



南京大學
NANJING UNIVERSITY

人工智能导论

强化学习

(Reinforcement Learning)

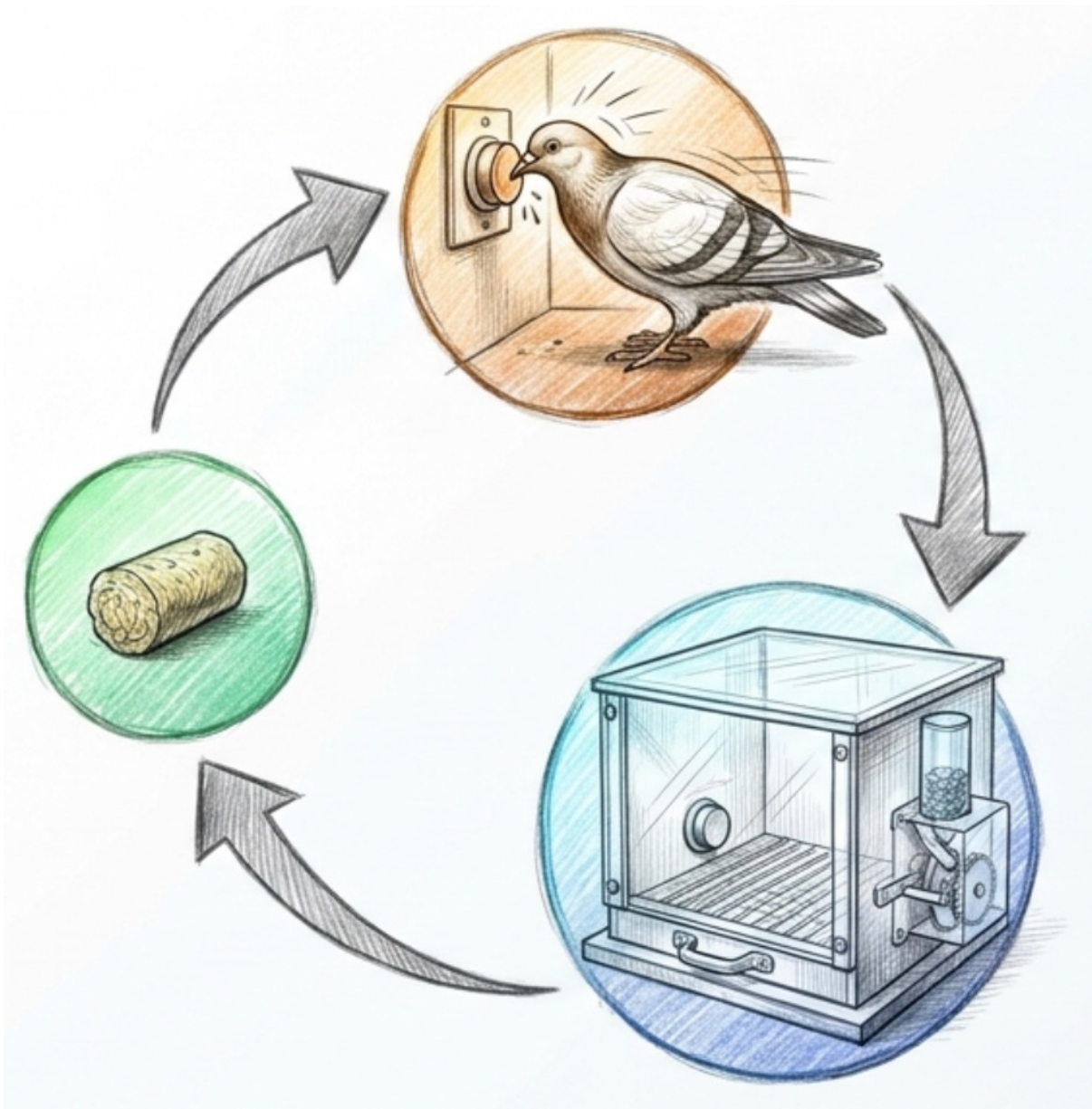
郭兰哲

南京大学 智能科学与技术学院

<https://www.lamda.nju.edu.cn/guolz>

Email: guolz@nju.edu.cn

斯金纳实验：行为与反馈的闭环 (1930s)

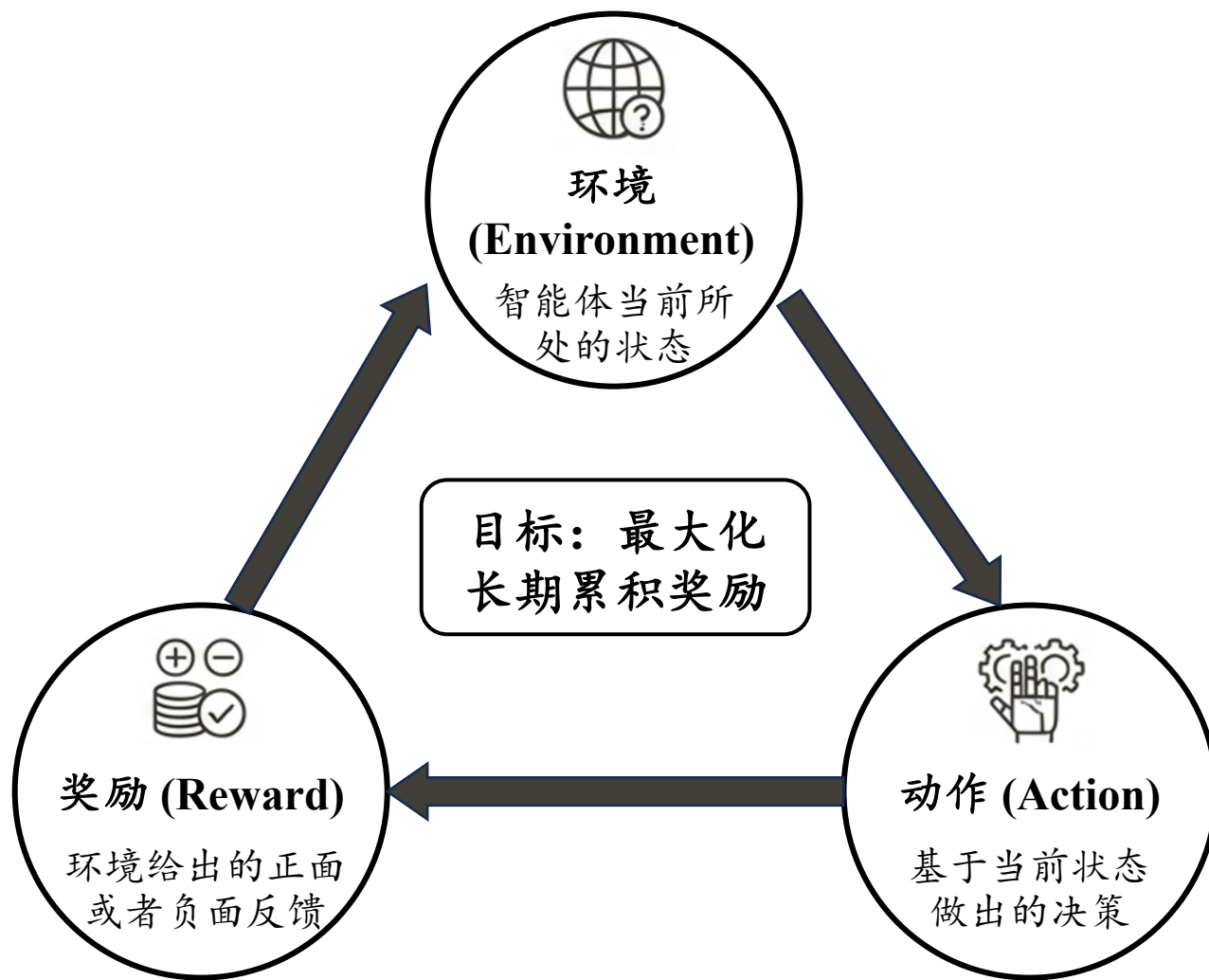


“被奖励的行为趋向于重复，被惩罚的行为趋向于消失。”

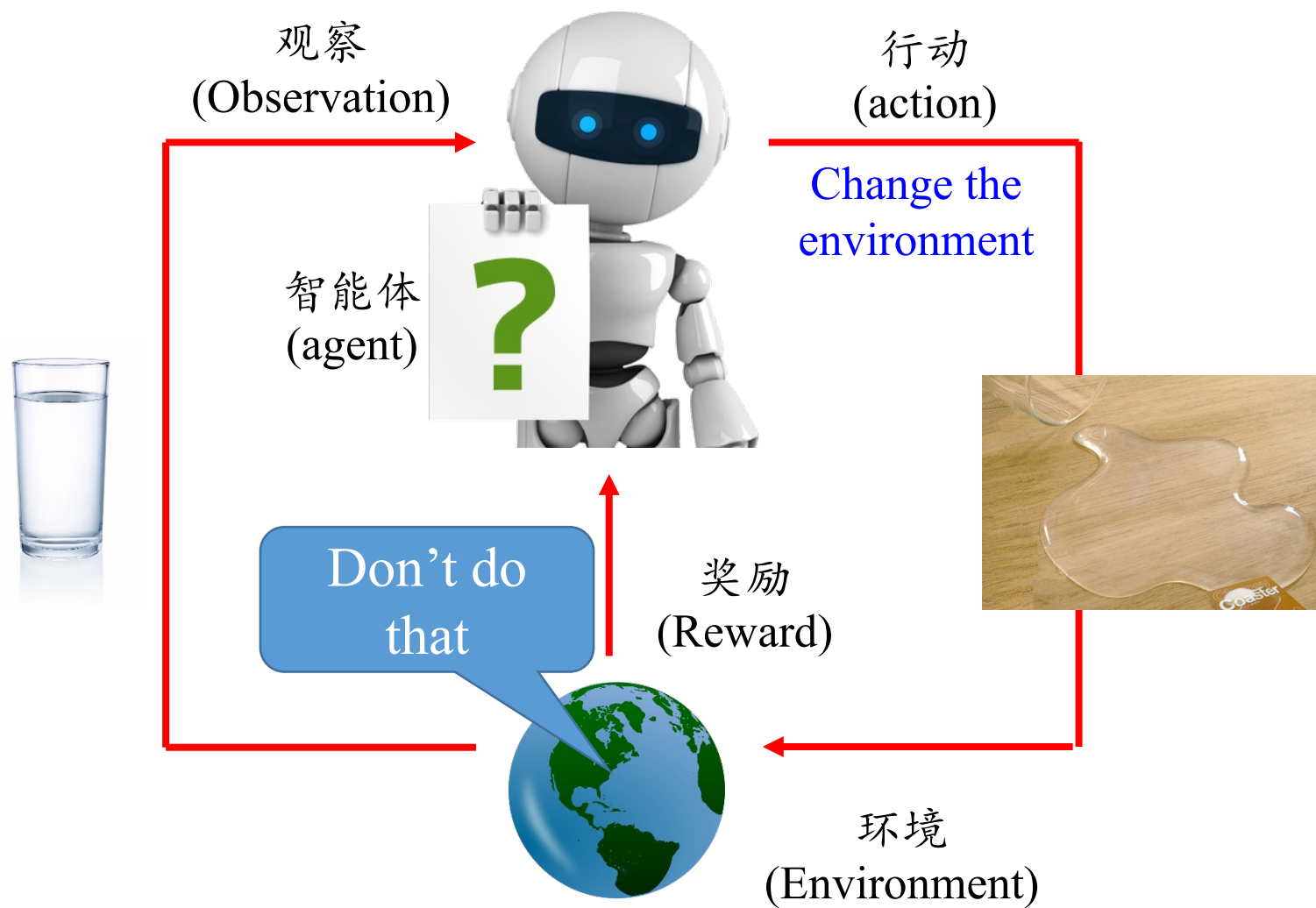
-- B.F. 斯金纳 《有机体的行为》，1938

斯金纳发现，鸽子通过“随机试错 + 获得反馈”学会了按压按钮，他将此命名为操作性条件反射

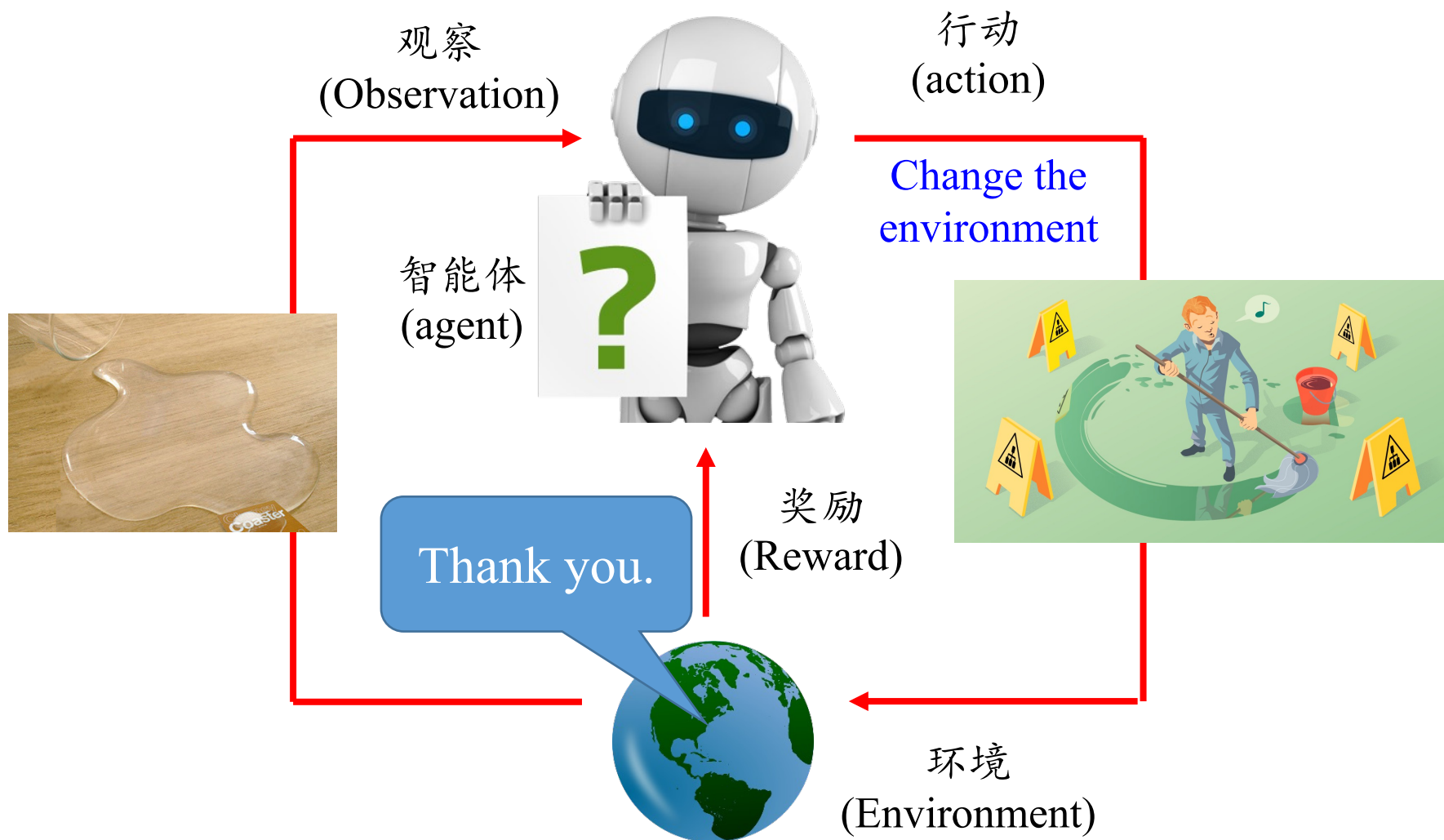
从生物学直接到计算框架：探索、反馈、强化



强化学习 (Reinforcement Learning)



强化学习 (Reinforcement Learning)



智能体从交互中学习得到最大化累计奖励的行动

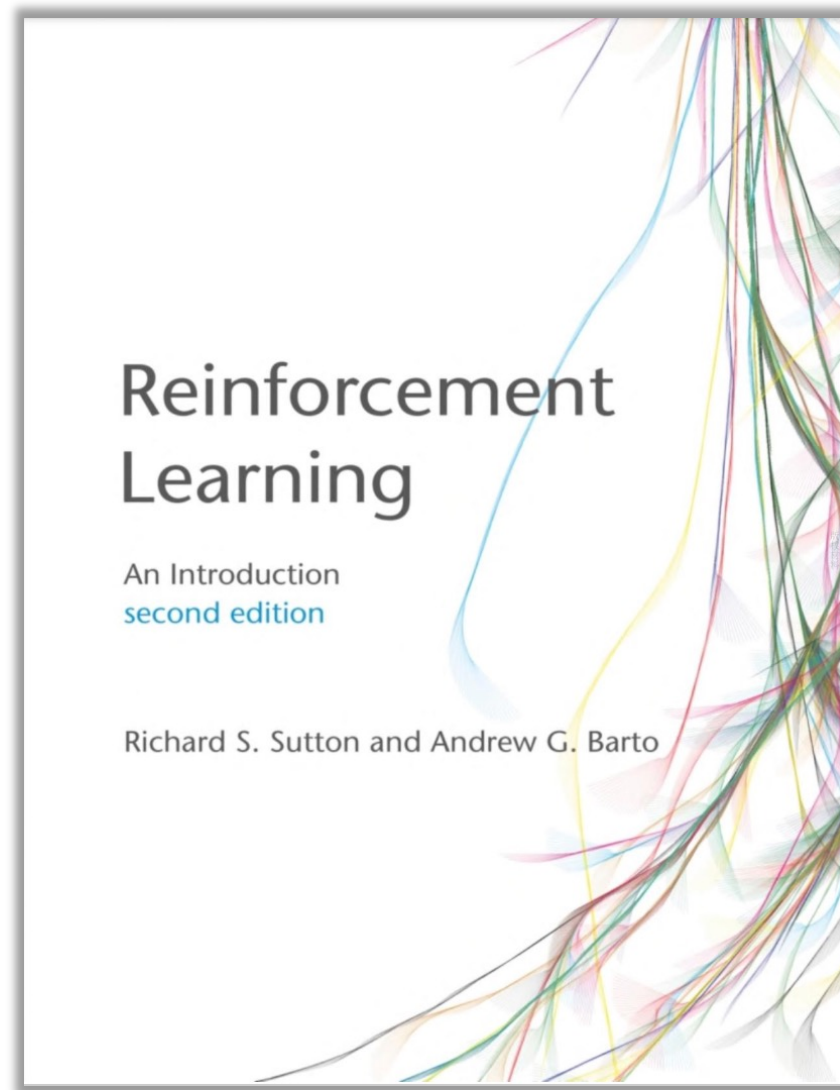
强化学习之父：理查德萨顿 (Richard Sutton)



Richard Sutton

2024年图灵奖得主

<http://incompleteideas.net/>



DeepMind的智力革命

2010年DeepMind在伦敦成立，愿景“解决智能，然后用它解决一切”

- 2010年DeepMind成立：Demis Hassabis, Shane Legg, Mustafa Suleyman
- 2013年发表DQN论文，在40款雅达利游戏中超越人类玩家
- 2014年被Google以6.5亿美元收购
- 2016年，AlphaGo，战胜世界围棋冠军李世石
- 2020年，AlphaFold，解决蛋白质预测问题
- 2024年，获诺贝尔化学奖



2008 年，姆尼赫在阿尔伯塔大学获得硕士学位后回到多伦多，加入了辛顿的团队，在一间改造过的储藏室里办公。⁵ 在那里，他遇到了与塞佩斯瓦里相同的情况——深度学习研究者也不愿采用其他学派的方法。

“你想研究什么？”辛顿问他的新博士生。

姆尼赫回答说，他想将强化学习与深度学习结合起来。“我试过了，行不通。”辛顿劝告他。

姆尼赫又向其他几位同事提起了自己的抱负，但一位又一位深度学习研究者都劝他放弃。“一旦你成为强化学习研究者，就会进入一个独立的圈子，我们再也不会听到你的消息了。”有人明确地告诉他。

2012 年夏天，姆尼赫在苏格兰的一场 AI 会议上展示了他的博士研究成果。这场会议的主题多是像他的项目这样的严谨项目，打造 AGI 的未来主义方案并未出现在议程中。但在第一天晚上的招待会上，氛围突然发生了变化。两位参会者出现在派对上，宣布他们正在打造 AGI。他们在伦敦有一家初创公司，正在寻找认同这一使命的加入者。

DeepMind 的人问他：“你想研究什么？”姆尼赫给出了那个通常会让他遭到白眼的回答：“我想尝试将深度学习与强化学习结合起来。”

