

ChatGPT 4o ▾

↑ Share
👤 Add people
...
**1. DDPM/DDIM Fun: From a Gaussian**

Consider generative diffusion models under the simplifying assumption that the target distribution we wish to sample from is a Gaussian with zero mean and small variance  $\sigma^2$ , where  $\sigma^2 \ll 1$ . The forward process is defined to run from time  $t = 0$  to  $t = 1$ .

Suppose the forward diffusion process starts at  $X_0 \sim \mathcal{N}(0, \sigma^2)$  and at each small interval of length  $\Delta t$  adds independent Gaussian noise with mean zero and variance  $\Delta t \ll \sigma^2$ .

- (a) **What is the distribution of  $X_1$ ?**

(HINT: Remember, you can think of  $\Delta t = \frac{1}{T}$  for some large  $T$  and  $X_1$  as the final result after doing  $T$  steps of adding independent Gaussian noise to the initial realization of  $X_0$ .)

- (b) Looking at the forward diffusion process around time  $t$ , **what are the marginal distributions of  $X_{t-\Delta t}$  and  $X_t$ ?**

- (c) In the previous part, the conditional distribution of  $(X_{t-\Delta t} | X_t = x_t) \sim \mathcal{N}\left(\frac{\sigma^2+t-\Delta t}{\sigma^2+t}x_t, \frac{(\sigma^2+t-\Delta t)\Delta t}{\sigma^2+t}\right)$ . Simplify and approximate the variance  $\frac{(\sigma^2+t-\Delta t)\Delta t}{\sigma^2+t}$  of this conditional distribution when  $\Delta t \ll \sigma^2 \ll 1$ .

Can you please the first 3 parts of this homework question?

This was critical: I passed the question as an image because there is a lot of mathematics that gets wiped away if I copy and paste the raw text from the PDF

Here's a complete step-by-step solution to parts (a), (b), and (c) from the image titled "DDPM/DDIM Fun: From a Gaussian":

---

**(a) What is the distribution of  $X_1$ ?**

We are told that the forward process starts at:

$$X_0 \sim \mathcal{N}(0, \sigma^2)$$

At each small interval  $\Delta t$ , we add **independent Gaussian noise**  $\sim \mathcal{N}(0, \Delta t)$ . Let  $T = \frac{1}{\Delta t}$  be the number of steps from  $t = 0$  to  $t = 1$ . Then, over  $T$  steps, the noise accumulates.

Using the fact that the sum of independent Gaussians is another Gaussian with variances adding up:

$$X_1 = X_0 + \sum_{i=1}^T \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \Delta t)$$

$$\text{Var}(X_1) = \text{Var}(X_0) + T \cdot \Delta t = \sigma^2 + 1$$

 **Final answer:**

$$X_1 \sim \mathcal{N}(0, \sigma^2 + 1)$$


---

**(b) What are the marginal distributions of  $X_{t-\Delta t}$  and  $X_t$ ?**

Let the process be defined as:

$$X_t = X_{t-\Delta t} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \Delta t)$$

Suppose the initial variable is  $X_0 \sim \mathcal{N}(0, \sigma^2)$ , and we have a continuous-time Brownian motion with additive Gaussian noise.

Let's define:

$$\text{Var}(X_t) = \sigma^2 + t$$

$$\text{Var}(X_{t-\Delta t}) = \sigma^2 + t - \Delta t$$

 **Final answer:**

$$X_{t-\Delta t} \sim \mathcal{N}(0, \sigma^2 + t - \Delta t), \quad X_t \sim \mathcal{N}(0, \sigma^2 + t)$$


---

**(c) Simplify and approximate the variance of the conditional distribution**

We're given:

$$(X_{t-\Delta t} \mid X_t = x_t) \sim \mathcal{N} \left( \frac{\sigma^2 + t - \Delta t}{\sigma^2 + t} x_t, \frac{(\sigma^2 + t - \Delta t) \Delta t}{\sigma^2 + t} \right)$$

We are asked to **simplify and approximate the variance term**:

$$\frac{(\sigma^2 + t - \Delta t) \Delta t}{\sigma^2 + t}$$

Under the assumption that  $\Delta t \ll \sigma^2 \ll 1$ , we can use a first-order approximation.

Let's write:

$$\begin{aligned} \frac{(\sigma^2 + t - \Delta t) \Delta t}{\sigma^2 + t} &= \Delta t \cdot \left( \frac{\sigma^2 + t - \Delta t}{\sigma^2 + t} \right) = \Delta t \cdot \left( 1 - \frac{\Delta t}{\sigma^2 + t} \right) \approx \\ &\quad \Delta t \left( 1 - \frac{\Delta t}{\sigma^2 + t} \right) \end{aligned}$$

Since  $\Delta t \ll 1$ , the second term is very small:

 **Final answer** (approximate variance):

$$\text{Var} \approx \Delta t \left( 1 - \frac{\Delta t}{\sigma^2 + t} \right) \approx \Delta t$$

So the conditional variance is approximately  $\Delta t$ , matching our intuition that only a small bit of uncertainty gets subtracted per step in the reverse process.

Let me know if you'd like part (d) or further intuition about DDPMs and reverse diffusion.



- (d) This part asks you to see what happens if you try to do reverse denoising naively without applying stochastic noise at each reverse diffusion step. Suppose we start with a sample  $X_1 \sim \mathcal{N}(0, 1)$  and iteratively apply only the conditional mean mapping backward to time  $t = 0$  to get a sample of  $\hat{X}_0$ . What is the resulting distribution of  $\hat{X}_0$ ? Remember that  $\Delta t \ll \sigma^2 \ll 1$  if you want to make any approximations or to interpret the result. Show your work.

(HINT 1: Consider  $T = \frac{1}{\Delta t}$  and construct the appropriate product to see how the final  $\hat{X}_0$  is distributed. Do you notice any telescoping?)

(HINT 2: You can use the information given in the next part to check your work here.)

Nice! Next, can you solve part D?

---

Absolutely! Let's work through **Part (d)** step-by-step:

## Problem Recap

We are told:

- Start with  $X_1 \sim \mathcal{N}(0, 1)$
- The forward diffusion is defined as: at each step, we add zero-mean Gaussian noise with variance  $\Delta t$
- In reverse, instead of sampling from the **full conditional**  $p(X_{t-\Delta t} | X_t)$ , we **only apply the conditional mean mapping** (i.e., we drop the stochastic part and only use the mean)

Goal:

- Find the **resulting distribution of  $\hat{X}_0$**  after reversing from  $X_1$  using **only the conditional mean** at each step

 **Step 1: Recall the Conditional Mean**

From part (c), the **mean** of  $X_{t-\Delta t} \mid X_t = x_t$  is:

$$\mathbb{E}[X_{t-\Delta t} \mid X_t = x_t] = \frac{\sigma^2 + t - \Delta t}{\sigma^2 + t} x_t$$

Let us define:

$$\alpha_t := \frac{\sigma^2 + t - \Delta t}{\sigma^2 + t} = 1 - \frac{\Delta t}{\sigma^2 + t}$$

Then, iteratively applying the conditional mean mapping **without any noise**, we get:

$$\hat{X}_{t-\Delta t} = \alpha_t \hat{X}_t$$

So, going backwards from  $t = 1$  to  $t = 0$  in steps of  $\Delta t$ , the full mapping is a product:

$$\hat{X}_0 = \left( \prod_{k=1}^T \alpha_k \right) X_1 \quad \text{where} \quad T = \frac{1}{\Delta t}$$


---

 **Step 2: Analyze the Product**

We want to evaluate:

$$\hat{X}_0 = \left( \prod_{k=1}^T \frac{\sigma^2 + 1 - k\Delta t}{\sigma^2 + 1 - (k-1)\Delta t} \right) X_1$$

This is a **telescoping product!**

Let's write out the first few terms:

$$\prod_{k=1}^T \frac{\sigma^2 + 1 - k\Delta t}{\sigma^2 + 1 - (k-1)\Delta t} = \frac{\sigma^2 + 1 - \Delta t}{\sigma^2 + 1} \cdot \frac{\sigma^2 + 1 - 2\Delta t}{\sigma^2 + 1 - \Delta t} \cdot \\ \frac{\sigma^2 + 1 - 3\Delta t}{\sigma^2 + 1 - 2\Delta t} \cdots \frac{\sigma^2}{\sigma^2 + \Delta t}$$

So, everything cancels except:

$$\frac{\sigma^2}{\sigma^2 + 1}$$

✓ Therefore:

$$\hat{X}_0 = \frac{\sigma^2}{\sigma^2 + 1} X_1$$


---

### ⌚ Step 3: What is the Distribution of $\hat{X}_0$ ?

Since  $X_1 \sim \mathcal{N}(0, 1)$ , and we multiply it by a constant:

$$\hat{X}_0 \sim \mathcal{N}\left(0, \left(\frac{\sigma^2}{\sigma^2 + 1}\right)^2\right)$$

This has **mean 0** and **variance**:

$$\text{Var}(\hat{X}_0) = \left(\frac{\sigma^2}{\sigma^2 + 1}\right)^2$$


---

### ⌚ Step 4: Approximate the Result for $\sigma^2 \ll 1$

When  $\sigma^2 \ll 1$ , then:

$$\frac{\sigma^2}{\sigma^2 + 1} \approx \sigma^2 \quad \Rightarrow \quad \left( \frac{\sigma^2}{\sigma^2 + 1} \right)^2 \approx \sigma^4$$

 Final Answer:

$$\hat{X}_0 \sim \mathcal{N}(0, \sigma^4)$$

## Intuition and Takeaway

- By skipping the stochastic noise during denoising, we **collapse the uncertainty**.
- The resulting distribution becomes **much narrower**: the variance drops from 1 (at  $X_1$ ) to  $\sigma^4 \ll 1$  at  $\hat{X}_0$
- This shows why **adding noise is critical** during reverse diffusion: without it, you lose diversity and collapse to a single mode.

Let me know if you'd like help with the next part or plotting this!



(e) In the previous part, you should have found that  $\hat{X}_0$  has far too little variance if you don't add independent stochastic noise along the way in DDPM-style reverse diffusion.

Let  $T = \frac{1}{\Delta t}$ . Suppose now that we add independent  $\mathcal{N}(0, \Delta t)$  noise at each of  $T$  reverse diffusion steps. It turns out that

$$\text{Var}(\hat{X}_0) = \left( \frac{\sigma^2}{\sigma^2 + 1} \right)^2 + \Delta t \sum_{k=0}^{T-1} \left( \frac{\sigma^2}{\sigma^2 + k\Delta t} \right)^2. \quad (1)$$

Take the limit  $\Delta t \rightarrow 0$  and write  $\text{Var}(\hat{X}_0)$  involving an integral assuming  $0 < \Delta t \ll \sigma^2 \ll 1$ . And then evaluate it approximately.

(HINT 1: For evaluating the integral,  $\int \frac{1}{(1+t)^2} dt = C - \frac{1}{1+t}$ )

(HINT 2: You know that the answer should come out to be close to our desired variance of  $\sigma^2$  and you can use that to check your work.)

