

（深圳）

实验报告

开课学期： 2020年秋季学期

课程名称： web信息处理

实验名称： 实验二

实验性质： 课内实验

学生专业： 计算机科学与技术

学生学号： SZ170110333

学生姓名： 段吕欣

评阅教师：

报告成绩：

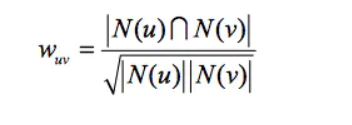
实验与创新实践教育中心印制

2018年9月

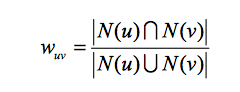
1. 基于协同过滤算法的推荐系统
2. 基于用户的协同过滤（User\_CF.py）
3. 算法原理

基于用户的协同过滤首先计算相似用户，然后再根据相似用户的喜好推荐物品。因此问题在于计算用户相似矩阵，当用户数量很多时，效率很低。计算用户相似度的方法有很多，如余弦相似度，Jaccard相似度，Pearson相关系数等。余弦相似度适用于评分数据，Jaccard适用于隐式反馈数据，Pearson相关系数适用于连续数据,正态分布,线性关系。因为用户相似度矩阵是评分数据，因此使用余弦相似度公式进行计算（也可以采用Pearson相关系数）。

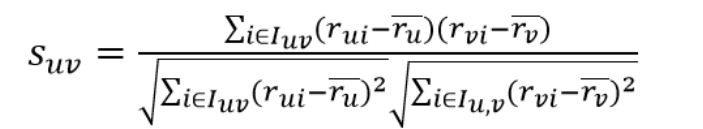
余弦相似度：



Jaccard公式：



Person相关系数：



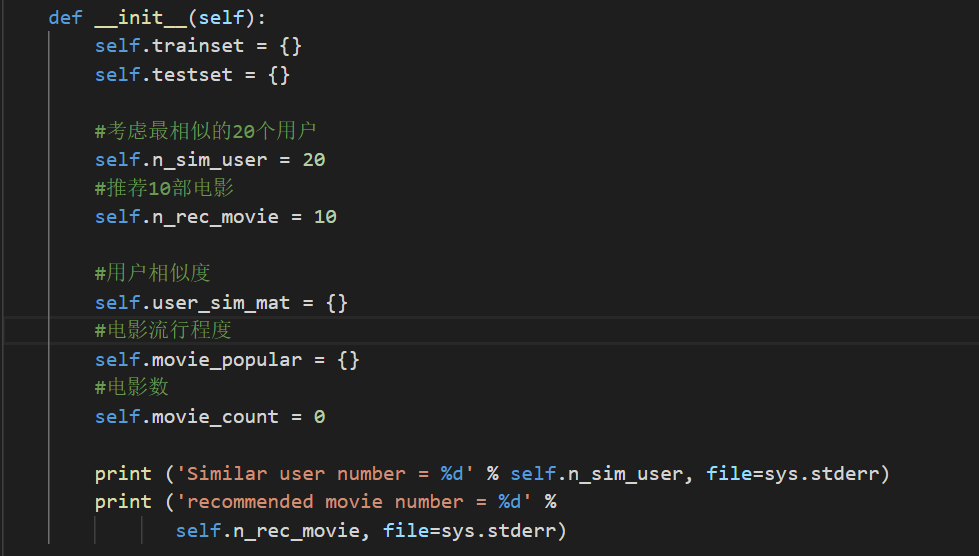
通过余弦相似度公式计算得到用户相似度矩阵后，就可以结合与用户相似的用户喜欢的物品给用户推荐物品了。可采用以下公式来预测用户u对物品i的评分（法一）：



