强化学习2024
第5节

涉及知识点：
规划与学习之入门算法和介绍、规划与学习之采样方法、规划与学习之决策时规划

基于规划的强化学习

张伟楠 – 上海交通大学

2024年上海交通大学ACM班强化学习课程大纲

强化学习基础部分

（中文课件）

1. 强化学习、探索与利用
2. MDP和动态规划
3. 值函数估计
4. 无模型控制方法
5. 规划与学习
6. 参数化的值函数和策略
7. 规划与学习
8. 深度强化学习价值方法
9. 深度强化学习策略方法

强化学习前沿部分

（英文课件）

9. 基于模型的深度强化学习
10. 离线强化学习
11. 模仿学习
12. 多智能体强化学习基础
13. 多智能体强化学习前沿
14. 基于扩散模型的强化学习
15. AI Agent与决策大模型
16. 技术交流与回顾

课程回顾

基于模型的动态规划

- 值迭代 $V(s) = r(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s')V(s')$
- 策略迭代 $\pi(s) = \arg\max_{a \in A} \sum_{s' \in S} P_{sa}(s')V(s')$

无模型的强化学习

- 在线策略蒙特卡洛 $V(s_t) \leftarrow V(s_t) + \alpha(g_t - V(s_t))$
- 在线策略时序差分 $V(s_t) \leftarrow V(s_t) + \alpha\big(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\big)$
- 在线策略时序差分 SARSA学习

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

- 离线策略时序差分 Q-学习

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$$

“思想总是走在行动的前面，就好像闪电总是走在雷鸣之前。”

德国诗人海涅

温故而知新:
策略评估与策略提升

讲师：张伟楠 – 上海交通大学

策略值函数估计（Policy Evaluation）

- 给定环境MDP和策略π，策略值函数估计如下

状态价值 $$V^{\pi}(s) = \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots | S_0 = s, \pi]$$

$$= \mathbb{E}_{a \sim \pi(s)}\left[r(s,a) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s') \right]$$

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

动作价值 $$Q^{\pi}(s,a) = \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots | S_0 = s, A_0 = a, \pi]$$

$$= r(s,a) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s')$$

策略提升（Policy Improvement）

- 对于两个策略π，π'，如果满足如下性质，π'是π的策略提升：
  - 对于任何状态s，有

$$Q^{\pi}\big(s,\pi'(s)\big) \geq V^{\pi}(s)$$

以π来记回报

- 一种特例：给定环境MDP和两个策略π，π'，如果满足如下性质：
  1. 在某个状态s下，两策略的输出不同，并且有

$$\pi'(s) \neq \pi(s) \qquad Q^{\pi}\big(s,\pi'(s)\big) > Q^{\pi}\big(s,\pi(s)\big) = V^{\pi}(s)$$

  2. 在其他所有状态s'下，两策略输出相同，即

$$\pi'(s') = \pi(s') \qquad Q^{\pi}\big(s,\pi'(s)\big) = Q^{\pi}\big(s,\pi(s)\big) = V^{\pi}(s)$$

那么π'是π的一种策略提升

策略提升定理（Policy Improvement Theorem）

- 对于两个策略π，π'，如果满足如下性质，π'是π的策略提升：
  - 对于任何状态s，有

$$Q^{\pi}\big(s,\pi'(s)\big) \geq V^{\pi}(s)$$

以π来记回报

- 进而，π和π'满足：对任何状态s，有

$$V^{\pi'}(s) \geq V^{\pi}(s)$$

以π'来记回报

也即是π'的策略价值（期望回报）超过π，π'比π更加优秀。

策略提升定理（Policy Improvement Theorem）

- 对于两个策略π，π'，如果满足如下性质， π'是π的策略提升：
  - 对于任何状态s，有$Q^{\pi}(s,\pi'(s)) \geq V^{\pi}(s)$，因此有$V^{\pi'}(s) \ \geq V^{\pi}(s)$
- 证明：

$$
\begin{aligned}
V^{\pi}(s) &\leq Q^{\pi}(s, \pi'(s)) \\
&= \mathbb{E}_{\text{Env}}[R_t + \gamma V^{\pi}(S_{t+1}) | S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_t + \gamma V^{\pi}(S_{t+1}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_t + \gamma Q^{\pi}(S_{t+1}, \pi'(S_{t+1})) | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_t + \gamma \mathbb{E}_{\text{Env}}[R_{t+1} + \gamma V^{\pi}(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_t + \gamma R_{t+1} + \gamma^2 V^{\pi}(S_{t+2}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 V^{\pi}(S_{t+3}) | S_t = s] \\
&\cdots \\
&\leq \mathbb{E}_{\pi'}[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots | S_t = s] \\
&= V^{\pi'}(s)
\end{aligned}
$$

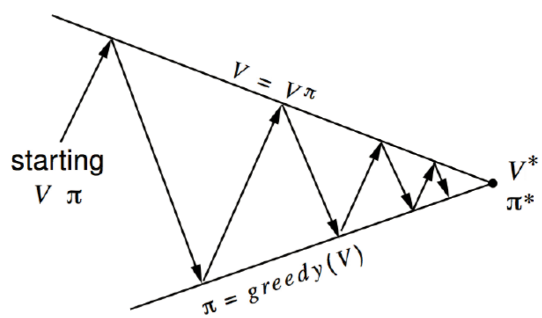
策略提升定理（Policy Improvement Theorem）

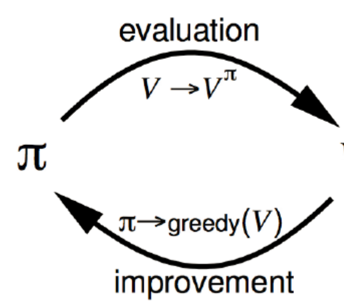
- 对于两个策略π，π'，如果满足如下性质， π'是π的策略提升：
  - 对于任何状态s，有$Q^{\pi}\big(s,\pi'(s)\big) \geq V^{\pi}(s)$
  - 因此有$V^{\pi'}(s) \ \geq V^{\pi}(s)$

- 策略提升定理带给我们的启示

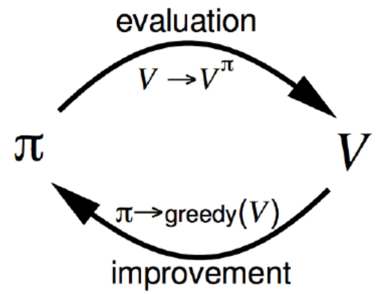
[找到(s,a)使得 $Q^{\pi}(s,a) \geq V^{\pi}(s)$]

策略 π ⟶ 评估策略价值 $V^{\pi}(\cdot)$ ⟶ 寻找到新策略π'使得 $Q^{\pi}\big(s,\pi'(s)\big) \geq V^{\pi}(s)$





价值评估指导策略提升



价值评估指导策略提升
那么如何更加精准地估计价值呢？

规划与学习:
入门介绍和算法

讲师：张伟楠 – 上海交通大学

目录

Contents

模型 (Model)

- 给定一个状态和动作，模型能够预测下一个状态和奖励的分布: 即 $p(s', r \mid s, a)$
  - s, a: 给定的状态和动作
  - r: 奖励
  - s'：下一个状态

模型的分类

- 分布模型(distribution model)
  - 描述了轨迹的所有可能性及其概率
- 样本模型(sample model)
  - 根据概率进行采样，只产生一条可能的轨迹

模型(Model)

