强化学习

黄政宇
北京大学北京国际数学研究中心
北京大学国际机器学习研究中心

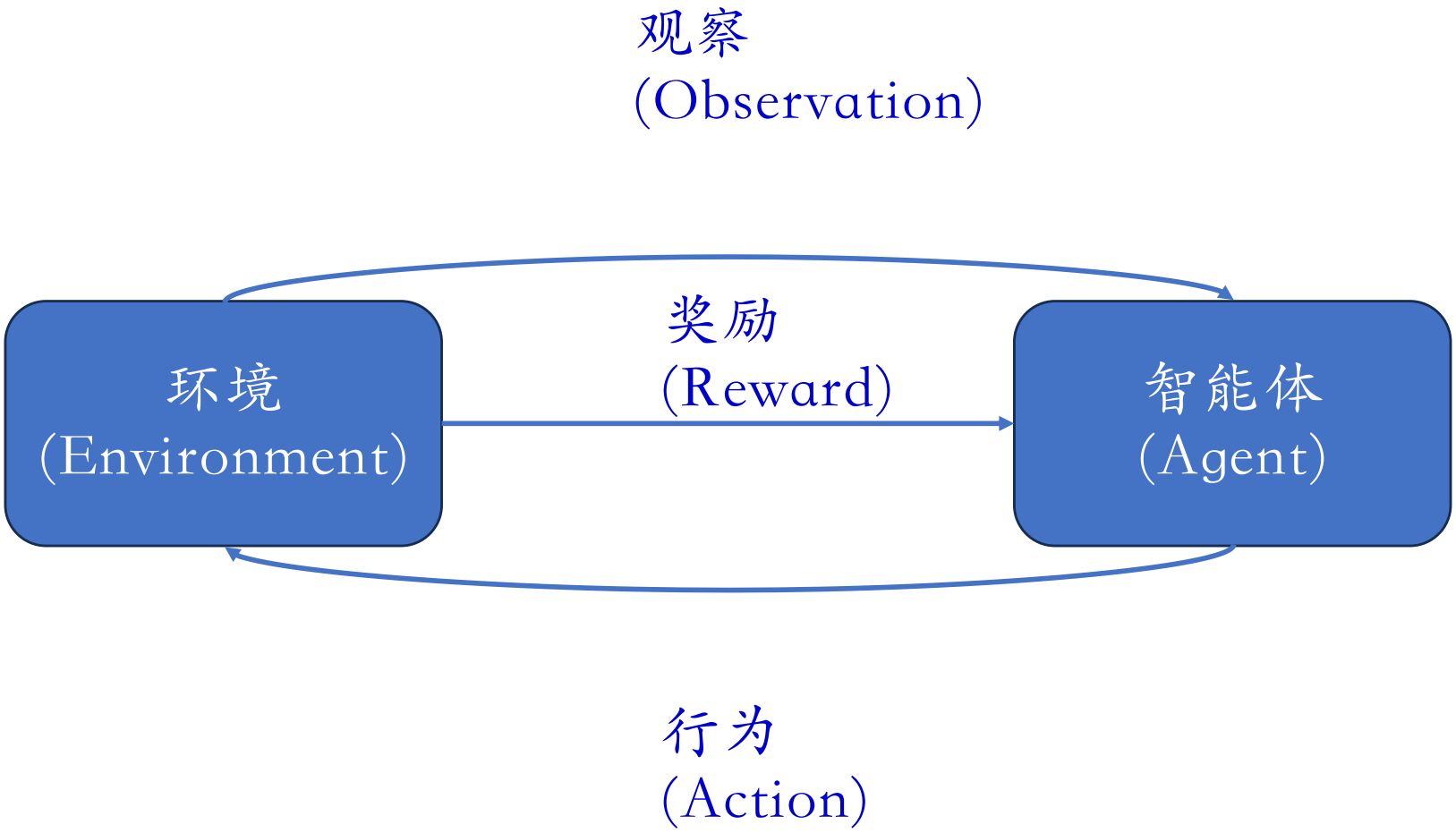
本堂课大纲

- 强化学习简介
  - 马尔科夫决策过程
  - 例子

- 强化学习方法
  - 值迭代
  - 策略梯度算法
  - 蒙特卡洛树搜索

强化学习

➢ 智能体与环境交互



马尔科夫决策过程

- 马尔科夫决策过程(Markov Decision Process)

状态空间：$\mathcal{S}$
动作空间：$\mathcal{A}$

$p: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$， $p(s'|s,a)$ 是在执行动作 a 的情况下，从状态 s 到达状态 s' 的概率

轨道：$\tau = (s_0, a_0, s_1, a_1, \cdots)$
奖励：$R(\tau)$， 给定（状态，动作）的情况下，奖励的分布

马尔科夫决策过程

- 马尔科夫决策过程(Markov Decision Process)

  时刻0 ：$s_0 \sim p(s_0)$
  智能体选择动作： a_t
  环境采样奖励：$r_t \sim R(\cdot\,|s_t, a_t)$
  环境采样下一个状态： $s_{t+1} \sim p(\cdot\,|s_t, a_t)$
  智能体得到奖励 r_t进入状态 s_{t+1}

