无监督深度学习——生成模型（1）

主讲：王皓 副研究员| 硕士导师

邮箱：wanghao3@ustc.edu.cn

主页：http://staff.ustc.edu.cn/~wanghao3

为什么无监督学习

- 有监督学习依赖于大量的标记数据，但获取数据标签开销巨大
- 若没有足够的有标签数据，应该怎么办呢?
- 可以利用无监督深度学习方法，利用大量未标记数据

本章内容

- ➢自编码器（AutoEncoder）
  - PCA
  - 降噪自编码器（Denoising AutoE
  - 稀疏自编码器（Sparse AutoEnco
- ➢变分自编码器（VAE）
- ➢生成对抗网络（GAN）

$$S = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top = \frac{1}{n-1}\widetilde{X}\widetilde{X}^\top,$$

$$\max_{G\in\mathbb{R}^{d\times K}} \frac{1}{n-1}\sum_{i=1}^{n}\|GG^\top\mathbf{x}_i - GG^\top\bar{\mathbf{x}}\|^2, \quad \text{s.t. } G^\top G = I.$$

Notice that

$$\begin{aligned}
\frac{1}{n-1}\sum_{i=1}^{n}\|GG^\top\mathbf{x}_i - GG^\top\bar{\mathbf{x}}\|^2 &= \frac{1}{n-1}\sum_{i=1}^{n}\langle GG^\top(\mathbf{x}_i-\bar{\mathbf{x}}), GG^\top(\mathbf{x}_i-\bar{\mathbf{x}})\rangle \\
&= \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i-\bar{\mathbf{x}})^\top GG^\top GG^\top(\mathbf{x}_i-\bar{\mathbf{x}}) \\
&= \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i-\bar{\mathbf{x}})^\top GG^\top(\mathbf{x}_i-\bar{\mathbf{x}}) \\
&= \frac{1}{n-1}\sum_{i=1}^{n}\mathrm{tr}\left((\mathbf{x}_i-\bar{\mathbf{x}})^\top GG^\top(\mathbf{x}_i-\bar{\mathbf{x}})\right) \\
&= \frac{1}{n-1}\sum_{i=1}^{n}\mathrm{tr}\left(G^\top(\mathbf{x}_i-\bar{\mathbf{x}})(\mathbf{x}_i-\bar{\mathbf{x}})^\top G\right) \\
&= \mathrm{tr}\left(G^\top\left(\frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i-\bar{\mathbf{x}})(\mathbf{x}_i-\bar{\mathbf{x}})^\top\right)G\right) \\
&= \mathrm{tr}\left(G^\top S G\right).
\end{aligned}$$

Denote

$$Q = U^\top G.$$

We can see that

$$Q^\top Q = I.$$

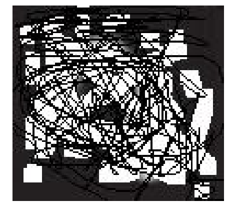
Thus, the problem in (11) reduces to

$$\max_{Q\in\mathbb{R}^{d\times K}} \mathrm{tr}(Q^\top \Sigma_d^2 Q), \quad \text{s.t. } Q^\top Q = I.$$

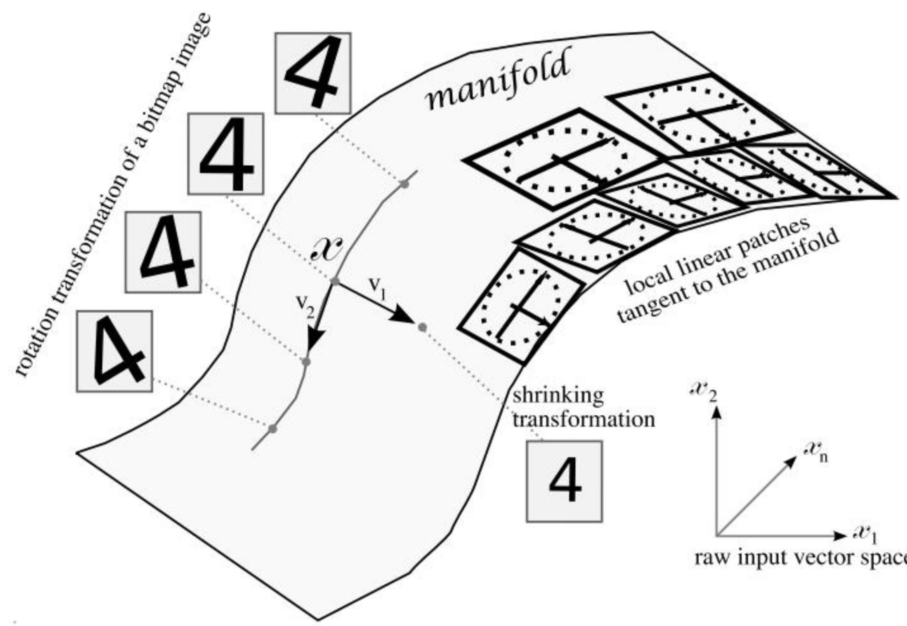
AutoEncoder

- 在MINST中，用28 × 28 = 784维矩阵表示每个数字

- 但不是每个784维的矩阵都表示数字

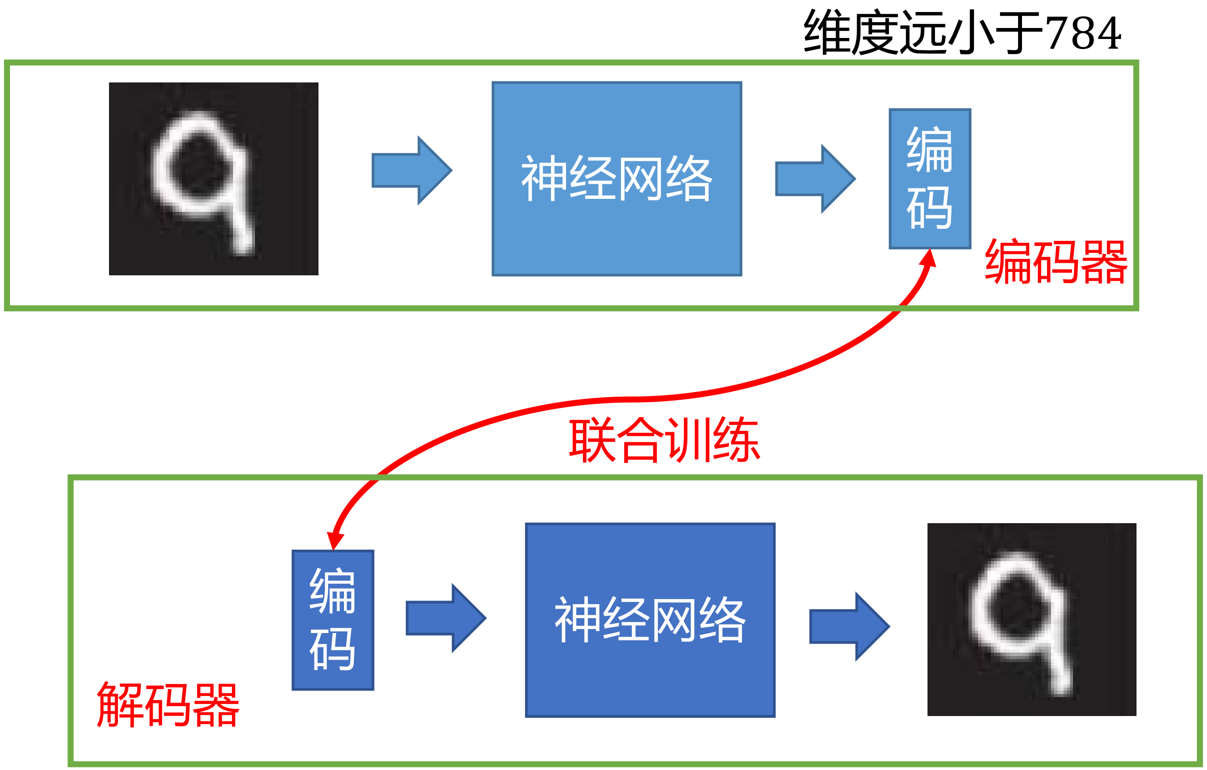


这些数字图像分布在784维空间内嵌的低维流形空间（manifold）上



因此可以用更紧凑的向量来表示这些数字图像

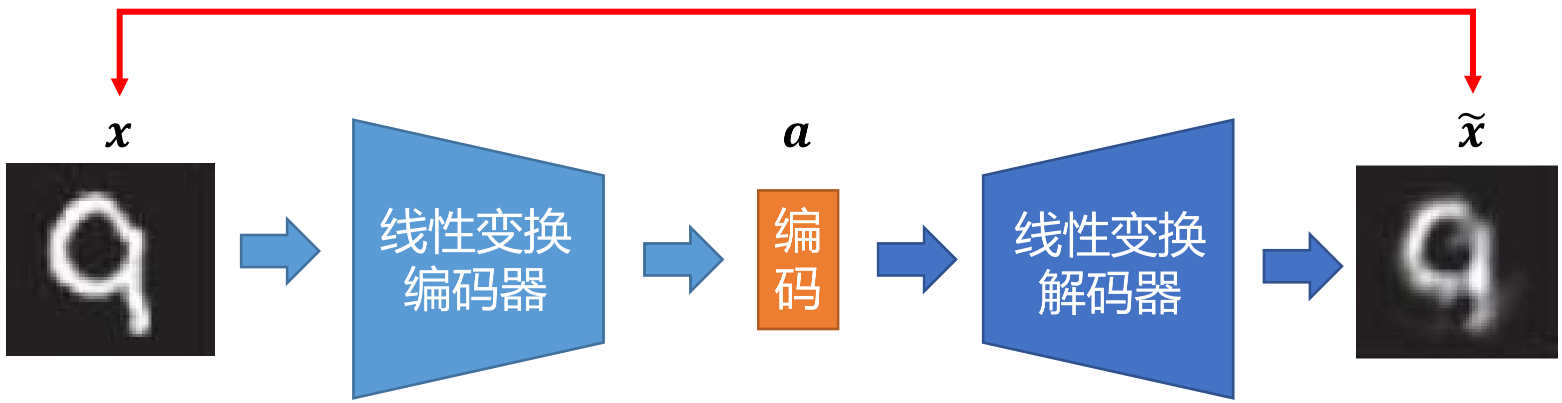
AutoEncoder



对比PCA&AutoEncoder

PCA

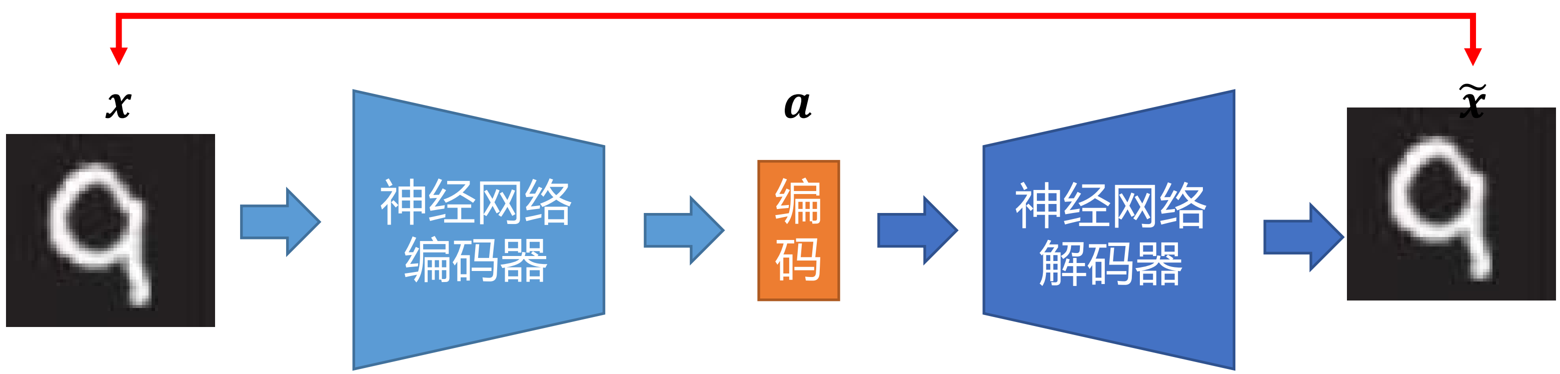
最小化输入和输出之间的差异$(\boldsymbol{x}-\tilde{\boldsymbol{x}})^2$



$\boldsymbol{a}=\boldsymbol{W}^{\top}\boldsymbol{x}$ $\qquad$ $\tilde{\boldsymbol{x}}=\boldsymbol{W}\boldsymbol{a}$

AutoEncoder

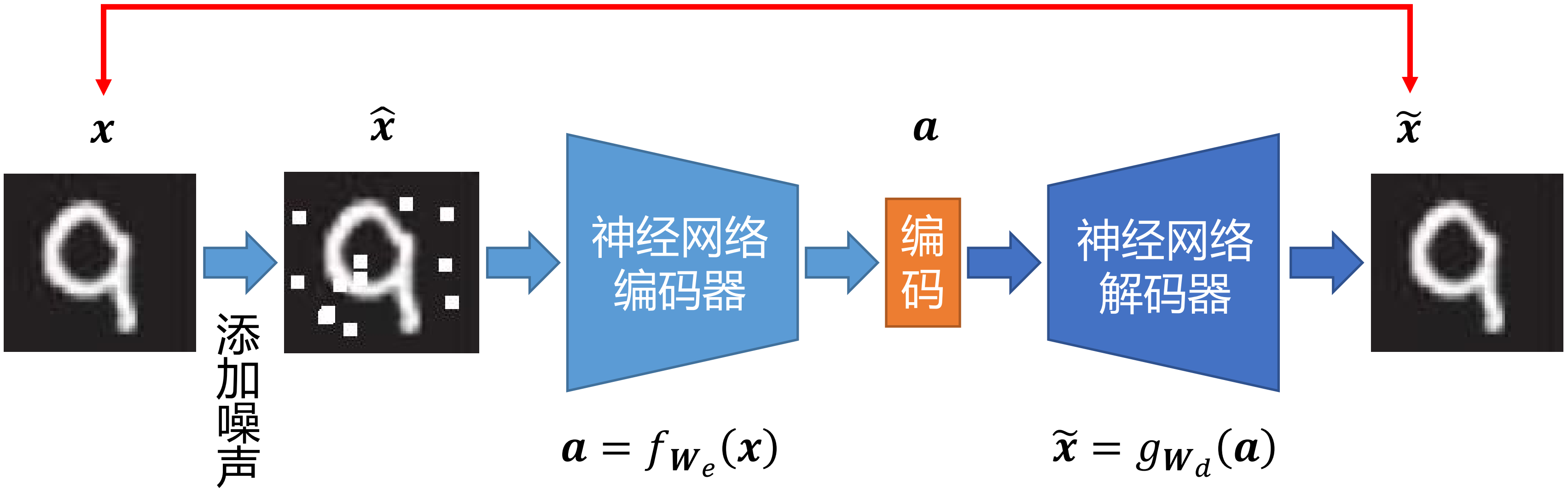
最小化输入和输出之间的差异$(\boldsymbol{x}-\tilde{\boldsymbol{x}})^2$



$\boldsymbol{a}=f_{W_e}(\boldsymbol{x})$ $\qquad$ $\tilde{\boldsymbol{x}}=g_{W_d}(\boldsymbol{a})$

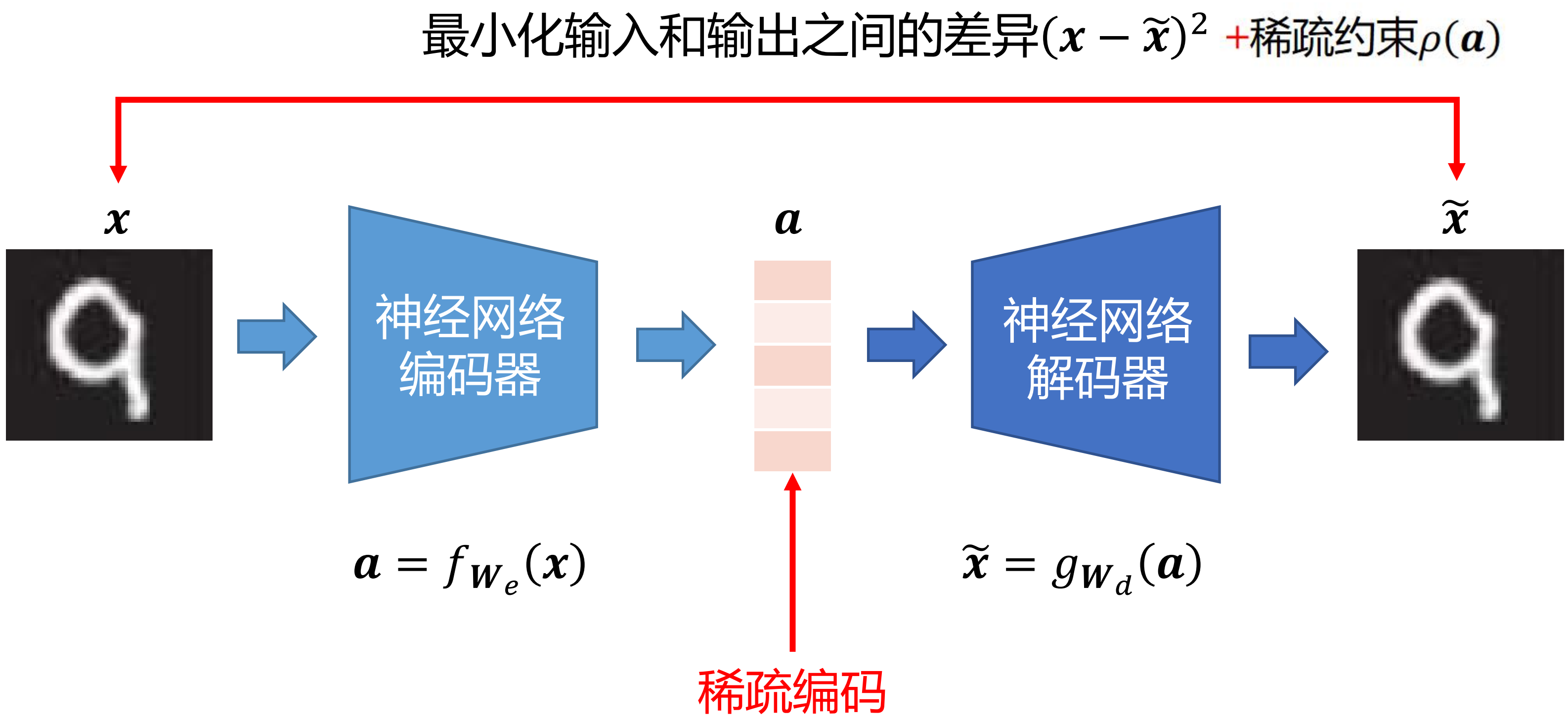
Denoising AutoEncoder

最小化输入和输出之间的差异$(\boldsymbol{x}-\tilde{\boldsymbol{x}})^2$



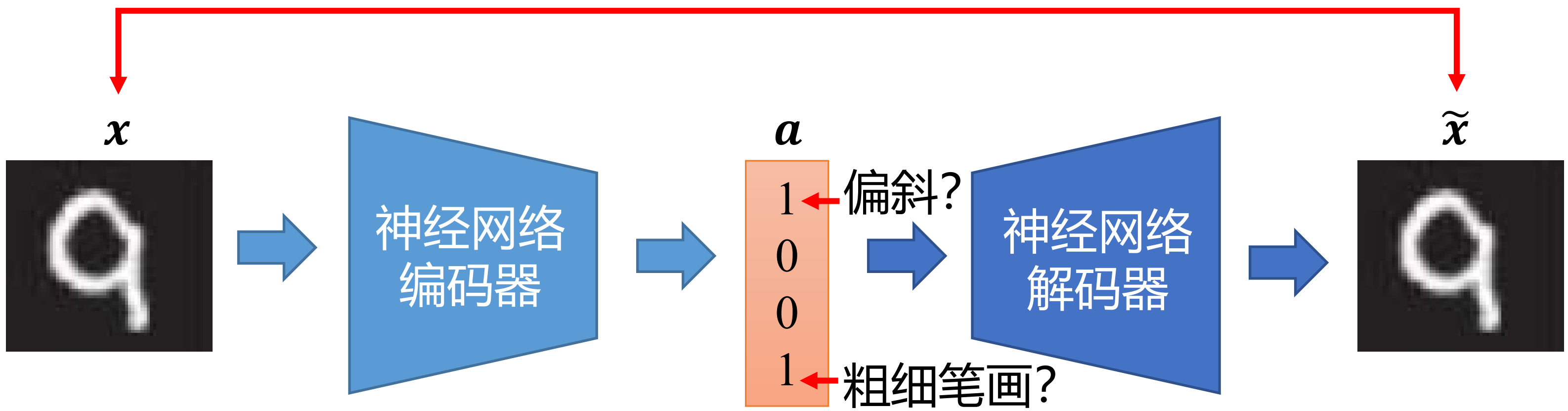
$\boldsymbol{a} = f_{W_e}(\boldsymbol{x})$ $\tilde{\boldsymbol{x}} = g_{W_d}(\boldsymbol{a})$

Sparse AutoEncoder

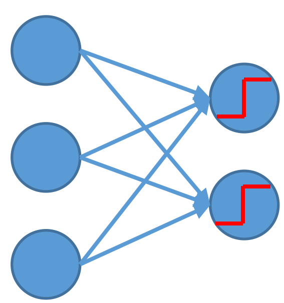


Discrete AutoEncoder

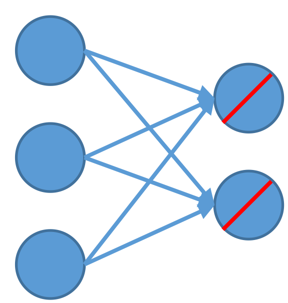
最小化输入和输出之间的差异$(\boldsymbol{x}-\tilde{\boldsymbol{x}})^2$



