Direct Preference Optimization (DPO)

Recap: Bradley Terry model and reward model (RM) learning

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

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

We parameter a reward function (e.g., neural network) $r(x, \tau)$, and learn via MLE / logistic regression

$$\hat{r} = \arg\max_{r} \sum_{x,\tau,\tau',z} \ln \frac{1}{1 + \exp\left(-z \cdot \left(r(x,\tau) - r(x,\tau')\right)\right)}$$

Recap: KL-reg RL for avoiding reward hacking

β : controls the strength of KL-reg;

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

"stay close" to the SFT policy π_{ref}.

ChatGPT uses PPO to optimize $J(\pi)$….

When models are large…

RM + PPO can be hard to optimize…

At least need to maintain 4 big models in GPU RAM (RM, π, V, π_{ref}…)

Question today:

Can we combine the two stages together and learn policy directly?

Outline

1. KL-reg RL revisit and its closed-form solution

2. Reparametrization trick — modeling RM difference using policy directly

3. DPO Algorithm

First thing…

We will directly operate at the trajectory level, i.e., a trajectory is an action

Given prompt x, and an “action” (a trajectory) $\tau = \{y_0, y_1, \ldots, y_{H-1}\}$, what’s the likelihood of the “action” under the policy π_θ?

$$\pi_\theta(\tau \mid x) = \prod_{h=0}^{H-1} \underbrace{\pi_\theta(y_h \mid x, y_{<h})}_{\text{Likelihood of predicting } y_h \text{ given the past..}}$$

KL-reg RL objective

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

What's the $\arg\max_{\pi} J(\pi)$?

Consider on a (x, τ) pair, what is $\partial J(\pi) / \partial \pi(\tau \,|\, x)$?

$$\frac{\partial J(\pi)}{\partial \pi(\tau \,|\, x)} = \hat{r}(x, \tau) - \beta\left(\ln \pi(\tau \,|\, x) - \ln \pi_{ref}(\tau \,|\, x) + 1\right)$$

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

$$\pi(\tau \,|\, x) = \pi_{ref}(\tau \,|\, x) \exp\left(\frac{\hat{r}(x, \tau)}{\beta}\right) / Z(x), \text{ where } Z(x) = \mathbb{E}_{\tau \sim \pi_{ref}(\cdot|x)} \exp(\hat{r}(x, \tau)/\beta)$$

KL-reg RL objective

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

In sum, the optimal policy is:

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

1. When $\beta \to 0$:

2. When $\beta \to \infty$:

Outline

1. KL-reg RL revisit and its closed-form solution

2. Reparametrization trick — modeling **<u>RM difference</u>** using policy directly

3. DPO Algorithm

Can we parameterize RM using policies?

In sum, the optimal policy of the KL-reg RL objective is:

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

$$\ln \hat{\pi}(\tau \mid x) = \ln \pi_{ref}(\tau \mid x) - \ln Z(x) + \frac{\hat{r}(x,\tau)}{\beta}$$

$$\hat{r}(x,\tau) = \beta\left(\ln \frac{\hat{\pi}(\tau \mid x)}{\pi_{ref}(\tau \mid x)} + \ln Z(x)\right)$$

Not done yet, this $Z(x)$ technically contains $\hat{r}$!

But $\ln Z(x)$ is a shift that is independent of τ...

Cancelling the normalization constant $Z(x)$ via modeling the difference

$$\hat{r}(x, \tau) = \beta \left(\ln \frac{\hat{\pi}(\tau \,|\, x)}{\pi_{ref}(\tau \,|\, x)} + \ln Z(x) \right)$$

Not done yet, this $Z(x)$ technically contains $\hat{r}$!

But $\ln Z(x)$ is a shift that is independent of τ...

Given (x, τ, τ'), we just model **reward difference**:

$$\hat{r}(x, \tau) - \hat{r}(x, \tau') = \beta \left(\ln \frac{\hat{\pi}(\tau \,|\, x)}{\pi_{ref}(\tau \,|\, x)} - \ln \frac{\hat{\pi}(\tau' \,|\, x)}{\pi_{ref}(\tau' \,|\, x)} \right)$$

The annoying normalization term gone!

