



清华大学

强化学习基本思想和问题模型

Basic Ideas of Reinforcement Learning



内容主要来自: Reinforcement Learning An Introduction Richard S. Sutton Andrew G. Barto

主讲人: 王乐业 2023年春

本讲主写人 李文新



1. 强化学习的例子

- 打砖块AI、王者鲁班七号AI

2. 强化学习的生物学基础

- 强化理论 - 斯金纳箱

3. 强化学习的问题模型和求解过程

- 问题模型（环境、智能体、策略 π ）、求解过程（小麻雀 - 井字棋）
- 问题模型的泛化（带概率的 P, R, π ）、状态价值 V_π 和动作价值 q_π

4. 寻找最优策略的几种思路

- 多臂老虎机、动作价值、贪心、 ϵ -贪心、UCB、梯度下降、方法比较



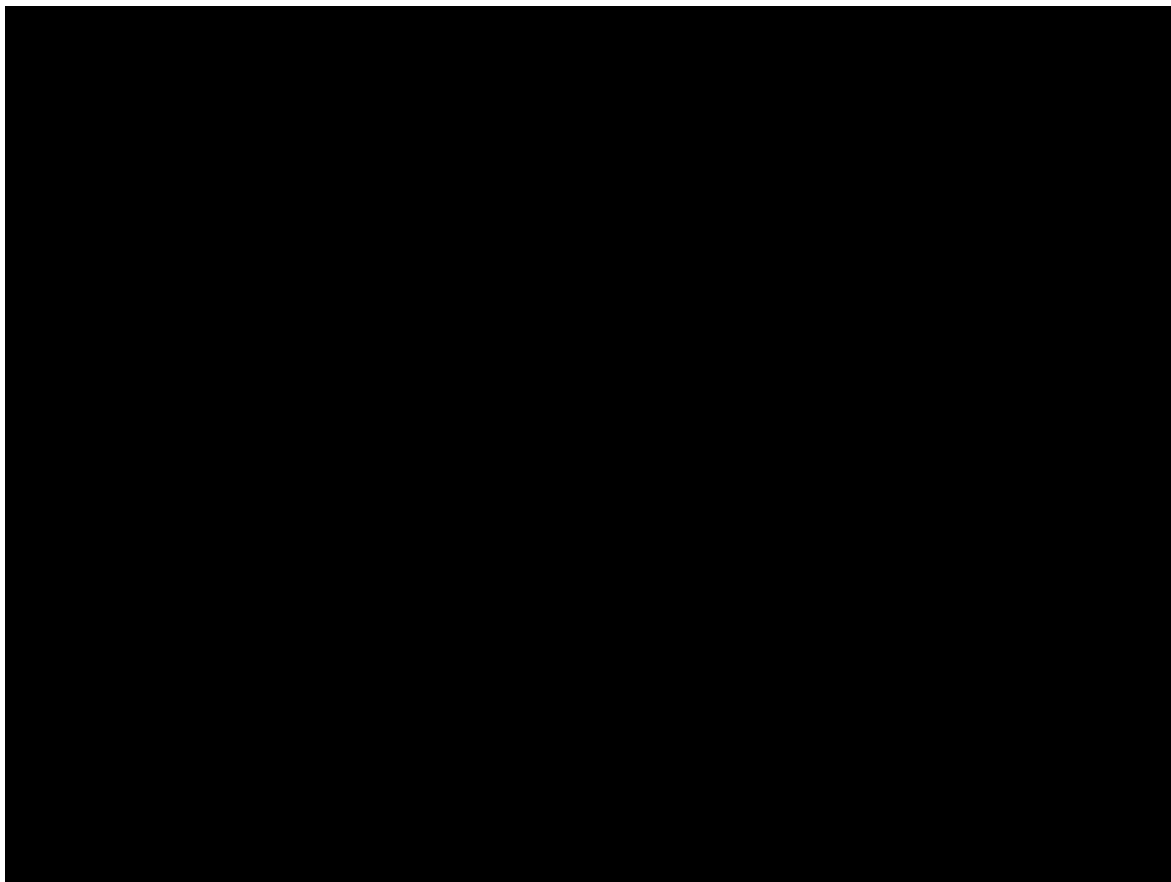
1011011
1101010
0110011
1010110



·AI·

0100101
0010110
1101001
0110110

强化学习训练打砖块游戏AI的效果



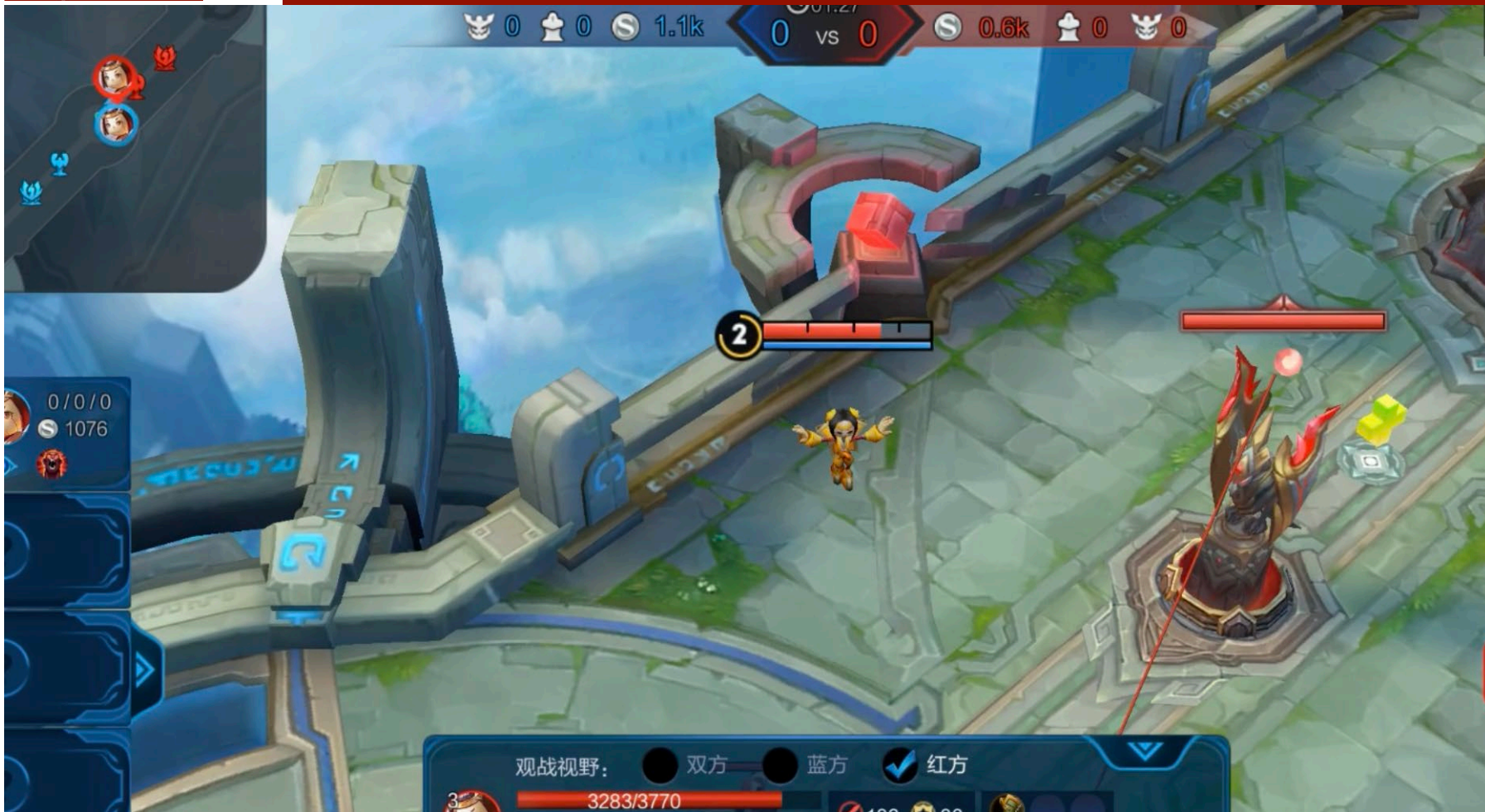
1011011
1101010
0110011
1010110



·AI·

0100101
0010110
1101001
0110110

强化学习训练鲁班七号AI的效果



1. 强化学习的例子

- 打砖块AI、王者鲁班七号AI

2. 强化学习的生物学基础

- 强化理论－斯金纳箱

3. 强化学习的问题模型和求解过程

- 问题模型（环境、智能体、策略 π ）、求解过程（小麻雀－井字棋）
- 问题模型的泛化（带概率的 P, R, π ）、状态价值 V_π 和动作价值 q_π

4. 寻找最优策略的几种思路

- 多臂老虎机、动作价值、贪心、 ϵ -贪心、UCB、梯度下降、方法比较



强化理论 斯金纳箱

斯金纳的操作性条件反射-强化理论 (Skinner Operant Conditioning- Rewards & Punishments)



