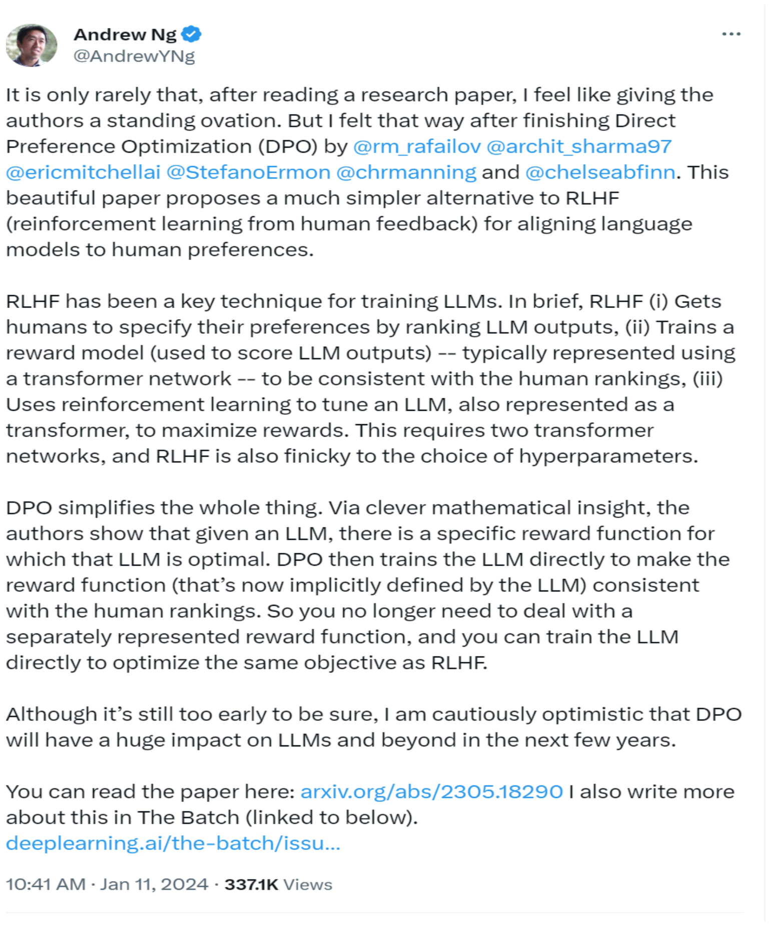
Reading Group 1/14/2023 Direct Preference Optimization

Paper

- https://arxiv.org/abs/2305.18290

Inspiration behind this reading



Andrew Ng
@AndrewYNg

It is only rarely that, after reading a research paper, I feel like giving the authors a standing ovation. But I felt that way after finishing Direct Preference Optimization (DPO) by @rm_rafailov @archit_sharma97 @ericmitchellai @StefanoErmon @chrmanning and @chelseabfinn. This beautiful paper proposes a much simpler alternative to RLHF (reinforcement learning from human feedback) for aligning language models to human preferences.

RLHF has been a key technique for training LLMs. In brief, RLHF (i) Gets humans to specify their preferences by ranking LLM outputs, (ii) Trains a reward model (used to score LLM outputs) -- typically represented using a transformer network -- to be consistent with the human rankings, (iii) Uses reinforcement learning to tune an LLM, also represented as a transformer, to maximize rewards. This requires two transformer networks, and RLHF is also finicky to the choice of hyperparameters.

DPO simplifies the whole thing. Via clever mathematical insight, the authors show that given an LLM, there is a specific reward function for which that LLM is optimal. DPO then trains the LLM directly to make the reward function (that's now implicitly defined by the LLM) consistent with the human rankings. So you no longer need to deal with a separately represented reward function, and you can train the LLM directly to optimize the same objective as RLHF.

Although it's still too early to be sure, I am cautiously optimistic that DPO will have a huge impact on LLMs and beyond in the next few years.

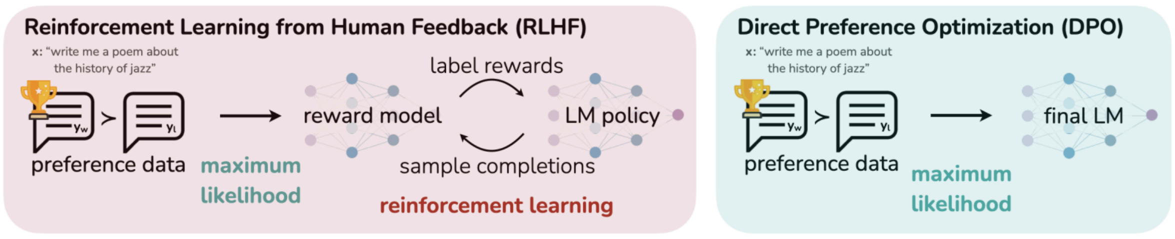
You can read the paper here: arxiv.org/abs/2305.18290 I also write more about this in The Batch (linked to below).
deeplearning.ai/the-batch/issu...

