DPO-style alignment

michele.garibbo

January 2025

1 Summary

Here I gather a collection of notes on DPO-style methods for the alignment of large language models (LLMs)

2 DPO original paper

(Rafailov et al., 2024)

- **What** provides novel method to align pre-trained LLMs using preference data.
- **Why** Standard alignment methods based on RLHF (e.g., PPO) require to first learn an explicit reward function, which often leads to instability (e.g., the agent adversarially exploits the uncertainty in the learned reward function), while being computationally expensive.
- **How** introduces a method which allows you to align a LLM directly from a preference dataset, without having to learn an explicit reward function.

2.1 Background on classical RLHF

The general RLHF pipeline (e.g., PPO) comprises of the 3 following steps,

1. For a set of prompts, $\{x^i\}_{i=0}^N$, use a (SFT) pre-trained LLM, π^{SFT}, to generate pairs of answers for each x^i prompt, $(y_1^i, y_2^i) \sim \pi^{\text{SFT}}(y \mid x^i)$. Pass the generated data to human labelers, who express a preference for each answer given the corresponding prompt. This gives you a preference dataset, $D = \{x^i, y_{\text{w}}^i, y_{\text{l}}^i\}_{i=0}^N$, where y_{w}^i denotes the preferred/winner answer over the 'loser' one, y_{l}^i for each prompt x^i.

2. Train an explicit reward model to capture the preference dataset. To do so, you first need to make an assumption on how the preferences were generated by the human labelers. A popular approach for pairs of preferences is to assume the (human) preferences were generated according to the Bradely-Terry

model,

$$p(y_1^i \prec y_2^i \mid x^i) = \frac{\exp\{r(x^i, y_1^i)\}}{\exp\{r(x^i, y_1^i)\} + \exp\{r(x^i, y_2^i)\}} \tag{1}$$

Namely, for a prompt x^i, the BT model assumes the probability that a human prefers y_1^i over y_2^i is proportional to some latent reward function, r (i.e., which is supposed to measure 'how good' the answer is for that prompt). This model assumption is key because it allows you to **link some preference data**, $D = \{x^i, y_\text{w}^i, y_\text{l}^i\}_{i=0}^N$, **to a reward value**, r, which we can try to learn with a reward function. Indeed, without this assumption, it is not directly clear how you would map pairs of preference data to a reward value. I think it is important to stress this point because even in standard RLHF where you learn an explicit scalar reward function, if you learned this function based on binary preference data, you are **still relying on the BT model being a good fit** for the preference data (note, if you have multiple raked preference data, then, you can use the Plackett-Luce model).

Now that you have assumed a specific distribution over the preference dataset, $D = \{x^i, y_\text{w}^i, y_\text{l}^i\}_{i=0}^N$, you can do maximum likelihood to find the latent reward function r that maximizes the probability of the observed preferences under the assumed distribution (i.e., under the BT model in this case). Specifically, you assume the latent reward function, r, to be parametrised by a reward model, r_ϕ, for which you want to find the parameter ϕ that best described the observed preferences. For our dataset, $D = \{x^i, y_\text{w}^i, y_\text{l}^i\}_{i=0}^N$, we can re-write the maximum likelihood objective in terms of an expectation over the entire dataset,

$$\arg\max_\phi \; \mathbb{E}_{(x^i, y_\text{w}^i, y_\text{l}^i) \sim D} \left[p(y_\text{w}^i \prec y_\text{l}^i \mid x^i) \right] \tag{2}$$

$$= \arg\max_\phi \; \mathbb{E}_{(x^i, y_\text{w}^i, y_\text{l}^i) \sim D} \left[\frac{\exp\{r_\phi(x^i, y_\text{w}^i)\}}{\exp\{r_\phi(x^i, y_\text{w}^i)\} + \exp\{r_\phi(x^i, y_\text{l}^i)\}} \right] \tag{3}$$

Basically, we want to find the parameters, ϕ of the reward model such that it maximizes the probability of all $\{y_\text{w}^i\}^N$ being the preferred answer given the corresponding prompt, $\{x^i\}^N$ (note, since p is a valid distribution, maximizing the probability of p for y_w implies reducing the probability of y_l).

