VQ-VAE

xinli*

August 11, 2024

Abstract

这里记录了我在学习 VQVAE 过程中的一些思考和记录，帮助我更好的理解 VQ-VAE。(仅供参考)

1 Introduction

在学习 VQVAE 的过程中，我常常会思考几个问题。1、VQVAE 想要解决的是 VAE 中的什么问题？2、VQVAE 是如何解决这个问题的？接下来让我们带着这些问题开始学习吧。

2 Preliminary

2.1 回顾 AE 与 VAE

我们先来回顾一下从 AE 到 VAE 的路程吧。AE 也叫做 autoencoder，它由一个编码器和解码器构成，其中编码器负责将图像压缩为潜在变量 z，而解码器负责将潜在变量重建为图像，所以 AE 的 loss 需要衡量输入的图像与输出的图像之间的差异，我们可以用 MSE-loss 来完成：

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\mathbf{x}}_i)^2 \tag{1}$$

AE 模型的缺点是它会带来过拟合问题。举个例子，即使我们设计了足够优秀的 encoder 和 decoder，并且模型已经可以拟合训练集中的所有样本。比如说训练集中的 z 可以由 1,2,3 等来代替，但是我推理的时候给出一个 1.5（照理来说，1.5 对应的图片应该和 1、2 对应的图片差不多），但是它生成的结果很糟糕。也就是说，**AE 的这种连续且固定的编码方式让编码本身失去了语义信息，编码之间的插值不能表示图像语义之间的插值**。

而 VAE 模型是通过引入了正则化和概率分布来解决优化 AE 模型的。相对于 AE 模型来说，VAE 的编码器输出的是潜在空间中每个维度的均值 μ 和方差 σ^2，然后从标准正态分布中采样一个噪声向量，接下来通过重参数化技巧来采样潜在向量 z。

$$z = \mu + \sigma \cdot \epsilon \tag{2}$$

这里我们再来推导一个 VAE 的 loss，当做复习：

$$\log p(x) = \log \int p(x, z)dz = \log \int p(x \mid z)p(z)dz \tag{3}$$

*https://github.com/xinli2008

引入变分推断 $q(z \mid x)$

$$\log p(x) = \log \int q(z \mid x) \frac{p(x,z)}{q(z \mid x)} dz \tag{4}$$

应用 Jensen 不等式，得到:

$$\log p(x) \geq \int q(z \mid x) \log \frac{p(x,z)}{q(z \mid x)} dz \tag{5}$$

而由于联合分布 $p(x,z) = p(x \mid z)p(z)$，则上面的公式等于:

$$\log p(\mathbf{x}) \geq \int q(\mathbf{z} \mid \mathbf{x}) \log \frac{p(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})} d\mathbf{z} \tag{6}$$

拆开，然后合并得到：

$$\log p(\mathbf{x}) \geq \int q(\mathbf{z} \mid \mathbf{x})(\log p(\mathbf{x} \mid \mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z} \mid \mathbf{x})) d\mathbf{z} \tag{7}$$

$$\log p(\mathbf{x}) \geq \int q(\mathbf{z} \mid \mathbf{x}) \log p(\mathbf{x} \mid \mathbf{z}) d\mathbf{z} + \int q(\mathbf{z} \mid \mathbf{x}) \log p(\mathbf{z}) d\mathbf{z} - \int q(\mathbf{z} \mid \mathbf{x}) \log q(\mathbf{z} \mid \mathbf{x}) d\mathbf{z} \tag{8}$$

所以，VAE 的 loss 如下。它由重构误差和 KL 散度组成，前者衡量重构质量，后者确保潜在空间的连续性和平滑性。

$$\mathcal{L}_{VAE} = -\mathbb{E}_{q_\phi(z|x)} \left[\underbrace{\log p_\theta(x|z)}_{\text{how good your decoder is}} \right] + \underbrace{\mathcal{D}_{\mathrm{KL}} \left(\underbrace{q_\phi(z|x)}_{\text{a Gaussian}} \parallel \underbrace{p(z)}_{\text{a Gaussian}} \right)}_{\text{how good your encoder is}} \tag{9}$$

$$\mathcal{L}_{VAE} = -\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x} \mid \mathbf{z})] + \frac{1}{2} \sum_{j=1}^{d} \left(\sigma_j^2(\mathbf{x}) + \mu_j^2(\mathbf{x}) - 1 - \log \sigma_j^2(\mathbf{x}) \right) \tag{10}$$

那么 VAE 的问题是什么呢?

