



上海交通大学  
SHANGHAI JIAO TONG UNIVERSITY



上海交通大学  
约翰·霍普克罗夫特  
计算机科学中心

John Hopcroft Center for Computer Science

# AI3601: Reinforcement Learning

Shuai Li

John Hopcroft Center, Shanghai Jiao Tong University

<https://shuaili8.github.io>

<https://shuaili8.github.io/Teaching/AI3601/index.html>

Part of slide credits: CMU & Berkeley

# Self Introduction

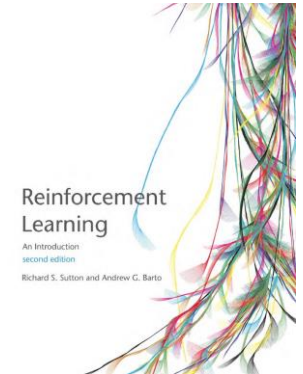
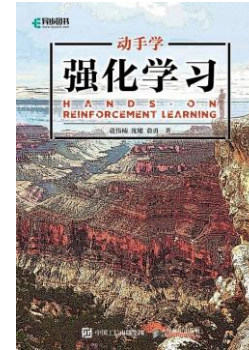
- Position
  - Tenure-track associate professor 2023-now
  - Assistant professor at John Hopcroft Center Aug 2019 - 2022
- Education
  - PhD in Computer Science from the Chinese University of Hong Kong
  - Master in Math from the Chinese Academy of Sciences
  - Bachelor in Math from Chu Kochen Honors College, Zhejiang University
- Research interests
  - Bandit algorithms
  - Reinforcement learning algorithms
  - Machine learning theory
  - Machine learning applications

# Teaching assistant

- Ruofeng Yang (杨若峰)
  - Email: wanshuiyin@sjtu.edu.cn
  - 2<sup>nd</sup> year PhD student
  - Research on diffusion models and reinforcement learning
  - Office hour: Wed 7-9 PM
- Zhijie Wang (王至捷)
  - Email: violetevergarden@sjtu.edu.cn
  - 4th year undergraduate
  - Research on in-context learning and bandit
  - Office hour: Thu 2-4 PM
- Xu Liu (刘旭)
  - Email: liu\_skywalker@sjtu.edu.cn
  - 2<sup>nd</sup> year Master student
  - Research on reinforcement learning and bandits
  - Office hour: Mon 7-9 PM
- Haitong Ma (马海桐)
  - Email: mahaitong@sjtu.edu.cn
  - 4th year undergraduate
  - Research on reinforcement learning and recommender systems
  - Office hour: Tue 7-9 PM

# References (will add more during course)

- 动手学强化学习, 张伟楠等
- Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto



- 参考课程
- UCL David Silver RL Course: <https://www.davidsilver.uk/teaching/>
- Berkeley Sergey Levine Deep RL Course: <http://rail.eecs.berkeley.edu/deeprlcourse/>
- OpenAI DRL Camp: <https://sites.google.com/view/deep-rl-bootcamp/lectures>
- Stanford Emma Brunskill RL Course: <https://web.stanford.edu/class/cs234/>

# Goal

- Know what is RL and what it usually covers
- Be familiar and understand popular RL problems and algorithms
- Be able to build RL models in applications
  - Know which algorithms to adopt and when to adopt
- Get a touch of latest research

# Prerequisites

- Basic computer science principles
  - Big-O notation
  - Comfortably write non-trivial code in Python/numpy/PyTorch
- Probability
  - Random Variables, Expectations, Distributions
- Linear Algebra & Multivariate/Matrix Calculus
  - Gradients and Hessians
  - Eigenvalue/vector
- Basic ideas in Artificial Intelligence and Machine Learning
  - Agents
  - Training & Test

# Grading

- 没有笔试
- Attendance and participance: 10%
- Homework (written & programming): 40%
- Project: 40%
- Presentations: 10%

# Course outline

- Markov decision processes (MDPs)
- Model-based/Model-free Methods
- Value-based/Policy-based Methods
- Deep Reinforcement Learning
- Recent Advances
  - Imitation Learning
  - Offline RL
  - Multi-agent RL

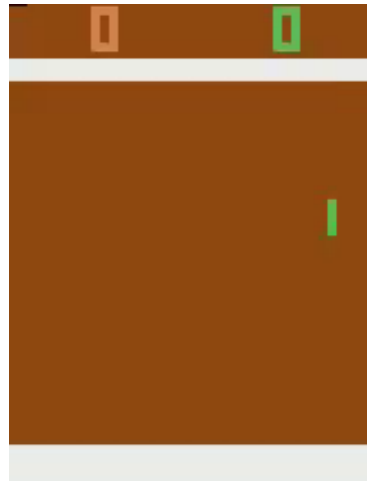


# Recent Progress by Deep Reinforcement Learning

# Deep Reinforcement Learning

2013

Atari (DQN)  
[Deepmind]



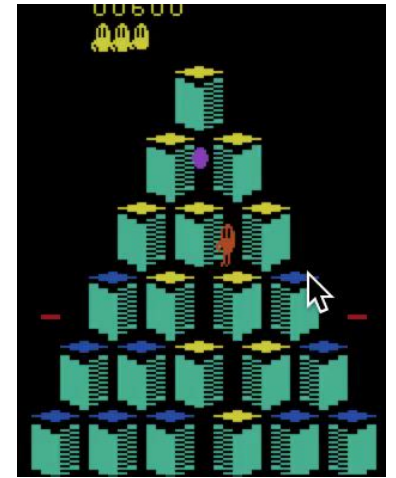
Pong



Enduro



Beamrider



Q\*bert

# Deep Reinforcement Learning 2

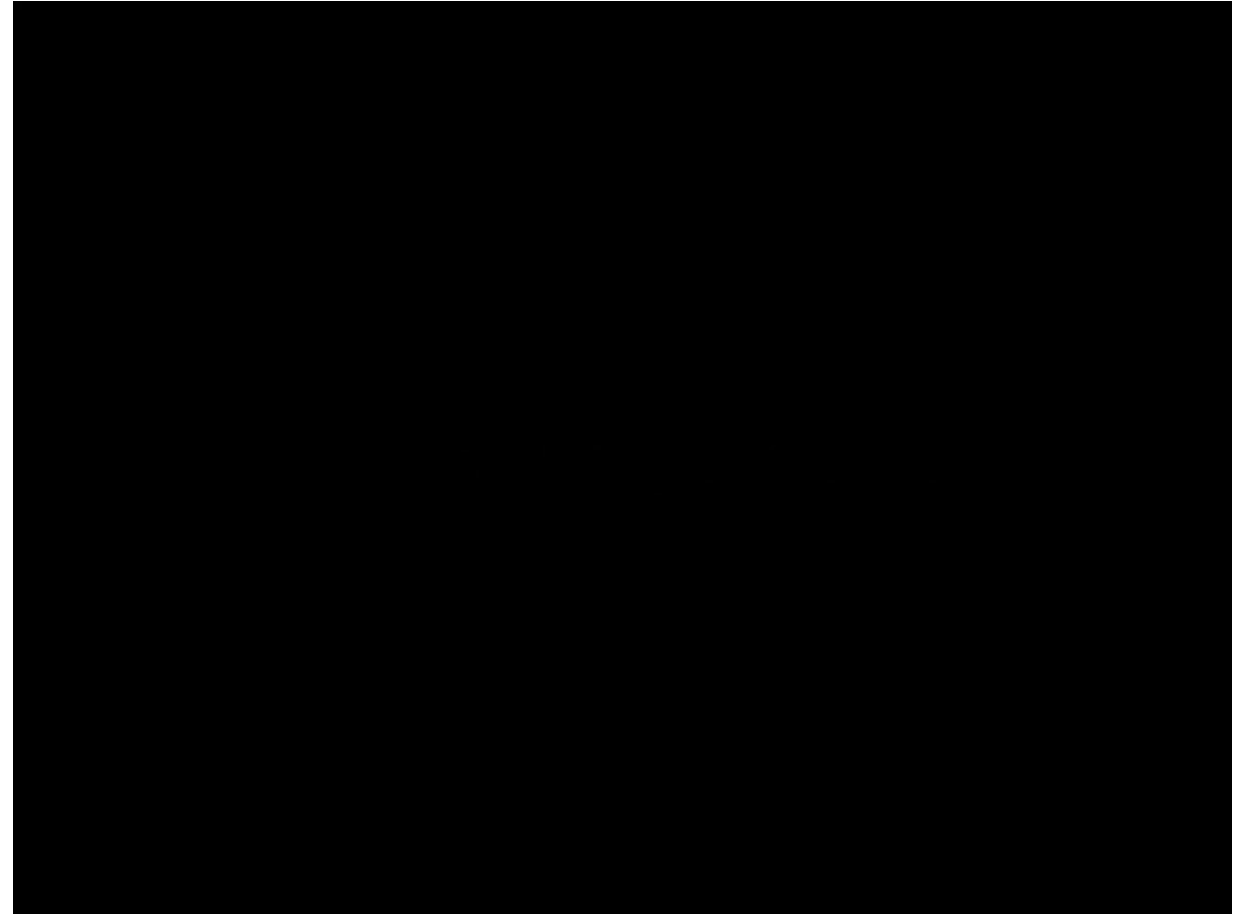
2013

Atari (DQN)  
[Deepmind]

2015

Human-level control  
[Deepmind]

Trained separate DQN agents for 50 different Atari games, without any prior knowledge of the game rules



# Deep Reinforcement Learning 3

2013

Atari (DQN)  
[Deepmind]

2015

Human-level control  
[Deepmind]

AlphaGo  
[Deepmind]



**AlphaGo** Silver et al, Nature 2015

**AlphaGoZero** Silver et al, Nature 2017

**AlphaZero** Silver et al, 2017

Tian et al, 2016; Maddison et al, 2014; Clark et al, 2015

# Deep Reinforcement Learning 4

2013

Atari (DQN)  
[Deepmind]

2015

Human-level control  
[Deepmind]

AlphaGo  
[Deepmind]

2016

3D locomotion (TRPO+GAE)  
[Berkeley]



[Schulman, Moritz, Levine, Jordan, Abbeel, ICLR 2016]

# Deep Reinforcement Learning 5

2013

Atari (DQN)  
[Deepmind]

2015

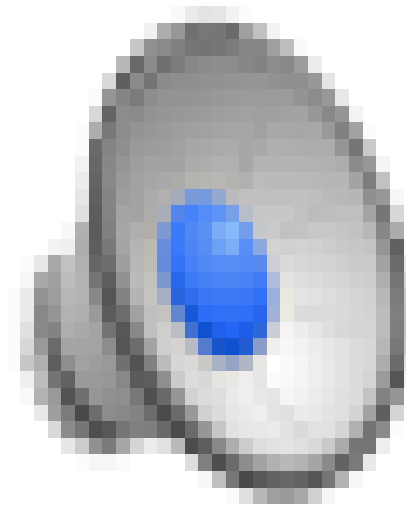
Human-level control  
[Deepmind]

AlphaGo  
[Deepmind]

2016

3D locomotion (TRPO+GAE)  
[Berkeley]

Real Robot Manipulation (GPS)  
[Berkeley]



[Levine\*, Finn\*, Darrell, Abbeel, JMLR 2016]

# Deep Reinforcement Learning 6

2013

Atari (DQN)  
[Deepmind]

2015

Human-level control  
[Deepmind]

AlphaGo  
[Deepmind]

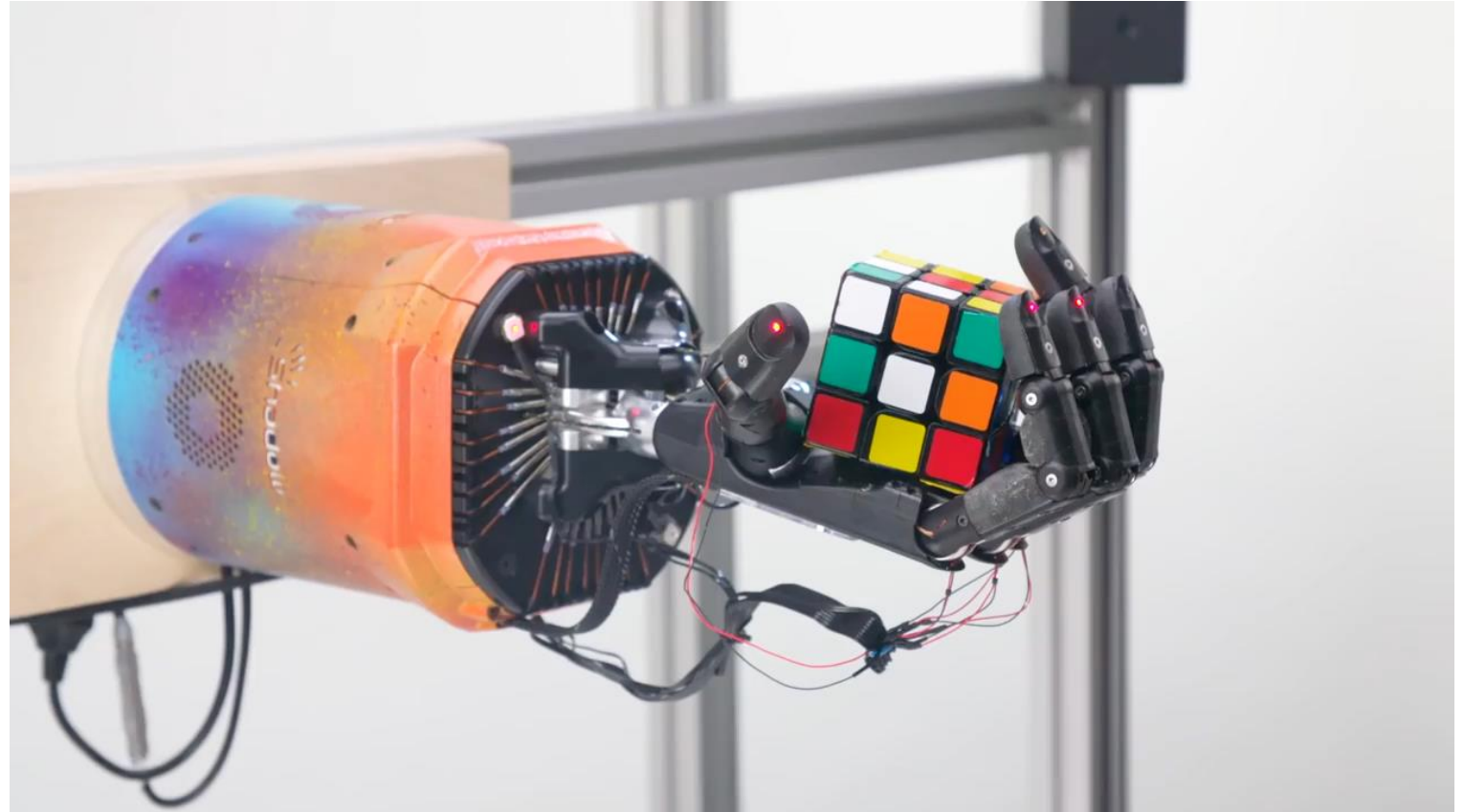
2016

3D locomotion (TRPO+GAE)  
[Berkeley]

Real Robot Manipulation (GPS)  
[Berkeley]

2019

Rubik's Cube (PPO+DR)  
[OpenAI]



OpenAI



# What are some recent advances we've seen in AI?



vibrant portrait painting of Salvador Dalí with a robotic half face



a shiba inu wearing a beret and black turtleneck



a close up of a handpalm with leaves growing from it



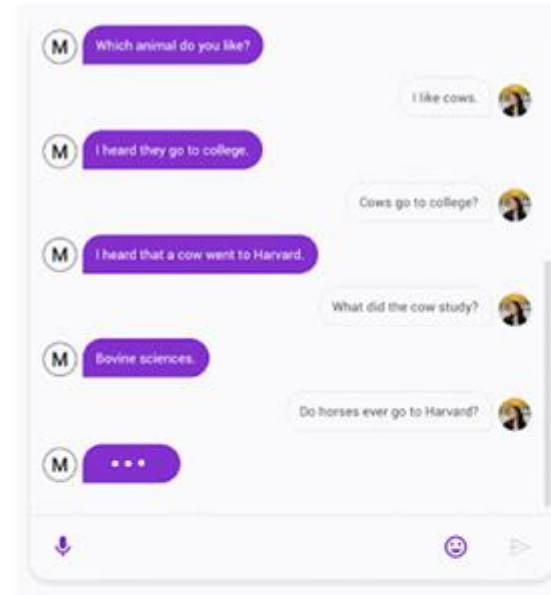
an espresso machine that makes coffee from human teeth, artstation



panda mad scientist mixing sparkling chemicals, artstation



a corgi's head depicted as an explosion of a nebula



## Explaining a joke

### Prompt

Explain this joke:

Joke: Did you see that Google just hired an eloquent whale for their TPU team? It showed them how to communicate between two different pods!

### Model Response

Prediction: TPUs are a type of computer chip that Google uses for deep learning. A "pod" is a group of TPUs. A "pod" is also a group of whales. The joke is that the whale is able to communicate between two groups of whales, but the speaker is pretending that the whale is able to communicate between two

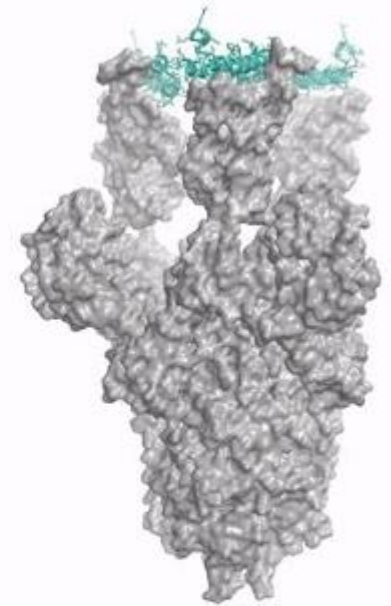
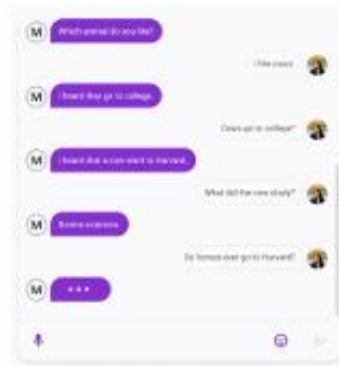


image credit: UW IPD



# What's the main idea behind this?



Explaining a joke

**Prompt**

Explain this joke:

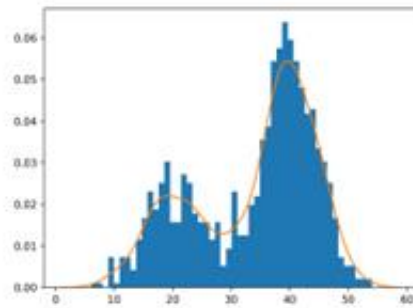
Joke: Did you see that Google just hired an eloquent whale for their TPU team? It showed them how to communicate between two different pods!

**Model Response**

Prediction: TPUs are a type of computer chip that Google uses for deep learning. A "pod" is a group of TPUs. A "pod" is also a group of whales. The joke is that the whale is able to communicate between two groups of whales, but the speaker is pretending that the whale is able to communicate between two groups of TPUs.

$$p_{\theta}(\mathbf{x})$$

$$p_{\theta}(\mathbf{y}|\mathbf{x})$$



# Reinforcement learning can discover new solutions

Impressive because no person had thought of it!



“Move 37” in Lee Sedol AlphaGo match: reinforcement learning “discovers” a move that surprises everyone

Impressive because it looks like something a person might draw!



vibrant portrait painting of Salvador Dalí with a robotic half face

a shiba inu wearing a beret and black turtleneck

a close up of a handpalm with leaves growing from it

an espresso machine that makes coffee from human souls, artistic

panda mad scientist mixing sparkling chemicals, artistic

a corgi's head depicted as an explosion of a nebula

# What is Reinforcement Learning?

# What is reinforcement learning?

- **Mathematical** formalism for learning-based decision making
- Approach for learning decision making and control from **experience**

# How is this different from other machine learning topics?

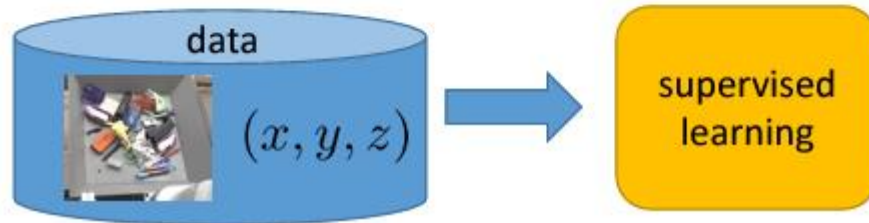
Standard (supervised) machine learning:

given  $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$

learn to predict  $y$  from  $\mathbf{x}$        $f(\mathbf{x}) \approx y$

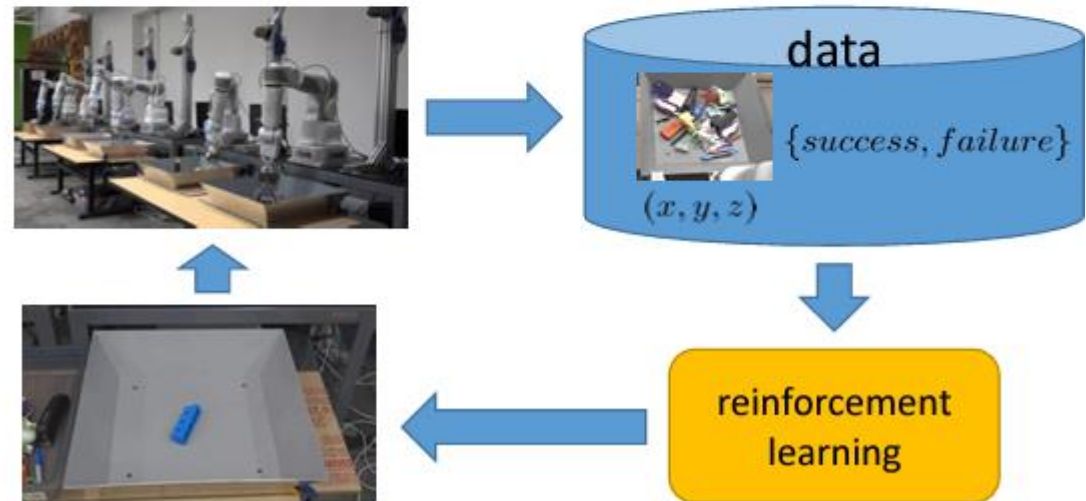
Usually assumes:

- i.i.d. data
- known ground truth outputs in training



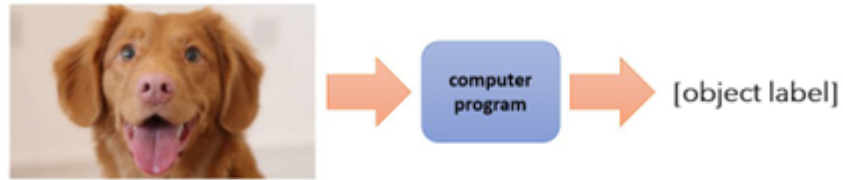
Reinforcement learning:

- Data is **not** i.i.d.: previous outputs influence future inputs!
- Ground truth answer is not known, only know if we succeeded or failed
  - more generally, we know the reward



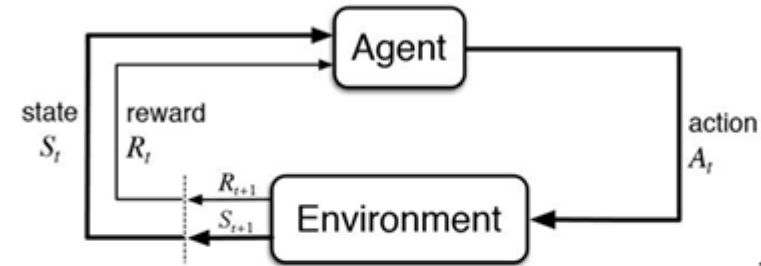
# What is reinforcement learning?

