

# 数据分析之推荐算法

原创 志扬工作室 志扬工作室 2019-10-26

**Start:** 关注本公众号后，可直接联系后台获取排版美化的详细文档！

**Hints:** 本篇文章所编纂的资料均来自网络，特此感谢参与奉献的有关人员。

## ○ 推荐算法：

推荐算法是计算机专业中的一种算法，通过一些数学算法，推测出用户可能喜欢的东西，目前应用推荐算法比较好的地方主要是网络，其中淘宝做的比较好。所谓推荐算法就是利用用户的一些行为，通过一些数学算法，推测出用户可能喜欢的东西。

## ○ 基于内容的推荐：

基于内容的推荐(Content-based Recommendation)是信息过滤技术的延续与发展，它是建立在项目的内容信息上做出推荐的，而不需要依据用户对项目的评价意见，更多地需要用机器学习的方法从关于内容的特征描述的事例中得到用户的兴趣资料。

在基于内容的推荐系统中，项目或对象是通过相关特征的属性来定义的，系统基于用户评价对象的特征、学习用户的兴趣，考察用户资料与待预测项目的匹配程度。用户的资料模型取决于所用的学习方法，常用的有决策树、神经网络和基于向量的表示方法等。基于内容的用户资料需要有用户的历史数据，用户资料模型可能随着用户的偏好改变而发生变化。

基于内容的推荐与基于人口统计学的推荐有类似的地方，只不过系统评估的中心转到了物品本身，使用物品本身的相似度而不是用户的相似度来进行推荐。

这一类一般依赖于自然语言处理NLP的一些知识，通过挖掘文本的TF-IDF特征向量，来得到用户的偏好，进而做推荐。这类推荐算法可以找到用户独特的小众喜好，而且还有较好的解释性。

### ● 算法优点：

用户兴趣可以很好地建模，并通过对物品属性维度的增加，获得更好的推荐精度。

### ● 算法缺点：

物品的属性有限，很难有效得到更多数据；

物品相似度的衡量标准只考虑到了物品本身，有一定的片面性；

需要用户的物品的历史数据，有冷启动的问题。

### ● 算法举例

CB算法的原理是将一个item的基本属性，内容等信息提取出来，抽成一个taglist，为每个tag赋一个权重。

剩下的事情就跟一个搜索引擎非常类似了，将所有item对应的taglist做一下倒排转换，放到倒排索引服务器中存储起来。

当要对某一个item做相关推荐的时候，将这个item对应的taglist拿出来拼成一个类似搜索系统中的query表达式，再将召回的结果做一下排序作为推荐结果输出。

当要对某个用户做个性化推荐的时候，将这个用户最近喜欢/操作过的item列表拿出来，将这些item的taglist拿出来并merge一下作为用户模型，并将这个模型的taglist请求倒排索引服务，将召回的结果作为候选推荐给该用

户。

#### ■ 优点：

不依赖于用户行为,即不需要冷启动的过程,随时到随时都能推荐

可以给出看起来比较合理的推荐解释

item被推荐的时效性可以做得很高,比如新闻类产品就需要用到该算法

#### ■ 缺点：

需要理解item的内容,对音频/视频等不好解析内容的就不好处理了

对于一次多义以及一义多词等情况处理起来比较复杂

容易出现同质化严重的问题,缺乏惊喜

#### ○ 协同过滤推荐：

基于协同过滤的推荐算法(Collaborative Filtering Recommendation)技术是推荐系统中应用最早和最为成功的技术之一。它一般采用最近邻技术,利用用户的历史喜好信息计算用户之间的距离,然后利用目标用户的最近邻居用户对商品评价的加权评价来预测目标用户对特定商品的喜好程度,从而根据这一喜好程度来对目标用户进行推荐。

基于协同过滤的推荐算法最大优点是对推荐对象没有特殊的要求,能处理非结构化的复杂对象,如音乐、电影。

基于协同过滤的推荐算法是基于这样的假设:为一用户找到他真正感兴趣的内容的好方法是首先找到与此用户有相似兴趣的其他用户,然后将他们感兴趣的内容推荐给此用户。其基本思想非常易于理解,在日常生活中,人们往往会利用好朋友的推荐来进行一些选择。基于协同过滤的推荐算法正是把这一思想运用到电子商务推荐系统中来,基于其他用户对某一内容的评价来向目标用户进行推荐。

基于协同过滤的推荐系统可以说是从用户的角度来进行相应推荐的,而且是自动的,即用户获得的推荐是系统从购买模式或浏览行为等隐式获得的,不需要用户努力地找到适合自己兴趣的推荐信息,如填写一些调查表格等。

