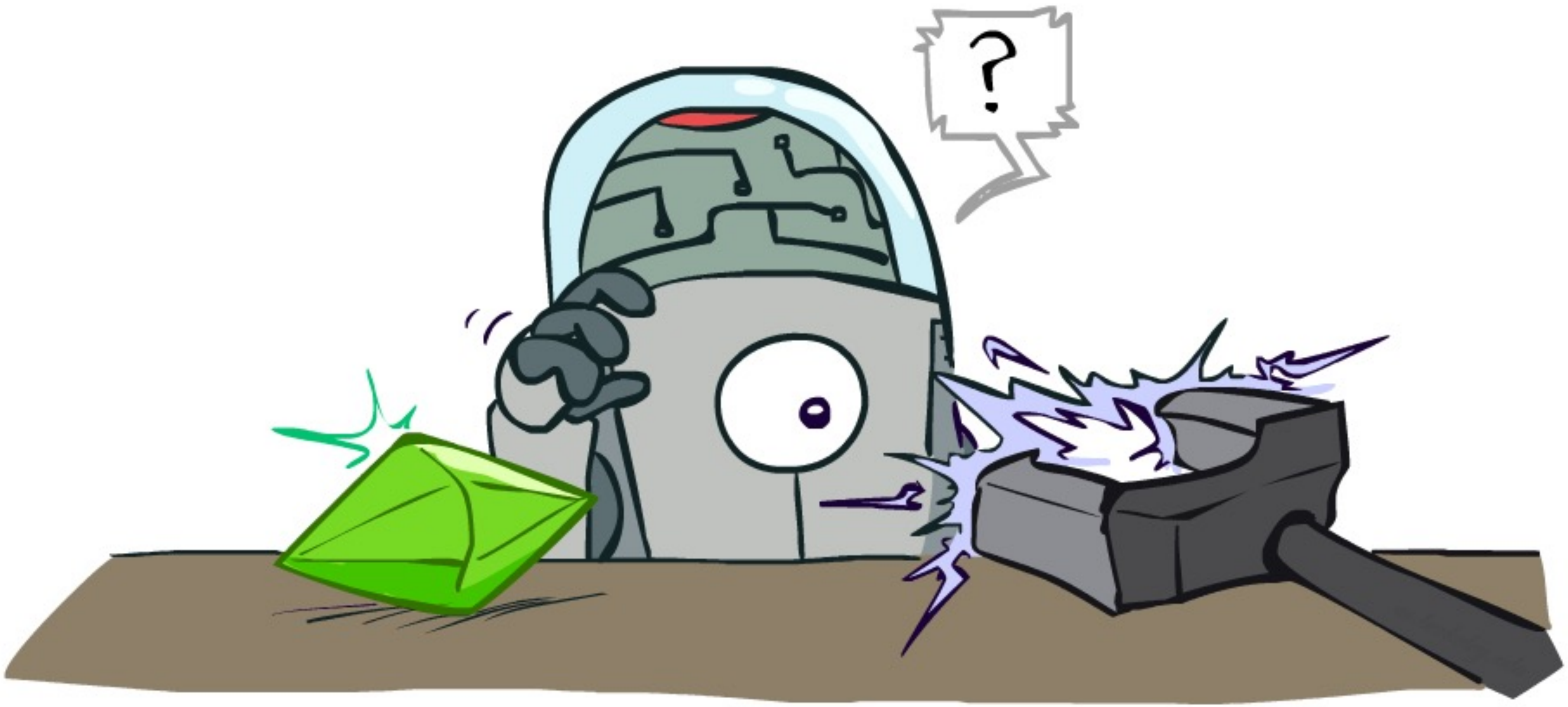
人工智能原理

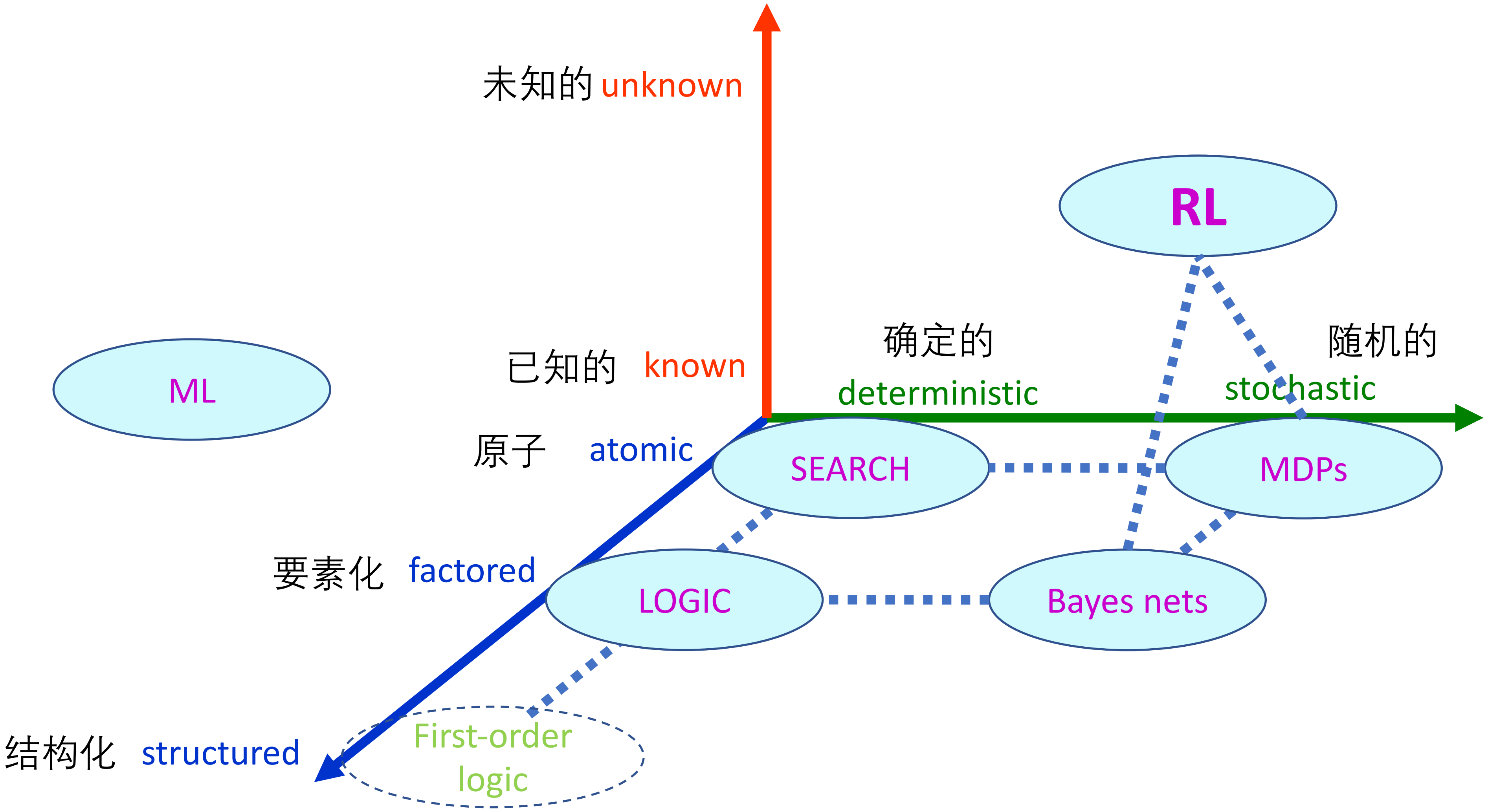
第22章 – 强化学习

彭振辉
中山大学人工智能学院
2024年春季学期

Reinforcement Learning



Reinforcement Learning



强化学习大纲

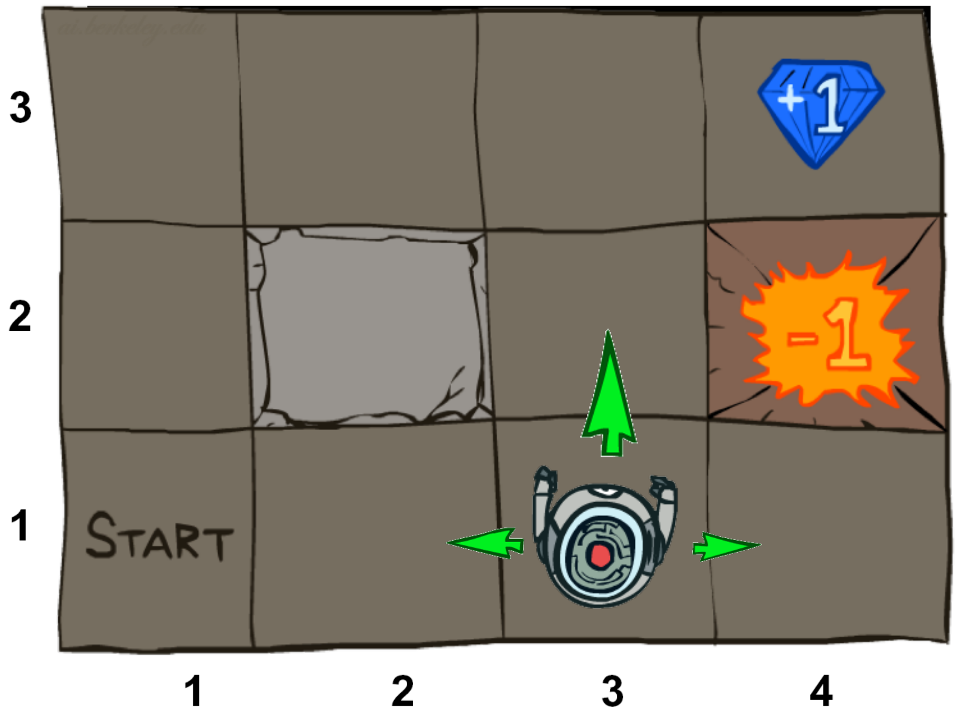
- **被动强化学习**
  - Model-based
    - **自适应动态规划**
  - Model-free
    - **直接效用估计**
    - **时序差分学习**
- 主动强化学习
  - Exploration 和 Exploitation
  - Q-learning
  - Approximate Q-learning

神秘难懂的强化学习?
原理很简单!

