第 13 章　深度生成模型

> 我不能创造的东西，我就不了解.
>
> ——理查德·菲利普斯·费曼（Richard Phillips Feynman）
> 1965 年诺贝尔物理奖获得者

概率生成模型（Probabilistic Generative Model），简称生成模型，是概率统计和机器学习领域的一类重要模型，指一系列用于随机生成可观测数据的模型. 假设在一个连续或离散的高维空间 $\mathcal{X}$ 中，存在一个随机向量 $\boldsymbol{X}$ 服从一个未知的数据分布 $p_r(\boldsymbol{x}), \boldsymbol{x} \in \mathcal{X}$. 生成模型是根据一些可观测的样本 $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}$ 来学习一个参数化的模型 $p_\theta(\boldsymbol{x})$ 来近似未知分布 $p_r(\boldsymbol{x})$，并可以用这个模型来生成一些样本，使得“生成”的样本和“真实”的样本尽可能地相似. 生成模型通常包含两个基本功能：概率密度估计和生成样本（即采样）. 图13.1以手写体数字图像为例给出了生成模型的两个功能示例，其中左图表示手写体数字图像的真实分布 $p_r(\boldsymbol{x})$ 以及从中采样的一些“真实”样本，右图表示估计出了分布 $p_\theta(\boldsymbol{x})$ 以及从中采样的“生成”样本.

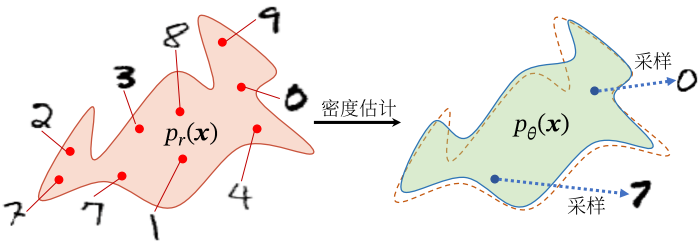


图 13.1　生成模型的两个功能

生成模型的应用十分广泛，可以用来建模不同的数据，比如图像、文本、声音等. 但对于一个高维空间中的复杂分布，密度估计和生成样本通常都不容易实现. 一是高维随机向量一般比较难以直接建模，需要通过一些条件独立性来简化模型，二是给定一个已建模的复杂分布，也缺乏有效的采样方法.

深度生成模型就是利用深度神经网络可以近似任意函数的能力来建模一个复杂分布 $p_r(\boldsymbol{x})$ 或直接生成符合分布 $p_r(\boldsymbol{x})$ 的样本. 本章先介绍概率生成模型的基本概念,然后介绍两种深度生成模型:变分自编码器和生成对抗网络.

13.1 概率生成模型

生成模型一般具有两个基本功能:密度估计和生成样本.

概率密度估计简称密度估计,参见第9.2节.

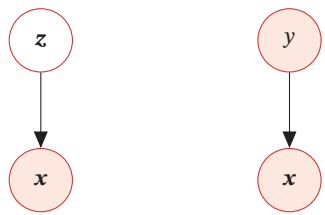
13.1.1 密度估计

给定一组数据 $\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$,假设它们都是独立地从相同的概率密度函数为 $p_r(\boldsymbol{x})$ 的未知分布中产生的. 密度估计(Density Estimation)是根据数据集 $\mathcal{D}$ 来估计其概率密度函数 $p_\theta(\boldsymbol{x})$.

在机器学习中,密度估计是一类无监督学习问题. 比如在手写体数字图像的密度估计问题中,我们将图像表示为一个随机向量 $\boldsymbol{X}$,其中每一维都表示一个像素值. 假设手写体数字图像都服从一个未知的分布 $p_r(\boldsymbol{x})$,希望通过一些观测样本来估计其分布. 但是,手写体数字图像中不同像素之间存在复杂的依赖关系(比如相邻像素的颜色一般是相似的),很难用一个明确的图模型来描述其依赖关系,所以直接建模 $p_r(\boldsymbol{x})$ 比较困难. 因此,我们通常通过引入隐变量 $\boldsymbol{z}$ 来简化模型,这样密度估计问题可以转换为估计变量 $(\boldsymbol{x}, \boldsymbol{z})$ 的两个局部条件概率 $p_\theta(\boldsymbol{z})$ 和 $p_\theta(\boldsymbol{x}|\boldsymbol{z})$. 一般为了简化模型,假设隐变量 $\boldsymbol{z}$ 的先验分布为标准高斯分布 $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. 隐变量 $\boldsymbol{z}$ 的每一维之间都是独立的. 在这个假设下,先验分布 $p(\boldsymbol{z};\theta)$ 中没有参数. 因此,密度估计的重点是估计条件分布 $p(\boldsymbol{x}|\boldsymbol{z};\theta)$.

