第 5 章 编码器-解码器模型

5.1 编码器-解码器模型基础

编码器-解码器(Encoder-Decoder)模型是一种常见的序列到序列学习模型,它在自然语言处理和计算机视觉等领域都有广泛的应用。

5.1.1 编码器-解码器模型的基本结构

编码器-解码器算法是一种深度学习模型结构,被广泛地应用于自然语言处理(NLP)、图像处理、语音识别等领域。它主要由两部分组成:编码器(Encoder)和解码器(Decoder)。这种结构能够处理序列到序列(Seq2Seq)的任务,如机器翻译、文本摘要、对话系统等。

在自然语言处理领域编码器的作用是接收输入序列,并将其转换成固定长度的上下文向量(Context Vector)。这个向量是输入序列的一种内部表示,捕获了输入信息的关键特征。在自然语言处理的应用中,输入序列通常是一系列词语或字符。

编码器可以是任何类型的深度学习模型,但循环神经网络(RNN)及其变体,如长短期记忆网络(LSTM)和门控循环单元(GRU),因其在处理序列数据方面的优势而被广泛使用。

解码器的目标是将编码器生成的上下文向量转换为输出序列。在开始解码过程时,它首先接收到编码器生成的上下文向量,然后基于这个向量生成输出序列的第 1 个元素。接下来,它将自己之前的输出作为下一步的输入,逐步生成整个输出序列。

解码器也可以是各种类型的深度学习模型,但通常与编码器使用相同类型的模型以保持一致性。

在训练编码器-解码器模型时,目标是最小化模型预测的输出序列与实际输出序列之间的差异。这通常通过计算损失函数(如交叉熵损失)实现,并使用反向传播和梯度下降等优化算法进行参数更新。

5.1.2　编码器-解码器模型在自然语音处理领域的应用

1. 机器翻译

机器翻译是编码器-解码器模型最为广泛的应用之一。在机器翻译任务中，编码器-解码器模型将一个源语言句子映射成一个目标语言句子，其中编码器将源语言句子编码成一个固定长度的向量，解码器将这个向量解码成一个目标语言句子。

在编码阶段，编码器部分的任务是处理输入序列（源语言文本）。每个输入词元（可以是词或字符）被转换为向量，然后输入编码器网络（通常是 RNN、LSTM 或 GRU）。编码器逐步处理输入序列中的每个元素，更新其内部状态。最后一个时间步的内部状态被认为是对整个输入序列的压缩表示，称为上下文向量或编码器隐藏状态。这个向量旨在捕获输入序列的语义信息。

在解码阶段，解码器接收编码器的上下文向量，并开始生成输出序列（目标语言文本）。解码器的初始状态通常是编码器的最终状态。在每个时间步，解码器基于当前的状态和前一时间步的输出（或在训练期间，可能是前一时间步的实际目标词元）预测下一个词元，然后这个预测被用作下一个时间步的输入，直到生成序列结束标记为止。

此外，为了改善性能，特别是在处理长序列时，注意力机制被引入。注意力允许解码器在生成每个词时“关注”输入序列的不同部分。具体来讲，它通过计算解码器的当前状态和编码器每个时间步状态的相似度，为编码器的每个输出分配一个权重，然后生成一个加权组合，这个加权组合被用作附加的上下文信息输入解码器，帮助生成更准确的输出。

2. 文本摘要

文本摘要是一种将长文本压缩成短文本的任务，其中编码器-解码器模型通常用于生成一个摘要句子。在文本摘要任务中，编码器将输入文本编码成一个向量，解码器根据这个向量生成一个与输入文本相对应的摘要句子。

在编码阶段，编码器读取整个输入文本（例如，一篇文章），逐词（或逐字符）进行处理，更新其内部状态。在处理序列的每个步骤，编码器试图捕获并累积文本的关键信息，并将这些信息编码进一个固定长度的向量中。

在解码阶段，使用编码器的输出（上下文向量）作为输入，解码器开始生成文本摘要。与机器翻译类似，解码器在每个时间步基于当前的上下文和前一时间步的输出（或在训练期间，前一时间步的实际目标词元）生成下一个词元。这个过程重复进行，直到达到预设的长度限制或生成了特殊的序列结束标记。

在文本摘要中，注意力机制同样重要，因为它使模型能够在生成摘要的每步“关注”输入文本的不同部分，从而提高摘要的相关性和准确性。此外，一些高级技术，如复制机制（允许直接从输入将词汇复制到输出）、覆盖机制（防止重复）和内容选择技术被用来进一步提高摘要质量。

3. 对话生成

在探讨对话系统和聊天机器人的实现过程中，我们聚焦于编码器-解码器模型的核心作

用，尤其是它在处理生成式对话任务时的应用。这一过程可以分为几个关键阶段，每个阶段都对生成流畅且相关的回应至关重要。

(1) 输入处理阶段：在与用户的交互中，自然语言文本作为对话系统的输入序列发挥着基础作用。编码器负责接收这些输入文本，并将其逐词或逐字符转换为向量表示。这些向量随后通过编码网络(例如 RNN、LSTM、GRU)进行处理，网络在此过程中更新其内部状态，以反映序列中累积的信息。编码过程的终点是生成一个或多个向量，这些向量综合概括了输入文本的内容及其上下文，为后续的回应生成奠定了基础。

(2) 回应生成阶段：在初始化阶段，解码器的起始状态通常由编码器的最终状态所决定，确保了从用户输入提取的信息被有效利用以引导回应的生成。解码器随后根据从编码阶段获得的上下文信息，逐步构建回应。它在每一时间步骤中预测下一个词元，直至遇到终止符号，如句号或特殊的结束标记，从而完成一个回应的生成。在此过程中，解码器会综合考量当前的内部状态及先前时间步骤生成的词元，以确定下一步最合适的输出。

(3) 注意力机制：在对话系统的构建中，注意力机制的引入极大地增强了模型的性能。它允许解码器在生成回应时专注于输入序列的特定部分，使系统能够根据对话的实际上下文调整其回应，从而产生更为相关和个性化的输出。这一机制通过计算解码器的当前状态与编码器各种状态之间的相似度，从而确定模型的焦点。相似度得分用于为编码器的输出创建加权组合，该组合随后作为额外的上下文信息辅助回应的生成，进一步提升了交互质量。

5.1.3 编码器-解码器模型在计算机视觉领域的应用

1. 图像去噪

在图像去噪的应用中，自编码器通过一个精妙的编码-解码过程，学习如何从含噪声的图像中恢复出清晰的图像。编码器部分将输入的含噪声图像转换成一个潜在的、紧凑的表示形式，在这一过程中，通过逐层减少数据维度，迫使网络捕获图像的本质特征，而忽略那些不重要的噪声。随后，解码器部分接手这个潜在表示，通过逐层增加数据维度，重构出去噪后的图像。在实现上，自编码器通过最小化重构图像与原始图像之间的差异(例如，使用均方误差作为损失函数)进行训练，从而有效地学习到如何去除图像中的噪声。

2. 特征提取与降维

自编码器在特征提取与降维方面的应用基于其能力将高维数据转换为低维的、有意义的表示。在编码过程中，自编码器通过减少数据的维度，将输入图像编码到一个潜在空间中，这个潜在空间的表示捕获了输入数据的关键信息。这种低维表示为各种视觉任务(如分类、检测等)提供了一种更为简洁且信息丰富的数据形式。在解码过程中，尽管目标是重构原始输入，但这一过程的副产品(潜在空间的表示)在实践中被广泛地用作特征向量。通过这种方式，自编码器不仅实现了数据的有效压缩，还提供了一种强大的特征学习机制。

3. 图像生成

在图像生成领域，变分自编码器(VAE)扩展了自编码器的概念，引入了能够学习输入

数据分布的能力。VAE的编码器不仅学习了如何将输入映射到潜在空间，而且还学习了潜在空间中数据分布的参数（如均值和方差）。这允许VAE通过在潜在空间进行随机采样，然后通过解码器生成新的、多样化的图像。这种方法的核心在于其损失函数，它由重构损失（鼓励模型准确重构输入数据）和KL散度（鼓励潜在空间的分布接近先验分布）组成。通过优化这个损失函数，VAE能够生成与训练数据相似但又全新的图像实例，为创造性图像生成和其他应用开辟了广阔的天地。

5.1.4　自编码器模型

接下来将介绍编码器解码器模型的特例：自编码器。这一无监督学习的典范，以一种近乎自省的方式工作，旨在通过其内部的编码器部分探索并捕获输入数据的本质特征，将这些丰富的信息压缩成一个紧凑的潜在空间表示。随后，其解码器部分努力重构出原始数据，尽可能地复原失去的细节。这一过程不仅体现了数据压缩和降噪的实用价值，也暗示了对数据内在结构的深刻理解。自编码器的魅力在于其能够在没有任何外部标签或指示的情况下，自我学习并发现数据的隐藏规律。

与自编码器的内省式学习不同，编码器-解码器模型则是一种旨在桥接序列之间差异的架构，常见于需要将一种形式的数据转换为另一种形式的监督学习任务中。这种模型以其灵活性著称，能够处理从语言翻译到语音识别等多样化的任务。在这一框架下，编码器首先解析输入序列，提炼出能够代表其核心意义的潜在表示；解码器随后接过这一表示，创造性地转换为目标序列。这一过程不仅需要对输入数据有深刻的理解，还要求模型具备高度的创造力，以生成准确且流畅的输出。

