

Recent_Note

LiXiaoLong

August 19, 2024

Contents

1	GAN	2
1.1	简介	2
1.1.1	设计思路	2
1.1.2	损失函数和训练策略	3
1.1.3	数学推导	3
1.2	Code	3
2	VAE model	5
2.1	设计思路	5
2.1.1	Auto Encoder 结构	5
2.1.2	灵感乍现, 这个 decoder 似乎有做图像生成的潜力	6
2.1.3	VAE 怎么满足这些要求的?	6
2.2	损失函数怎么设计出来的?	8
2.3	Code:	9
3	Diffusion Model	10
3.1	简介	10
3.2	前向——加噪声	10
3.3	反向——去噪声	11

Chapter 1

GAN

1.1 简介

1.1.1 设计思路

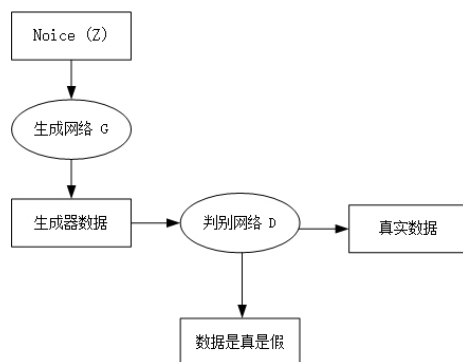


Figure 1.1: GAN 设计思路

GAN 包括两个模型，一个是生成模型 G (Generator)，一个是判别模型 D (Discriminator)。它们的功能分别是：

G 负责生成图片，接收一个随机的噪声 z ，通过该噪声生成图片，记为 $G(z)$ 。

D 负责判别一张图片是否“真实”。其输入是 x ，代表一张图片，输出 $D(x)$ 表示 x 为真实图片的概率。输出为 1 代表真实图片的概率为 100%，而输出为 0 则代表图片不可能是真实的（真实实例来源于数据集，伪造实例来源于生成模型）。

有一个很好的比喻，就是枯叶蝶的演化过程类似于树叶，枯叶蝶不需要认识树叶，但能通过变异逃避捕食者（筛选器），这样的自然选择使枯叶蝶越来越像树叶。同理，生成器产生的图片概率分布也会越来越接近真实数据集的概率分布。

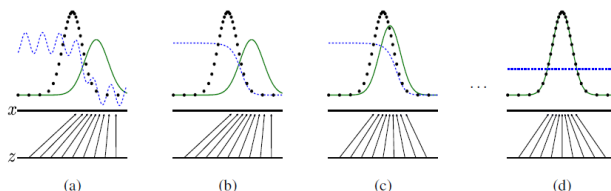
1.1.2 损失函数和训练策略

(1) 损失函数:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (1.1)$$

含义很直接, 对生成器, 尽可能让 $\mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$ 更小, 也就是 $D(G(z))$ 尽可能大, 前半段不涉及 z , 当常数处理。
对判别器, 就是真图像判别的结果越接近 1 越好, 假图像越接近 0 越好。

训练策略:



Note: 判别器是小蓝, 真实图片的概率分布是小黑, 生成器整的映射是下半边的箭头, 由全数据区域到生成的图像空间, 绿色是生成图像空间的概率分布。

- 生成器和真实图像的概率分布偏差大, 性能是比较差, 判别器虽然大体在真是图像概率大的地方高, 但是不稳定。
- 判别器被迭代了几轮, 区分良好。
- 生成器被迭代更新, 概率分布趋近了真实图像分布一些。
- 重复上面两步...
- 大结局, 以假乱真, 判别器 out。

1.1.3 数学推导

推荐文章:<https://www.cnblogs.com/LXP-Never/p/9706790.html>

1.2 Code

点击这里查看代码文件:<https://xiaolong-li1.github.io/Blogs/GAN/GAN.ipynb>

(1): 伪代码:

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Sample minibatch of m examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D(x^{(i)}) + \log \left(1 - D(G(z^{(i)})) \right) \right].$$

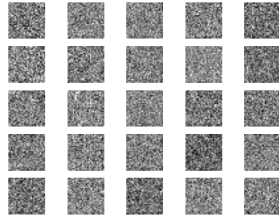
end for

- Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Update the generator by descending its stochastic gradient:

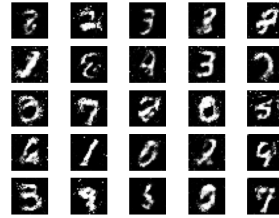
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log \left(1 - D(G(z^{(i)})) \right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.