如果要建模含隐变量的分布(如图13.2a),就需要利用 EM 算法来进行密度估计. 而在 EM 算法中,需要估计条件分布 $p(\boldsymbol{x}|\boldsymbol{z};\theta)$ 以及近似后验分布 $p(\boldsymbol{z}|\boldsymbol{x};\theta)$. 当这两个分布比较复杂时,我们可以利用神经网络来进行建模,这就是变分自编码器的思想.

EM 算法参见第11.2.2.1节.



(a) 含隐变量的生成模型 (b) 带标签的生成模型

图 13.2 生成模型

13.1.2 生成样本

生成样本就是给定一个概率密度函数为 $p_\theta(\boldsymbol{x})$ 的分布，生成一些服从这个分布的样本，也称为采样. 我们在第11.5节中介绍了一些常用的采样方法.

采样方法参见第11.5节.

对于图13.2a中的图模型，在得到两个变量的局部条件概率 $p_\theta(\boldsymbol{z})$ 和 $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ 之后，我们就可以生成数据 $\boldsymbol{x}$，具体过程可以分为两步进行：

（1） 根据隐变量的先验分布 $p_\theta(\boldsymbol{z})$ 进行采样，得到样本 $\boldsymbol{z}$.

（2） 根据条件分布 $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ 进行采样，得到样本 $\boldsymbol{x}$.

为了便于采样，通常 $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ 不能太过复杂. 因此，另一种生成样本的思想是从一个简单分布 $p(\boldsymbol{z}), \boldsymbol{z} \in \mathcal{Z}$（比如标准正态分布）中采集一个样本 $\boldsymbol{z}$，并利用一个深度神经网络 $g: \mathcal{Z} \to \mathcal{X}$ 使得 $g(\boldsymbol{z})$ 服从 $p_r(\boldsymbol{x})$. 这样，我们就可以避免密度估计问题，并有效降低生成样本的难度，这正是生成对抗网络的思想.

13.1.3 应用于监督学习

除了生成样本外，生成模型也可以应用于监督学习. 监督学习的目标是建模样本 $\boldsymbol{x}$ 和输出标签 y 之间的条件概率分布 $p(y|\boldsymbol{x})$. 根据贝叶斯公式，

$$p(y|\boldsymbol{x}) = \frac{p(\boldsymbol{x}, y)}{\sum_y p(\boldsymbol{x}, y)}. \tag{13.1}$$

我们可以将监督学习问题转换为联合概率分布 $p(\boldsymbol{x}, y)$ 的密度估计问题.

图13.2b给出了带标签的生成模型的图模型表示，可以用于监督学习. 在监督学习中，比较典型的生成模型有朴素贝叶斯分类器、隐马尔可夫模型.

参见第11.1.2节.

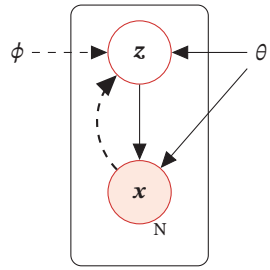
判别模型 和生成模型相对应的另一类监督学习模型是判别模型（Discriminative Model）. 判别模型直接建模条件概率分布 $p(y|\boldsymbol{x})$，并不建模其联合概率分布 $p(\boldsymbol{x}, y)$. 常见的判别模型有Logistic 回归、支持向量机、神经网络等. 由生成模型可以得到判别模型，但由判别模型得不到生成模型.

13.2 变分自编码器

13.2.1 含隐变量的生成模型

假设一个生成模型（如图13.3所示）中包含隐变量，即有部分变量是不可观测的，其中观测变量 $\boldsymbol{X}$ 是一个高维空间 $\mathcal{X}$ 中的随机向量，隐变量 $\boldsymbol{Z}$ 是一个相对低维的空间 $\mathcal{Z}$ 中的随机向量.

本章中，我们假设 $\boldsymbol{X}$ 和 $\boldsymbol{Z}$ 都是连续随机向量.



实线表示生成模型，虚线表示变分近似.

