第8章　注意力机制与外部记忆

> 智慧的艺术是知道该忽视什么.
>
> ——威廉·詹姆斯（William James）
> 美国心理学家和哲学家

根据通用近似定理，前馈网络和循环网络都有很强的能力. 但由于优化算法和计算能力的限制，在实践中很难达到通用近似的能力. 特别是在处理复杂任务时，比如需要处理大量的输入信息或者复杂的计算流程时，目前计算机的计算能力依然是限制神经网络发展的瓶颈.

为了减少计算复杂度，通过部分借鉴生物神经网络的一些机制，我们引入了局部连接、权重共享以及汇聚操作来简化神经网络结构. 虽然这些机制可以有效缓解模型的复杂度和表达能力之间的矛盾，但是我们依然希望在不“过度”增加模型复杂度（主要是模型参数）的情况下来提高模型的表达能力. 以阅读理解任务为例，给定的背景文章（Background Document）一般比较长，如果用循环神经网络来将其转换为向量表示，那么这个编码向量很难反映出背景文章的所有语义. 在比较简单的任务（比如文本分类）中，只需要编码一些对分类有用的信息，因此用一个向量来表示文本语义是可行的. 但是在阅读理解任务中，编码时还不知道可能会接收到什么样的问句. 这些问句可能会涉及背景文章的所有信息点，因此丢失任何信息都可能导致无法正确回答问题.

阅读理解任务是让机器阅读一篇背景文章，然后询问一些相关的问题，来测试机器是否理解了这篇文章.

神经网络中可以存储的信息量称为网络容量（Network Capacity）. 一般来讲，利用一组神经元来存储信息时，其存储容量和神经元的数量以及网络的复杂度成正比. 要存储的信息越多，神经元数量就要越多或者网络要越复杂，进而导致神经网络的参数成倍地增加.

在循环神经网络中，丢失信息的另外一个因素是长程依赖问题.

我们人脑的生物神经网络同样存在网络容量问题，人脑中的工作记忆大概只有几秒钟的时间，类似于循环神经网络中的隐状态. 而人脑每个时刻接收的外界输入信息非常多，包括来自于视觉、听觉、触觉的各种各样的信息. 单就视觉来说，眼睛每秒钟都会发送千万比特的信息给视觉神经系统. 人脑在有限的资源

下，并不能同时处理这些过载的输入信息. 大脑神经系统有两个重要机制可以解决信息过载问题：注意力和记忆机制.

我们可以借鉴人脑解决信息过载的机制，从两方面来提高神经网络处理信息的能力. 一方面是注意力，通过自上而下的信息选择机制来过滤掉大量的无关信息；另一方面是引入额外的外部记忆，优化神经网络的记忆结构来提高神经网络存储信息的容量.

8.1 认知神经学中的注意力

注意力是一种人类不可或缺的复杂认知功能，指人可以在关注一些信息的同时忽略另一些信息的选择能力. 在日常生活中，我们通过视觉、听觉、触觉等方式接收大量的感觉输入. 但是人脑还能在这些外界的信息轰炸中有条不紊地工作，是因为人脑可以有意或无意地从这些大量输入信息中选择小部分的有用信息来重点处理，并忽略其他信息. 这种能力就叫作注意力（Attention）. 注意力可以作用在外部的刺激（听觉、视觉、味觉等），也可以作用在内部的意识（思考、回忆等）.

注意力一般分为两种：

（1）自上而下的有意识的注意力，称为聚焦式注意力（Focus Attention）. 聚焦式注意力是指有预定目的、依赖任务的，主动有意识地聚焦于某一对象的注意力.

聚焦式注意力也常称为选择性注意力（Selective Attention）.

