

## 目录

请详细说说协同过滤的原理.....	2
搜索和推荐中的精度和召回(recall)分别是什么意思? .....	14
推荐系统有哪些常用的评价标准.....	18
请详细聊聊推荐系统里的排序算法.....	20
怎么解决推荐系统中的冷启动问题? .....	33
请聊聊你所了解的推荐系统算法.....	38
了解隐语义模型在推荐系统中的应用的么? .....	42
如何通俗理解奇异值分解.....	51
请通俗的解释下什么叫张量分解? .....	70
请说说隐语义模型 LFM 背后的原理.....	76
请详细说说你对 Learning to rank 的通俗理解.....	83
简述推荐系统的演进史 .....	116
推荐系统评估(从用户\内容提供方\网站角度).....	117
推荐系统常用评估指标 .....	117
推荐系统准确性(学术界的)的评估函数.....	118
推荐系统多样性&新颖性&惊喜性.....	119
解释 Exploitation & Exploration.....	119
Bandit 算法-原理 .....	119
什么是多层重叠试验框架?.....	120
预训练方法 BERT 和 OpenAI GPT 有什么区别? .....	120
对比 BERT、OpenAI GPT、ELMo 预训练模型架构间差异.....	121
GCN 方法分为哪两类? .....	121
什么是图卷积神经网络? .....	122
如何理解图卷积算法? .....	122
GCN 有哪些特征? .....	123
已知基于数据驱动的机器学习和优化技术在单场景内的 A/B 测试上, 点击率、转化率、成交额、单价都取得了不错的效果。但是, 目前各个场景之间是完全独立优化的, 这样会带来哪些比较严重的问题? .....	123
什么是多场景联合排序算法? .....	124
IRGAN 框架具体是什么? .....	124
推荐页与搜索页特性有什么不同? .....	125
推荐系统常用的评价指标有哪些? .....	125
请画出点击率预估流程图, 分为那两部分? .....	127
使用 FFM 有哪些需要注意的地方.....	128
什么是 DSSM?有什么优缺点? .....	128
什么是 wide&deep 模型? .....	129
Join training 和 ensemble training 有什么区别? .....	130
简述 DeepFM 模型? .....	130
Collaborative Knowledge Base Embedding 使用哪三种知识的学习? .....	131
怎样将知识图谱引入推荐系统? .....	131
普通的逻辑回归能否用于大规模的广告点击率预估, 为什么? .....	132
FTRL 在准备训练数据(特征工程)和训练模型时有哪些 trick? .....	132

阿里最新开源的 X-Deep Learning 为 Online Learning 提供了哪些解决方案? .....	133
特征交叉(特征组合)方式有哪些?.....	134
特征选择的方法有哪些? .....	135
简述 Multi-task learning(MLT)多任务学习 .....	135
如何离线评价召回阶段各种模型算法的好坏? 由于没有明确的召回预期值, 所以无论 rmse 还是 auc 都不知道该怎么做? .....	136

## 请详细说说协同过滤的原理

解析:

### 1 推荐引擎的分类

推荐引擎根据不同依据如下分类:

根据其是不是为不同的用户推荐不同的数据, 分为基于大众行为(网站管理员自行推荐, 或者基于系统所有用户的反馈统计计算出的当下比较流行的物品)、及个性化推荐引擎(帮你找志同道合, 趣味相投的朋友, 然后在此基础上实行推荐);

根据其数据源, 分为基于人口统计学的(用户年龄或性别相同判定为相似用户)、基于内容的(物品具有相同关键词和 Tag, 没有考虑人为因素), 以及基于协同过滤的推荐(发现物品, 内容或用户的相关性推荐, 分为三个子类, 下文阐述);

根据其建立方式, 分为基于物品和用户本身的(用户-物品二维矩阵描述用户喜好, 聚类算法)、基于关联规则的(The Apriori algorithm 算法是一种最有影响的挖掘布尔关联规则频繁项集的算法)、以及基于模型的推荐(机器学习, 所谓机器学习, 即让计算机像人脑一样持续学习, 是人工智能领域内的一个子领域)。

关于上述第二个分类(2、根据其数据源)中的基于协同过滤的推荐：随着 Web2.0 的发展，Web 站点更加提倡用户参与和用户贡献，因此基于协同过滤的推荐机制因运而生。它的原理很简单，就是根据用户对物品或者信息的偏好，发现物品或者内容本身的相关性，或者是发现用户的相关性，然后再基于这些关联性进行推荐。

而基于协同过滤的推荐，又分三个子类：

基于用户的推荐(通过共同口味与偏好找相似邻居用户，K-邻居算法，你朋友喜欢，你也可能喜欢)，

基于项目的推荐(发现物品之间的相似度，推荐类似的物品，你喜欢物品 A，C 与 A 相似，可能也喜欢 C)，

基于模型的推荐(基于样本的用户喜好信息构造一个推荐模型，然后根据实时的用户喜好信息预测推荐)。

我们看到，此协同过滤算法最大限度的利用用户之间，或物品之间的相似相关性，而后基于这些信息的基础上实行推荐。下文还会具体介绍此协同过滤。

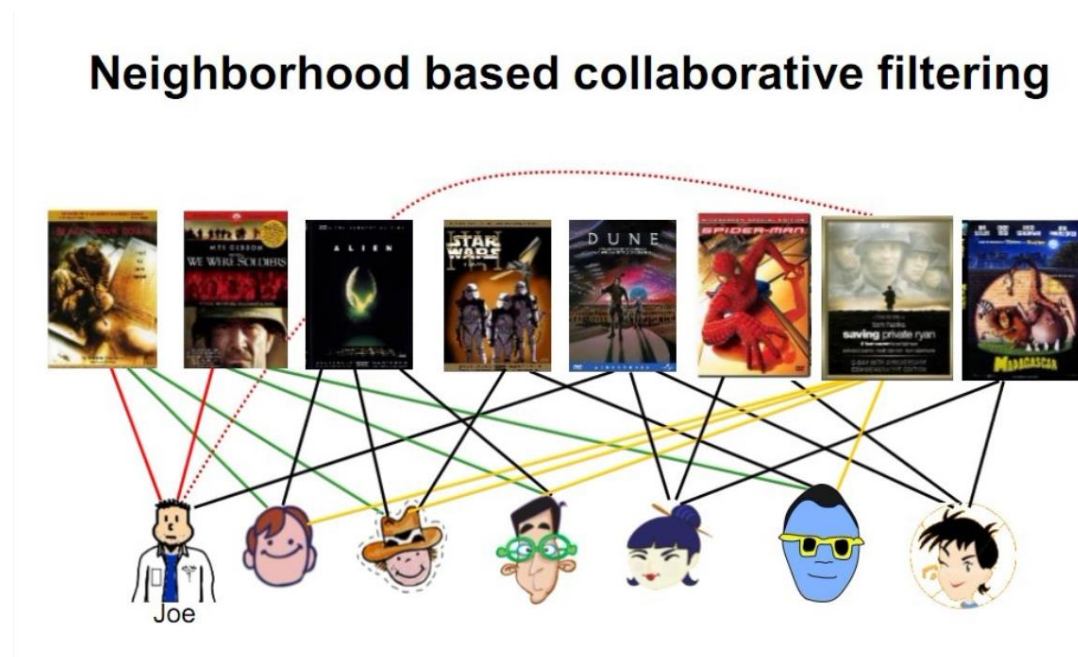
不过一般实践中，我们通常还是把推荐引擎分两类：

第一类称为协同过滤，即基于相似用户的协同过滤推荐（用户与系统或互联网交互留下的一切信息、蛛丝马迹，或用户与用户之间千丝万缕的联系），以及基于相似项目的协同过滤推荐（尽最大可能发现物品间的相似度）；

第二类便是基于内容分析的推荐（调查问卷，电子邮件，或者推荐引擎对本 blog 内容的分析）。

## 2 协同过滤推荐

协同过滤是利用集体智慧的一个典型方法。要理解什么是协同过滤 (Collaborative Filtering, 简称 CF)，首先想一个简单的问题，如果你现在想看个电影，但你不知道具体看哪部，你会怎么做？大部分的人会问问周围的朋友或者称之为广义上的邻居(neighborhood)，看看最近有什么好看的电影推荐，而我们一般更倾向于从口味比较类似的朋友那里得到推荐。这就是协同过滤的核心思想。如下图，你能从图中看到多少信息？



### 2.1 协同过滤推荐步骤

做协同过滤推荐，一般要做好以下几个步骤：

1) 若要做协同过滤，那么收集用户偏好则成了关键。可以通过用户的行为诸如评分（如不同的用户对不同的作品有不同的评分，而评分接近则意味着喜好口味相近，便可判定为相似用户），投票，转发，保存，书签，标记，评论，点击流，页面停留时间，是否购买等获得。如下面第 2 点所述：所有这些信息都可以数字化，如一个二维矩阵表示出来。

2) 收集了用户行为数据之后，我们接下来便要对数据进行减噪与归一化操作(得到一个用户偏好的二维矩阵，一维是用户列表，另一维是物品列表，值是用户对物品的偏好，一般是  $[0,1]$  或者  $[-1, 1]$  的浮点数值)。

下面再简单介绍下减噪和归一化操作。

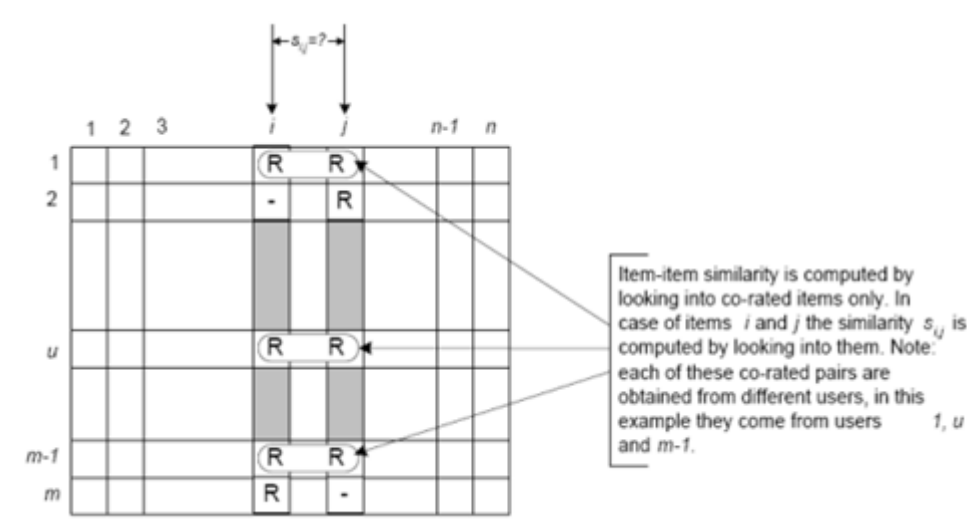
所谓减噪：用户行为数据是用户在使用应用过程中产生的，它可能存在大量的噪音和用户的误操作，我们可以通过经典的数据挖掘算法过滤掉行为数据中的噪音，这样可以是我们的分析更加精确（类似于网页的去噪处理）。

所谓归一化：将各个行为的数据统一在一个相同的取值范围中，从而使得加权求和得到的总体喜好更加精确。最简单的归一化处理，便是将各类数据除此类中的最大值，以保证归一化后的数据取值在  $[0,1]$  范围中。至于所谓的加权，很好理解，因为每个人占的权值

不同，类似于一场唱歌比赛中对某几个选手进行投票决定其是否晋级，观众的投票抵 1 分，专家评委的投票抵 5 分，最后得分最多的选手直接晋级。

3) 找到相似的用户和物品，通过什么途径找到呢？便是计算相似用户或相似物品的相似度。

4) 相似度的计算有多种方法，不过都是基于向量 Vector 的，其实也就是计算两个向量的距离，距离越近相似度越大。在推荐中，用户-物品偏好的二维矩阵下，我们将某个或某几个用户对某两个物品的偏好作为一个向量来计算两个物品之间的相似度，或者将两个用户对某个或某几个物品的偏好作为一个向量来计算两个用户之间的相似度。



所以说，很简单，找物品间的相似度，用户不变，找多个用户对物品的评分；找用户间的相似度，物品不变，找用户对某些个物品的评分。

5) 而计算出来的这两个相似度则将作为基于用户、项目的两项协同过滤的推荐。

常见的计算相似度的方法有：欧几里德距离，皮尔逊相关系数（如两个用户对多个电影的评分，采取皮尔逊相关系数等相关计算方法，可以抉择出他们的口味和偏好是否一致），Cosine 相似度，Tanimoto 系数。

下面，简单介绍其中的欧几里得距离与皮尔逊相关系数：

欧几里德距离 (Euclidean Distance) 是最初用于计算欧几里德空间中两个点的距离，假设  $x, y$  是  $n$  维空间的两个点，它们之间的欧几里德距离是：

$$d(x, y) = \sqrt{\sum (x_i - y_i)^2}$$

可以看出，当  $n=2$  时，欧几里德距离就是平面上两个点的距离。当用欧几里德距离表示相似度，一般采用以下公式进行转换：距离越小，相似度越大（同时，避免除数为 0）：

$$sim(x, y) = \frac{1}{1 + d(x, y)}$$

余弦相似度 Cosine-based Similarity

两个项目  $i, j$  视作为两个  $m$  维用户空间向量，相似度计算通过计算两个向量的余弦夹角，那么，对于  $m \times n$  的评分矩阵， $i, j$  的相似度  $sim(i, j)$  计算公式：

$$sim(i, j) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

（其中 " $\cdot$ " 记做两个向量的内积）

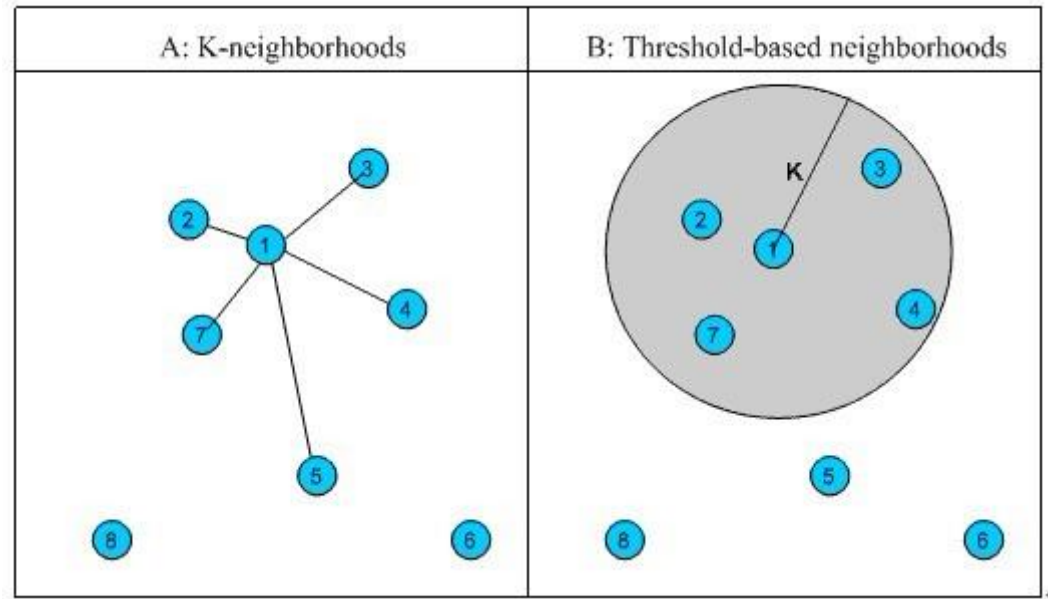
皮尔逊相关系数一般用于计算两个定距变量间联系的紧密程度，为了使计算结果精确，需要找出共同评分的用户。记用户集  $U$  为既评论了  $i$  又评论了  $j$  的用户集，那么对应的皮尔森相关系数计算公式为：

$$sim(i, j) = \frac{\sum_{u \in U} (R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U} (R_{u,i} - \bar{R}_i)^2} \sqrt{\sum_{u \in U} (R_{u,j} - \bar{R}_j)^2}}$$

其中  $R_{u,i}$  为用户  $u$  对项目  $i$  的评分，对应带横杠的为这个用户集  $U$  对项目  $i$  的评分平均。

6) 相似邻居计算。邻居分为两类：

- 1、固定数量的邻居 K-neighborhoods（或 Fix-size neighborhoods），不论邻居的“远近”，只取最近的  $K$  个，作为其邻居，如下图 A 部分所示；
- 2、基于相似度门槛的邻居，落在以当前点为中心，距离为  $K$  的区域中的所有点都作为当前点的邻居，如下图 B 部分所示。





再介绍一下 K 最近邻(k-Nearest Neighbor, KNN)分类算法: 这是一个理论上比较成熟的方法, 也是最简单的机器学习算法之一。该方法的思路是: 如果一个样本在特征空间中的 k 个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别, 则该样本也属于这个类别。

7) 经过 4)计算出来的基于用户的 CF(基于用户推荐之用: 通过共同口味与偏好找相似邻居用户, K-邻居算法, 你朋友喜欢, 你也可能喜欢), 基于物品的 CF(基于项目推荐之用: 发现物品之间的相似度, 推荐类似的物品, 你喜欢物品 A, C 与 A 相似, 那么你可能也喜欢 C)。

## 2.2 基于用户相似度与项目相似度

上述 3.1 节中三个相似度公式是基于项目相似度场景下的, 而实际上, 基于用户相似度与基于项目相似度计算的一个基本的区别是, 基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, 然后三个公式分别都可以适用, 如下图:

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	4	0	1	3	2	3
User 2	3	0	2	3	3	4
User 3	4	5	6	0	2	0
User 4	0	3	1	0	0	2
User 5	5	0	2	0	0	2
User 6	4	2	2	1	2	4

(其中，为 0 的表示未评分)

基于项目相似度计算式计算如 Item3, Item4 两列向量相似度;

基于用户相似度计算式计算如 User3, User4 量行向量相似度。

千言万语不如举个例子。我们来看一个具体的基于用户相似度计算的例子。

假设我们有一组用户，他们表现出了对一组图书的喜好。用户对一本图书的喜好程度越高，就会给其更高的评分。我们来通过一个矩阵来展示它，行代表用户，列代表图书。

如下图所示，所有的评分范围从 1 到 5，5 代表喜欢程度最高。第一个用户（行 1）对第一本图书（列 1）的评分是 4，空的单元格表示用户未给图书评分。

						
	4	3			5	
	5		4		4	
	4		5	3	4	
		3				5
		4				4
			2	4		5

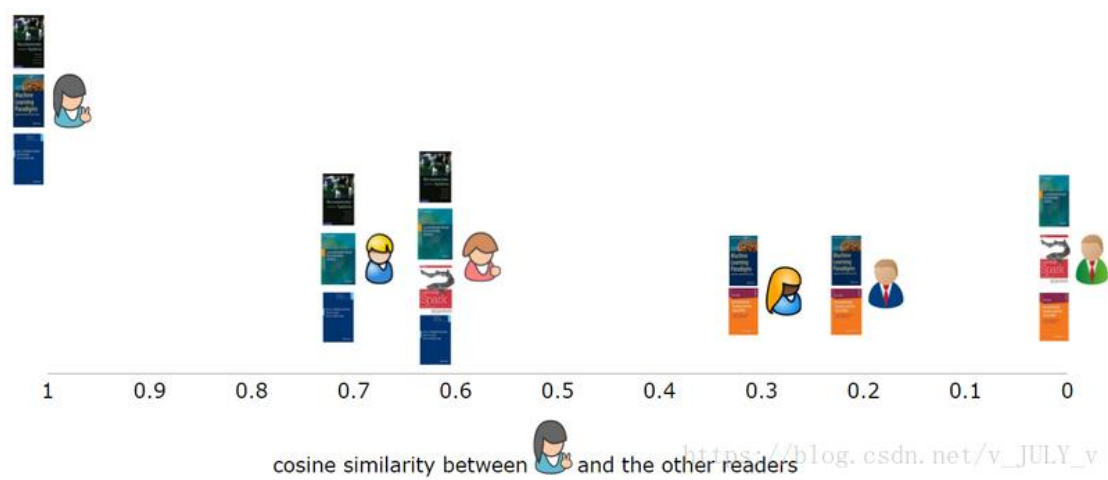
[https://blog.csdn.net/y\\_JULY\\_y](https://blog.csdn.net/y_JULY_y)

使用基于用户的协同过滤方法，我们首先要做的是基于用户给图书做出的评价，计算用户之间的相似度。













让我们从一个单一用户的角度考虑这个问题，看图 1 中的第一行，要做到这一点，常见的做法是将使用包含了用户喜好项的向量（或数组）代表每一个用户。相较于使用多样化的相似度量这种做法，更直接。

在这个例子中，我们将使用余弦相似性去计算用户间的相似度。当我们把第一个用户和其他五个用户进行比较时，就能直观的看到他和其他用户的相似程度。对于大多数相似度量，向量之间相似度越高，代表彼此更相似。本例中，第一个用户第二、第三个用户非常

相似，有两本共同书籍，与第四、第五个用户的相似度低一些，只有一本共同书籍，而与最后一名用户完全不相似，因为没有一本共同书籍。

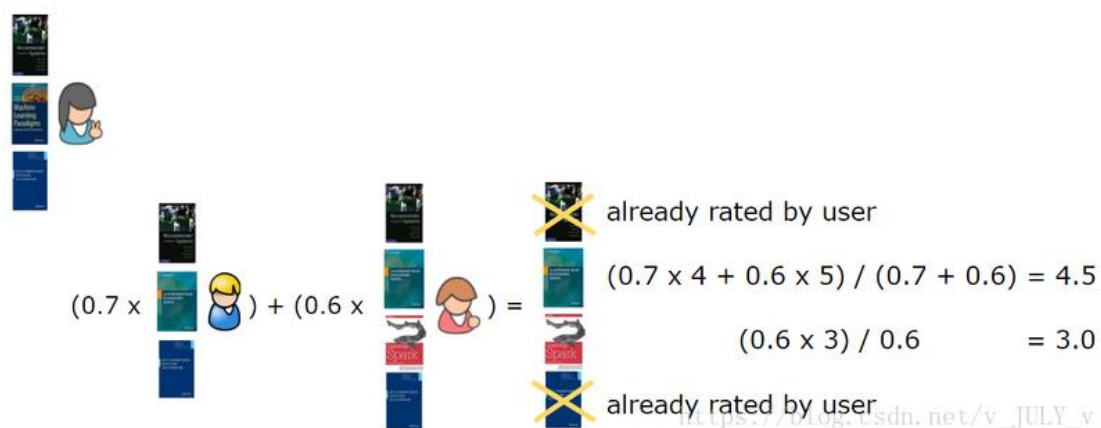


更一般的，我们可以计算出每个用户的相似性，并且在相似矩阵中表示它们。这是一个对称矩阵，单元格的背景颜色表明用户相似度的高低，更深的红色表示它们之间更相似。

						
	1.00	0.75	0.63	0.22	0.30	0.00
	0.75	1.00	0.91	0.00	0.00	0.16
	0.63	0.91	1.00	0.00	0.00	0.40
	0.22	0.00	0.00	1.00	0.97	0.64
	0.30	0.00	0.00	0.97	1.00	0.53
	0.00	0.16	0.40	0.64	0.53	1.00

所以，我们找到了与第一个用户最相似的第二个用户，删除用户已经评价过的书籍，给最相似用户正在阅读的书籍加权，然后计算出总和。

在这种情况下，我们计算出  $n=2$ ，表示为了产生推荐，需要找出与目标用户最相似的两个用户，这两个用户分别是第二个和第三个用户，然后第一个用户已经评价了第一和第五本书，故产生的推荐书是第三本（4.5 分），和第四本（3 分）。



此外，什么时候用 item-base，什么时候用 user-base 呢：<http://weibo.com/1580904460/zhZ9AilkZ?mod=weibotime?>

一般说来，如果 item 数目不多，比如不超过十万，而且不显著增长的话，就用 item-based 好了。为何？如@wuzh670 所说，如果 item 数目不多+不显著增长，说明 item 之间的关系在一段时间内相对稳定(对比 user 之间关系)，对于实时更新 item-similarity 需求就降低很多，推荐系统效率提高很多，故用 item-based 会明智些。

反之，当 item 数目很多，建议用 user-base。当然，实践中具体情况具体分析。如下图所示（摘自项亮的《推荐系统实践》一书）：

表2-11 UserCF和ItemCF优缺点的对比

	UserCF	ItemCF
性能	适用于用户较少的场合, 如果用户很多, 计算用户相似度矩阵代价很大	适用于物品数明显小于用户数的场合, 如果物品很多(网页), 计算物品相似度矩阵代价很大
领域	时效性较强, 用户个性化兴趣不太明显的领域	长尾物品丰富, 用户个性化需求强烈的领域
实时性	用户有新行为, 不一定造成推荐结果的立即变化	用户有新行为, 一定会导致推荐结果的实时变化
冷启动	在新用户对很少的物品产生行为后, 不能立即对他进行个性化推荐, 因为用户相似度表是每隔一段时间离线计算的  新物品上线后一段时间, 一旦有用户对物品产生行为, 就可以将新物品推荐给对它产生行为的用户兴趣相似的其他用户	新用户只要对一个物品产生行为, 就可以给他推荐和该物品相关的其他物品  但没有办法在不离线更新物品相似度表的情况下将新物品推荐给用户
推荐理由	很难提供令用户信服的推荐解释	利用用户的历史行为给用户做推荐解释, 可以令用户比较信服

### 3 参考文献

- 1 推荐引擎算法学习导论: 协同过滤、聚类、分类: [https://blog.csdn.net/v\\_july\\_v/article/details/7184318](https://blog.csdn.net/v_july_v/article/details/7184318)
- 2 协同过滤推荐算法和基于内容的过滤算法: <https://time.geekbang.org/article/194>

## 搜索和推荐中的精度和召回(recall)分别是什么意思?

解析:

精度/精确率, 和召回率是广泛用于信息检索和统计学分类领域的两个度量值, 用来评价结果的质量。

其中精度是检索出相关文档数与检索出的文档总数的比率，衡量的是检索系统的查准率；

召回率是指检索出的相关文档数和文档库中所有的相关文档数的比率，衡量的是检索系统的查全率。

一般来说，Precision 就是检索出来的条目（比如：文档、网页等）有多少是准确的，Recall 就是所有准确的条目有多少被检索出来了。

正确率、召回率和 F 值是在鱼龙混杂的环境中，选出目标的重要评价指标。不妨看看这些指标的定义先：

1. 精确率 = 提取出的正确信息条数 / 提取出的信息条数
2. 召回率 = 提取出的正确信息条数 / 样本中的信息条数

顺便说一句，如果两者取值在 0 和 1 之间，数值越接近 1，查准率或查全率就越高。比如

定义：F 值 = 正确率 \* 召回率 \* 2 / (正确率 + 召回率) (F 值即为正确率和召回率的调和平均值)

$$k\text{精度}(\textit{precision at } k) = \frac{\text{所推荐的}k\text{个物品中相关物品的个数}}{k}$$

$$k\text{召回}(\textit{recall at } k) = \frac{\text{所推荐的}k\text{个物品中相关物品的个数}}{\text{所有相关物品的个数}}$$

