多智能体系统与强化学习

主讲人：高阳、杨林、杨天培

https://reinforcement-learning-2025.github.io/

第二讲：时差学习

强化学习问题和范式

高　阳

大 纲

Bootstraps和Sampling

强化学习算法设计

N步回退学习

大 纲

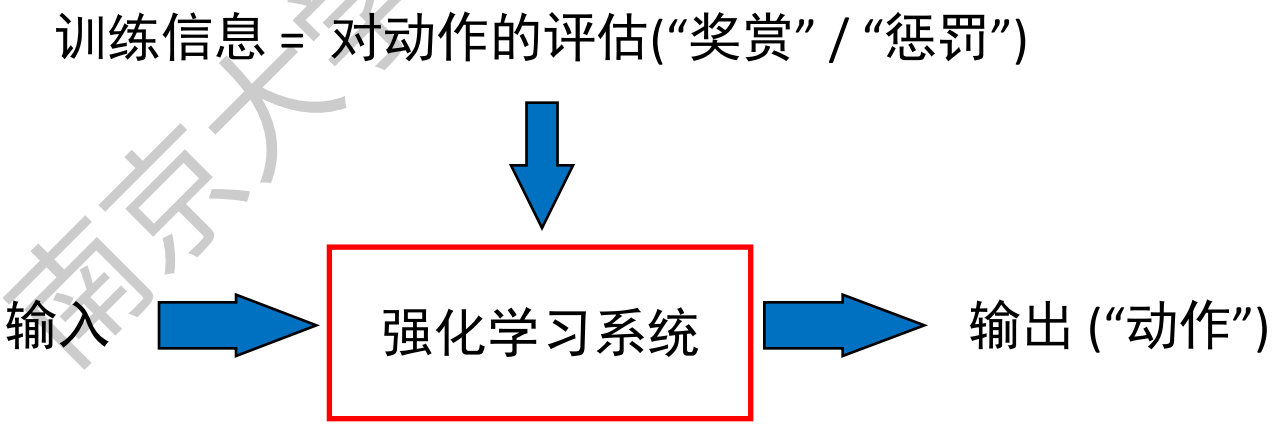
Bootstraps和Sampling

强化学习算法设计

N步回退学习

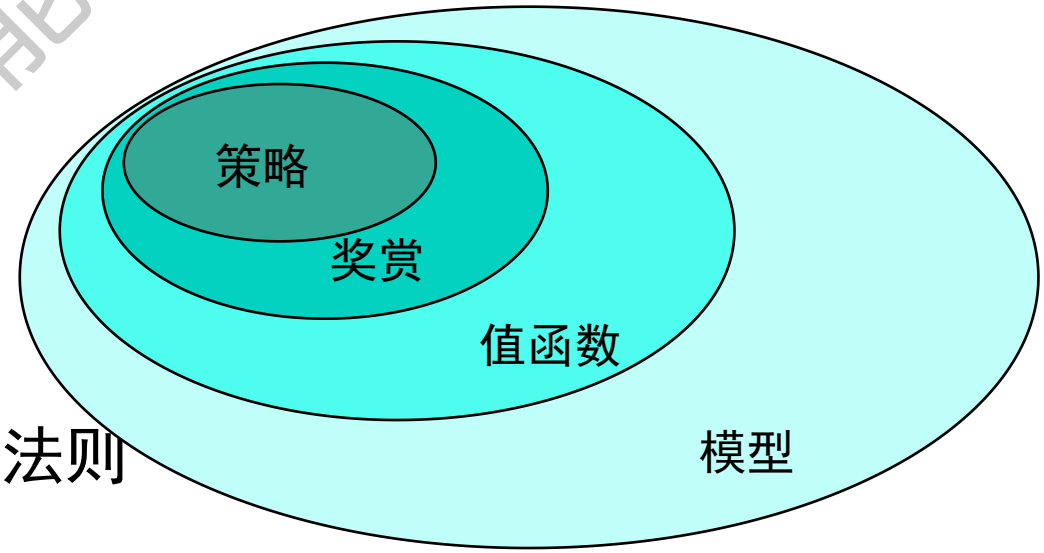
监督学习 VS 强化学习

- 监督学习
  - ✓ (正/反例)在样本上的分布是确定的
- 强化学习
  - ✓ (状态/奖赏)的分布是策略依赖的(policy dependent!!!)
  - ✓ 策略上小的变化都会导致返回值的巨大改变



