Learning to rank基本算法小结

原创 felix 浅梦的学习笔记 2019-12-09

收录于话题
#推荐&广告算法技术原理与实践

55个

点击上方蓝字 轻松关注

“本文主要介绍了Learning To Rank(LTR)中的一些基本方法，评价指标和相关算法模型(LambdaMART,RankNet,LambdaRank&FTRL)！”

作者：felix
来源：知乎专栏 有意思的数据挖掘。

我知道你们都是看题图进来的！！！

最近工作中需要调研一下搜索排序相关的方法，这里写一篇总结，总结记录一下几天的调研成果。包括

1. Learning to rank 基本方法
2. Learning to rank 指标介绍
3. LambdaMART 模型原理
4. FTRL 模型原理

Learning to rank

排序学习是推荐、搜索、广告的核心方法。排序结果的好坏很大程度影响用户体验、广告收入等。
排序学习可以理解为机器学习中用户排序的方法，这里首先推荐一本微软亚洲研究院刘铁岩老师关于LTR的著作，Learning to Rank for Information Retrieval，书中对排序学习的各种方法做了很好的阐述和总结。我这里是一个超级精简版。

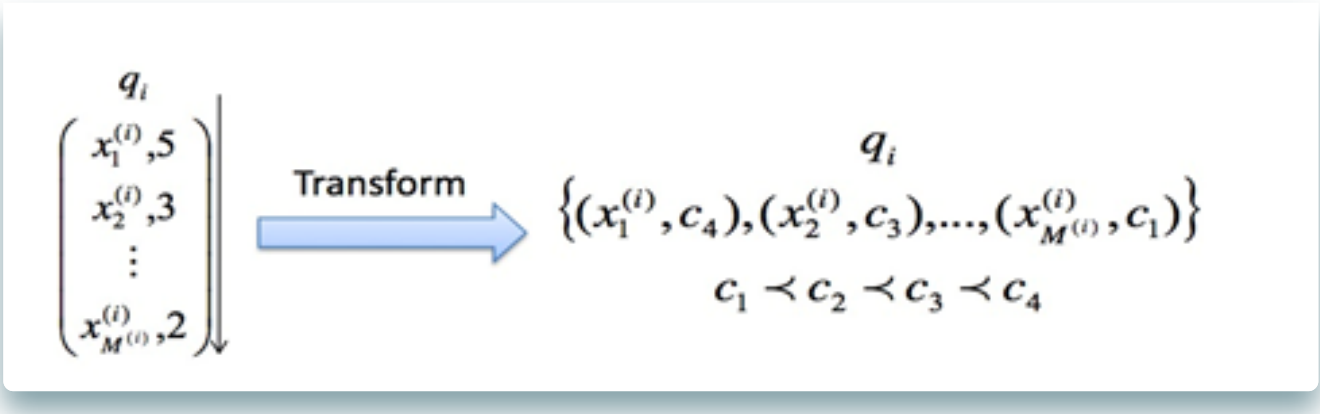
排序学习是一个有监督的机器学习过程，对每一个给定的查询 - 文档对，抽取特征，通过日志挖掘或者人工标注的方法获得真实数据标注。然后通过排序模型，

