【推荐算法】基于内存的协同过滤算法

后来遇见AI 2020-10-14

1 协同过滤

顾名思义，它是通过集体智慧的力量来进行工作，过滤掉那些用户不感兴趣的项目，筛选出用户感兴趣的内容。
核心思想：**物以类聚，人以群分**。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：基于内存的协同过滤推荐（Memory-based CF）和基于模型的协同过滤推荐(Model-based CF)。本文主要讨论基于内存的协同过滤方法。

2 基于内存的协同过滤（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**。选择合适的相似性度量指标来更好的表达两个项目或者用户的相似性是关键；最简单的推荐方法是**基于大多数的推荐策略**，即推荐那些大多数人产生过行为而目标用户未产生过行为的项目。
基于内存的协同过滤方法根据用户维度和项目维度的不同又可以分为Item-based CF和User-based CF。

- “跟你喜好相似的人喜欢的东西你也很有可能喜欢”：基于用户的协同过滤推荐（User-based CF）
- “跟你喜欢的东西相似的东西你也很有可能喜欢”：基于物品的协同过滤推荐（Item-based CF）

2.1 基于用户的协同过滤算法

举个例子，在学校时，刚进实验室的师弟都会询问学长一些诸如“我应该买什么专业书啊”、“我应该看什么论文啊”的问题。师弟之所以请教师兄，一方面是因为他们有社会关系，互相认识且信任对方，但更主要的原因是师兄和师弟有共同的研究领域和兴趣。
那么，在一个在线个性化推荐系统中，当一个用户A需要个性化推荐时，可以先找到和他有相似兴趣的其他用户，然后把那些用户喜欢的、而用户A没有听说过的物品推荐给A。基于用户的协同过滤算法主要包括以下4个步骤：

- （1）构建User-Item矩阵；

- （2）根据UI矩阵来计算行（用户维度）的相似度；
- （3）选择特定用户最相似的k个用户；
- （4）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项目。
  UserCF算法有一些缺点：随着网站的用户数目越来越大，计算用户兴趣相似度矩阵将越来越困难，其运算时间复杂度和空间复杂度的增长和用户数的增长近似于平方关系。

2.2 基于物品的协同过滤算法

著名的电子商务公司亚马逊提出了基于物品的协同过滤算法。基于物品的协同过滤算法（简称ItemCF）给用户推荐那些和他们之前喜欢的物品相似的物品。不过，ItemCF算法并不利用物品的内容属性计算物品之间的相似度，它主要通过分析用户的行为记录计算物品之间的相似度。该算法认为，物品A和物品B有很大的相似度是因为喜欢物品A的用户大都也喜欢物品B。基于物品的协同过滤算法可以利用用户的历史行为给推荐结果提供推荐解释，比如给用户推荐《天龙八部》的解释可以是因为用户之前喜欢《射雕英雄传》。亚马逊显示相关物品推荐时的标题是“Customers Who Bought This Item Also Bought”（购买了该商品的用户也经常购买的其他商品）。基于物品的协同过滤算法主要包括以下4个步骤：

- （1）构建User-Item矩阵；
- （2）根据UI矩阵来计算列（物品维度）的相似度；
- （3）选择特定物品最相似的k个物品；
- （4）推荐给特定用户还没有发生过行为的项目。

2.3 UserCF 和 ItemCF 的综合比较

从两个算法的原理可以看到，UserCF 的推荐结果着重于反映和用户兴趣相似的小群体的热点，而ItemCF 的推荐结果着重于维系用户的历史兴趣。换句话说，UserCF的推荐更社会化，反映了用户所在的小型兴趣群体中物品的热门程度，而ItemCF的推荐更加个性化，反映了用户自己的兴趣传承。在新闻网站中，用户的兴趣不是特别细化，绝大多数用户都喜欢看热门的新闻。即使是个性化，也是比较粗粒度的，比如有些用户喜欢体育新闻，有些喜欢社会新闻，UserCF 可以给用户推荐和他有相似爱好的一群其他用户今天都在看的新闻，这样在抓住热点和时效性的同时，保证了一定程度的个性化。UserCF 适合用于新闻推荐的另一个原因是从技术角度考量的。因为作为一种物品，新闻的更新非常快，每时每刻都有新内容出现，而ItemCF 需要维护一张物品相关度的表，如果物品更新很快，那么这张表也需要很快更新，这在技术上很难实现。绝大多数物品相关度表都只能做到一天一次更新，这在新闻领域是不可以接受的。而UserCF只需要用户相似性表，虽然UserCF对于新用户也需要更新相似度表，但在新闻网站中，物品的更新速度远远快于新用户的加入速度，而且对于新用户，完全可以给他推