举例

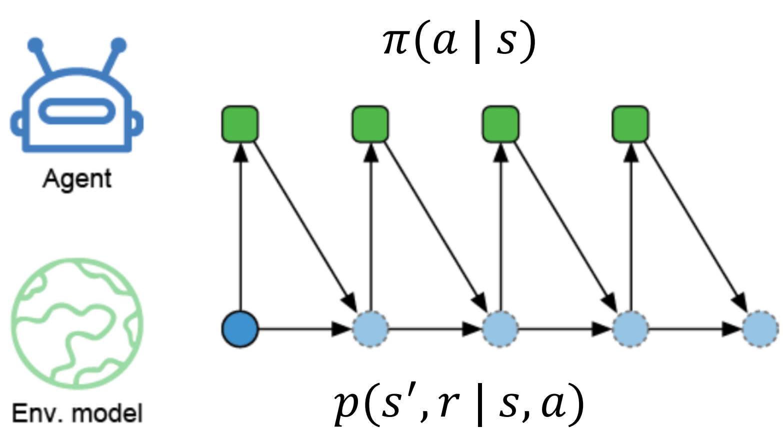
- 掷骰子(Dozen Dice Games)
  - 分布模型
    - 得到骰子数字总和的所有可能性及其概率
  - 样本模型
    - 只采样得到一种骰子数字总和

模型的作用

- 得到模拟的经验数据(simulated experiences)

智能体策略

环境模型



经验数据

$\{(s, a, r, s')\}$

规划(Planning)

- 输入一个模型，输出一个策略的搜索过程

$$\text{模型} \xrightarrow{\text{规划}} \text{策略}$$

规划的分类

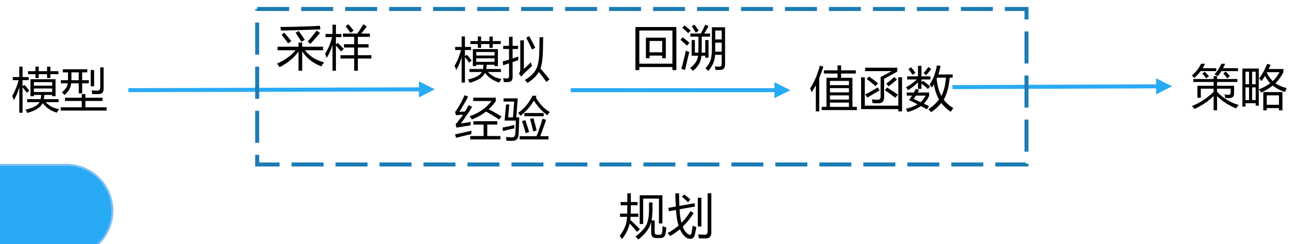
- 状态空间的规划 (state-space planning)
  - 在状态空间搜索最佳策略，本课程主要围绕这种
- 规划空间的规划 (plan-space planning)
  - 在规划空间搜索最佳策略，包括遗传算法和偏序规划
  - 这时，一个规划就是一个动作集合以及动作顺序的约束
  - 这时的状态就是一个规划，目标状态就是能完成任务的规划

https://www.cs.uni.edu/~wallingf/teaching/161/sessions/session24.pdf

规划(Planning)

规划的通用框架

- 通过模型采样得到模拟数据
- 利用模拟数据更新值函数从而改进策略



举例

- 动态规划
  - 搜索整个状态空间，生成所有的状态转移分布
  - 状态转移分布回溯更新状态的值函数

规划的好处

- 任何时间点可以被打断或者重定向
- 在复杂问题下，进行小而且增量式的时间步规划是很有效的

规划与学习(Planning and Learning)

- 不同点
  - 规划：利用**模型**产生的模拟经验
  - 学习：利用**环境**产生的真实经验
- 相同点
  - 通过回溯(back-up)更新值函数的估计
  - 统一来看，学习的方法可以用在模拟经验上

算法: 一时间步随机采样表格 Q 规划 (Q-planning)

重复以下步骤:

1. 随机选择一个状态 $s \in \mathcal{S}$ 和一个动作 $a \in \mathcal{A}(S)$
2. 把 s, a 输入采样模型，然后获得采样得到的奖励 r 和下一个状态 s'
3. 对 s, a, r, s' 进行一时间步表格 Q 学习:

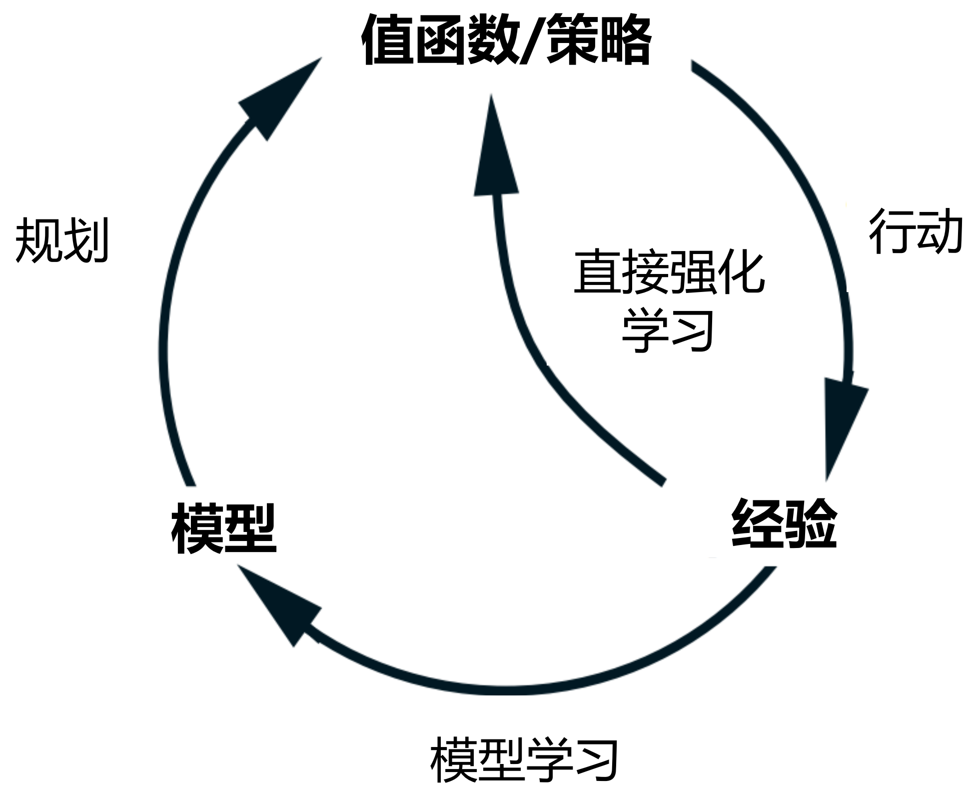
$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s,a) \right]$$

Dyna (集成规划、决策和学习)

经验的不同用途

- 更新模型
  - 模型学习, 或间接强化学习
  - 对经验数据的需求少

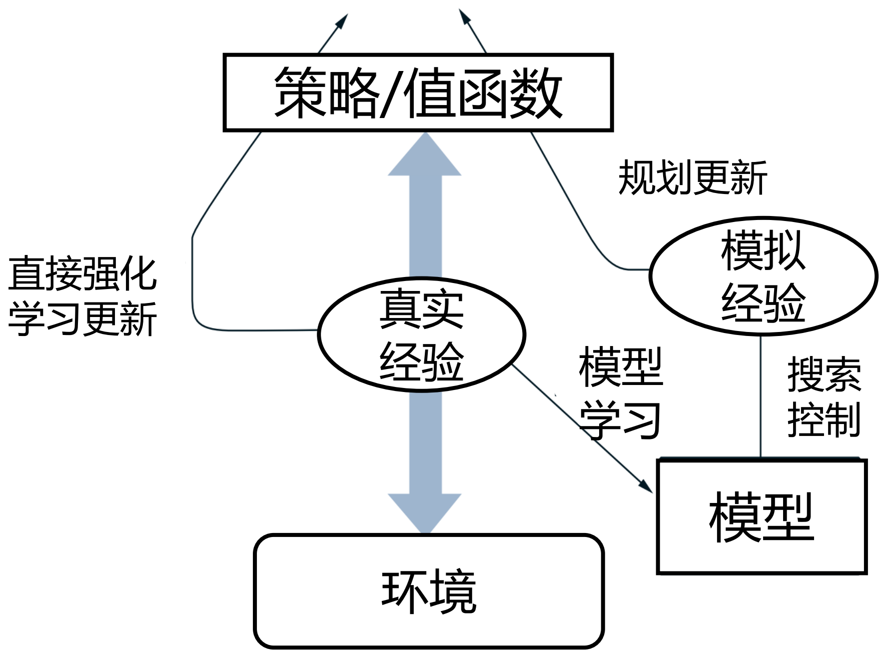
- 更新值函数和策略
  - 直接强化学习（无模型强化学习）
  - 简单且不受模型偏差的影响



