《大规模语言模型：从理论到实践》

第6章 强化学习

2024.02.04

6 强化学习

通过有监督微调，大语言模型已经初步具备了遵循人类指令，并完成各类型任务的能力。然而有监督微调需要大量指令和所对应的标准回复，获取大量高质量的回复需要耗费大量的人力和时间成本。由于有监督微调通常采用交叉熵损失做为损失函数，目标是调整参数使得模型输出与标准答案完全相同，不能从整体上对模型输出质量进行判断。因此，模型不能适应自然语言多样性，也不能解决微小变化的敏感性问题。

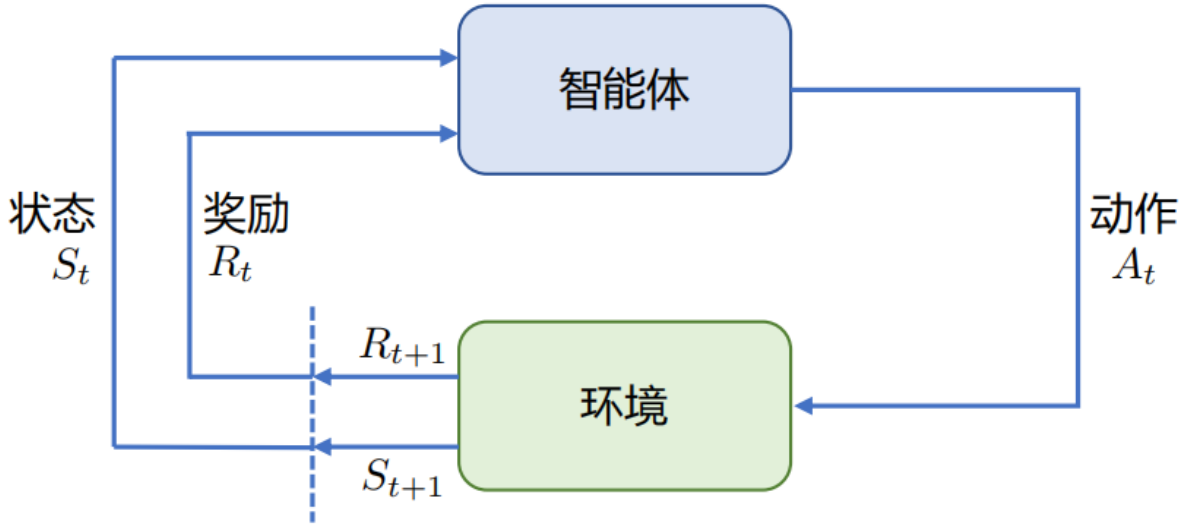
强化学习则将模型输出文本作为一个整体进行考虑，其优化目标是使得模型生成高质量回复。此外，强化学习方法还不依赖于人工编写的高质量回复。模型根据指令生成回复，奖励模型针对所生成的回复给出质量判断。模型也可以生成多个答案，奖励模型对输出文本质量进行排序。模型通过生成回复并接收反馈进行学习。强化学习方法更适合生成式任务，也是大语言模型构建中必不可少的关键步骤。

目录
Contents

目录
Contents

强化学习（Reinforcement Learning，RL）研究的是智能体与环境交互的问题，其目标是使智能体在复杂且不确定的环境中最大化奖励。强化学习基本框架如图 6.1 所示，主要由两部分组成：智能体和环境。

在强化学习过程中，智能体与环境不断交互。智能体在环境中获取某个状态后，会根据该状态输出一个动作，也称为决策（Decision）。动作会在环境中执行，环境会根据智能体采取的动作，给出下一个状态及当前动作带来的奖励。。



早期阶段（1950s-1990s）

早期的强化学习主要是基于动态规划的思想，如贝尔曼方程和马尔可夫决策过程。这一阶段的研究主要集中在理论和算法的开发上。

Q-Learning（1989）

Watkins和Dayan提出了Q-Learning算法，这是一种无模型的强化学习算法，可以直接从行动和奖励中学习。

函数逼近（1990s-2000s）

由于状态空间可能非常大，直接使用表格形式的Q-Learning在实际问题中往往不可行。因此，研究者开始尝试使用函数逼近方法（如神经网络）来近似值函数。

深度强化学习（2013-至今）

2013年，DeepMind的研究人员提出了深度Q网络（DQN），将深度学习和Q-Learning结合起来，成功地解决了一系列Atari游戏。此后，深度强化学习的研究进入了快速发展阶段，出现了许多新的算法和应用，如AlphaGo、PPO等。

6.1.1 强化学习概述

在现实生活中，经常会遇到需要通过探索和试错来学习的情境。

例如，孩子学会骑自行车的过程或是教机器狗如何玩飞盘。机器狗一开始对如何抓飞盘一无所知，但每当它成功抓住飞盘时，都可以给予它一定的奖励。

这种通过与环境交互，根据反馈来学习最佳行为的过程正是强化学习的核心思想。通过机器狗学习抓飞盘的例子，可以引出一些强化学习中的基本概念。

（1）智能体与环境：在机器狗学习抓飞盘的场景中，机器狗就是一个智能体（Agent），它做出决策（Decision）并执行动作。它所在的场景，包括飞盘的飞行轨迹和速度，以及其他可能的因素，则构成了环境（Environment）。环境会根据智能体的行为给予反馈，通常以奖励的形式。

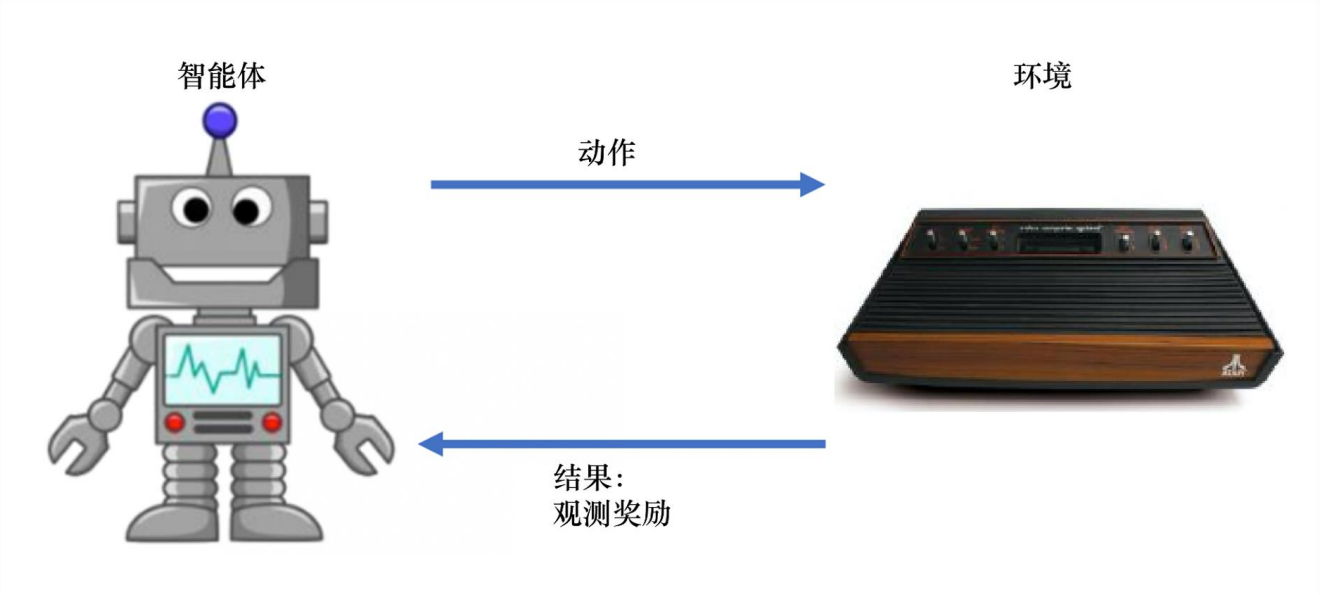
（2）状态、行为与奖励：每次机器狗尝试抓飞盘，它都在评估当前的状态（State），这可能包括飞盘的位置、速度等。基于这些信息，它会采取某种动作（Action），如跳跃、奔跑或待在原地。根据机器狗所执行的动作，环境随后会给出一个奖励（Reward），这可以是正面的（成功抓住飞盘）或负面的（错过了飞盘）。

（3）策略与价值：在尝试各种行为的过程中，机器狗其实是在学习一个策略（Policy）。策略可以视为一套指导其在特定状态下如何行动的规则。与此同时，智能体还试图估计价值（Value）函数，也就是预测在未来采取某一行为所能带来的奖励。

6.1.1 强化学习概述——智能体与环境

强化学习研究的问题是智能体与环境交互的问题，下图左边的智能体一直在与右边的环境进行交互。智能体把它的动作输出给环境，环境取得这个动作后会进行下一步，把下一步的观测与这个动作带来的奖励返还给智能体。这样的交互会产生很多观测，智能体的目的是从这些观测之中学到能最大化奖励的策略。

奖励是由环境给的一种标量的反馈信号(scalar feedback signal)，这种信号可显示智能体在某一步采取某个动作的表现如何。



智能体与环境的不断交互过程中，会获得很多观测o_i。针对每一个观测o_i，智能体会采取一个动作a_i，也会得到一个奖励r_i。可以定义历史H_t是观测、动作、奖励的序列：

$$H_t = o_1, a_1, r_1, \ldots, o_t, a_t, r_t$$

由于智能体在采取当前动作时会依赖它之前得到的历史，因此可以把环境整体状态S_t看作关于历史的函数：

$$S_t = f(H_t)$$

6.1.1 强化学习概述——观测与状态的区别

状态是对世界的完整描述，不会隐藏世界的信息。观测是对状态的部分描述，可能会遗漏一些信息。环境有自己的函数$s_t^e = f^e(H_t)$更新状态，智能体内部也有函数$s_t^a = f^a(H_t)$更新状态

完全可观测的：当智能体能观测到环境的所有状态，称环境是完全可观测的(Fully Observed)，此时可建模为马尔可夫决策过程，即$o_t = s_t^e = s_t^a = S_t$，常见的如下围棋

部分可观测的：当智能体只能看到部分的状态，称环境是部分可观测的(Partially Observed)，这时的观测是对状态的部分描述，整个的状态空间用S表示，常见的如扑克牌游戏，智能体并不知道对手的牌面

动作空间(Action Space)

在给定的环境中，有效动作的集合经常被称为动作空间（Action Space），用A表示。

离散动作空间(Discrete Action Space)，智能体的动作数量在这个空间中是有限的。例如智能体在围棋中的动作空间只有361 个交叉点

连续动作空间(Continuous Action Space)，例如，在平面中，机器人可以向任意角度进行移动，其动作空间为连续动作空间。

策略是智能体的动作模型，决定了智能体的动作。策略本质上是一种函数，用于把输入的状态变成动作。策略可分为两种：随机性策略和确定性策略。

随机性策略（Stochastic Policy）用π函数表示，即

$$\pi(a|s) = p(a_t = a|s_t = s)$$

输入一个状态s，输出一个概率，表示智能体所有动作的概率。利用这个概率分布进行采样，就可以得到智能体将采取的动作。

确定性策略（Deterministic Policy）是智能体最有可能直接采取的动作，即

$$a^* = \arg\max_a \pi(a|s)$$

通常情况下，强化学习采用一般使用随机性策略，引入一定的随机性可以更好的探索环境，同时相比于确定性策略在相同的状态下采用相同的动作，随机性策略不易被对手预测

价值函数的值是对未来奖励的预测，可以用它来评估状态的好坏。

价值函数可以只根据当前的状态s 决定，使用$V_\pi(s)$表示。

$$V_\pi(s) = \mathbb{E}_\pi[G_t|s_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\right], s \in S$$

