Direct Preference Optimization:

Your Language Model Is Secretly a Reward Model[1]

J. Setpal

March 28, 2024

[1]Rafailov, Sharma, Mitchell, et. al

Outline

1. Background, Intuition, Motivations
2. Deriving & Understanding DPO
3. Results

Outline

Task Formulation

Consider the sample conversation:
Human: What is your favourite pet?
LLM: I like _______

Q: What's the *correct* answer?

Task Formulation

Consider the sample conversation:
Human: What is your favourite pet?
LLM: I like _______

Q: What's the *correct* answer? Dog? Cat? Human? Something else?

Task Formulation

Consider the sample conversation:
Human: What is your favourite pet?
LLM: I like _______

Q: What's the *correct* answer? Dog? Cat? Human? Something else?
A: Depends[2] on our *preferences*.

[2] But still, definitely not human.

Task Formulation

Consider the sample conversation:
Human: What is your favourite pet?
LLM: I like _______

Q: What's the *correct* answer? Dog? Cat? Human? Something else?
A: Depends[2] on our *preferences*.

<u>Preference Modelling</u> involves embedding subjective biases within LLMs.

[2] But still, definitely not human.

Task Formulation

Consider the sample conversation:
Human: What is your favourite pet?
LLM: I like _______

Q: What's the *correct* answer? Dog? Cat? Human? Something else?
A: Depends[2] on our *preferences*.

Preference Modelling involves embedding subjective biases within LLMs.

Following are two popular objective models DPO solves for:

a. **Bradley-Terry:** Binary result ranking – $y_1 \succ y_2$
b. **Plackett-Luce:** Multi-result ranking – $y_1 \succ y_2 \succ y_3 \succ \ldots \succ y_n$

[2] But still, definitely not human.

RLHF Synopsis (1/2)

We'll review the RLHF pipeline per Zeiger et al. It has 3-primary phases:

1. **Supervised Fine-Tuning (SFT)**: A pre-trained LLM (π_{PT}) is fine-tuned on high-quality, domain-specific datasets to obtain π_{SFT}.

RLHF Synopsis (1/2)

We'll review the RLHF pipeline per Zeiger et al. It has 3-primary phases:

1. **Supervised Fine-Tuning (SFT)**: A pre-trained LLM (π_{PT}) is fine-tuned on high-quality, domain-specific datasets to obtain π_{SFT}.
2. **Reward Modelling:** Next, we obtain a reward model $r_\phi(x, y)$ that models user preferences.

RLHF Synopsis (1/2)

We'll review the RLHF pipeline per Zeiger et al. It has 3-primary phases:

1. **Supervised Fine-Tuning (SFT)**: A pre-trained LLM (π_{PT}) is fine-tuned on high-quality, domain-specific datasets to obtain π_{SFT}.
2. **Reward Modelling:** Next, we obtain a reward model $r_\phi(x, y)$ that models user preferences. We start by sampling from π_{SFT}:

$$\mathcal{D} := \{(x_j, y_1, y_2)\}_{i=1, j=1}^{N,K} \sim \pi_{SFT}(y|x), \{x_i\}_{i=1}^{K} \quad (1)$$

$$y_w \succ y_l | x \sim r^*(x, y) \; \forall \; (y_1, y_2) \in \mathcal{D} \quad (2)$$

   where $r^*(x, y)$ is the unknown optimal policy.

RLHF Synopsis (1/2)

We'll review the RLHF pipeline per Zeiger et al. It has 3-primary phases:

1. **Supervised Fine-Tuning (SFT)**: A pre-trained LLM (π_{PT}) is fine-tuned on high-quality, domain-specific datasets to obtain π_{SFT}.
2. **Reward Modelling:** Next, we obtain a reward model $r_\phi(x, y)$ that models user preferences. We start by sampling from π_{SFT}:

$$\mathcal{D} := \{(x_j, y_1, y_2)\}_{i=1,j=1}^{N,K} \sim \pi_{SFT}(y|x), \{x_i\}_{i=1}^{K} \tag{1}$$

$$y_w \succ y_l | x \sim r^*(x, y) \; \forall \; (y_1, y_2) \in \mathcal{D} \tag{2}$$

where $r^*(x, y)$ is the unknown optimal policy. Per Bradley-Terry:

$$p^*(y_1 \succ y_2 | x) = \sigma_{softmax[y_1]}(r^*(x, y)); \; y \in \{y_1, y_2\} \tag{3}$$

is the preference distribution optimized over negative log-likelihood on a parameterized model $r_\phi(x, y)$.

