教育部-华为智能基座课程

第10章：强化学习II

授课教师：丛润民

山东大学

控制科学与工程学院

章节目录 CONTENTS

章节目录 CONTENTS

深度Q网络算法

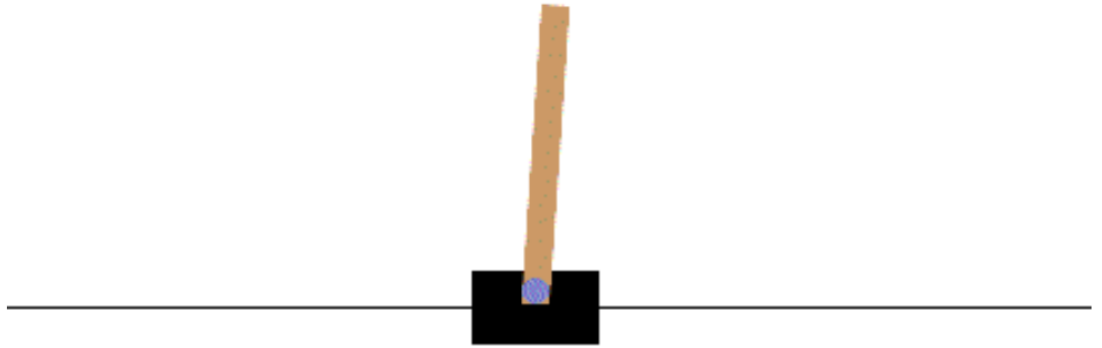
➢ 前情回顾

- 在之前的学习中，我们对强化学习有了初步的认识，知道了什么是马尔可夫链，理解了强化学习中最重要的Q值与V值，学习了蒙特卡罗算法与时序差分算法，了解了Q-learning是如何通过表格记录Q值。
- 接下来，我们将完成进一步的探索，学习强化学习领域的几个经典算法，如DQN深度Q网络算法，AC演员-评论家算法，以及目前公认的主流PPO算法。讲解中，我们以介绍为主，不涉及过多公式推导，让大家对这些算法有个初步认识，希望激发大家对强化学习的兴趣。
- 掌握DQN算法的两大稳定改进思路

深度Q网络算法

Q-learning

Q-learning核心是用Q值给‘状态+动作’打分，选评分最高的动作。比如在CartPole小车立杆任务中，状态是“小车位置、杆的角度”（只有4个数值），我们可以用一个表格把所有Q值存起来，这叫**表格型Q学习**。



问题导入

如果遇到复杂任务，比如训练一个RL模型玩马里奥游戏，状态是游戏画面的像素点，有几十万个像素，状态总数多到无法想象！这个时候还可以使用表格去记录Q值吗？

深度Q网络算法

Q-learning 局限性

显然，表格无法存储如此“庞大”的数据，这就是Q-learning的“瓶颈”：**无法处理高维状态**

解决方案

回顾机器学习相关知识，神经网络擅长处理高维数据，且具备拟合复杂映射关系的能力——能够将高维输入（如像素画面）映射为特定输出（如动作评分），这一特性恰好能解决表格型Q学习的高维状态存储与计算瓶颈。因此，我们可以用神经网络替代表格来拟合Q值，这就是今天要学习的深度Q网络，即**DQN算法。**

深度Q网络算法

DQN（Deep Q-Network）是强化学习领域的经典算法之一，由DeepMind团队于2013年提出。它通过结合深度神经网络和Q-Learning算法，解决了高维、连续状态空间的问题，具备了学习和规划能力，开启了“深度强化学习（DRL）”的时代。

DQN使用深度卷积神经网络（CNN）来逼近Q值函数，替代传统的表格形式。网络输入为状态，输出为各动作的Q值。

例如，针对马里奥游戏而言，输入就是彩色游戏画面，输出则是每个可能动作的Q值，神经网络会对各个动作进行评分，智能体选择最高评分对应的动作。

在DQN框架中，神经网络承担动作价值函数的拟合任务，通过学习将高维状态表征映射为各动作的Q值输出，其核心价值在于突破表格型Q学习对状态维度的限制，实现高维状态下Q值的高效估计与存储。

深度Q网络算法——DQN

相比于普通的Q-learning，DQN做出了两大重要的改进：

- 使用两个独立的神经网络：目标Q网络与当前Q网络，通过最小化损失函数来更新当前Q网络，当更新到达一定次数后，再更新目标Q网络。

- 引入经验回放池，将智能体的信息记录下来，并存储在一个回放缓冲区中。在训练时，从回放缓冲区中随机抽取一小批数据进行训练。这使样本满足独立假设，并提高样本的效率，每一个样本可以被使用多次，十分适合神经网络的梯度学习。

深度Q网络算法——DQN

➢ 目标网络的引入

前面提到，Q-learing算法是利用当前奖励R和未来状态的最优动作Q值去更新当前“状态+动作”的Q值：$Q(\mathrm{s},a) \leftarrow R(s,a)+\gamma \cdot \max_{a_{t+1}} Q(s_{t+1},a_{t+1})$

同样，DQN也是按照该原则去更新网络，若仅使用单一网络（当前网络）同时承担 “计算当前 Q 值” 和 “计算未来最优 Q 值（更新目标）” 的双重任务，网络参数的实时更新会导致更新基准持续波动，模型需不断追逐动态变化的目标，易引发训练震荡、难以收敛。

深度Q网络算法——DQN

以**猫追老鼠**为例。猫是 Q 估计，老鼠是 Q 目标。一开始，猫离老鼠很远，所以我们想让猫追上老鼠。如图 b所示，因为 Q 目标也是与模型参数相关的，所以每次优化后，Q 目标也会动。这就导致一个问题，猫和老鼠都在动。如图 c 所示，猫和老鼠会在优化空间里面到处乱动，使得训练过程十分不稳定。所以可以固定 Q 网络，让老鼠动得不那么频繁，可能让它每 5 步动一次，猫则是每一步都在动。如果老鼠每 5 次动一步，猫就有足够的时间来接近老鼠，它们之间的距离会随着优化过程越来越小，拟合后就可以得到一个很好的Q 网络。

(a) 猫追老鼠

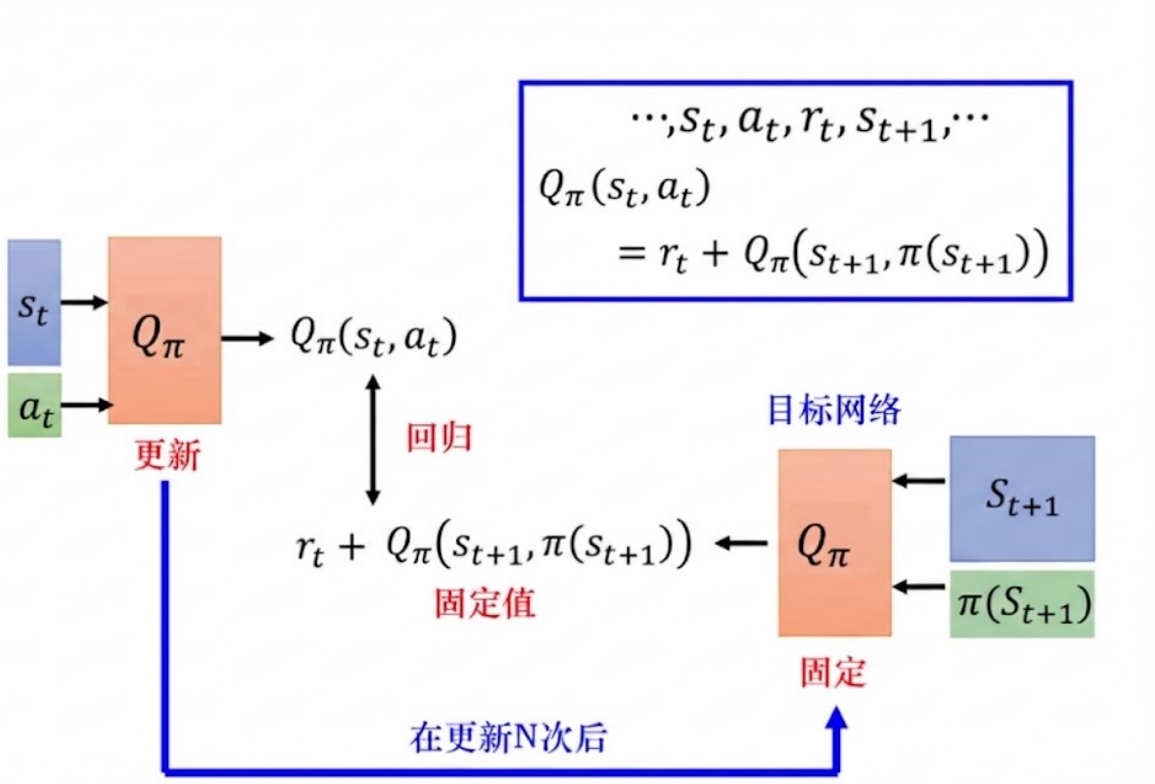
(b) 猫和老鼠都在动

(c) 猫和老鼠的优化轨迹

深度Q网络算法——DQN

➢ 目标网络的引入

基于上述问题，研究者在单一网络基础上加入**目标网络**，目标网络与当前网络结构完全一致，但目标网络的参数不随训练实时更新，仅定期从当前网络同步参数。这使得Q 值更新的基准在同步周期内保持固定，有效降低了目标值的波动性，为模型学习提供了稳定的参考框架，提升了DQN 训练的稳定性与收敛可靠性。