- 策略优化

  策略函数：$\pi: \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$ $\pi(s,a)$: 是状态s下执行动作a的概率
  目标是找到最佳决策该策略使累计折扣奖励最大化($\gamma \leq 1$)

$$\sum_{t=1} \gamma^t r_t$$

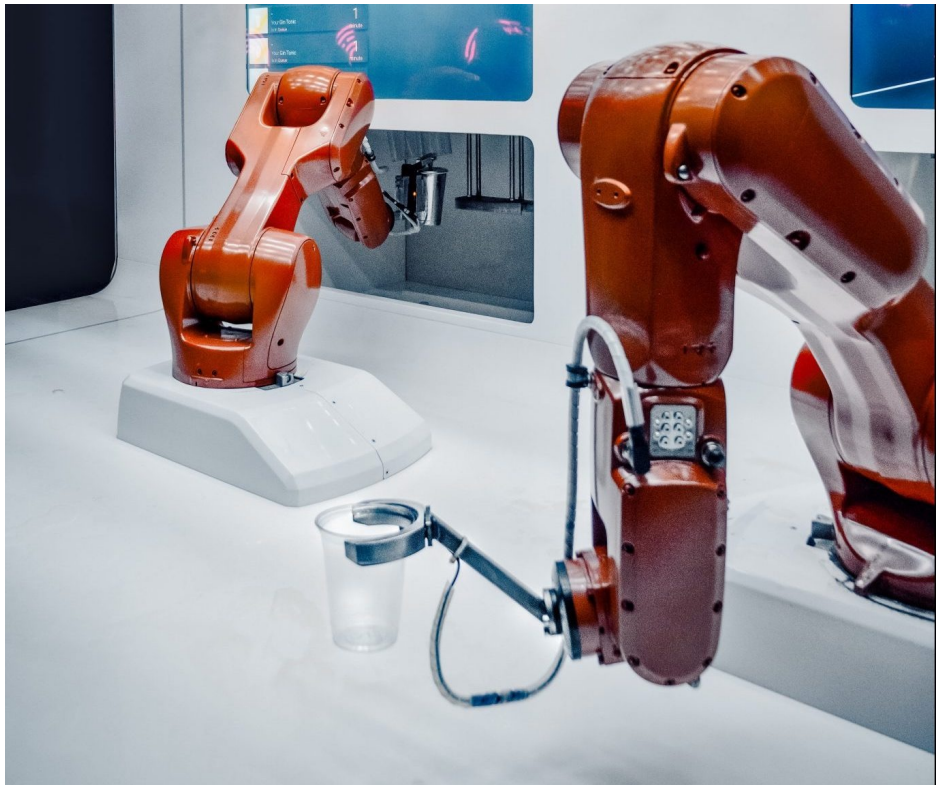
例子

- 围棋

环境
智能体
状态空间
动作空间
奖励

例子

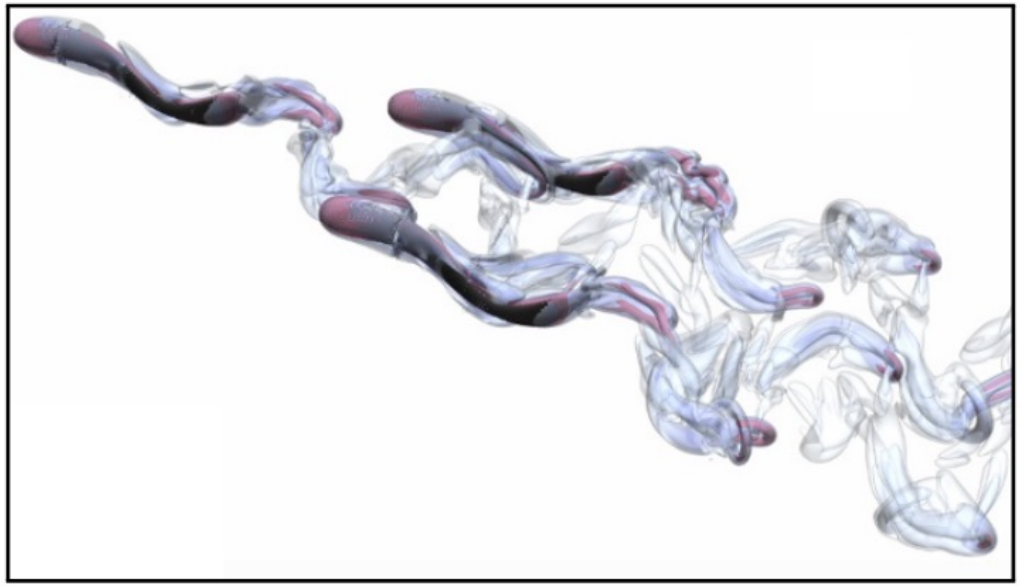
- 机器人抓取控制



环境
智能体
状态空间
动作空间
奖励

例子

- 机器鱼游动控制



环境
智能体
状态空间(历史信息)
动作空间
奖励

值迭代(VALUE ITERATION)

- 价值函数(Value function)

  状态价值函数

  $V^{\pi}(s) = \mathbb{E}_{\tau\sim\pi}[R(\tau)|s]$

  状态-动作价值函数

  $\mathrm{Q}^{\pi}(s,a) = \mathbb{E}_{\tau\sim\pi}[R(\tau)|s,a]$

- 最佳策略π^*

  使得$V^*(s) = \mathbb{E}_{\tau\sim\pi}[R(\tau)|s]$ 达到最大

  使得$Q^*(s,a) = \max_{\pi} \mathbb{E}_{\tau\sim\pi}[R(\tau)|s_0 = s, a_0 = a]$
  达到最大

值迭代 (VALUE ITERATION)

- 最佳策略

Bellman 方程(最佳策略)