$\boldsymbol{a}=f_{W_e}(\boldsymbol{x})$　　二值向量　　$\tilde{\boldsymbol{x}}=g_{W_d}(\boldsymbol{a})$

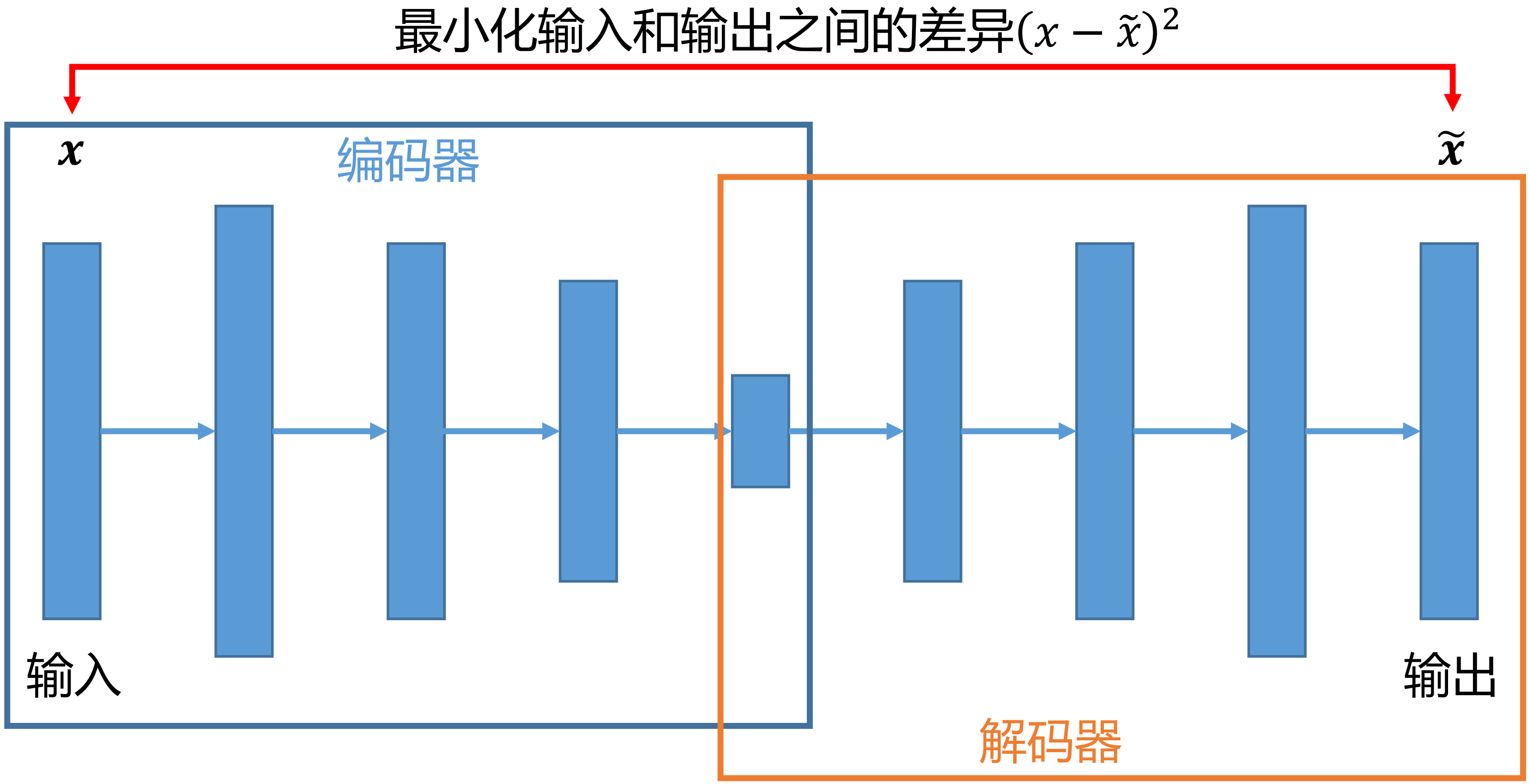


前向

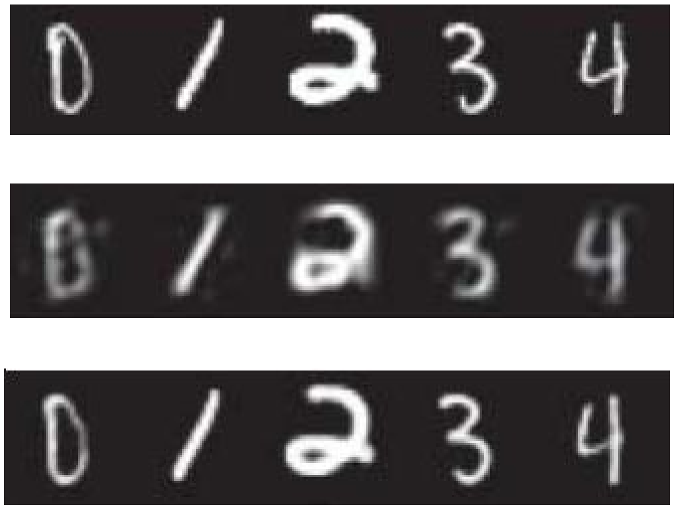


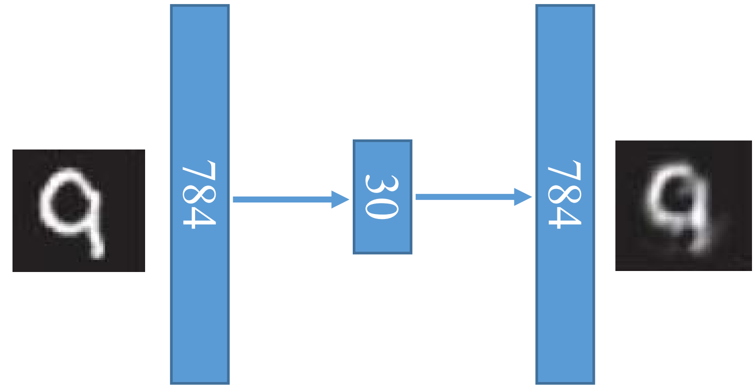
后向

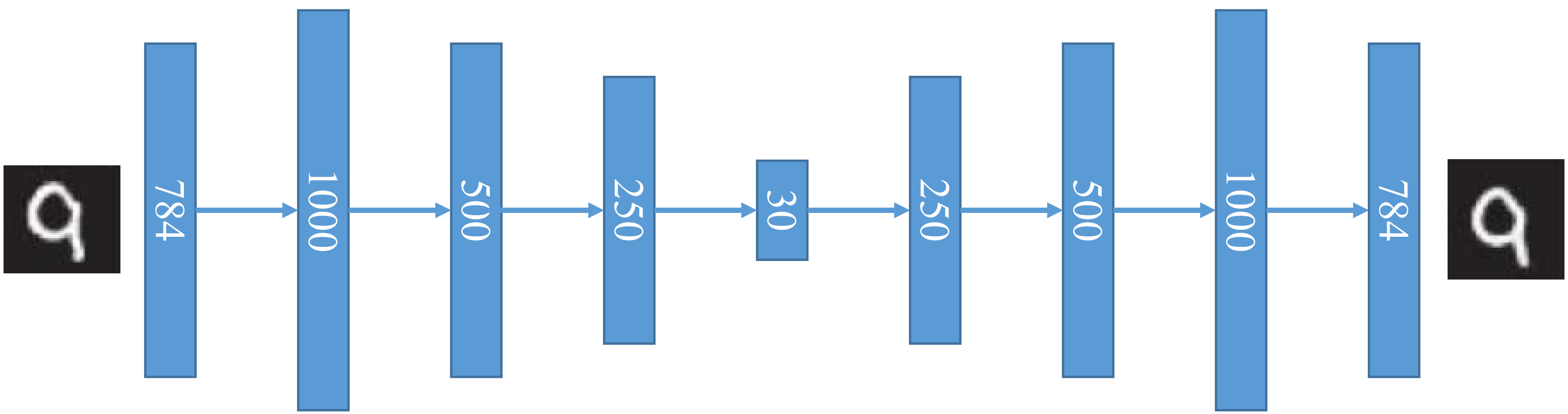
Deep AutoEncoder



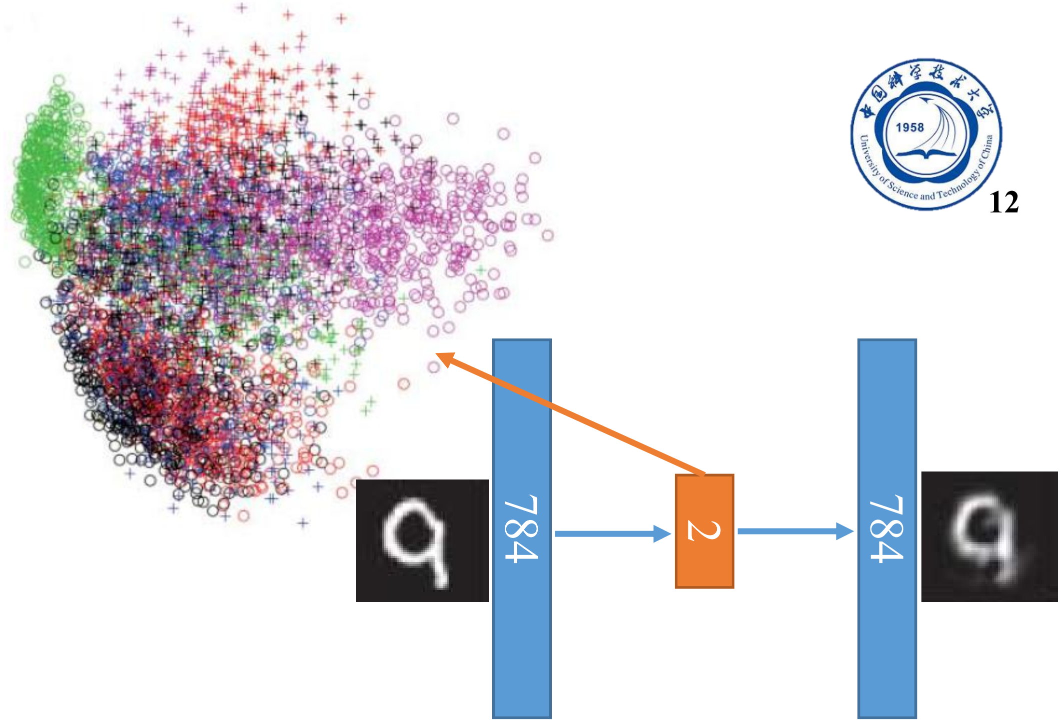
Deep AutoEncoder

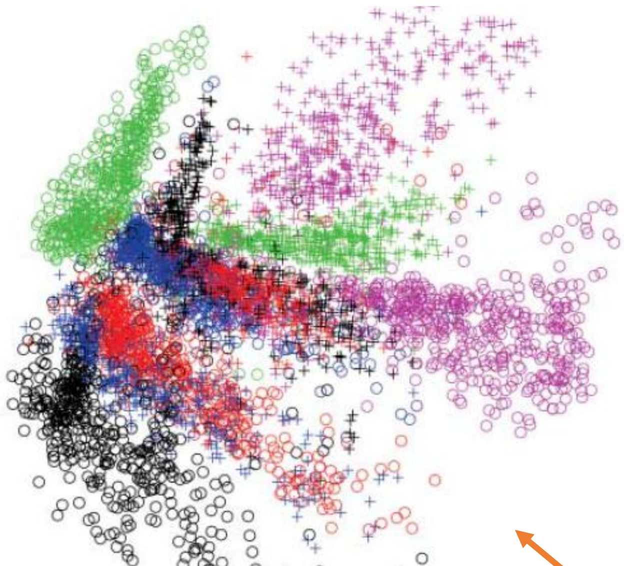


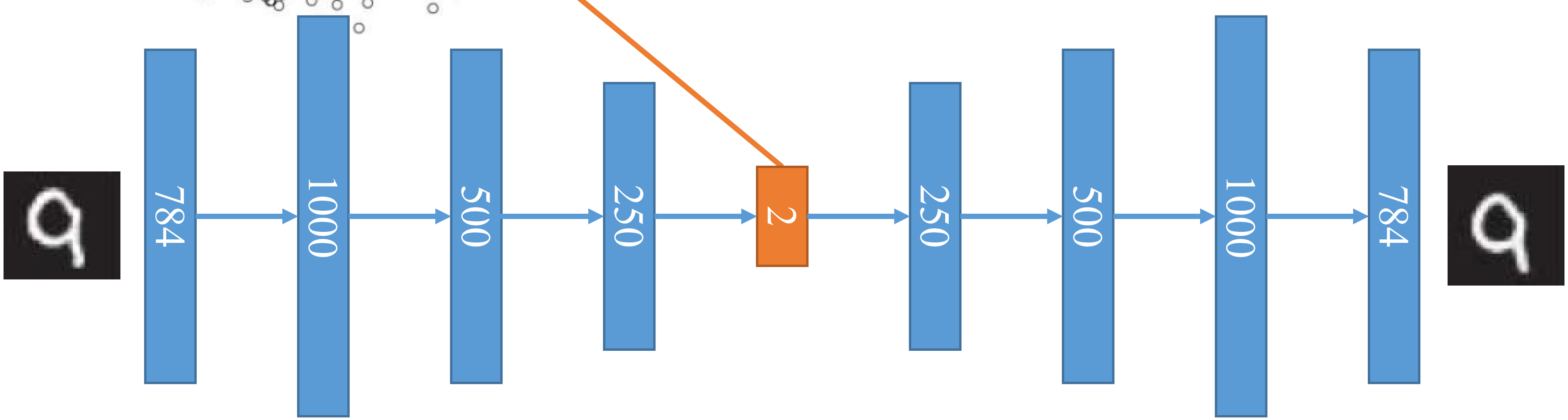




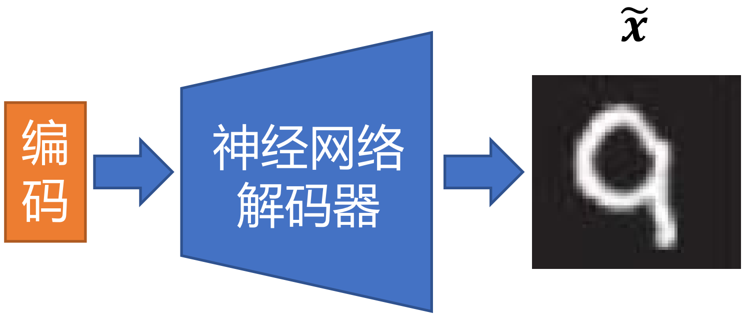
Deep AutoEncoder





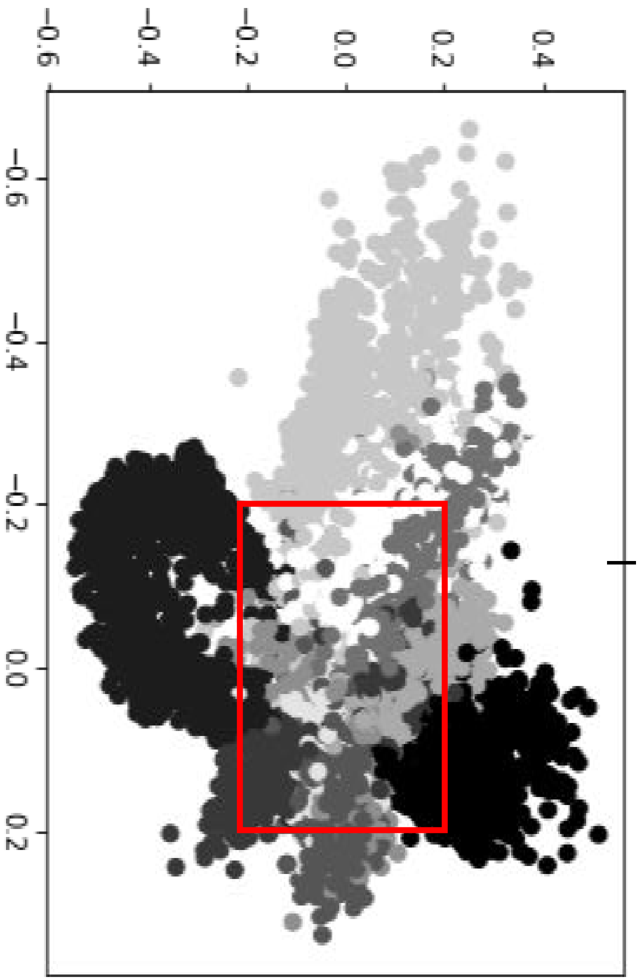


用AutoEncoder生成图片



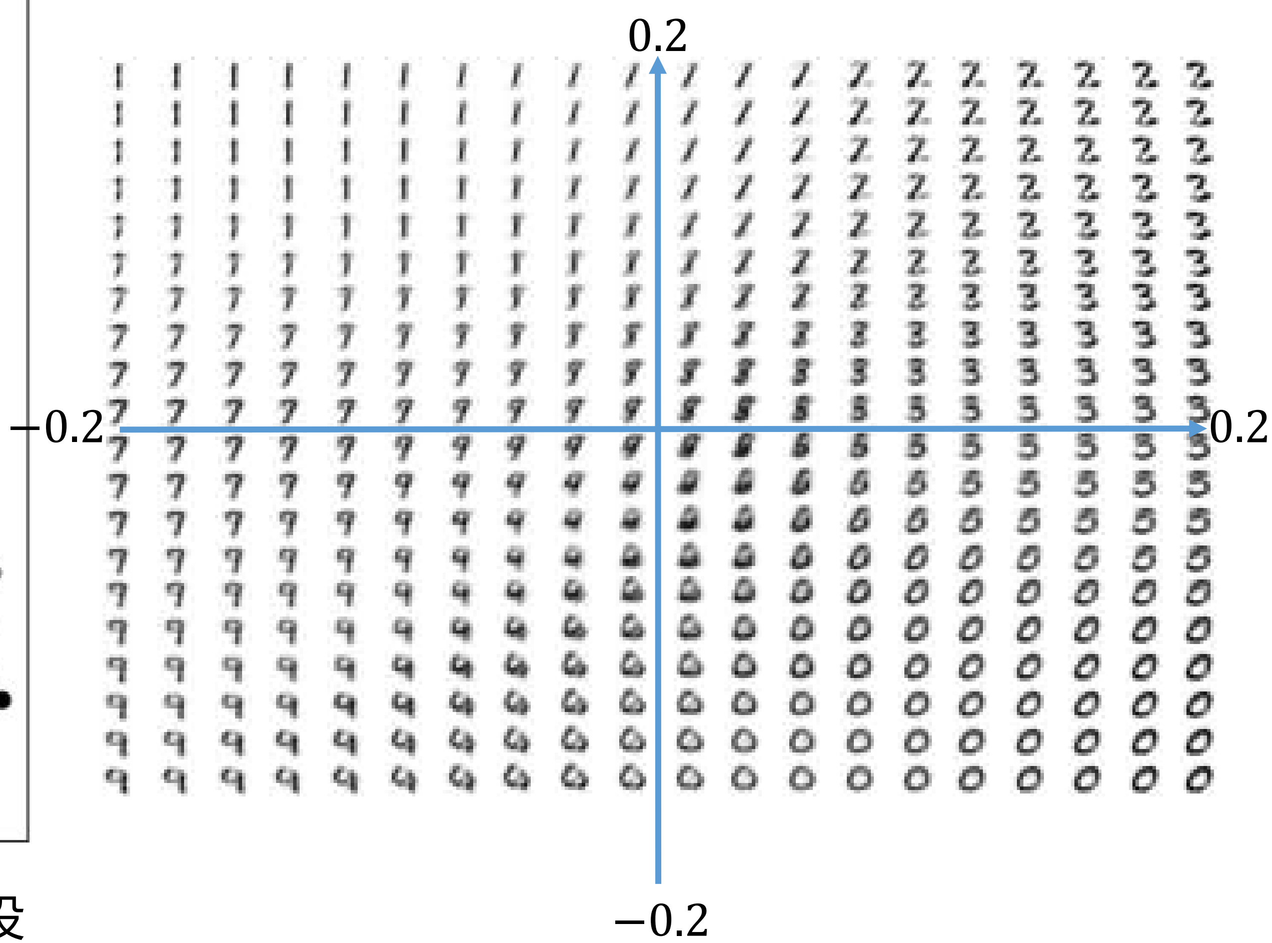
$$\tilde{\boldsymbol{x}} = g_{W_d}(\boldsymbol{a})$$

