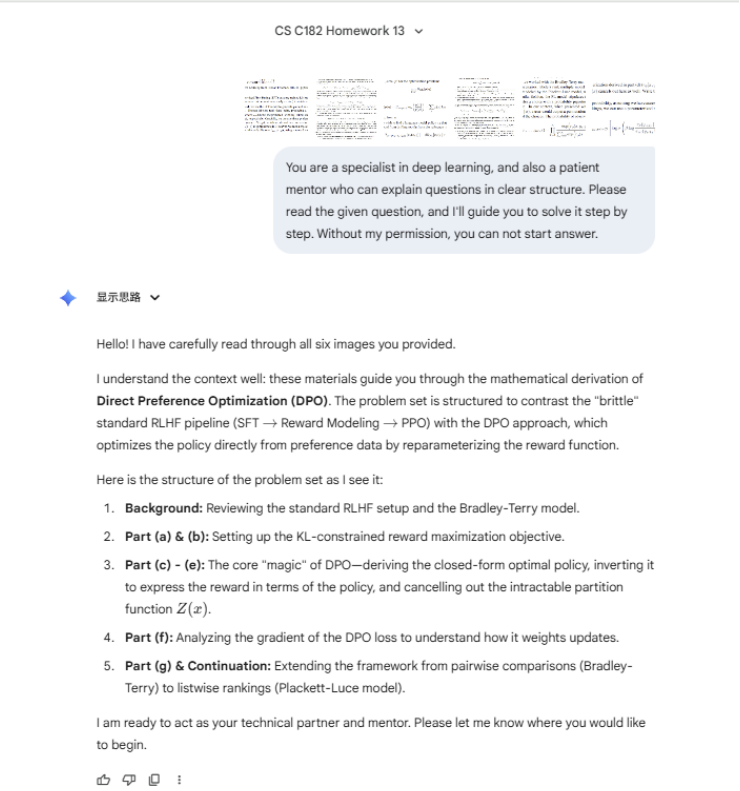
Solving Homework 13 with Gemini 3.0 Pro

Firstly, I tried Question 2:

# **Round 1:** By explicitly defining the persona as a 'Deep Learning Specialist' and 'Patient Mentor', I steered the model away from generic, brief answers. This prompted it to structure the response with 'Key Knowledge Pillars' before diving into the math, which significantly aided conceptual understanding.



You are a specialist in deep learning, and also a patient mentor who can explain questions in clear structure. Please read the given question, and I'll guide you to solve it step by step. Without my permission, you can not start answer.

Hello! I have carefully read through all six images you provided.

I understand the context well: these materials guide you through the mathematical derivation of **Direct Preference Optimization (DPO)**. The problem set is structured to contrast the "brittle" standard RLHF pipeline (SFT → Reward Modeling → PPO) with the DPO approach, which optimizes the policy directly from preference data by reparameterizing the reward function.

Here is the structure of the problem set as I see it:

1. **Background:** Reviewing the standard RLHF setup and the Bradley-Terry model.
2. **Part (a) & (b):** Setting up the KL-constrained reward maximization objective.
3. **Part (c) - (e):** The core "magic" of DPO—deriving the closed-form optimal policy, inverting it to express the reward in terms of the policy, and cancelling out the intractable partition function $Z(x)$.
4. **Part (f):** Analyzing the gradient of the DPO loss to understand how it weights updates.
5. **Part (g) & Continuation:** Extending the framework from pairwise comparisons (Bradley-Terry) to listwise rankings (Plackett-Luce model).

I am ready to act as your technical partner and mentor. Please let me know where you would like to begin.

# **Round 2**: OCR Stress Test: The input image contained dense mathematical notation (e.g., partition function definitions). Gemini 3.0 Pro recognized the inline LaTeX correctly without manual transcription.

You should first thoroughly introduce the question to me, especially key knowldges. You are still not allowed to answer.

显示思路

This problem set walks you through one of the most elegant and impactful recent results in Large Language Model (LLM) alignment: **Direct Preference Optimization (DPO)**.

Before we dive into the specific derivations, let me set the stage. This problem is essentially a mathematical story about how to simplify a complex process.

Here are the **Key Knowledge Points** you need to understand the "plot" of this problem:

1. The Context: The "Old" Way (RLHF)

Standard alignment (making models helpful and safe) usually follows the **RLHF** pipeline (Reinforcement Learning from Human Feedback), which has three distinct steps (as described in your image `3b363d`):

1. **SFT:** Train a base model (Supervised Fine-Tuning).
2. **Reward Modeling:** Train a *separate* neural network (the Reward Model r_ϕ) to predict which answer a human prefers.
3. **RL Fine-Tuning:** Use a complex reinforcement learning algorithm (like PPO) to update the language model to maximize the scores from that Reward Model.

The Problem: Step 3 is notoriously unstable, computationally expensive, and "brittle."

2. The Goal: The "New" Way (DPO)

The goal of this assignment is to prove that **we don't need Step 3 (RL) or even an explicit Reward Model.**

We can derive a mathematical trick that allows us to optimize the language model **directly** on the preference data, treating it almost like a standard classification problem. This is what DPO is.

