大模型面经

网友面经汇总

1. 阿里云瓴羊大模型算法一面 时长：45min

1. 为啥先sft再grpo

> 要点：先让模型“会说”，再让它“说得对”。
>
> - **SFT**（监督微调）：用高质量指令/示例让模型学到**格式、基本对齐与可用性**，稳定起步、降低 RL 难度。
> - **GRPO**：在会说的前提上，用奖励去**优化目标行为**（正确率、可执行性、遵从规范等），避免 RL 从零噪声起步。
> - 先 SFT 可显著**提高采样质量**，让 RL 的探索空间更小、收敛更稳，减少崩坏/跑偏。

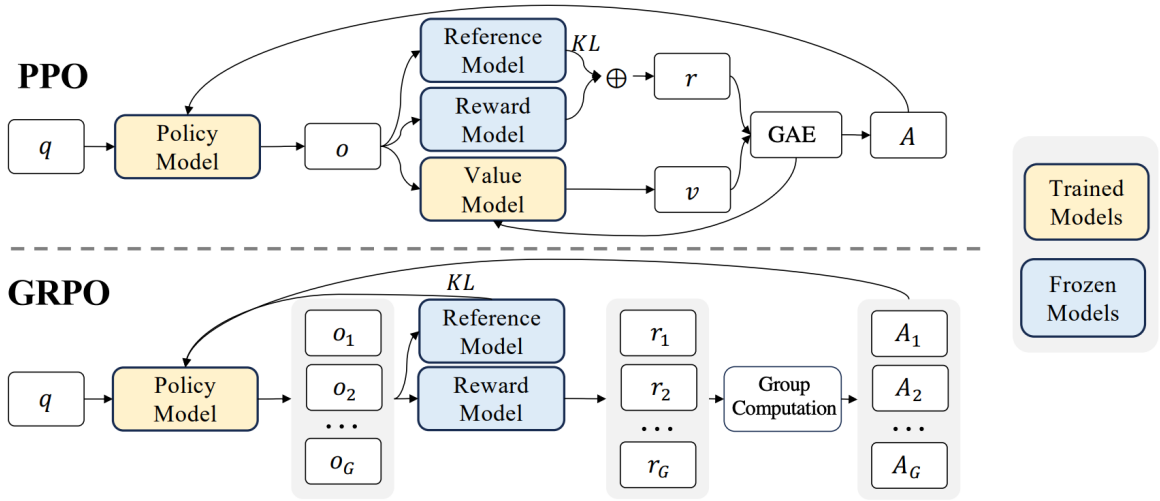
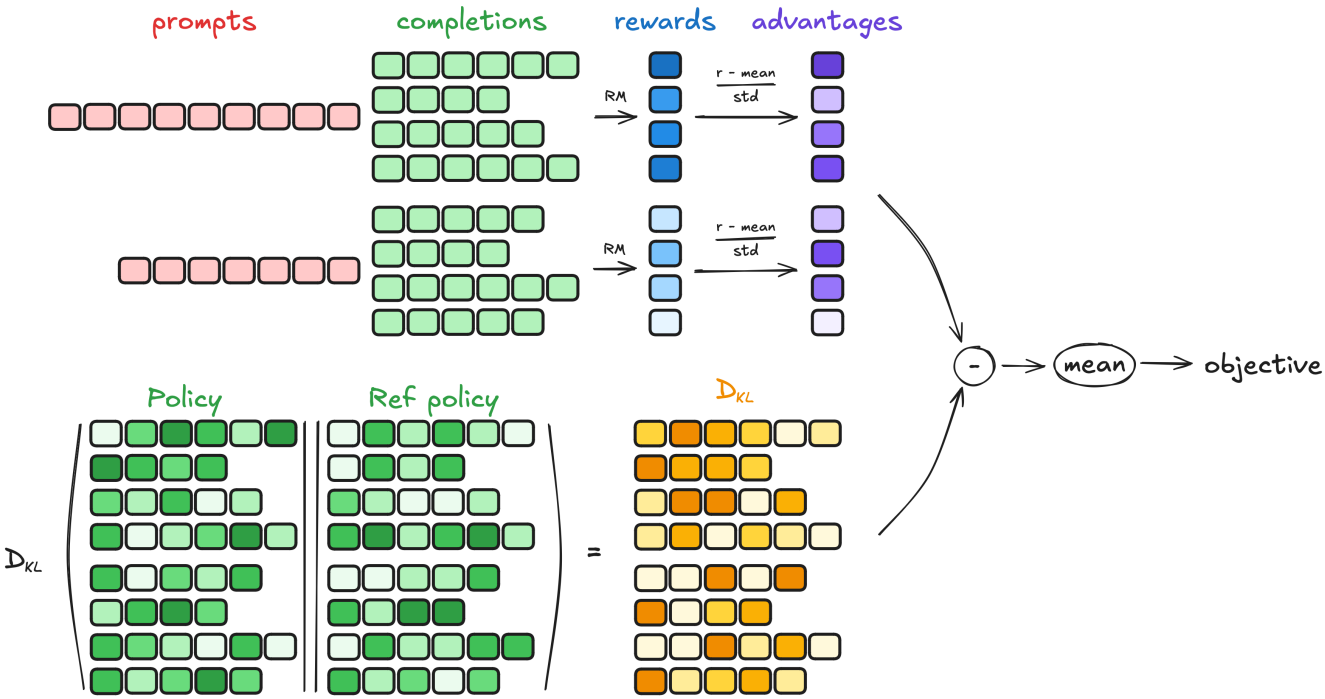


Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.



GRPO目标：

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}_{q\sim\rho(Q),\{O_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|O_i|}\sum_{t=1}^{|O_i|} \min\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon \right)\hat{A}_{i,t} \right) - \beta D_{KL}\left[\pi_\theta \| \pi_{\text{re}}\right.$$

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

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

2. grpo的loss函数?奖励加在哪的?

“奖励到底加在了哪里”=一句话版：

> 在 GRPO 里，**奖励不会加到 logits 里**，也不会改 token 概率；它被做成**优势** A_i，然后**乘在整段已生成 token 的对数似然和上形成策略梯度项**：
> `loss_pg = - mean( A_i.detach() * sum_over_tokens(log πθ(a_t|s_t)) )`。
> $A_i > 0$ 的样本，其对应 token 的对数似然被“放大”（梯度上升）；$A_i < 0$ 的被压低（梯度下降）。

下面我按你这份骨架代码，把“奖励 → 优势 → 乘到 log-prob 上”的具体落点**逐行串起来**，并给出可直接塞进你 `GRPOTrainer` 的实现草稿（核心几行清清楚楚看到奖励乘在什么地方）。

一、整体流程

1. `generate_samples`：对**同一条 prompt**采样 **K** 个回答（组内采样）。
    1. 用传进来的 `reward_funcs = [correctness_reward, digit_reward, hard_format_reward, mark_reward]` **逐个回答算奖励，再加权/求和**得到标量 r_i。
2. 组内做**标准化优势**：$A_i = \frac{r_i - \bar{r}}{\sigma_r + \epsilon}$。
3. `get_action_log_probs`：拿到**这条回答对应的最后 N 个生成 token 的 log-prob**，把它们**求和**得到 $\sum_t \log \pi_\theta(a_t|s_t)$。
4. **策略梯度损失**（奖励真正起作用的地方）：

```
loss_pg = -(A.detach() * action_log_probs_sum).mean()
```

   这行就是“奖励乘在对数似然上”。

5. **KL 约束**：对同一段生成，算 $\mathrm{KL}(\pi_\theta || \pi_{\text{ref}})$（一般是与 SFT 冻结参考模型），再

```
loss = loss_pg + beta * loss_kl
```

   这样既朝着“高奖励”的方向推，也不至于漂离参考策略。

要点：组内标准化优势 * 对数似然 + KL 约束，无价值网络。

- 典型做法：对同一 prompt 采样 K 个回答，记奖励 r_i。

- **组内标准化优势**：$A_i = \frac{r_i - \bar{r}}{\sigma_r + \epsilon}$。

  > 衡量奖励优势，$\bar{r}$是奖励的平均值，σ_r奖励的标准差，ϵ是一个很小的数，防止分母为0
  >
  > A_i用来衡量当前这个样本的奖励相当于组内其他样本的的优势有多大

- **目标**（无 critic 的 PPO 变体,损失函数）:$L_{\text{GRPO}} = -\mathbb{E}\left[A_i \cdot \sum_t \log \pi_\theta(a_t|s_t)\right] + \beta \text{KL}(\pi_\theta \,\|\, \pi_{\text{ref}})$

  依次解释数学符号含义：

  $\sum_{\text{t}} \log \pi_\theta(a_t|s_t)$

  $\sum_{\text{t}}$:对时间步进行求和，因为模型生成内容是一步一步来的，所以要把每一步的情况都考虑进去

  $\log \pi_\theta(a_t|s_t)$:π_θ是模型的策略（模型生成动作a_t的规则），也就是对策略在当前状态s_t下采取动作a_t的概率取对数，为了方便计算和优化

  β：一个系数，控制后面KL散度项的影响力大小，决定其对损失的贡献程度

  $KL(\pi_\theta||\pi_{\text{ref}})$:用来衡量两个策略 π_θ（当前模型的策略）和 π_{ref}（参考策略，比如 SFT 权重对应的策略）之间的差异。KL 散度越大，说明两个策略差别越大；越小则越相似。加上这一项是为了让模型的策略不要偏离参考策略太多。

- **奖励注入点**：奖励 r_i 用于构造优势 A_i，**乘在**当前策略的对数似然上；KL 项把模型拉回参考策略（如 SFT 权重）。

3. dsr1是咋训练出来的?dsr1的奖励是这样的?在哪个阶段?