## supervised learning



input:  $\mathbf{x}$   
output:  $\mathbf{y}$   
data:  $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$   
goal:  $f_{\theta}(\mathbf{x}_i) \approx \mathbf{y}_i$  ← someone gives this to you

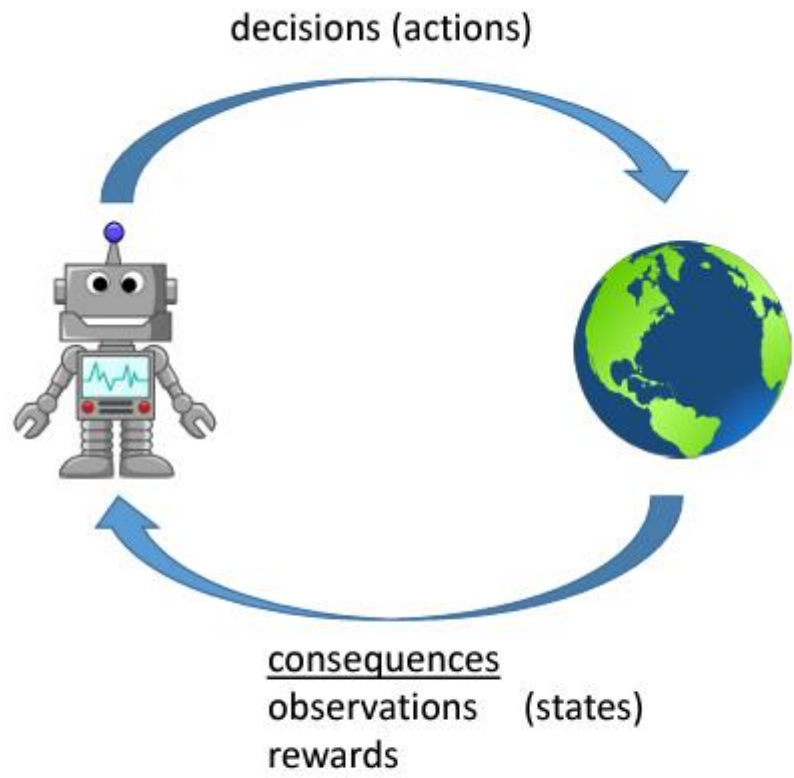
## reinforcement learning



input:  $\mathbf{s}_t$  at each time step  
output:  $\mathbf{a}_t$  at each time step  
data:  $(\mathbf{s}_1, \mathbf{a}_1, r_1, \dots, \mathbf{s}_T, \mathbf{a}_T, r_T)$   
goal: learn  $\pi_{\theta} : \mathbf{s}_t \rightarrow \mathbf{a}_t$   
to maximize  $\sum_t r_t$

pick your own actions





Actions: muscle contractions  
Observations: sight, smell  
Rewards: food

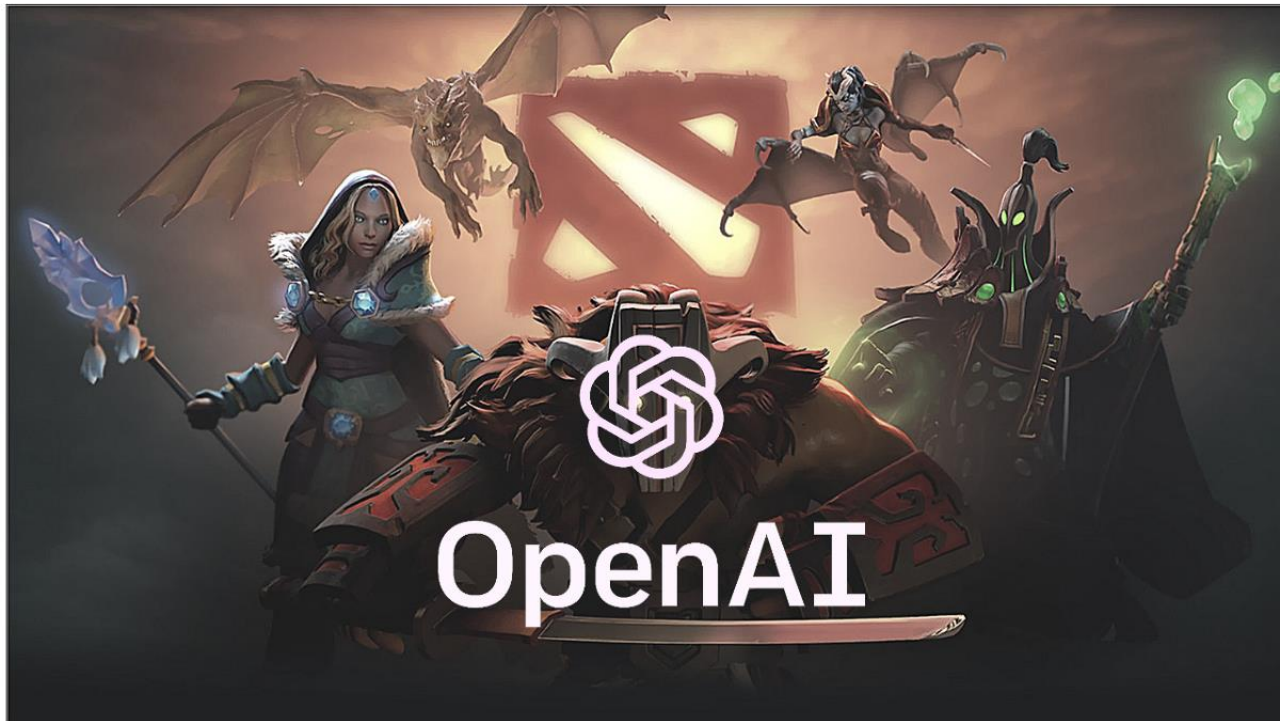


Actions: motor current or torque  
Observations: camera images  
Rewards: task success measure (e.g., running speed)



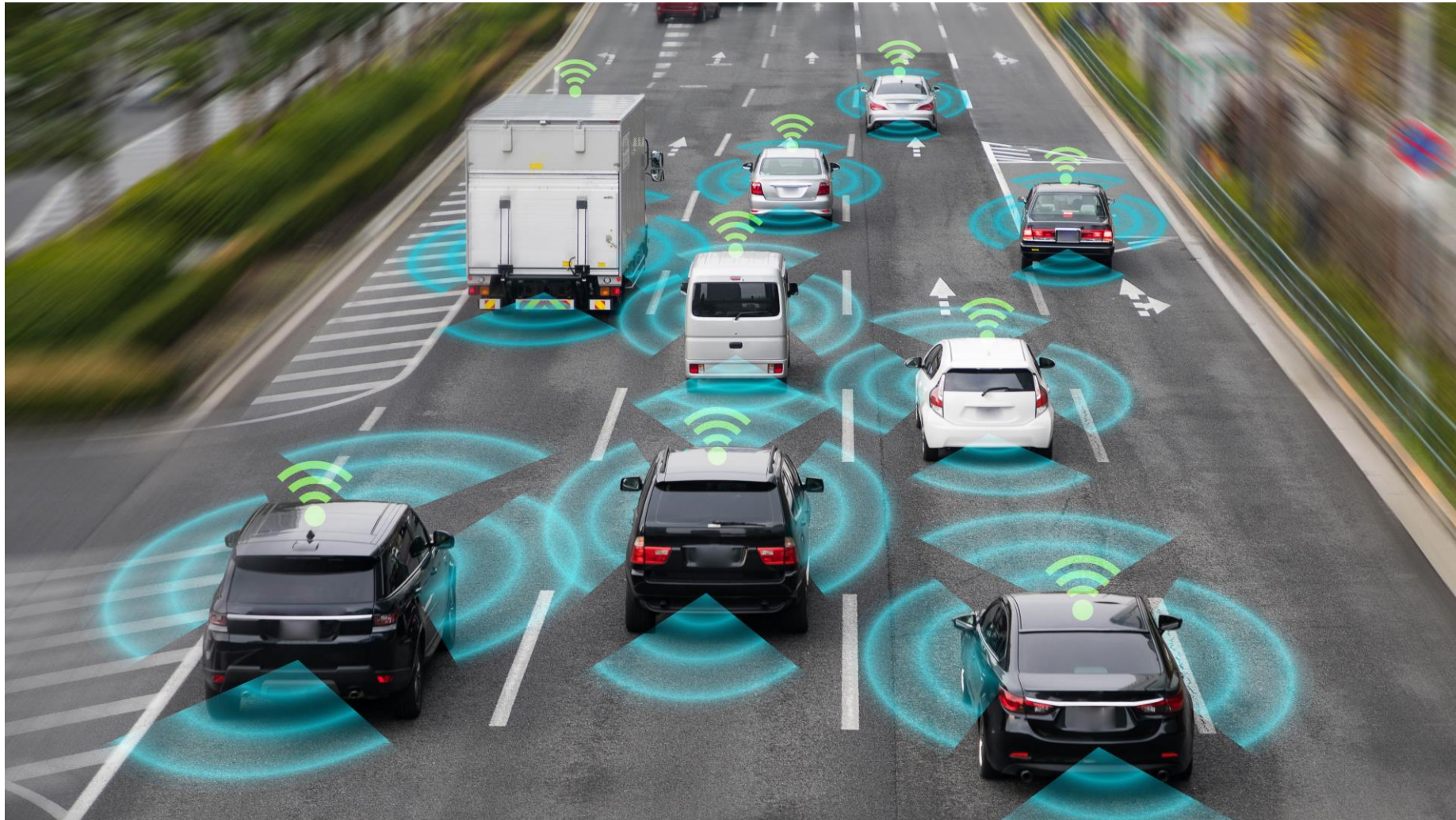
Actions: what to purchase  
Observations: inventory levels  
Rewards: profit

# Complex tasks!

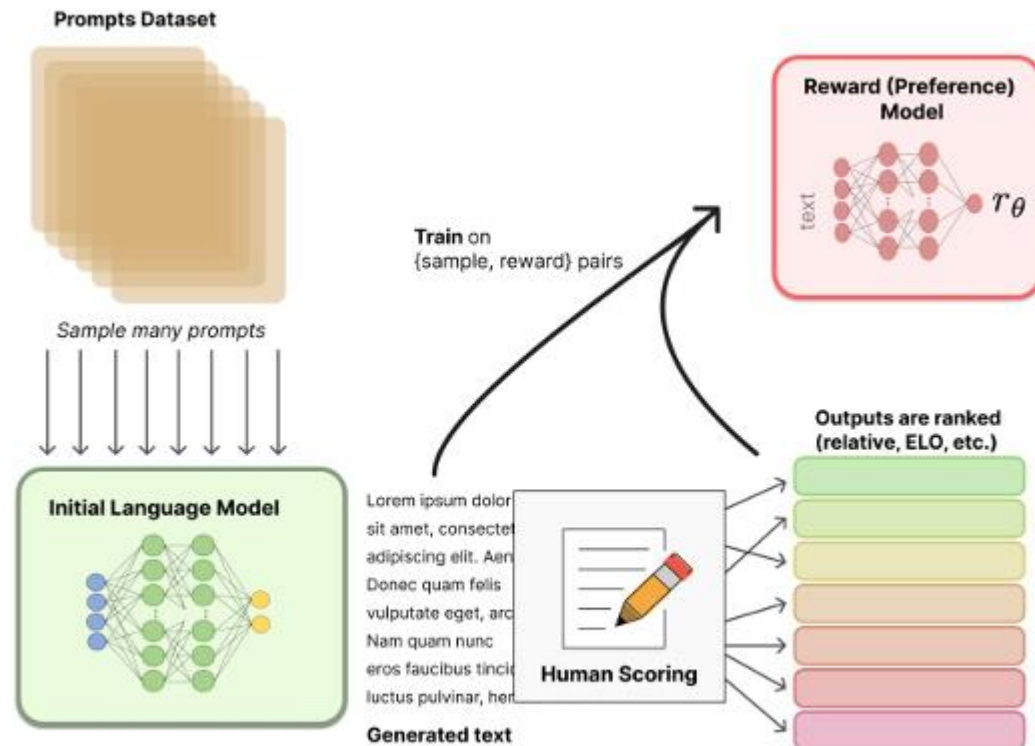




# Not just games and robots!



# Reinforcement learning with language models



LE Can you explain to me how RL with human feedback works for language models like I am five?

Sure! So, imagine you have a robot named Rufus who wants to learn how to talk like a human. Rufus has a language model that helps him understand words and sentences.

First, Rufus will say something using his language model. For example, he might say "I am a robot."

Then, a human will listen to what Rufus said and give him feedback on whether it sounded like a natural sentence a human would say. The human might say, "That's not quite right, Rufus. Humans don't usually say 'I am a robot.' They might say 'I'm a robot' or 'I am a machine.'"

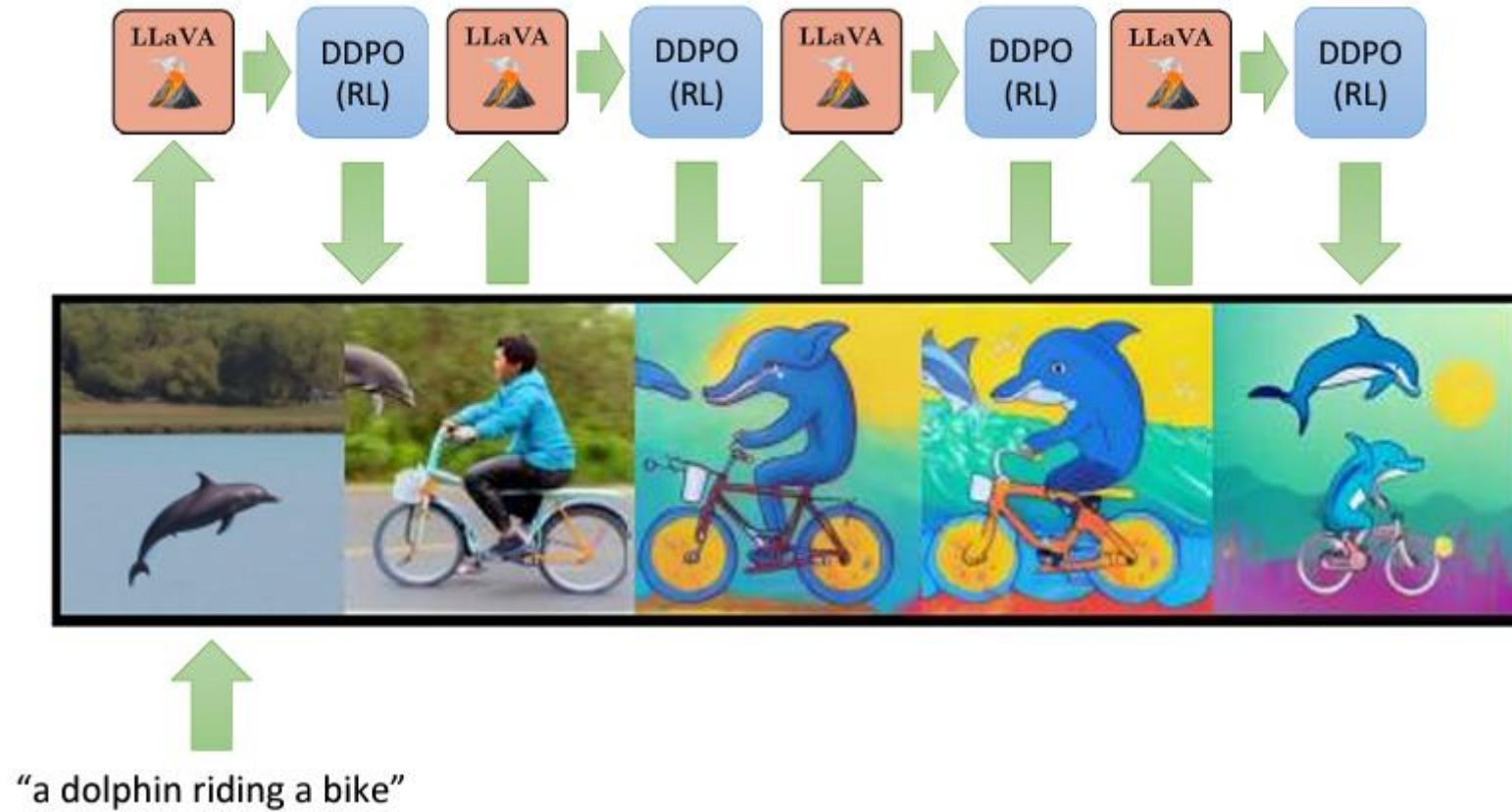
Rufus will take this feedback and use it to update his language model. He will try to say the sentence again, using the new information he received from the human. This time, he might say "I'm a robot."

The human will listen again and give Rufus more feedback. This process will continue until Rufus can say sentences that sound natural to a human.

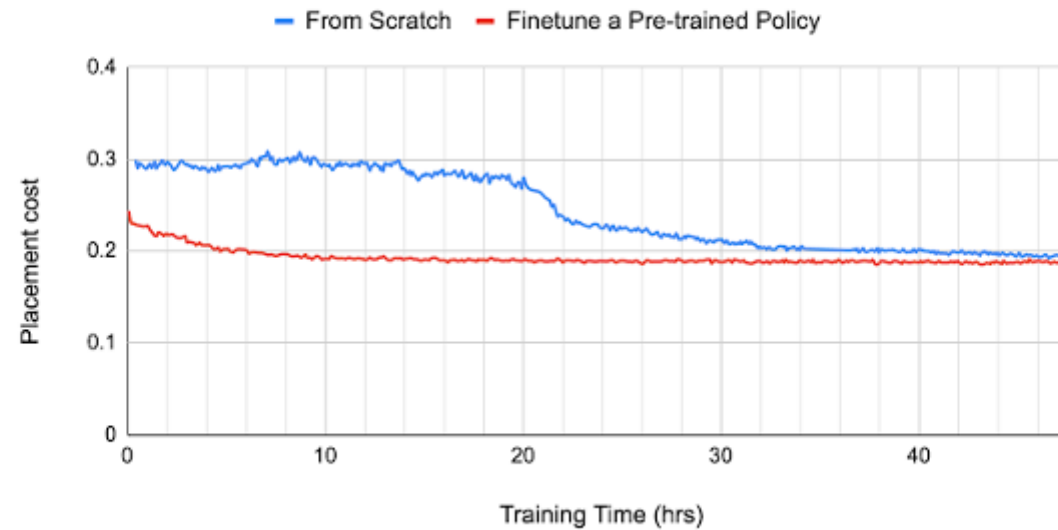
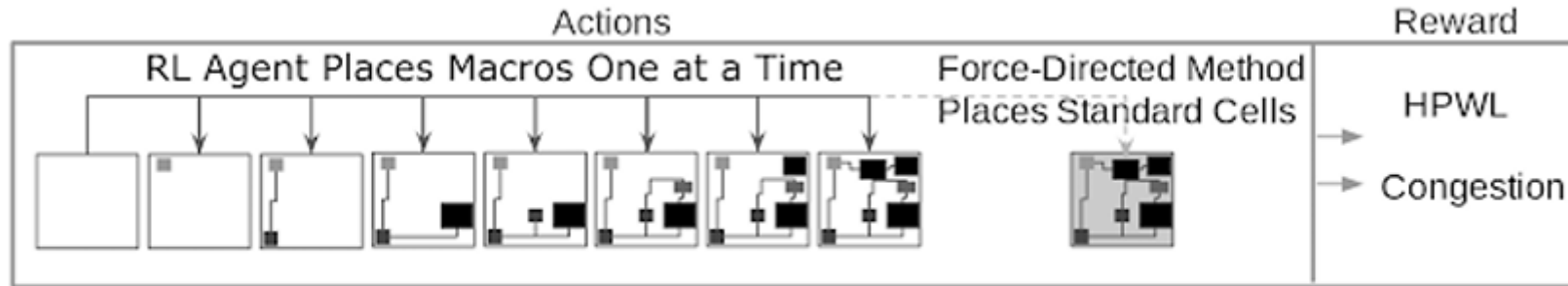
Over time, Rufus will learn how to talk like a human thanks to the feedback he receives from humans. This is how language models can be improved using RL with human feedback.