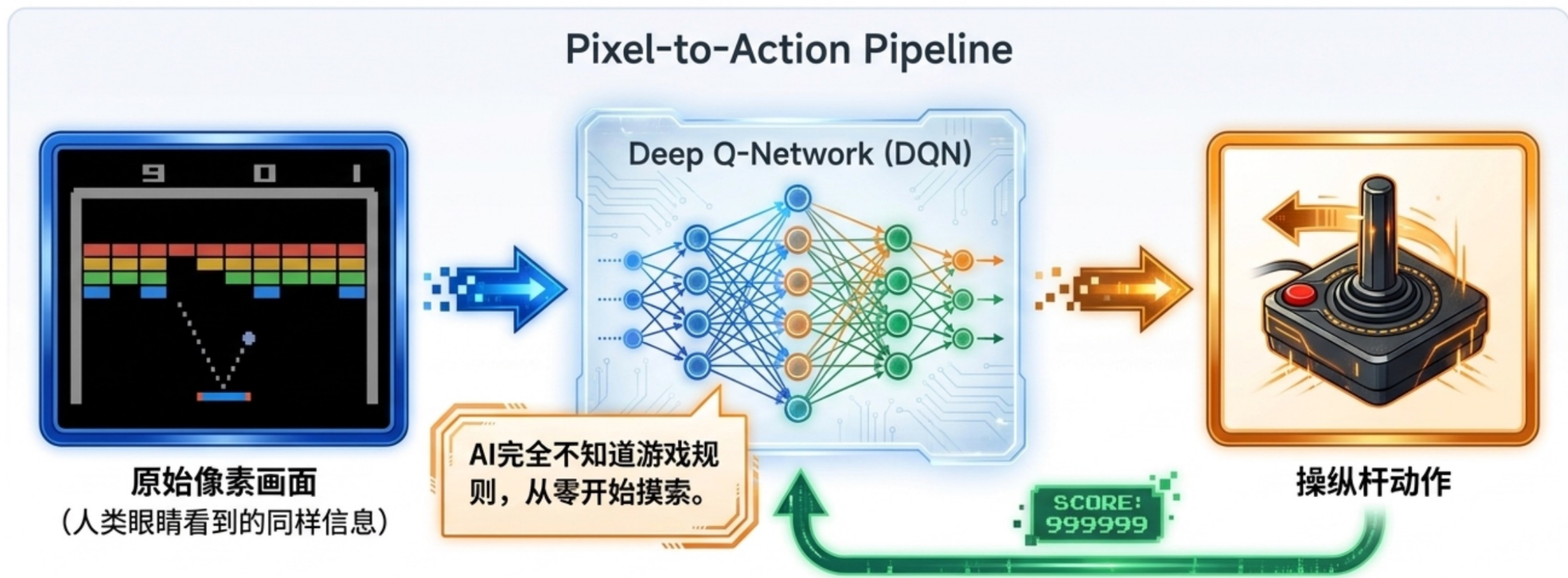
“我们正在做的就是这件事！”两人高兴地回答。在瑞士，这种跨领域结合的做法实际上是受到鼓励的。此外，神经科学也强烈表明，强化学习将是深度学习的必要补充。毕竟，强化学习中的奖励信号与人类大脑中的多巴胺信号非常相似。如果大脑是 AGI 的模板，那么强化学习对于打造 AGI 来说必不可少。⁷

姆尼赫开始觉得，这些“疯狂的家伙”可能真的知道些什么。他曾在多伦多和阿尔伯塔大学之间辗转，试图将自己对两大研究的热情结合起来。或许他应该把自己“转移”到伦敦去。

此外，姆尼赫意识到，打造 AGI 的抱负听起来可能有些狂妄，但它也有一个优势。根据他的经验，学术界的文化既枯燥保守又竞争激烈：枯燥是因为科学家们只追求渐进式进步，激烈是因为为了抢占首发权而互相倾轧。莱格和维尔斯特拉承诺会给他带来完全不同的体验——**追求巨大突破的激动人心的过程，以及几乎没有竞争对手的环境。**“他们说，是的，我们要做的事情没有竞争，因为没有人认为这是可能的，”姆尼赫回忆道，“而且如果成功了，它的影响将是巨大的。”⁸

第一声惊雷：仅凭像素学会玩雅达利游戏 (2013)

DeepMind两篇发表于NeurIPS和《Nature》的论文，提出了Deep Q-Network (DQN)算法，
在49款经典雅达利游戏中，29款超越了人类顶尖水平



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

Abstract

We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

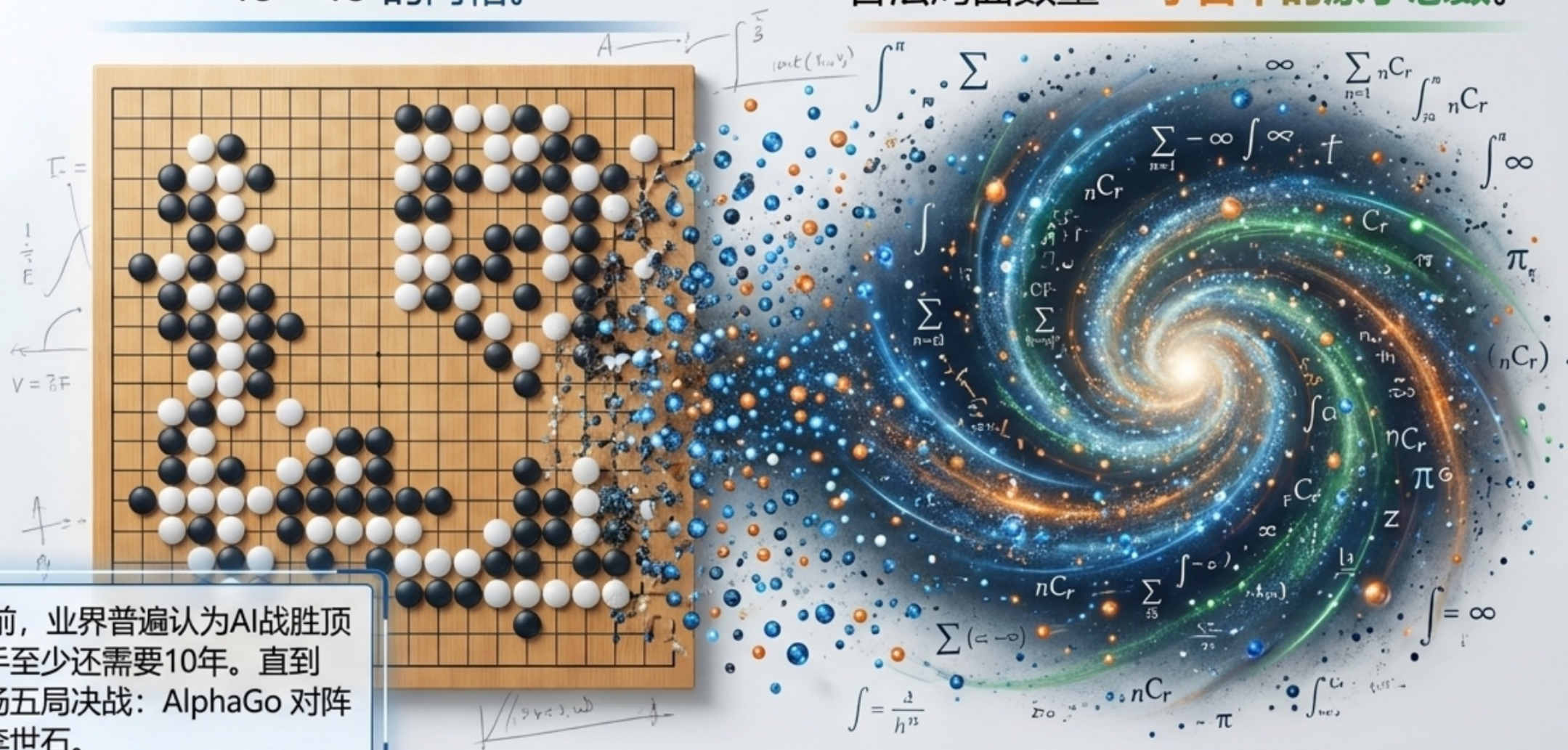
<https://arxiv.org/pdf/1312.5602>



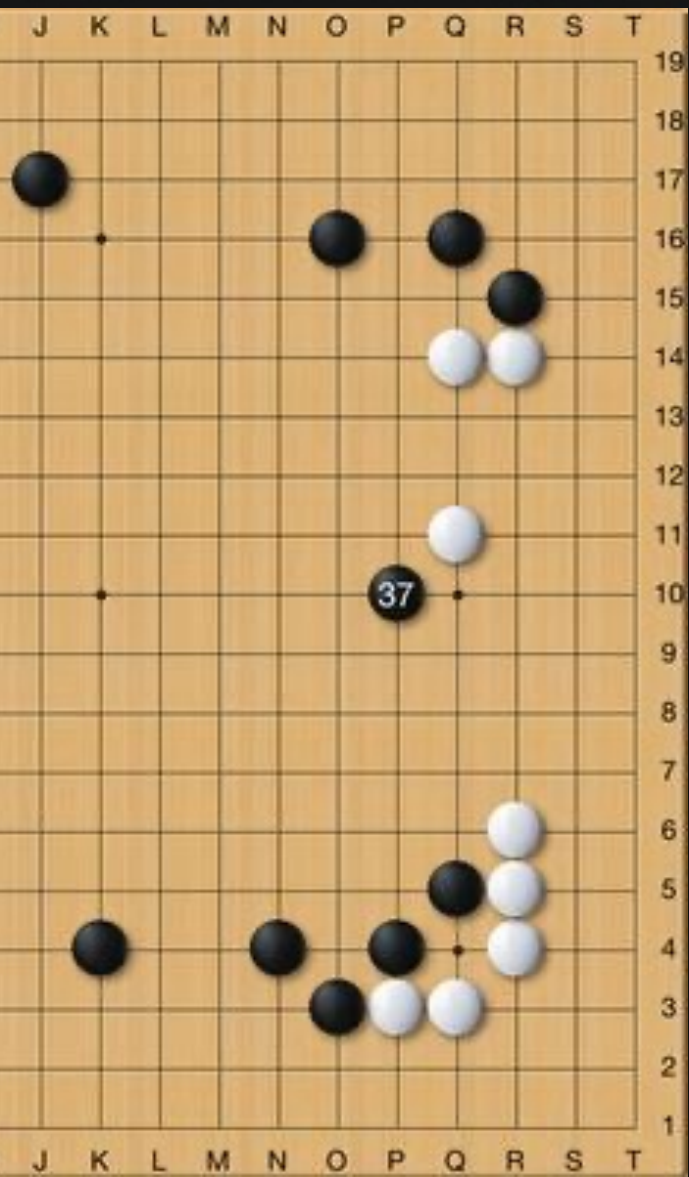
挑战人类智力巅峰：围棋 (2016)

19 × 19 的网格。

合法局面数量 > 宇宙中的原子总数。



AI的神之一手：不属于人类的、美丽的直觉



历史的涌现：神之一手（Move 37）

第二局，第37手。
AlphaGo下出了一步极其罕见的“五线肩冲”。

当下：解说员与人类顶级棋手
纷纷断言：“这步棋下错了，这
是机器的Bug。”

思考：李世石震惊，起身离开
棋盘，随后陷入长达11分钟的
死寂沉思。

15分钟后：全世界的围棋大师
突然醒悟——这是一步洞穿全局
的天才之举。

这步棋不是人类
会下的棋。但它是
美丽的。
——职业棋手评价

最终比分 4:1。它不仅赢下了一盘棋，更粉碎了
人类对于“机器无直觉”的傲慢。

Mastering the game of Go with deep neural networks and tree search

David Silver^{1*}, Aja Huang^{1*}, Chris J. Maddison¹, Arthur Guez¹, Laurent Sifre¹, George van den Driessche¹, Julian Schrittwieser¹, Ioannis Antonoglou¹, Veda Panneershelvam¹, Marc Lanctot¹, Sander Dieleman¹, Dominik Grewe¹, John Nham², Nal Kalchbrenner¹, Ilya Sutskever², Timothy Lillicrap¹, Madeleine Leach¹, Koray Kavukcuoglu¹, Thore Graepel¹ & Demis Hassabis¹

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses ‘value networks’ to evaluate board positions and ‘policy networks’ to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.

All games of perfect information have an optimal value function, $v^*(s)$, which determines the outcome of the game, from every board position or state s , under perfect play by all players. These games may be solved by recursively computing the optimal value function in a search tree containing approximately b^d possible sequences of moves, where b is the game’s breadth (number of legal moves per position) and d is its depth (game length). In large games, such as chess ($b \approx 35$, $d \approx 80$)¹ and especially Go ($b \approx 250$, $d \approx 150$)¹, exhaustive search is infeasible^{2,3}, but the effective search space can be reduced by two general principles. First, the depth of the search may be reduced by position evaluation: truncating the search tree at state s and replacing the subtree below s by an approximate value function $v(s) \approx v^*(s)$ that predicts the outcome from state s . This approach has led to superhuman performance in chess⁴, checkers⁵ and othello⁶, but it was believed to be intractable in Go due to the complexity of the game⁷. Second, the breadth of the search may be reduced by sampling actions from a policy $p(a|s)$ that is a probability distribution over possible moves a in position s . For example, Monte Carlo rollouts⁸ search to maximum depth without branching at all, by sampling long sequences of actions for both players from a policy p . Averaging over such rollouts can provide an effective position evaluation, achieving superhuman performance in backgammon⁸ and Scrabble⁹, and weak amateur level play in Go¹⁰.

Monte Carlo tree search (MCTS)^{11,12} uses Monte Carlo rollouts to estimate the value of each state in a search tree. As more simulations are executed, the search tree grows larger and the relevant values become more accurate. The policy used to select actions during

policies^{13–15} or value functions¹⁶ based on a linear combination of input features.

Recently, deep convolutional neural networks have achieved unprecedented performance in visual domains: for example, image classification¹⁷, face recognition¹⁸, and playing Atari games¹⁹. They use many layers of neurons, each arranged in overlapping tiles, to construct increasingly abstract, localized representations of an image²⁰. We employ a similar architecture for the game of Go. We pass in the board position as a 19×19 image and use convolutional layers to construct a representation of the position. We use these neural networks to reduce the effective depth and breadth of the search tree: evaluating positions using a value network, and sampling actions using a policy network.

We train the neural networks using a pipeline consisting of several stages of machine learning (Fig. 1). We begin by training a supervised learning (SL) policy network p_θ directly from expert human moves. This provides fast, efficient learning updates with immediate feedback and high-quality gradients. Similar to prior work^{13,15}, we also train a fast policy p_π that can rapidly sample actions during rollouts. Next, we train a reinforcement learning (RL) policy network p_ρ that improves the SL policy network by optimizing the final outcome of games of self-play. This adjusts the policy towards the correct goal of winning games, rather than maximizing predictive accuracy. Finally, we train a value network v_θ that predicts the winner of games played by the RL policy network against itself. Our program AlphaGo efficiently combines the policy and value networks with MCTS.

Mastering the game of Go without human knowledge

David Silver^{1*}, Julian Schrittwieser^{1*}, Karen Simonyan^{1*}, Ioannis Antonoglou¹, Aja Huang¹, Arthur Guez¹, Thomas Hubert¹, Lucas Baker¹, Matthew Lai¹, Adrian Bolton¹, Yutian Chen¹, Timothy Lillicrap¹, Fan Hui¹, Laurent Sifre¹, George van den Driessche¹, Thore Graepel¹ & Demis Hassabis¹

A long-standing goal of artificial intelligence is an algorithm that learns, *tabula rasa*, superhuman proficiency in challenging domains. Recently, AlphaGo became the first program to defeat a world champion in the game of Go. The tree search in AlphaGo evaluated positions and selected moves using deep neural networks. These neural networks were trained by supervised learning from human expert moves, and by reinforcement learning from self-play. Here we introduce an algorithm based solely on reinforcement learning, without human data, guidance or domain knowledge beyond game rules. AlphaGo becomes its own teacher: a neural network is trained to predict AlphaGo’s own move selections and also the winner of AlphaGo’s games. This neural network improves the strength of the tree search, resulting in higher quality move selection and stronger self-play in the next iteration. Starting *tabula rasa*, our new program AlphaGo Zero achieved superhuman performance, winning 100–0 against the previously published, champion-defeating AlphaGo.

Much progress towards artificial intelligence has been made using supervised learning systems that are trained to replicate the decisions of human experts^{1–4}. However, expert data sets are often expensive, unreliable or simply unavailable. Even when reliable data sets are available, they may impose a ceiling on the performance of systems trained in this manner⁵. By contrast, reinforcement learning systems are trained from their own experience, in principle allowing them to exceed human capabilities, and to operate in domains where human expertise is lacking. Recently, there has been rapid progress towards this goal, using deep neural networks trained by reinforcement learning. These systems have outperformed humans in computer games, such as Atari^{6,7} and 3D virtual environments^{8–10}. However, the most challenging domains in terms of human intellect—such as the game of Go, widely viewed as a grand challenge for artificial intelligence¹¹—require a precise and sophisticated lookahead in vast search spaces. Fully general methods have not previously achieved human-level performance in these domains.

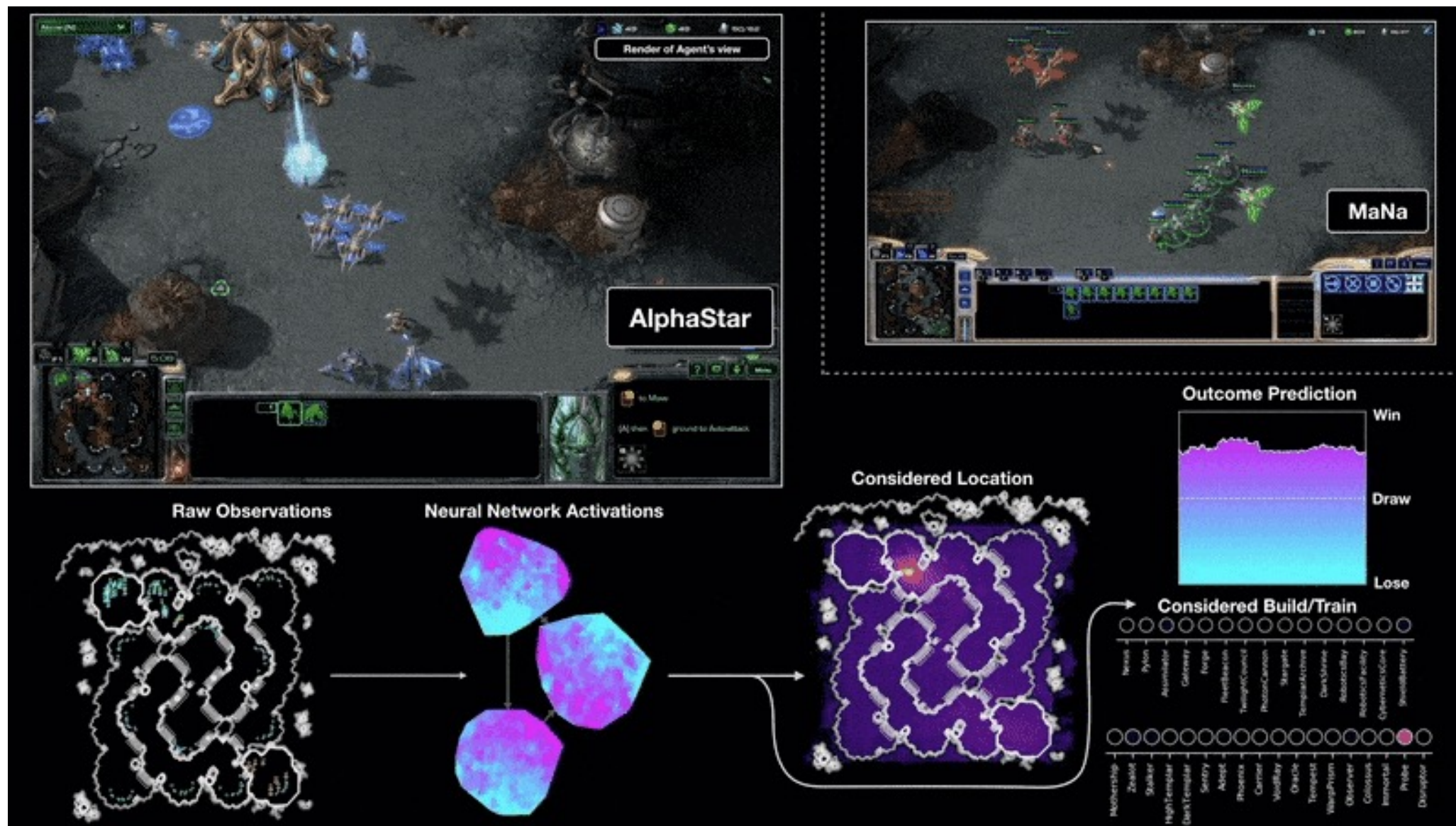
AlphaGo was the first program to achieve superhuman performance in Go. The published version¹², which we refer to as AlphaGo Fan, defeated the European champion Fan Hui in October 2015. AlphaGo Fan used two deep neural networks: a policy network that outputs move probabilities and a value network that outputs a position eval-

uated solely by self-play reinforcement learning, starting from random play, without any supervision or use of human data. Second, it uses only the black and white stones from the board as input features. Third, it uses a single neural network, rather than separate policy and value networks. Finally, it uses a simpler tree search that relies upon this single neural network to evaluate positions and sample moves, without performing any Monte Carlo rollouts. To achieve these results, we introduce a new reinforcement learning algorithm that incorporates lookahead search inside the training loop, resulting in rapid improvement and precise and stable learning. Further technical differences in the search algorithm, training procedure and network architecture are described in Methods.

Reinforcement learning in AlphaGo Zero

Our new method uses a deep neural network f_θ with parameters θ . This neural network takes as an input the raw board representation s of the position and its history, and outputs both move probabilities and a value, $(\mathbf{p}, v) = f_\theta(s)$. The vector of move probabilities $\mathbf{p}$ represents the probability of selecting each move a (including pass), $p_a = \Pr(a|s)$. The value v is a scalar evaluation, estimating the probability of the current player winning from position s . This neural network combines the roles of both policy network and value network¹² into a single architecture.