深度Q网络算法——DQN

➢ 经验回放池的引入

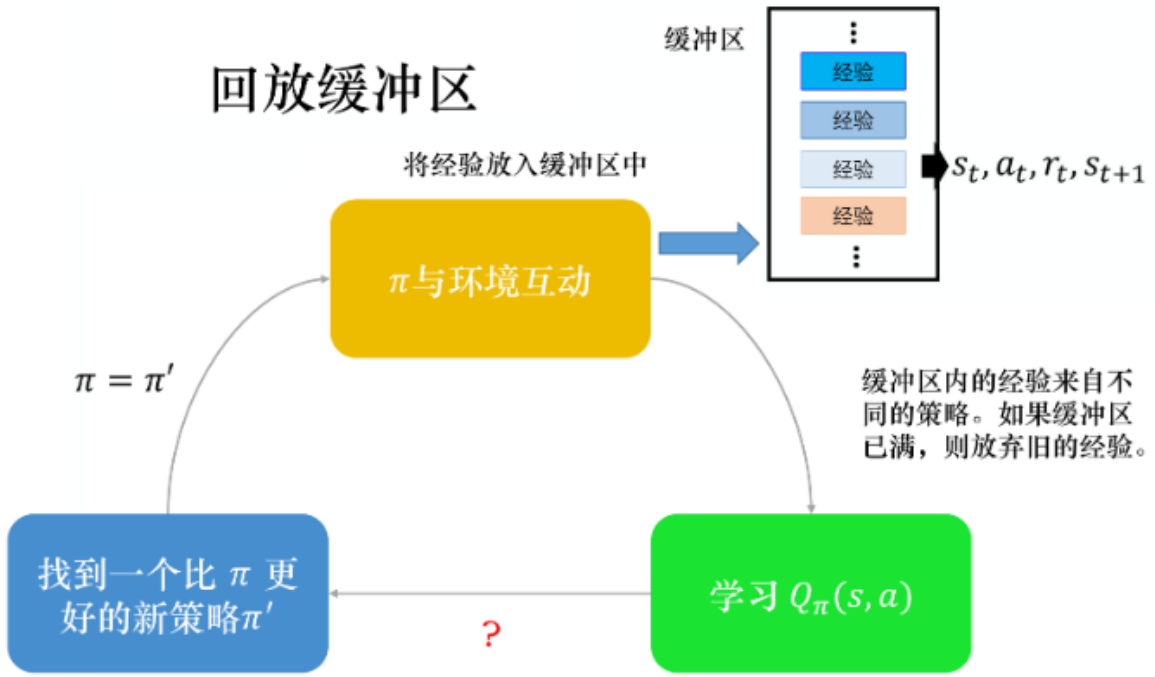
在进行强化学习的时候， 往往最花时间的步骤是与环境交互，训练网络反而是比较快的。用回放缓冲区可以减少与环境交互的次数，一些过去的策略所得到的经验可以放在回放缓冲区里面被使用很多次，被反复的再利用。

智能体与环境交互产生的训练数据天然具有时序相关性，直接使用这类连续相关数据训练，会导致模型学习偏倚、训练过程震荡，难以稳定收敛。经验回放池通过存储智能体的交互数据，并在训练时随机采样数据，打破数据的时序关联，使训练数据分布更均衡。

深度Q网络算法——DQN

➢ 经验回放池的引入

通过从经验回放池中随机采样不同策略与环境的交互数据，打破了时序关联，提升数据多样性，提升模型的训练效果。



深度Q网络算法——DQN

➢ DQN网络的更新原则

简单来说，DQN的更新就是为了最小化这么一个损失函数：

$$L(\theta)=E_{(s,a,r,s_{t+1})}\left[\left(r+\gamma\cdot\max_{a_{t+1}}Q_{target}(s_{t+1},a_{t+1};\theta_{target})-Q(s,a;\theta)\right)^2\right]$$

θ是当前Q网络的参数

θ_{target}是目标Q网络的参数

s和a是当前状态和动作

r是即时奖励，γ是折扣因子

s'是下一个状态，a'是下一步动作

深度Q网络算法——DQN

➢ DQN算法流程

- 初始化函数Q、目标函数$\hat{Q}$，令$\hat{Q} = Q$。
- 对于每一个回合。
  - 对于每一个时间步t。
    - 对于给定的状态s_t，基于Q（ϵ-贪婪）执行动作a_t。
    - 获得反馈r_t，并获得新的状态s_{t+1}。
    - 将(s_t, a_t, r_t, s_{t+1})存储到缓冲区中。
    - 从缓冲区中采样（通常以批量形式）(s_i, a_i, r_i, s_{i+1})。
    - 目标值是$y = r_i + \max_a \hat{Q}(s_{i+1}, a)$。
    - 更新Q的参数使得$Q(s_i, a_i)$尽可能接近于y（回归）。
    - 每C次更新重置$\hat{Q} = Q$。

深度Q网络算法——DQN

➢ DQN算法衍生

基于传统的DQN算法，研究者基于不同的问题进行了一系列改进，衍生出了Double DQN、Dueling DQN、Noisy DQN、Rainbow DQN等新的算法，分别解决不同的问题。

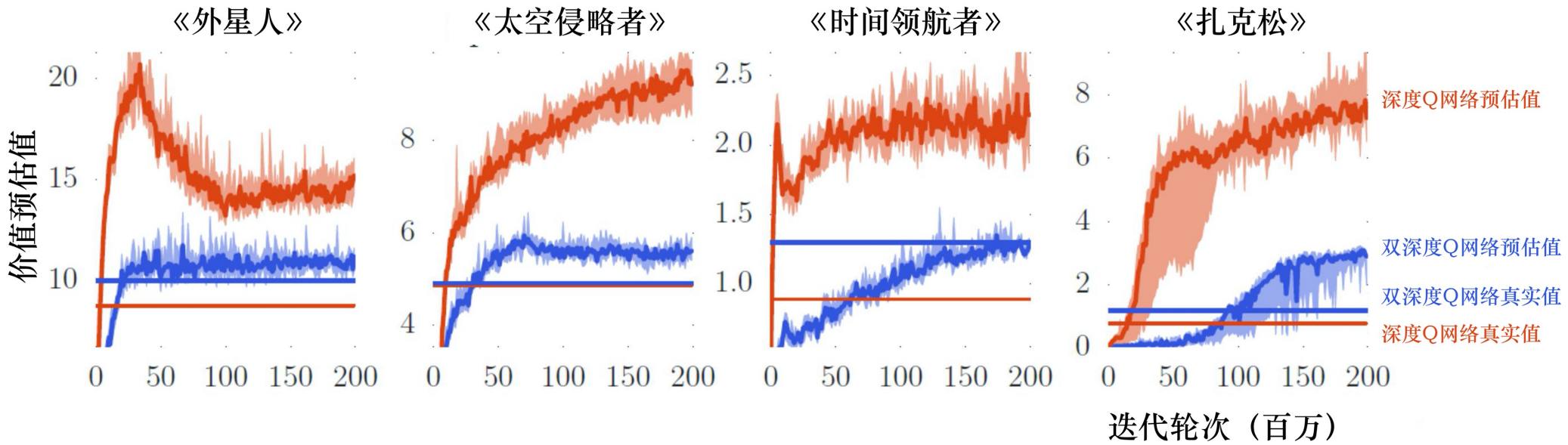
如Double DQN用于解决**Q值过估计**问题，Dueling DQN用于优化**价值函数的表示结构**，Noisy DQN则进行自适应探索，用于解决**局部最优**问题，Rainbow DQN则是DQN的集大成者，将其他衍生算法的核心改进策略整合到一个框架中。

本课程我们以**Double DQN**和**Dueling DQN**为例，结合其解决的具体问题进行简要介绍。

深度Q网络算法——DQN

➢ DQN过估计问题

这里有4种不同的小游戏，横轴代表**迭代轮次**，红色锯齿状线表示Q函数对不同的状态估计的平均 Q 值，有很多不同的状态，每个状态我们都进行采样，算出它们的 Q 值，然后进行平均。将DQN网络估计值与真实值进行对比，我们发现DQN网络存在明显的**高估问题**!