+ 关注



在其他的東西中間，箱子還有一個杠桿，當壓下杠桿時會釋放食物。

• 强化理论

- 通过奖励和惩罚的方式可以改变智能体的行为方式
- 随机奖励可以使智能体上瘾



1. 强化学习的例子

- 打砖块AI、王者鲁班七号AI

2. 强化学习的生物学基础

- 强化理论 - 斯金纳箱

3. 强化学习的问题模型和求解过程

- 问题模型（环境、智能体、策略 π ）、求解过程（小麻雀 - 井字棋）
- 问题模型的泛化（带概率的 P, R, π ）、状态价值 V_π 和动作价值 q_π

4. 寻找最优策略的几种思路

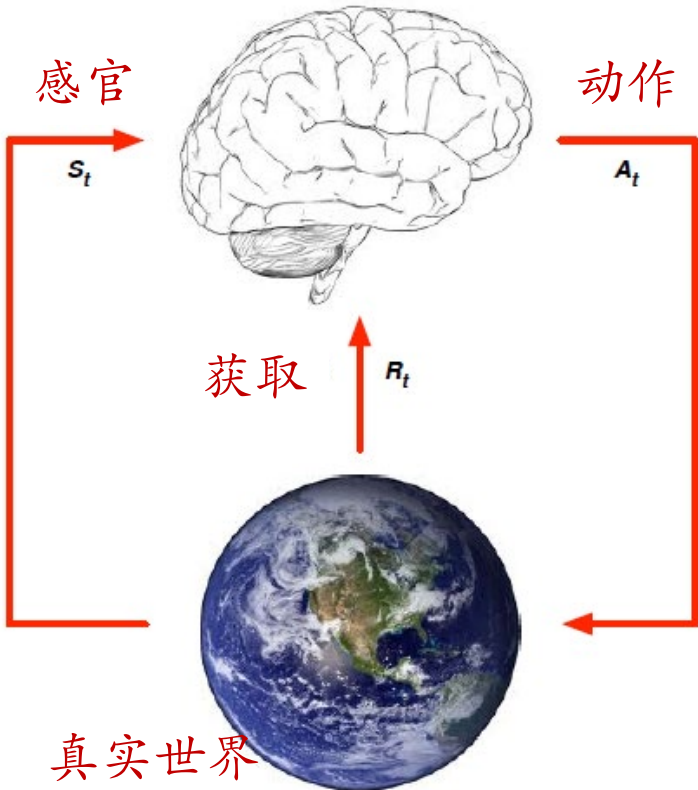
- 多臂老虎机、动作价值、贪心、 ϵ -贪心、UCB、梯度下降、方法比较



强化学习的模型

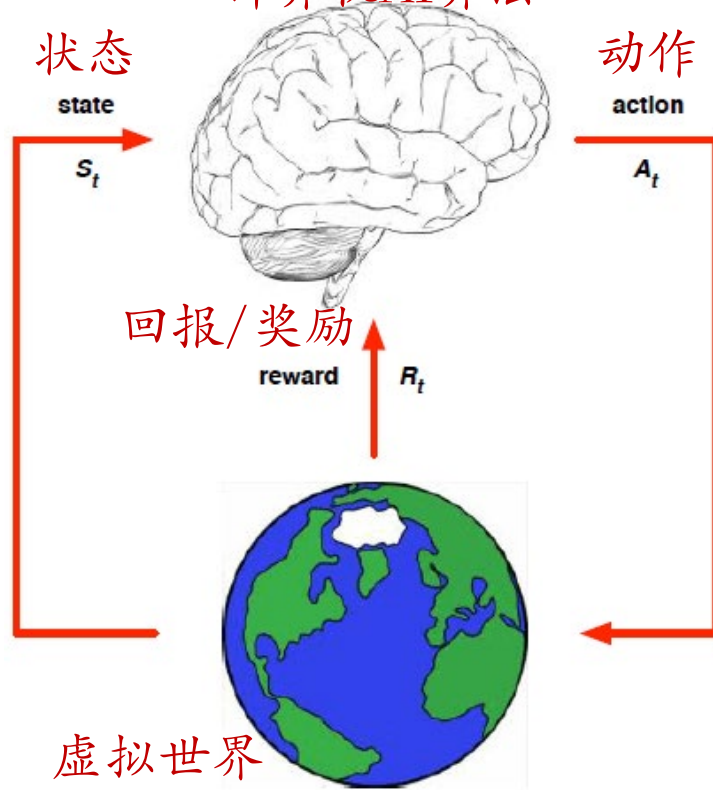
• 直接和真实世界交互

人类



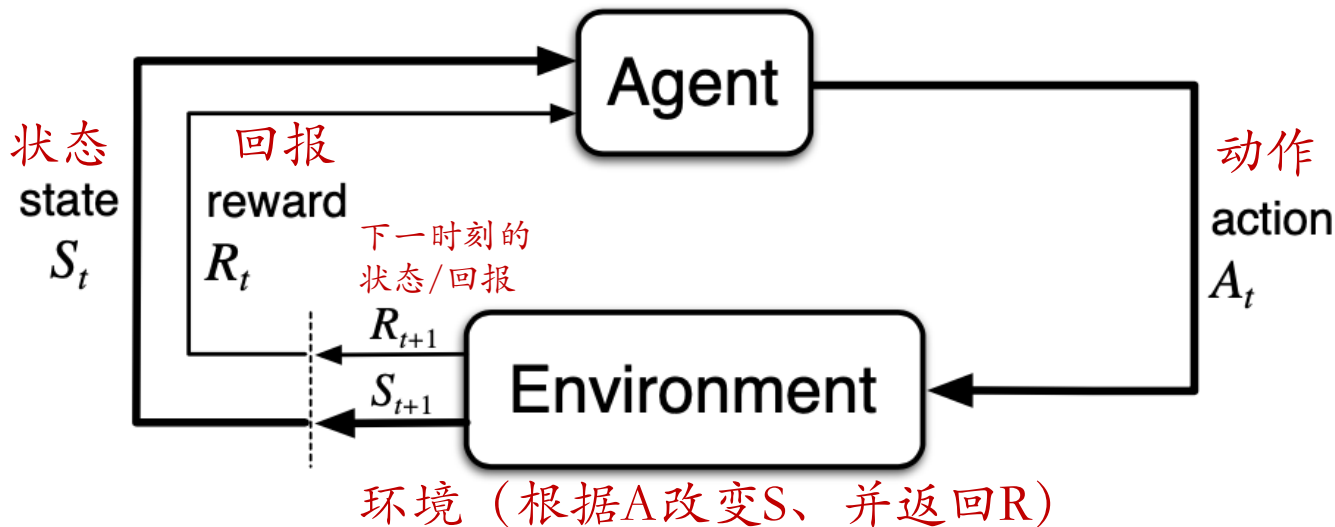
• 和世界的虚拟模型交互

计算机AI算法



强化学习的模型

智能体（输入S输出A，学习获得更好的R反馈）



- 环境（Environment）- 智能体与之交互的对象，描述了问题模型
- 智能体（Agent） - 学习者和决策者



环境（问题模型）

1. 初始状态 S_0 (state)
2. 当前玩家 C (current player(s))
3. 动作 A (action)
 - 智能体在某个状态下的合法动作集合。
4. 状态转移 P (transition) $P(S_{t+1} | S_t, A_t)$ 用以表示环境
 - 衡量一个环境的**复杂程度**：某个状态下，智能体采取某个动作后，转移到下一状态的状态转移模型。可能到达的所有状态构成了**状态空间** (state space)。所有状态下可行动作，构成**动作空间** (action space)。
5. 终止状态 S_T (terminate state)
6. 奖励 R (reward) $R_t \leftarrow S_t, A_t$
 - 某个状态下，智能体采取某个动作后得到的分数。

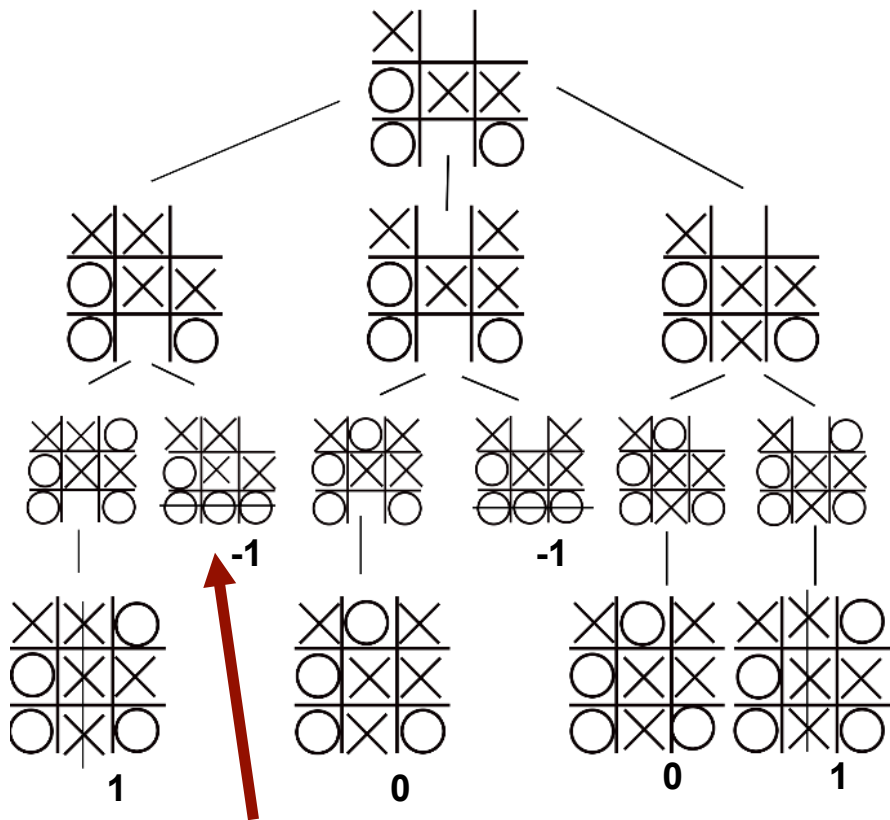


智能体（问题的解）

- 策略 π : $A_t \leftarrow \pi(S_t)$ 用以表示智能体
 - 状态 S 到动作 A 的映射关系，给出了智能体在状态 S 下如何选择动作 A 的决策方法。
 - 注意：策略 π 是全局性的，任何状态下都要能够给出动作选择
 - 目标（问题的解）：
 - 寻找最优策略 π ，使得从初始状态 S_0 到终止状态 S_T 的累积收益
- $$G(\text{gain}) = \sum_{i=1}^T R_i \text{ 最大}$$



井字棋



假设:

给定S有唯一A

敌人用的是确定性策略(未必最优)

我们可以和敌人下很多次

基本思想:

在和敌人的对弈中逐步得到一个

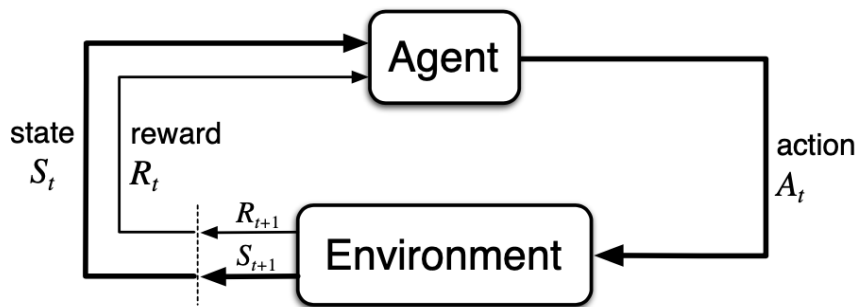
好的对敌策略

如果发现敌人不会下这里，就可以选左边获胜



井字棋问题建模

1. 初始状态 s_0 : 空棋盘
2. 当前玩家: 轮到下子的一方 (也可以把对手建模在环境里, 每次状态转移返回的状态是对手已经落子后的状态, 这样游戏就是单人游戏)
3. 动作 A : 落子到当前为空的位置
4. 状态转移 P : 落子之后的棋盘状态
5. 终止状态 s_T : 棋盘满或者一方获胜
6. 奖励 R : 终止状态, 胜者+1, 负者-1, 战平双方均为0; 其它状态为0



还没到终止状态的状态
下棋过程中

井字棋问题的解

策略 π :

- 使用状态估值表，每个状态一个入口，记录从该状态出发下到终局的胜率；根据状态估值表选择动作。

学习前

0	0	0	0
0	0	0	4

学习

0	起1	2	3
1	2	3	终4

学习策略 π_1 :

- 大概率选择估值最高的下一个状态，小概率随机选一个动作（探索）

目标策略 π^* :

- 每次选择通往估值最高的下一个状态（贪心）

目标（问题的解） :

- 最优策略 π^* ，使得智能体从初始状态 S_0 下到最终的效率/胜率最大

强化学习要回答：
如何学习状态估值表？

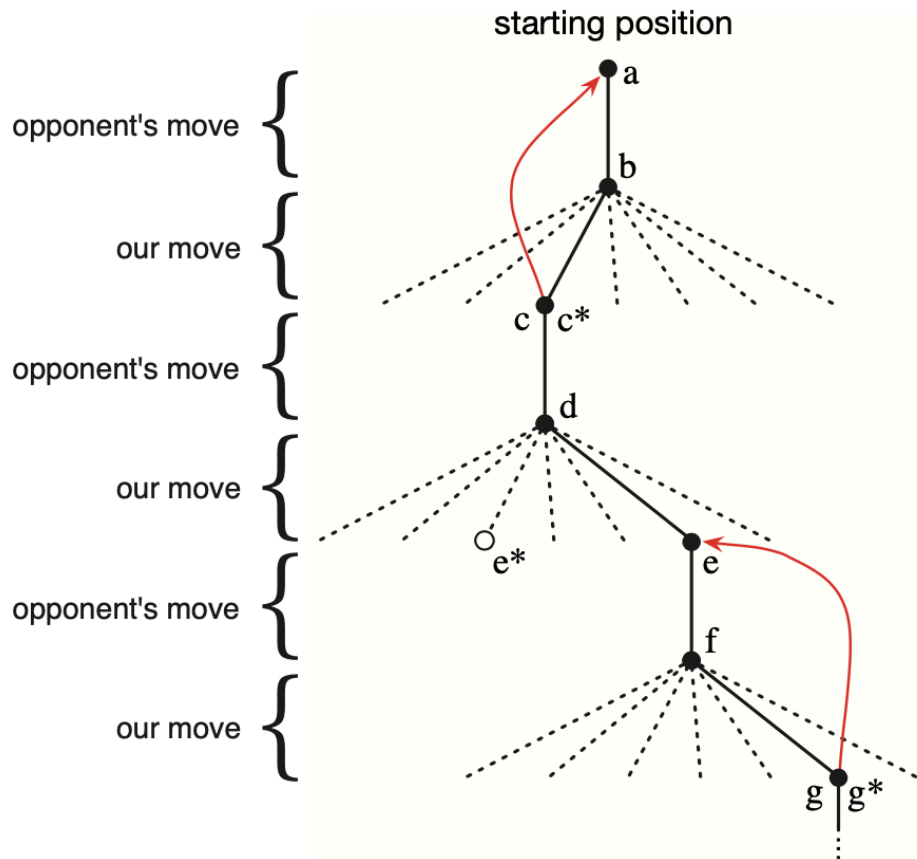
第一步：：建立值函数表

- 第一步：建立状态估值表（值函数表）
- 对于井字棋而言，因为状态数少（状态空间小），可用表格存下估值
- 每个状态一个估值，估值表示这个状态到最终的胜率。
- 整个表是值函数。
- 初值：（根据游戏规则）
 - 三个X连成一线的状态，价值为1，因为我们已经赢了。
 - 三个O连成一线的状态，价值为0，因为我们已经输了。
 - 其他状态的值都为0.5，表示有50%的概率能赢。

