

Lec05&06 - Latent Variable Models (LVM) & Variational Autoencoder (VAE)

Motivation

核心动机：图像数据中存在大量潜在的变异因素（如性别、眼睛颜色、头发颜色、姿势等），但这些因素在未标注数据中未被显式记录。Latent Variable Models (LVMs) 通过引入**潜在变量** z 显式建模这些隐含因素，从而简化复杂数据分布的建模。

关键点解析：

1. 数据复杂性的分解：

- 直接建模高维数据 $p(x)$ （如图像像素）非常困难，因为其分布可能极其复杂。
- 若通过潜在变量 z 分解问题，条件分布 $p(x|z)$ 可能更简单（例如，假设 z 包含“眼睛颜色”信息，则 $p(x|z)$ 只需生成对应颜色的图像区域）

2. 无监督表示学习：

- 训练完成后，模型可通过后验分布 $p(z|x)$ 推断潜在特征（如计算 $p(\text{EyeColor} = \text{Blue}|x)$ ，实现无监督的特征提取。

3. 挑战：

- 手动设计 $p(x|z)$ 和 $p(z)$ 的显式形式几乎不可行，尤其是当潜在变量维度高时。
- 后验推断 $p(z|x)$ 的计算通常难以解析求解（需变分近似或蒙特卡洛方法）。

Deep LVM

核心思想：利用深度神经网络参数化条件分布 $p(x|z)$ ，结合潜在变量的生成能力，构建灵活且高容量的生成模型。

模型结构：

- 潜在变量生成：** $z \sim \mathcal{N}(0, I)$ ，假设潜在变量服从标准正态分布（或其他简单先验）。
- 条件分布建模：** $p(x|z) = \mathcal{N}(\mu_\theta(z), \Sigma_\theta(z))$ ，其中均值 μ_θ 和协方差 Σ_θ 由神经网络生成（例如，多层感知机或卷积网络）。
- 目标：**训练后，潜在变量 z 应编码有意义的语义特征（如“微笑程度”“头发颜色”），实现无监督的表示学习。

混合高斯模型(MoG)

基本原理

1. 模型定义

混合高斯模型（MoG）是一种浅层潜在变量模型，通过组合多个高斯分布来建模复杂的数据分布。其核心思想是：

- 潜在变量 z ：**表示数据点所属的混合成分（即聚类簇），服从分类分布：

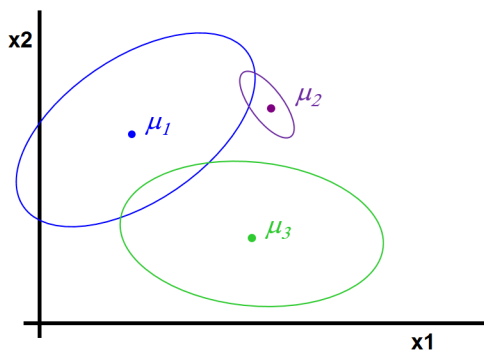
$$z \sim \text{Categorical}(1, \dots, K)$$

- 观测变量 x ：在给定潜在变量 $z = k$ 的条件下，服从第 k 个高斯分布：

$$p(x|z = k) = \mathcal{N}(x; \mu_k, \Sigma_k)$$

- 整体数据分布的边际概率为：

$$p(x) = \sum_{k=1}^K p(z = k) \mathcal{N}(x; \mu_k, \Sigma_k)$$



2. 生成过程

生成数据点的步骤如下：

1. 选择混合成分：从分类分布中采样 z ，确定数据点属于第 k 个高斯成分。
2. 生成观测值：从对应的高斯分布 $\mathcal{N}(\mu_k, \Sigma_k)$ 中采样 x 。

在无监督学习中的应用

1. 聚类 (Clustering)：后验概率 $p(z = k|x)$ 直接给出数据点的软聚类结果。
2. 密度估计 (Density Estimation)：混合高斯通过多个局部高斯分布的叠加，灵活拟合复杂数据分布。
3. 无监督特征学习：潜在变量 z 可视为数据的低维表示，但混合高斯的浅层结构限制了其表达能力。

与深度生成模型的对比

1. 混合高斯 vs. 变分自编码器 (VAE)

- 表达能力：
 - 混合高斯：浅层模型，仅通过线性组合高斯分布建模数据。
 - VAE：深层模型，利用神经网络参数化 $p(x|z)$ ，可生成高度非线性的复杂分布（如图像）。
- 潜在变量：
 - 混合高斯： z 为离散类别，解释性有限。
 - VAE： z 为连续向量，支持语义插值（如从“微笑”到“不微笑”的渐变）。

2. 应用场景

- 混合高斯：适用于低维数据聚类、简单密度估计（如金融数据分群）。
- VAE：适用于高维数据生成（如图像、文本）、无监督特征学习（如提取潜在语义）。

边际似然(Marginal Likelihood)

