机器学习

深 度 强 化 学 习

滴滴内部
学习资料
请勿外传

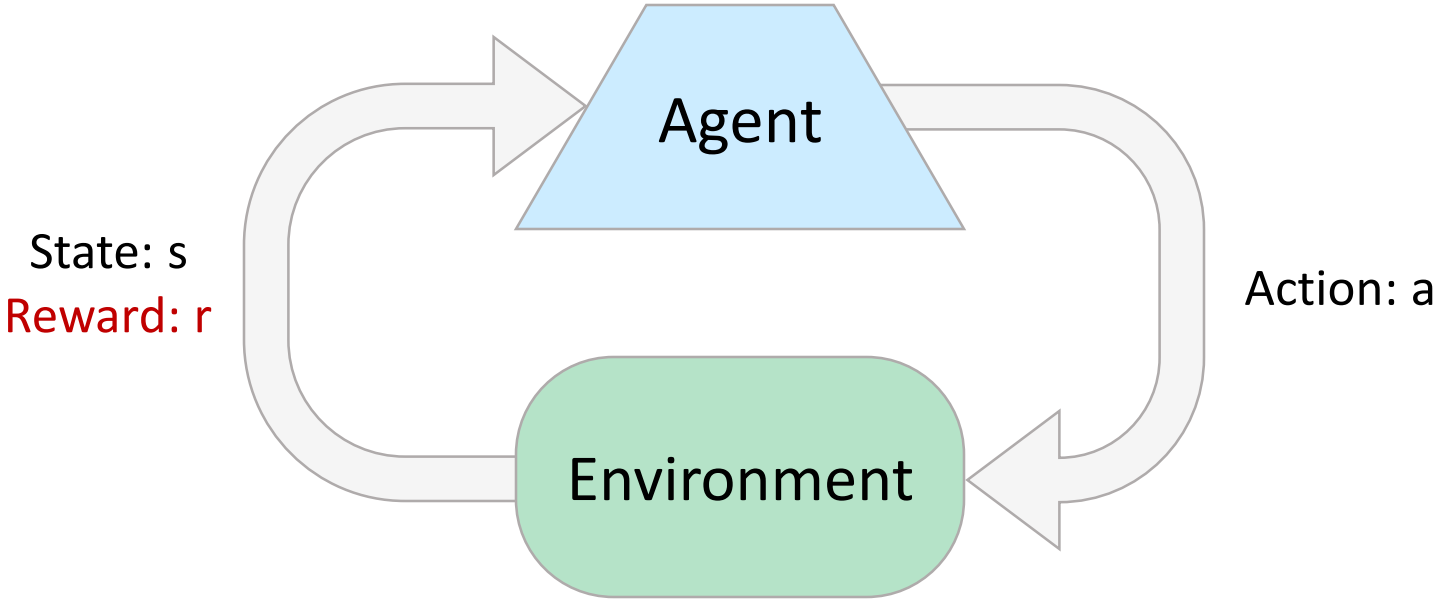
Q-学习/Pacman, FrozenLake

Tony Qin (秦志伟)

本课大纲

- 强化学习的基本要素
  - Agent，Environment，State，Action，Reward
- 有模型学习和无模型学习
- 时间差分学习（TD-learning）
- Q学习（Q-learning）
  - 查表的Q-learning (Tabular Q-learning)
  - 线性函数逼近
  - 非线性函数逼近
- Demo

强化学习（Reinforcement Learning）



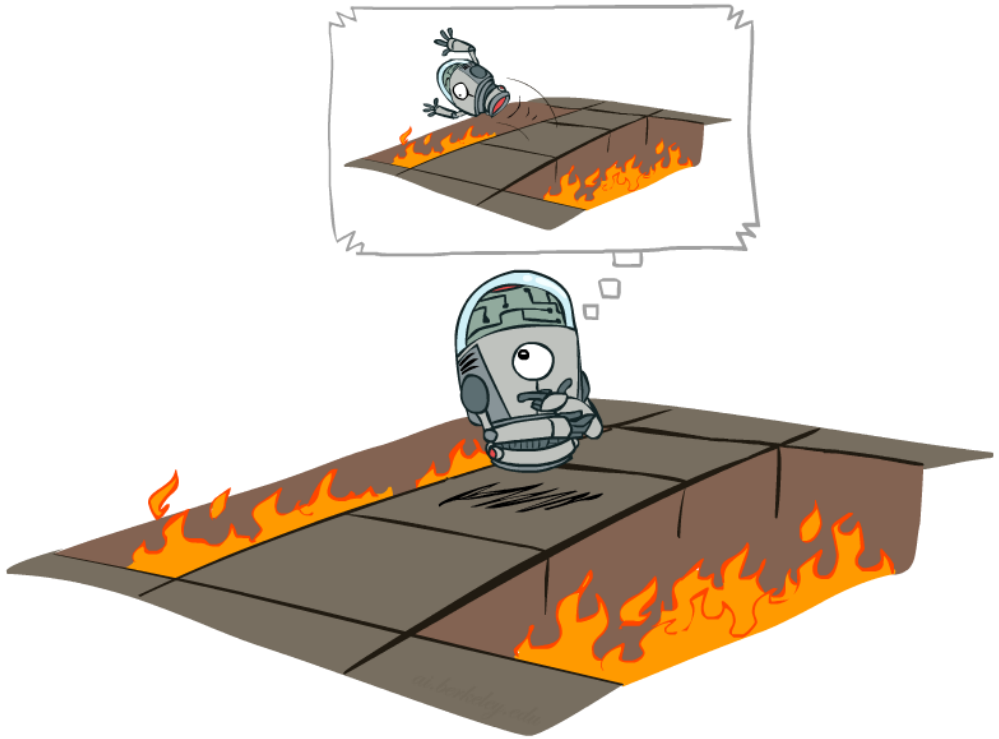
- 通过奖励（reward）获得动作的反馈
- 学习策略，使得期望总奖励尽可能大
- 通过观察环境和动作带来的结果进行学习

强化学习

- 仍然假设问题是一个MDP：
  - 状态空间 S
  - 动作空间 A
  - 转移概率 $T(s_0, a, s_1)$
  - 奖励函数 R(s, a)
- 仍然想要寻找一个好的策略π(s)
  - 每个状态下应执行的动作