状态	估值
...	0.5
...	1
...	0.5
...	0
...	0.5
...	0.5

第二步 和对手玩很多次


- 第二步，和对手玩很多次。
- 利用：大概率贪心选价值最大的地方下；
- 探索：偶尔随机地选择以便探索之前没有探索过的地方；
- 如图所示：
- 值函数表决定了我们的策略，改进值函数表就改进了策略



第三步 边下边修改状态的值

- 第三步，边下边修改状态的值，使得它更接近真实的胜率。

value



$$V(S_t) \leftarrow V(S_t) + \alpha [V(S_{t+1}) - V(S_t)],$$

修正后

未修正

修正值

- 初值：只有终局的价值是正确的，中间局面的价值都是估计值0.5；
- 过程中：状态价值从后面向前传导；
- 分析：假设我们一直在一条路径上反复下，每下一次，终局价值至少向上传导一步，下多了终将把这个终局的输赢带到最上面的初始节点，于是我们在初始节点就会知道最后的输赢；
- α ：是一个小的正的分数，称为步长参数，或者学习率。

不断修正

0.5	0.5	0.5	0.5	1
0.5	0.5	0.5	0.6	1
0.5	0.5	0.6	0.7	1
⋮				
0.5	0.7	0.8	0.9	1



小结

1. 不再假设对手使用最优策略;
2. 将对手建模在环境里; 每次采取动作后面临的状态都是对手执行完它的动作后的新状态; (也可以建模成多智能体博弈问题, 有一个对手决策模型, 轮到对手落子时让对手模型决策)
3. 用值函数表存储状态估值/值函数表 $V(S)$
4. 通过不断对弈更新值函数表
5. 根据值函数表贪心选最优动作 π^*



1. 强化学习的例子
 - 打砖块AI、王者鲁班七号AI
2. 强化学习的生物学基础
 - 强化理论 - 斯金纳箱
3. 强化学习的问题模型和求解过程
 - 问题模型（环境、智能体、策略 π ）、求解过程（小麻雀 - 井字棋）
 - 问题模型的泛化（带概率的 P, R, π ）、状态价值 V_π 和动作价值 q_π
4. 寻找最优策略的几种思路
 - 多臂老虎机、动作价值、贪心、 ϵ -贪心、UCB、梯度下降、方法比较



问题模型的泛化和分析

环境（问题模型）

1. 初始状态 S_0 (state)
2. 当前玩家 C (current player(s))
3. 动作 A (action)
4. 状态转移 P (transition)
5. 终止状态 S_T (terminate state)
6. 奖励 R (reward)

智能体

1. 策略 π
2. 目标（问题的解）

最大化期望累积收益 G



环境：状态转移模型 P 和奖励 R

- 状态转移 不一定是确定性的，可以按 **概率状态转移**
 - **P**：状态转移函数 $\langle S, A, S \rangle \rightarrow \mathcal{R}^+$, $P(s, a, s') = \Pr[s' | s, a]$, s, a 是当前状态和动作, s' 是下一状态。
 - 对于任意 s, a , 有 $\sum_{s'} P(s, a, s') = 1$ 给定 S 所有可行动作 A 下的状态概率之和为 1
- 奖励也不一定是确定性的，可以是一个 **概率奖励**
 - **R**：奖励函数 $\langle S, A, \mathcal{R}^+ \rangle \rightarrow \mathcal{R}^+$, $R(s, a, r) = \Pr[r | s, a]$, s, a 是当前状态和动作, r 是奖励。
 - 对于任意 s, a , $\sum_r R(s, a, r) = 1$ 给定 S 和 A 时所有可能奖励的概率之和为 1



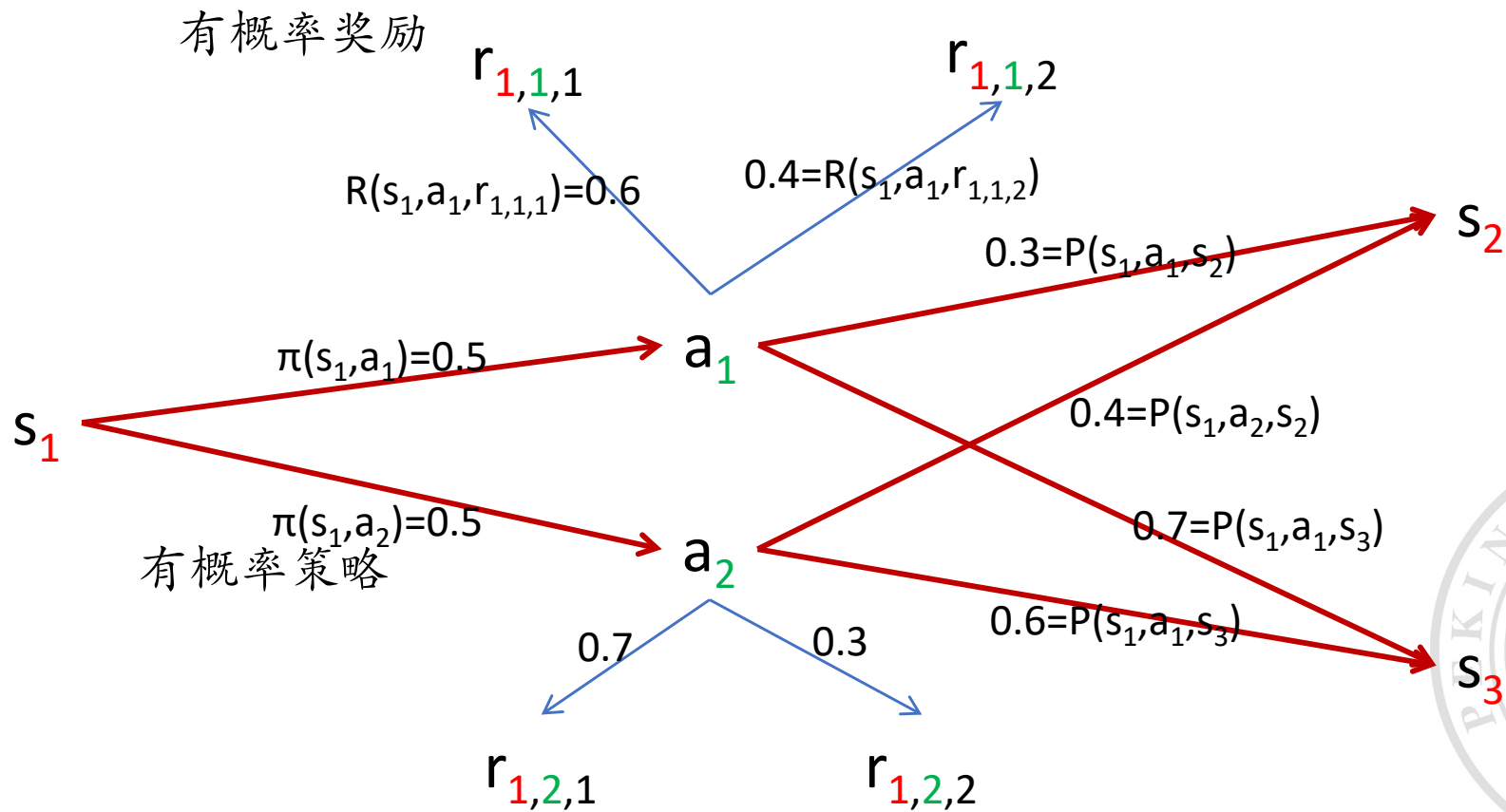
智能体：策略 π 和累积收益 G

- 策略 π 给出的动作选择可以是确定的，也可以是一个概率分布
- π : 策略函数 $\langle S, \mathcal{A} \rangle \rightarrow \mathcal{R}^+$, π 描述状态 s 下采取动作 a 的概率。
 $\pi(s, a) = \text{Pr}[a|s]$, s 是当前状态, a 是当前状态下的可选动作。
- 对于任意 s , $\sum_a \pi(s, a) = 1$

若为1, 则近的收益和远的收益一样重要
若为0, 则只看下一步的收益 (最贪心)
- 折扣因子 γ : $(0 \leq \gamma \leq 1)$, 描述未来收益的重要程度
- 累积收益 $G = R_1 + \gamma R_2 + \gamma^2 R_3 + \dots = \sum_{i=1}^T \gamma^{i-1} R_i$

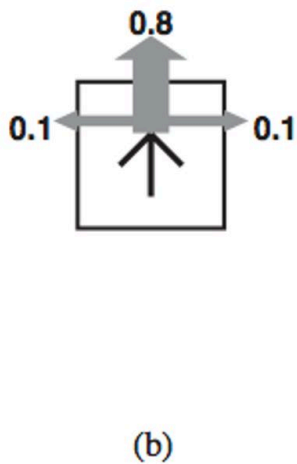
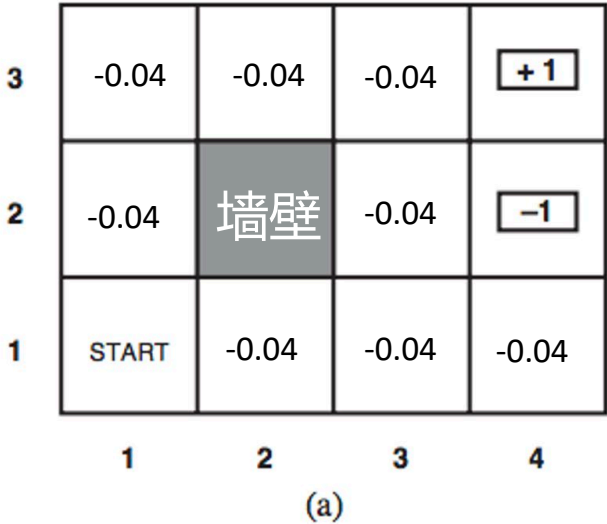
(T 可以是有限的也可以是无限制的)
- 描述对于某个 $s_1, a_1, s_2, a_2, s_3, a_3, \dots$ 状态动作序列的累积收益。