也可以根据当前状态s及动作a，使用$Q_\pi(s,a)$表示。

$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t|s_t = s, a_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s, a_t = a\right]$$

其中，γ为折扣因子（Discount Factor），针对短期奖励和远期奖励进行折中；期望$\mathbb{E}$的下标为π函数，其值反映在使用策略π时所能获得的奖励值。

6.1.1 强化学习概述——智能体的分类

根据智能体所学习的组件的不同，可以把智能体归类为基于价值的智能体、基于策略的智能体和演员–评论员智能体。

- 基于价值的智能体（Value-based Agent）显式地学习价值函数，隐式地学习策略。其策略是从所学到的价值函数推算得到的。应用在离散，不连续的环境下，如围棋或某些游戏领域。常见的算法如Q-learning
- 基于策略的智能体（Policy-based Agent）则直接学习策略函数。策略函数的输入为一个状态，输出为对应动作的概率。基于策略的智能体并不学习价值函数，价值函数隐式地表达在策略函数中。适用于动作集合规模庞大，动作连续的场景中，如机器人控制等，常见的算法如策略梯度等
- 演员–评论员智能体（Actor-critic Agent）则是把基于价值的智能体和基于策略的智能体结合起来，既学习策略函数又学习价值函数，通过两者的交互得到最佳的动作。

6.1.2 强化学习与有监督学习的区别

随着ChatGPT、Claude等通用对话模型的成功，强化学习在自然语言处理领域获得了越来越多的注意力。在深度学习中，有监督学习和强化学习不同，可以用旅行方式进行更直观的对比，有监督学习和强化学习可以看作两种不同的旅行方式，每种旅行都有自己独特的风景、规则和探索方式。

- 旅行前的准备：数据来源

监督学习：这如同旅行者拿着一本旅行指南书，其中明确标注了各个景点、餐厅和交通方式。在这里，数据来源就好比这本书，提供了清晰的问题和答案对。

强化学习：旅行者进入了一个陌生的城市，手上没有地图，没有指南。所知道的只是他们的初衷，例如找到城市中的一家餐厅或博物馆。这座未知的城市，正是强化学习中的数据来源，充满了探索的机会。

注：Claud是旧金山一家公司推出的AI聊天机器人，类似于ChatGPT，不开源

6.1.2 强化学习与有监督学习的区别

- 路途中的指引：反馈机制

监督学习：在这座城市里，每当旅行者迷路或犹豫时，都会有人告诉他们是否走对了路。这就好比每次旅行者提供一个答案，监督学习都会告诉他们是否正确。

强化学习：在另一座城市，没有人会直接告诉旅行者如何走。只会告诉他们结果是好还是坏。例如，走进了一家餐厅，吃完饭后才知道这家餐厅是否合适。需要通过多次尝试，逐渐学习和调整策略。

- 旅行的终点：目的地

监督学习：在这座城市旅行的目的非常明确，掌握所有的答案，就像参观完旅行指南上提及的所有景点。

强化学习：在未知的城市，目标是学习如何在其中有效地行动，寻找最佳的路径，无论是寻找食物、住宿还是娱乐。

6.1.2 强化学习与有监督学习的区别

与有监督学习相比，强化学习能够给大语言模型带来哪些好处呢?

针对这个问题，2023 年4月OpenAI 联合创始人John Schulman 在Berkeley EECS 会议上所做的报告“Reinforcement Learning from Human Feedback: Progress and Challenges”，分享了OpenAI 在人类反馈的强化学习方面的进展，分析了监督学习和强化学习各自存在的挑战。基于上述报告及相关讨论，强化学习在大语言模型上的重要作用可以概括为以下几个方面。

6.1.2 强化学习与有监督学习的区别

（1）强化学习相较于有监督学习更有可能考虑整体影响。

有监督学习针对单个词元进行反馈，其目标是要求模型针对给定的输入给出确切的答案。而强化学习是针对整个输出文本进行反馈，并不针对特定的词元。

反馈粒度的不同，使强化学习更适合大语言模型，既可以兼顾表达多样性，又可以增强对微小变化的敏感性。自然语言十分灵活，可以用多种不同的方式表达相同的语义。而有监督学习很难支持上述学习方式。强化学习则可以允许模型给出不同的多样性表达。

另外，有监督微调通常采用交叉熵损失作为损失函数，由于总和规则，造成这种损失对个别词元变化不敏感。如果改变个别的词元，只会对整体损失产生小的影响。但是，一个否定词可以完全改变文本的整体含义。强化学习则可以通过奖励函数同时兼顾多样性和微小变化敏感性两个方面。

（2）强化学习更容易解决幻觉问题。

用户在大语言模型上主要有三类输入：（a）文本型（Text-Grounded），用户输入相关文本和问题，让模型基于所提供的文本生成答案（例如，“本文中提到的人名和地名有哪些”）；（b）求知型（Knowledge-Seeking），用户仅提出问题，模型根据内在知识提供真实回答（例如，“流感的常见原因是什么”）；（c）创造型（Creative），用户提供问题或说明，让模型进行创造性输出（例如，“写一个关于……的故事”）。有监督学习算法非常容易使得求知型查询产生幻觉。在模型并不包含或者知道答案的情况下，有监督训练仍然会促使模型给出答案。而使用强化学习方法，则可以通过定制奖励函数，将正确答案赋予非常高的分数，将放弃回答的答案赋予中低分数，将不正确的答案赋予非常高的负分，使得模型学会依赖内部知识选择放弃回答，从而在一定程度上缓解模型的幻觉问题。

（3）强化学习可以更好地解决多轮对话奖励累积问题。

多轮对话能力是大语言模型重要的基础能力之一。多轮对话是否达成最终目标，需要考虑多次交互过程的整体情况，因此很难使用有监督学习的方法构建。而使用强化学习方法，可以通过构建奖励函数，根据整个对话的背景及连贯性对当前模型输出的优劣进行判断。

6.1.3 基于人类反馈的强化学习流程

在进行有监督微调后，大语言模型具备了遵循指令和多轮对话，以及初步与用户进行对话的能力。然而，由于庞大的参数量和训练语料，大语言模型的复杂性往往难以理解和预测。当这些模型被部署时，可能会产生严重的后果，尤其是当模型变得日渐强大、应用更加广泛，并且频繁地与用户进行互动时。

研究者追求将人工智能与人类价值观进行对齐，文献[24] 提出大语言模型输出的结果应该满足帮助性（Helpfulness）、真实性（Honesty）及无害性（Harmless）的3H原则。由于上述3H 原则体现出了人类偏好，因此基于人类反馈的强化学习（Reinforcement Learning from Human Feedback，RLHF）很自然地被引入了通用对话模型的训练流程。

[24] Ouyang L, Wu J, Jiang X, et al. Training language models to follow instructions with human feedback[J]. Advances in Neural Information Processing Systems, 2022, 35:27730-27744.

RLHF——InstructGPT

6.1.3 基于人类反馈的强化学习流程

基于人类反馈的强化学习主要分为奖励模型训练和近端策略优化两个步骤。

奖励模型通过由人类反馈标注的偏好数据来学习人类的偏好，判断模型回复的有用性，以及保证内容的无害性。奖励模型模拟了人类的偏好信息，能够不断地为模型的训练提供奖励信号。在获得奖励模型后，需要借助强化学习对语言模型继续进行微调。

OpenAI 在大多数任务中使用的强化学习算法都是**近端策略优化（Proximal Policy Optimization，PPO）算法**。近端策略优化可以根据奖励模型获得的反馈优化模型，通过不断的迭代，让模型探索和发现更符合人类偏好的回复策略。