Nice! Now, can you solve part E?

Absolutely! Let's now solve **part (e)** of this DDPM/DDIM derivation.

---

### Problem Summary

We're given the variance of  $\hat{X}_0$  when we **do add** independent noise  $\mathcal{N}(0, \Delta t)$  at every reverse diffusion step:

$$\text{Var}(\hat{X}_0) = \left( \frac{\sigma^2}{\sigma^2 + 1} \right)^2 + \Delta t \sum_{k=0}^{T-1} \left( \frac{\sigma^2}{\sigma^2 + k\Delta t} \right)^2 \quad (1)$$

We are asked to:

1. Take the limit  $\Delta t \rightarrow 0$  (i.e., large  $T = 1/\Delta t$ )
2. Replace the sum with an **integral**
3. Evaluate the expression **approximately**, under the assumption:

$$0 < \Delta t \ll \sigma^2 \ll 1$$


---

### Step 1: Replace the sum with an integral

Recall:

$$\Delta t \sum_{k=0}^{T-1} f(k\Delta t) \xrightarrow{\Delta t \rightarrow 0} \int_0^1 f(t) dt$$

Apply this to equation (1):

$$\text{Var}(\hat{X}_\theta) \approx \left( \frac{\sigma^2}{\sigma^2 + 1} \right)^2 + \int_0^1 \left( \frac{\sigma^2}{\sigma^2 + t} \right)^2 dt$$


---

## Step 2: Evaluate the integral

We need to evaluate:

$$\int_0^1 \left( \frac{\sigma^2}{\sigma^2 + t} \right)^2 dt$$

Make the substitution:

$$u = \sigma^2 + t \quad \Rightarrow \quad du = dt, \quad \text{as } t = 0 \Rightarrow u = \sigma^2, \quad t = 1 \Rightarrow u = \sigma^2 + 1$$

So the integral becomes:

$$\int_{\sigma^2}^{\sigma^2+1} \left( \frac{\sigma^2}{u} \right)^2 du = \sigma^4 \int_{\sigma^2}^{\sigma^2+1} \frac{1}{u^2} du$$

Now integrate:

$$= \sigma^4 \left[ -\frac{1}{u} \right]_{\sigma^2}^{\sigma^2+1} = \sigma^4 \left( \frac{1}{\sigma^2} - \frac{1}{\sigma^2 + 1} \right)$$


---

## Step 3: Simplify and combine

Now combine the two terms in the variance expression:

**1st term:**

$$\left( \frac{\sigma^2}{\sigma^2 + 1} \right)^2 = \frac{\sigma^4}{(\sigma^2 + 1)^2}$$

**2nd term:**

$$\sigma^4 \left( \frac{1}{\sigma^2} - \frac{1}{\sigma^2 + 1} \right) = \sigma^2 \left( 1 - \frac{\sigma^2}{\sigma^2 + 1} \right) = \sigma^2 \left( \frac{1}{\sigma^2 + 1} \right)$$

So:

$$\text{Var}(\hat{X}_0) \approx \frac{\sigma^4}{(\sigma^2 + 1)^2} + \frac{\sigma^2}{\sigma^2 + 1}$$


---

 **Step 4: Approximate for  $\sigma^2 \ll 1$**

Let's now approximate this assuming  $\sigma^2 \ll 1$ :

**Term 1:**

$$\frac{\sigma^4}{(\sigma^2 + 1)^2} \approx \sigma^4 \quad (\text{since denominator } \approx 1)$$

**Term 2:**

$$\frac{\sigma^2}{\sigma^2 + 1} \approx \sigma^2 \quad (\text{since } \sigma^2 \ll 1)$$

So:

$$\text{Var}(\hat{X}_0) \approx \sigma^2 + \sigma^4 \approx \boxed{\sigma^2(1 + \sigma^2)} \approx \boxed{\sigma^2}$$

 **Final Answer:**

$$\text{Var}(\hat{X}_0) \approx \sigma^2$$

## ✨ Intuition / Interpretation

This result tells us:

- When we include noise at each reverse step, the final variance correctly recovers the original variance of the data distribution  $\mathcal{N}(0, \sigma^2)$
- This confirms that the **DDPM reverse process** works — as long as we preserve both the **mean** and **variance** evolution over time

Let me know if you'd like a figure or derivation summary!



(f) Notice that the conditional mean step in DDPM is approximately a  $\Delta t/t$  step toward the ideal predictor  $\hat{X}_0^* = \frac{\sigma^2}{\sigma^2 + t} X_t$ . Namely that

$$\hat{X}_{t-\Delta t}^{\text{DDPM}} = \hat{x}_t + \frac{\Delta t}{t} (\hat{X}_0^* - \hat{x}_t) + \text{noise} = \left(1 - \frac{\Delta t}{\sigma^2 + t}\right) \hat{x}_t + \text{noise}$$

This can be incrementally viewed as a deterministic DDPM step of  $-\frac{\Delta t}{\sigma^2 + t} x_t$  together with a noise step.

DDIM takes no noise step, but a smaller time-varying deterministic step:

$$x_{t-\Delta t}^{\text{DDIM}} = x_t + \eta(t, \Delta t) (\text{deterministic DDPM step}) \quad \text{with } \eta(t, \Delta t) = \frac{\sqrt{t}}{\sqrt{t} - \Delta t + \sqrt{t}}$$

Describe briefly, if you were simply given a dataset of samples  $s_1, s_2, \dots, s_n$  drawn from the same distribution as our desired  $X_0$  distribution, how you would train a neural network to estimate the function  $g(x_t, t)$  that in this case, turns out to be  $\frac{\sigma^2}{\sigma^2 + t} x_t$  in analytic form. Specifically, what are the inputs to the neural net, how do you generate a batch of them, and how would you compute a loss on the outputs of the neural net for running an optimization algorithm like AdamW to set the parameters of the neural net?

Nice! Now, please solve part f

Great! Let's now tackle **Part (f)**, which is about using a neural network to estimate the ideal predictor in a DDPM-style setup.

---

## ✓ Objective Recap

We are given that the *ideal denoiser* at time  $t$  is the function:

$$g(x_t, t) = \frac{\sigma^2}{\sigma^2 + t} x_t$$

We want to train a neural network to **approximate this function**.

You are asked:

If you had a dataset of samples  $s_1, s_2, \dots, s_n$  drawn from the target distribution  $X_0 \sim \mathcal{N}(0, \sigma^2)$ , how would you train a neural net to learn this function?

---

### 12 34 Step 1: What are the inputs to the neural net?

We want the neural network to learn the mapping:

$$(x_t, t) \mapsto \hat{x}_0$$

So the inputs to the neural network are:

- **Noisy sample**  $x_t \in \mathbb{R}^d$
- **Time**  $t \in \mathbb{R}$ , typically scaled to  $[0, 1]$  and embedded (e.g., via sinusoidal or learned embeddings)

👉 **Inputs to the neural net:** A batch of tuples  $(x_t, t)$

## Step 2: How do you generate a batch of them?

We assume you are given clean samples  $s_i \sim X_0 \sim \mathcal{N}(0, \sigma^2)$

To create training pairs  $(x_t, t, x_0)$ :

1. **Sample a timestep**  $t \sim \text{Uniform}(0, 1)$  for each clean sample
2. **Add forward Gaussian noise** to simulate the DDPM forward process:

$$x_t = s_i + \epsilon, \quad \epsilon \sim \mathcal{N}(0, t)$$

This makes  $x_t \sim \mathcal{N}(0, \sigma^2 + t)$ , consistent with the forward process.

3. The input to the network is  $(x_t, t)$ , and the target is:

$$x_0^* = \frac{\sigma^2}{\sigma^2 + t} x_t$$

This gives you a batch of training data:  $\{(x_t^{(i)}, t^{(i)}, x_0^{*(i)})\}_{i=1}^B$

## Step 3: How do you compute the loss?

The neural net is trained to output an estimate  $\hat{x}_0 \approx x_0^*$

So we use a standard MSE loss:

$$\mathcal{L} = \frac{1}{B} \sum_{i=1}^B \left\| \text{NN}(x_t^{(i)}, t^{(i)}) - x_0^{*(i)} \right\|_2^2$$

This loss can then be used with an optimizer like **AdamW** to update the neural net parameters.

## ✓ Final Answer (in words)

To train a neural network to estimate the function  $g(x_t, t) = \frac{\sigma^2}{\sigma^2 + t}x_t$ :

**1. Inputs to the network** are pairs  $(x_t, t)$ , where:

- $x_t = x_0 + \epsilon$ , with  $x_0 \sim \mathcal{N}(0, \sigma^2)$ , and  $\epsilon \sim \mathcal{N}(0, t)$
- $t \sim \text{Uniform}(0, 1)$

**2. To generate a batch**, sample  $x_0^{(i)}$ , choose timestep  $t^{(i)}$ , and compute:

$$x_t^{(i)} = x_0^{(i)} + \epsilon^{(i)}, \quad \text{target: } x_0^{*(i)} = \frac{\sigma^2}{\sigma^2 + t^{(i)}} x_t^{(i)}$$

**3. Loss:** Use MSE between the predicted  $\hat{x}_0$  and the true  $x_0^*$ :

$$\mathcal{L} = \mathbb{E}_{x_t, t} \left[ \left\| \text{NN}(x_t, t) - \frac{\sigma^2}{\sigma^2 + t} x_t \right\|^2 \right]$$

Then use any optimizer (e.g. AdamW) to minimize this loss and train the model.

Let me know if you'd like the PyTorch code for this or a visual diagram!



- (g) Assuming you had exact analytic form access to  $g(x_t, t) = \frac{\sigma^2}{\sigma^2 + t}x_t$  in the previous part, **approximate the DDIM step for  $\Delta t \ll t$ , with no assumption about  $\sigma^2$  vs  $t$ .**  
*(HINT: Use the fact that  $\Delta t \ll t$  to approximate  $\eta(t, \Delta t)$  as an appropriate constant.)*

Can you please solve part g now?

Absolutely! Let's solve **Part (g)** — which asks you to **approximate the DDIM step** when  $\Delta t \ll t$ , using the known analytical form of  $g(x_t, t) = \frac{\sigma^2}{\sigma^2 + t} x_t$ , but **without assuming anything about  $\sigma^2 \ll t$** .

