第八节课笔记 Chapter 3.4 CTR预估算法 基于邻域的协同过滤

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

第八节课笔记		
讲义内容概括	知识点	第三次反馈

GBDT+LR算法

- 背景：CTR问题样本数量大，LR学习能力有限，人工特征工程成本高

Practical Lessons from Predicting Clicks on Ads at Facebook,2014（Facebook经典CTR预估论文）http://quinonero.net/Publications/predicting-clicks-facebook.pdf

- 原理：具有stacking思想的二分类器模型，通过GBDT将特征进行组合，然后传入给LR线性分类器 LR对GBDT产生的输入数据进行分类（使用L1正则化防止过拟合）

CART—强分类器
Boosting 多个弱分类器

Bagging 少数服从多数— 并行
Boosting 累加结果—串行

实验结果的性能分析

例子：8万个数据
为什么要分成2万GDBT+2万LR？为了防止污染

- 评价指标

NE：Normalized Cross-Entropy

$$NE = \frac{-\frac{1}{N}\sum_{i=1}^{n}\left(\frac{1+y_i}{2}log(p_i) + \frac{1-y_i}{2}log(1-p_i)\right)}{-(p*log(p)+(1-p)*log(1-p))}$$

NE = 每次展现时预测得到的log loss的平均值，除以对整个数据集的平均log loss值p代表训练数据的平均经验CTR，即background CTR，NE对background CTR不敏感，NE数值越小预测效果越好

Calibration：预估CTR除以真实CTR，即预测的点击数与真实点击数的比值。数值越接近1，预测效果越好。

AUC： 衡量排序质量的良好指标，但是无法进行校准，

也就是如果我们希望得到预估准确的比值，而不是最优的排序结果，那么需要使

用NE或者Calibration

Wide&Deep算法
论文：Wide & Deep Learning for Recommender Systems，2016
https://arxiv.org/abs/1606.07792

推荐系统的挑战是 memorization与generalization

- Wide推荐：

显性特征通过OneHot编码转换为离散特征，好处是可解释性强，不足在于特征组合需要人为操作 cross product transformation 交叉积构造新的特征

采用Linear Regression

- Deep推荐：

通过深度学习出一些向量，这些向量是隐性特征，往往没有可解释性的
连续特征，Embedding后的离散特征，

使用前馈网络模型，特征首先转换为低维稠密向量，作为第一个隐藏层的输入，解决维度爆炸问题

根据最终的loss反向训练更新。向量进行随机初始化，隐藏层的激活函数通常使用ReLU

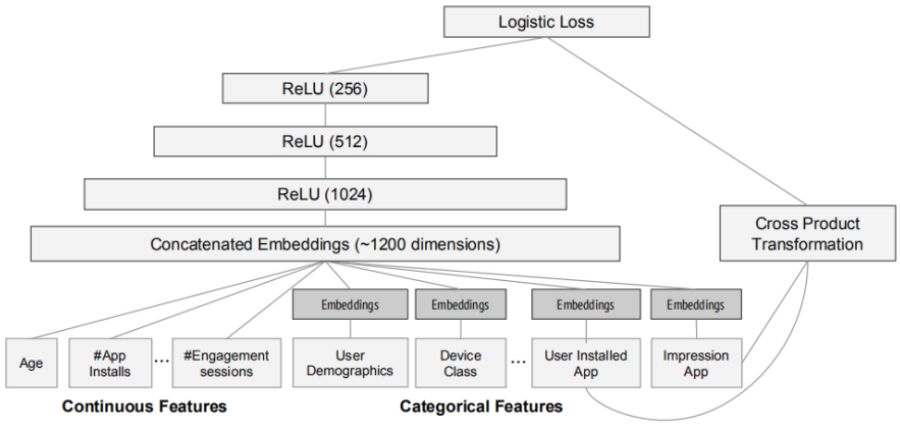
$$\alpha^{(l+1)} = f\left(W^{(l)}\alpha^{(l)} + b^{(l)}\right)$$