图 13.3 变分自编码器

这个生成模型的联合概率密度函数可以分解为

$$p(\boldsymbol{x}, \boldsymbol{z}; \theta) = p(\boldsymbol{x}|\boldsymbol{z}; \theta)p(\boldsymbol{z}; \theta), \tag{13.2}$$

其中 $p(\boldsymbol{z}; \theta)$ 为隐变量 $\boldsymbol{z}$ 先验分布的概率密度函数，$p(\boldsymbol{x}|\boldsymbol{z}; \theta)$ 为已知 $\boldsymbol{z}$ 时观测变量 $\boldsymbol{x}$ 的条件概率密度函数，θ 表示两个密度函数的参数. 一般情况下，我们可以假设 $p(\boldsymbol{z}; \theta)$ 和 $p(\boldsymbol{x}|\boldsymbol{z}; \theta)$ 为某种参数化的分布族，比如正态分布. 这些分布的形式已知，只是参数 θ 未知，可以通过最大化似然来进行估计.

给定一个样本 $\boldsymbol{x}$，其对数边际似然 $\log p(\boldsymbol{x}; \theta)$ 可以分解为

$$\log p(\boldsymbol{x}; \theta) = ELBO(q, \boldsymbol{x}; \theta, \phi) + \mathrm{KL}(q(\boldsymbol{z}; \phi), p(\boldsymbol{z}|\boldsymbol{x}; \theta)), \tag{13.3}$$

其中 $q(\boldsymbol{z}; \phi)$ 是额外引入的变分密度函数，其参数为 ϕ，$ELBO(q, \boldsymbol{x}; \theta, \phi)$ 为证据下界，

参见公式(11.49).

$$ELBO(q, \boldsymbol{x}; \theta, \phi) = \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z};\phi)}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z}; \theta)}{q(\boldsymbol{z}; \phi)}\right]. \tag{13.4}$$

最大化对数边际似然 $\log p(\boldsymbol{x}; \theta)$ 可以用 EM 算法来求解. 在 EM 算法的每次迭代中，具体可以分为两步：

EM 算法参见第11.2.2.1节.

（1） E 步：固定 θ，寻找一个密度函数 $q(\boldsymbol{z}; \phi)$ 使其等于或接近于后验密度函数 $p(\boldsymbol{z}|\boldsymbol{x}; \theta)$；

（2） M 步：固定 $q(\boldsymbol{z}; \phi)$，寻找 θ 来最大化 $ELBO(q, \boldsymbol{x}; \theta, \phi)$.

不断重复上述两步骤，直到收敛.

在 EM 算法的每次迭代中，理论上最优的 $q(\boldsymbol{z}; \phi)$ 为隐变量的后验概率密度函数 $p(\boldsymbol{z}|\boldsymbol{x}; \theta)$，即

$$p(\boldsymbol{z}|\boldsymbol{x}; \theta) = \frac{p(\boldsymbol{x}|\boldsymbol{z}; \theta)p(\boldsymbol{z}; \theta)}{\int_{\boldsymbol{z}} p(\boldsymbol{x}|\boldsymbol{z}; \theta)p(\boldsymbol{z}; \theta)d\boldsymbol{z}}. \tag{13.5}$$

后验概率密度函数 $p(\boldsymbol{z}|\boldsymbol{x}; \theta)$ 的计算是一个统计推断问题，涉及积分计算. 当隐变量 $\boldsymbol{z}$ 是有限的一维离散变量时，计算起来比较容易. 但在一般情况下，这个后验概

率密度函数是很难计算的，通常需要通过变分推断来近似估计. 在变分推断中，为了降低复杂度，通常会选择一些比较简单的分布 $q(\boldsymbol{z};\boldsymbol{\phi})$ 来近似推断 $p(\boldsymbol{z}|\boldsymbol{x};\theta)$. 当 $p(\boldsymbol{z}|\boldsymbol{x};\theta)$ 比较复杂时，近似效果不佳. 此外，概率密度函数 $p(\boldsymbol{x}|\boldsymbol{z};\theta)$ 一般也比较复杂，很难直接用已知的分布族函数进行建模.

变分推断参见第11.4节.

变分自编码器（Variational AutoEncoder，VAE）[Kingma et al., 2014] 是一种深度生成模型，其思想是利用神经网络来分别建模两个复杂的条件概率密度函数.

（1） 用神经网络来估计变分分布 $q(\boldsymbol{z};\boldsymbol{\phi})$，称为推断网络. 理论上 $q(\boldsymbol{z};\boldsymbol{\phi})$ 可以不依赖 $\boldsymbol{x}$. 但由于 $q(\boldsymbol{z};\boldsymbol{\phi})$ 的目标是近似后验分布 $p(\boldsymbol{z}|\boldsymbol{x};\theta)$，其和 $\boldsymbol{x}$ 相关，因此变分密度函数一般写为 $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$. 推断网络的输入为 $\boldsymbol{x}$，输出为变分分布 $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$.

