第十讲：基于模型的强化学习

主讲人 陈达贵

清华大学自动化系

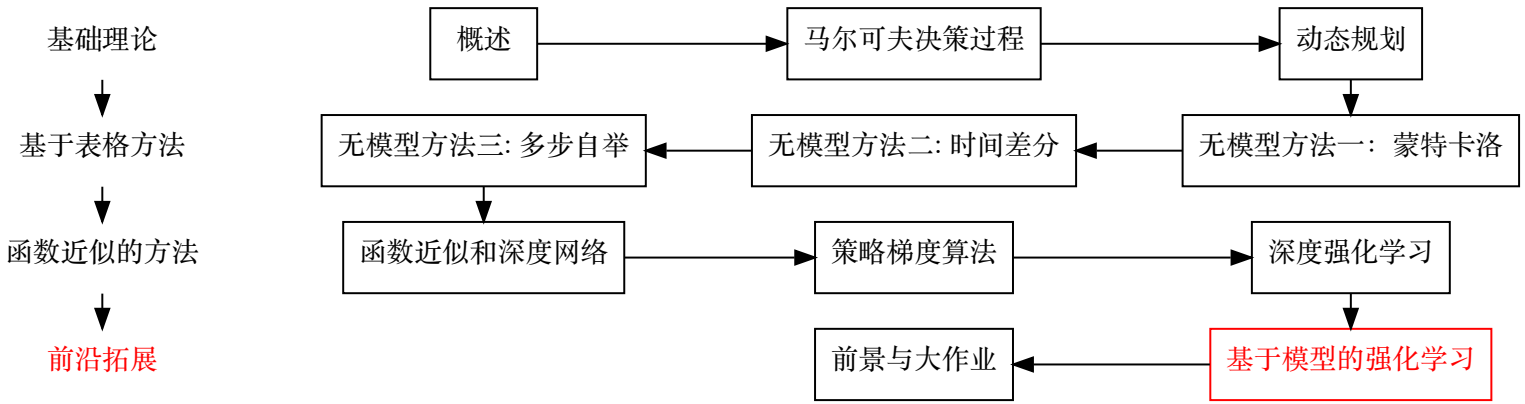
在读硕士

深蓝学院
shenlanxueyuan.com

强化学习理论与实践

目录

章节目录

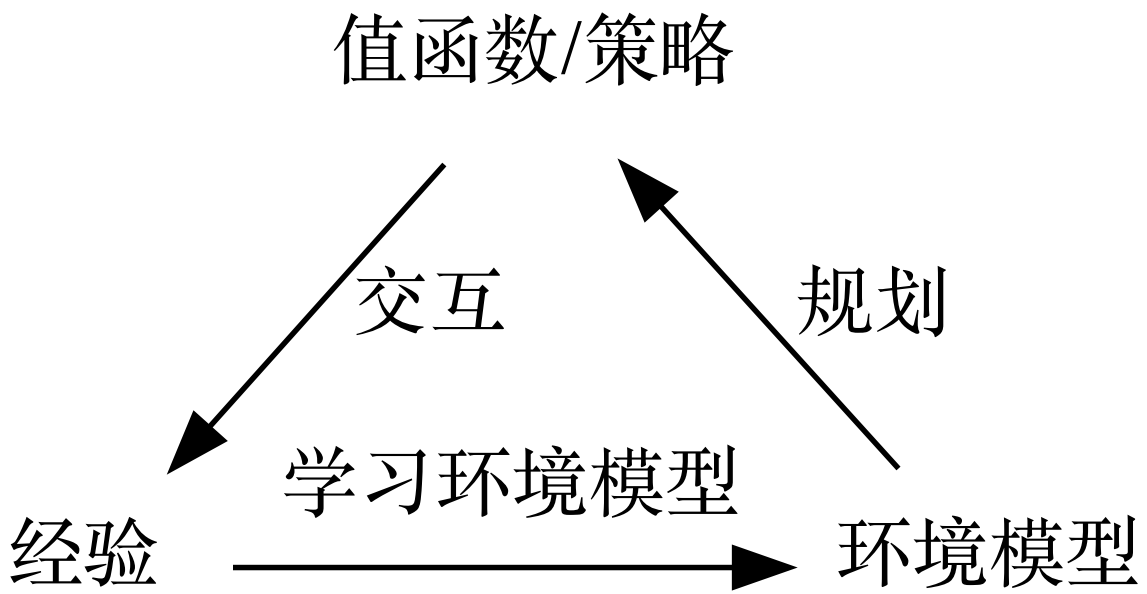

基础理论
基于表格方法
函数近似的方法
前沿拓展
概述
马尔可夫决策过程
动态规划
无模型方法一：蒙特卡洛
无模型方法二: 时间差分
无模型方法三: 多步自举
函数近似和深度网络
策略梯度算法
深度强化学习
基于模型的强化学习
前景与大作业

本章目录

学习与规划

- 学习 (Learning)
  - 未知环境模型 $\mathcal{P}, \mathcal{R}$
  - 智能体与环境交互产生经验
  - 从经验中学习 (学习值函数，策略等)
  - 无模型的方法 (MC, TD 等)
- 规划 (Planning)
  - 已知环境模型 $\mathcal{P}, \mathcal{R}$
  - 无需智能体的交互
  - 直接从模型学习
  - 动态规划的方法 (DP)
- 相同点
  - 都是根据未来的结果，计算当前的估计

基于模型的强化学习



无模型 RL 与基于模型的 RL

- 无模型 RL
  - 没有环境模型
  - 从经验中学习值函数 (或者策略)
- 基于模型的 RL
  - 从经验中学习一个环境模型
  - 利用环境模型做动态规划，从而计算出值函数或者策略

基于模型的强化学习相当于学习一个虚拟的环境。

目录

优势和劣势

- 优势
  - 能够通过监督学习有效地学习环境模型
  - 能够利用环境模型有效地学习
  - 减少不精确的值函数带来的影响
  - 直接利用环境模型的不确定性
- 劣势
  - 先学习环境模型，再构建值函数
    - 存在两次近似误差 $\Rightarrow$ 误差累计

什么是环境模型

- 环境模型即智能体能够预测环境的返回值
- 环境模型 $\mathcal{M}$ 是一个 MDP$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ 的参数化表达
- 我们假设状态空间 $\mathcal{S}$ 和动作空间 $\mathcal{A}$ 均已知
- 所以一个环境模型 $\mathcal{M} = \langle \mathcal{P}_\eta, \mathcal{R}_\eta \rangle$ 表示状态转移函数 $\mathcal{P}_\eta \approx \mathcal{P}$ 和奖励函数 $\mathcal{R}_\eta \approx \mathcal{R}$

$$S_{t+1} = \mathcal{P}_\eta \left(S_{t+1} | S_t, A_t \right)$$
$$R_{t+1} = \mathcal{R}_\eta \left(R_{t+1} | S_t, A_t \right)$$

