第3章

变分自编码器

在本章中，我们将介绍一种具有隐变量的生成模型——变分自编码器（Variational Autoencoder, VAE）。VAE 作为一种深度生成模型，其核心在于引入了隐变量的概念，使模型能够学习并捕捉到数据背后的潜在表示。与传统自编码器（Autoencoder，AE）相比，VAE 不仅致力于数据的重构，更侧重于通过编码过程将数据映射到低维隐空间，并通过解码过程重构原始数据。VAE 的优化过程确保了隐空间表示的连续性和完整性，具有重要的理论和应用价值。

本章的结构安排如下：首先，我们将复习一些与 VAE 相关的概率统计及机器学习知识，涵盖 MLE、隐变量模型、最大期望算法、AE 等内容。随后，我们将系统地阐述经典 VAE 的组成架构及其训练算法。由于 VAE 的提出时间较早，相关文献已经非常丰富。因此，本章将聚焦于经典 VAE 及一些著名的变体模型。最后，为了帮助读者更深入地理解 VAE 的实现步骤，我们将提供一个训练 VAE 的实例。需要注意的是，在本章中，如无特别说明，随机变量/向量均假设为连续型。

3.1 预备知识

3.1.1 AE

在介绍 VAE 之前，我们先来回顾一下 AE。AE 是一种无监督学习的神经网络模型，它通过编码器和解码器的组合，将原始数据映射到潜在空间表示，并尝试从该表示中重建原始数据。AE 通过限制信息在编码器和解码器之间的流动，迫使模型学习到原始数据的有效压缩表示（或称为编码特征）。这种表示通常具有比原始数据更低的维度或更稀疏的特性，但保留了足够的信息以重建原始数据。

如图 3-1 所示，AE 的结构分为**编码器**和**解码器**两部分。编码器负责将原始数据 $\boldsymbol{x}\in\mathbb{R}^D$ 映射到一个潜在空间的特征表示 $\boldsymbol{z}\in\mathbb{R}^M$。数学上，编码器可以表示为映射函数 $f:\mathbb{R}^D\to\mathbb{R}^M$，使 $\boldsymbol{z}=f(\boldsymbol{x})$。解码器负责将潜在空间的特征表示 $\boldsymbol{z}\in\mathbb{R}^M$ 映射回原始空间 $\hat{\boldsymbol{x}}\in\mathbb{R}^D$，以重建原始数据。数学上，解码器可以表示为映射函数 $g:\mathbb{R}^M\to\mathbb{R}^D$，使 $\hat{\boldsymbol{x}}=g(\boldsymbol{z})$。我们一般令 $M\ll D$，

或者对特征表示 $\boldsymbol{z}\in\mathbb{R}^M$ 加上稀疏性的限制，以使编码器能够学到有意义的原始数据的特征表示。

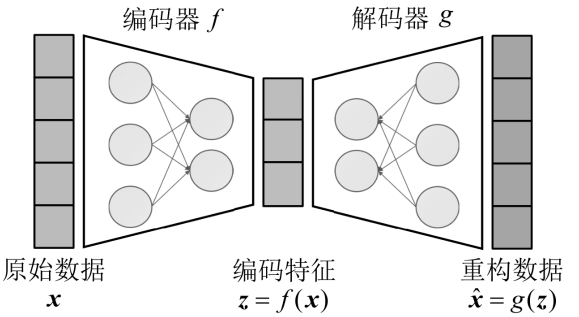


图 3-1　AE 结构示意图

编码器和解码器可以用神经网络进行建模。假设编码器的可学习参数为 $\boldsymbol{\varphi}$，则编码器可以记为 $f(\boldsymbol{x};\boldsymbol{\varphi})$。类似地，假设解码器的可学习参数为 $\boldsymbol{\theta}$，则解码器可以记为 $g(\boldsymbol{x};\boldsymbol{\theta})$。AE 的学习目标是最小化**重构误差**（Reconstruction Error）：

$$\mathcal{L}(\boldsymbol{\varphi},\boldsymbol{\theta})=E_{\boldsymbol{X}\sim p(\boldsymbol{x})}\left[\|\boldsymbol{X}-g(f(\boldsymbol{X};\boldsymbol{\varphi});\boldsymbol{\theta})\|_2^2\right] \tag{3-1}$$

式中，数据分布记为 $p(\boldsymbol{x})$；$\boldsymbol{X}$ 为服从 $p(\boldsymbol{x})$ 的随机样本。若从 $p(\boldsymbol{x})$ 中采样得到 $\boldsymbol{X}$ 的 N 个观测样本 $\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N\}$，则式（3-1）中的期望可以用样本均值来估计。

我们使用 AE，旨在获取有效的数据表示。训练完成后，通常会去除解码器，仅保留编码器，因为编码器的输出能够直接作为后续机器学习模型的输入。

3.1.2　KL 散度

KL 散度是衡量两个概率分布 $p(\boldsymbol{x})$ 和 $q(\boldsymbol{x})$ 差异的一种方法。在信息论中，KL 散度被用来衡量使用基于 $q(\boldsymbol{x})$ 的编码，编码来自 $p(\boldsymbol{x})$ 的样本所需的额外比特数。在机器学习中，KL 散度常被用作损失函数，特别是在 VAE 等生成模型中。

对于离散概率分布，KL 散度定义为

$$\mathrm{KL}(p(\boldsymbol{x})\|q(\boldsymbol{x}))=\sum_{\boldsymbol{x}}p(\boldsymbol{x})\ln\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \tag{3-2}$$

式中，$p(\boldsymbol{x})$ 和 $q(\boldsymbol{x})$ 为分布的概率质量函数。

对于连续概率分布，KL 散度定义为

$$\mathrm{KL}(p(\boldsymbol{x})\|q(\boldsymbol{x}))=\int p(\boldsymbol{x})\ln\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\mathrm{d}\boldsymbol{x} \tag{3-3}$$

式中，$p(\boldsymbol{x})$ 和 $q(\boldsymbol{x})$ 为分布的概率密度函数。

KL 散度具有非负性和不对称性。

（1）**非负性**：$\mathrm{KL}(p(\boldsymbol{x})\|q(\boldsymbol{x}))\geqslant 0$，等号当且仅当 $p(\boldsymbol{x})=q(\boldsymbol{x})$ 时成立。

（2）**不对称性**：$\mathrm{KL}(p(\boldsymbol{x})\|q(\boldsymbol{x}))\neq\mathrm{KL}(q(\boldsymbol{x})\|p(\boldsymbol{x}))$，即 KL 散度是不对称的。因此，KL 散度实际上并不是一个真正的度量或距离。

3.1.3　MLE

MLE 是统计学领域广泛应用的参数估计方法，其核心在于通过最大化观测数据在特定参数条件下出现的可能性（似然函数），来估算模型中的未知参数。MLE 基于一个基本假设：观测数据来自一个已知的概率分布，尽管该分布的具体参数尚未确定。其目标是确定一组参数，使在这组参数下，观测数据出现的可能性达到最大。简而言之，就是寻找能够最佳解释观测数据的参数。

假设 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_N$ 是采样自某个连续概率分布的 N 个独立同分布（Independent and Identically Distributed，IID）的随机样本，该分布的概率密度函数为 $p_{\boldsymbol{\theta}}(\boldsymbol{x})$，其中 $\boldsymbol{\theta}$ 为未知的待估参数。形式上，似然函数 $\mathcal{L}(\boldsymbol{\theta}\mid\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_N)$ 是随机样本 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_N$ 在给定参数 $\boldsymbol{\theta}$ 下的联合概率密度函数，即

$$\mathcal{L}(\boldsymbol{\theta}\mid\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_N)=\prod_{i=1}^{N}p_{\boldsymbol{\theta}}(\boldsymbol{X}_i) \tag{3-4}$$

MLE 的目标是找到一组参数 $\hat{\boldsymbol{\theta}}$，使 $\mathcal{L}(\boldsymbol{\theta}\mid\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_N)$ 最大化，即

$$\hat{\boldsymbol{\theta}}=\arg\max_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}\mid\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_N) \tag{3-5}$$

由于直接对连乘形式的似然函数求最大化比较困难，因此通常会对似然函数取以 e 为底数的对数，得到对数似然函数，即

$$l(\boldsymbol{\theta}\mid\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_N)=\ln\mathcal{L}(\boldsymbol{\theta}\mid\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_N)=\sum_{i=1}^{N}\ln p_{\boldsymbol{\theta}}(\boldsymbol{X}_i) \tag{3-6}$$

一般通过对数似然函数 $l(\boldsymbol{\theta}\mid\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_N)$ 求导，并令导数等于零，从而求解得到 $\hat{\boldsymbol{\theta}}$。

3.1.4　条件编码

在条件生成模型中，输入的条件可以是离散的类别标签、一段文本，也可以是一张图像。然而，这些原始条件往往无法直接输入生成模型，或者生成模型难以直接理解它们。为了使生成模型能够根据这些条件生成相应的样本，必须对条件进行编码，即将原始条件转换为生成模型能够理解和处理的表示形式。

1．独热编码

独热编码（One-Hot Encoding）是一种将类别标签转换为数值数据的编码方式，通常用于机器学习和数据挖掘。其核心思想是将每个类别标签表示为一个独特的二进制向量。这个二进制向量的长度等于类别标签可能取值的个数。而在这个向量中，只有一个位置为 1，其他位置全部为 0，这就是“独热”名称的由来。

假设有一个类别标签 y，它的可能取值为红色、绿色、蓝色，这三个颜色的独热编码如下。

- 红色：$[1,0,0]$。
- 绿色：$[0,1,0]$。
- 蓝色：$[0,0,1]$。

在这个例子中，每个颜色都被转换为一个长度为 3 的二进制向量。二进制向量的每个位

置都对应一个颜色，只有代表当前颜色的位置为 1，其他位置全部为 0。

独热编码在各种机器学习算法中得到了广泛应用，尤其是在需要数值输入的算法中，如逻辑回归、支持向量机和神经网络。通过将分类数据转换为独热编码，算法能够更有效地处理这些数据，因为每个类别标签都被表示为一个唯一的数值。然而，独热编码也存在一些局限性。当类别标签数量较多时，它会导致特征空间变得非常稀疏，从而增加模型的复杂性和计算成本。在这种情况下，可以考虑使用其他编码方式，如标签嵌入（Label Embedding）。

2．标签嵌入

标签嵌入是一种将离散标签（如类别标签）转换为连续向量的方法，广泛应用于自然语言处理、推荐系统及多标签分类等任务。与独热编码不同，标签嵌入能够将标签映射到低维的连续向量空间中，可以在处理复杂标签结构时捕捉到标签之间的潜在关系，并为生成模型和分类模型提供更丰富的语义信息。

假设标签 y 的可能取值有 n 个，且构成标签集合 $\{y_1, y_2, \cdots, y_n\}$。标签嵌入需要学习一个映射，能够将每个标签 y_i 映射到一个 d 维的嵌入向量 $\boldsymbol{e}_i \in \mathbb{R}^d$ 中。这些嵌入向量构成了一个嵌入矩阵 $\boldsymbol{E} \in \mathbb{R}^{n \times d}$，即

$$\boldsymbol{E} = \begin{bmatrix} \boldsymbol{e}_1 \\ \boldsymbol{e}_2 \\ \vdots \\ \boldsymbol{e}_n \end{bmatrix} \tag{3-7}$$

对于标签 y_i，其对应的嵌入向量 $\boldsymbol{e}_i$ 为嵌入矩阵 $\boldsymbol{E}$ 的第 i 行，可记为 $\boldsymbol{E}[i,:]$。在后续任务中，只需用嵌入向量 $\boldsymbol{e}_i$ 来表示标签 y_i。

通常，需要通过优化某种损失函数来学习嵌入矩阵 $\boldsymbol{E}$。在多标签分类问题中，常见的做法是最小化以下损失函数：

$$\mathcal{L}(\boldsymbol{E}) = \sum_{(\boldsymbol{x}, y) \in D} l(f(\boldsymbol{x}; \boldsymbol{E}), y) \tag{3-8}$$

式中，D 为训练数据集；$\boldsymbol{x}$ 为输入特征；y 为 $\boldsymbol{x}$ 的分类标签；$f(\boldsymbol{x}; \boldsymbol{E})$ 为通过模型（如神经网络）预测类别标签；$l(\cdot,\cdot)$ 为某个损失函数（如交叉熵损失函数）。

3．文本嵌入

当条件是文本时，**文本嵌入**（Text Embedding）能够将离散的文本转换为一个实数向量，从而使生成模型更容易理解和处理输入的条件。常见的文本嵌入主要包括**词嵌入**（Word Embedding）和**句子嵌入**（Sentence Embedding）两种。

词嵌入将文本中的每个单词都映射为一个嵌入向量，通常通过词嵌入矩阵（如 Word2Vec、GloVe 等）来实现。句子嵌入则将一整段文本（如一个句子或段落）整体映射为一个固定维度的实数向量。常用的方法包括对词嵌入进行平均或加权平均、使用预训练语言模型（如 BERT）生成句子的向量表示。