这就好比推荐系统根据你的喜好，推荐了 10 个商品，其中真正相关的是 5 个商品。在所有商品当中，相关的商品一共有 20 个，那么

$$k \text{ 精度} = 5 / 10$$

$$k \text{ 召回} = 5 / 20$$

咱们再看下先第二个例子。比如搜：北京大学，有三个网页被搜索到了：

- a. 北京大学保安考上研究生
- b. 北京互联网工作招聘
- c. 大学生活是什么样的

其中只有 a 是被正确搜索到的，其他两个其实是和用户搜索词无关，而事实上数据库里还有这种网页：

- d. 北大开学季
- e. 未名湖的景色

这两个没被搜索到，但 d、e 和“北京大学”的相关度是超过 b、c 的，也就是应该被搜索（被召回）到的却没有显示在结果里，即：

$$\text{精确率} = (a) / (a + b + c)$$

$$\text{召回率} = (a) / (a + d + e)$$

不妨再看第三个例子：某池塘有 1400 条鲤鱼，300 只虾，300 只鳖。现在以捕鲤鱼为目的。撒一大网，逮着了 700 条鲤鱼，200 只虾，100 只鳖。那么，这些指标分别如下：



代表查准率的正确率 =  $700 / (700 + 200 + 100) = 70\%$

代表查全率的召回率 =  $700 / 1400 = 50\%$

F 值 =  $70\% * 50\% * 2 / (70\% + 50\%) = 58.3\%$

不妨看看如果把池子里的所有的鲤鱼、虾和鳖都一网打尽，这些指标又有何变化：

正确率 =  $1400 / (1400 + 300 + 300) = 70\%$

召回率 =  $1400 / 1400 = 100\%$

F 值 =  $70\% * 100\% * 2 / (70\% + 100\%) = 82.35\%$

由此可见，正确率是评估捕获的成果中目标成果所占得比例；召回率，顾名思义，就是从关注领域中，召回目标类别的比例；而 F 值，则是综合这二者指标的评估指标，用于综合反映整体的指标。

当然希望检索结果 Precision 越高越好，同时 Recall 也越高越好，但事实上这两者在某些情况下有矛盾的。比如极端情况下，我们只搜索出了一个结果，且是准确的，那么 Precision 就是 100%，但是 Recall 就很低；而如果我们把所有结果都返回，那么比如 Recall 是 100%，但是 Precision 就会很低。因此在不同的场合中需要自己判断希望 Precision 比较高或是 Recall 比较高。如果是做实验研究，可以绘制 Precision-Recall 曲线来帮助分析。

# 推荐系统有哪些常用的评价标准

解析：

常用的评价标准：

一类是线上的评测，比如通过点击率、网站流量、A/B test 等判断。这类评价标准在这里就不细说了，因为它们并不能参与到线下训练模型和选择模型的过程当中。

第二类是线下评测。评测标准很多，我挑几个常用的。我就拿给用户推荐阅读相关链接来举例好了。

1. 精度 Precision:  $P(k)$

$$P(k) = c/k$$

我们给某个用户推荐了  $k$  个链接，他 / 她点击了其中的  $c$  个链接，那么精度就是  $c/k$ 。

2. 平均精度 Average Precision:  $ap@n$

$$ap@n = \sum_{k=1}^n \frac{P(k)}{\min(m, n)}$$

$n$  是被预测的链接的总数， $m$  是用户点击的链接的总数。

例子 1： 我们一共推荐了 10 个链接，用户实际上点击了我们推荐当中的第 1 个和第 4 个链接，以及另外两个其他的链接，那么对于这个用户，

$$ap@10 = (1/1 + 2/4) / 4 \approx 0.38$$

例子 2： 我们一共推荐了 10 个链接，用户实际上点击了我们推荐当中的第 2 个,第 3 个和第 5 个链接，以及另外三个其他的链接，那么对于这个用户，

$$ap@10=(1/2+2/3+3/5)/6\approx0.29$$

例子 3： 我们一共推荐了 10 个链接，用户实际上点击了我们推荐当中的第 2 个,第 7 个，此外没有点击其他联系，那么对于这个用户，

$$ap@10=(1/2+2/7)/2\approx0.39$$

例子 4： 我们一共推荐了 5 个链接，用户实际上点击了我们推荐当中的第 1 个,第 2 个和第 4 个，以及另外 6 个其他链接，那么对于这个用户，

$$ap@5=(1/1+2/2+3/4)/5\approx0.55$$

### 3. 平均精度均值 Mean Average Precision: MAP@n

MAP 计算的是 N 个用户的平均精度的均值。

$$MAP@n = \sum_{i=1}^N \frac{(ap@n)_i}{N}$$

这个 N 是用户数量。

比如说我们三个用户甲、乙、丙分别推荐了 10 个链接，

甲点击了我们推荐当中的第 1 个和第 4 个链接，以及另外两个其他的链接，那么 $(ap@10)_1=(1/1+2/4)/4\approx0.38$ .

乙点击了我们推荐当中的第 3 个链接，以及另外一个其他的链接，那么 $(ap@10)_2=(1/3)/2\approx0.17$ .

丙点击了我们推荐当中的第 1 个链接，第 7 个链接，以及另外三个其他的链接，那么 $\text{ap}@10)_3 = (1/1 + 2/7)/5 \approx 0.26$ .

那么这个模型的平均精度均值

$$\text{MAP}@10 = (0.38 + 0.17 + 0.26)/3 \approx 0.27$$

更多请参考：[http://sofasofa.io/forum\\_main\\_post.php?postid=1000292](http://sofasofa.io/forum_main_post.php?postid=1000292)

## 请详细聊聊推荐系统里的排序算法

解析：

### 一、个性化推荐

在海量的内容在满足了我们需求的同时，也使我们寻找所需内容更加困难，在这种情况下个性化推荐应运而生。

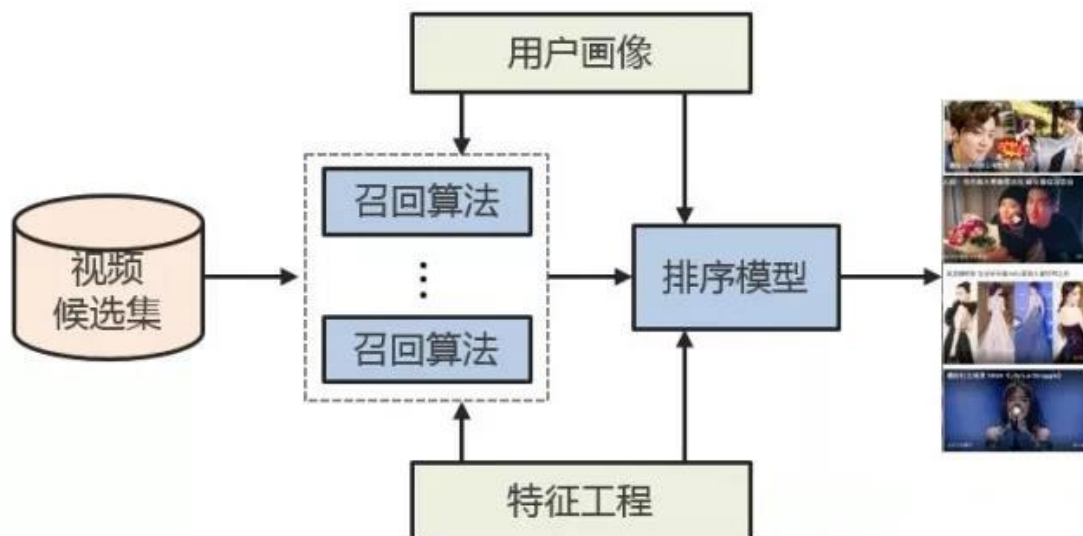
在当前这个移动互联网时代，除了专业内容的丰富，UGC 内容更是爆发式发展，每个用户既是内容的消费者，也成为了内容的创造者。这些海量的内容在满足了我们需求的同时，也使我们寻找所需内容更加困难，在这种情况下个性化推荐应运而生。

个性化推荐是在大数据分析和人工智能技术的基础上，通过研究用户的兴趣偏好，进行个性化计算，从而给用户提供高质量的个性化内容，解决信息过载的问题，更好的满足用户的需求。

## 二、爱奇艺推荐系统介绍

我们的推荐系统主要分为两个阶段，召回阶段和排序阶段。

召回阶段根据用户的兴趣和历史行为，同千万级的视频库中挑选出一个小的候选集（几百到几千个视频）。这些候选都是用户感兴趣的内容，排序阶段在此基础上进行更精准的计算，能够给每一个视频进行精确打分，进而从成千上万的候选中选出用户最感兴趣的少量高质量内容（十几个视频）。



推荐系统的整体结构如图所示，各个模块的作用如下：

- i) 用户画像：包含用户的人群属性、历史行为、兴趣内容和偏好倾向等多维度的分析，是个性化的基石
- ii) 特征工程：包含了了视频的类别属性，内容分析，人群偏好和统计特征等全方位的描绘和度量，是视频内容和质量分析的基础

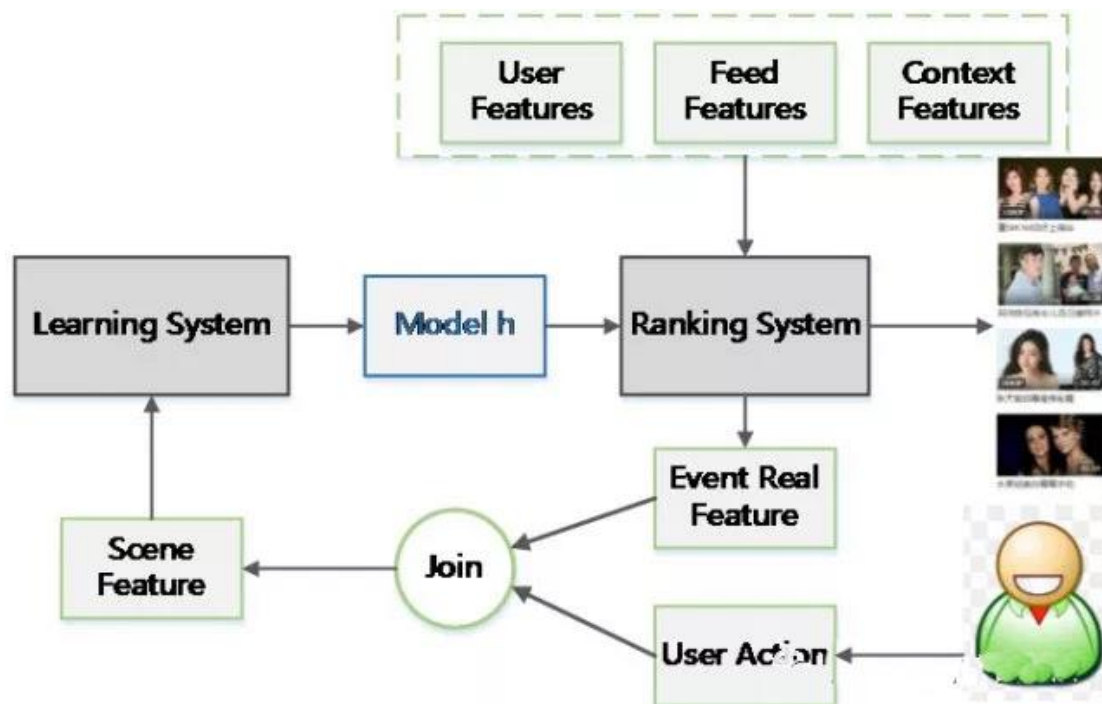
- iii)召回算法：包含了多个通道的召回模型，比如协同过滤，主题模型，内容召回和 SNS 等通道，能够从视频库中选出多样性的偏好内容
- iv)排序模型：对多个召回通道的内容进行同一个打分排序，选出最优的少量结果。

除了这些之外推荐系统还兼顾了推荐结果的多样性，新鲜度，逼格和惊喜度等多个维度，更能够满足用户多样性的需求。

### 三、推荐排序系统架构

在召回阶段，多个通道的召回的内容是不具有可比性的，并且因为数据量太大也难以进行更加精确的偏好和质量评估，因此需要在排序阶段对召回结果进行统一的准确的打分排序。

用户对视频的满意度是有很多维度因子来决定的，这些因子在用户满意度中的重要性也各不相同，甚至各个因子之间还有多层依赖关系，人为制定复杂的规则既难以达到好的效果，又不具有可维护性，这就需要借助机器学习的方法，使用机器学习模型来综合多方面的因子进行排序。



排序系统的架构如图所示，主要由用户行为收集，特征填充，训练样本筛选，模型训练，在线预测排序等多个模块组成。

机器学习的主体流程是比较通用的，设计架构并不需要复杂的理论，更多的是需要对细节，数据流和架构逻辑的仔细推敲。

这个架构设计吸取了以前的经验和教训，在通用机器学习的架构基础上解决了两个问题：

#### A 训练预测的一致性

机器学习模型在训练和预测之间的差异会对模型的准确性产生很大的影响，尤其是模型训练与在线服务时特征不一致，比如用户对推荐结果的反馈会实时影响到用户的偏好特征，在训练的时候用户特征的状态已经发生了变化，模型如果依据这个时候的用户特征就会产生非常大的误差。

我们的解决办法是，将在线服务时的特征保存下来，然后填充到收集的用户行为样本中，这样就保证了训练和预测特征的一致性。

## B 持续迭代

互联网产品持续迭代上线是常态，在架构设计的时候，数据准备，模型训练和在线服务都必须能够对持续迭代有良好的支持。

我们的解决方案是，数据准备和模型训练各阶段解耦，并且策略配置化，这种架构使模型测试变得非常简单，可以快速并行多个迭代测试。

## 四、推荐机器学习排序算法演进

### 4.1 上古时期

我们第一次上线机器学习排序模型时，选用了比较简单的 Logistic Regression，将重点放到架构设计上，尽量保证架构的正确性。除此之外，LR 模型的解释性强，方便 debug，并且通过特征权重可以解释推荐的内容，找到模型的不足之处。

在模型训练之前，我们首先解决的是评测指标和优化目标的问题。

评测指标 (metrics)



线上效果的评测指标需要与长远目标相匹配，比如使用用户的投入程度和活跃度等。在我们的实验中，业界流行的 CTR 并不是一个好的评测指标，它会更偏向于较短的视频，标题党 and 低俗内容。

离线评测指标是按照业务来定制的，以便与在线评测指标匹配，这样在离线阶段就能够淘汰掉无效策略，避免浪费线上流量。

### 优化目标 (objective)

机器学习会按照优化目标求解最优解，如果优化目标有偏差，得到的模型也存在偏差，并且在迭代中模型会不断地向这个偏差的方向学习，偏差会更加严重。

我们的方法是给样本添加权重，并且将样本权重加到 loss function 中，使得优化目标与评测指标尽可能的一致，达到控制模型的目的。

LR 是个线性分类模型，要求输入是线性独立特征。我们使用的稠密的特征（维度在几十到几百之间）往往都是非线性的，并且具有依赖性，因此需要对特征进行转换。

特征转换需要对特征的分布，特征与 label 的关系进行分析，然后采用合适的转换方法。

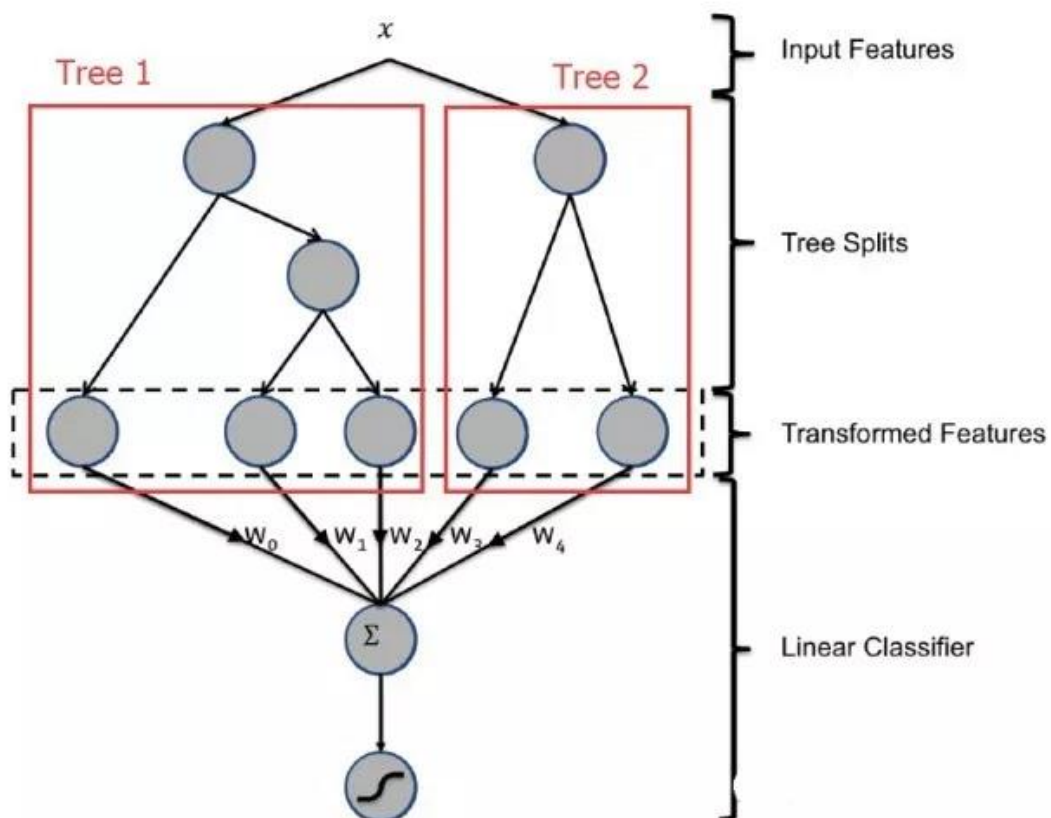
我们用到的有以下几种：Polynomial Transformation, Logarithmic or Exponential Transformation, Interaction Transformation 和 Cumulative Distribution Function 等。

虽然 LR 模型简单，解释性强，不过在特征逐渐增多的情况下，劣势也是显而易见的。

1. 特征都需要人工进行转换为线性特征，十分消耗人力，并且质量不能保证
2. 特征两两作 Interaction 的情况下，模型预测复杂度是。在 100 维稠密特征的情况下，就会有组合出 10000 维的特征。复杂度高，增加特征困难
3. 三个以上的特征进行 Interaction 几乎是不可行的

#### 4.2 中古时期

为了解决 LR 存在的上述问题，我们把模型升级为 Facebook 的 GBDT+LR 模型，模型结构如图所示。



GBDT 是基于 Boosting 思想的 ensemble 模型，由多颗决策树组成，具有以下优点：

- 1) 对输入特征的分布没有要求

- 2)根据熵增益自动进行特征转换、特征组合、特征选择和离散化，得到高维的组合特征，省去了人工转换的过程，并且支持了多个特征的 Interaction
- 3)预测复杂度与特征个数无关

假设特征个数  $n=160$  决策数个数  $k=50$ ，树的深度  $d=6$ ，两代模型的预测复杂度对比如下，升级之后模型复杂度降低到原来的 2.72%

Model Structure	Complexity	Complexity
LR	$n(n-1) / 2 + n$	12880
GBDT+LR	$k * d + k$	350
rate		2.72%

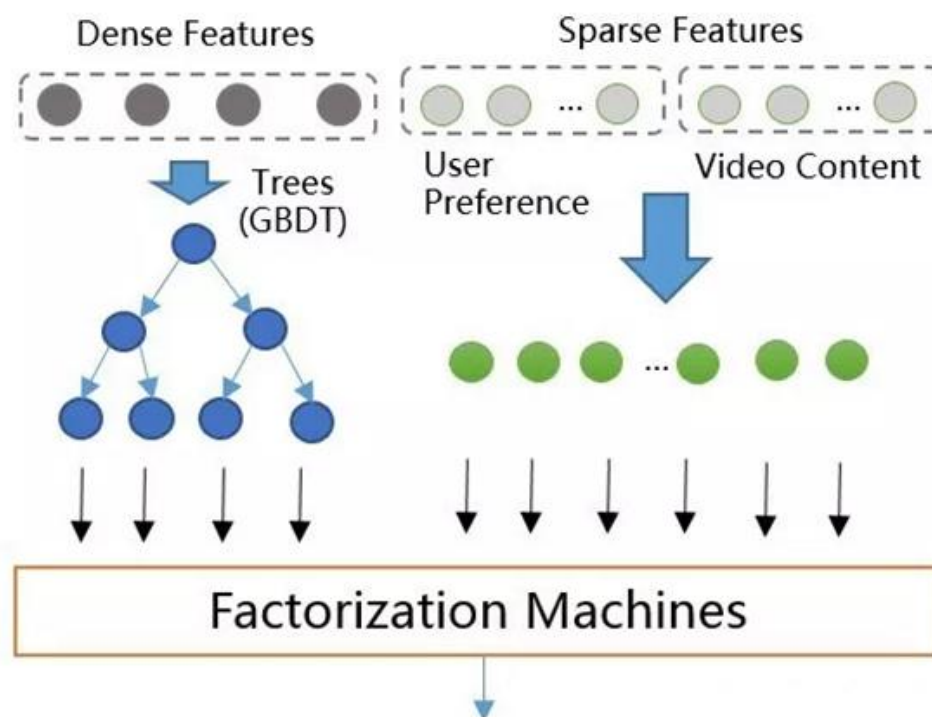
GBDT 与 LR 的 stacking 模型相对于只用 GBDT 会有略微的提升，更大的好处是防止 GBDT 过拟合。升级为 GBDT+LR 后，线上效果提升了约 5%，并且因为省去了对新特征进行人工转换的步骤，增加特征的迭代测试也更容易了。

#### 4.3 近代历史

GBDT+LR 排序模型中输入特征维度为几百维，都是稠密的通用特征。

这种特征的泛化能力良好，但是记忆能力比较差，所以需要增加高维的（百万维以上）内容特征来增强推荐的记忆能力，包括视频 ID，标签，主题等特征。

GBDT 是不支持高维稀疏特征的，如果将高维特征加到 LR 中，一方面需要人工组合高维特征，另一方面模型维度和计算复杂度会是  $O(N^2)$  级别的增长。所以设计了 GBDT+FM 的模型如图所示，采用 Factorization Machines 模型替换 LR。



Factorization Machines (FM) 模型如下所示：

模型公式

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=i+1}^n \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$

具有以下几个优点

- ①前两项为一个线性模型，相当于 LR 模型的作用
- ②第三项为一个二次交叉项，能够自动对特征进行交叉组合
- ③通过增加隐向量，模型训练和预测的计算复杂度降为了  $O(N)$
- ④支持稀疏特征

几个优点，使得 GBDT+FM 具有了良好的稀疏特征支持，FM 使用 GBDT 的叶子结点和稀疏特征（内容特征）作为输入，模型结构示意图如下，GBDT+FM 模型上线后相比 GBDT+LR 在各项指标的效果提升在 4%~6%之间。

典型的 FM 模型中使用 user id 作为用户特征，这会导致模型维度迅速增大，并且只能覆盖部分热门用户，泛化能力比较差。在此我们使用用户的观看历史以及兴趣标签代替 user id，降低了特征维度，并且因为用户兴趣是可以复用的，同时也提高了对应特征的泛化能力。

我们主要尝试使用了 L-BFGS、SGD 和 FTRL (Follow-the-regularized-Leader) 三种优化算法进行求解：

SGD 和 L-BFGS 效果相差不大，L-BFGS 的效果与参数初始化关系紧密

FTRL，较 SGD 有以下优势：

- a)带有 L1 正则，学习的特征更加稀疏
- b)使用累计的梯度，加速收敛

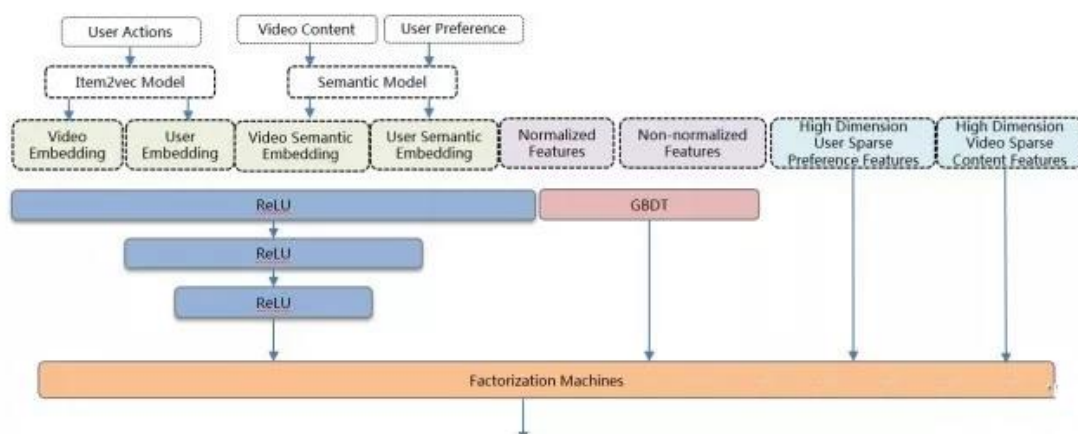
c)根据特征在样本的出现频率确定该特征学习率，保证每个特征有充分的学习

FM 模型中的特征出现的频次相差很大，FTRL 能够保证每个特征都能得到充分的学习，更适合稀疏特征。线上测试表明，在稀疏特征下 FTRL 比 SGD 有 4.5%的效果提升。

#### 4.4 当代模型

GBDT+FM 模型，对 embedding 等具有结构信息的深度特征利用不充分，而深度学习 (Deep Neural Network) 能够对嵌入式 (embedding) 特征和普通稠密特征进行学习，抽取出深层信息，提高模型的准确性，并已经成功应用到众多机器学习领域。因此我们将 DNN 引入到排序模型中，提高排序整体质量。

DNN+GBDT+FM 的 ensemble 模型架构如图所示，FM 层作为模型的最后一层，即融合层，其输入由三部分组成：DNN 的最后一层隐藏层、GBDT 的输出叶子节点、高维稀疏特征。DNN+GBDT+FM 的 ensemble 模型架构介绍如下所示，该模型上线后相对于 GBDT+FM 有 4%的效果提升。



#### DNN 模型

使用全连接网络，共三个隐藏层。

隐藏节点数目分别为 1024, 512 和 256。

预训练好的用户和视频的 Embedding 向量，包含基于用户行为以及基于语义内容的两种 Embedding。

DNN 能从具有良好数学分布的特征中抽取深层信息，比如 embedding 特征，归一化后统计特征等等。

虽然 DNN 并不要求特征必须归一化，不过测试发现有些特征因为 outlier 的波动范围过大，会导致 DNN 效果下降。

## GBDT 模型

单独进行训练，输入包含归一化和未归一化的稠密特征。

能处理未归一化的连续和离散特征。

能根据熵增益自动对输入特征进行离散和组合。

## FM 融合层

FM 模型与 DNN 模型作为同一个网络同时训练。

将 DNN 特征，GBDT 输出和稀疏特征进行融合并交叉。

使用分布式的 TensorFlow 进行训练

使用基于 TensorFlow Serving 的微服务进行在线预测

DNN+GBDT+FM 的 ensemble 模型使用的是 Adam 优化器。Adam 结合了 The Adaptive Gradient Algorithm (AdaGrad) 和 Root Mean Square Propagation (RMSProp) 算法。具有更优的收敛速率，每个变量有各自的下降步长，整体下降步长会根据当前梯度进行调节，能够适应带噪音的数据。实验测试了多种优化器，Adam 的效果是最优的。

#### 4.5 工业界 DNN ranking 现状

- 1、Youtube 于 2016 年推出 DNN 排序算法。
- 2、上海交通大学和 UCL 于 2016 年推出 Product-based Neural Network (PNN) 网络进行用户点击预测。PNN 相当于在 DNN 层做了特征交叉，我们的做法是把特征交叉交给 FM 去做，DNN 专注于深层信息的提取。
- 3、Google 于 2016 年推出 Wide And Deep Model，这个也是我们当前模型的基础，在此基础上使用 FM 替换了 Cross Feature LR，简化了计算复杂度，提高交叉的泛化能力。

阿里今年使用 attention 机制推出了 Deep Interest Network (DIN) 进行商品点击率预估，优化 embedding 向量的准确性，值得借鉴。

#### 五、总结

推荐系统的排序是一个经典的机器学习场景，对于推荐结果影响也十分重大，除了对模型算法的精益求精之外，更需要对业务的特征，工程的架构，数据处理的细节和 pipeline 的流程进行仔细推敲和深入的优化。



Ranking 引入 DNN 仅仅是个开始，后续还需要在模型架构，Embedding 特征，多样性，冷启动和多目标学习中做更多的尝试，提供更准确，更人性化的推荐，优化用户体验。

## 怎么解决推荐系统中的冷启动问题？

解析：

### 1.冷启动问题定义

推荐系统需要根据用户的历史行为和兴趣预测用户未来的行为和兴趣，对于 BAT 这类大公司来说，它们已经积累了大量的用户数据，不发愁。但是对于很多做纯粹推荐系统的网站或者很多在开始阶段就希望有个性化推荐应用的网站来说，如何在对用户一无所知（即没有用户行为数据）的情况下进行最有效的推荐呢？这就衍生了冷启动问题。

### 2.冷启动的分类

冷启动问题主要分为 3 类：

用户冷启动，即如何给新用户做个性化推荐

物品冷启动，即如何将新的物品推荐给可能对它感兴趣的用户

系统冷启动，即如何在一个新开发的网站（没有用户，没有用户行为，只有部分物品信息）上设计个性化推荐系统，从而在网站刚发布时就让用户体会到个性化推荐

### 3.冷启动问题的解决方案

#### 3.1 提供非个性化的推荐

最简单的例子就是提供热门排行榜，可以给用户推荐热门排行榜，等到用户数据收集到一定的时候，再切换为个性化推荐。

关于热门排行榜解决推荐问题的理论测试，可以参考着篇文章 [Performance of recommender algorithms on top-n recommendation tasks](#).