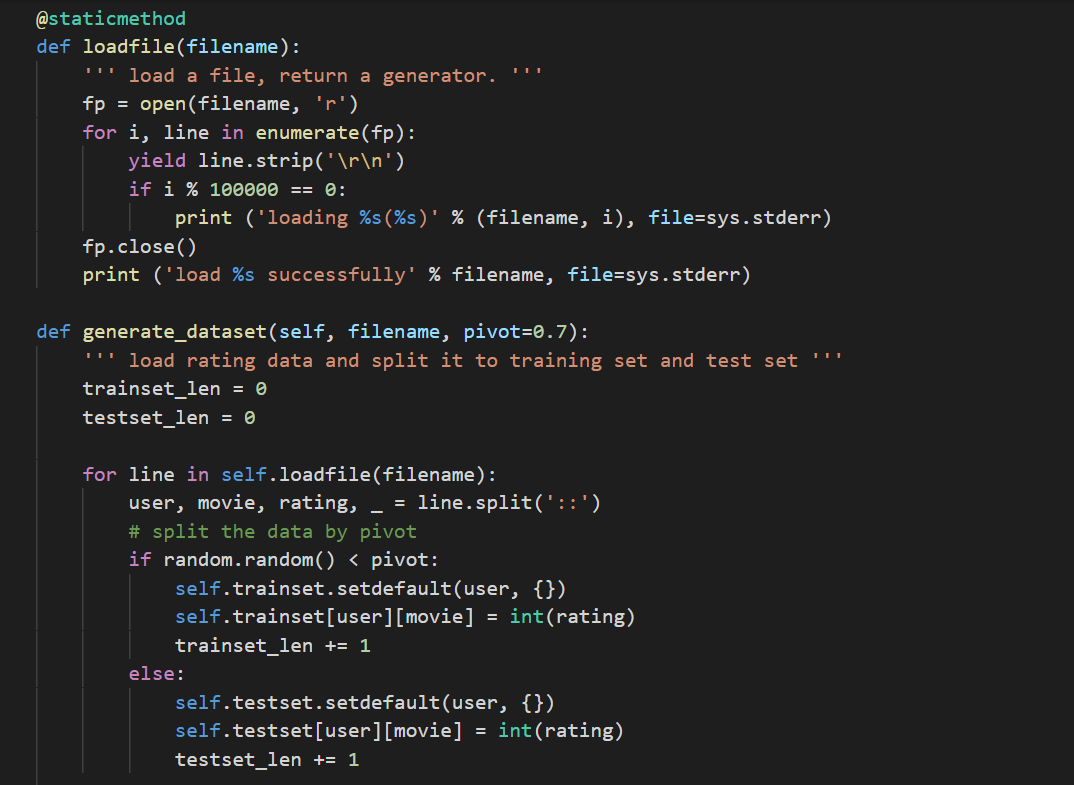
也可以不预测评分，直接进行推荐，找出与该用户最相似的K个用户，对他们使用过的物品进行计分（按照用户之间的相似度加分），得分最高的N个物品作为推荐物品（本次实验中采用的是这种方法（法二））

1. 实现步骤
2. 设定初始参数

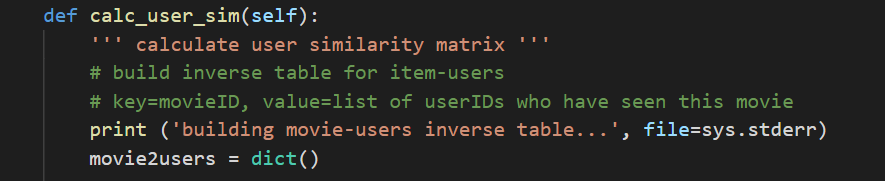
选定考虑的相似用户为20，推荐电影数为10。



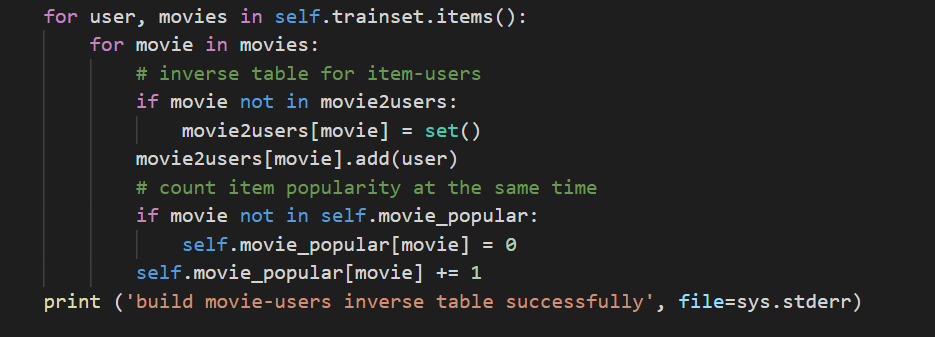
② 划分训练集和测试集，70%为训练集，30%为测试集。



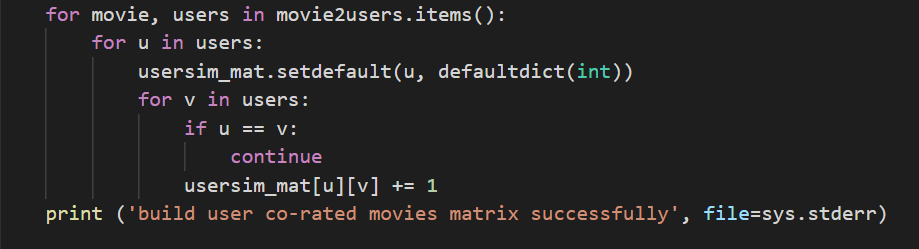
③ 计算用户相似性矩阵



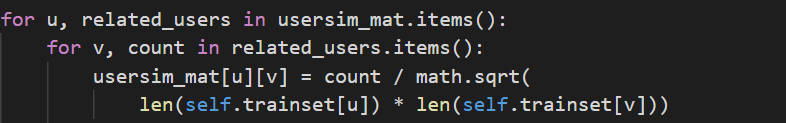
首先，计算出每部电影的观看用户，保存到movie2users中，即movie2users[movie]={user1,user2,...}；同时统计movie\_popular(用于计算popularity)



