CS 598-DGM: Spring'25
Deep Generative Models

Homework 2

(Due Monday, March 31, 11:59 pm)

- The homework is due at 11:59 pm on the due date. We will be using Gradescope for the homework assignments. **Please do NOT email a copy of your solution.** Contact the TAs (Zhijie, Rohan) if you are having technical difficulties in submitting the assignment. We will NOT accept late submissions.
- Please make sure that each question is clearly marked. You may use as many pages as needed but do not change the order of the questions and answers.
- You are expected to typeset the solutions, i.e., **handwritten solutions will not be graded**. We encourage you to use LaTeX. **When submitting on Gradescope, you are required to assign the correct pages for each sub-problem to the provided outline. If pages are incorrectly assigned or left unassigned on Gradescope, it will result in no credit, and regrade requests regarding this will be declined.** Double-check your submission to ensure accuracy.
- Please use Slack first if you have questions about the homework. You can also come to our (zoom) office hours and/or send us e-mails. If you are sending us emails with questions on the homework, please start subject with "CS 598-DGM: " and send the email to *all course staff*: Arindam, Zhijie, and Rohan.
- The homework consists of both written assignments and programming assignments. Please submit your report as a PDF file and your code/model following the below instruction.

  Programming Assignment Instructions

  – All programming needs to be in Python 3.
  – The homework will be graded using Gradescope. You will be able to submit your code as many times as you want.
  – We provided a starter Jupyter Notebook on Canvas. Please finish the programming assignment based on it.
  – Please do not change the seed in the starter code.
  – For submitting on Gradescope, you need to upload four files: a Jupyter Notebook and three model checkpoints. Please make sure they are named in this way:
    1. "hw2.ipynb"
    2. "model_nice_2025.pt"
    3. "model_wgan_2025.pt"
    4. "model_ddpm_2025.pt"
  – Please write your code entirely by yourself.

1. (15 points) Recall that the Noise Contrastive Estimate (NCE) depends on the choice of the noise distribution p_n and the ratio $\nu = T_n/T_d$ between T_n, the number of samples from the noise distribution p_n, and T_d, the number of samples from the data distribution p_d.

   (a) (6 points) For $\nu \to \infty$, clearly explain why the asymptotic distribution of the NCE $\hat{\theta}_T$ is independent of p_n, the noise distribution.[1]

   (b) (9 points) For normalized models and for $\nu \to \infty$, NCE is known to be asymptotically "Fisher-efficient."

      i. (3 points) When is an estimate called Fisher-efficient?

      ii. (3 points) Why is being Fisher-efficient a desirable property?

      iii. (3 points) Why is NCE Fisher-efficient?

   Clearly explain your answer.[2]

[1] The result holds given certain technical assumptions. You should acknowledge the assumptions in your answer.

[2] If you are using external sources for your answer, please cite the sources.

2. (20 points) We consider problems in the context of basic sampling based on rejection sampling and importance sampling.

   (a) (6 points) Consider rejection sampling from the distribution with density $p(x) = \frac{f(x)}{Z_f}, x \in [0, 1]$, where for given $\alpha, \beta \geq 1$

   $$f(x) = x^{\alpha-1}(1-x)^{\beta-1} \tag{1}$$

   using the proposal distribution with density $u(x) = 1$ if $x \in [0, 1]$, and 0 otherwise.

   - (2 points) Show that you can do such rejection sampling using the unnormalized $f(x)$ with $M = 1$.[3]
   - (4 points) What is the expected acceptance rate for the proposed rejection sampling algorithm? Clearly show the derivation.[4]

   (b) (6 points) Consider rejection sampling from the uniform distribution on the unit ball in $\mathbb{R}^d$, i.e., $x = (x_1, x_2, \ldots, x_d)$ with unnormalized distribution $f(x) = 1$ if $x \in B_2^d$, where $B_2^d = \{x \in \mathbb{R}^d : ||x||_2^2 = \sum_{i=1}^d x_i^2 \leq 1\}$ using the unnormalized distribution $g(x)$ which is product of coordinate-wise uniform functions, i.e.,

   $$\begin{aligned} g(x_1, x_2, \ldots, x_d) &= \prod_{i=1}^{d} g_i(x_i) \ , \\ \text{where} \quad g_i(x_i) &= \begin{cases} 1 \ , & \text{if } x_i \in [-1, 1] \\ 0 \ , & \text{otherwise} \ . \end{cases} \end{aligned} \tag{2}$$

   Let $p(x) = \frac{f(x)}{Z_f}$ and $q(x) = \frac{g(x)}{Z_g}$ be the corresponding normalized distributions for $f(x)$ and $g(x)$.

   - (2 points) Show that you can do such rejection sampling using the unnormalized $f(x)$ and $g(x)$ with $M = 1$.
   - (4 points) What is the expected acceptance rate for the proposed rejection sampling algorithm? Clearly show the derivation.[5] You can use order notation to avoid constants, but the dependence on d needs to be clearly stated.