（2） 用神经网络来估计概率分布 $p(\boldsymbol{x}|\boldsymbol{z};\theta)$，称为生成网络. 生成网络的输入为 $\boldsymbol{z}$，输出为概率分布 $p(\boldsymbol{x}|\boldsymbol{z};\theta)$.

将推断网络和生成网络合并就得到了变分自编码器的整个网络结构，如图13.4所示，其中实线表示网络计算操作，虚线表示采样操作.

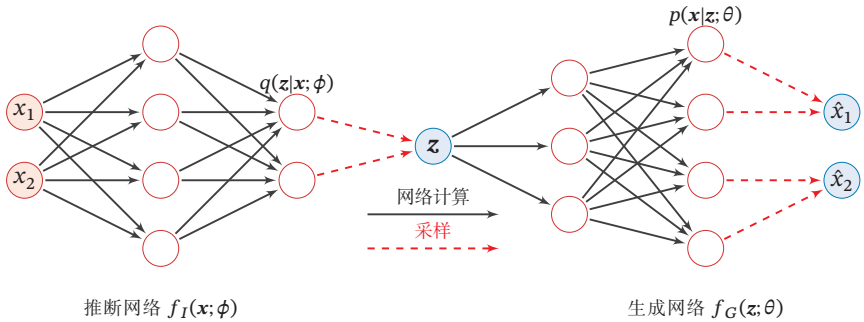


图 13.4 变分自编码器的网络结构

变分自编码器的名称来自于其整个网络结构和自编码器比较类似. 我们可以把推断网络看作“编码器”，将可观测变量映射为隐变量；把生成网络看作“解码器”，将隐变量映射为可观测变量. 然而，变分自编码器背后的原理和自编码器完全不同. 变分自编码器中的编码器和解码器的输出为分布（或分布的参数），而不是确定的编码.

自编码器参见第9.1.3节.

13.2.2 推断网络

为简单起见，假设 $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$ 是服从对角化协方差的高斯分布，

$$q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi}) = \mathcal{N}(\boldsymbol{z};\boldsymbol{\mu}_I, \boldsymbol{\sigma}_I^2\boldsymbol{I}), \tag{13.6}$$

其中 $\boldsymbol{\mu}_I$ 和 $\boldsymbol{\sigma}_I^2$ 是高斯分布的均值和方差，可以通过推断网络 $f_I(\boldsymbol{x};\boldsymbol{\phi})$ 来预测.

$$\begin{bmatrix}\boldsymbol{\mu}_I\\ \boldsymbol{\sigma}_I^2\end{bmatrix} = f_I(\boldsymbol{x};\boldsymbol{\phi}), \tag{13.7}$$

其中推断网络 $f_I(\boldsymbol{x};\boldsymbol{\phi})$ 可以是一般的全连接网络或卷积网络，比如一个两层的神经网络，

$$\boldsymbol{h} = \sigma\big(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\big), \tag{13.8}$$

$$\boldsymbol{\mu}_I = \boldsymbol{W}^{(2)}\boldsymbol{h} + \boldsymbol{b}^{(2)}, \tag{13.9}$$

$$\boldsymbol{\sigma}_I^2 = \text{softplus}\big(\boldsymbol{W}^{(3)}\boldsymbol{h} + \boldsymbol{b}^{(3)}\big), \tag{13.10}$$

$\text{softplus}(x) = \log(1 + e^x)$.

其中 $\boldsymbol{\phi}$ 代表所有的网络参数 $\{\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{W}^{(3)}, \boldsymbol{b}^{(1)}, \boldsymbol{b}^{(2)}, \boldsymbol{b}^{(3)}\}$，$\sigma$ 和 softplus 为激活函数. 这里使用 softplus 激活函数是由于方差总是非负的. 在实际实现中，也可以用一个线性层（不需要激活函数）来预测 $\log(\boldsymbol{\sigma}_I^2)$.

推断网络的目标 推断网络的目标是使得 $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$ 尽可能接近真实的后验 $p(\boldsymbol{z}|\boldsymbol{x};\theta)$，需要找到一组网络参数 $\boldsymbol{\phi}^*$ 来最小化两个分布的 KL 散度，即

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \text{KL}\Big(q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi}), p(\boldsymbol{z}|\boldsymbol{x};\theta)\Big). \tag{13.11}$$

然而，直接计算上面的 KL 散度是不可能的，因为 $p(\boldsymbol{z}|\boldsymbol{x};\theta)$ 一般无法计算. 传统方法是利用采样或者变分法来近似推断. 基于采样的方法效率很低且估计也不是很准确，所以一般使用的是变分推断方法，即用简单的分布 q 去近似复杂的分布 $p(\boldsymbol{z}|\boldsymbol{x};\theta)$. 但是，在深度生成模型中，$p(\boldsymbol{z}|\boldsymbol{x};\theta)$ 通常比较复杂，很难用简单分布去近似. 因此，我们需要找到一种间接计算方法.