RLHF Synopsis (1/2)

We'll review the RLHF pipeline per Zeiger et al. It has 3-primary phases:

1. **Supervised Fine-Tuning (SFT)**: A pre-trained LLM (π_{PT}) is fine-tuned on high-quality, domain-specific datasets to obtain π_{SFT}.
2. **Reward Modelling:** Next, we obtain a reward model $r_\phi(x, y)$ that models user preferences. We start by sampling from π_{SFT}:

$$\mathcal{D} := \{(x_j, y_1, y_2)\}_{i=1,j=1}^{N,K} \sim \pi_{SFT}(y|x), \{x_i\}_{i=1}^{K} \tag{1}$$

$$y_w \succ y_l | x \sim r^*(x, y) \; \forall \; (y_1, y_2) \in \mathcal{D} \tag{2}$$

where $r^*(x, y)$ is the unknown optimal policy. Per Bradley-Terry:

$$p^*(y_1 \succ y_2 | x) = \sigma_{softmax[y_1]}(r^*(x, y)); \; y \in \{y_1, y_2\} \tag{3}$$

is the preference distribution optimized over negative log-likelihood on a parameterized model $r_\phi(x, y)$. Some notes:

a. Rewards are normalized over x to motivate lower variance.

RLHF Synopsis (1/2)

We'll review the RLHF pipeline per Zeiger et al. It has 3-primary phases:

1. **Supervised Fine-Tuning (SFT)**: A pre-trained LLM (π_{PT}) is fine-tuned on high-quality, domain-specific datasets to obtain π_{SFT}.
2. **Reward Modelling:** Next, we obtain a reward model $r_\phi(x, y)$ that models user preferences. We start by sampling from π_{SFT}:

$$\mathcal{D} := \{(x_j, y_1, y_2)\}_{i=1,j=1}^{N,K} \sim \pi_{SFT}(y|x), \{x_i\}_{i=1}^{K} \tag{1}$$

$$y_w \succ y_l | x \sim r^*(x, y) \ \forall \ (y_1, y_2) \in \mathcal{D} \tag{2}$$

where $r^*(x, y)$ is the unknown optimal policy. Per Bradley-Terry:

$$p^*(y_1 \succ y_2|x) = \sigma_{softmax[y_1]}(r^*(x, y)); \ y \in \{y_1, y_2\} \tag{3}$$

is the preference distribution optimized over negative log-likelihood on a parameterized model $r_\phi(x, y)$. Some notes:

   a. Rewards are normalized over x to motivate lower variance.
   b. r_ϕ is π_{SFT} with the final linear layer returning the scalar reward.

RLHF Synopsis (2/2)

3. **RL Fine-Tuning:** Finally, we use r_ϕ to fine-tune π_{SFT}, with the following objectives:
   a. r_ϕ should be maximized. **Assumption:** $r^* \approx r_\phi$.

RLHF Synopsis (2/2)

3. **RL Fine-Tuning:** Finally, we use r_ϕ to fine-tune π_{SFT}, with the following objectives:
   a. r_ϕ should be maximized. **Assumption:** $r^* \approx r_\phi$.
   b. We *do not* want mode-collapse (random tokens that maximize reward).

RLHF Synopsis (2/2)

3. **RL Fine-Tuning:** Finally, we use r_ϕ to fine-tune π_{SFT}, with the following objectives:
   a. r_ϕ should be maximized. **Assumption:** $r^* \approx r_\phi$.
   b. We *do not* want mode-collapse (random tokens that maximize reward). **Solution:** KL Divergence.

