A Quick Start Guide for RL in LLMs and Diffusion

- STRUCT Tutorial -

Haofeng Huang

2025/11/02

Outline

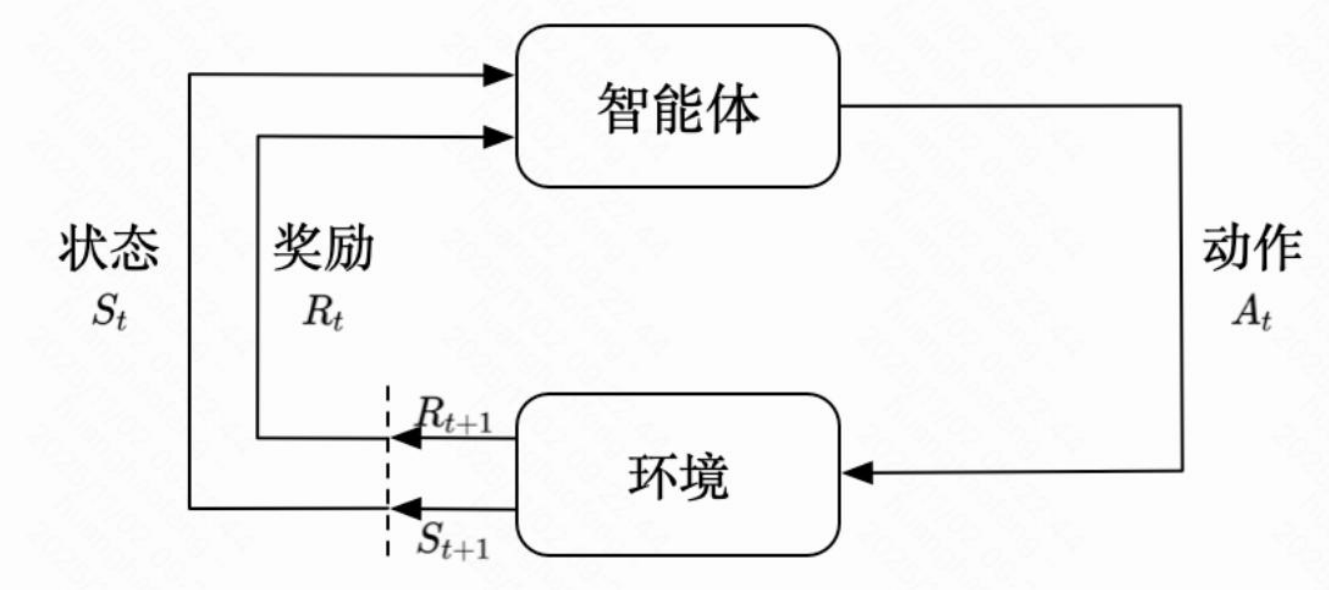
- 强化学习基础
- 策略梯度算法: REINFORCE, PPO, DPO
- RL for LLM: GRPO, GSPO, DAPO, Dr. GRPO
- RL for Diffusion: Flow-GRPO, ReFL, Mix-GRPO, Dance-GRPO

Outline

- 强化学习基础
- 策略梯度算法: REINFORCE, PPO
- RL for LLM: GRPO, GSPO
- RL for Diffusion: Flow-GRPO, ReFL

什么是强化学习

- 强化学习（Reinforcement Learning，RL）讨论的问题是**智能体（Agent）**怎么在复杂、不确定的**环境（Environment）**中确定自身的**动作（Action）**策略，从而最大化它能获得的**奖励（Reward）**



- https://datawhalechina.github.io/easy-rl/

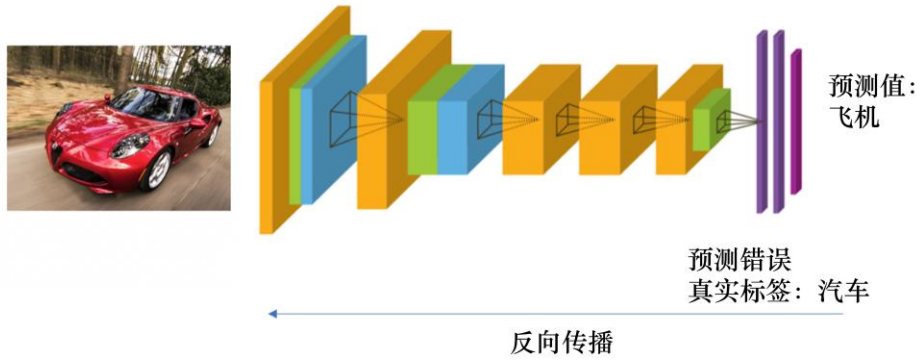
什么是强化学习

- 记住三个关键元素:
  - 环境: S
  - 奖励: R
  - 策略 / 动作 / : A （一般地，智能体的策略等价于动作的概率分布）

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

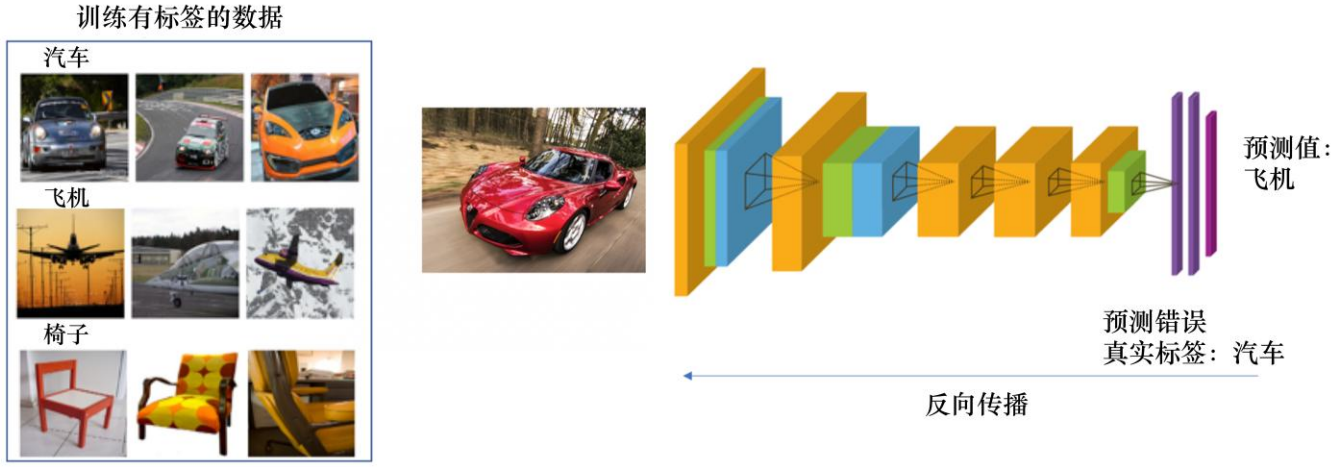
监督学习 vs 强化学习

- 监督学习需要大量被标注的数据，比如对于图像分类任务需要汽车、飞机、椅子这些被标注的图片，而这些图片要满足独立同分布，即它们之间是没有关联关系的。而预测的标签可以根据标注判断正确与否。



