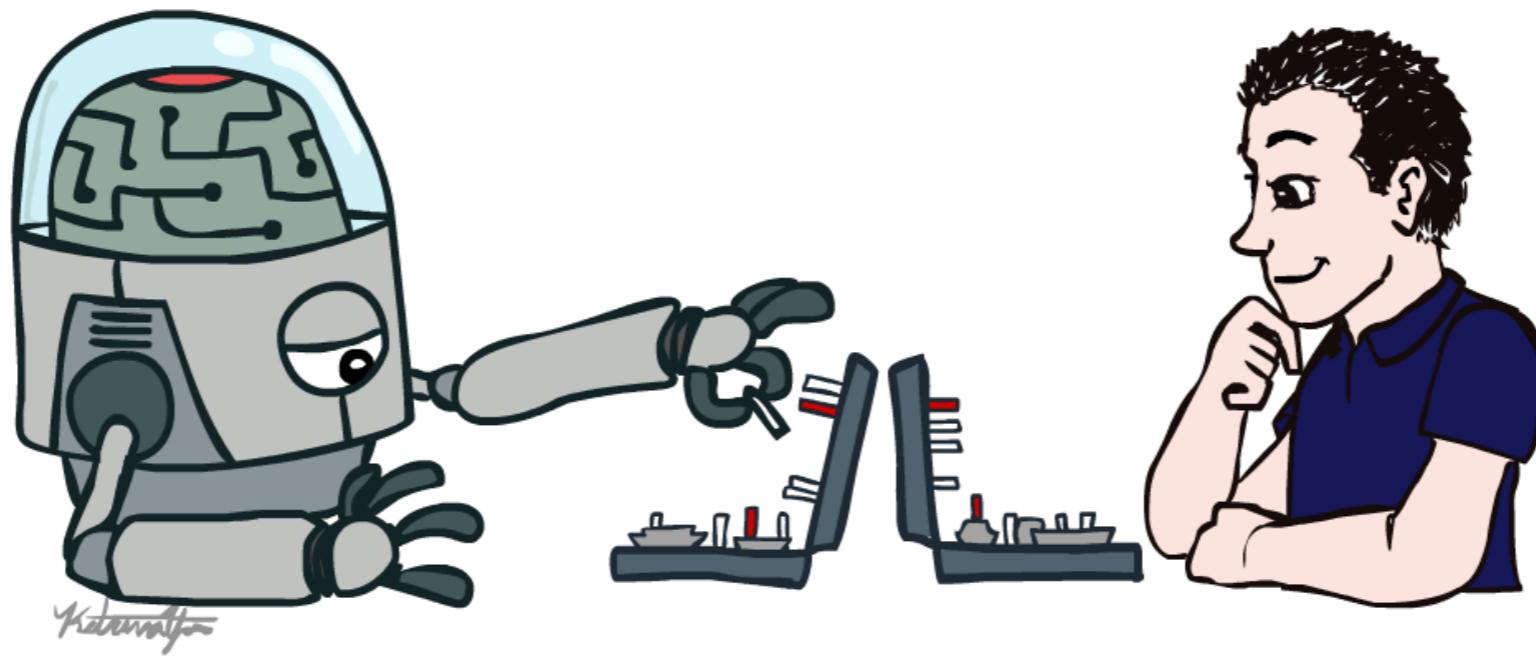
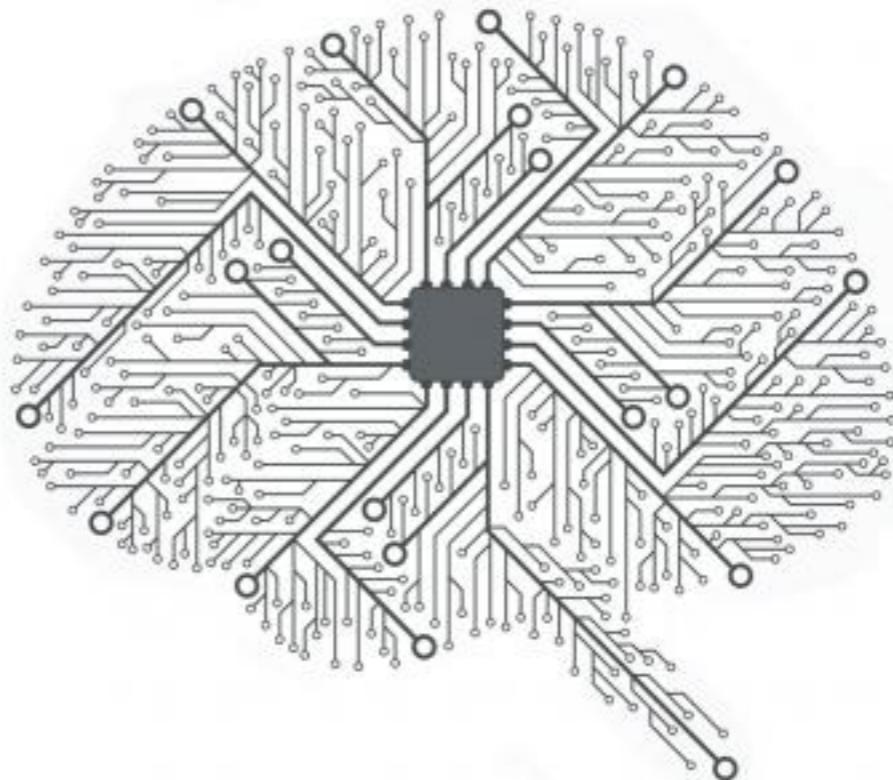


人工智能

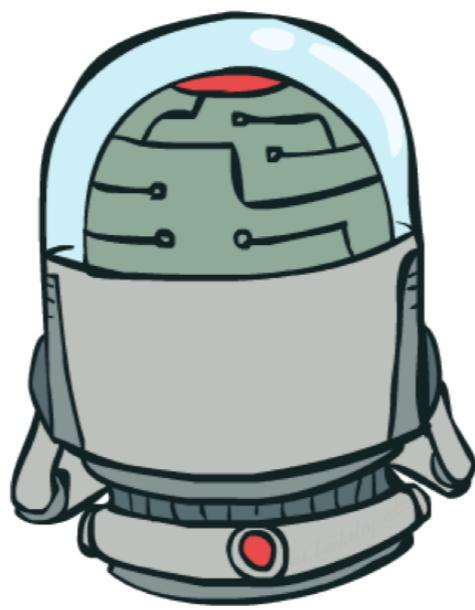


第六章·强化学习

- 规划 vs. 学习
- 强化学习
- 基于模型的学习和不基于模型的学习
- 被动学习和主动学习

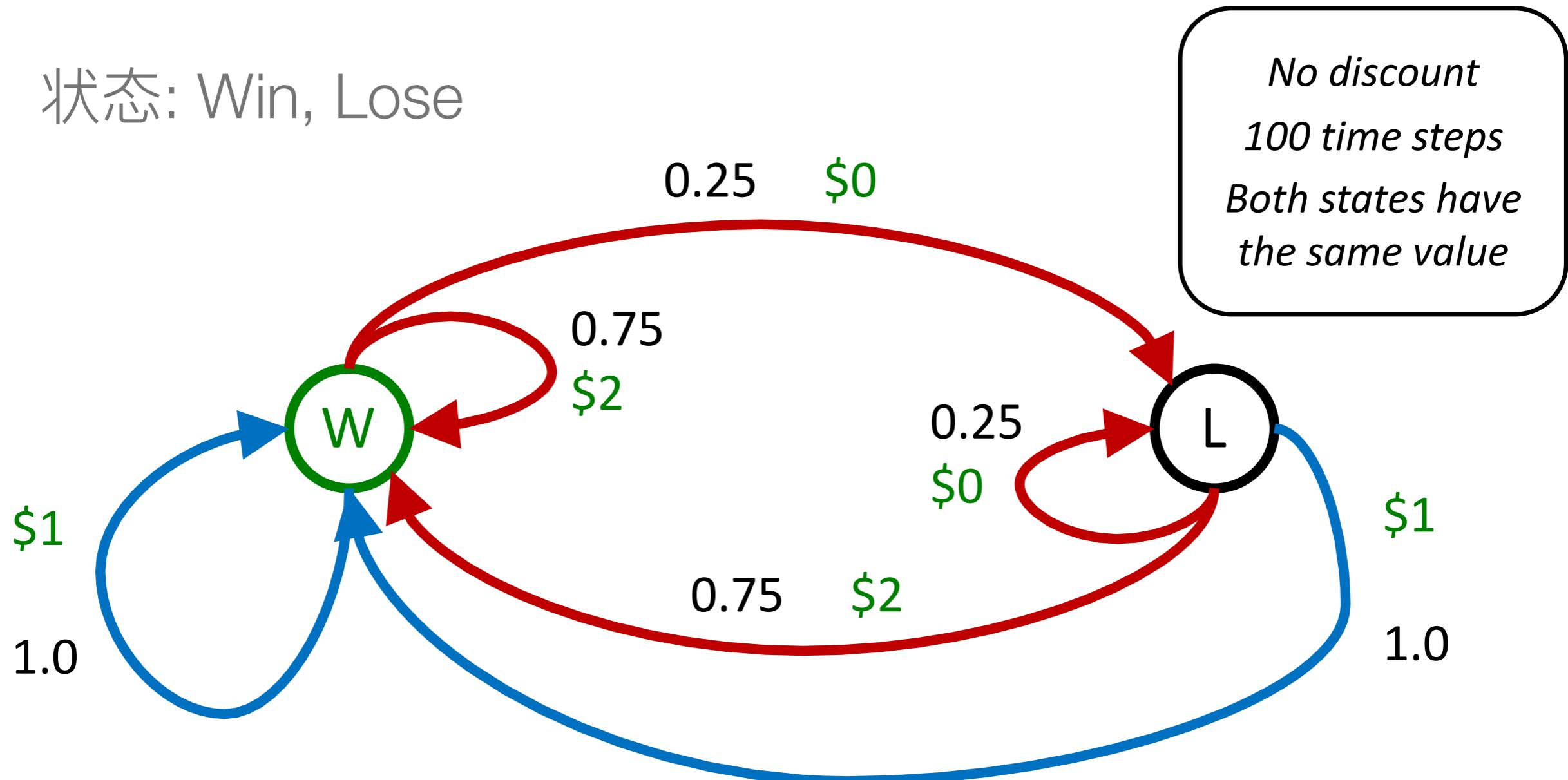


两个赌博机



两个赌博机 MDP

- 行动: Blue, Red
- 状态: Win, Lose



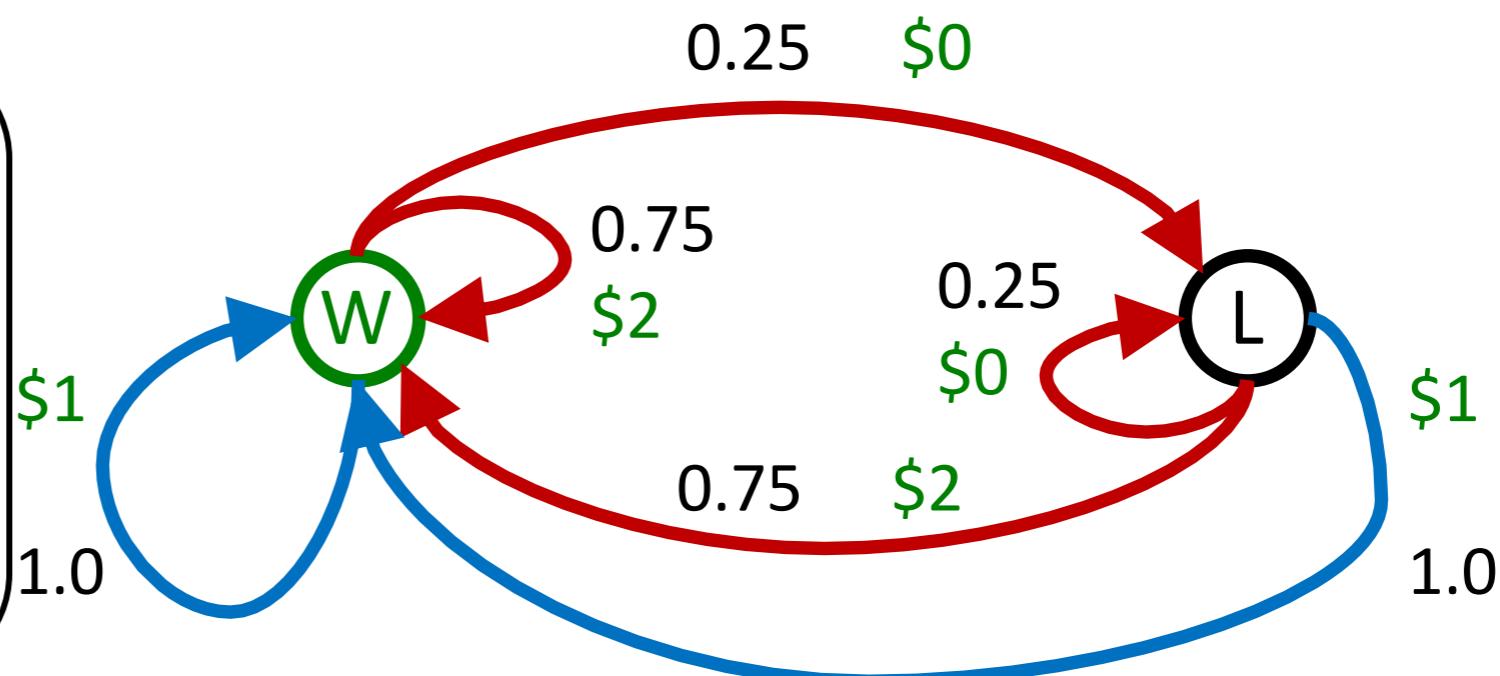
离线规划 Offline Planning

- 解决 MDPs 是在进行离线规划

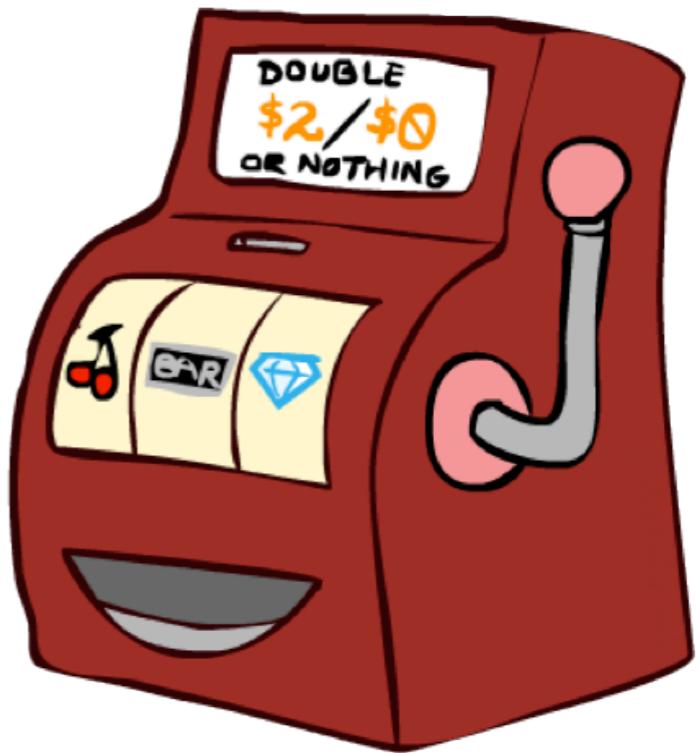
- 通过计算来确定所有的量
- 需要知道MDP的细节
- 并不是真的玩这个游戏！

*No discount
100 time steps
Both states have
the same value*

	Value
Play Red	150
Play Blue	100



Let's Play!

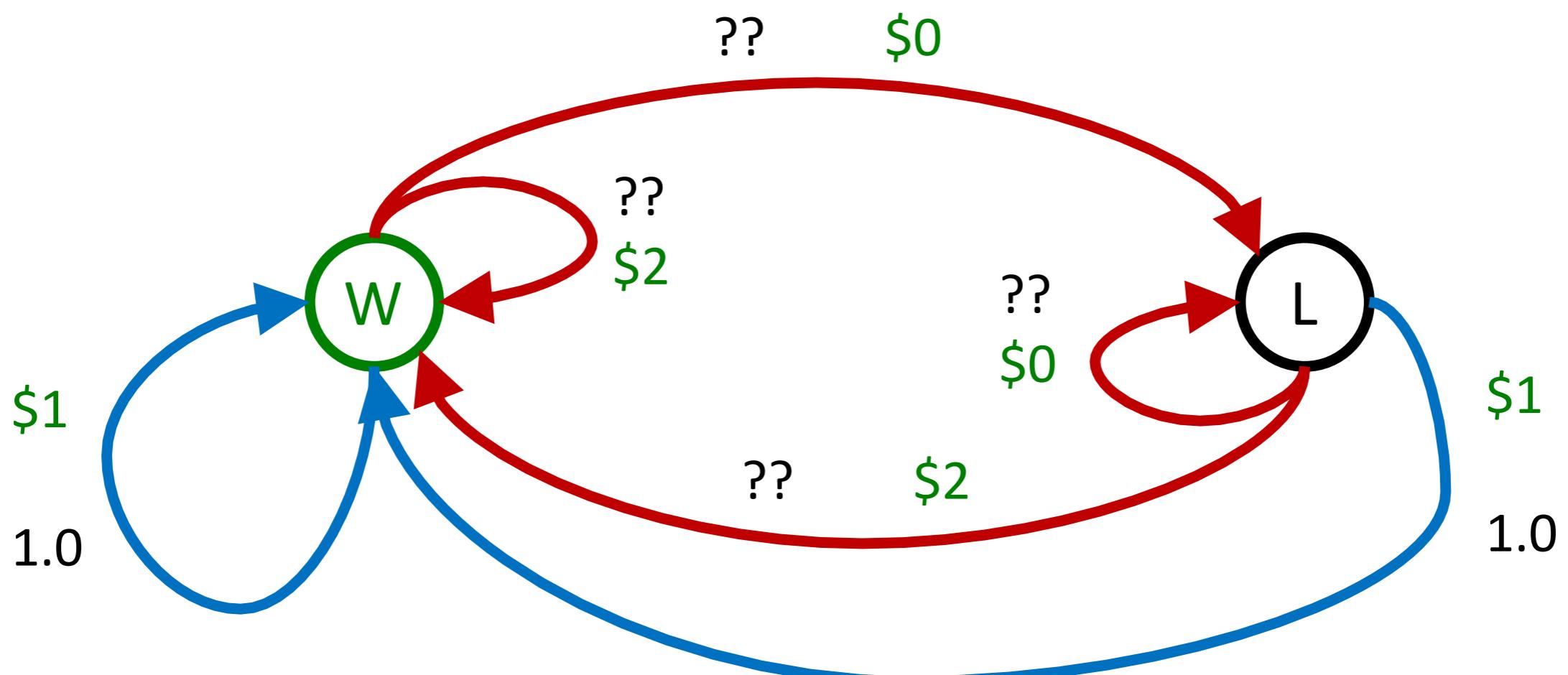


\$2 \$2 \$0 \$2 \$2

\$2 \$2 \$0 \$0 \$0

在线规划 Online Planning

- 红色的老虎机赢钱的概率未知.