状态、动作、状态转移和奖励示意图



奖励函数对智能体最优策略的影响

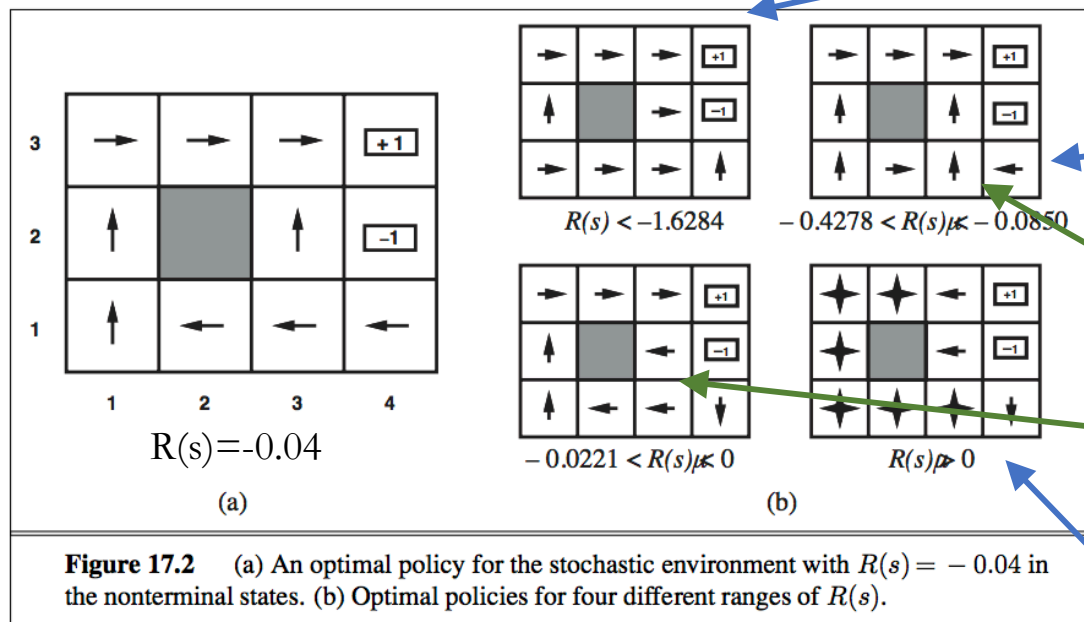
4*3 网格世界 环境



- 1. 初始状态：START
- 2. 当前玩家：单个玩家
- 3. 合法动作：上下左右（不撞墙不出界）
- 4. 状态转移：依概率进行，0.8动作成功，0.2动作不成功，偏左右各0.1概率
- 5. 终止状态：右上角，有两个
- 6. 奖励函数：到达+1得+1，到达-1得-1，其他状态-0.04

奖励函数对智能体最优策略的影响

4*3 网格世界 智能体最优策略:



如果其他状态R不是-0.04，会怎么样？

- $R(s) \leq -1.6284$, 生活如此痛苦, 直接奔向最近的出口即使奖励是 -1 .
- $-0.4278 \leq R(s) \leq -0.0850$, 生活不愉快, 走最短路去+1 出口, 甘冒偶尔落入-1出口的风险
- $-0.0221 < R(s) < 0$, 最优策略完全不冒险
- $R(s) > 0$, 生活是如此惬意, 躲避出口, 不出去

一个策略就是一张状态到动作的映射表

策略的评估和最优策略

策略 π 的好坏用 **状态价值 V_π** 来评估：

- **状态价值 $V_\pi(s)$** 表示从 s 出发执行策略 π 能获得的**累积收益**；
- **结束状态**（如果有）的价值，总是零

$$v_\pi(s) \doteq \underset{\text{期望}}{\mathbb{E}_\pi[G_t \mid S_t = s]} = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right], \text{ for all } s \in \mathcal{S},$$

显然从同一个状态 s 出发， V_π 越大， π 越好；使得 V 最大的 π 就是最优策略，记作 π^* ，执行 π^* 得到的价值，就是最优价值，记作 V^* 。

状态价值 V_π 和动作价值 Q_π

状态价值 $V_\pi(s)$ 表示从 s 出发执行策略 π 能获得的累积收益

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right], \text{ for all } s \in \mathcal{S},$$

动作价值 $Q_\pi(s, a)$ 表示从 s 出发并做动作 a ，之后执行策略 π 能获得的累积收益；有些时候计算动作价值更方便

$$Q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right].$$

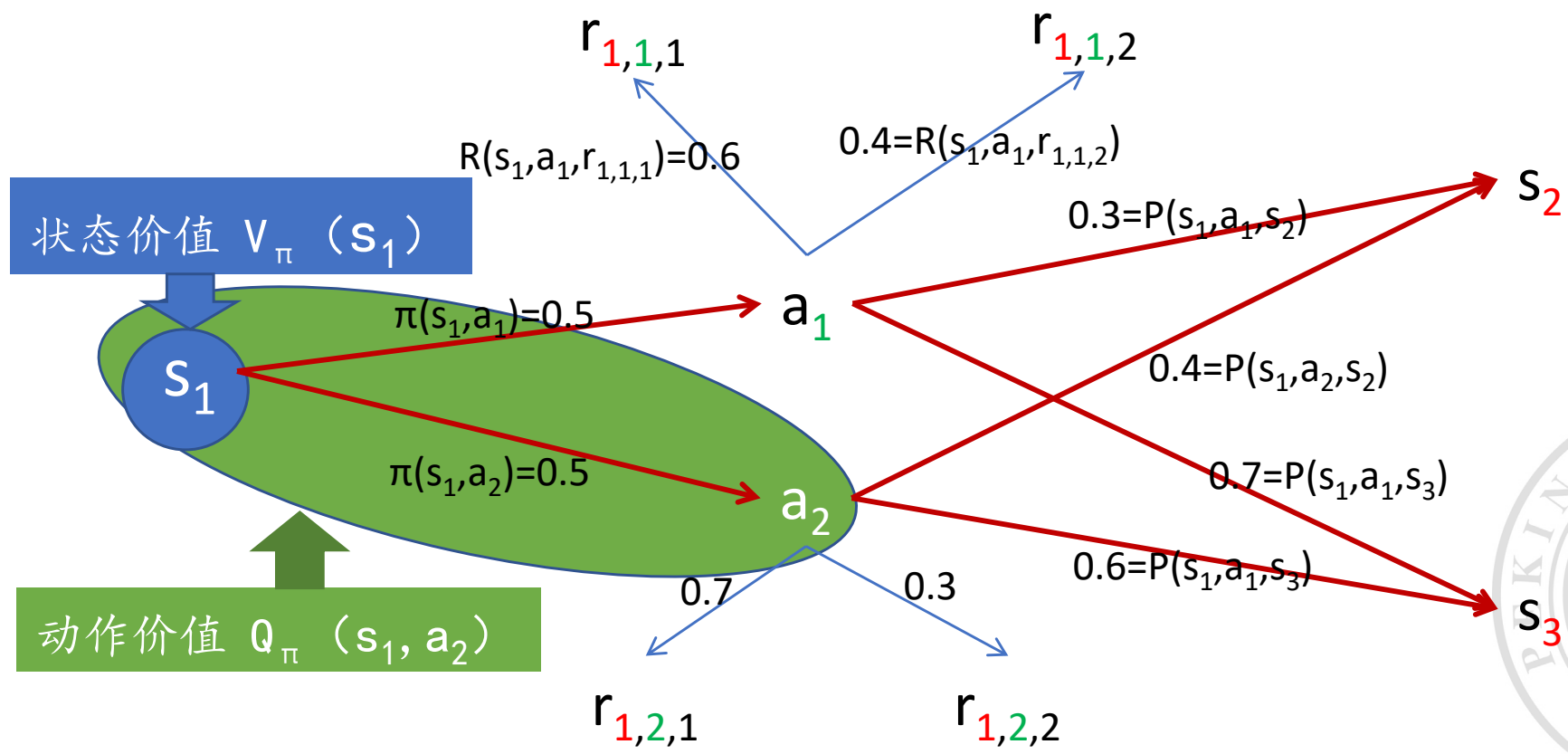
1011011
1101010
0110011
1010110



·AI·

0100101
0010110
1101001
0110110

状态价值 V_π 和动作价值 q_π



练习： V_π 和 Q_π 的关系

- Exercise 3.12 Give an equation for V_π in terms of Q_π and π .

Answer: s 的状态价值 = s 下所有可行 a 的概率 \times 其动作价值之和

$V \leftarrow Q$

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) Q_\pi(s, a)$$

- Exercise 3.13 Give an equation for Q_π in terms of V_π and the four-argument p .

Answer:

$Q \leftarrow V$

$$Q_\pi(s, a) = \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | s, a) [r + \gamma v_\pi(s')]$$

s' 下一时刻的状态
 r 一步的及时奖励



强化学习的任务

- 找到最优策略 π^* ，而得到 V^* 或者 Q^* 就能得到最优策略 π^*
- 算出来
 - 使用各种方法探索出 V^* 或者 Q^*
- 存起来
 - 状态多时，查找表保存所有状态的价值不现实
 - 用带参数的函数来保存 $V_\pi(s)$ 和 $Q_\pi(s,a)$ (参数数目小于状态数)
 - 学习过程中我们会调整参数，使之更符合观察到的实际收益。
 - 学习效果取决于带参数的近似函数的好坏。
- 后面也会讲到直接用函数模拟策略 $\pi(a|s)$ 的方法

智能体寻找最优策略的路径

- 智能体使用策略 π_0 （开始可能是随机的）与环境交互，产生经验（Experience）
 - $S_{01}, a_{01}, r_{11}, S_{11}, a_{11}, r_{21}, S_{21}, a_{21}, r_{31}, S_{31}, a_{31}, \dots$ 计算 $G_1 \sim \pi_0$
 - $S_{02}, a_{02}, r_{12}, S_{12}, a_{12}, r_{22}, S_{22}, a_{22}, r_{32}, S_{32}, a_{32}, \dots$ 计算 $G_2 \sim \pi_0$
 - $S_{03}, a_{03}, r_{13}, S_{13}, a_{13}, r_{23}, S_{23}, a_{23}, r_{33}, S_{33}, a_{33}, \dots$ 计算 $G_3 \sim \pi_0$
- 智能体根据经验改进 π_0 得到 π_1 ，再用 π_1 与环境交互以期获得更大的G
 - $S_{01}, a_{01}, r_{11}, S_{11}, a_{11}, r_{21}, S_{21}, a_{21}, r_{31}, S_{31}, a_{31}, \dots$ 计算 $G_1 \sim \pi_1$
 - $S_{02}, a_{02}, r_{12}, S_{12}, a_{12}, r_{22}, S_{22}, a_{22}, r_{32}, S_{32}, a_{32}, \dots$ 计算 $G_2 \sim \pi_1$
 - $S_{03}, a_{03}, r_{13}, S_{13}, a_{13}, r_{23}, S_{23}, a_{23}, r_{33}, S_{33}, a_{33}, \dots$ 计算 $G_3 \sim \pi_1$
- ……如此往复，直到得到最（更）好的 π
- 前面看到的打砖块AI和王者鲁班七号AI都是这么得来的