用AutoEncoder生成图片

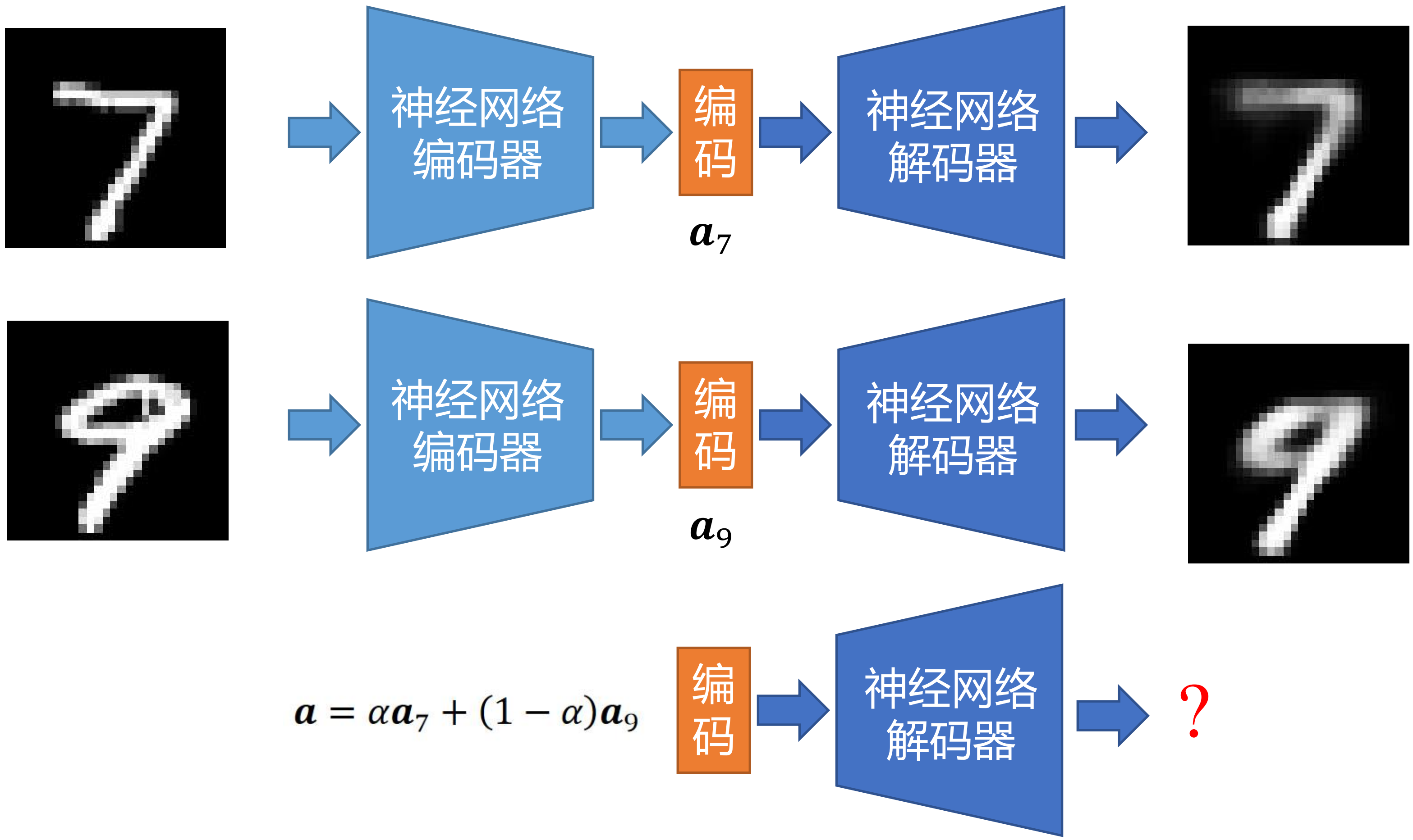


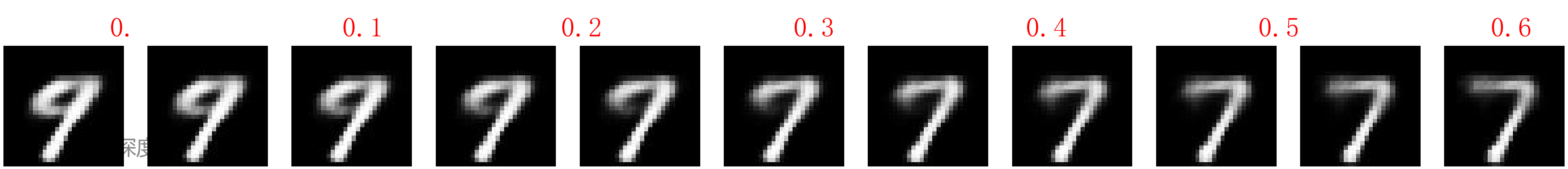
通过L2正则编码器将图片投影到2维空间

在红色区域内均匀采样400个坐标，作为解码器输入

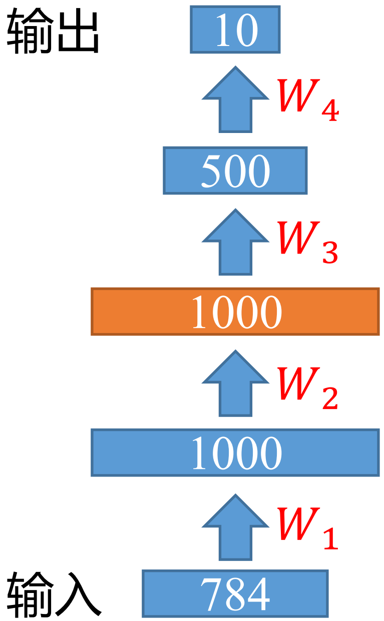


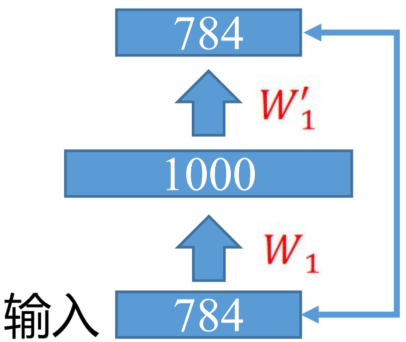
用AutoEncoder生成图片



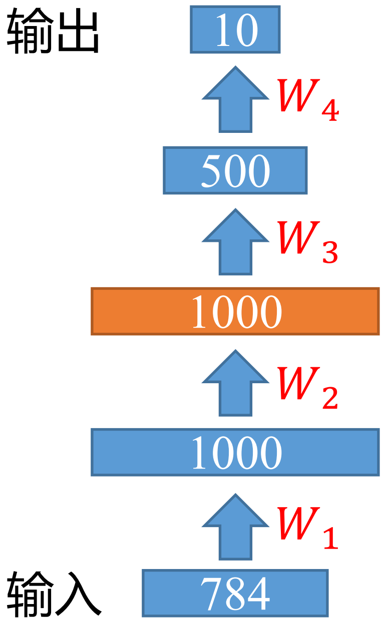


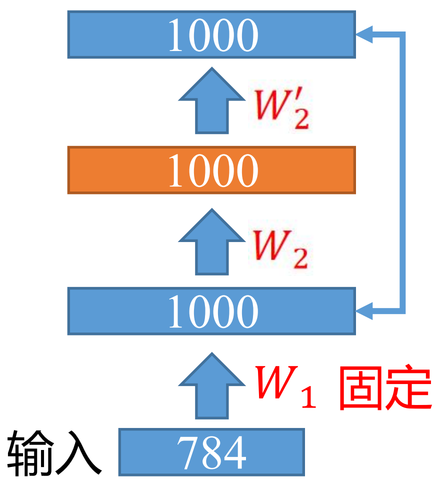
AutoEncoder用于预训练



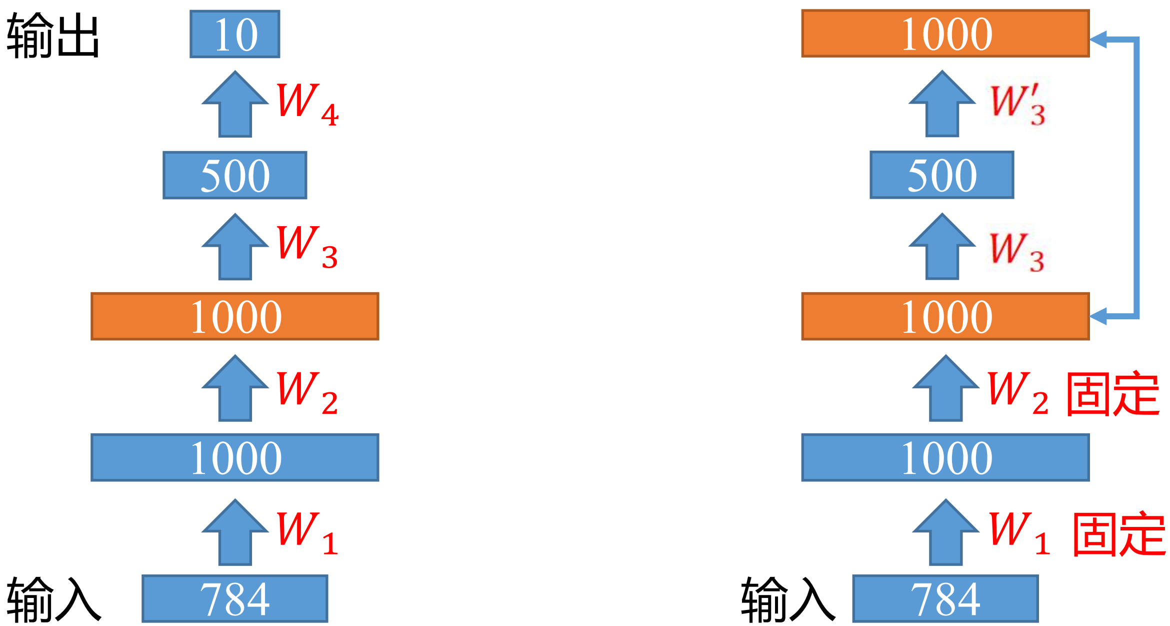


AutoEncoder用于预训练

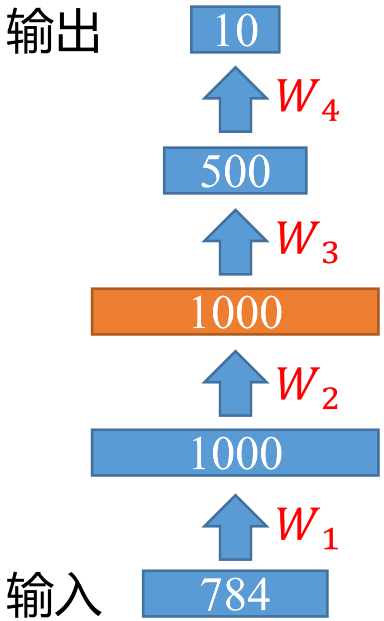


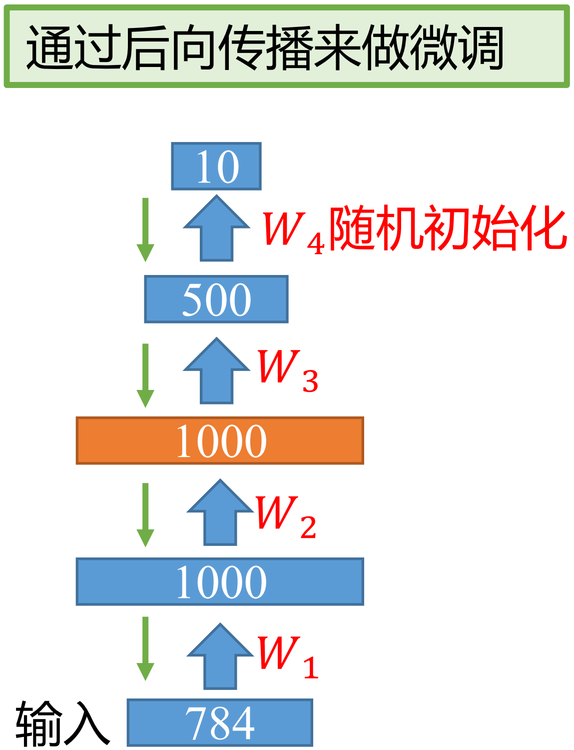


AutoEncoder用于预训练



AutoEncoder用于预训练



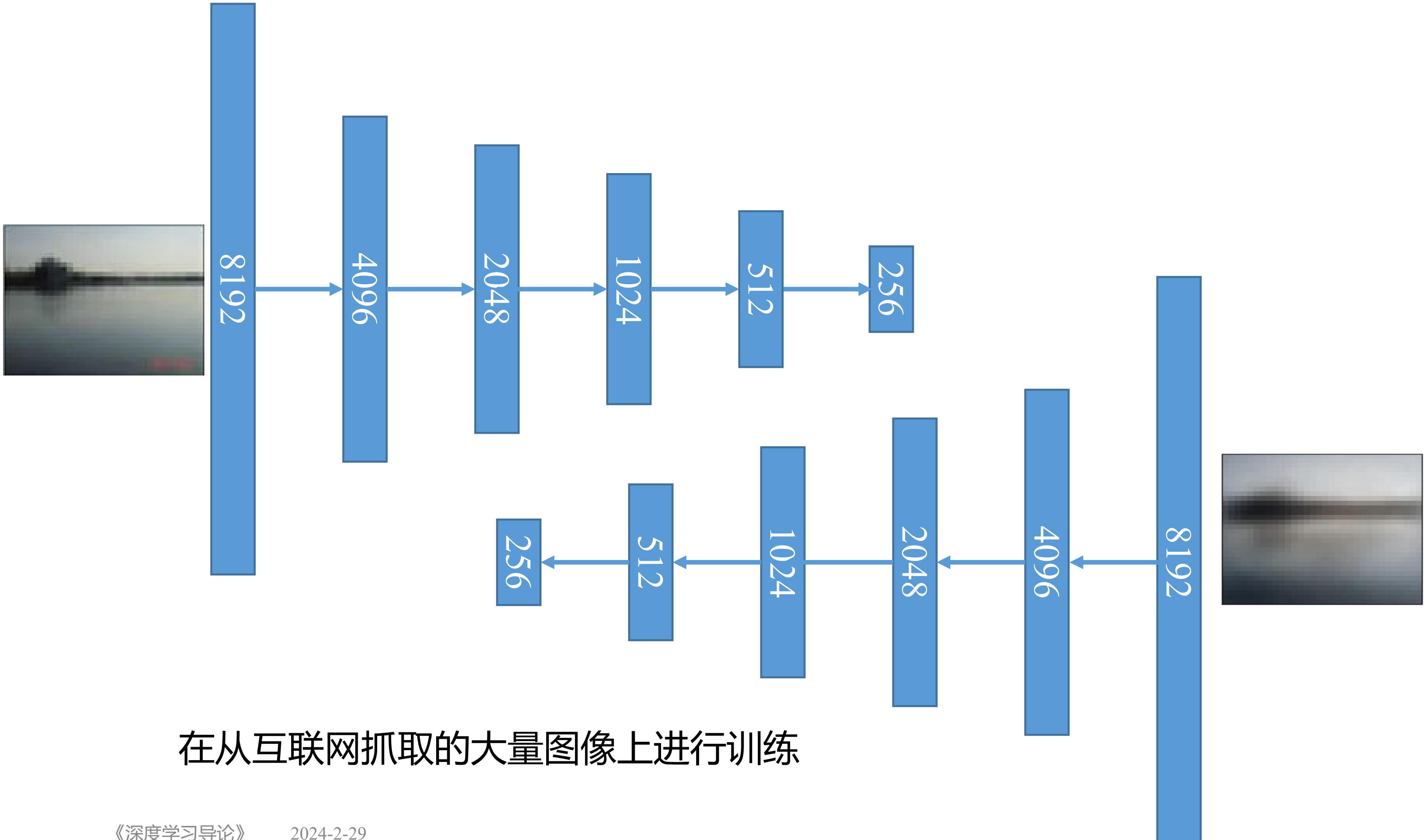


AutoEncoder用于相似图片检索

用迈克杰克逊的头像进行检索

按照图片像素相似度，最相似的图片

AutoEncoder用于相似图片检索



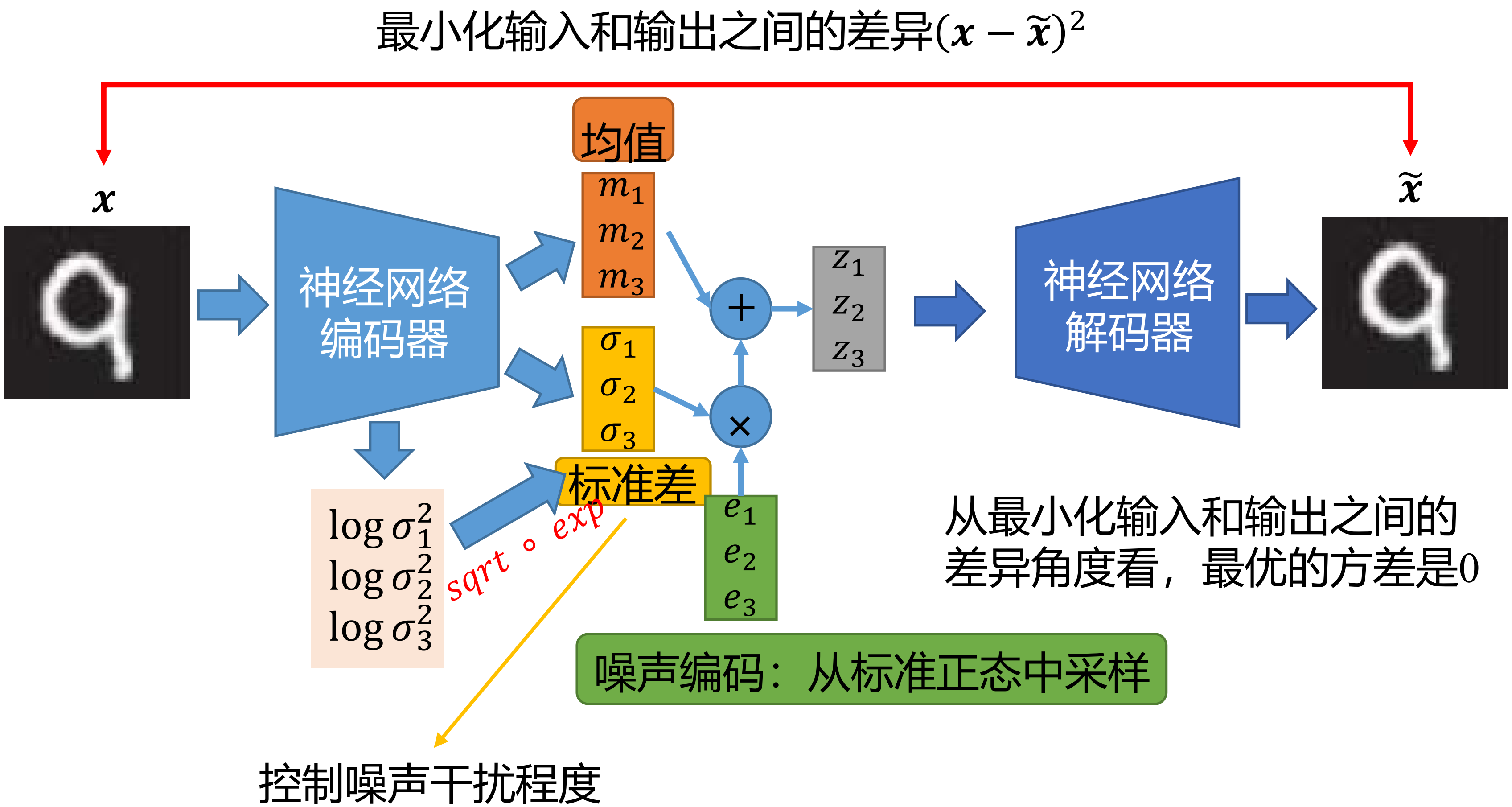
在从互联网抓取的大量图像上进行训练

AutoEncoder用于相似图片检索

用迈克杰克逊的头像进行检索

按照图片的256维的编码向量，最相似的图片

Variational AutoEncoder (VAE)

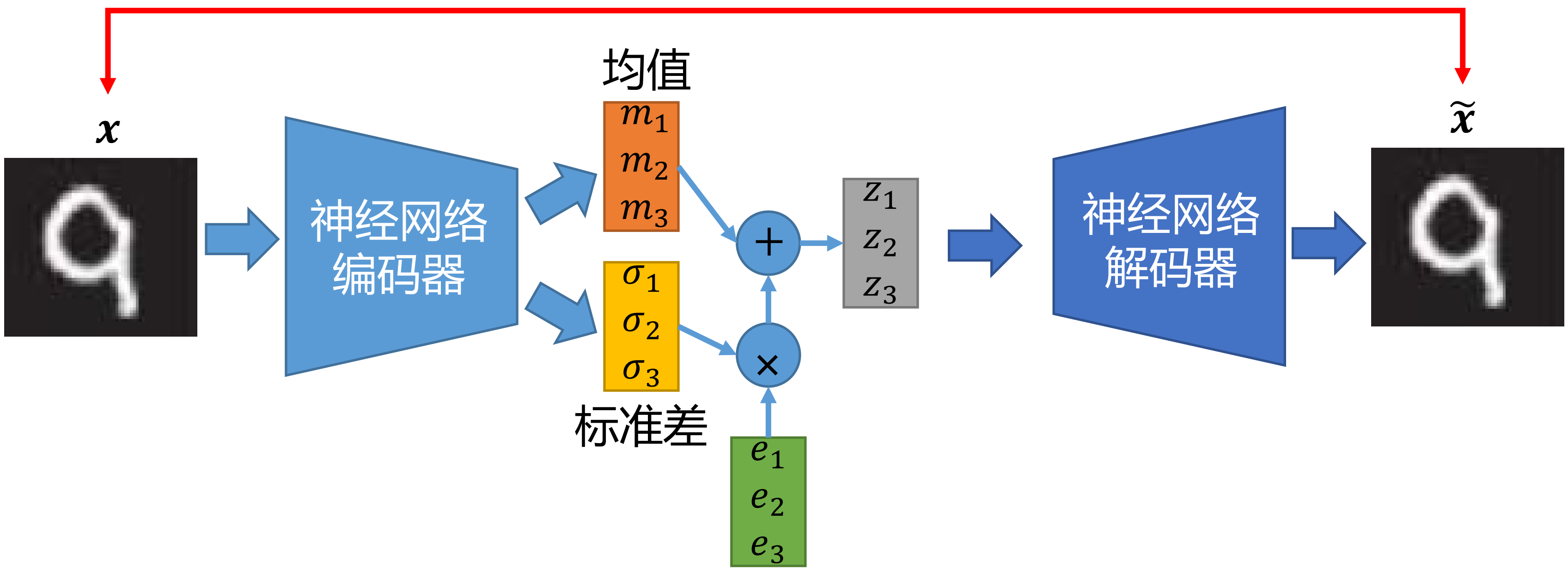


Variational AutoEncoder (VAE)

记$\sigma_i^2 = \exp(\epsilon_i)$

同时最小化 $$c = \sum_{i=1}^{3} \Big(\underbrace{\sigma_i^2 - \left(1 + \log \sigma_i^2\right)}_{\text{自动学习方差}} + \underbrace{(m_i)^2}_{\text{正则化，使其不离0远}} \Big)$$