其次，对于训练集中的每一个用户，和另外某一个用户，如果他们看过同一部电影，则“相似度”（两个用户共同看过的电影数量，即余弦相似度公式中的分子）加一，暂时保存到用户相似度矩阵usersim\_mat中。



对于相似度矩阵usersim\_mat中的“相似度”，根据余弦相似度公式还需要将原来的值（即u用户和v用户都看过的电影数量）/，得到最终的用户相似性矩阵。(usersim\_mat[user]形如{u:{user1:3,user2:2}})

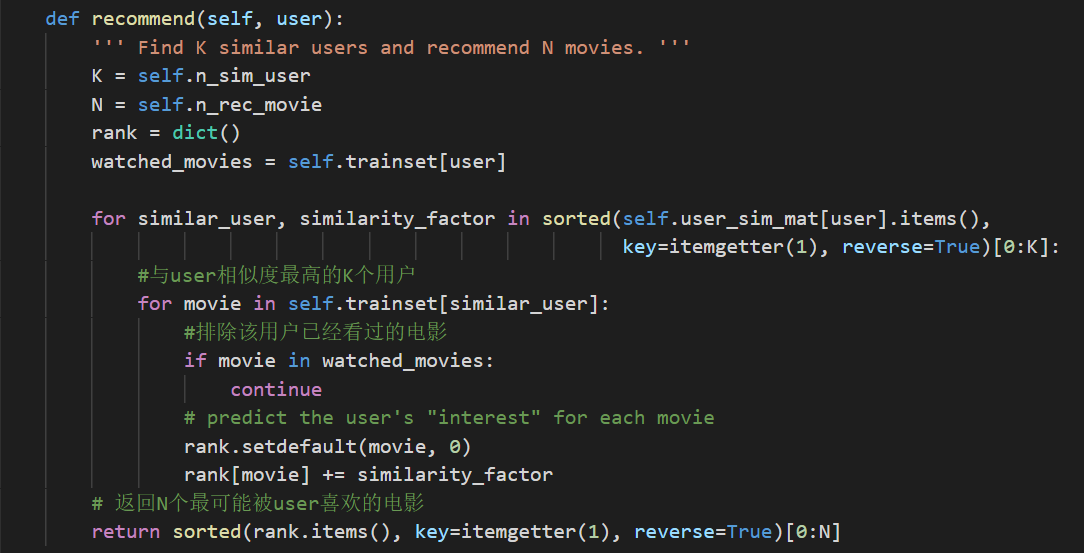


④ 对用户推荐电影

首先，根据用户相似度矩阵，找到与该用户最相似的K个用户；

之后，对于这K个用户看过的每一部电影，放入集合rank中，统计这部电影可能被该用户喜欢的概率作为电影的“得分”（根据用户相似度，用户相似度越高，被该用户喜欢的概率越大，则电影“得分”越高）。根据最后得出的每部电影的得分，取前10部电影。

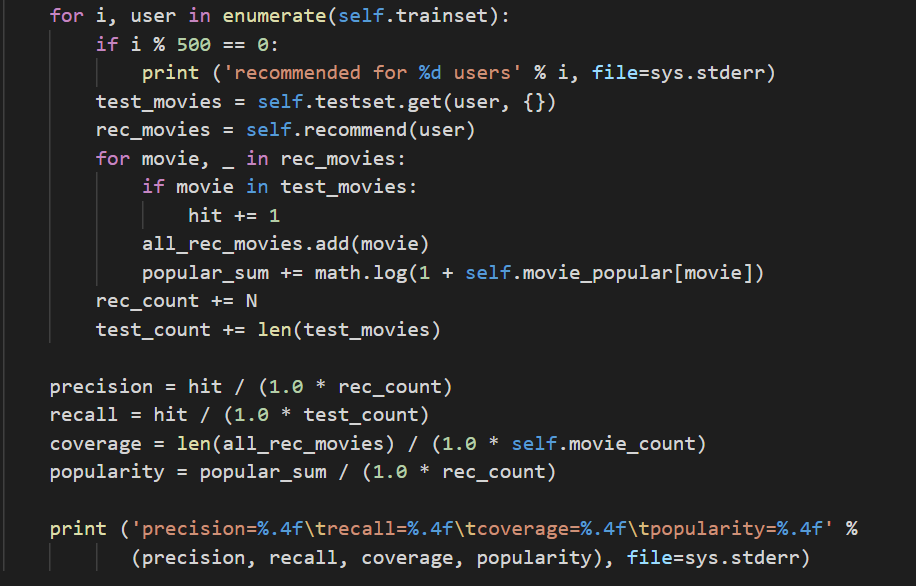
如果采用方法一，则user\_sim\_mat[user]形如{u:[(v1,0.5,1),(v2,1,3),(v3,0.3,4)]}，则评分为

