

RankNet To LambdaRank

RankNet是一种基于PairWise方式的排序算法。

在RankNet算法中将模型打分转换为概率，以两个文档 u_i 和 u_j 为例，默认对两个Doc的打分分别为 s_i 和 s_j ，则 u_i 应该排在 u_j 前面的概率可以表示为：

$$P_{ij} = P(u_i \triangleright u_j) = \frac{1}{1 + e^{-\sigma(s_i - s_j)}}$$

由概率公式可以看出，模型对文档 u_i 打分越高，则 $s_i - s_j$ 的值就越大，则分母的值就越小，概率也就越接近于1。反之，分母的值就越大，概率就越小。

两个文档 u_i 和 u_j 真实的排序则由其标注值决定，即标注值大的Doc排在前面，对真实排序的概率值我们可以做如下定义：

$$\bar{P}_{ij} = 1, \text{ 如果 } u_i \text{ 的 } Label \text{ 值大于 } u_j \text{ 的 } Label \text{ 值}$$

$$\bar{P}_{ij} = 0, \text{ 如果 } u_i \text{ 的 } Label \text{ 值小于 } u_j \text{ 的 } Label \text{ 值}$$

$$\bar{P}_{ij} = 0.5, \text{ 如果 } u_i \text{ 的 } Label \text{ 值与 } u_j \text{ 的 } Label \text{ 值相同}$$

则我们可以得到形式化的公式表达为：

$$\bar{P}_{ij} = \frac{1}{2} * (1 + S_{ij})$$

则 S_{ij} 的取值为当 u_i 的Label值大于 u_j 的值时为1，当 u_i 小于 u_j 的值时为-1，当两DocLabel值相同时为0。

根据模型得分得到的概率与真实的概率值，我们可以构建交叉熵，以此作为模型的损失函数：

$$C = -1 * \bar{P}_{ij} \log P_{ij} - (1 - \bar{P}_{ij}) \log (1 - P_{ij})$$

将 \bar{P}_{ij} 和 P_{ij} 分别带入上面的公式，可以得到：

$$C = -[\frac{1}{2} * (1 + S_{ij})] * \log(\frac{1}{1 + e^{-\sigma(s_i - s_j)}}) - [1 - \frac{1}{2} * (1 + S_{ij})] * [1 - \frac{1}{1 + e^{-\sigma(s_i - s_j)}}]$$

下面我们逐步将公式进行化简：

$$C = -\frac{1}{2} (1 + S_{ij}) * [\log 1 - \log(1 + e^{-\sigma(s_i - s_j)})] - [1 - \frac{1}{2} * (1 + S_{ij})] * \log(\frac{e^{-\sigma(s_i - s_j)}}{1 + e^{-\sigma(s_i - s_j)}})$$

$$C = \frac{1}{2} (1 + S_{ij}) * \log(1 + e^{-\sigma(s_i - s_j)}) + \frac{1}{2} (S_{ij} - 1) [\log(e^{-\sigma(s_i - s_j)}) - \log(1 + e^{-\sigma(s_i - s_j)})]$$

$$C = \log(1 + e^{-\sigma(s_i - s_j)}) + \frac{1}{2} (1 - S_{ij}) \sigma(s_i - s_j)$$

当 $S_{ij} = 1$ 时,

$$C = \log(1 + e^{-\sigma(s_i - s_j)})$$

当 $S_{ij} = -1$ 时,

$$C = \log(1 + e^{-\sigma(s_i - s_j)}) + \sigma(s_i - s_j)$$

$$C = \log((1 + e^{-\sigma(s_i - s_j)}) * e^{\sigma(s_i - s_j)})$$

$$C = \log(e^{\sigma(s_i - s_j)} + 1)$$

$$C = \log(e^{-\sigma(s_j - s_i)} + 1)$$

有损失函数的公式可以看出，在两个文档标签不同时，即使模型对两个文档打分相同，损失函数值仍为 $\log 2$ ，这样就会使得模型趋于将标签不同的文档打分区分开。

利用上面得到的损失函数分别计算 s_i 和 s_j 的梯度，可以得到：

$$\frac{\partial C}{\partial s_i} = \sigma \left[\frac{1}{2} (1 - S_{ij}) - \frac{1}{1 + e^{-\sigma(s_i - s_j)}} \right]$$

$$\frac{\partial C}{\partial s_j} = \sigma \left[1 + e^{-\sigma(s_i - s_j)} - \frac{1}{2} (1 - S_{ij}) \right]$$

