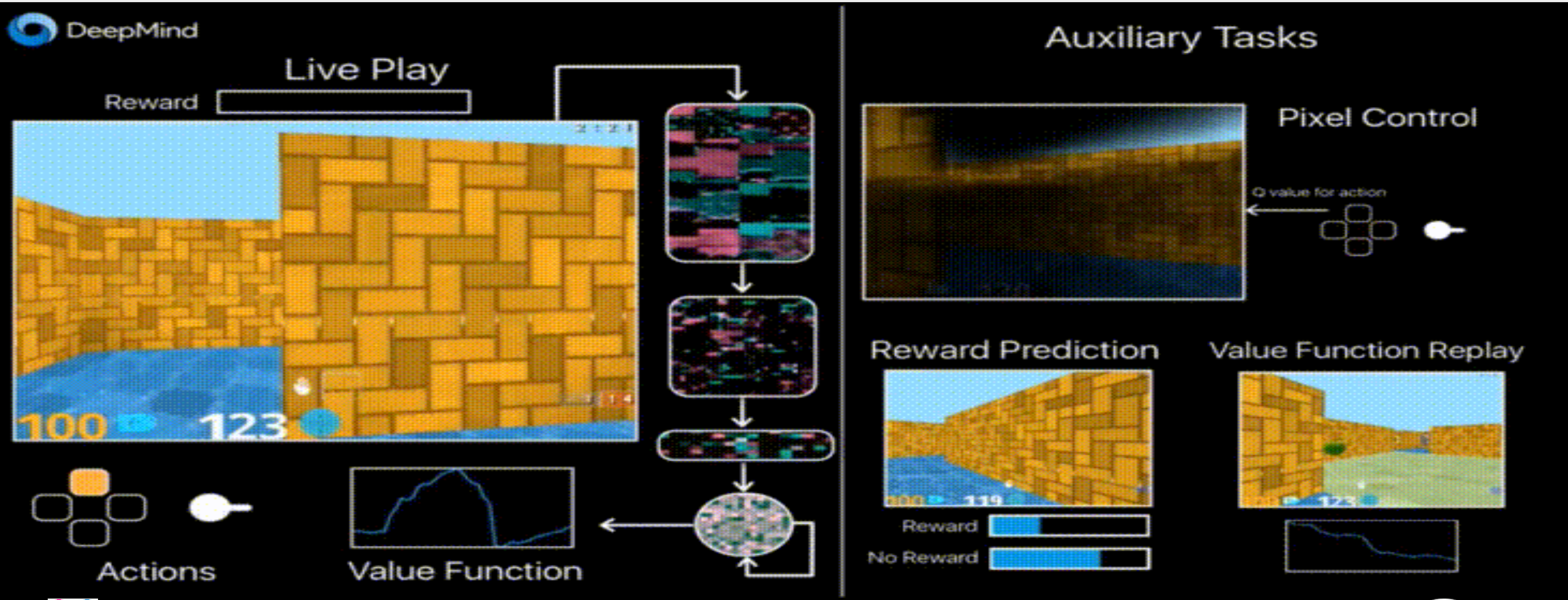
# 数据驱动的人工智能（7b）强化学习

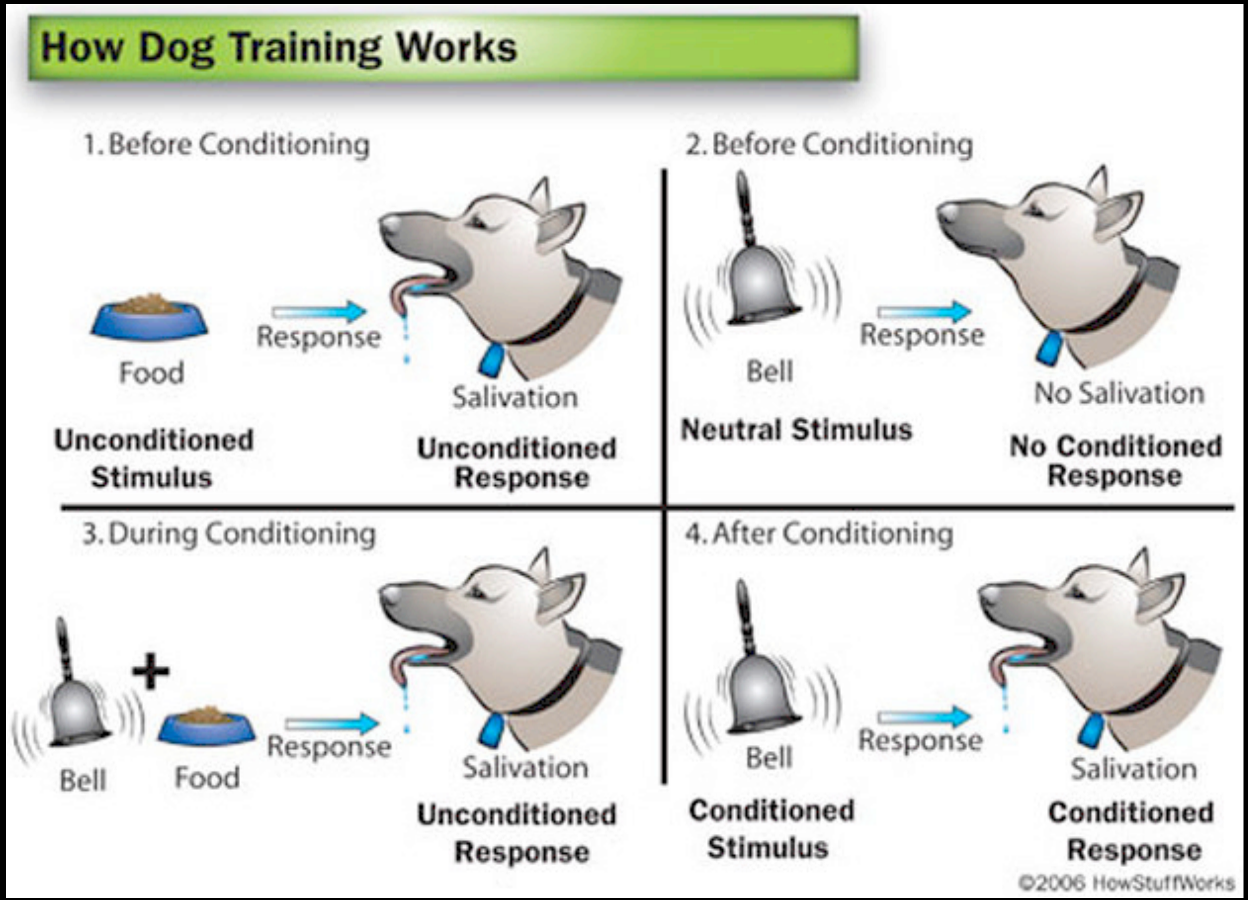
## Data Driven Artificial Intelligence

邬学宁 SAP硅谷创新中心

2017 / 04



# Learning doesn’t occur from a mistake happening, but from when the result differs from your expectation.





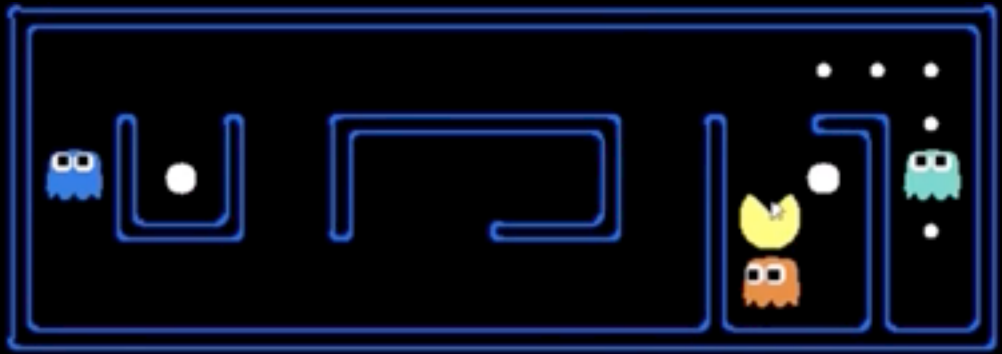
RL是动物心理学，控制论与AI的结合

## How Much Information Does the Machine Need to Predict?

Y LeCun

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Unsupervised/Predictive Learning (cake)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

# “ 定义1



Reinforcement learning problems involve learning what to do - how to map situations to actions – in order to maximize a numerical reward signal

Principles:

- Learn to make good decisions in unknown - dynamic environments
- You have a agent that “explores“ some space
- 从经验中学习: Reward
- 在遍历了整个空间之后，Agent学习到了在不同的状态下，采取不同行动，所获得的Reward的期望
- 例如: 学骑单车,下棋,自治机器人 , 在一个随机波动的市场中...