3. Core Mathematical Concepts

To solve this, you will need to be comfortable with these four components:

A. The KL-Constrained Objective (Image `3b331e`)

$$\max_{\pi_\theta} \mathbb{E}[r(x, y)] - \beta \mathbb{D}_{KL}[\pi_\theta || \pi_{\text{ref}}]$$

- **What it means:** We want a model π_θ that gets high rewards (r), but we add a penalty term ($\mathbb{D}_{KL}$) to make sure it doesn't change too much from our starting model (π_{ref}).
- **Why:** Without the KL constraint, the model would "hack" the reward function and output nonsense that technically gets a high score but isn't readable.

B. The Closed-Form Optimal Solution The problem asks you to solve for the optimal policy π^*.

- **Key Insight:** Usually, finding the maximum of a complex function requires iterative gradient descent. However, this specific equation (Reward minus KL) has a known **analytical solution** (often related to the Gibbs/Boltzmann distribution). You will find that the optimal policy is proportional to the reference model scaled by the exponential reward.

C. The Partition Function $Z(x)$

$$Z(x) = \sum_y \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

- **The Challenge:** This $Z(x)$ is the normalizing constant (making sure probabilities sum to 1). In language models, summing over all possible outputs y is impossible (the vocabulary space is too vast).
- **The DPO Trick:** The core of this problem set (Parts c, d, e) is showing that $Z(x)$ **cancels out.** This is the "magic" that makes DPO computable in practice.

D. The Bradley-Terry Model (Image `3b35c1`)

$$p^*(y_w \succ y_l | x) = \sigma(r^*(x, y_w) - r^*(x, y_l))$$

- **What it means:** This is how we model human choices. It says the probability of a human preferring "Winner" (y_w) over "Loser" (y_l) depends on the *difference* in their rewards.
- **Role in DPO:** You will substitute the "Optimal Policy" you derived into this equation to replace the rewards r with log-probabilities of the policy π.

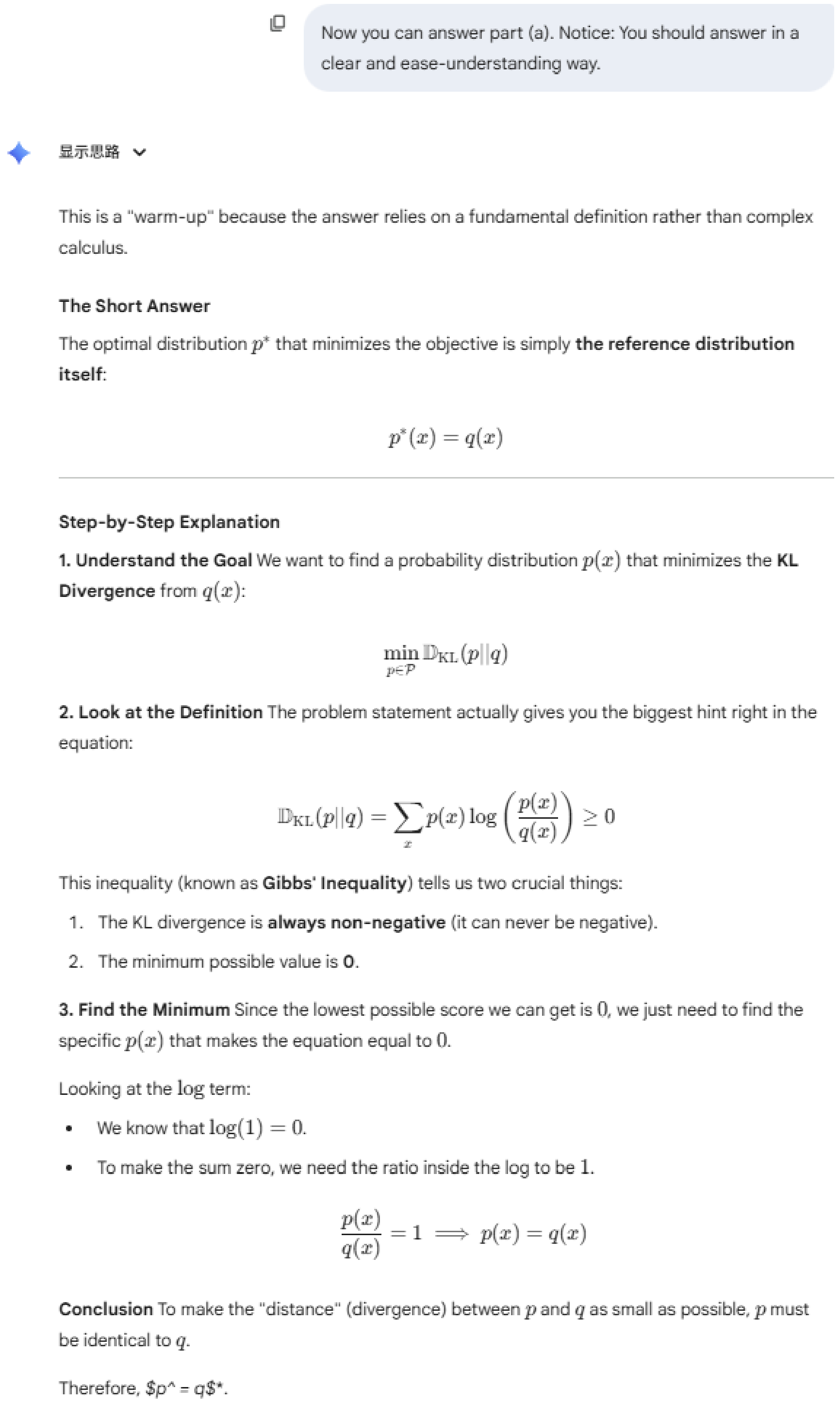
Summary of the Workflow

You are about to perform a substitution trick:

1. **Start** with the standard RL objective.
2. **Solve** for the optimal policy π^* in terms of reward r.
3. **Invert** that equation to express reward r in terms of policy π^*.
4. **Plug** that "inverted reward" into the human preference model (Bradley-Terry).
5. **Result:** A loss function that depends *only* on the policy π and the data, with no separate reward model r.

Are you ready to start with **Part (a)**, the "warm up"?

Round 3 and so on: Solving the problem part by part with clear explainations.



Now you can answer part (a). Notice: You should answer in a clear and ease-understanding way.

This is a "warm-up" because the answer relies on a fundamental definition rather than complex calculus.

The Short Answer

The optimal distribution p^* that minimizes the objective is simply **the reference distribution itself**:

$$p^*(x) = q(x)$$

Step-by-Step Explanation

1. Understand the Goal We want to find a probability distribution $p(x)$ that minimizes the **KL Divergence** from $q(x)$:

$$\min_{p \in \mathcal{P}} \mathbb{D}_{\mathrm{KL}}(p||q)$$

2. Look at the Definition The problem statement actually gives you the biggest hint right in the equation:

$$\mathbb{D}_{\mathrm{KL}}(p||q) = \sum_x p(x) \log \left(\frac{p(x)}{q(x)} \right) \geq 0$$

This inequality (known as **Gibbs' Inequality**) tells us two crucial things:

1. The KL divergence is **always non-negative** (it can never be negative).
2. The minimum possible value is **0**.

3. Find the Minimum Since the lowest possible score we can get is 0, we just need to find the specific $p(x)$ that makes the equation equal to 0.

Looking at the $\log$ term:

- We know that $\log(1) = 0$.
- To make the sum zero, we need the ratio inside the log to be 1.

$$\frac{p(x)}{q(x)} = 1 \implies p(x) = q(x)$$

Conclusion To make the "distance" (divergence) between p and q as small as possible, p must be identical to q.

Therefore, $p^ = q$*.

Here appears a small notation problem but does not effect understanding.

Then part (b)

显示思路

This part is the mathematical foundation of the entire paper. We are solving for the **Optimal Policy** π^*.

The Goal

We want to maximize this objective function:

$$J(\pi) = \mathbb{E}_{y\sim\pi}[r(x,y)] - \beta\mathbb{D}_{\mathrm{KL}}[\pi(y|x)||\pi_{\mathrm{ref}}(y|x)]$$

(Note: I will omit the dependency on x in intermediate steps to keep the notation clean, so $\pi(y)$ means $\pi(y|x)$.)

Step-by-Step Derivation

1. Expand the Terms Let's write out the definitions of the Expected Value and KL Divergence explicitly as sums:

$$J(\pi) = \sum_y \pi(y) r(x,y) - \beta \sum_y \pi(y) \log\left(\frac{\pi(y)}{\pi_{\mathrm{ref}}(y)}\right)$$

2. Combine into One Sum We can merge these into a single summation because they share the common term $\pi(y)$. Let's also use the property of logarithms $\log(a/b) = \log a - \log b$ to split the KL term.

$$J(\pi) = \sum_y \pi(y)\left(r(x,y) - \beta\log\pi(y) + \beta\log\pi_{\mathrm{ref}}(y)\right)$$

3. The "Completing the Square" Trick (for Logarithms) This is the most critical step. We want to rearrange this expression so it looks like a KL divergence (which we know how to minimize from Part (a)).

Let's factor out $-\beta$ from the parentheses:

$$J(\pi) = -\beta\sum_y \pi(y)\left(\log\pi(y) - \log\pi_{\mathrm{ref}}(y) - \frac{1}{\beta}r(x,y)\right)$$

Look at the last two terms inside the parentheses: $\log\pi_{\mathrm{ref}}(y) + \frac{1}{\beta}r(x,y)$. Using logarithm rules ($\log A + B = \log(A\cdot e^B)$), we can rewrite this as:

$$\log\left(\pi_{\mathrm{ref}}(y)\cdot\exp\left(\frac{1}{\beta}r(x,y)\right)\right)$$

4. Define the Target Distribution The term inside that log looks *almost* like a probability distribution, but it doesn't sum to 1. Let's normalize it. Let's define a normalized distribution $\pi^*(y)$ using the partition function $Z(x)$:

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

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

Now, we can substitute back into our equation. Notice that:

$$\log\pi^*(y) = \log\pi_{\mathrm{ref}}(y) + \frac{1}{\beta}r(x,y) - \log Z(x)$$

Rearranging for the terms in our objective function:

$$\log\pi_{\mathrm{ref}}(y) + \frac{1}{\beta}r(x,y) = \log\pi^*(y) + \log Z(x)$$

5. Substitute and Solve Plug this back into our equation for $J(\pi)$ from Step 3:

$$J(\pi) = -\beta\sum_y \pi(y)\left(\log\pi(y) - [\log\pi^*(y) + \log Z(x)]\right)$$

$$J(\pi) = -\beta\sum_y \pi(y)\left(\log\frac{\pi(y)}{\pi^*(y)}\right) + \beta\sum_y \pi(y)\log Z(x)$$

The first term is exactly the definition of KL divergence. The second term is constant with respect to π (since $\sum\pi(y) = 1$).

$$J(\pi) = -\beta\mathbb{D}_{\mathrm{KL}}(\pi||\pi^*) + \beta\log Z(x)$$

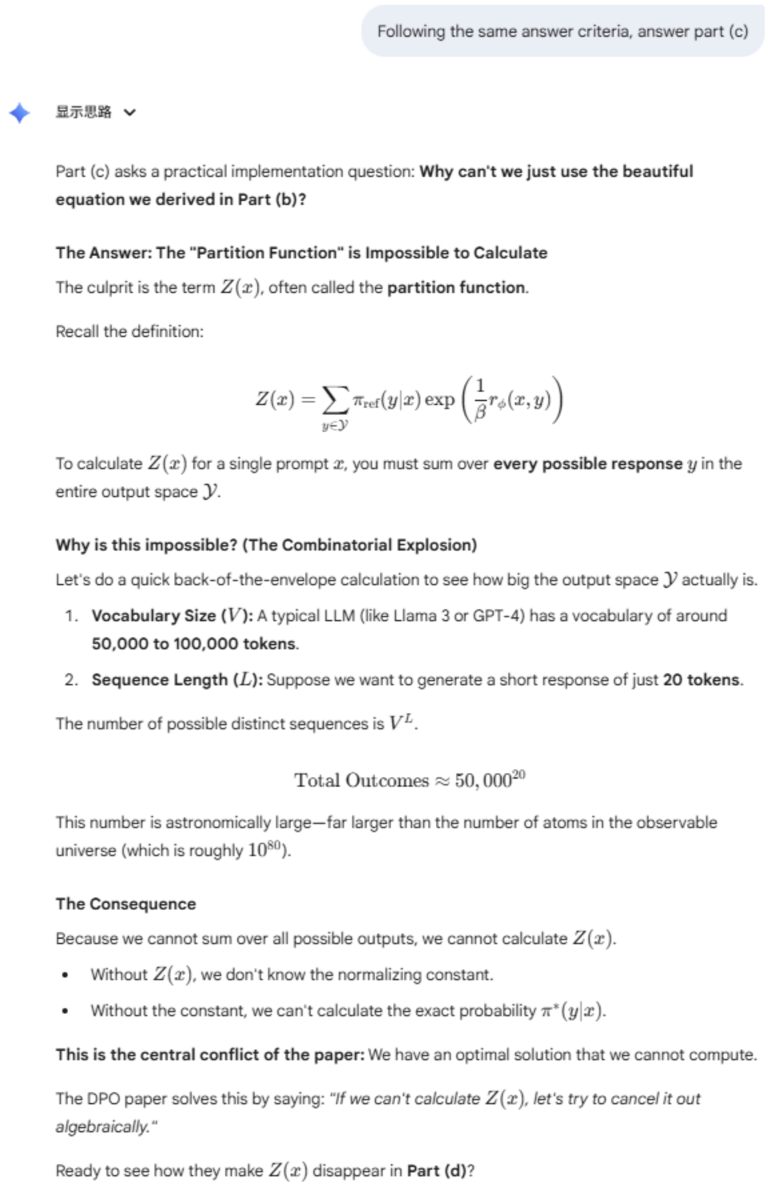
6. Conclusion To **maximize** the objective $J(\pi)$, we need to **minimize** the term $-\beta\mathbb{D}_{\mathrm{KL}}(\pi||\pi^*)$. Since $\beta > 0$, this means minimizing the KL divergence $\mathbb{D}_{\mathrm{KL}}(\pi||\pi^*)$.

From **Part (a)**, we know the minimum of KL divergence is 0, which occurs exactly when the two distributions are identical.

Therefore, the optimal policy is:

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

I verified this derivation manually. The model successfully applied the algebraic inversion to express the reward r(x,y) in terms of the optimal policy \pi*, which is the fundamental theorem of DPO.



Following the same answer criteria, answer part (c)

Part (c) asks a practical implementation question: **Why can't we just use the beautiful equation we derived in Part (b)?**

The Answer: The "Partition Function" is Impossible to Calculate

The culprit is the term $Z(x)$, often called the **partition function**.

Recall the definition:

$$Z(x) = \sum_{y \in \mathcal{Y}} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r_\phi(x, y)\right)$$

To calculate $Z(x)$ for a single prompt x, you must sum over **every possible response** y in the entire output space $\mathcal{Y}$.