- 22.5 – 22.7为课外阅读

回顾：马尔科夫决策过程

- An MDP is defined by:
  - A set of states 状态 $s \in S$
  - A set of actions 行动 $a \in A$
  - A transition model 转移模型 $T(s, a, s')$
    - Probability that a from s leads to s', i.e., $P(s' | s, a)$
  - A reward function 回报/奖励函数 $R(s, a, s')$ for each transition
  - A start state 初始状态
  - Possibly a terminal state 终止状态 (or absorbing state)
  - Utility function which is additive (discounted) rewards



- MDPs are **fully observable** but probabilistic search problems

课程回顾：Value Iteration 价值迭代

- 从 $U_0(s) = 0$ 和一些终止参数 ε 开始

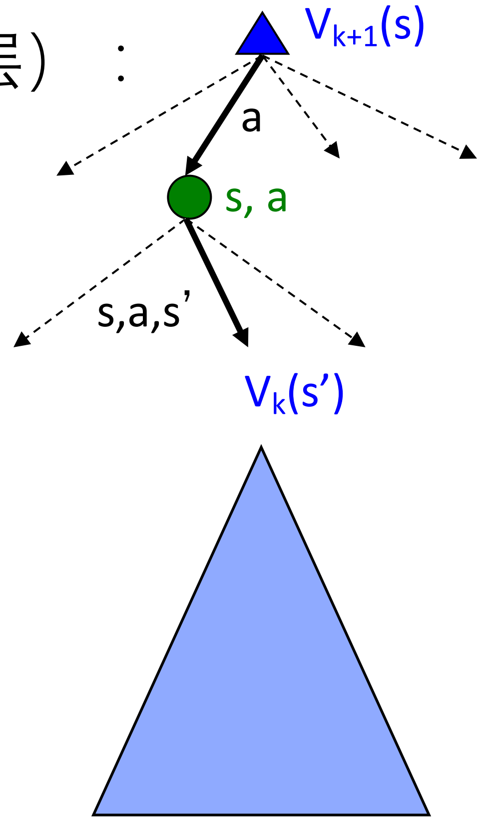
- 重复迭代直到收敛（即，直到所有更新小于 ε）
  - 对每个状态做一个**贝尔曼更新**（本质上是一个 expectimax 层）：

    $U_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s' \mid a,s)\,[R(s,a,s') + \gamma U_k(s')]$

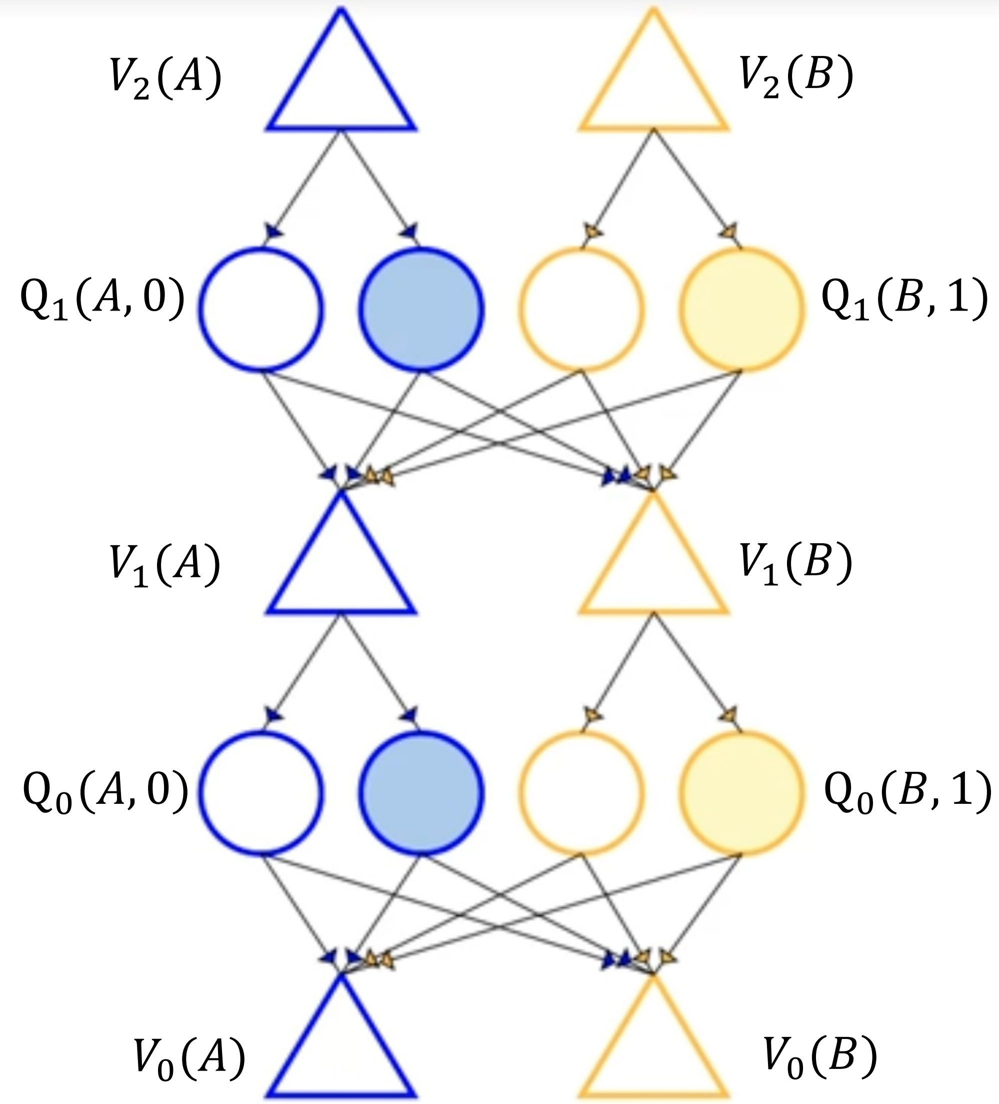
$$V_{k+1}(s) \leftarrow \max_a \underbrace{\sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s')\right]}_{Q_k(s,a)}$$

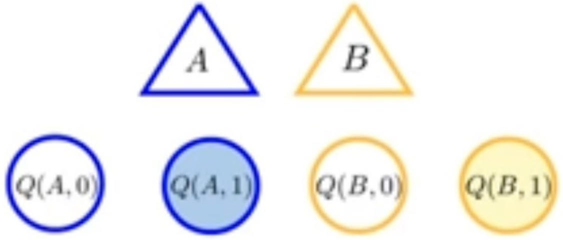
- 定理：将收敛到唯一的最优值

- 每次迭代的运行时间?
  - 每次迭代的复杂度: $O(S^2A)$



价值迭代计算案例





s	a	s'	$T(s,a,s')$	$R(s,a,s')$
A	0	A	0.50	2.0
A	0	B	0.50	-1.0
A	1	A	0.50	1.0
A	1	B	0.50	2.0
B	0	A	0.0	-2.0
B	0	B	1.0	-1.0
B	1	A	0.10	-3.0
B	1	B	0.90	-1.0

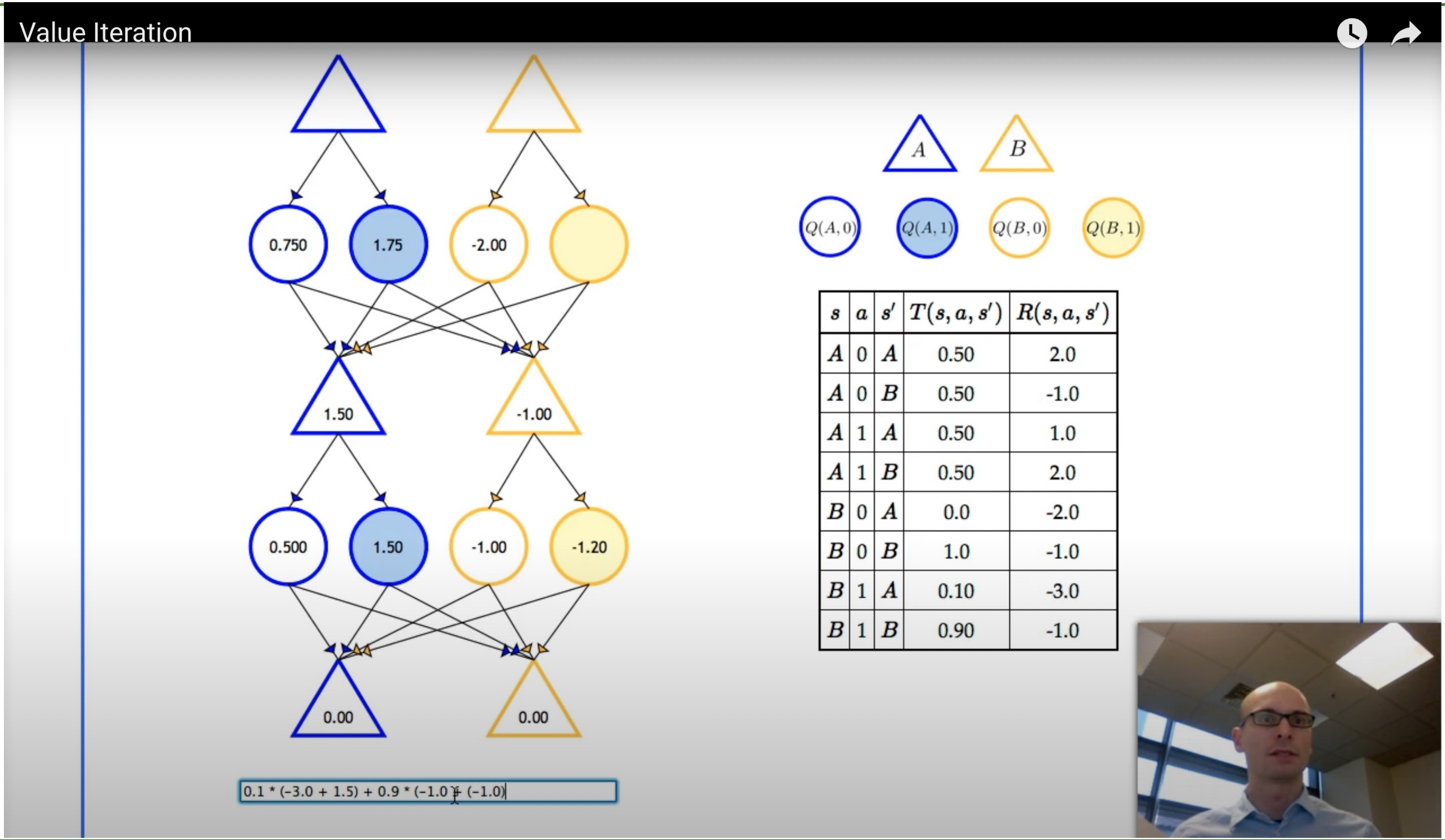
初始V_0=0; 折扣因子 γ=1

试着求左图各个值，比如

$V_1(A)$

$V_1(B)$

价值迭代计算案例



s	a	s'	$T(s,a,s')$	$R(s,a,s')$
A	0	A	0.50	2.0
A	0	B	0.50	-1.0
A	1	A	0.50	1.0
A	1	B	0.50	2.0
B	0	A	0.0	-2.0
B	0	B	1.0	-1.0
B	1	A	0.10	-3.0
B	1	B	0.90	-1.0

