Regressing Relative Reward

Recap: KL-reg RL objective

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

$$\hat{\pi}(\tau|x) \propto \pi_{ref}(\tau|x) \cdot \exp\left(\frac{\hat{r}(x,\tau)}{\beta}\right)$$

Stay close to π_{ref} — Optimize reward

Recap: DPO

$$\arg\max_{\theta} \sum_{x,\tau,\tau',z} \ln \frac{1}{1 + \exp\left(-z \cdot \beta \left(\ln \frac{\pi_\theta(\tau \mid x)}{\pi_{ref}(\tau \mid x)} - \ln \frac{\pi_\theta(\tau' \mid x)}{\pi_{ref}(\tau' \mid x)}\right)\right)}$$

Use policies to model the reward difference (aka your LLM is your secret reward model)

But DPO's performance isn't as strong as RM+PPO in practice..

Evaluation and Generation gap (aka evaluation is easier than generation..)

When a reward / verifier is easier to learn, RM + PPO can win…

1. DPO uses finite data + gradient descent to learn the generator directly

2. PPO uses the RM, and can take advantage of unseen prompts and new training data…

But DPO's performance isn't as strong as RM+PPO in practice..

PPO can also take advantage of the state-of-art RMs from the community…

RewardBench: Evaluating Reward Models

Evaluating the capabilities, safety, and pitfalls of reward models

Code | Eval. Dataset | Prior Test Sets | Results | Paper | Total models: 165 | * Unverified models | ⚠️ Dataset Contamination |
Last restart (PST): 22:01 PDT, 28 Mar 2025

⚠️ Many of the top models were trained on unintentionally contaminated, AI-generated data, for more information, see this gist.

🏆 RewardBench Leaderboard | 🔍 RewardBench - Detailed | Prior Test Sets | About | Dataset Viewer

Model Search (delimit with ,)

☑ Seq. Classifiers ☑ DPO ☑ Custom Classifiers ☑ Generative
☐ Prior Sets

	Model	Model Type	Score	Chat	Chat Hard	Safety	Reasoning
1	infly/INF-ORM-Llama3.1-70B	Seq. Classifier	95.1	96.6	91.0	93.6	99.1
2	ShikaiChen/LDL-Reward-Gemma-2-27B-v0.1	Seq. Classifier	95.0	96.4	90.8	93.8	99.0
3	nicolinho/QRM-Gemma-2-27B	Seq. Classifier	94.4	96.6	90.1	92.7	98.3
4	Skywork/Skywork-Reward-Gemma-2-27B-v0.2	Seq. Classifier	94.3	96.1	89.9	93.0	98.1
5	nvidia/Llama-3.1-Nemotron-70B-Reward *	Custom Classifier	94.1	97.5	85.7	95.1	98.1

https://huggingface.co/spaces/allenai/reward-bench

Today's question

PPO can be very expensive, can we develop RL algorithm that is more efficient and may be more effective?

Outline

1. Mirror descent — reward maximization subject to a KL reg to the old policy

2. Reparametrization trick and REBEL

3. Connections to old algorithms we learned

Mirror Descent

Let us assume that we are given a reward function $r(x, \tau)$
(e.g., learned by ourselves or an open-source model)

$$\text{Want to } \max_{\pi} \mathbb{E}_{x,\tau\sim\pi(\cdot|x)} \left[r(x, \tau)\right]$$

Mirror Descent

Mirror descent (MD) incrementally (iteratively) updates the policy:

Given π_t , we update to π_{t+1} as follows:

$$\pi_{t+1} = \arg\max_{\pi} \mathbb{E}_{x,\tau\sim\pi(\cdot|x)} \left[r(x,\tau) - \beta \mathsf{KL}\left(\pi(\cdot\,|\,x) \,|\, \pi_t(\cdot\,|\,x) \right) \right]$$

KL to the previous policy (e.g., recall NPG and the logic behind PPO's clipping)

Mirror Descent

In theory, using the same idea we had from KL-reg RL, π_{t+1} has a closed-form:

$$\pi_{t+1} = \arg\max_{\pi} \mathbb{E}_{x,\tau\sim\pi(\cdot|x)} \left[r(x,\tau) - \beta \text{KL}\left(\pi(\cdot \,|\, x) \,|\, \pi_t(\cdot \,|\, x)\right)\right]$$

$$\Rightarrow \pi_{t+1}(\tau \,|\, x) \propto \pi_t(\tau \,|\, x) \exp\left(r(x,\tau)/\beta\right)$$

Q: can we easily implement π_{t+1}?

