第十节课 Learning to Rank与Airbnb个性化推荐

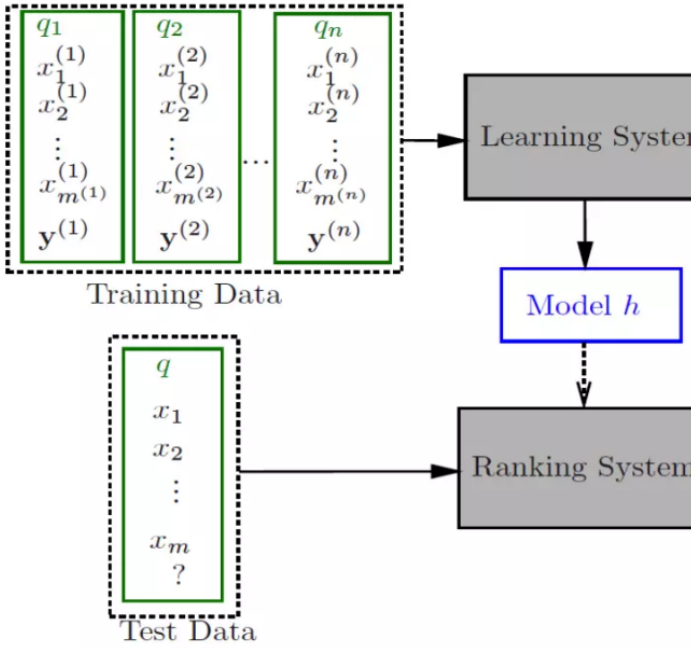
https://github.com/cystanford 老师的GitHub

第十节课笔记	
讲义内容概括	知识点

LTR learn to rank 排序学习
推荐系统，搜索，广告的核心算法之一
排序学习场景:
推荐系统，基于历史行为的“猜你喜欢”
搜索排序，基于某Query进行的结果排序，期望用户选中的在排序结果中是靠前的 =>

LTR学习策略:
Learning to rank for information retrieval
http://exp.newsmth.net/attachment/0dd952f21955fdfede030d8f5e9a9360
推荐不仅是预估问题，更应该关注排序问题
传统的排序算法（PageRank等）考虑的因素少，需要组合多种因素进行排序
使用机器学习，目标是最小化损失函数，根据每次迭代的信息反馈，自动优化模型参数
parameter)
有监督的排序学习方法，通过训练数据 => Ranking Model

所描述的步骤为:训练数据获取->特征提取->模型训练->测试数据预测->效果评估



特征提取

检索系统会使用一系列特征来表示一次查询，通过模型之后最终决定文档的排序顺序，询的文档,occur-term共现的词，则提取的特征主要有以下三大类:

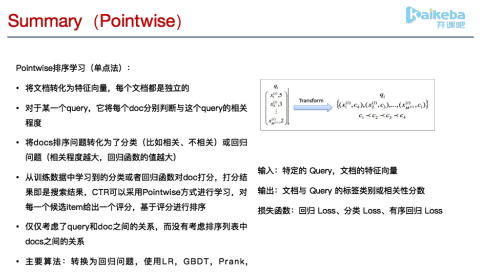
- occur-term特征
  - 共现在查询中的出现次数、比率等
- occur-term的特征
  - 共现在文档中的出现次数、比率等
  - 共现词与文档的相关性特征:BM25系列
- 自身特征
  - PageRank值
  - Spam信息
  - Quality质量分
  - 行为分,ctr，停留时间，二跳率等..

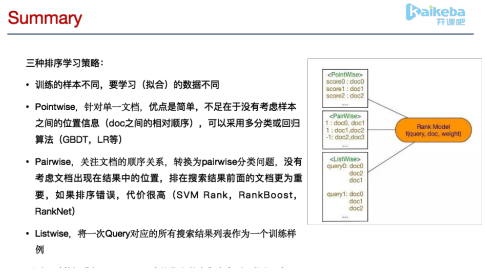
Learning to Rank

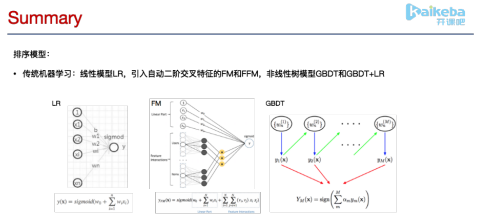
- 什么是Learning to Rank
- LTR学习的三种策略 Pointwise，Pairwise，Listwise
- 评价指标：MAP, NDCG, MRR
- 排序算法：RankNet, LambdaRank, LambdaMART
- 工具：ranklib
- 使用ranklib对MQ2008进行排序

