大模型的探索与实践

Introduction to Large Language Models

§ 2.3 RLHF

Reinforcement Learning from Human Feedback

滕佳烨
上海财经大学
www.tengjiaye.com

回顾 Recall

In-context Learning

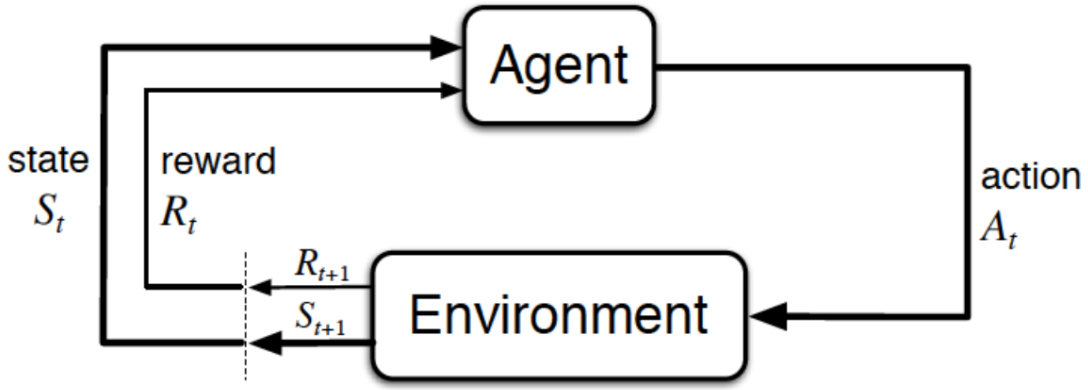
- 影响因素：示例顺序、示例分布
- 非影响因素：类别准确性

Fine-tuning

- 使用低秩技术进行快速微调

今天的任务：RLHF

强化学习中的基本元素



- Agent 决策主体
- Environment 环境

- State: 状态
- Action: 行动
- Reward: 奖励

在每一时刻，Agent需要根据当前环境状态 s_t 执行动作 a_t，并收到反馈奖励 r_t

目标：通过选取合适的策略 $\pi(s) = a$，能够最大化奖励 $\mathrm{R} = \sum r_t$.

马尔科夫决策过程 MDP

RL 是近似求解 MDP 的算法集

- State: 状态
- Action: 行动
- Reward: 奖励

MDP 五元组
S: 状态集：状态的集合
A: 动作集：行动的集合
P: 转移概率函数矩阵：给定状态与动作后下一个状态的概率
R: 奖励函数：给定状态与行动后的奖励
γ: 折现因子：下一时刻单位回报在当前的价值。

目标：通过选取合适的策略 $\pi(s) = a$，能够最大化奖励

$$\pi^* = \arg\max \mathrm{E}_\pi \left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \right).$$

值函数 value function

Value function $v_\pi(s) = \mathbb{E}_\pi(R_t|s_t = s)$.
Value q-function $q_\pi(s,a) = \mathbb{E}_\pi(R_t|s_t = s, a_t = a)$.

：我们想要优化的目标

Value function 迭代

$$v_\pi(s) = \mathbb{E}_\pi(r_{t+1} + \gamma v_\pi(s_{t+1})|s_t = s).$$
$$q_\pi(s,a) = \mathbb{E}_\pi(r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})|s_t = s, a_t = a).$$

：通过这个迭代求解value func

Bellman Equation

$$v_\pi(s) = \sum_{a\in A} \pi(a|s)\left[R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a v_\pi(s')\right]$$
$$q_\pi(s,a) = R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a \left(\sum_{a'\in A} \pi(a'|s') q_\pi(a', s')\right)$$

：将转移概率矩阵带入后，得到的迭代式

值函数 value function

最优值： $v^*(s) = \max_\pi v_\pi(s)\,, q^*(s,a) = \max_\pi q_\pi(s,a)\,.$

Value q-function $q_\pi(s,a) = \mathbb{E}_\pi(R_t|s_t = s, a_t = a).$

Bellman最优方程

$$v^*(s) = \max_a R_s^a + \gamma \sum P_{ss'}^a v^*(s')\,.$$
$$q^*(s,a) = R_s^a + \gamma \sum P_{ss'}^a \max q^*(a',s')$$