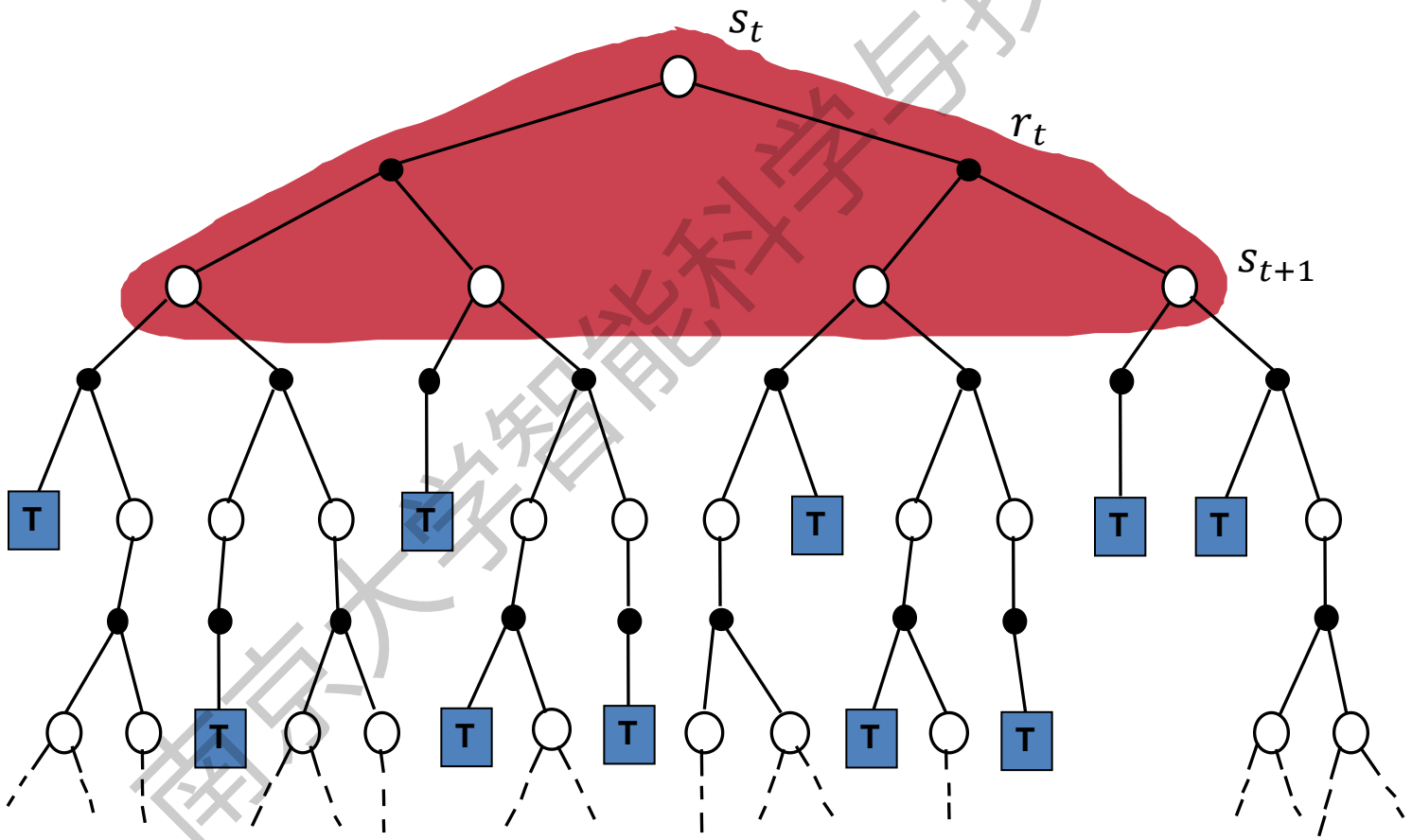
强化学习要素

- 策略
  - ✓ 选择动作的(确定/不确定)规则
- 奖赏/返回
  - ✓ 学习系统试图最大化的函数
- 值函数
  - ✓ 评估策略好坏的函数
- 模型
  - ✓ 环境(问题)演变遵循的法则



