生成网络

刘家锋

哈尔滨工业大学

生成网络

自编码器

线性自编码器

- 输入层：中心化的样本集

$$\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\} \subset R^d$$

- 隐含层：对应编码过程

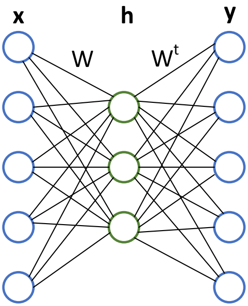
$$\mathbf{h} = \mathbf{W}\mathbf{x} \quad \in R^{d'}$$

- 输出层：对应解码过程

$$\mathbf{y} = \mathbf{W}^t\mathbf{h} \quad \in R^d$$

- 权值矩阵:

$$\mathbf{W} = \begin{pmatrix} \mathbf{w}_1^t \\ \vdots \\ \mathbf{w}_{d'}^t \end{pmatrix}_{d' \times d}$$



线性自编码器

- **线性AE的学习**：以训练样本作为输入，同时作为期望输出，优化权值矩阵$\mathbf{W}$

$$\min_{\mathbf{W}} J(\mathbf{W}) = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{y}_i||^2$$

  - 约束每个隐含层神经元权值矢量的长度：$||\mathbf{w}_i|| = 1$
  - 网络的输出：

$$\mathbf{y} = \mathbf{W}^t\mathbf{h} = \mathbf{W}^t\mathbf{W}\mathbf{x} = \sum_{k=1}^{d'} (\mathbf{w}_k^t\mathbf{x})\mathbf{w}_k$$

  - 可以看出：优化问题与PCA是一致的，解$\mathbf{w}_i$对应训练集$\mathcal{X}$协方差矩阵的最大特征矢量

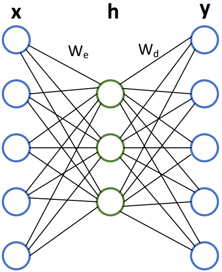
非线性自编码器

- 隐含层和输出层使用非线性激活函数

$$\text{输入：}\ \mathbf{x} \in [0,1]^d$$

$$\text{隐含层：}\ \mathbf{h} = F_e(\mathbf{W}_e\mathbf{x} + \mathbf{b}) \in [0,1]^{d'}$$

$$\text{输出层：}\ \mathbf{y} = F_d(\mathbf{W}_d\mathbf{h} + \mathbf{c}) \in [0,1]^d$$



- 网络学习
  - 平方误差优化：

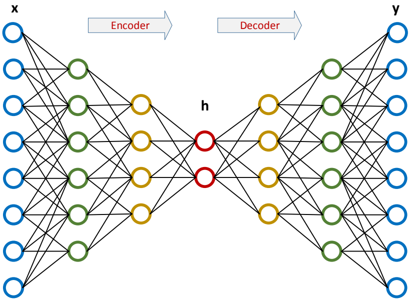
$$\min J(\mathbf{W}_e, \mathbf{W}_d, \mathbf{b}, \mathbf{c}) = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{y}_i||^2$$

  - 交叉熵优化：输出使用logistic函数，输入视为概率

$$\min_{\mathbf{W}_e, \mathbf{W}_d, \mathbf{b}, \mathbf{c}} H(\mathbf{x}, \mathbf{y}) = -\sum_{k=1}^{d} x_k \log y_k + (1 - x_k) \log (1 - y_k)$$

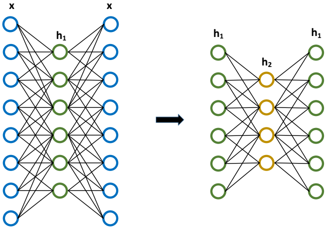
堆栈式自编码器

- **SAE: Stacked Autoencoder：**
  - 编码使用多个隐含层
  - 解码对称地增加隐含层数量



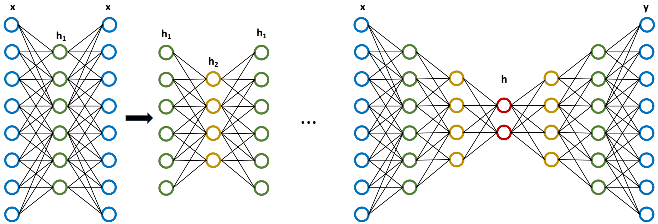
SAE的学习

- **第1隐含层：**
  - 加入一个（虚拟的）输出层
  - 使用训练样本集$\mathcal{X}$，无监督学习第1隐含层的参数
- **第2隐含层：**
  - 加入一个（虚拟的）输出层
  - 输入样本集$\mathcal{X}$，计算第1隐含层的输出$\mathcal{H}^1$
  - $\mathcal{H}^1$ 作为无监督样本集，学习第2隐层的参数
- **依次类推，直到最高隐含层**