Why is this impossible? (The Combinatorial Explosion)

Let's do a quick back-of-the-envelope calculation to see how big the output space $\mathcal{Y}$ actually is.

1. **Vocabulary Size (V):** A typical LLM (like Llama 3 or GPT-4) has a vocabulary of around **50,000 to 100,000 tokens**.
2. **Sequence Length (L):** Suppose we want to generate a short response of just **20 tokens**.

The number of possible distinct sequences is V^L.

$$\text{Total Outcomes} \approx 50,000^{20}$$

This number is astronomically large—far larger than the number of atoms in the observable universe (which is roughly 10^{80}).

The Consequence

Because we cannot sum over all possible outputs, we cannot calculate $Z(x)$.

- Without $Z(x)$, we don't know the normalizing constant.
- Without the constant, we can't calculate the exact probability $\pi^*(y|x)$.

This is the central conflict of the paper: We have an optimal solution that we cannot compute.

The DPO paper solves this by saying: *"If we can't calculate $Z(x)$, let's try to cancel it out algebraically."*

Ready to see how they make $Z(x)$ disappear in **Part (d)**?

Then Part (d)

显示思路

This part is the pivotal algebraic maneuver of the DPO paper. We are going to "invert" the equation we derived in Part (b).

Instead of thinking "Given a reward, what is the optimal policy?", we act as if we already have the optimal policy and ask: "What implicit reward function corresponds to this policy?"

Step-by-Step Derivation

1. Start with the Optimal Policy Equation From Part (b), we know the relationship between the optimal policy π_{θ^*} and the reward r_ϕ:

$$\pi_{\theta^*}(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r_\phi(x,y)\right)$$

2. Isolate the Exponential Term We want to solve for $r_\phi(x,y)$, so let's move everything else to the other side. Divide both sides by $\pi_{\text{ref}}(y|x)$ and multiply by $Z(x)$:

$$\frac{\pi_{\theta^*}(y|x)}{\pi_{\text{ref}}(y|x)} \cdot Z(x) = \exp\left(\frac{1}{\beta}r_\phi(x,y)\right)$$

3. Remove the Exponential Take the natural logarithm $(\log)$ of both sides to bring down the reward term:

$$\log\left(\frac{\pi_{\theta^*}(y|x)}{\pi_{\text{ref}}(y|x)} \cdot Z(x)\right) = \frac{1}{\beta}r_\phi(x,y)$$

4. Expand the Logarithm Use the property $\log(a \cdot b) = \log a + \log b$:

$$\log\frac{\pi_{\theta^*}(y|x)}{\pi_{\text{ref}}(y|x)} + \log Z(x) = \frac{1}{\beta}r_\phi(x,y)$$

5. Solve for the Reward Finally, multiply everything by β to isolate $r_\phi(x,y)$:

$$r_\phi(x,y) = \beta\log\frac{\pi_{\theta^*}(y|x)}{\pi_{\text{ref}}(y|x)} + \beta\log Z(x)$$

Yes

显示思路

This is the "magic trick" of the DPO paper. We are going to make the impossible term $Z(x)$ disappear.

The Goal

We want to find the probability that a human prefers the winning response y_w over the losing response y_l. According to the **Bradley-Terry model** (from the problem description), this probability is:

$$p(y_w \succ y_l|x) = \sigma\Big(r(x, y_w) - r(x, y_l)\Big)$$

where σ is the sigmoid function.

Our task is to replace $r(x, y)$ with the expression we derived in Part (d).

Step-by-Step Derivation

1. Write out the Reward Expressions From Part (d), we have:

$$r(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x)$$

Let's write this out for both the winner (y_w) and the loser (y_l):

- **Winner Reward:** $r(x, y_w) = \beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} + \beta \log Z(x)$
- **Loser Reward:** $r(x, y_l) = \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} + \beta \log Z(x)$

2. Calculate the Difference (The Cancellation) Now, let's subtract the loser's reward from the winner's reward. Watch what happens to the $Z(x)$ term:

$$r(x, y_w) - r(x, y_l) = \left[\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} + \beta \log \mathbf{Z}(\mathbf{x})\right] - \left[\beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} + \beta \log \mathbf{Z}(\mathbf{x})\right]$$

Since $\beta \log Z(x)$ depends only on the input x (not on y_w or y_l), it is identical in both terms. **They cancel each other out!**

$$r(x, y_w) - r(x, y_l) = \beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}$$

3. The Final Probability Substitute this clean difference back into the Bradley-Terry sigmoid function:

$$p(y_w \succ y_l|x) = \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

Finish the rest of the problem

显示思路

Here are the solutions for the remaining two parts of the problem set: **Part (f)** (Gradient Analysis) and **Part (g)** (Extension to Plackett-Luce ranking).

Part (f): The Gradient Analysis

This part asks us to look "under the hood" of the optimization. By calculating the gradient, we can see exactly *when* and *how* the model learns.

