Variational AutoEncoder
变分自动编码器（VAE）

Name: 王若琪

Date: 20210522

目录

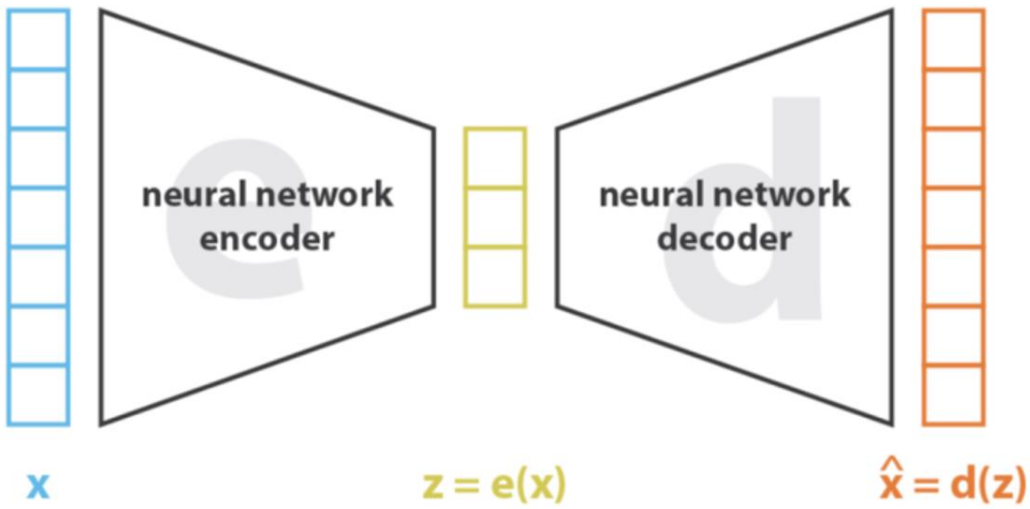
VAE概述

变分自编码器（Variational Auto-Encoders，VAE）作为深度生成模型的一种形式，是由 Kingma 等人于 2014 年提出的基于变分贝叶斯（**Variational Bayes，VB**）推断的生成式网络结构。

与传统的自编码器通过数值的方式描述潜在空间不同，**它以概率的方式描述对潜在空间的观察**，在数据生成方面表现出了巨大的应用价值。

VAE一经提出就迅速获得了深度生成模型领域广泛的关注，并和生成对抗网络（Generative Adversarial Networks，GAN）被视为无监督式学习领域最具研究价值的方法之一，在深度生成模型领域得到越来越多的应用。

自动编码器(AutoEncoder)



$$\text{loss} = \|x - \hat{x}\|^2 = \|x - d(z)\|^2 = \|x - d(e(x))\|^2$$

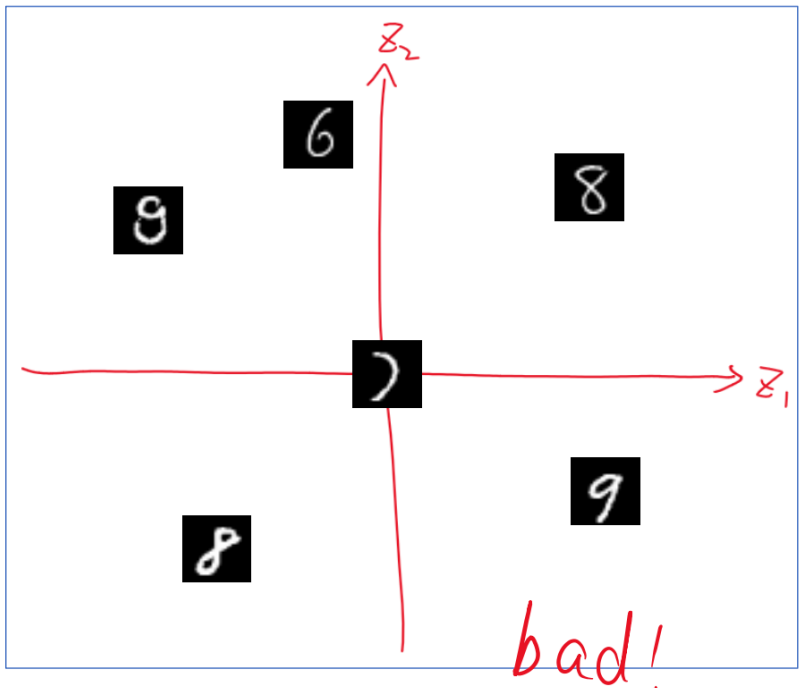
Why VAE?

