Diving into DPO

1.Preliminaries

1.1 高层次视角

DPO的初始动机是设计一个不依赖外部显式奖励函数的人类偏好学习算法，从而避免基于PPO的RLHF算法所需的高昂显存开销与训练困难（需要同时在显存中保持四个模型）。最终效果就是通过用模型本身建模了等价的隐式奖励函数，然后通过优化简单的交叉熵损失来最大化KL散度约束下的奖励，从而**不需要奖励模型、不需要critic模型，只需要reference模型与policy模型。**

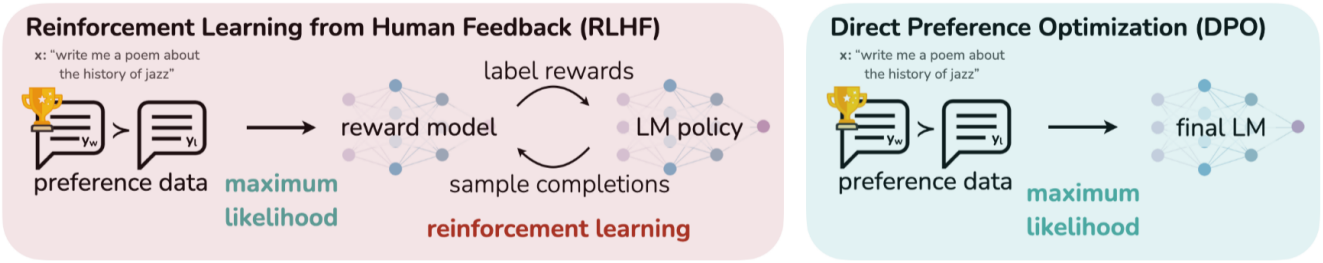


Figure 1: **DPO optimizes for human preferences while avoiding reinforcement learning.** Existing methods for fine-tuning language models with human feedback first fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward. In contrast, DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, fitting an *implicit* reward model whose corresponding optimal policy can be extracted in closed form.

1.2 如何建模偏好？-> Bradley-Terry model

想要对齐人类的偏好，我们首先需要一种方法来量化"偏好"这种概念。
Bradley-Terry（BT）模型是一种建模偏好的概率分布的方法，属于 Plackett-Luce 排名模型的一种特例（源于金融学，不用去了解）。对于由两个偏好数据组成的偏好数据对，一般使用BT模型来建模偏好。

- 对于DPO，我们假设有一个隐式奖励模型 $r^*(x,y)$ 为 prompt x 与其对应相应 y 给出奖励值。BT 模型的定义为：

$$p^*(y_1 \succ y_2|x) = \frac{\exp(r^*(x,y_1))}{\exp(r^*(x,y_1)) + \exp(r^*(x,y_2))} = \sigma(r^*(x,y_1) - r^*(x,y_2)) \tag{1}$$

  BT 模型给出了对于 prompt x ，响应 y_1 优于 y_2 的概率。先不用纠结这个隐式的奖励模型怎么来的，后续我们会推导出 r^* 实际上可以使用 policy 模型本身来计算（不需要额外训一个奖励模型）。
- 对于传统的RLHF算法，可以对 BT 模型建模的概率使用最小化负对数似然方法（NLL）训练一个参数化的实体奖励模型 $r_\phi(x,y)$，损失函数为：

$$\begin{aligned}\mathcal{L}_R(r_\phi, \mathcal{D}) &= -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log p_\phi(y_w \succ y_l|x)\right] \\ &= -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}[\log \sigma(r_\phi(x,y_w) - r_\phi(x,y_l))]\end{aligned} \tag{2}$$

  该训练目标的作用是，令奖励模型 r_ϕ 面对人类偏好的响应 y_w 时给出更高的奖励值，面对不偏好的响应 y_l 时给出较低的奖励值。（传统做法为了保证奖励函数的低方差，会令奖励模型对不偏好的响应输出负奖励值，以保证 $\mathbb{E}_{x,y\sim\mathcal{D}}[r_\phi(x,y)] = 0$）