We can re-arrange things in eq. 3 (see Sec 5.1) and express it in terms of a 'more-familiar' cross-entropy loss by computing the log probabilities (i.e., log likelihood),

$$\mathcal{L}(\phi) = -\mathbb{E}_{(x^i, y_\text{w}^i, y_\text{l}^i) \sim D} \left[\log \sigma \left(r_\phi(x^i, y_\text{w}^i) - r_\phi(x^i, y_\text{l}^i) \right) \right] \tag{4}$$

where σ denotes the sigomoid function.

3. After having learned an explicit reward function, r_ϕ, we can finally use it to align our pre-trained LLM, π_θ^SFT, to our preferences. Specifically, we want to change the pre-trained LLM such that its outputs (answers) maximize

the learned reward function. In standard RLHF approaches (e.g., PPO), we typically do this in terms of the following loss,

$$\mathcal{L}(\theta) = \\ - E_{x^i \sim D; y^i \sim \pi_\theta(y|x^i)} \left[r_\phi(x^i, y^i) \right] + \beta \; D_{\text{KL}} \left(\pi_\theta(y \mid x) \; || \; \pi^{\text{SFT}}(y \mid x) \right) \tag{5}$$

where θ denote the parameter of the LLM (not the reward function!) and the KL-divergence ensures that during alignment, the LLM does not depart 'too much' from the original pre-trained model (e.g., by doing reward hacking, just to maximise r_ϕ) with the hyper-parameter β controlling how much we allow it to depart to get higher rewards.

In practice, for each prompt x_i, the answer y_i is sampled from the LLM, $y^i \sim \pi_\theta(y \mid x^i)$, which means we cannot directly differentiate through r_ϕ to update the LLM parameters, θ (i.e., more formally, the expectation over the loss depends on the parameters we are trying to optimize, requiring to use some stochastic gradient computation approach). Standard RLHF approaches solve this stochastic gradient estimation issue by using the 'log-trick' (aka REINFORCE-style update), where the famous PPO just represents a more sophisticated version of it. Note, the KL-constraint is typically built into the scalar rewards used in the REINFORCE-style update,

$$r(x, y) = r_\phi(x, y) - \beta(\log \pi_\theta(y \mid x) - \log \pi^{\text{SFT}}(y \mid x)) \tag{6}$$

where the log terms arise from the using 'log-trick'.

2.2 DPO

The key idea behind DPO is to show that the BT model (eq. 1) used to align our LLM to the preference data can be re-written in terms of the LLM ouputs only, π_θ, with no explicit dependency on the (latent) reward function, r (i.e., bypassing the need of an explicit reward function, r_ϕ). The first and most important step is to show the KL-constrained reward loss optimized by RLHF (eq. 5) is minimized by the following (optimal) policy, π^* (derivation not included here, see appendix A.1 of the DPO paper),

$$\pi^*(y \mid x) = \frac{1}{Z(x)} \; \pi^{\text{SFT}}(y \mid x) \exp\{\frac{1}{\beta} r(x, y)\} \tag{7}$$

where π^{SFT} represents some reference policy we don't want to diverge too much from, while $Z()$ denotes the partition/normalising function (i.e., making π^* sum to 1),

$$Z(x) = \sum_y \pi^{\text{SFT}}(y \mid x) \exp\{\frac{1}{\beta} r(x, y)\} \tag{8}$$

Next, we can take the logarithm of both side of eq. 7 and re-arrange things to get,

$$r(x, y) = \beta \ \log \frac{\pi^*(y \mid x)}{\pi^{\text{SFT}}(y \mid x)} \ + \ \beta \ \log Z(x) \tag{9}$$