Dyna

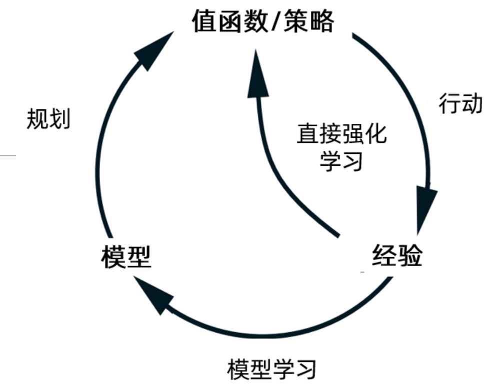
Dyna的框架

- 和环境交互产生真实经验
- 左边代表直接强化学习
  - 更新值函数和策略
- 右下角落边代表学习模型
  - 使用真实经验更新模型
- 右边代表基于模型的规划
  - 基于模型随机采样得到模拟经验
    - 只从以前得到的状态动作对随机采样
  - 使用模拟经验做规划更新值函数和策略



Sutton, Richard S. "Dyna, an integrated architecture for learning, planning, and reacting." ACM Sigart Bulletin 2.4 (1991): 160-163.

Dyna



- $Model(s,a)$: 预测 (s,a) 对的下一个状态和奖励
- 步骤(5), (6)去掉就是一时间步表格Q学习

算法: 表格 $\textbf{Dyna} - \textbf{Q}$

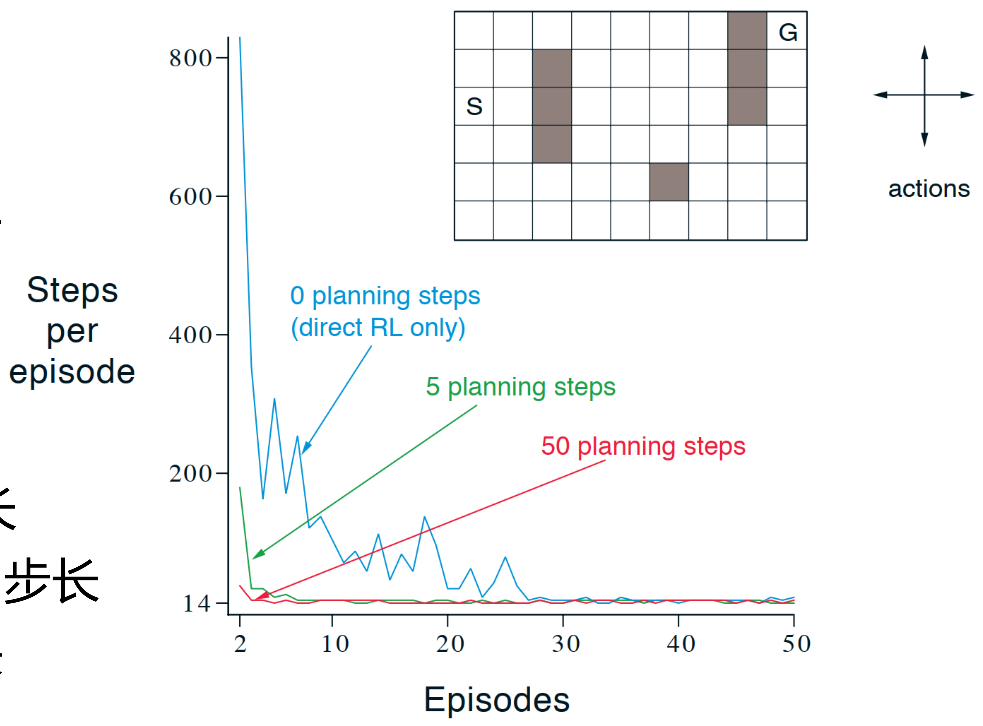
对于所有的 $s \in \mathcal{S}$ 和 $a \in \mathcal{A}(S)$，初始化值函数 $Q(s,a)$ 和模型 $Model(s,a)$
重复以下步骤:

1. 令 $s \leftarrow$ 当前（非终止）状态
2. 令 $a \leftarrow \epsilon$-greedy(s,Q)
3. 做动作 a；得到奖励 r 和状态s'
4. 令 $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
5. 令$Model(s,a) \leftarrow r, s'$（假设是确定性环境)
6. 重复以下步骤n次:
   a. 令 $s \leftarrow$ 随机采样之前见过的状态
   b. 令 $a \leftarrow$ 随机采样之前在状态S做过的动作
   c. 令 $r, s' \leftarrow Model(s,a)$
   $d.$ $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$

Dyna

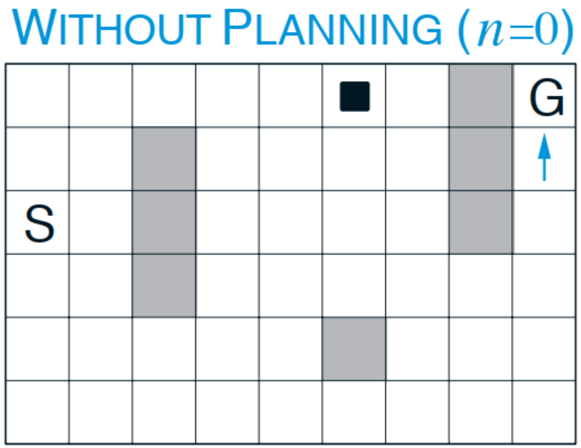
举例 1: 迷宫

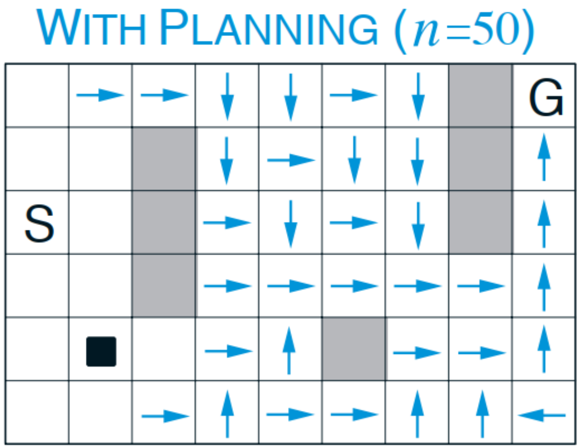
- 环境
  - 4个动作(上下左右)
  - 碰到障碍物和边界静止
  - 到达目标（G），得到奖励+1
  - 折扣因子 0.95
- 结果
  - 横轴代表游戏轮数
  - 纵轴代表到达 G 花的时间步长
  - 不同曲线代表采用不同的规划步长
  - 规划步长越长，表现收敛越快