Let's Play!



\$1 \$1 \$1 \$1 \$1



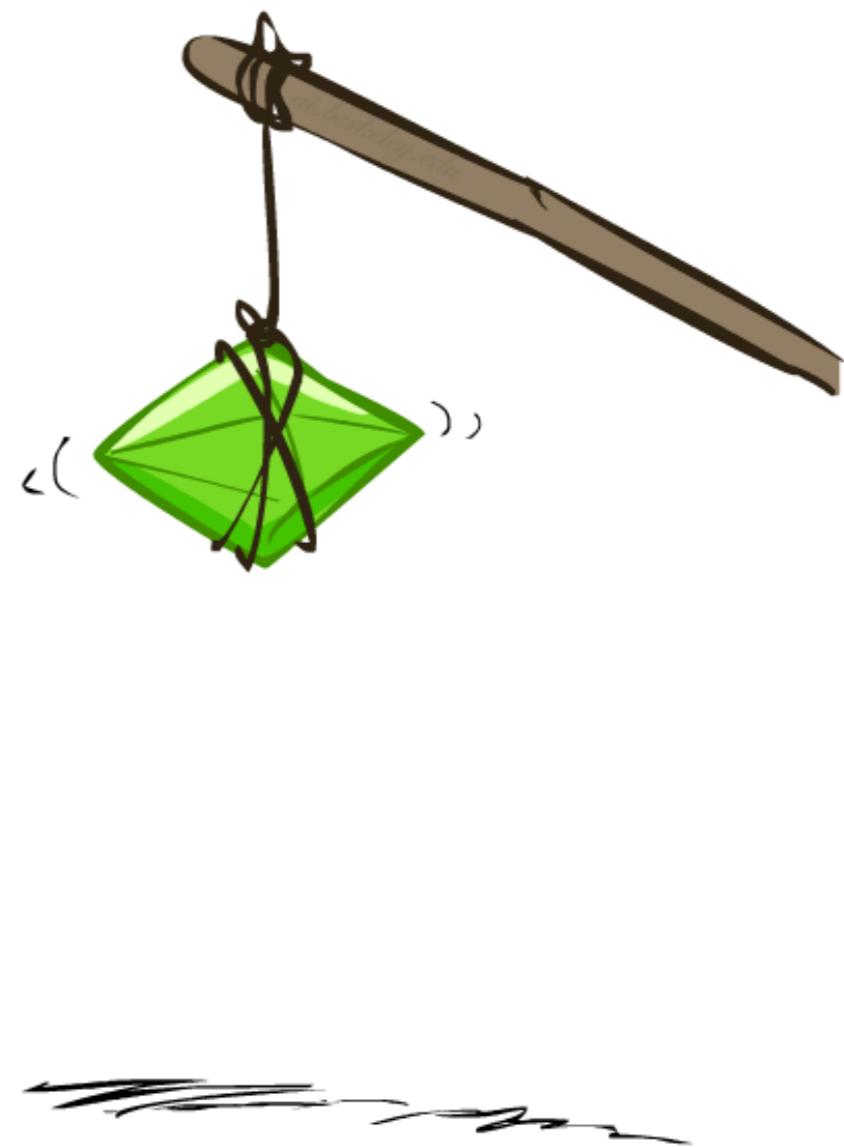
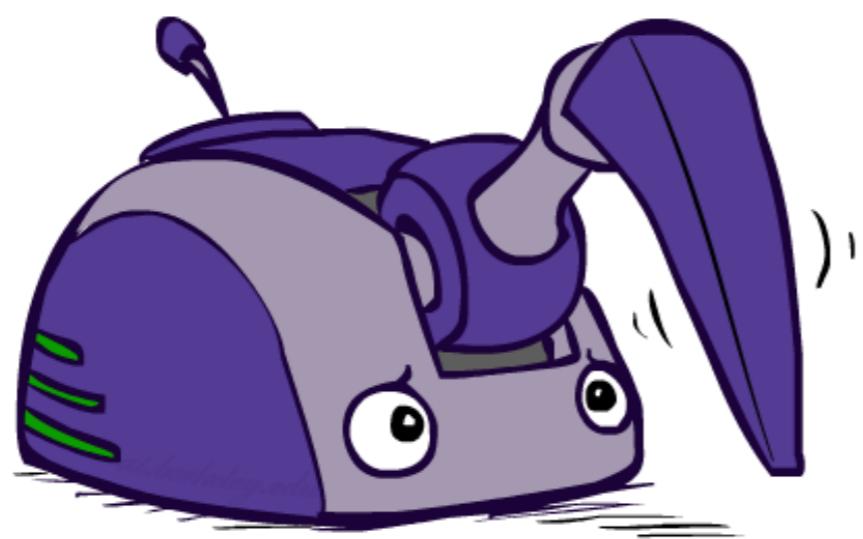
\$0 \$0 \$2 \$0 \$0

发生了什么变化?

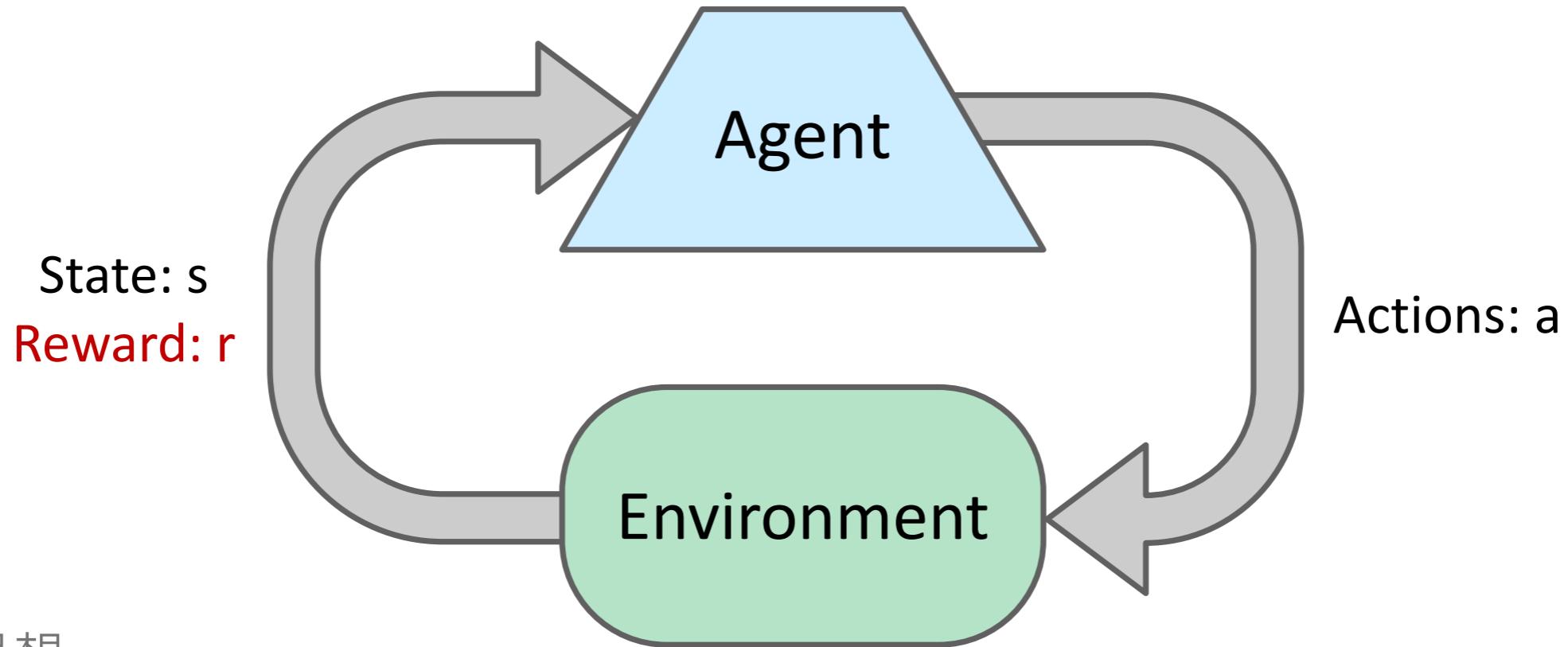
- 不再是规划，而是学习！
 - 具体是，属于增强学习reinforcement learning
 - 有一个MDP，但是你不能仅仅通过计算来解决它
 - 你需要采取实际行动来解决这个问题
- 增强学习中的一些基本思想
 - 探索 Exploration: 你必须尝试未知的行动来获取信息
 - 利用 Exploitation: 最终，你必须运用你所学习到的知识
 - 后悔 Regret: 即使你很聪明，你也会犯错
 - 采样 Sampling: 因为几率，你不得不反复尝试
- 学习可能比解决已知的MDP困难得多



强化学习



强化学习



- 基本思想:
 - 通过 奖赏值 的形式来获得反馈
 - 智能体的功效值是由 奖赏函数(reward function)来定义的
 - 必须学习如何行动，以获得最大化的期望奖赏值
 - 所有的学习是基于观察到的样本结果!

例子：学会走路



Initial



A Learning Trial



After Learning [1K Trials]

[Kohl and Stone, ICRA 2004]

例子：AIBO学走路



Initial

例子：AIBO学走路



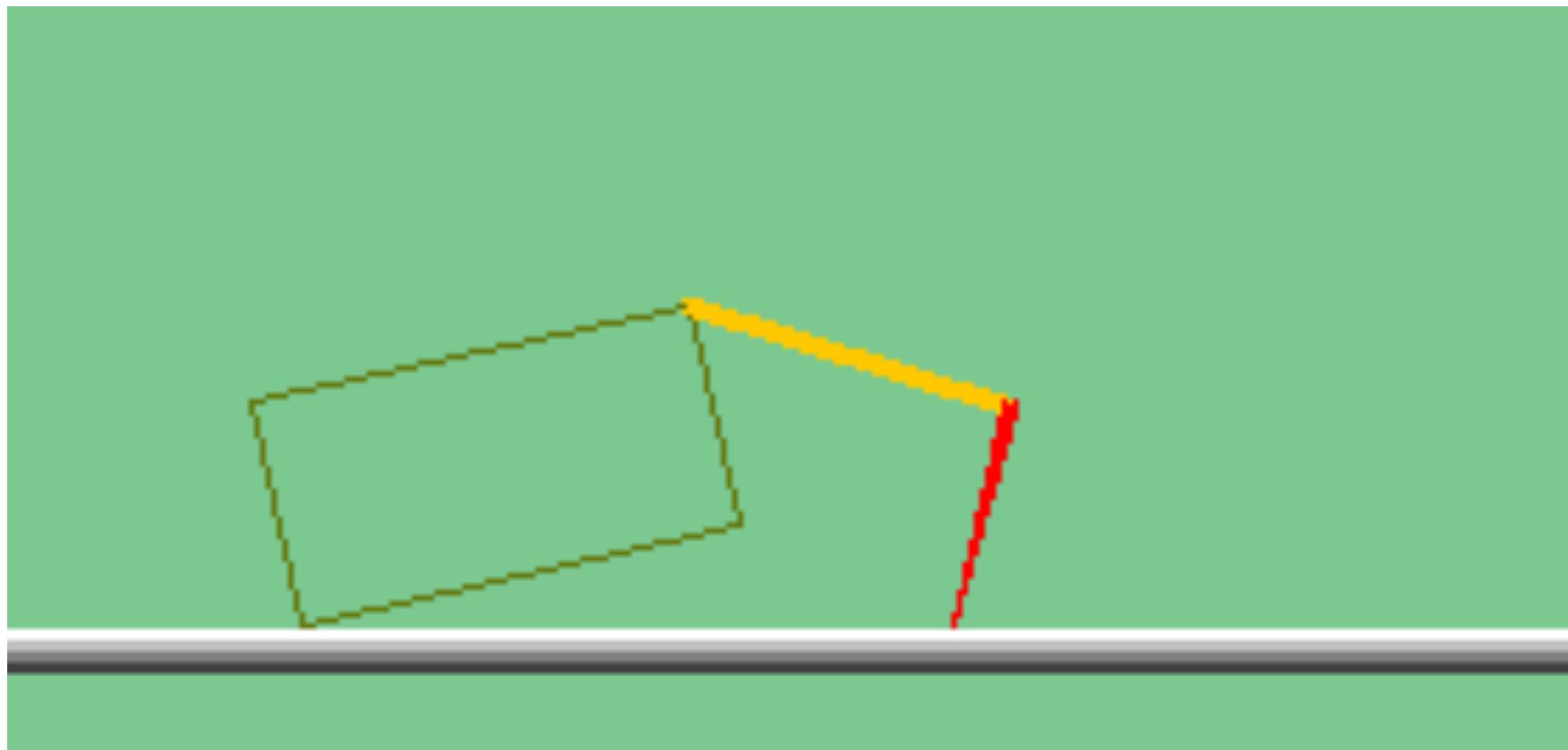
Training

例子：AIBO学走路



Finished

The Crawler!



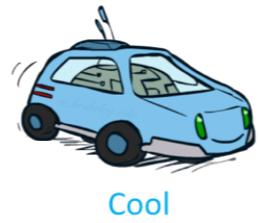
Video of Demo Crawler Bot



强化学习

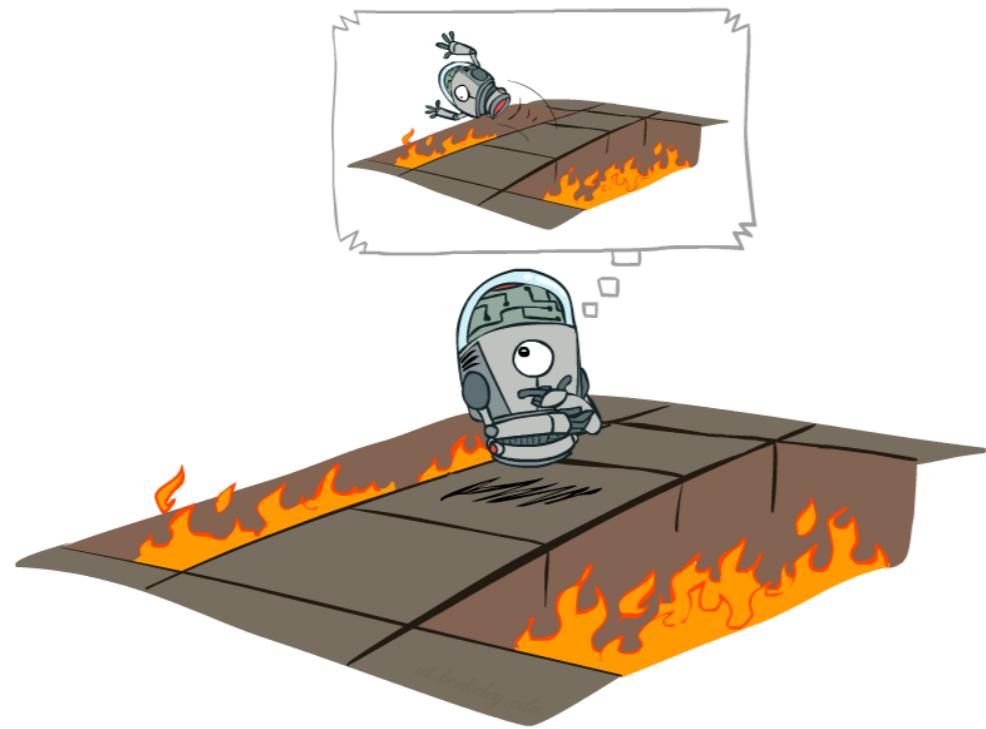
- 依然假设是一个马可夫决策过程(MDP):

- 一个 状态集合 $s \in S$
- 一个 行动集合 A
- 一个 转移模型 $T(s,a,s')$
- 一个 奖赏函数 $R(s,a,s')$

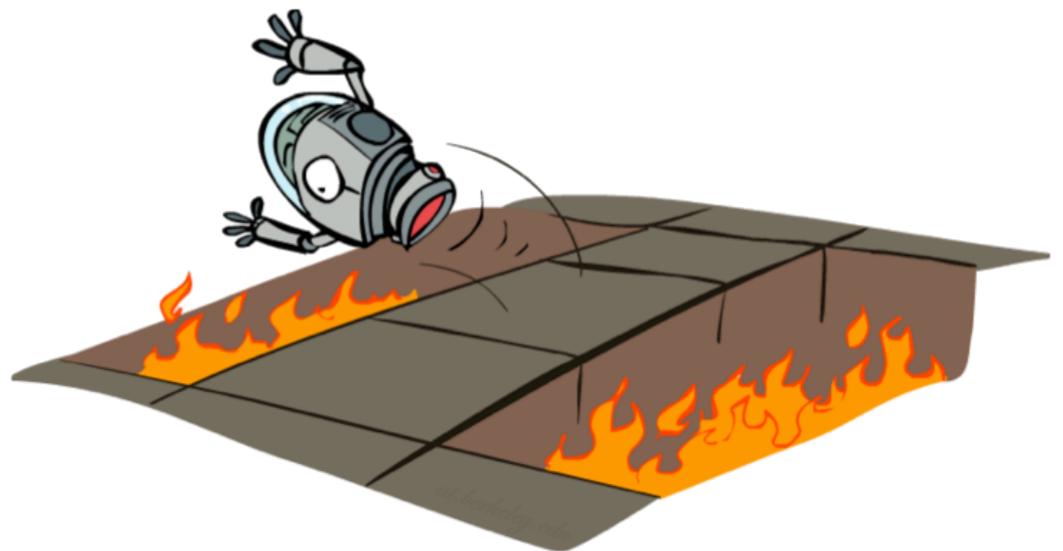


- 仍旧是搜寻一个策略 $\pi(s)$
- 与MDP不同的地方: 不知道 T 或 R
 - 不知道哪些状态是好的, 或哪些行动会带来什么
 - 必须在实践中尝试行动, 并从中进行学习

