计算机应用数学-2025 秋

Lecture 1.8: Generative Models

2025.12

Lecturer: 宋骐 *Scribe:* 王亦涵, 赵奇, 王馨语, 郑岭

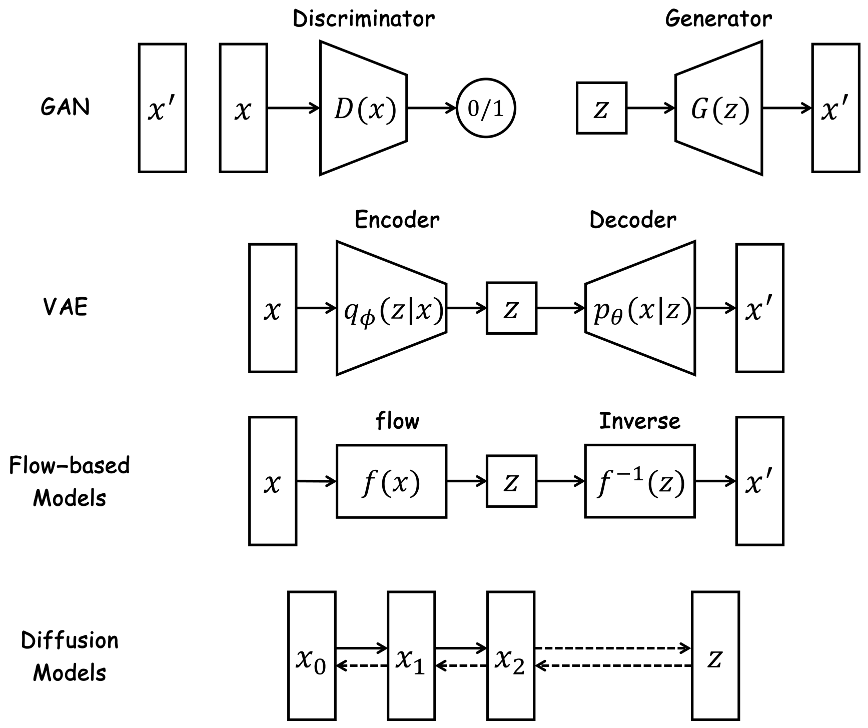
Given training data, generate new samples from the same distribution.

Training data $\to$ Generated samples

$\sim p_{data}(x)$ $\quad\quad \sim p_{model}(x)$

Objective: learn $p_{model}(x)$ similar to $p_{data}(x)$

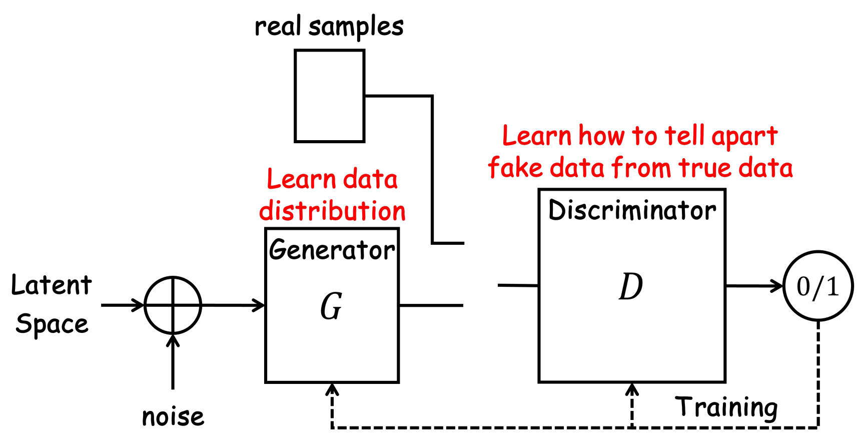
Overview:



1 Generative adversarial network

Discriminator D: estimates the probability of a given sample coming from the real dataset.

Generator G: output synthetic samples given a noise variable input z.



p_z - Data distribution over noise input z.

p_g - The generator's distribution over data x.

p_r - Data distribution over real sample x.