监督学习 vs 强化学习

- 类比于强化学习：
  - 输入类比于**环境**
  - “预测”类比于**动作**
  - 标签是否正确类比于**奖励**



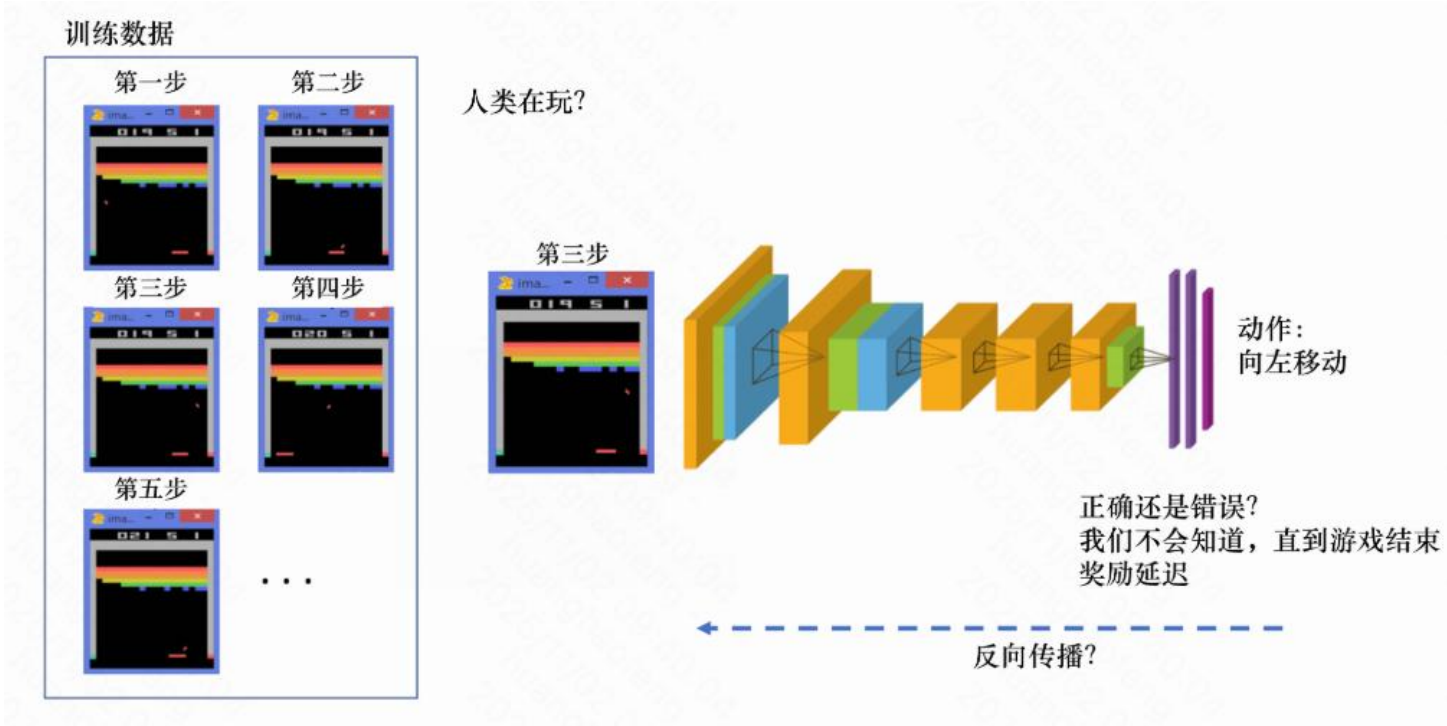
监督学习 vs 强化学习

- 在监督学习中：
  - 输入（环境）是独立同分布的➔ 智能体平等地对待每个输入
  - 输出的正确与否（奖励）是确定的➔智能体可以通过监督学习进行行为的训练训练

- 在强化学习中：
  - 输入（环境）不是独立同分布的，事实上环境是马尔科夫的➔智能体需要处理时间相关序列
  - 输出的正确与否（奖励）无法立刻确定➔需要构建“评估当前行动的价值”的算法

监督学习 vs 强化学习

- 例如打砖块游戏。对于这一游戏进行监督学习是困难的。



监督学习 vs 强化学习

- 因此，需要为智能体引入**价值函数（value function）**来衡量当前状态。即评估智能体进 入某个状态后，可以对后面的奖励带来多大的影响，值越大，说明智能体进入这个状态越有利。
- 其中，状态是指环境和动作的二元组 (s, a)。
- 考虑到收敛性，价值函数往往会引入折扣因子。价值函数的两种形式：

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

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

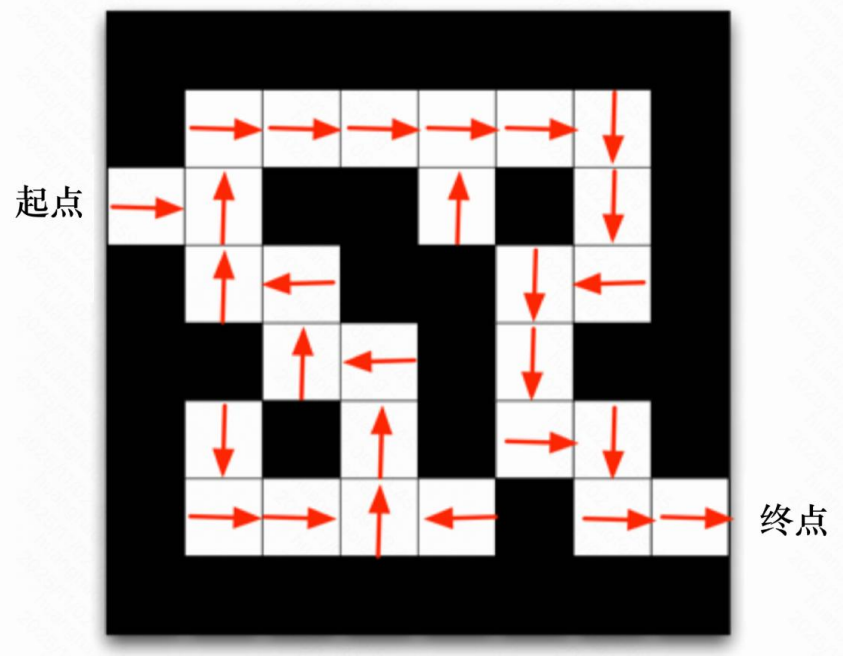
智能体的构成

- 至此，强化学习的关键构成介绍完毕，包括：
  - 环境 (S)
  - 奖励 (R)
  - 智能体 (Agent)：包含价值函数 (value function)，以及策略 (policy)

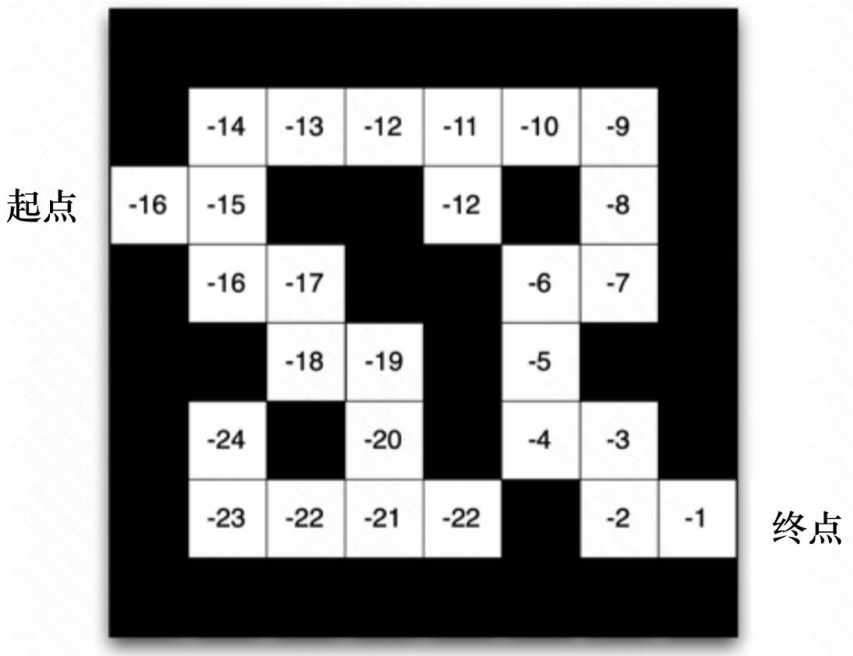
智能体分类

- 分为策略智能体 (policy-based agent)， 价值智能体 (value-based agent)和演员-评论员智能体（actor-critic agent）

基于策略:



基于价值:



智能体分类

- **基于价值的智能体（value-based agent）**显式地学习价值函数，隐式地学习它的策略。策略是其从学到的价值函数里面推算出来的。
- **基于策略的智能体（policy-based agent）**直接学习策略，我们给它一个状态，它就会输出对应动作的概率。基于策略的智能体并没有学习价值函数。
- 把基于价值的智能体和基于策略的智能体结合起来就有了**演员-评论员智能体（actor-critic agent）**。这一类智能体把策略和价值函数都学习了，然后通过两者的交互得到最佳的动作。

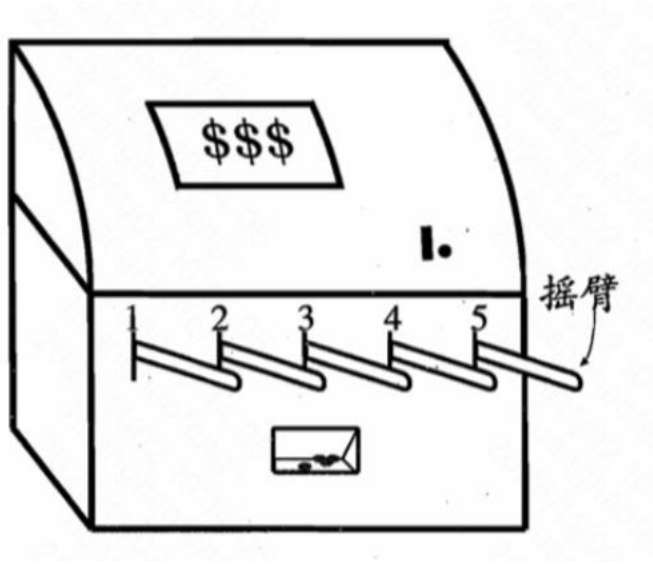
探索与利用

- 由于智能体通过强化学习会不断提升“当前评估下的最优动作”的概率，导致对于“当前评估下的词次优动作”采样概率下降。然而，对于随机场景，任意动作的准确评估需要进行多次采样，这意味着对于低概率动作的评估很有可能是不准确的。