（2）自下而上的无意识的注意力，称为基于显著性的注意力（Saliency-Based Attention）. 基于显著性的注意力是由外界刺激驱动的注意，不需要主动干预，也和任务无关. 如果一个对象的刺激信息不同于其周围信息，一种无意识的“赢者通吃”（Winner-Take-All）或者门控（Gating）机制就可以把注意力转向这个对象. 不管这些注意力是有意还是无意，大部分的人脑活动都需要依赖注意力，比如记忆信息、阅读或思考等.

一个和注意力有关的例子是鸡尾酒会效应. 当一个人在吵闹的鸡尾酒会上和朋友聊天时，尽管周围噪音干扰很多，他还是可以听到朋友的谈话内容，而忽略其他人的声音（聚焦式注意力）. 同时，如果背景声中有重要的词（比如他的名字），他会马上注意到（显著性注意力）.

除非特别声明，在本节及以后章节中，注意力机制通常指自上而下的聚焦式注意力.

聚焦式注意力一般会随着环境、情景或任务的不同而选择不同的信息. 比如当要从人群中寻找某个人时，我们会专注于每个人的脸部；而当要统计人群的人数时，我们只需要专注于每个人的轮廓.

8.2 注意力机制

在计算能力有限的情况下，注意力机制（Attention Mechanism）作为一种资源分配方案，将有限的计算资源用来处理更重要的信息，是解决信息超载问题的主要手段.

注意力机制也可称为注意力模型.

当用神经网络来处理大量的输入信息时，也可以借鉴人脑的注意力机制，只选择一些关键的信息输入进行处理，来提高神经网络的效率.

在目前的神经网络模型中，我们可以将最大汇聚（Max Pooling）、门控（Gating）机制近似地看作自下而上的基于显著性的注意力机制. 除此之外，自上而下的聚焦式注意力也是一种有效的信息选择方式. 以阅读理解任务为例，给定一篇很长的文章，然后就此文章的内容进行提问. 提出的问题只和段落中的一两个句子相关，其余部分都是无关的. 为了减小神经网络的计算负担，只需要把相关的片段挑选出来让后续的神经网络来处理，而不需要把所有文章内容都输入给神经网络.

用 $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in \mathbb{R}^{D\times N}$ 表示 N 组输入信息，其中 D 维向量 $\boldsymbol{x}_n \in \mathbb{R}^D, n \in [1, N]$ 表示一组输入信息. 为了节省计算资源，不需要将所有信息都输入神经网络，只需要从 $\boldsymbol{X}$ 中选择一些和任务相关的信息. 注意力机制的计算可以分为两步：一是在所有输入信息上计算注意力分布，二是根据注意力分布来计算输入信息的加权平均.

注意力分布 为了从 N 个输入向量 $[\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]$ 中选择出和某个特定任务相关的信息，我们需要引入一个和任务相关的表示，称为查询向量（Query Vector），并通过一个打分函数来计算每个输入向量和查询向量之间的相关性.

给定一个和任务相关的查询向量 $\boldsymbol{q}$，我们用注意力变量 $z \in [1, N]$ 来表示被选择信息的索引位置，即 $z = n$ 表示选择了第 n 个输入向量. 为了方便计算，我们采用一种“软性”的信息选择机制. 首先计算在给定 $\boldsymbol{q}$ 和 $\boldsymbol{X}$ 下，选择第 n 个输入向量的概率 α_n，

查询向量 $\boldsymbol{q}$ 可以是动态生成的，也可以是可学习的参数.

$$\begin{aligned}\alpha_n &= p(z = n|\boldsymbol{X}, \boldsymbol{q}) \\ &= \text{softmax}\Big(s(\boldsymbol{x}_n, \boldsymbol{q})\Big) \\ &= \frac{\exp\Big(s(\boldsymbol{x}_n, \boldsymbol{q})\Big)}{\sum_{j=1}^{N}\exp\Big(s(\boldsymbol{x}_j, \boldsymbol{q})\Big)},\end{aligned} \tag{8.1}$$

其中 α_n 称为注意力分布（Attention Distribution），$s(\boldsymbol{x}, \boldsymbol{q})$ 为注意力打分函数，可以使用以下几种方式来计算：