- Ensemble：

两个模型分别对全量数据进行预测，然后根据权重组合最终的预测结果

- joint training：

wide和deep的特征合一，构成一个模型进行预测



DeepCTR工具
https://github.com/shenweichen/DeepCTR
实现了多种CTR深度模型
与Tensorflow 1.4和2.0兼容
数据集：MovieLens_Sample
包括了多个特征：user_id, movie_id, rating, timestamp, title, genres, gender, age, occupation, zip
使用DeepCTR中的WDL，计算RMSE值

CTR模型回顾
Step1、对输入进行特征编码
在推荐系统，个性化信息检索中，需要对用户行为进行预测 需要通过OneHot编码进行转换 => 高维稀疏特征，然后进行Concatenate （大约1200维）

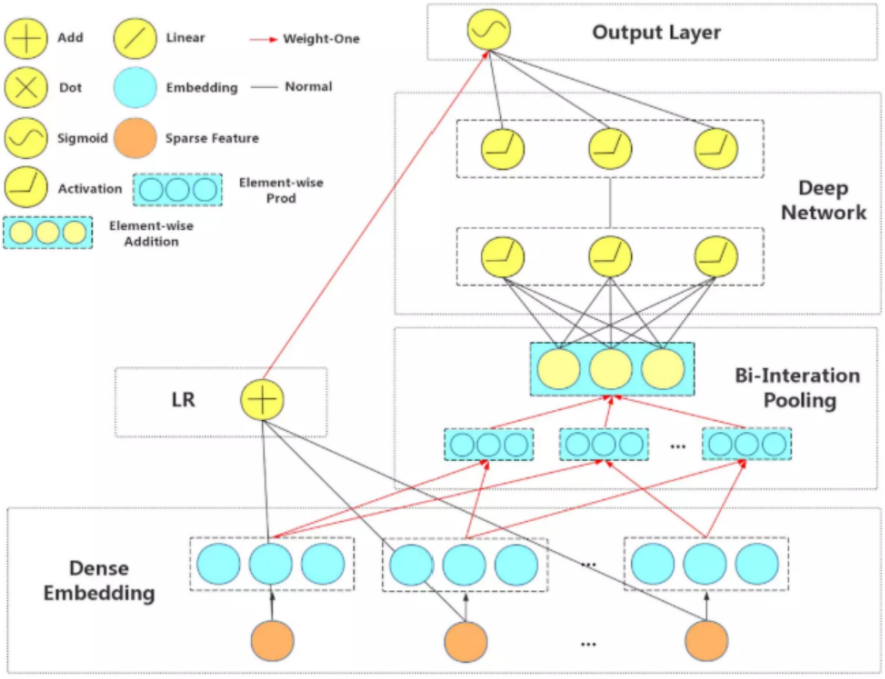
Step2，模型选择：
LR模型 or Poly2模型（Degree-2 Polynomial）or FM or FFM or FNN or W&D or DeepFM or NFM

- LR模型可解释性强，但是需要人工交叉特征，
- Poly2在LR的基础上考虑特征自动交叉，
- FM在poly2上做了一个矩阵补全，
- FNN用FM的特征进行初始化进入DNN，CTR中大部分特征是离散、高维且稀疏的，需要embedding后才能进行深度学习 使用FM对embedding层进行初始化，即每个特征对应一个偏置项 和一个k维向量
- FNN, Wide & Deep, DeepFM都是在DNN部分，对embedding之后的特征进行concatenate，没有充分进行特征交叉计算。
- NFM算法是对embedding直接采用对位相乘（element-wise）后相加起来作为交叉特征，然后通过DNN直接将特征压缩，最后concatenate linear部分和deep部分的特征。两种FM和DNN的结合方式： DeepFM, 并行结构，FM和DNN分开计算 NFM,串行架构，将FM的结果作为DNN的输入

