Generative Models - Assignment 4

Mohammad Mohammadi - 402208592

December 29, 2024

Problem 1:

Part (A)

The PDF of Z is:

$$p_Z(z_1, z_2) = \frac{1}{2\pi} \exp\left(-\frac{z_1^2 + z_2^2}{2}\right).$$

The transformation is:

$$Y = \begin{pmatrix} \frac{3}{5}Z_1(t) + \frac{2}{5}Z_2(t) \\ \frac{1}{5}Z_1(t) + \frac{4}{5}Z_2(t) \end{pmatrix}.$$

The transformation matrix is:

$$A = \begin{pmatrix} \frac{3}{5} & \frac{2}{5} \\ \frac{1}{5} & \frac{4}{5} \end{pmatrix}.$$

Thus, $Y = AZ$.

Since Z is Gaussian with mean $\mathbf{0}$ and covariance I, the random variable Y is also Gaussian with:

$$Y \sim \mathcal{N}(0, AA^T).$$

For AA^T we have:

$$AA^T = \begin{pmatrix} \frac{3}{5} & \frac{2}{5} \\ \frac{1}{5} & \frac{4}{5} \end{pmatrix} \begin{pmatrix} \frac{3}{5} & \frac{1}{5} \\ \frac{2}{5} & \frac{4}{5} \end{pmatrix},$$

$$AA^T = \begin{pmatrix} \frac{9}{25} + \frac{4}{25} & \frac{3}{25} + \frac{8}{25} \\ \frac{3}{25} + \frac{8}{25} & \frac{1}{25} + \frac{16}{25} \end{pmatrix} = \begin{pmatrix} \frac{13}{25} & \frac{11}{25} \\ \frac{11}{25} & \frac{17}{25} \end{pmatrix},$$

$$AA^T = \frac{1}{25} \begin{pmatrix} 13 & 11 \\ 11 & 17 \end{pmatrix}.$$

And the determinant is:

$$\det(AA^T) = \frac{1}{625} \det \begin{pmatrix} 13 & 11 \\ 11 & 17 \end{pmatrix} = \frac{1}{625}(13 \cdot 17 - 11 \cdot 11) = \frac{1}{625}(221 - 121) = \frac{100}{625} = \frac{4}{25}.$$

Hence:

$$\sqrt{\det(AA^T)} = \sqrt{\frac{4}{25}} = \frac{2}{5}.$$

And the inverse of $\begin{pmatrix} 13 & 11 \\ 11 & 17 \end{pmatrix}$ is:

$$\frac{1}{13 \cdot 17 - 11 \cdot 11} \begin{pmatrix} 17 & -11 \\ -11 & 13 \end{pmatrix} = \frac{1}{100} \begin{pmatrix} 17 & -11 \\ -11 & 13 \end{pmatrix},$$

$$(AA^T)^{-1} = 25 \cdot \frac{1}{100} \begin{pmatrix} 17 & -11 \\ -11 & 13 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 17 & -11 \\ -11 & 13 \end{pmatrix}.$$

Since $Y \sim \mathcal{N}(0, AA^T)$, its PDF is:

$$p_Y(y_1, y_2) = \frac{1}{2\pi\sqrt{\det(AA^T)}} \exp\left(-\frac{1}{2} y^T (AA^T)^{-1} y\right).$$

Having $\sqrt{\det(AA^T)} = \frac{2}{5}$:

$$p_Y(y_1, y_2) = \frac{1}{2\pi \cdot (2/5)} \exp\left(-\frac{1}{2} \begin{pmatrix} y_1 & y_2 \end{pmatrix} \frac{1}{4} \begin{pmatrix} 17 & -11 \\ -11 & 13 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}\right)$$

We havee:

$$y^T (AA^T)^{-1} y = \begin{pmatrix} y_1 & y_2 \end{pmatrix} \frac{1}{4} \begin{pmatrix} 17 & -11 \\ -11 & 13 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix},$$

$$= \frac{1}{4} \left(17y_1^2 - 11y_1y_2 - 11y_2y_1 + 13y_2^2\right) = \frac{1}{4}(17y_1^2 - 22y_1y_2 + 13y_2^2).$$

