Deep Generative Models
Homework Set 4 (Sol)

Mohammad Parsa Dini, std-id: 400101204

January 30, 2025

1 Problem 1

1.1 Recall of Contrastive Divergence (CD)

Well, the CD is used in many MCMC sampling algorithms and its aim is to find the divergence between the data distribution $p_{data}(.)$ and the model's distribution $p_\theta(.)$. In the case of neural networks and RBMs, CD aims to minimize the energy of the network. So we map this energy to the probability distribution of the model and data from a physical point of view. There is an energy function $E(.;\theta)$ to model the data distribution $p_\theta(.) = \frac{e^{-E(.;\theta)}}{Z_\theta}$. Due to the volume preservability of probability, they must sum to one and we get the partition function $Z_\theta = \sum_{v,h} \exp(-E(v,h;\theta))$, which is intractable since it must be summed over all states and values of the model. Specifically, the energy function $E(v,h;\theta)$ in an RBM is given by:

$$E(v,h;\theta) = -\sum_i b_i v_i - \sum_j c_j h_j - \sum_{i,j} W_{ij} v_i h_j = -b^T v - c^T h - v^T W h$$

The probability distribution over the visible and hidden units in the RBM is:

$$p(v,h;\theta) = \frac{1}{Z(\theta)} \exp(-E(v,h;\theta))$$

where $Z(\theta) = \sum_{v,h} \exp(-E(v,h;\theta))$ (or the other version: $Z(\theta) = \int_\chi \exp(-E(x;\theta))dx$) is the partition function, v represents the visible units, h represents the hidden units, b_i and c_j are the biases for the visible and hidden layers, respectively, and W_{ij} are the weights between the visible and hidden units.

Now it's obvious that if we want to maximize the log-likelihood, we must minimize the energy function. Since optimizing the log-likelihood directly is intractable due to the partition function $Z(\theta)$, we use CD instead, which approximates the gradient of the log-likelihood (which is the energy function) by performing a few steps of Gibbs sampling.

Here is the **Contrastive Divergence** Algorithm:

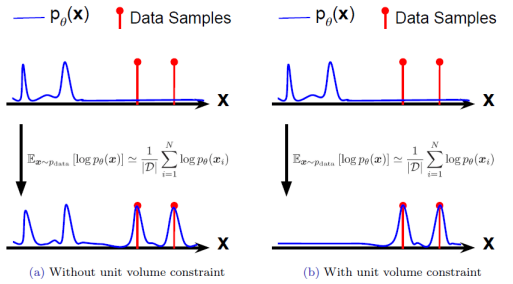
- **Positive Phase**: Sample the hidden units $h(1)$ from the visible units v in the data distribution.
- **Negative Phase**: Perform a few Gibbs sampling steps to reconstruct the visible units $v(2)$ from the hidden units $h(1)$, and then sample the hidden units $h(2)$ from $v(2)$.
- **Update**: Update the parameters based on the difference between the outer products of the data and the reconstructed hidden and visible units:

  $$\Delta W = \eta(\langle vh \rangle_{\text{data}} - \langle vh \rangle_{\text{model}})$$

  where η is the learning rate.

1.2 Introducing Noise Contrastive Estimation (NCE)

A. Using CD to estimate the model's parameters without access to the full partition function Z_θ involves adding energy from extraneous points and incorporating them into the local observed data points. However, without preserving the energy, this approach merely adds energy without altering the energy of other points. Conversely, if we had access to the partition function, the preservation of energy would necessitate that an increase in the probability (or equivalently, a decrease in the energy) of certain data points $E_\theta(x_\theta)$ would require a corresponding decrease in the probability (or equivalently, an increase in the energy) of other points $E_\theta(x_d)$.
The image below is accenting the reason:



(a) Without unit volume constraint (b) With unit volume constraint

Figure: The effect of unit-volume constraint on wiping out the spurious peaks in the generative model

B. Noise Contrastive Estimation (NCE) is a technique designed to address the difficulty of calculating the partition function Z_θ when estimating the parameters of probabilistic models. Instead of directly computing the partition function, which is computationally expensive due to the sum over all possible configurations of the model's variables, NCE proposes a more efficient approach by framing the parameter estimation as a binary classification problem.