4．图像嵌入

当条件是图像时，可以利用**图像嵌入**（Image Embedding）将给定的图像转换为一个低维向量表示，以便生成模型处理和理解。常用的方法是通过预训练的卷积神经网络（如 ResNet、

VGG 等）来提取图像的特征向量。

假设 $\boldsymbol{I}$ 为给定的图像，可以将 $\boldsymbol{I}$ 输入一个预训练的卷积神经网络，经过多层卷积和池化操作，得到一个特征向量 $f(\boldsymbol{I})$，即

$$f(\boldsymbol{I}) = \mathrm{CNN}(\boldsymbol{I}) \tag{3-9}$$

特征向量 $f(\boldsymbol{I})$ 可以作为 $\boldsymbol{I}$ 的替代输入生成模型，用于指导生成过程。

5．组合嵌入

在许多情况下，条件可能不仅仅是一种数据，而是多种数据的组合（如文本+标签、图像+文本）。在这种情况下，可以使用**组合嵌入**（Combined Embedding），将各个条件的嵌入向量进行组合，常用的方法有向量拼接和线性变换。

假设有两个输入条件，分别为标签和文本。标签可通过标签嵌入转换为嵌入向量 $\boldsymbol{e}$，而文本则可通过文本嵌入转换为嵌入向量 $\boldsymbol{s}$。向量拼接就是将这两个嵌入向量直接拼接为一个更大的向量作为条件，即

$$\boldsymbol{c} = [\boldsymbol{e}, \boldsymbol{s}] \tag{3-10}$$

另一种方法是对每个嵌入向量都应用一个线性变换，再组合，即

$$\boldsymbol{c} = \boldsymbol{W}_1\boldsymbol{e} + \boldsymbol{W}_2\boldsymbol{s} \tag{3-11}$$

式中，$\boldsymbol{W}_1$ 和 $\boldsymbol{W}_2$ 为两个线性变换矩阵。

3.1.5　马尔可夫链

马尔可夫链（Markov Chain）是一种数学模型，其描述了一个系统在不同状态之间的随机转移过程。它的特点是系统的未来状态只依赖当前状态，而与历史状态无关，这种特性被称为**马尔可夫性质**（Markov Property）。

设 $\{X_t\}_{t\geqslant 0}$ 是一个离散时间随机过程，X_t 表示 t 时刻系统的状态。若对于任意时刻 t 和任意状态 $x_0, x_1, \cdots, x_{t+1}$ 都满足以下条件：

$$\Pr(X_{t+1} = x_{t+1} \mid X_t = x_t, X_{t-1} = x_{t-1}, \cdots, X_0 = x_0) = \Pr(X_{t+1} = x_{t+1} \mid X_t = x_t) \tag{3-12}$$

则称这个随机过程具有**马尔可夫性质**，即系统的未来状态 X_{t+1} 只依赖当前状态 X_t，而与历史状态无关。满足马尔可夫性质的随机过程又被称为马尔可夫链。

对于状态空间 $\mathcal{S}$ 中的任意两个状态 $i, j \in \mathcal{S}$，定义从状态 i 转移到状态 j 的概率为

$$P_{ij} = \Pr(X_{t+1} = j \mid X_t = i) \tag{3-13}$$

则 P_{ij} 被称为状态转移概率。如果状态空间 $\mathcal{S}$ 的大小 K 是有限的，则马尔可夫链的状态转移概率可以用一个矩阵 $\boldsymbol{P} \in \mathbb{R}^{K\times K}$ 来表示，称为状态转移矩阵（Transition Matrix）。

马尔可夫链的初始状态 X_0 的随机性可由初始分布 π_0 描述，即

$$\pi_0(i) = \Pr(X_0 = i) \tag{3-14}$$

而在 t 时刻，系统状态 X_t 的随机性则由状态分布 π_t 描述。状态分布 π_t 是一个向量，每个元素都表示该时刻各个状态的概率分布，即

$$\pi_t(j) = \Pr(X_t = j) \tag{3-15}$$

不同时刻的状态分布可以通过以下递推关系得到

$$\pi_{t+1} = \pi_t \boldsymbol{P} \tag{3-16}$$

对于某些马尔可夫链，经过足够长的时间后，其状态分布可能会收敛到一个稳态分布 π，即

$$\pi = \pi \boldsymbol{P} \tag{3-17}$$

稳态分布表示在无限长时间后，系统的状态分布不再随时间变化而变化。

3.1.6 重参数化技巧

重参数化（Reparameterization）是处理具有如下形式目标函数的一种有效技巧：

$$L(\boldsymbol{\alpha}) = E_{\boldsymbol{Z}\sim p_{\boldsymbol{\alpha}}(\boldsymbol{z})}[h(\boldsymbol{Z})] \tag{3-18}$$

式中，$\boldsymbol{Z}$ 为离散型或连续型随机向量；$p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ 为含有参数 $\boldsymbol{\alpha}$ 的概率密度函数或概率质量函数。若 $\boldsymbol{Z}$ 是连续型随机向量，则 $L(\boldsymbol{\alpha})$ 可以写成如下积分形式：

$$L(\boldsymbol{\alpha}) = E_{\boldsymbol{Z}\sim p_{\boldsymbol{\alpha}}(\boldsymbol{z})}[h(\boldsymbol{Z})] = \int h(\boldsymbol{z}) p_{\boldsymbol{\alpha}}(\boldsymbol{z}) \mathrm{d}\boldsymbol{z} \tag{3-19}$$

然而，在实际应用中，精确计算式（3-19）中的积分并得出显式表达式通常是不可行的。因此，常见的做法是利用样本均值来估计该数学期望。首先，从分布 $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ 中采样得到 M 个样本 $\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_M$，然后，基于这些样本计算 $h(\boldsymbol{z})$ 的均值，即 $\frac{1}{M}\sum_{j=1}^{M} h(\boldsymbol{z}_j)$。但是，直接从分布 $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ 中采样会使样本 $\boldsymbol{z}_j$ 丧失参数 $\boldsymbol{\alpha}$ 的信息（梯度），从而导致梯度下降法无法更新参数 $\boldsymbol{\alpha}$。重参数化技巧提供了一种变换，使我们可以间接地从分布 $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ 中采样，同时保留参数 $\boldsymbol{\alpha}$ 的梯度。

具体而言，重参数化技巧假设分布 $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ 中的采样过程可以分为以下两个步骤。

（1）从一个与参数 $\boldsymbol{\alpha}$ 无关的 D 维分布 $p(\boldsymbol{\varepsilon})$ 中采样，得到一个样本 $\boldsymbol{\varepsilon}$。

（2）通过变换 $\boldsymbol{z} = t_{\boldsymbol{\alpha}}(\boldsymbol{\varepsilon})$ 生成 $\boldsymbol{z}$，其中 $t_{\boldsymbol{\alpha}}(\cdot)$ 是与参数 $\boldsymbol{\alpha}$ 相关的变换函数。

那么，$L(\boldsymbol{\alpha})$ 可以写成如下形式：

$$L(\boldsymbol{\alpha}) = E_{\boldsymbol{\varepsilon}\sim p(\boldsymbol{\varepsilon})}[h(t_{\boldsymbol{\alpha}}(\boldsymbol{\varepsilon}))] \tag{3-20}$$

而对 $E_{\boldsymbol{\varepsilon}\sim p(\boldsymbol{\varepsilon})}[h(t_{\boldsymbol{\alpha}}(\boldsymbol{\varepsilon}))]$ 的估计可以写为

$$L(\boldsymbol{\alpha}) = E_{\boldsymbol{\varepsilon}\sim p(\boldsymbol{\varepsilon})}[h(t_{\boldsymbol{\alpha}}(\boldsymbol{\varepsilon}))] \approx \frac{1}{M}\sum_{j=1}^{M} h(t_{\boldsymbol{\alpha}}(\boldsymbol{\varepsilon}_j)) \tag{3-21}$$

式中，$\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \cdots, \boldsymbol{\varepsilon}_M$ 为从 $p(\boldsymbol{\varepsilon})$ 中随机采样得到的 M 个观测样本。由式（3-21）可以看出，重参数化技巧将参数 $\boldsymbol{\alpha}$ 从分布 $p_{\boldsymbol{\alpha}}(\boldsymbol{z})$ 转移到变换函数 $t_{\boldsymbol{\alpha}}(\cdot)$ 上，使样本均值 $\frac{1}{M}\sum_{j=1}^{M} h(t_{\boldsymbol{\alpha}}(\boldsymbol{\varepsilon}_j))$ 可以保留参数 $\boldsymbol{\alpha}$ 的梯度。

3.2　数学符号

考虑本章的数学推导内容略为繁复，且包含较多的数学符号，因此我们在表 3-1 中汇总了后续推导所需的关键数学符号，以便读者能够更加顺畅地理解和掌握后续章节的内容。

表 3-1　VAE 中的数学符号

数学符号	含　义	数学符号	含　义
$\boldsymbol{X}$	连续随机向量，一般表示数据（如图像）	$p(\boldsymbol{z})$	隐变量 $\boldsymbol{Z}$ 服从的真实概率分布的密度函数
$\mathcal{X}$	数据 $\boldsymbol{X}$ 所存在的连续高维空间	$p_{\boldsymbol{\theta}}(\boldsymbol{z})$	隐变量 $\boldsymbol{Z}$ 先验分布的密度函数，一般假设为标准高斯分布
$\boldsymbol{x}$	$\boldsymbol{X}$ 的观测样本，是确定的观测值，而不是随机向量	$p_{\boldsymbol{\theta}}(\boldsymbol{z}\mid\boldsymbol{x})$	隐变量 $\boldsymbol{Z}$ 后验分布的密度函数
$\hat{\boldsymbol{x}}$	VAE 生成的样本/数据	$q_{\boldsymbol{\varphi}}(\boldsymbol{z}\mid\boldsymbol{x})$	变分分布，其被假设为高斯分布，且具有均值向量 $\boldsymbol{\mu}_{\boldsymbol{\varphi},\boldsymbol{x}}$ 与方差向量 $\boldsymbol{\sigma}^2_{\boldsymbol{\varphi},\boldsymbol{x}}$
$\boldsymbol{Z}$	连续随机向量，一般表示隐变量	$\ln p_{\boldsymbol{\theta}}(\boldsymbol{x})$	生成模型的对数边际似然函数
$\mathcal{Z}$	隐变量 $\boldsymbol{Z}$ 所在的连续低维空间	$\ln p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z})$	生成模型的联合对数似然函数
$\boldsymbol{z}$	$\boldsymbol{Z}$ 的观测样本，又称编码，是确定的观测值	$\mathrm{ELBO}(q,\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\varphi})$	证据下界，它是 $\ln p_{\boldsymbol{\theta}}(\boldsymbol{x})$ 的一个下界
$p(\boldsymbol{x})$	数据真实概率分布的密度函数	$f_{\mathrm{I}}(\boldsymbol{x};\boldsymbol{\varphi})$	推断网络，具有可学习参数 $\boldsymbol{\varphi}$
$p(\boldsymbol{x}\mid\boldsymbol{z})$	数据真实条件概率分布的密度函数	$f_{\mathrm{G}}(\boldsymbol{x};\boldsymbol{\theta})$	生成网络，具有可学习参数 $\boldsymbol{\theta}$
$p_{\boldsymbol{\theta}}(\boldsymbol{x})$	生成样本的边际概率分布，$\boldsymbol{\theta}$ 为模型的可学习参数	$\mathcal{L}(\boldsymbol{\varphi},\boldsymbol{\theta})$	理论损失函数
$p_{\boldsymbol{\theta}}(\boldsymbol{x}\mid\boldsymbol{z})$	生成样本的条件概率分布，其被假设为高斯分布，其中均值向量为 $\boldsymbol{\mu}_{\boldsymbol{\theta},\boldsymbol{z}}$，方差向量为 $\boldsymbol{\sigma}^2_{\boldsymbol{\theta},\boldsymbol{z}}$	$\mathcal{L}(\boldsymbol{\varphi},\boldsymbol{\theta}\mid\mathcal{D})$	定义在数据集 $\mathcal{D}$ 上的经验损失函数

3.3　总体思路

在探讨图像的生成式建模任务时（具体参见 2.1.3 节），我们将图像抽象为一个定义在高维空间 $\mathcal{X}$ 中的随机向量 $\boldsymbol{X}$，其服从一个未知的概率分布 $p(\boldsymbol{x})$。在这样的高维空间中，直接构建一个参数为 $\boldsymbol{\theta}$ 的生成模型 $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ 以逼近真实的概率密度函数 $p(\boldsymbol{x})$ 通常是十分困难的。为了简化这一复杂问题，我们引入了低维的隐变量 $\boldsymbol{Z}\in\mathcal{Z}$。这一策略允许我们将对 $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ 的直接建模分解为几个更为简单且易于处理的部分：首先是边际密度函数 $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ 的建模，其次是条件密度函数 $p_{\boldsymbol{\theta}}(\boldsymbol{x}\mid\boldsymbol{z})$ 的建模，以及条件密度函数 $p_{\boldsymbol{\theta}}(\boldsymbol{z}\mid\boldsymbol{x})$ 的建模。在此框架中，隐变量 $\boldsymbol{Z}$ 相较于可直接观测的显变量 $\boldsymbol{X}$（Observable Variable，又称可观测变量），是无法直接被观测到的。此外，除非另有说明，本章中我们默认 $\boldsymbol{X}$ 和 $\boldsymbol{Z}$ 均为连续向量，以便进行后续的理论分析与模型构建。