加性模型 $$s(\boldsymbol{x}, \boldsymbol{q}) = \boldsymbol{v}^{\mathrm{T}}\tanh(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{U}\boldsymbol{q}), \tag{8.2}$$

$$\text{点积模型} \qquad s(\boldsymbol{x},\boldsymbol{q}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{q}, \tag{8.3}$$

$$\text{缩放点积模型} \qquad s(\boldsymbol{x},\boldsymbol{q}) = \frac{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{q}}{\sqrt{D}}, \tag{8.4}$$

$$\text{双线性模型} \qquad s(\boldsymbol{x},\boldsymbol{q}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{q}, \tag{8.5}$$

其中 $\boldsymbol{W}, \boldsymbol{U}, \boldsymbol{v}$ 为可学习的参数，D 为输入向量的维度.

理论上，加性模型和点积模型的复杂度差不多，但是点积模型在实现上可以更好地利用矩阵乘积，从而计算效率更高.

当输入向量的维度 D 比较高时，点积模型的值通常有比较大的方差，从而导致 Softmax 函数的梯度会比较小. 因此，缩放点积模型可以较好地解决这个问题. 双线性模型是一种泛化的点积模型. 假设公式(8.5)中 $\boldsymbol{W} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{V}$，双线性模型可以写为 $s(\boldsymbol{x},\boldsymbol{q}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{V}\boldsymbol{q} = (\boldsymbol{U}\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{V}\boldsymbol{q})$，即分别对 $\boldsymbol{x}$ 和 $\boldsymbol{q}$ 进行线性变换后计算点积. 相比点积模型，双线性模型在计算相似度时引入了非对称性.

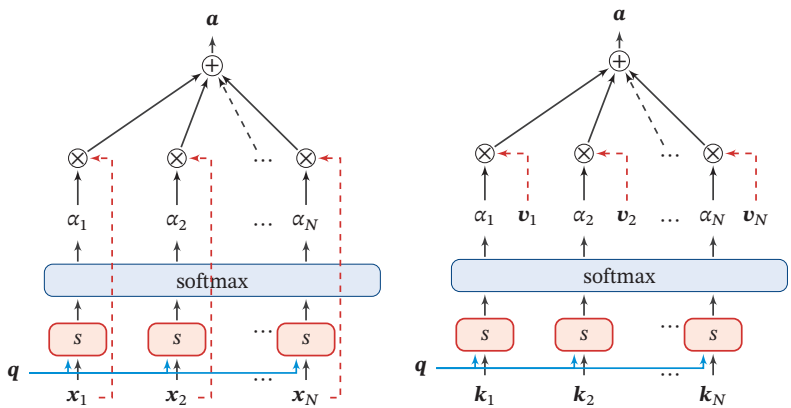
参见习题8-2.

加权平均 注意力分布 α_n 可以解释为在给定任务相关的查询 $\boldsymbol{q}$ 时，第 n 个输入向量受关注的程度. 我们采用一种“软性”的信息选择机制对输入信息进行汇总，即

$$\text{att}(\boldsymbol{X},\boldsymbol{q}) = \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n, \tag{8.6}$$

$$= \mathbb{E}_{z\sim p(z|\boldsymbol{X},\boldsymbol{q})}[\boldsymbol{x}_z]. \tag{8.7}$$

公式(8.7)称为软性注意力机制（Soft Attention Mechanism）. 图8.1a给出软性注意力机制的示例.



(a) 普通模式 (b) 键值对模式

图 8.1 注意力机制

注意力机制可以单独使用，但更多地用作神经网络中的一个组件.

8.2.1 注意力机制的变体

除了上面介绍的基本模式外，注意力机制还存在一些变化的模型.

8.2.1.1 硬性注意力

公式(8.7)提到的注意力是软性注意力，其选择的信息是所有输入向量在注意力分布下的期望. 此外，还有一种注意力是只关注某一个输入向量，叫作硬性注意力（Hard Attention）.