#### ● 算法优点

- 1)能够过滤难以进行机器自动内容分析的信息,如艺术品、音乐等。
- 2)共享其他人的经验,避免了内容分析的不完全和不精确,并且能够基于一些复杂的,难以表述的概念(如信息质量、个人品位)进行过滤。
- 3)有推荐新信息的能力。可以发现内容上完全不相似的信息,用户对推荐信息的内容事先是预料不到的。这也是基于协同过滤的推荐算法和基于内容的推荐一个较大的差别,基于内容的推荐很多都是用户本来就熟悉的内容,而基于协同过滤的推荐可以发现用户潜在的但自己尚未发现的兴趣偏好。
- 4)能够有效地使用其他相似用户的反馈信息,减少用户的反馈量,加快个性化学习的速度。

#### ● 算法举例

CF算法的原理是汇总所有<user, item>的行为对,利用集体智慧做推荐。其原理很像朋友推荐,比如通过对用户喜欢的item进行分析,发现用户A和用户B很像(他们都喜欢差不多的东西),用户B喜欢了某个item,而用户A没有喜欢,那么就把这个item推荐给用户A。(User-Based CF)

当然,还有另外一个维度的协同推荐。即对比所有数据,发现itemA和itemB很像(他们被差不多的人喜欢),那么就把用户A喜欢的所有item,将这些item类似的item列表拉出来,作为被推荐候选推荐给用户A。(Item-Based

CF)

### ■ 优点:

能起到意想不到的推荐效果, 经常能推荐出来一些惊喜结果

进行有效的长尾item

只依赖用户行为, 不需要对内容进行深入了解, 使用范围广

### ■ 缺点

一开始需要大量的<user,item>行为数据, 即需要大量冷启动数据

很难给出合理的推荐解释

### ● 算法类型:

协同过滤算法具体实现的时候, 又分为典型的两类:

#### ■ 基于领域的协同过滤算法

这类算法的主要思想是利用<user,item>的打分矩阵, 利用统计信息计算用户和用户, item和item之间的相似度。然后再利用相似度排序, 最终得出推荐结果。

#### ● User-Based CF

先看公式:

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

该公式要计算用户i和用户j之间的相似度,  $I_{ij}$ 是代表用户i和用户j共同评价过的物品,  $R(i,x)$ 代表用户i对物品x的评分,  $\bar{R}(i)$ 头上有一杠的代表用户i所有评分的平均分, 之所以要减去平均分是因为有的用户打分严有的松, 归一化用户打分避免相互影响。

该公式没有考虑到热门商品可能会被很多用户所喜欢, 所以还可以优化加一下权重, 这儿就不演示公式了。在实际生产环境中, 经常用到另外一个类似的算法Slope One, 该公式是计算评分偏差, 即将共同评价过的物品, 将各自的打分相减再求平均。

#### ● Item-Based CF

先看公式:

$$sim(i, j) = \frac{\sum_{x \in U_{ij}} (r_{i,x} - \bar{r}_i)(r_{j,x} - \bar{r}_j)}{\sqrt{\sum_{x \in U_{ij}} (r_{i,x} - \bar{r}_i)^2} \sqrt{\sum_{x \in U_{ij}} (r_{j,x} - \bar{r}_j)^2}}$$

该公式跟User-Based CF是类似的, 就不再重复解释了。

### ● 算法问题

这类算法会面临两个典型的问题:

矩阵稀疏问题

计算资源有限导致的扩展性问题

### ● 算法选择:

Item-CF和User-CF选择

- user和item数量分布以及变化频率

如果user数量远远大于item数量,采用Item-CF效果会更好,因为同一个item对应的打分会比较多,而且计算量会相对较少

如果item数量远远大于user数量,则采用User-CF效果会更好,原因同上

在实际生产环境中,有可能因为用户无登陆,而cookie信息又极不稳定,导致只能使用item-cf

如果用户行为变化频率很慢(比如小说),用User-CF结果会比较稳定

如果用户行为变化频率很快(比如新闻,音乐,电影等),用Item-CF结果会比较稳定

- 相关和惊喜的权衡

item-based出的更偏相关结果,出的可能都是看起来比较类似的结果

user-based出的更有可能有惊喜,因为看的是人与人的相似性,推出来的结果可能更有惊喜

- 数据更新频率和时效性要求

对于item更新时效性较高的产品,比如新闻,就无法直接采用item-based的CF,因为CF是需要批量计算的,在计算结果出来之前新的item是无法被推荐出来的,导致数据时效性偏低;但是可以采用user-cf,再记录一个在线的用户item行为对,就可以根据用户最近类似的用户的行为进行时效性item推荐;对于像影视,音乐之类的还是可以采用item-cf的;