最小化输入和输出之间的差异$(\boldsymbol{x} - \tilde{\boldsymbol{x}})^2$



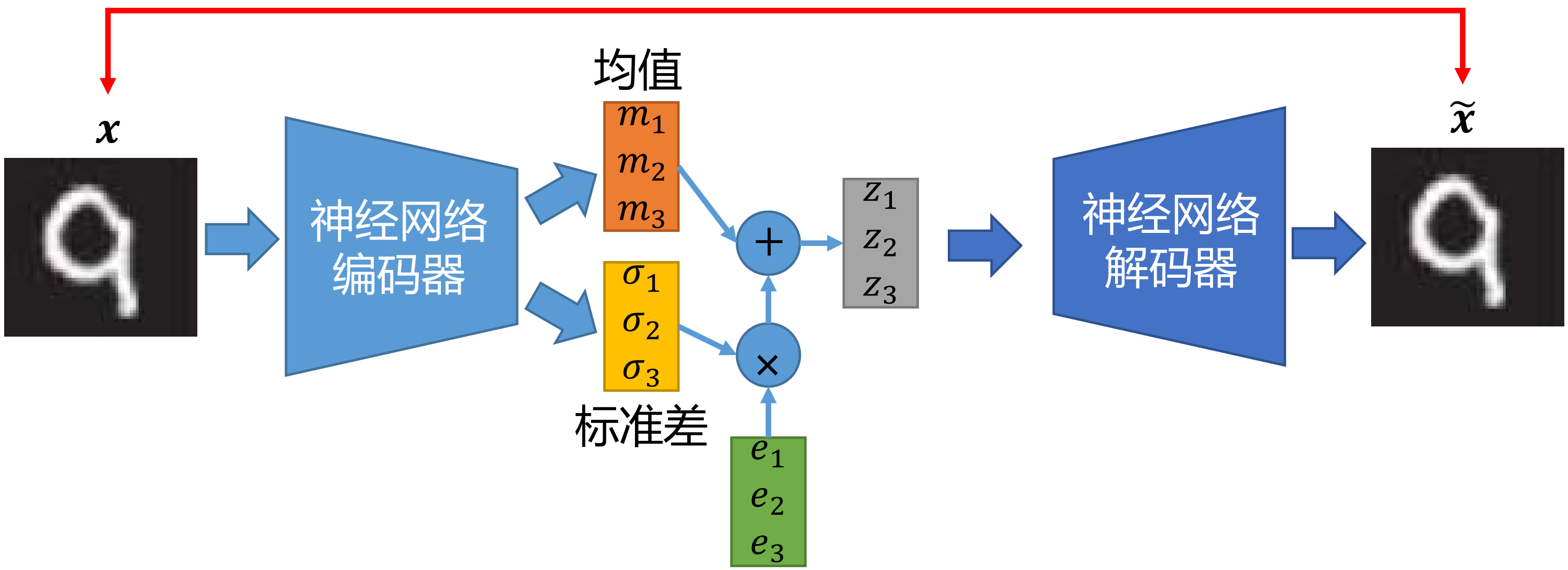
从标准正态中采样

Variational AutoEncoder (VAE)

记$\sigma_i^2 = \exp(\epsilon_i)$

同时最小化 $c = \sum_{i=1}^{3} \left(\underbrace{\exp(\epsilon_i) - (1+\epsilon_i)}_{\text{自动学习方差}} + \underbrace{(m_i)^2}_{\text{正则化，使其不离0远}} \right)$

最小化输入和输出之间的差异$(\boldsymbol{x} - \tilde{\boldsymbol{x}})^2$



从标准正态中采样

Variational AutoEncoder (VAE)

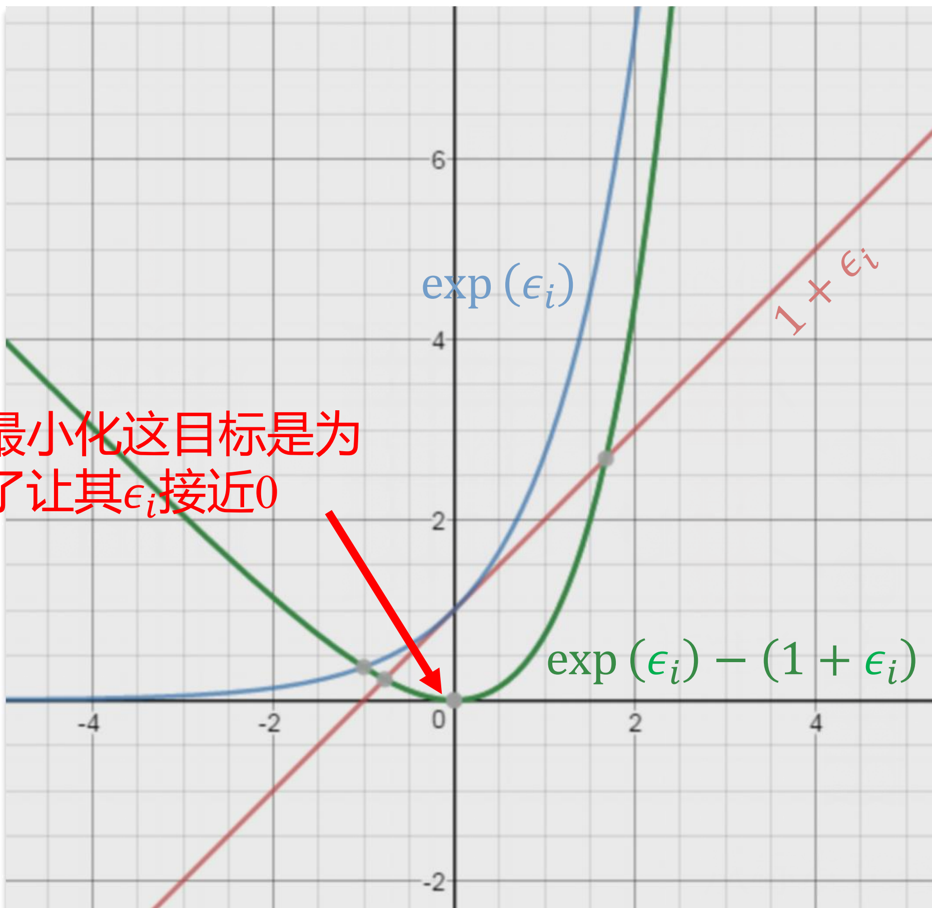
记$\sigma_i^2 = \exp(\epsilon_i)$

同时**最小化** $c = \sum_{i=1}^{3} \left(\underbrace{\exp(\epsilon_i) - (1 + \epsilon_i)}_{\text{自动学习方差}} + \underbrace{(m_i)^2}_{\text{正则化，使其不离0远}} \right)$

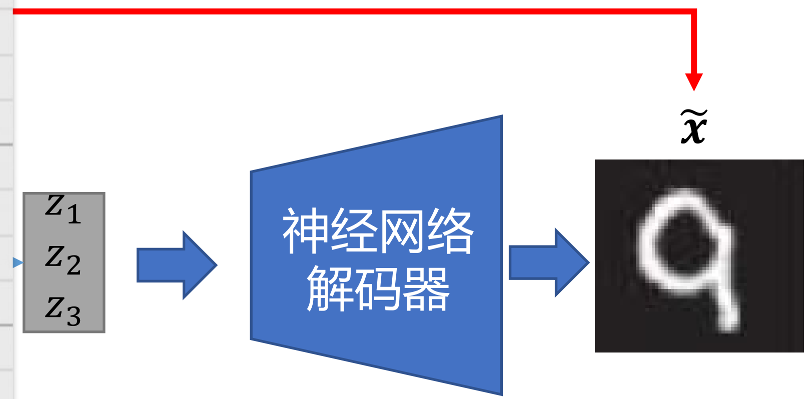
自动学习方差

正则化，使其不离0远

的差异$(\boldsymbol{x} - \tilde{\boldsymbol{x}})^2$



最小化这目标是为了让其ϵ_i接近0



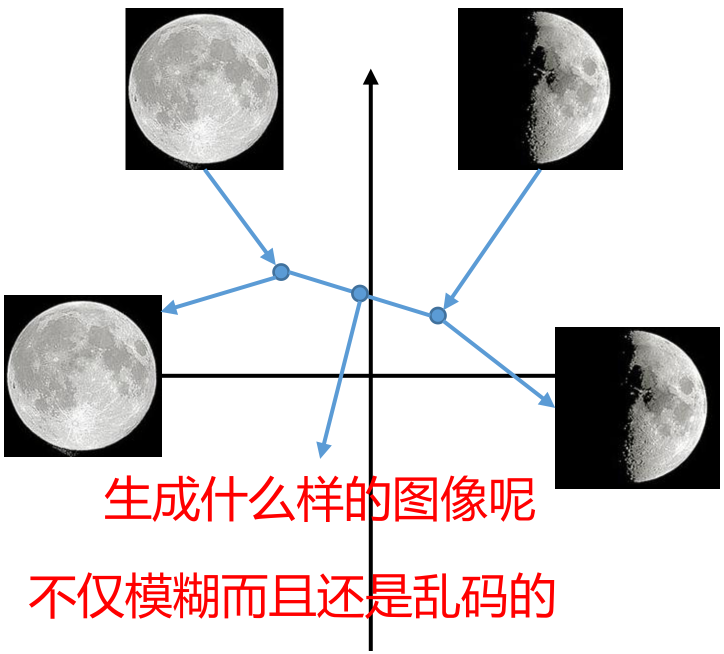
中采样

一个例子理解VAE

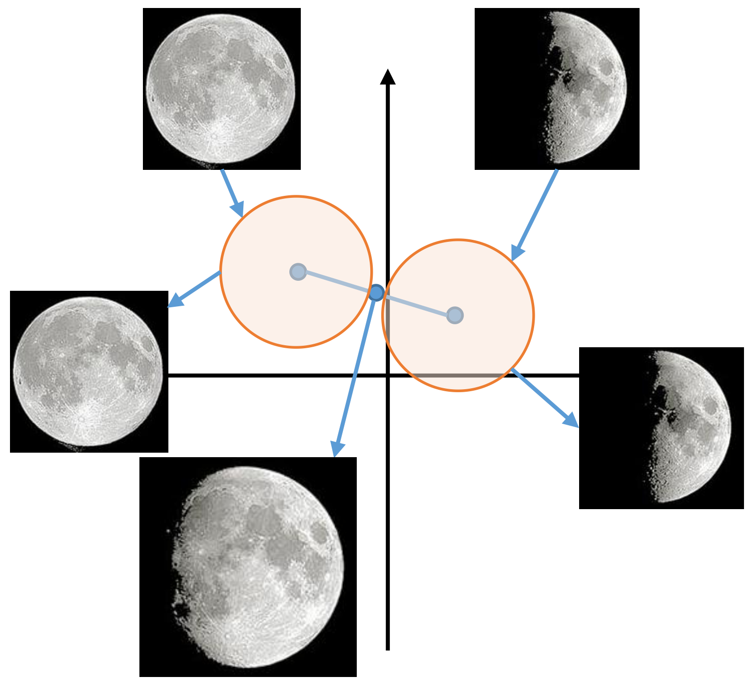
- **为什么引入噪声和重构误差损失?**
- 把VAE正在做的这件事情比作是在参加高考：
  - 一般为了能够真正在考场取得好成绩（模型预测），
  - 学生在平常的学习生活中需要做各式各样的测试（模型训练）。
  - 这些测试题的考试难度（方差编码σ）应该由老师来定，因为只有这样才能客观的检验学生的学习能力。
  - 假如没有老师监督（辅助loss），而是让学生（模型）来决定考题的难度（分配给噪声的权重），
  - 那么学生肯定是偏向让测试题难度降到最低（使噪声影响最小），最好一点难度（噪声）都没有，从而能够考满分（使最终的重构误差为0）。
  - 因此，为了能够真正在高考上取得好成绩，而不是以这种投机取巧的方式，所以必须引入老师（辅助的loss）这个中间人来监督这个课堂测试（训练过程）的难度。

VAE V.S. AE

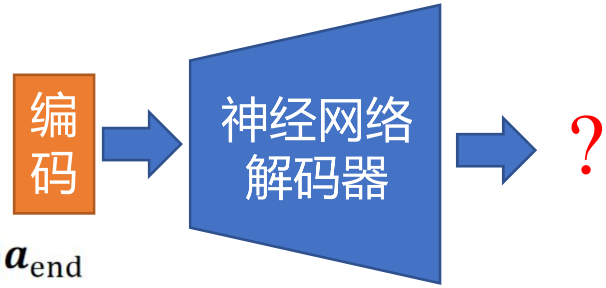
AutoEncoder



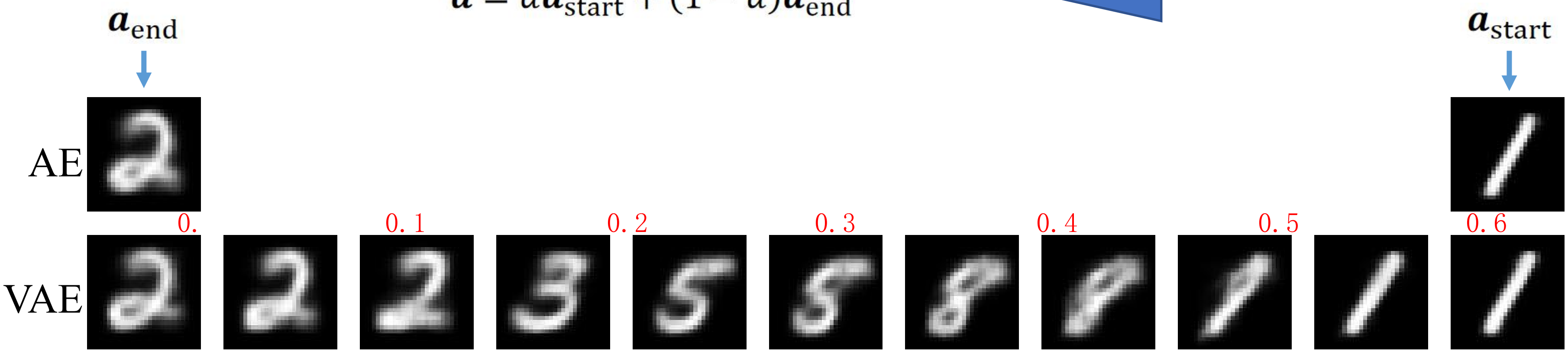
Variational AutoEncoder

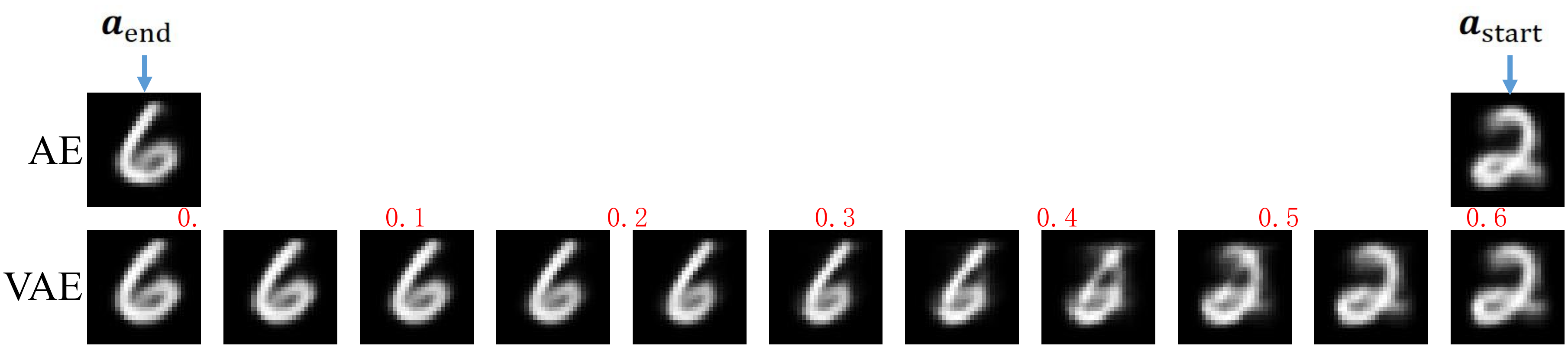


VAE V.S. AE



$$\boldsymbol{a} = \alpha \boldsymbol{a}_{\text{start}} + (1-\alpha)\boldsymbol{a}_{\text{end}}$$





高斯混合模型

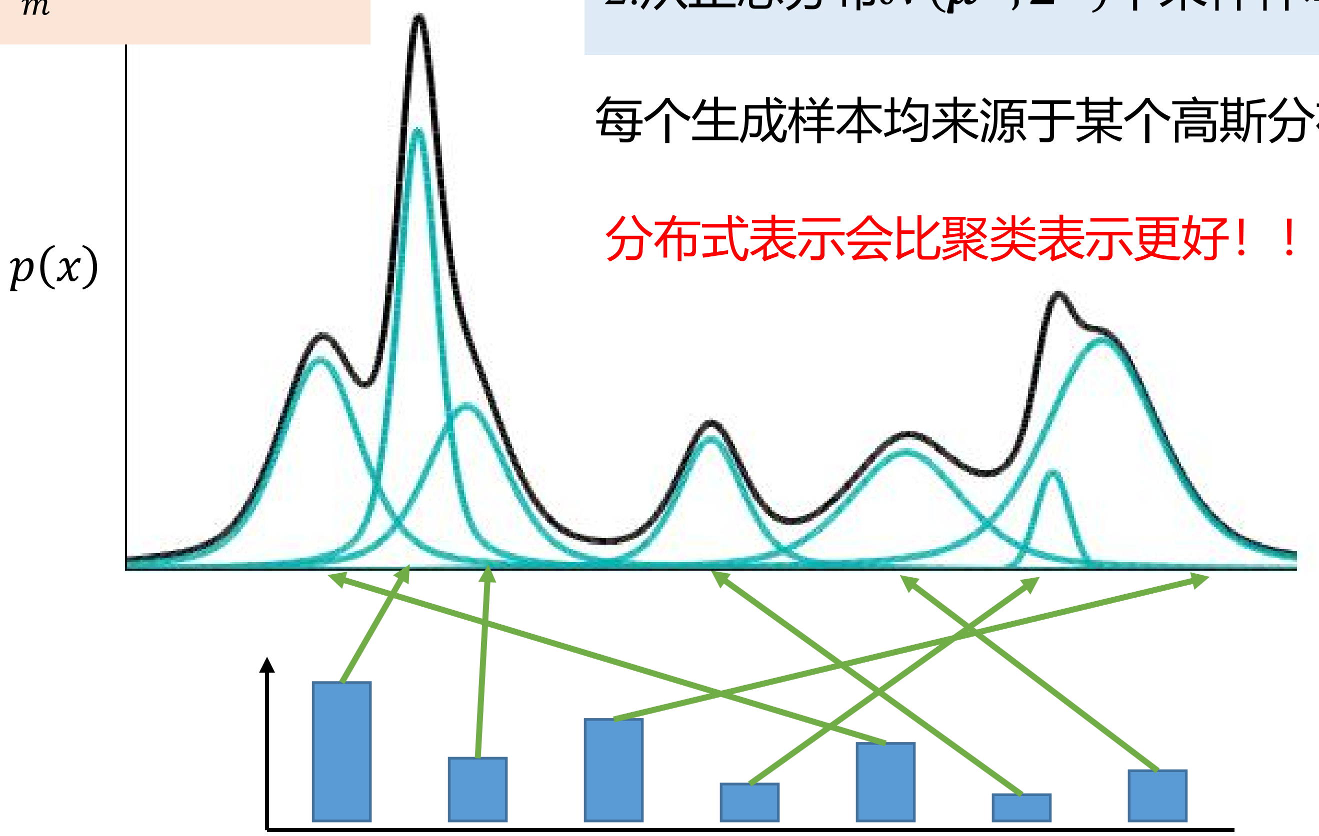
$$p(x) = \sum_{m} P(m)p(x|m)$$

样本生成过程

1.先从多项分布中$P(m)$采样一个m
2.从正态分布$\mathcal{N}(\boldsymbol{\mu}^m, \boldsymbol{\Sigma}^m)$中采样样本$\boldsymbol{x}$

