大模型原理与应用

第 8 课：对齐与 RL 后训练

理解模型如何从“会回答”走向“更符合人类偏好”

这一章看什么

- RLHF、DPO、GRPO 分别在解决什么问题
- 合成数据、奖励设计和 failure case 如何改变训练策略
- 为什么 alignment 最终会走向一个更大的社会技术问题

本章主线

1. 先明确为什么 SFT 之后还会继续走向 alignment
2. 再看三条主流路线：RLHF、DPO、GRPO
3. 最后回到更大的问题：奖励、数据和社会目标到底如何对齐

先给这一章一个定位

如果说预训练回答的是：

- 模型怎样学会语言与知识

SFT / instruction tuning 回答的是：

- 模型怎样学会按要求说话

那么这一章回答的就是：

- 当模型已经“会回答”以后，怎样让它“更符合人类偏好、推理方式和使用预期”？

因此这里讨论的重点不再是：

- base model 的能力来源
- 指令格式化和 adapter 工程细节

而是：

- 偏好数据
- 奖励设计

RLHF：最经典的对齐路线

RLHF 的基本链路是：

1. 先做 supervised fine-tuning
2. 再收集人类偏好数据
3. 训练 reward model
4. 用 RL 优化语言模型策略

它的核心想法是：

- 不只模仿答案
- 而是学习“人更喜欢什么样的答案”

RLHF 真正多出来了什么

和普通 SFT 相比，RLHF 多出来的不是一个小技巧， 而是整套偏好学习链路：

- 人类不再只写“标准答案”
- 人类开始比较“两个答案谁更好”
- 模型不再只拟合文本
- 模型开始拟合偏好信号

所以它优化的目标已经从：

- “像训练数据”

变成：

- “更符合人类选择”

RLHF 的四段式工作流

可以把 RLHF 粗略记成四段：

1. `SFT`：先得到一个基本可用的 assistant
2. `Preference Data`：收集排序或打分数据
3. `Reward Model`：学一个偏好代理
4. `RL`：让 policy 朝高奖励方向继续走

这四段里，真正最贵的常常不是最后的 RL， 而是中间那层偏好数据。

偏好数据到底长什么样

不是简单的“问题 - 标准答案”对， 而更像：

- 给定同一个 prompt
- 模型生成多个候选回答
- 让人类去排序、打分或二选一

这类数据的价值在于：

- 它更接近“整体体验”
- 而不是逐 token 的局部监督

奖励模型在 RLHF 里扮演什么角色

reward model 的任务不是生成答案， 而是评估答案：

- 哪个更 helpful
- 哪个更 safe
- 哪个整体更像人类会选的输出

因此 reward model 本质上是：

- 人类偏好的代理

而不是“真实价值本身”。

为什么会从 SFT 走到 RLHF

因为单纯 instruction tuning 往往有这些问题：

- 标注数据昂贵
- 监督信号有限
- 只能逐 token 模仿
- 难以直接表达整体偏好

而 RLHF 的优势是：

- 可以把稀疏但更高层的偏好反馈接进训练

RLHF 的代价也非常明显

- 流程长
- 模型多
- 数据贵
- 训练不稳定

并且还有一个根本问题：

- reward model 只是代理目标

一旦这个代理学偏了， 后面的 RL 只会更努力地把模型推向错误方向。

DPO：把 RLHF 流程再简化

RLHF 很有效，但也复杂：

- 要训练多个模型
- 要做在线采样
- 算力和工程成本都更高

DPO 的核心吸引力在于：

- 尝试用更直接的目标函数
- 在 preference data 上直接优化 policy

所以它常被看作“简化版 RLHF”。

DPO 的核心直觉到底是什么

这组课件强调的直觉非常重要：

- language model 本身“偷偷包含了 reward 信息”
- 如果我们有 preferred / dispreferred response
- 就可以直接优化它们的相对概率

换句话说，DPO 的重点不是：

- 再显式训练一个 reward model

而是：

- 直接把偏好约束写进 policy 优化目标

DPO 为什么会受欢迎

因为它在工程上明显更轻：

- 不需要完整 RLHF 的那套在线 RL 流程
- 不需要显式 value model
- 数据形式仍然可以是 preference pair

这让它特别适合：

- 开源社区
- 资源受限团队
- 想快速做 preference tuning 的场景

但 DPO 不是“白送的简化”

它仍然有自己的前提：

- 你得有高质量 preference data
- 你仍然依赖 base policy 作为参照
- 你仍然在做“偏好代理优化”