- 基于模型的协同过滤算法

- 用聚类算法做协同过滤

用聚类算法做协同过滤就和前面的基于用户或者项目的协同过滤有些类似了。我们可以按照用户或者按照物品基于一定的距离度量来进行聚类。如果基于用户聚类,则可以将用户按照一定距离度量方式分成不同的目标人群,将同样目标人群评分高的物品推荐给目标用户。基于物品聚类的话,则是将用户评分高物品的相似同类物品推荐给用户。常用的聚类推荐算法有K-Means, BIRCH, DBSCAN和谱聚类。

K-Means聚类算法原理

<https://www.cnblogs.com/pinard/p/6164214.html>

BIRCH聚类算法原理

<https://www.cnblogs.com/pinard/p/6179132.html>

DBSCAN密度聚类算法

<https://www.cnblogs.com/pinard/p/6208966.html>

谱聚类(spectralclustering)原理总结

<https://www.cnblogs.com/pinard/p/6221564.html>

- 用分类算法做协同过滤

如果我们根据用户评分的高低,将分数分成几段的话,则这个问题变成分类问题。比如最直接的,设置一份评分阈值,评分高于阈值的就是推荐,评分低于阈值就是不推荐,我们将问题变成了一个二分类问题。虽然分类问题的算法多如牛毛,但是目前使用最广泛的是逻辑回归。为啥是逻辑回归而不是看起来更加高大上的比如支持向量机呢?因为逻辑回归的解释性比较强,每个物品是否推荐我们都有一个明确的概率放在这,同时可以对数据的特征做工程化,得到调优的目的。目前逻辑回归做协同过滤在BAT等大厂已经非常成熟了。常见的分类推荐算法有逻辑回归和朴素贝叶斯,两者的特点是解释性很强。

用回归算法做协同过滤比分类算法看起来更加的自然。我们的评分可以是一个连续的值而不是离散的值,通过回归模型我们可以得到目标用户对某商品的预测打分。

常用的回归推荐算法有Ridge回归，回归树和支持向量回归。

### 线性回归原理小结

<https://www.cnblogs.com/pinard/p/6004041.html>

### 决策树算法原理(下)

<https://www.cnblogs.com/pinard/p/6053344.html>

### 支持向量机原理(五)线性支持回归

<https://www.cnblogs.com/pinard/p/6113120.html>

### 逻辑回归原理小结

<https://www.cnblogs.com/pinard/p/6029432.html>

### 朴素贝叶斯算法原理小结

<https://www.cnblogs.com/pinard/p/6069267.html>

#### ● 用矩阵分解做协同过滤

用矩阵分解做协同过滤是目前使用也很广泛的一种方法。由于传统的奇异值分解SVD要求矩阵不能有缺失数据，必须是稠密的，而我们的用户物品评分矩阵是一个很典型的稀疏矩阵，直接使用传统的SVD到协同过滤是比较复杂的。

目前主流的矩阵分解推荐算法主要是SVD的一些变种，比如FunkSVD，BiasSVD和SVD++。这些算法和传统SVD的最大区别是不再要求将矩阵分解为 $U\Sigma V^T$ 的形式，而变是两个低秩矩阵 $PTQ^T$ 的乘积形式。

### SVD算法概述

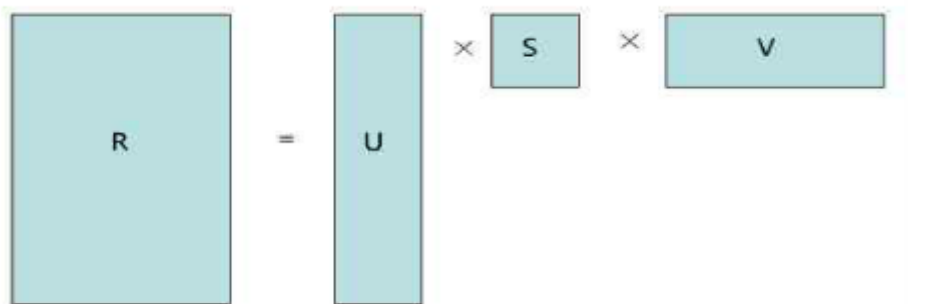
SVD(Singular Value Decomposition)的想法是根据已有的评分情况，分析出评分者对各个物品因子的喜好程度，以及各个物品对于这些因子的包含程度，最后再反过来根据分析结果预测评分。通过SVD的方式可以找出影响评分的显示因子和隐藏因子，这样更多有意义的关联关系就会被发现出来。