10:41 AM · Jan 11, 2024 · 337.1K Views

Introduction

- "Similarly, we might want our language model to be aware of a common misconception believed by 50% of people, but we certainly do not want the model to claim this misconception to be true in 50% of queries about it!"
  - Tldr says that current rlhf methods instill unwanted behaviors in LLMs
- Claims to improve upon RLHF using a simple binary cross entropy objective

- They show how to directly optimize a language model to adhere to human preferences, without explicit reward modeling or reinforcement learning.


Reinforcement Learning from Human Feedback (RLHF)
x: "write me a poem about the history of jazz"
y_w > y_l
preference data
maximum likelihood
reward model
label rewards
LM policy
sample completions
reinforcement learning
Direct Preference Optimization (DPO)
x: "write me a poem about the history of jazz"
y_w > y_l
preference data
maximum likelihood
final LM

-
- Similar to RLHF in that it is reward maximization with KL divergence constraint
    - Claims DPO is simpler to implement
- Mathematical stuff
    - DPO update increases relative log probability of preferred to dispreferred responses
    - Dynamic per example importance weight
        - This prevents model degeneration
- Author explained it as the closed form
- Claims it is just as effective as existing methods, like RLHF

RLAIF

- https://arxiv.org/abs/2309.00267

Related Work

- Instruction-tuning can improve performance
    - Methods like these first optimize a neural network reward with a preference dataset
    - then fine-tune a language model to maximize the given reward using reinforcement learning algorithms
- Learning from preferences has also been studied in contextual bandits
    - Contextual bandit learning using preferences or rankings of actions
        - Aka contextual dueling bandit
- Preference based RL
    - Learns from binary preferences generated by an unknown scoring function

SFT: RLHF typically begins by fine-tuning a pre-trained LM with supervised learning on high-quality data for the downstream task(s) of interest (dialogue, summarization, etc.), to obtain a model π^{SFT}.

RLHF pipeline

- Has 3 phases
  - 1. Supervised Fine tuning
  - 2. Preference sampling and reward learning
  - 3. RL optimization

Reward Modelling Phase: In the second phase the SFT model is prompted with prompts x to produce pairs of answers $(y_1, y_2) \sim \pi^{\text{SFT}}(y \mid x)$. These are then presented to human labelers who express preferences for one answer, denoted as $y_w \succ y_l \mid x$ where y_w and y_l denotes the preferred and dispreferred completion amongst (y_1, y_2) respectively. The preferences are assumed to be generated by some latent reward model $r^*(y, x)$, which we do not have access to. There are a number of approaches used to model preferences, the Bradley-Terry (BT) [5] model being a popular choice (although more general Plackett-Luce ranking models [30, 21] are also compatible with the framework if we have access to several ranked answers). The BT model stipulates that the human preference distribution p^* can be written as:

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp\left(r^*(x, y_1)\right)}{\exp\left(r^*(x, y_1)\right) + \exp\left(r^*(x, y_2)\right)}. \tag{1}$$

Assuming access to a static dataset of comparisons $\mathcal{D} = \{x^{(i)}, y_w^{(i)}, y_l^{(i)}\}_{i=1}^{N}$ sampled from p^*, we can parametrize a reward model $r_\phi(x, y)$ and estimate the parameters via maximum likelihood. Framing the problem as a binary classification we have the negative log-likelihood loss:

$$\mathcal{L}_R(r_\phi, \mathcal{D}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\big[\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))\big] \tag{2}$$

where σ is the logistic function. In the context of LMs, the network $r_\phi(x, y)$ is often initialized from the SFT model $\pi^{\text{SFT}}(y \mid x)$ with the addition of a linear layer on top of the final transformer layer that produces a single scalar prediction for the reward value [49]. To ensure a reward function with lower variance, prior works normalize the rewards, such that $\mathbb{E}_{x,y\sim\mathcal{D}}\left[r_\phi(x, y)\right] = 0$ for all x.