(a) 训练 0 个 epoch 的生成结果



(b) 训练 200 个 epoch 生成的结果

Figure 1.3: 训练不同 epoch 数的生成结果对比

Chapter 2

VAE model

2.1 设计思路

2.1.1 Auto Encoder 结构

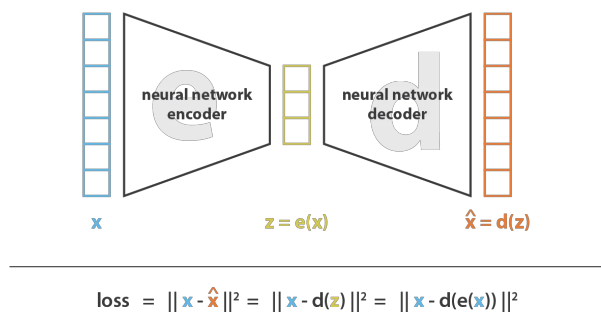


Figure 2.1: Auto Encoder 结构

VAE 结构基于 AE，这种网络结构有一个很经典的功能，数据降维，可以把高维度的数据变成很低的维度，但是仍然保留大部分语义特征（因为它能通过一个固定的函数（解码器）以比较小的误差恢复出来）。至于这个网络能干什么，去噪，数据压缩之类的。

- 这时候刚好来引入一个概念，**latent space**（我就叫它表示空间好了），也就是图中 z 向量所在的空间，
- 第二个概念是，latent space 的正则性，这里用两幅图描述。

