

# **Training language models to follow instructions with human feedback**

汇报人：许禹琪

- 回顾策略梯度系列算法
- PPO算法
- InstructGPT

# 一. 策略梯度算法

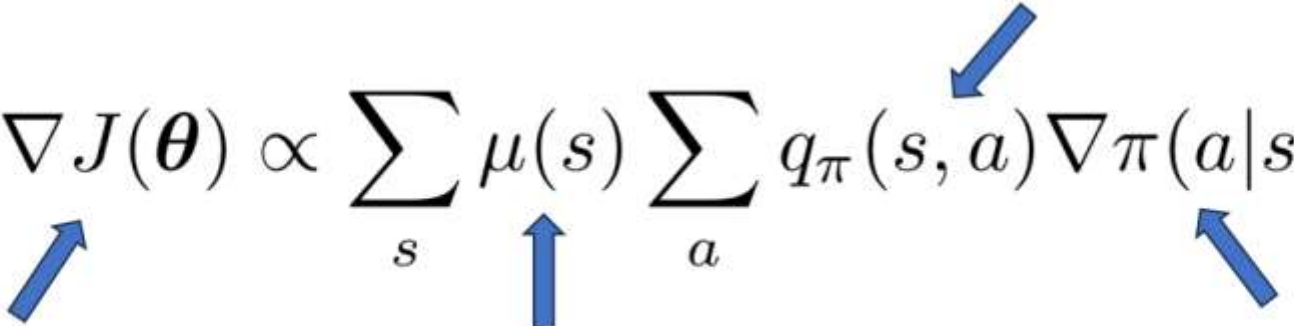
- 策略梯度定理
- REINFORCE
- REINFORCE with baseline
- Actor-Critic
- Advantage Function

策略梯度定理帮助我们得到更新决策函数参数的**梯度**

- $J(\theta) = v_{\pi_{\theta}}(S_0)$
- 即用初始状态的价值来评价策略的好坏

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_{\pi}(s, a) \nabla \pi(a|s, \theta)$$

S状态下选a行动的价值



策略目标函数梯度    以s为终点的总折现概率    s状态,  $\theta$ 参数选择器下选a概率

- $\mu_{\pi}(s) = \sum_{k=0}^{\infty} \gamma^k \Pr(S_0 \rightarrow s, k, \pi)$
- $\Pr(S_0 \rightarrow s, k, \pi)$ 表示按策略 $\pi$ 行动, 从状态 $S_0$ 出发经过k步到达状态s的概率

REINFORCE的更新公式——核心思想：用期望代替遍历

- 把S转化为期望

$$\begin{aligned}\nabla J(\boldsymbol{\theta}) &\propto \sum_s \mu(s) \sum_a q_\pi(s, a) \nabla \pi(a | s, \boldsymbol{\theta}) \\ &= \mathbb{E}_\pi \left[ \sum_a q_\pi(S_t, a) \nabla \pi(a | S_t, \boldsymbol{\theta}) \right]\end{aligned}$$

- 这里的 $\mathbb{E}_\pi$ 可以理解为依据  $\pi$  进行采样，在全部采样中s出现的折现概率
- 把A转化为期望

$$\begin{aligned}\nabla J(\boldsymbol{\theta}) &= \mathbb{E}_\pi \left[ \sum_a \pi(a | S_t, \boldsymbol{\theta}) q_\pi(S_t, a) \frac{\nabla \pi(a | S_t, \boldsymbol{\theta})}{\pi(a | S_t, \boldsymbol{\theta})} \right] \\ &= \mathbb{E}_\pi \left[ q_\pi(S_t, A_t) \frac{\nabla \pi(A_t | S_t, \boldsymbol{\theta})}{\pi(A_t | S_t, \boldsymbol{\theta})} \right] \quad (\text{replacing } a \text{ by the sample } A_t \sim \pi)\end{aligned}$$

- 这里的 $\mathbb{E}_\pi$ 多了一重含义，可以理解为依据  $\pi$  进行采样，在全部采样中出现(s,a)对的折现概率。
- 把  $q_\pi(S_t, A_t)$  写为等价形式

$$= \mathbb{E}_\pi \left[ G_t \frac{\nabla \pi(A_t | S_t, \boldsymbol{\theta})}{\pi(A_t | S_t, \boldsymbol{\theta})} \right], \quad (\text{because } \mathbb{E}_\pi[G_t | S_t, A_t] = q_\pi(S_t, A_t))$$

## REINFORCE算法——核心思想：用采样代替期望

初始化参数 $\theta$

repeat forever:

按策略 $\pi(a|s, \theta)$ 生成序列 $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

for  $t = 0, \dots, T-1$ :

$$G \leftarrow R_t + \gamma R_{t+1} + \dots + \gamma^{(T-t)} R_T$$

$$\theta \leftarrow \theta + \alpha \gamma^t G \nabla_{\theta} \log \pi(A_t | S_t, \theta)$$



$$\mathbb{E}_{\pi} \left[ G_t \frac{\nabla \pi(A_t | S_t, \theta)}{\pi(A_t | S_t, \theta)} \right]$$

REINFORCE with baseline的更新公式

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a \left( q_{\pi}(s, a) - b(s) \right) \nabla \pi(a|s, \boldsymbol{\theta}).$$

- 添加baseline目的是减少更新量的方差

Repeat forever:

生成轨迹  $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$ , 基于  $\pi(\cdot|\cdot, \boldsymbol{\theta})$

for  $t = 0, 1, \dots, T - 1$ :

$$G \leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k$$

$$\delta \leftarrow G - \hat{v}(S_t, \mathbf{w})$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t, \mathbf{w})$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$$

- $b(s)$ 在REINFORCE with baseline算法中实际上是一个状态价值函数网络

希望  $v(s_t, w)$  尽可能趋向于G

Repeat forever:

生成轨迹  $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$ , 基于  $\pi(\cdot|\cdot, \theta)$

for  $t = 0, 1, \dots, T - 1$ :

$$G \leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k$$

$$\delta \leftarrow G - \hat{v}(S_t, \mathbf{w})$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \delta \nabla \hat{v}(S_t, \mathbf{w})$$

$$\theta \leftarrow \theta + \alpha \gamma^t \delta \nabla \ln \pi(A_t | S_t, \theta)$$

## Updating the value network

- Recall  $v(s_t; \mathbf{w})$  is an approximation to  $V_\pi(s_t) = \mathbb{E}[G_t | s_t]$ .
- Prediction error:  $\delta_t = v(s_t; \mathbf{w}) - G_t$ .
- Gradient:  $\frac{\partial \delta_t^2 / 2}{\partial \mathbf{w}} = \delta_t \cdot \frac{\partial v(s_t; \mathbf{w})}{\partial \mathbf{w}}$ .
- Gradient descent:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \delta_t \cdot \frac{\partial v(s_t; \mathbf{w})}{\partial \mathbf{w}}.$$

作为对比：DQN网络的更新方式为

构造损失函数：最小化 Q 值的估计与 TD 目标之间的均方误差 (MSE)

$$L(\omega) = \frac{1}{2N} \sum_{i=1}^N \left[ Q_\omega(s_i, a_i) - \left( r_i + \gamma \max_{a'} Q_\omega(s'_i, a') \right) \right]^2$$



## Actor Critic的思想

- REINFORCE with baseline和Actor Critic的区别在于是否采用自举(bootstrapping)
  - sutton书中对**Bootstrapping**的解释:  
*Updating the value estimate for a state from the estimated values of subsequent states.*  
基于其他状态的价值函数估计来更新当前状态的价值函数估计
- 在REINFORCE算法中,  $\delta$  由蒙特卡罗方法得到, 而AC方法采用了时序差分方法

$$\begin{aligned} G &\leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k \\ \delta &\leftarrow G - \hat{v}(S_t, \mathbf{w}) \end{aligned} \quad \longrightarrow \quad \delta \leftarrow R + \gamma \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$$

## Actor Critic的更新公式

```
Loop forever (for each episode):  
  Initialize  $S$  (first state of episode)  
   $I \leftarrow 1$   
  Loop while  $S$  is not terminal (for each time step):  
     $A \sim \pi(\cdot|S, \theta)$   
    Take action  $A$ , observe  $S', R$   
     $\delta \leftarrow R + \gamma \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$  (if  $S'$  is terminal, then  $\hat{v}(S', \mathbf{w}) \doteq 0$ )  
     $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S, \mathbf{w})$   
     $\theta \leftarrow \theta + \alpha^{\theta} I \delta \nabla \ln \pi(A|S, \theta)$   
     $I \leftarrow \gamma I$   
     $S \leftarrow S'$ 
```

- 这里策略梯度也相应的使用了  $R + \gamma \hat{v}(S', \mathbf{w})$  作为采样值,  $\hat{v}(S, \mathbf{w})$  作为baseline。
- 自举带来的好处是减少方差, 加速学习。