定义与核心思想

边际似然是指在存在潜在变量（未观测变量）时，观测数据的概率。具体来说，假设我们有一个联合分布模型 $p(\mathbf{X}, \mathbf{Z}; \theta)$ ，其中：

- \mathbf{X} 是观测变量（例如图像中可见的像素）
- \mathbf{Z} 是潜在变量（例如图像的分类、缺失的像素值）
- θ 是模型参数



边际似然的目标是计算仅观测到 $\mathbf{X} = \bar{x}$ 时的概率，即对所有可能的潜在变量 \mathbf{Z} 进行边缘化：

$$p(\mathbf{X} = \bar{x}; \theta) = \sum_z p(\mathbf{X} = \bar{x}, \mathbf{Z} = z; \theta) \text{ (离散变量)}$$

$$p(\mathbf{X} = \bar{x}; \theta) = \int_{\mathbf{Z}} p(\mathbf{X} = \bar{x}, \mathbf{Z} = z; \theta) dz \text{ (连续变量)}$$

应用场景

- **图像补全**：若图像的上半部分像素缺失（即 \mathbf{X} 为下半部分， \mathbf{Z} 为上半部分），边际似然需考虑所有可能的补全方式。
- **聚类分析**：在混合高斯模型中， \mathbf{Z} 表示数据点所属的簇，边际似然对应数据点的总概率。

边际似然的意义

简单来说，边际似然函数衡量的是“模型生成的数据有多像真实数据”。

- 真实数据是你那批图像的像素点集合，记为 \mathbf{x} 。
- VAE作为一个生成模型，它试图学会一个概率分布 $p(\mathbf{x})$ ，表示“生成这些图像的可能性”。

为什么是这个目标？

1. **生成模型的本质**：VAE是一个生成模型，它的目标不是简单地记住训练数据，而是学会数据的概率分布 $p(\mathbf{x})$ ，然后从中采样生成新数据。最大化 $\log p(\mathbf{x})$ 就是在逼近这个真实分布。举个例子：如果你的图像是猫咪，最大化对数似然函数就是在让模型学会“猫咪图像的分布规律”，而不是只记住某几张特定的猫咪照片。
2. **从数据中“提炼规律”**：你的训练集（图像）只是真实世界数据的一个样本，背后有个更大的“真实分布”（比如所有可能的猫咪图像）。最大化 $\log p(\mathbf{x})$ 是在用有限的训练数据去估计这个真实分布，让模型不仅能重现训练集，还能生成没见过但合理的新图像。
3. **连接到潜在空间**：VAE通过编码器把 \mathbf{x} 映射到 \mathbf{z} ，然后通过解码器从 \mathbf{z} 生成 \mathbf{x} 。最大化 $\log p(\mathbf{x})$ 就是在优化整个过程（编码 + 解码），让潜在空间 \mathbf{z} 既能捕捉特征，又能生成符合训练数据的样本。

为什么部分观测数据下的参数学习具有挑战性？

1. 计算复杂性

- **边际似然的计算**：需要对所有可能的潜在变量值求和或积分。例如：
 - 若 \mathbf{Z} 是30维二元变量，求和项数为 2^{30} ，计算量爆炸。
 - 若 \mathbf{Z} 是连续变量，积分无解析解，需近似方法（如蒙特卡洛采样）。

2. 似然函数的非分解性

- **完全观测数据**：似然函数可分解为各变量条件概率的乘积，例如在贝叶斯网络中：

$$p(x_1, \dots, x_n) = \prod_i p(x_i | x_{pa(i)})$$

- **部分观测数据**：由于潜在变量 Z 的存在，似然函数无法分解，导致计算和优化困难。

3. 多峰性与局部最优

- **多峰性**：边际似然函数可能存在多个局部最大值，梯度下降法容易陷入局部最优。
- **初始化敏感**：例如在EM算法中，初始参数选择显著影响最终结果。

4. 梯度估计困难

- **蒙特卡洛梯度估计**：即使采用蒙特卡洛方法近似梯度，潜在变量的高方差会导致梯度估计不稳定。
- **高方差问题**：例如在混合高斯模型中，若某些簇的后验概率极低，采样效率低下。

重要性采样

1. **基本思想**：重要性采样是一种蒙特卡洛方法，用于高效估计难以直接计算的期望值或积分。其核心是通过引入一个**提议分布(Proposal Distribution)** $q(\mathbf{z})$ ，对高概率区域进行加权采样，从而减少估计的方差。