- 特别地，我们假设了状态转移函数和奖励函数的条件独立

$$\mathbb{P} \left[S_{t+1}, R_{t+1} | S_t, A_t \right] = \mathbb{P} \left[S_{t+1} | S_t, A_t \right] \mathbb{P} \left[R_{t+1} | S_t, A_t \right]$$

模型学习

- 从经验 $\{S_1, A_1, R_2, \cdots, S_T\}$ 估计环境模型 $\mathcal{M}_\eta$
- 这是一个监督学习问题

$$
\begin{aligned}
S_1, A_1 &\to R_2, S_2 \\
S_2, A_2 &\to R_3, S_3 \\
&\vdots \\
S_{T-1}, A_{T-1} &\to R_T, S_T
\end{aligned}
$$

- 学习 $s, a \to r$ 是一个回归问题
- 学习 $s, a \to s'$ 是一个概率分布估计问题
- 选用的损失函数, MSE, KL 距离
- 通过寻找参数 η 来最小化损失函数

模型的例子

- 查表模型
- 线性期望模型
- 线性高斯模型
- 高斯过程模型
- 神经网络
- 深度神经网络
- ...

查表模型

- 统计每个状态动作对出现的次数 $N(s,a)$

$$\hat{\mathcal{P}}^a_{ss'} = \frac{1}{N(s,a)} \sum_{t=1}^{T} \mathbf{1}\left(S_t, A_t, S_{t+1} = s, a, s'\right)$$

$$\hat{\mathcal{R}}^a_s = \frac{1}{N(s,a)} \sum_{t=1}^{T} \mathbf{1}(S_t, A_t = s, a) R_t$$

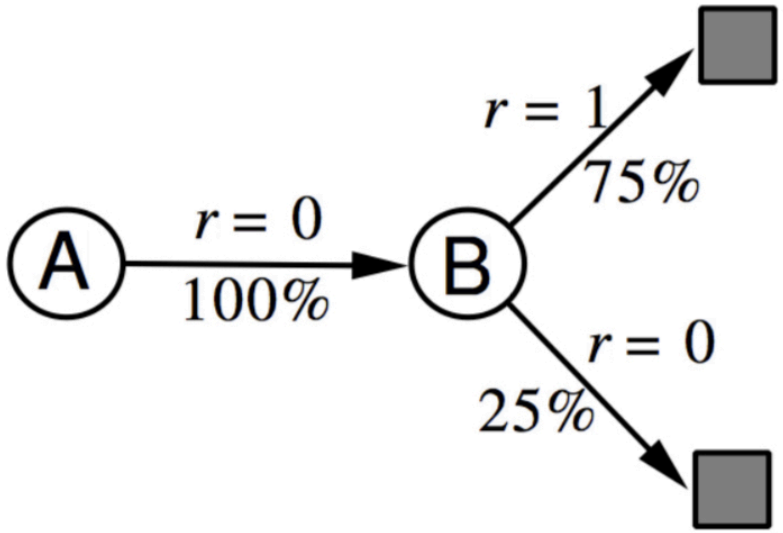
另外，我们也可以

- 存储每个时刻的经验 $\langle S_t, A_t, R_{t+1}, S_{t+1} \rangle$
- 采样模型：采样匹配的经验 $\langle s, a, \cdot, \cdot \rangle$

例子

采样了 8 个片段

- A, 0, B, 0
- B, 1
- B, 1
- B, 1
- B, 1
- B, 1
- B, 1
- B, 0



我们已经通过经验构建了一个表格查找模型

根据模型规划

- 给定模型 $\mathcal{M}_\eta = \langle \mathcal{P}_\eta, \mathcal{R}_\eta \rangle$
- 解 MDP，$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}_\eta, \mathcal{R}_\eta \rangle$
- 使用动态规划算法
  - 值迭代
  - 策略迭代
  - 树搜索
  - ⋮

基于样本的规划

- 一个简单但是有效的规划方法
- 模型仅仅用来生成样本
- 从模型中采样经验

$$S_{t+1} \sim \mathcal{P}_\eta \left(S_{t+1} | S_t, A_t \right)$$
$$R_{t+1} = \mathcal{R}_\eta \left(R_{t+1} | S_t, A_t \right)$$

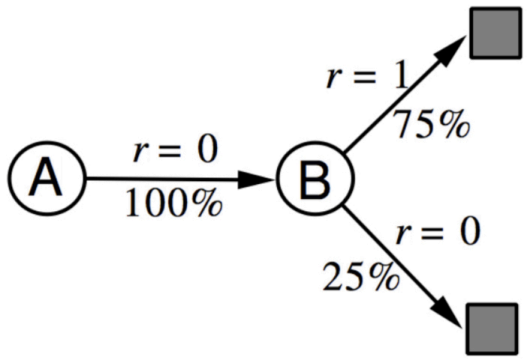
- 学习的时候仍然使用无模型的 RL，比如
  - 蒙特卡洛优化
  - Sarsa
  - Q 学习
- 这个方法效率通常很高

例子

- 首先根据真实经验构建表格查找模型
- 然后采样经验
- 利用无模型的 RL 来学习值函数

真实经验

- A, 0, B, 0
- B, 1
- B, 1
- B, 1
- B, 1
- B, 1
- B, 1
- B, 0



采样的经验

- B, 1
- B, 0
- B, 1
- A, 0, B, 1
- B, 1
- A, 0, B, 1
- B, 1
- B, 0

如果使用 MC 学习，可以估计得到 $V(A) = 1$, $V(B) = 0.75$.

使用不精确的模型规划

- 给定一个不精确的模型 $\langle \mathcal{P}_\eta, \mathcal{R}_\eta \rangle \neq \langle \mathcal{P}, \mathcal{R} \rangle$
- 所解出来的最优解仅仅适合于近似的 MDP$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}_\eta, \mathcal{R}_\eta \rangle$
- 当环境模型不精确时，动态规划只能计算到一个次优的策略
- 解决方法 1：当环境模型错误时，使用无模型的 RL
- 解决方法 2：在求解时考虑到模型的不精确性