## Advantage function

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

- 优势函数的意义：在状态s下选择动作a比按策略概率选取动作的**优越程度**。
- 在应用中， $Q^{\pi_{\theta}}(s, a)$ 将由蒙特卡罗，或时序差分方法得出，而  $V^{\pi_{\theta}}(s)$  则通常由一个参数网络构成
- REINFORCE with baseline其实已经应用了优势函数

## 二. Proximal Policy Optimization Algorithms

- PPO Insight
- Trust Region Policy Optimization(TRPO)
- PPO

- 传统的REINFORCE方法实际上是在线方法，数据不可复用

初始化参数 $\theta$

repeat forever:

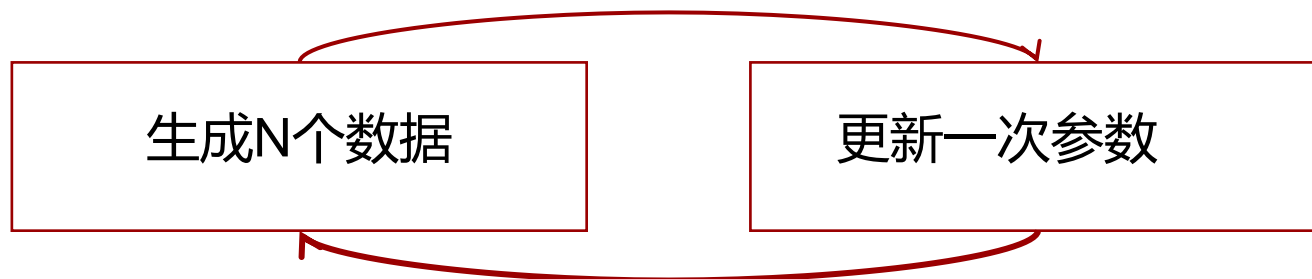
按策略 $\pi(a|s, \theta)$ 生成序列 $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

for  $t = 0, \dots, T-1$ :

$$G \leftarrow R_t + \gamma R_{t+1} + \dots + \gamma^{(T-t)} R_T$$

$$\theta \leftarrow \theta + \alpha \gamma^t G \nabla_{\theta} \log \pi(A_t | S_t, \theta)$$

- 真实的更新方式



- 下面设我们想要用来采样的决策函数为  $\pi'$  当前的策略为  $\pi$
- 回忆策略梯度公式，有

$$J_{\pi}(\theta) = \alpha \sum_s \mu_{\pi}(s) \sum_a q_{\pi}(s, a) \pi(a | s, \theta)$$

其中 $\alpha$ 是一个常数因子

- $\mu_{\pi}$  导致了需要基于  $\pi$  进行采样
- TRPO提出如下方法，优化下面这个函数同样可以达到相似的效果

$$L_{\pi}(\theta) = \alpha \sum_s \mu_{\pi'}(s) \sum_a q_{\pi}(s, a) \pi(a | s, \theta)$$

- 这是因为有如下良好性质

$$J_{\pi}(\theta) \geq L_{\pi}(\theta) - cD_{KL}^{max}(\pi, \pi')$$

- 其中c是一个常量
- TRPO考虑以下优化问题

$$\begin{aligned} \max_{\theta} L_{\pi}(\theta) &= \alpha \sum_s \mu_{\pi'}(s) \sum_a q_{\pi}(s, a) \pi(a | s, \theta) \\ \text{s.t.} \quad &D_{KL}^{max}(\pi, \pi') \leq \delta \end{aligned}$$

- 可以发现  $\mu_{\pi}$  的估计问题解决了，但是还需要  $\pi$  对a进行决策，
- TRPO提出采用重要性采样来解决这个问题

$$\begin{aligned} \nabla L_{\pi}(\theta) &= \alpha \sum_s \mu_{\pi'}(s) \sum_a q_{\pi}(s, a) \nabla \pi(a | s, \theta) \\ &= \alpha \sum_s \mu_{\pi'}(s) \sum_a \frac{\nabla \pi(a | s, \theta)}{\pi'(a | s, \theta)} q_{\pi}(s, a) \pi'(a | s, \theta) \end{aligned}$$

- 再次使用利用期望代替遍历的思想，同时用优势函数代替动作价值函数得到了如下的公式

$$\nabla L_{\pi}(\theta) = \mathbb{E}_{\pi'} \left[ \frac{\nabla \pi(a | s, \theta)}{\pi'(a | s, \theta)} A_{\pi}(s, a) \right]$$

- 再使用利用采样代替期望的思想，就可以得到算法。

Repeate forever:

生成轨迹  $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$  基于  $\pi'(\cdot | \cdot, \theta)$

for  $t = 0, 1, \dots, T - 1$  :

$$G \leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k$$

$$\delta \leftarrow G - \hat{v}(S_t, \mathbf{w})$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t, \mathbf{w})$$

$$\theta \leftarrow \theta + \alpha^{\theta} \gamma^t \delta \frac{\nabla \pi(A_t | S_t, \theta)}{\pi'(A_t | S_t, \theta)}$$



- 需要注意，在更新过程中还需要满足一些约束，不能随意的优化

<https://zhuanlan.zhihu.com/p/510136070>

- PPO不再用约束来控制更新策略和历史策略的变化差异，而是将其统一至损失函数。

$$L^{KL PEN}(\theta) = \hat{\mathbb{E}}_{\pi'} \left[ \frac{\pi(a | s, \theta)}{\pi'(a | s, \theta)} A_t - \beta \text{KL} [\pi'(a | s, \theta), \pi(a | s, \theta)] \right]$$

- 通过判断当前决策函数与原决策函数的偏离度，这个 $\beta$ 可以动态调节

- PPO的优化目标由三部分组成,

$$L(\theta) = L^{clip}(\theta) + c_1 L^{vf}(\theta) + c_2 S[\pi(a|s, \theta)]$$

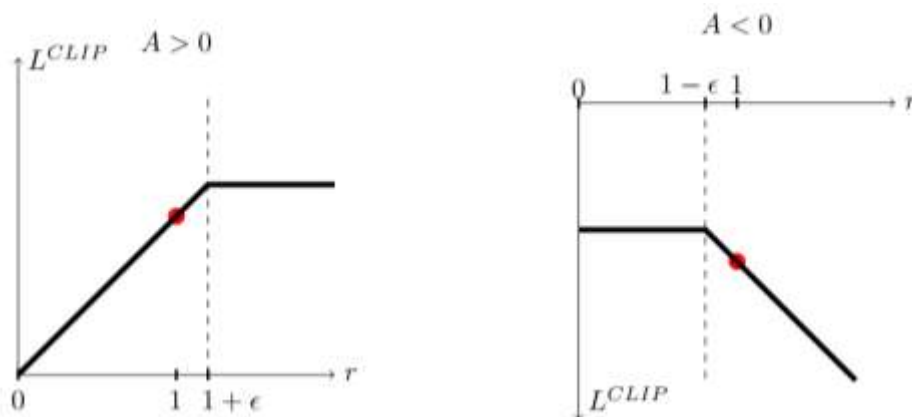
- 下面令  $r(\theta) = \frac{\pi(a | s, \theta)}{\pi_I(a | s, \theta)}$ ,

- 第一部分是

$$L^{CLIP}(\theta) = \mathbb{E}_{\pi'} [\min(r(\theta)A_{\pi}(s, a), \text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon) A_{\pi}(s, a))]$$

- 相比正常更新, CLIP希望保守的更新参数
- With this scheme, we only ignore the change in probability ratio when it would make the objective improve, and we include it when it makes the objective worse.*

不考虑那些使得目标向好得case, 只考虑那些让情况变糟的例子



- 这种设计类似于“保守主义”或“悲观主义”思想：优先避免损失，而不是追求高收益。
- TRPO通过复杂的约束（如KL散度约束）来限制策略更新，而PPO通过裁剪概率比实现了类似的效果，但计算更简单。两者的共同目标都是避免策略更新过大，从而保证训练的稳定性。

- 第二部分是

$$L^{vf}(\theta) = -\frac{1}{2}\mathbb{E} \left[ (V_{\theta}(s) - V_{\text{target}})^2 \right]$$

$$V_{\text{target}} = \sum_{k=1}^T \gamma^k R_k$$

- 这一部分会惩罚值函数的不准确性
- *If using a neural network architecture that shares parameters between the policy and value function, we must use a loss function that combines the policy surrogate and a value function error term.*  
通过增加这一部分损失，可以将价值函数的参数也统一为 $\theta$

- 第三部分是

$$S[\pi(a|s, \theta)] = -\mathbb{E}_{a \sim \pi} [\log \pi(a | s, \theta)]$$

- 可以理解为策略函数与均匀分布的交叉熵
- 策略函数越接近与均匀分布，值越大
- *This objective can further be augmented by adding an entropy bonus to ensure sufficient exploration*  
这一部分鼓励策略函数探索不同的状态

