Data Mining and Optimization

Lecture 8: Attention

Liu Yang

Nanjing University

Spring, 2025

Table of Contents

什么是attention?

- 我们观察事物时，之所以能够快速判断一种事物，是因为大脑能够很快把注意力放在事物最具有辨识度的部分而做出判断，而并非是从头到尾的观察一遍事物后，才能有判断结果，正是基于这样的理论，产生了注意力机制。
- 一段文字中不同词汇承载语义的“分量”是不同的，这就意味着我们需要在文字中不同词汇上投以不同的注意力，以表示对其重要性和贡献的区分。

Table of Contents

机器翻译

- 在NLP领域，注意力机制首先被用于机器翻译的Seq2Seq模型；
- 机器翻译是利用计算机将一种自然语言（源语言）转换为另一种自然语言（目标语言）的过程；
- 2014年，Yoshua Bengio团队提出了基于编码器(encoder)-解码器(decoder)架构的Seq2Seq模型，二者都被构造为循环神经网络(RNN)结构：输入的自然语言序列被编码器RNN转换为一个中间向量，然后再被解码器RNN构造为输出的目标语言序列。

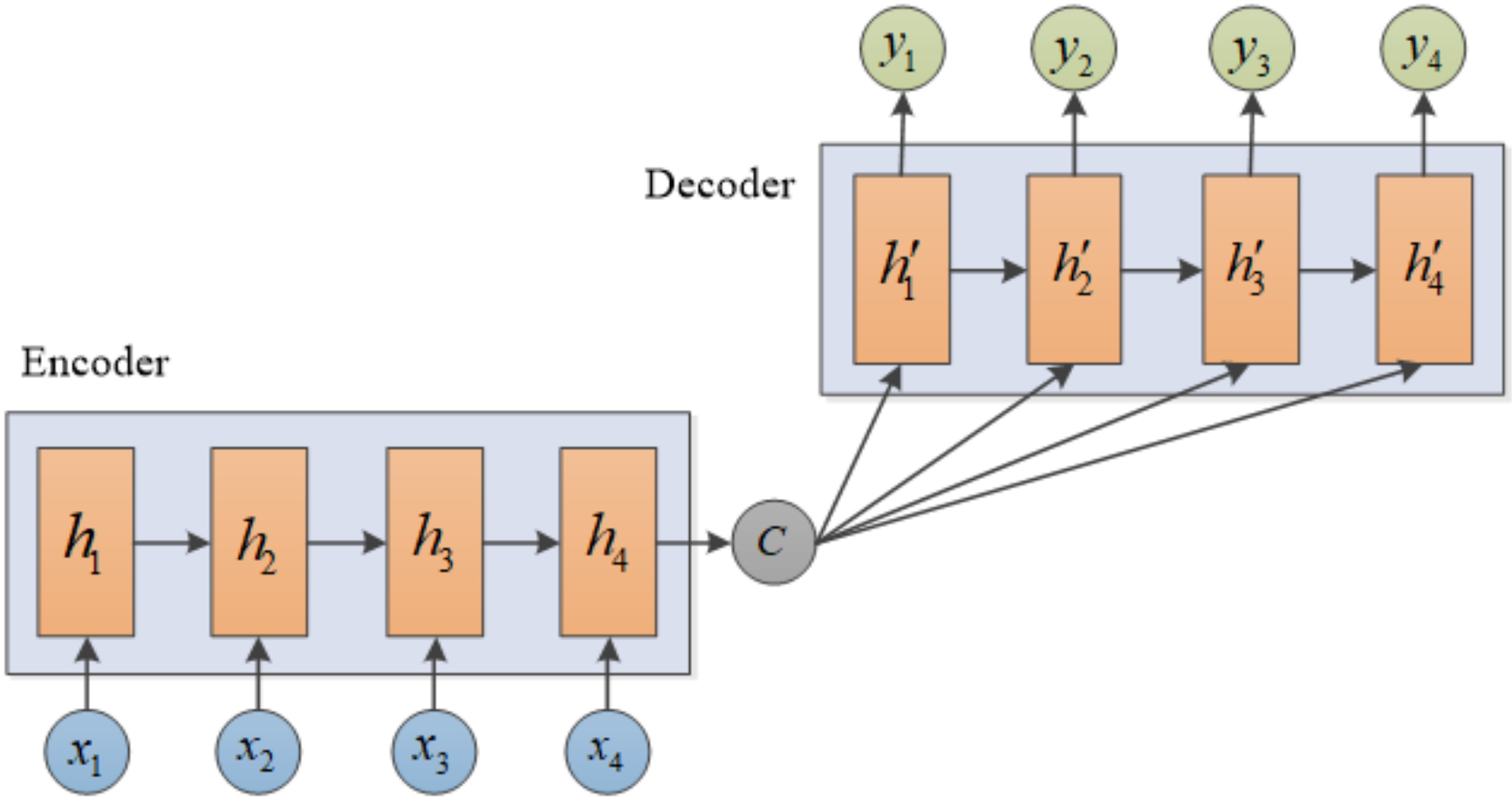
机器翻译

- 对于一个语言序列，不同词汇对于翻译的重要性是不同的，而标准Seq2Seq模型“雨露均沾”的将整个输入编码为一个定长向量，一定会导致重要信息的损失；
- Bengio团队于是基于Seq2Seq架构添加注意力机制，提出了注意力Seq2Seq模型，有效提升了机器翻译的质量。

Seq2Seq模型

- 给定一个长度为T的输入序列$x = \{x_1, \cdots, x_T\}$，Seq2Seq模型通过编码器-解码器架构来产生一个长度为T'的输出序列$y = \{y_1, \cdots, y_{T'}\}$；
- 当x和y为不同语言时，Seq2Seq模型完成真正意义的机器翻译功能；当x和y为同种语言时，y则可以认为是x的另一种表述（如x为一篇中文新闻稿，y为与之对应的中文摘要）。

Seq2Seq模型



