推荐算法

1. 推荐算法概念

通过数学算法对用户行为大数据的分析，来推导、预测用户的需求、兴趣。并根据计算结果实现个性化的推荐，达到产品推广、增加用户粘度的目的。

1. 推荐算法分类

目前推荐算法大致分为6种:

* **基于内容**

对物品进行解构、建模、挖掘并建立物品的特征。并基于用户以前的喜好推荐相同或相似特征的物品。

比如，对所有影片进行解构出不同类型的影片（战争片、爱情片等）。并结合用户历史喜好数据（喜欢战争片），来推荐相同类型的影片（战争片）

* **协同过滤**

根据所有用户对物品的偏好，发掘与当前用户口味和偏好相似的用户群。并根据用户群的喜好物品进行推荐。

比如，通过数据挖掘发现30岁左右男性对战争片评分很高。所以为30岁左右男性客户推荐战争片

* **关联规则**

关联规则计算的是物品之间的关联性为基础。通过把已购商品作为规则起始，挖掘其他商品在销售过程中的相关性（即购买其他商品的概率）。来为用户的推荐商品。

比如：通过计算，发现用户购买薯片时，其中50%的用户购买了可乐，20%购买了面包，其他商品约占1%以下。所以对购买薯片的用户推荐可乐和面包。

* **基于效用**

基于效用的推荐是建立在对用户使用项目的效用情况上计算的，其核心问题是怎么样为每个用户去创建一个效用函数。因此，用户资料模型非常大程度上是由系统所採用的效用函数决定的。基于效用推荐的优点是它能把非产品的属性，如提供商的可靠性和产品的可得性等考虑到效用计算中

由于需要为每个用户创建函数，对用及其不友好，暂时没有样例也不考虑使用。

* **基于知识**

知识往往以一种交互的形式提供服务。是一种引导式的方式挖掘用户的喜好。首先用户需要指定初始偏好，之后系统根据初始偏好提供内容，用户对提供内容评分，系统根据评分动态挖掘用户的喜好标签。

样例：今日头条、网易云音乐

1. 选型

根据以上的分类以及说明，总结其中的优缺点如下：

|  |  |  |
| --- | --- | --- |
| 推荐方法 | 长处 | 缺点 |
| 基于内容推荐 | 推荐结果直观，easy解释；  不须要领域知识 | 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、不须要领域知识；  随着时间推移性能提高；  推荐个性化、自己主动化程度高；  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。  系统開始时推荐质量差； |
| 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 | 无冷開始和稀疏问题；  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 |

根据当前需求以及资料反馈、算法完备性、多样性上考虑，选择“协同过滤推荐算法”。

1. 协同过滤推荐算法

* 概念

协同过滤 (Collaborative Filtering, 简称 CF)简单的说就是推荐与你相似的人喜欢的物品或者推荐你喜欢物品相似的物品。其算法核心一个是基于用户(user-based)还是基于物品(item-based)，另一个就是相似算法。目前主流的相似算法有欧几里德相似度、皮尔逊相似度、余弦相似度。

