**学习排序 Learning to Rank 小结**

原创 2013年05月26日 12:14:09

* 标签：
* [学习排序](http://so.csdn.net/so/search/s.do?q=%E5%AD%A6%E4%B9%A0%E6%8E%92%E5%BA%8F&t=blog) /
* [Learning to Rank](http://so.csdn.net/so/search/s.do?q=Learning%20to%20Rank&t=blog) /
* [机器学习](http://so.csdn.net/so/search/s.do?q=%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0&t=blog)
* 41074

**学习排序（Learning to Rank）**

LTR（Learning torank）学习排序是一种监督学习（SupervisedLearning）的排序方法。LTR已经被广泛应用到文本挖掘的很多领域，比如IR中排序返回的文档，推荐系统中的候选产品、用户排序，机器翻译中排序候选翻译结果等等。IR领域传统的排序方法一般通过构造相关度函数，然后按照相关度进行排序。影响相关度的因素很多，比如上面提到的tf，idf，dl等。有很多经典的模型来完成这一任务，比如VSM，Boolean model，概率模型等。对于传统的排序方法，很难融合多种信息，比如向量空间模型以tf\*idf作为权重构建相关度函数，就很难利用其他信息了，并且如果模型中参数比较多，也会使得调参非常困难，而且很可能会出现过拟合现象。于是人们很自然的想到了用机器学习（Machine Learning）来解决这一问题，于是就有了Learning to rank。机器学习方法很容易融合多种特征，而且有成熟深厚的理论基础，参数是通过迭代优化出来的，有一套成熟理论解决稀疏、过拟合等问题。

学习排序系统框架如图2.1所示：

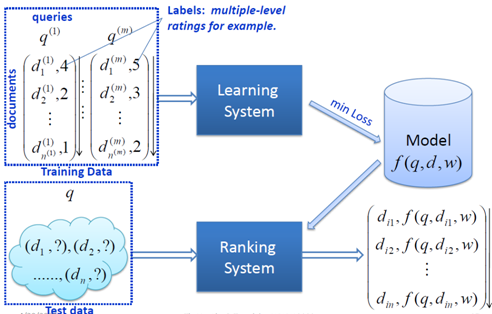


图2.1  排序学习系统框架

对于标注训练集，选定LTR方法，确定损失函数，以最小化损失函数为目标进行优化即可得到排序模型的相关参数，这就是学习过程。预测过程将待预测结果输入学习得到的排序模型中，即可得到结果的相关得分，利用该得分进行排序即可得到待预测结果的最终顺序。

LTR一般说来有三类方法：单文档方法（Pointwise），文档对方法（Pairwise），文档列表方法（Listwise）。

**1  Pointwise**

Pointwise处理对象是单一文档，将文档转化为特征向量后，主要是将排序问题转化为机器学习中常规的分类或回归问题。我们现以多类分类为例进行举例：表2-1是人工标注的部分训练集合，每个文档采用三个特征：查询与文档的BM25相似度，查询与文档的cosin相似度，以及页面的PageRank值，query与di的相关性是多元的，label分为 5个等级，即{perfect，Excellent，good，fair，bad}。于是，产生了5个具有label的训练实例，然后我们可以使用机器学习的任一种多类分类算法进行学习，比如最大熵，支持向量机等。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 文档ID | 查询 | BM25相似度 | Cosin相似度 | PageRank值 | Label |
| 1 | 苹果 库克 | 0.15 | 0.13 | 5 | Bad |
| 2 | 苹果 IPad | 0.32 | 0.24 | 7 | good |
| 3 | 微软 产品 | 0.22 | 0.19 | 2 | good |
| 4 | 谷歌 手机 | 0.55 | 0.53 | 3 | Perfect |
| 5 | 百度 战略 | 0.35 | 0.28 | 1 | excellent |

当模型参数学习完毕后，之后就可利用模型进行相关性判断，对新的查询和文档，通过模型的打分函数可以得到一个数值，利用该数值即可对文档进行排序了。

Pointwise完全从单文档的分类角度计算，没有考虑文档之间的相对顺序。而且它假设相关度是查询无关的，只要（query，di）的相关度相同，那么他们就被划分到同一个级别中，属于同一类。然而实际上，相关度的相对性是和查询相关的，比如一个常见的查询它会有很多相关的文档，该查询和它相关性相对靠后的文档的label标注级别时可能会比一个稀有的查询和它为数不多的高度相关文档的label标准级别更高。这样就导致训练样本的不一致，并且对于预测为同一label级别的文档之间也无法相对排序。