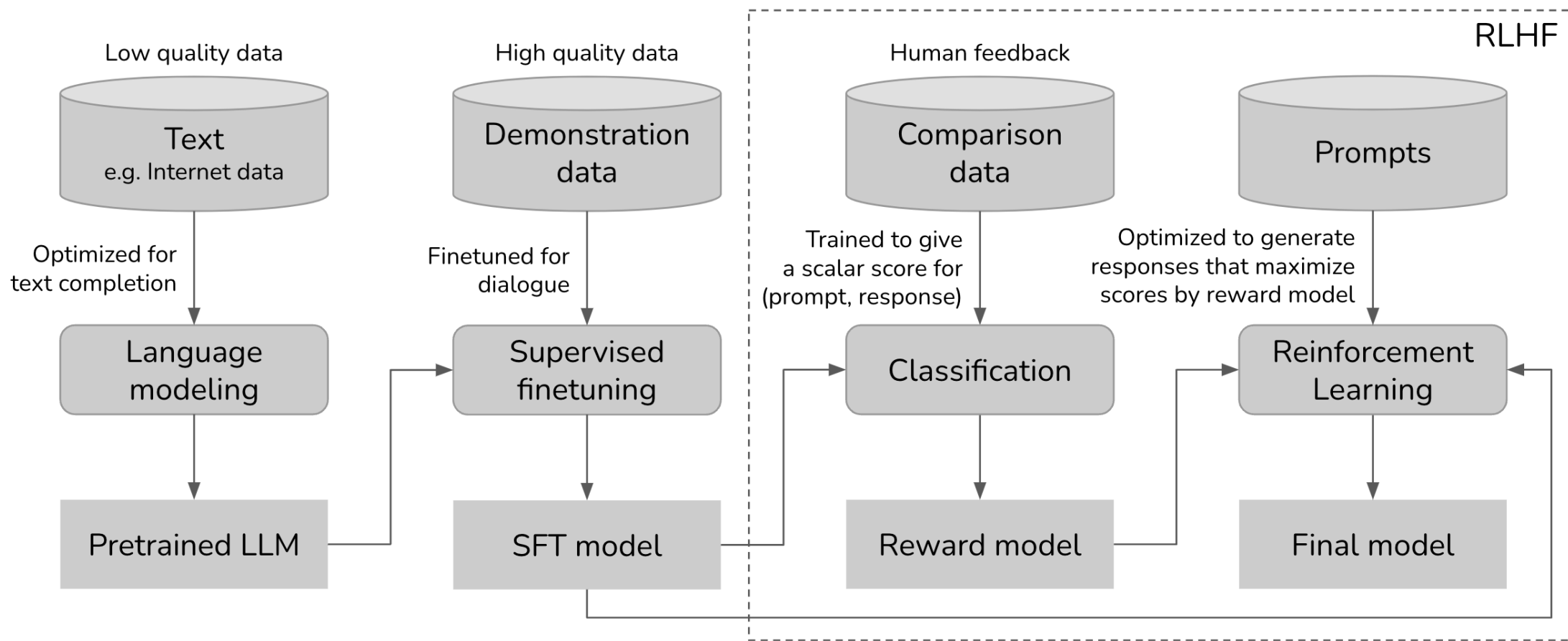
挑战星际争霸：AlphaStar (2019)



<https://deepmind.google/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii/>

大模型时代的强化学习 - 2022

ChatGPT训练的三阶段：预训练(Pre-Train)、有监督微调(Supervised Fine-Tuning)、
基于人类反馈的强化学习(Reinforcement Learning from Human Feedback)



DeepSeek-R1 (2025)



DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

DeepSeek-AI

research@deepseek.com

Abstract

General reasoning represents a long-standing and formidable challenge in artificial intelligence. Recent breakthroughs, exemplified by large language models (LLMs) (Brown et al., 2020; OpenAI, 2023) and chain-of-thought prompting (Wei et al., 2022b), have achieved considerable success on foundational reasoning tasks. However, this success is heavily contingent upon extensive human-annotated demonstrations, and models' capabilities are still insufficient for more complex problems. Here we show that the reasoning abilities of LLMs can be incentivized through pure reinforcement learning (RL), obviating the need for human-labeled reasoning trajectories. The proposed RL framework facilitates the emergent development of advanced reasoning patterns, such as self-reflection, verification, and dynamic strategy adaptation. Consequently, the trained model achieves superior performance on verifiable tasks such as mathematics, coding competitions, and STEM fields, surpassing its counterparts trained via conventional supervised learning on human demonstrations. Moreover, the emergent reasoning patterns exhibited by these large-scale models can be systematically harnessed to guide and enhance the reasoning capabilities of smaller models.

<https://arxiv.org/abs/2501.12948>

Article

DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning

<https://doi.org/10.1038/s41586-025-09422-z>

Received: 14 February 2025

Accepted: 17 July 2025

Published online: 17 September 2025

Open access

Check for updates

General reasoning represents a long-standing and formidable challenge in artificial intelligence (AI). Recent breakthroughs, exemplified by large language models (LLMs)^{1,2} and chain-of-thought (CoT) prompting³, have achieved considerable success on foundational reasoning tasks. However, this success is heavily contingent on extensive human-annotated demonstrations and the capabilities of models are still insufficient for more complex problems. Here we show that the reasoning abilities of LLMs can be incentivized through pure reinforcement learning (RL), obviating the need for human-labelled reasoning trajectories. The proposed RL framework facilitates the emergent development of advanced reasoning patterns, such as self-reflection, verification and dynamic strategy adaptation. Consequently, the trained model achieves superior performance on verifiable tasks such as mathematics, coding competitions and STEM fields, surpassing its counterparts trained through conventional supervised learning on human demonstrations. Moreover, the emergent reasoning patterns exhibited by these large-scale models can be systematically used to guide and enhance the reasoning capabilities of smaller models.

Reasoning capability, the cornerstone of human intelligence, enables complex cognitive tasks ranging from mathematical problem-solving to logical deduction and programming. Recent advances in AI have demonstrated that LLMs can exhibit emergent behaviours, including reasoning abilities, when scaled to a sufficient size^{4,5}. However, achieving such capabilities in pre-training typically demands substantial computational resources. In parallel, a complementary line of research has demonstrated that LLMs can be effectively augmented through CoT prompting. This technique, which involves either providing carefully designed few-shot examples or using minimalistic prompts such as “Let’s think step by step”^{3,6}, enables models to produce intermediate reasoning steps, thereby substantially enhancing their performance on complex tasks. Similarly, further performance gains have been observed when models learn high-quality, multistep reasoning trajectories during the post-training phase^{2,7}. Despite their effectiveness, these approaches exhibit notable limitations. Their dependence on human-annotated reasoning traces slows scalability and introduces cognitive biases. Furthermore, by constraining models to replicate human thought processes, their performance is inherently capped by the human-provided exemplars, which prevents the exploration of superior, non-human-like reasoning pathways.

To tackle these issues, we aim to explore the potential of LLMs for developing reasoning abilities through self-evolution in a RL framework, with minimal reliance on human labelling efforts. Specifically, we build on DeepSeek-V3 Base⁸ and use Group Relative Policy Optimization (GRPO)⁹ as our RL framework. The reward signal is only based on the correctness of final predictions against ground-truth answers, without imposing constraints on the reasoning process itself. Notably, we bypass the conventional supervised fine-tuning (SFT) phase before RL training. This design choice originates from our hypothesis that human-defined reasoning patterns may limit model exploration,

whereas unrestricted RL training can better incentivize the emergence of new reasoning capabilities in LLMs. Through this process, detailed in the next section, our model (referred to as DeepSeek-R1-Zero) naturally developed diverse and sophisticated reasoning behaviours. To solve reasoning problems, the model exhibits a tendency to generate longer responses, incorporating verification, reflection and the exploration of alternative approaches within each response. Although we do not explicitly teach the model how to reason, it successfully learns improved reasoning strategies through RL.

Although DeepSeek-R1-Zero demonstrates excellent reasoning capabilities, it faces challenges such as poor readability and language mixing, occasionally combining English and Chinese in a single CoT response. Furthermore, the rule-based RL training stage of DeepSeek-R1-Zero is narrowly focused on reasoning tasks, resulting in limited performance in broader areas such as writing and open-domain question answering. To address these challenges, we introduce DeepSeek-R1, a model trained through a multistage learning framework that integrates rejection sampling, RL and supervised fine-tuning, detailed in the ‘DeepSeek-R1’ section. This training pipeline enables DeepSeek-R1 to inherit the reasoning capabilities of its predecessor, DeepSeek-R1-Zero, while aligning model behaviour with human preferences through further non-reasoning data.

To enable broader access to powerful AI at a lower energy cost, we have distilled several smaller models and made them publicly available. These distilled models exhibit strong reasoning capabilities, surpassing the performance of their original instruction-tuned counterparts. We believe that these instruction-tuned versions will also greatly contribute to the research community by providing a valuable resource for understanding the mechanisms underlying long CoT reasoning models and for promoting the development of more powerful reasoning models. We release DeepSeek-R1-Zero, DeepSeek-R1, data samples and distilled models to the public as described in the ‘Code availability’ section.

A list of authors and their affiliations appears at the end of the paper.

Agentic RL (2026)

The Landscape of Agentic Reinforcement Learning for LLMs: A Survey

Guibin Zhang^{3†} Hejia Geng^{1†} Xiaohang Yu^{8†} Zhenfei Yin^{1*} Zaibin Zhang^{9,1}
Zelin Tan^{7,2} Heng Zhou^{7,2} Zhongzhi Li¹⁰ Xiangyuan Xue^{11,2} Yijiang Li¹³
Yifan Zhou¹² Yang Chen² Chen Zhang⁷ Yutao Fan² Zihu Wang¹⁴ Songtao Huang^{6,2}
Piedrahita-Velez, Francisco⁵ Yue Liao³ Hongru Wang¹¹ Mengyue Yang¹⁵
Heng Ji⁴ Jun Wang⁶ Shuicheng Yan³ Philip Torr¹ Lei Bai^{2*}

¹University of Oxford ²Shanghai AI Laboratory ³National University of Singapore
⁴University of Illinois Urbana-Champaign ⁵Brown University ⁶University College London
⁷University of Science and Technology of China ⁸Imperial College London
⁹Dalian University of Technology ¹⁰Chinese Academy of Sciences
¹¹The Chinese University of Hong Kong ¹²University of Georgia
¹³University of California, San Diego ¹⁴University of California, Santa Barbara
¹⁵University of Bristol

[†] *Equal contribution,* ^{*} *Corresponding Author*

关于大模型与强化学习的讨论与观点

Richard Sutton “The Bitter Lesson”

在人工智能的历史长河中，唯一被证明持续有效的方法，是利用无尽的算力(Computation)进行通用的搜索(Search)和学习(Learning)，而不是依赖人类手工编码的领域知识(Domain Knowledge)

- 正面：大模型充分利用了人类的数据与算力进行学习
- 反面：语言本身是人类符号化的知识，动物智能并不依赖语言

关于大模型与强化学习的讨论与观点



【 Dwarkesh Patel播客 】 强化学习之父 Richard Sutton ， 大模型...

https://www.bilibili.com/video/BV1z4HczgE1m/?share_source=copy_web&vd_source=b19e968ea4fbdf0cb9dfd7fbc468280e

大家说 LLM 很厉害，能写 code、拿 IMO/IOI 金牌。

Richard Sutton 的回答是：如果你觉得这很难，那你就这么觉得吧，我不这么觉得。我觉得**打造出一只松鼠的智能，才是真正难的问题**。一旦你能 build 松鼠的智能，让它在真实世界活下去，有自己的 goal、有 intrinsic reward、知道饥饿、有 emotion 和社群活动，那么写代码、上月球这些事都是再容易不过的。

如果抛下人类的自大，打造松鼠的智能其实是一个更难的问题。人类的智能不只是语言模型，有很多智能不能通过语言本身所决定。

站在宇宙和造物主的视角，重新造出一只松鼠，要比人类文明在 5.3 亿年进化史最后 8 秒创造的东西要伟大得多。

关于大模型与强化学习的讨论与观点

DeepMind强化学习掌门人David Silver离职创业！Alpha系列AI缔造者，哈萨比斯左膀右臂

关注前沿科技 量子位 2026年1月31日 09:34 北京 48人



DeepMind强化学习专家David Silver关于The Era of Experience论...

https://www.bilibili.com/video/BV19SdiY6EsG/?share_source=copy_web&vd_source=b19e968ea4fbdf0cb9dfd7fbc468280e

要做“能永无止境学习”的超级智能

Silver为什么要出来单干？

据知情人士透露，他的动机是希望回归“解决AI领域最难题所带来的敬畏与奇迹”，并将超级智能视为当前最大的未解挑战。

构建一个能够自我发现所有知识基础、永无止境学习的超级智能。

他倡导AI进入一个全新的“经验时代”（Age of Experience），即AI系统通过强化学习从经验中自我学习，从而发现人类未知的新事物。

这一理念的经典例证，就是2016年AlphaGo与李世石比赛中著名的第37手棋，当时所有人类专家都认为这一步是失误，但事后证明它是AlphaGo获胜的关键。

Silver认为，要实现真正的超级智能，AI必须摆脱对人类知识和直觉的依赖，从第一性原理^Q出发进行学习。

关于大模型与强化学习的讨论与观点



强化学习的形式化定义

强化学习的关键要素

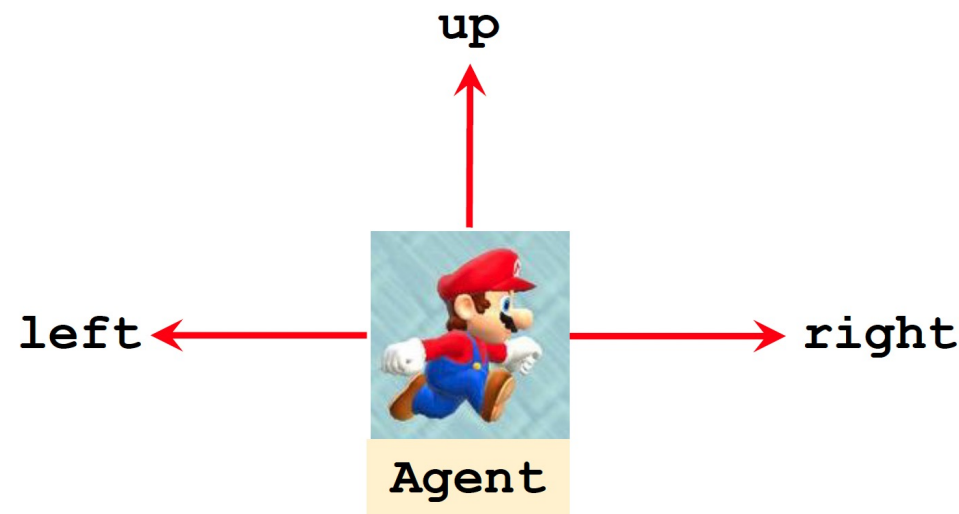
- 状态(State) : $s \in S$ 是对当前环境的描述

- 围棋任务中，当前的局面
- 超级马里奥中，当前的游戏画面
- 网格世界中，当前的位置和状态
- ...



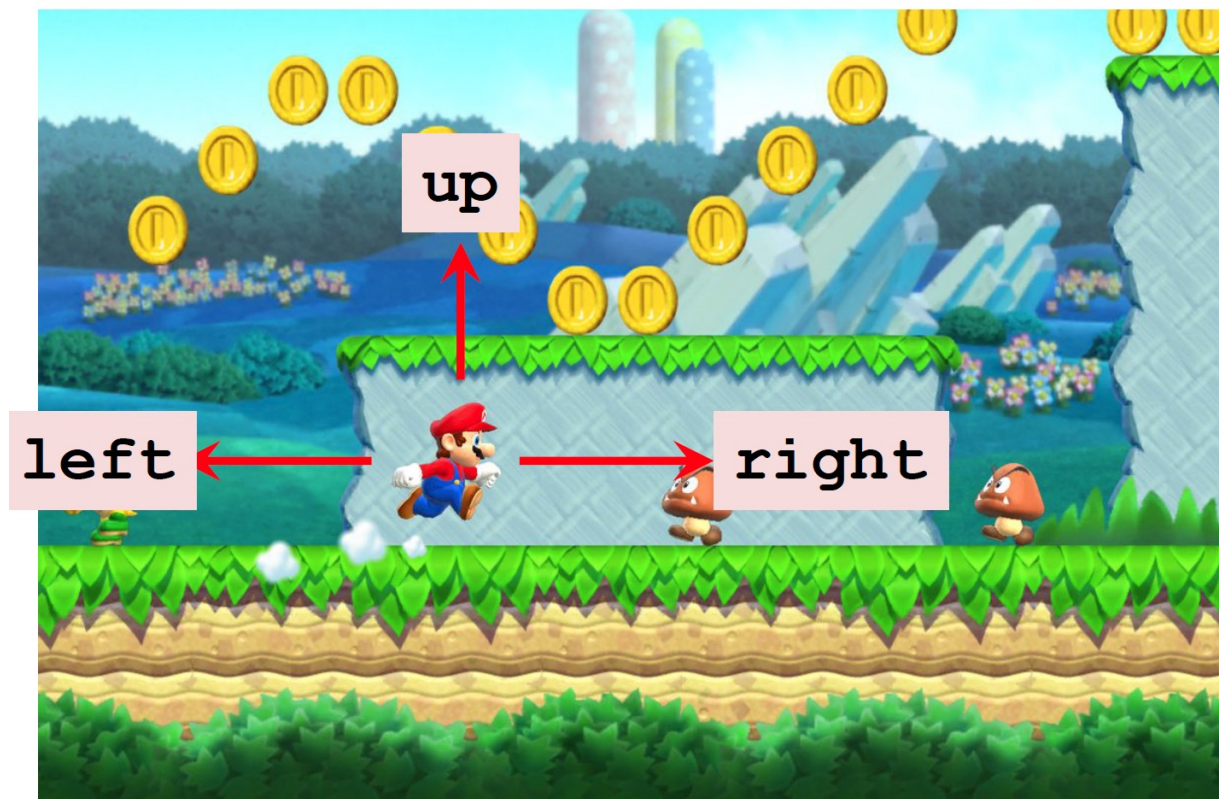
- 动作(Action): $a \in A$ 对智能体采取的行动

- 围棋任务中，选择一个位置落子
- 超级马里奥中，向左、向右、向上
- 网格世界中，选择一个方向移动
- ...