This step allowed us to represent the (latent) reward function in terms of the partition function and the (optimal) policy for the KL-constrained reward loss (eq. 5). **Note**, this derivation (including the derivation of eq. 7) does not depend on any preference model assumption, it is simply the optimal solution to the RLHF objective (eq. 5). Next, we introduce the preference model to model the preference data by plugin this reward parameterization into the BT model latent reward function (eq. 1),

$$p(y_1 \prec y_2 \mid x) = \frac{\exp\{\beta \ \log \frac{\pi^*(y_1|x)}{\pi^{\text{SFT}}(y_1|x)} \ + \ \beta \ \log Z(x)\}}{\exp\{\beta \ \log \frac{\pi^*(y_1|x)}{\pi^{\text{SFT}}(y_1|x)} \ + \ \beta \ \log Z(x)\} + \exp\{\beta \ \log \frac{\pi^*(y_2|x)}{\pi^{\text{SFT}}(y_2|x)} \ + \ \beta \ \log Z(x)\}}$$

and we can re-arrange this into (see sec 5.1),

$$= \sigma\left(\beta \ \log \frac{\pi^*(y_1 \mid x)}{\pi^{\text{SFT}}(y_1 \mid x)} \ - \beta \log \frac{\pi^*(y_2 \mid x)}{\pi^{\text{SFT}}(y_2 \mid x)}\right) \tag{10}$$

The amazing part of this re-arrangement is that the partition function, $Z(x)$, canceled out since it is the same value for both y_1 and y_2 (i.e., sum across all answers y given prompt x). This gives us exactly what we want: the BT-based preference distribution now depends on LLM outputs only, with no explicit dependency on a latent reward function, r (i.e., we no longer need an explicit reward model).

Therefore, given a preference dataset, $D = \{x^i, y_{\text{w}}^i, y_{\text{l}}^i\}_{i=0}^N$, we can now define a maximum likelihood objective for this BT-based preference distribution, which we directly optimize for the parameters of the LLM we want to align, θ (i.e., by substituting our LLM, π_θ, to the optimal policy, π^*).

$$\mathcal{L}^{\text{DPO}}(\theta) = -\mathbb{E}_{(x^i, y_{\text{w}}^i, y_{\text{l}}^i) \sim D}\left[\log p(y_{\text{w}}^i \prec y_{\text{l}}^i \mid x^i)\right] \tag{11}$$

$$- \mathbb{E}_{(x^i, y_{\text{w}}^i, y_{\text{l}}^i) \sim D}\left[\log \sigma\left(\beta \ \log \frac{\pi_\theta(y_{\text{w}} \mid x)}{\pi^{\text{SFT}}(y_{\text{w}} \mid x)} \ - \beta \log \frac{\pi_\theta(y_{\text{l}} \mid x)}{\pi^{\text{SFT}} a(y_{\text{l}} \mid x)}\right)\right] \tag{12}$$

This follows the same derivation as eq. 4, where the objective was represented in terms of a log likelihood loss.

Through a simple derivation, it can be shown the gradient of the DPO loss takes

the following form,

$$\nabla_\theta \mathcal{L}^{\text{DPO}} = \tag{13}$$

$$-\beta\ \mathbb{E}_{(x^i, y_{\text{w}}^i, y_{\text{l}}^i)\sim D}\left[\underbrace{\sigma\left(\hat{r}_\theta(x, y_{\text{l}}) - \hat{r}_\theta(x, y_{\text{w}})\right)}_{\text{Importance weight}}\left(\overbrace{\nabla_\theta \log \pi_\theta(y_{\text{w}} \mid x)}^{\substack{\text{Increase}\\ \text{likelihood of } y_{\text{w}}}} - \overbrace{\nabla_\theta \log \pi_\theta(y_{\text{l}} \mid x)}^{\substack{\text{Decrease}\\ \text{likelihood of } y_{\text{l}}}}\right)\right]$$

where,

$$\hat{r}_\theta(x, y) = \beta\ \log \frac{\pi_\theta(y \mid x)}{\pi^{\text{SFT}}(y \mid x)}$$