1. 强化学习的例子

- 打砖块AI、王者鲁班七号AI

2. 强化学习的生物学基础

- 强化理论－斯金纳箱

3. 强化学习的问题模型和求解过程

- 问题模型（环境、智能体、策略 π ）、求解过程（小麻雀－井字棋）
- 问题模型的泛化（带概率的 P, R, π ）、状态价值 V_π 和动作价值 Q_π

4. 寻找最优策略的几种思路

- 多臂老虎机、动作价值、贪心、 ϵ -贪心、UCB、梯度下降、方法比较



寻找最优策略的几种思路

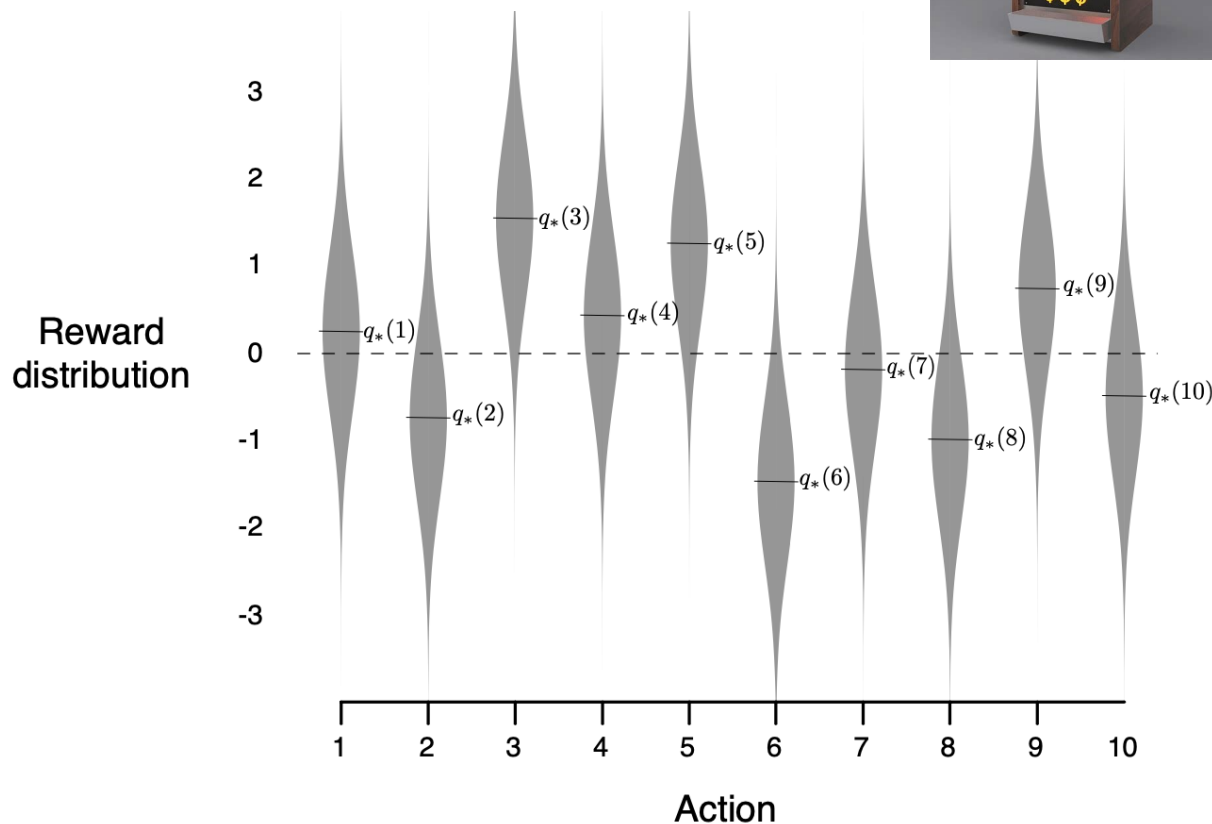
1. 多臂老虎机问题
2. 计算动作价值
3. 贪心 vs ϵ - 贪心
4. ϵ - 贪心 vs 乐观初值贪心
5. Upper-Confidence-Bound (UCB)
6. Gradient Bandit Algorithms 梯度下降
7. 算法比较



多臂老虎机问题



- k 个可行动作
(选择第 i 号老虎机)
- 每次选择一个动作, 获得一个数字化的奖励
- 每个动作的奖励服从一个确定的正态分布
- 目标是最大化一段时间的总收益的期望, 例如 1000 个金币的总收益.



求解最优策略的几种思路

1. 多臂老虎机问题
2. 计算动作价值 $q(s,a)$
3. 贪心 vs ϵ -贪心
4. ϵ -贪心 vs 乐观初值贪心
5. Upper-Confidence-Bound (UCB)
6. Gradient Bandit Algorithms 梯度下降
7. 算法比较



计算动作价值

- 没摇过的有个缺省值，例如：0
- 时间 t 时选的动作作为 A_t , 对应的奖励为 R_t .
- 动作 a 的价值为 $Q^*(a) \approx E[R_t \mid A_t = a]$.

老虎机问题中，状态 s 是不变的，不考虑 s 了

说白了就是求平均值

$$Q_t(a) = \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}},$$

- 大数定理，无限次后, $Q_t(a)$ 收敛至 $Q^*(a)$.

增量计算动作价值

$$\begin{aligned}
 Q_{n+1} &= \frac{1}{n} \sum_{i=1}^n R_i \\
 &= \frac{1}{n} \left(R_n + \sum_{i=1}^{n-1} R_i \right) \\
 &= \frac{1}{n} \left(R_n + (n-1) \frac{1}{n-1} \sum_{i=1}^{n-1} R_i \right) \\
 &= \frac{1}{n} (R_n + (n-1)Q_n) \\
 &= \frac{1}{n} (R_n + nQ_n - Q_n) \\
 &= Q_n + \frac{1}{n} [R_n - Q_n],
 \end{aligned}$$

但不能花完1000个币，才知道Q吧。。太晚了！
能不能玩的同时更新Q？

$$Q_n \doteq \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}.$$

取 $\alpha = 1/n$
 α 随时间变小
 Q_n 趋近于均值



计算动作价值的一般形式

$$\underset{\text{新的估计}}{NewEstimate} \leftarrow \underset{\text{老的估计}}{OldEstimate} + \underset{n}{StepSize} \left[\overset{\text{误差Error}}{Target} - \underset{\text{老的估计}}{OldEstimate} \right].$$

新收益

A simple bandit algorithm

Initialize, for $a = 1$ to k :

$$Q(a) \leftarrow 0$$

$$N(a) \leftarrow 0$$

Loop forever:

贪心利用

$$A \leftarrow \begin{cases} \operatorname{argmax}_a Q(a) & \text{with probability } 1 - \varepsilon \\ \text{a random action} & \text{with probability } \varepsilon \end{cases} \quad \begin{matrix} \text{(breaking ties randomly)} \\ \text{探索} \end{matrix}$$

$$R \leftarrow \text{bandit}(A)$$

$$N(A) \leftarrow N(A) + 1$$

$$Q(A) \leftarrow Q(A) + \frac{1}{N(A)} [R - Q(A)]$$

关于学习率 α 的讨论

$$Q_n + \frac{1}{n} [R_n - Q_n]$$

α 等于 $1/n$
 Q_n 新旧值一样看待

$$\begin{aligned} Q_{n+1} &= Q_n + \alpha [R_n - Q_n] \\ &= \alpha R_n + (1 - \alpha) Q_n \\ &= \alpha R_n + (1 - \alpha) [\alpha R_{n-1} + (1 - \alpha) Q_{n-1}] \\ &= \alpha R_n + (1 - \alpha) \alpha R_{n-1} + (1 - \alpha)^2 Q_{n-1} \\ &= \alpha R_n + (1 - \alpha) \alpha R_{n-1} + (1 - \alpha)^2 \alpha R_{n-2} + \\ &\quad \dots + (1 - \alpha)^{n-1} \alpha R_1 + (1 - \alpha)^n Q_1 \\ &= (1 - \alpha)^n Q_1 + \sum_{i=1}^n \alpha (1 - \alpha)^{n-i} R_i. \end{aligned}$$

α 保持不变 (< 1)
 Q_n 偏向新值



求解最优策略的几种思路

1. 多臂老虎机问题
2. 计算动作价值
3. 贪心 vs ϵ -贪心
4. ϵ -贪心 vs 乐观初值贪心
5. Upper-Confidence-Bound (UCB)
6. Gradient Bandit Algorithms 梯度下降
7. 算法比较