### SVD算法计算

SVD的数学定义：将给定评分矩阵 $R$ 分解成为三个矩阵的乘积，其中 $U$ 、 $V$ 称为左、右奇异向量， $\Sigma$ 对角线上的值称为奇异值；其中 $R$ 为 $n*m$ 的矩阵， $U$ 为 $n*n$ 的矩阵， $\Sigma$ 为 $n*m$ 的矩阵， $V$ 为 $m*m$ 的矩阵；可以使用前 $k$ 个奇异值来近似的替代 $R$ 矩阵，因为前1%的奇异值的和就占了全部奇异值和的99%以上。

### 基于矩阵分解的潜在语义模型的推荐算法

这儿只简单介绍一下基于矩阵分解的潜在语义模型的推荐算法。该算法首先将稀疏矩阵用均值填满，然后利用矩阵分解将其分解为两个矩阵相乘，如下图：


$$R_{m \times n} = U_{m \times r} * S_{r \times r} * V_{r \times n}$$

看一个实际的例子：



Index Words	Titles								
	T1	T2	T3	T4	T5	T6	T7	T8	T9
book			1	1					
dads						1			1
dummies		1						1	
estate							1		1
guide	1					1			
investing	1	1	1	1	1	1	1	1	1
market	1		1						
real							1		1
rich						2			1
stock	1		1					1	
value				1	1				

book	0.15	-0.27	0.04						
dads	0.24	0.38	-0.09						
dummies	0.13	-0.17	0.07						
estate	0.18	0.19	0.45						
guide	0.22	0.09	-0.46						
investing	0.74	-0.21	0.21						
market	0.18	-0.30	-0.28						
real	0.18	0.19	0.45						
rich	0.36	0.59	-0.34						
stock	0.25	-0.42	-0.28						
value	0.12	-0.14	0.23						

这个例子中, 原始矩阵中包含了网页的Title和切词后的term之间关系, 可以类比为推荐系统中的评分。然后用SVD做矩阵分解之后, 针对每个term会对应一个3维向量, 针对每个Title也会对应一个3维向量。

那么接下来可以做的事情就很多了, 如果要计算term和title的相似度, 只需要将这两个3维向量做内积得到的分值即可; 还可以将term和title都投影到这3维空间中, 然后利用各种聚类算法, 将用户和item, item和item, 用户和用户的分类都给求出来。该算法的核心是在于做矩阵分解, 在矩阵大了的情况下计算量是非常夸张的, 在实际生产环境中会常用梯度递归下降方法来求得一个近似解。

#### ● 用神经网络做协同过滤

用神经网络乃至深度学习做协同过滤应该是以后的一个趋势。目前比较主流的用两层神经网络来做推荐算法的是限制玻尔兹曼机(RBM)。在目前的Netflix算法比赛中, RBM算法的表现很牛。当然如果用深层的神经网络来做协同过滤应该会更好, 大厂商用深度学习的方法来做协同过滤应该是将来的一个趋势。后续我会专门开篇来讲讲RBM。

#### ● 用图模型做协同过滤

用图模型做协同过滤, 则将用户之间的相似度放到了一个图模型里面去考虑, 常用的算法是SimRank系列算法和马尔科夫模型算法。对于SimRank系列算法, 它的基本思想是被相似对象引用的两个对象也具有相似性。算法思想有点类似于大名鼎鼎的PageRank。而马尔科夫模型算法当然是基于马尔科夫链了, 它的基本思想是基于传导性来找出普通距离度量算法难以找出的相似性。后续我会专门开篇来讲讲SimRank系列算法。

#### ● 用隐语义模型做协同过滤

隐语义模型主要是基于NLP的, 涉及到对用户行为的语义分析来做评分推荐, 主要方法有隐性语义分析LSA和隐含狄利克雷分布LDA。

#### ● 协同过滤的新方向

当然推荐算法的变革也在进行中, 就算是最火爆的基于逻辑回归推荐算法也在面临被取代。哪些算法可能取代逻辑回归之类的传统协同过滤呢?

a) 基于集成学习的方法和混合推荐: 这个和混合推荐也靠在一起了。由于集成学习的成熟, 在推荐算法上也有较好的表现。一个可能取代逻辑回归的算法是GBDT。目前GBDT在很多算法比赛都有好的表现, 而有工业级的并行化实现类库。

b) 基于矩阵分解的方法: 矩阵分解, 由于方法简单, 一直受到青睐。目前开始渐渐流行的矩阵分解方法有分解机(Factorization Machine)和张量分解(Tensor Factorization)。

c) 基于深度学习的方法：目前两层的神经网络RBM都已经有非常好的推荐算法效果，而随着深度学习和多层神经网络的兴起，以后可能推荐算法就是深度学习的天下了？目前看最火爆的是基于CNN和RNN的推荐算法。

