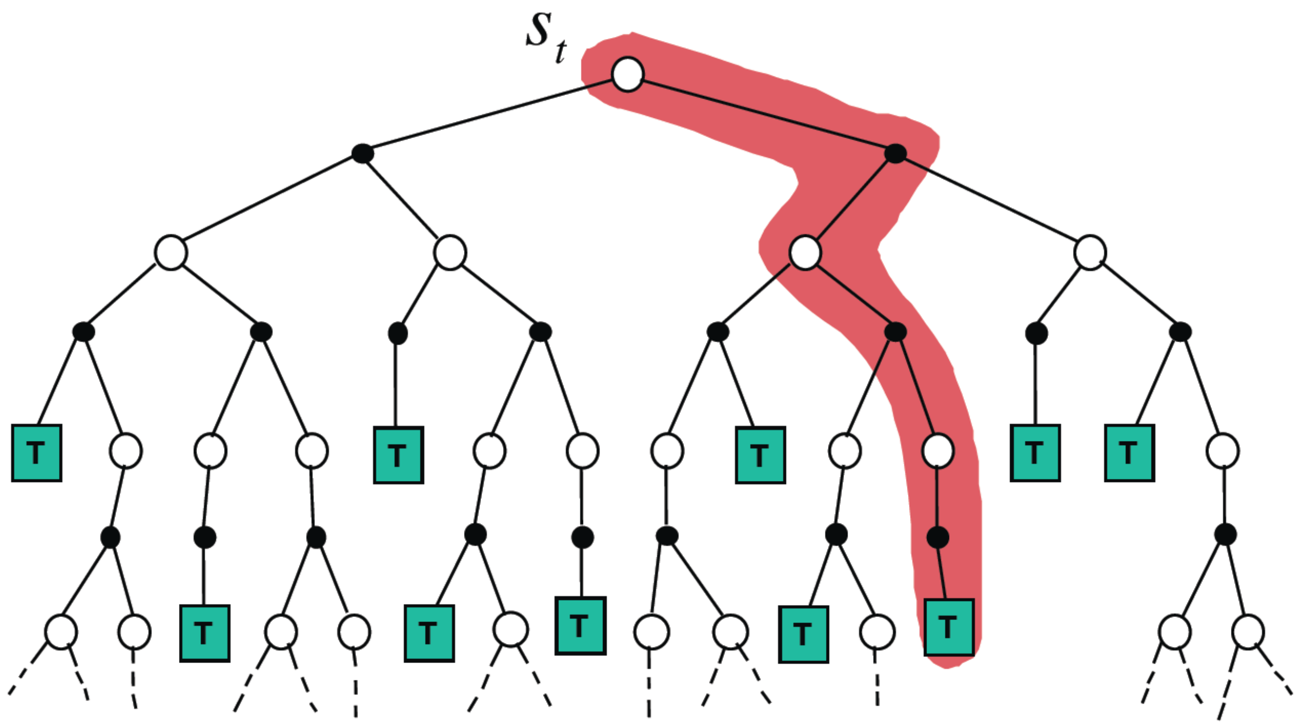


$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$

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