Training Objective:

① Maximize $\mathbb{E}_{x \sim p_r(x)}[log\ D(x)]$ - real data

Maximize $\mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$ - over fake sample

(D is expected to output a probability $D(G(z))$ close to zero)

② Minimize $\mathbb{E}_{z \sim p(z)}[log(1 - D(G(z)))]$

(G is trained to increase the chances of D producing a high probability for a fake example)

Combining ① and ②, D and G are playing a minimax game with the following loss function:

$$\begin{aligned}\min_G \max_D L(D, G) &= \mathbb{E}_{x \sim p_r(x)}[log(D(x))] + \mathbb{E}_{z \sim p(z)}[log(1 - D(G(z)))] \\ &= \mathbb{E}_{x \sim p(x)}[log(D(x))] + \mathbb{E}_{x \sim p_g(x)}[log(1 - D(x))]\end{aligned}$$

Optimal value for D

$$L(G, D) = \int_x (p_r(x)log(D(x)) + p_g(x)log(1 - D(x)))\, dx$$

We want to find the best value of $D(x)$ to maximize $L(G, D)$.

We label $\widetilde{x} = D(x)$, $A = p_r(x)$, $B = p_g(x)$

$f(\widetilde{x}) = Alog\widetilde{x} + Blog(1 - \widetilde{x})$

(x is sampled over all possible values, thus we can ignore the integral - 积分)

$$\begin{aligned}\frac{df(\widetilde{x})}{d\widetilde{x}} &= A\cdot\frac{1}{ln10}\frac{1}{\widetilde{x}} - B\cdot\frac{1}{ln10}\frac{1}{1-\widetilde{x}}\\ &= \frac{1}{ln10}(\frac{A}{\widetilde{x}} - \frac{B}{1-\widetilde{x}})\\ &= \frac{1}{ln10}\frac{A-(A+B)\widetilde{x}}{\widetilde{x}(1-\widetilde{x})}\end{aligned}$$

set $\frac{df(\widetilde{x})}{d\widetilde{x}} = 0$ we get

$$D^*(x) = \widetilde{x}^* = \frac{A}{A+B} = \frac{p_r(x)}{p_r(x)+p_g(x)} \in [0,1)$$

which is the best value of D.

Δ What is the global optimal?

when both G and D are at their optimal values, we have $p_g = p_r$ and $D^*(x) = 1/2$

$$\begin{aligned}L(G,D^*) &= \int_x (p_r(x)log(D^*(x)) - p_g(x)log(1-D^*(x)))dx\\ &= log\frac{1}{2}\int_x p_r(x)dx + log\frac{1}{2}\int_x p_g(x)dx\\ = -2log2\end{aligned}$$

Δ More about the loss function.

JS divergence between p_r and p_g:

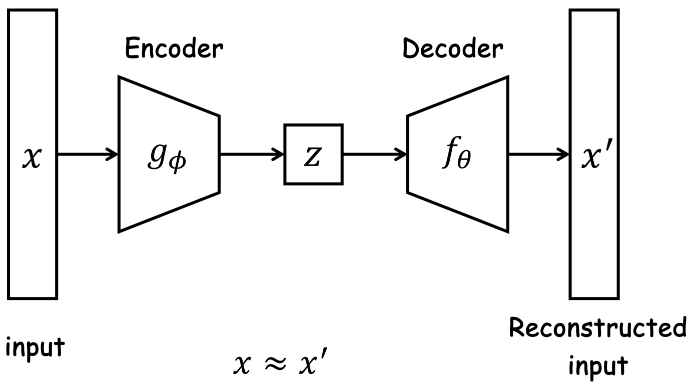
$$\begin{aligned}D_{JS}(p_r||p_g) =& \frac{1}{2}D_{KL}(p_r||\frac{p_r+p_g}{2}) + \frac{1}{2}D_{KL}(p_g||\frac{p_r+p_g}{2})\\ =& \frac{1}{2}(log2 + \int_x p_r(x)log\frac{p_r(x)}{p_r+p_g(x)}dx)+\\ & \frac{1}{2}(log2 + \int_x p_g(x)log\frac{p_g(x)}{p_g+p_r(x)}dx)\\ =& \frac{1}{2}(log4 + L(G,D^*))\end{aligned}$$

Thus $L(G,D^*) = 2D_{JS}(p_r||p_g) - 2log2$.

Thus, when D is optimal, the loss function quantifies the similarity between the p_g and p_r by JS divergence.

The best G^* leads to the minimum $L(G^*,D^*) = -2log2$.

2 Autoencoder



- Encoder $g(\cdot)$: translates the original high-dimension input into the latent low-dimensional code.
- Decoder $f(\cdot)$: recovers the data from the code.

$z = g_\phi(x) \qquad\qquad x' = f_\theta(g_\phi(x))$

ϕ and θ are parameters.