Airbnb个性化推荐

- Airbnb个性化推荐场景
- List Embedding
- Word2Vec的使用
- List Embedding的评估
- List Embedding的冷启动
- 基于List Embedding的相似房源推荐
- User Type Embedding与Listing Type Embedding
- 基于Embedding的实时个性化搜索







LTR学习的三种策略：

Pointwise，针对单一文档

Pairwise，关注文档的顺序关系

Listwise，将一次Query对应的所有搜索结果列表作为一个训练样例

LTR	优点	缺点
Pointwise	每个文档为单独训练数据，算法简单	从单文档分类角之间的相对顺序
Pairwise	只需要对所有文档对进行分类，得到文档集的偏序关系	只考虑每对文际上，文档对并而且不同的查询同，结果会向偏移
Listwise	将一个查询对应的所有搜索结果列表作为一个训练样本	很难找到合适的化目标，很难找行求解

Pointwise排序学习（单点法）：

- 构造训练样本，即得（V1,V2,V3,..., Y），V是特征，Y是label
- 特征选择：
  - item（doc）特征，评分、销量、价格等
  - user特征，性别、年龄、偏好等
  - user-item特征，是否看过此item、是否买过此item、对item的价 场景特征地理位置等
- label构造： 比如展示过的item label设置为1，用户点击过的label为3，用户购买
- 将训练样本转换为多分类问题（样本特征-类别标记）或者回归问题（样本特征-
  - 如果转换为多分类问题，模型最后的输出只能1、3或是7 => 在同一类中的
  - 如果转换为回归问题，常采用LR、GBDT等来解决
  - 优点是简单，不足在于没有考虑样本之间的位置信息（doc之间的相对顺序
- LR用于建模point-wise方式的数据，这种模型直接对标注好的point-wise数据，
- GBDT+LR进行预测
- Summary
  - 将文档转化为特征向量，每个文档都是独立的对于某一个query，它将每个相关程度
  - 将docs排序问题转化为了分类（比如相关、不相关）或回归问题（相关程
  - 从训练数据中学习到的分类或者回归函数对doc打分，打分结果即是搜索方式进行学习，对每一个候选item给出一个评分，基于评分进行排序
  - 仅仅考虑了query和doc之间的关系，而没有考虑排序列表中docs之间的关
  - 主要算法：转换为回归问题，使用LR，GBDT，Prank, McRank

Pairwise排序学习（配对法）：

- LTR学习方法之一，将排序问题转换为二元分类问题 接收到用户查询后，返回相先后顺序关系（多个pair的排序问题）
- 对于同一查询的相关文档集中，对任何两个不同label的文档，都可以得到一个训赋值+1，反之-1 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出现判断错误，代价会很高
- 主要算法：SVM Rank，RankBoost(2003)，RankNet(2007)

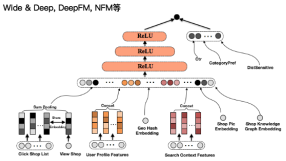
Listwise排序学习（列表法）：

- 它是将每个Query对应的所有搜索结果列表作为一个训练样例
- 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，即为最终的排序结果
- 直接考虑整体序列，针对Ranking评价指标（比如MAP, NDCG）进行优化
- 主要算法：ListNet, AdaRank，SoftRank，LambdaRank, LambdaMART等
- LambdaMART是对RankNet和LambdaRank的改进，在 Yahoo Learning to Ran
- 评测指标：
  - MAP：Mean Average Precision，平均准确率 对于每个真实相关的文档，置 P，统计该位置之前的文档集合的分类准确率，取所有这些准确率的平

Summary

排序模型：

- 基于DNN模型，Wide & Deep, DeepFM, NFM等



Summary

排序模型思维维度：

- 业务场景：推荐 or 搜索排序
- 是否关注Query内的相对顺序 => PointWise or ListWise
- 构造排序学习的数据样本
- 选择适合的模型（传统机器学习，DNN模型）