# Reinforcement learning with image generation



# Reinforcement learning for chip design





# 强化学习技术概览

## 两种人工智能任务类型

### ▣ 预测型任务

- 根据数据预测所需输出 (有监督学习)
- 生成数据实例 (无监督学习)

### ▣ 决策型任务

- 在动态环境中采取行动 (强化学习)
  - 转变到新的状态
  - 获得即时奖励
  - 随着时间的推移最大化累计奖励
  - Learning from interaction in a trial-and-error manner

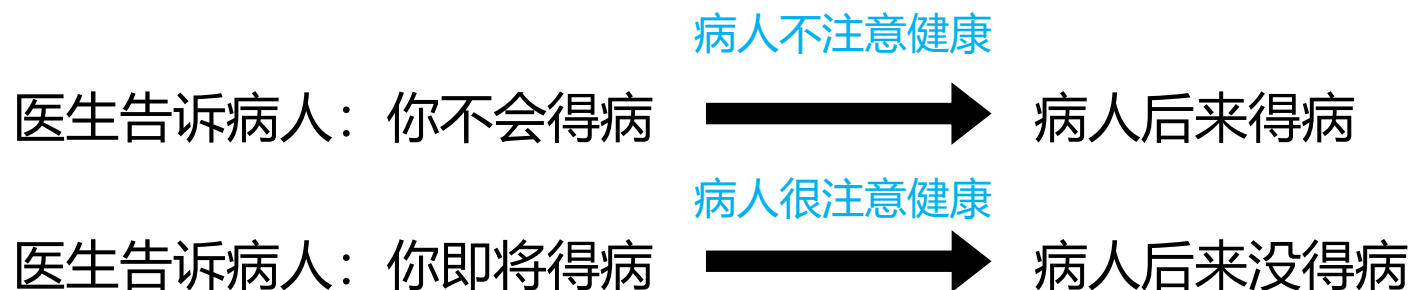


## 决策和预测的不同

- 决策亲自改变世界
  - 医生或者AI直接给病人下达治疗方案
- 预测辅助别人改变世界
  - AI告诉医生病人可能的得病预测，医生综合各方面判断最后给病人下达治疗方案



### 医生预判悖论



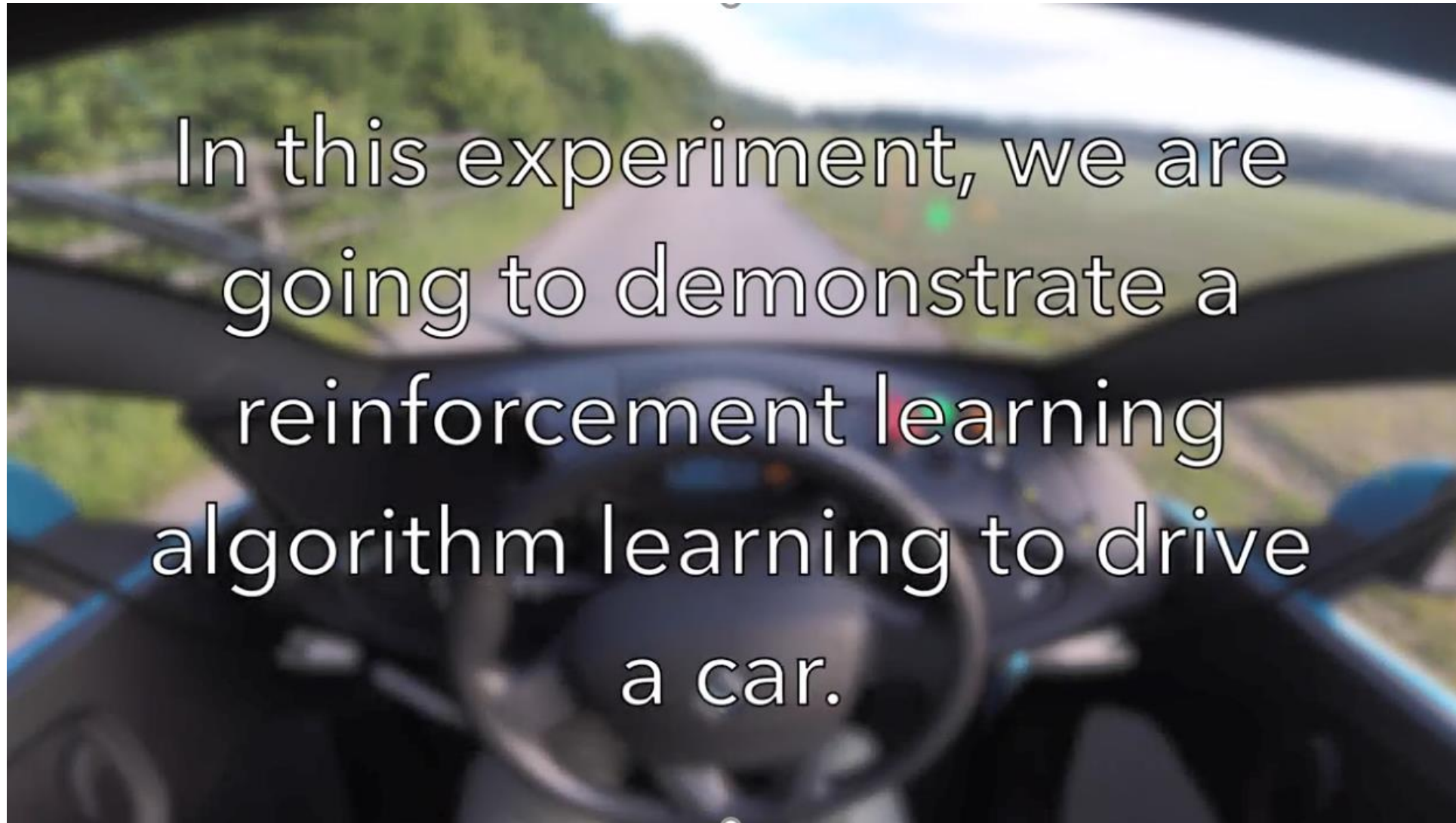
## 序贯决策 (Sequential Decision Making)



**只要是序贯决策问题，就可以用强化学习来解决**



## 强化学习应用案例：无人驾驶小车

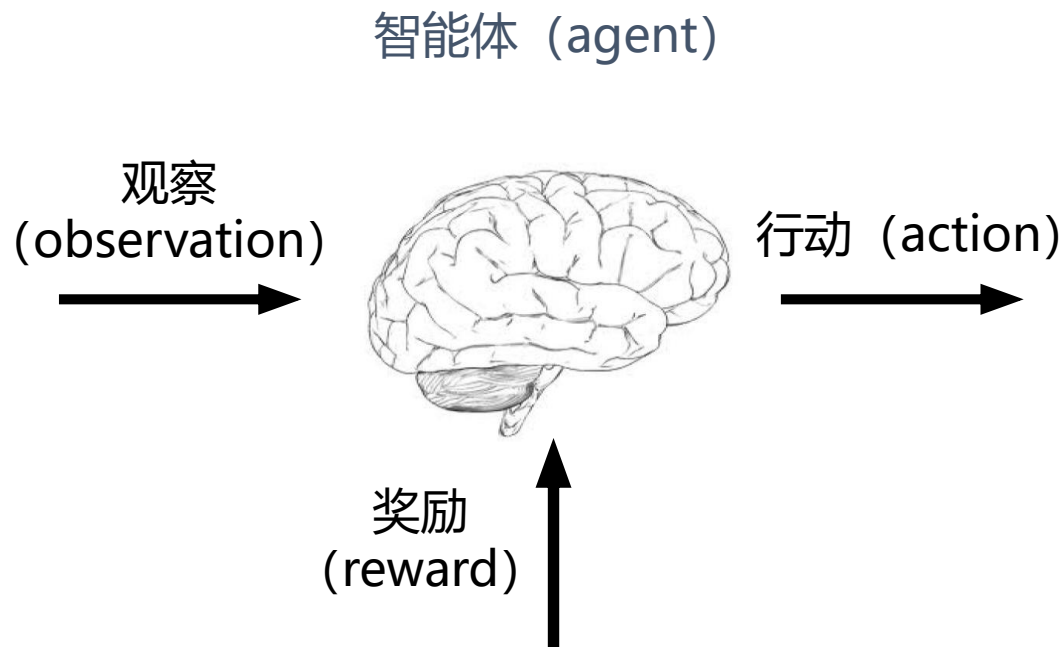


## ■ 主要内容

- 面向决策任务的人工智能
- 强化学习的基础概念和研究前沿
- 强化学习的落地现状与挑战

## 强化学习定义

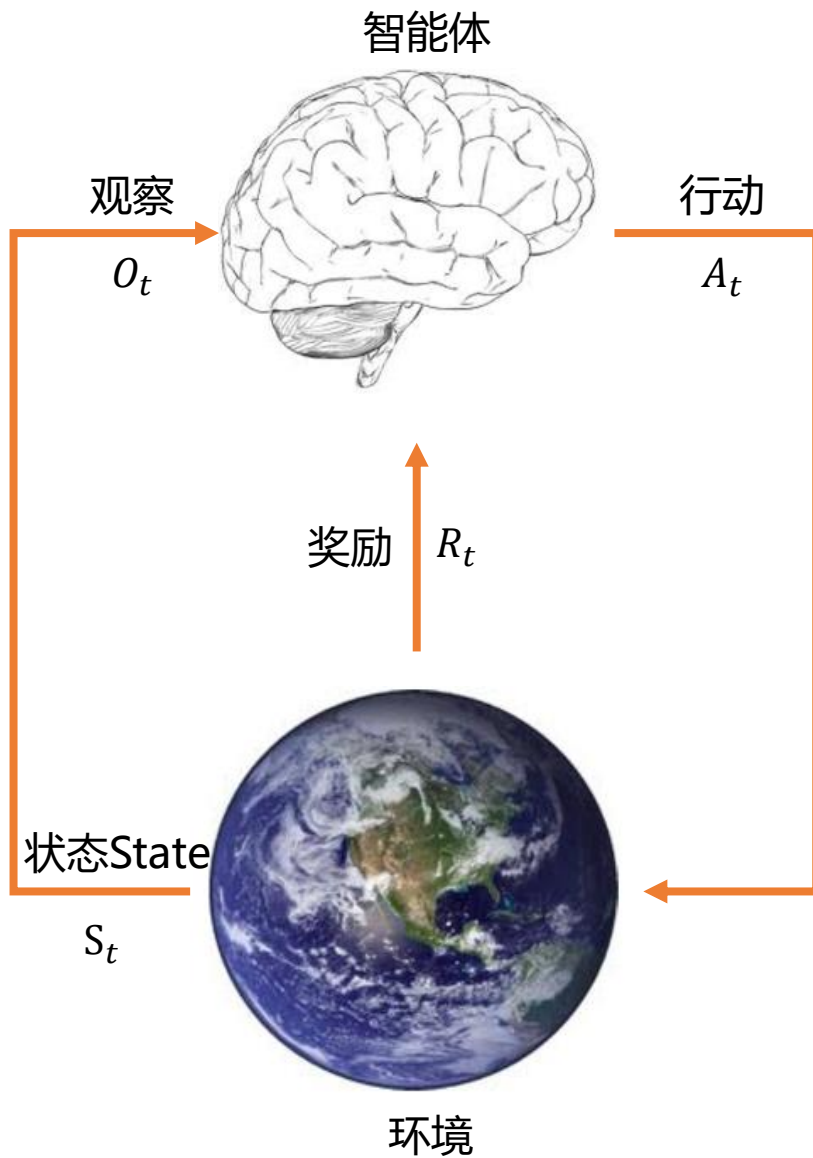
- 通过从交互中学习来实现目标的计算方法



- 三个方面:

- 感知: 在某种程度上感知环境的状态
- 行动: 可以采取行动来影响状态或者达到目标
- 目标: 随着时间推移最大化累积奖励

# 强化学习交互过程



## □ 在每一步 $t$ , 智能体:

- 获得观察  $O_t$
- 获得奖励  $R_t$
- 执行行动  $A_t$

## □ 环境:

- 获得行动  $A_t$
- 给出观察  $O_{t+1}$
- 给出奖励  $R_{t+1}$

## □ $t$ 在环境这一步增加

# 在与动态环境的交互中学习

有监督、无监督学习

Model ←



Fixed Data

强化学习

Agent ↔



Dynamic Environment

Agent不同，交互出的数据也不同！

# 强化学习系统要素

□ 历史 (History) 是观察、行动和奖励的序列

$$H_t = O_1, R_1, A_1, O_2, R_2, A_2, \dots, O_{t-1}, R_{t-1}, A_{t-1}, O_t, R_t$$

- 即, 一直到时间  $t$  为止的所有可观测变量
- 根据这个历史可以决定接下来会发生什么
  - 智能体选择行动
  - 环境选择观察和奖励



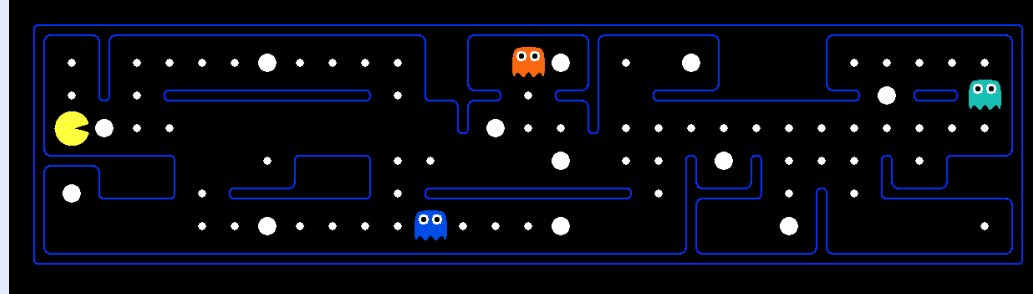
□ 状态 (state) 是描述环境的所有信息, 可用于确定接下来会发生的事情 (行动、观察、奖励)

- 状态是关于历史的函数

$$S_t = f(H_t)$$

# What is a state?

The **world state** includes every last detail of the environment



A **search state** keeps only the details needed for planning (abstraction)

- **Problem: Pathing**

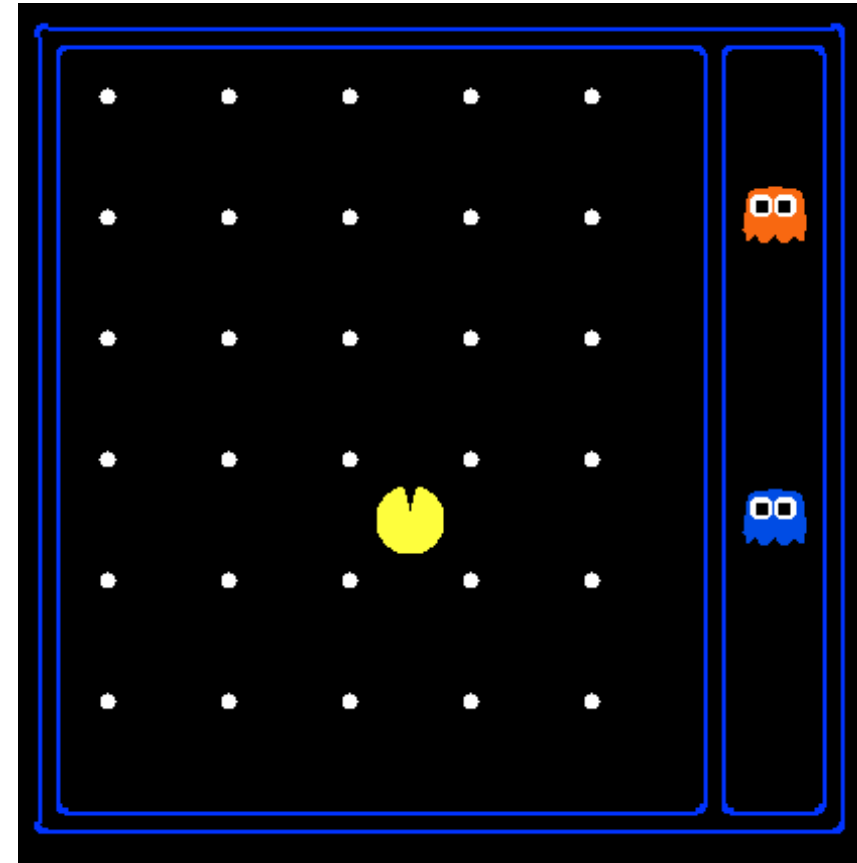
- States:  $(x,y)$  location
- Actions: NSEW
- Successor: update location only
- Goal test: is  $(x,y)=END$

- **Problem: Eat-All-Dots**

- States:  $\{(x,y), \text{dot booleans}\}$
- Actions: NSEW
- Successor: update location and possibly a dot boolean
- Goal test: dots all false

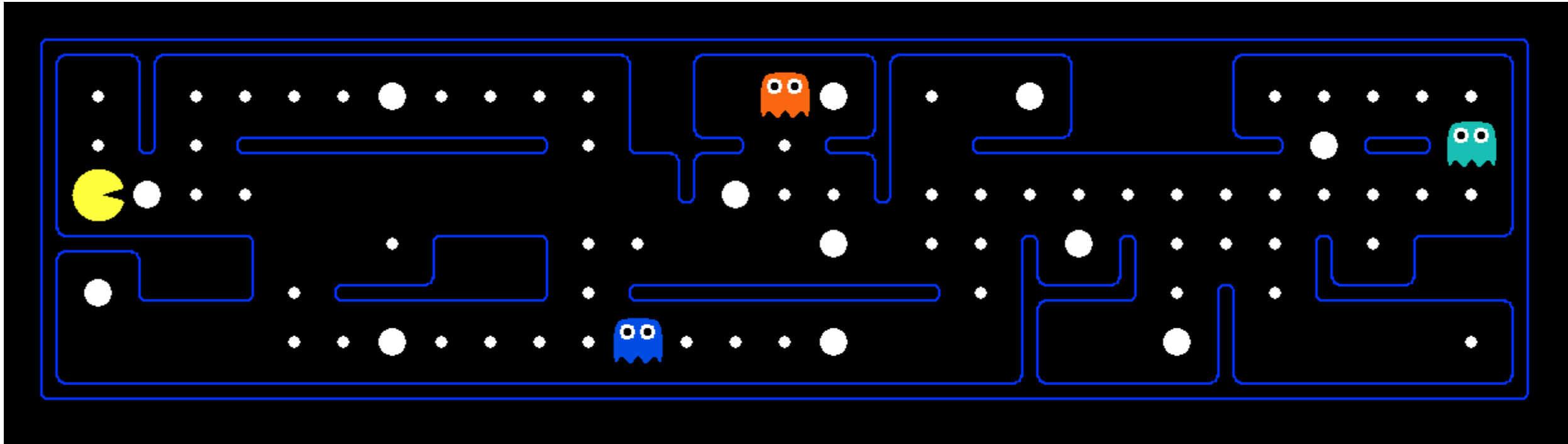
# State Space Sizes?