深度Q网络算法——DQN

➢ DQN网络的过估计问题

回想DQN网络的更新原则，训练时候，我们希望让左式与右式（目标）越接近越好。

$$Q(s_t, a_t) \leftrightarrow r_t + \max_a Q_{\text{target}}(s_{t+1}, a_{t+1})$$

在估计的时候，网络本身就存在误差，会将动作高估，智能体又总是会选择Q值最大的那个动作。因此网络的目标值很容易被设的过高，这种情况又会随着网络的更新被进一步加强。

深度Q网络算法——DQN

➢ DQN的衍生算法—Double DQN

在DQN网络中，目标 Q 值计算时“动作选择”与 “价值评估” 由同一目标网络完成，易放大估计偏差，导致 Q 值系统性偏高。为了解决这个问题，研究者拆分“选”与“评”的主体，分别由当前网络和目标网络单独完成，这就是DDQN网络，即双深度Q网络。

DDQN（Double DQN）网络相较于DQN网络修改最少，仅将价值评估交由目标网络完成，当前网络仅负责动作选择，这样避免了同一网络的偏差累积，让 Q 值估计更精准。

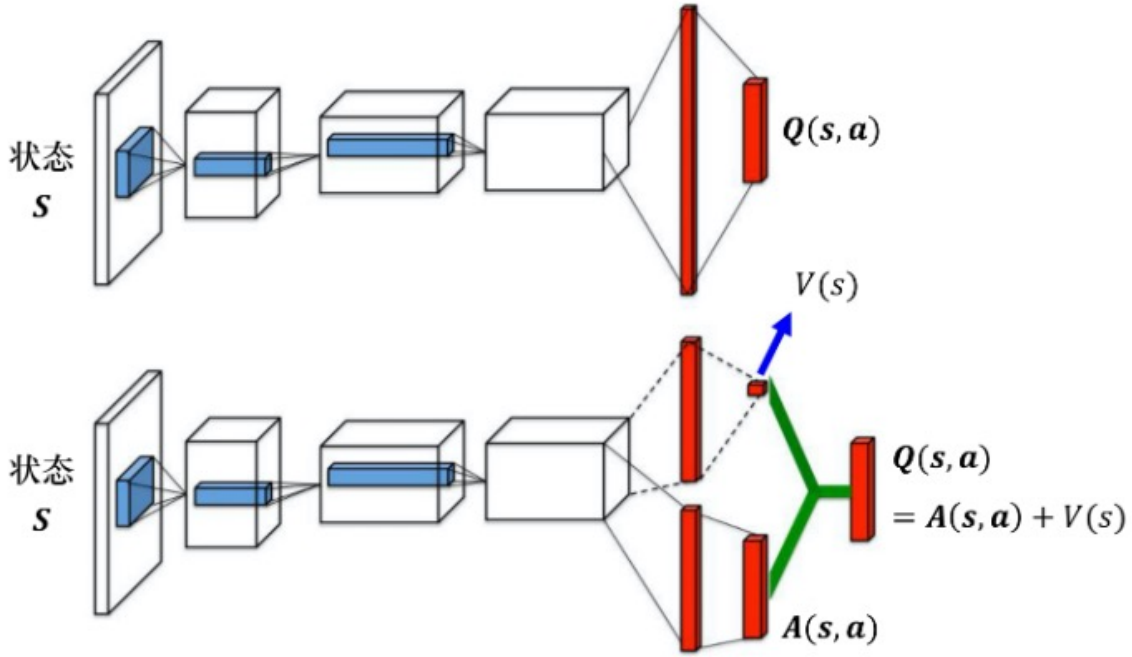
此时目标值发生改变： $r_t + \max_a Q(s_{t+1}, a_{t+1}) \rightarrow r_t + Q_{target}\left(s_{t+1}, \arg\max_a Q(s_{t+1}, a_{t+1})\right)$

其中，$\arg\max_a Q(s_{t+1}, a_{t+1})$是当前网络选择的动作

深度Q网络算法——DQN

➢ Dueling DQN 网络

Dueling DQN（竞争深度Q网络）是DQN的另一个改进算法,它能够很好地学习到不同动作的差异性，在动作空间较大的环境下非常有效。



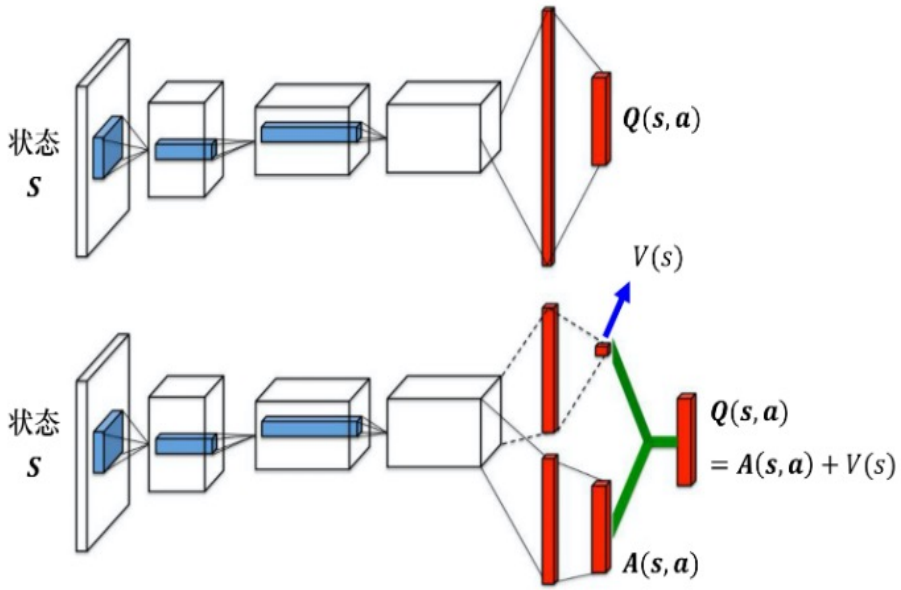
深度Q网络算法——DQN

Dueling DQN中定义 $A(s,a)=Q(s,a)-V(s)$

A(s,a)为每个动作的**优势函数**，Dueling DQN将价值函数与V(s)与优势函数A(s,a)作为神经网络的两个不同分支来输出，然后求和得到Q值。

$$Q(s,a)=V(s)+A(s,a)$$

传统的DQN每次更新只会更新某个动作的Q值，其他动作的Q值保持不变。而Dueling DQN可以学习状态价值函数，每次更新，状态价值函数与优势函数均会被更新，从而影响所有动作的Q值，使得网络具备“举一反三”的能力。