---

**Algorithm 1** PPO, Actor-Critic Style

---

```
for iteration=1, 2, ... do
  for actor=1, 2, ..., N do
    Run policy  $\pi_{\theta_{\text{old}}}$  in environment for  $T$  timesteps
    Compute advantage estimates  $\hat{A}_1, \dots, \hat{A}_T$ 
  end for
  Optimize surrogate  $L$  wrt  $\theta$ , with  $K$  epochs and minibatch size  $M \leq NT$ 
   $\theta_{\text{old}} \leftarrow \theta$ 
end for
```

---

on-policy  $\longrightarrow$  off-policy

相比于TRPO更易求解

## 三.InstructGPT



# 有监督微调（需要人工标注）

Step 1

**Collect demonstration data,  
and train a supervised policy.**

A prompt is  
sampled from our  
prompt dataset.



A labeler  
demonstrates the  
desired output  
behavior.



This data is used  
to fine-tune GPT-3  
with supervised  
learning.



先简单地微调一下GPT-3，需要人工标注：

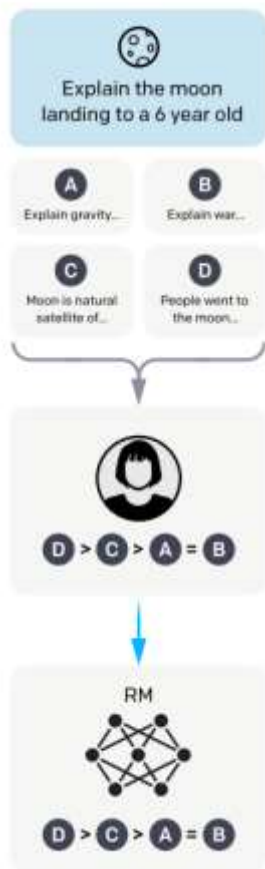
- 工程师团队设计了一个prompt dataset。
- 把这个prompt dataset发给人类标注员进行标注，其实就是回答问题。
- 用这个标注过的数据集微调GPT-3。

# 奖励模型训练（需要人工标注）

Step 2

Collect comparison data,  
and train a reward model.

A prompt and  
several model  
outputs are  
sampled.



A labeler ranks  
the outputs from  
best to worst.

This data is used  
to train our  
reward model.

训练一个奖励模型，目的是让RM模型学习人类导师对GPT-3输出的答案的排序的能力

- 拿这个微调过的GPT-3去预测prompt dataset里面的任务，获得一系列结果，图中是四个，实际应用是九个。
- 把这四个结果交给人类标注员进行标注，把预测的结果从好到坏进行标注。
- 用这些标注的结果训练一个奖励模型（reward model），这个奖励模型参数量为6M，把SFT模型最后的softmax层替换为线性层得到。

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x, y_w, y_l) \sim D} [\log(\sigma(r_\theta(x, y_w) - r_\theta(x, y_l)))]$$

Step 3

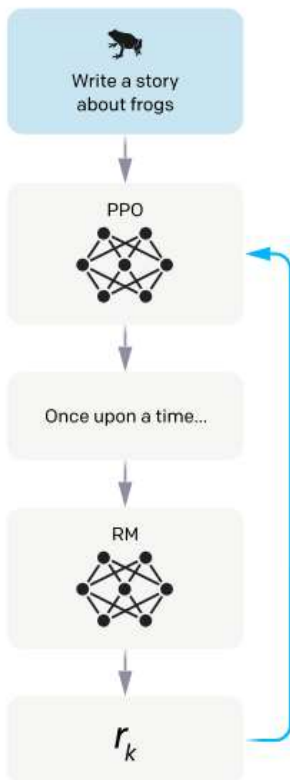
**Optimize a policy against the reward model using reinforcement learning.**

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



用PPO持续更新策略的参数，用第二步训练好的奖励模型给策略的预测结果打分，也就是reward。这个计算出来的分数会交给包着GPT-3内核的策略来更新梯度。

$$\text{objective}(\phi) = E_{(x,y) \sim D_{\pi_{\text{RL}}}} [r_{\theta}(x,y) - \beta \log(\pi_{\phi}^{\text{RL}}(y|x)/\pi^{\text{SFT}}(y|x))] + \gamma E_{x \sim D_{\text{pretrain}}} [\log(\pi_{\phi}^{\text{RL}}(x))]$$

- 实际上并不是照搬PPO，但是借鉴了思想
- 个人认为缺少了策略对数导数这一项，可能是OpenAI实验过程发现不需要这一项一样可以work，故省去

也可能是我理解没到位

# 谢谢