Summary

- 2018 KDD best paper
- 房源嵌入（Listing Embeddings）适合于短期实时个性化排序和推荐场景
- 用户类型嵌入（User-type Embedding）和房源类型嵌入(Listing-type Embedding)适合于长期个性化场景
- 训练上，使用了8亿点击Session，为450万房源训练embedding
- 去掉Session中没有价值的部分，比如页面停留时间少于30秒的点击（无效数据），Session中包含的点击行为少于1个（无效数据）
- 在训练模型的几个月里，训练集每天都会更新，新增当天的搜索数据，删除当前最早一天的数据

Summary

- Listing Embeddings粒度最细，将用户的租房点击信息（只有浏览超过30秒才算点击信息）分为多个session
- 在更粗的粒度上（User Type Embedding 和 Listing Type Embedding）上进行长期兴趣学习
- 考虑双边市场的特点，通过加入"booking"、"host rejection"强反馈信号来指导无监督学习
- embedding学习过程将搜索session中数据看作类似序列信息，并通过类似word2vec的方式（进行了改进）学习出每个房源id的embedding值

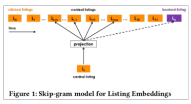


Figure 1: Skip-gram model for Listing Embeddings

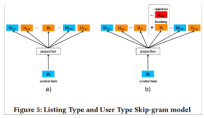


Figure 5: Listing Type and User Type Skip-gram model

Summary

Summary

- CG（cumulative gain）：累计增益，只考虑到了相关性，没有考虑到位
- DCG（Discounted CG）折损累计增益，在每一个CG的结果上除以一个
  能影响最后的结果，排序越往后价值越低。 $DCG_p = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)}$
- IDCG（ideal DCG）理想情况下最大的DCG值 $IDCG_p = \sum_{i=1}^{|REL|} \frac{rel_i}{\log_2(i+1)}$
- NDCG（Normalized DCG）归一化贴现累积收益，综合考虑模型排序结
  使用的排序指标 DCG是一个累加的值，没法针对两个不同的搜索结果进行 $NDCG_p = \frac{DCG_p}{IDCG_p}$
- MRR：平均倒数排名，Mean Reciprocal Rank 把相关文档在结果中的排
  平均 优点是计算简单，不足在于仅仅考虑了相关性最强的文档所在的位

排序算法：

From RankNet to LambdaRank to LambdaMART: An Overview

https://www.researchgate.net/publication/228936665_From_ranknet_to_lambdarank

三种算法：**RankNet，LambdaRank和LambdaMART**，在Yahoo！Learning To Rank
好的排序效果

在RankNet的基础上加入了lambda梯度 => LambdaRank lambda梯度和MART（GBD
结合 => LambdaMart

RankNet—for pairwise

是基于神经网络的排序学习方法，RankNet证明了如果知道一个待排序文档的排列中相
则通过推导可以算出每两个文档之间的排序概率。因此对于一个待排序文档序列，只
率，不需要计算所有pair，减少计算量。

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

神经网络中
输入是x
输出是s
利用logloss的最小化求得w矩阵

通过神经网络或GBDT、DNN等设定一个loss function（这里用logloss效果很好）：

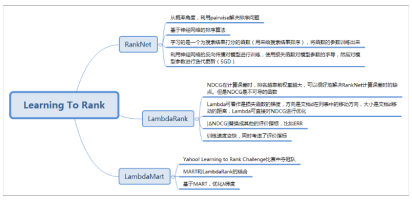
$$C = -\overline{P_{ij}} \log P_{ij} - \left(1-\overline{P_{ij}}\right)\log\left(1-P_{ij}\right)$$

其中 $\overline{P_{ij}} = \frac{1}{2}\left(1+S_{ij}\right)$，$S_{ij} = \begin{cases} 1 & Ui比Uj更相关 \\ 0 & Ui和Uj相关性相等 \\ -1 & Uj比Ui更相关 \end{cases}$ 所以

$$C = \begin{cases} \log\left(1+e^{-\sigma\left(s_i - s_j\right)}\right) & S_{ij} = 1 \\ \log\left(1+e^{-\sigma\left(s_j - s_i\right)}\right) & S_{ij} = -1 \end{cases}$$

