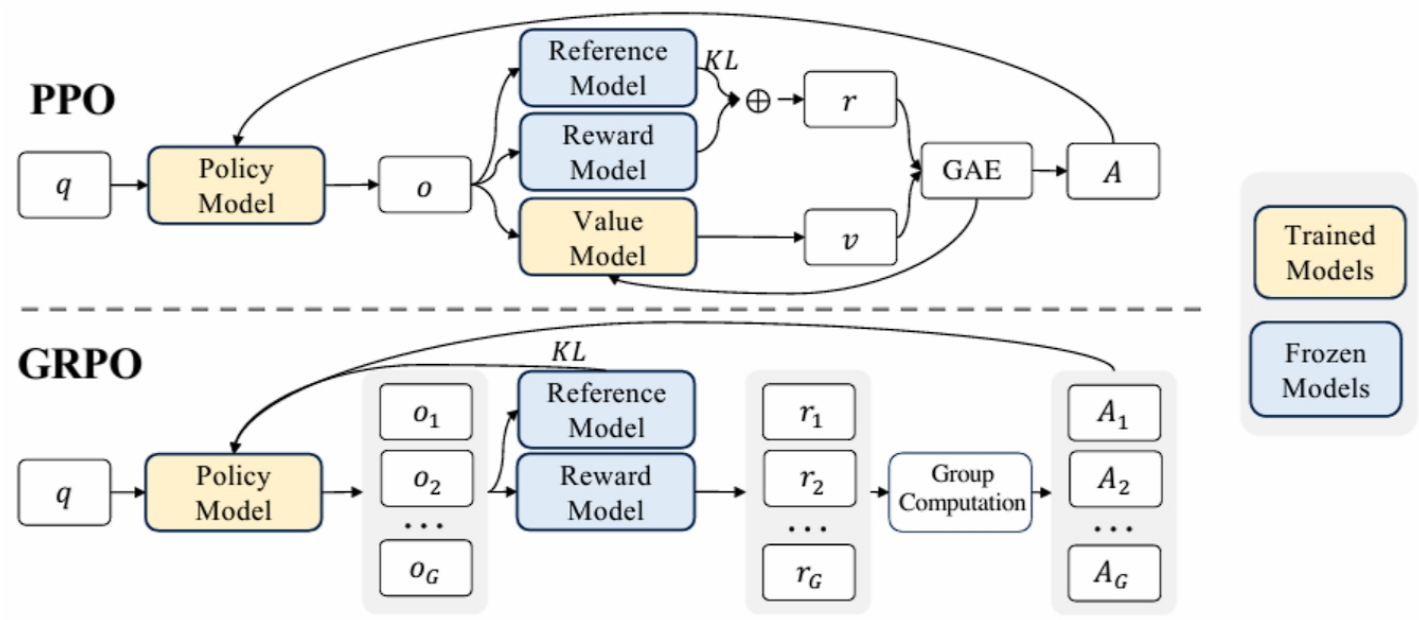
GRPO

这是一种类似于PPO的强化学习算法，核心思想是通过限制策略更新的幅度，确保新策略与旧策略的差异不会过大，从而稳定训练）。



GRPO算法采用群体分数取代了评估模型。对于用户问题q，从旧策略模型 $\pi_{\theta_{old}}$ 采样一系列的输出{o1,o2,· · · ,oG}，最大化以下目标：

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

$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right), \tag{1}$$

$$\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1, \tag{2}$$

其中 A_i 是Advantage，它使用对应于每个组内输出的一组奖励{ $r_1, r_2 \ldots\ldots r_G$ }计算：

$$A_i = \frac{r_i - mean(\{r_1, r_2, \cdots, r_G\})}{std(\{r_1, r_2, \cdots, r_G\})}. \tag{3}$$

GRPO原论文设置G=64，通过从旧策略中采样大量的输出，增大了能探索到正确答案的概率。

对目标函数的详解：

- $\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}$ 表示新模型和旧模型在同一输入下得到同一输出的概率比，用于限制更新速度，避免原始模型能力的丧失。