贪心 vs ϵ -贪心

• 方法一：贪心

- 简单选择估值最高的动作 - 贪心动作
- 多个动作估值一样高，则随机选

$$A_t \doteq \underset{a}{\operatorname{argmax}} Q_t(a),$$

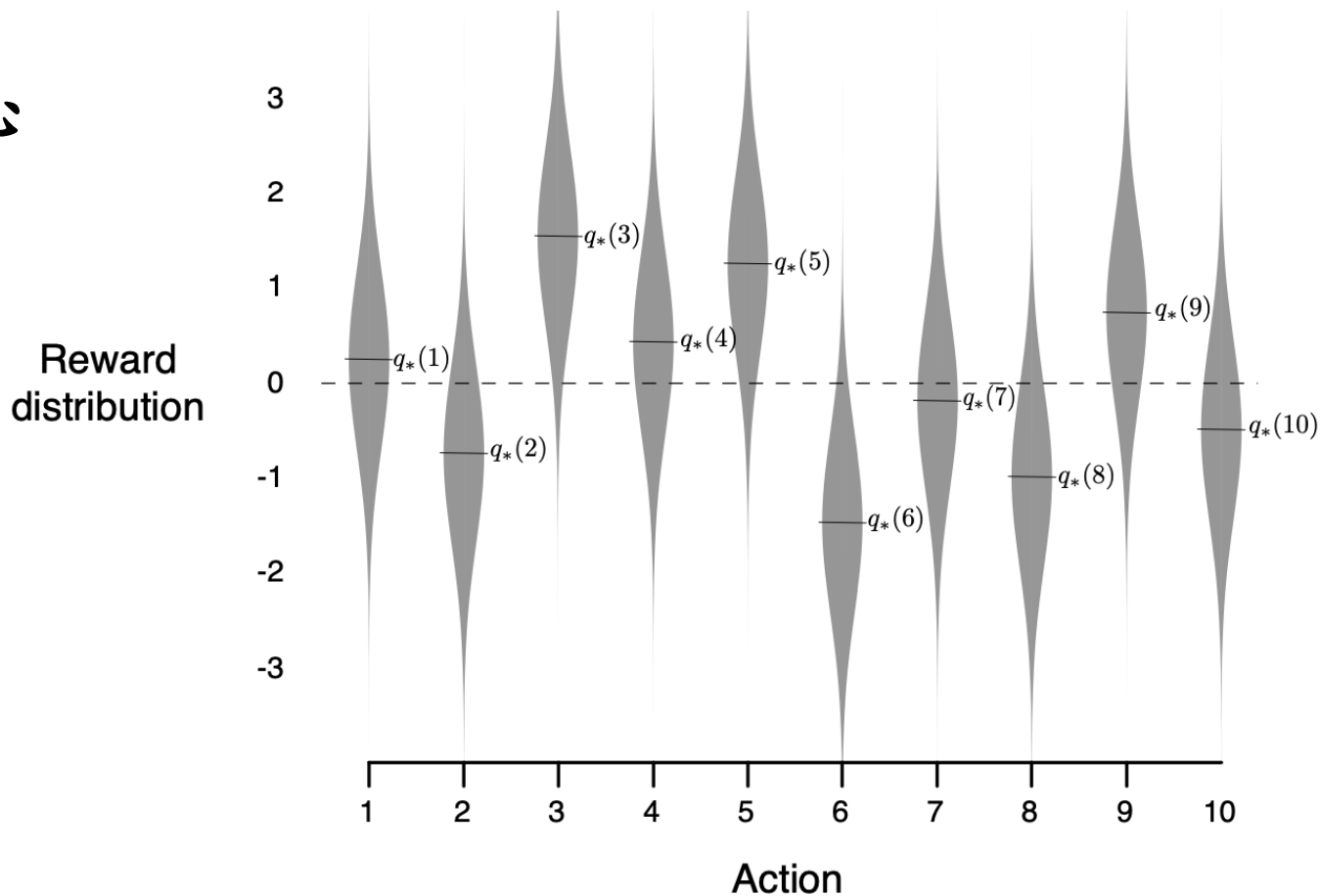
• 方法二： ϵ -贪心 ϵ -greedy

- 大部分时间贪心，偶尔随机选



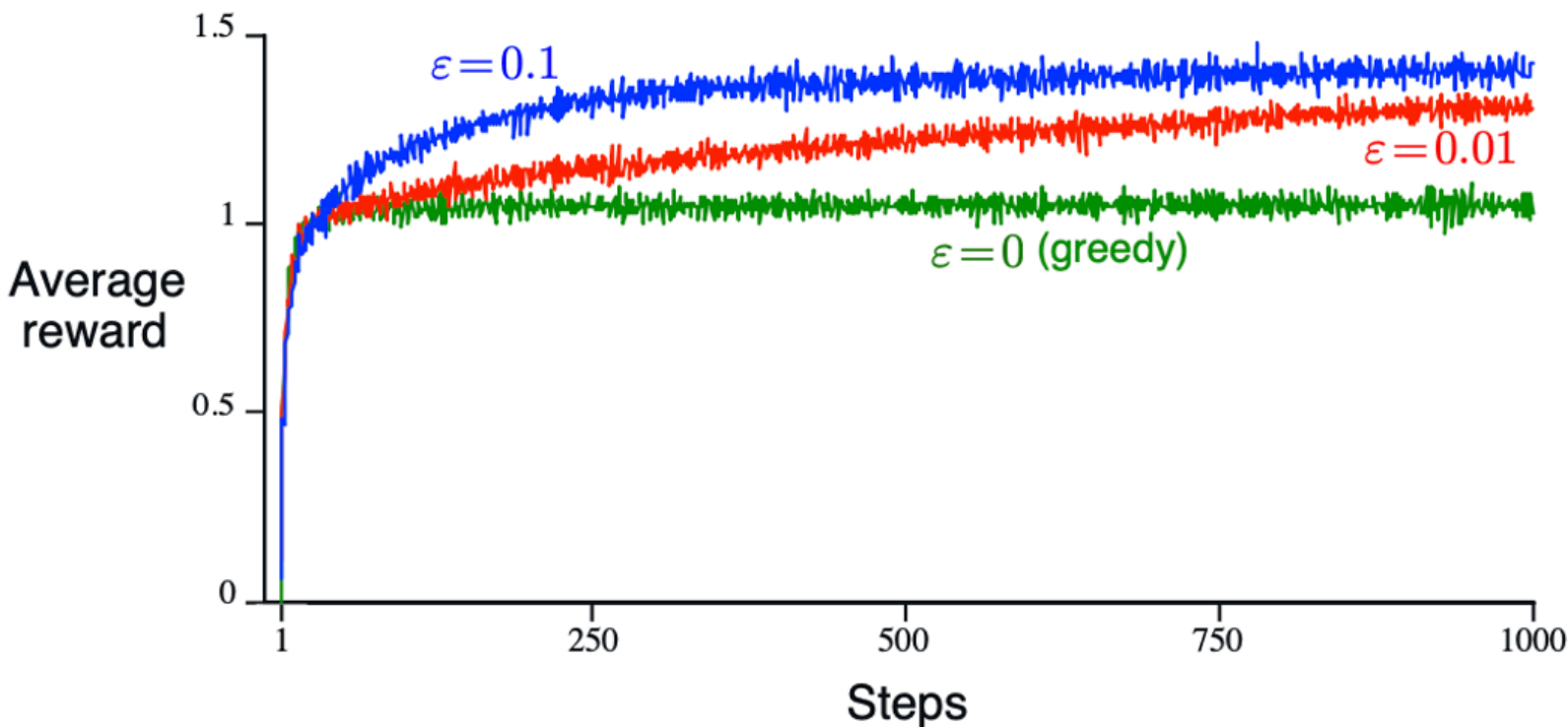
贪心 vs ϵ -贪心

- 比较贪心和 ϵ -贪心的实验，2000次10臂老虎机



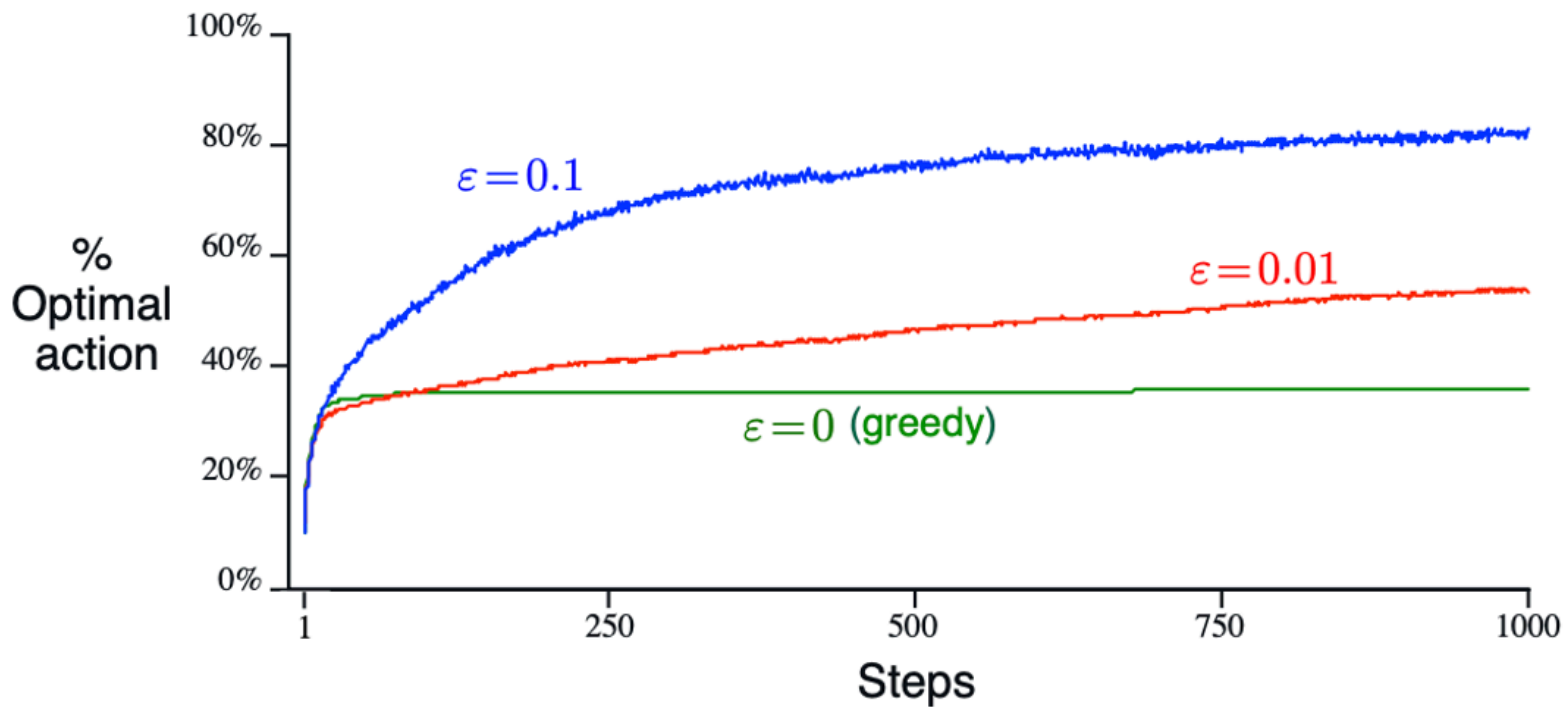
贪心 vs ϵ -贪心

2000 次运行，平均收益， ϵ 取不同的值



贪心 vs ϵ -贪心

2000 次运行，选中最优动作的概率， ϵ 取不同的值



求解最优策略的几种思路

1. 多臂老虎机问题
2. 计算动作价值
3. 贪心 vs ϵ -贪心
4. ϵ -贪心 vs 乐观初值贪心
5. Upper-Confidence-Bound (UCB)
6. Gradient Bandit Algorithms 梯度下降
7. 算法比较

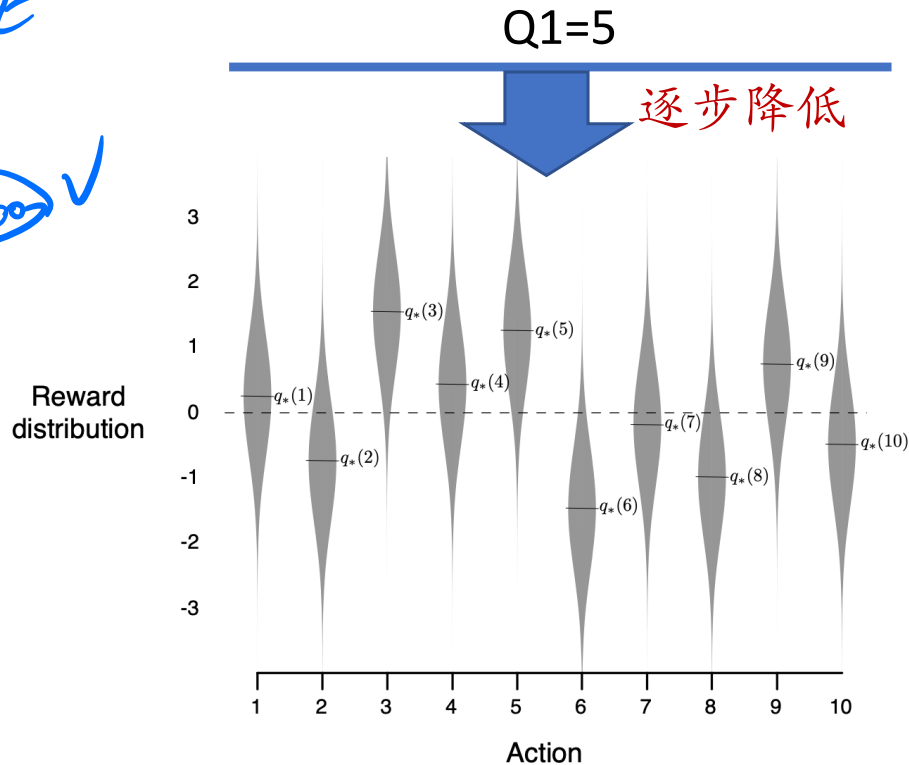


ϵ - 贪心 vs 乐观初值贪心

- 贪心和 ϵ - 贪心都在某种程度上依赖于初值, $Q_1(a)$.
之前都是设为0
- 初值通常不只是问题, 也可以加以利用
- 负面的影响是这些初值会成为一组参数, 必须由用户给出;
- 正面的影响是这提供了一种简单的方式来提供先验经验以确
定期望的奖励在哪个量级。
- 初值估计准确与否会决定需要多少次学习能把偏差纠正回来

ϵ - 贪心 vs 乐观初值贪心

- 初值提供一种简单的方法来鼓励探索。
- 假设我们不把初值设为0而是设为 +5, $Q^*(a)$ 是均值为 0 的, 所以初值为+5 相当乐观
 100
 55 → 10
- 这种乐观会鼓励探索, 不管开始选了哪个动作, 奖励都比初值小, 算法就会尝试其他动作
- 这会导致估值收敛前所有动作都被尝试几次, 系统做了相当的尝试即使每次都使用贪心方法