The key idea behind NCE is to use noise distributions as a way to contrast the model's data distribution with a simpler distribution that is easier to handle, typically referred to as the "noise distribution." Rather than calculating the exact partition function, NCE aims to learn the parameters of the model by distinguishing between real data samples and samples generated from the noise distribution. The data distribution, $p_{\text{data}}(v)$, is the true distribution we want to model, while the noise distribution, $q(v)$, is typically a simpler distribution, such as a uniform or Gaussian distribution, which is easy to sample from.

For each data sample v, NCE treats it as a binary classification problem: **is the sample drawn from the data distribution $p_{\text{data}}(v)$ or from the noise distribution $q(v)$?** The model then learns to classify each sample as either "real" (from the data distribution) or "fake" (from the noise distribution). This transforms the problem of parameter estimation into a logistic regression task, where the goal is to predict whether a given sample came from the true data distribution or from the noise distribution. For each sample v, a logistic regression model is trained to predict the probability of v being from the data distribution as opposed to the noise distribution. The output of this logistic model is given by:

$$p_{\text{data}}(v) = \frac{1}{1 + \exp(-f(v))}$$

where $f(v)$ is some function of the data, typically parameterized by the model we are trying to learn. This function $f(v)$ is trained using standard binary cross-entropy loss.

The objective in NCE is to maximize the likelihood of distinguishing between data samples and noise samples. The objective function is derived from the logistic regression task, and it does not require knowledge of the partition function Z_θ. The gradient of this objective function with respect to the parameters of the model allows us to update the model without needing to compute Z_θ, as the noise distribution is used as a reference for learning.

1.3 Formulating NCE

A. As we said earlier, in Noise Contrastive Estimation (NCE) the objective is to distinguish between samples drawn from the true data distribution $p_{\text{data}}(x)$ and samples drawn from a noise distribution $q(x)$. The role of the noise distribution in NCE is fundamental to the method, as it provides a contrasting distribution against which the true data distribution is learned. By using the noise distribution, NCE simplifies the problem of estimating the model's parameters by framing it as a binary classification task. This allows the model to estimate the parameters without requiring the explicit computation of the partition function, which would otherwise be intractable.

Choosing a suitable noise distribution is crucial because it impacts the effectiveness of the learning process. The noise distribution should ideally be chosen to be simple enough to sample from easily, yet it should still provide a meaningful contrast to the true data distribution. If the noise distribution is too similar to the data distribution, the model will find it difficult to distinguish between the two, leading to poor learning. On the other hand, if the noise distribution is too different from the data distribution, the binary classification task may become too easy, resulting in suboptimal learning that does not properly reflect the true data distribution.

B. Given a set of data samples $x_1, x_2, \ldots, x_n$ and a noise distribution $q(x)$, the objective function for Noise Contrastive Estimation (NCE) aims to maximize the likelihood of correctly distinguishing between data and noise samples. For each sample x_i, the model learns to classify it as coming from the data distribution $p_{\text{data}}(x)$ or the noise distribution $q(x)$.

The objective function is the log-likelihood of correctly identifying data and noise samples. The probability of x_i being from the data distribution is modeled using a logistic function:

$$p_{\text{data}}(x_i) = \frac{1}{1 + \exp(-f(x_i))}$$

The objective function is the sum of the log-likelihoods for all data samples:

$$\mathcal{L} = \sum_{i=1}^{n} \log \left(\frac{p_{\text{data}}(x_i)}{p_{\text{data}}(x_i) + q(x_i)} \right)$$

Maximizing this objective allows the model to learn the parameters without needing to compute the partition function.

1.4 NCE v.s. CD

A. Contrastive Divergence (CD) and Noise Contrastive Estimation (NCE) are methods used to train probabilistic models, but they differ in goals, learning objectives, and computational requirements.

CD approximates the gradient of the log-likelihood for models like RBMs by minimizing the difference between the model's data distribution and the data after a short MCMC sampling process using a single Gibbs sampling step, making it computationally less expensive but introducing bias.

