Assignment 2: *Explicit Methods for Generation*

Date: *June 16, 2025* Due : *July 7, 2025*

Preface

This is the second series of assignments for the course *ECE1508: Deep Generative Models*. The exercises are aimed to review the explicit methods for generation studied in Chapters 2 and 3. Please adhere to **Code of Honor** outlined on the course page. Below, you can find the information about the contents, as well as instructions on how to submit them. Please read them *carefully*.

General Information The assignments are given in two sections. In the first section, you have written questions that can be answered in words or by derivation. The questions are consistent with the material of Chapter 1 and you do not need any further resources to answer them. The second section includes the programming assignments. In the case that a question is unclear or there is any flaws, please contact over Piazza. Also, in case that any particular assumption is required to solve the problem, feel free to consider the assumption and state it in your solution. The total mark of the assignments is **100 points** with total mark of written questions adds to **50 points** and the total mark of the programming assignments adds to **50 points**.

How to Submit A notebook has been provided with starter code that you can complete. The submission is carried out through the Crowdmark. Please submit the answer of each question **separately**, following the**steps below**. Please note that *failure to meet the formatting can lead to mark deduction*.

1. For Written Questions, you can submit handwritten or typed answers as a `.pdf`, `.jpg` or `.png` file.
2. For Programming Questions, the answer should **complete the Python Notebook** shared with this assignment. For *each question*, please print the corresponding part of the notebook as a `.pdf` file and upload it in the corresponding field on Crowdmark. Your uploaded `.pdf` should contain **all figures, diagrams and codes requested** in the question.
3. The completed Notebook, i.e., the `.ipynb` file, including all the codes and the outputs should also be submitted as the attachment to the last item on Crowdmark.

When submitting your notebook, please pay attention to the following points:

1. The file should be renamed as `<YourLastname>_<YourFirstname>_Assgn2.ipynb`
2. Please make sure to name the files with the name that is displayed on the Quercus account

The deadline for your submission is on **July 7, 2025 at 11:59 PM**.

- You can delay up to two days, i.e., until **July 9, 2025 at 11:59 PM**. After this extended deadline no submission is accepted.
- For each day of delay, 10% of the mark after grading is deducted.

Please submit your assignment **only through Crowdmark, and not by email.**

1 Written Exercises

Question 1 [15 Points] **(Discriminative and Generative Learning with Latent)** Consider a classification task in which data sample $x \in \mathbb{R}^d$ is to be classified by label y. For this data, we have access to a *pretrained decoder* which transforms the latent representation $z \in \mathbb{R}^m$ to its corresponding data sample $x \in \mathbb{R}^d$

$$F_\theta : \mathbb{R}^m \mapsto \mathbb{R}^d.$$

Note that we have *only access to this decoder* and *do **not** know the encoding transform.* This means that for a given x, we *cannot* compute the latent representation directly.

In this question, we want to study how we can use this decoder to develop a latent-based discriminative and generative model. For sake of simplicity, assume that x, z and y are all discrete, i.e., each entry x_i for $i = 1, \dots, d$ and z_ℓ for $\ell = 1, \dots, m$ as well as y are all coming from sets with finite possible cases.

1. Let $P_{\mathbf{w}}(y|z)$ be a discriminative model that classify a data sample from its latent representation. We assume that the latent is distributed with some known $P(z)$. Derive a discriminative model $Q_{\mathbf{w},\theta}(y|x)$ that uses Bayes' rule to compute the label distribution for a given sample x from the model $P_{\mathbf{w}}(y|z)$ and encoder $F_\theta(z)$.

2. Write an algorithm (similar to those we have in lecture notes) that explains how we can train the discriminative model $P_{\mathbf{w}}(y|z)$ through the above formulation using a labeled dataset, i.e.,

$$\mathbb{D} = \left\{ \left(x^j, y^j\right) \;\; \textit{for} \;\; j = 1, \dots, n \right\}.$$

3. Compare the computational complexity of training this model to the training of a basic discriminative model $\tilde{P}_{\mathbf{w}}(y|x)$ that learns the class directly in the data domain.

