

## 拆解大语言模型RLHF中的PPO



TRiddle  
字节跳动 算法工程师

关注

1087 人赞同了该文章

收起

为什么大多数介绍大语言模型RLHF+的文章，一讲到PPO算法+的细节就戛然而止了呢？要么直接略过，要么就只扔出一个PPO的链接。然而LLM x PPO跟传统的PPO还是有些不同的呀。

其实在ChatGPT+推出后的相当一段时间内，我一直在等一篇能给我讲得明明白白的文章，但是一直未能如愿。我想大概是能写的人都没时间写吧。

前几个月，自己在工作中遇到要用到PPO的场景了。我心想，干脆自己啃算了。

于是我找到了InstructGPT引用的OpenAI自家的大语言模型RLHF论文《fine-tuning language models from human preferences》和《learning to summarize from human feedback》的源码，逐行阅读。然后用近似但不完全相同的风格复现了一遍。后来又和同事一起把自己的实现和微软的DeepSpeed-Chat+的实现相互印证，才算是理解了。

既然已经有了一些经验，为何不将它分享出来呢？就当是抛砖引玉吧。万一写的不对，也欢迎大家一起交流讨论。

由于本文以大语言模型RLHF的PPO算法为主，所以希望你在阅读前先弄明白大语言模型RLHF的前两步，即SFT Model和Reward Model的训练过程。另外因为本文不是纯讲强化学习的文章，所以我在叙述的时候不会假设你已经非常了解强化学习了。只是如果完全没有了解过，你可能会觉得有些操作看上去没有那么显然。但只要你非常了解语言模型和深度学习，应该不会影响你把整个流程给捋顺。

接下来，我会把大语言模型RLHF中的PPO分成三部分逐一介绍。这三部分分别是采样、反馈和学习。

在开始之前，我先用一段伪代码+把三部分的关系简要说明一下（先建立一个印象，看不懂也没关系，后面自然会看懂）：

```
policy_model = load_model()

for k in range(20000):
    # 采样（生成答案）
    prompts = sample_prompt()
    data = respond(policy_model, prompts)

    # 反馈（计算奖励）
```

赞同 1087

230 条评论

分享

喜欢



```
rewards = reward_func(reward_model, data)

# 学习（更新参数）
for epoch in range(4):
    policy_model = train(policy_model, prompts, data, rewards)
```

对于其中的每部分我都会用<sup>+</sup>计算图<sup>+</sup>来辅助描述，然后还会根据我的描述更新这段伪代码。

好了，让我们开始这趟旅程吧~



大语言模型的RLHF，实际上是模型先试错再学习的过程。

我们扮演着老师的角色，给出有趣的问题，而模型则会像小学生一样，不断尝试给出答案。模型会对着黑板写下它的答案，有时候是正确的，有时候会有错误。我们会仔细检查每一个答案，如果它表现得好，就会给予它高声赞扬；如果它表现不佳，我们则会给予它耐心的指导和反馈，帮助它不断改进，直到达到令人满意的水平。

采样

采样就是学生回答问题的过程，是模型根据提示（prompt）输出回答（response）的过程，或者说是模型自行生产训练数据的过程。

例如：

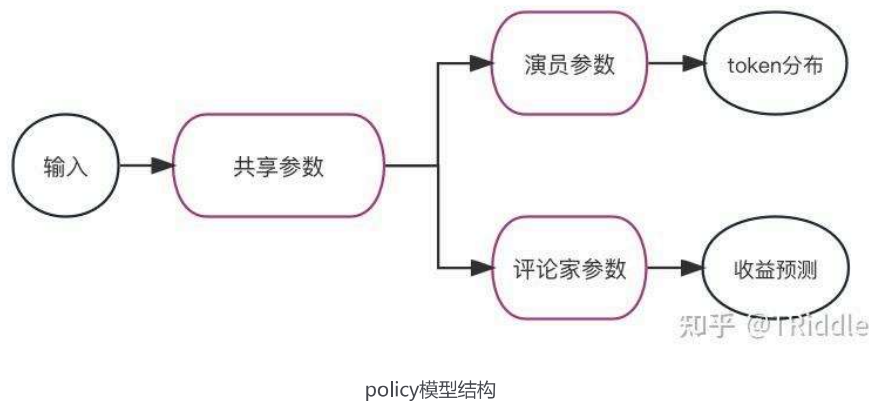
prompt	response
请告诉我三种常见的动物。	猫，狗，鹦鹉。
如何评价电影《爱乐之城 <sup>+</sup> 》？	音乐的经典令人赞叹不已，结局却让人感到五味杂陈。
詹姆斯和库里谁更伟大？	他们都很伟大，我无法比较。