一句话总结：BT 模型利用奖励函数（隐式或显式）算出偏好的概率分布 $p(y_1 \succ y_2|x)$，我们最大化这个概率以对齐人类偏好。

1.3 RLHF中的关键部分 -- KL散度约束

RLHF方法中强化学习部分的损失函数为：

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

其中第二项 $\mathbb{D}_{\mathrm{KL}}\left[\pi_\theta(y \mid x)||\pi_{\mathrm{ref}}(y \mid x)\right]$ 为当前policy模型 π_θ 之于参考模型 π_{ref} （一般设置为刚SFT后的模型）的KL散度。这一约束或者说惩罚十分重要，因为如果没有这一约束，模型参数的更新幅度过大，会导致模型输出的分布偏离奖励模型能够准确打分的分布，从而使得奖励模型失效。其次，如果没有这一约束，模型很可能会过分倾向于输出高奖励的响应，即出现模式崩溃，导致模型只会输出很单一的内容。最后，由

于LLM内部是一个十分复杂的系统，如果进行过于激进的更新，可能会导致原始模型中很多信息被破坏，导致模型出现灾难性遗忘（所以在sft、rlhf阶段通常会使用较小的学习率）。
DPO就是从这个视角出发，目标推导出一个KL约束下与原始RLHF中显式奖励模型等价的隐式奖励模型。作者经过推导发现可以直接用policy模型本身来表示隐式奖励模型，由此可以直接通过更新policy的参数来对齐偏好，从而绕过了训练奖励模型与强化学习。

1.4 DPO目标函数推导

这一节沿着作者的心路历程看看DPO的目标函数是怎么来的。作者并不是一上来就奔着消解RLHF目标中的奖励模型去的，而是首先尝试能否从原始RLHF目标函数中解出最优policy的闭式解，如果能解出来那就好办了，说不定可以直接跳过整个优化过程，直接从偏好数据中导出最优模型参数（通常也不可能，推出的解往往会存在很多难以计算的项，还是需要进行优化，但是推导闭式解的过程往往能启发我们转换优化目标，即用一些方便计算和优化的东西替换掉原始目标中难以计算的东西）。

1.4.1 首先尝试推导闭式解

从传统的RLHF目标函数式 (3) 出发，我们尝试推推看能不能直接求解出最优policy $\pi^*(y \mid x)$的闭式解，首先将目标函数变为：

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[r(x,y)\right]-\beta\mathbb{D}_{\mathrm{KL}}\left[\pi(y \mid x)\|\pi_{\mathrm{ref}}(y \mid x)\right] \tag{4}$$

将KL散度展开

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[r(x,y)\right]-\beta\mathbb{E}_{x\sim\mathcal{D}}[\pi(y \mid x)\log\frac{\pi(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)}] \tag{5}$$

注意上式第二项实际上是在求 $\log\frac{\pi(y|x)}{\pi_{\mathrm{ref}}(y|x)}$之于 $\pi(y \mid x)$ 的期望。
科普一下，$\mathbb{E}_{x\sim p(x)}[f(x)]$ 这种写法表示在随机变量 x 的分布 $p(x)$ 下，函数 $f(x)$ 的平均值，换句话说是在求 $f(x)$ 的期望，其中 x 服从分布 $p(x)$ ，即

$$\mathbb{E}_{x\sim p(x)}[f(x)]=\int p(x)\cdot f(x)dx$$

所以我们将式(5)写为

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[r(x,y)-\beta\log\frac{\pi(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)}\right] \tag{6}$$

提出一个$-\beta$ 将我们的优化目标从最大化变为最小化，其他乘上或除去一个常数对优化目标本身没有影响

$$\begin{aligned}
&Eq.\,(6)\\
&=\min_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\pi_{\mathrm{ref}}(y|x)}-\frac{1}{\beta}r(x,y)\right]\\
&=\min_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\pi_{\mathrm{ref}}(y|x)}-\log\exp\left(\frac{1}{\beta}r(x,y)\right)\right]\\
&=\min_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\pi_{\mathrm{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)}\right]
\end{aligned} \tag{7}$$

注意到上式分母为一个概率分布 $\pi_{\mathrm{ref}}(y \mid x)$, 乘上一个权重系数 $\exp\left(\frac{1}{\beta}r(x,y)\right)$, 一个自然的想法便是把分母重新归一化一下变成一个合法的新概率分布，然后便能又凑出KL散度的计算式，然后利用Gibbs 不等式求解出最优 policy 的闭式解 。于是我们定义归一化因子

$$\mathrm{Z(x)}=\sum_{y}\pi_{\mathrm{ref}}(y \mid x)\exp\left(\frac{1}{\beta}r(x,y)\right)$$