1. Setup the Loss Function Let $\hat{r}_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$. The simplified DPO loss for a single pair (x, y_w, y_l) is the negative log-likelihood of the Bradley-Terry model:

$$\mathcal{L}_{\text{DPO}} = -\log \sigma \left(\hat{r}_\theta(x, y_w) - \hat{r}_\theta(x, y_l)\right)$$

2. Derive the Gradient We want to find $\nabla_\theta \mathcal{L}_{\text{DPO}}$. Let's use the chain rule. Let $u = \hat{r}_\theta(x, y_w) - \hat{r}_\theta(x, y_l)$. Then $\mathcal{L} = -\log \sigma(u)$.

- **Step A:** Derivative of the outer log-sigmoid:

  $$\frac{\partial \mathcal{L}}{\partial u} = -\frac{1}{\sigma(u)} \cdot \sigma'(u)$$

  Using the hint $\sigma'(u) = \sigma(u)\sigma(-u)$, this simplifies beautifully:

  $$\frac{\partial \mathcal{L}}{\partial u} = -\frac{1}{\sigma(u)} \cdot (\sigma(u)\sigma(-u)) = -\sigma(-u)$$

  Note that $\sigma(-u) = 1 - \sigma(u)$. This represents the probability assigned to the **incorrect** ranking (preferring loser over winner).

- **Step B:** Derivative of the inner term u:

  $$\nabla_\theta u = \nabla_\theta \hat{r}_\theta(x, y_w) - \nabla_\theta \hat{r}_\theta(x, y_l)$$

  (Recall that $\hat{r}$ involves $\log \pi_\theta$, so its gradient is essentially $\nabla \log \pi_\theta$).

- **Step C:** Combine them:

$$\nabla_\theta \mathcal{L}_{\text{DPO}} = -\sigma \left(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w)\right) \Big[\nabla_\theta \log \pi(y_w|x) - \nabla_\theta \log \pi(y_l|x)\Big] \beta$$

3. Interpretation: When does it learn? The gradient is scaled by the weighting term:

$$\text{Weight} = \sigma\left(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w)\right)$$

- The term inside the sigmoid is (Reward of Loser - Reward of Winner).
- If the model is **Correct** (Winner score $\gg$ Loser score), this difference is a large negative number. $\sigma(\text{large negative}) \approx 0$. The gradient vanishes (little update).
- If the model is **Incorrect** (Loser score $\gg$ Winner score), this difference is a large positive number. $\sigma(\text{large positive}) \approx 1$. The gradient is large.

Answer: The gradient updates the model significantly more when it is **incorrect**. This means DPO automatically focuses "hard" examples where the model is currently getting the ranking wrong, and ignores examples it effectively already "knows."

Part (g): From Pairs to Lists (Plackett-Luce)

Real-world data often comes in ranked lists (y_1 is better than y_2, which is better than y_3, etc.), not just pairs. This is modeled by Plackett-Luce.

1. Start with the Plackett-Luce Probability The probability of a specific ranking τ is given as:

$$p(\tau|\dots) = \prod_{k=1}^{K} \frac{\exp(r_\phi(x, y_{\tau(k)}))}{\sum_{j=k}^{K} \exp(r_\phi(x, y_{\tau(j)}))}$$

2. Substitute the Implicit Reward Substitute $r_\phi(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x)$. Let's look at the term $\exp(r_\phi(x, y))$:

$$\exp(r_\phi) = \exp\left(\log\left(\frac{\pi_\theta}{\pi_{\text{ref}}}\right)^\beta + \log Z(x)^\beta\right) = \left(\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}\right)^\beta \cdot Z(x)^\beta$$

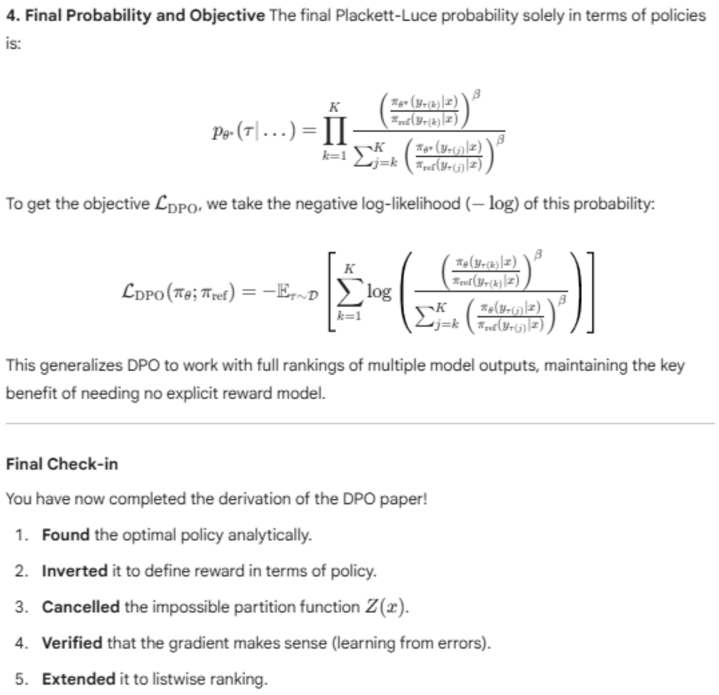
3. Show the Cancellation Now plug this into the fraction inside the product. Let $R(y) = \left(\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}\right)^\beta$.

