**注意力（attention）机制学习**

1. attention的通用定义及其计算步骤

attention的通用定义：给出一组值向量(values)和一个查询向量(query)，attention是一种根据查询向量(query)计算这组值向量(values)的加权和的方法。换句话说，attention就是一种根据某些规则或者某些额外信息（query）从值向量（values）中抽取特定的向量进行加权组合（attention）的方法。简单来讲，只要我们从部分向量里面搞了加权求和，那就算用了attention。有时候也把这种attention的机制叫做query的输出关注了（或者说考虑到了）原文的不同部分。

attention的重点就是这个值向量（values）中的每个value的“权值”的计算方法。这个“权值”就是对values中的信息的一个有选择性的概要，查询向量(query)决定了values中的哪一部分会被关注，会被着重提取。同时attention也是一种获取一组长度不定的值向量(values)依赖于另一个查询向量(query)的定长表示的方式。

假设我们有一组值向量(values) ，和一个查询向量(query) 。则由attention的定义，我们可以得到其计算步骤：

|  |  |
| --- | --- |
| 1. | 计算注意力分数 (这一步有很多种方法) |
| 2. | 通过softmax获得权重 |
| 3. | 根据得到的权重分布计算加权和从而得到attention输出(Context向量) |

计算注意力分数有以下常用方法：

1. 点乘(Dot Product): ，这种情况需要满足，同时这也是计算量最小的方法。
2. 乘法(Multiplicative)：，是一个可学习的矩阵。
3. 拼接(Concat):，是一个可学习的向量。
4. 加法(Addictive):，，其中维度为一个超参数。
5. Self-attention的计算公式及其分析

在attention的计算中，我们提前知道了查询向量(query)的值，但在很多情况下，我们无法得知查询向量(query)的值，但此时我们又想使用attention，针对这样的场景， Yang et al.在2016年提出了Self-attention。顾名思义，Self-attention的query和values都属于同一个序列。Self-attention的公式如下：

其中均为原始数据进行线性变换得到的，为一超参数。即不妨假设输入为，维度为特征长度，维度为通道数。分别有三个不同的矩阵，那么。在最后，通常还要进行一次映射，进行通道直接的信息融合。

之所以对原始数据进行线性变换，是为了提升模型的拟合能力, 矩阵都是可以训练的，可以起到一个缓冲的效果。下文将分析为何Self-attention的公式为公式一。

首先对于。对其分解成向量来看，即是的每一行乘以的每一列，即两个向量做点乘（内积）。而向量的点乘（内积）所表示的几何意义为：表征两个向量的夹角，表征一个向量在另一个向量上的投影。我们知道，投影的值大，说明两个向量相关度高，两者的夹角越小。当投影的值为0时，说明两个向量正交，即夹角为90°，两者没有相关性。因此从矩阵的角度来看，可以认为最终的结果里面保存了中每个行向量与中列向量进行内积运算的结果。得到的结果就可以作为attention中的步骤一中的注意力分数。但是在Self-attention中，还对其除了，其原因如下：

假设，里的每个元素均独立同分布于，那么 中元素均独立同分布于。当变得很大时， A 中的元素的方差也会变得很大，如果 A 中的元素方差很大，那么softmax(A) 的分布会趋于陡峭(分布的方差大，分布集中在绝对值大的区域)。总结一下就是softmax(A) 的分布会和有关。因此 A 中每一个元素除以后，方差又变为1。这使得softmax(A) 的分布“陡峭”程度与解耦，从而使得训练过程中梯度值保持稳定。

对于，是为了归一化后进行加权求和，在前一步中已经求出了各向量之间的相关性，得到了注意力分数，该步即为attention中的步骤二。

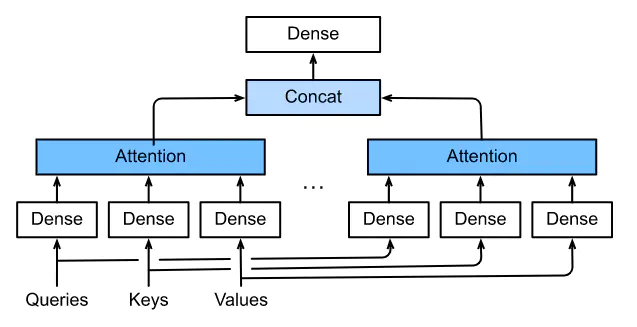
对于，即为加权求和之后的表示。通过已经得到的权重进行加权求和，该步即为attention中的步骤三。

1. MSA（Multi-Head attention）

由于单层的attention涵盖的信息可能不足以支持多种下游任务，且一个头就只有一个学习空间；多个头就有多个学习空间，可能学习的东西会多一点，不会别带跑偏或者说在高维空间下存在学习难度较大的问题。因此产生了多头注意力机制（Multi-Head attention）。同时多头使得建模能力更强，表征空间更丰富。

这种结构设计能让每个注意力机制通过映射到不同的空间中去学习特征，去优化每个词汇的不同特征部分，从而均衡同一种注意力机制可能产生的偏差，让词义拥有来自更多元的表达，实验表明可以从而提升模型效果。