2. **数学形式**

- 引入 $q(\mathbf{z})$

$$\begin{aligned} p_{\theta}(\mathbf{x}) &= \sum_{\text{All possible value of } \mathbf{z}} p_{\theta}(\mathbf{x}, \mathbf{z}) \\ &= \sum_{\mathbf{z} \in \mathcal{Z}} \frac{q(\mathbf{z})}{q(\mathbf{z})} p_{\theta}(\mathbf{x}, \mathbf{z}) \\ &= \mathbf{E}_{\mathbf{z} \sim q(\mathbf{z})} \left[\frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \right] \\ &\approx \frac{1}{k} \sum_{j=1}^k \frac{p_{\theta}(\mathbf{x}, \mathbf{z}^{(j)})}{q(\mathbf{z}^{(j)})} \quad \mathbf{z}^{(j)} \sim q(\mathbf{z}) \end{aligned}$$

- **步骤**：

1. 从 $q(\mathbf{z})$ 中采样 k 个潜在变量 $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(j)}$
2. 计算每个样本的权重 $w_j = \frac{p_{\theta}(\mathbf{x}, \mathbf{z}^{(j)})}{q(\mathbf{z}^{(j)})}$
3. 用样本平均近似 $p_{\theta}(\mathbf{x})$

3. **与朴素蒙特卡洛的对比**

- **朴素蒙特卡洛**：均匀采样 z ，但高维空间中大部分样本对期望贡献极小（低效）。
- **重要性采样**：通过 $q(\mathbf{z})$ 聚焦高概率区域，加权后提升估计效率。

对数似然的估计问题

目标：训练模型时需要最大化对数似然函数 $\log p_{\theta}(x)$

直接估计的问题：对蒙特卡洛估计值取对数会导致偏差：

$$\log p_{\theta}(x) \approx \log \left(\frac{1}{k} \sum_{j=1}^k \frac{p_{\theta}(\mathbf{x}, \mathbf{z}^{(j)})}{q(\mathbf{z}^{(j)})} \right)$$

关键矛盾：期望的对数不等于对数的期望，即：

$$\mathbf{E}_{\mathbf{z} \sim q(\mathbf{z})} \left[\log \left(\frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \right) \right] \neq \log \left(\mathbf{E}_{\mathbf{z} \sim q(\mathbf{z})} \left[\frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \right] \right)$$

证明：由于对数函数具有凹性，因此利用Jenson不等式可知： $\log \mathbf{E}[X] \geq \mathbf{E}[\log(X)]$

后果：

- 若仅用单样本 $z^{(1)}$ 近似，估计值偏差较大。
- 即使使用多个样本，对数似然的估计仍然存在高方差或系统性偏差。

单样本近似的局限性：假设我们仅仅取单个样本来进行近似，则

$$\log p_{\theta}(x) \approx \log \left(\frac{p_{\theta}(\mathbf{x}, \mathbf{z}^{(1)})}{q(\mathbf{z}^{(1)})} \right) \quad \mathbf{z}^{(1)} \sim q(\mathbf{z})$$

这种简化假设 $\frac{p_{\theta}(\mathbf{x}, \mathbf{z}^{(1)})}{q(\mathbf{z}^{(1)})}$ 能够准确代表整体的期望值，但是实际上，若 $q(\mathbf{z})$ 与真实后验 $p(\mathbf{z}|\mathbf{x})$ 差异较大，单样本权重 w_1 可能远离真实值。导致对数似然估计不稳定，尤其在 $q(\mathbf{z})$ 选择不当时。

提议分布 $q(\mathbf{z})$ 的选择：

- 理想情况： $q(\mathbf{z})$ 应接近真实后验 $p(\mathbf{z}|\mathbf{x})$ ，以减少方差
- 实际挑战：真实后验未知，通常需要**变分推断**或自适应重要性采样。

证据下界

由于期望的对数 $\log \mathbf{E}[X]$ 很难计算，我们可以尝试计算其下界，即对数的期望 $\mathbf{E}[\log(X)]$ 。

我们将这个下界称为**证据下界(ELBO)**，因此我们的目标改为最大化对数似然的证据下界

变分

变分的概念

1. 在数学中，“变分”（Variational）一词来源于**变分法**（Calculus of Variations），它是研究如何寻找泛函（函数的函数）极值的一门学科。**核心思想**是通过调整函数的形状或参数，找到使得某个目标函数（如时间、能量）达到极值的函数。
2. 在机器学习和概率模型中，“变分”通常指**变分推断**（Variational Inference），其核心是用简单的**参数化分布近似复杂分布**，并通过优化参数使近似分布尽可能接近真实分布。
3. 在变分自编码器（VAE）中，“变分”体现在以下两方面：

(1) 变分推断

- **问题：**计算真实后验 $p(\mathbf{z}|\mathbf{x})$ 困难（因潜在变量高维且模型复杂）。
- **解决：**用神经网络（编码器）参数化近似分布 $q_{\phi}(\mathbf{z}|\mathbf{x})$ ，通过优化使 $q_{\phi}(\mathbf{z}|\mathbf{x})$ 逼近 $p(\mathbf{z}|\mathbf{x})$ 。