并且 Netflix 的研究也表明新用户冷启动阶段确实是更倾向于热门排行榜的，老用户会更加需要长尾推荐。

### 3.2 利用用户注册信息

用户的注册信息主要分为 3 种：

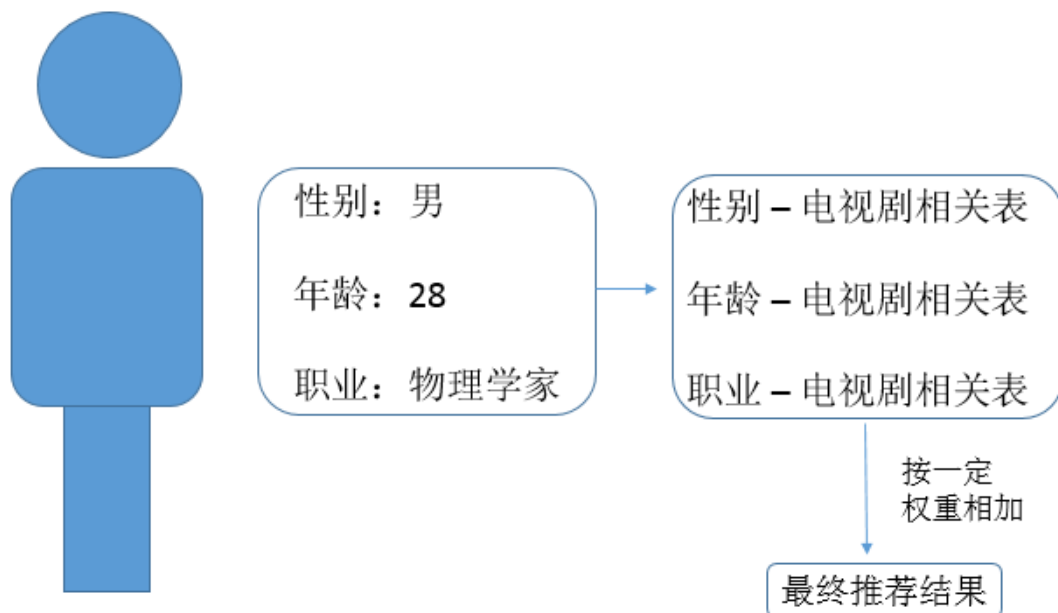
- 1) 人口统计学信息，包括年龄、性别、职业、民族、学历和居住地
- 2) 用户兴趣的描述，部分网站会让用户用文字来描述兴趣
- 3) 从其他网站导入的用户站外行为，比如用户利用社交网站账号登录，就可以在获得用户授权的情况下导入用户在该社交网站的部分行为数据和社交网络数据

这种个性化的粒度很粗，假设性别作为一个粒度来推荐，那么所有刚注册的女性看到的都是同样的结果，但是相对于男女不区分的方式，这种推荐精度已经大大提高了。

推荐流程基本如下：

- ① 获取用户的注册信息
- ② 根据用户的注册信息对用户分类
- ③ 给用户推荐他所属分类中用户喜欢的物品

下面便是一个利用用户的注册信息进行推荐的例子：



### 3.3 选择合适的物品启动用户的兴趣

用户在登录时对一些物品进行反馈，收集用户对这些物品的兴趣信息，然后给用户推荐那些和这些物品相似的物品。

一般来说，能够用来启动用户兴趣的物品需要具有以下特点：

1. 比较热门，如果要想用户对物品进行反馈，前提是用户得知道这是什么东西；
2. 具有代表性和区分性，启动用户兴趣的物品不能是大众化或老少咸宜的，因为这样的物品对用户的兴趣没有区分性；
3. 启动物品集合需要有多多样性，在冷启动时，我们不知道用户的兴趣，而用户兴趣的可能性非常多，为了匹配多样的兴趣，我们需要提供具有很高覆盖率的启动物品集合，这些物品能覆盖几乎所有主流的用户兴趣。

### 3.4 利用物品的内容信息

用来解决物品的冷启动问题，即如何将新加入的物品推荐给对它感兴趣的用戶。物品冷启动问题在新闻网站等时效性很强的网站中非常重要，因为这些网站时时刻刻都有新物品加入，而且每个物品必须能够再第一时间展现给用户，否则经过一段时间后，物品的价值就大大降低了。

针对协同过滤的两种推荐算法——userCF 算法、itemCF 算法来分别了解一下物品冷启动的问题。

### userCF 算法

针对推荐列表并不是给用户展示内容的唯一列表（大多网站都是这样的）的网站

当新物品加入时，总会有用户通过某些途径看到，那么当一个用户对其产生反馈后，和他历史兴趣相似的用户的推荐列表中就有可能出现该物品，从而更多的人对该物品做出反馈，导致更多的人的推荐列表中出现该物品。因此，该物品就能不断扩散开来，从而逐步展示到对它感兴趣用户的推荐列表中

针对推荐列表是用户获取信息的主要途径（例如豆瓣网络电台）的网站

userCF 算法就需要解决第一推动力的问题，即第一个用户从哪儿发现新物品。最简单的方法是将新的物品随机展示给用户，但是太不个性化。因此可以考虑利用物品的内容信息，将新物品先投放给曾经喜欢过和它内容相似的其他物品的用户

### itemCF 算法

对 itemCF 算法来说，物品冷启动就是很严重的问题了。因为该算法的基础是通过用户对物品产生的行为来计算物品之间的相似度，当新物品还未展示给用户时，用户就无法产生

行为。为此，只能利用物品的内容信息计算物品的相关程度。基本思路就是将物品转换成关键词向量，通过计算向量之间的相似度（例如计算余弦相似度），得到物品的相关程度。

下表列出了常见物品的内容信息：

表3-4 常见物品的内容信息	
图书	标题、作者、出版社、出版年代、丛书名、目录、正文
论文	标题、作者、作者单位、关键字、分类、摘要、正文
电影	标题、导演、演员、编剧、类别、剧情简介、发行公司
新闻	标题、正文、来源、作者
微博	作者、内容、评论

3.5 采用专家标注

很多系统在建立的时候，既没有用户的行为数据，也没有充足的物品内容信息来计算物品相似度。这种情况下，很多系统都利用专家进行标注。

代表系统：个性化网络电台 Pandora、电影推荐网站 Jinni

以 Pandora 电台为例，Pandora 雇用了一批音乐人对几万名歌手的歌曲进行各个维度的标注，最终选定了 400 多个特征。每首歌都可以标识为一个 400 维的向量，然后通过常见的向量相似度算法计算出歌曲的相似度。

以上均为项亮一书《推荐系统实践》中描述到的方法，下面再介绍两种方法。

3.6 利用用户在其他地方已经沉淀的数据进行冷启动

以 QQ 音乐举例：

QQ 音乐的猜你喜欢电台想要去猜测第一次使用 QQ 音乐的用户的口味偏好，一大优势是可以利用其它腾讯平台的数据，比如在 QQ 空间关注了谁，在腾讯微博关注了谁，更进一步，比如在腾讯视频刚刚看了一部动漫，那么如果 QQ 音乐推荐了这部动漫里的歌曲，用户会觉得很人性化。这就是利用用户在其它平台已有的数据。

再比如今日头条：

它是在用户通过新浪微博等社交网站登录之后，获取用户的关注列表，并且爬取用户最近参与互动的 feed（转发/评论等），对其进行语义分析，从而获取用户的偏好。

所以这种方法的前提是，引导用户通过社交网络账号登录，这样一方面可以降低注册成本提高转化率；另一方面可以获取用户的社交网络信息，解决冷启动问题。

### 3.7 利用用户的手机等兴趣偏好进行冷启动

Android 手机开放的比较高，所以在安装自己的 app 时，就可以顺路了解下手机上还安装了什么其他的 app。比如一个用户安装了美丽说、蘑菇街、辣妈帮、大姨妈等应用，就可以判定这是女性了，更进一步还可以判定是备孕还是少女。

目前读取用户安装的应用这部分功能除了 app 应用商店之外，一些新闻类、视频类的应用也在做，对于解决冷启动问题有很好的帮助。

## 请聊聊你所了解的推荐系统算法

解析：

推荐系统算法如果根据推荐的依据进行划分，有如下三大类算法：

一、Content-based recommenders: 推荐和用户曾经喜欢的商品相似的商品。主要是基于商品属性信息和用户画像信息的对比。核心问题是如何刻画商品属性和用户画像以及效用的度量。方法包括:

1.1 Heuristic-based method: 对于特征维度的构建, 例如基于关键字提取的方法, 使用 TF-IDF 等指标提取关键字作为特征。对于效用的度量, 例如使用启发式 cosine 相似性指标, 衡量商品特征和用户画像的相似性, 似性越高, 效用越大。

1.2 Machine learning-based method: 对于特征维度的构建, 使用机器学习算法来构建用户和商品的特征维度。例如建模商品属于某个类别的概率, 得到商品的刻画属性。对于效用的度量, 直接使用机器学习算法拟合效用函数。

二、Collaborative recommenders: 推荐和用户有相似品味和偏好的用户喜欢过的商品。主要是基于用户和商品历史交互行为信息, 包括显示的和隐式的。协同过滤方法进一步细分为:

2.1 Memory-based CF: 基于内存的协同过滤方法。直接对 User-Item 矩阵进行研究。通过启发式的方法来进行推荐。核心要素包括相似性度量和推荐策略。相似性度量包括 Pearson 或 Cosine 等; 而最简单的推荐方法是基于大多数的推荐策略。

User-based CF: 推荐给特定用户列表中还没有发生过行为、而在相似用户列表中产生过行为的高频商品。

Item-based CF: 推荐给特定用户列表中还没有发生过行为、并且和已经发生过行为的商品相似的商品。

2.2 Model-based CF: 基于模型的协同过滤方法。主要是运用机器学习的思想来进行推荐。主要包括：

基于流形学习的矩阵降维/分解算法: SVD、FunkSVD、BiasSVD、SVD++、NMF 等。

基于表示学习的深度学习算法: MLP、CNN、AutoEncoder、RNN 等。

基于图/网络模型的算法: MDP-based CF、Bayesian Belief nets CF、CTR(协同主题回归, 将概率矩阵分解和主题模型结合应用于推荐系统)等。

其它: 包括基于聚类的 CF、稀疏因子分析 CF、隐语义分析 CF 等等。

2.3 Hybrid CF: 结合多种方式的 CF 算法。如 Content-based CF、Content-boosted CF 或者结合 Memory-based 和 Model-based CF 混合方法。

TABLE 2: Overview of collaborative filtering techniques.

CF categories	Representative techniques	Main advantages	Main shortcomings
Memory-based CF	*Neighbor-based CF (item-based/user-based CF algorithms with Pearson/vector cosine correlation) *Item-based/user-based top-N recommendations	*easy implementation *new data can be added easily and incrementally *need not consider the content of the items being recommended *scale well with co-rated items	*are dependent on human ratings *performance decrease when data are sparse *cannot recommend for new users and items *have limited scalability for large datasets
Model-based CF	*Bayesian belief nets CF *clustering CF *MDP-based CF *latent semantic CF *sparse factor analysis *CF using dimensionality reduction techniques, for example, SVD, PCA	*better address the sparsity, scalability and other problems *improve prediction performance *give an intuitive rationale for recommendations	*expensive model-building *have trade-off between prediction performance and scalability *lose useful information for dimensionality reduction techniques
Hybrid recommenders	*content-based CF recommender, for example, <i>Fab</i> *content-boosted CF *hybrid CF combining memory-based and model-based CF algorithms, for example, Personality Diagnosis	*overcome limitations of CF and content-based or other recommenders *improve prediction performance *overcome CF problems such as sparsity and gray sheep	*have increased complexity and expense for implementation *need external information that usually not available



### 三、Hybrid approaches: 混合方法。综合集成上述两种方法。

当前推荐算法主要是基于内容(CB)、协同过滤(CF)、混合算法。基于内容的推荐依靠用户 profile 和 item 的描述做推荐。CF 基于过去的的表现和行为推荐。由于种种原因，收集过去的行为比收集用户画像要容易，但 CF 又有他的局限性，当打分 (rating) 很稀疏时，预测精度会下降很厉害，同时，新产品的冷启动也是 CF 的问题。因此，近年来，混合方法应用比较广。

Recommendation Approach	Recommendation Technique	
	Heuristic-based	Model-based
Content-based	<p>Commonly used techniques:</p> <ul style="list-style-type: none"> <li>• TF-IDF (information retrieval)</li> <li>• Clustering</li> </ul> <p>Representative research examples:</p> <ul style="list-style-type: none"> <li>• Lang 1995</li> <li>• Balabanovic &amp; Shoham 1997</li> <li>• Pazzani &amp; Billsus 1997</li> </ul>	<p>Commonly used techniques:</p> <ul style="list-style-type: none"> <li>• Bayesian classifiers</li> <li>• Clustering</li> <li>• Decision trees</li> <li>• Artificial neural networks</li> </ul> <p>Representative research examples:</p> <ul style="list-style-type: none"> <li>• Pazzani &amp; Billsus 1997</li> <li>• Mooney et al. 1998</li> <li>• Mooney &amp; Roy 1999</li> <li>• Billsus &amp; Pazzani 1999, 2000</li> <li>• Zhang et al. 2002</li> </ul>
Collaborative	<p>Commonly used techniques:</p> <ul style="list-style-type: none"> <li>• Nearest neighbor (cosine, correlation)</li> <li>• Clustering</li> <li>• Graph theory</li> </ul> <p>Representative research examples:</p> <ul style="list-style-type: none"> <li>• Resnick et al. 1994</li> <li>• Hill et al. 1995</li> <li>• Shardanand &amp; Maes 1995</li> <li>• Breese et al. 1998</li> <li>• Nakamura &amp; Abe 1998</li> <li>• Aggarwal et al. 1999</li> <li>• Delgado &amp; Ishii 1999</li> <li>• Pennock &amp; Horwitz 1999</li> <li>• Sarwar et al. 2001</li> </ul>	<p>Commonly used techniques:</p> <ul style="list-style-type: none"> <li>• Bayesian networks</li> <li>• Clustering</li> <li>• Artificial neural networks</li> <li>• Linear regression</li> <li>• Probabilistic models</li> </ul> <p>Representative research examples:</p> <ul style="list-style-type: none"> <li>• Billsus &amp; Pazzani 1998</li> <li>• Breese et al. 1998</li> <li>• Ungar &amp; Foster 1998</li> <li>• Chien &amp; George 1999</li> <li>• Getoor &amp; Sahami 1999</li> <li>• Pennock &amp; Horwitz 1999</li> <li>• Goldberg et al. 2001</li> <li>• Kumar et al. 2001</li> <li>• Pavlov &amp; Pennock 2002</li> <li>• Shani et al. 2002</li> <li>• Yu et al. 2002, 2004</li> <li>• Hofmann 2003, 2004</li> <li>• Marlin 2003</li> <li>• Si &amp; Jin 2003</li> </ul>
Hybrid	<p>Combining content-based and collaborative components using:</p> <ul style="list-style-type: none"> <li>• Linear combination of predicted ratings</li> <li>• Various voting schemes</li> <li>• Incorporating one component as a part of the heuristic for the other</li> </ul> <p>Representative research examples:</p> <ul style="list-style-type: none"> <li>• Balabanovic &amp; Shoham 1997</li> <li>• Claypool et al. 1999</li> <li>• Good et al. 1999</li> <li>• Pazzani 1999</li> <li>• Billsus &amp; Pazzani 2000</li> <li>• Tran &amp; Cohen 2000</li> <li>• Melville et al. 2002</li> </ul>	<p>Combining content-based and collaborative components by:</p> <ul style="list-style-type: none"> <li>• Incorporating one component as a part of the model for the other</li> <li>• Building one unifying model</li> </ul> <p>Representative research examples:</p> <ul style="list-style-type: none"> <li>• Basu et al. 1998</li> <li>• Condliff et al. 1999</li> <li>• Soboroff &amp; Nicholas 1999</li> <li>• Ansari et al. 2000</li> <li>• Popescul et al. 2001</li> <li>• Schein et al. 2002</li> </ul>

# 了解隐语义模型在推荐系统中的应用的么？

解析：

本篇记录在学习隐语义模型的一些总结，隐语义建模；隐语义模型的核心思想；隐语义模型在推荐系统中的应用；隐语义模型与推荐系统的关系；工程中常用的矩阵分解方法

## 前言

推荐系统中一个重要的分支，隐语义建模。隐语义模型 LFM: Latent Factor Model, 其核心思想就是通过隐含特征联系用户兴趣和物品。

过程分为三个部分，将物品映射到隐含分类，确定用户对隐含分类的兴趣，然后选择用户感兴趣的分类中的物品推荐给用户。它是基于用户行为统计的自动聚类。

隐语义模型在 Top-N 推荐中的应用十分广泛。常用的隐语义模型，LSA(Latent Semantic Analysis), LDA(Latent Dirichlet Allocation), 主题模型(Topic Model), 矩阵分解(Matrix Factorization)等等。

这些模型是本质上是相通的，目的就是找出潜在的主题或分类。这些技术一开始实在文本挖掘中提出来的。从 netflix 推荐大赛之后，这些技术被用在推荐领域，并且得到明显的效果。比如，在推荐系统中，能够用这些模型将用户的行为（用户的偏好）对 item 进行自动聚类，也就是把 item 划分到不同类别/主题，这些主题/类别可以理解为用户的兴趣。

凭借这种思想，著名的推荐领域大神 Yehuda Koren 更是凭借矩阵分解模型勇夺 Netflix Prize 推荐比赛冠军，以矩阵分解为基础，Yehuda Koren 在数据挖掘和机器学习相关的国际顶级会议(SIGIR,SIGKDD,RecSys 等)发表了很多文章，将矩阵分解模型的优势发挥得淋漓尽致。实验结果表明，在个性化推荐中使用矩阵分解模型要明显优于传统的基于邻域的协同过滤(又称基于记忆的协同过滤)方法，如 UserCF、ItemCF 等，这也使得矩阵分解成为了之前个性化推荐研究领域中的主流模型。

## 基本算法

### 与 UserCF 和 ItemCF 对比

首先通过一个例子来理解一下这个模型，比如说有两个用户 A 和 B，目前有用户的阅读列表，用户 A 的兴趣涉及侦探小说，科普图书以及一些计算机技术书，而用户 B 的兴趣比较集中在数学和机器学习方面。

那么如何给 A 和 B 推荐图书呢？

对于 UserCF，首先需要找到和他们看了同样书的其他用户(兴趣相似的用户)，然后在给他们推荐那些用户喜欢的其他书。

对于 ItemCF,需要给他们推荐和他们已经看的书相似的书，比如用户 B 看了很多数据挖掘方面的书，那么可以给他推荐机器学习或者模式识别方面的书。

还有一种方法，可以对书和物品的兴趣进行分类。对于某个用户，首先得到他的兴趣分类，然后从分类中挑选他可能喜欢的物品。

基于兴趣分类的方法大概需要解决的问题：

如何给物品进行分类？

如何确定用户对哪些类的物品感兴趣，以及感兴趣的程度？

对于一个给定的类，选择哪些属于这个类的物品推荐给用户，以及如何确定这些物品在一个类中的权重？

对于第一个问题最简单的解决方案是找编辑给物品打标签，或者采用豆瓣的方式，让用户自己打标签。这样做的确定是，这种打标签不自动发现类比，还有一点就是编辑的意见并不能达标各种用户的意见，比如一些书目的划分是模棱两可的。这就涉及到分类的粒度不易控制的问题。另外，编辑很难决定一个物品在某一个分类中的权重，比如编辑可以很容易的决定《数据挖掘导论》属于挖掘类的图书，但是这本书在这类书中的定位是什么样的，编辑就很难给出一个准确的数字来标示。

为了解决上面的问题，研究人员提出：为什么我们不从数据出发，自动地找到那些类，然后进行个性化推荐，隐语义分析技术因为采取基于用户行为统计的自动聚类，较好地解决了上面的问题。隐语义分析技术从诞生到今天产生了很多著名的模型和方法，其中和推荐技术相关的有 pLSA, LDA, 隐含类别模型 (latent class model), 隐含主题模型 (latent topic model), 矩阵分解 (matrix factorization)。

## 隐语义模型的应用

### 向用户推荐物品

在推荐系统中，可以通过隐含语义模型将用户 (user) 和物品 (item) 自动分类，这些类别是自动生成的，这些类比也可以叫做“隐含的分类”，或许我们看不懂分类后的结果。但是采用这种技术后，每个用户或者物品会被分到多个类别中，属于某个类别的权重会被计算出来。

假设现在有一个大小为  $m \times n$  的评分矩阵  $V$ ，包含了  $m$  个用户对  $n$  个物品的评分，评分从 0 到 5，值越大代表越喜欢，0 代表没有打分。设定共有  $r$  个隐含的分类。通过一些方法，将  $V$  展开为两个相乘的矩阵：

$$V = W * H$$

其中， $W$  的大小为  $m \times r$ ， $H$  的大小为  $r \times n$ 。在隐语义模型中， $W(i,j)$  被解释为用户  $i$  属于类别  $j$  的权重， $H(a,b)$  被解释为物品  $b$  属于类别  $a$  的的权重。

如果用户  $u$  对物品  $i$  没有评分，可以将这个评分  $r(u,i)$  预测为：

$$r(u,i) = \text{sum}(W(i, :) .* H(:, i)) // \text{向量点乘}$$

据此可以构建一个推荐系统。

## 文本分类

隐语义模型的另一个常见的应用领域是文本分类，这个类似于上面的推荐系统。在文本分类领域，将文档和词用一个矩阵表示，如常见的词袋模型，文档-词矩阵。我们将数据集中的一堆文本构造成文档-词矩阵  $V$ ，如果共有  $m$  个文本， $n$  个单词，那么  $V$  的大小为  $m \times n$ ， $V(i,j)$  表示文档  $i$  中出现单词  $j$  的次数。

设定共有  $r$  个隐含的分类。通过一些方法，将  $V$  展开为两个相乘的矩阵：

$$V = W \times H$$

其中， $W$  的大小为  $m \times r$ ， $H$  的大小为  $r \times n$ 。在隐语义模型中， $W(i,j)$  被解释为文档  $i$  属于类别  $j$  的权重， $H(a,b)$  被解释为单词  $b$  属于类别  $a$  的权重。

对于一个文档，其权重最大的类别被看作是该文档的类别。由于设定共有  $r$  个隐含的分类，分类结果也是  $r$  个份分类。

## 采用矩阵分解技术发现两种实体间潜在的特征

使用矩阵分解来预测评分的思想来源于，我们可以通过矩阵分解来发现一些用户打分的潜在特征，比如两个人都喜欢某一演员，那他们就倾向于给该演员演的电影打高分；或者两个都喜欢动作片，假如我们能够发现这些特征，我们就能够预测特定用户对特定电影的打分。

为了发现不同的特征，我们假设特征的数量少于用户和电影的数量，矩阵分解算法的数学理论基础是矩阵的行列变换。在《线性代数》中，我们知道矩阵  $A$  进行行变换相当于  $A$  左乘以一个矩阵，矩阵  $A$  进行列变换等价于矩阵  $A$  右乘以一个矩阵，因此矩阵  $A$  可以进行矩阵分解，一方面可以起到降低数据维度的作用，另一方面也可以找到潜在的特征。

矩阵分解可以带来非常好的结果，而且可以充分地考虑各种因素的影响，有非常好的扩展性，因为要考虑各种因素的综合作用，往往需要构造 cost function 来将矩阵分解转换为优化问题。根据要考虑的因素为优化问题其他限制条件，然后通过迭代的方法进行矩阵分解，原来评分矩阵中的 missing value 可以通过分解后得到的矩阵求得。

## pureSVD

其实，矩阵分解的核心是将一个非常稀疏的评分矩阵分解为两个矩阵，一个表示用户 user 的特性，一个表示 item 的特性，将两个矩阵各自取一行一列向量做内积就可以得到对应评分。那么如何将一个矩阵分解为两个矩阵就是唯一的问题啦。

这里可以借用在线性代数和数值分析中学到的各种矩阵分解方法，如 QR, Jordan, 三角分解, SVD。这里说明下 svd 方法的大概意思：将一个任意实矩阵分解为桑格矩阵  $U, S, V$ ，其中  $U, V$  是两个正交矩阵，称为左右奇异矩阵， $S$  是个对称阵，称为奇异值矩阵。

## Latent Factor Model

这是真正的矩阵分解算法，经过实践检验，具有非常高的准确性和易扩展性。正如上面提到的，实现推荐系统结果的目标是将那个稀疏的评分矩阵分解成为两个矩阵，一个表示 user 的 feature，一个表示 item 的 feature，然后做内积得到预测。

当然要实现满足一定约束条件的矩阵分解，不像上面的 pureSVD 那么容易，需要构造一个优化问题，用一些复杂的算法求最优化问题。而这些最优化问题往往是 NP 问题，只有局部最优解。首先构造一个优化目标函数，考虑到最后的评价指标是预测分值与实际分值之间的误差的平方 (RMSE)，所以构造的目标函数也是误差的平方的函数。为什么这样的算法可以得到更优的结果呢？因为算法可以很容易地扩展很多的 feature 进来，更加出色地考虑了多种影响推荐效果的实实在在的因素。