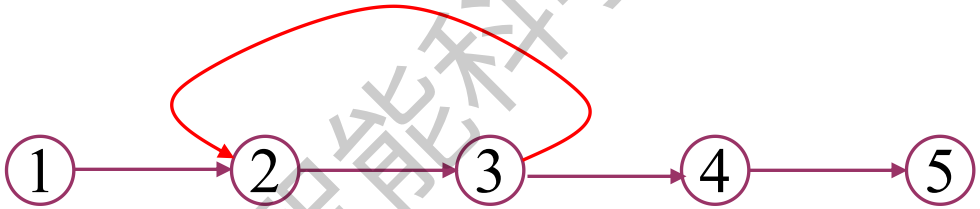
动态规划方法

$$V(s_t) \leftarrow E_\pi\{r_t + \gamma V(s_{t+1})\}$$



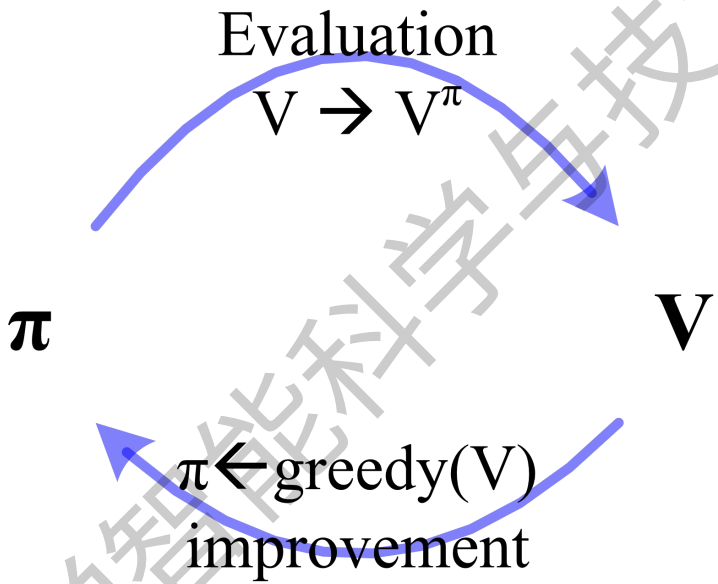
Monte Carlo策略评价

- 目标：学习$V^{\pi}(s)$；
- 给定：在访问状态s，采用策略下π，获得的若干经验；
- 思路：在访问状态s 后，对所获得的返回，进行平均。



- Every-Visit MC:
  - ✓ 在一次经验中，对每次访问到的s都进行平均
- First-visit MC
  - ✓ 在一次经验中，只对首次访问到的s进行平均