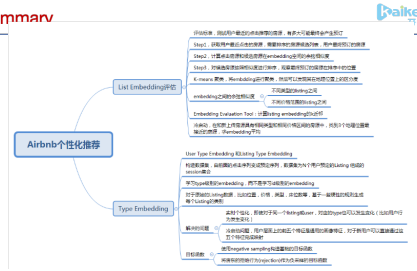
图像如下：

Summary

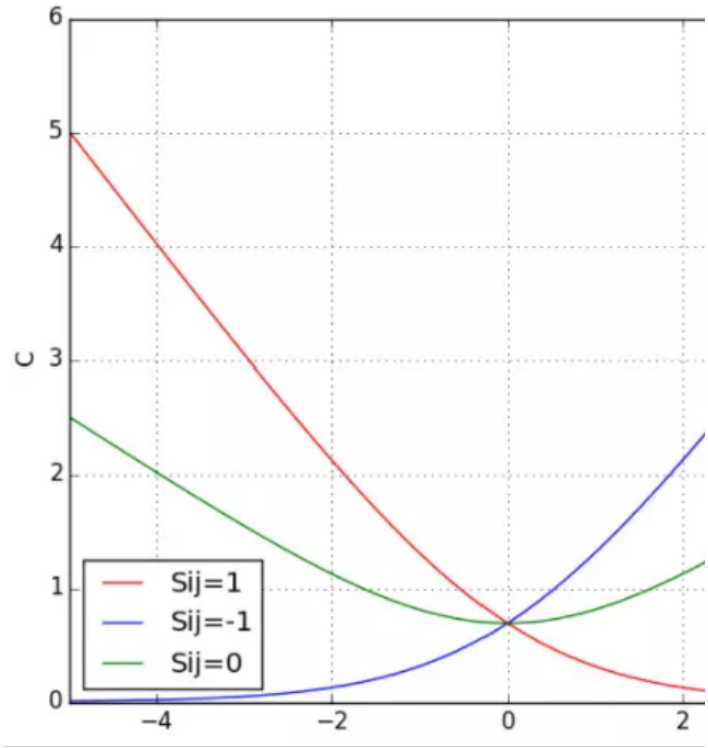


Summary

Summary



Summary



利用神经网络的反向传播对模型进行训练，使用损失函数对模型参数的求导，然后对模

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

通过调整参数的取值来逼近损失函数的最小值

- 采用SGD进行优化，对每一个pair对都会进行一次权重的更新
- 采用mini-batch learning的方式，是对同一个query下的所有doc进行一次权重的 O(n) => 加速RankNet的学习过程
- RankNet的训练数据为各查询下若干文档对，因此需针对文档对的输入去调整网

Algorithm RankNet Training.

Initialize $F_0(\mathbf{x}) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$
for each query $q \in Q$ **do**
  for each pair of URLs U_i, U_j with different labe
    $s_i = f(\mathbf{x}_i)$, $s_j = f(\mathbf{x}_j)$
    Estimate cost C
    Update model scores $w_k \rightarrow w_k - \eta \frac{\partial C}{\partial w_k}$
  end for
end for
Return **w**

人工标注

人工标注的数据主要有以下几大类型:

- 单点标注
  - 对于每个查询文档打上绝对标签
  - 二元标注：相关 vs 不相关
  - 五级标注：完美(Perfect),出色(Excellent),好(Good),一般(Fair),差(Bad)， 一般后面两档属于不相关

    好处：标注的量少O(n)

    坏处：难标。。。不好统一
- 两两标注
  - 对于一个查询Query,要标注文档d1比文档d2是否更加相关 (q,d1)>(q,d2)?

    好处：标注起来比较方便

LambdaRank算法：

RankNet为什么更倾向于对靠后位置的相关文档的排序位置的提升。但实际上，我们升

坏处：标注量大 估计得有O(n^2)

- 列表标注
  - 对于一个查询Query，将人工理想的排序整个儿标好

    好处： 相对于上面两种，标的效果会很好

    坏处： 这个工作量也太大了...-_-||

总结

在玩搜索引擎时敲定特征分的权重是非常疼的一件事儿，而LTR正好帮你定分，LTR的三种实现方法其实各有优劣：

1. 难点主要在训练样本的构建(人工或者挖日志)，另外就是训练复杂
2. 虽说Listwise效果最好，但是天下武功唯快不破，看看这篇文章 http://www.cnblogs.com/zjgtan/p/3652689.html体验下
3. 在工业界用的比较多的应该还是Pairwise，因为他构建训练样本相对方便，并且复杂度也还可以，所以Rank SVM就很火啊-