CTR预估算法
FM与MF回顾
GBDT+LR模型：
通过GDBT完成特征的自动组合
Wide & Deep模型：WDL
NFM算法：
NFM与

DeepFM的区别
Review学习过的CTR模型
**
工具：DeepCTR
使用GBDT+LR对8万个样本进行分类预测
使用WDL，NFM对Movielens进行推荐

基于邻域的协同过滤
什么是邻居
UserCF
ItemCF
Surprise中的邻域算法
Baseline算法
基于邻域的协同过滤的特点
UserCF与ItemCF的区别



- NFM模型

对于输入X的预测公式：

$$\hat{y}_{NFM}(X) = w_0 + \sum_{i=1}^{n} w_i x_i + f(X)$$

Embedding层：全连接层，将稀疏输入映射到一个密集向量，得到 $V_x = \{x_1 v_1, \cdots x_n v_n\}$

BI层：池化操作，将一系列的Embedding向量转换为一个向量 $f_{BI}(V_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i v_i \cdot x_j v_j$

隐藏层：神经网络的全连接层

预测层：将隐藏层输出到n*1的全连接层，$V_x = \{x_1 v_1, \ldots, x_n v_n\}$ 预测结果

BI层，将每个特征embedding进行两两做元素积，BI层的输出是一个 k维向量（隐向量的大小），BI层负责了二阶特征组合

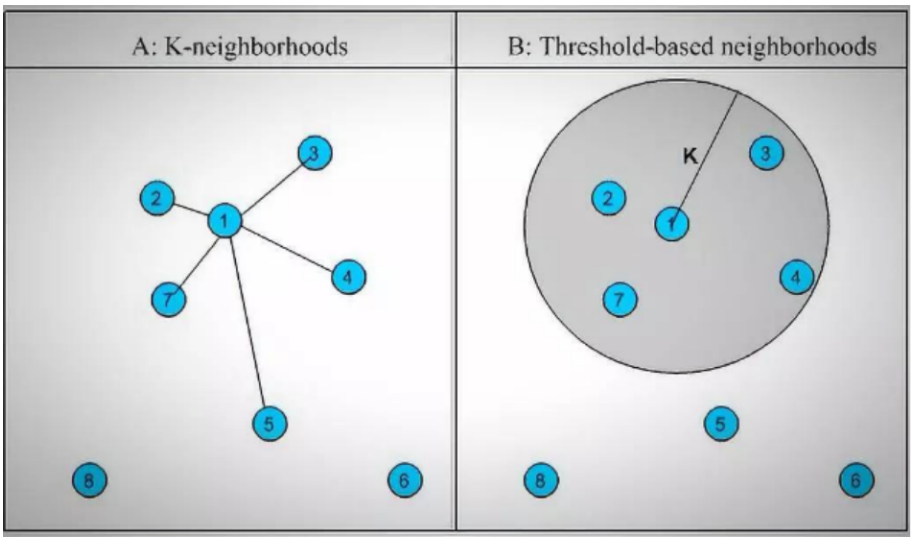
基于邻域的协同过滤

相似的邻居：2种定义

Question?

1. 如何使用GBDT进行新特征构造?

2. Deep模型？？深度学习模型要掌握些什么?



基于用户相似度，与基于物品相似度的区别：

- 于用户相似度是基于评分矩阵中的行向量相似度求解
- 基于项目相似度计算式基于评分矩阵中列向量相似度求解

	Item1	Item2	Item3	Item4	Item5	Item6
User1	4	0	1	3	2	3
User2	3	0	2	3	3	4
User3	4	5	6	0	2	0
User4	0	3	1	0	0	2
User5	5	0	2	0	0	2
User6	4	2	2	1	2	4