- World state:
  - Agent positions: 120
  - Food count: 30
  - Ghost positions: 12
  - Agent facing: NSEW
- How many
  - World states?  
 $120 \times (2^{30}) \times (12^2) \times 4$
  - States for pathing?  
120
  - States for eat-all-dots?  
 $120 \times (2^{30})$





# Safe Passage



- Problem: eat all dots while keeping the ghosts perma-scared
- What does the state space have to specify?
  - (agent position, dot booleans, power pellet booleans, remaining scared time)

# 强化学习系统要素

## 策略 (Policy) 是学习智能体在特定时间的行为方式

- $\pi: \mathcal{S} \rightarrow \Delta(\mathcal{A})$
- 确定性策略 (Deterministic Policy)

$$a = \pi(s)$$

- 随机策略 (Stochastic Policy)

$$\pi(a|s) = P(A_t = a | S_t = s)$$

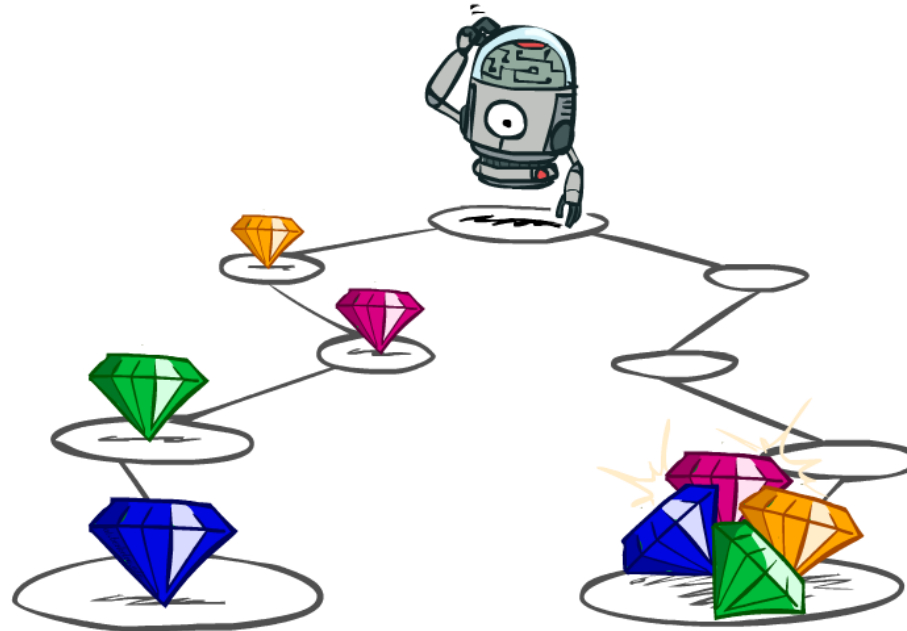
## 奖励 (Reward) $R(s, a, s')$

- 一个定义强化学习目标的标量
- 能立即感知到什么是“好”的



# Utilities of Sequences

- What preferences should an agent have over reward sequences?
- More or less?  
[1, 2, 2] or [2, 3, 4]
- Now or later?  
[0, 0, 1] or [1, 0, 0]



# Utilities of Sequences: Discounting

- It's reasonable to maximize the sum of rewards
- It's also reasonable to prefer rewards now to rewards later
- One solution: values of rewards decay exponentially



1

Worth Now



$\gamma$

Worth Next Step



$\gamma^2$

Worth In Two Steps

# Utilities of Sequences: Stationary Preferences

- Theorem: if we assume **stationary preferences**:

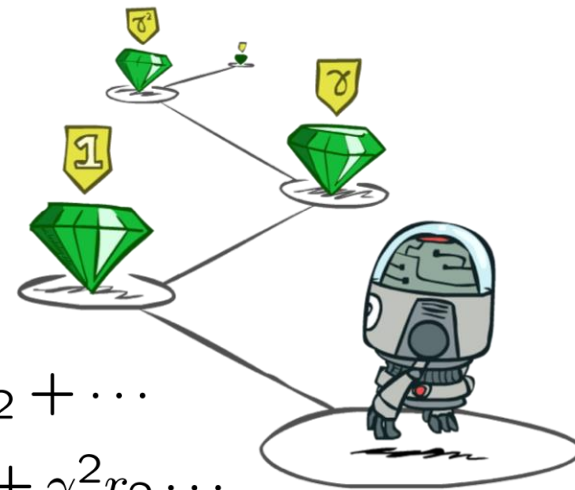
$$[a_1, a_2, \dots] \succ [b_1, b_2, \dots]$$



$$[r, a_1, a_2, \dots] \succ [r, b_1, b_2, \dots]$$

- Then: there are only two ways to define utilities

- Additive utility:  $U([r_0, r_1, r_2, \dots]) = r_0 + r_1 + r_2 + \dots$
- Discounted utility:  $U([r_0, r_1, r_2, \dots]) = r_0 + \gamma r_1 + \gamma^2 r_2 \dots$



# 强化学习系统要素

## ▣ 价值函数 (Value Function)

- 状态价值是一个标量，用于定义对于长期来说什么是“好”的
- 价值函数是对于未来累积奖励的预测
  - 用于评估在给定的策略下，状态的好坏



$$\begin{aligned} Q_{\pi}(s, a) &= \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s, A_t = a] \\ &= \mathbb{E}_{\pi}[R_{t+1} + \gamma Q_{\pi}(s', a') | S_t = s, A_t = a] \end{aligned}$$

# 强化学习系统要素

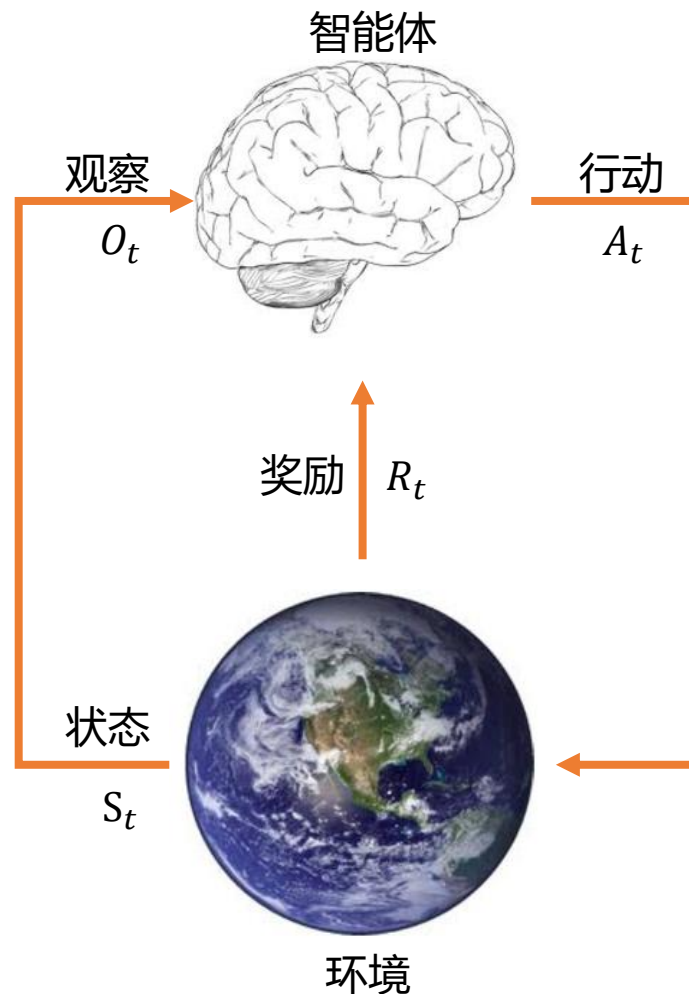
## □ 环境的模型 (Model) 用于模拟环境的行为

- 预测下一个状态

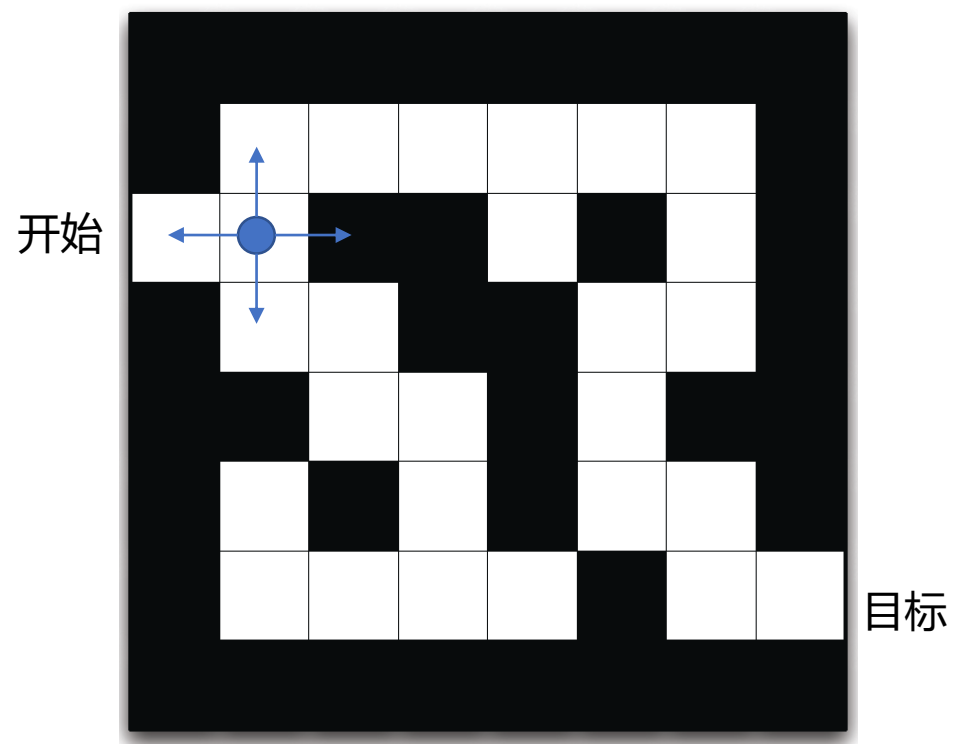
$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

- 预测下一个 (immediate 立即) 奖励

$$\mathcal{R}_s^a = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$



## 举例：迷宫

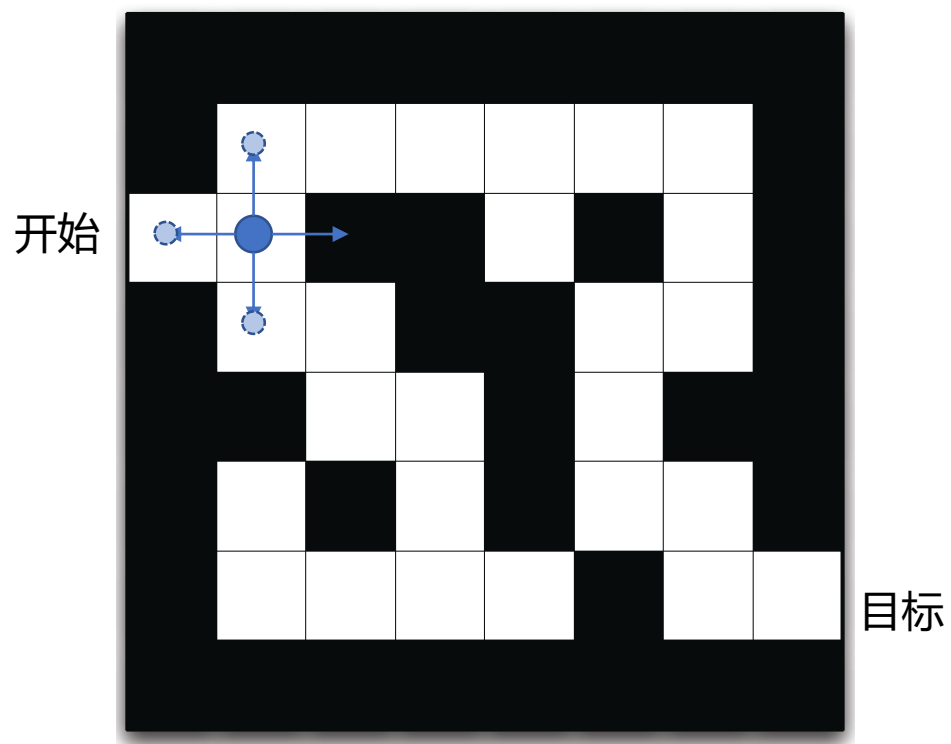


□ 状态：智能体的位置

□ 行动：N,E,S,W

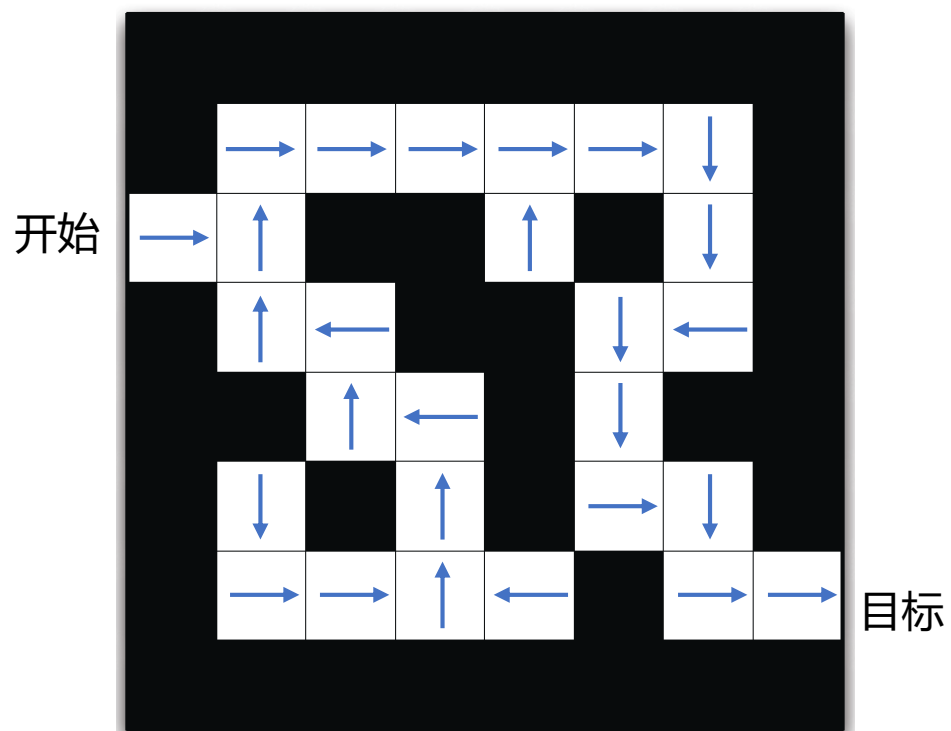


## 举例：迷宫



- 状态：智能体的位置
- 行动：N,E,S,W
- 状态转移：根据行动方向朝下一格移动
  - 如果行动的方向是墙则不动

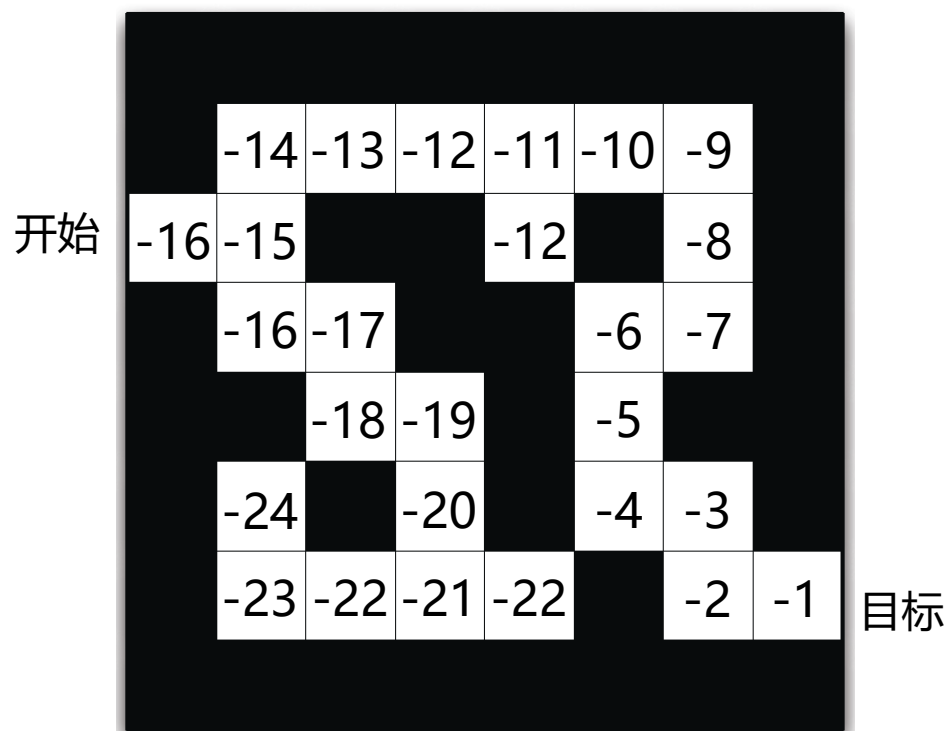
## 举例：迷宫



- 状态：智能体的位置
- 行动：N,E,S,W
- 状态转移：根据行动方向朝下一格移动
- 奖励：每一步为-1

- 给定一个上图所示的策略
  - 箭头表示每一个状态 $s$ 下的策略 $\pi(s)$

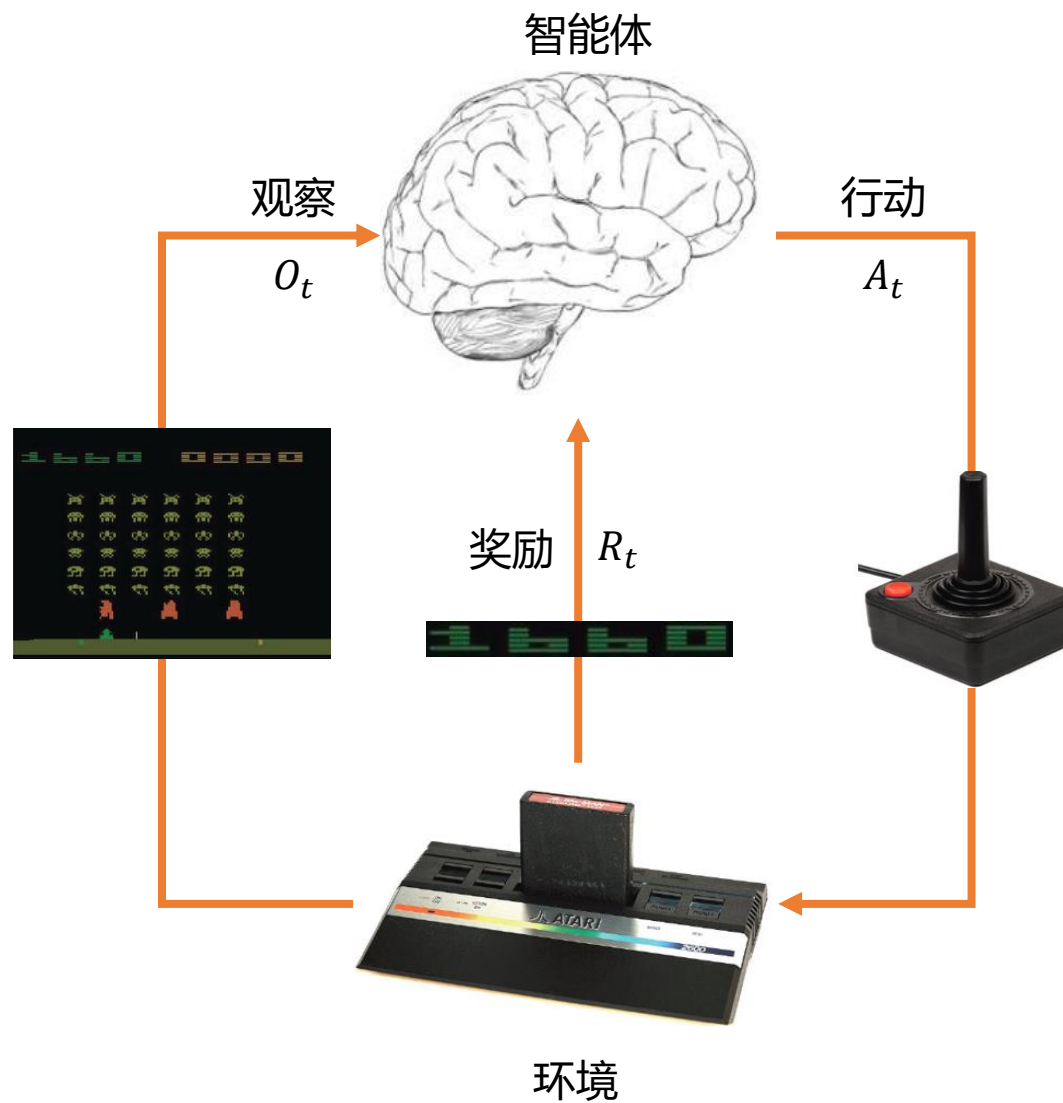
## 举例：迷宫



- 状态：智能体的位置
- 行动：N,E,S,W
- 状态转移：根据行动方向朝下一格移动
- 奖励：每一步为-1

- 数字表示每一个状态 $s$ 下的状态价值 $v_{\pi}(s)$

## 举例：Atari游戏



- 游戏规则未知
- 从交互游戏中进行学习
- 在操纵杆上选择行动并查看分数和像素画面

## 强化学习的方法分类

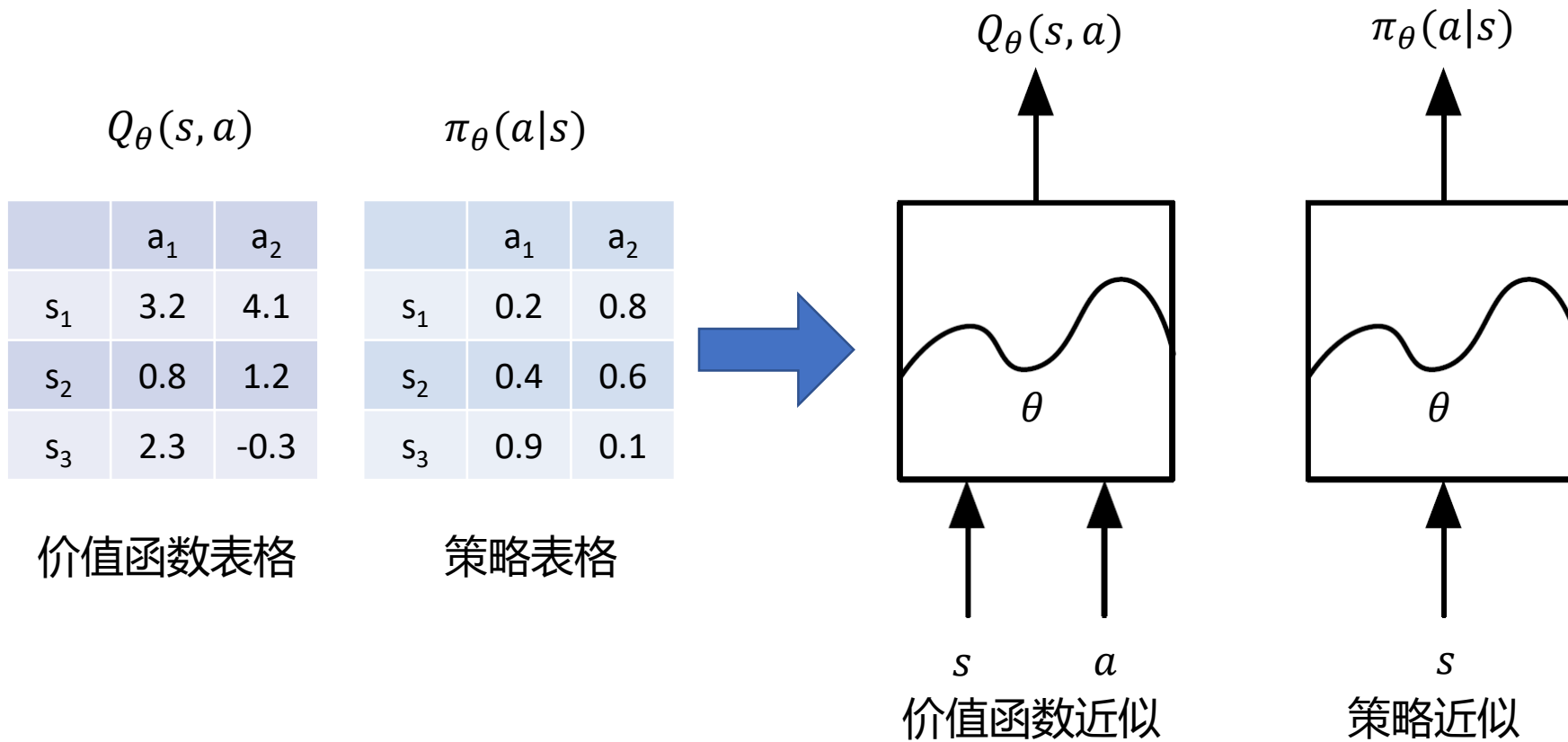
- 基于价值：知道什么是好的什么是坏的
  - 没有策略（隐含）
  - 价值函数

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot); s' \sim \mathcal{E}} \left[ \left( r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i) \right) \nabla_{\theta_i} Q(s, a; \theta_i) \right]$$

- 基于策略：知道怎么行动
  - 策略
  - 没有价值函数
- Actor-Critic：学生听老师的
  - 策略
  - 价值函数



# 价值和策略近似



- 假如我们直接使用深度神经网络建立这些近似函数呢?
- 深度强化学习!