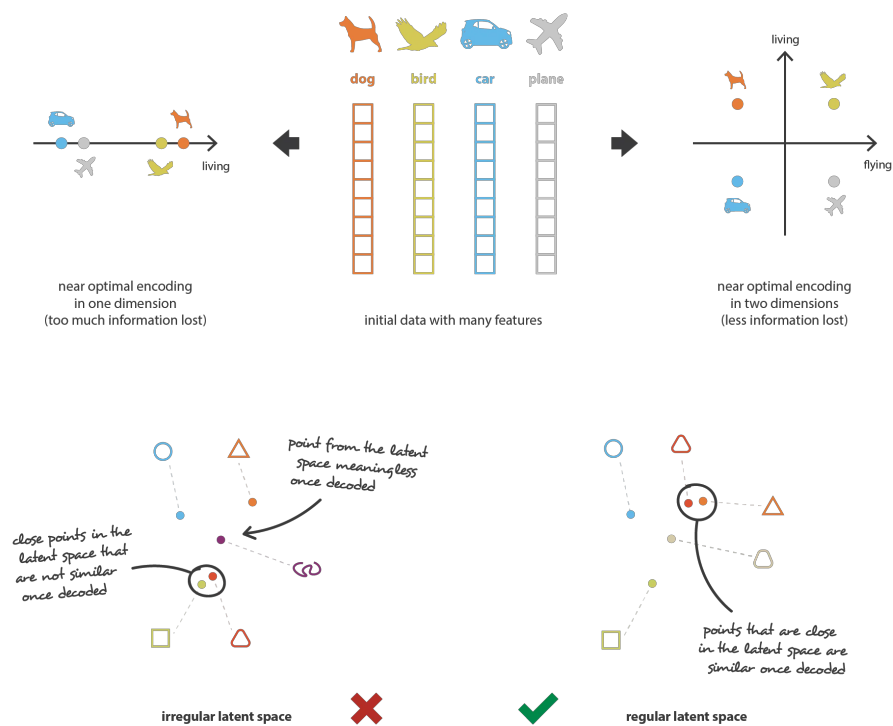


Figure 2.2: 正则化的直观感受

这里的左侧的表示空间和右侧比较就没什么结构，差别很大的东西在表示空间里的位置却没有散开，相似的没有聚在一堆。

2.1.2 灵感乍现，这个 **decoder** 似乎有做图像生成的潜力

它能用低维度的一些看似随机的向量整出来有意义的图片，如果我们直接随机生成一些向量输进去会不会也能搞出来新的作品？不过先考虑下面的要求。

1. 随机的向量都能有图片对应，这说明一个事情，latent space 的大部分空间都会在高维空间得有有意义的图像对应。我们原有的训练策略从没有对表示空间做任何要求，它可以就好多离得很远的片区有有意义的图像对应。
2. 它一个输入绝对只有一个输出，这不好，我们应该生成多个类似的但又不一样的图像以供选择。

2.1.3 VAE 怎么满足这些要求的？

VAE 结构

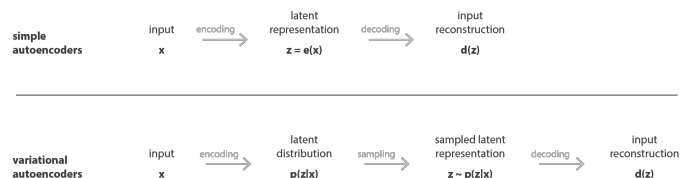


Figure 2.3: VAE 和 AE 区别

它的编码器不再是和表征空间的一个点对应了，而是一个概率分布，解码器也是得到的一个概率分布，不过方差是提前固定的

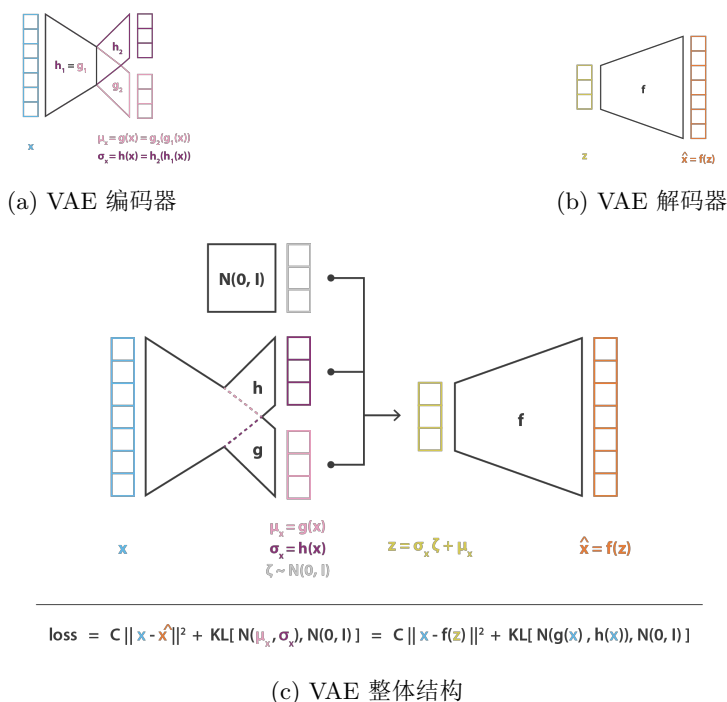
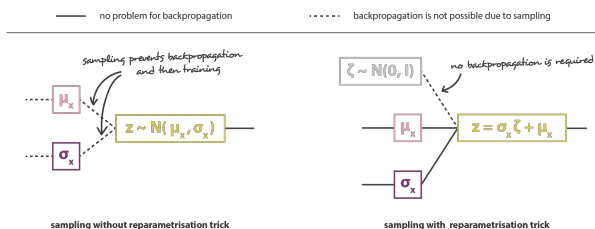