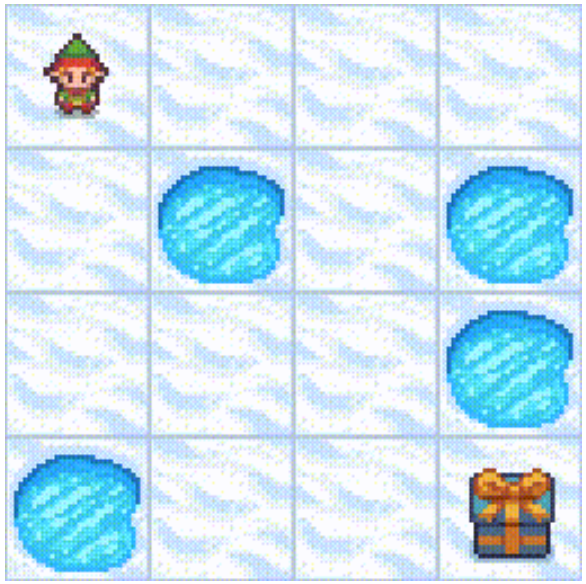
$$Q^*(s,a) = \max_{\pi} \mathbb{E}_{\tau\sim\pi}[\sum_{t=0} \gamma^t r_t | s_0 = s, a_0 = a]$$
$$= \mathbb{E}_{s'\sim p(\cdot|s,a)}[r_0 + \gamma \max_{a'} Q^*(s',a')]$$

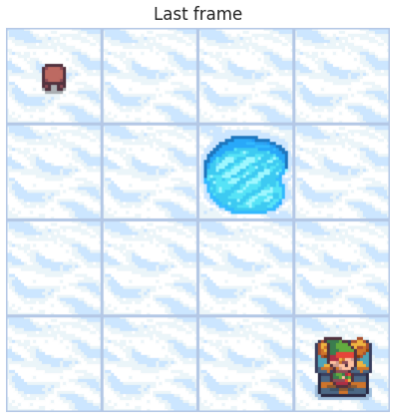
- 值迭代

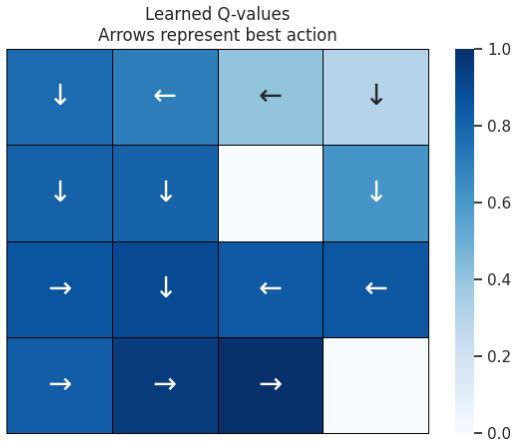
$$Q_{n+1}(s,a) = \mathbb{E}_{s'\sim p(\cdot|s,a)}[r_0 + \gamma \max_{a'} Q_n(s',a')]$$

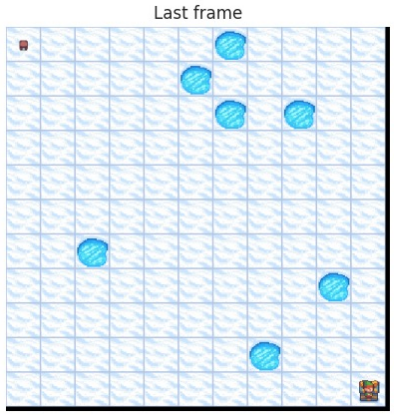
值迭代 (VALUE ITERATION)

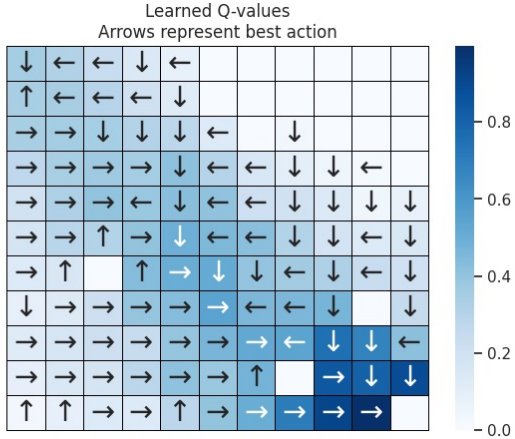
➢ 结冰的湖 （FrozenLake）











https://gymnasium.farama.org/tutorials/training_agents/FrozenLake_tuto/

值迭代 (VALUE ITERATION)

- 最佳策略

Bellman 方程(最佳策略)

$$Q^*(s,a) = \max_{\pi} \mathbb{E}_{\tau\sim\pi}[\sum_{t=0} \gamma^t r_t | s_0 = s, a_0 = a]$$
$$= \mathbb{E}_{s'\sim p(\cdot|s,a)}[r_0 + \gamma \max_{a'} Q^*(s',a')]$$

- 值迭代

$$Q_{n+1}(s,a) = \mathbb{E}_{s'\sim p(\cdot|s,a)}[r_0 + \gamma \max_{a'} Q_n(s',a')]$$

不可扩展。必须为每一对状态-动作计算 $Q^*(s,a)$ 。
例如状态是当前游戏的状态像素，那么在整个状态空间上计算将是计算上不可行的！
解决办法：使用神经网络来估计 $Q^*(s,a)$ 。

深度 (DEEP) 值迭代

- Bellman方程(最佳策略)

$$Q^*(s,a) = \mathbb{E}_{s'\sim p(\cdot|s,a)}[r_0 + \gamma \max_{a'} Q^*(s',a')]$$

- 损失函数

$$L_k(\theta_k) = \mathbb{E}_{s,a\sim p(\cdot)}\left[\left(y_k - Q(s,a;\theta_k)\right)^2\right]$$
$$y_k = \mathbb{E}_{s'\sim p(\cdot|s,a)}[r_0 + \gamma \max_{a'} Q(s',a';\theta_{k-1})]$$

$$\nabla_\theta L(\theta) = \mathbb{E}_{s,a\sim p(\cdot),s'\sim p(\cdot|s,a)\pi}[(r_0 + \gamma \max_{a'} Q(s',a';\theta) - Q(s,a;\theta))\nabla_\theta Q(s,a;\theta)]$$

