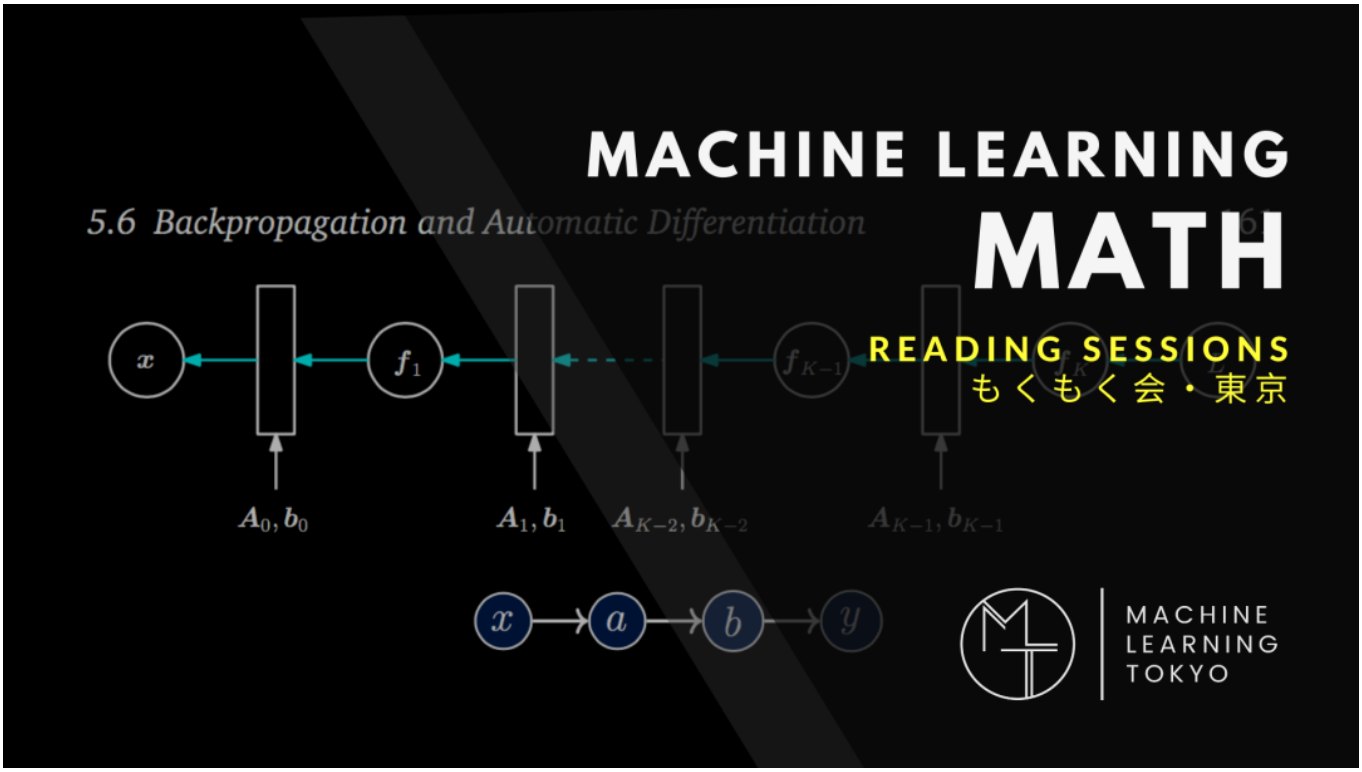
[技术]《白话大数据与机器学习》- 5 推荐算法协同过滤CF

原创 erixhao 未来阅读空间 2020-06-13

收录于话题
#技术博文

22个

> **“高屋建瓴地化繁为简，凡人饮水处，皆言机器学习”。**



未来阅读

读完需要
23
分钟
速读仅需 15 分钟

目录

1

简评

《白话大数据与机器学习》，通俗易懂, 但不失理论深度。对概念的入门级了解非常有用，提到了几乎所有关键点。

先从数据的概念以及基本的微积分、统计学、概率论知识点讲起，为后续理解算法的概念和公式打下基础；然后对监督学习和非监督学习的几类重点算法进行了详细的讲述，包括分类、聚类、回归、关联、推荐、神经网络等，最后还介绍了大数据的一些框架，及工业上的一些方法。

2

作者： 高扬 卫峥 尹会生等

3

机器学习算法有很多，有分类、回归、聚类、推荐、图像识别领域等等，具体算法比如**线性回归、逻辑回归、朴素贝叶斯、随机森林、支持向量机、神经网络**等等。在机器学习算法中，没有最好的算法，只有“更适合”解决当前任务的算法。

第五部分分享书中提到的关于**推荐算法 - 协同过滤**。

1)

推荐算法

推荐算法在现在很多电子商务网站中普遍应用。推荐系统作为现在众多电商系统、内容分发系统等网站的必要子系统，越来越受到运营者的重视。推荐系统核心要解决的问题是提高转化率，也就是经过分析，要猜测某一个用户更喜欢什么商品，更可能购买什么商品，或者更喜欢哪些歌曲、文章，在系统中要进行适当形式的推荐，如页面飘窗、营销邮件、短信息等。

总体来说，推荐系统作为一个确定目的的系统——提高转化率，有很多的线索可以去做。因为只要有片段性的信息，就有提高转化率的可能性，只要转化率比推荐系统不参与的时候高，推荐系统应该说就是有效的，只是不同方式的有效程度有可能会非常悬殊。

推荐系统中流派比较多，具体落实起来也是千差万别。在没有介绍协同过滤个思路之前，先想想看，利用已有知识可以尝试做这种推荐呢?