变分推断参见第11.4节.

根据公式(13.3)可知，变分分布 $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$ 与真实后验 $p(\boldsymbol{z}|\boldsymbol{x};\theta)$ 的 KL 散度等于对数边际似然 $\log p(\boldsymbol{x};\theta)$ 与其下界 $ELBO(q,\boldsymbol{x};\theta,\boldsymbol{\phi})$ 的差，即

$$\text{KL}(q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi}), p(\boldsymbol{z}|\boldsymbol{x};\theta)) = \log p(\boldsymbol{x};\theta) - ELBO(q,\boldsymbol{x};\theta,\boldsymbol{\phi}), \tag{13.12}$$

因此，推断网络的目标函数可以转换为

可以看作EM算法中的E步.

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \text{KL}\Big(q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi}), p(\boldsymbol{z}|\boldsymbol{x};\theta)\Big) \tag{13.13}$$

$$= \arg\min_{\boldsymbol{\phi}} \log p(\boldsymbol{x};\theta) - ELBO(q,\boldsymbol{x};\theta,\boldsymbol{\phi}) \tag{13.14}$$

第一项与 $\boldsymbol{\phi}$ 无关.

$$= \arg\max_{\boldsymbol{\phi}} ELBO(q,\boldsymbol{x};\theta,\boldsymbol{\phi}), \tag{13.15}$$

即推断网络的目标转换为寻找一组网络参数 $\boldsymbol{\phi}^*$ 使得证据下界 $ELBO(q,\boldsymbol{x};\theta,\boldsymbol{\phi})$ 最大，这和变分推断中的转换类似.

参见公式(11.85).

13.2.3 生成网络

生成模型的联合分布 $p(\boldsymbol{x}, \boldsymbol{z}; \theta)$ 可以分解为两部分：隐变量 $\boldsymbol{z}$ 的先验分布 $p(\boldsymbol{z}; \theta)$ 和条件概率分布 $p(\boldsymbol{x}|\boldsymbol{z}; \theta)$.

先验分布 $p(\boldsymbol{z}; \theta)$ 为简单起见，我们一般假设隐变量 $\boldsymbol{z}$ 的先验分布为各向同性的标准高斯分布 $\mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$. 隐变量 $\boldsymbol{z}$ 的每一维之间都是独立的.

条件概率分布 $p(\boldsymbol{x}|\boldsymbol{z}; \theta)$ 条件概率分布 $p(\boldsymbol{x}|\boldsymbol{z}; \theta)$ 可以通过生成网络来建模. 为简单起见，我们同样用参数化的分布族来表示条件概率分布 $p(\boldsymbol{x}|\boldsymbol{z}; \theta)$，这些分布族的参数可以用生成网络计算得到.

根据变量 $\boldsymbol{x}$ 的类型不同，可以假设 $p(\boldsymbol{x}|\boldsymbol{z}; \theta)$ 服从不同的分布族.

（1）如果 $\boldsymbol{x} \in \{0, 1\}^D$ 是 D 维的二值的向量，可以假设 $p(\boldsymbol{x}|\boldsymbol{z}; \theta)$ 服从多变量的伯努利分布，即

$$p(\boldsymbol{x}|\boldsymbol{z}; \theta) = \prod_{d=1}^{D} p(x_d|\boldsymbol{z}; \theta) \tag{13.16}$$

$$= \prod_{d=1}^{D} \gamma_d^{x_d} (1 - \gamma_d)^{(1-x_d)}, \tag{13.17}$$

其中 $\gamma_d \triangleq p(x_d = 1|\boldsymbol{z}; \theta)$ 为第 d 维分布的参数. 分布的参数 $\boldsymbol{\gamma} = [\gamma_1, \cdots, \gamma_D]^{\mathrm{T}}$ 可以通过生成网络来预测.