硬性注意力有两种实现方式：

（1）一种是选取最高概率的一个输入向量，即

$$\text{att}(\boldsymbol{X}, \boldsymbol{q}) = \boldsymbol{x}_{\hat{n}}, \tag{8.8}$$

其中 $\hat{n}$ 为概率最大的输入向量的下标，即 $\hat{n} = \arg\max_{n=1}^{N} \alpha_n$.

（2）另一种硬性注意力可以通过在注意力分布式上随机采样的方式实现.

硬性注意力的一个缺点是基于最大采样或随机采样的方式来选择信息，使得最终的损失函数与注意力分布之间的函数关系不可导，无法使用反向传播算法进行训练. 因此，硬性注意力通常需要使用强化学习来进行训练. 为了使用反向传播算法，一般使用软性注意力来代替硬性注意力.

8.2.1.2 键值对注意力

更一般地，我们可以用键值对（key-value pair）格式来表示输入信息，其中“键”用来计算注意力分布 α_n，“值”用来计算聚合信息.

用 $(\boldsymbol{K}, \boldsymbol{V}) = [(\boldsymbol{k}_1, \boldsymbol{v}_1), \cdots, (\boldsymbol{k}_N, \boldsymbol{v}_N)]$ 表示 N 组输入信息，给定任务相关的查询向量 $\boldsymbol{q}$ 时，注意力函数为

$$\text{att}\Big((\boldsymbol{K}, \boldsymbol{V}), \boldsymbol{q}\Big) = \sum_{n=1}^{N} \alpha_n \boldsymbol{v}_n, \tag{8.9}$$

$$= \sum_{n=1}^{N} \frac{\exp\Big(s(\boldsymbol{k}_n, \boldsymbol{q})\Big)}{\sum_j \exp\Big(s(\boldsymbol{k}_j, \boldsymbol{q})\Big)} \boldsymbol{v}_n, \tag{8.10}$$

其中 $s(\boldsymbol{k}_n, \boldsymbol{q})$ 为打分函数.

图8.1b给出键值对注意力机制的示例. 当 $\boldsymbol{K} = \boldsymbol{V}$ 时，键值对模式就等价于普通的注意力机制.

8.2.1.3 多头注意力

多头注意力（Multi-Head Attention）是利用多个查询 $\boldsymbol{Q} = [\boldsymbol{q}_1, \cdots, \boldsymbol{q}_M]$，来并行地从输入信息中选取多组信息. 每个注意力关注输入信息的不同部分.

$$\text{att}\Big((\boldsymbol{K}, \boldsymbol{V}), \boldsymbol{Q}\Big) = \text{att}\Big((\boldsymbol{K}, \boldsymbol{V}), \boldsymbol{q}_1\Big) \oplus \cdots \oplus \text{att}\Big((\boldsymbol{K}, \boldsymbol{V}), \boldsymbol{q}_M\Big), \tag{8.11}$$

其中 $\oplus$ 表示向量拼接.

8.2.1.4 结构化注意力

在之前介绍中，我们假设所有的输入信息是同等重要的，是一种扁平（Flat）结构，注意力分布实际上是在所有输入信息上的多项分布. 但如果输入信息本身具有层次（Hierarchical）结构，比如文本可以分为词、句子、段落、篇章等不同粒度的层次，我们可以使用层次化的注意力来进行更好的信息选择 [Yang et al., 2016]. 此外，还可以假设注意力为上下文相关的二项分布，用一种图模型来构建更复杂的结构化注意力分布 [Kim et al., 2017].

8.2.1.5 指针网络

注意力机制主要是用来做信息筛选，从输入信息中选取相关的信息. 注意力机制可以分为两步：一是计算注意力分布 α，二是根据 α 来计算输入信息的加权平均. 我们可以只利用注意力机制中的第一步，将注意力分布作为一个软性的指针（pointer）来指出相关信息的位置.

