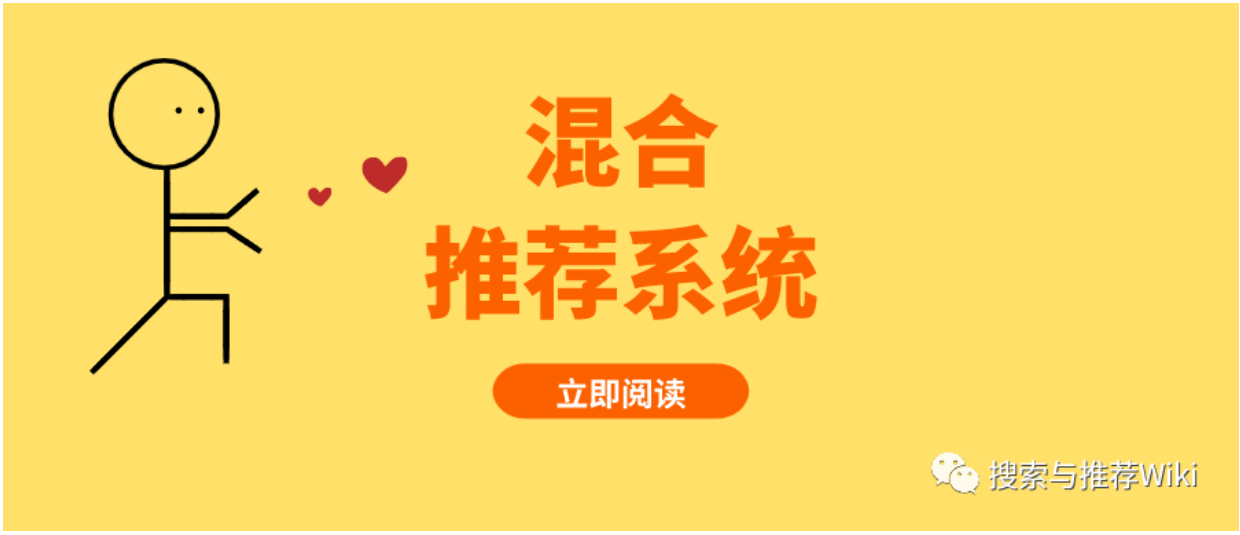
晓得嘛？混合推荐系统速览和技巧锦囊

原创 Thinkgamer 搜索与推荐Wiki 2020-05-19

收录于话题
#推荐相关笔记

32个



> 本文为《推荐系统与深度学习》第五章的复习笔记，只记录了一些要点，希望能够快速的进行复习，如果发现哪一个点不明白的话，可以自行展开学习或者加小编微信进行技术交流。

5.1 什么是混合推荐系统

混合推荐系统的含义

海量数据推荐系统中通常存在三部分：

- 在线系统（Online）
  - 直接与用户进行交互，具有高性能、高可用的特性，通常利用缓存系统，处理热门请求的重复计算
- 近在线系统（Nearline）

    - 接受在线系统的请求，执行比较复杂的推荐算法，缓存在线系统的结果，并及时收集用户的反馈，快速调整结果
- 离线系统（Offline）
    - 利用海量用户的行为日志进行挖掘，进行高质量的推荐，通常运算周期长，资源消耗大
- 经典的Netflix推荐架构：离线、近在线、 在线

混合推荐系统的算法分类

从混合技术看，Robin Burke 在其不同混合推荐算法设计方案的调研报告中将其分为：

- 加权型
- 切换型
- 交叉型
- 特征组合型
- 瀑布型
- 特征递增型
- 元层次型

5.2 推荐系统特征的处理方法

数据和特征决定了模型的上限，而模型算法则为逼近这个上限

特征处理方法

数值特征处理

- 无量纲处理
    - z-score规范化$x' = \frac{x-\mu}{\sigma}$
    - max-min标准化$x' = \frac{x-min}{max-min}$
    - 标准化是将值整合到 0～1，规范化的结果可能会在 0，1之外（**两者实验效果差别不大**）
- 非线性变换
    - 对特征进行非线性变换，增加模型复杂度
    - 常见的变换手段有
        - 基于多项式

