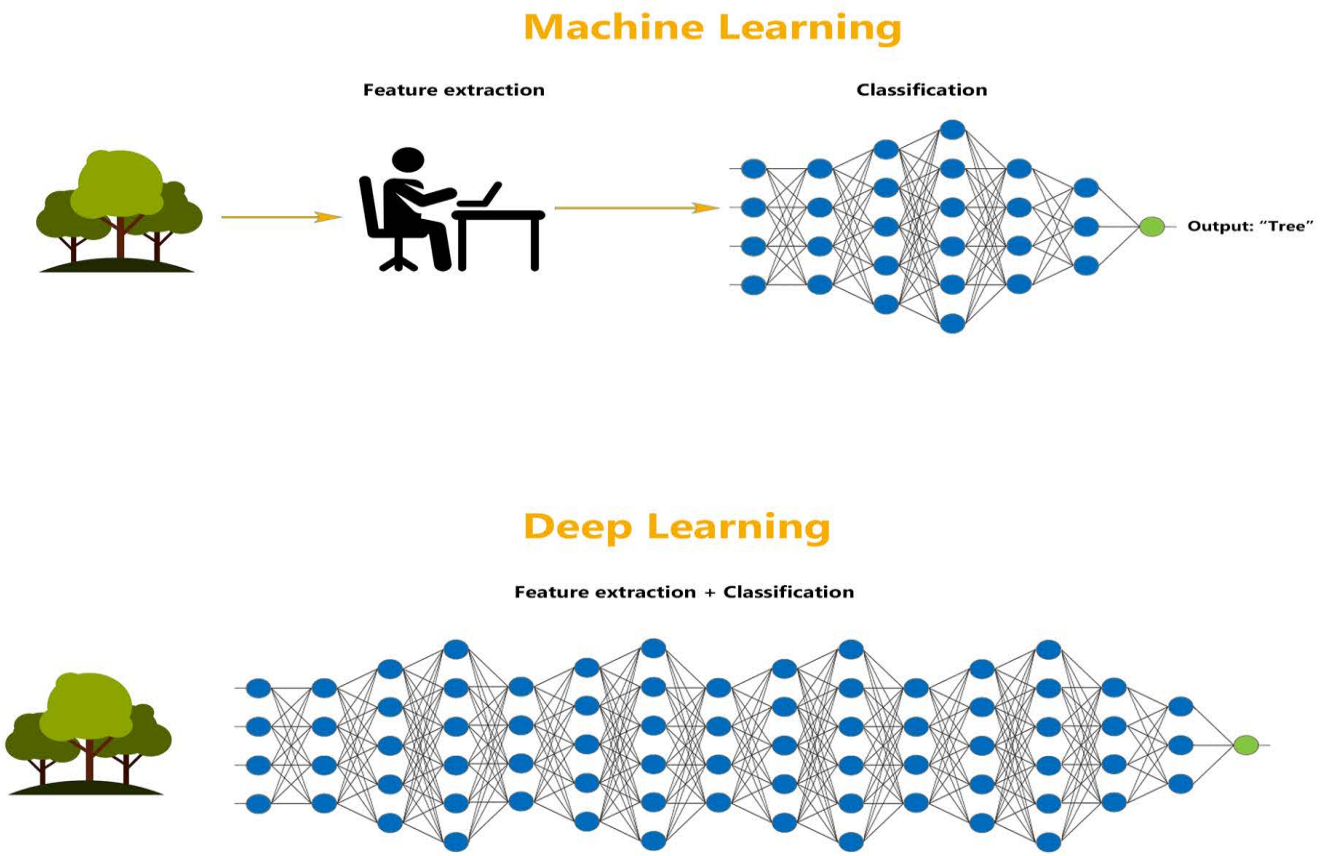
具身智能-03

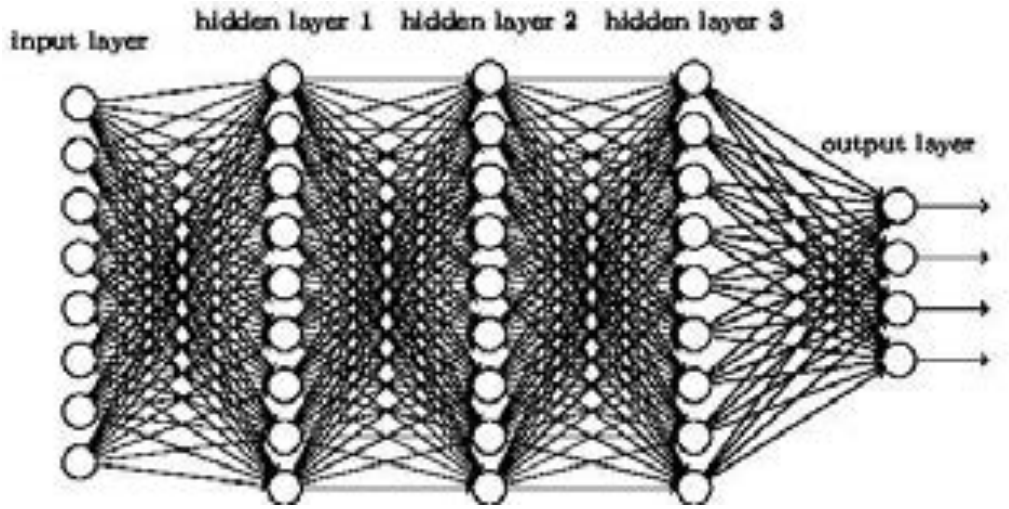
刘华平

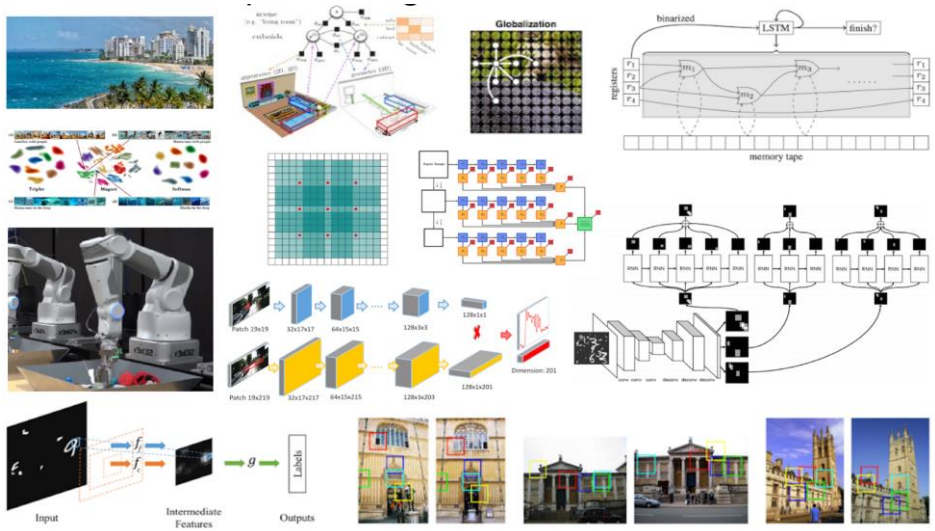
2024年3月5日

背景

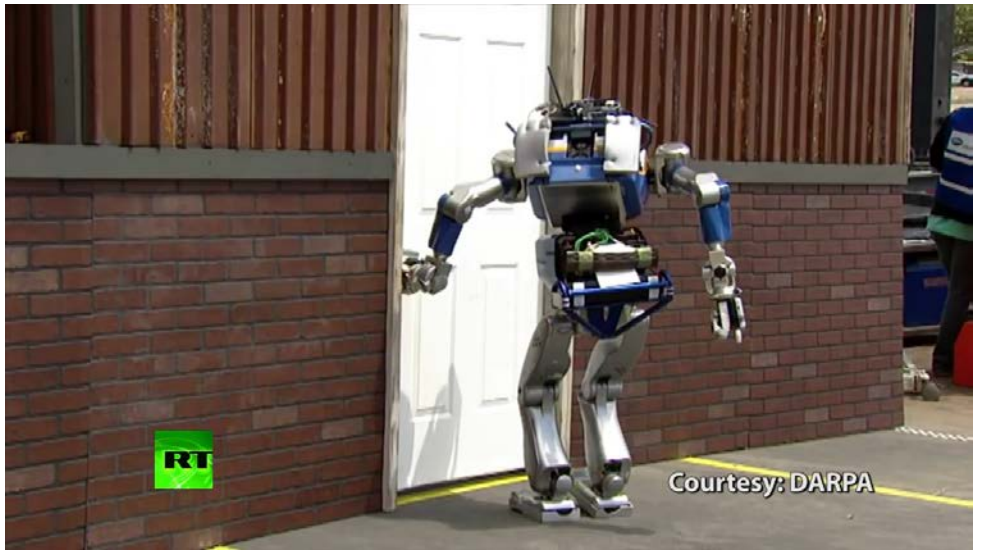


Feature learning, representation learning

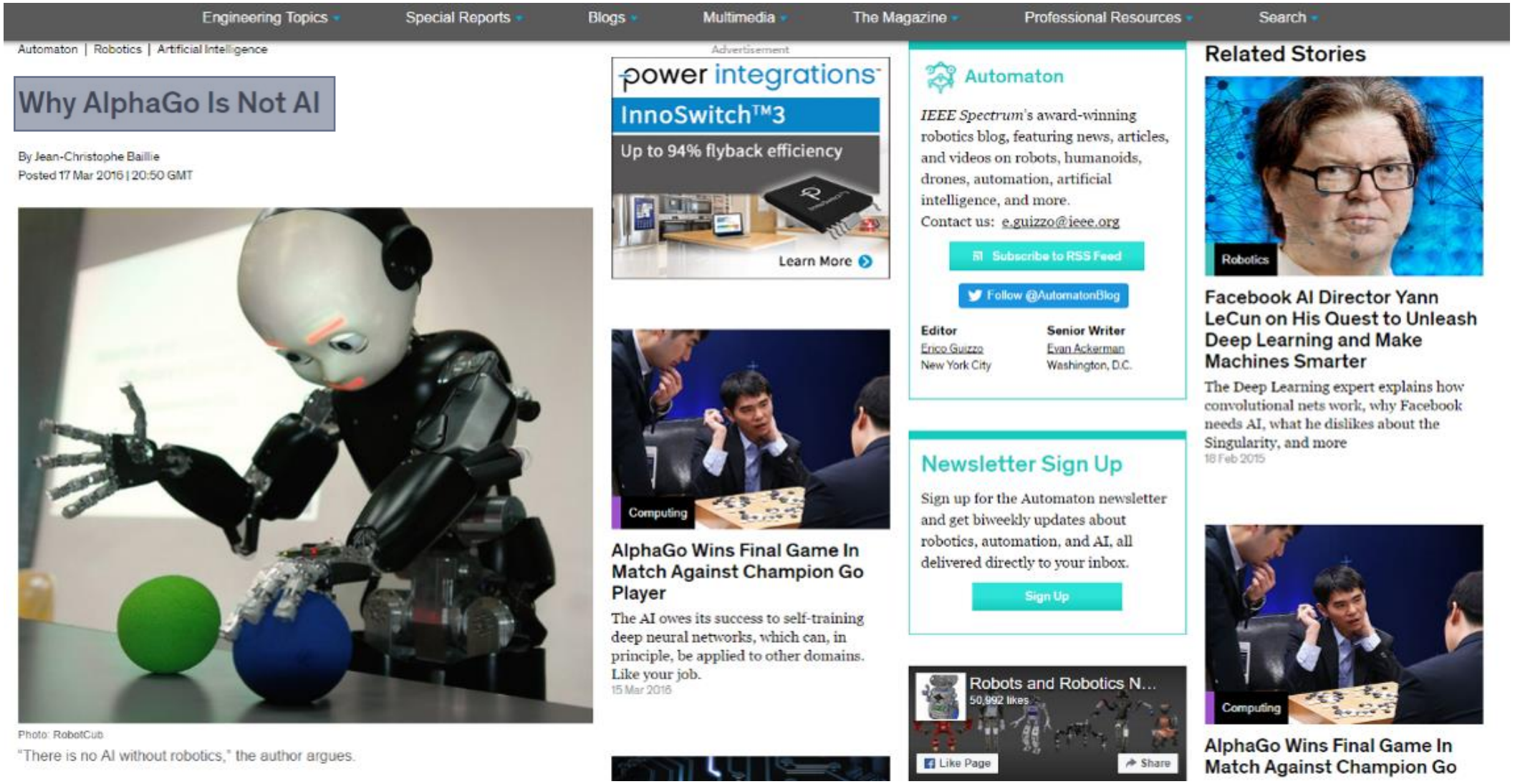

input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer


Globalization
binarized
LSTM
finish?
registers
memory tape
Input
Intermediate Features
Outputs
Labels

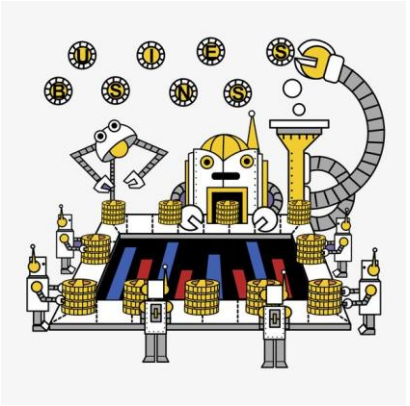

FAIRPLEX
FAIRPLEX
6:16:34 05/06/2015

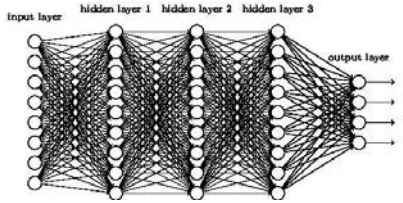

RT
Courtesy: DARPA

已经在“人类最后智力骄傲”上碾压人类的AlphaGo ...



却连挪动一枚小小的棋子都需要人类帮助才能完成。





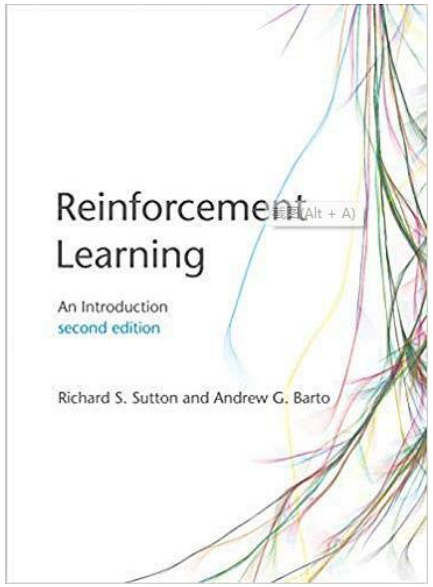
Labels **Images** **Texts**

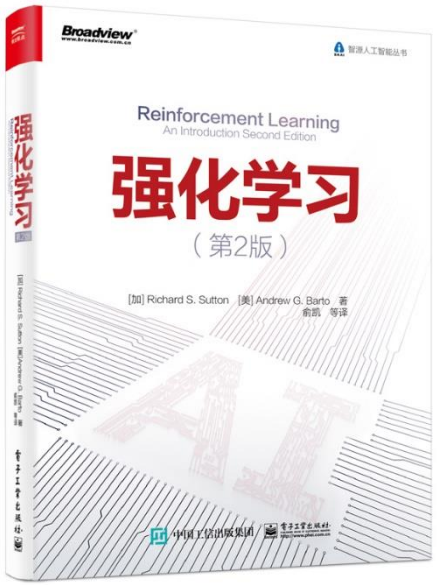
Actions

Actions

Actions

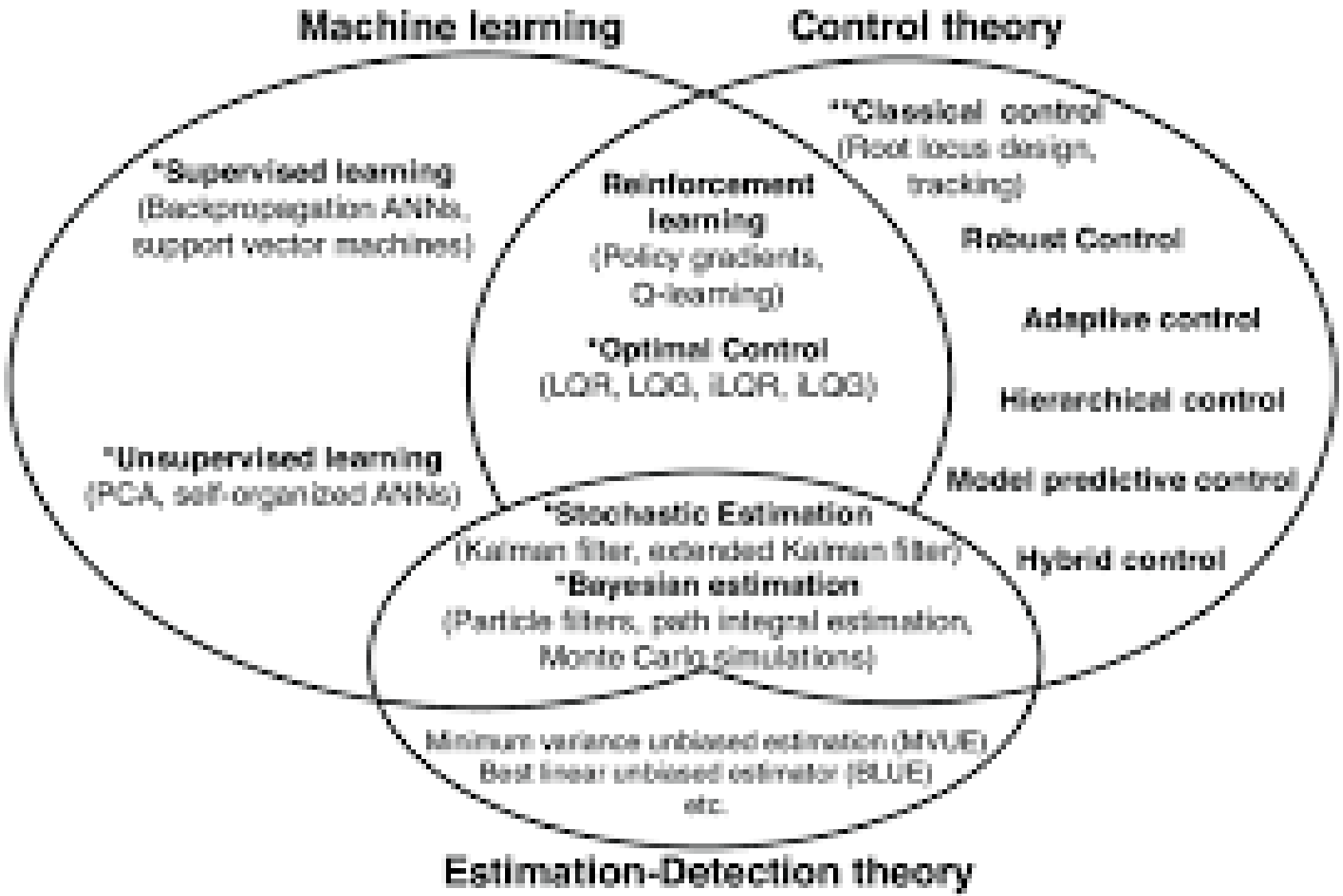
Reinforcement learning solves a particular kind of problem where decision making is sequential, and the goal is long-term, such as game playing, robotics, resource management, or logistics.