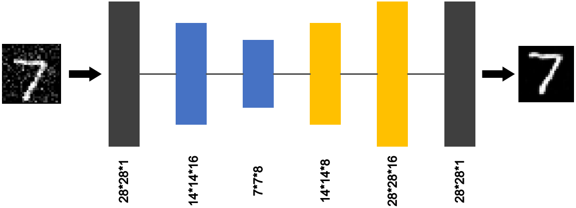
SAE的学习

- **Fine Tuning**:
  - 使用训练样本集$\mathcal{X}$
  - BP算法学习整个网络的参数



SAE的应用：图像去噪声

- **网络结构**
  - 输入有噪声图像
  - 编码器：卷积编码到隐空间
  - 解码器：反卷积解码到图像空间，输出降噪图像
- **网络学习**
  - 有噪声图像-无噪声图像对作为训练数据，学习网络



变分自编码器

变分自编码器

- **VAE: Variational Autoencoder**
  - 本质上仍然是采用无监督学习的方法，实现一个自编码器
  - 与AE和SAE的差别：输入空间到隐空间的映射不是确定性的，而是依据概率的
  - 与FA和Probabilistic PCA的差别：实现的是非线性映射，不要求样本在输入空间服从高斯分布

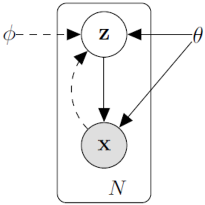
VAE的概率图模型表示

- 输入空间$\mathbf{x}$，隐空间$\mathbf{z}$
- **生成模型**：$\mathbf{x}$和$\mathbf{z}$的联合概率分布（实线部分）

$$p_\theta(\mathbf{x},\mathbf{z}) = p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})$$

- **变分近似**：条件分布$p_\theta(\mathbf{z}|\mathbf{x})$的近似（虚线部分）

$$q_\phi(\mathbf{z}|\mathbf{x}) \to p_\theta(\mathbf{z}|\mathbf{x})$$



神经网络实现

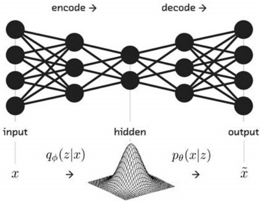
- **编码器**
  - 实现$q_\phi(\mathbf{z}|\mathbf{x})$，$\phi$为神经网络的参数：

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z};\boldsymbol{\mu}(\mathbf{x}),\boldsymbol{\sigma}^2(\mathbf{x})),\quad \boldsymbol{\sigma}^2(\mathbf{x})\text{为对角矩阵}$$

  - 高斯分布的实现：Reparameterize trick

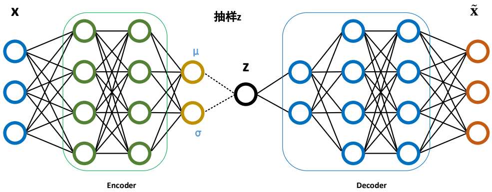
$$\mathbf{z} = g_\phi(\boldsymbol{\epsilon},\mathbf{x}) = \boldsymbol{\mu}(\mathbf{x}) + \boldsymbol{\sigma}^2(\mathbf{x})\cdot\boldsymbol{\epsilon},\quad \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$$

- **解码器**
  - 实现$p_\theta(\mathbf{x}|\mathbf{z})$，$\boldsymbol{\theta}$为神经网络的参数



VAE的工作过程

- 编码过程
  - 输入样本$\mathbf{x}$
  - 编码器网络计算：$\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\sigma}^2(\mathbf{x})$
  - 产生随机矢量：$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - 计算：$\mathbf{z} = \boldsymbol{\mu}(\mathbf{x}) + \boldsymbol{\sigma}^2(\mathbf{x}) \cdot \boldsymbol{\epsilon}$
- 解码过程
  - 解码器网络计算：$\boldsymbol{\mu}(\mathbf{z}), \boldsymbol{\sigma}^2(\mathbf{z})$
  - 随机抽样$\tilde{\mathbf{x}}$