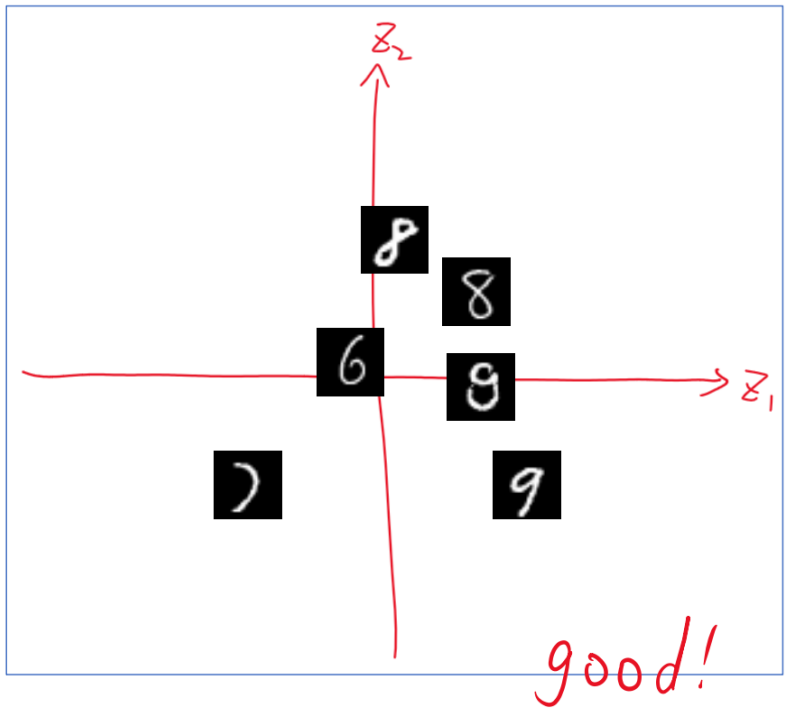
评估指标：

- 隐层表示的质量
- 数据生成的质量

Why VAE?