- 在有限资源的情况下，上述困境必然需要在探索与利用之间进行取舍，即探索-利用窘境（exploration-exploitation dilemma）

探索与利用

- 一个经典的例子：K臂老虎机
- 每个摇臂对应一个奖励的随机分布。仅探索策略与仅利用策略各自代表什么？

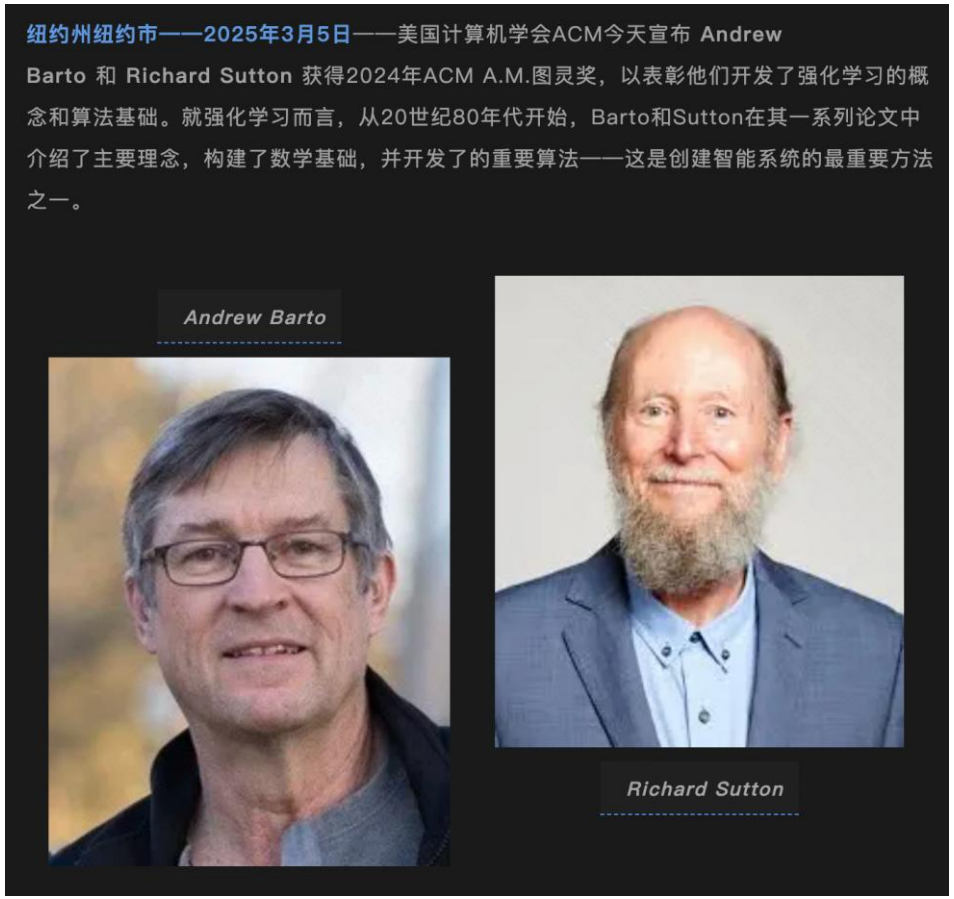


强化学习的应用

- 游戏AI
- Agent
- 自动驾驶
- 通用人工智能

大模型时代的RL

- 事实上，目前强化学习已经成为大模型在后训练阶段的标准流程



Outline

- 强化学习基础
- 策略梯度算法: REINFORCE, PPO
- RL for LLM: GRPO, GSPO
- RL for Diffusion: Flow-GRPO, ReFL

策略梯度算法

- 考虑时序，**轨迹（Trajectory）**定义为从开始到结束的所有环境和采用的动作

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

- 对于一阶马尔可夫智能体，其产生某个轨迹的概率为：

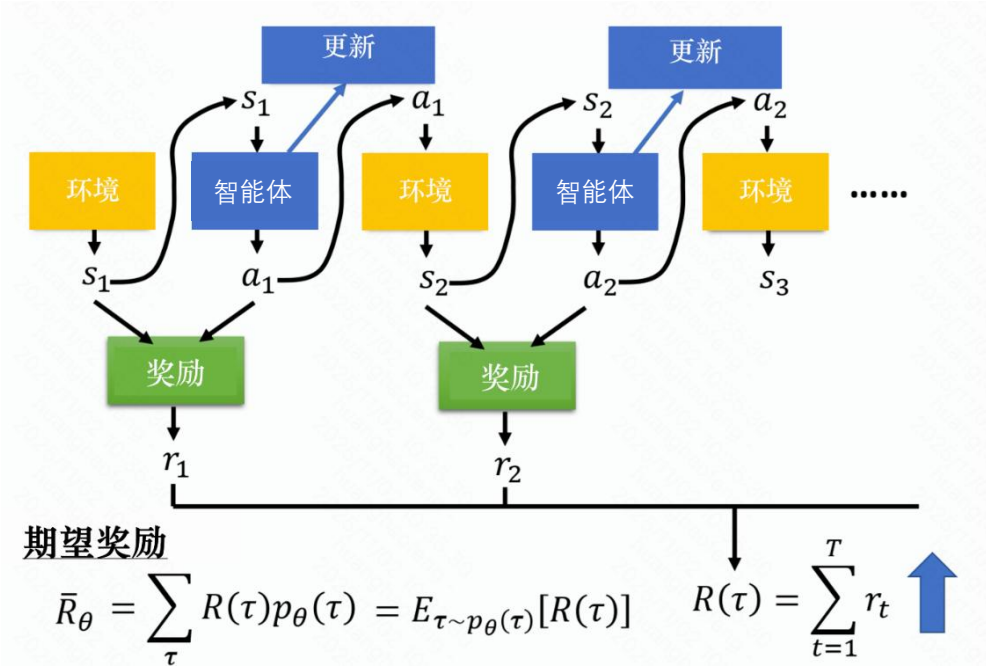
$$\begin{aligned} p_\theta(\tau) &= p(s_1)\, p_\theta(a_1|s_1)\, p(s_2|s_1, a_1)\, p_\theta(a_2|s_2)\, p(s_3|s_2, a_2) \cdots \\ &= p(s_1) \prod_{t=1}^{T} p_\theta(a_t|s_t)\, p(s_{t+1}|s_t, a_t) \end{aligned}$$

- 而将所有奖励加起来可以获得某个轨迹的奖励 $R(\tau)$

策略梯度算法

- 而智能体的目标是最大化其获得的奖励，即最大化奖励的期望

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



简单的推导…

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

监督学习 vs 基于策略的强化学习

- 事实上基于策略的强化学习的损失函数和最大化对数似然仅相差一个奖励

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

PyTorch / TensorFlow……

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

策略梯度算法的优化

1. 针对奖励设置baseline

reward的值通常设置为0-1，由于所有action的概率之和固定，最终会导致较低reward的action概率下降，较高reward的action概率上升。

然而，在训练过程中，采样每条轨迹（或者是少量的轨迹）都会更新智能体的策略，并不能保证所有action都被采样到，然而损失函数会使得被采样到的action的概率上升，未被采样到的概率下降