深度 (DEEP) 值迭代

4个动作的经典街机游戏(Atari Games)

https://deepmind.google/discover/blog/agent57-outperforming-the-human-atari-benchmark/

$Q(s,a)$ 太复杂s,a都是很高维度
可能每个状态s仅会采用很少的动作a

策略梯度算法(POLICY GRADIENTS)

- 极大化价值函数

参数化策略函数 $\pi_\theta(s,a)$，概率函数$\pi_\theta(a|s)$

$$L(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] = \int R(\tau)p(\tau\sim\pi_\theta)d\tau$$

$$\nabla_\theta L(\theta) = \int R(\tau)\nabla_\theta p(\tau\sim\pi_\theta)d\tau$$

$$= \int R(\tau)\nabla_\theta \log p(\tau\sim\pi_\theta)\, p(\tau\sim\pi_\theta)d\tau$$

$$= \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)\nabla_\theta \log p(\tau\sim\pi_\theta)]$$

策略梯度算法(POLICY GRADIENTS)

- 极大化价值函数

计算 $\nabla_\theta \log p(\tau \sim \pi_\theta)$

$$p(\tau \sim \pi_\theta) = \prod_{t \geq 0} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$$

$$\nabla_\theta \log p(\tau \sim \pi_\theta) = \sum_{t \geq 0} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

$$\nabla_\theta L(\theta) \approx R(\tau) \sum_{t \geq 0} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

策略梯度算法(POLICY GRADIENTS)

- 极大化价值函数

$$\nabla_\theta L(\theta) \approx R(\tau) \sum_{t \geq 0} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

如果轨迹 $R(\tau)$ 的奖励很高，那么提升在该轨迹中所采取行动的概率。

如果轨迹 $R(\tau)$的奖励很低，那么降低在该轨迹中所采取行动的概率。

这种简单的更新规则在实际应用中会导致方差过高，因为很难准确地判断哪些行动真正导致了高奖励，哪些没有。归因问题（credit assignment problem）指的是在长序列的交互中，确定哪个行动对最终结果影响最大的挑战。为了改善这一点，研究者们开发了多种技术来减少估计的方差，例如使用基线（baseline）或者引入更复杂的策略梯度算法如Actor-Critic等。

在线规划

- 离线规划(offline planning): 值迭代、策略梯度方法

  离线解决了所有可能状态下的问题，然后在线使用解决方案（即策略）来进行行动。

- 在线规划(online planning)

  规划和执行是交织进行的。一旦执行了一个动作（或者可能是一系列动作），我们就从新状态开始再次规划。每个动作的质量通过重复模拟获得的轨迹的预期奖励平均值来近似，从而给出 的近似值。

蒙特卡洛树搜索

- 蒙特卡洛方法

给定当前状态 s ，估计$Q^*(s,a), Q^*(s,b), Q^*(s,c), \cdots\cdots$
采取$\max_{a'} Q^*(s,a')$

- 蒙特卡洛估计$Q^*(s,a)$

$$Q^{\pi}(s,a) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0} \gamma^t r_t | s_0 = s, a_0 = a]$$