VAE的学习

网络学习的优化目标：样本集$\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ 的对数似然

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ln p_\theta(\mathbf{x}_i)$$

引入变分近似：$q_\phi(\mathbf{z}|\mathbf{x}) \rightarrow p_\theta(\mathbf{z}|\mathbf{x})$，两者之间的KL-散度

$$\begin{aligned}
D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) &= \int q_\phi(\mathbf{z}|\mathbf{x}) \ln \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\
&= -\int q_\phi(\mathbf{z}|\mathbf{x}) \left[\ln \frac{p_\theta(\mathbf{z}, \mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} - \ln p_\theta(\mathbf{x}) \right] d\mathbf{z} \\
&= \ln p_\theta(\mathbf{x}) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\ln p_\theta(\mathbf{z}, \mathbf{x}) - \ln q_\phi(\mathbf{z}|\mathbf{x})]
\end{aligned}$$

VAE的学习

对数似然: 由于$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) \geq 0$，因此

$$\begin{aligned}
\ln p_\theta(\mathbf{x}) &= D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\ln p_\theta(\mathbf{z},\mathbf{x}) - \ln q_\phi(\mathbf{z}|\mathbf{x})] \\
&\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\ln p_\theta(\mathbf{z},\mathbf{x}) - \ln q_\phi(\mathbf{z}|\mathbf{x})]
\end{aligned}$$

变分下界：对数似然的优化可以转化为对变分下界的优化

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi},\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\ln p_\theta(\mathbf{z},\mathbf{x}) - \ln q_\phi(\mathbf{z}|\mathbf{x})] \\
&= -D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})\right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\ln p_\theta(\mathbf{x}|\mathbf{z})\right]
\end{aligned}$$

变分下界的梯度

第一项: $D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})\right]$ 可以直接积分得到解析式

$$-D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})\right] = \ln 2\pi + \frac{m}{2} + \frac{1}{2}\left[\sum_{j=1}^{m}(\sigma_j^2(\mathbf{x}))^2 + \|\boldsymbol{\mu}(\mathbf{x})\|^2\right]$$
$$+\frac{1}{2}\sum_{j=1}^{m}\ln\sigma_j^2(\mathbf{x})$$

其中，m为隐含空间中特征的维数。
第二项：$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\ln p_\theta(\mathbf{x}|\mathbf{z})\right]$计算梯度的困难在于对随机矢量$\mathbf{z}$取数学期望，使用Reparameterize trick，抽样L个随机矢量$\{\boldsymbol{\epsilon}_k\}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$

$$\mathbf{z}_k = \boldsymbol{\mu}(\mathbf{x}) + \boldsymbol{\sigma}^2(\mathbf{x})\cdot\boldsymbol{\epsilon}_k$$

Monte-Carlo近似计算：

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\ln p_\theta(\mathbf{x}|\mathbf{z})\right] \approx \frac{1}{L}\sum_{k=1}^{L}\ln p_\theta(\mathbf{x}|\mathbf{z}_k)$$

VAE的学习过程

I. 小批量抽样M个样本$\{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$

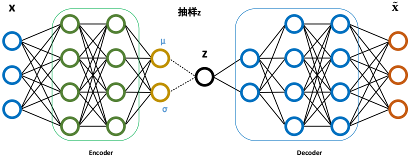
1. $\mathbf{x}_i$ 前馈计算到隐含层：$\boldsymbol{\mu}(\mathbf{x}_i), \boldsymbol{\sigma}^2(\mathbf{x}_i)$
2. 计算梯度$\nabla_\phi D_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}_i)||p_\theta(\mathbf{z})\right]$
3. 随机抽样L个矢量$\{\boldsymbol{\epsilon}_k\} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$，计算

$$\mathbf{z}_k = \boldsymbol{\mu}(\mathbf{x}_i) + \boldsymbol{\sigma}^2(\mathbf{x}_i) \cdot \boldsymbol{\epsilon}_k$$

4. $\{\mathbf{z}_k\}$ 前馈计算到输出层，近似计算梯度

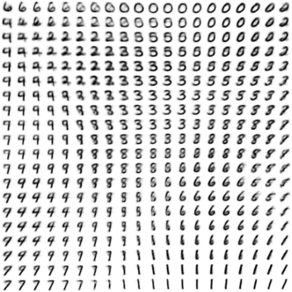
$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}\left[\ln p_\theta(\mathbf{x}|\mathbf{z})\right]$$

