Generative Model Midterm Solutions

whoami

2025 年 11 月 22 日

Problem 1 (a):

Training Phase（训练阶段）:

- RNN: The complexity is $O(n)$. 因为它是按顺序一个接一个算的 (Sequential)。
- Transformer: The complexity is $O(n^2)$. 因为 Self-Attention 要算两两之间的关系。
- **Conclusion:** Transformer is more training-efficient (RNN 的第 n 个要等前面的第 n-1 个算完, Transformer 可以并行计算).

Inference Phase（推理阶段）:

- RNN: Complexity is $O(n)$ total.
- Transformer: Complexity is $O(n^2)$ total.
- **Conclusion:** RNN is more inference-efficient .

Problem 1 (b): Positional Embeddings

Self-Attention 机制本质上是将输入视为一个无序集合并行处理的，输入输出是排列等变的，只计算内容相似度而忽略距离，因此缺乏捕捉序列顺序（如语法结构）的内在能力，需要位置编码。

Problem 1 (c): Self-Attention Mechanism

Given queries $\mathbf{q}$, keys $\mathbf{k}$, and values $\mathbf{v}$. The output is calculated as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

$$\mathbf{o}_i = \sum_{j=1}^{n} \alpha_{ij}\mathbf{v}_j, \quad \text{where} \quad \alpha_{ij} = \frac{\exp\left(\frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d}}\right)}{\sum_{k=1}^{n} \exp\left(\frac{\mathbf{q}_i^\top \mathbf{k}_k}{\sqrt{d}}\right)}$$

Problem 1 (d): Layer Normalization

题目给出的输入 $\mathbf{X}$ 是 4 个向量。LayerNorm 是对每一个向量单独做归一化 (mean=0, std=1)。

Step 1: Analyze Input For the first vector $x_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$:

$$\text{Mean } \mu = \frac{2+1}{2} = 1.5$$

$$\text{Variance } \sigma^2 = \frac{(2-1.5)^2 + (1-1.5)^2}{2} = 0.25 \implies \sigma = 0.5$$

Step 2: Normalize

$$\hat{x}_1 = \frac{x_1 - \mu}{\sigma} = \begin{bmatrix} (2-1.5)/0.5 \\ (1-1.5)/0.5 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

同理 (Similarly)，对于其他向量计算后，最终结果都是 $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$ 或 $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$。

Final Answer:

$$\mathbf{X}_{out} = \left[\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right]$$

Problem 2 (a): ELBO Proof (via KL Divergence)

We derive the lower bound by analyzing the log-likelihood $\log p_\theta(\mathbf{x})$. Since $\int q_\phi(\mathbf{z}|\mathbf{x})d\mathbf{z} = 1$, we can write:

$$\begin{aligned}
\log p_\theta(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right] \quad (\text{Bayes Rule: } p(\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{z})}{p(\mathbf{z}|\mathbf{x})}) \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \cdot \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right)\right] \quad (\text{Multiply by } \frac{q}{q}) \\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]}_{\text{ELBO}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]}_{D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))}
\end{aligned}$$

Since the KL divergence is always non-negative ($D_{KL} \geq 0$):

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]$$

Thus, the variational lower bound is proved.

Problem 2 (b): Reparameterization Trick

1. 什么是重参数化技巧 (What is the trick?)

重参数化技巧是将随机变量 $\mathbf{z}$ 表示为确定性参数和独立噪声的函数。我们不再直接从分布 $\mathcal{N}(\mu, \sigma^2)$ 中采样，而是引入一个标准高斯噪声 $\epsilon \sim \mathcal{N}(0, \mathbf{I})$，通过以下公式生成 $\mathbf{z}$:

$$\mathbf{z} = \mu + \sigma \odot \epsilon$$

其中 μ 和 σ 是 Encoder 网络的输出，$\odot$ 表示逐元素相乘。

2. 为什么需要它 (Why do we need it?)

正如题目背景所述，在 Encoder 部分，输入 $\mathbf{x}$ 被映射为隐空间分布的参数 μ 和 σ。

- **直接采样的断路问题**：如果我们直接从 $\mathcal{N}(\mu, \sigma^2)$ 中采样 $\mathbf{z}$，这是一个不可导的随机操作（Stochastic operation）。计算图（Computational Graph）在此处中断，导致 Loss 对 Encoder 参数（即对 μ 和 σ）的梯度无法通过反向传播（Backpropagation）传回去。
- **重参数化的解决之道**：通过变换 $\mathbf{z} = \mu + \sigma \odot \epsilon$，所有的随机性被转移到了外部常量 ϵ 上。此时，对于网络参数 μ 和 σ 而言，$\mathbf{z}$ 变成了一个平滑的可导函数（加法和乘法）。这打通了梯度流，使得神经网络可以进行端到端的训练。

Problem 3: GAN (20 points)

Problem 3 (a): Training and Inference Process

GAN consists of two networks: a **Generator** f_G (creates fake data from noise) and a **Discriminator** f_D (distinguishes real data from fake).