* 协同过滤算法步骤：

1. 收集数据

这里的数据指的是用户的历史消费数据，往往包括用户、购买物品、评分、日期等。由于历史数据往往数据量较大，往往使用dat或者csv文件。样例如下：

userId,movieId,rating,timestamp

1,11,3.5,1112486027

1,12,3.5,1112484676

1,13,3.5,1112484819

2,12,3.5,1112484727

2,13,3.5,1112484580

2,14,3.5,1094785740

3,15,4.0,1094785734

3,11,4.0,1112485573

3,13,4.0,1112484940

1. 降噪

采集的数据需要进行筛选与过滤，比如去除非法数据等。

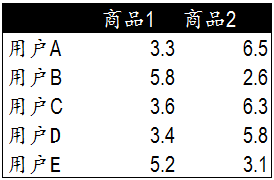
1. 归一化

物品的各个评分策略可能不同，需要统一度量

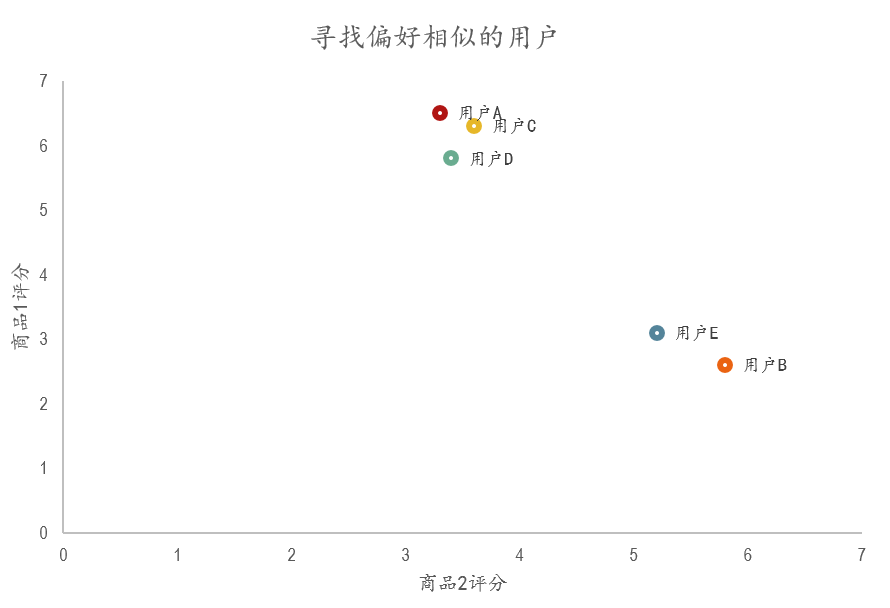
1. 建立散列图进行推演（逻辑推演步骤，正常流程没有）

该步骤仅为了好推演。其中二维图像最为直观，

比如基于用户存在如下数据：



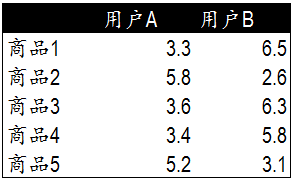
那么基于用户的散列图如下：

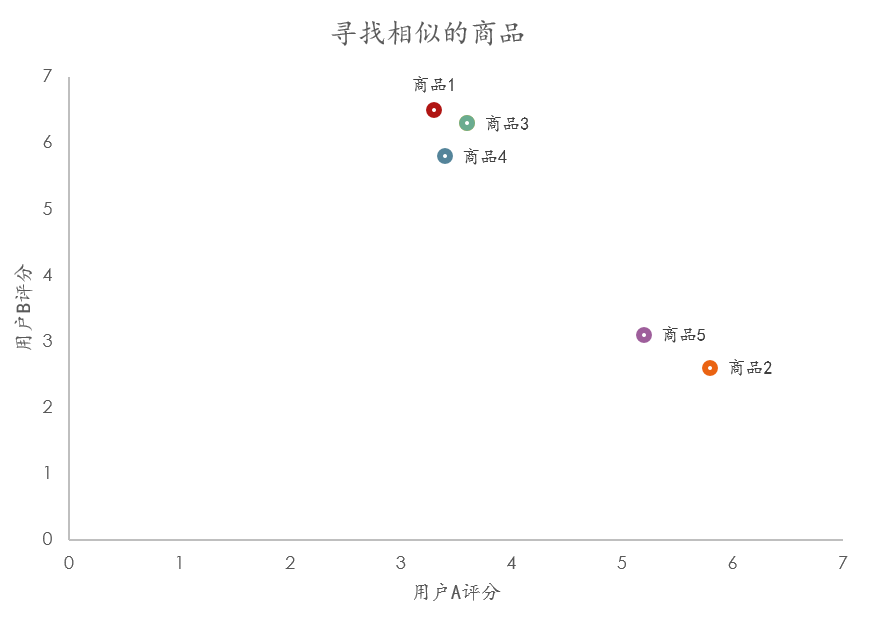


通过上面图片可以直观的发现用户A C D距离相近、兴趣评分相对一致的；而用户E和B相似。以上虽然直观但是实际场景往往不是二维图像（两种物品）。

那么基于物品的散列图如下：

数据如下：





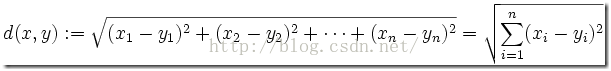
通过上面图片可以直观的发现商品1 3 4距离相近、兴趣评分相对一致的；而用户E和B相似。以上虽然直观但是实际场景往往不是二维图像（两种物品）。

1. 计算相似性

* 欧几里得度量（euclidean metric）

是一个通常采用的距离定义，指在m维空间中两个点之间的真实距离，或者向量的自然长度（即该点到原点的距离）。在二维和三维空间中的欧氏距离就是两点之间的实际距离。

公式如下：

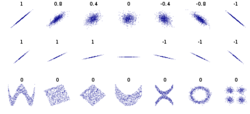


* 皮尔逊相关系数（ Pearson correlation coefficient）

参考：https://baike.baidu.com/item/皮尔逊相关系数

公式：

样图：



简单的说皮尔逊系数反应的是向量之间的离散程度。当系数偏1时表名向量方向相近；当系数偏0时，向量方向相离；当系数偏-1是向量方向相悖。

* 余弦相似度（ Pearson correlation coefficient）

参考：https://baike.baidu.com/item/余弦相似度

公式：