离线 (MDPs) vs. 在线 (RL)

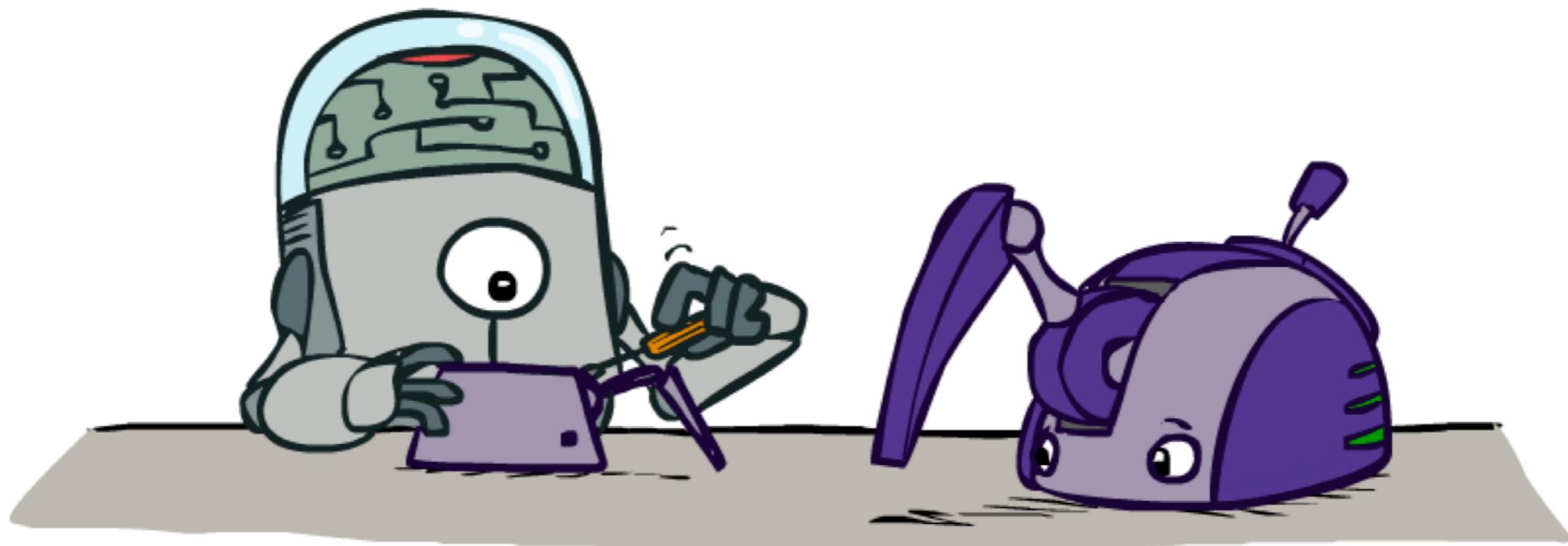


Offline Solution



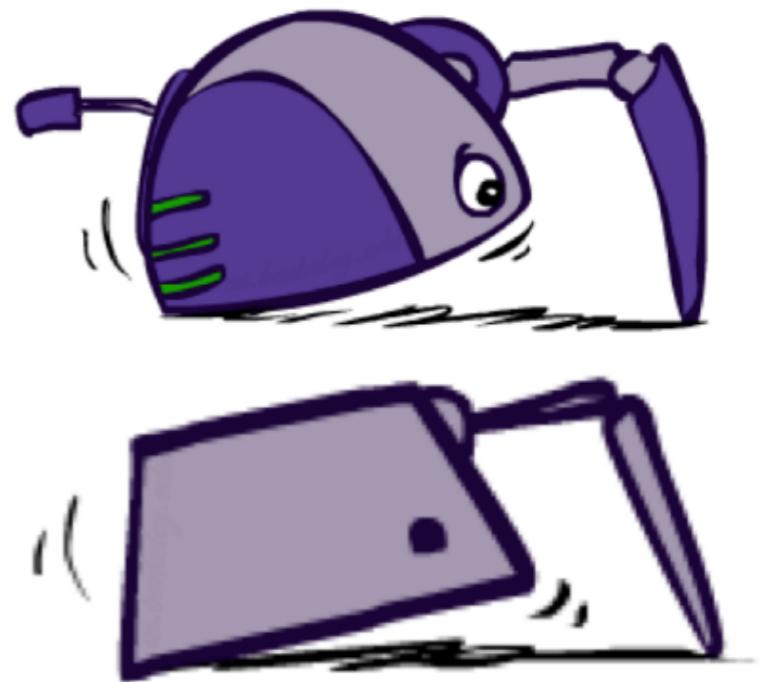
Online Learning

基于模型的学习 Model-Based Learning



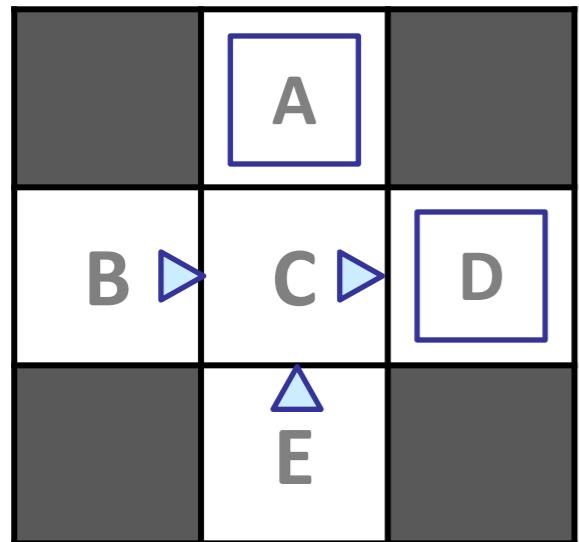
基于模型的学习

- 思想:
 - 基于经验学习一个近似的模型
 - 利用学好的模型去求解当前的MDP问题
- 步骤 1: 学习经验化的 MDP 模型
 - 对于每对 s, a , 数出结果是 s' 的数量
 - 归一化后给出一个估计 $\hat{T}(s, a, s')$
 - 当我们经历 (s, a, s') 后, 发现奖赏值 $\hat{R}(s, a, s')$
- 步骤 2: 求解基于学到的模型的 MDP
 - 例如, 使用状态值迭代法, 就像之前一样进行离线求解



举例：基于模型的学习

输入策略 π



Assume: $\gamma = 1$

观察到的(训练过程)

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

学到的模型

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

$T(B, \text{east}, C) = 1.00$
 $T(C, \text{east}, D) = 0.75$
 $T(C, \text{east}, A) = 0.25$

...

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

$R(B, \text{east}, C) = -1$
 $R(C, \text{east}, D) = -1$
 $R(D, \text{exit}, x) = +10$

...

举例：年龄期望值

目标：计算班上同学的年龄期值

Known P(A)

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

如果事先没有 P(A)，那么可以进行采样 $[a_1, a_2, \dots, a_N]$

Unknown P(A): “Model Based”

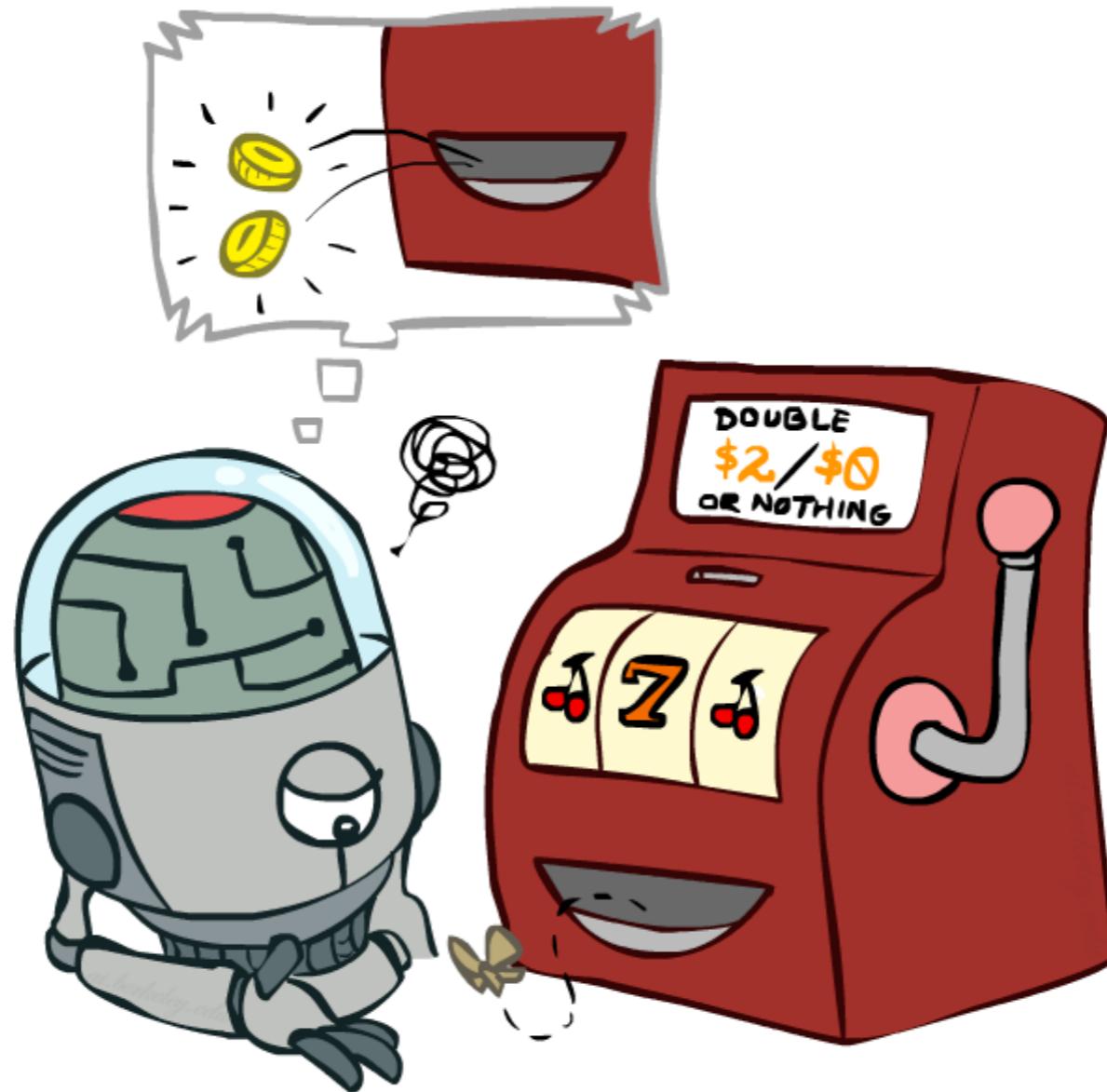
$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$

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

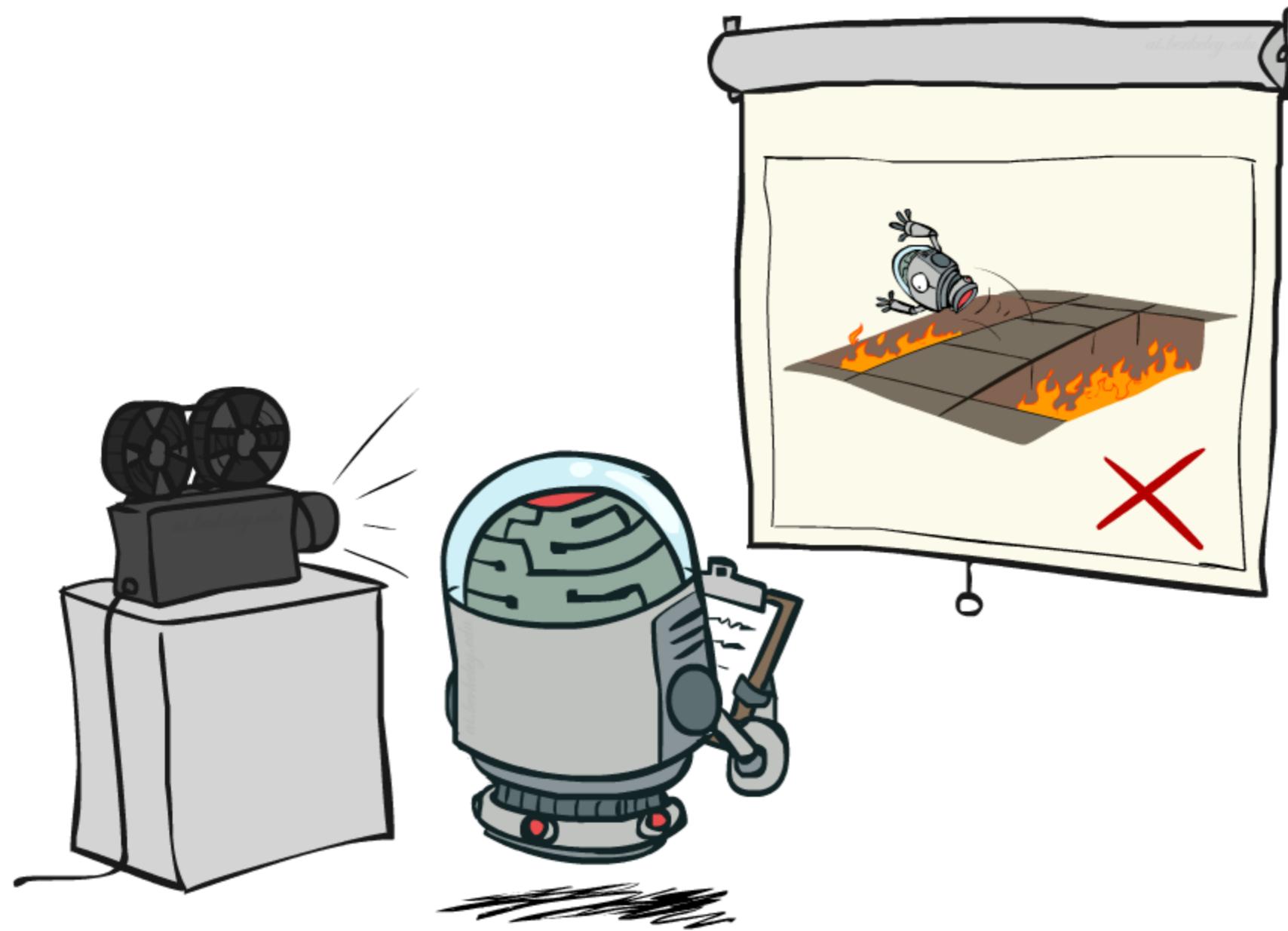
Unknown P(A): “Model Free”

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

不基于模型的学习 Model-Free Learning



被动强化学习 Passive Reinforcement Learning



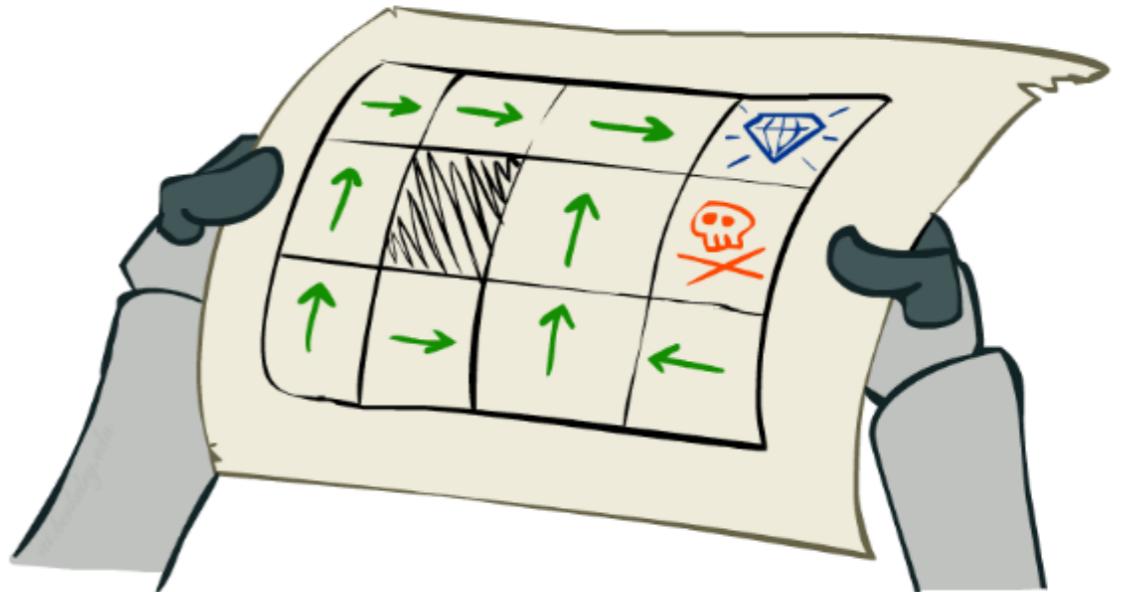
被动(passive)强化学习

- 简单的任务: 策略评价 policy evaluation

- 输入: 一个给定的策略 $\pi(s)$
- 不知道转移概率 $T(s,a,s')$
- 不知道奖赏函数 $R(s,a,s')$
- 目标: 计算出状态值 (state values)

- 在这种情况下:

- 学习者需要“顺其自然”
- 智能体自身没有行动的选择权
- 只执行策略, 并从经验中学习
- 这并不是线下规划! 你在世界上采取真正的行动。



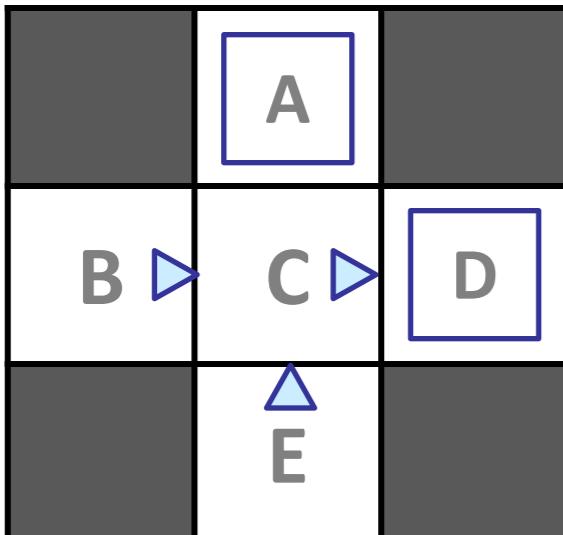
直接估值

- 目标: 在策略 π 下, 计算每个状态的值
- 思想: 对每个样本进行估值, 然后再平均化所有样本的值
 - 按照策略 π 进行行动
 - 每次你访问到一个状态, 记录下沿路折扣后的奖赏值之和
 - 再平均化那些样本估值
- 这就叫做直接估值



举例：直接估值

输入策略 π



Assume: $\gamma = 1$

观察到的样本 (训练)

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

输出的状态值

	-10		
A	+4	+10	
B	+8	C	D
	-2	E	

直接估值的问题所在

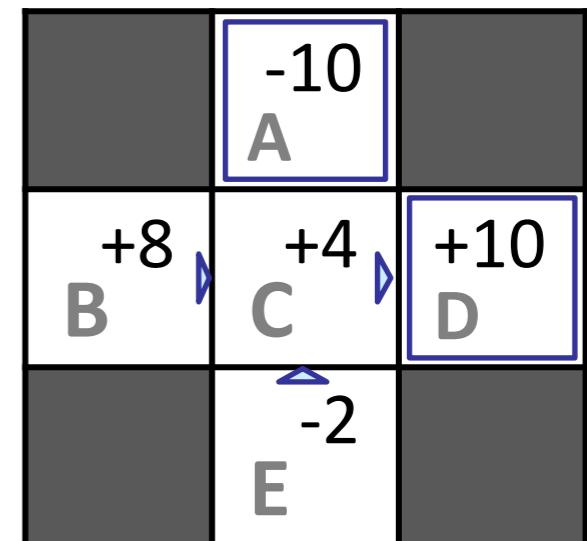
- 其优势?