Outline

1. KL-reg RL revisit and its closed-form solution

2. Reparametrization trick — modeling RM difference using policy directly

3. DPO Algorithm

DPO

1. Take any policy π_θ, we can use it to model the reward difference:

$$r_\theta(\tau \mid x) - r_\theta(\tau' \mid x) := \beta \left(\ln \frac{\pi_\theta(\tau \mid x)}{\pi_{ref}(\tau \mid x)} - \ln \frac{\pi_\theta(\tau' \mid x)}{\pi_{ref}(\tau' \mid x)} \right)$$

2. Now plug this into the MLE loss we had for learning the reward difference:

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

DPO

DPO optimizes policy π_θ directly using the following loss:

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

The squared loss version of DPO

Optimizing Logistic loss can lead to overfit, we can use square loss (e.g., regression) instead:

$$\arg\max_{\theta} \sum_{x,\tau,\tau',z} \left(\beta \left(\ln \frac{\pi_\theta(\tau \mid x)}{\pi_{ref}(\tau \mid x)} - \ln \frac{\pi_\theta(\tau' \mid x)}{\pi_{ref}(\tau' \mid x)} \right) - z \right)^2$$

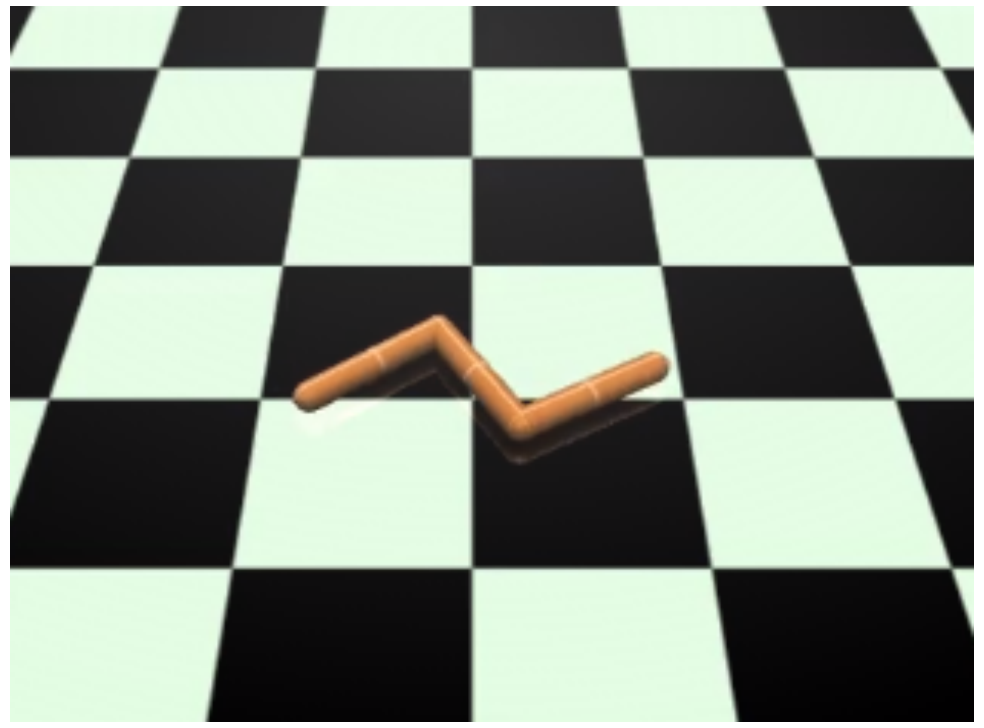
Applying DPO on the openAI gym tasks (next PA)

Q: But these tasks have unknown transition $\rho(\tau) = \prod_h \pi(a_h \mid s) P(s_{h+1} \mid s_h, a_h)$, can we still do DPO?

Note that we only care about trajectory density ratio, so transition cancels out!