Dyna

为什么更快





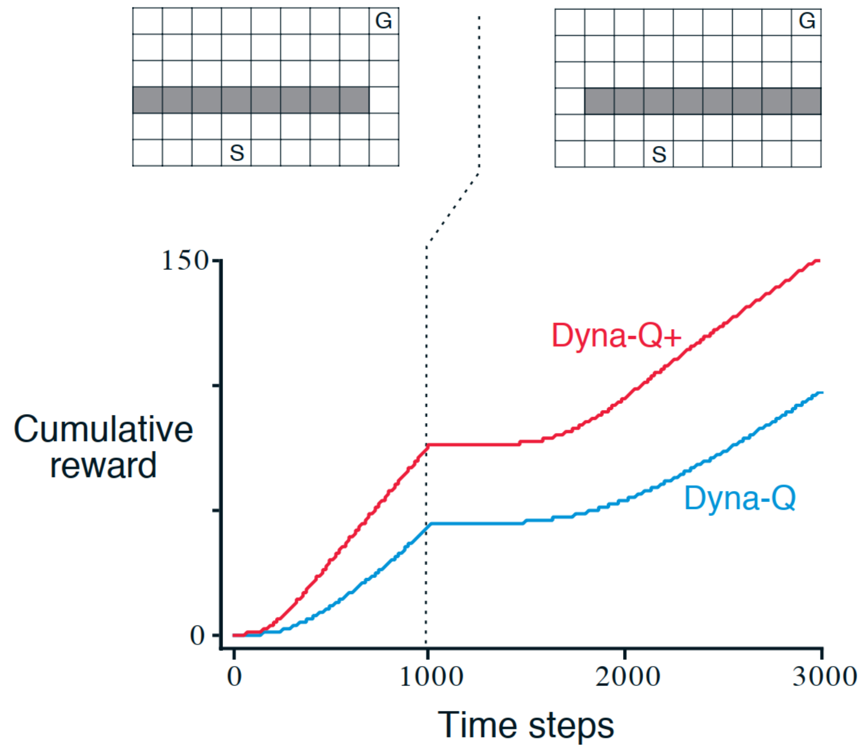
模型不准了怎么办

- 环境是随机的，并且只观察到了有限的样本
- 模型使用了泛化性不好的函数估计
- 环境改变了，并且还没有被算法检测到

Dyna

举例1：阻碍迷宫

- 环境:
  - 1000步障碍向右移动
- 结果:
  - 横轴代表时间步
  - 纵轴代表累计的收益
  - Dyna-Q+加了探索



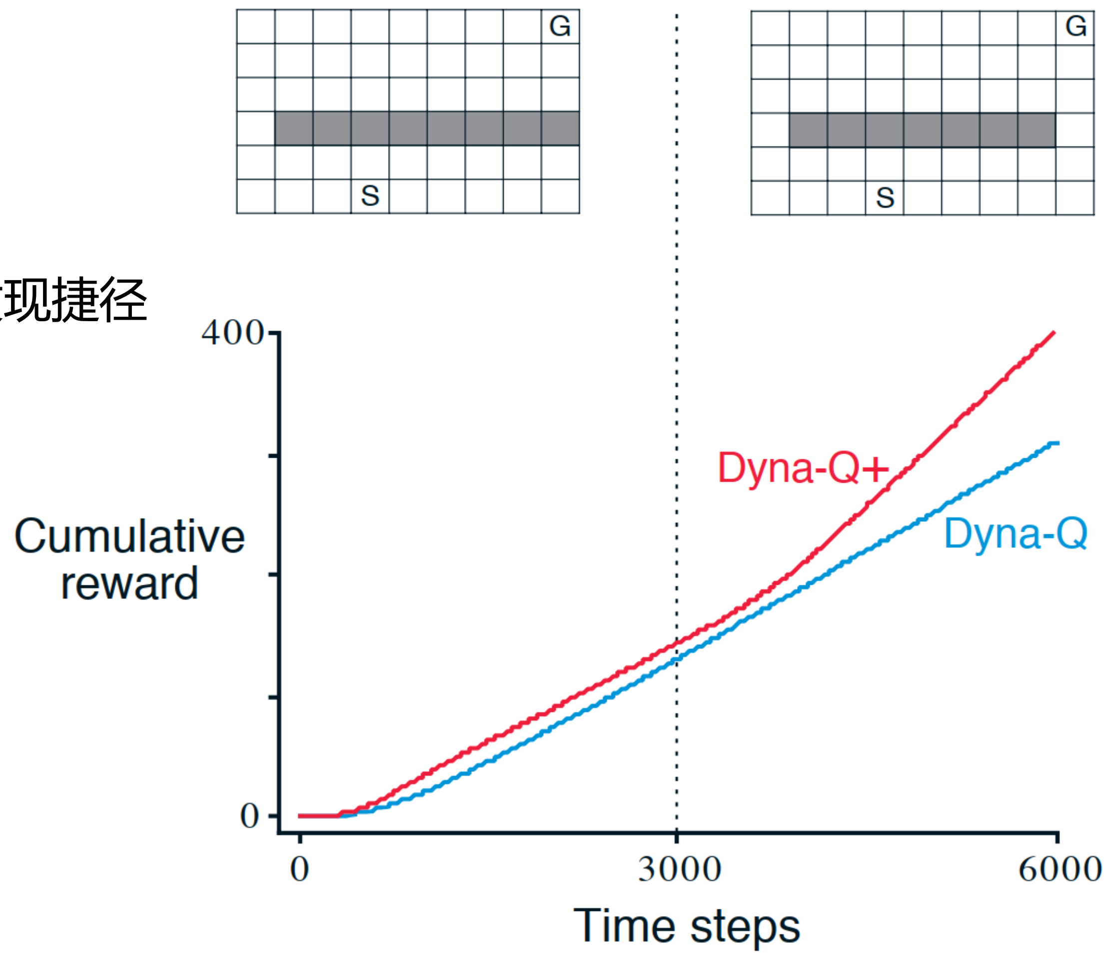
Dyna-Q+

- 奖励更改为 $r + \mathcal{K}\sqrt{\tau}$
  - r: 原来的奖励
  - $\mathcal{K}$: 小的权重参数
  - τ: 某个状态多久未到达过了

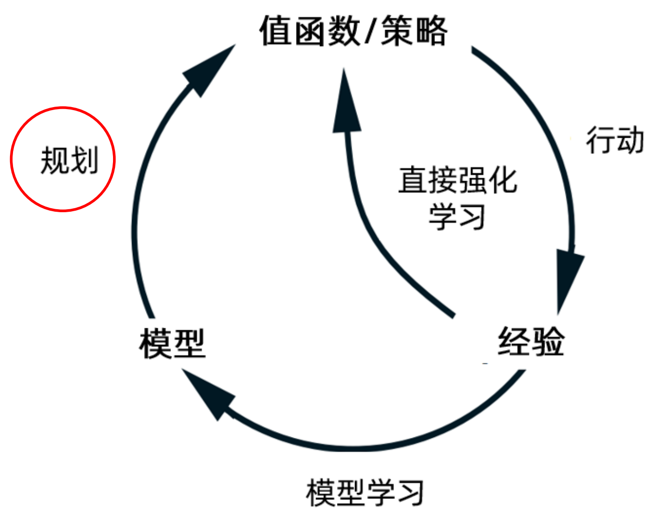
Dyna

举例2: 捷径迷宫

- 环境:
  - 3000步出现捷径
- 结果:
  - Dyna-Q+能够发现捷径



$$\text{模型} \xrightarrow{\text{采样}} \text{模拟经验} \xrightarrow{\text{回溯}} \text{值函数} \longrightarrow \text{策略}$$

