華中科技大学

课 程 实 验 报 告

课程名称： 大数据分析

专业班级： **CS1804 交换生**

学　　号： **X2020I1015**

X2020I1007

姓　　名： 李延波

刘日星

指导教师： 杨驰

报告日期： **2021/12/30**

计算机科学与技术学院

目录

实验五 推荐系统算法及其实现

1.1 实验目的

1、 了解推荐系统的多种推荐算法并理解其原理。
2、 实现 **User-User 的协同过滤算法**并对用户进行推荐。
3、 实现**基于内容的推荐算法**并对用户进行推荐。
4、 对两个算法进行电影预测评分对比
5、 在学有余力的情况下，**加入 minihash 算法**对效用矩阵**进行降维处理**

1.2 实验内容

给定 MovieLens 数据集，包含电影评分，电影标签等文件，其中电影评分文件分为训练集 train_set 和测试集 test_set 两部分

基础版必做一：**基于用户的协同过滤推荐算法**

对训练集中的评分数据构造用户-电影效用矩阵，使用 **pearson 相似度计算方法计算用户之间的相似度**，也即相似度矩阵。**对单个用户进行推荐时，找到与其最相似的 k 个用户，用这 k 个用户的评分情况对当前用户的所有未评分电影进行评分预测，选取评分最高的 n 个电影进行推荐。**

在测试集中包含 100 条用户-电影评分记录，用于计算推荐算法中预测评分的准确性，对测试集中的每个用户-电影需要**计算其预测评分，再和真实评分进行对比，误差计算使用 SSE 误差平方和**。

选做部分提示：此算法的进阶版采用 minihash 算法对效用矩阵进行降维处理，从而得到相似度矩阵，注意 minihash 采用 jarcard 方法计算相似度，需要对效用矩阵进行 01 处理，也即**将 0.5-2.5 的评分置为 0，3.0-5.0 的评分置为 1**。

基础版必做二：**基于内容的推荐算法**

将数据集 movies.csv 中的电影类别作为特征值，**计算这些特征值的 tf-idf 值**，得到关于电影与特征值的 n（电影个数）*m（特征值个数）的 **tf-idf 特征矩阵**。根据得到的 tf-idf 特征矩阵，**用余弦相似度的计算方法，得到电影之间的相似度矩阵**。

对某个用户-电影进行预测评分时，获取当前用户的已经完成的所有电影的打分，通过电影相似度矩阵获得已打分电影与当前预测电影的相似度，按照下列方式进行打分计算：

$$\text{score} = \frac{\sum_{i=1}^{n} score'(i) * sim(i)}{\sum_{i=1}^{n} sim(i)}$$

选取相似度大于零的值进行计算，如果已打分电影与当前预测用户-电影相似度大于零，加入计算集合，否则丢弃。（相似度为负数的，强制设置为 0，表示无相关）假设计算集合中一共有 n 个电影，score 为我们预测的计算结果，score'(i)为计算集合中第 i 个电影的分数，sim(i)为第 i 个电影与当前用户-电影的相似度。如果 n 为零，则 score 为该用户所有已打分电影的平均值。

要求能够对指定的 userID 用户进行电影推荐，推荐电影为预测评分排名前 k 的电影。userID 与 k 值可以根据需求做更改。

推荐算法准确值的判断：**对给出的测试集中对应的用户-电影进行预测评分，输出每一条预测评分，并与真实评分进行对比，误差计算使用 SSE 误差平方和**。

选做部分提示：进阶版采用 minihash 算法对特征矩阵进行降维处理，从而得到相似度矩阵，注意 minihash 采用 jarcard 方法计算相似度，特征矩阵应为 01 矩阵。因此进阶版的特征矩阵选取采用方式为，**如果该电影存在某特征值，则特征值为 1，不存在则为 0，从而得到 01 特征矩阵**。

选做（进阶）部分：

本次大作业的进阶部分是在基础版本完成的基础上大家可以尝试做的部分。进阶部分的主要内容是使用**迷你哈希（MiniHash）算法对协同过滤算法和基于内容推荐算法的相似度计算进行降维**。同学可以把迷你哈希的模块作为一种近似度的计算方式。

协同过滤算法和基于内容推荐算法都会涉及到相似度的计算，迷你哈希算法在牺牲一定准确度的情况下对相似度进行计算，其能够有效的降低维数，尤其是对大规模稀疏 01 矩阵。同学们可以**使用哈希函数或者随机数映射来计算哈希签名**。哈希签名可以计算物品之间的相似度。