- 传统人工智能领域的三大学派：以逻辑推断和贝叶斯学习为代表的符号主义学派、以神经网络为代表的联结主义学派以及以控制论和强化学习为代表的行为主义学派，在不同的历史阶段都产生了很多绝妙的思想和理论成果，而技术应用的水平和范围也让它们受到的关注度起起落落。
- 20世纪 40年代到 50年代，行为主义的控制论因其在航空、航天、机械、化工等领域的巨大成功受到了极大重视，也独立产生了自动控制等技术学科，甚至连早期的计算机专业也都是从控制相关的专业中分出来的，但其应用往往不被认为是一种“智能”，因而长期独立发展，游离于人工智能研究者的视野之外。
- 直到 21世纪初，深度学习理论被提出，借助 GPU等计算机硬件的算力飞跃并与大数据结合，迅速产生了巨大的产业技术红利，使得联结主义一跃成为当前人工智能研究最炙手可热的学派。而无论技术应用如何风云变幻，产业发展如何潮起潮落，在人工智能的发展历程中，始终有一批思想的先行者以近乎顽固的执着态度在不同时代的“非主流”方向上进行着思考和探索，而正是这些执着甚至孤独的思想者，在技术应用热潮冷却后的暗夜里保留了火种，照亮了人类不停息的探索之路。
- 如火如荼的产业应用掩盖不住冷静的研究者们对人工智能未来走向的担忧，越来越多的研究者把深度学习的改良性研究视为工业界的应用技巧，而开始关注与联结主义的经典深度学习不同的人工智能范式探索。
- 将联结主义与行为主义融合起来，将基于静态数据和标签的、数据产生与模型优化相互独立的“开环学习”，转变为与环境动态交互的、在线试错的、数据（监督信号）产生与模型优化紧密耦合在一起的“闭环学习”。强化学习就是“闭环学习”范式的典型代表。

Machine Learning Control Theory



- **背景：自动控制**
- 强化学习：Q学习
- 强化学习：策略梯度

背景：自动控制

控制是一门“隐藏得很深”的技术

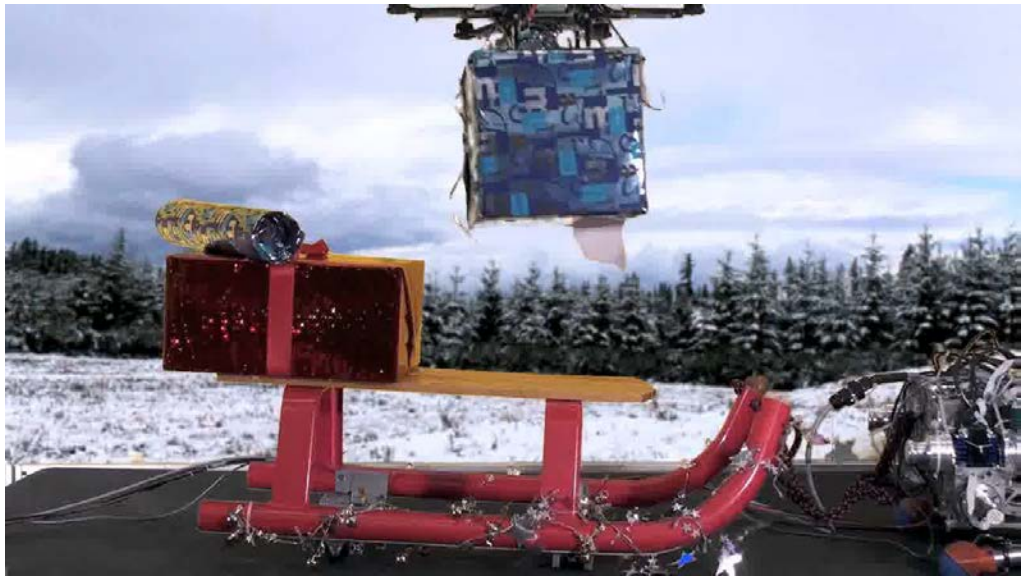
大多数情况下，只有发生了故障，我们才意识到它们其实是自动运行的


凤凰视频
v.ifeng.com
團結緊張嚴肅
國之大器

背景：自动控制

背景：自动控制

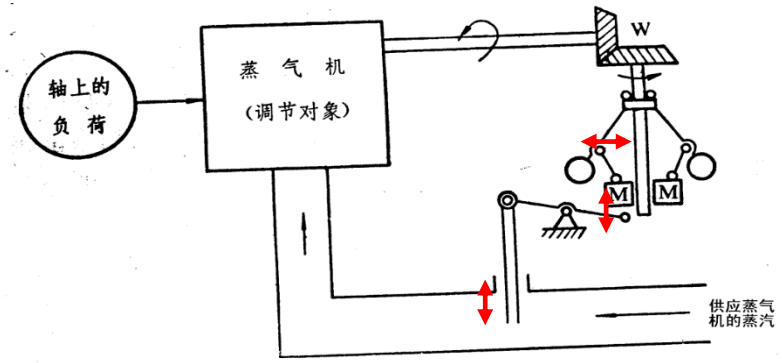
背景：自动控制

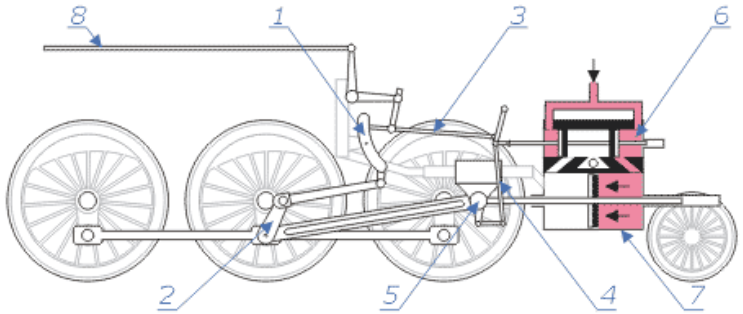


背景：自动控制

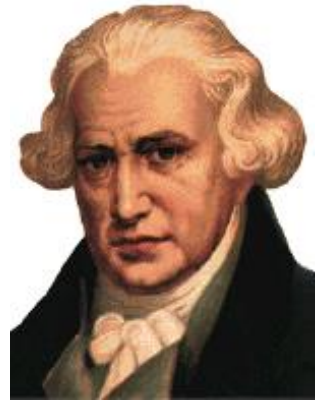
➢离心调速器

1784年，瓦特发明了离心调速器

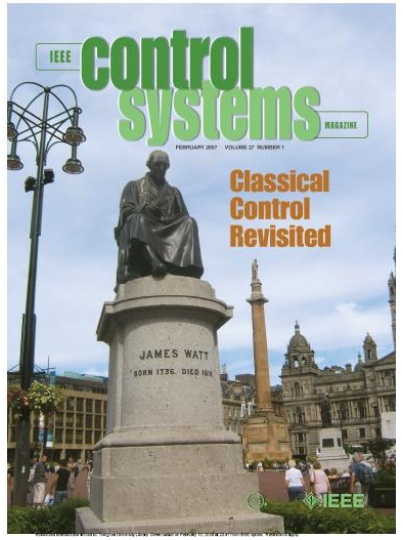





The flyball governor of James Watt.



James Watt
(1736～1819)



背景：自动控制

➢ 自动控制的发展

Watt

Maxwell

On Governors

论调节器

Wiener

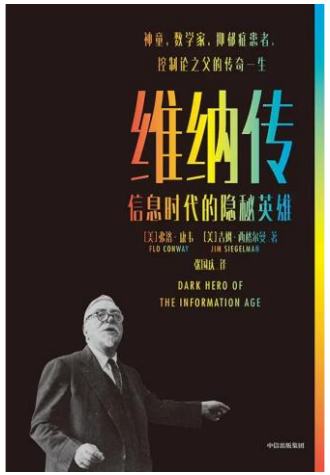
Cybernetics

控制论

1784 ---------- **84** ---------- **1868** ---------- **80** ---------- **1948**

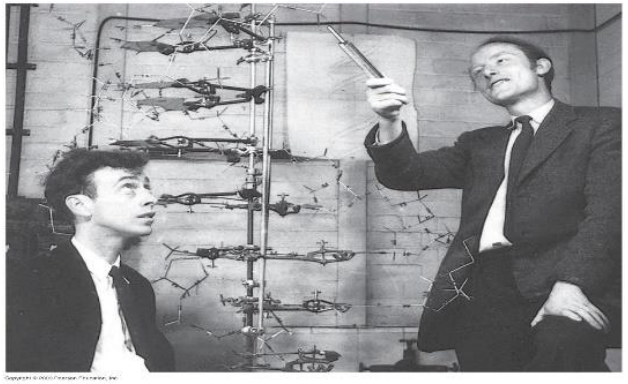
背景：自动控制

➢自动控制的发展

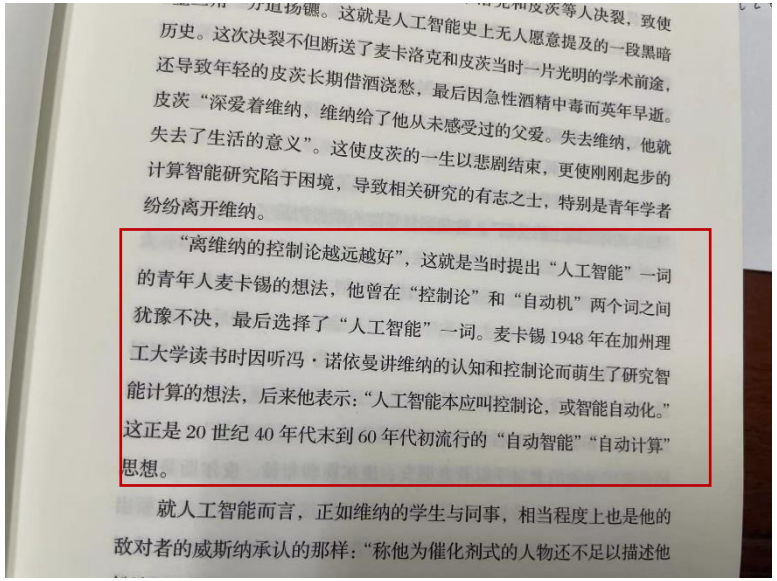


马克斯·玻恩（1954）

海森堡
（1932）



沃森，克里克
（1962）

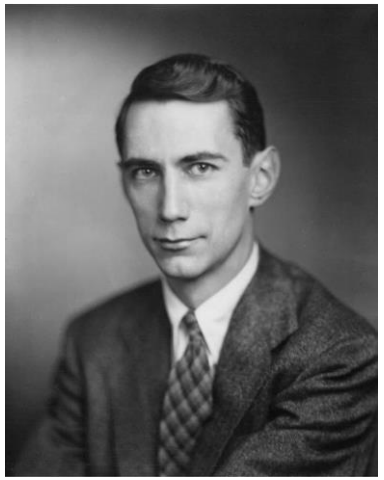


历史。这次决裂不但断送了麦卡洛克和皮茨当时一片光明的学术前途，还导致年轻的皮茨长期借酒浇愁，最后因急性酒精中毒而英年早逝。皮茨“深爱着维纳，维纳给了他从未感受过的父爱。失去维纳，他就失去了生活的意义”。这使皮茨的一生以悲剧结束，更使刚刚起步的计算智能研究陷于困境，导致相关研究的有志之士，特别是青年学者纷纷离开维纳。

“离维纳的控制论越远越好”，这就是当时提出“人工智能”一词的青年人麦卡锡的想法，他曾在“控制论”和“自动机”两个词之间犹豫不决，最后选择了“人工智能”一词。麦卡锡 1948 年在加州理工大学读书时因听冯·诺依曼讲维纳的认知和控制论而萌生了研究智能计算的想法，后来他表示：“人工智能本应叫控制论，或智能自动化。”这正是 20 世纪 40 年代末到 60 年代初流行的“自动智能”“自动计算”思想。

就人工智能而言，正如维纳的学生与同事，相当程度上也是他的敌对者的威斯纳承认的那样：“称他为催化剂式的人物还不足以描述他

冯诺依曼

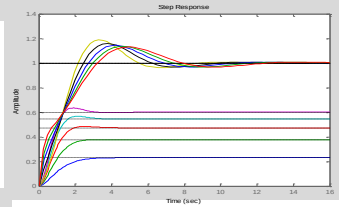


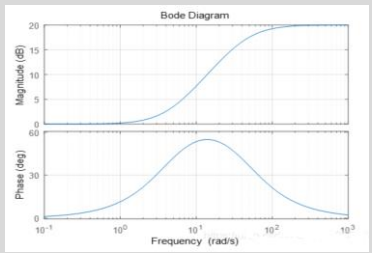
香农

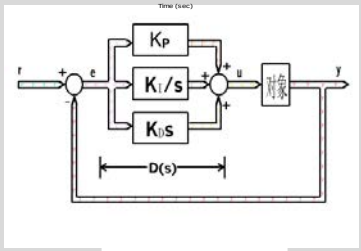
背景：自动控制

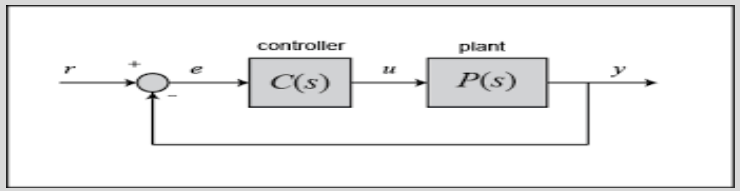
经典控制：经验

$$G_c(s) = \frac{s + \frac{1}{T}}{s + \frac{1}{\alpha T}} \quad (\alpha < 1)$$

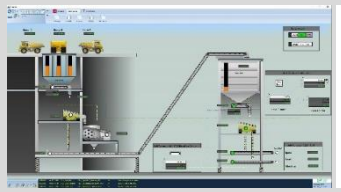


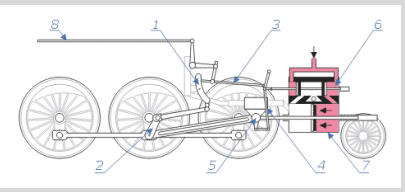






$$u_t = K_P e_t + K_I \int e_t dt + K_D \frac{de_t}{dt}$$

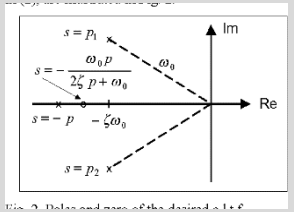




现代控制：模型

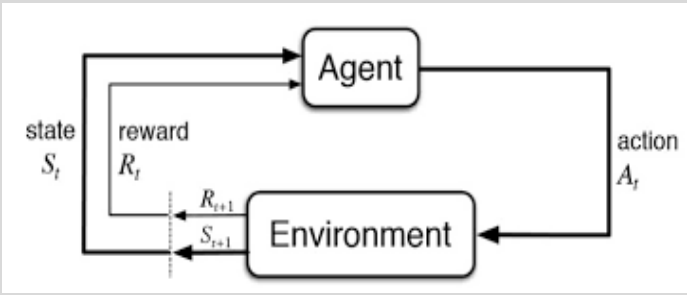
$$\begin{cases} \text{state equations} \\ \text{output equations} \end{cases} \begin{cases} \dot{x}_1 = f_1(x_1, x_2, \ldots, x_n, u_1, \ldots, u_m) \\ \vdots \\ \dot{x}_n = f_n(x_1, x_2, \ldots, x_n, u_1, \ldots, u_m) \\ y_1 = h_1(x_1, x_2, \ldots, x_n, u_1, \ldots, u_m) \\ \vdots \\ y_p = h_p(x_1, x_2, \ldots, x_n, u_1, \ldots, u_m) \end{cases}$$

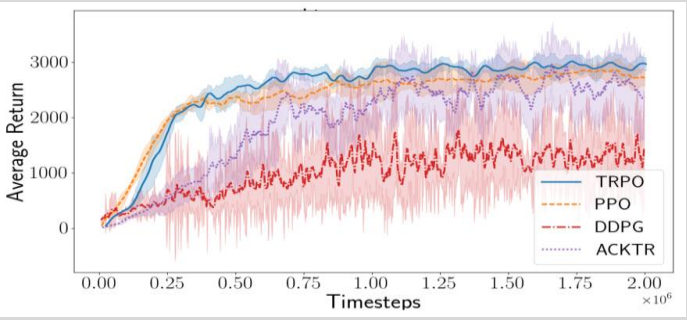
$$\begin{cases} \dot{x}_t = Ax_t + Bu_t \\ y_t = Cx_t + Du_t \end{cases}$$