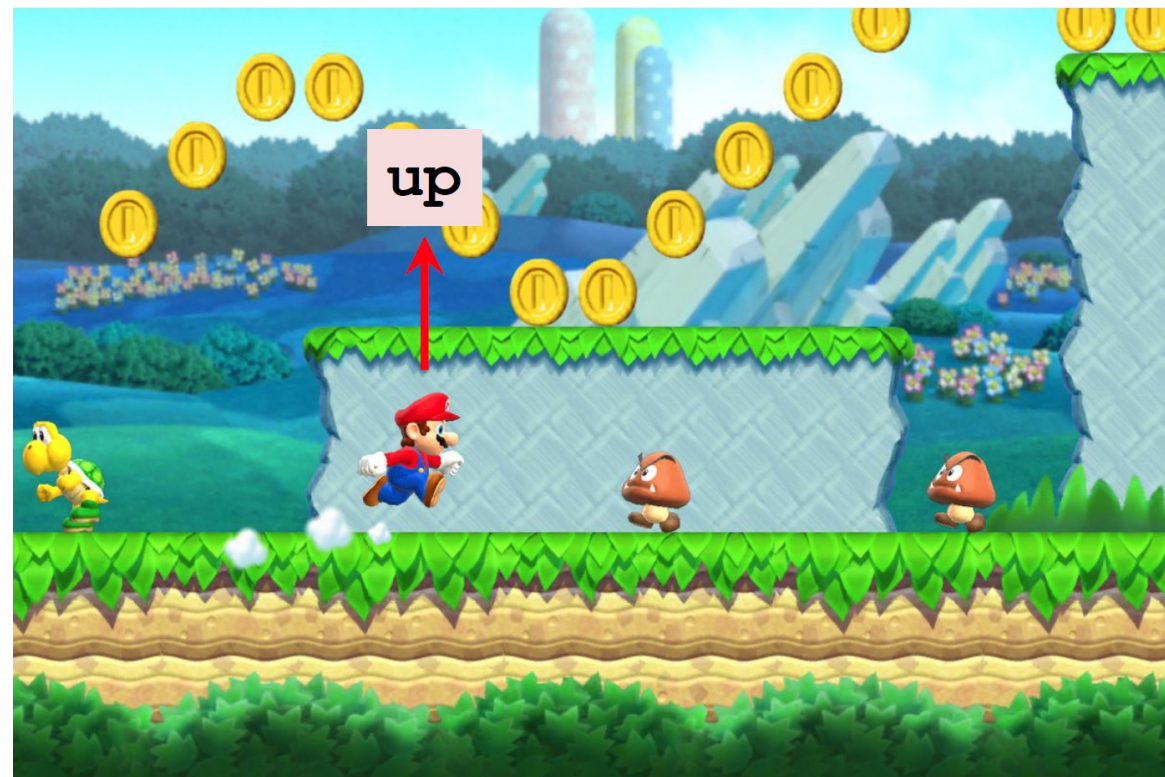
强化学习的关键要素

- 策略(Policy) π : 智能体如何根据观察到的状态做决策, 即从动作空间中选择一个动作
- 强化学习最终学得的就是一个策略函数
- 确定性策略: $\pi: S \rightarrow A$
- 随机性策略: $\pi: S \times A \rightarrow \mathbb{R}$



强化学习的关键要素

- 状态转移(state transition) $P: S \times A \times S \rightarrow \mathbb{R}$
- 智能体从当前 t 时刻状态 s_t 转移到下一个时刻的状态 s_{t+1}

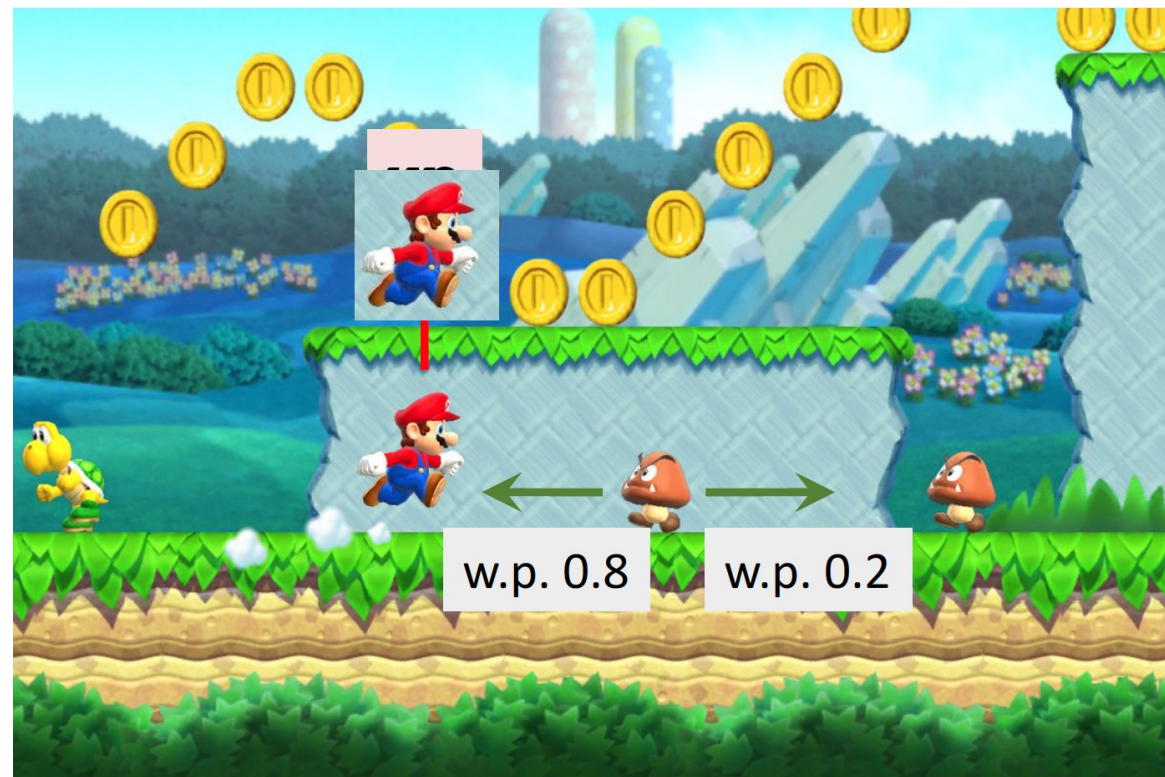


强化学习的关键要素

- 状态转移(state transition) $P: S \times A \times S \rightarrow \mathbb{R}$
- 智能体从当前 t 时刻状态 s_t 转移到下一个时刻的状态 s_{t+1}



考虑一般情况：非确定性环境
(Non-Deterministic Environment)



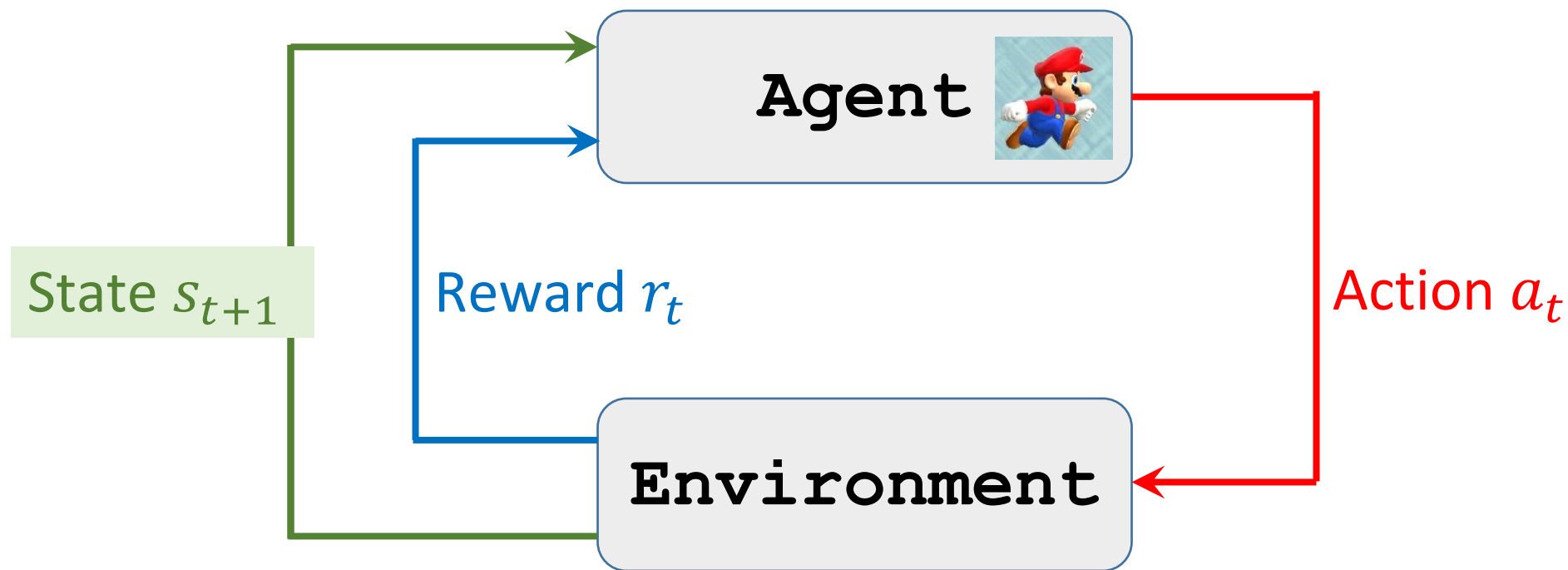
强化学习的关键要素

- 奖励(Reward) $R: S \times A \times S \mapsto \mathbb{R}$ (或 $R: S \times S \rightarrow \mathbb{R}$)
- 智能体执行一个动作之后, 环境返回给智能体一个数值
- 通常需要人来定义
- 奖励函数的好坏影响强化学习的结果



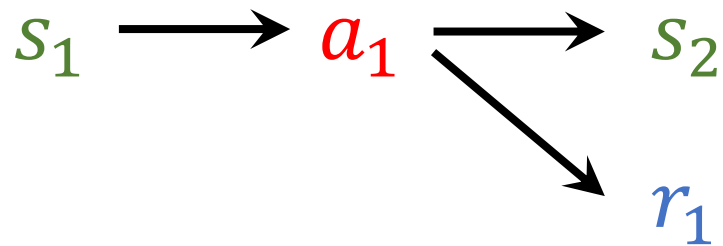
- 收集硬币: $R=+1$
- 被板栗仔撞到: $R=-1000$
- 通关: $R=+1000$
- 无事发生: $R=0$

强化学习的交互过程



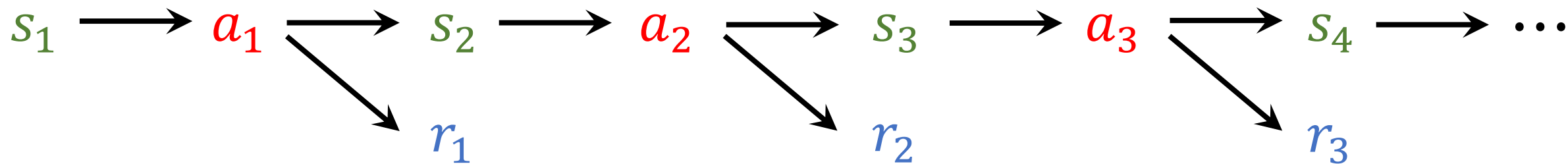
强化学习的交互过程

- 观察到状态 s_t ，根据策略 $\pi(s_t)$ 选择动作 a_t
- 执行动作 a_t ，状态转移至 s_{t+1} ，获得奖励 r_t



强化学习的交互过程

- 观察到状态 s_t ，根据策略 $\pi(s_t)$ 选择动作 a_t
- 执行动作 a_t ，状态转移至 s_{t+1} ，获得奖励 r_t



- 轨迹(trajecory): 智能体观测到的所有状态、动作、奖励的序列

$s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_n, a_n, r_n$

什么是好的策略

- 回报(return): 从当前时刻开始到本回合结束的所有奖励总和

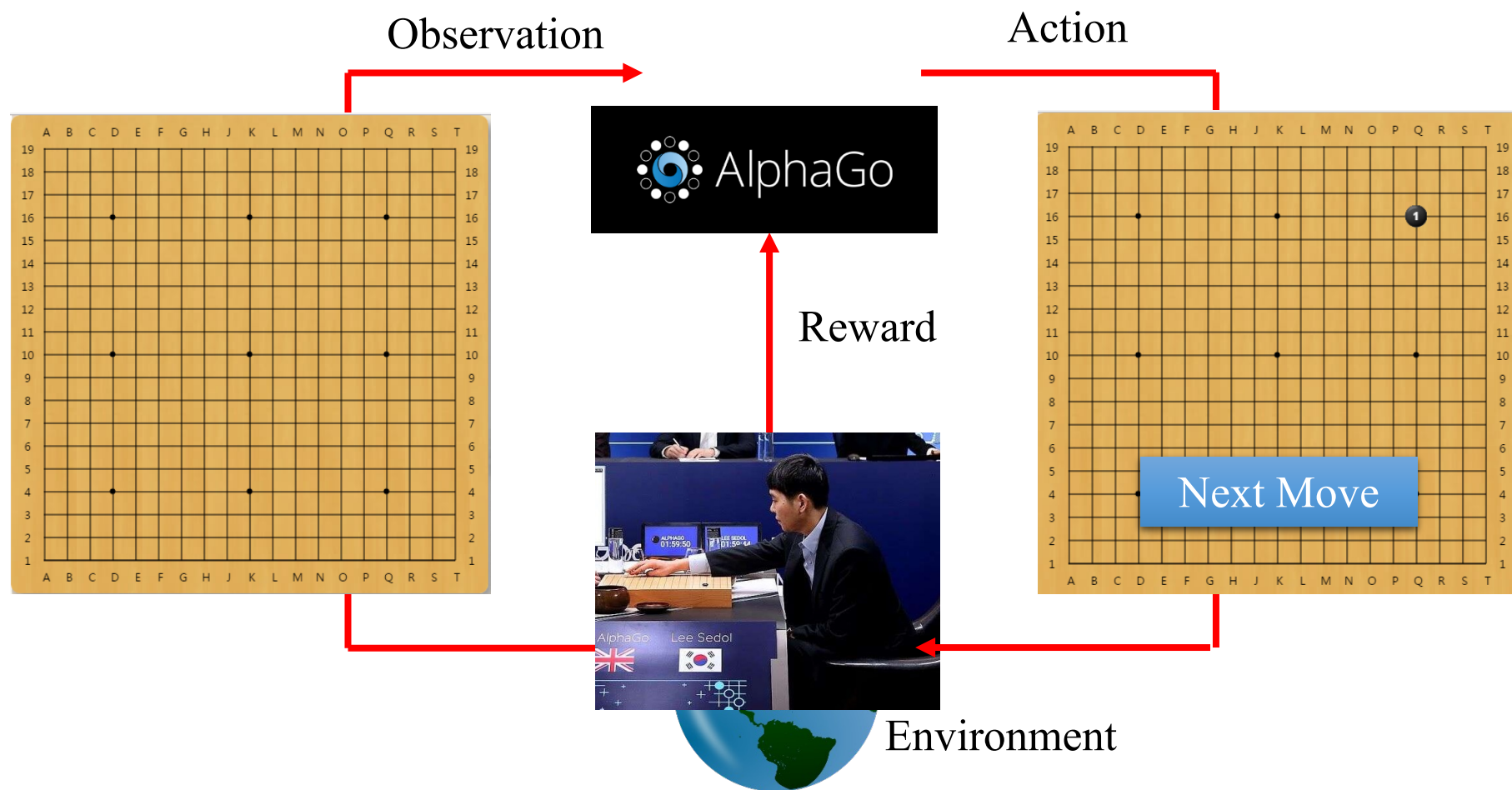
$$G_t = R_t + R_{t+1} + R_{t+2} + R_{t+3} + \dots$$

- 强化学习的目标是最大化回报，即累计奖励，而不是最大化当前奖励（好比下棋的时候目标是赢得比赛，而不是吃掉对方的棋子）
- 通常考虑折扣汇报： $\gamma \in [0,1]$ 是未来奖励的折扣因子，使得和未来奖励相比起来智能体更重视即时奖励

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \dots$$

- 以金融为例，今天的\$1比明天的\$1更有价值

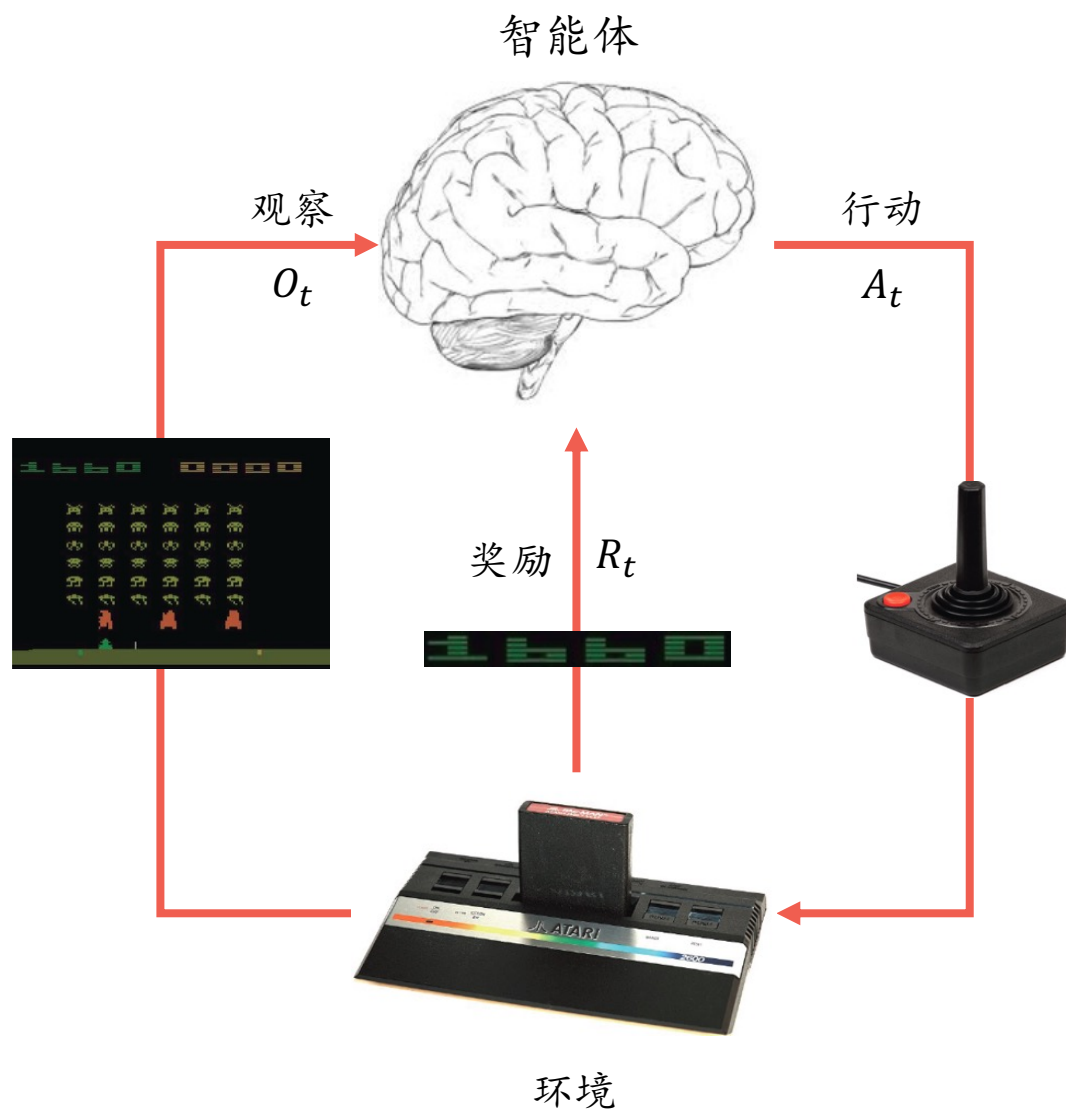
例子：围棋



例子：围棋



例子：游戏

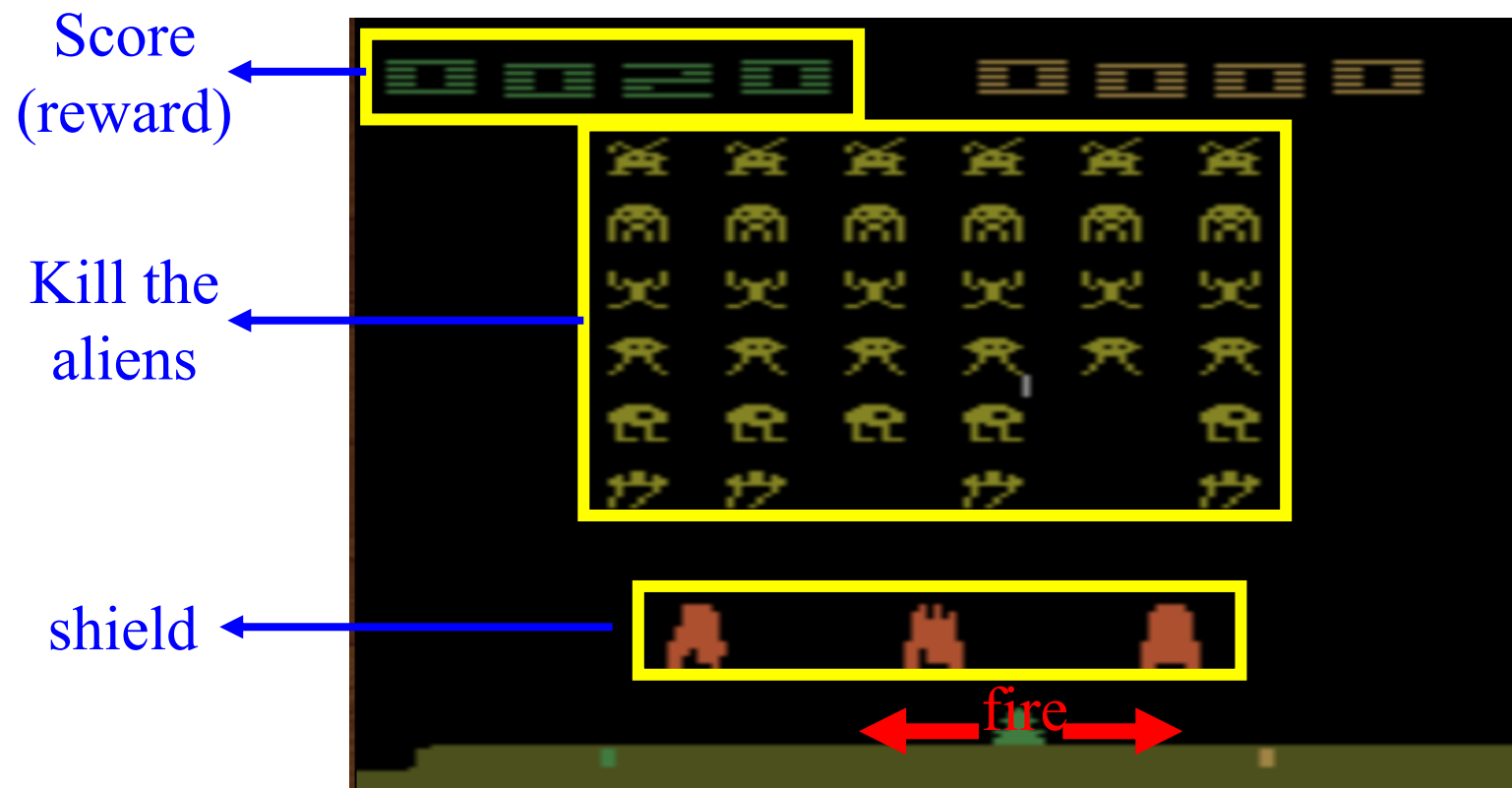


- 从交互游戏中进行学习
- 在操纵杆上选择行动并查看分数和像素画面

例子：游戏

- Space invader

Termination: all the aliens are killed, or your spaceship is destroyed.