Mirror Descent

Ignoring the implementation issue, mirror descent in theory has very good convergence rate

After T iterations, we can find a policy $\hat{\pi}$, s.t.,

$$\left| \mathbb{E}_{x,\tau \sim \hat{\pi}(\cdot|x)} r(x,\tau) - \mathbb{E}_{x,\tau \sim \pi^\star(\cdot|x)} r(x,\tau) \right| \leq O(1/T)$$

(Proof out of the scope, see CS6789)

Outline

1. Mirror descent — reward maximization subject to a KL reg to the old policy

2. Reparametrization trick and REBEL

3. Connections to old algorithms we learned

Reparameterization

Mirror descent indicates the following ideal update:

$$\pi_{t+1}(\tau \mid x) = \pi_t(\tau \mid x) \exp\left(r(x,\tau)/\beta\right)/Z(x)$$

---

1. Take log on both sides and rearrange terms, we get

$$r(x,\tau) = \beta\left(\ln \frac{\pi_{t+1}(\tau \mid x)}{\pi_t(\tau \mid x)} + \ln Z(x)\right)$$

2. Instead of modeling reward, we model reward difference to cancel $Z(x)$:

$$r(x,\tau) - r(x,\tau') = \beta\left(\ln \frac{\pi_{t+1}(\tau \mid x)}{\pi_t(\tau \mid x)} - \ln \frac{\pi_{t+1}(\tau' \mid x)}{\pi_t(\tau' \mid x)}\right)$$

Reparameterization

We obtained the following relationship between r and π_{t+1} & π_t:

$$\forall (x, \tau, \tau') : \; r(x, \tau) - r(x, \tau') = \beta \left(\ln \frac{\pi_{t+1}(\tau \mid x)}{\pi_t(\tau \mid x)} - \ln \frac{\pi_{t+1}(\tau' \mid x)}{\pi_t(\tau' \mid x)} \right)$$

This indicates π_{t+1} is **the minimizer** of the following least square regression problem:

π_{t+1} should be the minimizer regardless of what the distribution is;

$$\mathbb{E}_{x,\tau,\tau'} \left(\beta \left(\ln \frac{\pi_{t+1}(\tau \mid x)}{\pi_t(\tau \mid x)} - \ln \frac{\pi_{t+1}(\tau' \mid x)}{\pi_t(\tau' \mid x)} \right) - \underbrace{\left(r(x, \tau) - r(x, \tau') \right)}_{\text{Relative reward}} \right)^2$$

In pratice, we often use $x, \tau \sim \pi_t(\cdot \mid x), \tau' \sim \pi_t(\cdot \mid x)$

REBEL algorithm

Put things together, we arrive at the following iterative algorithm:

Given π_t, we compute π_{t+1} via least square regression:

$$\pi_{t+1} = \arg\min_{\pi} \mathbb{E}_{x,(\tau,\tau')\sim\pi_t(\cdot|x)} \left(\underbrace{\beta\left(\ln \frac{\pi(\tau|x)}{\pi_t(\tau|x)} - \ln \frac{\pi(\tau'|x)}{\pi_t(\tau'|x)} \right)}_{\text{Regressor}} - \underbrace{\left(r(x,\tau) - r(x,\tau')\right)}_{\text{Relative reward}} \right)^2$$

sample τ, τ' from the latest policy $\pi_t(\cdot|x)$, independently;