DPO outline: DPO trains offline based on a preference dataset, $D = \{x^i, y_{\text{w}}^i, y_{\text{l}}^i\}_{i=0}^N$. If you are using an existing preference dataset to align your LLM, instead of creating the dataset yourself based on your LLM outputs and subsequent human labeling, you may encounter a distributional shift problem. This is because the DPO update requires having access to the reference model, π^{SFT}, which generated the training examples $D = \{x^i, y_{\text{w}}^i, y_{\text{l}}^i\}_{i=0}^N$ (or at least to its predicted probabilities for all training data, $\pi^{\text{SFT}}(y \mid x)$). However, in case of a public dataset we likely do not have access to the model that generated the data. If we just set our 'frozen' LLM as π^{SFT} in the DPO update, the training data may be out of distribution for our model, giving wild probability values for the training data. To overcome this issue, we should first supervised fine-tune our LLM to the preferred completions, (x, y_{w}) (i.e., $\pi^{\text{SFT}} = \arg\max_\pi \mathbb{E}[\log \pi(y_{\text{w}} \mid x)]$) and then, set this SFT version of our LLM as the reference model (π^{SFT}) in the DPO update.

2.3 Extra: DPO with rankings of answers

In the previous section, we assumed for each prompt x to have a pair of answers (i.e., y_1 vs y_2) and we modeled the probability of one answer being preferred over the other, $p(y_1 \prec y_2 \mid x)$, using the BT model. What if, for each a prompt x, we have k possible answers (i.e., $(y_1, \cdots, y_k)$). In that case, we should assume a 'ranking' over the k answers and model the probability of one ranking being preferred over the others. The 'ranking space' is constructed over all the possible permutations of $y_1, \cdots, y_k$, since each permutation gives you a different order of preferred answers (i.e., a different ranking). Note, here the subscript k on each answer y is used to denote a different answer not to indicate an order (ranking) of preferences (i.e., y_1 is not necessarily preferred to y_2 and so on).

Plackett-Luce model: We can model a distribution over the 'ranking space' (i.e., over possible raking) using the Plackett-Luce (PL) preference model:

$$p(\tau \mid y_1, \cdots, y_k, x) = \prod_{k=1}^{K} \frac{\exp\{r(x, y_{\tau(k)})\}}{\sum_{j=k}^{K} \exp\{r(x, y_{\tau(j)}\}} \tag{14}$$

where τ denotes a specific ranking (permutation) of $y_1^i, \cdots, y_k^i$. Here, I don't include any intuition on why the PL model provides a valid distribution over rankings, but the general idea behind it is to consider the generation of each ranking as a sequential decision process. As shown below, this key property allows us to derive a DPO update for ranking of answers.

DPO with the PL model: Similarly to the BT mode, the PL model assumes the ranking of answers is generated by a latent reward model, r, where an answer is preferred with a probability proportional to this latent reward (i.e., the most likely ranking according to PL will be the one where the answers are ordered according to their later reward value). As a result, we can again substitute eq. 9 into the reward terms of the PL model to represent these rewards in terms of the LLM outputs only, with no explicit dependency on a reward model:

$$p(\tau \mid y_1^i, \cdots, y_k^i, x^i) = \prod_{k=1}^{K} \frac{\exp\{\beta \log \frac{\pi^*(y_{\tau(k)}|x)}{\pi^{\text{SFT}}(y_{\tau(k)}|x)}\}}{\sum_{j=k}^{K} \exp\{\beta \log \frac{\pi^*(y_{\tau(j)}|x)}{\pi^{\text{SFT}}(y_{\tau(j)}|x)}\}} \tag{15}$$

where the partition function, $Z(x)$, canceled out again.

Given a a preference dataset of rankings, $D = \{x^i, y_{w_1}^i, \cdots, y_{w_k}^i\}_{i=0}^N$, where we used the subscript w to denote the ranking, we can define a maximum likelihood objective of this PL-based distribution in terms of the LLM, π_θ, we want to align to the preference dataset. Again, we express this maximum likelihood objective in terms of the expected negative log likelihood over the dateset,

$$\mathcal{L}^{\text{DPO}}(\theta) = -\mathbb{E}_{(x^i, y_{w_1}^i, \cdots, y_{w_k}^i) \sim D} \left[\log p(\tau = y_{w_1}^i, \cdots, y_{w_k}^i \mid y_1^i, \cdots, y_k^i, x^i)\right] \tag{16}$$

$$= -\mathbb{E}_{(x^i, y_{w_1}^i, \cdots, y_{w_k}^i) \sim D} \left[\log \prod_{k=1}^{K} \frac{\exp\{\beta \log \frac{\pi^*(y_{w_k}^i|x^i)}{\pi^{\text{SFT}}(y_{w_k}^i|x^i)}\}}{\sum_{j=k}^{K} \exp\{\beta \log \frac{\pi^*(y_{w_j}^i|x^i)}{\pi^{\text{SFT}}(y_{w_j}^i|x^i)}\}}\right] \tag{17}$$

