第15章 降维算法

《人工智能算法》

清华大学出版社
2022年7月

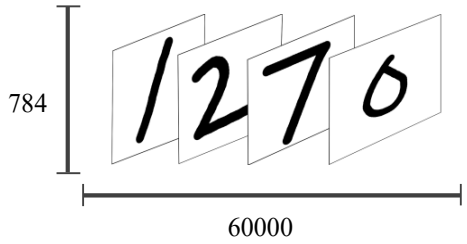
提纲

- 引例
- 降维算法概述
- 自编码器
- 自编码器的改进
- 变分自编码器
- 生成对抗网络
- 总结

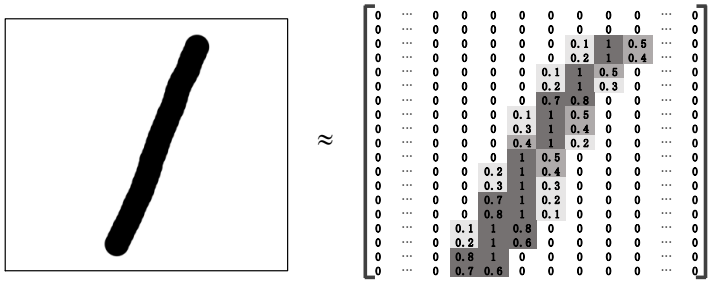
引例 (1)

◆ MNIST数据集：60000个训练样本和10000个测试样本

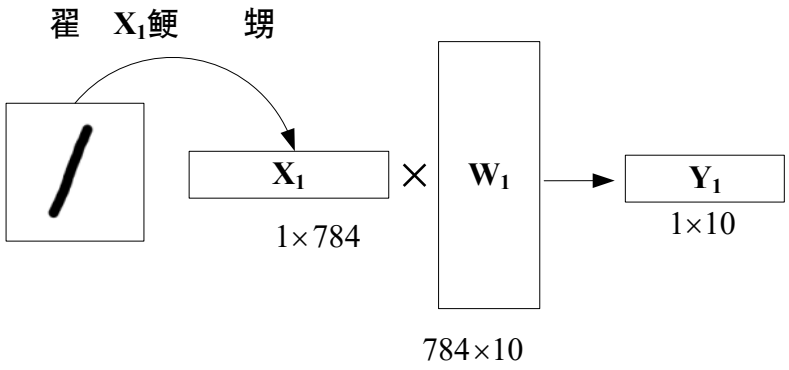
✓ 28×28像素手写数字图片的MNIST数据集



◆ 手写数字“1”的图片及相应的像素矩阵



✓ 手写数字“1”的图片识别模型训练过程



乘法运算次数：

$1\times784\times10 = 7840$

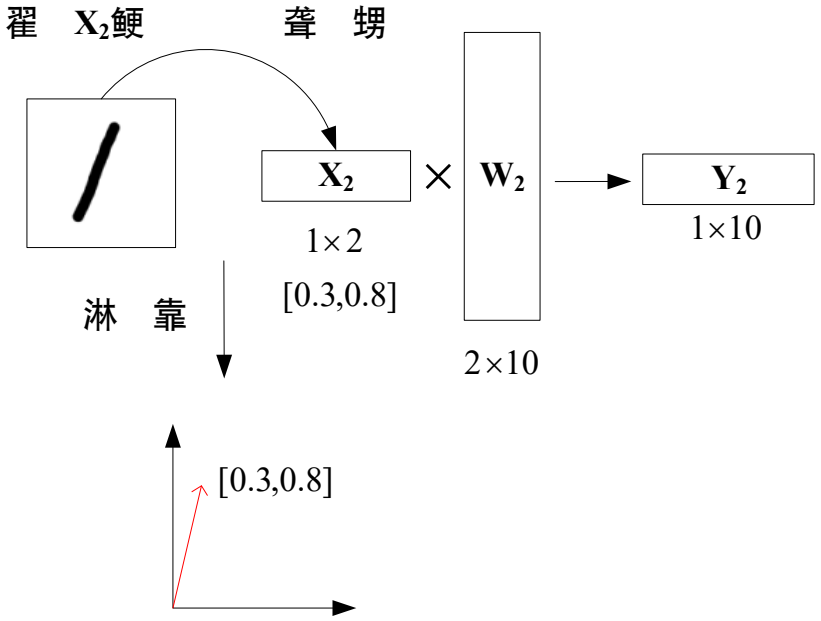
- 完成数据分类☺
- 数据维度高，计算复杂度高☹
- 可视化程度不高☹

引例 (2)

◆ 对MNIST数据集降维

降低数据维度，保证其有效信息不丢失

手写数字“1”的图片识别模型训练过程（假设降维后维度为2）



乘法运算次数：

$1\times2\times10 = 20 << 7840$

- 完成数据分类☺
- 压缩数据☺
- 降低计算复杂度☺
- 提高可视化程度☺

提纲

- 引例
- 降维算法概述
- 自编码器
- 自编码器的改进
- 变分自编码器
- 生成对抗网络
- 总结

降维算法概述

- **什么是降维**

  降低数据维度，保证其有效信息不丢失

- **为什么要降维**

  ✓ 缓解高维数据维数灾难问题

  ✓ 提高数据可视化程度

  ✓ 数据压缩减少存储空间

- **传统的降维方法**

  ✓ 主成分分析，奇异值分解，线性判别分析

  ✓ 不能较好地保持数据集的非线性特性

- **基于深度学习的降维方法**

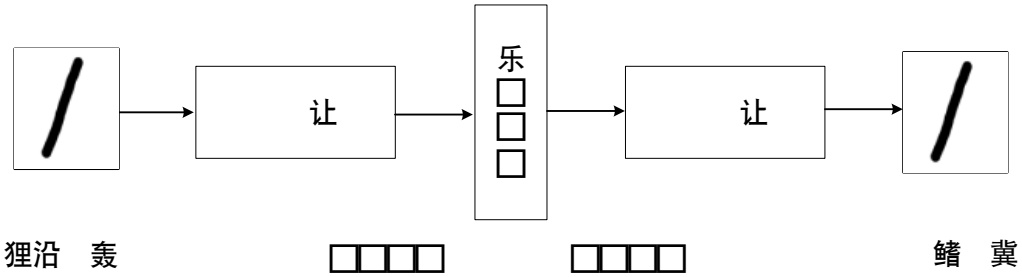
  自编码器，变分自编码器（生成模型），对抗神经网络 （生成模型）

提纲

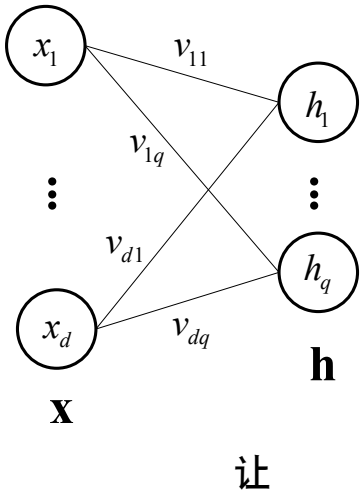
- 引例
- 降维算法概述
- 自编码器
- 自编码器的改进
- 变分自编码器
- 生成对抗网络
- 总结

自编码器 (1)

- **基本思想**



（1）编码阶段



- 编码器将原始输入映射为低维数据，实现对输入数据的降维

$$\mathbf{h} = f(\mathbf{x}) = s_f(\mathbf{V}\mathbf{x} + \boldsymbol{\Upsilon})$$

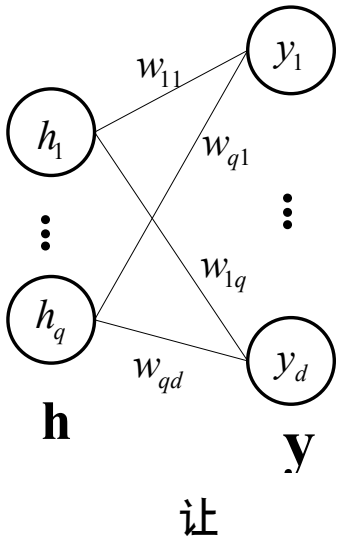
其中：$\mathbf{V}$：权重矩阵

$\boldsymbol{\Upsilon}$：偏置矩阵

s_f：编码器的激活函数

自编码器 (2)