- clip函数也是为了将更新限制在一定范围内，即将 $\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}$ 限制在 $[1-\epsilon, 1+\epsilon]$ 之间。
- 采用Advantage而不是reward，目的是为了考虑当前回答相比其他回答的优劣，最大化生成更高Advantage回答的概率。（PPO是对比当前回答和参考策略回答的优劣）。
- 与PPO中Advantage的计算相比，GRPO只需要奖励模型，减少了推理的内存占用和时间，但缺点是只关注当下表现，忽略了长期回报。
- KL散度惩罚项，强制新策略与参考策略 π_{ref} 接近，也是为了防止策略偏离特定约束（如安全策略或预训练模型）。

🐵 **对比PPO、GRPO、DPO：**

- **GRPO：一组prompt经过LLM生成N个结果，每个结果打分，超过平均的被接受，反之被拒绝**
- **PPO：一组prompt经过LLM生成结果，对结果打分，和上一个的自己比较，有优势被接受，反之被拒绝**
- **DPO：提前构造好了拒绝和接受的数据，模型参数往接受的方向调整**

🎹 **去掉评估模型，GRPO可能面临的问题：**

- **奖励估计不准确**：Critic Model的核心作用是**评估状态或动作的长期价值。在某些任务中，奖励可能只有在完成特定的长期目标后才能获得，这种场景中失去Critic Model会导致奖励估计不准确，从而导致训练不稳定。**

以下是DAPO论文中总结的GRPO面临的问题：

- **熵崩溃**：策略的熵（不确定性）逐渐趋近于零，导致智能体行为过于确定化，失去探索能力。
- **奖励噪音**：奖励信号存在随机波动或误差，导致模型接收到的奖励信息不准确、不稳定，从而影响策略学习的效果。
- **训练不稳定**：在强化学习模型的训练过程中，性能指标（如奖励、损失等）出现大幅波动、震荡或不收敛的现象，导致模型无法稳定地学习到有效的策略。

KL-散度设计

KL-散度的计算公式为：

$$KL(q||p) = \sum_x q(x) log \frac{q(x)}{p(x)} = E_{x\sim q}[log \frac{q(x)}{p(x)}]$$

KL 散度非负，当且仅当 $q(x) = p(x)$ 时，KL散度为0。实际计算中，直接计算KL散度很难，因为需要对所有的x求和或求积分，计算成本和内存效果过大。因此在实际计算中，往往使用近似方法计算KL-

散度。

K1估计（无偏高方差）

定义 $r = \frac{p(x)}{q(x)}$，K1估计为：

$$K1 = log\frac{q(x)}{p(x)} = -log(r)$$

K1是对KL-散度的无偏估计：

$$E[k1] = E_{x\sim q}[log\frac{q(x)}{p(x)}] = KL(q||p)$$

> **KL-散度和K1估计的区别**
>
> 第一次看到这里的你是否会感觉KL-散度和K1估计没啥区别，nonono，KL-散度的计算量比K1估计大很多。
>
> 具体来说，KL-散度的本质是衡量分布 q 相对于分布 p 的信息损失。对于离散分布，需要计算所有可能事件的概率再求和（对于大模型来说就是整个词表中每个token的概率）；对于连续分布，则需要计算积分，这都是需要大量计算的。
>
> 而K1估计中， q是通过策略生成事件的概率，p是目标策略生成事件概率。K1估计只需要计算单个事件的概率，计算量大减。

K1估计的方差很大，若 q(x)<p(x)，k1 为负值，导致估计值波动大。

K2估计（有偏低方差）

K2估计定义为：

$$K2 = \frac{1}{2}(log\frac{p(x)}{q(x)})^2 = \frac{1}{2}(logr)^2$$

K2确保所有事件的值都是正数，使其在计算中更加稳定。

K3估计（无偏低方差）

$$K3 = \frac{p(x)}{q(x)} - 1 - log\frac{p(x)}{q(x)} = r - 1 - logr$$

由于 $E(-logr) = KL(q||p)$，$E(r-1) = E_{x\sim q}[\frac{p(x)}{q(x)}] = \int q(x)\frac{p(x)}{q(x)}dx = \int p(x)dx = 1$