## 深度强化学习的崛起

- 2012年AlexNet在ImageNet比赛中大幅度领先对手获得冠军
- 2013年12月, 第一篇深度强化学习论文出自NIPS 2013 Reinforcement Learning Workshop

---

### **Playing Atari with Deep Reinforcement Learning**

---

**Volodymyr Mnih   Koray Kavukcuoglu   David Silver   Alex Graves   Ioannis Antonoglou**

**Daan Wierstra   Martin Riedmiller**

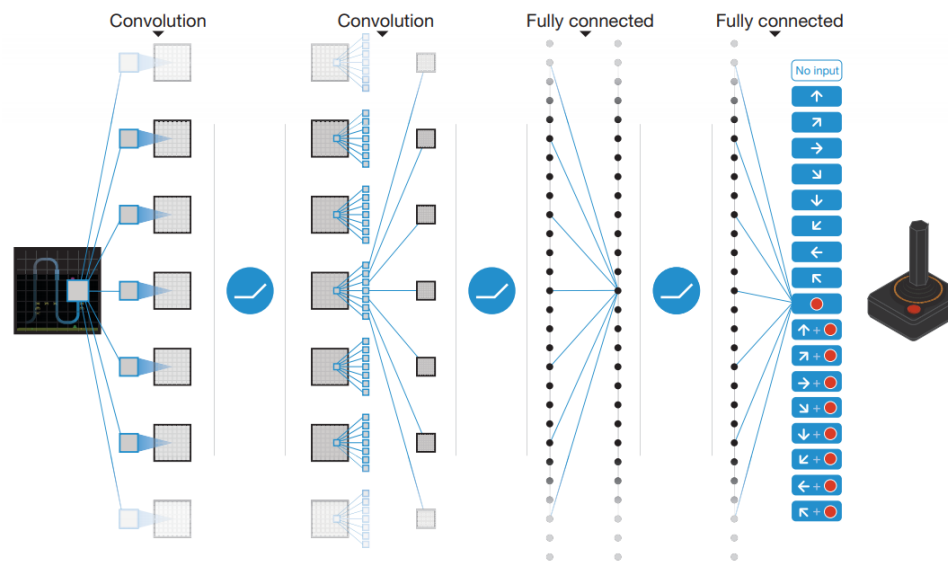
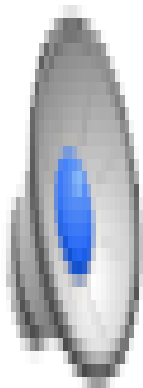
DeepMind Technologies

`{vlad, koray, david, alex.graves, ioannis, daan, martin.riedmiller} @ deepmind.com`

# 深度强化学习

## 深度强化学习

- 利用深度神经网络进行价值函数和策略近似
- 从而使强化学习算法能够以端到端的方式解决复杂问题

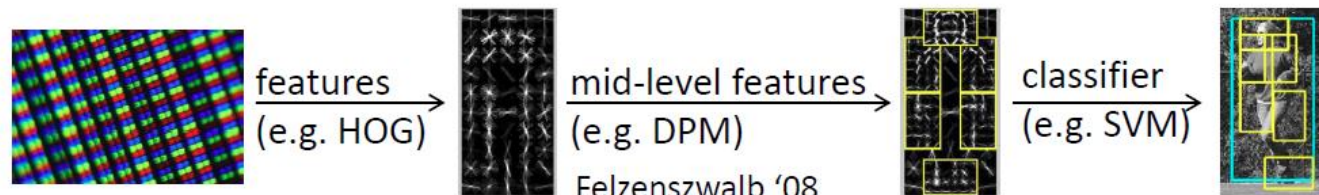


Q函数的参数通过神经网络反向传播学习

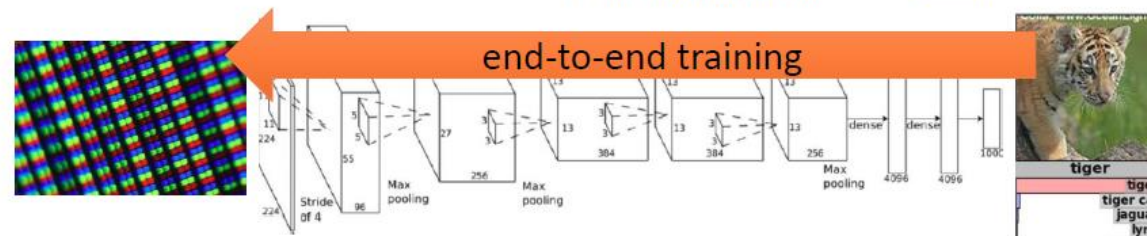
$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot); s' \sim \mathcal{E}} \left[ \left( r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i) \right) \nabla_{\theta_i} Q(s, a; \theta_i) \right]$$

# 端到端强化学习

标准 (传统)  
计算机视觉



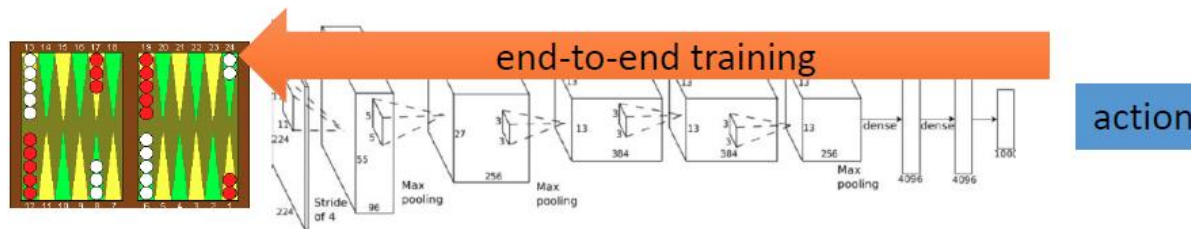
深度学习



标准 (传统)  
强化学习

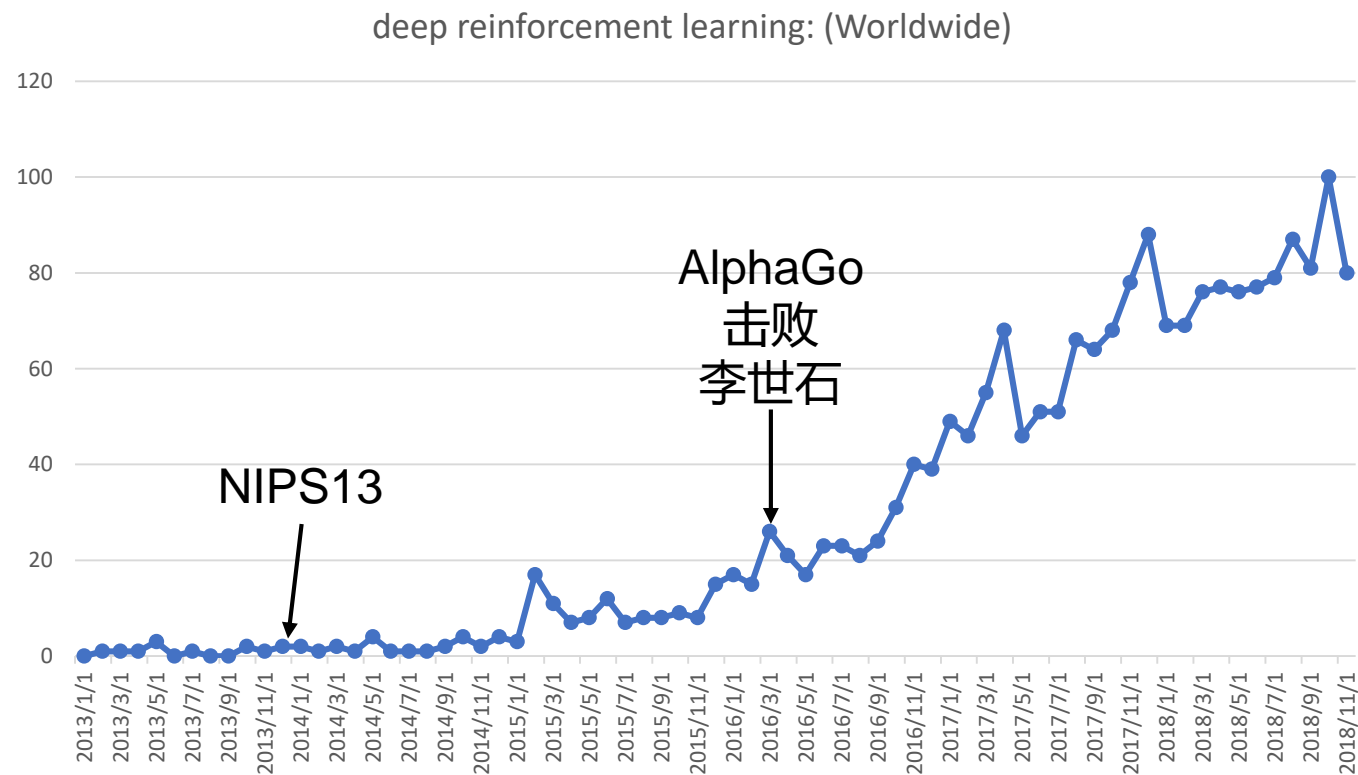


深度强化学习



- 深度强化学习使强化学习算法能够以端到端的方式解决复杂问题
- 从一项实验室学术技术变成可以产生GDP的实际技术

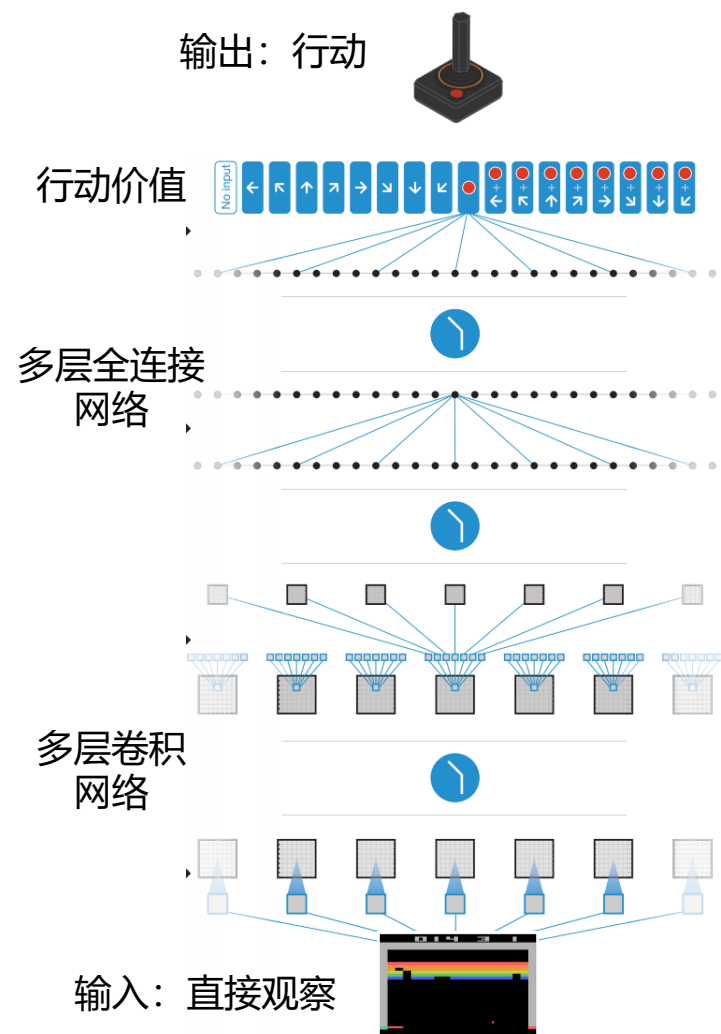
# 深度强化学习趋势



- Google搜索中词条“深度强化学习 (deep reinforcement learning)”的趋势

# 深度强化学习带来的关键变化

- 将深度学习 (DL) 和强化学习 (RL) 结合在一起会发生什么?
  - 价值函数和策略变成了深度神经网络
  - 相当高维的参数空间
  - 难以稳定地训练
  - 容易过拟合
  - 需要大量的数据
  - 需要高性能计算
  - CPU (用于收集经验数据) 和GPU (用于训练神经网络) 之间的平衡
  - ...
- 这些新的问题促进着深度强化学习算法的创新



# 深度Q网络 (DQN)

- Q 学习算法学习一个由  $\theta$  作为参数的函数  $Q_{\theta}(s, a)$ 
  - 更新方程  $Q_{\theta}(s_t, a_t) \leftarrow (1 - \alpha)Q_{\theta}(s_t, a_t) + \alpha \left( r_t + \gamma \max_{a'} Q_{\theta}(s_{t+1}, a') \right)$   
 $\leftarrow Q_{\theta}(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q_{\theta}(s_{t+1}, a') - Q_{\theta}(s_t, a_t))$

## 直观想法

- 使用神经网络来逼近  $Q_{\theta}(s, a)$ , 面对算法不稳定问题
  - 连续采样得到的  $\{(s_t, a_t, s_{t+1}, r_t)\}$  不满足独立分布
  - $\{(s_t, a_t, s_{t+1}, r_t)\}$  为状态-动作-下一状态-回报输入
  - $Q_{\theta}(s, a)$  的频繁更新

## 解决办法

- 经验回放: 均匀采样和优先经验回放
- 使用双网络结构: 评估网络 (evaluation network) 和目标网络 (target network)

“Human-Level Control Through Deep Reinforcement Learning”, Mnih, Kavukcuoglu, Silver et al. (2015)

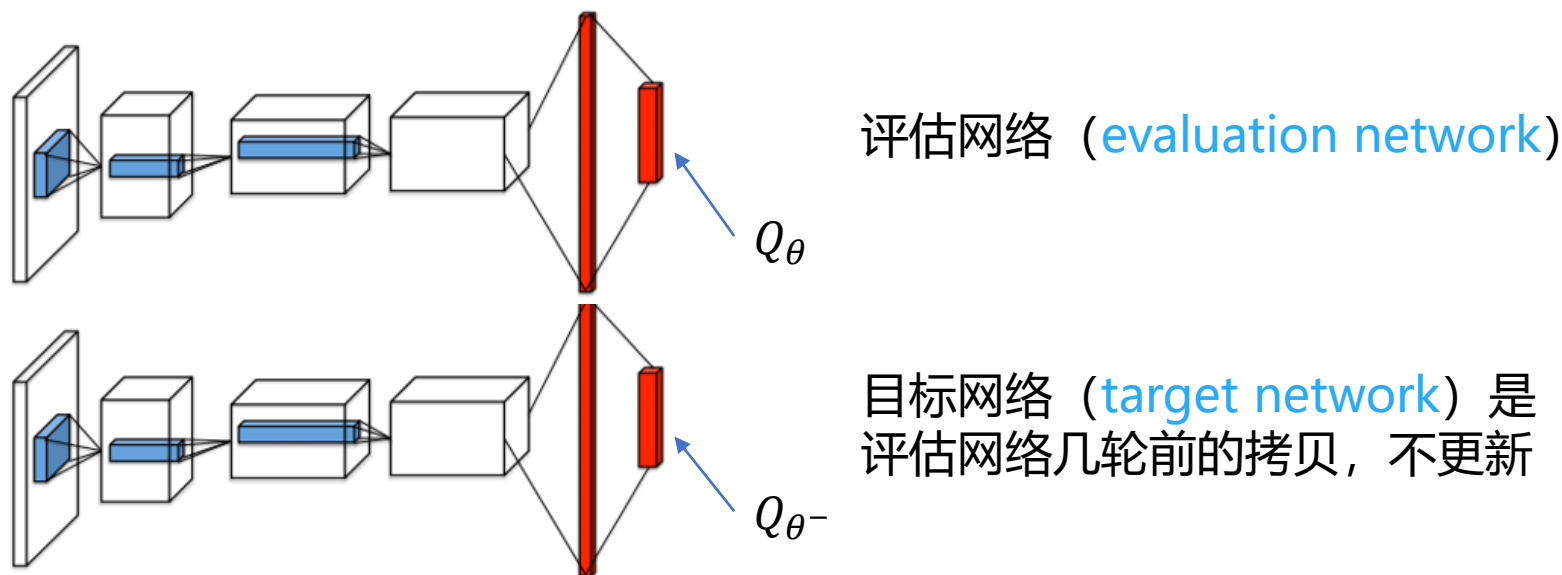


# 目标网络

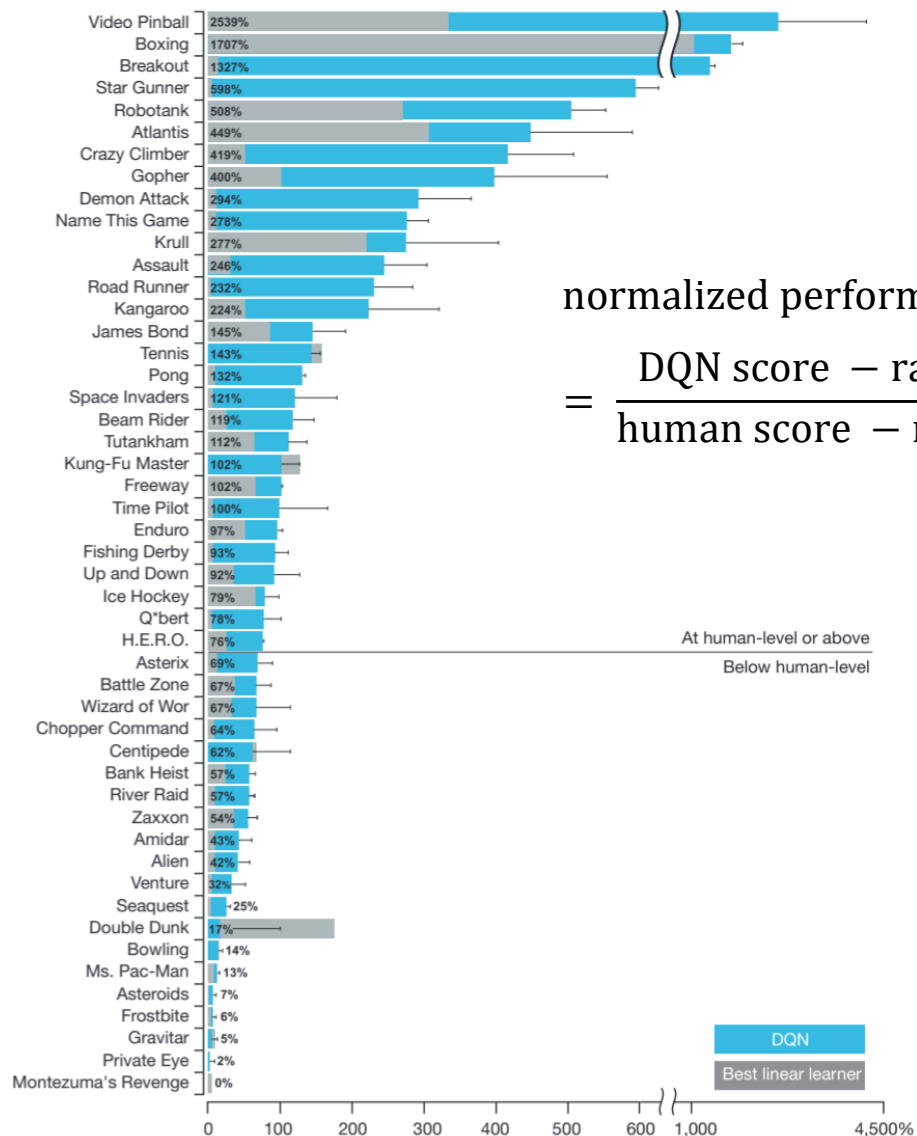
## 目标网络 $Q_{\theta^-}(s, a)$

- 使用较旧的参数, 记为  $\theta^-$ , 每隔  $C$  步和训练网络的参数同步一次。
- 第  $i$  次迭代的损失函数为

$$L_i(\theta_i) = \mathbb{E}_{s_t, a_t, s_{t+1}, r_t, p_t \sim D} \left[ \frac{1}{2} \omega_t (r_t + \underbrace{\gamma \max_{a'} Q_{\theta_i^-}(s_{t+1}, a')}_{\text{target}} - Q_{\theta_i}(s_t, a_t))^2 \right]$$



# 在 Atari 环境中的实验结果



normalized performance

$$= \frac{\text{DQN score} - \text{random play score}}{\text{human score} - \text{random play score}}$$

The performance of DQN is normalized with respect to a professional human games tester (that is, 100% level)