也就是说，DPO 只是把 RLHF 的实现负担降下来了， 并没有消灭 alignment 的本质难题。

先别急着看 GRPO：它是怎么长出来的

`7-alignment_grpo.pptx` 的核心提醒是：

- GRPO 不是凭空冒出来的新技巧
- 它站在一整条 RL 算法链路上

这条线大致是：

1. REINFORCE
2. baseline / value function / advantage
3. TRPO
4. PPO
5. GRPO

最朴素的起点：Policy Gradient / REINFORCE

最基础的想法很直接：

- 采样一条轨迹
- 看最终 reward
- 增大奖励高的动作概率
- 降低奖励低的动作概率

这就是最经典的 policy gradient 直觉。

但问题也马上出现：

- 方差很大
- 更新不稳定
- 很容易出现 distribution drift

为什么要引入 baseline

REINFORCE 的关键问题是：

- 同一个动作到底是“真的好”，还是“只是这次运气好”？

所以 RL 文献里会引入 baseline：

- baseline 不依赖具体动作
- 但尽量和未来回报相关

直觉上，它在回答：

- 这个动作比“平均水平”到底好多少

从 baseline 走向 advantage

进一步地，我们不一定关心动作的绝对回报， 更关心它相对于平均动作有多好。

于是就得到 advantage 的思路：

- 不问“奖励高不高”
- 而问“比通常动作高多少”

这样做的好处是：

- 梯度方差更低
- 学习信号更稳定

TRPO：先给更新加上“护栏”

即便有了 advantage， policy gradient 仍然可能一步走太猛。

TRPO 的重要想法是：

- 不让新 policy 离旧 policy 太远
- 用 KL 约束定义 trust region

本质上，它在平衡两件事：

- plasticity：继续学新东西
- stability：不要把模型一下子改坏

PPO：把 TRPO 做得更可用

PPO 是 RLHF 里最常见的 RL 算法之一。
它保留了“不要偏离太多”的思路， 但换了一个更工程化的做法：

- 用 ratio 比较新旧 policy
- 用 clipping 防止更新过大

相比 TRPO，它更容易实现， 也更适合大规模实验。

PPO 在 RLHF 里为什么常见

因为它正好贴合 RLHF 的需求：

- 有 policy model
- 有 reference policy
- 有 reward model
- 还有 value model 来估计 advantage

但这也带来一个很现实的问题：

- 内存和训练复杂度都很高

在 LLM 场景里，这一点尤其痛。

PPO 的代价：为什么很多人开始嫌它重

`7-alignment_grpo.pptx` 里专门强调：

- PPO 在 LLM 场景里常常混着 4 个模型
- reward、value、policy、reference 都要参与

这意味着：

- 显存压力大
- 实现细节多
- 超参数很敏感
- 训练稳定性也不总是理想

所以问题变成了：

- 如果我们能更便宜地估计 advantage，能不能把 value model 去掉?

GRPO：就是在这个问题下出现的

GRPO 的关键出发点不是“重新发明 RL”， 而是更具体的工程目标：

- 保留 PPO 这类方法里“相对好坏比较”的优点
- 但去掉 value model，降低内存和复杂度

这也是它和 PPO 的真正关系：

- 不是替代整个 RL 思想
- 而是替代其中一部分昂贵结构

GRPO 的核心直觉是什么

它不再单独训练一个 value model 来估计 advantage， 而是：

- 对同一个 prompt 采样多条 rollout
- 看这一组回答里谁更好、谁更差
- 用组内相对奖励来估计“优势”

也就是说：

- advantage 不再来自一个单独的 critic 网络
- 而来自同组样本之间的相对比较

为什么“group relative”这个名字很关键

因为它强调的不是绝对 reward， 而是组内相对位置：

- 比平均值高多少
- 比其他回答强多少
- 在这一组样本里排在哪里

通常还会做标准化处理， 让训练信号更稳定。

所以它本质上是一种：

- 相对奖励驱动的 advantage 估计

GRPO 和 PPO 到底差在哪里

可以粗略地记成：

- PPO：常靠 value model 估计 advantage
- GRPO：常靠组内 rollout 的相对 reward 估计 advantage

带来的直接后果是：

- PPO 更重
- GRPO 更省内存
- GRPO 特别适合“大模型 + 多采样 + 推理任务”的场景

这正是它后来被 DeepSeekMath / R1 路线采用的重要原因。

为什么 GRPO 特别适合 reasoning

因为推理题往往天然允许：

- 同一个问题采样多个回答
- 对这些回答做规则或结果比较
- 选出“相对更好”的推理过程

所以在数学、代码、可验证任务里， GRPO 的组内比较机制特别顺手。

这和开放式偏好打分类任务不完全一样。

从 GRPO 再往后：RLVR 与 rule-based reward

课件里还提到一个重要方向：

- 用 rule-based reward
- 甚至进一步去掉实值 reward model

例如：

- 可验证答案
- 格式约束
- 数学正确性

这条线常被称为：

- RL with Verifiable Feedback

它说明 reasoning 模型训练正在继续“去黑盒化”。

为什么后来还会出现各种 GRPO 变体

课件最后提到几类变体：

- remove KL
- decouple clipping
- over-sample and filter
- RLOO

这背后的共同动机其实很简单：

- 原始 GRPO 也不是终点
- 大家还在继续优化稳定性、效率和训练信号质量

所以我们现在看到的是一整片快速演化的家族， 而不是一个已经完全定型的算法。

现在再回来看 GRPO 的位置

把整条线连起来看：

- REINFORCE 给出最原始的 policy gradient
- baseline / advantage 解决高方差
- TRPO / PPO 给更新加稳定机制
- GRPO 进一步针对 LLM 后训练降低成本