https://www.youtube.com/watch?v=A5oqQDNntYc

强化学习

- 仍然假设一个马尔可夫决策过程(MDP):
  - A set of states 状态 s ∈ S
  - A set of actions 行动 (per state) A
  - A model T(s,a,s') 转移模型
  - A reward function R(s,a,s') 回报函数
- 仍然想要求解一个策略 policy π(s)

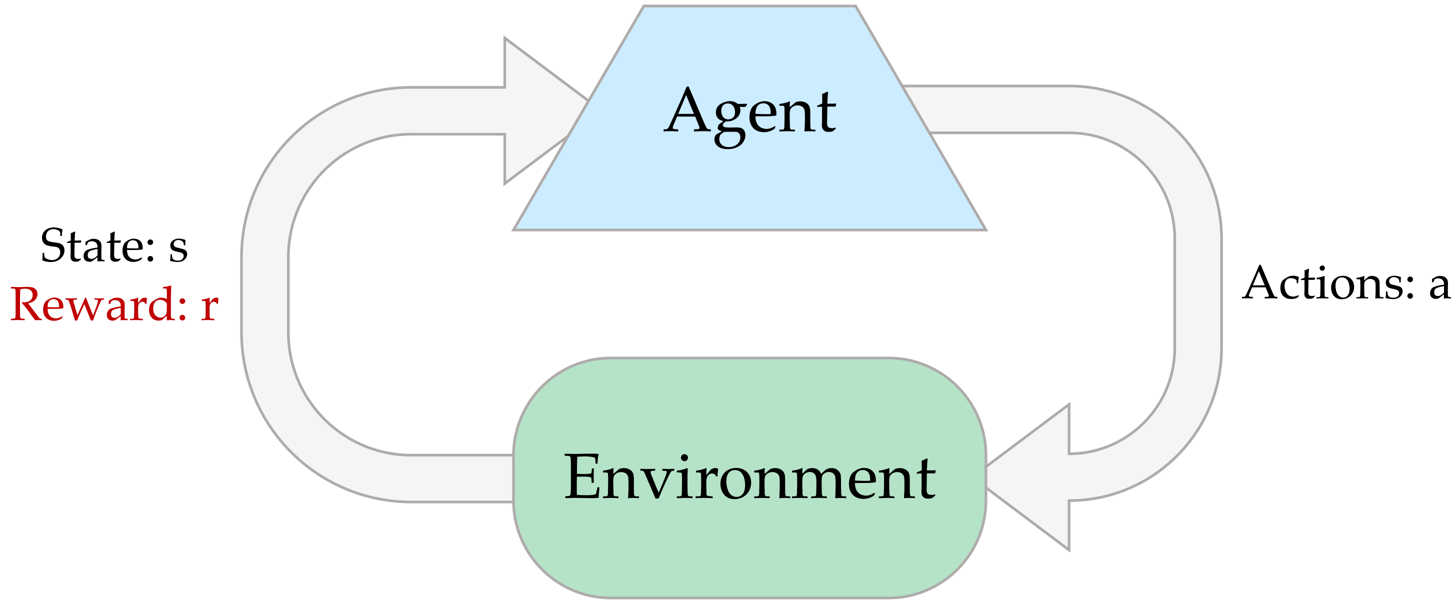
- 新的情况：**不知道 T 或 R**
  - I.e. 我们不知道哪些状态是好的或行动的后果是什么
  - 必须实际尝试行动和状态去**学习**
  - **Note: 不知道的事物，并不代表它们不存在，去学习！**

Reinforcement Learning

Basic ideas:

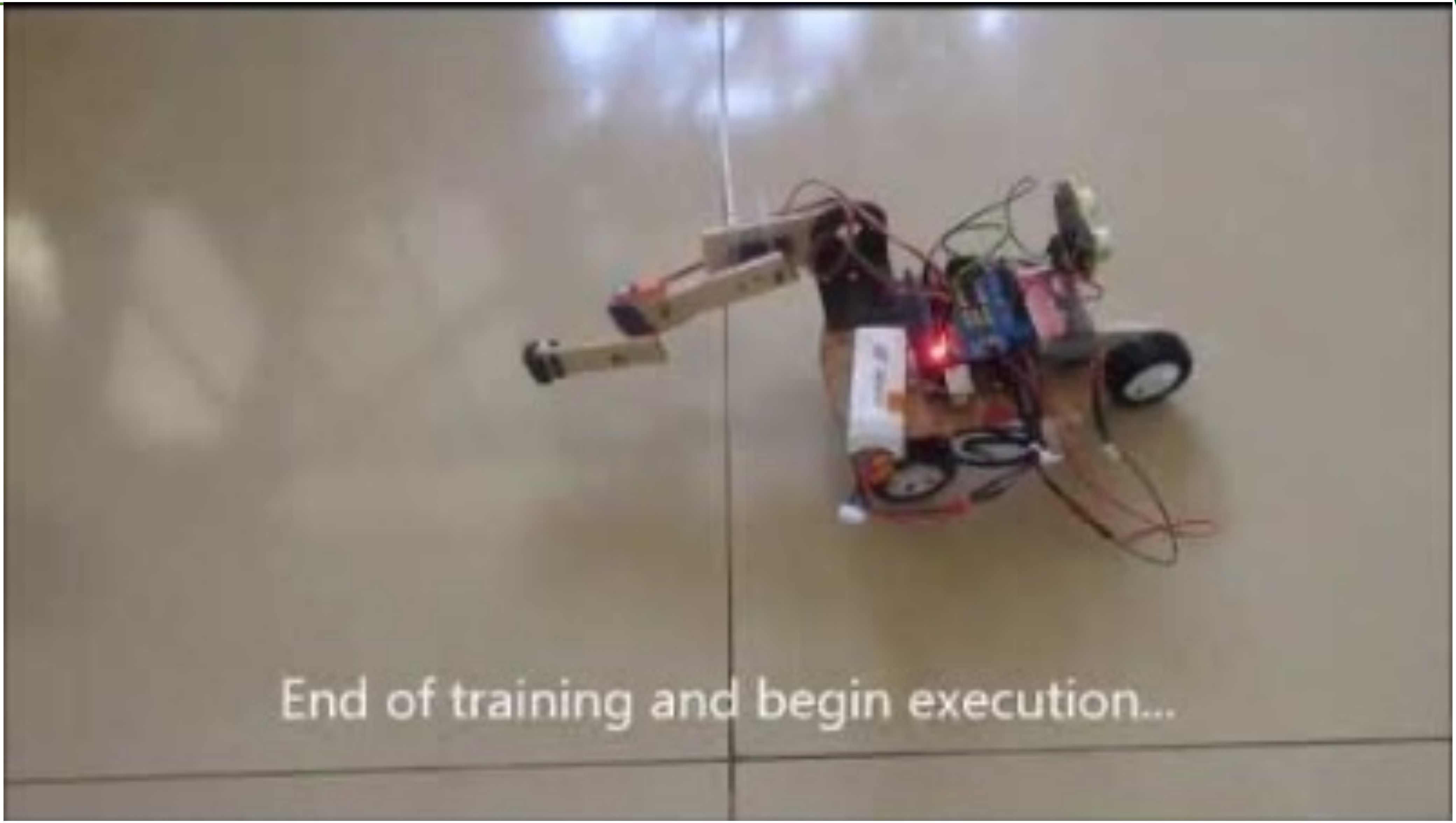
- ***Exploration* 探索**: 你必须***尝试未知的行动***才能获取信息
- ***Exploitation* 利用 (信息)**: 最终，你必须使用你所知道的信息
- ***Sampling* 采样**: 你可能需要重复多次才能获得良好的估计
- ***Generalization* 泛化**: 你在一个状态学到的东西也可能适用于其它状态