- 问题是：事先不知道T和R
  - 必须通过尝试和观察反馈来学习最好的策略

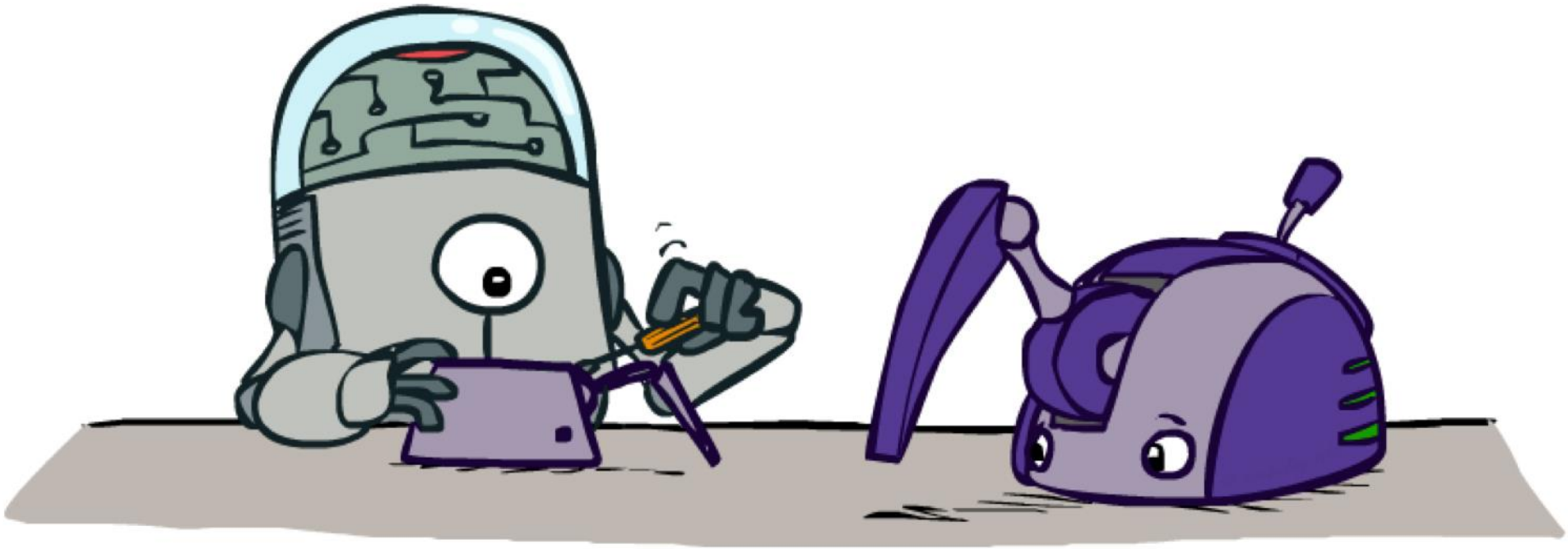
离线思考和在线学习



离线思考

在线学习

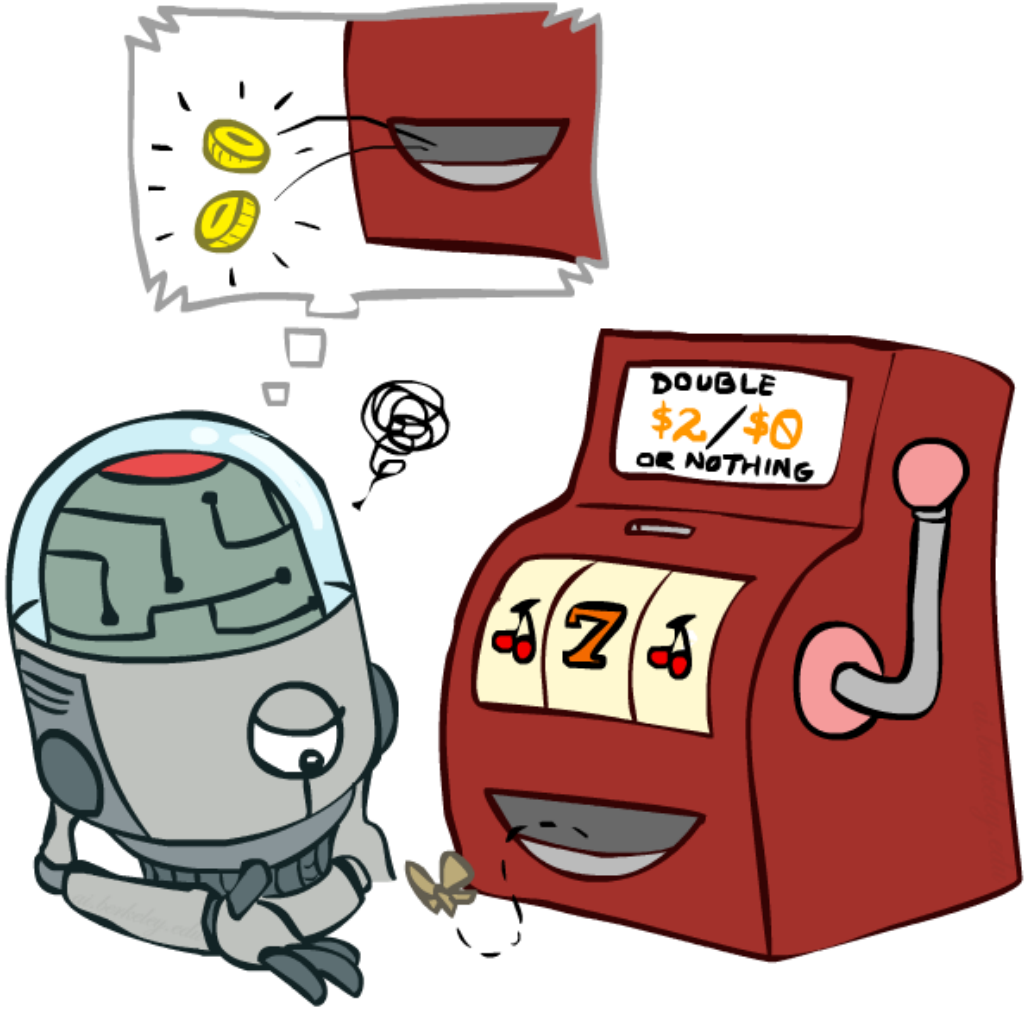
有模型学习（Model-Based Learning）



Model-based Learning

- 思路
  - 第一步：从经验数据中学习环境模型
  - 第二步：求解价值（假设模型正确）
- 第一步：简单的经验式模型学习
  - 对每对(s, a)计数它的转换输出（下一步）
  - 归一后估计转移概率T(s, a, s’）
  - 在探索过程中估计R(s, a, s’）
- 第二步：用学到的环境模型求解MDP
  - 价值迭代，策略迭代

