协同过滤

协同过滤也叫基于临近的推荐算法，主要思想是：利用已有的用户群过去的行为或者意见来预测数据，根据和当前用户/当前商品比较相似的近邻数据来产生推荐结果。

主要的实现方式有：

-基于用户的最近邻推荐UserCF

-基于物品的最近邻推荐ItemCF

1.UserCF

主要思想：通过评分数据集和当前用户ID，找出与当前用户过去有相似偏好的其他用户，利用这些用户做最近邻，然后通过这些最近邻的用户对当前用户没有见过的每个产品p做评分预测，最后选择所有产品评分最高的TopN个产品推荐给当前用户。

1.1 计算流程

1. 计算所有用户和用户之间的相似度矩阵（基于用户之间共同评论的商品）
2. 计算当前用户u对于当前商品i的评分（2.1获取当前用户u的最近邻用户--【要求这些最近邻用户对当前商品i都有评分】；2.2 根据K个近邻用户对商品i的评分预测当前用户u对商品i的评分）
3. 重复步骤2计算出当前用户对所有物品的评分。
4. 重复步骤2、3计算所有用户对所有物品的评分
5. 对于每个用户，提取该用户对所有物品评分排序后，评分最高的N个商品作为推荐商品。

   注意（在提取商品列表的时候，可以考虑将用户已访问的商品去除）

1.2 UserCF中基于k个近邻用户对商品评分预测的3种方法

1.2.1 直接对分数进行拟合

$$r_{ui} = \frac{\sum_{v \in N_i^k(u)} sim(u,v) * r_{vi}}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

其中，$\hat{r}_{ui}$表示预测用户u对商品i的评分，k表示用户u的k个近邻用户，u表示用户u，i表示商品i。

1.2.2 拟合分数到均值的偏差(基于normal) ——考虑用户偏好

$$r_{ui} = \bar{r}_u + \frac{\sum_{v \in N_i^k(u)} sim(u,v) * (r_{vi} - \bar{r}_v)}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

其中，$\bar{r}_u$表示用户u对所有商品的平均评分，$\bar{r}_v$表示近邻用户v对所有商品的平均评分。

1.2.3 对均值做基线转换

进行基线转换，使用baseline的值来代替均值即可。因为均值体现的是当前用户在所有物品中的评分均值，而baseline可以认为是当前用户在当前物品上可能的评分，通过计算相似用户实际评分和baseline可能评分之间的差值从而可以得到当前用户的预测评分(**预测评=baseline+可能的差值**)。

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

其中，b_{ui}表示baselineonly算法中的$b_{ui} = \mu + b_u + b_i$。

2.ItemCF

主要思想是基于物品之间的相似度，而不是基于用户之间的相似度来进行预测评分值。同UserCF相比只有一个主要区别点：UserCF计算的是用户之间的相似度，从而是将相似用户喜好的物品推荐给当前用户；ItemCF中计算的是物品与物品之间的相似度，从而是根据当前用户喜好的物品来推荐其它物品列表。

2.1 计算流程

1. 计算所有商品和商品之间的相似度矩阵（基于商品之间共同被相同用户评论）
2. 计算当前用户u对于当前商品i的评分（2.1获取和当前商品i的最近邻的k个商品--【要求这些最近邻商品都被对当前用户u评分过】；2.2 根据用户u对商品i的k个近邻商品的评分评分预测当前用户u对商品i的评分）
3. 重复步骤2计算出当前用户对所有物品的评分。
4. 重复步骤2、3计算所有用户对所有物品的评分
5. 对于每个用户，提取该用户对所有物品评分排序后，评分最高的N个商品作为推荐商品。

   注意（在提取商品列表的时候，可以考虑将用户已访问的商品去除）

2.2 ItemCF中基于k个近邻用户对商品评分预测的3种方法

2.2.1 直接对分数进行拟合

$$\hat{r}_{ui} = \frac{\sum_{v \in N_u^k(i)} sim(i,j) * r_{uj}}{\sum_{v \in N_i^k(i)} sim(i,j)}$$

其中，$\hat{r}_{ui}$表示预测用户u对商品i的评分，k表示用户u的k个近邻用户，u表示用户u，i表示商品i。

2.2.2 拟合分数到均值的偏差(基于normal) ——考虑用户偏好

$$r_{ui} = \bar{r}_i + \frac{\sum_{j \in N_i^k(i)} sim(i,j) * (r_{uj} - \bar{r}_j)}{\sum_{j \in N_i^k(i)} sim(i,j)}$$

其中，$\bar{r}_u$表示用户u对所有商品的平均评分，$\bar{r}_v$表示近邻用户v对所有商品的平均评分。

2.2.3 对均值做基线转换

进行基线转换，使用baseline的值来代替均值即可。因为均值体现的是当前用户在所有物品中的评分均值，而baseline可以认为是当前用户在当前物品上可能的评分，通过计算相似用户实际评分和baseline可能评分之间的差值从而可以得到当前用户的预测评分(**预测评=baseline+可能的差值**)。

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

其中，b_{ui}表示baselineonly算法中的$b_{ui} = \mu + b_u + b_i$。

3.协同过滤的优缺点

3.1 协同过滤的优点

- 简单性：实现简单，而且在调整参数的过程中只有一个近邻数需要调整。
- 合理性：对于预测推荐提供了简洁并且直观的理由。
- 高效性：基于近邻方法的推荐效果特别高，因为可以先进行预处理，构建出相似度矩阵，在实际应用过程中，可以提供近似实时的推荐结果
- 稳定性：当相似度矩阵构建完成后，如果有用户对物品产生新的评分，那么影响的范围就很小。

3.2 协同过滤的缺点

用户物品评分矩阵基本上都是一个非常稀疏的矩阵，它存在两个缺陷：

- 覆盖有限：由于计算两个用户之间的相似性是基于他们对相同物品的评分，而且只有对相同物品进行评分的用户才能作为近邻，但是在实际应用中，有些用户有很少或者没有共同评分，但是他们可能具有相似的爱好，所以推荐算法的覆盖将会受到影响。
- 对稀疏数据的敏感：由于用户只会对一部分物品进行评分，所以评分矩阵的稀疏性是大多数推荐系统的共同问题。当数据是稀疏的时候，两个用户或者物品之间的相似性计算仅适用很少量有限的近邻。另外，相似性权重的计算也可能依赖小部分评分，从而有可能导致推荐偏差。这也是一个比较重要的问题：冷启动问题。

4.ItemCF和UserCF的区别

	UserCF	ItemCF
性能	适用于用户较少的场合，如果用户过多，计算用户的相似度矩阵代价比较高。	适用于物品数明显小于用户数的场合，如果物品数过多，计算物品之间的相似度矩阵代价过大。
领域	时效性较强，用户个性化兴趣不明显的领域	长尾物品丰富，用户个性化需求强烈的领域
实时性	用户有新行为，不一定会对推荐结果产生影响	用户有新行为的时候，一定会导致推荐结果实时变化
冷启动	在新用户对很少物品产生行为的时候，不能立即对用户进行个性化推荐 新物品上线一段时间后，一旦有用户对该物品产生行为，那么就可以将新物品推荐给其他兴趣相似的用户	新用户只要对一个物品产生行为，就可以给他推荐和该物品相似的其它物品 必须等到更新物品相似度矩阵的时候才可以将新物品更新进去。
推荐理由	很难提供令用户信服的推荐解释	利用用户的历史行为做推荐解释，可以令用户比较信服