指针网络（Pointer Network）[Vinyals et al., 2015] 是一种序列到序列模型，输入是长度为 N 的向量序列 $\boldsymbol{X} = \boldsymbol{x}_1, \cdots, \boldsymbol{x}_N$，输出是长度为 M 的下标序列 $\boldsymbol{c}_{1:M} = c_1, c_2, \cdots, c_M, c_m \in [1, N], \forall m$.

和一般的序列到序列任务不同，这里的输出序列是输入序列的下标（索引）. 比如输入一组乱序的数字，输出为按大小排序的输入数字序列的下标. 比如输入为 $20, 5, 10$，输出为 $1, 3, 2$.

条件概率 $p(c_{1:M}|\boldsymbol{x}_{1:N})$ 可以写为

$$p(c_{1:M}|\boldsymbol{x}_{1:N}) = \prod_{m=1}^{M} p(c_m|c_{1:(m-1)}, \boldsymbol{x}_{1:N}) \tag{8.12}$$

$$\approx \prod_{m=1}^{M} p(c_m|\boldsymbol{x}_{c_1}, \cdots, \boldsymbol{x}_{c_{m-1}}, \boldsymbol{x}_{1:N}), \tag{8.13}$$

其中条件概率 $p(c_m|\boldsymbol{x}_{c_1}, \cdots, \boldsymbol{x}_{c_{(m-1)}}, \boldsymbol{x}_{1:N})$ 可以通过注意力分布来计算. 假设用一个循环神经网络对 $\boldsymbol{x}_{c_1}, \cdots, \boldsymbol{x}_{c_{m-1}}, \boldsymbol{x}_{1:N}$ 进行编码得到向量 $\boldsymbol{h}_m$，则

$$p(c_m|c_{1:(m-1)}, \boldsymbol{x}_{1:N}) = \text{softmax}(s_{m,n}), \tag{8.14}$$

其中 $s_{m,n}$ 为在解码过程的第 m 步时，$\boldsymbol{h}_m$ 对 $\boldsymbol{h}_n$ 的未归一化的注意力分布，即

$$s_{m,n} = \boldsymbol{v}^{\mathrm{T}} \tanh(\boldsymbol{W}\boldsymbol{x}_n + \boldsymbol{U}\boldsymbol{h}_m), \forall n \in [1, N], \tag{8.15}$$

其中 $\boldsymbol{v}, \boldsymbol{W}, \boldsymbol{U}$ 为可学习的参数.

图8.2给出了指针网络的示例，其中 $\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3$ 为输入数字 20, 5, 10 经过循环神经网络的隐状态，$\boldsymbol{h}_0$ 对应一个特殊字符‘<’. 当输入‘>’时，网络一步一步输出三个输入数字从大到小排列的下标.

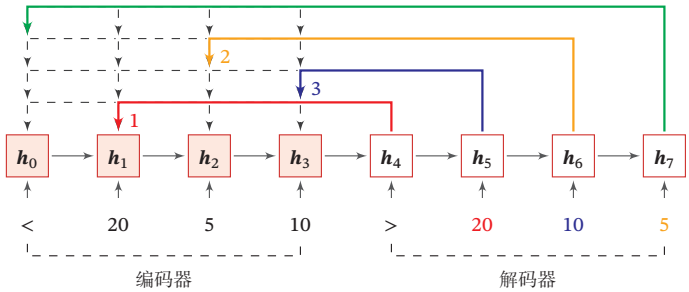
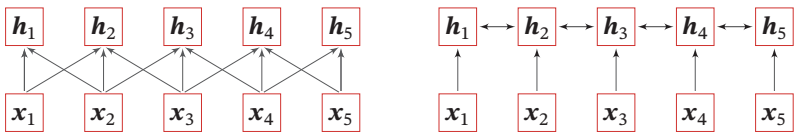


图 8.2 指针网络

8.3 自注意力模型

当使用神经网络来处理一个变长的向量序列时，我们通常可以使用卷积网络或循环网络进行编码来得到一个相同长度的输出向量序列，如图8.3所示.