We can further simplify this to (which is helpful for the weighted DPO derivation),

$$= -\mathbb{E}_{(x^i, y_{w_1}^i, \cdots, y_{w_k}^i) \sim D} \left[\sum_{k=1}^{K} \log \frac{\exp\{\beta \log \frac{\pi^*(y_{w_k}^i|x^i)}{\pi^{\text{SFT}}(y_{w_k}^i|x^i)}\}}{\sum_{j=k}^{K} \exp\{\beta \log \frac{\pi^*(y_{w_j}^i|x^i)}{\pi^{\text{SFT}}(y_{w_j}^i|x^i)}\}}\right] \tag{18}$$

$$= -\mathbb{E}_{(x^i, y_{w_1}^i, \cdots, y_{w_k}^i) \sim D} \left[\sum_{j=k}^{K} \beta \log \frac{\pi^*(y_{w_k}^i \mid x^i)}{\pi^{\text{SFT}}(y_{w_k}^i \mid x^i)} - \log \sum_{j=k}^{K} \exp\{\beta \log \frac{\pi^*(y_{w_j}^i \mid x^i)}{\pi^{\text{SFT}}(y_{w_j}^i \mid x^i)}\}\right] \tag{19}$$

Namely, we can change the LLMs parameters θ, such that given a prompt, x^i, the LLM probability of generating the answers according to their corresponding ranking, $p(\tau = (y_{w_1}^i, \cdots, y_{w_k}^i) \mid y_1^i, \cdots, y_k^i, x^i)$, is maximized.

2.4 Key DPO insights

DPO still estimates a reward function, albeit it does so implicitly through the LLM outputs. This is evident in eq. 9, which explicitly defines a reward parametrization. When assuming a BT/PL model, this reward parametrization takes the following form,

$$r(x, y) = \beta \ \log \frac{\pi^*(y \mid x)}{\pi^{\text{SFT}}(y \mid x)} \tag{20}$$

where the BT/PL model assumption allowed to eliminate the partition function Z from eq. 9. In particular, this reward parametrization enables DPO to merge the two RLHF steps into a single (offline) one (i.e., 1. Use pairwise rankings of responses to train a reward model using some loss function and 2. Use trained reward and online RL (PPO) to improve the model).

This point about reward re-parametrization represents a key difference between standard RLHF approaches (PPO) and DPO. Specifically, DPO optimizes π_θ equivalently to the (maximum likelihood) reward optimization objective used by RLHF to learn an explicit reward model (eq. 4) but, with the key difference that DPO optimizes this objective under the reward parametrization (change of variable) in (eq 20). [1] Therefore, to compare DPO and standard RLHF, we need to ask whether DPO reward parametrization (eq. 20) is as good as classical (RLHF) regression-based ones, in terms of generalization/robustness. Interestingly, one of the DPO main authors, Rafael Rafailov, argued that the DPO reward parametrization may even be better than classical regression-based ones. From an inverse RL perspective, any policy defines a reward function. Therefore, by parametrizing the reward in terms of the policy, DPO is actually 'fine-tuning' a reward function rather than training one from scratch. Reporting his actual words,

> "By training a regression-based reward, we need to connect the inner knowledge of the model to a scalar value with limited data (compared to pre-training). With DPO we can directly tune the IRL implicit reward. This should be better".
>
> (Rafael Rafailov)

2.5 Some personal notes

- For a unique global minima, minimizing the DPO objective (eq. 11) should lead to the same optimal policy π^* that minimizes the KL-constrained reward loss (eq. 5). This should gives us the same aligned LLM as standard RLHF approaches (e.g., PPO) that converge to a unique global minima

[1] In the DPO paper (Rafailov et al., 2024), they include proof that this reparameterization does not constrain the class of learned reward models and, allows for the exact recovery of the optimal policy.

of the same loss. Nevertheless, in practice, we never converge to a unique global minima (which often does not exist), leaving a lot of potential differences between the solutions found by DPO and by standard RLHF approaches.