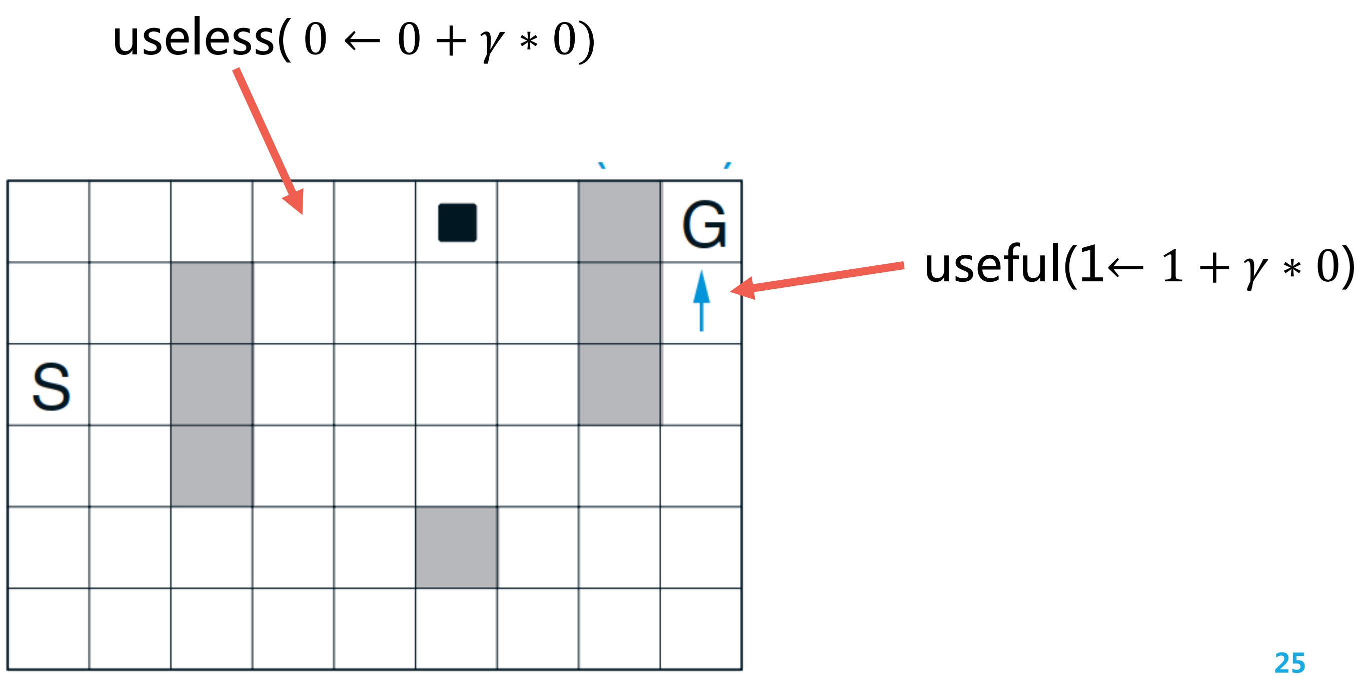


规划与学习: 采样方法

讲师：张伟楠 – 上海交通大学

常用的采样方法

- 均匀随机采样
- 模拟的经验和更新应集中在一些特殊的状态动作



更好的采样方法

- 后向聚焦（backward focusing）：
  - 很多状态的值发生变化带动前继状态的值发生变化
- 有的值改变很多，有的改变很少
  - 因此需要根据紧急程度，给这些更新设置优先度进行更新

优先级采样

- 设置优先级更新队列
  - 根据值改变的幅度定义优先级：$P \leftarrow \left| r + \gamma \max_{a'} Q\left(s', a'\right) - Q(s, a) \right|$

采样方法：优先级采样

算法: 确定性环境中的优先级采样

对于所有的$s \in \mathcal{S}$和$a \in \mathcal{A}(S)$，初始化值函数$Q(s,a)$和模 $Model(s,a)$；初始化优先级队列$PQueue$为空

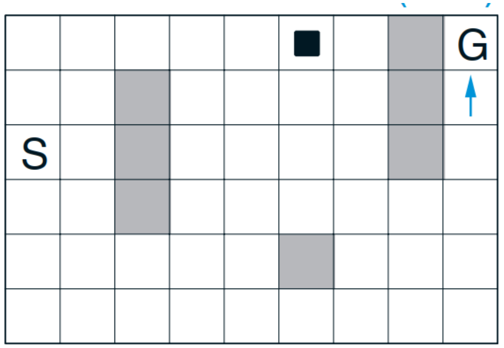
重复以下步骤:

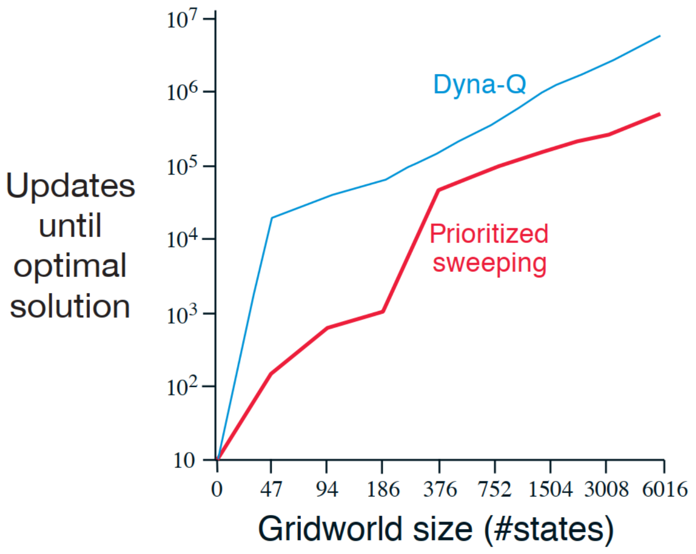
1. 令$s \leftarrow$ 当前（非终止）状态
2. 令$a \leftarrow \epsilon$-greedy(s,Q)
3. 做动作a；得到奖励r和状态s'
4. 令$P \leftarrow \left| r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right|$
5. 如果$P > \theta$, 将s,a以优先级P插入$PQueue$
6. 重复以下步骤n次:
   a. 令$s,a \leftarrow PQueue$队列头元素
   b. 令$r, s' \leftarrow Model(s,a)$
   c. $Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$
   d. 对于所有能够到达s的$\bar{s}, \bar{a}$：
      a. 令$\bar{r} \leftarrow$模型对于$\bar{s}, \bar{a}, s$预测的奖励
      b. 令$P \leftarrow \left| \bar{r} + \gamma \max_{a'} Q(s,a') - Q(\bar{s},\bar{a}) \right|$
      c. 如果$P > \theta$, 将$\bar{s}, \bar{a}$以优先级P插入$PQueue$

采样方法：优先级采样

举例: 迷宫

- 横轴代表格子世界的大小
- 纵轴代表收敛到最优策略的更新次数
- 优先级采样收敛更快





局限性及改进

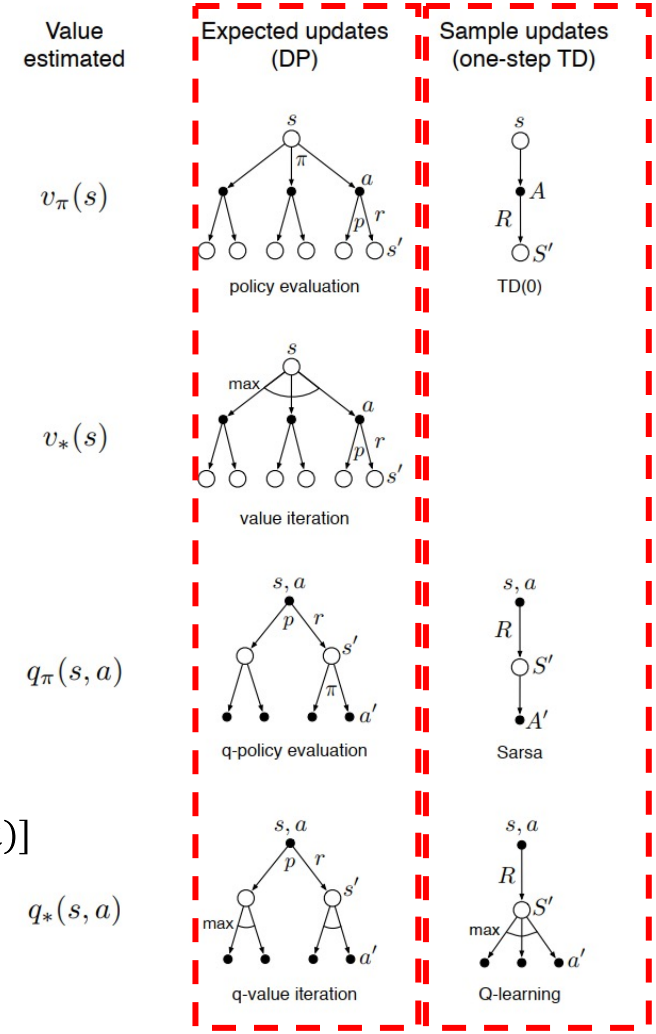
- 随机环境中利用期望更新（expected updates）的方法
  - 浪费很多计算资源在一些低概率的状态转移（transitions）上
- 引入采样更新（sample updates）