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

策略梯度算法的优化

1. 针对奖励设置baseline

因此，可以考虑对于reward设置baseline分数，使得较低reward对应的action概率下降：

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

策略梯度算法的优化

2. 针对轨迹中action计算其未来reward

整条轨迹获得的reward较高并不意味着每个action都是更优的。例如：

轨迹1在初期做了reward很高的action，但是后期再也没有获得reward

轨迹2在初期没有获得reward，但是后期获得了一定的reward

如果只计算整体reward，轨迹1后期的action将优于轨迹2后期的action，这是不合理的

策略梯度算法的优化

2. 针对轨迹中action计算其未来reward

因此，将原始的reward调整为：当前动作之后新获得的reward

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

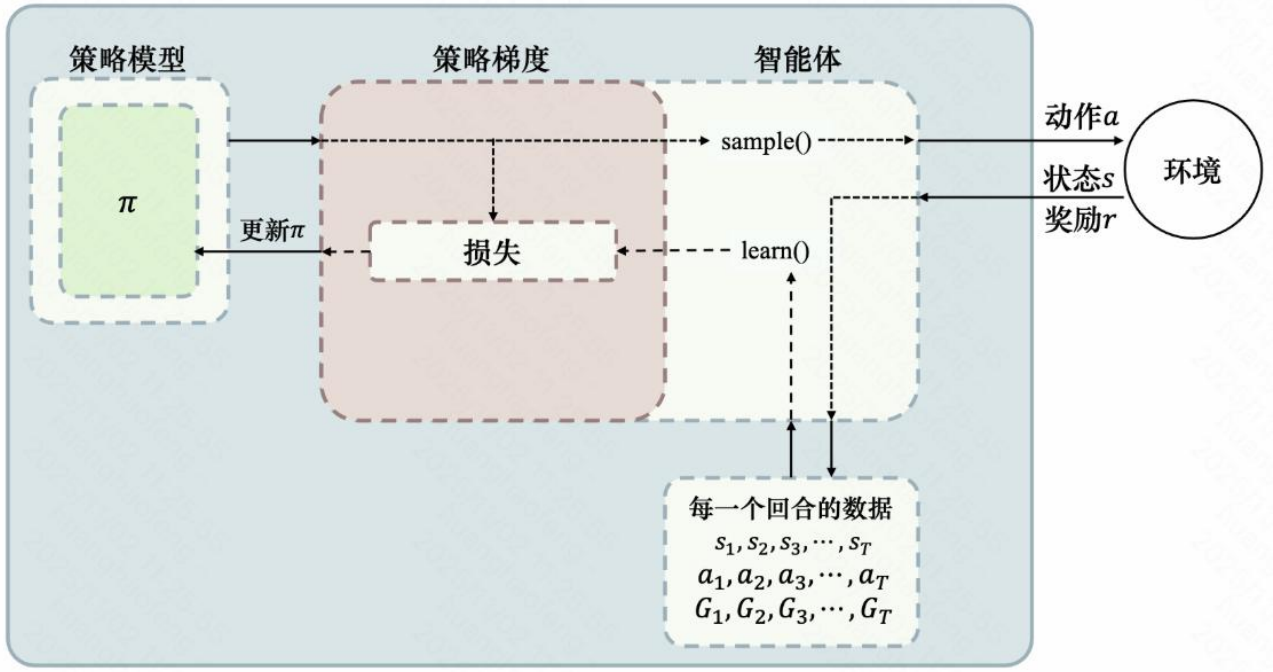
同时，不希望距其很远的reward影响当前action的判断（因为那可能来源于之后的action而非当前action），进行奖励折扣

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

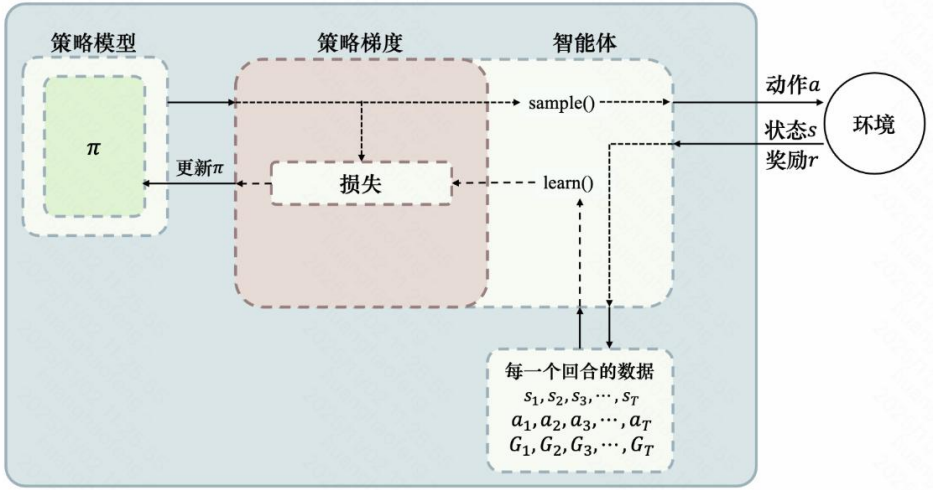
REINFORCE算法

REINFORCE 算法的具体算法流程如下：

- 初始化策略参数θ
- **for** 序列 $e = 1 \to E$ **do**：
- 用当前策略π_θ采样轨迹$\{s_1, a_1, r_1, s_2, a_2, r_2, \dots s_T, a_T, r_T\}$
- 计算当前轨迹每个时刻t往后的回报$\sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$，记为$\psi_t$
- 对θ进行更新，$\theta = \theta + \alpha \sum_t^T \psi_t \nabla_\theta \log \pi_\theta(a_t|s_t)$
- **end for**



同策略与异策略



同策略：sample 和 learn 是同一个模型

异策略：sample 和 learn 不是同一个模型

为什么需要异策略？一旦进行learn，之前sample出的样本就不再match当前的概率分布。因此同策略需要花费大量时间在sample上。

采用异策略需要进行**重要性采样**，否则二者的概率分布不同，奖励的期望会有偏

重要性采样

重要性采样旨在改变所求期望的分布

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

重要性采样后的整体奖励计算:

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

重要性采样后的每个action优势计算:

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

重要性采样

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

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

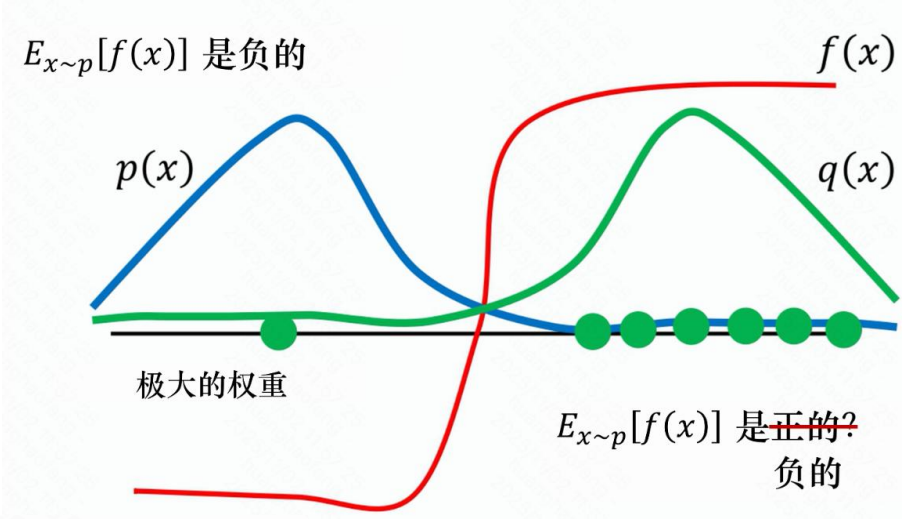
忽略 $p_\theta(s_t)$ 和 $p_{\theta'}(s_t)$ 之间的差异，有

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

PPO算法

- 对于异策略，如果两个策略相差太大会有问题

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



- 因此，PPO算法提出
  - 使用KL散度进行约束 $J_{\mathrm{PPO}}^{\theta'}(\theta) = J^{\theta'}(\theta) - \beta \mathrm{KL}(\theta, \theta')$
  - 使用clip进行约束

