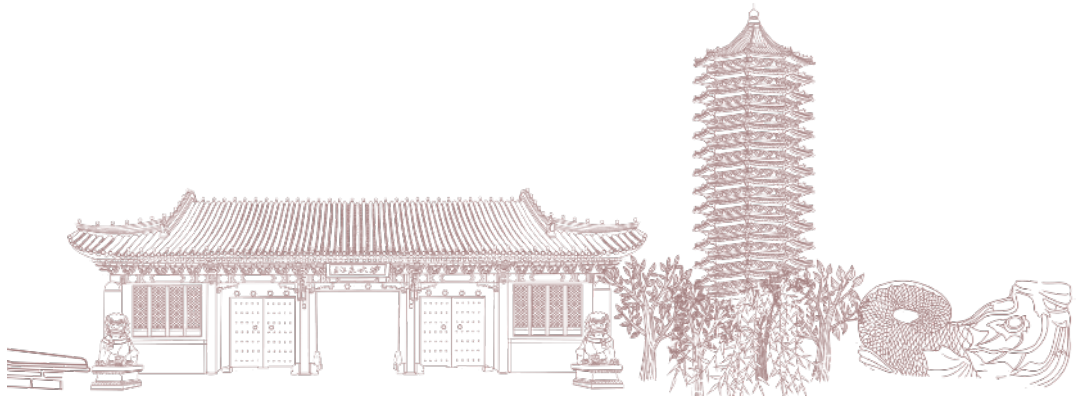
北京大学
PEKING UNIVERSITY

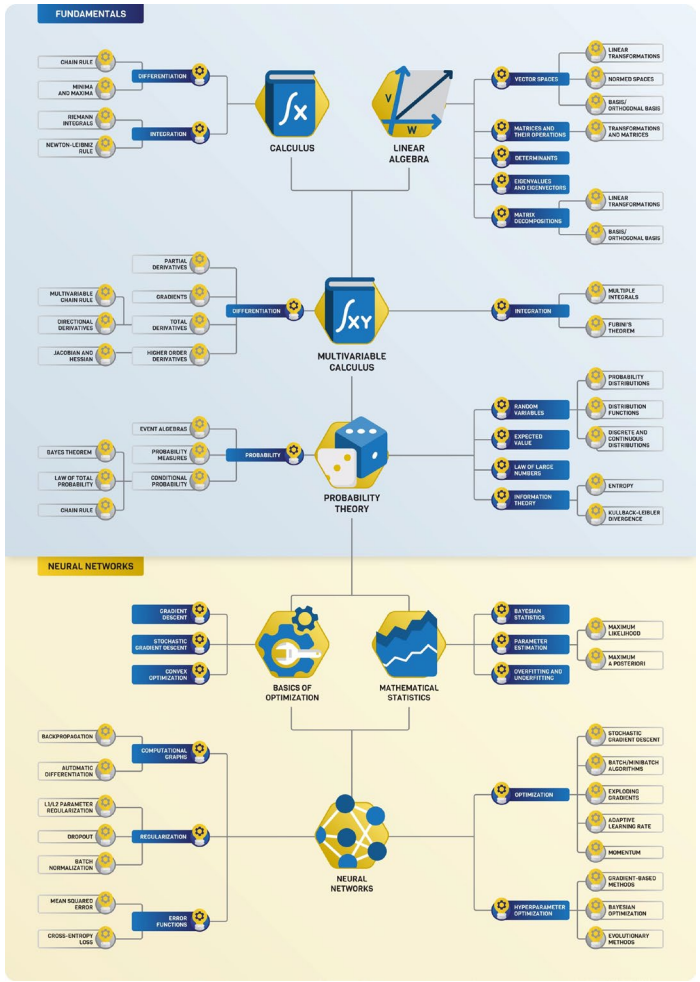
深度学习中的数学
生成模型的案例分析
Mathematics in Deep Learning
Case Study of Generative Models

阮良旺

2024.7.1



深度学习中的数学



https://tivadardanka.com/blog/roadmap-of-mathematics-for-machine-learning

22. Appendix: Mathematics for Deep Learning

Brent Werness (*Amazon*), **Rachel Hu** (*Amazon*), and authors of this book

One of the wonderful parts of modern deep learning is the fact that much of it can be understood and used without a full understanding of the mathematics below it. This is a sign that the field is maturing. Just as most software developers no longer need to worry about the theory of computable functions, neither should deep learning practitioners need to worry about the theoretical foundations of maximum likelihood learning.

But, we are not quite there yet.

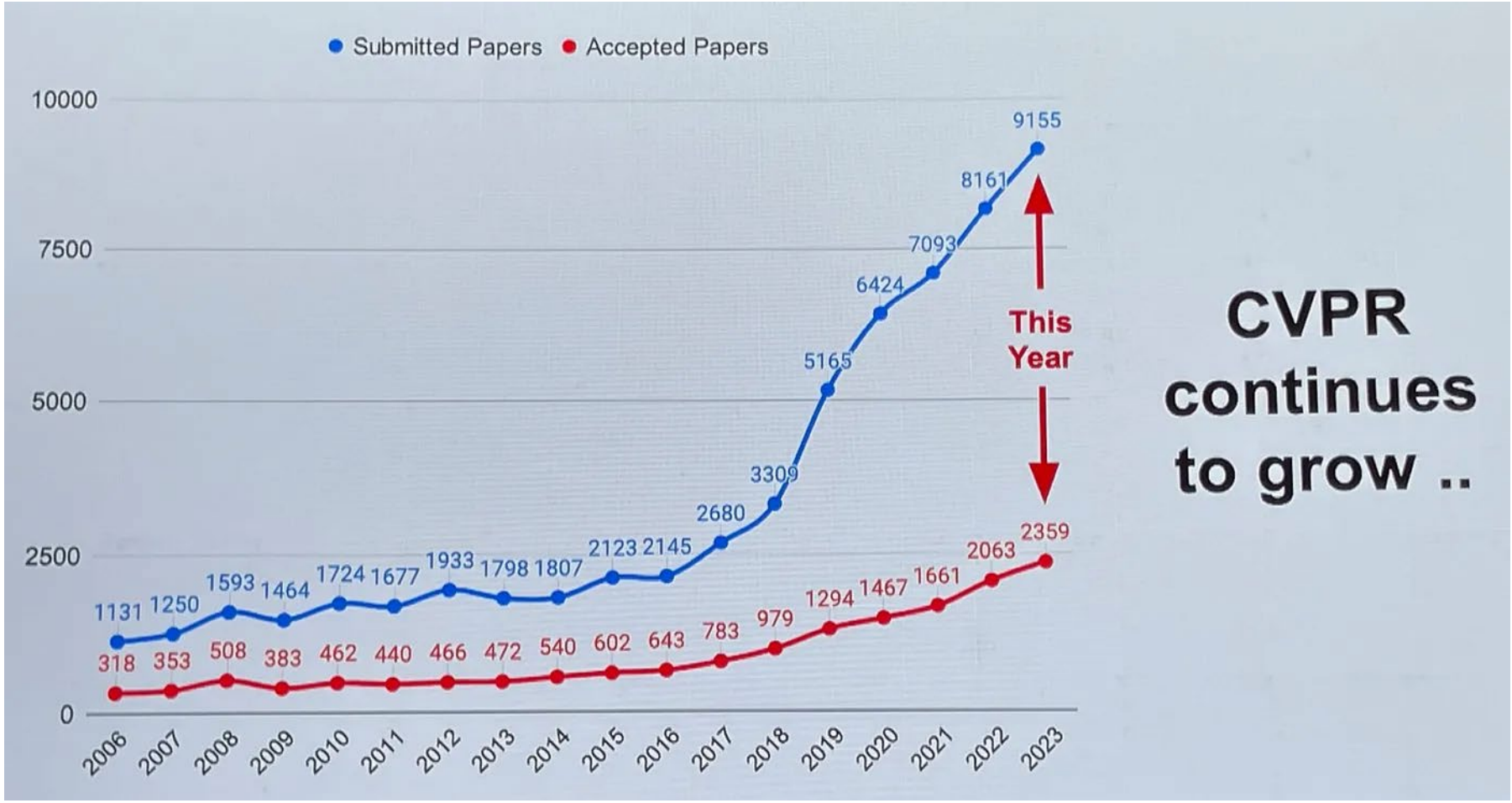
In practice, you will sometimes need to understand how architectural choices influence gradient flow, or the implicit assumptions you make by training with a certain loss function. You might need to know what in the world entropy measures, and how it can help you understand exactly what bits-per-character means in your model. These all require deeper mathematical understanding.

This appendix aims to provide you the mathematical background you need to understand the core theory of modern deep learning, but it is not exhaustive. We will begin with examining linear algebra in greater depth. We develop a geometric understanding of all the common linear algebraic objects and operations that will enable us to visualize the effects of various transformations on our data. A key element is the development of the basics of eigen-decompositions.