---

### Step 1: Recall the DDIM update rule

The DDIM step is given by:

$$x_{t-\Delta t}^{\text{DDIM}} = x_t + \eta(t, \Delta t) \cdot (g(x_t, t) - x_t)$$

We are told  $g(x_t, t) = \frac{\sigma^2}{\sigma^2 + t} x_t$ , so:

$$g(x_t, t) - x_t = \left( \frac{\sigma^2}{\sigma^2 + t} - 1 \right) x_t = \left( -\frac{t}{\sigma^2 + t} \right) x_t$$

So the update becomes:

$$x_{t-\Delta t}^{\text{DDIM}} = x_t - \eta(t, \Delta t) \cdot \left( \frac{t}{\sigma^2 + t} \right) x_t = x_t \left( 1 - \eta(t, \Delta t) \cdot \frac{t}{\sigma^2 + t} \right)$$


---

### Step 2: Approximate $\eta(t, \Delta t)$ for small $\Delta t$

From earlier parts, we know:

$$\eta(t, \Delta t) = \frac{\sqrt{t}}{\sqrt{t - \Delta t} + \sqrt{t}} = \left( 1 + \sqrt{1 - \frac{\Delta t}{t}} \right)^{-1}$$

Now for  $\Delta t \ll t$ , use the first-order Taylor expansion:

$$\sqrt{t - \Delta t} \approx \sqrt{t} \left( 1 - \frac{\Delta t}{2t} \right)$$

Then:

$$\eta(t, \Delta t) = \frac{\sqrt{t}}{\sqrt{t - \Delta t} + \sqrt{t}} \approx \frac{\sqrt{t}}{\sqrt{t} \left( 1 - \frac{\Delta t}{2t} \right) + \sqrt{t}} = \frac{1}{2 - \frac{\Delta t}{2t}} \approx \frac{1}{2} + \frac{\Delta t}{8t}$$

So to first order:

$$\eta(t, \Delta t) \approx \frac{1}{2}$$

---

### Step 3: Final DDIM Step Approximation

Substitute  $\eta(t, \Delta t) \approx \frac{1}{2}$  into the update rule:

$$x_{t-\Delta t}^{\text{DDIM}} \approx x_t \left( 1 - \frac{1}{2} \cdot \frac{t}{\sigma^2 + t} \right) = x_t \left( \frac{\sigma^2 + t - \frac{t}{2}}{\sigma^2 + t} \right) = x_t \left( \frac{\sigma^2 + \frac{t}{2}}{\sigma^2 + t} \right)$$


---

### Final Answer:

$$x_{t-\Delta t}^{\text{DDIM}} \approx x_t \cdot \left( \frac{\sigma^2 + \frac{t}{2}}{\sigma^2 + t} \right)$$

---

 **Interpretation:**

- This DDIM update shrinks the signal toward zero, depending on both  $t$  and  $\sigma^2$
- For large  $t$ , this ratio becomes close to  $\frac{1}{2}$ , as expected
- For small  $t$ , it's closer to 1, meaning less denoising is needed early on

Let me know if you want this turned into pseudocode or extended to higher-order terms!



**It made an error here where it vastly over complicated the question so I had to intervene. Note: in the later parts, it does this quite frequently**

Are you sure? I think you get the  $1/2$  term correctly but try checking that the other term is correct

You're right to be skeptical — let's **double-check** the derivation of the DDIM update step in **Part (g)** and **rederive it carefully**, keeping track of all approximations.

---

 **Step 1: DDIM Update Rule**

From part (f), the DDIM step is:

$$x_{t-\Delta t}^{\text{DDIM}} = x_t + \eta(t, \Delta t) \cdot (g(x_t, t) - x_t)$$

And from the problem setup:

$$g(x_t, t) = \frac{\sigma^2}{\sigma^2 + t} x_t \quad \Rightarrow \quad g(x_t, t) - x_t = \left( \frac{\sigma^2}{\sigma^2 + t} - 1 \right) x_t = -\frac{t}{\sigma^2 + t} x_t$$

So the update becomes:

$$x_{t-\Delta t}^{\text{DDIM}} = x_t - \eta(t, \Delta t) \cdot \frac{t}{\sigma^2 + t} x_t = x_t \left( 1 - \eta(t, \Delta t) \cdot \frac{t}{\sigma^2 + t} \right)$$


---

## ✓ Step 2: Approximate $\eta(t, \Delta t)$ for small $\Delta t \ll t$

Recall the exact expression:

$$\eta(t, \Delta t) = \frac{\sqrt{t}}{\sqrt{t - \Delta t} + \sqrt{t}}$$

As before, expand  $\sqrt{t - \Delta t}$  using Taylor:

$$\sqrt{t - \Delta t} = \sqrt{t} \left( 1 - \frac{\Delta t}{2t} + \mathcal{O}(\Delta t^2) \right)$$

Then:

$$\eta(t, \Delta t) = \frac{\sqrt{t}}{\sqrt{t - \Delta t} + \sqrt{t}} = \frac{1}{\left( 1 - \frac{\Delta t}{2t} \right) + 1} = \frac{1}{2 - \frac{\Delta t}{2t}} = \frac{1}{2} \left( 1 + \frac{\Delta t}{4t} + \mathcal{O}(\Delta t^2) \right)$$

So up to first order:

$$\eta(t, \Delta t) \approx \frac{1}{2} + \frac{\Delta t}{8t}$$

---

## ✓ Step 3: Plug this into the DDIM update

We have:

$$x_{t-\Delta t}^{\text{DDIM}} = x_t \left( 1 - \eta(t, \Delta t) \cdot \frac{t}{\sigma^2 + t} \right)$$

Substitute in  $\eta(t, \Delta t) \approx \frac{1}{2} + \frac{\Delta t}{8t}$ :

$$\begin{aligned} 1 - \left( \frac{1}{2} + \frac{\Delta t}{8t} \right) \cdot \frac{t}{\sigma^2 + t} &= 1 - \left( \frac{t}{2(\sigma^2 + t)} + \frac{\Delta t}{8(\sigma^2 + t)} \right) \\ &= \frac{\sigma^2 + t - \frac{t}{2} - \frac{\Delta t}{8}}{\sigma^2 + t} = \frac{\sigma^2 + \frac{t}{2} - \frac{\Delta t}{8}}{\sigma^2 + t} \end{aligned}$$


---

## ✓ Final Answer

$$x_{t-\Delta t}^{\text{DDIM}} \approx x_t \cdot \left( \frac{\sigma^2 + \frac{t}{2} - \frac{\Delta t}{8}}{\sigma^2 + t} \right)$$

This includes the **correct linear correction in  $\Delta t$**  and avoids assuming anything about the size of  $\sigma^2$  relative to  $t$ .

---

## 🧠 Interpretation

- When  $\Delta t \rightarrow 0$ , the factor becomes  $\frac{\sigma^2 + t/2}{\sigma^2 + t}$ , as expected.
- When  $\sigma^2 \rightarrow 0$ , it reduces to  $\frac{1}{2} - \frac{\Delta t}{8t}$ , which aligns with the intuition that DDIM steps shrink  $x_t$  over time.

Thanks for catching that detail — let me know if you'd like this visualized or implemented in code!



$$x_{t-\Delta t}^{\text{DDIM}} = \left(1 - \frac{\eta(t, \Delta t) \Delta t}{\sigma^2 + t}\right) x_t$$

Try starting your answer in this form:

Great! Given your preferred starting point, let's begin with the DDIM step in the following form:

$$x_{t-\Delta t}^{\text{DDIM}} = \left(1 - \frac{\eta(t, \Delta t) \Delta t}{\sigma^2 + t}\right) x_t$$

Now we will approximate this expression under the assumption that:

- $\Delta t \ll t$
- No assumption is made about  $\sigma^2$  vs.  $t$

### Step 1: Approximate $\eta(t, \Delta t)$

Recall:

$$\eta(t, \Delta t) = \frac{\sqrt{t}}{\sqrt{t - \Delta t} + \sqrt{t}}$$

Use the first-order Taylor approximation:

$$\sqrt{t - \Delta t} \approx \sqrt{t} \left( 1 - \frac{\Delta t}{2t} \right) \Rightarrow \eta(t, \Delta t) = \frac{\sqrt{t}}{\sqrt{t - \Delta t} + \sqrt{t}} \approx \frac{1}{2 - \frac{\Delta t}{2t}} = \frac{1}{2} + \frac{\Delta t}{8t}$$

So we substitute:

$$\eta(t, \Delta t) \approx \frac{1}{2} + \frac{\Delta t}{8t}$$


---

## ✓ Step 2: Plug into the DDIM update

We now substitute this approximation into the original formula:

$$x_{t-\Delta t}^{\text{DDIM}} = \left( 1 - \frac{\left( \frac{1}{2} + \frac{\Delta t}{8t} \right) \Delta t}{\sigma^2 + t} \right) x_t = \left( 1 - \left[ \frac{\Delta t}{2(\sigma^2 + t)} + \frac{\Delta t^2}{8t(\sigma^2 + t)} \right] \right) x_t$$

Now simplify the expression:

$$x_{t-\Delta t}^{\text{DDIM}} = \left( \frac{\sigma^2 + t - \frac{\Delta t}{2} - \frac{\Delta t^2}{8t}}{\sigma^2 + t} \right) x_t$$


---

## ✓ Final Answer (boxed)

$$x_{t-\Delta t}^{\text{DDIM}} \approx \left( \frac{\sigma^2 + t - \frac{\Delta t}{2} - \frac{\Delta t^2}{8t}}{\sigma^2 + t} \right) x_t$$

This expression is accurate up to second order in  $\Delta t$ , and makes **no assumption** about the size of  $\sigma^2$  relative to  $t$ , as requested.

Let me know if you'd like the same result in a LaTeX format or as a function in PyTorch!



Ok nice! now, try factoring the terms out

Great! Let's now take our boxed expression from before and **factor it** for cleaner interpretation.

---

### ⌚ Starting Expression

We previously derived:

$$x_{t-\Delta t}^{\text{DDIM}} \approx \left( \frac{\sigma^2 + t - \frac{\Delta t}{2} - \frac{\Delta t^2}{8t}}{\sigma^2 + t} \right) x_t$$

---

### 🔧 Step 1: Factor out $\frac{1}{\sigma^2+t}$

Let's write:

$$x_{t-\Delta t}^{\text{DDIM}} \approx \left( 1 - \frac{\frac{\Delta t}{2} + \frac{\Delta t^2}{8t}}{\sigma^2 + t} \right) x_t$$

We can now **factor out**  $\frac{\Delta t}{2(\sigma^2+t)}$  from the numerator:

$$\frac{\frac{\Delta t}{2} + \frac{\Delta t^2}{8t}}{\sigma^2 + t} = \frac{\Delta t}{2(\sigma^2 + t)} \left(1 + \frac{\Delta t}{4t}\right)$$

 **Final Factored Answer:**

$$x_{t-\Delta t}^{\text{DDIM}} \approx \left(1 - \frac{\Delta t}{2(\sigma^2 + t)} \left(1 + \frac{\Delta t}{4t}\right)\right) x_t$$

The final solution doesn't exactly match the answer sheet but it's the same thing, just written differently!

 **Interpretation:**

- This is now factored into a clean **step size** term that controls how much the update shrinks  $x_t$
- The expression clearly shows how:
  - The update shrinks more with larger  $\Delta t$
  - The curvature correction (the  $\Delta t^2$  term) becomes significant only if  $t$  is very small

Let me know if you want a comparison with Euler steps or a plot showing this scaling behavior!



A cool thing about 4o is that it always tries to extract a broader takeaway at the end of an explanation  
—> good learning tool :)

- (h) Compute  $\hat{X}_0$  from the random sample  $\hat{X}_1$  (drawn from  $\mathcal{N}(0, 1)$ ) by applying all  $T = \frac{1}{\Delta t}$  DDIM steps from the previous part and expressing the total contraction as a product. Then turn it into a sum via logarithms and take the  $\Delta t \rightarrow 0$  limit, using integrals as needed to evaluate the resulting distribution for  $\hat{X}_0$ . **Show your work.**

(HINT 1: You know that the answer is supposed to be approximately  $\mathcal{N}(0, \sigma^2)$  under our assumption that  $\sigma^2 \ll 1$ . Use this to check your work.)