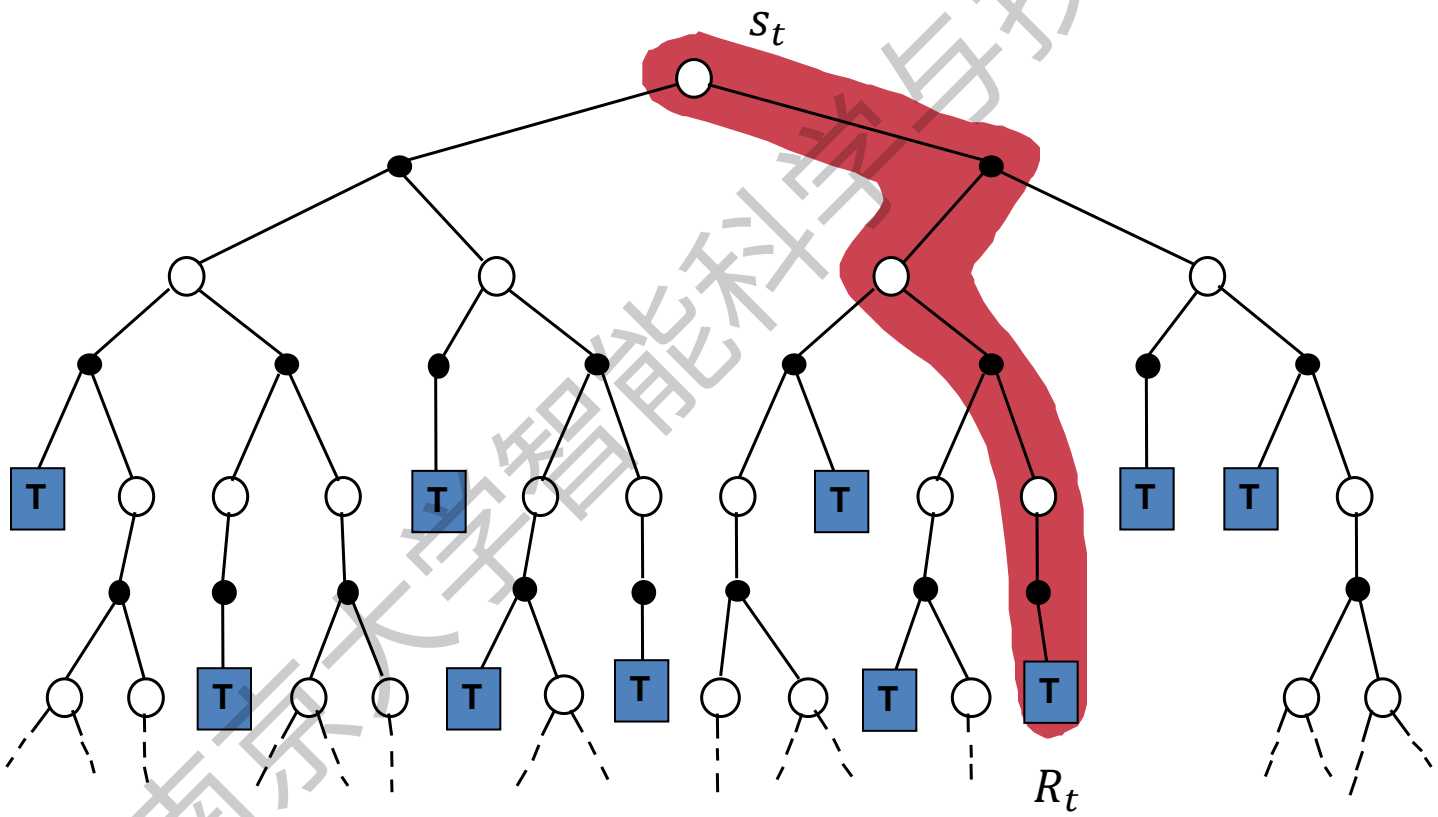
Monte Carlo最优控制



- MC策略迭代：使用MC方法对策略进行评估，计算值函数；
- MC策略修正：根据值函数(或者状态-动作对值函数)，采用贪心策略进行策略修正；

Monte Carlo方法

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)]$$



时差学习

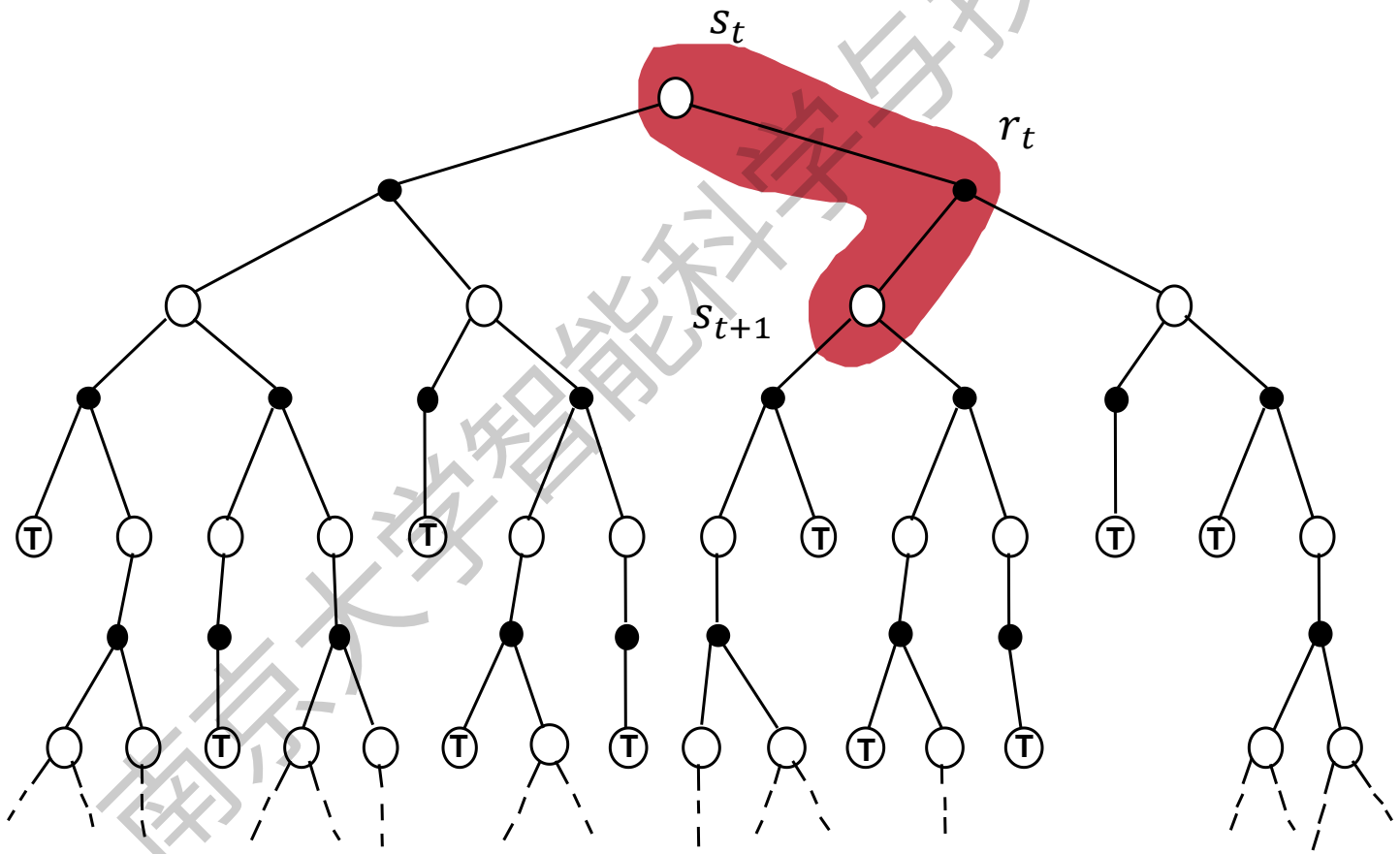
- Every-Visit MC:
  - ✓ $V(s_t) = V(s_t) + \alpha[R_t - V(s_t)]$

  - ✓ 目标：在经过若干次平均后，得到真实的返回值
- 最简单的时间差分方法(Temporal Difference)
  - ✓ $V(s_t) = V(s_t) + \alpha[r_t + \gamma V(s_{t+1}) - V(s_t)]$

  - ✓ 目标：在每一次经验后，都对返回值进行估计

时差方法

$$V(s_t) \leftarrow V(s_t) + \alpha[r_t + \gamma V(s_{t+1}) - V(s_t)]$$



Bootstraps和Sampling

- Bootstraps
  - ✓ 通过一个估计值进行更新
  - ✓ 动态规划/时差学习中采用
  - ✓ 蒙特卡罗方法不采用
- 采样
  - ✓ 不通过估计值进行更新，而根据经验进行更新
  - ✓ 蒙特卡罗方法/时差学习中采用
  - ✓ 动态规划中不采用