### ○ 基于规则的推荐：

基于关联规则的推荐(Association Rule-based Recommendation)是以关联规则为基础，把已购商品作为规则头，规则体为推荐对象。关联规则挖掘可以发现不同商品在销售过程中的相关性，在零售业中已经得到了成功的应用。 [2]

关联规则就是在一个交易数据库中统计购买了商品集X的交易中有多大比例的交易同时购买了商品集y，其直观的意义就是用户在购买某些商品的时候有多大倾向去购买另外一些商品。比如购买牛奶的同时很多人会购买面包。

算法的第一步关联规则的发现最为关键且最耗时，是算法的瓶颈，但可以离线进行。其次，商品名称的同义性问题也是关联规则的一个难点。

这类算法常见的比如基于最多用户点击，最多用户浏览等，属于大众型的推荐方法，在目前的大数据时代并不主流。

一般我们可以找出用户购买的所有物品数据里频繁出现的项集活序列，来做频繁集挖掘，找到满足支持度阈值的关联物品的频繁N项集或者序列。如果用户购买了频繁N项集或者序列里的部分物品，那么我们可以将频繁项集或序列里的其他物品按一定的评分准则推荐给用户，这个评分准则可以包括支持度，置信度和提升度等。

常用的关联推荐算法有Apriori，FP Tree和PrefixSpan。

#### Apriori算法原理总结

<https://www.cnblogs.com/pinard/p/6293298.html>

#### FP Tree算法原理总结

<https://www.cnblogs.com/pinard/p/6307064.html>

#### PrefixSpan算法原理总结

<https://www.cnblogs.com/pinard/p/6323182.html>

### ○ 基于效用的推荐

基于效用的推荐 (Utility-based Recommendation) 是建立在对用户使用项目的效用情况上计算的，其核心问题是怎样为每一个用户去创建一个效用函数，因此，用户资料模型很大程度上是由系统所采用的效用函数决定的。

基于效用推荐的好处是它能把非产品的属性，如提供商的可靠性( Vendor Reliability)和产品的可得性(Product Availability)等考虑到效用计算中。

### ○ 基于知识的推荐

基于知识的推荐( Knowledge-based Recommendation)在某种程度是可以看成是一种推理(Inference)技术，它不是建立在用户需要和偏好基础上推荐的。

基于知识的方法因它们所用的功能知识不同而有明显区别。效用知识( FunctionalKnowledge)是一种关于一个项目如何满足某一特定用户的知识, 因此能解释需要和推荐的关系, 所以用户资料可以是任何能支持推理的知识结构, 它可以是用户已经规范化的查询, 也可以是一个更详细的用户需要的表示。

### ○ 基于流行度的推荐

基于流行度的算法非常简单粗暴, 类似于各大新闻、微博热榜等, 根据PV、UV、日均PV或分享率等数据来按某种热度排序来推荐给用户。

### ○ 基于人口统计信息的推荐:

这一类是最简单的推荐算法了, 它只是简单的根据系统用户的基本信息发现用户的相关程度, 然后进行推荐, 目前在大型系统中已经较少使用。

### ○ 组合推荐

由于各种推荐方法都有优缺点, 所以在实际中, 组合推荐( Hybrid Recommendation)经常被采用。研究和应用最多的是内容推荐和协同过滤推荐的组合。

最简单的做法就是分别用基于内容的方法和协同过滤推荐方法去产生一个推荐预测结果, 然后用某方法组合其结果。尽管从理论上有很多种推荐组合方法, 但在某一具体问题中并不见得都有效, 组合推荐的一个最重要原则就是通过组合来避免或弥补各自推荐技术的弱点。

这个类似我们机器学习中的集成学习, 博才众长, 通过多个推荐算法的结合, 得到一个更好的推荐算法, 起到三个臭皮匠顶一个诸葛亮的作用。比如通过建立多个推荐算法的模型, 最后用投票法决定最终的推荐结果。混合推荐理论上不会比单一任何一种推荐算法差, 但是使用混合推荐, 算法复杂度就提高了, 在实际应用中有使用, 但是并没有单一的协调过滤推荐算法, 比如逻辑回归之类的二分类推荐算法广泛。

其实从实践中来看, 没有哪种推荐技术敢说自己没有弊端, 往往一个好的推荐系统也不是只用一种推荐技术就解决

#### ● 常见的算法组合方式:

混合推荐技术: 同时使用多种推荐技术再加权取最优;

切换推荐技术: 根据用户场景使用不同的推荐技术;

特征组合推荐技术: 将一种推荐技术的输出作为特征放到另一个推荐技术当中;

层叠推荐技术: 一个推荐模块过程中从另一个推荐模块中获取结果用于自己产出结果;