II. 累加小批量样本的梯度，修正神经网络的参数$\boldsymbol{\phi}, \boldsymbol{\theta}$



VAE的图像重构

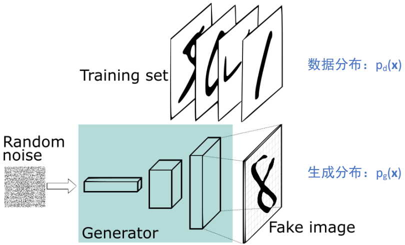
- 人脸图像和数字图像分别学习VAE，隐空间维数为2
- 在隐空间中等间隔抽样，生成$\mathbf{z}$，解码输出，产生图像



分布的距离及GAN

生成网络

- **生成网络的学习**
  - 给定采样自真实分布$p_d(\mathbf{x})$的训练数据集
  - 学习生成网络，希望网络生成的数据分布$p_g(\mathbf{x})$能够尽量接近真实分布$p_d(\mathbf{x})$



分布之间的相似性

- **生成网络的学习目标**
  - 生成网络学习的优化目标是两个分布之间的差异
  - 数据的真实分布$p_d(\mathbf{x})$是未知的，只有抽样的训练集
  - 生成数据的分布$p_g(\mathbf{x})$决定于网络的参数
  - 生成网络的学习过程，就是调整网络的参数，减小两个分布之间的差异

分布距离的定义

- 分布距离的简单定义
  - Total Variation

$$D\left[p_d(\mathbf{x}), p_g(\mathbf{x})\right] = \frac{1}{2}\int \left|p_d(\mathbf{x}) - p_g(\mathbf{x})\right| d\mathbf{x}$$

  - Hellinger

$$D\left[p_d(\mathbf{x}), p_g(\mathbf{x})\right] = \sqrt{\int \left(\sqrt{p_d(\mathbf{x})} - \sqrt{p_g(\mathbf{x})}\right)^2 d\mathbf{x}}$$

  - L_2

$$D\left[p_d(\mathbf{x}), p_g(\mathbf{x})\right] = \int \left(p_d(\mathbf{x}) - p_g(\mathbf{x})\right)^2 d\mathbf{x}$$

  - χ^2

$$D\left[p_d(\mathbf{x}), p_g(\mathbf{x})\right] = \int \frac{\left(p_d(\mathbf{x}) - p_g(\mathbf{x})\right)^2}{p_g(\mathbf{x})} d\mathbf{x}$$

分布距离的定义

- **生成网络的分布距离**
  - 上述距离，只有当已知分布$p_d(\mathbf{x})$和$p_g(\mathbf{x})$时才能够计算
  - 生成网络的学习，只有训练数据集，并且生成分布也是由网络参数隐式表示的，无法直接计算
  - 常用的生成网络学习分布距离定义
    - KLD：Kullback-Leibler Divergence
    - JSD：Jensen-Shannon Divergence
    - MMD：Maximum Mean Discrepancy
    - Wasserstein Distance

Information Entropy

- 信息熵
  - KLD和JSD需要用到信息论中熵的概念
  - 信息熵可以用来度量“无序性”或者“不确定性”
  - 离散随机变量x：N个取值，概率分别为$p_1, \cdots, p_N$

$$H(x) = -\sum_{i=1}^{N} p_i \log p_i$$

  - 连续随机变量x：概率密度函数为$p(x)$

$$H(x) = -\int p(x) \log p(x) dx$$

  - 取以2为底对数时，信息熵度量了编码离散随机变量x所需的平均比特数

信息编码举例

Table 编码8个字符

字符	概率	等长编码	非等长编码
a_1	0.9	000	0
a_2	0.09	001	10
a_3	0.01/6	010	11000
a_4	0.01/6	011	11001
a_5	0.01/6	100	11010
a_6	0.01/6	101	11011
a_7	0.01/6	110	11100
a_8	0.01/6	111	11101
平均码长		3	1.13

Conditional Entropy

- **条件熵**
  - 给定随机变量X的条件下，随机变量Y的熵：

$$
\begin{aligned}
H(Y|X) &= \int p(x) H(Y|X=x) dx \\
&= -\int p(x) \left\{ \int p(y|x) \log p(y|x) dy \right\} dx \\
&= -\iint p(x,y) \log p(y|x) dx dy \\
&= -\iint p(x,y) \log \frac{p(x,y)}{p(x)} dx dy \\
&= H(X,Y) - H(X)
\end{aligned}
$$