RLHF Synopsis (2/2)

3. **RL Fine-Tuning:** Finally, we use r_ϕ to fine-tune π_{SFT}, with the following objectives:
   a. r_ϕ should be maximized. **Assumption:** $r^* \approx r_\phi$.
   b. We *do not* want mode-collapse (random tokens that maximize reward). **Solution:** KL Divergence.

   Mathematically, RLHF posits the following optimization problem:

$$\max_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(y|x)}(r_\phi(x, y)) - \beta \mathbb{D}_{KL}[\pi_\theta(y|x) \mid\mid \pi_{SFT}(y|x)] \tag{4}$$

RLHF Synopsis (2/2)

3. **RL Fine-Tuning:** Finally, we use r_ϕ to fine-tune π_{SFT}, with the following objectives:
   a. r_ϕ should be maximized. **Assumption:** $r^* \approx r_\phi$.
   b. We *do not* want mode-collapse (random tokens that maximize reward). **Solution:** KL Divergence.

   Mathematically, RLHF posits the following optimization problem:

$$\max_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(y|x)}(r_\phi(x, y)) - \beta \mathbb{D}_{KL}[\pi_\theta(y|x) \mid\mid \pi_{SFT}(y|x)] \quad (4)$$

   This is equivalent to the reward function:

$$r(x, y) = r_\phi(x, y) - \beta(\log \pi_\theta(y|x)) - \log(\pi_{SFT}(y|x)) \quad (5)$$

   Which is maximized using **Proximal Policy Optimization**.

Why DPO?

The primary issue with using RLHF is language generation is discrete.

Why DPO?

The primary issue with using RLHF is language generation is discrete. As a consequence, the objective is **non-differentiable**.

Why DPO?

The primary issue with using RLHF is language generation is discrete. As a consequence, the objective is **non-differentiable**.

Actor-Critic Algorithms are unstable because of the **normalization term**:

$$\max_{\pi_\theta} \mathbb{E}_{\pi_\theta(y|x)} \left[r_\phi(x, y) - \beta \log \sum_y \pi_{SFT}(y|x) \exp\left(\frac{r_\phi(x, y)}{\beta}\right) - \beta \log \frac{\pi_\theta(y|x)}{\pi_{SFT}(y|x)} \right] \tag{6}$$

High variance $\propto$ instability, and the normalization term is hard to optimize.

Why DPO?

The primary issue with using RLHF is language generation is discrete. As a consequence, the objective is **non-differentiable**.

Actor-Critic Algorithms are unstable because of the **normalization term**:

$$\max_{\pi_\theta} \mathbb{E}_{\pi_\theta(y|x)} \left[r_\phi(x,y) - \beta \log \sum_y \pi_{SFT}(y|x) \exp\left(\frac{r_\phi(x,y)}{\beta}\right) - \beta \log \frac{\pi_\theta(y|x)}{\pi_{SFT}(y|x)} \right] \tag{6}$$

High variance $\propto$ instability, and the normalization term is hard to optimize. This can be learned or sampled by human-completion baselines.

Why DPO?

The primary issue with using RLHF is language generation is discrete. As a consequence, the objective is **non-differentiable**.

Actor-Critic Algorithms are unstable because of the **normalization term**:

$$\max_{\pi_\theta} \mathbb{E}_{\pi_\theta(y|x)} \left[r_\phi(x, y) - \beta \log \sum_y \pi_{SFT}(y|x) \exp\left(\frac{r_\phi(x, y)}{\beta}\right) - \beta \log \frac{\pi_\theta(y|x)}{\pi_{SFT}(y|x)} \right] \tag{6}$$

High variance $\propto$ instability, and the normalization term is hard to optimize. This can be learned or sampled by human-completion baselines.

DPO creates r_ϕ that enables optimal policy extraction in closed form.

Why DPO?

The primary issue with using RLHF is language generation is discrete. As a consequence, the objective is **non-differentiable**.

