目录

1 CS 285 深度学习强化学习—第 14 讲详细讲义

原文来源: lec-14.pdf **生成日期**: 2026-05-09 **总页数**: 51 页 | **总章节**: 5 章 **语言**: 简体中文 **课程名称**: RL with Sequences & LLMs（序列决策与大语言模型的强化学习）**讲师**: Sergey Levine, UC Berkeley **说明**: 本讲义基于课程幻灯片深度扩写而成, 每页幻灯片均扩展为详细讲解, 补充了背景知识、公式推导、实例分析和关键要点总结。

1.1 目录

1. 第 1 章: 奖励学习与逆强化学习回顾
2. 第 2 章: RL 与大语言模型基础
3. 第 3 章: LLM 强化学习算法
4. 第 4 章: 偏好与验证器
5. 第 5 章: 部分可观测性与 POMDP

1.2 第 1 章: 奖励学习与逆强化学习回顾

涵盖范围: Slide 2-12 | **核心主题**: 回顾逆强化学习（IRL）的核心框架——从学习最优性变量、配分函数到与 GAN 的等价关系, 为后续 LLM 中的奖励学习奠定基础

1.2.1 1.1 为什么学习奖励函数（Slide 3）

概念详解

Slide 3 从强化学习本身的视角重新提出了一个根本问题：“what is the reward?”——奖励函数到底是什么？在许多现实任务中，设计合理的奖励函数（reward engineering）恰恰是最困难的环节。自动驾驶的奖励如何定义？是一个复合函数——包含安全距离、舒适度、通行效率、交通规则遵守等几十个维度的加权组合？还是由数据驱动来自动学习？这个问题在 LLM 时代变得尤为突出：我们希望语言模型生成的文本”有帮助、无害、真实”，但如何将这些抽象的人类价值量化为一个可优化的标量奖励信号？

这正是逆强化学习（Inverse Reinforcement Learning, IRL）的核心动机：与其手工设计奖励函数，不如从专家演示中**自动推断**奖励函数。在”控制即推断”的框架下（第 12-13 讲），我们建立了用概率推断来解决控制问题的理论体系。IRL 是这个框架的自然反转——如果正向问题是”给定奖励，推断最优策略”（控制即推断），那么逆向问题就是”给定（近似）最优的行为数据，推断奖励函数”。

深度剖析

学习奖励函数相比于直接模仿策略有一个关键优势：**奖励函数是比策略更紧凑、更可迁移的任务表示**。一个策略 $\pi(a|s)$ 只在特定的环境动态和状态空间中有效——换一个环境（如从仿真到真实世界），策略就必须重新训练。但奖励函数 $r(s,a)$ 编码的是任务的**本质目标**（如”安全地到达目的地”），它在不同环境中都可以被复用。

这个洞察在 LLM 的训练流程中得到了完美的体现。在 RLHF（RL from Human Feedback）中，我们先从人类偏好数据中学习一个奖励模型（reward model），然后用这个奖励模型来微调语言模型。这个奖励模型扮演的角色与传统 IRL 中学到的奖励函数完全相同——它是从数据中推断出的”什么是好的文本”的量化标准。

关键要点

- 手工设计奖励函数在很多任务中极其困难（自动驾驶、对话系统等）
- IRL 从专家演示中自动推断奖励函数，避免了手工 reward engineering
- 奖励函数比策略更可迁移——编码任务本质目标而非特定环境中的动作序列
- IRL 的框架直接延伸到了 LLM 的 RLHF 训练流程中

过渡衔接：IRL 的具体实现依赖于”最优性变量”的概率建模——这正是 Slide 4 的核心。

1.2.2 1.2 学习最优性变量（Slide 4-6）

概念详解

Slide 4 回顾了 IRL 的核心技术组件：将奖励函数的学习表述为**最优性变量** $p(\mathcal{O}_t|s_t,a_t)$ 的参数估计问题。回顾”控制即推断”框架中的定义：

$$p(\mathcal{O}_t = 1|s_t, a_t) = \exp(r_\psi(s_t, a_t))$$

其中 r_ψ 是以 ψ 为参数的奖励函数（通常是一个神经网络）。IRL 的目标是找到奖励参数 ψ，使得专家演示轨迹在 soft-optimal 后验分布下具有最大的似然：

$$\max_\psi \frac{1}{|\mathcal{D}_{\text{demo}}|} \sum_{\tau \in \mathcal{D}_{\text{demo}}} \left[\sum_t r_\psi(s_t, a_t) \right] - \log Z(\psi)$$

其中 $Z(\psi) = \int p(\tau) \exp(\sum_t r_\psi(s_t, a_t)) d\tau$ 是配分函数（partition function），Slide 5 专门标注了它的核心地位。

深度剖析

配分函数 $Z(\psi)$ 是 IRL 的计算瓶颈。它在**所有可能的轨迹**空间上进行积分（或求和），在连续高维空间中这是不可精确计算的。梯度的形式揭示了 IRL 的对抗性本质：

$$\nabla_\psi \mathcal{L} = \mathbb{E}_{\tau \sim \text{demo}} \left[\sum_t \nabla_\psi r_\psi \right] - \mathbb{E}_{\tau \sim p(\tau|\mathcal{O}_{1:T}, \psi)} \left[\sum_t \nabla_\psi r_\psi \right]$$

第一项在专家轨迹上**增加**奖励，第二项在当前 soft-optimal 策略生成的轨迹上**降低**奖励。两者的博弈驱动着奖励函数的学习。Slide 6 专门讨论了这一梯度的估计方法。