Biases 用户评分的偏见：

因为有的用户总是会打出比别人高的分，或者说有的用户他的评价尺度比较宽松，同样有的 item 总是被打高分，所以在真正实践中，构造目标函数需要增加几项：所有评分的平均值  $\mu$ , user 的偏见分数，item 的偏见分数。

implicit feedback 隐式反馈数据

用户在使用 web 应用的时候，会产生大量的行为，充分挖掘这部分的价值，将会很好的提升推荐的效果，利用用户的隐式反



馈，相当于充分的利用评分矩阵中的 missing value 的价值，用户在页面的停留时间，检索，浏览，点击等等各种行为都可以建模到目标函数中。在看到的论文中，这里，可以将简单的用一个 boolean 来描述一种行为有还是没有。补过在使用的使用需要进行归一化处理。

### User-associated attributes

基于用户的社会化属性进行推荐也是一种很基本的推荐，当然也可以考虑到目标函数中。

### Temporal dynamics

用户的兴趣包括长期和短期，动态地考虑一段时间内用户的兴趣是很有必要的。

### Confidence level

因为在处理上述的因素的时候，很多都是比较主观的，所以需要给每个因素添加一个置信权重，以平衡整个结果。

通过构造出这个目标函数，然后添加相应的约束条件，接下来的问题就是求解这个优化问题，通常比较好的方法是 随机梯度下降算法 (Stochastic gradient descent)

这种方法在学术上主流的方法，参考 [http://research.yahoo.com/Yehuda\\_Koren](http://research.yahoo.com/Yehuda_Koren) 以及上海交大在 kddcup 的论文集开源系统 [http://apex.sjtu.edu.cn/apex\\_wiki/svdfeature](http://apex.sjtu.edu.cn/apex_wiki/svdfeature)

## 非负矩阵分解

基本原理：NMF，非负矩阵分解，它的目标很明确，就是将大矩阵分解为两个小矩阵，使得这两个小矩阵相乘后能够还原到大矩阵。而非负表示分解的矩阵都不包含负值。从应用的角度来说，矩阵分解能够用于发现两种实体间的潜在特征，一个最常见的应用就是协同过滤中的预测打分值，而从协同过滤的这个角度来说，非负也很容易理解：打分都是正的，不会出现负值。

考虑到 svd 或者 latent factor model 会得到负值的情况，所以非负矩阵分解的物理意义比较明确。同样的道理，NMF 也是将评分矩阵的转置矩阵分解成两个矩阵。不同的是这两个矩阵有着和上面的矩阵不同的物理意义。其中一个是基于矩阵  $W$ ，另一个是投影矩阵  $H$ ，即  $R(n \times m) = W(n \times r)H(r \times m)$ 。

$W$ ：每一列包含一个基向量，这组基向量构成一个  $r$  维的空间。

$H$ ：每一列则近似为原始数据对应的列向量在该  $r$  维空间的投影。

做这个分解的方法也是构造一个目标函数和一些约束条件，然后用梯度下降的算法计算得到一个局部最优解。这种方法的大概思路

- 1 将评分矩阵转置然后分解称为两个矩阵  $W$  和  $H$ 。
- 2 根据基矩阵  $W$ ，可以计算得到目标用户评分向量  $a$  对基矩阵  $W$  的投影向量  $h$ 。
- 3 计算投影向量  $h$  与投影矩阵  $H$  各行之间的欧氏距离，将其中距离最小的前  $k$  个用户组成目标用户  $a$  的最近邻集合。
- 4 然后用皮尔逊讲最近邻集合中的数据进行加权计算，然后拍下进行推荐。

这种方法的思路和上面的两种还是不同相同的，主要是在计算目标用户的最近邻集合，主要思想还是 knn,只是在中间用了矩阵分解的方法降低了计算的时间复杂度。

参考：[blog.csdn.net/zyvscc/article/details/7551842](http://blog.csdn.net/zyvscc/article/details/7551842)

[www.cnblogs.com/LeftNotEasy/archive/2011/01/19/svd-and-applications.html](http://www.cnblogs.com/LeftNotEasy/archive/2011/01/19/svd-and-applications.html)

## 如何通俗理解奇异值分解

解析：

特征值和奇异值在大部分人的印象中，往往是停留在纯粹的数学计算中。而且线性代数或者矩阵论里面，也很少讲任何跟特征值与奇异值有关的应用背景。

奇异值分解是一个有着很明显的物理意义的一种方法，它可以将一个比较复杂的矩阵用更小更简单的几个子矩阵的相乘来表示，这些小矩阵描述的是矩阵的重要的特性。就像是描述一个人一样，给别人描述说这个人长得浓眉大眼，方脸，络腮胡，而且带个黑框的眼镜，这样寥寥的几个特征，就让别人脑海里面就有一个较为清楚的认识，实际上，人脸上的特征是有着无数种的，之所以能这么描述，是因为人天生就有着非常好的抽取重要特征的能力，让机器学会抽取重要的特征，SVD 是一个重要的方法。

在机器学习领域，有相当多的应用与奇异值都可以扯上关系，比如做 feature reduction 的 PCA，做数据压缩（以图像压缩为代表）的算法，还有做搜索引擎语义层次检索的 LSI (Latent Semantic Indexing)

## 一、特征值与奇异值

特征值分解和奇异值分解在机器学习领域都是属于满地可见的方法。两者有着很紧密的关系，接下来会谈到特征值分解和奇异值分解的目的都是一样，就是提取出一个矩阵最重要的特征。先谈特征值分解。

### 1.1 特征值

如果说一个向量  $v$  是方阵  $A$  的特征向量，将一定可以表示成下面的形式：

$$Av = \lambda v$$

这时候  $\lambda$  就被称为特征向量  $v$  对应的特征值，一个矩阵的一组特征向量是一组正交向量。特征值分解是将一个矩阵分解成下面的形式：

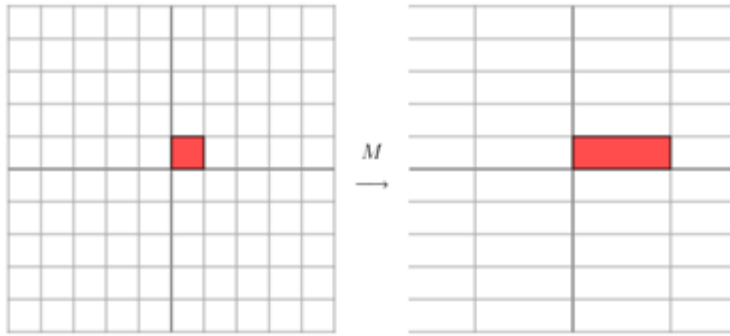
$$A = Q\Sigma Q^{-1}$$

其中  $Q$  是这个矩阵  $A$  的特征向量组成的矩阵， $\Sigma$  是一个对角阵，每一个对角线上的元素就是一个特征值。我这里引用了一些参考文献中的内容来说明一下。

首先，要明确的是，一个矩阵其实就是一个线性变换，因为一个矩阵乘以一个向量后得到的向量，其实就相当于将这个向量进行了线性变换。比如说下面的一个矩阵：

$$M = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$$

它其实对应的线性变换是下面的形式：



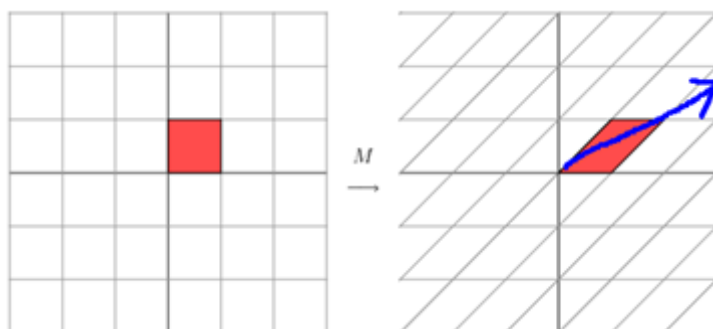
因为这个矩阵  $M$  乘以一个向量  $(x, y)$  的结果是：

$$\begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3x \\ y \end{bmatrix}$$

上面的矩阵是对称的，所以这个变换是一个对  $x$ ,  $y$  轴的方向一个拉伸变换（每一个对角线上的元素将会对一个维度进行拉伸变换，当值  $> 1$  时，是拉长，当值  $< 1$  时时缩短），  
当矩阵不是对称的时候，假如说矩阵是下面的样子：

$$M = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

它所描述的变换是下面的样子：

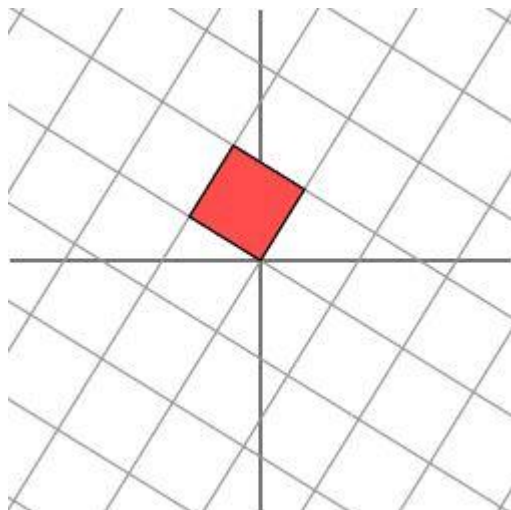


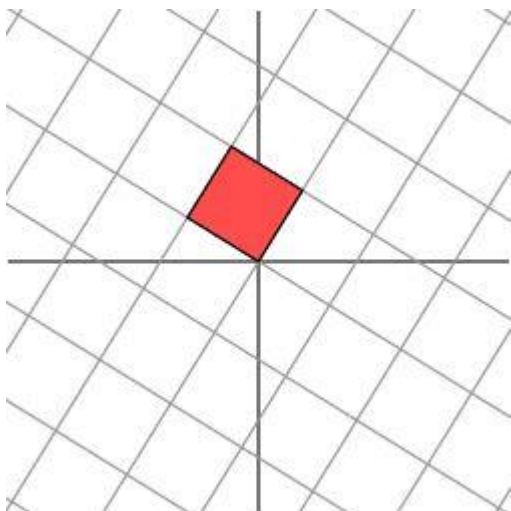
这其实是在平面上对一个轴进行的拉伸变换（如蓝色的箭头所示），在图中，蓝色的箭头是一个最主要的变化方向（变化方向可能有不止一个），如果我们想要描述好一个变换，那我们就描述好这个变换主要的变化方向就好了。反过头来看看之前特征值分解的式子，分解得到的 $\Sigma$ 矩阵是一个对角阵，里面的特征值是由大到小排列的，这些特征值所对应的特征向量就是描述这个矩阵变化方向（从主要的变化到次要的变化排列）。

考虑更一般的非对称矩阵

很遗憾，此时我们再也找不到一组网格，使得矩阵作用在该网格上之后只有拉伸变换（找不到背后的数学原因是对于一般非对称矩阵无法保证在实数域上可对角化，不明白也不要介意）。

我们退而求其次，找一组网格，使得矩阵作用在该网格上之后允许有拉伸变换和旋转变换，但要保证变换后的网格依旧互相垂直，这是可以做到的，如下图所示。





简言之，当矩阵是高维的情况下，那么这个矩阵就是高维空间下的一个线性变换，这个变换也同样有很多的变换方向，我们通过特征值分解得到的前  $N$  个特征向量，那么就对应了这个矩阵最主要的  $N$  个变化方向。我们利用这前  $N$  个变化方向，就可以近似这个矩阵（变换）。

也就是之前说的：提取这个矩阵最重要的特征。总结一下，特征值分解可以得到特征值与特征向量，特征值表示的是这个特征到底有多重要，而特征向量表示这个特征是什么，可以将每一个特征向量理解为一个线性的子空间，我们可以利用这些线性的子空间干很多的事情。不过，特征值分解也有很多的局限，比如说变换的矩阵必须是方阵。

下面我们就可以自然过渡到奇异值分解的引入。

## 1.2 奇异值

下面谈谈奇异值分解。特征值分解是一个提取矩阵特征很不错的方法，但是它只是对方阵而言的，在现实的世界中，我们看到的大部分矩阵都不是方阵，比如说有  $N$  个学生，

每个学生有 M 科成绩，这样形成的一个  $N * M$  的矩阵就不可能是方阵，我们怎样才能描述这样普通的矩阵呢的重要特征呢？

奇异值分解可以用来干这个事情，奇异值分解是一个能适用于任意的矩阵的一种分解的方法：

$$A = U \Sigma V^T$$

假设 A 是一个  $N * M$  的矩阵，那么得到的 U 是一个  $N * N$  的方阵（里面的向量是正交的，U 里面的向量称为左奇异向量）， $\Sigma$  是一个  $N * M$  的矩阵（除了对角线的元素都是 0，对角线上的元素称为奇异值）， $V^T$ （V 的转置）是一个  $N * N$  的矩阵，里面的向量也是正交的，V 里面的向量称为右奇异向量），从图片来反映几个相乘的矩阵的大小可得下面的图片

$$\begin{matrix} \text{blue} & & \text{green} & & \text{blue} & & \text{orange} \\ A & = & U & \times & \Sigma & \times & V^T \\ m \times n & & m \times m & & m \times n & & n \times n \end{matrix}$$

那么奇异值和特征值是怎么对应起来的呢？首先，我们将一个矩阵 A 的转置 \* A，将会得到一个方阵，我们用这个方阵求特征值可以得到：

$$(A^T A)v_i = \lambda_i v_i$$

这里得到的  $v_i$ ，就是我们上面的右奇异向量。此外我们还可以得到：



$$\sigma_i = \sqrt{\lambda_i}$$

$$u_i = \frac{1}{\sigma_i} A v_i$$

这里的 $\sigma$ 就是上面说的奇异值， $u$ 就是上面说的左奇异向量。奇异值 $\sigma$ 跟特征值类似，在矩阵 $\Sigma$ 中也是从大到小排列，而且 $\sigma$ 的减少特别的快，在很多情况下，前 10%甚至 1%的奇异值的和就占了全部的奇异值之和的 99%以上了。也就是说，我们也可以用前  $r$  大的奇异值来近似描述矩阵，这里定义一下部分奇异值分解：

$$A_{m \times n} \approx U_{m \times r} \Sigma_{r \times r} V^T_{r \times n}$$

$r$  是一个远小于  $m$ 、 $n$  的数，这样矩阵的乘法看起来像是下面的样子：

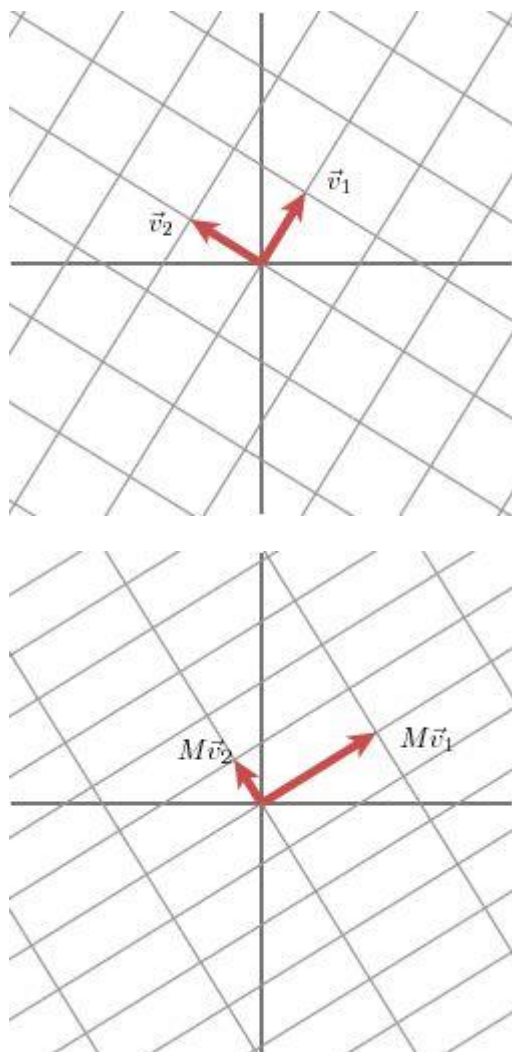
$$\begin{array}{c}
 \text{Blue square } A \\
 m \times n
 \end{array}
 =
 \begin{array}{c}
 \text{Green vertical rectangle } U \\
 m \times r
 \end{array}
 \times
 \begin{array}{c}
 \text{Blue square } \Sigma \\
 r \times r
 \end{array}
 \times
 \begin{array}{c}
 \text{Orange horizontal rectangle } V^T \\
 r \times n
 \end{array}$$

右边的三个矩阵相乘的结果将会是一个接近于  $A$  的矩阵，在这儿， $r$  越接近于  $n$ ，则相乘的结果越接近于  $A$ 。而这三个矩阵的面积之和（在存储观点来说，矩阵面积越小，存储量就越小）要远远小于原始的矩阵  $A$ ，我们如果想要压缩空间来表示原矩阵  $A$ ，我们存下这里的三个矩阵： $U$ 、 $\Sigma$ 、 $V$  就好了。

说句大白话，称作「奇异值」可能无法顾名思义迅速理解其本质，那咱们换个说法，称作「主特征值」，你可能就迅速了然了。

而奇异值分解的几何含义为：对于任何一个矩阵，我们要找到一组两两正交单位向量序列，使得矩阵作用在此向量序列上后得到新的向量序列保持两两正交。

继续拿 1.1 节的例子进一步阐述，奇异值的几何含义为：这组变换后的新的向量序列的长度。



当矩阵  $M$  作用在正交单位向量  $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$  和  $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$  上之后, 得到  $\begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$  和  $\begin{pmatrix} -\sin\theta \\ \cos\theta \end{pmatrix}$  也是正交的。

令  $\begin{pmatrix} u_1 \\ v_1 \end{pmatrix}$  和  $\begin{pmatrix} u_2 \\ v_2 \end{pmatrix}$  分别是  $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$  和  $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$  方向上的单位向量, 即  $\begin{pmatrix} u_1 \\ v_1 \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$ ,  $\begin{pmatrix} u_2 \\ v_2 \end{pmatrix} = \begin{pmatrix} -\sin\theta \\ \cos\theta \end{pmatrix}$ , 写在一起就是  $\begin{pmatrix} u_1 & u_2 \\ v_1 & v_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ , 整理得:

这样就得到矩阵  $M$  的奇异值分解。奇异值  $\sigma_1$  和  $\sigma_2$  分别是  $\begin{pmatrix} u_1 \\ v_1 \end{pmatrix}$  和  $\begin{pmatrix} u_2 \\ v_2 \end{pmatrix}$  的长度。很容易可以把结论推广到一般  $n$  维情形。

现在咱们给出一个更简洁更直观的奇异值的几何意义。先来一段线性代数的推导。

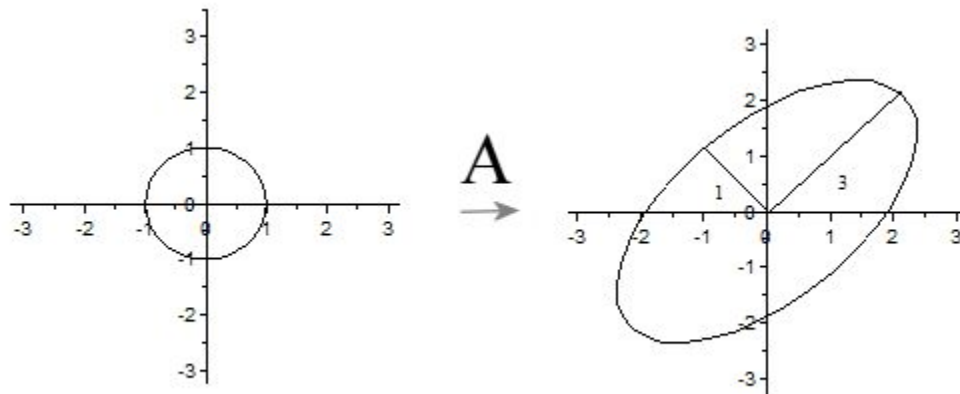
假设矩阵  $A$  的奇异值分解为

其中  $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$  是二维平面的向量。根据奇异值分解的性质,  $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$  线性无关,  $\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$  线性无关。那么对二维平面上任意的向量  $x$ , 都可以表示为:  $x = \begin{pmatrix} u_1 & u_2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ 。

当  $A$  作用在  $x$  上时,

令  $\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$ , 我们可以得出结论: 如果  $x$  是在单位圆  $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$  上, 那么  $y$  正好在椭圆  $\begin{pmatrix} \sigma_1 u_1 \\ \sigma_2 u_2 \end{pmatrix}$  上。这表明: 矩阵  $A$  将二维平面中单位圆变换成椭圆, 而两个奇异值正好是椭圆的两个半轴长, 长轴所在的直线是  $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ , 短轴所在的直线是  $\begin{pmatrix} u_2 \\ -u_1 \end{pmatrix}$ 。

推广到一般情形：一般矩阵  $A$  将单位球 变换为超椭球面  
那么矩阵  $A$  的每个奇异值恰好就是超椭球的每条半轴长度。



奇异值的计算是一个难题，是一个  $O(N^3)$  的算法。在单机的情况下当然是没问题的，matlab 在一秒钟内就可以算出  $1000 * 1000$  的矩阵的所有奇异值，但是当矩阵的规模增长的时候，计算的复杂度呈 3 次方增长，就需要并行计算参与了。Google 的吴军老师在数学之美系列谈到 SVD 的时候，说起 Google 实现了 SVD 的并行化算法，说这是对人类的一个贡献，但是也没有给出具体的计算规模，也没有给出太多有价值的信息。

其实 SVD 还是可以用并行的方式去实现的，在解大规模的矩阵的时候，一般使用迭代的方法，当矩阵的规模很大（比如说上亿）的时候，迭代的次数也可能会上亿次，如果使用 Map-Reduce 框架去解，则每次 Map-Reduce 完成的时候，都会涉及到写文件、读文件的操作。个人猜测 Google 云计算体系中除了 Map-Reduce 以外应该还有类似于 MPI 的计算模型，也就是节点之间是保持通信，数据是常驻在内存中的，这种计算模型比 Map-Reduce 在解决迭代次数非常多的时候，要快了很多倍。

Lanczos 迭代就是一种解对称方阵部分特征值的方法（之前谈到了，解  $A' * A$  得到的对称方阵的特征值就是解  $A$  的右奇异向量），是将一个对称的方程化为一个三对角矩阵再进行求解。按网上的一些文献来看，Google 应该就是用这种方法去做的奇异值分解的。请见 Wikipedia 上面的一些引用的论文，如果理解了那些论文，也“几乎”可以做出一个 SVD 了。

## 二、奇异值的直观应用

### 2.1 女神图片压缩

下面，咱们从女神上野树里（Ueno Juri）的一张像素为高度 450\*宽度 333 的照片，来直观理解奇异值在物理上到底代表什么意义（请屏幕前的痴汉暂停舔屏）。



我们都知道，图片实际上对应着一个矩阵，矩阵的大小就是像素大小，比如这张图对应的矩阵阶数就是  $450 \times 333$ ，矩阵上每个元素的数值对应着像素值。我们记这个像素矩阵为  $A$

现在我们对矩阵  $A$  进行奇异值分解。直观上，奇异值分解将矩阵分解成若干个秩一矩阵之和，用公式表示就是：

其中等式右边每一项前的系数  $\sigma_i$  就是奇异值， $u$  和  $v$  分别表示列向量，秩-1矩阵的意思是矩阵秩为 1。注意到每一项  $\sigma_i u_i v_i^T$  都是秩为 1 的矩阵。我们假定奇异值满足（奇异值大于 0 是个重要的性质，但这里先别在意）

如果不满足的话重新排列顺序即可，这无非是编号顺序的问题。

既然奇异值有从大到小排列的顺序，我们自然要问，如果只保留大的奇异值，舍去较小的奇异值，这样(1)式里的等式自然不再成立，那会得到怎样的矩阵——也就是图像？

令  $\tilde{U} = [u_1, \dots, u_k]$ ，这只保留(1)中等式右边第一项，然后作图：



结果就是完全看不清是啥.....我们试着多增加几项进来:

再作图





隐约可以辨别这是短发伽椰子的脸.....但还是很模糊，毕竟我们只取了 5 个奇异值而已。下面我们取 20 个奇异值试试，也就是(1)式等式右边取前 20 项构成



虽然还有些马赛克般的模糊，但我们总算能辨别出这是 Juri 酱的脸。当我们取到(1)式等式右边前 50 项时：



我们得到和原图差别不大的图像。也就是说当  $k$  从 1 不断增大时，不断的逼近  $A$ 。让我们回到公式

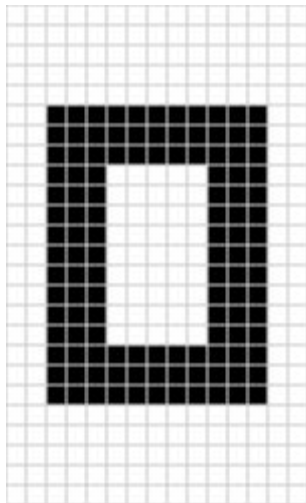
矩阵  $A$  表示一个  $450 \times 333$  的矩阵，需要保存 个元素的值。等式右边  $u$  和  $v$  分别是  $450 \times 1$  和  $333 \times 1$  的向量，每一项有  $1 + 450 + 333 = 784$  个元素。如果我们要存储很多高清的图片，而又受限于存储空间的限制，在尽可能保证图像可被识别的精度的前提下，我们可以保留奇异值较大的若干项，舍去奇异值较小的项即可。例如在上面的例子中，如果

我们只保留奇异值分解的前 50 项，则需要存储的元素为  $50 \times 25 + 50 \times 15 = 1250$ ，和存储原始矩阵 A 相比，存储量仅为后者的 26%。

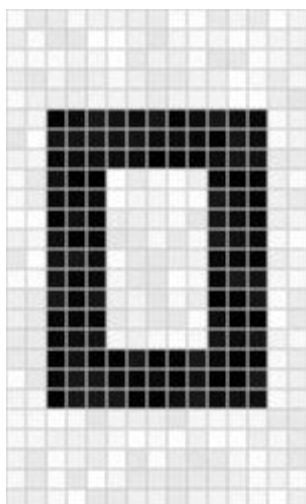
奇异值往往对应着矩阵中隐含的重要信息，且重要性和奇异值大小正相关。每个矩阵 A 都可以表示为一系列秩为 1 的“小矩阵”之和，而奇异值则衡量了这些“小矩阵”对于 A 的权重。

## 2.2 图像去噪

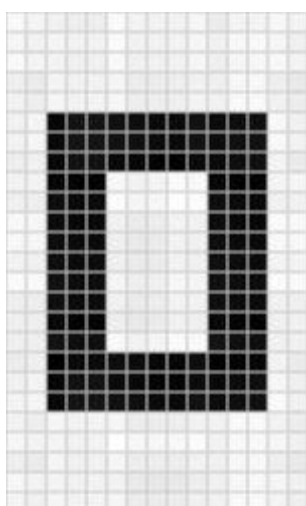
在图像处理领域，奇异值不仅可以应用在数据压缩上，还可以对图像去噪。如果一副图像包含噪声，我们有理由相信那些较小的奇异值就是由于噪声引起的。当我们强行令这些较小的奇异值为 0 时，就可以去除图片中的噪声。如下是一张 25\*15 的图像



