

Claude HW13 output

- 👤 You will be answering a series of homework questions from a deep learning class. I will input multiple subquestions as a screenshots, and questions with the same number will build upon previous subquestions. Some questions may take multiple screenshots so make sure to answer all subquestions. You will need to use context from the first image. Avoid hallucinating or making up definitions if information is missing; explicitly say so. Answer with clear math notation and concise justification. Do not skip derivation steps. If the problem contains multiple subparts (a), (b), (c), or (i),(ii), answer each clearly.

12/9/2025, 4:42:31 PM

I understand! I'm ready to help you with your deep learning homework questions.

I'll:

1. Answer each subquestion clearly and completely
2. Use proper mathematical notation
3. Provide concise justifications without skipping derivation steps
4. Build upon context from previous subquestions when needed
5. Explicitly state if information is missing rather than making assumptions
6. Address all parts (a), (b), (c) or (i), (ii), etc. separately

Please go ahead and share the screenshots of your homework questions, and I'll work through them systematically.

12/9/2025, 4:42:35 PM



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 σ^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 Δt 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 $\Delta t \ll \sigma^2 \ll 1$.

- (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.)

- (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 σ^2 and you can use that to check your work.)

- (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?

- (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 σ^2 vs t .**

(HINT: Use the fact that $\Delta t \ll t$ to approximate $\eta(t, \Delta t)$ as an appropriate constant.)

- (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.)

here is 1a-h

12/9/2025, 4:43:17 PM

I'll work through each part systematically.

Problem 1: DDPM/DDIM Fun - From a Gaussian

Setup: Target distribution is $\mathcal{N}(0, \sigma^2)$ where $\sigma^2 \ll 1$. Forward process starts at $X_0 \sim \mathcal{N}(0, \sigma^2)$ and adds independent Gaussian noise with variance $\Delta t \ll \sigma^2$ at each step from $t = 0$ to $t = 1$.

(a) What is the distribution of X_1 ?

We can think of X_1 as the result of $T = \frac{1}{\Delta t}$ steps of adding independent Gaussian noise to X_0 .

Starting with $X_0 \sim \mathcal{N}(0, \sigma^2)$, after adding T independent Gaussian noises each with variance Δt :

$$X_1 = X_0 + \sum_{i=1}^T \epsilon_i$$

where $\epsilon_i \sim \mathcal{N}(0, \Delta t)$ are independent.

Since the sum of independent Gaussians is Gaussian:

- Mean: $0 + 0 = 0$
- Variance: $\sigma^2 + T \cdot \Delta t = \sigma^2 + \frac{1}{\Delta t} \cdot \Delta t = \sigma^2 + 1$

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

(b) Marginal distributions of $X_{t-\Delta t}$ and X_t

By the same reasoning as part (a), at any time t we've added $(t/\Delta t)$ noise steps:

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

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

(c) Simplify the variance of the conditional distribution

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

We need to simplify the variance $\frac{(\sigma^2 + t - \Delta t)\Delta t}{\sigma^2 + t}$ when $\Delta t \ll \sigma^2 \ll 1$.

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

Since $\Delta t \ll \sigma^2 \ll 1$ and $t \in [0, 1]$, we have $(\Delta t)^2 \ll \Delta t$ and $\sigma^2 + t \geq \sigma^2 > 0$.

Therefore: $\frac{(\Delta t)^2}{\sigma^2 + t} \ll \Delta t$

$$\boxed{\text{Variance} \approx \Delta t}$$

(d) Naive reverse denoising without stochastic noise

Start with $\hat{X}_1 \sim \mathcal{N}(0, 1)$ and iteratively apply conditional mean mapping backward with $T = \frac{1}{\Delta t}$ steps.

At each step: $\hat{X}_{t-\Delta t} = \frac{\sigma^2 + t - \Delta t}{\sigma^2 + t} \hat{X}_t$

After T steps from $t = 1$ to $t = 0$:

$$\begin{aligned}\hat{X}_0 &= \prod_{k=0}^{T-1} \frac{\sigma^2 + 1 - (k+1)\Delta t}{\sigma^2 + 1 - k\Delta t} \cdot \hat{X}_1 \\ &= \frac{\sigma^2}{\sigma^2 + 1} \hat{X}_1\end{aligned}$$

Since $\hat{X}_1 \sim \mathcal{N}(0, 1)$:

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

Telescoping: Notice the product telescopes - most terms cancel, leaving only $\frac{\sigma^2 + 0}{\sigma^2 + 1} = \frac{\sigma^2}{\sigma^2 + 1}$.