$$\approx \frac{1}{N}\sum_{i=1} R_i(\tau \sim \pi | s_0 = s, a_0 = a)$$

较为浪费计算资源

蒙特卡洛树搜索

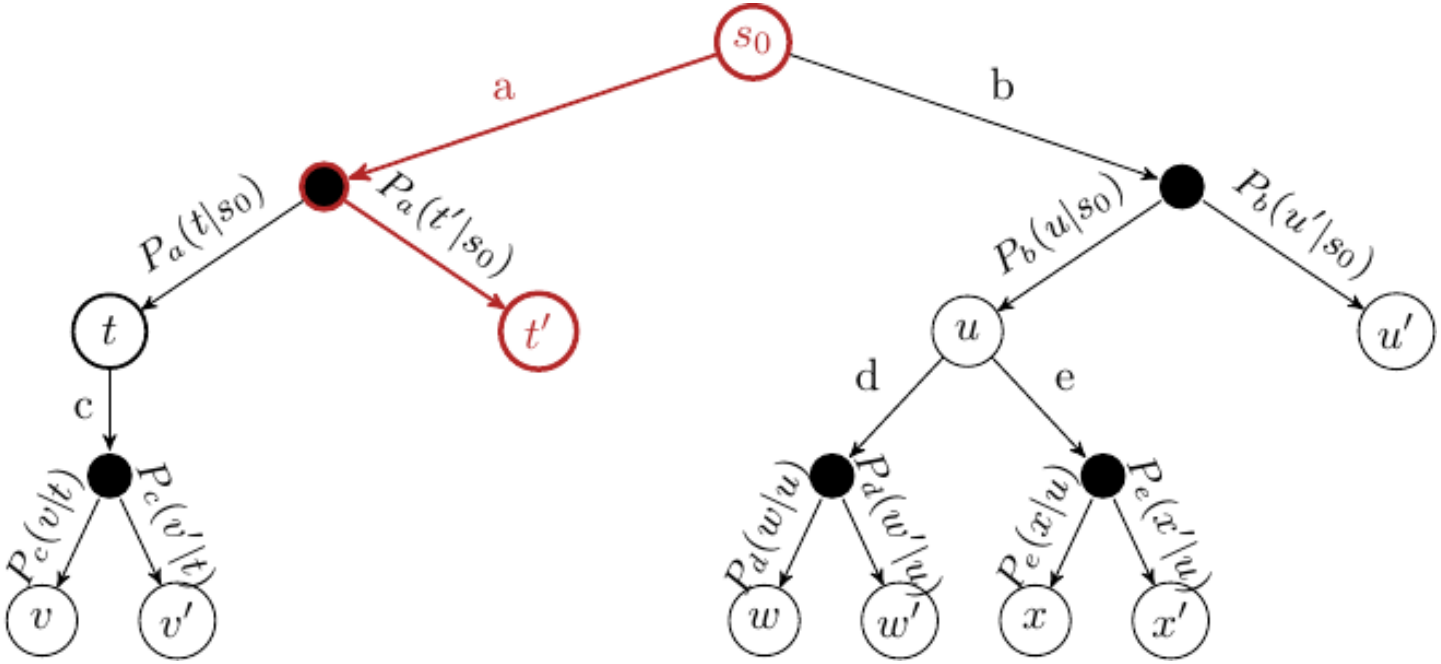
- 蒙特卡洛方法

我们逐步构建搜索树(状态能较好的表示成序列)。树中的每个节点存储着$Q^*(s,a)$的近似

- 选择(selection)
- 扩展(expansion)
- 模拟(simulation)
- 方向传播(backpropagation)

蒙特卡洛树搜索

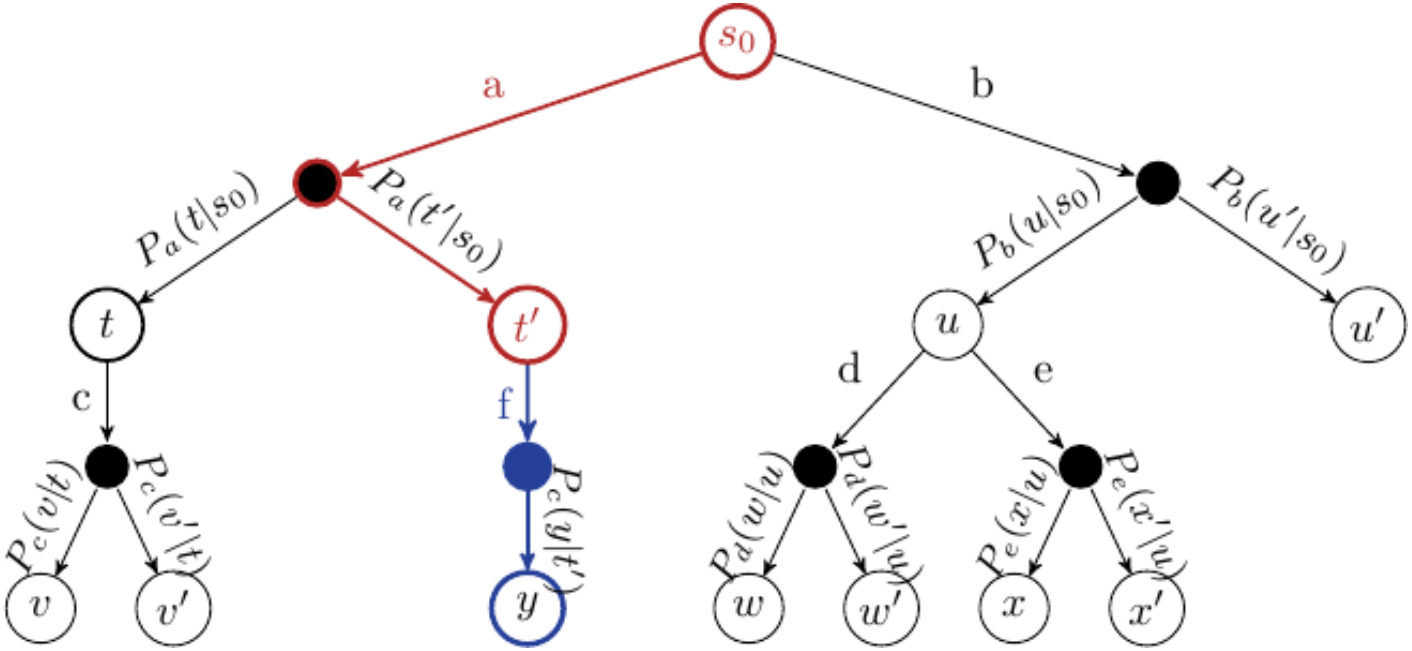
➢ 选择 (selection)