Thus the PDF of Y is:

$$p_Y(y_1, y_2) = \frac{5}{4\pi} \exp\left(-\frac{1}{2}\left(\frac{17}{4}y_1^2 - \frac{22}{4}y_1y_2 + \frac{13}{4}y_2^2\right)\right),$$

$$p_Y(y_1, y_2) = \frac{5}{4\pi} \exp\left(-\frac{1}{2}\left(\frac{17}{4}y_1^2 - \frac{11}{2}y_1y_2 + \frac{13}{4}y_2^2\right)\right).$$

Part (B - 1)

Our flow is a continuous-time one defined by the ODE belwo:

$$\frac{d}{dt} \begin{bmatrix} Z_1(t) \\ Z_2(t) \end{bmatrix} = \begin{bmatrix} \tanh(Z_1(t)^3) \\ \tanh(Z_2(t)^3) \end{bmatrix}.$$

With the progress of time from $t = 0$ to $t = 0.5$, the distribution of $Z(t)$ changes continuously.

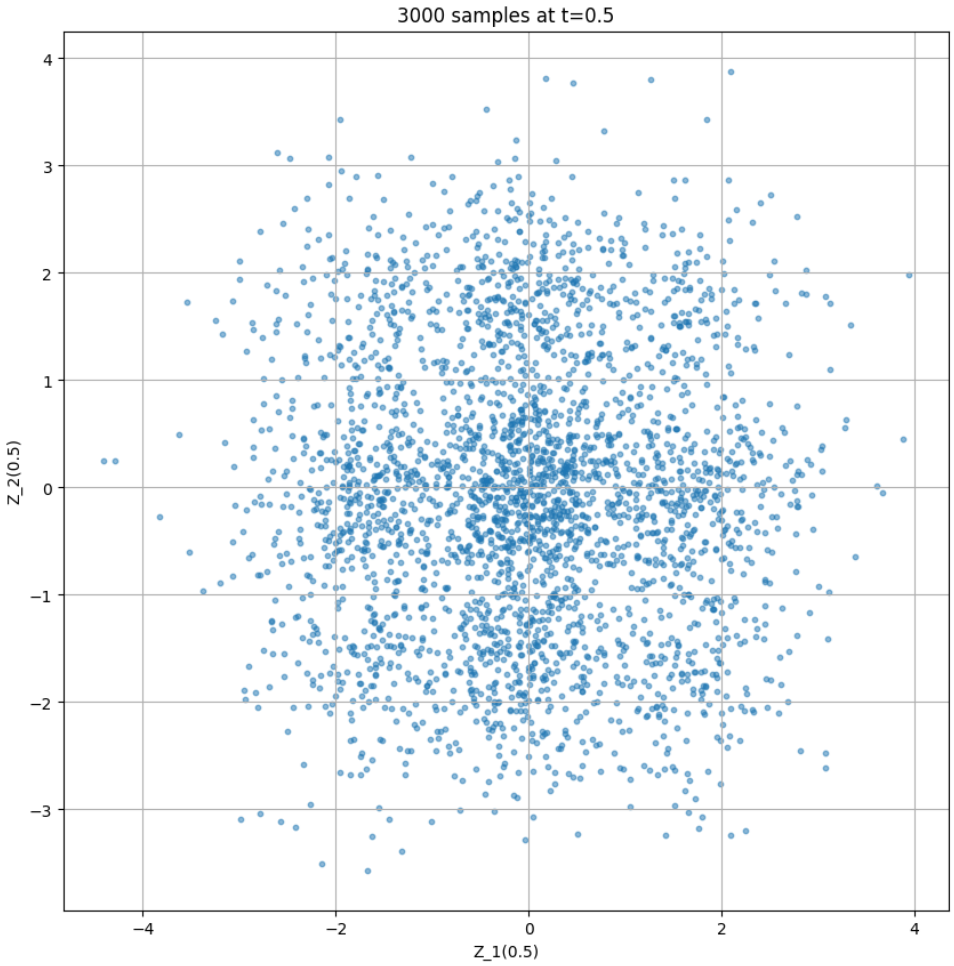

Figure 1: 2D histogram of 3000 sample points at timestamp t=0.5.