实践中处理配分函数的主要策略包括：- **直接采样**：在当前的 soft-optimal 策略下采样轨迹（但需要在内部循环中运行 RL）- **重要性采样**（Slide 8）：用旧策略的样本来估计当前期望，利用重要性权重重新加权 - **对抗方法**：将配分函数估计问题转化为 GAN 式的判别器训练问题

关键要点

- 最优性变量 $p(\mathcal{O}_t = 1|s_t, a_t) = \exp(r_\psi(s_t, a_t))$ 是 IRL 的核心建模元素
- IRL 目标 = 最大化专家似然 = 专家奖励期望 $-\log Z(\psi)$
- 配分函数 $Z(\psi)$ 的计算是核心瓶颈：需要遍历所有轨迹空间
- 梯度自然地呈现为”专家 vs soft-optimal 策略”的对抗形式

过渡衔接：Slide 7-9 展示了 IRL 训练中的实际效率改进和与 GAN 的惊人联系。

1.2.3 1.3 IRL 与 GAN 的等价性（Slide 7-12）

概念详解

Slide 7-12 展开了 IRL 领域最具洞察力的发现之一：IRL 的训练过程与生成对抗网络（GAN, Goodfellow et al. 2014）在数学上等价。

Slide 9 指出了引导式代价学习（Guided Cost Learning）的交替优化结构——奖励函数试图区分专家和策略样本，策略试图逼近专家——与 GAN 的”判别器 vs 生成器”博弈完全一致。Slide 10 回顾了 GAN 在图像生成等领域的革命性应用，Slide 11-12 则将这一联系精确化为数学上的对应关系。

在 GAN 形式的 IRL 中：- **判别器** $D_\psi(s,a)$ 对应 **奖励函数**：$D_\psi(s,a) = \frac{\exp(r_\psi(s,a))}{\exp(r_\psi(s,a))+\pi_\theta(a|s)}$ - **生成器** π_θ 对应 **策略** - Min-max 博弈：$\min_\theta \max_\psi \mathbb{E}_{\text{demo}}[\log D] + \mathbb{E}_{\pi_\theta}[\log(1-D)]$

Slide 11 讨论了一个重要的简化：**能不能直接用一个普通的判别器来做模仿学习?** 这就是生成对抗模仿学习（GAIL, Ho & Ermon 2016）。

深度剖析

IRL 与 GAN 等价的发现不仅仅是理论上的优雅，它带来了实际的算法简化。传统的 IRL 需要在每一步奖励更新后解决一个完整的 RL 问题（或至少估计配分函数），而 GAN 形式的模仿学习将这个内循环替换为标准的判别器训练——这是一个更稳定、更高效的过程。

然而，这种简化也有代价。Slide 11 明确指出：GAIL 的判别器在收敛时”一无所知”（knows nothing at convergence）——当策略完全匹配专家分布后，判别器输出恒为 0.5，学到的”奖励函数”失去了泛化能力。相比之下，传统 IRL 中学到的奖励函数在策略收敛后仍然编码了关于任务结构的有用信息，可以迁移到新环境。

Slide 12 用一个并列对比总结了这一关系：“实际上是一回事!”（actually the same thing!）——IRL（学习奖励函数）和 GAIL（学习分类器）在数学上等价，差异更多体现在实践细节和侧重点上：IRL 更重视奖励的泛化性，GAIL 更重视训练的简洁性。

关键要点

- IRL 的交替优化结构在数学上与 GAN 的 min-max 博弈精确对应
- 判别器 $D(s,a)$ 奖励函数 $r(s,a)$：通过 $\exp(r)$ 变换互相转换
- GAIL 提供了更简单的实现：直接用判别器输出作为策略的奖励信号
- 但 GAIL 收敛后判别器失效，学到的”奖励”无法泛化——这是与 IRL 的核心取舍

过渡衔接：至此我们回顾了 IRL 的核心框架。现在我们转向本讲的新内容——将这些概念应用于大语言模型的强化学习训练。

1.3 第 2 章：RL 与大语言模型基础

涵盖范围：Slide 13-24 | **核心主题**：理解大语言模型（LLM）的基本架构和训练流程，以及如何将 RL 问题形式化应用于文本生成

1.3.1 2.1 语言模型与 Transformer 架构（Slide 13-14）

概念详解

Slide 13-14 引入了大语言模型（Large Language Model, LLM）的基本概念。现代 LLM 几乎都基于 Transformer 架构（Vaswani et al., 2017），其核心组件包括：

- **掩码自注意力（Masked Self-Attention）**：每个 token 只能”看到”它之前的 token，实现对序列的条件概率建模。这是 Transformer 区别于双向模型（如 BERT）的关键——它确保了语言模型的自回归（autoregressive）性质。
- **逐位置前馈网络（Position-wise Feedforward Network）**：对每个位置的表示进行非线性变换，增强模型的表达能力
- **逐位置 Softmax 输出**：在词表上产生下一个 token 的概率分布

Transformer 通过堆叠多个这样的层（通常数十到上百层）来逐步构建越来越抽象的文本表示。Slide 14 上的动画展示了信息如何在 Transformer 中逐层流动。

语言模型的核心训练目标是**下一个 token 预测**（next token prediction）：

$$p(x_1, x_2, \dots, x_T) = \prod_{t=1}^{T} p(x_t | x_{1:t-1})$$

这个看似简单的目标具有惊人的力量——要准确预测互联网文本中的下一个词，模型必须隐式地学习语法、事实知识、推理能力以及大量的”世界知识”。

深度剖析

为什么”下一个 token 预测”这个看似简单的目标能够产生如此强大的能力？答案在于**规模**和**数据的多样性**。在数万亿 token 的网络文本上进行训练时，为了准确预测下一个词，模型被迫学习：