Since $\sigma^2 \ll 1$, the variance $\left(\frac{\sigma^2}{\sigma^2 + 1}\right)^2 \approx (\sigma^2)^2 = \sigma^4 \ll \sigma^2$, which is **far too small** compared to the desired σ^2 .

(e) Variance with added stochastic noise

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

Taking $\Delta t \rightarrow 0$ (so $T \rightarrow \infty$), convert the sum to an integral with $k\Delta t \rightarrow s$:

$$\text{Var}(\hat{X}_0) = \left(\frac{\sigma^2}{\sigma^2 + 1} \right)^2 + \int_0^1 \left(\frac{\sigma^2}{\sigma^2 + s} \right)^2 ds$$

Evaluate the integral: $\int_0^1 \frac{\sigma^4}{\sigma^2 + s} ds = \sigma^4 \int_0^1 \frac{1}{\sigma^2 + s} ds$

Using the hint $\int \frac{1}{(1+t)^2} dt = C - \frac{1}{1+t}$:

$$\begin{aligned} &= \sigma^4 \left[-\frac{1}{\sigma^2 + s} \right]_0^1 = \sigma^4 \left(-\frac{1}{\sigma^2 + 1} + \frac{1}{\sigma^2} \right) \\ &= \sigma^4 \cdot \frac{(\sigma^2 + 1) - \sigma^2}{\sigma^2(\sigma^2 + 1)} = \frac{\sigma^4}{\sigma^2(\sigma^2 + 1)} = \frac{\sigma^2}{\sigma^2 + 1} \end{aligned}$$

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

Since $\sigma^2 \ll 1$, the first term $\approx \sigma^4 \ll \sigma^2$:

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

This matches the desired variance σ^2 !

(f) Training a neural network for DDIM

Given dataset s_1, s_2, \dots, s_n drawn from $\mathcal{N}(0, \sigma^2)$, we want to train a neural net to estimate $g(x_t, t) = \frac{\sigma^2}{\sigma^2 + t} x_t$.

Inputs to the neural net: (x_t, t) where x_t is a noisy sample and t is the time/noise level.

Generating a batch:

1. Sample s_i from the dataset
2. Sample random time $t \sim \text{Uniform}(0, 1)$
3. Generate noisy version: $x_t = s_i + \epsilon$ where $\epsilon \sim \mathcal{N}(0, t)$
4. Batch consists of pairs (x_t, t) with targets $g(x_t, t) = \frac{\sigma^2}{\sigma^2 + t} x_t$

Loss function: Mean squared error between network output $f_\theta(x_t, t)$ and target $\frac{\sigma^2}{\sigma^2 + t} x_t$:

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

(g) Approximate the DDIM step for $\Delta t \ll t$

Given $g(x_t, t) = \frac{\sigma^2}{\sigma^2 + t} x_t$ and the DDIM update with $\eta(t, \Delta t) = \frac{\sqrt{t}}{\sqrt{t - \Delta t} + \sqrt{t}}$.

The DDIM step is: $x_{t-\Delta t}^{\text{DDIM}} = x_t + \eta(t, \Delta t) \cdot (\text{deterministic DDPM step})$

When $\Delta t \ll t$, approximate $\eta(t, \Delta t)$:

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

Using $\sqrt{1 - x} \approx 1 - x/2$ for small x :

$$\approx \frac{\sqrt{t}}{\sqrt{t}(1 - \Delta t/(2t)) + \sqrt{t}} = \frac{1}{2 - \Delta t/(2t)} \approx \frac{1}{2}$$

Therefore:

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

(h) Compute \hat{X}_0 from $\hat{X}_1 \sim \mathcal{N}(0, 1)$ via DDIM

Applying the DDIM step from part (g) iteratively:

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

After T steps, expressing as a product:

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

Taking logarithm and converting to sum:

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

Using $\ln(1 - x) \approx -x$ for small x :

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

Taking $\Delta t \rightarrow 0$, convert to integral with $s = 1 - k\Delta t$:

$$\rightarrow - \int_0^1 \frac{1}{2(\sigma^2 + s)} ds = -\frac{1}{2} [\ln(\sigma^2 + s)]_0^1 = -\frac{1}{2} \ln \frac{\sigma^2 + 1}{\sigma^2}$$

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

Since $\hat{X}_1 \sim \mathcal{N}(0, 1)$:

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

This matches the desired distribution!