4. ppo的loss函数?奖励是加在哪的?

5. 过拟合欠拟合，大参数模型会过拟合还是欠拟合?原因?

6.反向传播的时候梯度是如何一层一层计算的?

7.lora微调的层数，每个层数的参数减到多少了?

8.了解k-means，svm，xgboost嘛?

9.deepspeed三阶段?

10.模型为啥要量化部署?一般咋量化部署?

11.bf16和fp16的区别?

12.大模型之后咋发展?

13.tob大模型应用场景熟悉吗?

下面给你一份“面试速答卡”，按题号逐条作答。每条都先给“一句话要点”，再给 2–5 个关键点/公式，便于你按需展开。

---

2. 为啥先 SFT 再 GRPO?
要点：先让模型“会说”，再让它“说得对”。

- **SFT**（监督微调）：用高质量指令/示例让模型学到**格式、基本对齐与可用性**，稳定起步、降低 RL 难度。
- **GRPO**：在会说的前提上，用奖励去**优化目标行为**（正确率、可执行性、遵从规范等），避免 RL 从零噪声起步。
- 先 SFT 可显著**提高采样质量**，让 RL 的探索空间更小、收敛更稳，减少崩坏/跑偏。

---

3. GRPO 的 loss 函数？奖励加在哪?
要点：组内标准化优势 * 对数似然 + KL 约束，无价值网络。

- 典型做法：对同一 prompt 采样 K 个回答，记奖励 r_i。
- **组内基线/标准化优势**：$A_i = \frac{r_i - \bar{r}}{\sigma_r + \epsilon}$。
- **目标**（无 critic 的 PPO 变体）：

  $\mathcal{L}_{\text{GRPO}} = -\mathbb{E}\Big[A_i \cdot \sum_t \log \pi_\theta(a_t|s_t)\Big] + \beta\, \text{KL}(\pi_\theta \,\|\, \pi_{\text{ref}})$
- **奖励注入点**：奖励 r_i 用于构造优势 A_i，**乘在**当前策略的对数似然上；KL 项把模型拉回参考策略（如 SFT 权重）。

---

4. DSR1 是咋训练出来的？奖励怎样、在哪个阶段用?
要点：先 RL 得“会推理”的老师，再蒸馏成学生；奖励只在 RL 阶段用。

- 常见流水线（以 R1 系列为代表）：
  1. **RL-from-scratch / RL-on-top（GRPO/PPO）**：用**结果奖励**（如单测/Verifier 通过=1/0、数学答案正确）+ **过程/格式奖励**（思考框、结构化步骤、长度约束）得到高质量推理轨迹；
  2. **蒸馏 SFT（Distill）**：把 RL 产生的高质 CoT 作为监督数据训练学生（常被称作 *R1-Distill/DSR1*）；
  3. 视业务再做**指令补齐/安全对齐**或轻量 RL 打磨。
- **奖励在哪用**：仅在**RL 阶段**参与优化（构造优势），**蒸馏阶段不使用奖励**，是标准 NLL。

---

5. PPO 的 loss？奖励加在哪?
要点：剪切比率 * 优势 + 值函数 + 熵正则。

- 比率：$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$；
- **主项（剪切）**：
  $\mathcal{L}^{\text{clip}} = \mathbb{E}\big[\min(r_t A_t,\ \text{clip}(r_t, 1-\epsilon, 1+\epsilon)A_t)\big]$
- **总损失**：$-\mathcal{L}^{\text{clip}} + c_v \cdot \text{V-loss} - c_H \cdot \text{Entropy}$。
- **奖励注入点**：奖励进入**回报/优势** A_t 的计算（如 GAE），从而影响主项；有时再加 **KL 惩罚**。

---

6. 过拟合/欠拟合，大参数更容易哪种？为什么?
要点：大模型**更易过拟合**，但配好正则与数据也可能更好泛化。

- **过拟合**：训练高、测试低，原因：模型容量>数据信息量、噪声被记忆。
- **欠拟合**：训练/测试都低，模型表达力不足或优化不到位。
- 大参数更易过拟合，但通过**大数据、正则（Dropout/WD/早停）、数据增强、预训练+微调**可缓解，常见"**双降**"现象。

---

7. 反向传播梯度如何一层层算?
要点：链式法则 + 层局部梯度。

- 前向：$z_l = W_l x_{l-1} + b_l,\ x_l = f(z_l)$。
- 误差递推：$\delta_l = \left(W_{l+1}^\top \delta_{l+1}\right) \odot f'(z_l)$，顶层从 $\partial L/\partial x_L$ 起。
- 参数梯度：$\frac{\partial L}{\partial W_l} = \delta_l\, x_{l-1}^\top,\ \frac{\partial L}{\partial b_l} = \delta_l$。
- CNN/RNN 同理，用对应算子的雅可比/卷积转置实现。