RNN结构

- RNN是一种用来处理序列数据的神经网络架构：面对输入序列$x = \{x_1, \cdots, x_T\}$，RNN为其中的第t个元素$x_t (t = 1, \cdots, T)$计算一个对应的隐状态h_t；
- 在计算隐状态时，RNN会综合使用上一步隐状态h_{t-1}和当前步输入x_t得到当前步的隐状态h_t，即

$$h_t = f(h_{t-1}, x_t), \tag{1}$$

其中,f为针对输入x_t和隐状态h_{t-1}所进行的某种变换操作，其具体形式可繁可简。

三种RNN结构

- 标准RNN：h是线性变换加非线性激活函数；
- 长短期记忆模型(Long Short-Term Memory,LSTM)：有三道门控机制；
- 门控循环单元(Gated Recurrent Unit, GRU)：LSTM的门控简化版。

RNN的作用

- RNN在决定当下输入的表示时，不是仅仅“看着眼前”x_t，而是还要“想着过去”h_{t-1}，以此来建模历史对当下的影响；
- 在机器翻译中，历史信息即上文信息，这就意味着翻译模型在考虑当前词汇的表达时，不是仅考虑当前输入的待翻译词汇是什么，而是要看其出现的前文说了什么；
- "I want to read a magazine" v.s. "A gun magazine"。

编码器

- Seq2Seq中的编码器按照先后顺序读取输入序列，最终得到隐状态序列$h_0, h_1, \cdots, h_T$，其中，h_0为计算隐状态h_1时用到的隐状态初值；
- 编码器基于上述隐状态再计算得到一个具有固定长度的向量c，以此作为整个输入序列的“浓缩”表示；
- 因此，编码器可表示为从输入序列到上下文向量的映射，即

$$c = encoder(x_1, \cdots, x_T), \tag{2}$$

- 获得上下文向量c的方式有多种，只要蕴含输入序列的完整信息即可，其中最简单的方式是取RNN编码器的最后一个隐状态，即$c = h_T$；也可以是最后一个隐状态的某种变换，即$c = q(h_T)$，还可以是针对所有隐状态所作的某种变换，即$c = q(h_1, \cdots, h_T)$。