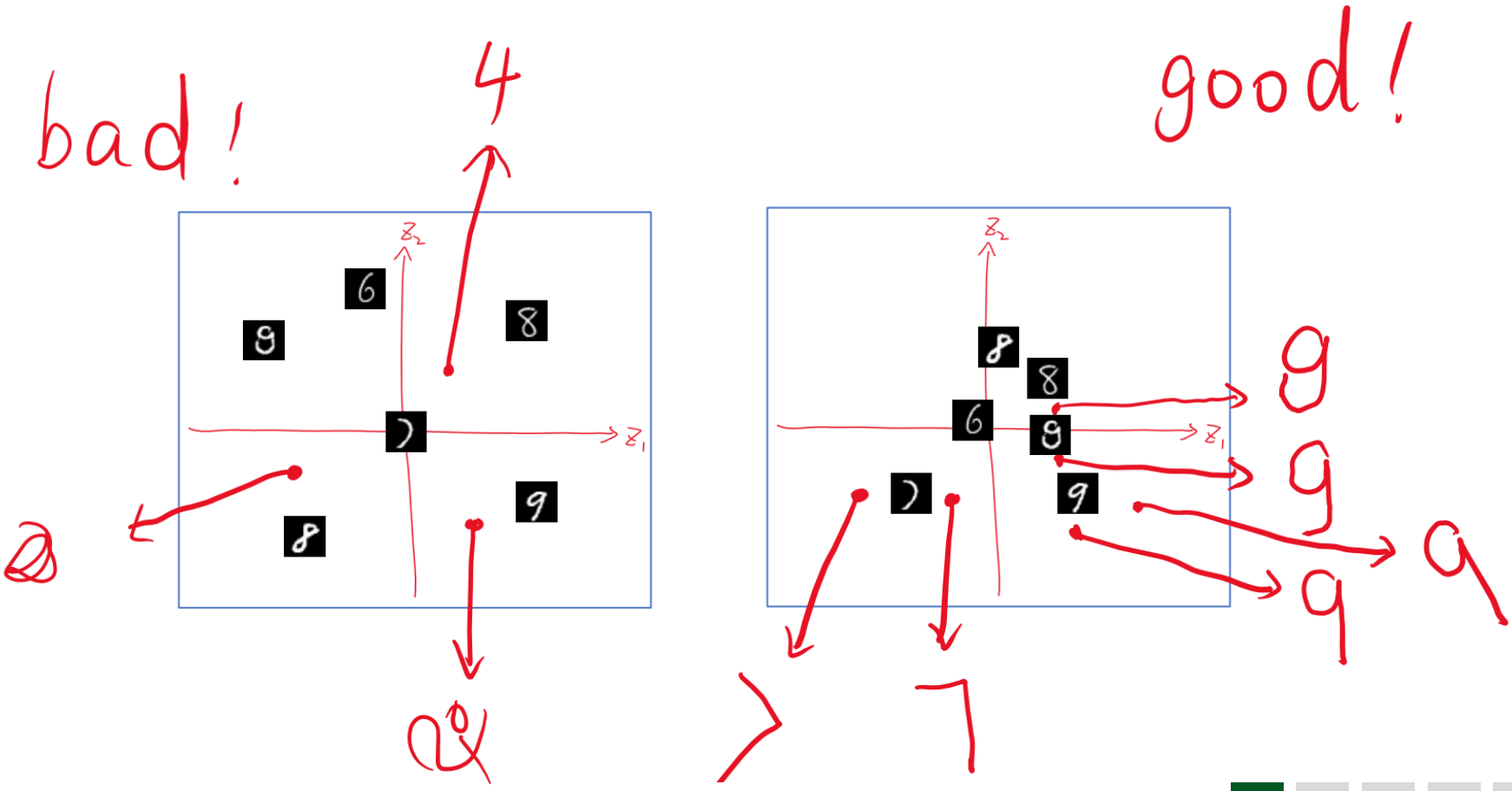
- 隐层表示的质量



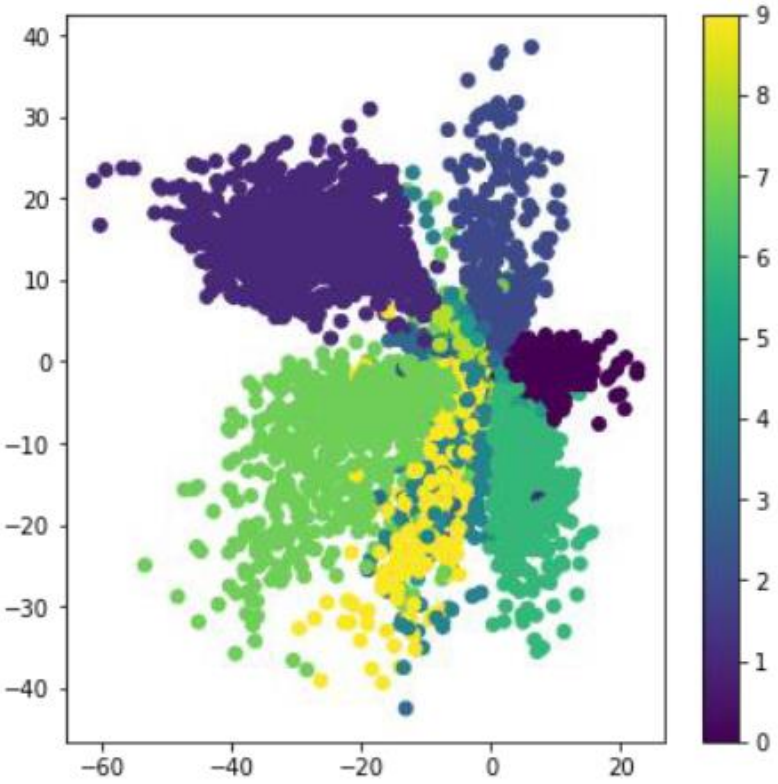


Why VAE?