Actor-Critic Algorithms are unstable because of the **normalization term**:

$$\max_{\pi_\theta} \mathbb{E}_{\pi_\theta(y|x)} \left[r_\phi(x,y) - \beta \log \sum_y \pi_{SFT}(y|x) \exp\left(\frac{r_\phi(x,y)}{\beta}\right) - \beta \log \frac{\pi_\theta(y|x)}{\pi_{SFT}(y|x)} \right] \tag{6}$$

High variance $\propto$ instability, and the normalization term is hard to optimize. This can be learned or sampled by human-completion baselines.

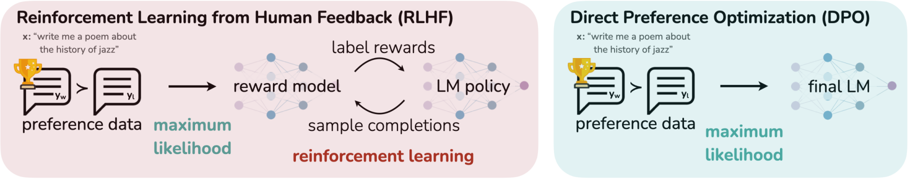
DPO creates r_ϕ that enables optimal policy extraction in closed form.

The optimization policy represents both: the language model and *implicit* reward, that is optimized with log-loss.[3]

[3] "Your Language Model Is Secretly a Reward Model"

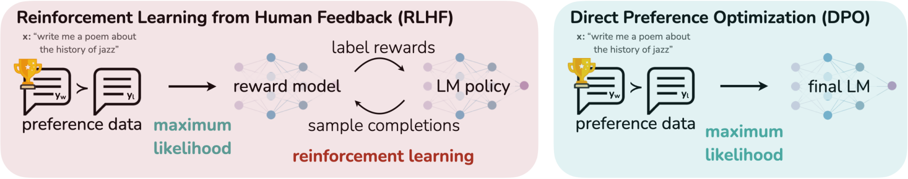
Outline

Objective



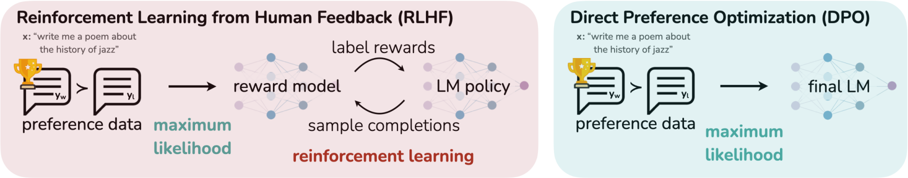
Reinforcement Learning from Human Feedback (RLHF)
x: "write me a poem about the history of jazz"
preference data
maximum likelihood
reward model
label rewards
sample completions
LM policy
reinforcement learning
Direct Preference Optimization (DPO)
x: "write me a poem about the history of jazz"
preference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

Objective



Objective: Loss over *reward* $\xrightarrow{Tr}$ Loss over **policy**.

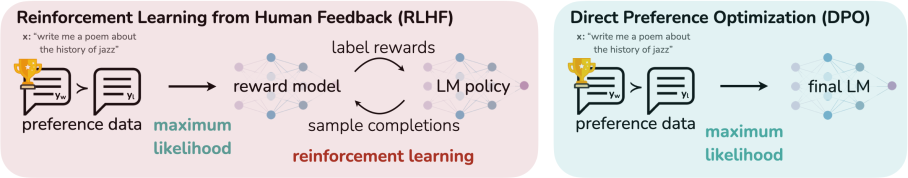
Objective



Objective: Loss over *reward* $\xrightarrow{Tr}$ Loss over **policy**.

We must first identify $(\pi_{SFT}, r) \xrightarrow{Tr} \pi^*$; π^* is a valid probability distribution.