VAE 正是这种含有隐变量的生成模型，其假设随机向量 $\boldsymbol{X}\sim p(\boldsymbol{x})$ 的观测值 $\boldsymbol{x}$ 是经过以下两个步骤得到的。

（1）获得隐变量 $\boldsymbol{Z}$ 的某个取值 $\boldsymbol{z}$。

（2）从真实条件分布 $p(\boldsymbol{x}|\boldsymbol{z})$ 中采样得到观测值 $\boldsymbol{x}$。

若按照以上步骤生成观测值 $\boldsymbol{x}$，则每个观测值 $\boldsymbol{x}$ 都应对应隐变量 $\boldsymbol{Z}$ 的一个取值 $\boldsymbol{z}$。

那么，我们该如何建模，从而实现以上的样本生成步骤，并使生成样本服从真实概率分布 $p(\boldsymbol{x})$ 呢？一种直观的想法是，在引入隐变量 $\boldsymbol{Z}$ 后，首先对 $\boldsymbol{X}$ 和 $\boldsymbol{Z}$ 的联合分布 $p_\theta(\boldsymbol{x},\boldsymbol{z})$ 进行建模，随后以 $p_\theta(\boldsymbol{x},\boldsymbol{z})$ 为似然函数，希望通过 MLE 寻得最优的参数 $\boldsymbol{\theta}$。然而，问题在于，由于 $\boldsymbol{Z}$ 是隐变量，我们无法直接获取它的实际观测样本。因此，直接采用 $p_\theta(\boldsymbol{x},\boldsymbol{z})$ 作为似然函数进行 MLE，在实践中是行不通的。

VAE 的建模思路首先涉及对联合分布 $p_\theta(\boldsymbol{x},\boldsymbol{z})$ 进行以下两种方式的分解：

$$p_\theta(\boldsymbol{x},\boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})p_\theta(\boldsymbol{z}) \tag{3-22}$$

$$p_\theta(\boldsymbol{x},\boldsymbol{z}) = p_\theta(\boldsymbol{z}|\boldsymbol{x})p_\theta(\boldsymbol{x}) \tag{3-23}$$

其中，$p_\theta(\boldsymbol{z})$ 为隐变量的边际分布（又称**先验分布**）；$p_\theta(\boldsymbol{z}|\boldsymbol{x})$ 为隐变量的条件分布（又称**后验分布**）；$p_\theta(\boldsymbol{x}|\boldsymbol{z})$ 为生成样本的条件分布；$p_\theta(\boldsymbol{x})$ 为生成样本的边际分布。为了简化计算，VAE 一般假设先验分布 $p_\theta(\boldsymbol{z})$ 是标准高斯分布，即

$$p_\theta(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z};\boldsymbol{0},\boldsymbol{I}) \tag{3-24}$$

同时，假设条件分布 $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ 也是高斯分布，并具有如下形式：

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_{\theta,\boldsymbol{z}},\mathrm{diag}(\boldsymbol{\sigma}^2_{\theta,\boldsymbol{z}})) \tag{3-25}$$

式中，$\boldsymbol{\mu}_{\theta,\boldsymbol{z}}$ 为均值向量；$\mathrm{diag}(\boldsymbol{\sigma}^2_{\theta,\boldsymbol{z}})$ 为对角线元素是方差向量 $\boldsymbol{\sigma}^2_{\theta,\boldsymbol{z}}$ 的对角化协方差矩阵。此处，均值向量与方差向量均有下标 $\boldsymbol{\theta}$ 与 $\boldsymbol{z}$，表示它们与参数 $\boldsymbol{\theta}$ 及条件 $\boldsymbol{Z}=\boldsymbol{z}$ 有关。

VAE 利用了一个包含可学习参数 $\boldsymbol{\theta}$ 的神经网络，以实现对条件分布 $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ 的参数化建模，并且该网络被命名为**生成网络**（Generative Network）。生成网络能够根据输入 $\boldsymbol{z}$，预测均值向量 $\boldsymbol{\mu}_{\theta,\boldsymbol{z}}$ 和方差向量 $\boldsymbol{\sigma}^2_{\theta,\boldsymbol{z}}$。从条件分布 $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ 中采样，即可产生 VAE 的生成样本。另外，VAE 还采用了一个可以参数化的条件分布 $q_\varphi(\boldsymbol{z}|\boldsymbol{x})$ 来近似后验分布 $p_\theta(\boldsymbol{z}|\boldsymbol{x})$。在下文中，条件分布 $q_\varphi(\boldsymbol{z}|\boldsymbol{x})$ 被称为**变分分布**（Variational Distribution），一般被设定为具有如下形式的高斯分布：

$$q_\varphi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z};\boldsymbol{\mu}_{\varphi,\boldsymbol{x}},\mathrm{diag}(\boldsymbol{\sigma}^2_{\varphi,\boldsymbol{x}})) \tag{3-26}$$

式中，$\boldsymbol{\varphi}$ 为可学习参数；$\boldsymbol{\mu}_{\varphi,\boldsymbol{x}}$ 为均值向量；$\mathrm{diag}(\boldsymbol{\sigma}^2_{\varphi,\boldsymbol{x}})$ 为对角化协方差矩阵。类似地，VAE 用一个包含可学习参数 $\boldsymbol{\varphi}$ 的神经网络来实现对变分分布 $q_\varphi(\boldsymbol{z}|\boldsymbol{x})$ 的参数化建模。该网络被称为**推断网络**（Inference Network），其能够根据观测值 $\boldsymbol{x}$ 来预测变分分布 $q_\varphi(\boldsymbol{z}|\boldsymbol{x})$ 的均值向量 $\boldsymbol{\mu}_{\varphi,\boldsymbol{x}}$ 和方差向量 $\boldsymbol{\sigma}^2_{\varphi,\boldsymbol{x}}$。通过变分分布 $q_\varphi(\boldsymbol{z}|\boldsymbol{x})$，我们可以生成与观测值 $\boldsymbol{x}$ 对应的隐变量取值 $\boldsymbol{z}$。

VAE 希望通过极大化对数边际似然函数 $\ln p_\theta(\boldsymbol{x})$ 来训练生成网络和推断网络。但是，由于 $p_\theta(\boldsymbol{x})$ 的具体形式未知，因此我们无法直接最大化 $\ln p_\theta(\boldsymbol{x})$。幸运的是，根据联合分布 $p_\theta(\boldsymbol{x},\boldsymbol{z})$ 的两种分解形式［式（3-22）与式（3-23）］，我们可以推导出 $\ln p_\theta(\boldsymbol{x})$ 的一个下界。通过最大化该下界，我们可以间接地实现 $\ln p_\theta(\boldsymbol{x})$ 的最大化，进而求得最优的网络参数。

接下来，我们将对以上思路进行详细的介绍。

3.4　理论分析

3.4.1　证据下界

为了使生成模型 $p_\theta(\boldsymbol{x})$ 近似真实概率密度 $p(\boldsymbol{x})$，我们通常采用 MLE（见 3.1.3 节）来估计 $p_\theta(\boldsymbol{x})$ 的未知参数 $\boldsymbol{\theta}$，即最大化对数边际似然函数 $\ln p_\theta(\boldsymbol{x})$。在进行 MLE 之前，我们先进行一些预备推导与符号定义。首先，对式（3-23）等号两侧取对数，可得

$$\ln p_\theta(\boldsymbol{x},\boldsymbol{z}) = \ln p_\theta(\boldsymbol{z}\mid\boldsymbol{x}) + \ln p_\theta(\boldsymbol{x}) \tag{3-27}$$

移项可得

$$\ln p_\theta(\boldsymbol{x}) = \ln p_\theta(\boldsymbol{x},\boldsymbol{z}) - \ln p_\theta(\boldsymbol{z}\mid\boldsymbol{x}) \tag{3-28}$$

此外，我们还需要引入一个隐变量 $\boldsymbol{Z}$ 的额外条件分布，该条件分布以 $\boldsymbol{X}=\boldsymbol{x}$ 为条件，其密度函数被标记为 $q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})$，其中 $\boldsymbol{\varphi}$ 是可学习的参数。这个额外的条件分布通常被称为变分分布，能够帮助我们推导出对数边际似然函数 $\ln p_\theta(\boldsymbol{x})$ 的分解表达式。如式（3-26）所示，我们一般将 $q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})$ 设定为具有对角化协方差矩阵的多元高斯分布 $\mathcal{N}(\boldsymbol{z};\boldsymbol{\mu}_{\varphi,\boldsymbol{x}},\mathrm{diag}(\boldsymbol{\sigma}^2_{\varphi,\boldsymbol{x}}))$。

那么，基于以上公式和定义，对数边际似然函数 $\ln p_\theta(\boldsymbol{x})$ 可被分解为以下形式：

$$\begin{aligned}
&\ln p_\theta(\boldsymbol{x})\\
&=\int q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})\mathrm{d}\boldsymbol{z}\cdot\ln p_\theta(\boldsymbol{x}) && \left[\text{因为}\int q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})\mathrm{d}\boldsymbol{z}=1\right]\\
&=\int q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})\cdot\ln p_\theta(\boldsymbol{x})\mathrm{d}\boldsymbol{z}\\
&=\int q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})\cdot\big(\ln p_\theta(\boldsymbol{x},\boldsymbol{z})-\ln p_\theta(\boldsymbol{z}\mid\boldsymbol{x})\big)\mathrm{d}\boldsymbol{z}\\
&=\int q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})\cdot\big(\ln p_\theta(\boldsymbol{x},\boldsymbol{z})-\ln p_\theta(\boldsymbol{z}\mid\boldsymbol{x})-\ln q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})+\ln q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})\big)\mathrm{d}\boldsymbol{z} && (3\text{-}29)\\
&=\int q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})\ln\left(\frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})}\right)\mathrm{d}\boldsymbol{z}-\int q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})\ln\left(\frac{p_\theta(\boldsymbol{z}\mid\boldsymbol{x})}{q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})}\right)\mathrm{d}\boldsymbol{z}\\
&=E_{\boldsymbol{Z}\sim q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})}\left[\ln\frac{p_\theta(\boldsymbol{x},\boldsymbol{Z})}{q_\varphi(\boldsymbol{Z}\mid\boldsymbol{x})}\right]-\int q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})\ln\left(\frac{p_\theta(\boldsymbol{z}\mid\boldsymbol{x})}{q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})}\right)\mathrm{d}\boldsymbol{z}\\
&=\mathrm{ELBO}(q,\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\varphi})+\mathrm{KL}(q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})\,\|\,p_\theta(\boldsymbol{z}\mid\boldsymbol{x}))\\
&\geqslant\mathrm{ELBO}(q,\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\varphi})
\end{aligned}$$

式中，$\mathrm{KL}(q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})\,\|\,p_\theta(\boldsymbol{z}\mid\boldsymbol{x}))$ 为变分分布 $q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})$ 和后验分布 $p_\theta(\boldsymbol{z}\mid\boldsymbol{x})$ 的 KL 散度；$\mathrm{ELBO}(q,\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\varphi})$ 具有如下形式：

$$\mathrm{ELBO}(q,\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\varphi})=\int q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})\ln\left(\frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})}\right)\mathrm{d}\boldsymbol{z}=E_{\boldsymbol{Z}\sim q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})}\left[\ln\frac{p_\theta(\boldsymbol{x},\boldsymbol{Z})}{q_\varphi(\boldsymbol{Z}\mid\boldsymbol{x})}\right] \tag{3-30}$$

由于 $\mathrm{KL}(q_\varphi(\boldsymbol{z}\mid\boldsymbol{x})\,\|\,p_\theta(\boldsymbol{z}\mid\boldsymbol{x}))\geqslant 0$，因此 $\mathrm{ELBO}(q,\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\varphi})$ 是对数边际似然函数 $\ln p_\theta(\boldsymbol{x})$ 的一个下界，称为**证据下界**（Evidence Lower Bound，ELBO）。通过最大化 $\mathrm{ELBO}(q,\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\varphi})$，我们

可以间接地最大化对数边际似然函数 $\ln p_\theta(\boldsymbol{x})$。

当且仅当 $q_\varphi(\boldsymbol{z}|\boldsymbol{x})=p_\theta(\boldsymbol{z}|\boldsymbol{x})$ 时，$\ln p_\theta(\boldsymbol{x})=\mathrm{ELBO}(q,\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\varphi})$。这也意味着，当变分分布 $q_\varphi(\boldsymbol{z}|\boldsymbol{x})$ 越接近后验分布 $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ 时，$\mathrm{ELBO}(q,\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\varphi})$ 对 $\ln p_\theta(\boldsymbol{x})$ 的替代性越强。