贝叶斯分类

朴素贝叶斯分类的思路来做一下推荐。通过统计某用户所有购买物品的分布特性，统计该用户购买物品的分布情况。

如某用户，他在某网站一共购买过100件商品，其中70件是数码产品或小家电，20件是登山用品，10件是其他各类用品。这种情况下，其实拿来一个新的商品，判断要不要推荐给某人，就看他购买的记录的偏重，可以说购买音像制品的概率是最高的，在全部的100件货物里占了40%，其他类别的分析也是同理。所以这种情况下，有理由相信，为这个用户推荐一个音像制品要比什么都不推荐是有更高的转化率的，刚刚的统计结果就是根据。这种做法从思路上应该是没有问题的，但是同样地这些转化率是要进行量化的，只需要记录推荐的商品有多少确实被购买了就知道这种方法的可靠性有多高了。

利用搜索记录

不管是在一些网站的广告位上，还是在京东商城、淘宝等网店上，都有过类似的经历，就是在浏览器中搜索过的东西，会从广告位上显示出来。在网店里，网店会根据用户搜索过的关键词猜测用户想要买某些产品，所以在列出产品列表的同时也会在右侧给用户推荐一些商品。

至于我们发现在其他网站的广告位上也会发现一些有趣的事情，就是曾经在搜索引擎上搜索过的东西会出现在这些网站的广告位上，或者这个网站的广告位上会出现一些与当前页面内容的关键词相关的内容推荐。这是网站广告位的JavaScript代码读取了浏览器的**本地Cookies**（通常可以用来存储浏览器上的表单信息、用户名、搜索关键词等信息）和当前页面的文本信息，并做了相应的关键词提取，最后根据这些关键词来猜测用户可能感兴趣的内容再推荐到广告位上。

有时候也会看到一些让人感觉推荐系统很傻的地方，就是搜索并购买了一些产品之后它还在不停地进行最类似产品的推荐。就是我们平时在网上购物的时候经常看到现象，买了鞋子还在推荐鞋子，这道可以理解，因为鞋子这种很可能一个人会买不止一双。有的推荐就显得很不智能，比如我买了电视还推荐电视，买了电冰箱还推荐电冰箱。

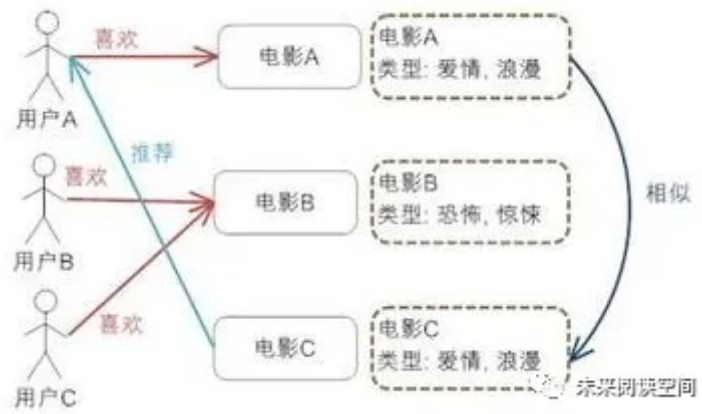
这种根据**购买产品对象的保有率**去对推荐内容做调优的尝试应该不是它最优先改进的内容。

总结, 简单来说根据数据源的不同推荐引擎可以分为三类

- 基于人口的统计学推荐(Demographic-based Recommendation)
- 基于内容的推荐(Content-based Recommendation)
- 基于协同过滤的推荐(Collaborative Filtering-based Recommendation)

补充说明一下基于内容的推荐 (上文利用搜索记录即是一种案例):

根据物品或内容的元数据，发现物品或内容的相关性，然后基于用户以前的喜好记录推荐给用户相似的物品，如图所示:



上图给出了基于内容推荐的一个典型的例子，电影推荐系统，首先我们需要对电影的元数据有一个建模，这里只简单的描述了一下电影的类型；然后通过电影的元数据发现电影间的相似度，因为类型都是"爱情，浪漫"电影 A 和 C 被认为是相似的电影（当然，只根据类型是不够的，要得到更好的推荐，我们还可以考虑电影的导演，演员等等）；最后实现推荐，对于用户 A，他喜欢看电影 A，那么系统就可以给他推荐类似的电影 C。

2）

协同过滤 (Collaborative Filtering, CF)

协同过滤 (Collaborative Filtering, CF) 是一种流行的推荐算法，是一种基于领域的一种算法，其实在1992年就被提出，也算老古董了。其基于系统中其他用户的评分或行为进行预测和推荐。简单来说是利用某兴趣相投、拥有共同经验之群体的喜好来推荐用户感兴趣的信息，个人通过合作的机制给予信息相当程度的回应（如评分）并记录下来以达到过滤的目的进而帮助别人筛选信息，回应不一定局限于特别感兴趣的，特别不感兴趣信息的纪录也相当重要。

这种方法背后的基本假设是:

通过选择和汇总其他用户的意见，以便对活跃用户的偏好提供合理的预测。

直观上，他们认为，如果用户对某些项目的质量或相关性达成一致，那么他们可能会就其他项目达成一致意见 -如果一组用户喜欢与Mary相同的内容，那么Mary很可能喜欢他们喜欢的东西，即便她还没有看到。

协同过滤一般是在海量的用户中发掘出一小部分和你品位比较类似的，在协同过滤中，这些用户成为邻居，然后根据他们喜欢的其他东西组织成一个排序的目录作为推荐给你。

基于内容的推荐 v.s. 协同过滤 (Collaborative Filtering, CF)