# 定义2

• Environment: can be stochastic (Tetris), adversarial (Chess), partially unknown (bicycle), partially observable (robot)
• Available information: the reward (may be delayed)
• Goal: maximize the expected sum of future rewards.

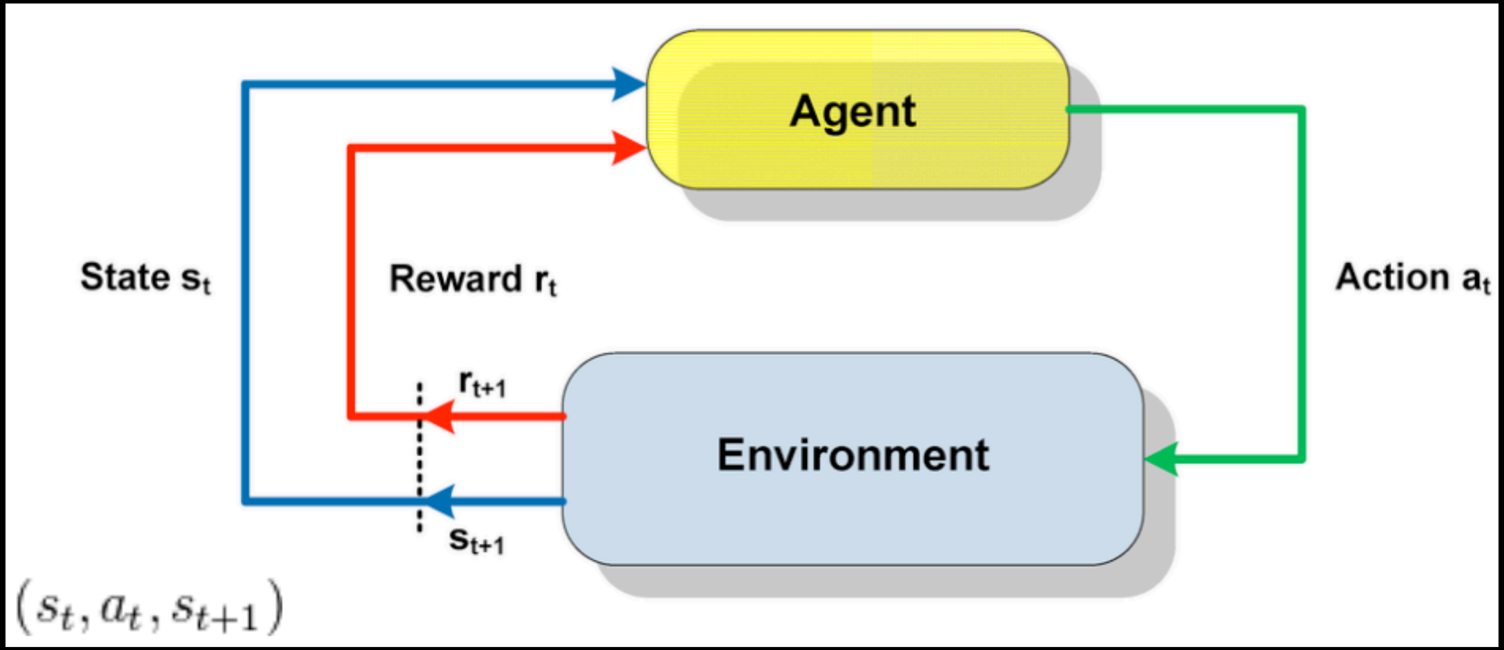
Fully Observable


Not Fully Observable

问题／目标：
如何牺牲短期回报以获得长期更大的回报？

# Exploration vs. Exploitation

- Exploitation: make the best decision given current information
- Exploration: gather more information
- 最佳的长期策略可能需要短期的牺牲
- 收集足够的信息进行最佳的整体决策