4. Now, let $P_{\mathbf{w}}(z|y)$ be a label generative model that describes the distribution of the latent representation for a given label y. Using Bayes' rule, develop a model $Q'_{\mathbf{w},\theta}(y|x)$ which deploys the generative model $P_{\mathbf{w}}(z|y)$ and encoder $F_\theta(z)$ to classify sample x.

5. Write the training loop as an algorithm assuming a labeled dataset, i.e.,

$$\mathbb{D} = \left\{ \left(x^j, y^j\right) \;\; \textit{for} \;\; j = 1, \dots, n \right\}.$$

6. Do you think that the latter model $Q'_{\mathbf{w},\theta}(y|x)$ developed based on a generative model can have any benefit as compared to the discriminative learning by the model $Q_{\mathbf{w},\theta}(y|x)$ in Part 1? Explain your answer.

Question 2 [10 Points] **(Changing Flow Direction)** We want to re-do whatever we did in the lecture for flow-based models with only one simple change: we define the invertible flow model to be the one *which computes the latent representation*, i.e., we consider a flow model $f_{\mathbf{w}} : \mathbb{R}^d \mapsto \mathbb{R}^d$ which for a given data sample x computes the latent representation as

$$z = f_{\mathbf{w}}(x).$$

Considering this definition, the *inverse flow* is defined as the mapping that takes the latent representation back to the data domain, i.e.,

$$x = f_{\mathbf{w}}^{-1}(z).$$

This formulation is in many implementation used, although it makes no computational or theoretical difference. In this exercise, we practice what we did in the lecture for this alternative representation.

1. Assume that x and z are scalar, i.e., $x, z \in \mathbb{R}$, and that $f_{\mathbf{w}}$ is strictly increasing. Let $z \sim Q(z)$. Derive the data distribution $P_{\mathbf{w}}(x)$ in terms of $Q(z)$ and $F_{\mathbf{w}}$.

2. Without any derivation, extend the formula derived in Part 1 to the multi-dimensional case, i.e., when $x, z \in \mathbb{R}^d$.

3. Assume that we have the dataset

$$\mathbb{D} = \left\{ x^j \ \textit{for} \ j = 1, \ldots, n \right\}.$$

   Using the maximum likelihood method, derive an empirical risk for training of the flow model on this dataset.

4. Compare your derivations with what we had in the lectures. What is the key change?

QUESTION 3 [15 Points] **(Jensen-Shannon Divergence)** The Jensen-Shannon (JS) divergence between two distributions P and Q is defined in terms of KL divergence as

$$D_{\mathrm{JD}}(P \| Q) = \frac{1}{2} D_{\mathrm{KL}}(P \| M) + \frac{1}{2} D_{\mathrm{KL}}(Q \| M)$$

where M is the so-called *average distribution*, which at any point x is defined as

$$M(x) = \frac{Q(x) + P(x)}{2}.$$

In this assignment, we want to work a bit with this divergence metric to get some feelings about it.

1. Show that M satisfies both properties of a distribution and hence it is a distribution.

2. Can this divergence metric be negative?

3. Show that JS divergence is symmetric, i.e.,

$$D_{\mathrm{JD}}(P \| Q) = D_{\mathrm{JD}}(Q \| P).$$

4. Assume that P and Q are two binary distributions defined as

$$P(x) = \begin{cases} \theta & x = 1 \\ 1 - \theta & x = 0 \end{cases} \qquad\qquad Q(x) = \begin{cases} 0.3 & x = 1 \\ 0.7 & x = 0 \end{cases}.$$

   Compute both the KL and JS divergence between P and Q as a function of θ.

5. Plot both divergences against $\theta \in (0, 1)$, and find $\theta^\star$ at which both divergence metrics are minimized.