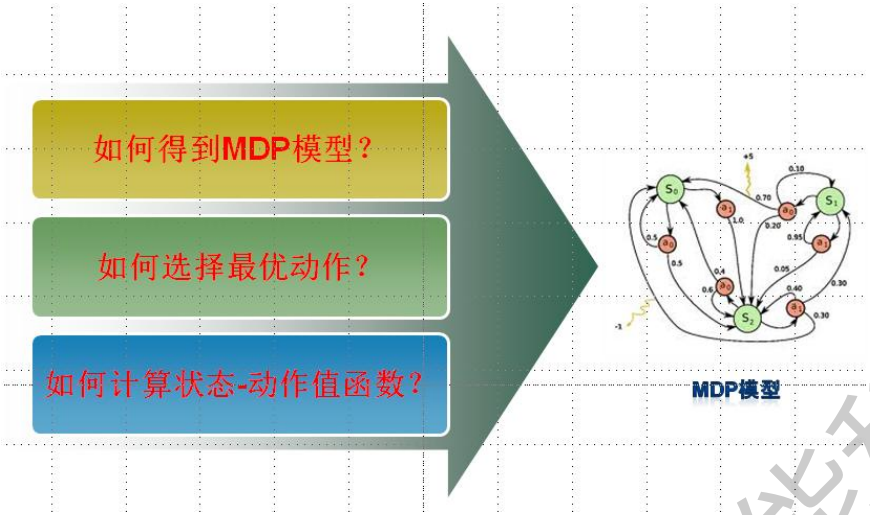
大 纲

Bootstraps和Sampling

强化学习算法设计

N步回退学习

强化学习算法



- 值函数
  - $V^{\pi}(s) = E_{s' \sim \pi(s)}\left[R\left(s, \pi(s)\right) + \gamma V^{\pi}(s')\right]$
- 状态-动作对值函数
  - $Q^{\pi}(s,a) = E[R(s,a)] + \gamma \sum_{s'} \delta(s,a,s')\, V^{\pi}(s')$
  - 其中，$V^{\pi}(s) = Q^{\pi}\left(s, \pi(s,a)\right)$

算法构造思路

- ✓ 根据先验得到初始认知(值函数)
- ✓ 根据认知选择动作(伴随一定的随机性)
- ✓ 获得经验
- ✓ 根据反馈，修改认知
- ✓ 根据延迟的反馈，回退修改历史认知

算法设计的核心要素

- 值函数的表达(V, Q)
- 实现随机的动作选择(探索和利用)
- 值函数更新(在策略、离策略)
- 代表性的学习算法(SARSA, Q, AC)

值函数的表达

- 值函数
  - $V^{\pi}(s) = E_{s' \sim \pi(s)}\left[R\left(s, \pi(s)\right) + \gamma V^{\pi}(s')\right]$
- 状态-动作对值函数
  - $Q^{\pi}(s,a) = E[R(s,a)] + \gamma \sum_{s'} \delta(s,a,s')\, V^{\pi}(s')$
  - 其中，$V^{\pi}(s) = Q^{\pi}\left(s, \pi(s,a)\right)$

随机的动作选择

- 目的
  - 实现探索(Exploration)和利用(Exploitation)的平衡
  - 原则：算法初期倾向于探索，后期强调利用

ε-贪心策略

- ✓ 以1- ε概率选择，$\pi(s) = argmax_a Q^{\pi}(s,a)$
- ✓ 以ε概率选择其他动作
- ✓ ε随学习的episodic次数下降

SARSA：在策略TD学习

- 在策略(on-policy): 对于当前状态值函数，采用下一状态所选择动作的值函数(依据当前Policy)进行更新

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Initialize $Q(s, a)$ arbitrarily
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)
    $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma Q(s', a') - Q(s, a)\right]$
    $s \leftarrow s'$; $a \leftarrow a'$;
  until s is terminal