例子：游戏

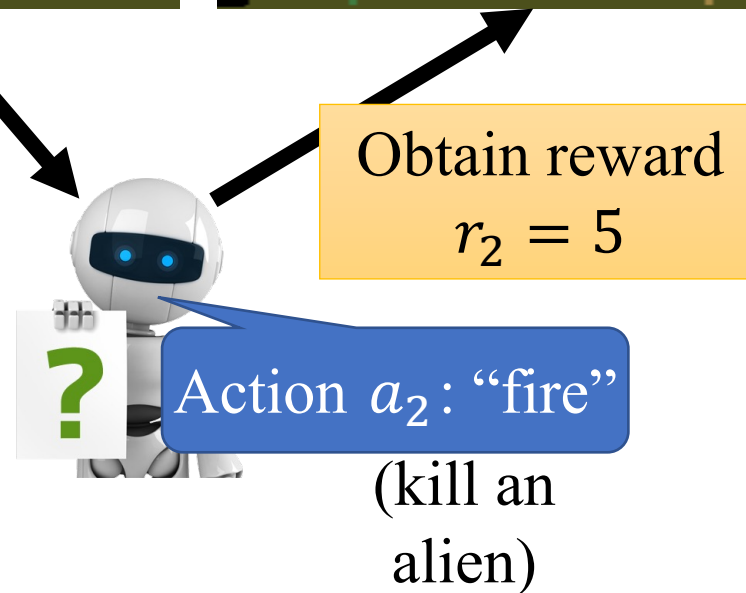
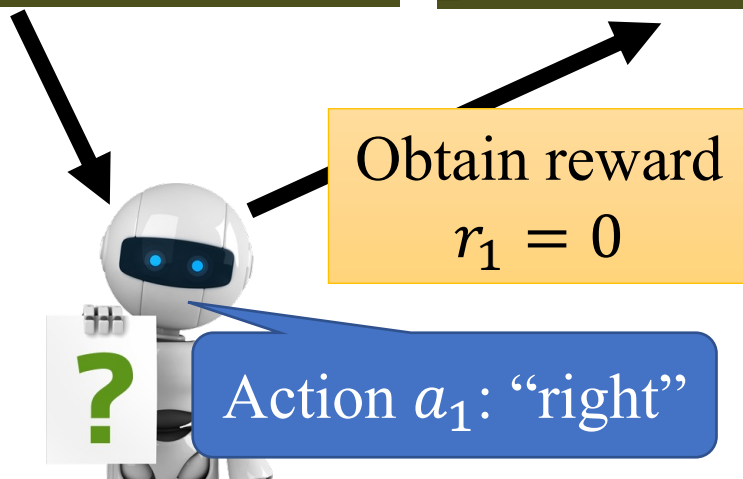
Start with
observation s_1



Observation s_2



Observation s_3



例子：游戏

Start with
observation s_1



Observation s_2



Observation s_3



After many turns



Action a_T

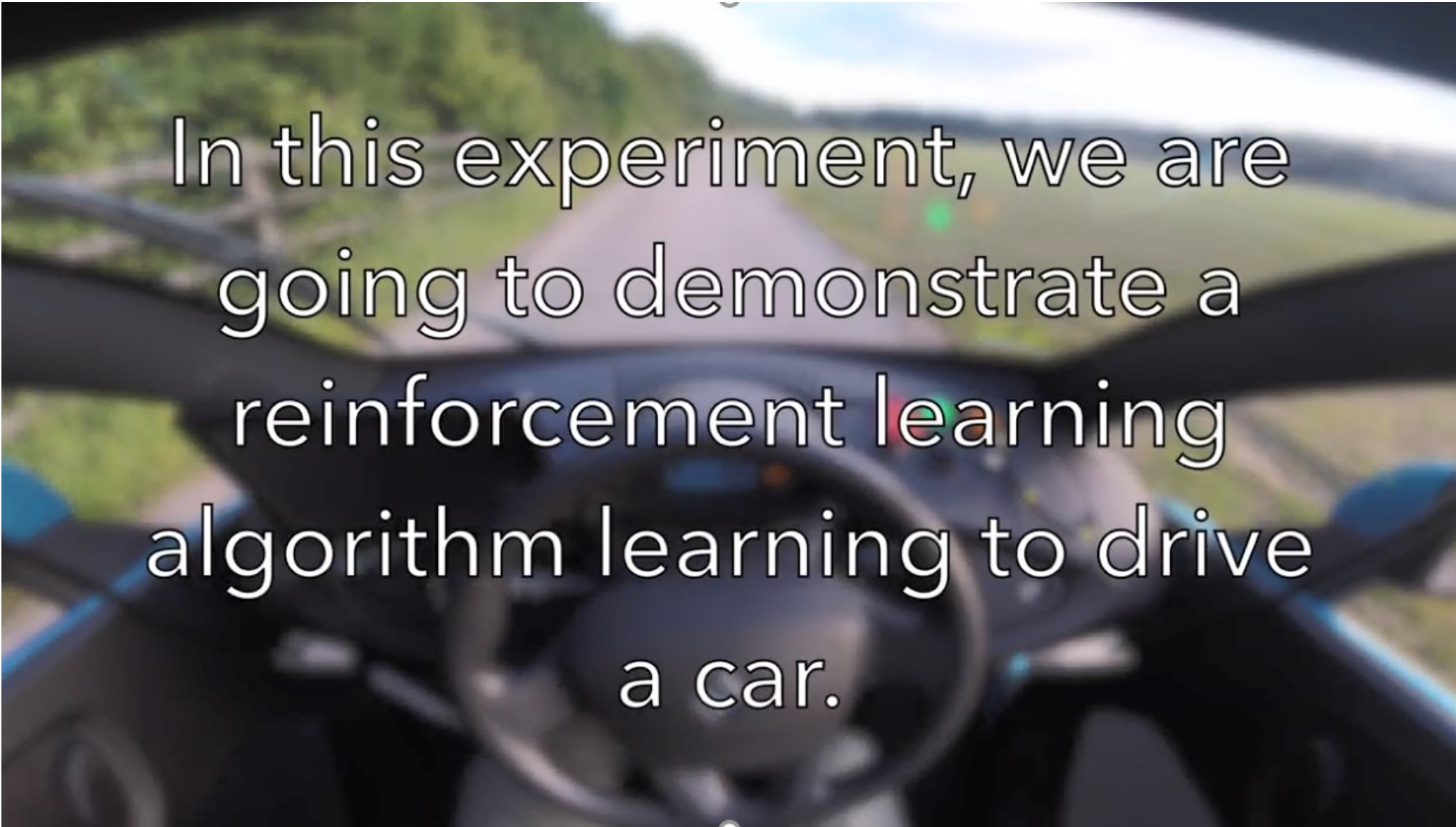
Obtain reward r_T

This is an *episode*.

应用案例：机械臂操作



应用案例：无人驾驶小车



In this experiment, we are going to demonstrate a reinforcement learning algorithm learning to drive a car.

应用案例：黑神话

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

```

2025-06-09 18:59:25 - --- step start
2025-06-09 18:59:25 - action 轻击
2025-06-09 18:59:26 - obs: hp 100, boss hp 73
2025-06-09 18:59:26 - reward 0.510 win False
2025-06-09 18:59:26 - --- step end
2025-06-09 18:59:26 - --- step start
2025-06-09 18:59:26 - action 轻击
2025-06-09 18:59:26 - obs: hp 100, boss hp 73
2025-06-09 18:59:26 - reward 0.510 win False
2025-06-09 18:59:26 - --- step end
2025-06-09 18:59:26 - --- step start
2025-06-09 18:59:26 - action 轻击
2025-06-09 18:59:26 - obs: hp 100, boss hp 73
2025-06-09 18:59:26 - reward 0.510 win False
2025-06-09 18:59:26 - --- step end
2025-06-09 18:59:26 - --- step start
2025-06-09 18:59:26 - action 定身法
2025-06-09 18:59:27 - obs: hp 100, boss hp 72
2025-06-09 18:59:27 - reward 0.010 win False
2025-06-09 18:59:27 - --- step end
2025-06-09 18:59:27 - --- step start
2025-06-09 18:59:27 - action 轻击
2025-06-09 18:59:27 - obs: hp 100, boss hp 72
2025-06-09 18:59:27 - reward 0.510 win False
2025-06-09 18:59:27 - --- step end
2025-06-09 18:59:27 - --- step start
2025-06-09 18:59:27 - action 轻击
2025-06-09 18:59:27 - obs: hp 100, boss hp 72
2025-06-09 18:59:27 - reward 0.010 win False
2025-06-09 18:59:27 - --- step end
2025-06-09 18:59:27 - --- step start
2025-06-09 18:59:28 - action 轻击
2025-06-09 18:59:28 - obs: hp 100, boss hp 72
2025-06-09 18:59:28 - reward 0.510 win False
2025-06-09 18:59:28 - --- step end
2025-06-09 18:59:28 - --- step start
2025-06-09 18:59:28 - action 轻击
2025-06-09 18:59:28 - obs: hp 100, boss hp 10
2025-06-09 18:59:28 - reward 6.000 win False
2025-06-09 18:59:28 - --- step end
2025-06-09 18:59:28 - --- step start
2025-06-09 18:59:28 - action 轻击
2025-06-09 18:59:29 - obs: hp 100, boss hp 10
2025-06-09 18:59:29 - reward 0.510 win False
2025-06-09 18:59:29 - --- step end
2025-06-09 18:59:29 - --- step start
2025-06-09 18:59:29 - action 轻击
2025-06-09 18:59:29 - game over, hp 100, boss hp 246
2025-06-09 18:59:29 - obs: hp 100, boss hp 246
2025-06-09 18:59:29 - reward 3.000 win True
2025-06-09 18:59:29 - episode 7 end, loss 1, win 6
2025-06-09 18:59:29 - --- step end
2025-06-09 18:59:29 - reset, hp: 100, boss hp: 246
2025-06-09 18:59:29 - restart game...

```





rerender 

幽魂 2 - 5 天命人

目标检测: 20.1 ms

深度估计: 48.9 26.6 ms

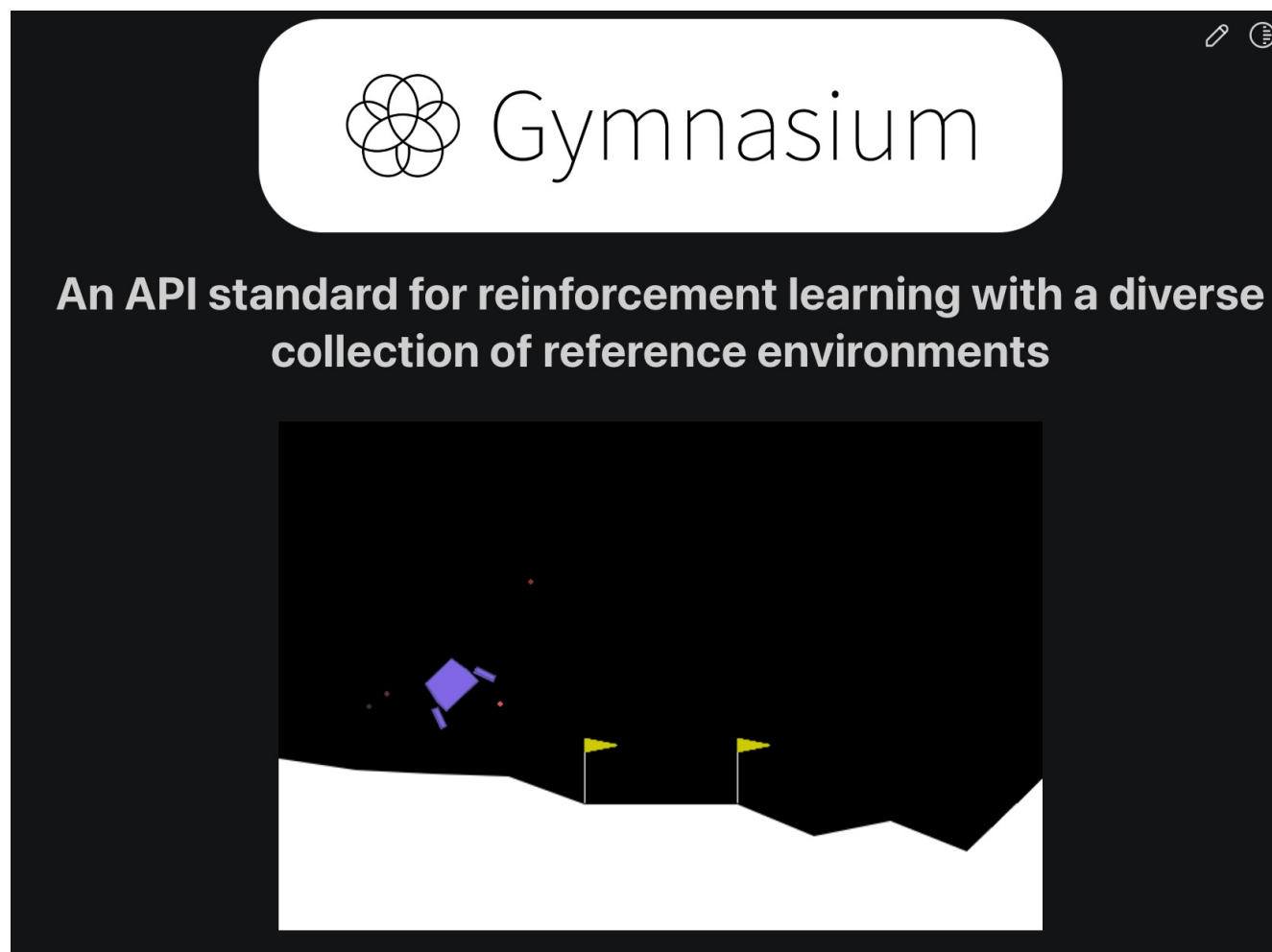


扩展材料

- Sutton's book:
 - <http://incompleteideas.net/sutton/book/the-book.html>
- David Silver's Lecture:
 - <https://www.davidsilver.uk/teaching/>
- Sergey Levine
 - <https://rail.eecs.berkeley.edu/deeprlcourse/>
- 动手学强化学习:
 - <https://hrl.boyuai.com/chapter/intro>

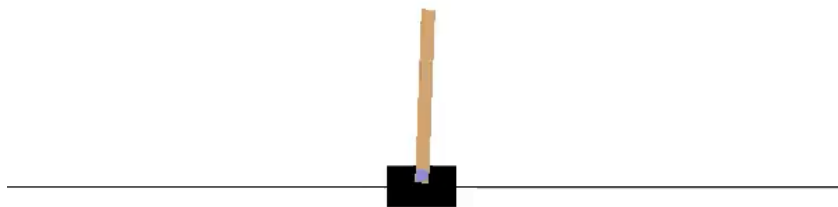
实验平台：OpenAI Gym

<https://gymnasium.farama.org/>



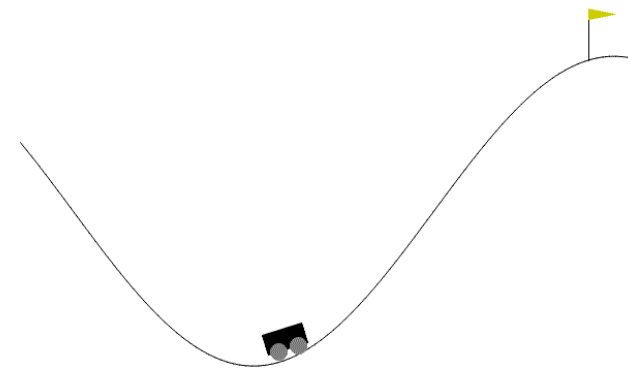
实验平台：OpenAI Gym

经典控制问题



Cart Pole

Num	Observation	Min	Max
0	Cart Position	-4.8	4.8
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -0.418 rad (-24°)	~ 0.418 rad (24°)
3	Pole Angular Velocity	-Inf	Inf

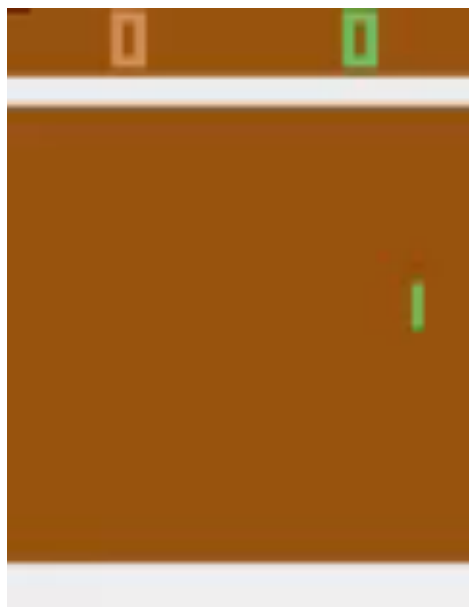


Mountain Car

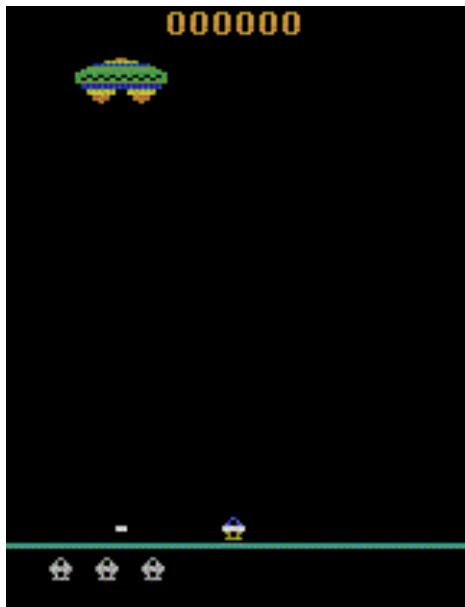
Num	Action
0	Push cart to the left
1	Push cart to the right

实验平台：OpenAI Gym

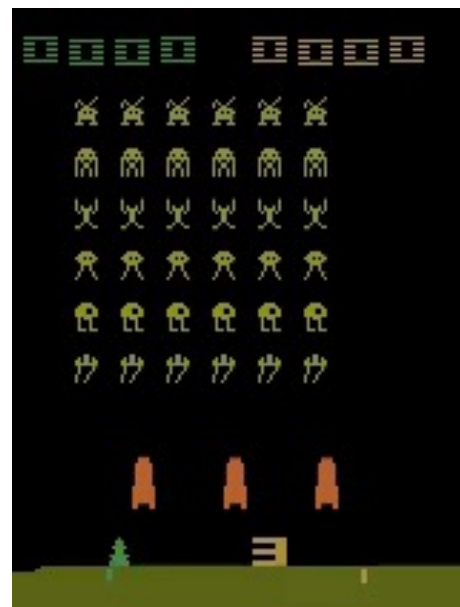
Atari游戏



Pong



Assault



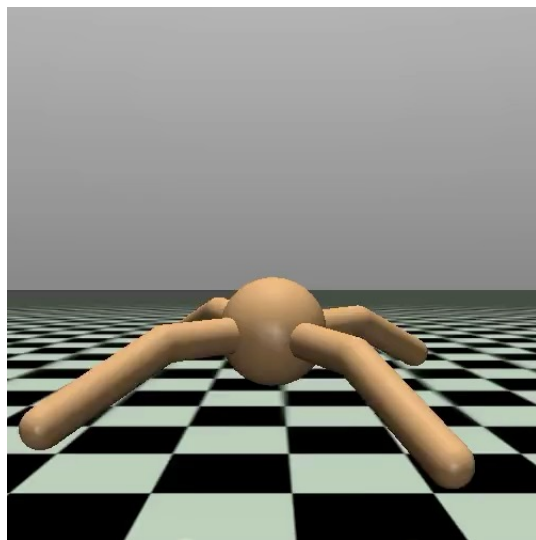
Space Invader



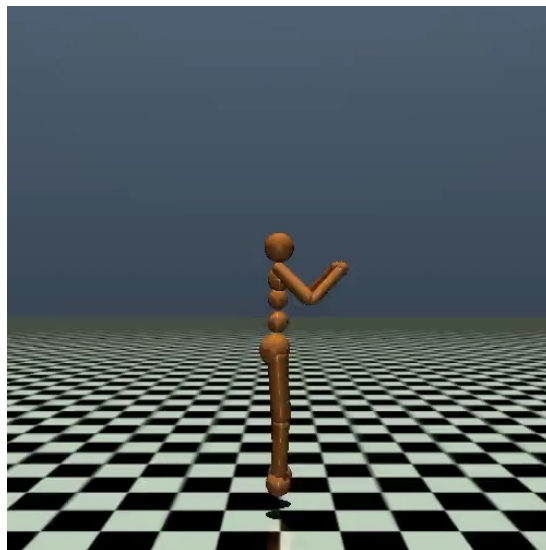
Breakout

实验平台：OpenAI Gym

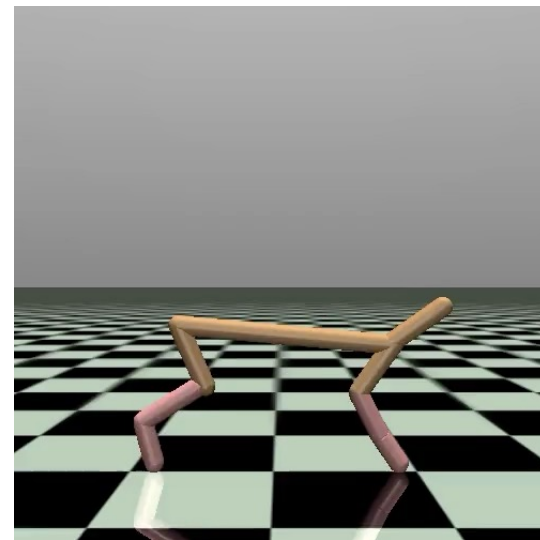
机器人的连续控制问题



Ant

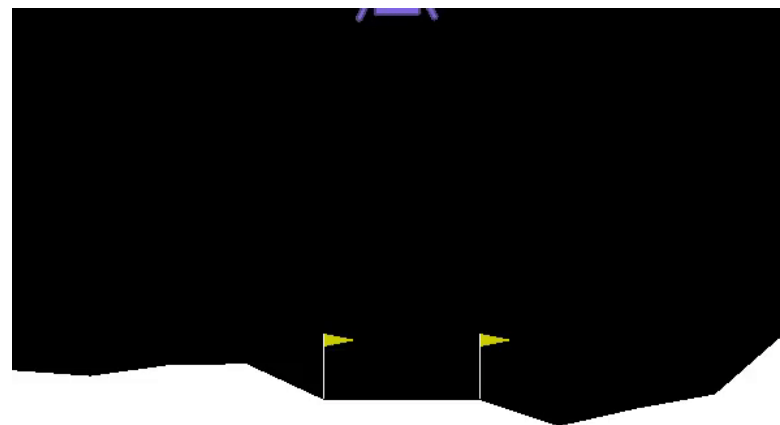
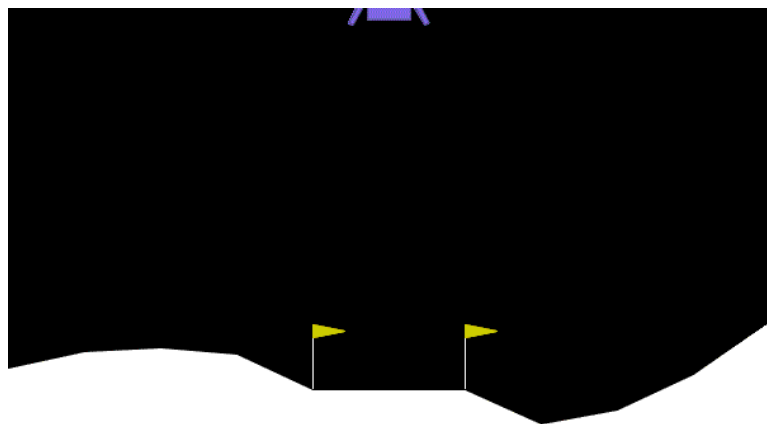