采样方法：期望更新和采样更新

- 值函数：$V(s)$
- 动作值函数：$Q(s,a)$
- 期望更新或者采样更新

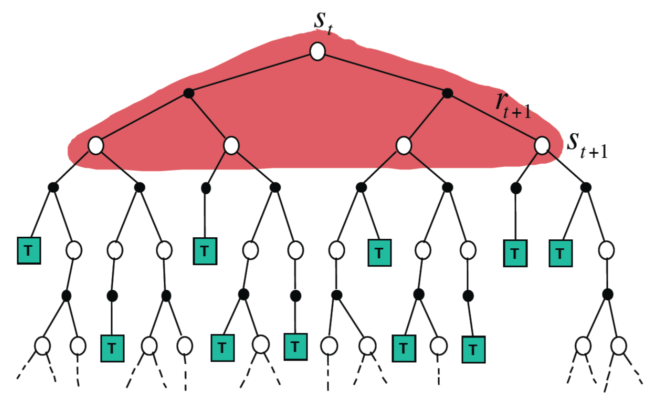
比较

- 期望更新 $Q(s,a) \leftarrow \sum_{s',r} \hat{p}(s',r\,|s,a)[r+\gamma \max_{a'} Q(s',a')]$
  - 需要分布模型
  - 需要更大的计算量
  - 没有偏差更准确
- 采样更新 $Q(s,a) \leftarrow Q(s,a) + \alpha[r+\gamma \max_{a'} Q(s',a') - Q(s,a)]$
  - 只需要采样模型
  - 计算量需求更低
  - 受到采样误差(sampling error)的影响



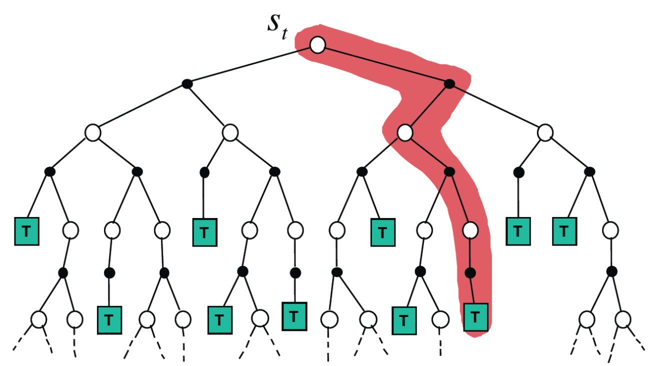
采样方法：轨迹采样

- 动态规划
  - 对整个状态空间进行遍历
  - 没有侧重实际需要关注的状态上
- 在状态空间中按照特定分布采样
  - 根据当前策略下所观测的分布进行采样



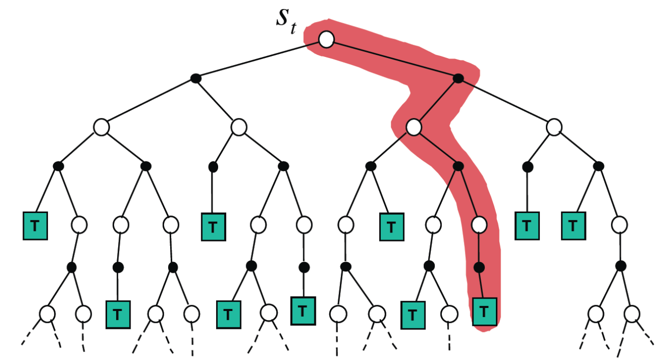
轨迹采样

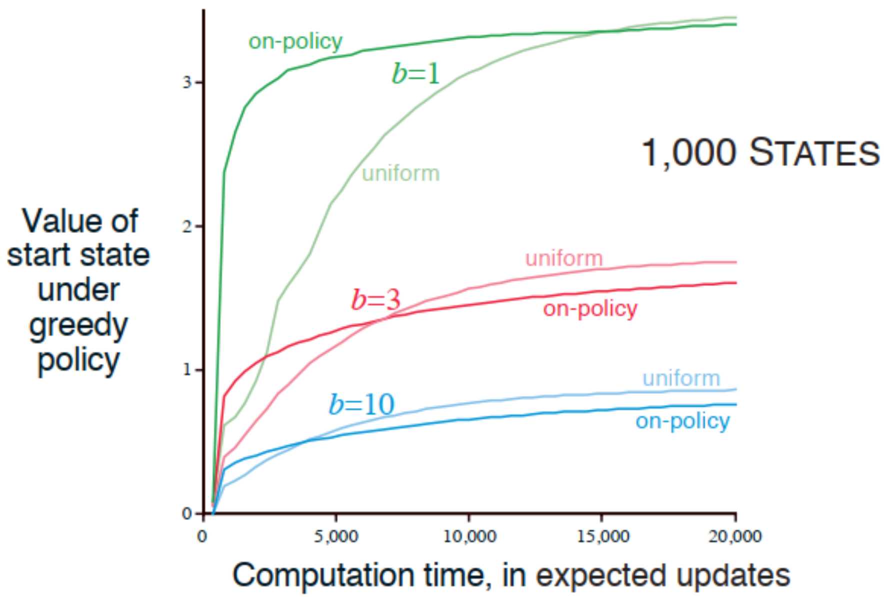
- 状态转移和奖励由模型决定
- 动作由当前的策略决定



采样方法：轨迹采样

- 优点
  - 不需要知道当前策略下状态的分布
  - 计算量少，简单有效
- 缺点
  - 不断重复更新已经被访问的状态





- 不同的分支因子下的表现
- 确定性环境中表现比较好

小结

- 优先级采样
  - 收敛更快
  - 随机环境使用期望更新，计算量大
- 期望更新和采样更新
  - 期望更新计算量大但是没有偏差
  - 采样更新计算量小但是存在采样偏差
- 轨迹采样
  - 采样更新，计算量小
  - 不断重复某些访问过的状态

规划与学习: 决策时规划

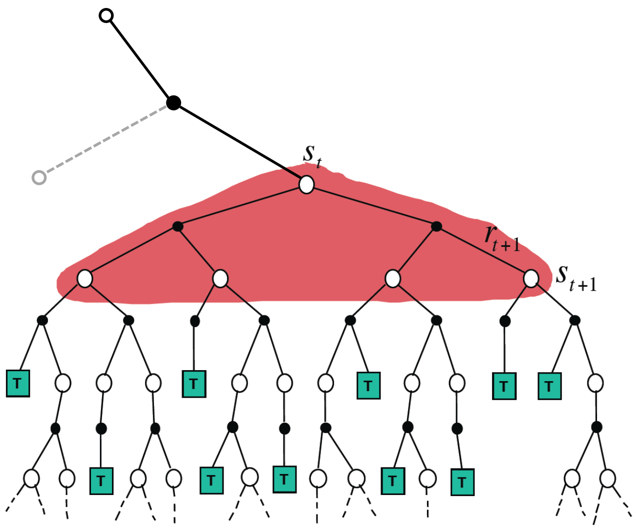
讲师：张伟楠 – 上海交通大学

目录

Contents

实时动态规划

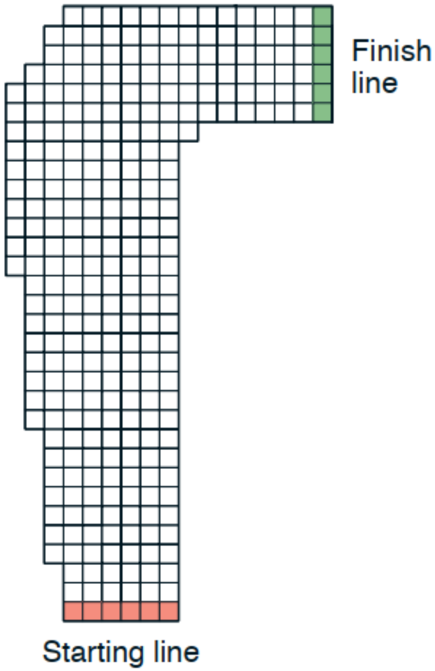
- 和传统动态规划的区别
  - 实时的轨迹采样
  - 只更新轨迹访问的状态值