但往往我们只能得到如下带有噪声的图像（和无噪声图像相比，下图的部分白格子中带有灰色）：



通过奇异值分解，我们发现矩阵的奇异值从大到小分别为：14.15, 4.67, 3.00, 0.21, ....., 0.05。除了前 3 个奇异值较大以外，其余奇异值相比之下都很小。强行令这些小奇异值为 0，然后只用前 3 个奇异值构造新的矩阵，得到



可以明显看出噪声减少了（白格子上灰白相间的图案减少了）。

奇异值分解还广泛的用于主成分分析 (Principle Component Analysis, 简称 PCA) 和推荐系统 (如 Netflix 的电影推荐系统) 等。在这些应用领域, 奇异值也有相应的意义。

#### 参考文献

- 1 <https://www.cnblogs.com/LeftNotEasy/archive/2011/01/19/svd-and-applications.html>
- 2 <https://www.zhihu.com/question/22237507>
- 3 We Recommend a Singular Value Decomposition (Feature Column from the AMS)

## 请通俗的解释下什么叫张量分解?

解析:

在介绍张量分解 (tensor decomposition) 之前, 我们可能需要先简单地了解一下张量是什么, 然后再考虑张量分解有什么用途, 并如何像稀疏矩阵分解 (matrix decomposition/factorization) 一样来对稀疏张量进行分解。

从初中到大学, 我们接触最多的可能只是标量 (scalar)、向量 (vector) 和矩阵 (matrix), 而张量则不那么常见, 但实际上, 标量是第 0 阶张量, 向量是第 1 阶张量, 矩阵是第 2 阶张量, 第 3 阶或阶数更高的张量被称为高阶张量 (higher-order tensor), 一般提到的张量都是特指高阶张量, 这在接下来的叙述中也不例外。

我们也知道，在一个矩阵中，某一元素的位置可以说成“第  $i$  行第  $j$  列”的形式，要表达某一元素的位置需要两个索引构成的组合  $(i,j)$ ，类似地，在一个第 3 阶张量里面，表达某一元素的位置需要三个索引构成的组合  $(i,j,k)$ 。在处理稀疏矩阵和稀疏张量时，用索引来标记元素的位置会带来很多便利。

另外，阶数  $d$  的张量可以理解为矩阵的  $d$  维泛化，在这里，阶数  $d$  其实就是空间维度 (spatial dimension)，张量可以被视为多维数组。

张量分解从本质上来说是矩阵分解的高阶泛化。对矩阵分解有所了解的读者可能知道，矩阵分解有三个很明显的用途，即降维处理、缺失数据填补（或者说成“稀疏数据填补”）和隐性关系挖掘，其实张量分解也能够很好地满足这些用途。因此，在介绍张量分解之前，我们先来看看相对简单的矩阵分解，并掌握稀疏矩阵的分解过程。

## 1 推荐系统中常用的矩阵分解

在我们常见的推荐系统（如商品推荐系统、电影推荐系统等）中，给定一个大小为  $n \times m$  的评分矩阵  $R$ ，元素  $R_{ij}$  表示用户 (user)  $i$  对项 (item, 如电影、商品等)  $j$  的评分值，如 1~5 分不等。

当矩阵  $R$  的秩为  $k$ ，并且能够写成如下形式

其中， $U$  是大小为  $n \times k$  的矩阵（用户因子矩阵，user-factor matrix）， $V$  是大小为  $k \times m$  的矩阵（项因子矩阵，item-factor matrix）。这一过程就是矩阵分解。

当  $k \ll \min(n, m)$  时，我们可以将矩阵分解的过程看作是一个低秩逼近问题（low-rank approximation problem），原来的分解过程则变成

和前面相同， $U$  是大小为  $n \times k$  的矩阵（用户因子矩阵，user-factor matrix）， $V$  是大小为  $k \times m$  的矩阵（项因子矩阵，item-factor matrix）。在这个低秩逼近问题中，可以很明显地看出整体误差为残差矩阵  $E$  中所有元素的平方和，即  $\|E\|_F^2$ （这种写法含义是矩阵  $F$  的范数的平方，等价于矩阵中所有元素的平方和）。

如果简单地使总的误差最小，那么，可以将矩阵分解的逼近问题转化为一个无约束的优化问题，即

在实际应用中，这里的评分矩阵  $R$  往往是一个稀疏矩阵，即很多位置上的元素是空缺的，或者说根本不存在。试想一下，如果有 10000 个用户，同时存在 10000 部电影，如果我



们需要构造一个评分矩阵，难道每个用户都要把每部电影都看一遍才知道用户的偏好吗？

其实不是，我们只需要知道每个用户仅有的一些评分就可以利用矩阵分解来估计用户的偏好，并最终推荐用户可能喜欢的电影。

在这里，我们将矩阵  $R$  中存在评分的位置记为  $S$ ，所有观测到的位置索引记作集合  $S$ ，其中，用户的索引为  $i$ ，项的索引为  $j$ ，需要注意的是，推荐系统中的矩阵分解对原矩阵  $R$  有一定的要求，即矩阵的每行和每列至少有一个元素。此时，任意位置  $(i, j)$  所对应的评分估计值为  $\hat{r}_{ij}$ 。

则原来的优化问题等价于

对目标函数  $J$  中的  $u_i$  和  $v_j$  求偏导数，得

；

这里，可以将两个偏导数分别简写为  $\delta u_i$  和  $\delta v_j$ ，其中， $\delta u_i = \frac{\partial J}{\partial u_i}$ ， $\delta v_j = \frac{\partial J}{\partial v_j}$ 。

根据梯度下降（gradient descent）方法， $u_i$  和  $v_j$  在每次迭代过程中的更新公式为

这里的  $\eta$  表示梯度下降的步长，又称为学习率（learning rate），另外，更新公式中的求和项下标  $i$  和  $j$  分别表示向量  $\mathbf{u}$  和  $\mathbf{v}$  上所有非零元素的位置索引构成的集合。

## 2 隐性因子模型

上面已经简单地介绍了矩阵分解的原理，对推荐系统有了解的读者可能对上面这一过程并不陌生，上面的矩阵分解在推荐系统中常常被称为隐性因子模型（latent factor model, LFM），其实张量分解与上述过程非常相似，为了便于读者初步地理解张量分解，这里将会沿用矩阵分解类似的推导过程。

定义一个关于用户（user） $u$  在环境（context，如时间，注意：这里将 context 翻译成“环境”不一定准确，在隐性语义分析中常常理解为“语境”或“上下文”） $c$  下对项（item） $i$  的评分为  $r_{uic}$ ，评分张量的大小为  $I_u \times I_i \times I_c$ ，所有观测到位置索引仍然记作集合  $\Omega$ ，其中，用户的索引为  $u$ ，项的索引为  $i$ ，环境的索引为  $c$ 。

Charu C. Aggarwal 在其著作《Recommender systems》中给出了一个特殊的张量分解结构，即大小为  $I_u \times I_i \times I_c$  的评分张量  $\mathbf{R}$  分解后会得到三个矩阵，这三个矩阵分别是：大小为  $I_u \times K$  的用户因子矩阵（user-factor matrix）、大小为  $I_i \times K$  的项因子矩阵

(item-factor matrix) 和大小为  $I \times J$  的环境因子矩阵  $C$  (context-factor matrix)  $J \times C$ ), 这种分解结构是隐性因子模型的一种高阶泛化。

此时, 第 3 阶张量  $\mathcal{R}$  上任意位置  $(i, j, c)$  所对应的评分估计值为

即

与矩阵分解中的低秩逼近问题相似, 评分张量分解的逼近问题为

对目标函数  $J$  中的  $u_{\{iq\}}$ 、 $v_{\{jq\}}$ 和  $w_{\{cq\}}$ 求偏导数, 得

$$\frac{\partial J}{\partial u_{iq}} = 0;$$

$$\frac{\partial J}{\partial v_{jq}} = 0;$$

根据梯度下降方法,  $u_{\{iq\}}$ 、 $v_{\{jq\}}$ 和  $w_{\{cq\}}$ 在每次迭代过程中的更新公式为

$$u_{iq} = u_{iq} - \eta \frac{\partial J}{\partial u_{iq}};$$

$$v_{jq} = v_{jq} - \eta \frac{\partial J}{\partial v_{jq}};$$

$$w_{cq} = w_{cq} - \eta \frac{\partial J}{\partial w_{cq}}.$$

需要注意的是，更新公式中的求和项下标  $i$ 、 $j$  和  $k$  分别表示矩阵  $\mathbf{U}$ 、 $\mathbf{V}$  和  $\mathbf{W}$  上所有非零元素的位置索引构成的集合。

介绍到这里，可能部分读者会疑问，这里介绍的张量分解过程为何与我们常在大量文献中见到的 Tucker 张量分解和 CP 张量分解（可认为是 Tucker 张量分解的特例）不一样呢？  
接下来我们来看看采用 Tucker 分解的情况是怎样的。

#### 参考文献

1 <https://zhuanlan.zhihu.com/p/24798389>

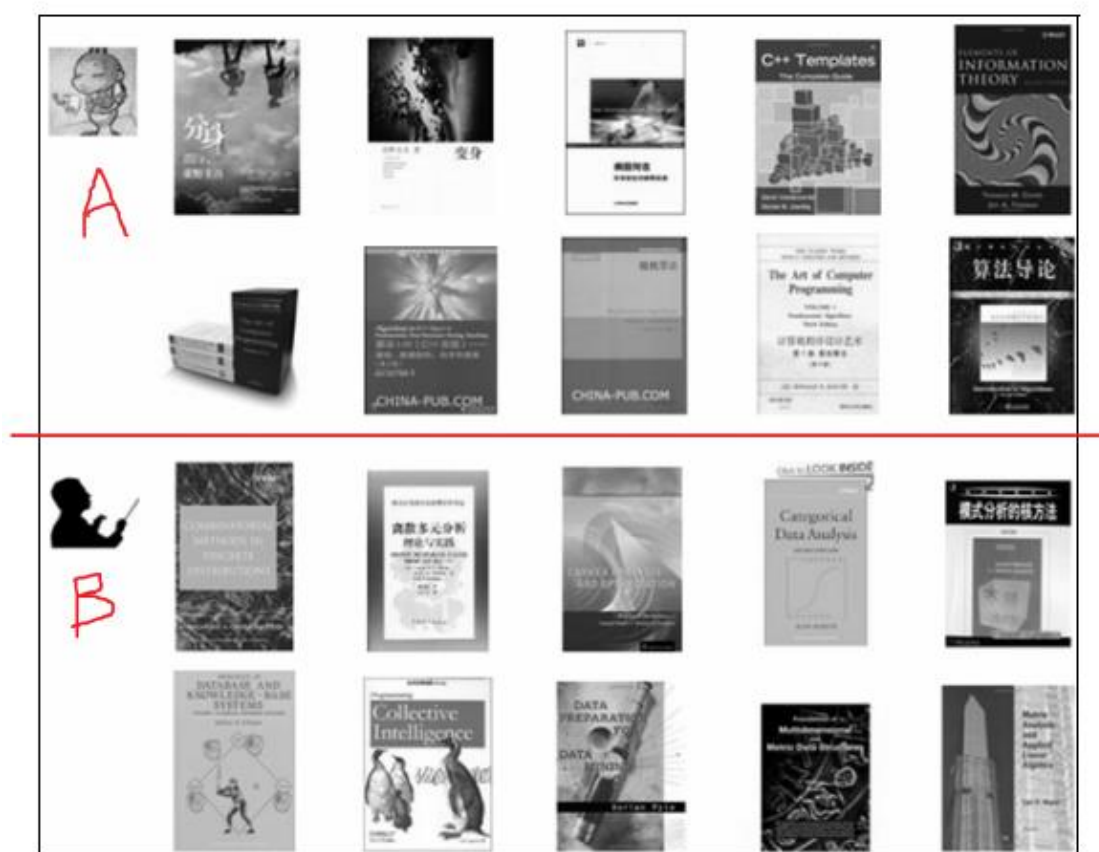
## 请说说隐语义模型 LFM 背后的原理

解析：

### 一、隐语义模型的基本思想

隐语义模型是近年来推荐系统领域较为热门的话题，它主要是根据隐含特征将用户与物品联系起来。

现从简单例子出发介绍隐语义模型的基本思想。假设用户 A 喜欢《算法导论》，用户 B 喜欢《离散多元分析》，现在小编要对用户 A 和用户 B 推荐其他书籍。



基于 UserCF(基于用户的协同过滤), 找到与他们偏好相似的用户, 将相似用户偏好的书籍推荐给他们;

基于 ItemCF(基于物品的协同过滤), 找到与他们 当前偏好书籍相似的其他书籍, 推荐给他们。

其实还有一种思路，就是根据用户的当前偏好信息，得到用户的兴趣偏好，将该类兴趣对应的物品推荐给当前用户。比如，用户 A 喜欢的《算法导论》属于计算机类的书籍，那我们可以将其他的计算机类书籍推荐给用户 A；用户 B 喜欢的是数学类数据，可将《微积分概念发展史》等这类文字作品推荐给用户 B。这就是隐语义模型，依据“兴趣”这一隐

含特征将用户与物品进行连接，需要说明的是此处的“兴趣”其实是对物品类型的一个分类而已。

说白了，就是找出隐含的特征、关联，然后根据做关联推荐。

二、隐语义模型的数学理解

我们从数学角度来理解隐语义模型。如下图所示，R 矩阵是用户对物品的偏好信息（Rij 表示的是 user i 对 item j 的兴趣度），P 矩阵是用户对各物品类别的一个偏好信息（Pij 表示的是 user i 对 class j 的兴趣度），Q 矩阵是各物品所归属的的物品类别的信息（Qij 表示的是 item j 在 class i 中的权重）。

隐语义模型就是要将矩阵 R 分解为矩阵 P 和矩阵 Q 的乘积，即通过矩阵中的物品类别(class)将用户 user 和物品 item 联系起来。实际 上我们需要根据用户当前的物品偏好信息 R 进行计算，从而得到对应的矩阵 P 和矩阵 Q。

	item 1	item 2	item 3	item 4
user 1	R11	R12	R13	R14
user 2	R21	R22	R23	R24
user 3	R31	R32	R33	R34

R

=

	class 1	class 2	class 3
user 1	P11	P12	P13
user 2	P21	P22	P23
user 3	P31	P32	P33

P

×

	item 1	item 2	item 3	item 4
class 1	Q11	Q12	Q13	Q14
class 2	Q21	Q22	Q23	Q24
class 3	Q31	Q32	Q33	Q34

Q

三、隐语义模型所解决的问题

从上面的陈述我们可以知道，要想实现隐语义模型，我们需解决以下问题：

如何对物品进行分类，分成几类？

如何确定用户对哪些物品类别有兴趣，兴趣程度如何？

对于一个给定的类，选择这个类中的哪些物品进行推荐，如何确定物品在某个类别中的权重？

对于第一个问题，就是找对应的编辑人进行人工分类，但是人工分类会存在以下问题：

当前的人工分类不能代表用户的意见：比如一本《算法导论》，从编辑人员的角度看会属于计算机类，但从用户来看，可能会归到数学类；难以把握分类的粒度：仍以《算法导论》为例，可分类得粗一点，属于计算机类，也可分得再细一点，属于计算机类别中的算法类；

难以给一个物品多个类别：一本小说可归为文学类，或言情类等；

难以给出多维度的分类：对物品的分类可从多个角度进行，比如一本书可从内容进行分类，也可从作者角度进行分类；

难以确定一个物品在某一分类中的权重：一个物品可能属于多个类别，但是权重不同。

基于以上局限性，显然我们不能靠由个人的主观想法建立起来的分类标准对整个平台用户喜好进行标准化。

隐语义模型是从用户的偏好数据出发进行个性推荐的，即基于用户的行为统计进行自动聚类，所以能解决以上提到的 5 个问题：

隐语义模型是基于用户的行为数据进行自动聚类的，能反应用户对物品的分类意见；

我们可以指定将物品聚类的类别数  $k$ ， $k$  越大，则粒度越细；

隐语义模型能计算出物品在各个类别中的权重，这是根据用户的行为数据统计的，不会只将其归到一类中；

隐语义模型得到的物品类别不是基于同一个维度的，维度是由用户的共同兴趣决定的。

#### 四、隐语义模型的样本问题

隐语义模型在显性反馈数据（也就是评分数据）上能解决评分预测问题并达到了很好的精度。不过推荐系统主要讨论的是隐性反馈数据集，这种数据集的特点是只有正样本（用户喜欢什么物品），而没有负样本（用户对什么物品不感兴趣）。

那么，在隐性反馈数据集上应用隐语义模型解决推荐的第一个关键问题就是如何给每个用户生成负样本。我们发现对负样本采样时应该遵循以下原则：

对每个用户，要保证正负样本的平衡（数目相似）；

每个用户采样负样本时，要选取那些很热门，而用户却没有行为的物品：一般认为，很热门而用户却没有行为更加代表用户对这个物品不感兴趣。因为对于冷门的物品，用户可能是压根没在网站中发现这个物品，所以谈不上是否感兴趣；

#### 五、隐语义模型的推导思路

隐语义模型是根据如下公式来计算用户 U 对物品 I 的兴趣度。

$$R_{UI} = P_U Q_I = \sum_{k=1}^K P_{U,k} Q_{k,I}$$

其中，隐语义模型会把物品分成 K 个类型，这个是我们根据经验和业务知识进行反复尝试决定的， $p(u,k)$ 表示用户 u 对于第 k 个分类的喜爱程度( $1 < k \leq K$ )， $q(k, i)$ 表示物品 i 属于第 k 个分类的权重( $1 < k \leq K$ )。



现在我们讨论下如何计算矩阵 P 和矩阵 Q 中的参数值。一般做法就是最优化损失函数来求参数。损失函数如下所示：

$$C = \sum_{(U,I) \in K} (R_{UI} - \hat{R}_{UI})^2 = \sum_{(U,I) \in K} (R_{UI} - \sum_{k=1}^K P_{U,k} Q_{k,I})^2 + \lambda \|P_U\|^2 + \lambda \|Q_I\|^2$$

上式中的

$$\lambda \|P_U\|^2 + \lambda \|Q_I\|^2$$

是用来防止过拟合的正则化项， $\lambda$  需要根据具体应用场景反复实验得到。损失函数的意义是用户 u 对物品 i 的真实喜爱程度与推算出来的喜爱程度的均方根误差，通俗来说就是真实的喜爱程度与推算的喜爱程度的误差，要使模型最合理当然就是使这个误差达到最小值。

公式中最后两项是惩罚因子，用来防止分类数取得过大而使误差减少的不合理做法的发生， $\lambda$  参数是一个常数，需要根据经验和业务知识进行反复尝试决定的。损失函数的优化使用随机梯度下降算法：1) 对两组未知数求偏导数

$$\frac{\partial C}{\partial P_{Uk}} = -2(R_{UI} - \sum_{k=1}^K P_{U,k} Q_{k,I}) Q_{kI} + 2\lambda P_{Uk}$$

$$\frac{\partial C}{\partial Q_{kI}} = -2(R_{UI} - \sum_{k=1}^K P_{U,k} Q_{k,I}) P_{Uk} + 2\lambda Q_{kI}$$

2) 根据随机梯度下降法得到递推公式

$$P_{Uk} = P_{Uk} + \alpha((R_{UI} - \sum_{k=1}^K P_{U,k} Q_{k,I}) Q_{kI} - \lambda P_{Uk})$$

$$Q_{kI} = Q_{kI} + \alpha((R_{UI} - \sum_{k=1}^K P_{U,k} Q_{k,I}) P_{Uk} - \lambda Q_{kI})$$

其中 $\alpha$ 是在梯度下降的过程中的步长(也可以称作学习速率)，这个值不宜过大也不宜过小，过大会产生震荡而导致很难求得最小值，过小会造成计算速度下降，需要经过试验得到合适的值。最终会求得每个用户对于每个隐分类的喜爱程度矩阵 P 和每个物品与每个隐分类的匹配程度矩阵 Q。

在用户对物品的偏好信息矩阵 R 中，通过迭代可以求得每个用户对每个物品的喜爱程度，选取喜爱程度最高而且用户没有反馈过的物品进行推荐。

在隐语义模型中，重要的参数有以下 4 个：

- 1) 隐分类的个数 F;
- 2) 梯度下降过程中的步长(学习速率) $\alpha$ ;
- 3) 损失函数中的惩罚因子 $\lambda$ ;
- 4) 正反馈样本数和负反馈样本数的比例 ratio; 这四项参数需要在试验过程中获得最合适的值

1) 3) 4) 这三项需要根据推荐系统的准确率、召回率、覆盖率及流行度作为参考, 而 2) 步长 $\alpha$ 要参考模型的训练效率。

## 六、优缺点分析

隐语义模型在实际使用中有一个困难，那就是它很难实现实时推荐。经典的隐语义模型每次训练时都需要扫描所有的用户行为记录，这样才能计算出用户对于 每个隐分类的喜爱程度矩阵  $P$  和每个物品与每个隐分类的匹配程度矩阵  $Q$ 。而且隐语义模型的训练需要在用户行为记录上反复迭代才能获得比较好的性能，因此 LFM 的每次训练都很耗时，一般在实际应用中只能每天训练一次，并且计算出所有用户的推荐结果。从而隐语义模型不能因为用户行为的变化实时地调整推荐结果 来满足用户最近的行为。

## 七、参考文献

1 [https://www.jianshu.com/p/7b6bb28c1753?tdsourcetag=s\\_pcqq\\_aiomsg](https://www.jianshu.com/p/7b6bb28c1753?tdsourcetag=s_pcqq_aiomsg)

2 <https://www.cnblogs.com/wkang/p/10091797.html>

# 请详细说说你对 Learning to rank 的通俗理解

解析：

作者：刘丁

来源：<https://tech.meituan.com/2018/12/20/head-in-l2r.html>

## 一、引言

我们正处在一个知识爆炸的时代，伴随着信息量的剧增和人工智能的蓬勃发展，互联网公司越发具有强烈的个性化、智能化信息展示的需求。而信息展示个性化的典型应用主要包括搜索列表、推荐列表、广告展示等等。

很多人不知道的是，看似简单的个性化信息展示背后，涉及大量的数据、算法以及工程架构技术，这些足以让大部分互联网公司望而却步。究其根本原因，个性化信息展示背后的技术是排序学习问题（Learning to Rank）。市面上大部分关于排序学习的文章，要么偏算法、要么偏工程。虽然算法方面有一些系统性的介绍文章，但往往对读者的数学能力要求比较高，也比较偏学术，对于非算法同学来说门槛非常高。而大部分工程方面的文章又比较粗糙，基本上停留在 Google 的 Two-Phase Scheme 阶段，从工程实施的角度来说，远远还不够具体。

对于那些由系统开发工程师负责在线排序架构的团队来说，本文会采用通俗的例子和类比方式来阐述算法部分，希望能够帮助大家更好地理解和掌握排序学习的核心概念。如果是算法工程师团队的同学，可以忽略算法部分的内容。本文的架构部分阐述了美团点评到店餐饮业务线上运行的系统，可以作为在线排序系统架构设计的参考原型直接使用。该架构在服务治理、分层设计的理念，对于保障在线排序架构的高性能、高可用性、易维护性也具有一定的参考价值。包括很多具体环节的实施方案也可以直接进行借鉴，例如流量分桶、流量分级、特征模型、级联模型等等。

总之，让开发工程师能够理解排序学习算法方面的核心概念，并为在线架构实施提供细颗粒度的参考架构，是本文的重要目标。

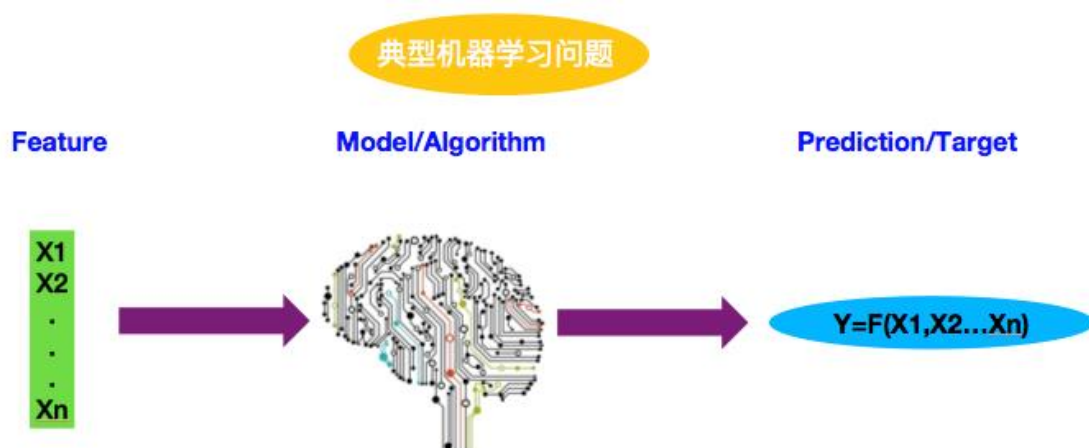
## 算法部分

机器学习涉及优化理论、统计学、数值计算等多个领域。这给那些希望学习机器学习核心概念的系统开发工程师带来了很大的障碍。不过，复杂的概念背后往往蕴藏着朴素的道理。本节将尝试采用通俗的例子和类比方式，来对机器学习和排序学习的一些核心概念进行揭秘。

## 二、机器学习

### 2.1 什么是机器学习？

典型的机器学习问题，如下图所示：



机器学习模型或算法（Model/Algorithm）会根据观察到的特征值（Feature）进行预测，给出预测结果或者目标（Prediction/Target）。这就像是一个函数计算过程，对于特定  $X$  值（Feature），算法模型就像是函数，最终的预测结果是  $Y$  值。不难理解，机器学习的核心问题就是如何得到预测函数。

Wikipedia 的对机器学习定义如下：

“Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to learn with data, without being explicitly programmed.”