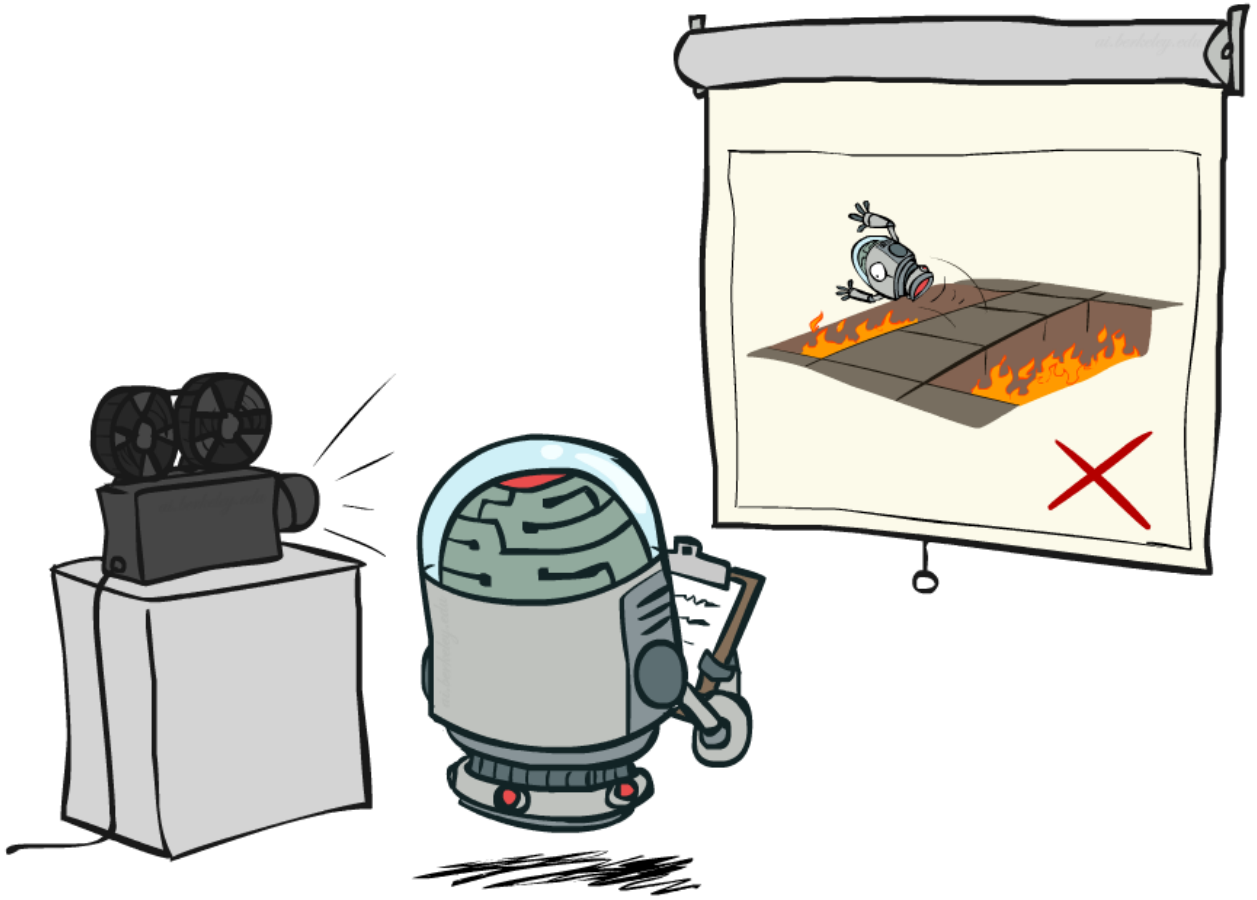
无模型学习（Model-Free Learning）



无模型学习

- 很多问题中，建立一个模型是不现实的
  - 状态动作空间（S，A）可能很大
- Agent可以仅从过往经验中学习策略，不建立模型
  - 这就是无模型学习（Model-Free Learning）
- 大体思路仍然和之前相同
  - 在一定策略下，计算V/Q函数
  - 根据V/Q函数改进策略
  - 不同：不依赖于 T 和 R

给定策略，估计价值函数



基于采样的Policy Evaluation

- 我们希望这样更新V：

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

- 没有T和R，可以多次采样然后计算结果的平均值：

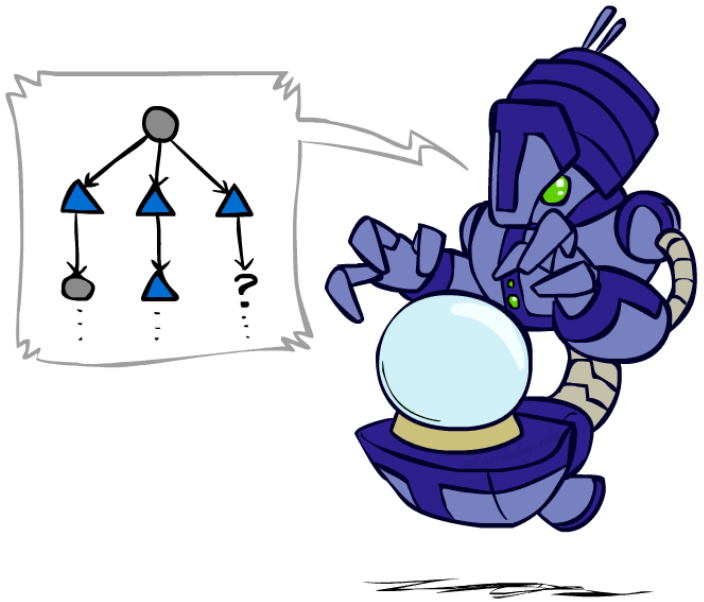
$$sample_1 = R(s, \pi(s), s_1') + \gamma V_k^{\pi}(s_1')$$