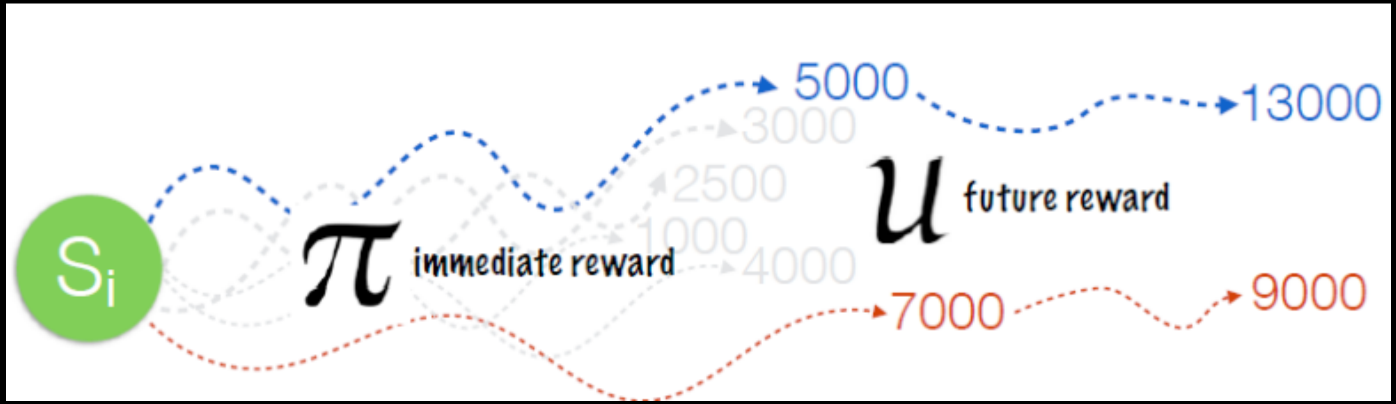
# 主要元素

- Policy
    - Reward Function
    - Value Function
- 环境
- Given:
    - S: Set of Status
    - A: Set of Action
    - T: T(s,a,s’) Transitional Model
    - R: Reward Function