尽管自编码器与编码器-解码器模型在目标、应用和训练方法上各不相同，但它们之间的联系不容忽视。两者都采用了将数据编码到某种形式的潜在表示这一共同理念，这一点体现了深度学习模型设计中的一种普遍策略：通过某种形式的内部表示，捕获并利用数据的本质特性。无论是自编码器在无监督学习中的应用，还是编码器-解码器模型在监督学习任务中的广泛使用都深刻地展示了深度学习的灵活性和强大能力，为我们提供了理解复杂数据结构和解决实际问题的强有力工具。

5.2　CV中的编码器-解码器：VAE模型

变分自编码器（Variational Autoencoder，VAE）是一种生成模型，最终目标是掌握如何构建一个生成模型。生成模型的本质是定义一个联合概率分布 $p(x,z)$，其中 x 表示观测数据，z 表示潜在变量。得到联合概率分布后，就可以通过采样 z 生成不同的样本 x。

以生成手写数字图片为例子。假设有一个手写数字的图片集合。在这个集合里，每张图片都有一个固定的尺寸，用 X 来代表这个图片集合。在这个环境下，有一个名为潜空间的概念，记为 Z。这个潜空间包含了所有可能的手写数字和它们的风格。可将其理解为一

个“隐藏的”空间，其中每个点都代表一个独特的手写体数字风格。P 是一个概率分布，用于表示在潜空间中生成手写数字的可能性。当从 P 中随机抽样一个点，这个点就会对应到 Z 中的一个特定数字和风格的组合。现在所面临的主要挑战是：如何使用观察到的真实手写数字图片(X)来建立这个 P 概率分布？

变分自编码器是自编码器的一种扩展。与普通自编码器不同，VAE 假定输入数据是由某种潜在变量及该潜在变量的概率分布生成的。在编码过程中，它不仅输出一个隐藏表示，还输出一组参数，这些参数定义了潜在空间中的一个概率分布。解码器则从该分布中抽取样本，并根据这些样本生成新的输入实例。

因此，变分自编码器可以生成新的、与输入数据相似的实例，这使它们非常适合于生成模型的任务，而且，由于它们明确地学习了潜在空间的概率分布，所以可以生成具有连续性和结构性的数据，这对于许多任务来讲是很有价值的。下面将逐步展开讲解 VAE 模型的理论知识。5.2.1 节是 VAE 的简明讲解；5.2.2 节和 5.2.6 节都是学习 VAE 模型的数学理论知识，其关系也是逐层递进的，其中 5.2.6 节是 VAE 损失函数的数学理论推导，比较难理解，可以选看。

5.2.1 VAE 模型简明指导

VAE 最想解决的问题是如何构造编码器和解码器，使图片经过编码器能够编码成合理的特征向量，并且能够通过解码器尽可能无损地解码回原真实图像。

这听起来似乎与 PCA(主成分分析)有些相似，而 PCA 本身是用来做矩阵降维的。

如图 5-1 所示，$\boldsymbol{x}$ 本身是一个矩阵，通过一个变换 $\boldsymbol{W}$ 变成了一个低维矩阵 $\boldsymbol{c}$，因为这一过程是线性的，所以再通过一个变换就能还原出一个矩阵，现在要找到一种变换 $\boldsymbol{W}$，使矩阵 $\boldsymbol{x}$ 与 $\hat{\boldsymbol{x}}$ 能够尽可能地一致，这就是 PCA 做的事情。在 PCA 中找这个变换 $\boldsymbol{W}$ 用到的方法是奇异值分解(Singular Value Decomposition，SVD)算法，这是一个纯数学方法，不再细述，而在 VAE 中不再需要使用 SVD，直接用神经网络代替。

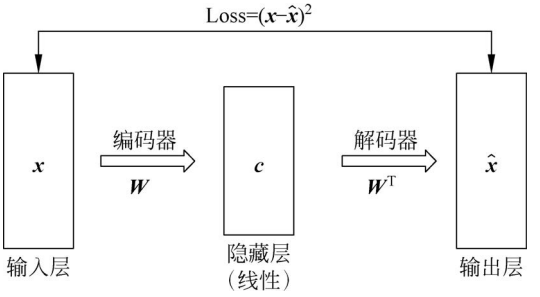

图 5-1 自编码器模型

PCA 与我们想要构造的自编码器的相似之处是：如果把矩阵 $\boldsymbol{x}$ 视作输入图像，将 $\boldsymbol{W}$ 视作一个编码器，将低维矩阵 $\boldsymbol{c}$ 视作图像的编码，将 $\boldsymbol{W}^{\mathrm{T}}$ 和 $\hat{\boldsymbol{x}}$ 分别视作解码器和生成图像，PCA 就变成了一个自编码器网络模型的雏形。用两个分别被称为编码器和解码器的神经

网络代替 $\boldsymbol{W}$ 变换和 $\boldsymbol{W}^{\mathrm{T}}$ 变换，就得到了自编码器模型。

这一替换的明显好处是，引入了神经网络强大的拟合能力，使编码的维度能够比原始图像的维度低很多。至此通过自编码器(Auto-Encoder，AE)构造出了一个比 PCA 更加清晰的自编码器模型，但这并不是真正意义上的生成模型。对于一个特定的生成模型，它一般应该满足以下两点：

(1) 编码器和解码器是可以独立拆分的(类比 GAN 的 Generator 和 Discriminator)。

(2) 固定维度下任意采样出来的编码都应该能通过解码器生成一张清晰且真实的图片。

解释下第二点。假设用一些全月图和一些半月图去训练一个 AE，经过训练，模型能够很好地还原出这两张图片，如图 5-2 所示。接下来在潜空间中取两张图片编码点中的任意一点，将这点交给解码器进行解码，直觉上会得到一张介于全月图和半月图之间的图片(如阴影面积覆盖 3/4)，然而，实际上这个点经过解码器解码后的结果不仅模糊而且还是乱码的，这就是自编码的过拟合现象。

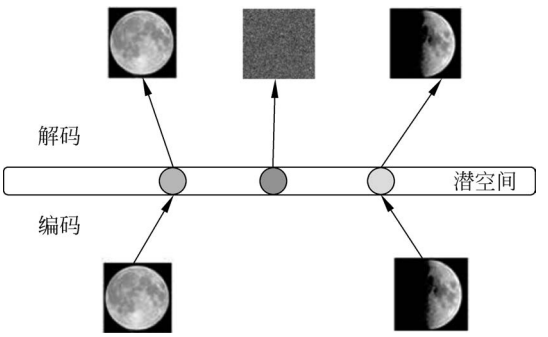

图 5-2　自编码的过拟合现象

为什么会出现这种现象？一个直观上的解释是 AE 的 Encoder 和 Decoder 都使用了神经网络，神经网络是一个非线性的变换过程，因此在潜空间中点与点之间的关系往往没有规律可循。解决此问题的一种方法是引入噪声，使图片的编码区域得到扩大，从而掩盖失真的空白编码点。

在对两张图片进行编码时引入一定的噪声，使每张图片的编码点出现在潜空间的矩形阴影范围内，如图 5-3 所示，因此，在训练模型时，矩形阴影范围内的点都有可能被采样到，这样解码器在训练过程中会尽可能地将矩形阴影内的点还原为与原图相似的图片。接着，对之前提到的失真点，此时它位于全月图和半月图编码的交界处，因此，解码器希望失真点既能尽量与全月图相似，又能尽量与半月图相似，因此它的还原结果将是两种图的折中(例如 3/4 的全月图)。通过这个例子发现给编码器增加一些噪声，可以有效地覆盖失真区域。然而，引入区域噪声的方法还不够充分，因为噪声的范围总是有限的，不可能覆盖所有采样点。为了解决此问题，可以尝试将噪声的范围无限延伸，以使对于每个样本，其编码能够覆盖整个编码空间，但是需要确保，在原始编码附近的编码点具有最高的概率，随着离原始编码点的距离的增加，编码的概率逐渐减小。在这种情况下，图像的编码将从原来离散的编码点变成一个连续的编码分布曲线，如图 5-4 所示。

这种将图像编码由离散变为连续的方法，就是变分自编码的核心思想。接下来介绍 VAE 的模型架构，以及 VAE 如何实现上述构思。

VAE 架构就是在原本的 AE 结构上，为编码添加合适的噪声，如图 5-5 所示。首先将 input 输入编码器，计算出两组编码：一组编码为均值编码 $m=(m_1,m_2,m_3)$，另一组为控

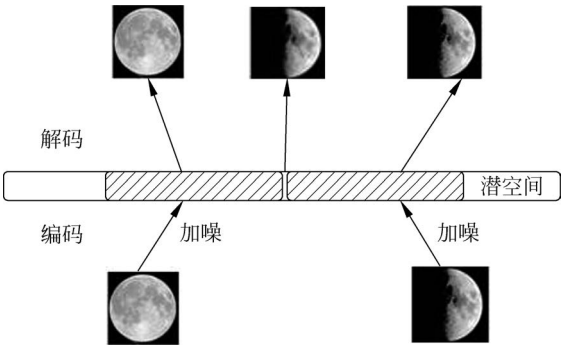

图 5-3 自编码中噪声的引入

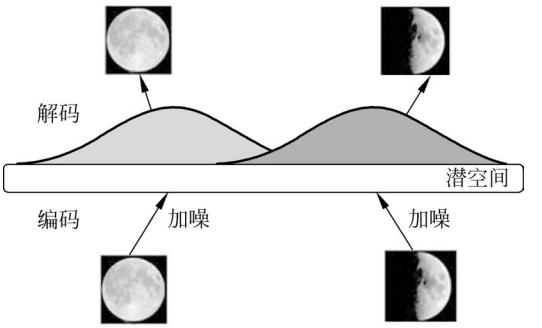

图 5-4 编码分布曲线

制噪声干扰程度的方差编码 $\sigma=(\sigma_1,\sigma_2,\sigma_3)$，这两组参数分别通过两个神经网络计算得到，其中方差编码 σ 主要用来为噪声编码 $z=(e_1,e_2,e_3)$ 分配权重，在分配权重之前对方差编码 σ 进行了指数运算，主要因为神经网络学习出来的权重值是有正负值的，加入指数运算后可保证分配到的权重是正值。最后，将原编码 m 和经过权重分配后的噪声编码进行叠加，得到新的隐变量，再送入解码器。观察图 5-5 可以看出，损失函数这一项除了之前传统 AE 的重构损失以外，还多了一项损失：$\sum_{i=1}^{3}(\exp(\sigma_i)-(1+\sigma_i)+(m_i)^2)$。