- 容易理解
- 不需要知道 T, R
- 最终计算正确的状态值, 通过大量样本

- 其劣势?

- 浪费了关于状态之间连接的信息
- 每个状态值必须分开单独学习
- 所以, 将花费很长的时间去学习

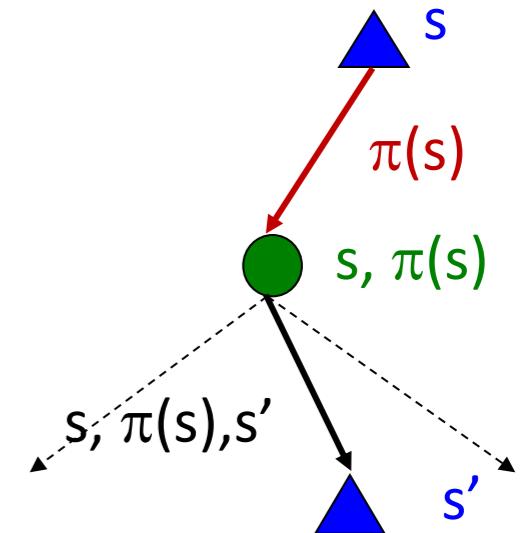
输出的状态值



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

为什么不能直接用策略评价?

- 给定一个策略, 根据Bellman更新公式计算 V-值:
 - 每一轮, 用向前一步更新计算结果替换当前的 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节点间的连接关系
 - 但是, 我们需要 T 和 R 才能计算!
- 关键问题: 如何在不知道T和R的情况下, 更新 V-值?
 - 换句话说, 如何在不知道权值的情况下, 估计一个加权均值?



能否用基于样本的策略评价？

- 我们想通过计算下面这些加权均值改进对 V 值的估计：

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

- 思想：通过行动进行采样，得到结果状态 s' ，然后进行均值

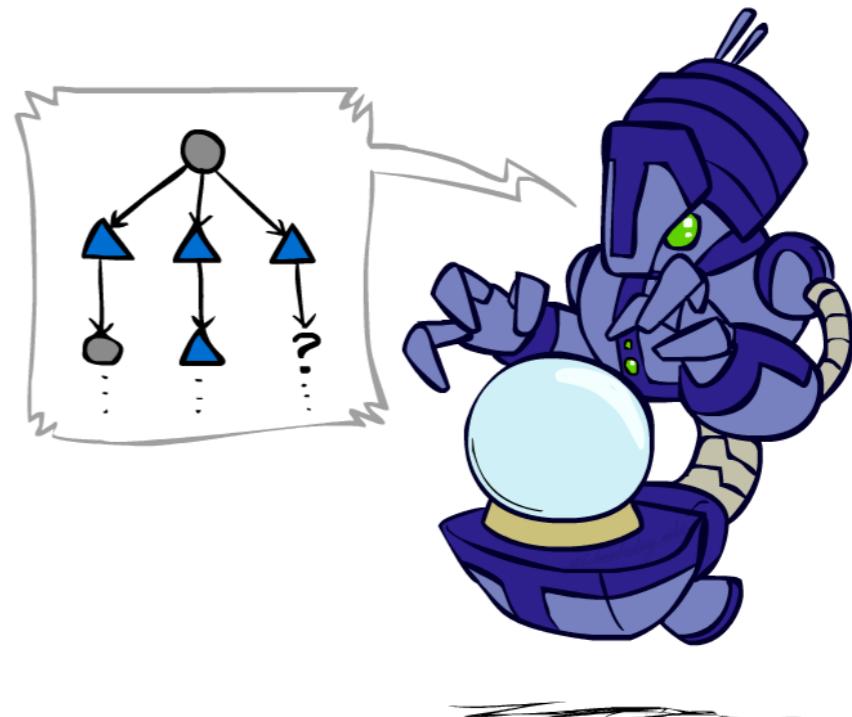
$$\text{sample}_1 = R(s, \pi(s), s'_1) + \gamma V_k^{\pi}(s'_1)$$

$$\text{sample}_2 = R(s, \pi(s), s'_2) + \gamma V_k^{\pi}(s'_2)$$

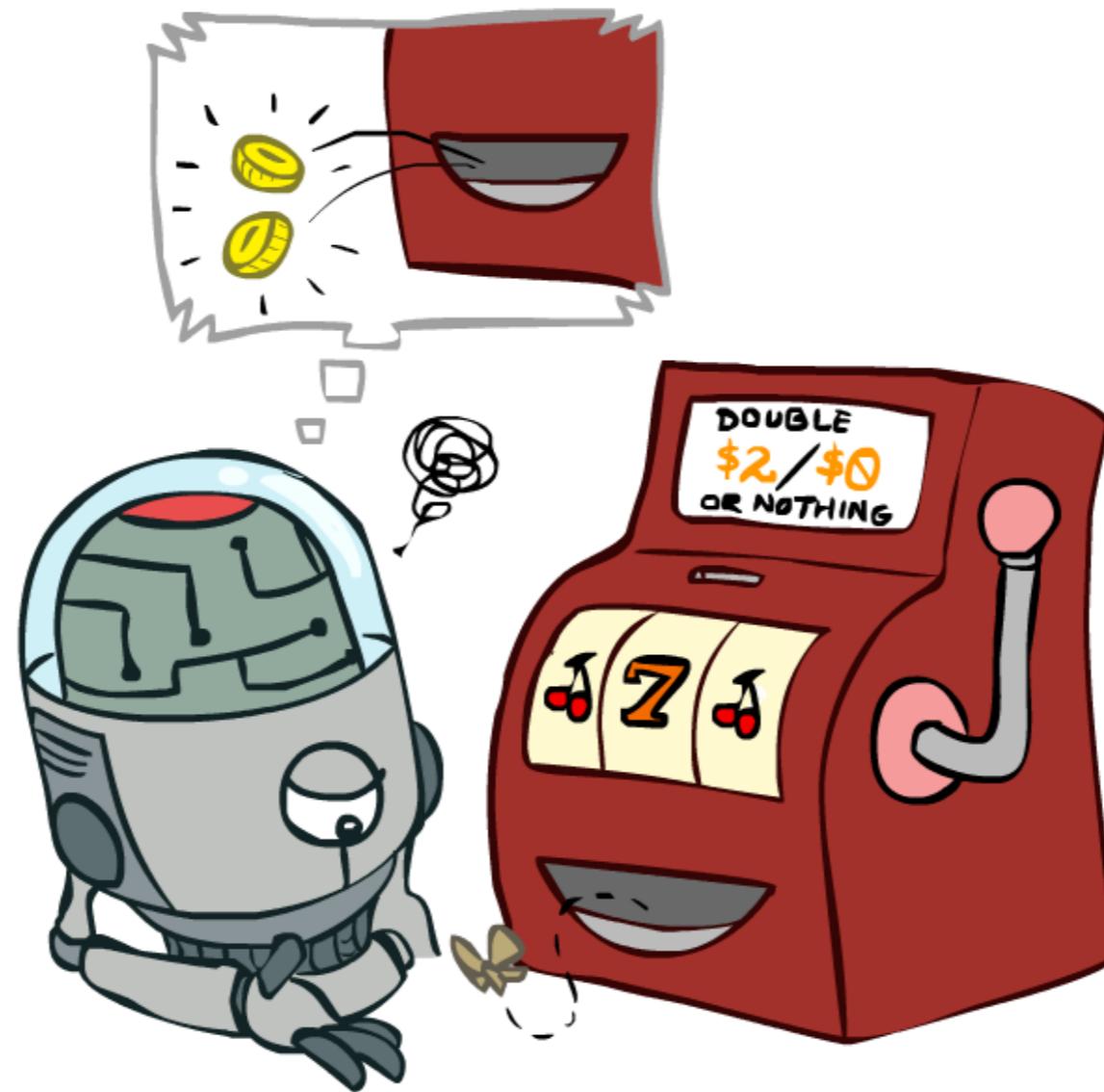
...

$$\text{sample}_n = R(s, \pi(s), s'_n) + \gamma V_k^{\pi}(s'_n)$$

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

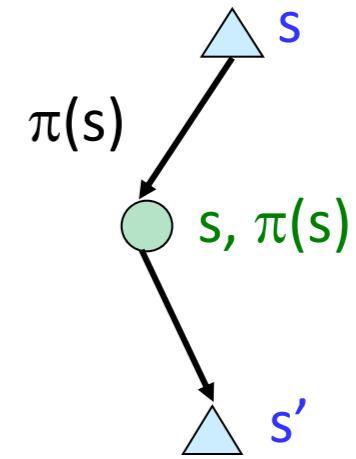


时间差分学习 Temporal Difference Learning



时间差分学习

- 基本思想: 从每一次经验中学习!
 - 每经历一次状态转移 (s, a, s', r) , 就更新一下 $V(s)$
 - 逐渐地, 结果状态 s' 将会越来越多地参与到更新过程中
- V 值的时间差分学习
 - 固定的策略, 还是在做策略评价!
 - 把 V 值逐渐地向最终结果的值方向移动: 逐渐在计算均值



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

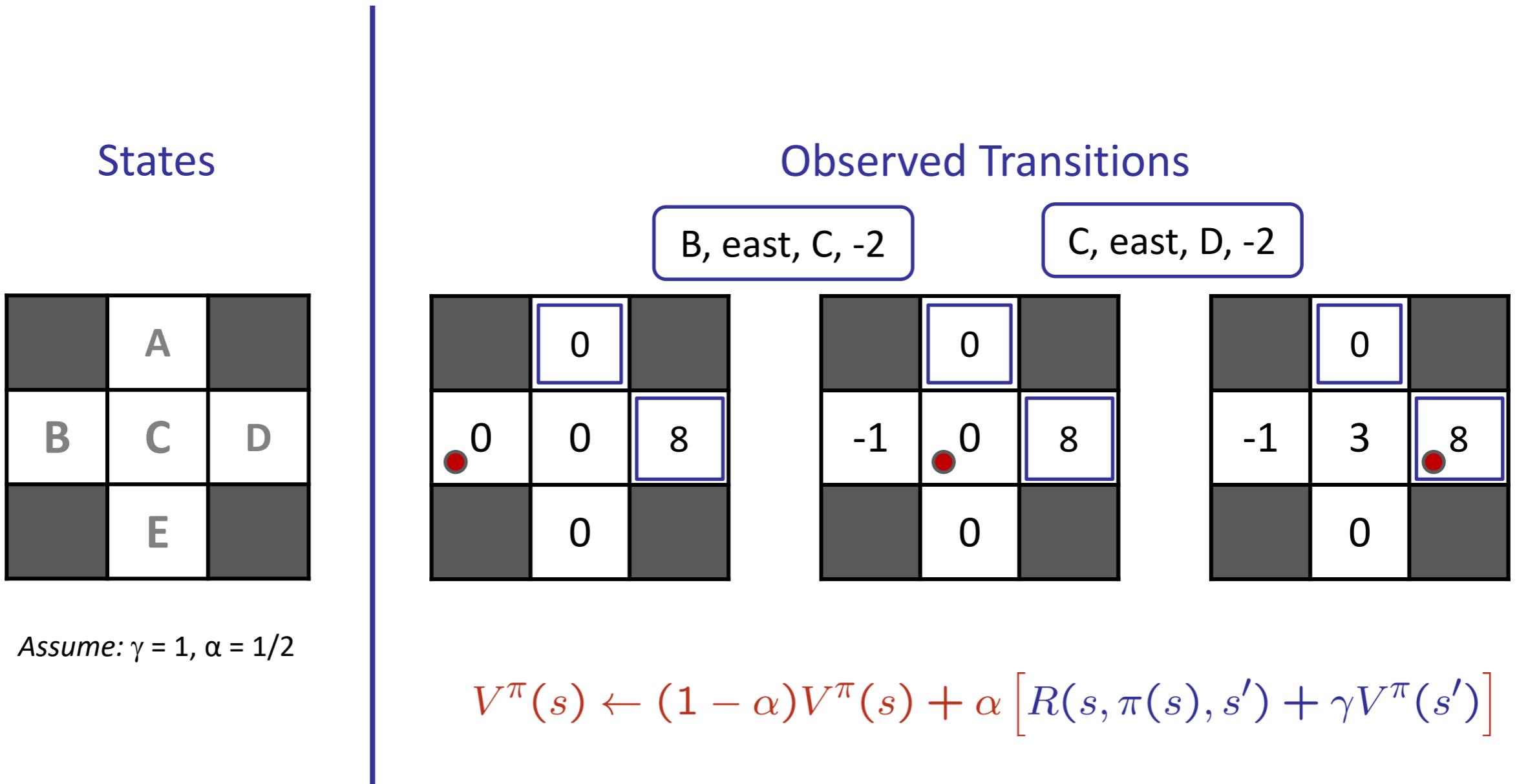
Update to $V(s)$: $V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + (\alpha)sample$

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

指数移动平均法 Exponential Moving Average

- 指数移动平均
 - 插值更新法: $\bar{x}_n = (1 - \alpha) \cdot \bar{x}_{n-1} + \alpha \cdot x_n$
 - 使最近的样本更重要
- $$\bar{x}_n = \frac{x_n + (1 - \alpha) \cdot x_{n-1} + (1 - \alpha)^2 \cdot x_{n-2} + \dots}{1 + (1 - \alpha) + (1 - \alpha)^2 + \dots}$$
- 逐渐遗忘过去的值(distant past values were wrong anyway)
- (逐渐) 降低学习率参数 alpha 能够使该均值收敛

举例：时间差分学习法



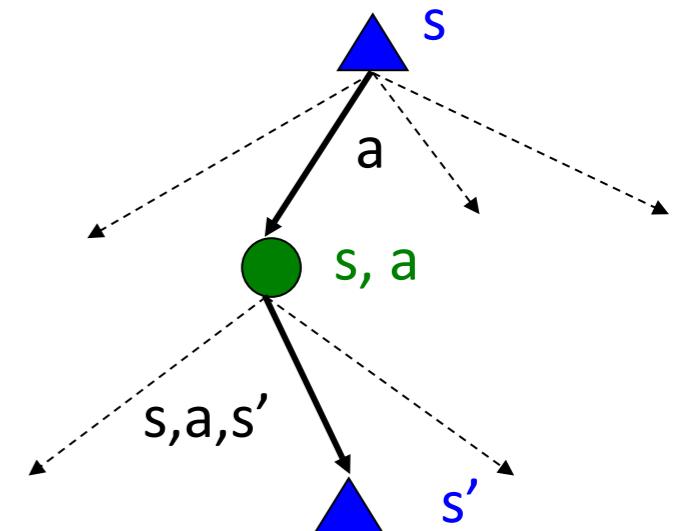
学习到状态值以后呢?