- Find

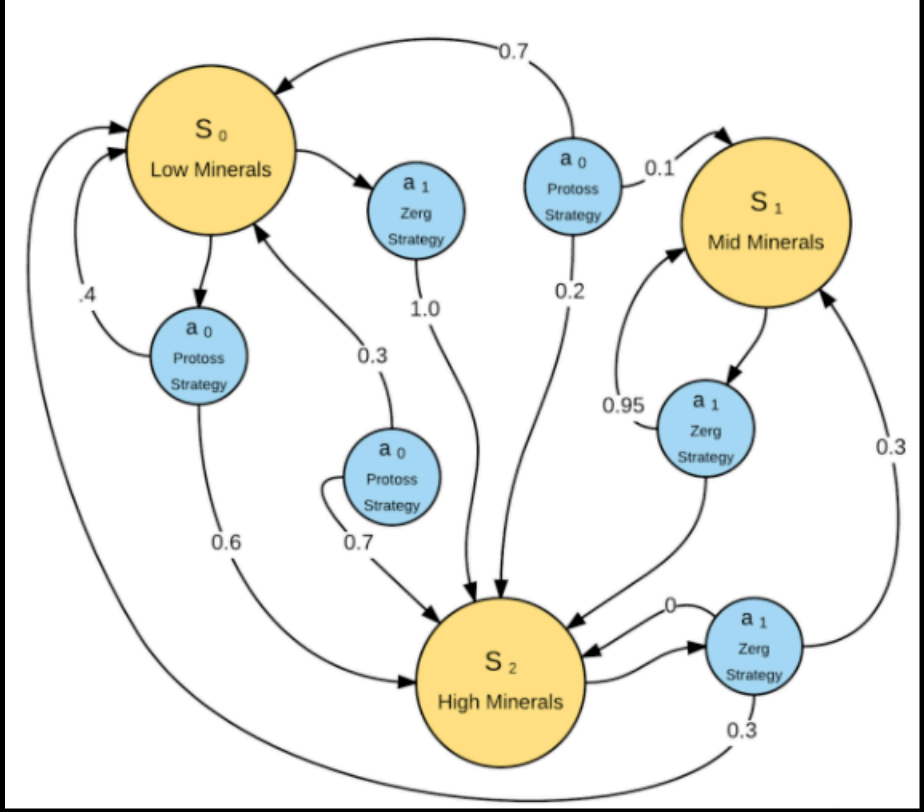
  $\pi(s)$: a policy that maximize

  $V^{\pi}(S_t) = \sum_{i=0}^{\infty} \gamma^i r_{t+1} \quad 0 \leq \gamma \leq 1$



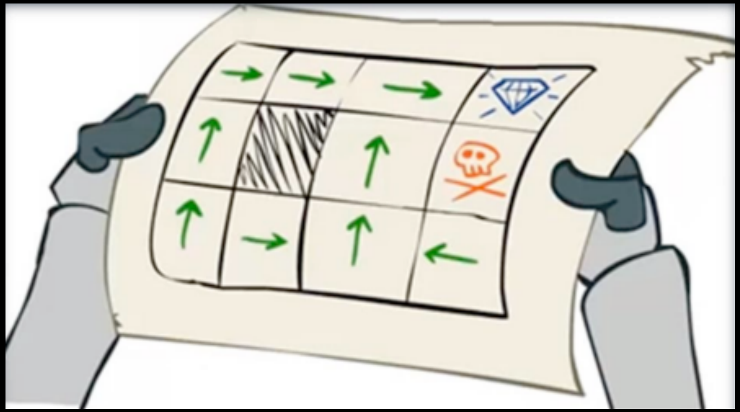
# Markov Decision Process (MDP)

- MDP is a 5-tuple $(S, A, T, R, \gamma)$, where
  - S is set of states $s \epsilon S$
  - A is set of actions $a \epsilon A$
  - T is the Transition function T(s,a,s')
    - Probability that a from s will lead to s'
    - Equivalent to $P$ ($s$ '|s, a) $s$ = $state$, $a$ = $action$
  - R is the Reward function R(s,a,s')
  - $\gamma$ is the discount factor, in which $\gamma$ $\epsilon$ [0,1]
- Action should depend only on the current state!
- 是一个对决策进行建模的数学框架
- 决策结果一部分有决策者决定，一部分随机
- An MDP is a *discrete time stochastic control process*

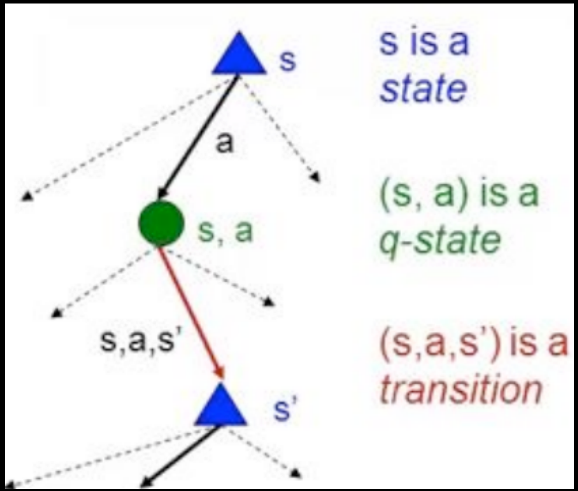



Example of a MDP of Starcraft

# MDP Optimal Policy



- For MDP's we want an optimal policy $\pi^*: S \rightarrow A$
  - A policy $\pi$ gives an action for each state $a_t = \pi(s_t)$
  - An optimal policy is one that maximizes expected value reward
  - $R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \qquad \gamma \in [0,1]$
  - Value Function 是对未来Reward的预测（How much reward will get from action a in status s）
- 状态s的Value： $V^*(s)$ = expected value starting in s and acting optimally
- Q-value函数给出总Reward的期望值：
  - 从state-action pair (s,a)开始
  - 在policy $\pi$ 之下
  - $Q^\pi(s,a) = \mathrm{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s, a]$
- 最优的Value Function是值的最大化
  - $Q^*(s,a) = Max_a Q^\pi(s,a)$
- 找到了$Q^*$，就能确定优化的动作a了



# Q-Function

- Discount Future Reward: $R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-t} r_n$

也可以被写成：

- $R_t = r_t + \gamma R_{t+1}$

Q-Function 的定义是在状态s选择动作a最大化Reward

- $Q(s_t, a_t) = Max(R_{t+1})$

所以可以将Q-Function重写为Bellman Equation：

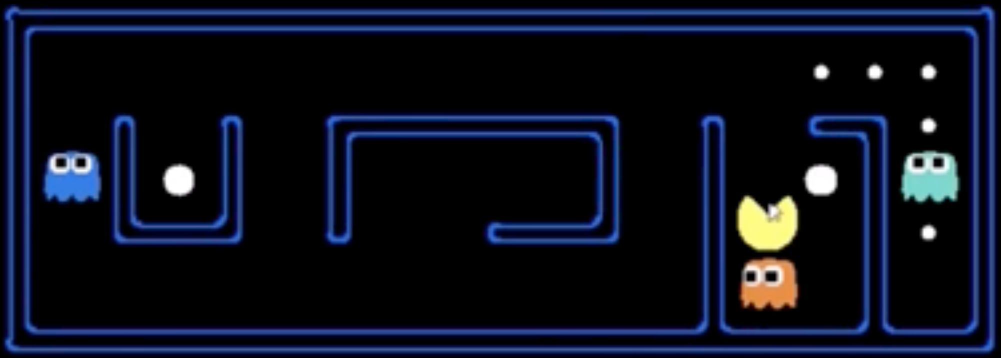
- $Q(s,a) = r + \gamma \cdot Max_{a'} Q(s', a')$