使得输入能够和实际的数据相似。

常用的排序学习分为三种类型：PointWise，PairWise和ListWise。

PointWise

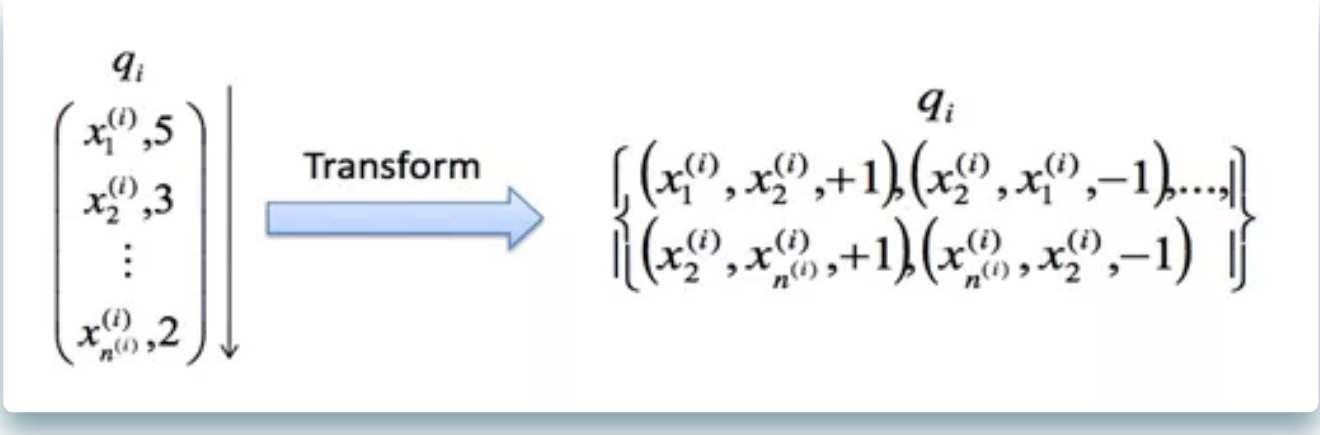
单文档方法的处理对象是单独的一篇文档，将文档转换为特征向量后，机器学习系统根据从训练数据中学习到的分类或者回归函数对文档打分，打分结果即是搜索结果



PointWise方法很好理解，即使用传统的机器学习方法对给定查询下的文档的相关度进行学习，比如CTR就可以采用PointWise的方法学习，但是有时候排序的先后顺序是很重要的，而PointWise方法学习到全局的相关性，并不对先后顺序的优劣做惩罚。

PairWise

对于搜索系统来说，系统接收到用户查询后，返回相关文档列表，所以问题的关键是确定文档之间的先后顺序关系。单文档方法完全从单个文档的分类得分角度计算，没有考虑文档之间的顺序关系。文档对方法将排序问题转化为多个pair的排序问题，比较不同文章的先后顺序。



但是文档对方法也存在如下问题：

1. 文档对方法考虑了两个文档对的相对先后顺序，却没有考虑文档出现在搜索列表中的位置，排在搜索结果前面的文档更为重要，如果靠前的文档出现判断错误，代价明显高于排在后面的文档。

2. 同时不同的查询，其相关文档数量差异很大，所以转换为文档对之后， 有的查询对能有几百个对应的文档对，而有的查询只有十几个对应的文档对，这对机器学习系统的效果评价造成困难

常用PairWise实现：

1. SVM Rank
2. RankNet(2007)
3. RankBoost(2003)

ListWise:

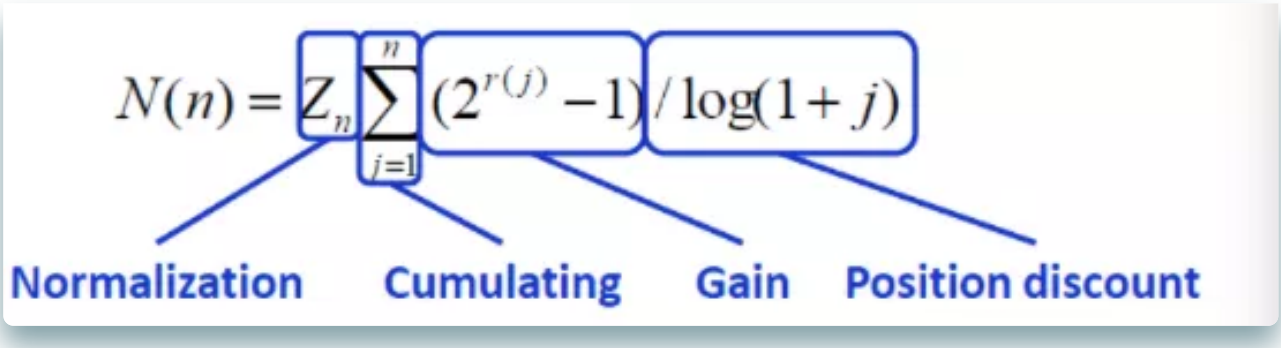
单文档方法将训练集里每一个文档当做一个训练实例，文档对方法将同一个查询的搜索结果里任意两个文档对作为一个训练实例，文档列表方法与上述两种方法都不同，ListWise方法直接考虑整体序列，针对Ranking评价指标进行优化。比如常用的MAP，NDCG。常用的ListWise方法有：