6. Assume P and Q are both defined of data-space $\mathbb{X}$ and $P(x) \neq 0$ $Q(x) \neq 0$ for all $x \in \mathbb{X}$. Show that JS divergence is zero *if and only if* KL divergence is zero.

QUESTION 4 [10 Points] **(Fréchet Inception Distance)** In many visual applications, one of the key evaluation metrics is the so-called Fréchet inception distance (FID) [2]. In this question, we get to know this metric and learn how we can compute it in practice.

The Fréchet distance between two distributions is in general complicated; see `https://en.wikipedia.org/wiki/Fr%C3%A9chet_distance`. However when both distributions are Gaussian, we can compute it readily: let the two distributions be $\mathcal{N}(\mu_1, \Sigma_1)$ and $\mathcal{N}(\mu_2, \Sigma_2)$ with $\mu_i \in \mathbb{R}^d$ being the mean vector and

$\Sigma_i \in \mathbb{R}^{d\times d}$ being the covariance matrix for $i = 1, 2$. The Fréchet distance between these two Gaussian distributions is

$$D_{\mathrm{F}} = \|\mu_1 - \mu_2\|^2 + \mathrm{tr}\{\Sigma_1 + \Sigma_2 - 2\,(\Sigma_1\Sigma_2)^{1/2}\}.$$

In the ideal case, we would like to estimate the Fréchet distance between the learned distribution and the true one. Nevertheless, given that this is a computationally-expensive, we compute the Fréchet distance between their Gaussian-distributed latent representations. The process is in a nutshell as follows: we use a pretrained model F_θ to map the data sample x to a latent representation $z = F_\theta(x)$. This representation is assumed to be Gaussian. Invoking F_θ, we transform both true data distribution and learned one into Gaussian latent distributions and compute the Fréchet distance between them.

In the sequel, we go step by step through the process of computing FID. To this end, assume that we have a trained encoding model F_θ, which transforms $x \sim P(x)$ into a latent representation $z \sim \mathcal{N}(\mu, \Sigma)$ with specific mean μ and covariance Σ, i.e., as P changes, μ and Σ change as well.

1. Use a test set of n true data samples drawn i.i.d. from data distribution P, i.e.,

$$\mathbb{D} = \left\{x^j \ \textit{for}\ \ j = 1, \dots, n\right\},$$

   and F_θ to *estimate* mean and covariance of $z = F_\theta(x)$ when $x \sim P$.

2. Let $P_{\mathbf{w}}$ denote our *trained* generative model. Explain how we can estimate the mean and covariance for $\hat{z} = F_\theta(\hat{x})$ with $\hat{x} \sim P_{\mathbf{w}}$.
   Hint: *For both Parts 1 and 2, we can use law of large numbers.*

3. Using the estimates for the mean and variance of the latent representations z and $\hat{z}$, write down the expression for the FID, which is defined to be the Fréchet distance between $P(z)$ and $\hat{P}(\hat{z})$.

4. Take a look at the evaluations in [2] to find out what kind of model is used for latent encoding, i.e., F_θ, and how large n is.

2 Programming Exercises

Question 1 [25 Points] **(Image Inpainting by Masked CNN)** In this question, we implement a basic masked CNN (similar to PixelCNN) for image inpainting. The image inpainting refers to the generation of missed pixels in an image, which is considered as one of the key applications of visual generative model; see [3] for instance.

1. Complete Question 1 in `Lastname_Firstname_Asgn2.ipynb` to implement a masked CNN model for autoregressive generation.

2. Answer all parts marked by `# COMPLETE`.

Question 2 [25 Points] **(Real NVP Implementation)** In this question, we implement a basic form of Real NVP with fully-connected scaling and translation models for MNIST image generation. You can learn more on Real NVP in [1], though what we had in the lecture is enough to complete this question.

1. Complete Question 2 in `Lastname_Firstname_Asgn2.ipynb`.

2. Answer all parts marked by `# COMPLETE`.

References

[1] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations (ICLR)*, 2017.

[2] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017. Available at `https://arxiv.org/abs/1706.08500`.

[3] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018.