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

PPO算法

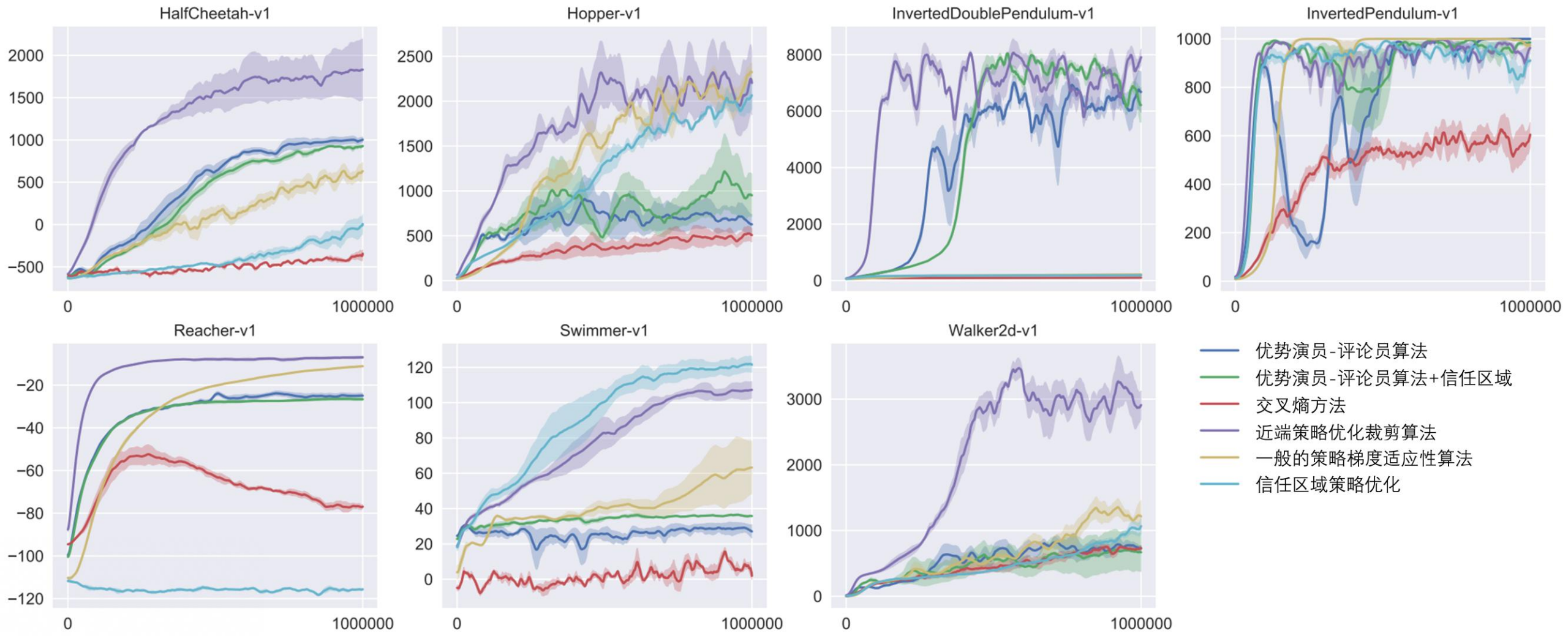
- 对于优势函数 $A^{\theta}(s_t, a_t) = \sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b$ ，PPO算法使用一个value model来建模当前状态的价值

$$b = V_{\pi}(s)$$

- 如前所述，对奖励进行衰减后，可得Generalized Advantage Estimation （没看懂也没关系）

$$A_t = \delta_t + \lambda\gamma A_{t+1}$$

其中 $\delta_t = r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$ 为value model的TD error

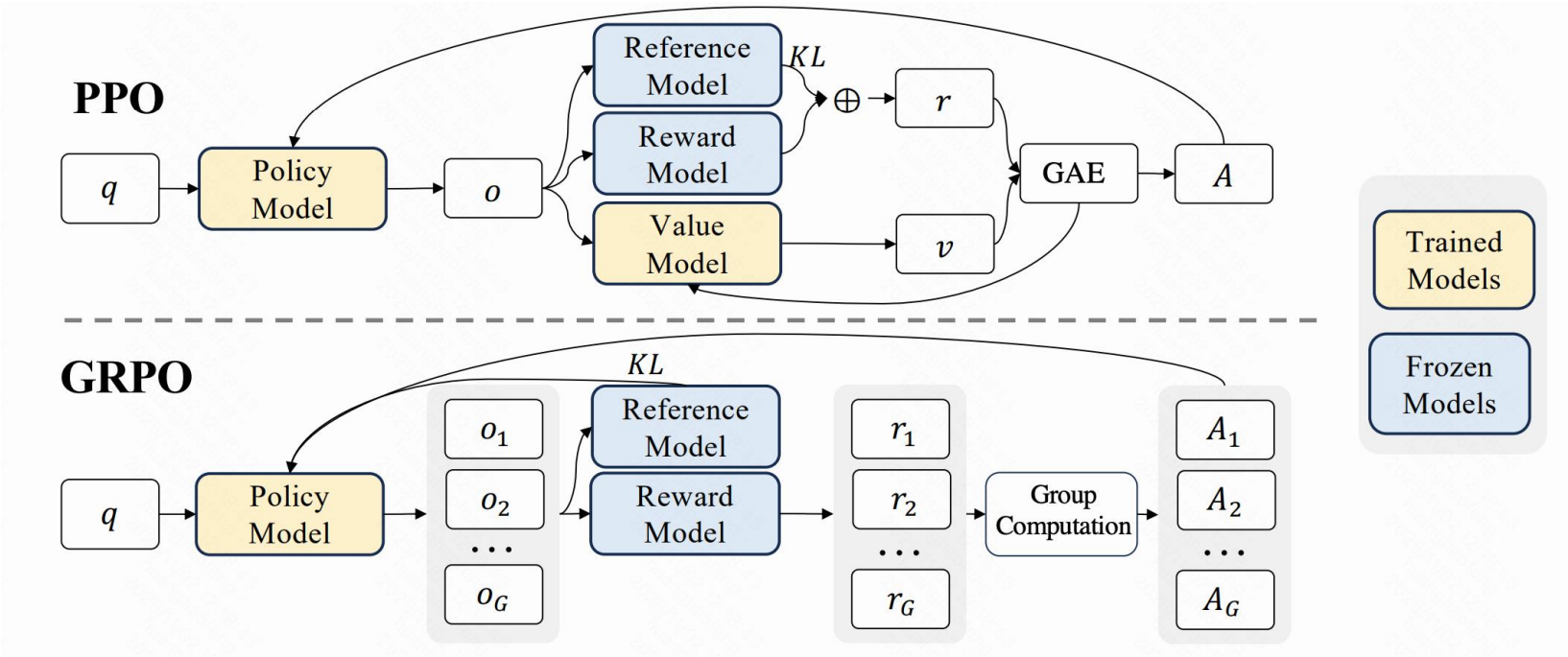

HalfCheetah-v1
Hopper-v1
InvertedDoublePendulum-v1
InvertedPendulum-v1
Reacher-v1
Swimmer-v1
Walker2d-v1
0
1000000
优势演员-评论员算法
优势演员-评论员算法+信任区域
交叉熵方法
近端策略优化裁剪算法
一般的策略梯度适应性算法
信任区域策略优化

Outline

- 强化学习基础
- 策略梯度算法: REINFORCE, PPO
- RL for LLM: GRPO, GSPO
- RL for Diffusion: Flow-GRPO, ReFL

GRPO算法*

- DeepSeek提出的GRPO算法主要在action value的预测上进行了优化



* DeepSeek, DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models

GRPO算法

- 具体的，GRPO对于当前状态采样多次，使用组内相对优势来替代掉PPO算法中的GAE

$$\hat{A}_{i,t} = \widetilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$$

- 整体的目标函数为：

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right] - \beta\mathbb{D}_{KL}\left[\pi_\theta||\pi_{ref}\right]\right\}$$

- 这一改进解决了value model的训练困难

Model	Size	English Benchmarks		Chinese Benchmarks	
		GSM8K	MATH	MGSM-zh	CMATH
Chain-of-Thought Reasoning					
Closed-Source Model					
Gemini Ultra	-	94.4%	53.2%	-	-
GPT-4	-	92.0%	52.9%	-	86.0%
Inflection-2	-	81.4%	34.8%	-	-
GPT-3.5	-	80.8%	34.1%	-	73.8%
Gemini Pro	-	86.5%	32.6%	-	-
Grok-1	-	62.9%	23.9%	-	-
Baichuan-3	-	88.2%	49.2%	-	-
GLM-4	-	87.6%	47.9%	-	-
Open-Source Model					
InternLM2-Math	20B	82.6%	37.7%	-	-
Qwen	72B	78.9%	35.2%	-	-
Math-Shepherd-Mistral	7B	84.1%	33.0%	-	-
WizardMath-v1.1	7B	83.2%	33.0%	-	-
DeepSeek-LLM-Chat	67B	84.1%	32.6%	74.0%	80.3%
MetaMath	70B	82.3%	26.6%	66.4%	70.9%
SeaLLM-v2	7B	78.2%	27.5%	64.8%	-
ChatGLM3	6B	72.3%	25.7%	-	-
WizardMath-v1.0	70B	81.6%	22.7%	64.8%	65.4%
DeepSeekMath-Instruct	7B	82.9%	46.8%	73.2%	84.6%
DeepSeekMath-RL	7B	**88.2%**	**51.7%**	**79.6%**	**88.8%**

Tool-Integrated Reasoning					
Closed-Source Model					
GPT-4 Code Interpreter	-	97.0%	69.7%	-	-
Open-Source Model					
InternLM2-Math	20B	80.7%	54.3%	-	-
DeepSeek-LLM-Chat	67B	86.7%	51.1%	76.4%	85.4%
ToRA	34B	80.7%	50.8%	41.2%	53.4%
MAmmoTH	70B	76.9%	41.8%	-	-
DeepSeekMath-Instruct	7B	83.7%	57.4%	72.0%	84.3%
DeepSeekMath-RL	7B	**86.7%**	**58.8%**	**78.4%**	**87.6%**