机器学习的最重要本质是从数据中学习，得到预测函数。人类的思考过程以及判断能力本质上也是一种函数处理。从数据或者经验中学习，对于人类来说是一件再平常不过的事情了。例如人们通过观察太阳照射物体影子的长短而发明了日晷，从而具备了计时和制定节气的能力。尼罗河畔法力无边的古埃及人通过尼罗河水的涨落发明了古埃及历法。



**24节气**

又比如人们通过观察月亮形状的变化而发明了阴历。



阴历

如果机器能够像人一样具备从数据中学习的能力，从某种意义上讲，就具备了一定的“智能”。现在需要回答的两个问题就是：

到底什么是“智能”？

如何让机器具备智能？

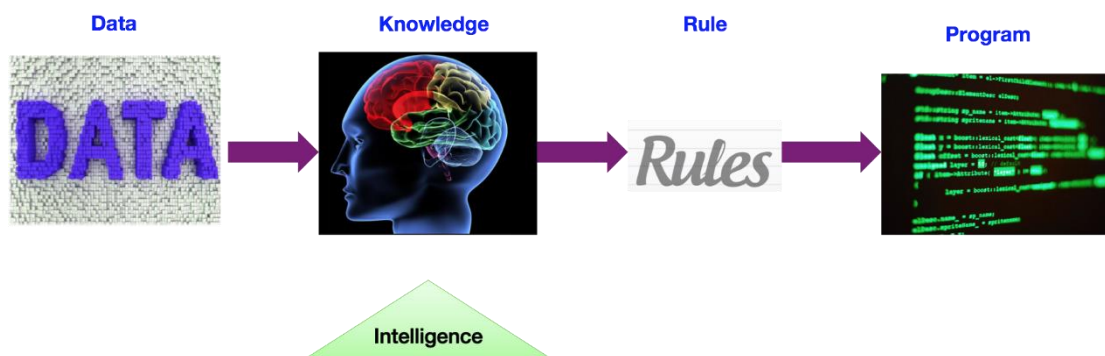
## 2.2 什么是智能？

在回答这个问题之前，我们先看看传统的编程模式为什么不能称之为“智能”。传统的编程模式如下图所示，它一般要经历如下几个阶段：

人类通过观察数据（Data）总结经验，转化成知识（Knowledge）。

人类将知识转化成规则（Rule）。

工程师将规则转化成计算机程序（Program）。



在这种编程模式下，如果一个问题被规则覆盖，那么计算机程序就能处理。对于规则不能覆盖的问题，只能由人类来重新思考，制定新规则来解决。所以在这里“智能”角色主要由人类来承担。人类负责解决新问题，所以传统的程序本身不能称之为“智能”。

所以，“智能”的一个核心要素就是“举一反三”。

## 2.3 如何让机器具备智能？

在讨论这个问题之前，可以先回顾一下人类是怎么掌握“举一反三”的能力的？基本流程如下：

老师给学生一些题目，指导学生如何解题。学生努力掌握解题思路，尽可能让自己的答案和老师给出的答案一致。

学生需要通过一些考试来证明自己具备“举一反三”的能力。如果通过了这些考试，学生将获得毕业证书或者资格从业证书。

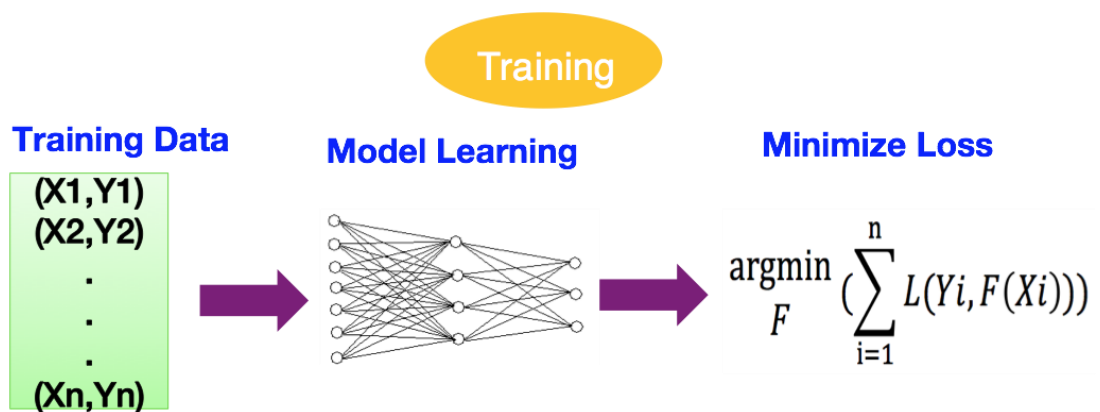
学生变成一个从业者之后将会面临并且处理很多之前没有碰到过的新问题。

机器学习专家从人类的学习过程中获得灵感，通过三个阶段让机器具备“举一反三”的能力。这三个阶段是：训练阶段（Training）、测试阶段（Testing）、推导阶段（Inference）。下面逐一进行介绍。

### 2.3.1 训练阶段

训练阶段如下图所示：





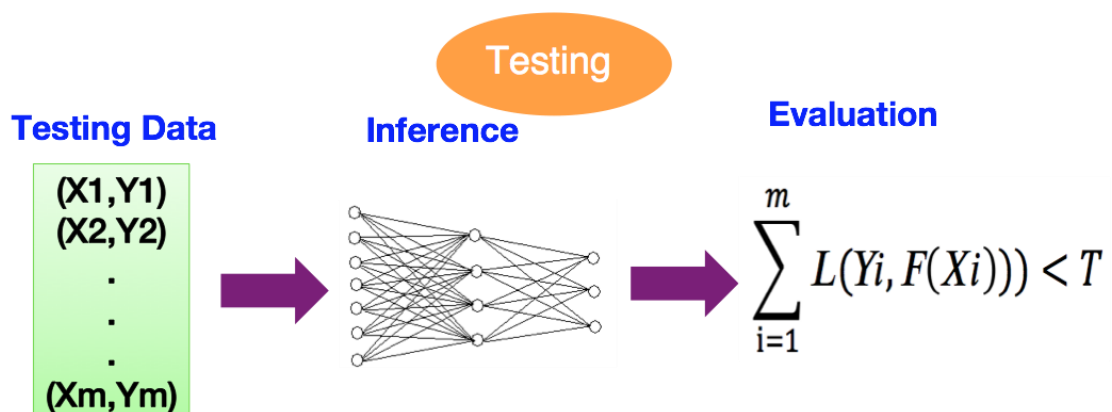
人类给机器学习模型一些训练样本  $(X, Y)$ ， $X$  代表特征， $Y$  代表目标值。这就好比老师教学生解题， $X$  代表题目， $Y$  代表标准答案。

机器学习模型尝试想出一种方法解题。

在训练阶段，机器学习的目标就是让损失函数值最小。类比学生尝试让自己解题的答案和老师给的标准答案差别最小，思路如出一辙。

### 2.3.2 测试阶段

测试阶段如下图所示：



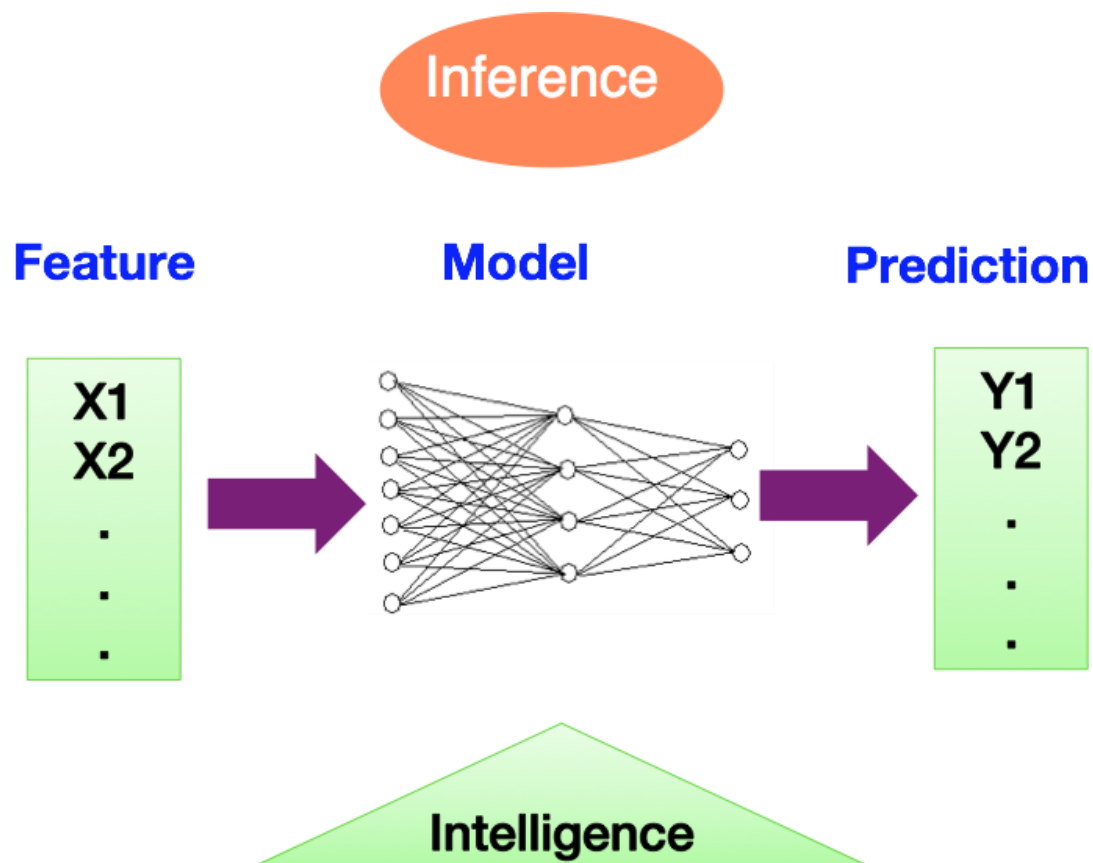
人类给训练好的模型一批完全不同的测试样本  $(X, Y)$ 。这就好比学生拿到考试试卷。

模型进行推导。这个过程就像学生正在考试答题。

人类要求测试样本的总损失函数值低于设定的最低目标值。这就像学校要求学生的考试成绩必须及格一样。

### 2.3.3 推导阶段

推导阶段如下图所示：



在这个阶段机器学习模型只能拿到特征值  $X$ ，而没有目标值。这就像工作中，人们只是在解决一个个的问题，但不知道正确的结果到底是什么。

在推导阶段，机器学习的目标就是预测，给出目标值。

## 三、排序学习

### 3.1 什么是排序学习？

Wikipedia 的对排序学习的定义如下：

“Learning to rank is the application of machine learning, typically supervised, semi-supervised or reinforcement learning, in the construction of ranking mo

dels for information retrieval systems. Training data consists of lists of items with some partial order specified between items in each list. This order is typically induced by giving a numerical or ordinal score or a binary judgment (e.g. "relevant" or "not relevant" ) for each item. The ranking model's purpose is to rank, i.e. produce a permutation of items in new, unseen lists in a way which is "similar" to rankings in the training data in some sense."

排序学习是机器学习在信息检索系统里的应用，其目标是构建一个排序模型用于对列表进行排序。排序学习的典型应用包括搜索列表、推荐列表和广告列表等等。

列表排序的目标是对多个条目进行排序，这就意味着它的目标值是有结构的。与单值回归和单值分类相比，结构化目标要求解决两个被广泛提起的概念：

列表评价指标

列表训练算法

### 3.2 列表评价指标

以关键词搜索返回文章列表为例，这里先分析一下列表评价指标要解决什么挑战。

第一个挑战就是定义文章与关键词之间的相关度，这决定了一篇文章在列表中的位置，相关度越高排序就应该越靠前。

第二个挑战是当列表中某些文章没有排在正确的位置时候，如何给整个列表打分。举个例子来说，假如对于某个关键词，按照相关性的高低正确排序，文档 1、2、3、4、5 应该依

次排在前 5 位。现在的挑战就是，如何评估 “2, 1, 3, 4, 5” 和 “1, 2, 5, 4, 3” 这两个列表的优劣呢？

列表排序的评价指标体系总来的来说经历了三个阶段，分别是 Precision and Recall、Discounted Cumulative Gain(DCG)和 Expected Reciprocal Rank(ERR)。我们逐一进行讲解。

### Precision and Recall(P-R)

本评价体系通过准确率（Precision）和召回率（Recall）两个指标来共同衡量列表的排序质量。对于一个请求关键词，所有的文档被标记为相关和不相关两种。

Precision 的定义如下：

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

Recall 的定义如下：

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

举个例子来说，对于某个请求关键词，有 200 篇文章实际相关。某个排序算法只认为 100 篇文章是相关的，而这 100 篇文章里面，真正相关的文章只有 80 篇。按照以上定义：

准确率=80/100=0.8

召回率=80/200=0.4。

Discounted Cumulative Gain(DCG)

P-R 的有两个明显缺点:

所有文章只被分为相关和不相关两档, 分类显然太粗糙。

没有考虑位置因素。

DCG 解决了这两个问题。对于一个关键词, 所有的文档可以分为多个相关性级别, 这里以  $rel_1, rel_2 \dots$  来表示。文章相关性对整个列表评价指标的贡献随着位置的增加而对数衰减, 位置越靠后, 衰减越严重。基于 DCG 评价指标, 列表前  $p$  个文档的评价指标定义如下:

$$DCG_p = \sum_{i=1}^p \frac{2^{rel_i} - 1}{\log_2(i + 1)}$$

对于排序引擎而言, 不同请求的结果列表长度往往不相同。当比较不同排序引擎的综合排序性能时, 不同长度请求之间的 DCG 指标的可比性不高。目前在工业界常用的是 Normalized DCG(nDCG), 它假定能够获取到某个请求的前  $p$  个位置的完美排序列表, 这个完美列表的分值称为 Ideal DCG(IDC), nDCG 等于 DCG 与 IDC 比值。所以 nDCG 是一个在 0 到 1 之间的值。

nDCG 的定义如下:

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

IDCG 的定义如下：

$$IDCG_p = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{\log_2(i + 1)}$$

|REL|代表按照相关性排序好的最多到位置 p 的结果列表。

Expected Reciprocal Rank(ERR)

与 DCG 相比，除了考虑位置衰减和允许多种相关级别（以 R1，R2，R3...来表示）以外，ERR 更进了一步，还考虑了排在文档之前所有文档的相关性。举个例子来说，文档 A 非常相关，排在第 5 位。如果排在前面的 4 个文档相关度都不高，那么文档 A 对列表的贡献就很大。反过来，如果前面 4 个文档相关度很大，已经完全解决了用户的搜索需求，用户根本就不会点击第 5 个位置的文档，那么文档 A 对列表的贡献就不大。

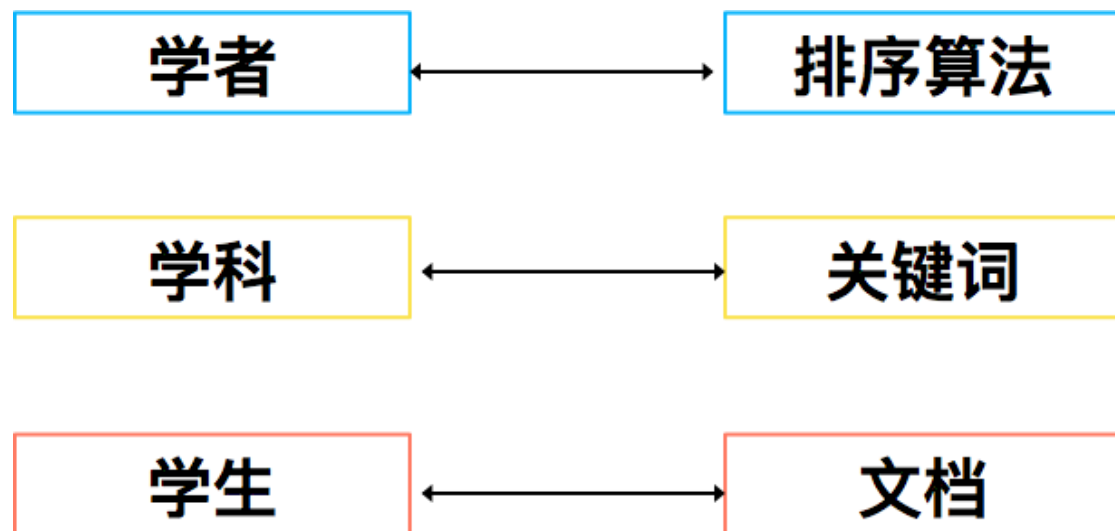
ERR 的定义如下：

$$ERR = \sum_{r=1}^n \frac{1}{r} \prod_{i=1}^{r-1} (1 - R_i) R_r$$

### 3.3 列表训练算法

做列表排序的工程师们经常听到诸如 Pointwise、Pairwise 和 Listwise 的概念。这些是什么东西呢，背后的原理又是什么呢？这里将逐一解密。

仍然以关键词搜索文章为例，排序学习算法的目标是为给定的关键词对文章列表进行排序。做为类比，假定有一个学者想要对各科学生排名进行预测。各种角色的对应关系如下：



首先我们要告诉学者每个学生的各种属性，这就像我们要告诉排序算法文档特征。对于目标值，我们却有三种方式来告诉学者：

1) 对于每个学科，我们可以告诉学者每个学生的成绩。比较每个学生的成绩，学者当然可以算出每个学生的最终排名。这种训练方法被称为 Pointwise。对于 Pointwise 算法，如果最终预测目标是一个实数值，就是一个回归问题。如果目标是概率预测，这就是一个分类问题，例如 CTR 预估。

2) 对于每个学科，我们可以告诉学者任意两个学生的相互排名。根据学生之间排名的情况，学者也可以算出每个学生的最终排名。这种训练方法被称为 Pairwise。Pairwise 算法的目标是减少逆序的数量，所以是个二分类问题。

3) 对于每个学科，我们可以直接告诉学者所有学生的整体排名。这种训练方法被称为 Listwise。Listwise 算法的目标往往是直接优化 nDCG、ERR 等评价指标。

这三种方法表面看上去有点像玩文字游戏，但是背后却是工程师和科学家们不断探索的结果。

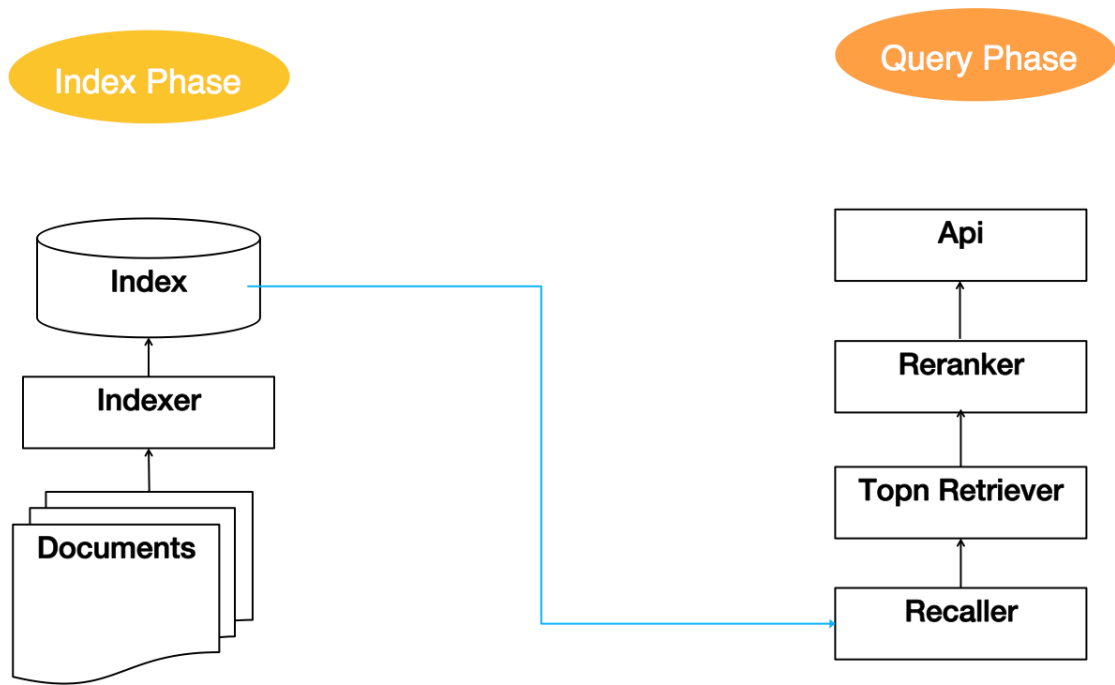
最直观的方案是 Pointwise 算法，例如对于广告 CTR 预估，在训练阶段需要标注某个文档的点击概率，这相对来说容易。Pairwise 算法一个重要分支是 Lambda 系列，包括 LambdaRank、LambdaMart 等，它的核心思想是：很多时候我们很难直接计算损失函数的值，但却很容易计算损失函数梯度（Gradient）。这意味着我们很难计算整个列表的 nDCG 和 ERR 等指标，但却很容易知道某个文档应该排的更靠前还是靠后。

Listwise 算法往往效果最好，但是如何为每个请求对所有文档进行标注是一个巨大的挑战。

### 3.4 在线排序架构

典型的信息检索包含两个阶段：索引阶段和查询阶段。这两个阶段的流程以及相互关系可以用下图来表示：





索引阶段的工作是由索引器（Indexer）读取文档（Documents）构建索引（Index）。

查询阶段读取索引做为召回，然后交给 Topn Retriever 进行粗排，在粗排后的结果里面将前 n 个文档传给 Reranker 进行精排。这样一个召回、粗排、精排的架构最初是由 Google 提出来的，也被称为 “Two-Phase Scheme”。

索引部分属于离线阶段，这里重点讲述在线排序阶段，即查询阶段。

### 三大挑战

在线排序架构主要面临三方面的挑战：特征、模型和召回。

特征挑战包括特征添加、特征算子、特征归一化、特征离散化、特征获取、特征服务治理等。

模型挑战包括基础模型完备性、级联模型、复合目标、A/B 实验支持、模型热加载等。

召回挑战包括关键词召回、LBS 召回、推荐召回、粗排召回等。

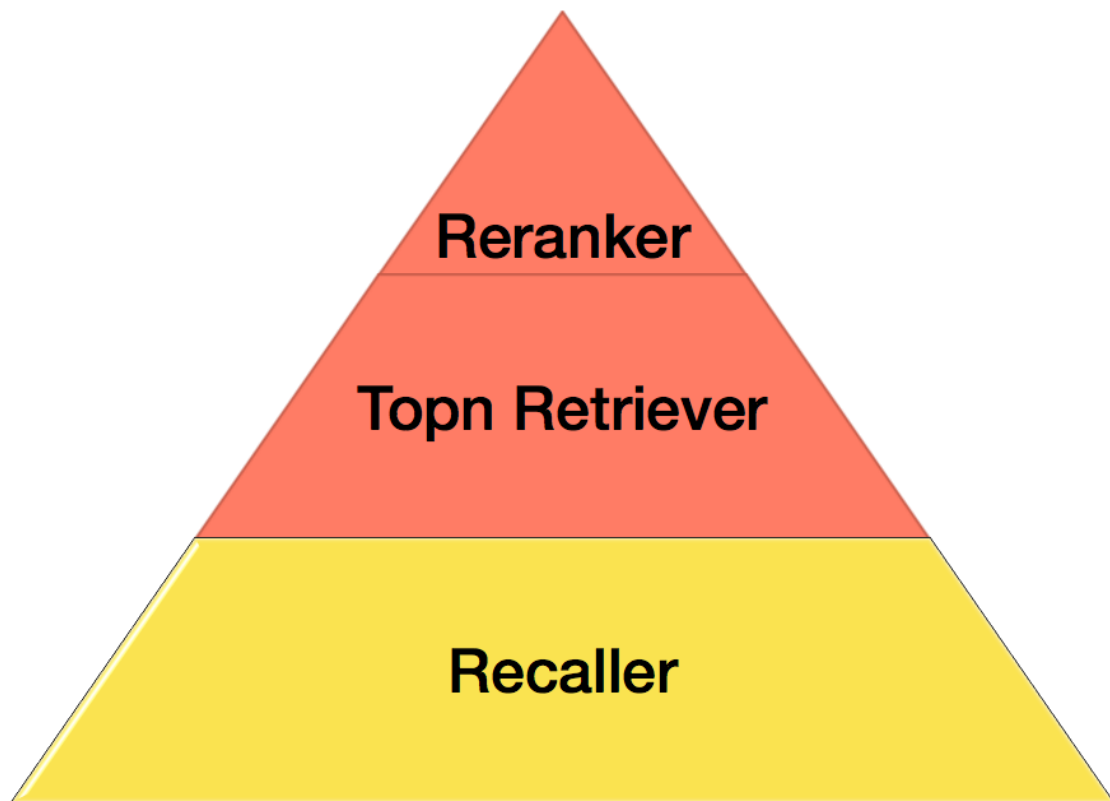
三大挑战内部包含了非常多更细粒度的挑战，孤立地解决每个挑战显然不是好思路。在线排序作为一个被广泛使用的架构值得采用领域模型进行统一解决。Domain-driven design(DDD)的三个原则分别是：领域聚焦、边界清晰、持续集成。

基于以上分析，我们构建了三个在线排序领域模型：召回治理、特征服务治理和在线排序分层模型。

### 3.5 召回治理

经典的 Two-Phase Scheme 架构如下图所示，查询阶段应该包含：召回、粗排和精排。

但从领域架构设计的角度来讲，粗排对于精排而言也是一种召回。和基于传统的文本搜索不同，美团点评这样的 O2O 公司需要考虑地理位置和距离等因素，所以基于 LBS 的召回也是一种召回。与搜索不同，推荐召回往往基于协同过滤来完成的，例如 User-Based CF 和 Item-Based CF。



综上所述，召回总体而言分成四大类：

关键词召回，我们采用 Elasticsearch 解决方案。

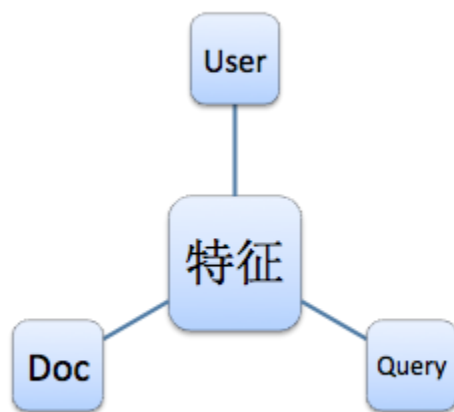
距离召回，我们采用 K-D tree 的解决方案。

粗排召回。

推荐类召回。

### 3.6 特征服务治理

传统的视角认为特征服务应该分为用户特征（User）、查询特征（Query）和文档特征（Doc），如下图：



这是比较纯粹的业务视角，并不满足 DDD 的领域架构设计思路。由于特征数量巨大，我们没有办法为每个特征设计一套方案，但是我们可以把特征进行归类，为几类特征分别设计解决方案。每类技术方案需要统一考虑性能、可用性、存储等因素。从领域视角，特征服务包含四大类：