最终降维后的维数等于我们定义映射函数的数量，我们设置的映射函数越少，整体计算量就越少，但是准确率就越低。**大家可以分析不同映射函数数量下，最终结果的准确率有什么差别**。

对基于用户的协同过滤推荐算法和基于内容的推荐算法进行推荐效果对比和分析，选做的完成后再进行一次对比分析。

1.3 实验过程

1.3.1 编程思路

1.本次实验旨在两种推荐算法来完成实验。

第一种为基于用户的协同过滤推荐算法。

协同过滤是利用集体智慧的一个典型方法。要理解什么是协同过滤(Collaborative Filtering, 简称 CF)，首先想一个简单的问题，如果你现在想看个电影，但你不知道具体看哪部，你会怎么做？大部分的人会问问周围的朋友，看看最近有什么好看的电影推荐，而我们一般更倾向于从口味比较类似的朋友那里得到推荐。这就是协同过滤的核心思想。本次实验我们采用的是基于用户的协同过滤算法。

第一步：收集用户偏好（建立用户模型）用训练集中的评分数据构造用户-电影效用矩阵。

第二步：找到相似的用户或物品。对每个用户进行推荐时，分别调用函数计算出该用户和其他用户的相似度，取相似度高的前 K 个用户。计算公式为：

$$sim(x,y)=\frac{\sum_{s\in S_{xy}}(r_{xs}-\overline{r_x})(r_{ys}-\overline{r_y})}{\sqrt{\sum_{s\in S_{xy}}(r_{xs}-\overline{r_x})^2}\sqrt{\sum_{s\in S_{xy}}(r_{ys}-\overline{r_y})^2}}$$

第三步：基于用户的 CF（user CF）。基于用户对物品的偏好找到相邻邻居用户，然后将邻居用户喜欢的推荐给当前用户。计算上，就是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，找到 K 邻居后，根据邻居的相似度权重以及他们对物品的偏好，预测当前用户没有偏好的未涉及物品，计算得到一个排序的物品列表作为推荐。

prediction functions as in user-user model

$$r_{xi}=\frac{\sum_{j\in N(i;x)}s_{ij}\cdot r_{xj}}{\sum_{j\in N(i;x)}s_{ij}}$$

s_{ij}… similarity of items i and j
r_{xj}…rating of user u on item j
$N(i;x)$… set items rated by x similar to i

通过预测评分公式，取前 k 个与该用户相似度最高的用户，对于这 k 个用户中与该用户相似度大于 0 的用户套用该公式，小于 0 的舍弃。

选取前 n 个电影进行推荐时，这些电影都是与其相似的 k 个用户看过的，但是该用户还未看过。

误差计算：对测试集中的 100 条用户-电影记录预测评分$\hat{y}_i$，与真实评分y_i一起带入下列公式计算。

$$SSE=\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

第二种为基于内容的推荐算法

基于内容的协同过滤算法与基于用户的协同过滤算法很像，将商品和用户互换。通过计算不同用户对不同物品的评分获得物品间的关系。基于物品间的关系对用户进行相似物品的推荐。这里的评分代表用户对商品的态度和偏好。简单来说就是如果用户 A 同时购买了商品 1 和商品 2，那么说明商品 1 和商品 2 的相关度较高。当用户 B 也购买了商品 1 时，可以推断他也有购买商品 2 的需求。
再了解了相关算法后，首先对数据集 movies.csv 中的电影类别进行分词，获取 m 个特征值。先生成电影与类别的矩阵。利用该矩阵生成 tf-idf 矩阵。

公式：$$tf_{ij} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

即：$$TF_w = \frac{\text{在某一类中词条}w\text{出现的次数}}{\text{该类中所有的词条数目}}$$

此处 tf 取特征值在该电影类别中的出现频率，也就是该电影对应一行中相加的和的倒数，idf 为总电影数 n 与包含某特征值的电影数的商，再取以 10 为底的对数，也就是对应矩阵[,类别]这一列的和除以总行数，再取以 10 为底的对数就是 idf 值。

对每个电影，调用余弦相似度函数根据 tf-idf 矩阵计算出该电影于其他电影的相似度，生成电影相似度矩阵。

- **Cosine similarity measure**
  - $\text{sim}(\boldsymbol{x}, \boldsymbol{y}) = \cos(\boldsymbol{r}_x, \boldsymbol{r}_y) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$