    - 基于指数函数
    - 基于对数函数
  - **一般对数变换后特征分布更平稳，对数变换能够很好的解决随着自变量的增加，因变量的方差增大的问题**
  - **将非线性的数据通过对数变换，转换为线性数据，便于使用线性模型进行学习**
- 离散化
  - 离散化的好处
    - 离散化后的特征对异常数据有**很强的鲁棒性**（比如年龄 > 30岁 为1，否则为0，如果没有经过离散化，异常数据，年龄为100岁，就会带来很大的干扰）
    - 特征离散化后，可以**进行交叉，特征内积乘法运输速度快，进一步引入非线性，提升表达能力**，计算结果方便存储，容易扩展
    - 特征离散化后，**模型更加稳定**（比如20-30岁为一个区间，这样不会因为用户年龄增长了一岁就完全变成一个不同的人）
  - 离散化的方式
    - 无监督离散化
      - 等宽度离散化：对异常值较敏感，倾向于将特征不均匀的分到各个箱中，会破坏特征的决策能力。需要指定区间个数
      - 等频度离散化：会避免等宽离散化的坏处，会把相同标签的特征划分到不同的箱中，同样会造成特征决策能力下降。需要指定区间个数
      - 基于聚类分析的离散化方法
    - 有监督离散化
      - 基于熵的离散化方法
      - 基于卡方的离散化方法
      - MDLP方法（最小描述距离长度法则）

离散化处理

- One-Hot编码
  - 稀疏表示，减小存储，同时在一定程度上起到扩充特征的作用
- 特征哈希
  - 目标是把原始的高维特征向量压缩成较低维特征向量，且尽量不损失原始特征的表达能力，是一种快速且很节省空间的特征向量化方法
- 时间特征处理
  - 通常方案是按照业务逻辑以及业务目的进行相关业务在处理
  - Christ.M 提出了一种层次化处理时间特征的方案

- Step 1: 对时间窗口统计特征进行聚合
  - 最大值、最小值、均值、分位数
- Step 2: 利用标签相关性进行特征选择

- Python tsfresh工具可以进行特征提取

特征选择方法

单变量特征选择

对每一个特征进行测试，衡量该特征和响应变量之间的关系。

优点：易于运行，易于理解，通常对于理解数据有较好的结果，但其与设计的算法模型无关。

常见的方法：

1、皮尔逊相关系数

皮尔逊相关系数表示两个变量之间的协方差和标准差的商

$$\rho_{X,Y} = \frac{E[(X-\mu x)(Y-\mu y)]}{\sigma x \sigma y}$$

2、距离相关系数

基于距离协方差进行变量间相关性度量

$$Dcor(X,Y) = \frac{dCov(X,Y)}{((dCov^2(X,X))^{1/2}(dCov^2(Y,Y))^{1/2})^{1/2}}$$

3、卡方检验

思想：通过观察实际值与理论值的偏差来确定理论的正确与否

具体做法：假设两个变量独立，然后观察实际值与理论值的偏差程度，如果偏差足够小，则认为是正常误差，如果偏差大到一定程度，则认为两者相关。

基于模型的特征选择

- 方法1、基于逻辑回归和正则化的特征选择
  - 逻辑回归中越是重要的特征在模型中对应的系数就会越大，而跟输出变量越是无关的特征对应的系数就会越接近于0
  - $L1$正则化将系数w的$L1$范数作为惩罚项加到损失函数上，由于正则项非0，迫使那些弱的特征所对应的系数变成0，因此$L1$正则化往往会使学到的模型很稀疏（系数w经常为

0），这个特性使得$L1$正则化成为一种很好的特征选择方法。

- $L1$正则化像非正则化线性模型一样也是不稳定的，如果特征集合中具有相关联的特征，当数据发生细微变化时也有可能导致很大的模型差异。
- $L2$正则化将稀疏向量的$L2$范数添加到了损失函数中。$L2$正则化会让系数的取值变得平均。对于相关联的特征，意味着他们能够获得更相近的对应系数。
- $L2$正则化对于特征选择来说是一种稳定的模型，$L2$正则化对于特征理解来说更加有用，表示能力强的特征对应的系数是非零。
- $L2$防止模型过拟合。