每个生成样本均来源于某个高斯分布

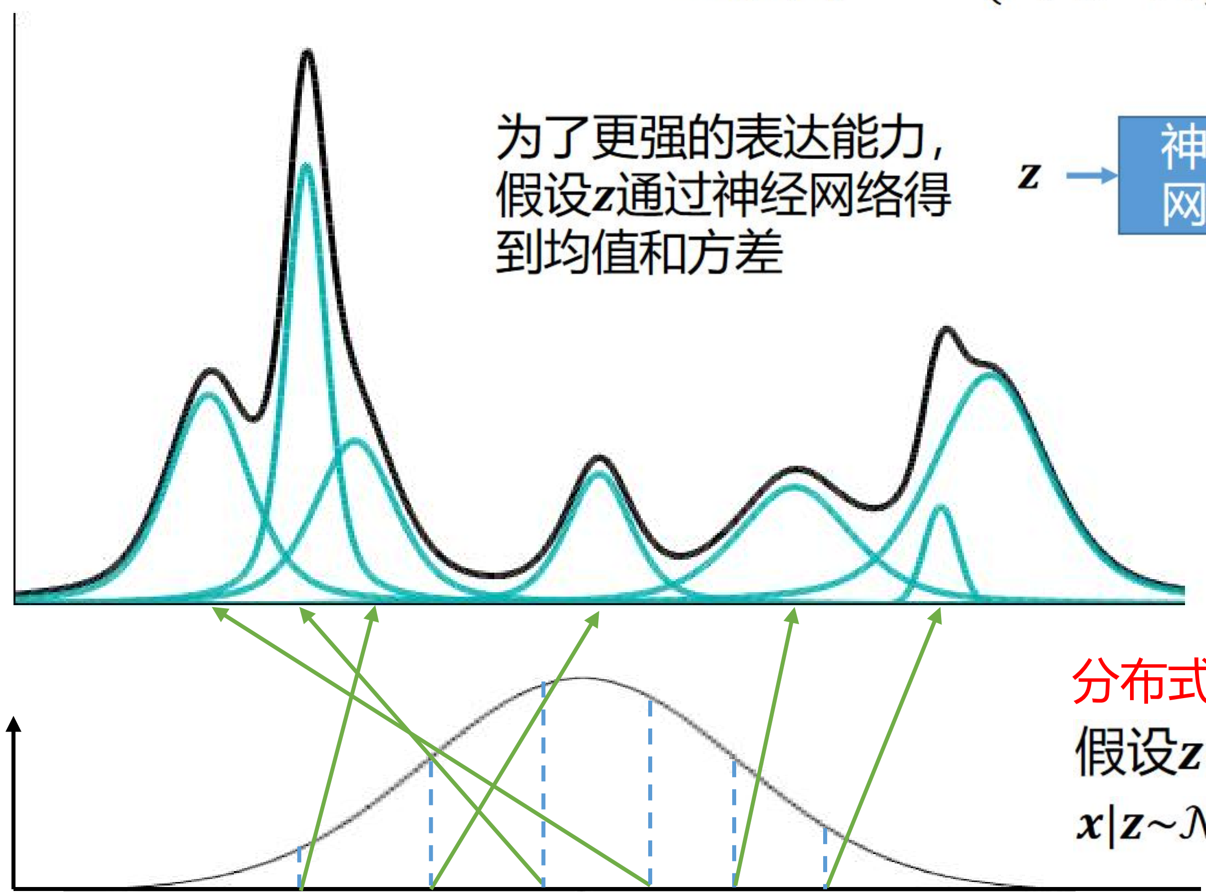
分布式表示会比聚类表示更好！！！



无限高斯混合模型

$$P(\boldsymbol{x}) = \int \underbrace{P(\mathbf{z})}_{\mathcal{N}(\mathbf{0},\boldsymbol{I})} \underbrace{P(\boldsymbol{x}|\mathbf{z})}_{\mathcal{N}(\boldsymbol{\mu}(\mathbf{z}),\boldsymbol{\sigma}(\mathbf{z}))} d\mathbf{z}$$

为了更强的表达能力，假设z通过神经网络得到均值和方差



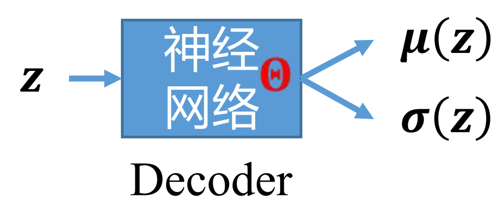
分布式表示

假设$\mathbf{z} \sim \mathcal{N}(\mathbf{0},\boldsymbol{I})$

$\boldsymbol{x}|\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{z}),\boldsymbol{\sigma}(\mathbf{z}))$

VAE最大化似然概率

$$P(\boldsymbol{x}) = \int \underbrace{P(\boldsymbol{z})}_{\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}\underbrace{P(\boldsymbol{x}|\boldsymbol{z})}_{\mathcal{N}\left(\boldsymbol{\mu}(\boldsymbol{z}),\boldsymbol{\sigma}(\boldsymbol{z})\right)}d\boldsymbol{z}$$



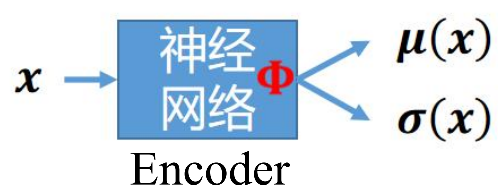
优化目标：$\log P(\boldsymbol{x}) = \log \int P(\boldsymbol{z})P(\boldsymbol{x}|\boldsymbol{z})d\boldsymbol{z}$
优化参数：神经网络中的参数$\boldsymbol{\Theta}$

当$Q(\boldsymbol{z}) = P(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Theta})$时等式成立

$$\ln P(\boldsymbol{x}|\boldsymbol{\Theta}) = \ln \int Q(\boldsymbol{z})\frac{P(\boldsymbol{x},\boldsymbol{z}|\boldsymbol{\Theta})}{Q(\boldsymbol{z})} \geq \underbrace{\int Q(\boldsymbol{z})\ln\frac{P(\boldsymbol{x},\boldsymbol{z}|\boldsymbol{\Theta})}{Q(\boldsymbol{z})}}_{\mathcal{L}(Q,\boldsymbol{\Theta})}$$

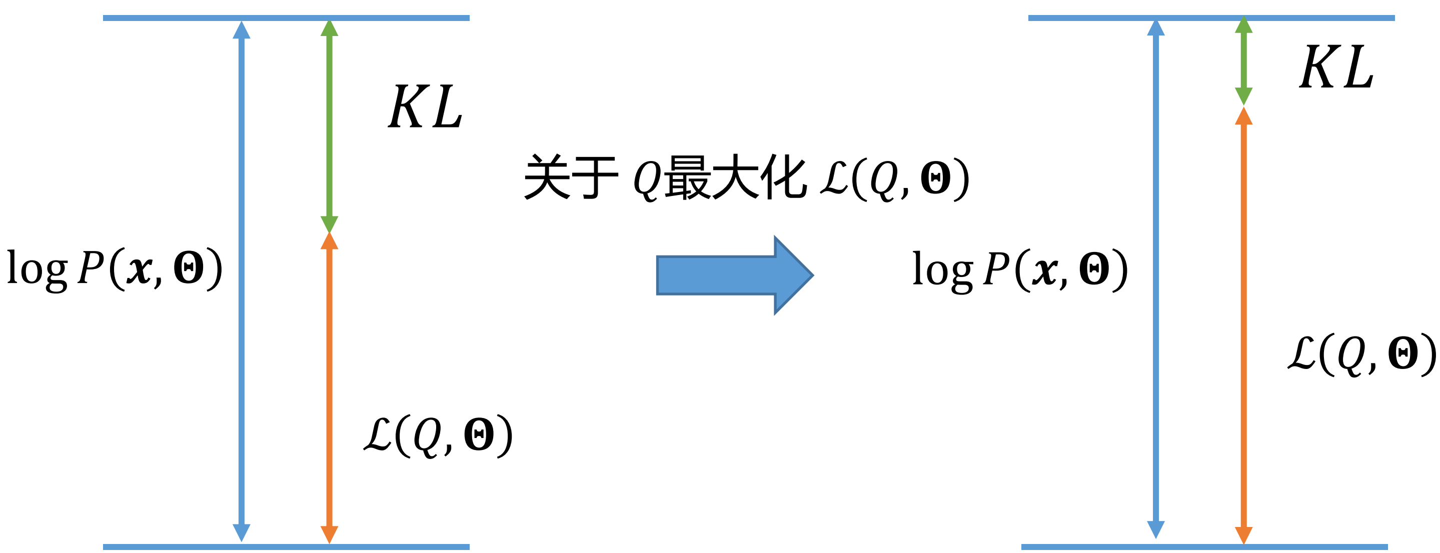
由于 $P(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Theta})$ 通常难以计算，假设其为正态分布，记为$Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi})$

用神经网络近似其参数



VAE最大化似然概率

$$\mathcal{L}(Q,\mathbf{\Theta}) = \int Q(\mathbf{z}|\boldsymbol{x},\mathbf{\Phi}) \ln \frac{P(\boldsymbol{x},\mathbf{z}|\mathbf{\Theta})}{Q(\mathbf{z}|\boldsymbol{x},\mathbf{\Phi})} \quad = \ln P(\boldsymbol{x}|\mathbf{\Theta}) - KL\big(Q(\mathbf{z}|\boldsymbol{x},\mathbf{\Phi})||P(\mathbf{z}|\boldsymbol{x},\mathbf{\Theta})\big)$$



关于 Q最大化 $\mathcal{L}(Q,\mathbf{\Theta})$ 会使得$Q(\mathbf{z}|\boldsymbol{x},\mathbf{\Phi})$与$P(\mathbf{z}|\boldsymbol{x},\mathbf{\Theta})$ 差异减小
即$Q(\mathbf{z}|\boldsymbol{x},\mathbf{\Phi})$是$P(\mathbf{z}|\boldsymbol{x},\mathbf{\Theta})$的一个好的近似

VAE最大化似然概率

$$\ln P(\boldsymbol{x}|\boldsymbol{\Theta}) = \ln \int Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi}) \frac{P(\boldsymbol{x},\boldsymbol{z}|\boldsymbol{\Theta})}{Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi})}$$

$$\geq \underbrace{\int Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi}) \ln \frac{P(\boldsymbol{x},\boldsymbol{z}|\boldsymbol{\Theta})}{Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi})}}_{\mathcal{L}(Q,\boldsymbol{\Theta})} \qquad = \int Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi}) \ln \frac{P(\boldsymbol{x},\boldsymbol{z}|\boldsymbol{\Theta})}{Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi})}$$

$$= \int Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi}) \ln \frac{P(\boldsymbol{x}|\boldsymbol{z},\Theta)P(\boldsymbol{z})}{Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi})}$$

$$= \int Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi}) \ln P(\boldsymbol{x}|\boldsymbol{z},\Theta) + \underbrace{\int Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi}) \ln \frac{P(\boldsymbol{z})}{Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi})}}_{-KL\left(Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi})||P(\boldsymbol{z})\right)}$$

VAE最大化似然概率

$$\mathcal{L}(Q,\boldsymbol{\Theta}) = \int Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi}) \ln P(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{\Theta}) - KL\big(Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi})||P(\boldsymbol{z})\big)$$

$\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$

$$\mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{\exp\left(-1/2(\boldsymbol{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}|}}$$

用神经网络近似其参数

$\boldsymbol{x} \rightarrow$ 神经网络 $\boldsymbol{\Phi}$ $\rightarrow$ $\boldsymbol{\mu}(\boldsymbol{x})$, $\log\boldsymbol{\sigma}^2(\boldsymbol{x})$

$$\begin{aligned}
&KL\big(\mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu},\boldsymbol{\sigma}^2)||\mathcal{N}(\boldsymbol{z}|\boldsymbol{0},\boldsymbol{I})\big)\\
&= \int \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu},\boldsymbol{\sigma}^2)\log\frac{\mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu},\boldsymbol{\sigma}^2)}{\mathcal{N}(\boldsymbol{z}|\boldsymbol{0},\boldsymbol{I})}\\
&= \int \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu},\boldsymbol{\sigma}^2)\left(-\frac{1}{2}\log|\mathrm{diag}(\boldsymbol{\sigma}^2)| - \frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu})^\top\mathrm{diag}(\boldsymbol{\sigma}^2)^{-1}(\boldsymbol{z}-\boldsymbol{\mu}) + \frac{1}{2}\boldsymbol{z}^\top\boldsymbol{z}\right)\\
&= -\frac{1}{2}\log|\mathrm{diag}(\boldsymbol{\sigma}^2)| - \frac{D}{2} + \frac{1}{2}\mathrm{tr}\big(\boldsymbol{\mu}\boldsymbol{\mu}^\top + \mathrm{diag}(\boldsymbol{\sigma}^2)\big)\\
&= -\frac{1}{2}\sum_i \log\sigma_i^2 - \frac{D}{2} + \frac{1}{2}\left(\sum_i \mu_i^2 + \sum_i \sigma_i^2\right)\\
&= \frac{1}{2}\sum_i\left(-1-\log\sigma_i^2+\mu_i^2+\sigma_i^2\right)
\end{aligned}$$

$\boldsymbol{z}$为D维向量

$$\begin{aligned}
&\mathbb{E}_{\mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu},\boldsymbol{\Sigma})}[(\boldsymbol{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})]\\
&= \mathrm{tr}\big(\boldsymbol{\Sigma}^{-1}\mathbb{E}_{\mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu},\boldsymbol{\Sigma})}[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^\top]\big)\\
&= \mathrm{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma})\\
&= D
\end{aligned}$$

VAE最大化似然概率

最大化 $\mathcal{L}(Q, \boldsymbol{\Theta}) = \underbrace{\int Q(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\Phi}) \ln P(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\Theta})}_{} - \underbrace{KL\big(Q(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\Phi}) || P(\boldsymbol{z})\big)}_{}$

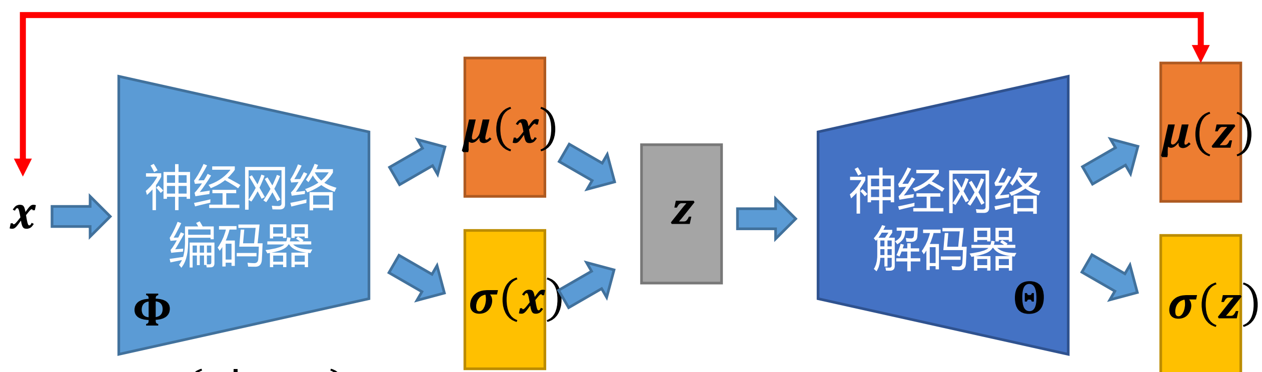
$$\mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\Phi})}[\ln P(\boldsymbol{x}|\boldsymbol{z}, \Theta)]$$

$$\approx \frac{1}{L}\sum_{l} \ln P(\boldsymbol{x}|\boldsymbol{z}_l, \Theta) \qquad \boldsymbol{z}_l \sim Q(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\Phi})$$

$$\sim \mathcal{N}\big(\boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{\sigma}(\boldsymbol{x})\big)$$

$$\frac{1}{2}\sum_{i}\left(-1 - \log \sigma_i^2 + \mu_i^2 + \sigma_i^2\right)$$

最小化输入和输出之间的差异$(\boldsymbol{x} - \tilde{\boldsymbol{x}})^2$ 同时最小化



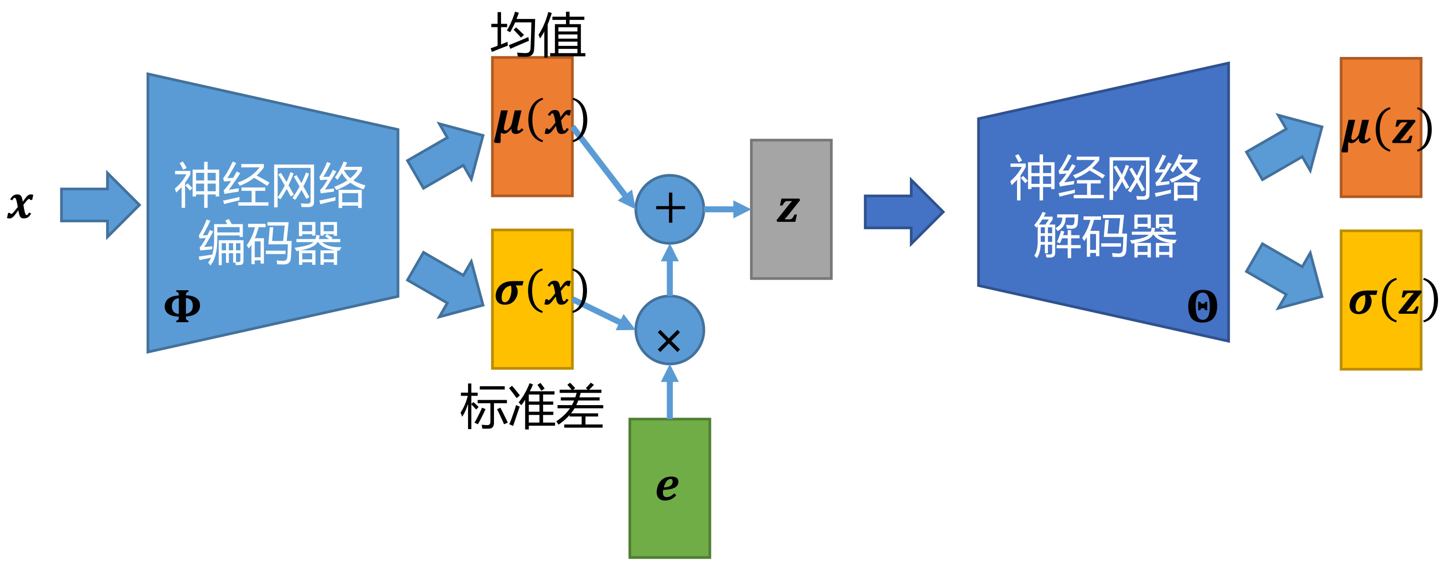
$Q(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\Phi})$ $\qquad$ $P(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\Theta})$

直接从$\mathcal{N}\big(\boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{\sigma}(\boldsymbol{x})\big)$采样编码会导致梯度无法回传

VAE的重采样技术

直接从$\mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}),\boldsymbol{\sigma}(\boldsymbol{x}))$采样$\boldsymbol{z}$　　等价于　　从$\mathcal{N}(\mathbf{0},\boldsymbol{I})$ 采样$\boldsymbol{e}$ 后，运用如下变换

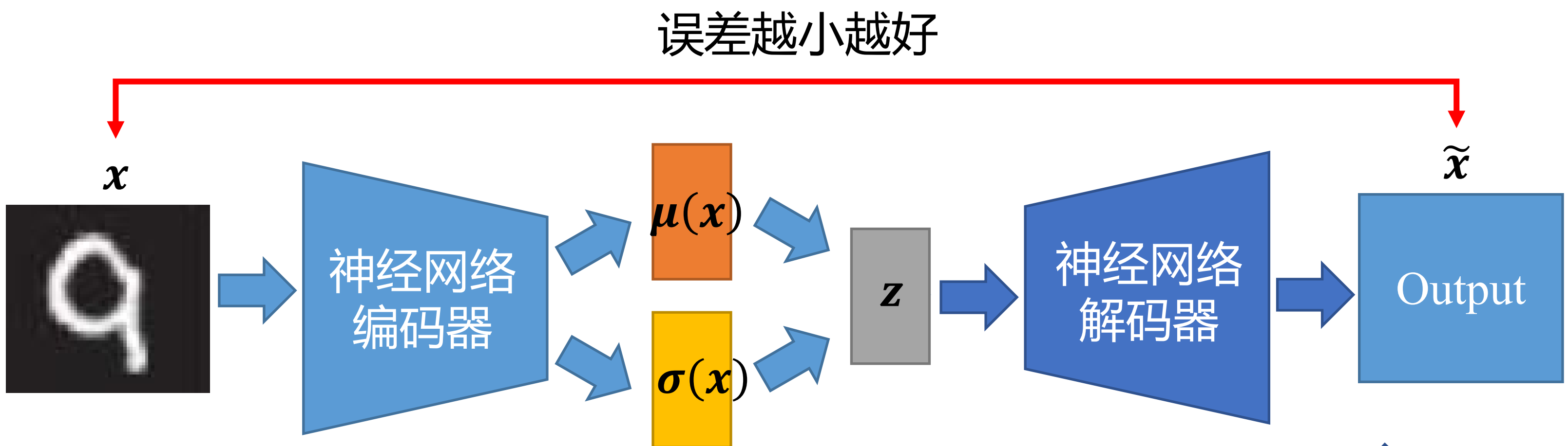
$$\boldsymbol{z}=\boldsymbol{e}\odot\boldsymbol{\sigma}(\boldsymbol{x})+\boldsymbol{\mu}(\boldsymbol{x})$$