- 时间差分状态值学习 是一种model-free的策略评价方法.通过计算样本均值,模仿Bellman的更新,
- 然而, 如果想要把状态值转换为(新的)策略, 就变得很难了:

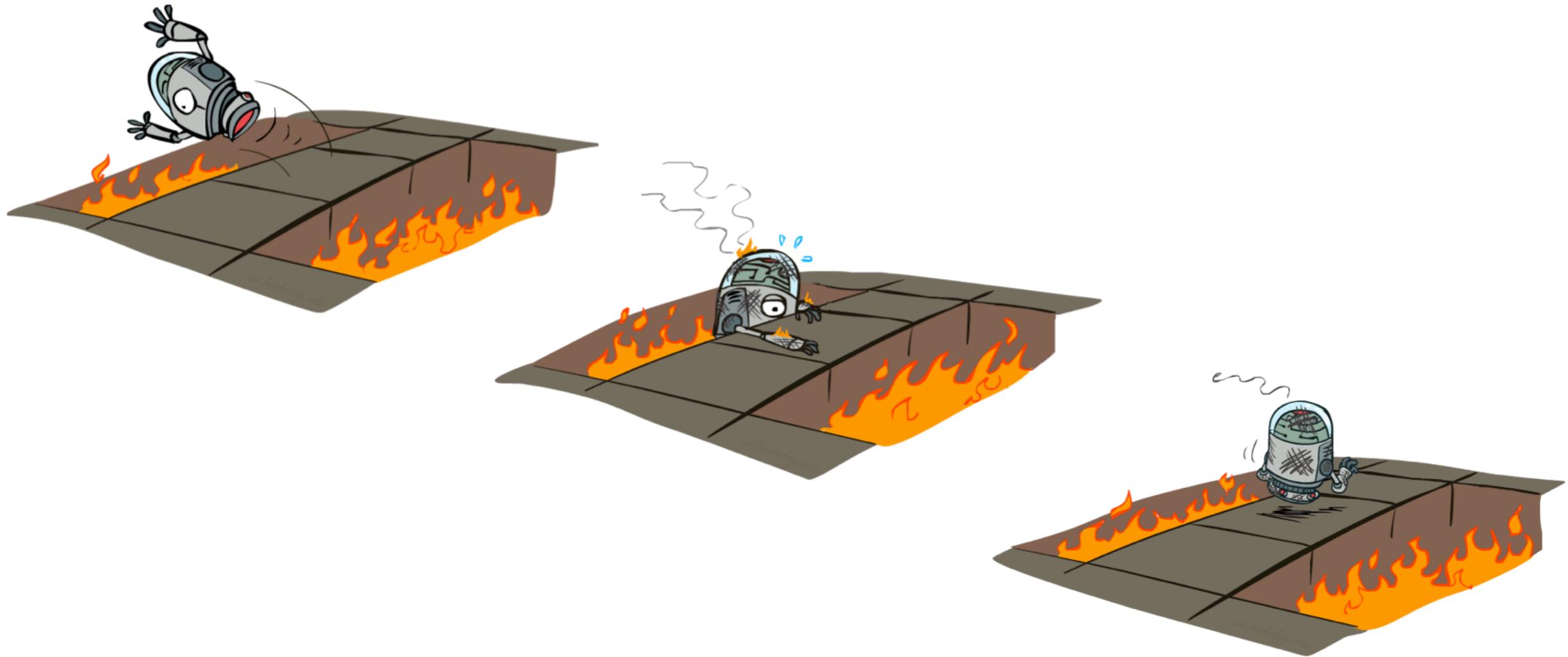
$$\pi(s) = \arg \max_a Q(s, a)$$

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

- 因此: 应该学习 Q-值, 而不是V-值
- 从而使行动的选择变得也是model-free的!



主动强化学习 Active Reinforcement Learning



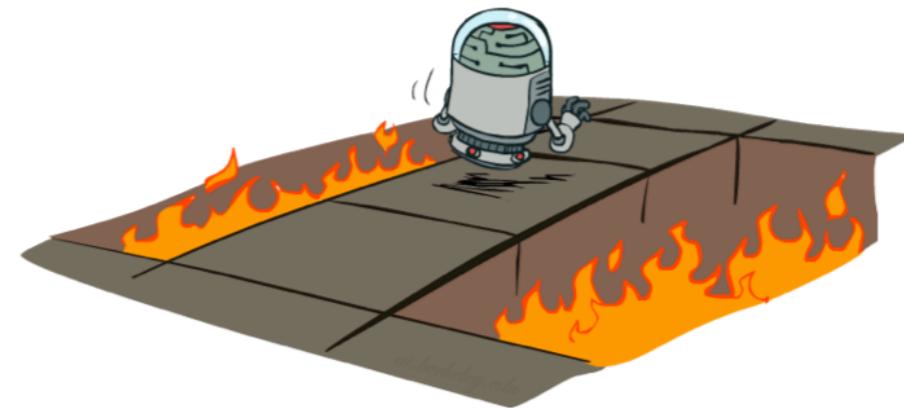
主动强化学习

- 完整的强化学习: 找到最优策略

- 不知道转移概率 $T(s,a,s')$
 - 不知道奖赏函数 $R(s,a,s')$
 - 现在可以自主选择行动
 - 目标: 学习最优策略 / 状态值

- 在这种情况下:

- 智能体决定做什么(行动)!
 - 基本的平衡制约: 探索 vs. 利用
 - 这并不是离线规划! You actually take actions in the world and find out what happens...



MDP的V值迭代和Q值迭代

- 状态值迭代: 找到后继(有限深度的)值

- 初始化 $V_0(s) = 0$

- 给定 V_k , 计算深度为 $k+1$ 的值:

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

- 但是 Q-值更有用, 所以转而计算它们

- 初始化 $Q_0(s, a) = 0$

- 给定 Q_k , 计算深度为 $k+1$ 的 Q 值:

$$Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma \max_{a'} Q_k(s', a')]$$

Q值学习 Q-Learning

- Q-Learning: 基于样本的 Q值迭代

$$Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$$

- 在实践过程中学习 $Q(s, a)$ 的值

- 获得一个样本 (s, a, s', r)

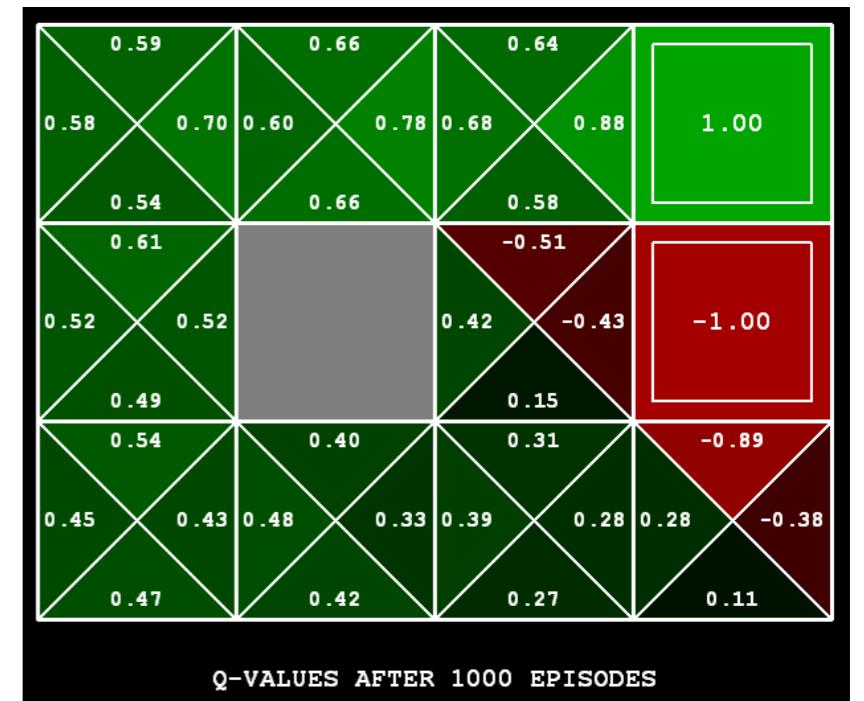
- 考虑旧的估值: $Q(s, a)$

- 考虑新的估值:

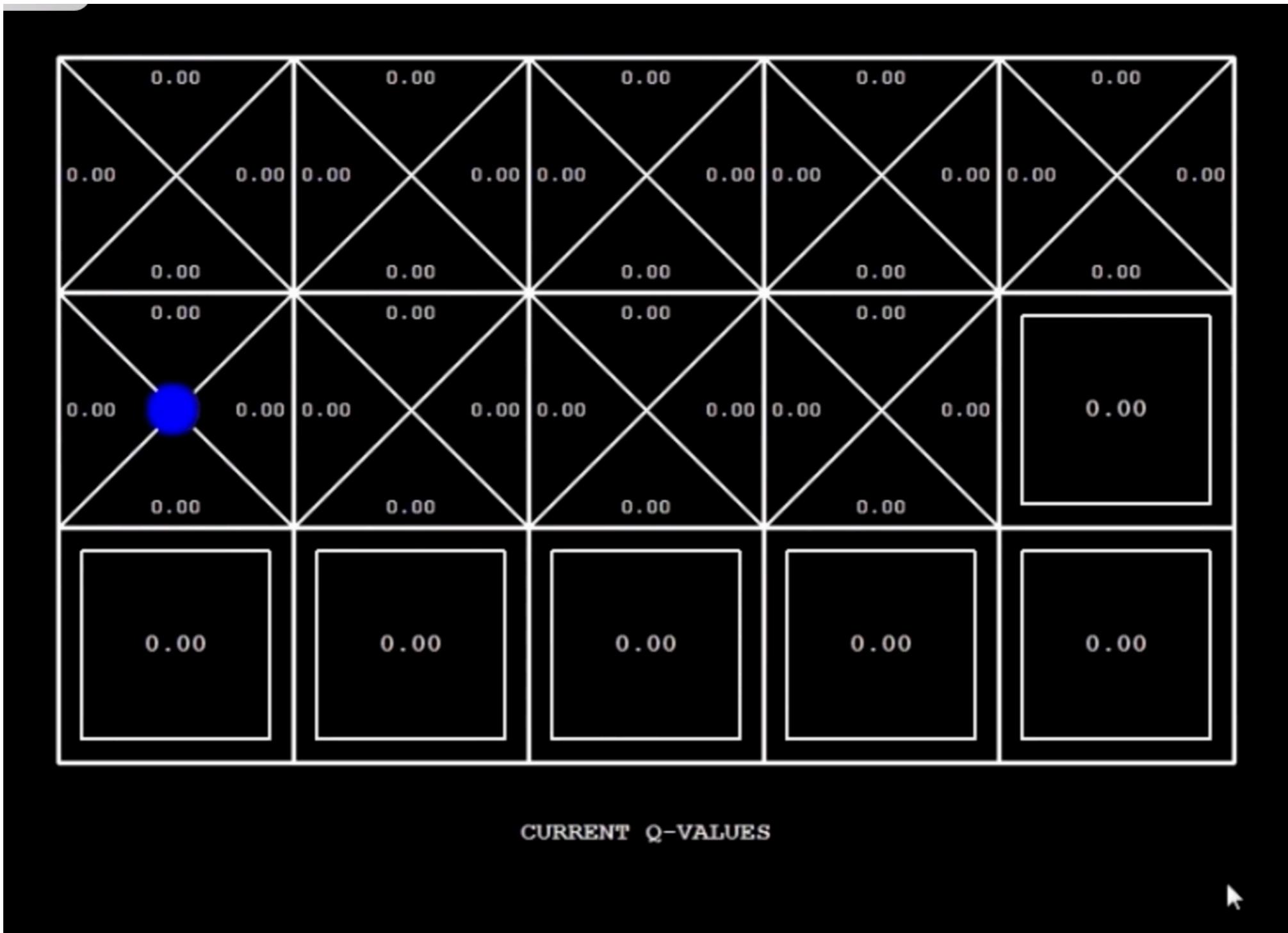
$$sample = R(s, a, s') + \gamma \max_{a'} Q(s', a')$$

- 把新的估值结合起来, 计算实时均值更新:

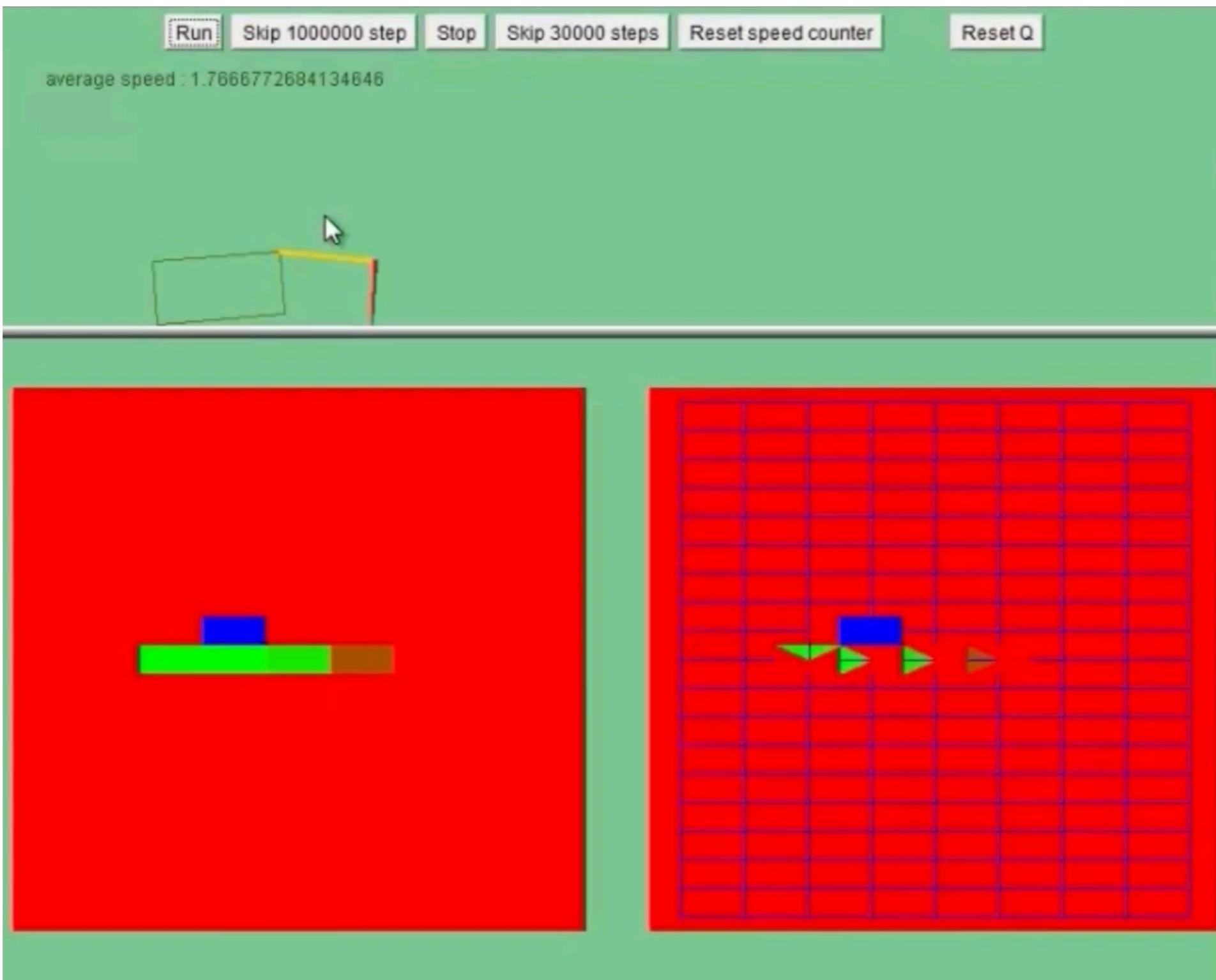
$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + (\alpha) [sample]$$



视频演示 Q-Learning -- Gridworld

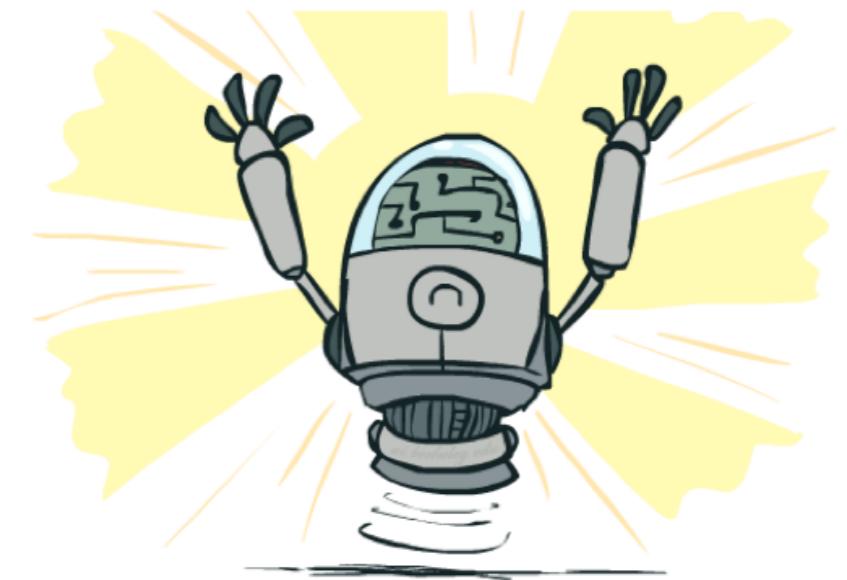


视频演示 Q-Learning -- Crawler

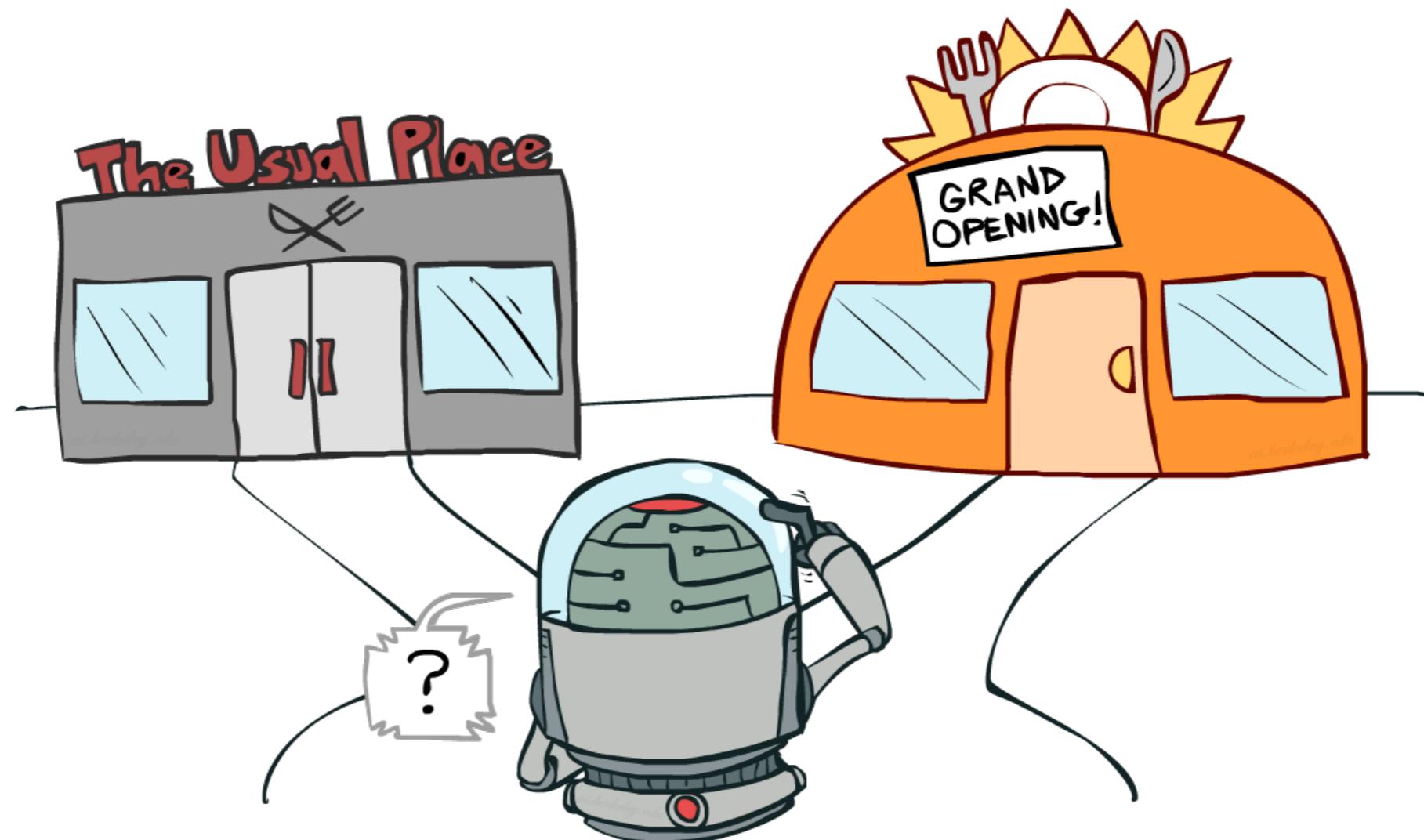


Q-Learning 性质

- Q-learning 收敛于最优策略 – 即便在实践采样过程中行动都不是最优的!
- 这叫做 没有策略的学习 off-policy learning
- 需要注意:
 - 需要充分探索
 - 最终要调整learning rate 到足够小
 - ... 但也不能减小太快
 - 学习过程中如何选择行动无关紧要(!)