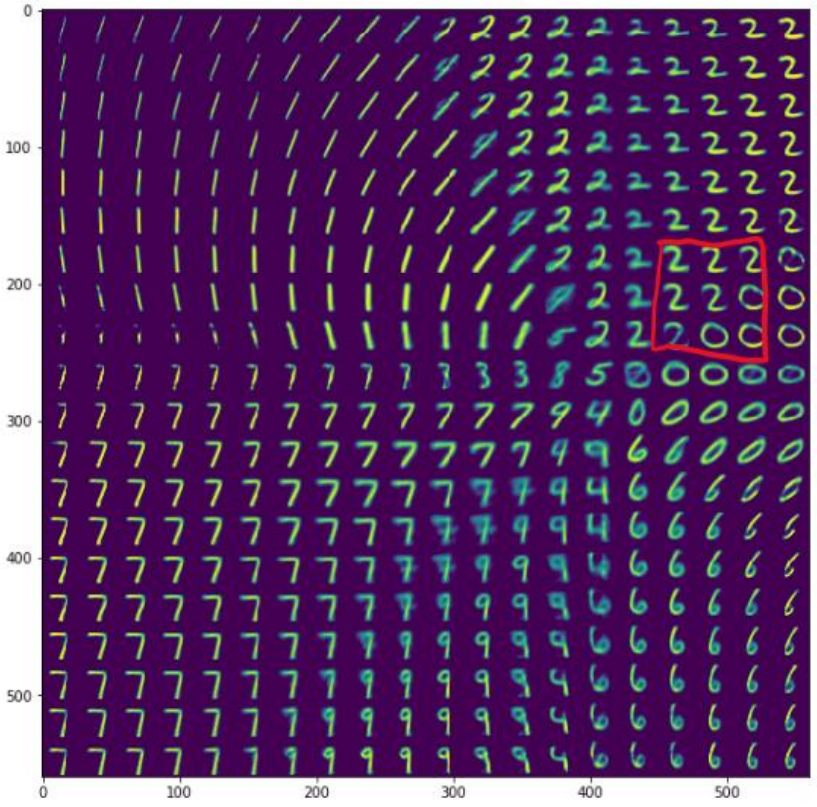
- 数据生成的质量

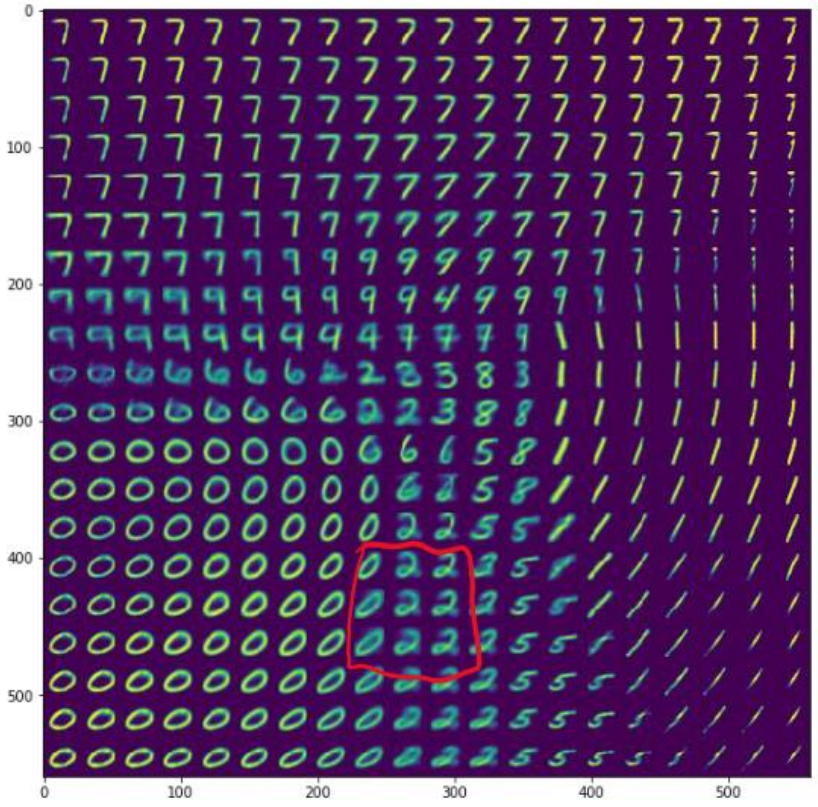


AutoEncoder –隐层表示

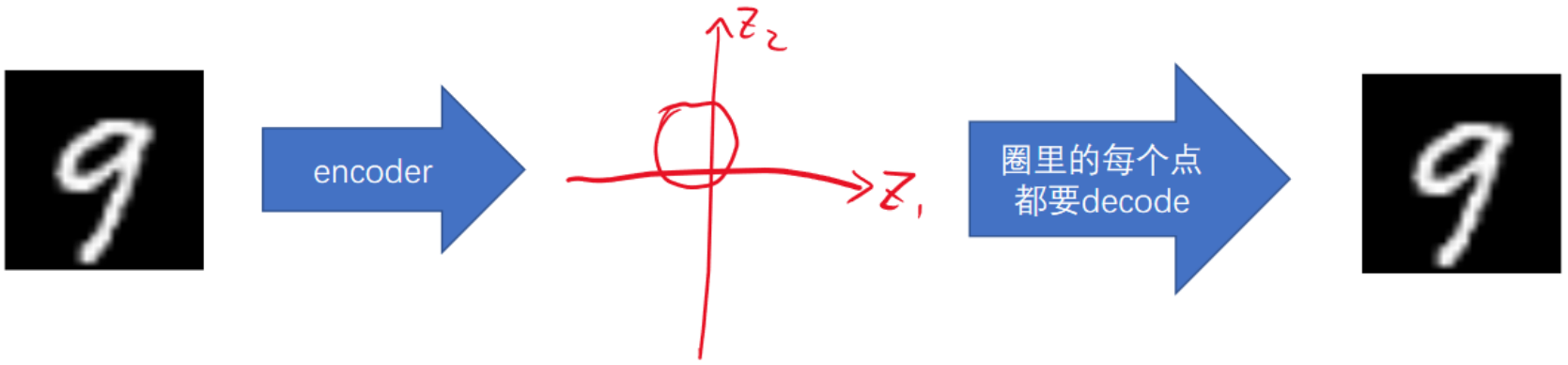


AutoEncoder –隐空间可解释性差

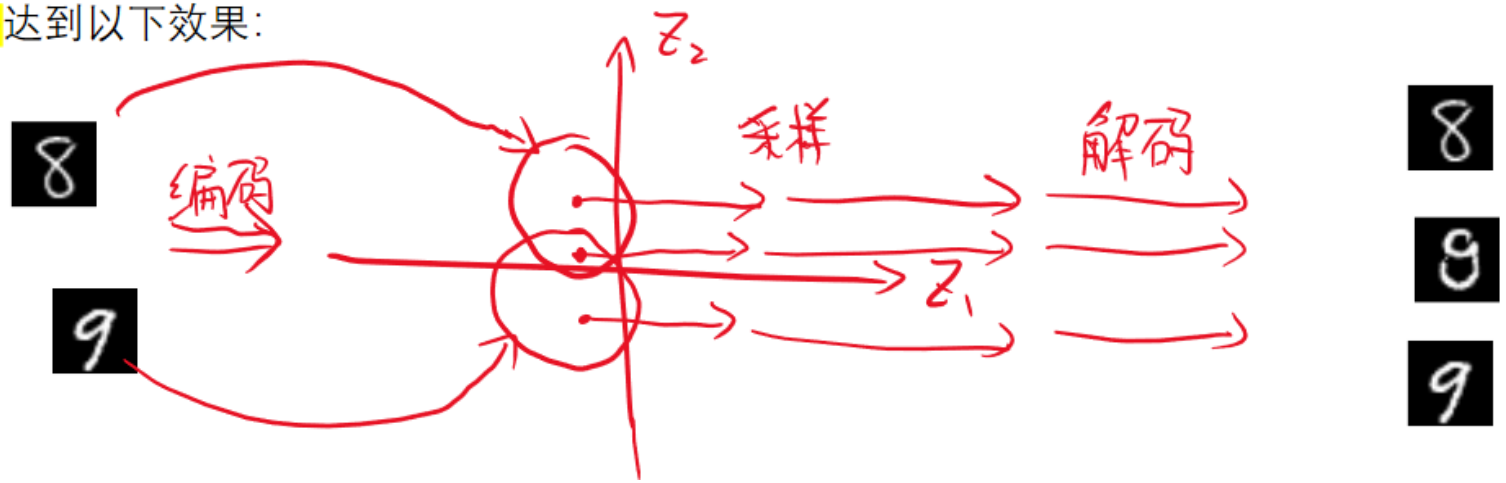




VAE：将每张图片编码为一个分布



这样就容易达到以下效果：



Questions

如何让一个图片对应隐空间的一个分布？