$$sample_2 = R(s, \pi(s), s_2') + \gamma V_k^{\pi}(s_2')$$

$$\ldots$$

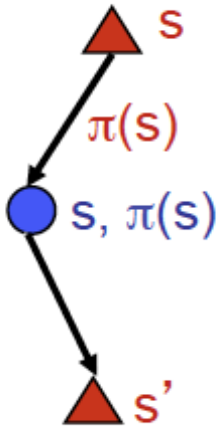
$$sample_n = R(s, \pi(s), s_n') + \gamma V_k^{\pi}(s_n')$$

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



时间差分学习（Temporal Difference Learning）

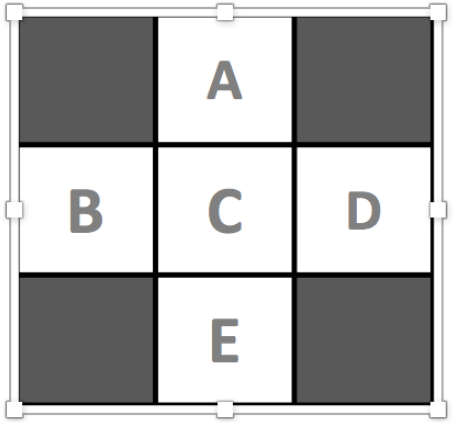
- 一个问题：我们必须记下所有(s, a)出发的经验
  - 造成非常大的记忆负担！
  - 没法倒回时间去从s起抽样其他样本。。。



- TD Learning：在线更新，不需要记忆已知样本
  - 每次获得经验(s, a, r, s′)后就更新V(s)
  - 访问s′的概率越大，s′的价值就会用得越多
  - V(s)样本：$sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$
  - 更新V(s)：$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$
  - 等价于：$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

例子：TD Learning

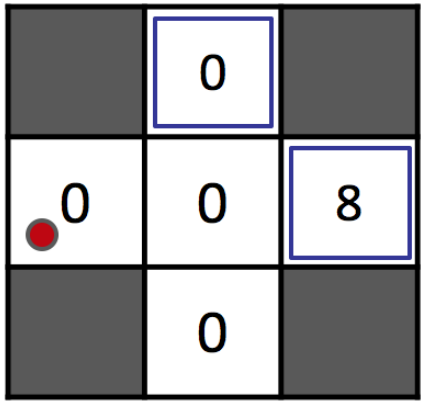
States



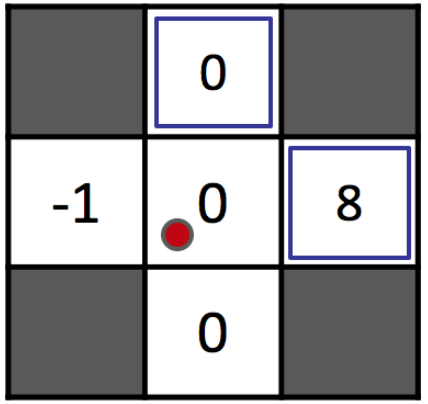
Assume: γ = 1, α = 1/2

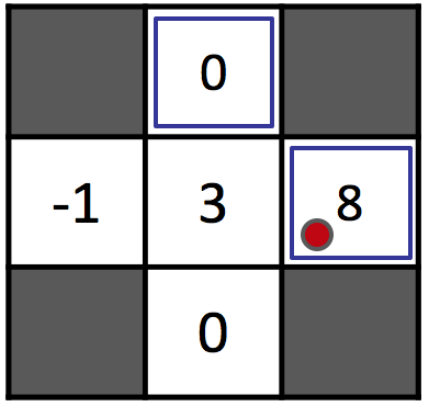
Observed Transitions

B, east, C, -2



C, east, D, -2

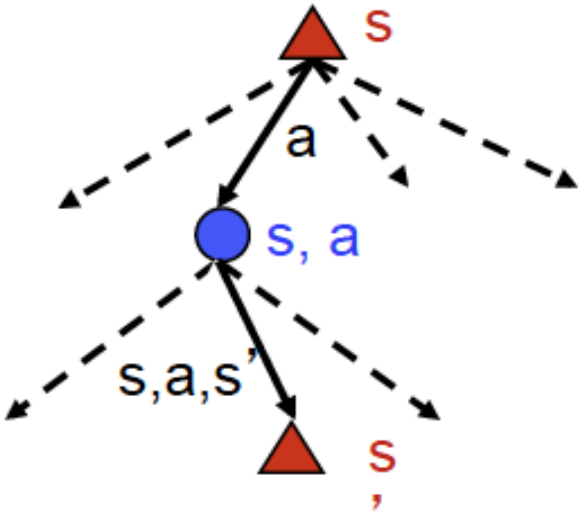




$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right]$$

TD学习的局限性

- 有了V函数，如何导出策略？
- 如果想把价值转化为策略
  - $\pi(s) = \text{argmax}_a Q^*(s,a)$
  - $Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$
  - 转移概率是不知道的！
- 思路：直接学习Q^*价值函数
  - 这样在s下直接对Q取argmax就行了
  - 动作选择也不需要模型！



加入控制：Q-Learning

- 与计算V的方法非常相似：

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

- 动态地学习Q函数
  - 获取一个转移: (s, a, r, s’)
  - Q(s, a)新的目标值：$sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$
  - 维护Q(s, a)的running average：