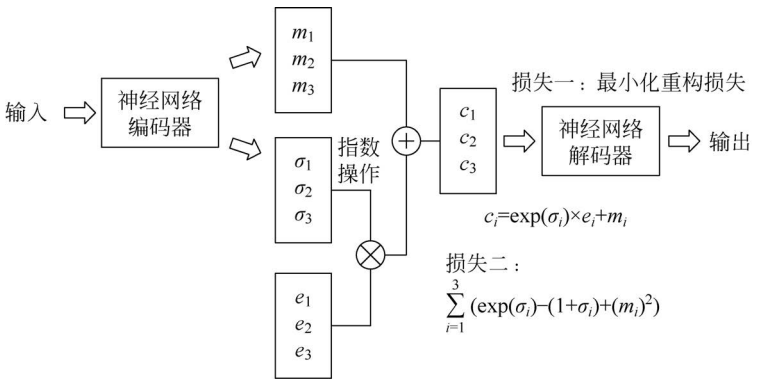

图 5-5 VAE 结构图

运用反证法的思想来推敲这个新损失的意义。当不引入这个损失函数时，模型会努力减少生成图片的重构误差来提高图片质量。为了实现这一点，编码器会期望减少噪声对生成图片的影响，降低任务难度，因此，它会倾向于给噪声分配较低的权重。如果没有任何约束限制，则网络只需将方差编码设置为接近负无穷大的值，从而消除噪声的影响，即 $\exp(\sigma_i)e_i=0$，此时 m_i 就等于它本身，模型就退化成了普通的自编码器，过拟合问题就会卷土重来。尽管此时模型的训练效果可能非常好，但生成的图片往往会非常糟糕。

为了方便理解，可以做一个生活中的类比。将变分自编码器(VAE)的工作过程类比为参加高考的过程。在学生准备高考的过程中，他们需要进行大量的模拟考试以提高最终的

考试成绩，这就像 VAE 在训练阶段所做的事情。模拟考试的题目和难度由老师安排，这能够公正地评估学生的学习能力。类似地，在 VAE 的训练过程中，它生成的数据的分布，即方差编码 σ，应由某个损失函数（可以理解为“老师”）来决定。如果没有老师的监督，让学生自己设置模拟考试的难度，则他们很可能将试题设得非常简单，以便得高分。这就像 VAE 在没有适当的损失函数约束时，可能倾向于降低噪声的影响，让模型重构误差尽可能地接近于零。因此，为了保证 VAE 在实际应用中的表现，而不是通过降低噪声影响来“投机取巧”，需要引入一个适当的损失函数。这个损失函数就像老师一样，监督 VAE 的训练过程，确保模型在适当的难度下进行训练，从而能够在复杂的真实世界任务中表现得更好。

由此可知，除了重构误差，VAE 还让所有的矩阵 $\boldsymbol{c}$ 都向标准正态分布看齐，这样就防止了 $\exp(\sigma_i)e_i$ 接近零，进而造成噪声为 0 的情况，保证 VAE 模型不会退化成自编码器，如图 5-6 所示。

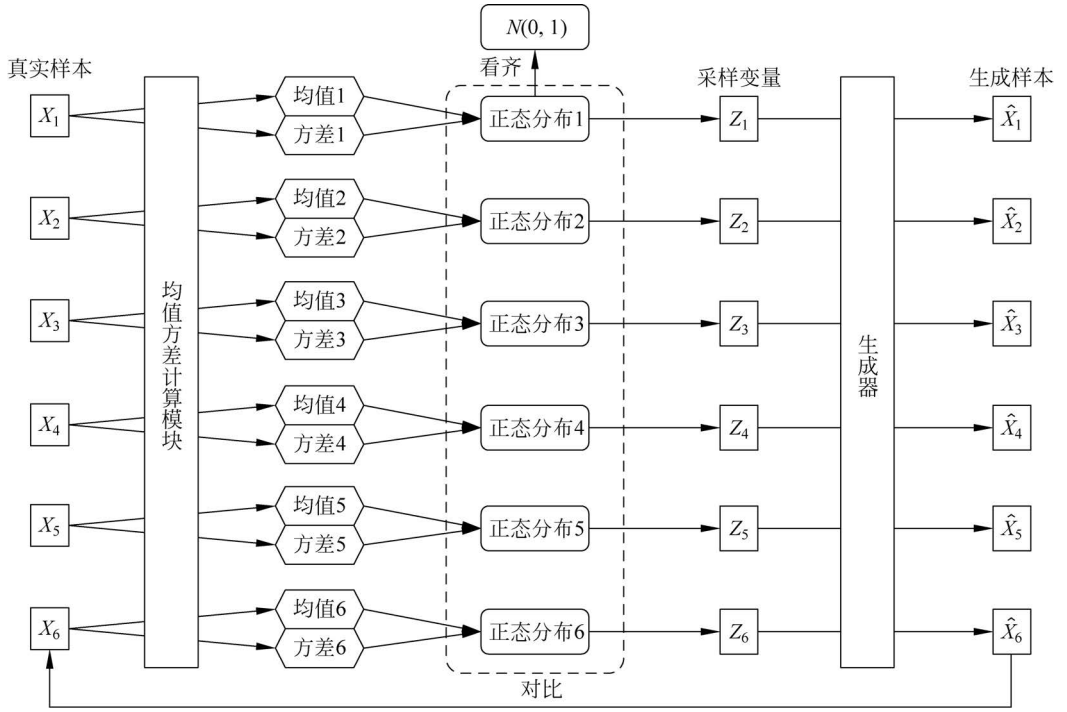

图 5-6　标准化 VAE 的参数分布图

让所有的 $\boldsymbol{c}$ 都向标准正态分布看齐最直接的方法就是在重构误差的基础上加入额外的损失。此时，KL 散度就派上用场了，可以让 $p(z|x)$（也就是 $\boldsymbol{c}$）和 $\mathcal{N}(0,I)$ 看齐，相当于一个约束，确保方差不为 0，所以加一个约束项 $\mathrm{KL}(q_\theta(z|x)\,\|\,\mathcal{N}(0,I))=\mathrm{KL}(p(z|x)\,\|\,p(z))$，这个约束项经过推导即为 VAE 中的第 2 项损失。推导过程，参见 5.2.2～5.2.6 节。

5.2.2　潜空间

在机器学习和深度学习领域中，由编码器将输入数据映射到低维向量空间。得到的低

维空间被称为潜空间(Latent Space),因为它包含了输入数据的隐藏特征和表示。

在深度学习中,编码器通常是一个神经网络,它通过学习将高维输入数据(如图像、音频或文本)转换为潜空间中的低维向量表示。这个低维向量表示捕捉了输入数据的重要特征和结构,其中每个维度可能对应着数据的某个抽象特征。

潜空间具有一些有用的属性。首先,潜空间具有较低的维度,因此可以更有效地表示数据,并且可以减少冗余信息,其次,潜空间的向量可以进行数学运算,例如向量加减法,这种运算在潜空间中对应着对输入数据的语义操作,例如在图像中添加或去除特定特征。这使潜空间成为生成模型和重构模型的重要组成部分。

潜空间在许多机器学习任务中发挥着重要作用,包括图像生成、图像重构、特征提取和数据压缩等。通过学习潜空间的结构和特征,可以实现更高级别的数据分析和操作。

一个好的潜空间应该具备以下几个特点。

(1) 有意义的表示:潜空间中的每个维度应该对应着输入数据的某个有意义的特征。这意味着相似的数据在潜空间中应该更接近,而不相关的数据应该更远离。例如,在图像领域,潜空间的某个维度可以表示图像中的颜色,另一个维度可以表示形状。这种有意义的表示使在潜空间中的运算和操作更加直观和可解释。

(2) 低维度:潜空间的维度应该相对较低,以便有效地表示数据并减少冗余信息。通过将高维数据映射到低维空间,可以提取数据中最重要的特征,并且可以更高效地进行计算和操作。

(3) 连续性:潜空间中的向量应该具有连续性,即在潜空间中相邻的向量应该对应着在输入空间中相似的数据。这种连续性使在潜空间中进行插值或插入新的向量时,能够产生合理和平滑的结果。例如,在图像生成任务中,通过在潜空间中对两个不同的向量进行线性插值,可以生成一个介于它们之间的新图像。

(4) 可操作性:潜空间中的向量应该具有可操作性,即可通过对向量进行数学运算实现对输入数据的语义操作。例如,在图像生成任务中,可以通过在潜空间中对某个向量的特定维度进行增减操作,来改变生成图像中的某个特征,如颜色、形状等。

(5) 一致性:潜空间应该在不同的输入数据之间保持一致,即相同类型的数据在潜空间中应该有相似的表示。这样可以确保潜空间的泛化能力,使相似的数据具有相似的表示,而不同类别的数据有明显的区分。

一个好的潜空间设计可以使在该空间中的数据表示更加有效、有意义,并且可以支持各种任务,包括生成、重构、插值和语义操作等。潜空间和图像生成模型之间有密切的关系,潜空间是图像生成模型的关键组成部分之一。