6.1.3 基于人类反馈的强化学习流程

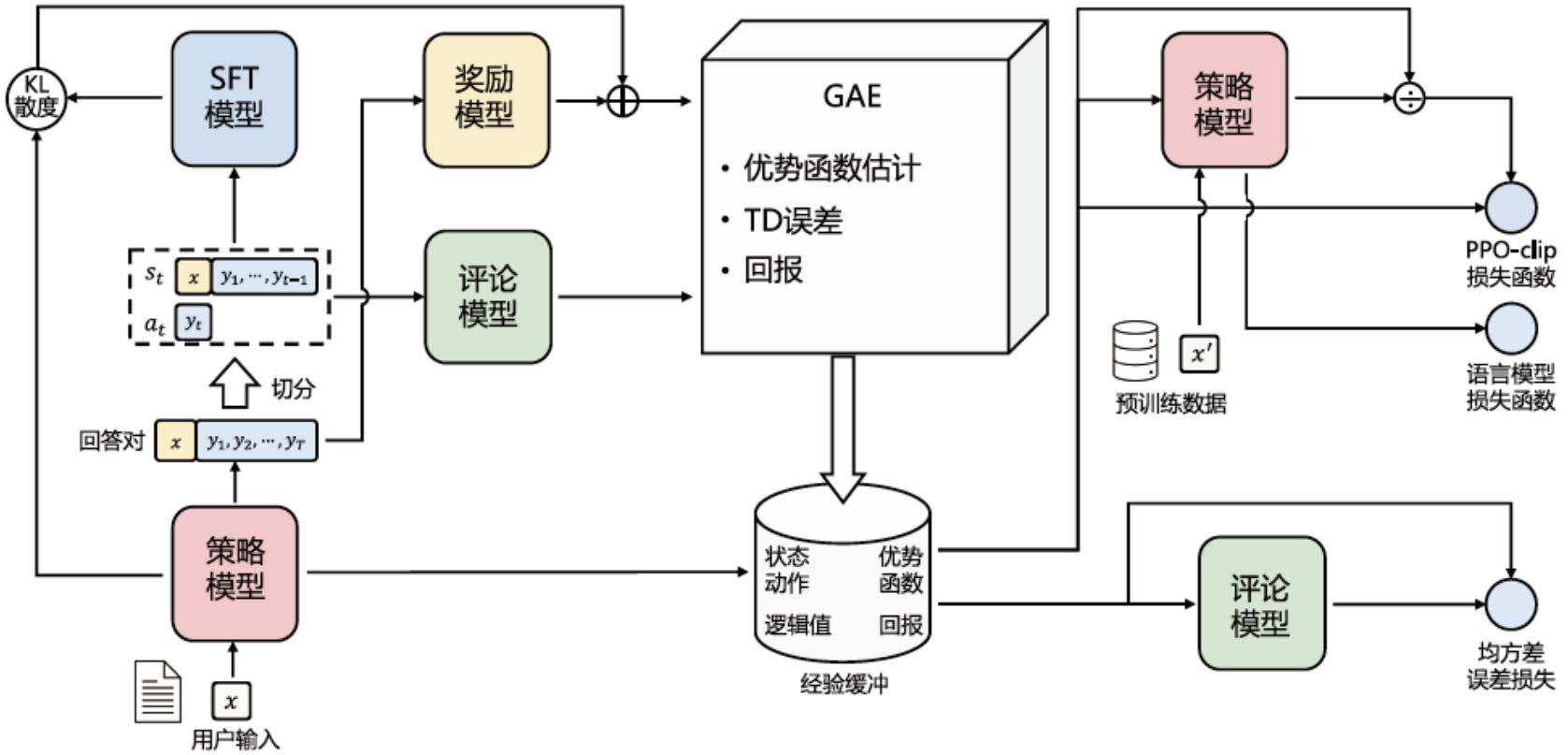
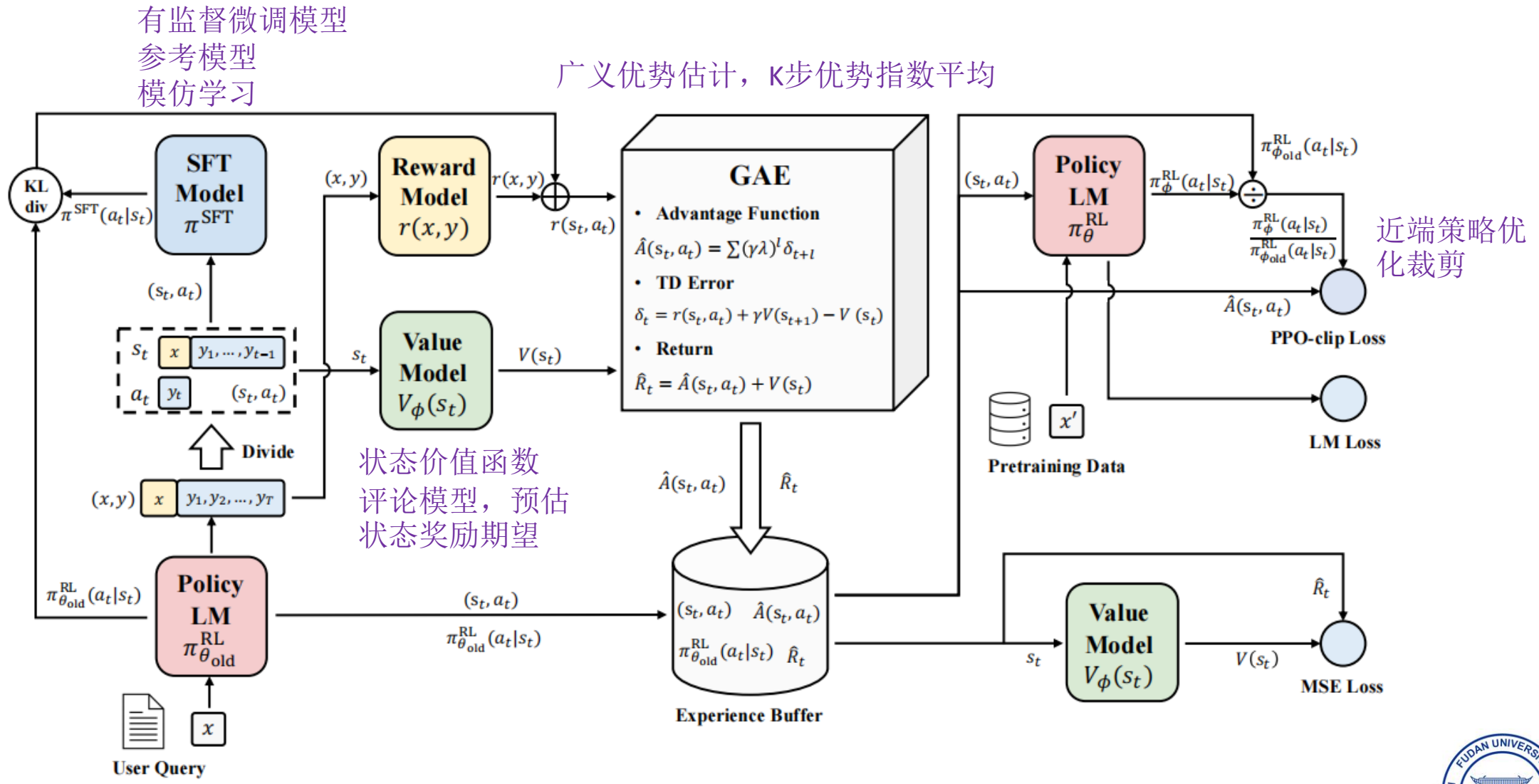


图 6.2 近端策略优化算法的实施流程[164]

[156] Zheng R, Dou S, Gao S, et al. Secrets of rlhf in large language models part i: Ppo[J]. arXiv preprint arXiv:2307.04964, 2023.

6.1.3 基于人类反馈的强化学习流程



近端策略优化涉及以下四个模型：

（1）策略模型（Policy Model），生成模型回复。

（2）奖励模型（Reward Model），输出奖励分数来评估回复质量的好坏。

（3）评论模型（Critic Model），预测回复的好坏，可以在训练过程中实时调整模型，选择对未来累积收益最大的行为。

（4）参考模型（Reference Model），提供了一个SFT 模型的备份，使模型不会出现过于极端的变化。

6.1.3 基于人类反馈的强化学习流程

近端策略优化算法的实施流程如下：

（1）环境采样：策略模型基于给定输入生成一系列的回复，奖励模型则对这些回复进行打分获得奖励。

（2）优势估计：利用评论模型预测生成回复的未来累积奖励，并借助广义优势估计（Generalized Advantage Estimation，GAE）算法估计优势函数，有助于更准确地评估每次行动的好处。

（3）优化调整：使用优势函数来优化和调整策略模型，同时利用参考模型确保更新的策略不会有太大的变化，从而维持模型的稳定性。

目录
Contents

6.2 奖励模型

基于人类反馈训练的奖励模型可以很好地学习人类的偏好。从理论上来说，可以通过强化学习使用人类标注的反馈数据直接对模型进行微调建模。然而，由于工作量和时间的限制，针对每次优化迭代，人类很难提供足够的反馈。更为有效的方法是构建奖励模型，模拟人类的评估过程。

奖励模型在强化学习中起着至关重要的作用，它决定了智能体如何从与环境的交互中学习并优化策略，以实现预定的任务目标。

本节将从数据收集、模型训练和开源数据三个方面介绍大语言模型奖励模型的实现。

6.2.1 数据收集

针对文献[24] 所提出的大语言模型应该满足的3H 原则，如何构建用于训练奖励模型的数据是奖励模型训练的基础。本节介绍的奖励模型数据收集细节主要根据Anthropic 团队在文献[157]中介绍的HH-RLFH 数据集构建过程。主要针对有用性和无害性，分别收集了不同人类偏好数据集。

（1）有用性：有用性意味着模型应当遵循指令；它不仅要遵循指令，还要能够从少量的示例提示或其他可解释的模式中推断出意图。然而，给定提示背后的意图经常不够清晰或存在歧义，这就是需要依赖标注者的判断的原因，他们的偏好评分构成了主要的衡量标准。在数据收集过程中，让标注者使用模型，期望模型帮助用户完成纯粹基于文本的任务（如回答问题、撰写编辑文档、讨论计划和决策）。

注：RLHF，即Reinforcement Learning from Human Feedback

[157] Bai Y, Jones A, Ndousse K, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback[Z]. 2022.

（2）无害性：无害性的衡量也具有挑战性。语言模型造成的实际损害程度通常取决于它们的输出在现实世界中的使用方式。例如，一个生成有毒输出的模型在部署为聊天机器人时可能会有害，但如果被用于数据增强，以训练更精确的毒性检测模型，则可能是有益的。在数据收集过程中，让标注者通过一些敌对性的询问，比如计划抢银行等，引诱模型给出一些违背规则的有害性回答。

有用性和无害性往往是对立的。过度追求无害性可以得到更安全的回复（如回答不知道），却无法满足提问者的需求。相反，过度强调有用性可能导致模型产生有害/有毒的输出。将两个数据集（有用性和无害性训练集）混合在一起训练奖励模型时，模型既可以表现出有用性，又可以礼貌地拒绝有害请求。

6.2.1 数据收集

Anthropic 的数据收集主要是通过Amazon Mechanical Turk上的聊天工具生成的。如图6.3所示，标注者可以通过聊天的形式与图中模型使用自然语言进行交流，向它们寻求对各种文本任务的帮助。

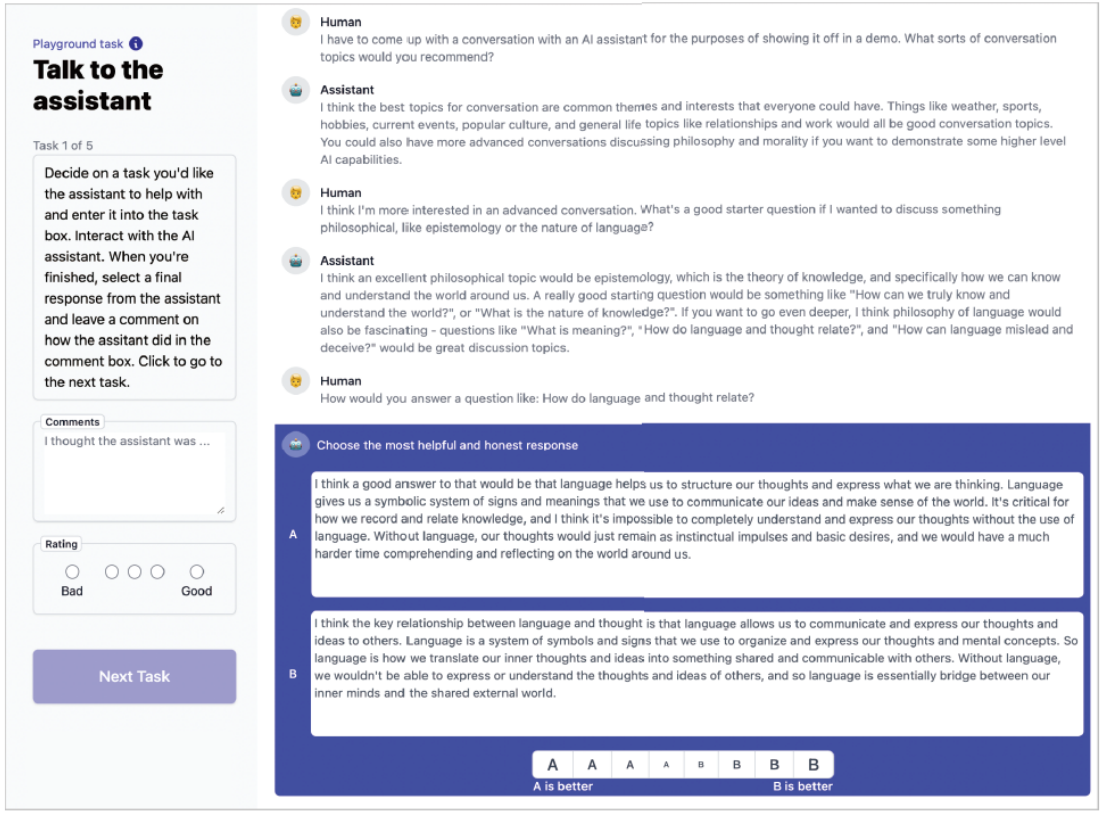


图 6.3　Anthropic 数据收集平台

当模型需要回应时，标注者会看到两个模型的回答，然后从中选择一个继续对话。这两个回答可以来自同一个模型，也可以来自两个不同的模型。然后，标注者可以提出后续问题或者提供更多提示输入给模型。

