第 8 章　强化学习

教材： 王万良《人工智能导论》（第4版）

https://www.icourse163.org/course/ZJUT-1002694018

社区资源： https://github.com/Microsoft/ai-edu

参考MOOC： 人工智能：模型与算法（浙大 吴飞）

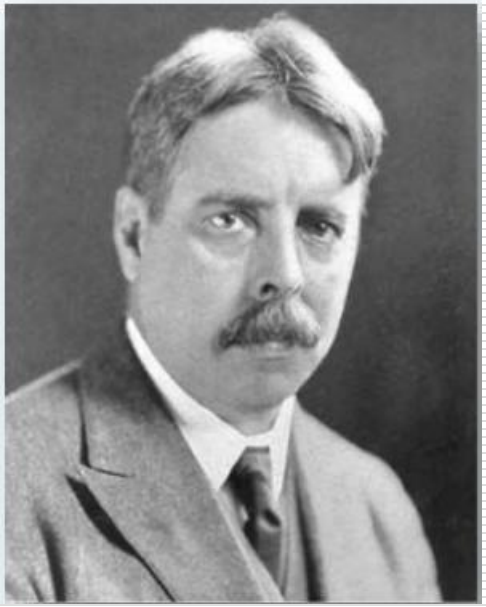
人工智能与信息社会（北大 陈斌）

导论

- 强化学习基础：试错学习理论
- 强化学习中策略优化与策略评估
- 强化学习求解：Q-Learning
- 深度强化学习：深度学习+强化学习

试错学习与强化学习

- 爱德华·桑代克
  - 美国心理学家
  - 现代教育心理学之父
  - 心理学行为主义的代表人物之一
  - 提出试错式学习理论



试错学习与强化学习

- 机关盒子（puzzle box）
  - 将饿猫关入此笼中，笼中放一条鱼，饿猫急于冲出笼门去吃笼外鱼，但是要想打开笼门，饿猫必须一气完成若干个机关。

试错学习与强化学习

- 效果律（law of effect）
  - 紧接着有利后果的行为更有可能再次发生。
    - 被老师称赞的工作或行为，你会继续保持。
  - 不良后果的行为不太可能再次发生。
    - 如果你上课迟到并错过重要内容，之后就会吸取教训。

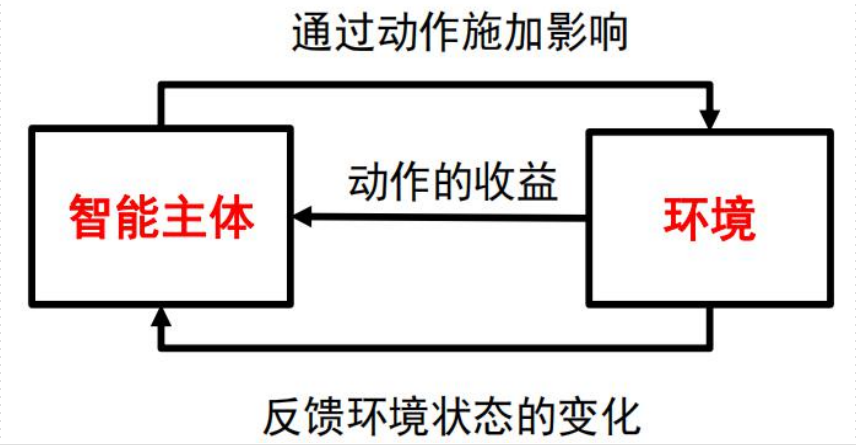
试错学习与强化学习

- 试错式学习（trail and error）
  - 猫的学习是经过多次的试误，由刺激情境与正确反应之间形成的联结所构成的。
  - 人的学习的过程也是一种渐进的尝试错误的过程。在这个过程中，无关的错误的反应逐渐减少，而正确的反应最终形成。

强化学习：在与环境交互之中进行学习

- 生活中常见的学习过程
  - 人通过动作对环境产生影响　　（向前走一步）
  - 环境向人反馈状态的变化　　（撞到了树上）
  - 人估计动作得到的收益　　（疼痛）
  - 更新做出动作的策略　（下次避免向有树这一障碍的方向前进）

强化学习模仿了这个过程，在智能主体、动作的收益体与环境的交互中，学习能最大化收益的行动模式



强化学习（reinforcement learning）

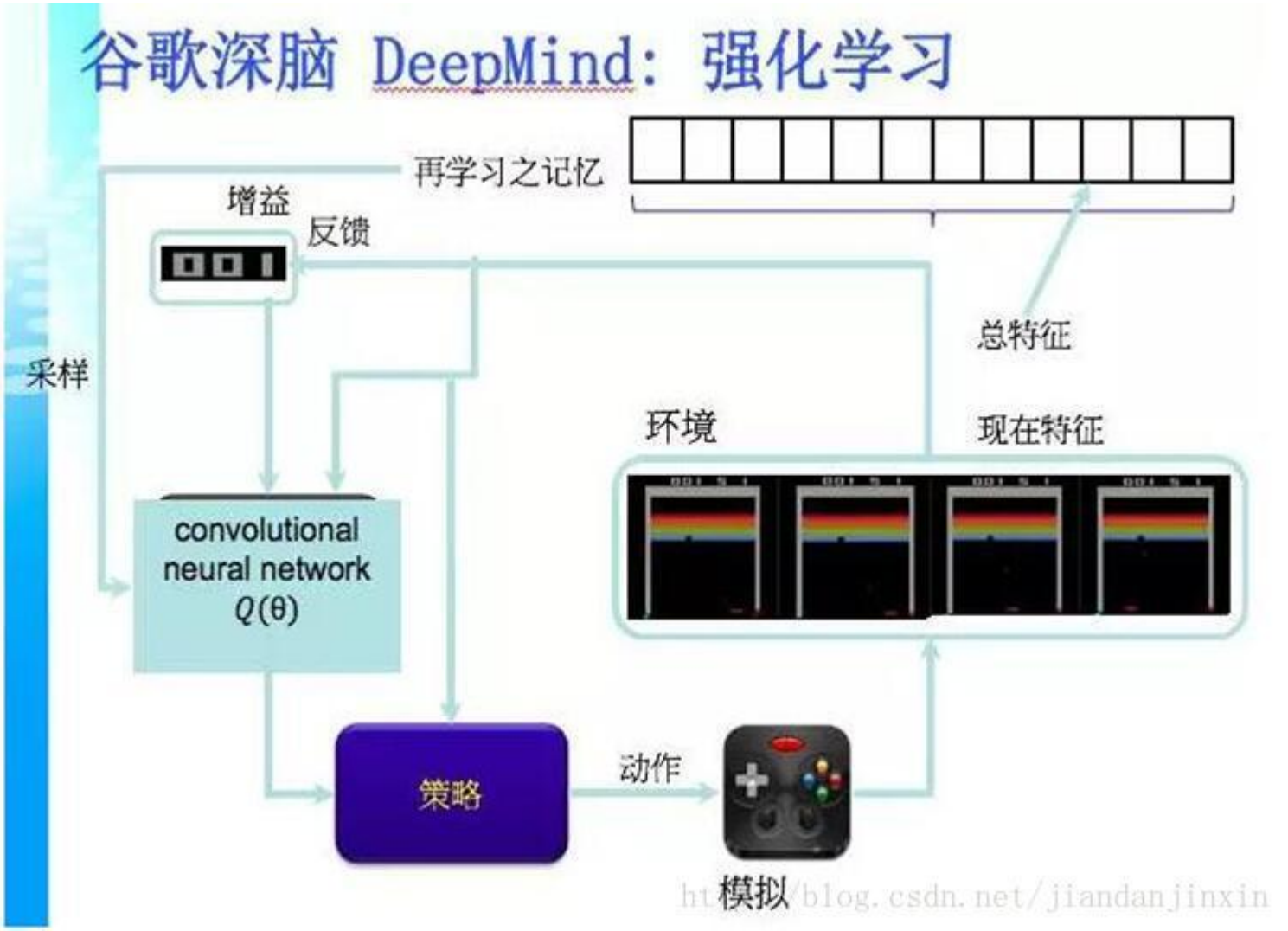
- 使得计算机能够像人一样通过不断试错式学习，完全自主掌握一项技能
- 不需要借鉴人类的经验
- 具有发展强人工智能潜力

Alpha Zero

- 利用试错式学习思想，自己跟自己不断对弈来提升水平
- 用这种通用的学习方式，在围棋、国际象棋、日本将棋等多个领域超越人类水平

强化学习

- DeepMind 就是将深度学习应用到强化学习中去的范例



Example: Learning to Walk

Initial

A Learning Trial

After Learning [1K Trials]

[Kohl and Stone, ICRA 2004]

Example: Learning to Walk

Initial

[Kohl and Stone, ICRA 2004]

[Video: AIBO WALK – initial]

Example: Learning to Walk

Training

[Kohl and Stone, ICRA 2004]

[Video: AIBO WALK – training]

Example: Learning to Walk

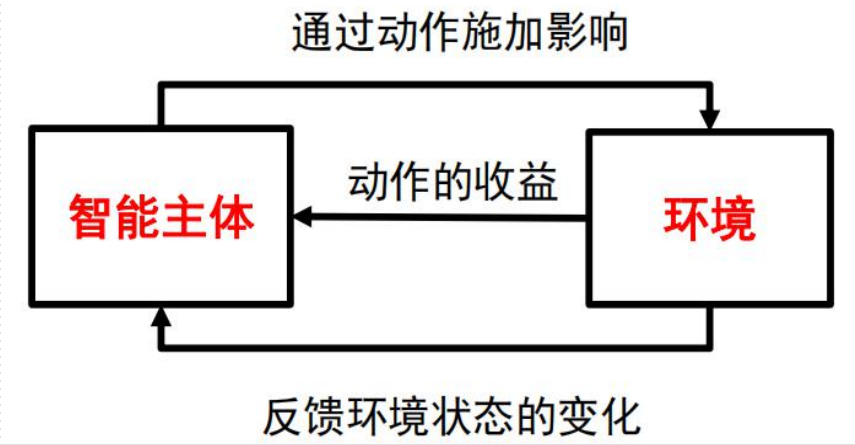
Finished

[Kohl and Stone, ICRA 2004]