RL Fine-Tuning Phase: During the RL phase, we use the learned reward function to provide feedback to the language model. In particular, we formulate the following optimization problem

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)}\big[r_\phi(x, y)\big] - \beta\mathbb{D}_{\text{KL}}\big[\pi_\theta(y \mid x) \mid\mid \pi_{\text{ref}}(y \mid x)\big] \tag{3}$$

where β is a parameter controlling the deviation from the base reference policy π_{ref}, namely the initial SFT model π^{SFT}. In practice, the language model policy π_θ is also initialized to π^{SFT}. The added constraint is important, as it prevents the model from deviating too far from the distribution on which the reward model is accurate, as well as maintaining the generation diversity and preventing mode-collapse to single high-reward answers. Due to the discrete nature of language generation, this objective is not differentiable and is typically optimized with reinforcement learning. The standard approach [49, 38, 1, 26] has been to construct the reward function $r(x, y) = r_\phi(x, y) - \beta(\log \pi_\theta(y \mid x) - \log \pi_{\text{ref}}(y \mid x))$, and maximize using PPO [37].

Direct preference optimization

- Doesnt learn a reward and optimize via RL
- Instead, uses reward model parameterization that lets us use the closed form of the optimal policy
    - Doesn't use rl in a loop

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)}\big[r_\phi(x, y)\big] - \beta\mathbb{D}_{\text{KL}}\big[\pi_\theta(y \mid x) \mid\mid \pi_{\text{ref}}(y \mid x)\big] \tag{3}$$

Gets derived into

$$\pi_r(y \mid x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y \mid x)\exp\left(\frac{1}{\beta}r(x, y)\right), \tag{4}$$

Tldr

- Learning from preferences is scalable and powerful for training intelligent language models
- introducing DPO
    - For training language models from preferences **without reinforcement learning**

    - Rather than coercing the preference learning problem into a standard RL setting in order to use off-the-shelf RL algorithms, DPO identifies a mapping between language model policies and reward functions that enables training a language model to satisfy human preferences directly
    - with a simple cross-entropy loss, without reinforcement learning or loss of generality.
    - With virtually no tuning of hyperparameters, DPO performs similarly or better than existing RLHF algorithms, including those based on PPO; DPO thus meaningfully reduces the barrier to training more language models from human preferences

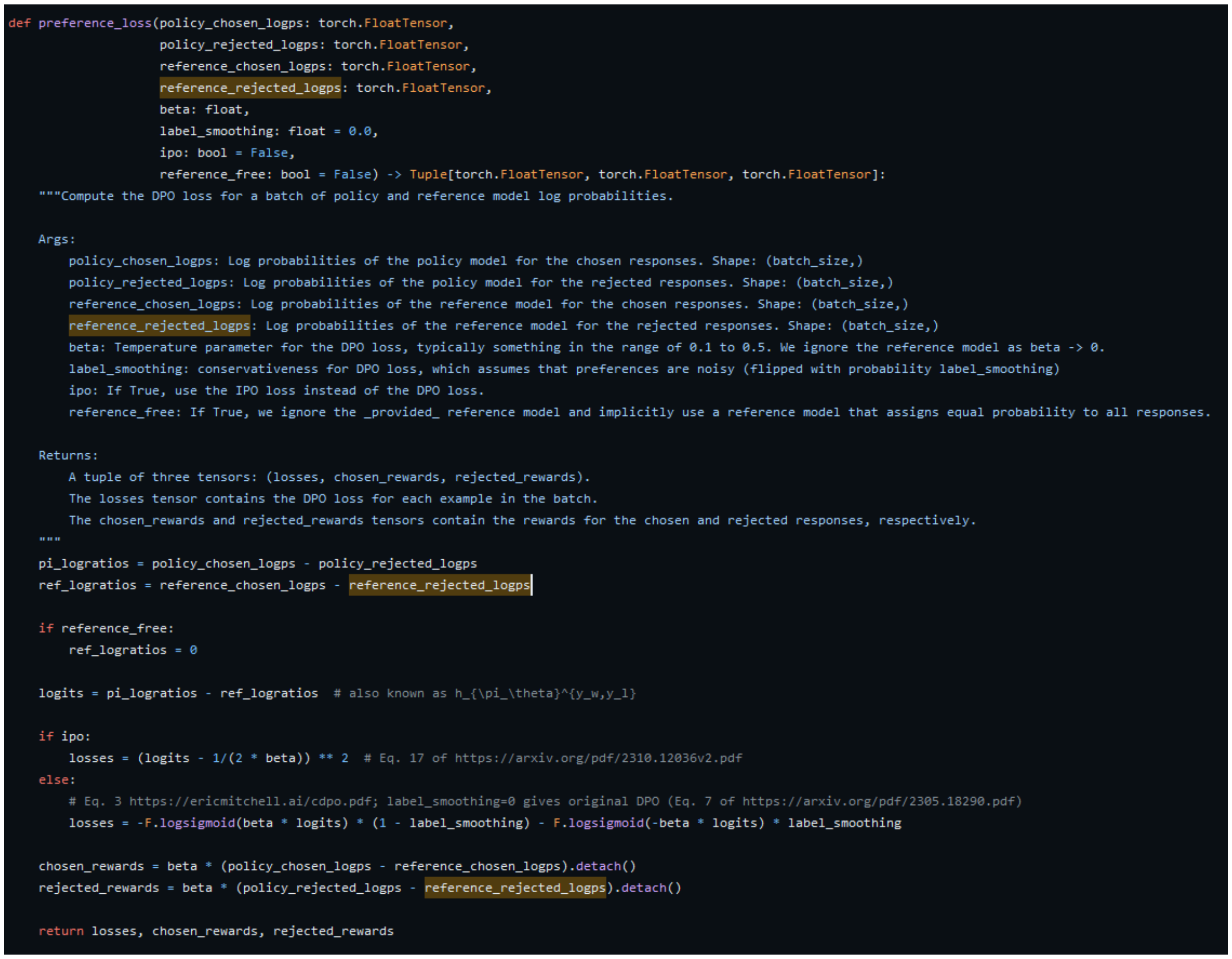
https://towardsdatascience.com/understanding-kl-divergence-f3ddc8dff254