目录

真实的和仿真的经验

我们考虑两种形式的经验

- 真实的经验，从环境中真实采样 (真实的 MDP)

$$S' \sim \mathcal{P}_{ss'}^{a}$$
$$R = \mathcal{R}_{s}^{a}$$

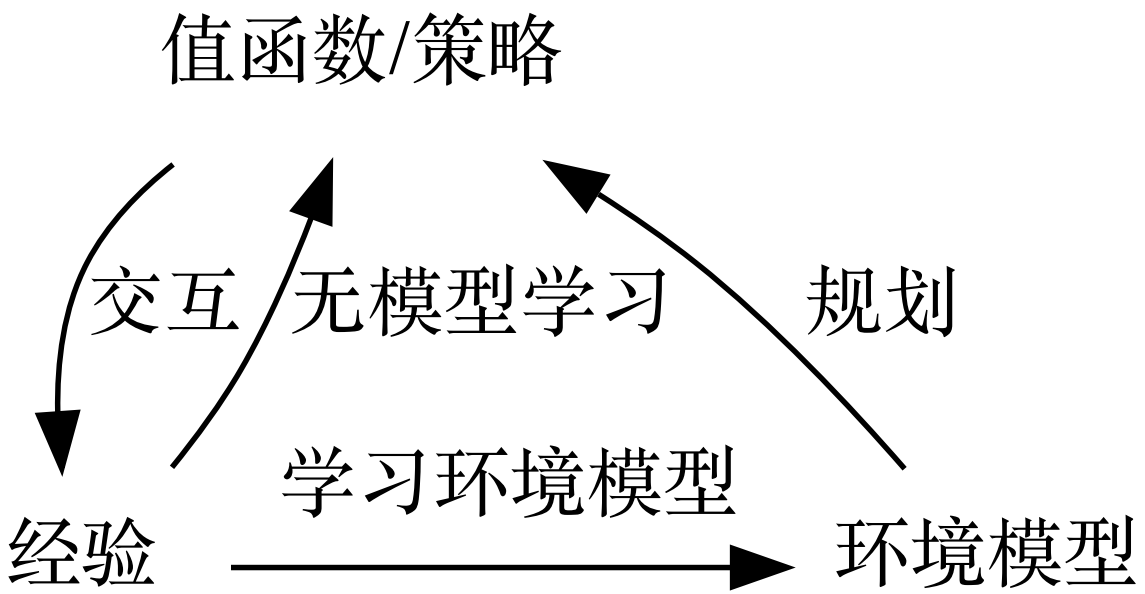
- 仿真的经验，从环境模型中采样 (近似的 MDP)

$$S' \sim \mathcal{P}_{\eta}(S'|S, A)$$
$$R = \mathcal{R}_{\eta}(R|S, A)$$

整合学习和规划

- 无模型 RL
  - 没有环境模型
  - 从经验中学习值函数 (或者策略)
- 基于模型的 RL
  - 从经验中学习一个环境模型
  - 利用环境模型做动态规划，从而计算出值函数或者策略
- Dyna
  - 从经验中学习一个环境模型
  - 同时利用真实的和仿真的经验来学习值函数和策略

Dyna 的结构图



Dyna-q 的算法

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

Do forever:

(a) $S \leftarrow$ current (nonterminal) state

(b) $A \leftarrow \varepsilon$-greedy(S,Q)

(c) Execute action A; observe resultant reward, R, and state, S'

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

(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)

(f) Repeat n times:

$S \leftarrow$ random previously observed state

$A \leftarrow$ random action previously taken in S

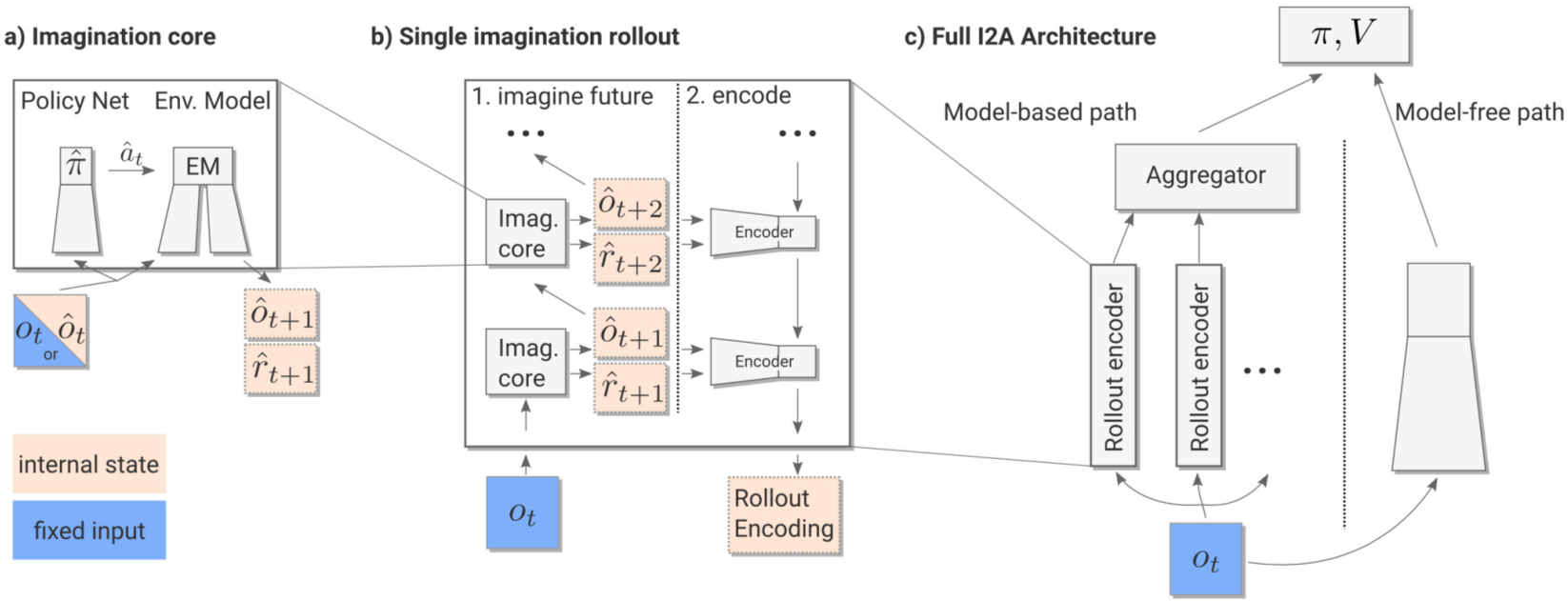
$R, S' \leftarrow Model(S,A)$

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

一个深度强化学习的例子

Imagination-Augmented Agents for Deep Reinforcement Learning

- 混合了无模型的方法和基于模型的方法
- 考虑了环境模型的不精确性



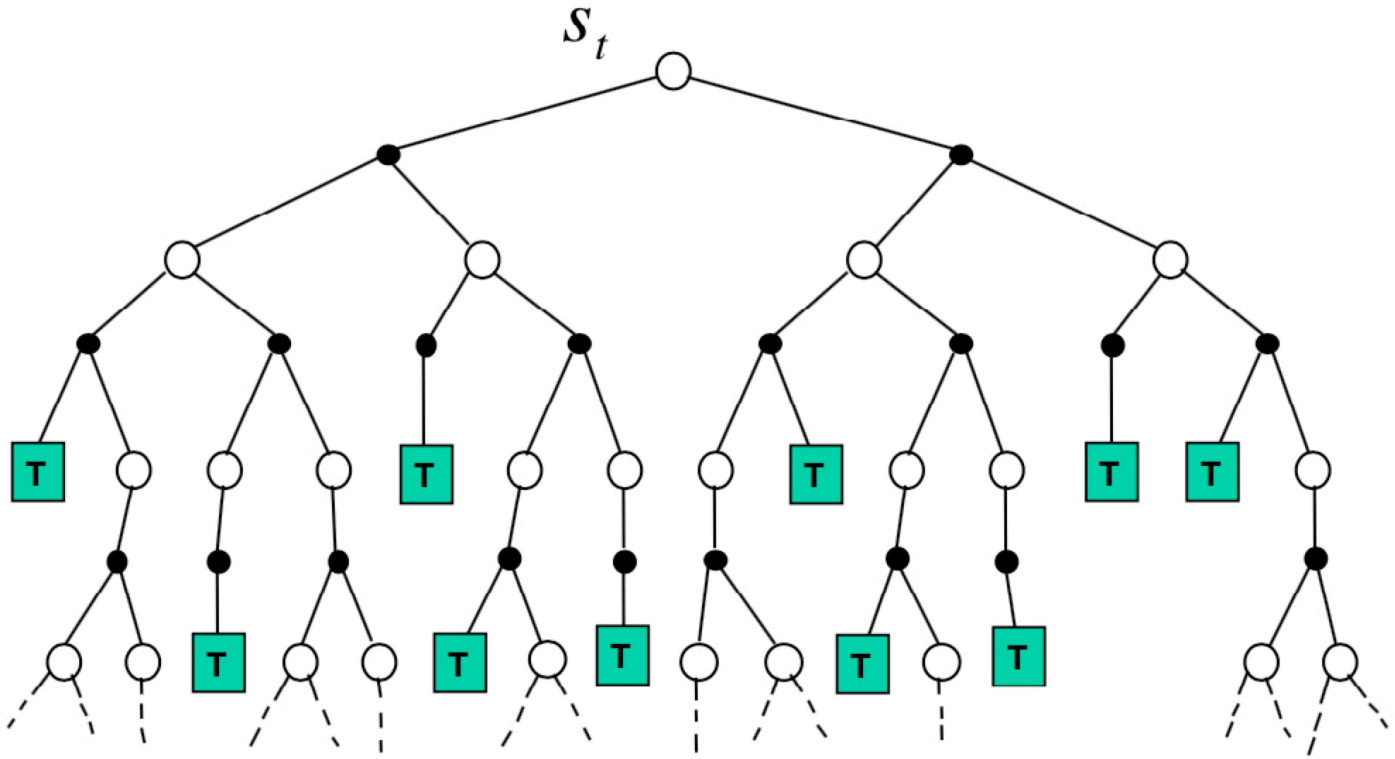
目录

仿真与环境模型

- 环境模型指我们知道了状态转移函数和奖励函数 $\mathcal{P}_\eta$ 和 $\mathcal{R}_\eta$.
- 仿真是一个更加弱化的条件
  - 我们不需要知道 $\mathcal{P}$ 和 $\mathcal{R}$ 的具体形式
  - 只要能以某种过程，获得样本即可
- 如果已知环境模型，那么一定可以仿真
- 如果不已知环境模型，也有可能进行仿真

前向搜索

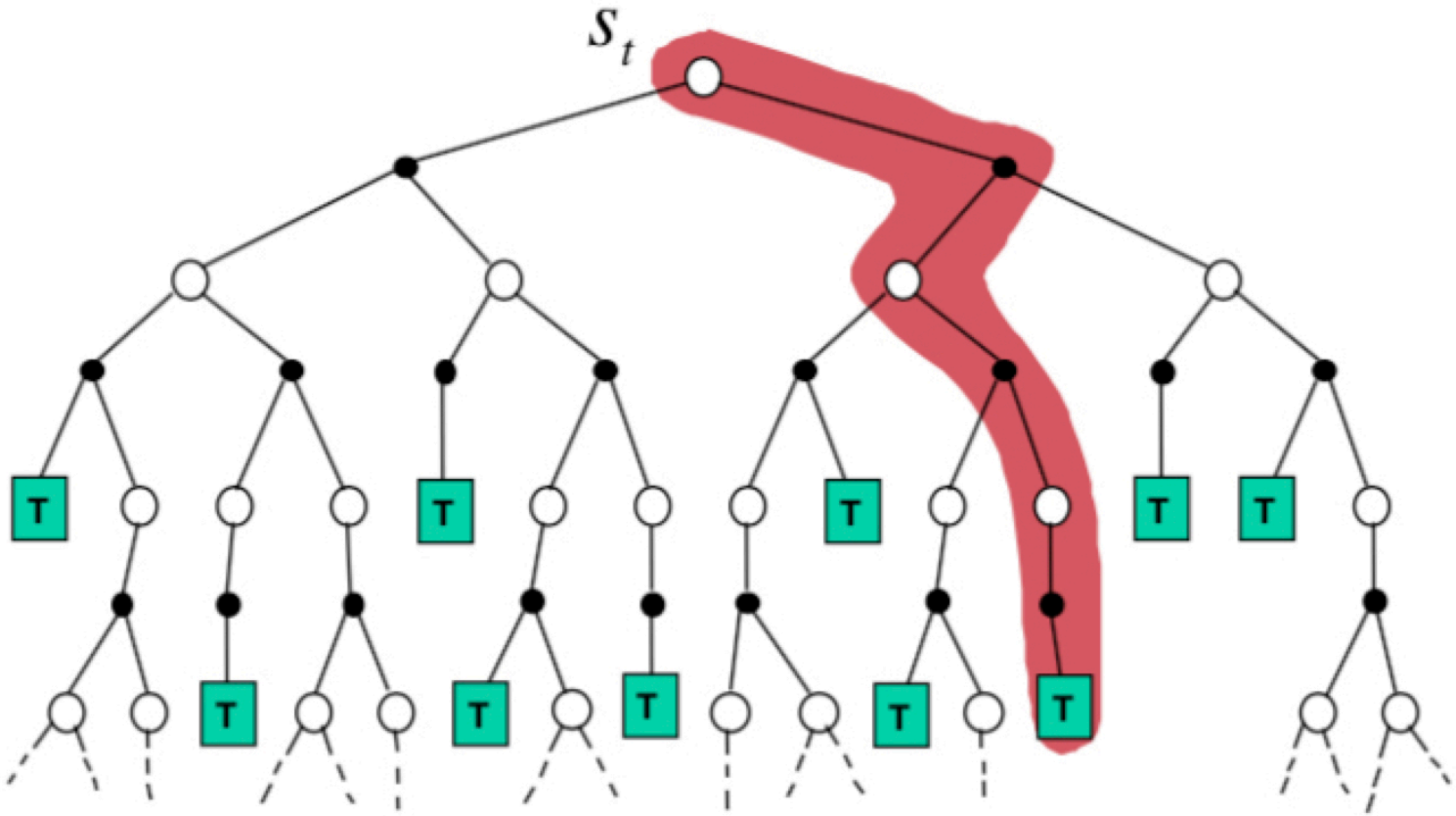
- 前向搜索主要是为了选择最好的动作
- 以当前状态 s_t 为根建立了一颗搜索树
- 可以使用环境模型来向前搜索