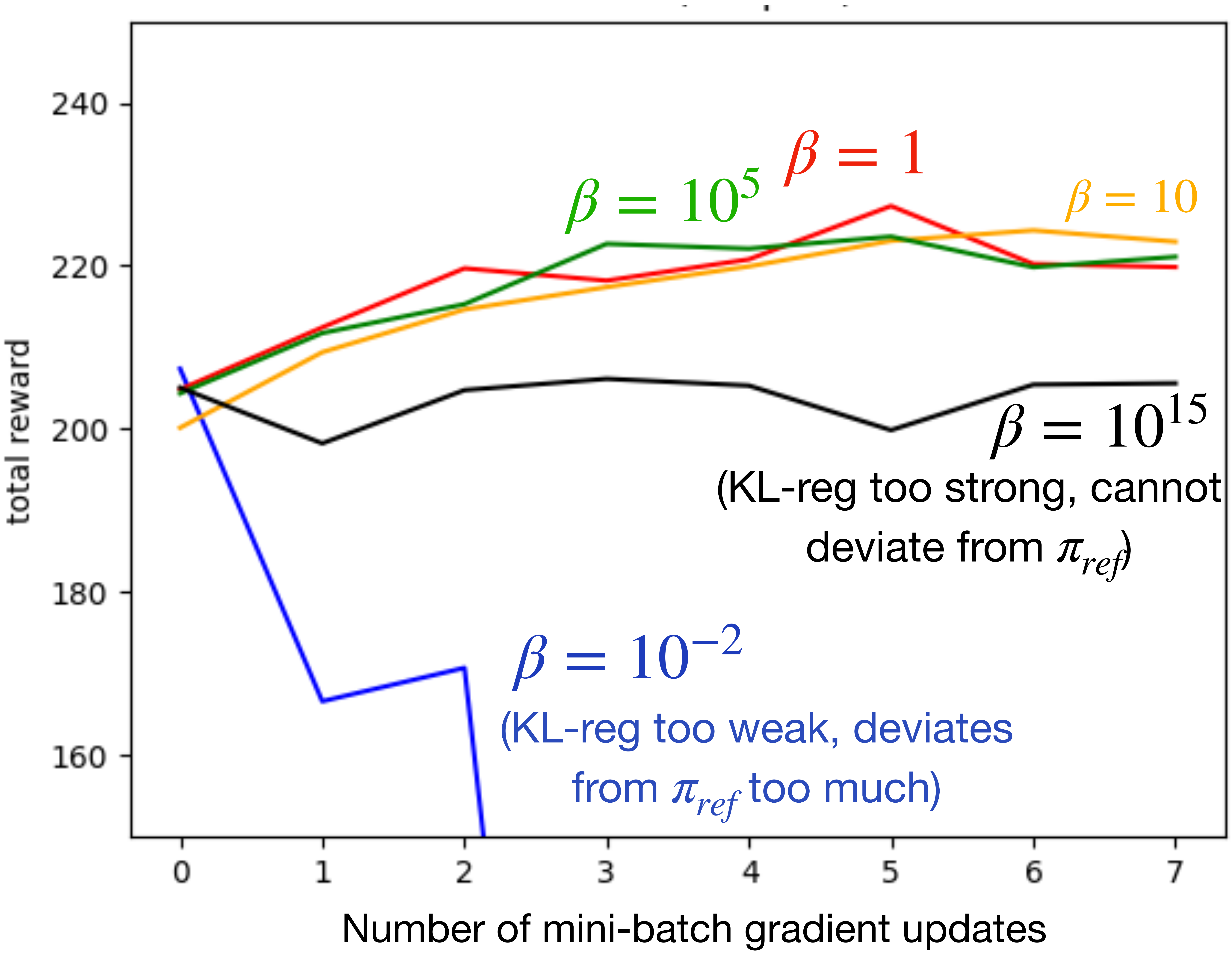
$$\ln \frac{\rho_\pi(\tau)}{\rho_{\pi_{ref}}(\tau)} = \ln \prod_h \frac{\pi(a_h \mid s_h)}{\pi_{ref}(a_h \mid s_h)}$$

Swimmer: continuous controll; goal: move forward fast



1. Collect pair of trajs using a π_{ref}; label via the ground truth reward

2. Run DPO (squared loss) w/ different β



Summary

Closed-form solution of the optimal policy of KL-reguarlized RL

DPO reparameterizes the reward difference via policy directly

Plug the reward difference parameterized by policy into the BT-inspired MLE loss to directly optimize policy