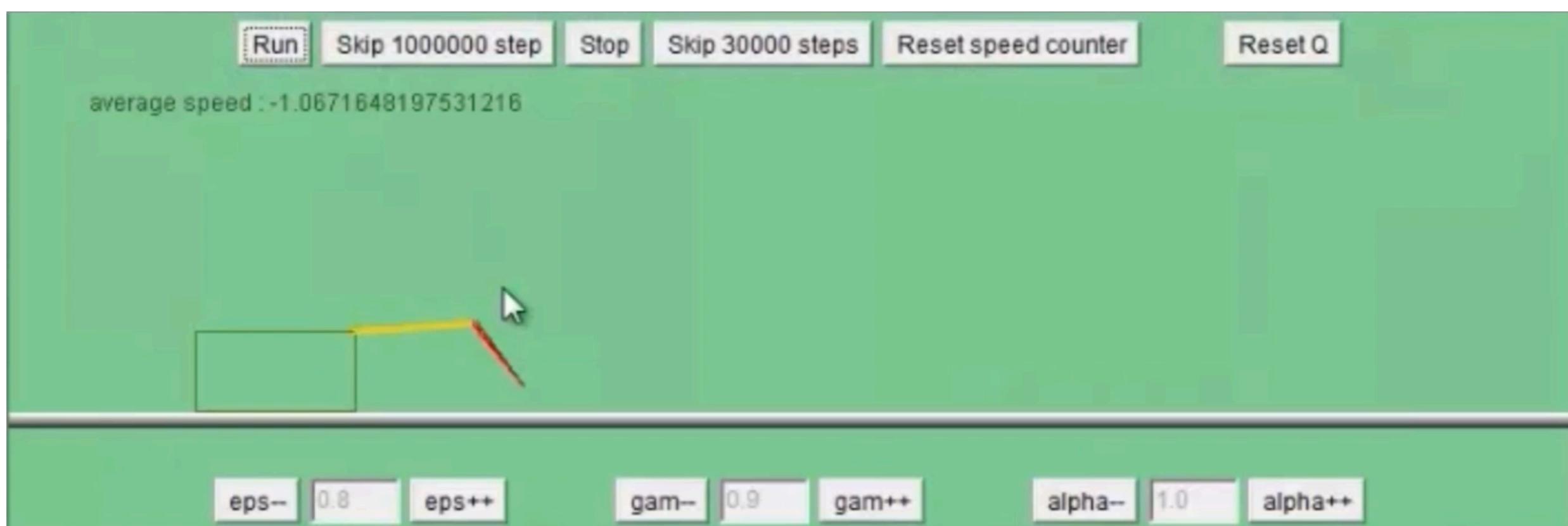
Exploration探索 vs. Exploitation利用



如何探索?

- 几种方法用来执行探索
 - 最简单的: 随机选择行动 (ϵ -greedy)
 - 每个时间步, 都生成一个随机数
 - 以一个比较小的 ϵ 概率, 执行随机的行动
 - 以一个比较大的 $1-\epsilon$ 概率, 执行当前策略
 - 随机选择行动的问题所在?
 - 即使学习已经完成了, 仍然会持续探索整个空间
 - 一个解决方法: 逐渐减小 ϵ
 - 另一个解决方法: 使用探索函数

视频演示 Q-learning – Epsilon-Greedy – Crawler



探索函数 Exploration Functions

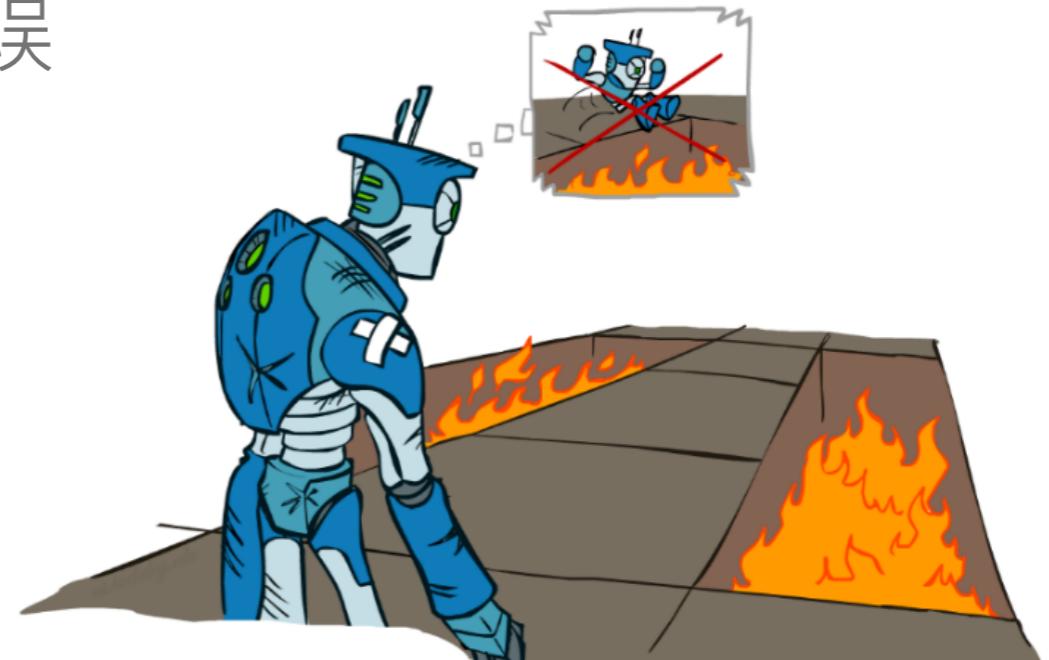
- 何时探索?
 - 基本的策略: 探索一个确定的步数
 - 更好的想法: 只探索那些未知的地方, 最终停止探索
- Exploration function 探索函数
 - 两个输入, 一个是估计的Q值 u , 另一个是访问Q状态的次数 n , 返回乐观效用. $f(u, n) = u + k/n$
 - 正常的 Q-Update: $Q(s, a) \leftarrow_{\alpha} R(s, a, s') + \gamma \max_{a'} Q(s', a')$
 - 修改的 Q-Update: $Q(s, a) \leftarrow_{\alpha} R(s, a, s') + \gamma \max_{a'} f(Q(s', a'), N(s', a'))$

视频演示Q-learning – 探索函数 – Crawler

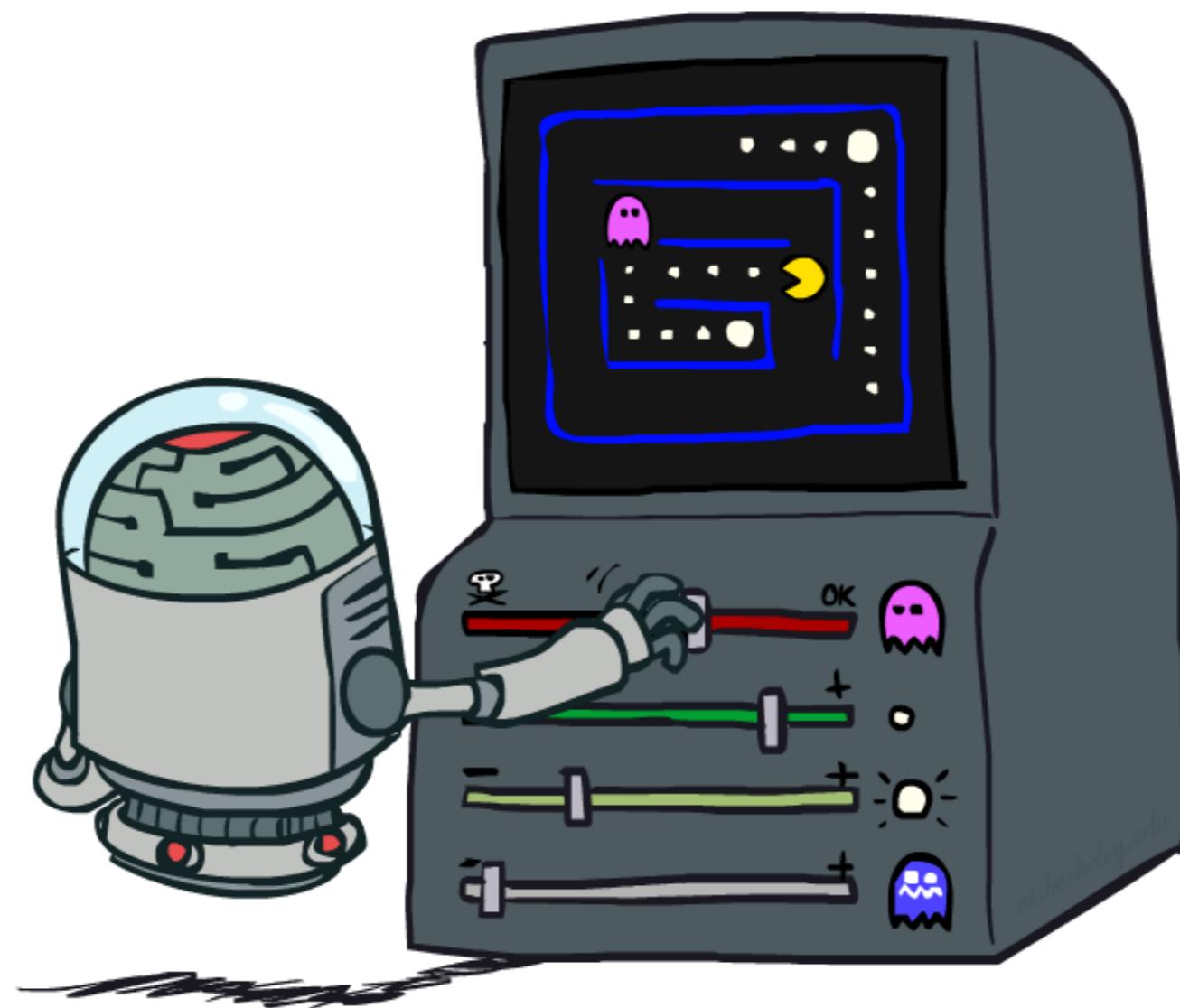


后悔 Regret

- 在学习最优策略的过程中，会犯错误
(选错行动，导致不好的结果)
- Regret 度量你所犯错误导致的代价之和：既你的（预期的）回报和最佳（预期的）回报之间的差异
- 最小化 regret 本身涉及到学习优化策略以外的事 – 需要你的学习方法（算法）本身也是最优化的
- 例如：随机探索算法random exploration 和 利用探索函数exploration functions的算法，都可以找到最优行动策略，但是前者有更大的regret

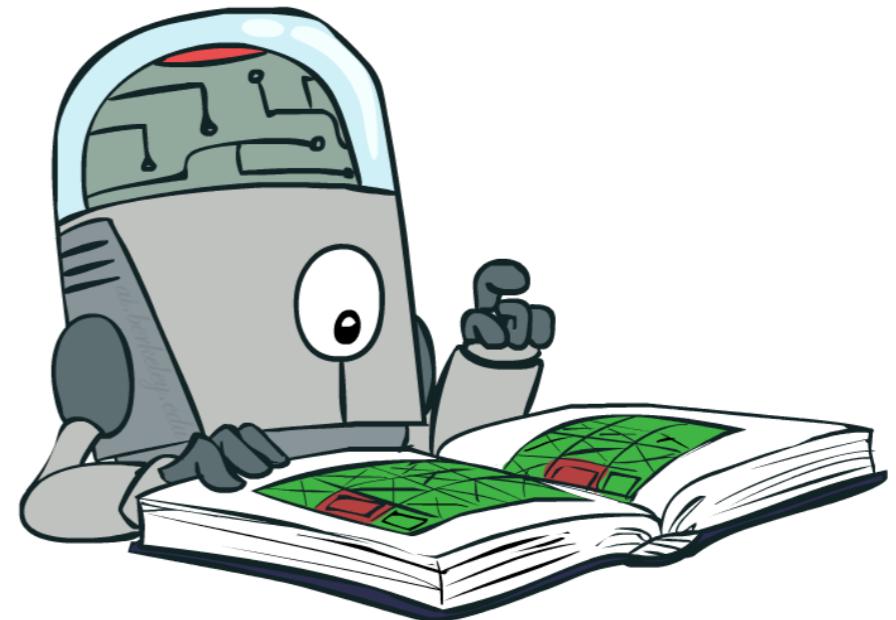


近似Q学习 Approximate Q-Learning



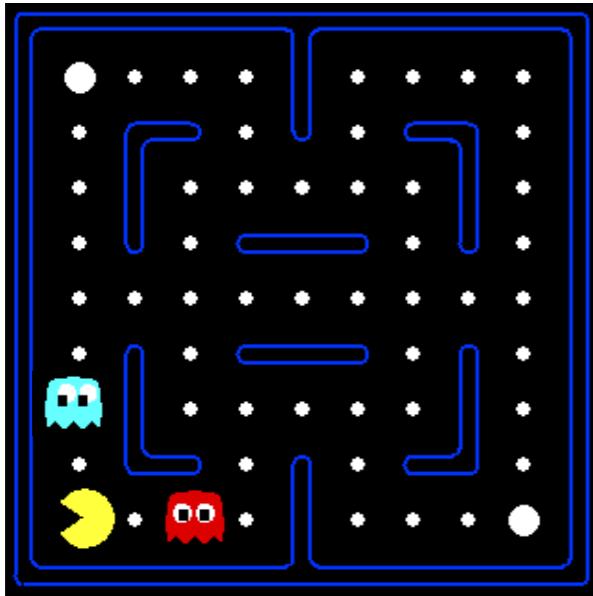
状态的泛化

- 基本Q-Learning保存了一张所有Q值的表格
- 在现实情境中, 不可能学习每个状态!
 - 有太多的状态需要学习
 - Q值表格太大, 内存放不下
- 因而, 我们希望能够泛化:
 - 对一小部分状态进行学习
 - 将学习到的经验泛化到新的, 近似的情况

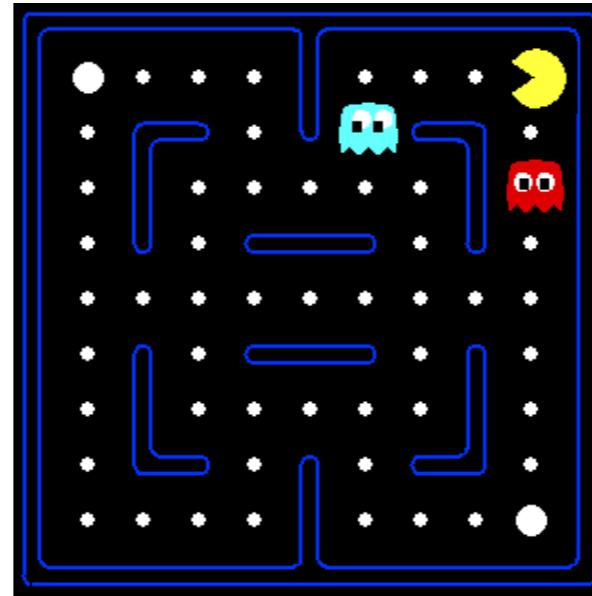


示例: Pacman

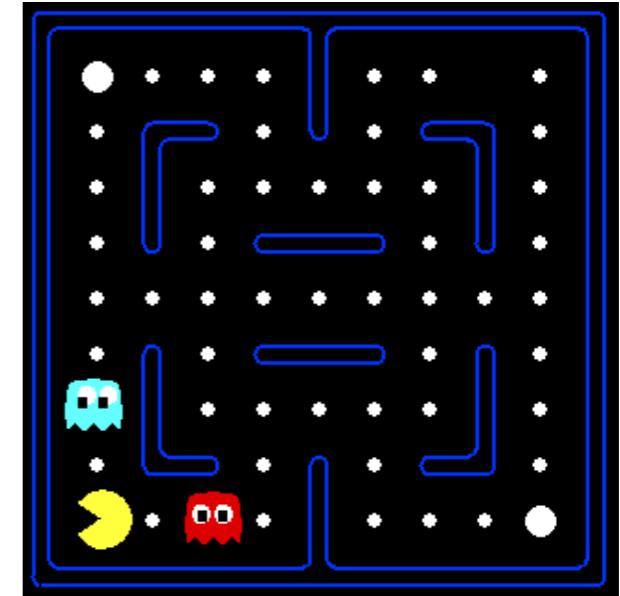
假设我们通过经验发现这个状态是糟糕的:



在基本的Q-Learning中，我们对这种状态一无所知：

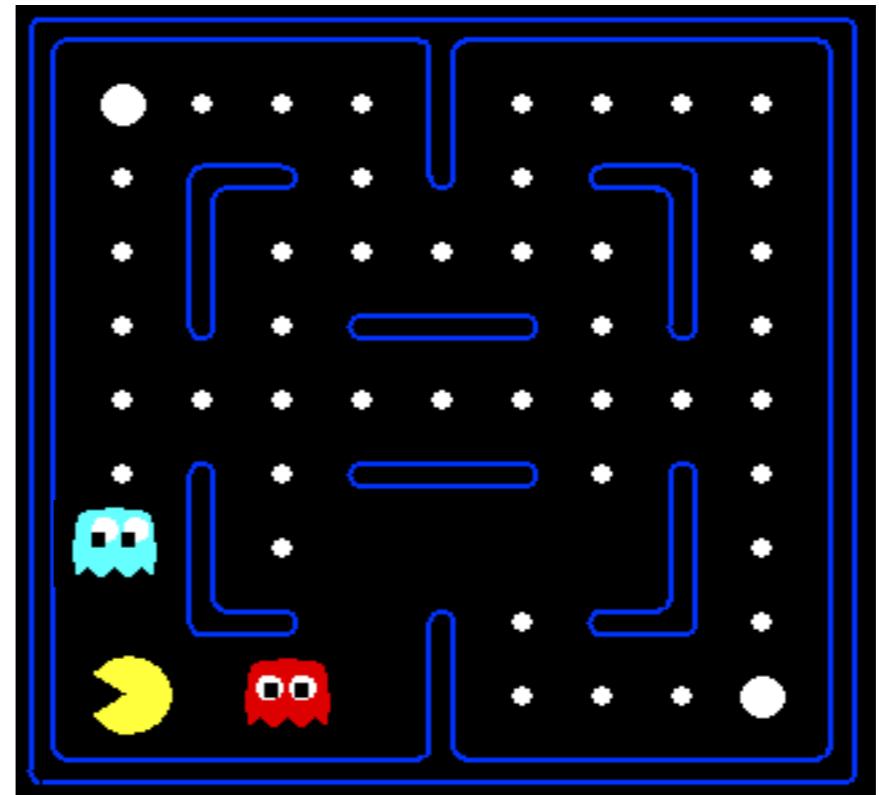


甚至是这个状态！



基于特征的状态表达

- 解决方案：使用一组特征（属性）描述状态
 - 特征是从状态到实数（通常为0/1）的函数，它们捕捉状态的重要属性
 - 特征示例：
 - Pacman到最近的鬼魂的距离
 - Pacman到最近的豆子的距离
 - 鬼魂的数量
 - $1 / (\text{到最近的豆子的距离})^2$
 - Pacman是否在一个“隧道”里？(0/1)
 -
 - 这能描述有图的确切状态吗？
 - 也可以用特征来描述一个 q-state (s, a)



特征值的线性拟合

- 使用特征和权重的组合, 我们可以给任何一个状态写出一个 V 值函数, 或者是 Q 值函数:

$$V(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$$

$$Q(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a) + \dots + w_n f_n(s, a)$$

- 优势: 学习的内容变成了几个权重数值
- 劣势: 各个状态虽然共享了特征, 但是取值仍然十分不同!

Approximate 近似 Q-Learning

$$Q(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a) + \dots + w_n f_n(s, a)$$

- 用线性 Q-functions 表示 Q-learning :

transition = (s, a, r, s')

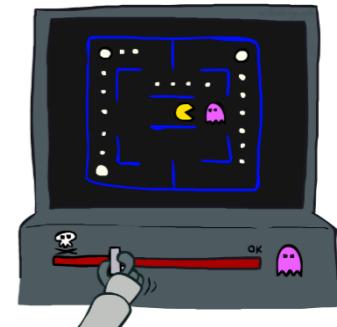
difference = $[r + \gamma \max_{a'} Q(s', a')] - Q(s, a)$

$Q(s, a) \leftarrow Q(s, a) + \alpha [\text{difference}]$

Exact Q's

$w_i \leftarrow w_i + \alpha [\text{difference}] f_i(s, a)$

Approximate Q's



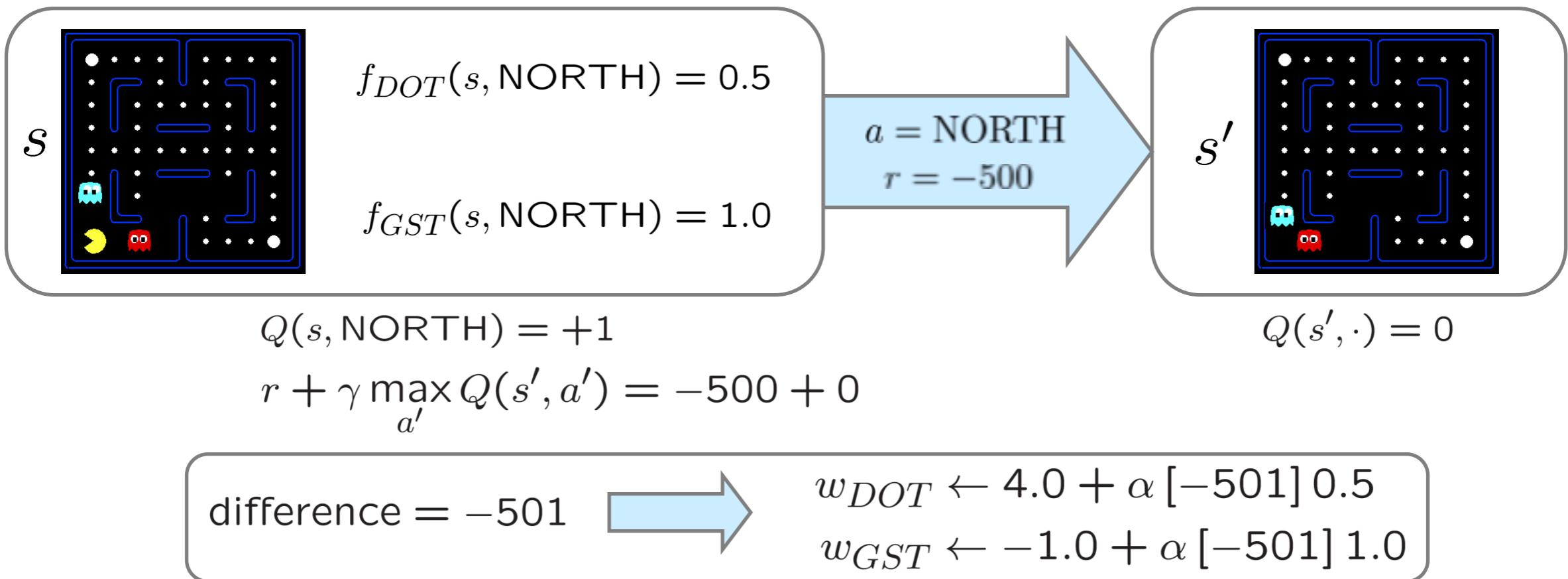
- 简单的解释:

- 调整激活的特征值的权重

- 例如, 如果一些不符合预知的坏事情发生了, 降低当前状态的特征权重, 从而影响到所有当前状态有着相同特征的状态

举例: Q-Pacman

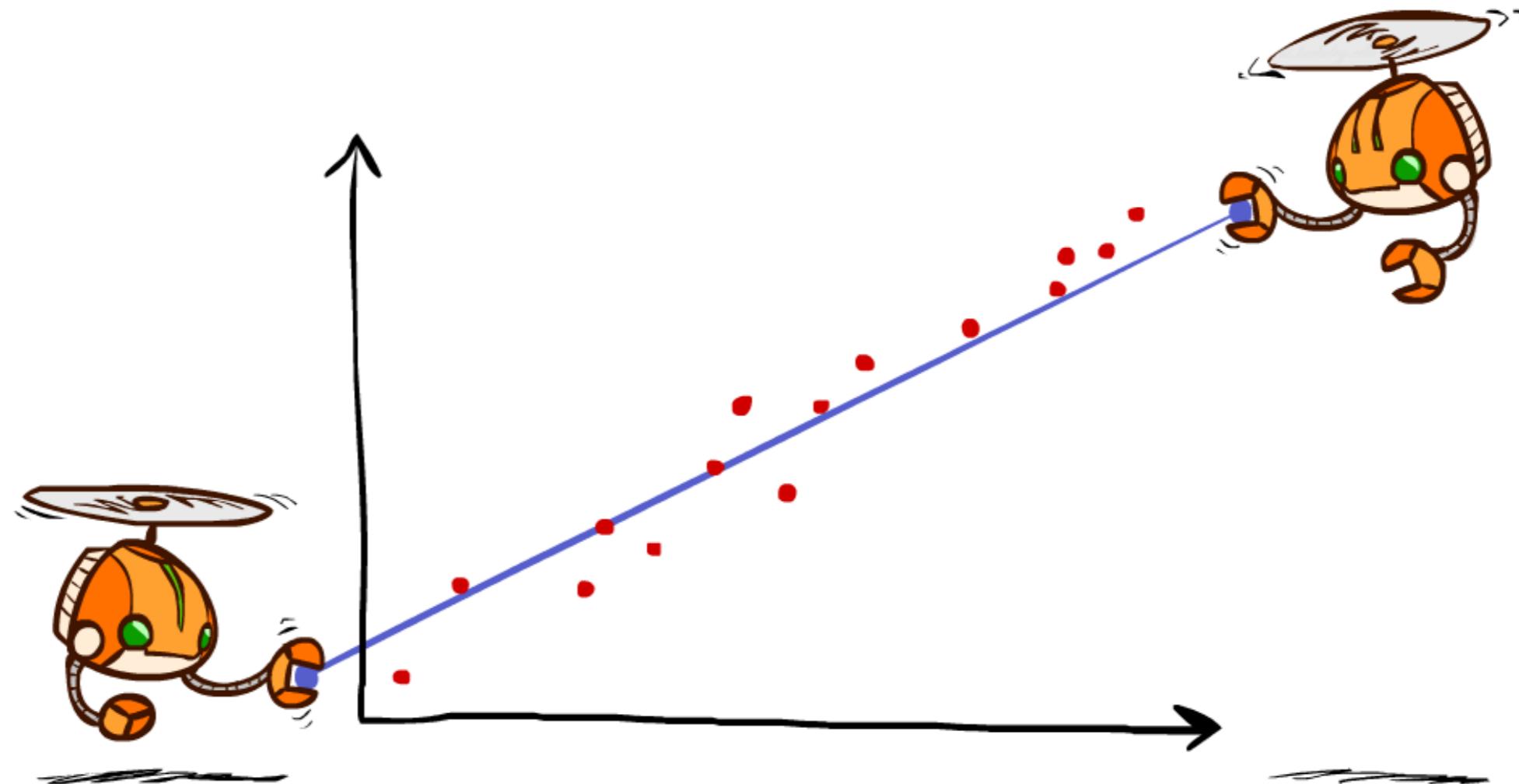
$$Q(s, a) = 4.0f_{DOT}(s, a) - 1.0f_{GST}(s, a)$$



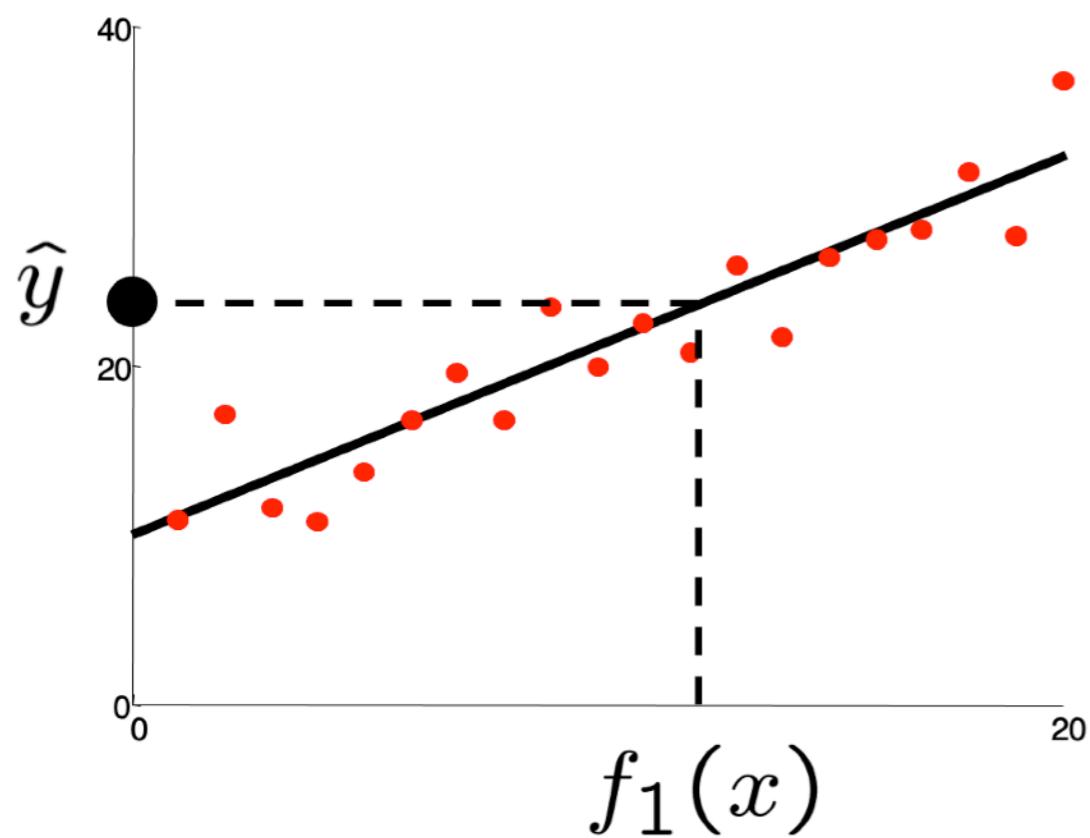
$$Q(s, a) = 3.0f_{DOT}(s, a) - 3.0f_{GST}(s, a)$$

学习率 $\alpha=0.005$

Q-Learning 和 最小二乘法

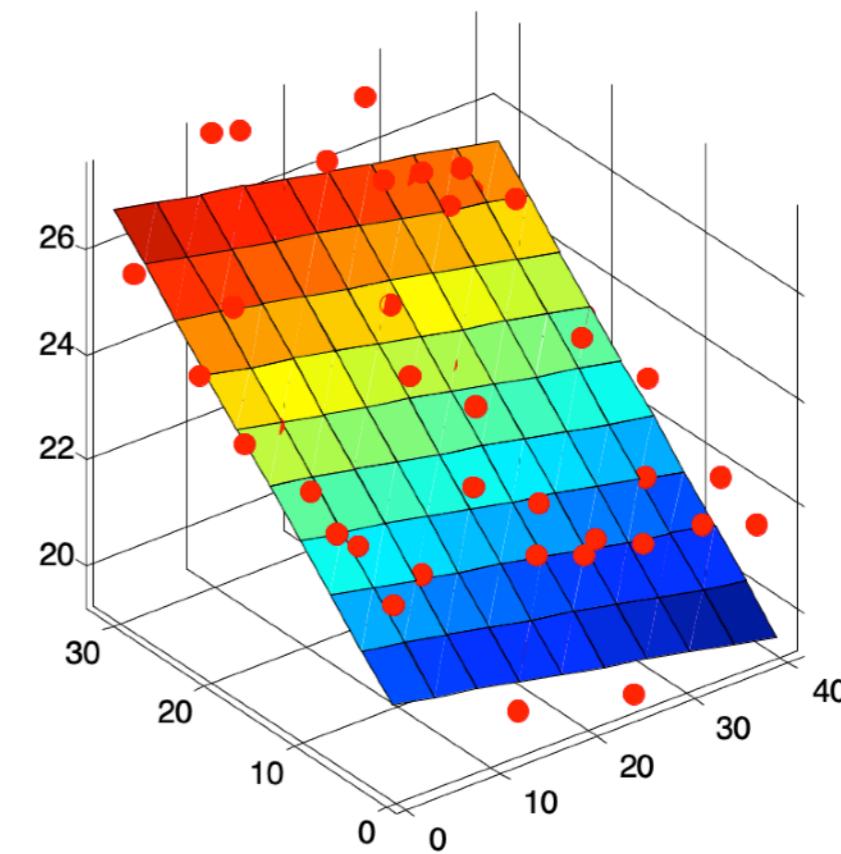


线性近似：回归



Prediction:

$$\hat{y} = w_0 + w_1 f_1(x)$$

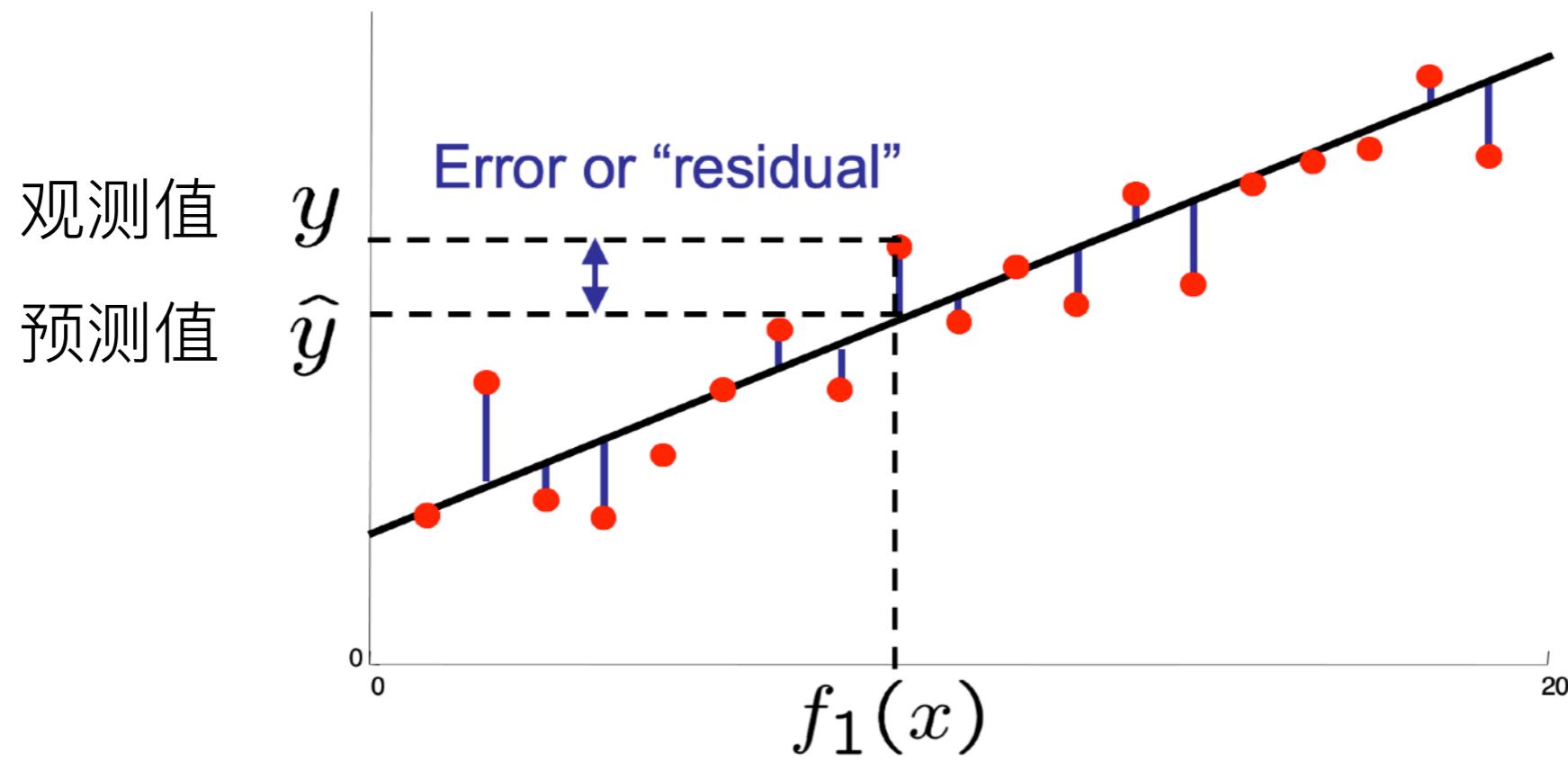


Prediction:

$$\hat{y}_i = w_0 + w_1 f_1(x) + w_2 f_2(x)$$

优化：最小二乘法

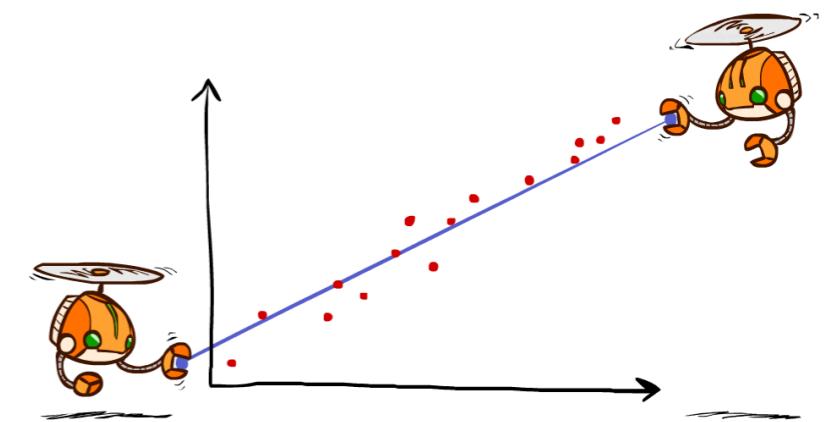
$$\text{total error} = \sum_i (y_i - \hat{y}_i)^2 = \sum_i \left(y_i - \sum_k w_k f_k(x_i) \right)^2$$



最小化误差

- 假设我们只有一个点 x , 具有特征 $f(x)$ 、目标值 y 和权重 w :

$$\text{error}(w) = \frac{1}{2} \left(y - \sum_k w_k f_k(x) \right)^2$$
$$\frac{\partial \text{error}(w)}{\partial w_m} = - \left(y - \sum_k w_k f_k(x) \right) f_m(x)$$
$$w_m \leftarrow w_m + \alpha \left(y - \sum_k w_k f_k(x) \right) f_m(x)$$

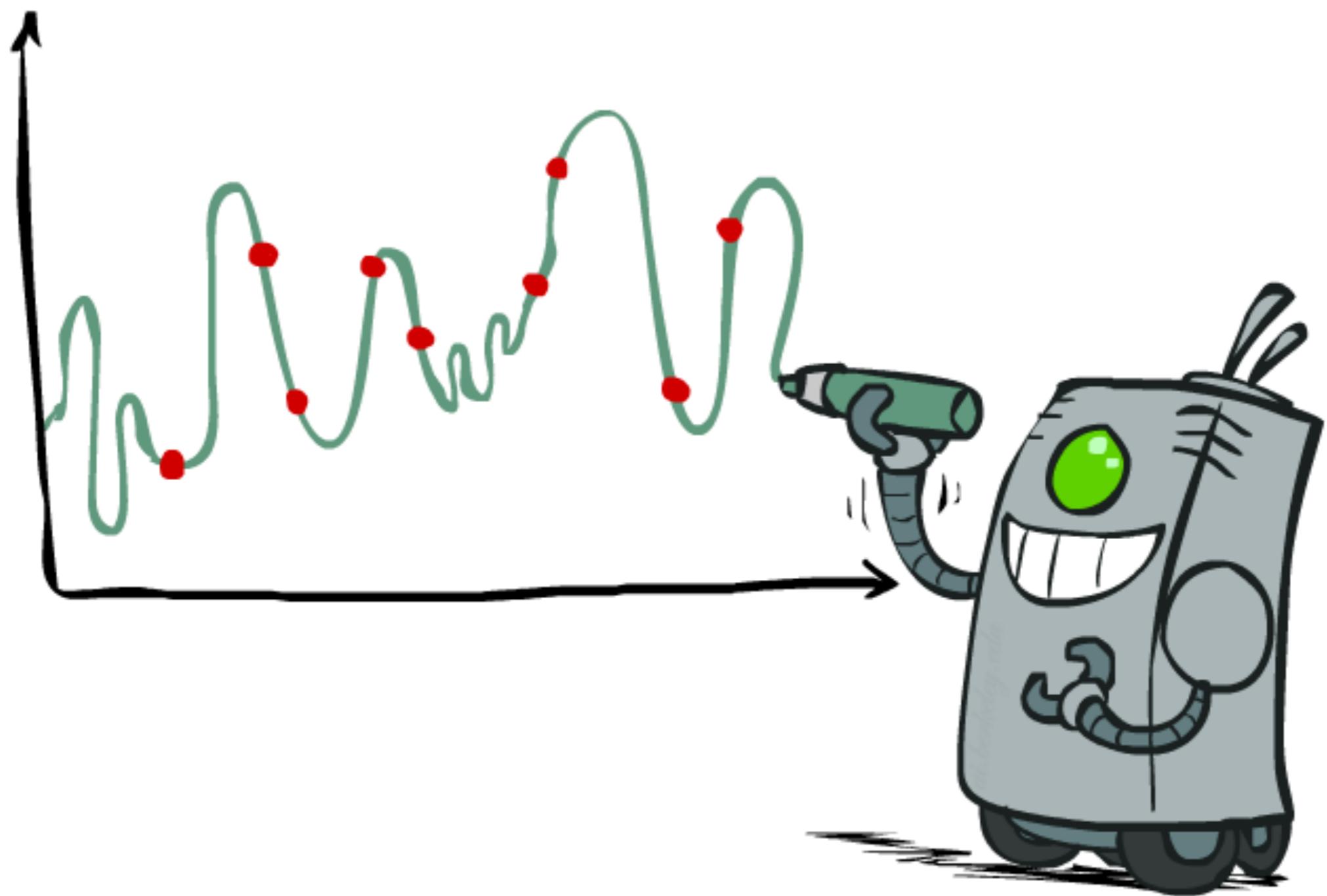


- 近似 q 更新的解释:

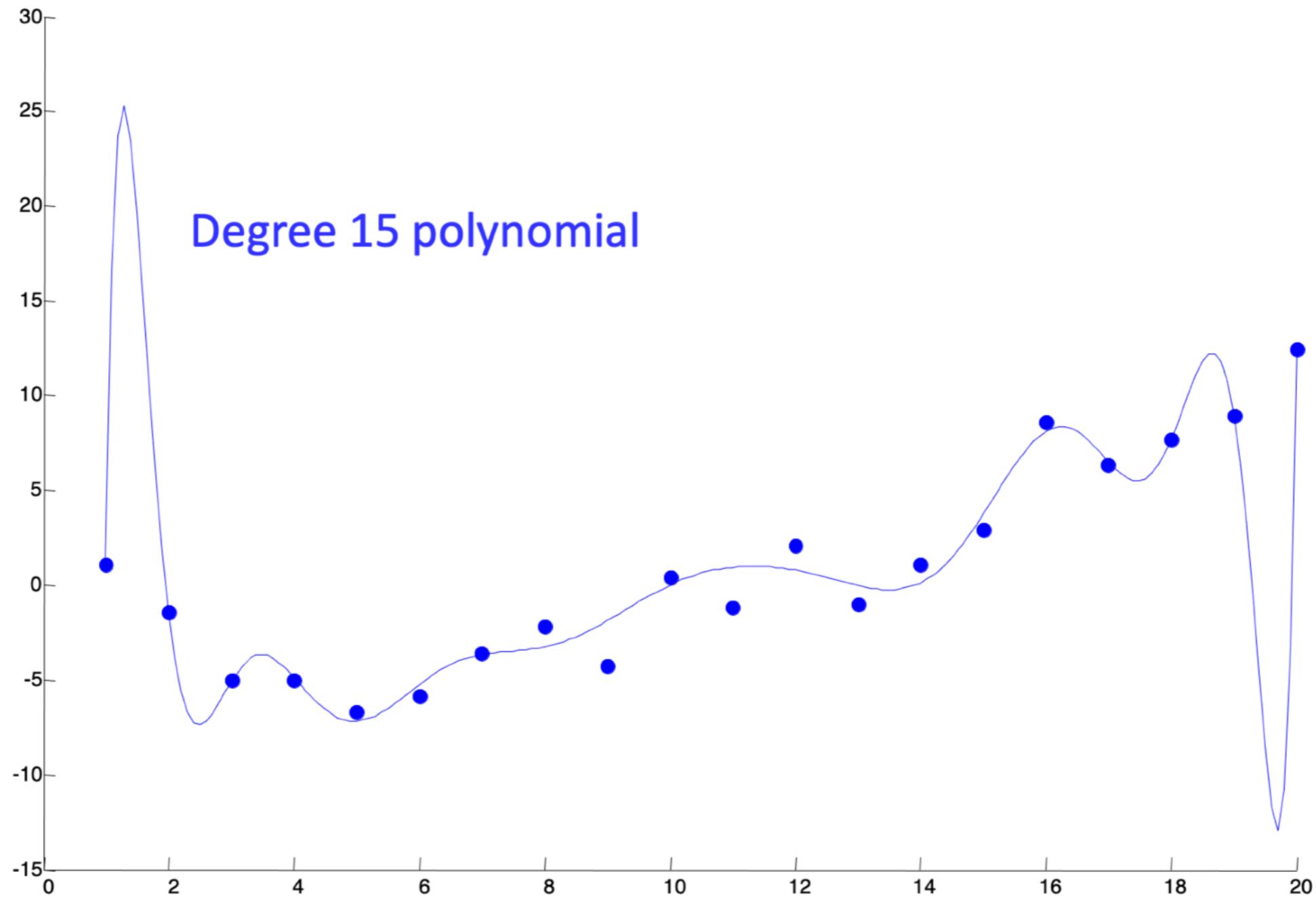
$$w_m \leftarrow w_m + \alpha \left[r + \gamma \max_a Q(s', a') - Q(s, a) \right] f_m(s, a)$$

目标值 预测值

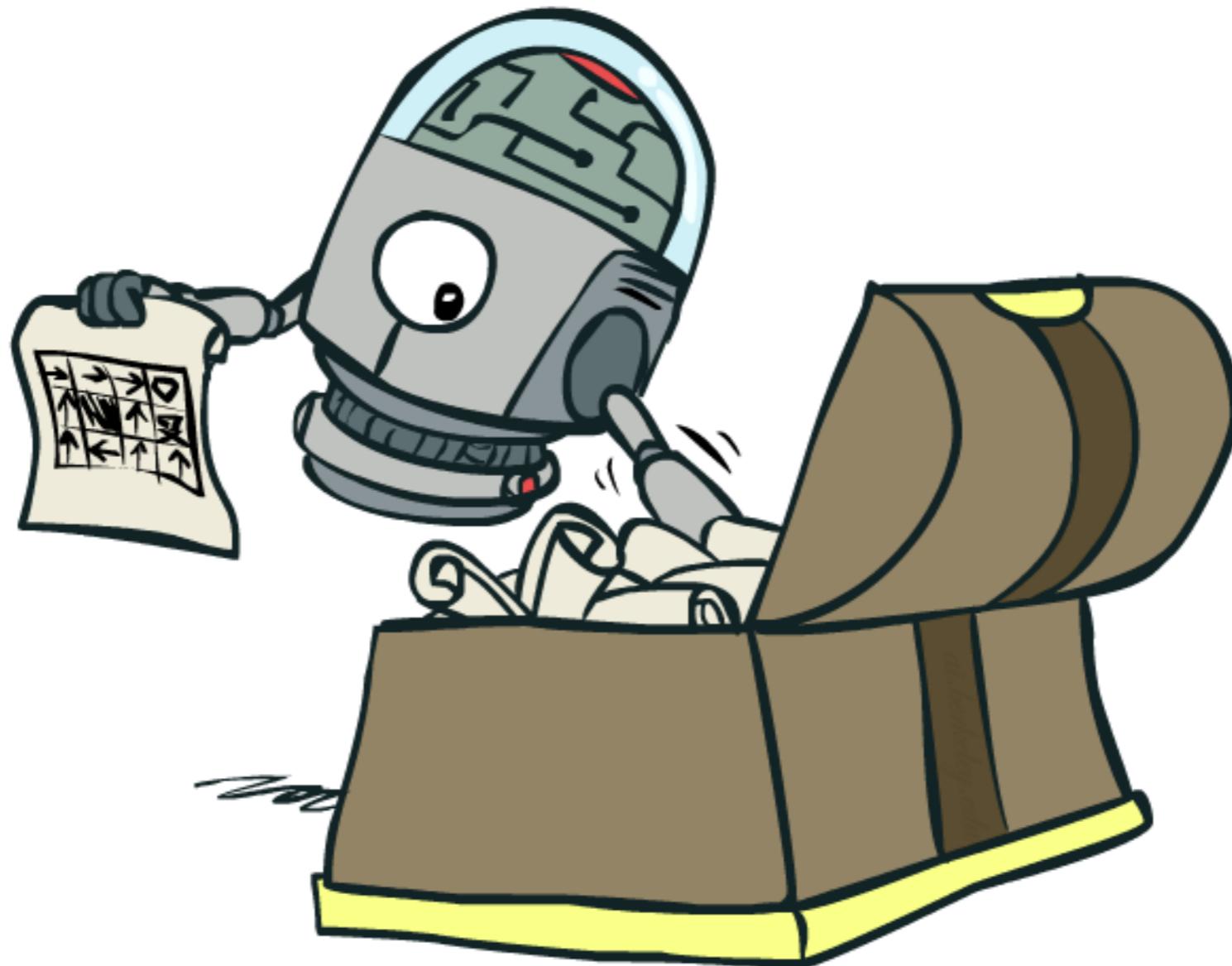
过拟合：为什么限制模型能力会有所帮助*



过拟合



策略搜索



策略搜索

- 问题：通常基于特性的策略运行良好（赢得游戏、最大化效用）并不是最接近V/Q的策略
 - 比如，价值函数可能是对未来回报的错误估计，但它们仍然产生了良好的决策
 - Q学习的优先目标：接近Q值（建模）
 - 操作选择的优先目标：正确排序Q值（预测）
- 解决方案：学习最大化回报的策略，而不是得到这个策略的Q值
- 策略搜索：从一个好的解决方案开始（例如Q-learning），然后通过局部搜索对特征权重进行调优

策略搜索

- 最简单的策略搜索：
 - 从初始线性值函数或Q函数开始
 - 轻微改变每个特征的权重，看看新的策略是否比以前更好
- 问题：
 - 我们如何判断策略变好了
 - 需要运行许多样本!
 - 如果有很多的特征，这在计算上可能是不可行的
- 更好的方法利用前瞻结构，更好地采样，一次改变多个参数.....

作业

- 在一个 8×8 的迷宫世界中：
 - 随机设定2个出口，并且给出出口随机设定一个+10或者-10的奖励；
 - Agent在迷宫中的行动存在不确定性，例如向action=east时，会有 $(1-p)$ 的概率滑到north或者说是south；
 - 随机指定 p 值、Living reward和折扣系数 γ ，但是不要让Agent知道；
 - 让Agent在这个迷宫世界中进行主动探索，并求出最佳的策略，与上一章作业所求出的最佳策略是否一致？