4.1 Variation Autoencoder

任务

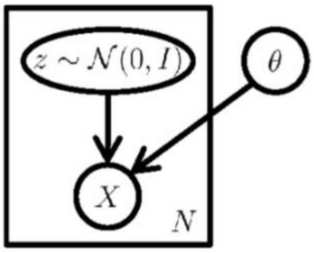
假设有一个无标签数据集$\boldsymbol{X}=\left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{N}$，其中$\boldsymbol{x}^{(i)}$代表一幅图像，$\boldsymbol{x}^{(i)}$当中的每个元素是图像个每个像素。我们想要学习一个概率模型$p(\boldsymbol{x})$，以生成和数据集相似但不同的更多图像。如果解决了该项问题，则拥有了学习高维数据复杂概率模型的能力。这种生成看起来真实的图像的能力对于视频游戏的设计非常有用。

假设与生成模型：我们假设每个$\boldsymbol{x}^{(i)}$都拥有一个未观测到的标签$\boldsymbol{z}$，而$\boldsymbol{z}$拥有比$\boldsymbol{x}^{(i)}$低得多的维数；同时假设图像的生成模型如下：

$$\boldsymbol{z} \sim p(\boldsymbol{z})=\mathcal{N}(\mathbf{0}, \boldsymbol{I})$$
$$\boldsymbol{x} \sim p_{\theta}(\boldsymbol{x} \mid \boldsymbol{z})$$

其中$\boldsymbol{I}$是单位矩阵，θ是模型参数。从而我们有

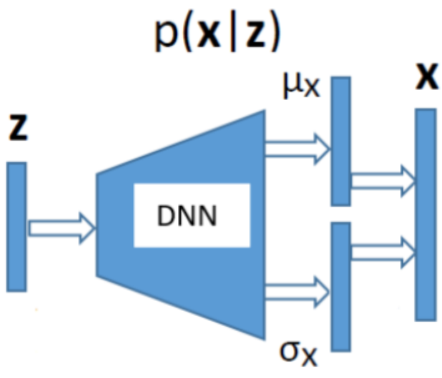
$$p_{\theta}(\boldsymbol{x})=\int p_{\theta}(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z}) \mathrm{d} \boldsymbol{z}$$



进而我们假设，$p_{\theta}(\boldsymbol{x} \mid \boldsymbol{z})$也是一个高斯分布：

$$p_{\theta}(\boldsymbol{x} \mid \boldsymbol{z})=\mathcal{N}\left(\boldsymbol{x} \mid \mu_{x}(\boldsymbol{z}, \boldsymbol{\theta}), \sigma_{x}^{2}(\boldsymbol{z}, \boldsymbol{\theta}) \boldsymbol{I}\right)$$

其中$\mu_{x}(\boldsymbol{z}, \boldsymbol{\theta})$是均值向量，$\sigma_{x}^{2}(\boldsymbol{z}, \boldsymbol{\theta}) \boldsymbol{I}$是对角协方差矩阵，以上两个向量通过一个参数为$\boldsymbol{\theta}$的深度神经网络由$\boldsymbol{z}$决定。



目标函数

与之前类似，我们需要最大化似然函数来为模型参数取值：

$$\log p_\theta(\boldsymbol{X}) = \sum_{i=1}^{N} \log p_\theta(\boldsymbol{x}^{(i)})$$

其中

$$\log p_\theta(\boldsymbol{x}^{(i)}) = \log \int p_\theta(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z}) \mathrm{d}\boldsymbol{z}$$

我们可以使用梯度上升方法来最大化似然函数，但是梯度$\nabla_\theta \log p_\theta(\boldsymbol{x}^{(i)})$因为包含了积分，因而无法追踪其方向。