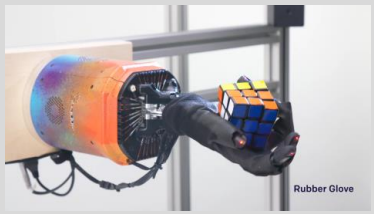
$$u_t = -Kx_t$$

智能控制：学习



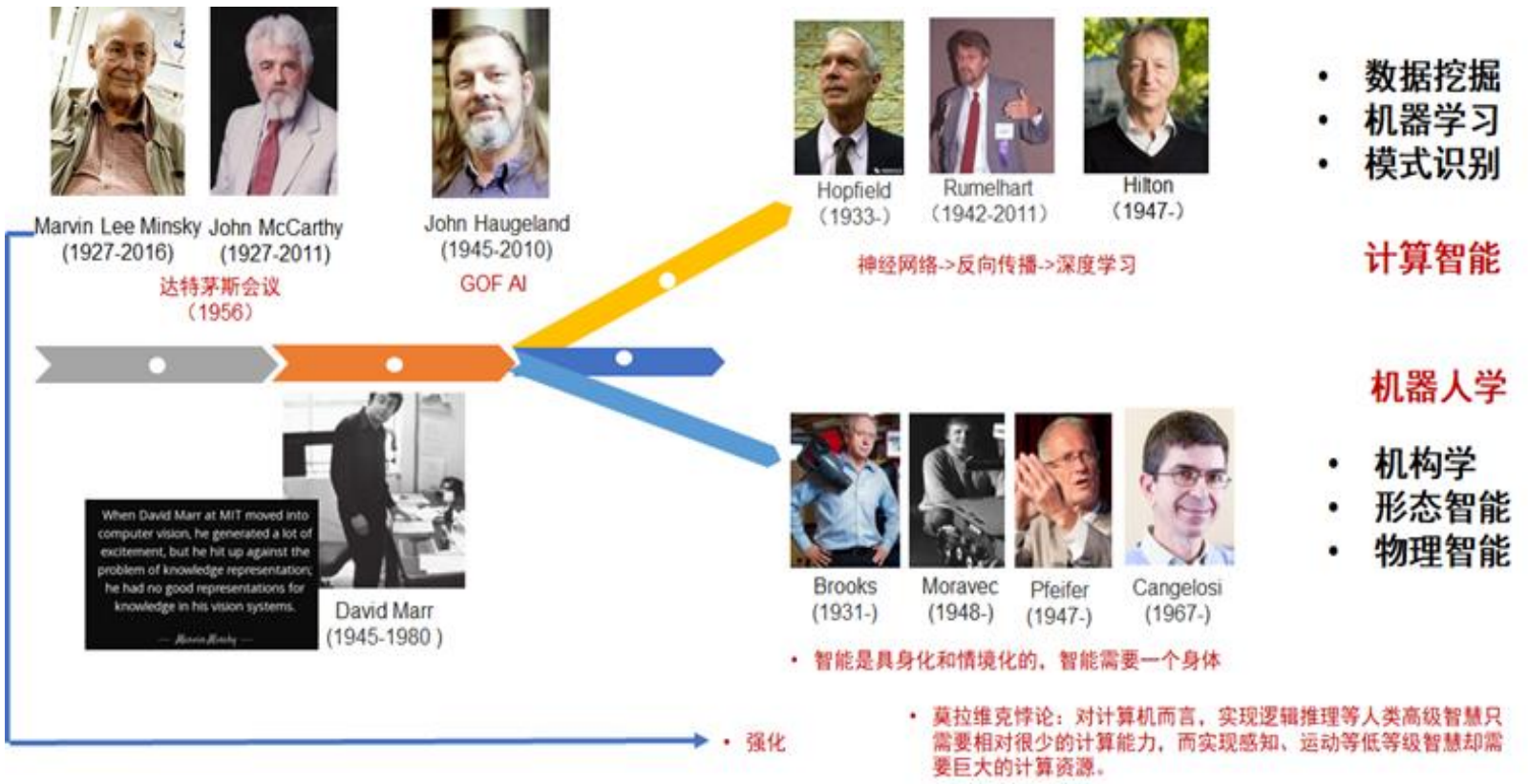


$$\pi_\theta(a_t \mid s_t)$$



背景：自动控制

➢ 起源



人工神经网络
Artificial Neural Networks

Walter Pitts, 1943

深度学习
Deep Learning

Geoffrey Hinton, 2006

最优控制与动态规划
Markov Decision Process

Richard Bellman, 1957

时序差分与最优控制
Q-learning

Chris Watkins, 1989

深度强化学习
DQN

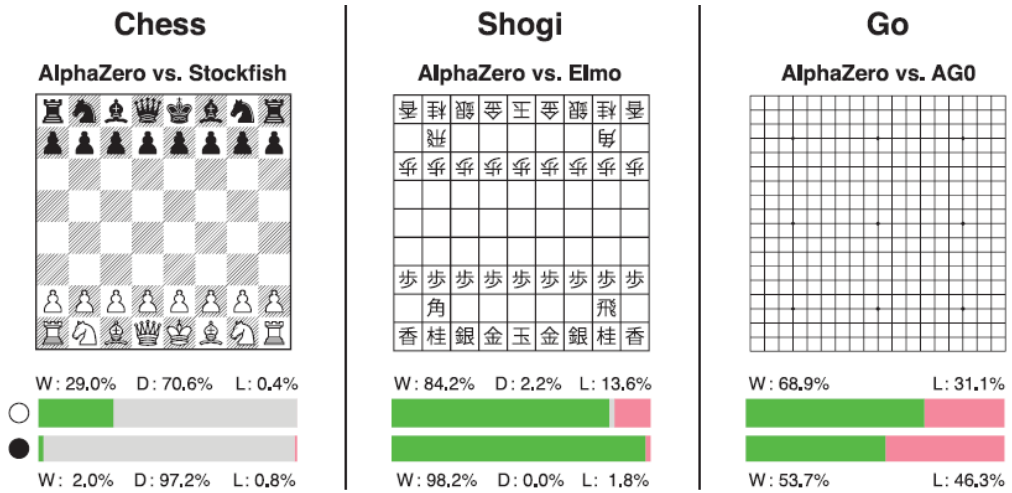
DeepMind, 2015

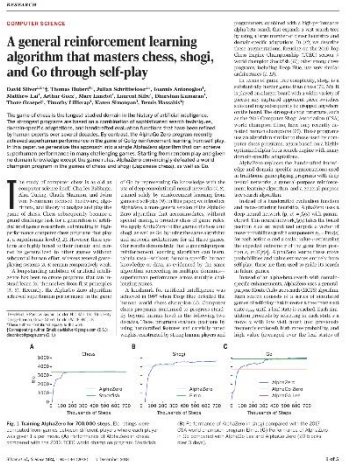
背景：自动控制

➢复苏


LETTER

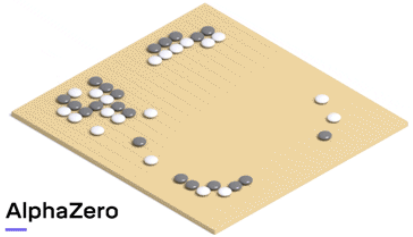
Human-level control through deep reinforcement learning

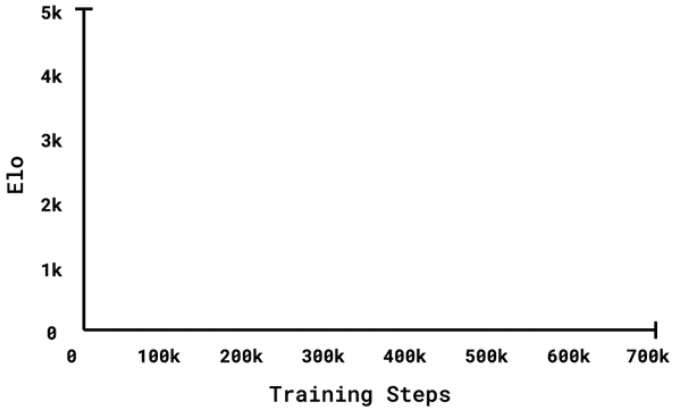



RESEARCH

COMPUTER SCIENCE

A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play

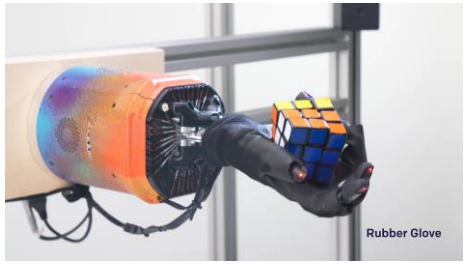


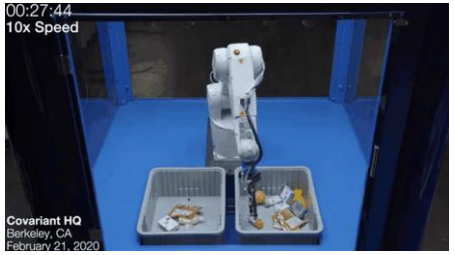


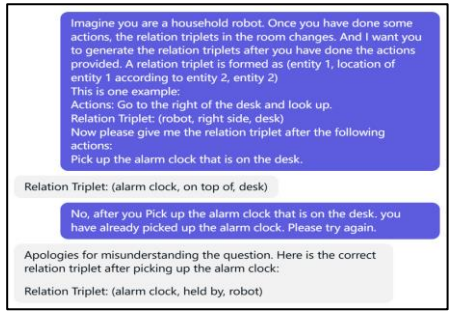
背景：自动控制

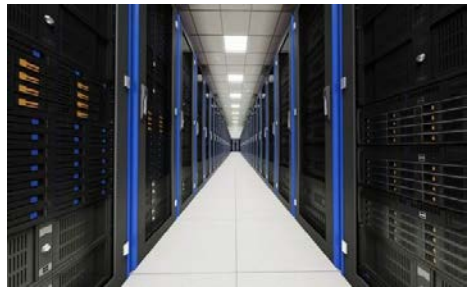
➢应用

- 控制：机器人、工程系统
- 交互：推荐、搜索
- 调度：网络流量、电力优化
- 博弈：游戏、棋艺









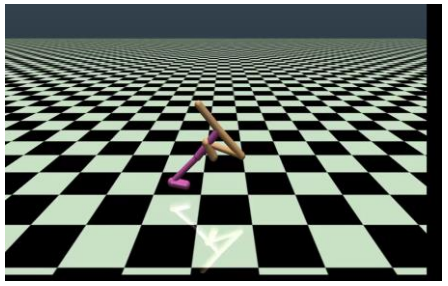
背景：自动控制

- 应用：机器人

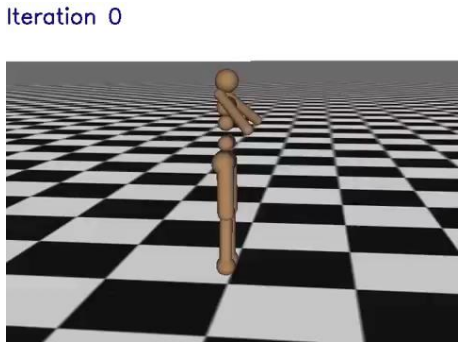
Tedrake et al, 2005

Kober and Peters, 2009

Mnih et al, 2015 (A3C)



Lillicrap et al, 2015 (DDPG)



Schulman et al, 2016
(TRPO + GAE)

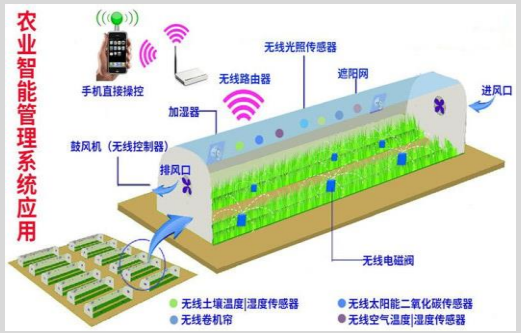
Levine*, Finn*, et al, 2016
(GPS)

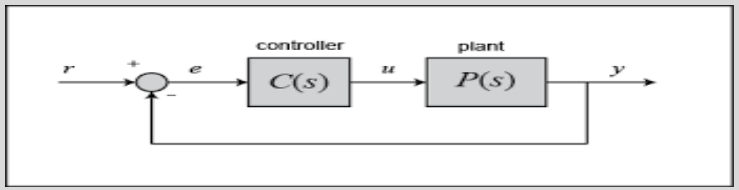
- 背景
- 强化学习：Q学习
- 强化学习：策略梯度

强化学习

经典控制：经验

黑箱建模



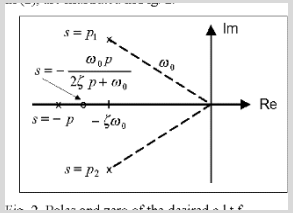


$$u_t = K_P e_t + K_I \int e_t dt + K_D \frac{de_t}{dt}$$

现代控制：模型

白箱建模

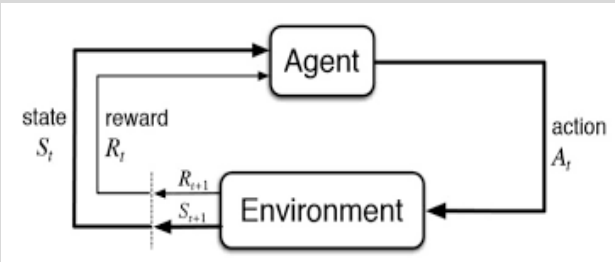
$$\begin{cases} \dot{x}_t = Ax_t + Bu_t \\ y_t = Cx_t + Du_t \end{cases}$$