Objective

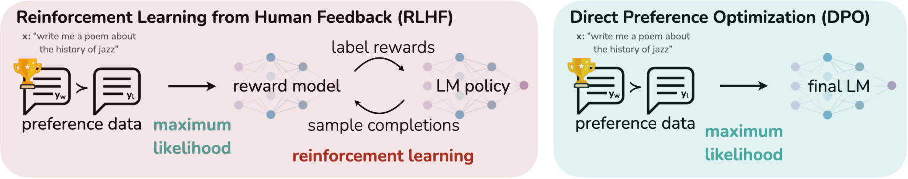


Objective: Loss over *reward* $\xrightarrow{Tr}$ Loss over **policy**.

We must first identify $(\pi_{SFT}, r) \xrightarrow{Tr} \pi^*$; π^* is a valid probability distribution.

We use this to construct $\mathcal{L}_{DPO}$ which maximizes NLL over $\mathbb{D}_{KL}(\pi_\theta, \pi^*)$.

Objective



Objective: Loss over *reward* $\xrightarrow{Tr}$ Loss over **policy**.

We must first identify $(\pi_{SFT}, r) \xrightarrow{Tr} \pi^*$; π^* is a valid probability distribution.

We use this to construct $\mathcal{L}_{DPO}$ which maximizes NLL over $\mathbb{D}_{KL}(\pi_\theta, \pi^*)$.

If π_{SFT} is not available, we obtain it by training π_{PT} for the MLE:

$$\pi_{SFT} \overset{def}{=} \max_{\pi_{PT}} \mathbb{E}_{\{x, y_w\} \sim \mathcal{D}}(\log(\pi_{PT}(y_w|x))) \tag{7}$$

which minimizes distribution shift.

Finding $(\pi_{SFT}, r) \xrightarrow{Tr} \pi^*$ (1/2)

We begin by restructuring the maximization objective from RLHF:

$$\max_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(y|x)}(r_\phi(x, y)) - \beta \mathbb{D}_{KL}[\pi_\theta(y|x) \mid\mid \pi_{SFT}(y|x)] \tag{4}$$

$$= \min_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi_\theta(y|x)} \left[\log \frac{\pi_\theta(y|x)}{\frac{1}{Z(x)} \pi_{SFT}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)} - \log Z(x) \right] \tag{8}$$

Where $Z(x)$ is a partition function (scalar that induces proportionality).

Finding $(\pi_{SFT}, r) \xrightarrow{Tr} \pi^* $ (1/2)

We begin by restructuring the maximization objective from RLHF:

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_\theta(y|x)}(r_\phi(x,y)) - \beta\mathbb{D}_{KL}[\pi_\theta(y|x) \mid\mid \pi_{SFT}(y|x)] \tag{4}$$

$$= \min_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi_\theta(y|x)} \left[\log \frac{\pi_\theta(y|x)}{\frac{1}{Z(x)}\pi_{SFT}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)} - \log Z(x)\right] \tag{8}$$

Where $Z(x)$ is a partition function (scalar that induces proportionality).

In English this time, here's what's happening:

1. The maximization objective is reward minus KL divergence.

Finding $(\pi_{SFT}, r) \xrightarrow{Tr} \pi^*$ (1/2)

We begin by restructuring the maximization objective from RLHF:

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)}(r_\phi(x,y)) - \beta\mathbb{D}_{KL}[\pi_\theta(y|x) \mid\mid \pi_{SFT}(y|x)] \tag{4}$$

$$= \min_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi_\theta(y|x)}\left[\log \frac{\pi_\theta(y|x)}{\frac{1}{Z(x)}\pi_{SFT}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)} - \log Z(x)\right] \tag{8}$$

Where $Z(x)$ is a partition function (scalar that induces proportionality).

In English this time, here's what's happening:

1. The maximization objective is reward minus KL divergence.
2. max $\xrightarrow{Tr}$ min objective is divergence minus reward.

Finding $(\pi_{SFT}, r) \xrightarrow{Tr} \pi^*$ (1/2)

We begin by restructuring the maximization objective from RLHF:

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)}(r_\phi(x,y)) - \beta\mathbb{D}_{KL}[\pi_\theta(y|x) \mid\mid \pi_{SFT}(y|x)] \tag{4}$$

$$= \min_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi_\theta(y|x)} \left[\log \frac{\pi_\theta(y|x)}{\frac{1}{Z(x)}\pi_{SFT}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)} - \log Z(x)\right] \tag{8}$$

Where $Z(x)$ is a partition function (scalar that induces proportionality).