K3 估计是KL-散度的无偏估计。并且通过求导可知，K3估计的值恒大于0，因此方差也更低。**GRPO中采用了K3估计。OpenRLHF库中ppo算法采用k1估计和k3估计计算KL散度，区别是将KL约束加在奖励之中：**

$$R_t = \begin{cases} -kt\_ctl * (log\frac{\pi(a_t|s_t)}{\pi_{ref}(a_t|s_t)}), t \neq T \\ -kt\_ctl * (log\frac{\pi(a_t|s_t)}{\pi_{ref}(a_t|s_t)}) + R_t, t = T \end{cases}$$

- 当 t≠T 时，我们更加关心Actor是否有在Ref的约束下生产token a_t (T表示最后一个时刻)
- 当 t=T 时，我们不仅关心Actor是否遵从了Ref的约束，也关心真正的即时收益 R_t

优势函数设计

结果监督强化学习

也就是采用**ORM（Outcome Reward Model）**，对于问题q，从旧策略 $\pi_{\theta_{old}}$ 采样一系列的输出{o1,o2,$\cdots$,oG}，然后用奖励模型对输出进行评分，得到G个奖励 $r=\{r1,r2,...,rG\}$，计算出优势后将每个token的优势值都设置为此优势值。

$$A_{i,t}=\frac{r_i-mean(r)}{std(r)}$$

也就是对采样出的这一组输出的奖励计算归一化奖励作为每个输出的优势函数，每个输出的优势赋值给输出中每个token作为该token的优势函数。

> **为什么用奖励值表示优势**
>
> 回顾贝尔曼方程中价值动作函数的定义：
>
> $$Q_\pi(s,a)=E_\pi[G_t|S_t=s,A_t=a]=E_\pi[R_t+\gamma R_{t+1}+...|S_t=s,A_t=a]$$
>
> 将从问题q到输出o的过程视为**单步MDP**，这样状态就是q，动作就是 o_i，对应的奖励也就是 $r(q,o_i)$,价值动作函数可以表示为：
>
> $$Q(s,a)=E_\pi[R_t|S_t=s,A_t=a]=r(q,o_i)$$
>
> 只要采样个数足够多，就可以用 $mean(r)=\frac{1}{G}\sum_{i=1}^{G}r(q,o_i)$ 可来估计 $V(q)$，因此优势 $A(q,o_i)=Q(q,o_i)-V(q)$,除以 $std(r)$ 是为了进行归一化。
>
> 这样计算出的优势是对整个回复的，GRPO将每个token的优势值都设置为此值，表示使用当前策略去更新输出 o_i 中每一个token的条件输出概率 $\pi_\theta(a_t|s_t)$ 的更新幅度和方向是一样的。

过程监督强化学习

采用**PRM(Process-supervised Reward Model)**，为每个输出的每个token都计算奖励值：

$$r=\{\{r_1^{index(1)},...,r_1^{index(K_1)}\},...,\{r_G^{index(1)},...,r_G^{index(K_G)}\}\}$$

其中 $index(j)$ 表示第j个步骤的end token的index， K_i 表示第i个输出的步骤总数，第i个输出的第t个步骤的优势为：

$$A_{i,t}=\sum_{index(j)\geq t}\frac{r_i^{index(j)}-mean(r)}{std(r)}$$

输出实际是由多个步骤组成的，所以当前步骤的优势应该包含后续步骤的期望，即当前步骤的优势为当前步骤和后续步骤的优势之和。这里的步骤可以理解为token，也就是每个token的优势为当前

token和后续token的优势之和。