对测试集中的每条用户-电影记录，根据下列公式预测评分：

$$\text{score} = \frac{\sum_{i=1}^{n} score'(i) * sim(i)}{\sum_{i=1}^{n} sim(i)}$$

其中，sim(i)为第 i 个电影与当前用户-电影的相似度，score’(i)为计算集合中第 i 个电影的分数。计算集合不一定为 K，因为对于相似度为负数的电影不纳入计算。当 n 为 0 时，score 为该用户所有已打分电影的平均值。

为用户推荐电影时，利用该公式通过用户已经打分的电影来依次计算电影相似度矩阵中每一个电影的预测打分(除去用户已看过的电影)，对其进行排序，选出其中前 k 个电影进行推荐。

1.3.2 遇到的问题及解决方式

1. 基于用户的协同过滤算法

（1）计算用户之间相似度的时候，矩阵的计算量比较大，在用上述公式计算的时候，程序的开销有些大，需要很长的时间才可以运行出来，后来调用了

python 中的库函数，这个问题得到了解决。运行就比较快了。

（2）对于用户 i，假如与其相近的 k 个用户都未看过电影 m，但是测试集里要求用户 i 对电影 m 进行评分，这就会造成无法评分。这与数据稀疏性有关系，不同用户之间重叠性较低，导致算法无法找到一个用户的邻居，即偏好相似的用户。因此不得不将 K 变的比较大。

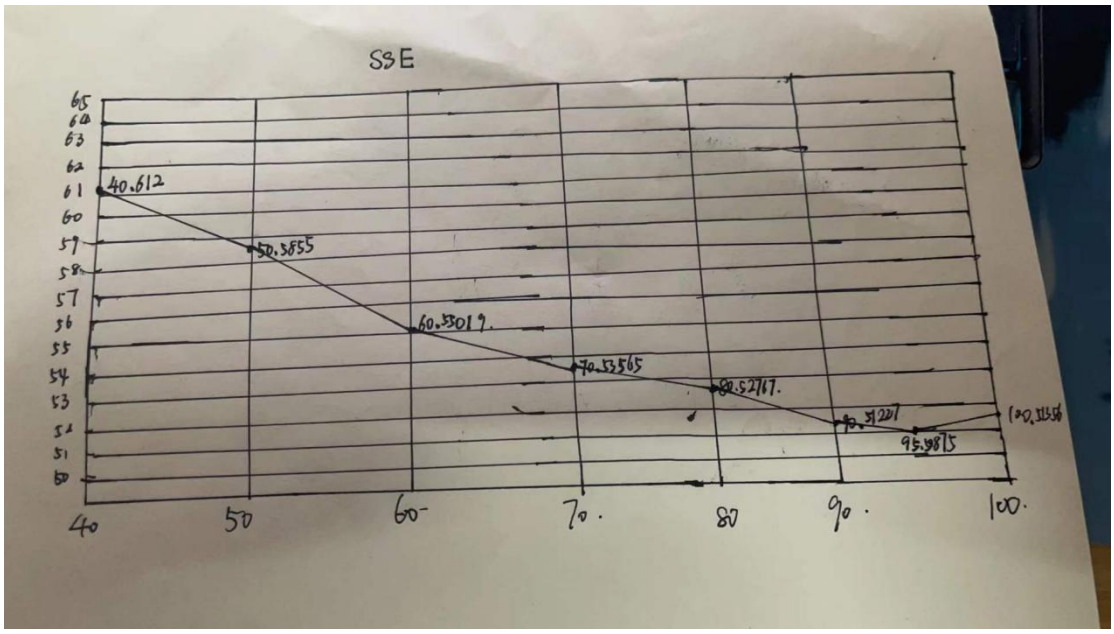
2. 基于内容的推荐算法

（1）余弦相似度矩阵的计算时间过长，导致程序运行是占用了很多 CPU，降低了运行效率。因此我查了一些资料，调用了库函数 cosine_similarity()计算余弦相似度矩阵。这与基于内容的推荐算法的缺点也有关。物品多的时候，计算物品相似度矩阵代价很大。

（2）在得到一系列数据后，发现不知道怎么给电影排名，后来我利用用户已经观看的电影所打出的分数以及给出的预测分数公式，对所有电影进行预测打分，选出排名前 k 个预测打分高的电影，输出电影名字进行推荐。