Reinforcement Learning



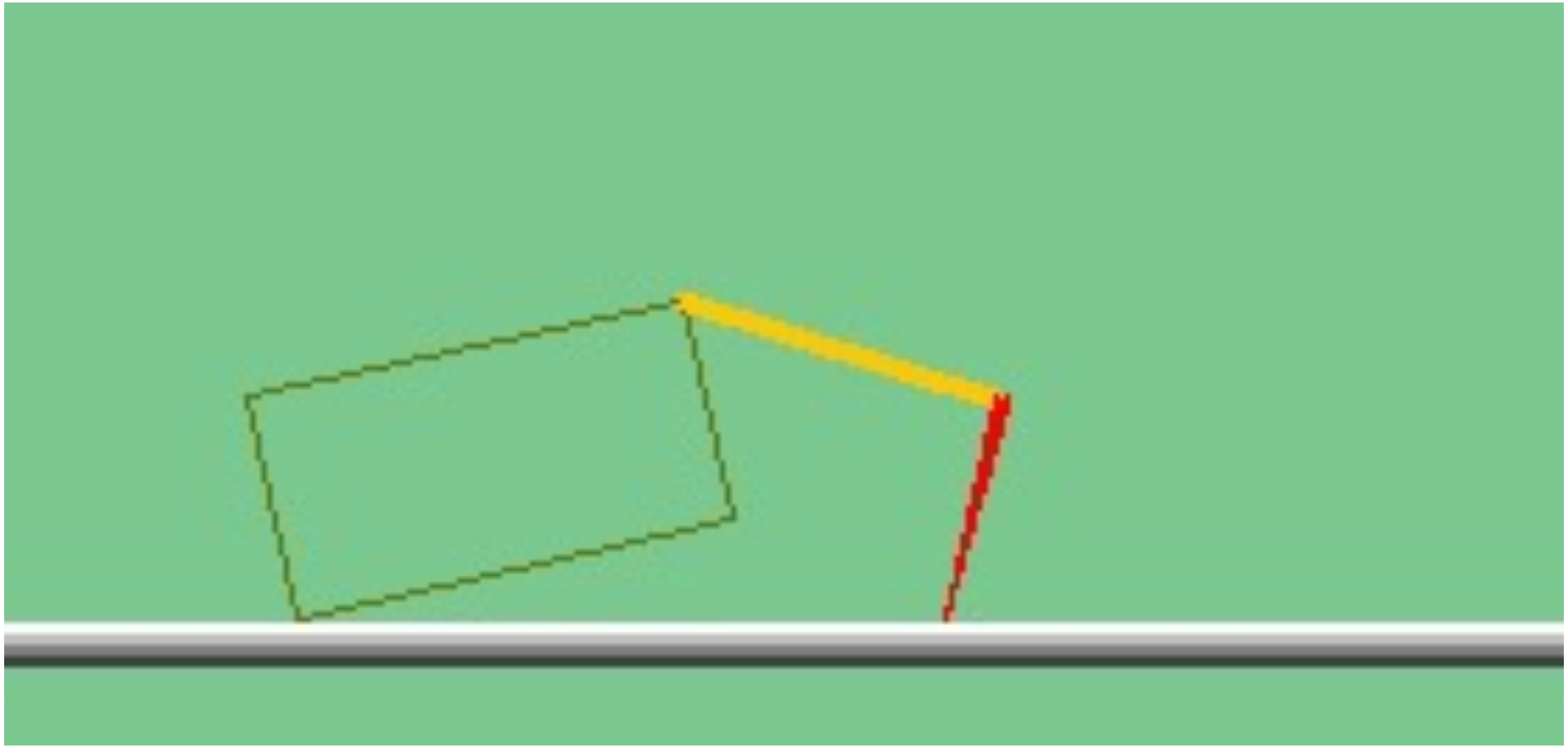
- Basic idea:
  - 以回报(rewards)的形式接收反馈
  - Agent 的效用由回报函数定义
  - 必须（学会）采取行动以最大化期望回报(maximize expected rewards)
  - 所有的学习都是基于观察到的结果样本！

The Real Crawler!



Source: https://www.instructables.com/Q-Learning-machine-Learning-Crawler/

The Crawler!



DeepMind Atari

Learning to Walk

Initial

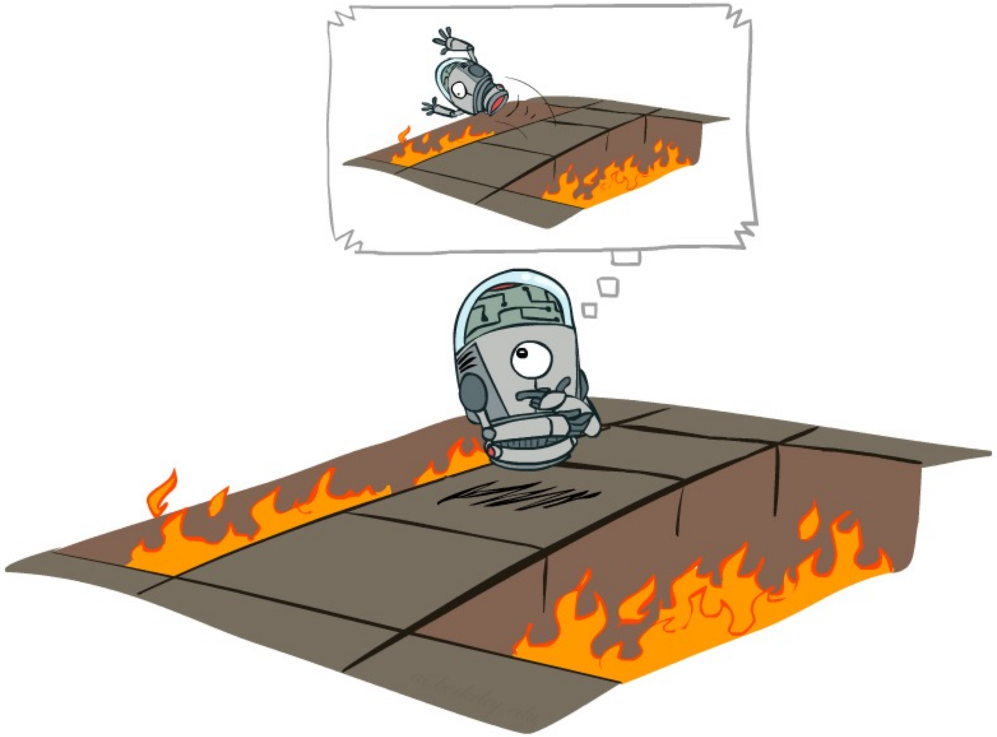
[Kohl and Stone, ICRA 2004]

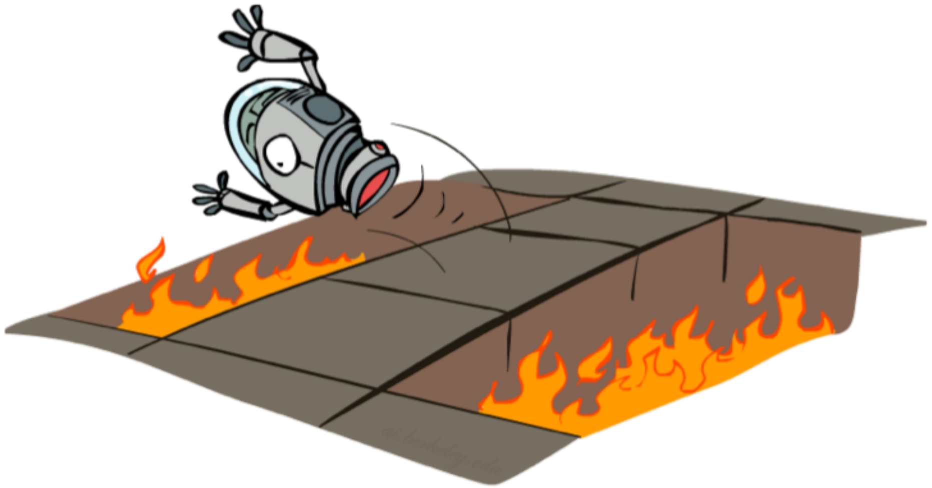
Learning to Walk

Finished

[Kohl and Stone, ICRA 2004]

Offline (MDPs) vs. Online (RL)





