EECS 182 Deep Neural Networks
Fall 2025 Anant Sahai and Gireeja Ranade

Homework 13

This homework is due on Wednesday December 10th, at 10:59PM.

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 $\widehat{X}_0$. **What is the resulting distribution of $\widehat{X}_0$?** 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 $\widehat{X}_0$ 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 $\widehat{X}_0$ 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

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

Take the limit $\Delta t \to 0$ and **write Var$(\widehat{X}_0)$ 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 $\widehat{X}_0^* = \frac{\sigma^2}{\sigma^2+t}X_t$. Namely that

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

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 $s_1, s_2, \ldots, s_n$ 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 $\widehat{X}_0$ from the random sample $\widehat{X}_1$ (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 $\Delta t \to 0$ limit, using integrals as needed to evaluate the resulting distribution for $\widehat{X}_0$. **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.)

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 $(y_1, y_2) \sim \pi^{\text{SFT}}(y \mid x)$ 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?

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:

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

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:

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

In the context of LMs, $r^*(x, y)$ is often initialized from the SFT model $\pi^{\text{SFT}}(y \mid x)$ 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.

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

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

where β is a parameter controlling the deviatation 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 $r(x, y) = r_\phi(x, y) - \beta(\log \pi_\theta(y \mid x) - \log \pi_{\text{ref}}(y \mid x))$, 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:

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

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

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

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

(c) Even though we've obtained the optimal policy $\pi_{\theta^*}(y \mid x)$, 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

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

which is exactly the result we wanted!

Recall the complex RLHF pipeline we started with: **SFT** $\rightarrow$ **Reward Modeling** $\rightarrow$ **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 $y_1, \ldots, y_K$ 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 prportional 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 $y_1, \ldots, y_K \sim \pi^{\text{SFT}}(y \mid x)$, a user would output a permutation $\tau\colon \{1, \ldots, K\} \rightarrow \{1, \ldots, K\}$ indicating the preferred order of the choices. The probability of observing this ranking is given by:

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

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:

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

3. Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student!
We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

(a) **What sources (if any) did you use as you worked through the homework?**

(b) **If you worked with someone on this homework, who did you work with?**
List names and student ID's. (In case of homework party, you can also just describe the group.)

(c) **Roughly how many total hours did you work on this homework?**

The coverage of the following question is redundant. It is presented here to aid with exam preparation.

4. Diffusion Models

Previously in discussion, we considered sampling from a discrete distribution. Let's now see how iteratively adding Gaussian noise to a data point leads to a noisy sequence, and how the reverse process refines noise to generate realistic samples.

The classes of generative models we've considered so far (VAEs, GANs), typically introduce some sort of bottleneck (*latent representation* $\mathbf{z}$) that captures the essence of the high-dimensional sample space $(\mathbf{x})$. An alternate view of representing probability distributions $p(\mathbf{x})$ is by reasoning about the *score function* i.e. the gradient of the log probability density function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$.

Given a data point sampled from a real data distribution $\mathbf{x}_0 \sim q(\mathbf{x})$, let us define a *forward diffusion process* iteratively adding small amount of Gaussian noise to the sample in T steps, producing a sequence of noisy samples $\mathbf{x}_1, ..\mathbf{x}_T$.

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t I) \qquad q(\mathbf{x}_{1:T}|x_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{2}$$

The data sample $\mathbf{x}_0$ gradually loses its distinguishable features as the step t becomes larger. Eventually when $T \to \infty$, $\mathbf{x}_T$ is equivalent to an isotropic Gaussian distribution. (You can assume $\mathbf{x}_0$ is Gaussian).

To generative model is therefore the *reverse diffusion process*, where we sample noise from an isotropic Gaussian, and iteratively refine it towards a realistic sample by reasoning about $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$.

(a) **Anytime Sampling from Intermediate Distributions**
Given $\mathbf{x}_0$ and the stochastic process in eq. (2), **show that there exists a closed form distribution for sampling directly at the t^{th} time-step of the form**

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)I)$$

(b) **Reversing the Diffusion Process**
Reversing the diffusion process from *real* to *noise* would allow us to sample from the real data distribution. In particular, we would want to draw samples from $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$. **Show that given $\mathbf{x}_0$, the reverse conditional probability distribution is tractable and given by**

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \mu(\mathbf{x}_t, \mathbf{x}_0), \hat{\beta}_t I)$$

(Hint: Use Bayes Rule on eq. (2) , assuming that $\mathbf{x}_0$ is drawn from Gaussian $q(\mathbf{x})$)

Contributors:

- Anant Sahai.
- Patrick Mendoza.
- Kumar Krishna Agrawal.