(HINT 2: Remember  $\ln(1 - x) \approx -x$  if  $0 < x \ll 1$ .)

(HINT 3: The log of a limit is the limit of logs.)

Ok yeah that's basically correct. Now, try solving part F

Let's solve **Part (h)** step-by-step.

---

## Problem Overview

We are to compute the distribution of:

$$\hat{X}_0 = \text{DDIM-step-reversed}(\hat{X}_1), \quad \hat{X}_1 \sim \mathcal{N}(0, 1)$$

by applying  $T = \frac{1}{\Delta t}$  **DDIM steps**, each of which shrinks  $x_t$  by a factor of:

$$x_{t-\Delta t} = \left(1 - \frac{\eta(t, \Delta t) \Delta t}{\sigma^2 + t}\right) x_t$$

From part (g) and (factored form in part g'), this becomes (up to second order in  $\Delta t$ ):

$$x_{t-\Delta t} \approx \left(1 - \frac{\Delta t}{2(\sigma^2 + t)} \left(1 + \frac{\Delta t}{4t}\right)\right) x_t$$


---

## Step 1: Express total contraction as a product

Let the contraction from  $t = 1 \rightarrow 0$  be:

$$\hat{X}_0 = \hat{X}_1 \cdot \prod_{k=0}^{T-1} \left(1 - \frac{\Delta t}{2(\sigma^2 + t_k)} \left(1 + \frac{\Delta t}{4t_k}\right)\right), \quad t_k = 1 - k\Delta t$$

We're applying  $T = 1/\Delta t$  steps in reverse from  $t = 1 \rightarrow 0$ , so this product represents the total



- (h) Compute  $\hat{X}_0$  from the random sample  $\hat{X}_1$  (drawn from  $\mathcal{N}(0, 1)$ ) by applying all  $T = \frac{1}{\Delta t}$  DDIM steps from the previous part and expressing the total contraction as a product. Then turn it into a sum via logarithms and take the  $\Delta t \rightarrow 0$  limit, using integrals as needed to evaluate the resulting distribution for  $\hat{X}_0$ . **Show your work.**

*(HINT 1: You know that the answer is supposed to be approximately  $\mathcal{N}(0, \sigma^2)$  under our assumption that  $\sigma^2 \ll 1$ . Use this to check your work.)*

*(HINT 2: Remember  $\ln(1 - x) \approx -x$  if  $0 < x \ll 1$ .)*

*(HINT 3: The log of a limit is the limit of logs.)*

Sorry, try solving part H. My bad I made a mistake in what I called it.

No worries — let's solve **Part (h)** now!

---

### Goal:

Given a sample  $\hat{X}_1 \sim \mathcal{N}(0, 1)$ , we apply  $T = \frac{1}{\Delta t}$  **DDIM steps** from time  $t = 1 \rightarrow 0$ , each of which contracts  $x_t$ , and we want to compute the **final distribution of  $\hat{X}_0$**  after all the steps.

We'll:

1. Express total contraction as a product
  2. Take logs to convert to a sum
  3. Take the  $\Delta t \rightarrow 0$  limit (i.e. Riemann sum  $\rightarrow$  integral)
  4. Recover the variance of  $\hat{X}_0$  in closed form
-

## Step 1: DDIM Contraction Per Step

From **Part (g)** and (factored form), each step contracts the signal by:

$$x_{t-\Delta t}^{\text{DDIM}} = \left(1 - \frac{\Delta t}{2(\sigma^2 + t)}\right) x_t \quad (\text{to first order in } \Delta t)$$

So applying this over  $T = 1/\Delta t$  steps from  $t = 1 \rightarrow 0$ , the total contraction is:

$$\hat{X}_0 = \hat{X}_1 \cdot \prod_{k=0}^{T-1} \left(1 - \frac{\Delta t}{2(\sigma^2 + t_k)}\right), \quad \text{where } t_k = 1 - k\Delta t$$


---

## Step 2: Take Logs

Take logarithms:

$$\ln \hat{X}_0 = \ln \hat{X}_1 + \sum_{k=0}^{T-1} \ln \left(1 - \frac{\Delta t}{2(\sigma^2 + t_k)}\right)$$

Use the first-order expansion  $\ln(1 - x) \approx -x$  for small  $x$ :

$$\approx \ln \hat{X}_1 - \sum_{k=0}^{T-1} \frac{\Delta t}{2(\sigma^2 + t_k)}$$


---

## Step 3: Convert Sum to Integral

Now let  $\Delta t \rightarrow 0$ , turning the sum into a Riemann integral:

$$\sum_{k=0}^{T-1} \frac{\Delta t}{2(\sigma^2 + t_k)} \rightarrow \int_0^1 \frac{1}{2(\sigma^2 + t)} dt$$

This gives:

$$\begin{aligned}\ln \hat{X}_0 &= \ln \hat{X}_1 - \int_0^1 \frac{1}{2(\sigma^2 + t)} dt \\ &= \ln \hat{X}_1 - \left[ \frac{1}{2} \ln(\sigma^2 + t) \right]_0^1 = \ln \hat{X}_1 - \frac{1}{2} \ln \left( \frac{\sigma^2 + 1}{\sigma^2} \right)\end{aligned}$$

Exponentiate:

$$\hat{X}_0 = \hat{X}_1 \cdot \exp \left( -\frac{1}{2} \ln \left( \frac{\sigma^2 + 1}{\sigma^2} \right) \right) = \hat{X}_1 \cdot \left( \frac{\sigma^2}{\sigma^2 + 1} \right)^{1/2}$$


---

#### ✓ Step 4: Final Distribution of $\hat{X}_0$

Since  $\hat{X}_1 \sim \mathcal{N}(0, 1)$ , multiplying by a constant gives:

$$\hat{X}_0 \sim \mathcal{N} \left( 0, \frac{\sigma^2}{\sigma^2 + 1} \right)$$

---

#### Sanity Check: Assume $\sigma^2 \ll 1$

If  $\sigma^2 \ll 1$ , then:

$$\frac{\sigma^2}{\sigma^2 + 1} \approx \sigma^2$$

 Which matches the expected answer:

$$\hat{X}_0 \sim \mathcal{N}(0, \sigma^2)$$

## ✓ Final Boxed Result:

$$\hat{X}_0 = \left( \frac{\sigma^2}{\sigma^2 + 1} \right)^{1/2} \cdot \hat{X}_1 \quad \text{so} \quad \hat{X}_0 \sim \mathcal{N} \left( 0, \frac{\sigma^2}{\sigma^2 + 1} \right)$$

Let me know if you want a visual intuition or a derivation for the mean as well!



It was able to solve question 1 with almost no issues!

**2 Rewarding Meiosis.** Now, we use the SFT model to generate pairs of scores  $(x_1, x_2) = \pi^{\text{SFT}}(x)$  for various groups  $x$ . Human females rank these pairs, producing a dataset of preferences  $Z^* = \{(x_1, x_2, y)\}_{(x_1, x_2) \in \mathcal{X}^2}$ , where  $y \in \{0, 1\}$  indicates whether the female prefers the first output meiosis  $x_1$  over  $x_2$ , respectively. Crucially, we assume these preferences are generated by some latent function  $\pi^*(x_1, x_2)$ , which we do not have access to. Next, we implement a piecewise linear model  $\pi(x_1, x_2)$  to predict the probability of each outcome. However, we want to make sure our preferences, which are inherently discrete ( $y_i \in \{0, 1\}$ ), align a reward model that outputs continuous

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

What is remarkable is that it doesn't use thinking mode so it's not checking its work and its basing its answers off of what it thinks is the best answer at a first pass. I only had to nudge it once

Wow! That was a tough one. Now, can you help me solve the second question on the homework? It's very long, so be sure to read all of it. Then, solve part a

Absolutely! Let's walk through **Question 2 (a)** step-by-step.

## Problem Summary

We're given an optimization problem:

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

where:

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

and the question is:

**What is the optimal  $p^*$  that minimizes  $\mathbb{D}_{\text{KL}}(p \| q)$ ?**

---

### Intuition

KL divergence  $\mathbb{D}_{\text{KL}}(p \| q)$  is always **non-negative** and equals **zero if and only if**  $p(x) = q(x)$  for all  $x$ .

Thus, we're minimizing a function whose global minimum is known:

$$\mathbb{D}_{\text{KL}}(p \| q) = 0 \quad \text{when } p(x) = q(x)$$


---

## Final Answer:

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

This is the **unique minimizer**, since KL divergence is convex in  $p$  and achieves its minimum when  $p = q$ .

---

## Sanity Check

If we plug in  $p^*(x) = q(x)$ , then:

$$\mathbb{D}_{\text{KL}}(p^* \| q) = \sum_x q(x) \log \left( \frac{q(x)}{q(x)} \right) = \sum_x q(x) \cdot 0 = 0$$

 Confirmed. While thinking mode isn't enabled, it usually always checks its answer.  
However, I haven't seen a case where it does a check and then realizes the answer is wrong  
so i'm wondering what happens in that case...

Let me know when you're ready to move on to part (b) — it looks like we'll be diving into Direct Preference Optimization!



- (b) As discussed earlier, we wish to find a language model policy  $\pi_\theta$  that maximizes the expected reward while preventing the model from drifting too far from the reference model:

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

Show that the optimal solution  $\pi_{\theta^*}(y | x)$  takes the form:

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

where  $Z(x) = \sum_y \pi_{\text{ref}}(y | x) \exp \left( \frac{1}{\beta} r_\phi(x, y) \right)$  is the partition function.

Nice, you got that right! Now, try solving part b

Excellent! Let's solve **Part (b)**.

---

## Problem Setup

We are asked to **maximize** the following objective over  $\pi_\theta(y|x)$ :

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

Our goal is to derive the form of the optimal policy  $\pi_\theta^*(y|x)$ , and show it equals:

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


---

## Step-by-step Derivation

We'll treat this as a **constrained optimization problem** over the space of conditional distributions  $\pi_\theta(y|x)$ , for fixed  $x$ .

We'll fix  $x$  and drop the  $x$ -dependence for clarity during derivation, and restore it later.