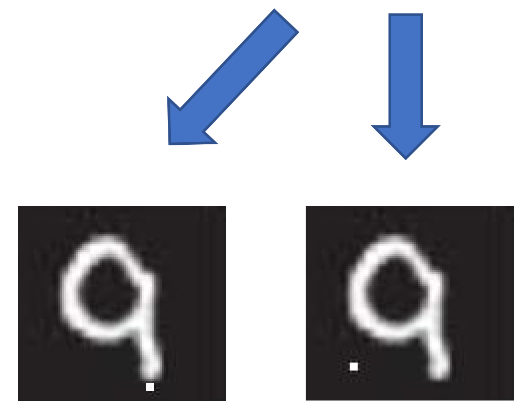


从标准正态中采样

VAE生成出来的图像

VAE的问题





这两张图片和原始图片只有一个像素的差异

这两图片对VAE来说没有差别

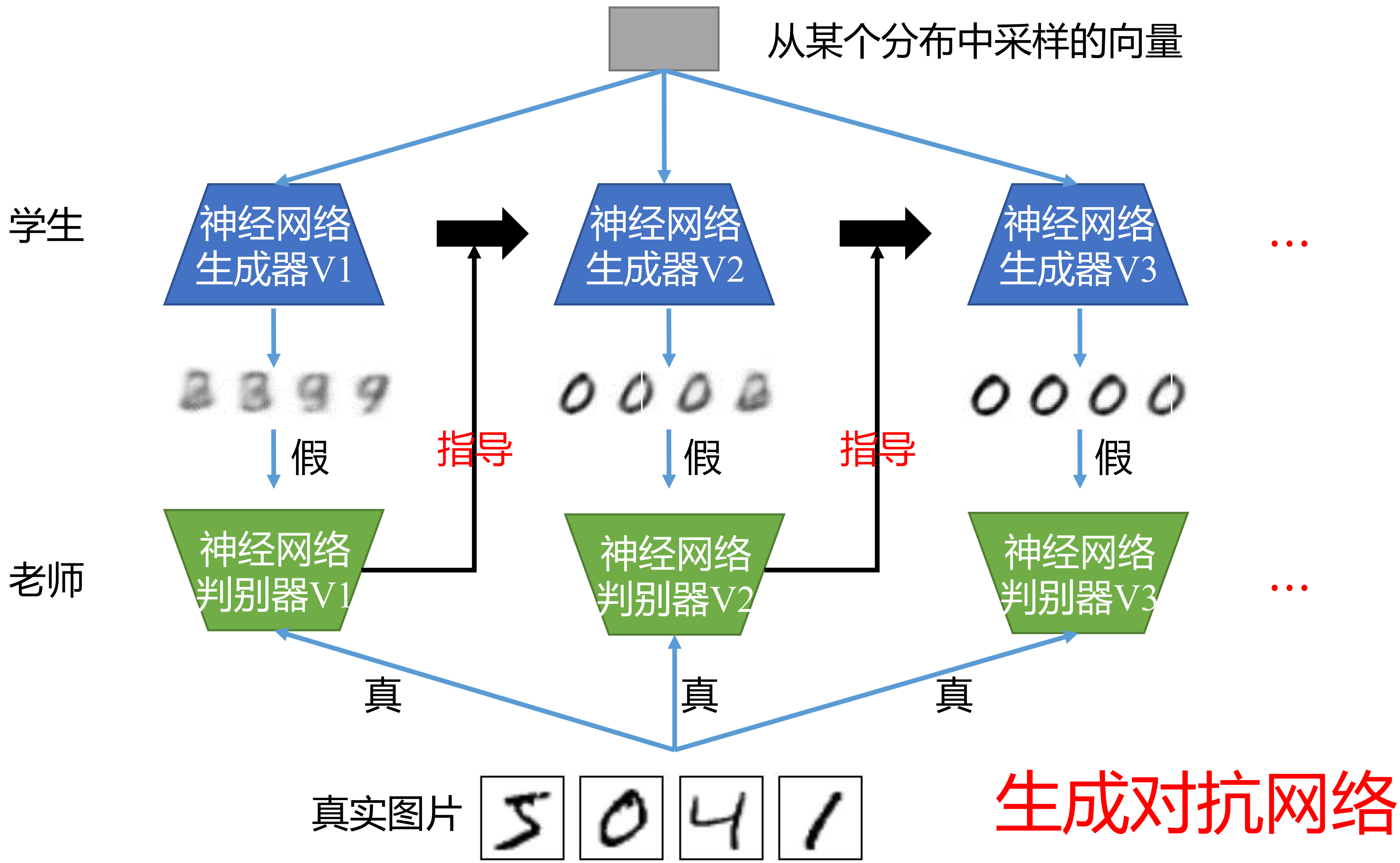
和右边对比，左边更加真实

VAE只是记住了数据库里面的图片，并没有学习到如何生成新图片

VAE生成的图片

VAE模型往往产生不现实的、模糊的样本。这是由于它是通过直接计算生成图片和原始图片之间的均方误差来保证生成图片的真实性，所以得到的是一张“平均图像”。

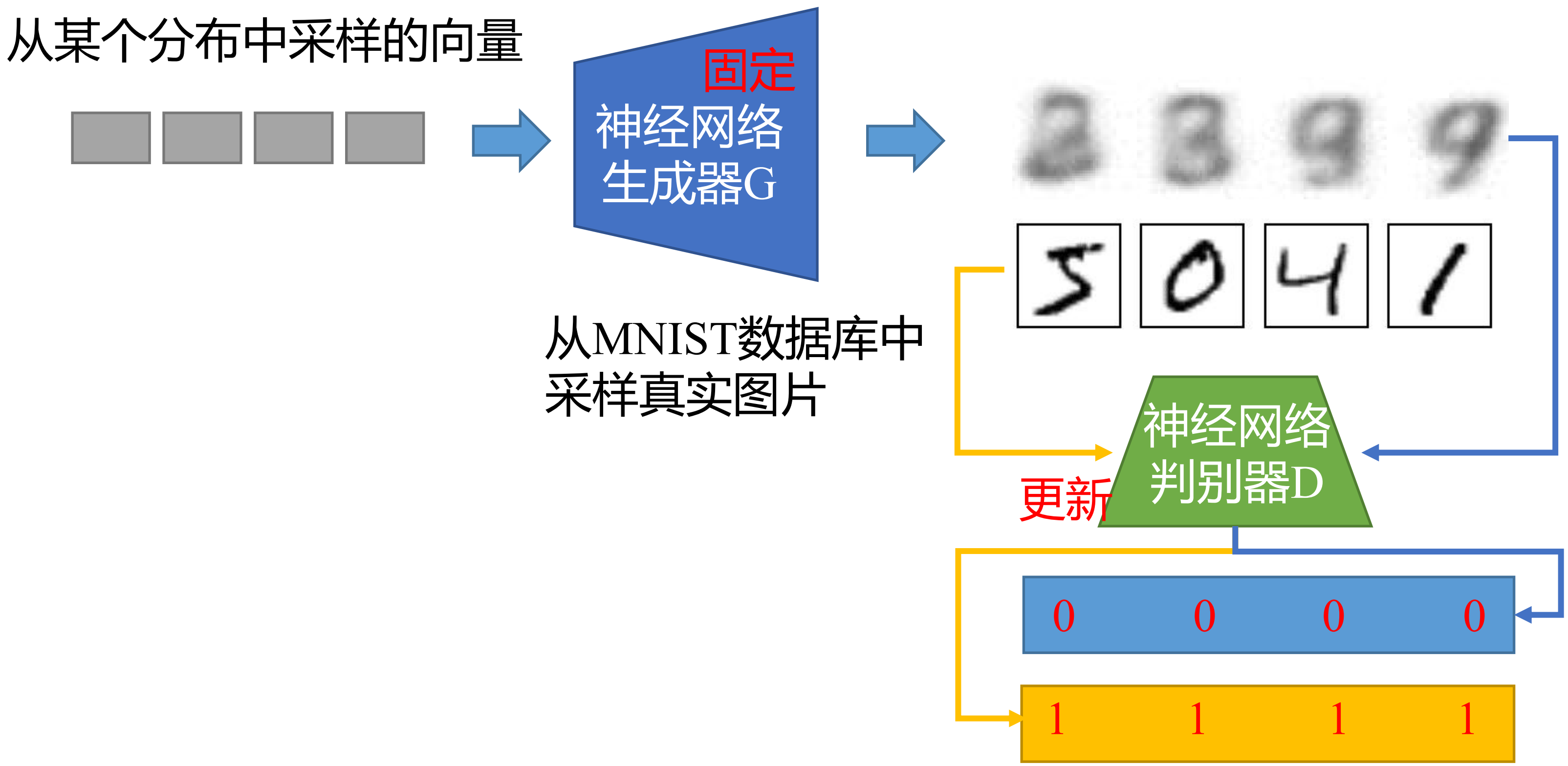
解码器训练的新思路—引入判别器



生成对抗网络的算法

- 初始化生成器和判别器
- 在每个训练步
- 1）固定生成器，更新判别器D

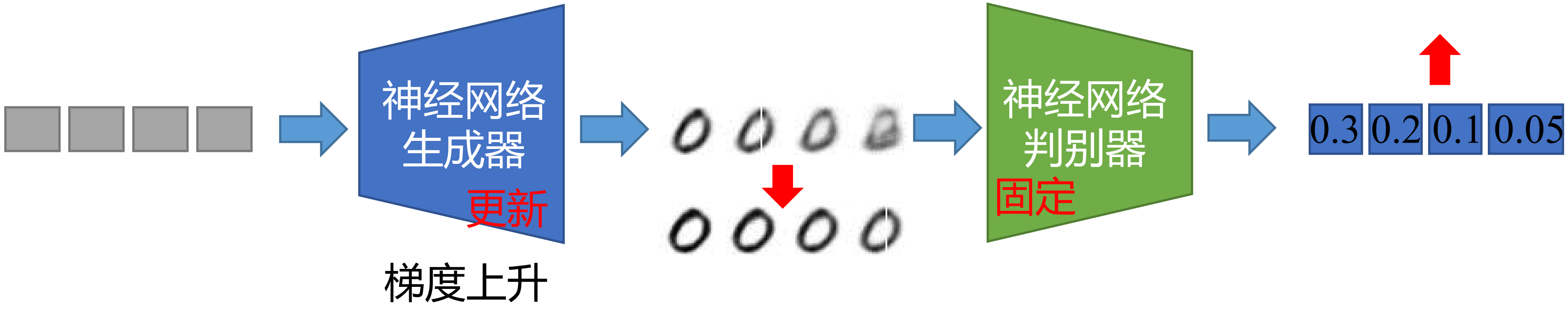
学会给真实图片高分，给生成图片低分



生成对抗网络的算法

- 初始化生成器和判别器
- 在每个训练步
- 2）固定判别器，更新生成器G

生成得分更高的图片
让判别器更易混淆的图片



生成对抗网络的算法

- 随机初始化判别器参数 $\boldsymbol{\theta}_d$和生成器参数 $\boldsymbol{\theta}_g$
- While not convergent:

更新D

- 从数据库中采样 m 样本$\{\boldsymbol{x}^1, \boldsymbol{x}^2, \dots, \boldsymbol{x}^m\}$
- 从某个分布中采样m个噪声样本 $\{\boldsymbol{z}^1, \boldsymbol{z}^2, \dots, \boldsymbol{z}^m\}$
- 用生成器生成样本 $\{\tilde{\boldsymbol{x}}^1, \tilde{\boldsymbol{x}}^2, \dots, \tilde{\boldsymbol{x}}^m\}$, $\tilde{\boldsymbol{x}}^i = G(\boldsymbol{z}^i)$
- 更新判别器参数 $\boldsymbol{\theta}_d$，通过最大化如下目标

$$\tilde{\mathcal{L}} = \frac{1}{m}\sum_{i=1}^{m} \log D\left(\boldsymbol{x}^i\right) + \frac{1}{m}\sum_{i=1}^{m} \log\left(1 - D\left(\tilde{\boldsymbol{x}}^i\right)\right)$$

$$\boldsymbol{\theta}_d \leftarrow \boldsymbol{\theta}_d + \eta \nabla \tilde{\mathcal{L}}(\boldsymbol{\theta}_d)$$

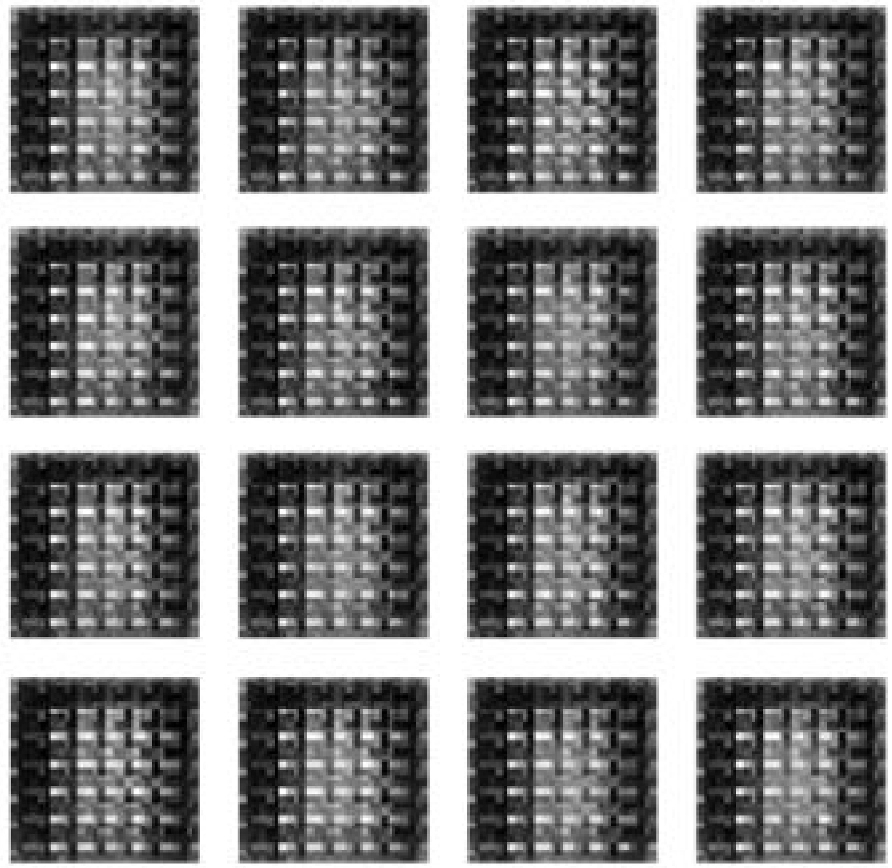
更新G

- 从某个分布中采样m个噪声样本 $\{\boldsymbol{z}^1, \boldsymbol{z}^2, \dots, \boldsymbol{z}^m\}$
- 更新生成器参数 $\boldsymbol{\theta}_g$，通过最大化如下目标

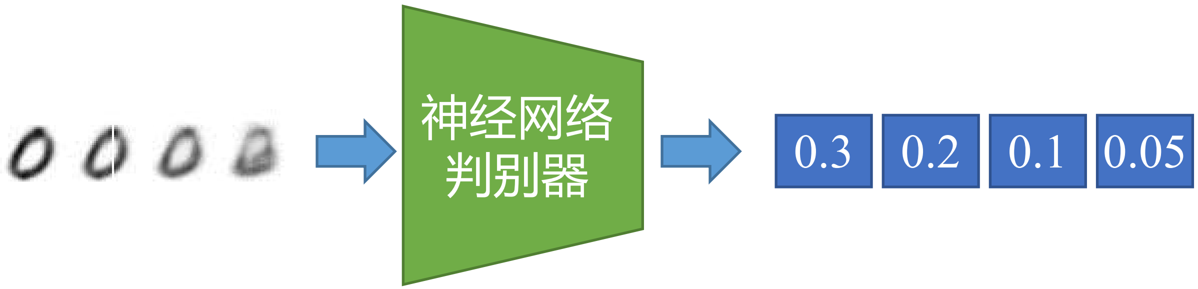
$$\tilde{\mathcal{L}} = \frac{1}{m}\sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^i\right)\right)\right)$$

$$\boldsymbol{\theta}_g \leftarrow \boldsymbol{\theta}_g + \eta \nabla \tilde{\mathcal{L}}(\boldsymbol{\theta}_g)$$

生成对抗网络用于MNIST数据集



判别器D的更多理解—假设只有判别器



输入一个对象$\boldsymbol{x}$

输出一个标量，代表输入$\boldsymbol{x}$的好坏

能否用判别器生成对象呢？　答案是可以的！！

判别器D的更多理解—假设只有判别器

- 假设$D(\boldsymbol{x})$已经训练的很好，那么可以通过如下方式生成图像

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\in\mathcal{X}} D(\boldsymbol{x})$$

- 即找到 判别器 打分最高（最真实）的图片
  - 可以枚举所有的$\boldsymbol{x} \in \mathcal{X}$找最高分的$\boldsymbol{x}^*$
  - 也可以通过梯度上升法 $\boldsymbol{x}^t \leftarrow \boldsymbol{x}^t + \eta D(\boldsymbol{x}^t)$