另外，我们还可以借助**琴生不等式**（Jensen's Inequality）来推导 ELBO，即

$$\begin{aligned}\ln p_\theta(\boldsymbol{x}) &= \ln\int p_\theta(\boldsymbol{x},\boldsymbol{z})\mathrm{d}\boldsymbol{z}\\ &= \ln\int\frac{p_\theta(\boldsymbol{x},\boldsymbol{z})q_\varphi(\boldsymbol{z}|\boldsymbol{x})}{q_\varphi(\boldsymbol{z}|\boldsymbol{x})}\mathrm{d}\boldsymbol{z}\\ &= \ln E_{\boldsymbol{Z}\sim q_\varphi(\boldsymbol{z}|\boldsymbol{x})}\left[\frac{p_\theta(\boldsymbol{x},\boldsymbol{Z})}{q_\varphi(\boldsymbol{Z}|\boldsymbol{x})}\right]\\ &\geqslant E_{\boldsymbol{Z}\sim q_\varphi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln\frac{p_\theta(\boldsymbol{x},\boldsymbol{Z})}{q_\varphi(\boldsymbol{Z}|\boldsymbol{x})}\right]\\ &= \mathrm{ELBO}(q,\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\varphi})\end{aligned}\tag{3-31}$$

但是，基于琴生不等式的下界推导无法展现证据下界与对数边际似然函数的理论差值是什么，即忽略了 $\mathrm{KL}(q_\varphi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z}|\boldsymbol{x}))$。

3.4.2 目标函数

根据上述推导，最大化对数边际似然函数的问题被转化为最大化证据下界的问题。随后，我们首先针对单一观测样本 $\boldsymbol{x}$ 来定义 VAE 的**理论目标函数**（Theoretical Objective），即

$$\begin{aligned}&\max_{\boldsymbol{\varphi},\boldsymbol{\theta}}\mathrm{ELBO}(q,\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\varphi})\\ &=\max_{\boldsymbol{\varphi},\boldsymbol{\theta}}E_{\boldsymbol{Z}\sim q_\varphi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln\frac{p_\theta(\boldsymbol{x},\boldsymbol{Z})}{q_\varphi(\boldsymbol{Z}|\boldsymbol{x})}\right]\end{aligned}\tag{3-32}$$

根据式（3-22）中对联合分布的分解 $p_\theta(\boldsymbol{x},\boldsymbol{z})=p_\theta(\boldsymbol{x}|\boldsymbol{z})p_\theta(\boldsymbol{z})$，进一步推导可得

$$\begin{aligned}&\max_{\boldsymbol{\varphi},\boldsymbol{\theta}}\mathrm{ELBO}(q,\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\varphi})\\ &=\max_{\boldsymbol{\varphi},\boldsymbol{\theta}}E_{\boldsymbol{Z}\sim q_\varphi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln\frac{p_\theta(\boldsymbol{x},\boldsymbol{Z})}{q_\varphi(\boldsymbol{Z}|\boldsymbol{x})}\right]\\ &=\max_{\boldsymbol{\varphi},\boldsymbol{\theta}}E_{\boldsymbol{Z}\sim q_\varphi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln\frac{p_\theta(\boldsymbol{x}|\boldsymbol{Z})p_\theta(\boldsymbol{Z})}{q_\varphi(\boldsymbol{Z}|\boldsymbol{x})}\right]\\ &=\max_{\boldsymbol{\varphi},\boldsymbol{\theta}}\left\{\underbrace{E_{\boldsymbol{Z}\sim q_\varphi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln p_\theta(\boldsymbol{x}|\boldsymbol{Z})\right]}_{\text{重构项}}-\underbrace{\mathrm{KL}(q_\varphi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z}))}_{\text{先验匹配项}}\right\}\end{aligned}\tag{3-33}$$

式中，$p_\theta(\boldsymbol{z})$ 为隐变量 $\boldsymbol{Z}$ 的先验分布，一般被假设为标准高斯分布［见式（3-24）］；条件分布 $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ 则被假设为高斯分布 $\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_{\theta,z},\mathrm{diag}(\boldsymbol{\sigma}^2_{\theta,z}))$［见式（3-25）］。

式（3-33）所探讨的理论目标函数仅基于单一观测样本 $\boldsymbol{x}$ 进行定义。若要将其扩展至所有可能服从真实数据分布 $p(\boldsymbol{x})$ 的观测样本，则完整的理论目标函数将呈现为以下形式：

$$\begin{aligned}&\max_{\varphi,\theta} E_{X\sim p(x)}\left[\mathrm{ELBO}(q,\boldsymbol{X};\boldsymbol{\theta},\boldsymbol{\varphi})\right]\\&=\max_{\varphi,\theta} E_{X\sim p(x)}\left[E_{\boldsymbol{Z}\sim q_{\varphi}(z|X)}\left[\ln p_{\theta}(\boldsymbol{X}\mid\boldsymbol{Z})\right]-\mathrm{KL}(q_{\varphi}(\boldsymbol{z}\mid\boldsymbol{X})\,\|\,p_{\theta}(\boldsymbol{z}))\right]\end{aligned}\tag{3-34}$$

接下来，我们将分别对式（3-33）中的**重构项**（Reconstruction Term）与**先验匹配项**（Prior Matching Term）进行分析。

（1）**重构项**：数学期望 $E_{\boldsymbol{Z}\sim q_{\varphi}(z|x)}\left[\ln p_{\theta}(\boldsymbol{x}\mid\boldsymbol{Z})\right]$ 一般可以用样本均值进行估计。假设对于每个样本 $\boldsymbol{x}$，从变分分布 $q_{\varphi}(\boldsymbol{z}\mid\boldsymbol{x})$ 中采样得到 M 个样本，即 $\boldsymbol{z}_1,\boldsymbol{z}_2,\cdots,\boldsymbol{z}_M$，则重构项的估计可以写为

$$E_{\boldsymbol{Z}\sim q_{\varphi}(z|x)}\left[\ln p_{\theta}(\boldsymbol{x}\mid\boldsymbol{Z})\right]\approx\frac{1}{M}\sum_{j=1}^{M}\ln p_{\theta}(\boldsymbol{x}\mid\boldsymbol{z}_j)\tag{3-35}$$

请注意，数学期望 $E_{\boldsymbol{Z}\sim q_{\varphi}(z|x)}\left[\ln p_{\theta}(\boldsymbol{x}\mid\boldsymbol{Z})\right]$ 是推断参数 $\boldsymbol{\varphi}$ 的函数（将在 3.5.1 节中详细介绍）。但是，重构项的样本均值 $\frac{1}{M}\sum_{j=1}^{M}\ln p_{\theta}(\boldsymbol{x}\mid\boldsymbol{z}_j)$ 不再包含参数 $\boldsymbol{\varphi}$。那么，$\frac{1}{M}\sum_{j=1}^{M}\ln p_{\theta}(\boldsymbol{x}\mid\boldsymbol{z}_j)$ 关于参数 $\boldsymbol{\varphi}$ 的梯度将为 0，无法进行梯度的反向传播。在这种情况下，梯度下降法对式（3-33）所代表的优化问题不再适用。这个问题的出现主要是由于样本 $\boldsymbol{z}_j$ 是从 $q_{\varphi}(\boldsymbol{z}\mid\boldsymbol{x})$ 中随机采样得到的，即 $\boldsymbol{z}_j$ 与 $\boldsymbol{\varphi}$ 之间的关系是不确定的，无法用固定的、可导的函数表示。然而，这个问题能够通过**重参数化技巧**解决（见 3.1.6 节）。

具体来说，我们可以令 3.1.6 节中的 $p(\boldsymbol{\varepsilon})$ 为 D 维的标准高斯分布 $\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$。同时，将变换函数 $t_{\varphi}(\boldsymbol{\varepsilon})$ 定义为如下形式：

$$\boldsymbol{z}=t_{\varphi}(\boldsymbol{\varepsilon})=\boldsymbol{\mu}_{\varphi,x}+\boldsymbol{\sigma}_{\varphi,x}\odot\boldsymbol{\varepsilon}\tag{3-36}$$

式中，$\boldsymbol{\varepsilon}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$；$\odot$ 表示对应元素相乘；$\boldsymbol{\mu}_{\varphi,x}$ 和 $\boldsymbol{\sigma}_{\varphi,x}$ 由推断网络 $f_1(\boldsymbol{x};\boldsymbol{\varphi})$ 根据 $\boldsymbol{x}$ 输出（将在 3.5 节中详细介绍），因此它们均可以看作参数 $\boldsymbol{\varphi}$ 的函数。经过如上变换，$\boldsymbol{z}$ 不再是直接从变分分布 $q_{\varphi}(\boldsymbol{z}\mid\boldsymbol{x})$ 中采样得到的，而是通过均值向量 $\boldsymbol{\mu}_{\varphi,x}$、方差向量 $\boldsymbol{\sigma}_{\varphi,x}$ 及高斯白噪声 $\boldsymbol{\varepsilon}$ 的线性组合得到的。这意味着，$\boldsymbol{z}$ 和 $\boldsymbol{\varphi}$ 之间的关系从不确定的“采样关系”变成了确定性的“函数关系”，从而可以求解 $\boldsymbol{z}$ 关于 $\boldsymbol{\varphi}$ 的梯度。

若使用重参数化技巧生成 $\boldsymbol{z}$ 后，再次用样本均值来估计重构项中的数学期望，则样本均值 $\frac{1}{M}\sum_{j=1}^{M}\ln p_{\theta}(\boldsymbol{x}\mid\boldsymbol{z}_j)$ 中的 $\boldsymbol{z}_j$ 不再单纯是 $\boldsymbol{Z}$ 的观测值，还是参数 $\boldsymbol{\varphi}$ 的函数。样本均值关于参数 $\boldsymbol{\varphi}$ 的梯度将不再是 0，从而可以根据梯度反向传播来更新参数 $\boldsymbol{\varphi}$。

（2）**先验匹配项**：根据式（3-26），$q_{\varphi}(\boldsymbol{z}\mid\boldsymbol{x})$ 被假设为高斯分布 $\mathcal{N}(\boldsymbol{z};\boldsymbol{\mu}_{\varphi,x},\mathrm{diag}(\boldsymbol{\sigma}_{\varphi,x}^2))$。对于隐变量的先验分布 $p_{\theta}(\boldsymbol{z})$，我们一般假设其为标准高斯分布 $\mathcal{N}(\boldsymbol{z};\boldsymbol{0},\boldsymbol{I})$。在这种情况下，$\mathrm{KL}(q_{\varphi}(\boldsymbol{z}\mid\boldsymbol{x})\,\|\,p_{\theta}(\boldsymbol{z}))$ 具有显式表达式。

根据 KL 散度的定义，在 D 维空间中的两个高斯分布 $\mathcal{N}(\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1)$ 和 $\mathcal{N}(\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2)$ 的 KL 散度具有显式表达式，即

$$\begin{aligned}&\mathrm{KL}(\mathcal{N}(\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1)\,\|\,\mathcal{N}(\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2))\\&=\frac{1}{2}\left(\mathrm{tr}(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1)+(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)-D+\ln\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}\right)\end{aligned}\tag{3-37}$$

式中，$\mathrm{tr}(\cdot)$ 表示矩阵的迹；$|\cdot|$ 表示矩阵的行列式。

由式（3-37）可得，若 $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})=\mathcal{N}(\boldsymbol{z};\boldsymbol{\mu}_{\varphi,x},\mathrm{diag}(\boldsymbol{\sigma}^2_{\varphi,x}))$ 和 $p_{\theta}(\boldsymbol{z})=\mathcal{N}(\boldsymbol{z};\boldsymbol{0},\boldsymbol{I})$，则先验匹配项的显式表达式为

$$\mathrm{KL}(q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})\|p_{\theta}(\boldsymbol{z}))=\frac{1}{2}\Big(\mathrm{tr}(\mathrm{diag}(\boldsymbol{\sigma}^2_{\varphi,x}))+\boldsymbol{\mu}^{\mathrm{T}}_{\varphi,x}\boldsymbol{\mu}_{\varphi,x}-D-\ln(|\,\mathrm{diag}(\boldsymbol{\sigma}^2_{\varphi,x})\,|)\Big) \tag{3-38}$$

式中，D 为隐变量 $\boldsymbol{z}$ 的维度，是一个超参数。

3.5 模型结构

在 3.3 节和 3.4 节中，我们介绍了 VAE 需要用两个神经网络分别对变分分布 $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})$ 和条件分布 $p_{\theta}(\boldsymbol{x}|\boldsymbol{z})$ 进行建模。如图 3-2 所示，这两个网络分别被称为**推断网络**和**生成网络**，它们构成了 VAE 的主体结构。本节将深入剖析这两个神经网络的具体细节。