### Objective:

$$\mathcal{L}[\pi] = \sum_y \pi(y) r_\phi(x, y) - \beta \sum_y \pi(y) \log \frac{\pi(y)}{\pi_{\text{ref}}(y)}$$

We include a Lagrange multiplier  $\lambda$  to enforce the constraint  $\sum_y \pi(y) = 1$ :

$$\mathcal{J}[\pi, \lambda] = \sum_y \pi(y) r_\phi(x, y) - \beta \sum_y \pi(y) \log \frac{\pi(y)}{\pi_{\text{ref}}(y)} - \lambda \left( \sum_y \pi(y) - 1 \right)$$


---

### Take the functional derivative and set to zero

We differentiate  $\mathcal{J}$  w.r.t.  $\pi(y)$ :

$$\frac{\partial \mathcal{J}}{\partial \pi(y)} = r_\phi(x, y) - \beta \left( \log \frac{\pi(y)}{\pi_{\text{ref}}(y)} + 1 \right) - \lambda = 0$$

Rearranging:

$$\log \frac{\pi(y)}{\pi_{\text{ref}}(y)} = \frac{1}{\beta} (r_\phi(x, y) - \lambda - \beta)$$

Exponentiate both sides:

$$\frac{\pi(y)}{\pi_{\text{ref}}(y)} = \exp \left( \frac{1}{\beta} (r_\phi(x, y) - \lambda - \beta) \right) = C \cdot \exp \left( \frac{1}{\beta} r_\phi(x, y) \right) \quad \text{where } C = \exp \left( -\frac{\lambda + \beta}{\beta} \right)$$

Therefore:

$$\pi(y) = C \cdot \pi_{\text{ref}}(y) \cdot \exp \left( \frac{1}{\beta} r_\phi(x, y) \right)$$


---

 **Enforce normalization:**  $\sum_y \pi(y) = 1$ 

$$1 = \sum_y \pi(y) = C \sum_y \pi_{\text{ref}}(y) \exp\left(\frac{1}{\beta} r_\phi(x, y)\right) \Rightarrow C = \frac{1}{Z(x)}$$

where:

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


---

 **Final Answer:**

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

Which is exactly what we were asked to show.

---

Let me know when you're ready for part (c) — we'll likely begin to connect this to Direct Preference Optimization!



Had to help it out here, because it forgot to frame the question as a minimization problem

Ok great! You got almost all the way there. Try reorganizing it as a minimization function of some expected value

Excellent observation — yes, we can reorganize the problem into a **minimization form**, which makes the connection to KL-divergence-based optimization (and DPO) even clearer.

Let's go through it again with the goal of **recasting the objective as a minimization of an expected KL-like divergence**, which is often how these problems are framed in variational inference and energy-based models.

---

### ✓ Step 1: Original Objective (from problem)

We're trying to **maximize** the following:

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

Let's isolate the **inner optimization problem per input  $x$** . For fixed  $x$ , define:

$$\mathcal{L}_x(\pi_\theta) = \mathbb{E}_{y \sim \pi_\theta(y|x)} [r_\phi(x, y)] - \beta \mathbb{D}_{\text{KL}}(\pi_\theta(y|x) \| \pi_{\text{ref}}(y|x))$$

We can now **reorganize this as a minimization** problem by multiplying by  $-1$ :

$$\min_{\pi_\theta(y|x)} \mathbb{D}_{\text{KL}}(\pi_\theta(y|x) \| \pi_{\text{ref}}(y|x)) - \frac{1}{\beta} \mathbb{E}_{y \sim \pi_\theta(y|x)} [r_\phi(x, y)]$$

This is now clearly a **KL-regularized expected utility minimization**. We're searching for a policy  $\pi_\theta(y|x)$  that stays close to  $\pi_{\text{ref}}(y|x)$ , but leans toward outputs with higher reward.

---

## ✓ Step 2: Solve with variational calculus (as before)

As we derived earlier, this is minimized by:

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

where the partition function ensures normalization:

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

## ✓ Summary

So, in **minimization form**, the problem becomes:

$$\min_{\pi_\theta(y|x)} \mathbb{E}_{x \sim \mathcal{D}} \left[ \mathbb{D}_{\text{KL}} (\pi_\theta(y|x) \| \pi_{\text{ref}}(y|x)) - \frac{1}{\beta} \mathbb{E}_{y \sim \pi_\theta(y|x)} [r_\phi(x, y)] \right]$$

This is equivalent to **Relative Entropy Policy Search** or the **softmax of rewards** from control theory. And the solution is a **Boltzmann (exponentiated reward)** distribution reweighted from the reference model.

and as you can see, it helps the model out  
but it doesn't completely figure out the answer  
(in the correct format)

Let me know when you're ready to connect this to Direct Preference Optimization (DPO) in part (c)!



Could you use your knowledge of what  $Z(x)$  is to help simplify the answer further?

Absolutely! This is a great follow-up — let's now use our knowledge of the **partition function**  $Z(x)$  to express the optimal policy  $\pi_\theta^*(y|x)$  as the **minimizer of an expected log-ratio**, which will be helpful for connecting to **Direct Preference Optimization (DPO)** later.

---

### ✓ Step 1: Recall the Optimal Policy Expression

From the previous derivation, the optimal policy is:

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

and the partition function is:

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

---

### ✓ Step 2: Plug into KL-regularized objective

We had the minimization form:

$$\min_{\pi_\theta(y|x)} \mathbb{D}_{\text{KL}}(\pi_\theta(y|x) \| \pi_{\text{ref}}(y|x)) - \frac{1}{\beta} \mathbb{E}_{y \sim \pi_\theta(y|x)} [r_\phi(x, y)]$$

Now plug in the closed-form solution for  $\pi_\theta^*(y|x)$  into this expression:

Let's compute:

$$\mathbb{D}_{\text{KL}}(\pi_\theta^* \| \pi_{\text{ref}}) = \sum_y \pi_\theta^*(y|x) \log \frac{\pi_\theta^*(y|x)}{\pi_{\text{ref}}(y|x)}$$

Substitute:

$$\begin{aligned} \frac{\pi_\theta^*(y|x)}{\pi_{\text{ref}}(y|x)} &= \frac{1}{Z(x)} \exp\left(\frac{1}{\beta} r_\phi(x, y)\right) \Rightarrow \log \frac{\pi_\theta^*(y|x)}{\pi_{\text{ref}}(y|x)} = \\ &- \log Z(x) + \frac{1}{\beta} r_\phi(x, y) \end{aligned}$$

So:

$$\begin{aligned} \mathbb{D}_{\text{KL}}(\pi_\theta^* \| \pi_{\text{ref}}) &= \sum_y \pi_\theta^*(y|x) \left( \frac{1}{\beta} r_\phi(x, y) - \log Z(x) \right) = \\ &\frac{1}{\beta} \mathbb{E}_{y \sim \pi_\theta^*} [r_\phi(x, y)] - \log Z(x) \end{aligned}$$

Now plug back into the objective:

$$\begin{aligned} \mathbb{D}_{\text{KL}}(\pi_\theta^* \| \pi_{\text{ref}}) - \frac{1}{\beta} \mathbb{E}_{\pi_\theta^*} [r_\phi(x, y)] &= \left( \frac{1}{\beta} \mathbb{E} [r_\phi] - \log Z(x) \right) - \frac{1}{\beta} \mathbb{E} [r_\phi] = \\ &- \log Z(x) \end{aligned}$$

## ✓ Final Simplified Objective

Therefore, the **minimum value** of the original objective is:

$$\min_{\pi_\theta(y|x)} \left[ \mathbb{D}_{\text{KL}}(\pi_\theta(y|x) \| \pi_{\text{ref}}(y|x)) - \frac{1}{\beta} \mathbb{E}[r_\phi(x, y)] \right] = -\log Z(x)$$

<sup>^ My hint makes it think that the solution is  $Z(x)$</sup>

And this is achieved **exactly when**:

I think this is a common behavior in gpt-4o where it tries to “glaze”  
the user that they are always correct

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

This behavior is destructive because it leads to simple misunderstandings  
such as here

### 🧠 Why this matters for DPO

This expression shows that the **KL-regularized reward maximization** objective is minimized by reweighting the reference policy via an exponentiated reward, and the **log partition function  $\log Z(x)$**  plays the role of the **log-normalizer** in DPO’s loss.

It’s the exact same structure you get in logistic regression, Boltzmann distributions, and energy-based models.

Let me know when you want to proceed to part (c), which will probably ask us to relate this to preferences and DPO!



$$\begin{aligned}
 &= \min_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}} \left[ \mathbb{E}_{y \sim \pi_\theta(y|x)} \left[ \log \left( \frac{\pi_\theta(y|x)}{\pi^*(y|x)} \right) \right] - \log(Z(x)) \right] \\
 &= \min_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}} [\mathbb{D}_{\text{KL}}(\pi_\theta(y|x) \| \pi^*(y|x)) - \log(Z(x))]
 \end{aligned}$$

Ok, not quite. What I was looking for was this. Can you show how you get that?

I finally give the model the solution to see if it can get its way to that answer

Ah, thank you for the clarification — you're absolutely right that the goal is to write the KL-regularized reward maximization problem as a **KL minimization between the policy  $\pi_\theta(y|x)$  and the optimal policy  $\pi^*(y|x)$** , up to a constant involving the partition function  $\log Z(x)$ .

Let's derive the following:

$$\min_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}} [\mathbb{D}_{\text{KL}}(\pi_\theta(y|x) \| \pi^*(y|x)) - \log Z(x)]$$

## Step-by-Step Derivation

We begin with the original KL-regularized reward maximization problem:

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

Now **flip the sign** to turn this into a minimization:

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

Let's now factor this inside the expectation over  $x$ :

---

### Step 1: Use the definition of KL divergence

Recall:

$$\mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) = \mathbb{E}_{y \sim \pi_\theta(y|x)} \left[ \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} \right]$$

So the objective becomes:

$$\min_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}} \left[ \mathbb{E}_{y \sim \pi_\theta(y|x)} \left[ \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} - r_\phi(x, y) \right] \right]$$

Rewriting this:

$$\min_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}} \left[ \mathbb{E}_{y \sim \pi_\theta(y|x)} \left[ \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x) \exp \left( \frac{1}{\beta} r_\phi(x, y) \right)} \right] \right]$$

Note that the denominator inside the log is proportional to  $\pi^*(y|x)$ :

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

So we can write:

It kind of complicates it below by expanding it like that, but overall, it has the right idea

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

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


---

## ✓ Step 2: Final rearrangement

So we now have:

$$\min_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}} \left[ \mathbb{E}_{y \sim \pi_\theta(y|x)} \left[ \log \frac{\pi_\theta(y|x)}{\pi^*(y|x)} \right] + \log Z(x) \right]$$

That is:

$$\min_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}} [\mathbb{D}_{\text{KL}}(\pi_\theta(y|x) \| \pi^*(y|x)) - \log Z(x)]$$

---

### 🧠 Interpretation

- The original reward-maximizing RL objective becomes a **KL minimization** to the optimal reweighted policy  $\pi^*(y|x)$ , up to a constant shift  $\log Z(x)$
- This formulation will be the foundation for DPO, where we don't need to estimate reward values explicitly — only preference-induced ranking between outputs

Let me know if you'd like to move on to part (c), where DPO is likely derived from this!



- (c) Even though we've obtained the optimal policy  $\pi_{\theta^*}(y | x)$ , this particular representation of it is hard to utilize in practice. Why? Hint: How large can the output space  $\mathcal{Y}$  be?

Ok great. Glad you got it now. Now, try solving part c

---

Thanks! Let's now solve **Part (c)**.

### ❓ Problem Restatement

Even though we've obtained the optimal policy  $\pi_{\theta}^*(y | x)$ , this particular representation is **hard to utilize in practice**.