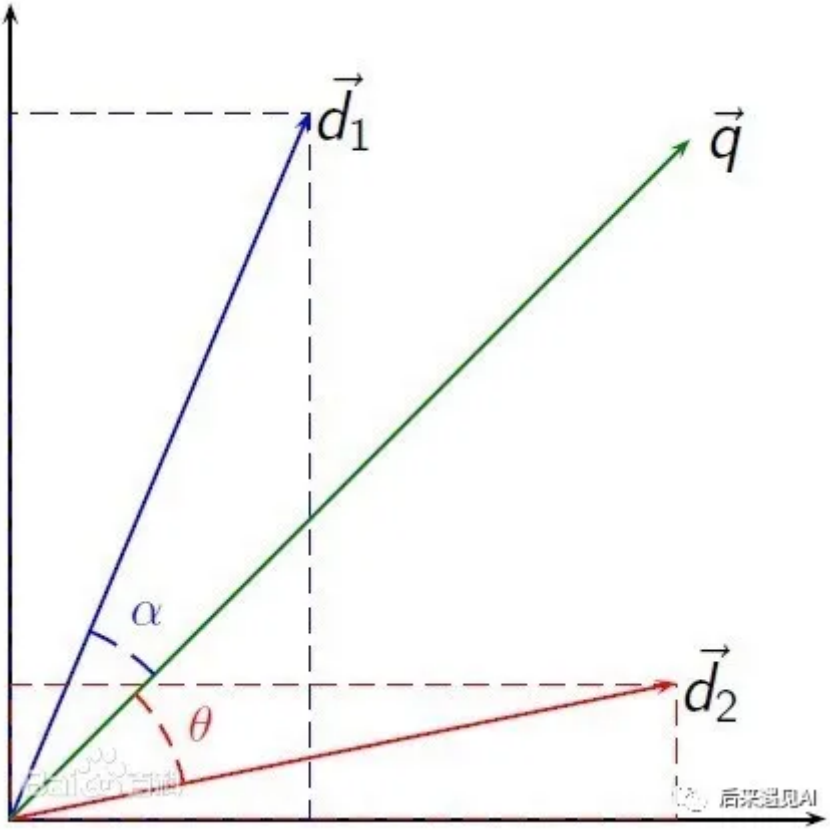
荐最热门的新闻，因此UserCF显然是利大于弊。但是，在图书、电子商务和电影网站，比如亚马逊、豆瓣、Netflix中，ItemCF 则能极大地发挥优势。首先，在这些网站中，用户的兴趣是比较固定和持久的。这些系统中的用户大都不太需要流行度来辅助他们判断一个物品的好坏，而是可以通过自己熟悉领域的知识自己判断物品的质量。因此，这些网站中个性化推荐的任务是帮助用户发现和他研究领域相关的物品。此外，这些网站的物品更新速度不会特别快，一天一次更新物品相似度矩阵对它们来说不会造成太大的损失，是可以接受的。

从技术上考虑，UserCF需要维护一个用户相似度的矩阵，而ItemCF需要维护一个物品相似度矩阵。从存储的角度说，如果用户很多，那么维护用户兴趣相似度矩阵需要很大的空间，同理，如果物品很多，那么维护物品相似度矩阵代价较大。在图书、电子商务网站中，物品的数目则是比较少的。新闻网站是个例外，在那儿，物品的相似度变化很快，物品数目庞大。