1.3.3 实验测试与结果分析

1. 基于用户的协同过滤算法



改变 K 值得到不同的 SSE 值，如图所示, K 值为 95 时，SSE 最小为 50.87。

对于用户 2，根据最相似前 50 个用户进行推荐，为该用户推荐出的 5 个电影如下图所示，包括电影 id、推荐指数、电影名称、以及电影派系（种类）。

```
请输入用户id: 2
请输入与用户id'2'最相似的前k位用户: 50
请输入为这些用户推荐的电影数: 5
 2          1       3.8781158455825793

 电影id          推荐指数          电影名称                                  电影流派
   255     5.299926217412691    Jerky Boys, The (1995)                    ['Comedy']
   534     5.238941533152815    Shadowlands (1993)                        ['Drama', 'Romance']
   162     5.203691828829079    Crumb (1994)                              ['Documentary']
   778     5.097183342039127    Trainspotting (1996)                      ['Comedy', 'Crime', 'Drama']
   219     5.090290381125227    Cure, The (1995)                          ['Drama']
```

对用户 2 的推荐结果

2. 基于内容的推荐算法

测试集预测打分结果：

```
['Adventure', 'Animation', 'Children', 'Comedy', 'Fantasy', 'Romance', 'Drama', 'Action', 'Crime', 'Thriller', 'Horror'
预测分值: {547: [(1, 3.5), (6, 2.5)], 564: [(1, 4.0), (2, 4.0)], 624: [(1, 5.0), (2, 3.0)], 15: [(1, 2.0), (2, 2.0)], 73:
实际分值: {547: [(1, 3.252042375634131), (6, 3.385064560575199)], 564: [(1, 3.408251103150588), (2, 3.276935909638767)],
SSE误差平方和: 67.06801578815222
```

结果分析：由图可知预测打分结果中有的分数预测和实际分数接近，有的预测分数与实际分数相差一定值，最终 SSE 误差平方和计算为 67.068。

测试为用户 ID 为 3 的推荐前 10 部电影：

```
请输入用户id: 3
请输入为该用户推荐的电影数: 10

 电影名称                                                    推荐指数
 Prisoner of the Mountains (Kavkazsky plennik) (1996)        4.151463
 Pork Chop Hill (1959)                                       4.151463
 Run Silent Run Deep (1958)                                  4.151463
 Cross of Iron (1977)                                        4.151463
 Duel at Diablo (1966)                                       4.151463
 Murphy's War (1971)                                         4.151463
 Richard III (1995)                                          4.115199
 Misérables, Les (1995)                                      4.115199
 Before the Rain (Pred dozhdot) (1994)                       4.115199
 Walking Dead, The (1995)                                    4.115199
```

1.4 实验总结

李延波：

本次实验是大数据分析的最后一个实验，也是相对来说比较重要的一部分，推荐系统。本次实验是基于用户的协同过滤推荐系统和基于内容的协同过滤的推荐系统。两种推荐系统各有各自的优势。通过本次实验让我对推荐系统有了一些了解。尤其是基于内容的协同过滤推荐系统。利用余弦相似度矩阵运行，耗费了很长的时间，后来调用库函数，则比较快速的完成了实验。而且基于内容的协同

过滤推荐系统也很好的弥补了基于用户的一些不足之处。因为物品直接的相似性相对比较固定，所以可以预先在线下计算好不同物品之间的相似度，把结果存在表中，当推荐时进行查表，计算用户可能的打分值。这样就解决了数据稀疏性和算法扩展性的问题。另外，对于测试集计算 SSE 是发现了该算法的一些问题，然我对该算法有了更深入的理解。本次实验让我收获颇丰。

刘日星：

这次实验对比于前几次的实验工程量要大一些，因此在和朋友一起完成代码时，首先要一起确定一些思路，然后通过确定好的思路进行编写。我主要是负责基于用户协同过滤算法的代码编写。同时在写的时候也出现了几点问题。在用户对电影做评价环节。由于数据的稀疏性，导致很多电影没有评分，这最开始让我很头疼。后来通过商量决定，将 K 值变大。另外在 SSE 上我也发现了一些算法上的问题。这次试验让我对推荐系统有了进一步的了解。也让我对于基于用户和基于内容的两个不同的算法进行了比较之后，有了更多收获。这次实验让我明白，推荐系统取自于生活，也用之于生活。