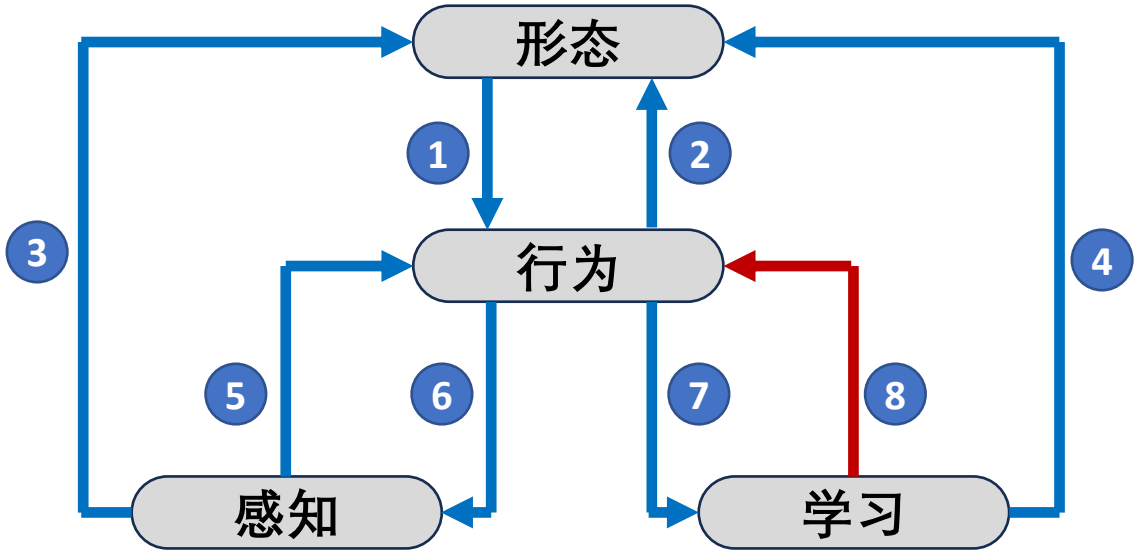
$$u_t = -Kx_t$$

智能控制：学习

交互建模

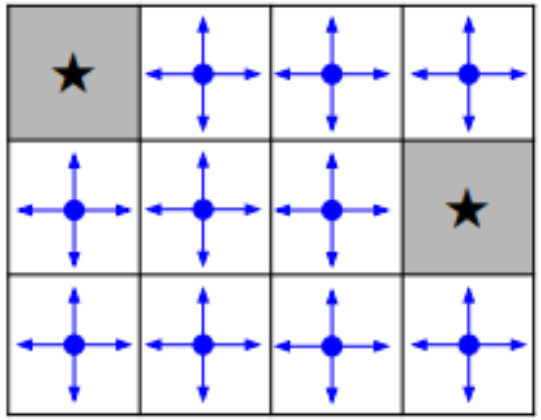


$$\pi_\theta(a_t \mid s_t)$$

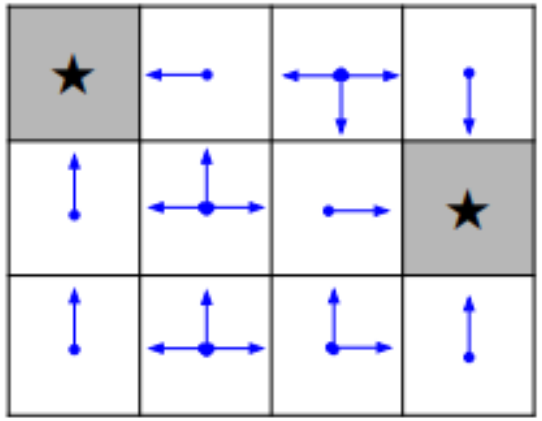


① 基于形态的行为生成　③ 基于感知的形态变换　⑤ 基于感知的行为生成　⑦ 基于行为的自主学习

② 基于行为的形态控制　④ 基于学习的形态优化　⑥ 基于行为的主动感知　⑧ 基于学习的行为优化

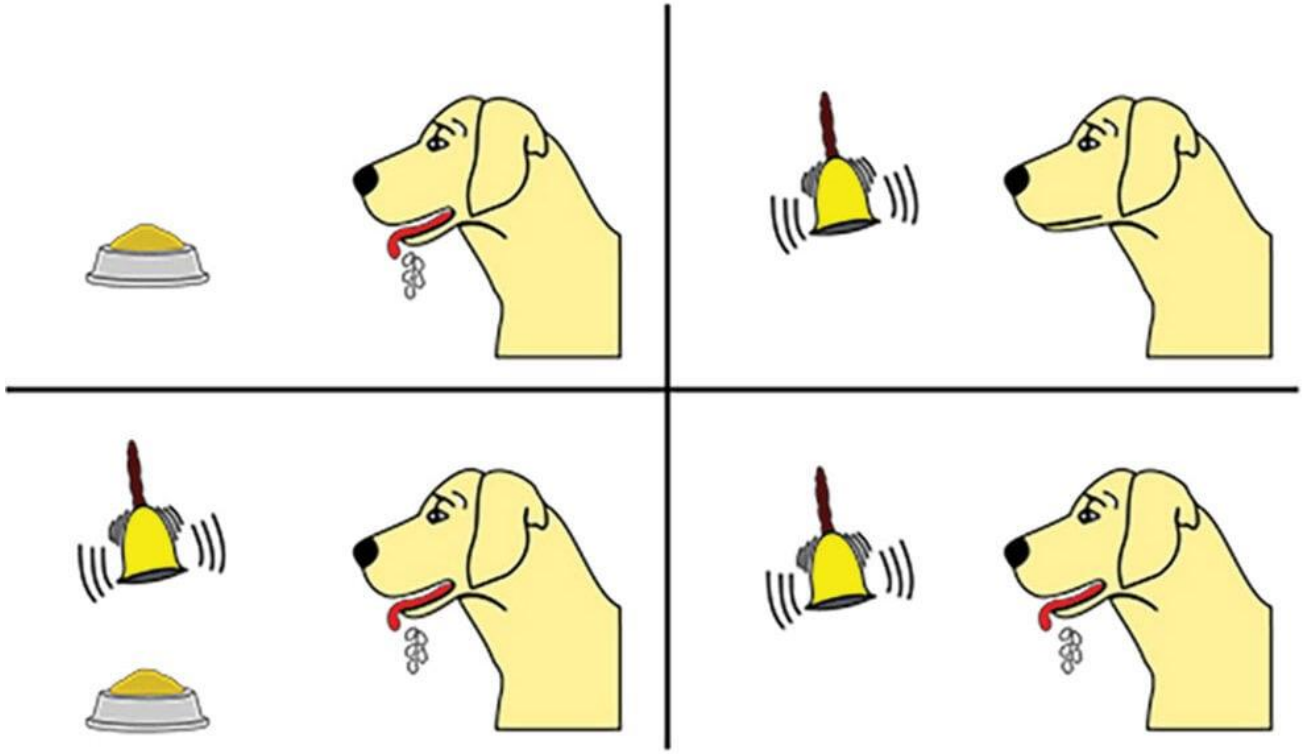


Random Policy

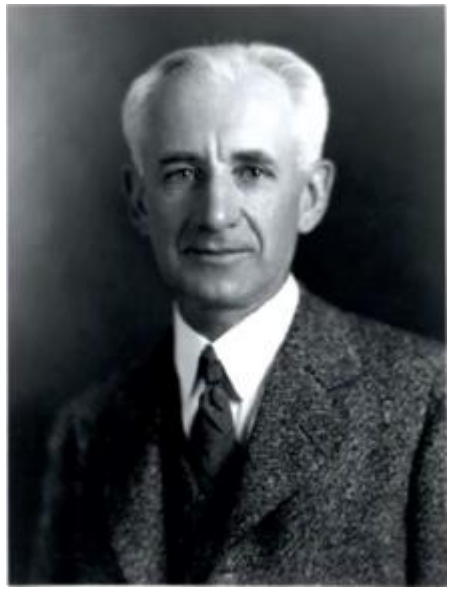


Optimal Policy

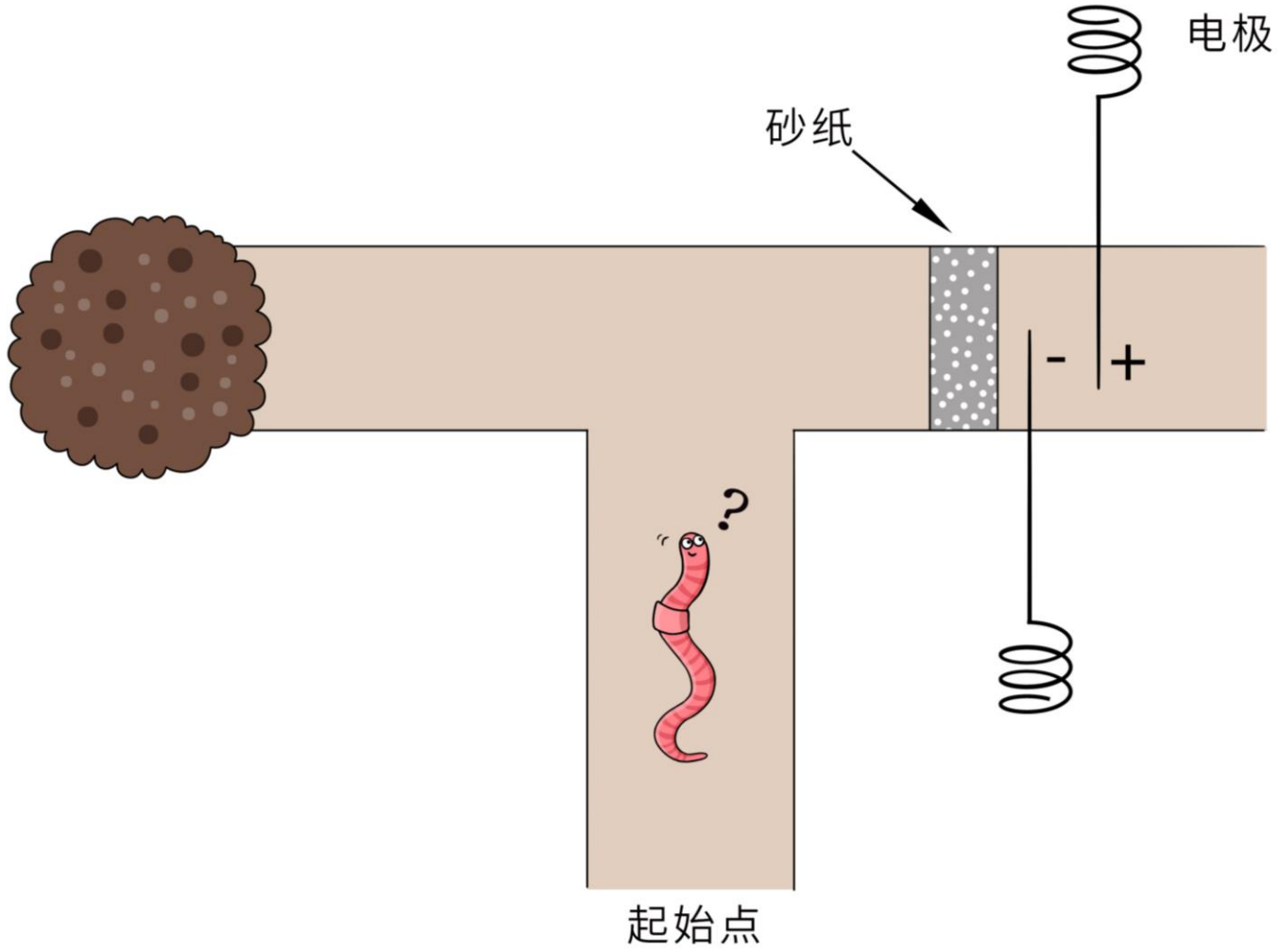
强化学习



强化学习



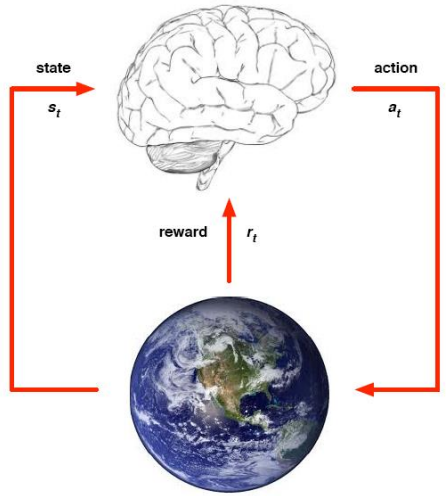
Robert Mearns Yerkes
（1876 -1956）



强化学习

➢强化学习的基本框架

Select actions to maximise future total reward

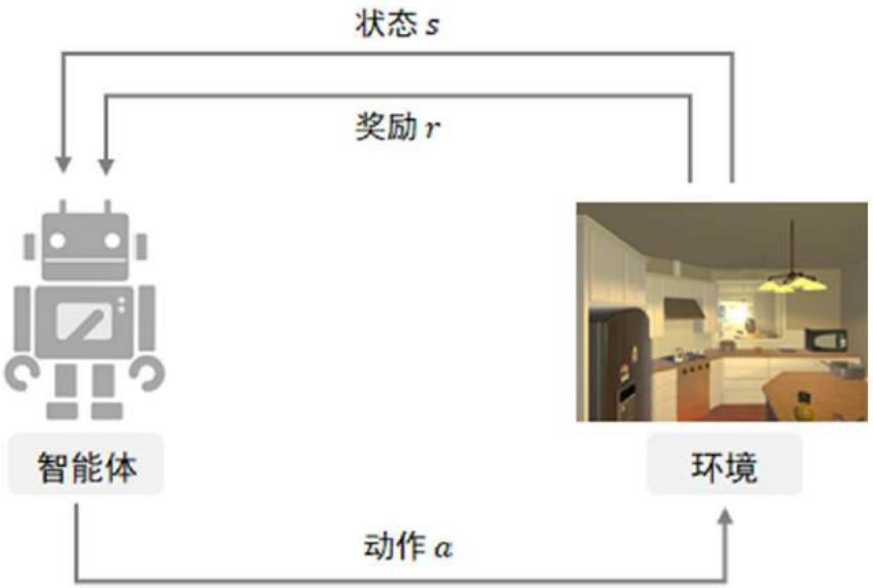


- At each step t the agent:
  - Receives state s_t
  - Receives scalar reward r_t
  - Executes action a_t

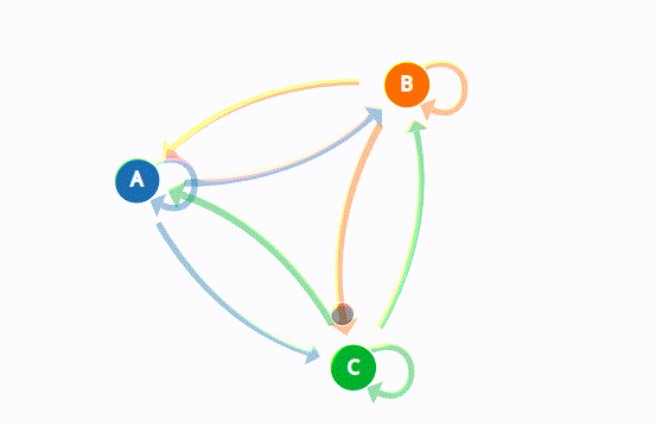
$$a = \pi_\theta(s)$$

强化学习

➢强化学习的基本框架



- $\mathcal{S}$ 为关于智能体状态的集合
- $\mathcal{A}$ 为关于智能体动作的集合
- T 为状态转移函数，定义为：$T(s'|s,a)=\text{Prob}\left[s_{t+1}=s'|s_t=s,a_t=a\right]$，其中 $s_t\in\mathcal{S}$ 与 $a_t\in\mathcal{A}$ 分别代表 t 时刻的状态与动作取值，而 $s_{t+1}\in\mathcal{S}$ 则代表 t+1 时刻的状态取值。这个概率转化关系是 Markov 决策过程的具体体现。即只要 t 时刻的状态 s_t 与动作 a_t 可知，则不再需要所有的历史信息就可以确定 t+1 时刻的状态取值 s_{t+1} 取值。通俗地说就是：t+1 时刻的状态取值 s_{t+1} 仅与 t 时刻的状态 s_t 与动作 a_t 有关，而与之前的信息无关。此外，即使状态 s_t 与动作 a_t 已经确定了，s_{t+1} 也是可以在状态集合 $\mathcal{S}$ 中随机取值的。状态转移函数显然满足：$\sum_{s'\in\mathcal{S}}T(s'|s,a)=1$。
- r 为一标量，称为立即奖励，用于表示在状态 s 下，采取动作 a，并转移到状态 s' 的情况下，环境给智能体反馈的奖励。这一奖励反映了智能体与环境的交互，其核心是一个函数映射：$r:\mathcal{S}\times\mathcal{A}\times\mathcal{S}\to\mathbb{R}$。也可具体展开记为 $r(s,a,s')$。注意我们也可以进一步定义 $r(s,a)=\mathbb{E}_{s'\sim T(s'|s,a)}\left(r(s,a,s')\right)$，用于刻画转移到不同状态的平均奖励。
- $\gamma\in[0,1)$ 为一标量，称为遗忘因子，用于刻画奖励函数的累计效应。具体作用将在后面介绍。



$$\langle S,A,T,r,\gamma\rangle$$

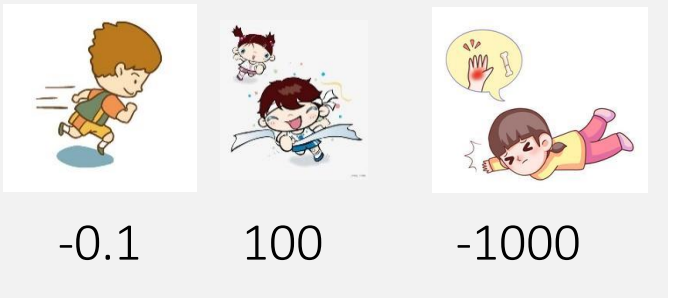
基于上面关于 MDP 的基本描述，我们就可以考虑基本的强化学习问题了。具体来说，就是智能体应该学习一个策略映射 $\pi:\mathcal{S}\to\mathcal{A}$，对任意时刻 t 的状态 $s_t\in\mathcal{S}$，确定动作 $a_t=\pi(s_t)$，从而可以使得智能体达到下一时刻的状态 $s_{t+1}\sim T(s'|s_t,a_t)$，并在与环境交互过程中获得奖励 $r(s_t,a_t,s_{t+1})$。这一过程持续演进，最终可以得到一条轨迹：

$$\tau=\{s_0,a_0;s_1,a_1;s_2,a_2;.....\}$$

强化学习

➢目标

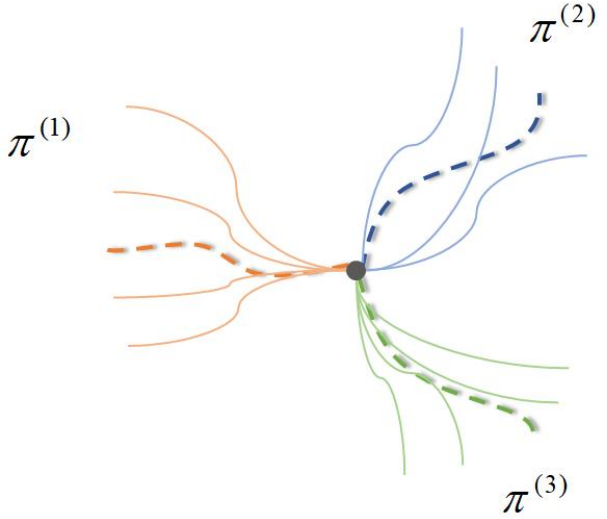
$$\max_{\pi} \quad \mathbb{E}_{\tau \sim p_{\pi}(\tau)} \left\{ \sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t, s_{t+1}) \right\}$$



$$\langle S, A, T, r, \gamma \rangle$$

基于上面关于 MDP 的基本描述，我们就可以考虑基本的强化学习问题了。具体来说，就是智能体应该学习一个策略映射 $\pi : \mathcal{S} \to \mathcal{A}$，对任意时刻 t 的状态 $s_t \in \mathcal{S}$，确定动作 $a_t = \pi(s_t)$，从而可以使得智能体达到下一时刻的状态 $s_{t+1} \sim T(s' | s_t, a_t)$，并在与环境交互过程中获得奖励 $r(s_t, a_t, s_{t+1})$。这一过程持续演进，最终可以得到一条轨迹：

$$\tau = \{s_0, a_0; s_1, a_1; s_2, a_2; \ldots\ldots\}$$



强化学习

➢强化学习与监督学习的区别

$$y = f_\theta(x)$$

$$a = \pi_\theta(s)$$

监督学习
目标分类：窗户、门框？

强化学习
导航策略：向左、向右？

强化学习

➢奖励 $r(s_t, a_t, s_{t+1})$

$$\max_{\pi} \ \mathbb{E}_{\tau \sim p_\pi(\tau)} \left\{ \sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t, s_{t+1}) \right\}$$

0

(-0.1)

100

-1000

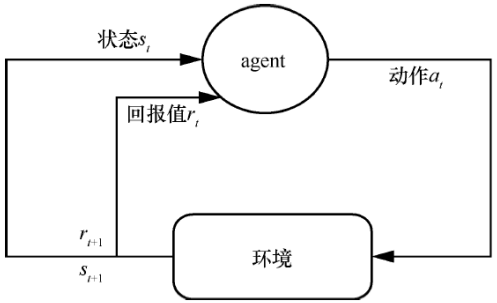
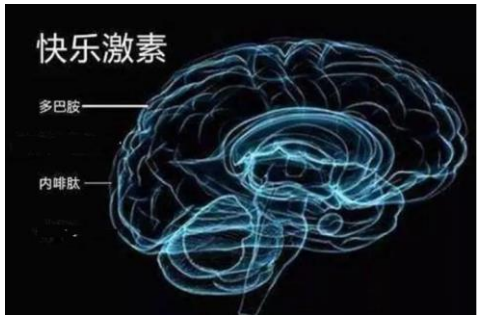