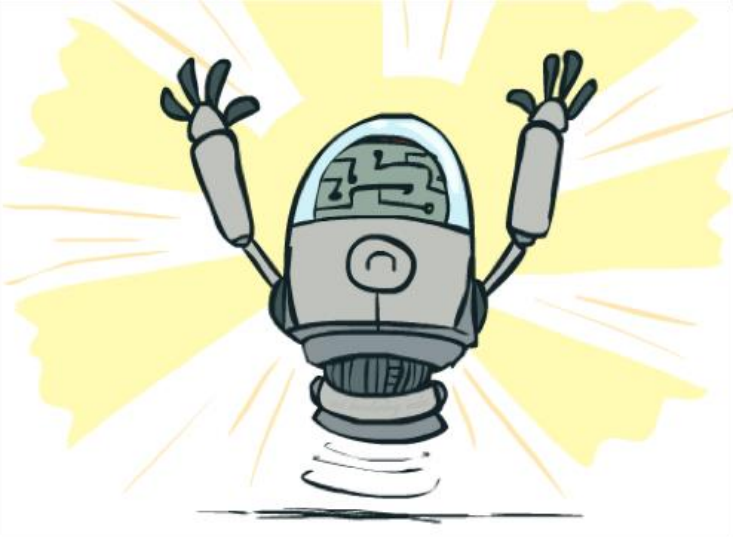
$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)[sample]$$

  - 等同于

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

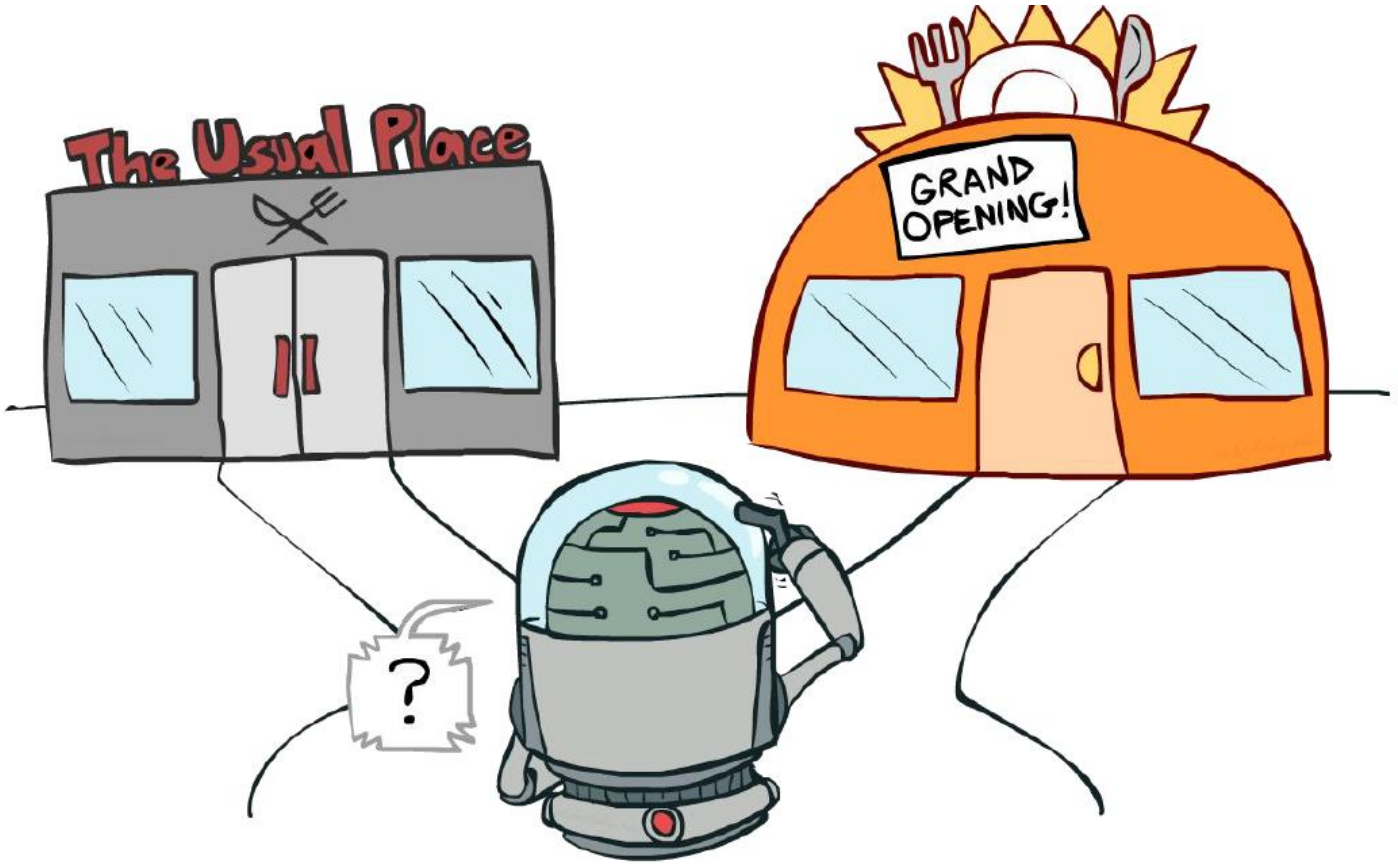
Q-Learning的性质

- Q学习会收敛到最优策略
  - 虽然收集经验时用的动作并不一定最优
  - 这叫做异策略学习(off-policy learning)
- 前提条件
  - 探索要足够多
  - 学习率(α)最终要降到足够小，但不能降的太快