基于内容的推荐只考虑了对象的本身性质，将对象按标签形成集合，如果你消费集合中的一个则向你推荐集合中的其他对象；

基于协同过滤的推荐算法，充分利用集体智慧，即在大量的人群的行为和数据中收集答案，以帮助我们对整个人群得到统计意义上的结论，推荐的个性化程度高

当然对于CF其中有一个核心的问题：

- **如何确定一个用户是不是和你有相似的品位?**
- **如何将邻居们的喜好组织成一个排序的目录?**

目前协同过滤公认的应该是两种思路: 邻居Neighborhoods和基于模型的矩阵分解 。

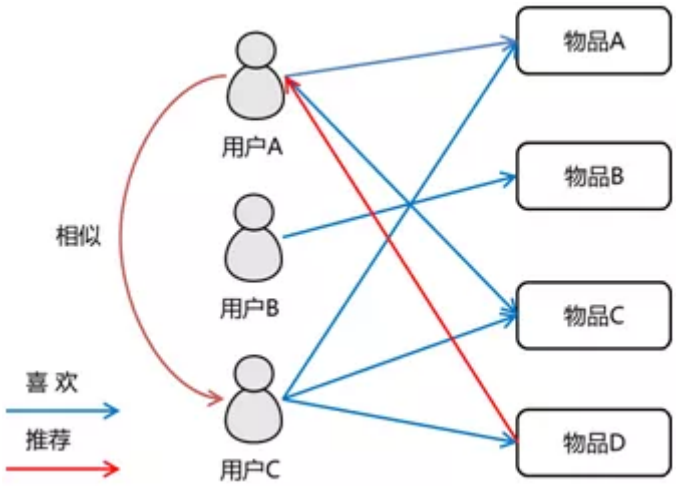
邻居方法 Neighborhoods

而邻居方法中也有两种视角，我们分别来看一下。

1) 第一种，基于用户 (User-based CF)

也就是说，系统通过分析一个用户和哪些用户的特征比较像，然后看看这些用户喜欢买哪类的商品，再从这些商品里挑出一些推荐给该用户。或许在一些银行或者理财产品售卖的机构会有这样的一种列表，利用这种列表同样也能够去观察哪些用户之间更相似，然后找到相似的用户，再把这些用户比较喜好的产品推荐给他。

称为**User-based CF（User-based Collaborative Filtering）**，或者叫基于用户的协同过滤；



实现：将一个用户对所有物品的偏好作为一个向量（Vector）来计算用户之间的相似度，找到K-邻居后，根据邻居的相似度权重以及他们对物品的喜好，为目标用户生成一个排序的物品列表作为推荐，列表里面都是目标用户为涉及的物品。

2) 第二种，基于商品 (Item-based CF)。

也就是说，系统通过分析用户的购买行为来判断用户喜欢的商品类型，然后从那些用户喜欢的商品类型里挑出一些推荐给用户。比如在淘宝里面，推荐栏里面有和该商品类似的商品XXX。这就是基于物品的协同过滤，把你感兴趣的东西的类似属性的东西推荐给你。

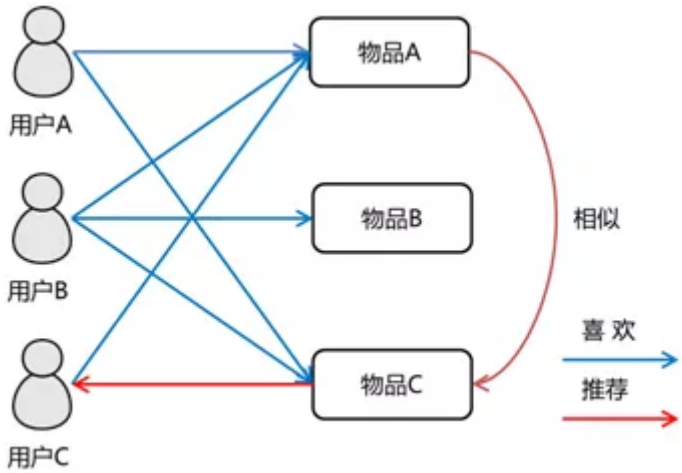
基于物品的协同过滤算法最早是由著名的电商公司亚马逊提出的。这种算法给用户推荐那些和他们之前喜欢的商品相似的商品。但是，这种算法和前面的基于用户的协同过滤算法不一样——它并不是要建立一个商品属性的矩阵来计算物品之间的相似度。其实想想也知道这种方式很可能行不通，一来是由于商品之间的属性相差较大，做起来可能会比较困难，二来是由于计算量太大难以实现。所以这个算法主要通过分析用户的行为来计算物品之间的相似度。

一句话概括就是这样："有很多人喜欢商品A，同时他们也喜欢商品B，所以A和B应该是比较类似的商品。"这就是整个算法的核心思路。

计算起来可以分成以下两个步骤:

（1）计算商品之间的相似度。
（2）根据物品的相似度和用户的偏好来给用户生成推荐列表。

称为**Item-based CF（Item-based Collaborative Filtering）**，或者叫基于商品的协同过滤。



实现：将所有用户对某一个物品的喜好作为一个向量来计算物品之间的相似度，得到物品的相似物品后，根据用户历史的喜好预测目标用户还没有涉及的物品，计算得到一个排序的物品列表作为推荐。

基于用户 (User-based CF)与基于商品 (Item-based CF)算法比较

项目	UserCF	ItemCF
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

不同的业务场景，可以用的算法也不同。ItemCF需要维护一张物品相关度的表，当物品量更新速度太快时，此表的维护在技术上有难度，所以对于电商、音乐、图书等网站而言，ItemCF的优势更大。新闻类网站对于新用户可直接推荐热门新闻即可，为了满足实时性，新闻类更适合用UserCF。