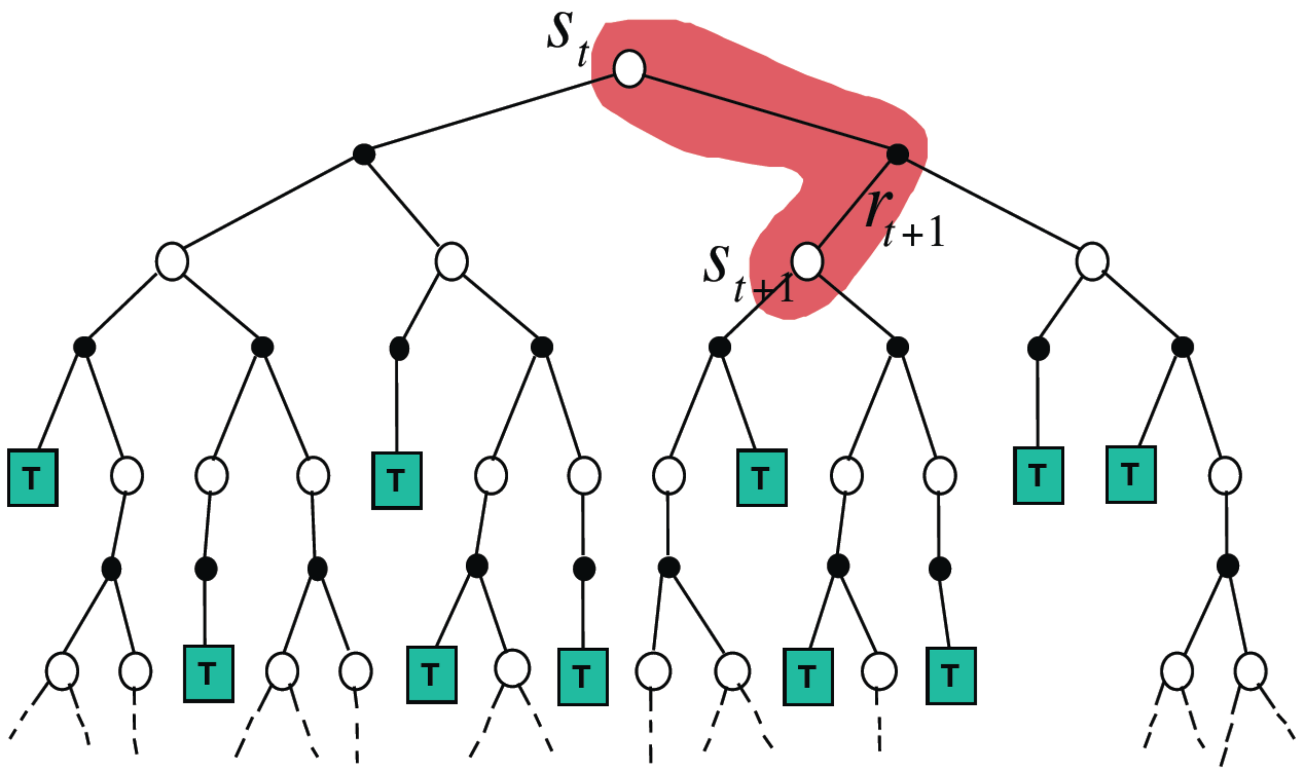
蒙特卡洛反向传播（Backup）

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



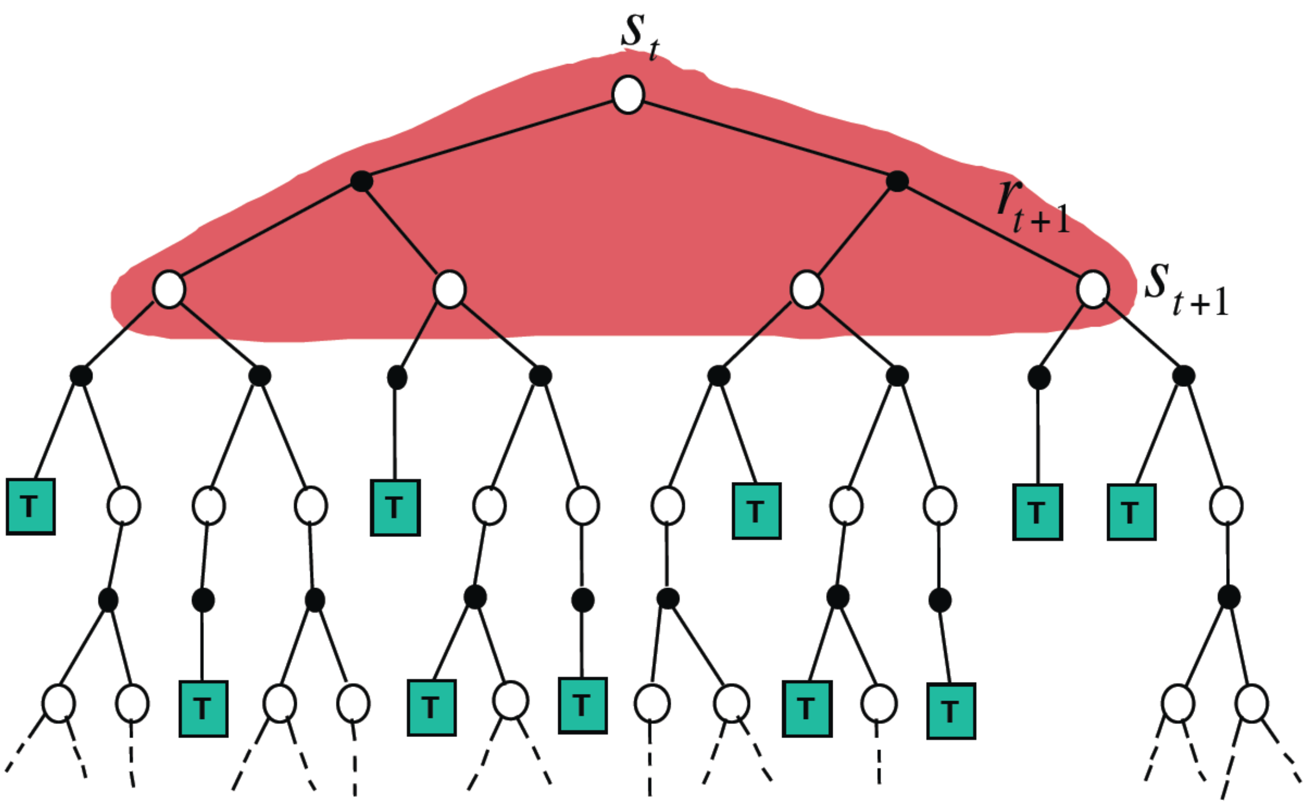