保守还是探索（Exploit vs. Explore）

expoitation



exploration

Drawing by Ketrina Yim

探索方法

- 如果完全按照Q值挑选动作，很容易陷入局部最优

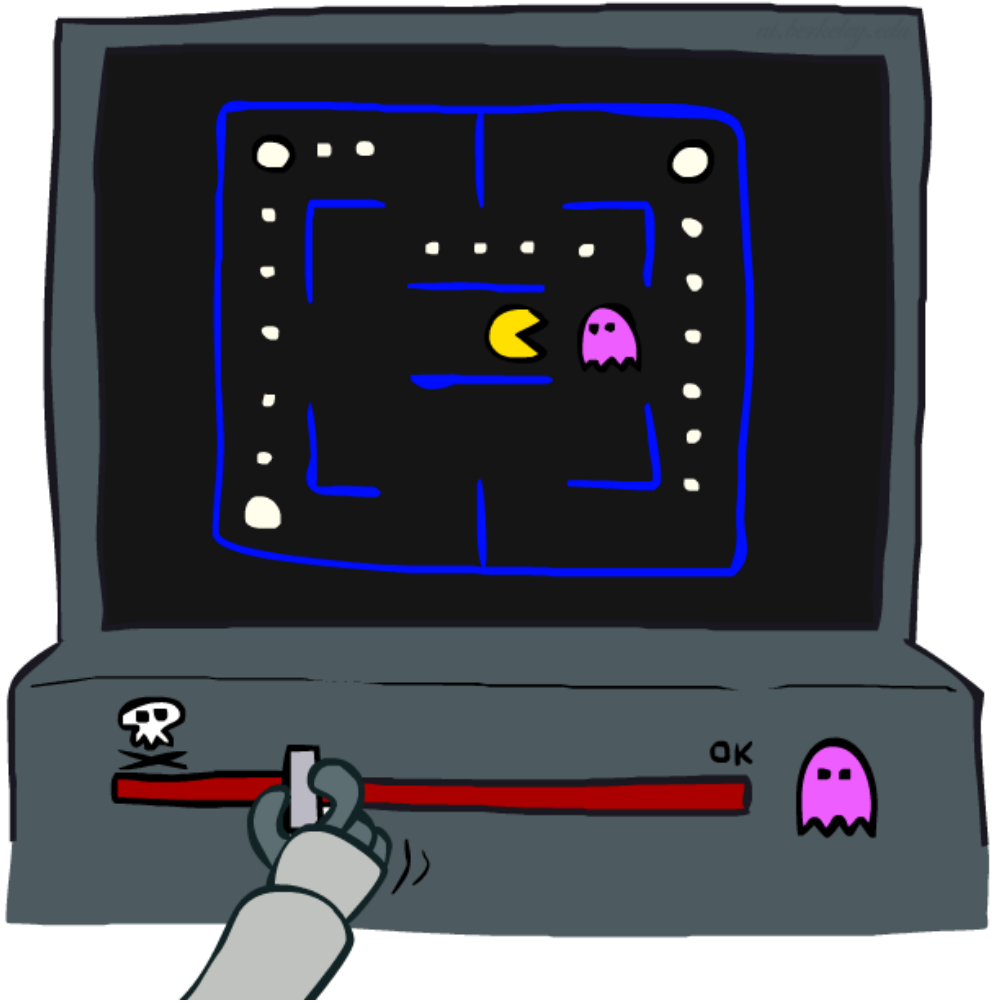
- 简单的探索策略：ϵ-greedy
  - ϵ是个较小的概率
  - 每一步： ϵ的概率随机执行一个动作
  - （1- ϵ）概率执行策略的动作

- 问题
  - 即使学习完成了，还在执行一些随机动作
  - 解决方法1：逐渐减小ϵ
  - 解决方法2：探索函数

Demo

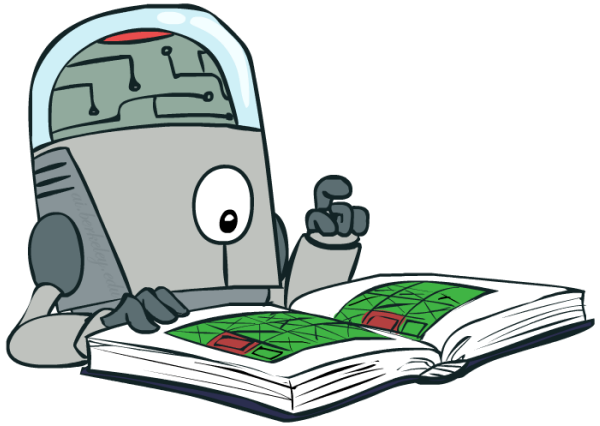
- Env: FrozenLake
  - 从起点出发，在冻湖上游走直到飞盘所在位置
  - 掉下窟窿，回合结束
- 湖网格的构造
  - SFFF (S: 起始点，安全)
  - FHFH (F: 冻面，安全)
  - FFFH (H: 窟窿，危险！)
  - HFFG (G: 目的地，飞盘的位置)
- 动作
  - 上下左右
  - 由于冰面滑，不是每次都能到想要的位置
- 奖励
  - 目的地：+1；其他都为0.

带函数近似的Q-Learning



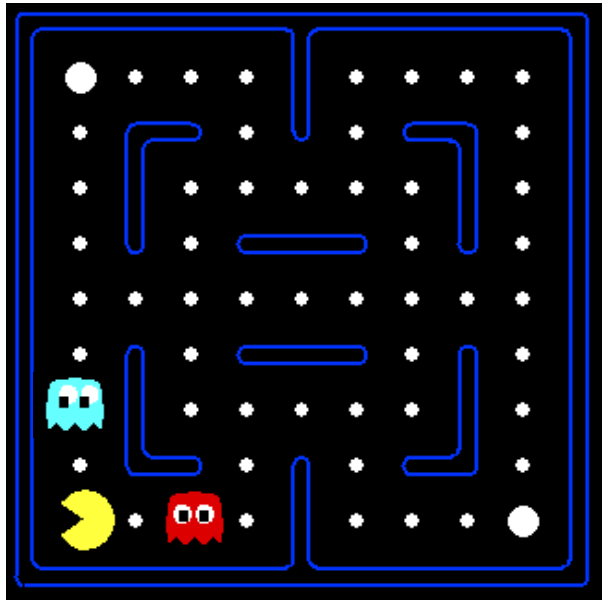
状态间的推广

- 基本Q-Learning需要维护一张Q(s, a)的表
  - 格子数 = |A| x |S|
- 不适用很多真实场景
  - 太多状态需要探索
  - 内存要求太高，存不下
- 解决办法：推广 (generalization)
  - 从经验中学习小部分训练状态的价值
  - 推广到新的，类似的状态
  - 听起来很熟悉？