#### ● 常见的特征组合方式

1)加权 (Weight): 加权多种推荐技术结果。

2)变换( Switch);根据问题背景和实际情况或要求决定变换采用不同的推荐技术。

3)混合( Mixed):同时采用多种推荐技术给出多种推荐结果, 为用户提供参考。

4)特征组合(Feature Combination):组合来自不同推荐数据源的特征被另一种推荐算法所采用。

5)层叠( Cascade): 先用一种推荐技术产生一种粗糙的推荐结果, 第二种推荐技术在此推荐结果的基础上进一步做出更精确的推荐。

6)特征扩充( Feature Augmentation): 将一种技术产生附加的特征信息嵌入另一种推荐技术的特征输入中。

7)元级别( Meta-Ievel):用一种推荐方法产生的模型作为另一种推荐方法的输入。



- 推荐算法总结和对比
- 推荐算法种类



- 推荐算法对比

表1 主要推荐方法对比		
推荐方法	优点	缺点
基于内容推荐	推荐结果直观，容易解释； 不需要领域知识	新用户问题； 复杂属性不好处理； 要有足够数据构造分类器
协同过滤推荐	新异兴趣发现、不需要领域知识； 随着时间推移性能提高； 推荐个性化、自动化程度高； 能处理复杂的非结构化对象	稀疏问题； 可扩展性问题； 新用户问题； 质量取决于历史数据集； 系统开始时推荐质量差；
基于规则推荐	能发现新兴趣点； 不要领域知识	规则抽取难、耗时； 产品名同义性问题； 个性化程度低；
基于效用推荐	无冷开始和稀疏问题； 对用户偏好变化敏感； 能考虑非产品特性	用户必须输入效用函数； 推荐是静态的，灵活性差； 属性重叠问题；
基于知识推荐	能把用户需求映射到产品上； 能考虑非产品属性	知识难获得； 推荐是静态的

- 相似度计算
- 欧几里德距离评价

欧几里德距离评价是一个较为简单的用户关系评价方法。原理是通过计算两个用户在散点图中的距离来判断不同的用户是否有相同的偏好。

- 皮尔逊相关系数

皮尔逊相关系数(Pearson Correlation, PC)：在基于用户近邻推荐算法中，常用的一种度量公式，效果最佳。皮尔逊相关系数用来说明两个用户间联系的强弱程度。

相关系数的分类

0.8-1.0 极强相关

0.6-0.8 强相关

0.4-0.6 中等程度相关

0.2-0.4 弱相关

0.0-0.2 极弱相关或无相关

使用皮尔森/皮尔逊相关系数(Pearson Correlation Coefficient)来表示两个用户之间的相关性,取值范围为[-1,+1], -1表示强负相关, +1表示强正相关, 0表示不相关。

$$\ell_{u,v} = \frac{\sum_{p \in P_{u,v}} (r_{u,p} - \bar{r}_u)(r_{v,p} - \bar{r}_v)}{\sqrt{\sum_{p \in P_{u,v}} (r_{u,p} - \bar{r}_u)^2} \sqrt{\sum_{p \in P_{u,v}} (r_{v,p} - \bar{r}_v)^2}}$$

Pearson相关系数的应用要求两个变量的标准差都不为零的时候,该相关系数才具有定义,使用场景如下:

- (1) 两个变量之间是线性关系,都是连续数据;
- (2) 两个变量的总体是正态分布或者解决正态的单峰分布;
- (3) 两个变量的观察值是成对的,每对观测值之间是相互对立的;
- (4) 当计算出用户a和其它用户的相关性的时候,我们可以选择出最相似的N个近邻用户计算对物品p的评分预测值(N个近邻用户对物品p都有评分值)。

$$pred(u, p) = \bar{r}_u + \frac{\sum_{v \in N} \ell_{u,v} * (r_{v,p} - \bar{r}_v)}{\sum_{v \in N} \ell_{u,v}}$$

#### • 余弦向量相似度(AdjustedCosine, AC):

在基于物品近邻推荐算法中,最常用的一个相似度度量计算公式,效果最佳。

通过测量两个向量内积空间的夹角的余弦值来度量它们之间的相似性。0度角的余弦值是1,而其他任何角度的余弦值都不大于1;并且其最小值是-1。从而两个向量之间的角度的余弦值确定两个向量是否大致指向相同的方向。两个向量有相同的指向时,余弦相似度的值为1;两个向量夹角为90°时,余弦相似度的值为0;两个向量指向完全相

反的方向时,余弦相似度的值为-1。在比较过程中,向量的规模大小不予考虑,仅仅考虑到向量的指向方向。余弦相似度通常用于两个向量的夹角小于90°之内,因此余弦相似度的值为0到1之间。