1) 适用场景

UserCF算法：当物品数量远超用户数量时，可以考虑UserCF算法，例如：新闻类、短视频、热点多、社交性质重、需常更新数据的网站或APP；（抖音、微博）
ItemCF算法：当用户数量远远超过物品数量，可以考虑使用Item算法，例如：购物网站、技术博客、文章等不常更新、数据稳定的网站或APP；（小说APP）

2) 推荐系统多样性

- 单用户多样性：ItemCF算法小于UserCF算法多样性丰富，ItemCF算法覆盖面小、丰富度低
- 多用户多样性：ItemCF算法大于UserCF算法多样性丰富，因为UserCF算法注重社会热点物品

ItemCF算法推荐具有新颖性，容易发现推荐长尾里的物品，ItemCF算法计算精度小于UserCF算法。如果考虑多样性，那么ItemCF算法计算精度大于UserCF算法，UserCF算法推荐长尾物品能力不足。

3) 用户特点

UserCF算法适用于用户邻居多情况，例如：微信、微博、支付宝等

ItemCF算法适用于用户喜欢他以前购买过物品类型相似的物品，喜欢同类型物品的自相似度要大的情况

3）

相似度（Similarity）

相似度的计算，现有的几种基本方法都是基于向量（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。在推荐的场景中，在用户 - 物品偏好的二维矩阵中，我们可以将一个用户对所有物品的偏好作为一个向量来计算用户之间的相似度，或者将所有用户对某个物品的偏好作为一个向量来计算物品之间的相似度。

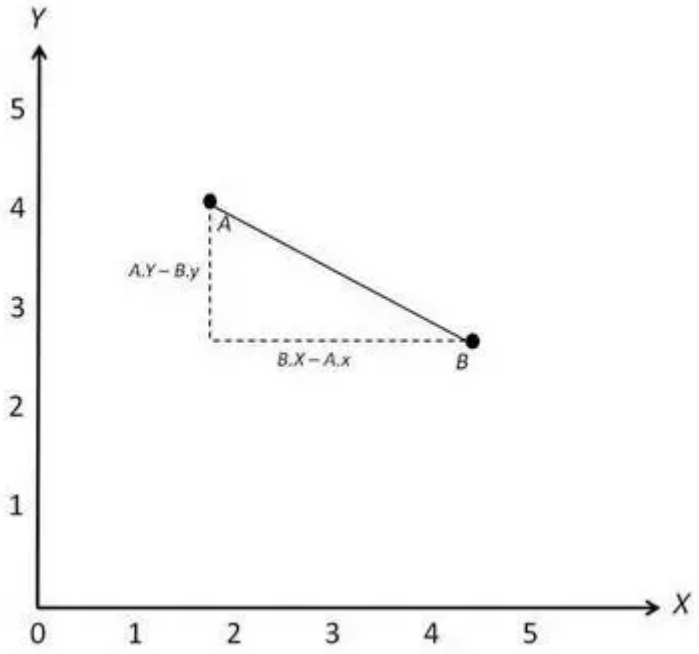
- **欧氏距离（常用）**
- 曼哈顿距离
- **皮尔森相关系数**
- 切比雪夫距离
- 马氏距离
- **夹角余弦距离（常用）**
- 杰卡德相似系数与杰拉德距离
- 相关系数与相关距离

下面详细介绍几种常用的相似度计算方法：

- **欧几里德距离（Euclidean Distance）**

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：

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



当 n=2 时，欧几里德距离就是平面上两个点的距离:

```
# 欧式距离
def EuclideanDistance(a,b):
    return sqrt( (a[0]-b[0])**2 + (a[1]-b[1])**2 )
```

- **皮尔森相关系数（Pearson Correlation Coefficient)**

皮尔森相关系数一般用于计算两个定距变量间联系的紧密程度，它的取值在 [-1，+1] 之间:

$$\rho_{X,Y} = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{(\sum X^2 - \frac{(\sum X)^2}{N})(\sum Y^2 - \frac{(\sum Y)^2}{N})}}$$

皮尔森相关系数当两个变量的线性关系增强时，相关系数趋于1或-1；当一个变量增大，另一个变量也增大时，表明它们之间是正相关的，相关系数大于0；如果一个变量增大，另一个变量却减小，表明它们之间是负相关的，相关系数小于0；如果相关系数等于0，表明它们之间不存在线性相关关系。

用数学公式表示，皮尔森相关系数等于两个变量的协方差除于两个变量的标准差:

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E((X-\mu_X)(Y-\mu_Y))}{\sigma_X \sigma_Y} = \frac{E(XY)-E(X)E(Y)}{\sqrt{E(X^2)-E^2(X)}\sqrt{E(Y^2)-E^2(Y)}}$$

$$\rho_{X,Y} = \frac{N\sum XY - \sum X \sum Y}{\sqrt{N\sum X^2 - (\sum X)^2}\sqrt{N\sum Y^2 - (\sum Y)^2}}$$

$$\rho_{X,Y} = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{(\sum X^2 - \frac{(\sum X)^2}{N})(\sum Y^2 - \frac{(\sum Y)^2}{N})}}$$

上文用到的几个数学公式:

协方差（Covariance）：在概率论和统计学中用于衡量两个变量的总体误差。如果两个变量的变化趋于一致，也就是说如果其中一个大于自身的期望值，另一个也大于自身的期望值，那么两个变量之间的协方差就是正值；如果两个变量的变化趋势相反，则协方差为负值。

$$\operatorname{cov}(X,Y) = \mathrm{E}((X-\mu)(Y-$$

其中u表示X的期望E(X), v表示Y的期望E(Y)

标准差（Standard Deviation）：标准差是方差的平方根

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

$$\sigma = \sqrt{\mathrm{E}((X-\mathrm{E}(X))^2)} = \sqrt{\mathrm{E}(X^2) - (\mathrm{E}(X))^2}$$

