MIT CSAIL

6.s978 Deep Generative Models

Fall 2024

Problem Set 4

Posted: Tuesday, Oct. 22, 2024 **Due:** Tuesday 11:59 pm, Nov. 05, 2024

We provide a Python notebook with the code to be completed. You can run it locally or on Google Colab. To use Colab, upload it to Google Drive and double-click the notebook (or right-click and select Open with Google Colaboratory), which will allow you to complete the problems without setting up your own environment.

Submission Instructions: Please submit two files on Canvas: Submit (1) Your report named `kerberos.pdf` which should include your answers to all required questions with images/plots showing your results as well as the code you wrote (e.g., using screenshots); (2) The provided Python notebook with the relevant source code and with all the cells run.

Attention: If you fail to include your code in your report in your submission, we will not be able to give you credit for your code, so please do not forget.

Late Submission Policy: If your Problem Set is submitted within 7 days of the original deadline, you will receive partial credit. Such submissions will be penalized by a multiplicative coefficient that linearly decreases from 1 to 0.5, step-wise on each day's 11:59pm cutoff.

In this problem set, you will be experimenting with GANs using PyTorch. To reduce training time, we recommend using GPU acceleration. Colab comes with free GPU support. On Colab, select GPU as your runtime type as follows:
Runtime → Change runtime type → Hardware accelerator → GPU → Save.

If you exceed your GPU usage limit on Colab, don't fret. You can also complete your problem set using a regular CPU in a reasonable amount of time.

Problem 0 *Notebook Submission, .ipynb version* (1 point, required)

Problem 1 *DDPM on MNIST* (40 points)

In this problem set, you will implement a Denoising Diffusion Probabilistic Model (DDPM) for the MNIST dataset. The DDPM is a generative model that progressively corrupts a data sample through a series of noise additions (forward process) and then learns to reverse this process to generate new samples (reverse process). This assignment will guide you through implementing key components of DDPM and training the model on MNIST.

The forward process in DDPM progressively adds Gaussian noise to an image over T timesteps.

The noise schedule is controlled by a series of variance values β_t, which determine how much noise is added at each timestep.

The key quantities for the forward process contain: β_t: the variance schedule for each timestep, and $\bar{\alpha}_t$: the cumulative product of α_t up to timestep t.

The mathematical formulation for these terms is as follows:

$$\beta_t = \beta_{\min} + t \cdot \frac{\beta_{\max} - \beta_{\min}}{T-1}, \quad t = 0, 1, 2, \ldots, T-1$$

where $\beta_{\min}$ and $\beta_{\max}$ are hyperparameters controlling the noise at the first and final timesteps.

We define α_t and $\bar{\alpha}_t$ as:

$$\alpha_t = 1 - \beta_t$$

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

(a) Implement the `ddpm_schedules()` function.

The forward process of DDPM gradually adds noise to the data at each timestep t. Given a data sample x_0 and a timestep t, the noisy version x_t can be computed as:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

Here, ϵ is Gaussian noise sampled at each timestep. You will need to use the precomputed values of $\bar{\alpha}_t$ from the `ddpm_schedules()` function.

(b) Implement `forward()` to return x_t given x_0 and timestep t.

Given $x_T \sim \mathcal{N}(0, I)$ (a pure Gaussian noise), the reverse process can be written as:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t z, \quad z \sim \mathcal{N}(0, I)$$

where σ_t is related to the variance at each step.

(c) Implement `sample()` to generate new samples by iteratively applying the reverse process starting from x_T.

(d) Train the DDPM model on the MNIST dataset using the code provided.

Problem 2 *Ablation Study of DDPM on MNIST* (30 points)

You will investigate the performance of DDPM on MNIST under varying conditions. Specifically, you will implement two modifications:

1. Vary the number of timesteps to 2000 and 500 and observe the effect on generation quality.
2. Implement a resampling schedule for inference with 50 and 100 steps instead of 1000-step used during training.

For each experiment, train the DDPM on the MNIST dataset, and evaluate the quality of the generated images. Compare the results of each experiment to the baseline DDPM trained in Problem 1.

Problem 3 *Continuous-time Stochastic Process* (40 points)

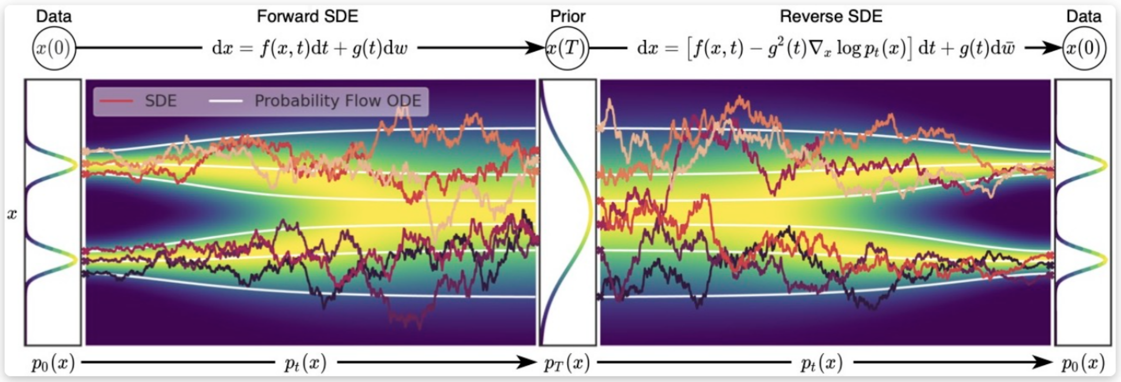
Data
Forward SDE
Prior
Reverse SDE
Data
$x(0)$
$\mathrm{d}x = f(x,t)\mathrm{d}t + g(t)\mathrm{d}w$
$x(T)$
$\mathrm{d}x = \left[f(x,t) - g^2(t)\nabla_x \log p_t(x)\right]\mathrm{d}t + g(t)\mathrm{d}\bar{w}$
$x(0)$
SDE
Probability Flow ODE
x
$p_0(x)$
$p_t(x)$
$p_T(x)$
$p_t(x)$
$p_0(x)$

Figure 1: Continuous-time Stochastic Process.

In this problem you will implement a toy model of continuous-time stochastic process, producing a figure similar to Figure 1. We will follow the notations in the paper of "Score-Based Generative Modeling through Stochastic Differential Equations" (`https://arxiv.org/pdf/2011.13456`). Read carefully the Appendix C about the SDE formulation of DDPM. Assume the data distribution $p(x(0)) = \sum_{i=1}^{2} \pi_i \mathcal{N}(x(0); \mu_i, \sigma_i),\ x \in \mathbb{R}$ is a mixture of two gaussians.

(a) Write the formula for $p(x(t)) = \int p(x(t)|x(0))p(x(0))dx_0$ in terms of t, $\beta_{\min}$, and $\beta_{\max}$.

(b) Calculate the analytical form of $\nabla_{x_t} \log p_t(x)$. Note that in practice this term is being learned by the neural network in the SDE formulation.

(c) Implement both functions in code and visualize the forward and reverse SDE.