(2) 变分下界（ELBO）

- **目标：**最大化数据的对数似然 $\log p(\mathbf{x})$ ，但因边缘化困难，转而优化其下界
- **“变分”的体现：**通过调整 ϕ （编码器参数）和 θ （解码器参数），最大化ELBO，间接提升对数似然。

4. 直观类比：真实后验 $p(\mathbf{z}|\mathbf{x})$ 是一座复杂的山峰，直接攀登（计算）困难。变分推断则是用一块可塑的橡皮泥（参数化分布 $q(\mathbf{z}; \phi)$ ），不断调整形状（优化 ϕ ），使其尽可能贴合山峰的轮廓。

变分下界

我们将目标函数的证据下界进行展开可以得到：

$$\begin{aligned}\log p(\mathbf{x}; \theta) &\geq \mathbf{E}_{\mathbf{z} \sim q(\mathbf{z})} \left[\log \left(\frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \right) \right] \\&= \sum_{\mathbf{z}} q(\mathbf{z}) \log \left(\frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \right) \\&= \sum_{\mathbf{z}} q(\mathbf{z}) \log (p_{\theta}(\mathbf{x}, \mathbf{z})) - \sum_{\mathbf{z}} q(\mathbf{z}) \log q(\mathbf{z}) \\&= \sum_{\mathbf{z}} q(\mathbf{z}) \log (p_{\theta}(\mathbf{x}, \mathbf{z})) + H(q)\end{aligned}$$

其中 $H(q)$ 是概率分布 q 的熵，定义为 $H(q) = -\sum_{\mathbf{z}} q(\mathbf{z}) \log q(\mathbf{z})$

如果存在理想情况，即 $q(\mathbf{z})$ 等于真实后验 $p(\mathbf{z}|\mathbf{x})$ ，此时取等。

于是我们希望让 $q(\mathbf{z})$ 尽可能贴近真实后验分布，我们可以用KL散度来衡量

$$\begin{aligned}D_{KL}(q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}; \theta)) &= \sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x}; \theta)} \\&= \sum_{\mathbf{z}} q(\mathbf{z}) \log q(\mathbf{z}) - \sum_{\mathbf{z}} q(\mathbf{z}) \log (p(\mathbf{z}|\mathbf{x}; \theta)) \\&= \sum_{\mathbf{z}} q(\mathbf{z}) \log q(\mathbf{z}) - \sum_{\mathbf{z}} q(\mathbf{z}) \log \left(\frac{p(\mathbf{z}, \mathbf{x}; \theta)}{p(\mathbf{x}; \theta)} \right) \\&= -\sum_{\mathbf{z}} q(\mathbf{z}) \log (p(\mathbf{z}, \mathbf{x}; \theta)) + \log p(\mathbf{x}; \theta) - H(q) \\&\geq 0\end{aligned}$$

(这同样可以用来证明变分下界的存在。)

当 $D_{KL}(q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}; \theta)) = 0$ 的时候，可以取等。

后验分布的变分推断

实际上在变分推断中的后验分布 $p(\mathbf{z}|\mathbf{x}; \theta)$ 很难计算，因为其实际意义是：给定了一个观测数据 \mathbf{X} ，需要试图找到哪些 \mathbf{Z} 可能是产生这个 \mathbf{X} 的真实分布。我必须以概率的方式把整个神经网络翻转过来。

所以我们可以定义一系列潜变量的分布，并通过一组变分参数 ϕ 来进行参数化。

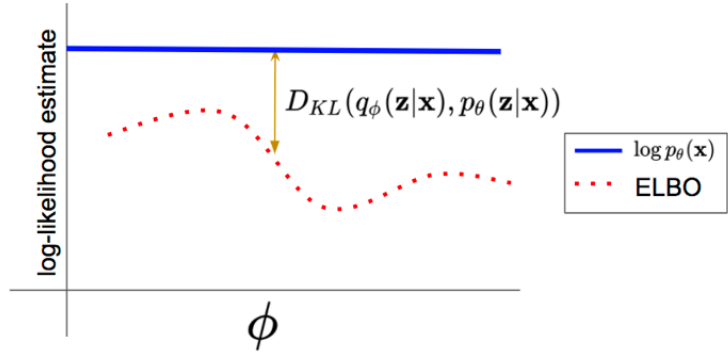
进行假设：假设 $q(\mathbf{z}; \phi)$ 是一种易处理的概率分布，通过变分参数 ϕ 来进行定义，例如 $q(\mathbf{z}; \phi) = \mathcal{N}(\phi_1, \phi_2)$

变分推断：不断调整 ϕ 使得 $q(\mathbf{z}; \phi)$ 尽可能接近真实后验分布 $p(\mathbf{z}|\mathbf{x}; \theta)$