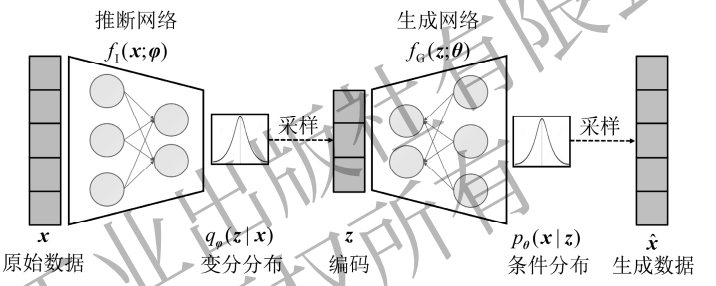


图 3-2 VAE 的主体结构

3.5.1 推断网络

推断网络记为 $f_{\mathrm{I}}(\boldsymbol{x};\boldsymbol{\varphi})$，其定义了一个从数据空间到分布空间的映射。具体来说，推断网络将原始数据 $\boldsymbol{x}$ 映射到定义在隐空间 $\mathcal{Z}$ 中的某个变分分布 $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})$ 上。推断网络可以是以网络参数为 $\boldsymbol{\varphi}$ 的全连接神经网络或卷积神经网络。在这种情况下，推断网络的输入为 $\boldsymbol{x}$，输出为变分分布 $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})$ 的分布参数。

为了简化计算，在式（3-26）中，我们将变分分布 $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})$ 设定为具有对角化协方差的多元高斯分布，即 $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})=\mathcal{N}(\boldsymbol{z};\boldsymbol{\mu}_{\varphi,x},\mathrm{diag}(\boldsymbol{\sigma}^2_{\varphi,x}))$。因为高斯分布由其均值和方差唯一决定，所以，推断网络只需将原始数据 $\boldsymbol{x}$ 映射为变分分布 $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})$ 的均值向量 $\boldsymbol{\mu}_{\varphi,x}$ 和方差向量 $\boldsymbol{\sigma}^2_{\varphi,x}$，即

$$\begin{bmatrix}\boldsymbol{\mu}_{\varphi,x}\\ \boldsymbol{\sigma}^2_{\varphi,x}\end{bmatrix}=f_{\mathrm{I}}(\boldsymbol{x};\boldsymbol{\varphi}) \tag{3-39}$$

但请注意，出于训练的稳定性等考虑，在实际训练中，$f_{\mathrm{I}}(\boldsymbol{x};\boldsymbol{\varphi})$ 并不直接输出 $\boldsymbol{\sigma}^2_{\varphi,x}$，而是输出 $\ln(\boldsymbol{\sigma}_{\varphi,x})$。

给定一个原始数据 $\boldsymbol{x}$，我们可以通过推断网络得到一组均值向量 $\boldsymbol{\mu}_{\varphi,x}$ 和方差向量 $\boldsymbol{\sigma}^2_{\varphi,x}$，两者唯一决定了一个高斯分布 $\mathcal{N}(\boldsymbol{z};\boldsymbol{\mu}_{\varphi,x},\mathrm{diag}(\boldsymbol{\sigma}^2_{\varphi,x}))$。从该高斯分布中随机采样得到一个编码 $\boldsymbol{z}$

作为后续生成网络的输入。请注意，不同的 $\boldsymbol{x}$ 可能对应不同的高斯分布。因此，$\boldsymbol{\mu}_{\varphi,x}$ 和 $\boldsymbol{\sigma}^2_{\varphi,x}$ 的下标均包含 $\boldsymbol{x}$，表示它们与原始数据 $\boldsymbol{x}$ 有关。另外，$\boldsymbol{\mu}_{\varphi,x}$ 和 $\boldsymbol{\sigma}^2_{\varphi,x}$ 的下标均包含 $\boldsymbol{\varphi}$，表示它们是参数 $\boldsymbol{\varphi}$ 的函数。

3.5.2　生成网络

生成网络记为 $f_{\mathrm{G}}(\boldsymbol{x};\boldsymbol{\theta})$，其定义了一个从隐空间到分布空间的映射。具体来说，假设从变分分布 $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})$ 中采样得到一个编码 $\boldsymbol{z}$，生成网络可将编码 $\boldsymbol{z}$ 映射为数据的条件分布 $p_{\theta}(\boldsymbol{x}|\boldsymbol{z})$。类似地，生成网络可以是以网络参数为 $\boldsymbol{\theta}$ 的全连接神经网络或卷积神经网络。在这种情况下，生成网络的输入为编码 $\boldsymbol{z}$，输出为条件分布 $p_{\theta}(\boldsymbol{x}|\boldsymbol{z})$ 的分布参数。

为了简化计算，在式（3-25）中，我们假设条件分布 $p_{\theta}(\boldsymbol{x}|\boldsymbol{z})$ 是多元高斯分布 $\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_{\theta,z},\mathrm{diag}(\boldsymbol{\sigma}^2_{\theta,z}))$。那么，生成网络的任务就是根据编码 $\boldsymbol{z}$，输出 $p_{\theta}(\boldsymbol{x}|\boldsymbol{z})$ 的均值向量 $\boldsymbol{\mu}_{\theta,z}$ 和方差向量 $\boldsymbol{\sigma}^2_{\theta,z}$，即

$$\begin{bmatrix}\boldsymbol{\mu}_{\theta,z}\\ \boldsymbol{\sigma}^2_{\theta,z}\end{bmatrix}=f_{\mathrm{G}}(\boldsymbol{z};\boldsymbol{\theta}) \tag{3-40}$$

类似地，在实际训练中，$f_{\mathrm{G}}(\boldsymbol{z};\boldsymbol{\theta})$ 并不直接输出 $\boldsymbol{\sigma}^2_{\theta,z}$，而是输出 $\ln(\boldsymbol{\sigma}_{\theta,z})$。

在下文中，为了进一步简化计算，我们假设 $p_{\theta}(\boldsymbol{x}|\boldsymbol{z})$ 的协方差矩阵为单位矩阵，即

$$p_{\theta}(\boldsymbol{x}|\boldsymbol{z})=\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_{\theta,z},\boldsymbol{I}) \tag{3-41}$$

在这种情况下，生成网络只需输出均值向量 $\boldsymbol{\mu}_{\theta,z}$，即

$$\hat{\boldsymbol{x}}\triangleq\boldsymbol{\mu}_{\theta,z}=f_{\mathrm{G}}(\boldsymbol{z};\boldsymbol{\theta}) \tag{3-42}$$

在该设定下，我们一般直接把均值向量 $\boldsymbol{\mu}_{\theta,z}$ 作为 VAE 的生成数据，并记为 $\hat{\boldsymbol{x}}$，而不再需要从 $p_{\theta}(\boldsymbol{x}|\boldsymbol{z})=\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_{\theta,z},\boldsymbol{I})$ 中进行采样。

3.5.3　VAE 与 AE 的对比

VAE 的名称一半来自“变分分布”，另一半则来自“AE”。比较图 3-1 和图 3-2 可知，VAE 的结构和 AE 的结构类似，也具有类似“编码器–解码器”的结构。我们可以把推断网络看作编码器，而把生成网络看作解码器。但是 AE 和 VAE 所定义的映射却完全不同。AE 的编码器先将原始数据 $\boldsymbol{x}$ 映射到编码 $\boldsymbol{z}$，再由解码器将编码 $\boldsymbol{z}$ 映射到重构数据 $\hat{\boldsymbol{x}}$。但是，VAE 的推断网络则先将原始数据 $\boldsymbol{x}$ 映射为隐变量的变分分布 $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})$，再由生成网络将隐变量的取值 $\boldsymbol{z}$ 映射为生成数据的条件分布 $p_{\theta}(\boldsymbol{x}|\boldsymbol{z})$。并且，AE 是依赖重构误差进行训练的，其核心在于最小化原始数据与重构数据之间的差异；而 VAE 的训练，本质上则是基于 MLE 的原理，旨在最大化对数边际似然函数的 ELBO。因此，AE 与 VAE 在基本原理上存在根本性的不同。

3.6 训练算法

在式（3-34）中，我们定义了 VAE 的理论目标函数。对该目标函数取负，可得到**理论损失函数**（Theoretical Loss Function），即

$$\mathcal{L}(\boldsymbol{\varphi},\boldsymbol{\theta}) = E_{\boldsymbol{X}\sim p(\boldsymbol{x})}\Big[-E_{\boldsymbol{Z}\sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{X})}\big[\ln p_{\boldsymbol{\theta}}(\boldsymbol{X}\mid\boldsymbol{Z})\big] + \mathrm{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}\mid\boldsymbol{X})\,\|\,p_{\boldsymbol{\theta}}(\boldsymbol{z}))\Big] \tag{3-43}$$

给定样本量为 N 的训练集 $\mathcal{D}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N\}$，对于每个训练样本 $\boldsymbol{x}_i$，随机采样 M 个高斯噪声 $\boldsymbol{\varepsilon}_{i,j}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$，$1\leqslant j\leqslant M$，并根据式（3-36）计算 $\boldsymbol{z}_{i,j}$。那么，VAE 在训练集 $\mathcal{D}$ 上的**经验损失函数**（Empirical Loss Function）被定义为

$$\begin{aligned}&\mathcal{L}(\boldsymbol{\varphi},\boldsymbol{\theta}\mid\mathcal{D})\\ &=\frac{1}{N}\sum_{i=1}^{N}\left(-\frac{1}{M}\sum_{j=1}^{M}\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}_i\mid\boldsymbol{z}_{i,j}) + \mathrm{KL}\Big(\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\varphi},\boldsymbol{x}_i},\mathrm{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\varphi},\boldsymbol{x}_i}))\,\|\,\mathcal{N}(\boldsymbol{0},\boldsymbol{I})\Big)\right)\end{aligned} \tag{3-44}$$

经验损失函数 $\mathcal{L}(\boldsymbol{\varphi},\boldsymbol{\theta}\mid\mathcal{D})$ 可以看作理论损失函数 $\mathcal{L}(\boldsymbol{\varphi},\boldsymbol{\theta})$ 的一种估计。

在上文中，我们已经假设 $p_{\boldsymbol{\theta}}(\boldsymbol{x}\mid\boldsymbol{z})=\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_{\boldsymbol{\theta},\boldsymbol{z}},\boldsymbol{I})$。那么，$\mathcal{L}(\boldsymbol{\varphi},\boldsymbol{\theta}\mid\mathcal{D})$ 可进一步简化为

$$\begin{aligned}&\mathcal{L}(\boldsymbol{\varphi},\boldsymbol{\theta}\mid\mathcal{D})\\ &=\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2M}\sum_{j=1}^{M}\|\boldsymbol{x}_i-\boldsymbol{\mu}_{\boldsymbol{\theta},\boldsymbol{z}_{i,j}}\|_2^2 + \mathrm{KL}\Big(\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\varphi},\boldsymbol{x}_i},\mathrm{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\varphi},\boldsymbol{x}_i}))\,\|\,\mathcal{N}(\boldsymbol{0},\boldsymbol{I})\Big)\right)\\ &=\frac{1}{2NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\|\boldsymbol{x}_i-\boldsymbol{\mu}_{\boldsymbol{\theta},\boldsymbol{z}_{i,j}}\|_2^2+\frac{1}{N}\sum_{i=1}^{N}\mathrm{KL}(\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\varphi},\boldsymbol{x}_i},\mathrm{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\varphi},\boldsymbol{x}_i}))\,\|\,\mathcal{N}(\boldsymbol{0},\boldsymbol{I}))\end{aligned} \tag{3-45}$$

其中，两个高斯分布之间的 KL 散度可以根据式（3-37）来计算。

对于式（3-45），我们一般还会在 KL 散度之前加上一个超参数 $\lambda>0$，即

$$\begin{aligned}&\mathcal{L}(\boldsymbol{\varphi},\boldsymbol{\theta}\mid\mathcal{D})\\ &=\underbrace{\frac{1}{2NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\|\boldsymbol{x}_i-\boldsymbol{\mu}_{\boldsymbol{\theta},\boldsymbol{z}_{i,j}}\|_2^2}_{\text{重构损失}\mathcal{L}_{\text{rec}}}+\lambda\cdot\underbrace{\frac{1}{N}\sum_{i=1}^{N}\mathrm{KL}(\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\varphi},\boldsymbol{x}_i},\mathrm{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\varphi},\boldsymbol{x}_i})\boldsymbol{I})\,\|\,\mathcal{N}(\boldsymbol{0},\boldsymbol{I}))}_{\text{KL损失}\mathcal{L}_{\text{KL}}}\end{aligned} \tag{3-46}$$

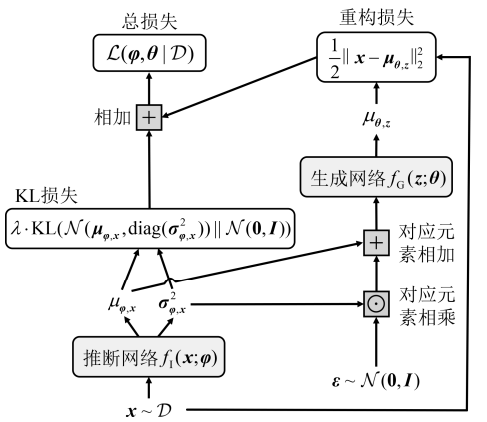


图 3-3 变分自编码器的训练过程

其中，第一项是重构损失 $\mathcal{L}_{\text{rec}}$；第二项是 KL 损失 $\mathcal{L}_{\text{KL}}$。KL 损失可以看作一个正则化项，λ 便是正则化系数。若 $\lambda=1$，则表示原始 VAE。若 $\lambda>1$，则表示后文介绍的 Beta-VAE（见 3.8.2 节）。

