MIT CSAIL

6.s978 Deep Generative Models

Fall 2024

Problem Set 1

Posted: Tuesday, Sep. 05, 2024 **Due:** Tuesday 11:59 pm, Sep. 24, 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 Gradescope: 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 Auto-Encoders (AEs) and Variational Auto-Encoders (VAEs) 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 *Auto-Encoder on MNIST* (20 points)

We will start by implementing a simple auto-encoder (AE) using PyTorch and training it on the MNIST dataset.

(a) Finish the implementation of `self.encoder` and `self.decoder` in `AE()` model.

(b) Train the AE model on the MNIST dataset (manual tuning of parameters such as `epochs`, `hidden_dims`, and `lr` may be necessary). Use the provided evaluation code to visualize the reconstruction results and the generated images (in 2D grid) for the last epoch.

In the following two problems, we explore the VAE framework and its application to the MNIST dataset.

Problem 2 *Variational Auto-Encoder on MNIST: Stoachastic Gradient Variational Bayes (SGVB) Estimator* (30 points)

We first focus on the Stochastic Gradient Variational Bayes (SGVB) Estimator, which is a technique used to train VAEs by maximizing the Evidence Lower Bound (ELBO) on the marginal log-likelihood of the data.

Given data x (binary images for MNIST) and latent variable z, ELBO can be expressed as

$$\log p(x) \geq \mathbb{E}_{q(z|x)}[\log \frac{p(x,z)}{q(z|x)}] = \text{ELBO}$$

(a) Give a detailed mathematical proof of the ELBO starting from the marginal log-likelihood $\log p(x)$. Show all steps clearly.

In practice, the expectation $\mathbb{E}_{q(z|x)}[\cdot]$ is estimated using Monte Carlo sampling, yielding the SGVB estimator:

$$\text{ELBO} \approx \sum_{i,j} [\log p(x_i|z_{i,j}) + \log p(z_{i,j}) - \log q(z_{i,j}|x_i)],$$

where $z_{i,j}$ is sampled from $q(z|x_i) = \mathcal{N}(z; \mu_i, \sigma_i^2 \mathbf{I})$. In this assignment, by default, we only sample one $z_{i,j}$ for each x_i (see the function `reparameterize()` in the `VAE()` class).

(b) Complete the implementation of the `self.encoder` and `self.decoder` in the `VAE()` model.

(c) Implement the reparameterization trick in the `reparameterize()` function.

(d) Finalize the SGVB estimator by completing the `log_normal_pdf()` function, which computes the log probability for a normal distribution given its mean and variance.

Problem 3 *Variational Auto-Encoder on MNIST: Analytical KL Divergence w/o Estimation* (30 points)

In many cases, Monte Carlo sampling is not necessary to estimate all the terms of the ELBO, as some terms can be integrated analytically. In particular, when both $q(z|x)$ and $p(z)$ are

Gaussian distributions, the ELBO can be decomposed into an analytical KL divergence plus the expected reconstruction error.

Assume that:

- The prior $p(z)$ is a standard Gaussian distribution: $p(z) = \mathcal{N}(z; 0, I)$.
- The approximate posterior $q(z|x)$ is a Gaussian distribution with mean $\mu(x)$ and diagonal covariance matrix $\text{diag}(\sigma^2(x))$: $q(z|x) = \mathcal{N}(z; \mu(x), \text{diag}(\sigma^2(x)))$.

The ELBO can then be expressed as:

$$\begin{aligned}\text{ELBO} &= \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x)\|p(z)) \\ &= \frac{1}{2}\sum_{d}(1 + \log((\sigma_d)^2) - (\mu_d)^2 - (\sigma_d)^2) + \sum_{i,j}\log p(x_i|z_{i,j}),\end{aligned}$$

where d is the dimension of the latent space, and i, j are the indices of the data and the latent samples, respectively.

(a) Give a detailed mathematical proof of the above ELBO decomposition. Show all steps clearly.

(b) Run the verfirication code to check if the analytical KL divergence matches the Monte Carlo estimate.

(c) Using the above two losses, train two VAE models on the MNIST dataset (manual tuning of parameters such as `epochs`, `hidden_dims`, `lr`, `coeff` may be necessary). Use the provided evaluation code to visualize the reconstruction results and the generated images (in 2D grid) for both models.

Problem 4 *Training a VAE on a Point Cloud of a Torus* (20 points)

In this problem, you will train a VAE on a 3D point cloud of a torus. The goal is to learn a latent space representation that allows for meaningful interpolation between points on the torus. The training code and point cloud generation code are provided. The VAE should be trained using an analytical KL divergence for the latent space distribution.

(a) Train the VAE on the torus point cloud data using the provided training code. After training, visualize the reconstruction results by decoding latent vectors back to 3D points and comparing the reconstructed torus to the original point cloud.

Next, you will explore the interpolation properties of the latent space by performing linear interpolation between two randomly selected points in the latent space.

(b) Run the provided code to perform linear interpolation between two randomly selected points in the latent space. Visualize the resulting 3D trajectory.

Without surprise, the linear interpolation trajectory exhibits characteristics more akin to a straight-line in Euclidean space rather than adhering to the curved nature of the torus

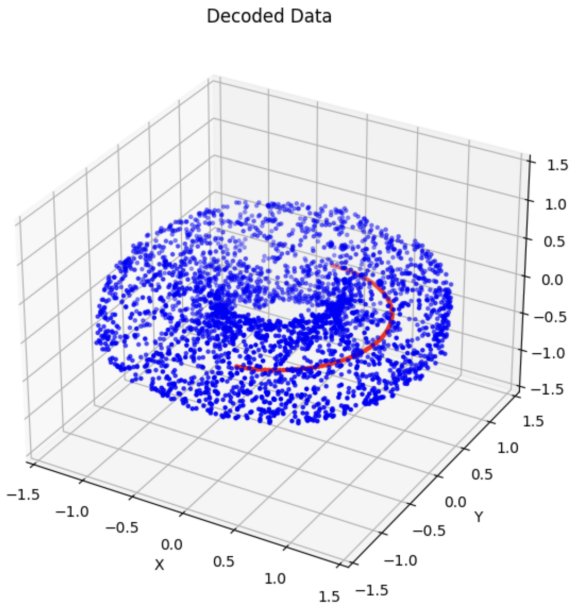

Figure 1: Ideal interpolation trajectory on the torus surface.

surface. This discrepancy is indicative of the limitations inherent in the current latent space representation, which fails to adequately capture the underlying toroidal geometry of the data.

(c) [Optional] Modify the VAE architecture and the latent code representation to better capture the intrinsic toroidal structure of the data, ensuring that the output of linear interpolation trajectories aligns with and follows the torus surface. The resulting interpolation trajectory should be similar to Figure 1.