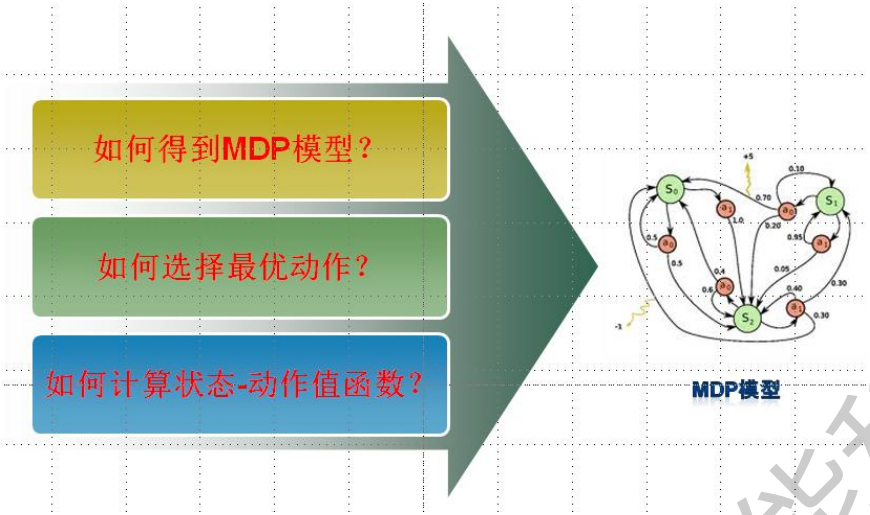
Q-Learning：离策略TD学习

- 离策略(off-policy): 对于当前状态值函数，采用下一状态值函数最大值进行更新

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Initialize $Q(s, a)$ arbitrarily
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'
        $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right]$
        $s \leftarrow s'$;
    until s is terminal

强化学习算法



- 值函数
  - $V^{\pi}(s) = E_{s' \sim \pi(s)}\left[R\left(s, \pi(s)\right) + \gamma V^{\pi}(s')\right]$
- 状态-动作对值函数
  - $Q^{\pi}(s,a) = E[R(s,a)] + \gamma \sum_{s'} \delta(s,a,s')\, V^{\pi}(s')$
  - 其中，$V^{\pi}(s) = Q^{\pi}\left(s, \pi(s,a)\right)$

算法构造思路

- ✓ 根据先验得到初始认知(值函数)
- ✓ 根据认知选择动作(伴随一定的随机性)
- ✓ 获得经验
- ✓ 根据反馈，修改认知
- ✓ 根据延迟的反馈，回退修改历史认知

Initialize $Q(s,a)$ arbitrarily and $e(s,a) = 0$, for all s, a
Repeat (for each episode):
  Initialize s, a
  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)
    $a^* \leftarrow \arg\max_b Q(s', b)$ (if a' ties for the max, then $a^* \leftarrow a'$)
    $\delta \leftarrow r + \gamma Q(s', a^*) - Q(s,a)$
    $e(s,a) \leftarrow e(s,a) + 1$
    For all s, a:
      $Q(s,a) \leftarrow Q(s,a) + \alpha \delta e(s,a)$
      If $a' = a^*$, then $e(s,a) \leftarrow \gamma \lambda e(s,a)$
                    else $e(s,a) \leftarrow 0$
    $s \leftarrow s'$; $a \leftarrow a'$
  until s is terminal