[Video: AIBO WALK – finished]

强化学习中的概念

- 智能主体（agent）
  - 按照某种策略（policy），根据当前的状态（state）选择合适的动作（action）
  - 状态指的是智能主体对环境的一种解释
  - 动作反映了智能主体对环境主观能动的影响，动作带来的收益称为奖励（reward）
  - 智能主体可能知道也可能不知道环境变化的规律
- 环境（environment）
  - 系统中智能主体以外的部分
  - 向智能主体反馈状态和奖励
  - 按照一定的规律发生变化



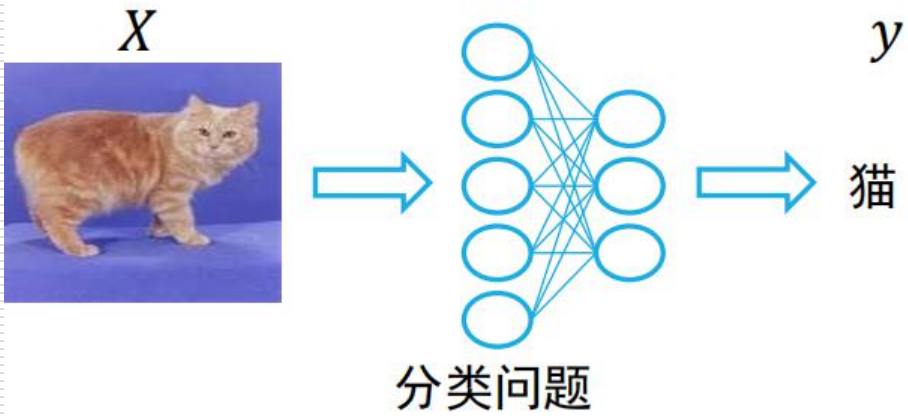
机器学习的不同类型

有监督学习

从数据X和标签y中学习映射$f: X \mapsto y$

X：图像、文本、音频、视频等

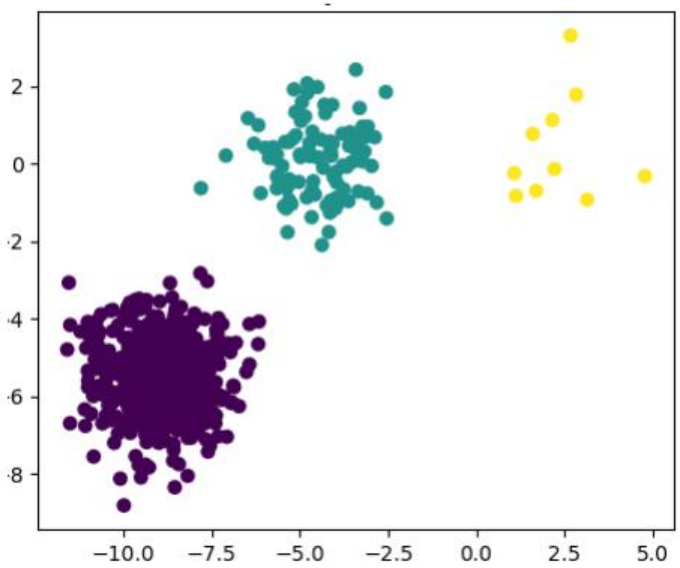
y：连续或离散的标量、结构数据



分类问题

无监督学习

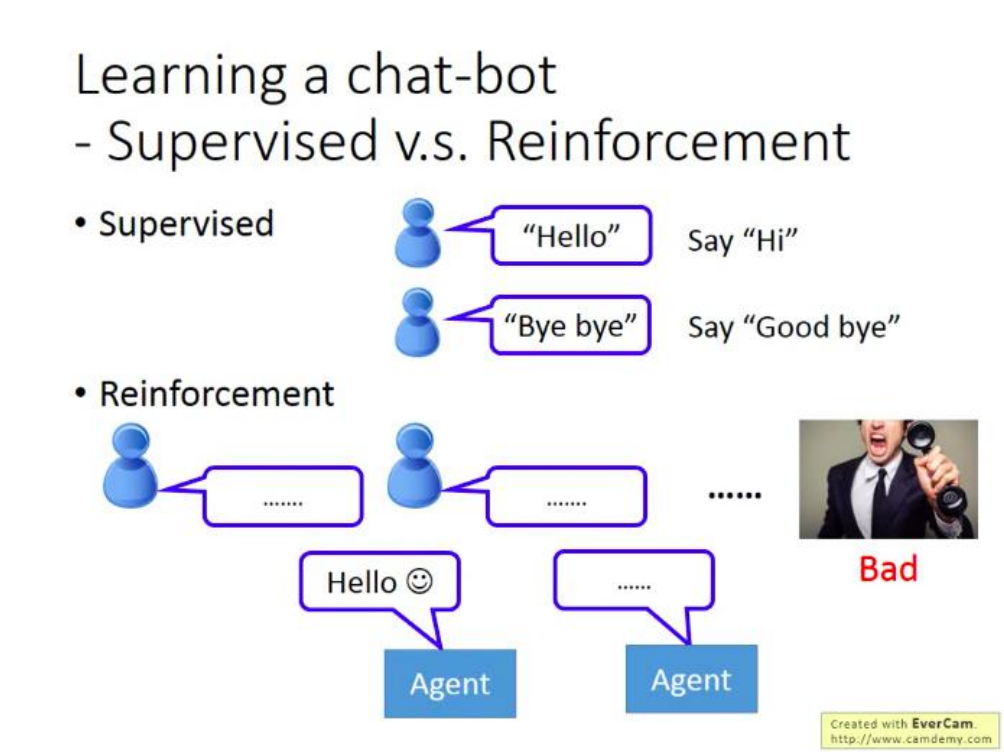
寻找数据X中存在的结构和模式



聚类问题

监督学习vs强化学习

- 监督学习
  - 如果对方说”hello”，机器应该回复”Hi”;如果对方说”Bye bye”，机器应该回复”Good bye”
- 强化学习
  - 机器和对方瞎说，最后获得人类怒挂电话的负反馈（滑稽）。但机器仍不知道是自己哪句讲对，要靠自己去弄清楚。



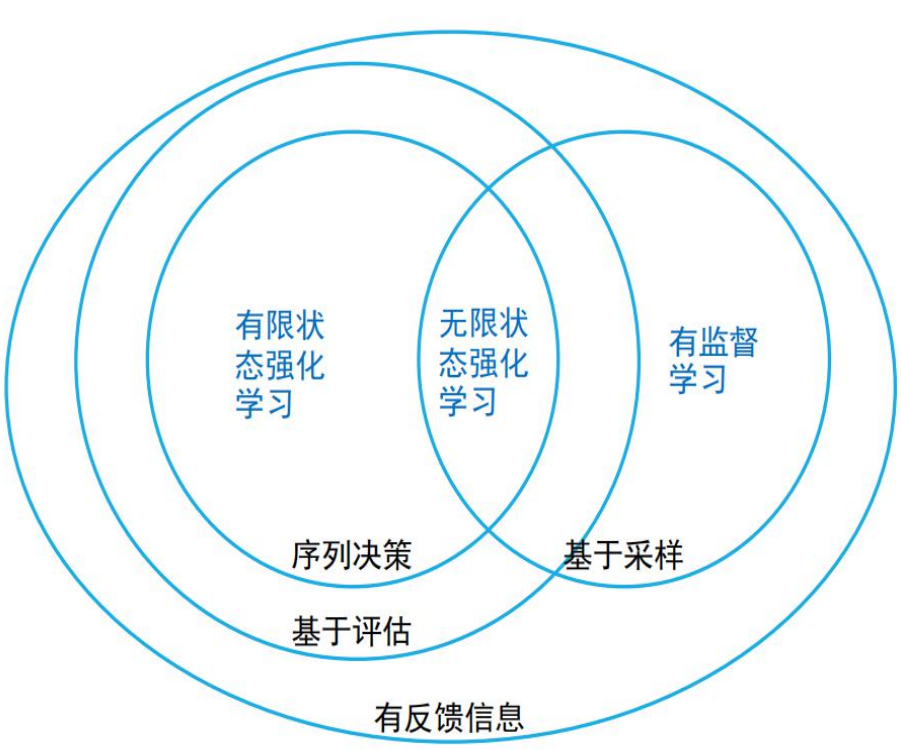
强化学习的特点

	有监督学习	无监督学习	强化学习
学习依据	基于监督信息	基于对数据结构的假设	基于评价（evaluative）
数据来源	一次性给定	一次性给定	在交互中产生（interactive）
决策过程	单步（one-shot）	无	序列（sequential）
学习目标	样本到语义标签的映射	同一类数据的分布模式	选择能够获取最大收益的的状态到动作的映射

- 基于评估：强化学习利用环境评估当前策略，以此为依据进行优化
- 交互性：强化学习的数据在与环境的交互中产生
- 序列决策过程：智能主体在与环境的交互中需要作出一系列的决策，这些决策往往是前后关联的
- 注：现实中常见的强化学习问题往往还具有奖励滞后，基于采样的评估等特点

