RL from Human Feedback (RLHF)

Recap: prelim exam

The last question shows a proof of Q-learning converging to $Q^\star$
and provides a way to calcuate the convergence rate

Recap

We have covered a few RL algorithms, TD, DQN, REINFORCE, PPO;

They all rely on a key and strong assumption: reward function/signal is given

Question today:

What to do when the reward function is unknown

Outline

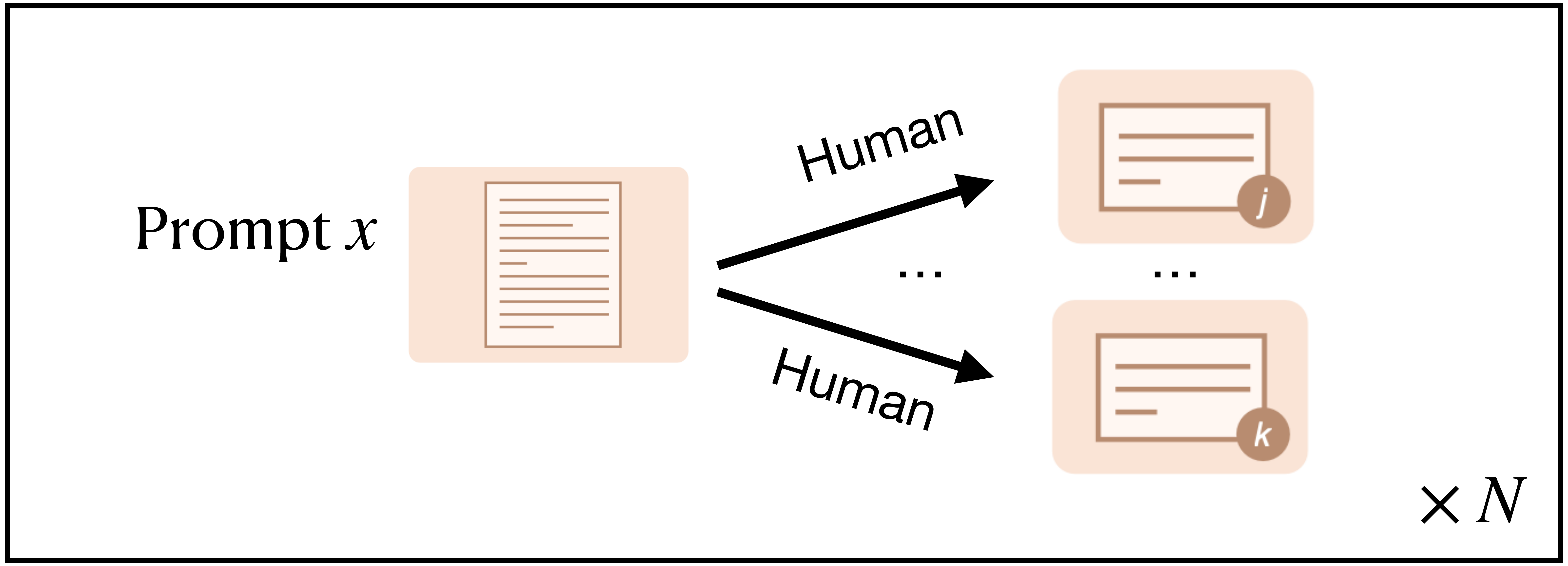
1. LLM as a policy
2. Learning reward functions from preference data
3. KL-regularized RL

Motivation

Modern chatbots are pre-trained via next-token prediction on web data, followed by fine-tuning using human preference via RL (post-training)

The post-training Pipeline: Supervised Fine-tuning (SFT)