VAE 的总体训练过程可以用图 3-3 和算法 3-1 概括。

算法 3-1：VAE 的训练算法

输入：数据集 $\mathcal{D}$、隐空间维度 D、学习率 α、训练轮次（Epoch）T、批（Batch）大小 B

输出：完成训练的推断网络 $f_{\mathrm{I}}(\boldsymbol{x};\boldsymbol{\varphi})$ 和生成网络 $f_{\mathrm{G}}(\boldsymbol{z};\boldsymbol{\theta})$

初始化推断网络和生成网络的参数 $\boldsymbol{\varphi}$ 和 $\boldsymbol{\theta}$；
for $t=1$ to T **do**
　for 每批 $\mathcal{B}\subset\mathcal{D}$（大小为 B）**do**
　　for 每个样本 $\boldsymbol{x}\in\mathcal{B}$ **do**
　　通过推断网络计算 $[\boldsymbol{\mu}_{\varphi,x},\ln\boldsymbol{\sigma}_{\varphi,x}]^{\mathrm{T}}=f_{\mathrm{I}}(\boldsymbol{x};\boldsymbol{\varphi})$；
　　采样 $\boldsymbol{\varepsilon}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$；
　　计算隐变量 $\boldsymbol{z}=\boldsymbol{\mu}_{\varphi,x}+\boldsymbol{\sigma}_{\varphi,x}\odot\boldsymbol{\varepsilon}$（重参数化技巧）；
　　通过解码器重构原始数据 $\hat{\boldsymbol{x}}\triangleq\boldsymbol{\mu}_{\theta,z}=f_{\mathrm{G}}(\boldsymbol{z};\boldsymbol{\theta})$；
　　计算重构损失 $\mathcal{L}_{\mathrm{rec}}=0.5\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2^2$；
　　计算 KL 损失 $\mathcal{L}_{\mathrm{KL}}=\mathrm{KL}(\mathcal{N}(\boldsymbol{\mu}_{\varphi,x},\mathrm{diag}(\boldsymbol{\sigma}_{\varphi,x}^2))\|\mathcal{N}(\boldsymbol{0},\boldsymbol{I}))$；
　　计算总损失 $\mathcal{L}=\mathcal{L}_{\mathrm{rec}}+\lambda\cdot\mathcal{L}_{\mathrm{KL}}$（其中 $\lambda>0$ 是超参数）；
　　end for
　　使用反向传播算法和优化器更新 $\boldsymbol{\varphi}$ 和 $\boldsymbol{\theta}$ 以最小化 $\frac{1}{B}\sum_{\boldsymbol{x}\in\mathcal{B}}\mathcal{L}$；
　end for
end for

3.7　采样算法

由于编码 $\boldsymbol{z}$ 既可以从变分分布 $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ 中采样获得，又可以从先验分布 $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ 中采样获得，并且变分分布 $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ 被要求近似后验分布 $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$。因此，从预训练的生成网络中采样可以分为**后验采样**和**先验采样**两种方式。

3.7.1　后验采样

后验采样需要基于训练集 $\mathcal{D}$ 中的样本和预训练的推断网络完成（见图 3-4）。具体来说，首先将训练集中的某个样本 $\boldsymbol{x}$ 输入已完成训练的推断网络 $f_{\mathrm{I}}(\boldsymbol{x};\boldsymbol{\varphi})$，得到变分分布 $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ 的均值向量 $\boldsymbol{\mu}_{\varphi,x}$ 和方差向量 $\boldsymbol{\sigma}_{\varphi,x}^2$。然后基于 $\boldsymbol{\mu}_{\varphi,x}$、$\boldsymbol{\sigma}_{\varphi,x}^2$ 及高斯白噪声 $\boldsymbol{\varepsilon}$，根据重参数化技巧［见式（3-36）］得到隐变量的取值 $\boldsymbol{z}$。最后将 $\boldsymbol{z}$ 输入已完成训练的生成网络 $f_{\mathrm{G}}(\boldsymbol{z};\boldsymbol{\theta})$，得到条件分布 $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})=\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_{\theta,z},\boldsymbol{I})$ 的均值 $\boldsymbol{\mu}_{\theta,z}$，并将 $\boldsymbol{\mu}_{\theta,z}$ 作为生成样本 $\hat{\boldsymbol{x}}$。

后验采样能够生成较为真实的样本，但其多样性相对较低，即输出数据与输入数据具有较高的相似性。我们在 MNIST 数据集上训练了一个 VAE，并采用后验采样，结合图 3-5（a）中的输入图像来生成样本。从图 3-5（b）中可以看出，输出图像与输入图像较为相似。

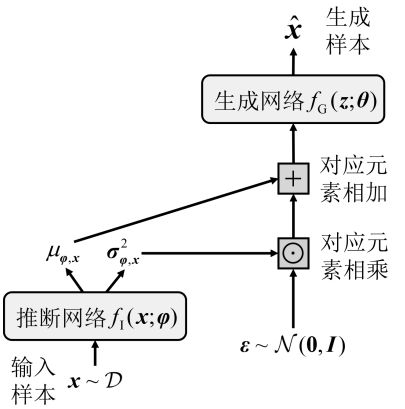
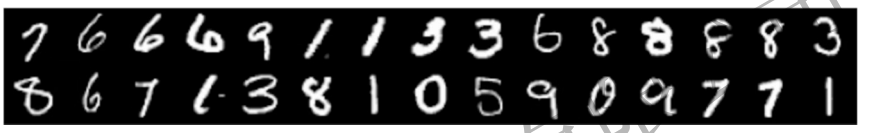
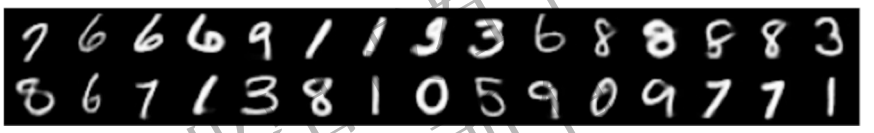


图 3-4　后验采样



（a）输入图像



（b）输出图像

图 3-5　后验采样示例

3.7.2　先验采样

与后验采样不同，先验采样只需生成网络，而不需要输入样本及推断网络，并且流程也更加简单（见图 3-6）。具体而言，从隐变量的先验分布 $p_\theta(z)=\mathcal{N}(z;0,I)$ 中随机生成一个高斯噪声 z，直接输入生成网络 $f_G(z;\theta)$ 即可生成样本 $\hat{x}$。使用先验采样并不总是生成较为真实的数据，但其具有很高的多样性。如图 3-7 所示，在 MNIST 数据集上训练的 VAE 通过先验采样可能生成视觉质量较差的数据，但是可以生成多样性更高的样本。

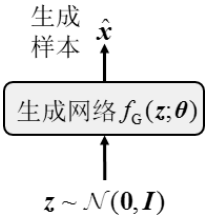


图 3-6　先验采样

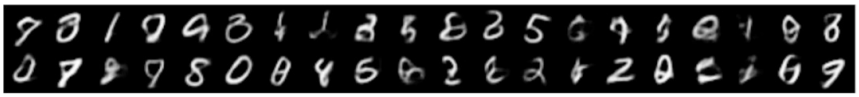


图 3-7　先验采样示例

3.8　变体模型

以上所介绍的是经典 VAE 的主要架构、损失函数、训练过程及采样方法。自 VAE 被提出以来，涌现出大量变体模型。在本节中，我们仅对其中几个著名的变体模型进行介绍。

3.8.1　CVAE

上文所介绍的 VAE 是无监督模型，其训练目的是对数据的边际分布 $p(\boldsymbol{x})$ 进行估计，因此它无法对生成样本的某些特定属性 $\boldsymbol{y}$ 进行有效控制。这些属性可以是类别标签、文字描述、图像等额外信息。

条件变分自编码器（Conditional Variational Autoencoder，CVAE）是 VAE 的变体模型，它可以对条件分布 $p(\boldsymbol{x}\,|\,\boldsymbol{y}) \triangleq p(\boldsymbol{x}\,|\,\boldsymbol{Y}=\boldsymbol{y})$ 进行估计，并根据给定条件 $\boldsymbol{y}$ 进行图像生成。将条件 $\boldsymbol{y}$ 输入 VAE 的方式有多种，此处我们介绍基于线性投影（Linear Projection）的方法。如图 3-8 所示，从数据集 $\mathcal{D}$ 中抽取一张图像 $\boldsymbol{x}$ 及其条件 $\boldsymbol{y}$，我们假设 $\boldsymbol{y}$ 是经过编码后的条件。例如，在具有 5 个类别的分类任务中，类别标签 $y=1$（第一类），可以被独热编码为向量 $\boldsymbol{y}=[1,0,0,0,0]^{\mathrm{T}}$。条件 $\boldsymbol{y}$ 被输入一个全连接网络 MLP，该网络的输出维度与隐变量 $\boldsymbol{Z}$ 一致，即 $\boldsymbol{v}=\mathrm{MLP}(\boldsymbol{y})$。该全连接网络的输出 $\boldsymbol{v}$ 将会按元素加在隐变量的取值 $\boldsymbol{z}$ 之后，即

$$\boldsymbol{z}+\boldsymbol{v}=\boldsymbol{\mu}_{\varphi,x}+\boldsymbol{\sigma}_{\varphi,x}\odot\boldsymbol{\varepsilon}+\boldsymbol{v} \tag{3-47}$$

求和之后得到 $\boldsymbol{z}+\boldsymbol{v}$ 再输入生成网络。基于式（3-47），我们可以对 CVAE 进行训练。在采样阶段，无论是采用先验采样还是后验采样，均可以通过控制条件 $\boldsymbol{y}$ 来控制生成图像的某些属性。

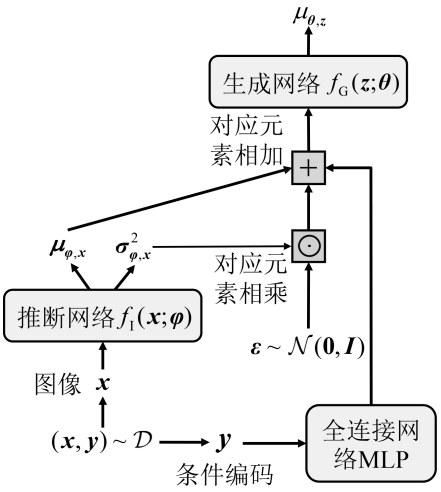


图 3-8　基于线性投影的方法

3.8.2 Beta-VAE

Beta-VAE 是一种基于 VAE 的深度学习模型，它在传统 VAE 的基础上进行了改进，以增强模型在潜在空间中的解耦（Disentanglement）能力。Beta-VAE 通过引入一个可调的 Beta 系数［见式（3-46）中的λ］来平衡重构误差和潜在空间的解耦合性，从而学习数据的低维表示，实现数据压缩、重构和生成等功能。

在通常情况下，Beta 系数是一个大于或等于 1 的正值，其大小取决于任务的性质及所需的潜在空间的表达能力。Beta-VAE 通过调整 Beta 系数来增强模型在潜在空间中的解耦能力，使潜在空间中的不同维度能够表示数据中相互独立的信息。若 Beta 系数等于 1［见式（3-46）中的$\lambda=1$］，则 Beta-VAE 退化为传统 VAE。

3.8.3 MHVAE

层级变分自编码器（Hierarchical Variational Autoencoder，HVAE）是 VAE 的一种扩展形式，旨在通过引入层级结构来更好地捕捉数据的潜在表示。在这种结构下，隐变量本身被解释为由其他更高层次、更抽象的隐变量生成。

在一般的具有 T 个层级结构的 HVAE 中，每个隐变量都被允许依赖其之前的所有隐变量。然而，我们关注一种特殊情况，将其命名为**马尔可夫层级变分自编码器**（Markovian HVAE，MHVAE）。在马尔可夫中，生成过程构成了一个马尔可夫链（见 3.1.5 节），换句话说，层级结构中的每步转移都遵循马尔可夫性质，即解码每个隐变量 $\boldsymbol{z}_t$ 时，它仅依赖其前一个隐变量 $\boldsymbol{z}_{t+1}$。马尔可夫性质及马尔可夫链请参见 3.1.5 节。从直观和视觉的角度来看，这可以简单地理解为将多个 VAE 依次堆叠在一起，如图 3-9 所示。描述这种模型的另一个恰当的术语是**递归变分自编码器**（Recursive VAE）。