- **语法和拼写**：显而易见的基础能力
- **事实知识**：要补全”法国的首都是”这个序列，模型必须”知道”答案是巴黎
- **推理能力**：要补全一个数学推导或代码片段，模型必须隐式地执行计算步骤
- **上下文理解**：要维持长对话的一致性，模型必须追踪对话历史和隐含意图

Slide 14 上还有一个重要的补充说明: “Language models are often trained with supervised learning” —— 强调语言模型的预训练 (pre-training) 阶段使用的是监督学习 (或无监督的 self-supervision), 而非强化学习。RL 的介入是在后训练（post-training）阶段。

关键要点

- Transformer 的核心: 掩码自注意力 + 逐位置前馈网络 + Softmax 输出
- 训练目标 = 下一个 token 预测: $p(x_t|x_{1:t-1})$
- “下一个 token 预测”迫使模型隐式地学习语法、事实、推理和世界知识
- 预训练使用监督/自监督学习, RL 在后训练阶段才介入

> **过渡衔接**: LLM 的训练分为两个阶段——预训练和后训练。理解这两个阶段的区别对于理解 RL 在 LLM 中的角色至关重要。

1.3.2 2.2 LLM 训练的全貌（Slide 15-17）

概念详解

Slide 15 展示了 LLM 训练的完整流程, 分为两个阶段:

预训练 (Pre-training): - 爬取海量的多样化数据（网页、书籍、代码等）- 使用下一个 token 预测目标进行大规模训练 - 这一阶段赋予了模型”世界知识”和基础语言能力 - 计算代价极高: 需要数千 GPU 运行数月

后训练 (Post-training): - 高质量 SFT（Supervised Fine-Tuning）数据集: 人工标注的指令-回答对 - RL 微调（如 RLHF): 使用强化学习根据人类偏好进一步优化模型

Slide 16 提出了一个关键的区分: 预训练的 LLM “不是智能体、助手或任何类似的东西——它们只是文本补全引擎”（they are “text completion” engines)。预训练模型会忠实地补全任何给定的文本前缀——无论这个前缀是关于帮助用户回答问题的, 还是关于生成有害内容的。后训练的作用是告诉模型**如何使用**它在预训练中获得的知识。

深度剖析

Slide 17 讨论了最基本的后训练方法: **指令微调** (Instruction Tuning, Chung et al. 2022)。在指令微调中, 我们收集大量的人工标注数据——对于每条指令（如”写一首关于春天的诗”), 标注员写出理想的回答。然后使用标准的监督学习（交叉熵损失）在这些数据上微调模型。

指令微调的局限性体现在多个方面: 1. **劳动密集**: 高质量的标注极其昂贵, 特别是对于需要专业知识的任务（如编程、数学、法律）2. **覆盖不足**: 不可能为所有可能的指令提供标注 3. **质量上限**: 模型只能学到标注员的水平, 无法超越 4. **偏好表达困难**: 对于”这个回答好还是那个回答好”这样的偏好判断, 比”写一个完美回答”容易得多

这些局限性正是 RL 在后训练中发挥作用的原因——RL 可以从**相对偏好**（而非绝对正确回答）中学习，只需要标注员比较两个回答哪个更好，而不需要写出完美的回答。

关键要点

- 预训练赋予 LLM 知识，后训练告诉 LLM 如何使用这些知识
- 预训练的 LLM 是”文本补全引擎”，不是助手或智能体
- 指令微调使用监督学习，受限于标注成本和质量上限
- RL 后训练的优势：可以从相对偏好中学习，降低标注难度

过渡衔接：RL 在后训练中有两种主要形式——从偏好中学习和从验证器中学习。Slide 18-19 分别介绍了这两种范式。

1.3.3 2.3 RL 后训练的两种范式（Slide 18-19）

概念详解

Slide 18 展示了从人类偏好中学习的方法——RLHF（Reinforcement Learning from Human Feedback）。核心流程是：（1）收集人类对模型多个输出的偏好比较数据；（2）训练一个奖励模型来预测这些偏好；（3）使用 RL（通常是 PPO）来优化语言模型，使其生成高奖励的文本。

Slide 19 展示了从验证器（verifier）中学习的方法。与从主观偏好中学习不同，验证器提供的是**客观的正确性信号**——例如，对于数学问题，答案是否正确；对于代码，测试用例是否通过。图中的引用（Marjanovic et al. ’ 25）表明这是一个较新的研究方向。核心思想是：“使用指示答案正确性的奖励信号，训练模型生成能导向正确答案的思考过程”。

深度剖析

这两种 RL 后训练范式代表了 AI 对齐研究中的两条主线：

偏好学习（RLHF）适用于**没有客观标准答案**的任务——如写作风格、对话礼貌性、创意生成等。奖励信号本质上是主观的：不同的人对”好回答”有不同的定义。RLHF 通过从大量偏好标注中学习一个共识奖励模型来处理这种主观性。

验证器学习适用于**有客观标准答案**的任务——如数学证明、代码生成、逻辑推理等。奖励信号是客观的：答案对就是对、错就是错。这种方法的一个关键优势是奖励信号可以**自动获取**（代码可以自动运行测试、数学题可以自动验算），大大降低了人工标注成本。其挑战在于：如何为部分正确的中间推理步骤提供适当的奖励/信用分配?