PPO在这一部分做了什么呢？

先明确一个概念——策略（policy），它就是RLHF中的“学生”。policy由两个模型组成，一个叫做<sup>+</sup>演员模型<sup>+</sup>（Actor），另一个叫做评论家模型（Critic）。它们就像是学生大脑中的两种意识，一个负责决策，一个负责总结得失。

其中演员就是我们想要训练出来的大模型。在用PPO训练它之前，它就是RLHF的第一步训练出来的SFT（Supervised Fine-Tuning）model。输入一段上下文，它将输出下一个token的<sup>+</sup>概率分布<sup>+</sup>  $p(\cdot|\text{context})$ 。评论家是强化学习的辅助模型，输入一段上下文，它将输出下一个token的“收益”。

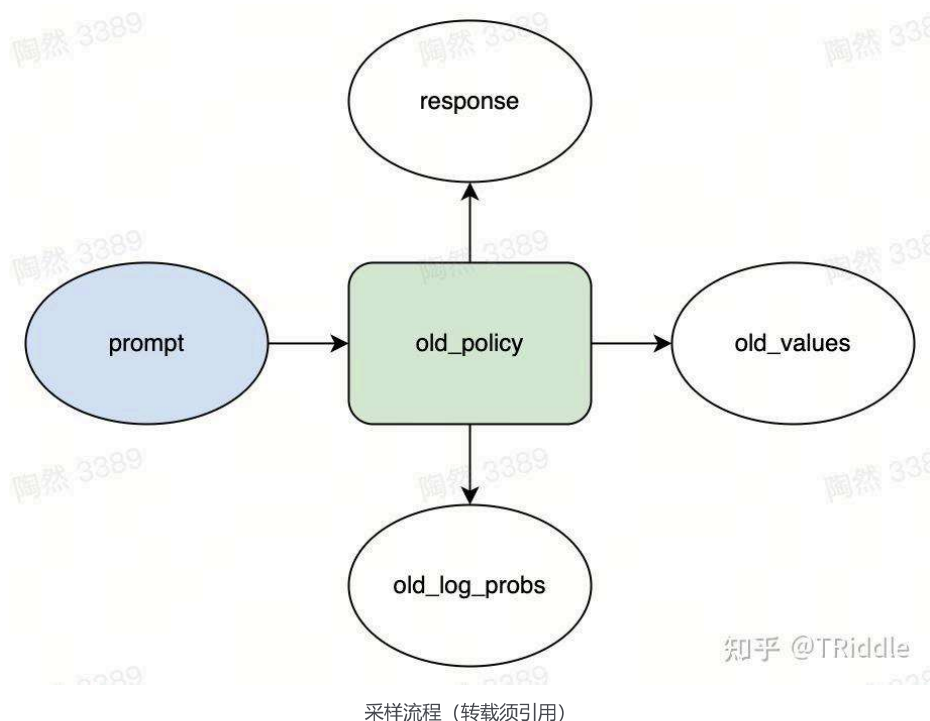
什么是“收益”呢？简单来说就是从下一个token开始，模型能够获得的总奖励（浮点数标量）。这里说的奖励包括Reward Model给出的奖励。奖励是怎么给的，<sup>+</sup>以及收益有什么用<sup>+</sup>，这些内容我们后面会详细介绍。



从实现上说，评论家就是将演员模型的倒数第二层连接到一个新的全连接层上。除了这个全连接层+之外，演员和评论家的参数都是共享的（如上图）。

**（更新：上面提到的模型结构是较早期的版本，后续不共享参数的实现方式也有很多）**

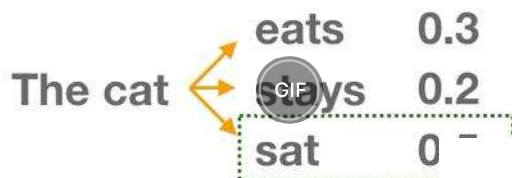
现在来看看PPO的采样过程中有哪些模型和变量。如下图，矩形表示模型，椭圆表示变量。



图中的“old\_policy”矩形就是刚刚说的policy（为啥有个“old”前缀呢？后面我会详细解释）。

**采样指的是old\_policy从prompt池中抽出M个prompt后，对每个prompt进行语言模型的token采样：**

- 计算response的第1个token的概率分布，然后从概率分布中采样出第1个token
- 根据第1个token，计算response的第2个token的概率分布，然后从概率分布中采样出第2个token
- .....
- 根据前N-1个token，计算response的第N个token的概率分布，然后从概率分布中采样出第N个token



然后就得到了三个输出。假设对每个prompt，policy生成的token的个数为N，那么这三个输出分别是

- response: M个字符串，每个字符串包含N个token
- old\_log\_probs: 演员输出的  $M \times N$  的张量<sup>+</sup>，包含了response中token的对数概率  $\log(p(\text{token}|\text{context}))$
- old\_values: 评论家输出的  $M \times N$  的张量，包含了每次生成token时评论家预估的收益