(θ, ϕ) are learned together to have $x \approx f_\theta(g_\phi(x))$

We can use different metrics, e.g. MSE loss:

$$L_{AE}(\theta,\phi) = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - f_\theta(g_\phi(x^{(i)})))^2$$

$x^{(i)} \sim i$th data point.

Δ VAE: Variational AutoEncoder.

Instead of mapping x into a fixed vector, we want to map it into a distribution, p_θ (parameterized by θ).

Assuming we know θ (denoted as θ^*), we can generate a sample that looks like a real data point $x^{(i)}$.

1. Sample $z^{(i)}$ from a prior distribution $p_{\theta^*}(z)$

2. Generate $x^{(i)}$ from a conditional distribution $p_{\theta^*}(x|z = z^{(i)})$.

θ^* maximizes the probability of generating real data samples

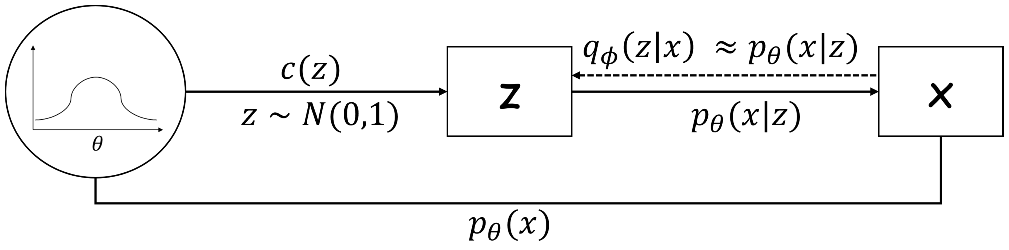
$$\theta^* = arg\max_\theta \prod_{i=1}^{n} p_\theta(x^{(i)})$$

Commonly we use log probabilities to covert the above to

$$\theta^* = \arg\max_\theta \sum_{i=1}^{n} \log p_\theta(x^{(i)})$$

$$p_\theta(x^{(i)}) = \int_{p_\theta} (x^{(i)}|z) p_\theta(z) dz. \ (Bayesian)$$

It's very expensive to check all the possibles of z, thus $p_\theta(x^{(i)})$ is hard to compute in this way. We next introduce an approximation function.



- Encoder: the conditional probability $p_\theta(x|z)$, similar to $f_\theta(x|z)$. We call $p_\theta(x|z)$ probabilistic encoder.
- Decoder: the approximation function $q_\phi(z|x)$ is the probabilistic encoder, similar to $g_\phi(z|x)$.

Loss function: Estimated posterior $q_\phi(z|x) \simeq p_\theta(z|x)$ real one. We use KL-divergence here

$$\min_\phi D_{KL}(q_\phi(z|x)|p_\theta(z|x))$$

Note: Here we use $D_{KL}(q_\phi|p_\theta)$ (reversed KL) instead of $D_{KL}(p_\theta|q_\phi)$ (forward KL). (Reason can be found in the extend reading material.)

$$
\begin{aligned}
&D_{KL}(q_\phi(z|x)|p_\theta(z|x)) \\
&= \int_{q_\phi(z|x)} \log \frac{q_\phi(z|x)}{p_\theta(z|x)} dz \\
&= \int_{q_\phi(z|x)} \log \frac{q_\phi(z|x) p_\theta(x)}{p_\theta(z,x)} dz \\
&= \int_{q_\phi(z|x)} (\log p_\theta(x) + \log \frac{q_\phi(z|x)}{p_\theta(z,x)}) dz \\
&= \log p_\theta(x) + \int_{q_\phi(z|x)} \log \frac{q_\phi(z|x)}{p_\theta(x|z)} p_\theta(z) dz \\
&= \log p_\theta(x) + \mathbb{E}_{z\sim q_\phi(z|x)}[\log \frac{q_\phi(z|x)}{p_\theta(z)} - \log p_\theta(x|z)] \\
&= \log p_\theta(x) + D_{KL}(q_\phi(z|x)||p_\theta(z)) - \mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x|z)
\end{aligned}
$$

$\therefore \log p_\theta(x) - D_{KL}(q_\phi(z|x)||p_\theta(z|x)) = \mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x|z) - D_{KL}(q_\phi(z|x)||p_\theta(z))$

For $\log p_\theta(x) - D_{KL}(q_\phi(z|x)||p_\theta(z|x))$, we want to maximize when learning the true distributions: we want to maximize the $(\log)$-likelihood of generating real data (i.e. $\log p_\theta(x)$), while minimize the difference between the real and estimated posterior distributions. ELBO-Evidence lower bound.