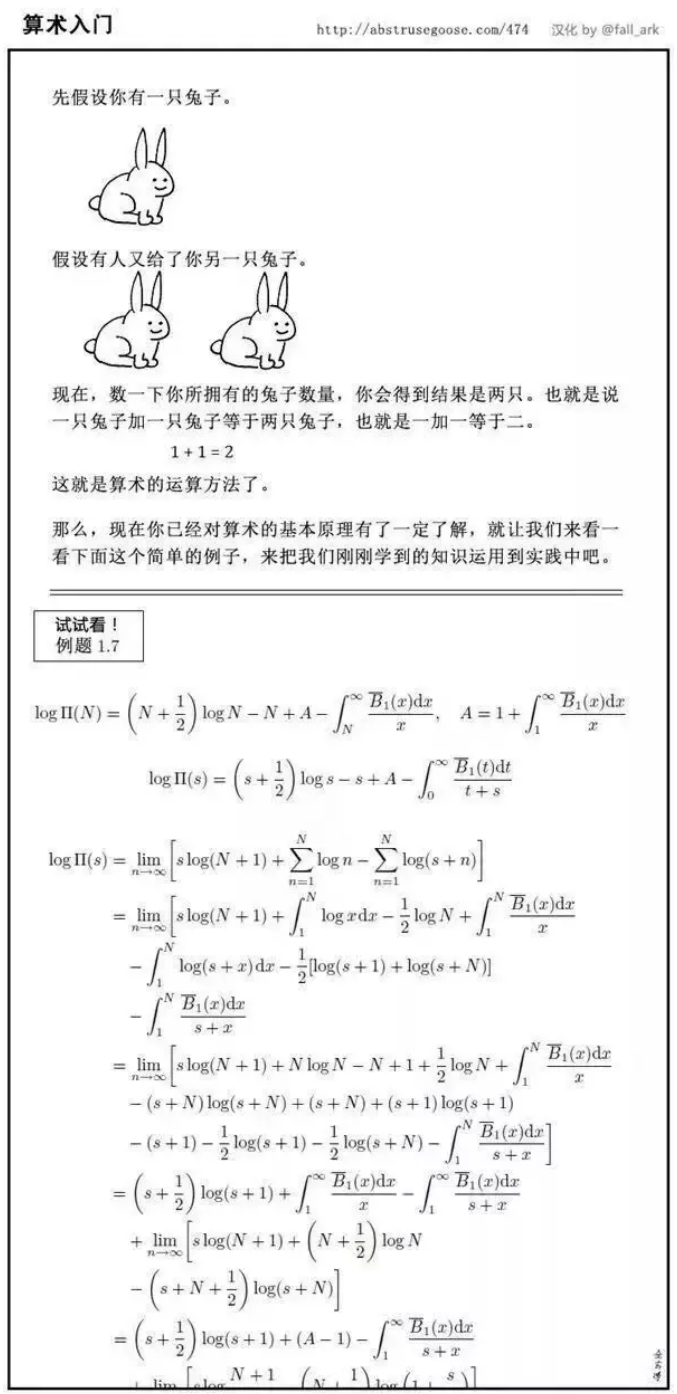
We next develop the theory of differential calculus to the point that we can fully understand why the gradient is the direction of steepest descent,

https://d2l.ai/chapter_appendix-mathematics-for-deep-learning/index.html

领域快速发展



https://medium.com/@dobko_m/cvpr-2023-summary-ad271d383404



算术入门 http://abstrusegoose.com/474 汉化 by @fall_ark

先假设你有一只兔子。

假设有人又给了你另一只兔子。

现在，数一下你所拥有的兔子数量，你会得到结果是两只。也就是说一只兔子加一只兔子等于两只兔子，也就是一加一等于二。

$$1+1=2$$

这就是算术的运算方法了。

那么，现在你已经对算术的基本原理有了一定了解，就让我们来看一看下面这个简单的例子，来把我们刚刚学到的知识运用到实践中吧。

试试看！
例题 1.7

$$\log\Pi(N)=\left(N+\frac{1}{2}\right)\log N-N+A-\int_N^\infty \frac{\overline{B}_1(x)\mathrm{d}x}{x},\quad A=1+\int_1^\infty \frac{\overline{B}_1(x)\mathrm{d}x}{x}$$

$$\log\Pi(s)=\left(s+\frac{1}{2}\right)\log s-s+A-\int_0^\infty \frac{\overline{B}_1(t)\mathrm{d}t}{t+s}$$

$$\begin{aligned}
\log\Pi(s)&=\lim_{n\to\infty}\left[s\log(N+1)+\sum_{n=1}^{N}\log n-\sum_{n=1}^{N}\log(s+n)\right]\\
&=\lim_{n\to\infty}\left[s\log(N+1)+\int_1^N\log x\mathrm{d}x-\frac{1}{2}\log N+\int_1^N\frac{\overline{B}_1(x)\mathrm{d}x}{x}\right.\\
&\quad-\int_1^N\log(s+x)\mathrm{d}x-\frac{1}{2}[\log(s+1)+\log(s+N)]\\
&\quad\left.-\int_1^N\frac{\overline{B}_1(x)\mathrm{d}x}{s+x}\right]\\
&=\lim_{n\to\infty}\left[s\log(N+1)+N\log N-N+1+\frac{1}{2}\log N+\int_1^N\frac{\overline{B}_1(x)\mathrm{d}x}{x}\right.\\
&\quad-(s+N)\log(s+N)+(s+N)+(s+1)\log(s+1)\\
&\quad\left.-(s+1)-\frac{1}{2}\log(s+1)-\frac{1}{2}\log(s+N)-\int_1^N\frac{\overline{B}_1(x)\mathrm{d}x}{s+x}\right]\\
&=\left(s+\frac{1}{2}\right)\log(s+1)+\int_1^\infty\frac{\overline{B}_1(x)\mathrm{d}x}{x}-\int_1^\infty\frac{\overline{B}_1(x)\mathrm{d}x}{s+x}\\
&\quad+\lim_{n\to\infty}\left[s\log(N+1)+\left(N+\frac{1}{2}\right)\log N\right.\\
&\quad\left.-\left(s+N+\frac{1}{2}\right)\log(s+N)\right]\\
&=\left(s+\frac{1}{2}\right)\log(s+1)+(A-1)-\int_1^\infty\frac{\overline{B}_1(x)\mathrm{d}x}{s+x}\\
&\quad+\lim\left[s\log\frac{N+1}{s+N}+\left(N+\frac{1}{2}\right)\log\left(1+\frac{s}{N}\right)\right]
\end{aligned}$$