图 1　agent 与环境交互

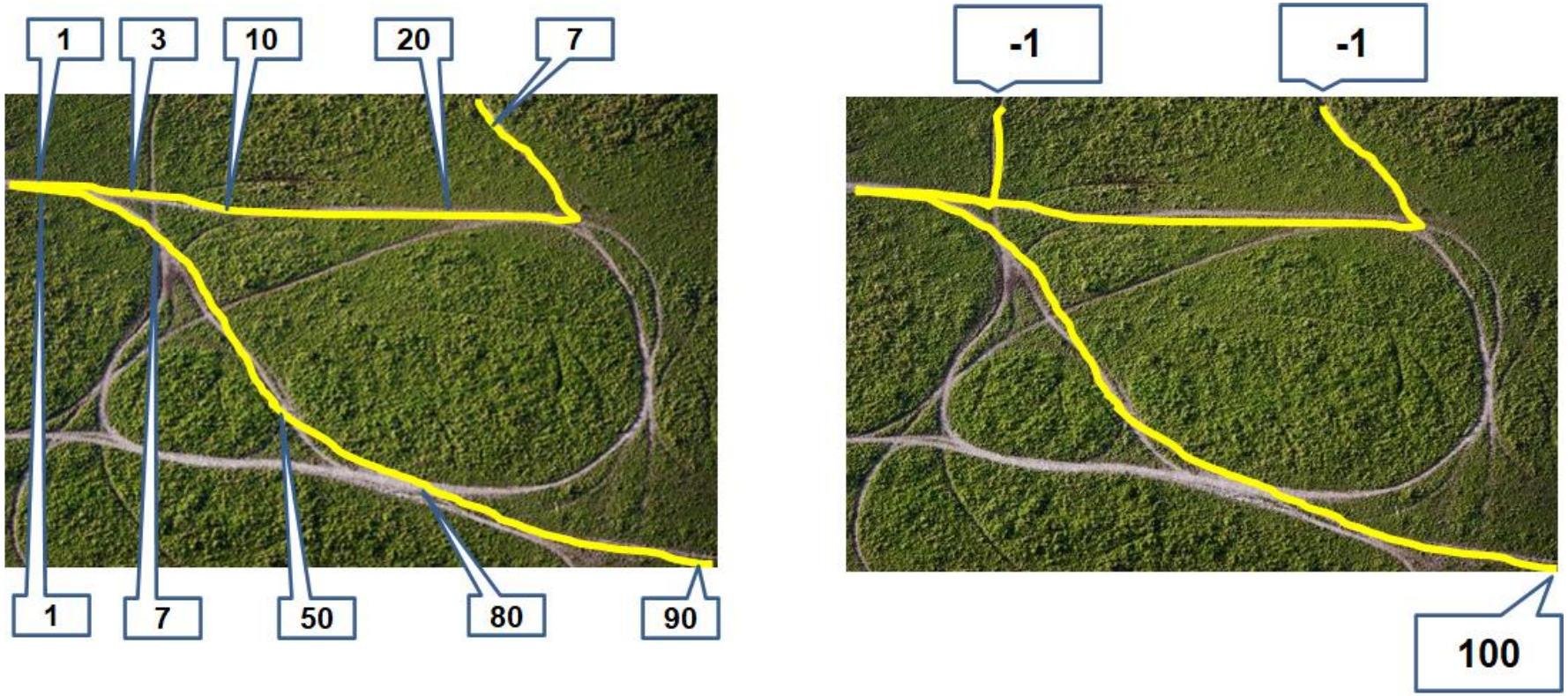
$$\langle S, A, T, r, \gamma \rangle$$



强化学习

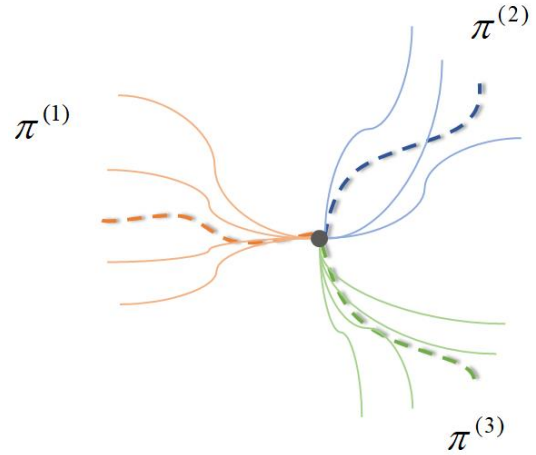
- 奖励 $r(s_t, a_t, s_{t+1})$

$$\max_{\pi} \quad \mathbb{E}_{\tau \sim p_\pi(\tau)} \left\{ \sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t, s_{t+1}) \right\}$$

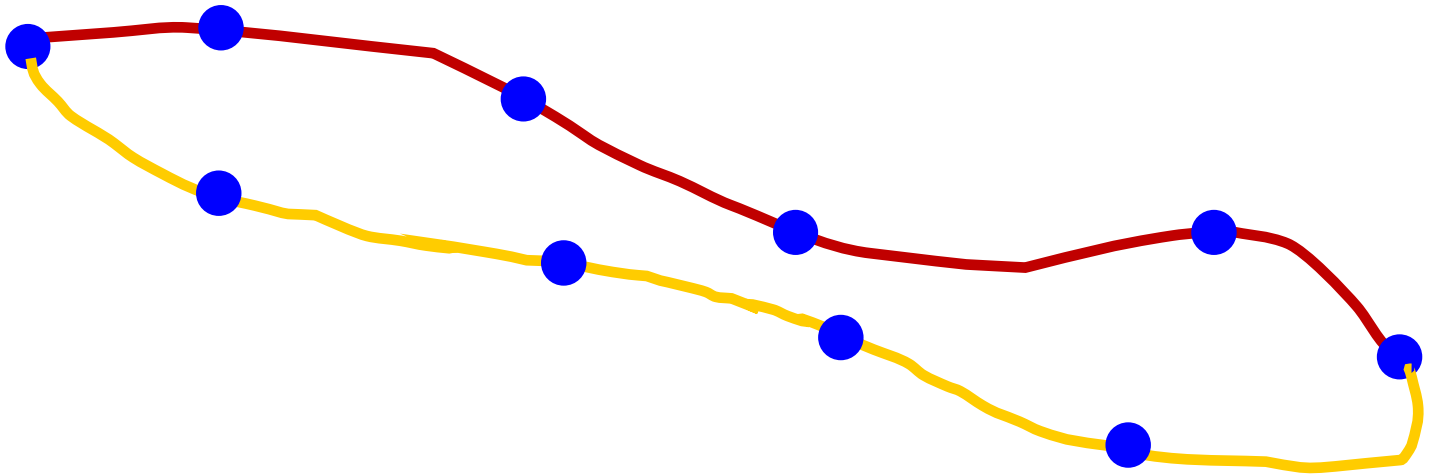


- 值函数的引入

$$\max_{\pi} \quad \mathbb{E}_{\tau \sim p_{\pi}(\tau)} \left\{ \sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t, s_{t+1}) \right\}$$



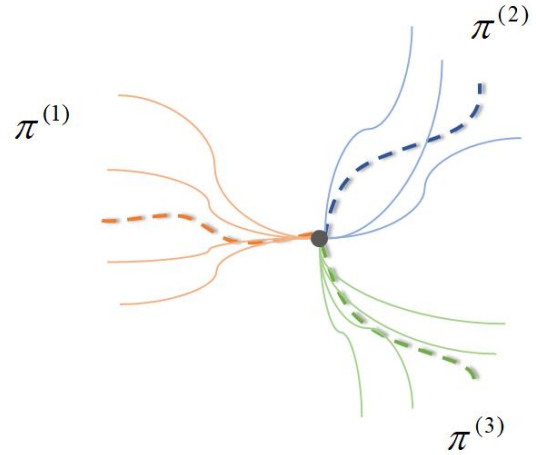
$$V^{\pi}(s) = \mathrm{E}_{\tau \sim p_{\pi}(\tau)} \left\{ \sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t, s_{t+1}) \,|\, s_0 = s \right\}$$



强化学习

- 值函数的引入

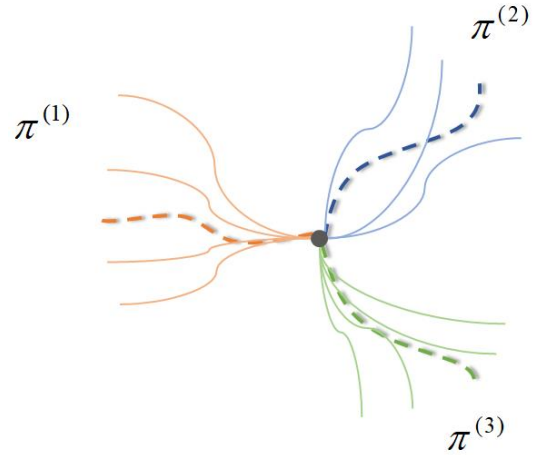
$$\max_{\pi} \quad \mathbb{E}_{\tau \sim p_\pi(\tau)} \left\{ \sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t, s_{t+1}) \right\}$$



$$V^\pi(s) = \mathrm{E}_{\tau \sim p_\pi(\tau)} \left\{ \sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t, s_{t+1}) \mid s_0 = s \right\}$$

➢ 值函数的解释

$$\max_{\pi} \quad \mathbb{E}_{\tau \sim p_\pi(\tau)} \left\{ \sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t, s_{t+1}) \right\}$$



$$V^\pi(s) = \mathrm{E}_{\tau \sim p_\pi(\tau)} \left\{ \sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t, s_{t+1}) \mid s_0 = s \right\}$$

一副好牌 $V^\pi(s_1) > V^\pi(s_2)$

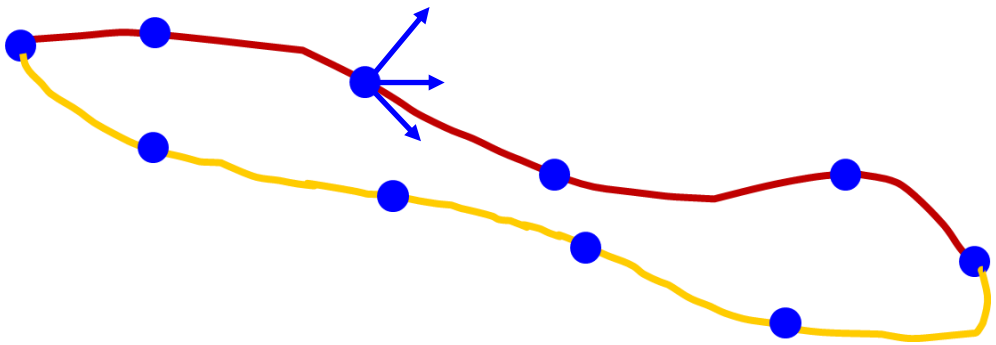
打得稀烂 $V^{\pi_1}(s_1) < V^{\pi_2}(s_2)$

➢ 最佳策略

$$\max_{\pi} \quad \mathbb{E}_{\tau \sim p_{\pi}(\tau)} \left\{ \sum_{t=0}^{+\infty} \gamma^{t} r(s_t, a_t, s_{t+1}) \right\}$$

$$V^{\pi}(s) = \mathbb{E}_{s' \sim T(s'|s,a)} \left(r(s,a,s') + \gamma \cdot V^{\pi}(s') \right)$$

$$\pi^{*}(s) = \arg\max_{a \in \mathcal{A}} \quad \mathbb{E}_{s' \sim T(s'|s,a)} \left(r(s,a,s') + \gamma \cdot V^{*}(s') \right)$$



➢ 贝尔曼方程

$$V^{\pi}(s)=\mathbb{E}_{s'\sim T(s'|s,a)}\left(r(s,a,s')+\gamma\cdot V^{\pi}(s')\right)$$

Richard Bellman
(1920～1984)

$$\max_{\pi}\ \mathbb{E}_{\tau\sim p_{\pi}(\tau)}\left\{\sum_{t=0}^{+\infty}\gamma^{t}r(s_t,a_t,s_{t+1})\right\}$$

$$V^{\pi}(s)=\mathrm{E}_{\tau\sim p_{\pi}(\tau)}\left\{\sum_{t=0}^{+\infty}\gamma^{t}r(s_t,a_t,s_{t+1})\,|\,s_0=s\right\}$$

$$\sum_{t=0}^{+\infty}\gamma^{t}r(s_t,a_t,s_{t+1})|_{s_0=s}=r(s,a,s')+\sum_{t=1}^{+\infty}\gamma^{t}r(s_t,a_t,s_{t+1})|_{s_1=s'}$$

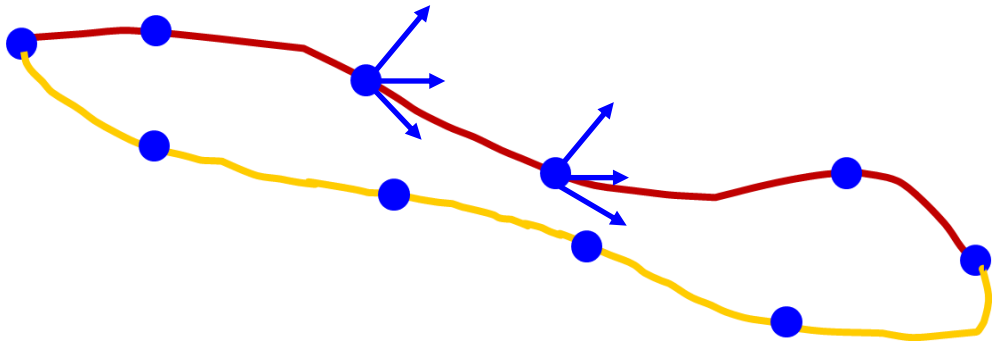
$$=r(s,a,s')+\gamma\sum_{t=1}^{+\infty}\gamma^{t-1}r(s_t,a_t,s_{t+1})|_{s_1=s'}$$

$$V(s)=\mathbb{E}_{s'\sim T(s'|s,a)}\left(r(s,a,s')+\gamma V(s')\right)$$

- 最佳值函数

$$V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim p_\pi(\tau)} \left(\sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t, s_{t+1}) \right)$$

$$V^*(s) = \max_{a \in \mathcal{A}} \mathbb{E}_{s' \sim T(s'|s,a)} \left(r(s,a,s') + \gamma V^*(s') \right)$$



➢ 迭代算法

$$V^*(s)=\max_{a\in\mathcal{A}}\ \mathbb{E}_{s'\sim T(s'|s,a)}\left(r(s,a,s')+\gamma V^*(s')\right)$$

算法 3.1：基于模型的值函数学习方法

目标：最优值函数$V^*(s)$。

Step 1：初始化：令$k=0$，对所有状态$s\in\mathcal{S}$，设$V^{(k)}(s)=0$。

Step 2：$k=k+1$

对所有$s\in\mathcal{S}$与$a\in\mathcal{A}$，计算

$$V^{(k)}(s)=\max_{a\in\mathcal{A}}\left\{\sum_{s'\in S}T(s,a,s')\left(r(s,a,s')+\gamma V^{(k-1)}(s')\right)\right\}$$

Step 3：如果$\sum_{s\in\mathcal{S}}|V^{(k)}(s)-V^{(k-1)}(s)|<\mathrm{TOL}$，令$V^*(s)=V^{(k)}(s)$，退出循环；

否则，返回 Step 2。

$V(s_1)=100$

$V(s_2)=0.1$

➢ 迭代算法

$$V^*(s)=\max_{a\in\mathcal{A}}\ \mathbb{E}_{s'\sim T(s'|s,a)}\left(r(s,a,s')+\gamma V^*(s')\right)$$

算法 3.1：基于模型的值函数学习方法

目标：最优值函数$V^*(s)$。

Step 1：初始化：令$k=0$，对所有状态$s\in\mathcal{S}$，设$V^{(k)}(s)=0$。

Step 2：$k=k+1$

对所有$s\in\mathcal{S}$与$a\in\mathcal{A}$，计算

$$V^{(k)}(s)=\max_{a\in\mathcal{A}}\left\{\sum_{s'\in S}T(s,a,s')\left(r(s,a,s')+\gamma V^{(k-1)}(s')\right)\right\}$$

Step 3：如果$\sum_{s\in\mathcal{S}}|V^{(k)}(s)-V^{(k-1)}(s)|<\mathrm{TOL}$，令$V^*(s)=V^{(k)}(s)$，退出循环；

否则，返回 Step 2。

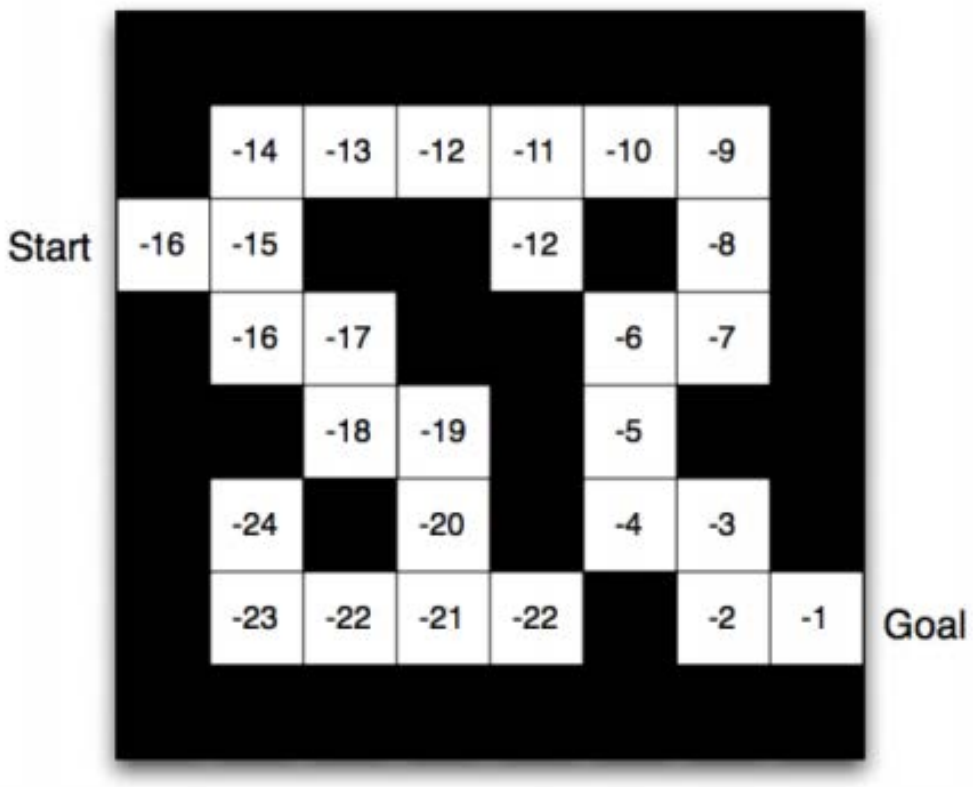
$$\pi^*(s)=\arg\max_{a\in\mathcal{A}}\ \mathbb{E}_{s'\sim T(s'|s,a)}\left(r(s,a,s')+\gamma\cdot V^*(s')\right)$$