In English this time, here's what's happening:

1. The maximization objective is reward minus KL divergence.
2. max $\xrightarrow{Tr}$ min objective is divergence minus reward.
3. We can combine reward and the SFT model by plugging in a partition function.

Finding $(\pi_{SFT}, r) \xrightarrow{Tr} \pi^*$ (2/2)

From our new objective, we extract on optimal policy π^*:

$$\min_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi_\theta(y|x)}\left[\log \frac{\pi_\theta(y|x)}{\frac{1}{Z(x)}\pi_{SFT}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)} - \log Z(x)\right] \tag{8}$$

$$\pi^*(y|x) = \frac{1}{Z(x)}\pi_{SFT}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right) \tag{9}$$

Finding $(\pi_{SFT}, r) \xrightarrow{Tr} \pi^*$ (2/2)

From our new objective, we extract on optimal policy π^*:

$$\min_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi_\theta(y|x)}\left[\log\frac{\pi_\theta(y|x)}{\frac{1}{Z(x)}\pi_{SFT}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)} - \log Z(x)\right] \quad (8)$$

$$\pi^*(y|x) = \frac{1}{Z(x)}\pi_{SFT}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right) \quad (9)$$

From here, we log both sides and solve for $r(x, y)$:

$$r(x,y) = \beta\left[\log\frac{\pi^*(y|x)}{\pi_{SFT}(y|x)} + \log(Z(x))\right] \quad (10)$$

Finding $(\pi_{SFT}, r) \xrightarrow{Tr} \pi^*$ (2/2)

From our new objective, we extract on optimal policy π^*:

$$\min_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}} \mathbb{E}_{y\sim\pi_\theta(y|x)} \left[\log \frac{\pi_\theta(y|x)}{\frac{1}{Z(x)}\pi_{SFT}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)} - \log Z(x) \right] \tag{8}$$

$$\pi^*(y|x) = \frac{1}{Z(x)}\pi_{SFT}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right) \tag{9}$$

From here, we log both sides and solve for $r(x, y)$:

$$r(x,y) = \beta \left[\log \frac{\pi^*(y|x)}{\pi_{SFT}(y|x)} + \log(Z(x)) \right] \tag{10}$$

This is *still* unsolvable, because it's hard to approximate $Z(x)$. However, we can fit this to the **Bradley-Terry Model**.

Obtaining $\mathcal{L}_{DPO}$

Recall from RLHF definition, we have eq. (3):

$$p^*(y_1 \succ y_2 | x) = \sigma_{softmax[y_1]}(r^*(x, y)); \;\; y \in \{y_1, y_2\} \tag{3}$$

$\frac{dZ(x)}{dy} = 0$, so it does not play a role in optimization.

Obtaining $\mathcal{L}_{DPO}$

Recall from RLHF definition, we have eq. (3):

$$p^*(y_1 \succ y_2|x) = \sigma_{softmax[y_1]}(r^*(x,y));\ \ y \in \{y_1, y_2\} \tag{3}$$

$\frac{dZ(x)}{dy} = 0$, so it does not play a role in optimization. We get:

$$p^*(y_1 \succ y_2|x) = \sigma_{softmax}\left(\beta \log \frac{\pi^*(y_1|x)}{\pi_{SFT}(y_1|x)} - \beta \log \frac{\pi^*(y_2|x)}{\pi_{SFT}(y_2|x)}\right) \tag{11}$$

which is loss over a single sample.

Obtaining $\mathcal{L}_{DPO}$

Recall from RLHF definition, we have eq. (3):

$$p^*(y_1 \succ y_2|x) = \sigma_{softmax[y_1]}(r^*(x,y));\; y \in \{y_1, y_2\} \tag{3}$$

$\frac{dZ(x)}{dy} = 0$, so it does not play a role in optimization. We get:

$$p^*(y_1 \succ y_2|x) = \sigma_{softmax}\left(\beta \log \frac{\pi^*(y_1|x)}{\pi_{SFT}(y_1|x)} - \beta \log \frac{\pi^*(y_2|x)}{\pi_{SFT}(y_2|x)}\right) \tag{11}$$

which is loss over a single sample.