1. Training Process (Min-Max Game)

The training is a zero-sum game where f_D tries to maximize the discrimination accuracy, while f_G tries to minimize it (fool the discriminator). The objective function is:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))]$$

The process is usually iterative:

- **Step 1 (Train Discriminator):** Sample a batch of real data x and noise z. Fix the Generator's parameters. Update f_D to **maximize** the probability of assigning 1 to real data and 0 to fake data $G(z)$.
- **Step 2 (Train Generator):** Sample a new batch of noise z. Fix the Discriminator's parameters. Update f_G to **minimize** $\log(1 - D(G(z)))$ (or practically, maximize $\log D(G(z))$) to trick the discriminator.

2. Inference Process

During inference, the Discriminator f_D is discarded. We only use the trained Generator f_G:

$$z \sim p_z \quad \xrightarrow{f_G} \quad \hat{x} = f_G(z)$$

We sample random noise z (e.g., from Gaussian) and pass it through f_G to generate new samples.

Problem 3 (b): KL Divergence as f-divergence

Definition of f-divergence: Given two distributions P and Q with densities $p(x)$ and $q(x)$, the f-divergence is defined as:

$$D_f(P\|Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

where $f(t)$ is a convex function with $f(1) = 0$.

Proof: We need to find a specific function $f(t)$ such that D_f becomes the KL divergence. Let us choose the generator function:

$$f(t) = t \log t$$

1. Check convexity and constraint:

- Defined on the domain $t > 0$, non-negative.
- Derivative: $f'(t) = 1 + \log t$.
- Second derivative: $f''(t) = \frac{1}{t}$. Since t represents a probability ratio $\frac{p(x)}{q(x)}$, $t > 0$, so $f''(t) > 0$. Thus, $f(t)$ is convex.
- Constraint: $f(1) = 1 \log 1 = 0$. Satisfied.

2. **Substitution:** Substitute $f(t) = t \log t$ and $t = \frac{p(x)}{q(x)}$ into the f-divergence formula:

$$\begin{aligned} D_f(P\|Q) &= \int q(x) \left[\frac{p(x)}{q(x)} \log \left(\frac{p(x)}{q(x)} \right) \right] dx \\ &= \int p(x) \log \frac{p(x)}{q(x)} dx \\ &= D_{KL}(P\|Q) \end{aligned}$$

Thus, KL divergence is a special case of f-divergence where $f(t) = t \log t$.

Problem 4: Normalizing Flow (20 points)

Problem 4 (a): Change of Variables Theorem

Given the relationship $\mathbf{X} = g(\mathbf{Z})$ where g is an invertible and smooth function, and $\mathbf{Z} \sim p_{\mathbf{Z}}(z)$. According to the multivariate change of variables theorem, the probability density $p_{\mathbf{X}}(x)$ is:

$$p_{\mathbf{X}}(x) = p_{\mathbf{Z}}(z) \left| \det \left(\frac{\partial g^{-1}(x)}{\partial x} \right) \right|$$

Alternatively, it can be written using the forward transformation Jacobian:

$$p_{\mathbf{X}}(x) = p_{\mathbf{Z}}(g^{-1}(x)) \cdot \left| \det \left(\frac{\partial g(z)}{\partial z} \right) \right|^{-1}$$

Intuition: Probability mass must be conserved ($p(x)dx = p(z)dz$). The term $\left|\det \frac{\partial z}{\partial x}\right|$ represents the change in volume (distortion) caused by the transformation.

Problem 4 (b): Designing Invertible MLP

We define the layer as $g(\mathbf{x}) = \sigma(W\mathbf{x} + \mathbf{b})$. To ensure $g(\mathbf{x})$ is invertible and the Jacobian determinant is easy to compute, we impose the following constraints:

1. **Weights W (Constraint: Triangular):**
   We restrict W to be an **Upper (or Lower) Triangular Matrix** with non-zero diagonal elements.

   $$W_{ij} = 0 \text{ if } i > j \quad \text{(Upper Triangular)}$$

   Reason: This makes the determinant simply the product of diagonal elements ($\det W = \prod w_{ii}$), and triangular matrices are invertible if diagonals are non-zero.