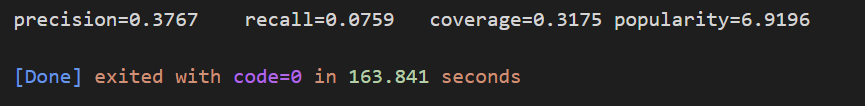


1. 模型评估

计算精确率，召回率，覆盖率，新颖性（推荐物品的平均流行度）。

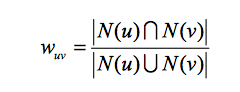
,,，



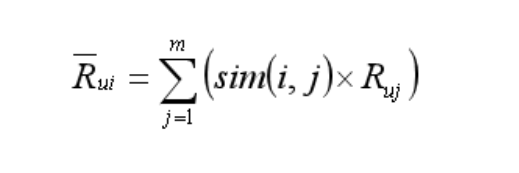


1. 基于物品的协同过滤（Item\_CF.py）
2. 算法原理

给用户推荐用户喜欢的物品相似的物品。问题在于“相似的物品”是如何度量的，即计算物品相似度矩阵。衡量公式有余弦相似度，Jaccard相似度，Pearson相关系数。因为物品相似度矩阵属于隐式反馈数据，因此最好采用Jaccard公式来计算。也可以采用余弦相似度公式来计算。（本实验中采用余弦相似度）



通过相似度计算计算得到物品相似度矩阵后，就可以结合用户已经喜欢过的物品集合给用户推荐物品了，可以通过下面的公式计算用户u对待推荐物品i的评分：

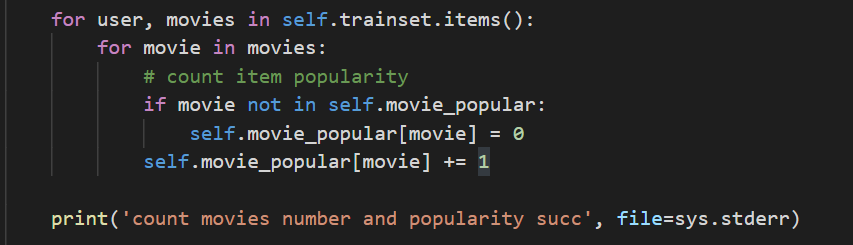


1. 实现步骤

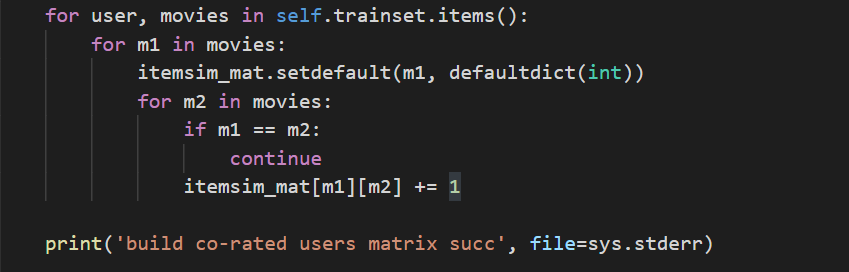
基本与基于用户的协同过滤相同，唯一不同的计算的是物品相似矩阵以及最后是根据电影向用户推荐。

①对于物品相似矩阵的计算：

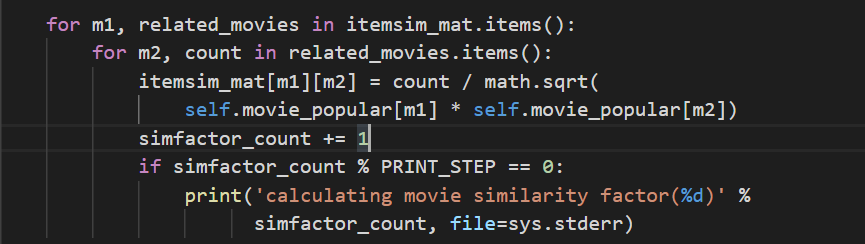
首先，需要计算电影流行度。对于训练集中每一个用户打过分（看过）的电影，统计数量。



其次，计算两部不同电影的“相似度”。遍历每一个用户看过的电影，如果这两部电影不是同一部电影，则“相似度”加一。



最后，利用余弦相似度公式，需要将原来的数值/sqrt（m1电影的流行度\*m2电影的流行度）



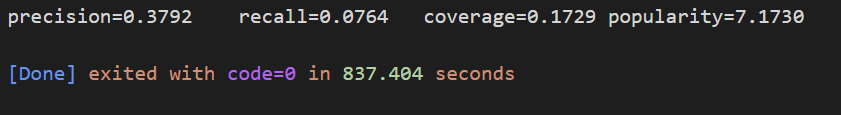
②根据电影向用户推荐

首先，对于该用户看过的每一部电影，计算与其物品相似度最高的K部电影，对于这K部电影，统计得分，得分是根据这K部电影与原电影的物品相似度\*原用户对原电影的打分得到的。最后取排名前N的电影推荐给用户。