强化学习的特点

- 根据以下特点直观定位强化学习
  - 有/无可靠的反馈信息
  - 基于评估/基于监督信息
  - 序列决策/单步决策
  - 基于采样/基于穷举



强化学习的要素

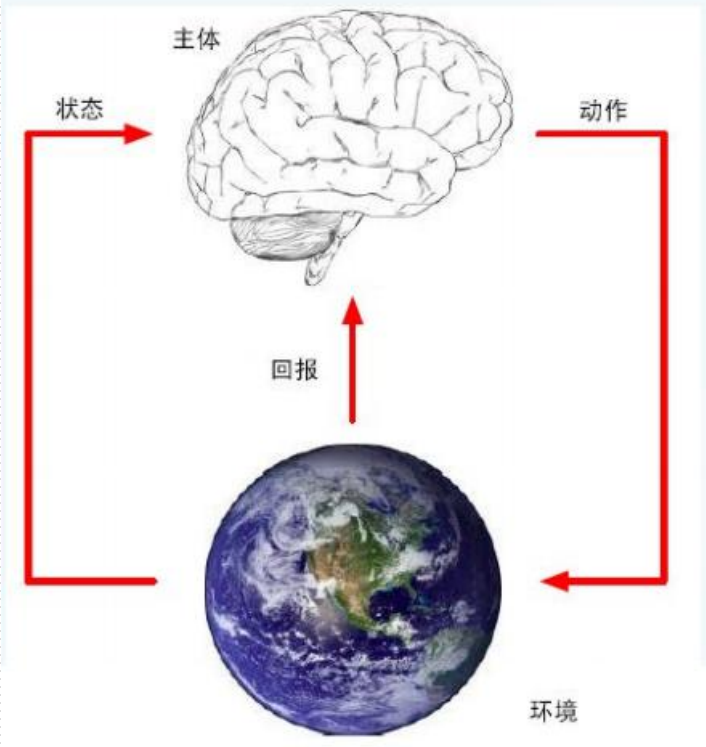
- 主体（agent）
  - 负责做出决策的实体。
  - 比如Alpha Go、人、玩flappy bird的AI
- 环境（environment）
  - 主体存在于环境之中，主体的行为作用于环境，并接受环境的反馈。比如一个完整的游戏程序
- 状态（state）
  - 环境的状态不断发生变化。不同时刻的棋盘状况、游戏画面各不相同。

强化学习的要素

- 动作（action）
  - 主体通过执行动作来改变环境的状态。
- 回报（reward）
  - 环境状态改变之后会返回主体一个回报，主体可以根据回报来判断动作的好坏

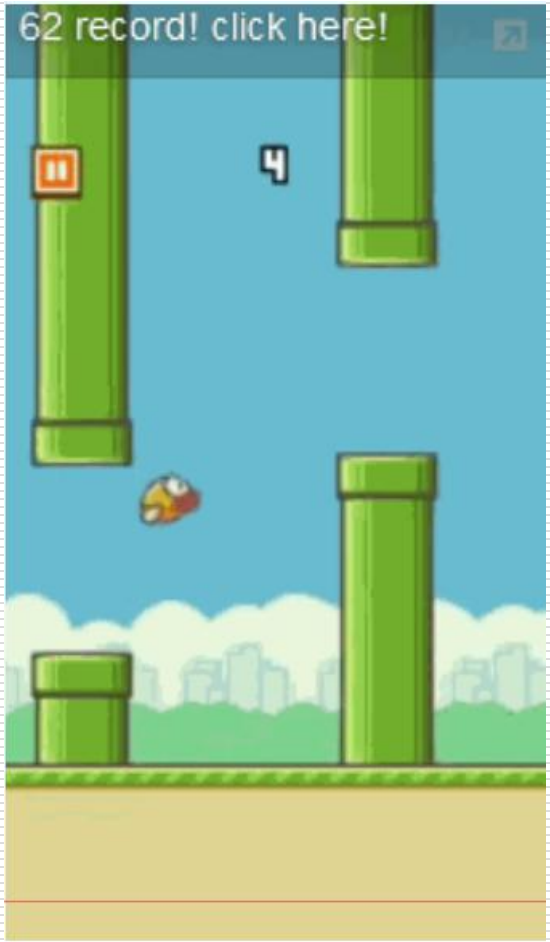
强化学习的主要流程

- 主体与环境不断地进行交互，产生多次尝试的经验，再利用这些经验去修改自身策略。
- 经过大量迭代学习，最终获得最佳策略。



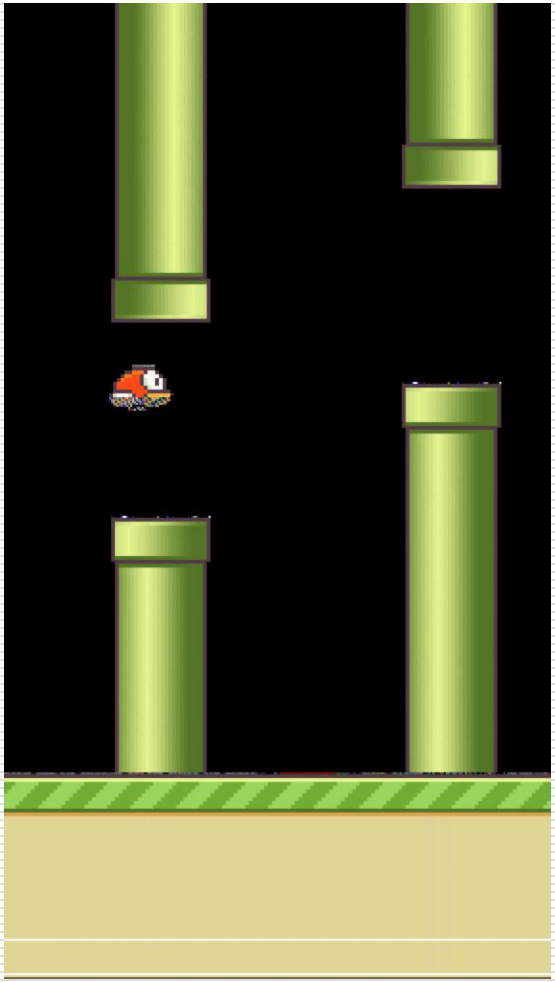
Flappy Bird

- 一款2013年5月发布的游戏，2014年1月，此游戏成为iTunes最受欢迎免费应用软件。
- 玩家操控小鸟飞行且避开绿色的管道
- 如果小鸟碰到了障碍物，游戏就会结束。每当小鸟飞过一组管道，玩家就会获得一分