得到这三个输出后，采样阶段就就结束了。这三个输出都是后续阶段重要的输入数据。

我们先将采样部分的伪代码更新一下：

```
# 采样
prompts = sample_prompt()
responses, old_log_probs, old_values = respond(policy_model, prompts)
```

就像是一场考试，学生已经完成了答题环节，他们在黑板上留下了答案。但这只是整个学习过程的一个环节，接下来是关键的反馈步骤。

反馈

反馈就是老师检查答案的过程，是奖励模型（Reward Model）给response打分的过程，或者说 是奖励模型给训练数据X标上Y值的过程。

打出的分数衡量了response的正确性，它也可以被视为prompt和response的匹配程度。

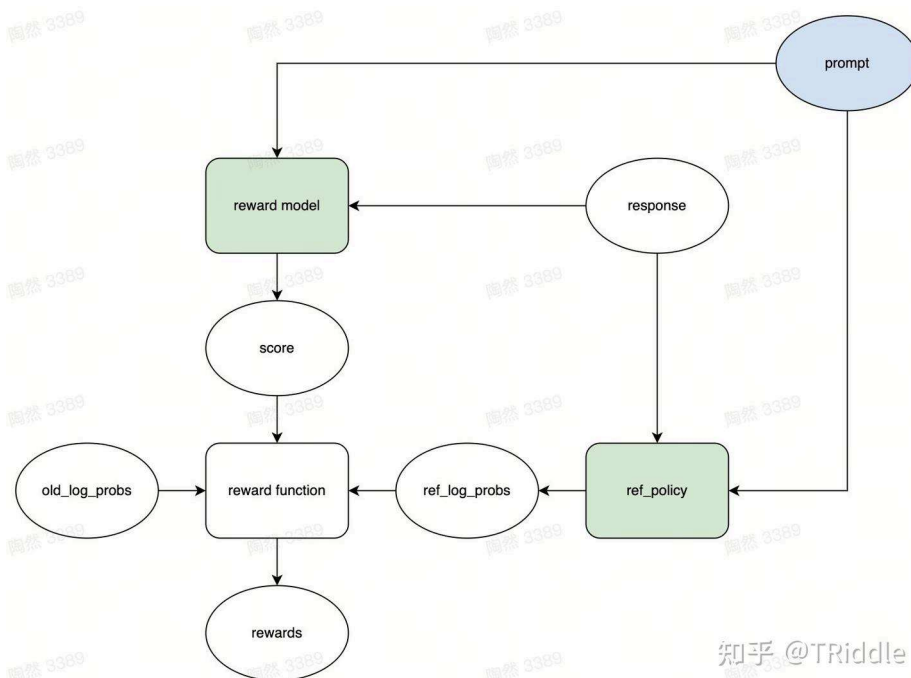
例如：

prompt	response	score
请告诉我三种常见的动物。	猫，狗，鹦鹉。	1
请告诉我三种常见的动物。	海豹，猫头鹰。	0.5
请告诉我三种常见的动物。	咖啡和芝士蛋糕。	0

Reward Model可以被比作班级里成绩最好的学生，他能够辅助老师批改作业。就像老师先教会这个学生如何批改作业，之后这个学生就能独立完成作业批改一样，Reward Model通过学习和训练，也能够独立地完成任务并给出正确的答案。

网上有很多资料介绍Reward Model<sup>+</sup>的训练过程，这也不是本文的重点，我就不再赘述了。

PPO拿训练好的Reward Mode做了什么呢？我们接着看图说话：

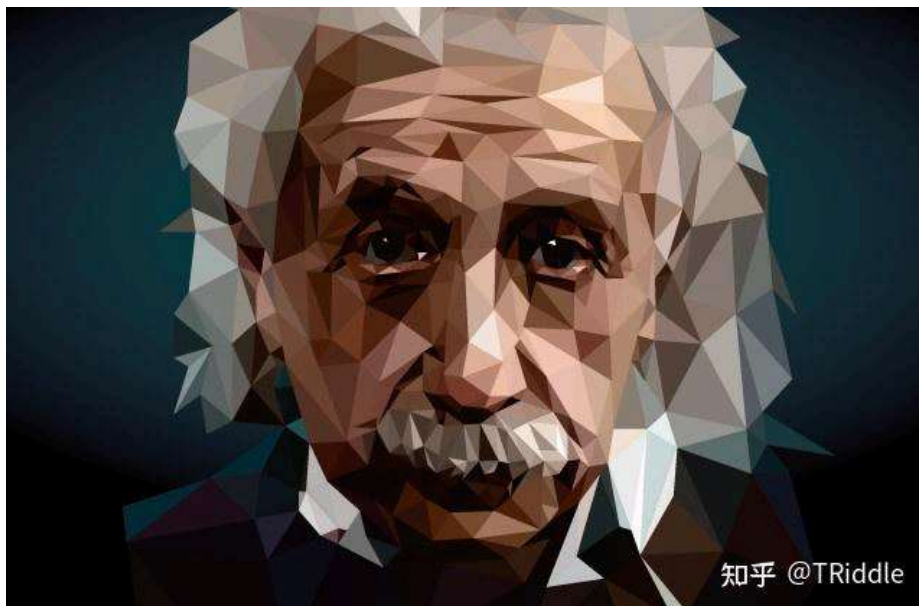


奖励流程（转载须引用）

从图中我们可以看出，左上角的绿色矩形reward model拿到prompt和response，然后输出了分数score。实际上发生的事情是，prompt和response被拼接成一个字符串，接着被送入到reward model中，最后reward model计算出了匹配分数。