解码器

- Seq2Seq中的解码器也是RNN架构，以编码器输出的上下文向量c作为输入，一边逐个产生自身的隐状态序列$h'_1, \cdots, h'_{T'}$，一边逐个生成目标序列$y = \{y_1, \cdots, y_{T'}\}$；
- 其中，第t步隐状态h'_t由上下文向量c，上一个隐状态h'_{t-1}，以及上一个输出y_{t-1}共同决定，即

$$h'_t = g(h'_{t-1}, y_{t-1}, c), \tag{3}$$

- 对于第t步的输出y_t，其计算方法表示为

$$y_t = \phi(h'_t, y_{t-1}, c), \tag{4}$$

通常，y_t是能够使得在给定输入以及一系列先前翻译输出这两个条件下概率最大的那个词汇，即

$$y_t = \arg\max p(y|y_{t-1}, y_{t-2}, \cdots, y_1, c). \tag{5}$$

Seq2Seq模型的训练

- 给定训练集$\{(x^{(i)}, y^{(i)}) : i = 1, \cdots, N\}$，其中，$x^{(i)}$和$y^{(i)}$分别为成对的输入和输出序列；
- 利用MLE的思想，最优模型参数θ^*能够使条件似然函数$p_\theta(y^{(i)}|x^{(i)})$取得最大，也即对数似然函数$\log p_\theta(y^{(i)}|x^{(i)})$取得最大，即

$$\theta^* = \arg\max \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(y^{(i)}|x^{(i)}), \tag{6}$$

- 对于一个联合分布，可以用链式法则将其展开为一系列条件概率的乘积形式，因此有

$$\begin{aligned} p_\theta(y^{(i)}|x^{(i)}) &= p_\theta(y_1^{(i)}, \cdots, y_{T'}^{(i)}|c^{(i)}) \\ &= p_\theta(y_1^{(i)}|c^{(i)}) p_\theta(y_2^{(i)}|y_1^{(i)}, c^{(i)}) \cdots p_\theta(y_{T'}^{(i)}|y_{T'-1}^{(i)}, \cdots, y_1^{(i)}, c^{(i)}), \end{aligned} \tag{7}$$

- 将(7)代入(6)可得

$$\theta^* = \arg\max \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T'} \log p_\theta(y_t^{(i)}|y_{t-1}^{(i)}, \cdots, y_1^{(i)}, c^{(i)}). \tag{8}$$

Seq2Seq模型的问题

- 上下文向量语意表达能力有限：机器翻译实践表明，Seq2Seq模型的输入序列越长，翻译质量越差，这是由于无论输入序列的长短，编码器都会将其“浓缩”并“抽象”的表达为一个具有固定长度的上下文向量c。当输入序列过长（信息量过多）时，上下文向量在语义表达方面将显得力不从心；
- 输入序列“不划重点，平等对待”：解码器能够看到编码器的唯一输出即上下文向量，这就意味着在生成目标序列每一个元素y_t时使用的上下文向量c都是相同的，也就意味着输入序列每一个元素x_t对输出序列每一个元素y_t的影响是相同的。这是有悖常理的，毕竟在一个输入序列中，不同元素携带的信息量是不同的，在翻译任务中受到关注的程度也应该存在差异。

Table of Contents

注意力Seq2Seq模型

- 针对传统Seq2Seq模型存在的问题，注意力Seq2Seq模型不再要求编码器将所有输入序列的信息都压缩为一个固定长度的上下文向量c，取而代之的是将输入序列映射为多个上下文向量$c_1, \cdots, c_{T'}$，其中，c_t是与输出y_t对应的上下文信息$(t = 1, \cdots, T')$；
- 对于注意力Seq2Seq模型而言，解码器第t个隐状态的计算方法可表示为

$$h'_t = g(h'_{t-1}, y_{t-1}, c_t), \tag{9}$$

其中，c_t为h'_t自身专有的上下文向量，可写为编码器所有隐状态向量的加权和，即

$$c_t = \sum_{i=1}^{T} \alpha_{ti} h_i, \tag{10}$$

其中，α_{ti}即为注意力权重系数（也称为注意力得分）。

注意力得分

- 在编码器中，隐状态h_i蕴含了输入序列第i个元素的信息，因此对编码器隐状态按照不同权重求和表示在生成预测结果y_t时，对输入序列中的各个元素上分配的注意力是不同的——α_{ti}越大，表示第t个输出在第i个输入上分配的注意力越多，即生成第t个输出时受到第i个输入的影响也就越大；
- 由于权重向量必然满足归一化性质，即$\sum_{i=1}^{T} \alpha_{ti} = 1$，因此可使用softmax函数进行变换，即