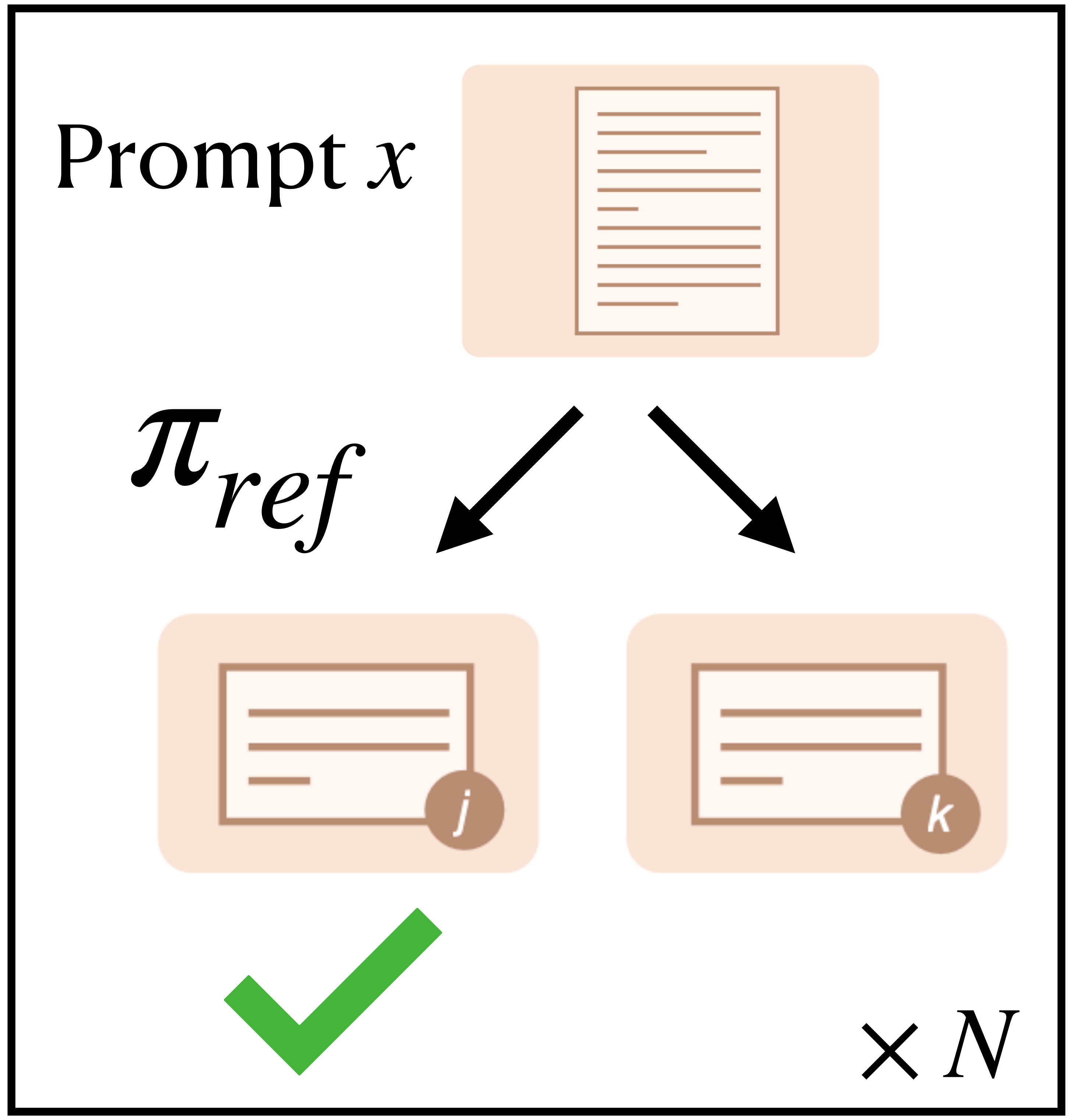
Collect instruction-response data



SFT: given prompts, train LLM to predict tokens in human responses

The post-training Pipeline: RLHF

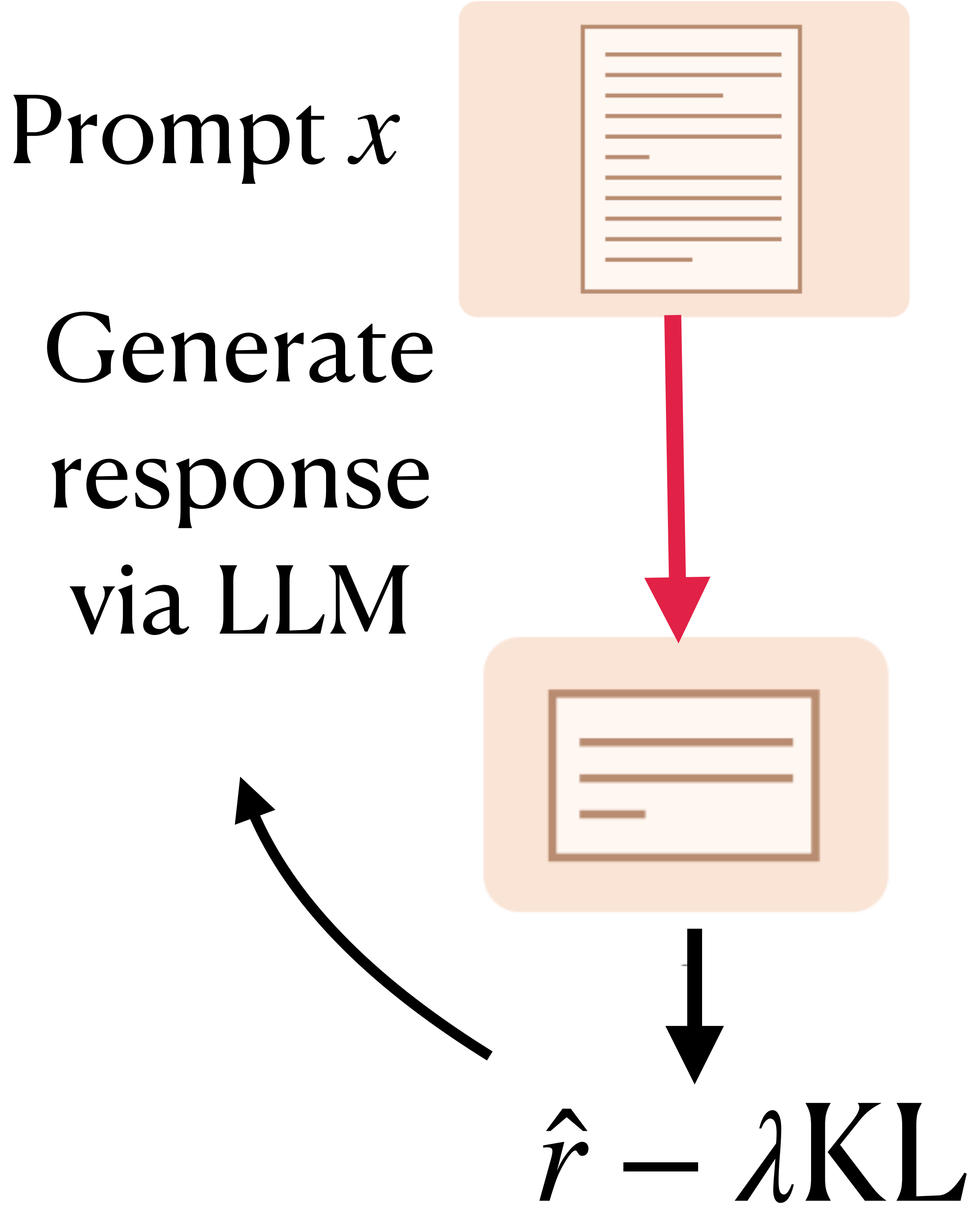
1. Collect preference dataset



$\mathcal{D}_{off} = \{x, \tau, \tau', z\}$

2. Learn a reward model $\hat{r}$ using the data from step 1

3. train policy via RL (e.g., PPO)



[ChatGPT blog post: https://openai.com/index/chatgpt/]

What's the benefit of RLHF over SFT?

Evaluation is often easier than generation

Given a high quality reward, RLHF can often make model outperform humans:

Model size	Algorithm	Winrate ($\uparrow$)
6.9B	SFT	45.2 ($\pm$2.49)
	DPO	68.4 ($\pm$2.01)
	REINFORCE	70.7*
	PPO	77.6‡
	RLOO ($k = 2$)	74.2*
	RLOO ($k = 4$)	77.9*
	REBEL	**78.1** ($\pm$1.74)

* directly obtained from Ahmadian et al. (2024)

‡ directly obtained from Huang et al. (2024)

RL-based methods learn a model better than humans (task: writing short summaries of reddit posts)