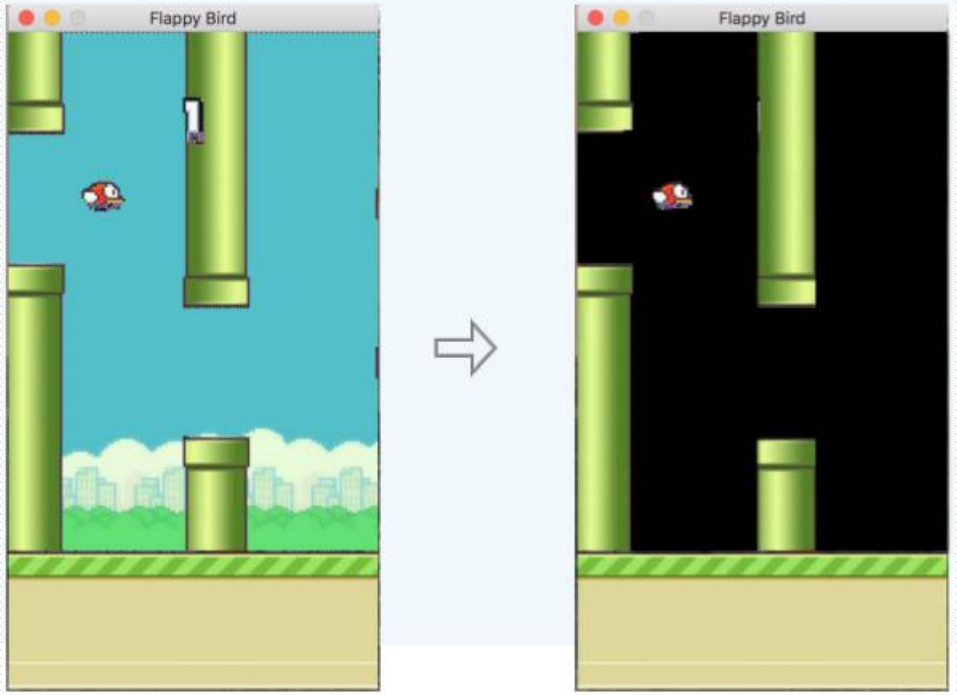
Flappy Bird

- 类似人的手眼配合学习
- 仅通过分析游戏时的截屏图像
- AI通过强化学习学会了玩 Flappy Bird



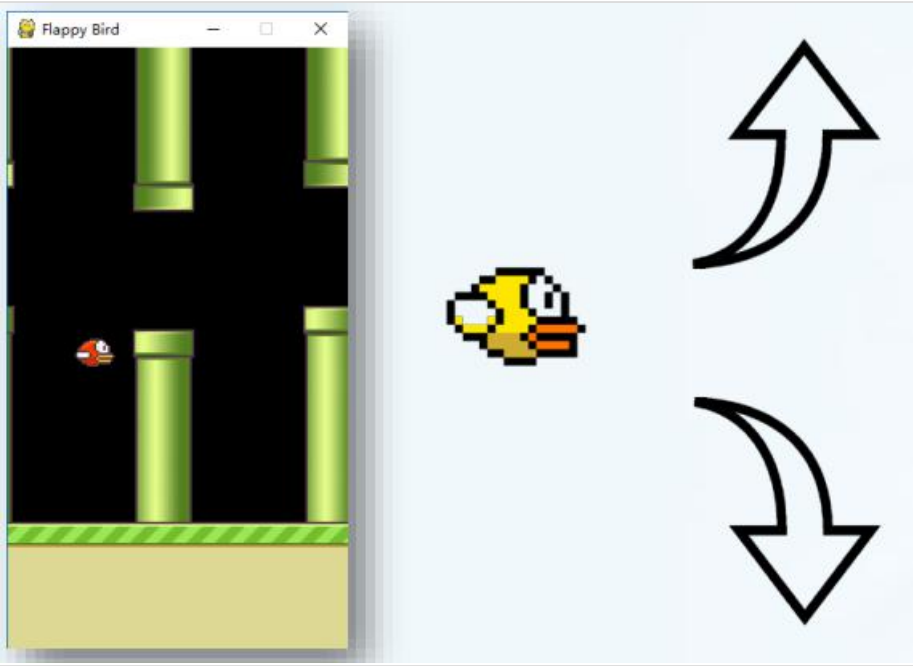
Flappy Bird 状态

- 每一帧的画面都是一个状态
- 对画面简化，保留AI用于学习的关键信息



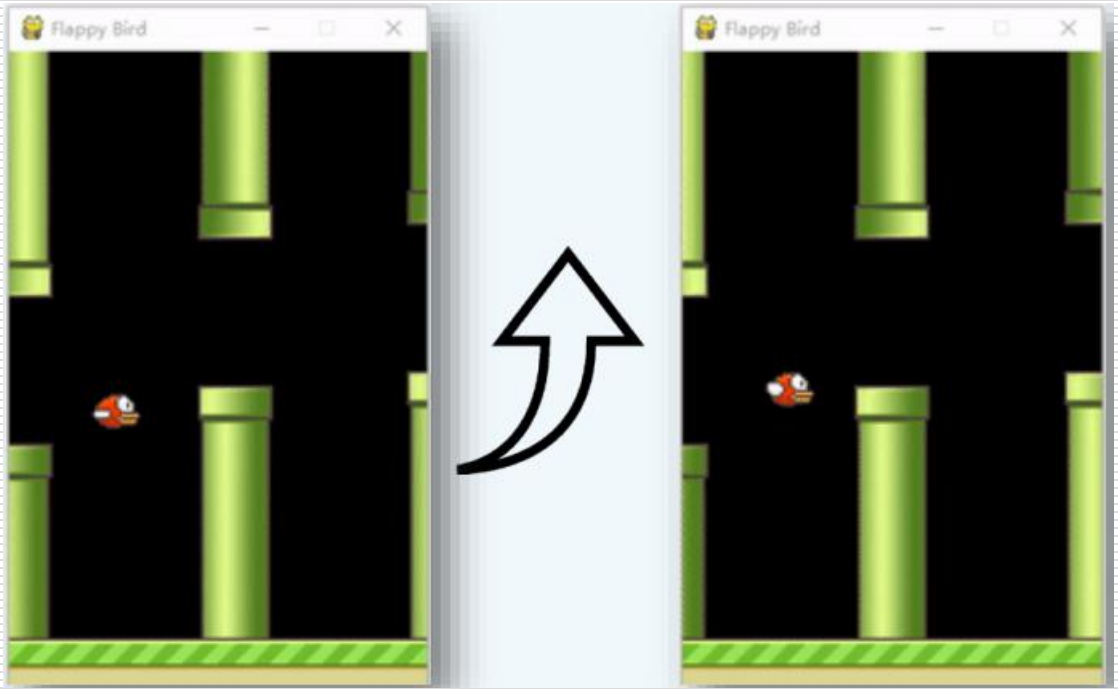
Flappy Bird 动作

- 每个状态下都有两个可选择的动作
- 让鸟往上跳 or 什么都不做



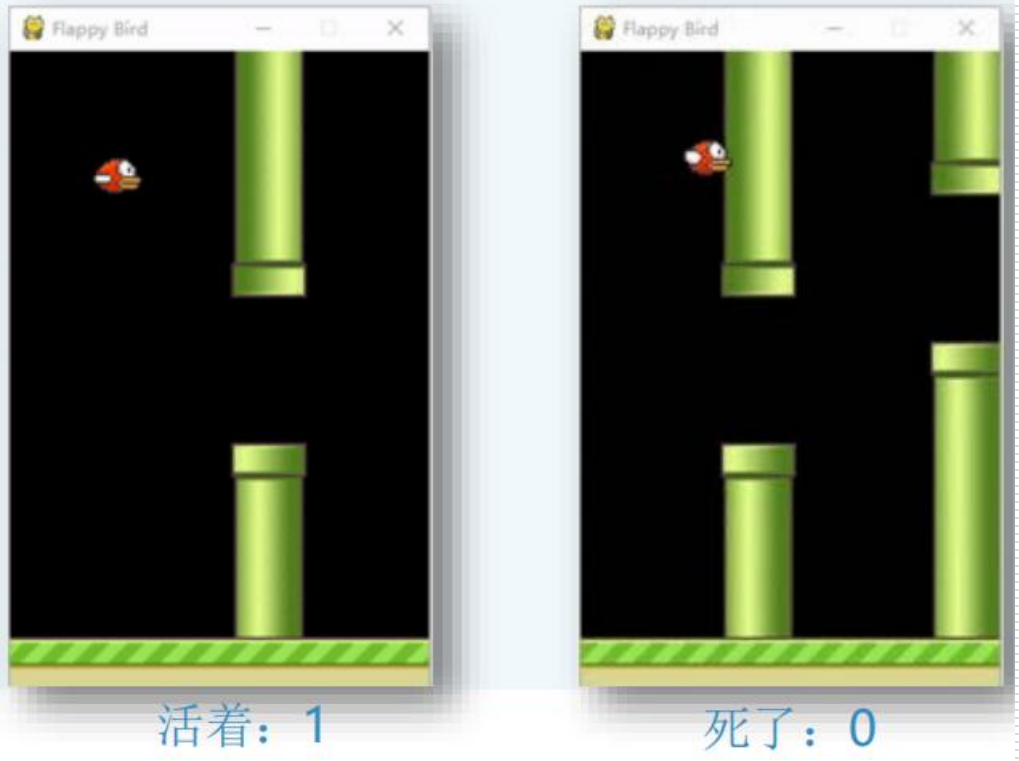
Flappy Bird 动作

- 不同的动作会产生不同的新状态



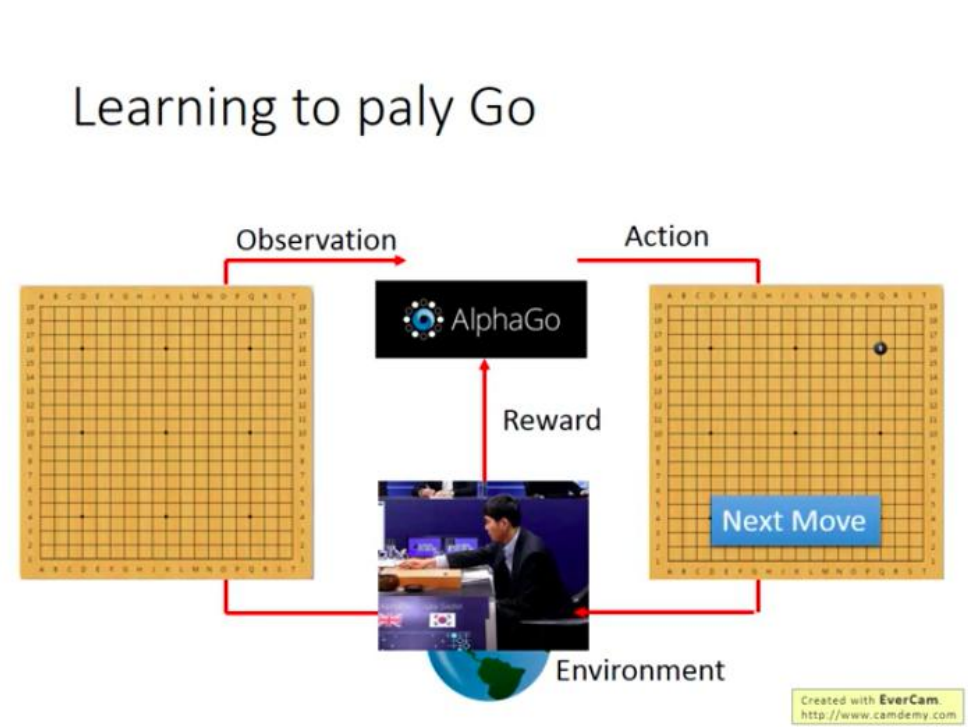
Flappy Bird 回报

- 主体在得到环境给的新状态时也会得到一个回报。简单情况下活着是1，死了是0



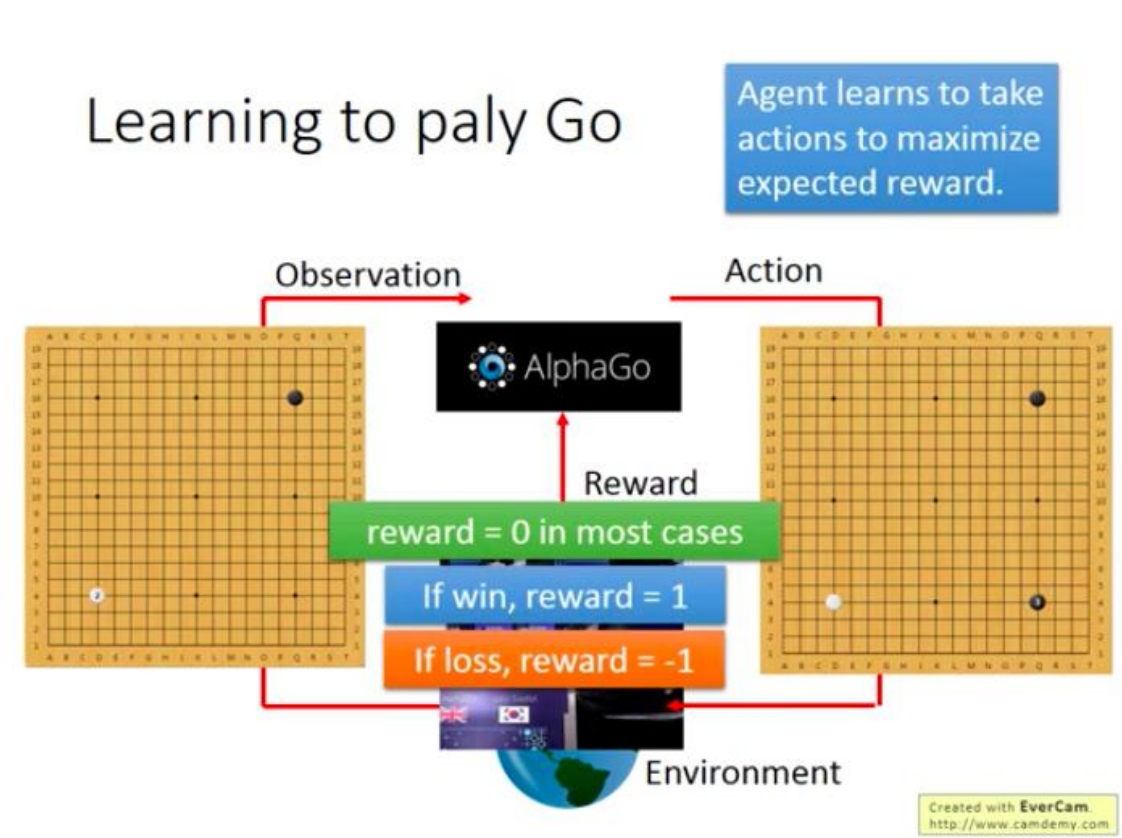
回报

- Agent学习的目标就是使得期望的回报（reward）最大化
  - 对Alphago来说，Observation就是19×19的一个棋盘，于是它落下一黑子：