1、**模糊的训练样本**：VAE 的生成样本通常比 GAN 和 diffusion 系列要模糊。这是由于 VAE 采用了均方误差 MSE 作为重建损失函数，而 MSE 会倾向于生成平均值，从而导致模糊的图像或样本。

2、**后验崩塌 (posterior collapse)**: VAE 的编码器的目的是学习一个有意义的潜在空间，使得解码器可以从中重建输入数据。然后，当解码器足够强大的时候，它可以不依赖于编码器生成的潜在表示，直接从先验分布中采样并生成高质量的输出数据。这种情况下，编码器会输出一个与先验分布非常接近的分布（例如非常接近均值为零、方差为 1 的正态分布），导致潜在变量 z 的后验分布 $q(z \mid x)$ 退化成了先验分布 $p(z)$。

2.2 向量量化（Vector Quantization, VQ）

向量量化是一种将连续信号转化为离散信号的过程，它的基本思想是用一个**有限数量**的“码字”来表示一个大的、连续的向量空间中的数据点。向量量化可能涉及到的一些关键概念是：

1、码书 (codebook): 这是一个包含多个”码字”的集合。每个码字都是一个向量，用来代表输入空间中的某个区域。

2、编码：对于一个输入向量，**VQ 的编码过程是寻找最接近这个输入向量的码字，并用该码字来表示输入向量**。这种最接近的码字通常是通过最小化输入向量与码字之间的欧氏距离来确定的。

Abstract

Learning useful representations without supervision remains a key challenge in machine learning. In this paper, we propose a simple yet powerful generative model that learns such discrete representations. Our model, the Vector Quantised-Variational AutoEncoder (VQ-VAE), differs from VAEs in two key ways: the encoder network outputs discrete, rather than continuous, codes; and the prior is learnt rather than static. In order to learn a discrete latent representation, we incorporate ideas from vector quantisation (VQ). Using the VQ method allows the model to circumvent issues of "posterior collapse" -— where the latents are ignored when they are paired with a powerful autoregressive decoder -— typically observed in the VAE framework. Pairing these representations with an autoregressive prior, the model can generate high quality images, videos, and speech as well as doing high quality speaker conversion and unsupervised learning of phonemes, providing further evidence of the utility of the learnt representations.

Figure 1: vqvae abstract

3 VQVAE 的模型结构

我们先来阅读一下 VQVAE 的摘要部分，如图(1)所示。从中我们可以看到，VQVAE 与 VAE 的不同点有两处：第一点是，编码器网络输出离散而非连续的编码：在传统的 VAE 中，编码器将输入数据映射到一个连续的潜在空间。而**VQVAE 通过引入了向量化机制，它的编码器输出的是离散的编码，而不是连续的编码**。具体来说，VQVAE 是先通过一个 encoder 得到一个潜在变量 z，然后在通过 embedding space 将一个潜在变量 z 映射成一个离散的嵌入向量。

第二点不同的是先验分布是学习到的，而不是静态的。在 VAE 中，我们记得它的 loss 分别两个部分，一个是重建 loss，另一个是 KL 散度的 loss。其中 KL 散度的 loss 指的是：

$$\mathrm{KL}(q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z})) = \int q(\mathbf{z} \mid \mathbf{x}) \log \frac{q(\mathbf{z} \mid \mathbf{x})}{p(\mathbf{z})} d\mathbf{z} \tag{11}$$

这里的 KL 散度衡量的是 encoder 输出的分布 $q(z \mid x)$ 与先验分布 $p(z)$ 之间的差距。我们在实现 VAE 时，这里的先验分布 $p(z)$ 是高斯分布，它是一个静态的，一成不变的。但是**在 VQVAE 中，先验分布是一个可以学习的参数。具体来说，我们先用 nn.embedding 来设置一个 embedding space，然后将 encoder 的输出 z 通过 embedding space 转化为一个离散的数据表示**。