（2）解码阶段



让

- 解码器则将低维数据映射成高维数据，实现对输入数据的重构。

$$\mathbf{y} = g(\mathbf{h}) = s_g(\mathbf{W}\mathbf{h} + \mathbf{\Theta})$$

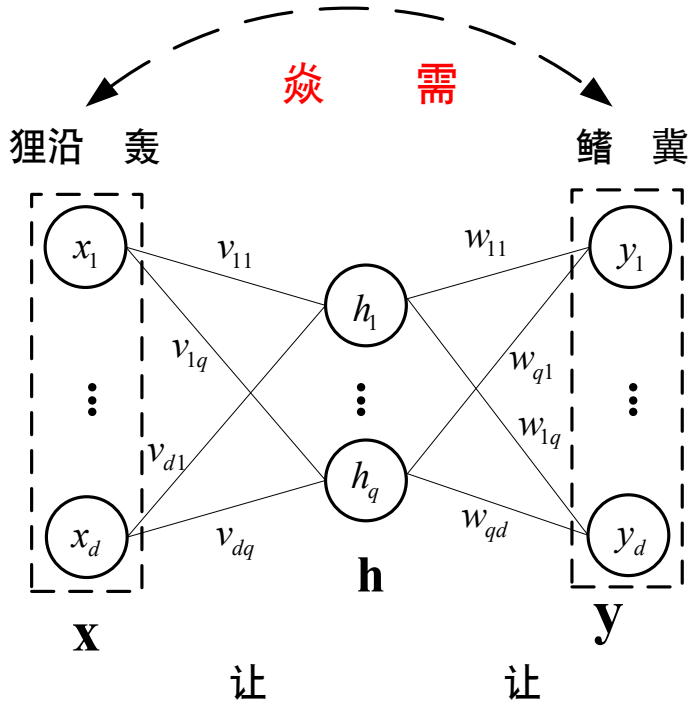
其中：$\mathbf{W}$：权重矩阵

$\mathbf{\Theta}$：偏置矩阵

s_g：解码器的激活函数

自编码器 (3)

（3）损失函数



$$J_{AE}(\mathbf{V},\mathbf{W},\boldsymbol{\Upsilon},\boldsymbol{\Theta}) = L(\mathbf{x},\mathbf{y}) = L(\mathbf{x},g(f(\mathbf{x})))$$

其中：

$L(\mathbf{x},\mathbf{y})$可为均方误差，也可为交叉熵

自编码器 (4)

训练算法：

随机初始化网络中的所有权重矩阵**V**和**W**，偏置向量**ϒ**和**Θ**

$t \leftarrow 1$; $L \leftarrow 0$

While $t \le N$ Do

  For each **x** In D Do

    通过编码器解码器计算得到**x**对应的输出**y**

    $L \leftarrow L + \frac{1}{2}\sum_{j=1}^{m}(\mathbf{y}_j - \mathbf{x}_j)^2$

  End For

  梯度下降更新权重和偏置

  $t \leftarrow t+1$

End While

Return **V**, **W**, **ϒ**, **Θ**

时间复杂度分析：

- 输入规模：
  - D：训练数据集
  - $\eta(0 < \eta < 1)$：学习率
  - N：总迭代次数
  - m：输入数据个数
  - d：输入数据维度
  - q：降维之后数据维度
- 时间复杂度：

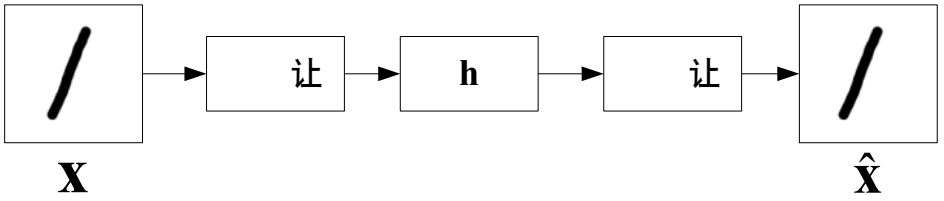
  $O(N \times m \times d \times q)$

提纲

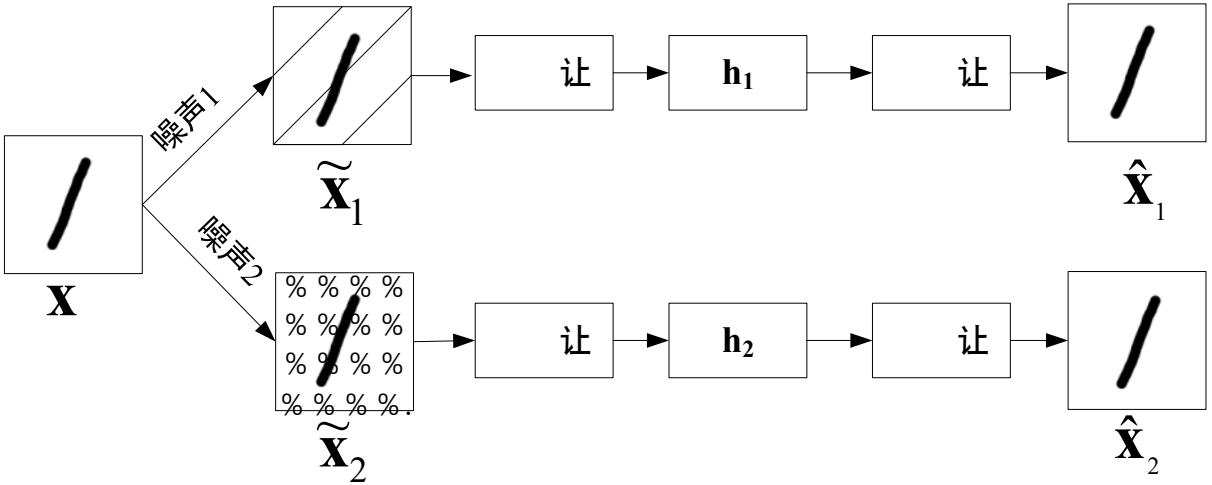
- 引例
- 降维算法概述
- 自编码器
- 自编码器的改进
- 变分自编码器
- 生成对抗网络
- 总结

降噪自编码器 (1)

自编码器



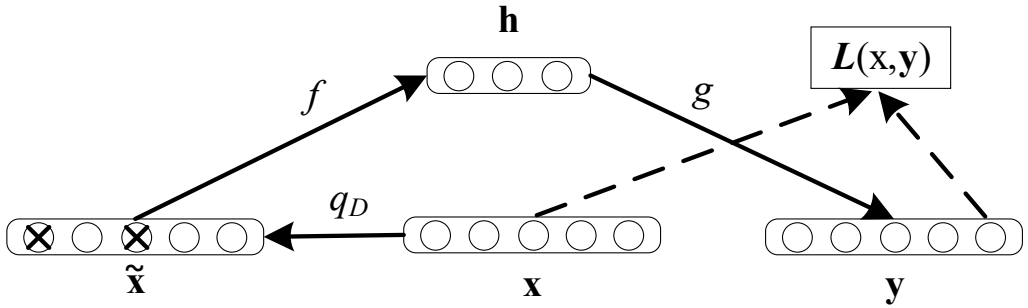
降噪自编码器



- 降噪自编码器具备抗噪声的能力，能得到更健壮的隐层表示
- 降噪自编码器中$\mathbf{h}_2 \neq \mathbf{h}_3$

降噪自编码器 (2)