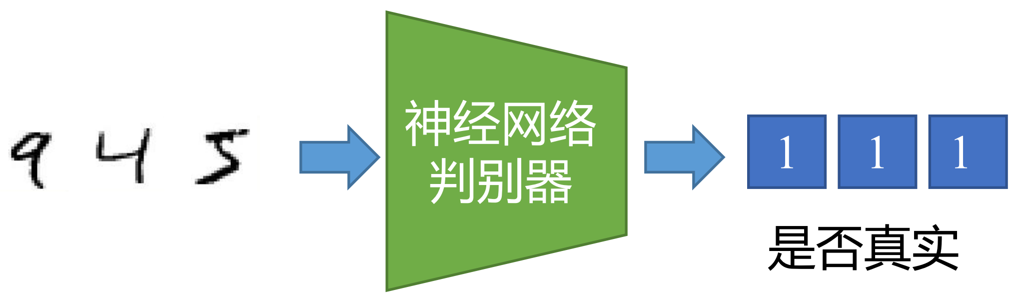
但是如何学习判别器呢？

判别器D的更多理解—假设只有判别器

- 在没有生成器的情况下，如何学习判别器呢

- 我们只有真实的图像，如果只用真实数据训练，

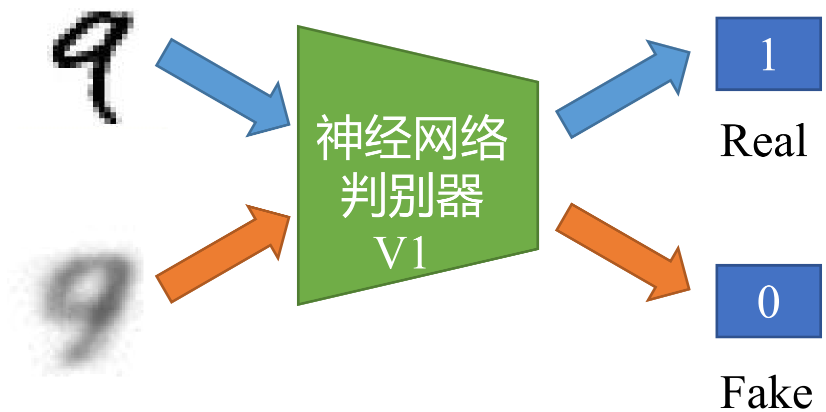
那么判别器会学习到总是输出1



- 因此，判别器学习需要负样本，即不真实样本

判别器D的更多理解—假设只有判别器

• 要训练好判别器，负样本非常关键

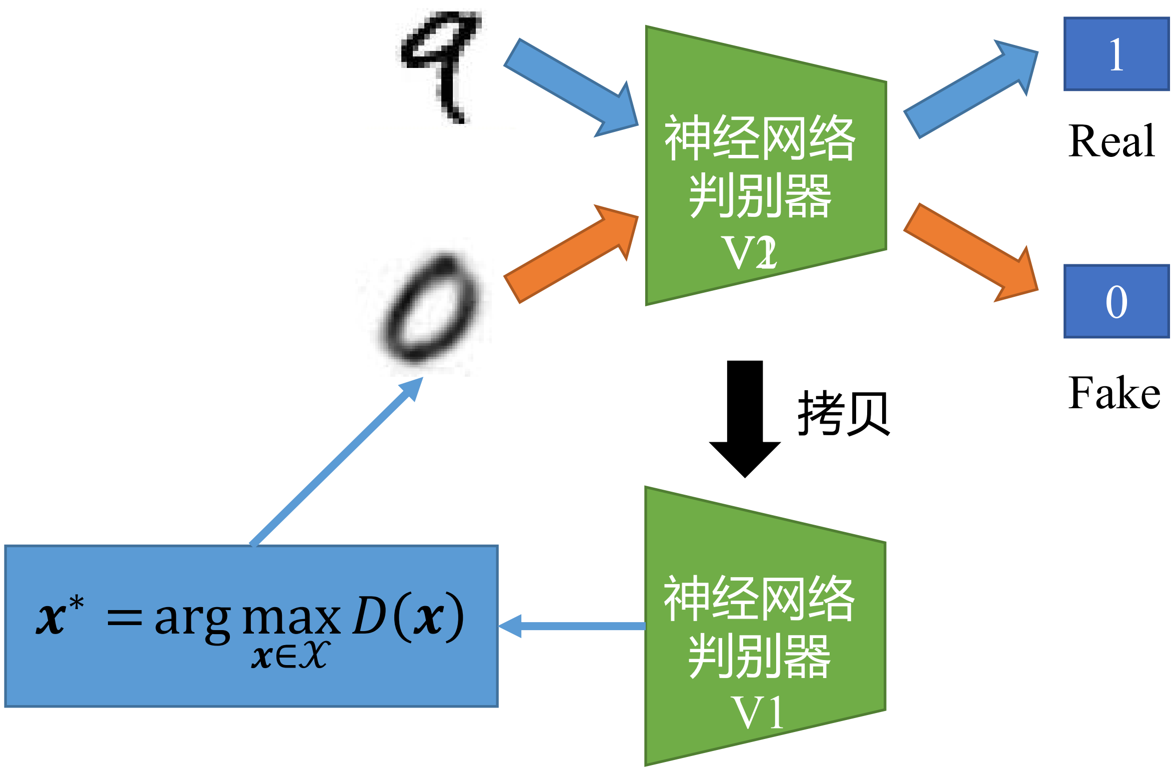


在训练完后，输入

输出这么高的分
是因为训练时负样本太烂了，
使得判别器会误认为已经非常好了

判别器D的更多理解—假设只有判别器

- 要训练好判别器，负样本非常关键

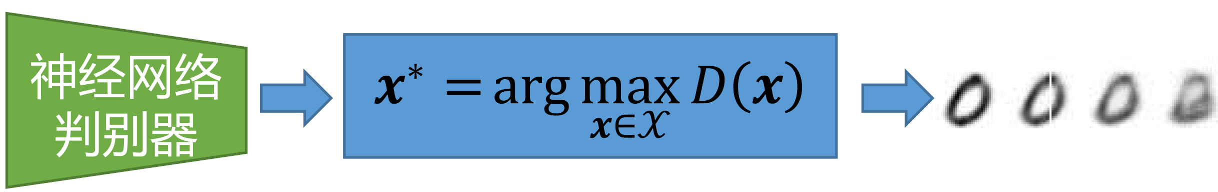


判别器D的更多理解—假设只有判别器

在没有生成器的情况下判别器D的训练算法

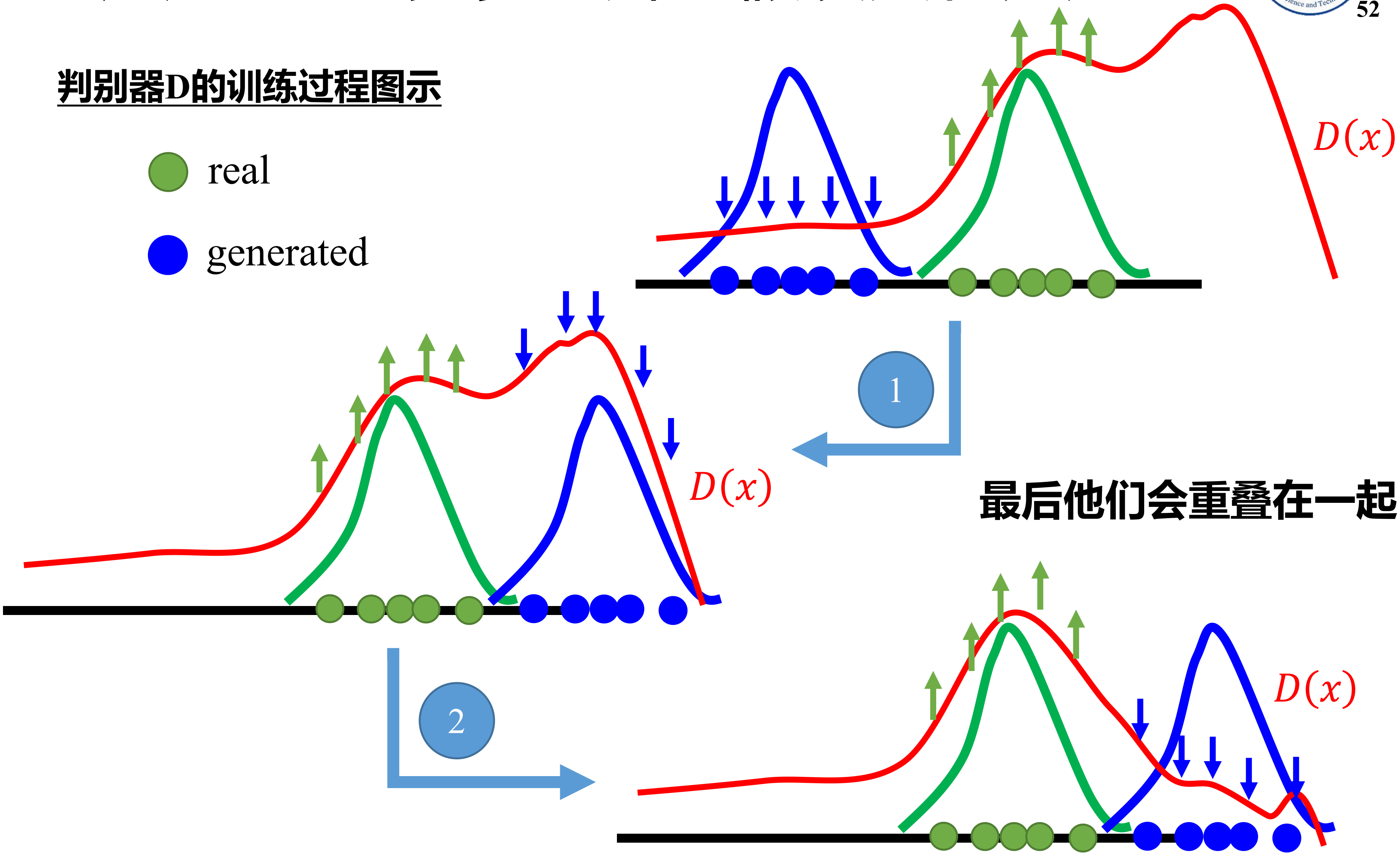
- 给定真实样本集，随机生成负样本集
- While not convergent
  - 学习判别器D，使得给正样本高分，负样本低分

  - 用判别器D 生成负样本集



判别器D的更多理解—假设只有判别器

判别器D的训练过程图示



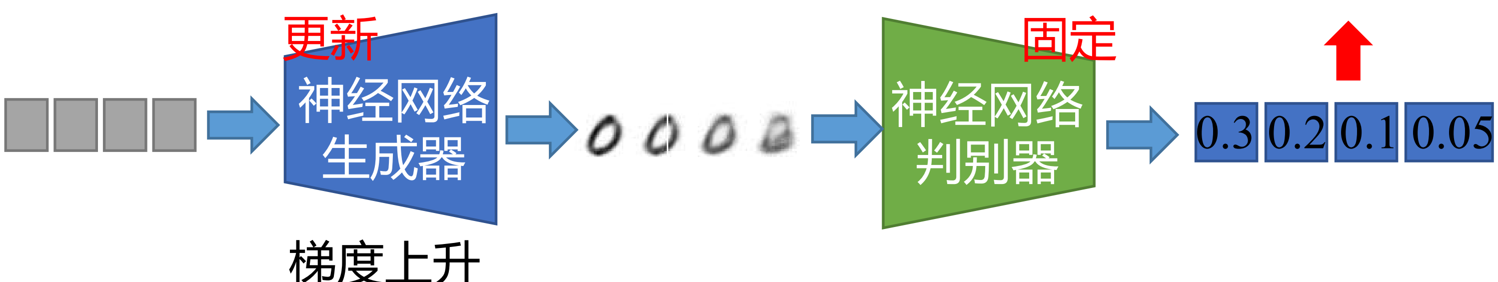
判别器+生成器

GAN的训练算法

- 给定真实样本集，随机生成负样本集
- While not convergent
  - 学习判别器D，使得给正样本高分，负样本低分

  - 用判别器D 生成负样本集

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\in\mathcal{X}} D(\boldsymbol{x})$$



GAN的优势

生成器

- 优点
    - 容易生成样本
- 缺点
    - 缺乏全局观

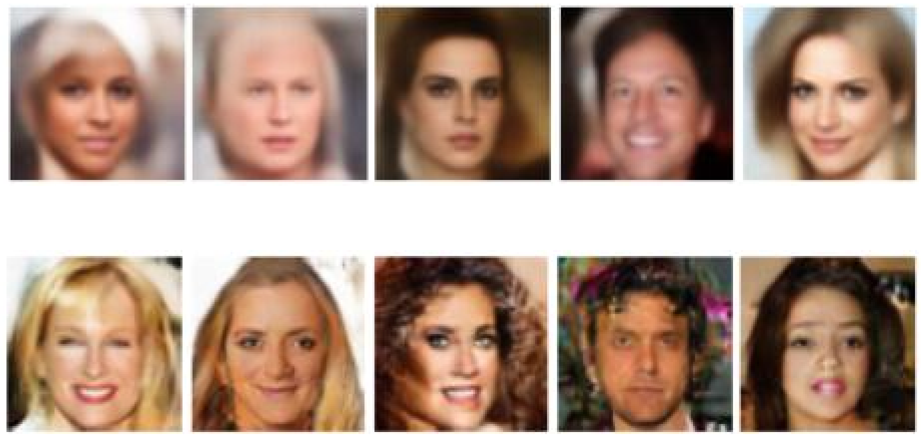
判别器

- 优点
    - 有全局意识
- 缺点
    - 难生成

综合了生成器和判别器的优点，弥补了各自的缺点

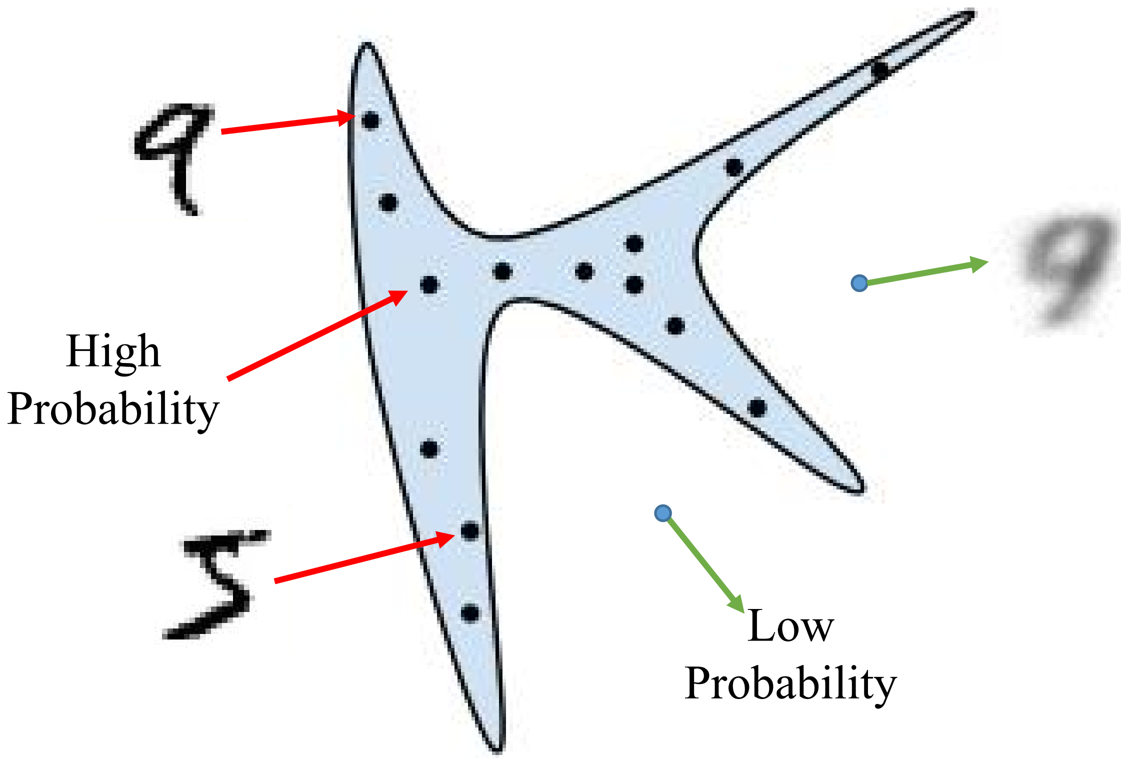
通过生成器快速生成样本，通过判别器提升生成器的全局观

GAN和VAE的生成质量对比



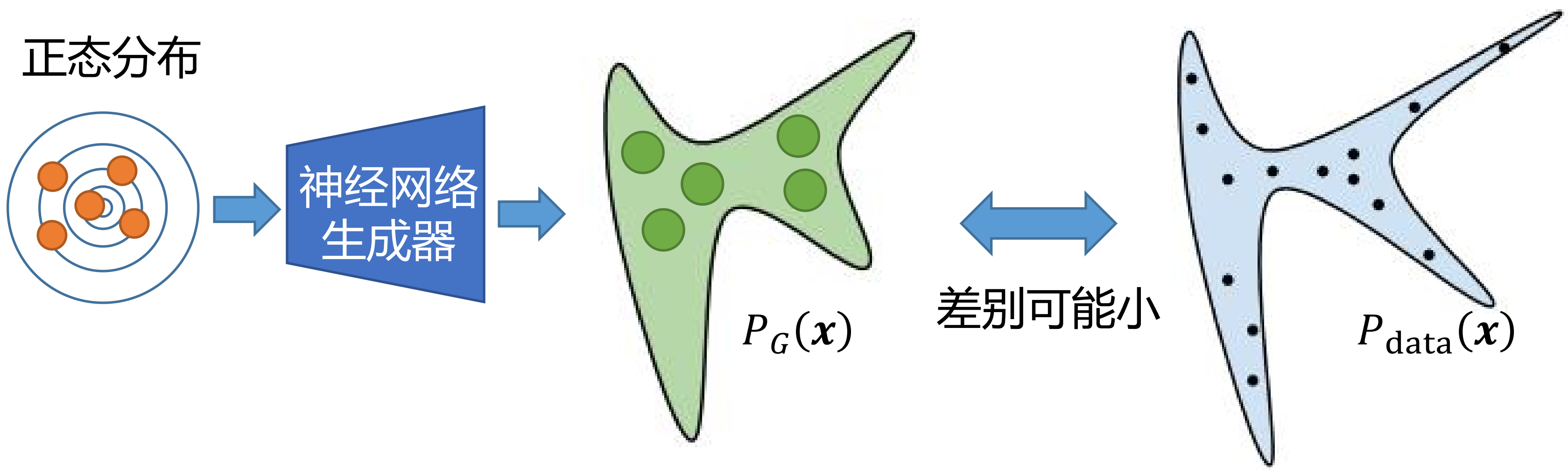
GAN背后的数学

- GAN的目标是找到真实的数据分布



GAN背后的数学

- 生成器G实际上隐式地定义了一个分布$P_G(\boldsymbol{x})$，因为从生成器中生成样本相当于从分布$P_G(\boldsymbol{x})$采样



$$G^* = \arg\min_G \underbrace{\mathrm{Div}(P_G, P_{data})}$$

P_G和P_{data}间的距离 ，但是应该如何计算呢?

GAN背后的数学

- 虽然不知道$P_G(\boldsymbol{x})$和$P_{\text{data}}(\boldsymbol{x})$的具体形式，但是都很容易从中采样
- 判别器的训练就是将给正样本（从$P_{\text{data}}(\boldsymbol{x})$采样的样本）高分，给负样本（从$P_G(\boldsymbol{x})$中采样的）低分，其目标函数如下

$$V(G,D)=E_{\boldsymbol{x}\sim P_{data}}[\log D(\boldsymbol{x})]+E_{\boldsymbol{x}\sim P_G}\left[\log\left(1-D(\boldsymbol{x})\right)\right]$$

- 在给定G的情况下，最优的判别器 $D^*=\arg\max_D V(G,D)$

其中$\max_D V(G,D)$与P_G和P_{data}间的散度（JS散度）是相关的

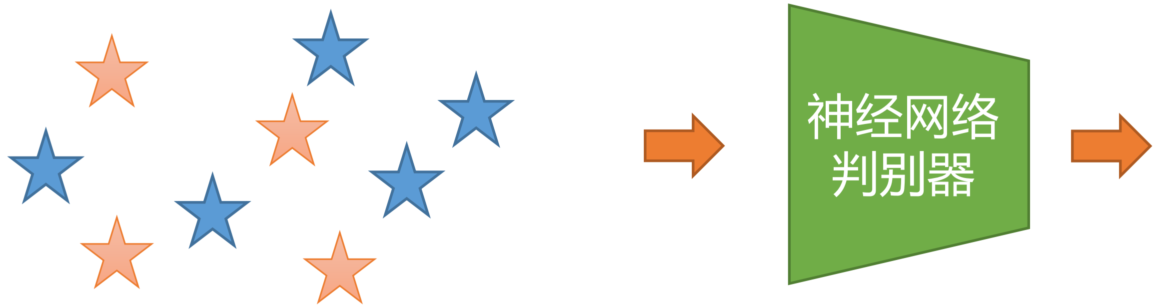
GAN背后的数学

从$P_{\text{data}}(\boldsymbol{x})$采样

从$P_G(\boldsymbol{x})$采样

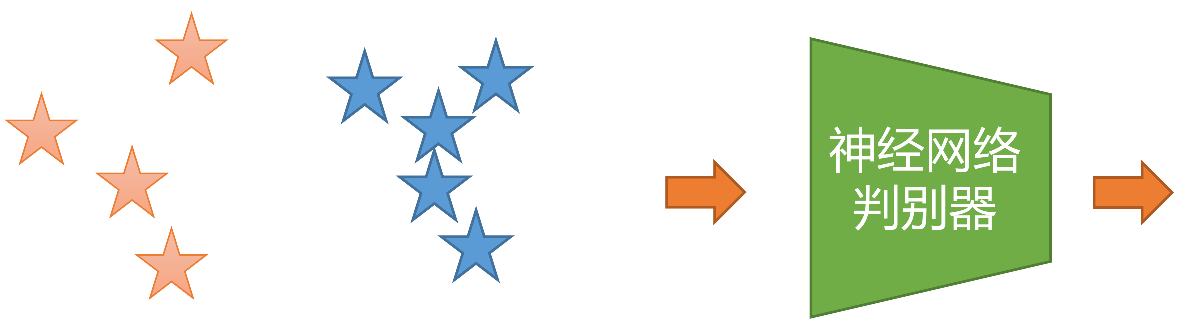
- 直观上理解

$$D^* = \arg\max_D V(G, D)$$



样本更难区分，所以 $\max_D V(G,D)$更小

$P_G(\boldsymbol{x})$和$P_{\text{data}}(\boldsymbol{x})$差异小



样本很容易区分 $\max_D V(G,D)$更大

$P_G(\boldsymbol{x})$和$P_{\text{data}}(\boldsymbol{x})$差异大

GAN背后的数学

$$V(G,D)=E_{\boldsymbol{x}\sim P_{data}}[\log D(\boldsymbol{x})]+E_{\boldsymbol{x}\sim P_G}\left[\log\left(1-D(\boldsymbol{x})\right)\right]$$

$$=\int P_{data}(\boldsymbol{x})\log D(\boldsymbol{x})\,d\boldsymbol{x}+\int P_G(\boldsymbol{x})\log\left(1-D(\boldsymbol{x})\right)d\boldsymbol{x}$$

$$=\int\left[P_{data}(\boldsymbol{x})\log D(\boldsymbol{x})+P_G(\boldsymbol{x})\log\left(1-D(\boldsymbol{x})\right)\right]d\boldsymbol{x}$$

$$\frac{\partial V(G,D)}{\partial\,D(\boldsymbol{x})}=P_{data}(\boldsymbol{x})\frac{1}{D(\boldsymbol{x})}+P_G(\boldsymbol{x})\frac{1}{1-D(\boldsymbol{x})}(-1)\quad=0$$

$$\Rightarrow\ \frac{1-D(\boldsymbol{x})}{D(\boldsymbol{x})}=\frac{P_G(\boldsymbol{x})}{P_{data}(\boldsymbol{x})}\quad\Rightarrow\quad D^*(\boldsymbol{x})=\frac{P_{data}(\boldsymbol{x})}{P_{data}(\boldsymbol{x})+P_G(\boldsymbol{x})}$$

GAN背后的数学

$$M = \frac{1}{2}(P+Q)$$

$$\mathrm{JSD}(P \parallel Q) = \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M)$$

$$V(G,D^*) = E_{\boldsymbol{x}\sim P_{data}}[\log D^*(\boldsymbol{x})] + E_{\boldsymbol{x}\sim P_G}\left[\log\left(1 - D^*(\boldsymbol{x})\right)\right]$$

$$= \int P_{data}(\boldsymbol{x}) \log \frac{P_{data}(\boldsymbol{x})}{P_{data}(\boldsymbol{x}) + P_G(\boldsymbol{x})} d\boldsymbol{x} + \int P_G(\boldsymbol{x}) \log\left(\frac{P_G(\boldsymbol{x})}{P_{data}(\boldsymbol{x}) + P_G(\boldsymbol{x})}\right) d\boldsymbol{x}$$

$$= \int P_{data}(\boldsymbol{x}) \log \frac{P_{data}(\boldsymbol{x})}{1/2\left(P_{data}(\boldsymbol{x}) + P_G(\boldsymbol{x})\right)} d\boldsymbol{x}$$

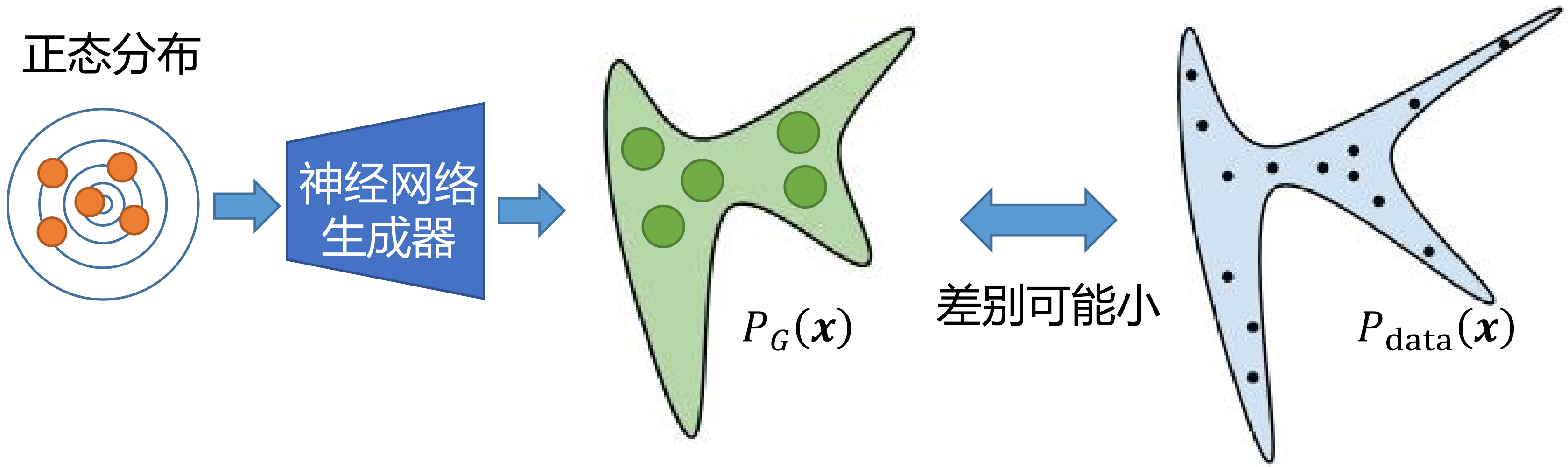
$$+ \int P_G(\boldsymbol{x}) \log\left(\frac{P_G(\boldsymbol{x})}{1/2\left(P_{data}(\boldsymbol{x}) + P_G(\boldsymbol{x})\right)}\right) d\boldsymbol{x} - 2\log 2$$

$$= KL\left(P_{data}(\boldsymbol{x}) || \frac{P_{data}(\boldsymbol{x}) + P_G(\boldsymbol{x})}{2}\right) + KL\left(P_G(\boldsymbol{x}) || \frac{\left(P_{data}(\boldsymbol{x}) + P_G(\boldsymbol{x})\right)}{2}\right) - 2\log 2$$

$$= 2\mathrm{JS}\left(P_{data}(\boldsymbol{x}) || P_G(x)\right) - 2\log 2$$

GAN背后的数学

- 生成器G实际上隐式地定义了一个分布$P_G(\boldsymbol{x})$，因为从生成器中生成样本相当于从分布$P_G(\boldsymbol{x})$采样



$$G^* = \arg\min_G \underbrace{\mathrm{Div}(P_G, P_{data})}_{P_G\text{和}P_{data}\text{间的距离，但是应该如何计算呢?}} \Rightarrow G^* = \arg\min_G \boxed{\max_D V(G, D)}$$

P_G和P_{data}间的距离 ，但是应该如何计算呢?