两种方法也可以结合使用：在同一个训练流程中同时使用偏好奖励（确保回答风格合适）和验证器奖励（确保回答事实正确）。

关键要点

- RLHF 从人类偏好比较中学习奖励模型 → 适用于主观任务
- 验证器方法从客观正确性信号中学习 → 适用于数学、代码等可自动评测的任务
- 偏好标注比”写出完美答案”容易得多 → 降低了后训练的标注成本
- 两种方法可以结合：偏好确保风格，验证器确保正确性

过渡衔接：有了高层框架，现在让我们看 RL 在 LLM 中的基本形式化——这构成了后续所有算法讨论的数学基础。

1.3.4 2.4 LLM 作为 RL 问题的基本形式化（Slide 20-24）

概念详解

Slide 20-21 展示了将 LLM 文本生成形式化为 RL 问题的两种等价视角。

基本形式化（Slide 20）：- **状态** s_t：当前的上下文/前缀（包括 prompt + 已生成的 token）- **动作** a_t：下一个 token（从词表中选一个）- **转移** $p(s_{t+1}|s_t, a_t)$：确定性的——将 token 追加到前缀 - **奖励** $r(s_T, a_T)$：仅在序列结束时给定（或使用中间奖励）- Prompt 是初始状态，模型的输出是动作序列

这构成了一个**单步 RL 问题**（在提示词之后，模型生成一个完整的回答序列）。Slide 20 上明确标注”Basic one step RL problem”——强调从 RL 的角度看，每个 prompt 是一个 episode，生成过程是一次决策序列。

等价视角（Slide 21）：将 token-by-token 的生成视为多步决策过程。这个视角在使用中间奖励（intermediate rewards）和值函数基线（value function baselines）时更有意义。与其只在序列末尾给予奖励，我们可以在每个生成步骤评估”进展如何”。

深度剖析

将 LLM 文本生成看作 RL 问题时，有几个独特的设计考量：

动作空间：词表大小 $|\mathcal{V}|$ 通常为 30,000-100,000+ token。这是一个**极大且离散**的动作空间。计算所有动作的 Q 值或 Softmax 是不现实的——每次只能采样少量动作。这解释了为什么在实际 LLM RL 中，我们通常使用策略梯度方法（如 PPO）而非值函数方法（如 Q-learning）。

奖励稀疏性：在标准的 LLM RL 设置中，奖励只在完整序列生成后才给出（sparse reward）。这意味着信用分配（credit assignment）是一个关键挑战——序列中的每个 token 对最终奖励的贡献需要被估计。

自回归结构：每个 token 的选择依赖于之前的所有 token（状态 s_t 就是整个前缀），这使得问题自然地符合 MDP 的结构——当前状态 s_t 包含了所有历史信息，马尔可夫性（通过将整个上下文编码为状态）被满足。

Slide 22-24 引入了策略梯度在 LLM 中的应用，包括 REINFORCE 估计器和重要性加权估计器（PPO），以及重复 K 次采样的实现细节。

关键要点

- LLM 生成 = 序列决策过程：状态 = 前缀，动作 = 下一个 token
- 动作空间巨大（词表 30k-100k+）→ 策略梯度优于 Q-learning
- 奖励通常是稀疏的（仅在序列结束），需要有效的信用分配
- 自回归结构自然地满足马尔可夫性

过渡衔接：有了 RL 形式化，现在让我们进入具体的算法设计——如何为 LLM 设计有效的策略梯度算法。

1.4 第 3 章：LLM 强化学习算法

涵盖范围: Slide 25-30 | **核心主题**: 设计适用于大语言模型的强化学习算法，包括值函数基线、GRPO、参考模型正则化等关键技术组件

1.4.1 3.1 策略梯度与基线（Slide 25-27）

概念详解

Slide 25-26 总结了 LLM 强化学习的核心算法框架。重复 K 次采样的 PPO 风格更新是当前 LLM 后训练的标准做法。Slide 26 指出了算法设计的三个关键决策点：

1. **智能地选择基线（baseline）和正则化器（regularizer）**
2. **智能地设置或学习奖励（reward）**
3. （隐含在架构图中）

Slide 27 深入讨论了值函数基线（value function baselines）和广义优势估计（GAE, Generalized Advantage Estimation）。值函数 $V(s_t)$ 被用来计算优势函数 $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$，从而降低策略梯度的方差。

LLM 场景中值函数的一个关键设计选择：**值函数只在后缀（suffix）上训练，而不在 prompt 上训练**。这是因为 prompt 是用户给定的、不可控的——对 prompt 中的 token 估计"价值"没有意义。值函数网络可以是 Transformer 的一个额外输出头（a new head），也可以是整个网络的副本。

深度剖析

在 LLM 的 RL 训练中，值函数的设计有几个微妙之处：

参数共享 vs 独立网络：Slide 27 展示了两种选择—— “add a new head”（在现有 Transformer 上加一个值函数输出头）或”make a copy of the entire network”（复制整个网络作为值函数）。前者的优势是参数效率高（值函数与策略共享 Transformer 的表示层），后者的优势是避免了策略和值函数训练之间的干扰。在实践中，如果计算资源允许，独立的 Critic 网络通常更稳定。

只训练后缀的值函数：这一设计选择反映了 LLM 场景的特殊性。在标准 RL 中，值函数对所有状态都有定义。但在 LLM 的 RL 微调中，prompt 是外生给定的——用户问什么模型无法控制。对 prompt token 估计值函数不仅没有意义，还可能导致值函数网络将计算资源浪费在不可控的状态上。

GAE 的适用性：GAE (Schulman et al., 2016) 通过组合多步 TD 误差来平衡偏差和方差。在 LLM 的 token 级别决策中，GAE 可以帮助在稀疏奖励下更有效地分配信用——即使最终奖励只在序列末尾给出，GAE 可以通过值函数引导，将功劳分配给”关键”的 token。

关键要点

- PPO + 重要性加权是 LLM RL 的标准算法范式
- 三个关键设计：基线选择、正则化器、奖励设计
- 值函数只在后缀上训练——prompt 是外生的，不可控
- GAE 帮助在 token 级别的稀疏奖励中进行信用分配

过渡衔接：值函数是好的基线，但它需要训练一个额外的网络。有没有不需要值函数的替代方案呢?