# 深度强化学习的研究前沿



## 基于模拟模型的强化学习

- 模拟器的无比重要性



## 目标策动的层次化强化学习

- 长程任务的中间目标是桥梁的基石



## 模仿学习

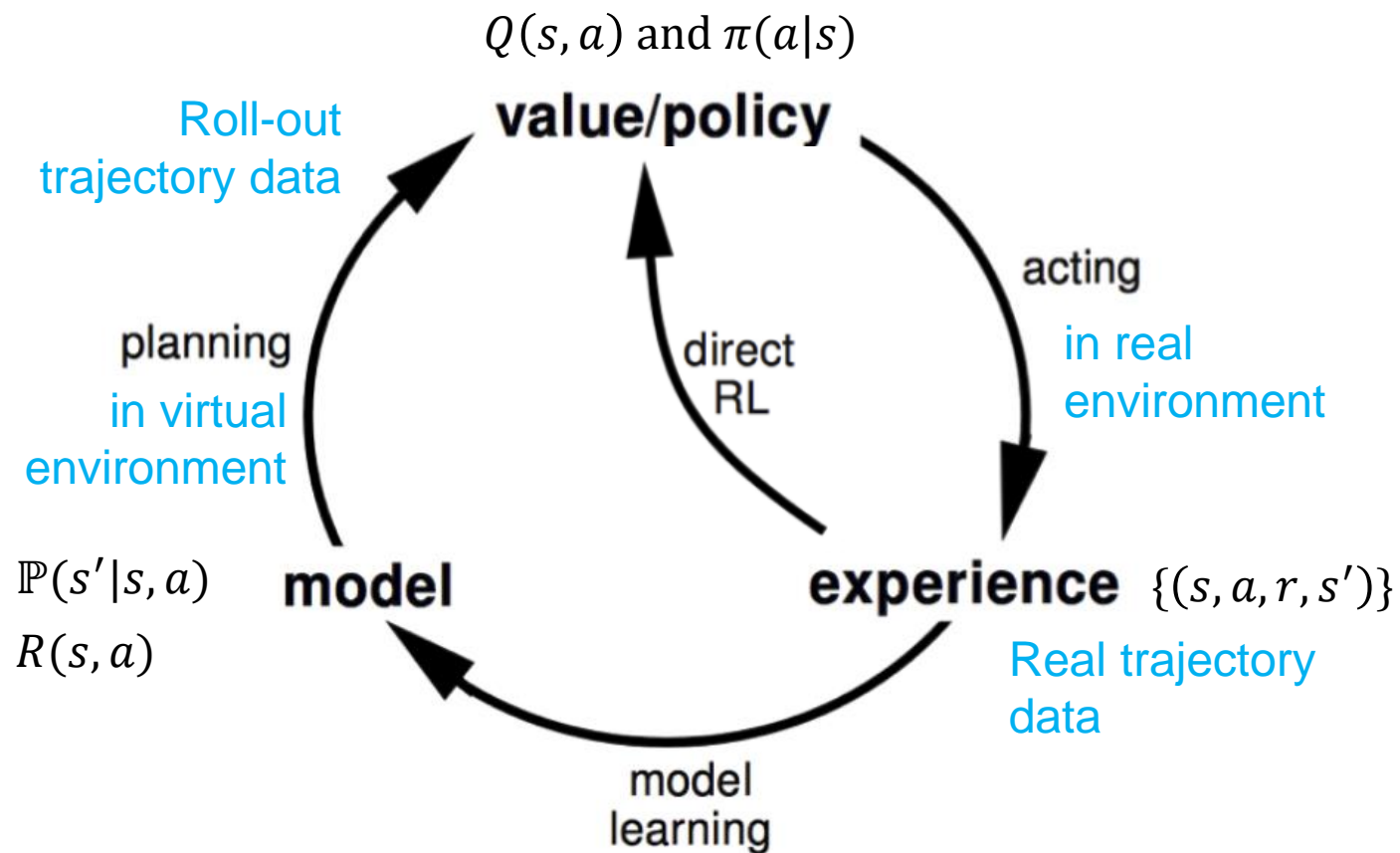
- 无奖励信号下跟随专家做策略学习



## 多智能体强化学习

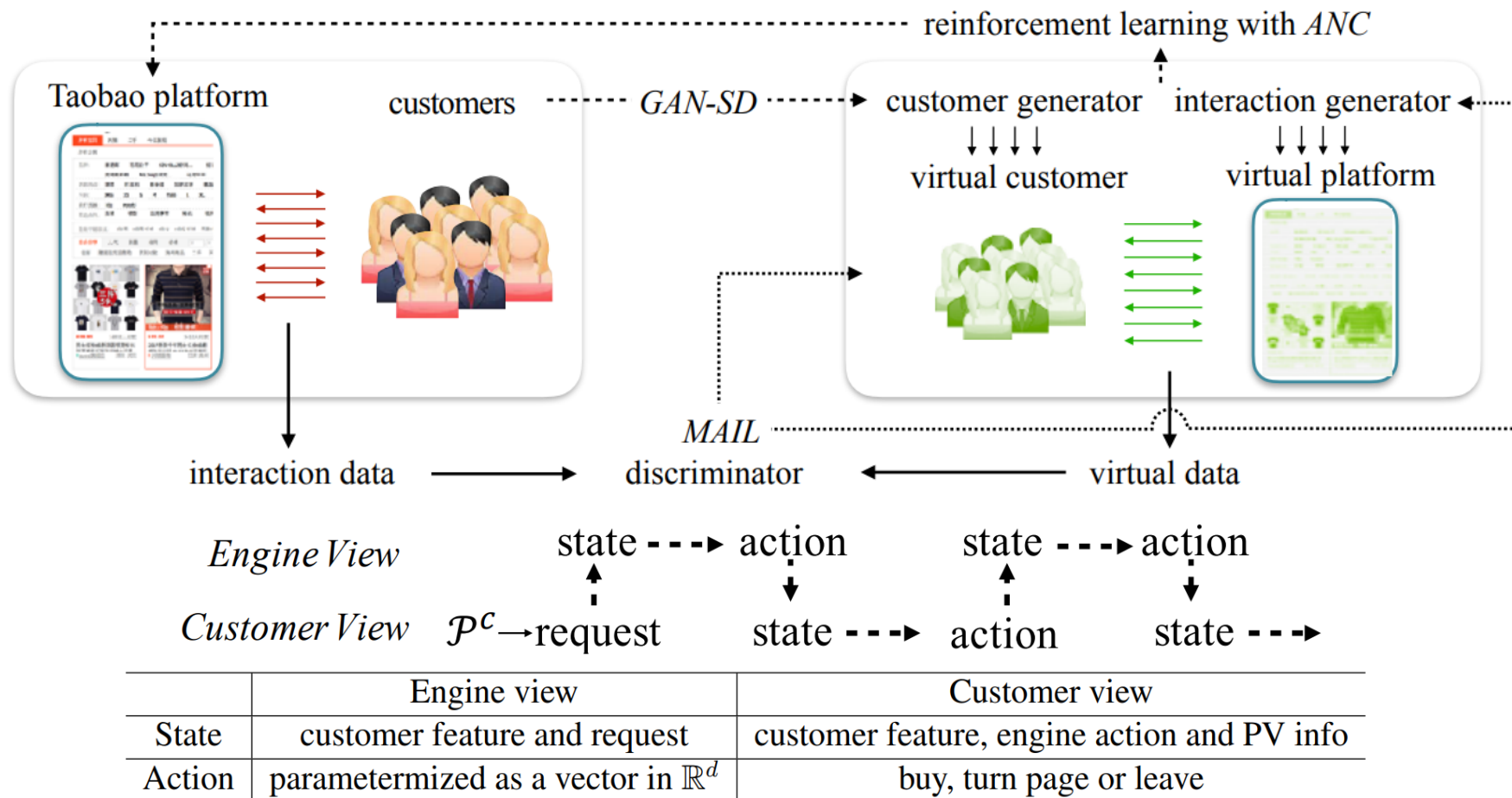
- 分散式、去中心化的人工智能

## 基于模拟模型的强化学习(Model-based RL)



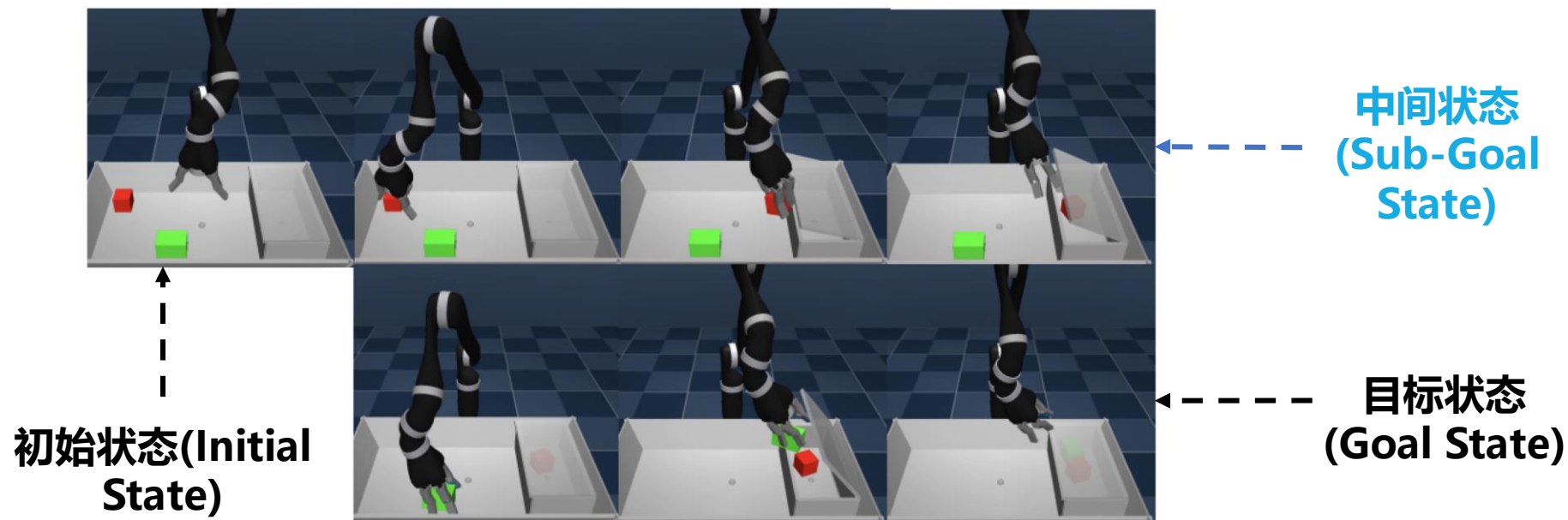
- 建立环境模拟器，在模拟器中训练强化学习策略，减少对真实环境的影响，也可以生成更多特定场景数据

# 基于模拟模型的强化学习(Model-based RL)



- 建立用户在电商平台行为模拟器，模拟不同商品推荐策略下用户的浏览、点击、购买等行为，进而优化电商推荐策略

## 目标策动的强化学习(Goal-oriented RL)



**长  
期  
任  
务  
挑  
战  
(Long-Horizon)**

- ❑ 累计建模误差(Compounding Model Error).
- ❑ 稀疏反馈(Sparse Cost).

生成中间状态，将长期任务分割成多个简单的短期任务

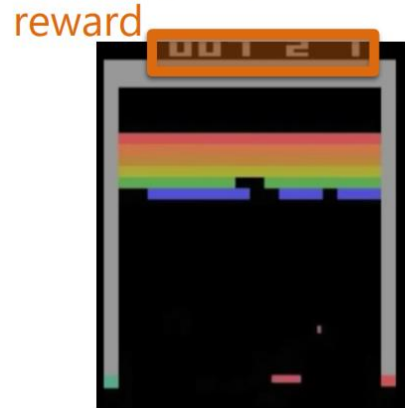


# 目标策动的强化学习(Goal-oriented RL)



# 模仿学习(Imitation Learning)

Computer Games



Mnih et al. '15

Real World Scenarios

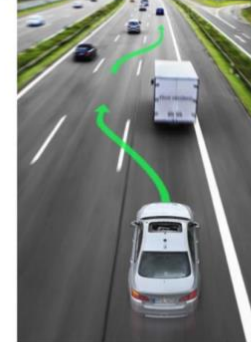
robotics



dialog

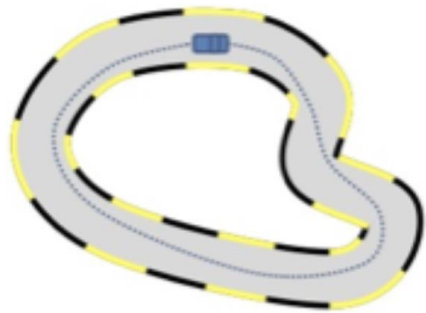


autonomous driving



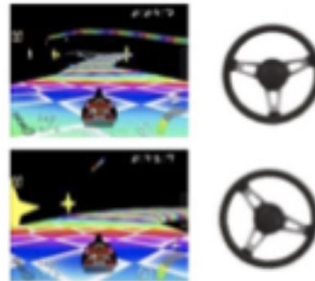
what is the **reward**?  
often use a proxy

Expert Demonstrations



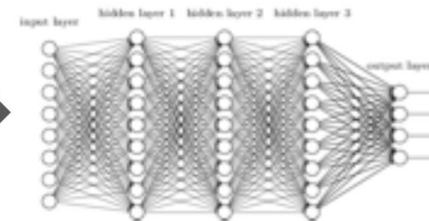
Agent Experience

$(s, a)$  pairs



$(s, a, r, s', a')$  tuples

Imitation Learning



Reinforcement Learning

## 模仿学习(Imitation Learning)



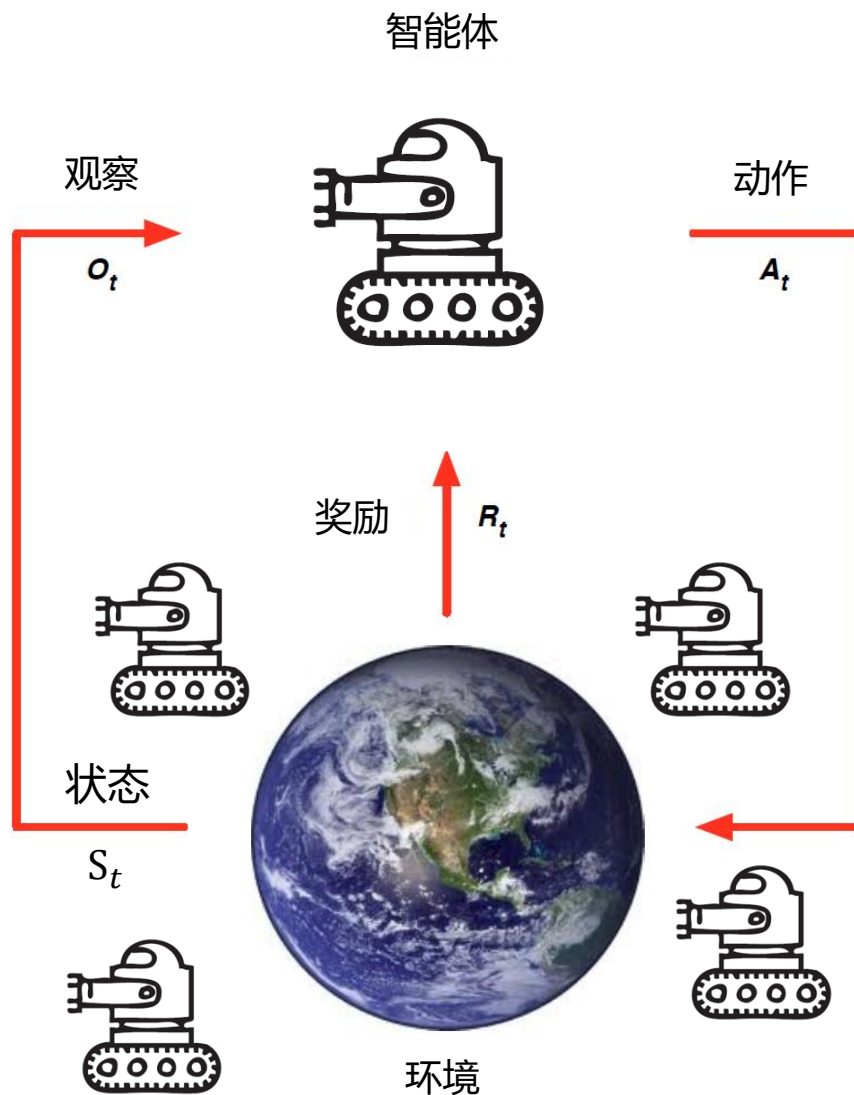
Waymo has made simulation one of the pillars of its autonomous vehicle development program. But **Latent Logic** could help Waymo make its simulation more realistic by using a form of machine learning called **imitation learning**.

**Imitation learning** models human behavior of motorists, cyclists and pedestrians. The idea is that by modeling the mistakes and imperfect driving of humans, the simulation will become more realistic and theoretically improve Waymo's behavior prediction and planning.

# 多智能体强化学习(Multi-agent RL)

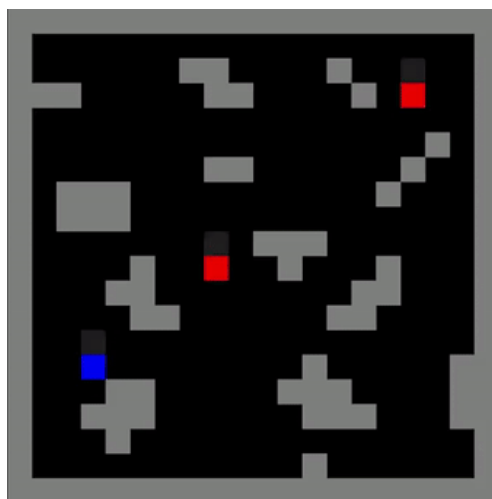
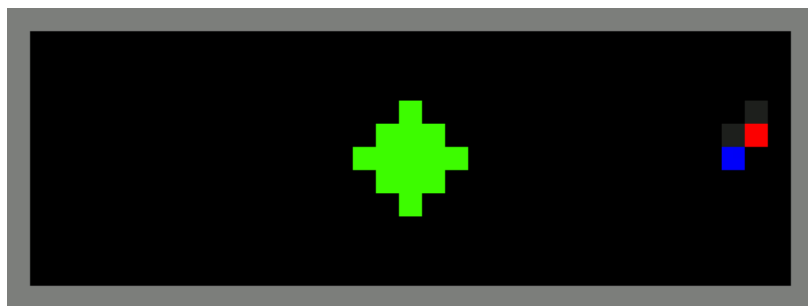
在与环境的交互过程中学习

- 环境包含有不断进行学习和更新的其他智能体
- 在任何一个智能体的视角下，环境是**非稳态的 (non-stationary)**
  - 环境迁移的分布会发生改变

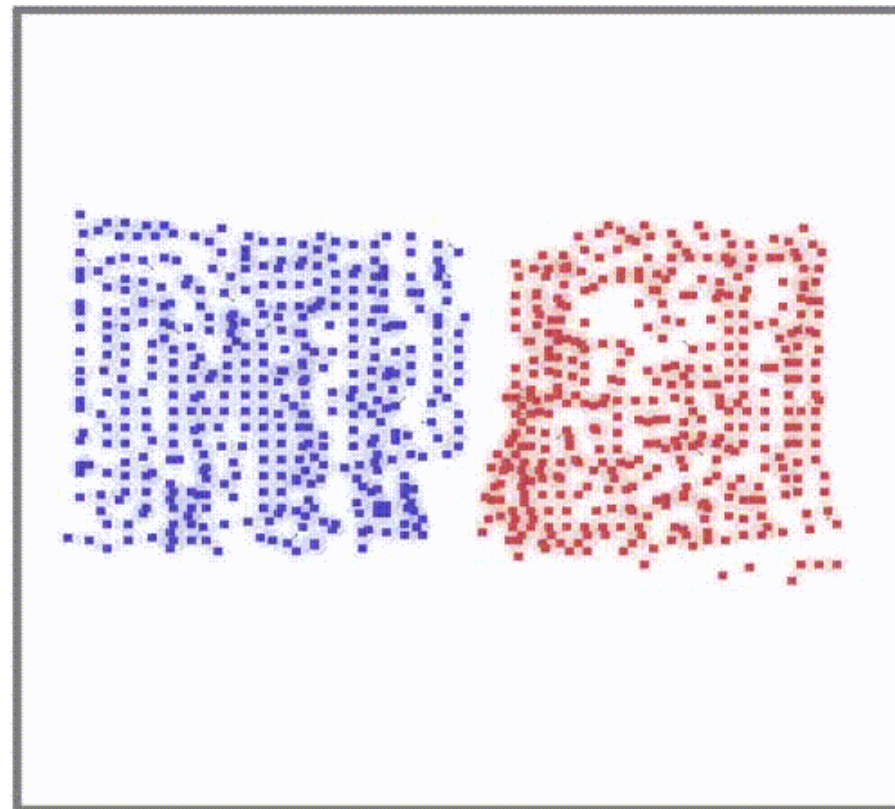




## 多智能体强化学习(Multi-agent RL)



双智能体对抗与合作



大规模智能体战斗模拟

## 强化学习的落地场景

- 无人驾驶
- 游戏AI
- 交通灯调度
- 网约车派单
- 组合优化
- 推荐搜索系统
- 数据中心节能优化
- 对话系统
- 机器人控制
- 路由选路
- 工业互联网场景
- ...