Part (B - 2)

Our log-density $\log p(Z(t), t)$ satisfies:

$$\frac{\partial \log p(Z(t), t)}{\partial t} = -\text{trace}\left(\frac{\partial f(Z(t))}{\partial Z}\right),$$

where

$$f(Z(t)) = \begin{bmatrix} \tanh(Z_1(t)^3) \\ \tanh(Z_2(t)^3) \end{bmatrix}.$$

For each of the partial derivatives we compute individually.
$f_1(Z_1) = \tanh(Z_1^3)$:

$$\frac{d}{dZ_1}\tanh(Z_1^3) = \text{sech}^2(Z_1^3) \cdot 3Z_1^2.$$

Similarly for $f_2(Z_2) = \tanh(Z_2^3)$:

$$\frac{d}{dZ_2}\tanh(Z_2^3) = \text{sech}^2(Z_2^3) \cdot 3Z_2^2.$$

The Jacobian is diagonal:

$$\frac{\partial f}{\partial Z} = \begin{bmatrix} 3Z_1^2 \operatorname{sech}^2(Z_1^3) & 0 \\ 0 & 3Z_2^2 \operatorname{sech}^2(Z_2^3) \end{bmatrix}.$$

So the trace is:

$$\operatorname{trace}\left(\frac{\partial f}{\partial Z}\right) = 3Z_1^2 \operatorname{sech}^2(Z_1^3) + 3Z_2^2 \operatorname{sech}^2(Z_2^3).$$

Thus:

$$\frac{d}{dt} \log p(Z(t), t) = -\left[3Z_1(t)^2 \operatorname{sech}^2(Z_1(t)^3) + 3Z_2(t)^2 \operatorname{sech}^2(Z_2(t)^3)\right].$$

Part (B - 3)

We sample from $Z(0) \sim \mathcal{N}(0, I)$ and integrate to different times $t = [0, 0.1, 0.2, 0.3, 0.4, 0.5]$. However, once we have the samples $Z(t)$, we:

Approach

1. Generate 1000 samples from $Z(0) \sim \mathcal{N}(0, I)$.
2. For each chosen time t^*, integrate the ODE forward for each sample from 0 to t^*.
3. Collect the transformed samples $Z(t^*)$ and compute a 2D histogram.
4. Plot the histogram as a heatmap.

By producing heatmaps at multiple time points, we can clearly see how the initial Gaussian distribution spreads, and kind of transforms under the nonlinear flow. Each heatmap shows the distribution at a specific time. This allows us to seee the changes of density over time.