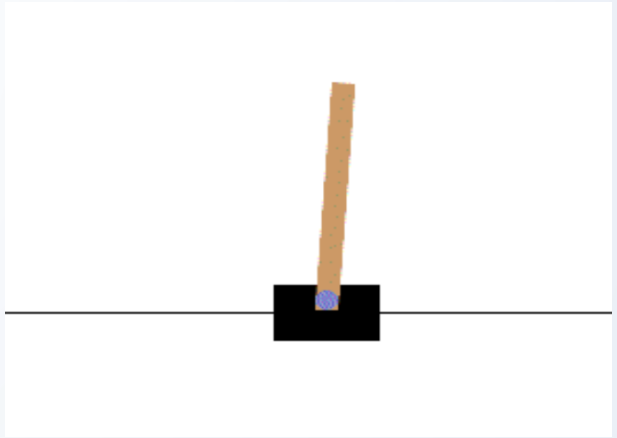
章节目录 CONTENTS

其他典型算法

连续动作问题

DQN及其变体虽优化了Q值计算效率，但仍有个核心局限——只适合“**离散动作**”。

什么是离散动作？就是动作只有有限几个选项，比如CartPole的“左推、右推”（2个动作）。



但如果是机器人控制，比如机械臂转动角度、四足机器人迈腿的力度，这些“**连续动作**”就没法用DQN处理了——因为DQN要给每个动作算Q值，连续动作有无数个。

其他典型算法

前面我们学习了DQN算法，通过对价值Q预测来间接优化动作选择，这种算法称为Value-based算法，即基于价值的算法。由于自身的局限，无法应对连续动作问题。

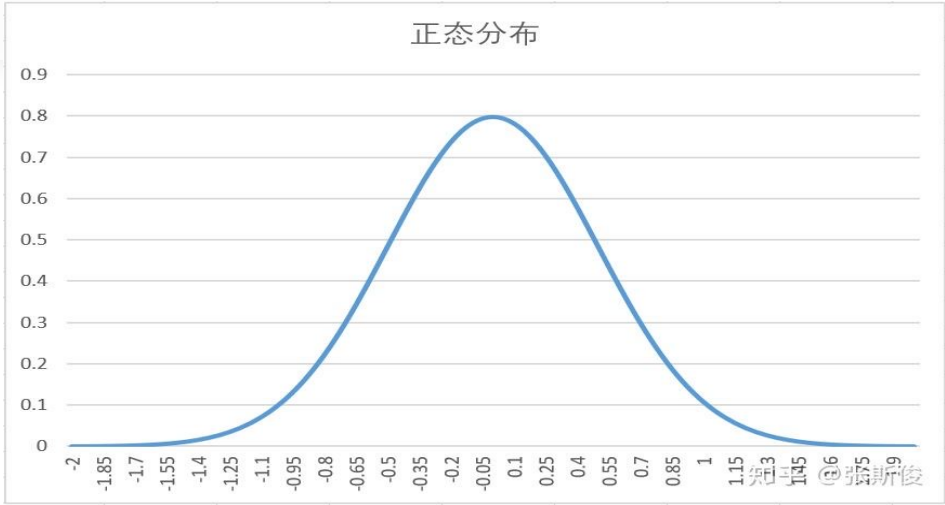
为了解决连续动作问题，研究者提出了一种**基于策略**的算法，即Policy-based算法，与Value-based算法不同的是，该类算法直接学习策略函数，通过提升 “**高收益动作**” 的选择概率更新网络。

接下来，我们以Policy-based中的典型算法——PG算法为例向大家讲解。

其他典型算法

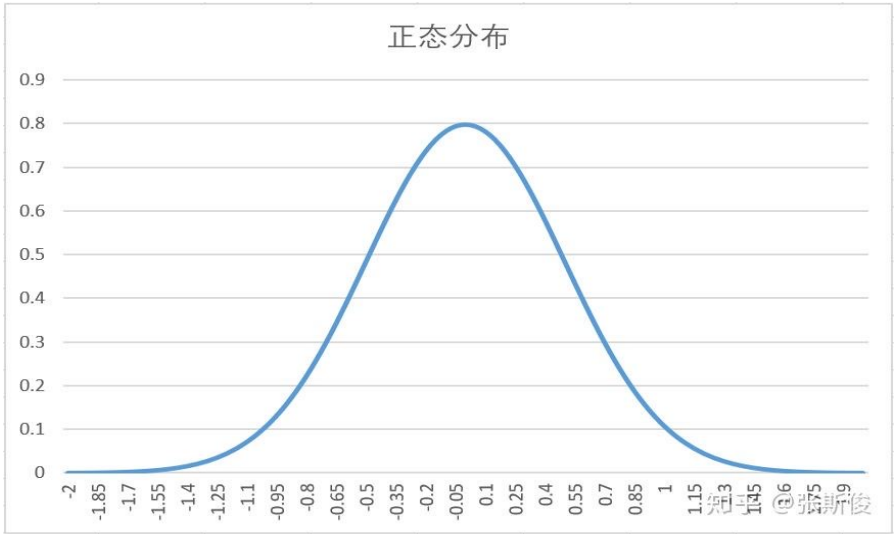
策略梯度算法，即Policy Gradient Algorithm是“**纯基于策略算法**”的典型代表，核心是直接学习“策略函数π(a|s)”——输入当前状态s，直接输出动作选择的概率分布（比如离散动作下“左移60%、右移40%”，连续动作下输出动作分布参数），无需先计算Q值/V值，直接通过策略概率选动作。

简单说，在连续动作问题中，PG网络的输入是智能体当前所处的状态，输出两个参数，一个是均值，另一个是方差，有了这两个参数我们就可以构建一个**正态分布函数**，后续通过采样得到具体的动作值。



其他典型算法

如图所示，通过PG网络输出的均值与方差参数构建了一个正态分布函数，其中横坐标表示**动作值**，纵坐标表示执行某个动作的概率。智能体在与环境交互的过程中获得“奖励”，PG网络通过更新输出的均值和方差使得“奖励”最大的动作拥有最大的执行概率，进而获得最有的动作策略。



其他典型算法

PG算法

PG算法仅用了一个网络就可以同时解决离散和连续的动作问题，这么看，相对于DQN算法而言简直PG算法简直是”降维打击“！但事实真的是如此吗?

我们可以把PG网络看成是一个篮球新手投中了才知道 “这个动作对” （高奖励，提高动作概率），投歪了只能瞎调整（低奖励，降低动作概率）；全程没人指导，不知道动作错在哪（无价值评估），进步慢、经常越练越乱（训练不稳定、方差大）。

➢ Actor-Critic 算法介绍

将PG网络视作一个篮球新手，仅靠自己练习费时费力，且进步缓慢，那我们干脆去找一个教练指导，每投完一次篮，教练都会给予指导，告诉新手说是”手腕没压好“还是”发力太猛“等等。

这就是我们接下来要讲的Actor-Critic算法，简称**AC算法**，即**演员-评论家算法**。AC算法在基于PG算法加入了一个网络用于价值评估，以此来指导策略网络的更新。