基于用户的协同过滤（UserCF）

基于用户的协同过滤（UserCF）

- 利用行为的相似度计算用户的相似度
- Step1，找到和目标用户兴趣相似的用户集合

Jaccard相似度计算 $$w_{uv}=\frac{|N(u)\cap N(v)|}{|N(u)\cup N(v)|}$$

余弦相似度 $$w_{uv}=\frac{|N(u)\cap N(v)|}{\sqrt{|N(u)||N(v)|}}$$

改进的相似度 $$w_{uv}=\frac{\sum_{i\in N(u)\cap N(v)}\frac{1}{\log(1+|N(i)|}}{\sqrt{|N(u)||N(v)|}}$$

基于流行度改进的相似度计算

通过 $\frac{1}{\log(1+|N(i)|}$ 惩罚了热门物品对相似度的影响力

基于用户的协同过滤（UserCF）

- Step2，用户u对物品i的相似度，等价于K个邻居对物品i的兴趣度

$$p(u,i)=\sum_{v\in S(u,K)\cap N(i)} w_{uv} r_{vi}$$

S(u,K)表示和用户u兴趣最接近的K个用户
N(i)表示对物品i有过行为的用户集合
w_{uv}表示用户u和v的兴趣相似度
r_{vi}表示用户v对物品i的兴趣

- Step3，为用户u生成推荐列表

把和用户兴趣相同的k个邻居，喜欢的物品进行汇总，去掉用户u已经喜欢过的物品，剩下按照从大到小进行推荐

假设喜欢同一物品的用户喜好相似

基于物品的协同过滤（ItemCF）—原理和UserCF类似，一个用行向量一个用列向量

基于物品的协同过滤（ItemCF）

- 利用行为的相似度计算物品的相似度
- Step1，计算物品之间相似度

$$w_{ij}=\frac{|N(i)\cap N(j)|}{|N(i)|}$$

N(i)，喜欢物品i的用户数

$$w_{ij}=\frac{|N(i)\cap N(j)|}{\sqrt{|N(i)||N(j)|}}$$

如果N(j)过大，说明j是热门物品很多人都喜欢
需要对N(j)进行惩罚，避免推荐热门物品

基于物品的协同过滤（ItemCF）

- Step2，用户u对物品i的兴趣度，等价于物品i的K个邻居物品，受到用户u的兴趣度

$$p(u,i)=\sum_{j\in S(i,K)\cap N(u)} w_{ij} r_{uj}$$

S(i,K)表示和物品i最相似的K个物品集合
N(u)表示用户u喜欢的物品集合
w_{ij}表示物品i和j的相似度
r_{uj}表示用户u对物品j的兴趣

- Step3，为用户u生成推荐列表

和用户历史上感兴趣的物品越相似的物品，越有可能在用户的推荐列表中获得比较高的排名

预测用户u对物品的兴趣度，去掉用户u已经喜欢过的物品，剩下按照从大到小进行推荐

Surprise工具
https://surprise.readthedocs.io/en/stable/prediction_algorithms_package.html
基于邻域的协同过滤算法

1. knns.KNNBasic
2. knns.KNNWithMeans
3. knns.KNNWithZScore
4. knns.KNNBaseline

这四个算法都是在用户对物品的评分矩阵上做文章，2号是在1号上加权一个与平均值的差值，最后再把平均值加上，3号加权的是正态分布值，四号是加权一个与baseline项的差值

Surprise中的相似度度量方式

Surprise中的相似度度量方式

相似度度量标准	度量标准说明
1：cosine	用户（items）之间的cosine 相似度
2：msd	用户（items）之间的均方差误差
3：pearson	用户（items）之间的皮尔逊相关系数
4：pearson_baseline	计算用户（item）之间的（缩小的）皮尔逊相关系数，使用基准值进行居中而不是平均值。

$$\text{cosine\_sim}(u,v)=\frac{\sum_{i\in I_{uv}} r_{ui}\cdot r_{vi}}{\sqrt{\sum_{i\in I_{uv}} r_{ui}^2}\cdot\sqrt{\sum_{i\in I_{uv}} r_{vi}^2}}$$