1.4.2 3.2 无值函数的基线方法（Slide 28-29）

概念详解

Slide 28 引入了 GRPO（Group Relative Policy Optimization）——一种不需要值函数的基线方法。GRPO 的核心思想非常简洁：**对于同一个 prompt，采样 K 个不同的回答**，然后使用这些回答的平均奖励作为”基线”。

具体来说，对于 prompt x，采样 $y_1, \dots, y_K \sim \pi_{\theta_{\text{old}}}$，然后计算相对优势：

$$A(y_i) = \frac{r(y_i) - \text{mean}(\{r(y_1), \dots, r(y_K)\})}{\text{std}(\{r(y_1), \dots, r(y_K)\})}$$

优势函数的”基线”就是这批样本的平均奖励——这避免了训练一个独立的值函数网络的额外复杂性和计算开销。

Slide 29 引入了另一个关键的算法组件：**参考模型正则化**（reference model regularization)。核心思想是：在优化过程中，给策略加上一个 KL 散度惩罚项，约束其不要偏离原始的预训练模型太远：

$$\mathcal{L} = \mathbb{E}[r(y)] - \beta \cdot D_{KL}(\pi_\theta(y|x) \| \pi_{\text{ref}}(y|x))$$

引用 Slide 29 的语言:“避免产生无意义的文本”(avoid producing nonsense text)和”避免利用不完美的奖励模型”(avoids exploiting imperfect reward models)。

深度剖析

GRPO 和参考模型正则化分别解决了两个关键问题:

GRPO 解决”基线从哪里来”的问题。在标准 PPO 中,基线由值函数网络提供。但在 LLM 场景中,由于动作空间巨大且状态(前缀)本身就可以被 Transformer 有效编码,K 个独立采样的平均奖励本身就构成了一个合理的基线——它度量了”对于这个 prompt,模型平均能生成多好的回答”。如果一个回答的奖励高于平均,那就是”好”的;低于平均就是”差”的。这种相对比较比绝对奖励值更适合驱动策略改进。

参考模型正则化解决”奖励黑客”(reward hacking)问题。当使用不完美的奖励模型时(无论是由人类偏好训练的 Bradley-Terry 模型还是自动验证器),策略可能会学会利用奖励模型的漏洞——生成在奖励模型看来高分但实际质量很差的文本。KL 散度约束确保了策略不会”飘移到奖励模型盲区的区域”——如果偏离预训练模型太远,策略会受到惩罚。这在实践中是一个至关重要的保护机制。

关键要点

- GRPO 用同一 prompt 的 K 个回答的平均奖励作为基线,取代值函数
- 避免了训练独立值函数网络的计算开销和工程复杂性
- 参考模型正则化通过 KL 散度约束防止策略过度偏离预训练模型
- KL 约束的双重作用:防止无意义文本生成 + 防止奖励黑客

> **过渡衔接**:Slide 30 将所有组件整合为一个完整的算法流程图,展示了基线、正则化和奖励学习如何协同工作。这自然引出了下一章的主题——如果奖励本身就是从偏好数据中学出来的,这个学习过程应该如何设计?

1.5 第 4 章:偏好与验证器

> **涵盖范围**:Slide 31-39 | **核心主题**:从人类偏好数据中学习奖励模型 (Bradley-Terry 模型)、RLHF 的完整算法流程、以及验证器奖励等其他奖励来源

1.5.1 4.1 Bradley-Terry 偏好模型(Slide 31-33)

概念详解

Slide 31 以一个问题开启了本章：“如果我们不知道奖励函数怎么办?”（What if we don’t know the reward?）这正是 RLHF 面临的核心挑战——我们没有一个预先定义好的奖励函数来判断 LLM 的输出质量，我们有的只是人类对”回答 A 比回答 B 更好”的**偏好比较**数据。

Slide 32 引入了 Bradley-Terry 模型（也称为”验光师算法”——optometrist algorithm），这是一个经典的成对比较统计模型。模型假设：对于两个选项（两个 LLM 回答），人类选择”更好”的那个的概率由它们隐含的”真实质量分数”决定：

$$p(y_1 \succ y_2) = \frac{\exp(r(y_1))}{\exp(r(y_1)) + \exp(r(y_2))} = \sigma(r(y_1) - r(y_2))$$

其中 $r(y)$ 是回答 y 的隐含奖励/质量，σ 是 Sigmoid 函数。Slide 33 指出这个模型与棋类 Elo 评分系统使用的是同一种方法。

深度剖析

Bradley-Terry 模型的优雅之处在于它只需要**相对比较**而非**绝对评分**。让标注员在 1-10 分的尺度上给一个 LLM 回答打分是非常困难且不稳定的——不同标注员的基准不同、同一个标注员在不同时间的标准也可能漂移。但让标注员判断”回答 A 和 B 哪个更好”则相对容易和一致。