你也许发现了，在图中，score并不是最终的奖励。它和最终的奖励rewards之间还隔着一个reward function函数。

**这是因为score只能衡量结果的对错，不能衡量过程的合理性。怎么衡量过程的合理性呢？一种简单粗暴的方法是：循规蹈矩，即为合理。**



当年爱因斯坦的相对论理论首次发表时，遭遇了许多质疑。后来，该理论被证明并得到了应有的认可。大家的目光可能都聚焦于爱因斯坦是如何坚定不移地坚持自己的理念并获得成功的。

然而，你有没有想过，那些反对和质疑其实也是必要的。

在相对论理论出现之前，已经有一个相对完整的物理系统。当时，一个年轻人突然出现挑战这个系统。在不知道他的路数的情况下，有必要基于现有的经验给予适当的质疑。因为并非每个人都是伟人啊。如果他的理论真的得到验证，那么就是给予肯定和荣誉的时候了。

**语言模型也是一样，在我们给予最终奖励之前，最好也对它的“叛逆”（即对现有模型的质疑）。**



怎么做呢？我们给它立一个规矩，只要它按照这个规矩来，就能获得少量奖励。而这个规矩就是我们在SFT阶段已经训练好的语言模型`ref_policy`（图中右下角的绿色矩形），或者说是完全还没经过强化学习训练的语言模型。

过程合理性奖励的计算方式是这样的。`ref_policy`拿到prompt，然后给`old_policy`生成的response的每个token计算对数概率，得到一个二维张量`ref_log_prob`。现在我们来查看`old_policy`的演员模型为第*i*个response生成的第*j*个token，生成这个token应该获得的奖励为

$$\text{ref\_log\_prob}[i, j] - \text{old\_log\_prob}[i, j] \quad (1)$$

来理解一下这个式子：

- `ref_log_prob[i, j]`越高，`ref_policy`越认可`old_policy`的输出，说明`old_policy`更守规矩，因此该获得更高的奖励
- `old_log_prob[i, j]`越高，`old_policy`获得的奖励反而越低。`old_log_prob[i, j]`作为正则项，保证了概率分布的多样性

有了这两个直觉上的解释，我们说式(1)是比较合理的。顺便说一句，熟悉信息论的人也许注意到了，式(1)是KL散度的简化版本。实际上式(1)完全可以改成计算两个token的概率分布的KL散度。这是另一个话题，就不延伸了。

**最终，我们将过程合理性奖励和结果正确性奖励合并起来，就得到了最终奖励的计算方式。**

注意，我们只在最后一个token上应用结果正确性奖励（`reward_model`的输出）。也就是说，第*i*个response的第*j*个token的奖励的计算方式为：

$$\text{reward}[i, j] = \begin{cases} \text{ref\_log\_prob}[i, j] - \text{old\_log\_prob}[i, j], & j < N \\ \text{ref\_log\_prob}[i, N] - \text{old\_log\_prob}[i, N] + \text{score}, & j = N \end{cases} \quad (2)$$

式(2)就是图中“reward function”的计算内容。

**通俗来说，整个reward function的计算逻辑是典型的霸总逻辑：除非你能拿到好的结果，否则你就得给我守规矩。**



注意，我们只对response计算奖励。另外在整个反馈阶段，`reward_model`和`ref_policy`是不更新参数的。

一旦给出reward，就完成了反馈阶段。现在我们将反馈部分的伪代码更新一下：

```
# 采样
prompts = sample_prompt()
responses, old_log_probs, old_values = respond(policy_model, prompts)

# policy_model的副本，不更新参数
ref_policy_model = policy_model.copy()

# 反馈
scores = reward_model(prompts, responses)
```

```
ref_log_probs = analyze_responses(ref_policy_model, prompts, responses)
rewards = reward_func(scores, old_log_probs, ref_log_probs)
```

这就像是老师在检查学生的答案并给出评价后，学生们就可以了解他们的表现如何，并从中学习和进步。然而，获得反馈并不是结束，而是新的开始。正如学生需要用这些反馈来进行复习和改进一样，模型也需要通过学习阶段来优化其性能和预测能力。

