机器学习算法系列（13）：推荐系统（2）—基于领域的协同过滤

基于邻域的算法是推荐系统中最基本的算法，在学术界和业界都有广泛研究与应用。它分为两大类，一类是基于用户的协同过滤算法，另一类是基于物品的协同过滤算法。

一、基于用户的协同过滤算法（user-based collaborative filtering）

UserCF是推荐系统的元老级算法，也是最为著名的算法，它标志了推荐系统的诞生。这里首先介绍最基础的算法，然后在此基础上提出不同的改进方法。

1.1 基础算法

在一个在线个性推荐系统中，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，然后将那些用户喜欢的、而用户A没有听说过的物品推荐给A。这种方法就是**基于用户的协同过滤算法**

用户/物品	物品A	物品B	物品C	物品D
用户A	√		√	
用户B		√		
用户C	√		√	√

如上图，我们收集到用户-电影评价矩阵，假设用户A对于物品D的评价为NULL，这时我们对比用户A、用户B、用户C的特征向量（以物品评价为特征），可以发现用户A和用户C的相似度较大，这时我们可以认为，对于用户C喜欢的物品D，用户A也应该喜欢它，这时就把物品D推荐给用户A。

用户/物品	物品A	物品B	物品C	物品D
用户A	√		√	推荐
用户B		√		
用户C	√		√	√

UserCF主要包括两个步骤：

- 1）寻找和目标用户兴趣相似的用户集合
  这里的关键就是计算两个用户的兴趣相似度，这里，协同过滤算法主要利用行为的相似度计

算兴趣的相似度。给定用户u和用户v，令$N(u)$表示用户u曾经有过正反馈（用户的行为倾向于指用户喜欢该物品，反之，负反馈指用户的行为倾向于指用户不喜欢该物品）的物品集合，令$N(v)$为用户v曾经有过正反馈的物品集合。我们可以使用Jaccard或者余弦相似度来计算它们之间的兴趣相似度。

- Jaccard公式：

$$w_{uv} = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$$

- 余弦相似度：

$$w_{uv} = \frac{|N(u) \cap N(v)|}{\sqrt{|N(u)||N(v)|}}$$

- 2）找到这个集合中的用户喜欢的，且目标用户没有听说过的物品推荐给目标用户。

1.2 离线算法评测

书中通过MovieLens数据集上的离线试验来测评算法的性能。UserCF只有一个重要参数K，即为每个用户选出K个和他兴趣最相似的用户，然后推荐那K个用户感兴趣的物品。下图为选择不同的K值时算法的性能。

表2-4　MovieLens数据集中UserCF算法在不同K参数下的性能

K	准　确　率	召　回　率	覆　盖　率	流　行　度
5	16.99%	8.21%	51.33%	6.813293
10	20.59%	9.95%	41.49%	6.978854
20	22.99%	11.11%	33.17%	7.10162
40	24.50%	11.83%	25.87%	7.203149
80	**25.20%**	**12.17%**	**20.29%**	**7.289817**
160	24.90%	12.03%	15.21%	7.369063

逐一分析各个指标：

- 准确率和召回率：准确率、召回率与K不成线性关系，在此数据集中，K=80左右会获得比较高的准确率和召回率。选择合适的K对获得高的推荐系统精要比较重要，但对K值不是很敏感，保持在一定区域内即可。
- 流行度：K越大则UserCF推荐结果就越热门，流行度就越高。因为K决定了UserCFA给你做推荐时参考多少和你兴趣相似的其他用户的兴趣，如果K越大，参考的人越多，结果就越来越趋近于全局热门的物品。
- 覆盖率：覆盖率随着K的增大而减小。因为随着K的增大，UserCF越来越倾向于推荐热门的物品，从而对长尾物品的推荐越来越少，覆盖率就越来越小了

此外对于Random算法（每次随机挑选10个用户没有产生过行为的物品推荐给当前用户）和MostPopular算法（按照物品的流行度给用户推荐他没有产生过行为的物品中最热门的10个物品）这两种基础算法（选取K=80），MostPopular算法准确率和召回率很高，但覆盖率非常低。与这两个极端相比，UserCF的准确率和召回率高得多，覆盖率也很高。

1.3 用户相似度计算的改进

用余弦相似度来度量用户之间兴趣相似度过于粗糙，因为两个用户对热门物品采取过相同的行为并不能说明他们兴趣相似，但对于冷门物品才去过相同的行为更能说明他们兴趣的相似度。John S.Breese提出了以下公式：

$$w_{uv} = \frac{\sum_{i \in N(u) \cap N(v)} \frac{1}{\log(1+|N(i)|)}}{\sqrt{|N(u)||N(v)|}}$$

该公式通过$1/\log(1+|N(i)|)$惩罚了用户u和用户v共同兴趣列表中热门物品对他们相似度的影响。将基于上述用户相似度公式的UserCF算法记为User-IIF算法。

1.4 实际应用

相比基于物品的协同过滤算法ItemCF，UserCFA在目前的实际应用中使用并不多。其中最著名的使用者是$Digg$，它的推荐思路为：
用户在$Digg$中主要通过"顶"和"踩"两种行为表达自己对文章的看法。当用户顶了一篇文章，$Digg$就认为该用户对这篇文章有兴趣，而且愿意把这篇文章推荐给其他用户。然后$Digg$找到所有在该用户顶文章之前也顶了这一篇文章的其他用户，然后给他推荐那些人最近顶的其他文章。

二、基于物品的协同过滤算法（item-based collaborative filtering）

基于物品的协同过滤算法是目前业界应用最多的算法。亚马逊、Netflix、Hulu、Youtube的推荐算法的基础都是ItemCF。

2.1 基础算法

UserCF存在一些缺点：

- 1）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）很难对推荐结果作出解释

所以，亚马逊提出了基于物品的协同过滤算法。它给用户推荐那些和他们之前喜欢的物品相似的物品，主要通过分析用户的行为记录计算物品之间的相似度，它认为物品A和物品B具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，它也可以利用用户的历史行为给推荐结果提供解释。

用户/物品	物品A	物品B	物品C
用户A	√		√
用户B	√	√	√
用户C	√		推荐

同样，我们对比物品A、物品B、物品C的特征向量（以用户对该物品的喜好程度为特征），发现物品A和物品C很像，就把用品C推荐给喜欢物品A的用户C。

ItemCF主要包括两个步骤：

- 1）计算物品之间的相似度

可以使用下面的公式定义物品的相似度：

$$w_{ij} = \frac{|N(i) \cap N(j)|}{|N(i)|}$$

分母$|N(i)|$是喜欢物品i的用户数，而分子$|N(i) \cap N(j)|$是同时喜欢物品i和物品j的用户数。可以理解为喜欢物品i的用户中有多少比例的用户也喜欢物品j。但是如果j很热门，很多人喜欢，那么w_{ij}就会很大，接近1，也就是会造成任何物品都会和热门的物品有很大的相似度。为了避免推荐出热门的物品，可以使用下面的公式：

$$w_{ij} = \frac{|N(i) \cap N(j)|}{\sqrt{|N(i)||N(j)|}}$$

它惩罚了物品j的权重，因此减轻了热门物品会和很多物品相似的可能性。从上面的定义可以看到，在协同过滤中两个物品产生相似度是因为它们共同被很多用户喜欢，也就是每个用户都可以通过它们的历史兴趣列表给物品“贡献”相似度。

- 2）根据物品的相似度和用户的历史行为给用户生成推荐列表