Cross Entropy

- **交叉熵**
  - 离散随机变量x的真实分布为$\{p_1, \cdots, p_N\}$
  - 交叉熵表示以另外一个分布$\{q_1, \cdots, q_N\}$来编码x，所需要的平均比特数：

$$H(p,q) = -\sum_{i=1}^{N} p_i \log q_i$$

  - 连续形式：

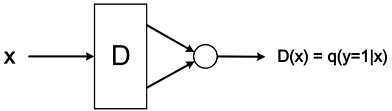
$$H(p,q) = -\int p(x) \log q(x) dx$$

交叉熵损失函数

- **判别网络**
  - 考虑一个两分类的判别神经网络D，输出神经元采用对数Sigmoid函数，模型化后验概率

$$q(y=1|\mathbf{x}) = D(\mathbf{x}), \quad q(y=0|\mathbf{x}) = 1 - D(\mathbf{x})$$

  - 网络参数的学习需要采用对数似然函数作为损失函数，与交叉熵的损失函数是一致的



交叉熵损失函数

- **对数似然**
  - 输入类别标记为y_i的训练样本$\mathbf{x}_i$，网络的输出为$D(\mathbf{x}_i)$，预测的后验概率可以表示为：

$$q(y_i|\mathbf{x}_i) = D(\mathbf{x})^{y_i}[1 - D(\mathbf{x})]^{1-y_i}$$

  - 对数似然：

$$\ln q(y_i|\mathbf{x}_i) = y_i \ln D(\mathbf{x}_i) + (1 - y_i) \ln [1 - D(\mathbf{x}_i)]$$

  - 训练集上的对数似然函数：

$$L = \sum_{\mathbf{x} \in \text{正例}} \ln D(\mathbf{x}) + \sum_{\mathbf{x} \in \text{反例}} \ln [1 - D(\mathbf{x})]$$

交叉熵损失函数

- **交叉熵的观点来看**
  - 训练样本$\mathbf{x}$类别标记y的分布为

$$p: \quad y \sim \{1-y, y\}$$

  - 网络预测类别标记的分布为

$$q: \quad \hat{y} \sim \{1-D(\mathbf{x}), D(\mathbf{x})\}$$

  - 分布p对q的交叉熵与负的对数似然是一致的：

$$H(p,q) = -y \ln D(\mathbf{x}) - (1-y) \ln D(\mathbf{x})$$

  - 对于所有的训练样本，可以写成数学期望的形式：

$$H(p,q) = -\mathbb{E}_{\mathbf{x}\sim p_{\text{正}}(\mathbf{x})} \ln D(\mathbf{x}) - \mathbb{E}_{\mathbf{x}\sim p_{\text{反}}(\mathbf{x})} \ln\left[1-D(\mathbf{x})\right]$$

Kullback – Leibler divergence

- **K-L散度(相对熵)**
  - 随机变量x上的两个分布$p(x)$和$q(x)$：

$$\begin{aligned} KL(p||q) &= \int p(x)\log\frac{p(x)}{q(x)}dx \\ &= -\int p(x)\log q(x)dx + \int p(x)\log p(x)dx \\ &= H(q,p) - H(p) \end{aligned}$$

- 含义
  - 编码：使用对于q最优的码本编码p需要增加的额外比特数
  - 机器学习：
    1. 使用分布p代替分布q所获得的信息增益
    2. 以分布q来近似真实分布p所带来的信息损失
    3. 分布q与p之间的差异

Kullback – Leibler divergence

- **非对称性**
  - KLD是非对称的距离度量

$$KL\big(p_d(\mathbf{x})||p_g(\mathbf{x})\big) \neq KL\big(p_g(\mathbf{x})||p_d(\mathbf{x})\big)$$

  - 非对称的KLD不能直接作为loss函数，用于优化生成网络
  - 如果使用不同的KLD优化网络，会得到不同的结果

优化KLD的结果

- **优化loss函数**$KL(p_d(\mathbf{x})||p_g(\mathbf{x}))$

$$KL(p_d(\mathbf{x})||p_g(\mathbf{x})) = \int p_d(\mathbf{x}) \log \frac{p_d(\mathbf{x})}{p_g(\mathbf{x})} d\mathbf{x}$$

  - $p_d(\mathbf{x}) > p_g(\mathbf{x})$时，loss比较大；含义是生成的样本要尽量覆盖数据的真实分布，不能漏掉真实数据
  - $p_d(\mathbf{x}) < p_g(\mathbf{x})$时，loss比较小；含义是多生成一些非真实样本也无所谓，损失很小
- **优化loss函数**$KL(p_g(\mathbf{x})||p_d(\mathbf{x}))$
  - 含义是生成的样本一定要是真实的样本，但有一些真实样本可能生成不出来