The MDP formulation of text generation

Initial state s_0: prompt x

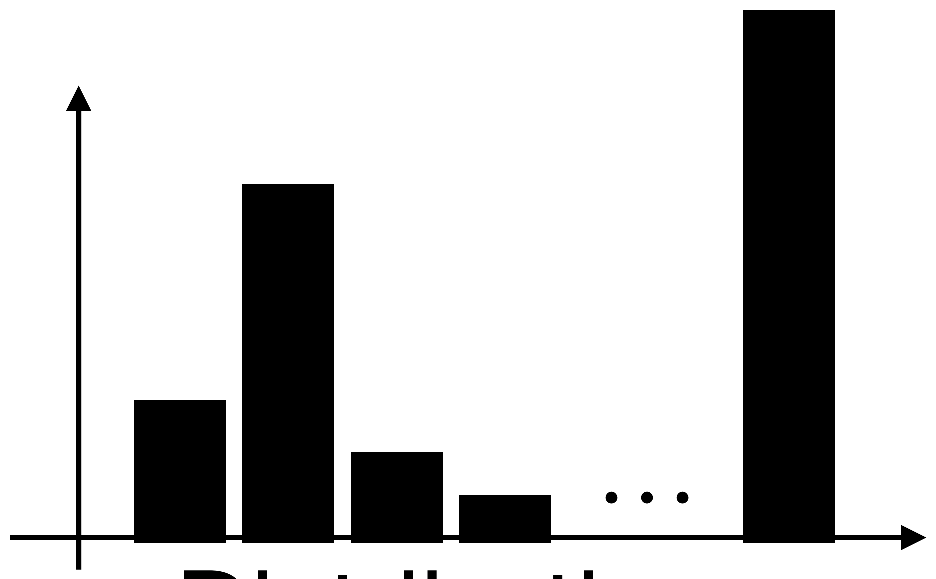
Action: token y; action space: all possible tokens

State: prompt + generated tokens, e.g., $s_h = (x, y_0, y_1, \ldots, y_{h-1})$

Transition: concatenation, i.e., given s_h and y_h, $s_{h+1} = (s_h, y_h)$

Terminate: either hits the maximum content length or hits the special EOS token

The LLM itself is a differentiable policy



Distribution over all tokens

$\pi_\theta(\cdot \mid s_h)$

$y_h \sim \pi_\theta(\cdot \mid s_h)$

LLM (decoder only transformer w/ parameters θ)

Differentiable: can compute $\nabla_\theta \ln \pi_\theta(y \mid s_h)$ via backprop