将$\mathrm{Z}(x)$ 除到式(7)的分母上，对分母进行归一化

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

于是分母便转化为一个合法的概率分布，我们将其定义为 $\pi^*(y \mid x)$，即

$$\pi^*(y \mid x)=\frac{1}{\mathrm{Z}(x)}\pi_{\mathrm{ref}}(y \mid x)\exp\left(\frac{1}{\beta}r(x,y)\right)$$

则优化目标转化为

$$\min_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[\log\frac{\pi(y \mid x)}{\pi^*(y \mid x)}-\log\mathrm{Z}(x)\right] \tag{9}$$

注意上式中 $\mathrm{Z}(x)$ 与 y 无关，所以把它提出去

$$\min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y\mid x)}{\pi^*(y\mid x)}\right]-\log \mathrm{Z}(x)\right] \tag{10}$$

上式中第一项便是计算 $\pi(y\mid x)$ 与 $\pi^*(y\mid x)$ 之间的KL散度 $\mathbb{D}_{\mathrm{KL}}\left[\pi(y\mid x)\|\pi^*(y\mid x)\right]$，所以

$$\min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{D}_{\mathrm{KL}}\left[\pi(y\mid x)\|\pi^*(y\mid x)\right]-\log \mathrm{Z}(x)\right] \tag{11}$$

上式中 $\mathrm{Z}(x)$ 与我们要优化的策略 $\pi(y\mid x)$ 无关，由Gibbs不等式可知KL散度的最小值为0，当且仅当两个分布完全相等，所以我们终于推导出了原始RLHF优化目标的最优解：

$$\pi(y\mid x)=\pi^*(y|x)=\frac{1}{\mathrm{Z}(x)}\pi_{\mathrm{ref}}(y\mid x)\exp\left(\frac{1}{\beta}r(x,y)\right) \tag{12}$$

1.4.2 重参数化 $r(x,y)$

观察式(12)的最优解，我们会发现该解需要我们首先拥有一个奖励模型 $r(x,y)$，其次归一化系数 $\mathrm{Z}(x)$ 难以直接计算，需要使用大量响应 y 来估计计算，所以推半天推出来的这个解没啥用。没办法直接求出最优策略，但能不能直接求个奖励模型？于是，我们尝试把式(12)变换为：

$$r(x,y)=\beta\log\frac{\pi^*(y\mid x)}{\pi_{\mathrm{ref}}(y\mid x)}+\beta\log \mathrm{Z}(x) \tag{13}$$

我们发现可以用最优policy + 参考policy + 归一化因子$\mathrm{Z}(x)=\sum_y \pi_{\mathrm{ref}}(y\mid x)\exp\left(\frac{1}{\beta}r(x,y)\right)$ 来重参数化奖励函数。看起来没什么卵用，但回忆一下 BT 模型的偏好概率 $p^*(y_1\succ y_2|x)=\sigma\left(r^*(x,y_1)-r^*(x,y_2)\right)$，就会发现把这种重参数化的奖励函数带进去可以把 $\mathrm{Z}(x)$ 消掉！这样就可以直接用模型和参考模型来导出奖励信号！即

$$p^*(y_1\succ y_2|x)=\frac{1}{1+\exp\left(\beta\log\frac{\pi^*(y_2|x)}{\pi_{\mathrm{ref}}(y_2|x)}-\beta\log\frac{\pi^*(y_1|x)}{\pi_{\mathrm{ref}}(y_1|x)}\right)} \tag{14}$$

1.4.3 计划有变，直接推DPO目标函数

原本想推出闭式解的作者一番折腾后消解掉了对奖励模型的依赖，得到式(14)的偏好概率分布后一切都好办了，一篇 oral 级论文在招手。直接用简单的最大似然估计来最大化式(14)，我们就得到了DPO的损失函数：

$$\mathcal{L}_{\mathrm{DPO}}(\pi_\theta;\pi_{\mathrm{ref}})=-\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w\mid x)}{\pi_{\mathrm{ref}}(y_w\mid x)}-\beta\log\frac{\pi_\theta(y_l\mid x)}{\pi_{\mathrm{ref}}(y_l\mid x)}\right)\right] \tag{15}$$