Multi-Head attention的图示如下图所示，其是多个attention输出进行concat并且再次通过一个线性映射得到最后的输出。



如果attention为Self-attention，那么就是多头自注意力机制（Multi-Head Self-attention），即MSA（MHSA）。

在实际中，为了减小计算量，常将原始数据经过变换后在通道上进行分段，将每一段计算一个attention，然后再将多个attention的结果进行concat，最后相乘一个融合矩阵得到结果。

1. Cross-attention

Cross-attention与Self-attention类似，只是其输入来自不同的序列，而Self-attention是同一序列。Cross-attention中一个序列作为输入的，定义了输出的序列长度，另一个序列提供输入的。两个序列可以是不同的模式形态（如：文本、声音、图像），但两个序列必须具有相同的维度。

Cross-attention将两个相同维度的嵌入序列不对称地组合在一起，而其中一个序列用作查询Q输入，而另一个序列用作键K和值V输入。即其公式如下：

其与Self-attention的计算公式一模一样，只是的计算有所不同。其通常是一个序列作为输入的Q，定义了输出的序列长度，另一个序列提供输入的K&V。

1. W-MSA（Windows Multi-head Self-Attention）

W-MSA是Swin Transformer中提出的，其主要目的是为了解决Transformer应用到视觉任务中时其计算量太大的问题。其首先将feature map划分多个window，然后在各个window中计算MSA。而由于这样会导致window与window之间无法进行信息交流，因此又引入了SW-MAS（shifted windows Multi-head Self-Attention）。通过将W-MSA中的window进行一定的平移，这样新的window就可以融合原来多个window的信息，达到信息交流的目的，同时也减少了计算量。

计算量计算（在视觉任务中）

首先对于矩阵相乘的计算量大小计算：假设，那么矩阵乘法：

的计算量为。

Self-attention计算量：

对于feature map中的每个像素（或称作token，patch），都要通过生成对应的，这里假设的向量长度与feature map的深度C保持一致。那么对应所有像素生成Q的过程如下式：

其中为将所有像素（token）拼接在一起得到的矩阵（一共有hw个像素，每个像素的深度为C），为生成query的变换矩阵， 为所有像素通过得到的query拼接后的矩阵。根据矩阵运算的计算量公式可以得到生成的计算量为，同样的生成和的计算量均为，那么总共就是。接下来，对应的计算量为。

忽略除以与的计算量，假设得到的结果为，则对应的计算量为。

那么总共的计算量为。

Multi-Head attention计算量：

对于多头的attention计算量，其首先进行的线性变换是不变的，即数据通过对应的生成对应的，这里同样假设的向量长度与feature map的深度C保持一致。那么对应所有像素生成Q的过程如下式：

根据矩阵运算的计算量公式可以得到生成的计算量为，同样的生成和的计算量均为，那么总共就是。在生成之后，就需要对其在通道反向进行拆分，此处假设拆分为h份，那么就会进行h次attention。下面将计算单次attention的计算量：

由上述计算公式知对应的计算量为。忽略除以与的计算量，假设得到的结果为，则对应的计算量为。

那么单次attention的计算量为。因此h次attention的计算量为。之后还需要进行一个融合，即将h个拼接后得到后相乘一个矩阵进行融合。那么计算量为。

因此综上可得Multi-Head attention的计算量为。

W-MSA计算量：

对于W-MSA模块首先要将feature map划分到一个个窗口（Windows）中，假设每个窗口的宽高都是M，那么总共会得到h/M×w/M个窗口，然后对每个窗口内使用多头注意力模块。在Multi-Head attention中得到高为h，宽为w，深度为C的feature map的计算量为，那么对于高宽均为M的feature map，其计算量为。又由于共有h/M×w/M个窗口，那么总共的计算量为：

八股

计算qkv的时候为什么不加偏置？

1. 注意力机制已经是一个相对复杂的模型，添加偏置项可能会进一步增加模型的参数数量，从而增加模型的复杂性和过拟合的风险。
2. 关于过拟合的风险，其分析如下：如果说，，那么计算得到的注意力就为，那么即使softmax的时候，就可能影响真正注意力的计算，
3. softmax权重分配存在赢者通吃，极其容易饱和，增加bias将加剧现象。

计算attention的时候除了，为什么？

可以从方差或二阶矩来考虑，对于两个d维向量q,k，假设它们都采样自“均值为0、方差为1”的分布，那么它们的内积的二阶矩是：

也就是内积的二阶矩为d，由于均值也为0，所以这也意味着方差也是d。除以后，q与k的内积的方差变为1，这样计算softmax的时候各个数的值不会太大和太小，这样softmax之后也不至于变成one hot而梯度消失了。

对于q,k，假设它们都采样自“均值为0、方差为1”的分布，这个可以pre Norm，其输入的x是服从均值为0、方差为1的正态分布，那么经过一个线性变换后得到Q,K,V那么只需要保证经过线性变换后Q,K,V仍然服从均值为0、方差为1的正态分布即可，那么当不加偏置时，，(设输入节点数为m，输出节点数为n)根据上面的计算公式，可以得到

此时只需要使用初始化策略为“从均值为0、方差为1/m的随机分布中独立重复采样”，这就是Xavier初始化。