（2）如果 $\boldsymbol{x} \in \mathbb{R}^D$ 是 D 维的连续向量，可以假设 $p(\boldsymbol{x}|\boldsymbol{z}; \theta)$ 服从对角化协方差的高斯分布，即

$$p(\boldsymbol{x}|\boldsymbol{z}; \theta) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_G, \boldsymbol{\sigma}_G^2 \boldsymbol{I}), \tag{13.18}$$

其中 $\boldsymbol{\mu}_G \in \mathbb{R}^D$ 和 $\boldsymbol{\sigma}_G \in \mathbb{R}^D$ 同样可以用生成网络 $f_G(\boldsymbol{z}; \theta)$ 来预测.

生成网络的目标 生成网络 $f_G(\boldsymbol{z}; \theta)$ 的目标是找到一组网络参数 θ^* 来最大化证据下界 $ELBO(q, \boldsymbol{x}; \theta, \phi)$，即

可以看作EM算法中的M步.

$$\theta^* = \arg\max_{\theta} ELBO(q, \boldsymbol{x}; \theta, \phi). \tag{13.19}$$

13.2.4 模型汇总

结合公式(13.15)和公式(13.19)，推断网络和生成网络的目标都为最大化证据下界 $ELBO(q, \boldsymbol{x}; \theta, \phi)$. 因此，变分自编码器的总目标函数为

$$\max_{\theta, \phi} ELBO(q, \boldsymbol{x}; \theta, \phi) = \max_{\theta, \phi} \mathbb{E}_{\boldsymbol{z} \sim q(\boldsymbol{z}; \phi)} \left[\log \frac{p(\boldsymbol{x}|\boldsymbol{z}; \theta) p(\boldsymbol{z}; \theta)}{q(\boldsymbol{z}; \phi)} \right] \tag{13.20}$$

$$= \max_{\theta,\phi} \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x};\phi)}\Big[\log p(\boldsymbol{x}|\boldsymbol{z};\theta)\Big] - \mathrm{KL}\Big(q(\boldsymbol{z}|\boldsymbol{x};\phi), p(\boldsymbol{z};\theta)\Big), \tag{13.21}$$

其中 $p(\boldsymbol{z};\theta)$ 为先验分布，θ 和 ϕ 分别表示生成网络和推断网络的参数.

从 EM 算法角度来看，变分自编码器优化推断网络和生成网络的过程，可以分别看作 EM 算法中的 E 步和 M 步. 但在变分自编码器中，这两步的目标合二为一，都是最大化证据下界. 此外，变分自编码器可以看作神经网络和贝叶斯网络的混合体. 贝叶斯网络中的所有节点都是随机变量. 在变分自编码器中，我们仅仅将隐藏编码对应的节点看成是随机变量，其他节点还是作为普通神经元. 这样，编码器变成一个变分推断网络，而解码器变成一个将隐变量映射到观测变量的生成网络.

我们分别来看公式(13.21)中的两项.

（1）通常情况下，公式(13.21)中第一项的期望 $\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x};\phi)}[\log p(\boldsymbol{x}|\boldsymbol{z};\theta)]$ 可以通过采样的方式近似计算. 对于每个样本 $\boldsymbol{x}$，根据 $q(\boldsymbol{z}|\boldsymbol{x};\phi)$ 采集 M 个 $\boldsymbol{z}^{(m)}$，$1 \leq m \leq M$，有

$$\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x};\phi)}[\log p(\boldsymbol{x}|\boldsymbol{z};\theta)] \approx \frac{1}{M}\sum_{m=1}^{M} \log p(\boldsymbol{x}|\boldsymbol{z}^{(m)};\theta). \tag{13.22}$$

期望 $\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x};\phi)}[\log p(\boldsymbol{x}|\boldsymbol{z};\theta)]$ 依赖于参数 ϕ. 但在上面的近似中，这个期望变得和参数 ϕ 无关. 当使用梯度下降法来学习参数时，期望 $\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x};\phi)}[\log p(\boldsymbol{x}|\boldsymbol{z};\theta)]$ 关于参数 ϕ 的梯度为 0. 这种情况是由于变量 $\boldsymbol{z}$ 和参数 ϕ 之间不是直接的确定性关系，而是一种“采样”关系. 这种情况可以通过两种方法解决：一种是再参数化，我们在下一节具体介绍；另一种是梯度估计的方法，具体参考第14.3节.

（2）公式(13.21)中第二项的 KL 散度通常可以直接计算. 特别是当 $q(\boldsymbol{z}|\boldsymbol{x};\phi)$ 和 $p(\boldsymbol{z};\theta)$ 都是正态分布时，它们的 KL 散度可以直接计算出闭式解.