问题：为了让模型不受噪声的影响



改进：

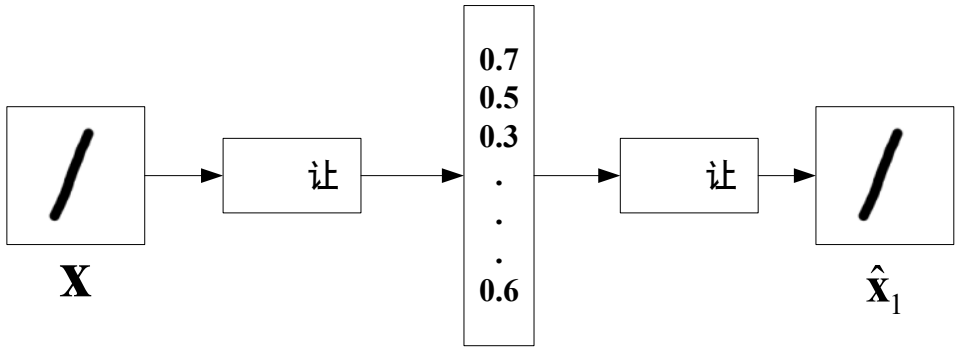
对原始输入数据加入噪声，产生与原始输入数据相对应的噪声数据

原始输入：x，利用q_D“加噪”生成最终的输入：$\tilde{x}$，

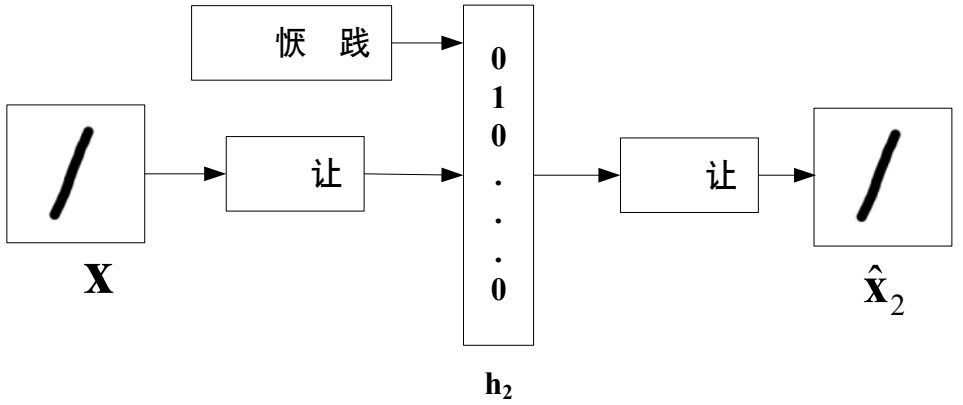
- **优点：**提高模型鲁棒性
- **缺点：**加入噪声，增加模型计算开销

稀疏自编码器（1）

自编码器



稀疏自编码器

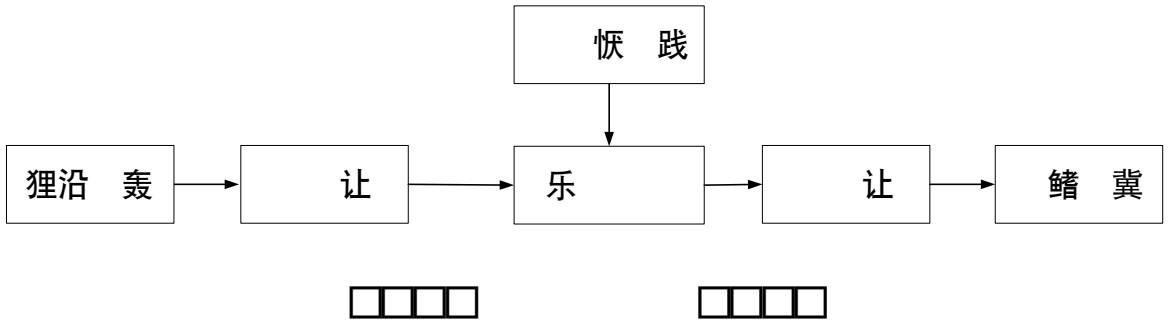


稀疏自编码器将隐藏层数据稀疏化表示，得到更健壮的隐层表示

稀疏自编码器 (2)

问题：从无类别标签的数据学习到数据的稀疏表示

在隐藏层加入稀疏性限制，减少隐藏层“活跃”的神经元个数，获得隐藏层的稀疏表示



改进:

- 在自编码器基础上增加稀疏性限制
- 损失函数增加稀疏惩罚项

稀疏自编码器（3）

损失函数

$$J_{SAE}(\mathbf{W},\mathbf{b}) = \sum\left(L\left(\mathbf{x}, g\left(f(\mathbf{x})\right)\right)\right) + \beta\sum_{j=1}^{q} KL\left(\rho||\hat{\rho}_j\right)$$

$\hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m}\left(h_j(x_i)\right)$：表示$m$个训练样本在隐藏层神经元$j$上的平均激活值

h_j：隐藏层神经元j上的激活值

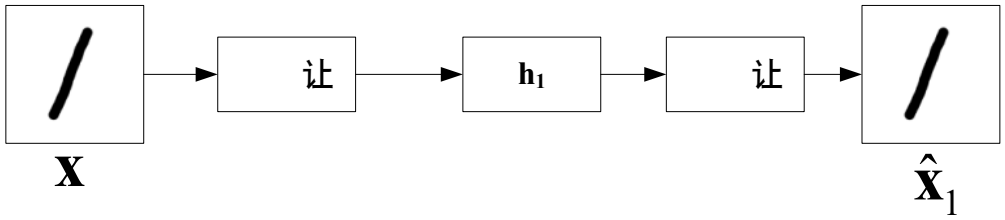
β：控制稀疏惩罚项的权重

ρ：一般接近于0，确保大部分隐层神经元被抑制

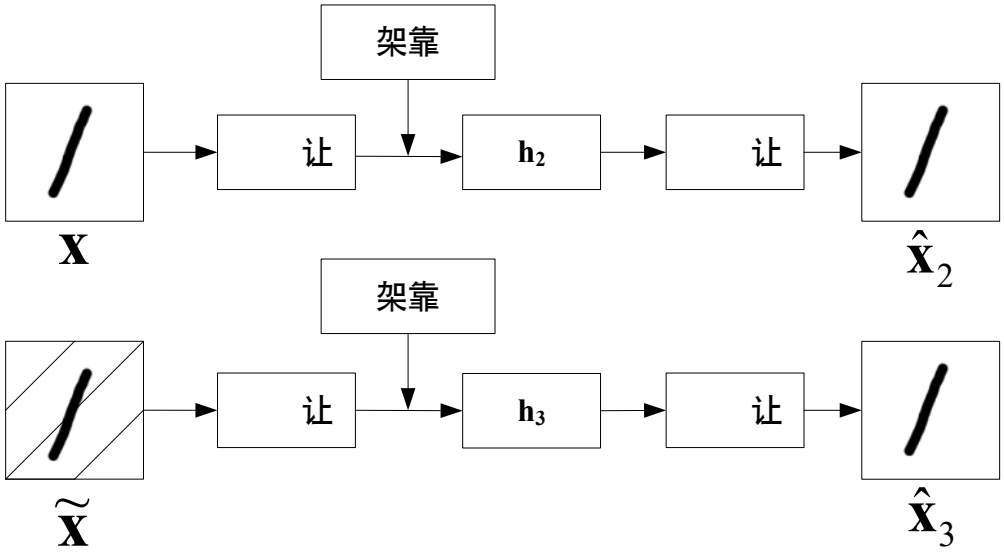
- **优点：**学习隐藏层的稀疏表示，对数据过拟合具有泛化能力，降低数据维度，提升性能
- **缺点：**无法指定抑制哪些隐藏层神经元，学习到的隐藏层的稀疏表示的物理意义不明确

收缩自编码器 (1)

自编码器



收缩自编码器



- 收缩自编码器在编码过程中增加正则化项，提高模型应对干扰的能力
- 收缩自编码器中$\mathbf{h}_2 \approx \mathbf{h}_3$

收缩自编码器（2）

问题：数据传播过程中受噪声干扰

噪声会影响图像的视觉效果、掩盖图像细节

改进：

- 抑制训练样本在所有方向上的扰动
- 在自编码器的损失函数上增加正则化项

$$J_{CAE}(\mathbf{W},\mathbf{b}) = \sum\left(L\left(\mathbf{x}, g\left(f(\mathbf{x})\right)\right)\right) + \lambda\left\|J_f(\mathbf{x})\right\|_F^2$$

$\left\|J_f(x)\right\|_F^2$表示矩阵$J_f(x)$的Frobenius范数，$\left\|J_f(x)\right\|_F^2 = \sum_{i=1}^{d}\sum_{j=1}^{q}\left(\frac{\partial h_j}{\partial x_i}\right)^2$