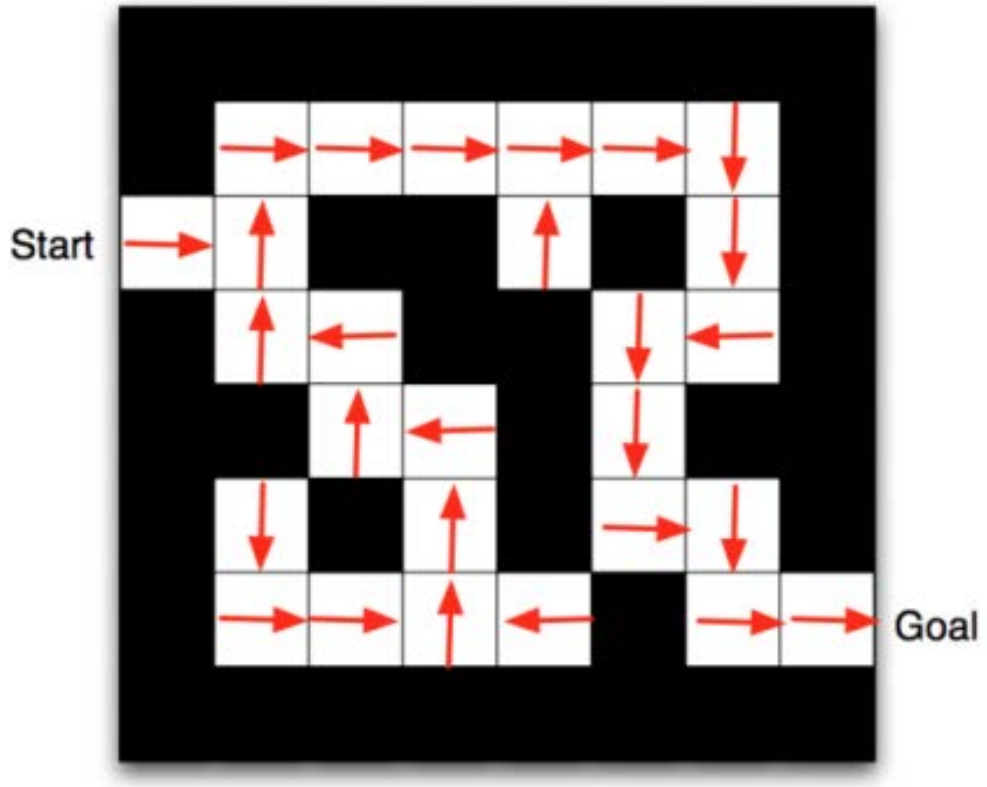
- 小结

- ✓ 强化学习的基础
- ✓ 贝尔曼方程（不同形式）
- ✓ 最优值函数的计算（间接、难用）

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} \ \mathbb{E}_{s' \sim T(s'|s,a)} \left(r(s,a,s') + \gamma \cdot V^*(s') \right)$$

强化学习：Q学习





直接学出动作？

强化学习：Q学习

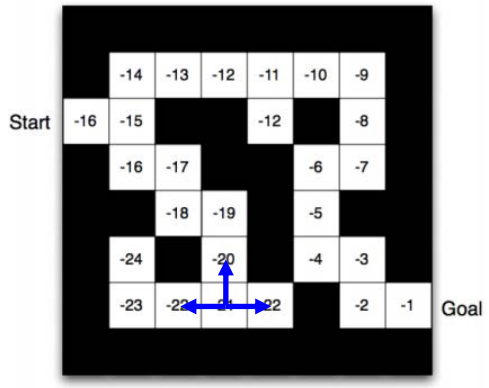
- 从值函数到Q函数

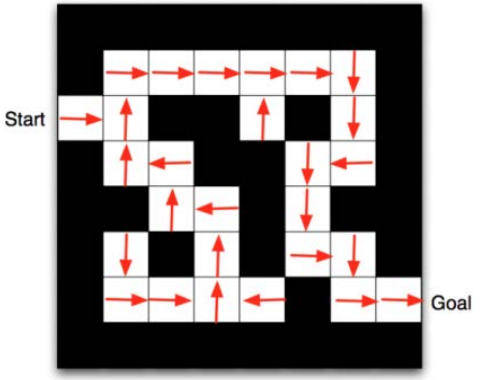
$$V^{\pi}(s)=\mathrm{E}_{\tau\sim p_{\pi}(\tau)}\left\{\sum_{t=0}^{+\infty}\gamma^{t}r(s_t,a_t,s_{t+1})\,|\,s_0=s\right\}$$

$$\pi^{*}(s)=\arg\max_{a\in\mathcal{A}}\ \mathbb{E}_{s'\sim T(s'|s,a)}\left(r(s,a,s')+\gamma\cdot V^{*}(s')\right)$$

“状态-动作”价值函数

$$Q^{\pi}(s,a)=\mathbb{E}_{s'\sim T(s'|s,a)}\left(r(s,a,s')+\gamma\cdot V^{\pi}(s')\right)$$





强化学习：Q学习

- Q函数的物理意义

$$Q^{\pi}(s,a)=\mathbb{E}_{s'\sim T(s'|s,a)}\left(r(s,a,s')+\gamma\cdot V^{\pi}(s')\right)$$

智能体在某一状态s时，如果使用动作a（可以是任意可行的动作），然后继续一直使用策略的累计奖励期望值

$$\pi^{*}(s)=\arg\max_{a\in\mathcal{A}}Q^{*}(s,a)$$

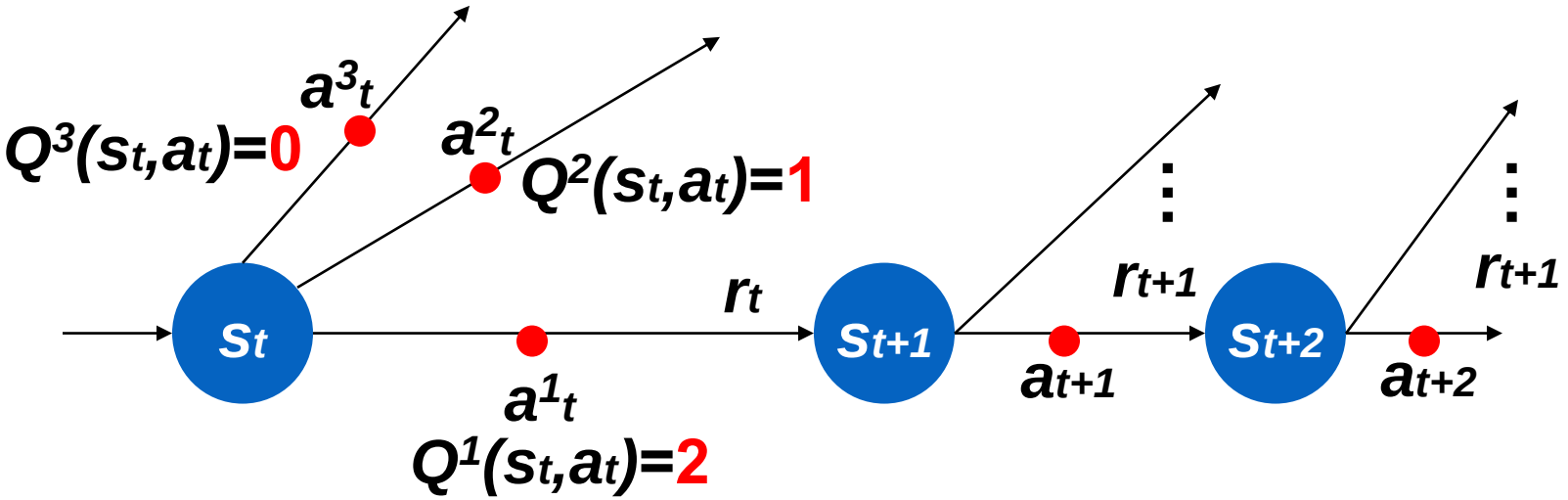
一旦最优的Q确定，策略的计算异常方便。

强化学习：Q学习

- Q函数的性质

Q value

$$Q(s_t, a_t) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots$$



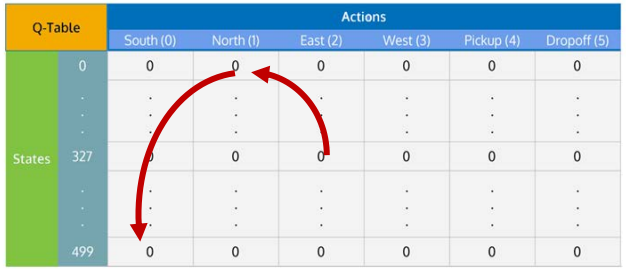
强化学习：Q学习

➢ Q函数的性质

$$Q(s_t, a_t) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots$$

$$= \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2}$$

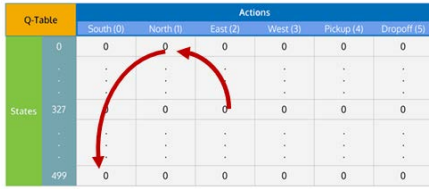
$$= r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$$



As a result, the Q value at time *t* is easily calculated by r_{t+1} and Q value of the next step.

强化学习：Q学习

- Q函数的计算



$$Q(s_t, a_t) = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$$

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\left[r + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

target value　　current value

α: step size parameter　(learning rate)

强化学习：Q学习

➢ 迭代算法

$$Q^*(s,a)=\mathbb{E}_{s'\sim T(s'|s,a)}\left(r(s,a,s')+\gamma\cdot\max_{a'\in\mathcal{A}}Q^*(s',a')\right)$$

$$Q(s,a)\leftarrow Q(s,a)+\eta\left(r(s,a,s')+\gamma\max_{a'\in\mathcal{A}}Q(s',a')-Q(s,a)\right)$$

算法 3.2：Q 学习算法

目标：最优“状态-动作”价值函数 $Q^*(s,a)$。

Step 1：初始化：令 $k=0$，对所有 $s\in\mathcal{S}$，$a\in\mathcal{A}$，设 $Q^{(k)}(s,a)=0$。

Step 2：$k=k+1$

Step 3：随机选择初始状态 $s\in\mathcal{S}$，按照 ε 贪婪策略选择动作

$$a=\begin{cases}\mathcal{A}\text{中随机选择} & \varepsilon\text{概率}\\ \arg\max\limits_{a\in\mathcal{A}}Q^{(k-1)}(s,a) & 1-\varepsilon\text{概率}\end{cases}$$

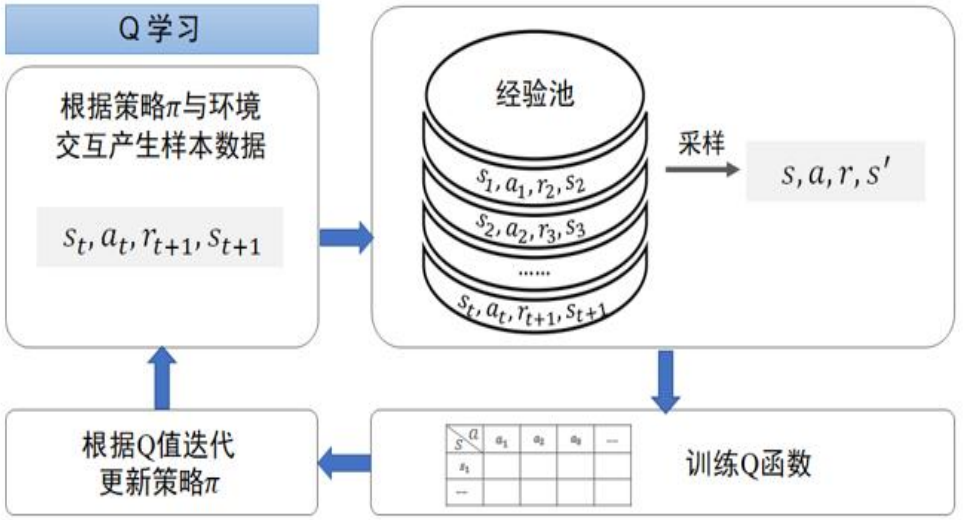
并记录更新的状态 s' 与奖励 $r(s,a,s')$

Step 4：更新 Q 函数

$$Q^{(k)}(s,a)=Q^{(k-1)}(s,a)+\alpha\left(r(s,a,s')+\gamma\max_{a'\in\mathcal{A}}Q^{(k-1)}(s',a')-Q^{(k-1)}(s,a)\right)$$

Step 5：更新状态：$s\leftarrow s'$。如果 s 不是终止节点，返回 Step 3。

Step 6：如果 $\sum_{s\in\mathcal{S},a\in\mathcal{A}}|Q^{(k)}(s,a)-Q^{(k-1)}(s,a)|<\text{TOL}$，令 $Q^*(s,a)=Q^{(k)}(s,a)$，退出循环；



强化学习：Q学习

➢ 迭代算法的收敛

- **Theorem**: if every state-action pair visited infinitely often, 0≤γ<1, and |rewards| ≤ C (some constant), then

∀s, a

$$\lim_{t->\infty} \hat{Q}_t(s,a) = Q_{actual}(s,a)$$

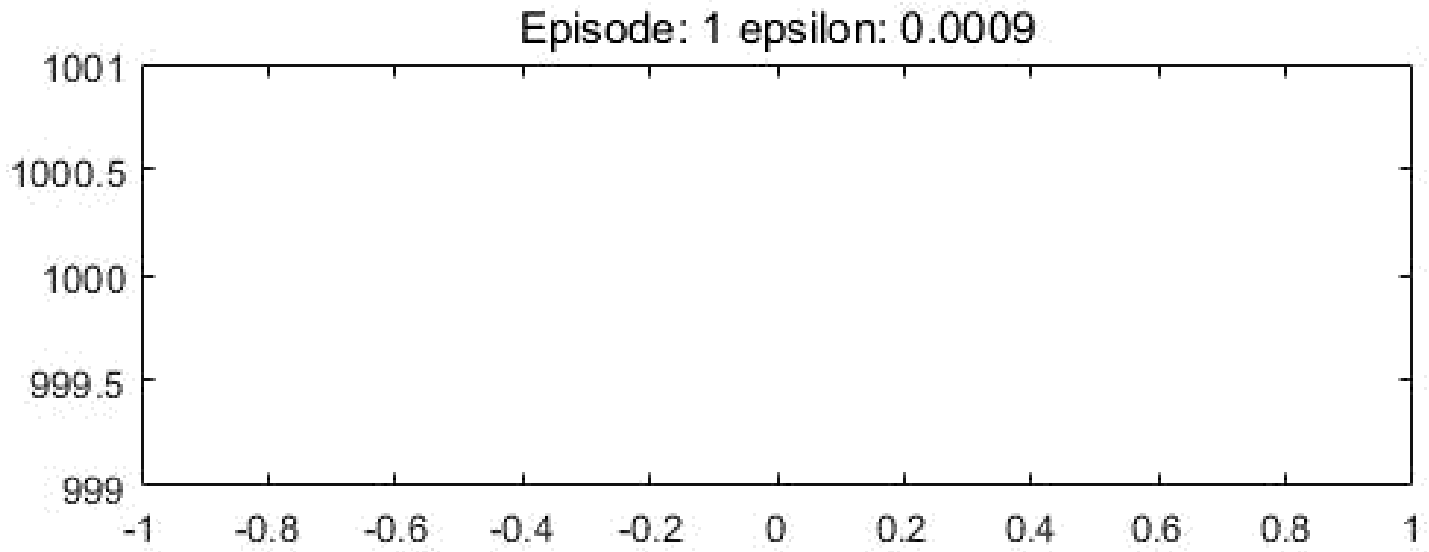
Initialized

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	327	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	499	0	0	0	0	0	0

Training

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

强化学习：Q学习


Episode: 1 epsilon: 0.0009
1001
1000.5
1000
999.5
999
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

强化学习：Q学习

➢ 探索与利用（Exploration v.s. Exploitation）

算法 3.2：Q 学习算法

目标：最优“状态-动作”价值函数 $Q^*(s,a)$。

Step 1：初始化：令 $k=0$，对所有 $s\in\mathcal{S}$，$a\in\mathcal{A}$，设 $Q^{(k)}(s,a)=0$。

Step 2：$k=k+1$

Step 3：随机选择初始状态 $s\in\mathcal{S}$，按照 ε 贪婪策略选择动作

$$a=\begin{cases}\mathcal{A}\text{中随机选择} & \varepsilon\text{概率}\\ \arg\max\limits_{a\in\mathcal{A}} Q^{(k-1)}(s,a) & 1-\varepsilon\text{概率}\end{cases}$$

并记录更新的状态 s' 与奖励 $r(s,a,s')$

Step 4：更新 Q 函数

$$Q^{(k)}(s,a)=Q^{(k-1)}(s,a)+\alpha\left(r(s,a,s')+\gamma\max_{a'\in\mathcal{A}}Q^{(k-1)}(s',a')-Q^{(k-1)}(s,a)\right)$$

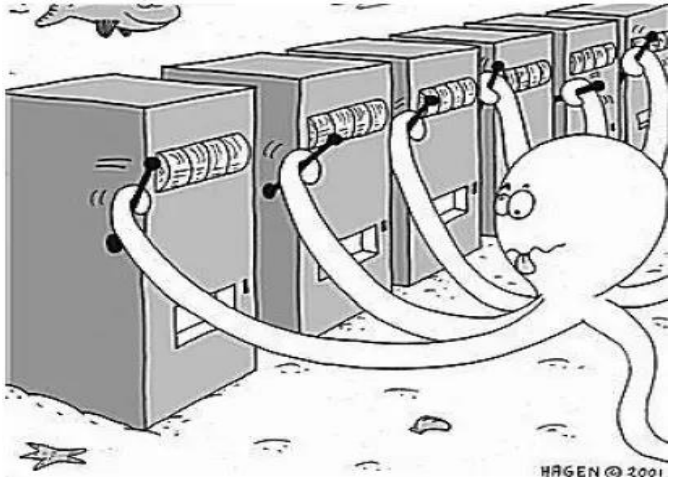
Step 5：更新状态：$s\leftarrow s'$。如果 s 不是终止节点，返回 Step 3。

Step 6：如果 $\sum\limits_{s\in\mathcal{S},a\in\mathcal{A}}|Q^{(k)}(s,a)-Q^{(k-1)}(s,a)|<\text{TOL}$，令 $Q^*(s,a)=Q^{(k)}(s,a)$，退出循环；