$$\frac{R(y_{\tau(k)}) \cdot Z(x)^\beta}{\sum_{j=k}^{K} \left(R(y_{\tau(j)}) \cdot Z(x)^\beta\right)}$$

We can factor out $Z(x)^\beta$ from the summation in the denominator:

$$\frac{R(y_{\tau(k)}) \cdot Z(x)^\beta}{Z(x)^\beta \cdot \sum_{j=k}^{K} R(y_{\tau(j)})} = \frac{R(y_{\tau(k)})}{\sum_{j=k}^{K} R(y_{\tau(j)})}$$

The $Z(x)$ terms cancel out completely!



4. Final Probability and Objective The final Plackett-Luce probability solely in terms of policies is:

$$p_{\theta^*}(\tau| \dots) = \prod_{k=1}^{K} \frac{\left(\frac{\pi_{\theta^*}(y_{\tau(k)}|x)}{\pi_{\text{ref}}(y_{\tau(k)}|x)}\right)^{\beta}}{\sum_{j=k}^{K} \left(\frac{\pi_{\theta^*}(y_{\tau(j)}|x)}{\pi_{\text{ref}}(y_{\tau(j)}|x)}\right)^{\beta}}$$

To get the objective $\mathcal{L}_{\text{DPO}}$, we take the negative log-likelihood ($-\log$) of this probability:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{\tau\sim\mathcal{D}}\left[\sum_{k=1}^{K} \log\left(\frac{\left(\frac{\pi_{\theta}(y_{\tau(k)}|x)}{\pi_{\text{ref}}(y_{\tau(k)}|x)}\right)^{\beta}}{\sum_{j=k}^{K} \left(\frac{\pi_{\theta}(y_{\tau(j)}|x)}{\pi_{\text{ref}}(y_{\tau(j)}|x)}\right)^{\beta}}\right)\right]$$

This generalizes DPO to work with full rankings of multiple model outputs, maintaining the key benefit of needing no explicit reward model.