1. LambdaRank
2. AdaRank
3. SoftRank
4. LambdaMART

Learning to rank指标介绍

- MAP(Mean Average Precision):
  假设有两个主题，主题1有4个相关网页，主题2有5个相关网页。某系统对于主题1检索出4个相关网页，其rank分别为1，2，4，7；对于主题2检索出3个相关网页，其rank分别为1,3,5。对于主题1，平均准确率为(1/1+2/2+3/4+4/7)/4=0.83。对于主题2，平均准确率为(1/1+2/3+3/5+0+0)/5=0.45。则MAP= (0.83+0.45)/2=0.64。

- NDCG(Normalized Discounted Cumulative Gain):



NDCG把相关度分为r个等级，如果r = 5，等级设定分别文2^5-1，2^4-1等等

Relevance Rating	Value (Gain)
Perfect	$31=2^5-1$
Excellent	$15=2^4-1$
Good	$7=2^3-1$
Fair	$3=2^2-1$
Bad	$0=2^0-1$

那么加入现在有一个query为abc， 返回如下图所示的列表，假设用户选择和排序结果无关，则累积增益值如右列所示：

	URL	Gain	Cumulative Gain
#1	http://abc.go.com/	31	31
#2	http://www.abcteach.com/	3	34 = 31 +3
#3	http://abcnews.go.com/sections/scitech/	15	49 = 31 + 3 + 15
#4	http://www.abc.net.au/	15	64 = 31 + 3 + 15 + 15
#5	http://abcnews.go.com/	15	79 = 31 + 3 + 15 + 15 + 15
#6	...	...	...

考虑到靠前的位置点击概率越大，那么靠下的位置需要加上衰减因子

log2/log(1+j)，求和就可以得到DCG的值，最后为了使得不同不搜索结果可以比较，用DCG/MaxDCG就可以得到NDCG的值了。

MaxDCG就是当前情况下最佳排序的DCG值。如图所示 **MaxDCG就是1、3、4、5、2的排序情况下的DCG的值(rank 2的gain较低，应该排到后面)**

	URL	Gain	DCG	Max DCG	NDCG
#1	http://abc.go.com/	31	31	31	1 = 31/31
#2	http://www.abcteach.com/	3	32.9	40.5	0.81=32.9/40.5
#3	http://abcnews.go.com/sections/scitech/	15	40.4	48.0	0.84=40.4/48.0
#4	http://www.abc.net.au/	15	46.9	54.5	0.86=46.9/54.5
#5	http://abcnews.go.com/	15	52.7	60.4	0.87=52.7/60.4
#6	...	...	...	...	...

NDCG值

- MRR(Mean Reciprocal Rank)
  给定查询q，q在相关文档的位置是r，那么MRR(q)就是1/R

LambdaMART算法:

LambdaMART是Learning to rank其中的一个算法，在Yahoo! Learning to Rank Challenge比赛中夺冠队伍用的就是这个模型。

LambdaMART模型从名字上可以拆分成Lambda和MART两部分，训练模型采用的是MART也就是GBDT，lambda是MART求解使用的梯度，其物理含义是一个待排序文档下一次迭代应该排序的方向。

但Lambda最初并不是诞生于LambdaMART，而是在LambdaRank模型中被提出，而LambdaRank模型又是在RankNet模型的基础上改进而来。所以了解LambdaRank需要从RankNet开始说起。

论文:
From RankNet to LambdaRank to LambdaMART: AnOverview(https://www.researchgate.net/publication/228936665_From_ranknet_to_lambdarank_to_lambdamart_An_overview)

RankNet

RankNet是一个pairwise模型，上文介绍在pairwise模型中，将排序问题转化为多个pair的排序问题，比较文档di排在文档dj之前的概率。如下图所示

$$P_{ij} \equiv P(U_i \rhd U_j) \equiv \frac{1}{1+e^{-\sigma(s_i-s_j)}}$$

最终的输出的sigmoid函数，RankNet采用神经网络模型优化损失函数，故称为RankNet。

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

可是这样有什么问题呢？排序指标如NDCG、MAP和MRR都不是平滑的函数，RankNet的方法采用优化损失函数来间接优化排序指标。

LambdaRank

如图所示，蓝色表示相关的文档，灰色表示不相关的文档。RankNet以pairwise计算cost左边为13，右图将第一个文档下调3个位置，将第二个文档下调5个位置，cost就降为11。如此以来，虽然RankNet的损失函数得到优化，但是NDCG和ERR等指标却下降了。
RankNet优化的方向是黑色箭头，而我们想要的其实是红色的箭头。
LambdaRank就是基于这个，其中lambda表示红色箭头。