NCE estimates model parameters without the partition function by maximizing the likelihood of distinguishing between true data and noise samples, framing it as a binary classification task, which simplifies estimation and makes it scalable for large datasets.

B. Noise Contrastive Estimation (NCE) might be preferred over Contrastive Divergence (CD) when the partition function is computationally expensive or intractable, such as in models with large state spaces. NCE eliminates the need for calculating the partition function by reformulating the problem as a binary classification task, which is computationally more efficient, especially for large datasets.

One of the main advantages of NCE over CD is its scalability. Since NCE only requires distinguishing between data and noise samples, it avoids the need for iterative MCMC sampling, which can be slow and prone to introducing bias, especially in models with complex dependencies. This makes NCE more

efficient for models that are difficult to sample from directly, such as deep generative models or models with large latent spaces.

Moreover, NCE can be more robust in settings where the data distribution is difficult to model directly, as it relies on learning to differentiate between the data and a simpler noise distribution, which can lead to more stable learning. In contrast, CD's reliance on a limited number of MCMC steps can sometimes result in biased parameter estimates, especially when the model's distribution is far from the data distribution.

1.5 Practical Use of NCE

A practical application where Noise Contrastive Estimation (NCE) might be used is in training large-scale language models, where the goal is to estimate the probability distribution over words in a given vocabulary. In natural language processing tasks, such as language modeling or word prediction, the partition function is computationally expensive to calculate because it involves summing over all possible words in the vocabulary. This becomes particularly difficult when dealing with large vocabularies, as the number of possible words can be extremely large.

NCE is useful in this scenario because it allows for efficient parameter estimation by avoiding the need to compute the partition function. Instead of directly modeling the entire probability distribution over words, NCE frames the problem as a binary classification task, where the model learns to distinguish between real words from the vocabulary and noise words sampled from a simpler noise distribution. By training the model to differentiate real words from noise, NCE effectively approximates the true distribution over words without requiring the costly computation of the partition function. This makes NCE scalable and computationally efficient, especially for large-scale language models with massive vocabularies.

2 Problem 2

2.1 Theoretical Foundations of Score Functions

A. The score function is the gradient of the log-likelihood function with respect to the model parameters. It provides information about the direction in which the likelihood increases most rapidly and is used to update the model parameters during optimization. The score function is related to the likelihood function because it represents the rate of change of the likelihood with respect to the parameters, and its value can guide the search for optimal parameter values.

Mathematically, the score function $s_\theta(x)$ is given by the derivative of the log-likelihood function $\log p_\theta(x)$ with respect to the parameters θ:

$$s_\theta(x) = \nabla_\theta \log p(x;\theta) = -\nabla_\theta Z_\theta - \nabla_\theta E_\theta(x)$$

where $p(x;\theta) = \frac{e^{-E_\theta(x)}}{Z_\theta}$ is the likelihood function, and x represents the data.

B. In Maximum Likelihood Estimation (MLE), the score function is the gradient of the log-likelihood function and indicates the direction of the greatest increase in likelihood with respect to the parameters θ. The score function is used to find the parameter values that maximize the likelihood of the observed data x.

At the maximum likelihood estimate, the score function equals zero, indicating a local maximum. To find the optimal parameters, we set the score function to zero and solve for θ. This is typically done using numerical optimization methods, which iteratively adjust the parameters until convergence.

Thus, the score function efficiently links the parameters to the likelihood, enabling optimization algorithms to find the maximum likelihood estimate.

2.2 Score Function for Specific Distributions

A. The score function will be calculated as($\theta = (\mu, \sigma)$):

$$s_\theta(x) = \nabla_\theta \log p(x;\theta) = [\nabla_\mu \log p(x;\theta), \nabla_\sigma \log p(x;\theta)]^T = [s_\mu(x), s_\sigma(x)]^T$$

we can compute the score function for both the mean μ and the variance σ^2.