2. Honey, Where's My Reward Model?

In the standard Reinforcement Learning from Human Feedback (RLHF) pipeline like in [Ziegler et al](#), there are usually three phases:

- 1 **SFT:** We perform Supervised Fine-Tuning (SFT) on a pre-trained LM using a dataset of high-quality demonstrations (e.g., summarization, sentiment analysis, etc.) to obtain a model π^{SFT} .
- 2 **Reward Modeling:** Next, we use the SFT model to generate pairs of answers $(y_1, y_2) \sim \pi^{\text{SFT}}(y | x)$ for various prompts x . Human labelers rank these pairs, producing a dataset of preferences $\mathcal{D} = \{(x^{(i)}, y_w^{(i)}, y_l^{(i)})\}_{i=1}^N$, where y_w denotes the preferred (winning) output and y_l the dispreferred (losing) output amongst (y_1, y_2) , respectively. Crucially, we assume these preferences are generated by some latent (hidden) reward model $r^*(x, y)$, which we do not have access to. Instead, we parameterize a proxy reward model $r_\phi(x, y)$ to approximate it. However, we face a challenge: How do we model these preferences, which are inherently discrete ($y_w \succ y_l$), using a reward model that outputs continuous scalars?

Homework 13, © Faculty teaching EECS 182, Fall 2025. All Rights Reserved. This may not be publicly shared without explicit permission.

2

Homework 13 @ 2025-12-03 22:16:54-08:00

To bridge this gap, we employ the Bradley-Terry (BT) model. The BT model assumes that the probability of a human preferring one response over another is given by the sigmoid of the difference in their latent rewards:

$$p^*(y_w \succ y_l | x) = \sigma(r^*(x, y_w) - r^*(x, y_l)) = \frac{\exp(r^*(x, y_w))}{\exp(r^*(x, y_w)) + \exp(r^*(x, y_l))}$$

This formulation allows us to frame preference learning as a binary classification problem because it treats the reward difference as a logit: if $r^*(x, y_w)$ is significantly higher than $r^*(x, y_l)$, the sigmoid outputs a probability near 1, matching the “label” that y_w is the winner. As a result, we can estimate the parameters ϕ via Maximum Likelihood Estimation (MLE) by minimizing the negative log-likelihood loss:

$$\mathcal{L}_R(\phi) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} [\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))]$$

In the context of LMs, $r^*(x, y)$ is often initialized from the SFT model $\pi^{\text{SFT}}(y | x)$ with the addition of a classification head to produce a single scalar prediction for the reward value. Furthermore, to ensure a reward function with lower variance, prior works normalize the rewards, such that $\mathbb{E}_{(x, y) \sim \mathcal{D}} [r_\phi(x, y)] = 0$ for all x .

$$\mathbb{E}_{(x,y) \sim \mathcal{D}} [r_\phi(x, y)] = 0 \text{ for all } x.$$

3 RL Fine-Tuning: Finally, we use learned reward function as feedback to the language model. The optimization problem is formulated as

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

where β is a parameter controlling the deviation from the base reference policy π_{ref} , namely the initial SFT model π^{SFT} . In practice, the language model policy π_θ is also initialized to π^{SFT} . Due to the discrete nature of language generation, the objective is not differentiable so it is typically optimized using reinforcement learning. The standard approach has been to construct the reward function $r(x, y) = r_\phi(x, y) - \beta(\log \pi_\theta(y|x) - \log \pi_{\text{ref}}(y|x))$, and maximize using PPO.

Unfortunately, this three-step process is brittle and computationally expensive. This where Direct Preference Optimization (DPO) comes in.

In this problem, you will derive **Direct Preference Optimization (DPO)**, a method introduced by [Rafailov et al.](#) that provides a simple yet effective approach for policy optimization using preferences directly.

- (a) As a warm up, find the optimal p^* for the optimization problem:

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

where

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

is the KL divergence of p from q .

- (b) As discussed earlier, we wish to find a language model policy π_θ 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:

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

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.