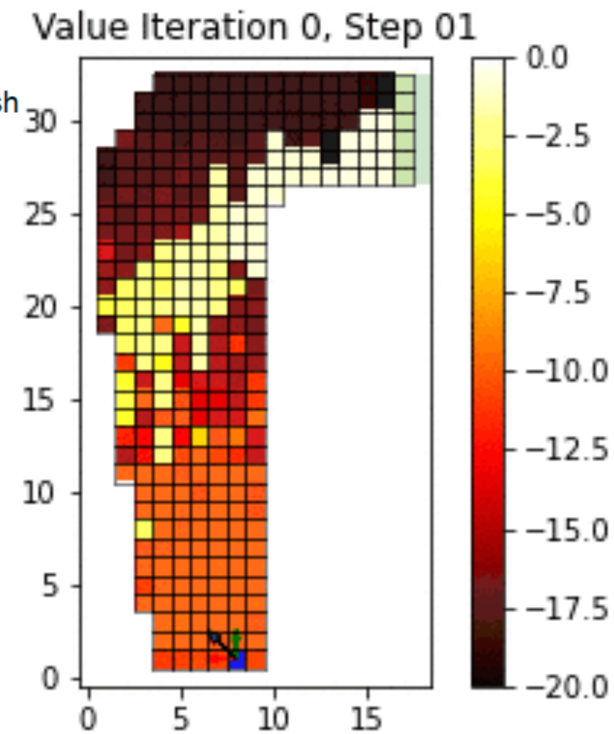


- 优势
  - 能够跳过策略无关的状态
  - 在解决状态集合规模大的问题上具有优势
  - 满足一定条件下可以以概率1收敛到最优策略

实时动态规划(RTDP)

- 跑道问题（Racetrack）
- 环境:
  - 任务: 从起点跑到终点
  - 状态: 二维坐标、二维速度
  - 动作: 每维速度的+1，-1，不变
- 结果:
  - 可到达状态:
    - 随机策略: 9115
    - 最优策略: 599
  - 更新次数少了一半





	DP	RTDP
Average computation to convergence	28 sweeps	4000 episodes
Average number of updates to convergence	252,784	127,600
Average number of updates per episode	—	31.9
% of states updated ≤ 100 times	—	98.45
% of states updated ≤ 10 times	—	80.51
% of states updated 0 times	—	3.18

https://towardsdatascience.com/solving-racetrack-in-reinforcement-learning-using-monte-carlo-control-bdee2aa4f04e

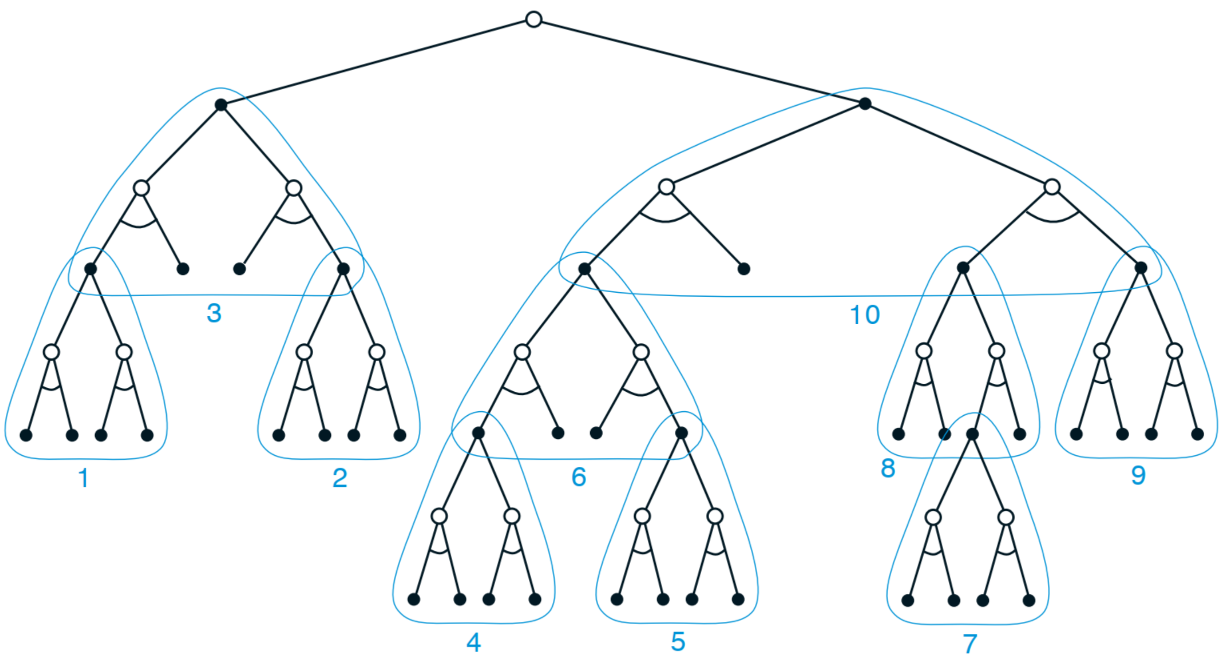
决策时规划

- 背景规划（Background Planning）
  - 规划是为了更新很多状态值供后续动作的选择
  - 如动态规划，$Dyna$

- 决策时规划（Decision-time Planning）
  - 规划只着眼于当前状态的动作选择
  - 在不需要快速反应的应用中很有效，如棋类游戏

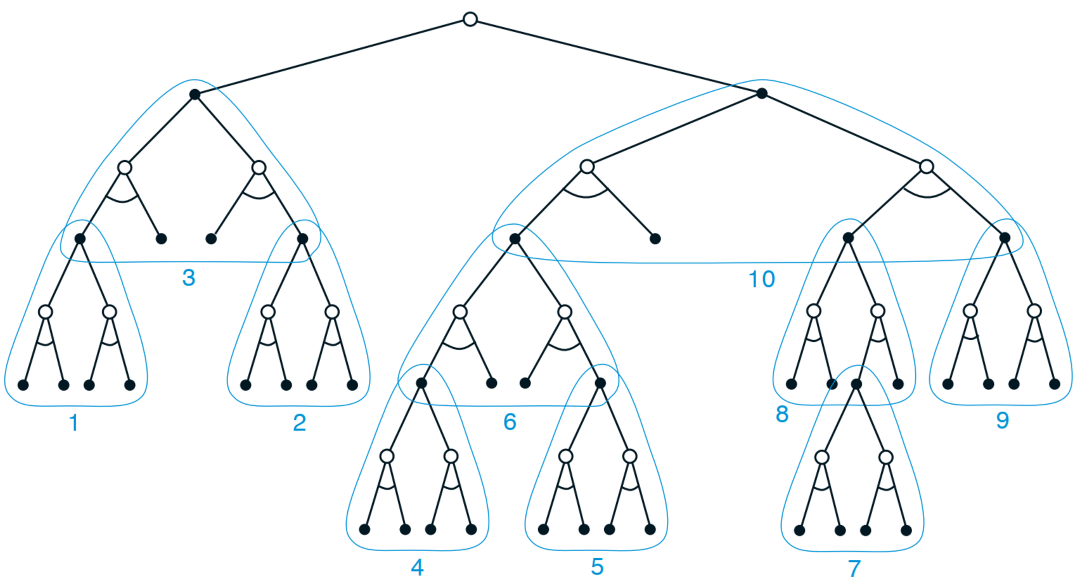
启发式搜索

- 访问到当前状态(根节点)，对后续可能的情况进行树结构展开
- 叶节点代表估计的值函数
- 回溯到当前状态(根节点)，方式类似于值函数的更新方式



启发式搜索

- 决策时规划，着重于当前状态
- 贪婪策略在单步情况下的扩展
  - 启发式搜索看多步规划下，当前状态的最优行动
- 搜索越深，计算量越大，得到的动作越接近最优
- 性能提升不是源于多步更新，而是源于专注当前状态的后续可能