- 不需要解整个 MDP，仅仅解从当前状态开始的部分 MDP 即可

基于仿真的搜索

- 前向搜索算法使用了基于样本的规划算法
- 利用环境模型仿真了从当前状态开始的数个经验片段
- 然后对这些仿真的片段使用无模型的 RL 来学习



基于仿真的搜索

- 利用环境模型仿真了从当前状态开始的数个经验片段

$$\{s_t^k, A_t^k, R_{t+1}^k, \cdots, S_T^k\}_{k=1}^K \sim \mathcal{M}_v$$

- 然后对这些仿真的片段使用无模型的 RL 来学习
  - 蒙特卡洛优化 $\rightarrow$ 蒙特卡洛搜索
  - Sarsa$\rightarrow$TD 搜索

简单蒙特卡洛搜索

- 给定环境模型 $\mathcal{M}_v$ 和一个仿真策略 π
- 对于每个动作 $a \in \mathcal{A}$
  - 从当前状态 s_t 开始采样 K 个片段

$$\left\{s_t, a, R_{t+1}^k, S_{t+1}^k, A_{t+1}^k, \cdots, S_T^k\right\}_{k=1}^K \sim \mathcal{M}_v, \pi$$

  - 使用平均回报值评价 Q 函数 (蒙特卡洛评价)

$$Q(s_t, a) = \frac{1}{K} \sum_{k=1}^{K} G_t \xrightarrow{P} q_\pi(s_t, a)$$

- 根据 Q 函数的最大值来选择当前的动作

$$a_t = \arg\max_{a \in \mathcal{A}} Q(s_t, a)$$

蒙特卡洛树搜索 (评价)

- 给定一个模型 $\mathcal{M}_v$
- 从当前状态 s_t 开始使用当前仿真策略 π 采样 K 个片段

$$\left\{s_t, A_t^k, R_{t+1}^k, S_{t+1}^k, \cdots, S_T^k\right\}_{k=1}^K \sim \mathcal{M}_v, \pi$$

- 建立一颗包含了所有访问的状态和动作的搜索树
- 使用从 s, a 开始的片段的回报值来评价 $Q(s, a)$

$$Q(s, a) = \frac{1}{N(s, a)} \sum_{k=1}^{K} \sum_{u=t}^{T} \mathbf{1}\left(S_u, A_u = s, a\right) G_u \xrightarrow{P} q_\pi(s, a)$$

- 搜索结束之后，选择 Q 值最大的动作

$$a_t = \arg\max_{a \in \mathcal{A}} Q(s_t, a)$$

蒙特卡洛树搜索 (仿真)

- 在 MCTS 中 (Monte-Carlo Tree Search)，仿真策略 π 需要策略提升
- 每次仿真有两个阶段 (树搜索内，树搜索外)
  - 树策略(提升)：选择动作以最大化 $Q(S, A)$
  - 默认策略(固定)：快速计算到终止状态
- Repeat（每次仿真）
  - 使用 MC 评价来估计 $Q(S, A)$
  - 提升树策略，比如 ε 贪婪，UCB 等
- 对仿真出来的经验做 MC 优化
- 收敛到最优的搜索树 $Q(S, A) \rightarrow q_*(S, A)$

例子：围棋

- 被认为是最难的经典棋类博弈
- 解空间 $2^{361} \approx 10^{108}$
- Combinatorial Game: 零和，完美信息，确定性，离散，序列化
- MCTS 适合用来解 Combinatorial Game 的问题

围棋中的位置评价

- 评价位置 s 有多好？
- 奖励函数

$$R_t = 0 \quad \text{对于所有非终止状态} t < T$$

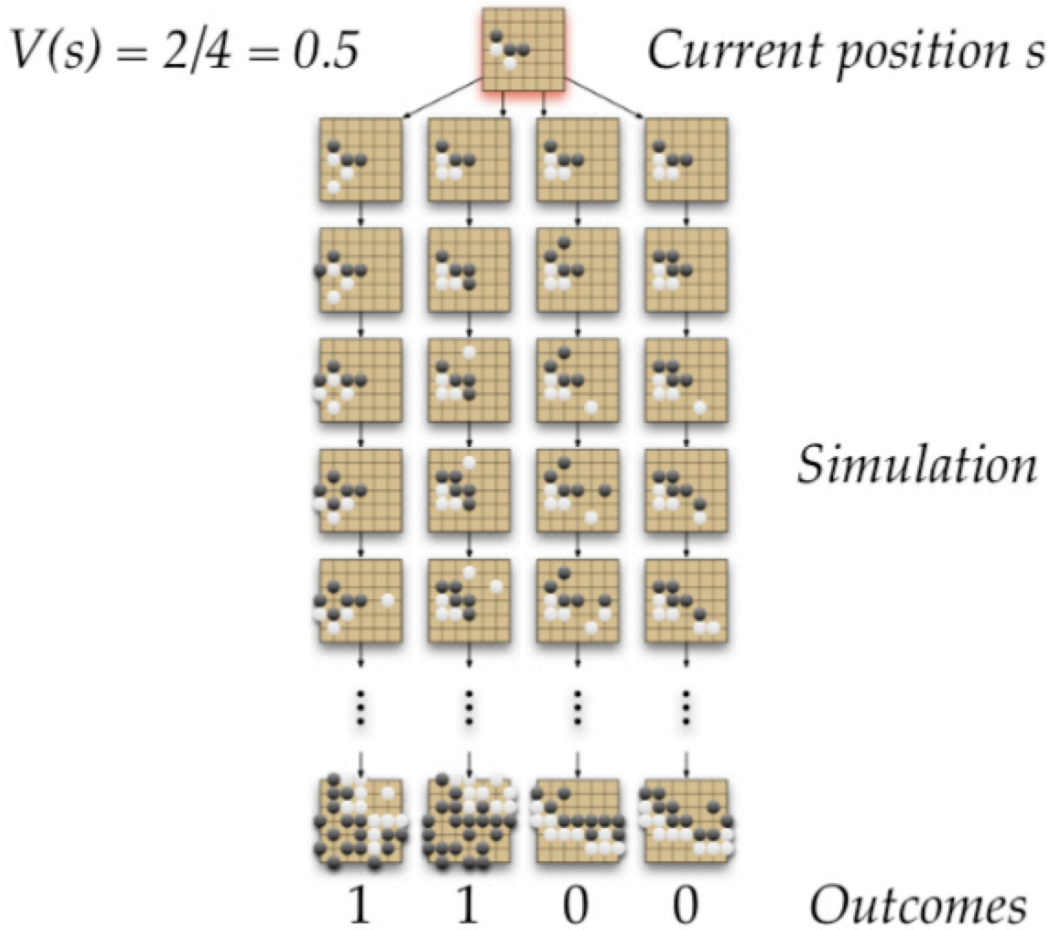
$$R_T = \begin{cases} 1 & \text{黑棋赢} \\ 0 & \text{白棋赢} \end{cases}$$