图像生成模型旨在从潜空间中生成逼真的图像。这些模型通常使用生成对抗网络或变分自编码器等方法。在这些模型中,一个重要的步骤是将输入图像映射到潜空间中的向量表示,这个过程通常由编码器完成。编码器将输入图像转换为潜空间中的向量表示,其中每个向量维度对应着图像的某个特征。这个向量可以被看作图像的隐含表示或特征向量。此特征向量可以被用于进行各种图像操作,例如生成新的图像、重构原始图像或者在潜空间中

进行插值操作。

生成模型的另一部分是解码器，它的任务是将潜空间中的向量转换回图像空间。解码器接收潜空间中的向量，并将其解码为逼真的图像。这个过程可以被视为对潜空间向量的逆映射。

通过训练生成模型，可以学习到一个优化的潜空间表示，其中潜空间中的向量可以被解码成高质量的图像，从而可以在潜空间中进行图像操作，例如通过在潜空间中调整特定维度的值来改变图像的特征，或者通过在潜空间中进行插值来生成介于两个向量之间的新图像。

总之，潜空间为图像生成模型提供了一个有效的表示和操作图像的方式，可以通过对潜空间中的向量进行操作来生成和操纵图像。

5.2.3　最大似然估计

最大似然估计(Maximum Likelihood Estimation，MLE)是一个由样本来估计总体的算法，就像我们想根据有限的图片样本估计它总体的潜空间表示，然而，最大似然估计是参数估计方法，而我们求的是概率分布，该如何进行应用呢？这里需要运用统计建模方法，假设样本 $p(x)$服从某种分布，对于连续数据最常用的就是高斯分布$\mathcal{N}(\mu,\sigma^2)$。当 $p(x)$的分布确定后，概率分布估计问题就被转换为参数估计问题，因此联合概率分布 $p(x,z)$可以分解为

$$L(\theta;X)=P(X;\theta)=\prod_{i=1}^{n}p(x_i;\theta) \tag{5-1}$$

其中，θ 可以看作 z 分布的参数，式(5-1)也被称为似然函数。最大似然估计准则是最大化似然函数，这等同于最小化负对数似然函数，取对数是为了将连乘符号变成连加符号，方便计算，公式如下：

$$\hat{\theta}=\underset{\theta}{\operatorname{argmax}}L(\theta;X)=\underset{\theta}{\operatorname{argmin}}-\sum_{i=1}^{n}\ln p(x_i;\theta) \tag{5-2}$$

在优化的过程中需要求解关于参数 θ 的梯度：

$$\nabla_\theta L(\theta;X)=-\nabla_\theta\sum_{i=1}^{n}\ln p(x_i;\theta)=-\sum_{i=1}^{n}\nabla_\theta\ln p(x_i;\theta) \tag{5-3}$$

按照梯度下降的方法，最大似然估计就可以求出参数 θ，得到概率分布，最后采样生成图片。

实际上神经网络的有监督训练与最大似然估计很相似。首先，读者需要明白两者的基本概念。

(1) 有监督训练：在有监督训练中，存在一组标记过的训练数据，每个训练样本都有一个对应的标签(或目标值)。目标是训练一个模型，当给定一个新的输入时，能够准确地预测出对应的标签。

(2) 最大似然估计：最大似然估计是一种用来估计一个概率模型参数的方法，其基本思想是，给定一组观测数据，应该选择那些使这组数据出现的可能性(似然性)最大的参数。

现在，让我们来看它们之间的关系：在很多情况下，神经网络的有监督训练可以看作在进行最大似然估计。例如，当使用交叉熵损失函数（一种常用于分类问题的损失函数）训练一个神经网络时，实际上就是在对模型的参数进行最大似然估计。因为在这种情况下，训练目标是最小化交叉熵损失函数，这等价于最大化数据的对数似然。简而言之，希望找到一组参数，使在这组参数下，观察到的实际标签出现的可能性达到最大。这也就是为什么说神经网络的有监督训练与最大似然估计的思路相同。它们都是在寻找一组参数，使给定这组参数，观察到的实际数据的可能性最大。

而且，在神经网络的有监督训练中，虽然不直接假设一个显式的数据生成分布，但在使用特定的损失函数（如均方误差或交叉熵）时，实际上隐含地做了一些关于数据和模型错误的分布假设。例如，使用均方误差损失函数通常假设了数据误差遵循高斯分布，而使用交叉熵损失函数则假设了分类任务中数据遵循伯努利或多项分布。

具体来讲，对于均方误差，其形式是预测值和实际值之差的平方的期望（或平均值）。如果将这种误差视为随机变量的“噪声”，MSE 就等价于这种噪声的方差。根据中心极限定理，当大量独立同分布的随机变量叠加时，其和的分布将接近正态分布（也就是高斯分布），因此，当使用 MSE 作为损失函数时，隐含地假设了噪声遵循高斯分布。对于交叉熵，它常常用于衡量两个概率分布之间的差异。在二元分类问题中，每个样本只有两种可能的类别，因此其目标值可以看作一个伯努利随机变量的实现结果。目标就是找到一个模型，其预测的概率分布尽可能地接近目标的伯努利分布，所以当使用交叉熵作为损失函数时，隐含地假设了数据遵循伯努利分布。同样，对于多分类问题，目标值可以看作多项分布的实现结果，因此使用交叉熵也就隐含了数据遵循多项分布的假设，所以虽然在神经网络训练的过程中并没有显式地声明这些分布假设，但这些假设确实是损失函数选择的结果，并且对模型的训练和预测有重要影响。

损失函数的选择对模型的训练影响很大，这同样也是最大似然估计存在的一个致命的问题：该方法是有假设存在的，假设了 $p(x)$服从某种分布。分布的选择是需要领域知识或先验的，需要对生成过程很了解，否则如果选择的分布和真实分布不一致，则结果可能很差。真实世界的问题通常是很复杂的，没有人能够完全了解它的生成过程，同时也没有能够描述如此复杂过程的概率分布形式，因此假设的分布基本是错误的。

5.2.4 隐变量模型

隐变量就是潜空间中的向量。它可以作为解决图片生成这一困难问题的跳板，这个思路在数学中非常常见，例如我们想直接根据变量 a，求解结果 b 非常困难，而由 a 求解 c 和由 c 求解 b 都很简单，那么可以选择绕开最难的部分，而选择 $a \rightarrow c \rightarrow b$ 的求解方法。在图像生成中体现为根据图片样本直接求解数据分布 $p(x)$很难，那么可以通过隐变量实现，例如考虑手写体数字例子，一般在写数字的时候会首先想到要写哪个数字，同时脑子里想象它的样子，然后才是写下来的图像。这个过程可以总结成两个阶段：先决定数字及其他影响因

素，用隐变量 z 来表示；再根据隐变量 z 生成数字图像。这就是隐变量模型，用数学描述为

$$P(X)=\int P(X\mid z;\theta)P(z)\mathrm{d}z \tag{5-4}$$

通常假定 z 服从正态分布 $z\sim\mathcal{N}(0,I)$。$P(X|z;\theta)$可以转换成 $f(z;\theta)$，即用一个参数为 θ 的函数去计算样本 X 的概率分布 $P(X|z;\theta)$。这里采用条件分布形式是因为它可以显式地表明 X 依赖 z 生成。隐变量模型把概率密度估计问题转换为函数逼近问题。首先分开解释这两部分。

(1) 概率密度估计：这是一个统计问题，目标是从有限的样本数据中估计出整个概率分布。对于复杂的、多维的数据，直接估计这个分布可能是困难的。

(2) 函数逼近：这是一个优化问题，目标是找到一个函数，使其在某种意义下尽可能地接近给定的目标函数。在神经网络中，这个函数通常是通过一个可微分的参数化模型(例如深度神经网络)来表示的。

隐变量模型，如变分自编码器，可以把概率密度估计问题转换为函数逼近问题。在VAE中，并不直接估计数据的概率密度，而是假设存在一个隐含的潜在空间，数据是由这个潜在空间中的点通过一个可微分的生成网络生成的，即 VAE 的解码器。这个生成网络可以被看作一个函数，它把潜在空间中的点映射到数据空间。我们的目标是找到这个生成网络的参数，使由它生成的数据的分布尽可能地接近真实数据的分布。这就把一个概率密度估计问题转换为一个函数逼近问题。

这样的做法有几个优点。首先，函数逼近问题可以通过诸如梯度下降等优化方法来解决，其次，可以利用深度神经网络的强大表示能力来建模复杂的、高维的数据分布。最后，通过学习一个潜在空间，可以得到数据的一种低维表示，这对于许多任务，例如生成模型、异常检测等都是有用的。

隐变量模型背后的关键思想是：任何一个概率分布经过一个足够复杂的函数后可以映射到任意概率分布。如示例中，z 服从标准高斯分布，采样后经过函数 $f(z;\theta)$的变换后可以变成手写体数字的真实分布 $P(X)$。

通过隐变量模型，最大似然估计的问题已经被绕过去了，不再需要指定复杂的概率分布形式，也不怕出现分布不一致的情况。只需求解函数 $f(z;\theta)$，这个函数看似很难求解，不过我们有神经网络这一利器，深度学习最有魅力的一点就是拟合能力，但凡直接求解很困难的问题都可以交给神经网络进行拟合。