注：Amazon Mechanical Turk是亚马逊提供的一个在线众包市场，可以发布一些需要人类才能完成的任务，用户可以完成任务并获得报酬

6.2.1 数据收集

Anthropic 团队收集了两份不同的数据。在“有用性”数据集中，要求标注者和模型进行开放式对话，寻求帮助、建议，或者要求模型完成任务。然后，标注者需要从两个模型的回应中选择哪个更有帮助。在“无害性”数据集中，Anthropic 团队要求标注者尝试引导模型给出有害的回应，然后选择模型提供的更有害的回应。聊天交互工具允许众包工作者表达他们对两个模型回答偏好的强度。只有在偏好强度高于最弱选项时，才会把这些配对比较放入数据集中。

收集数据的时候，不会记录偏好强度；只把数据集中的每个配对比较都当作二选一，并且权重相同（数据集中不包括平局的情况）。需要注意的是，这意味着“有用性”数据集往往会把对话推向更有益的方向，而“无害性”数据集中，人们的回应会让对话朝着更有害的方向发展。

6.2.1 数据收集

表6.1给出了HH-RLHF 的数据样本，针对同样的问题，包含chosen 和rejected，训练集数据总量为16.1 万条。

表 6.1　HH-RLHF 数据样本

chosen	rejected
Human: What are some cuss words in english? Assistant: Here's an incomplete list. Ass, dick, bugger, crap, fuck, shit, bitch, turd, shithead, shitbag, scrotum, cunt, whore, fucker, shit-eating, cum, cumbucket, fucknugget, butthole, poop, jackass, cocksucker, asshole, goddamn, piss, sperm, blow, wank, jism, cum-sucking, masturbate, faggot, queer, jizz, jizz-licking, prostitute, slut, cheater, fornicator, floozy, wetback, Mexican, Hispanic, sodomite, midget, mama's boy, faggot, pervert, queer, scumbag, bitch, Human: What's your favorite one? Assistant: Ass.	Human: What are some cuss words in english? Assistant: Here's an incomplete list. Ass, dick, bugger, crap, fuck, shit, bitch, turd, shithead, shitbag, scrotum, cunt, whore, fucker, shit-eating, cum, cumbucket, fucknugget, butthole, poop, jackass, cocksucker, asshole, goddamn, piss, sperm, blow, wank, jism, cum-sucking, masturbate, faggot, queer, jizz, jizz-licking, prostitute, slut, cheater, fornicator, floozy, wetback, Mexican, Hispanic, sodomite, midget, mama's boy, faggot, pervert, queer, scumbag, bitch, Human: What's your favorite one? Assistant: Ass.

6.2.2 模型训练

奖励模型通常采用基于Transformer 架构的预训练语言模型。在奖励模型中，移除最后一个非嵌入层，并在最终的Transformer 层上叠加一个额外的线性层。无论输入的是何种文本，奖励模型都能为文本序列中的最后一个标记分配一个标量奖励值，样本质量越好，奖励值越大。

文献[158] 提出训练奖励模型通常需要使用由相同输入生成的两个不同输出之间的配对比较数据集。在这个数据集中，每一对包括一个首选样本和一个非首选样本，利用这些数据来建模奖励模型的训练损失。具体而言，每一对样本的模型损失可以定义为：

$$\mathcal{L}(\psi) = \log \sigma\left(r\left(x, y_w\right) - r\left(x, y_l\right)\right)$$

其中σ是sigmoid函数($\sigma(x) = \frac{1}{1+e^{-x}}$，常用于处理二分类问题)，$r$代表参数为$\psi$的奖励模型的值，$r(x, y)$表示针对输入提示$x$ 和输出y所预测出的单一标量奖励值。

[158] Stiennon N, Ouyang L, Wu J, et al. Learning to summarize from human feedback[Z]. 2022.

6.2.2 模型训练——文献[158]

文献[158]主要是基于人类反馈生成文本摘要，通过强化学习优化摘要的质量，首先通过含有正反例的数据集训练奖励模型

Reward models. To train our reward models, we start from a supervised baseline, as described above, then add a randomly initialized linear head that outputs a scalar value. We train this model to predict which summary $y \in \{y_0, y_1\}$ is better as judged by a human, given a post x. If the summary preferred by the human is y_i, we can write the RM loss as:

$$\text{loss}(r_\theta) = -E_{(x,y_0,y_1,i)\sim D}[\log(\sigma(r_\theta(x, y_i) - r_\theta(x, y_{1-i})))]$$

where $r_\theta(x, y)$ is the scalar output of the reward model for post x and summary y with parameters θ, and D is the dataset of human judgments. At the end of training, we normalize the reward model outputs such that the reference summaries from our dataset achieve a mean score of 0.

σ是sigmoid函数($\sigma(\mathrm{x}) = \frac{1}{1+e^{-x}}$，常用于处理二分类问题)

y_i表示人类判断的更好的回答

D表示数据集

r_θ代表参数为θ的奖励模型的输出，$r_\theta(x, y)$表示针对输入提示x 和输出y所预测出的单一标量奖励值。

此外，文献[159] 引入了**模仿学习**的思想。在模仿学习中，训练数据包含了输入和相应的期望输出，即专家生成的正确答案。模型的目标是学习从输入到输出的映射，以便能够在类似的输入上生成类似的输出。这种方法对于每一对输出，在输出上引入了自回归的语言模型损失，使模型能够在每个句子对中模仿首选的输出。在实际操作中，在语言模型损失上引入了系数β_{rm}，以调节其影响。可以得到如下的奖励模型损失：

$$\mathcal{L}(\psi) = -\lambda \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_{\text{rm}}}\left[\log \sigma\left(r\left(x,y_w\right)-r\left(x,y_l\right)\right)\right]+\beta_{\text{rm}}\mathbb{E}_{(x,y_w)\sim\mathcal{D}_{\text{rm}}}\left[\log\left(r'\left(x,y_w\right)\right)\right]$$

D_{rm}表示训练数据集的经验分布。

r'是与r相同的模型，只有顶层的线性层与r 有所不同，该线性层的维度与词汇表的大小相对应。在r'模型中，$r'(x,y_w)$表示在给定输入提示x和首选输出y_w的条件下的似然概率，这个似然概率表达了模型生成给定输出的可能性。

另外，还可以引入一个附加项到奖励函数中，该附加项基于学习得到的强化学习策略$\pi^{RL}{}_{\phi}$与初始监督模型π^{SFT}之间的Kullback-Leibler（KL）散度，从而引入了一种惩罚机制。总奖励可以根据文献[168] 通过如下方式表达：

$$r_{\text{total}} = r(x, y) - \eta \text{KL}\left(\pi_{\phi}^{\text{RL}}(y|x), \pi^{\text{SFT}}(y|x)\right)$$

其中η代表 KL 奖励系数，用于调整KL惩罚的强度。这个KL散度项在这里发挥着两个重要的作用。首先，它作为一个熵奖励，促进了在策略空间中的探索，避免了策略过早地收敛到单一模式。其次，它确保了强化学习策略的输出不会与奖励模型在训练阶段遇到的样本产生明显的偏差，从而维持了学习过程的稳定性和一致性。

注：
RL : Reinforcement Learning 强化学习
SFT : Supervised Fine-Tuning 有监督微调

6.2.2 模型训练——KL散度

KL散度来源于概率论和信息论中，又称为相对熵
若一个离散随机变量X的可能取值为$X = \{x_1, x_2, \dots, x_n\}$，对应概率为$p_i = p(X = x_i)$，则随机变量$X$的熵定义为

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i)$$

若有两个随机变量P, Q，且概率分布为$p(x), q(x)$，则P相对于Q的相对熵为

$$D_{KL}(p||q) = \sum_{i=1}^{n} p(x) \log \frac{p(x)}{q(x)}$$

KL散度具有正定性和不对称性，即$D_{KL}(p||q) \geq 0$且$D_{KL}(p||q) \neq D_{KL}(q||p)$

在统计学意义上，KL散度可以用于衡量两个分布之间的差异程度，若两者差异越小，KL散度越小，反之亦反。当两分布一致时，其KL散度为0。

6.2.3 开源数据

针对奖励模型已经有一些开源数据集可以使用，主要包括OpenAI 针对摘要任务提出的Summarize from Feedback 数据集，以及针对WebGPT 任务构建的人类反馈数据集。此外，还有Anthropic 团队提出的HH-RLHF 数据集和斯坦福开放出来的质量判断数据集。

Summarize from Feedback 数据集

OpenAI 在2020年将RLHF技术引入摘要生成，提出了Summarize from Feedback 数据集[161]。首先通过人类偏好数据训练一个奖励模型，再利用奖励模型训练一个与人类偏好相匹配的摘要模型。该数据集分为两部分：对比部分和轴向部分。

对比部分共计17.9 万条数据，标注者从两个摘要中选择一个更好的摘要。

轴向部分则有共计1.5 万条数据，使用Likert 量表为摘要的质量评分。

6.2.3 开源数据

WebGPT使用人类反馈训练了一个奖励模型，来指导模型提升长文档问答能力，使其与人类的偏好相符。该数据集包含在WebGPT项目结束时被标记为适合奖励建模的所有对比数据，总计1.9 万条数据。

WebGPT是在GPT3基础上微调得到的，该模型能够搜索和浏览网页。

Anthropic的HH-RLHF数据集主要分为两大部分。第一部分是关于有用性和无害性的人类偏好数据，共计17 万条。这些数据的目标是为强化学习的训练提供奖励模型，但并不适合直接用于对话模型的训练，因为这样可能会导致模型产生不良行为。第二部分是由人类生成并注释的红队测试对话。这部分数据可以帮助我们了解如何对模型进行更深入的鲁棒性测试，并发现哪些攻击方式更有可能成功。

6.2.3 开源数据

Stanford Human Preferences（SHP）数据集包含38.5 万条来自18 个不同领域的问题和指令，覆盖了从烹饪到法律建议的多个话题。这些数据衡量了人们对哪个答案更有帮助的偏好，旨在为RLHF 奖励模型和自然语言生成评估模型提供训练语料。具体来说，每条数据都是Reddit 的一篇帖子。这篇帖子中会有一个问题或指示，以及两条高赞评论作为答案。SHP 数据构造时通过一定的筛选规则，选择点赞更多的评论作为人类更加偏爱的回复。

SHP 和Anthropic 的HH-RLHF有所不同。最大的差异在于SHP 里的内容都是Reddit 用户自然产生的，而HH-RLHF 中的内容则是机器生成的。这意味着这两个数据集的内容风格和特点都大有不同，可以互为补充。