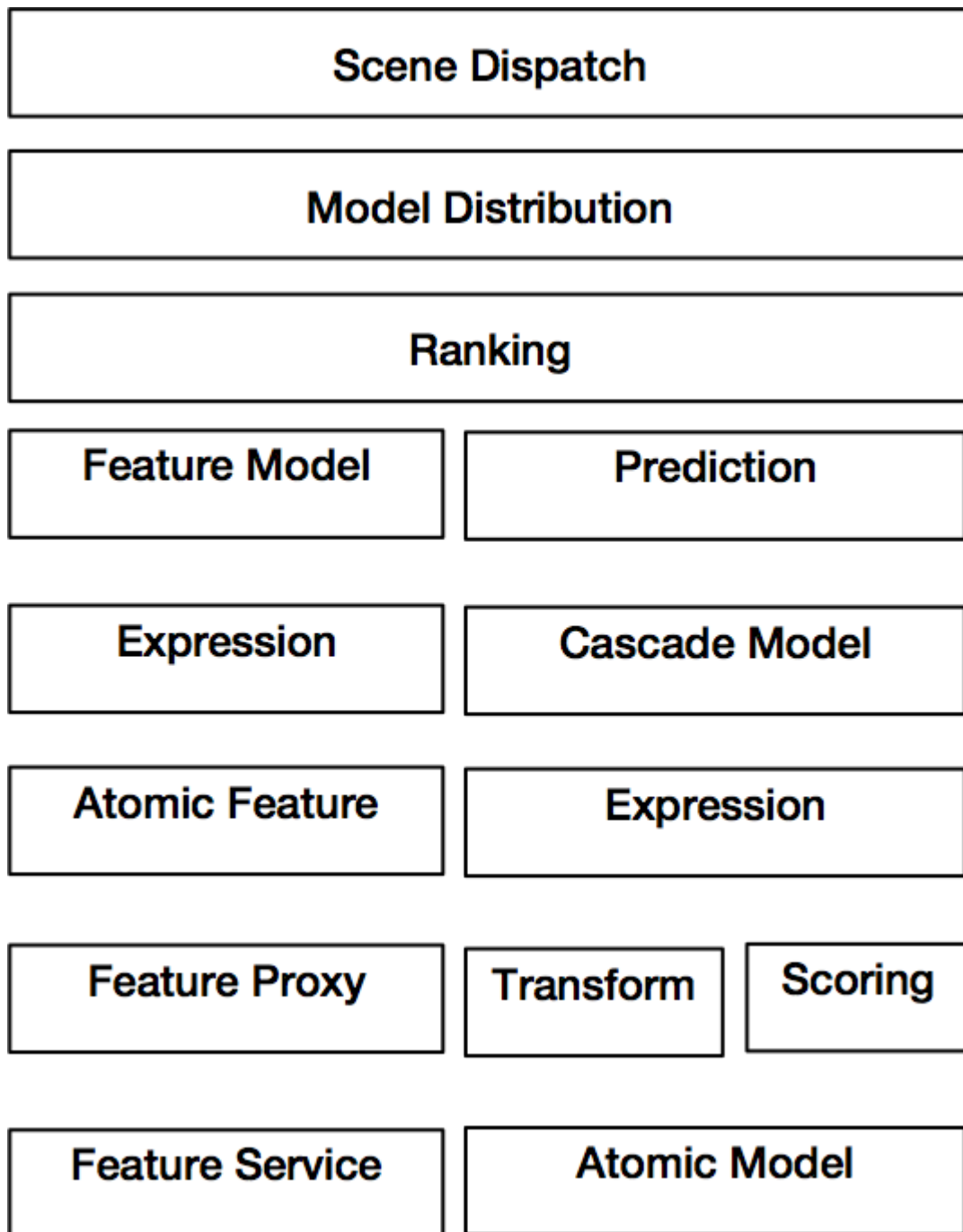
- i) 列表类特征。一次请求要求返回实体列表特征，即多个实体，每个实体多个特征。这种特征建议采用内存列表服务，一次性返回所有请求实体的所有特征。避免多次请求，从而导致数量暴增，造成系统雪崩。
- ii) 实体特征。一次请求返回单实体的多个特征。建议采用 Redis、Tair 等支持多级的 Key-Value 服务中间件提供服务。
- iii) 上下文特征。包括召回静态分、城市、Query 特征等。这些特征直接放在请求内存里面。
- iv) 相似性特征。有些特征是通过计算个体和列表、列表和列表的相似度而得到的。建议提供单独的内存计算服务，避免这类特征的计算影响在线排序性能。本质上这是一种计算转移的设计。

### 3.7 在线排序分层模型

如下图所示，典型的排序流程包含六个步骤：场景分发（Scene Dispatch）、流量分配（Traffic Distribution）、召回（Recall）、特征抽取（Feature Retrieval）、预测（Prediction）、排序（Ranking）等等。

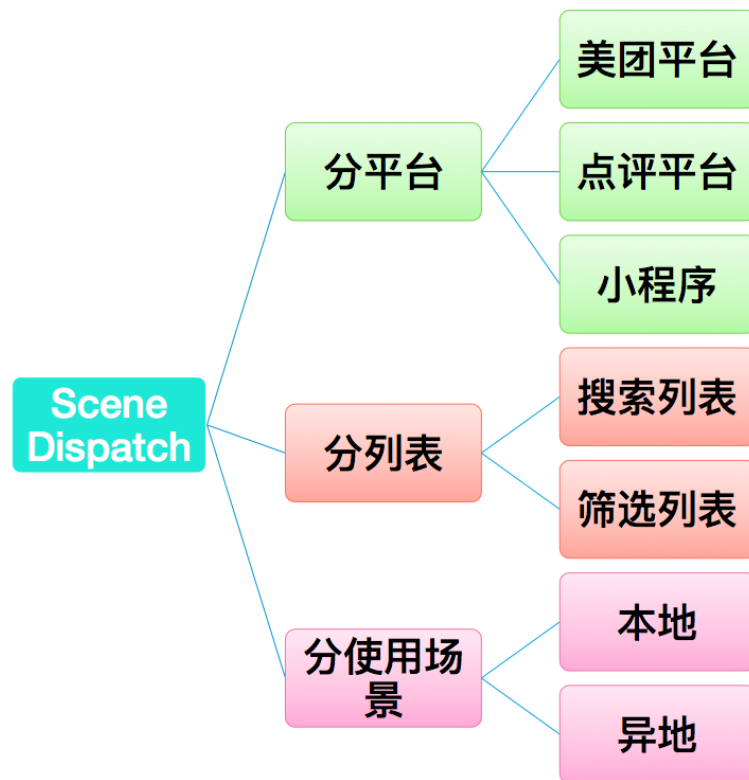


按照 DDD 的设计原则，我们设计了如下在线排序分层模型，包括：场景分发（Scene Dispatch）、模型分发（Model Distribution）、排序（Ranking）、特征管道（Feature Pipeline）、预测管道（Prediction Pipeline）。我们将逐一进行介绍。



### 3.7.1 场景分发 (Scene Dispatch)

场景分发一般是指业务类型的分发。对于美团点评而言包括：分平台、分列表、分使用场景等。如下图所示：



### 3.7.2 模型分发 (Model Distribution)

模型分发的目标是把在线流量分配给不同的实验模型，具体而言要实现三个功能：

为模型迭代提供在线流量，负责线上效果收集、验证等。

A/B 测试，确保不同模型之间流量的稳定、独立和互斥、确保效果归属唯一。

确保与其他层的实验流量的正交性。

流量的定义是模型分发的一个基础问题。典型的流量包括：访问、用户和设备。

如何让一个流量稳定地映射到特定模型上面，流量之间是否有级别？这些是模型分发需要重点解决的问题。

### 3.7.3 流量分桶原理

采用如下步骤将流量分配到具体模型上面去：

把所有流量分成 N 个桶。

每个具体的流量 Hash 到某个桶里面去。

给每个模型一定的配额，也就是每个策略模型占据对应比例的流量桶。

所有策略模型流量配额总和为 100%。

当流量和模型落到同一个桶的时候，该模型拥有该流量。

A	B	C	C	A	A	B	B
B	C	C	C	C	A	A	A
B	B	C	C	C	A	A	A
B	B	C	C	C	A	A	A

举个例子来说，如上图所示，所有流量分为 32 个桶，A、B、C 三个模型分别拥有 37.

5%、25%和 37.5%的配额。对应的，A、B、C 应该占据 12、8 和 12 个桶。

为了确保模型和流量的正交性，模型和流量的 Hash Key 采用不同的前缀。

### 3.7.4 流量分级

每个团队的模型分级策略并不相同，这里只给出一个建议模型流量分级：



基线流量。本流量用于与其他流量进行对比，以确定新模型的效果是否高于基准线，低于基准线的模型要快速下线。另外，主力流量相对基线流量的效果提升也是衡量算法团队贡献的重要指标。

实验流量。该流量主要用于新实验模型。该流量大小设计要注意两点：第一不能太大而伤害线上效果；第二不能太小，流量太小会导致方差太大，不利于做正确的效果判断。

潜力流量。如果实验流量在一定周期内效果比较好，可以升级到潜力流量。潜力流量主要是要解决实验流量方差大带来的问题。

主力流量。主力流量只有一个，即稳定运行效果最好的流量。如果某个潜力流量长期好于其他潜力流量和主力流量，就可以考虑把这个潜力流量升级为主力流量。

做实验的过程中，需要避免新实验流量对老模型流量的冲击。流量群体对于新模型会有一定的适应期，而适应期相对于稳定期的效果一般会差一点。如果因为新实验的上线而导致整个流量群体的模型都更改了，从统计学的角度讲，模型之间的对比关系没有变化。但这可能会影响整个大盘的效果，成本很高。

为了解决这个问题，我们的流量分桶模型优先为模型列表前面的模型分配流量，实验模型尽量放在列表尾端。这样实验模型的频繁上下线不影响主力和潜力流量的用户群体。当然当发生模型流量升级的时候，很多流量用户的服务模型都会更改。这种情况并不是问题，因为一方面我们在尝试让更多用户使用更好的模型，另一方面固定让一部分用户长期使用实验流量也是不公平的事情。

### 3.7.5 排序 (Ranking)

排序模块是特征模块和预测模块的容器，它的主要职责如下：

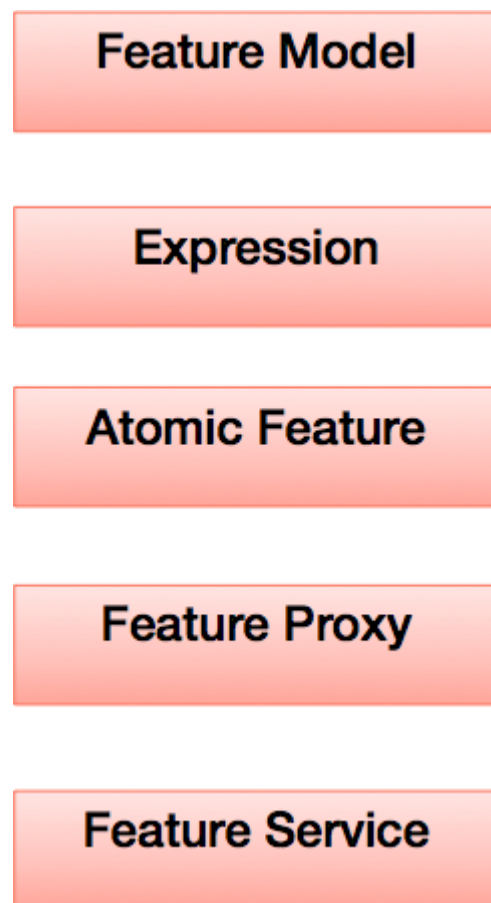
获取所有列表实体进行预测所需特征。

将特征交给预测模块进行预测。

对所有列表实体按照预测值进行排序。

### 3.7.6 特征管道 (Feature Pipeline)

特征管道包含特征模型 (Feature Model)、表达式 (Expression)、原子特征 (Atomic Feature)、特征服务代理 (Feature Proxy)、特征服务 (Feature Service)。如下图所示：



特征管道要解决两个核心问题：

给定特征名获取对应特征值。这个过程很复杂，要完成特征名->特征服务->特征类->特征值的转化过程。

特征算子问题。模型使用的特征往往是对多个原子特征进行复合运算后的结果。另外，特征离散化、归一化也是特征算子需要解决的问题。

完整的特征获取流程如下图所示，具体的流程如下：

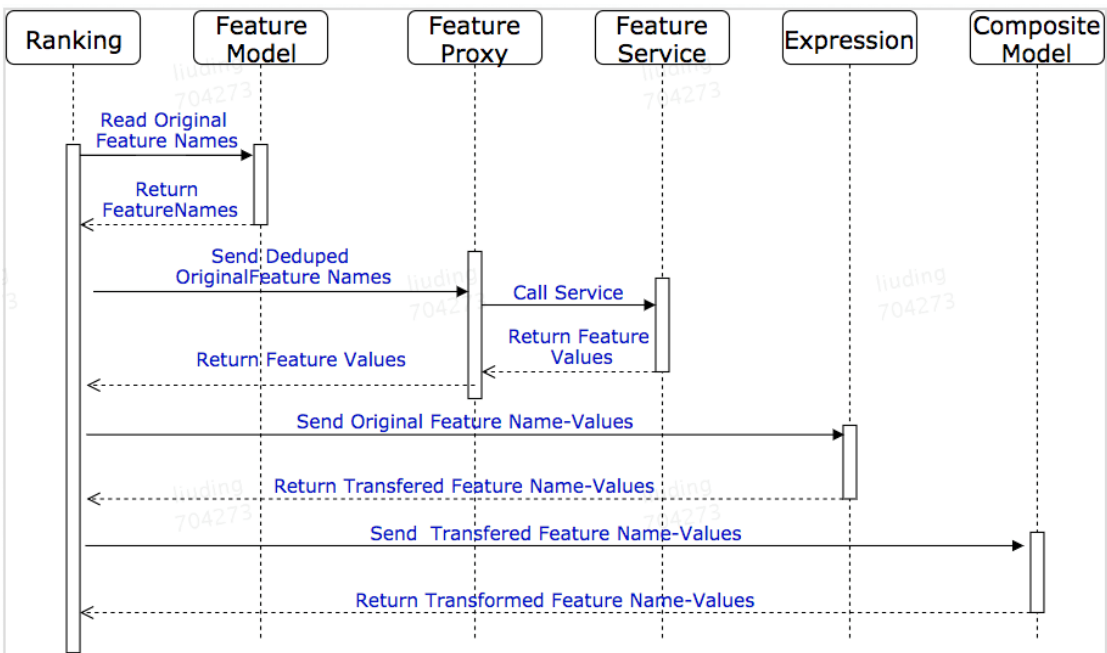
Ranking 模块从 FeatureModel 里面读取所有的原始特征名。

Ranking 将所有的原始特征名交给 Feature Proxy。

Feature Proxy 根据特征名的标识去调用对应的 Feature Service，并将原始特征值返回给 Ranking 模块。

Ranking 模块通过 Expression 将原始特征转化成复合特征。

Ranking 模块将所有特征交给级联模型做进一步的转换。



## 特征模型 (Feature Model)

我们把所有与特征获取和特征算子的相关信息都放在一个类里面，这个类就是 FeatureModel，定义如下：

```
//包含特征获取和特征算子计算所需的 meta 信息

public class FeatureModel {

    //这是真正用在 Prediction 里面的特征名

    private String featureName;

    //通过表达式将多种原子特征组合成复合特征。

    private IExpression expression;

    //这些特征名是真正交给特征服务代理(Feature Proxy)去从服务端获取特征值的
    特征名集合。

    private Set originalFeatureNames;

    //用于指示特征是否需要被级联模型转换

    private boolean isTransformedFeature;

    //是否为 one-hot 特征

    private boolean isOneHotIdFeature;

    //不同 one-hot 特征之间往往共享相同的原始特征，这个变量>用于标识原始特
    征名。

    private String oneHotIdKey;

    //表明本特征是否需要归一化

    private boolean isNormalized;

}
```

## 表达式 (Expression)

表达式的目的是为了将多种原始特征转换成一个新特征，或者对单个原始特征进行运算符转换。我们采用前缀法表达式 (Polish Notation) 来表示特征算子运算。例如表达式 $(5-6)*7$  的前缀表达式为  $* - 5 6 7$ 。

复合特征需要指定如下分隔符：

- ①复合特征前缀。区别于其他类型特征，我们以 “\$” 表示该特征为复合特征。
- ②表达式各元素之间的分隔符，采用 “\_” 来标识。
- ③用 “O” 表示运算符前缀。
- ④用 “C” 表示常数前缀。
- ⑤用 “V” 表示变量前缀。

例如：表达式  $v1 + 14.2 + (2*(v2+v3))$  将被表示为  $\$O+_O+_Vv1\_C14.2\_O*_C2\_O+_Vv2\_Vv3$

## 原子特征 (Atomic Feature)

原子特征（或者说原始特征）包含特征名和特征值两个部分。原子特征的读取需要由 4 种实体类共同完成：

POJO 用于存储特征原始值。例如 DealInfo 保存了所有与 Deal 实体相关的特征值，包括 Price、maxMealPerson、minMealPerson 等等。

ScoringValue 用于存储从 POJO 中返回的特征值。特征值包含三种基本类型，即数量型 (Quantity)、序数型 (Ordinal)、类别型 (Categorical)。

ScoreEnum 实现特征名到特征值的映射。每类原子特征对应于一个 ScoreEnum 类，特征名通过反射 (Reflection) 的方式来构建对应的 ScoreEnum 类。ScoreEnum 类和 POJO 一起共同实现特征值的读取。

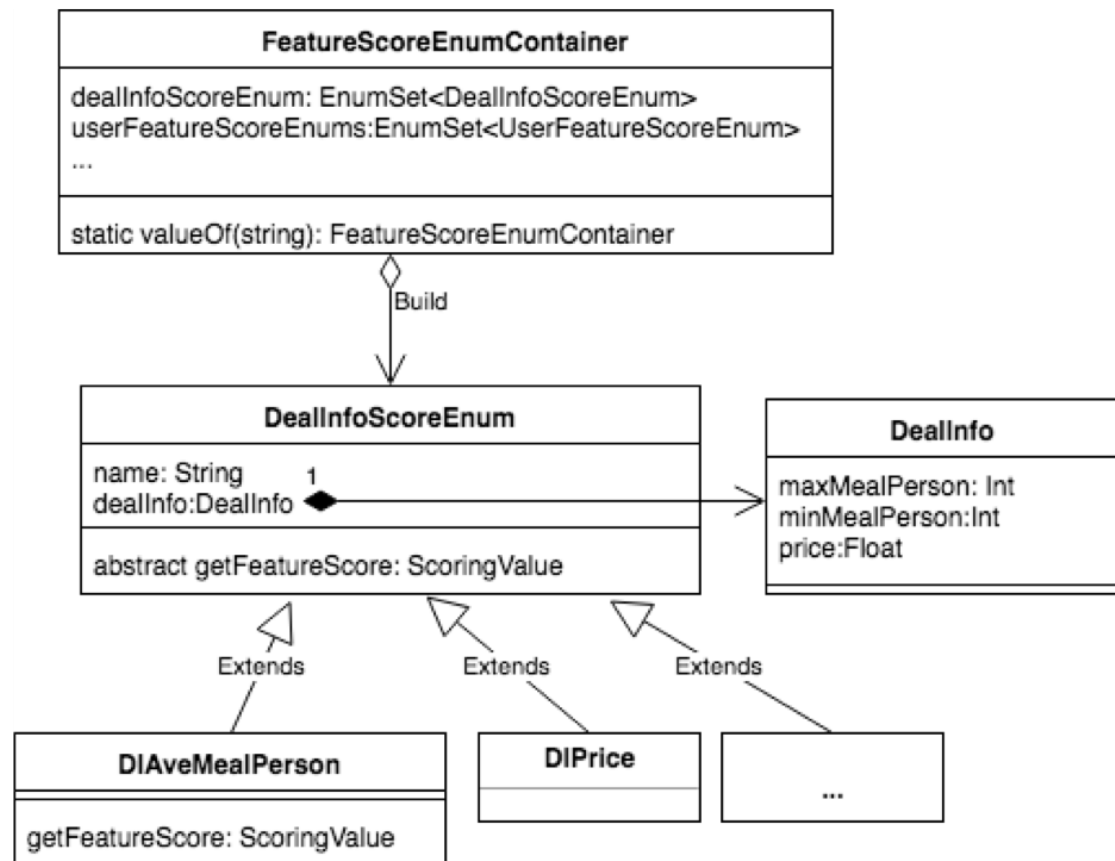
FeatureScoreEnumContainer 用于保存一个算法模型所需所有特征的 ScoreEnum。

一个典型的例子如下图所示：

DealInfo 是 POJO 类。

DealInfoScoreEnum 是一个 ScoreEnum 基类。对应于平均用餐人数特征、价格等特征，我们定义了 DIAveMealPerson 和 DIPrice 的具体 ScoreEnum 类。

FeatureScoreEnumContainer 用于存储某个模型的所有特征的 ScoreEnum。



复杂系统设计需要充分的利用语言特性和设计模式。建议的优化点有三个：

为每个原子特征定义一个 `ScoreEnum` 类会导致类数量暴增。优化方式是 `ScoreEnum` 基类定义为 `Enum` 类型，每个具体特征类为一个枚举值，枚举值继承并实现枚举类的方法。  
**FeatureScoreEnumContainer** 采用 **Build** 设计模式将一个算法模型的所需特征转换成 `ScoreEnum` 集合。

`ScoreEnum` 从 `POJO` 类中读取具体特征值采用的是 **Command** 模式。

这里稍微介绍一下 **Command** 设计模式。**Command** 模式的核心思想是需求方只要求拿到相关信息，不关心谁提供以及怎么提供。具体的提供方接受需求方的需求，负责将结果交给需求方。

在特征读取里面，需求方是模型，模型仅提供一个特征名（FeatureName），不关心怎么读取对应的特征值。具体的 ScoreEnum 类是具体的提供方，具体的 ScoreEnum 从 POJO 里面读取特定的特征值，并转换成 ScoringValue 交给模型。

### 特征服务代理（Feature Proxy）

特征服务代理负责远程特征获取实施，具体的过程包括：

每一大类特征或者一种特征服务有一个 FeatureProxy，具体的 FeatureProxy 负责向特征服务发起请求并获取 POJO 类。

所有的 FeatureProxy 注册到 FeatureServiceContainer 类。

在具体的一次特征获取中，FeatureServiceContainer 根据 FeatureName 的前缀负责将特征获取分配到不同的 FeatureProxy 类里面。

FeatureProxy 根据指定的实体 ID 列表和特征名从特征服务中读取 POJO 列表。只有对应 ID 的指定特征名（keys）的特征值才会被赋值给 POJO。这就最大限度地降低了网络读取的成本。

### 3.7.7 预测管道（Prediction Pipeline）

预测管道包含：预测（Prediction）、级联模型（Cascade Model）、表达式（Expression）、特征转换（Transform）、计分（Scoring）和原子模型（Atomic Model）。

#### 预测（Prediction）



预测本质上是对模型的封装。它负责将每个列表实体的特征转化成模型需要的输入格式，让模型进行预测。

## 级联模型 (Cascade Model)

我们构建级联模型主要是基于两方面的观察：

基于 Facebook 的《Practical Lessons from Predicting Clicks on Ads at Facebook》的 Xgboost+LR 以及最近很热门的 Wide&Deep 表明，对一些特征，特别是 ID 类特征通过树模型或者 NN 模型进行转化，把转化后的值做为特征值交给预测模型进行预测，往往会能实现更好的效果。

有些训练目标是复合目标，每个子目标需要通过不同的模型进行预测。这些子目标之间通过一个简单的表达式计算出最终的预测结果。

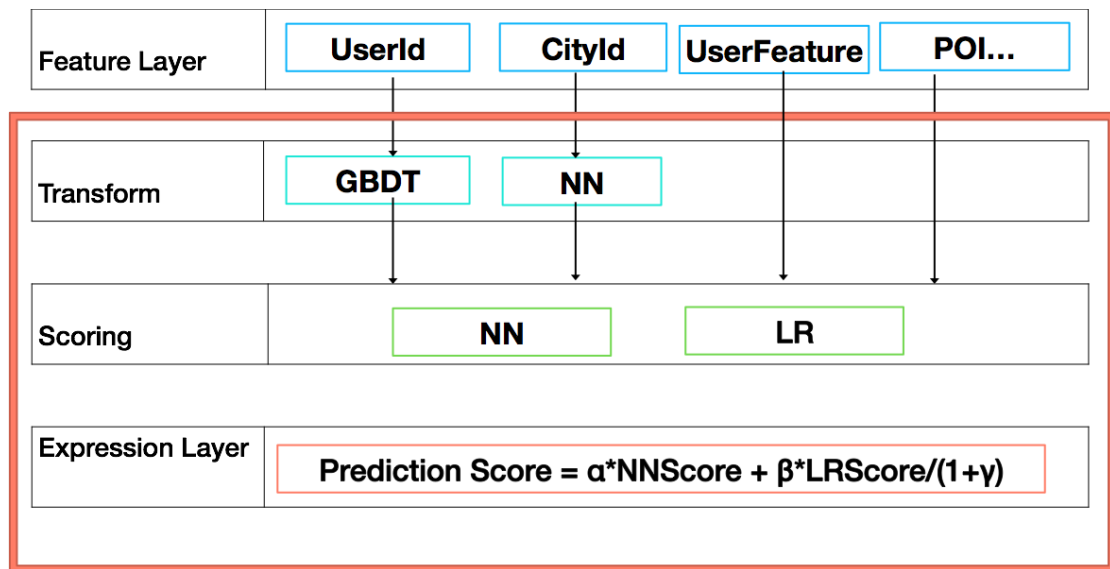
举例如下图所示，我们自上而下进行讲解：

该模型有 UserId、CityId、UserFeature、POI 等特征。

UserId 和 CityId 特征分别通过 GBDT 和 NN 模型进行转换 (Transform) 。

转换后的特征和 UserFeature、POI 等原始特征一起交给 NN 和 LR 模型进行算分 (Scoring) 。

最终的预测分值通过表达式  $\text{Prediction Score} = \alpha * \text{NNScore} + \beta * \text{LRScore} / (1 + \gamma)$  来完成。表达式中的  $\alpha$ 、 $\beta$  和  $\gamma$  是事先设置好的值。



原子模型 (Atomic Model)

在这里原子模型指的是一种原子计算拓扑结构，比如线性模型、树模型和网络模型。

常用的模型像 Logistic Regression 和 Linear Regression 都是线性模型。GBDT、Random Forest 都是树模型。MLP、CNN、RNN 都是网络模型。

这里定义的原子模型主要的目的是为了工程实施的便利。一个模型被认定为原子模型有如下两个原因：

该模型经常做为独立预测模型被使用。

该模型有比较完整的实现代码。

#### 四、总结

本文总结了作者在美团点评解决到店餐饮个性化信息展示的相关经验，从算法和架构两方面进行阐述。在算法部分，文章采用通俗的例子和类比方式进行讲解，希望非算法工程师能够理解关键的算法概念。架构部分比较详细地阐述了到店餐饮的排序架构。

根据我们所掌握的知识，特征治理和召回治理的思路是一种全新的视角，这对于架构排序系统设计有很大的帮助。这种思考方式同样也适用于其他领域模型的构建。与 Google 提供的经典 Two-Phase Scheme 架构相比，在线排序分层模型提供了更细颗粒度的抽象原型。该原型细致的阐述了包括分流、A/B 测试、特征获取、特征算子、级联模型等一系列经典排序架构问题。同时该原型模型由于采用了分层和层内功能聚焦的思路，所以它比较完美地体现了 DDD 的三大设计原则，即领域聚焦、边界清晰、持续集成。

## 作者简介

刘丁，目前负责到店餐饮算法策略方向，推进 AI 在到店餐饮各领域的应用。2014 年加入美团，先后负责美团推荐系统、智能筛选系统架构、美团广告系统的架构和上线、完成美团广告运营平台的搭建。曾就职于 Amazon、TripAdvisor 等公司。

## 五、参考文章：

[1]Gamma E, Helm R, Johnson R, et al. Design Patterns-Elements of Reusable Object-Oriented Software. Machinery Industry, 2003

[2]Wikipedia, Learning to rank.

[3]Wikipedia, Machine learning.

[4]Wikipedia, Precision and recall.

[5]Wikipedia,Discounted cumulative gain.

[6]Wikipedia,Domain-driven design.

[7]Wikipedia,Elasticsearch.

[8]Wikipedia,k-d tree.

[9]百度百科,太阳历.

[10]百度百科,阴历.