我们可以采用朴素蒙特卡洛梯度估计 Naïve Monte Carlo Gradient Estimator 来估计$p_\theta(\boldsymbol{x}^{(i)})$进而估计其梯度：

$$p_\theta(\boldsymbol{x}^{(i)}) \approx \frac{1}{L} \sum_{l=1}^{L} p_\theta(\boldsymbol{x}^{(i)}|\boldsymbol{z}^{(l)}) = E_{\boldsymbol{z}}[p_\theta(\boldsymbol{x}^{(i)}|\boldsymbol{z})]$$

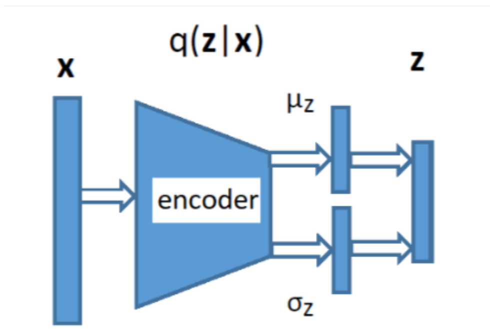
进而可以计算$\nabla_\theta \log p_\theta(\boldsymbol{x}^{(i)})$。

但是这样实际上仍然无法达成我们的目标。因为$\boldsymbol{x}$是个非常高维的数据，拥有成千上百万的维度，因而给定一个$\boldsymbol{z}$，$p_\theta(\boldsymbol{x}|\boldsymbol{z})$实际上是极度扭曲 highly skewed 的，仅在一些较小的区域取得较大的值，其他区域的值均可忽略不计；从而使得上式当中L的取值要非常大才能得到一个近似的估计。

为了解决以上问题，我们引入一个识别模型$q_\phi(\boldsymbol{z}|\boldsymbol{x})$，其也是一个高斯分布：

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\mu_z(\boldsymbol{x},\phi), \sigma_z^2(\boldsymbol{x},\phi)\boldsymbol{I})$$

其中$\mu_z(\boldsymbol{x},\phi)$是均值向量，$\sigma_{\boldsymbol{z}}^2(\boldsymbol{x},\phi)\boldsymbol{I}$是对角协方差矩阵，以上两个向量通过一个参数为$\boldsymbol{\phi}$的深度神经网络由$\boldsymbol{x}$决定。



现在的问题是，如何利用以上的识别模型$q_\theta(\boldsymbol{z}|\boldsymbol{x})$来最大化似然函数：

$$\log p_\theta(\boldsymbol{X}) = \sum_{i=1}^{N} \log p_\theta(\boldsymbol{x}^{(i)})$$

变分下确界 Variational Lower Bound:

利用以上的识别模型，我们可以把似然函数重写为：

$$\log p_\theta\left(\boldsymbol{x}^{(i)}\right) = E_{\boldsymbol{z}\sim q_\phi\left(\boldsymbol{z}\middle|\boldsymbol{x}^{(i)}\right)}\left[\log p_\theta\left(\boldsymbol{x}^{(i)}\right)\right]$$

$$= E_{\boldsymbol{z}\sim q_\phi}\left[\log \frac{p_\theta\left(\boldsymbol{x}^{(i)}|\boldsymbol{z}\right)p_\theta(\boldsymbol{z})}{p_\theta\left(\boldsymbol{z}|\boldsymbol{x}^{(i)}\right)}\right]$$

$$= E_{\boldsymbol{z}\sim q_\phi}\left[\log \frac{p_\theta\left(\boldsymbol{x}^{(i)}|\boldsymbol{z}\right)p_\theta(\boldsymbol{z})}{p_\theta\left(\boldsymbol{z}|\boldsymbol{x}^{(i)}\right)} \frac{q_\phi\left(\boldsymbol{z}\middle|\boldsymbol{x}^{(i)}\right)}{q_\phi\left(\boldsymbol{z}\middle|\boldsymbol{x}^{(i)}\right)}\right]$$