与去噪自编码器的区别：

- 收缩自编码器通过对损失函数添加正则化项，
- 去噪自编码器通过对输入样本添加噪声

优点：学习到的隐藏层表示对输入数据的微小变化不敏感，会提高模型对输入噪声数据的鲁棒性

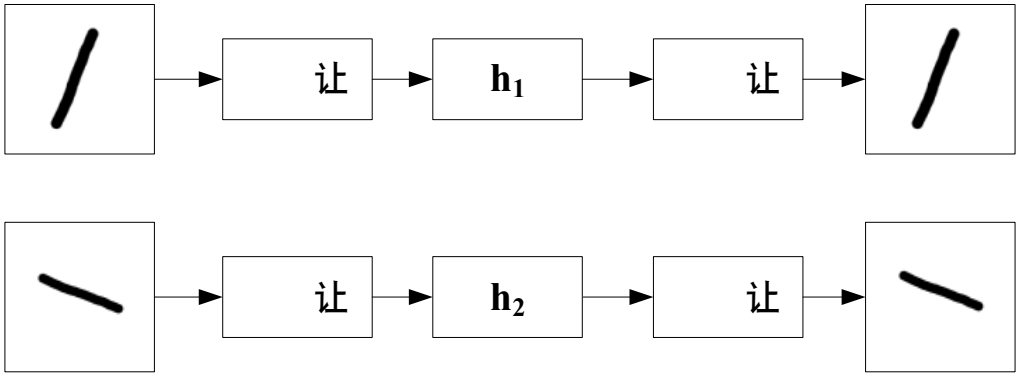
缺点：含有多层隐藏层时，计算时间复杂度较高

提纲

- 引例
- 降维算法概述
- 自编码器
- 自编码器的改进
- 变分自编码器
- 生成对抗网络
- 总结

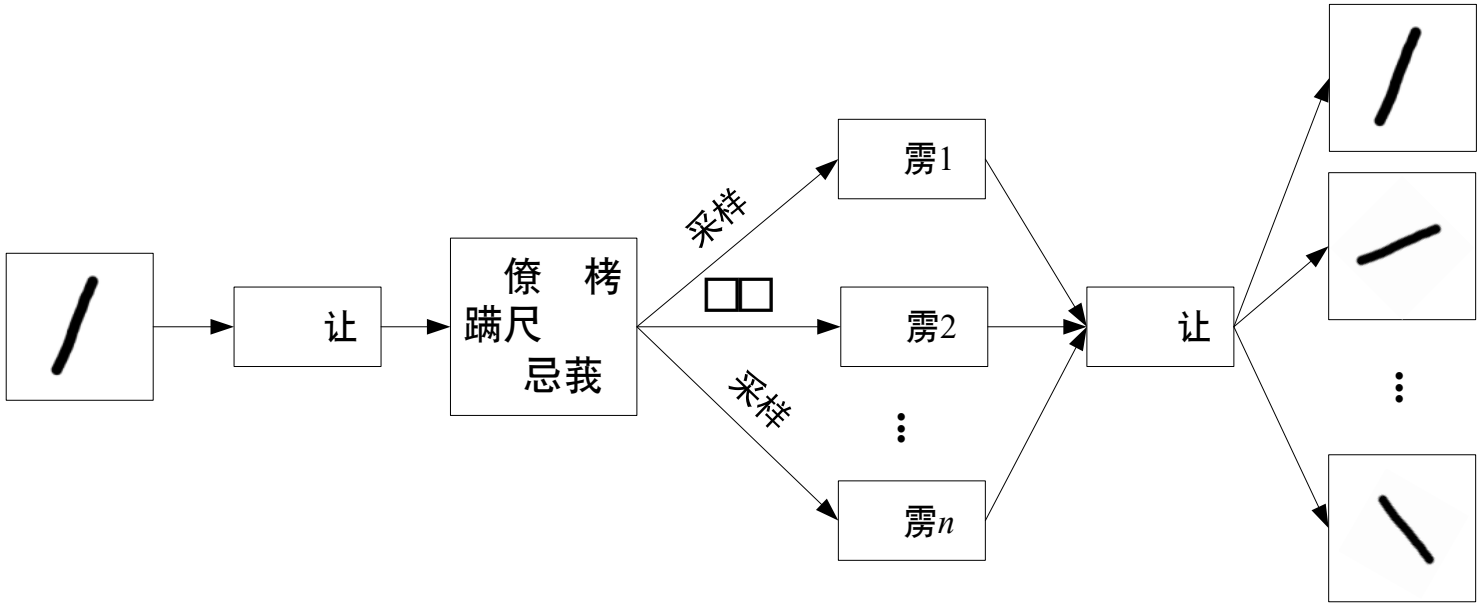
变分自编码器 (1)

◆ **概述**



- 自编码器编码解码结构简单
- 自编码器中学习到的隐变量（例如h_1、h_2）为一个特定的向量，不能学习到服从隐变量的数据分布
- 自编码器中不能生成和原始输入相似的数据

变分自编码器 (2)



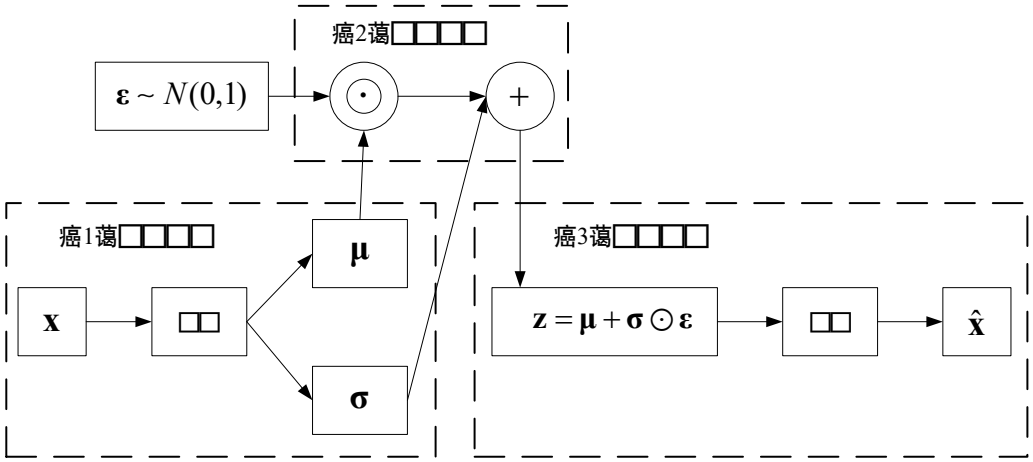
- 变分自编码器是一个生成模型
- 变分自编码器能学习隐变量所服从的概率分布，并通过概率分布采样生成和原始数据相似的数据，即支持新样本的生成

变分自编码器 (3)

◆ **模型训练**

（2）采样阶段

从高斯分布中生成隐变量的随机采样样本z



（1）编码阶段

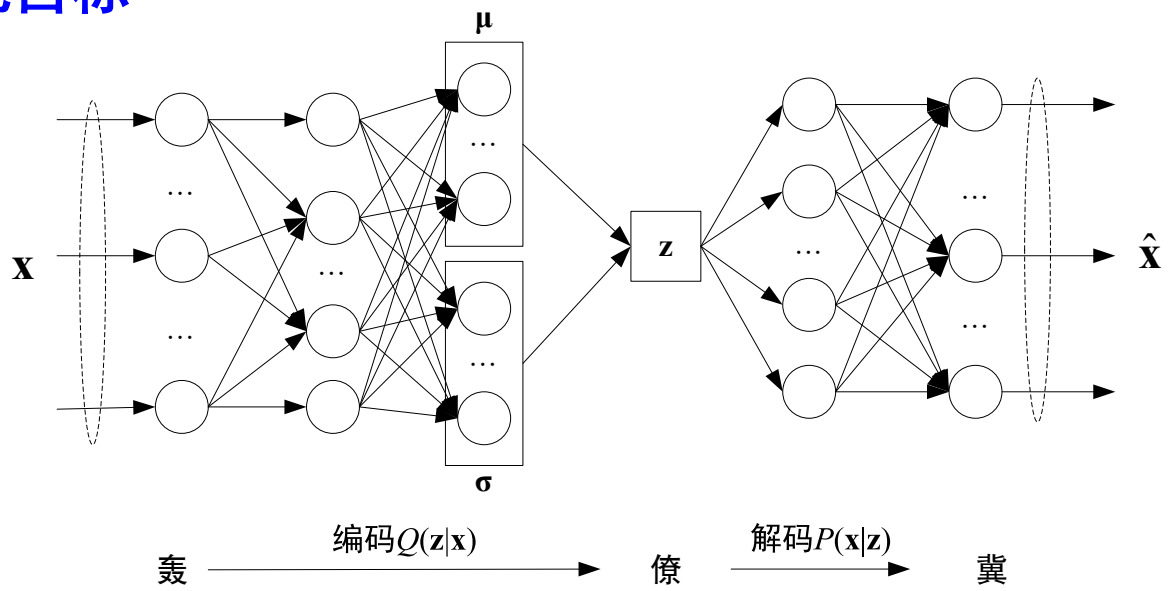
x经过编码器后映射为隐变量所服从的均值μ且标准差σ的多元高斯分布

（3）解码阶段

将采样样本z通过解码器映射为重构数据x̂

变分自编码器 (4)

◆ 优化目标



- 假设z服从多元高斯分布
- $Q(\mathbf{z}|\mathbf{x})$：编码过程学习到的概率分
- $P(\mathbf{x}|\mathbf{z})$：解码过程学习到的概率分布

变分自编码器 (5)

◆ 优化目标

$$J_{VAE} = D_{KL}\big(Q(\mathbf{z}|\mathbf{x}) \parallel P(\mathbf{z})\big) - E_{Q(\mathbf{z}|\mathbf{x})}\big[\log\big(P(\mathbf{x}|\mathbf{z})\big)\big]$$