GSPO算法

* Qwen, Group Sequence Policy Optimization

GSPO算法

- GSPO算法主要对GRPO对于token级别的重要性采样进行了优化

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\,\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\left(w_{i,t}(\theta)\widehat{A}_{i,t},\ \text{clip}\left(w_{i,t}(\theta),1-\varepsilon,1+\varepsilon\right)\widehat{A}_{i,t}\right)\right]$$

$$w_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t}|x,y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|x,y_{i,<t})}$$

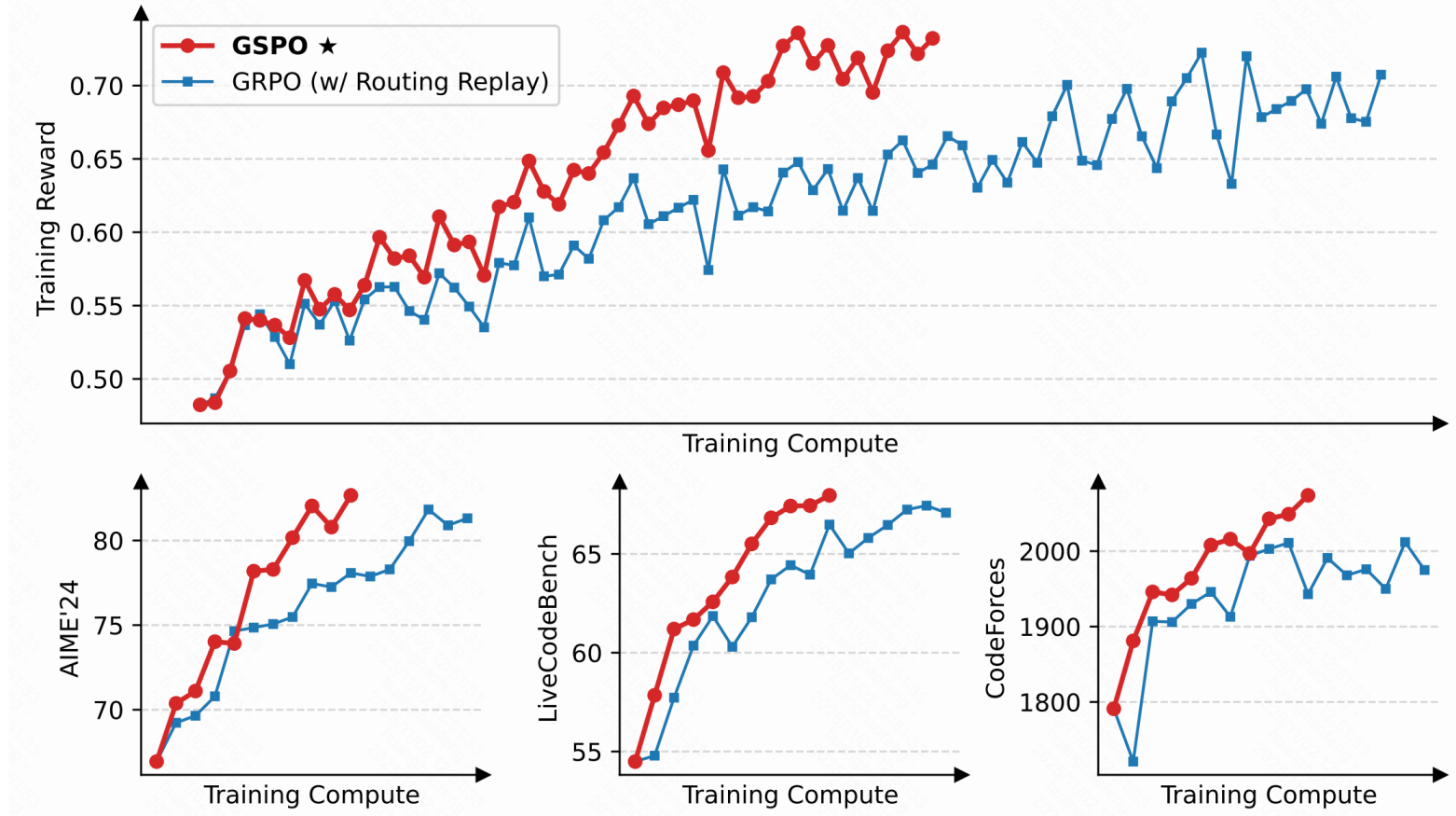
$$\mathcal{J}_{\text{GSPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\,\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\min\left(s_i(\theta)\widehat{A}_i,\ \text{clip}\left(s_i(\theta),1-\varepsilon,1+\varepsilon\right)\widehat{A}_i\right)\right]$$

$$s_i(\theta) = \left(\frac{\pi_\theta(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}\right)^{\frac{1}{|y_i|}} = \exp\left(\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\log\frac{\pi_\theta(y_{i,t}|x,y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|x,y_{i,<t})}\right)$$

GSPO算法

- 这样做的目的是为了防止trajectory中某些token由于采样导致了极高的重要性系数，而这些异常高的系数会累积到一起导致高方差
- 这一问题在MoE模型上更为严重，研究者往往需要使用Route Replay的方式，即激活相同的专家来解决

- 而GSPO的做法从sequence级别缓解了方差，超越了GRPO+Route Replay的效果



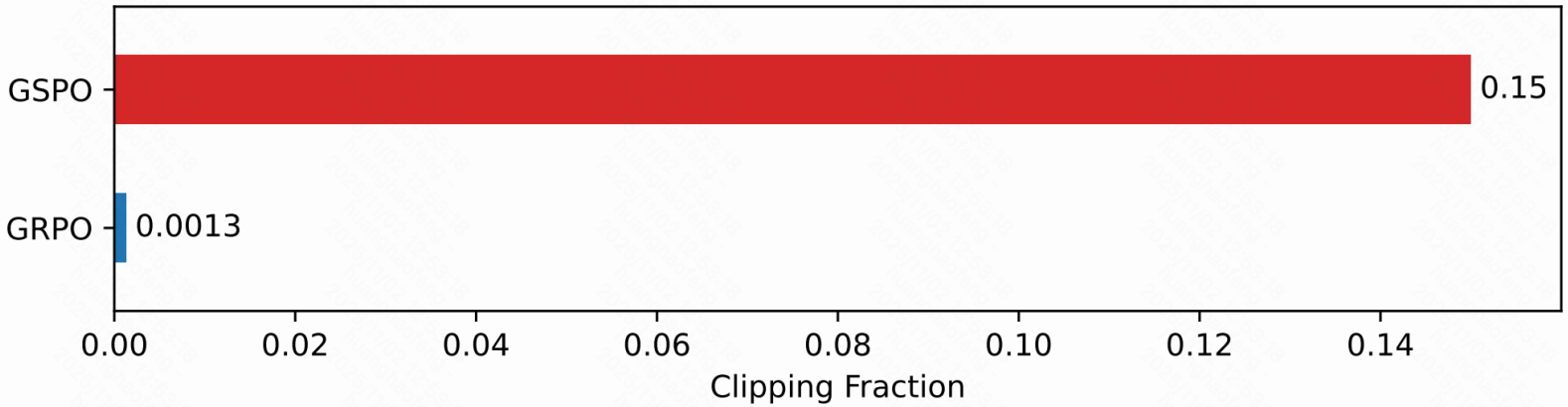


Figure 2: Average fractions of clipped tokens over the RL training of GSPO and GRPO.