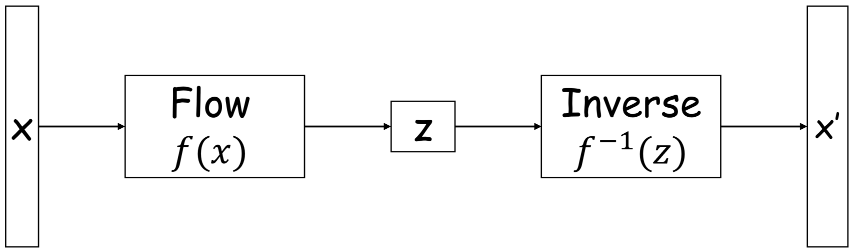
We can take negation of the above result.

$$
\begin{aligned}
L_{VAE}(\theta,\phi) &= -\log p_\theta(x) + D_{KL}(q_\phi(z|x)||p_\theta(z|x)) \\
&= -\mathbb{E}_{z\sim q_\phi} \log p_\theta(x|z) + D_{KL}(q_\phi(z|x)||p_\theta(z))
\end{aligned}
$$

$$
\theta^*, \phi^* = \arg\min_{\theta,\phi} L_{VAE}
$$

3 Flow-based Generative Models

Note that GAN and VAE do not explicitly learns the probability density function of real data $p(x)$.



I.Recap Linear Algebra.

Jacobian Matrix.

Given a function of mapping a n-dimensional input vector x to a m-dimensional output vector $f : \mathbb{R}^n \to \mathbb{R}^m$, the matrix of all first-order partial derivatives of this function is called the Jacobian matrix.

$$J_{ij} = \frac{\partial f_i}{\partial x_j}$$

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

Determinant (行列式)

$$\det M = \det \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \sum_{j_1 j_2 \cdots j_n} (-1)^{\tau(j_1 j_2 \cdots j_n)} a_1 j_1 a_2 j_2 \dots a_n j_n$$

M is a $n \times n$ matrix.

If $det(M) = 0$ then M is not invertible, otherwise, if $det(M) \neq 0$, M is invertible.

Change of Variable Theorem (变量替换定理)

Given a random variable z and its known probability density function $z \sim \pi(z)$, we would like to construct a new random variable using a $1-1$ mapping function $x = f(z)$. f is invertible, so $z = f^{-1}(x)$.

Now the question is how to infer the unknown probability density function of the new variable $p(x)$?

$$\int p(x)dx = \int \pi(z)dz = 1$$

$$p(x) = \pi(z)|\frac{dz}{dx}| = \pi(f^{-1}(x))|\frac{df^{-1}}{dx}| = \pi(f^{-1}(x))|(f^{-1})^1(x)|$$

Here $\int \pi(z)dz$ is the sum of an infinite number of rectangles of infinitesimal width Δz.

The height of such a rectangle at position z is the value of the density function $\pi(z)$.

We substitute the variable

$$z = f^{-1}(x) \to \frac{\Delta z}{\Delta x} = (f^{-1}(x))'$$

$$\Delta z = (f^{-1}(x))'\Delta x$$

Here $|(f^{-1}(x))'|$ indicates the ratio between the area of rectangles defined in two different coordinate of variables z and x respectively.

Above is a single variable case.

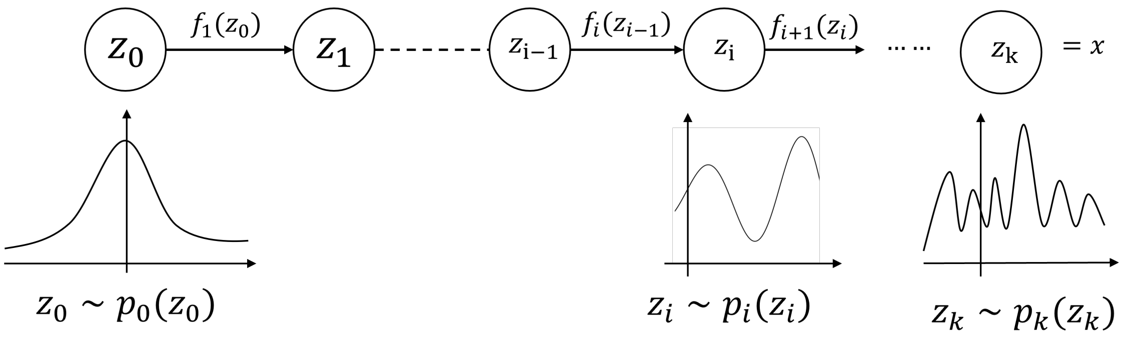
Multi-variable version looks similar:

$$z \sim \pi(z), x = f(z), z = f^{-1}(x)$$

$$p(x) = \pi(z)|\det \frac{dz}{dx}| = \pi(f^{-1}(x))|\det \frac{df^{-1}}{dx}|$$

Normalizing Flows.