模型参数的训练等价于一个逻辑回归（logistic regression）问题——这恰好是 Slide 34 中的观察。给定偏好数据 $\mathcal{D} = \{(y_1, y_2, \text{pref})\}$（其中 $\text{pref} \in \{1, 2\}$ 表示哪个回答被偏好），奖励模型 r_ψ 的训练目标是最小化：

$$\mathcal{L}(\psi) = -\sum_{(y_1, y_2, \text{pref})} \log \sigma(r_\psi(y_{\text{pref}}) - r_\psi(y_{\text{other}}))$$

这等价于一个二分类交叉熵损失——奖励模型只需要学会”分辨”更好和更差的回答。

Bradley-Terry 模型的一个隐含假设是：**更好的轨迹在指数尺度上更可能被选择**（“better trajectories are exponentially more likely to be chosen based on their reward”）。这个指数关系的假设意味着奖励差异的单位具有一致的含义——奖励差 1 分意味着”更好”的概率是 $\sigma(1) \approx 73\%$。

关键要点

- Bradley-Terry 模型从成对偏好比较中学习隐含的奖励分数
- 只需要相对比较，不需要绝对评分 $\rightarrow$ 降低了标注难度
- $p(y_1 \succ y_2) = \sigma(r(y_1) - r(y_2))$：奖励差距通过 Sigmoid 映射为选择概率
- 训练等价于逻辑回归：学会区分”更好的回答”和”更差的回答”

过渡衔接：有了 Bradley-Terry 奖励模型，RLHF 的完整流程就可以展开了。

1.5.2 4.2 RLHF 完整算法（Slide 34-38）

概念详解

Slide 34-38 展示了从人类反馈中进行强化学习的完整算法流程（RLHF）。这个流程由三个步骤组成：

步骤 1：收集偏好数据。使用当前（或初始）策略模型，对同一个 prompt 生成多个不同的回答（“on-policy responses for the same prompt”）。让人工标注员比较这些回答，选择更好的那个。

步骤 2：训练奖励模型。使用 Bradley-Terry 模型（等价于逻辑回归）从偏好数据中学习奖励函数 r_ψ。Slide 34 特别强调”this can be incomplete optimization”——奖励模型的训练不需要达到完美收敛。事实上，在实践中通常只做少量迭代（“Typically use only 1 or a few iterations”），因为奖励模型的过度优化也会导致过拟合。

步骤 3：用 RL 微调策略。将学到的奖励模型作为奖励函数，使用 PPO（或 GRPO 等变体）来优化语言模型策略。这一步通常包含 KL 散度正则化来约束策略不偏离参考模型太远。

深度剖析

Slide 34 提出了一个关键问题：“Why?”——为什么通常只用 1 次或很少几次 RL 迭代?

答案涉及多个因素：1. **奖励模型的不完美性**：Bradley-Terry 奖励模型只是一个近似——它无法完美地捕获人类偏好的所有细微之处。如果进行太多轮 RL，策略会逐渐”利用”奖励模型的缺陷（reward hacking），偏离真实的偏好。2. **分布漂移**：每一轮 RL 都会改变策略，从而改变策略生成的数据分布。但奖励模型是在”旧”策略的数据上训练的。随着策略漂移，奖励模型的预测变得越来越不可靠。3. **计算成本**：收集人类偏好数据需要时间和金钱，在每一轮 RL 后重新收集大量新鲜偏好数据是昂贵的。4. **实践经验**：在 ChatGPT 和类似模型的训练中，通常 1-3 轮 RLHF 就足以在不引入显著 reward hacking 的情况下改善模型行为。

Slide 37 展示了 RLHF 在 LLM 中的具体实现细节。关键点包括：奖励只在生成的最后一步给出（“Reward is assigned only at the last step of the generation”）——这意味着信用分配需要处理整个生成序列的稀疏奖励。Slide 38 用一图总结了所有涉及的模型：预训练 LLM、SFT 模型、奖励模型和参考模型之间的关系。

关键要点

- RLHF 三步骤：收集偏好 $\rightarrow$ 训练奖励模型 $\rightarrow$ RL 微调策略
- 通常只需 1-3 轮迭代：避免奖励黑客和分布漂移
- 奖励模型训练是”不完整优化”——过度训练反而有害
- 奖励只在序列最后一步给出（稀疏奖励），需要有效的信用分配

过渡衔接：RLHF 使用的奖励模型基于人类偏好。但有些任务的奖励可以更直接、更客观地获取——这就是验证器和过程奖励的作用。

1.5.3 4.3 验证器奖励与过程奖励（Slide 39）

概念详解

Slide 39 展示了 RLHF 之外的另外两种奖励来源：

验证器奖励（Verifier Reward）：模型评估最终答案并判断其是否正确。对于数学问题，这可以是”最终数值是否正确”；对于代码生成，这可以是”测试用例是否全部通过”。验证器奖励是**客观的**——它不依赖于人类的主观偏好判断。但验证器奖励可能过于稀疏：只知道最终答案对或错，不知道中间过程的哪些步骤是对的、哪些是错的。

过程奖励（Process Reward）：模型评估思维链（Chain-of-Thought）生成过程中的每一步，判断每一步推理是否正确。过程奖励提供了更细粒度的监督信号——每一步推理都能得到反馈，而不是等到最后才知道对错。但实现过程奖励需要能够自动评估中间推理步骤的正确性，这在技术上更具挑战性。