$$= E_{\boldsymbol{z}\sim q_\phi}\left[\log p_\theta\left(\boldsymbol{x}^{(i)}|\boldsymbol{z}\right)\right] - E_{\boldsymbol{z}\sim q_\phi}\left[\log \frac{q_\phi\left(\boldsymbol{z}\middle|\boldsymbol{x}^{(i)}\right)}{p_\theta(\boldsymbol{z})}\right]$$

$$+ E_{\boldsymbol{z}\sim q_\phi}\left[\log \frac{q_\phi\left(\boldsymbol{z}\middle|\boldsymbol{x}^{(i)}\right)}{p_\theta\left(\boldsymbol{z}|\boldsymbol{x}^{(i)}\right)}\right]$$

$$= E_{\boldsymbol{z}\sim q_\phi}\left[\log p_\theta\left(\boldsymbol{x}^{(i)}|\boldsymbol{z}\right)\right] - \mathcal{D}_{KL}\left[q_\phi\left(\boldsymbol{z}\middle|\boldsymbol{x}^{(i)}\right)||p_\theta(\boldsymbol{z})\right]$$

$$+ E_{\boldsymbol{z}\sim q_\phi}\left[\log \frac{q_\phi\left(\boldsymbol{z}\middle|\boldsymbol{x}^{(i)}\right)}{p_\theta\left(\boldsymbol{z}|\boldsymbol{x}^{(i)}\right)}\right]$$

$$= \mathcal{L}\left(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}, \boldsymbol{\phi}\right) + \mathcal{D}_{KL}\left[q_\phi\left(\boldsymbol{z}\middle|\boldsymbol{x}^{(i)}\right)||p_\theta\left(\boldsymbol{z}|\boldsymbol{x}^{(i)}\right)\right]$$

其中 KL 散度 KL Divergence $\mathcal{D}_{KL}$用于衡量两个分布之间的差异程度：

对于连续变量的分布，有：

$$\mathcal{D}_{KL}(p||q) = \int p(x)\log\frac{p(x)}{q(x)}$$

对于离散变量的分布，有：

$$\mathcal{D}_{KL}(p||q) = \sum p(x)\log\frac{p(x)}{q(x)} = E_p[\log p(x)] - E_p[\log q(x)]$$

KL 散度特性为：

$$\mathcal{D}_{KL}(p||q) \geq 0$$

当且仅当$p = q$时取等号，即两个分布是相同的。

因此我们有如下变分下确界：

$$\log p_\theta\left(\boldsymbol{x}^{(i)}\right) \geq \mathcal{L}\left(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}, \boldsymbol{\phi}\right)$$

其中

$$\mathcal{L}\left(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}, \boldsymbol{\phi}\right) = E_{\boldsymbol{z}\sim q_\phi}\left[\log p_\theta\left(\boldsymbol{x}^{(i)}|\boldsymbol{z}\right)\right] - \mathcal{D}_{KL}\left[q_\phi\left(\boldsymbol{z}\middle|\boldsymbol{x}^{(i)}\right)||p_\theta(\boldsymbol{z})\right]$$

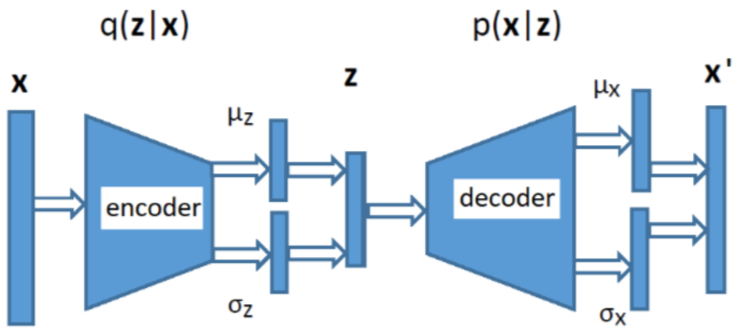
因而我们的目标函数转化为：求模型参数$\boldsymbol{\theta}, \boldsymbol{\phi}$使得$\mathcal{L}\left(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}, \boldsymbol{\phi}\right)$最大化。