LambdaRank不是通过显示定义损失函数再求梯度的方式对排序问题进行求解，而是分析排序问题需要的梯度的物理意义，直接定义梯度，Lambda梯度由两部分相乘得到：(1)RankNet中交叉熵概率损失函数的梯度；(2)交换Ui，Uj位置后IR评价指标Z的差值。具体如下：

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

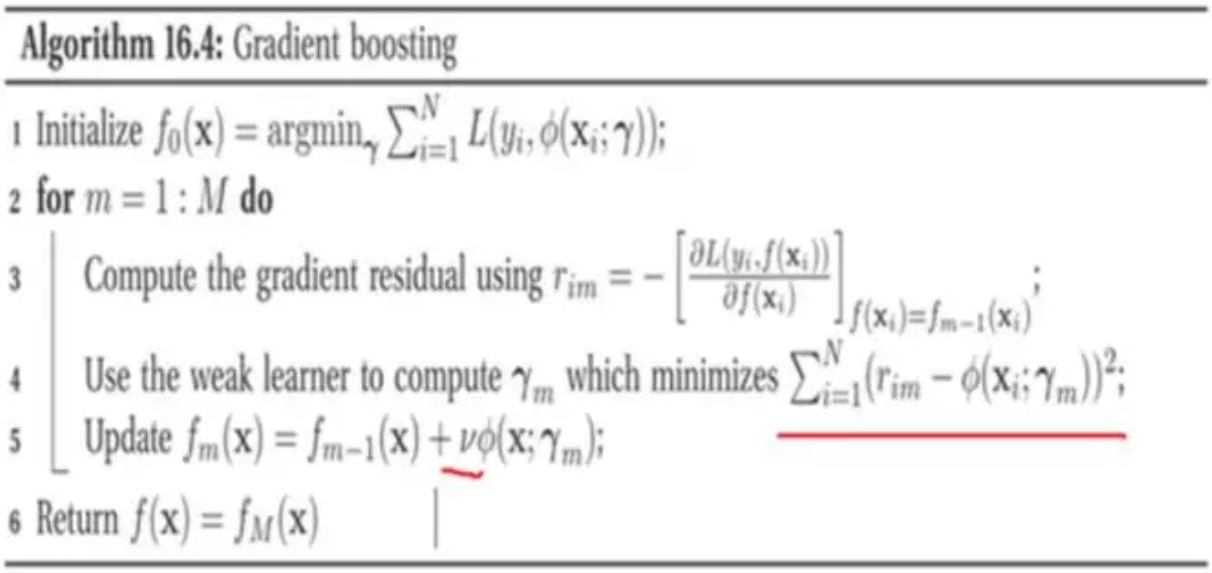
lambdaMART

我们知道GBDT算法每一次迭代中， 需要学习上一次结果和真实结果的残差。在lambdaMART中，每次迭代用的并不是残差，lambda在这里充当替代残差的计算方法。

- LambdaMART算法流程

Algorithm: LambdaMART
set number of trees N, number of training samples m, number of leaves per tree L, learning rate η
for $i = 0$ to m **do**
  $F_0(x_i) = \text{BaseModel}(x_i)$ //If BaseModel is empty, set $F_0(x_i) = 0$
end for
for $k = 1$ to N **do**
  for $i = 0$ to m **do**
    $y_i = \lambda_i$
    $w_i = \frac{\partial y_i}{\partial F_{k-1}(x_i)}$
  end for
  $\{R_{lk}\}_{l=1}^{L}$ // Create L leaf tree on $\{x_i, y_i\}_{i=1}^{m}$
  $\gamma_{lk} = \frac{\sum_{x_i \in R_{lk}} y_i}{\sum_{x_i \in R_{lk}} w_i}$ // Assign leaf values based on Newton step.
  $F_k(x_i) = F_{k-1}(x_i) + \eta \sum_l \gamma_{lk} I(x_i \in R_{lk})$ // Take step with learning rate η.
end for