Score function for μ:

$$s_\mu(x) = \frac{\partial}{\partial \mu}\ell(x;\mu,\sigma) = \frac{\partial}{\partial \mu}\sum_{i=1}^{n}\log f(x_i;\mu,\sigma^2) = \frac{\partial}{\partial \mu}\left(-\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2\right)$$

$$= \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i - \mu)$$

Score function for σ^2:

$$s_\sigma(x) = \frac{\partial}{\partial \sigma}\ell(x;\mu,\sigma^2) = \frac{\partial}{\partial \sigma}\left(-\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma}\sum_{i=1}^{n}(x_i-\mu)^2\right)$$

$$= -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^3}\sum_{i=1}^{n}(x_i-\mu)^2$$

B. The score function will be calculated as($\theta = \lambda$):

$$\forall x \geq 0 : s_\lambda(x) = \nabla_\lambda \log p(x;\lambda) = \nabla_\lambda \log(\lambda e^{-\lambda x}) = \nabla_\lambda(\log(\lambda) - \lambda x) = \frac{1}{\lambda} - x$$

2.3 Properties of Score Functions

A. We obtain that (let $\chi = \text{supp}(p_\theta(.))$):

$$\mathbb{E}_{x\sim p_\theta(x)}[s_\theta(x)] = \mathbb{E}_{x\sim p_\theta(x)}[\nabla_\theta \log p_\theta(x)] = \int_\chi p_\theta(x)\nabla_\theta \log p_\theta(x)\, dx =$$

$$\int_\chi p_\theta(x)\frac{\nabla_\theta\{p_\theta(x)\}}{p_\theta(x)}\, dx = \int_\chi \nabla_\theta\{p_\theta(x)\}\, dx = \nabla_\theta \int_\chi p_\theta(x)\, dx = \nabla_\theta\{1\} = \vec{0}$$

B. Let us define the Fischer-Information as: $I(\theta) = \mathbb{E}[\|\mathbf{s}_\theta(\mathbf{x})\|^2] = \text{Var}_{p_\theta(.)}[s_\theta(x)] \geq 0$ since $\mathbb{E}_{x\sim p_\theta(x)}[s_\theta(x)] = \vec{0}$ and also $\mathbb{E}_{x\sim p_\theta(x)}[s_\theta(x)^T s_\theta(x)] \geq 0$, which proves:

$$\forall \theta \in \Theta : I(\theta) \geq 0$$

2.4 Application of Score Functions in Statistical Inference

A. Both loss functions aim to align the model's score function $s_\theta(x)$ with the data score function $s_{\text{data}}(x)$. The explicit loss is defined as the L_2 norm between the two score functions:

$$L_{\text{explicit}}(\theta) = \mathbb{E}_{p_{\text{data}}(x)}\left[\|\nabla_x \log p_\theta(x) - \nabla_x \log p_{\text{data}}(x)\|^2\right].$$

This loss requires access to the data score function, which is generally intractable. An alternative approach is the implicit loss function, which avoids direct dependency on the data score function:

$$L_{\text{implicit}}(\theta) = \mathbb{E}_{p_{\text{data}}(x)}\left[\|\nabla_x \log p_\theta(x)\|^2 + 2\,\text{tr}(\nabla_x^2 \log p_\theta(x))\right].$$

It has been shown that this loss function is equivalent to the explicit loss up to a constant C:

$$L_{\text{implicit}}(\theta) = C + L_{\text{explicit}}(\theta).$$

While the implicit loss is computationally more feasible, computing the trace of the Hessian remains expensive.

In both cases, optimizing over μ by setting $\nabla_\mu = 0$ leads to the following result:

$$\nabla_\mu L_{\text{explicit}}(\mu) = \nabla_\mu L_{\text{implicit}}(\mu) = \frac{2\mu - 2\mathbb{E}[x]}{\sigma^4} = 0 \quad \Rightarrow \quad \hat{\mu} = \mathbb{E}[x].$$

Thus, both loss functions yield the sample mean as the optimal solution.

The score test is used to determine the parameters of a probability distribution, with the null hypothesis given by $H_0 : \theta = \theta_0$. The test statistic is formulated as:

$$S(\theta_0) = \frac{s(\theta_0; D)^2}{I(\theta_0)}.$$

This test is also known as the "Lagrange Multiplier" test because it originates from constrained optimization. When maximizing the likelihood under H_0, a Lagrange multiplier is introduced, which corresponds to the score function. The test evaluates whether this multiplier, and consequently the score, is statistically zero.