Figure 2.4: VAE 结构

- 图 a: 解码器介绍: 两部分构成的神经网络（前半边公用），输入是图像，输出是确定一种分布所需的参数（高斯分布为例，均值向量和协方差矩阵，不过为了减少计算量，假定各个特征也不相关，就成了方差向量了）
- 图 b: 编码器介绍: 输入时从前面的分布从 latent space 采样的特征表示，输出是高斯分布的均值，实际上也就是输出图像的预期，有一个固定的方差，采样得到输出结果。

- 图 c: 整合的小策略: 为了能让网络可以训练, 特意用 $N(0, 1)$ 这个模块来帮助表示各种高斯分布, 完成采样。



进入正题

VAE 巧妙地用了概率分布作为编解码器映射的对象, 而且选用高斯分布, 训练时每一个真实样本, 相当和表征空间的某个点为中心的一片区域对应上了, 而且损失函数还要让这个概率分布接近 $N(0, 1)$, 意味着映射的区域, 或者点云都集中在一块, 但又有一定的距离来确保准确性 ($\hat{x} - x$ 这个损失限制)

2.2 损失函数怎么设计出来的?

这里就进入了数学部分。

先定义

- x : 模型输入
- z : 中间层的表示, 在这个模型中是 latent space 采样后的结果
- $g(x)$: 输入: 同编码器输入, 输出: z 所满足的高斯分布的均值
- $h(x)$: 输入: 同编码器输入, 输出: z 所满足的高斯分布的方差
- $q_x(z)$: z 的概率密度分布, $q_x(z) \equiv \mathcal{N}(g(x), h(x)), g(x) \in G, h(x) \in H$; G, H 是函数族

再假设

- z 应该满足 $\mathcal{N}(0, 1)$
- $p(x|z)$ 也满足正态分布, 方差是固定的, 均值是通过解码器给出的。即 $p(x|z) = \mathcal{N}(f(z), c)$

$p(z|x)$ 根据上述假设, 没法直接求出来, 不过可以用正态分布函数族拟合, g 和 h 是为了整出来一个正态分布来拟合 $p(z|x)$

$$\begin{aligned}
(g^*, h^*) &= \arg \min_{(g, h) \in G \times H} KL(q_x(z), p(z|x)) \\
&= \arg \min_{(g, h) \in G \times H} \left(\mathbb{E}_{z \sim q_x} (\log q_x(z)) - \mathbb{E}_{z \sim q_x} \left(\log \frac{p(x|z)p(z)}{p(x)} \right) \right) \\
&= \arg \min_{(g, h) \in G \times H} (\mathbb{E}_{z \sim q_x} (\log q_x(z)) - \mathbb{E}_{z \sim q_x} (\log p(z)) - \mathbb{E}_{z \sim q_x} (\log p(x|z)) + \mathbb{E}_{z \sim q_x} (\log p(x))) \\
&= \arg \max_{(g, h) \in G \times H} (\mathbb{E}_{z \sim q_x} (\log p(x|z)) - KL(q_x(z), p(z))) \\
&= \arg \max_{(g, h) \in G \times H} \left(\mathbb{E}_{z \sim q_x} \left(-\frac{\|x - f(z)\|^2}{2c} \right) - KL(q_x(z), p(z)) \right)
\end{aligned}$$

又因为解码器还要满足 就是得选一个好的解码器 f 能让 $\hat{x} = x$ 的概率最高。

$$\begin{aligned}
f^* &= \arg \max_{f \in F} \mathbb{E}_{z \sim q_x^*} (\log p(x|z)) \\
&= \arg \max_{f \in F} \mathbb{E}_{z \sim q_x^*} \left(-\frac{\|x - f(z)\|^2}{2c} \right)
\end{aligned}$$

综合一下, 就是要优化这个:

$$(f^*, g^*, h^*) = \arg \max_{(f, g, h) \in F \times G \times H} \left(\mathbb{E}_{z \sim q_x} \left(-\frac{\|x - f(z)\|^2}{2c} \right) - KL(q_x(z), p(z)) \right)$$