Pointwise常用方法有McRank等。

**2  pairwise**

Pairwise是目前比较流行的方法，相对pointwise他将重点转向文档顺序关系。它主要将排序问题归结为二元分类问题，这时候机器学习的方法就比较多了，比如Boost、SVM、神经网络等。对于同一query的相关文档集中，对任何两个不同label的文档，都可以得到一个训练实例（di,dj），如果di>dj则标记为+1，反之标记为-1，于是我们就得到了二元分类器训练所需的训练样本了，如图2.2所示。测试时，只要对所有pair进行分类就可以得到所有文档的一个偏序关系，从而实现排序。

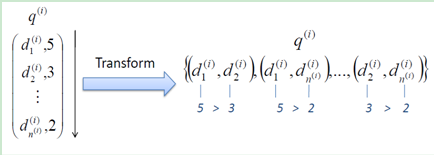


图2.2  Pairwise排序方法示意

尽管Pairwise对Pointwise做了改进，但该方法还是存在明显的问题：

a.只考虑了两篇文档的相对顺序，没有考虑他们出现在搜索结果列表中的位置。排在前面的文档更为重要，如果出现在前面的文档判断错误，惩罚函数要明显高于排在后面判断错误。因此需要引入位置因素，每个文档对根据其在结果列表中的位置具有不同的权重，越排在前面权重越大，如果排错顺序其受到的惩罚也越大。

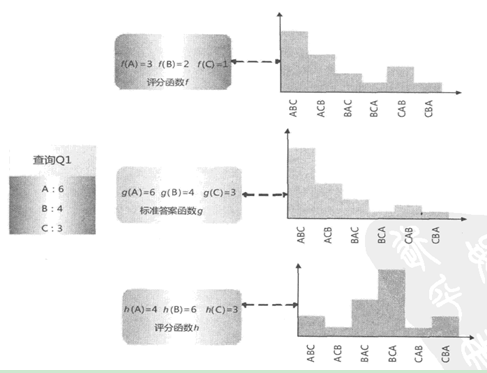
b.对于不同的查询相关文档集的数量差异很大，转换为文档对后，有的查询可能只有十几个文档对，而有的查询可能会有数百个对应的文档对，这对学习系统的效果评价带来了偏置。假设查询1对应500个文档对，查询2对应10个文档对，假设机器学习系统对应查询1能够判断正确480个文档对，对应查询2能够判断正确2个。对于总的文档对该系统准确率是（480+2）/（500+10）=95%，但从查询的角度，两个查询对应的准确率分别为：96%和20%，平均为58%，与总的文档对判断准确率相差巨大，这将使得模型偏向于相关文档集大的查询。

Pairwise有很多的实现，比如Ranking SVM，RankNet，Frank，RankBoost等。

**3  Listwise**

Listwise与上述两种方法不同，它是将每个查询对应的所有搜索结果列表作为一个训练样例。Listwise根据训练样例训练得到最优评分函数F，对应新的查询，评分F对每个文档打分，然后根据得分由高到低排序，即为最终的排序结果。

对应如何训练最优评分函数F，本文介绍一种基于搜索结果排列组合的概率分布情况来训练的方法。如图2-2所示，对应查询Q，假设搜索引擎返回结果A、B、C三个文档，这三篇文档可以产生6中排列方式，对应评分函数F，对三篇文档进行相关度打分，得到F(A)、F(B)、F(C)，根据这三个值可以计算6种排列组合情况各自的概率值。对应不同的评分函数F，六种排列的概率分布是不同的。



假设评分函数g是由人工标记得到的标准答案对应的评分函数，它是怎样的我们暂时不清楚，我们试图找到一个评分函数f，使得f产生的打分和人工的打分尽可能相同。假设存在两个其他评分函数h和f，他们的计算方法已知，对应的搜索排列组合概率分布如图所示，通过KL距离可知，f比h更接近于虚拟的最优函数g。训练过程就是在尽可能的函数中寻找最接近虚拟函数g的那个函数，预测时用该评分函数进行打分。

Listwise方法往往更加直接，它专注于自己的目标和任务，直接对文档排序结果进行优化，因此往往效果也是最好的。Listwise常用方法有AdaRank，SoftRank，LambdaMART等。

**LTR训练数据的获取**

1.人工标注。如果需要大量的训练数据，人工标注不太现实

2.对应搜索引擎来说，可以通过用户点击记录来获取训练数据。对应查询返回的搜索结果，用户会点击其中的某些网页，假设用户优先点击的是和查询更相关的网页。尽管很多时候这种假设并不成立，但实际经验表明这种获取训练数据是可行的。