Rollout算法

- 从当前状态进行模拟的蒙特卡洛估计
- 选取最高估计值的动作
- 在下一个状态重复上述步骤

特点

- 决策时规划，从当前状态进行$rollout$
- 直接目的类似于策略迭代和改进，寻找更优的策略
- 表现取决于蒙特卡洛方法估值的准确性

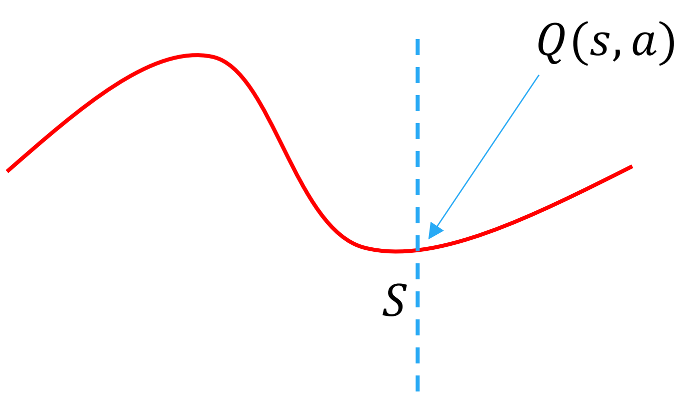
Rollout算法

时间复杂度

- $Time = \sum_{tr=1}^{N} \sum_{t=1}^{K} [\sum_{a=1}^{A} T_{eval}(S(t), a) + T_{choose}(A)]$
  - A: 决策的动作空间
  - K: $rollout$ 一个轨迹的平均步数
  - $T_{eval}(S(t), a)$: 在第 s 步下，估计 (s, a) 值函数的时间
  - $T_{choose}(A)$: $rollout$ 每步做出决策的时间
  - N: $rollout$ 轨迹的次数

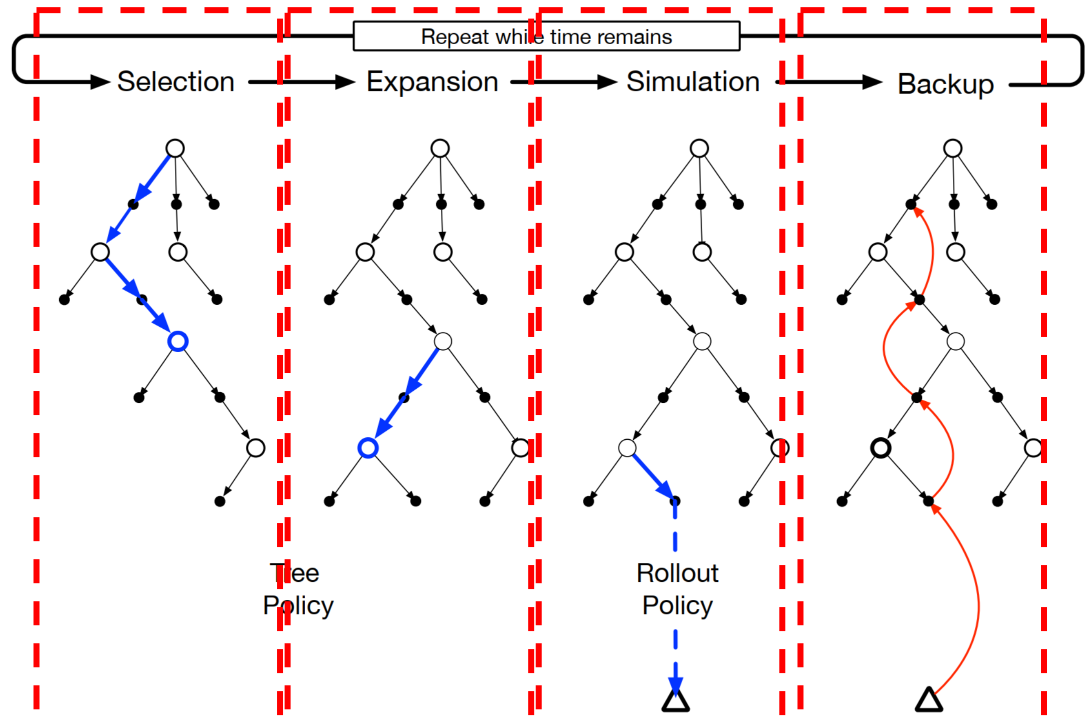
Rollout算法的加速方法

- 多个处理器并行采样
- 轨迹截断，用存储的值估计代替回报
- 剔除不可能成为最佳动作的动作



蒙特卡洛树搜索

1. 选择: 根据树策略(动作值函数)遍历树到一个叶节点
2. 扩展: 从选择的叶节点出发选择未探索过的动作到达新的状态
3. 模拟: 从新的状态出发按照 $rollout$ 策略进行轨迹模拟
4. 回溯: 得到的回报回溯更新树策略, $rollout$ 访问的状态值不会被保存
5. 重复上述步骤直至计算资源耗尽，从根节点选择最优动作
6. 得到新状态, 保留原有树的新状态下的部分节点
7. 重复上述步骤直至游戏结束



蒙特卡洛树搜索

- 蒙特卡洛控制+决策时规划（类似于 $rollout$）
- 保留了过去一部分的经验数据。
  - 下一个状态树的初始树是上一个状态树具有高回报的部分

应用

- AlphaGo
  - 16 年五番棋比赛中 4:1 李世石
  - 17 年乌镇围棋峰会中 3:0 柯洁

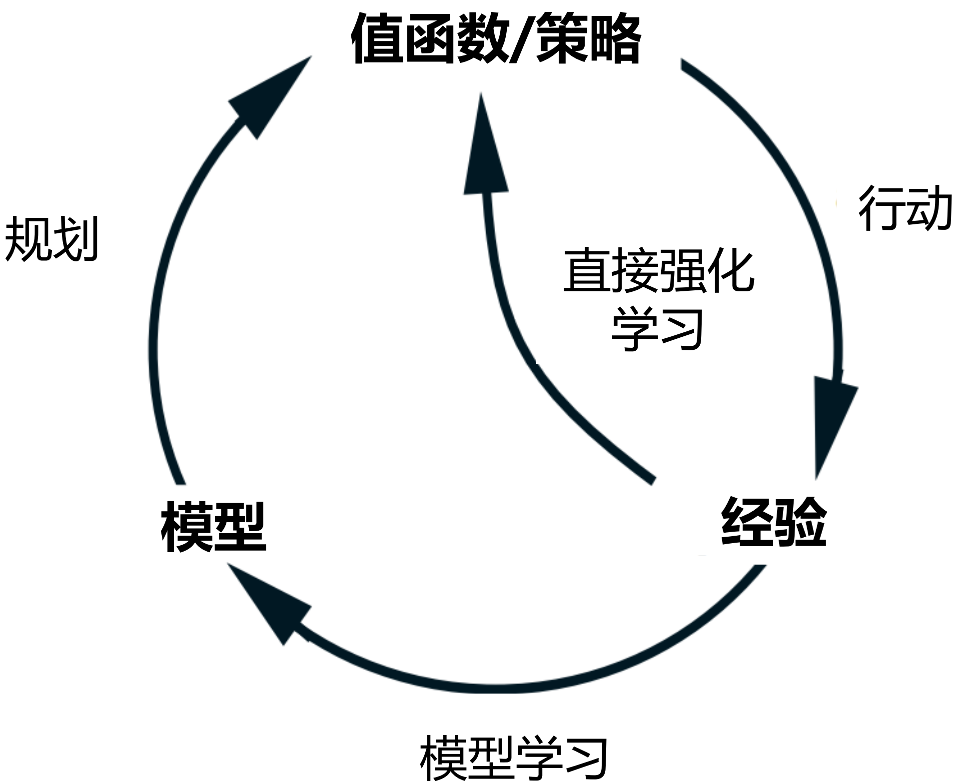
基于规划的强化学习方法总结

- 模型和规划
- 模型是什么
- 规划是什么

- 基于模型的算法
  - Dyna
  - Dyna-Q+

- 期望更新和采样更新
  - 优先级采样
  - 轨迹采样

- 实时动态规划
- 决策时规划
  - 启发式算法
  - Rollout 算法
  - 蒙特卡洛树搜索



THANK YOU