以上的识别模型$q_\phi\left(\boldsymbol{z}\middle|\boldsymbol{x}^{(i)}\right)$可以视作一个编码器 encoder，其利用输入$\boldsymbol{x}^{(i)}$的概率编码成为一个隐向量 latent vector $\boldsymbol{z}$；生成模型$p_\theta\left(\boldsymbol{z}|\boldsymbol{x}^{(i)}\right)$可以看做一个解码器 decoder，其利用编码器生成的隐向量$\boldsymbol{z}$重新转化为数据空间的一个输出和输入

$\boldsymbol{x}^{(i)}$相匹配。

$\mathcal{L}(\boldsymbol{x}^{(i)},\boldsymbol{\theta},\boldsymbol{\phi})$当中的第一项$E_{\boldsymbol{z}\sim q_\phi}[\log p_\theta(\boldsymbol{x}^{(i)}|\boldsymbol{z})]$描述了解码输出和之前的输入$\boldsymbol{x}^{(i)}$的匹配程度，其称之为重构误差 Reconstruction Error；第二项$\mathcal{D}_{KL}$是一个正则项，用于激励生成$\boldsymbol{z}$的后验概率分布$q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)})$更接近于先验概率分布$p_\theta(\boldsymbol{z})$。

以上的方法被称作变分自动编码器 Variational Autoencoder/VAE.



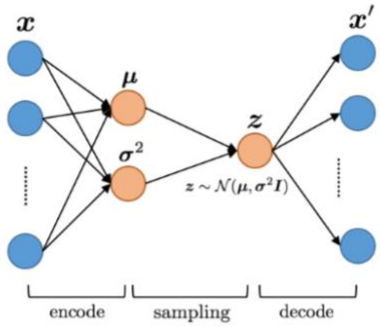
优化

令

$$\mathcal{L}_1 = E_{\boldsymbol{z}\sim q_\phi}[\log p_\theta(\boldsymbol{x}^{(i)}|\boldsymbol{z})]$$
$$\mathcal{L}_2 = -\mathcal{D}_{KL}[q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)})||p_\theta(\boldsymbol{z})]$$

从而

$$\mathcal{L}(\boldsymbol{x}^{(i)},\boldsymbol{\theta},\boldsymbol{\phi}) = \mathcal{L}_1 + \mathcal{L}_2$$



先看重构误差$\mathcal{L}_1$：计算需要用到采样

$$\mathcal{L}_1 = E_{\boldsymbol{z}\sim q_\phi}[\log p_\theta(\boldsymbol{x}^{(i)}|\boldsymbol{z})] \approx \frac{1}{L}\sum_{l=1}^{L}\log(p_\theta(\boldsymbol{x}^{(i)}|\boldsymbol{z}^{(i,l)}))$$

其中$\boldsymbol{z}^{(i,l)}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)})$.

但是该式存在一个问题，即采样过程会使梯度弥散，因为等式的左半边依赖于模

型参数ϕ，但是右半边并没有体现出来，因而从$\mu_z(\boldsymbol{x},\phi)$和$\sigma_z^2(\boldsymbol{x},\phi)$到$\boldsymbol{z}$的随机连接并不能采用 BP 算法来得到模型参数。为了解决该问题，我们采用一个计算技巧，称为重新参数化 Reparameterization.