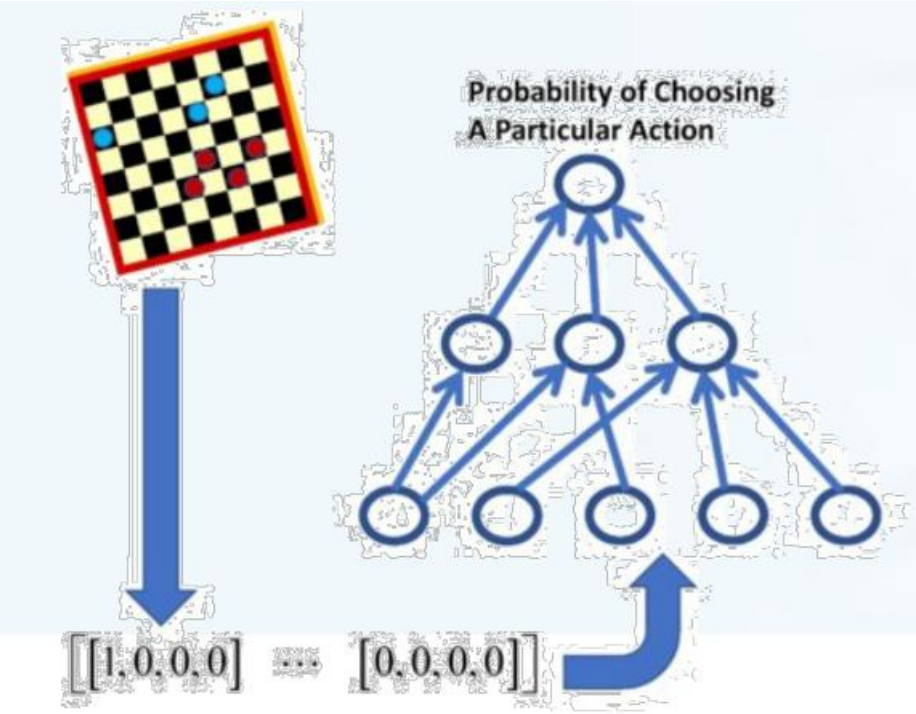
回报

- 然后对手下了一个白子，Alphago观察到一个新Observation（有两颗棋子的），再下一颗黑子：



策略

- 从状态集（所有可能出现的状态）到动作集（所有可能采取的动作）的一个对应关系



目标：求得最佳策略

- 与手写数字识别不同，在强化学习中我们不关心把当前的状态分为什么类型，而是关心它能否执行最佳动作。

判断状态

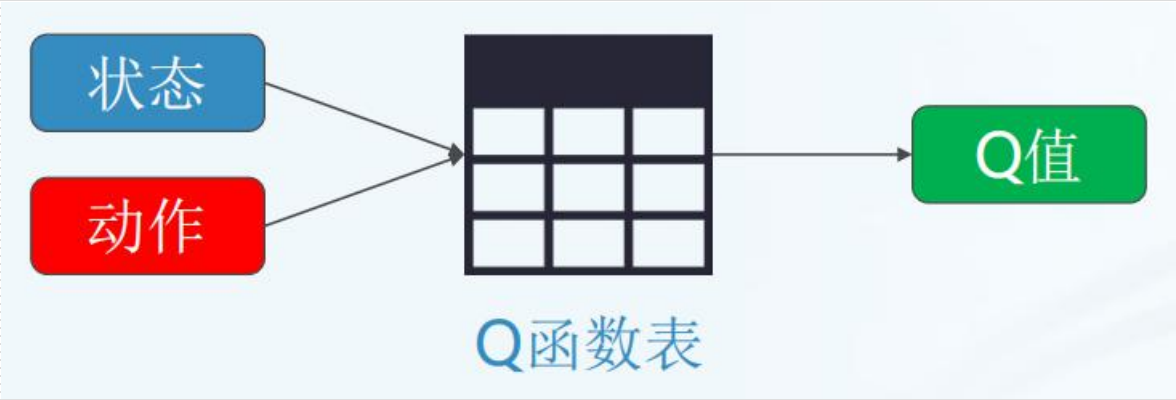
- 状态值函数V
  - 只和状态相关，用于对某个局面状态进行估值。
- 状态动作函数Q
  - 和状态以及在该状态下采取的动作相关，用于对某个局面状态下采取某个动作进行估值。

Q-Learning

- 强化学习中一种常用算法
- 基于状态动作函数Q，如果知道了某一状态下每个动作的估值，那么就可以选择估值最好的一个动作去执行了

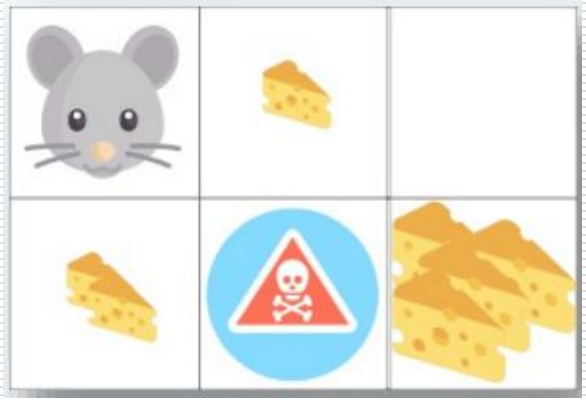
简单的Q函数表（Q-Table）

- Q函数表中行表示状态，列表示动作，表中的值表示特定状态下执行某动作的评估值Q
- 主体通过不断更新并查找该表，找到当前状态回报最高的动作执行



简单的Q函数表（Q-Table）

- 示例

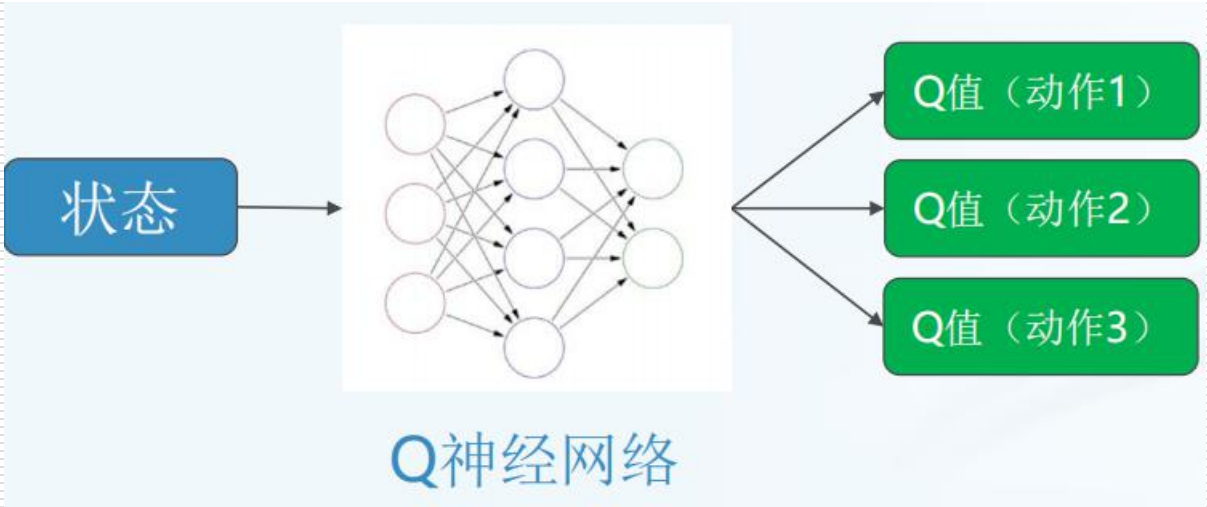


- 某个策略的Q函数表

状态\动作	上	下	左	右
开始点	0	20	0	10
一小块奶酪	0	-100	1	2
空白	0	100	10	0
两小块奶酪	5	0	0	-100
毒药	0	0	0	0
一堆奶酪（终点）	0	0	0	0

基于神经网络计算Q函数

- 对于复杂的状态，无法用表格表示，可使用神经网络对Q函数进行建模，其输入为状态，输出为各个动作的评估值。还是选取最高的动作执行



ε-greedy

- 智能主体的策略（即按照动作-价值函数选择反馈最大的行为）始终不变，因此与环境交互的轨迹是固定的
  - 外力：缺乏推动智能主体改变策略的外在因素
  - 内因：智能主体缺乏从内部改变策略的动力
- 智能主体的“创新精神”：
  - 根据目前已知的最优策略来选择动作，被称为开发（exploitation）
  - 不根据当前策略而去尝试未知的动作被称为探索（exploration）

ε-greedy

- 探索（Exploration）
  - 为了更好地学习最佳Q函数而尝试各种情况
  - 也就是说，应该选择与当前不同的其它动作
- 开发（Exploitation）
  - 直接选择当前认为最佳的动作
  - 再进一步修改新状态下的Q值

ε-greedy

- 从零开始
  - 刚开始时并不知道正确的策略以及Q函数应该是多少
  - 初始化一个随机Q函数，从零开始不断学习
- 如何尝试
  - 在Q函数不够准确的时候，每次尝试该如何选择动作？
  - 涉及到探索和开发两者的平衡
    - 只开发而不探索 ✖
    - 只探索而不开发（则训练过程完全没有意义） ✖
    - 大体上开发，偶尔探索 √