从根节点(当前状态)开始，连续选择子节点，直到达到一个未完全展开的节点(比如t')，未完全展开的节点: 没有子节点在搜索树里

蒙特卡洛树搜索

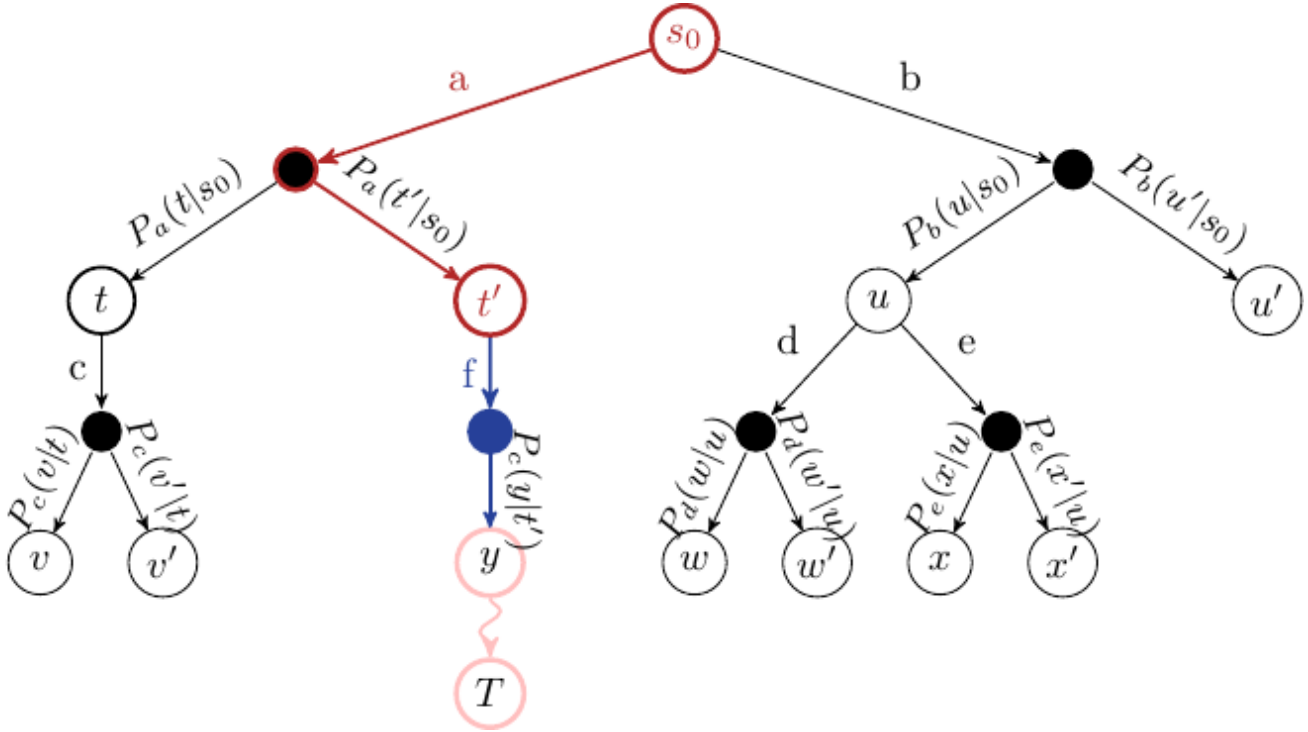
➢ 扩展(expansion)



从未展开的节点t'选择一个动作 f，可以随机选择或使用启发式方法，扩展得到新的节点y。

蒙特卡洛树搜索

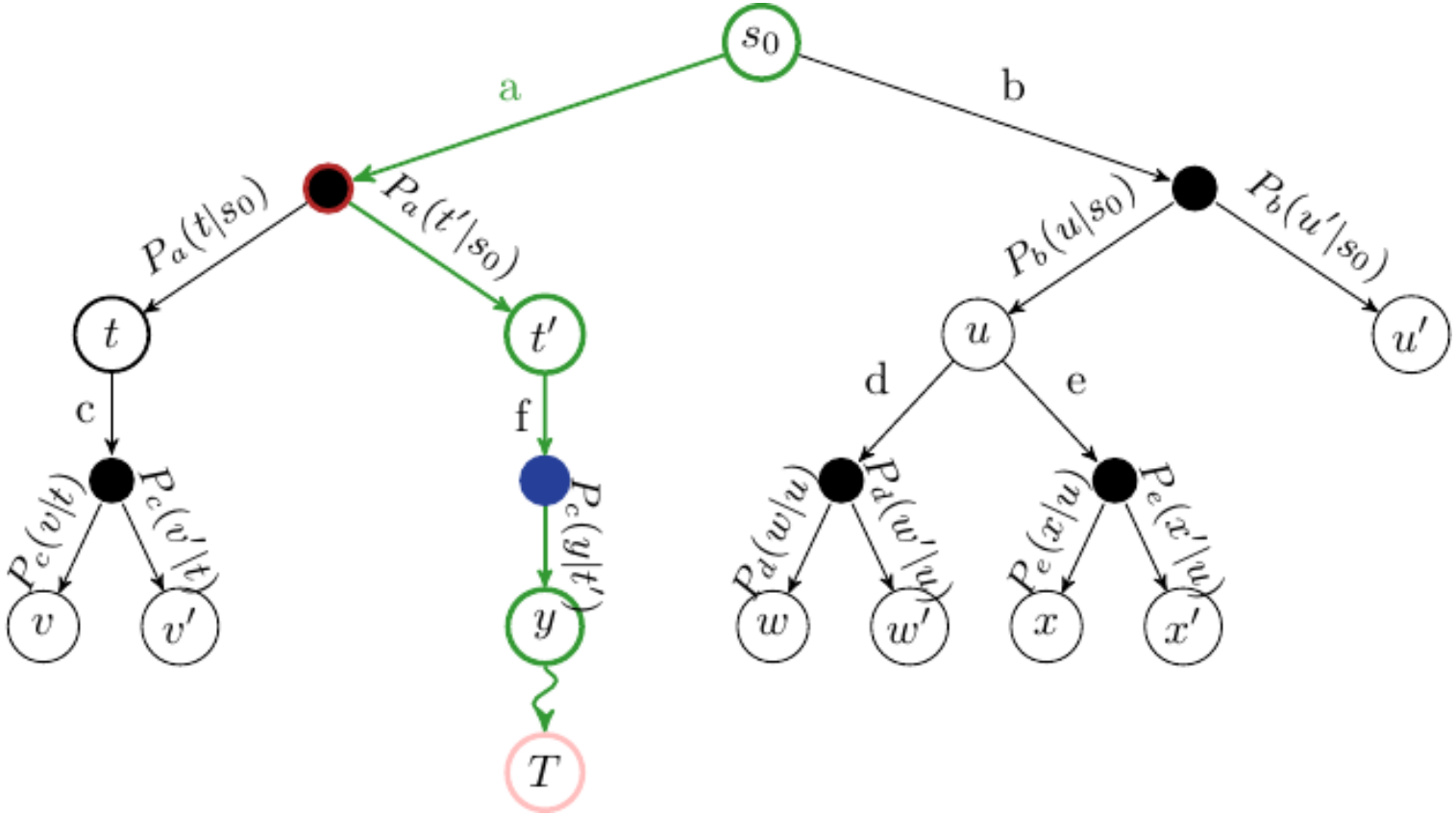
➢ 模拟(simulation)