让一个图片encode成一个四维向量：分别表示二维高斯分布的均值和方差（假设二维高斯的二维独立）

如何让圈里的每个点（无数个点）的解码算加权的loss？

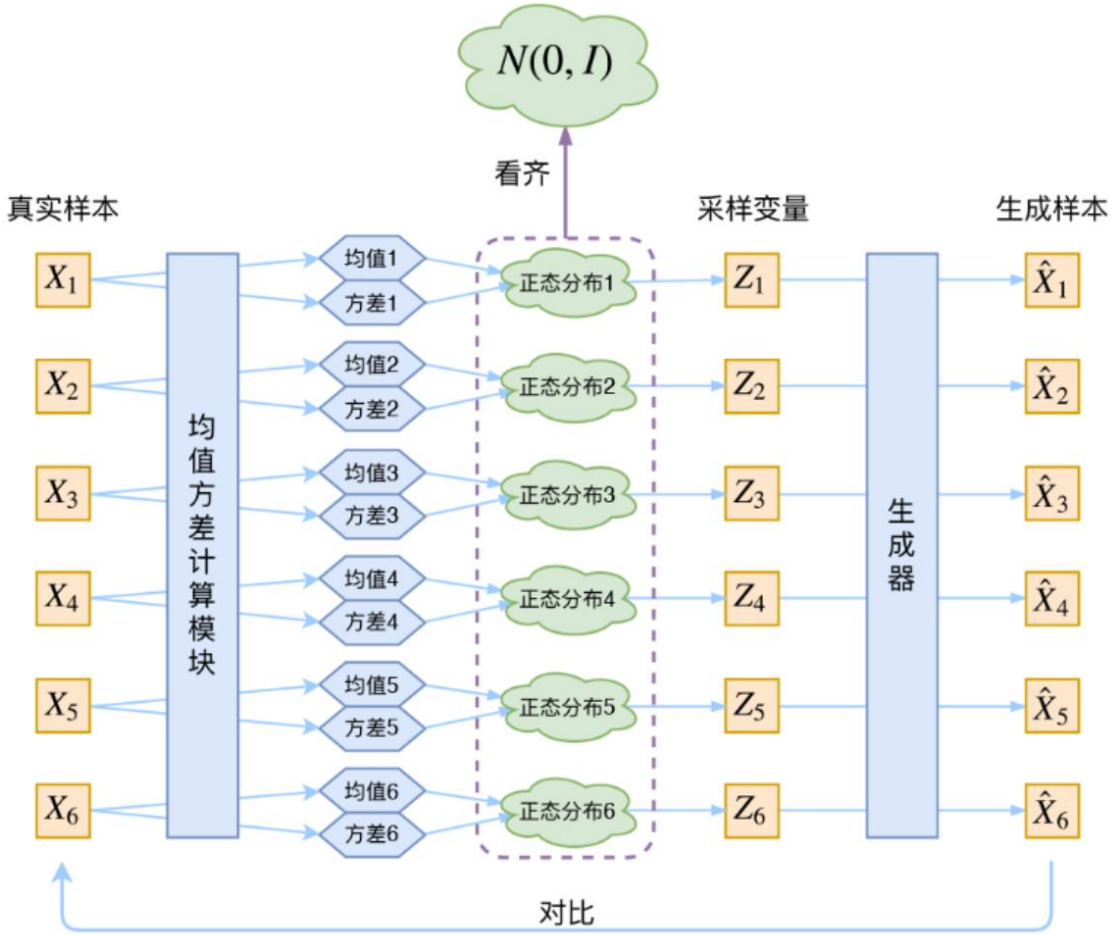
用采样代替全部计算，用采样数量代替加权

Problem

但是！如果Loss不变的话，依然会跟AE一模一样。

解决方法：从改变 Loss 函数入手。

那么，我们改变Loss，是要达到什么约束呢？



$$Loss = D_{KL}(q_\theta(z \mid x_i) \| p(z)) - E_{\sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)]$$