给定 D 维空间中的两个正态分布 $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ 和 $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$，其 KL 散度为

$$\begin{aligned}&\mathrm{KL}\Big(\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)\Big)\\ &= \frac{1}{2}\Big(\mathrm{tr}(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1) + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) - D + \log\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}\Big),\end{aligned} \tag{13.23}$$

其中 $\mathrm{tr}(\cdot)$ 表示矩阵的迹，$|\cdot|$ 表示矩阵的行列式.

矩阵的“迹”为主对角线（从左上方至右下方的对角线）上各个元素的总和.

这样，当 $p(\boldsymbol{z};\theta) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I})$ 以及 $q(\boldsymbol{z}|\boldsymbol{x};\phi) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_I, \sigma_I^2\boldsymbol{I})$ 时，

$$\begin{aligned}&\mathrm{KL}\Big(q(\boldsymbol{z}|\boldsymbol{x};\phi), p(\boldsymbol{z};\theta)\Big)\\ &= \frac{1}{2}\Big(\mathrm{tr}(\sigma_I^2\boldsymbol{I}) + \boldsymbol{\mu}_I^{\mathrm{T}}\boldsymbol{\mu}_I - d - \log(|\sigma_I^2\boldsymbol{I}|)\Big),\end{aligned} \tag{13.24}$$

其中 $\boldsymbol{\mu}_I$ 和 σ_I 为推断网络 $f_I(\boldsymbol{x};\phi)$ 的输出.

13.2.5 再参数化

再参数化（Reparameterization）是将一个函数 $f(\theta)$ 的参数 θ 用另外一组参数表示 $\theta = g(\vartheta)$，这样函数 $f(\theta)$ 就转换成参数为 ϑ 的函数 $\hat{f}(\vartheta) = f\big(g(\vartheta)\big)$. 再参数化通常用来将原始参数转换为另外一组具有特殊属性的参数. 比如当 θ 为一个很大的矩阵时，可以使用两个低秩矩阵的乘积来再参数化，从而减少参数量.

再参数化的另一个例子是逐层归一化，参见第7.5节.

在公式(13.21)中，期望 $\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})}\Big[\log p(\boldsymbol{x}|\boldsymbol{z};\theta)\Big]$ 依赖于分布 q 的参数 $\boldsymbol{\phi}$. 但是，由于随机变量 $\boldsymbol{z}$ 采样自后验分布 $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$，它们之间不是确定性关系，因此无法直接求解 $\boldsymbol{z}$ 关于参数 $\boldsymbol{\phi}$ 的导数. 这时，我们可以通过再参数化方法来将 $\boldsymbol{z}$ 和 $\boldsymbol{\phi}$ 之间随机性的采样关系转变为确定性函数关系.

我们引入一个分布为 $p(\epsilon)$ 的随机变量 ϵ，期望 $\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})}\Big[\log p(\boldsymbol{x}|\boldsymbol{z};\theta)\Big]$ 可以重写为

$$\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})}\Big[\log p(\boldsymbol{x}|\boldsymbol{z};\theta)\Big] = \mathbb{E}_{\epsilon\sim p(\epsilon)}\Big[\log p(\boldsymbol{x}|g(\boldsymbol{\phi},\epsilon);\theta)\Big], \tag{13.25}$$

其中 $\boldsymbol{z} \triangleq g(\boldsymbol{\phi},\epsilon)$ 为一个确定性函数.

假设 $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$ 为正态分布 $N(\boldsymbol{\mu}_I, \boldsymbol{\sigma}_I^2\boldsymbol{I})$，其中 $\{\boldsymbol{\mu}_I, \sigma_I\}$ 是推断网络 $f_I(\boldsymbol{x};\boldsymbol{\phi})$ 的输出，依赖于参数 $\boldsymbol{\phi}$，我们可以通过下面方式来再参数化：

$$\boldsymbol{z} = \boldsymbol{\mu}_I + \boldsymbol{\sigma}_I \odot \epsilon, \tag{13.26}$$

参见习题13-1.

其中 $\epsilon \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{I})$. 这样 $\boldsymbol{z}$ 和参数 $\boldsymbol{\phi}$ 的关系从采样关系变为确定性关系，使得 $\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$ 的随机性独立于参数 $\boldsymbol{\phi}$，从而可以求 $\boldsymbol{z}$ 关于 $\boldsymbol{\phi}$ 的导数.

13.2.6 训练

通过再参数化，变分自编码器可以通过梯度下降法来学习参数，从而提高变分自编码器的训练效率.