从新的节点y开始执行随机模拟，直至达到终止状态。可以使用启发式方法来改进随机模拟，通过引导模拟朝向更有前景的状态。计算奖励$R(\tau \sim \pi | s_0 = s, a_0 = a)$。

蒙特卡洛树搜索

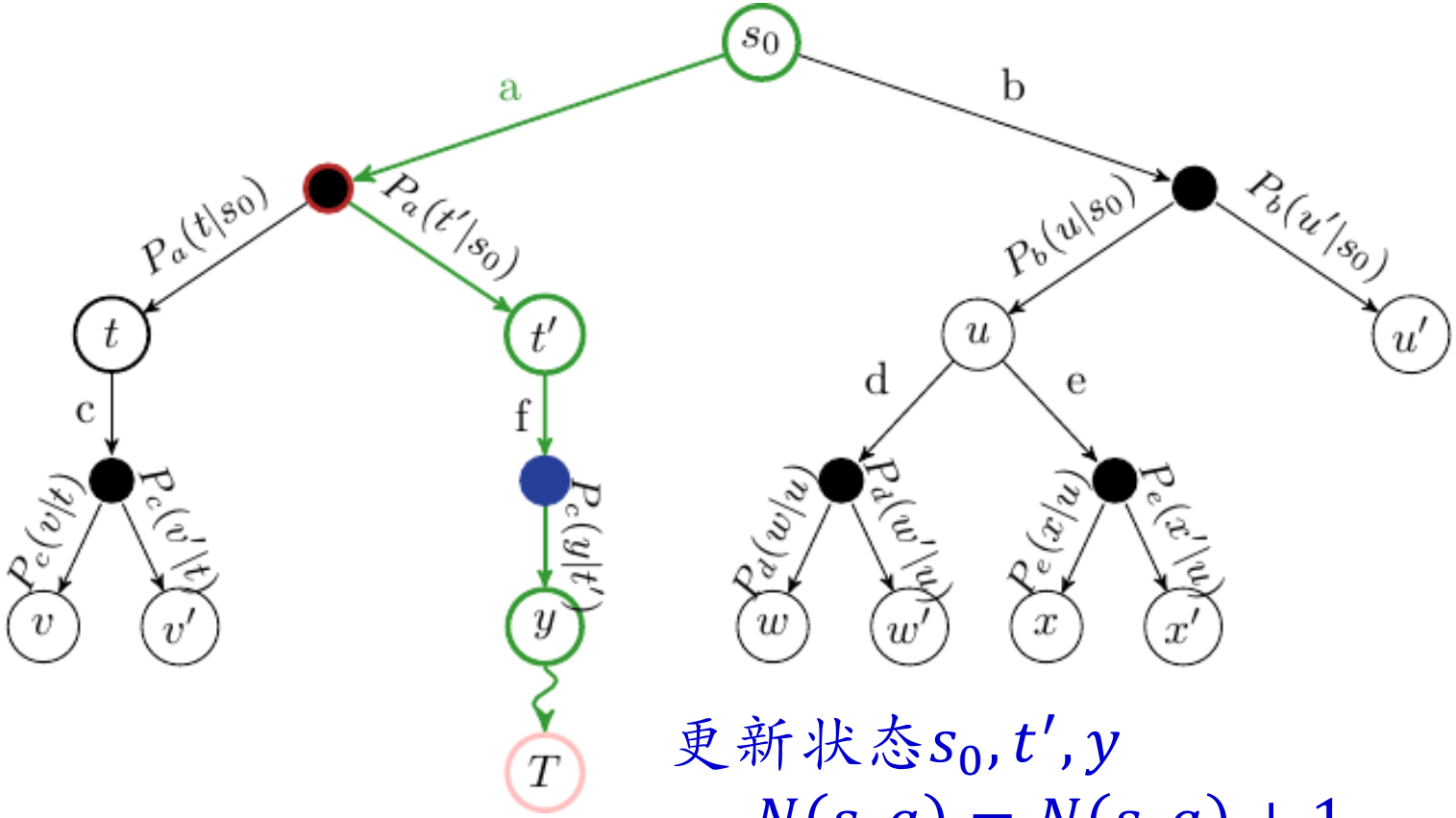
➢ 反向传播(backpropagation)



根据终止状态处的奖励 R ，反向传播该奖励以计算路径上每个状态的价值 $Q(s,a)$ 。

蒙特卡洛树搜索

➢ 反向传播(backpropagation)