- 方法2、随机森林特征选择
  - mean decrease impurity（平均不纯度减少，对于分类问题通常采用基尼不纯度或者信息增益。对于回归问题，通常采用方差或者最小二乘拟合。）
  - mean decrease accuracy（平均精确度减少）
  - 准确率高、鲁棒性好、易于使用等优点
  - 随机森林提供了两种特征选择方法：
- 方法3、XGBoost特征选择
  - 某个特征的重要性（feature score）等于它被选中为树节点分裂特征的次数的和
- 方法4、基于深度学习的特征选择
  - 深度学习具有很强的自动特征抽取能力

5.3 常见的预测模型

基于逻辑回归的模型

- 模型满足二项式分布
- 通常使用最大似然函数求解（最大似然函数求负对数）

基于支持向量机的模型

- 支持向量机：Support Vector Algorithm，SVM
- 支持向量机模型把训练样本映射到高维空间中，以使不同类别的样本能够清晰的被超平面分割出来，而后，新样本同样被映射到高维空间，基于其落在哪一面，决定其类别
- SVM是非概率的线性模型

基于梯度提升树的模型

熵介绍

信息量

事件发生概率带来的惊喜度，$I = -log_2 p_i$，只有小概率的事件才有信息量

信息熵

信息量的期望，$H(x) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i)$

条件熵：

$H(x|y) = -\sum_{i=1}^{n} p(x_i|y) log_2 p(x_i|y)$

信息增益

分类前，数据中可能出现多种类别数据，比较混乱，不确定性强，熵比较高，分类后，不同类的数据得到了比较好的划分，比较有序，不确定性降低，熵比较低，信息增益就是用于这种熵的变化。

信息增益：定义：特征A对训练数据集D的信息增益$g(D, A)$，定义为集合D的经验熵$H(D)$与特征A在条件下D的经验条件熵$H(D|A)$之差。$g(D, A) = H(D) - H(D|A)$

信息增益：D根据特征A分为n份$D_1, D_2, \ldots, D_n$，那么$H(D|A)$就是所有$H(D_i)$的期望（平均值），$H(D|A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i)$

信息增益比

单纯的信息增益只是一个相对值，因为其依赖于$H(D)$的大小，而信息增益比更能客观地反映信息增益。

信息增益比定义：特征A对训练数据集D的信息增益比$g_R(D, A)$，定义为其信息增益$g(D, A)$与分裂信息熵$split_info(A)$之比：

$$g_R(D, A) = \frac{g(D, A)}{split\_info(A)}$$

其中$split_info(A) = H(A) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} log_2(\frac{|D_j|}{|D|})$

基尼

基尼定义：

$$\sum_{i=1}^{c} f_i(1-f_i)$$

f_i是某个分区内第i个标签的频率，C是该分区中的类别总数量。基尼（Gini）不纯度度量是类型被分错的可能性（类型的概率乘以分错的概率）

方差

方差不纯度定义：

$$\frac{1}{N}\sum_{i=1}^{N}(y_i-\mu)^2$$

y_i是某个实例的标签，N是实例的总数，μ是所有实例的均值。

决策树

- 决策树学习的本质是从训练数据集上归纳出一组分类规则，通常采用启发式的方法，即局部最优
- 决策树学习的三个步骤：特征选择、决策树的生成、决策树的剪枝
- 选择最佳分裂特征的标准式找出局部最优的特征，常用的方法有：熵、基尼、方差，前两个针对分类，后者针对回归
- 三种常见的决策树算法
  - ID3：使用信息增益寻找最佳分裂特征
  - C4.5：使用信息增益比寻找最佳分裂特征
  - CART：全称是Classification And Regression Tree，即可用于分类，也可以用于回归，使用基尼系数寻找最佳分裂特征
- 决策树的优缺点
  - 缺点：
    - 容易发生过拟合（随机森林）可以在很大程度上减少过拟合
    - 容易忽略数据集中属性的相互关联
    - 对于那些各类别样本数量不一致的数据，在决策树中，进行属性划分时，不同的判定准则会带来不同的属性选择倾向；信息增益准则对可取数目较多的属性有所偏好（典型代表ID3算法），而增益率准则（CART）则对可取数目较少的属性有所偏好，但CART进行属性划分时候不再简单地直接利用增益率进行划分，而是采用一种启发式规则）（只要是使用了信息增益，都有这个缺点，如RF）
    - ID3算法计算信息增益时结果偏向数值比较多的特征
  - 优点：
    - 决策树易于理解和解释，可以可视化分析，容易提取出规则