于是我们可以跳过奖励模型训练、强化学习阶段，直接使用偏好数据对 (x,y_w,y_l) 就可以引导梯度下降向着对其人类偏好的方向更新我们的语言模型 π_θ

1.4.4 DPO损失的梯度

$$\nabla_\theta\mathcal{L}_{\mathrm{DPO}}(\pi_\theta;\pi_{\mathrm{ref}})=$$
$$-\beta\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Bigg[\underbrace{\sigma(\hat{r}_\theta(x,y_l)-\hat{r}_\theta(x,y_w))}_{\text{higher weight when reward estimate is wrong}}\Bigg[\underbrace{\nabla_\theta\log\pi(y_w\mid x)}_{\text{increase likelihood of }y_w}-\underbrace{\nabla_\theta\log\pi(y_l\mid x)}_{\text{decrease likelihood of }y_l}\Bigg]\Bigg],$$

上式中 $\hat{r}_\theta(x,y)=\beta\log\frac{\pi_\theta(y|x)}{\pi_{\mathrm{ref}}(y|x)}$，便是我们建模的隐式奖励函数。
分析一下DPO loss 的梯度可以发现，实际上DPO在做一个加权的对比更新，首先每步更新会增加偏好响应 y_w 的似然，同时降低不偏好响应 y_l 的似然，外层乘上一个隐式奖励函数之差的权重。直观理解，当非偏好响应被错误地赋予了更高的奖励时，更新的权重会较大，此时以大力度增强正例，削弱负例。

2.DPO实验表现

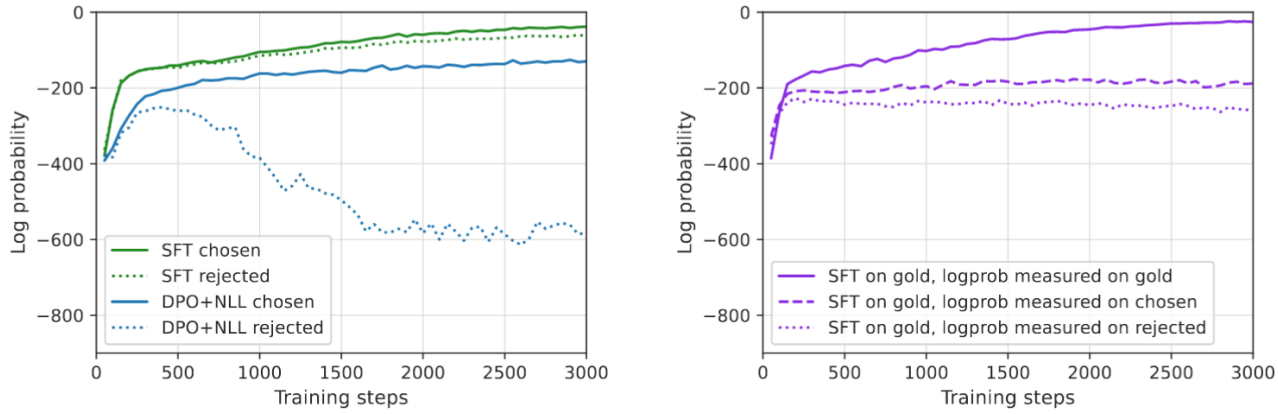
作为一个偏好学习算法，DPO拥有着强正例、削负例这个很强大的性质，但是算法本身存在许多细节，如果不注意往往会导致与预期效果出现偏差。一些工作也报道了DPO存在一些问题，这一节将展开介绍一下。

2.1 用其他模型生成的偏好数据

我们在推导DPO损失时，是从KL约束下的优化目标出发的，也就是说DPO隐式建模的奖励函数 $\hat{r}_\theta(x,y)=\beta\log\frac{\pi_\theta(y|x)}{\pi_{\mathrm{ref}}(y|x)}$ 是需要考虑参考模型输出概率的。一般来说，偏好数据对由 π^{SFT} 生成，同时我们会初始化 π_{ref} 为 π^{SFT}，但当我们想复用网上的开源偏好数据时，可能无法获取 π^{SFT}，这就会导致DPO更新不准确。原论文作者指出，这种情况下可以先用偏好数据中的正例 (x,y_w) 来SFT一下当前模型，再使用这个SFT后的模型作为 π_{ref}，以缓解一点分布偏移。