# 总结强化学习技术与落地挑战

## 强化学习做什么

- ▣ 序列型决策任务
- ▣ 让AI做完一切事情，而不仅仅是一个辅助的角色

## 强化学习的技术发展

- ▣ 2013年12月的NIPS workshop论文开启了深度强化学习时代
- ▣ 目前深度强化学习方法已经可以解决部分序列决策任务，但距离真正普及还有很长的路要走

## 强化学习的落地挑战

- ▣ 决策权力交给AI，人对AI有更高的要求
- ▣ 强化学习技术人才短缺，决策场景千变万化，并不统一
- ▣ 当前强化学习算法对数据和算力的极大需求





# 探索与利用



# 目录

Contents

**01 探索与利用**

**02 多臂老虎机问题**

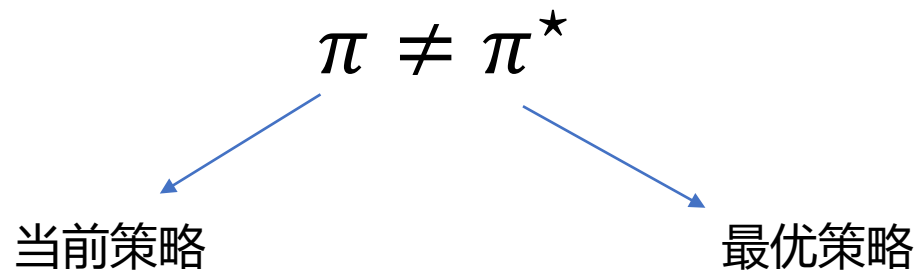


01

探索与利用

## 序列决策任务中的一个基本问题

- 基于目前策略获取已知最优收益还是尝试不同的决策
  - **Exploitation** 执行能够获得已知最优收益的决策
  - **Exploration** 尝试更多可能的决策，不一定会是最优收益



## 序列决策任务中的一个基本问题

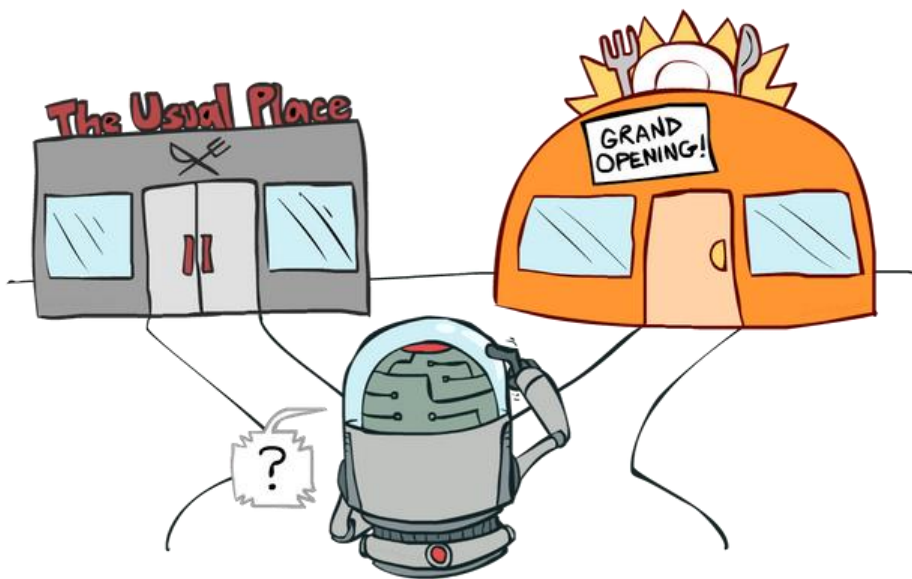
- 基于目前策略获取已知最优收益还是尝试不同的决策
  - **Exploitation** 执行能够获得已知最优收益的决策
  - **Exploration** 尝试更多可能的决策，不一定会是最优收益

$$\mathcal{E}_t = \{\pi_t^i \mid i = 1, \dots, n\} \xrightarrow{\text{探索}} \mathcal{E}_{t+1} = \{\pi_t^i \mid i = 1, \dots, n\} \cup \{\pi_e^j \mid j = 1, \dots, m\}$$

$$\exists V^*(\cdot \mid \pi_t^i \sim \mathcal{E}_t) \leq V^*(\cdot \mid \pi_{t+1}^i \sim \mathcal{E}_{t+1}) \quad \pi_{t+1}^i \sim \{\pi_e^i \mid i = 1, \dots, m\}$$

**探索**：可能发现更好的策略

# 一个例子



图片引用: <https://steemit.com/technology/@mor/machine-learning-series-part-5-exploration-vs-exploitation-dilemma-in-reinforcement-learning>

## 策略探索的一些原则

- 朴素方法 (Naïve Exploration)
  - 添加策略噪声  $\epsilon$ -greedy
- 积极初始化 (Optimistic Initialization)
- 基于不确定性的度量 (Uncertainty Measurement)
  - 尝试具有不确定收益的策略, 可能带来更高的收益
- 概率匹配 (Probability Matching)
  - 基于概率选择最佳策略
- 状态搜索 (State Searching)
  - 探索后续状态可能带来更高收益的策略



02

# 多臂老虎机

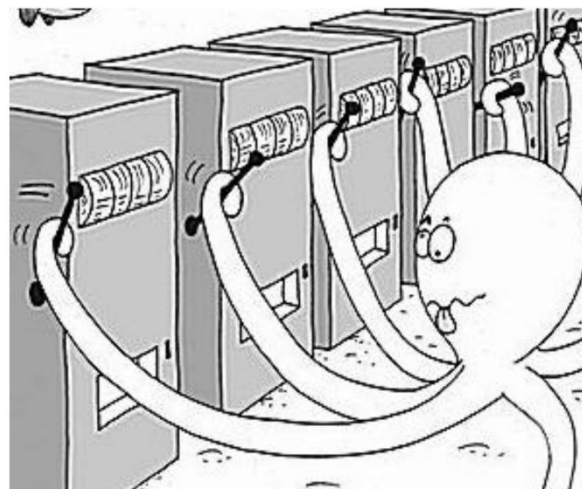
# 多臂老虎机

- 多臂老虎机 (multi-armed bandit) 问题的形式化描述

动作集合:  $a^i \in \mathcal{A}, i = 1, \dots, K$

$\langle \mathcal{A}, \mathcal{R} \rangle$

收益 (反馈) 函数分布:  $\mathcal{R}(r | a^i) = \mathbb{P}(r | a^i)$



- 最大化累积时间的收益:  $\max \sum_{t=1}^T r_t, r_t \sim \mathcal{R}(\cdot | a_t)$

**问题**

不确定的反馈函数, 如何估计?

# 收益估计

- 期望收益和采样次数的关系

$$Q_n(a^i) = \frac{r_1 + r_2 + \dots + r_{n-1}}{n-1}$$

- 缺点：每次更新的空间复杂度是  $O(n)$

## 增量实现

$$Q_{n+1}(a^i) := \frac{1}{n} \sum_{i=1}^n r_i = \frac{1}{n} \left( r_n + \frac{n-1}{n-1} \sum_{i=1}^{n-1} r_i \right) = \frac{1}{n} r_n + \frac{n-1}{n} Q_n = Q_n(a^i) + \frac{1}{n} (r_n - Q_n)$$

误差项:  $\Delta_n^i$



空间复杂度为  $O(1)$

## 算法：多臂老虎机

- I. 初始化:  $Q(a^i) := c^i, N(a^i) = 0, i = 1, \dots, n$
- II. 主循环  $t = 1:T$ 
  1. 利用策略  $\pi$  选取某个动作  $a$
  2. 获取收益:  $r_t = \text{Bandit}(a)$
  3. 更新计数器:  $N(a) := N(a) + 1$
  4. 更新估值:  $Q(a) := Q(a) + \frac{1}{N(a)}[r_t - Q(a)]$

### 权衡探索与利用

应当选取什么样的策略  $\pi$  ?

# Regret函数

- ▣ 决策的期望收益:  $Q(a^i) = \mathbb{E}_{r \sim \mathbb{P}(r|a^i)}[r]$
- ▣ 最优收益:  $Q^* = \max_{a^i \in \mathcal{A}} Q(a^i)$

## Regret

- ▣ 决策与最优决策的收益差:  $R(a^i) = Q^* - Q(a^i)$
- ▣ Total Regret 函数:  $\sigma_R = \mathbb{E}_{a \sim \pi} [\sum_{t=1}^T R(a_t^i)]$

## 等价性

- ▣  $\min \sigma_R = \max \mathbb{E}_{a \sim \pi} [\sum_{t=1}^T Q(a_t^i)]$

随着时间推移, 单步regret越来越小  
那么探索一直都是必须的吗?

# Regret函数

- 如果一直探索新决策:  $\sigma_R \propto T \cdot R$ , total regret 将线性递增, 无法收敛
- 如果一直不探索新决策:  $\sigma_R \propto T \cdot R$ , total regret 仍将线性递增

是否存在一个方法具有次线性 (sublinear) 收敛保证的 regret?

## 下界 (Lai & Robbins)

- 使用  $\Delta_a = Q^* - Q(a)$  和反馈函数分布相似性:  $D_{KL}(\mathcal{R}(r | a) \parallel \mathcal{R}^*(r | a))$  描述

$$\lim_{T \rightarrow \infty} \sigma_R \geq \log T \sum_{a | \Delta_a > 0} \frac{\Delta_a}{D_{KL}(\mathcal{R}(r | a) \parallel \mathcal{R}^*(r | a))} \sim \log T \sum_{a | \Delta_a > 0} \frac{1}{\Delta_a}$$

Lai, Tze Leung, and Herbert Robbins. "Asymptotically efficient adaptive allocation rules." *Advances in applied mathematics* 6.1 (1985): 4-22.

# 贪心策略和 $\epsilon$ -greedy 策略

## 贪心(Greedy)策略

exploitation

$$\hat{Q}(a^i) = \frac{1}{N(a^i)} \sum_{t=1}^T r_t \cdot 1(a_t = a^i)$$

$$a^* = \arg \max_{a^i} \hat{Q}(a^i)$$

$$\sigma_R \propto T \cdot [Q^* - Q(a^i)]$$

线性增长的 Total regret

## $\epsilon$ -greedy 策略

exploitation

$$a_t = \begin{cases} \arg \max_a \hat{Q}(a) & \text{采样概率: } 1 - \epsilon \\ U(0, |\mathcal{A}|) & \text{采样概率: } \epsilon \end{cases}$$

exploration

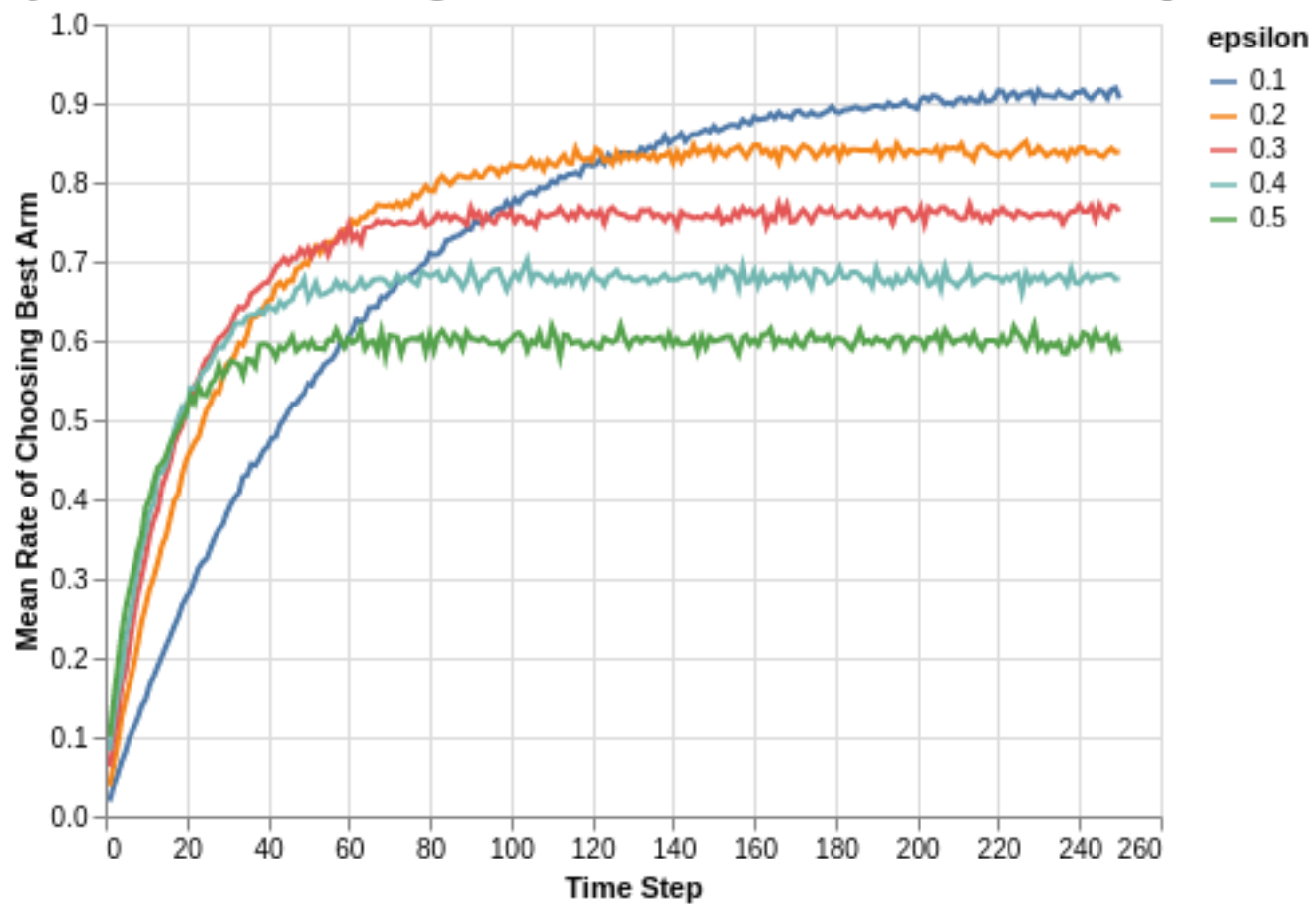
常量  $\epsilon$  保证 total regret 满足

$$\sigma_R \geq \frac{\epsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \Delta_a$$

Total regret 仍然是线性递增的, 只是增长率比贪心策略小

## 不同 $\epsilon$ -greedy 策略对比

Eps-Greedy: Mean Rate of Choosing Best Arm from 5000 Simulations. 5 Arms = [4 x 0.1, 1 x 0.9]





## 衰减贪心策略

- $\epsilon$ -greedy 的变种,  $\epsilon$  随着时间衰减
- 理论上对数渐进收敛!
- 最优的衰减方式:  $\text{regret } R(T) = O\left(\frac{L}{\Delta} \log T\right)$

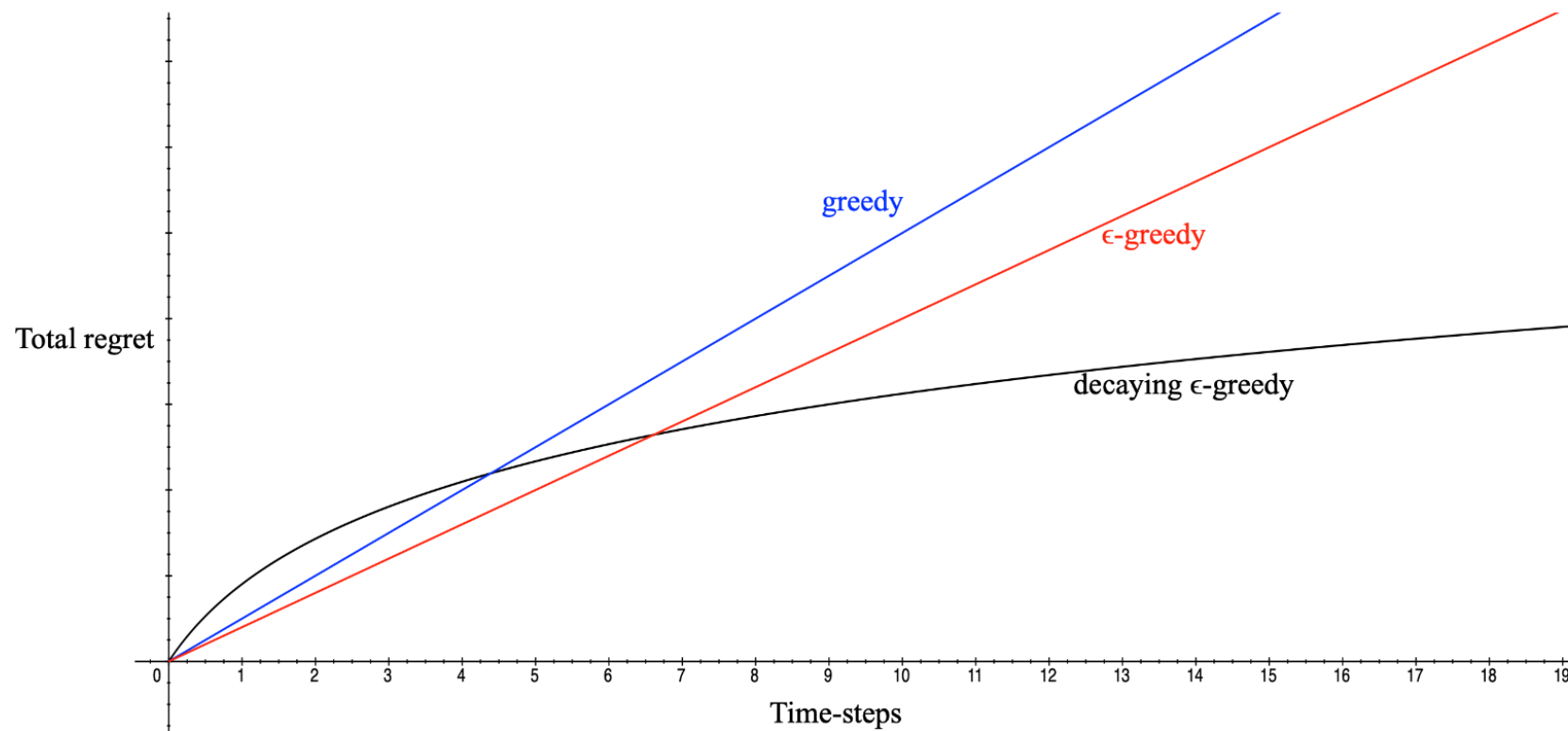
$$c \geq 0, \quad \Delta = \min_{a:\Delta_a > 0} \Delta_a, \quad \epsilon_t = \min \left\{ 1, \frac{c|\mathcal{A}|}{\Delta^2 t} \right\}$$

### 缺点

- 很难找到合适的衰减规划

need the knowledge of  $\Delta$

# 不同 $\epsilon$ -greedy 策略对比: Total Regret

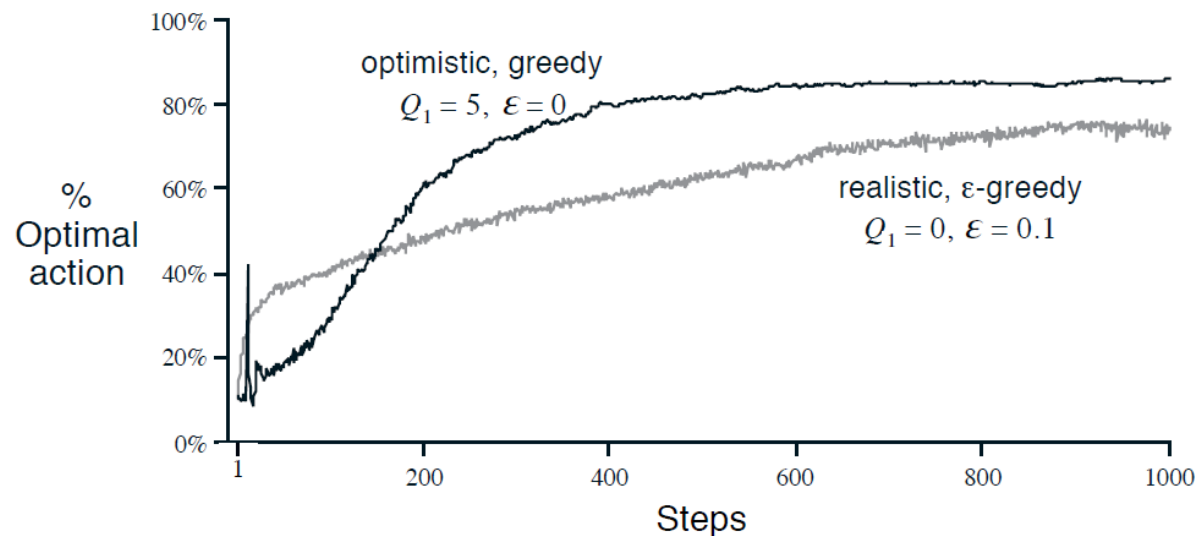


## 乐观初始化

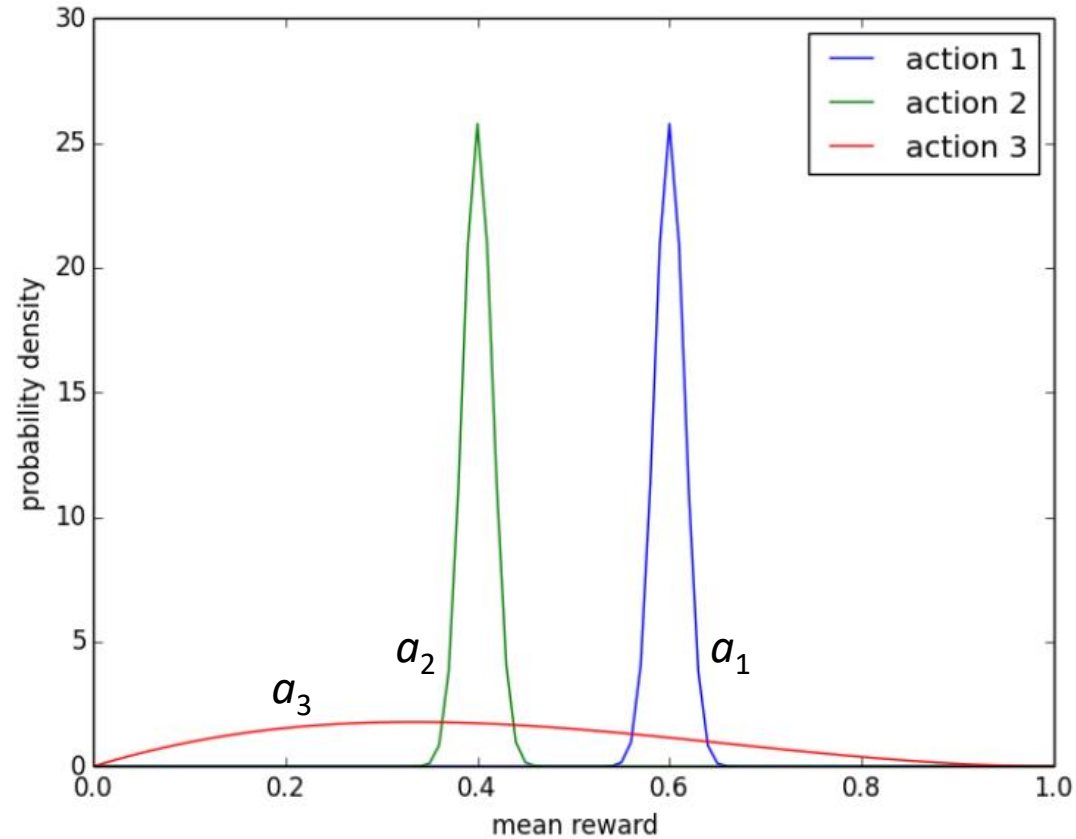
- 给  $Q(a^i)$  一个较高的初始化值
- 增量式蒙特卡洛估计更新  $Q(a^i)$