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

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 π_θ

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

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 π_θ 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 θ , 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 $\sigma'(z) = \sigma(z)(1 - \sigma(z)) = \sigma(z)\sigma(-z)$.

- (g) Throughout this problem, we have worked with the Bradley-Terry model to model human preferences. However, in many real-world scenarios, labelers rank multiple model outputs y_1, \dots, y_K rather than just comparing pairs. This is modeled by the Plackett-Luce model, which is a generalization of the Bradley-Terry model. In a similar fashion, the PL model stipulates that when presented with a set of possible choices, people prefer a choice with a probability proportional to the value of some latent reward function for that choice. In our context, when presented with a prompt x and a set of K answers $y_1, \dots, y_K \sim \pi^{\text{SFT}}(y | x)$, a user would output a permutation $\tau: \{1, \dots, K\} \rightarrow \{1, \dots, K\}$ indicating the preferred order of the choices. The probability of observing this ranking is given by:

... / ... / ...

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 π_θ 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 θ , 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 $\sigma'(z) = \sigma(z)(1 - \sigma(z)) = \sigma(z)\sigma(-z)$.

- (g) Throughout this problem, we have worked with the Bradley-Terry model to model human preferences. However, in many real-world scenarios, labelers rank multiple model outputs y_1, \dots, y_K rather than just comparing pairs. This is modeled by the Plackett-Luce model, which is a generalization of the Bradley-Terry model. In a similar fashion, the PL model stipulates that when presented with a set of possible choices, people prefer a choice with a probability proportional to the value of some latent reward function for that choice. In our context, when presented with a prompt x and a set of K answers $y_1, \dots, y_K \sim \pi^{\text{SFT}}(y | x)$, a user would output a permutation $\tau: \{1, \dots, K\} \rightarrow \{1, \dots, K\}$ indicating the preferred order of the choices. The probability of observing this ranking is given by:

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

Homework 13, © Faculty teaching EECS 182, Fall 2025. All Rights Reserved. This may not be publicly shared without explicit permission.

4

Homework 13 @ 2025-12-03 22:16:54-08:00

Using the implicit reward parameterization derived in part (d) ($r_\phi(x, y) = \beta \log \frac{\pi_{\theta^*}(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x)$), show that the partition function $Z(x)$ cancels out here as well. Write the final Plackett-Luce probability solely in terms of π_{θ^*} and π_{ref} .

Once you've written out that final probability, assuming we have access to a dataset $\mathcal{D} = \{(\tau^{(i)}, x^{(i)}, \{y_j^{(i)}\}_{j=1}^K)\}_{i=1}^N$ of prompts and user-specified rankings, we can use a parameterized model and optimize this objective with maximum-likelihood:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(\tau, y_1, \dots, y_K, x\tau) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_{\tau(k)} | x)}{\pi_{\text{ref}}(y_{\tau(k)} | x)} - \beta \log \frac{\pi_\theta(y_{\tau(j)} | x)}{\pi_{\text{ref}}(y_{\tau(j)} | x)} \right) \right]$$

here is 2a-g

12/9/2025, 4:45:06 PM

I'll work through each part of Problem 2 on Direct Preference Optimization (DPO).

Problem 2: Honey, Where's My Reward Model?

Context: RLHF pipeline has 3 phases: SFT → Reward Modeling → RL Fine-Tuning.
DPO bypasses the explicit reward model and RL loop.

(a) Find the optimal p^* for $\min_{p \in \mathcal{P}} \mathbb{D}_{\text{KL}}(p \| q)$

Given: $\mathbb{D}_{\text{KL}}(p \| q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \geq 0$

To minimize, we use the method of Lagrange multipliers with the constraint $\sum_x p(x) = 1$:

$$\mathcal{L} = \sum_x p(x) \log \frac{p(x)}{q(x)} - \lambda \left(\sum_x p(x) - 1 \right)$$

Taking the derivative with respect to $p(x)$: $\frac{\partial \mathcal{L}}{\partial p(x)} = \log \frac{p(x)}{q(x)} + 1 - \lambda = 0$

This gives: $\log p(x) - \log q(x) = \lambda - 1$ $p(x) = e^{\lambda - 1} q(x)$

Using the constraint $\sum_x p(x) = 1$: $e^{\lambda - 1} \sum_x q(x) = e^{\lambda - 1} \cdot 1 = 1$ $e^{\lambda - 1} = 1$

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

The KL divergence is minimized when p equals q , giving $\mathbb{D}_{\text{KL}}(p^* \| q) = 0$.

