MIT CSAIL

6.s978 Deep Generative Models

Fall 2024

Problem Set 5

Posted: Tuesday, Nov. 5, 2024 **Due:** Tuesday 11:59 pm, Nov. 12, 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)

All the notations in this assignment follow those in the paper of "Flow Matching for Generative Modeling" (`https://arxiv.org/pdf/2210.02747`).

Problem 1 *Conditional Flow Matching on MNIST* (40 points)

In this problem set, you will implement Conditional Flow Matching (CFM) for the MNIST dataset. CFM is a generative model that learns a conditional flow field. This assignment will guide you through implementing key components of CFM and training the model on MNIST.

In CFM, the model learns a time-dependent vector field that progressively transforms a noise sample into a data sample through a continuous flow. The conditional probability paths for this process are defined by a mean $\mu_t(x)$ and standard deviation $\sigma_t(x)$ that change linearly over time. We use Optimal Transport Conditional Vector Field (VF) in this problem.

The key quantities for the flow process are:

$$\mu_t(x) = tx_1, \quad \sigma_t(x) = 1 - (1 - \sigma_{\min})t,$$

where $\sigma_{\min}$ is a hyperparameter controlling the noise level at $t = 1$.

According to the flow matching approach, this conditional path is generated by the vector field:

$$u_t(x|x_1) = \frac{x_1 - (1 - \sigma_{\min})x}{1 - (1 - \sigma_{\min})t}.$$

The conditional flow that corresponds to $u_t(x|x_1)$ is given by:

$$\psi_t(x) = (1 - (1 - \sigma_{\min})t)x + tx_1.$$

In this case, the CFM loss is defined as:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,q(x_1),p(x_0)} \left\| v_t(\psi_t(x_0)) - (x_1 - (1 - \sigma_{\min})x_0) \right\|^2.$$

We use $\sigma_{\min} = 0$ in this assignment.

(a) Implement the function `compute_mu_t()`, which computes the mean $\mu_t(x)$.

(b) Implement the function `compute_sigma_t()`, which computes the standard deviation $\sigma_t(x)$.

(c) Implement the function `compute_conditional_flow()`, which calculates the conditional vector field $u_t(x|x_1)$.

(d) Implement the function `sample_xt()`, which draws a sample from the probability path $\mathcal{N}(\mu_t(x), \sigma_t(x))$.

(e) Train the CFM model on the MNIST dataset using the code provided.

Problem 2 *Ablation Study of Conditional Flow Matching on MNIST* (30 points)

In this problem, you will investigate the performance of CFM on MNIST under varying conditions. Specifically, you will implement modifications to the timestep schedule and the underlying Ordinary Differential Equation (ODE) solver. You will also evaluate the impact of replacing the Optimal Transport conditional VF in CFM with that of diffusion conditional VFs.

1. Change the ODE solver to two of the following methods and compare the performance:
    - `dopri8`: Runge-Kutta of order 8 (Dormand-Prince-Shampine)

- `bosh3`: Runge-Kutta of order 3 (Bogacki-Shampine)
- `fehlberg2`: Runge-Kutta-Fehlberg of order 2
- `adaptive_heun`: Runge-Kutta of order 2

2. Modify `compute_conditional_flow()` in CFM to use the formulation of Diffusion conditional VFs.

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

Problem 3 *2D Gaussian to Gaussian using a conditional flow* (30 points)

In this problem, you will implement a toy model of a continuous flow matching process that transforms a Gaussian distribution to another Gaussian distribution in 2D, producing a figure similar to Figure 11 and 12 in `https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html`.

Both the source $p(x_0) = \mathcal{N}(\mu_0, \sigma_0)$ and the target distribution $p(x_1) = \mathcal{N}(\mu_1, \sigma_1)$ is modeled as a 2D Gaussian with a specific mean and variance. We will define the conditional path from an initial sample x_0 to a final sample x_1, and derive the marginal flow for this Gaussian to Gaussian transformation.

The formula for the conditional paths that interpolate linearly between the initial sample x_0 and the final sample x_1 is a function of t:

$$x_t = (1-t)x_0 + tx_1$$

where $t \in [0,1]$.

(a) Derive the closed form of the marginal flow $u(x_t)$ describing how each point x evolves over time. Note that $u(x_t) = \int u(x_t|x_1)\frac{p_t(x|x_1)q_1(x_1)}{p_t(x)}dx_1$, following the notations in the paper of "Flow Matching for Generative Modeling".

(b) Implement the function `conditional_paths()` and `marginal_flow()` based on the derived formula.

(d) Visualize the conditional paths and marginal flow in 2D using the provided code.