将这个变分参数 ϕ 回代入原式可得：

$$\begin{aligned}\log p(\mathbf{x}; \theta) &\geq \sum_{\mathbf{z}} q(\mathbf{z}; \phi) \log (p(\mathbf{z}, \mathbf{x}; \theta)) + H(q(\mathbf{z}; \phi)) = \mathcal{L}(\mathbf{x}; \theta, \phi) \\ \log p(\mathbf{x}; \theta) &= \mathcal{L}(\mathbf{x}; \theta, \phi) + D_{KL}(q(\mathbf{z}; \phi) \parallel p(\mathbf{z}|\mathbf{x}; \theta))\end{aligned}$$

变分推断越接近， D_{KL} 越小，证据下界ELBO和真实的函数 $\log p(\mathbf{x}; \theta)$ 越接近。



最终结果：会有两个神经网络，一个解码器和一个编码器，解码器主要由 θ 控制，编码器主要由 ϕ 控制。

推广至多样本情况：学习目标是：最大化整个数据集的似然函数

$$l(\theta; \mathcal{D}) = \sum_{\mathbf{x}^i \in \mathcal{D}} \log p(\mathbf{x}^i; \theta) \geq \sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$$

注意：此处我们需要对于不同的样本采取不同的编码方式，也就是对于每一个 \mathbf{x}^i ，都有一个 ϕ^i 与其对应。

$$\max_{\theta} l(\theta; \mathcal{D}) \geq \max_{\theta, \phi^1, \dots, \phi^M} \sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$$

利用随机变分推断(Stochastic Variational Inference SVI)进行学习

采用SGD方法对整个数据集的似然函数的变分下界进行优化，即 $\sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$

$$\begin{aligned} \sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\mathbf{x}^i; \theta, \phi) &= \sum_{\mathbf{z}} q(\mathbf{z}; \phi) \log(p(\mathbf{z}, \mathbf{x}^i; \theta)) + H(q(\mathbf{z}; \phi^i)) \\ &= \mathbf{E}_{q(\mathbf{z}; \phi^i)} [\log p(\mathbf{z}, \mathbf{x}^i; \theta) - \log q(\mathbf{z}; \phi^i)] \end{aligned}$$

1. 对参数 $\theta, \phi^1, \dots, \phi^M$ 进行初始化。
2. 从数据集 \mathcal{D} 中随机采样 \mathbf{x}^i
3. 将变分下界视为关于 ϕ^i 的函数对其求偏导后优化。
 - $\phi^i = \phi^i + \eta \nabla_{\phi^i} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$ ，由于优化目标是最大化而不是最小化，所以这里要用加号。
 - 直到向 $\phi^{i,*} \approx \arg \max_{\phi} \mathcal{L}(\mathbf{x}^i; \theta, \phi)$
4. 将变分下界视为关于 θ 的函数对其求偏导后优化。
5. 回到步骤2，重新采样进行优化。

我们可以将对 ϕ 的优化过程视为编码过程，将对 θ 的优化过程视为解码过程。

如何计算梯度？

有关于参数 θ, ϕ 的梯度实际上没有解析解，因此我们需要采用蒙特卡洛模拟来进行近似，即：

$$\mathbf{E}_{q(\mathbf{z}; \phi)} [\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q(\mathbf{z}; \phi)] \approx \frac{1}{K} \sum_k (\log p(\mathbf{z}^k, \mathbf{x}; \theta) - \log q(\mathbf{z}^k; \phi))$$

计算 θ 的梯度相对比较简单：

$$\begin{aligned}
\nabla_{\theta} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i) &= \nabla_{\theta} \mathbf{E}_{q(\mathbf{z}; \phi)} [\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q(\mathbf{z}; \phi)] \\
&= \mathbf{E}_{q(\mathbf{z}; \phi)} [\log p(\mathbf{z}, \mathbf{x}; \theta)] \\
&\approx \frac{1}{K} \sum_k \nabla_{\theta} \log p(\mathbf{z}^k, \mathbf{x}; \theta)
\end{aligned}$$

而计算 ϕ 的梯度则复杂的多，一般需要用到强化学习的技巧。但是有一个简单但是泛用性较差的方法可供选择，即**重参数化技巧**

重参数化

现在我想从下式中计算 ϕ 的梯度：

$$\mathbf{E}_{q(\mathbf{z}; \phi)} [r(\mathbf{z})] = \int q(\mathbf{z}; \phi) r(\mathbf{z}) d\mathbf{z}$$

假设 $q(\mathbf{z}; \phi) \sim \mathcal{N}(\mu, \sigma^2 \mathbf{I})$ 是以 $\phi = (\mu, \sigma)$ 为参数的高斯分布，则以下两种采样方式是完全等价的：

- 从 $q(\mathbf{z}; \phi)$ 中直接采样 \mathbf{z}
- 从标准高斯分布中采样 $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ ，计算 $\mathbf{z} = \mu + \sigma \epsilon = g(\epsilon; \phi)$