Jensen-Shannon Divergence

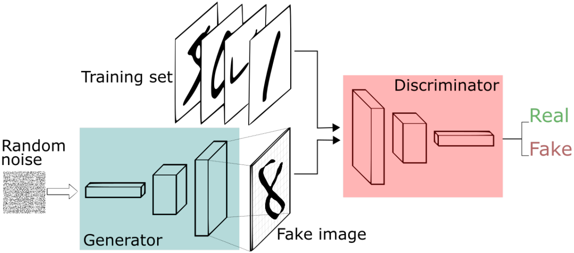
- **JSD：Jensen-Shannon Divergence**
  - JSD是一种对称的距离度量

$$
\begin{aligned}
JSD(p_d(\mathbf{x}), p_g(\mathbf{x})) = & \frac{1}{2}\left[KL\left(p_d(\mathbf{x}) || \frac{p_d(\mathbf{x}) + p_g(\mathbf{x})}{2}\right)\right. \\
& \left. + KL\left(p_g(\mathbf{x}) || \frac{p_d(\mathbf{x}) + p_g(\mathbf{x})}{2}\right)\right]
\end{aligned}
$$

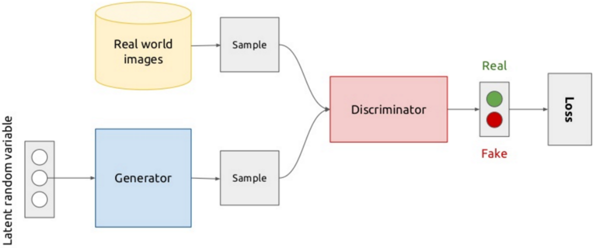
  - 当$p_d(\mathbf{x}) = p_g(\mathbf{x})$时，$JSD = 0$，取得最小值
  - 可以证明，生成对抗网络本质上优化的就是JSD损失函数

GAN:Generative Adversarial Nets

- 对抗神经网络
  - GAN是一种学习生成模型的方法
  - 可以产生逼真的图像或视频



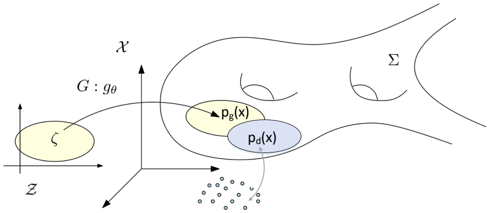
GAN的学习



- 随机产生$m/2$个输入，G前馈产生$m/2$幅图像（标签0）
- 抽样$m/2$幅真实图像（标签1）
- 生成网络(G)的学习：以D的输出交叉熵最大为目标，更新生成网络G
- 判别网络(D)的学习：以D的输出交叉熵最小为目标，更新生成网络D

GAN的理解

- 真实样本分布于嵌入在图像空间$\mathcal{X}$ 中的流形Σ
- 真实样本服从分布$p_d(\mathbf{x})$，但很难描述和估计
- 在一个潜在空间$\mathcal{Z}$上抽样服从分布ζ的随机样本，然后由G映射到空间$\mathcal{X}$
- 映射之后的样本在空间$\mathcal{X}$上的分布为$p_g(\mathbf{x})$
- 学习的目标是找到一个适合的G，使得$p_g(\mathbf{x}) \rightarrow p_d(\mathbf{x})$



GAN的理解

- **GAN的实现**
  - 生成网络G实现映射g_θ，网络的参数表示为$\boldsymbol{\theta}$
  - 生成网络G的输入是潜在空间$\mathcal{Z}$上抽样，ζ可以使用均匀分布或正态分布
- **生成网络的学习**
  - 引入判别网络D，用于度量分布$p_g(\mathbf{x})$与$p_d(\mathbf{x})$之间的差异
  - 生成网络G与判别网络D以对抗的方式优化目标函数

$$\min_G \max_D L(D, G) = \mathbb{E}_{\mathbf{x}\sim p_d(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim\zeta(\mathbf{z})}[1 - D(G(\mathbf{z}))]$$

GAN与JSD

- **GAN的学习本质**
  - 可以证明GAN的对抗学习过程，本质上优化的是真实数据分布$p_d(\mathbf{x})$与生成数据分布$p_g(\mathbf{x})$之间的Jensen-Shannon Divergence
  - 当真实数据分布于一个低维流形时，JSD关于生成器的参数$\boldsymbol{\theta}$是不连续的，会给GAN的学习带来困难