- 策略 $pi = \langle \pi_B, \pi_W \rangle$ 表示白方和黑方的策略
- 值函数（奖励无衰减）

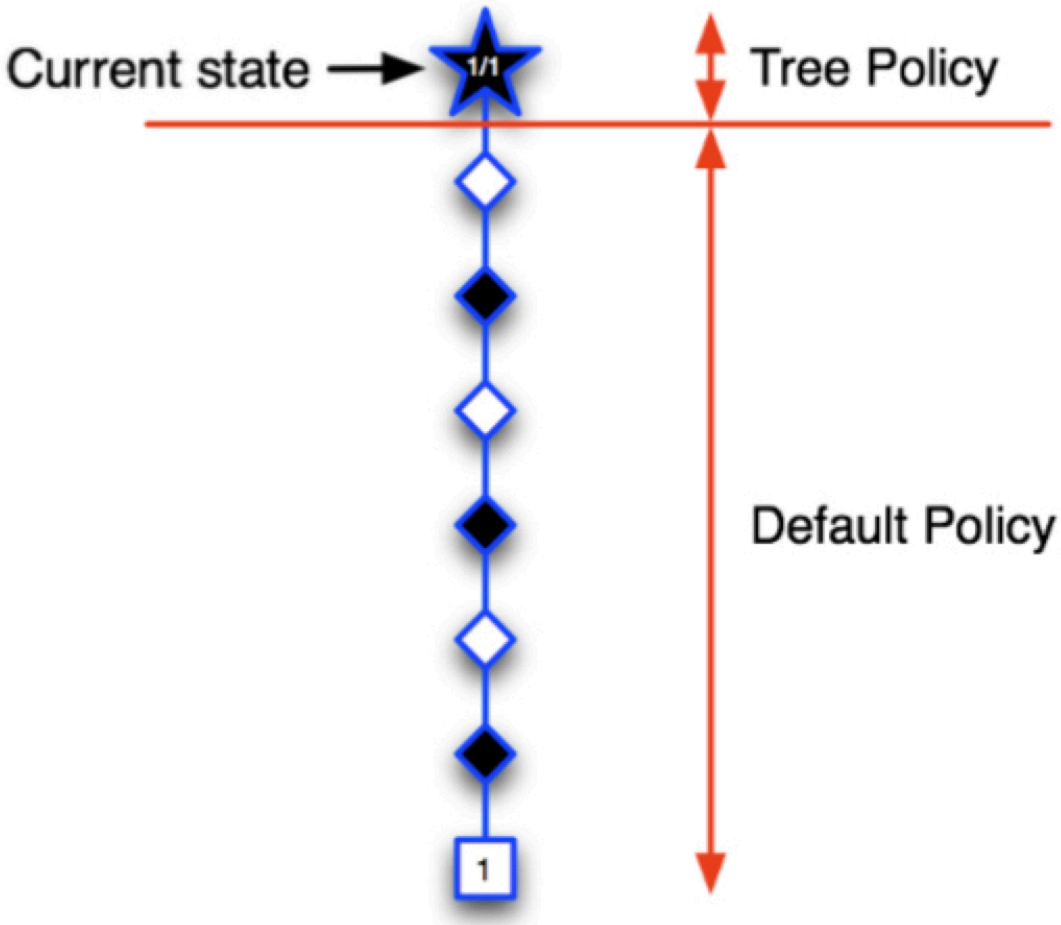
$$v_\pi(s) = \mathbb{E}_\pi \left[R_T | S = s \right] = \mathbb{P} \left[\text{黑棋赢} | S = s \right]$$