下面采用最大似然的思想去优化隐变量模型。因为 $P(X|z;\theta)$是确定性函数，最大化 $P(X)$等于最大化 $P(z)$。我们想要的是如果从 $P(z)$采样的 z 的概率很大，则它生成的 X 出现在数据集的概率也很大。隐变量模型的负对数似然函数为

$$L(\theta;X)=-\sum_{i=1}^{n}\ln p(x_i;\theta)=-\sum_{i=1}^{n}\ln\int p(x_i\mid z;\theta)p(z)\mathrm{d}z \tag{5-5}$$

关于 θ 的梯度为

$$\nabla_\theta L(\theta;X)=-\sum_{i=1}^{n}\nabla_\theta \ln\int p(x_i\mid z;\theta)p(z)\mathrm{d}z=-\sum_{i=1}^{n}\frac{\int \nabla_\theta p(x_i\mid z;\theta)p(z)\mathrm{d}z}{\int p(x_i\mid z;\theta)p(z)\mathrm{d}z} \tag{5-6}$$

根据式(5-6)的梯度公式就可以优化参数,得到最后的隐变量模型。

但隐变量模型存在问题:在计算$\nabla_\theta L(\theta;X)$的过程中需要计算分子和分母的积分,为了使$z$能表达更多的信息,通常假设$z$是一个连续的随机变量。在这种情况下,由于一般无法直接求解准确值,因此会存在计算困难的问题。

5.2.5 蒙特卡洛采样

隐变量模型在计算梯度时存在积分难计算的问题。针对求积分问题,很难计算准确值,因此通常采用蒙特卡洛采样去近似求解。

原来的积分可以写成期望的形式$\int p(x\mid z;\theta)p(z)\mathrm{d}z=\mathbb{E}_{z\sim p(z)}[p(x\mid z;\theta)]$,然后利用期望法求积分,步骤如下。

(1) 从$p(z)$中多次采样$z_1,z_2,\cdots,z_m$。

(2) 根据$p(x\mid z;\theta)$计算$x_1,x_2,\cdots,x_m$。

(3) 求x的均值。用数学表达为

$$\int p(x\mid z;\theta)p(z)\mathrm{d}z=\mathbb{E}_{z\sim p(z)}[p(x\mid z;\theta)]\approx\frac{1}{m}\sum_{j=1}^{m}p(x_j\mid z_j;\theta) \tag{5-7}$$

通过对z多次采样,可以计算$\nabla_\theta L(\theta;X)$的近似值。简单来讲,蒙特卡洛采样就是通过样本的均值来近似总体的积分。

蒙特卡洛采样存在的问题是:采样次数一般需要很大,这主要是由两个因素引起的。

(1) 高维度:对于高维问题,由于"维度的诅咒",可能需要指数级的采样数才能在所有维度上获得足够的覆盖。这是因为在高维空间中,大部分的体积在靠近边界的区域,所以需要大量的样本才能精确地估计整个空间的性质。

(2) 稀疏区域:如果感兴趣的分布在某些区域中非常稀疏,则大多数的蒙特卡洛样本可能会落入不关心的区域,而真正关心的区域可能会被严重地欠采样。这会导致估计结果严重偏离真实值。

解决这两个问题的方法可以缩小z的取值空间。缩小$p(z)$的方差σ^2,那么z的采样范围就会缩小,采样的次数m也不需要那么大。同时,也可能把生成坏样本的z排除,生成更像真实样本的图像。这其实就是下面要介绍的VAE的原理。

5.2.6 变分推断

继续5.2.5节的问题,怎么能缩小z的取值空间呢?原来z从先验概率分布$p(z)$中采样,现在可以考虑从z的后验概率分布$p(z|X)$中采样。具体来讲,给定一个真实样本X,假设存在一个专属于X的分布$p(z|X)$(后验分布),并进一步假设这个分布是独立的、多

元的正态分布。

如何理解后验概率 $p(z|X)$会比先验概率 $p(z)$更好呢？在变分自编码器中，“后验概率”通常是指在给定观察数据的情况下，隐变量的概率分布，而“先验概率”则是在没有观察数据的情况下，隐变量的概率分布。后验概率包含了更多的信息。先验概率只反映了在没有观察数据之前对隐变量的知识或者假设，它通常被设置为一种简单的分布，如高斯分布或者均匀分布，而后验概率则是在观察到数据之后，对隐变量的最新认识，它包含了数据的信息。

例如，假设我们的任务是对人脸图片进行建模，隐变量可能代表一些人脸的特性，如性别、年龄等。在没有看到任何图片的情况下，可能假设所有的性别和年龄都是等可能的，这就是先验概率，但是当看到一些图片之后，可能会发现实际上某些性别或者年龄的人脸图片更常见，这就是后验概率。

因此，当我们说后验概率比先验概率更好时，其实是说，后验概率包含了更多的来自数据的信息，能够更准确地反映真实世界的情况。在变分自编码器中，编码器的目标就是学习表示后验概率分布。

从数学角度出发，如何求出后验概率分布 $p(z|X)$呢？求隐变量的后验分布是变分推断的一个核心问题。

一般是无法准确地求出后验分布的，但是变分推断可以用另一个分布 $q_\theta(z|X)$近似估计 $p(z|X)$，然后从 $q_\theta(z|X)$中采样来近似从 $p(z|X)$中采样。这种方法是用一个函数近似另一个函数，其实就是用神经网络来近似概率分布参数。用变分法的术语来讲，变分推断的主要思想是通过优化来近似复杂的概率分布。这种方法的名字“变分”来自微积分中的变分法，这是一种数学方法，用于寻找函数或者泛函（函数的函数）的极值。

在变分推断的背景下，我们有一个复杂的概率分布，通常是后验概率分布 $p(z|X)$，这里 z 是隐变量，X 是观察到的数据。目标是找到一个相对简单的分布（例如高斯分布），称其为 $q_\theta(z|X)$，用它来近似真实的后验分布，其中，θ 表示分布的参数，需要找到合适的 θ 来最大化这种近似的准确性。

那么，如何度量这种“近似”的准确性呢？这就需要用到 KL 散度，它是一种衡量两个概率分布之间“距离”的方法。目标就是找到参数 θ，使 $q_\theta(z|X)$和 $p(z|X)$之间的 KL 散度最小。

然后这个最小化问题可以通过梯度下降等优化算法来求解。在每步中都会计算 KL 散度关于 θ 的梯度（这就是变分），然后沿着梯度的负方向更新 θ，以减小 KL 散度。这个过程就像在函数空间中“下山”，因此被称为函数空间的梯度下降。

由此可知，变分推断就是一种用优化的方式来逼近复杂的概率分布的方法。通过在函数空间中进行梯度下降，找到一个可以用来近似真实分布的简单分布，但是，还有一个很大的问题：$p(z|X)$ 是未知的，所以无法直接计算 KL 散度。解决这个问题的方法是通过引入一个叫作证据下界（Evidence Lower Bound，ELBO）的量。ELBO 是模型对数似然的一个下界，它与 KL 散度的和是一个常数，这个常数就是观测数据的对数似然，因此，最大化 ELBO

等价于最小化 KL 散度。

ELBO 可以写成以下形式：

$$\text{ELBO}=E_{q_\theta(z|X)}[\log p(X\mid z)]-\text{KL}(q_\theta(z\mid X)\parallel p(z)) \tag{5-8}$$

其中，第 1 项 $E_{q_\theta(z|X)}[\log p(X\mid z)]$是重构误差，代表了生成的数据与实际数据的相似程度，具体来讲，$X\sim q_\theta(z|X)$表示有一个 X 可以根据分布 $q_\theta(z|X)$采样一个 z，这个过程可以理解为把 X 编码成 z，此过程被称为 VAE 模型的编码器。$p(X|z)$表示根据 z 生成输出结果，此过程被称为 VAE 模型的解码器。整体表示先将给定 x 编码成 z 再重构得到输出结果，这个过程的期望，被称为重构误差。如果这个期望很大，则表明得到的 z 是 X 的一个好的表示，能够抽取 X 足够多的信息来重构输出结果，让它与 X 尽可能相似。第 2 项 $\text{KL}(q_\theta(z|X)\parallel p(z))$是 $q_\theta(z|X)$和先验分布 $p(z)$的 KL 散度，代表了隐变量的分布偏离先验分布的程度。

可以看到，计算 ELBO 并不需要知道后验分布 $p(z|X)$的具体形式，只需知道数据的生成模型 $p(X|z)$和隐变量的先验分布 $p(z)$，而这两者通常是可以设定的，所以可以计算，因此，可以通过最大化 ELBO 实现变分推断。

最后，证据下界的推导过程是基于 Jensen 不等式和 KL 散度的性质，了解即可，不要求掌握。

先定义一个模型的对数似然：

$$\log p(X)=\log\int p(X,z)\mathrm{d}z \tag{5-9}$$

其中，z 是隐变量，X 是观察数据。

再引入变分分布 $q_\theta(z|X)$，然后对上式两边乘以 $q_\theta(z|X)$并对 z 积分，得到：

$$\log p(X)=\log\int q_\theta(z\mid X)\frac{p(X,z)}{q_\theta(z\mid X)}\mathrm{d}z \tag{5-10}$$

由于对数函数是凹函数，所以这里可以使用 Jensen 不等式，Jensen 不等式说明对于凹函数 f 和随机变量 Z，有 $E[f(Z)]\leqslant f(E[Z])$，所以：

$$\log p(X)\geqslant\int q_\theta(z\mid X)\log\frac{p(X,z)}{q_\theta(z\mid X)}\mathrm{d}z \tag{5-11}$$

不等式(5-11)的右边，就是定义的 ELBO：

$$\text{ELBO}=E_{q_\theta(z|X)}[\log p(X\mid z)]-\text{KL}(q_\theta(z\mid X)\parallel p(z)) \tag{5-12}$$