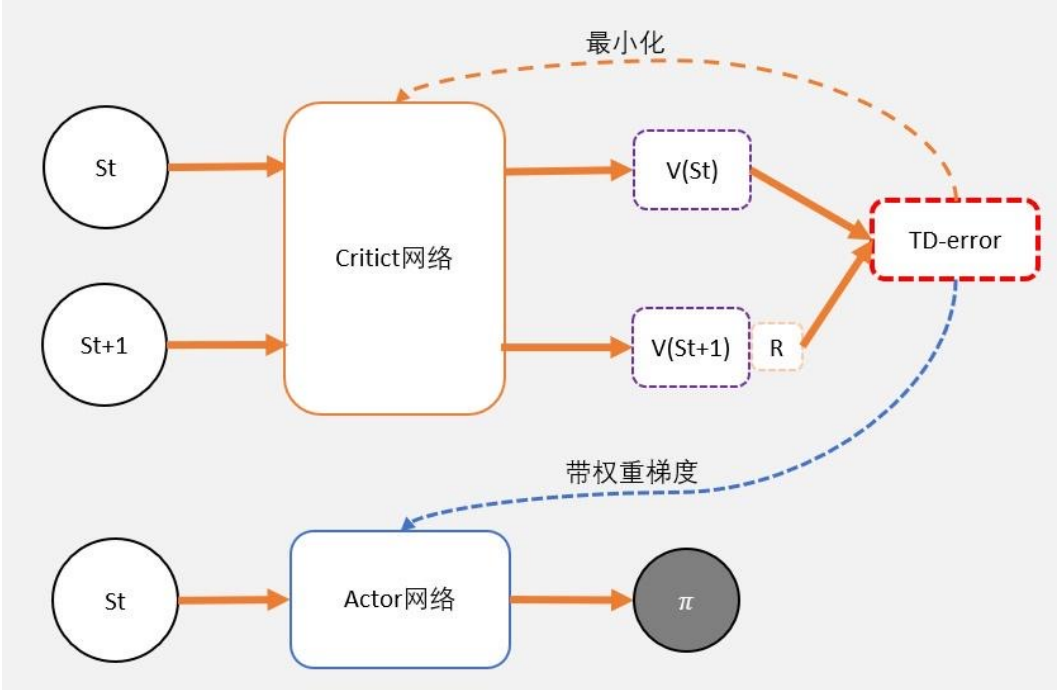
其他算法

➢ AC算法网络框架

如图所示，Critic网络估计相邻状态的V值，由此计算TD-error用于对自身网络的更新。同时TD-error也用来指导Actor网络的更新。

$$TD-error=\delta=\gamma\cdot V_{s_{t+1}}+R-V_{s_t}$$

TD-error即为Critic网络的损失函数，Critic网络通过更新使得TD-error最小化。



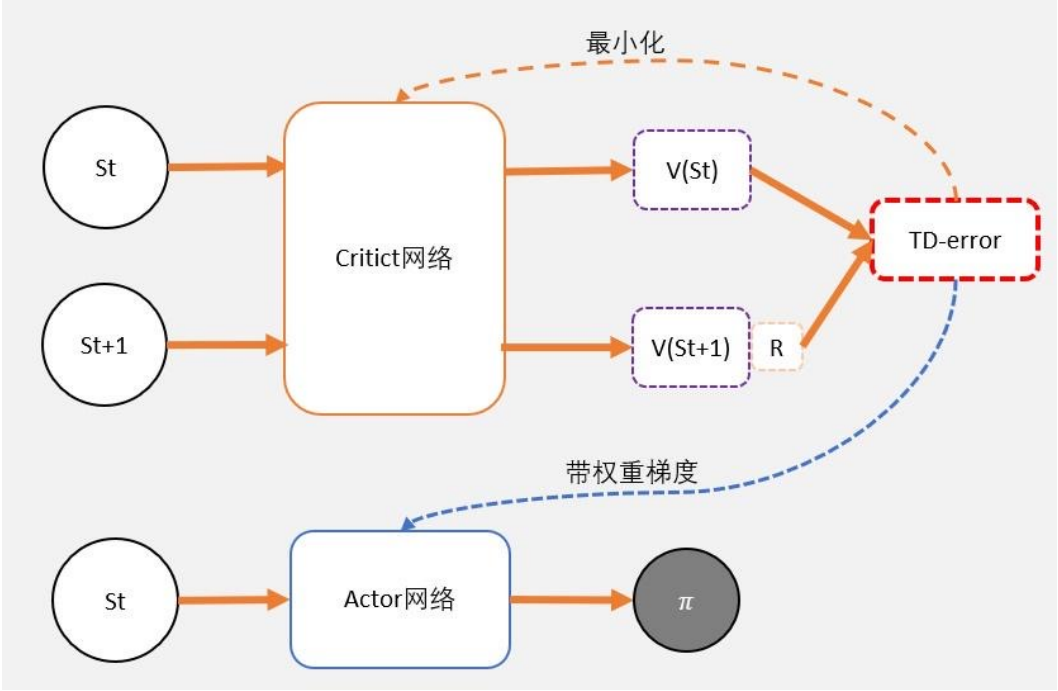
其他算法

➢ AC算法网络框架

Actor网络的更新原则是使得这样一个损失函数最小化：$loss = -log\pi_\theta(a|s) \cdot \delta$

由于负号的存在，可以理解为是让$log\pi_\theta(a|s) \cdot \delta$最大化。

可以看出，AC网络中既有策略更新，也有价值评估，可以看作是DQN算法与PG算法的结合，因此AC网络是一种”策略+价值“网络的融合体。



➢ A3C算法

强化学习有一个问题，就是它很慢，怎么提高训练的速度呢？例如，如图所示，在动漫《火影忍者》中，有一次鸣人想要在一周之内打败晓，所以要加快修行的速度，鸣人的老师就教他一个方法：用影分身进行同样的修行。两个一起修行，经验值累积的速度就会变成两倍，所以鸣人就使用了 1000 个影分身来进行修行。这就是**异步优势演员-评论员算法**（A3C）思想的体现。

其他算法

➢ A3C算法

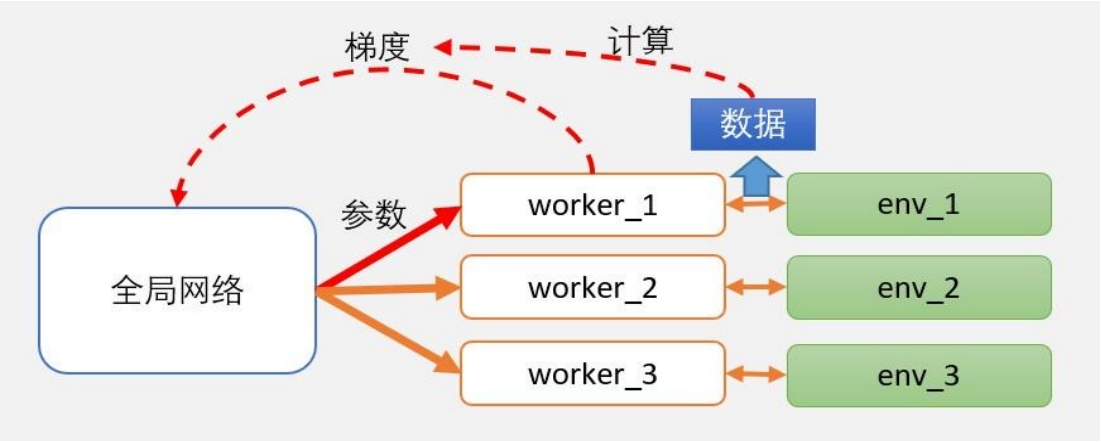
A3C 算法的全称是 Asynchronous Advantage Actor-Critic，即异步优势演员-评论家算法。由 DeepMind 于 2016 年提出，是首个将 “异步分布式训练” 与 “Actor-Critic 框架” 结合的经典算法。

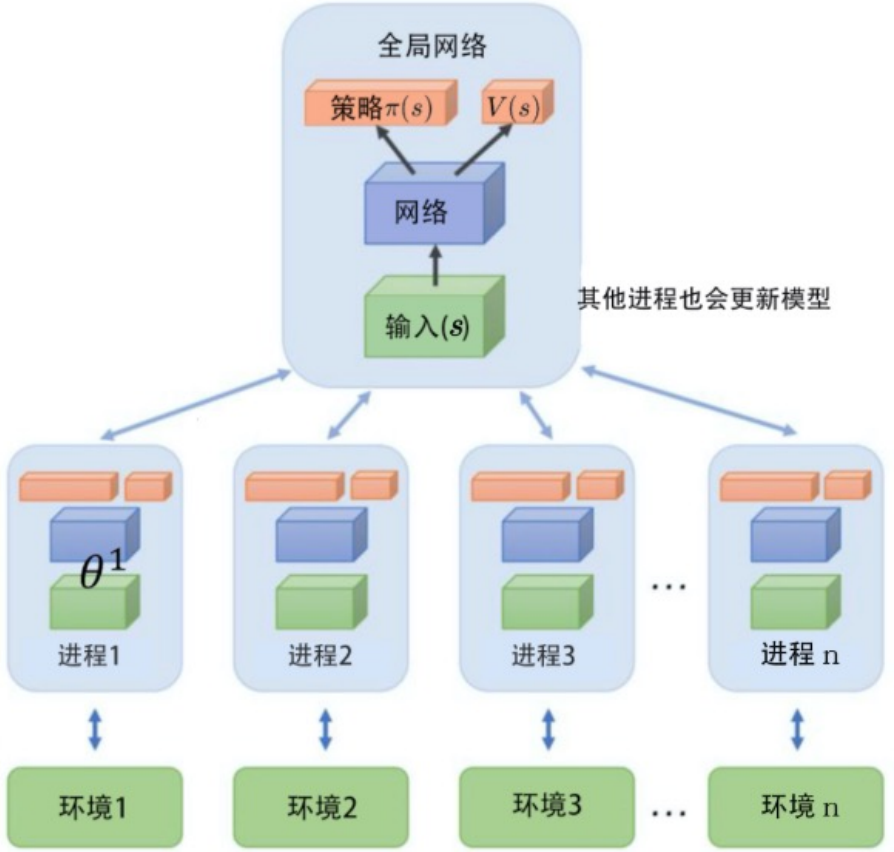
它解决了传统 AC 算法训练速度慢、样本利用率低的问题，为后续分布式强化学习（如 DPPO）奠定了架构基础；其本质是 “分布式 AC 算法”，通过多 Worker 并行产生数据，异步更新全局参数，大幅提升训练效率。