即：最大化$(s,a)$未来的Reward等于立即Reward $r$ + 折扣后的下一个状态和动作$(s', a')$未来Reward的最大化，可以通过Dynamic Programming或迭代来解决。

SAP

复旦大学 FUDAN UNIVERSITY 1905

# Q-Learning



- 强化学习的一种无模型实现
- 我们有
  - 环境状态的集合s
  - 在那些状态可能的动作a
  - 对于每个状态／动作所对应的值Q
    - 左边有堵墙
    - 下面有妖怪
    - 上面和右边，继续活着
- 初始状态：Q=0
- Explore空间
- 通过Reward的正负，来增减Q
- 在计算Q时，可以通过”discount factor” 来看2步以上

$$\begin{bmatrix} \boxed{0 \quad 0 \quad 0 \quad 0} \\ 0 & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Q-TABLE

$$\begin{array}{cc} & \text{ACTIONS} \rightarrow \\ \text{STATES} \downarrow & \begin{bmatrix} 0.1 & 0.4 & 1.1 & 0 \\ 0 & 0.2 & 0.1 & 0.5 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 1.2 & 0.7 & 0 & 0 \end{bmatrix} \end{array}$$

$$\Delta Q(s,a) = Q'(s,a) - Q(s,a)$$

$$Q(s,a) = Q(s,a) + \eta \Delta Q$$

SAP

复旦大學 FUDAN UNIVERSITY 1905