它由两部分组成，第一部分 $E_{q_\theta(z|X)}[\log p(X|z)]$是对数似然的期望，第二部分 $\text{KL}(q_\theta(z|X)\parallel p(z))$是变分分布与先验分布之间的 KL 散度。

由此可知，ELBO 实际上是对数似然的一个下界。在变分推断中，试图最大化 ELBO，这等价于最小化变分分布与真实后验分布之间的 KL 散度，从而使变分分布尽可能地接近真实的后验分布。

实际上，ELBO 就是 VAE 的损失函数，VAE 的训练目标就是最大化 ELBO(证据下界)。负 ELBO 由两部分组成：第一部分是期望的重构误差，它衡量的是模型生成的数据与真实数据的匹配程度；第二部分是 KL 散度，它衡量的是隐变量的分布偏离先验分布的程度。

重构误差可以用各种不同的方式来计算，例如使用均方误差或者交叉熵损失。至于KL散度，由于通常假设先验分布是标准正态分布，所以KL散度可以用隐变量的均值和方差来显式地进行计算。值得注意的一点是，在VAE简明推导中提到的KL散度损失是：

$$\frac{1}{2}\sum_{i=1}^{l}(\exp(\sigma_i)-(1+\sigma_i)+(\mu_i)^2) \tag{5-13}$$

它是在假设隐变量服从高斯分布时的结果。在变分自编码器中，通常会假设隐变量 z 的先验分布是标准正态分布，即 $p(z)=\mathcal{N}(0,I)$。还会假设编码器给出的后验分布也是一个高斯分布，即 $q_\theta(z|X)=\mathcal{N}(\mu,\sigma^2 I)$，其中 μ 和 σ 是由神经网络给出的。在这种情况下，$q_\theta(z|X)$和 $p(z)$之间的KL散度可以显式地计算出来，结果就是式(5-13)，其中，μ_i 是隐变量 z 第 i 个维度的均值，σ_i 是第 i 个维度的标准差的对数(在实际实现中，神经网络直接输出的通常是 $\log\sigma^2$，为了保证其非负)，l 是隐变量的维度。

下面给出具体的推导，首先，两个高斯分布之间的KL散度的公式为

$$\mathrm{KL}(\mathcal{N}(\mu_1,\sigma_1^2)\parallel\mathcal{N}(\mu_2,\sigma_2^2))=\frac{(\mu_1-\mu_2)^2+\sigma_1^2-\sigma_2^2+2(\log\sigma_2-\log\sigma_1)}{2\sigma_2^2} \tag{5-14}$$

在VAE中，设定先验分布 $p(z)=\mathcal{N}(0,1)$，即 $\mu_2=0,\sigma_2^2=1$，而后验分布 $q_\theta(z|X)=\mathcal{N}(\mu,\sigma^2)$，即 $\mu_1=\mu,\sigma_1^2=\sigma^2$，将它们代入式(5-14)，可以得到：

$$\mathrm{KL}(q_\theta(z\mid X)\parallel p(z))=\frac{\mu^2+\sigma^2-1-\log\sigma^2}{2} \tag{5-15}$$

这就是KL散度的公式，但是，由于通常使用 $\log\sigma^2$(记作 σ)作为神经网络的输出(为了确保 σ^2 的非负性)，所以可以对式(5-15)进行一些变换：

$$\mathrm{KL}(q_\theta(z\mid X)\parallel p(z))=\frac{\mu^2+\mathrm{e}^\sigma-1-\sigma}{2} \tag{5-16}$$

这就是VAE论文中所给出的KL散度项的公式。如果有 l 个隐变量，则整个KL散度项就是所有维度上的KL散度之和，公式如下：

$$\sum_{i=1}^{l}\left(\frac{1}{2}(\mu_i^2+\exp(\sigma_i)-1-\sigma_i)\right) \tag{5-17}$$

这就是在VAE损失函数中需要最小化的KL散度项的公式。

5.3　NLP中的编码器-解码器：Seq2Seq模型

Seq2Seq模型可以被认为是一种编码器-解码器模型的变体，其特别适用于处理序列到序列的任务，编码器将输入序列映射为一个固定长度的向量表示，解码器则使用这个向量表示来生成输出序列。它已被广泛地应用于机器翻译、对话系统、语音识别等自然语言处理任务。Seq2Seq模型的结果框架如图5-7所示。

以机器翻译任务为例来讲解图5-7的Seq2Seq结构。假设现在的模型输入是文本序列hello world，想得到输出结果为“你好，世界”。

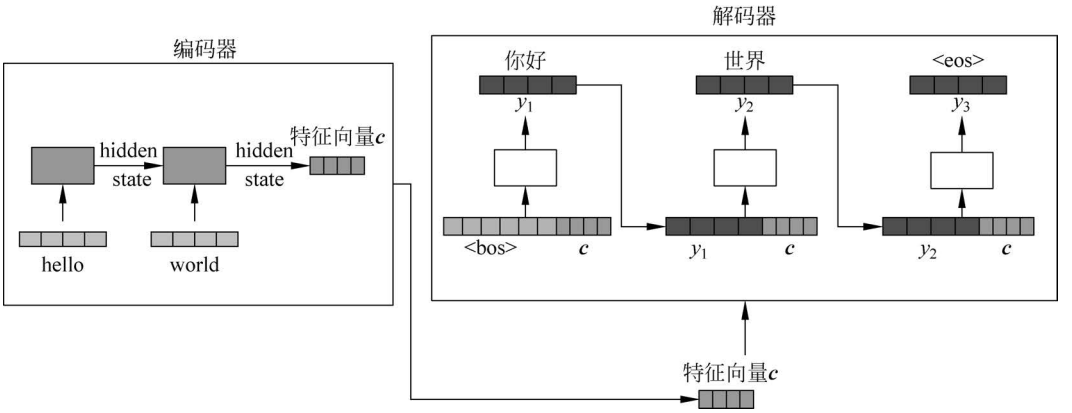

图 5-7 Seq2Seq 模型的结果框架

5.3.1 Seq2Seq 编码器

在 Seq2Seq 模型中，编码器负责将输入序列映射到一个特征向量 $\boldsymbol{c}$，如图 5-8 所示。

希望通过训练可以让该向量提取到输入信息的语义特征，将来送入解码器中作为解码器的一部分输入信息。编码器通常采用循环神经网络或卷积神经网络来处理输入序列。

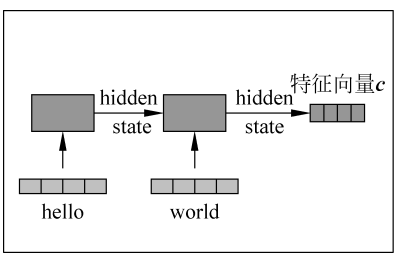

图 5-8 Seq2Seq 编码器

以 RNN 为例，编码器由多个时间步组成，每个时间步接收输入序列中的一个元素，并生成一个隐藏状态。这个隐藏状态通过一个非线性函数，例如 tanh 或 ReLU 激活函数，被映射到一个连续的向量表示。这个过程可以表示为

$$h_t = f_{\text{enc}}(x_t, h_{t-1}) \tag{5-18}$$

其中，x_t 是输入序列的第 t 个元素，h_t 是编码器在时间步 t 的隐藏状态，h_{t-1} 是前一个时间步的隐藏状态，f_{enc} 是一个非线性函数，它将输入序列的元素和前一个隐藏状态作为输入，并返回当前隐藏状态。

在许多 Seq2Seq 模型中，最后一个时间步的隐藏状态被视为输入序列的向量表示，即

$$z = h_T \tag{5-19}$$

其中，T 是输入序列的长度。这个向量将用作解码器的部分输入信息。除了基本的 RNN 编码器，还有许多其他类型的编码器，例如双向 RNN 编码器，它可以同时处理正向和反向序列，以捕捉输入序列中的双向信息。此外，还有一些基于卷积神经网络的编码器和 Transformer 编码器。这些编码器在不同的任务和数据集上具有不同的优点和局限性。简单来讲，编码器只是一种概念，具体使用哪种算法没有规定，只要能对输入数据进行特征提取即可，可以根据当前输入信息的特点来选择适合的算法作为编码器。

5.3.2　Seq2Seq 解码器

在 Seq2Seq 模型中，解码器负责生成输出序列，它通常也采用循环神经网络或其他变体来处理输出序列。与编码器类似，解码器也由多个时间步组成。在每个时间步中，解码器使用前一个时间步的输出元素和当前时间步的隐藏状态来生成当前时间步的输出元素。这个过程可以表示为

$$y_t = f_{\text{dec}}(y_{t-1}, s_t) \tag{5-20}$$

其中，y_t 是输出序列的第 t 个元素，s_t 是解码器在时间步 t 的隐藏状态，y_{t-1} 是前一个时间步的输出元素，f_{dec} 是一个非线性函数，它将前一个输出元素和当前隐藏状态作为输入，并返回当前输出元素，如图 5-9 所示。