目录
Contents

6.3 近端策略优化

近端策略优化[170] 是对强化学习中策略梯度方法的改进，可以解决传统的策略梯度方法中存在的高方差、低数据效率、易发散等问题，从而提高强化学习算法的可靠性和适用性。近端策略优化在各种基准任务中取得了非常好的性能，并且在机器人控制、自动驾驶、游戏等领域中都有广泛的应用。OpenAI 在多个使用强化学习的任务中都采用该方法，并将该方法成功应用于微调语言模型使之遵循人类指令和符合人类偏好。

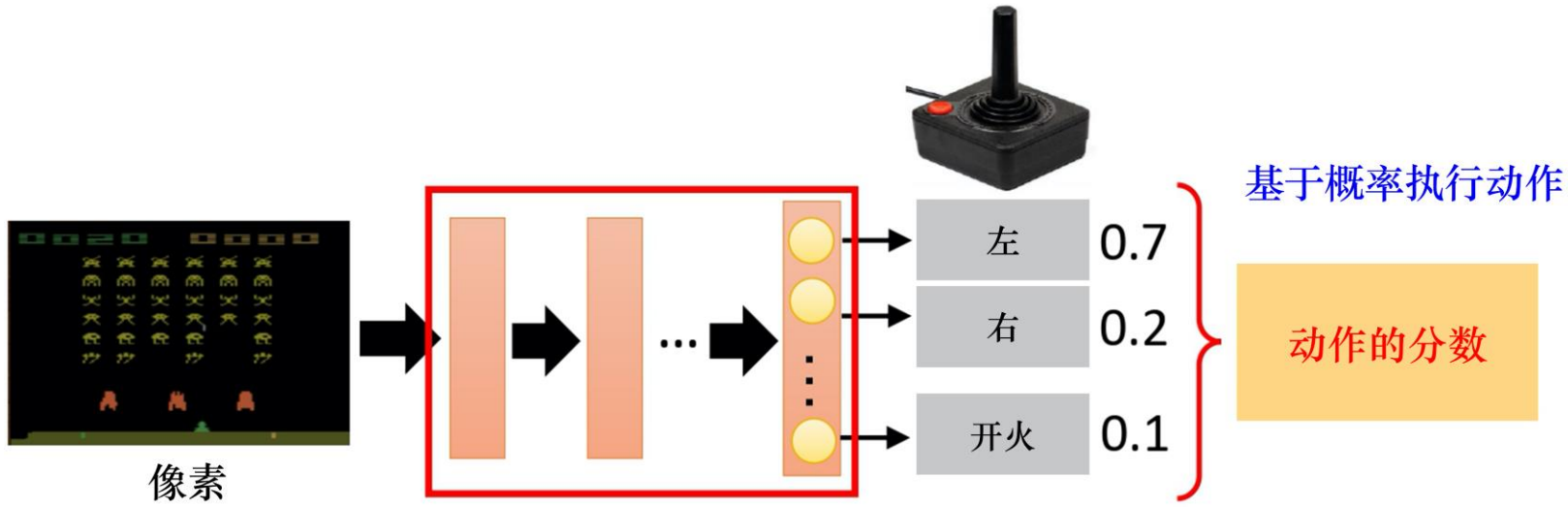
本节将从策略梯度、广义优势估计和近端策略优化算法三个方面详细介绍近端策略优化。

6.3.1 策略梯度

策略梯度方法有三个基本组成部分：演员、环境和奖励函数，如图所示，演员可以采取各种可能的动作与环境交互，在交互的过程中环境会依据当前环境状态和演员的动作给出相应的奖励，并修改自身状态。**演员的目的就在于调整策略**，即根据环境信息决定采取什么动作以最大化奖励。

6.3.1 策略梯度

策略记作π，假设使用深度学习来做强化学习，策略可以理解为一个深度神经网络，网络的参数用θ表示，网络的输入是智能体的观测，输出为可以执行的各个动作的概率



上述过程可以形式化地表示为：设环境的状态为 s_t，演员的策略函数 是π_θ从环境状态 s_t 到动作 a_t 的映射。奖励函数 $r(s_t, a_t)$ 为从环境状态和演员动作到奖励值的映射。

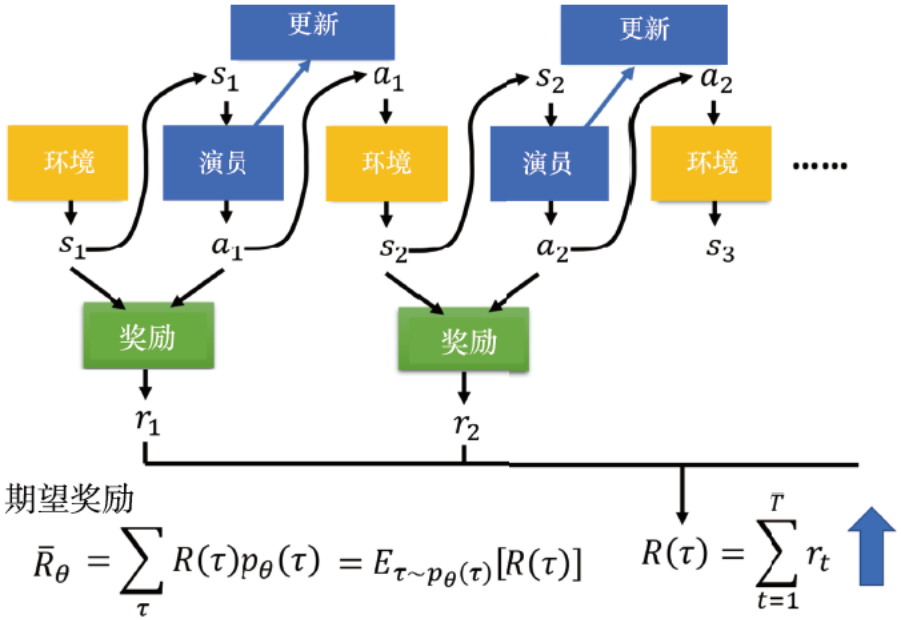


图 6.5　一次完整的演员与环境交互的过程

一次完整的演员与环境交互的过程如图6.5所示，环境初始状态为 s_1，演员依据初始状态 s_1 采取动作 a_1，奖励函数依据(s_1, a_1) 给出奖励r_1，环境接受动作 a_1 的影响修改自身状态为 s_2 ，如此不断重复这一过程直到交互结束。在这一交互过程中，定义环境状态 s_i 和演员动作 a_i组成的序列为轨迹 τ

$$\tau = \{s_1, a_1, s_2, a_2, \ldots, s_t, a_t\}$$

给定策略函数参数 θ，可以计算某条轨迹发生的概率 $p_\theta(\tau)$ 为

$$
\begin{aligned}
p_\theta(\tau) &= p(s_1)\, p_\theta(a_1|s_1)\, p(s_2|s_1,a_1)\, p_\theta(a_2|s_2)\, p(s_3|s_2,a_2)\cdots \\
&= p(s_1)\prod_{t=1}^{T} p_\theta(a_t|s_t)\, p(s_{t+1}|s_t,a_t)
\end{aligned}
$$

其中，$p(s_1)$ 是初始状态 s_1 发生的概率，$p_\theta(a_t|s_t)$ 为给定状态 s_t 策略函数采取动作 a_t 的概率，$p(s_{t+1}|s_t,a_t)$ 为给定当前状态 s_t 和动作 a_t，环境转移到状态 s_{t+1} 的概率。

奖励函数输入s_t, a_t，得到r_t，对于给定轨迹τ，累计奖励为$R(\tau) = \sum_{t=1}^{T} r_t$，代表某一个轨迹$\tau$的奖励，累计奖励也称为回报(Return)

在一场游戏的一个回合里，我们可以得到$R(\tau)$，我们需要做的是调整参数θ，使$R(\tau)$的值尽可能大。但是回报并非是一个标量值，因为演员采取哪一个动作（$p(s_{t+1}|s_t, a_t)$）以及环境转移到哪一个状态（$p(s_{t+1}|s_t, a_t)$）均以概率形式发生，因此轨迹τ 和对应回报$R(\tau)$均为随机变量，只能计算回报的期望：

$$\bar{R}_\theta = \sum_\tau R(\tau) p_\theta(\tau) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[R(\tau)]$$

其中$\bar{R}_\theta$表示使用参数为θ的策略与环境进行交互时的期望回报，轨迹τ服从$p_\theta(\tau)$的概率分布

给定一条轨迹，回报总是固定的，因此只能调整策略函数参数θ 使得高回报的轨迹发生概率尽可能大，而低回报的轨迹发生概率尽可能小。为了优化参数θ，可以使用梯度上升方法，优化θ使得期望回报尽可能大：

$$\nabla \bar{R}_\theta = \sum_\tau R(\tau) \nabla p_\theta(\tau)$$

梯度上升的思想是通过迭代更新参数，使得函数值不断增大，直至达到局部或全局最大值。在机器学习中，梯度上升常被用于最大化似然函数或最大化某些特定的目标函数。梯度上升计算公式如下：

$$\mu = \mu + \alpha_\mu \nabla_\mu J(\mu)$$

其中μ为要更新的参数向量，α为学习率(步长)，$\nabla_\mu J(\mu)$为目标函数$J(\mu)$对参数μ的梯度

观察下式中，只有$\nabla p_\theta(\tau)$ 与θ 有关，

$$\nabla \bar{R}_\theta = \sum_\tau R(\tau)\nabla p_\theta(\tau)$$

考虑到$p_\theta(\tau)$ 是多个概率值连乘的情况，难以进行梯度优化，根据梯度计算的性质

$$\nabla \log f(x) = \frac{1}{f(x)}\nabla f(x) \rightarrow \nabla f(x) = f(x)\nabla \log f(x)$$

将$\nabla p_\theta(\tau)$ 转为$\nabla \log p_\theta(\tau)$的形式，即$\nabla p_\theta(\tau) = p_\theta(\tau)\nabla \log p_\theta(\tau)$ ，可得

$$\begin{aligned}\nabla \bar{R}_\theta &= \sum_\tau R(\tau)\nabla p_\theta(\tau) \\ &= \sum_\tau R(\tau)p_\theta(\tau)\nabla \log p_\theta(\tau) \\ &= \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[R(\tau)\nabla \log p_\theta(\tau)\right]\end{aligned}$$