   (c) (8 points) Recall that computation of the expectation $\mathbb{E}_{x\sim p}[H(x)]$ can be done using importance sampling with samples $x_i \sim q(x), i \in [n]$ using

   $$\hat{H}_n = \frac{\sum_{i=1}^n H(x_i)\frac{f(x_i)}{g(x_i)}}{\sum_{i=1}^n \frac{f(x_i)}{g(x_i)}} \tag{3}$$

   where $p(x) = \frac{f(x)}{Z_f}$ and $q(x) = \frac{g(x)}{Z_g}$ with known $f(x), g(x)$ but unknown Z_f, Z_g. As $n \to \infty$, shown that $\hat{H}_n \to \mathbb{E}_{x\sim p}[H(x)]$.

[3] Note that since $u(x)$ is a uniform distribution on $[0, 1]$, it is automatically normalized.

[4] You can use standard special functions, refer to the source (e.g., wikipedia) if you use such functions.

[5] You can use standard special functions, refer to the source (e.g., wikipedia) if you use such functions.

3. (20 points) Consider sampling from $p(x) = \frac{e^{-f(x)}}{Z_f}$ with a known $f(x)$ and unknown Z_f using variants of the Metropolis-Hastings (MH) algorithm with Gaussian transition density with variance $\sigma^2\mathbb{I}$, e.g., $T(x|x') = \mathcal{N}(x', \sigma^2\mathbb{I})$, as $\sigma \to 0$:

   (a) (10 points) Show that the rejection rate of the Metropolis algorithm with a Gaussian proposal distribution $T(x|x') = \mathcal{N}(x', \sigma^2\mathbb{I})$ is $O(\sigma)$ as $\sigma \to 0$.

   (b) (10 points) Show that the rejection rate of the Metropolis Adjusted Langevin Algorithm with a Gaussian proposal distribution $T(x|x') = \mathcal{N}(x' - \eta\nabla f(x'), 2\eta\mathbb{I})$ with step size $\eta = \frac{1}{2}\sigma^2$ is $O(\sigma^2)$ as $\sigma \to 0$.

4. (45 points) The programming assignment will focus on developing your own code for: Non-linear Independent Component Analysis (NICE), Wasserstein GAN (WGAN), and Denoising Diffusion Probabilistic Model (DDPM).

   Dataset: All three models will be evaluated on the 2D Two Moon dataset. We have provided code to load the dataset(s) in the starter Jupyter notebook. The assignment will use three specific variants of the 2D Two Moon dataset: a training set of 5000 points and two validation sets, each with 1000 points, all generated from the 2D Two Moon dataset generator.

   Generative Models. We will consider three different generative models

   - (15 points) Non-linear Independent Component Analysis (NICE): You will be implementing a NICE model based on the objective as in (Dinh et al., 2015). You need to do the following:
     - Implement the NICE using a logistic (not Gaussian) distribution prior with the 4 coupling layers as specified in Section 5.1 of the paper. The network in the coupling function is modeled as a feedforward neural network with 3 hidden layers and each with 32 units.
     - Implement a sampler from the NICE model based a samples from an isotropic Gaussian distribution $z \sim \mathcal{N}(0, \mathbb{I})$ and the trained NICE model.

     In your written report, you need to do the following:
     - Sample 1000 points from learned model, and visulaize the samples.
     - Compute the average log-likelihood of the two provided validation datasets and report the results.
     - Compute the Wasserstein distance between the 1000 sampled points and the 1000 points in each of the two provided validation datasets, and report the results.
   - (15 points) Wasserstein GAN (WGAN): You will be implementing a WGAN model, as in (Gulrajani, et al., 2017). You need to do the following:
     - Implement a training algorithm for WGAN with gradient penalty. The coefficient for the penalty term should be 0.1. Run 5 critic iterations per generator iteration.
     - Implement a sampler from the WGAN model from an isotropic Gaussian distribution $z \sim \mathcal{N}(0, \mathbb{I})$.

     In your written report, you need to do the following:
     - Sample 1000 points from learned WGAN model and visualize the samples.
     - Compute the Wasserstein distance between the 1000 sampled points and the 1000 points in each of the two provided validation datasets, and report the results.
   - (15 points) Denoising Diffusion Probabilistic Model (DDPM): You will be implementing a DDPM model , as in (Ho, et al., 2020). You need to do the following:
     - Implement a training algorithm for DDPM using the feed-forward architecture provided in the starter code.
     - Implement a sampler from the DDPM model based a samples from an isotropic Gaussian distribution $z \sim \mathcal{N}(0, \mathbb{I})$ and the trained DDPM model.

     In your written report, you need to do the following:
     - Sample 1000 points from the DDPM model and visualize the samples.

- Compute the Wasserstein distance between the 1000 sampled points and the 1000 points in each of the two provided validation datasets, and report the results.

Training. You can train these three models on your local machine or a resource like Colab (use of GPU is not necessary). We have fixed most hyper-parameters like batch size, learning rate, etc., except the ones we let you investigate. We provided the architectures and the function to compute the Wasserstein distance in the starter code. You need to save the checkpoints (we have provided required code) and upload them along with your Jupyter Notebook.