其中：$D_{KL}\big(Q(\mathbf{z}|\mathbf{x}) \parallel P(\mathbf{z})\big)$：正则化项，使得编码器返回的分布接近正态分布

$E_{Q(\mathbf{z}|\mathbf{x})}\big[\log\big(P(\mathbf{x}|\mathbf{z})\big)\big]$：重构误差，使生成数据和原始数据尽可能相似

- 假设隐变量服从多元标准高斯分布，即$P(\mathbf{z})\sim N(0, I)$，$Q(\mathbf{z}|\mathbf{x})\sim N(\mu, \sigma^2)$

$$D_{KL}\big(Q(\mathbf{z}|\mathbf{x}) \parallel P(\boldsymbol{z})\big) = \frac{1}{2}\sum_{i=1}^{f}\big(\boldsymbol{\mu}_i^2 + \boldsymbol{\sigma}_i^2 - \log \boldsymbol{\sigma}_i^2 - 1\big)$$

- 假设 $P(\mathbf{x}|\mathbf{z})$ 服从高斯分布，解码器（$\hat{\mathbf{x}} = g(\mathbf{z})$）用于拟合高斯分布的均值，且标准差为常数$c$，则$P(\mathbf{x}|\mathbf{z})$可以表示为$N(g(\mathbf{z}), c^2)$

$$-E_{Q(\mathbf{z}|\mathbf{x})}\big[\log\big(P(\mathbf{x}|\mathbf{z})\big)\big] \simeq \|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

变分自编码器 (6)

训练算法

随机初始化网络中的权重矩阵$\mathbf{W}$和偏置$\mathbf{b}$；$t \leftarrow 1$

While $t \le N$ Do

$J_{VAE} \leftarrow 0$

For each $\mathbf{x}$ In D Do

$\mathbf{h} \leftarrow \sigma_h(\mathbf{W}_h\mathbf{x} + \mathbf{b}_h)$

$\boldsymbol{\mu} \leftarrow f_1(\mathbf{x}) = \sigma_\mu(\mathbf{W}_\mu\mathbf{h} + \mathbf{b}_\mu), \boldsymbol{\sigma} \leftarrow f_2(\mathbf{x}) = \sigma_\sigma(\mathbf{W}_\sigma\mathbf{h} + \mathbf{b}_\sigma)$

从$N(0, I)$采样$\boldsymbol{\epsilon}$

$\mathbf{z} \leftarrow \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$

$\mathbf{h}' \leftarrow \sigma_{h'}(\mathbf{W}_{h'}\mathbf{z} + \mathbf{b}_{h'})$

$\hat{\mathbf{x}} \leftarrow \sigma_g(\mathbf{W}_g\mathbf{h}' + \mathbf{b}_g)$

$J_{VAE} \leftarrow J_{VAE} + \sum_{i=1}^{f}(\boldsymbol{\mu}_i^2 + \boldsymbol{\sigma}_i^2 - \log \boldsymbol{\sigma}_i^2 - 1) + \|\mathbf{x} - \hat{\mathbf{x}}\|^2$

End For

$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J_{VAE}}{\partial \mathbf{W}}$； $\mathbf{b} \leftarrow \mathbf{b} - \eta \frac{\partial J_{VAE}}{\partial \mathbf{b}}$; $t \leftarrow t+1$

End While

Return $\mathbf{W}$, $\mathbf{b}$

时间复杂度分析

- 输入规模

  $|D|$：数据集规模

  N：总迭代次数

  d：输入数据维度

- 时间复杂度

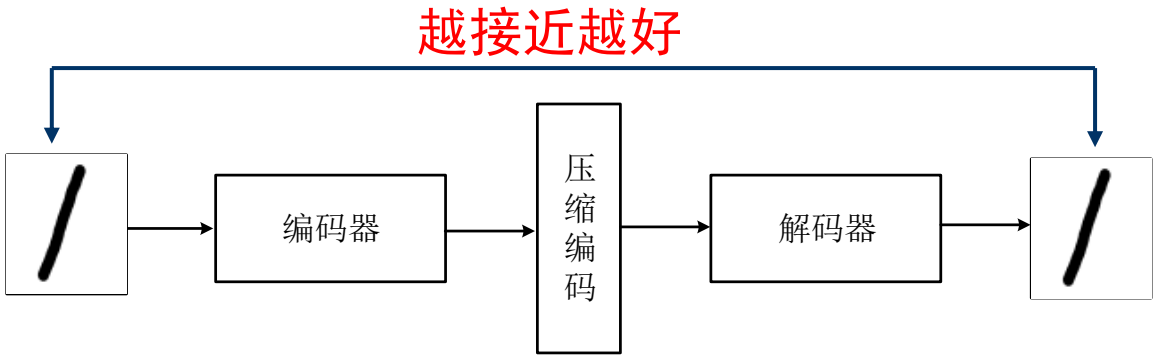
  $O(N \times |D| \times d)$

提纲

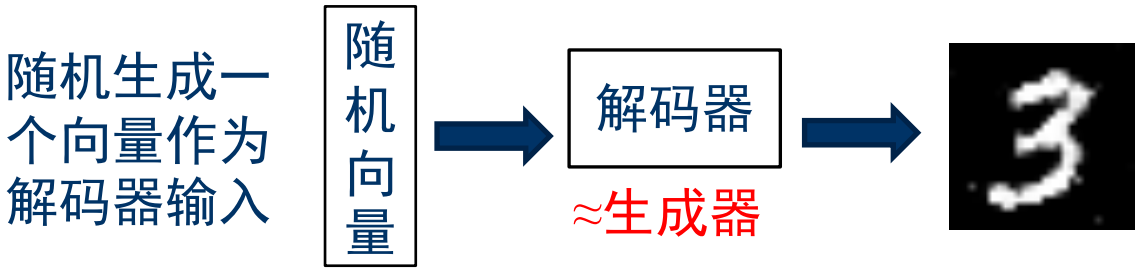
- 引例
- 降维算法概述
- 自编码器
- 自编码器的改进
- 变分自编码器
- 生成对抗网络
- 总结

生成对抗网络 (1)

- 引例



自编码器不能生成数据，且解码器的输出与原来的输入相比是有损的



生成对抗网络 (2)

- **传统的生成模型**

– 根据训练集估计样本分布$P(x)$，之后根据$P(x)$采样，生成和训练集“类似”的新样本

需要大量先验知识对潜在分布建模☹

– 对于低维数据，可以使用简单的，只有少量参数的概率模型（如高斯分布）拟合$P(x)$，但对于高维数据（例如图像）处理比较困难

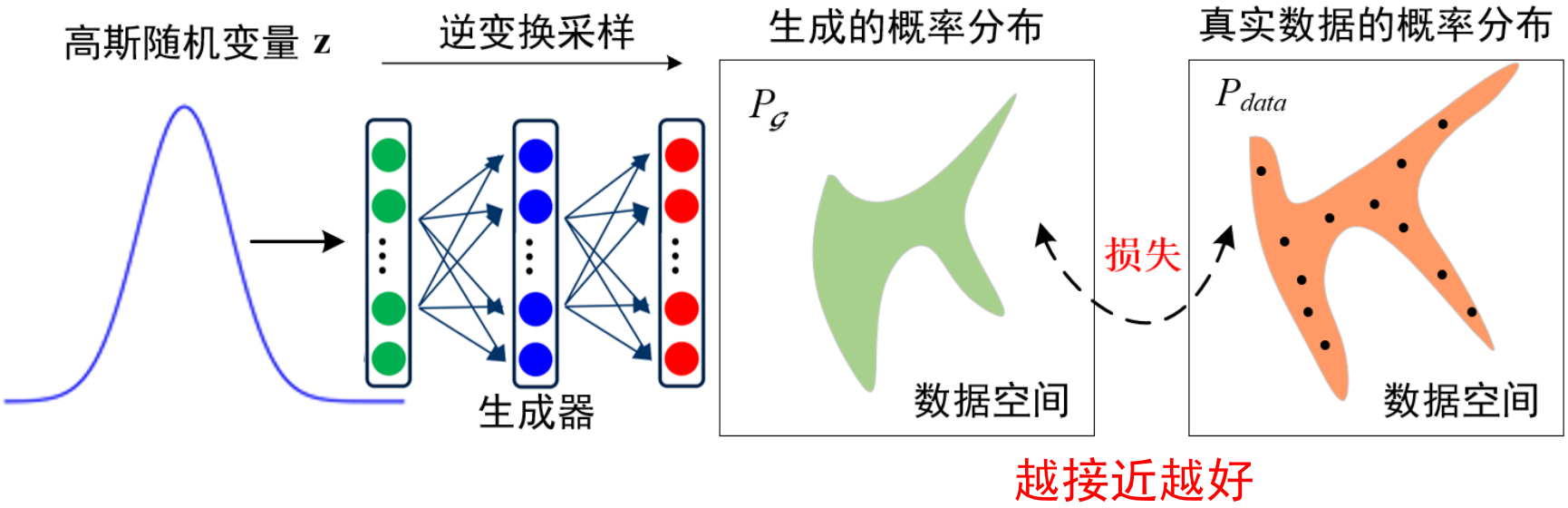
计算复杂☹

生成对抗网络 (3)