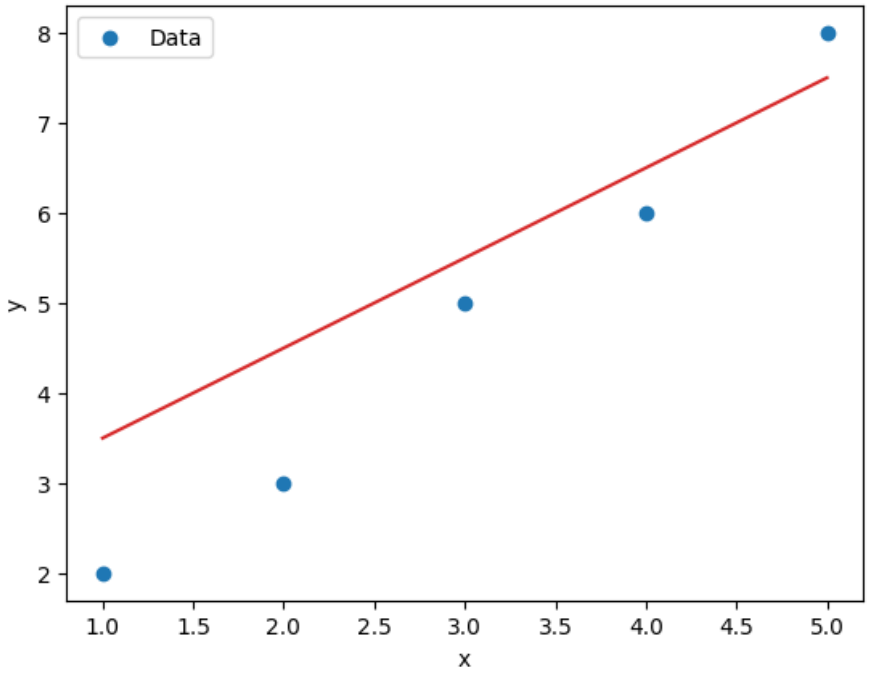
假如让写编程书的那群人来出数学书……

概率论
线性代数
优化基础
…

理解关键推导思路就好
红色部分

Transformer
NeRF
Diffusion Model
…

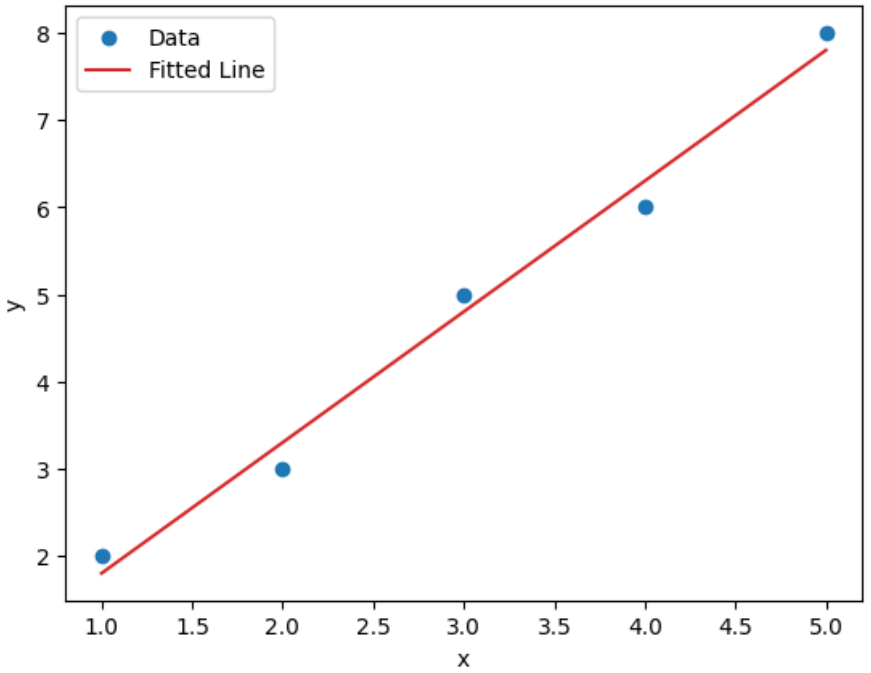
借助拟合理解深度学习



假定数据分布满足某种规律

$$y = f_\theta(x)$$

$$e.g.\ y = \theta_1 x + \theta_2$$



寻找最佳的参数最小化误差

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_i \mathrm{Dist}(f_\theta(x_i), y_i)$$

深度学习

- 由于今天我们不涉及神经网络架构的具体细节（CNN，Transformer等），可以简单认为神经网络就是一个充满待定参数θ的函数$f_\theta(x)$
- 深度学习就是针对某个损失函数L，优化参数θ

$$\theta = \arg\min L(f_\theta(x))$$

- 对应的优化算法一般是随机梯度下降算法，因此我们必须要能够得到目标关于θ的梯度

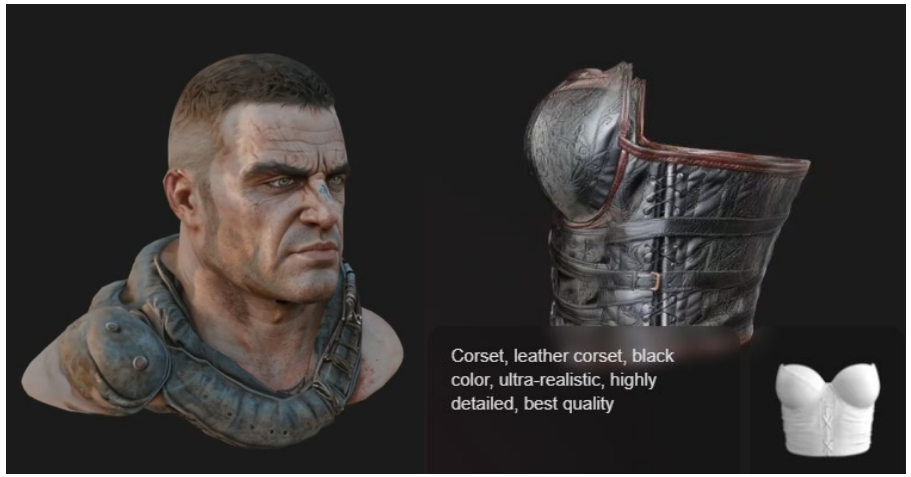
$$\frac{\partial L}{\partial \theta}$$

- 我们今天主要通过案例分析，介绍如何利用概率论为生成问题设计对应的损失函数L

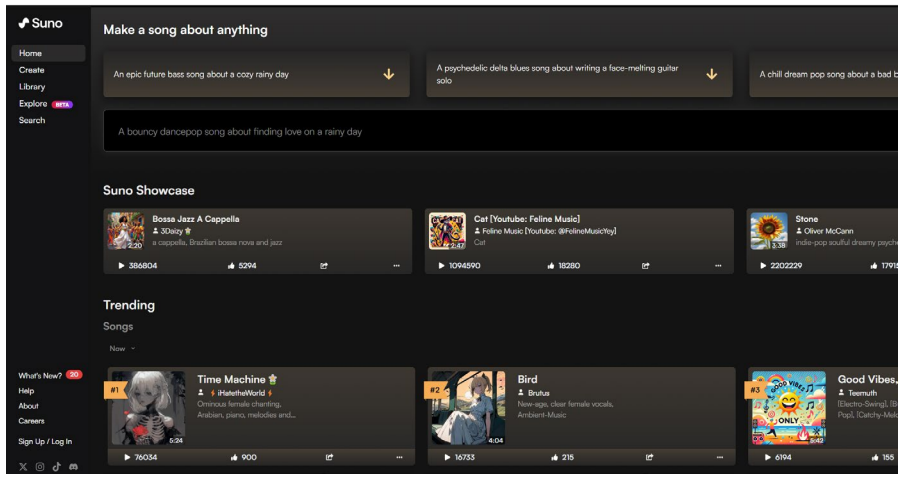
生成模型 (Generative Model)

Dalle3

Sora

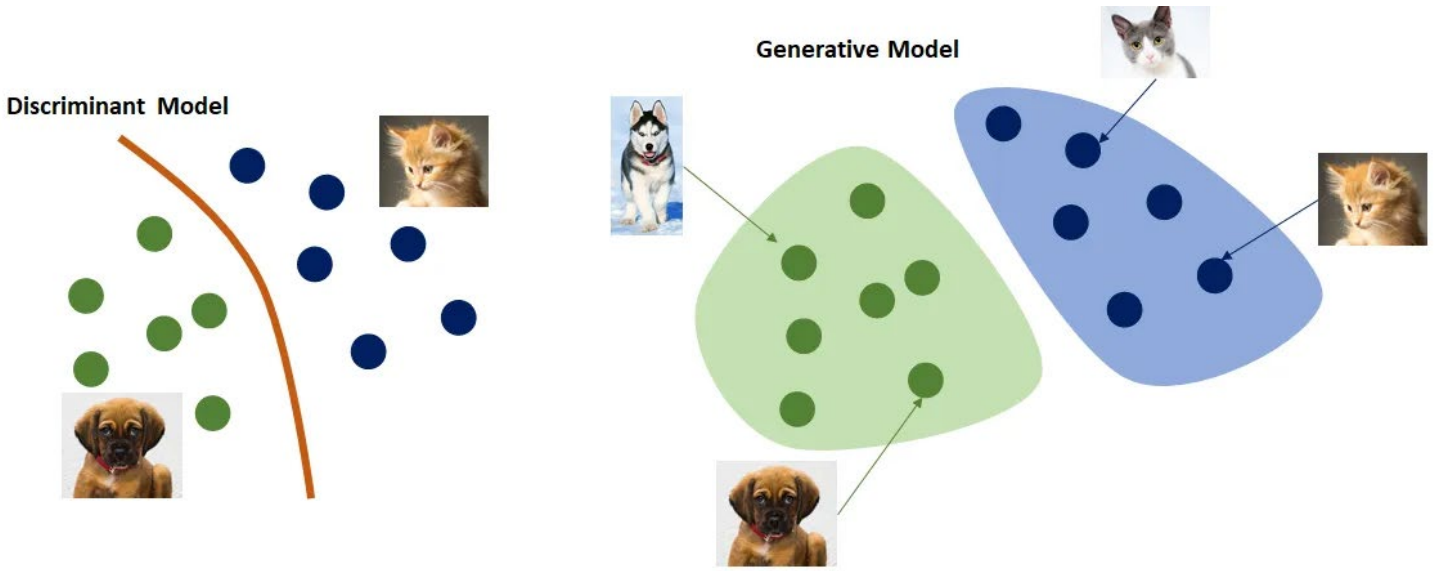


Meshy



Suno

生成模型的概率建模



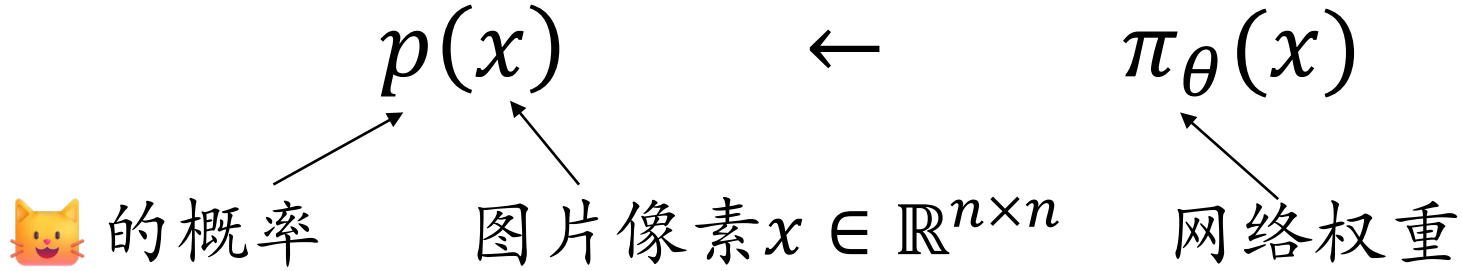
$$p(x) \quad \leftarrow \quad \pi_\theta(x)$$