大 纲

Bootstraps和Sampling

强化学习算法设计

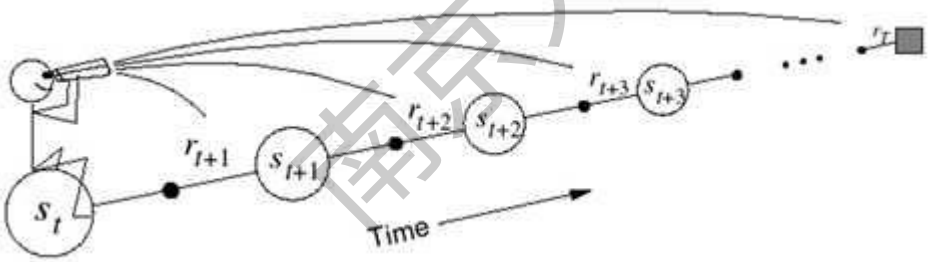
N步回退学习

N步TD预测

- 思路：当做TD回退时，可以看到“更远的未来”；

  - 目的
    - 实现探索(Exploration)和利用(Exploitation)的平衡
    - 原则：算法初期倾向于探索，后期强调利用

- ε-贪心策略
  - ✓ 以1- ε概率选择，$\pi(s) = argmax_a Q^{\pi}(s,a)$
  - ✓ 以ε概率选择其他动作



N步TD预测

- Every-Visit MC:
  - ✓ $V(s_t) = V(s_t) + \alpha[R_t - V(s_t)]$
  - ✓ $R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^T r_{t+T}$
- TD(0)
  - ✓ $V(s_t) = V(s_t) + \alpha[r_t + \gamma V(s_{t+1}) - V(s_t)]$

$\widehat{R_t}$

- TD(n)
  - ✓ 2步：$R^{(2)}_t = r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2})$
  - ✓ N步：$R^{(n)}_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n V(s_{t+n})$

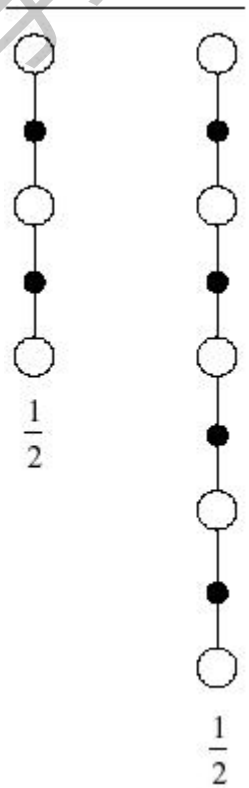
N步回退学习

N步回退

✓ $\Delta V(s_t) = \alpha\left[R^{(n)}_t - V(s_t)\right]$

✓ $R^{avg}_t = \frac{1}{2}R^{(n)}_2 + \frac{1}{2}R^{(n)}_4$

两次多步回退的平均

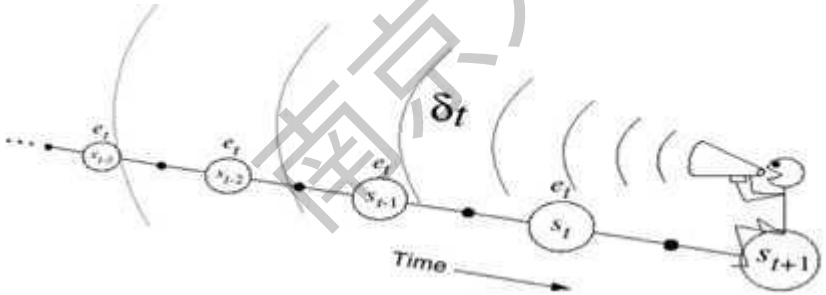


N步回退学习

λ-返回

✓ $R^{\lambda}{}_{t} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}R^{(n)}{}_{t}$

✓ $\Delta V(s_t) = \alpha\left[R^{(\lambda)}{}_{t} - V(s_t)\right]$



TD(λ)算法

1. 初始化V(s), e(s)=0
2. 对每一个episode，重复

   初始化s

   对episode中的每一步

   根据ε-贪心策略选择动作a

   执行动作a，获得r和s'

   $\Delta \leftarrow r + \gamma V(s') - V(s)$

   $e(s) \leftarrow e(s)+1$

   对于所有s

   $V(s) \leftarrow V(s)+\alpha\Delta e(s)$

   $e(s) \leftarrow \gamma\lambda e(s)$

   $s \leftarrow s'$

   直到s为终止状态

思考和讨论

1. 回报函数和值函数
2. 动态规划和蒙特卡罗采样的区别
3. 区分SARSA和Q学习算法的区别
4. 学习TD(λ)算法

实验

1. 实现TD和TD(λ)算法，并实现http://mnemstudio.org/path-finding-q-learning-tutorial.htm所描述的例子 (此实验不做)

思考和讨论

1. 理解强化学习范式和概念学习的不同
2. 理解离散型MDP模型中各个要素
3. 掌握动态规划方法

谢 谢！