方差(Variance)：在概率论和统计学中，一个随机变量的方差表述的是它的离散程度，也就是该变量与期望值的距离

$$Var(X) = E(X^2) - E^2(X)$$

即方差等于误差的平方和的期望

> **数学参考** [技术]《白话大数据与机器学习》- 1

基于皮尔森相关系数的相似度有两个缺点：

(1) 没有考虑用户间重叠的评分项数量对相似度的影响；
(2) 如果两个用户之间只有一个共同的评分项，相似度也不能被计算

比如两个用户共同观看了200部电影，虽然不一定给出相同或完全相近的评分，他们之间的相似度也应该比另一位只观看了2部相同电影的相似度高吧！但事实并不如此，如果对这两部电影，两个用户给出的相似度相同或很相近，通过皮尔森相关性计算出的相似度会明显大于观看了相同的200部电影的用户之间的相似度。

- **Cosine 相似度（Cosine Similarity）**

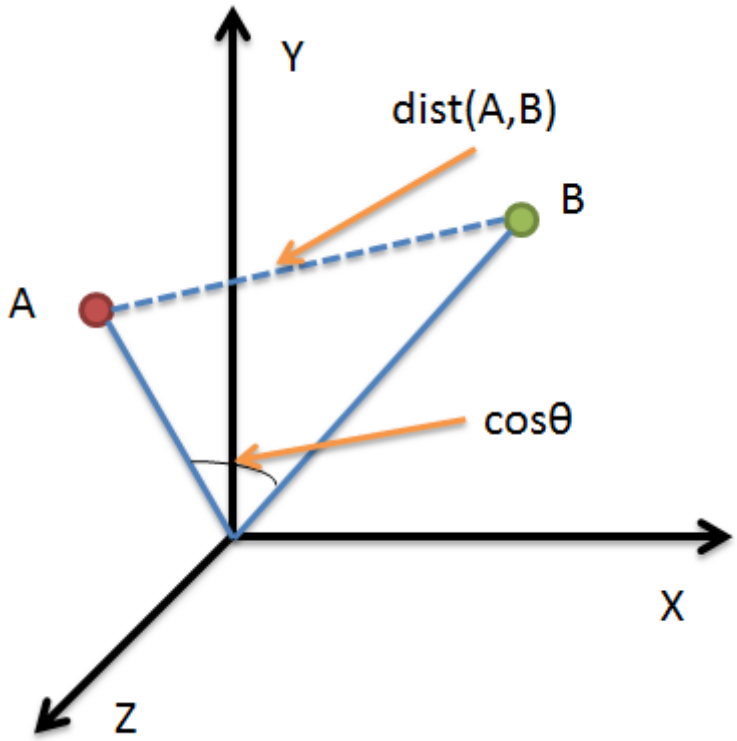
Cosine 相似度被广泛应用于计算文档数据的相似度：

$$sim(X,Y) = cos\theta = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \cdot \|\vec{y}\|}$$

余弦相似度用向量空间中两个向量夹角的余弦值作为衡量两个个体间差异的大小。相比距离度量，余弦相似度更加注重两个向量在方向上的差异，而非距离或长度上。

与欧几里德距离类似，基于余弦相似度的计算方法也是把用户的喜好作为n-维坐标系中的一个点，通过连接这个点与坐标系的原点构成一条直线（向量），两个用户之间的相似度值就是两条直线（向量）间夹角的余弦值。因为连接代表用户评分的点与原点的直线都会相交于原点，夹角越小代表两个用户越相似，夹角越大代表两个用户的相似度越小。同时在三角系数中，角的余弦值是在[-1, 1]之间的，0度角的余弦值是1，180角的余弦值是-1。

借助三维坐标系来看下欧氏距离和余弦相似度的区别：



从图上可以看出距离度量衡量的是空间各点间的绝对距离，跟各个点所在的位置坐标（即个体特征维度的数值）直接相关；而余弦相似度衡量的是空间向量的夹角，更加的是体现在方向上的差异，而不是位置。如果保持A点的位置不变，B点朝原方向远离坐标轴原点，那么这个时候余弦相似度cosθ是保持不变的，因为夹角不变，而A、B两点的距离显然在发生改变，这就是欧氏距离和余弦相似度的不同之处。

```
# 二维余弦
def CosineSimilarity(a,b):
    cos = (a[0]*b[0]+a[1]*b[1]) / (sqrt(a[0]**2 + a[1]**2) * sqrt(b[0]**2 + b[1

    return cos
print('a,b 二维夹角余弦距离：',CosineSimilarity((1,1),(2,2)))
```

根据欧氏距离和余弦相似度各自的计算方式和衡量特征，分别适用于不同的数据分析模型：欧氏距离能够体现个体数值特征的绝对差异，所以更多的用于需要从维度的数值大小中体现差异的分析，如使用用户行为指标分析用户价值的相似度或差异；而余弦相似度更多的是从方向上区分差异，而对绝对的数值不敏感，更多的用于使用用户对内容评分来区分用户兴趣的相似度和差异，同时修正了用户间可能存在的度量标准不统一的问题（因为余弦相似度对绝对数值不敏感）。