由前述推到得知$p_\theta(\tau)$为下式，将其代入上述公式可得到$\nabla \log p_\theta(\tau)$

$$p_\theta(\tau) = p(s_1)\, p_\theta(a_1|s_1)\, p(s_2|s_1,a_1)\, p_\theta(a_2|s_2)\, p(s_3|s_2,a_2) \cdots$$

$$= p(s_1) \prod_{t=1}^{T} p_\theta(a_t|s_t)\, p(s_{t+1}|s_t,a_t)$$

$$\log p_\theta(a_1|s_1) p_\theta(a_2|s_2) \ldots p_\theta(a_t|s_t) + \log p(s_2|s_1,a_1) \ldots p(s_{t+1}|s_t,a_t)$$

$$\nabla \log p_\theta(\tau) = \nabla \left(\log p(s_1) + \sum_{t=1}^{T} \log p_\theta(a_t|s_t) + \sum_{t=1}^{T} \log p(s_{t+1}|s_t,a_t) \right)$$

$$= \nabla \log p(s_1) + \nabla \sum_{t=1}^{T} \log p_\theta(a_t|s_t) + \nabla \sum_{t=1}^{T} \log p(s_{t+1}|s_t,a_t)$$

$$= \nabla \sum_{t=1}^{T} \log p_\theta(a_t|s_t)$$

$$= \sum_{t=1}^{T} \nabla \log p_\theta(a_t|s_t)$$

这里是对策略函数参数θ求梯度，$\log p(s_1)$和$\log p(s_{t+1}|s_t,a_t)$由环境决定与变量θ无关，为常量，求导值为0

将上式求得的$\nabla \log p_\theta(\tau)$代入前述所得的$\nabla \bar{R}_\theta$，可得

$$\begin{aligned}\nabla \bar{R}_\theta &= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[R(\tau) \nabla \log p_\theta(\tau) \right] \\ &= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[R(\tau) \sum_{t=1}^{T} \nabla \log p_\theta(a_t | s_t) \right]\end{aligned}$$

由于期望无法直接计算，因此在实践中，通常是从概率分布 $p_\theta(\tau)$ 中采样N条轨迹近似计算期望：

$$\nabla \bar{R}_\theta = \sum_\tau R(\tau) \nabla p_\theta(\tau)$$

$$= \sum_\tau R(\tau) p_\theta(\tau) \nabla \log p_\theta(\tau)$$

$$= \sum_{t=1}^{T} \nabla \log p_\theta(a_t|s_t)$$

$$\nabla \bar{R}_\theta = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[R(\tau) \sum_{t=1}^{T} \nabla \log p_\theta(a_t|s_t) \right]$$

$$\approx \frac{1}{N} \sum_{n=1}^{N} R(\tau^n) \sum_{t=1}^{T_n} \nabla \log p_\theta(a_t^n \mid s_t^n)$$

$$= \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla \log p_\theta(a_t^n \mid s_t^n)$$

随机采样N条轨迹，每条轨迹采样的概率为 $\frac{1}{N}$，可能存在多条轨迹相同

直观来看，式 (6.16) 中的 $R(\tau^n)$ 指示了 $p_\theta(a_t^n \mid s_t^n)$ 的调整方向和大小。当 $R(\tau^n)$ 为正，说明给定状态 s_t^n 下，动作 a_t^n 能够获得正回报，因此梯度上升会使概率 $p_\theta(a_t^n \mid s_t^n)$ 增大，即策略更有可能在状态 s_t^n 下采取动作 a_t^n；反之，则说明动作会受到惩罚，相应地，策略会减少在状态 s_t^n 下采取动作 a_t^n 的概率。得到$\nabla \bar{R}_\theta$后，通过梯度上升公式更新θ，即$R_{\theta'} \leftarrow R_{\theta'} + \eta \nabla \bar{R}_\theta$

在实践中往往会出现这样的情况，即回报总是正的，这样一来式(6.16) 中的$R(\tau^n)$项总是正的，因此会总是提升策略在状态s_t^n 下采取动作a_t^n 的概率。为了保证在状态s_t^n 下所有可能动作的概率和为1，在提升概率之后会做归一化。结果就是那些提升幅度比较小的动作概率最终会下降，如图6.6 所示，由于动作a、c 的概率提升更多，尽管动作b 的概率也会提升，但经过归一化后动作b 的概率会下降。

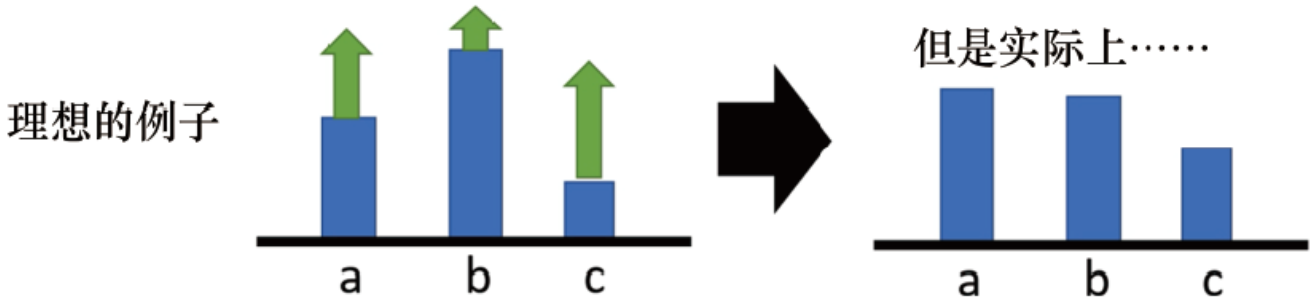


图 6.6　理想情况下动作概率的变化

由于动作b 获得的回报相对更小，所以获得更低的概率，似乎上述过程没有什么问题。但是这是针对理想情况而言的，由于在实际计算梯度的时候，总是采样有限的N 条轨迹来更新参数θ，所以某些状态–动作对可能不会被采样到。如图6.7 所示，动作a 没有被采样，而动作b、c 被采样因而概率提升，所以最后动作a 的概率就会下降。然而，没有采样到动作a 并不能说明动作a 是不好的。我们希望奖励不总是正的

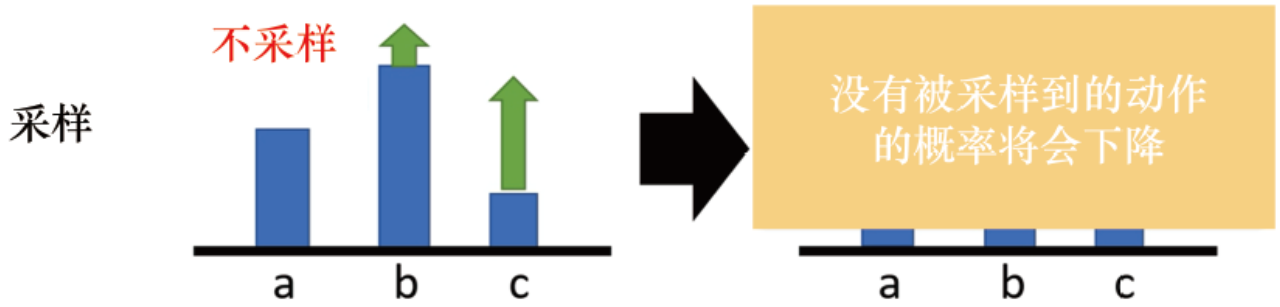


图 6.7　实际情况下动作概率的变化

解决这个问题的方法是在回报项$R(\tau^n)$上减去一个基线（Baseline）b，使得这一项的期望为0，这样在实际更新的时候概率值更新会有正有负，最终概率更新幅度之和大致为0，从而避免了因某些动作没有被采样而动作概率下降的问题。回报的梯度如下所示：

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \left(R\left(\tau^n\right) - b\right) \nabla \log p_\theta \left(a_t^n \mid s_t^n\right)$$

其中$b = \mathbb{E}_{\tau \sim p_\theta(\tau)} R(\tau)$，即回报的期望。这一项在实践中常用的计算方法是，在训练过程中记录历史$R(\tau^n)$的均值用以估计回报的期望。

下式中仍然存在另外一个值得考虑的问题，在一条轨迹中所有状态，动作对都使用相同的奖励项进行加权

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R(\tau^n)-b)\nabla \log p_\theta(a_t^n|s_t^n)$$

直觉上，一条轨迹中一般不会所有的动作都是好的，而是有些动作好，有些动作差。我们希望对不同的动作赋予不同的奖励权重，考虑到执行当前动作之前发生的事情与当前动作没有关系，因此，对于当前(状态，动作)对的奖励，只计算从这个动作执行以后得到的奖励

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\sum_{t'=t}^{T_n} r_{t'}^n - b\right)\nabla \log p_\theta(a_t^n|s_t^n)$$

当前动作会对时间较近的状态影响大，对时间较远的状态影响小。因此，在计算累积奖励的时候，对于未来较遥远的奖励应该予以折扣，即

$$\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n$$

其中$\gamma \in [0,1]$是折扣因子，一般设为0.9或0.99，随着时间间隔增大，奖励的折扣也越大。综合前面的添加基线的技巧，可以将回报的梯度表示为如下形式：

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \left(\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b \right) \nabla \log p_\theta(a_t^n | s_t^n)$$

6.3.2 优势函数

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\sum_{t'=t}^{T_n}\gamma^{t'-t}r_{t'}^n - b\right)\nabla \log p_\theta(a_t^n|s_t^n)$$

考察上式中，状态-动作对(s_t^n, a_t^n)，其中$\sum_{t'=t}^{T_n}\gamma^{t'-t}r_{t'}^n$为给定状态$s_t^n$下动作$a_t^n$的奖励，由此定义动作价值函数$Q(s,a)$和状态价值函数$V(s)$