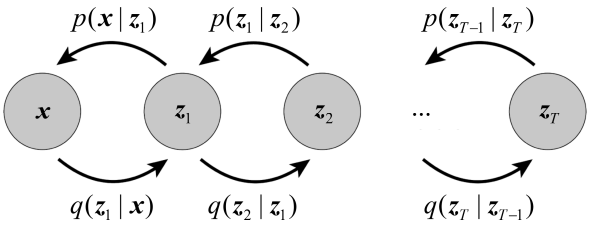


图 3-9　具有 T 层隐变量的 MHVAE

数学上，我们可以将 MHVAE 的联合分布和变分分布写为

$$p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z}_{1:T})=p_{\boldsymbol{\theta}}(\boldsymbol{z}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}\mid\boldsymbol{z}_1)\prod_{t=2}^{T}p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}\mid\boldsymbol{z}_t) \tag{3-48}$$

$$q_{\boldsymbol{\varphi}}(\boldsymbol{z}_{1:T}\mid\boldsymbol{x})=q_{\boldsymbol{\varphi}}(\boldsymbol{z}_1\mid\boldsymbol{x})\prod_{t=2}^{T}q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t\mid\boldsymbol{z}_{t-1}) \tag{3-49}$$

其中，$\boldsymbol{z}_{1:T}$ 为 $\boldsymbol{z}_1,\boldsymbol{z}_2,\cdots,\boldsymbol{z}_T$ 的缩写；$p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z}_{1:T})$ 为可观测变量 $\boldsymbol{X}$ 和所有隐变量的联合分布；$q_{\boldsymbol{\varphi}}(\boldsymbol{z}_{1:T}\mid\boldsymbol{x})$ 为变分分布。

MHVAE 的关键步骤是推导对数边际似然函数的 ELBO。类似 VAE，有

$$\begin{aligned}\ln p_\theta(\boldsymbol{x}) &= \ln \int p_\theta(\boldsymbol{x}, \boldsymbol{z}_{1:T}) \mathrm{d}\boldsymbol{z} \\ &= \ln \int \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z}_{1:T}) q_\varphi(\boldsymbol{z}_{1:T} \mid \boldsymbol{x})}{q_\varphi(\boldsymbol{z}_{1:T} \mid \boldsymbol{x})} \mathrm{d}\boldsymbol{z} \\ &= \ln E_{q_\varphi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\frac{p_\theta(\boldsymbol{x}, \boldsymbol{z}_{1:T})}{q_\varphi(\boldsymbol{z}_{1:T} \mid \boldsymbol{x})}\right] \\ &\geqslant E_{q_\varphi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z}_{1:T})}{q_\varphi(\boldsymbol{z}_{1:T} \mid \boldsymbol{x})}\right]\end{aligned} \tag{3-50}$$

为了简化式（3-50），我们将 $E_{\boldsymbol{Z}_1,\cdots,\boldsymbol{Z}_T \sim q_\varphi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}$ 中的随机向量 $\boldsymbol{Z}_i$ 省略，简写为 $E_{q_\varphi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}$，并将大写 $\boldsymbol{Z}_{1:T}$ 改写为小写 $\boldsymbol{z}_{1:T}$。将式（3-48）和式（3-49）代入式（3-50）可得

$$E_{q_\varphi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z}_{1:T})}{q_\varphi(\boldsymbol{z}_{1:T} \mid \boldsymbol{x})}\right] = E_{q_\varphi(\boldsymbol{z}_{1:T}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{z}_T) p_\theta(\boldsymbol{x} \mid \boldsymbol{z}_1) \prod_{t=2}^{T} p_\theta(\boldsymbol{z}_{t-1} \mid \boldsymbol{z}_t)}{q_\varphi(\boldsymbol{z}_1 \mid \boldsymbol{x}) \prod_{t=2}^{T} q_\varphi(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1})}\right] \tag{3-51}$$

式（3-51）将在后文介绍 DM 时被用到。

3.8.4　VQ-VAE

向量量化变分自编码器（Vector Quantized Variational Autoencoder，VQ-VAE）是一种结合了 VAE 和向量量化（Vector Quantization）技术的生成模型。它通过引入向量量化步骤，对潜在空间进行离散化，从而生成高质量的样本。VQ-VAE 结合了 VAE 的生成能力和向量量化的数据压缩优势，广泛应用于图像、音频生成及数据压缩等领域。

VQ-VAE 主要由编码器、向量量化器（Vector Quantizer）和解码器三部分组成。如图 3-10 所示，编码器将原始数据 $\boldsymbol{x}$ 映射到潜在空间中的连续向量 $\boldsymbol{z}_e(\boldsymbol{x})$ 上。向量量化器将编码器输出的连续向量 $\boldsymbol{z}_e(\boldsymbol{x})$ 映射到离散编码 $\boldsymbol{z}_q(\boldsymbol{x})$ 上，这通常是通过查找最接近的码本（Code Book）向量 $\boldsymbol{e}_k$ 完成的，其中 k 是码本向量的索引。离散编码 $\boldsymbol{z}_q(\boldsymbol{x})$ 和索引 k 需要根据如下向量量化公式获得：

$$\begin{aligned}\boldsymbol{z}_q(\boldsymbol{x}) &= \boldsymbol{e}_k \\ k &= \arg\min_j \| \boldsymbol{z}_e(\boldsymbol{x}) - \boldsymbol{e}_j \|_2^2\end{aligned} \tag{3-52}$$

式中，$\boldsymbol{e}_j$ 为码本中的第 j 个向量；$\boldsymbol{z}_e(\boldsymbol{x})$ 为编码器输出的连续向量；$\boldsymbol{z}_q(\boldsymbol{x})$ 为量化后的离散编码，通过查找与 $\boldsymbol{z}_e(\boldsymbol{x})$ 最接近的码本向量 $\boldsymbol{e}_k$ 得到。最后，解码器根据离散编码 $\boldsymbol{z}_q(\boldsymbol{x})$ 生成重构数据 $\hat{\boldsymbol{x}}$。

VQ-VAE 的损失函数通常包括重构损失、量化损失（也称为码本损失）和承诺损失。重构损失（Reconstruction Loss）L_{recon} 用于评估解码器生成的重构数据 $\hat{\boldsymbol{x}}$ 与原始数据 $\boldsymbol{x}$ 之间的差异，通常使用均方误差（Mean Square Error，MSE）或其他距离度量来表示。量化损失（Quantization Loss）L_{vq} 用于推动编码器输出的连续向量 $\boldsymbol{z}_e(\boldsymbol{x})$ 接近码本中的离散编码 $\boldsymbol{z}_q(\boldsymbol{x})$。由于量化过程是不可微的，通常使用直通估计器（Straight-Through Estimator）技巧来处理梯

度传播问题。量化损失可以表示为

$$L_{\text{vq}} = \| \text{sg}[\boldsymbol{z}_e(\boldsymbol{x})] - \boldsymbol{z}_q(\boldsymbol{x}) \|_2^2 \tag{3-53}$$

式中，sg[·]表示停止梯度运算，即在该部分不计算梯度。

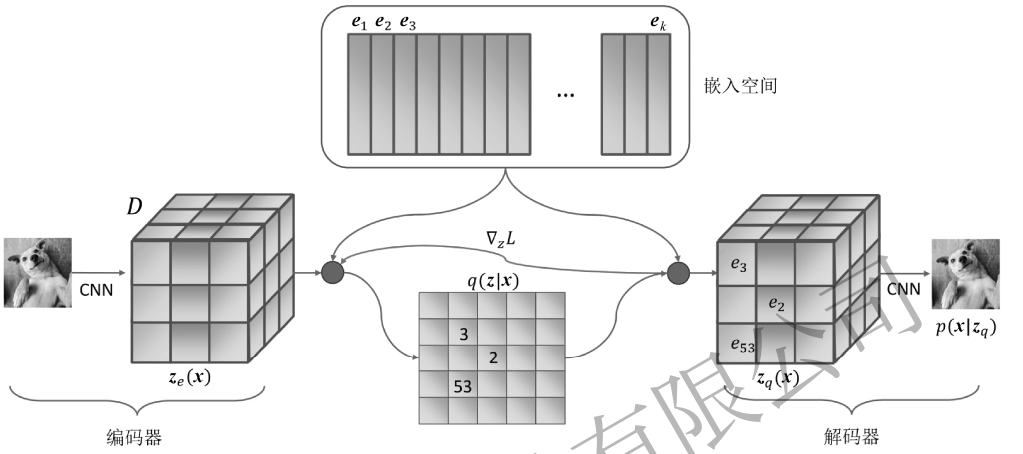


图 3-10　VQ-VAE 的结构示意图

承诺损失（Commitment Loss）L_{c} 用于防止编码器输出的向量在量化过程中发生过大变化，从而保持潜在编码的稳定性。承诺损失可以表示为

$$L_{\text{c}} = \beta \cdot \| \boldsymbol{z}_e(\boldsymbol{x}) - \text{sg}[\boldsymbol{z}_q(\boldsymbol{x})] \|_2^2 \tag{3-54}$$

式中，β 为超参数，用于平衡量化损失和承诺损失。于是，VQ-VAE 的总损失函数可以表示为

$$L = L_{\text{recon}} + L_{\text{vq}} + L_{\text{c}} \tag{3-55}$$

在训练过程中，VQ-VAE 通过最小化总损失函数来优化编码器、解码器和码本。由于量化过程是不可微的，因此需要使用直通估计器或其他技巧来处理梯度传播问题。具体来说，在反向传播过程中，量化损失的梯度会直接传递给编码器输出的连续向量 $\boldsymbol{z}_e(\boldsymbol{x})$，而承诺损失的梯度则会通过停止梯度运算传递给码本向量。

VQ-VAE 在图像生成、音频生成、数据压缩等领域具有广泛的应用前景。其优势在于通过引入向量量化步骤对潜在空间进行离散化，从而生成高质量且多样化的样本。此外，VQ-VAE 还可以与其他生成模型（如 GAN 等）结合使用，以进一步提升生成效果。

3.9　代码实践：VAE

在本节中，我们将在 Fashion MNIST 数据集上利用 PyTorch 深度学习框架训练 VAE。为此，我们将提供一系列必要的代码模块，涵盖数据预处理、推断网络与生成网络的构建、训练算法的实现，以及评价指标的计算。我们衷心希望，通过本节提供的代码，读者能够更深入地理解 VAE 背后的数学原理与机制。

3.9.1　数据载入与实验设置

Fashion MNIST 是一个广泛使用的图像分类数据集，由德国在线时尚零售商 Zalando 的研究部门创建。该数据集旨在成为经典 MNIST 数据集的替代品，为机器学习算法在图像分类任务上提供更具挑战性的测试平台。Fashion MNIST 数据集包含 70000 张灰度图像，这些图像被划分为 60000 个训练样本和 10000 个测试样本，每张图像均为 28 像素×28 像素的分辨率。这些图像覆盖了 10 个类别的服饰物品，包括上衣、裤子、套衫、连衣裙、外套、凉鞋、衬衫、运动鞋、包和踝靴。图 3-11 中展示了部分样本示例。

图 3-11　Fashion MNIST 数据集

Fashion MNIST 数据集之所以受欢迎，是因为它不仅继承了 MNIST 数据集易于获取和使用的优点，还在分类难度上有所提升。与手写数字相比，服饰物品的分类涉及更多的视觉特征，如形状、颜色、纹理等，这对机器学习算法提出了更高的要求。此外，Fashion MNIST 数据集的大小适中，既不会让训练过程过于漫长，又能充分展现算法的性能差异，因此非常适合作为入门级数据集供初学者实践和学习。

如代码清单 3-1 所示，为了在 Fashion MNIST 数据集上训练 VAE，我们先载入必要的包。

代码清单 3-1　载入必要的包

```
## 载入必要的包
import torch
import torch.utils.data
from torch import nn, optim
from torchvision import datasets, transforms
from torch.nn import functional as F
from torchvision.utils import save_image
import numpy as np
import matplotlib.pyplot as plt
```

之后，如代码清单 3-2 所示，进行必要参数的设置，以及数据的载入。需要注意的是，在该实验中，我们将隐变量 $\boldsymbol{Z}$ 的维度设置为 64，并将图像的像素值归一化至[−1,1]。

代码清单 3-2　参数设置及数据载入

```
## 超参数设定
BATCH_SIZE=256  #批大小
LATENT_DIM=64  #隐变量 Z 的维度
IMG_SIZE=28  #图像维度
LR=1e-4  #学习率
WEIGHT_DECAY=1e-4  #权重衰减参数，参见 4.4.3 节
```

```
ALPHA=1e-4  #式（3-46）中 lambda 的值
EPOCHS=400  #训练轮数，与循环次数成正比

## 设定设备，若有 GPU，则用 GPU 计算，否则用 CPU 计算
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# 数据载入与预处理
transform = transforms.Compose([
    transforms.ToTensor()  #将像素值从[0,255]归一化为[0,1]
    transforms.Normalize((0.5,), (0.5,)) #将像素值从[0,1]归一化为[-1,1]
])
train_dataset = datasets.FashionMNIST(root='./data', train=True, download=True,
transform=transform)
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=BATCH_SIZE,
shuffle=True)
```

3.9.2 推断网络和生成网络

接下来，我们将定义一个 VAE 的 Python 类，如代码清单 3-3 所示。这个 Python 类将实现包括推断网络（又称编码器）和生成网络（又称解码器）在内的神经网络结构，同时涵盖编码过程、解码过程及重参数化技巧。此外，我们还将定义模型训练完成后生成样本的过程。