🐱 的概率　　图片像素$x \in \mathbb{R}^{n \times n}$　　网络权重

只要网络能近似数据集的分布，我们就能通过随机采样进行生成

https://medium.com/@jordi299/about-generative-and-discriminative-models-d8958b67ad32

问题



- 如何度量$p(x)$与$\pi_\theta(x)$的相似程度
  - 我们并不知道$p(x)$的形式，只有数据集的采样$\{x_1, \cdots, x_n\}$
- 如何用网络表示$\pi_\theta(x)$
  - x的维度非常高，不可能直接表示

度量：KL散度 (Kullback–Leibler Divergence)

$$KL(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx = \mathbb{E}_{x\sim p}(\log p - \log q)$$

- 当$p = q$时$KL(p||q) = 0$，$p \neq q$时$KL(p||q) > 0$
- KL散度不是对称的，$KL(p||q) \neq KL(q||p)$
- KL散度在$p(x)$的分布上检查p与q的差距，集中在$p(x)$不为0的地方

最大似然估计

$$KL(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx = \mathbb{E}_{x\sim p}(\log p - \log q)$$

- 假设p是已知分布，最小化KL散度，等价于最大化$\mathbb{E}_{x\sim p}(\log q)$
- 使用采样$\{x_1, \cdots, x_n\}$近似p，得到

$$\mathbb{E}_{x\sim p}(\log q) \approx \frac{1}{n}\sum_i \log q(x_i) = \frac{1}{n}\log\left(\prod_i q(x_i)\right)$$

- 等价于最大化$\prod_i q(x_i)$：所有采样点的总概率，也即最大似然估计

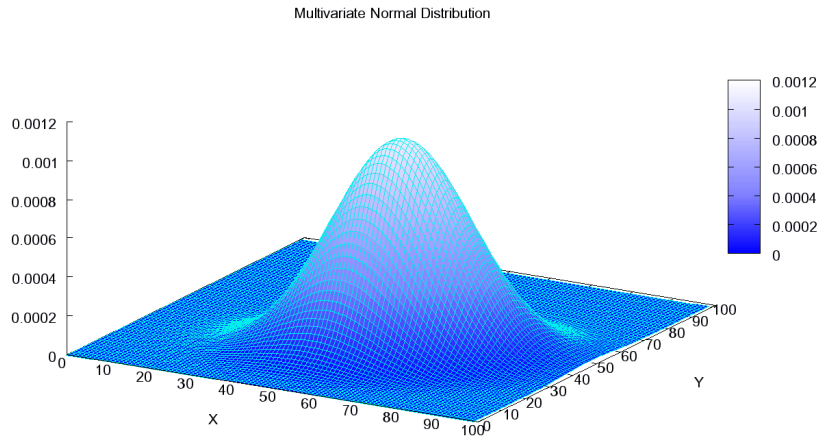
最小化观测与模型的KL散度⇔最大化观测点上模型的概率

变分自编码器 (Variational Autoencoder, VAE)

如何构造$\pi_\theta(x)$?

- 借助隐变量$z \sim \mathcal{N}(0, I)$，$\dim(z) \ll \dim(x)$

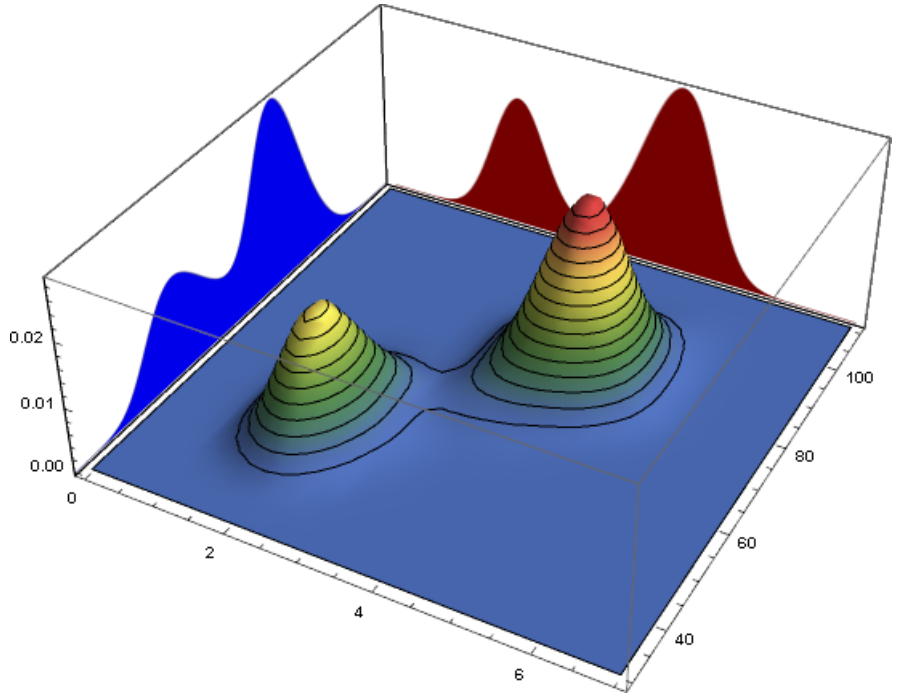
$$\mathcal{N}(\mu, \sigma^2 \mathrm{I}) = \frac{1}{\left(\sqrt{2\pi\sigma^2}\right)^n} \exp\left(-\frac{1}{2\sigma^2} \|x - \mu\|^2\right)$$



回顾条件概率与边缘分布

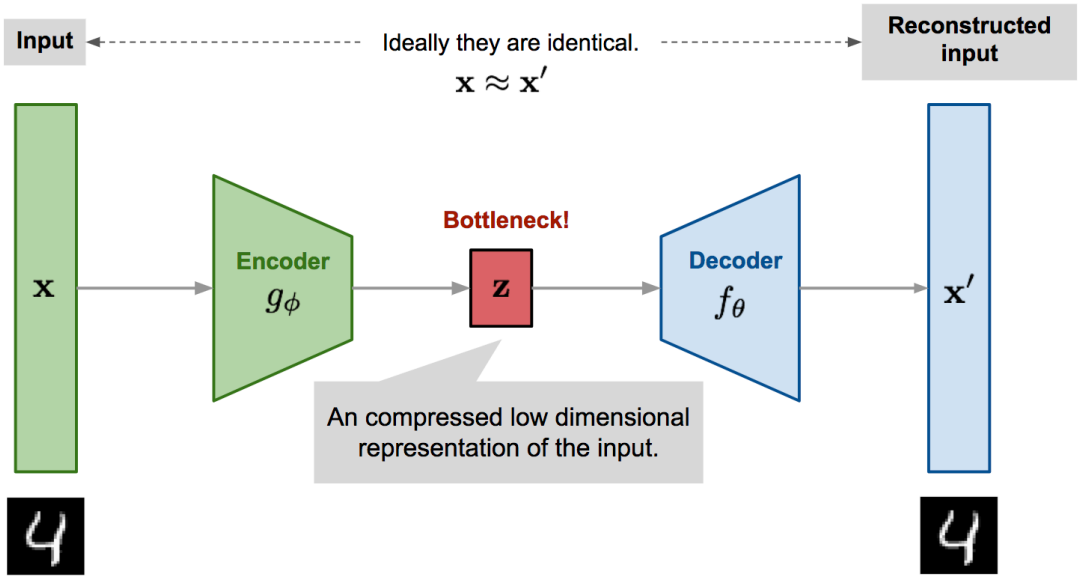
$$p(x,z) = p(x|z)p(z) = p(z|x)p(x)$$

$$p(x) = \int p(x,z)\,dz, \qquad p(z) = \int p(x,z)dx$$



变分自编码器

- $p(x)$是已知数据集分布
- $p(z)$是给定分布， $z \sim \mathcal{N}(0, I)$
- $p(x|z)$描述给定z，如何得到x，对应解码器 Decoder
- $p(z|x)$描述给定x，如何得到z，对应编码器 Encoder