结果

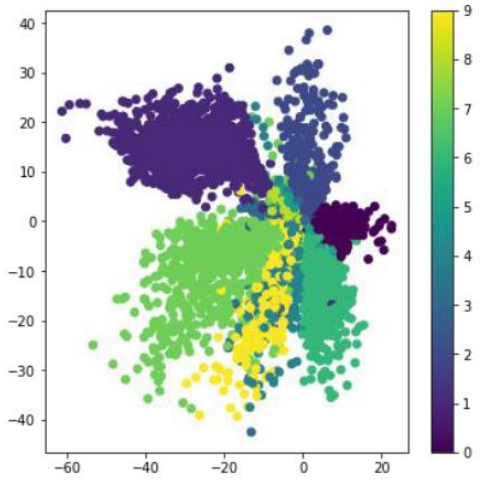
隐空间有规律可循，长得像的图片离得近

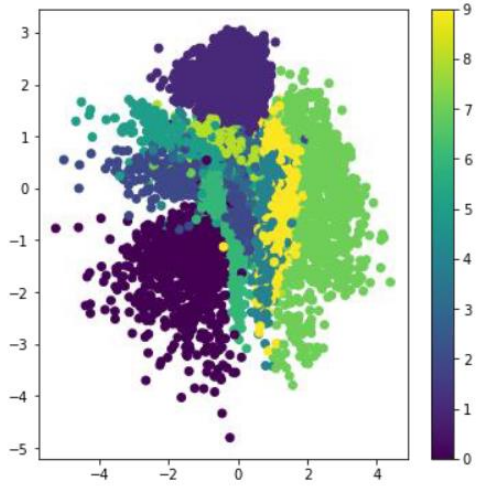
隐空间随机取点解码之后，得到的点有意义。

隐空间中对应不同标签的点不会离得很远（因为过渡点要被采样去算loss），但也不会离得太近（因为每个高斯的中心部分因为被采样次数多必须特色鲜明，不能跟别的类别的高斯中心离得太近）

隐空间对应相同标签的点离得比较近，但又不会聚成超小的小簇，然而也不会有相聚甚远的情况。

AE



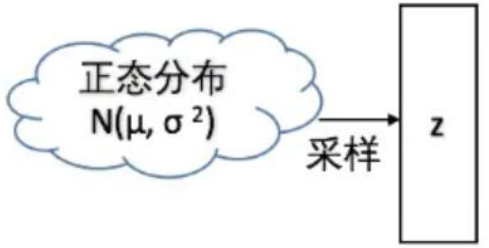


VAE

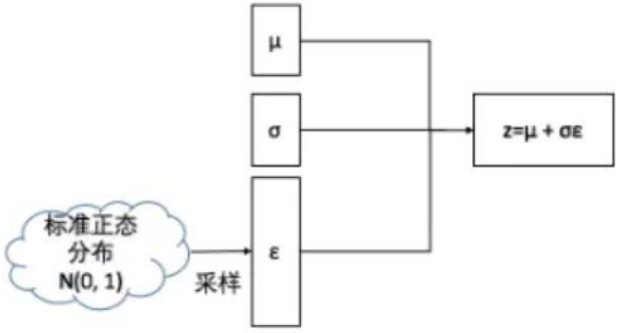
```python
class VAE(nn.Module):
    def __init__(self, image_size=784, h_dim=400, z_dim=20):
        super(VAE, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(image_size, h_dim),
            nn.LeakyReLU(0.2),
            nn.Linear(h_dim, z_dim*2)
        )
        self.decoder = nn.Sequential(
            nn.Linear(z_dim, h_dim),
            nn.ReLU(),
            nn.Linear(h_dim, image_size),
            nn.Sigmoid()
        )

    def reparameterize(self, mu, logvar):
        std = logvar.mul(0.5).exp_()
        esp = torch.randn(*mu.size())
        z = mu + std * esp
        return z

    def forward(self, x):
        h = self.encoder(x)
        mu, logvar = torch.chunk(h, 2, dim=-1)
        z = self.reparameterize(mu, logvar)
        return self.decoder(z), mu, logvar
```



一般采样



Reparameterization trick采样

```python
def loss_fn(recon_x, x, mu, logvar, beta):
    BEC = F.binary_cross_entropy(recon_x, x, size_average=False)
    KLD = -0.5 * torch.sum(1 + logvar - mu**2 - logvar.exp())
    return BCE + beta * KLD
```