- **基于深度学习的生成模型**
  - ✓ **变分自编码器**
  - 将输入编码为低维空间中的概率分布（如高斯分布）
  - 低维空间中的两个相邻取值解码后呈现相似的内容
  - ✓ **生成对抗网络（Generative Adversarial Network, GAN）**
  - GAN没有变分下界（即变化的下限），是渐进一致的，理论上能完美恢复真实数据的分布
  - 相比变分自编码器，GAN训练时不需要对隐变量分布做推断，即无需确定样本的概率模型，不用显式地定义概率密度函数

生成对抗网络 (4)

- **基本思想**



为了让P_G与P_{data}接近，需要最小化它们之间的距离

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

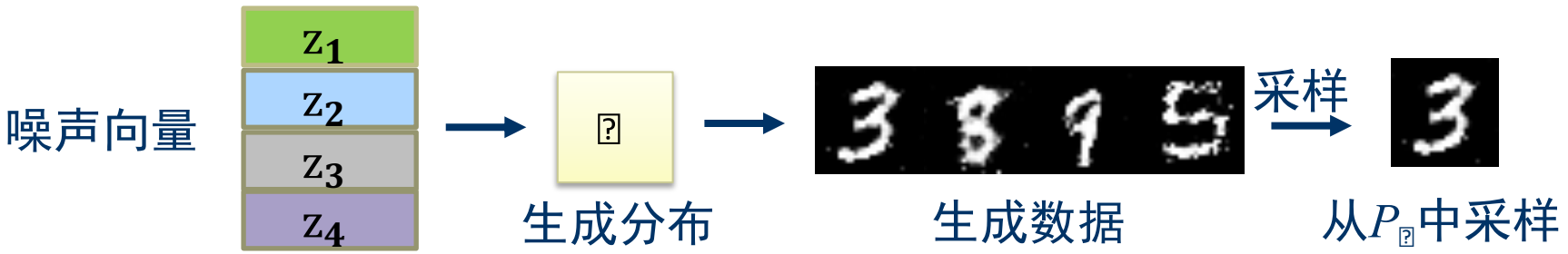
生成对抗网络 (5)

- **基本思想**

$$G^* = \min_{\theta_g} Div(P_{\theta_g}, P_{data})$$

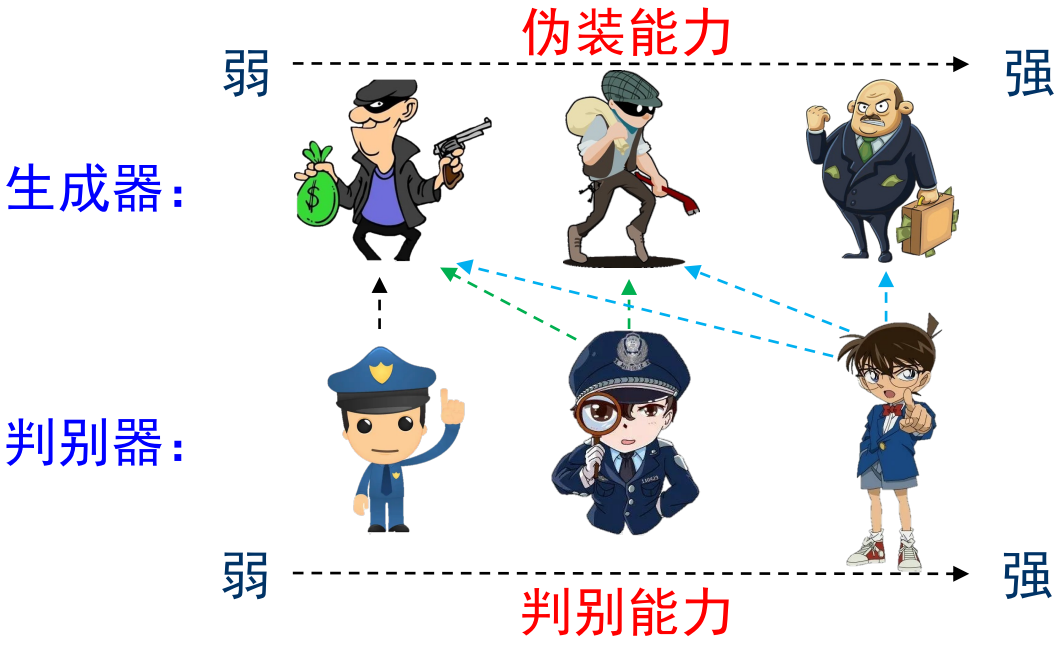
✓ 当分布P_{θ_g} 和 P_{data}未知时，无法直接计算分布间的距离

✓ 通过采样得到的生成数据和真实数据来捕获样本数据的分布



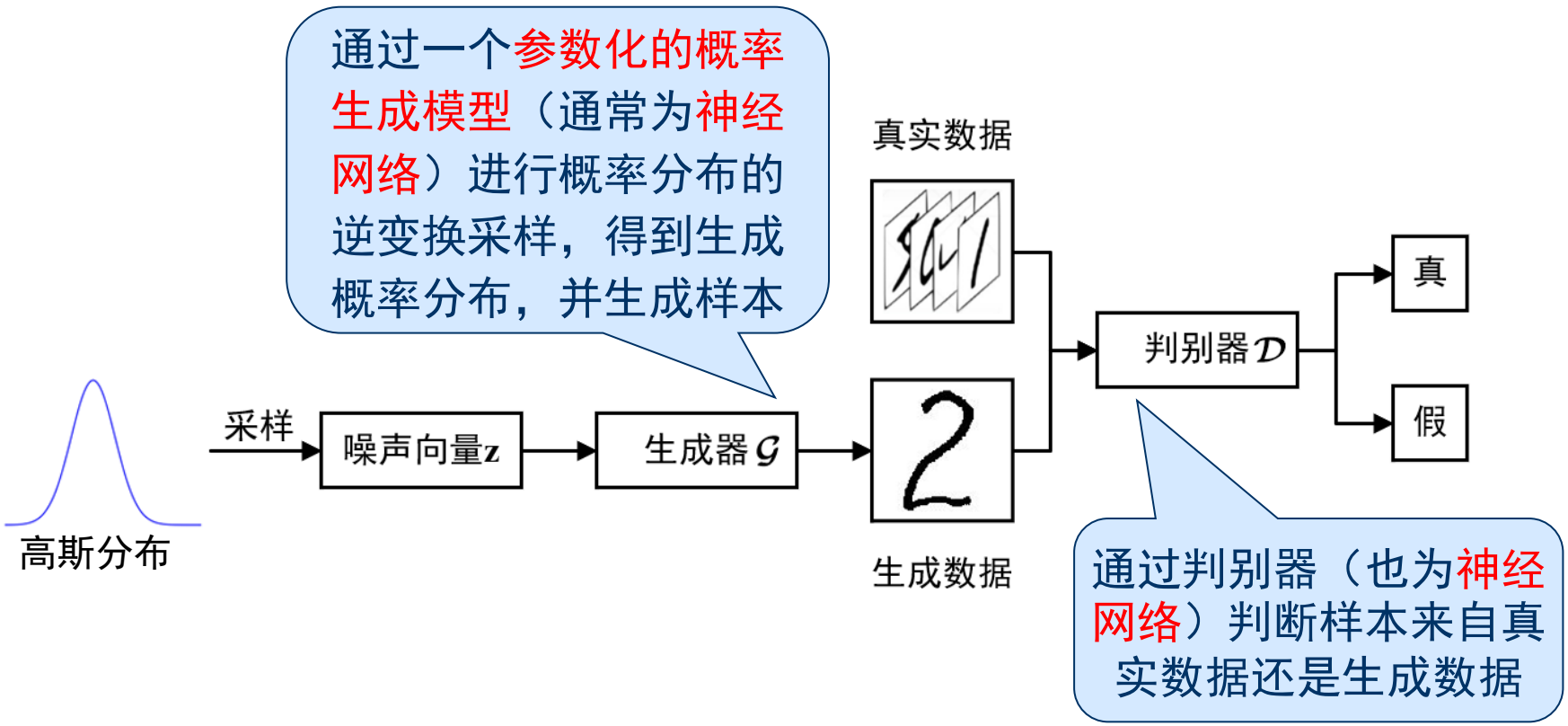
生成对抗网络 (6)