Bellman方程

$$v_\pi(s) = \sum_{a\in A} \pi(a|s) \left[R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a v_\pi(s') \right]$$

$$q_\pi(s,a) = R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a \left(\sum_{a'\in A} \pi(a'|s') q_\pi(a',s') \right)$$

推导核心：在MDP中，最优策略一定是确定性策略

$$v^*(s) = \max_a q^*(s,a)$$

值迭代与策略迭代 value iteration and policy iteration

策略迭代：

1. 初始化策略 π_0， $v_{\pi_0}(s)$
2. 策略迭代 $t = 0, 1, 2, \cdots$
   a. 策略评估：
      - ★ 迭代 $k = 1, 2, \cdots$
      - ★ $v_{k+1}(s) = \sum_{a \in A} \pi_t(a|s) \left[R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_k(s') \right]$
      - ★ 收敛得到 $v_{\pi_t}(s)$
   b. 策略改进：$\pi_{t+1}(s) = \arg\max_a q_{\pi_t}(s, a) = \arg\max_a R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_{\pi_t}(s')$

Bellman方程

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left[R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right]$$

$$q_\pi(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \left(\sum_{a' \in A} \pi(a'|s') q_\pi(a', s') \right)$$

由于我们已知最优策略为确定性策略，需要最后进行策略改进

值迭代与策略迭代 value iteration and policy iteration

值迭代:
每次策略评估只迭代一次，可以写成

$$v_{t+1}(s) = \max_a \left[R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_t(s') \right]$$

Bellman最优方程

$$v^*(s) = \max_a R_s^a + \gamma \sum P_{ss'}^a v^*(s').$$
$$q^*(s,a) = R_s^a + \gamma \sum P_{ss'}^a \max q^*(a', s')$$

当求解出最优value后，可以计算 $q^*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v^*(s')$
而后寻找最优策略 π^*

模型（转移矩阵）未知？

没有办法直接使用转移矩阵 P，怎么办？
使用采样的方式进行！

采样是对状态转移矩阵的近似

时序差分（TD）：考虑到

$$q_\pi(s,a) = \mathrm{E}_\pi(r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a)$$

用 $r_{t+1} + \gamma q(s_{t+1}, a_{t+1})$ 替换 $\tilde{R}_t(s,a)$，每一步更新一次 $q(s,a)$

TD(n) 使用计算 n 步 TD 目标的指数加权平均作为 TD 目标

- 一步 TD 目标：$r_{t+1} + \gamma q(s_{t+1}, a_{t+1})$
- 两步 TD 目标：$r_{t+1} + \gamma r_{t+2} + \gamma^2 q(s_{t+2}, a_{t+2})$
- 类似可得 k 步 TD 目标

结合一般策略评估 -策略改进框架，可以得到时序差分算法。

- 著名的Q-Learning 就属于TD算法中的一种

用参数模型近似value function

$$\theta \leftarrow \theta + \eta[r_{t+1} + \gamma \hat{q}(s_{t+1}, a_{t+1}, \theta) - \hat{q}(s, a, \theta)]\nabla_\theta \hat{q}(s, a, \theta)$$

梯度下降

估计值　　近似值

值函数近似的 TD(1) 的强化学习

- 初始化 θ，环境状态 s_0
- 策略迭代 $t = 0, 1, 2, \cdots$
  1. 基于$\hat{q}(s, a, \theta)$ 的 $\epsilon-$ 贪婪策略，在状态 s_t 下选择动作 a_t
  2. 执行动作 a_t，进入状态 s_{t+1}，并得到回报 r_{t+1}
  3. 根据某个策略，在状态 s_{t+1} 下选择动作 a_{t+1}
  4. 使用值函数近似的 TD 方法，更新参数 θ

$$\theta \leftarrow \theta + \eta[r_{t+1} + \gamma \hat{q}(s_{t+1}, a_{t+1}, \theta) - \hat{q}(s_t, a_t, \theta)]\nabla_\theta \hat{q}(s_t, a_t, \theta)$$