变分自编码器

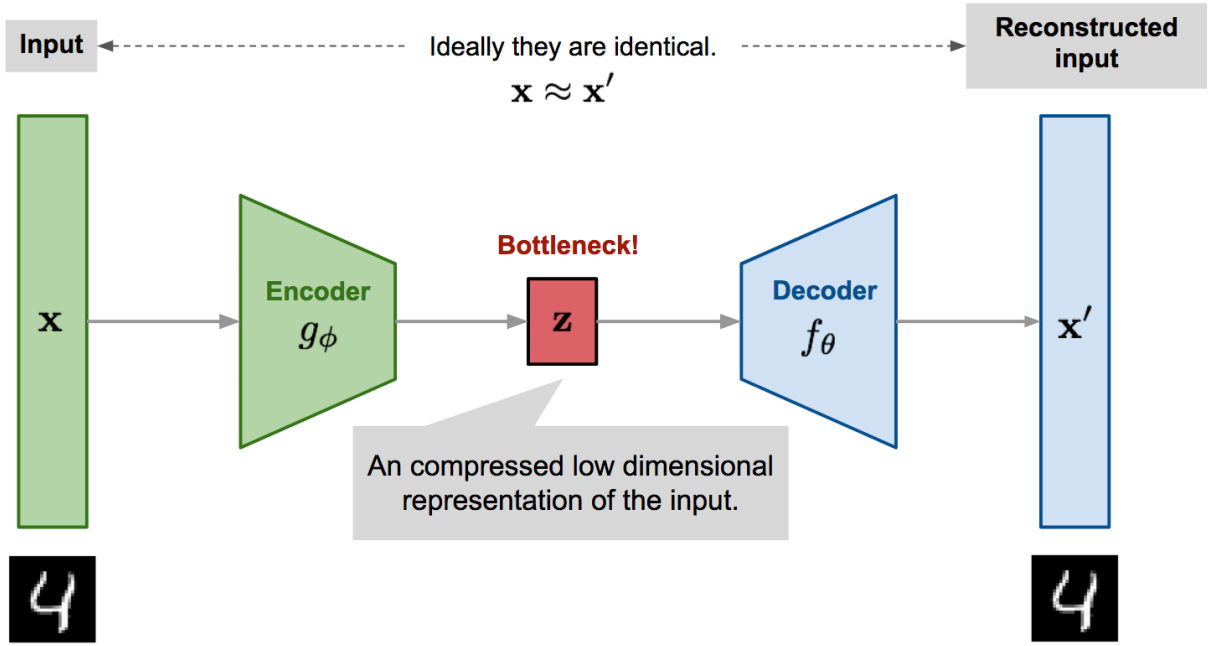
- 用网络表示的x的分布对应什么？

$$\pi_\theta(x) = \int p_\theta(x|z)\, p(z) dz = \mathbb{E}_{z\sim p(z)} p_\theta(x|z)$$

- 如果我们直接优化$D_{KL}(p(x)||\pi_\theta(x))$，也就是对于每个采样点$x_i$，我们最大化$\pi_\theta(x_i)$，这需要对$z$在$p(z) = \mathcal{N}(0, I)$中采样
- 采多少合适？由于z相当于压缩之后的数据集分布，因此可以想象我们需要采非常多的点
- 我们希望z与x之间还是有相对有序的对应关系：相似的z应该生成相似的x，而不是完全随机的映射
- 于是，我们需要借助编码器来规范z的行为

变分自编码器

- 我们构造两个用神经网络表示的条件概率$p_\theta(x|z), q_\phi(z|x)$
- $p_\theta(x|z) = \mathcal{N}(D_\theta(z), \mathrm{I})$， $D_\theta(z)$是解码器，输出x的均值
- $q_\phi(z|x) = \mathcal{N}(E_\phi(x), \sigma_\phi^2(x)\mathrm{I})$， $E_\phi(x)$和$\sigma_\phi(x)$是编码器，输出z的均值和方差

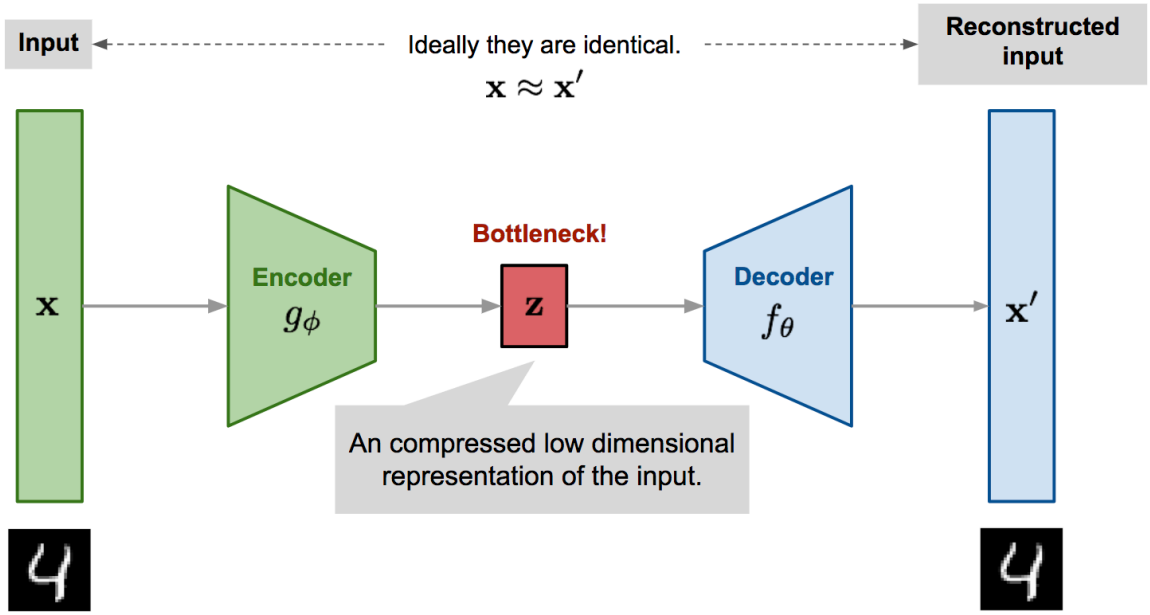


变分自编码器

- 我们有两种方式表达x和z的联合分布：

$$p(x,z) = p_\theta(x|z)p(z)$$
$$q(x,z) = q_\phi(z|x)p(x)$$

- 我们可以转而优化$D_{KL}(q(x,z)||p(x,z))$



损失函数

$$D_{KL}\big(q(x,z)\parallel p(x,z)\big)=\int q(x,z)\log\frac{q(x,z)}{p(x,z)}dxdz$$

$$=\int q_\phi(z|x)p(x)\log\frac{q_\phi(z|x)p(x)}{p(x,z)}dxdz$$

$$=\mathbb{E}_{x\sim p(x)}\Big[\mathbb{E}_{z\sim q_\phi(z|x)}[\log p(x)]\Big]+\mathbb{E}_{x\sim p(x)}\left[\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log\frac{q_\phi(z|x)}{p(x,z)}\right]\right]$$

- 第一项是由$p(x)$定义的常数，与z无关
- 第二项中$\mathbb{E}_{x\sim p(x)}$可以替换为数据集的采样，换句话说，对数据集中的每个点x_i，我们定义其损失函数为

$$L=\mathbb{E}_{z\sim q_\phi(z|x_i)}\left[\log\frac{q_\phi(z|x_i)}{p(x_i,z)}\right]$$

证据下界 (Evidence Lower Bound)

$$L = \mathbb{E}_{z\sim q_\phi(z|x_i)}\left[\log \frac{q_\phi(z|x_i)}{p(x_i,z)}\right]$$

$$\log p(x) = \log \int p(x,z)dz = \log \int p(x,z)\frac{q(z)}{q(z)}dz$$

$$= \log \mathbb{E}_{z\sim q(z)} \frac{p(x,z)}{q(z)} \geq \mathbb{E}_{z\sim q(z)} \log \frac{p(x,z)}{q(z)}$$

$$\text{ELBO} = \mathbb{E}_{z\sim q(z)} \log \frac{p(x,z)}{q(z)}$$

最小化L，就是在最大化ELBO，也就是最大化$\log p(x)$的下界，对应最大似然估计

损失函数

$$L = \mathbb{E}_{z\sim q_\phi(z|x_i)}\left[\log \frac{q_\phi(z|x_i)}{p(x_i,z)}\right]$$

$$= \mathbb{E}_{z\sim q_\phi(z|x_i)}\left[\log \frac{q_\phi(z|x_i)}{p_\theta(x_i|z)p(z)}\right]$$

$$= \mathbb{E}_{z\sim q_\phi(z|x_i)}\left[-\log p_\theta(x_i|z) + \log \frac{q_\phi(z|x_i)}{p(z)}\right]$$

$$L = \mathbb{E}_{z\sim q_\phi(z|x_i)}[-\log p_\theta(x_i|z)] + D_{KL}(q_\phi(z|x_i) \parallel p(z))$$

损失函数

$$L = \mathbb{E}_{z\sim q_\phi(z|x_i)}[-\log p_\theta(x_i|z)] + D_{KL}(q_\phi(z|x_i) \parallel p(z))$$