深度剖析

验证器奖励和过程奖励代表了从”结果监督”（outcome supervision）到”过程监督”（process supervision）的演进。在数学推理训练中，这一区分尤为重要：

结果监督（验证器奖励）：只知道”最终答案 42 是正确的”。如果模型在 Step 3 犯了一个错误但在 Step 7 幸运地修正了，或者反之在 Step 1-6 完全正确但在 Step 7 计算出错——结果监督无法区分这两种情况。它只能给出通过/失败的二元信号。

过程监督（过程奖励）：对每一步给出正确/不正确的判断。这允许模型更精确地定位和修正错误。研究表明，过程监督在数学推理任务中显著优于结果监督——它提供了更丰富的训练信号，尤其对于长链条推理任务。

过程奖励的实现方法包括：（1）人工标注每一步的正确性（昂贵但最准确）、（2）使用自动验证工具检查中间步骤（如对于代数变形，可以用符号计算引擎验证等价性）、（3）训练一个独立的”过程评估模型”来自动判断中间步骤。

关键要点

- 验证器奖励判断最终答案的正确性——客观但稀疏
- 过程奖励评估思维链中每一步推理的正确性——细粒度但更难实现
- 过程监督比结果监督提供更丰富的训练信号
- 尤其适用于长链条推理任务（数学证明、代码生成）

> **过渡衔接**：到目前为止我们讨论的都是在完全可观测 MDP 框架下的 RL。但现实世界中的许多问题——以及 LLM 的某些应用场景——涉及部分可观测性。这是本章的额外主题。

1.6 第 5 章：部分可观测性与 POMDP

涵盖范围：Slide 40-51 | **核心主题**：部分可观测马尔可夫决策过程（POMDP）的基本概念、对 RL 方法的挑战、以及状态空间模型和历史状态两类解决方案

1.6.1 5.1 超越 MDP：部分可观测性（Slide 40-42）

概念详解

Slide 40-42 引入了部分可观测马尔可夫决策过程（Partially Observed MDP, POMDP）。在标准的 MDP 中，我们假设智能体能够完整地观测到环境状态 s_t。但在许多现实问题中，这个假设不成立。“大多数现实世界问题都是这样的!”（most real-world problems are like this!）

POMDP 的两个关键特征（Slide 41）：1. **状态不满足马尔可夫性**：当前的观测 o_t 不能唯一确定下一时刻的状态转移概率——历史信息（过去的观测和动作）对于预测未来是必要的 2. **真实状态未知**：智能体永远无法直接观测到完整的环境状态，只能通过不完整、可能有噪声的传感器获取部分信息

Slide 42 展示了 POMDP 会导致两种”奇怪”的现象：

例 1：信息收集动作（Information-Gathering Actions）：在 POMDP 中，有些动作的主要目的不是获得即时奖励，而是**获取信息**来减少不确定性。例如，在导航中”探头看看拐角后面”——这个动作本身不会让你更接近目标，但能帮你做出更好的后续决策。

例 2：随机最优策略可以是严格最优的（Stochastic Optimal Policies）：在 MDP 中，最优策略通常是确定性的（对每个状态总是采取同一动作）。但在 POMDP 中，**随机策略可以比任何确定性策略严格更优**。Slide 42 展示了一个经典的 POMDP 例子（状态 A、B、C，观测模糊），随机化可以防止对手或环境利用智能体的确定性行为模式。

深度剖析

POMDP 的随机最优策略性质是对标准 RL 直觉的一个重要挑战。为什么随机策略在 POMDP 中可以严格优于确定性策略?

考虑 Slide 42 所示的简化例子：智能体在两个无法区分的状态 A 和 B 中（观测相同），从 A 开始，选择动作 1（总是正确）或动作 2（总是错误）。智能体在 A 有 50% 概率到 B，在 B 有 100% 概率到 C（终止）。确定性策略”总是选动作 1”在 B 状态会犯错，而随机策略”以 50% 概率选 1 和 2”可能表现更好。关键是：在 POMDP 中，随机化可以打破由部分可观测性造成的”信息陷阱”。

关键要点

- POMDP 的特征：不完全观测 + 状态不满足马尔可夫性
- 信息收集动作在 POMDP 中至关重要——主动寻求减少不确定性

- POMDP 中随机策略可以严格优于确定性策略——这是与 MDP 的根本区别
- 大多数现实 RL 问题都具有部分可观测性

过渡衔接：面对 POMDP，不同的 RL 方法有不同的应对能力。有些方法可以直接处理，有些则需要特别小心。

1.6.2 5.2 各方法在 POMDP 中的表现（Slide 43-47）

概念详解

Slide 43-47 系统地分析了不同 RL 方法在 POMDP 中的适用性。Slide 43 首先提出了一个”陷阱问题”：“哪些方法能处理部分可观测性?”（Which methods handle partial observability?）答案取决于”handle”的含义。

如果”handle”意味着”找到给定策略参数化下的最优策略”，那么答案因方法而异：

策略梯度方法（Policy Gradient）：可以直接处理 POMDP。策略梯度不依赖于值函数的贝尔曼方程（需要马尔可夫性），它直接对策略参数求期望累积奖励的梯度。这意味着策略梯度方法即使在不满足马尔可夫性的情况下也能正常工作——只需要策略本身能”记住”足够的历史信息（如通过 RNN 或 Transformer）。