**Why?**

*Hint: How large can the output space  $\mathcal{Y}$  be?*

---

### ✓ Key Insight

The optimal policy we derived was:

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

Where the partition function is:

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

To compute or even sample from this policy, we would need to:

1. Evaluate the reward function  $r_\phi(x, y')$  for **every** possible output  $y' \in \mathcal{Y}$
  2. Sum over all  $y' \in \mathcal{Y}$  to compute  $Z(x)$
  3. Normalize
- 

## The Problem: Size of Output Space

In the context of language modeling:

- The output space  $\mathcal{Y}$  is **exponentially large**
- For an output of length  $T$ , and vocabulary size  $V$ , there are:

$$|\mathcal{Y}| = V^T$$

This makes exact computation of the sum over  $\mathcal{Y}$  in  $Z(x)$ , or even sampling from  $\pi^*(y|x)$ , **intractable**.

Even for short responses (say, 50 tokens) and a vocab size of 50k, the number of possible completions is astronomical.

---

## Final Answer:

Because the output space  $\mathcal{Y}$  is exponentially large (e.g., all possible

Hence, the form of  $\pi^*(y|x)$  is theoretically optimal, but **impractical to compute** or use directly in large language models.

---

Let me know when you're ready to move on to part (d), where we'll likely start approximating or reformulating this!



**Explains the answer differently but conceptually it understands!**

- (d) Using the expression for the optimal policy  $\pi_{\theta^*}$  derived in part (b), express the reward function  $r_\phi(x, y)$  in terms of  $\pi_{\theta^*}$ ,  $\pi_{\text{ref}}$ , and  $Z(x)$ .

Yeah! You're right. The vocabulary would be extremely large! Now, please solve part D

---

Great! Let's now solve **Part (d)**.

---

## ? Problem

Using the expression for the optimal policy  $\pi_\theta^*(y \mid x)$  derived in part (b), express the reward function  $r_\phi(x, y)$  in terms of  $\pi_\theta^*$ ,  $\pi_{\text{ref}}$ , and  $Z(x)$ .

---

## ✓ Step-by-step Derivation

Recall from part (b) that:

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

We want to **solve for  $r_\phi(x, y)$** .

### Step 1: Isolate the exponential

Multiply both sides by  $Z(x)$ :

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

Divide both sides by  $\pi_{\text{ref}}(y | x)$ :

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

### Step 2: Take the logarithm

Take the natural log of both sides:

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

### Final Answer:

Multiply both sides by  $\beta$ :

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

This is the reward that makes the optimal policy  $\pi_\theta^*$  consistent with the exponentiated reward form.

 **Intuition**

- This tells us that the **reward function** under the DPO framework is proportional to the **log odds ratio** between the optimal and reference policies, adjusted by the **log partition function**.
- If you know  $\pi_\theta^*$  and  $\pi_{\text{ref}}$ , then you can directly compute the reward value (up to temperature  $\beta$ ).

Let me know when you're ready to proceed to part (e)!



- 
- (e) With this reparametrization of  $r_\phi(x, y)$ , substitute it into the Bradley-Terry model to obtain the probability of human preference data in terms of the optimal policy. Furthermore, show that the partition function  $Z(x)$  cancels out in your computation. Why is this desirable?

That's right! Now, please solve part e.

Great — let's work through part (e) step by step using the reparameterization of  $r_\phi(x, y)$  from part (d), and apply it to the **Bradley-Terry model**.

---

 **Recap of Part (d): Reward Reparameterization**

From part (d), we derived:

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

This rewrites the learned reward function in terms of the optimal policy  $\pi_{\theta^*}$ , the reference policy  $\pi_{\text{ref}}$ , and the partition function  $Z(x)$ .

---

### Step 1: Plug into the Bradley-Terry model

The Bradley-Terry model gives the probability that a human prefers  $y_1$  over  $y_2$  given context  $x$  as:

$$P(y_1 > y_2 \mid x) = \frac{\exp(r_\phi(x, y_1))}{\exp(r_\phi(x, y_1)) + \exp(r_\phi(x, y_2))}$$

Substituting the reparameterized form of  $r_\phi$ :

$$\frac{P(y_1 > y_2 \mid x) = \exp\left(\log \frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} + \log Z(x)\right)}{\exp\left(\log \frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} + \log Z(x)\right) + \exp\left(\log \frac{\pi_{\theta^*}(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)} + \log Z(x)\right)}$$


---

### Step 2: Simplify using log/exp rules

Since  $\exp(\log a + \log Z(x)) = a \cdot Z(x)$ , we get:

$$P(y_1 > y_2 \mid x) = \frac{\frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} \cdot Z(x)}{\frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} \cdot Z(x) + \frac{\pi_{\theta^*}(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)} \cdot Z(x)}$$

The  $Z(x)$  terms cancel out:

$$P(y_1 > y_2 \mid x) = \frac{\frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)}}{\frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} + \frac{\pi_{\theta^*}(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)}}$$


---

 **Final Answer (Expression in Terms of Optimal Policy):**

$$P(y_1 > y_2 \mid x) = \frac{\frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)}}{\frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} + \frac{\pi_{\theta^*}(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)}}$$


---

 **Why is it desirable that  $Z(x)$  cancels?**

Because:

- $Z(x) = \sum_y \pi_{\text{ref}}(y \mid x) \exp(r_\phi(x, y))$  is a **partition function**, and:
  - It's **intractable** to compute when the action space  $y$  is large (e.g., entire language vocabularies).
  - By canceling out, we **avoid computing  $Z(x)$** , making preference learning **practical and scalable**.
  - The preference model becomes directly computable from **policy ratios**, which are tractable and learnable.
- 

Let me know if you'd like me to help derive the gradient of the loss function from this, or explain how this fits into RLHF pipelines.



ok so on this problem, it really struggles.

it seems to get the equations and notation confused very easily,  
i suspect this kind of data is not very common in its training distribution

$$p_{\theta^*}(y_1 \succ y_2 | x) = \frac{\exp(r_\phi(x, y_1))}{\exp(r_\phi(x, y_1)) + \exp(r_\phi(x, y_2))}$$

Try showing your work starting with this equation, then try to solve it.

Absolutely — let's start from the **Bradley-Terry model** expression you just shared:

---

### 📌 Step 1: Start from the preference model

$$p_{\theta^*}(y_1 \succ y_2 | x) = \frac{\exp(r_\phi(x, y_1))}{\exp(r_\phi(x, y_1)) + \exp(r_\phi(x, y_2))}$$

Now recall from **part (d)** that we can **reparameterize** the reward function as:

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


---

### 📌 Step 2: Plug in the reparameterization

We substitute  $r_\phi(x, y_1)$  and  $r_\phi(x, y_2)$ :

$$\frac{p_{\theta^*}(y_1 > y_2 \mid x) =}{\exp \left( \log \frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} + \log Z(x) \right)} \\ \frac{\exp \left( \log \frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} + \log Z(x) \right) + \exp \left( \log \frac{\pi_{\theta^*}(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)} + \log Z(x) \right)}{\exp \left( \log \frac{\pi_{\theta^*}(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)} + \log Z(x) \right) + \exp \left( \log \frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} + \log Z(x) \right)}$$


---

### 📌 Step 3: Simplify the exponentials

Using the identity  $\exp(a + b) = \exp(a) \cdot \exp(b)$ , we get:

$$= \frac{\frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} \cdot Z(x)}{\left( \frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} + \frac{\pi_{\theta^*}(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)} \right) \cdot Z(x)}$$


---

### 📌 Step 4: Cancel out $Z(x)$

It keeps trying to do this double policy fraction again!

Since it appears in both numerator and denominator:

$$= \frac{\frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)}}{\frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} + \frac{\pi_{\theta^*}(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)}}$$


---

### ✅ Final Answer (simplified expression):

$$p_{\theta^*}(y_1 > y_2 \mid x) = \frac{\frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)}}{\frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} + \frac{\pi_{\theta^*}(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)}}$$


---

This shows that **pairwise preferences under the learned reward function are entirely determined by the ratio between the optimal and reference policies**, and the normalization constant  $Z(x)$  cancels out — making the whole expression computationally tractable.

Would you like to now write this in expectation form (e.g. minimizing KL divergence or cross-entropy loss)?



$$\begin{aligned} p_{\theta^*}(y_1 \succ y_2 | x) \\ = \frac{\exp \left( \beta \log \left( \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} \right) + \beta \log(Z(x)) \right)}{\exp \left( \beta \log \left( \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} \right) + \beta \log(Z(x)) \right) + \exp \left( \beta \log \left( \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} \right) + \beta \log(Z(x)) \right)} \end{aligned}$$

Can you try again where the next step after the equation I gave above is the following equation?

Yes! You're now expressing the Bradley-Terry preference model using the reparameterized reward function:

$$\begin{aligned} p_{\theta^*}(y_1 \succ y_2 | x) = \\ \frac{\exp \left( \beta \log \left( \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} \right) + \beta \log Z(x) \right)}{\exp \left( \beta \log \left( \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} \right) + \beta \log Z(x) \right) + \exp \left( \beta \log \left( \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} \right) + \beta \log Z(x) \right)} \end{aligned}$$

**Step-by-step simplification:**

We simplify the numerator and denominator using exponent rules:

**📌 Step 1: Pull out  $\beta \log Z(x)$** 

Each term in the numerator and denominator has a shared additive term  $\beta \log Z(x)$ , so we factor it out:

$$\frac{\exp\left(\beta \log\left(\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)\right) \cdot \exp(\beta \log Z(x))}{\left[\exp\left(\beta \log\left(\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)\right) + \exp\left(\beta \log\left(\frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)}\right)\right)\right] \cdot \exp(\beta \log Z(x))}$$

**📌 Step 2: Cancel out the shared  $\exp(\beta \log Z(x))$** 

$$= \frac{\left(\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)^\beta}{\left(\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)^\beta + \left(\frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)}\right)^\beta}$$

**✓ Final Answer:**

$$p_{\theta^*}(y_1 > y_2 | x) = \frac{\left(\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)^\beta}{\left(\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)^\beta + \left(\frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)}\right)^\beta}$$

**💡 Why is canceling out  $Z(x)$  desirable?**

- **Numerical stability:**  $Z(x)$  is typically a partition function over a large (possibly infinite) action space, which is hard or intractable to compute.
- **Efficiency:** You can compute preferences without ever needing to evaluate  $Z(x)$ .
- **Interpretability:** Preference learning reduces to a comparison of relative likelihoods under  $\pi^*$  and  $\pi_{\text{ref}}$ , raised to a temperature-like parameter  $\beta$ .

Let me know if you'd like this boxed as a LaTeX expression!



Can you try seeing how it cancels out to turn into a sigmoid?

Absolutely! Let's now show how the expression simplifies to a **sigmoid** form.