$$\alpha_{ti} = \frac{exp(e_{ti})}{\sum_{i=1}^{T} exp(e_{ti})}, \tag{11}$$

  其中，e_{ti}为待转换的"logits"；
- 假设模型正在生成第t个输出y_t，则e_{ti}由解码器上一个隐状态h'_{t-1}以及编码器的第i个隐状态h_i共同确定，表示为

$$e_{ti} = a(h'_{t-1}, h_i), \tag{12}$$

  其中，$a(h'_{t-1}, h_i)$也称为“对齐模型”(alignment model)，表示编码器隐状态h_i与解码器隐状态h'_{t-1}之间的匹配程度。

注意力Seq2Seq模型

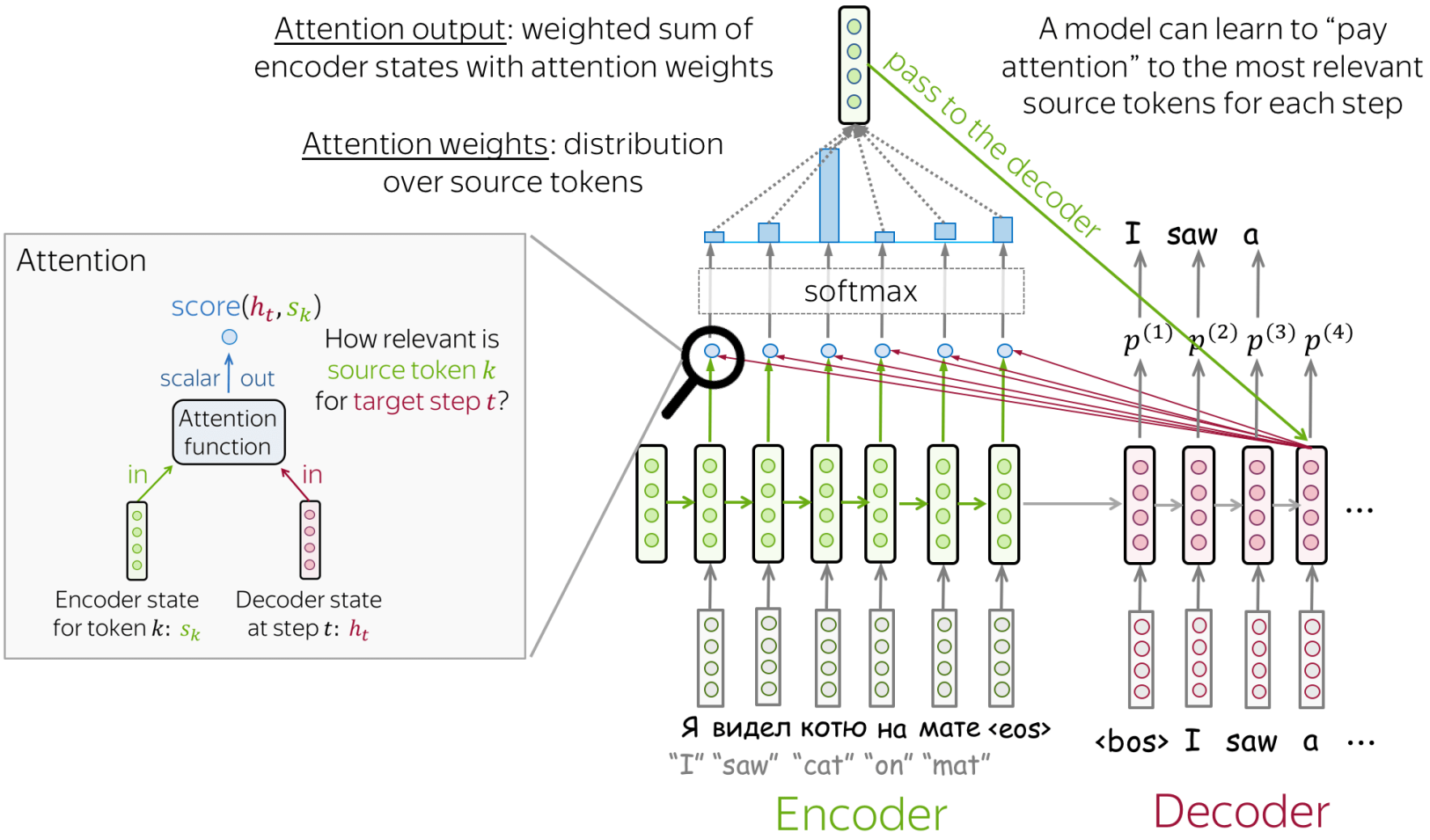


Table of Contents

注意力机制的形式化表示

- 所谓注意力机制，就是针对一个具有多个元素的输入序列，对其计算出一组能够表达重要性程度的值向量——注意力权重，然后以此权重对数据项进行加权求和作为输出序列中的一个元素或该元素的某种表示；
- 注意力机制的形式化表示定义在查询(query)、键(key)和值(value)3个集合之上，其中，注意力权重由查询集合和键集合产生，然后作用在值集合上。

注意力机制的形式化表示

- 记查询集合为$Q = \{q_1, \cdots, q_N\}$，键集合为$K = \{k_1, \cdots, k_M\}$，值集合为$V = \{v_1, \cdots, v_M\}$。三个集合中向量的维度分别为d_q, d_k和d_v（注意键集合和值集合的大小相等，均为M）；
- 注意力机制的工作模式可表示为

$$\begin{aligned} e_{ij} &= a(q_i, k_j) \\ \alpha_{ij} &= \frac{exp(e_{ij})}{\sum_{j=1}^{M} exp(e_{ij})} \\ c_i &= \sum_{j=1}^{M} \alpha_{ij} v_j, \end{aligned} \tag{13}$$

其中，$a(q_i, k_j)$为对齐模型，给出了查询向量q_i和键向量k_j之间的匹配程度；

- 针对对齐模型，其具体实现方式可繁可简，常见形式包括如下两大类

$$\begin{aligned} a(q_i, k_j) &= \beta q_i' W k_j \\ a(q_i, k_j) &= NN(q_i, k_j), \end{aligned} \tag{14}$$