期望就在多轮的训练中间接实现, 前后项可以加上一个权重, 结合我们的假设, 这就得到 loss 函数

$$Loss = \|x - \hat{x}\|^2 + KL[\mathcal{N}(g(x), h(x)), \mathcal{N}(0, 1)] \quad (2.1)$$

另外, 通过把 KL 散度的计算按照定义展开, 再根据高斯分布的性质化简, 可得:

$$KL(q(z|x) \parallel p(z)) = \frac{1}{2} \sum_{j=1}^J (\log(\sigma_j^2) + 1 - \sigma_j^2 - \mu_j^2) \quad (2.2)$$

Note:关于 ELBO 的文章:<https://www.zyvvvd.com/notes/study/probability/elbo/elbo/>

2.3 Code:

点击[这里](https://xiaolong-li1.github.io/Blogs/GAN/VAE.ipynb)查看代码文件 (<https://xiaolong-li1.github.io/Blogs/GAN/VAE.ipynb>)

Chapter 3

Diffusion Model

3.1 简介

3.2 前向——加噪声

在前向过程中，来自训练集的图像 x_0 会被添加 T 次噪声，使得 x_T 为符合标准正态分布。准确来说，「加噪声」并不是给上一时刻的图像加上噪声值，而是从一个均值与上一时刻图像相关的正态分布里采样出一幅新图像。如下面的公式所示， \mathbf{x}_{T-1} 是上一时刻的图像， x_T 是这一时刻生成的图像，该图像是从一个均值与有关的正态分布里采样出来的。

$$\mathbf{x}_t \sim \mathcal{N}(\mu_t(\mathbf{x}_{t-1}), \sigma_t^2 \mathbf{I})$$

绝大多数扩散模型会把这个正态分布设置为这个形式：

$$\mathbf{x}_t \sim \mathcal{N}(\sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

这个正态分布公式乍看起来很奇怪： $\sqrt{1 - \beta_t}$ 是哪里冒出来的？为什么会有这种奇怪的系数？别急，我们先来看另一个问题：假如给定 \mathbf{x}_0 ，也就是从训练集里采样出一幅图片，该怎么计算任意一个时刻 T 的噪声图像呢？

我们不妨按照公式，从开始倒推。其实可以通过一个标准正态分布的样本算出来：

$$\begin{aligned} \mathbf{x}_t &\sim \mathcal{N}(\sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}) \\ \Rightarrow \mathbf{x}_t &= \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \epsilon_{t-1}; \epsilon_{t-1} \sim \mathcal{N}(0, \mathbf{I}) \end{aligned}$$

$$\begin{aligned} \mathbf{x}_t &= \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \epsilon_{t-1} \\ &= \sqrt{1 - \beta_t} (\sqrt{1 - \beta_{t-1}} \mathbf{x}_{t-2} + \sqrt{\beta_{t-1}} \epsilon_{t-2}) + \sqrt{\beta_t} \epsilon_{t-1} \\ &= \sqrt{(1 - \beta_t)(1 - \beta_{t-1})} \mathbf{x}_{t-2} + \sqrt{(1 - \beta_t)\beta_{t-1}} \epsilon_{t-2} + \sqrt{\beta_t} \epsilon_{t-1} \end{aligned}$$

由正态分布的性质可知，均值相同的正态分布「加」在一起后，方差也会加到一起。也就是 $\mathcal{N}(0, \sigma_1^2 I)$ 与 $\mathcal{N}(0, \sigma_2^2 I)$ 合起来会得到 $\mathcal{N}(0, (\sigma_1^2 + \sigma_2^2) I)$ 。根据这一