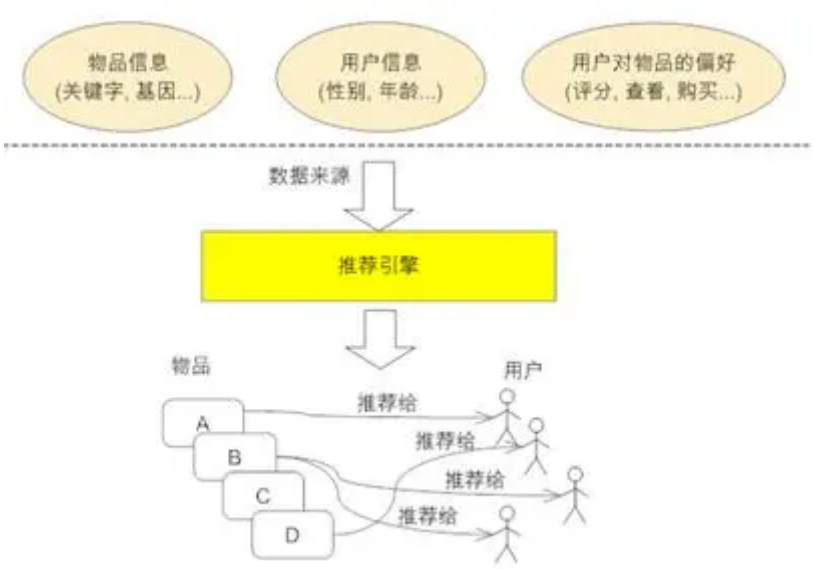
调整余弦相似度 —— Adjusted Cosine Similarity

余弦相似度更多的是从方向上区分差异，而对绝对的数值不敏感。因此没法衡量每个维数值的差异，会导致这样一个情况：比如用户对内容评分，5分制，X和Y两个用户对两个内容的评分分别为(1,2)和(4,5)，使用余弦相似度得出的结果是0.98，两者极为相似，但从评分上看X似乎不喜欢这2个内容，而Y比较喜欢，余弦相似度对数值的不敏感导致了结果的误差，需要修正这种不合理性，就出现了调整余弦相似度，即所有维度上的数值都减去一个均值，比如X和Y的评分均值都是3，那么调整后为(-2,-1)和(1,2)，再用余弦相似度计算，得到-0.8，相似度为负值并且差异不小，但显然更加符合现实。

4）

推荐引擎概述

推荐引擎利用特殊的信息过滤技术，将不同的物品或内容推荐给可能对它们感兴趣的用户。



将推荐引擎看作黑盒，它接受的输入是推荐的数据源，一般情况下，推荐引擎所需要的数据源包括：

- 要推荐物品或内容的元数据，例如关键字，基因描述等；
- 系统用户的基本信息，例如性别，年龄等
- 用户对物品或者信息的偏好，根据应用本身的不同，可能包括用户对物品的评分，用户查看物品的记录，用户的购买记录等。其实这些用户的偏好信息可以分为两类：
- **显式的用户反馈**：这类是用户在网站上自然浏览或者使用网站以外，显式的提供反馈信息，例如用户对物品的评分，或者对物品的评论。

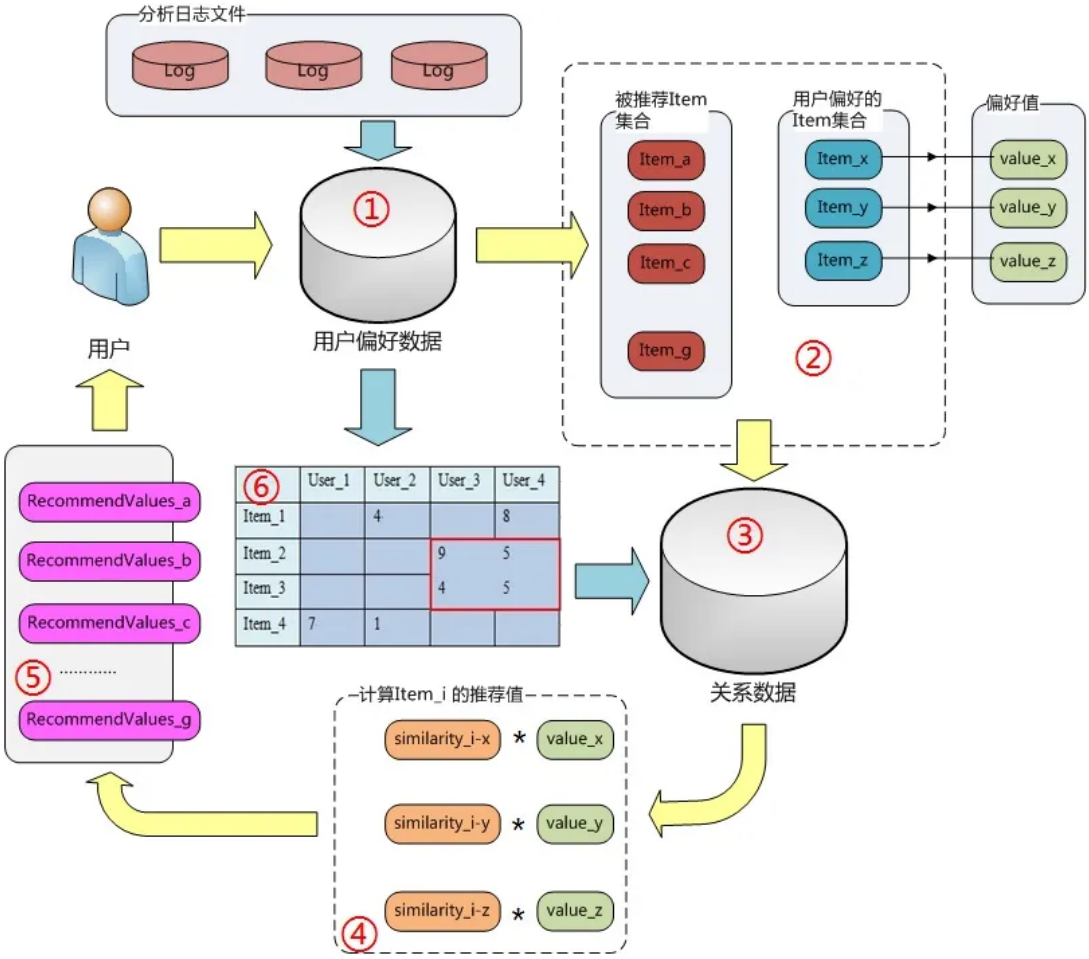
- **隐式的用户反馈**：这类是用户在使用网站是产生的数据，隐式的反应了用户对物品的喜好，例如用户购买了某物品，用户查看了某物品的信息等等。
-