Offline Solution

Agent拥有环境的完整模型，并且知道回报函数

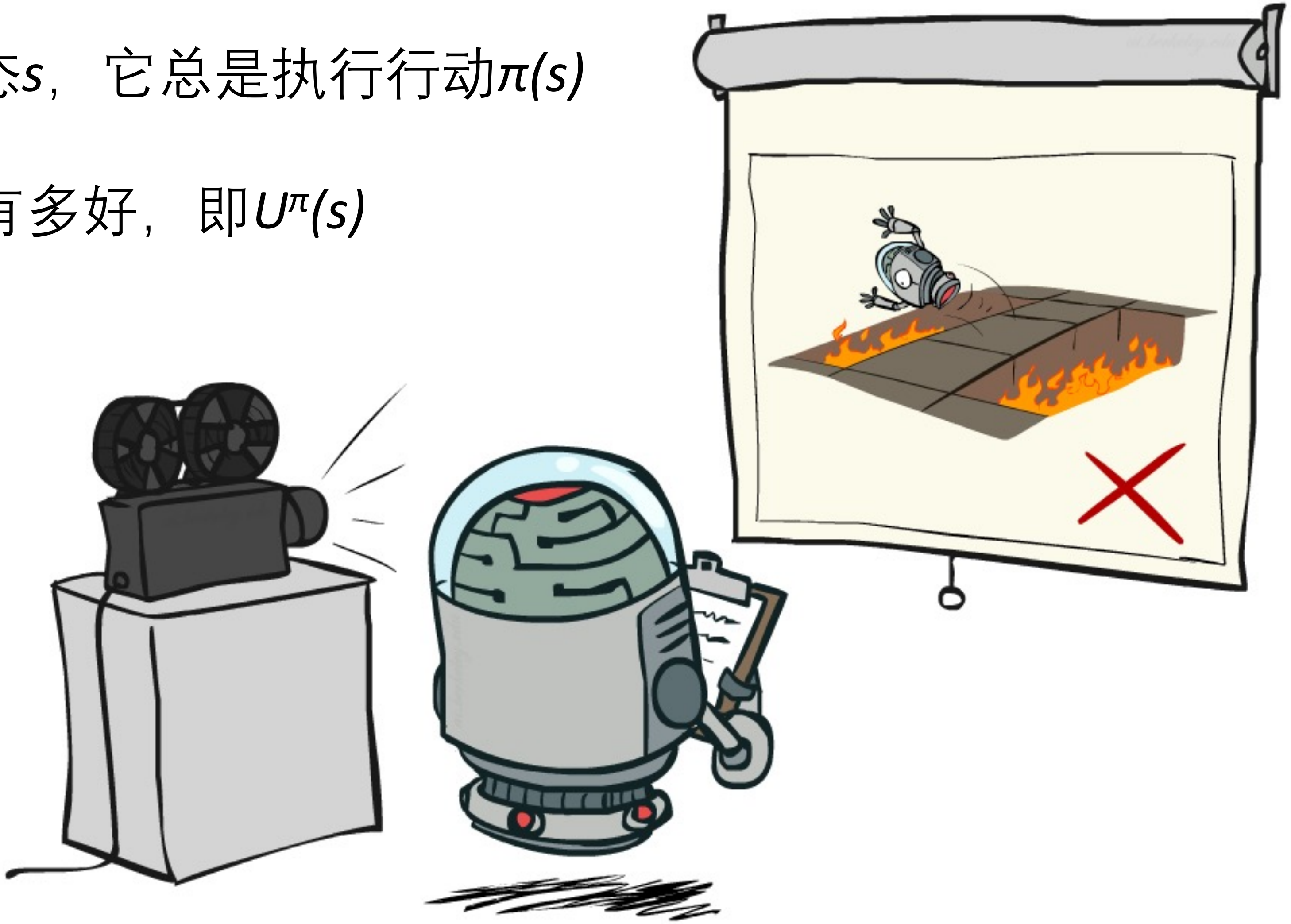
Online Learning

Agent没有关于两者的先验知识

Passive Reinforcement Learning 被动强化学习

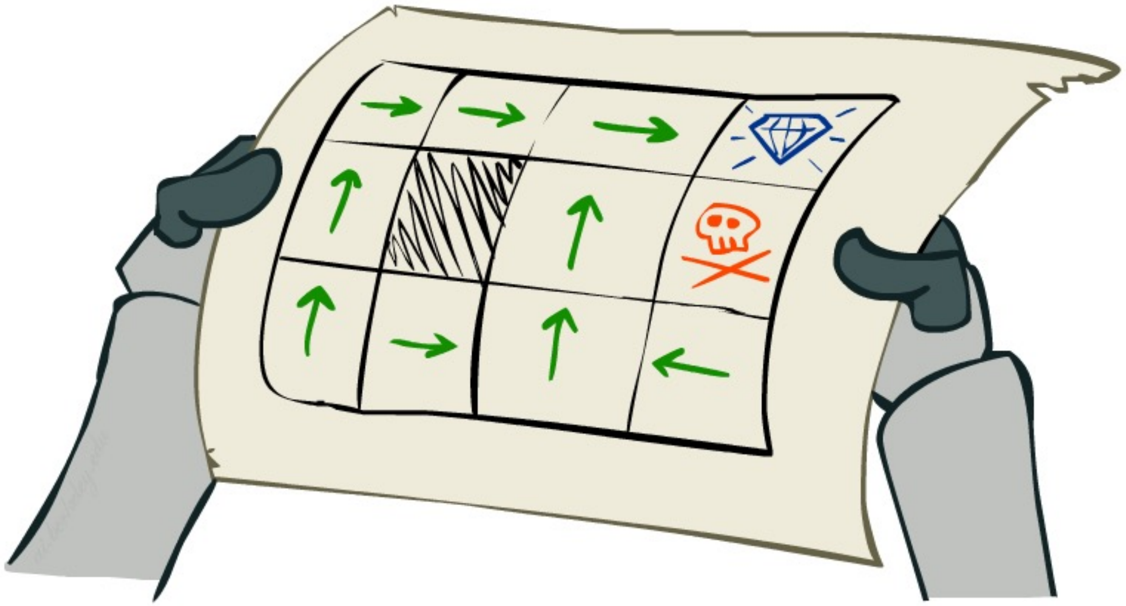
Agent的策略π是固定的：在状态*s*，它总是执行行动*π(s)*

其目标只是简单地学习该策略有多好，即$U^{\pi}(s)$



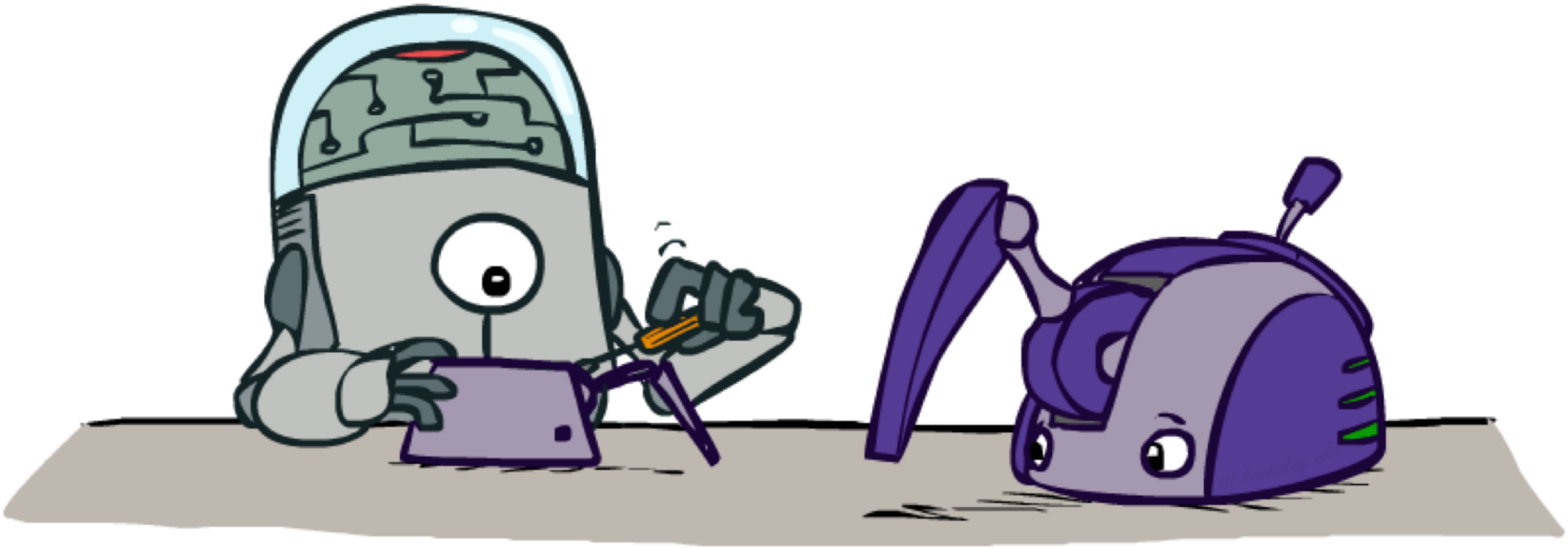
Passive Reinforcement Learning

- 简化的任务：策略评估 policy evaluation
  - 输入：固定的策略 $\pi(s)$
  - 你不知道转移模型 T(s,a,s')
  - 你不知道回报 R(s,a,s')
  - 目标：学习状态值



- 在这种情况下：
  - 学习者“随心所欲”
  - 无法选择采取什么行动
  - 只需执行策略并吸取经验
  - 这不是离线规划！你实际上在世界上采取了行动

Model-Based Learning



Model-Based Learning 基于模型的学习

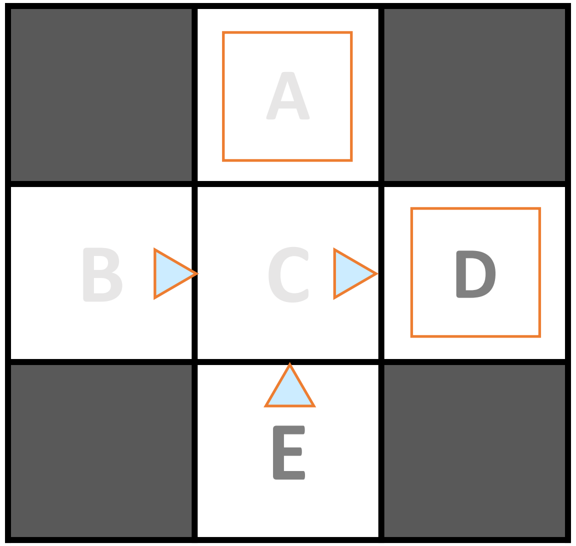
- Model-Based Idea:
  - 根据经历学习近似的模型
  - 求解学习到的 MDP

- 第 1 步：根据经历学习MDP 模型
  - 计算每个 s, a 的结果 s'
  - 归一化以估计 $\hat{T}(s, a, s')$
  - 当我们经历（s, a, s' ）时发现每一个 $\hat{R}(s, a, s')$

- 第 2 步：求解学习到的 MDP
  - 例如，像以前一样使用 value iteration

Example: Model-Based Learning

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Learned Model

$$\hat{T}(s, a, s')$$

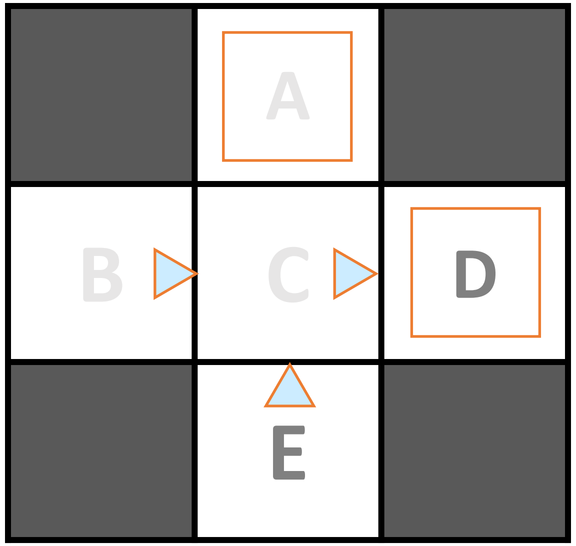
T(B, east, C) =
T(C, east, D) =
T(C, east, A) =
…

$$\hat{R}(s, a, s')$$

R(B, east, C) =
R(C, east, D) =
R(D, exit, x) =
…

Example: Model-Based Learning

Input Policy π



Assume: γ = 1

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Learned Model

$\hat{T}(s, a, s')$

T(B, east, C) = 1.00
T(C, east, D) = 0.75
T(C, east, A) = 0.25
…

$\hat{R}(s, a, s')$

R(B, east, C) = -1
R(C, east, D) = -1
R(D, exit, x) = +10
…

Pros and cons

- Pro:
  - 有效地利用了经历
- Con:
  - 可能无法扩展到大型状态空间
    - 一次学习一个状态-动作对模型
    - 无法解决 |S| 非常大的 MDP