The likelihood ratio test (LRT) compares the log-likelihoods under the null and alternative hypotheses. Its test statistic is given by:

$$\Lambda = -2(l(\theta_0) - l(\hat{\theta})).$$

Under standard regularity conditions and for large sample sizes, both tests are asymptotically equivalent and follow a chi-squared distribution. This equivalence emerges because the LRT statistic can be approximated using a Taylor expansion of $l(\hat{\theta})$ around θ_0:

$$\Lambda \approx -2\left(l(\theta_0) - l(\theta_0) - \delta\theta\nabla_\theta l(\theta_0) - \frac{1}{2}\frac{s(\theta_0; D)}{I(\theta_0)}\delta\theta^2\right) \equiv S(\theta_0).$$

Thus, the score test and the likelihood ratio test share a fundamental connection in their asymptotic properties.

B. Asymptotic normality refers to a key property of estimators, where the sampling distribution of an estimator approaches a normal distribution as the sample size increases indefinitely. This property is central to large-sample statistical inference, particularly for estimating population parameters. The maximum likelihood estimator (MLE), denoted as $\hat{\theta}_{\text{MLE}}$, satisfies the equation $s(\hat{\theta}_{\text{MLE}}; X) = 0$. Expanding the score function $s(\hat{\theta}_{\text{MLE}}; X)$ around the true parameter θ_0 yields:

$$0 = s(\theta_0; X) + \nabla_\theta s(\theta_0; X)(\hat{\theta}_{\text{MLE}} - \theta_0) + O((\hat{\theta}_{\text{MLE}} - \theta_0)^2).$$

Neglecting the higher-order terms, which is justified asymptotically, leads to the approximation:

$$\hat{\theta}_{\text{MLE}} - \theta_0 \approx -[\nabla_\theta s(\theta_0; X)]^{-1} s(\theta_0; X).$$

The term $\nabla_\theta s(\theta_0; X)$ corresponds to the negative observed Fisher information:

$$-\nabla_\theta s(\theta_0; X) = -\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta^2 \log f(X_i; \theta_0).$$

By the law of large numbers, this term converges in probability to the Fisher information:

$$-\nabla_\theta s(\theta_0; X) \xrightarrow{p} I(\theta_0).$$

The score function $s(\theta_0; X)$ is a sum of independent and identically distributed (i.i.d.) terms:

$$s(\theta_0; X) = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \log f(X_i; \theta_0).$$

By the central limit theorem,

$$\sqrt{n} s(\theta_0; X) \xrightarrow{d} N(0, I(\theta_0)),$$

since $\mathbb{E}[s(\theta_0; X)] = 0$ and $\text{Var}(s(\theta_0; X)) = \frac{1}{n} I(\theta_0)$. Consequently,

$$\sqrt{n}(\hat{\theta}_{\text{MLE}} - \theta_0) \approx I(\theta_0)^{-1} \cdot \sqrt{n} s(\theta_0; X) \xrightarrow{d} N\left(0, I(\theta_0)^{-1}\right),$$

and it follows that:

$$\hat{\theta}_{\text{MLE}} \xrightarrow{d} N\left(\theta_0, \frac{1}{n} I(\theta_0)^{-1}\right).$$

This result establishes the asymptotic normality of the maximum likelihood estimator. First, we calculate the Fisher information for the mean of a Gaussian distribution:

$$I(\mu) = \mathbb{E}\left[s^2(\mu; X)\right] = \mathbb{E}\left[\left(\frac{x-\mu}{\sigma^2}\right)^2\right] = \frac{1}{\sigma^2}$$

Using this result, we observe that the maximum likelihood estimator (MLE) of μ asymptotically follows a normal distribution:

$$\hat{\mu}_{\text{MLE}} \xrightarrow{d} \mathcal{N}(\mu, \sigma^2) \Rightarrow \hat{\mu}_{\text{MLE}} - \mu \xrightarrow{d} \mathcal{N}(0, \sigma^2).$$