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta$$

#### • 均方差(Mean SquaredDifference, MSD):

使用用户u和用户v对相同物品评分差的平方和均值的倒数来表示两个人的相似度。缺点是:无法表示负关联。

#### • 斯皮尔曼等级关联(Spearman RankCorrelation, SRC):

利用用户对物品评分的排名来计算两个用户之间的相似度。缺点是:计算排名,消耗比较大,对于用户评分只有少量可选值的情况下,会产生大量并列的排名,效果不佳。

当两个用户对物品进行评分的时候，如果评分商品都比较少，而且分数比较一致的情况下，我们会认为，这两个用户比较相似，基于物品近邻算法也是一样的。但是这个时候实际上这两个用户的爱好事实上可能是完全不同的，解决这个问题，我们可以通过在相似度上添加一个重要性权重或者进行相似度缩放的方式对相似度进行转换。

- **重要性权重:**

当两个用户/物品之间的共同评分的物品/用户数量小于给定阈值的时候，就降低相似度重要性的权重。阈值大于等于25的时候，有比较好的预测结果。

- **相似度缩放:**

给定一个收缩因子对相似度进行收缩转换，当共同评分数量远远大于收缩因子的时候，物品的相似度几乎没有变化；当收缩因子为100的时候，效果不错。

$$\ell'_{a,b} = \frac{\min\{|P_{a,b}|, \alpha\}}{\alpha} * \ell_{a,b} \quad \ell'_{a,b} = \frac{|P_{a,b}|}{|P_{a,b}| + \alpha} * \ell_{a,b}$$

- **反用户频率(Inverse UserFrequency, IUF)**

两个用户对物品给出一致的喜欢和不喜欢的评分，可能会不如他们给出差异更大的评分时提供的信息量多。可以通过使用反用户频率(Inverse User Frequency, IUF)来对相似度计算公式进行转换，每个物品*i*都会赋以权重*c<sub>i</sub>*，对应评论了物品*i*的用户比例的log值：

$$c_i = \log \frac{|U|}{|U_i|}$$

- **频率加权皮尔逊相关系数(Frequency-Weighted Pearson Correlation, FWPC)**

公式为：

$$FWPC(u, v) = \ell_{u,v} = \frac{\sum_{p \in P_{u,v}} c_p (r_{u,p} - \bar{r}_u) (r_{v,p} - \bar{r}_v)}{\sqrt{\sum_{p \in P_{u,v}} c_p (r_{u,p} - \bar{r}_u)^2} \sqrt{\sum_{p \in P_{u,v}} c_p (r_{v,p} - \bar{r}_v)^2}}$$

- **反物品频率(Inverse ItemFrequency, IIF)**

类似于反用户频率，在基于物品近邻推荐中，可以使反物品频率(Inverse ItemFrequency, IIF)来对相似度计算公式进行转换，每个用户*u*都会赋予一个权重*c<sub>u</sub>*，对应用户*u*进行评论的物品数量比例的log值：

$$c_u = \log \frac{|P|}{|P_u|}$$

- **频率加权改进余弦相似度(Frequency-Weighted Adjusted Cosine, FWAC)**

公式为：

$$FWAC(a,b) = \ell_{a,b} = \frac{\sum_{u \in U} c_u (r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U} c_u (r_{u,a} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} c_u (r_{u,b} - \bar{r}_u)^2}}$$

- Tanimoto 系数 (TanimotoCoefficient)

Tanimoto 系数也称为Jaccard 系数，是 Cosine 相似度的扩展，也多用于计算文档数据的相似度：

$$T(x,y) = \frac{x \bullet y}{\|x\|^2 + \|y\|^2 - x \bullet y} = \frac{\sum x_i y_i}{\sqrt{\sum x_i^2} + \sqrt{\sum y_i^2} - \sum x_i y_i}$$

- 今日头条推荐系统

- 推荐系统的输入维度：

推荐系统，如果用形式化的方式去描述实际上是拟合一个用户对内容满意度的函数，这个函数需要输入三个维度的变量。

第一个维度是内容。头条现在已经是一个综合内容平台，图文、视频、UGC小视频、问答、微头条，每种内容有很多自己的特征，需要考虑怎样提取不同内容类型的特征做好推荐。

第二个维度是用户特征。包括各种兴趣标签，职业、年龄、性别等，还有很多模型刻画出的隐式用户兴趣等。

第三个维度是环境特征。这是移动互联网时代推荐的特点，用户随时随地移动，在工作场合、通勤、旅游等不同的场景，信息偏好有所偏移。

- 推荐系统的推荐特征

典型的推荐特征，主要有四类特征会对推荐起到比较重要的作用。

第一类是相关性特征，就是评估内容的属性和与用户是否匹配。显性的匹配包括关键词匹配、分类匹配、来源匹配、主题匹配等。像FM模型中也有一些隐性匹配，从用户向量与内容向量的距离可以得出。