A normalizing flow transforms a simple distribution into a complex one by applying a sequence of invertible transformation functions.



$$z_{i-1} \sim p_{i-1}(z_{i-1})$$

$$z_i = f_i(z_{i-1}), z_{i-1} = f_i^{-1}(z_i)$$

$$p_i(z_i) = p_{i-1}(f_i^{-1}(z_i))|\det \frac{df_i^{-1}}{dz_i}|$$

$$\begin{aligned} p_i(z_i) &= p_{i-1}(f_i^{-1}(z_i))|\det \tfrac{df_i^{-1}}{dz_i}| \\ &= p_{i-1}(z_{i-1})|\det(\tfrac{df_i}{dz_{i-1}})^{-1}| \text{ (inverse func theorem)} \\ &= p_{i-1}(z_{i-1})|\det \tfrac{df_i}{dz_{i-1}}|^{-1} \text{ (a property of Jacobians of invertible func)} \end{aligned}$$

$$log\ p_i(z_i) = log\ p_{i-1}(z_{i-1}) - log\ |det \frac{df_i}{dz_{i-1}}|$$

* Inverse function theorem:

$$\text{if } y = f(x) \quad and \quad x = f^{-1}(y)$$
$$\frac{df^{-1}(y)}{dy} = \frac{dx}{dy} = (\frac{dy}{dx})^{-1} = (\frac{df(x)}{dx})^{-1}$$

Given such a chain of probability density functions, we know the relationship between each pair of consecutive variables.

$$x = z_k = f_k \circ f_{k-1} \circ \cdots \circ f_1(z_0)$$
$$\begin{aligned} logp(x) &= log\ \pi_k(z_k) = log\ \pi_{k-1} - log\ |det \frac{df_k}{dz_{k-1}}| \\ &= log\ \pi_{k-2}(z_{k-2}) - log\ |det \frac{df_{k-1}}{dz_{k-2}}| - log|det \frac{df_k}{dz_{k-1}}| \\ &= \cdots \\ &= log\ \pi_0(z_0) - \sum_{i=1}^{k} log\ |det \frac{df_i}{dz_{i-1}}| \end{aligned}$$

f_i should have two properties.

1. it is easily invertible.
2. It's Jacobian determinant is easy to compute.

Now the exact log-likelihood of input data $logp(x)$ becomes tractable. The training criterion:

$$L(p) = -\frac{1}{|D|} \sum_{x \in D} log\ p(x)$$

RealNLP (Real-valued Non-Volume Preserving)
The input dimensions are split into two parts:

- The first d dimensions stay same.
- The d_{t+1} to D dimensions,

$$\begin{aligned} y_{1:d} &= x_{1:d} \\ y_{d+1:D} &= x_{d+1:D} \odot exp(s(x_{1:d}) + t(x_{1:d})). \end{aligned}$$

$s(\cdot)$ and $t(\cdot)$ are scale and translation functions, $\mathbb{R} \to \mathbb{R}^{D-d}$.

$\odot$ operator is the element-wise product.

–<u>Condition 1.</u> Easy invertible.

$$\begin{aligned} y_{1:d} &= x_{1:d} \\ y_{d+1:D} &= x_{d+1:D} \odot exp(s(x_{1:d}) + t(x_{1:d})) \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} x_{1:d} &= y_{1:d} \\ x_{d+1:D} &= (y_{d+1:D} - t(y_{1:d})) \odot exp(-s(y_{1:d})) \end{aligned}$$

–<u>Condition 2.</u> Jacobian determinant is easy to compute.

$$J = \begin{bmatrix} \mathbb{I}_d & O_{dx(D-d)} \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}} & diag(exp(s(x_{1:d}))) \end{bmatrix}$$

4 Diffusion Models.

Diffusion Models are inspired by non-equilibrium thermodynamics(非平衡热力学).

They define a Markov chain of diffusion steps to slowly add random noise to data and then learn to reverse the diffusion process to construct desired data samples from the noise.