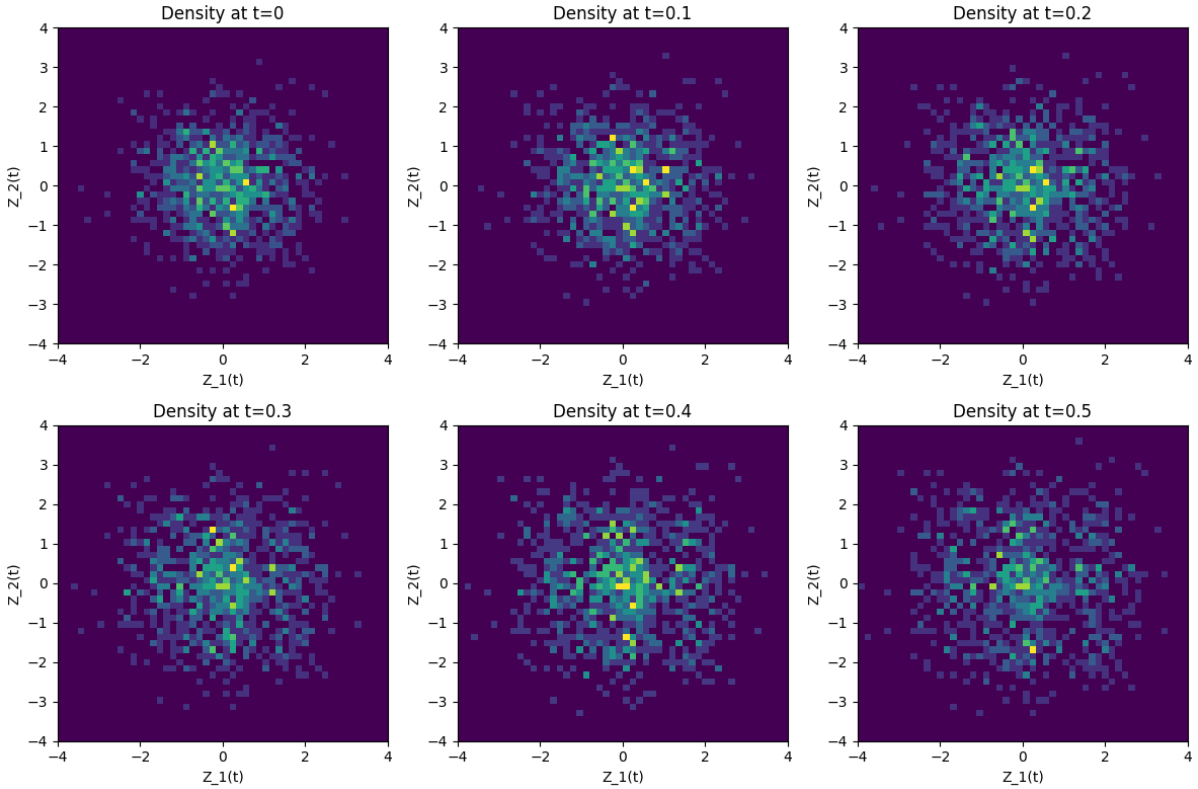

Figure 2: Heatmaps of 2D histograms of densities from t=0 to 0.5 with 0.1 step.

Part (B - 4)

Our goal is to fit θ so that the final distribution at $t = 1$ matches some observed data distribution as closely as possible.

The learning process will be as below:

1. **Defining a Generative Model:** If we think of starting with $Z(0) \sim \mathcal{N}(0, I)$ and then evolving $Z(t)$ up to $t = 1$, we effectively obtain a generative process: we can produce samples from a potentially complex distribution by drawing from a simple Gaussian and integrating forward through time. The complexity of the final distribution depends on θ, which controls the nonlinearities in the ODE.

2. **Likelihood Computation:** One of the beauties of continuous normalizing flows is that we can compute the log-likelihood of the final distribution exactly. We know how the log-density evolves over time and can integrate it just like we integrate the state. Thus, if we have observed data points $x \in \mathbb{R}^2$ that we believe were generated at $t = 1$, we can compute $\log p(x; \theta)$ by "running the flow backward" (or equivalently, running it forward with initial conditions and tracking the log-density changes).

3. **Optimization Goal:** Once we can compute the likelihood $\log p(x; \theta)$ for observed data x, we can define a training objective:

$$\theta^* = \arg\max_{\theta} \mathbb{E}_{x \sim \text{Data}}[\log p(x; \theta)].$$

Which means we choose θ to maximize the average log-likelihood of the observed data, effectively making our continuous normalizing flow "learn" how to morph the initial Gaussian into a distribution that closely resembles the data distribution.

Part (B - 5)

When we model continuous transformations of probability distributions using an ODE, we must numerically solve this ODE to map from a simple base distribution (like a standard Gaussian) to a more complex target distribution. A key point is that we want to be able to accurately compute both the transformed samples and their corresponding log-densities.