State $s_h = (s_0, y_0, y_1, \ldots y_{h-1})$

Outline

1. LLM as a policy

2. Learning reward functions from preference data

3. KL-regularized RL

Learning reward from human data

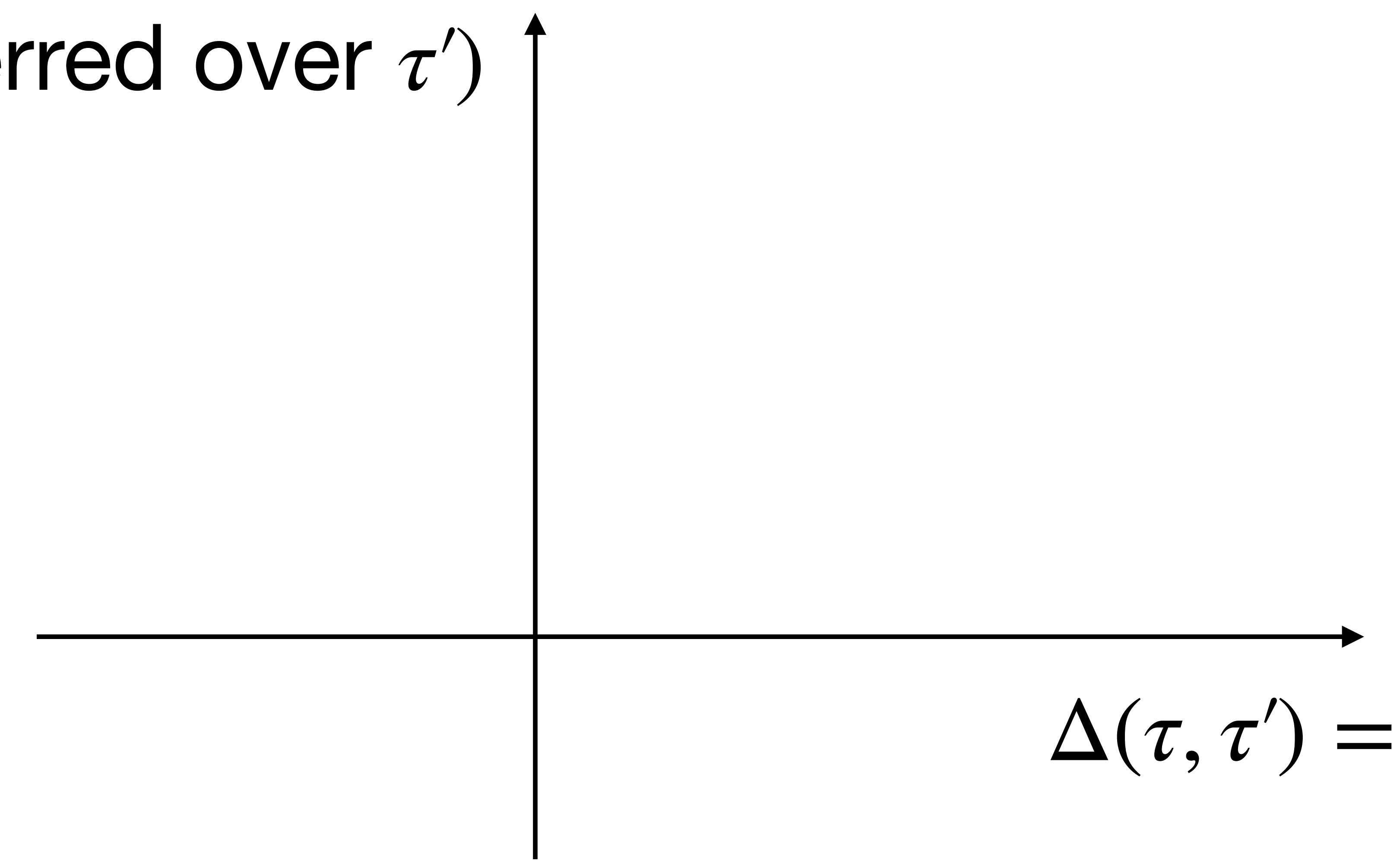
Reward design can be challenging in RL

Bradley-Terry Model

Assume there is a ground truth reward $r^\star(x, \tau)$ (i.e., high reward means response is good)

The BT model assumes that humans generate labels based on the following probablistic model:

$$P(\tau \text{ is prefered over } \tau' \text{ given } x) = \frac{1}{1 + \exp\left(-\left(\underbrace{r^\star(x,\tau) - r^\star(x,\tau')}_{\Delta(\tau,\tau')}\right)\right)}$$



Learning reward based on the Bradley-Terry assumption

Given a preference dataset $\mathscr{D} = \{x, \tau, \tau', z\},$ where label $z \in \{1, -1\}$ is generated via BT on $r^\star$

(1 indicates τ is prefered over τ'; -1 otherwise)

Q: can you write down the reward learning loss via MLE?

Q: let's assume we have infinite data and perform MLE optimization, can we discover the exact $r^\star$?

Outline

1. LLM as a policy

2. Learning reward functions from preference data

3. KL-regularized RL

RL is very good at reward hacking

The boat racing example

https://openai.com/index/faulty-reward-functions/

To avoid reward hacking

We form the following KL regularized RL objective

β : controls the strength of KL-reg;

$$J(\pi_\theta) = \mathbb{E}_{x\sim\nu}\left[\mathbb{E}_{\tau\sim\pi(\cdot|x)}\hat{r}(x,\tau) - \beta \mathrm{KL}\left(\pi(\cdot\,|\,x)\middle|\pi_{ref}(\cdot\,|\,x)\right)\right]$$

“stay close” to the SFT policy π_{ref}.

Q: Why this can help avoid reward hacking?

How to optimize the KL-reg RL objective

A simple heuristic is to add KL to reward

$$J(\pi_\theta) = \mathbb{E}_{x\sim\nu}\left[\mathbb{E}_{\tau\sim\pi(\cdot|x)}\hat{r}(x,\tau) - \beta \mathrm{KL}\left(\pi(\cdot\,|\,x)\middle|\pi_{ref}(\cdot\,|\,x)\right)\right]$$

$$= \mathbb{E}_{x\sim\nu}\left[\mathbb{E}_{\tau\sim\pi(\cdot|x)}\underbrace{\left(\hat{r}(x,\tau) - \beta\ln\frac{\pi(\tau\,|\,x)}{\pi_{ref}(\tau\,|\,x)}\right)}_{:=r_{new}(x,\tau)}\right]$$

Run PG (reinforce or PPO) w/ $r_{new}(x,\tau)$ as the reward signal

Remark: it works, but it is not the exact gradient (see Prelim Q5)

Summary

RLHF is a tool for post-training LLMs so that llms can understand and follow human instructions

Reward Model (RM) is learned from human feedback (i.e., pair-wise preference)

RM learning is based on the Bradley-Terry model

KL regularization is important to avoid hacking the learned RM