We'll start from the equation you posted:

$$p_{\theta^*}(y_1 > y_2 \mid x) = \frac{\exp(r_\phi(x, y_1))}{\exp(r_\phi(x, y_1)) + \exp(r_\phi(x, y_2))}$$


---

### Step 1: Plug in the expression for $r_\phi(x, y)$

From earlier parts (like part d), we derived:

$$r_\phi(x, y) = \beta \log \left( \frac{\pi^*(y \mid x)}{\pi_{\text{ref}}(y \mid x)} \right) + \beta \log Z(x)$$

Plug that into the Bradley-Terry model:

$$\frac{p_{\theta^*}(y_1 > y_2 \mid x) =}{\exp \left( \beta \log \left( \frac{\pi^*(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} \right) + \beta \log Z(x) \right)} \\ \frac{\exp \left( \beta \log \left( \frac{\pi^*(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} \right) + \beta \log Z(x) \right) + \exp \left( \beta \log \left( \frac{\pi^*(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)} \right) + \beta \log Z(x) \right)}{\exp \left( \beta \log \left( \frac{\pi^*(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)} \right) + \beta \log Z(x) \right) + \exp \left( \beta \log \left( \frac{\pi^*(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} \right) + \beta \log Z(x) \right)}$$


---

**Step 2: Factor the exponentials**

Using the identity:

$$\exp(a + b) = \exp(a) \cdot \exp(b)$$

This becomes:

$$= \frac{\left( \frac{\pi^*(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} \right)^\beta \cdot Z(x)^\beta}{\left( \frac{\pi^*(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} \right)^\beta \cdot Z(x)^\beta + \left( \frac{\pi^*(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)} \right)^\beta \cdot Z(x)^\beta}$$

Now cancel the common  $Z(x)^\beta$  term:

$$= \frac{\left( \frac{\pi^*(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} \right)^\beta}{\left( \frac{\pi^*(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} \right)^\beta + \left( \frac{\pi^*(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)} \right)^\beta}$$

Again, it tries to write it using this notation. And it goes down the wrong path trying to solve the problem

**Step 3: Write in terms of difference of rewards (sigmoid form)**

Recall that:

$$\frac{1}{1 + \exp(-z)} = \text{sigmoid}(z)$$

We want to rewrite this probability in terms of a **difference between scores**.

So define:

$$s(y) := r_\phi(x, y) = \beta \log \left( \frac{\pi^*(y \mid x)}{\pi_{\text{ref}}(y \mid x)} \right) + \beta \log Z(x)$$

Then:

$$p_{\theta^*}(y_1 > y_2 \mid x) = \frac{e^{s(y_1)}}{e^{s(y_1)} + e^{s(y_2)}} = \frac{1}{1 + \exp(s(y_2) - s(y_1))} =$$

$$\sigma(s(y_1) - s(y_2))$$

So we've shown:

$$p_{\theta^*}(y_1 > y_2 \mid x) = \sigma(r_\phi(x, y_1) - r_\phi(x, y_2))$$

This is a **sigmoid over reward differences**, and:

### Why is this desirable?

- 1. Partition function  $Z(x)$  cancels out:** You can compute relative preferences without needing to normalize over all  $y$  — which could be very large (like all possible strings).
- 2. Efficient training:** We can train reward models using preference pairs without evaluating the full distribution.
- 3. Nice probabilistic interpretation:** This form corresponds to a **logistic model** over the difference in reward scores — simple and interpretable.

Let me know if you'd like this typeset as a final boxed LaTeX answer.



Now that we have the probability of human preference data in terms of the optimal policy rather than the reward model, we can formulate the Maximum Likelihood objective for the parameterized policy  $\pi_\theta$

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

which is exactly the result we wanted!

Recall the complex RLHF pipeline we started with: **SFT** → **Reward Modeling** → **RL Fine-Tuning**. By deriving the equation above, we have bypassed the need for an explicit reward model  $r_\phi(x, y)$  and the RL loop entirely. Instead, we are fitting an *implicit* reward model (defined by the log-ratio of our policy and the reference) using a simple binary cross-entropy loss.

Crucially, because we derived this objective by algebraically solving for the reward in part (c), the policy  $\pi_\theta$  that minimizes this loss is guaranteed to be the optimal policy for that implicit reward function. We have thus reduced the RLHF pipeline to a standard supervised classification problem!

- (f) To verify how this optimization works in practice, let the implicit reward be defined as  $\hat{r}_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ . Derive the gradient of the DPO loss with respect to the parameters  $\theta$ , i.e.,  $\nabla_\theta \mathcal{L}_{\text{DPO}}$ .

Afterwards, interpret the role of the weighting term  $\sigma(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w))$  in the  $\nabla_\theta \mathcal{L}_{\text{DPO}}$ . Does the gradient update the model more when it is correct, or when it is incorrect?

*Hint: Recall that  $\sigma'(z) = \sigma(z)(1 - \sigma(z)) = \sigma(z)\sigma(-z)$ .*

Ok yeah that's the right way to write the final answer! Nice.

Now, please try solving the following

---

Great! Let's now solve part (f), which asks us to:

### ■ Derive the gradient of the DPO loss:

We are given the **DPO loss**:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} [\log \sigma (\hat{r}_\theta(x, y_w) - \hat{r}_\theta(x, y_l))]$$

Where the implicit reward is:

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

Let's define:

$$z := \hat{r}_\theta(x, y_w) - \hat{r}_\theta(x, y_l) \quad \Rightarrow \quad \mathcal{L}_{\text{DPO}} = -\log \sigma(z)$$


---

 **Step 1: Compute the derivative of the loss wrt  $z$**

We know:

$$\frac{d}{dz} [-\log \sigma(z)] = -\frac{1}{\sigma(z)} \cdot \sigma'(z) = -\frac{1}{\sigma(z)} \cdot \sigma(z)(1 - \sigma(z)) = \frac{\sigma(z) - 1}{\sigma(z)}$$

So the derivative of the loss is:

$$\frac{d\mathcal{L}_{\text{DPO}}}{dz} = \sigma(z) - 1$$


---

 **Step 2: Apply chain rule to get gradient wrt model parameters  $\theta$**

We want:

$$\nabla_\theta \mathcal{L}_{\text{DPO}} = \frac{d\mathcal{L}}{dz} \cdot \nabla_\theta z = (\sigma(z) - 1) \cdot \nabla_\theta [\hat{r}_\theta(x, y_w) - \hat{r}_\theta(x, y_l)]$$

Now compute the gradient of the **reward difference**:

$$\begin{aligned} \nabla_\theta z &= \nabla_\theta \left[ \beta \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right] = \\ &\quad \beta [\nabla_\theta \log \pi_\theta(y_w | x) - \nabla_\theta \log \pi_\theta(y_l | x)] \end{aligned}$$

So the full gradient becomes:

$$\nabla_{\theta} \mathcal{L}_{\text{DPO}} = (\sigma(z) - 1) \cdot \beta [\nabla_{\theta} \log \pi_{\theta}(y_w | x) - \nabla_{\theta} \log \pi_{\theta}(y_l | x)]$$


---

 **Final Answer (boxed):**

$$\nabla_{\theta} \mathcal{L}_{\text{DPO}} = \beta \cdot (\sigma(\hat{r}_{\theta}(x, y_w) - \hat{r}_{\theta}(x, y_l)) - 1) \cdot [\nabla_{\theta} \log \pi_{\theta}(y_w | x) - \nabla_{\theta} \log \pi_{\theta}(y_l | x)]$$

---

## Interpretation

This gradient shows how much we **upweight or downweight** the log-probability gradients depending on correctness:

- If the **preferred sample**  $y_w$  is assigned a **higher reward than**  $y_l \rightarrow$   
Then  $\hat{r}_{\theta}(x, y_w) > \hat{r}_{\theta}(x, y_l) \Rightarrow \sigma(z) \approx 1 \rightarrow$   
So  $\sigma(z) - 1 \approx 0$ , i.e., **small update** — model is already correct.
- If the model is **incorrect**, i.e.,  $\hat{r}_{\theta}(x, y_w) < \hat{r}_{\theta}(x, y_l) \Rightarrow \sigma(z) \ll 1$ , then  
 $\sigma(z) - 1 \ll 0$ , and the gradient will be large — we push the model to assign **higher log-prob** to the winner and **lower log-prob** to the loser.

In other words:

The weighting term  $\sigma(z) - 1$  ensures the model updates **more when it is wrong** and **less when it is already correct**.

Let me know if you'd like a diagram to visualize this or have it formatted as a LaTeX theorem box!



Not quite, but you were actually super close.

Here, I try to nudge it to recheck its answer for mistakes, but I end up leading it down the wrong path

Great — let's carefully walk through it again and this time **land exactly on the correct result**, which involves both the **sigmoid derivative** and the **difference in implicit rewards**. Here's the clean derivation from scratch, aligned with the setup from your image.

---

## Setup

You're given the **DPO loss**:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l)} [\log \sigma(\hat{r}_\theta(x, y_w) - \hat{r}_\theta(x, y_l))]$$

where the **implicit reward** is:

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


---

## Step 1: Define $z$

Let:

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


---

## Step 2: Compute gradient of loss

We apply chain rule:

$$\nabla_{\theta} \mathcal{L}_{\text{DPO}} = -\nabla_{\theta} \log \sigma(z) = -\sigma(-z) \cdot \nabla_{\theta} z$$

Now compute:

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

We now plug in:

$$\hat{r}_{\theta}(x, y) = \beta \log \frac{\pi_{\theta}(y | x)}{\pi_{\text{ref}}(y | x)} \Rightarrow \nabla_{\theta} \hat{r}_{\theta}(x, y) = \beta \cdot \nabla_{\theta} \log \pi_{\theta}(y | x)$$

(since  $\pi_{\text{ref}}$  is fixed and doesn't depend on  $\theta$ )

So the full gradient becomes:

$$\nabla_{\theta} \mathcal{L}_{\text{DPO}} = -\sigma(-z) \cdot \beta \cdot [\nabla_{\theta} \log \pi_{\theta}(y_w | x) - \nabla_{\theta} \log \pi_{\theta}(y_l | x)]$$


---

### Final Answer

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

Or, equivalently:

$$\nabla_{\theta} \mathcal{L}_{\text{DPO}} = \beta \cdot \sigma(-z) \cdot [\nabla_{\theta} \log \pi_{\theta}(y_l | x) - \nabla_{\theta} \log \pi_{\theta}(y_w | x)]$$

---

### Intuition: What is this saying?

- When the reward difference  $\hat{r}(y_w) - \hat{r}(y_l)$  is **large and positive**, we have  $\sigma(-z) \approx 0$ , and the gradient is **close to 0** → no update needed — model is correct.
- When the model is **wrong**, i.e.,  $\hat{r}(y_w) < \hat{r}(y_l)$ , then  $\sigma(-z) \approx 1$  → we make a **large update** in the direction of increasing  $\log \pi_\theta(y_w)$  and decreasing  $\log \pi_\theta(y_l)$ .

So the **sigmoid term acts as an error-weighting function**, guiding updates more aggressively when the preference is violated.

---

Let me know if you'd like this rendered as LaTeX or if you want to derive the second-order (Hessian) version too!