时序差分反向传播（Backup）

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



动态规划反向传播（Backup）

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



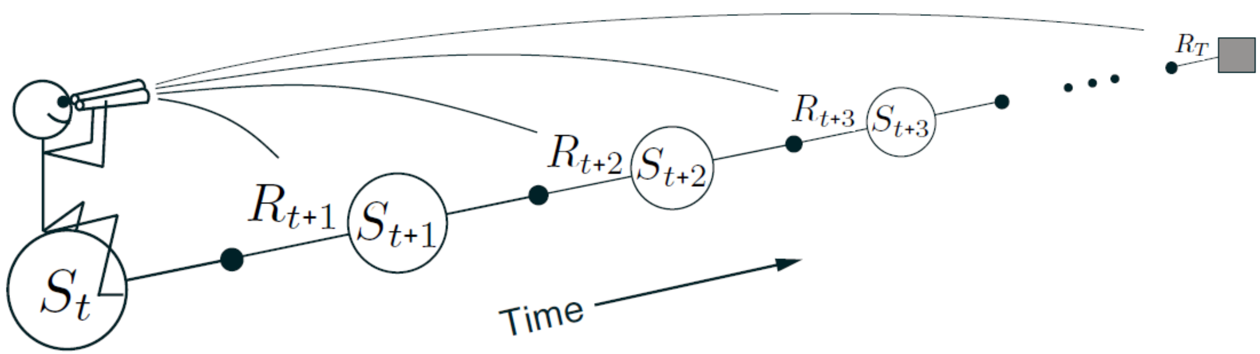
多步时序差分学习

- 定义n步累计奖励

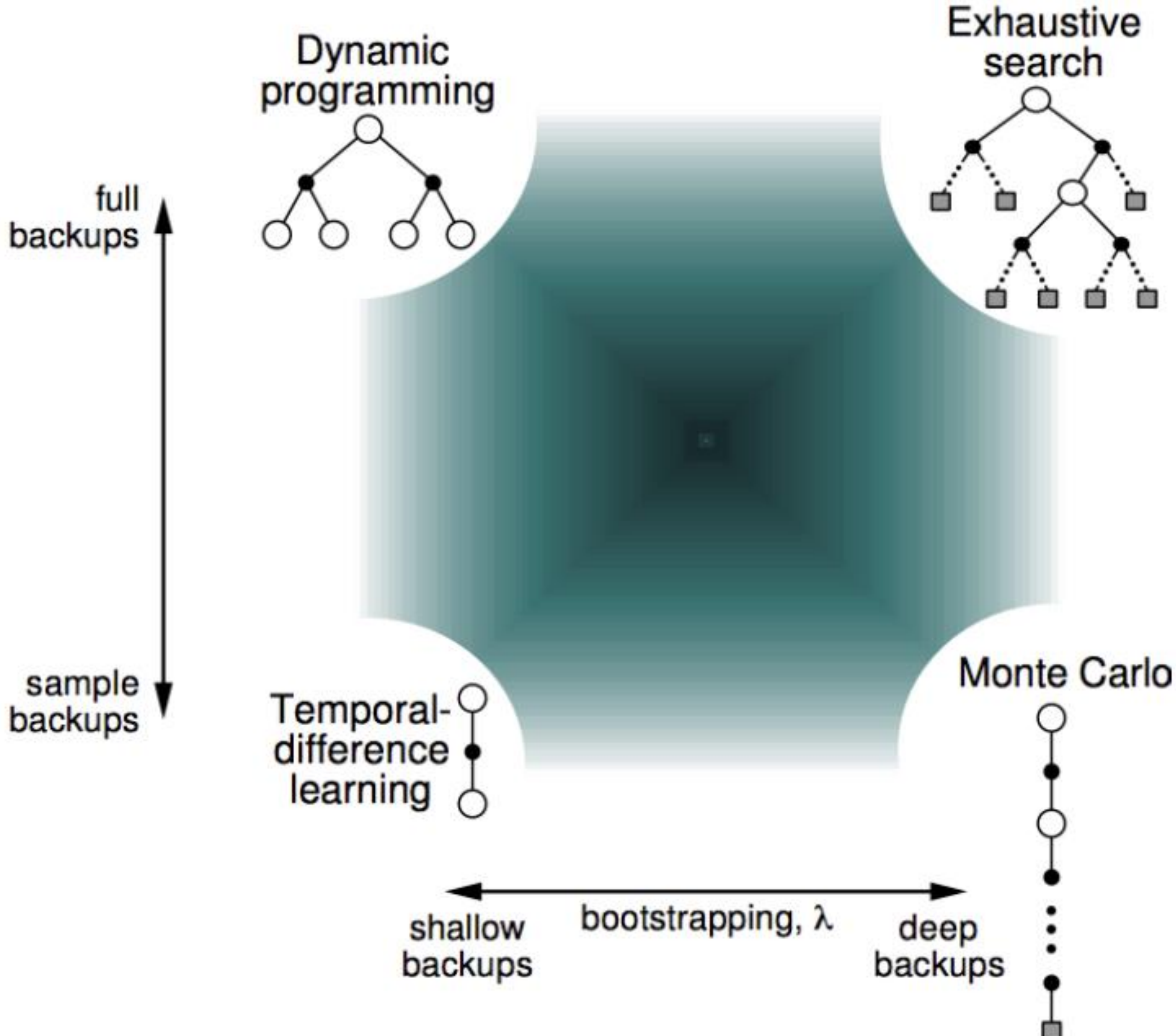
$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