其他算法

A3C的架构图分为两个主要部分：Global Network（全局网络）和 worker（工人）。

Global network和 worker都是一样的AC结构网络。全局网络并不直接参加和环境的互动，工人与环境有直接的互动，并把学到的东西（即梯度）汇报给全局网络。worker向全局网络汇总梯度后并应用在全局网络的参数更新后，全局网络会把当前学习到的最新版本的参数直接给worker。





其他典型算法

➢ 近端策略优化算法

单纯基于策略的算法（如策略梯度）直接学习动作策略，无价值评估模块，不仅存在训练不稳定、效率低的问题，还存在两个核心痛点——“更新步长敏感”和“样本利用率低”；AC框架虽加入价值评估解决了部分稳定性问题，但未根治这两个痛点。

PPO（Proximal Policy Optimization，**近端策略优化算法**）是由 OpenAI 团队在 2017 年提出的强化学习算法，目前已成为该领域的“业界标准”。PPO 是为了解决标准策略梯度算法（Policy Gradient）中“更新步长敏感”和“样本利用率低”的问题而诞生的。

➢ PPO算法解决更新步长敏感问题

PPO算法基于AC框架，重点解决了更新步长敏感与样本数据利用率低的问题。

首先，针对”更新步长敏感“的问题，PPO算法采用“重要性采样”的方法，即引入了KL散度，在策略更新时进行约束。

通过计算 $r_t(\theta)=\dfrac{\pi_{\theta_{new}}(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}$ 这样一个概率比例，反应新旧策略之间的差异。

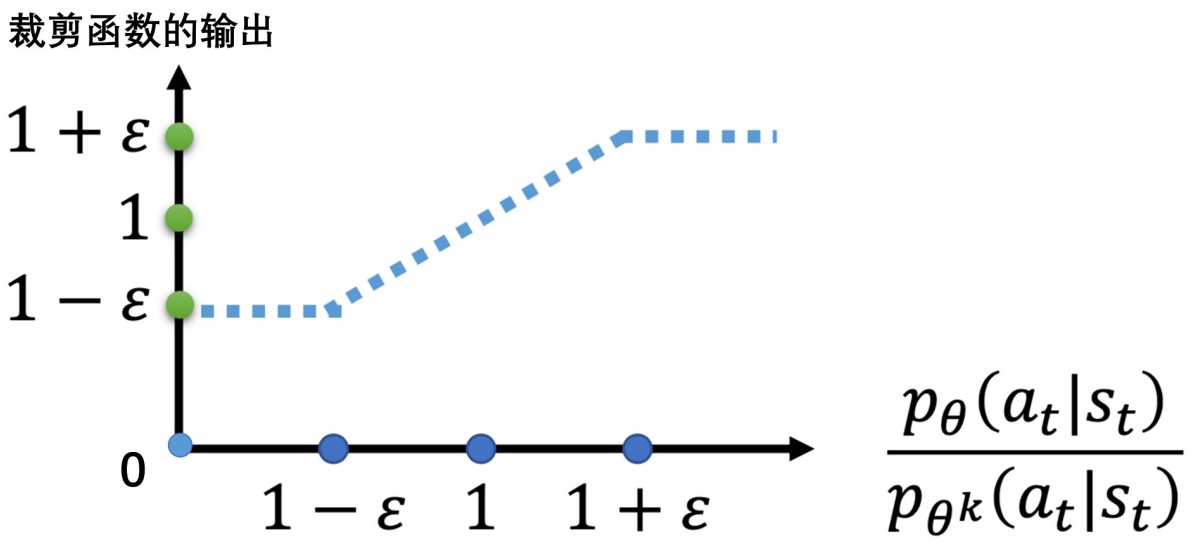
若 $r_t > 1$，说明新策略比旧策略更看好这个动作。

若$r_t < 1$，说明新策略开始“嫌弃”这个策略。

➢ PPO算法解决更新步长敏感问题

此外，PPO加入裁剪方法，对$r_t(\theta)$进行裁剪，将其限制在（$1-\varepsilon$，$1+\varepsilon$）区间内，最终将裁剪后的$r_t(\theta)$作为一个系数用于网络参数，由此可以很好的限制网络的更新幅度，保证训练的稳定性。

这就是Proximal Policy Optimization中Proximal（近端）的由来，即缩小相邻策略的差异，限制更新步长。



其他典型算法

➢ PPO算法解决样本数据利用率低的问题

在AC算法中，必须用 “当前正在学习的策略” 与环境交互产生的实时数据训练。旧数据对应的动作分布、价值评估标准会与当前策略偏差越来越大，直接复用会引入训练偏差，导致模型震荡、收敛困难。

在PPO算法中，由于我们通过引入KL散度并对其进行裁剪限制了新旧策略之间的差异，因此在PPO算法中，可以对同一份样本数据进行多次利用，大大提高了样本的利用效率。

➢ DPPO算法介绍

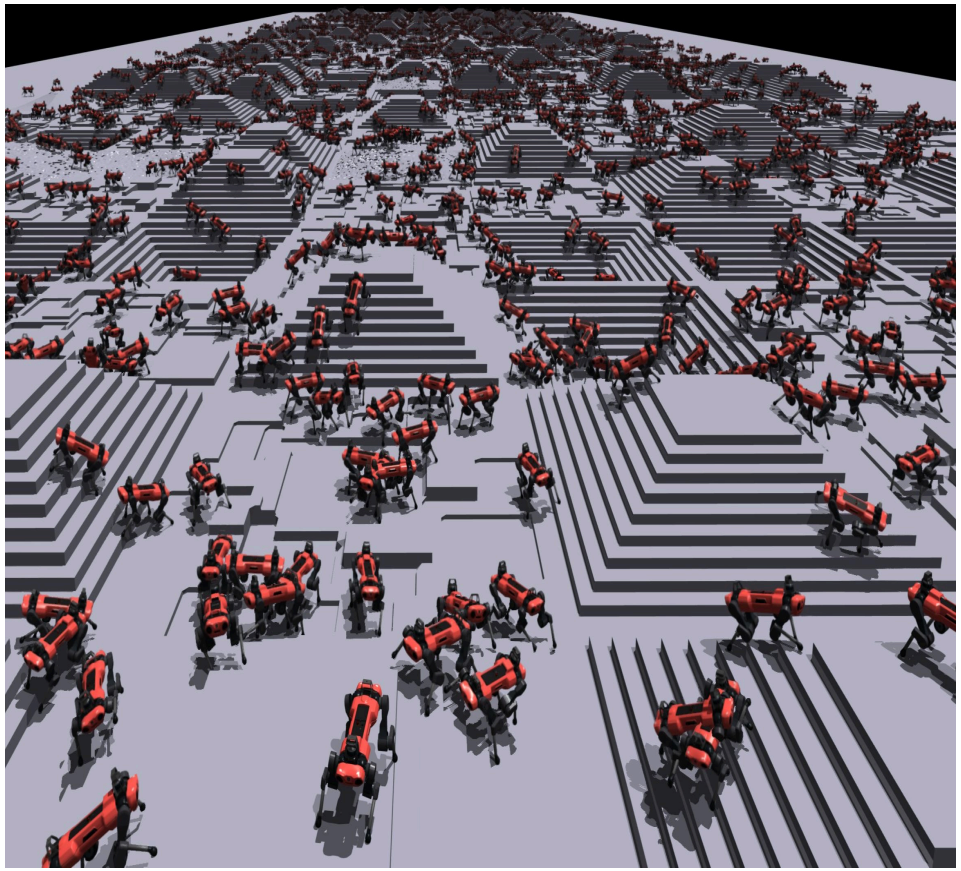
接下来，我们介绍一个PPO算法的重要变体DPPO算法。DPPO和A3C的思路其实是一致的，希望用多个智能体同时和环境互动，并对全局的PPO网络进行更新。

在A3C，我们需要跑数据并且计算好梯度，再更新全局网络。这是因为AC是一个在线的算法，即在更新的时候，产生数据的策略和更新的策略需要时同一个网络。所以我们不能把worker产出的数据，直接给全局网络计算梯度用。