---

8. LoRA 微调哪些层？参数减少到多少?
要点：对大矩阵做**低秩增量** $\Delta W = BA$（r≪d,k），大幅减参。

- 常插在 **Attention 的 Q/K/V/O**，有时加 **FFN 的上/下投影**。
- 参量：从 $d \times k$ 变为 $r(d+k)$（再加少量缩放/偏置），**显著下降**。
- 例：4096×4096，全参≈16.8M；LoRA **r=16** ⇒ $16(4096+4096) = 131k$（≈**0.8%**）。
- 典型 r=8–64，几乎**1–2%** 训练参数即可达成接近全参效果。

---

9. 了解 K-Means / SVM / XGBoost 吗?
要点：各自适用面与优势劣势。

- **K-Means**：无监督聚类，迭代分配–更新中心；快、易实现；非凸、对初始/尺度敏感，偏球形簇。
- **SVM**：最大间隔分类，核技巧处理非线性；小中等样本、边界清晰时强；对大规模需核近似/线性 SVM。
- **XGBoost**：梯度提升树，处理非线性/缺失值/类别特征强，特征工程友好；需调参，易过拟合要正则。

---

10. DeepSpeed 三阶段（ZeRO）?
要点：分片优化器→再分片梯度→再分片参数。

- **Stage-1**：优化器状态分片（如 Adam 的 m/v）到各 GPU。
- **Stage-2**：再把**梯度**也分片。
- **Stage-3**：连**模型参数**也分片（加载/广播按需），可 **CPU/NVMe offload**；配合 **流水/张量并行、激活检查点** 极致扩展。

---

11. 为啥要量化部署？怎么量化?
要点：省显存、提吞吐、降时延、降成本。

- **动机**：KV 缓存/权重压缩，提升 QPS 与边缘部署可行性。
- **方法**：
  - **PTQ**（后训练量化）：GPTQ、AWQ、SmoothQuant、LLM.int8/8-bit-move 等；
  - **QAT**（量化感知训练）：W8A8/W4A8，训练期加入假量化；
  - **KV-Cache 量化**：INT8/FP8/INT4；**对称/非对称**、**逐通道/逐组**缩放。
- **实践**：TensorRT-LLM / vLLM / llama.cpp，A/B 以 **困惑度/准确率/延迟/QPS** 验证回归。

---

12. BF16 和 FP16 的区别?
要点：同内存，不同指数位；BF16 更稳。

- **FP16**：1/5/10（符号/指数/尾数），**动态范围小**，易溢出/下溢；
- **BF16**：1/8/7，与 FP32 同指数宽度，**范围大**、训练更稳；
- A100/TPU 原生 BF16，吞吐/显存与 FP16 近似，**优选 BF16** 做训练；推理看硬件内核支持。

---

13. 会 SQL 吗?
要点：会，用于离线分析/线上探查/指标核对。

- 去重汇总：

```sql
SELECT user_id, COUNT(*) AS n, COUNT(DISTINCT session_id) AS ds
FROM logs
WHERE dt BETWEEN '2025-09-01' AND '2025-09-07'
GROUP BY user_id;
```

- 窗口与去重：

```sql
SELECT *
FROM (
  SELECT *, ROW_NUMBER() OVER(PARTITION BY uid ORDER BY ts DESC) AS rn
  FROM events
) t WHERE rn=1;
```

---

14. 大模型之后怎么发展?
要点：**多模态、工具化、检索化、可控与高效化**。

- **多模态/Agent**：视觉/语音/动作，流程编排与工具调用。
- **RAG-native** 与 **可验证生成**（verifier/程序化奖励）。
- **高效推理**：蒸馏、组合与**推理时扩展**（speculative/先验树搜索）。
- **安全与治理**：可解释、审计、合规；**行业小模型**与**端侧/混合云**协同。

---

15. ToB 大模型应用场景熟悉吗?
要点：围绕"**提效、降本、控风控**"落地并量化 KPI。

- **知识问答/RAG**（客服、售后、销售赋能）：**首响时延、一次解决率、每单成本**。
- **文档/合同处理**（抽取、比对、合规审阅）：**周转时长、误报漏报率**。
- **流程自动化**（工单路由、审批流、报表生成）：**处理吞吐/人效提升**。
- **代码/数据助理**（单测生成、SQL 生成/审查）：**缺陷率、交付周期**。
- **风控合规**（敏感信息检测、政策对齐）：**召回/误报、审单率**。
- 强调**隐私与治理**（最小化收集、可追溯、PDPO/GDPR 合规）、**可观测**与**持续评测**。