- n步时序差分学习

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



总览强化学习值函数估计多种方法



Slide credit: David Silver

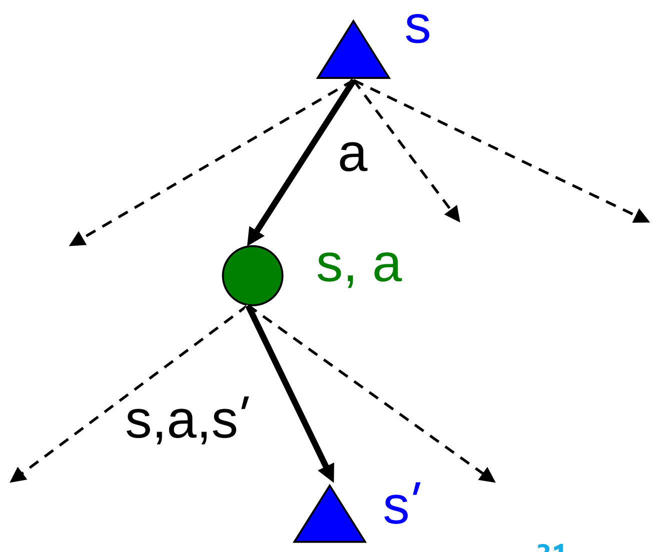
Problems with TD Value Learning

- TD value leaning is a model-free way to do policy evaluation, mimicking Bellman updates with running sample averages
- However, if we want to turn values into a (new) policy, we're sunk:

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

$$Q(s,a) = \sum_{s'} P_{s,a}(s')\left[R(s,a,s') + \gamma V(s')\right]$$

- Idea: learn Q-values, not values
- Makes action selection model-free too!



重要性采样

重要性采样

- 估计一个不同分布的期望

$$\begin{aligned}
\mathbb{E}_{x\sim p}[f(x)] &= \int_x p(x)f(x)dx \\
&= \int_x q(x)\frac{p(x)}{q(x)}f(x)dx \\
&= \mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)}f(x)\right]
\end{aligned}$$

- 将每个实例的权重重新分配为 $\beta(x) = \frac{p(x)}{q(x)}$

使用重要性采样的离线策略蒙特卡洛

- 使用策略μ产生的累计奖励评估策略π
- 根据两个策略之间的重要性比率（importance ratio）对累计奖励G_t加权
- 每个片段乘以重要性比率

$$\{s_1, a_1, r_2, s_2, a_2, \dots, s_T\} \sim \mu$$

$$G_t^{\pi/\mu} = \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)} \frac{\pi(a_{t+1}|s_{t+1})}{\mu(a_{t+1}|s_{t+1})} \cdots \frac{\pi(a_T|s_T)}{\mu(a_T|s_T)} G_t$$

使用重要性采样的离线策略蒙特卡洛

- 更新值函数以逼近修正的累计奖励值

$$V(s_t) \leftarrow V(s_t) + \alpha\left(G_t^{\pi/\mu} - V(s_t)\right)$$

无法在π非零而μ为零时使用

重要性采样将显著增大方差（variance）

使用重要性采样的离线策略时序差分

- 使用策略μ产生的时序差分目标评估策略π
- 根据重要性采样对时序差分目标$r + \gamma V(s')$加权
- 仅需要一步来进行重要性采样修正

$$V(s_t) \leftarrow V(s_t) + \alpha\left(\underbrace{\frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}(r_{t+1} + \gamma V(s_{t+1}))}_{\text{时序差分目标}} - V(s_t)\right)$$

重要性
采样修正

时序差分目标

具有比蒙特卡洛重要性采样更低的方差

策略仅需在单步中被近似

值函数估计总结

- 无模型的强化学习在黑盒环境下使用
- 要优化智能体策略，首要任务则是精准、高效地估计状态或者(状态、动作)的价值
- 在黑盒环境下，值函数的估计方法主要包括蒙特卡洛方法和时序差分法
- 蒙特卡洛方法通过采样到底的方式直接估计价值函数
- 时序差分学习通过下一步的价值估计来更新当前一步的价值估计
- 实际使用中，时序差分方法更加常见

THANK YOU