[11]Xinran H, Junfeng P, Ou J, et al. Practical Lessons from Predicting Clicks on Ads at Facebook

[12]Olivier C, Donald M, Ya Z, Pierre G. Expected Reciprocal Rank for Graded Relevance

[13]Heng-Tze C, Levent K, et al. Wide & Deep Learning for Recommender Systems

## 简述推荐系统的演进史

解析：

分类目录(1990):覆盖少量热门网站(Hao123 Yahoo)

搜索引擎(2000):通过搜索词明确需求(Google Baidu)

推荐系统(2010):不需要用户提供明确的需求,通过分析用户的历史行为给用户的兴趣进行建模,从而主动给用户推荐能够满足他们兴趣和需求的信息

# 推荐系统评估(从用户\内容提供方\网站角度)

解析：

从用户角度:

满足需求

获取快乐

扩展视野

内容提供方:

获取长尾流量

获得互动和认可

获得收益

网站:

留住用户

实现商业目标

## 推荐系统常用评估指标

解析：

准确性

满意度

覆盖率

多样性

新颖性

惊喜度

信任度

实时性

鲁棒性

可扩展性

商业目标

用户留存

## 推荐系统准确性(学术界)的评估函数

解析：

- 评分预测

- RMSE: 均方根误差

$$RMSE = \sqrt{\frac{\sum_{u,i \in T} (r_{ui} - \hat{r}_{ui})^2}{|T|}}$$

- MAE: 平均绝对误差

$$MAE = \frac{\sum_{u,i \in T} |r_{ui} - \hat{r}_{ui}|}{|T|}$$

- topN推荐

- Recall: 召回率

$$Recall = \frac{\sum_{\mu \in U} |R(\mu) \cap T(\mu)|}{\sum_{\mu \in U} |T(\mu)|}$$

- Precision: 准确率

$$Precision = \frac{\sum_{\mu \in U} |R(u) \cap T(u)|}{\sum_{\mu \in U} |R(u)|}$$

# 推荐系统多样性&新颖性&惊喜性

解析：

多样性: 推荐列表中两两物品的不相似性

新颖性: 未曾关注的类别\作者; 推荐结果的平均流行度

惊喜性: 历史不相似(惊)但很满意(喜)

## 解释 Exploitation & Exploration

解析：

Exploitation: 选择现在可能最佳的方案(可能会让推荐的结果范围越来越小)

Exploration: 选择现在不确定的一些方案,但未来可能会有高收益的方案(会让推荐的面越来越广)

在做两类决策的过程中,不断更新对所有决策的不确定性的认知,优化长期的目标函数

## Bandit 算法 - 原理

解析：

- Thompson Sampling: 每个臂维护一个beta(wins,lose)分布,每次用现有的beta分布产生一个随机数,选择随机数最大的臂.
  - 假设每个臂是否产生收益,其背后有一个概率分布,产生收益的概率为 P
  - 我们不断地实验,去估计出一个置信度较高的“概率 P 的概率分布”就能近似解决这个问题了
  - 假设概率 P 的概率分布符合 beta(wins,lose)分布,它有两个参数: wins, lose
  - 每个臂都维护一个beta分布的参数,每次试验后,选中一个臂,摇一下,有收益则该臂的 wins 增加1\*\*,否则该臂的 lose 增加1\*\*
  - 每次选择臂的方式是: 用每个臂现有的beta分布产生一个随机数b,选择所有臂产生的随机数中最大的那个臂去摇
- Upper Confidence Bound(置信区间上界): 均值越大,标准差越小,被选中的概率会越来越大
  - 初始化: 先对每一个臂都试一遍
  - 按照如下公式计算每个臂的分数,然后选择分数最大的臂作为选择

$$\bar{x}_j(t) + \sqrt{\frac{2 \ln t}{T_{j,t}}}$$

- 观察选择结果,更新 $t$ 和 $T_{j,t}$ ,其中加号前面是这个臂到目前的 收益均值,后面的叫做 bonus,本质上是均值的标准差, $t$ 是目前的试验次数, $T_{j,t}$ 是这个臂被试次数
- Epsilon-Greedy: 以1-epsilon的概率选取当前收益最大的臂,以epsilon的概率随机选取一个臂. epsilon 的值可以控制对 Exploit 和 Explore 的偏好程度. 越接近0, 越靠近 Exploit
  - 选择一个(0,1)之间较小的数作为 epsilon
  - 每次以概率 epsilon 做一件事: 所有臂中随机选择一个
  - 每次以概率 1-epsilon 选择截止到目前,平均收益最大的那个臂
- 朴素Bandit算法: 先随机试若干次,计算每个臂的平均收益,一直选均值最大的那个臂.

## 什么是多层重叠试验框架?

解析:

保留单层实验框架易用,快速的优点的同时,增加可扩展性,灵活性,健壮性.

核心思路: 将参数划分到 N 个子集,每个子集都关联一个实验层,每个请求会被 N 个实验处理,同一个参数不能出现在多个层中

## 预训练方法 BERT 和 OpenAI GPT 有什么区别?

解析:

1.GPT 在 BooksCorpus(800M 单词)训练; BERT 在 BooksCorpus(800M 单词)和维基百科(2,500M 单词)训练

2.GPT 使用一种句子分隔符([SEP])和分类符词块([CLS]), 它们仅在微调时引入; BERT 在预训练期间学习[SEP], [CLS]和句子 A/B 嵌入

3.GPT 用一个批量 32,000 单词训练 1M 步; BERT 用一个批量 128,000 单词训练 1M 步



4.GPT 对所有微调实验使用的  $5e-5$  相同学习率；BERT 选择特定于任务的微调学习率，在开发集表现最佳

## 对比 BERT、OpenAI GPT、ELMo 预训练模型架构间差异

解析：

BERT 使用双向变换器，模型的表示在所有层中，共同依赖于左右两侧的上下文

OpenAI GPT 使用从左到右的变换器，利用了 Transformer 的编码器作为语言模型进行预训练的，之后特定的自然语言处理任务在其基础上进行微调即可。

ELMo 使用独立训练的从左到右和从右到左 LSTM 级联来生成下游任务的特征。是一种双层双向的 LSTM 结构，其训练的语言模型可以学习到句子左右两边的上下文信息，但此处所谓的上下文信息并不是真正意义上的上下文。

三种模型中只有 BERT 表征基于所有层左右两侧语境。

## GCN 方法分为哪两类？

解析：

基于谱的方法，通过从图信号处理的角度引入滤波器来定义图卷积，其中图卷积运算被解释为从图信号中去除噪声，包括 Spectral CNN、Chebyshev Spectral CNN (ChebN

et)、First order of ChebNet (1stChebNet) 和 Adaptive Graph Convolution Network (AGCN)

基于空间的方法将图卷积表征为聚合来自近邻的特征信息。虽然 GCN 在节点级别上运行，但是图池化模块可以与 GCN 层交替，将图粗粒化为高级子结构，包括 Recurrent-based Spatial GCN 和 Composition Based Spatial GCN

## 什么是图卷积神经网络？

解析：

图卷积神经网络（Graph Convolutional Network）是一种能对图数据进行深度学习的方法

图卷积算子：

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in N_i} \frac{1}{c_{ij}} h_j^l w_{R_j}^l\right)$$

$h_i^l$  节点*i*在第*l*层的特征表达

$c_{ij}$ 归一化因子

$N_i$ 节点*i*的邻居

$R_i$ 节点*i*的类型

$w_{R_i}$   $R_i$ 类型节点的变换权重参数

## 如何理解图卷积算法？

解析：

1) 发射（send）每一个节点将自身的特征信息经过变换后发送给邻居节点。这一步是在对节点的特征信息进行抽取变换

2) 接收 (receive) 每个节点将邻居节点的特征信息聚集起来。这一步是在对节点的局部结构信息进行融合

3) 变换 (transform) 把前面的信息聚集之后做非线性变换, 增加模型的表达能力

## GCN 有哪些特征?

解析:

1.GCN 是对卷积神经网络在 graph domain 上的自然推广

2.它能同时对节点特征信息与结构信息进行端对端学习, 是目前对图数据学习任务的最佳选择

3.图卷积适用性极广, 适用于任意拓扑结构的节点与图

4.在节点分类与边预测等任务上, 在公开数据集上效果要远远优于其他方法

**已知基于数据驱动的机器学习和优化技术在单场景内的 A/B 测试上, 点击率、转化率、成交额、单价都取得了不错的效果。但是, 目前各个场景之间是完全独立优化的, 这样会带来哪些比较严重的问题 ?**

解析:

1.不同场景的商品排序仅考虑自身，会导致 用户的购物体验是不连贯或者雷同的 。例

如：从冰箱的详情页进入店铺，却展示手机；各个场景都展现趋同，都包含太多的 U2I  
(点击或成交过的商品) ；

2.多场景之间是博弈（竞争）关系， 期望每个场景的提升带来整体提升这一点是无法保证的 。很有可能一个场景的提升会导致其他场景的下降，更可怕的是某个场景带来的提升甚至小于其他场景更大的下降。这并非是不可能的，这种情况下，单场景的 A/B 测试就显得没那么有意义，单场景的优化也会存在明显的问题。

## 什么是多场景联合排序算法 ？

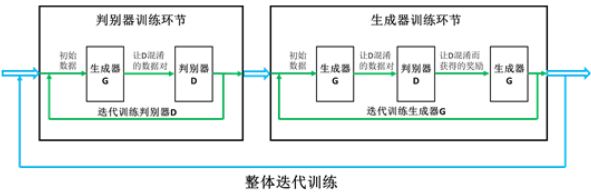
解析：

多场景联合排序算法，旨在提升整体指标。我们将多场景的排序问题看成一个完全合作的、部分可观测的多智能体序列决策问题，利用 Multi-Agent Reinforcement Learning 的方法来尝试着对问题进行建模。该模型以各个场景为 Agent，让各个场景不同的排序策略共享同一个目标，同时在一个场景的排序结果会考虑该用户在其他场景的行为和反馈。这样使得各个场景的排序策略由独立转变为合作与共赢。由于我们想要使用用户在所有场景的行为，而 DRQN 中的 RNN 网络可以记住历史信息，同时利用 DPG 对连续状态与连续动作空间进行探索，因此我们算法取名 MA-RDPG

## IRGAN 框架具体是什么？

解析：

IRGAN 的训练数据必须满足一定的格式要求。比如，每条训练数据要包括一个查询语句 query 以及多个（个数大于 2）与查询相关的网页数据。此外，每条训练数据还要在多个网页数据中标明哪个与本条 query 最为相关。记与当前查询最为相关的网页为 doc<sub>+</sub>，其他网页信息为 doc<sub>-</sub>，那么训练集的格式即为 (query, doc<sub>+</sub>, doc<sub>-</sub>, doc<sub>-</sub>, ... , doc<sub>-</sub>)。



如图，IRGAN 的训练过程可分为判别器的训练和生成器的训练两部分。在训练判别器时，生成器首先会拿到形如 (query, doc<sub>+</sub>, doc<sub>-</sub>, doc<sub>-</sub>, ... , doc<sub>-</sub>) 的初始数据。为了最大程度的欺骗判别器，生成器会从诸多 doc<sub>-</sub> 中挑选一个它认为与查询最相关的网页数据，并将这个 doc<sub>-</sub> 与 doc<sub>+</sub>（实际上最相关的网页数据）拼在一起形成新的数据对 (query, doc<sub>+</sub>, doc<sub>-</sub>)。判别器会对生成器产生的数据对做出选择：究竟哪个网页与查询最相关。如果判别器选择错误（换言之，被生成器骗到了），那么它将会被惩罚，从而得到训练。

# 推荐页与搜索页特性有什么不同？

解析：

搜索带着 query 来的，结果与之相关性越高越好，不用太关心结果的多样性

推荐页用户没有明确的目的，但是有兴趣偏好和对结果的多样性需求，推荐既要准确又要多样化。

# 推荐系统常用的评价指标有哪些？

解析：

- 准确率: 准确率表示的是分类正确的样本数占样本总数的比例

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$$

- 精确率: 表示预测结果中, 预测为正样本的样本中, 正确预测为正样本的概率

$$Precision = \frac{TP}{TP+FP}$$

- 召回率: 表示在原始样本的正样本中, 最后被正确预测为正样本的概率

$$Recall = \frac{TP}{TP+FN}$$

- F1值: 折中精确率和召回率的结果

$$F_1\text{-score} = \frac{2 * Recall * Precision}{Recall + Precision}$$

- AUC: 横轴为FP, 纵轴为TP
- ROC: AUC底线的面积
- Hit Ratio: 一种常用的衡量召回率的指标

$$HR@K = \frac{\text{Number of Hits@K}}{GT}$$

分母是所有的测试集合, 分子是每个用户top-K推荐列表中属于测试集合的个数的总和。

- AP: 用户u, 给他推荐一些物品, 那么 u 的平均准确率定义为:

$$\frac{1}{|\Omega_u|} \sum_{i \in \Omega_u} \frac{\sum_{j \in \Omega_u} h(p_{uj} - p_{ui}) + 1}{p_{ui}}$$

$\Omega_u$  表示ground-truth的结果,  $p_{ui}$  表示物品在推荐列表的位置,  $p_{uj} < p_{ui}$  表示j在物品列表中排在i之前

- MAP: 所有用户 u 的AP再取均值(mean)

$$MAP = \frac{\sum_{u \in U} AP_u}{|U|}$$

- CG(Cummulative Gain), 在推荐系统中, CG即将每个推荐结果相关性(relevance)的分值累加后作为整个推荐列表(list)的得分

$$CG_k = \sum_{i=1}^k rel_i$$

,  $rel_i$  表示处于位置 i 的推荐结果的相关性, k 表示所要考察的推荐列表的大小

- DCG: CG的一个缺点是没有考虑每个推荐结果处于不同位置对整个推荐效果的影响, 例如我们总是希望相关性高的结果应排在前面。显然, 如果相关性低的结果排在靠前的位置会严重影响用户体验, 所以在CG的基础上引入位置影响因素, 即DCG(Discounted Cummulative Gain), "Discounted"有打折, 折扣的意思, 这里指的是对于排名靠后推荐结果的推荐效果进行"打折处理":

$$DCG_k = \sum_{i=1}^k \frac{2^{rel_i}}{\log_2(i+1)}$$

- NDCG: DCG仍然有其局限之处, 即不同的推荐列表之间, 很难进行横向的评估。那么不同用户的推荐列表的评估分数就需要进行归一化, 也即NDCG(Normalized Discounted Cummulative Gain),

$$NDCG@k = \frac{\sum_{u \in \mathcal{V}^{te}} NDCG_u @ k}{|\mathcal{V}^{te}|}$$

- MRR:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$$

|Q|是用户的个数，rank<sub>i</sub>是对于第i个用户，推荐列表中第一个在ground-truth结果中的item所在的排列位置

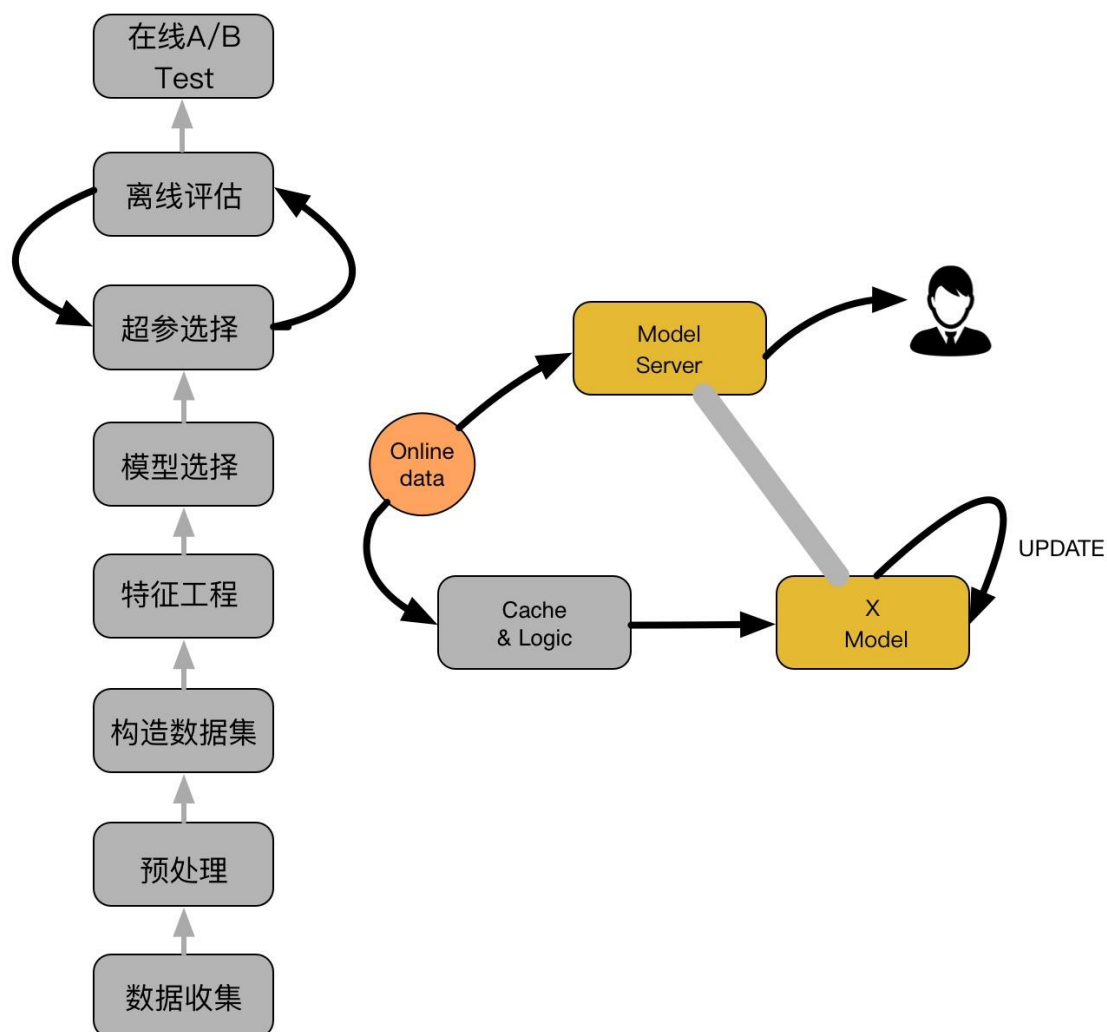
- ILS:衡量推荐列表多样性的指标

$$ILS(L) = \frac{\sum_{b_i \in L} \sum_{b_j \in L, b_j \neq b_i} S(b_i, b_j)}{\sum_{b_i \in L} \sum_{b_j \in L, b_j \neq b_i} 1}$$

如果S(b<sub>i</sub>,b<sub>j</sub>)计算的是b<sub>i</sub>和b<sub>j</sub>两个物品的相似性，如果推荐列表中物品越不相似，ILS越小，那么推荐结果的多样性越好

## 请画出点击率预估流程图，分为那两部分？

解析：



主要包括两大部分：离线部分、在线部分，其中离线部分目标主要是训练出可用模型，而在线部分则考虑模型上线后，性能可能随时间而出现下降，弱出现这种情况，可选择使用 Online-Learning 来在线更新模型

## 使用 FFM 有哪些需要注意的地方

解析：

样本归一化。即 `pa.norm` 为真，对样本进行归一化，否则容易造成数据溢出，梯度计算失败

特征归一化。为了消除不同特征取值范围不同，量纲不同造成的问题，需要对特征进行归一化

Early stopping。一定要设置早停策略，FFM 很容易过拟合

省略零值特征。零值特征对模型没有任何贡献，无论是 1 次项还是 2 次项都为 0。这也是系数样本采用 FFM 的显著优势

## 什么是 DSSM?有什么优缺点?

解析：

DSSM (Deep Structured Semantic Models) 的原理很简单，通过搜索引擎里 Query 和 Title 的海量的点击曝光日志，用 DNN 把 Query 和 Title 表达为低维语义向量，并通过 cosine 距离来计算两个语义向量的距离，最终训练出语义相似度模型。该模型既可以用来预测两个句子的语义相似度，又可以获得某句子的低维语义向量表达。



优点:DSSM 用字向量作为输入既可以减少切词的依赖, 又可以提高模型的泛化能力, 因为每个汉字所能表达的语义是可以复用的。另一方面, 传统的输入层是用 Embedding 的方式 (如 Word2Vec 的词向量) 或者主题模型的方式 (如 LDA 的主题向量) 来直接做词的映射, 再把各个词的向量累加或者拼接起来, 由于 Word2Vec 和 LDA 都是无监督的训练, 这样会给整个模型引入误差, DSSM 采用统一的有监督训练, 不需要在中间过程做无监督模型的映射, 因此精准度会比较高。

缺点: DSSM 采用词袋模型 (BOW), 因此丧失了语序信息和上下文信息。另一方面, DSSM 采用弱监督、端到端的模型, 预测结果不可控。

## 什么是 wide&deep 模型?

解析:

wide 部分就是传统的 LR 模型, LR 模型简单、快速并且模型具有可解释, 有着很好的拟合能力, 但是 LR 模型是线性模型, 表达能力有限, 泛化能力较弱, 需要做好特征工程, 尤其需要交叉特征, 才能取得一个良好的效果

deep 部分就是 DNN, DNN 模型不需要做太精细的特征工程, 就可以取得很好的效果, 已经在很多领域开始应用了, DNN 可以自动交叉特征, 学习到特征之间的相互作用, 尤其是可以学到高阶特征交互, 具有很好的泛化能力。另外, DNN 通过增加 embedding 层, 可以有效的解决稀疏数据特征的问题, 防止特征爆炸。

wide&deep 就是将 LR 和 DNN 结合起来, 将两者的结果组合进行输出。

# Join training 和 ensemble training 有什么区别？

解析：

(1) ensemble 中每个模型需要单独训练，并且各个模型之间是相互独立的，模型之间互不感知，当预测样本时，每个模型的结果用于投票，最后选择得票最多的结果。而 join train 这种方式模型之间不是独立的，是相互影响的，可以同时优化模型的参数。

(2) ensemble 的方式中往往要求存在很多模型，这样就需要更多的数据集和数据特征，才能取得比较好的效果，模型的增多导致难以训练，不利于迭代。而在 wide&deep 中，只需要两个模型，训练简单，可以很快的迭代模型。

## 简述 DeepFM 模型？

解析：

DeepFM模型由两部分组成：

- 神经网络部分，负责低阶特征提取
- 因子分解机部分，负责高阶特征的提取

这两部分共享同样的输入，预测结果可以写为：

$$\hat{y} = \text{sigmoid}(y_{FM} + y_{DNN})$$

FM部分是一个因子分解机,深度部分是一个前馈神经网络。

# Collaborative Knowledge Base Embedding 使用哪三种知识的学习？

解析：

结构化知识学习：TransR。TransR 是一种基于距离的翻译模型，可以学习得到知识实体的向量表示

文本知识学习：去噪自编码器。去噪自编码器可以学习得到文本的一种泛化能力较强的向量表示

图像知识学习：卷积-反卷积自编码器。卷积-反卷积自编码器可以得到图像的一种泛化能力较强的向量表示

## 怎样将知识图谱引入推荐系统 ？

解析：

基于特征的知识图谱辅助推荐，核心是知识图谱特征学习的引入。一般而言，知识图谱是一个由三元组<头节点，关系，尾节点>组成的异构网络。由于知识图谱天然的高维性和异构性，首先使用知识图谱特征学习对其进行处理，从而得到实体和关系的低维稠密向量表示。这些低维的向量表示可以较为自然地与推荐系统进行结合和交互。

基于结构的推荐模型，更加直接地使用知识图谱的结构特征。具体来说，对于知识图谱中的每一个实体，我们都进行宽度优先搜索来获取其在知识图谱中的多跳关联实体从中得到推荐结果。

# 普通的逻辑回归能否用于大规模的广告点击率预估，为什么？

解析：

不能

第一，数据量太大。传统的逻辑回归参数训练过程都依靠牛顿法（Newton's Method）

或者 L-BFGS 等算法。这些算法并不太容易在大规模数据上得以处理。

第二，不太容易得到比较稀疏（Sparse）的答案（Solution）。也就是说，虽然数据中特征的总数很多，但是对于单个数据点来说，有效特征是有限而且稀疏的。

# FTRL 在准备训练数据（特征工程）和训练模型时有哪些 trick ？

解析：

（1）特征工程

特征预处理：ID 化、离散化、归一化等；

特征选择：方差、变异系数、相关系数、Information Gain、Information Gain-Ratio、IV 值等；

特征交叉和组合特征：根据特征具有的业务属性特征交叉，利用 FM 算法、GBDT 算法做高维组合特征等。

## (2) Subsampling Training Data

正样本全采样，负样本使用一个比例  $r$  采样，并在模型训练的时候，对负样本的更新梯度乘以权重  $1/r$ ;

负采样的方式：随机负采样、Negative sampling、邻近负采样、skip above 负采样等。

## (3) 在线丢弃训练数据中很少出现的特征(probabilistic feature inclusion)

Poisson Inclusion：对某一维度特征所来的训练样本，以  $p$  的概率接受并更新模型;

Bloom Filter Inclusion：用 bloom filter 从概率上做某一特征出现  $k$  次才更新。

# 阿里最新开源的 X-Deep Learning 为 Online Learning 提供了哪些解决方案？

解析：

去 ID 化的稀疏特征学习：传统的机器学习框架一般要求对稀疏特征进行 ID 化表征（从 0 开始紧凑编码），以此来保证训练的高效性。XDL 则允许直接以原始的特征进行训练，大幅简化了特征工程的复杂度，极大地增加了全链路数据处理效率，这一特性在实时在线学习场景下显得更加有意义。

实时特征频控：用户可以设置一个特征过滤的阈值，例如出现次数大于 N 次的特征才纳入模型训练，系统会自动的采用自动概率丢弃的算法进行特征选择，这样可以大幅降低无效超低频特征在模型中的空间占用。

过期特征淘汰：长周期的在线学习时，用户也可以通过打开过期特征淘汰功能，系统会自动的对影响力弱且长周期没有碰触到的特征参数进行自动淘汰

## 特征交叉(特征组合)方式有哪些？

解析：

### 1.Dense 特征组合

将一个特征与其本身或其他特征相乘(称为特征组合)(二阶或者高阶)；

两个特征相除；

对连续特征进行分桶,以分为多个区间分箱。

### 2.ID 特征之间的组合

笛卡尔积:假如拥有一个特征 A,A 有两个可能值{A1,A2}。拥有一个特征 B,存在{B1,B2}等可能值。然后,A&B 之间的交叉特征

如下:{(A1,B1),(A1,B2),(A2,B1),(A2,B2)},比如经纬度,一个更好地诠释好的交叉特征的实例是类似于(经度,纬度)。一

个相同的经度对应了地图上很多的地方,纬度也是一样。但是一旦你将经度和纬度组合到一起,它们就代表了地理上特定的

一块区域,区域中每一部分是拥有着类似的特性。

# 特征选择的方法有哪些？

解析：

Filter:过滤法,按照发散性或者相关性对各个特征进行行评分,设定阈值或者待选择阈值的个数,选择特征。

Wrapper:包装法,根据目标函数(通常是预测效果评分),每次选择若干特征,或者排除若干特征。

Embedded:嵌入法,先使用某些机器学习的算法和模型进行训练,得到各个特征的权值系数,根据系数从大到小选择特征。类似于 Filter 方法,但是是通过训练来确定特征的优劣。

## 简述 Multi-task learning(MLT)多任务学习

解析：

在机器学习中，我们通常关心优化某一特定指标，不管这个指标是一个标准值，还是企业 KPI。为了达到这个目标，我们训练单一模型或多个模型集合来完成指定得任务。然后，我们通过精细调参，来改进模型直至性能不再提升。尽管这样做可以针对一个任务得到一个可接受得性能，但是我们可能忽略了一些信息，这些信息有助于在我们关心的指标上做得更好。具体来说，这些信息就是相关任务的监督数据。通过在相关任务间共享表示信息，我们的模型在原始任务上泛化性能更好。这种方法称为多任务学习（Multi-Task Learning）

如何离线评价召回阶段各种模型算法的好坏？由于没有明确的召回预期值，所以无论 rmse 还是 auc 都不知道该怎么做？

解析：

答：召回最直接的评估就是召回率，也就是召回集里正样本的比例；也可以不同的召回算法+同一个排序算法，还是用排序之后的 AUC 和 RMSE 来评估。