$$\text{cosine\_sim}(i,j)=\frac{\sum_{u\in U_{ij}} r_{ui}\cdot r_{uj}}{\sqrt{\sum_{u\in U_{ij}} r_{ui}^2}\cdot\sqrt{\sum_{u\in U_{ij}} r_{uj}^2}}$$

$$\text{msd}(u,v)=\frac{1}{|I_{uv}|}\cdot\sum_{i\in I_{uv}}(r_{ui}-r_{vi})^2$$

$$\text{msd}(i,j)=\frac{1}{|U_{ij}|}\cdot\sum_{u\in U_{ij}}(r_{ui}-r_{uj})^2$$

$$\text{pearson\_sim}(u,v)=\frac{\sum_{i\in I_{uv}}(r_{ui}-\mu_u)\cdot(r_{vi}-\mu_v)}{\sqrt{\sum_{i\in I_{uv}}(r_{ui}-\mu_u)^2}\cdot\sqrt{\sum_{i\in I_{uv}}(r_{vi}-\mu_v)^2}}$$

$$\text{pearson\_sim}(i,j)=\frac{\sum_{u\in U_{ij}}(r_{ui}-\mu_i)\cdot(r_{uj}-\mu_j)}{\sqrt{\sum_{u\in U_{ij}}(r_{ui}-\mu_i)^2}\cdot\sqrt{\sum_{u\in U_{ij}}(r_{uj}-\mu_j)^2}}$$

$$\text{pearson\_baseline\_sim}(u,v)=\hat{\rho}_{uv}=\frac{\sum_{i\in I_{uv}}(r_{ui}-b_{ui})\cdot(r_{vi}-b_{vi})}{\sqrt{\sum_{i\in I_{uv}}(r_{ui}-b_{ui})^2}\cdot\sqrt{\sum_{i\in I_{uv}}(r_{vi}-b_{vi})^2}}$$

$$\text{pearson\_baseline\_sim}(i,j)=\hat{\rho}_{ij}=\frac{\sum_{u\in U_{ij}}(r_{ui}-b_{ui})\cdot(r_{uj}-b_{uj})}{\sqrt{\sum_{u\in U_{ij}}(r_{ui}-b_{ui})^2}\cdot\sqrt{\sum_{u\in U_{ij}}(r_{uj}-b_{uj})^2}}$$

Baseline算法
使用ALS进行优化 Step1，固定bu，优化bi Step2，固定bi，优化bu

机器学习的目的：预测和可解释性
信用卡欺诈分析数据集：
https://www.kaggle.com/mlg-ulb/creditcardfraud

- 推荐系统中的可解释性：

以物品为媒介
以用户为媒介
以特征为媒介
对这三种媒介间的关联探索还不够

- 模型的可解释性

线性模型
深度模型DNN

总结Summary

- 记录作业内容
  - Thinking1：在CTR点击率预估中，使用GBDT+LR的原理是什么？
  - Thinking2：Wide & Deep的模型结构是怎样的，为什么能通过具备记忆和泛化能力（memorization and generalization）
  - Thinking3：在CTR预估中，使用FM与DNN结合的方式，有哪些结合的方式，代表模型有哪些？
  - Thinking4：Surprise工具中的baseline算法原理是怎样的？BaselineOnly和KNNBaseline有什么区别？
  - Thinking5：GBDT和随机森林都是基于树的算法，它们有什么区别？
  - Thinking6：基于邻域的协同过滤都有哪些算法，请简述原理
  - Action1：使用Wide&Deep模型对movielens进行评分预测
  - Action2：使用基于邻域的协同过滤（KNNBasic, KNNWithMeans, KNNWithZScore, KNNBaseline中的任意一种）对MovieLens数据集进行协同过滤，采用k折交叉验证(k=3)，输出每次计算的RMSE, MAE

-

L8优秀作业-陈学良
https://github.com/xueliang787/RS06-08

L9优秀作业
陈学良
https://github.com/xueliang787/RS06-09
陶学节
https://github.com/arthurt53/RS6-work/tree/master/L9