Maximum Mean Discrepancy

- **MMD的定义：**

  Let $\mathbf{x}$ be random variable defined on a topological space $\mathcal{X}$, with respective Borel probability measures $p(\mathbf{x})$ and $q(\mathbf{x})$, Let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \to R$. Define the maximum mean discrepancy(MMD) as

$$MMD[\mathcal{F}, p, q] := \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim q(\mathbf{x})}[f(\mathbf{x})] \right)$$

Maximum Mean Discrepancy

- **再生核希尔伯特空间中计算MMD**
  - MMD需要计算空间$\mathcal{F}$中所有函数$f(\mathbf{x})$的均值
  - 将$\mathcal{F}$限定为定义在一个有界Hilbert空间(Universal RKHS)中的所有函数，对应的核函数为$k(\cdot,\cdot)$
  - 可以证明，在这样的函数空间$\mathcal{F}$上定义的MMD可以使用核函数进行计算：

$$
\begin{aligned}
MMD^2[\mathcal{F},p,q] =& \mathbb{E}_{\mathbf{x},\mathbf{x}'\sim p(\mathbf{x})}[k(\mathbf{x},\mathbf{x}')] + \mathbb{E}_{\mathbf{x},\mathbf{x}'\sim q(\mathbf{x})}[k(\mathbf{x},\mathbf{x}')] \\
&- 2\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}),\mathbf{x}'\sim q(\mathbf{x})}[k(\mathbf{x},\mathbf{x}')]
\end{aligned}
$$

MMD GAN

- **GMMN: Generative Moment Matching Network**
  - 取消GAN的判别器D，将生成器的输出分布与数据的真实分布用MMD度量距离，直接优化MMD
  - 小批量抽样真实数据：$\{\mathbf{x}_1^d, \cdots, \mathbf{x}_N^d\} \sim p_d(\mathbf{x})$
  - GMMN小批量生成数据：$\{\mathbf{x}_1^g, \cdots, \mathbf{x}_M^g\} \sim p_g(\mathbf{x})$
  - 优化目标函数：

$$L_{MMD^2} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} k(\mathbf{x}_i^d, \mathbf{x}_j^d) + \frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1}^{M} k(\mathbf{x}_i^g, \mathbf{x}_j^g) - \frac{2}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} k(\mathbf{x}_i^d, \mathbf{x}_j^g)$$

MMD GAN

- **GMMN: Generative Moment Matching Network**
  - 核函数一般选择高斯函数：

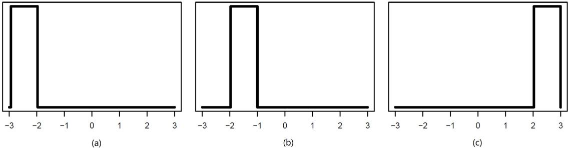
$$k(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{y}\|^2\right)$$

  - 计算目标函数关于GMMN的输出$\mathbf{x}_j^g$的梯度：

$$\begin{aligned}\frac{\partial L_{MMD^2}}{\partial \mathbf{x}_j^g} =& \frac{2}{M^2}\sum_{i=1}^{M}\frac{1}{\sigma^2}k(\mathbf{x}_i^g,\mathbf{x}_j^g)(\mathbf{x}_j^g-\mathbf{x}_i^g) \\ &-\frac{2}{NM}\sum_{i=1}^{N}\frac{1}{\sigma^2}k(\mathbf{x}_i^d,\mathbf{x}_j^g)(\mathbf{x}_j^g-\mathbf{x}_i^d)\end{aligned}$$

Wasserstein Distance

- **分布距离度量的缺陷**
  - KLD和MMD这些分布距离度量，都是在随机变量的每一个点上计算两个分布之间的差异
  - 当随机变量的每个取值之间没有先后次序关系时，这样度量是合理的，例如分类问题
  - 当取值之间存在次序关系时，存在度量的不合理性；例如下图中分布a和b更相似，与c的差异更大，但按照之前方法度量的相似性均为0