I. Forward diffusion process.

Given a data point sampled from a real data distribution $x_0 \sim q(x)$, a forward diffusion process add small amount of Gaussian noise to the sample in T steps, producing a sequence of noisy samples $x_1, \ldots, x_T$.

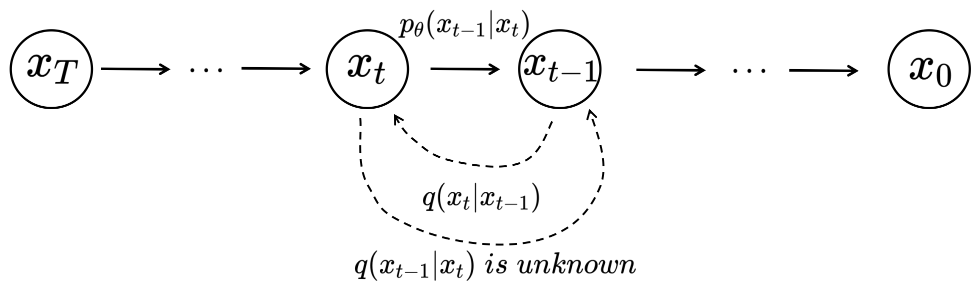
The step sizes are controlled by a variance schedule

$$\{\beta_t \in (0,1)\}_{t=1}^{T}$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t, \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

$$q(x_{1:T}|x_o) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$

A nice property of the forward diffusion process is that we can sample x_t at any arbitrary time step t in a dosed from using <u>reparameterization trick</u>.



Let $\partial_t = 1 - \beta_t \quad and \quad \bar{\partial}_t = \prod_{i=1}^{t} \partial_i$

$$\begin{aligned} x_t &= \sqrt{\partial_t}\, x_{t-1} + \sqrt{1-\partial_t}\, \epsilon_{t-1} \qquad \epsilon_{t-1}, \epsilon_{t-2}, \cdots \sim N(0, I) \\ &= \sqrt{\partial_t \partial_{t-1}} x_{t-2} + \sqrt{1 - \partial_t \partial_{t-1}} \bar{\epsilon}_{t-2} \qquad \bar{\epsilon}_{t-2} \text{ merges two Gaussians} \\ &= \cdots = \sqrt{\bar{\partial}_t}\, x_0 + \sqrt{1 - \bar{\partial}_t}\, \epsilon \end{aligned}$$

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\partial}_t}\, x_0, (1 - \bar{\partial}_t) I)$$

when we merge two Gaussians $\mathcal{N}(0, \sigma_1^2 I), \mathcal{N}(0, \sigma_2^2 I)$, the two distribution is $\mathcal{N}(0, (\sigma_1^2 + \sigma_2^2) I)$, the merged standard deviation is

$$\sqrt{(1 - \partial_t) + \partial_t (1 - \partial_{t-1})} = \sqrt{1 - \partial_t \partial_{t-1}}$$

Usually, we can afford a larger update step when the sample gets noisier, so $\beta_1 < \beta_2 \cdots < \beta_T, \bar{\partial}_1 > \cdots > \bar{\partial}_T$.

II. Reverse diffusion process.

If we can reverse the above process and sample from $q(x_{t-1}|x_t)$, we will be able to recreate the true sample from a Gaussian noise input, $\mathcal{X}_T \sim \mathcal{N}(0, 1)$. Unfortunately, we cannot easily estimate $q(x_{t-1}|x_t)$ because it needs to use the entire dataset and we therefore need to learn a model p_θ to approximate these condtional probabilities in order to run the reverse diffusion process.

$$\begin{gathered} p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1|x_t}) \\ p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; M_\theta(x_t, t), \textstyle\sum_\theta (x_t, t)) \end{gathered}$$

Following the standard Gaussian density function, the mean and variance can be parameterized as follows:

$$
\begin{aligned}
&(\partial_t = 1-\beta_t,\ \bar{\partial}_t = \textstyle\prod_{i=1}^{T}\partial_i):\\
\tilde{\beta}_t &= 1/(\frac{\partial_t}{\beta_t}+\frac{1}{1-\bar{\partial}_{t-1}}) = 1/(\frac{\partial_t-\bar{\partial}_t+\beta_t}{\beta_t(1-\bar{\partial}_{t-1})})\\
&= \frac{1-\bar{\partial}_{t-1}}{1-\bar{\partial}_t}\cdot\beta_t\\
\tilde{M}_t(x_t,x_0) &= (\frac{\sqrt{\partial_t}}{\beta_t}x_t+\frac{\sqrt{\bar{\partial}_{t-1}}}{1-\bar{\partial}_{t-1}}x_0)/(\frac{\partial_t}{\beta_t}+\frac{1}{1-\bar{\partial}_{t-1}})\\
&= (\frac{\sqrt{\partial_t}}{\beta_t}x_t+\frac{\sqrt{\bar{\partial}_{t-1}}}{1-\bar{\partial}_{t-1}}x_0)\cdot\frac{1-\bar{\partial}_{t-1}}{1-\bar{\partial}_t}\cdot\beta_t\\
&= \frac{\sqrt{\partial_t}(1-\bar{\partial}_{t-1})}{1-\bar{\partial}_t}x_t+\frac{\sqrt{\bar{\partial}_{t-1}\beta_t}}{1-\bar{\partial}_t}\cdot x_0
\end{aligned}
$$

we can represent $x_0 = \frac{1}{\sqrt{\bar{\partial}_t}}(x_t-\sqrt{1-\bar{\partial}_t}\ \epsilon_t)$ and

$$
\begin{aligned}
\tilde{M}_t &= \frac{\sqrt{\partial_t}(1-\bar{\partial}_{t-1})}{1-\bar{\partial}_t}x_t+\frac{\sqrt{\bar{\partial}_{t-1}\beta_t}}{1-\bar{\partial}_t}\cdot\frac{1}{\sqrt{\bar{\partial}_t}}\cdot(x_t-\sqrt{1-\bar{\partial}_t}\ \epsilon_t)\\
&= \frac{1}{\sqrt{\partial_t}}(x_t-\frac{1-\partial_t}{\sqrt{1-\bar{\partial}_t}}\ \epsilon_t)
\end{aligned}
$$

Such a setup is very similar to VAE, thus we can use the variational lower bound to optimize the negative log-likelihood.

$$
\begin{aligned}
-\log p_\theta(x_0) &\le -\log p_\theta(x_0) + D_{KL}\left(q(x_{1:T}|x_0)||p_\theta(x_{1:T}|x_0)\right)\\
&= -\log p_\theta(x_0) + \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[\log\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})|p_\theta(x_0)}\right]\\
&= -\log p_\theta(x_0) + \mathbb{E}_q\left[\log\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}+\log p_\theta(x_0)\right]\\
&= \mathbb{E}_q\left[\log\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right]
\end{aligned}
$$

Let $L_{VLB} = \mathbb{E}_q(x_{0:T})\left[\log\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right] \ge -\mathbb{E}_q(x_0)\log p_\theta(x_0)$
We want to minimize the cross entropy as the learning objective

$$
\begin{aligned}
L_{CE} &= -\mathbb{E}_{q(x_0)} \log p_\theta(x_0) \\
&= -\mathbb{E}_{q(x_0)} \log \left(\int p_\theta(x_{0:T}) dx_{1:T} \right) \\
&= -\mathbb{E}_{q(x_0)} \cdot \log \left(\int q(x_{1:T}|x_0) \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} dx_{1:T} \right) \\
&= -\mathbb{E}_{q(x_0)} \log \left(\mathbb{E}_{q(x_{1:T}|x_0)} \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right) \\
&\leq -\mathbb{E}_{q(x_{0:T})} \log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \\
&= \mathbb{E}_{q(x_{0:T})} \left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} \right] = \mathrm{L}_{\mathrm{VLB}}
\end{aligned}
$$

We next rewrite the objective to be a combination of several KL-divergence and entropy terms.

$$
\begin{aligned}
\mathrm{L}_{\mathrm{VLB}} &= \mathbb{E}_{q(x_{0:T})}\left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right] \\
&= \mathbb{E}_q\left[\log \frac{\prod_{t=1}^T q(x_t|x_{t-1})}{p_\theta(x_T)\prod_{t=1}^T p_\theta(x_{t-1}|x_t)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(x_T) + \sum_{t=1}^T \log \frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(x_T) + \sum_{t=2}^T \log \frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)} + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(x_T) + \sum_{t=2}^T \log \left(\frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)} \cdot \frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}\right) + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(x_T) + \sum_{t=2}^T \log \frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)} + \sum_{t=2}^T \log \frac{q(x_t|x_0)}{q(x_{t-1}|x_0)} + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(x_T) + \sum_{t=2}^T \log \frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)} + \log \frac{q(x_t|x_0)}{q(x_1|x_0)} + \log \frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right] \\
&= \mathbb{E}_q\left[log \frac{q(x_T|x_0)}{p_\theta(x_T)} + \sum_{t=2}^T \log \frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)} - \log p_\theta(x_0|x_1)\right] \\
&= \mathbb{E}_q\Bigg[D_{KL}(q(x_T|x_0)||p_\theta(x_T)) \qquad - L_T \\
&+ \sum_{t=2}^T D_{KL}(q(x_{t-1}|x_t,x_0) \parallel p_\theta(x_{t-1}|x_t)) \qquad - L_{t-1} \\
&- \log p_\theta(x_0|x_1)\Bigg] \qquad - L_0 \\
& L_{\mathrm{VLB}} = L_T + L_{T-1} + \cdots + L_0
\end{aligned}
$$