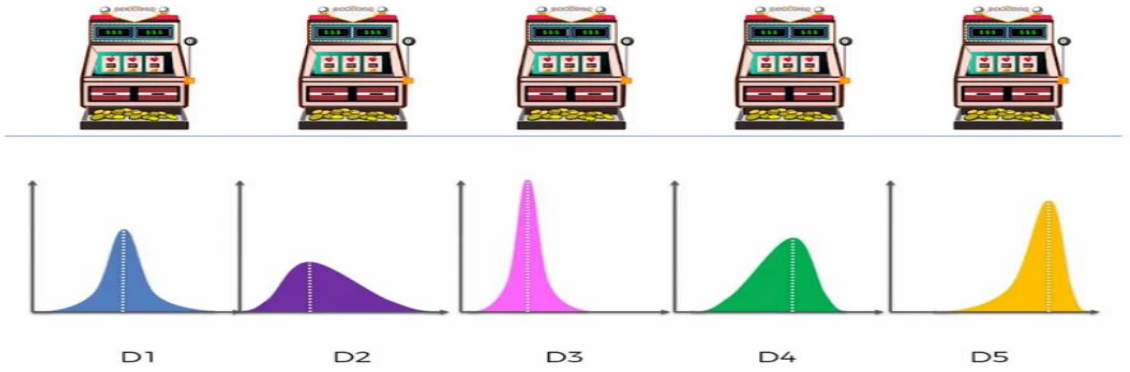
Step 3：随机选择初始状态 $s\in\mathcal{S}$，按照 ε 贪婪策略选择动作

$$a=\begin{cases}\mathcal{A}\text{中随机选择} & \varepsilon\text{概率}\\ \arg\max\limits_{a\in\mathcal{A}} Q^{(k-1)}(s,a) & 1-\varepsilon\text{概率}\end{cases}$$

- 探索（Exploration ）的目的是找到更多有关环境的信息，是尝试还未尝试过的动作行为
- 利用（Exploitation）的目的是利用已知的环境信息来最大限度地提高奖励，是从已知动作中选择下一步的动作



➢ 多臂老虎机问题 Multi-armed Bandit Problem



强化学习：Q学习

- 值函数V v.s. “状态-动作”值函数Q

值函数$V^{\pi}(s)$与“状态-动作”价值函数$Q^{\pi}(s,a)$的区别

值函数$V^{\pi}(s)$反映智能体处于状态s时，如果利用策略π可以获得的期望累计奖励。“状态-动作”价值函数$Q^{\pi}(s,a)$反映智能体处于状态s时，如果使用动作a，然后继续一直使用策略π到结束的累计奖励的期望值。二者的一个重要区别就是$Q^{\pi}(s,a)$考虑了执行动作a后的后果。

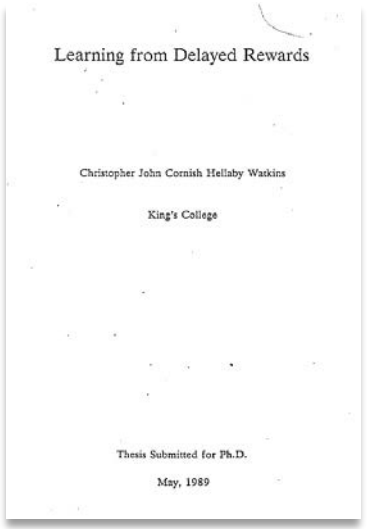
一旦二者确定，我们可以用如下两个式子来对应设计最优动作策略：

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} \mathbb{E}_{s' \sim T(s'|s,a)} \left(r(s,a,s') + \gamma \cdot V^*(s') \right)$$

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s,a)$$

可以看到，如果利用最优值函数$V^*(s)$，则在确定最优策略的时候仍然需要状态转移概率$T(s'|s,a)$，这一信息通常是无法获取的。而利用最优“状态-动作”价值函数$Q^*(s,a)$，则只需简单枚举即可。

强化学习：Q学习



Learning from Delayed Rewards

Christopher John Cornish Hellaby Watkins

King's College

Thesis Submitted for Ph.D.

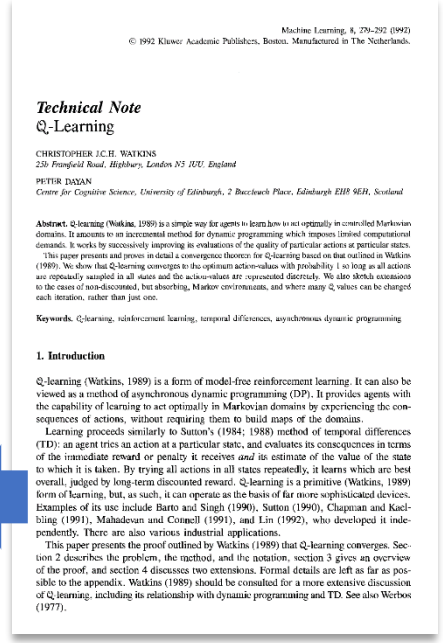
May, 1989

A simpler method of comparing f and g using V_f only is as follows. Consider the expected returns from following policy g for one step, and then following policy f thereafter. Suppose that the policy g recommends action b at state x, while policy f recommends action a. The expected return from starting at x, following policy g for one step (i.e. taking action b) and then following policy f thereafter is

$$Q_f(x,b) = \rho(x,b) + \sum_y P_{xy}(b)\, V_f(y)$$

Abstract. Q-learning (Watkins, 1989) is a simple way for agents to learn how to act optimally in controlled Markovian domains. It amounts to an incremental method for dynamic programming which imposes limited computational demands. It works by successively improving its evaluations of the quality of particular actions at particular states.

This paper presents and proves in detail a convergence theorem for Q-learning based on that outlined in Watkins (1989). We show that Q-learning converges to the optimum action-values with probability 1 so long as all actions are repeatedly sampled in all states and the action-values are represented discretely. We also sketch extensions to the cases of non-discounted, but absorbing, Markov environments, and where many Q values can be changed each iteration, rather than just one.



Machine Learning, 8, 279–292 (1992)
© 1992 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Technical Note
Q-Learning

CHRISTOPHER J.C.H. WATKINS
25b Framfield Road, Highbury, London N5 1UU, England

PETER DAYAN
Centre for Cognitive Science, University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9EH, Scotland

Abstract. Q-learning (Watkins, 1989) is a simple way for agents to learn how to act optimally in controlled Markovian domains. It amounts to an incremental method for dynamic programming which imposes limited computational demands. It works by successively improving its evaluations of the quality of particular actions at particular states.

This paper presents and proves in detail a convergence theorem for Q-learning based on that outlined in Watkins (1989). We show that Q-learning converges to the optimum action-values with probability 1 so long as all actions are repeatedly sampled in all states and the action-values are represented discretely. We also sketch extensions to the cases of non-discounted, but absorbing, Markov environments, and where many Q values can be changed each iteration, rather than just one.

Keywords. Q-learning, reinforcement learning, temporal differences, asynchronous dynamic programming

1. Introduction

Q-learning (Watkins, 1989) is a form of model-free reinforcement learning. It can also be viewed as a method of asynchronous dynamic programming (DP). It provides agents with the capability of learning to act optimally in Markovian domains by experiencing the consequences of actions, without requiring them to build maps of the domains.

Learning proceeds similarly to Sutton's (1984; 1988) method of temporal differences (TD): an agent tries an action at a particular state, and evaluates its consequences in terms of the immediate reward or penalty it receives *and* its estimate of the value of the state to which it is taken. By trying all actions in all states repeatedly, it learns which are best overall, judged by long-term discounted reward. Q-learning is a primitive (Watkins, 1989) form of learning, but, as such, it can operate as the basis of far more sophisticated devices. Examples of its use include Barto and Singh (1990), Sutton (1990), Chapman and Kaelbling (1991), Mahadevan and Connell (1991), and Lin (1992), who developed it independently. There are also various industrial applications.

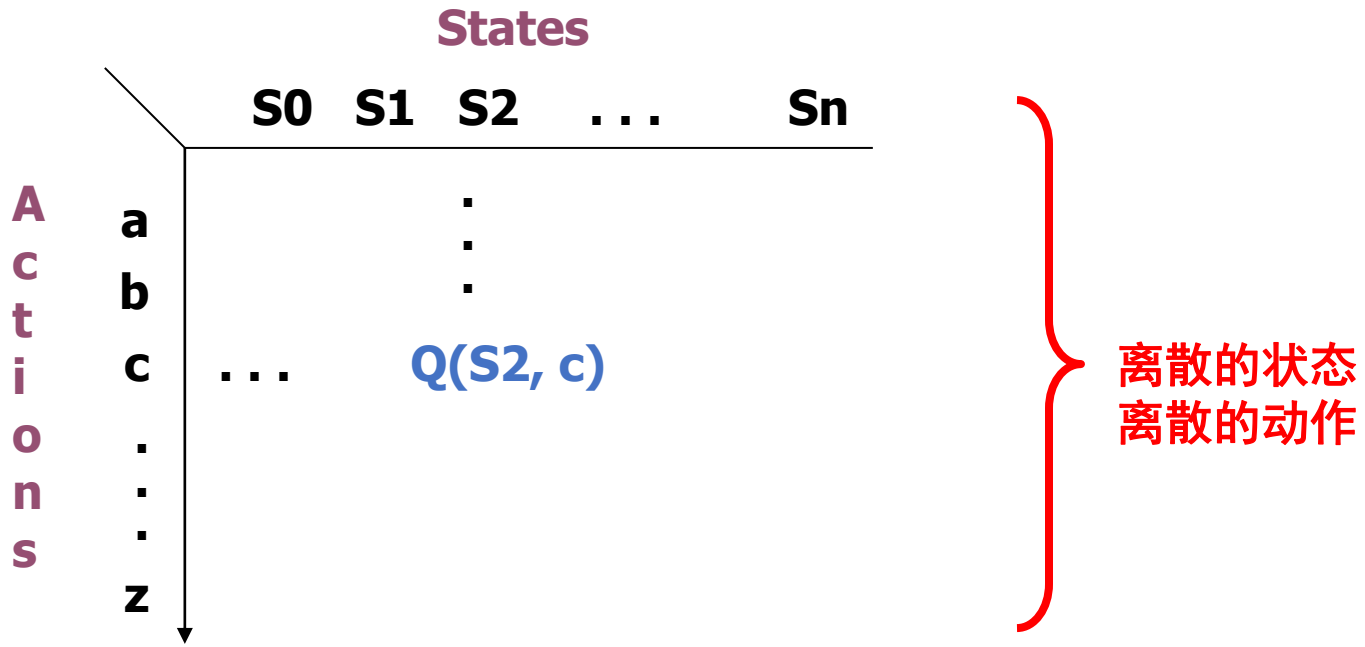
This paper presents the proof outlined by Watkins (1989) that Q-learning converges. Section 2 describes the problem, the method, and the notation, section 3 gives an overview of the proof, and section 4 discusses two extensions. Formal details are left as far as possible to the appendix. Watkins (1989) should be consulted for a more extensive discussion of Q-learning, including its relationship with dynamic programming and TD. See also Werbos (1977).

It occurred to me to consider reinforcement learning as a form of incremental dynamic programming. My old 1989 PhD thesis "Learning from Delayed Rewards", introduced a model of reinforcement learning (learning from rewards and punishments) as incrementally optimising control of a Markov Decision Process (MDP), and proposed a new algorithm – which was dubbed "Q-learning" – that could in principle learn optimal control directly without modelling the transition probabilities or expected rewards of the MDP. The first rigorous proof of convergence was in Watkins and Dayan (1992). These innovations helped to stimulate much subsequent research in reinforcement learning. The notion that animals, or 'learning agents' inhabit a MDP, or a POMDP, and that learning consists of finding an optimal policy, has been dominant in reinforcement learning research since, and perhaps these basic assumptions have not been sufficiently examined.

From 1990 to 1995, I worked for hedge funds in London, so I stopped doing active research in reinforcement learning. In 1995 I visited AT&T Bell Labs at Holmdel for a year, in the Adaptive Systems Group, that was led by Larry Jackel, and then by Yann Le Cun. Vladimir Vapnik was in this immensely productive group, and I became interested in kernel methods, which were just becoming popular.

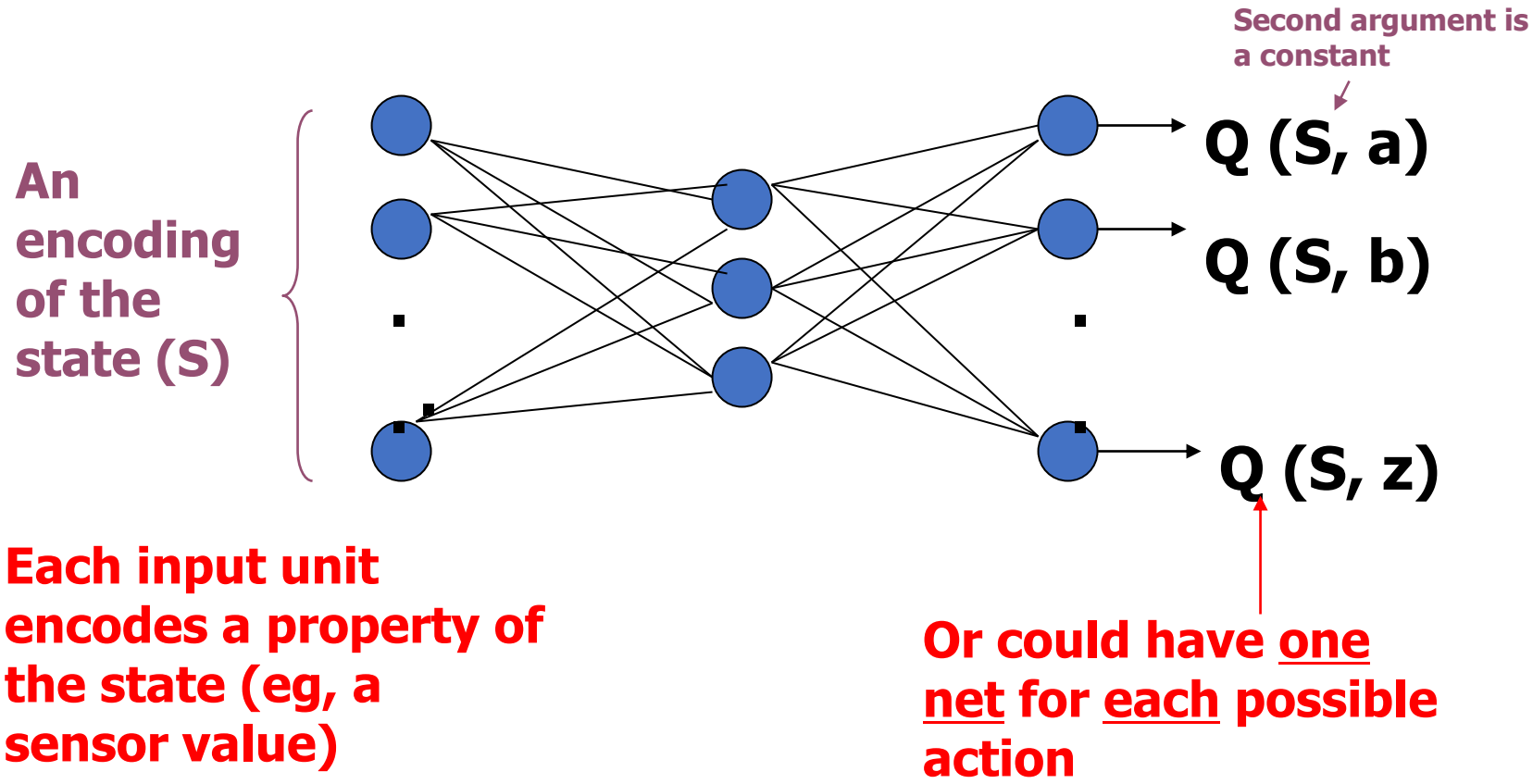
强化学习：Q学习

- Q-学习的缺陷



强化学习：深度Q学习

- 用神经网络逼近Q函数



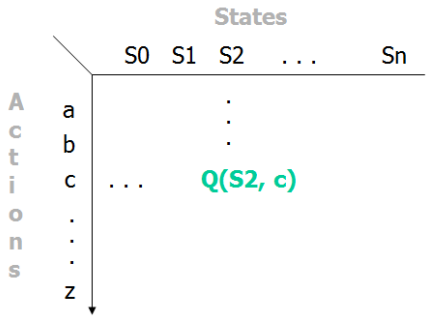
强化学习：深度Q学习

➢ Q Table v.s. Q Net

Given: 100 Boolean-valued features, 10 possible actions

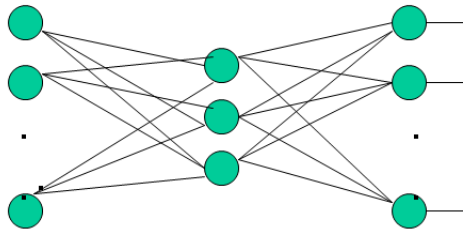
Size of Q table

$10 * 2^{100}$



Size of Q net (100 HU's)