Outline

- 强化学习基础
- 策略梯度算法: REINFORCE, PPO
- RL for LLM: GRPO, GSPO
- RL for Diffusion: Flow-GRPO, ReFL

Flow-GRPO

- 将GRPO迁移到diffusion过程是直观的
  - Diffusion本身就是马尔可夫过程，model根据当前状态预测分布间的向量
  - Diffusion本身作为生成模型具有随机性

- 而现有的大部分图像生成模型采用ReFlow架构，直接训练ODE，在采样过程中没有引入随机性。因此Flow-GRPO提出将ODE转化为对应的SDE进行采样

Flow-GRPO

- 对于ODE

$$\mathrm{d}\boldsymbol{x}_t = \boldsymbol{v}_t \mathrm{d}t,$$

- 有SDE

$$\mathrm{d}\boldsymbol{x}_t = \left[\boldsymbol{v}_t(\boldsymbol{x}_t) + \frac{\sigma_t^2}{2t}\left(\boldsymbol{x}_t + (1-t)\boldsymbol{v}_t(\boldsymbol{x}_t)\right)\right]\mathrm{d}t + \sigma_t \mathrm{d}\boldsymbol{u}$$

* 事实上，这个推导有小bug，详见https://arxiv.org/abs/2509.05952

Flow-GRPO

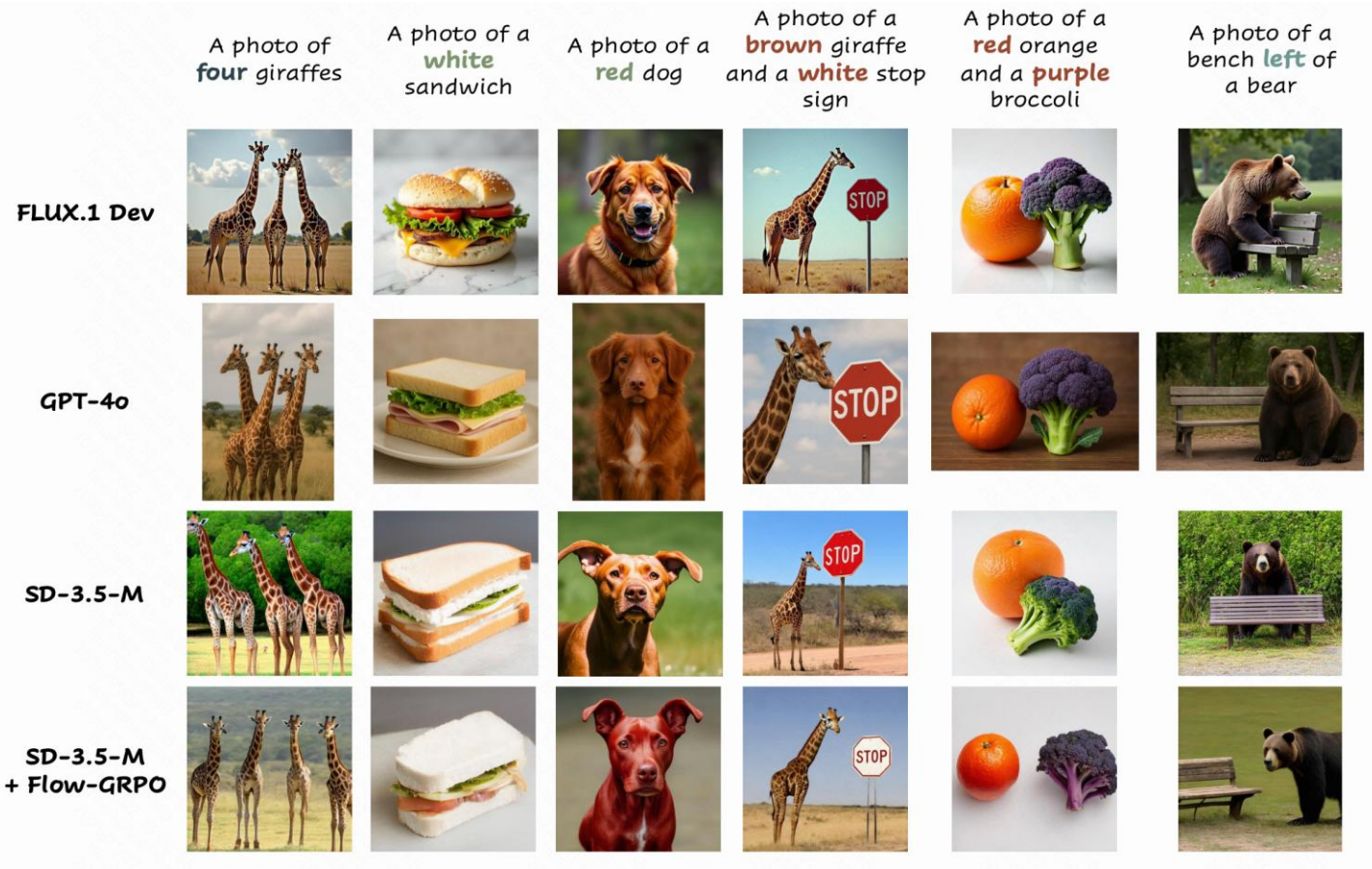
- 而后按照GRPO公式进行损失函数的计算。此处Flow-GRPO同样使用逐timestamp的重要性采样

$$f(r, \hat{A}, \theta, \varepsilon, \beta) = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{T}\sum_{t=0}^{T-1}\left(\min\left(r_t^i(\theta)\hat{A}_t^i,\ \text{clip}\Big(r_t^i(\theta), 1-\varepsilon, 1+\varepsilon\Big)\hat{A}_t^i\right) - \beta D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})\right)$$

$$r_t^i(\theta) = \frac{p_\theta(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_t^i, \boldsymbol{c})}{p_{\theta_{\text{old}}}(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_t^i, \boldsymbol{c})}.$$

Flow-GRPO

- 算法主要优化了文生图模型的图文一致性



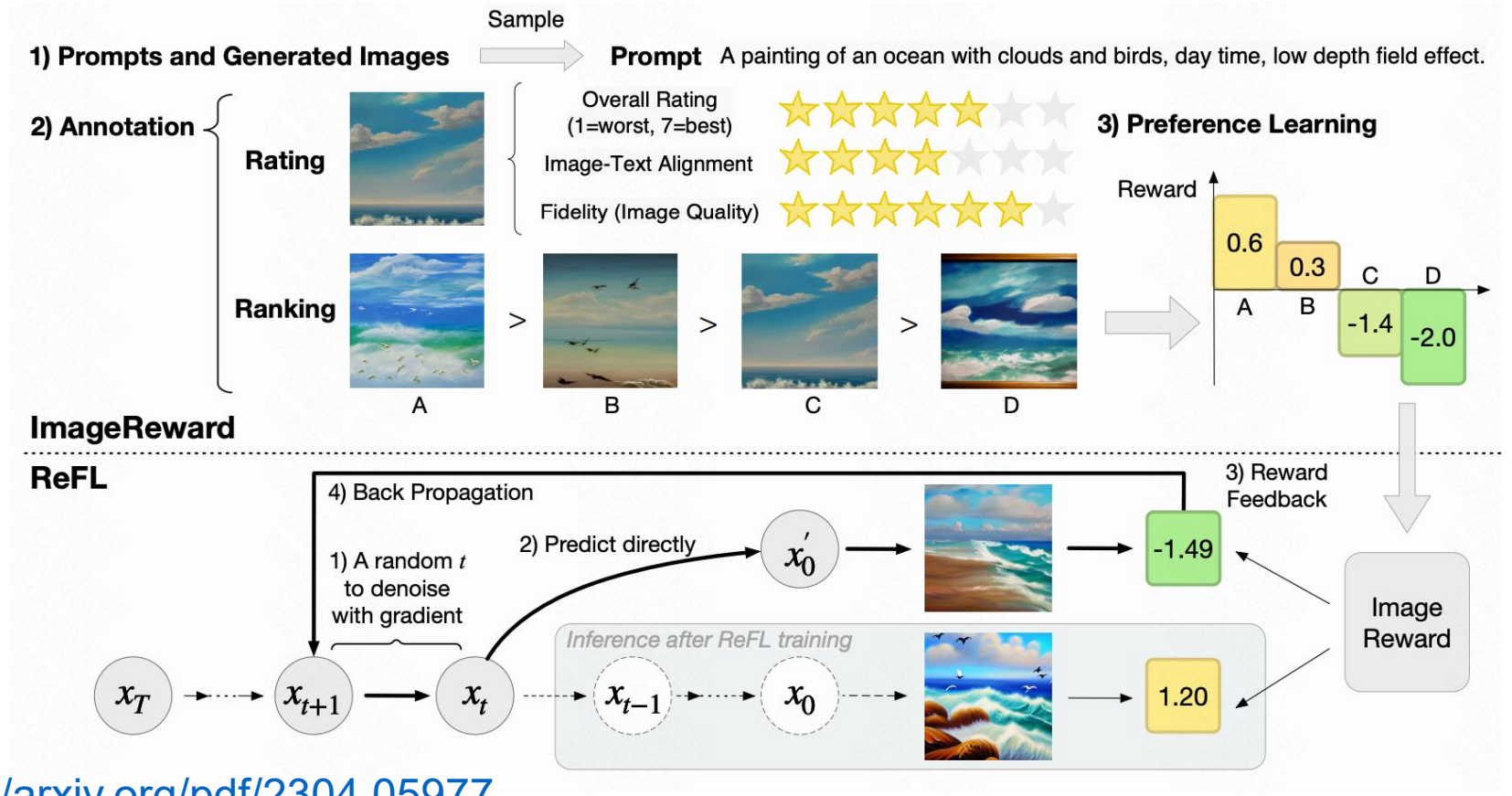
ReFL

- ReFL (Reward Feedback Learning) 严格意义上并不属于RL的范畴，但也多被用于图像生成模型的后训练。字节Seedream、腾讯Hunyuan均采用了该技术路线