更新状态s_0, t', y

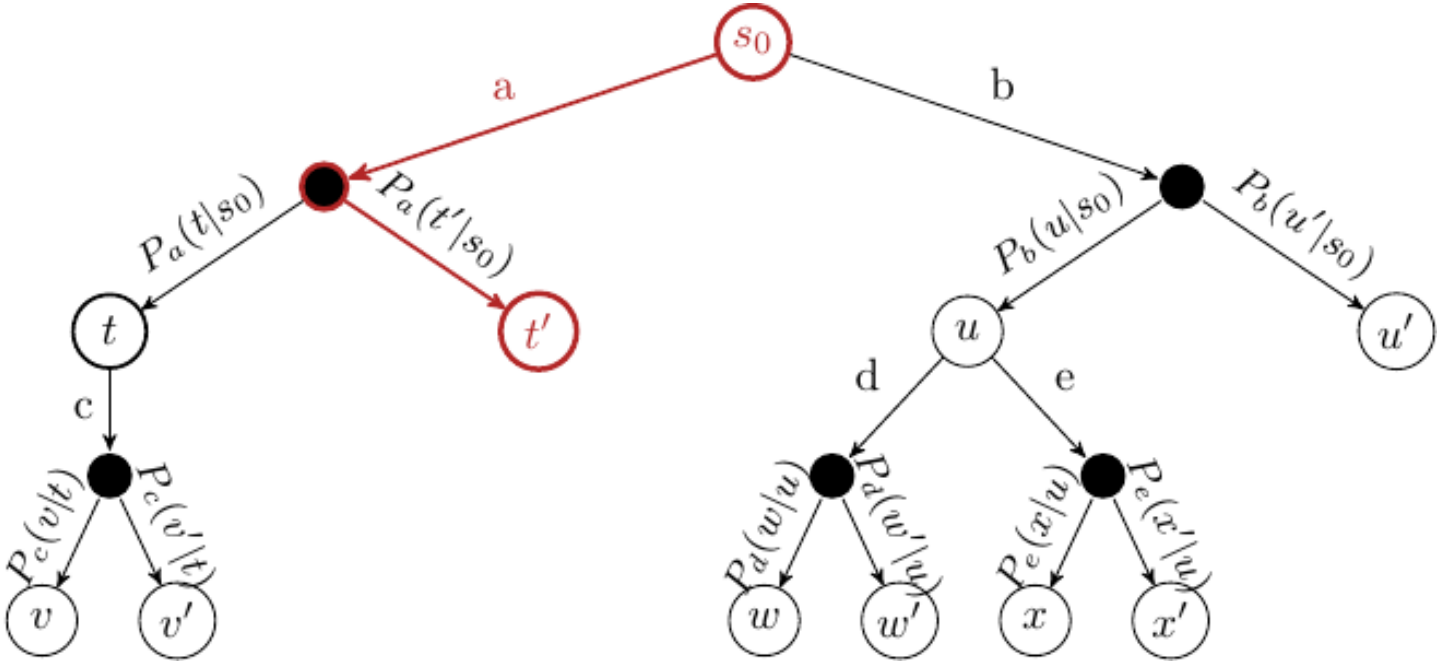
$$N(s,a) = N(s,a) + 1$$

$$R = r + \gamma R$$

$$Q(s,a) = Q(s,a) + \frac{R - Q(s,a)}{N(s,a)}$$

蒙特卡洛树搜索

➢ 选择(selection)



多臂老虎机算法(multi-armed bandit algorithm)

$$a = \underset{a}{\operatorname{argmax}}\, Q(s,a) + 2C_p\sqrt{\frac{2\ln N(s)}{N(s,a)}}$$

蒙特卡洛树搜索：例子

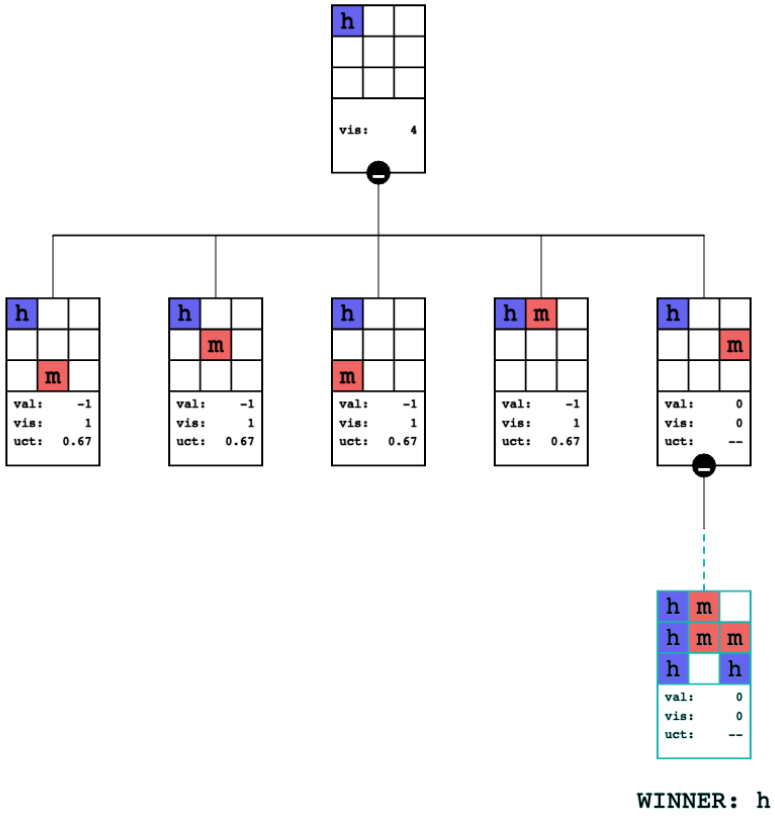
➢ 围棋

多次迭代
- 选择(selection)
- 扩展(expansion)
- 模拟(simulation)
- 方向传播(backpropagation)

确定下一步棋

蒙特卡洛树搜索：例子

➢ 井字棋



胜 val +1

平 val + 0

负 val -1

$$\mathrm{UCT} = \frac{\mathrm{val}}{\mathrm{vis}} + \sqrt{\frac{2\ln(\mathrm{parent.vis})}{\mathrm{vis}}}$$

https://vgarciasc.github.io/mcts-viz/

参考文献

参考文献

斯坦福CS231课件：
https://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture14.pdf

伯克利CS294课件
https://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_5_actor_critic_pdf

博客：
https://gibberblot.github.io/rl-notes/single-agent/mcts.html