$$v_*(s) = \max_{\pi_B} \min_{\pi_W} v_\pi(s)$$

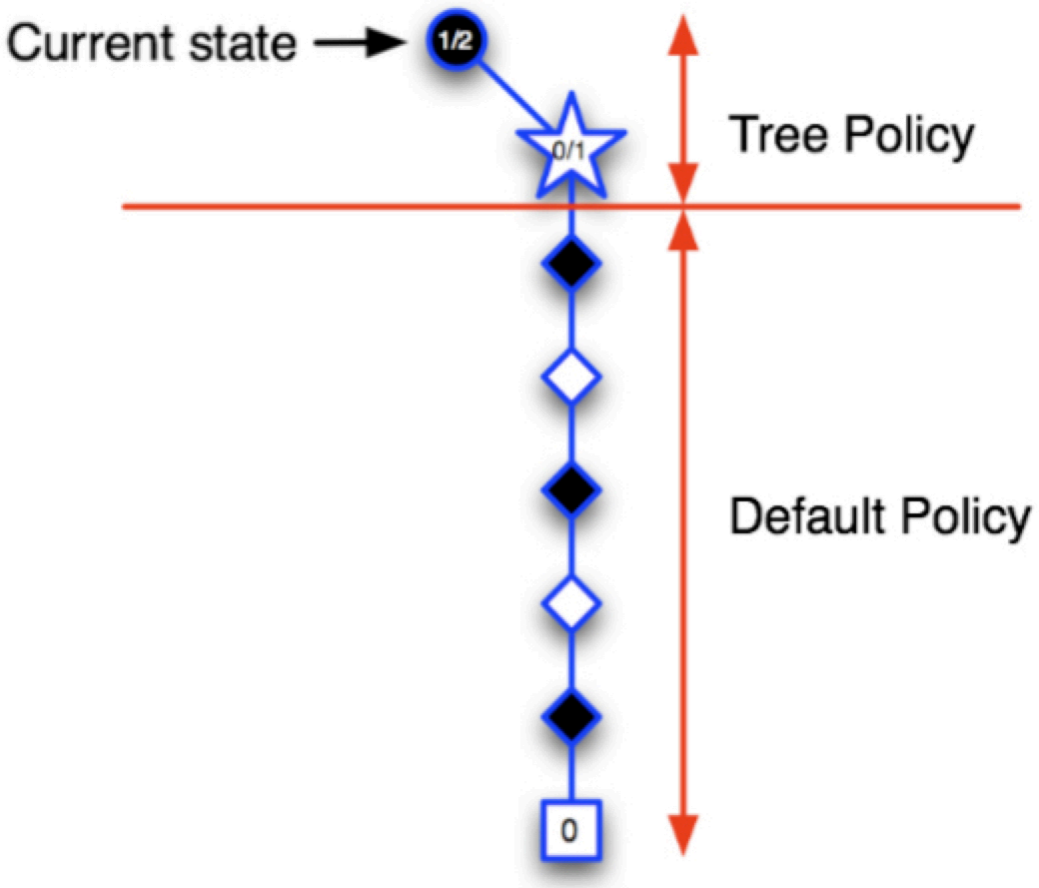
围棋中的 MC 评价



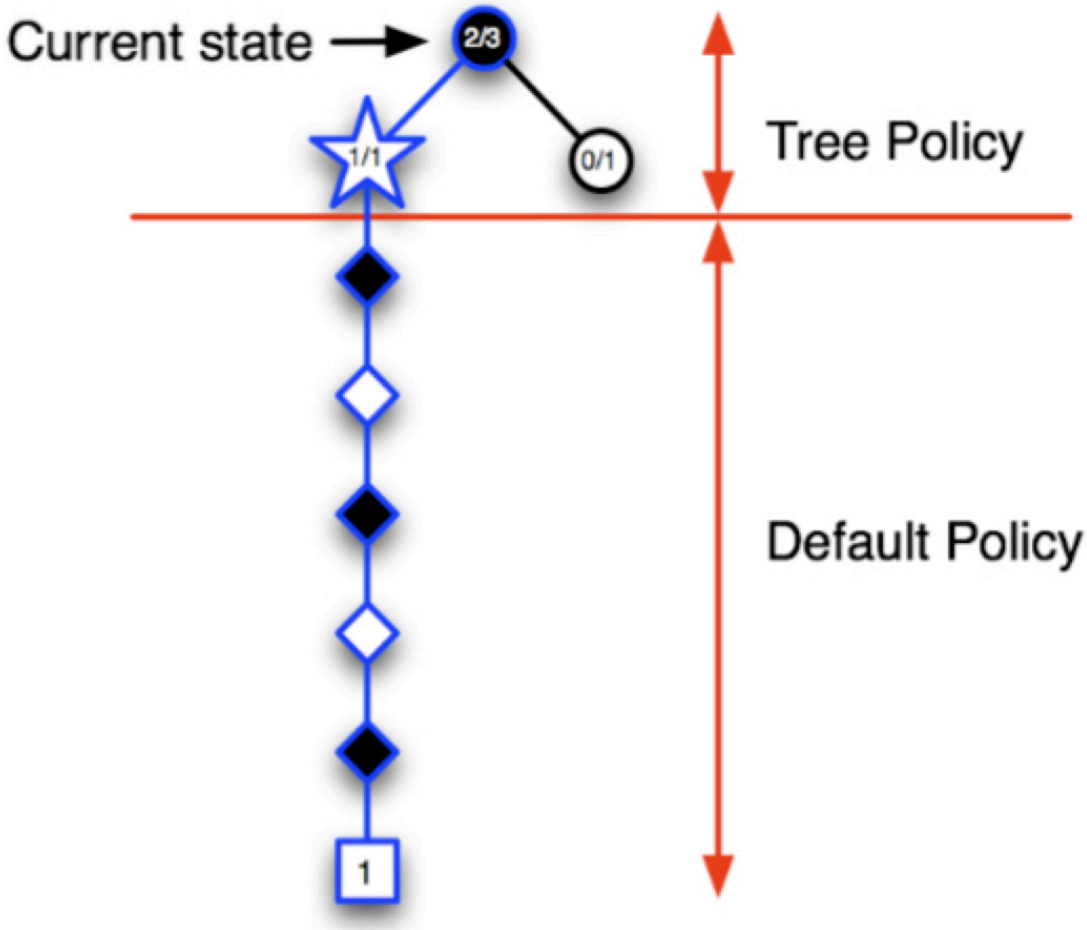
使用 MCTS(1)



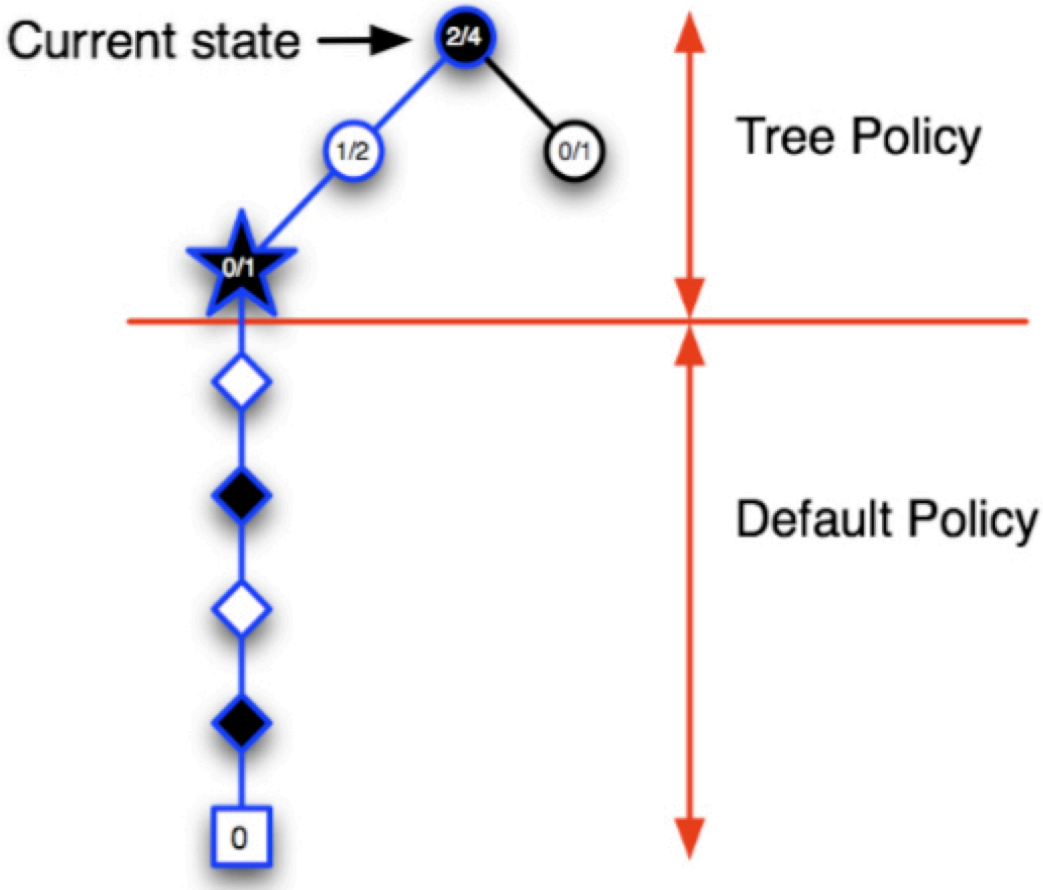
使用 MCTS(2)