GAN背后的数学

$$V(G, D^*) = E_{x \sim P_{data}}[\log D^*(\boldsymbol{x})] + E_{x \sim P_G}\left[\log\left(1 - D^*(\boldsymbol{x})\right)\right]$$

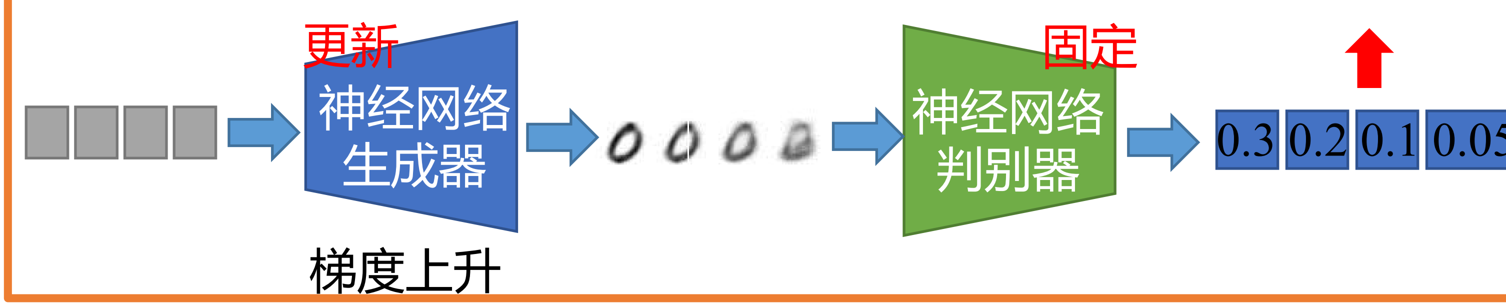
GAN的训练算法

- 给定真实样本集，随机生成负样本集
- While not convergent
  - 学习判别器D，使得给正样本高分，负样本低分

$$D^* = \arg\max_{D} V(G, D)$$

  - 用判别器D 生成负样本集

$$G^* = \arg\min_{G}\max_{D} V(G, D)$$



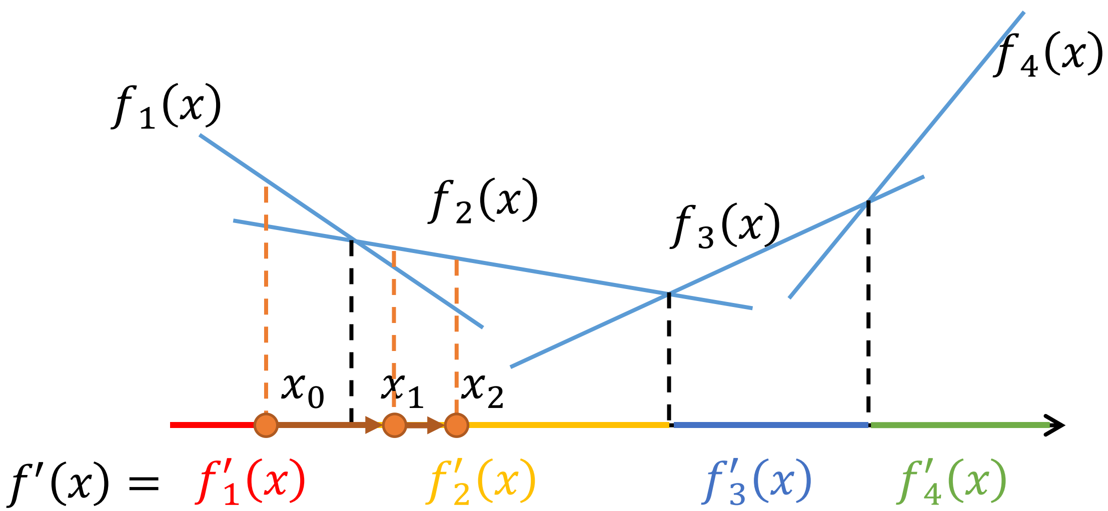
GAN背后的数学

$$G^* = \arg\min_G \max_D V(G, D)$$

- 为了找到最优的G^*，可以通过梯度下降法进行优化

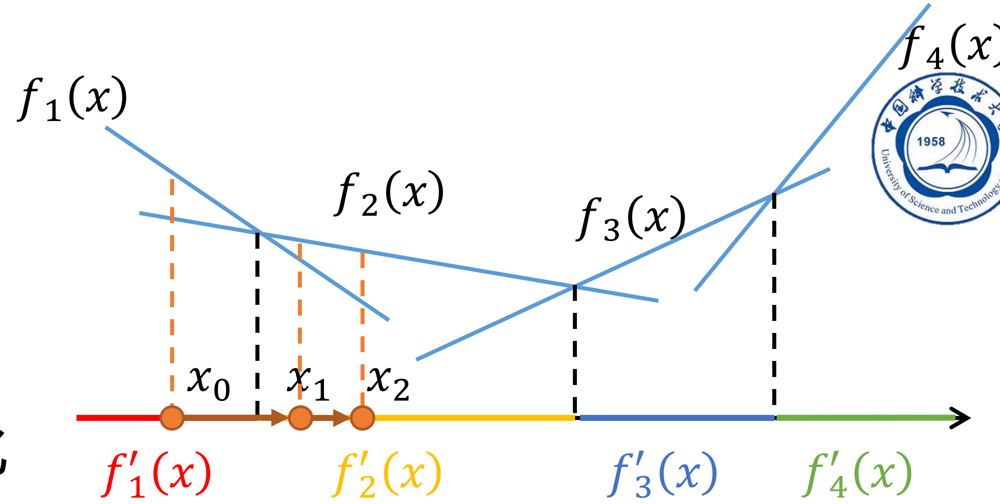
$$\theta_G \leftarrow \theta_G - \eta \frac{\partial \left(\max_D V(G, D) \right)}{\partial \theta_G}$$

$f(x) = \max\{f_1(x), f_2(x), f_3(x), f_4(x)\}$ $\quad$ $f'(x) = ?$



GAN背后的数学

GAN的训练算法



- 初始化 G_0
- 找到 D_0^*使得$V(G_0, D)$最大化

$V(G_0, D_0^*)$ 是分布$P_{G_0}(\boldsymbol{x})$和$P_{data}(\boldsymbol{x})$的JS散度

- $\theta_{G_1} \leftarrow \theta_{G_0} - \eta \frac{\partial V(G_0, D_0^*)}{\partial \theta_G}$ → G_1 减小了$P_{G_0}(\boldsymbol{x})$和$P_{data}(\boldsymbol{x})$的JS散度
- 找到 D_1^*使得$V(G_1, D)$最大化

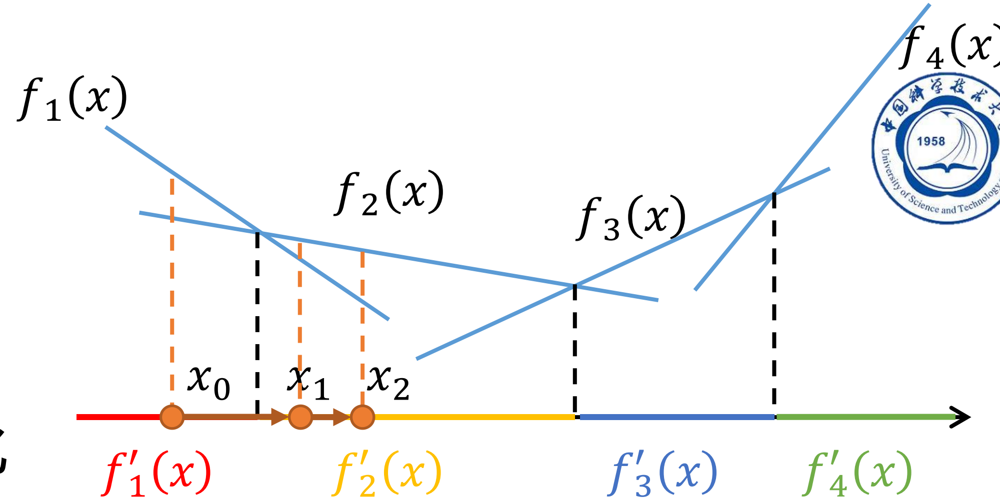
$V(G_1, D_1^*)$ 是分布$P_{G_1}(x)$和$P_{data}(x)$的JS散度

- $\theta_{G_2} \leftarrow \theta_{G_1} - \eta \frac{\partial V(G_1, D_1^*)}{\partial \theta_G}$ → G_2 减小了$P_{G_1}(\boldsymbol{x})$和$P_{data}(\boldsymbol{x})$的JS散度
- …

GAN背后的数学

GAN的训练算法

- 初始化 G_0
- 找到 D_0^*使得$V(G_0, D)$最大化



$V(G_0, D_0^*)$ 是分布$P_{G_0}(\boldsymbol{x})$和$P_{data}(\boldsymbol{x})$的JS散度

- $\theta_{G_1} \leftarrow \theta_{G_0} - \eta \frac{\partial V(G_0, D_0^*)}{\partial \theta_G}$ → G_1
- $\theta_{G_2} \leftarrow \theta_{G_1} - \eta \frac{\partial V(G_1, D_0^*)}{\partial \theta_G}$ → G_2
- $\theta_{G_3} \leftarrow \theta_{G_2} - \eta \frac{\partial V(G_2, D_0^*)}{\partial \theta_G}$ → G_3
- …

减小了$P_{G_0}(\boldsymbol{x})$和$P_{data}(\boldsymbol{x})$的JS散度

每次D^*的求解需要不断迭代直到收敛，而G只更新一次，G的更新效率太低

?

只能学习率较小且更新次数不多

否则$V(G_i, D_0^*)$无法准确衡量$P_{G_i}(\boldsymbol{x})$和$P_{data}(\boldsymbol{x})$的JS散度

生成对抗网络的算法实现技巧

出于效率考虑，对D^*的求解只迭代了K步，因而只得到$\max_D V(G,D)$的下界

- 随机初始化判别器参数 $\boldsymbol{\theta}_d$和生成器参数 $\boldsymbol{\theta}_g$
- While not convergent:

更新D

重复K次

- 从数据库中采样 m 样本$\{\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^m\}$
- 从某个分布中采样m个噪声样本 $\{\boldsymbol{z}^1, \boldsymbol{z}^2, ..., \boldsymbol{z}^m\}$
- 用生成器生成样本 $\{\tilde{\boldsymbol{x}}^1, \tilde{\boldsymbol{x}}^2, ..., \tilde{\boldsymbol{x}}^m\}$, $\tilde{\boldsymbol{x}}^i = G(\boldsymbol{z}^i)$
- 更新判别器参数 $\boldsymbol{\theta}_d$，通过最大化如下目标

$$\tilde{\mathcal{L}} = \frac{1}{m}\sum_{i=1}^{m} \log D(\boldsymbol{x}^i) + \frac{1}{m}\sum_{i=1}^{m} \log\left(1 - D(\tilde{\boldsymbol{x}}^i)\right)$$

$$\boldsymbol{\theta}_d \leftarrow \boldsymbol{\theta}_d + \eta\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}_d)$$

更新G

只做一次

- 从某个分布中采样m个噪声样本 $\{\boldsymbol{z}^1, \boldsymbol{z}^2, ..., \boldsymbol{z}^m\}$
- 更新生成器参数 $\boldsymbol{\theta}_g$，通过最小化如下目标

$$\tilde{\mathcal{L}} = \cancel{\frac{1}{m}\sum_{i=1}^{m} \log D(\boldsymbol{x}^i)} + \frac{1}{m}\sum_{i=1}^{m} \log\left(1 - D\left(G(\boldsymbol{z}^i)\right)\right)$$

$$\boldsymbol{\theta}_g \leftarrow \boldsymbol{\theta}_g - \eta\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}_g)$$

生成对抗网络的算法实现技巧

在更新G的时候，最小化如下目标

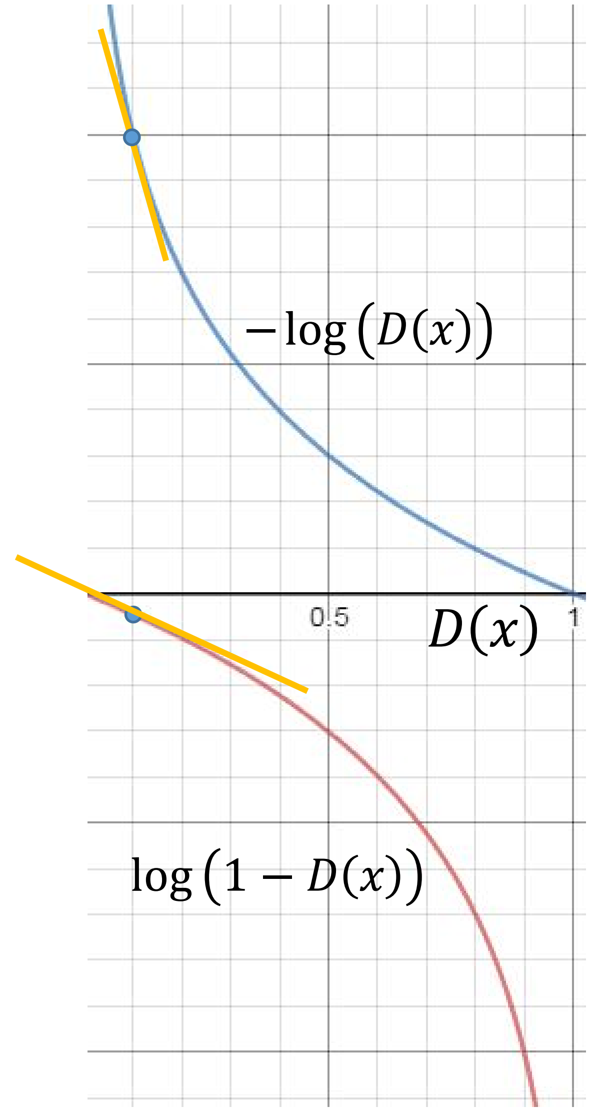
$$\tilde{\mathcal{L}} = \cancel{\frac{1}{m}\sum_{i=1}^{m}\log D(\boldsymbol{x}^i)} + \frac{1}{m}\sum_{i=1}^{m}\log\left(1 - D\left(G(\boldsymbol{z}^i)\right)\right)$$

但是在初始时候会很慢，因为G比较差，生成的样本很容被D分辨为假的，也就是说$D\left(G(\boldsymbol{z}^i)\right)$是趋于0的，$\log\left(1 - D\left(G(\boldsymbol{z}^i)\right)\right)$的梯度很小

在更新G的时候，最小化如下目标

$$\tilde{\mathcal{L}} = -\frac{1}{m}\sum_{i=1}^{m}\log\left(D\left(G(\boldsymbol{z}^i)\right)\right)$$

Log D技巧



生成对抗网络的算法实现技巧

出于效率考虑，对D^* 的求解只迭代了K步，因而只得到$\max_D V(G,D)$的下界

- 随机初始化判别器参数 $\boldsymbol{\theta}_d$和生成器参数 $\boldsymbol{\theta}_g$
- While not convergent:

更新D

重复K次

- 从数据库中采样 m 样本$\{\boldsymbol{x}^1,\boldsymbol{x}^2,\dots,\boldsymbol{x}^m\}$
- 从某个分布中采样m个噪声样本 $\{\boldsymbol{z}^1,\boldsymbol{z}^2,\dots,\boldsymbol{z}^m\}$
- 用生成器生成样本 $\{\tilde{\boldsymbol{x}}^1,\tilde{\boldsymbol{x}}^2,\dots,\tilde{\boldsymbol{x}}^m\}$, $\tilde{\boldsymbol{x}}^i=G(\boldsymbol{z}^i)$
- 更新判别器参数 $\boldsymbol{\theta}_d$，通过最大化如下目标

$$\tilde{\mathcal{L}}=\frac{1}{m}\sum_{i=1}^m\log D(\boldsymbol{x}^i)+\frac{1}{m}\sum_{i=1}^m\log\left(1-D(\tilde{\boldsymbol{x}}^i)\right)$$

$$\boldsymbol{\theta}_d\leftarrow\boldsymbol{\theta}_d+\eta\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}_d)$$

更新G

只做一次

- 从某个分布中采样m个噪声样本 $\{\boldsymbol{z}^1,\boldsymbol{z}^2,\dots,\boldsymbol{z}^m\}$
- 更新生成器参数 $\boldsymbol{\theta}_g$，通过最小化如下目标

$$\tilde{\mathcal{L}}=-\frac{1}{m}\sum_{i=1}^m\log\left(D\left(G(\boldsymbol{z}^i)\right)\right)$$

$$\boldsymbol{\theta}_g\leftarrow\boldsymbol{\theta}_g-\eta\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}_g)$$

GAN背后的数学

- 不用Log D技巧，生成网络的目标函数为

$$V(G, D^*) = 2\mathrm{JS}\big(P_{data}(\boldsymbol{x}) || P_G(x)\big) - 2\log 2$$

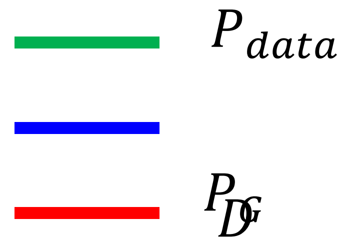
- 但用Log D技巧时，生成网络的目标函数为

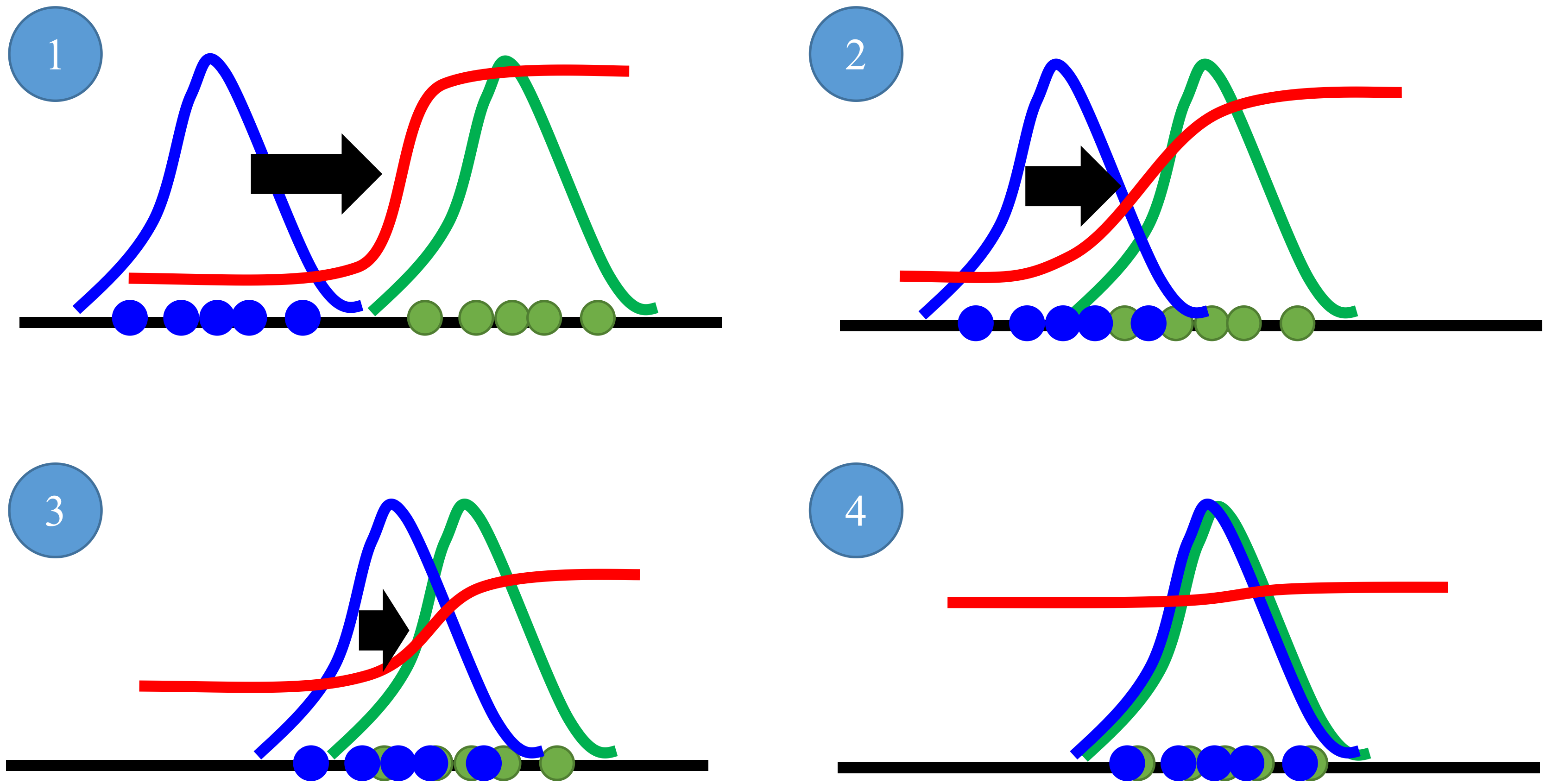
$$V(G, D^*) \propto \mathrm{KL}\big(P_G(\boldsymbol{x}) || P_{data}(\boldsymbol{x})\big) - 2\ \mathrm{JS}\big(P_{data}(\boldsymbol{x}) || P_G(x)\big)$$

同时最小化生成分布与真实分布的KL散度，并最大化两者的JS散度，

这和直觉相冲突，在数值上则会导致梯度不稳定

GAN训练的直观理解





谢谢