Difference between REBEL and DPO

Discussion: what is the difference between DPO, REBEL, and PPO

Outline

1. Mirror descent — reward maximization subject to a KL reg to the old policy

2. Reparametrization trick and REBEL

3. Connections to old algorithms we learned

REBEL

Consider parameterized policy π_θ, recall that rebel solves least square regression every iteration:

$$\theta_{t+1} = \arg\min_{\theta} \mathbb{E}_{x,(\tau,\tau')\sim\pi_{\theta_t}(\cdot|x)} \left(\beta \left(\ln \frac{\pi_\theta(\tau \mid x)}{\pi_{\theta_t}(\tau \mid x)} - \ln \frac{\pi_\theta(\tau' \mid x)}{\pi_{\theta_t}(\tau' \mid x)} \right) - \left(r(x,\tau) - r(x,\tau') \right) \right)^2$$

In practice, hard to solve it exactly

What happens if we solve it approximately? What happens if we perform

1. one step of gradient descent

2. one step of Gauss-newton method

REBEL recovers variance-reduced policy gradient

Approximately solve the least square regression problem via just one step of gradient descent

$$\theta_{t+1} = \arg\min_{\theta} \underbrace{\mathbb{E}_{x,(\tau,\tau')\sim\pi_{\theta_t}(\cdot|x)} \left(\beta \left(\ln \frac{\pi_\theta(\tau|x)}{\pi_{\theta_t}(\tau|x)} - \ln \frac{\pi_\theta(\tau'|x)}{\pi_{\theta_t}(\tau'|x)} \right) - \left(r(x,\tau) - r(x,\tau') \right) \right)^2}_{:\ell(\theta)}$$

$\theta_{t+1} \Leftarrow \theta_t - \eta \nabla_\theta \ell(\theta_t)$ Let's try this out!

REBEL recovers variance-reduced NPG

Approximately solve the least square regression problem via just one step of Gauss-newton

$$\theta_{t+1} = \arg\min_{\theta} \mathbb{E}_{x,(\tau,\tau')\sim\pi_{\theta_t}(\cdot|x)} \left(\beta \left(\ln \frac{\pi_\theta(\tau|x)}{\pi_{\theta_t}(\tau|x)} - \ln \frac{\pi_\theta(\tau'|x)}{\pi_{\theta_t}(\tau'|x)} \right) - \left(r(x,\tau) - r(x,\tau') \right) \right)^2$$

GN approximate the non-linear part inside the square ***via first-order Taylor expansion at*** θ_t

$$\ln \frac{\pi_\theta(\tau|x)}{\pi_{\theta_t}(\tau|x)} - \ln \frac{\pi_\theta(\tau'|x)}{\pi_{\theta_t}(\tau'|x)} \approx \left(\nabla \ln \pi_{\theta_t}(\tau|x) - \nabla \ln \pi_{\theta_t}(\tau'|x) \right)^\top (\theta - \theta_t)$$

Plug the linear approximation back and solve for θ:

$$\theta_{t+1} = \arg\min_{\theta} \mathbb{E}_{x,(\tau,\tau')\sim\pi_{\theta_t}(\cdot|x)} \left(\beta \left(\left(\nabla \ln \pi_{\theta_t}(\tau|x) - \nabla \ln \pi_{\theta_t}(\tau'|x) \right)^\top (\theta - \theta_t) \right) - \left(r(x,\tau) - r(x,\tau') \right) \right)^2$$

Claim: this recovers the NPG update procedure (try this out after class!)

Using RL to optimize 7B size model on TL;DR

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

* directly obtained from Ahmadian et al. (2024)
‡ directly obtained from Huang et al. (2024)

1. Online RL + RM is often much better than DPO

2. REBEL is in par w/ PPO and RLOO (k=4), but much more computation and memory efficient

[REBEL: Reinforcement Learning via Regressing Relative Rewards, Neurips 2024]

Swimmer experiments in openAI Gym

Given the same amount of **labeled** preference data, rebel can continue learning using fresh online data