- ReFL的核心思想是：
  - 通过神经网络学习人类偏好，构建reward model
  - 固定reward model参数，优化当前模型以获得更高的reward分数

ReFL

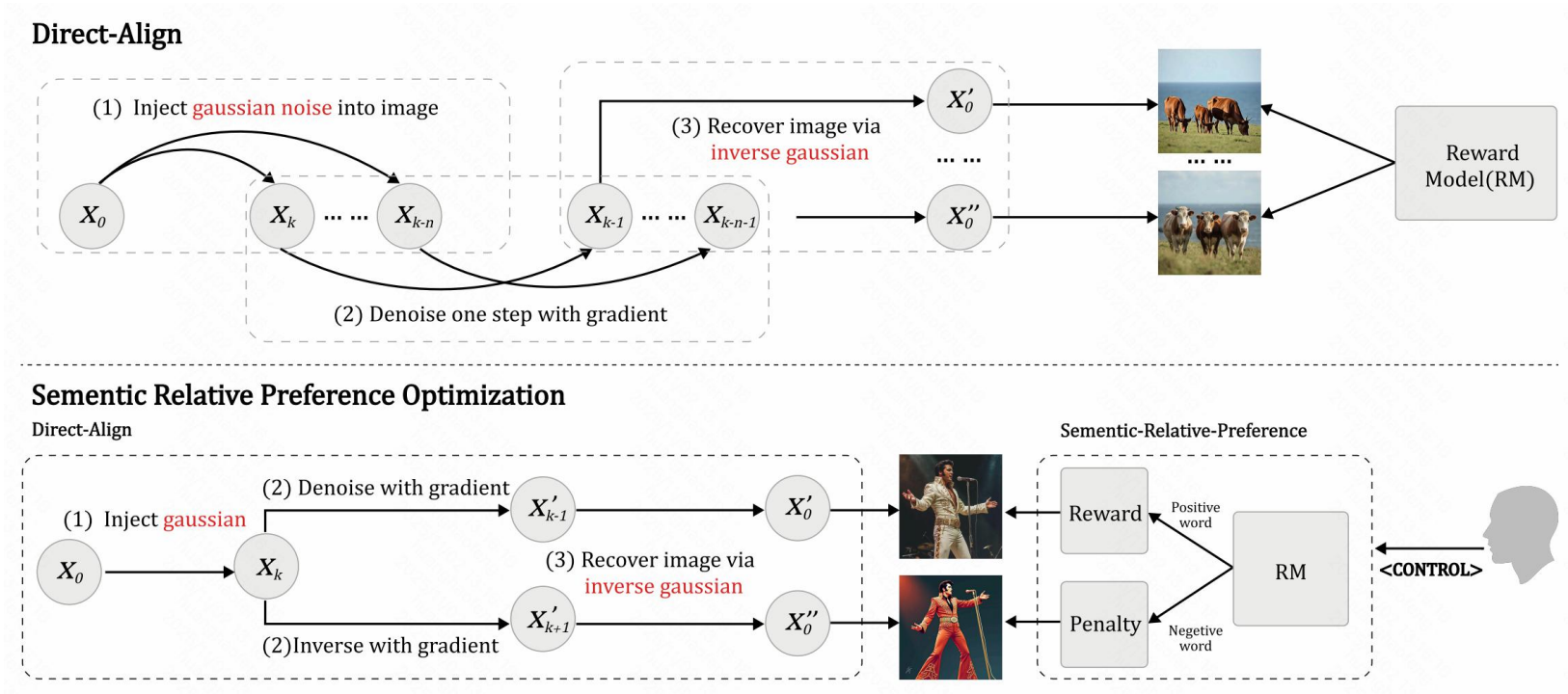
- Image Reward *



* https://arxiv.org/pdf/2304.05977

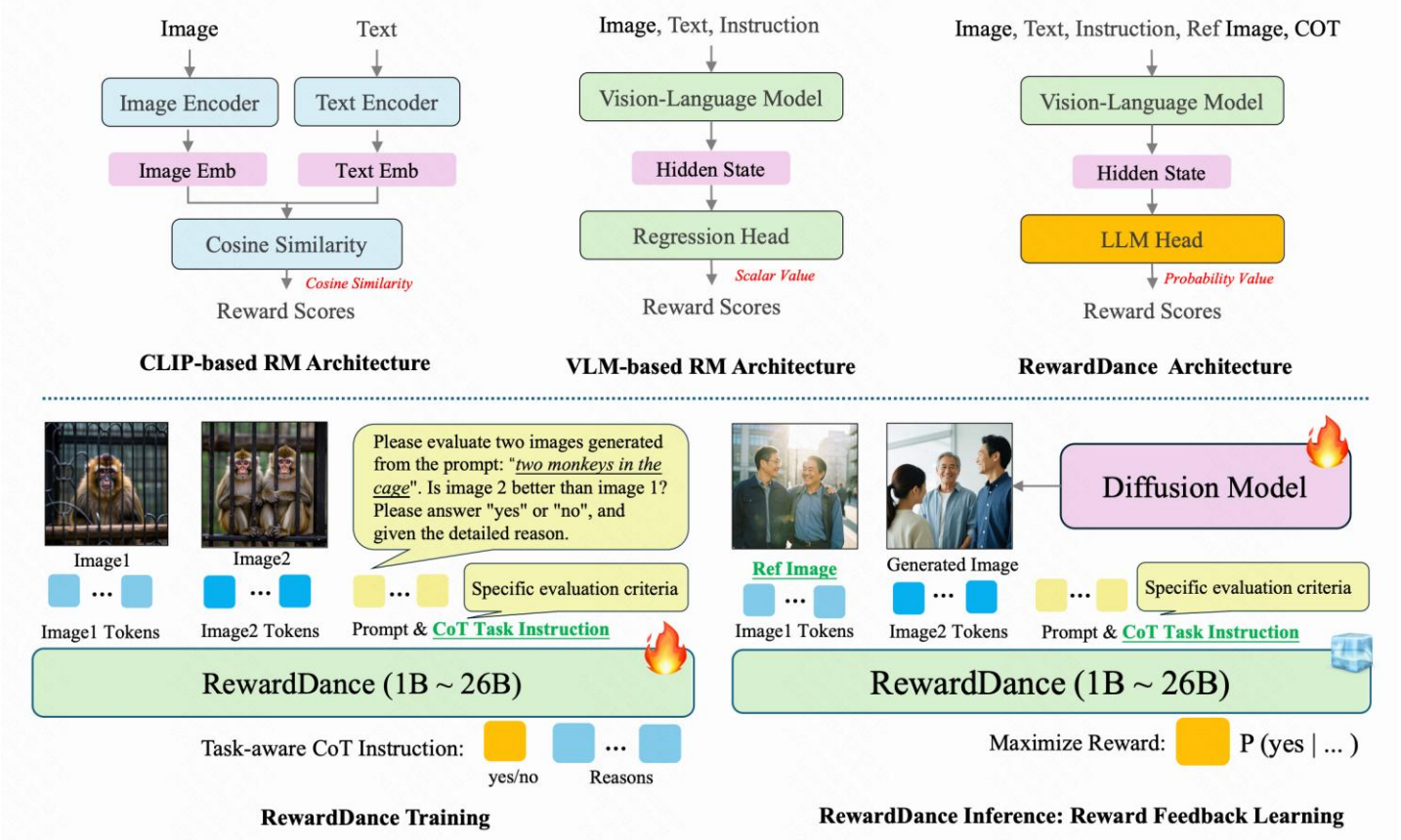
SRPO (Hunyuan 改进版 ReFL)

- 记录前向噪声进行逆向预测
- 使用正反reward进行优化，防止单一方向产生hacking



RewardDance (Seedream 改进版 ReFL)

- VLM-based Reward Model Scaling
- 使用成对数据，CoT以及 token prob. 构建 reward 分数



总结

- 目前强化学习，尤其是Policy-based算法在文字和图像生成领域正在被广泛使用，以提高模型的推理、理解、人类偏好对齐等能力。但还存在一系列问题：如奖励稀疏、reward难以定义及构造、生成质量多样性降低，reward hacking等

- 而强化学习本身的训练是较为快速的，基本上2-3天可以进行一次新的尝试。所需要的额外显存也取决于setting

- 比较热门的研究内容包含：生成分布的熵、reward model的置信度、提升采样利用率等方向。