显式的用户反馈能准确的反应用户对物品的真实喜好，但需要用户付出额外的代价，而隐式的用户行为，通过一些分析和处理，也能反映用户的喜好，只是数据不是很精确，有些行为的分析存在较大的噪音。但只要选择正确的行为特征，隐式的用户反馈也能得到很好的效果，只是行为特征的选择可能在不同的应用中有很大的不同，例如在电子商务的网站上，购买行为其实就是一个能很好表现用户喜好的隐式反馈。

5）

基于邻居方法 Neighborhoods的推荐算法系统实现

- **基于商品 (Item-based CF)**



①. 查找这个用户喜欢过的物品（即偏好的产品，并查出偏好值后面会用），以及还没有喜欢过的商品，前者是推荐运算的根据，后者作为一个产生推荐的一个集合；如② 画的那样。

②. 这里是一个可扩展的地方；因为这两部分的数据的作用非常明显，修改这两个集合对后面产生的推荐结果可产生非常直观的影响，比如清洗过滤，或根据用户属性缩小集合；不仅使后面推荐效果更优，运算性能也可以大幅度提高。

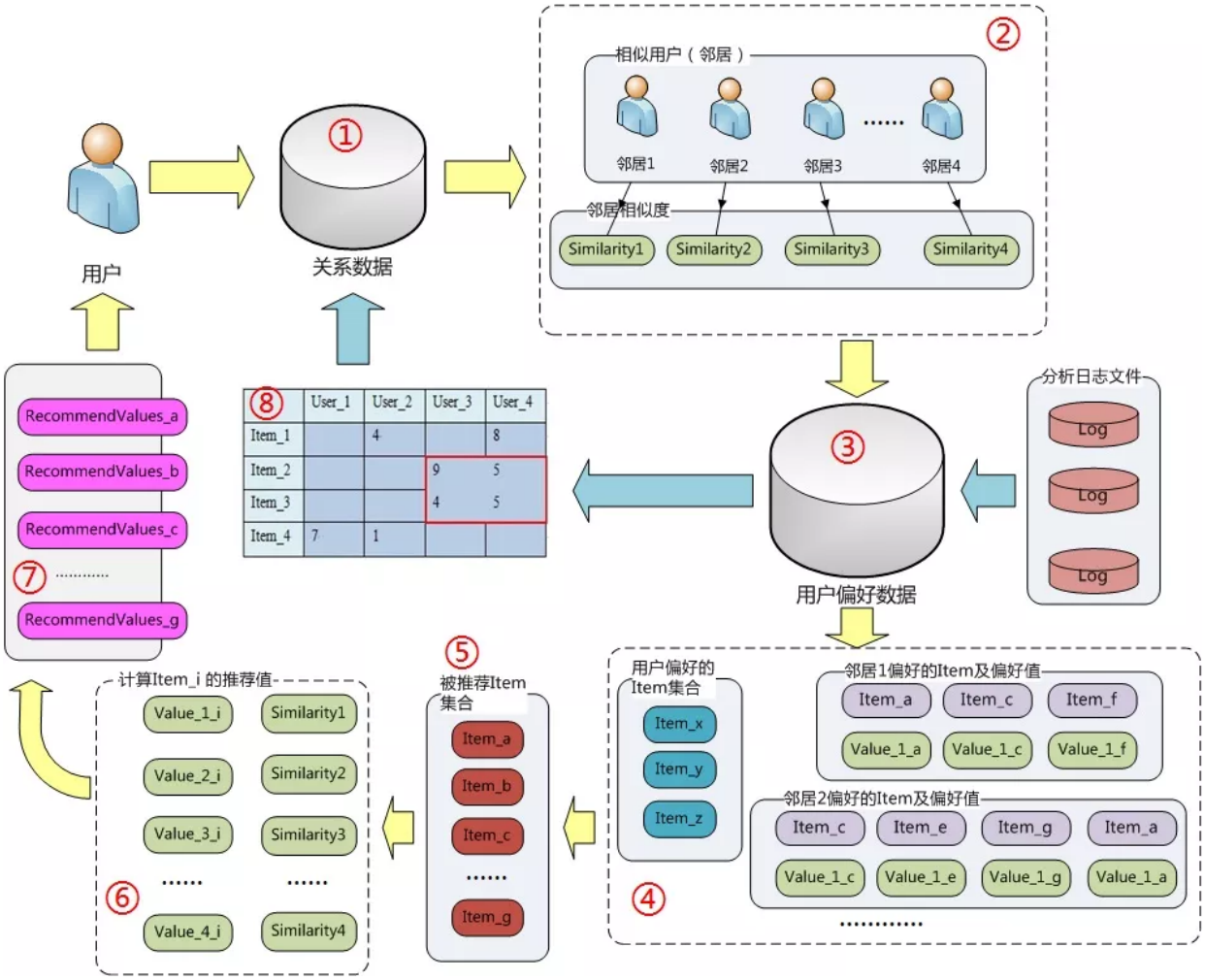
③. 查找这两个集合之间的关系，这是一对多的关系：一个没有偏好过的物品与该用户所有偏好过的物品间的关系，有一个值来衡量这个关系叫相似度Similarity；这个关系怎么来的，看蓝色箭头的指向。步骤⑥