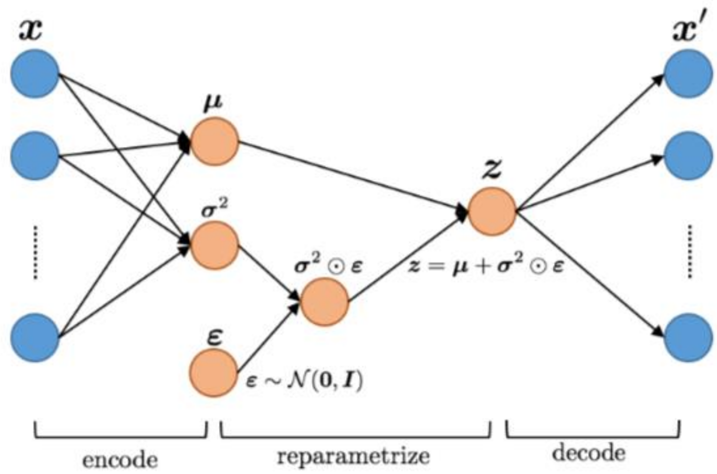
考虑识别模型：

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\mu_z(\boldsymbol{x},\phi),\sigma_z^2(\boldsymbol{x},\phi)\boldsymbol{I})$$

重新参数化后得到以下等式：

$$\boldsymbol{z} = \mu_z(\boldsymbol{x},\phi) + \sigma_z^2(\boldsymbol{x},\phi) \odot \epsilon,\ \ \epsilon \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{I})$$

其中$\odot$代表点积。现在$\boldsymbol{z}$取决于$\mu_z,\sigma_z^2,\epsilon$， 其中$\epsilon$是网络输入的一个随机变量。



现在重构误差可以写作：

$$\mathcal{L}_1 = E_{\boldsymbol{z}\sim q_\phi}\left[\log p_\theta\left(\boldsymbol{x}^{(i)}|\boldsymbol{z}(\boldsymbol{x}^{(i)},q_\phi,\epsilon)\right)\right] \approx \frac{1}{L}\sum_{l=1}^{L}\log(p_\theta\left(\boldsymbol{x}^{(i)}|\boldsymbol{z}^{(i,l)}\right))$$

其中$\boldsymbol{z}^{(i,l)} = \mu_z\left(\boldsymbol{x}^{(i)},\phi\right) + \sigma_z^2\left(\boldsymbol{x}^{(i)},\phi\right) \odot \epsilon^{(l)},\ \ \epsilon^{(l)} \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{I})$

此时梯度$\nabla_{\theta,\phi}\mathcal{L}_1$可以计算，因为给定任意$\epsilon$，网络是确定的；从而可以根据梯度上升解出$\mathcal{L}_1$的最大值。

再看$\mathcal{L}_2$：

$$\mathcal{L}_2 = -\mathcal{D}_{KL}[q_\phi\left(\boldsymbol{z}|\boldsymbol{x}^{(i)}\right)||p_\theta(\boldsymbol{z})]$$

因为两个分布均为高斯分布，所以有：

$$\mathcal{L}_2 = \frac{1}{2}\sum_{j=1}^{J}(1 + \log[\left(\sigma_j^{(i)}\right)^2] - \left(\mu_j^{(i)}\right)^2 - \left(\sigma_j^{(i)}\right)^2)$$

其中J为$\boldsymbol{z}$的维度，$\sigma_j^{(i)}$是$\sigma_z\left(\boldsymbol{x}^{(i)},\phi\right)$的第 j 项，$\mu_j^{(i)}$是$\mu_z\left(\boldsymbol{x}^{(i)},\phi\right)$的第 j 项，从而$\nabla_\phi\mathcal{L}_2$可以直接通过偏导计算。

进一步，将刚刚的分析进行合并，得到新的$\mathcal{L}\left(\boldsymbol{x}^{(i)},\boldsymbol{\theta},\boldsymbol{\phi}\right)$，并使用梯度上升对其求最大值：

$$\mathcal{L}\left(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}, \boldsymbol{\phi}\right) = \mathcal{L}_1 + \mathcal{L}_2$$

$$\approx \frac{1}{L}\sum_{l=1}^{L}\log(p_\theta\left(\boldsymbol{x}^{(i)}\middle|\boldsymbol{z}^{(i,l)}\right)) + \frac{1}{2}\sum_{j=1}^{J}(1 + \log[\left(\sigma_j^{(i)}\right)^2] - \left(\mu_j^{(i)}\right)^2 - \left(\sigma_j^{(i)}\right)^2)$$