可以看出

$$\frac{\partial C}{\partial s_i} = - \frac{\partial C}{\partial s_j}$$

对模型中的参数计算梯度，可以得到：

$$\frac{\partial C}{\partial w_k} = \frac{\partial C}{\partial s_i} \frac{\partial s_i}{\partial w_k} + \frac{\partial C}{\partial s_j} \frac{\partial s_j}{\partial w_k}$$

我们将上面的公式代入，则可以得到

$$\frac{\partial C}{\partial w_k} = \frac{\partial C}{\partial s_i} \frac{\partial s_i}{\partial w_k} - \frac{\partial C}{\partial s_i} \frac{\partial s_j}{\partial w_k}$$

进一步提出公共项可以得到

$$\frac{\partial C}{\partial w_k} = \frac{\partial C}{\partial s_i} \left(\frac{\partial s_i}{\partial w_k} - \frac{\partial s_j}{\partial w_k} \right)$$

我们定义

$$\lambda_{ij} = \frac{\partial C}{\partial s_i} = \sigma\left(\frac{1}{2}(1 - s_{ij})\right) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}$$

则可以得到下式:

$$\frac{\partial C}{\partial w_k} = \lambda_{ij} \left(\frac{\partial s_i}{\partial w_k} - \frac{\partial s_j}{\partial w_k} \right)$$

上面的梯度计算都是以单个Pair对进行计算的，但是在实际计算中，我们会以一个Query对该Query下的Pair对聚合在一起作为一个Batch进行训练，因此，在下面展开计算一个Batch下梯度计算的公式。

在构建训练数据时，只选取正序的Pair对，以 u_i 和 u_j 为例，如果两者相等，则我们不将该对Pair加入训练数据，如果 u_i 的Label值大于 u_j 的Label值，则将 (i, j) 作为一个Pair对加入训练数据，如果 u_i 的Label值小于 u_j 的Label值，则将 (j, i) 作为一个Pair对加入训练数据。则一个Query下所有的Pair对记作 I ，则一个Batch下的梯度计算为：

$$\frac{\partial C}{\partial w_k} = \sum_{i,j \in I} (\lambda_{ij} \frac{\partial s_i}{\partial w_k} - \lambda_{ji} \frac{\partial s_j}{\partial w_k}) = \sum_i \lambda_i \frac{\partial s_i}{\partial w_k}$$

其中 λ_i 为涉及到 u_i 的所有Pair对对应的 λ_{ij} 及 λ_{ji} 的加和，所以可以把该值看做是一个箭头或者方向，该值反应的是对于 u_i 需要向哪个方面前进，值的大小则反应的是改动步幅的大小。

上面的公式推导，正是LambdaRank的基础，下面我们就继续解释LambdaRank相比RankNet做了什么改进，以及这些改进的意义是什么。

我们基于上面的RankNet的整个推导过程可以看出，整个RankNet的训练过程就是使得所有的Pair对与真实的Pair对排序情况趋于一致，但是仅仅使得Pair对之间的关系一致与检索系统中的评价指标之间并不能达到一致，我们以论文中举的一个例子为例来说明这个情况，在例子中的评价指标我们使用NDCG。



我们可以看到左侧的逆序对数为13个，右侧的逆序对数为11个，但是如果我们使用NDCG计算指标的话，则左侧会优于右侧，因此，虽然右侧的逆序对数更少，但是从搜索效果还有评估指标看结果都变差了。因此，为了使得训练目标与评估指标之间能够一致，在LambdaRank中考虑针对评估指标进行优化，并且在LambdaRank中比较有趣的一点时，对评估指标的优化并不是通过直接构建损失函数来处理的，因为NDCG是不可导的。

LambdaRank中根据梯度的含义，直接将交互Pair对后产生的NDCG差值加入到 λ_{ij} 中，得到如下公式：

$$\lambda_{ij} = \frac{\partial C}{\partial s_i} = \frac{-\sigma}{1 + e^{\sigma(s_i - s_j)}} |\Delta_{NDCG}|$$

由该公式我们可以进一步反推出LambdaRank的损失函数为：

$$C = \log(1 + e^{-\sigma(s_i - s_j)}) |\Delta_{NDCG_{ij}}|$$

至于作者是如何想到将 Δ_{NDCG} 加入到 λ_{ij} 中作为子项，从论文中提到的只言片语看可能也是作者做了一些尝试，并且实验效果比较好所以做了此项改变。