where $L_t = D_{KL}\left(q(x_t|x_{t+1},x_0) \parallel p_\theta(x_t|x_{t+1})\right)$ for $1 \le t \le T-1$.

Every KL term in L_{VLB} (except for L_0) compares two Gaussian distributions and therefore they can be computed in closed form.

L_T is constant and can be ignored during training because q has no learnable parameters and x_T is Gaussian noise.

III. Parameterization of L_t for Training Loss

Recall that we need to learn a neural network to approximate the conditional probability distributions in the reverse diffusion process.

$$p_\theta(x_{t-1}|x_t) = \mu\left(x_{t-1}; \mu_\theta(x_t, t), \sum_\theta(x_t, t)\right)$$

We would like to train μ_θ to predict $\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{\sqrt{1-\bar{\alpha}_t}}}\epsilon_t\right)$.

x_t is available as input at training time, we can reparameterize the Gaussian noise term instead to make it predict ϵ_t from the input x_t at time step t.

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)$$

Thus,

$$x_{t-1} = \mu\left(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right), \sum_\theta(x_t, t)\right)$$

L_t is parameterized to minimize the difference from $\tilde{\mu}$:

$$\begin{aligned}
L_t &= \mathbb{E}_{x_0,\epsilon}\left[\frac{1}{2||\sum_\theta(x_t,t)||_2^2}||\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)||^2\right] \\
&= \mathbb{E}_{x_0,\epsilon}\left[\frac{1}{2||\sum_\theta||_2^2}||\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t\right) - \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)||^2\right] \\
&= \mathbb{E}_{x_0,\epsilon}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)||\sum_\theta||_2^2}||\epsilon_t - \epsilon_\theta(x_t, t)||^2\right] \\
&= \mathbb{E}_{x_0,\epsilon}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)||\sum_\theta||_2^2}||\epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, t)||^2\right]
\end{aligned}$$

We can further simplify the above:

$$\begin{aligned}
L_t^{simple} &= E_{t\sim[1,T],x_0,\epsilon_t}[||\epsilon_t - \epsilon_\theta(x_t, t)||^2] \\
&= E_{t\sim[1,T],x_0,\epsilon_t}[||\epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, t)||^2] \\
L_{simple} &= L_t^{simple} + C
\end{aligned}$$

Other topics;

Parameterization of β_t

Parameterization of $\sum_\theta$

Algorithm 1 Training

```
1 : repeat
2 :     $x_0 \sim q(x_0)$
3 :     $t \sim \text{Uniform}(\{1, ..., T\})$
4 :     $\epsilon \sim \mu(0, I)$
5 :     Take gradient descent step on$\nabla_\theta ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha_t}} x_0 + \sqrt{1 - \bar{\alpha_t}}\epsilon, t)||^2$
6 : Until converged
```

Algorithm 2 Sampling

```
1: $x_T \sim \mu(0, 1)$
2: **for** $t = T, ..., 1$ **do**
3:     $z \sim \mu(0, I)\text{if}t > 1, \text{else}z = 0$
4:     $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)) + \delta_t z$
5: **end for**
6: **return** $x_0$
```

4.1 Summary

GAN

pros: - Sample generation is faster compared with flow-based methods as it does not need to generate the full sequence.

-Don't introduce any deterministic bias compared with VAE

cons: - Training a GAN requires finding a Nash equilibrium(hard)

AE/VAE

pros: – Can handle missing data effectively, allowing for more robust analysis.

cons: – The generated samples are much more blurred than those coming from GANs

Diffusion Model

pros: - Can generate complex samples

cons: - rely on a long Markov chain of diffusion steps to generate samples, thus quite expensive in terms of time and computation.