2.2 直观理解DPO的效果

对于LLM，了解一个训练算法最直观的方法就是观察训练过程中，训练数据的对数概率变化。下面的图出自《Iterative Reasoning Preference Optimization》，作者对比了模型自我生成的偏好数据集中拒绝响应和接受响应的对数概率随训练过程的变化情况：



(a) SFT trained on chosen seqs; init from Llama　　(b) SFT trained on gold CoTs; init from Llama

Figure 2: **Effect of SFT training**. **(a)** Although SFT training (solid green) is on chosen sequences (D_0^{pairs}, from iterative RPO iteration 1) only, the rejected sequence log probabilities (dotted green) also increase and are close to the chosen sequence probabilities. In contrast, our DPO+NLL training (blue) manages to decrease the rejected probabilities while increasing the chosen probabilities. This observation could potentially help explain why SFT-only performance lags significantly behind Iterative RPO Iteration 1 performance. **(b)** We show a similar plot but where SFT is trained on gold (dataset-provided) CoTs. Chosen and rejected sequence probabilities (which are from D_0^{pairs}) are still close to each other, but with a slightly bigger gap. Another observation is that the chosen sequence probabilities barely increase.

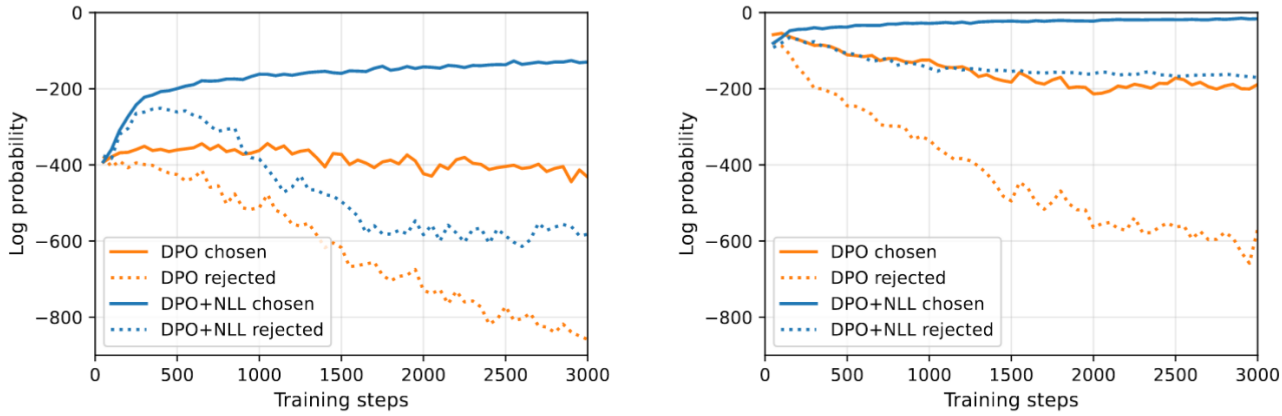
上面左图的SFT是单纯在接受的正例上训练，有趣的是不管是正例还是负例，其对数概率都在增加而且增加的程度都差不多。作者没有在论文中说明具体使用什么响应来测的对数概率，但我猜测可能是因为该实验所测的正例与负例十分相似，也许只在几个token上存在不一致，所以导致了正例负例对数概率同时增长。但这也为我们提了一个醒，单纯的SFT并不能十分细致地区分正例负例，在提升正例的同时可能不小心地顺带提升了相似负例，从而导致响应正确性提升不大，或者必须使用贪心解码，稍加放松可能就会采样到同样高概率但是导向错误结果的token。

对于DPO，可以看出正例和负例的对数概率之间的 gap 在不断扩大，正例提升幅度较小（弱于SFT），但负例出现了较大程度的下降。

注意，上面的实验结果是作者魔改的DPO目标，就是在原始DPO目标上加了一个负对数似然（NLL）项，简单来说就是加强了对正例的增强幅度。可以看出即使在增加了NLL项之后，DPO对正例的增强能力还是比不上SFT，这个问题在原始DPO上更为严重。下面一节将介绍一下DPO存在的问题。

2.3 DPO的问题

还是刚才那篇论文，作者对比了原始DPO与 DPO+NLL 在GSM8K上收集到的偏好数据对的效果



(a) Initialized from Llama　　(b) Initialized from SFT trained on chosen seqs

Figure 3: **Effect of NLL loss term on DPO training for GSM8K**. In our GSM8K experiments we observe the log probability of chosen sequences in standard DPO without NLL loss (solid orange) decreases over training steps, especially if the model is initialized from SFT training on chosen sequences (right). However, they *increase* over training steps when using DPO with NLL loss (solid blue). In all four settings, the margin between the two curves continues increasing. We find that DPO+NLL loss gives superior test accuracy in our experiments.