- 可以同时处理标称型（分类变量）和数值型（回归变量）数据
- 比较适合处理有缺失值的样本
- 能够处理不相关的特征
- 测试数据集时，运行速度比较快
- 在较短时间内，能够对大型数据源进行处理，并取得不错的效果

GBDT

- GBDT是一种加法模型（Additive Tree Model），属于Boosting家族的一员，同样类似的树模型有XGBoost、LightGBM
- 利用前一轮迭代的误差来更新训练集的权重，校正前一轮迭代被错误分类的样本，下一轮迭代会将重心放在上一轮分错的样本上
- GBDT理论公式推导可参考：Greedy FUnction Approximation：A Gradient Boosting Machine和Stochastic Gradient Boosting。

5.4 排序学习

基于排序的指标来优化

- 排序学习：Learning To Rank，L2R
- 经典的回归预测的评价指标是：RMSE（均方根误差）
- 实际的推荐系统场景中，更加关注的是头部推荐结果是否准确，Top N个结果的偏序关系是否满足用户需求。
- 作为排序任务，优化的目标是维持一个相对偏序关系，对预测的分数的绝对值不那么敏感
- 经典的排序指标：MRR（Mean Reciprocal Rank）、MAP（Mean Average Precision）
- NDCG（Normalized Discounted Cumulative Gain）是一个常用的指标

L2R算法的三种情况

L2R的三类算法：Point-wise、Pair-wise、List-wise

更加详细的介绍可以参考：怎么理解基于机器学习“四大支柱”划分的学习排序方法

Point-wise

- 基于单个样本进行优化，排序问题退化成通用的回归/分类问题，一般是一个二分类的任务，是机器学习的典型判别问题。

- 对于用户（query）q，两个商品D_i和D_j，排序模型的核心是根据两个商品的特征来学习一个分数映射f，使得$S_i = f(X_i)$，x_i可以是一些手工特征，也可以是一些其他模型的结果放进来集成学习
- f可以是一个逻辑回归模型、迭代决策树GBDT、也可以是一个多层的神经网络
- 问题：对于头部商品不敏感，无法有效的容忍某个用户或者某个物品的偏置
-

Pair-wise

- pair-wise将排序问题约减成一个对偏序对的二分类问题，即偏序对关系正确还是错误
- 在给定查询q的场景下，文档对的差值归一化成一个概率分布（其实就是一个二项分布，包含预测偏序关系成立和不成立的两个概率），然后根据该分布与目标标签的差异（例如交叉熵损失）来通过标准梯度下降方法进行优化
- 把两个分数的差值$Si - S_j$通过$sigmoid$函数归一化到0~1（满足概率的定义），它的含义为D_i比D_j更好的概率

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

- 定义损失函数为交叉熵损失函数，其中P'是实际的标签，所以上式P'和$(1 - P')$必有一个是零项，也就是下式只有一项不为0，然后按照标准的梯度下降就可以优化损失函数

$$C = -P'_{ij} log P_{ij} - (1 - P'_{ij}) log(1 - P_{ij})$$

- 评价指标：NDCG

List-wise

- 基于整个排序列表去优化，对于单个用户（query）而言，把整个需要排序的列表当成一个学习样本（instance），直接通过NDCG等指标来优化
- eg：AdaRank和ListNet，直接使用定义在一个排序结果列表上的损失函数
- AdaRank直接针对每一个query对整个排序列表计算与理想列表的差异，然后通过boost策略来调节不同query的权重
- 一般来说，给予list-wise比pair-wise更有效，而pair-wise比point-wise更有效，实际经验上的结果或许会有部分差异