相信读到这里，大家对 VQVAE 的模型结构已经有了大概的理解，我们可以看一下它的模型结构图(2)。

3.1 编码器

首先，输入数据（如图像）经过编码器 $E(x)$ 被映射到一个潜在空间中的连续表示 $z_e(x)$：

$$z_e(x) = E(x) \tag{12}$$

这个表示 $z_e(x)$ 是一个实数向量，它捕捉了输入数据的特征，**但它仍然是连续的，类似于 AE 和 VAE 的潜在表示**。

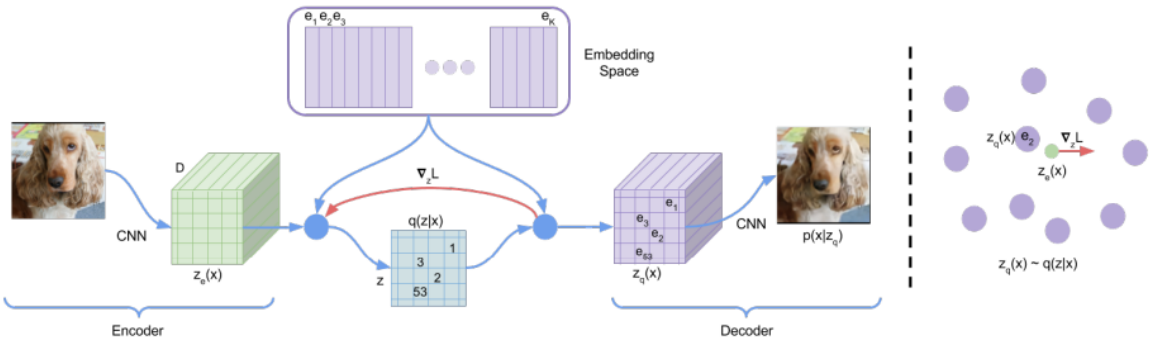

Figure 2: vqvae architecture

3.2 向量量化（VQ）

接下来是向量量化模块，这是 VQ-VAE 的核心创新部分。

在这个阶段，连续表示 $z_e(x)$ 会被映射到一个离散的嵌入向量空间（即码书，通常用 e_k 表示），这个空间包含了 K 个嵌入向量 $e_1, e_2, \ldots, e_K$。每个嵌入向量 e_k 都是一个固定大小的向量，作为这个空间中一个点的代表。

为了将连续表示 $z_e(x)$ 映射到离散的嵌入向量，VQ 模块会选择离 $z_e(x)$ 最近的嵌入向量 e_k：

$$z_q(x) = e_k \quad \text{where} \quad k = \arg\min_j \|z_e(x) - e_j\| \tag{13}$$

这个过程将连续表示 $z_e(x)$ 量化为一个离散值 $z_q(x)$，而这个值就是嵌入向量空间中的某个嵌入向量 e_k。

3.3 解码器

量化后的离散嵌入向量 $z_q(x)$ 会被传递给解码器 $D(z_q(x))$ 进行重建或生成：

$$\hat{x} = D(z_q(x)) \tag{14}$$

解码器的任务是将量化后的离散嵌入向量 $z_q(x)$ 转换回原始数据空间，生成一个与输入数据 x 尽可能接近的输出 $\hat{x}$。

4 VQ-VAE 的损失函数

VQ-VAE 的训练过程基于一个复合损失函数，包含三个主要部分：

4.1 重建损失（Reconstruction Loss）

这是一个标准的重建损失，用于衡量解码器生成的 $\hat{x}$ 与原始输入 x 之间的差异。通常使用均方误差（MSE）来计算：

$$\mathcal{L}_{\text{rec}} = \|x - \hat{x}\|^2 \tag{15}$$

4.2 向量量化损失（Vector Quantization Loss）

为了确保编码器输出的连续表示 $z_e(x)$ 能够接近嵌入空间中的某个嵌入向量 e_k，引入了一个向量量化损失：

$$\mathcal{L}_{\text{vq}} = \|\text{sg}[z_e(x)] - e_k\|^2 \tag{16}$$

这里的 sg[·] 是 stop-gradient 操作，表示这个项不会对编码器的参数产生梯度影响，只有用于更新嵌入向量。

4.3 批量更新损失（Commitment Loss）

为了让编码器输出的 $z_e(x)$ 对量化的嵌入向量 e_k 负责，同时避免 $z_e(x)$ 偏离嵌入向量太多，引入了一个承诺损失（commitment loss）：

$$\mathcal{L}_{\text{commit}} = \|z_e(x) - \text{sg}[e_k]\|^2 \tag{17}$$

这个损失确保编码器不会产生过大的偏离，从而使得嵌入向量能够更好地表示输入数据。

4.4 总损失

最终的总损失是上述三部分的加权和：

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{vq}} + \beta\mathcal{L}_{\text{commit}} \tag{18}$$

其中，β 是一个超参数，用于调节承诺损失的权重。