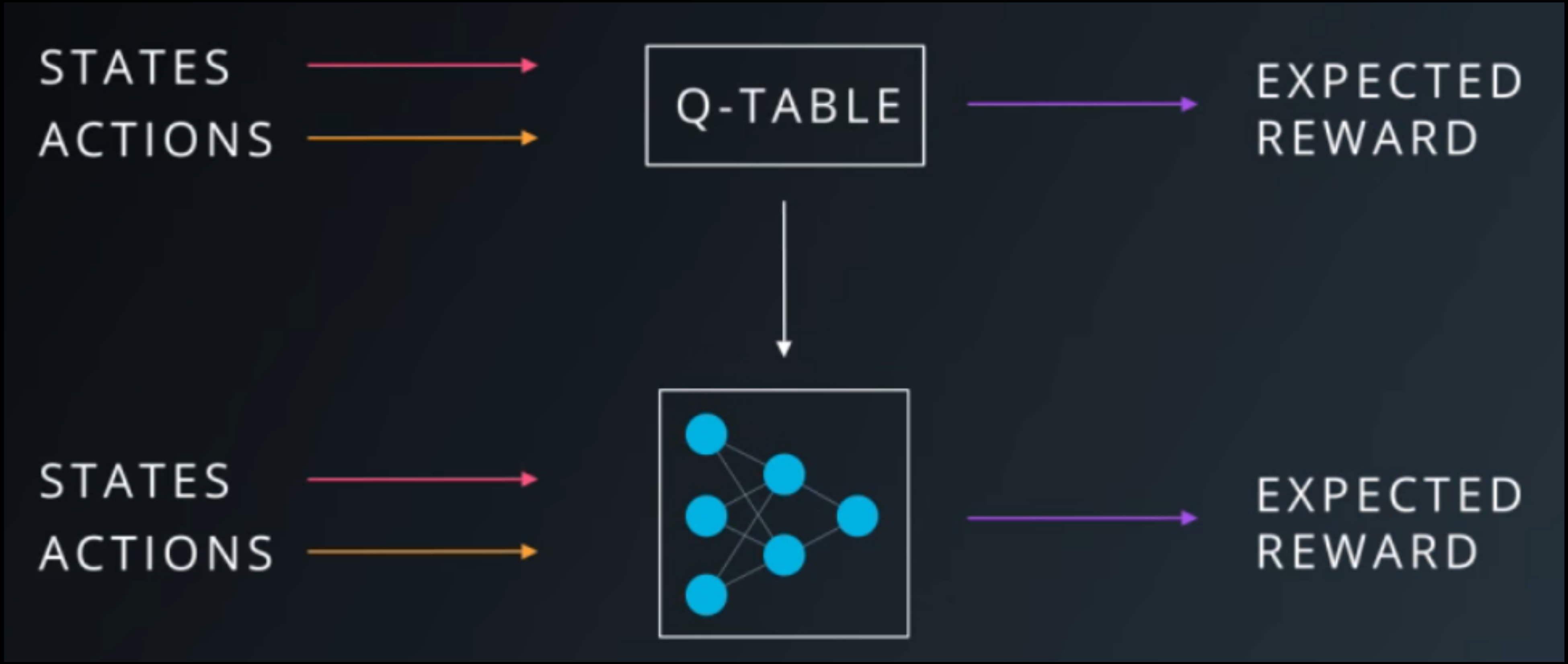
# “ The Exploration Problem

- 如何有效的遍历所有可能的状态呢？
  - 简单方法：对于某给定的状态，总是选择Q最大的Action
    - 效率很低，可能错过很多路径
  - 更好的方法：引入ε项
    - 如果随机值小于ε，不采用Q值最大的Action，进行随机选择
    - Exploration 从不完全停止
    - 如何选择ε很难

# 不同的实现方法

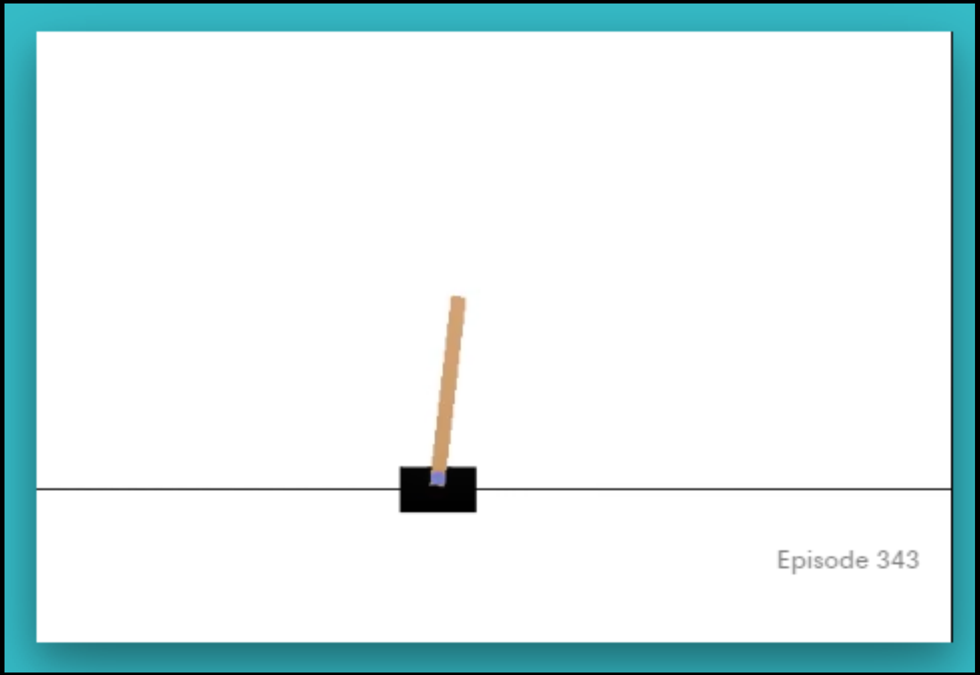
- RL实现的3种方法
  - Dynamic Programming
  - Monte Carlo Methods
  - Temporal-difference Learning
- 三种方法都能解决问题，只是效率与收敛速度不同
- TD Learning是前两种方法的结合
- TD 的核心思想是：参考一个更准确的预测值来调整预测值

# Deep Q-Network (DQN)



# Deep Q-Network (DQN)

- https://gym.openai.com/
- https://gym.openai.com/envs/CartPole-v0



# Thank you!

Contact information:

邬学宁 (i025497)
Chief Data Scientist, SAP Silicon Valley Innovation Center
Address: No. 1001, Chenghui Road, Shanghai, 201023
Phone number: +8621-6108 5287
Email: x.wu@sap.com