ϵ - 贪心 vs 乐观初值贪心

讨论：如何解释图上的曲线？

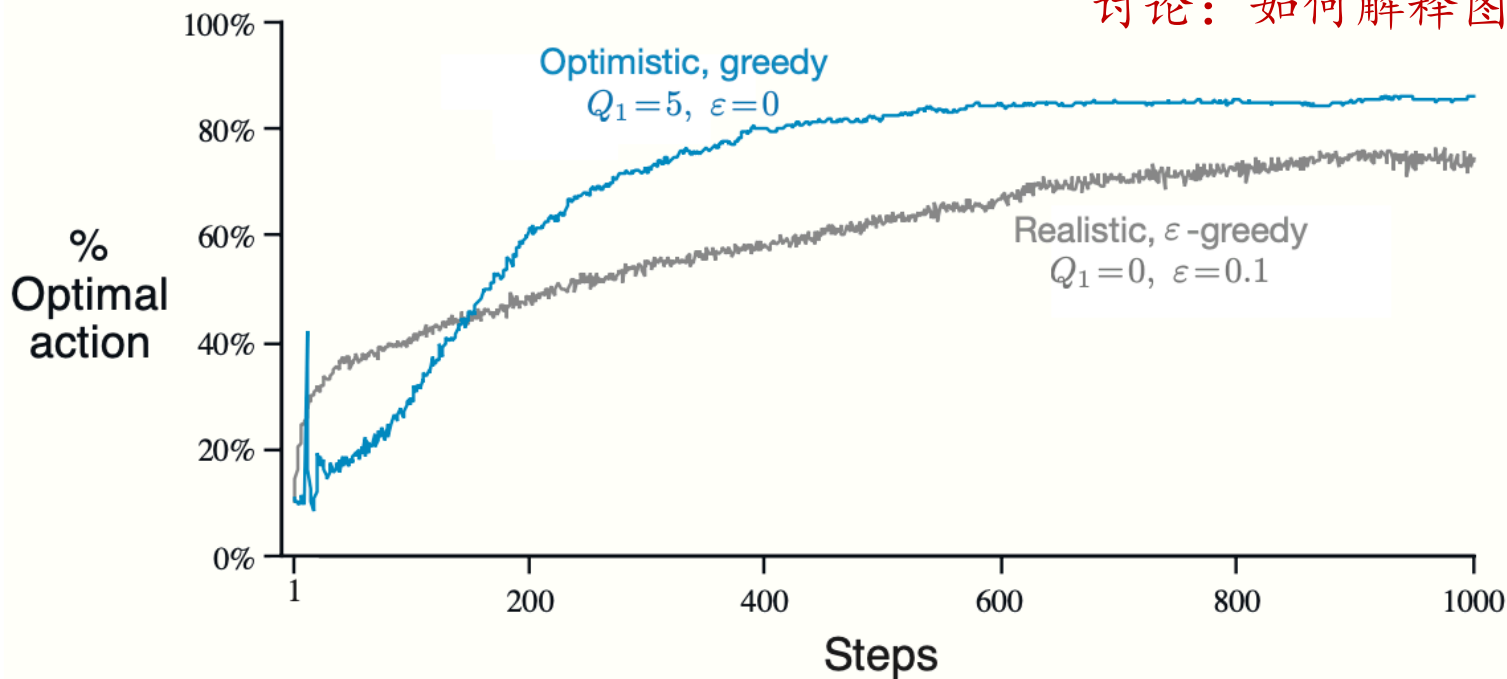
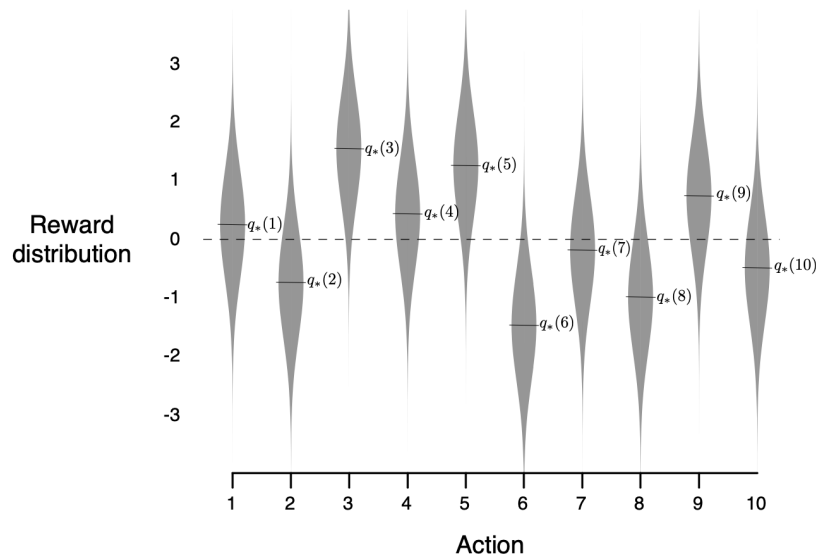


Figure 2.3: The effect of optimistic initial action-value estimates on the 10-armed testbed. Both methods used a constant step-size parameter, $\alpha = 0.1$.

ϵ - 贪心 vs 乐观初值贪心

- 对比 $Q_1(a) = +5$ 和 $Q_1(a) = 0$
- 如同人生，期望高就会勇于尝试新路径！
- 但找到方向的时候，别乱探索了！
- 乐观初值鼓励探索，对于固定问题是非常有效的小技巧，但是并非是普遍有效的鼓励探索的方法
- 所以，如果时代变了，你还是要探索的！
- 例如，对于非固定问题不管用，因为它鼓励探索是临时性的，为什么？



求解最优策略的几种思路

1. 多臂老虎机问题
2. 计算动作价值
3. 贪心 vs ϵ -贪心
4. ϵ -贪心 vs 乐观初值贪心
5. Upper-Confidence-Bound (UCB)
6. Gradient Bandit Algorithms 梯度下降
7. 算法比较



Upper-Confidence-Bound (UCB)

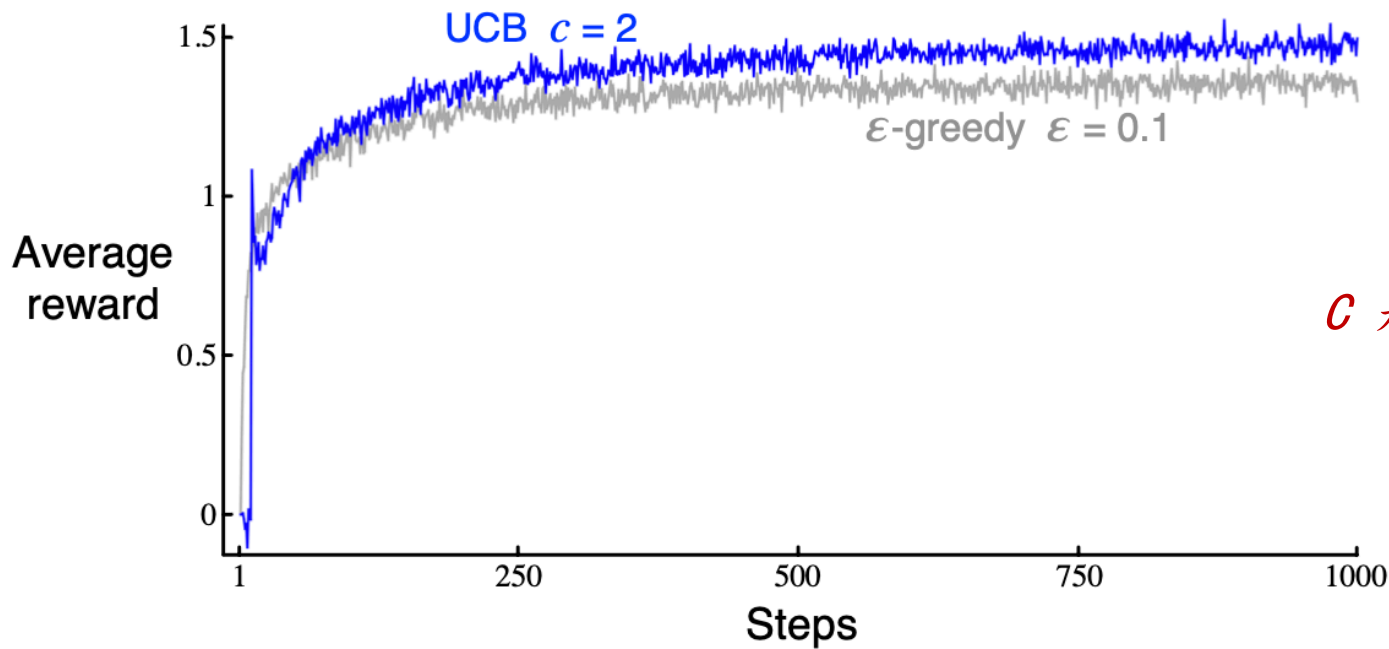
- 不确定性总是存在，所以需要探索
- 贪心只能选目前看似好的动作，但是其他动作可能会更好
- 朴素的想法是估值时，将“当前估值”和“新鲜程度”，取个加权和

$$A_t \doteq \operatorname{argmax}_a \left[\underbrace{Q_t(a)}_{\text{估值}} + c \sqrt{\frac{\ln t}{\underbrace{N_t(a)}_{\text{新鲜程度 (越新鲜越探索)}}}} \right],$$

- $N_t(a)$: a 在时间 t 前被选中的次数， $c > 0$ 控制探索的度。
- 如果 $N_t(a) = 0$, a 被认为是一个取值最大的动作



Upper-Confidence-Bound (UCB)



c 是一个经验值

Figure 2.4: Average performance of UCB action selection on the 10-armed testbed. As shown, UCB generally performs better than ϵ -greedy action selection, except in the first k steps, when it selects randomly among the as-yet-untried actions.



求解最优策略的几种思路

1. 多臂老虎机问题
2. 计算动作价值
3. 贪心 vs ϵ -贪心
4. ϵ -贪心 vs 乐观初值贪心
5. Upper-Confidence-Bound (UCB)
6. Gradient Bandit Algorithms 梯度下降
7. 算法比较



Gradient Bandit Algorithms 梯度下降

- 前面我们用估值选动作。这是个好办法，但不是唯一的办法。
- ϵ -贪心大概率选最好的动作，其他动作等同对待，其实也可以给每个动作一个概率
- 这一节我们考虑给每个动作 a 一个数值的优先度, $H_t(a) \in \mathbb{R}$.
- 不同动作有不同的优先度，会影响它们被选择的概率
- 优先度越大，动作选中概率越大，优先度大表示一个动作比另一更优先；
- 如果我们给每个动作加1000，不会改变动作选择的优先顺序，



Gradient Bandit Algorithms 梯度下降

- soft-max distribution (i.e., Gibbs or Boltzmann distribution) 如下:

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^k e^{H_t(b)}} \doteq \pi_t(a),$$

Softmax函数, 所有可行动作A的概率和为1

- 这里我们引入, $\pi_t(a)$, 表示在时间t选择动作a的可能性.
- 最开始所有动作的倾向性都相同 (例如, $H_1(a) = 0$, 对于所有a) 所以所有动作被选中的概率相同.