分别称为“乘积型”对齐模型和“网络预测型”对齐模型。

乘积型对齐模型

- 对乘积型对齐模型$a(q_i, k_j) = \beta q_i' W k_j$而言，$W$为一个$d_q \times d_k$可学习的权重矩阵，$\beta$为缩放比例因子；
- 特别地，如果$d_q = d_k = d$，且W为一个d维的单位矩阵，则有$a(q_i, k_j) = \beta q_i' k_j$，即对齐模型被简单定义为两个向量的内积，这种情况也被称为“点积型”对齐模型。Transformer采用的就是这种简单形式（其中$\beta = 1/\sqrt{d}$）。

网络预测型对齐模型

- 对网络预测型对齐模型而言，$NN(q_i, k_j)$表示作用在向量q_i和k_j上的某种神经网络结构，例如

$$a(q_i, k_j) = w_3' tanh(W_1 q_i + W_2 k_j), \tag{15}$$

其中，w_3为参数向量，W_1和W_2分别为前馈神经网络的参数矩阵，$tanh(\cdot)$为双曲正切激活函数。

Table of Contents

注意力机制的类型

- QKV模式：$Q \neq K \neq V$，即三个集合互不同源；
- QVV模式：$Q \neq K = V$，即键集合与值集合同源，但二者与查询集合不同源。例如，在注意力Seq2Seq模型中，查询集合Q由解码器隐状态h'_t构成，而键集合K和值集合V均由编码器隐状态h_t构成；
- VVV模式：$Q = K = V$，三个集合同源，也被称为“自注意力”模式；
- 注意三个集合相等于否不是严格意义上的数值相等，而是强调“同源”这一概念。例如，如果有$Q = \psi_1(X), K = \psi_2(X), V = \psi_3(X)$，尽管三个集合的值不同，但我们认为都是来自相同的源头，故我们认为"$Q = K = V$"。

Table of Contents

Pytorch中的bmm运算

- 当两个矩阵都是三维张量且第一维大小相等时，可利用Pytorch中的bmm运算，是一种特殊的张量乘法运算。

```
mat1 = torch.randn(10, 3, 4)
mat1 = torch.randn(10, 4, 5)
res = torch.bmm(mat1, mat2)
print(res.size())
```