- 我们定义了$p_\theta(x|z) = \mathcal{N}(D_\theta(z), \mathrm{I})$，因此$-\log p_\theta(x_i|z) = \frac{1}{2}\|x_i - D_\theta(z)\|^2 + C$，所以第一项可以理解为生成结果$D_\theta(z)$与数据集中的$x_i$的L2距离
- z需要在$q_\phi(z|x_i)$中采样来求期望，但是由于$q_\phi(z|x) = \mathcal{N}(E_\phi(x), \sigma_\phi^2(x)\mathrm{I})$是由编码器定义的映射关系，我们可以认为$\sigma_\phi(x)$比较小，也即$x$只会对应到一小片区域的$z$，所以一般采一个样就可以
- 因此第一项对应为

$$L_1 = \frac{1}{2}\|x_i - D_\theta(z_i)\|^2, z_i \sim q_\phi(z|x_i)$$

重参数化 (Reparameterization)

$$L_1 = \frac{1}{2}\|x_i - D_\theta(z_i)\|^2, z_i \sim q_\phi(z|x_i)$$

- 注意最终我们需要通过随机梯度下降优化L_1，但是z_i的采样过程是不可导的，也就是我们不知道$\frac{\partial z}{\partial \phi}$
- 但是$q_\phi(z|x_i) = \mathcal{N}(E_\phi(x_i), \sigma_\phi^2(x_i)\mathrm{I})$是正态分布，我们可以对采样进行重参数化

$$z_i = E_\phi(x_i) + \sigma_\phi(x_i)\varepsilon_i, \varepsilon_i \sim \mathcal{N}(0, \mathrm{I})$$

- 其中ε_i是从标准正态中的采样，与网络参数无关，现在z_i与$E_\phi(x_i)$和$\sigma_\phi(x_i)$有算术关系，这样我们就能正确反传梯度了

损失函数

$$L = \mathbb{E}_{z\sim q_\phi(z|x_i)}[-\log p_\theta(x_i|z)] + D_{KL}(q_\phi(z|x_i) \parallel p(z))$$

- 第二项最小化$q_\phi(z|x_i)$与$p(z) = \mathcal{N}(0, I)$之间的差异，也就是我们希望$q_\phi(z|x_i)$尽可能接近标准正态
- 如果没有这一项，那么编码器完全可以对于固定的x_i输出固定的z_i来最小化数据集上的损失函数，这样模型就没有了泛化能力

正态分布的KL散度

定理：对于两个正态分布$\mathrm{P}_1 = \mathcal{N}(\mu_1, \sigma_1^2\mathrm{I})$和$\mathrm{P}_2 = \mathcal{N}(\mu_2, \sigma_2^2\mathrm{I})$，其KL散度为

$$D_{KL}(P_1||P_2) = \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2}{2\sigma_2^2} + \frac{\|\mu_1 - \mu_2\|^2}{2\sigma_2^2} - \frac{1}{2}$$

损失函数

$$L = \mathbb{E}_{z\sim q_\phi(z|x_i)}[-\log p_\theta(x_i|z)] + D_{KL}(q_\phi(z|x_i) \parallel p(z))$$

- 第二项最小化$q_\phi(z|x_i)$与$p(z) = \mathcal{N}(0, I)$之间的差异，也就是我们希望$q_\phi(z|x_i)$尽可能接近标准正态
- 如果没有这一项，那么编码器完全可以对于固定的x_i输出固定的z_i来最小化数据集上的损失函数，这样模型就没有了泛化能力
- 由于$q_\phi(z|x_i) = \mathcal{N}(E_\phi(x_i), \sigma_\phi^2(x_i)\mathrm{I})$也是正态分布，我们可以直接显式计算二者的KL散度，得到得到损失函数的第二项，其中d是z的维度

$$L_2 = \frac{1}{2}(\|E_\phi(x_i)\|^2 + d\sigma_\phi^2(x_i) - 2d\log\sigma_\phi(x_i))$$

训练与生成

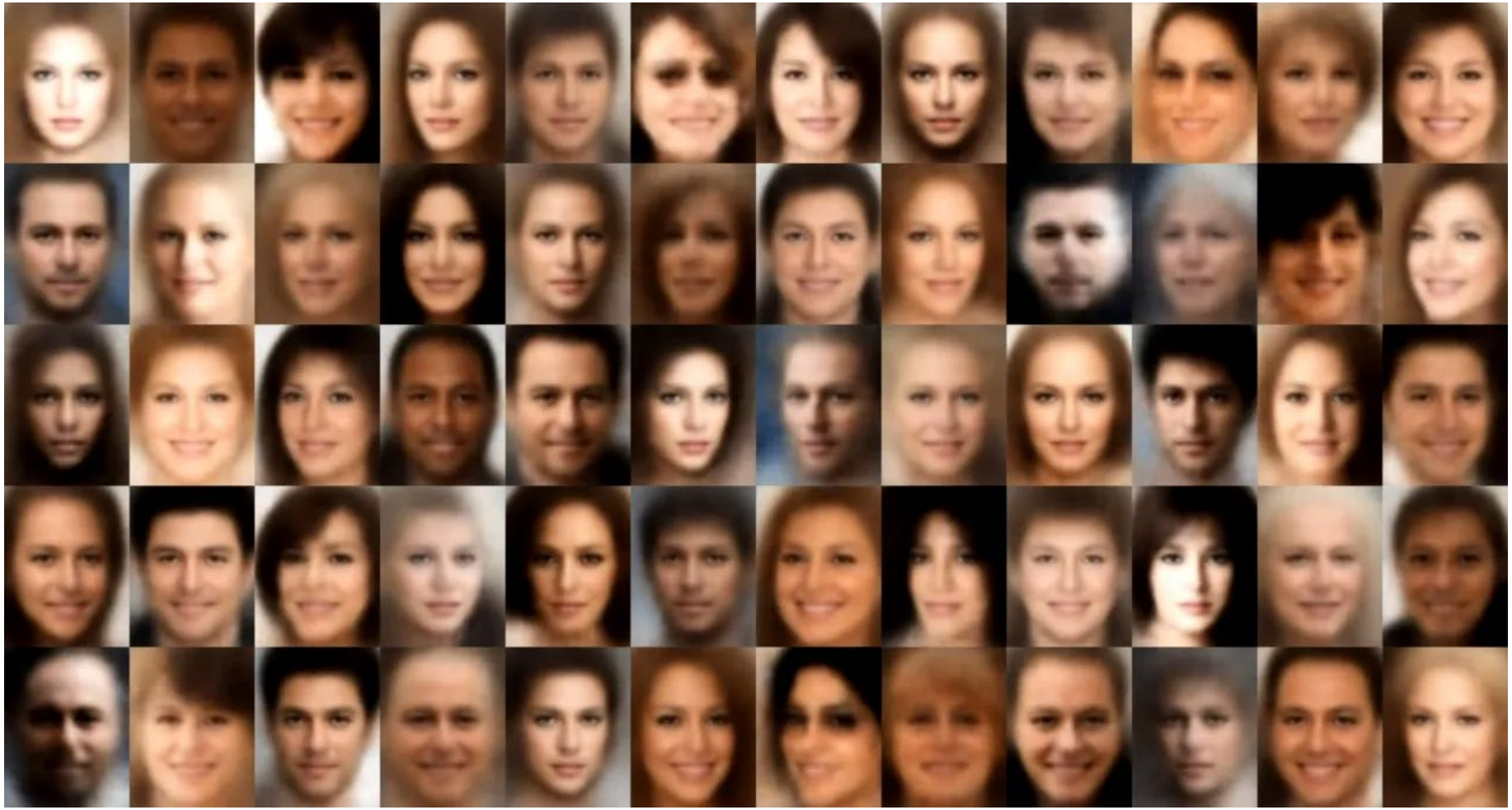
$$L = \frac{1}{2}\|x_i - D_\theta(z_i)\|^2 + \frac{1}{2}\left(\left\|E_\phi(x_i)\right\|^2 + d\sigma_\phi^2(x_i) - 2d\log\sigma_\phi(x)\right)$$

$$z_i = E_\phi(x_i) + \sigma_\phi(x_i)\varepsilon_i, \varepsilon_i \sim \mathcal{N}(0, \mathrm{I})$$

- 训练时，对于每个数据集中的x_i，经过编码器得到$E_\phi(x_i)$和$\sigma_\phi(x_i)$，采样得到z_i，经过解码器得到$D_\theta(z_i)$，计算上面的损失函数并反传优化网络参数
- 生成时，在$\mathcal{N}(0, I)$中随机采样得到z，经过解码器得到$D_\theta(z)$作为网络生成结果