**LTR特征选取**

使用LTR时会选取一系列文本特征，利用机器学习方法很好的融合到一个排序模型中，来决定最终结果的顺序，其中每一个特征我们称为一个“feature”。对于一个网页文本，feature所在的文档区域可以包括body域，anchor域，title域，url域，whole document域等。

文档的feature又可以分为两种类型：一是文档本身的特征，比如Pagerank值、内容丰富度、spam值、number of slash、url length、inlink number、outlink number、siterank等。二是Query-Doc的特征：文档对应查询的相关度、每个域的tf、idf值，bool model，vsm，bm25，language model相关度等。

综合上述的文档feature的两种类型和位于文档的不同域，我们可以组合出很多feature，当然有些feature是正相关有些是负相关，这需要我们通过学习过程去选取优化。

Pointwise

http://ai.51cto.com/art/201710/555065.htm

轻松入门机器学习排序LTR之：线性模型

上面这篇文章我觉得应该就是Pointwise

Pairwise

Learning to Rank的思想是用机器学习模型解决排序问题。RankSVM是其中Pairwise的方法。

Pairwise方法的直观理解是，对于查询q, 若文档d1比d2更相关（d1>d2）, x1、x2分别是d1、d2的特征，取(x1, x2)为正样本，(x2, x1)为负样本，训练一个二分类器对样本进行分类，就能将排序转化为一个分类问题，从而用机器学习的方法进行排序。RankSVM就是以支持向量机（SVM）为分类模型的Pairwise方法。

那么问题来了，怎么标注文档d1比d2更相关呢？答案是利用点击（Clickthrough）数据。假设查询“support vector machine”，得到如下查询结果，用户最终点击了文档1、3、7。因为用户倾向于点击排在前面的文档，所以若用户点击了文档3而没有点击文档2，且文档2排在文档3前面，就可以认为文档3比文档2更相关。从这个case可以得到3>2, 7>2, 7>4, 7>5, 7>6。

样本示例：

**数据格式:**

 数据的格式与LIBSVM的输入格式比较相似, 第一列代表文档的相关性, 值越大代表越相关, 第二列代表查询号, 后面的代表特征。#后面代表文档注释

3 qid:1 1:1 2:1 3:0 4:0.2 5:0 # 1A  
2 qid:1 1:0 2:0 3:1 4:0.1 5:1 # 1B   
1 qid:1 1:0 2:1 3:0 4:0.4 5:0 # 1C  
1 qid:1 1:0 2:0 3:1 4:0.3 5:0 # 1D    
1 qid:2 1:0 2:0 3:1 4:0.2 5:0 # 2A    
2 qid:2 1:1 2:0 3:1 4:0.4 5:0 # 2B   
1 qid:2 1:0 2:0 3:1 4:0.1 5:0 # 2C   
1 qid:2 1:0 2:0 3:1 4:0.2 5:0 # 2D    
2 qid:3 1:0 2:0 3:1 4:0.1 5:1 # 3A   
3 qid:3 1:1 2:1 3:0 4:0.3 5:0 # 3B   
4 qid:3 1:1 2:0 3:0 4:0.4 5:1 # 3C   
1 qid:3 1:0 2:1 3:1 4:0.5 5:0 # 3D

the following set of pairwise constraints is generated (examples are referred to by the info-string after the # character):

1A>1B, 1A>1C, 1A>1D, 1B>1C, 1B>1D, 2B>2A, 2B>2C, 2B>2D, 3C>3A, 3C>3B, 3C>3D, 3B>3A, 3B>3D, 3A>3D

Ranknet： https://github.com/ysyyork/RankNet

Learning to rank using gradient descent. A simplified implementation of the

algorithm described in http://research.microsoft.com/en-us/um/people/cburges/papers/icml\_ranking.pdf

数据源： LETOR, http://research.microsoft.com/en-us/um/beijing/projects/letor/

Each row is a query-document pair. The first column is relevance label of this pair, the second column is query id, the following columns are features, and the end of the row is comment about the pair, including id of the document. The larger the relevance label, the more relevant the query-document pair. A query-document pair is represented by a 46-dimensional feature vector. Here are several example rows from MQ2007 dataset

'''

3.3 公共数据集

现存一批公开的数据集可以使用

1. LETOR, <http://research.microsoft.com/en-us/um/beijing/projects/letor/>
2. Microsoft Learning to Rank Dataset, <http://research.microsoft.com/en-us/projects/mslr/>
3. Yahoo Learning to Rank Challenge, <http://webscope.sandbox.yahoo.com/>