ε-greedy

- 探索与开发
  - 探索：随机的生成一个动作
    - 探索未知的动作会产生的效果，有利于更新Q值，获得更好的策略。
  - 开发：根据当前的Q值计算出一个最优的动作（greedy）
    - 相对来说就不好更新出更好的Q值，但可以得到更好的测试效果用于判断算法是否有效

ε-greedy

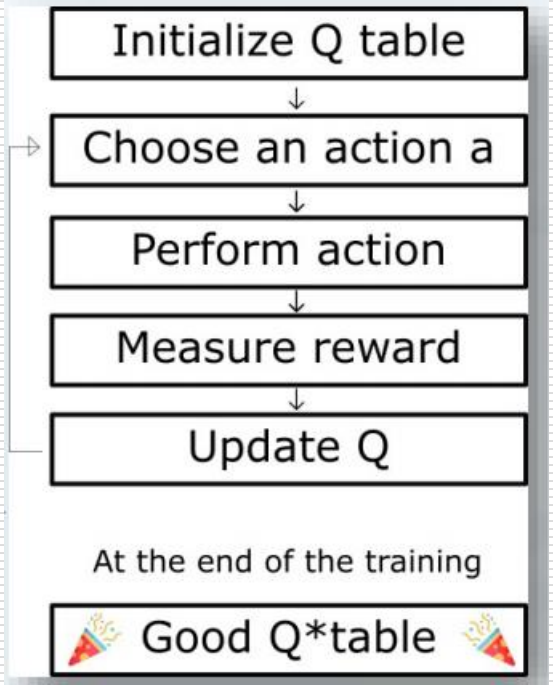
- 一种简单的平衡探索与开发的策略
- 有ε概率选取一个随机动作，剩下的情况依然选取Q值最大的动作。
  - ε一般是一个很小的值，表示不断尝试的趋势

e-greedy

- 可以更改ε的值从而得到不同的探索和开发的比例
  - 通常在刚开始学习时，可以稍微调大（0.01）
  - 在Q函数逐渐优化后，可以调小（0.001）
  - 甚至可以直接设为0，不再探索。

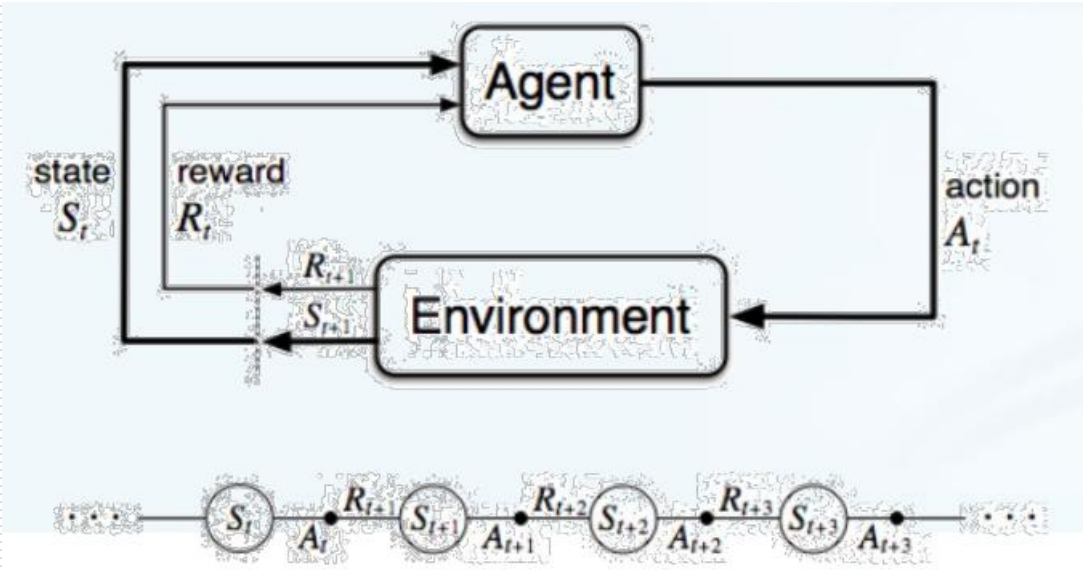
学习流程

- 初始化Q函数
- 不断重复每一局游戏
  - 选择动作
  - 得到回报
  - 更新Q函数
- 最终得到一个好的Q函数



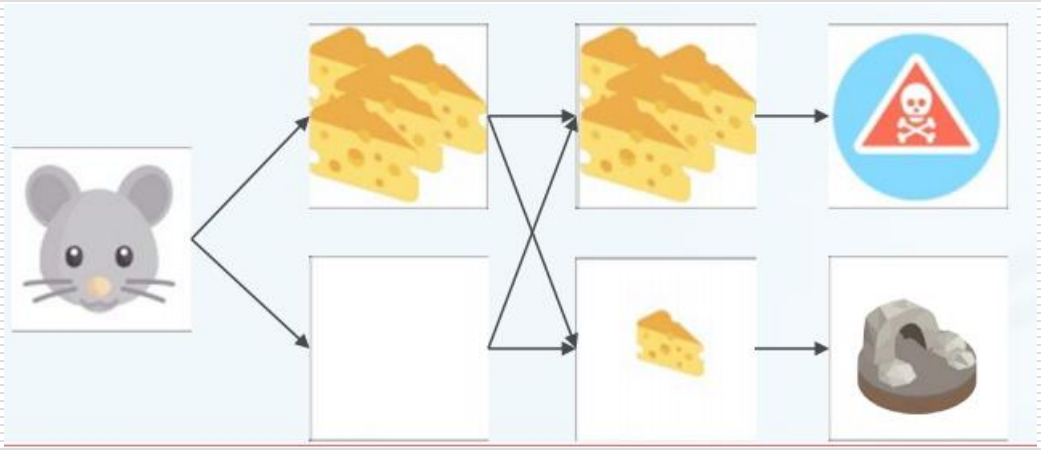
动作-状态序列

- 每一局游戏都是一个动作状态序列
- 下一个状态只和当前的状态+动作有关（马尔可夫性质）



长期回报

- 除了试错式搜索之外，强化学习的另一个重要的特点是回报的滞后性。
- 当前状态下的动作所产生的回报不仅取决于下一个状态，还取决于整个序列之后的每一个状态

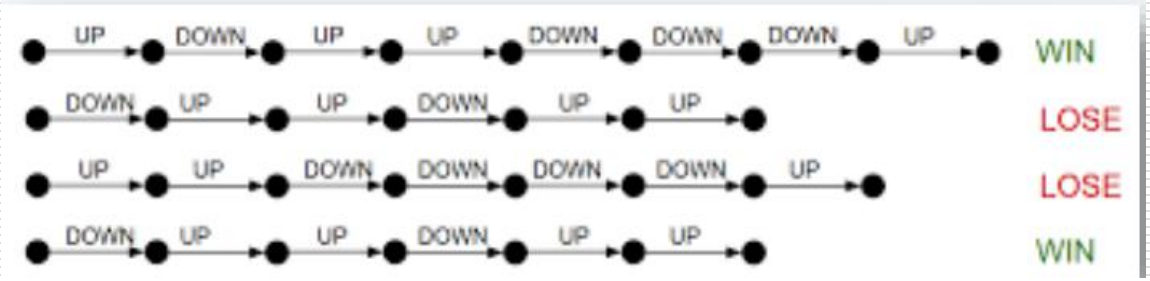


回报率

- 当前的动作对下一状态的影响是最直接的，对后续状态影响没那么直接
- 某些动作产生的当前回报值比较高，但从长远来看，可能并没有那么高
- 因此我们用一个回报率来平衡下一状态回报和更远状态回报。

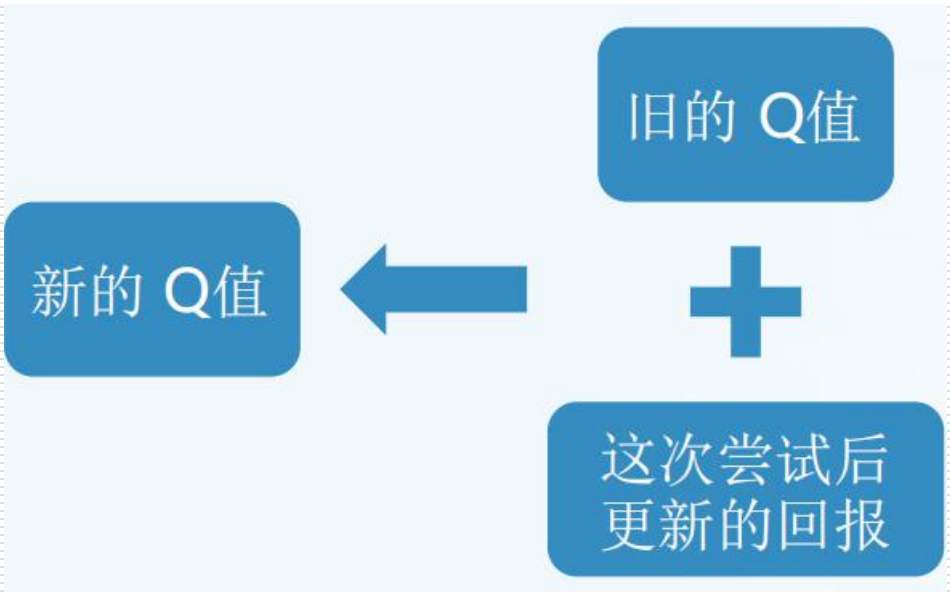
回报函数

- 每一次游戏会产生不同的状态动作序列，即每一次对后续状态的回报计算都不相同
- 我们用后续状态的期望，即所有之后的序列的回报平均值作为回报函数
- 回报函数值就是Q值