- **基本思想**
  - ✓ 每一个GAN框架都包含一对生成器和判别器
  - ✓ 源于博弈论中的二人零和博弈（Zero-Sum Game）思想



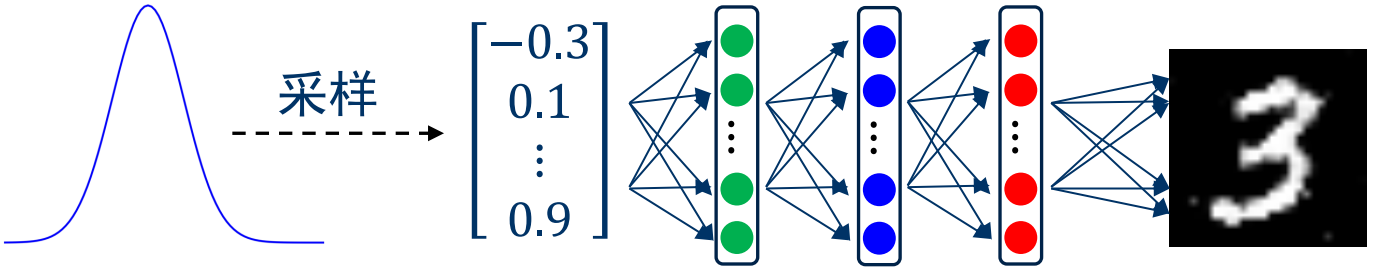
生成对抗网络 (7)

- **基本思想**



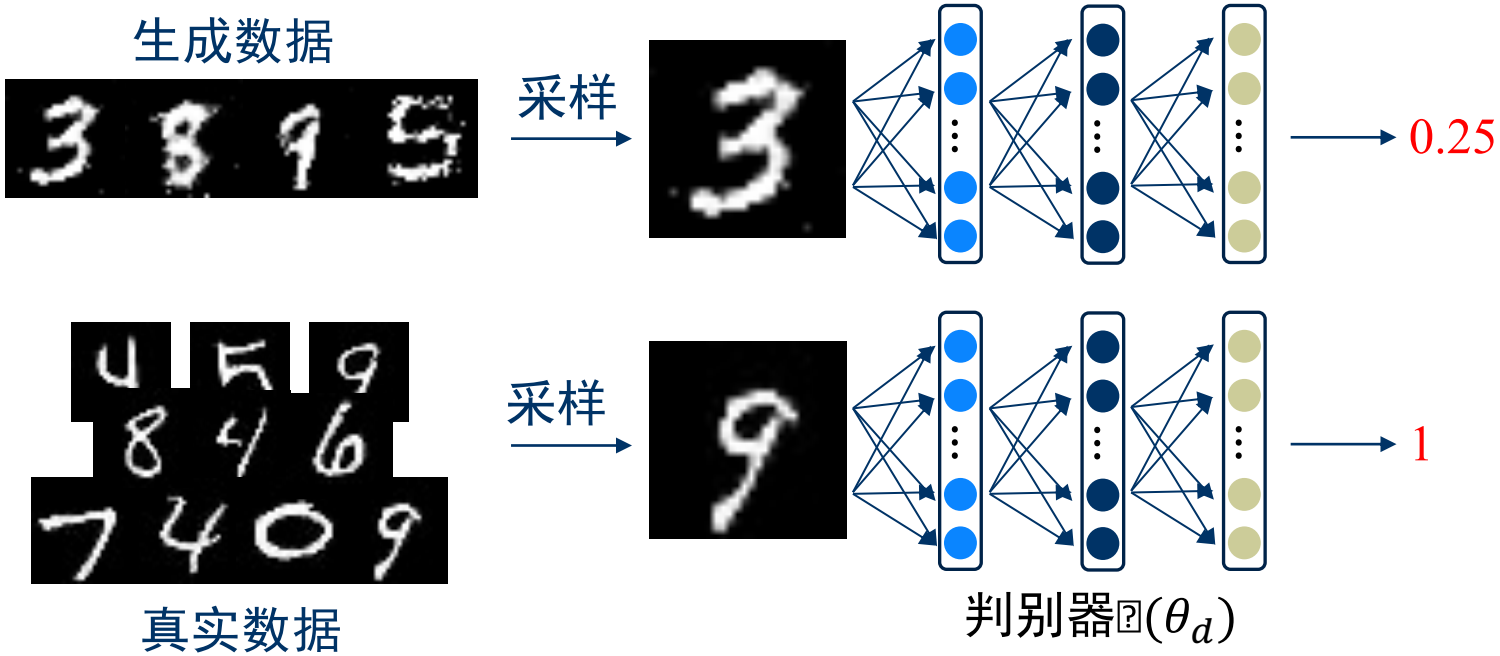
生成对抗网络 (8)

- **生成器**
  - ✓ 由参数θ_g控制来捕捉真实数据的分布
  - ✓ 从服从某一分布（如均匀分布、高斯分布）的噪声向量z生成服从真实数据分布的样本



生成对抗网络 (9)

- **判别器**
  - ✓ 由参数θ_d控制的二分类器
  - ✓ 估计一个样本来自真实数据（而非生成数据）的概率



生成对抗网络 (10)

- **损失函数**

生成器和判别器分别通过最小化和最大化函数$V(\mathcal{D},\mathcal{G})$来完成训练

$$\min_{\mathcal{G}}\max_{\mathcal{D}}V(\mathcal{D},\mathcal{G})=\min_{\mathcal{G}}\max_{\mathcal{D}}E_{\mathbf{x}\sim P_{data}}\left[\log\left(\mathcal{D}(\mathbf{x})\right)\right]+E_{\mathbf{z}\sim P_z}\left[\log\left(1-\mathcal{D}\left(\mathcal{G}(\mathbf{z})\right)\right)\right]$$

✓ 第一项

$E_{\mathrm{x}\sim P_{data}}\left[\log\left(\mathcal{D}(\mathrm{x})\right)\right]$为判别器将真实数据判别为真实数据的期望

✓ 第二项

$E_{\mathrm{z}\sim P_z}\left[\log\left(1-\mathcal{D}\left(\mathcal{G}(\mathrm{z})\right)\right)\right]$为判别器将生成数据判别为生成数据的期望

生成对抗网络 (11)

- **生成器的优化目标**
  - ✓ 将生成数据分布与真实数据分布相近
  - ✓ 让判别器难以区分

- **生成器的目标函数**

$V(\mathcal{D},\mathcal{G})$的第一项与$\mathcal{G}$无关

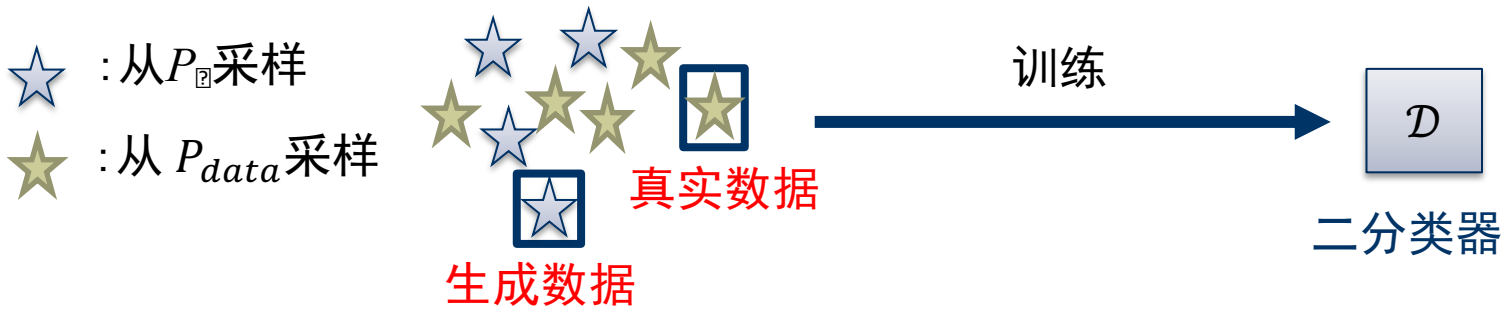
$$\mathcal{G}^* = \min_G Div(P_G, P_{data})$$

$$\rightarrow \min_G V(\mathcal{D},\mathcal{G}) = E_{\mathbf{z}\sim P_z}\left[log\left(1-\mathcal{D}(G(\mathbf{z}))\right)\right]$$