- GBDT算法流程



Algorithm 16.4: Gradient boosting

1 Initialize $f_0(\mathbf{x}) = \text{argmin}_\gamma \sum_{i=1}^{N} L(y_i, \phi(\mathbf{x}_i; \gamma))$;
2 **for** $m = 1 : M$ **do**
3 Compute the gradient residual using $r_{im} = -\left[\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\right]_{f(\mathbf{x}_i)=f_{m-1}(\mathbf{x}_i)}$;
4 Use the weak learner to compute γ_m which minimizes $\sum_{i=1}^{N}(r_{im} - \phi(\mathbf{x}_i; \gamma_m))^2$;
5 Update $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \nu\phi(\mathbf{x}; \gamma_m)$;
6 Return $f(\mathbf{x}) = f_M(\mathbf{x})$

对比lambdaMART和GBDT算法流程，主要框架是相同的，只不过LambdaMART模型用lambda梯度代替了GBDT的残差。

FTRL算法（Follow the regularized Leader Proximal)

论文：Ad click prediction: a view from the trenches(https://www.researchgate.net/publication/262412214_Ad_click_prediction_a_view_from_the_trenches)

点击率预估问题（CTR）是搜索、广告和推荐中一个非常重要的模块。在CTR计算的过程中，常常采用LR模型。

FTRL属于在线算法，和SGD等常用的在线优化方法对比，可以产生较好的稀疏性，非常适合ID特征较多，维度较高的特征。

google的论文中已经给出了很详细的工程化实现的说明，该方法也已经广泛的应用。

- 参数优化

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left(\mathbf{g}_{1:t} \cdot \mathbf{w} + \frac{1}{2} \sum_{s=1}^{t} \sigma_s \|\mathbf{w} - \mathbf{w}_s\|_2^2 + \lambda_1 \|\mathbf{w}\|_1 \right),$$

第一项：保证参数不偏移历史参数
第二项：保证w不会变化太大
第三项：代表L1正则，获得稀疏解

- 算法流程：

Algorithm 1 Per-Coordinate FTRL-Proximal with L_1 and L_2 Regularization for Logistic Regression

#With per-coordinate learning rates of Eq. (2).
Input: parameters α, β, λ_1, λ_2
$(\forall i \in \{1, \dots, d\})$, initialize $z_i = 0$ and $n_i = 0$
for $t = 1$ **to** T **do**
  Receive feature vector $\mathbf{x}_t$ and let $I = \{i \mid x_i \neq 0\}$
  For $i \in I$ compute

$$w_{t,i} = \begin{cases} 0 & \text{if } |z_i| \leq \lambda_1 \\ -\left(\frac{\beta+\sqrt{n_i}}{\alpha} + \lambda_2\right)^{-1}(z_i - \text{sgn}(z_i)\lambda_1) & \text{otherwise.} \end{cases}$$

  Predict $p_t = \sigma(\mathbf{x}_t \cdot \mathbf{w})$ using the $w_{t,i}$ computed above
  Observe label $y_t \in \{0, 1\}$
  for all $i \in I$ **do**
    $g_i = (p_t - y_t)x_i$ *#gradient of loss w.r.t. w_i*
    $\sigma_i = \frac{1}{\alpha}\left(\sqrt{n_i + g_i^2} - \sqrt{n_i}\right)$ *#equals $\frac{1}{\eta_{t,i}} - \frac{1}{\eta_{t-1,i}}$*
    $z_i \leftarrow z_i + g_i - \sigma_i w_{t,i}$
    $n_i \leftarrow n_i + g_i^2$
  end for
end for

相关推荐

- **用户画像必会的行为偏好计算方法**
- **【CTR预估】FLEN: 一种时空高效的利用特征场信息缓解梯度耦合的CTR预测模型**
- **【CTR预估】CTR模型如何加入稠密连续型和序列型特征?**
- **【CTR预估】你真的需要 pairwise LTR吗？速览搜索推荐中pointwise和pairwise方法**
- **视频点击预测大赛 Top5 DeepCTR baseline**