## 学习

**“学习”就是学生根据反馈总结得失并自我改进的过程，或者说是强化优势动作的过程。**

如果说前两步分别是在收集数据X，以及给数据打上标签Y。那么这一步就是在利用数据(X, Y)训练模型。

"强化优势动作"是PPO学习阶段的焦点。在深入探讨之前，我们首先要明确一个关键概念——优势。

**此处，我们将优势定义为“实际获得的收益超出预期的程度”。**



为了解释这个概念，请允许我举一个例子。假设一个高中生小明，他在高一时数学考试的平均分为100分，在此之后，大家对他的数学成绩的预期就是100分了。到了高二，他的数学平均分提升到了130分。在这个学期，小明的数学成绩显然是超出大家的预期的。

表现是可用分数量化的，故表现超出预期的程度也是可以用分数差来量化的。我们可以认为，在高二阶段，小明超出预期的程度为30分（130 - 100）。根据优势的定義我们可以说，在高二阶段，小明相对于预期获得了30分的优势。

**在这个例子中，实际已经给出了PPO计算优势的方法：优势 = 实际收益 - 预期收益。**

对于语言模型而言，为第*i*个response生成第*j*个token的实际收益就是：从生成第*j*个token开始到生成第*N*个token为止，所能获得的所有奖励的总和。我们用return来表示实际收益，它的计算方式如下：

$$\text{return}[i, j] = \sum_{k=j}^N \text{reward}[i, k] = \text{reward}[i, j] + \dots + \text{reward}[i, N] \quad (3)$$

(写给熟悉RL的人：简单起见，在这里我们**既不考虑贴现也不计算广义优势估计GAE**)

预期收益又该如何计算呢？记得我们在“采样”阶段提到过，policy包含演员模型和评论家模型，其中后者是用来预估收益的。其实，当时说的收益old\_values就是现在我们想要计算的预期收益。评论家会为response中的每个token计算一个预期收益，第i个response的第j个token的预期收益记为 $values[i, j]$ （它预估的是刚才提到的  $return[i, j] = \sum_{k=j}^N reward[i, k]$ ）。

现在，我们可以这样计算为第i个response生成第j个token的优势a（这里我们使用采样阶段计算出来的old\_values）：

$$a[i, j] = return[i, j] - old\_values[i, j] \quad (4)$$

好的，我们已经理解了优势的含义了。现在终于可以揭开这个关键主题的面纱——在PPO学习阶段，究竟什么是“强化优势动作”。

**所谓“强化优势动作”，即强化那些展现出显著优势的动作。**

在上面的小明的例子中，这意味着在高三阶段，小明应该持续使用高二的学习方法，因为在高二阶段，他的学习策略展示出了显著的优势。



在语言模型中，根据上下文生成一个token就是所谓的“动作”。“强化优势动作”表示：如果在上下文（context）中生成了某个token，并且这个动作的优势很高，那么我们应该增加生成该token的概率，即增加  $p(token|context)$  的值。

由于policy中的演员模型建模了  $p(token|context)$ ，所以我们可以给演员模型设计一个**损失函数<sup>+</sup>**，通过优化损失函数来实现“强化优势动作”：

$$actor\_loss = -\frac{1}{M} \frac{1}{N} \sum_{i=1}^M \sum_{j=1}^N a[i, j] \times p(token[i, j] | context) \quad (5)$$

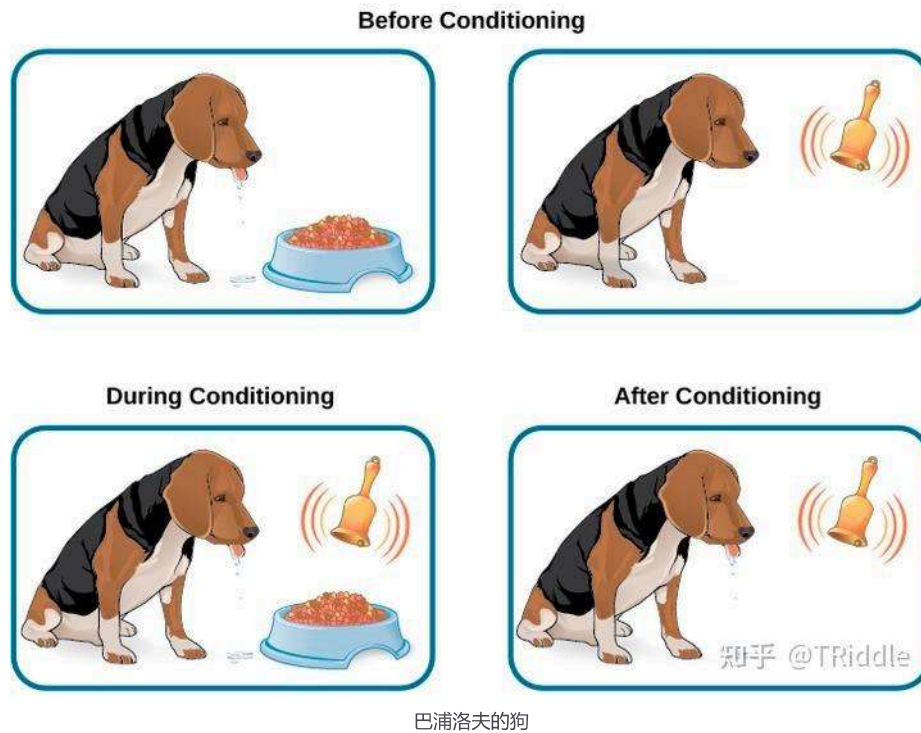
别被这个双重和式给吓住。它只是在用两个**矩阵计算<sup>+</sup>**元素乘法，得到一个新矩阵，然后计算新矩阵中所有元素的均值。最后加上一个负号，仅此而已。