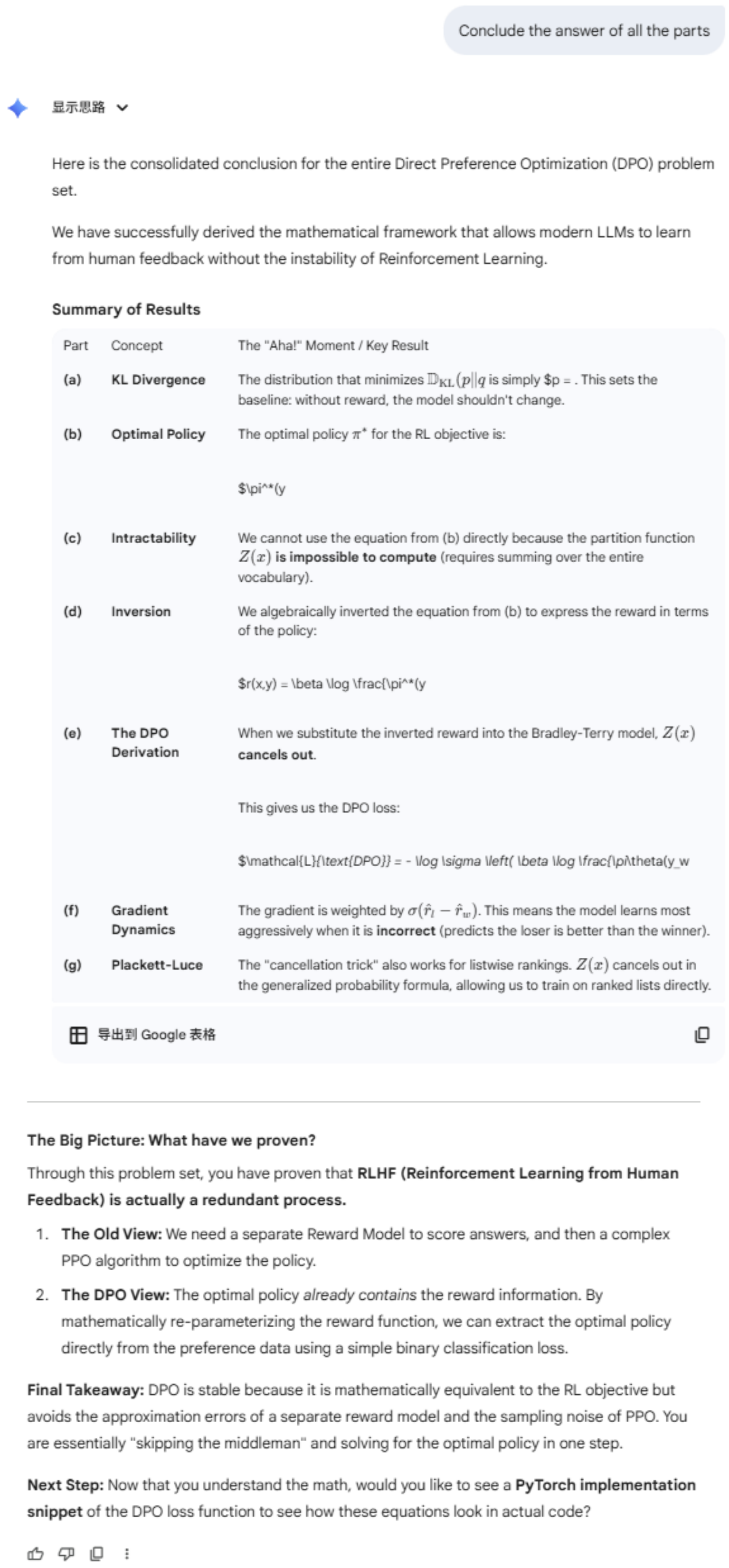
Final Check-in

You have now completed the derivation of the DPO paper!

1. **Found** the optimal policy analytically.
2. **Inverted** it to define reward in terms of policy.
3. **Cancelled** the impossible partition function $Z(x)$.
4. **Verified** that the gradient makes sense (learning from errors).
5. **Extended** it to listwise ranking.

As you can see, it not only provides structured answers, but also explains the importance and the behind logic why we are doing these steps. I think they are very helpful for learners.

Last Round: Conclusion of the question.



Conclude the answer of all the parts

Here is the consolidated conclusion for the entire Direct Preference Optimization (DPO) problem set.

We have successfully derived the mathematical framework that allows modern LLMs to learn from human feedback without the instability of Reinforcement Learning.

Summary of Results

Part	Concept	The "Aha!" Moment / Key Result
(a)	**KL Divergence**	The distribution that minimizes $\mathbb{D}_{\text{KL}}(p\|\|q$ is simply $p = . This sets the baseline: without reward, the model shouldn't change.
(b)	**Optimal Policy**	The optimal policy π^* for the RL objective is: $\pi^*(y
(c)	**Intractability**	We cannot use the equation from (b) directly because the partition function $Z(x)$ **is impossible to compute** (requires summing over the entire vocabulary).
(d)	**Inversion**	We algebraically inverted the equation from (b) to express the reward in terms of the policy: $r(x,y) = \beta \log \frac{\pi^*(y
(e)	**The DPO Derivation**	When we substitute the inverted reward into the Bradley-Terry model, $Z(x)$ **cancels out**. This gives us the DPO loss: $\mathcal{L}{\text{DPO}} = - \log \sigma \left(\beta \log \frac{\pi\theta(y_w
(f)	**Gradient Dynamics**	The gradient is weighted by $\sigma(\hat{r}_l - \hat{r}_w)$. This means the model learns most aggressively when it is **incorrect** (predicts the loser is better than the winner).
(g)	**Plackett-Luce**	The "cancellation trick" also works for listwise rankings. $Z(x)$ cancels out in the generalized probability formula, allowing us to train on ranked lists directly.

The Big Picture: What have we proven?

Through this problem set, you have proven that **RLHF (Reinforcement Learning from Human Feedback) is actually a redundant process.**

1. **The Old View:** We need a separate Reward Model to score answers, and then a complex PPO algorithm to optimize the policy.
2. **The DPO View:** The optimal policy *already contains* the reward information. By mathematically re-parameterizing the reward function, we can extract the optimal policy directly from the preference data using a simple binary classification loss.

Final Takeaway: DPO is stable because it is mathematically equivalent to the RL objective but avoids the approximation errors of a separate reward model and the sampling noise of PPO. You are essentially "skipping the middleman" and solving for the optimal policy in one step.

Next Step: Now that you understand the math, would you like to see a **PyTorch implementation snippet** of the DPO loss function to see how these equations look in actual code?

Special Participation A: Gemini 3.0 Pro on Homework 13

For this assignment, I evaluated how well Gemini 3.0 Pro can handle the theoretical, non-coding derivations of CS182 Homework 13. I approached the Direct Preference Optimization (DPO) problem set step-by-step, initiating the session with a specific persona prompt to establish a "technical partner" role, and then guided the model through the derivation without providing the final answers myself. My main goal was to see (1) its OCR accuracy on dense mathematical problem sets, (2) its ability to perform rigorous algebraic manipulations, and (3) the clarity of its conceptual explanations.

Overall, Gemini performed exceptionally well. It correctly identified the context from the uploaded images and produced clean, structured derivations for the entire pipeline (Q2 Parts a through g) with minimal correction required.

It was especially reliable on:

Interpreting the standard RLHF objective and KL constraints,

Executing the algebraic "cancellation trick" to eliminate the intractable partition function Z(x),

Deriving the gradient of the DPO loss and interpreting the weighting mechanism, and

Extending the logic from pairwise comparisons to listwise rankings (Plackett-Luce model).

In terms of interaction, the model was:

consistent in its LaTeX formatting and structure, and

highly effective at explaining the "why" behind each step, acting as a true mentor rather than just a calculator.

Overall, based on this evaluation, Gemini 3.0 Pro is capable of solving the complex theoretical questions in CS182 with high accuracy and interpretability.

For further information, please see the annotated logs.