- The DPO objective (eq. 11) looks a lot like performing maximum likelihood on some preferred high quality data, $\{y_w\}^N$, like we do during the SFT phase. However, while SFT does not work well for alignment, leading to forgetting and other issues (knowledge blocking and hallucinations), the DPO update does not lead to the same issue, working well for alignment. Is this only because of the KL constraint ? **No**, not only, in the DPO paper, they report the *importance weight* in the DPO gradient (eq. 13) is crucial for avoiding model degeneration. I guess this is the key 'trick' of DPO, being able to extract an implicit reward weight from the LLM, which is crucial for avoiding the issue SFT has.

2.6 Some thoughts on DPO vs RLHF

Initially, I wondered if the difference between working with discrete preferences vs continuous rewards may not be one of the reasons why standard RLHF (e.g., PPO) tends to work better than DPO. However, I am now sure this followed from misunderstanding how DPO works. Indeed, when **working with preference datasets**, both DPO and RLHF go from discrete (e.g., binary) preferences to a continuous reward signal, since both approaches use the BT/PL model to model the discrete preferences in terms of a continuous (latent) reward (e.g., eq. 4) with the key difference of the reward parametrization discussed in Sec 2.4. There is however a key difference in how this continuous reward signal is estimated.

Standard RLHF builds an explicit model of the continuous value/reward of each data point from the preference dataset. As a result, standard RLHF may be able to leverage the reward model generalization capabilities to improve the alignment process. Although DPO also models the preferences in terms of continuous rewards , it doesn't have access to an explicit reward model for generalization. Conversely, it does 'value/reward' generalization based on the (LLM) policy itself. I guess it would be interesting to investigate whether the reward model or the policy gives you better extrapolation in this preference settings. This point is tightly related to the reward parametrization point discussed in Sec 2.4.

Additionally, I believe there is also the point whether separating vs merging reward model training with policy training plays any important part in the 'learnability' of the optimal policy.

3 Alignment with a continuous reward value

In natural language, we typically work with preference dataset, where the data is already organized in term of preferred responses (e.g., human labelers ranking

several LLM responses to the same prompt in order of preference). However, in certain domains (e.g., protein datasets), we work with datasets where each sample is associated with a continuous value (e.g., the activity of each protein), instead of being organized by preferences. I denote this type of datasets as *value dataset* to distinguish it from preference datasets. The seemingly irrelevant difference between value and preference datasets may have some important implications.

For instance, when doing standard RLHF (e.g., PPO) with a preference dataset, we assume a BT/PL model to represent the preferences. This assumption allows us to map (pairwise) preferences to a scalar reward signal, which we can learn with a regression-based reward model. Conversely, with a value dataset, we can directly fit the reward model to the value associated to each data point, without assuming a preference model (e.g., BT, PL models).

Nevertheless, things get a bit more complicated if we want to use DPO with a value dataset. This is because DPO (maximum likelihood) update assumes the data to be distributed according to the BT (or PL) preference model, so that we don't need to compute the 'tricky' partition function, $Z()$ (see Sec. 2.2). As a result, to use the DPO update with a value dataset, we first need to convert the value dataset into a preference dataset. Although this conversion may be trivial (e.g., rank each data point according to its value), it carries some important implications. First, it assumes a preference model (e.g., BT or PL) is an appropriate model of your value dataset. However, there may be scenarios where this is not the case. For instance, in a setting, where we know that a data point with a higher value is always preferred over a data point with a lower value, then the BT/PL models may not be good models (because they assume some noise in the preferences).