但PPO解决了离线更新策略的问题，所以DPPO的Worker只需要提供数据给全局网络，由全局网络从数据中直接学习。

其他典型算法

PPO算法的强大优势使其成为强化学习领域的主流算法，在游戏AI，四足机器人步态优化，人形机器人平衡控制，自动驾驶等领域获得了广泛的应用。



应用场景一：游戏AI

PPO能够有效处理由像素构成的高维视觉输入和包含数百个可能动作的复杂离散动作空间，通过与环境的持续交互，学习到接近甚至超越人类专业选手的游戏策略。

OpenAI Five是一个由五个AI智能体组成的团队，它们从零开始学习DOTA 2，最终在一场备受瞩目的比赛中击败了当时的世界冠军OG战队，展示了PPO在团队协作和长期战略规划方面的巨大潜力。

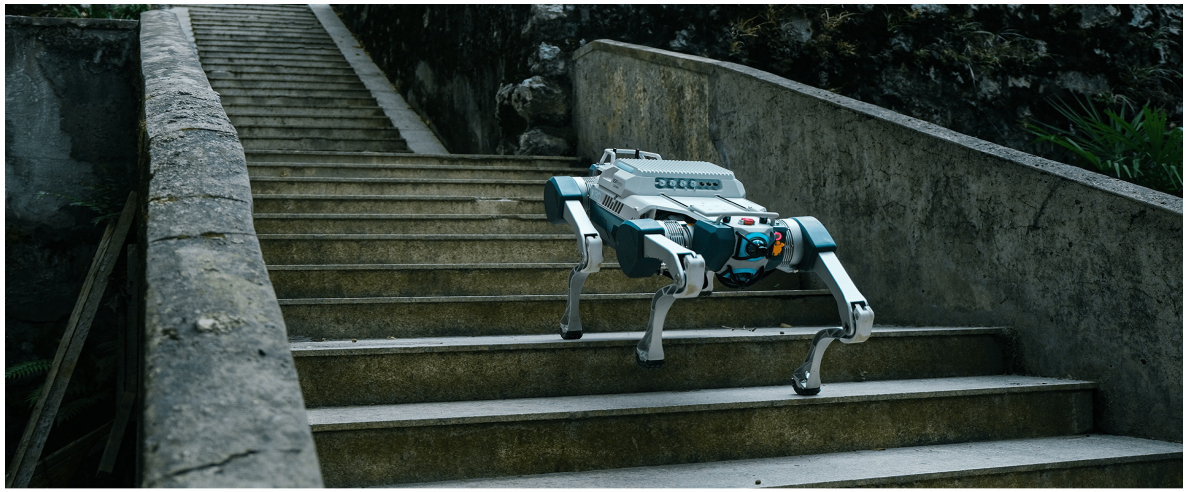
PPO的优势在于其能够学习到长期的、具有预见性的策略，并能快速适应游戏环境中的各种动态变化和对手的不同打法。

应用场景二：机器人控制

PPO非常适合处理机器人控制中普遍存在的高维连续动作空间问题，例如控制每个关节的力矩或角度。它能够从与物理世界的交互中学习到鲁棒性强的控制策略，无需依赖精确的动力学模型。

四足机器人步态：学习在平坦、崎岖甚至动态变化的地形上稳定地行走、奔跑和跳跃，通过调整步幅、步频和腿部姿态来保持平衡和前进。

人形机器人：控制复杂的人形机器人身体结构，完成诸如行走、上下楼梯、搬运物体等需要高度平衡和协调能力的任务。



应用场景三：自动驾驶

PPO在自动驾驶领域主要用于优化决策和控制模块。它可以处理从感知到控制的端到端学习，也可以用于优化特定的决策任务，如车道变换、跟车距离控制和交通灯识别。

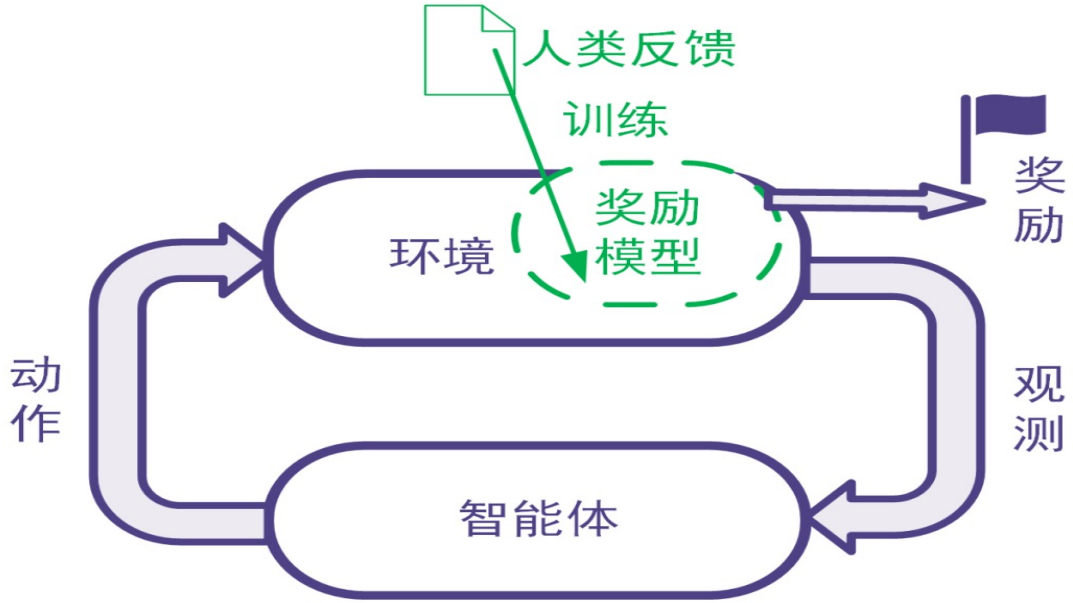
PPO的优势在于其能够在高保真的仿真环境中进行大规模的、安全的训练，然后将学到的策略迁移到真实世界的车辆上，这大大降低了研发成本和道路测试的风险。

章节目录 CONTENTS

大模型强化学习

大模型强化学习(LMRL)是将大规模预训练模型与强化学习(RL)相结合的范式，旨在通过与环境交互和试错，让模型学会最优决策，提升性能与泛化能力。

其核心思想是借鉴人类“奖励”与“惩罚”的学习方式，模型在互动中根据反馈信号调整策略，以最大化累积奖励。



大模型强化学习

基于人类反馈的强化学习（RLHF）技术是大模型与强化学习融合的重要成果之一。这种创新方法将人类的主观判断转化为机器可理解的信号，从而优化语言模型的输出。RLHF的核心思想是通过构建奖励模型来引导语言模型的学习过程，使AI系统能够更好地理解和满足人类的偏好。

RLHF的训练过程可以分为以下三个关键步骤：

- 预训练语言模型： 使用海量文本数据训练出具有良好语言理解能力的基础模型。
- 收集人类反馈数据并训练奖励模型 ： 通过人类标注者对模型输出的排序，训练出能够预测人类偏好的奖励模型。
- 使用强化学习算法微调语言模型 ： 利用奖励模型作为反馈信号，通过强化学习算法优化语言模型的参数。

PPO

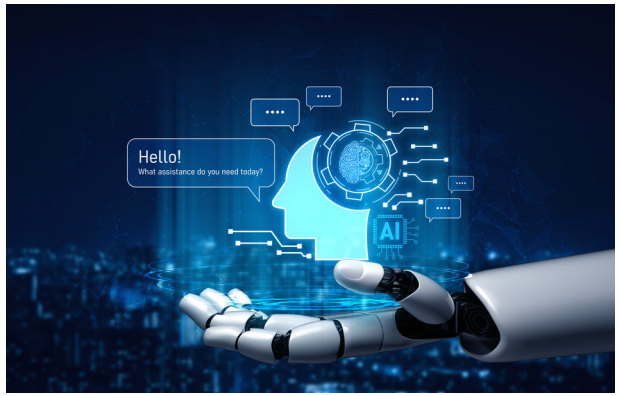
在大模型与强化学习的融合过程中，近端策略优化（PPO）算法作为一种先进的强化学习方法，在大模型训练中发挥着关键作用。PPO算法通过巧妙的设计，有效解决了传统策略梯度方法中存在的数据使用效率低和训练不稳定等问题，为大模型的强化学习带来了显著的性能提升。