性质，上面的公式可以化简为：

$$\begin{aligned}
& \sqrt{(1-\beta_t)(1-\beta_{t-1})}x_{t-2} + \sqrt{(1-\beta_t)\beta_{t-1}}\epsilon_{t-2} + \sqrt{\beta_t}\epsilon_{t-1} \\
&= \sqrt{(1-\beta_t)(1-\beta_{t-1})}x_{t-2} + \sqrt{(1-\beta_t)\beta_{t-1} + \beta_t}\epsilon \\
&= \sqrt{(1-\beta_t)(1-\beta_{t-1})}x_{t-2} + \sqrt{1-(1-\beta_t)(1-\beta_{t-1})}\epsilon
\end{aligned}$$

再往前推一步的话，结果是：

$$\sqrt{(1-\beta_t)(1-\beta_{t-1})(1-\beta_{t-2})}x_{t-3} + \sqrt{1-(1-\beta_t)(1-\beta_{t-1})(1-\beta_{t-2})}\epsilon$$

我们已经能够猜出规律来了，可以一直把公式推到 x_0 。令 $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$ ，则：

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$$

有了这个公式，我们就可以讨论加噪声公式为什么是 $x_t \sim \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$ 了。这个公式里的 β_t 是一个小于 1 的常数。在 DDPM 论文中， β_t 从 $\beta_1 = 10^{-4}$ 到 $\beta_T = 0.02$ 线性增长。这样， β_t 变大， α_t 也越小， $\bar{\alpha}_t$ 趋于 0 的速度越来越快。最后， $\bar{\alpha}_T$ 几乎为 0，代入 $x_T = \sqrt{\bar{\alpha}_T}x_0 + \sqrt{1 - \bar{\alpha}_T}\epsilon$ ， x_T 就满足标准正态分布了，符合我们对扩散模型的要求。上述推断可以简单描述为：加噪声公式能够从慢到快地改变原图像，让图像最终均值为 0，方差为 I。

3.3 反向——去噪声

在正向过程中，我们人为设置了 T 步加噪声过程。而在反向过程中，我们希望能够倒过来取消每一步加噪声操作，让一幅纯噪声图像变回数据集里的图像。这样，利用这个去噪声过程，我们就可以把任意一个从标准正态分布里采样出来的噪声图像变成一幅和训练数据长得差不多的图像，从而起到图像生成的目的。

现在问题来了：去噪声操作的数学形式是怎么样的？怎么让神经网络来学习它呢？数学原理表明，当 β_t 足够小时，每一步加噪声的逆操作也满足正态分布。

$$\mathbf{x}_{t-1} \sim \mathcal{N}(\tilde{\mu}_t, \tilde{\beta}_t \mathbf{I})$$

已知的：

$$\begin{aligned}
q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) &= q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_0)}{q(\mathbf{x}_t | \mathbf{x}_0)} \\
q(\mathbf{x}_t | \mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)\mathbf{I}) \\
q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})
\end{aligned}$$

首先，把其他几个式子带入贝叶斯公式的等式右边。

$$\begin{aligned}
q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) &= \frac{1}{\beta_t \sqrt{2\pi}} \exp \left(-\frac{(\mathbf{x}_t - \sqrt{1 - \beta_t} \mathbf{x}_{t-1})^2}{2\beta_t} \right) \cdot \\
&\quad \frac{1}{(1 - \bar{\alpha}_{t-1}) \sqrt{2\pi}} \exp \left(-\frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}} \mathbf{x}_0)^2}{2(1 - \bar{\alpha}_{t-1})} \right) \cdot \\
&\quad \left(\frac{1}{(1 - \bar{\alpha}_t) \sqrt{2\pi}} \exp \left(-\frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0)^2}{2(1 - \bar{\alpha}_t)} \right) \right)^{-1}
\end{aligned}$$

由于多个正态分布的乘积还是一个正态分布，我们知道 $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \sim \mathcal{N}(\tilde{\mu}_t, \epsilon_t)$

$$\begin{aligned}
\epsilon_t &= \tilde{\beta}_t \\
\tilde{\mu}_t &= \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_t \right)
\end{aligned}$$

ϵ_t 是用公式算 \mathbf{x}_t 时从标准正态分布采样出的样本，它来自公式

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_t$$