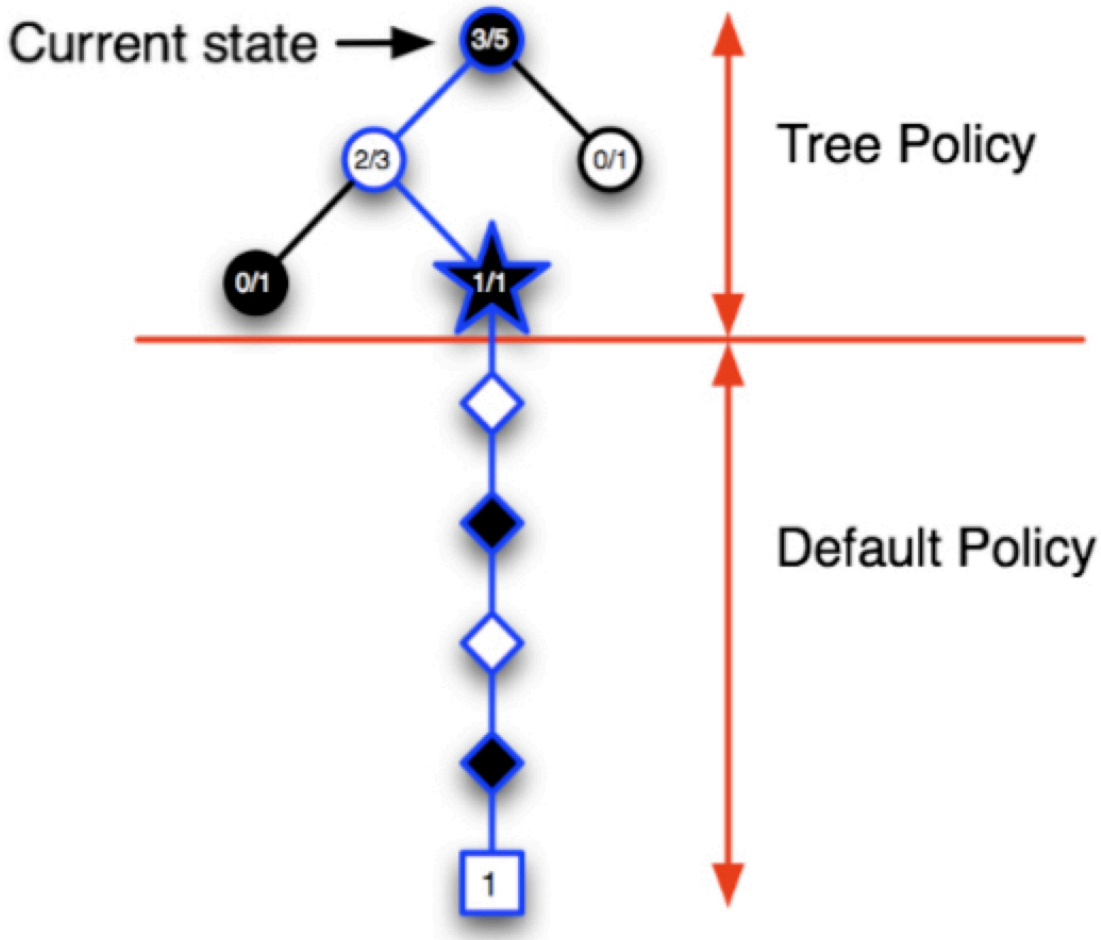
使用 MCTS(3)



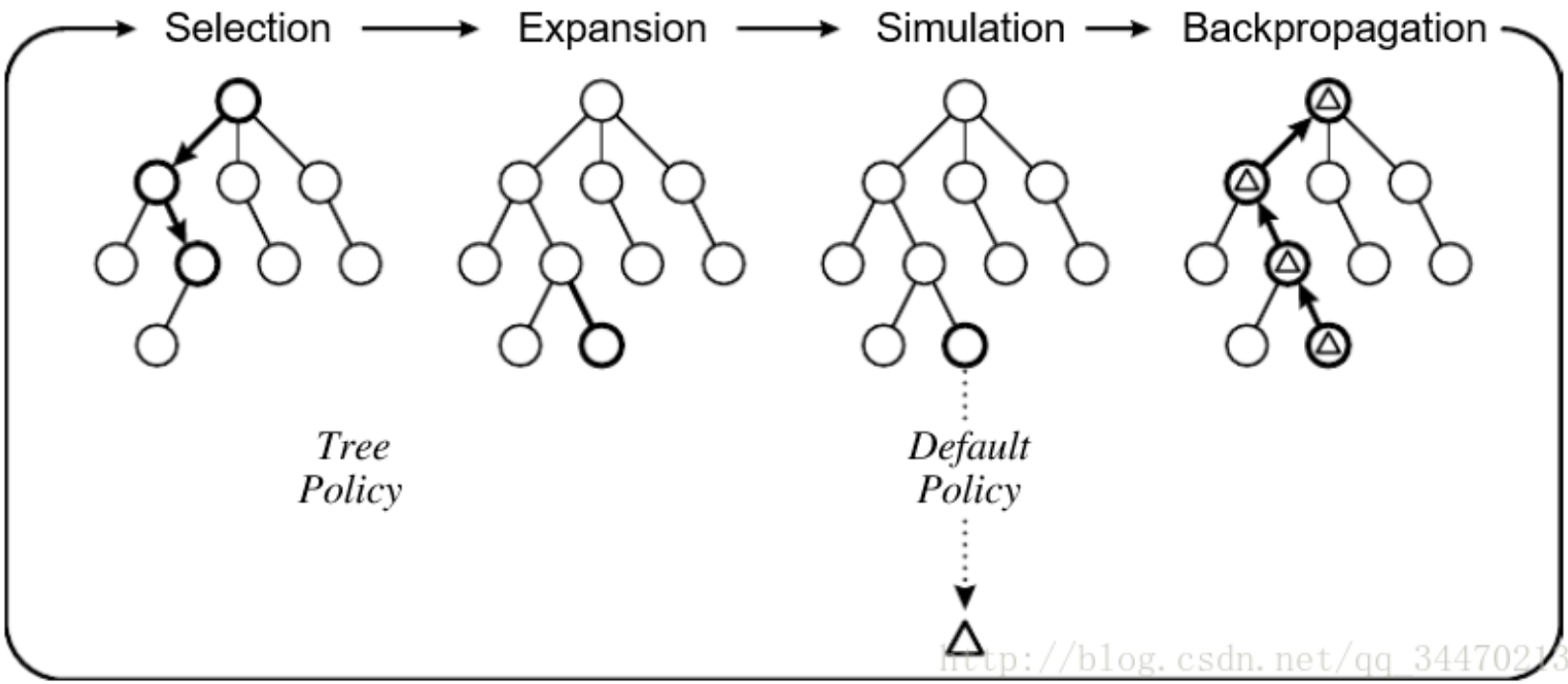
使用 MCTS(4)



使用 MCTS(5)



MCTS



MCTS 的优势

- Highly selective best-first search
- 动态评价状态
- 结合了采样去打破维度诅咒
- 适用于各种黑盒模型
- 计算有效，很容易并行

TD 搜索

- 基于采样的搜索
- 对于采样的经验使用 TD 去更新 (使用了自举)
- MCTS 使用 MC 优化算法
- TDTS 使用 Sarsa 优化算法

TD 搜索

- 从当前状态 s_t 开始采样片段
- 估计 $Q(s, a)$
- 对于每一步的仿真，使用 Sarsa 算法更新 Q 函数

$$\Delta Q(S, A) = \alpha \left(R + \gamma Q(S', A') - Q(S, A) \right)$$

- 基于 $Q(s, a)$ 选择动作