给定一个数据集 $\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^N$，对于每个样本 $\boldsymbol{x}^{(n)}$，随机采样 M 个变量 $\epsilon^{(n,m)}, 1 \leq m \leq M$，并通过公式(13.26)计算 $\boldsymbol{z}^{(n,m)}$. 变分自编码器的目标函数近似为

$$\mathcal{J}(\boldsymbol{\phi},\theta|\mathcal{D}) = \sum_{n=1}^{N}\left(\frac{1}{M}\sum_{m=1}^{M}\log p(\boldsymbol{x}^{(n)}|\boldsymbol{z}^{(n,m)};\theta) - \mathrm{KL}\Big(q(\boldsymbol{z}|\boldsymbol{x}^{(n)};\boldsymbol{\phi}),\mathcal{N}(\boldsymbol{z};\boldsymbol{0},\boldsymbol{I})\Big)\right). \tag{13.27}$$

如果采用随机梯度方法，每次从数据集中采集一个样本 $\boldsymbol{x}$ 和一个对应的随机变量 ϵ，并进一步假设 $p(\boldsymbol{x}|\boldsymbol{z};\theta)$ 服从高斯分布 $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_G,\lambda\boldsymbol{I})$，其中 $\boldsymbol{\mu}_G = f_G(\boldsymbol{z};\theta)$ 是生成网络的输出，λ 为控制方差的超参数，则目标函数可以简化为

$$\mathcal{J}(\boldsymbol{\phi},\theta|\boldsymbol{x}) = -\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{\mu}_G\|^2 - \lambda\,\mathrm{KL}\Big(\mathcal{N}(\boldsymbol{\mu}_I,\boldsymbol{\sigma}_I),\mathcal{N}(\boldsymbol{0},\boldsymbol{I})\Big), \tag{13.28}$$

参见习题13-2.

其中第一项可以近似看作输入 $\boldsymbol{x}$ 的重构正确性，第二项可以看作正则化项，λ 可以看作正则化系数. 这和自编码器在形式上非常类似，但它们的内在机理是完全不同的.

参见习题13-3.

变分自编码器的训练过程如图13.5所示，其中空心矩形表示“目标函数”.

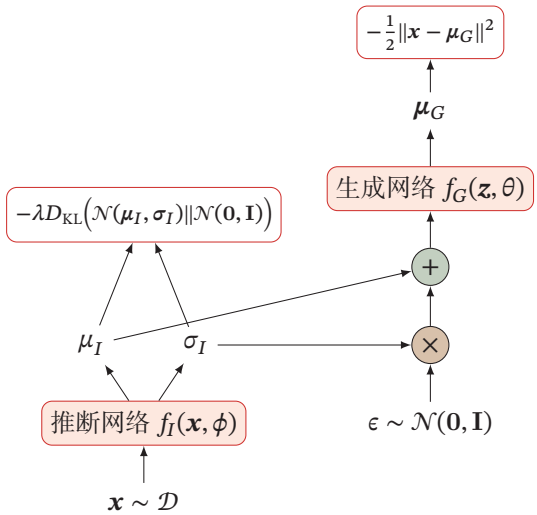
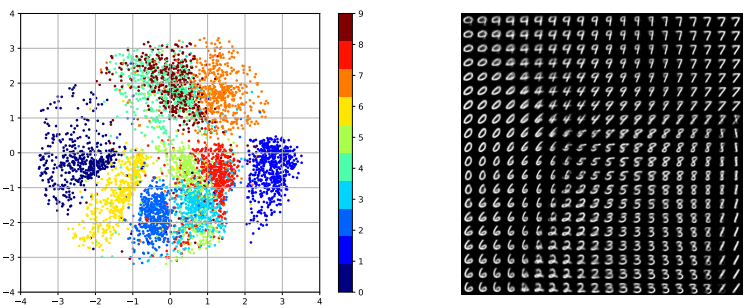


图 13.5 变分自编码器的训练过程

图13.6给出了在 MNIST 数据集上变分自编码器学习到的隐变量流形的可视化示例. 图13.6a是将训练集上每个样本 $\boldsymbol{x}$ 通过推断网络映射到 2 维的隐变量空间，图中的每个点表示 $\mathbb{E}[\boldsymbol{z}|\boldsymbol{x}]$，不同颜色表示不同的数字. 图13.6b是对 2 维的标准高斯分布上进行均匀采样得到不同的隐变量 $\boldsymbol{z}$，然后通过生成网络产生 $\mathbb{E}[\boldsymbol{x}|\boldsymbol{z}]$.



(a) 训练集上所有样本在隐空间上的投影 (b) 隐变量 $\boldsymbol{z}$ 在图像空间的投影

图 13.6 在 MNIST 数据集上变分自编码器学习到的隐变量流形的可视化示例