我们分析一下它是怎么“强化优势动作”的：

- 当优势大于0时，概率越大，loss越小；因此优化器会通过增大概率（即强化优势动作）来减小loss
- 当优势小于0时，概率越小，loss越小；因此优化器会通过减小loss



这很像巴浦洛夫的狗不是吗？



另外还有两个点值得注意：

- 优势的绝对值越大，loss的绝对值也就越大
- 优势是不接收梯度回传的

实际上，式5只是一个雏形。PPO真正使用的演员的损失函数是这样的：

$$\text{actor\_loss} = -\frac{1}{M} \frac{1}{N} \sum_{i=1}^M \sum_{j=1}^N a[i, j] \times \frac{p(\text{token}[i, j]|\text{context})}{p_{old}(\text{token}[i, j]|\text{context})} \quad (6)$$

（写给熟悉RL的人：简单起见，在这里我们既不考虑损失的截断，也不考虑优势的白化）

式子6相比式5子多了一个分母  $p_{old}$ 。在式子6里， $p_{old}$  表示p的一个较老的版本。因为它不接收梯度回传，所以我们可以将  $p_{old}(\text{token}[i, j]|\text{context})$  当作常量，或者说，把它当成p的学习率的一部分。我们来分析一下它的作用。以优势大于0的情况为例，对任意i，当  $p_{old}(\text{token}[i, j]|\text{context})$  有较大的值的时候，p的参数的学习率更小。

**直观来说，当生成某个token的概率已经很大的时候，即便这个动作的优势很大，也不要再使劲增大概率了。或者更通俗地说，就是步子不要迈得太大。**

现在的问题就是，我们应该使用p的哪个老版本。还记得我们在本文开头时给出的伪代码吗（后来在介绍“采样”和“反馈”阶段时又各更新了一次），我们对着代码来解释：

```
policy_model = load_model()
ref_policy_model = policy_model.copy()

for k in range(20000):
    # 采样（已更新）
    prompts = sample_prompt()
    responses, old_log_probs, old_values = respond(policy_model, prompts)

    # 反馈（已更新）
    scores = reward_model(prompts, responses)
    ref_log_probs = analyze_responses(ref_policy_model, prompts, responses)
    rewards = reward_func(reward_model, scores, old_log_probs, ref_log_probs)

    # 学习
```

```
for epoch in range(4):
    policy_model = train(policy_model, prompts, responses, old_log_probs, old_valu
```

简单来说，这段代码做的事情是：迭代2万次。在每次迭代中，通过采样和反馈得到一份数据，然后在学习阶段使用数据微调语言模型。每份数据我们都拿来训练4个epoch。

那  $p_{old}$  使用2万次迭代开始之前的演员模型的参数可以吗？不行，那个版本过于老了（实际上就是SFT，我们已经在奖励阶段中的ref\_policy中用过了）。不妨使用同一次迭代的还未进入学习阶段的演员模型吧。如果是这样的话，仔细一看， $p_{old}(\text{token}|\text{context})$  不就是采样阶段得到的  $\text{old\_log\_probs}$  吗？只是少了一个对数而已。

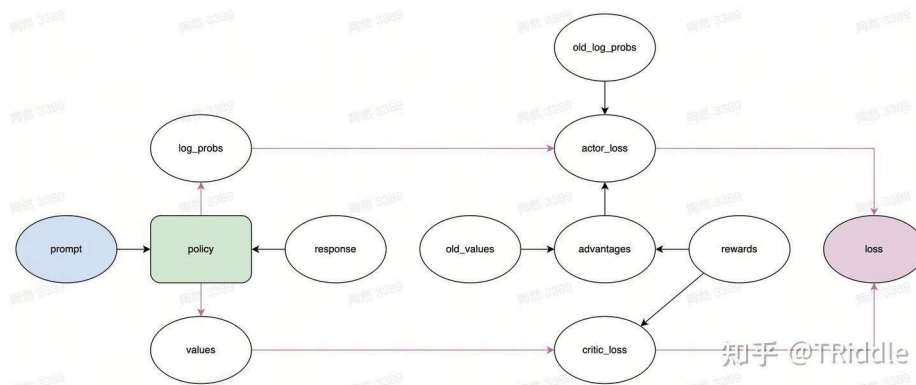
这就是为什么我们在采样阶段，对所有的模型和参数都使用“old”前缀，就是为了区分模型和变量的版本。