Wasserstein Distance

- **离散分布的Wasserstein Distance**
  - 离散随机变量A有n个取值$\{a_1, \cdots, a_n\}$，对应的概率为$\{p_1, \cdots, p_n\}$
  - 离散随机变量B有m个取值$\{b_1, \cdots, b_m\}$，对应的概率为$\{q_1, \cdots, q_m\}$
  - 令d_{ij}为取值a_i和b_j之间的距离，求解最小流$F = [f_{ij}]$

$$\min_F \sum_{i=1}^{n} \sum_{j=1}^{m} f_{ij} d_{ij}$$

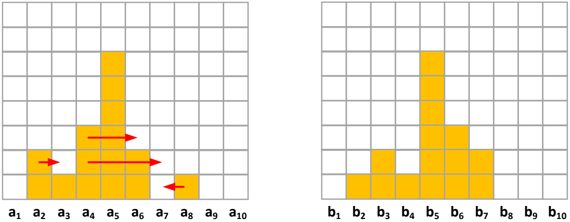
subject to:

$$f_{ij} \geq 0, \quad \sum_{i=1}^{n} \sum_{j=1}^{m} f_{ij} = 1$$

$$\sum_{j=1}^{m} f_{ij} = p_i, \quad \sum_{i=1}^{n} f_{ij} = q_j$$

Wasserstein Distance

- 离散分布的**Wasserstein Distance**
  - Wasserstein Distance也称为是Earth Move Distance，求解的优化问题也称为最优传输问题
  - “流(Flow)” f_{ij}表示由a_i向b_j传输的量，d_{ij}是传输的代价
  - 优化问题求解的就是最小代价的传输“流”



Optimal Transport

Wasserstein Distance

- **离散分布的Wasserstein Distance**
  - 令F^*是最优传输问题的解
  - 分布p和q之间的Earth Move Distance定义为传输的最小代价

$$EMD(p,q) = \sum_{i=1}^{n}\sum_{j=1}^{m} f^*_{ij} d_{ij}$$

Wasserstein Distance

- **由离散分布到连续分布**
  - 由于$f_{ij} \geq 0, \sum_{i=1}^{n} \sum_{j=1}^{m} f_{ij} = 1$，流$F$可以看作是随机变量$A$和$B$上的二维分布，两个边缘分布刚好是$p$和$q$
  - 对于连续的随机变量$\mathbf{x}$和$\mathbf{y}$，可以类似地定义联合分布密度$\gamma(\mathbf{x}, \mathbf{y})$，满足

$$\gamma(\mathbf{x}, \mathbf{y}) \geq 0, \quad \int \gamma(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} = 1$$

$$\int \gamma(\mathbf{x}, \mathbf{y}) d\mathbf{y} = p(\mathbf{x}), \quad \int \gamma(\mathbf{x}, \mathbf{y}) d\mathbf{x} = q(\mathbf{y})$$

Wasserstein Distance

- **连续分布的最优传输问题**
  - 令$d(\mathbf{x},\mathbf{y})$为$\mathbf{x}$与$\mathbf{y}$之间的距离，最优传输问题：

$$\min_{\gamma(\mathbf{x},\mathbf{y})} \int d(\mathbf{x},\mathbf{y})\gamma(\mathbf{x},\mathbf{y})d\mathbf{x}d\mathbf{y}$$

- **Wasserstein metric**
  - 给定$\beta \geq 1$，Wasserstein metric定义为

$$W_\beta(p,q) := \left[\min_{\gamma} \int \|\mathbf{x}-\mathbf{y}\|^\beta \gamma(\mathbf{x},\mathbf{y})d\mathbf{x}d\mathbf{y}\right]^{\frac{1}{\beta}}$$

计算问题

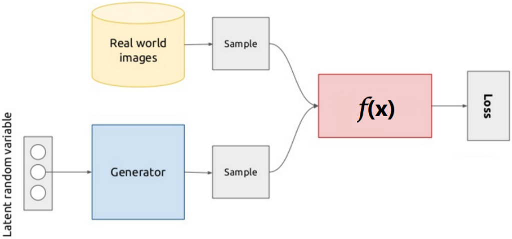
- **Kantorovich-Rubinstein对偶**
  - 最优传输问题的求解是很困难的
  - 可以证明，Wasserstein metric等价于

$$W(p_d, p_g) = \sup_{\|f\|_L \leq 1} \left\{ \mathbb{E}_{\mathbf{x}\sim p_d}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_g}[f(\mathbf{x})] \right\}$$

  - Wasserstein metric的对偶形式与MMD是类似的

WGAN

- **Wasserstein GAN**
  - WGAN的结构与GAN类似，不同的是用一个神经网络来实现函数$f(\mathbf{x})$，代替了GAN中的判别器D
  - 学习过程也类似GAN，不同的是对抗学习的目标函数为$W(p_d, p_g)$，而不是判别器输出的交叉熵
  - 与GMMN的差别是用一个神经网络来实现函数$f(\mathbf{x})$，而不是核函数



END