(a) 卷积网络 (b) 双向循环网络

图 8.3 基于卷积网络和循环网络的变长序列编码

基于卷积或循环网络的序列编码都是一种局部的编码方式，只建模了输入信息的局部依赖关系. 虽然循环网络理论上可以建立长距离依赖关系，但是由于信息传递的容量以及梯度消失问题，实际上也只能建立短距离依赖关系.

如果要建立输入序列之间的长距离依赖关系，可以使用以下两种方法：一种方法是增加网络的层数，通过一个深层网络来获取远距离的信息交互；另一种方法是使用全连接网络. 全连接网络是一种非常直接的建模远距离依赖的模型，但

是无法处理变长的输入序列. 不同的输入长度, 其连接权重的大小也是不同的. 这时我们就可以利用注意力机制来 “动态” 地生成不同连接的权重, 这就是自注意力模型 (Self-Attention Model).

自注意力也称为内部注意力 (Intra-Attention).

为了提高模型能力, 自注意力模型经常采用查询-键-值 (Query-Key-Value, QKV) 模式, 其计算过程如图8.4所示, 其中红色字母表示矩阵的维度.

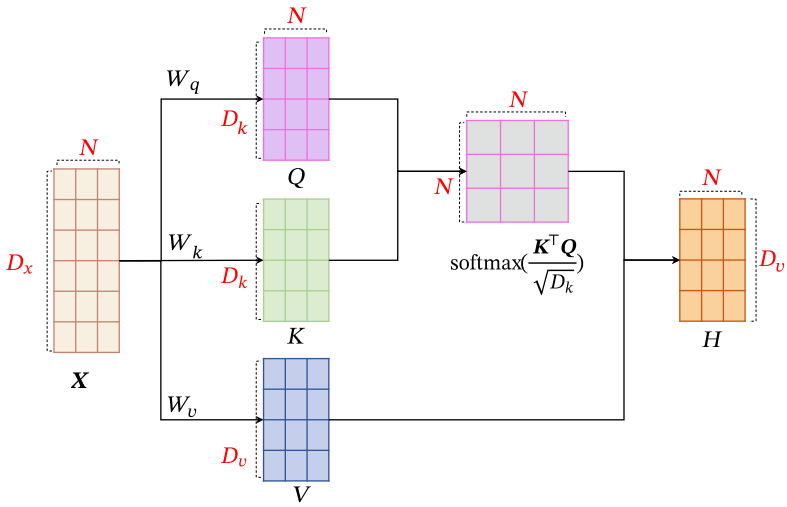


图 8.4 自注意力模型的计算过程

假设输入序列为 $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in \mathbb{R}^{D_x \times N}$, 输出序列为 $\boldsymbol{H} = [\boldsymbol{h}_1, \cdots, \boldsymbol{h}_N] \in \mathbb{R}^{D_v \times N}$, 自注意力模型的具体计算过程如下:

(1) 对于每个输入 $\boldsymbol{x}_i$, 我们首先将其线性映射到三个不同的空间, 得到查询向量 $\boldsymbol{q}_i \in \mathbb{R}^{D_k}$、键向量 $\boldsymbol{k}_i \in \mathbb{R}^{D_k}$ 和值向量 $\boldsymbol{v}_i \in \mathbb{R}^{D_v}$.

由于在自注意力模型中通常使用点积来计算注意力打分, 这里查询向量和键向量的维度是相同的.