Loss 的推导

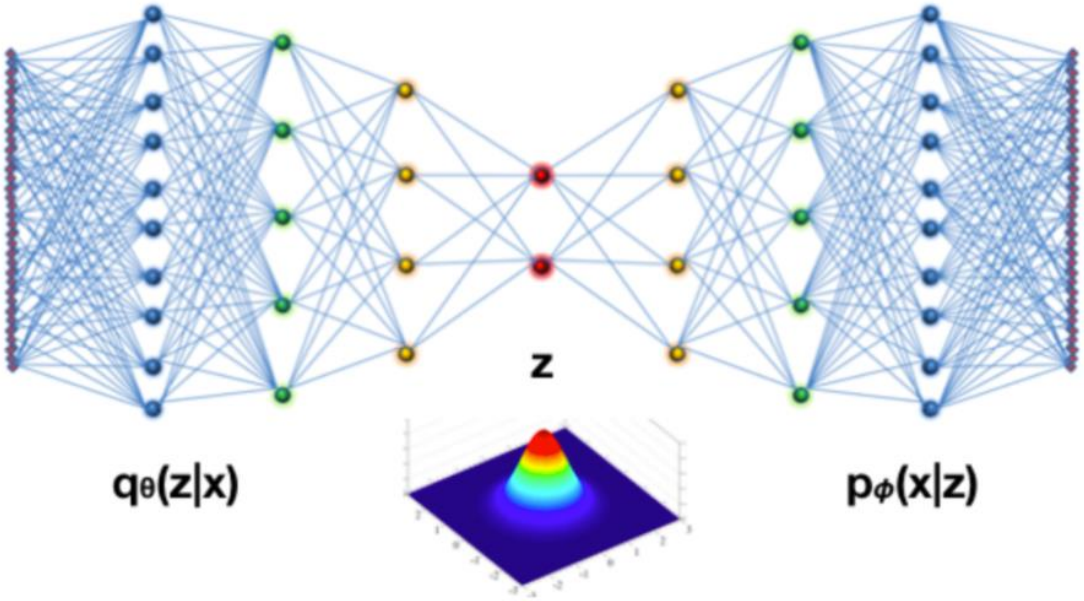
在贝叶斯机器学习中，后验分布(posterior distribution)往往难以计算，因此通常需要变分推断(variational inference)。通过这种方法，我们在训练过程中最大化数据的对数似然的证据下界(evidence lower bound)。变分自动编码器(variational autoencoder)是使用变分推理的一个重要例子。

数学基础回顾

1. 贝叶斯定理

2. 熵、交叉熵、KLD

网络结构与符号说明

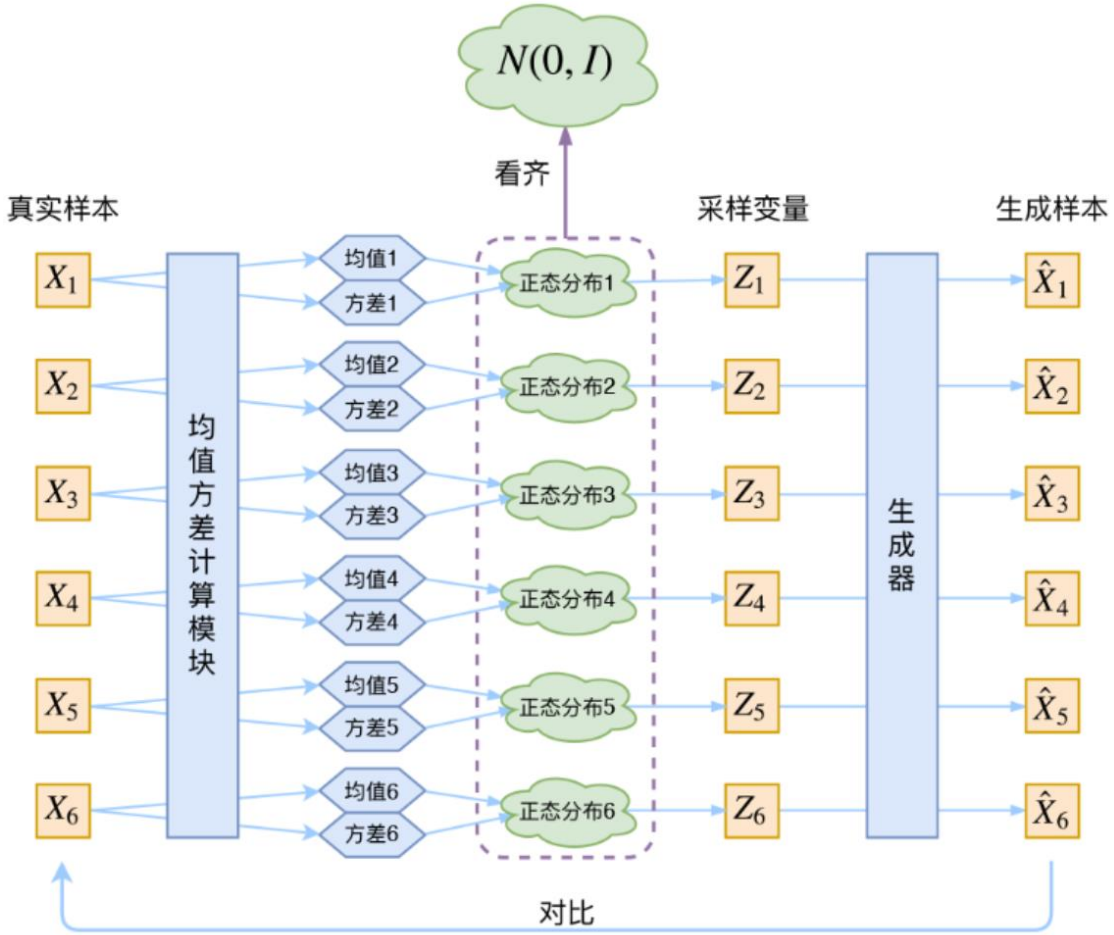


推导过程

$$Loss = D_{KL}(q_\theta(z \mid x_i) \| p(z \mid x_i)) = -\int q_\theta(z \mid x_i) \log\left(\frac{p(z \mid x_i)}{q_\theta(z \mid x_i)}\right) dz \geq 0$$

$$Loss = D_{KL}(q_\theta(z \mid x_i) \| p(z)) - E_{\sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)]$$

$$\log p(x_i) \geq -D_{KL}(q_\theta(z|x_i) \| p(z)) + E_{\sim q_\theta(z|x_i)}[\log p_\phi(x_i|z)]$$



$$Loss = D_{KL}(q_\theta(z \mid x_i) \| p(z)) - E_{\sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)]$$