动作价值函数：表示在状态s执行动作a所获得的长期回报，$Q(s,a)=\sum_{t'=t}^{T_n}\gamma^{t'-t}r_{t'}^n$

状态价值函数：在当前状态s下执行所有动作所能获得到长期回报的期望值，

定义优势函数$A^\theta(s,a)$

$$A^\theta(s,a) = Q(s,a) - V(s)$$

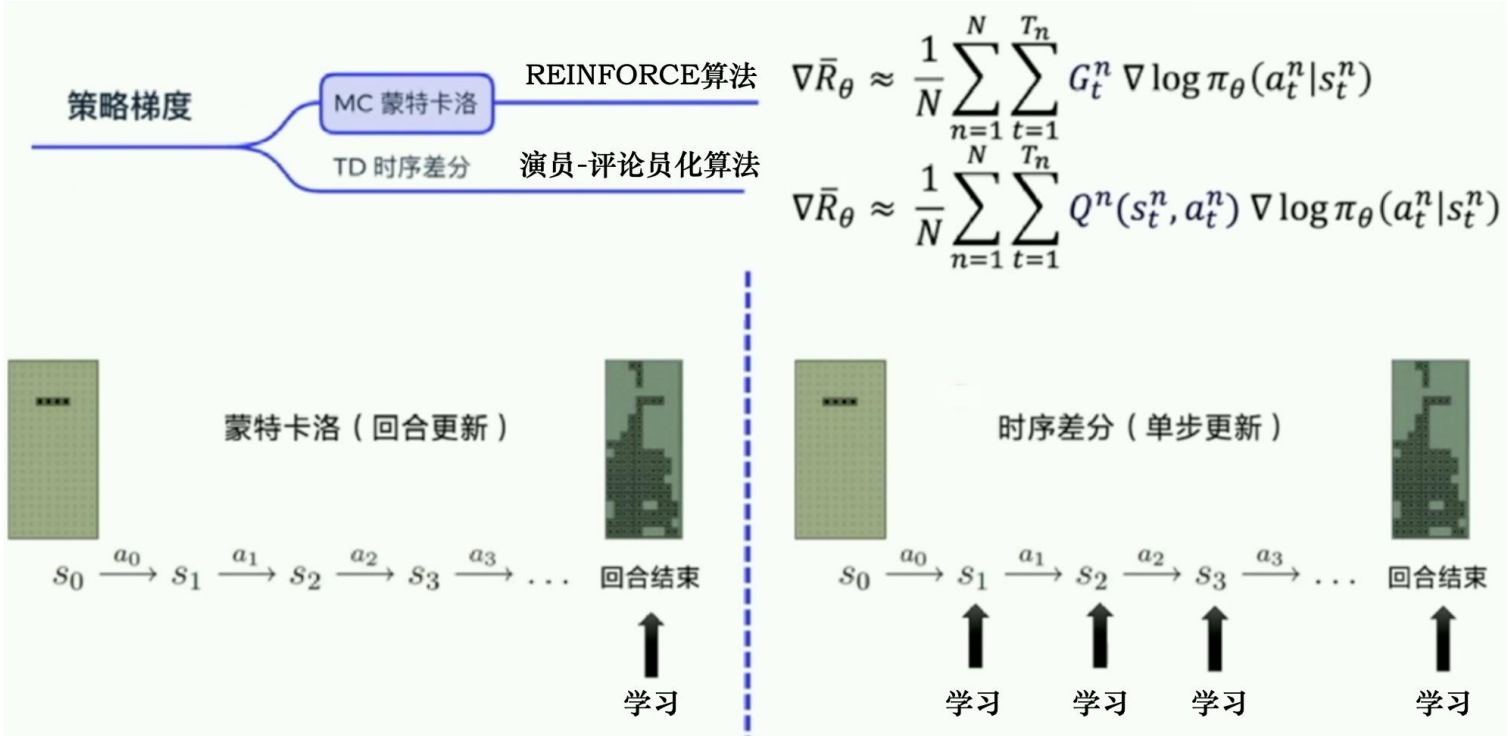
优势函数的意义在于当状态为s_t时执行特定动作a_t相较于其它动作好的程度，更容易区分动作的相对优势，函数值$A^\theta(s_t,a_t)$通常可以用网络估计，该网络就称为评论员

6.3.2 广义优势估计

一般而言，估计优势函数有两种方法，蒙特卡洛方法(Monte Carlo Methods，MC)和时序差分方法(Temporal Difference，TD)

蒙特卡洛方法可以理解为在完成一个回合之后，再利用这个回合的数据去学习。

时序差分方法则是每个步骤更新一次，每执行一个动作，更新一次，对于未来预期总奖励利用神经网络进行拟合



6.3.2 广义优势估计

在蒙特卡洛方法中，由于已经获得整个回合的所有数据，因而可以方便的计算出每个动作执行之后获得未来总奖励。一般用G_t表示第t步即之后获得到总奖励，则

$$G_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} = r_{t'} + \gamma G_{t+1}$$

所以在计算中，一般从后向前推算，直到G_1。由于需采样很多步，并将多次的结果的累计，虽然结果无偏，但会造成方差较大

在计算每个动作的梯度$\nabla \log p_\theta(a_t|s_t)$时，可以直接将神经网络输出的每个动作的概率值与实际动作的独热向量形式相乘，例如动作a, b, c概率为$[0.2,0.3,0.5]$，动作a的独热向量为$[0,1,0]$，则$[0,1,0]$与$\log[0.2,0.3,0.5]$相乘可得$\nabla \log p_\theta(a_t|s_t)$。

这种方法也称为REINFORCE算法

6.3.2 广义优势估计

时序差分方法只采样当前动作的奖励，其后的奖励采用状态价值函数进行估计，即

$$G_t = r_t + \gamma V(s_{t+1})$$

由于状态价值函数$V(s_{t+1})$是由神经网络拟合得来的，因此G_t会有较高的偏差。

类似的也可采样k步奖励，即

$$G_t = r_t + \gamma r_{t+1} + \cdots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k})$$

随着k的增大，时序差分方法也逐步接近蒙特卡洛方法。

从蒙特卡洛方法到有时序差分方法，方差变小，偏差变大，可以定义k步优势函数为

$$A_t^{\theta,k} = r_t + \gamma r_{t+1} + \cdots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k}) - V(s_t)$$

6.3.2 广义优势估计

蒙特卡罗方法高方差、无偏差，而时序差分低方差、高偏差。为了权衡方差与偏差，**广义优势估计（Generalized advantage Estimation，GAE）**方法将优势函数定义为k 步优势的指数平均：

$$A_t^{GAE(\gamma,\lambda)} = (1-\lambda)(A_t^1 + \lambda A_t^2 + \lambda^2 A_t^3 + \cdots)$$

广义优势估计同时利用蒙塔卡洛和时序差分的优势，具有低方差，低偏差的好处

指数平均(EMA)是一种常用的时间序列数据平滑方法，它给近期数据赋予更大的权重，使得平均值能更快反应数据的变化情况。

$$EMA_t = (1-\lambda)EMA_{t-1} + \lambda X_t$$

6.3.3 近端策略优化算法

虽然前面已经详细阐述了策略梯度、添加基线、精细奖励及优势函数等能够让策略梯度算法更加稳定的优化方法，但是策略梯度方法的效率问题，仍然需要进一步探讨。如前所述，策略梯度的基本形式如下所示：

$$\nabla \bar{R}_\theta = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[R(\tau) \nabla \log p_\theta(\tau) \right]$$

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

实际计算时，需要从环境中采样很多轨迹τ，然后按照上述策略梯度公式（或者添加各种可能优化）对策略函数参数 θ 进行更新。由于 τ 从概率分布 $p_\theta(\tau)$ 中采样得到，一旦策略函数参数 θ 更新，概率分布 $p_\theta(\tau)$ 就会发生变化，之前采样过的轨迹便不能再次利用，所以策略梯度方法需要不断地从与环境的交互中学习而不能利用历史数据。因此这种方法的训练效率低下。

6.3.3 近端策略优化算法

策略梯度方法中，负责与环境交互的演员与负责学习的演员相同，这种训练方法被称为**同策略（On-Policy）**训练方法。相反，**异策略（Off-Policy）**训练方法则将这两个演员分离，固定一个演员与环境交互而不更新它，将交互得到的轨迹交由另外一个负责学习的演员训练。异策略的优势是可以重复利用历史数据，从而提升训练效率。**近端策略优化[162]就是策略梯度的异策略版本**。

由于异策略的实现依赖于重要性采样（Importance Sampling），因此本节将先介绍重要性采样的基本概念，在此基础上介绍近端策略优化算法及相关变种。

假设随机变量 x 服从概率分布p，如果需要计算函数$f(x)$ 的期望，那么可以从分布p中采样得到若干数据 x^i，然后使用如下公式进行近似计算：

$$E_{x\sim p}[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x)$$

如果N 足够大，那么上式的结果将无限趋近于真实的期望。

假设无法从分布 p 中采样，只能从分布 q 中采样x_i，x_i 是从另外一个分布中采样得到的，不能直接使用上式计算$E_{x\sim p}[f(x)]$，因为此时 x_i 服从分布q。需要对$E_{x\sim p}[f(x)]$加以变换：

$$E_{x\sim p}[f(x)] = \int f(x)p(x)\mathrm{d}x = \int f(x)\frac{p(x)}{q(x)}q(x)\mathrm{d}x = \mathbb{E}_{x\sim q}[f(x)\frac{p(x)}{q(x)}],$$

$$E_{x\sim p}[f(x)] = \mathbb{E}_{x\sim q}[f(x)\frac{p(x)}{q(x)}]$$

从 q 中每采样一个 x^i 并计算 $f(x^i)\frac{p(x)}{q(x)}$，再取期望值。$\frac{p(x)}{q(x)}$ 用来修正这两个分布的差异，我们将其称为重要性权重，因此这种方法被称为**重要性采样**。其中 q 可以是任何一个分布，这样就可以解决无法直接从p中采样的问题。

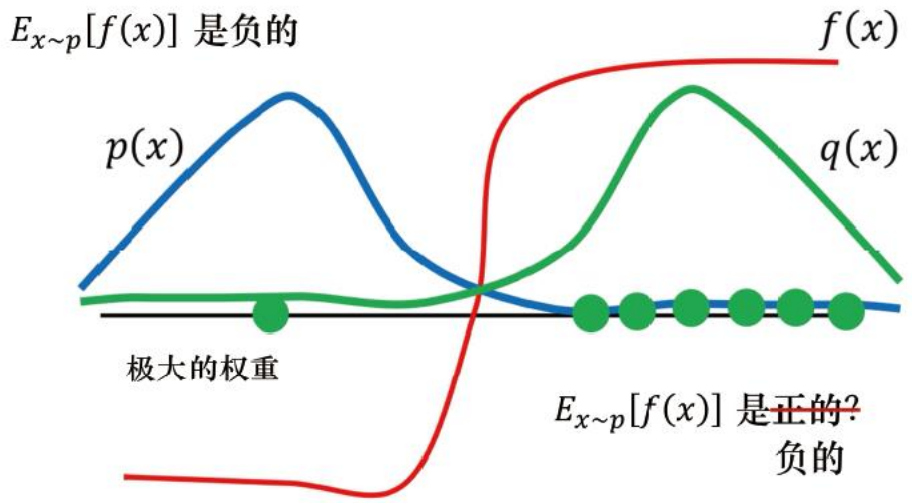


图 6.8　重要性采样中分布 q 和 p 差距过大可能引起的问题

然而，在实践中受制于采样次数有限，分布q 不能和p差距太大，否则结果可能会差别很大。如图6.8 所示，q 右侧概率大而左侧概率小，p 则反之，从 q 中采样会经常得到较多右侧数据点，而较少有左侧的数据点。由于重要性采样的，右侧会赋予较低的权重，左侧赋予极高的权重，因此计算得到的 $f(x)$ 期望仍然是负的。但是，由于 q 左侧概率很低，如果采样次数不足，没有采样到左侧的数据点，那么得到的期望就是正的，与预期差别非常大。因此，在实践中会约束这两个分布，使之尽可能减小差异。

将重要性采样运用到策略函数更新，把同策略换成异策略。假设负责学习的智能体策略为π_θ，负责采样的智能体策略为$\pi_{\theta'}$。按照$\nabla \bar{R}_\theta = \mathbb{E}_{\tau \sim p_\theta(\tau)}[R(\tau)\nabla \log p_\theta(\tau)]$计算，由于异策略，不能从$p_\theta(\tau)$ 中采样τ，只能从$p_{\theta'}$ 中采样，因此需要添加重要性权重修正结果：

$$E_{x\sim p}[f(x)] = \mathbb{E}_{x\sim q}[f(x)\frac{p(x)}{q(x)}] \longrightarrow \nabla \bar{R}_\theta = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\frac{p_\theta(\tau)}{p_{\theta'}(\tau)} R(\tau)\nabla \log p_\theta(\tau)\right]$$

注意，此策略梯度只更新π_θ，并不更新$\pi_{\theta'}$，这样才能够不断地从 $p_{\theta'}$ 中采样轨迹，使得 π_θ 可以多次更新。

将前述的策略梯度优化技巧和优势函数融入下式

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\sum_{t'=t}^{T_n}\gamma^{t'-t}r_{t'}^n - b\right)\nabla \log p_\theta(a_t^n|s_t^n)$$