与之前提到的朴素蒙特卡洛方法

$$\log p_\theta\left(\boldsymbol{x}^{(i)}\right) \approx \log\frac{1}{L}\sum_{l=1}^{L}p_\theta\left(\boldsymbol{x}^{(i)}\middle|\boldsymbol{z}^{(l)}\right)$$

相比，使用识别模型的新的估计

$$\log p_\theta\left(\boldsymbol{x}^{(i)}\right) \geq \mathcal{L}\left(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}, \boldsymbol{\phi}\right)$$

$$\approx \frac{1}{L}\sum_{l=1}^{L}\log(p_\theta\left(\boldsymbol{x}^{(i)}\middle|\boldsymbol{z}^{(i,l)}\right)) + \frac{1}{2}\sum_{j=1}^{J}(1 + \log[\left(\sigma_j^{(i)}\right)^2] - \left(\mu_j^{(i)}\right)^2 - \left(\sigma_j^{(i)}\right)^2)$$

更加贴近$\log p_\theta\left(\boldsymbol{x}^{(i)}\right)$。

生成样例

根据以上的公式

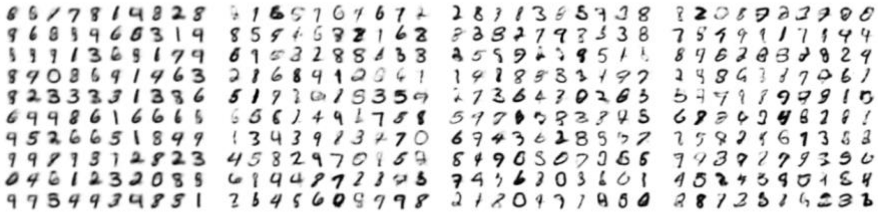
$$\mathcal{L} \approx \frac{1}{L}\sum_{l=1}^{L}\log(p_\theta\left(\boldsymbol{x}^{(i)}\middle|\boldsymbol{z}^{(i,l)}\right)) + \frac{1}{2}\sum_{j=1}^{J}(1 + \log[\left(\sigma_j^{(i)}\right)^2] - \left(\mu_j^{(i)}\right)^2 - \left(\sigma_j^{(i)}\right)^2)$$

利用梯度上升方法计算出其最大值以及对应的$\boldsymbol{\theta}, \boldsymbol{\phi}$。学习过程当中编码器和解码器是同步训练的。训练完成后，我们根据解码器，即生成模型来生成样例：

$$\boldsymbol{z} \sim p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

$$\boldsymbol{x} \sim p_\theta(\boldsymbol{x}|\boldsymbol{z})$$

生成样例的一个方法是在$\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$当中为$\boldsymbol{z}$采样，输入解码器，解码器通过训练好的网络得到$\mu_x$和$\sigma_x$，进而得到一个概率分布$p_\theta(\boldsymbol{x}|\boldsymbol{z})$，之后在$p_\theta(\boldsymbol{x}|\boldsymbol{z})$当中为$\boldsymbol{x}$采样，可以得到一些$\boldsymbol{x}$。以下是针对 MNIST 数据集进行操作后的结果，k-D latent space 当中的 k 代表$\boldsymbol{z}$的维数。



(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

一些讨论

VAE 是一种结合深度学习（用到了神经网络）和概率方法的机器学习算法，可以用于解决复杂高维数据的概率模型。从具体功能上来说：

1. VAE 中的解码器/生成模型可以用于生成相似但不同于原训练集的图片；
2. 解码器/生成模型同时给出了一个概率分布$p(x) = \int p(x|z)p(z)dz$，可以用于近似求变分下确界；
3. 编码器/识别模型可以用于提取高维数据的低维特征。

自动编码器 Autoencoders：

于变分自动编码器有些类似，不过它仅仅最小化重构误差来得到高维数据的低维特征，并没有正则项；同时因为其没有定义一个数据空间上的概率分布模型，所以不能生成新的样例。