对于整个输入序列 $\boldsymbol{X}$, 线性映射过程可以简写为

$$\boldsymbol{Q} = \boldsymbol{W}_q \boldsymbol{X} \in \mathbb{R}^{D_k \times N}, \tag{8.16}$$

$$\boldsymbol{K} = \boldsymbol{W}_k \boldsymbol{X} \in \mathbb{R}^{D_k \times N}, \tag{8.17}$$

$$\boldsymbol{V} = \boldsymbol{W}_v \boldsymbol{X} \in \mathbb{R}^{D_v \times N}, \tag{8.18}$$

其中 $\boldsymbol{W}_q \in \mathbb{R}^{D_k \times D_x}, \boldsymbol{W}_k \in \mathbb{R}^{D_k \times D_x}, \boldsymbol{W}_v \in \mathbb{R}^{D_v \times D_x}$ 分别为线性映射的参数矩阵, $\boldsymbol{Q} = [\boldsymbol{q}_1, \cdots, \boldsymbol{q}_N], \boldsymbol{K} = [\boldsymbol{k}_1, \cdots, \boldsymbol{k}_N], \boldsymbol{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N]$ 分别是由查询向量、键向量和值向量构成的矩阵.

(2) 对于每一个查询向量 $\boldsymbol{q}_n \in \boldsymbol{Q}$, 利用公式(8.9)的键值对注意力机制, 可以得到输出向量 $\boldsymbol{h}_n$,

$$\boldsymbol{h}_n = \text{att}\Big((\boldsymbol{K}, \boldsymbol{V}), \boldsymbol{q}_n\Big) \tag{8.19}$$

$$= \sum_{j=1}^{N} \alpha_{nj} \boldsymbol{v}_j \tag{8.20}$$

$$= \sum_{j=1}^{N} \text{softmax}\big(s(\boldsymbol{k}_j, \boldsymbol{q}_n)\big)\boldsymbol{v}_j, \tag{8.21}$$

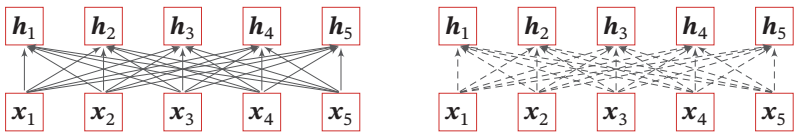
其中 $n, j \in [1, N]$ 为输出和输入向量序列的位置，α_{nj} 表示第 n 个输出关注到第 j 个输入的权重.

如果使用缩放点积来作为注意力打分函数，输出向量序列可以简写为

$$\boldsymbol{H} = \boldsymbol{V}\,\text{softmax}(\frac{\boldsymbol{K}^{\top}\boldsymbol{Q}}{\sqrt{D_k}}), \tag{8.22}$$

其中 $\text{softmax}(\cdot)$ 为按列进行归一化的函数.

图8.5给出全连接模型和自注意力模型的对比，其中实线表示可学习的权重，虚线表示动态生成的权重. 由于自注意力模型的权重是动态生成的，因此可以处理变长的信息序列.



(a) 全连接模型 (b) 自注意力模型

图 8.5 全连接模型和自注意力模型

自注意力模型可以作为神经网络中的一层来使用，既可以用来替换卷积层和循环层 [Vaswani et al., 2017]，也可以和它们一起交替使用（比如 $\boldsymbol{X}$ 可以是卷积层或循环层的输出）. 自注意力模型计算的权重 α_{ij} 只依赖于 $\boldsymbol{q}_i$ 和 $\boldsymbol{k}_j$ 的相关性，而忽略了输入信息的位置信息. 因此在单独使用时，自注意力模型一般需要加入位置编码信息来进行修正 [Vaswani et al., 2017]. 自注意力模型可以扩展为多头自注意力（Multi-Head Self-Attention）模型，在多个不同的投影空间中捕捉不同的交互信息.

参见习题8-3.

多头自注意力参见第15.6.3节.

8.4 人脑中的记忆

在生物神经网络中，记忆是外界信息在人脑中的存储机制. 大脑记忆毫无疑问是通过生物神经网络实现的. 虽然其机理目前还无法解释，但直观上记忆机制和神经网络的连接形态以及神经元的活动相关. 生理学家发现信息是作为