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

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

For fixed x , this is: $\max_{\pi_\theta(\cdot|x)} \sum_y \pi_\theta(y|x) r_\phi(x,y) - \beta \sum_y \pi_\theta(y|x) \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$

Rearranging: $\max_{\pi_\theta(\cdot|x)} \sum_y \pi_\theta(y|x) [r_\phi(x,y) - \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}]$

$$= \max_{\pi_\theta(\cdot|x)} \sum_y \pi_\theta(y|x) \left[\frac{1}{\beta} r_\phi(x,y) + \log \pi_{\text{ref}}(y|x) \right] - \beta \sum_y \pi_\theta(y|x) \log \pi_\theta(y|x)$$

This is equivalent to minimizing: $D_{KL}(\pi_\theta(\cdot|x) \| \pi_{\text{ref}}(y|x)) = \frac{1}{Z(x)} \sum_y \pi_\theta(y|x) \exp \left(\frac{1}{\beta} r_\phi(x,y) \right) - \log Z(x)$

From part (a), the optimal π_{θ^*} equals the target distribution:

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

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

(c) Why is this representation hard to utilize in practice?

Answer: The output space \mathcal{Y} can be extremely large (or even infinite for text generation).

For language models, \mathcal{Y} consists of all possible sequences of tokens, which grows exponentially with sequence length. Computing the partition function $Z(x) = \sum_y \pi_{\text{ref}}(y|x) \exp \left(\frac{1}{\beta} r_\phi(x,y) \right)$ requires summing over all possible outputs, which is computationally intractable.

(d) Express the reward function $r_\phi(x,y)$ in terms of π_{θ^*} , π_{ref} , and $Z(x)$

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

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

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

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

(e) Substitute into Bradley-Terry model and show that $Z(x)$ cancels

The Bradley-Terry model is: $p^*(y_w | \text{succ } y_l / x) = \frac{\exp(r^\wedge(x, y_w))}{\exp(r^\wedge(x, y_w)) + \exp(r^\wedge(x, y_l))}$

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

$$\begin{aligned} p^*(y_w \succ y_l | x) &= \frac{\exp \left(\beta \log \frac{\pi_{\theta^*}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} + \beta \log Z(x) \right)}{\exp \left(\beta \log \frac{\pi_{\theta^*}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} + \beta \log Z(x) \right) + \exp \left(\beta \log \frac{\pi_{\theta^*}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} + \beta \log Z(x) \right)} \\ &= \frac{Z(x)^\beta \left(\frac{\pi_{\theta^*}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} \right)^\beta}{Z(x)^\beta \left(\frac{\pi_{\theta^*}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} \right)^\beta + Z(x)^\beta \left(\frac{\pi_{\theta^*}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)^\beta} \end{aligned}$$

The $Z(x)^\beta$ terms cancel:

$$p^*(y_w \succ y_l | x) = \frac{\left(\frac{\pi_{\theta^*}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} \right)^\beta}{\left(\frac{\pi_{\theta^*}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} \right)^\beta + \left(\frac{\pi_{\theta^*}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)^\beta}$$

Why is this desirable? We no longer need to compute the intractable partition function $Z(x)$! The preference probability depends only on the ratio of policies,

which we can compute directly.

(f) Derive the gradient of DPO loss and interpret the weighting term

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

The DPO loss is: $\mathcal{L} = -\mathbb{E}_{(x, y_w, y_l)} [\log \sigma(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)})]$

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

Then $\mathcal{L} = -\log \sigma(z)$.