Humanoid



Half Cheetah

实验平台：OpenAI Gym



```
import gym
env = gym.make("LunarLander-v2", render_mode="human")
env.action_space.seed(42)

observation, info = env.reset(seed=42)

for _ in range(1000):
    observation, reward, terminated, truncated, info = env.step(env.action_space.sample())

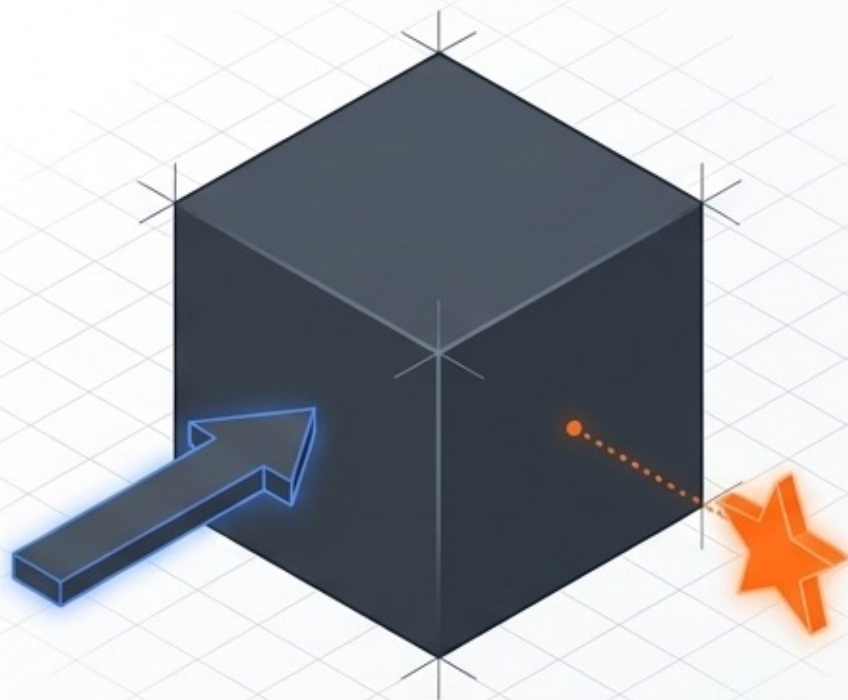
    if terminated or truncated:
        observation, info = env.reset()

env.close()
```

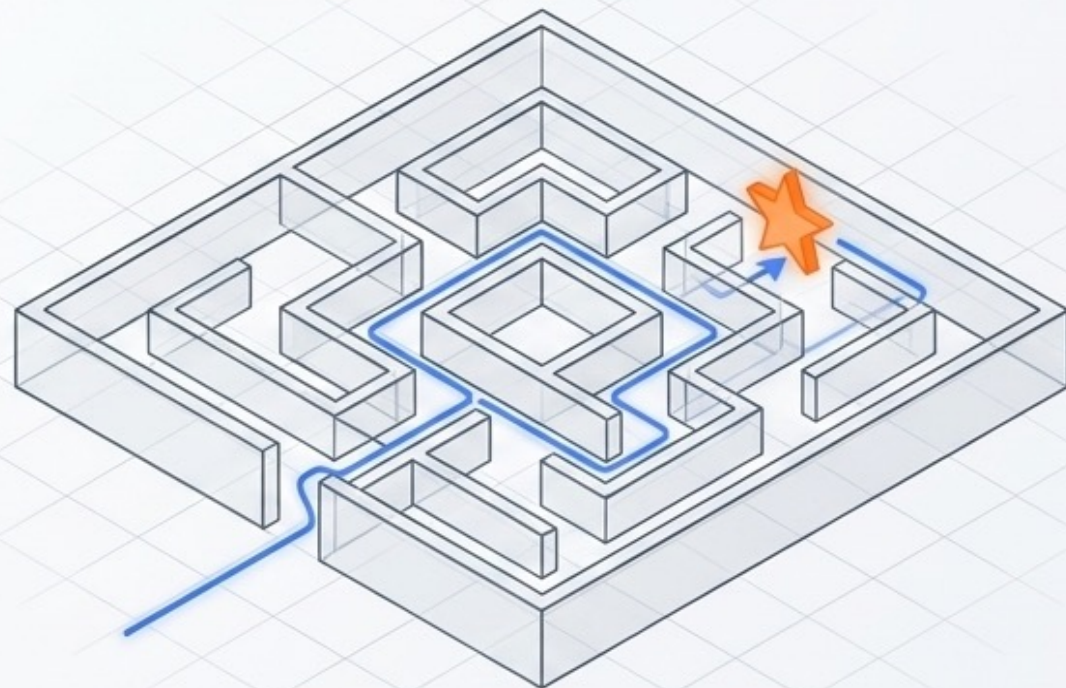
有模型学习 (Model-based Learning)

有模型学习：世界的运行规则完全透明

假设：状态空间 S 、动作空间 A 、奖励函数 R 、转移概率 P 均已知

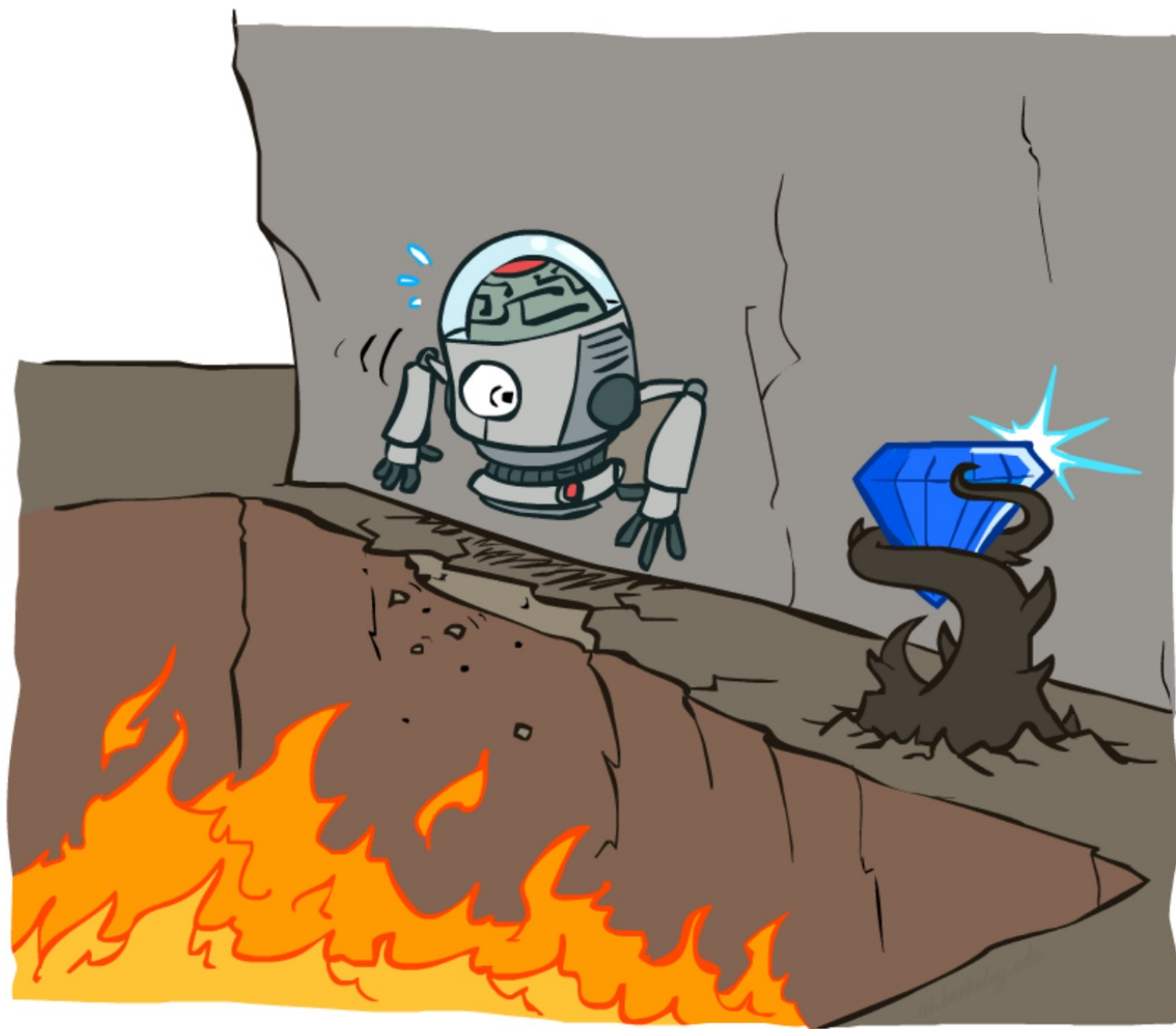


无模型(Model-Free)
规则未知，全靠探索

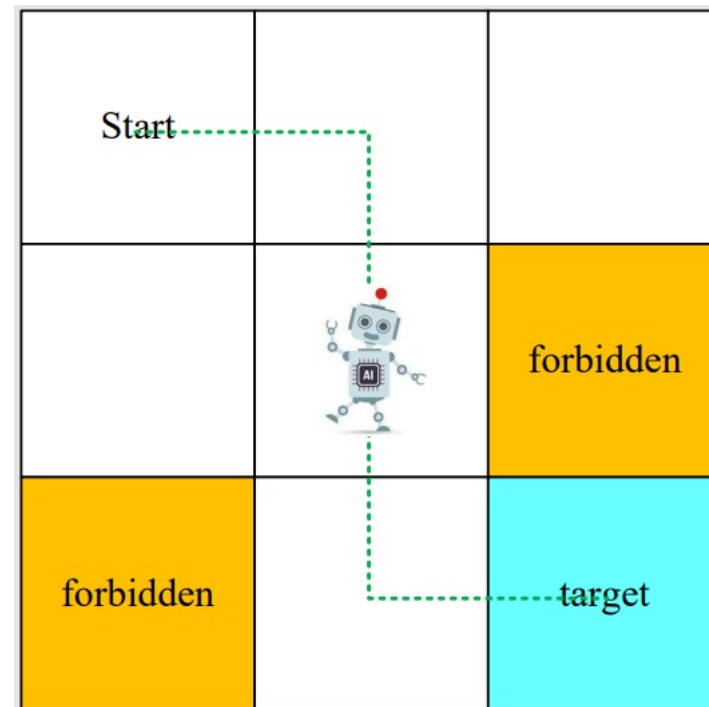
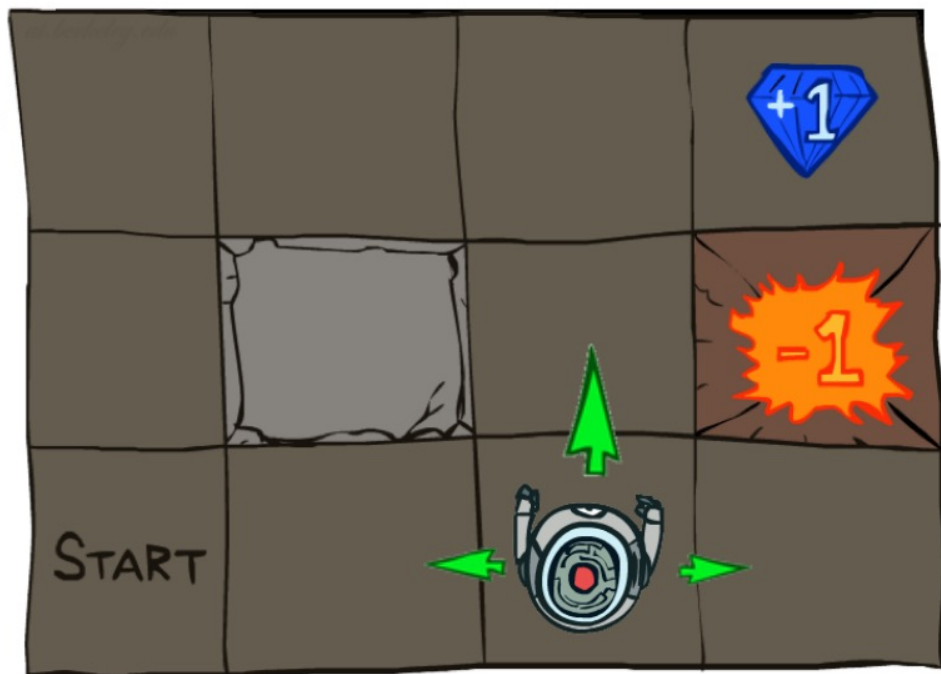


有模型学习(Model-based)
规则已知，只需数学计算

一个例子：网格世界(Grid World)



网格世界(Grid World)



- 环境：由若干网格单元组成的网格世界
- 可通行单元 / 禁止进入单元 / 目标单元 / 边界
- 任务：给定任意起始位置，寻找一条通往目标的“较好”路径
- 如何定义“较好”？

网格世界(Grid World)

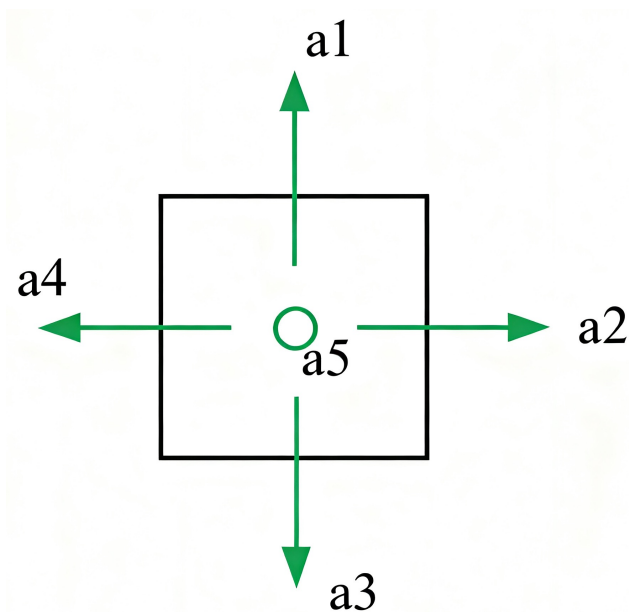
- **状态 (State)**: agent在环境中的状况
- 在grid-world中, 状态就是agent所在的位置
- 共有 9 个可能的位置, 那么就有 9 个状态 $S_1, S_2, \dots, S_9$

s1	s2	s3
s4	s5	s6
s7	s8	s9

状态空间(State Space): 所有状态的集合 $\{S_1, S_2, \dots, S_9\}$

网格世界(Grid World)

- **动作(Action)**: agent在每个状态可选的行动
- 向上、向右、向下、向左、保持不动



s1	s2	s3
s4	s5	s6
s7	s8	s9

动作空间(Action Space): 所有可行动作的集合

网格世界(Grid World)

s1	s2	s3
s4	s5	s6
s7	s8	s9

禁止区域 (Forbidden area) :

在状态 S_5 下，如果我们选择动作 a_2 ，下一个状态会是什么？存在两种情形：

- 情形 1（禁止区可进入但有惩罚）
- 情形 2（禁止区不可进入，例如被墙包围）

考虑第一种，让算法学会自己避免进入禁止区

网格世界(Grid World)

- 表格表示(Tabular Representation)

s1	s2	s3
s4	s5	s6
s7	s8	s9

	a_1 (upwards)	a_2 (rightwards)	a_3 (downwards)	a_4 (leftwards)	a_5 (unchanged)
s_1	s_1	s_2	s_4	s_1	s_1
s_2	s_2	s_3	s_5	s_1	s_2
s_3	s_3	s_3	s_6	s_2	s_3
s_4	s_1	s_5	s_7	s_4	s_4
s_5	s_2	s_6	s_8	s_4	s_5
s_6	s_3	s_6	s_9	s_5	s_6
s_7	s_4	s_8	s_7	s_7	s_7
s_8	s_5	s_9	s_8	s_7	s_8
s_9	s_6	s_9	s_9	s_8	s_9

网格世界(Grid World)

s1	s2	s3
s4	s5	s6
s7	s8	s9

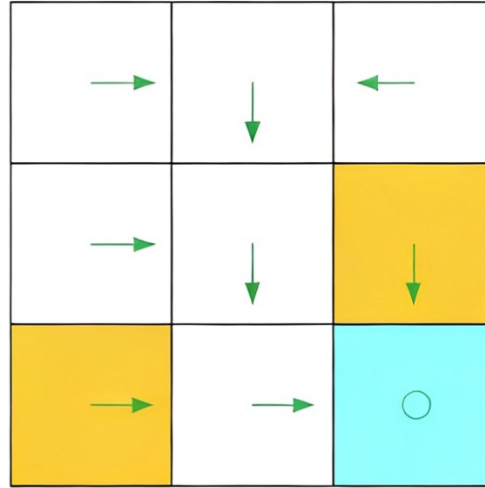
- 状态转移概率(State Transition Probability)
- Deterministic or Stochastic

$$P(S_2|S_1, a_2) = 1$$

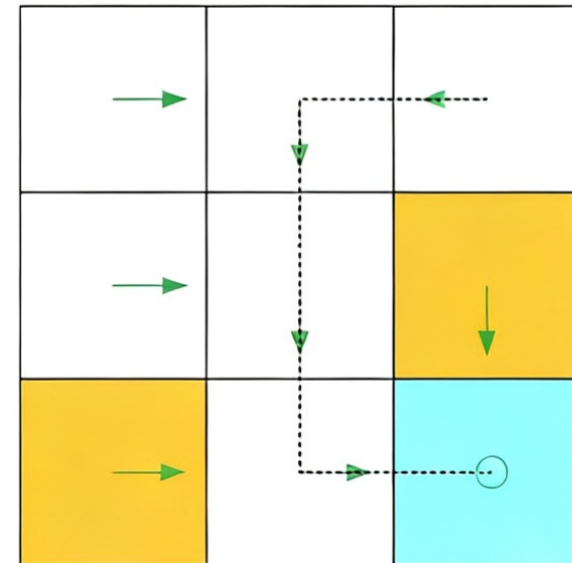
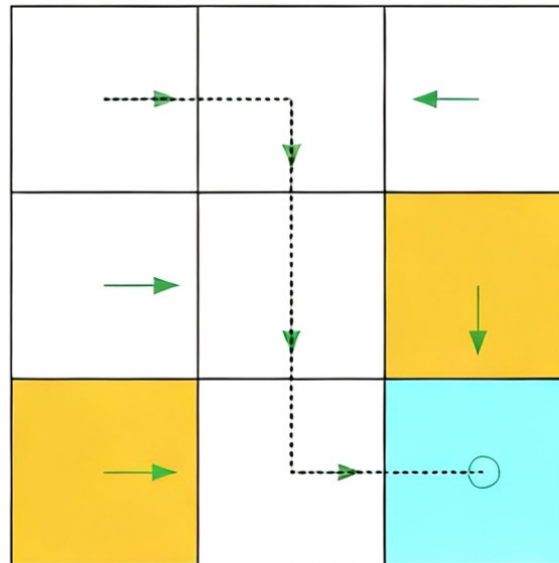
$$P(S_i|S_1, a_2) = 0, \forall i \neq 2$$