Thus, a confidence interval with significance level α can be expressed as:

$$\mu = \hat{\mu} \pm z_{\alpha/2}\sigma$$

For a sample of size $n > 1$, we adjust the variance by replacing σ^2 with σ^2/n, leading to:

$$\mu = \hat{\mu} \pm z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$$

3 Problem 3

3.1 Theoretical Proof of Equivalence between ISM and ESM

A. Given a generative model with a parameterized density $p_\theta(x)$, where θ represents the model parameters, the loss functions are defined as follows:

The explicit score matching objective involves the gradient of the log-likelihood of the model's distribution with respect to the data. The loss for explicit score matching is given by:

$$L_{\text{explicit}}(\theta) = \mathbb{E}_{p_{\text{data}}(x)}\left[\|\nabla_x \log p_\theta(x) - \nabla_x \log p_{\text{data}}(x)\|^2\right]$$

On the other hand, the implicit score matching objective involves the gradient of the model's score function. The loss for implicit score matching is:

$$\mathcal{L}_{\text{explicit}}(\theta) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\|\nabla_x \log p_\theta(x) - \nabla_x \log p_{\text{data}}(x)\|^2\right] = \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\left\|\frac{(x-\mu)}{\sigma^2} - \nabla_x \log p_{\text{data}}(x)\right\|^2\right]$$

B. Expanding $L_{\text{explicit}}(\theta)$, we arrive at:

$$L_{\text{explicit}}(\theta) = \mathbb{E}_{p_{\text{data}}(x)} \left[\|\nabla_x \log p_\theta(x) - \nabla_x \log p_{\text{data}}(x)\|^2 \right]$$

Expanding the squared norm:

$$= \mathbb{E}_{p_{\text{data}}(x)} \left[\|\nabla_x \log p_\theta(x)\|^2 - 2(\nabla_x \log p_\theta(x)) \cdot (\nabla_x \log p_{\text{data}}(x)) + \|\nabla_x \log p_{\text{data}}(x)\|^2 \right]$$

Rearranging terms:

$$= \mathbb{E}_{p_{\text{data}}(x)} \left[\|\nabla_x \log p_\theta(x)\|^2 - 2(\nabla_x \log p_\theta(x)) \cdot (\nabla_x \log p_{\text{data}}(x)) \right] + C$$

Using integration by parts:

$$\mathbb{E}\left[\nabla_x \log p_\theta(x) \cdot \nabla_x \log p_{\text{data}}(x)\right] = -\mathbb{E}\left[\nabla_x \cdot (\nabla_x \log p_\theta)\right] = -\mathbb{E}\left[\text{tr}(\nabla_x^2 \log p_\theta(x))\right]$$

Thus, we obtain:

$$L_{\text{explicit}}(\theta) = \mathbb{E}_{p_{\text{data}}(x)} \left[\|\nabla_x \log p_\theta(x)\|^2 + 2\text{tr}(\nabla_x^2 \log p_\theta(x)) \right] + C$$

$$= L_{\text{implicit}}(\theta) + C$$

3.2

A. Let $p_\theta(x) = \mathcal{N}(x; \mu, \sigma) \rightarrow \log p_\theta(x) = -\frac{1}{2}\log(2\pi\sigma^2) - \frac{(x-\mu)^2}{2\sigma^2}$. Then rewriting implicit and explicit score matching implies that:

$$\mathcal{L}_{\text{explicit}}(\theta) = \mathbb{E}_{x\sim p_{\text{data}}(x)} \left[\|\nabla_x \log p_\theta(x) - \nabla_x \log p_{\text{data}}(x)\|^2 \right] = \mathbb{E}_{x\sim p_{\text{data}}(X)} \left[\frac{1}{2} \left(\frac{x-\mu}{\sigma^2} \right)^2 + s_{\text{data}}(x) \frac{x-\mu}{\sigma^2} \right] + \text{const}$$

and also:

$$\mathcal{L}_{\text{implicit}}(\theta) = \mathbb{E}_{x\sim p_{\text{data}}(x)} \left[\|\nabla_x \log p_\theta(x)\|^2 \right] = \mathbb{E}_{x\sim p_{\text{data}}(X)} \left[\frac{1}{2} \left(\frac{x-\mu}{\sigma^2} \right)^2 - \frac{1}{\sigma^2} \right]$$