所以 GRPO 最好理解成：

- “面向大模型后训练场景的轻量化 PPO 亲属”

而不是孤立的新名词。

GRPO：推理模型时代的后训练路线

这组材料里，GRPO 代表的是另一种强化方向：

- 特别适合 reasoning tasks
- 常出现在 DeepSeekMath、DeepSeek-R1 等脉络里
- 用 RL 继续强化模型的多步推理能力

这里的关注点不只是“偏好”， 而是“推理质量与推理结构”。

Synthetic Data：对齐不一定全靠人标

材料专门补了一个重要方向：

- 用模型自己生成 instruction / response 数据
- 通过 Self-Instruct 等方式扩展 alignment data

它的价值是：

- 减少纯人工标注依赖
- 扩展任务多样性
- 提高 generalist model 的覆盖面

但它也会放大已有偏差。

Synthetic Data 的典型路线：Self-Instruct

这组材料里给出的 Self-Instruct 流程很清楚：

1. 先让人类写少量 seed tasks
2. 放进 task pool
3. 让 LLM 扩展出更多新任务
4. 再让 LLM 为这些任务生成答案
5. 过滤掉重复、低质量或高重叠样本
6. 把筛过的数据继续加回池子里

这是一种“让模型帮助制造对齐数据”的思路。

为什么 synthetic data 很诱人

- 人类标注太贵
- 高质量多样化指令数据更贵
- 通用助手模型又特别依赖任务多样性

所以 synthetic data 的吸引力在于：

- 用少量人工种子，换出大规模任务扩展

这也是很多 open-source instruct model 都采用它的原因。

Synthetic Data 的风险是什么

材料也讲得很直白：

- 它会放大已有模型偏差
- 它容易被 prompt 边界限制
- 当数据很稀薄时，模型生成的数据也会变差

因此 synthetic data 不是“替代人类”， 更像是：

- 在人类设计边界之内做规模放大

对齐为什么会失败

材料里强调了几类典型 failure：

- reward hacking
- reward over-optimization
- length bias
- 过度道歉、过度冗长、过度拒答

这说明：

- 奖励函数不是“真目标”
- 它只是一个代理

一旦代理错了，模型就会学会“钻空子”。

Reward Hacking：最典型的失败方式

reward hacking 的核心不是模型“坏”， 而是模型非常认真地优化了一个不完美指标。

例如：

- 表面上得分更高
- 但实际行为更偏离真实目标

所以 reward hacking 的真正含义是：

- 模型学会了优化评分器，而不是优化我们真正想要的东西

Reward Over-Optimization：越训越怪

课件里举的现象很典型：

- 过度冗长
- 过度道歉
- 过度犹豫
- 过度拒答

这说明模型可能在做一件事：

- 拼命迎合代理 reward 的弱点

而不是变成“更自然的助手”。

Length Bias：为什么“更长”常常被误认为“更好”

材料特别提醒了一个常见陷阱：

- 更长的回答
- 更多独特 token

常常更容易被偏好系统认为“更好”。

这会导致模型逐渐学会：

- 把回答拉长
- 用更多铺垫和包装

即使信息价值并没有同步提升。

Human inconsistency：问题不只在模型

alignment 的困难还来自人本身：

- 标注者之间可能标准不同
- 同一个人前后也可能不一致
- 偏好往往依赖情境

所以“人类偏好数据”并不是一个干净、稳定、唯一的真值源。

这也是 alignment 总会带着不确定性的原因。

Bigger Picture：对齐不只是模型训练

材料最后把问题拉高到更大的层面：

- 人类社会本身就在用各种机制做 alignment
- 市场、法律、规范、制度，都是对齐机制
- AI alignment 只是这个大问题的一个新技术版本

因此，alignment 不能只看成一个 loss function 设计题。

从 broader picture 该怎么理解 alignment

如果把问题放大来看：

- 规范是一种 alignment 机制
- 市场是一种 alignment 机制
- 法律和制度也是 alignment 机制

那么 AI alignment 的特殊之处就在于：

- 我们试图把这种社会层面的目标压缩成数据、奖励和训练流程

而这个压缩过程天然会丢失信息。

先给一个总判断

- RLHF 让模型开始学习人类偏好
- DPO 让 preference optimization 更简单
- GRPO 让推理模型后训练更具可操作性
- synthetic data 让对齐数据来源更丰富
- failure cases 提醒我们：奖励设计永远不是中性的

这几条路线共同构成了现代 alignment 的基本图景。

这章你该带走什么

- 对齐是预训练和 SFT 之后的关键阶段
- RLHF、DPO、GRPO 不是彼此孤立，而是不同复杂度与目标下的路线选择
- 合成数据、奖励设计和 failure analysis 同样是 alignment 的核心组成部分

本章收束

- 对应材料：`local/lesson-3/rl/*.pptx`
- 课程位置：独立成章，聚焦对齐、偏好学习与 RL 后训练