类比例子: 期望年龄

目标：计算 人工智能原理课 学生的期望年龄

已知P(A) 的情况

$$E[A] = \sum_a P(a) \cdot a \qquad = 0.35 \times 20 + \ldots$$

未知P(A) 时, 收集样本[a_1, a_2, ... a_N]

未知 P(A): “Model Based”

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$

$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

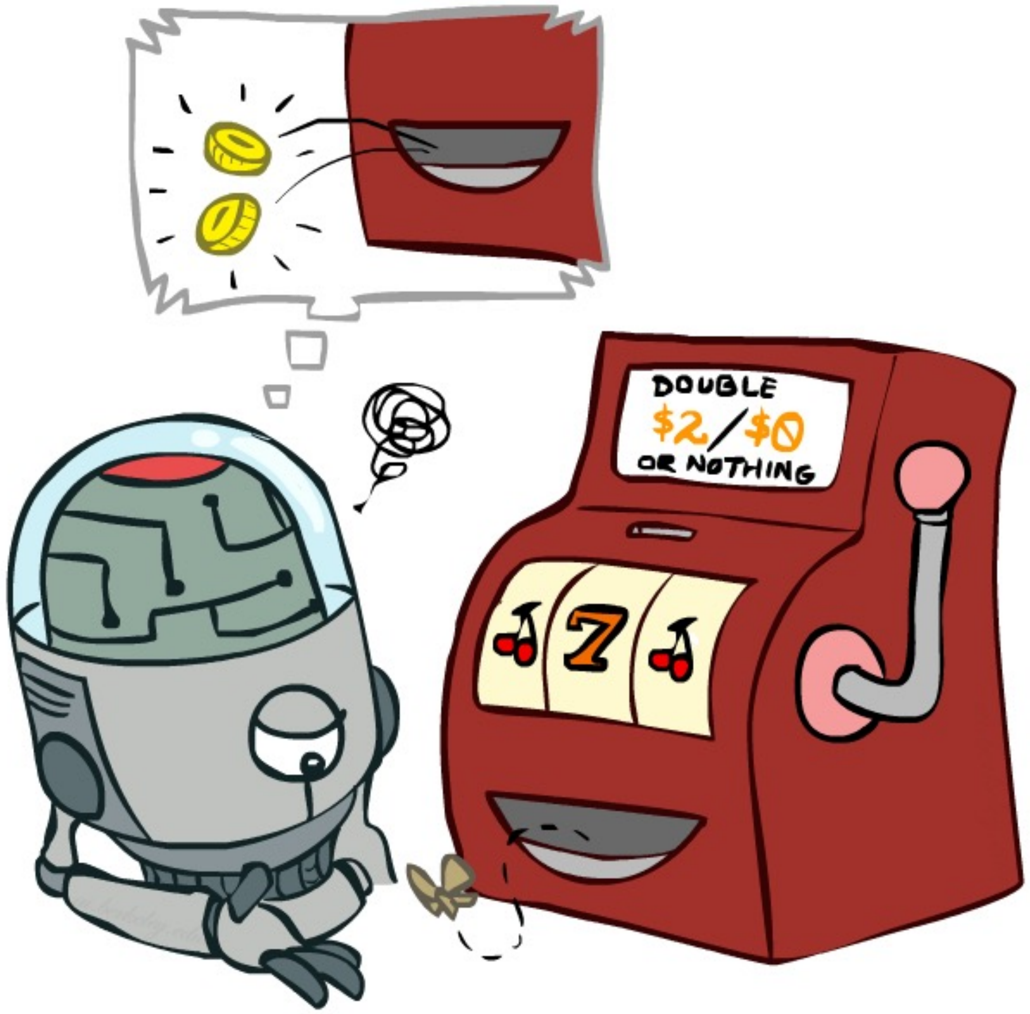
为什么这行得通？
因为最终你会学
习到正确的模型。

未知 P(A): “Model Free”

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

为什么这行得通？
因为样本以正确
的频率出现。

Model-Free Learning



Direct Evaluation 直接(效用)估计

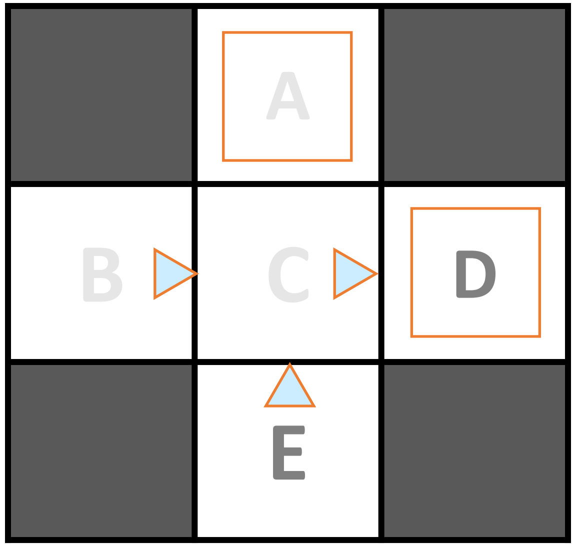
- 目标：计算 π 下每个状态的（效用）值

- idea：平均化观察到的样本值
  - 按π行动
  - 每次访问一个状态时，写下折扣回报的总和是多少
  - 取这些样本的折扣回报平均值

- 这称为直接（效用）估计

Example: Direct Evaluation

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

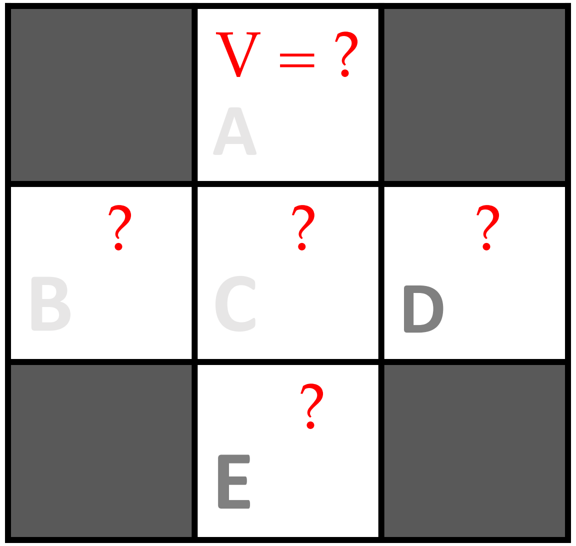
Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

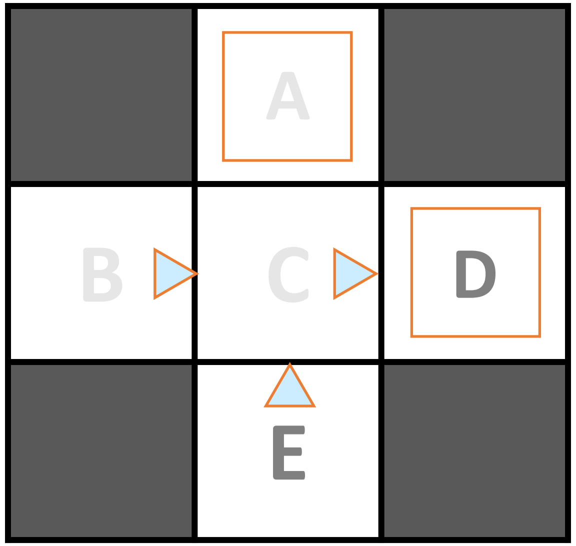
E, north, C, -1
C, east, A, -1
A, exit, x, -10

Output Values



Example: Direct Evaluation

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

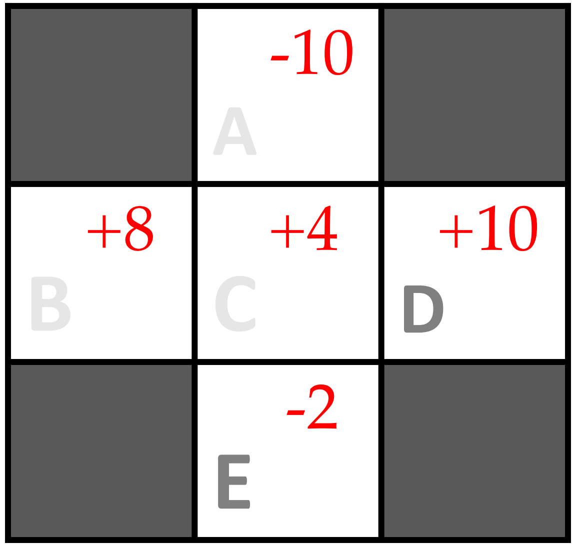
Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

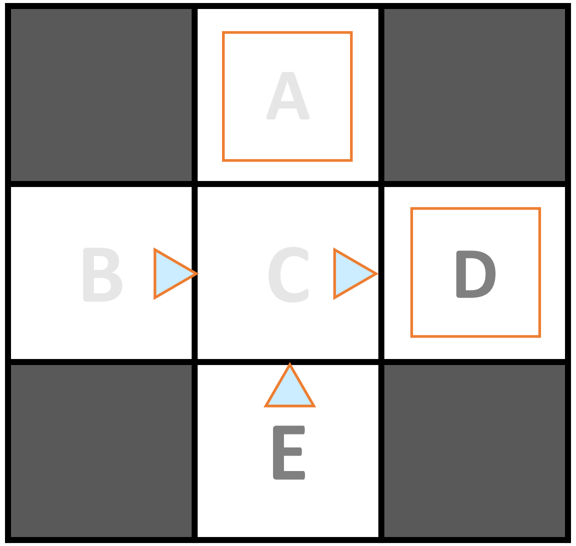
E, north, C, -1
C, east, A, -1
A, exit, x, -10

Output Values



Quiz: Direct Evaluation

Input Policy π



What if γ = 0.5?