于是我们可以用以下方式进行计算：

$$\begin{aligned}
\mathbf{E}_{q(\mathbf{z}; \phi)} [r(\mathbf{z})] &= \int q(\mathbf{z}; \phi) r(\mathbf{z}) d\mathbf{z} = \mathbf{E}_{\epsilon \sim \mathcal{N}(0, \mathbf{I})} [r(g(\epsilon; \phi))] = \int \mathcal{N}(\epsilon) r(\mu + \sigma \epsilon) d\epsilon \\
\nabla_{\phi} \mathbf{E}_{q(\mathbf{z}; \phi)} [r(\mathbf{z})] &= \nabla_{\phi} \mathbf{E}_{\epsilon} [r(g(\epsilon; \phi))] = \mathbf{E}_{\epsilon} [\nabla_{\phi} r(g(\epsilon; \phi))] \approx \frac{1}{K} \sum_k \nabla_{\phi} r(g(\epsilon_k; \phi))
\end{aligned}$$

这仅仅适用于连续变量中的高斯分布。

现在我们回代入原式可得：

$$\begin{aligned}
\mathcal{L}(\mathbf{x}; \theta, \phi) &= \sum_{\mathbf{z}} q(\mathbf{z}; \phi) \log(p(\mathbf{z}, \mathbf{x}; \theta)) + H(q(\mathbf{z}; \phi)) \\
&= \mathbf{E}_{q(\mathbf{z}; \phi)} [\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q(\mathbf{z}; \phi)] \\
&= \mathbf{E}_{q(\mathbf{z}; \phi)} [r(\mathbf{z}, \phi)] \\
&= \mathbf{E}_{q(\mathbf{z}; \phi)} [r(g(\epsilon; \phi), \phi)] \\
&= \frac{1}{K} \sum_k r(g(\epsilon^k; \phi), \phi)
\end{aligned}$$

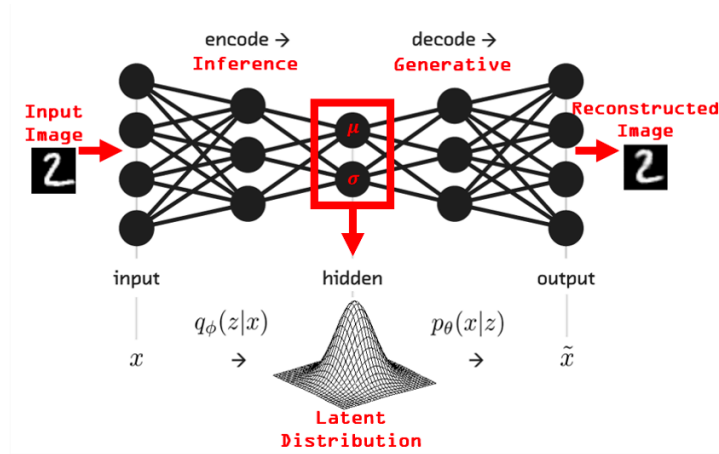
摊销

现在已经得到了对于每一个样本 \mathbf{x}^i 对应的最佳变分参数 ϕ^i ，但是如果数据集过大，对每一个样本都存储一个 ϕ 是很昂贵的，因此我们可以再定义一个函数 f_{λ} 来将每一个样本映射到其对应的最佳变分参数，即 $\mathbf{x}^i \mapsto \phi^i$ ，接下来我们就可以把后验分布变成以下形式：

$$q(\mathbf{z} | \mathbf{x}^i) = q_{\lambda}(\mathbf{z} | \mathbf{x})$$

我们可以在上面的推断中把 $q(\mathbf{z}; \phi^i)$ 替换为 $q_{\phi}(\mathbf{z}, \mathbf{x})$

总结 (VAE模型整体架构)



$$\begin{aligned}
 \mathcal{L}(\mathbf{x}; \theta, \phi) &= \mathbf{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q_{\phi}(\mathbf{z}|\mathbf{x})] \\
 &= \mathbf{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{z}, \mathbf{x}; \theta) - \log p(\mathbf{z}) + \log p(\mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x})] \\
 &= \mathbf{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z}; \theta)] - D_{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))
 \end{aligned}$$

我们需要最大化第一项并最小化第二项。其中第一项为重建损失，衡量解码器生成的数据与真实数据的相似程度，对应于解码器；第二项为KL正则项，约束编码器输出的分布 $q_{\phi}(\mathbf{z}|\mathbf{x})$ 接近先验分布 $p(\mathbf{z})$ ，对应于编码器

模型架构

编码器（推断网络）：将输入数据 \mathbf{x} 映射到潜在空间，输出潜在变量 \mathbf{z} 的分布参数（均值 μ 和方差 σ^2 ），即：

$$q_{\phi}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu_{\phi}(\mathbf{x}), \sigma_{\phi}(\mathbf{x}))$$