（补充：前面提到的old\_policy指的是上面伪代码中采样出old\_log\_probs的那个时刻的policy\_model）

而对于  $p(\text{token}|\text{context})$  我们可以使用实时的演员模型的参数计算出来，然后用log\_prob来表示它。于是，我们可以将式子6改写成以下形式：

$$\text{actor\_loss} = -\frac{1}{M} \frac{1}{N} \sum_{i=1}^M \sum_{j=1}^N a[i, j] \times \exp(\log\_prob[i, j] - \text{old\_log\_prob}[i, j]) \quad (7)$$

至此，我们完整地描述了PPO的学习阶段中“强化优势动作”的方法。就像下面的计算图展示的那样（policy与前面的图中的old\_policy不一样，是实时版本的模型）。



学习流程（转载须引用）

等等，似乎还没完。图中还有一个叫critic\_loss的没提到过的东西。

当然了，负责决策的演员需要学习，难道总结得失的评论家就不需要学习了？评论家也是需要与时俱进的嘛，否则画评家难道不怕再次错过梵高那样的天才？

前面我们提到过，评论家会为response中的每个token计算一个预期收益，第i个response的第j个token的预期收益记为  $\text{values}[i, j]$ ，它预估的是  $\text{returns}[i, j] = \sum_{k=j}^N \text{reward}[i, k]$ 。

既然如此，就设计一个损失函数来衡量评论家预期收益和真实收益之间的差距。

PPO用的是均方差损失<sup>★</sup>（MSE）：

$$\text{critic\_loss} = \frac{1}{2MN} \sum_{i=1}^M \sum_{j=1}^N (\text{values}[i, j] - \text{returns}[i, j])^2 \quad (8)$$

（写给熟悉RL的人：由于我们不考虑GAE，所以returns的计算也做了相应的简化）

最终优化policy时用的loss是演员和评论家的loss的加权和：

$$\text{loss} = \text{actor\_loss} + 0.1 \times \text{critic\_loss}$$

(9)

这才算是真正完事儿了。现在我们将整个PPO的伪代码都更新一下：

```
policy_model = load_model()
ref_policy_model = policy_model.copy()

for k in range(20000):
    # 采样
    prompts = sample_prompt()
    responses, old_log_probs, old_values = respond(policy_model, prompts)

    # 反馈
    scores = reward_model(prompts, responses)
    ref_log_probs, _ = analyze_responses(ref_policy_model, prompts, responses)
    rewards = reward_func(scores, old_log_probs, ref_log_probs)

    # 学习
    advantages = advantage_func(rewards, old_values)
    for epoch in range(4):
        log_probs, values = analyze_responses(policy_model, prompts, responses)
        actor_loss = actor_loss_func(advantages, old_log_probs, log_probs)
        critic_loss = critic_loss_func(rewards, values)
        loss = actor_loss + 0.1 * critic_loss
        train(loss, policy_model.parameters())
```

总结

到这里，大语言模型RLHF中PPO算法的完整细节就算介绍完了。掌握这些细节之后，我们可以做的有趣的事情就变多了。例如：

- 你可以照着伪代码从头到尾自己实现一遍，以加深理解。相信我，这是非常有趣且快乐的过程
- 你可以以此为契机，把强化学习知识系统性地学一遍。你会发现很多强化学习的概念一下变得具象化了
- 你可以在你的产品或者研究方向中思考PPO是否可以落地
- 你也许会发现PPO算法的不合理之处，那么就深入研究下去，直到做出自己的改进
- 你可以跟周围不熟悉PPO的小伙伴吹牛，顺便嘲讽对方（大误）

总之，希望我们都因为掌握了知识变得更加充实和快乐～

对了，因为我也不知道自己是否讲透彻了，所以如果愿意交流的话可以加一下微信。不胜感激！

（另外，我这里有一些大模型和强化学习的实习机会，欢迎有兴趣的同学加微信T\_Riddle咨询～）

<https://u.wechat.com/ME4jZrLsGFIRNZPdh3xP6fw> (二维码自动识别)

编辑于 2024-08-06 11:51 · IP 属地北京

内容所属专栏



**自然语言处理NLP论文阅读**  
自然语言处理，深度学习，NLP, DL



**人工智能研究**  
欢迎大家关注“人工智能与算法学习”微信公众号！

订阅专栏

订阅专栏


[ChatGPT](#) [大语言模型](#) [RLHF](#)