Gradient Bandit Algorithms 梯度下降

- 在每一步, 按概率选择了动作 A_t 后得到及时奖励 R_t , 我们可以根据 R_t 的大小, 修改所有动作的优先度

- R_t 比均值 \bar{R}_t 大表明选对了, 这个动作优先级提升:

如果 $R_t > \bar{R}_t$ 这项为正

提升当前 A_t 的H

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t)), \quad \text{and}$$

$$H_{t+1}(a) \doteq H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a), \quad \text{for all } a \neq A_t,$$

降低其他 a 的H

- 其中 $\alpha > 0$, 是学习率



Gradient Bandit Algorithms 梯度下降

- 这里有个作为比较参照值的R平均
- 如果比R平均大，就说明这个动作好，给H (A_t) 增加，
- 同时给其他动作的优选值减少，增加和减少的一样，相当于总量不变的情况下抬升好的，降低差的动作
- R_t比平均小，就减小H (A_t) ，增加别的动作的H。
- 所有π的和为1，推一下增加和减少的量相同 - 第一式一项，第二式多项 - 合起来概率与第一项相同

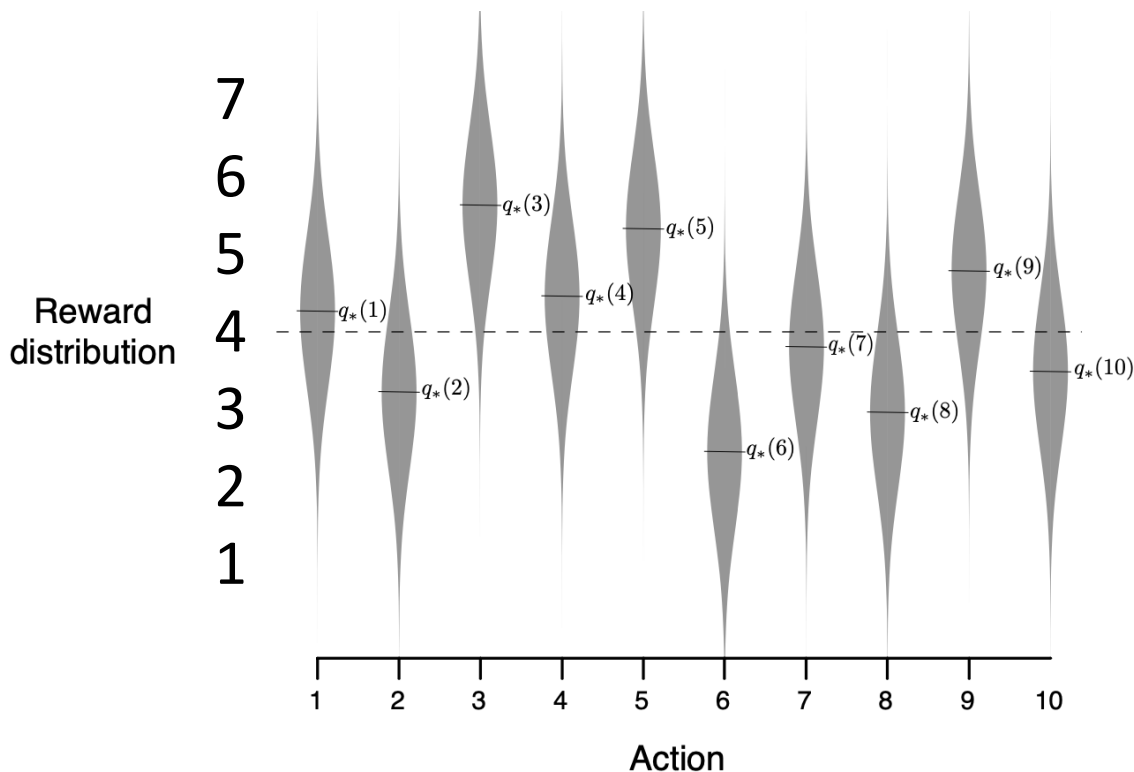
$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t)), \quad \text{and}$$

$$H_{t+1}(a) \doteq H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a), \quad \text{for all } a \neq A_t,$$



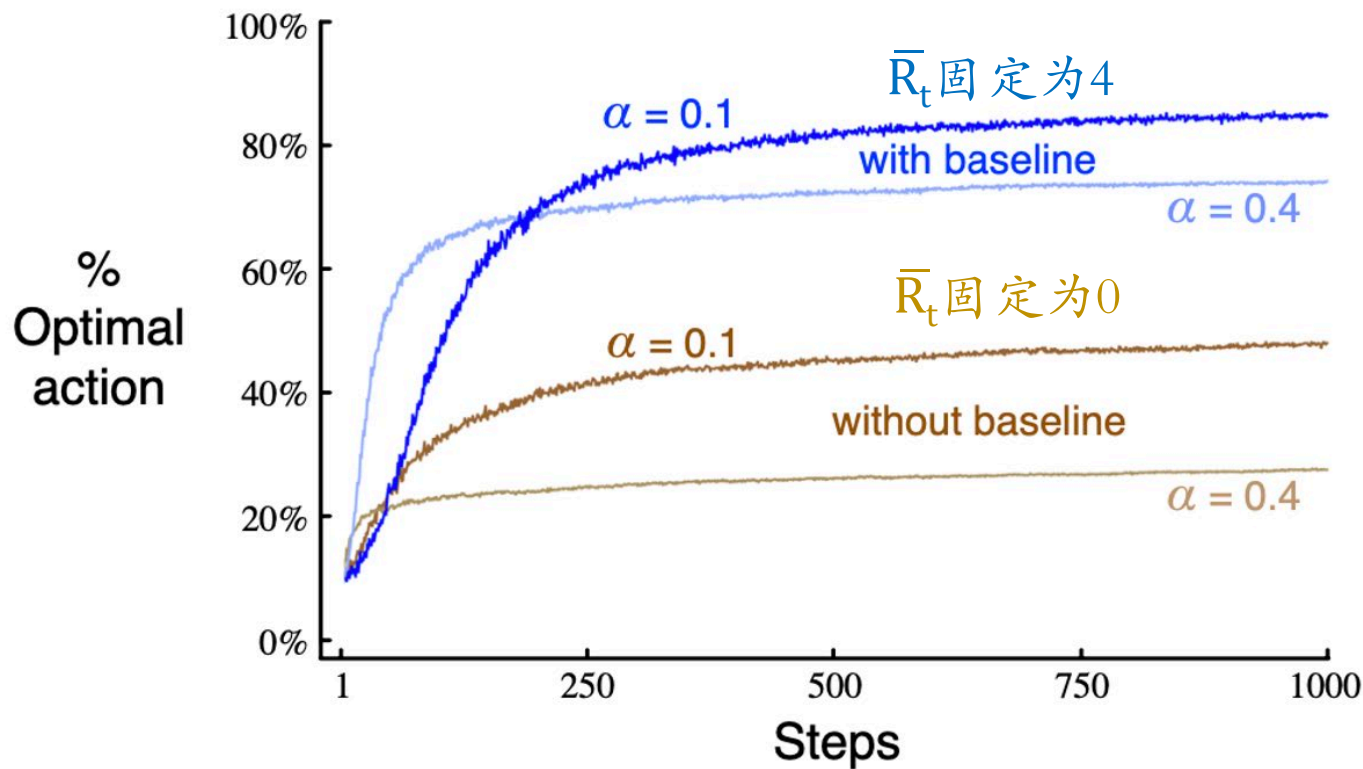
Gradient Bandit Algorithms 梯度下降

- 下面这个例子把老虎机的平均收益上提到均值为+4



Gradient Bandit Algorithms 梯度下降

- 有 baseline 的话原来算法的效果不变，如果没有的话，差别就很大



当然在实际中，我们可以自己算 \bar{R}_t



求解最优策略的几种思路

1. 多臂老虎机问题
2. 计算动作价值
3. 贪心 vs ϵ -贪心
4. ϵ -贪心 vs 乐观初值贪心
5. Upper-Confidence-Bound (UCB)
6. Gradient Bandit Algorithms 梯度下降
7. 算法比较



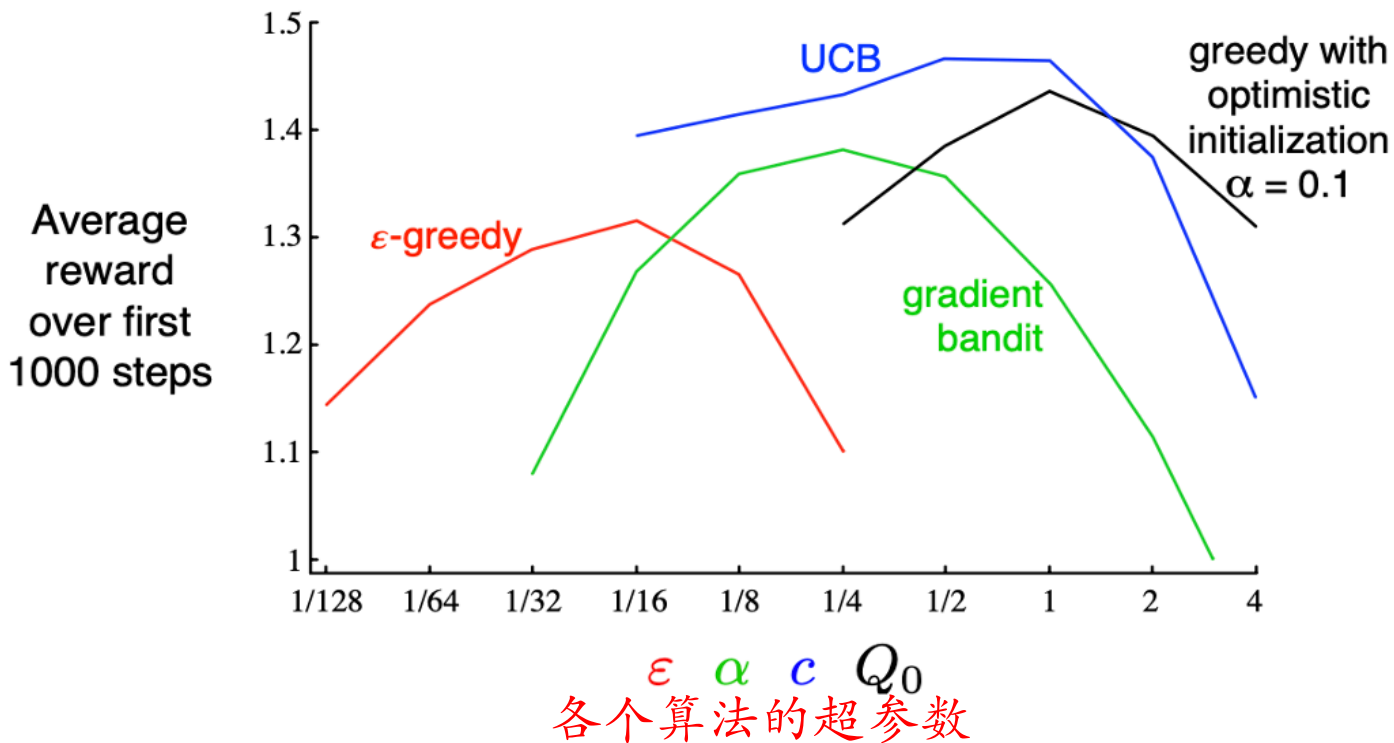
- 我们前面给出了几个简单的**寻找最优策略**的方法
 1. ϵ 贪心算法有一小部分时间随机选：**随机的探索**
 2. UCB 偏向尝试次数小的动作：**根据统计的探索**
 3. 梯度下降法不是估计动作的价值，而是动作的**优先顺序**，使用 soft-max 选动作.
 4. 简单的设置乐观初值可以使得贪心算法也具有相当的探索性



- 哪个方法更好呢?
- 在10臂老虎机问题上比较
- 复杂性在于他们都有参数
- 为了提供有意义的比较我们需要考量他们的效率作为参数的函数
- 实验给出了参数设置及随时间推移的学习效率
- 如果我们画出所有参数和算法的学习曲线，图像就太复杂了不易比较
- 我们用1000步上的平均价值来比较，价值是曲线下围成的面积



超参数研究



- 所有算法都是参数取中间时效果最好
- 不仅要看他最好的参数的效果，还要看它相对于参数是否敏感
- 所有这些算法都不很敏感，在一段参数上表现得挺好
- 总的来说在这个问题上，UCB 看起来最好。

强化学习中的挑战

- 强化学习中最大的挑战
 - 平衡开发和利用的关系
 - 要最大化某个目标，就要选择好的动作，
 - 但是要发现这些好的动作，又要去尝试那些未知收益的新动作。
 - 数学家们已经在这个问题上探索了很久，依旧没有解决这个问题。
- 强化学习是最接近人类和动物行为的机器学习方法，
 - 很多强化学习方法也直接受到了生物学习理论的启发；
 - 强化学习的一些学习模型也对研究心理学模型和动物脑模型有启发作用。



A close-up shot from the movie Inception featuring Leonardo DiCaprio. He is wearing a dark suit and tie, looking slightly to his right with a serious expression. Another person's head and shoulder are visible in the foreground on the right, partially obscuring the frame. The lighting is warm and dim, typical of the film's aesthetic.

WE NEED TO GO

DEEPER