其中 ϕ 为编码器参数。

解码器（生成网络）：从潜在变量 \mathbf{z} 重构数据 \mathbf{x} ，输出生成分布的参数

$$p_{\theta}(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mu_{\theta}(\mathbf{z}), \sigma_{\theta}^2(\mathbf{z}))$$

其中 θ 为解码器参数

潜在变量生成与重参数化

采样潜在变量：从编码器输出的分布中采样 \mathbf{z} ：

$$\mathbf{z} = \mu_{\phi}(\mathbf{x}) + \sigma_{\phi}(\mathbf{x}) \cdot \epsilon, \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

重参数化技巧将随机采样过程转换为确定性计算，使梯度可通过神经网络反向传播。

训练过程

1. 采样样本 \mathbf{x}^i
2. 通过从 $q_{\phi}(\mathbf{z}|\mathbf{x}^i)$ 中采样，将其映射到潜变量 $\hat{\mathbf{z}}$ （编码器）（相当于是从图像中提取一些特征）
3. 通过从 $p(\mathbf{x}|\hat{\mathbf{z}}; \theta)$ （解码器）采样来重构 $\hat{\mathbf{x}}$ （相当于从特征中采样，组成一个类似的带有这些特征的图片）
4. 计算损失并反向传播：通过梯度下降优化 θ, ϕ ，最大化ELBO。

生成新数据

- 从先验分布 $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$ 中采样 \mathbf{z} 。
- 通过解码器生成数据 $\mathbf{x} \sim p_{\theta}(\mathbf{x}|\mathbf{z})$ 。

核心优势与特点

- **无监督特征学习**：潜在变量 \mathbf{z} 编码数据的高层语义（如姿态、颜色）。
- **生成多样性**：通过采样 \mathbf{z} 生成多样化样本。
- **正则化潜在空间**：KL散度约束潜在空间结构，支持插值（如从“微笑”到“不微笑”的渐变）。

宏观层面思考：为什么需要优化潜在空间？

VAE的核心目标是生成与训练数据相似的新数据，同时还能捕捉数据的内在结构。潜在空间（latent space）在这里就像一个“压缩的蓝图”，它把高维的原始数据（比如图片、声音）映射到一个低维的、更有组织的形式。如果这个潜在空间没有被优化好，模型就没法很好地完成生成任务，也无法真正理解数据的规律。

1. 数据的结构化表示

想象一下，假如你要把一堆杂乱无章的照片整理成一个相册，你需要找到一种方法来“总结”每张照片的主题（比如风景、人像）。潜在空间就是这个“总结”的地方。VAE通过优化潜在空间，确保每个点（潜在变量）都能对应数据的某种有意义的特征。如果不优化，潜在空间可能会变得混乱，像一堆随机的数字，失去了表示能力。

2. 生成能力的需要

VAE不仅仅是压缩数据，它还要能解码潜在空间的点，生成新的样本。为了让生成的样本既有多样性又符合数据的分布，潜在空间需要被塑造成一个连续且规则的结构（通常假设服从标准正态分布）。优化潜在空间就是在调教这个“生成工厂”的布局，让它既能覆盖数据的多样性，又不会生成完全离谱的结果。

3. 平衡重建和正则化

VAE的损失函数有两个部分：重建误差（让生成的样本尽量接近输入）和KL散度（让潜在空间的分布接近一个简单的先验分布，比如正态分布）。优化潜在空间就是在两者之间找平衡。如果不优化，模型可能会过分专注于重建输入（像普通的自编码器），但潜在空间会变得无序，无法用来生成新数据或进行推断。

从更广义的角度看，VAE的潜在空间优化是在试图“提炼数据的本质”。它有点像人类大脑处理信息的方式：我们看到的世界是复杂的高维数据（视觉、听觉等），但大脑会把这些压缩成更简单的概念（比如“猫”“树”）。VAE通过优化潜在空间，试图模仿这种能力，让机器也能从杂乱的数据中提取规律，进而理解和创造。

如果不优化潜在空间，VAE就退化成了一个普通的压缩工具，失去了生成模型的灵魂——无法探索数据的可能性，也无法真正“理解”数据的深层结构。所以，优化潜在空间是VAE能从“模仿”走向“创造”的关键一步。

在VAE里怎么用先验分布？

VAE通过编码器把输入数据 \mathbf{x} 映射到潜在变量 \mathbf{z} ，但这些 \mathbf{z} 的分布（叫做 $q(\mathbf{z}|\mathbf{x})$ ）一开始是杂乱的，可能每个数据点映射出来的 \mathbf{z} 都不一样。VAE用KL散度“惩罚”这个分布，让它尽量靠近先验分布 $p(\mathbf{z})$ （比如 $\mathcal{N}(0, \mathbf{I})$ ）。这就像在说：“嘿，你的潜在空间别太花哨，尽量按我的简单规矩来！”