生成结果

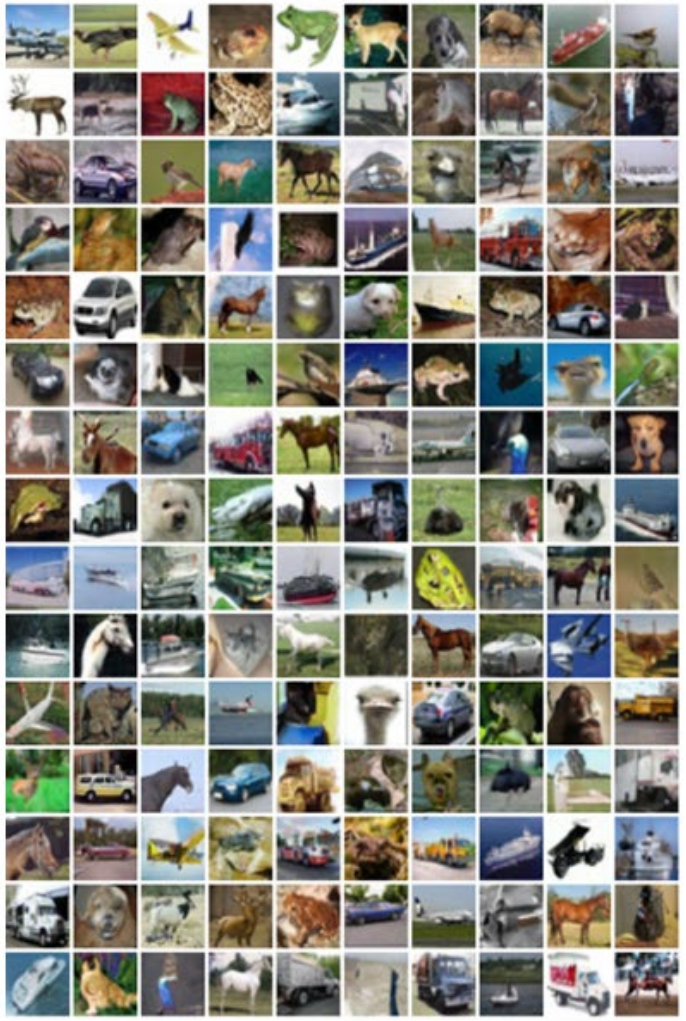
“糊”



https://github.com/wojciechmo/vae

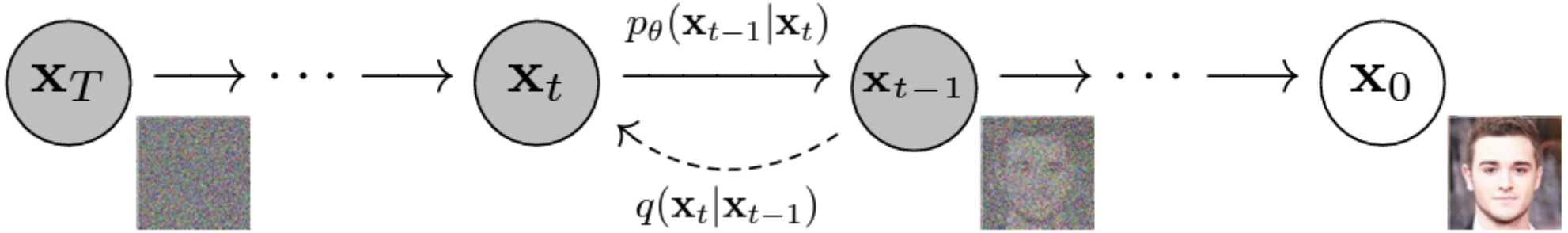
扩散模型 (Diffusion Model)

“不糊”



从VAE理解扩散模型

- 在VAE中，我们构建了数据集分布$p(x)$与隐空间分布$p(z)$之间的双向映射关系
- 扩散模型定义了一系列的隐变量，$x_0, \cdots, x_T$，x_0对应数据集，$x_T = z$对应标准正态分布
- 通过多步过程，扩散模型的表达能力比VAE更强，生成的结果质量更高



开始推导之前

- 所有的q都表示已知的概率，不包含任何需要学习的参数，要么有定义，要么可以使用条件概率公式推导出来具体形式（在下面的推导中都是已知高斯分布）
- 所有p_θ表示的概率都是神经网络表示的概率（在下面的推导中都是由神经网络输出均值和方差的高斯分布）

前向过程：马尔可夫链

- 从一张图片出发，不断增加高斯噪声，直到完全变成纯噪声
- 马尔可夫链：第t步的结果x_t只与x_{t-1}有关，与$x_0, \cdots, x_{t-2}$无关

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

- β_t是每步给定的常数，一般$\beta_t \ll 1$

前向过程：马尔可夫链

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

- 使用重参数化技巧，并令$\alpha_t = 1 - \beta_t$，可以改写为

$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\varepsilon_{t-1}, \qquad \varepsilon_{t-1} \sim \mathcal{N}(0, \mathrm{I})$$

- 我们可以继续展开x_{t-1}

$$x_t = \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-1}}\varepsilon_{t-2}\right) + \sqrt{1-\alpha_t}\varepsilon_{t-1}$$

高斯分布相加

定理：如果$x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2 I), x_2 \sim \mathcal{N}(\mu_2, \sigma_2^2 I)$是两个相互独立的高斯随机变量，那么随机变量$y = x_1 + x_2$也满足高斯分布，并且均值为$\mu_1 + \mu_2$，方差为$\sqrt{\sigma_1^2 + \sigma_2^2}$

$$x_1 + x_2 \sim \mathcal{N}(\mu_1 + \mu_2, (\sigma_1^2 + \sigma_2^2) I)$$

前向过程：马尔可夫链

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

- 使用重参数化技巧，并令$\alpha_t = 1 - \beta_t$，可以改写为

$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\varepsilon_{t-1}, \qquad \varepsilon_{t-1} \sim \mathcal{N}(0, \mathrm{I})$$

- 我们可以继续展开x_{t-1}

$$x_t = \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-1}}\varepsilon_{t-2}\right) + \sqrt{1-\alpha_t}\varepsilon_{t-1}$$

- 注意ε_{t-2}和ε_{t-1}是相互独立的高斯随机变量，因此可以写为

$$\sqrt{\alpha_t(1-\alpha_{t-1})}\varepsilon_{t-2} + \sqrt{1-\alpha_t}\varepsilon_{t-1} = \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\varepsilon}_{t-2}$$

$$\bar{\varepsilon}_{t-2} \sim \mathcal{N}(0, \mathrm{I})$$

$$x_t = \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\varepsilon}_{t-2}$$

前向过程：马尔可夫链

$$x_t = \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t}\varepsilon_{t-1}$$
$$x_t = \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \sqrt{1-\alpha_t \alpha_{t-1}}\bar{\varepsilon}_{t-2}$$

- 注意到二者形式上的相似性，意味着两步加噪的过程可以看成是一步加了更大的噪声
- 这个展开过程可以持续下去，得到x_t（完全噪声）和x_0（真实图片）之间的关系，其中$\bar{\alpha}_t = \alpha_t \alpha_{t-1} \cdots \alpha_1$

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon_t, \varepsilon_t \sim \mathcal{N}(0, \mathrm{I})$$

前向过程总结

- 前向过程是一个给定的马尔可夫过程，状态转移定义为

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

- 我们可以直接从x_0一步得到x_t的采样结果

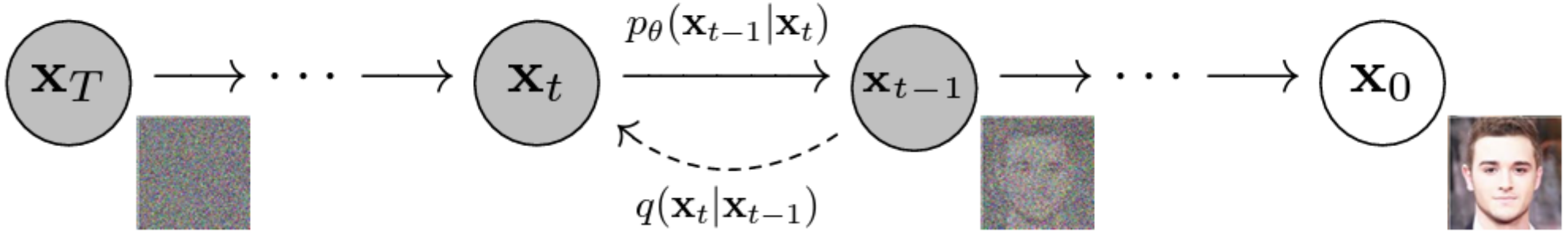
$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon_t, \varepsilon_t \sim \mathcal{N}(0, \mathrm{I})$$

反向过程

- 反向过程即是从完全噪声x_T出发，逐步恢复x_0的过程
- 在每一步，我们使用神经网络来拟合反向概率$p_\theta(x_{t-1}|x_t)$
- 和VAE中一样，我们假设$p_\theta(x_{t-1}|x_t)$是正态分布，由网络输出分布的均值

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(\mu_\theta(x_t, t), \sigma_t^2 \mathrm{I})$$

- 其中方差σ_t一般提前给定



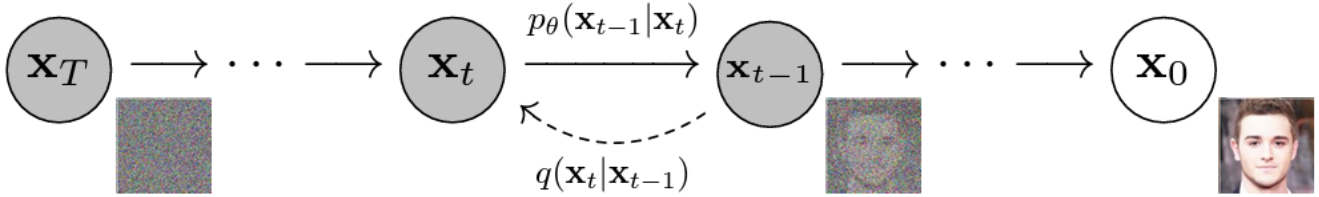
损失函数

- 对于VAE，我们优化如下ELBO损失函数，其中z是隐变量，x是从数据集中的采样

$$L = \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p(x,z)}\right]$$

- 对于扩散模型，我们优化相同的目标，只不过现在隐变量包含$x_T,\cdots,x_1$，而数据集采样对应x_0，因此扩散模型的损失函数可以定义为

$$L = \mathbb{E}_{q(x_1\cdots x_T|x_0)}\left[\log \frac{q(x_1\cdots x_T|x_0)}{p(x_0\cdots x_T)}\right]$$



损失函数

$$L = \mathbb{E}_{q(x_1 \cdots x_T | x_0)} \left[\log \frac{q(x_1 \cdots x_T | x_0)}{p(x_0 \cdots x_T)} \right]$$

- 我们可以使用条件概率将上面的概率拆成每步概率的乘积

$$L = \mathbb{E}_{q(x_1 \cdots x_T | x_0)} \left[\log \frac{q(x_1 | x_0) q(x_2 | x_1) \cdots q(x_T | x_{T-1})}{p_\theta(x_0 | x_1) p_\theta(x_1 | x_2) \cdots p_\theta(x_{T-1} | x_T) p(x_T)} \right]$$

- 分子的每一项$q(x_t | x_{t-1})$对应前向过程的已知正态分布
- 分母的每一项$p_\theta(x_t | x_{t-1})$对应反向过程神经网络拟合的概率，也是正态分布，$p(x_T)$是加上所有噪声的结果，可以认为是标准正态
- 问题在于前面的期望，如何减少采样的次数?

