Unpacking DPO and PPO

Archit Sharma,
Stanford
(DPO, 2024)

John Schulman,
OpenAI (now
thinking machines
(PPO, 2017)

PPO vs DPO

$$\theta^* = \arg\max_{\theta} \sum_{k=0}^{K-1} \mathbb{E}_{x \sim D_\pi, y \sim \pi_{\theta_k}(y|x)} \left[\sum_{t=1}^{T} \min\left(\frac{\pi_\theta(y_t|x, y_{<t})}{\pi_{\theta_k}(y_t|x, y_{<t})} \hat{A}_t^{(k)}, \mathrm{clip}\left(\frac{\pi_\theta(y_t|x, y_{<t})}{\pi_{\theta_k}(y_t|x, y_{<t})}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t^{(k)} \right) - \beta D_{\mathrm{KL}} \left(\pi_\theta(y|x) || \pi_{\mathrm{ref}}(y|x) \right) \right]$$

$$L_{\mathrm{DPO}}(\pi_\theta; \pi_{\mathrm{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim D} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\mathrm{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\mathrm{ref}}(y_l|x)} \right) \right]$$

PPO example - a puppy on the leash wants to learn new tricks

You want to train a puppy to sit

1. You deal with puppy's old habits (old policy)
2. But you want to teach puppy the new trick (new policy)
3. You reward it with treat
4. Everytime you train you check
    a. Is the new behavior better than puppy's old habits? Is puppy close to sitting?
    b. How big is the change? Is the puppy distracted? Is the puppy jumpy? We would go for clipping in that case.
5. You could think of KL divergence as puppy on the leash and the variables as tension on the leash :)

In GRPO there's many puppy in the leash

1. There's no personal trainer - no puppy gets individual value scores.
2. We only rely on group reward. If group does well, then we take into the account of the score

   1/(num puppies) * (score of all puppies)
3. Less personal training overhead, but less precision - thus more compute efficient since we train same number of puppies in both PPO and GRPO.

DPO - just make my puppy better

Reference Puppy

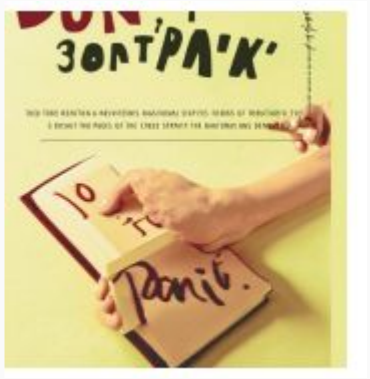
My Puppy

If my puppy isn’t
prefered over reference puppy

The puppy does not get a treat until it is prefered over reference puppy :(

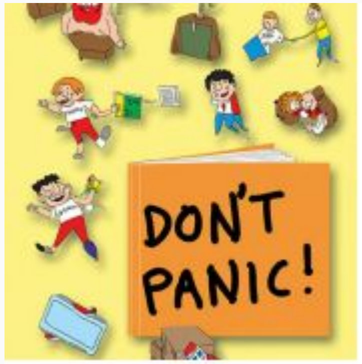
Prompt: a book with words 'Don't panic' written on it

The probability of a good outcome 😔



Dispreferred completion by model

The probability of a good outcome 🥰



Preferred completion by model

The conclusions from the paper

1. Quality of the preferences matter more than the quality of actual generations considered.
2. PPO outperforms DPO given the same models and same initial data.
3. Increasing reward model size or dataset size used to train the reward model results in improved reward model performance on benchmarks directly testing reward model performance. Though marginal improvements in some tests. (Conclusion: improvements in reward models result in surprisingly small improvements in overall downstream performance)
4. Using unlabelled prompts that better match the test setting during policy can further improve model performance in domain-specific settings (e.g., when focusing on improving math performance), but has limited to no effect when targeting overall performance

Intuition for PPO

Two goals to look for:

1. Take the biggest possible improvement step on a policy using data we currently have.
2. Don’t step too far that we accidently cause the performance collapse.

PPO may have very similar explanation to GRPO

Few things we make sure when we consider writing PPO policy:

1. We want to improve the policy, but not too drastically. We would want the old policy to produce the new policy.
2. We compare how the new policy differs from the old one via a ratio.
3.

Intuition for DPO - Direct Preference Optimization

Compare the log likelihood ratios of the preferred and dispreferred completions

Is based on the following question - if I prefer the response y(w) over y(r) - can I bake this preference directly into my model?

This way DPO is:

Stable
Computationally light
Easier to implement

Winning response

Loser Response

Training details

Implementation details. PPO comes with many implementation details. They made a simplistic implementation that results in stable training. Notably:

- They initialize the value model from the reward model. This follows from InstructGPT. Some other implementations initialize from the SFT model or the base model, while replacing the LM head with a regression head.
- For truncated completions, they set the reward to a large negative number (e.g., -10.0), which is referred to as the EOS trick.

Training details

- They do not perform normalization on the rewards. This follows from AlpacaFarm. Although reward models trained under different settings can have very different output ranges, they found their PPO experiments quite robust to such variation.
- They do not whiten the step-level rewards within each batch. They do whiten the step-level advantages within each batch, following other implementations.
- They use a fixed KL penalty coefficient. The original PPO algorithm has an adaptive KL controller, but most recent implementations have moved away from this.