A naive approach might be to use simple numerical schemes like the forward Euler method (or similarly straightforward additive update rules). To approximate the solution of:

$$\frac{dZ}{dt} = f(Z(t), t)$$

we may do something like:

$$Z(t + \Delta t) \approx Z(t) + f(Z(t), t)\Delta t.$$

The downsides are as below:

1. Simple methods like Euler's method can be unstable and may produce large errors over time. As we repeatedly step forward, small inaccuracies can accumulate and cause the numerical solution to drift away significantly from the true path. (Sort of a curse of time.)

2. To achieve even a moderate level of accuracy, you would need a very small step size, dramatically increasing computation time. More sophisticated numerical methods (such as Runge-Kutta methods, Dormand-Prince methods, or other ones implemented in common ODE solvers) give a much better balance between accuracy and efficiency. (Better accuracy for same amount of compute.)

3. In continuous normalizing flows, we don't just need the final samples at time $t = 1$; we also need the exact change in log-density. This involves integrating an additional ODE for the log-density that depends on the trace of the Jacobian of f. Any errors in the state integration can be magnified when computing the log-density. Naive methods may lead to poor approximations of log-probabilities, which defeats the whole purpose of using a continuous normalizing flow (where exact likelihood computation is our major advantage).

Problem 2:

Part (A)

Our EBM:

$$p_\phi(x) \propto e^{-E_\phi(x)}$$

And normalization constant:

$$Z_\phi = \int e^{-E_\phi(x)}\, dx$$

Hence

$$p_\phi(x) \;=\; \frac{e^{-\,E_\phi(x)}}{Z_\phi}.$$

With MLE, we need the gradient below:

$$\nabla_\phi \,\log p_\phi(x) \;=\; -\,\nabla_\phi E_\phi(x) \;-\; \nabla_\phi \log Z_\phi.$$

—

Now to calculate the term $\nabla_\phi \log Z_\phi$ from MLE:

$$\log Z_\phi \;=\; \log \int e^{-\,E_\phi(x)}\,dx.$$

If we differentiate from under the integral we get:

$$\nabla_\phi \,\log Z_\phi = \frac{1}{Z_\phi}\int \big(-\,\nabla_\phi E_\phi(x)\big)\, e^{-\,E_\phi(x)}\,dx = -\int \nabla_\phi E_\phi(x)\, p_\phi(x)\,dx.$$

Hence

$$\nabla_\phi \,\log Z_\phi = -\,\mathbb{E}_{x\sim p_\phi}\big[\nabla_\phi E_\phi(x)\big].$$

Part (B)

To train via MLE, we require expectations under $p_\phi(x)$. Sampling from $p_\phi(x)$ directly is generally difficult, so we use **MCMC** methods (e.g. Metropolis-Hastings, or Langevin Dynamics as we had in class) to draw approximate samples $\{x_i'\}$ from the current model $p_\phi(x)$. Then:

$$\nabla_\phi \,\log p_\phi(x) \;\approx\; -\,\nabla_\phi E_\phi(x) \;+\; \frac{1}{N}\sum_{i=1}^{N} \nabla_\phi E_\phi(x_i'),$$

where $x_i' \sim p_\phi(x)$. Summing over data points x (or a minibatch) gives us the total gradient.

The practical downside is that accurate sampling is slow and can lead to large variance and unstable training. In specific:

1. Sampling from the current model $p_\phi(x)$ is difficult because $p_\phi(x)$ is only defined up to a partition function Z_ϕ and can be a complex, high-dimensional and unnormalized distribution.

2. MCMC methods (such as Langevin or Metropolis-Hastings) may take many iterations to approximately converge to $p_\phi(x)$, especially if $E_\phi(x)$ creates a multimodal or sharply peaked energy landscape. This slows down training and introduces approximation error in the gradient.

3. Even if MCMC converges, finite sampling ($\{x_i'\}_{i=1}^N$) can lead to high-variance estimates of the gradient term $\mathbb{E}_{p_\phi}[\nabla_\phi E_\phi(x)]$. This makes learning noisy and unstable.

4. For each gradient update, we must run MCMC to generate fresh model samples. This significantly increases computational overhead compared to models that allow direct sampling or have tractable partition functions.