左图是直接对 Llama base model 进行训练，右图是先用偏好数据中的正例对 base model 进行SFT，然后对 SFT-ed 的模型进行训练。结果显示对于原始DPO（橙色），正例对数概率几乎没有提升，甚至还出现了下降，这一现象也在众多工作中被报道。此外，观察左右两图发现对SFT后的模型进行DPO，其正例出现了较大幅度的下降，这一现象也表明DPO对SFT十分敏感，如果想要迭代使用SFT+DPO，必须深入理解DPO 为什么会对不同的初始模型表现出不同的更新效果。

《Towards Analyzing and Understanding the Limitations of DPO: A Theoretical Perspective》一文指出DPO对正例更新效果差是由于DPO的目标函数对于 $\pi(y_l|x)$ 的梯度比 $\pi(y_w|x)$ 的更大。此外，DPO对SFT敏感是由于DPO的梯度向量场中不同区域的大小和方向一直变化, 所以DPO会对初始的经过SFT的policy十分敏感。下面将展开讲讲作者是如何得出这种观察的。

首先，简化一下DPO的损失函数

$$\begin{aligned}\mathcal{L}_{\mathrm{DPO}}(\pi_\theta;\pi_{\mathrm{ref}}) &= -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w\mid x)}{\pi_{\mathrm{ref}}(y_w\mid x)}-\beta\log\frac{\pi_\theta(y_l\mid x)}{\pi_{\mathrm{ref}}(y_l\mid x)}\right)\right]\\ &= -\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w\mid x)}{\pi_{\mathrm{ref}}(y_w\mid x)}-\beta\log\frac{\pi_\theta(y_l\mid x)}{\pi_{\mathrm{ref}}(y_l\mid x)}\right)\right]\\ &= -\log\left(\frac{x_1^\beta}{x_1^\beta+x_2^\beta}\right)\end{aligned} \tag{15}$$

其中 $x_1=\frac{\pi_\theta(y_w|x)}{\pi_{\mathrm{ref}}(y_w|x)}$，$x_2=\frac{\pi_\theta(y_l|x)}{\pi_{\mathrm{ref}}(y_l|x)}$，为了可视化DPO损失的梯度向量场，我们把 x_1,x_2 视为变量，于是就可以计算DPO的梯度对于x_1,x_2 的偏导数：

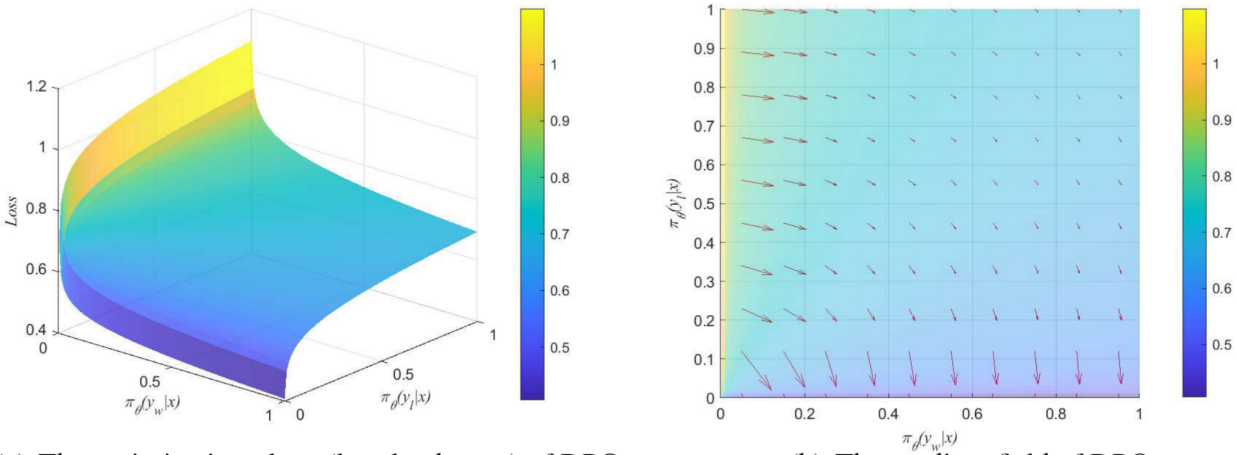
$$\begin{cases}\frac{\partial\mathcal{L}_{DPO}(x_1;x_2)}{\partial x_1}=-\frac{\beta x_2^\beta}{x_1\left(x_1^\beta+x_2^\beta\right)},\\ \frac{\partial\mathcal{L}_{DPO}(x_1;x_2)}{\partial x_2}=\frac{\beta x_2^{\beta-1}}{x_1^\beta+x_2^\beta}.\end{cases} \tag{16}$$