https://gss3.bdstatic.com/-Po3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D187/sign=50b538a78244ebf869716037eef8d736/0df431adcbef760970e3983b2bdda3cc7dd99ead.jpg

余弦相似度与皮尔逊系统相似，描述的是向量之间的偏移程度。

1. 筛选邻居

根据上面算法进行排序，并挑选相似的用户数量。

1. 实行推荐

挑选该用户组喜欢的产品，并实行对你的推荐

* 协同过滤的优劣

优点：

1.能够过滤机器难以自动内容分析的信息，如艺术品，音乐等。

2.共用其他人的经验，避免了内容分析的不完全或不精确，并且能够基于一些复杂的，难以表述的概念（如信息质量、个人品味）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。可以发现用户潜在的但自己尚未发现的兴趣偏好。

4.推荐个性化、自动化程度高、能够有效的利用其他相似用户的回馈信息、加快个性化学习的速度。

缺点：

1.新用户问题(New User Problem) 系统开始时推荐质量较差；

2.新项目问题(New Item Problem) 质量取决于历史数据集；

|  |  |  |
| --- | --- | --- |
|  | 基于用户 | 基于物品 |
| 性能 | 适用于用户较少的场合，如果用户过多，计算用户相似度矩阵的代价交大 | 适用于物品数明显小于用户数的场合，如果物品很多，计算物品相似度矩阵的代价交大 |
| 领域 | 实效性要求高，用户个性化兴趣要求不高 | 长尾物品丰富，用户个性化需求强烈 |
| 实时性 | 用户有新行为，不一定需要推荐结果立即变化 | 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，因为用户相似度是离线计算的； 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其他用户 | 新用户只要对一个物品产生行为，就能推荐相关物品给他，但无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 （但是新的item到来也同样是冷启动问题） |
| 推荐理由 | 很难提供令用户信服的推荐解释 | 可以根据用户历史行为归纳推荐理由 |

1. Java支持

目前算法在世界上如火如荼，所以java存在很多开源软件支持算法。其中对推荐算法支持比较全面的软件有mahout。

<http://mahout.apache.org>

样例演示如下：

* 收集数据源：

下载电影评价：<http://grouplens.org/datasets/movielens/的ml-10m.zip>或ml-20m.zip

* 编码如下:



* 该软件优势:

1. 对算法的支持比较完备，对推荐算法支持完备
2. 支持机器学习，如有该方面需求可以直接使用
3. 支持单机运行
4. 支持Hadoop、Spark，对大数据的支持良好
5. 支持集群部署，易于扩展计算能力

* 该软件缺点:

1. 第三方软件，算法已经固定，计算的灵活性稍差。
2. 稳定性需要进行测试。
3. 库设计

|  |  |  |
| --- | --- | --- |
| SQY\_RZDK\_SPTJ 商品推荐(序列:YH\_ID,SP\_ID) | | |
| 字段 | 字段说明 | 备注 |
| YH\_ID | 用户ID |  |
| SP\_ID | 商品ID |  |
| PX | 排序的号 |  |
| SCORE | 评分 |  |
| UPDATE\_TIME | 更新时间 |  |

|  |  |  |
| --- | --- | --- |
| SQY\_RZDK\_SPTJ\_CPJQ 推荐算法:产品加权 | | |
| 字段 | 字段说明 | 备注 |
| SP\_ID | 物品ID |  |
| SCORE | 加权评分 |  |
| UPDATE\_TIME | 更新时间 |  |

1. 配置参数

|  |  |  |
| --- | --- | --- |
| 编号 | 缩写 | 说明 |
| 01 | TJSF\_ REFRESH\_CRON | 推荐算法刷新策略CRON表达式 |
| 02 | TJSF\_ BASE\_TYPE | 推荐算法基于什么策略:  USER-用户(默认),ITEM-商品 |
| 03 | TJSF\_ NEIGHBOR\_SIZE | 推荐算法配置邻居数量(9<X<101),默认:30 |
| 04 | TJSF\_RECOMMEND \_RETURN\_SIZE | 推荐算法推荐个数（9<X<51）,默认:20 |
| 05 | TJSF\_RECOMMEND \_HANDLE\_SIZE | 推荐算法处理推荐个数（保存到内存个数，等待加权等后续处理，如果不配置为推荐个数\*2，不超过100）默认:40 |
| 06 | TJSF\_RECOMMEND\_WEIGHT \_SIZE | 推荐算法加权推荐个数,默认:5个 |
| 07 | TJSF\_NEW\_ITEM\_WEIGHT\_OPEN | 是否为新物品加权,默认:是 |
| 08 | TJSF\_NEW\_ITEM\_WEIGHT\_SCORE | 新物品加权值,默认:0 |
| 09 | TJSF\_NEW\_ITEM\_WEIGHT\_SIZE | 新物品推荐个数,默认:1 |
| 10 | TJSF\_NEW\_ITEM\_WEIGHT\_TIMEOUT | 新物品加权时长(单位:小时):72(3天) |

1. 流程图

* 推荐算法数据制作完整流程图



流程说明:

1. 启动数据的流程为:启动Kettle任务,查询数据库并计算评分,导出csv文件。并刷新加权表
   * + CSV的数据结构(一行为一个消费记录)：

<用户ID>,<商品ID>,<评分>

* + - 评分构成

用户购买商品+2.5，用户评价商品每星+0.5

1. 推荐算法读取CSV文件并计算（此时比较消耗内存）
2. 根据推荐算法返回TJSF\_RECOMMEND \_HANDLE\_SIZE个数商品
3. 查询数据库SQY\_RZDK\_SPTJ\_CPJQ获取物品加权数据
4. 为推荐算法返回产品集合加权（分数相加）
5. 为推荐算法产品集排序:
   * + - 在加权表选择TJSF\_RECOMMEND\_WEIGHT \_SIZE个，放在推荐集合头部
       - 为剩下的元素排序并补充到列表，直到列表数量达到TJSF\_RECOMMEND \_RETURN\_SIZE个
       - 如果查询数量不足，查询产品TOP热门产品进行补充
6. 刷新推荐表SQY\_RZDK\_SPTJ

* 获取产品推荐

1. 查询数据获取推荐产品
2. 如果查询不到，查询产品加权表进行替代
3. 如果加权数量不足，查询产品Top热门产品进行补充