假设：

$$p(z) \to \frac{1}{\sqrt{2\pi\sigma_p^2}} \exp\left(-\frac{(z-\mu_p)^2}{2\sigma_p^2}\right)$$

$$q_\theta(z \mid x_i) \to \frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\left(-\frac{(z-\mu_q)^2}{2\sigma_q^2}\right)$$

代入化简：

$$-D_{KL}(q_\theta(z \mid x_i)||p(z))$$

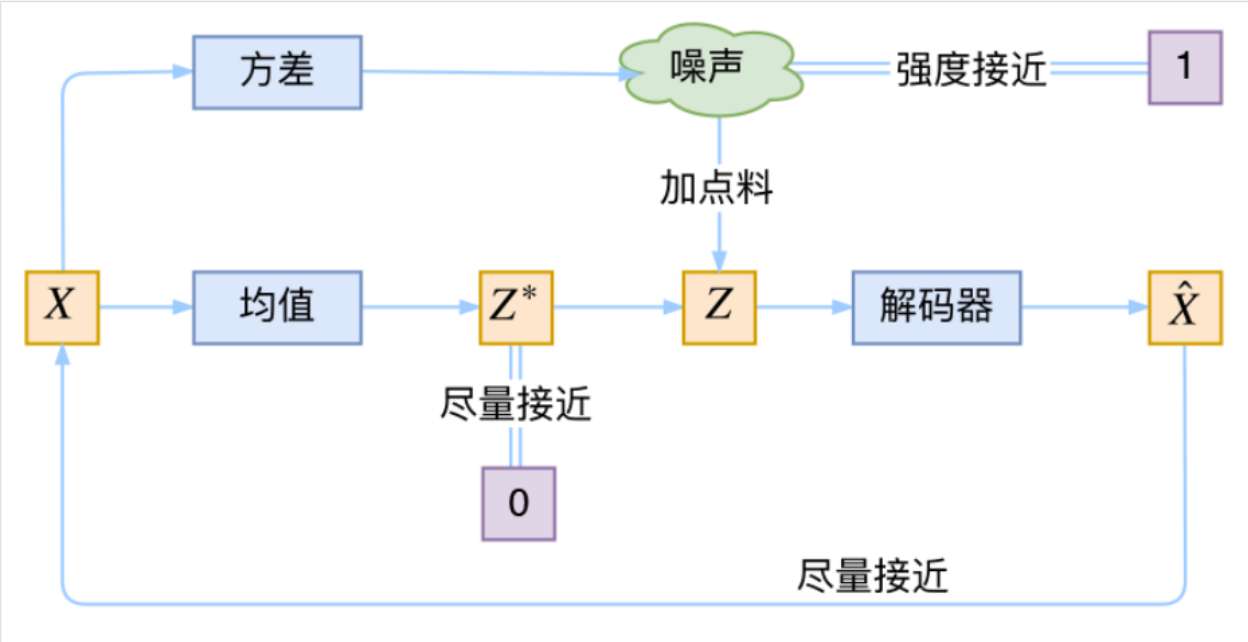
$$= \int \frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\left(-\frac{(x-\mu_q)^2}{2\sigma_q^2}\right) \log\left(\frac{\frac{1}{\sqrt{2\pi\sigma_p^2}} \exp\left(-\frac{(x-\mu_p)^2}{2\sigma_p^2}\right)}{\frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\left(-\frac{(x-\mu_q)^2}{2\sigma_q^2}\right)}\right) dz$$

$$= \log\left(\frac{\sigma_q}{\sigma_p}\right) - \frac{\sigma_q^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2} + \frac{1}{2}$$

假设P为标准高斯分布，可得：

$$
\begin{aligned}
-D_{KL}(q_\theta(z|x_i)||p(z)) &= \log(\sigma_q) - \frac{\sigma_q^2 + \mu_q^2}{2} + \frac{1}{2} \\
&= \frac{1}{2}\log(\sigma_q^2) - \frac{\sigma_q^2 + \mu_q^2}{2} + \frac{1}{2} \\
&= \frac{1}{2}\left[1 + \log(\sigma_q^2) - \sigma_q^2 - \mu_q^2\right]
\end{aligned}
$$

```
KLD = -0.5 * torch.sum(1 + logvar - mu**2 - logvar.exp())
```



vae的本质结构

它本质上就是在我们常规的自编码器的基础上，对encoder的结果加上了“高斯噪声”，使得结果decoder能够对噪声有鲁棒性。重构的过程是希望没噪声的，而KL 部分则希望有高斯噪声的，两者是对立的。所以VAE的训练也是一个寻求平衡的过程。

References

- https://arxiv.org/abs/1907.08956

THANKS

- 王若琪
- 20210522