There is a second potential issue when converting a value dataset to a preference dataset to apply DPO, which I cannot get my head around. Specifically, by converting continuous values to discrete preferences, we may lose relative information. For instance, one data point may have a reward value that is much greater than all the others. Intuitively, a discrete raking system would give this data point the same weight as it would if the data point had only a minimally higher value than all the other data points. I actually this no longer applies when we model preferences/raking with PL/BT models, since these models model the p. of a preference is proportional to a latent reward. Therefore, as long as we build the preferences based on the value of each data points, we should still preserve relative information. Additionally, things get even more complicated when applying DPO in this setting. This is because DPO still estimates an (implicit) continuous reward signal from the 'transformed' preference dataset, potentially preserving relative information between data points. So, I guess a key question is whether the implicit reward estimated by DPO behaves like the original value associated to each data point?. I guess this could be tested empirically.

In summary, while with preference dataset the differences between DPO and standard RLHF may be more subtle, due to both approaches relaying on BT/PL

models to go from discrete (e.g., binary) preferences to a continuous reward, in the presence of value datasets, there may be more key differences between the two approaches since standard RLHF does not need to assume a preference model (I think this is also because the issue number 1 raised above, I actually don't buy issue number 2).

In this regard a **key question for DPO with value datasets** is: "Does bootstrapping pairwise preferences from scalar reward functions like protein activity work well for all settings?

3.1 Weighted DPO

Widatalla et al. (2024) introduces an alternative DPO update, which they argue it may work better with value datasets. This version of DPO allows you to use value datasets directly, without needing to convert the continuous values into discrete preferences based on BT/PL models. However, after some thinking, this claim is not true, the reward distribution assumed by Widatalla et al. (2024) is the same distribution as that induced by BT/PL, leaving us with the same assumption made by the BT/PL models. The authors argue BT/PL may not be good models of activity-based protein preferences, because 1) BT/PL assume the preferences to be probabilistic (noise), which is not the case for protein activity? 2) Any monotonic transformation of the reward would lead to the same preference distribution under the BT/PL model thus, converting protein activity to preferences based on BT/PL model may cause a loss in relative information. I think this latter point is related to the under-specificity of the PL/BT models. However, even in standard RL settings, it is not clear that a monotonic transformation of the reward function would have any effect on the resulting policy. So, I find the first point more convincing for not wanting to use a BT/PL model with protein activity rather than the second point.

In the DPO derivation, eq. 9 shows how the reward can be parametrized in terms of the LLM outputs and the partition function (this parametrization does not depend on any preference model assumption),

$$r(x, y) = \beta \ \log \frac{\pi^*(y \mid x)}{\pi^{\text{SFT}}(y \mid x)} \ + \ \beta \ \log Z(x) \tag{21}$$

If we have access to a continuous reward/value for each data point, we can introduce a distance-based loss between the rewards/values and this reward parametrization (eq. 21), enabling us to update the LLM (polciy) directly based on the scalar rewards/values, $D(r(x, y), r_\theta(r(x, y))$. Nevertheless, there is one big complication since eq. 9 deepens on the 'tricky' partition function, Z. Widatalla et al. (2024) solve this problem by assuming the rewards are distributed according to,

$$w^i = \frac{\exp\{r(x, y^i)\}}{\sum_i \exp\{r(x, y^i)\}} \tag{22}$$

which allows to eliminate the dependence on the partition function from the LLM reward parametrization in eq 21 (with some simple algebra),

$$w_\theta^i = \frac{\exp\{\beta \log \frac{\pi_\theta(y^i|x)}{\pi_\theta^{SFT}(y^i|x)}\}}{\sum_i \exp\left\{\beta \log \frac{\pi_\theta(y^i|x)}{\pi_\theta^{SFT}(y^i|x)}\right\}} \tag{23}$$