100 * 100 + 100 * 10 = 11,000

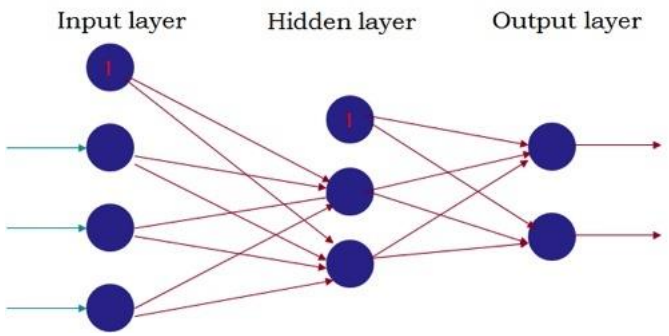


强化学习：深度Q学习

➢ 用神经网络逼近Q函数

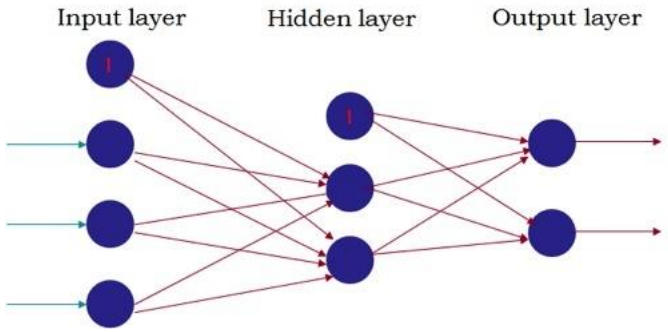
Represent value function by Q-network with weights *w*

$$Q(s, a, w) \approx Q^{\pi}(s, a)$$



强化学习：深度Q学习

- Q网络学习

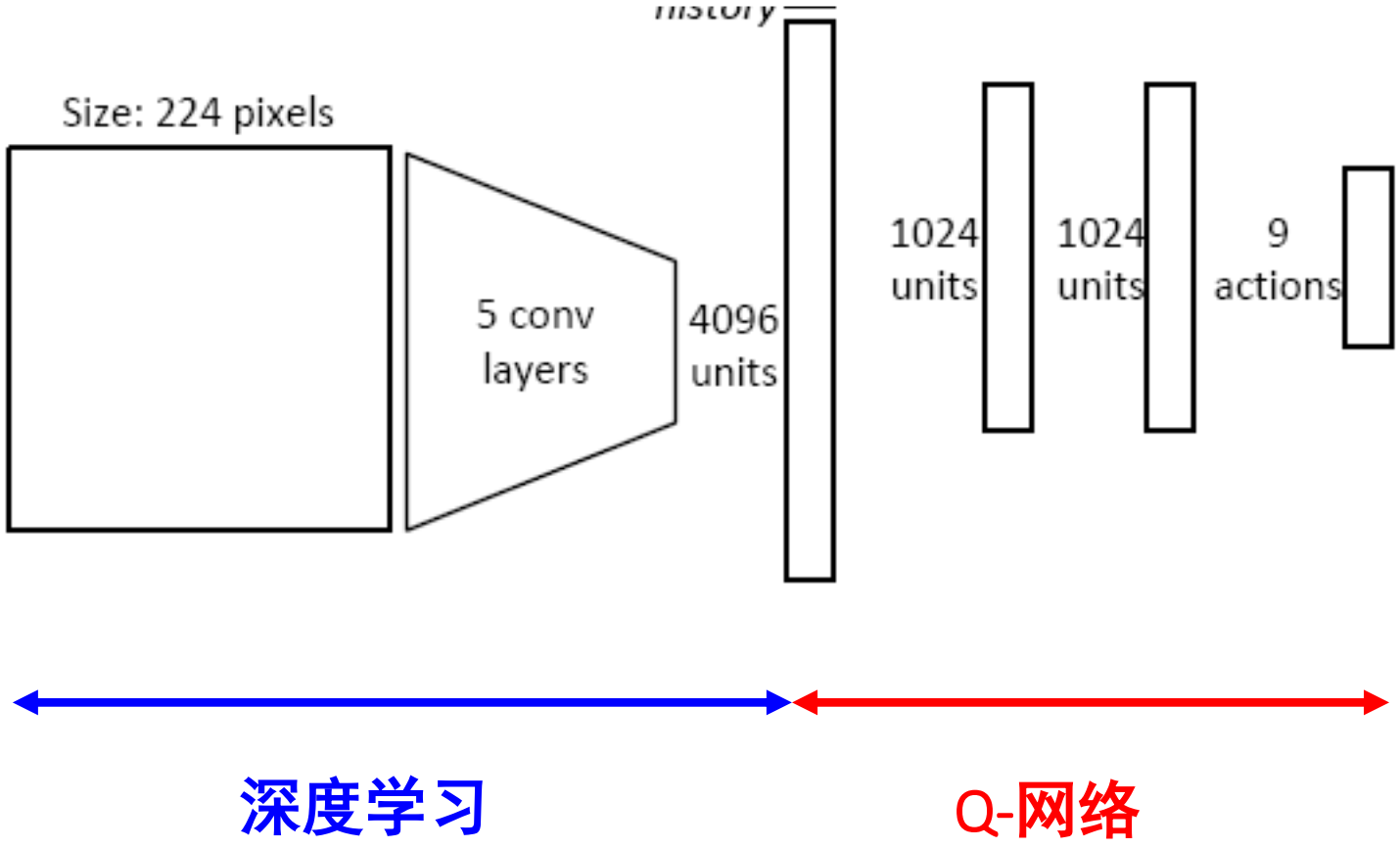


$$\mathcal{L}(w) = \mathbb{E}\left[\left(\underbrace{r + \gamma \max_{a'} Q(s', a', w)}_{\text{target}} - Q(s, a, w)\right)^2\right]$$

$$\frac{\partial \mathcal{L}(w)}{\partial w} = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w)\right) \frac{\partial Q(s, a, w)}{\partial w}\right]$$

强化学习：深度Q学习

- 深度Q网络（DQN）



强化学习：深度Q学习

- 深度Q网络（DQN）

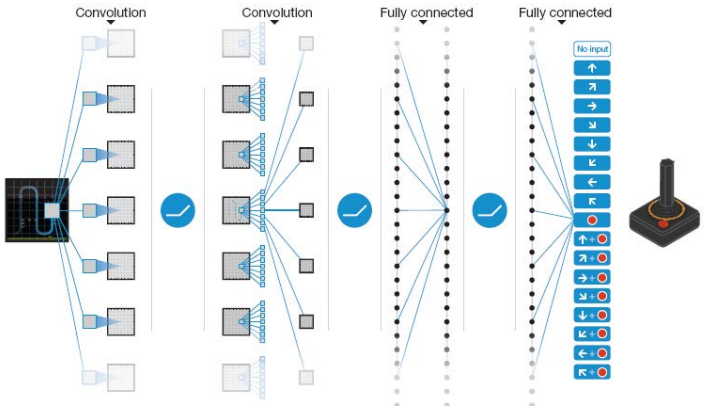
特征提取　　分类识别　　操作执行

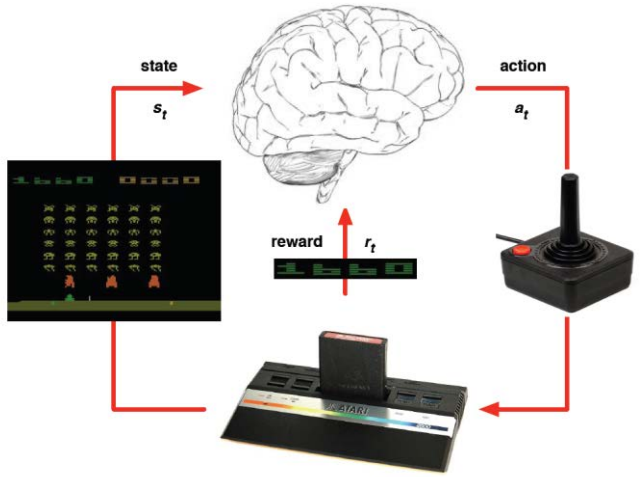
深度学习 ＋ 强化学习

深度强化学习

强化学习：深度Q学习

➢ 深度Q网络（DQN）





LETTER

doi:10.1038/nature14236

Human-level control through deep reinforcement learning

Volodymyr Mnih[1]*, Koray Kavukcuoglu[1]*, David Silver[1]*, Andrei A. Rusu[1], Joel Veness[1], Marc G. Bellemare[1], Alex Graves[1], Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]

强化学习：深度Q学习

➢ 深度Q网络（DQN）

Naive Q-learning oscillates or diverges with neural nets

1. Data is sequential
   - Successive samples are correlated, non-iid
2. Policy changes rapidly with slight changes to Q-values
   - Policy may oscillate
   - Distribution of data can swing from one extreme to another
3. Scale of rewards and Q-values is unknown
   - Naive Q-learning gradients can be large unstable when backpropagated

强化学习：深度Q学习

➢ 深度Q网络（DQN）

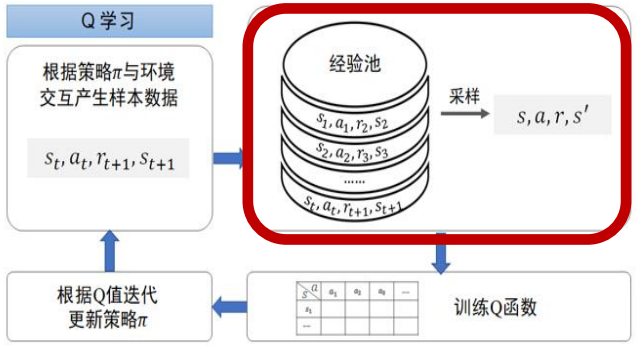
Use experience replay（经验回放）

- Break correlations in data, bring us back to iid setting
- Learn from all past policies

To remove correlations, build data-set from agent's own experience

- Take action a_t according to -greedy policy
- Store transition (s_t; a_t; r_{t+1}; s_{t+1}) in replay memory D
- Sample random mini-batch of transitions (s; a; r ; s_0) from D
- Optimise MSE between Q-network and Q-learning targets, e.g.

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



强化学习：深度Q学习

➢ 深度Q网络（DQN）

To avoid oscillations, fix parameters used in Q-learning target

- Compute Q-learning targets w.r.t. old, fixed parameters w^-

$$r + \gamma \max_{a'} Q(s', a', w^-)$$

- Optimise MSE between Q-network and Q-learning targets

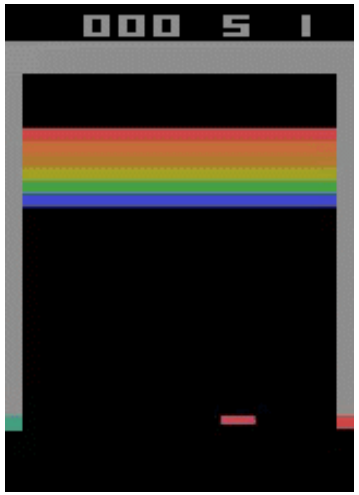
$$\mathcal{L}(w) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\left(r + \gamma \max_{a'} Q(s', a', w^-) - Q(s, a, w) \right)^2 \right]$$

- Periodically update fixed parameters $w^- \leftarrow w$

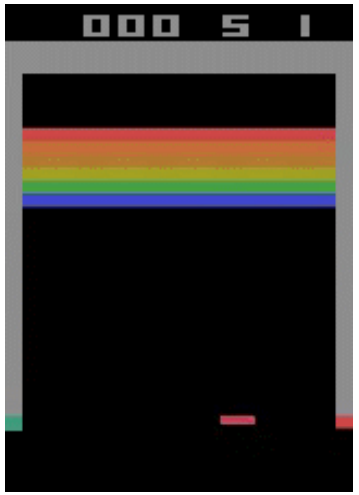
强化学习：深度Q学习

➢ 深度Q网络（DQN）

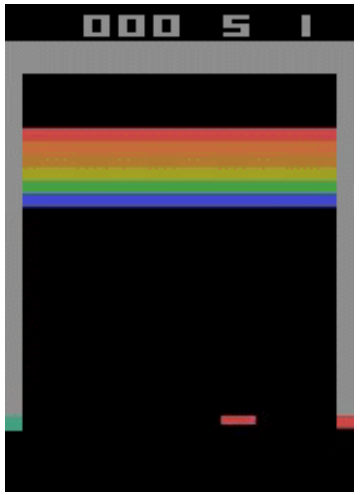
- 图像输入，算法做出向左、向右决策



Episode = 1



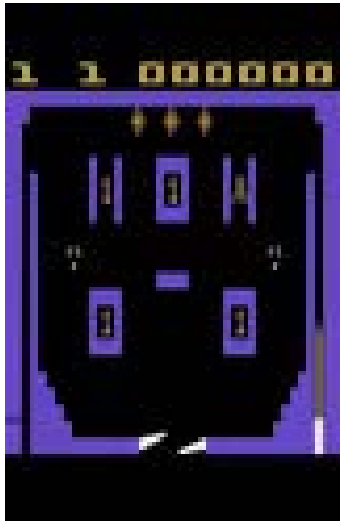
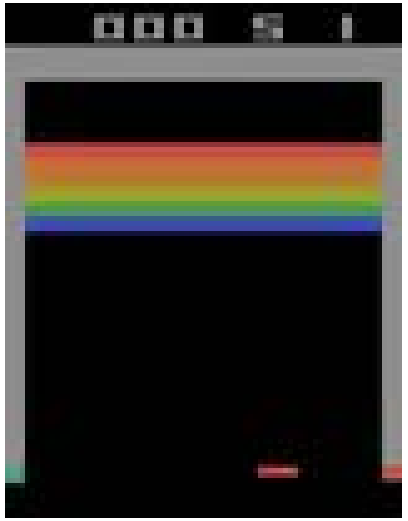
Episode = 900000

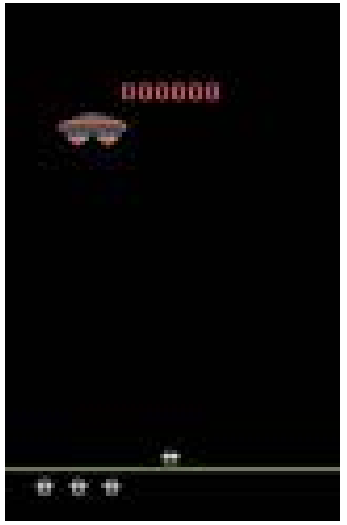


最好效果

强化学习：深度Q学习

- 深度Q网络（DQN）



Goal：

Complete the game with the highest score.

State：

Raw pixel inputs of the game state

Action：

Game controls e.g. Left, Right, Up, Down

Reward：

Score increase/decrease at each time step.

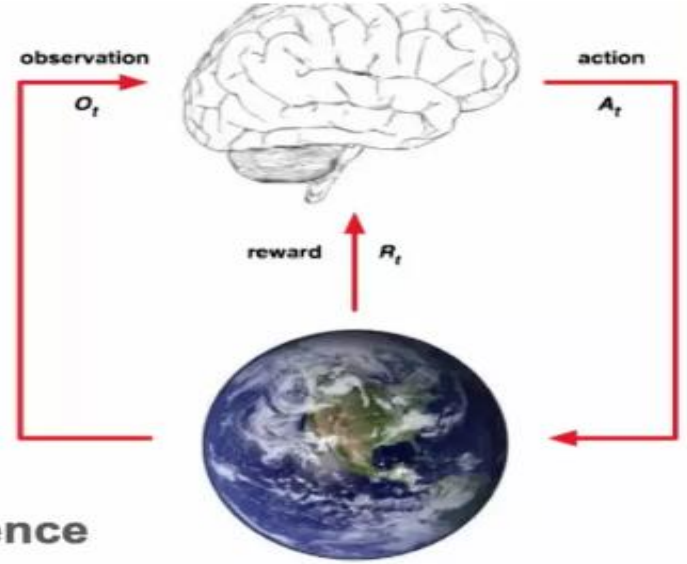
强化学习：深度Q学习

RL+ DL = AI

DeepMind

Deep Reinforcement Learning

Goal: an artificial agent with human-level intelligence (AGI)



Reinforcement learning defines the objective

- Conjecture 1: **RL is enough to formalise the *problem* of intelligence**

Deep learning gives mechanism for optimising objective

- Conjecture 2: **Deep neural networks can *represent* and *learn* any computable function**

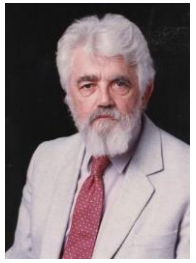
Deep RL combines the RL problem with DL solution

- => Conjecture 3: **RL + DL can solve the problem of intelligence**

强化学习：小结

符号主义

Marvin Lee Minsky
(1927-2016)



John McCarthy
(1927-2011)

强化学习**不是**真的人工智能，因为它缺乏符号表示和逻辑推理，无法处理抽象和复杂的问题

联接主义

Hilton
（1947-）

Yann LeCun

强化学习**是**真的人工智能，因为它模仿了人类大脑的神经网络，可以通过自适应和学习来解决各种问题

行为主义

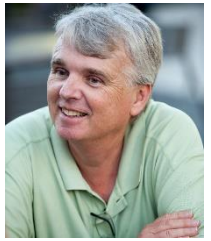
Brooks
(1954-)

Hans Moravec

强化学习**是**真的人工智能，因为它关注了智能体的行为和环境的交互，通过反馈和探索实现自主和适应

统计主义

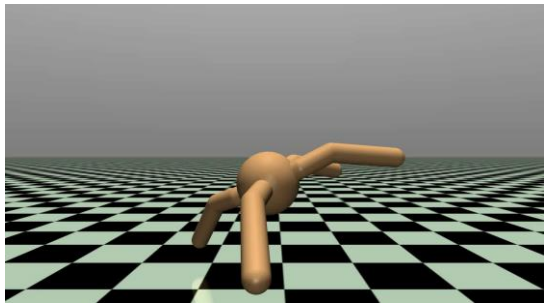
Judea Pearl
（1936-）

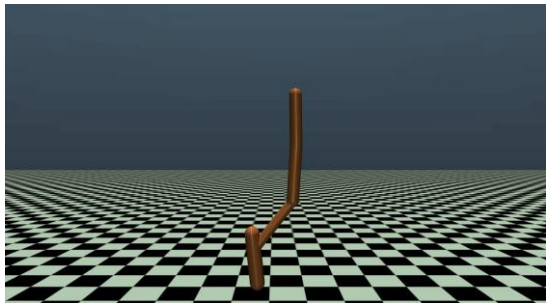


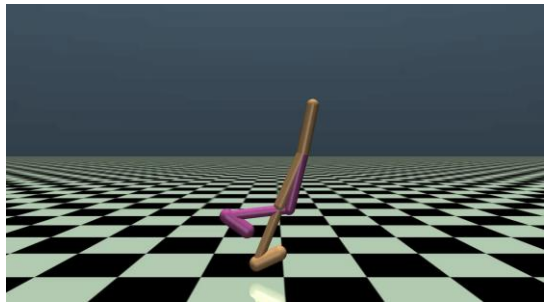
David Heckerman

强化学习**不是**真的人工智能，因为它缺乏概率模型和因果推断，无法处理不确定性和复杂性

➢ 深度Q网络（DQN）的局限







Robot Locomotion

Goal：

Make the robot move forward.

State：

Angle and position of the joints

Action：

Torques applied on joints

Reward：

1 at each time step upright + forward movement.

- 背景
- 强化学习：Q学习
- 强化学习：策略梯度


REINFORCEMENT
LEARNING
CONFERENCE
August 9-12, 2024
Amherst, MA