理性发言，友善互动

230 条评论

默认 最新



沃克

大致总结是，ppo状态是当前生成的序列，的动作是下一个token，奖励是与sft动作的kl距离，加上序列结尾的reward模型输出？

2023-08-27 · 广东

回复

20




TRiddle 作者

是这样的，课代表同学

2023-08-27 · 北京

回复

2



美洲大蠊很顽强

LLM中PPO算法部分大佬们只开源了基本算法，而没有开源具体怎么训练的，PPO训练过程是极其不稳定的，效果好与不好还取决于Reward Model，而且PPO还会有reward hacking现象，会导致LLM拿了不错的反馈，但却偏离了目标，建议参考MOSS开源RLHF，重点是对PPO过程中各个过程增加监控。

2023-08-23 · 中国香港

回复

9



算法全栈之路 · 美洲大蠊很顽强

是的，感觉你是实践过的

2024-06-04 · 北京

回复

3



美洲大蠊很顽强 · Justeagles

抱歉，我表述的有问题，trlx、deepspeed-chat等都有PPO的实现代码，我想表述的就是ppo的训练过程不稳定，这些开源的并没有相应的监控，在实际应用的过程中可以添加一些eval，看看生成的答案

2023-10-19 · 江苏

回复

3

展开其他 2 条回复



一念心宽

目前为止，这块看到的最好的文档

2024-08-16 · 江苏

回复

5




TRiddle 作者

谢谢

2024-08-18 · 北京

回复

1



我不是温尚翊

另外critic是必须的吗？如果直接用reward作为a（即优势）来训练actor，会怎么样？

2024-04-15 · 广东

回复

4




TRiddle 作者

没错，这样基本上就退化成policy gradient算法了

2024-04-17 · 北京

回复

4



我不是温尚翊 · TRiddle

谢谢！

2024-04-17 · 中国香港

回复

1

查看全部 13 条回复



蜡笔小熊猫

大力深入，温柔浅出

2024-04-11 · 广东

回复

1



TRiddle 作者

这个形容怎么这么神奇

2024-04-16 · 北京

回复

1



流浪法师

写得太好了

2024-02-27 · 山东

回复

2



TRiddle 作者

感谢鼓励

2024-03-02 · 河北



潇酒哲思

伪代码写得真好

2024-03-28 · 北京

回复 2



TRiddle

作者



谢谢鼓励

2024-04-01 · 北京

回复 1



贝吉塔ta

博主能帮忙解释下公式 (4) 优势函数的具体意义吗，我理解old\_values学的就是return，那理想情况下return[i]-old\_values[i]就是0，这里挣的是不是一个时间差，old\_values里对应的策略并不是最新的，从而得到当前策略动作的优势；还是说依据策略选动作的随机性，强化对应动作，更新策略

2023-10-11 · 上海

回复 2



TRiddle

作者



values学的是return的期望，而return的方差比较大，所以即便values学得特别好，values和return相等的情况其实是不容易出现的。这样讲可能有点抽象，举个例子吧。假如今天是周五，周末我可以等概率地做3件事：旅游（心情好，奖励为1）、发呆（心情一般，奖励为0）、跟人吵架（心情糟糕，奖励为-1），那么在周五我预估周末的return会是0。但实际return为0的概率并不高（1/3）。

2023-10-17 · 北京

回复 6



十点雨

return里有reward model的直接反馈，相当于gt

18 小时前 · 山东

回复 喜欢



开心花甲

“评论家就是将演员模型的倒数第二层连接到一个新的全连接层上”，一般critic的预训练基座模型不是比actor的要小么？

2023-08-29 · 广东

回复 2



TRiddle

作者



一般是两种实现的。一种是论文《Fine-Tuning Language Models from Human Preferences》里和TRL里实现的那种，是共享参数的。另一种是InstructGPT论文的附录里提到的和DeepSpeed-Chat里实现的那种，不共享参数。后面这种实现方式通常就像你说的那样，critic比actor要小得多

2023-08-29 · 北京

回复 1



李鹏宇

介绍方式非常友好。

2024-09-23 · 河北

回复 1



TRiddle

作者



谢谢反馈

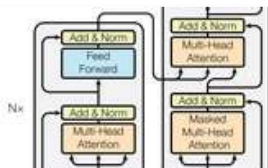
2024-10-08 · 北京

回复 1

点击查看全部评论



理性发言, 友善互动



### 预训练语言模型 (PLM) - 基础汇总篇 (一)

知识专场

### 基于世界模型的深度强化学习

起源&#34;世界模型&#34;被Iecun认为是弥补RL不足和通向下一代AI的要点。虽然Model-Based RL不是新概念，但是世界模型的构建方法，以及提升其泛化能力、注意力能力和记忆容量的设计还是颇具...

白楚

[1 分]

脆皮咕