1. 模型评估

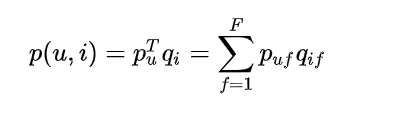
方法同基于用户的协同过滤。



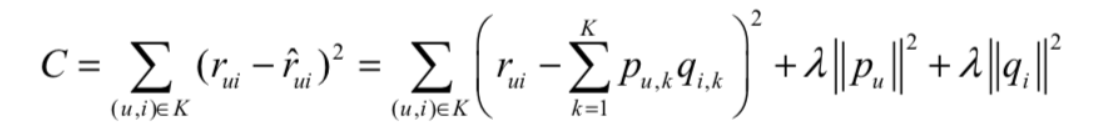
1. 基于LFM的推荐系统
2. 算法原理：

LFM（later factor model）是一种基于矩阵分解的召回算法，输入UI点展矩阵，输出是P和Q两个矩阵。

建模公式（用户u对于商品i的喜爱程度等于，用户u对于类别k的喜爱程度 乘以 商品i在类别k的比重）：

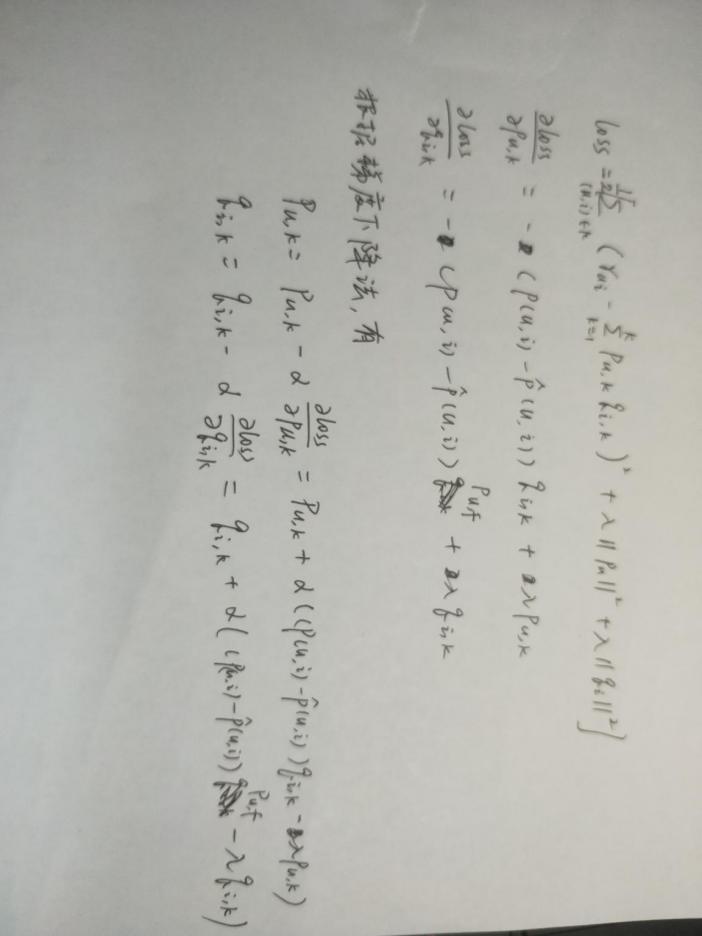


损失函数（K==F）：



为了方便计算，损失函数乘以1/2（如下）

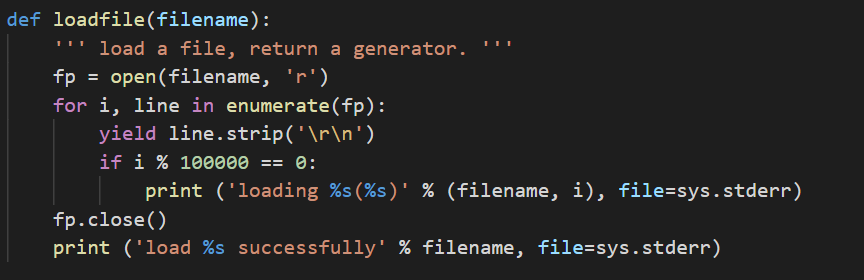
之后利用随机梯度下降法来最小化损失函数。



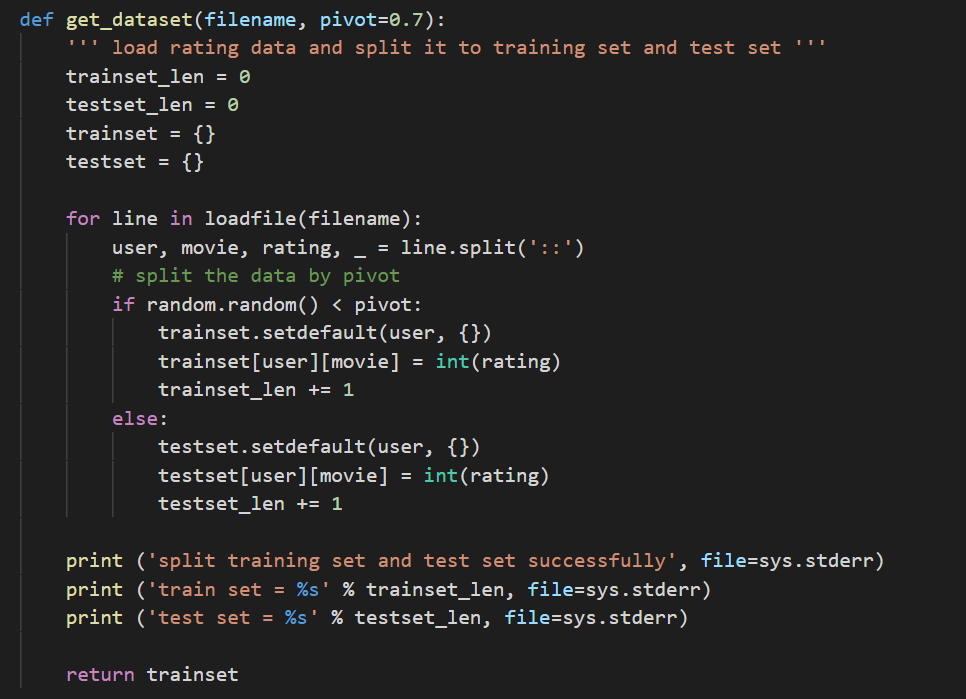
同时，每一步改变alpha的值。

1. 实现步骤：
2. 数据预处理

从文件中导入数据集：