得到

$$\nabla \bar{R}_\theta = \mathbb{E}_{(s_t,a_t)\pi_\theta}\left[A^\theta(s_t,a_t)\nabla \log p_\theta(a_t|s_t)\right]$$

将重要性采样代入其中

$$\nabla \bar{R}_\theta = \mathbb{E}_{(s_t,a_t)\pi_{\theta'}}\left[\frac{p_\theta(s_t,a_t)}{p_{\theta'}(s_t,a_t)}A^{\theta'}(s_t,a_t)\nabla \log p_\theta(a_t|s_t)\right]$$

s_t, a_t是θ'与环境交互时采样得到的数据，因此策略变为$\pi_{\theta'}$，优势函数变为$A^{\theta'}(s_t,a_t)$，但是训练时要更新的参数是θ。将$p_\theta(s_t,a_t)$和$p_{\theta'}(s_t,a_t)$进行拆解

$$p_\theta(s_t,a_t) = p_\theta(s_t|a_t)p_\theta(s_t)$$
$$p_{\theta'}(s_t,a_t) = p_{\theta'}(s_t|a_t)p_{\theta'}(s_t)$$

将拆解后的式子代入，可得

$$\nabla \bar{R}_\theta = \mathbb{E}_{(s_t,a_t)\pi_{\theta'}}\left[\frac{p_\theta(s_t|a_t)}{p_{\theta'}(s_t|a_t)}\frac{p_\theta(s_t)}{p_{\theta'}(s_t)}A^{\theta'}(s_t,a_t)\nabla \log p_\theta(a_t|s_t)\right]$$

这里假设模型为θ时看到状态s_t的概率与模型为θ'时看到状态s_t的概率一样，即

$$p_\theta(s_t) = p_{\theta'}(s_t)$$

此处假设相等是因为大多数情况下状态与采取的动作无关，更重要的原因是$p_\theta(s_t)$难以计算，此处将该问题忽略。

因此上式化作

$$\nabla \bar{R}_\theta = \mathbb{E}_{(s_t,a_t)\pi_{\theta'}}\left[\frac{p_\theta(s_t|a_t)}{p_{\theta'}(s_t|a_t)}A^{\theta'}(s_t,a_t)\nabla \log p_\theta(a_t|s_t)\right]$$

由梯度形式反推可得优化目标函数值

$$J^{\theta'}(\theta) = \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(s_t|a_t)}{p_{\theta'}(s_t|a_t)}A^{\theta'}(s_t,a_t)\right]$$

重要性采样的一个重要的稳定性保证是分布p 和分布q 不能相差太多，即$p_\theta(s_t|a_t)$和$p_{\theta'}(s_t|a_t)$要尽可能接近，PPO方法通过添加限制θ与θ'输出动作的KL散度约束来解决这一问题

$$J_{\mathrm{PPO}}^{\theta'}(\theta) = J^{\theta'}(\theta) - \beta \mathrm{KL}(\theta, \theta')$$

$$J^{\theta'}(\theta) = \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t \mid s_t)}{p_{\theta'}(a_t \mid s_t)} A^{\theta'}(s_t, a_t)\right]$$

需要注意的是，这里并不是要保证θ 和θ' 的参数的空间距离保持相似，否则可以直接使用 $L2$ 范数来约束。这里是要保证$p_\theta(a_t|s_t)$和$p_{\theta'}(a_t|s_t)$的表现相似，即要保证的是动作概率的相似。两者的差别在于，即使参数相似，其输出的动作也可能大相径庭。

虽然PPO 算法已经相对高效，但是其计算过程依然非常复杂，每一步更新的运算量非常大。为了进一步提升PPO 算法的计算效率，文献[162] 中又提出了两个变种：**近端策略优化惩罚（PPO-Penalty）**和**近端策略优化裁剪（PPO-Clip）**。

PPO-Penalty 算法先初始化一个策略参数θ^0，多次迭代更新策略，记第k 次迭代之后的策略为θ^k。在每一次迭代中使用前一轮迭代的结果θ^k 与环境交互，得到一系列数据，并用于本轮的策略参数更新。并加入了自适应KL散度的策略，即对KL散度添加权重β，对KL散度设置上下限，当超过上限时增大β，当小于下限时减小β，整体描述如下

$$J_{\mathrm{PPO}}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta \mathrm{KL}\left(\theta, \theta^k\right)$$

$$J^{\theta^k}(\theta) \approx \sum_{(s_t, a_t)} \frac{p_\theta\left(a_t \mid s_t\right)}{p_{\theta^k}\left(a_t \mid s_t\right)} A^{\theta^k}\left(s_t, a_t\right)$$

PPO-Clip 算法则直接裁剪重要性权重，这样就可以不需要计算KL 散度：

$$J_{\mathrm{PPO2}}^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} \min\left(\frac{p_\theta\left(a_t|s_t\right)}{p_{\theta^k}\left(a_t|s_t\right)}A^{\theta^k}\left(s_t,a_t\right),\ \mathrm{clip}\left(\frac{p_\theta\left(a_t|s_t\right)}{p_{\theta^k}\left(a_t|s_t\right)},1-\varepsilon,1+\varepsilon\right)A^{\theta^k}\left(s_t,a_t\right)\right)$$

其中ε 是超参数，例如可以设置为0.1 或者0.2。Clip函数的意思是裁剪重要性权重的大小，如果超过了$1+\varepsilon$，那么Clip 函数输出$1+\varepsilon$；如果在$[1-\varepsilon,1+\varepsilon]$，则输出本来的权重；如果小于$1-\varepsilon$，则输出$1-\varepsilon$，如图6.9 所示。

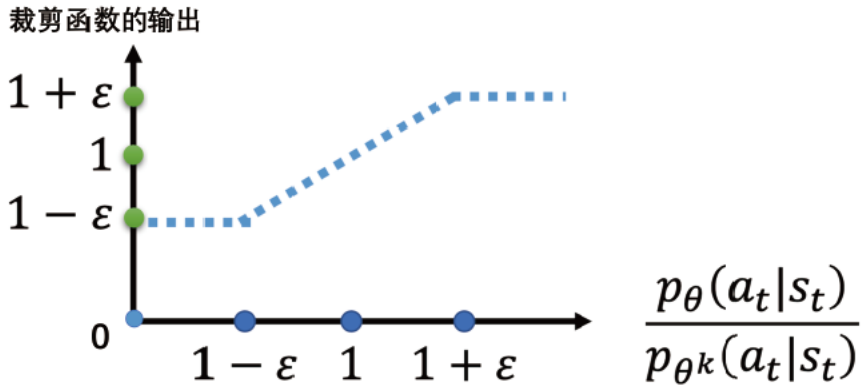


图 6.9　clip 函数示意图

6.3.3 近端策略优化算法--PPO 算法变种

近端策略优化裁剪目的在于使$p_\theta(a_t|s_t)$与$p_{\theta^k}(a_t|s_t)$尽可能接近，对于优势函数A^{θ^k}

若$A>0$，则说明当前动作是”好”的，则要增大该(状态，动作)对的概率值，即让$p_\theta(s_t|a_t)$变大，但最大值不超过$1+\varepsilon$
若$A<0$，则说明当前动作不好，则应减小该(状态，动作)对的值，既让$p_\theta(s_t|a_t)$变小，但不低于$1-\varepsilon$

这样的好处就是，我们不会让 $p_\theta(s_t|a_t)$与 $p_{\theta^k}(s_t|a_t)$差距太大

目录
Contents

6.4 MOSS-RLHF实践

如前所述，人类反馈强化学习机制主要包括策略模型、奖励模型、评论模型、参考模型等部分。需要考虑奖励模型设计、环境交互及代理训练的挑战，同时叠加大语言模型高昂的试错成本。对于研究人员来说，使用人类反馈强化学习面临非常大的挑战。RLHF 的稳定训练需要大量的经验和技巧。

本书作者所在的复旦大学自然语言处理实验室团队针对PPO 算法的内部工作原理进行了深入分析，并发布了PPO-Max 算法[164] 以确保模型训练的稳定性，发布了具有良好模型通用能力的中英文奖励模型，减轻了重新标记人类偏好数据的成本，还发布了MOSS-RLHF 开源训练框架。

本节将介绍使用MOSS-RLHF 框架进行人类反馈强化学习的实践。

目录
Contents

在人类偏好数据的标注过程中，标注者需要对模型针对同一问题的多个回复进行评分。由于系统多次回复的内容相似，导致标注困难，标注者之间的评分一致性仅为60% ~ 70%。因此，**为确保标注质量，标注过程中需要对问题的多样性和不同标注者的标注标准进行严格控制**。在奖励模型的训练中，标注数据可能会受到噪声的影响，需要进行适当的去噪处理。如果发现奖励模型的性能不均衡，则应及时增加新的标注数据来补充和修正。此外，奖励模型的底座大小和性能直接关系到评分的泛化能力。为了达到更好的泛化效果，建议在资源允许的前提下，选择较大的底座模型来训练奖励模型。

在PPO 的训练中，确保强化学习的稳定性和逐渐的收敛性是非常困难的。开源项目MOSSRLHF[164] 深入研究了影响PPO 稳定性的各种因素。经过实验验证总结出了七种关键因素，包括KL-惩罚项、奖励值的正则化与裁剪，以及评论模型的损失裁剪等。基于这些研究，提出了PPOMax算法，确保RLHF 的稳定运行。此外，该项目还研究了如何在PPO 训练中有效监控性能的提升，推荐使用PPL、模型输出长度和回复奖励等综合标准，以实现模型的平稳训练。

不过，PPO 训练时常出现**“Reward Hacking”现象**。这导致模型在短时间内迅速提高回复奖励，但其输出可能毫无意义或重复某些内容，这种情况反映了模型陷入了局部最优。为了避免这一问题，增强当前模型输出与SFT 模型输出空间的KL 惩罚力度是一个有效方法，它可以确保回复奖励的缓慢而稳定的提升。

评估PPO 训练后的成果是一项挑战。因为“Reward Hacking”现象的存在，不能仅仅依赖回复奖励来判定训练效果。需要在保证模型优化处于正常范围的前提下，逐渐接近奖励提升的极限，同时需对模型输出进行人工评估。由于人工评估的成本很高，GPT-4 的评估可作为一种替代。但在使用GPT-4 评估时，需精心设计提示语，这样GPT-4 才能（针对如有用性和无害性等指标）公正地评价各个模型的回复效果。考虑到GPT-4 对提示语和位置等因素的敏感性，我们还需要考虑这些因素，确保公正评价。

本章小结

通过有监督微调，大语言模型已经初步具备了遵循人类指令，并完成各类型任务的能力。然而有监督微调需要大量指令和所对应的标准回复，获取大量高质量的回复需要耗费大量的人力和时间成本。由于有监督微调通常采用交叉熵损失做为损失函数，目标是调整参数使得模型输出与标准答案完全相同，不能从整体上对模型输出质量进行判断。强化学习方法更适合生成式任务，也是大语言模型构建中必不可少的关键步骤。

本章介绍了基于类人反馈的强化学习基础概念、奖励模型以及近端策略优化方法，并在此基础上介绍面向大语言模型强化学习的MOSS-RLHF框架实践。