LambdaRank是一个经验算法，它不是通过显式定义损失函数再求梯度的方式对排序问题需要的分数s的梯度的物理含义，直接定义梯度，即Lambda梯度

LambdaRank在此基础上考虑评价指标Z（比如NDCG,ERR）的变化

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

这里|△Z| = |△NDCG|= |第一种方法下的NDCG|- |第二种方法下的NDCG|

LambdaMart算法：
MART（Gradient Boosting Decision Tree）
用stacking的树集合（类似GDBT），通过优化梯度

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

Algorithm: LambdaMART N棵树，m个样本

set number of trees N, number of training samples m, number
learning rate η

for $i = 0$ to m **do**
  $F_0(x_i) = \text{BaseModel}(x_i)$ //If BaseModel is empty, set $F_0(x$
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 lea
end for

计算梯度lambda和weight

根据训练样本训练处一棵

计算叶子节点output，乘以学习率，模型更

论文 Greedy function Approximation – A Gradient Boosting
Machinehttps://projecteuclid.org/download/pdf_1/euclid.aos/1013203451
算法在每一轮迭代中，先计算出当前模型在所有样本上的负梯度，然后以该值为目标训并计算出该弱分类器的权重，最终实现对模型的更新。

Ranklib：

Airbnb:
稀疏性
secsion:聚类
word2vec: 通过点击的方式完成一个sentence，每一个Word是一个list，即list embedd

List Embedding的目标函数构造：--短期
使用skip-gram构造基础的目标函数

$$\mathcal{L} = \sum_{s \in \mathcal{S}} \sum_{l_i \in s} \left(\sum_{-m \geq j \leq m, i \neq 0} \log \mathbb{P}(l_{i+j} \right.$$

采用negative sampling构造采样之后的目标函数

$$\underset{\theta}{\operatorname{argmax}} \sum_{(l,c)\in\mathcal{D}_p} \log \frac{1}{1+e^{-\mathbf{v}'_c\mathbf{v}_l}} + \sum_{(l,c)\in\mathcal{D}_}$$

全集负采样，同城负采样，booking是正采样

所有的房子离线评估，并embedding

新房源冷启动：
上传特征，附近房源

Embedding evaluation tool—证明embedding的用处

Listing Embedding不能够解决的问题：—user type（基于一类人、房子）

- 需要基于当前的点击来计算
- 只提取了用户的short-term兴趣
- 只能用于“相似房源”场景中
- 只针对同地区下的用户兴趣房源的挖掘

基于Embedding的搜索排序：
Step1，准备3种embedding值，Listing Embedding，User Type Embedding和Listing
Step2，获取基础指标H*
Step3，提取地区embedding
Step4，计算Embedding Features
Step5，在原有的Ranking Model中加入embedding features进行计算

总结Summary

- 记录作业内容
  - Thinking1：排序模型按照样本生成方法和损失函数的不同，可以划分成Pointwise, Pairwise, Listwise三类方法，这三类排序方法有何
  - Thinking2：排序模型按照结构划分，可以分为线性排序模型、树模型、深度学习模型，这些模型都有哪些典型的代表?
  - Thinking3：NDCG如何计算
  - Thinking4：搜索排序和推荐系统的相同和不同之处有哪些
  - Thinking5：Listwise排序模型能否应用到推荐系统中
  - Action1：数据集：porto seguro safe driver prediction（kaggle 2017年比赛）https://www.kaggle.com/c/porto-seguro-safe-driver-p 车与住房保险公司之一，根据汽车保单持有人的数据建立机器学习模型，分析该持有人是否会在次年提出索赔。数据已进行脱敏 使用 Cross Entropy
  - Action2：熟悉RankLib的使用，针对MQ2008数据集中的Fold1，使用RankNet, ListNet, LambdaMart三种模型进行排序学习，并对比 https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/
  - Action3：如果你是某P2P租车的技术负责人，你会如何设计个性化推荐和搜索排序阐述相似车型，搜索排序的设计方法 可能的embe
-