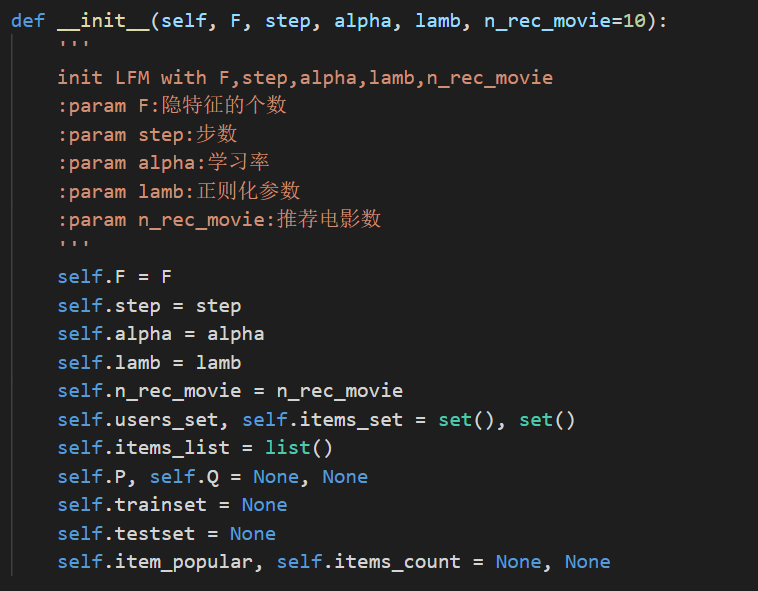


划分数据集，70%作为训练集，30%作为测试集：

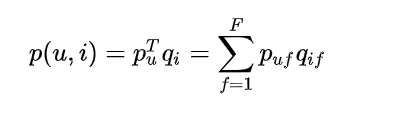


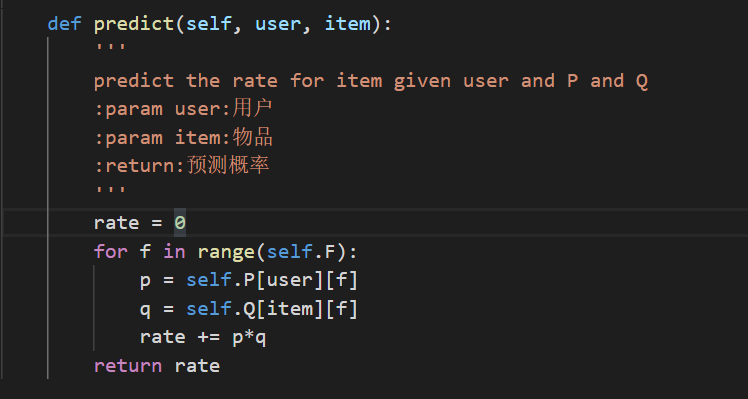
1. 初始化相关参数

设置隐参数的个数F为10，梯度下降的步数为20，学习速率alpha为0.1,正则化参数lamada为0.01，推荐的电影数为10

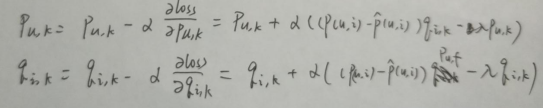
 

1. 预测函数



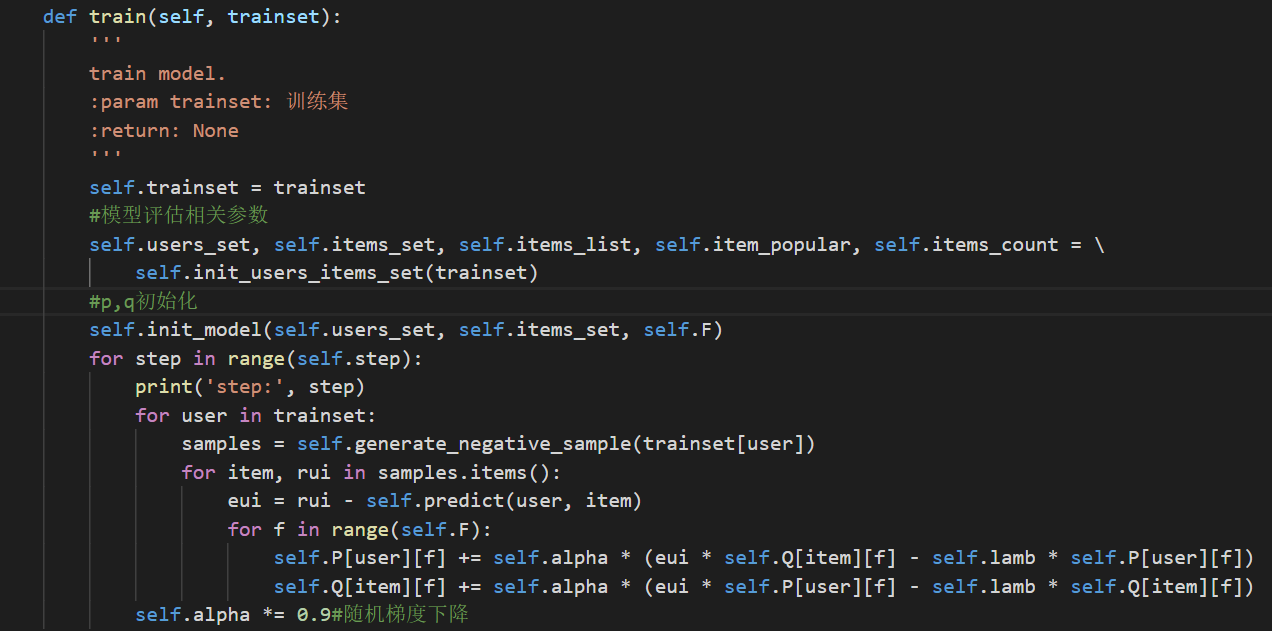


1. 训练得到p,q矩阵（随机梯度下降法）

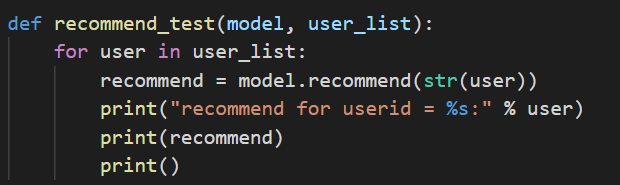


先计算之后模型评估需要用到的一些参数；然后初始化p，q矩阵，数值为随机值；

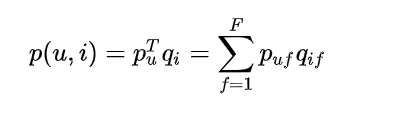