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

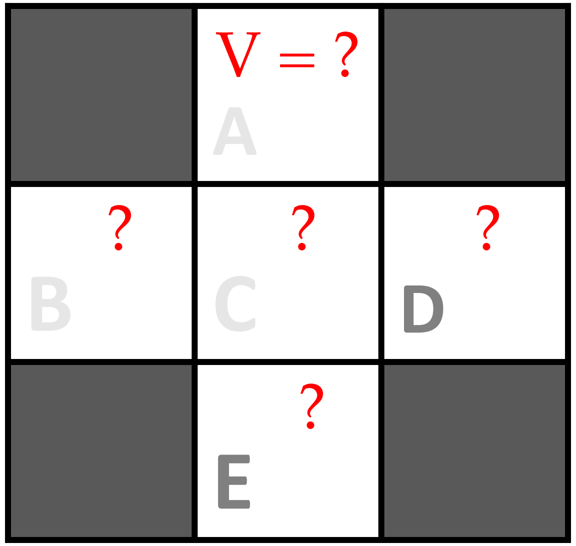
Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

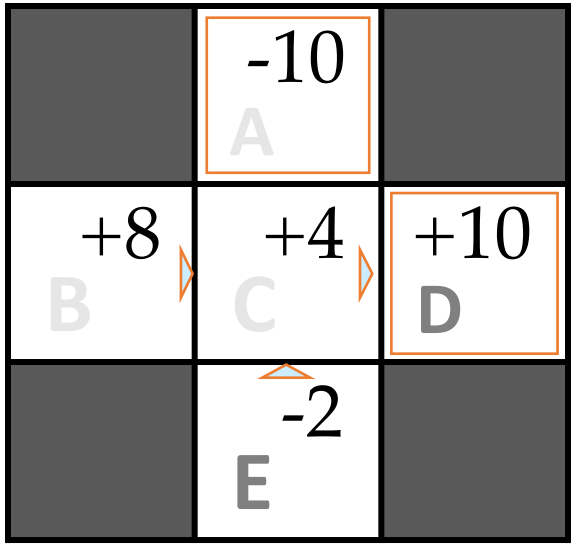
Output Values



直接效用估计的好处和问题

- 好处
  - 很容易理解
  - 它不需要任何关于 T、R 的知识
  - 它使用样本中体现的转换关系，最终 (大样本下) 能计算正确的平均效用值
- 问题
  - 得等到一次经历(episode)完成才能学习
  - 浪费了有关状态与状态之间的信息
  - 每个状态都必须分开学习
  - 所以学起来需要很长时间
  - 思考：所以为什么学习起来需要很长时间? 怎么办呢?

Output Values

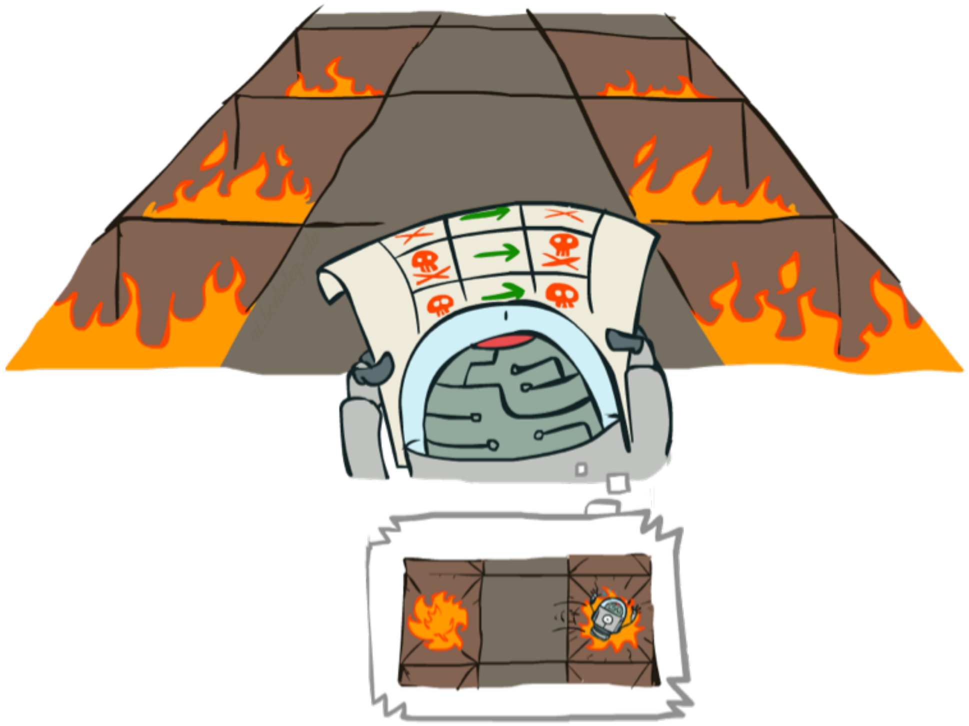


If B and E both go to C under this policy, how can their values be different?

忽略了C->D, C->A的转移概率，上上页的例子中，恰好是从E出发的那次经历最后到达了A，得到-10。

E.g., B=at home, study hard
E=at library, study hard
C=know material, go to exam

Temporal difference (TD) learning 时序差分学习



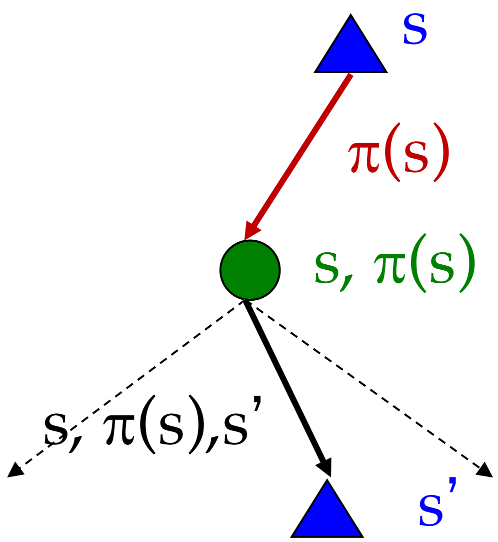
策略迭代？

- 简化的 Bellman 更新计算**固定策略**的 V:
  - 每一轮，用 V 上的一步前瞻(one-step-look-ahead)层替换 V

$$V_0^{\pi}(s) = 0$$

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_k^{\pi}(s')]$$

  - 这种方法充分利用了状态之间的联系
  - 不幸的是，我们需要 T 和 R 来做到这一点！



- 关键问题：**我们如何在不知道 T 和 R 的情况下对 V 进行更新?**
  - 在model-free的思想下，我们如何在不知道权重的情况下取加权平均值?

TD as approximate Bellman update

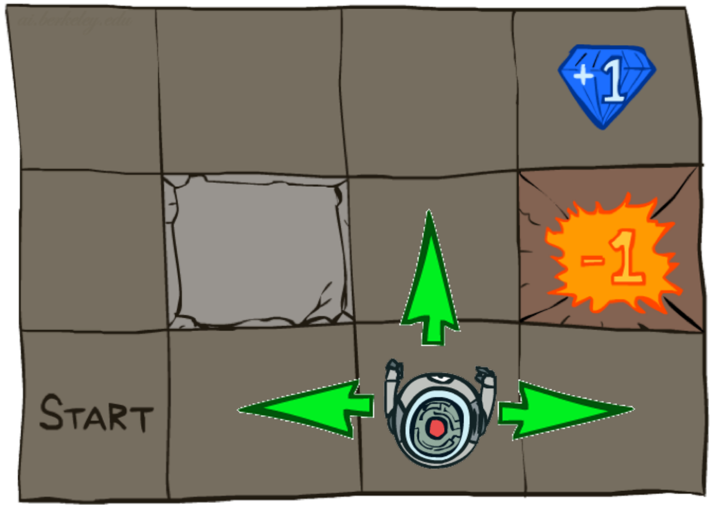
- 我们想通过计算一些平均值来改进我们对 V 的估计:

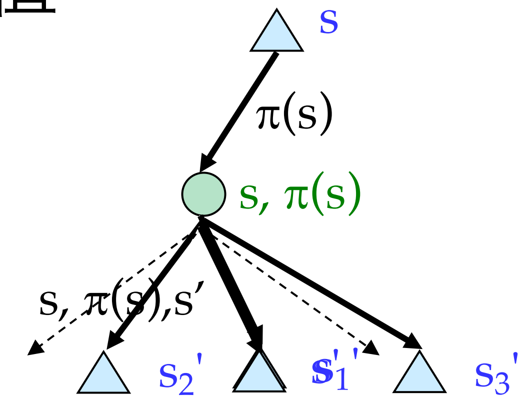
$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_k^{\pi}(s')]$$

- Idea 1: 对结果 s’ 进行采样（通过执行行动！）并取平均值

$$sample_1 = R(s,\pi(s),s_1') + \gamma V_k^{\pi}(s_1')$$
$$sample_2 = R(s,\pi(s),s_2') + \gamma V_k^{\pi}(s_2')$$
$$\dots$$
$$sample_n = R(s,\pi(s),s_n') + \gamma V_k^{\pi}(s_n')$$

$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n}\sum_i sample_i$$

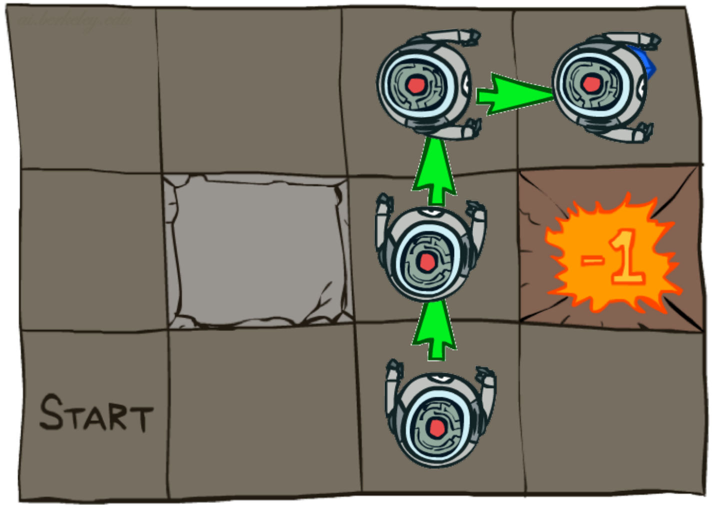




Almost! But we can’t rewind time to get sample after sample from state s.
我们在尝试时，不能重复回到状态s去尝试

TD as approximate Bellman update

- Idea 2: Update value of s after each transition s,a,s',r :
  - Update $V^{\pi}([3,1])$ based on $R([3,1],up,[3,2])$ and $\gamma V^{\pi}([3,2])$
  - Update $V^{\pi}([3,2])$ based on $R([3,2],up,[3,3])$ and $\gamma V^{\pi}([3,3])$
  - Update $V^{\pi}([3,3])$ based on $R([3,3],right,[4,3])$ and $\gamma V^{\pi}([4,3])$
- Idea 3: Update values by maintaining a ***running average***



Running Averages

- How do you compute the average of 1, 4, 7?
- Method 1: add them up and divide by N
  - 1+4+7 = 12
  - average = 12/N = 12/3 = 4
- Method 2: keep a running average μ_n and a running count n
  - $n=0 \quad \mu_0=0$
  - $n=1 \quad \mu_1 = (0 \cdot \mu_0 + x_1)/1 = (0 \cdot 0 + 1)/1 = 1$
  - $n=2 \quad \mu_2 = (1 \cdot \mu_1 + x_2)/2 = (1 \cdot 1 + 4)/2 = 2.5$
  - $n=3 \quad \mu_3 = (2 \cdot \mu_2 + x_3)/3 = (2 \cdot 2.5 + 7)/3 = 4$
  - General formula: $\mu_n = ((n-1) \cdot \mu_{n-1} + x_n)/n$

    $= [(n-1)/n]\, \mu_{n-1} + [1/n]\, x_n$ (weighted average of old mean, new sample)

两个权重值有什么联系?