Using $\sigma'(z) = \sigma(z)(1 - \sigma(z)) = \sigma(z)\sigma(-z)$ and the chain rule:

$$\frac{\partial \mathcal{L}}{\partial \theta} = -\frac{\sigma'(z)}{\sigma(z)} \frac{\partial z}{\partial \theta} = -\sigma(-z) \frac{\partial z}{\partial \theta}$$

Now: $\frac{\partial z}{\partial \theta} = \beta \frac{\partial}{\partial \theta} \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \frac{\partial}{\partial \theta} \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}$

$$= \beta \nabla_\theta \log \pi_\theta(y_w|x) - \beta \nabla_\theta \log \pi_\theta(y_l|x)$$

Therefore: $\boxed{\nabla_\theta \mathcal{L} = -\beta \sigma(-z) [\nabla_\theta \log \pi_\theta(y_w|x) - \nabla_\theta \log \pi_\theta(y_l|x)]}$

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

Interpretation of weighting term $\sigma(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w))$:

- When $\hat{r}_\theta(x, y_w) \gg \hat{r}_\theta(x, y_l)$ (model is **correct**, strongly prefers winner): $\sigma(-z) \approx 0$, so gradient is small → **updates less**

- When $\hat{r}_\theta(x, y_w) \ll \hat{r}_\theta(x, y_l)$ (model is **incorrect**, prefers loser): $\sigma(-z) \approx 1$, so gradient is large → **updates more**

The gradient update is larger when the model is incorrect, providing adaptive learning that focuses on mistakes.

(g) Plackett-Luce model - show $Z(x)$ cancels

Given the Plackett-Luce probability for ranking τ :

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

Using the implicit reward from part (d): $r_\phi(x, y) = \beta \log \frac{\pi_{\theta^*}(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x)$

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

Each term in the product becomes:

$$\frac{\exp(r_\phi(x, y_{\tau(k)}))}{\sum_{j=k}^K \exp(r_\phi(x, y_{\tau(j)}))} = \frac{Z(x)^\beta}{\left(\frac{\pi_{\theta^*}(y_{\tau(k)}|x)}{\pi_{\text{ref}}(y_{\tau(k)}|x)} \right)^\beta \sum_{j=k}^K \left(\frac{\pi_{\theta^*}(y_{\tau(j)}|x)}{\pi_{\text{ref}}(y_{\tau(j)}|x)} \right)^\beta}$$

The $Z(x)^\beta$ cancels from numerator and denominator:

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

Therefore, the final Plackett-Luce probability is:

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

This is solely in terms of π_{θ^*} and π_{ref} , with no partition function $Z(x)$!

The corresponding DPO loss becomes:

$$\mathcal{L}_{\text{DPO}}(\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(\tau, y_1, \dots, y_K, x|\tau)} [\log \sigma[\beta \log \frac{\pi_{\theta}(y_{\tau(k)}|x)}{\pi_{\text{ref}}(y_{\tau(k)}|x)} + \beta \log \frac{\pi_{\theta}(y_{\tau(j)}|x)}{\pi_{\text{ref}}(y_{\tau(j)}|x)}]]$$

for appropriate pairs in the ranking.

12/9/2025, 4:46:02 PM

Question 1

(a) Correct

Claude correctly used the steps of adding independent Gaussians to solve the problem.

(b) Correct

Claude correctly extended the logic from (a) to obtain $X_{\{t-\Delta t\}} \sim N(0, \sigma^2 + t - \Delta t)$ and $X_t \sim N(0, \sigma^2 + t)$ by recognizing variance accumulates linearly with time in the forward diffusion process.

(c) Correct

Claude properly simplified $(\sigma^2 + t - \Delta t)\Delta t / (\sigma^2 + t)$ by factoring and recognizing that when $\Delta t \ll \sigma^2 \ll 1$, the term $\Delta t / (\sigma^2 + t)$ is negligible, yielding the approximation $\text{Var} \approx \Delta t$. The algebraic steps were clear and justified.

(d) Correct

Claude successfully demonstrated the telescoping product $\hat{X}_0 = X_1 \cdot \prod(\sigma^2 + 1 - k\Delta t) / (\sigma^2 + 1 - (k+1)\Delta t)$, correctly identified this telescopes to $(\sigma^2 / (\sigma^2 + 1)) \cdot X_1$, and concluded $\text{Var}(\hat{X}_0) \approx \sigma^4 \ll \sigma^2$. The explanation clearly identified why this is problematic. The variance collapses too much without stochastic noise.

(e) Correct, good derivation.

Claude set up the Riemann sum correctly, converted to the integral $\int_0^1 \sigma^4 / (\sigma^2 + s)^2 ds$, and then performed the integration correctly and explicitly.