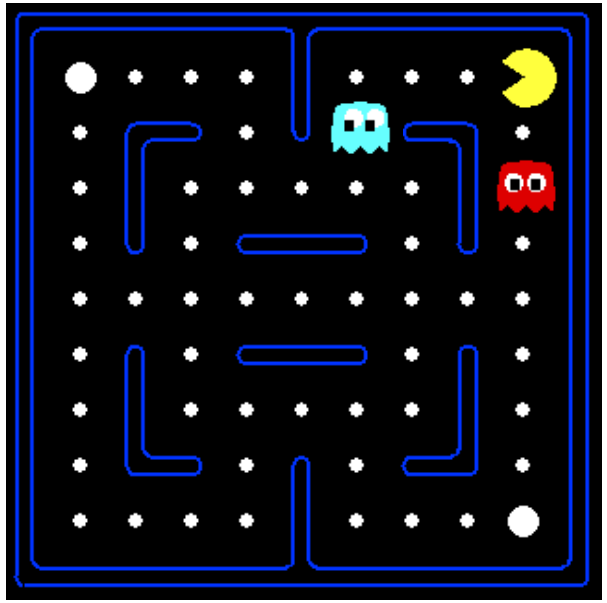


Pacman例子

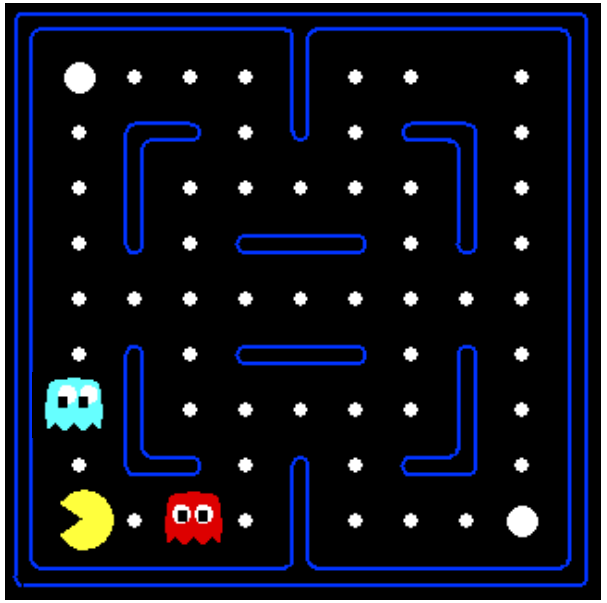
通过经验发现了这是个坏状态



在基本Q学习中，完全不知道这个状态也是坏的



甚至这个也不知道！



推广例子：监督学习

Dog? Yes

Dog? Yes

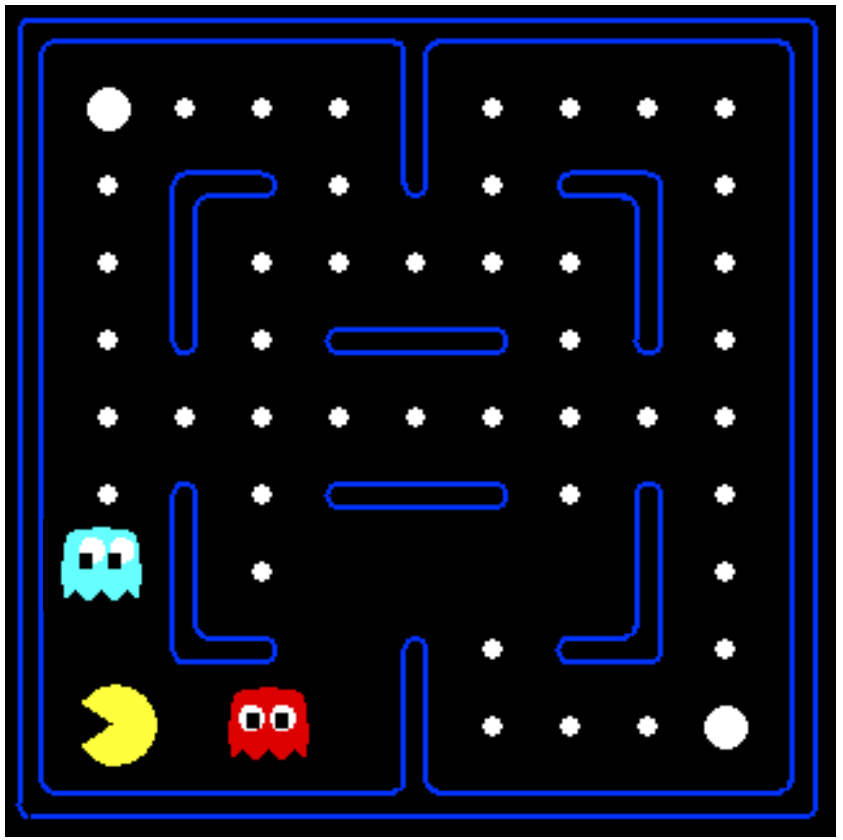
Dog? Yes

Dog? No

Dog?

用特征表示状态

- 用一系列特征来描述状态
  - 和监督学习中的概念类似
  - 特征：状态 -> 实数，抓住重要性质
- 特征例子
  - 离幽灵的距离
  - 离最近豆子的距离
  - 一定范围内幽灵的数量
  - Pacman是否在隧道中
  - 状态是否和某个特定状态一样（例如这个）
- 也可描述状态-动作（Q-state）
  - 移动离豆子更近了