值函数方法（Q-learning，DQN）：需要特别小心。值函数的贝尔曼方程依赖于马尔可夫性：$Q(s,a) = r + \gamma \max_{a'} Q(s',a')$ 隐含地假设了”给定当前状态 s，未来的期望回报与过去无关”。但在 POMDP 中这个假设不成立——Slide 44 明确指出” every time we see this state, we expect to get this value, regardless of past states”在 POMDP 中不成立。

Actor-Critic 方法：取决于 Critic 的设计。如果 Critic 也使用了历史信息（如 RNN-based Critic），可以在 POMDP 中工作。

深度剖析

Slide 47 用一个具体例子展示了为什么值函数方法在 POMDP 中会出问题。考虑状态在观测空间中”重叠”（两个不同的真实状态产生相同的观测）的情况。标准的 Q-learning 会为这两个状态学习相同的 Q 值（因为它们”看起来一样”），但在真实环境中它们可能需要完全不同的动作。策略梯度方法可以避免这个问题，因为它学习的是”给定观测历史，应该采取什么动作的概率分布”——即使当前观测模糊，策略也可以利用历史来做出区分。

但策略梯度方法在 POMDP 中也有局限。Slide 43 指出，它们只能找到”策略类别中的最优策略”（best policy in policy class）。如果策略被参数化为”只依赖当前观测”（无记忆策略），那么即使是策略梯度也无法克服 POMDP 的根本性信息缺失——它需要一个能够利用历史的策略架构。

关键要点

- 策略梯度方法可以直接在 POMDP 中工作——不依赖马尔可夫性

- 值函数方法（Q-learning）在 POMDP 中会失败——贝尔曼方程需要马尔可夫性
- 策略梯度只能找到策略类别中的最优策略——策略架构决定能力上限
- 关键在于策略是否能够访问和利用历史信息

过渡衔接：既然原始观测不满足马尔可夫性，自然的解决方案是学习一个”更好”的状态表示——这就是状态空间模型和历史状态的思路。

1.6.3 5.3 状态空间模型与历史状态（Slide 48-51）

概念详解

Slide 48-51 介绍了处理 POMDP 的两大类方法。

状态空间模型（State Space Models, SSM）（Slide 48）：学习一个满足马尔可夫性的潜在状态空间。核心思想是：虽然原始观测 o_t 不满足马尔可夫性，但如果我们能学习一个潜在状态 z_t 的表示，使得 z_t 捕获了所有与未来预测相关的历史信息，那么在这个学习到的状态空间上，马尔可夫性可以被（近似地）恢复。

Slide 48 指出了这种方法的潜在局限：预测可能很困难（“Prediction can be hard”），而且也许我们不需要好的预测就能获得高奖励（“Maybe we don’t need good prediction to get high rewards”）。这暗示着为预测而优化的状态表示不一定是为控制而优化的状态表示。

历史状态（History States）（Slide 49）：更直接的方法——将整个观测-动作历史编码为一个状态。如果我们将迄今为止的所有观测和动作序列 $h_t = (o_1, a_1, o_2, a_2, \dots, o_t)$ 作为”状态”，那么这个状态定义上是满足马尔可夫性的——它包含了所有历史信息。

深度剖析

历史状态方法看似简单粗暴，但它触发了一个自然的追问：**如何表示和编码历史？** Slide 50 讨论了模型架构的选择：

- **固定长度历史（Fixed Short History）**：只用最近 N 步的观测和动作。简单但有信息损失——太久之前的信息会被丢弃。Slide 50 评价”有时这是不好的”（Sometimes⋯Is that bad? Sometimes⋯）。
- **序列模型（Sequence Model）**：使用 RNN、LSTM 或 Transformer 将任意长度的历史编码为一个固定维度的状态向量。这是实践中更常见的方法——它允许模型自适应地决定哪些历史信息是重要的、哪些可以遗忘。

Slide 51 总结了 POMDP 的全貌：“POMDP 很奇怪”（POMDPs are weird）——它们挑战了许多基于 MDP 的直觉。“有些方法可以直接用”——策略梯度方法。“但最高效的方法不能，因为它们需要值函数”——Q-learning 等需要马尔可夫性。“我们甚至不需要为控制优化而完美地建模状态”——在 model-based RL 部分会有更深入的讨论。

关键要点

- 状态空间模型：学习一个满足马尔可夫性的潜在状态表示
- 历史状态：将完整历史编码为状态——定义上满足马尔可夫性
- 预测 vs 控制的张力：好的预测模型不一定导向好的策略
- 序列模型（RNN/LSTM/Transformer）是编码历史的实际工具
- 本讲的 POMDP 讨论为后续 model-based RL 的内容埋下了伏笔

过渡衔接：第 14 讲从 IRL 出发，穿过 LLM 的 RL 训练，最终到达 POMDP——这条路线勾勒了现代 RL 从理论框架到大规模实践再到前沿挑战的完整图景。

本讲总结：第 14 讲是 CS 285 课程中最”广阔”的一讲，横跨了逆强化学习、大语言模型的 RL 训练和部分可观测性三个看似独立但实则通过”从数据中学习决策”这一主线相连的领域。第一部分回顾了 IRL 与 GAN 的等价性——为理解”从比较中学习奖励”打下了基础。第二部分将这一思想引入 LLM 后训练——展示了 RLHF（从偏好中学习奖励）和验证器方法（从客观信号中学习奖励）如何成为现代 LLM 训练的核心组件。第三部分深入了算法的具体设计——PPO、GRPO、值函数基线、参考模型正则化。最后，第五部分以 POMDP 的视角提醒我们：现实世界中的决策几乎总是面临部分可观测性——这一认识为后续的 model-based RL 讨论提供了关键的动机。