最终，训练好的模型可以用这个简单的先验分布来采样（比如从 $\mathcal{N}(0, \mathbf{I})$ 里随便挑个数），然后通过解码器生成新的数据。因为潜在空间已经被调教得接近先验分布，所以生成的东西既有规律，又能反映数据的特性。

为什么要让潜在空间的分布接近简单的先验分布？

生成能力的可控性

VAE的目标不仅是重建数据，还要能生成新的、合理的样本。如果潜在空间的分布是混乱的、不可预测的，那么从里面采样（比如随机挑一个点来生成数据）就会变得很困难。假设我们让潜在空间接近一个简单的先验分布（比如标准正态分布 $\mathcal{N}(0, \mathbf{I})$ ），我们就可以轻松地从中采样，因为我们很熟悉正态分布的性质。这样，生成过程就变得可控且高效——你随便

从 $\mathcal{N}(0, \mathbf{I})$ 里挑个数，就能生成一个像样的样本。

避免过拟合，增加泛化性

如果潜在空间的分布完全由数据自由决定（没有先验约束），模型可能会过分记住训练数据，导致潜在空间变得过于复杂，甚至每个数据点都被“硬编码”到一个特定的潜在变量上。这样虽然重建效果很好，但生成新样本时就容易失控，生成的可能是一些奇怪的东西。让潜在空间接近简单先验分布（通过KL散度正则化），就像给模型加了个“纪律约束”，防止它过于“任性”，从而提升泛化能力。

连续性和平滑性

一个接近简单先验分布的潜在空间（比如正态分布）通常是连续且平滑的。这意味着潜在空间里相邻的点生成的样本也会有相似的特征。比如在一张人脸生成模型中，潜在空间里两个靠近的点可能生成两张脸，只是鼻子大小略有不同。如果分布不规则，潜在空间可能会出现“断层”，采样时稍微偏一点就生成完全不相关的东西，失去了结构的意义。

数学上的方便性

简单先验分布（比如正态分布）有解析形式，计算起来方便。在VAE中，KL散度用来衡量潜在分布 $q(\mathbf{z}|\mathbf{x})$ 和先验分布 $p(\mathbf{z})$ 的差距。如果 $p(\mathbf{z})$ 是个简单的正态分布，KL散度可以直接算出来，优化过程就更稳定。如果先验分布太复杂，计算和优化都会变得麻烦。

为什么不直接定义一个简单的潜在空间就好了？

这里有个关键问题：潜在空间不是我们能完全手动指定的东西，它必须从数据中“学”出来。原因如下：

1. 数据驱动的本质

VAE是个生成模型，它的潜在空间需要反映数据的真实分布。如果我们强行定义一个简单的潜在空间（比如直接说“潜在变量就是正态分布”），但不通过训练去调整它，那这个空间可能完全无法捕捉数据的特征。比如，你硬定义一个为什么不能直接定义一个简单的潜在空间就好了？

2. 编码器和解码器的协调

VAE有两个部分：编码器（把输入 \mathbf{x} 映射到潜在变量 \mathbf{z} ）和解码器（从 \mathbf{z} 生成输出）。编码器学到的潜在分布 $q(\mathbf{z}|\mathbf{x})$ 是数据驱动的，取决于输入 \mathbf{x} 。如果我们直接定义一个固定的潜在空间，但不让编码器去适应它，编码器和解码器就会“脱节”——编码器可能把数据映射到完全不符合先验的地方，解码器也不知道怎么处理这些点。

3. 优化过程的必要性

VAE通过优化损失函数（重建误差 + KL散度）来“拉近” $q(\mathbf{z}|\mathbf{x})$ 和 $p(\mathbf{z})$ 的距离。这个过程其实是在动态调整潜在空间，让它既能反映数据的特性，又尽量贴近简单的先验分布。如果直接定义一个先验而不优化，模型就失去了学习的灵活性，无法在“数据真实性”和“生成可控性”之间找到平衡。

4. 直接定义的局限性

如果你直接定义潜在空间为某个分布（比如正态分布），但不通过训练去约束编码器输出，编码器可能会输出任意分布（比如均值方差完全不规则）。这样，采样时你还是得从那个定义的分里挑点，但解码器面对编码器给出的混乱输入，根本没法正常工作。换句话说，直接定义解决不了潜在空间和模型整体一致性的问题。

宏观比喻

想象潜在空间是个“模具”，而数据是“面团”。我们希望模具简单好用（接近先验分布），但又能做出符合面团特性（数据分布）的形状。直接定义一个模具而不调整，可能会完全压不出面团的样子；而通过优化，我们让模具在保持简单的同时，慢慢适应面团的特性，最终既能批量生产（生成），又能保证质量（符合数据规律）。

所以，VAE不直接定义一个潜在空间，而是通过训练让它“逐渐靠近”简单先验分布，这样既保留了数据的特性，又保证了生成过程的可控性和实用性。