④. 得到这个一对多的关系后，就可以计算这个物品对于这个用户的推荐值了，图中similarity_i-x表示Item_i 与 Item_x 之间的相似度，Item_x是该用户偏好过得，该用户对其偏好值记为 value_x ，相乘；Item_i 与 该用户偏好过的所有物品以此做以上运算后，得到的值取平均值 便是 Item_i的推荐值了。注：有可能Item_i 不是与所有 该用户偏好过的物品都都存在相似性，不存在的，不计算即可；另外这里方便理解介绍的都是最简单的实现；你也可以考一些复杂的数学元素，比如方差来判断离散性等。

⑤. 这步就简单多了，刚才对该用户没有偏好过的集合中的所有Item都计算了推荐值，这里就会得到一个list，按推荐值由大到小排序，返回前面的一个子集即可。

⑥前面已经提到，关系数据时怎么来的，也是根据用户的偏好数据；你把其看成一个矩阵，横着看过来，参考两个Item间的共同用户，以及共同用户的偏好的值的接近度；这里的可选择的相似度算法很多。

- **基于用户 (User-based CF)**



①. 同样也是查询，只是查询的对象不一样了，查询的是与该用户相似的用户，所以一来直接查了关系数据源。以及相似用户与该用户的相似度。

②. 与刚才类似，也是对数据集的一个优化，不过作用可能没那么大。

③. 查询关系数据源，得到相似用户即邻居偏好过的物品；如步骤④；图中由于空间小，没有把所有邻居的偏好关系都列出来，用……表示。其次还要得到该用户偏好过的物品集合。

④. 被推荐的Item集合是由该用户的所有邻居的偏好过的物品的并集，同时再去掉该用户自己偏好过的物品。作用就是得到你的相似用户喜欢的物品，而你还没喜欢过的。

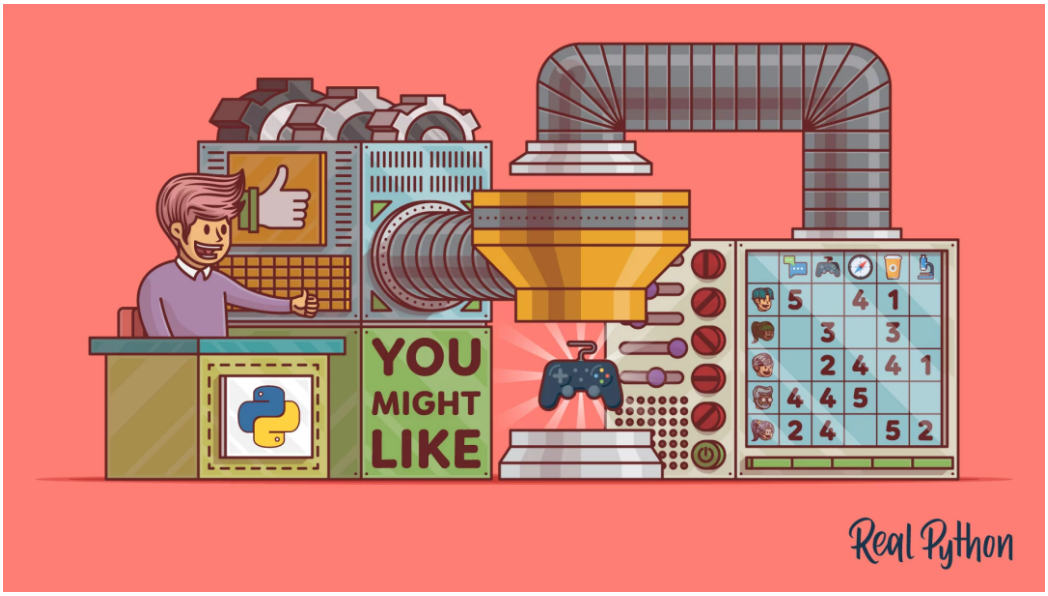
⑤. 集合优化同基于物品的协同过滤算法的步骤②。

⑥. 也是对应类似的，依次计算被推荐集合中Item_i 的推荐值，计算的方式略有不同，Value_1_i表示邻居1对，Item_i的偏好值，乘以该用户与邻居1的相似度 Similarity1；若某个邻居对Item_i偏好过，就重复上述运算，然后取平均值；得到Item_i的推荐值。

⑦、⑧. 与上一个算法的最后两部完全类似，判断两个用户相似的法子和判断两个物品相似的法子一样。

5）

基于模型的推荐算法（Matrix Factorization）



基于模型的协同过滤推荐算法通过分析用户和项目的内部规律模型，预测用户对项目的偏好，其中概率矩阵分解技术是其典型代表。

基于矩阵分解的方法（LFM）：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)。

基于模型的协同过滤推荐算法

- **矩阵分解模型**
- **交替最小二乘**
- **概率矩阵分解**

基于模型的协同算法这里不展开了，可以参考前文:

> [技术杂谈]《机器学习算法实践:推荐系统的协同过滤理论及其应用》

4

总结

《白话大数据与机器学习》，第五部分 - 推荐算法。

基于协同过滤的推荐策略的基本思想就是基于大众行为，为每个用户提供个性化的推荐，从而使用户能更快速更准确的发现所需要的信息。

白话是洞察一切却能深入浅出地娓娓道来，是高屋建瓴地化繁为简，总之得先有高度，然后还得接地气把小白都讲明白——这才敢叫白话。

请点击底部“在看”支持，分享精品！

“可惜，

世界上最聪明的大脑，想的都是如何推荐，
怎么样让用户去点击广告。”