偏导的意义是多元函数对某一个变量的变化率，也就是说我们可以通过比较上面两个偏导的大小来分析DPO梯度更新会更偏向哪一部分。

$$\left|\frac{\partial\mathcal{L}_{DPO}\left(x_1;x_2\right)}{\partial x_1}/\frac{\partial\mathcal{L}_{DPO}\left(x_1;x_2\right)}{\partial x_2}\right|=\frac{x_2}{x_1} \tag{17}$$

注意 x_1,x_2 为 $\pi_\theta(y\mid x),\pi_{\mathrm{ref}}(y\mid x)$ 两个概率值之比，其中分子上的 $\pi_{\mathrm{ref}}(y\mid x)$ 为参考模型给出的概率值，我们假设 $\pi_{\mathrm{ref}}(y_w\mid x)=\frac{1}{a},\pi_{\mathrm{ref}}(y_l\mid x)=\frac{1}{b}$，其中 $a,b\geq 1$。由此可得，$x_1\in[0,a],x_2\in[0,b]$。随着优化过程进行，$\pi_\theta(y_w\mid x)$ 将会越来越高，然后超过 $\frac{1}{a}$，最终 x_1 将会超过1；同理可得，x_2 将会小于1，于是 $\frac{x_2}{x_1}$ 将会小于1，我们便推导出 $\left|\frac{\partial\mathcal{L}_{DPO}(x_1;x_2)}{\partial x_1}\right|<\left|\frac{\partial\mathcal{L}_{DPO}(x_1;x_2)}{\partial x_2}\right|$。这从DPO loss函数的角度证明了，DPO倾向于对非偏好数据施加更大的更新。

由于 $\pi_{\mathrm{ref}}(y\mid x)$ 为参考模型给出的概率分布，与我们当前训练的参数 θ 无关，所以可以将其视为一个常数。我们知道除以一个常数只会对我们的图像进行伸缩变换，而不会改变图像的形式，所以我们简单地将$\pi_{\mathrm{ref}}(y\mid x)$固定为1，然后可视化一下DPO loss相对于变量 x_1,x_2 的梯度场：



(a) The optimization plane (loss landscape) of DPO

(b) The gradient field of DPO

Figure 1: The optimization plane (loss landscape) and gradient field of DPO. Figure (a) illustrates the values of DPO loss under different probabilities of generating prefer and dispref responses, known as the optimization plane (loss landscape) of DPO. Figure (b) provides a top-down view of the optimization plane (loss landscape) and incorporates the gradient field at different positions using red arrows. The direction of the red arrows represents the gradient-based optimization direction, while the length of the red arrows represents magnitudes.

右侧俯视图可以很清楚地看出DPO梯度的方向以及大小，当我们的模型落到图中的右上角时（即对正例、负例都具有较大概率，常出现于刚经过SFT初始化的模型）此时DPO梯度大小非常小，算法可能被困于鞍点。当模型处于图中下半部分时时（生成负例的概率已经较小），算法会快速降低负例生成概率，同时对于正例几乎不做增强，这也对应了上面观测到的实验现象。当落入图像左上角时（负例概率高，正例概率小）此时模型难以生成正例，DPO会快速增强正例，但对负例做较小的削弱，然后可能陷入右上部的鞍点。