I try to reground it on telling it to go back to its previous answer

We can rewrite the RHS of the above equation as

$$\nabla_{\theta} \mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[ \frac{\sigma'(u)}{\sigma(u)} \nabla_{\theta} u \right]$$

where  $u = \beta \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} - \beta \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)}$ .

Not quite. Try starting at this equation and doing what you need above (your initial response to the question)

Perfect — now that you've given the rewritten gradient expression:

$$\nabla_{\theta} \mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[ \frac{\sigma'(u)}{\sigma(u)} \cdot \nabla_{\theta} u \right]$$

with:

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

Let's simplify this and derive the correct gradient form, step-by-step.

I like this! It tries to simplify the notation it uses to make the work more clear

### Step 1: Use the identity for the sigmoid derivative

We are given the useful identity:

$$\sigma'(z) = \sigma(z)(1 - \sigma(z)) = \sigma(z)\sigma(-z)$$

So:

$$\frac{\sigma'(u)}{\sigma(u)} = \sigma(-u)$$

Therefore, the gradient becomes:

$$\nabla_\theta \mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(x, y_w, y_l)} [\sigma(-u) \cdot \nabla_\theta u]$$

### Step 2: Compute $\nabla_\theta u$

We now compute the gradient of:

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

Since the reference policy is fixed and doesn't depend on  $\theta$ , we treat those terms as constants. So:

$$\nabla_{\theta} u = \beta \cdot \nabla_{\theta} \log \pi_{\theta}(y_l \mid x) - \beta \cdot \nabla_{\theta} \log \pi_{\theta}(y_w \mid x)$$

### ✓ Step 3: Plug back into the full expression

$$\begin{aligned}\nabla_{\theta} \mathcal{L}_{\text{DPO}} &= \\ -\mathbb{E}_{(x, y_w, y_l)} [\sigma(-u) \cdot (\beta \cdot \nabla_{\theta} \log \pi_{\theta}(y_l \mid x) - \beta \cdot \nabla_{\theta} \log \pi_{\theta}(y_w \mid x))] \\ &= \beta \cdot \mathbb{E}_{(x, y_w, y_l)} [\sigma(-u) \cdot (\nabla_{\theta} \log \pi_{\theta}(y_w \mid x) - \nabla_{\theta} \log \pi_{\theta}(y_l \mid x))]\end{aligned}$$

### ✓ Final Answer (boxed)

$$\nabla_{\theta} \mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = \beta \cdot \mathbb{E}_{(x, y_w, y_l)} [\sigma(-u) \cdot (\nabla_{\theta} \log \pi_{\theta}(y_w \mid x) - \nabla_{\theta} \log \pi_{\theta}(y_l \mid x))]$$

This is **equivalent** to the earlier expression we derived before but fully written in terms of  $u$ . It perfectly aligns with your image and the DPO loss gradient interpretation.

Let me know if you'd like this rendered as full LaTeX or want to derive the Hessian next!

It still can't get the answer correct despite simplifying its notation since I think it gets overwhelmed that so many equations exist.

I also wonder if this lack in performance relates to the context length being so large (because images are known to take up a lot of space in the model's context). So I wonder how much of this can be attributed to context degradation.

$\frac{\partial(x, y_w)}{\partial(x, y_w)}$  estimate is wrong

increases  $y_1, \dots, y_K, x$  =  $\prod_{k=1}^K \frac{\exp(r_{\phi}(x, y_{\tau(k)})}{\sum_{j=k}^K \exp(r_{\phi}(x, y_{\tau(j)}))} \sim_{\mathcal{D}}$

$$\left[ \log \sigma \left( \beta \log \frac{\pi_{\theta}(y_{\tau(k)} \mid x)}{\pi_{\text{ref}}(y_{\tau(k)} \mid x)} \right) \right]$$

Ok, not quite. The sign should be negative. I posted the correct answer. Try solving part g now (this question has 2 images)

Let's now walk through **part (g)** step-by-step, using the implicit reward parameterization:

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

and show that the partition function  $Z(x)$  cancels out in the Plackett-Luce likelihood expression.

---

### Step 1: Write out Plackett-Luce probability with the reward parameterization

From the problem:

$$p_{\theta^*}(\tau | y_1, \dots, y_K, x) = \prod_{k=1}^K \frac{\exp(r_\phi(x, y_{\tau(k)}))}{\sum_{j=k}^K \exp(r_\phi(x, y_{\tau(j)}))}$$

Substitute the reward function:

$$\begin{aligned} \exp(r_\phi(x, y)) &= \exp\left(\beta \log \frac{\pi_{\theta^*}(y | x)}{\pi_{\text{ref}}(y | x)} + \beta \log Z(x)\right) = \\ &\quad \left(\frac{\pi_{\theta^*}(y | x)}{\pi_{\text{ref}}(y | x)} \cdot Z(x)\right)^\beta \end{aligned}$$

Now plug this into the PL probability:

$$p_{\theta^*}(\tau \mid y_1, \dots, y_K, x) = \prod_{k=1}^K \frac{\left( \frac{\pi_{\theta^*}(y_{\tau(k)} \mid x)}{\pi_{\text{ref}}(y_{\tau(k)} \mid x)} \cdot Z(x) \right)^{\beta}}{\sum_{j=k}^K \left( \frac{\pi_{\theta^*}(y_{\tau(j)} \mid x)}{\pi_{\text{ref}}(y_{\tau(j)} \mid x)} \cdot Z(x) \right)^{\beta}}$$


---

### Step 2: Cancel the partition function $Z(x)$

Note that for each term in the product,  $Z(x)^{\beta}$  appears both in the numerator and all terms in the denominator:

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

So the final expression for the probability becomes:

$$p_{\theta^*}(\tau \mid y_1, \dots, y_K, x) = \prod_{k=1}^K \frac{\left( \frac{\pi_{\theta^*}(y_{\tau(k)} \mid x)}{\pi_{\text{ref}}(y_{\tau(k)} \mid x)} \right)^{\beta}}{\sum_{j=k}^K \left( \frac{\pi_{\theta^*}(y_{\tau(j)} \mid x)}{\pi_{\text{ref}}(y_{\tau(j)} \mid x)} \right)^{\beta}}$$


---

### Step 3: Take log and form DPO loss

We now define the loss as the **negative log-likelihood** of the observed permutation under this probability:

$$\mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\mathbb{E}_{(\tau, y_1, \dots, y_K, x) \sim \mathcal{D}} [\log p_{\theta^*}(\tau \mid y_1, \dots, y_K, x)]$$

Plug in the expression from above:

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

Simplify the log of the quotient:

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

This is the **multi-answer generalization of DPO** using the Plackett-Luce model. The final DPO loss is:

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


---

### ✓ Final Answer (Boxed)

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

Let me know if you'd like this reformatted into LaTeX or converted into code!



This is way too complicated and not really close to the answer

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

Ok this problem is pretty difficult. Try starting with the following simplification:

so I try giving it a hint!

Great! Let's now walk through the **simplification shown in your latest image** to help you derive the expression used in **DPO (Direct Preference Optimization)**. You've already factored the reward into two clean components:

### Step 1: Start from the implicit reward form

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

### Step 2: Use exponent rules

Recall that  $\exp(a + b) = \exp(a) \cdot \exp(b)$ . Applying that:

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

### Step 3: Simplify each exponential

The first term becomes:

$$\left( \frac{\pi_{\theta^*}(y | x)}{\pi_{\text{ref}}(y | x)} \right)^\beta$$

The second term:

$$\exp(\beta \log Z(x)) = Z(x)^\beta$$


---

#### Final result:

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


---

This simplification is **crucial** in DPO derivations (especially the binary or pairwise case), because when we form ratios like:

$$\frac{\exp(r_\phi(x, y_1))}{\exp(r_\phi(x, y_1)) + \exp(r_\phi(x, y_2))}$$

the  $Z(x)^\beta$  terms **cancel out**, allowing the final expression to collapse into a **sigmoid function over log-ratio scores**:

$$\sigma(u) = \frac{1}{1 + \exp(-u)} \quad \text{where } u = \beta \left( \log \frac{\pi_\theta(y_1 | x)}{\pi_{\text{ref}}(y_1 | x)} - \log \frac{\pi_\theta(y_2 | x)}{\pi_{\text{ref}}(y_2 | x)} \right)$$

It doesn't even think of applying the summation by trying to determine the k-th PL product. It just tries to hail-mary the final solution

Let me know if you'd like to continue and finish that simplification too!



Your final probability should be a product of terms

You're absolutely right to point out that the **final form of  $\exp(r_\phi(x, y))$**  should reflect a **product** of terms—specifically a product involving the ratio of policy and reference probabilities, and the normalization term  $Z(x)$ . Let's now carefully walk through this step-by-step and clarify where the **product form** emerges in the context of **DPO** and implicit reward modeling.

---

### Recap: Start with the expression

We are given the following decomposition:

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

Apply the identity:

$$\exp(a + b) = \exp(a) \cdot \exp(b)$$

---

### Step 1: Distribute the exponential

$$\exp(r_\phi(x, y)) = \underbrace{\exp\left(\beta \log \frac{\pi_{\theta^*}(y | x)}{\pi_{\text{ref}}(y | x)}\right)}_{\text{Policy/reference ratio term}} \cdot \underbrace{\exp(\beta \log Z(x))}_{\text{Normalization term}}$$


---

## Step 2: Simplify each exponential

- First term:

$$\exp\left(\beta \log \frac{\pi_{\theta^*}(y | x)}{\pi_{\text{ref}}(y | x)}\right) = \left(\frac{\pi_{\theta^*}(y | x)}{\pi_{\text{ref}}(y | x)}\right)^\beta$$

- Second term:

$$\exp(\beta \log Z(x)) = Z(x)^\beta$$

It still doesn't get the answer right... so i gave up :/

## Final Result (Product Form):

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

---

This **product form** is critical because:

- The first factor captures how much more likely  $y$  is under the optimal policy  $\pi_{\theta^*}$  than under the reference policy  $\pi_{\text{ref}}$ .
- The second factor  $Z(x)^\beta$  acts like a **normalization constant** that ensures we recover a proper softmax distribution when summing over all  $y$ .

## Usage in Preference Probabilities

When calculating preference probabilities like:

$$p(y_1 > y_2 \mid x) = \frac{\exp(r_\phi(x, y_1))}{\exp(r_\phi(x, y_1)) + \exp(r_\phi(x, y_2))}$$

the  $Z(x)^\beta$  term cancels out:

$$= \frac{\left(\frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)}\right)^\beta}{\left(\frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)}\right)^\beta + \left(\frac{\pi_{\theta^*}(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)}\right)^\beta}$$

Then taking log-odds:

$$\log \frac{p(y_1 > y_2 \mid x)}{1 - p(y_1 > y_2 \mid x)} = \beta \left[ \log \frac{\pi_\theta(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} - \log \frac{\pi_\theta(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)} \right]$$

This motivates the form of the DPO objective:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E} \left[ \log \sigma \left( \beta \left[ \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \right] \right) \right]$$

Let me know if you want to go **from this to the gradient** (with the  $\sigma'(u)/\sigma(u)$  term), or explain where the sigmoid appears directly!