损失函数

$$L = \mathbb{E}_{q(x_1 \cdots x_T | x_0)} \left[\log \frac{q(x_1|x_0) q(x_2|x_1) \cdots q(x_T|x_{T-1})}{p_\theta(x_0|x_1) p_\theta(x_1|x_2) \cdots p_\theta(x_{T-1}|p_T) p(x_T)} \right]$$

- 在VAE中，我们也需要处理损失函数中隐变量z的期望，我们的解决方法是引入编码器，找到每个数据集点x_i对应的z
- 在扩散模型中，我们解决这个问题的方法是在$q(x_t|x_{t-1})$中引入生成结果x_0，并考虑下面形式的条件概率

$$q(x_t|x_{t-1}) = q(x_t|x_{t-1}, x_0) = \frac{q(x_{t-1}|x_t, \boldsymbol{x_0}) q(x_t|\boldsymbol{x_0})}{q(x_{t-1}|\boldsymbol{x_0})}$$

- 第一个等号利用了马尔可夫过程的无后效性，第二个等号利用了贝叶斯公式，这样我们就将所有$q(x_t|x_{t-1})$都写成了相对于x_0的条件概率

损失函数

$$L = \mathbb{E}_{q(x_1 \cdots x_T | x_0)} \left[\log \frac{q(x_1|x_0) q(x_2|x_1) \cdots q(x_T|x_{T-1})}{p_\theta(x_0|x_1) p_\theta(x_1|x_2) \cdots p_\theta(x_{T-1}|p_T) p(x_T)} \right]$$

$$q(x_t|x_{t-1}) = q(x_{t-1}|x_t, x_0) \frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}$$

- 我们将第二个等式带入损失函数，并注意蓝色部分在连乘中可以相互消除，得到log中的形式为

$$\frac{q(x_T|x_0)}{p(x_T) p_\theta(x_0|x_1)} \prod_{t=2}^{T} \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)}$$

- 这个结果可以变成最终的求和形式的损失函数

损失函数

$$L = \mathbb{E}_{q(x_1 \cdots x_T|x_0)} \left[\log \frac{q(x_T|x_0)}{p(x_T) p_\theta(x_0|x_1)} \prod_{t=2}^{T} \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} \right]$$

$$= \mathbb{E}_{q(x_1|x_0)} \log p_\theta(x_0|x_1) + \mathbb{E}_{q(x_T|x_0)} \log \frac{q(x_T|x_0)}{p(x_T)}$$

$$+ \sum_{t=2}^{T} \mathbb{E}_{q(x_{t-1}, x_t|x_0)} \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)}$$

$$= \mathbb{E}_{q(x_1|x_0)} \log p_\theta(x_0|x_1) + D_{KL}\big(q(x_T|x_0) \parallel p(x_T)\big)$$

$$+ \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)} D_{KL}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t))$$

损失函数

$$L = \mathbb{E}_{q(x_1|x_0)} \log p_\theta(x_0|x_1) + D_{KL}\big(q(x_T|x_0) \parallel p(x_T)\big)$$
$$+ \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)} D_{KL}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t))$$

- 第一项对应从x_0开始的第一步加噪的过程，可以采样一次来估计
- 第二项中，$q(x_T|x_0)$是在加噪过程中最终x_T的分布，$p(x_T)$就是标准正态，这一项不包含任何的网络参数θ，因此可以忽略

损失函数

$$L = \mathbb{E}_{q(x_1|x_0)} \log p_\theta(x_0|x_1) + D_{KL}\big(q(x_T|x_0) \parallel p(x_T)\big)$$
$$+ \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)} D_{KL}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t))$$

- 剩下的就是在反向过程中每步的损失函数，可以理解为我们希望网络概率$p_\theta(x_{t-1}|x_t)$与$q(x_{t-1}|x_t, x_0)$尽可能接近

$$L_t = \mathbb{E}_{q(x_t|x_0)} D_{KL}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t))$$

- 由于在前向过程中我们可以由x_0直接得到x_t，因此实际训练时我们会将L中的求和改成对t的期望，也就是对于每个数据集中的图片，我们只均匀随机采样一个t，在x_t的结果上优化L_t，而不会经过整个前向过程

损失函数

$$L_t = \mathbb{E}_{q(x_t|x_0)} D_{KL}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t))$$

- $q(x_{t-1}|x_t, x_0)$是不包含网络参数的概率，理论上我们可以根据前向过程推出取具体的形式

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

- 可以发现右手边的每一项都是已知的正态分布，直接代入计算，我们就可以发现$q(x_{t-1}|x_t, x_0)$也是一个正态分布$\mathcal{N}(\tilde{\mu}, \tilde{\beta}_t \mathrm{I})$

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(\tilde{\mu}, \tilde{\beta}_t \mathrm{I})$$

$$\tilde{\mu}(x_t, x_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0, \qquad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

损失函数

$$L_t = \mathbb{E}_{q(x_t|x_0)} D_{KL}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t))$$

- 我们确定了$q(x_{t-1}|x_t, x_0)$是已知的正态分布， $p_\theta(x_{t-1}|x_t)$是网络输出的正态分布

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(\mu_\theta(x_t, t), \sigma_t^2 \mathrm{I})$$

- 二者的KL散度是有解析表达式的

正态分布的KL散度

定理：对于两个正态分布$\mathrm{P}_1 = \mathcal{N}(\mu_1, \sigma_1^2\mathrm{I})$和$\mathrm{P}_2 = \mathcal{N}(\mu_2, \sigma_2^2\mathrm{I})$，其KL散度为

$$D_{KL}(P_1||P_2) = \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2}{2\sigma_2^2} + \frac{\|\mu_1 - \mu_2\|^2}{2\sigma_2^2} - \frac{1}{2}$$

- 如果只关注均值，我们会发现KL散度对应着均值的L2距离平方

损失函数

$$L_t = \mathbb{E}_{q(x_t|x_0)} D_{KL}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t))$$

- 在简化的情形下，我们可以直接取KL散度近似为两个分布均值的L2差距

$$L_t \approx \mathbb{E}_{q(x_t|x_0)} \|\tilde{\mu}(x_t, x_0) - \mu_\theta(x_t, t)\|^2$$

- 而对于其中在$q(x_t|x_0)$的分布下求期望，我们可以利用之前给出的关系，直接对x_t进行采样

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \varepsilon_t, \varepsilon_t \sim \mathcal{N}(0, \mathrm{I})$$

- 到这里，我们就可以计算损失函数并梯度回传了，不过扩散模型算法的最终形式还进行了一次参数变换

损失函数

$$L_t \approx \mathbb{E}_{q(x_t|x_0)}\|\tilde{\mu}(x_t, x_0) - \mu_\theta(x_t, t)\|^2$$

$$\tilde{\mu}(x_t, x_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0$$

- 我们可以将$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon_t$代入$q(x_{t-1}|x_t, x_0)$的形式中，消去$x_0$，得到

$$\tilde{\mu}(x_t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_t)$$

- 模仿这个形式，我们可以将网络的输出重新定义为$\varepsilon_\theta(x_t, t)$，而不是$p_\theta(x_{t-1}|x_t)$的均值$\mu_\theta(x_t, t)$，二者关系为

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

损失函数

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

- 进行了这样的变换之后，最终的损失函数形式为

$$L_t = \mathbb{E}_{q(x_t|x_0)}\|\varepsilon_t - \varepsilon_\theta(x_t, t)\|^2$$

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

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

参考链接

- A Brief Introduction to Generative Models
- https://blog.rexking6.top/2019/06/09/变分自编码器VAE
- https://lilianweng.github.io/posts/2021-07-11-diffusion-models
- https://speech.ee.ntu.edu.tw/~hylee/ml/2023-spring.php

北京大学
PEKING UNIVERSITY

谢谢