- 著名的DQN就属于此

用参数模型近似value function

$$\theta \leftarrow \theta + \eta[r_{t+1} + \gamma \hat{q}(s_{t+1}, a_{t+1}, \theta) - \hat{q}(s, a, \theta)]\nabla_\theta \hat{q}(s, a, \theta)$$

梯度下降

估计值　　近似值

DQN 论文给出不收敛的原因

1. 输入数据不是独立同分布的，可能具有高度序列相关性
2. 神经网络近似的值函数 $\hat{q}(s, a, \theta)$ 不稳定将导致基于 $\hat{q}(s, a, \theta)$ 的贪婪策略不稳定，不利于生成好的 TD 目标用于网络训练。
3. 某次回报的（绝对）值过大会导致梯度下降训练不稳定。

改进方法

1. 引入经验回放
2. 引入目标网络
3. 限制 TD 误差在 [-1,1] 区间：
   即目标损失使用 Huber Loss，将 [-1,1] 区间之外的平方损失替换为绝对值损失。

- 著名的DQN就属于此

用参数模型近似policy function

- 用值函数近似直接对策略进行建模，优化$\pi_\theta = \pi(a|s,\theta)$

- REINFOECE 算法：

1. 初始化策略 π_θ
2. 多局制（episode）环境下，迭代 $k = 0, 1, 2, \cdots$
   - 2.1 根据策略 π_θ，抽样得到第 k 局状态 -动作 - 回报的轨迹
     $s_0, a_0, r_1, s_1, a_1, r_2, \cdots, r_T$
   - 2.2 迭代 $t = 0, 1, 2, \cdots$
     - a. 计算 $G_t = \sum_{j=t+1}^{T} \gamma^{j-t-1} r_j$
     - b. $\theta \leftarrow \theta + \eta G_t \nabla_\theta \log \pi(a_t|s_t,\theta)$

RLHF in LLM: 用人类偏好训练大模型

- 大模型中每个时刻生成一个token → action
- 当前人类用户的输入+大模型已经有的输出 → state
- 人类对每个语言有一定的偏好 → reward

在RLHF中，我们往往需要四个模型

- Actor Model: 我们想要训练的LM模型
- Critic Model: 预估总收益 V_t
- Reward Model: 预估即时收益 R_t （参数冻结）
- Reference Model: 在RLHF阶段为语言增加“约束” （参数冻结）

RLHF in LLM: 用人类偏好训练大模型

Actor Model：使用SFT结束后的模型来进行初始化

训练方式：每次给一个prompt，用Actor生成对应的response，而后送入奖励模型计算loss，并用于更新actor

RLHF in LLM: 用人类偏好训练大模型

Actor Model：使用SFT结束后的模型来进行初始化
Reference Model：使用SFT结束后的模型来进行初始化

在训练过程中，reference需要冻结，作用为防止RLHF将Actor Model“训坏了”
具体操作：惩罚actor model (可变) 与 Reference model (不可变) 之间的KL

RLHF in LLM: 用人类偏好训练大模型

Actor Model：使用SFT结束后的模型来进行初始化
Reference Model：使用SFT结束后的模型来进行初始化
Reward Model：额外训练得到，在训练中冻结

可以由人类数据进行训练，一般是一个比较小的神经网络
例如可以利用人类偏好数据，提前训练reward model

RLHF in LLM: 用人类偏好训练大模型

Actor Model：使用SFT结束后的模型来进行初始化
Reference Model：使用SFT结束后的模型来进行初始化
Reward Model：额外训练得到，在训练中冻结
Critic Model：一般由Reward Model初始化得到，并在RLHF中进行参数更新

虽然我们有一个reward model，但其为即时收益，并非是全部的总收益。因此我们需要训练一个模型去预测它。

总结 Take-away Messages

RL

- Agent, Environment
- State, Action, Reward
- Policy Iteration, Value Iteration
- No transition matrix → Sampling
- No policy function → DL

RLHF

- Actor Model, Reference Model, Reward Model, Critic Model

第八次作业：请在Github上找一个RL玩游戏的算法，并跑通它