第二类是环境特征，包括地理位置、时间。这些既是bias特征，也能以此构建一些匹配特征。

第三类是热度特征。包括全局热度、分类热度，主题热度，以及关键词热度等。内容热度信息在大的推荐系统特别在用户冷启动的时候非常有效。

第四类是协同特征，它可以在部分程度上帮助解决所谓算法越推越窄的问题。协同特征并非考虑用户已有历史。而是通过用户行为分析不同用户间相似性，比如点击相似、兴趣分类相似、主题相似、兴趣词相似，甚至向量相似，从而扩展模型的探索能力。

- 推荐系统的关键环节

- 内容分析

内容分析包括文本分析，图片分析和视频分析。头条一开始主要做资讯，今天我们主要讲一下文本分析。文本分析在推荐系统中一个很重要的作用是用户兴趣建模。没有内容及文本标签，无法得到用户兴趣标签。举个例子，只有知道文章标签是互联网，用户看了互联网标签的文章，才能知道用户有互联网标签，

文本内容的标签可以直接帮助推荐特征，比如魅族的内容可以推荐给关注魅族的用户，这是用户标签的匹配。如果某段时间推荐主频道效果不理想，出现推荐窄化，用户会发现到具体的频道推荐（如科技、体育、娱乐、军事等）中阅读后，再回主feed,推荐效果会更好。因为整个模型是打通的，子频道探索空间较小，更容易满足用户需求。只通过单一信道反馈提高推荐准确率难度会比较大，子频道做的好很重要。而这也需要好的内容分析。

结合三方面的维度，模型会给出一个预估，即推测推荐内容在这一场景下对这一用户是否合适。

推荐模型中，点击率、阅读时间、点赞、评论、转发包括点赞都是可以量化的目标，能够用模型直接拟合做预估，看线上提升情况可以知道做的好不好。但一个大体量的推荐系统，服务用户众多，不能完全由指标评估，引入数据指标以外的要素也很重要。

比如广告和特型内容频控。像问答卡片就是比较特殊的内容形式，其推荐的目标不完全是让用户浏览，还要考虑吸引用户回答为社区贡献内容。这些内容和普通内容如何混排，怎样控制频控都需要考虑。

此外，平台出于内容生态和社会责任的考量，像低俗内容的打压，标题党、低质内容的打压，重要新闻的置顶、加权、强插，低级别账号内容降权都是算法本身无法完成，需要进一步对内容进行干预。

### ■ 用户标签

今日头条常用的用户标签包括用户感兴趣的类别和主题、关键词、来源、基于兴趣的用户聚类以及各种垂直兴趣特征（车型，体育球队，股票等）。还有性别、年龄、地点等信息。性别信息通过用户第三方社交账号登录得到。年龄信息通常由模型预测，通过机型、阅读时间分布等预估。常驻地点来自用户授权访问位置信息，在位置信息的基础上通过传统聚类的方法拿到常驻点。常驻点结合其他信息，可以推测用户的工作地点、出差地点、旅游地点。这些用户标签非常有助于推荐。

#### ● 推荐系统的效果评估

全面的评估推荐系统，需要完备的评估体系、强大的实验平台以及易用的经验分析工具。所谓完备的体系就是并非单一指标衡量，不能只看点击率或者停留时长等，需要综合评估。

#### ○ 参考资料：

<https://blog.csdn.net/hlang8160/article/details/81433356>

<https://www.cnblogs.com/xuanku/p/recsys.html>

<https://www.cnblogs.com/chenliyang/p/6548306.html>

[https://blog.csdn.net/gongxifacai\\_believe/article/details/82144031](https://blog.csdn.net/gongxifacai_believe/article/details/82144031)

<http://www.woshipm.com/pd/934582.html>

<https://blog.csdn.net/y1535623813/article/details/88179486>

<https://blog.csdn.net/wdr2003/article/details/80248150>

<https://blog.csdn.net/u012879957/article/details/80939101>

<https://blog.csdn.net/sdksdk0/article/details/80248999>

[https://blog.csdn.net/slx\\_share/article/details/80241764](https://blog.csdn.net/slx_share/article/details/80241764)

<https://www.cnblogs.com/xiangpiaopiao2011/p/8655897.html>

<https://www.jianshu.com/p/b564c19567b7>

[http://www.sohu.com/a/216961197\\_114778](http://www.sohu.com/a/216961197_114778)

<https://www.jianshu.com/p/87896dbcbb92>