生成对抗网络 (12)

- **判别器的优化目标**

  尽可能区分生成数据和真实数据



- **判别器的目标函数**

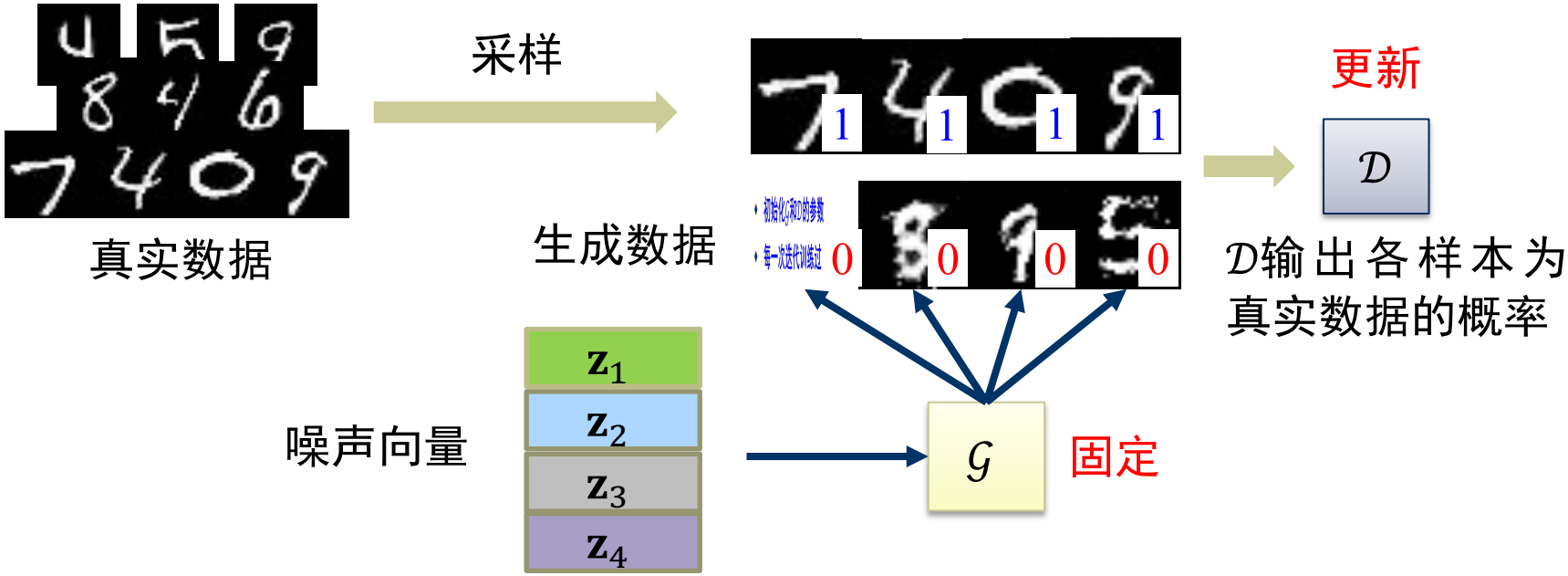
$$\mathcal{D}^* = \max_{\mathcal{D}} V(\mathcal{D}, \mathcal{G})$$

$$= \min_{\mathcal{D}} -E_{\mathbf{x}\sim P_{data}}[log\mathcal{D}(\mathbf{x})] - E_{\mathbf{z}\sim P_z}\left[log\left(1-\mathcal{D}(?(\mathbf{z}))\right)\right]$$

生成对抗网络 (13)

- **训练过程**
  - 初始化$\mathcal{G}$和$\mathcal{D}$的参数
  - 每一次迭代训练过程中

Step 1. 固定$\mathcal{G}$, 更新$\mathcal{D}$

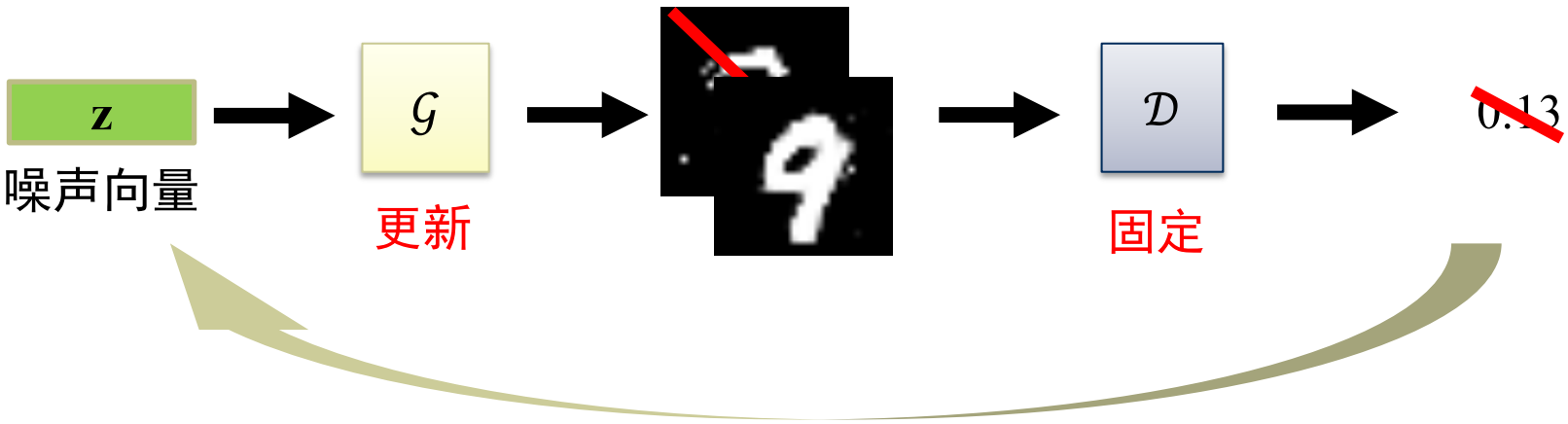


生成对抗网络 (14)

- **训练过程**

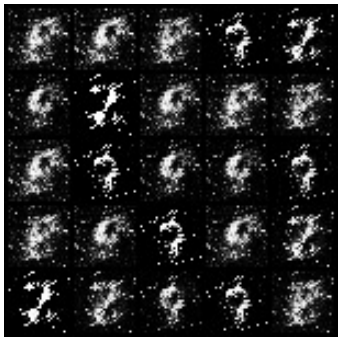
Step 2. 固定$\mathcal{D}$, 更新$\mathcal{G}$

$\mathcal{G}$的目的是“欺骗”$\mathcal{D}$

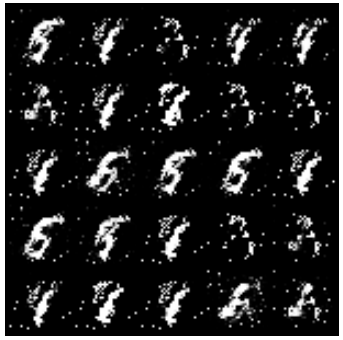


生成对抗网络 (15)

- **GAN训练效果展示**
  - ✓ 手写数字生成



迭代次数 1



50

100

150

200

250

生成对抗网络 (16)

- **全局最优解**

✓ GAN是存在全局最优解的

✓ 当生成的数据分布和真实数据分布完全一致时（即$P_G = P_{data}$），优化函数达到全局最小值

- **收敛性**

✓ 理论上如果生成器G和判别器D的学习能力足够强，两个模型可以收敛

✓ 实际中，GAN的收敛是很困难的，源于梯度消失和模式崩溃问题

✓ GAN的收敛性和均衡点存在性需要新的理论突破，模型结构和训练稳定性需进一步提高

生成对抗网络 (17)

- **优点**
  - ✓ 根据实际的结果，GAN比其他生成模型产生了更好的样本
  - ✓ 任何可微函数都可以构建生成器G和判别器D
  - ✓ 无需反复采样
- **缺点**
  - ✓ 训练需达到纳什平衡，现在还没有一个很好的达到纳什平衡的方法
  - ✓ GAN不适合处理离散形式的数据，如文本数据
  - ✓ 可解释性差，生成模型的分布 P_G 没有显式的表达

提纲

- 引例
- 降维算法概述
- 自编码器
- 自编码器的改进
- 变分自编码器
- 生成对抗网络
- 总结

总结

- 降维算法的基本思想与常见方法
- 典型的降维算法：
  - 自编码器的主要思想、模型结构及训练算法
  - 改进的自编码器
  - 变分自编码器的主要思想、模型结构及训练算法
  - 生成对抗网络的主要思想、模型结构及训练算法

结语

谢谢！