Finally, we can maximize expectation over $p^*(y_1 \succ y_2|x)$ with NLL:

$$\mathcal{L}_{DPO} = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}[\log(p^*(y_1 \succ y_2|x))] \tag{12}$$

over which we find our MLE.

Outline

1. Background, Intuition, Motivations
2. Deriving & Understanding DPO
3. Results

Benchmark Scores

DPO's authors evaluated their approach on the following tasks:

1. **Controlled Sentiment Generation**
2. **Text Summarization**
3. **Single-Turn Dialogue**

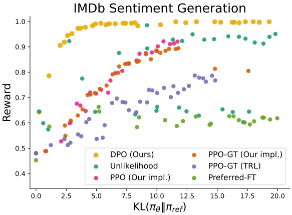
They used GPT-4 to perform 'auto-evaluations' on the data, by asking GPT-4 to select a winner output through blind testing.

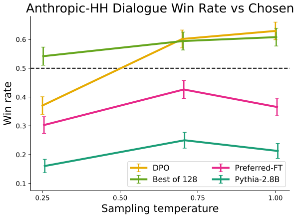
Benchmark Scores

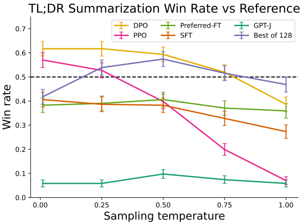
DPO's authors evaluated their approach on the following tasks:

1. **Controlled Sentiment Generation**
2. **Text Summarization**
3. **Single-Turn Dialogue**

They used GPT-4 to perform 'auto-evaluations' on the data, by asking GPT-4 to select a winner output through blind testing.







Scalability

DPO was shown to work well at scale, outperforming PPO *without* tuning β.

Scalability

DPO was shown to work well at scale, outperforming PPO *without* tuning β. It was also compared with human evaluators for robustness:

	DPO	**SFT**	**PPO-1**
N respondents	272	122	199
GPT-4 (S) win %	47	27	13
GPT-4 (C) win %	54	32	12
Human win %	58	43	17
GPT-4 (S)-H agree	70	77	86
GPT-4 (C)-H agree	67	79	85
H-H agree	65	-	87

Generalizability & Future Work

To evaluate **generalizability**, DPO is also compared against PPO on out-of-distribution inference on CNN/DailyMail articles.

	Win rate vs. ground truth	
Alg.	Temp 0	Temp 0.25
DPO	0.36	0.31
PPO	0.26	0.23

Here, DPO outperforms PPO *despite* not using additional unlabelled prompts, that PPO requires.

Generalizability & Future Work

To evaluate **generalizability**, DPO is also compared against PPO on out-of-distribution inference on CNN/DailyMail articles.

	Win rate vs. ground truth	
Alg.	Temp 0	Temp 0.25
DPO	0.36	0.31
PPO	0.26	0.23

Here, DPO outperforms PPO *despite* not using additional unlabelled prompts, that PPO requires.

The authors note that more extensive study on the generalizability of DPO is necessary.

Generalizability & Future Work

To evaluate **generalizability**, DPO is also compared against PPO on out-of-distribution inference on CNN/DailyMail articles.

	Win rate vs. ground truth	
Alg.	Temp 0	Temp 0.25
DPO	0.36	0.31
PPO	0.26	0.23

Here, DPO outperforms PPO *despite* not using additional unlabelled prompts, that PPO requires.

The authors note that more extensive study on the generalizability of DPO is necessary.

Finally, DPO is also *only* evaluated on 6B parameter models, and an exploration of it's performance at scale is also necessitated.

Thank you!

Have an awesome rest of your day!

Slides: `https://cs.purdue.edu/homes/jsetpal/slides/dpo.pdf`