

树注意力机制

作者：张俊东

加法。那么对一个 q 和 n 个 $k-v$ 对，它做了 n 次匹配和 n 次加法。

我在构思一种新的神经网络，

它主要包括这样一些要素：

分割线

(1) 布尔代数

(2) 注意力机制, query, key, value

(3) 树搜索, 二叉树

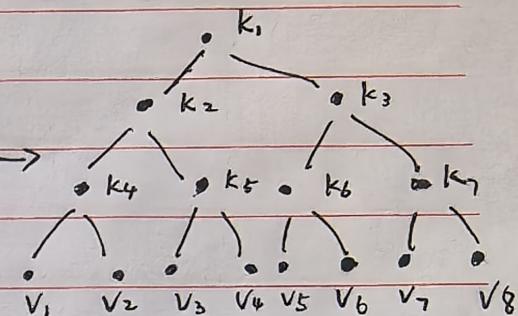
树注意力机制。

针对上面介绍的情景是否可以做一些优化呢？下面是我做出来的一个方案

总体而言，我想做一个可学习的、可解释的、检索高效的树结构。

分割线

经典的注意力机制



$q \rightarrow k_1, k_2, k_3, k_4, \dots, k_n$

让经典的注意力机制会遍历每一个单独的 q ，去找每一个 k_i 去计算一次匹配度，再根据这些匹配度对 v_i 进行加权平均。经典的计算匹配度的方法是做向量的内积，然后 softmax。

可以来数一下做了多少次匹配和 v 的

可以让 key 和 value 组成一个二叉树的结构，来提高检索速度，节省显存。如果这种做法可行，那么它只需要做

$\log(n)$ 次匹配，不需要做加法。

~~那么，主要的技术上的难点点是什么呢？~~

~~检索的规则如下，从根节点出发，如果匹配走右子树，否则走左子树，一直到叶节点。~~

匹配规则是通过学习得到的。

那么这个方案的技术难点有哪些呢？

(1) 向量的内积的匹配规则无

法满足二叉树的要求，这部分

我打算使用布尔运算的规则来

代替

(2) 二叉树的搜索过程不可微

分，因此基于梯度来进行学习未必

有效。我的解决方案是引入“支持度”

的概念，也就是说，在模型做

前向过程时，它做的每一个选择，

会计算它的支持度，最后输出的结果

会有两种情况。

(a) 支持度高，检索出来的 value

也符合预期

(b) 支持度低，不在乎输出的是

什么。

把支持度记为 s ，与预期输出的
匹配度记为 m ，那么这个逻辑应该
是

$$(s \geq m) \vee \neg s \rightarrow 1$$

目前，我写了一些代码来实现这个想法。

下一步会想一下有什么实验来验证一下。

今天就写到这里。2025.1.18