- Applied the hint $\int 1/(1+t)^2 dt = C - 1/(1+t)$
- Evaluated $\sigma^4 [1/(\sigma^2 + s)]_0^1 = \sigma^4 (1/\sigma^2 - 1/(\sigma^2 + 1))$
- Simplified to $\sigma^2 \cdot [(\sigma^2 + 1) - \sigma^2] / (\sigma^2 + 1) = \sigma^2 / (\sigma^2 + 1)$
- Combined with the deterministic term $(\sigma^2 / (\sigma^2 + 1))^2$ to get total variance
- Showed the first term $\approx \sigma^4 \ll \sigma^2$ when $\sigma^2 \ll 1$
- Concluded $\text{Var}(\hat{X}_0) \approx \sigma^2 / (\sigma^2 + 1) \approx \sigma^2$

(f) Correct with good procedure

1. Specified inputs: (x_t, t) pairs
2. Detailed batch generation procedure:
 - Sample s_i from dataset
 - Sample $t \sim \text{Uniform}(0,1)$
 - Generate $x_t = s_i + \epsilon$ where $\epsilon \sim N(0, t)$
 - Form pairs (x_t, t) with targets $g^*(x_t, t) = \sigma^2 / (\sigma^2 + t) \cdot x_t$
3. Specified loss function: MSE between network output $f_\theta(x_t, t)$ and target

- Clear objective: $L = E[|f_\theta(x_t, t) - \sigma^2/(\sigma^2 + t) \cdot x_t|^2]$

Claude gave a good level of specificity (which is what the problem requested) and gave actionable implementation details.

(g) Correct

Claude approximated $\eta(t, \Delta t) = \sqrt{t}/(\sqrt{t-\Delta t} + \sqrt{t})$ for $\Delta t \ll t$:

- Factored as $\sqrt{t}/(\sqrt{t}(\sqrt{1-\Delta t/t} + 1))$
- Approximated $\sqrt{1-\Delta t/t} \approx 1 - \Delta t/(2t)$ using $\sqrt{1-x} \approx 1 - x/2$
- Obtained $\eta \approx 1/(2 - \Delta t/(2t)) \approx 1/2$
- Derived final form: $x_{\{t-\Delta t\}} \approx x_t(1 - \Delta t/(2(\sigma^2 + t)))$

The approximation was properly justified and the algebraic steps were shown clearly.

(h) Correct

Claude provided the complete calculation:

- Started with the product: $\hat{X}_0 = \hat{X}_1 \prod (1 - \Delta t/(2(\sigma^2 + 1 - k\Delta t)))$
- Took logarithms: $\ln(\hat{X}_0/\hat{X}_1) = \sum \ln(1 - \Delta t/(2(\sigma^2 + 1 - k\Delta t)))$
- Applied $\ln(1-x) \approx -x: \approx -\sum \Delta t/(2(\sigma^2 + 1 - k\Delta t))$
- Converted to Riemann integral: $\rightarrow -\int_0^1 ds/(2(\sigma^2 + s))$
- Evaluated: $= -(1/2)[\ln(\sigma^2 + s)]_0^1 = -(1/2)\ln((\sigma^2 + 1)/\sigma^2)$
- Exponentiated: $\hat{X}_0 = \hat{X}_1 \cdot \sqrt{(\sigma^2/(\sigma^2 + 1))}$
- Concluded: $\hat{X}_0 \sim N(0, \sigma^2/(\sigma^2 + 1)) \approx N(0, \sigma^2)$

Claude showed every step of converting the product to a sum, sum to integral, evaluating the integral, and exponentiating back.

Questions 2:

(a) Correct

Claude correctly solved using Lagrange multipliers, took derivatives with respect to $p(x)$, set equal to zero, and obtained $p^*(x) = q(x)$. The derivation was complete and properly showed that $e^{(\lambda-1)} = 1$ from the normalization constraint.

(b) Correct

Claude was able to solve this correctly with explicit derivation.

- Wrote out the objective for fixed x : $\max \sum \pi_\theta(y|x)[r_\phi(x,y) - \beta \log(\pi_\theta(y|x)/\pi_{ref}(y|x))]$
- Rearranged to show it's equivalent to minimizing KL divergence from $(1/Z(x))\pi_{ref}(y|x)\exp((1/\beta)r_\phi(x,y))$
- Applied result from part (a) to conclude the optimal policy equals this distribution

4. Identified $Z(x)$ as the partition function

One thing I noticed is that Claude did not show every Lagrange multiplier calculation step but it did show the connection to KL minimization and showing how the optimal form emerges. This demonstrates understanding of the constrained optimization structure.