网格世界(Grid World)

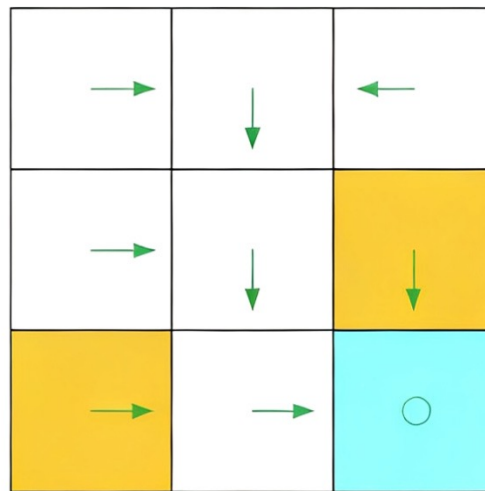
Policy: 告诉Agent每个状态下应该选择什么动作



基于Policy可以得到任意
起点出发的路径



网格世界(Grid World)



Deterministic Policy

$$\pi(a_1|s_1) = 0$$

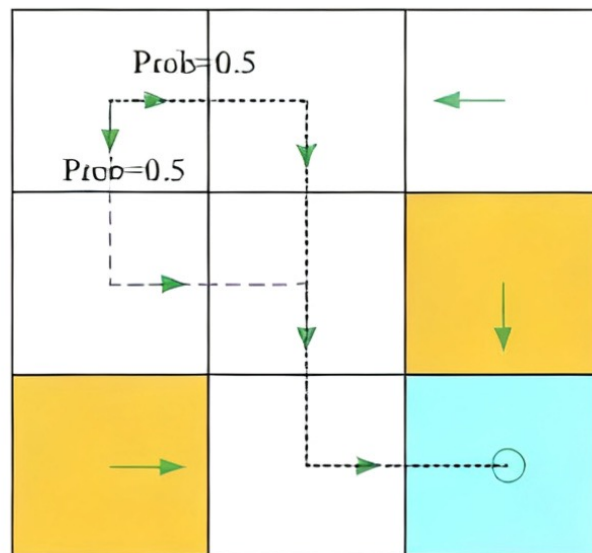
$$\pi(a_2|s_1) = 1$$

$$\pi(a_3|s_1) = 0$$

$$\pi(a_4|s_1) = 0$$

$$\pi(a_5|s_1) = 0$$

网格世界(Grid World)



Stochastic Policy

$$\pi(a_1|s_1) = 0$$

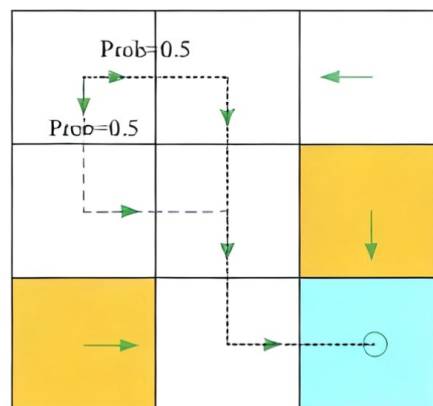
$$\pi(a_2|s_1) = 0.5$$

$$\pi(a_3|s_1) = 0.5$$

$$\pi(a_4|s_1) = 0$$

$$\pi(a_5|s_1) = 0$$

网格世界(Grid World)



策略的表格表示

	a_1 (upwards)	a_2 (rightwards)	a_3 (downwards)	a_4 (leftwards)	a_5 (unchanged)
s_1	0	0.5	0.5	0	0
s_2	0	0	1	0	0
s_3	0	0	0	1	0
s_4	0	1	0	0	0
s_5	0	0	1	0	0
s_6	0	0	1	0	0
s_7	0	1	0	0	0
s_8	0	1	0	0	0
s_9	0	0	0	0	1

网格世界(Grid World)

如何设计奖励函数？

s1	s2	s3
s4	s5	s6
s7	s8	s9

在网格世界 (grid-world) 示例中，奖励函数的设计如下：

- 若Agent尝试越出边界，则奖励 $r_{bound} = -1$
- 若Agent尝试进入禁止单元，则奖励 $r_{forbid} = -1$
- 若Agent到达目标单元，则奖励 $r_{target} = 1$
- 其他情况下，奖励 $r = 0$

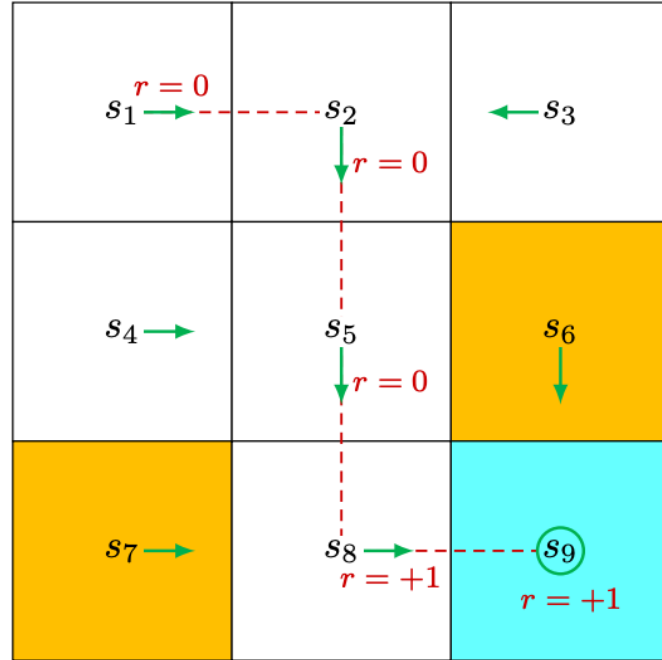
网格世界(Grid World)

s1	s2	s3
s4	s5	s6
s7	s8	s9

奖励的表格表示

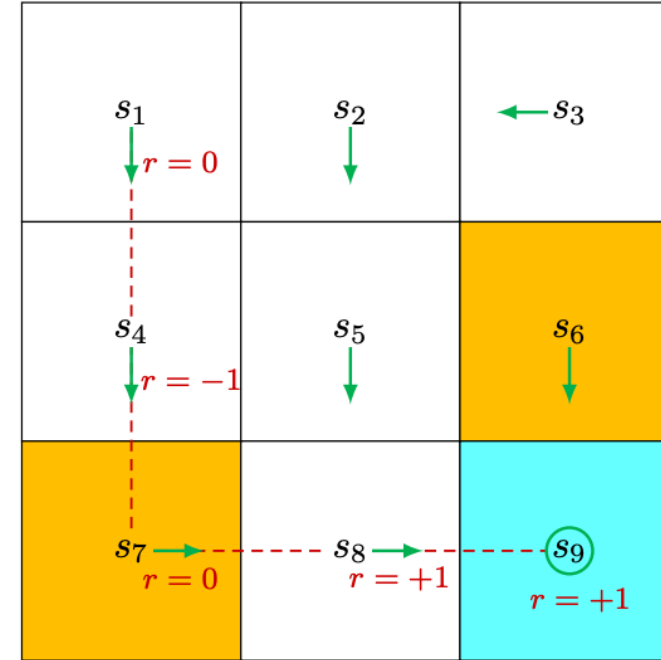
	a_1 (upward)	a_2 (rightward)	a_3 (downward)	a_4 (leftward)	a_5 (still)
s_1	r_{boundary}	0	0	r_{boundary}	0
s_2	r_{boundary}	0	0	0	0
s_3	r_{boundary}	r_{boundary}	$r_{\text{forbidden}}$	0	0
s_4	0	0	$r_{\text{forbidden}}$	r_{boundary}	0
s_5	0	$r_{\text{forbidden}}$	0	0	0
s_6	0	r_{boundary}	r_{target}	0	$r_{\text{forbidden}}$
s_7	0	0	r_{boundary}	r_{boundary}	$r_{\text{forbidden}}$
s_8	0	r_{target}	r_{boundary}	$r_{\text{forbidden}}$	0
s_9	$r_{\text{forbidden}}$	r_{boundary}	r_{boundary}	0	r_{target}

Trajectory and Return



$$s_1 \xrightarrow[r=0]{a_2} s_2 \xrightarrow[r=0]{a_3} s_5 \xrightarrow[r=0]{a_3} s_8 \xrightarrow[r=+1]{a_2} s_9$$

$$\text{return} = 0 + 0 + 0 + 1 = 1$$



$$s_1 \xrightarrow[r=0]{a_3} s_4 \xrightarrow[r=-1]{a_3} s_7 \xrightarrow[r=0]{a_2} s_8 \xrightarrow[r=+1]{a_2} s_9$$

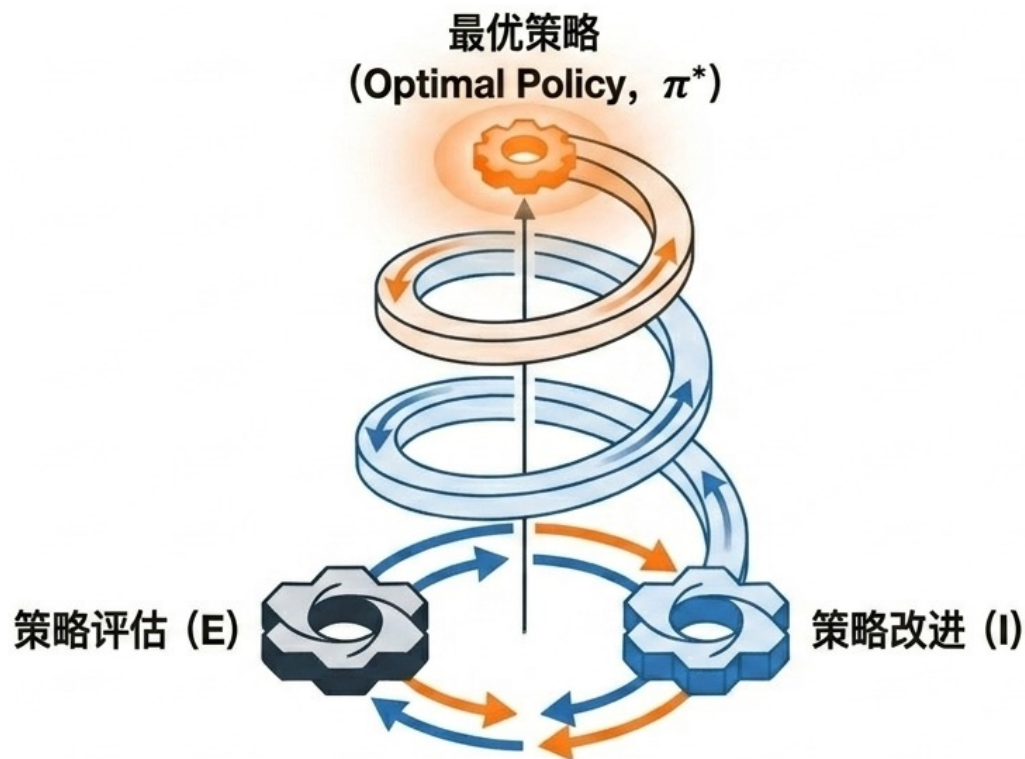
$$\text{return} = 0 - 1 + 0 + 1 = 0$$

有模型学习

- 目标：选择能够最大化累积奖励期望的策略 $\pi(s): S \rightarrow A$

$$\mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \dots]$$

- 三个步骤：
 - 策略评估
 - 策略改进
 - 策略迭代



策略评估：给定一个策略，如何评估好坏？

价值函数(Value Function)

- 给策略 π 定义价值函数，价值函数是状态（或状态-动作二元组）的函数，用来评估当前智能体在给定状态（获给定状态与动作）下有多好。这里的“好”是基于未来预期的回报（累计奖励）来定义的

- 状态值函数(State value function)

$$V^\pi(s_t) = \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots | s = s_t]$$

- 状态-动作值函数(State-Action value function)

$$Q^\pi(s_t, a_t) = \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots | s = s_t, a = a_t]$$

- 二者的关系：

$$V^\pi(s_t) = \sum_a \pi(s_t, a) \cdot Q^\pi(s_t, a)$$

贝尔曼与动态规划

1957年，贝尔曼提出**动态规划(Dynamic Programming)**理论

核心思想是将复杂的长期决策问题，分解为一系列简单的短期决策



贝尔曼方程(Bellman Equation)

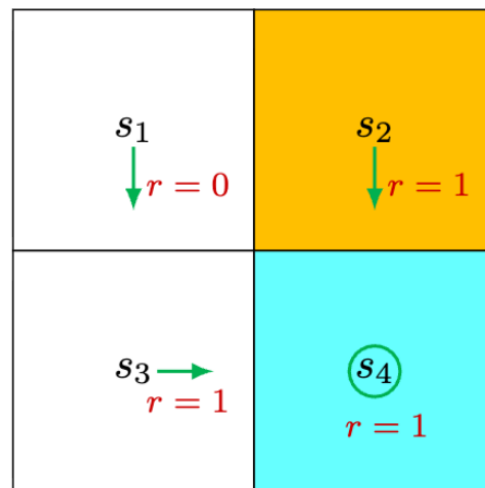
$$V^\pi(s) = \mathbb{E}[r_1 + \underbrace{\gamma r_2 + \gamma^2 r_3 + \cdots}_{\gamma V^\pi(s_{t+1})} | s_0 = s]$$

$$= \sum_a \pi(s, a) \sum_{s' \in \mathcal{S}} P_{sa}(s') (r_{sa}(s') + \gamma V^\pi(s'))$$

Bellman等式

$$Q^\pi(s, a) = \sum_{s' \in \mathcal{S}} P_{sa}(s') (r_{sa}(s') + \gamma V^\pi(s'))$$

贝尔曼方程示例



贝尔曼方程: $V^\pi(s_t) = \sum_a \pi(s, a) \sum_{s' \in S} P_{sa}(s') (r_{sa}(s') + \gamma V^\pi(s'))$ **General Case**

$$V^\pi(s_1) = 0 + \gamma V^\pi(s_3)$$

$$V^\pi(s_2) = 1 + \gamma V^\pi(s_4)$$

$$V^\pi(s_3) = 1 + \gamma V^\pi(s_4)$$

$$V^\pi(s_4) = 1 + \gamma V^\pi(s_4)$$

贝尔曼方程示例

如何求解？

$$V^\pi(s_1) = 0 + \gamma V^\pi(s_3)$$

$$V^\pi(s_2) = 1 + \gamma V^\pi(s_4)$$

$$V^\pi(s_3) = 1 + \gamma V^\pi(s_4)$$

$$V^\pi(s_4) = 1 + \gamma V^\pi(s_4)$$

$$V^\pi(s_4) = \frac{1}{1 - \gamma}$$

$$V^\pi(s_3) = \frac{1}{1 - \gamma}$$

$$V^\pi(s_2) = \frac{1}{1 - \gamma}$$

$$V^\pi(s_1) = \frac{\gamma}{1 - \gamma}$$

贝尔曼方程示例

令 $\gamma = 0.9$,

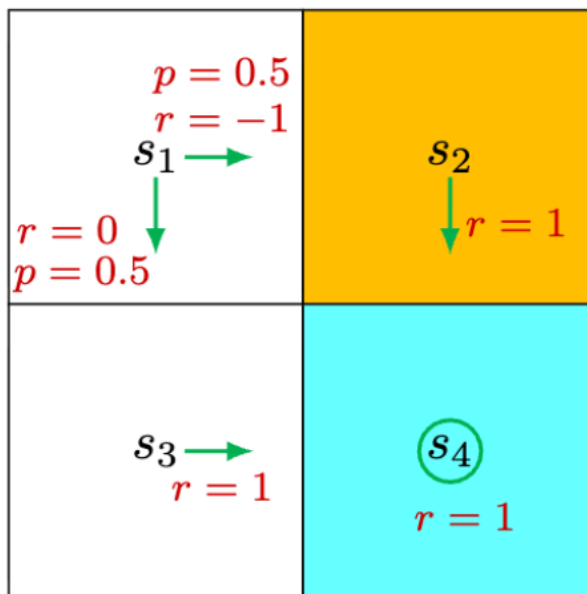
$$V^\pi(S_4) = \frac{1}{1 - 0.9} = 10$$

$$V^\pi(S_3) = \frac{1}{1 - 0.9} = 10$$

$$V^\pi(S_2) = \frac{1}{1 - 0.9} = 10$$

$$V^\pi(S_1) = \frac{0.9}{1 - 0.9} = 9$$

Exercise



贝尔曼方程: $V^\pi(s_t) = \sum_a \pi(s, a) \sum_{s' \in S} P_{sa}(s') (r_{sa}(s') + \gamma V^\pi(s'))$ **General Case**

- 写出每个状态的贝尔曼方程
- 计算状态值
- 该策略和上一例子中的策略相比如何?

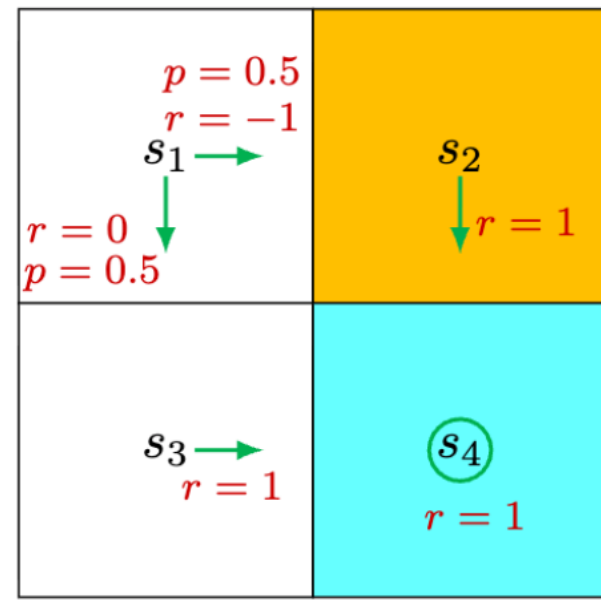
Solution

$$V^\pi(s_1) = 0.5[0 + \gamma V^\pi(s_3)] + 0.5[-1 + \gamma V^\pi(s_2)]$$

$$V^\pi(s_2) = 1 + \gamma V^\pi(s_4)$$

$$V^\pi(s_3) = 1 + \gamma V^\pi(s_4)$$

$$V^\pi(s_4) = 1 + \gamma V^\pi(s_4)$$



求解上述方程可得：

$$V^\pi(s_2) = V^\pi(s_3) = V^\pi(s_4) = \frac{1}{1-\gamma}$$

$$V^\pi(s_1) = -0.5 + \frac{\gamma}{1-\gamma}$$

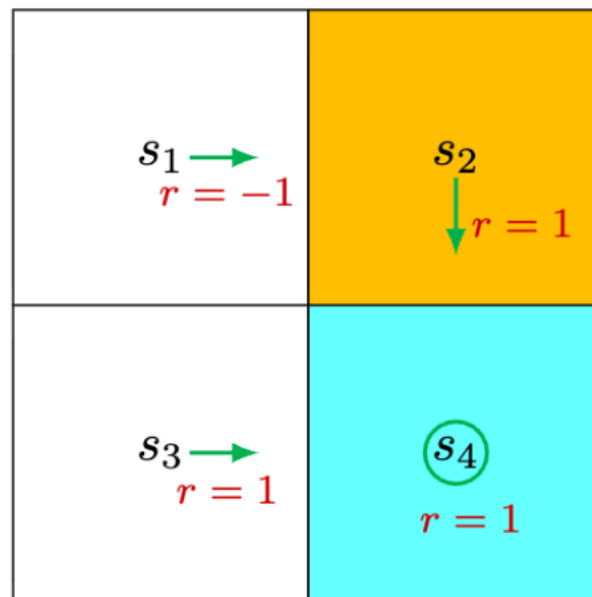
令 $\gamma=0.9$ 可得

$$V^\pi(s_2) = V^\pi(s_3) = V^\pi(s_4) = 10$$

$$V^\pi(s_1) = -0.5 + 9 = 8.5$$

策略改进：如何改进一个策略？

示例



$$V^\pi(s_1) = -1 + \gamma V^\pi(s_2)$$

$$V^\pi(s_2) = 1 + \gamma V^\pi(s_4)$$

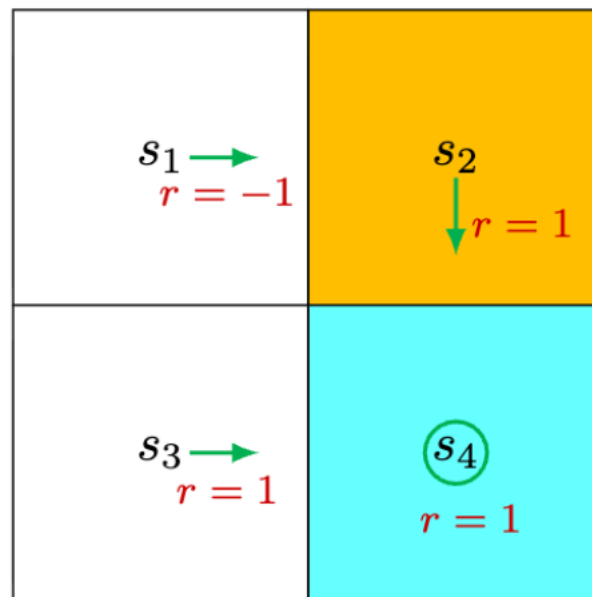
$$V^\pi(s_3) = 1 + \gamma V^\pi(s_4)$$

$$V^\pi(s_4) = 1 + \gamma V^\pi(s_4)$$

令 $\gamma = 0.9$,

$$V^\pi(s_1) = 8, \quad V^\pi(s_2) = V^\pi(s_3) = V^\pi(s_4) = 10$$

示例



令 $\gamma = 0.9$,

$$V^\pi(s_1) = 8$$

$$V^\pi(s_2) = V^\pi(s_3) = V^\pi(4) = 10$$

$$q^\pi(s_1, a_1) = -1 + \gamma V^\pi(s_1) = 6.2$$

$$q^\pi(s_1, a_2) = -1 + \gamma V^\pi(s_2) = 8$$

$$\mathbf{q^\pi(s_1, a_3) = 0 + \gamma V^\pi(s_3) = 9}$$

$$q^\pi(s_1, a_4) = -1 + \gamma V^\pi(s_1) = 6.2$$

$$q^\pi(s_1, a_5) = 0 + \gamma V^\pi(s_1) = 7.2$$

如果我们更新策略，使之选择具有最大动作值的动作，就有望得到一个更好的策略

贝尔曼最优方程

- 最优策略对应的值函数称为最优值函数

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

- 最优价值函数的Bellman等式

$$V^*(s) = \max_a \sum_{s' \in S} P_{sa}(s') (r_{sa}(s') + \gamma V^{\pi}(s')) = \max_a Q^{\pi^*}(s, a)$$