These are valid distribution, resembling Boltzmann Distribution (i.e., soft-max distribution). Based on this reward distribution form, the authors introduce a KL-distance loss between the distribution of (observed) rewards/values and the the distribution of rewards as parametrized by the LLM, $D_{\mathrm{KL}}\left(w_\theta \,||\, w\right)$ for $w_\theta = (w_\theta^i, \cdots, w^N)$ and $w = (w^i, \cdots, w^N)$. With some algebra, they show this KL based loss simplifies to (shown here for pairs of sequences, though can be extended to groups of sequences),

$$\begin{aligned}\mathcal{L}^{\text{weighted-DPO}}(\theta) = -\mathbb{E}_{(x^i,y_1^i,y_2^i)\sim D}[\\ & w^1 \log\sigma\left(\beta \ \log\frac{\pi_\theta(y_1^i \mid x^i)}{\pi^{\mathrm{SFT}}(y_1^i \mid x^i)} \ - \beta \log\frac{\pi_\theta(y_2^i \mid x^i)}{\pi^{\mathrm{SFT}}(y_2^i \mid x^i)}\right) \\ & w^2 \log\sigma\left(\beta \ \log\frac{\pi_\theta(y_1^i \mid x^i)}{\pi^{\mathrm{SFT}}(y_1^i \mid x^i)} \ - \beta \log\frac{\pi_\theta(y_2^i \mid x^i)}{\pi^{\mathrm{SFT}}(y_2^i \mid x^i)}\right) \\ &]\end{aligned} \tag{24}$$

This is just the **expected** DPO update (for the BT model) over the distribution induced by the assumed reward distribution,

$$\mathcal{L}^{\text{weighted-DPO}}(\theta) = -\mathbb{E}_{(x^i,y_1^i,y_2^i)\sim D}\left[\mathbb{E}_w\left[\log\sigma\left(\beta \ \log\frac{\pi_\theta(y_1^i \mid x^i)}{\pi^{\mathrm{SFT}}(y_1^i \mid x^i)} \ - \beta \log\frac{\pi_\theta(y_2^i \mid x^i)}{\pi^{\mathrm{SFT}}(y_2^i \mid x^i)}\right)\right]\right] \tag{25}$$

In practice, this means it is doing the exact same thing as standard DPO, just reducing some variance of the update at a greater computational cost, by taking the 'analytical' expectation over pairs of preferences under the BT model. I think this is the case because the assumed reward distribution is identical to the preference distribution assumed by the BT model (i.e., $w^i = p^{\mathrm{BT}}(y_1^i \prec y_2^i \mid x^i)$. Specifically, in standard DPO with preference datasets, we 'only' have access to samples of preferences, $(x, y_{\mathrm{w}}, y_{\mathrm{l}})$, not the actual BT/PL distribution given by the latent reward parameter. Therefore, we rely on a Monte-Carlo estimate of $\mathbb{E}_w$, increasing the variance of the update. Here, we use the actual rewards/values to compute the actual distribution over preferences/rewards and then, based on this distribution, we compute the analytical expectation. I think the Widatalla et al. (2024)'s paper is misleading in this regard, because it seems they introduce an alternative distribution over rewards, which enables to use a distance function, while eliminating the partition function. However, this distribution is the same distribution assumed by the BT model, leaving us with an expected DPO update over the (BT) distribution of preferences. Therefore, all we should expect from weighted-DPO is a reduction of variance (i.e. faster learning ?), while still relying on all the assumption made by the BT/PL models.

4 Acknowledgments

- Amazing blog post on DPO vs RLHF from Nathan Lambert.
- Super interesting Twitter thread on DPO insights.

5 Useful math identities

5.1 Binary soft-max

$$p(a) = \frac{\exp\{a\}}{\exp\{a\} + \exp\{b\}} \tag{26}$$

$$= \frac{\exp\{a\}}{\exp\{a\}\ \left(1 + \frac{\exp\{b\}}{\exp\{a\}}\right)} \tag{27}$$

$$= \frac{1}{\left(1 + \frac{\exp\{b\}}{\exp\{a\}}\right)} \tag{28}$$

$$= \frac{1}{1 + \exp\{b - a\}} \tag{29}$$

We can finally factorize a -1 in the exponential to get the sigmoid functions,

$$= \sigma(a - b) \tag{30}$$

References

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., & Finn, C. (2024). Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, *36*.

Widatalla, T., Rafailov, R., & Hie, B. (2024). Aligning protein generative models with experimental fitness via direct preference optimization. *bioRxiv*, 2024–05.