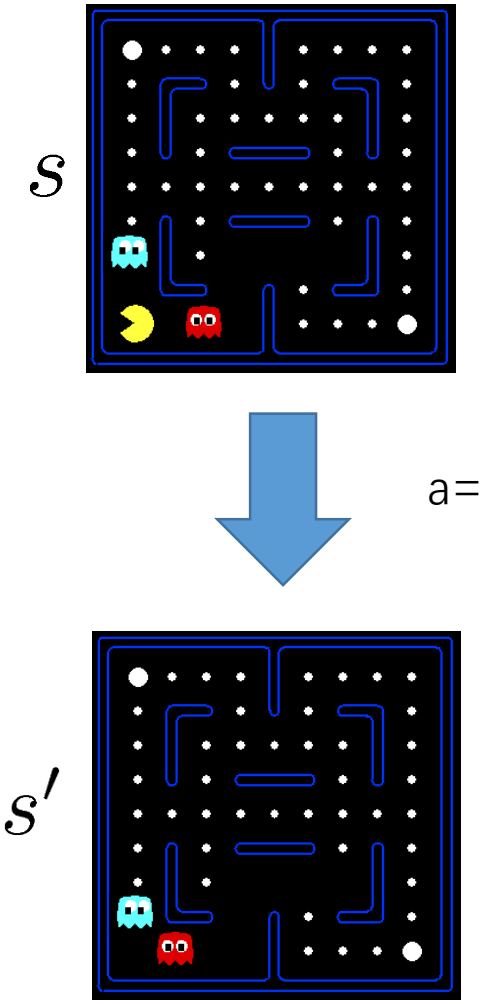


线性函数逼近

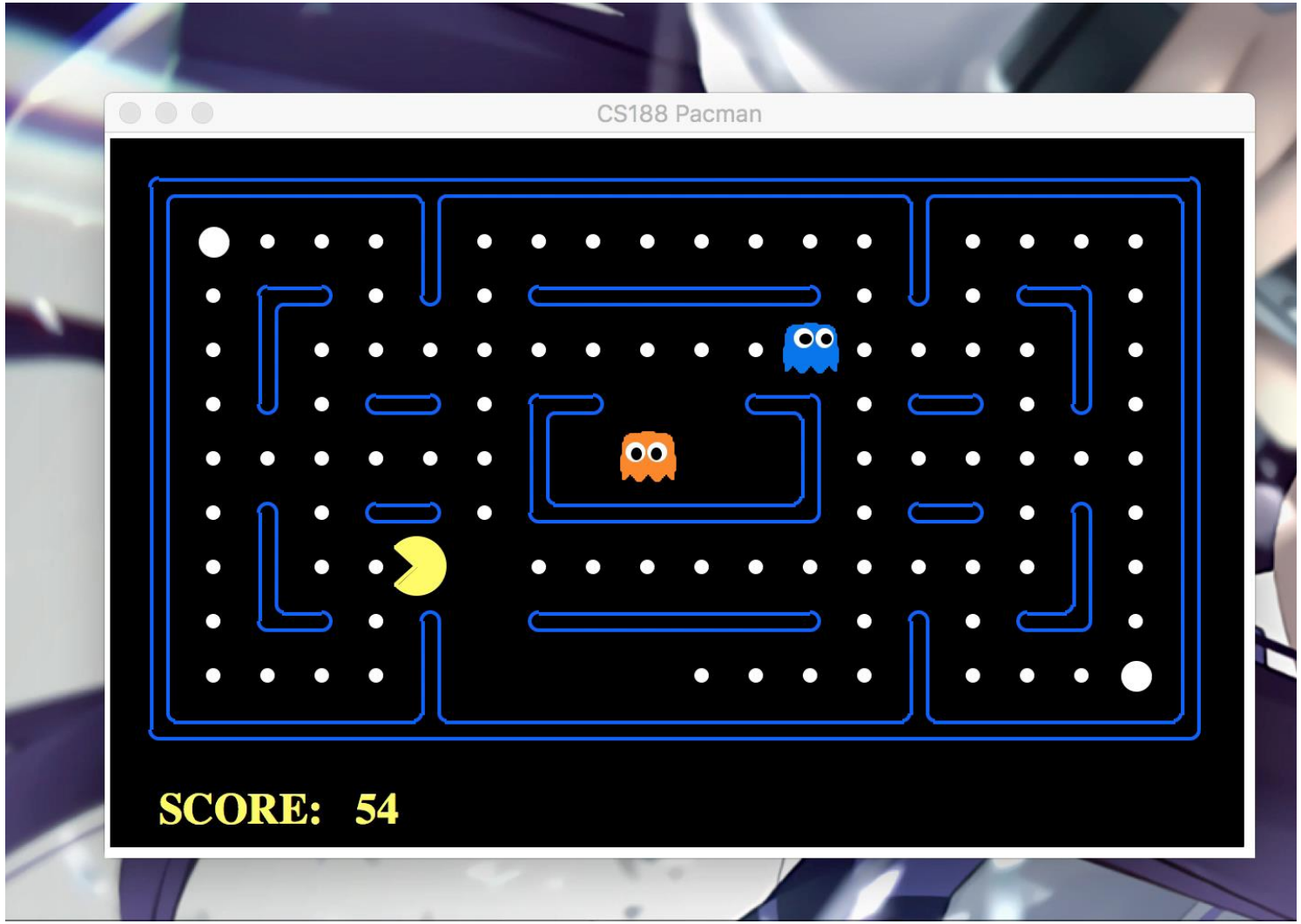
- 我们可以用一系列特征组成的线性函数来表示Q(s, a)
- $Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)$
  - w: 权重，f: 特征函数
- 回忆Q学习更新
  - $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
- 线性函数逼近的Q学习实际上更新的是特征权重w
  - $w_i \leftarrow w_i + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] f_i(s,a)$
  - 在线地最小化square loss，使$Q(s,a)$与$r + \gamma \max Q(s',a')$尽可能接近

Pacman例子

- $Q(s,a) = 4.0 f_{dot}(s,a) - 1.0 f_{gst}(s,a)$
- $f_{dot}(s, north) = 0.5$
- $f_{gst}(s, north) = 1.0$
  - $Q(s, north) = 1.0$
- $Q(s', .) = 0$
  - $r + \max_{a'} Q(s', a') = -500$
  - Error = -501
- $w_{dot} \leftarrow 4.0 + \alpha(-501)0.5$
- $w_{gst} \leftarrow 1.0 + \alpha(-501)1.0$
- $Q(s,a) = 3.0 f_{dot}(s,a) - 3.0 f_{gst}(s,a)$



Demo：用线性函数训练的Pacman



非线性函数逼近

- 线性函数太简单，能用更复杂的函数近似吗？
- $Q(s,a) \approx \hat{Q}(s,a;\theta)$
  - $\hat{Q}(\theta)$为非线性函数，例如神经网络
  - 可能不收敛
    - 移动靶问题
    - 经验的前后关联性

- DQN：用深度神经网络逼近Q价值函数的Q学习
  - 有实际中的技巧克服收敛问题

Acknowledgement

- The slides are adapted from the lecture slides of UC Berkeley CS 188, Stanford CS 231, and Reinforcement Learning: An Introduction.
- Special thanks to Prof. Jieping Ye, Ke Weicheng, Xu Zhe, and Bian Wei for many constructive feedbacks on this presentation.

机器学习

Thank You

www.xiaojukeji.com

滴滴
滴滴一下 美好出行