B. We take the partial derivatives with respect to parameters:

$$\frac{\partial \mathcal{L}_{\text{explicit}}(\theta)}{\partial \mu} = \mathbb{E}_{x\sim p_{\text{data}}(X)} \left[\frac{\mu - x}{\sigma^2} - \frac{s_{\text{data}}(x)}{\sigma^2} \right] = 0$$

Considering the expectation of the data score function:

$$\mathbb{E}_{x\sim p_{\text{data}}(X)} \left[s_{\text{data}}(x) \right] = \int_{-\infty}^{+\infty} p_{\text{data}}(x) \frac{d \log p_{\text{data}}(x)}{dx} dx$$

$$= \int_{-\infty}^{+\infty} \frac{dp_{\text{data}}(x)}{dx} dx = p_{\text{data}}(x)|_{-\infty}^{+\infty} = 0$$

Thus, simplifying the partial derivative:

$$\frac{\partial \mathcal{L}_{\text{explicit}}(\theta)}{\partial \mu} = \mathbb{E}_{x\sim p_{\text{data}}(X)} \left[\frac{\mu - x}{\sigma^2} \right] = 0 \Longrightarrow \hat{\mu}_{\text{explicit}} = \mathbb{E}_{x\sim p_{\text{data}}(X)}[x]$$

Similarly, for the implicit case:

$$\frac{\partial \mathcal{L}_{\text{implicit}}(\theta)}{\partial \mu} = \mathbb{E}_{x\sim p_{\text{data}}(X)} \left[\frac{\mu - x}{\sigma^2} \right] = 0 \Longrightarrow \hat{\mu}_{\text{implicit}} = \mathbb{E}_{x\sim p_{\text{data}}(X)}[x]$$

This shows that:

$$\hat{\mu}_{\text{explicit}} = \hat{\mu}_{\text{implicit}}$$

4 Problem 4

Starting from the forward diffusion process and the properties of additive Gaussian processes, we know that the forward process of DDPM can be written as follows, with $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$:

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

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

Now, using Bayes' rule, we aim to find $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. Since $q(\mathbf{x}_t, \mathbf{x}_{t-1}|\mathbf{x}_0)$ is a Gaussian distribution, we can write:

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

Substituting the expressions for each term:

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

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

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

we compute:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{\mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I}) \cdot \mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0, (1-\bar{\alpha}_{t-1})\mathbf{I})}{\mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})}.$$

Expanding the Gaussian terms and simplifying:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_q(\mathbf{x}_t, \mathbf{x}_0, t), \Sigma_q(t)),$$

where:

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

$$\Sigma_q(t) = \frac{1-\bar{\alpha}_t}{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}\mathbf{I}.$$

This derivation was based on this fact that if we have three gaussians $X_i \sim p_i(x) = \mathcal{N}(x|\mu_i, \sigma_i^2 I)$, then $f(X) = \frac{p_1(x)p_2(x)}{p_3(x)}$ is a valid gaussian pdf with variance $\frac{\sigma_1^2\sigma_2^2}{\sigma_3^2}$ and mean $\frac{\sigma_1^2\sigma_2^2}{\sigma_3^2}(\frac{\mu_1}{\sigma_1^2} + \frac{\mu_2}{\sigma_2^2} + \frac{\mu_3}{\sigma_3^2})$.

The mean $\mu_q(\mathbf{x}_t, \mathbf{x}_0, t)$ linearly interpolates between $\mathbf{x}_t$ (the current noisy sample) and $\mathbf{x}_0$ (the original data), weighted by terms dependent on the noise schedule. The covariance $\Sigma_q(t)$ balances the noise added at step t and the cumulative noise up to step $t-1$.

This parameterization ensures tractability in the reverse process, enabling efficient sampling by gradually denoising $\mathbf{x}_t$ toward $\mathbf{x}_0$. The probability flow viewpoint of diffusion models provides an intuitive understanding of this process.