(c) Correct

Claude correctly identified that computing $Z(x) = \sum_y \pi_{\text{ref}}(y|x) \cdot \exp(r_\phi(x,y)/\beta)$ is intractable for large output spaces like text generation.

(d) Correct

Claude correctly rearranged: $\log \pi_{\theta^*}(y|x) = -\log Z(x) + \log \pi_{\text{ref}}(y|x) + (1/\beta)r_\phi(x,y)$, then solved to get $r_\phi(x,y) = \beta \log(\pi_{\theta^*}(y|x)/\pi_{\text{ref}}(y|x)) + \beta \log Z(x)$. The algebra was shown step-by-step.

(e) Correct

Claude substituted the reward parametrization into the Bradley-Terry formula, showed that $\exp(r_\phi(x,y)) = Z(x)^\beta \cdot (\pi_{\theta^*}(y|x)/\pi_{\text{ref}}(y|x))^\beta$, demonstrated the $Z(x)^\beta$ terms cancel from numerator and denominator, and clearly explained why this is desirable.

(f) Correct

1. Defined $z = \hat{r}_\theta(x, y_w) - \hat{r}_\theta(x, y_l)$ and $L = -\log \sigma(z)$
2. Applied chain rule: $\partial L / \partial \theta = -(\sigma'(z) / \sigma(z)) \cdot \partial z / \partial \theta$
3. Used $\sigma'(z) = \sigma(z)\sigma(-z)$ to get $\partial L / \partial \theta = -\sigma(-z) \cdot \partial z / \partial \theta$
4. Computed $\partial z / \partial \theta = \beta \nabla_\theta \log \pi_\theta(y_w|x) - \beta \nabla_\theta \log \pi_\theta(y_l|x)$
5. Combined to get final form: $\nabla_\theta L_{\text{DPO}} = -\beta \sigma(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w)) \cdot (\nabla_\theta \log \pi_\theta(y_w|x) - \nabla_\theta \log \pi_\theta(y_l|x))$

Claude explicitly showed the chain rule application and intermediate steps. The interpretation of the sigmoid weighting was also correct: large gradient when model is wrong ($\sigma(-z) \approx 1$), small when correct ($\sigma(-z) \approx 0$).

(g) Correct

Claude substituted $r_\phi(x,y) = \beta \log(\pi_{\theta^*}(y|x)/\pi_{\text{ref}}(y|x)) + \beta \log Z(x)$ into the Plackett-Luce formula, showed that $\exp(r_\phi(x,y)) = Z(x)^\beta \cdot (\pi_{\theta^*}(y|x)/\pi_{\text{ref}}(y|x))^\beta$, demonstrated that $Z(x)^\beta$ appears in both numerator and denominator of each term in the product and cancels, and wrote the final probability depending only on π_{θ^*} and π_{ref} . The algebra was thorough and correct.

Summary

Overall, I am very impressed with how Claude did on this homework. I was expecting it to solve most of the earlier parts correctly (which it did) but I was pleasantly surprised it was able to do the full derivations for the later parts correctly and (mostly) in full. I think some of the solutions could have used a little more detail, such as in 2b, but overall definitely very good. I would say it solved every question correctly. All the derivations were very explicit and easy to understand, and Claude rarely skipped steps in the process even when the traces were getting long. All of this was in one attempt.

For example In part 1(e), the integral evaluation required several non-trivial steps: recognizing the Riemann sum structure, applying the substitution correctly, evaluating the antiderivative at the bounds, and then combining terms. And claude was able to execute each step well.

Similarly, in part 1(h), the product-to-integral conversion involved taking logarithms, applying a first-order approximation, recognizing the Riemann sum, evaluating the integral, and exponentiating back. This is a long chain of reasoning where earlier mistakes could cascade through the rest of the calculation, but Claude was able to maintain accuracy throughout.

Part 2(b) reveals an interesting aspect of Claude's approach. While it didn't write out every single step of the Lagrange multiplier calculation (taking derivatives, setting to zero, solving the system), it showed the key conceptual insight: recognizing that the constrained optimization problem is equivalent to KL minimization from a specific target distribution. This suggests Claude prioritizes conveying the mathematical structure and reasoning over mechanical symbol manipulation.