Limitations and future work

- Does DPO generalize out of distribution compared with learning form an explicit reward function ?
    - Suggests that DPO **can** generalize similarly to PPO based models, but more study is needed
    - Can training with self labeling form the dpo policy make effective use of unlabeled prompts ?
- How does reward over optimization manifest in the DPO setting and is slight decrease in performance in figure 3 an instance of it ?
- Also they want to scale up DPO to SOTA models
- Does dpo work with training models in other modalities?

Code

- https://github.com/eric-mitchell/direct-preference-optimization
- Datasets: https://github.com/eric-mitchell/direct-preference-optimization/blob/main/preference_datasets.py

- 

```python
def preference_loss(policy_chosen_logps: torch.FloatTensor,
                    policy_rejected_logps: torch.FloatTensor,
                    reference_chosen_logps: torch.FloatTensor,
                    reference_rejected_logps: torch.FloatTensor,
                    beta: float,
                    label_smoothing: float = 0.0,
                    ipo: bool = False,
                    reference_free: bool = False) -> Tuple[torch.FloatTensor, torch.FloatTensor, torch.FloatTensor]:
    """Compute the DPO loss for a batch of policy and reference model log probabilities.

    Args:
        policy_chosen_logps: Log probabilities of the policy model for the chosen responses. Shape: (batch_size,)
        policy_rejected_logps: Log probabilities of the policy model for the rejected responses. Shape: (batch_size,)
        reference_chosen_logps: Log probabilities of the reference model for the chosen responses. Shape: (batch_size,)
        reference_rejected_logps: Log probabilities of the reference model for the rejected responses. Shape: (batch_size,)
        beta: Temperature parameter for the DPO loss, typically something in the range of 0.1 to 0.5. We ignore the reference model as beta -> 0.
        label_smoothing: conservativeness for DPO loss, which assumes that preferences are noisy (flipped with probability label_smoothing)
        ipo: If True, use the IPO loss instead of the DPO loss.
        reference_free: If True, we ignore the _provided_ reference model and implicitly use a reference model that assigns equal probability to all responses.

    Returns:
        A tuple of three tensors: (losses, chosen_rewards, rejected_rewards).
        The losses tensor contains the DPO loss for each example in the batch.
        The chosen_rewards and rejected_rewards tensors contain the rewards for the chosen and rejected responses, respectively.
    """
    pi_logratios = policy_chosen_logps - policy_rejected_logps
    ref_logratios = reference_chosen_logps - reference_rejected_logps

    if reference_free:
        ref_logratios = 0

    logits = pi_logratios - ref_logratios  # also known as h_{\pi_\theta}^{y_w,y_l}

    if ipo:
        losses = (logits - 1/(2 * beta)) ** 2  # Eq. 17 of https://arxiv.org/pdf/2310.12036v2.pdf
    else:
        # Eq. 3 https://ericmitchell.ai/cdpo.pdf; label_smoothing=0 gives original DPO (Eq. 7 of https://arxiv.org/pdf/2305.18290.pdf)
        losses = -F.logsigmoid(beta * logits) * (1 - label_smoothing) - F.logsigmoid(-beta * logits) * label_smoothing

    chosen_rewards = beta * (policy_chosen_logps - reference_chosen_logps).detach()
    rejected_rewards = beta * (policy_rejected_logps - reference_rejected_logps).detach()

    return losses, chosen_rewards, rejected_rewards
```