它们都随n的变化而变化，每次都要新算一个值，有没有简化方法?

Running Averages

- What if we use a weighted average with a fixed weight?
  - $\mu_n = (1-\alpha)\, \mu_{n-1} + \alpha\, x_n$
  - n=1 $\mu_1 = x_1$
  - n=2 $\mu_2 = (1-\alpha) \cdot \mu_1 + \alpha x_2 = (1-\alpha) \cdot x_1 + \alpha x_2$
  - n=3 $\mu_3 = (1-\alpha) \cdot \mu_2 + \alpha x_3 = (1-\alpha)^2 \cdot x_1 + \alpha(1-\alpha) x_2 + \alpha x_3$
  - n=4 $\mu_4 = (1-\alpha) \cdot \mu_3 + \alpha x_4 = (1-\alpha)^3 \cdot x_1 + \alpha(1-\alpha)^2 x_2 + \alpha(1-\alpha) x_3 + \alpha x_4$
- **早期的x值在当前时间点，会发生什么变化?**
- I.e., ***exponential forgetting*** of old values
- 拓展：
  - μ_n is a convex combination[1] of sample values (weights sum to 1)
  - $E[\mu_n]$ is a convex combination of $E[X_i]$ values, hence unbiased

[1] 凸组合 https://baike.baidu.com/item/%E5%87%B8%E7%BB%84%E5%90%88/18999826

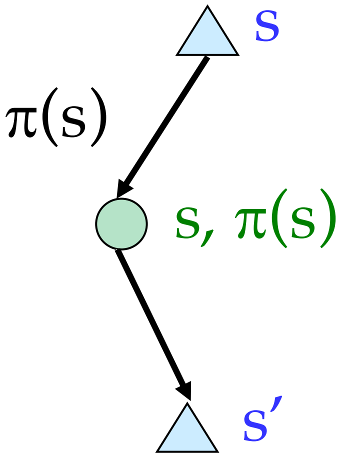
TD as approximate Bellman update

- Idea 3: Update values by maintaining a ***running average***
- sample = $R(s,\pi(s),s') + \gamma V^{\pi}(s')$
- $V^{\pi}(s) \leftarrow (1-\alpha) \cdot V^{\pi}(s) + \alpha \cdot \text{sample}$
- $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha \cdot [\text{sample} - V^{\pi}(s)]$
- This is the ***temporal difference learning rule*** 时序差分
- $[\text{sample} - V^{\pi}(s)]$ is the “TD error”
- α is the ***learning rate 学习率***
- I.e., observe a sample, move $V^{\pi}(s)$ a little bit to make it more consistent with its neighbor $V^{\pi}(s')$

Temporal Difference Learning 时序差分学习

Note: 人也一样！

- Big idea: 从每一次状态转移中学习!
  - **我们每经历一次状态转移 (s, a, s’, r)，更新一次 V(s)**
  - 越可能的转移结果s’将越频繁地参与到更新中
- 状态值的时序差分学习
  - 状态还是固定的，同样也是做状态评估！
  - 将效用估计朝着理想均衡的方向调整
  - **α 作为学习速度参数，随某个状态被访问次数的增加而递减，状态值函数能收敛**



Sample of V(s): $sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$

Update to V(s): $V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$

Same update: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

Temporal Difference Learning 时序差分学习

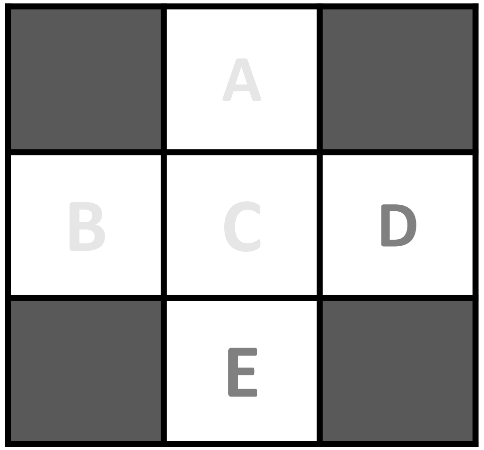
```
function PASSIVE-TD-LEARNER(percept) returns 一个动作
  inputs: percept, 指示当前状态s'与奖励信号r的某个感知
  persistent: π，一个确定性策略
              s，之前的状态，初始为空
              U，关于状态效用的表，初始化为空
              N_s，关于状态出现频率的表，初始为零

  if s'是一个新的状态 then U[s'] ← 0
  if s非空 then
    增加N_s[s]
    U[s] ← U[s] + α(N_s[s]) × (r + γU[s'] − U[s])
  s ← s'
  return π[s']
```

图22-4 一种使用时序差分方法学习效用估计的被动强化学习智能体。我们选择适当的步长函数$\alpha(n)$以确保收敛

例子: 时序差分学习

States



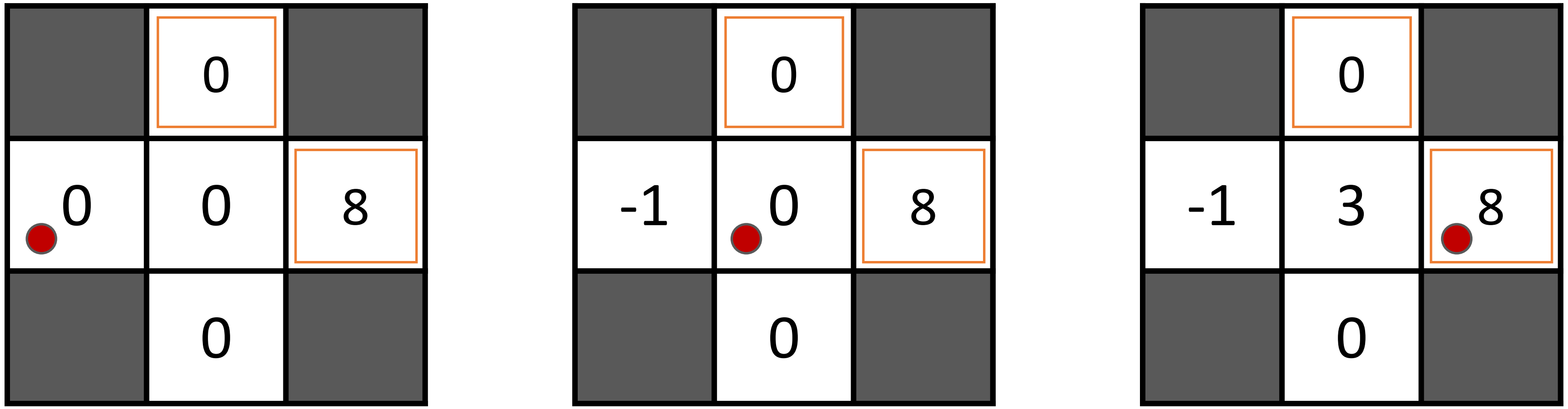
Assume: $\gamma = 1$,

$\alpha = 1/2$

Observed Transitions

B, east, C, -2

C, east, D, -2



$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right]$$

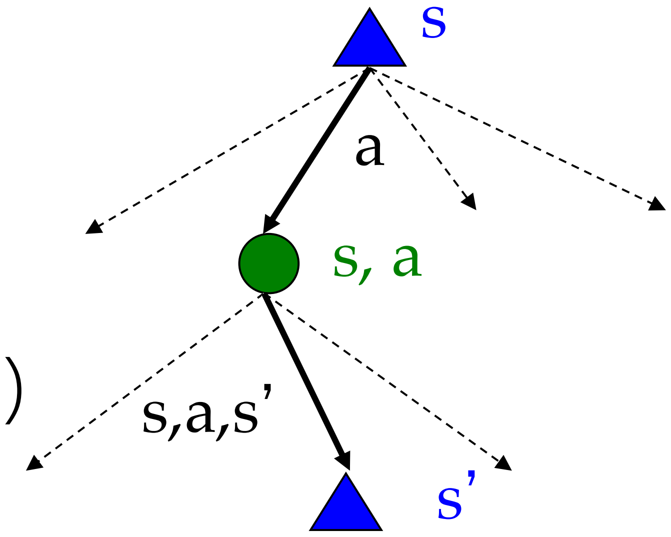
$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$$

Problems with TD Value Learning

- TD 价值学习是一种无模型(model-free)的策略评估方法，通过运行样本获得的状态值的平均值模拟贝尔曼更新
- 但是，如果我们想将价值转变为（新的）策略，我们就会遇到问题：

$$\pi(s) = \arg\max_a Q(s,a)$$

$$Q(s,a) = \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V(s')\right]$$



- 我们不知道 T 和 R!
- 应对方法: 学习 Q-values (Q值), 而不是 values (效用值)
- 让行动选择也变得无模型