最优Bellman等式

- 非最优策略的改进方式：将策略选择的动作改为当前最优的动作

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

价值迭代 (Value Iteration)

- 对于一个动作空间和状态空间有限的MDP

$$|S| < \infty, |A| < \infty$$

- 价值迭代过程

1. 对每个状态 s , 初始化 $V(s)=0$

2. 重复以下过程直到收敛 {

对每个状态, 更新

$$V(s) = \max_a \sum_{s' \in S} P_{sa}(s') (r_{sa}(s') + \gamma V^\pi(s')) = \max Q^\pi(s, a)$$

}

价值迭代 (Value Iteration)

网格世界



Q表

q-table	a_1	a_2	a_3	a_4	a_5
s_1	$-1 + \gamma v(s_1)$	$-1 + \gamma v(s_2)$	$0 + \gamma v(s_3)$	$-1 + \gamma v(s_1)$	$0 + \gamma v(s_1)$
s_2	$-1 + \gamma v(s_2)$	$-1 + \gamma v(s_2)$	$1 + \gamma v(s_4)$	$0 + \gamma v(s_1)$	$-1 + \gamma v(s_2)$
s_3	$0 + \gamma v(s_1)$	$1 + \gamma v(s_4)$	$-1 + \gamma v(s_3)$	$-1 + \gamma v(s_3)$	$0 + \gamma v(s_3)$
s_4	$-1 + \gamma v(s_2)$	$-1 + \gamma v(s_4)$	$-1 + \gamma v(s_4)$	$0 + \gamma v(s_3)$	$1 + \gamma v(s_4)$

价值迭代 (Value Iteration)

K=0: 初始值 v_0 可任意选择, 不失一般性, 令

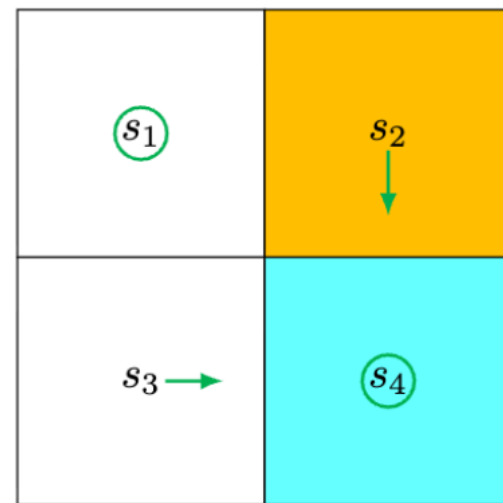
$$v_0(s_1) = v_0(s_2) = v_0(s_3) = v_0(s_4) = 0$$

q-table	a_1	a_2	a_3	a_4	a_5
s_1	-1	-1	0	-1	0
s_2	-1	-1	1	0	-1
s_3	0	1	-1	-1	0
s_4	-1	-1	-1	0	1

价值更新:

$$V(s) = \max_a \sum_{s' \in S} P_{sa}(s') (r_{sa}(s') + \gamma V^\pi(s')) = \max_a Q^\pi(s, a)$$

$$v_1(s_1) = 0, v_1(s_2) = 1, v_1(s_3) = 1, v_1(s_4) = 1$$



价值迭代 (Value Iteration)

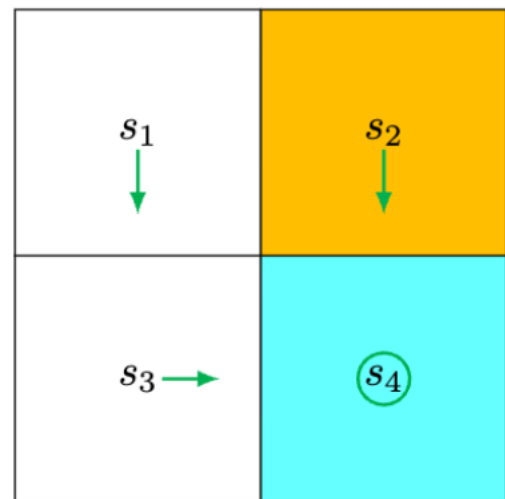
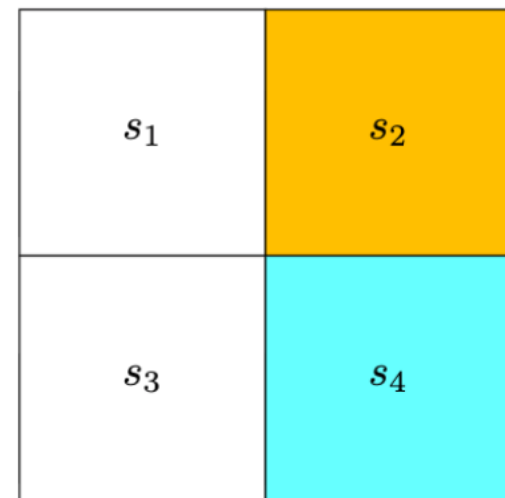
K=1: $v_1(s_1) = 0, v_1(s_2) = 1, v_1(s_3) = 1, v_1(s_4) = 1$

q-table	a_1	a_2	a_3	a_4	a_5
s_1	$-1 + \gamma 0$	$-1 + \gamma 1$	$0 + \gamma 1$	$-1 + \gamma 0$	$0 + \gamma 0$
s_2	$-1 + \gamma 1$	$-1 + \gamma 1$	$1 + \gamma 1$	$0 + \gamma 0$	$-1 + \gamma 1$
s_3	$0 + \gamma 0$	$1 + \gamma 1$	$-1 + \gamma 1$	$-1 + \gamma 1$	$0 + \gamma 1$
s_4	$-1 + \gamma 1$	$-1 + \gamma 1$	$-1 + \gamma 1$	$0 + \gamma 1$	$1 + \gamma 1$

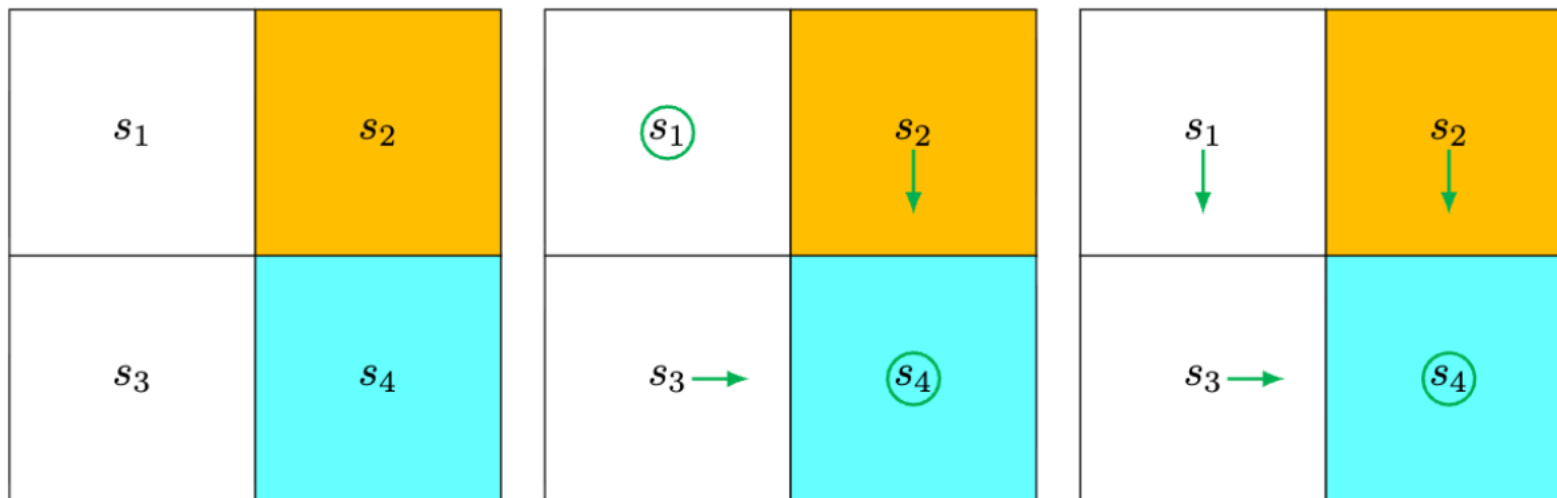
价值更新:

$$V(s) = \max_a \sum_{s' \in S} P_{sa}(s') (r_{sa}(s') + \gamma V^\pi(s')) = \max_a Q^\pi(s, a)$$

$v_1(s_1) = \gamma 1, v_1(s_2) = 1 + \gamma 1, v_1(s_3) = 1 + \gamma 1, v_1(s_4) = 1 + \gamma 1$



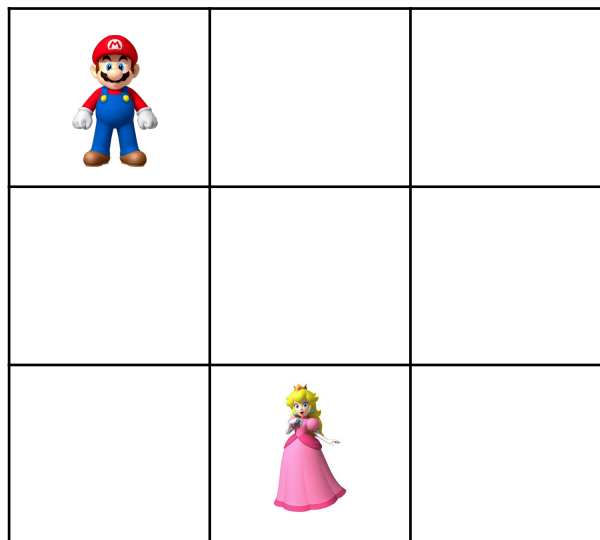
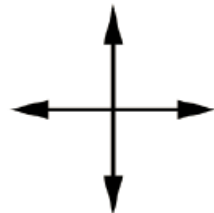
价值迭代 (Value Iteration)



$$v_0(s_1) = v_0(s_2) = v_0(s_3) = v_0(s_4) = 0$$

$$v_1(s_1) = \gamma 1, v_1(s_2) = 1 + \gamma 1, v_1(s_3) = 1 + \gamma 1, v_1(s_4) = 1 + \gamma 1$$

Exercise: 价值迭代 – 公主营救



- 非折扣MDP ($\gamma = 1$)
- 每走一步，奖励为-1，直到到达终止状态
- 找到公主游戏结束
- 最小体力消耗找到公主
- 智能体的策略为均匀随机策略

$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

Exercise: 价值迭代 – 公主营救

 0	0	0
0	0	0
0	0 	0

Exercise: 价值迭代 – 公主营救

 0	0	0
0	0	0
0	0 	0

 -1	-1	-1
-1	-1	-1
-1	0 	-1

 -2	-2	-2
-2	-1	-2
-1	0 	-1

 -3	-2	-3
-2	-1	-2
-1	0 	-1

 -3	-2	-3
-2	-1	-2
-1	0 	-1

策略迭代 (Policy Iteration)

- 对于一个动作空间和状态空间有限的MDP

$$|S| < \infty, |A| < \infty$$

- 策略迭代过程

1. 随机初始化策略 π

2. 重复以下过程直到收敛 {

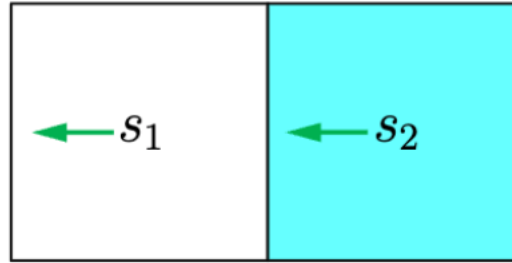
- a) 策略评估: $V := V^\pi = \sum_a \pi(s, a) \sum_{s' \in S} P_{sa}(s') (r_{sa}(s') + \gamma V^\pi(s'))$

- b) 策略改进: 对每个状态, 更新

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

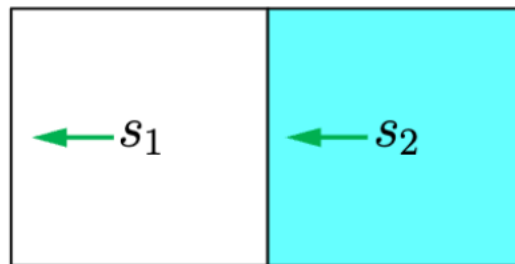
}

策略迭代 (Policy Iteration)



- 动作：向左、向右、不动
- $r_{boundary} = -1, r_{target} = 1$
- $\gamma = 0.9$

策略迭代 (Policy Iteration)



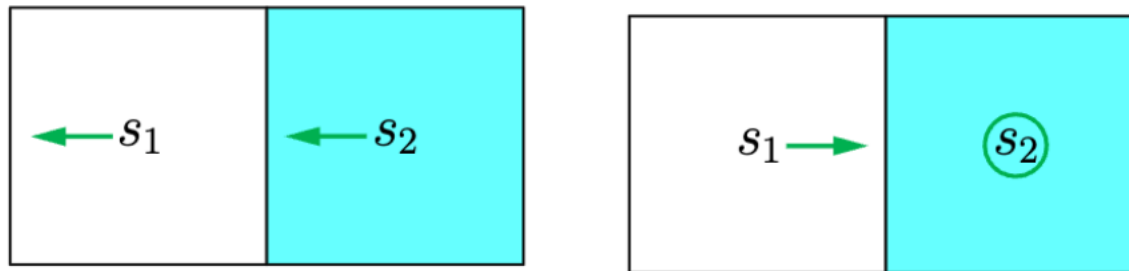
$$V^{\pi_0}(s_1) = -1 + \gamma V^{\pi_0}(s_1)$$

策略评估

$$V^{\pi_0}(s_2) = 0 + \gamma V^{\pi_0}(s_2)$$

$$V^{\pi_0}(s_1) = -10, \quad V^{\pi_0}(s_2) = -9$$

策略迭代 (Policy Iteration)



策略改进

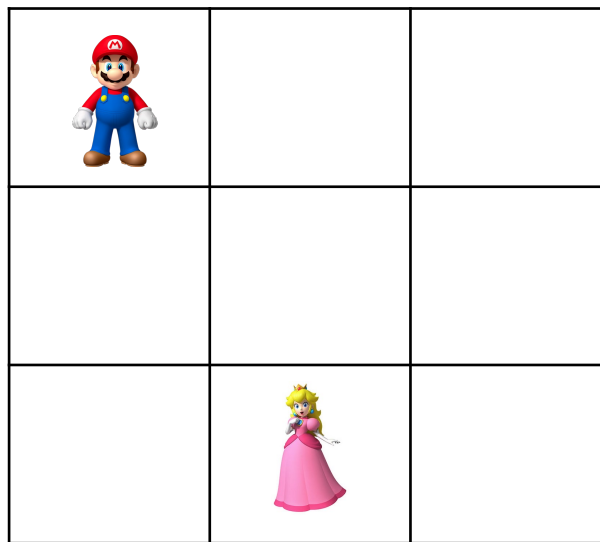
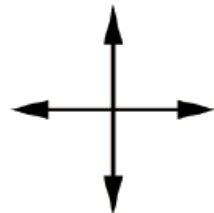
$q_{\pi_k}(s, a)$	a_ℓ	a_0	a_r
s_1	$-1 + \gamma v_{\pi_k}(s_1)$	$0 + \gamma v_{\pi_k}(s_1)$	$1 + \gamma v_{\pi_k}(s_2)$
s_2	$0 + \gamma v_{\pi_k}(s_1)$	$1 + \gamma v_{\pi_k}(s_2)$	$-1 + \gamma v_{\pi_k}(s_2)$

代入 $V^{\pi_0}(s_1) = -10, V^{\pi_0}(s_2) = -9$, 可得

$q_{\pi_0}(s, a)$	a_ℓ	a_0	a_r
s_1	-10	-9	-7.1
s_2	-9	-7.1	-9.1

在状态 s_1 应该选择
向右, 在状态 s_2 应
该选择不动

Exercise: 策略迭代之公主营救



- 非折扣MDP ($\gamma = 1$)
- 每走一步，奖励为-1，直到到达终止状态
- 找到公主游戏结束
- 最小体力消耗找到公主
- 智能体的策略为均匀随机策略

$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

Exercise: 策略迭代之公主营救

 0	0	0
0	0	0
0	0	

策略评估



 -1	-1	-1
-1	-1	-1
-1	0	-1

Exercise: 策略迭代之公主营救

 -1	-1	-1
-1	-1	-1
-1	0 	-1



 -2	-2	-2
-2	-1.75	-2
-2	0 	-1.75

Exercise: 策略迭代之公主营救

 -2	-2	-2
-2	-1.75	-2
-2	0 	-1.75

迭代至收敛



 -3	-2.6	-3
-2.9	-2.5	-2.9
-2.4	0 	-2.4

Exercise: 策略迭代之公主营救

 -3	-2.6	-3
-2.9	-2.5	-2.9
-2.4	0 	-2.4

策略提升



 →	↓	←
↓	↓	↓
→		←

价值迭代 vs. 策略迭代

- 价值迭代过程

1. 对每个状态 s ，初始化 $V(s)=0$

2. 重复以下过程直到收敛 {

对每个状态，更新

$$V(s) = \max_a \sum_{s' \in S} P_{sa}(s') (r_{sa}(s') + \gamma V^\pi(s'))$$

}

- 策略迭代过程

1. 随机初始化策略 π

2. 重复以下过程直到收敛 {

- a) 策略评估: $V := V^\pi$

- b) 策略改进: 对每个状态，更新

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

}

- ✓ 均可归结为基于动态规划的寻优算法
- ✓ 价值迭代是贪心更新法
- ✓ 策略迭代中，用Bellman等式更新价值函数代价很大
- ✓ 对于空间较小的MDP，策略迭代通常很快收敛
- ✓ 对于空间较大的MDP，价值迭代更实用（效率更高）

如果模型未知呢？

- 在模型未知的情形下，从目前我们关注在给出一个已知MDP模型后：
(也就是说，**状态转移**明确给定)
 - 计算最优价值函数
 - 学习最优策略
- 现实问题中，通常难以明确地给出状态转移和奖励函数,导致策略无法评估
(模型未知导致无法做全概率展开)

学习一个MDP模型

$$\begin{array}{lcl} \text{Episode1:} & s_0^{(1)} & \xrightarrow{a_0^{(1)}, R(s_0)^{(1)}} s_1^{(1)} \xrightarrow{a_1^{(1)}, R(s_1)^{(1)}} s_2^{(1)} \xrightarrow{a_2^{(1)}, R(s_2)^{(1)}} s_3^{(1)} \dots s_T^{(1)} \\ \text{Episode2:} & s_0^{(2)} & \xrightarrow{a_0^{(2)}, R(s_0)^{(2)}} s_1^{(2)} \xrightarrow{a_1^{(2)}, R(s_1)^{(2)}} s_2^{(2)} \xrightarrow{a_2^{(2)}, R(s_2)^{(2)}} s_3^{(2)} \dots s_T^{(2)} \\ & \vdots & \vdots \end{array}$$

- 从“经验”中学习一个MDP模型：
 - 学习状态转移概率

$$P_{sa}(s') = \frac{\text{在 } s \text{ 下采取动作 } a \text{ 并转移到 } s' \text{ 的次数}}{\text{在 } s \text{ 下采取动作 } a \text{ 的次数}}$$

World Model

“所谓世界模型，就是世界在人脑或AI的神经网络中的内部表征（心理模型）。在大脑里对外部世界形成抽象表示，形成内部模拟，这可以用于理解、预测和规划外部环境的交互。

大脑构建了一个关于世界如何运作的、经过压缩的、可预测的内部模型。人知道苹果熟了会掉下来而不是飞上去，sora等视频生成AI知道人走在水面会沉下去而不是如履平地，这都是“世界模型”的作用“”

什么是 world models/世界模型



俞扬  
新知答主

 编辑推荐 等 2 项收录

 收录于 · 超龄儿童梦想 >

<https://zhuanlan.zhihu.com/p/661768957>

有没有办法不学习MDP?

能不能从经验数据中直接学习价值函数和策略函数?

模型无关的强化学习
(Model-Free Reinforcement Learning)