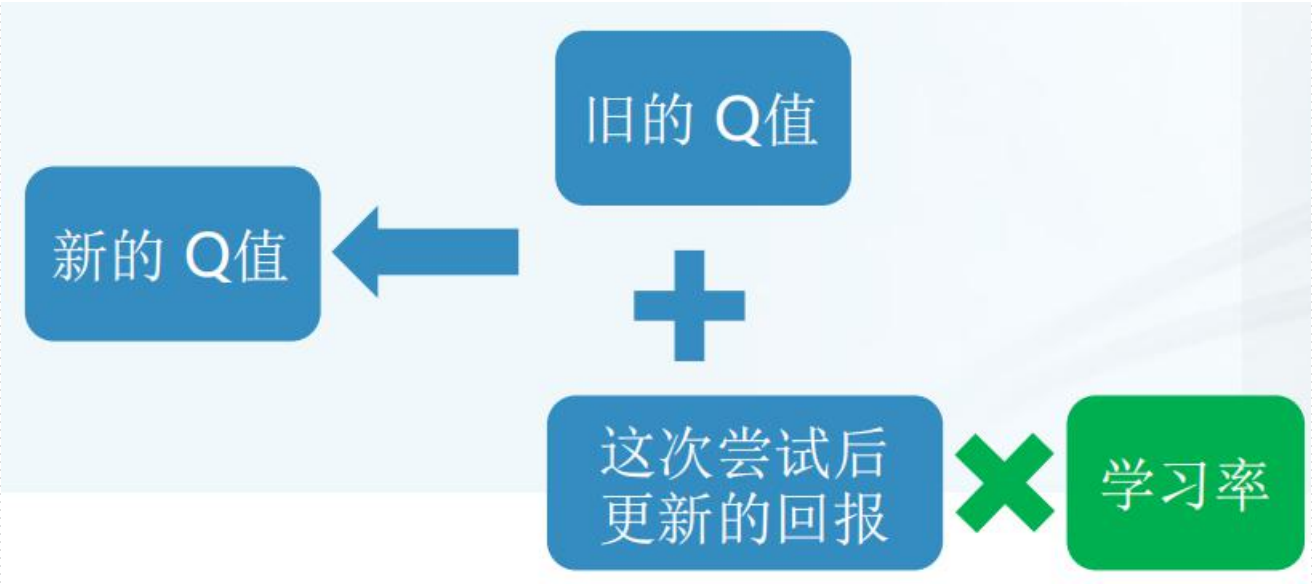
学习过程

- 每完成一局之后，就持续更新Q函数
- 完成的局数越多，更新的次数就越多，结果也越准确



学习率

- 既要利用好已经学好的值，也要善于学习新的值
- 这两者就通过学习率来平衡，一开始学习率可以大一些，最后稳定时学习率可以小一些



熟能生巧

- 通过上述公式学习，在足够多的尝试之后，AI所学到的状态动作值函数Q就能够达到一个较优的结果
- 再根据这个Q函数来选择动作，就“熟能生巧”了

Q学习算法

- Q学习算法是一种用来解决马尔可夫决策过程中最优化问题的方法。
- Q学习算法最大的特点是它具有选择瞬时奖励和延迟奖励的能力。
  - 在每一步中，agent通过观察状态s的向量，然后选择并执行行动a，随着状态从s转移到s‘，agent能够收到一个强化值r（s， a）。
  - 训练的目标是发现一条行动的路径，从而使得整个过程强化值的和最大，也就是从起点到终点间的一条最短路径。

Q学习算法

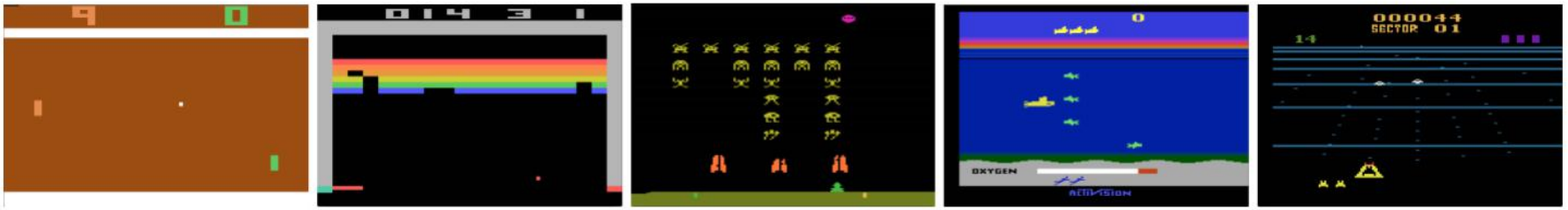
- Q学习算法的转移规则表示为以下形式：
  - Q (state， action) = R(state， action) + gamma * Max[Q(next state， all actions)]
    - 参数gamma的范围是[0，1]，从而保证结果收敛。
    - 如果gamma更接近0，agent趋向于只考虑瞬时奖励值，反之如果更接近1，则agent为延迟奖励赋予更大的权重，更侧重于延迟奖励

Q学习算法的步骤

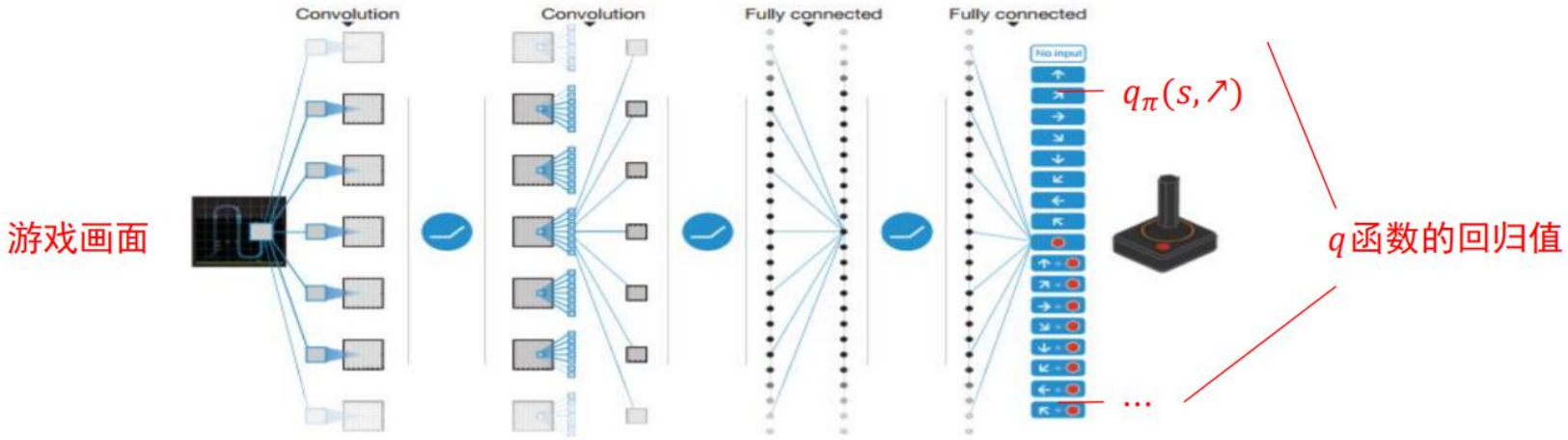
- 设置gamma以及环境奖励矩阵R；
- 将矩阵Q初始化为0；
- 对于每次迭代（episode）：
  - 随机选择一个初始状态；
  - 直到达到目标状态：
    - 从所有可能的行动中选择一个行动；
    - 执行行动到达下一状态；
    - 获取下一状态所有行动中最大的Q值；
  - 利用公式更新Q值；

深度Q学习的应用实例

- 雅达利游戏



卷积神经网络



q函数的学习模型

Mnih, Volodymyr, et al, Human-level control through deep reinforcement learning, Nature 518.7540 (2015)

深度Q学习的应用实例

- Q值神经网络化，也就是DQN

我们使用Q-learning为QNetwork提供有标签的样本$R_{t+1} + \lambda \max_a Q(S_{t+1}, a)$,利用Reward和Q计算出来的目标Q值。所以损失函数可以表示为

$$L(w) = \mathbb{E}[(r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w))^2]$$