$$\hat{Q}(a^i) := \hat{Q}(a^i) + \frac{1}{N(a^i)} (r_t - \hat{Q}(a^i))$$

- 有偏估计，但是随着采样增加，这个偏差带来的影响会越来越小
- 但是仍然可能陷入局部最优



## 显式地考虑动作的价值分布



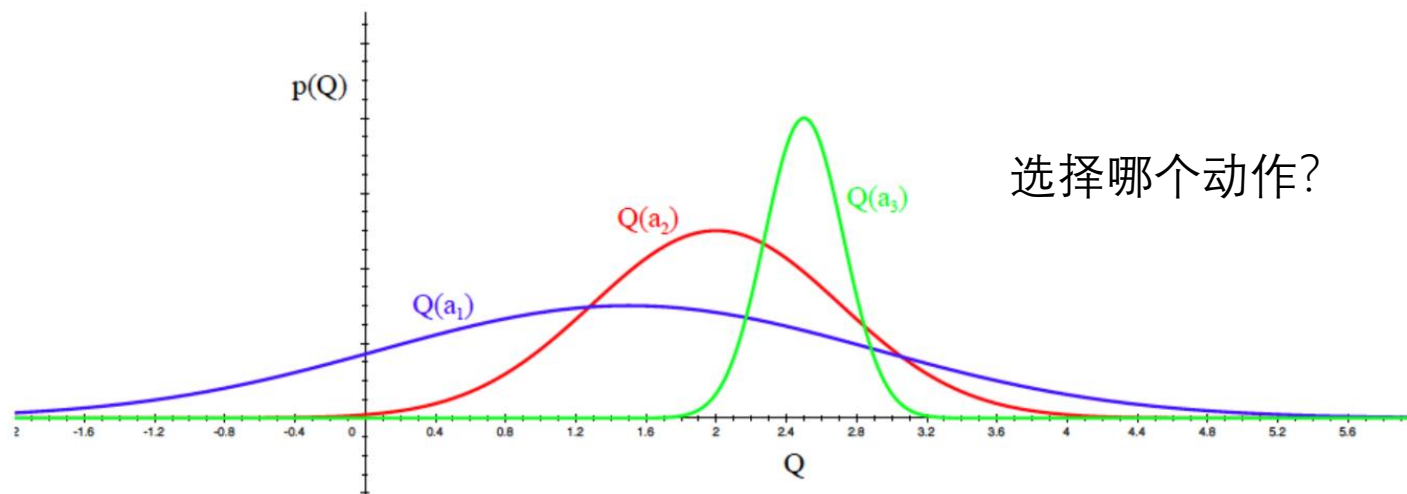
- 考虑以上三个动作的价值分布，平衡探索和利用，选择哪个动作？
- 1. 显式地鼓励不确定性(frequentist); 2. 直接根据分布采样来选择(Bayesian)

## 基于不确定性测度

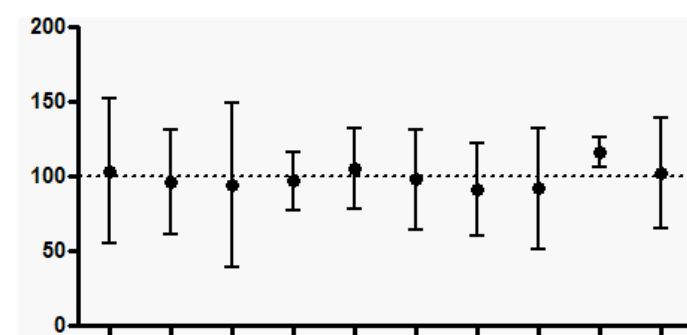
- 不确定性越大的  $Q(a^i)$ , 越具有探索的价值, 有可能会是最好的策略
- 一个经验性指导:
  - $N(a)$  大,  $U(a)$  小
  - $N(a)$  小,  $U(a)$  大

$$\text{策略 } \pi: a = \arg \max_{a \in \mathcal{A}} \hat{Q}(a) + \hat{U}(a)$$

也称为 UCB: 上置信界法 (Upper Confidence Bounds)



# UCB: 置信上界算法



Hoeffding 不等式:  $\mathbb{P}[\mathbb{E}[x] > \bar{x}_t + u] \leq e^{-2tu^2}$  for  $x \in [0,1]$

- 为每个动作收益估值估计一个上置信界:  $\hat{U}(a)$
- 显然有:  $Q_t(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$  以高概率  $p$  成立 (Hoeffding不等式)
- 依据以下原则挑选进行决策:

$$a = \arg \max_{a \in \mathcal{A}} \hat{Q}_t(a) + \hat{U}_t(a) \quad e^{-2N_t(a)U_t(a)^2} = p \Rightarrow \hat{U}_t(a) = \sqrt{-\frac{\log p}{2N_t(a)}}$$

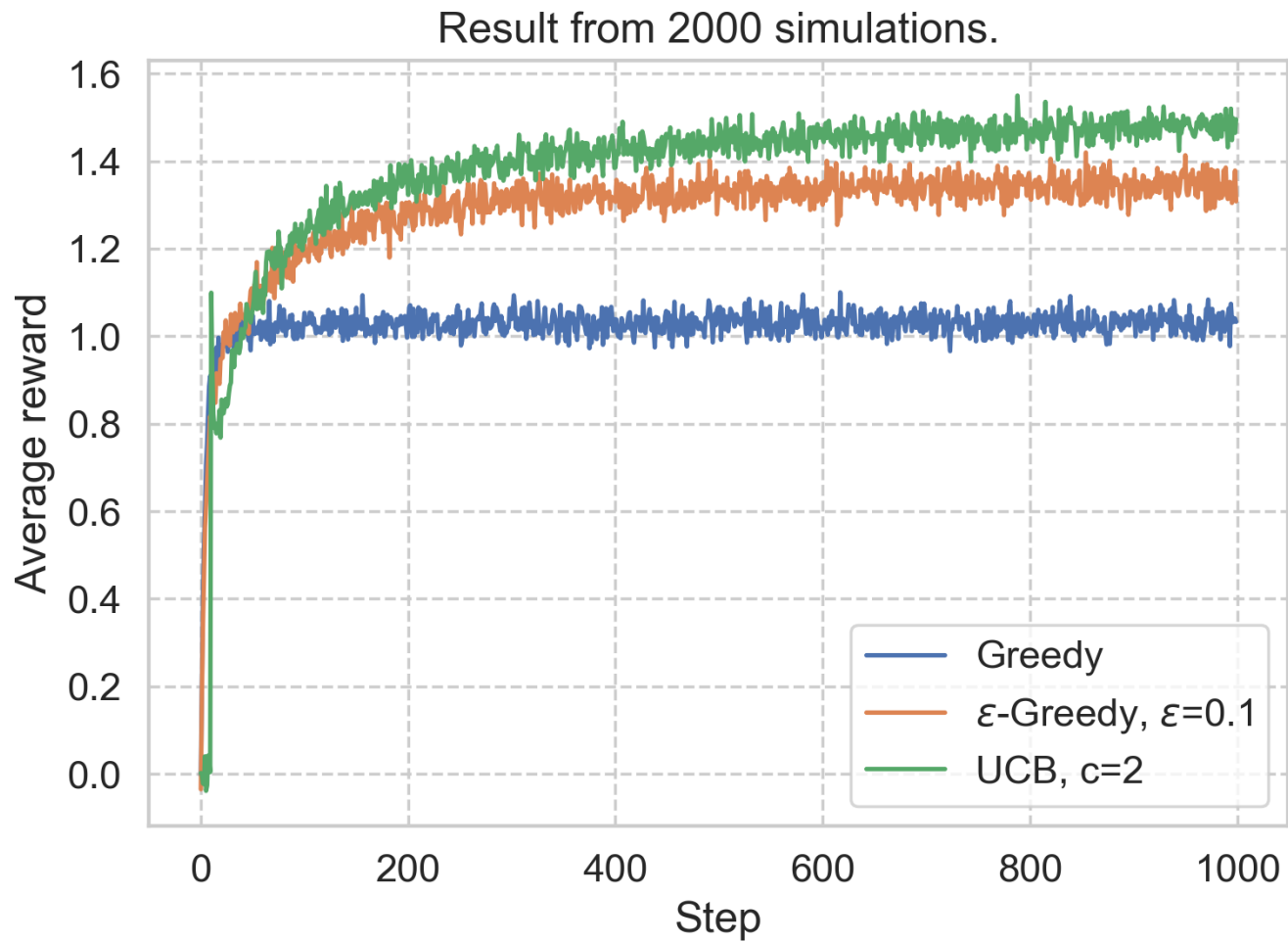
- 收敛性:

exploitation

exploration

$$\lim_{t \rightarrow \infty} \sigma_R \leq 8 \log T \sum_{a: \Delta_a > 0} \Delta_a$$

# UCB的实验效果



# Thompson Sampling

- Assume each arm has prior Gaussian(0,1)
- Then posterior distribution for  $\alpha_a$  is Gaussian  $\left( \hat{Q}_a(t), \frac{1}{1 + T_a(t)} \right)$

- Sample

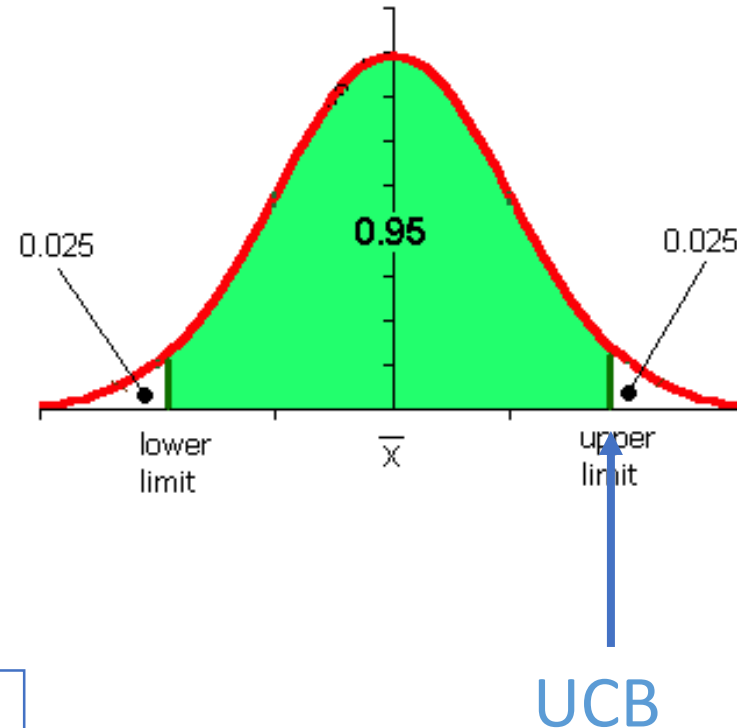
$$\tilde{Q}_a(t) \sim \text{Gaussian} \left( \hat{Q}_a(t), \frac{1}{1 + T_a(t)} \right)$$

and select

$$a_t = \operatorname{argmax}_a \tilde{Q}_a(t)$$

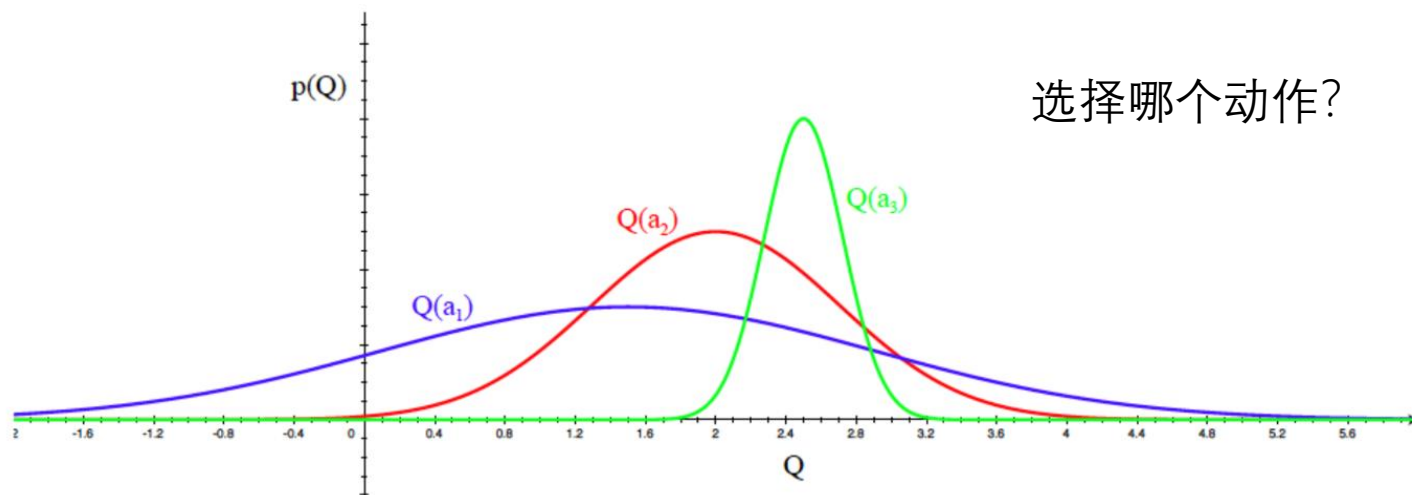
exploitation

exploration





# Thompson Sampling方法

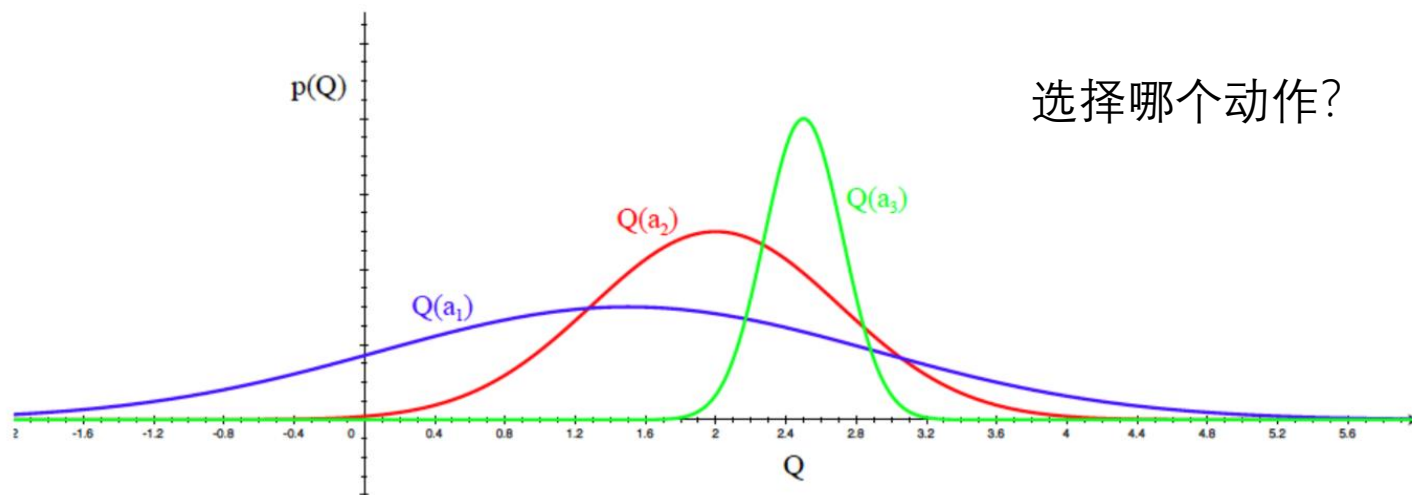


- 想法：根据每个动作成为最优的概率来选择动作

$$p(a) = \int \mathbb{I} \left[ \mathbb{E}_{p(Q(a))} [Q(a; \theta)] = \max_{a' \in \mathcal{A}} \mathbb{E}_{p(Q(a'))} (Q(a'; \theta)) \right] d\theta$$

- 实现：根据当前每个动作 $a$ 的价值概率分布 $p(Q(a))$ 来采样到其价值 $Q(a)$ ，选择价值最大的动作

# Thompson Sampling方法



- 实现：根据当前每个动作 $a$ 的价值概率分布 $p(Q(a))$ 来采样到其价值 $Q(a)$ ，选择价值最大的动作

---

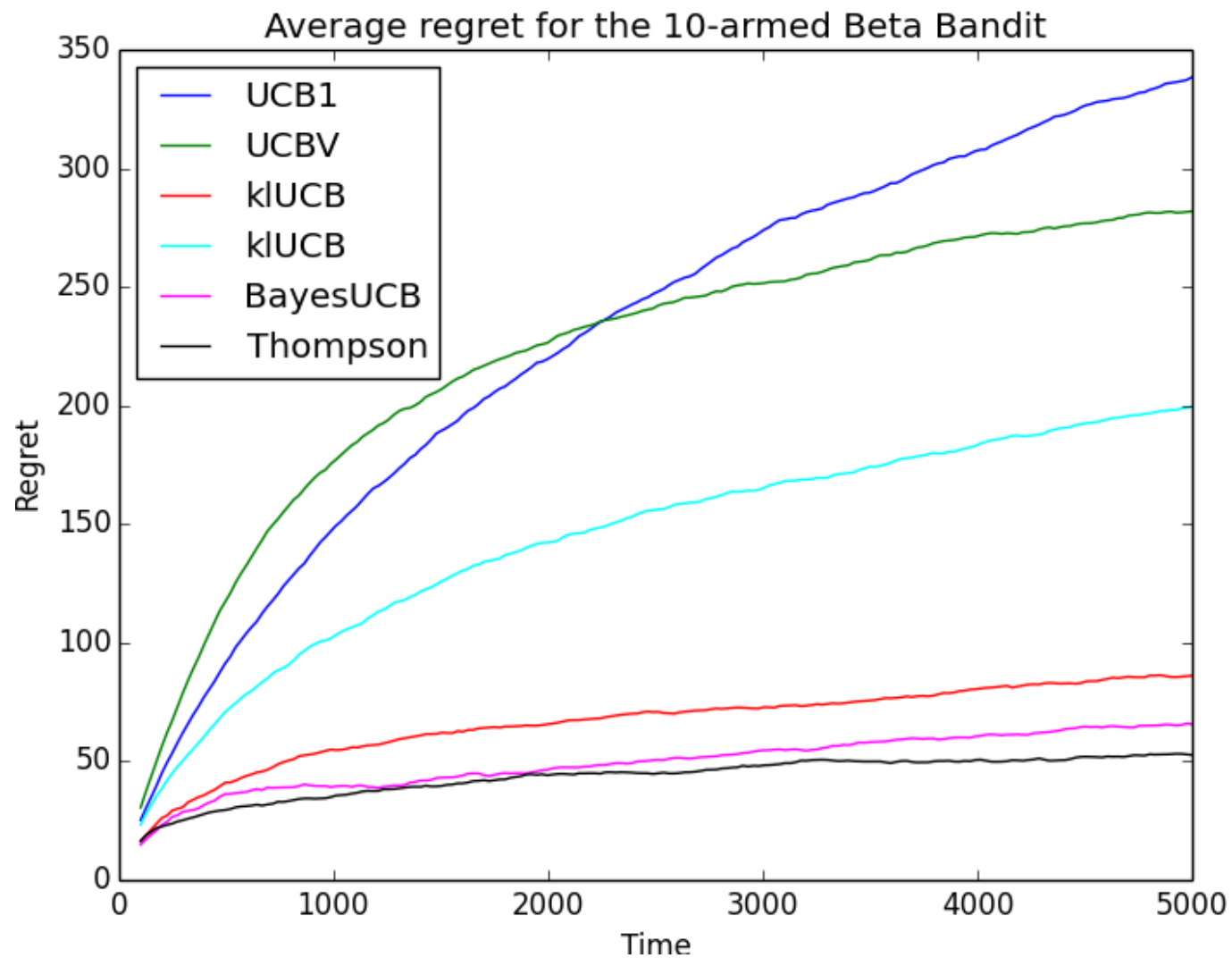
## Algorithm 1 Thompson sampling

---

```
 $D = \emptyset$   
for  $t = 1, \dots, T$  do  
  Receive context  $x_t$   
  Draw  $\theta^t$  according to  $P(\theta|D)$   
  Select  $a_t = \arg \max_a \mathbb{E}_r(r|x_t, a, \theta^t)$   
  Observe reward  $r_t$   
   $D = D \cup (x_t, a_t, r_t)$   
end for
```

---

# Thompson Sampling 实验对比



## 探索与利用总结

- 探索与利用是强化学习的trial-and-error中的必备技术
- 多臂老虎机可以被看成是无状态(state-less)强化学习
- 多臂老虎机是研究探索与利用技术理论的最佳环境
  - 理论的渐近最优regret为  $O(\log T)$
- $\epsilon$ -greedy、UCB和Thompson Sampling方法在多臂老虎机任务中十分常用，在强化学习的探索中用也十分常用，最常见的是 $\epsilon$ -greedy

**THANK YOU**