2. **Activation σ (Constraint: Strictly Monotonic):**

   We choose a strictly increasing function, such as **LeakyReLU** or **PReLU**.

$$\sigma(y) = \begin{cases} y & \text{if } y > 0 \\ \alpha y & \text{if } y \leq 0 \end{cases} \quad (\text{where } \alpha > 0)$$

   Reason: Strictly monotonic functions are bijective (invertible). ReLU is not suitable because it is not invertible for negative values (maps to 0).

3. **Bias b:** No specific constraint (translation is always invertible).

Problem 4 (c): Training and Inference Algorithm

Since Normalizing Flows are generative models trained by **Maximum Likelihood Estimation (MLE)**.

1. Training Objective (Exact Log-Likelihood)

Our goal is to maximize the log-likelihood of the observed data $\mathbf{x}$. Let $\mathbf{z} = g^{-1}(\mathbf{x})$ (mapping data to latent space). The loss function (negative log-likelihood) to minimize is:

$$\mathcal{L} = -\log p_{\mathbf{X}}(\mathbf{x}) = -\left(\log p_{\mathbf{Z}}(g^{-1}(\mathbf{x})) + \log\left|\det \frac{\partial g^{-1}(\mathbf{x})}{\partial \mathbf{x}}\right|\right)$$

We update parameters to minimize $\mathcal{L}$ using gradient descent.

2. Inference Algorithm (Sampling)

To generate new data samples:

1. Sample a latent vector from the prior distribution: $\mathbf{z} \sim p_{\mathbf{Z}}(z)$ (usually $\mathcal{N}(0, \mathbf{I})$).
2. Transform it to data space using the forward network: $\mathbf{x} = g(\mathbf{z})$.

Problem 5: Diffusion Models (20 points)

Problem 5 (a): Forward Process

The forward process (diffusion process) is a Markov chain that gradually adds Gaussian noise to the data according to a variance schedule $\beta_1, \ldots, \beta_T$.

For a single step transition from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$:

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

The joint distribution for the entire forward process is:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

Problem 5 (b): Marginal Distribution Proof

We define $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. Using the reparameterization trick, we can express $\mathbf{x}_t$ in terms of $\mathbf{x}_{t-1}$:

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1}, \quad \text{where } \epsilon_{t-1} \sim \mathcal{N}(0, \mathbf{I})$$

We can expand this recursively. Substitute $\mathbf{x}_{t-1}$:

$$\begin{aligned}\mathbf{x}_t &= \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon_{t-2}) + \sqrt{1-\alpha_t}\epsilon_{t-1} \\ &= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})}\epsilon_{t-2} + \sqrt{1-\alpha_t}\epsilon_{t-1}\end{aligned}$$

Since the sum of two independent Gaussians $\mathcal{N}(0, \sigma_1^2) + \mathcal{N}(0, \sigma_2^2)$ is $\mathcal{N}(0, \sigma_1^2 + \sigma_2^2)$, the noise terms merge. By induction to $t = 0$:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

Thus, the marginal distribution is:

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

Problem 5 (c): Posterior Distribution Proof

We want to find $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. Using Bayes' rule:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}$$

Since all distributions are Gaussian, the posterior is also Gaussian. We focus on the exponent terms (completing the square for $\mathbf{x}_{t-1}$):

$$\begin{aligned}&\log q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \\ &\propto -\frac{1}{2\beta_t}\|\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1}\|^2 - \frac{1}{2(1-\bar{\alpha}_{t-1})}\|\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\|^2 + C\end{aligned}$$

By expanding the squares and matching coefficients with the standard Gaussian form $-\frac{1}{2\tilde{\beta}_t}\|\mathbf{x}_{t-1} - \tilde{\mu}_t\|^2$, we derive the mean $\tilde{\mu}_t$ and variance $\tilde{\beta}_t$:

Variance $\tilde{\beta}_t$: The coefficient of $\mathbf{x}_{t-1}^2$ gives $\frac{1}{\tilde{\beta}_t} = \frac{1}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}$, which simplifies to $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$.

Mean $\tilde{\mu}_t$: By matching the linear term of $\mathbf{x}_{t-1}$, we obtain:

$$\tilde{\mu}_t(\mathbf{x}_0, \mathbf{x}_t) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0$$

Problem 5 (d): Training and Inference Algorithm

1. Training Algorithm (Learning to Denoise) We train a network ϵ_θ to predict the noise added to the image.

- Sample data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$.
- Sample time step $t \sim \text{Uniform}(\{1, \dots, T\})$.
- Sample noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$.
- Compute noisy image: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$.
- **Minimize Loss (MSE):** $\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2$.

2. Inference Algorithm (Sampling) We start from pure noise and iteratively remove it using the trained network.

- Start with $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.
- For $t = T, \dots, 1$:

$$\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}) \text{ (if } t > 1 \text{ else } 0)$$
$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$$

- Return $\mathbf{x}_0$.