其中$r + \gamma \max_{a'} Q(s', a', w)$为target

FlappyBird

- Deep Q-Network Algorithm

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

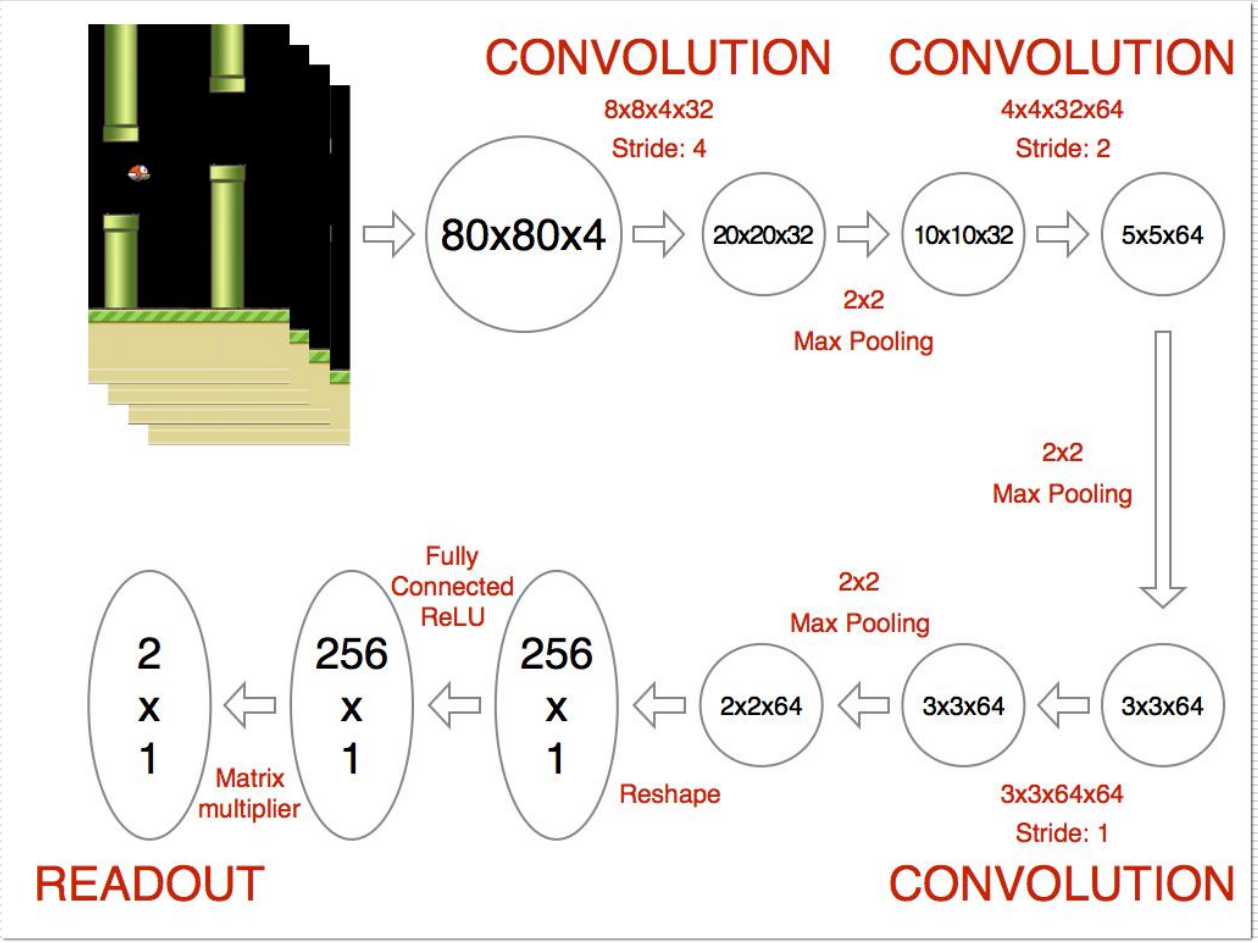
FlappyBird——Network Architecture

- 输入值预处理
  - 把图像大小resize成80x80
  - 把图像转换成灰度图
  - 把图像二值化，只有黑白两色0或者255。
  - 把连续的四帧图像作为一个输入（State）

```
x_t = cv2.cvtColor(cv2.resize(x_t, (80, 80)), cv2.COLOR_BGR2GRAY)
ret, x_t = cv2.threshold(x_t,1,255,cv2.THRESH_BINARY)
s_t = np.stack((x_t, x_t, x_t, x_t), axis=2)
```

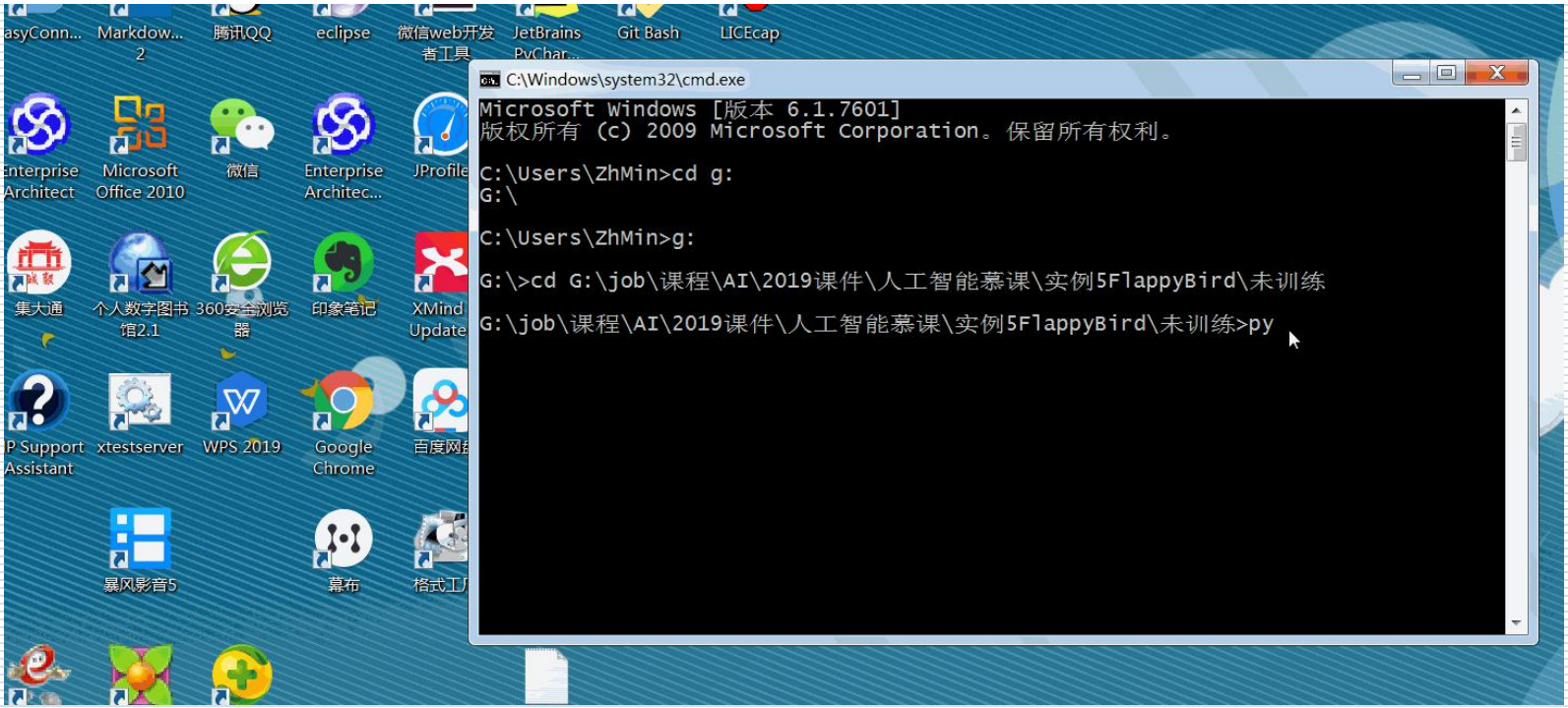
FlappyBird——Network Architecture

- 构建智能体——一个6层的神经网络



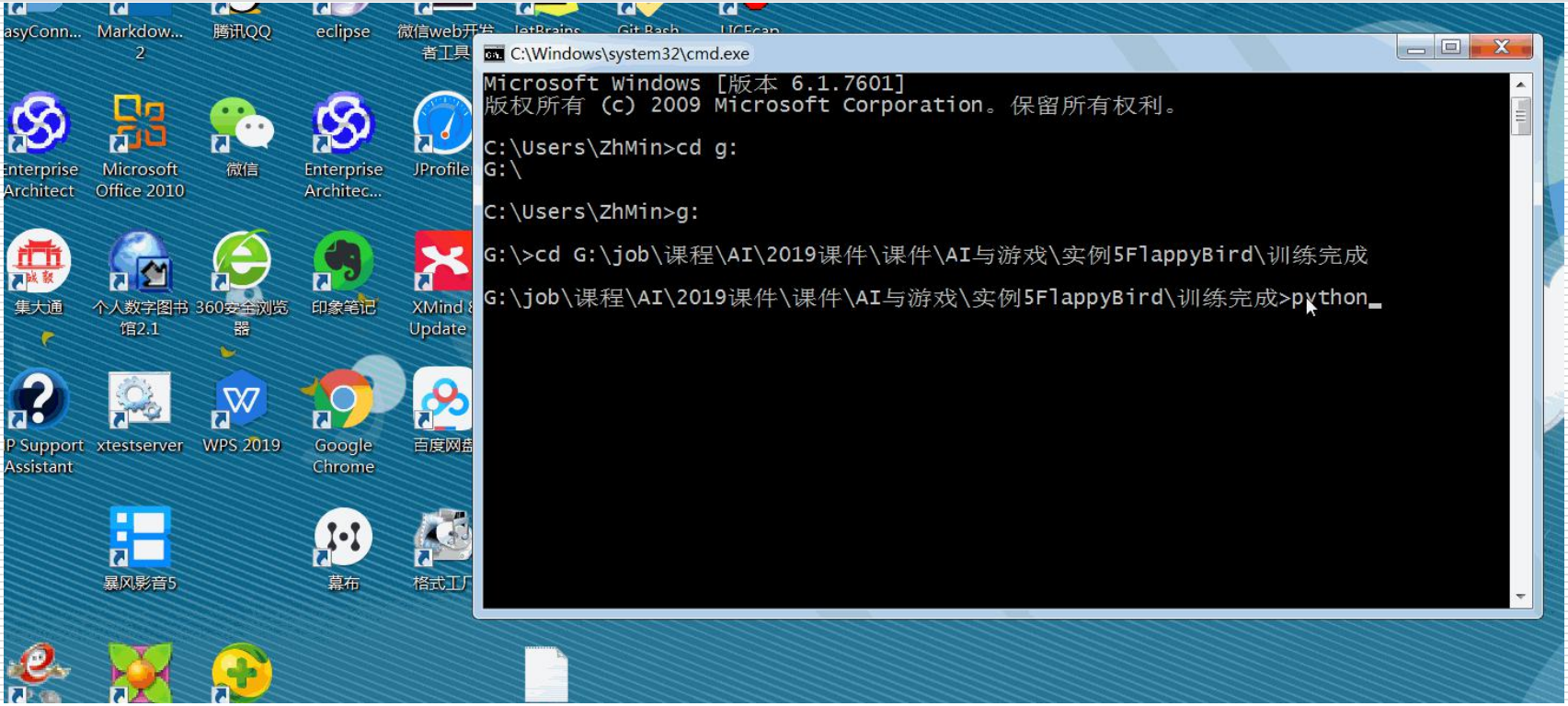
Example：FlappyBird

- 未训练



Example：FlappyBird

- 训练



总结

- 价值函数是强化学习的核心，比如在深度 Q 网络及其许多扩展中。
- 策略优化方法已经在许多不同的应用领域得到了关注
  - 比如：机器人、神经架构设计、口语对话系统、机器翻译、注意（attention）和学习去学习（learning to learn）等等，不能胜举。
- 新的学习机制也在涌现
  - 比如：使用无监督/半监督/迁移学习来提升学习的质量和速度，而且更多的新机制还将涌现。这是强化学习的复兴（Krakovsky, 2016）。
- 事实上，即使是在「人工智能的冬天」，强化学习和深度学习也在不断发展进步

THE END