推断网络的主体架构是一个卷积神经网络，其中每个卷积块依次包含一个步长卷积层、一个批处理层及一个 LeakyReLU 激活函数。每个步长卷积层都会将输入特征图的维度减半，同时将通道数翻倍（除了首个卷积层）。在卷积模块之后，我们利用两个全连接层，将卷积模块输出的特征图转换为 $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})$ 的均值和标准差对数的预测值。

生成网络的主体架构包含一个全连接层和一个转置卷积模块。全连接层的输入是一个隐变量 $\boldsymbol{z}$，输出是一个 512 维的向量。这个向量随后被重塑成一个 512×1×1 的数组。接着该数组被输入转置卷积模块，经过一系列升维操作，最终输出一张具有 28×28 维度的图像。值得注意的是，转置卷积模块的最后一个卷积层采用了步长卷积。并且，网络的最后使用了 Tanh() 激活函数，以确保输出的图像值域范围为[−1,1]。

代码清单 3-3　定义 VAE 的 Python 类

```
class VAE(nn.Module):
    def __init__(self, img_size, latent_dim):
        super(VAE, self).__init__()
        # 输入图像的通道数、高度、宽度
        self.in_channel, self.img_h, self.img_w = img_size
        # 隐藏编码 Z 的维度
        self.latent_dim = latent_dim

        ## 开始构建推断网络
        # 卷积模块
        self.encoder = nn.Sequential(
```

```
            ## 输入维度 (n,1,28,28)
            nn.Conv2d(in_channels=self.in_channel, out_channels=32, kernel_size=3, stride=2, padding=1),#h=h//2
            nn.BatchNorm2d(32),
            nn.LeakyReLU(),
            ## 特征图维度 (n,32,14,14)
            nn.Conv2d(32, 64, 3, 2, 1),#h=h//2
            nn.BatchNorm2d(64),
            nn.LeakyReLU(),
            ## 特征图维度 (n,64,7,7)
            nn.Conv2d(64, 128, 4, 1, 0),#h=h-3
            nn.BatchNorm2d(128),
            nn.LeakyReLU(),
            ## 特征图维度 (n,128,4,4)
            nn.Conv2d(128, 256, 3, 2, 1),#h=h//2
            nn.BatchNorm2d(256),
            nn.LeakyReLU(),
            ## 特征图维度 (n,256,2,2)
            nn.Conv2d(256, 512, 1, 1, 0),#h=h
            nn.BatchNorm2d(512),
            nn.LeakyReLU(),
            ## 特征图维度 (n,512,2,2)
        )
        # 全连接层：将特征向量转化为分布均值 mu
        self.fc_mu = nn.Linear(512*4, self.latent_dim) #
        # 全连接层：将特征向量转化为分布方差的对数 ln(var)
        self.fc_var = nn.Linear(512*4, self.latent_dim)

        ## 开始构建生成网络
        # 全连接层
        self.decoder_input = nn.Linear(self.latent_dim, 512*4)
        # 转置卷积模块
        self.decoder = nn.Sequential(
            ## 输入维度 (n,512,2,2)
            nn.ConvTranspose2d(in_channels=512, out_channels=256, kernel_size=3, stride=2, padding=1, output_padding=1),#h=2h
            nn.BatchNorm2d(256),
            nn.LeakyReLU(),
            ## 特征图维度 (n,256,4,4)
            nn.ConvTranspose2d(256, 128, 4, 1, 0, 0),#h=h+3
            nn.BatchNorm2d(128),
            nn.LeakyReLU(),
            ## 特征图维度 (n,128,7,7)
            nn.ConvTranspose2d(128, 64, 3, 2, 1, 1),#h=2h
```

```
            nn.BatchNorm2d(64),
            nn.LeakyReLU(),
            ## 特征图维度 (n,64,14,14)
            nn.ConvTranspose2d(64, 32, 3, 2, 1, 1),#h=2h
            nn.BatchNorm2d(32),
            nn.LeakyReLU(),
            ## 特征图维度 (n,32,28,28)
            nn.ConvTranspose2d(32, 32, 1, 1, 0, 0),#h=h
            nn.BatchNorm2d(32),
            nn.LeakyReLU(),
            ## 特征图维度 (n,32,28,28)
            nn.Conv2d(in_channels=32, out_channels=self.in_channel, kernel_size=1, stride=1, padding=0), #h=h
            nn.Tanh()
            ## 输出图像维度 (n,1,28,28)
        )

    # 定义编码过程
    def encode(self, x):
        result = self.encoder(x)  # encoder 结构,(n,1,28,28)-->(n,512,2,2)
        result = torch.flatten(result, 1)  # 将特征层转化为特征向量,(n,512,2,2)-->(n,512*4)
        mu = self.fc_mu(result)  # 计算分布均值 mu, (n,512*4)-->(n,128)
        ln_var = self.fc_var(result)  # 计算分布方差的对数 ln(var), (n,512*4)-->(n,128)
        return [mu, ln_var]

    # 定义解码过程
    def decode(self, z):
        x_hat = self.decoder_input(z).view(-1, 512, 2, 2)  # 将采样变量 Z 先转化为特征向量，再转化为特征层,(n,128)-->(n,512*4)-->(n,512,2,2)
        x_hat = self.decoder(x_hat)  # decoder 结构,(n,512,2,2)-->(n,1,28, 28)
        return x_hat  # 返回生成样本

    # 重参数化技巧
    def reparameterize(self, mu, ln_var):
        std = torch.exp(0.5 * ln_var)  # 分布标准差 std
        eps = torch.randn_like(std)  # 从标准正态分布中采样,(n,128)
        return mu + eps * std  # 返回对应正态分布中的采样值

    # 前传函数
    def forward(self, x):
        mu, ln_var = self.encode(x)  # 经过编码过程，得到分布的均值和方差对数
        z = self.reparameterize(mu, ln_var)  # 经过重参数化技巧，得到隐变量 Z
```

```
        x_hat = self.decode(z)  # 经过解码过程，得到生成样本 x_hat
        return [x_hat, x, mu, ln_var]

    # 定义生成过程
    def sample(self, n, device):
        z = torch.randn(n, self.latent_dim).to(device)  # 从标准正态分布中采样得到 n 个隐变量 Z，长度为 latent_dim
        images = self.decode(z)  # 经过解码过程，得到生成样本 Y
        return images
```

3.9.3　模型训练

在代码清单 3-4 中，我们将代码清单 3-3 中定义的 VAE 的 Python 类实例化，并定义优化器。实例化的参数如 image_size 和 latent_dim，以及优化器的参数 lr 和 weight_decay 可由代码清单 3-2 确定。

代码清单 3-4　VAE 的 Python 类实例化及优化器的定义

```
# 实例化模型
model=VAE(img_size=[1,IMG_SIZE,IMG_SIZE], latent_dim=LATENT_DIM).to(device)
# 定义优化器
optimizer=optim.Adam(model.parameters(), lr=LR, weight_decay=WEIGHT_DECAY)
```

损失函数与训练函数可由代码清单 3-5 定义。

代码清单 3-5　定义损失函数与训练函数

```
# 定义损失函数；参照式（3-46）
def loss_function(x, x_hat, mean, ln_var, alpha=ALPHA):
    reproduction_loss = F.mse_loss(x_hat, x)
    term1 = 0.5*torch.mean(torch.sum(torch.pow(ln_var.exp(), 2), dim=1))
    term2 = 0.5*torch.mean(torch.sum(torch.pow(mean, 2), dim=1))
    term3 = -0.5*torch.mean(2*torch.sum(ln_var,dim=1))
    # 式（3-38）中的 D 为常数，不影响优化，因此可以忽略
    KLD = term1 + term2 + term3
    return reproduction_loss + alpha * KLD

# 定义训练函数
def train(model, optimizer, epochs, device):

    for epoch in range(epochs):
        model.train()
        overall_loss = 0
        for batch_idx, (x, _) in enumerate(train_loader):
            x = x.to(device)
            batch_size = x.size(0)
            optimizer.zero_grad()
```

```
            x_hat, _, mean, ln_var = model(x)
            loss = loss_function(x, x_hat, mean, ln_var)
            overall_loss += loss.item()
            loss.backward()
            optimizer.step()

        print("\tEpoch", epoch+1, "\tAverage Loss: ", overall_loss/(batch_idx*
    batch_size))

        # 每 10 个 epoch 生成 100 个样本用于可视化
        if (epoch+1)%10==0:
            model.eval()
            with torch.no_grad():
                sample_images = model.sample(100, device)  #生成 100 个样本
                sample_images = sample_images.detach().cpu()
                save_file = "./output/{}.png".format(epoch+1)
                os.makedirs(os.path.dirname(save_file), exist_ok=True)
                save_image(sample_images.data, save_file, nrow=10, normalize=True)

    return
```

通过运行代码清单 3-6，我们可以执行训练过程。

代码清单 3-6　执行训练过程

```
train(model, optimizer, epochs=EPOCHS, device=device)
```

图 3-12 展示了 VAE 在训练过程中，损失函数随训练轮次变化的趋势图。从图中可以观察到，VAE 逐渐趋于收敛。

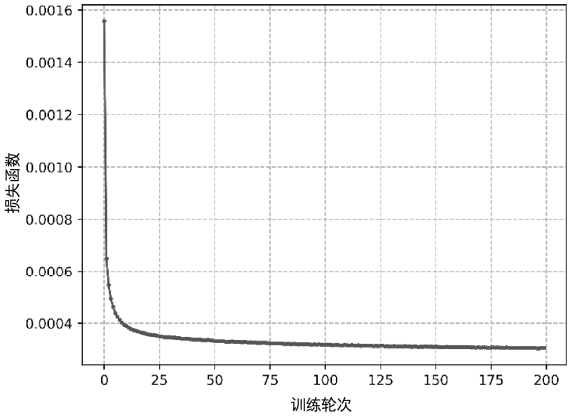


图 3-12　VAE 的损失函数随训练轮次变化的趋势图

3.9.4　采样与可视化

训练完成后，我们依据 3.7.2 节所述的先验采样，从已训练好的生成网络中随机抽取 25 张

图像，并进行可视化处理，如代码清单 3-7 所示。如图 3-13 所示，这些生成的样本与图 3-11 中的真实样本相似。这一结果也间接验证了 VAE 在 Fashion MNIST 数据集上的有效性。

代码清单 3-7　采样与可视化

```
# 采样与可视化
model.eval()  # 将 VAE 设置为推理模式
sample_images = model.sample(25, device) # 随机采样 25 张图像
sample_images = sample_images.detach().cpu()

sample_images = [image[0][i,0] for i in range(len(sample_images))]
fig = plt.figure(figsize=(5, 5))
grid = ImageGrid(fig, 111, nrows_ncols=(5, 5), axes_pad=0.1)

for ax, im in zip(grid, sample_images):
    ax.imshow(im, cmap='gray')
    ax.axis('off')
plt.savefig("./fake_imgs.png", dpi=500,bbox_inches="tight")
```

图 3-13　VAE 生成样本示例（先验采样）

3.10　习　　题

1．AE 和 VAE 的主要区别是什么？AE 是生成模型吗？

2．如何用样本均值去近似 AE 损失函数的数学期望［见式（3-1）］？

3．在重参数化技巧中，式（3-18）中的 $h(\cdot)$ 和 $p_{\alpha}(\boldsymbol{z})$ 分别对应式（3-33）的重构项中的哪些部分？

4．将式（3-36）中的 $\boldsymbol{\varepsilon}$ 看作随机向量，求 $\boldsymbol{z}$ 的期望和方差。

5．利用马尔可夫性质推导式（3-48）和式（3-49）。

3.11 参考文献

[1] 邱锡鹏. 神经网络与深度学习[M]. 北京：机械工业出版社，2020.

[2] KINGMA D P, WELLING M. Auto-encoding variational Bayes[C]//International Conference on Learning Representations, 2014.

[3] KINGMA D P, SALIMANS T, JOZEFOWICZ R, et al. Improved variational inference with inverse autoregressive flow[J]. Advances in Neural Information Processing Systems, 2016, 29.

[4] MATVEICHEV D. How to sample from latent space with variational autoencoder[DB/OL]. Hackernoon Website, 2024.

[5] MURPHY K P. Probabilistic machine learning: Advanced topics[M]. Cambridge: MIT Press, 2023.

[6] WASSERMAN L. All of statistics: A concise course in statistical inference[M]. Springer Science & Business Media, 2013.

[7] SØNDERBY C K, RAIKO T, MAALØE L, et al. Ladder variational autoencoders[J]. Advances in Neural Information Processing Systems, 2016, 29.

[8] VAN DEN OORD A, VINYALS O, KAVUKCUOGLU K, et al. Neural discrete representation learning[J]. Advances in Neural Information Processing Systems, 2017, 30.

[9] XIAO H, RASUL K, VOLLGRAF R. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms[J]. arXiv preprint arXiv:1708.07747, 2017.