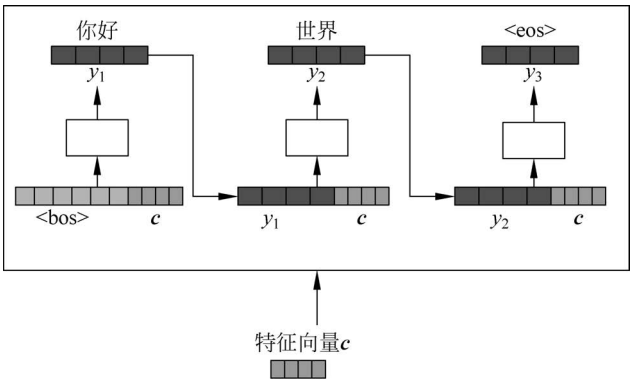

图 5-9　Seq2Seq 解码器

在解码器的初始时刻，通常会将编码器输出的特征向量 $\boldsymbol{c}$ 和起始向量< bos >拼接到一起，作为初始时刻的隐藏状态 s_0，并计算得到当前时刻的输出信息 y_1。在下一时刻 t_1 会将编码器输出的特征向量 $\boldsymbol{c}$ 和 y_1 拼接成一个向量作为当前时刻 t_1 的输入，并计算得到当前时刻的输出 y_2，以此类推，直到模型预测出结束向量< eos >时停止。不难发现，解码器的输入信息不仅包含了当前要翻译的单词，还包含了当前这句话的上下文语义信息；解码器会结合这两部分信息预测当前的输出，这就是解码器翻译文本为什么有效的原因。

5.3.3　Seq2Seq 的 Attention 机制

在 Seq2Seq 模型中，Attention 机制通常是在解码器端使用的。具体来讲，解码器在生成每个输出时会根据当前的解码状态和编码器中每个输入的状态计算一个注意力权重，用于表示当前输出与输入序列中每个位置的相关程度，如图 5-10 所示。

图 5-10 表示的是一个常规的 Seq2Seq 模型，只是将编码器的两个中间时刻输出的隐藏状态(O_1，O_2)也画在图中。考虑这样一个问题：将编码器最后时刻输出的隐藏状态，即特征向量 $\boldsymbol{c}$ 作为上下文语义信息送入解码器，并在解码器翻译每个单词时都用这同一个特征向量 $\boldsymbol{c}$ 作为输入，这种方式真的是最佳方案吗？

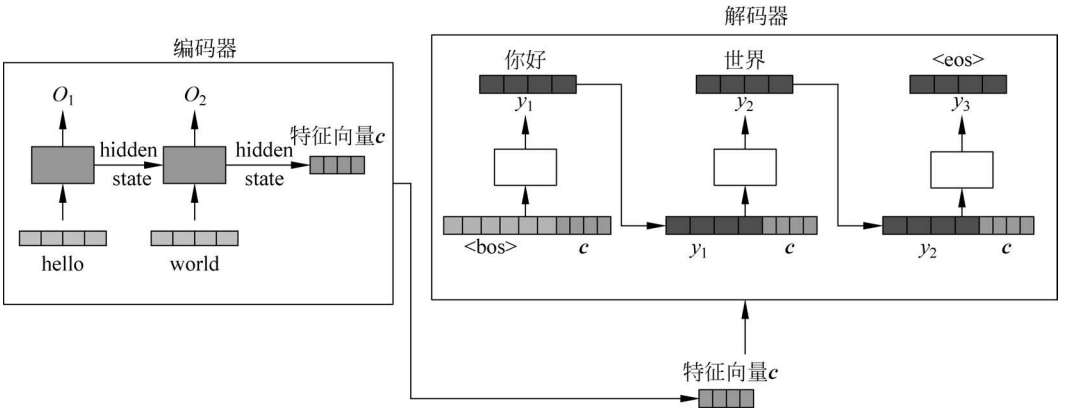

图 5-10 Seq2Seq 模型

(1) 考虑将特征向量 $\boldsymbol{c}$ 作为上下文语义信息是否合适?

由于 RNN 的特性,特征向量 $\boldsymbol{c}$ 实际上可以包含之前时刻隐藏状态 O_1,O_2 的信息,但这是因为此序列较短。如果是一个很长的序列,例如 O_1,O_2,O_3,…,O_n,则此时,特征向量 $\boldsymbol{c}$ 并不能很好地保存之前所有中间时刻隐藏状态的信息。有一种解决方案是将之前所有输出的隐藏状态加起来作为特征向量 $\boldsymbol{c}$,而不是简单地只用最后一个时刻的隐藏状态,即 $\boldsymbol{c}=O_1+O_2+O_3+\cdots+O_n$。

(2) 继续考虑在解码器翻译每个单词时都用这同一个求和得到的特征向量 $\boldsymbol{c}$ 作为输入是否合适?

语言这种数据其实很复杂,既存在局部相关性,即跟某个单词关系最大的单词肯定是它附近的单词,又存在远距离依赖,即当前单词也可能与较远处的单词相关。不管是哪种情况都说明句子中,单词之间是按照不同的重要程度进行相互依赖的。例如将图 5-10 的例句稍微变长一些:"hello,world is beautiful"翻译成"你好,世界是美好的"。在翻译"美丽"时,它可以与 beautiful 本身 90%相关,跟 is 这个词 1%相关,跟 world 这个词 7%相关等,即翻译"美丽"时,抓取原文的语言信息可能是"0.9×beautiful+0.01×is+0.07×world+…",而在翻译"是"这个单词时,抓取原文的语言信息是不同的,可能是"0.001×beautiful+0.99×is+0.005×world+…"。

由此可知,在解码器翻译每个单词时都用这同一个特征向量 $\boldsymbol{c}$ 作为上下文信息送进输入中是不合适的,如图 5-10 所示,在翻译得到 y_1 时,特征向量 $\boldsymbol{c}$ 可能等于 $0.82\times O_1+0.18\times O_2$,在翻译得到 y_2 时,特征向量 $\boldsymbol{c}$ 可能等于 $0.07\times O_1+0.93\times O_2$。推广下去,在翻译得到 y_n 时,特征向量 $\boldsymbol{c}$ 可能等于 $a_1\times O_1+a_2\times O_2+a_3\times O_3+\cdots+a_n\times O_n$,其中[$a_1$,$a_2$,$a_3$,…,$a_n$]就是要求解的注意力分数,它应该与要翻译的输入文本的向量[O_1,O_2,O_3,…,O_n]维度一致,这样才能通过相乘的方法进行重要程度的重分配。

问题的关键来了,如何对注意力分数[a_1,a_2,a_3,…,a_n]进行建模?

假设输入序列是 x_1,x_2,…,x_n,对应的编码器隐藏状态为 h_1,h_2,…,h_n。解码器的当

前隐藏状态为 s_t。首先计算解码器隐藏状态 s_t 与所有编码器隐藏状态 h_i 之间的相似性分数 $e_{t,i}$。可以通过点积、加权点积或其他相似性度量来进行计算，具体的相似性分数函数详见 4.2.5 节的 Attention-Based RNN 模型：

$$e_{t,i}=\text{score}(s_t,h_i) \tag{5-21}$$

s_t 和某些 h_i 越相似，意味着这些语义信息 h_i 就是在翻译文本 x_t 时应该着重关注的地方。就像在翻译“美丽”时，应该着重注意输入信息 beautiful，当然也不能忽略 world 等其他单词。接下来，将这些分数转换为权重，通过 Softmax 函数进行归一化：

$$\alpha_{t,i}=\frac{\exp(e_{t,i})}{\sum_{j=1}^{n}\exp(e_{t,i})} \tag{5-22}$$

其中，$\alpha_{t,i}$ 可以看作解码器在时间步 t 时关注编码器隐藏状态 h_i 的权重。接着计算加权和的上下文向量 c_t，公式如下：

$$c_t=\sum_{i=1}^{n}\alpha_{t,i}h_i \tag{5-23}$$

上下文向量 c_t 捕获了输入序列与解码器当前时间步的相关性，然后将上下文向量 c_t 与解码器的隐藏状态 s_t 结合起来，以生成下一个输出单词的概率分布，公式如下：

$$y_t=\text{Softmax}(W(s_t,c_t)+b) \tag{5-24}$$

其中，W 和 b 是需要学习的权重和偏置项。通过这种方式，注意力机制帮助 Seq2Seq 模型关注输入序列的重要部分，从而提高预测性能。

5.3.4　Seq2Seq 的 Teacher Forcing 策略

Teacher Forcing 用于训练阶段，主要针对 Seq2Seq 模型的解码器结构，神经元的输入包括上一个神经元的输出，如果上一个神经元的输出是错误的，则下一个神经元的输出也很容易是错误的，导致错误会一直传递下去并累加，而 Teacher Forcing 可以在一定程度上缓解上面的问题。

在 Teacher Forcing 策略中，模型在训练期间以一定的概率使用真实的输出序列作为输入，而不是使用前一个时间步的输出元素。这个策略可以加速模型的收敛，因为它强制模型学习如何生成正确的输出序列，而不是仅仅学习如何在没有真实输出的情况下生成下一个输出元素。

具体来讲，假设有一个输入序列 x 和相应的输出序列 y，则在训练期间，使用以下步骤：首先将输入序列 x 输入编码器，生成一个向量表示 z。接着将 z 作为解码器的初始隐藏状态 s_0，并使用 Teacher Forcing 策略，将 y_{t-1} 作为解码器在时间步 t 的输入，生成当前时间步的输出元素 y_t，如图 5-11 所示，解码器 3 个时刻 t_0，t_1，t_2 的输出结果分别是 y_1，y_2，y_3。假设，这 3 个时刻对应的真值(Label)分别是“你好，世界，< eos >”。以 t_1 时刻为例，在正常的 Seq2Seq 的训练过程中，t_1 时刻的输入应该是 concat(y_1，c)，但是在 Teacher Forcing 策略的训练过程中，t_1 时刻的输入应该是 concat(你好，c)。这样做的好处是即使在 t_0 时刻模