DPO

在大模型与强化学习的融合过程中，直接偏好优化（DPO）技术作为一种新兴的方法，为解决传统RLHF方法的复杂性和不稳定性问题提供了新的思路。DPO的核心思想是通过直接里利用偏好数据优化语言模型，避免了显式训练奖励模型的复杂过程。

大模型强化学习

大模型强化学习在自然语言处理、机器人技术、推荐系统的优化、游戏AI等领域得到了广泛大的应用。



自然语言处理
对话系统、机器翻译、文本摘要

机器人技术
机器人控制、路径规划

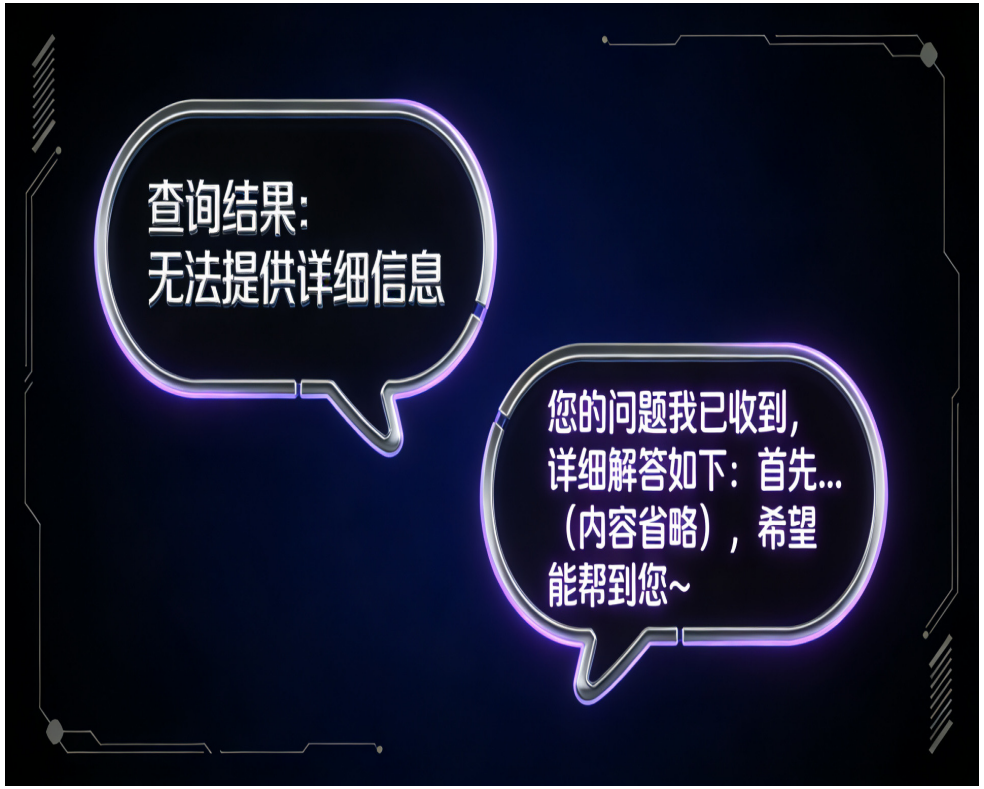
推荐系统
个性化推荐、内容排序

游戏AI
游戏策略、对手模拟

应用场景一：打造更智能的对话助手

优化前

早期对话模型常“答非所问”或“冷冰冰”，机械复述信息，缺乏共情，甚至给出不当回答。



优化后

更有帮助：提供更详细、准确、有逻辑的信息。

更安全：识别并拒绝有害或不良请求。

更有同理心：理解用户情绪，给出恰当回应。

应用场景二：赋能开发者的代码伙伴

优化前

早期的代码生成模型虽然能生成代码，但质量参差不齐。生成的代码可能存在语法错误、逻辑漏洞、效率低下、不符合最佳实践，或根本无法运行

优化后

代码质量更高：生成的代码更正确、更优雅、更符合行业标准。

更安全：能够有效避免一些常见的安全漏洞，如SQL注入、缓冲区溢出、跨站脚本攻击（XSS）等。模型通过学习安全编码规范和常见的漏洞模式，能够在生成代码时主动规避这些风险。

应用场景三：构建能自主决策的AI Agent

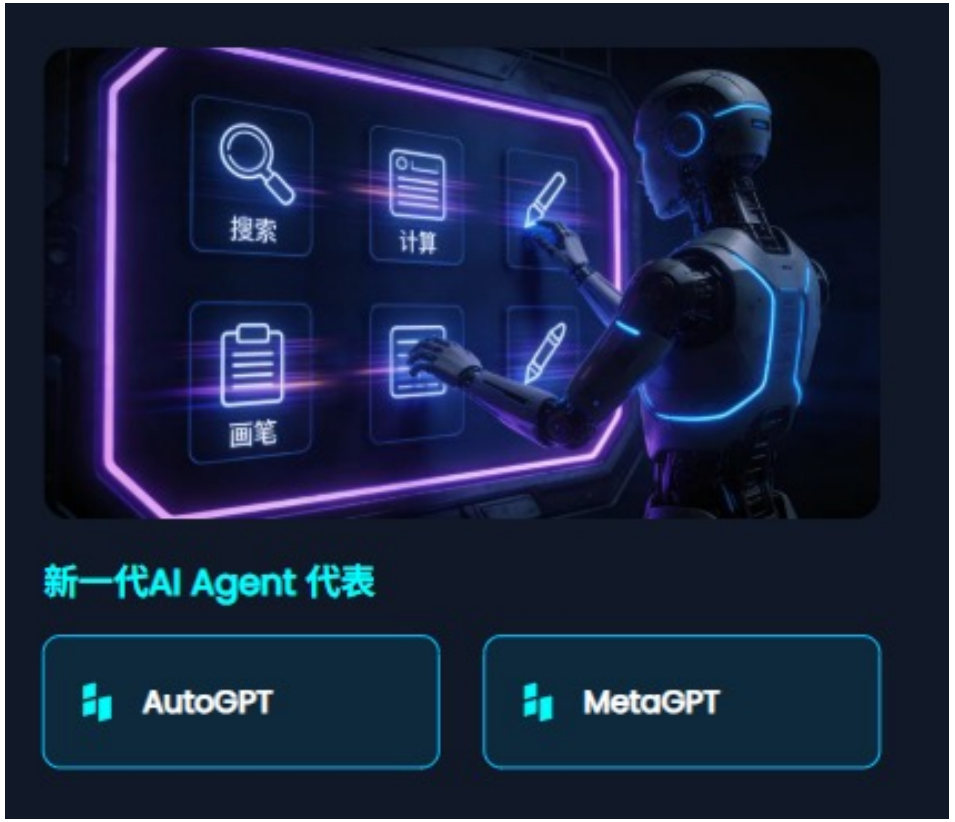
优化前

之前，AI Agent在完成复杂任务时常常显得力不从心。它们可能无法规划出合理的步骤，不知道该在什么时候使用什么工具，或者在遇到挫折时容易陷入死循环。

优化后

规划能力更强：能够将一个复杂的目标，分解成一系列可执行的、循序渐进的步骤。强大的AI Agent能够进行深度的思考和推理。

更能反思：能够对自己的行为进行反思和评估，从失败中吸取教训，并及时调整策略。



未来发展方向

- **RLAIF(AI Feedback)：**让AI自我监督，形成学习闭环，是降低成本实现自动化迭代的重要方向。
- **多模态RLHF:**将技术拓展至图像、音频、视频，让AI更好地理解和创造多感官信息。
- **更高效的算法:**持续改进PPO等算法，并探索全新范式，是推动技术前进的核心动力。
- 技术不断进步，大模型强化学习的未来将充满无限可能。

小结

- 这节课我们首先探讨了Q-learning的局限性，由此引入了深度强化学习的DQN算法，解决高维状态问题。
- DQN算法的两大核心改进方法是经验池和回放和目标网络。
- AC网络采用“决策+评估”分离的网络框架，其中Actor网络输出动作分布参数，解决连续动作问题。
- PPO算法基于AC网络架构解决了”更新步长敏感“与”样本数据重复利用“的问题，成为目前强化学习领域的主流算法。
- 强化学习与大模型结合，通过与环境交互和试错，让模型学会最优决策，提升性能与泛化能力。