	UserCF	**ItemCF**
性能	适用于用户数量较少的场景，如果用户数量过大，很难维护巨大的用户相似度矩阵	适用于商品数量远小于用户数量的场景，如果商品数量很大，计算商品相似度矩阵代价很大
领域	时效性强，用户个性化兴趣不强的领域，如新闻	长尾物品丰富，用户个性化兴趣丰富强烈
时效性	用户有新行为不一定造成推荐结果立即变化	用户有新行为一定会造成推荐结果实时变化
冷启动	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，因为用户相似度矩阵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他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，就可以给他推荐与该物品相似的其他物品。无法在不离线更新物品相似度矩阵的条件下将新物品推荐给用户
推荐理由	很难给出让用户信服的解释	利用用户的历史行为给用户做出解释，容易让人信服

3 相似性度量指标

3.1 cosine相似度



给定两个属性向量，A 和 B，其余弦相似性θ由点积和向量长度给出，如下所示：

$$sim(A,B) = cos\theta = \frac{A \cdot B}{|A| \cdot |B|} = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

3.2 pearson相似度

Pearson相关系数是用协方差除以两个变量的标准差得到的（协方差大于0的时候表示两者正相关，小于0的时候表示两者负相关）。

$$协方差\ Cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{n-1}$$

$$sim(A,B) = \frac{Cov(X,Y)}{\sigma_X \cdot \sigma_Y}$$

当数据标准化以后（$\mu = 0, \sigma = 1$）Pearson相关系数与余弦相似度、欧氏距离等价。具体可参考如何理解皮尔逊相关系数（Pearson Correlation Coefficient）？

3.3 MSD均方差相似度

仍然考虑物品i和j的相似度，MSD考虑的角度为同时喜欢物品i和j的用户对于这两个物品的评分差距程度：

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

上式表示均方差，值越小，物品i和j相似度越大。为了与之前的相似度表示一致（值越大，物品相似度越大），定义相似度为：

$$msdsim(i,j) = \frac{1}{msd(i,j)+1}$$

4 评分预测

4.1 基于相似用户的加权平均

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} \mathrm{sim}(u,v) \cdot r_{vi}}{\sum\limits_{v \in N_i^k(u)} \mathrm{sim}(u,v)}$$

or

$$\hat{r}_{ui} = \frac{\sum\limits_{j \in N_u^k(i)} \mathrm{sim}(i,j) \cdot r_{uj}}{\sum\limits_{j \in N_u^k(i)} \mathrm{sim}(i,j)}$$

式中$N_i^k(u)$表示对当前商品i有评分行为的其他用户中与当前用户u最相似的k个用户。

4.2 基于相似用户并考虑用户评分均值

$$\hat{r_{ui}} = \mu_u + \frac{\sum_{v \in N_i^k(u)} sim(u,v) \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

or

$$\hat{r_{ui}} = \mu_i + \frac{\sum_{j \in N_u^k(i)} sim(i,j) \cdot (r_{uj} - \mu_j)}{\sum_{j \in N_u^k(i)} sim(i,j)}$$

式中μ_u表示用户u评分的平均值。

4.3 基于相似用户并考虑用户评分均值、方差

$$\hat{r_{ui}} = \mu_u + \sigma_u \frac{\sum_{v \in N_i^k(u)} sim(u,v) \cdot (r_{vi} - \mu_v)/\sigma_v}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

or

$$\hat{r_{ui}} = \mu_i + \sigma_i \frac{\sum_{j \in N_u^k(i)} sim(i,j) \cdot (r_{uj} - \mu_j)/\sigma_j}{\sum_{j \in N_u^k(i)} sim(i,j)}$$

式中σ_u表示用户u评分的标准差。

4.4 基于相似用户并考虑用户评分偏置

$$\hat{r_{ui}} = b_{ui} + \frac{\sum_{v \in N_i^k(u)} sim(u,v) \cdot (r_{vi} - b_{vi})}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

or

$$\hat{r_{ui}} = b_{ui} + \frac{\sum_{j \in N_u^k(i)} sim(i,j) \cdot (r_{uj} - b_{uj})}{\sum_{j \in N_u^k(i)} sim(i,j)}$$

其中$b_{ui} = \mu + b_u + b_i$是基线模型中用户u给物品i打分的预估值。μ为所有用户评分的均值。b_u为user偏差（如果某用户比较苛刻，打分都相对偏低，则会为负值。相反，如果某用户经常对很多商品都打正分，则bu为正值）。为item偏差，反映商品受欢迎程度。