型预测的 y_1 不准，也不会影响到 y_2 时刻的预测。

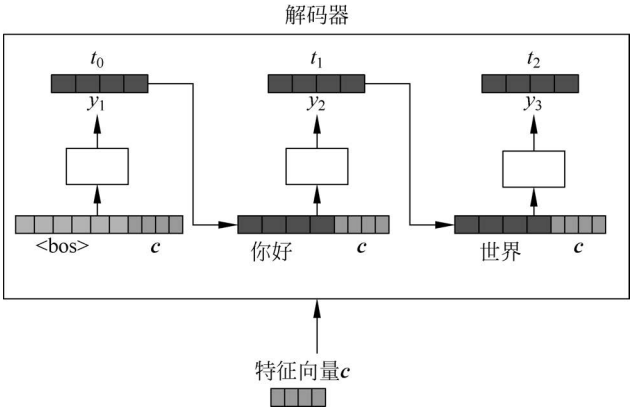

图 5-11 使用 Teacher Forcing 策略的 Seq2Seq

在测试期间不能使用 Teacher Forcing 策略，因为在测试期间没有真实的输出序列来作为输入。在测试期间，一般使用 Beam Search 来生成输出序列。具体来讲，在生成目标序列时，模型为每个时间步生成一个词。Beam Search 的核心思想是在每个时间步保留固定数量(Beam 宽度)的最优部分序列，而不是仅保留单个最优序列。通过这种方法，Beam Search 能够在搜索空间中更有效地探索，并有更大的概率找到全局最优的目标序列。

Beam Search 的过程如下。

(1) 初始化：将开始符号(如< bos >)作为输入传递给模型，并设置 Beam 宽度 K。

(2) 首次扩展：计算以开始符号为输入时所有可能的下一个词的概率，并保留概率最高的 K 个词，形成 K 部分序列。

(3) 迭代扩展：对于每部分序列，计算其下一个可能的词的概率，再将这些概率与之前的部分序列概率相乘，然后从所有可能的扩展中选择概率最高的 K 个作为新的部分序列。重复这个过程直到达到预定的最大长度，或者所有部分序列都已经生成了结束符号(如< eos >)。

(4) 结果选择：当搜索过程结束时，从 K 部分序列中选择概率最高的一个作为最终的目标序列。

需要注意的是，Beam Search 是一种启发式搜索方法，并不能保证找到全局最优解，但在实际应用中，相较于贪婪搜索(Greedy Search)或穷举搜索，Beam Search 能够在较短的时间内得到更好的结果。同时，Beam 宽度 K 的选择会影响搜索过程的效率和结果质量，K 值越大，搜索空间越大，结果质量可能越好，但计算成本也越高。

总之，Beam Search 是一种常用的解码策略，它可以在 Seq2Seq 模型中用于生成输出序列。与 Teacher Forcing 策略不同，Beam Search 用于测试阶段，以便在没有真实输出序列作为输入的情况下生成输出序列。

最后，在回归 Teacher Forcing 策略的讨论中，使用 Seq2Seq 模型的 Teacher Forcing 策略可以加速模型的训练，但也存在一些缺点。

(1) 不稳定性：由于训练和测试的输入方式不同，模型可能在测试期间生成不稳定的输出序列。在训练期间，模型使用真实的输出序列作为输入，而在测试期间，模型必须使用前一个时间步的输出作为输入。这种差异可能会导致模型在测试期间表现不佳。

(2) 训练偏差：由于 Teacher Forcing 策略只使用真实输出序列作为输入，它可能导致模型在训练期间出现偏差。在测试期间，模型必须学习如何在没有真实输出序列作为输入的情况下生成下一个输出元素。如果模型在训练期间没有充分地学习这种情况，则可能无法正确地生成输出序列。

(3) 限制序列长度：使用 Teacher Forcing 策略可能会限制生成的输出序列的长度。因为在训练期间，必须使用真实的输出序列作为输入，这意味着只能生成与训练数据中输出序列长度相同的序列。如果需要生成更长的序列，则必须修改模型或使用其他策略。

综上所述，虽然 Teacher Forcing 策略可以加速模型的训练，但也存在一些缺点。在实践中，需要根据具体的任务和数据集选择合适的解码策略，以实现更好的性能。

5.3.5　Seq2Seq 评价指标 BLEU

首先，BLEU(Bilingual Evaluation Understudy)是 Seq2Seq 模型的评价函数，而不是损失函数。在 Seq2Seq 模型中，通常使用一些常见的损失函数，例如交叉熵损失函数或均方误差损失函数，来衡量生成的输出序列与真实输出序列之间的差距。这些损失函数的目标是最小化生成序列的错误。

而 BLEU 是一种广泛使用的自动评价机器翻译系统性能的指标。BLEU 的主要原理是通过计算机器翻译结果与人工翻译参考之间的 n-gram 精度(n-gram Precision)来衡量翻译质量。同时，BLEU 还考虑了短句惩罚(Brevity Penalty，BP)，以防止模型生成过短的翻译结果。

BLEU 的计算过程如下。

1. n-gram 精度

对于不同的 n(如 1-gram，2-gram，3-gram，4-gram 等)，计算机器翻译结果中 n-gram 与参考翻译中 n-gram 的匹配程度。具体来讲，对于每个 n-gram，计算其在机器翻译结果中出现的次数与参考翻译中出现的次数的最小值，然后将这些最小值相加并除以机器翻译结果中 n-gram 的总数。这一步骤计算出的精度表示为 p_n。举例如下：

参考翻译 1：The cat is on the mat.

参考翻译 2：There is a cat on the mat.

机器翻译输出：The cat is on the mat.

现在，使用 BLEU 指标来计算机器翻译的质量。

首先需要确定 n-gram 的大小。假设选择 2-gram 作为 n-gram 的大小，然后需要计算机器翻译输出中的 2-gram 与参考翻译中的 2-gram 的重合度。

对于参考翻译 1，其 2-gram 序列为{The cat，cat is，is on，on the，the mat}。

对于参考翻译 2，其 2-gram 序列为{There is，is a，a cat，cat on，on the，the mat}。

对于机器翻译输出，其 2-gram 序列为{The cat，cat is，is on，on the，the mat}。

因此，机器翻译输出中的 2-gram 在参考翻译 1 中的精确匹配率为 5/5＝1，在参考翻译 2 中的精确匹配率为 2/5＝0.4。

需要注意的是，BLEU 通常计算多个 n 的 n-gram 精度。为了综合这些精度，可以使用加权几何平均计算 BLEU 分数的一部分。给定权重 ω_n（通常取对数加权，即 $\omega_n=1/N$，其中 N 为最大 n-gram 长度），可以计算加权几何平均值 $\prod_{n=1}^{N} p_n^{w_n}$。

2. 短句惩罚

为了避免过短的翻译获得高分，BLEU 引入了短句惩罚。如果机器翻译结果的长度（c）小于参考翻译的最佳匹配长度（r），则 BP 计算如下：

$$\mathrm{BP}=\begin{cases}1 & \text{如果 } c>r\\ \exp\left(1-\dfrac{r}{c}\right) & \text{如果 } c\leqslant r\end{cases} \tag{5-25}$$

将加权几何平均值与短句惩罚相乘，得到最终 BLEU 分数，公式如下：

$$\mathrm{BLEU}=\mathrm{BP}\times\prod_{n=1}^{N} p_n^{w_n} \tag{5-26}$$

需要注意的是，尽管 BLEU 是一个广泛使用的评价指标，但它有一些局限性，例如忽略了词序、句法和语义等因素，因此，在实际应用中，通常会结合其他评价指标进行评价。

5.3.6 Seq2Seq 模型小结

Seq2Seq 模型是一种广泛用于自然语言处理和其他序列生成任务的神经网络架构。Seq2Seq 模型的核心思想是将一个序列（如文本）编码成一个固定大小的向量表示，然后解码成另一个序列。这种模型通常包括一个编码器和一个解码器，分别负责将输入序列编码成隐藏表示和将隐藏表示解码成输出序列。

编码器：编码器的目标是将输入序列编码为固定大小的隐藏向量表示。可能的编码器算法有以下几种。

（1）RNN（循环神经网络）：如 LSTM（长短时记忆网络）或 GRU（门控循环单元），这些循环神经网络可以捕捉序列中的长距离依赖关系。

（2）CNN（卷积神经网络）：使用卷积层捕获局部特征，并通过池化层降维以降低计算成本。此外，基于 CNN 的编码器还可以接受图片作为输入，配合基于 RNN 的解码器可以实现“看图说话”的应用。

（3）Transformer：通过自注意力机制和多头注意力在序列中捕捉全局依赖关系，这种架构在许多自然语言处理任务中取得了显著的性能提升。相比于基于 RNN 的架构，效果更好。

解码器：解码器的目标是将隐藏向量表示解码成目标序列。解码器在每个时间步生成一个输出词，然后将这个词作为下一个时间步的输入。可能的解码器算法有以下几种。

(1) RNN：如 LSTM 或 GRU，可以在每个时间步接收来自编码器的隐藏表示及前一个时间步的输出，并生成当前时间步的输出。

(2) CNN：使用卷积层捕捉输出序列中的局部特征，并通过逐步生成序列的方式进行解码。基于 CNN 的解码器也可以生成图片，配合基于 RNN 的编码器可以实现“根据文本描述，生成图片或视频”的应用。

(3) Transformer：利用自注意力机制和多头注意力生成目标序列，还可以使用编码器-解码器注意力层来关注输入序列的不同部分。

Seq2Seq 模型可以应用于机器翻译、文本摘要、问答系统、聊天机器人等任务。为了生成更好的输出序列，Seq2Seq 模型通常与一些搜索策略(如贪婪搜索、Beam Search)结合使用。尽管基于 RNN 的 Seq2Seq 模型取得了很大成功，但它也有一些挑战，如捕捉长距离依赖关系、处理不同长度的输入和输出序列等。为了解决这些问题，研究人员不断地提出新的架构和技术，如注意力机制、Transformer 等。