之后利用随机梯度下降法计算得到p,q矩阵

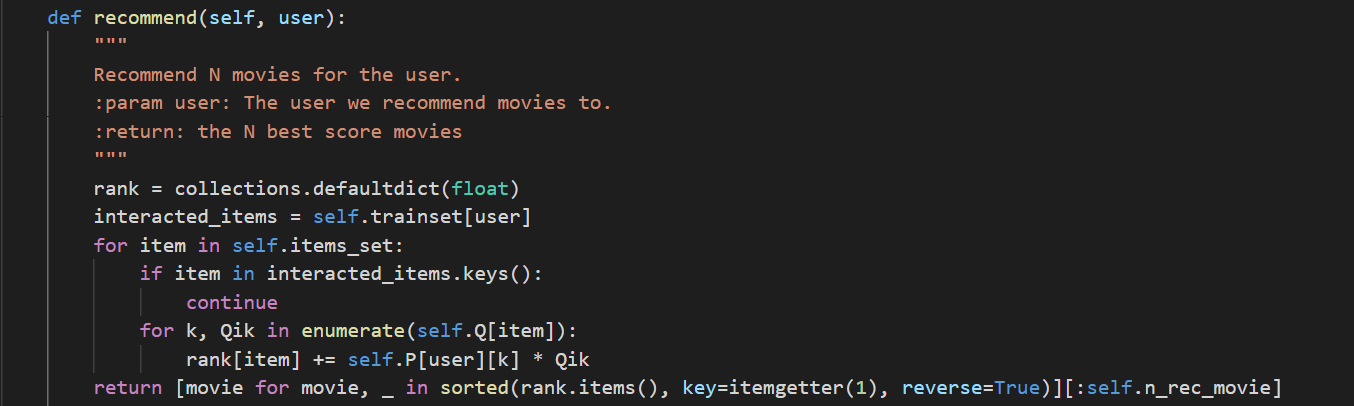


1. 执行推荐以及推荐结果：
2. 对user\_list中的每一个用户，为他们推荐电影，并打印结果：

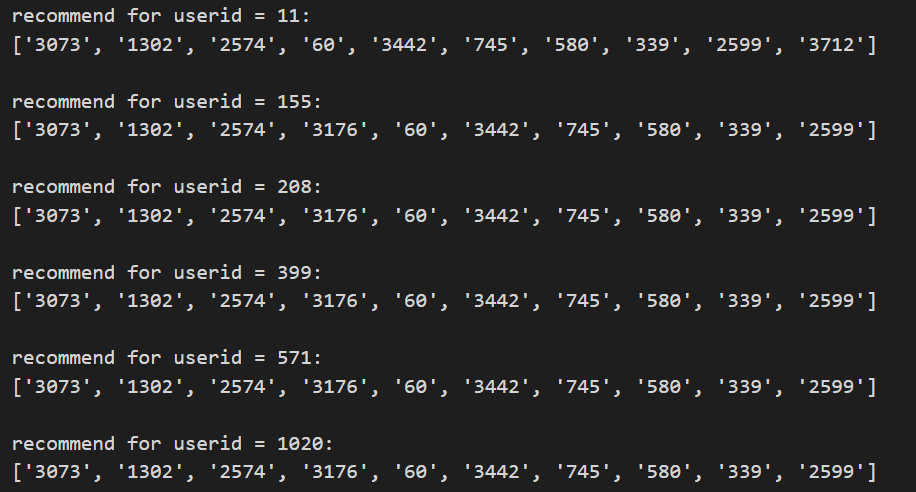


1. 根据以下函数编写recommend函数：



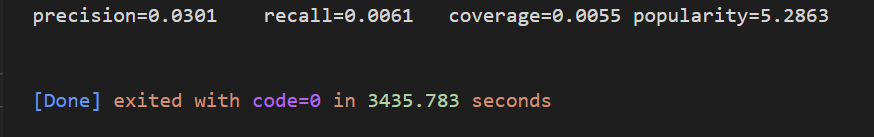


推荐结果：



1. 模型评估：

评估方法同协同过滤中的评估。

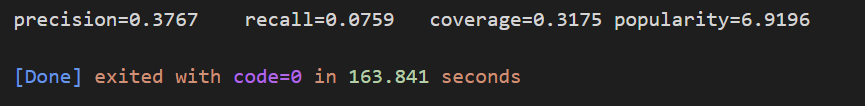


1. 调参优化

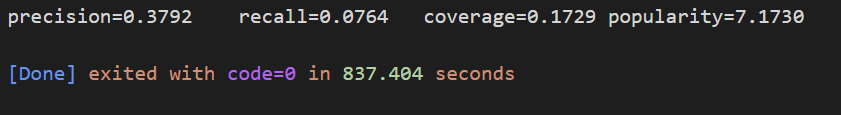
LFM模型的参数主要有隐参数的个数F，学习速率alpha,正则化参数lamada,负样本 /正样本的比例ratio;其中ratio对LFM的性能影响最大（本次实验中选用0.1）

1. 评估结果比较

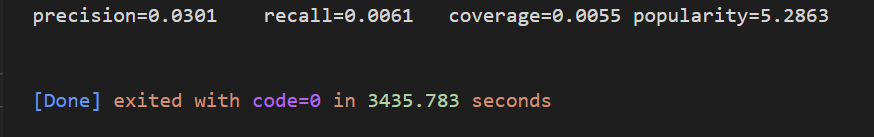
User-cf



Item-cf



LFM（ratio为0.1）



LFM（ratio为0.2）



从上面的评估结果可以看出：

时间复杂度方面LFM远高于user-cf和Item-cf;

准确率方面user-cf约等于item-cf,在未进行调参的情况下LFM的准确率较差。

1. 协同过滤和LFM的比较
2. 首先比较两种不同的协同过滤（item\_cf和user\_cf)



优缺点：

1. Item-cf

优点：以项目为基础的协同过滤不用考虑使用者间的差别，所以有些时候精度比较差。但是却不需要使用者的历史资料，或是进行使用者识别。对于项目来讲，它们之间的相似性要稳定很多，因此可以离线完成工作量最大的相似性计算步骤，从而降低了线上计算量，提高推荐效率，尤其是在使用者多于项目的情形下尤为显著。

缺点：但其仍有许多问题需要解决，最典型的有稀疏问题(Sparsity)和冷启动问题 (Cold-start)，开始时效果较差。此外还有新使用者问题和算法健壮性等问题。

1. User-cf

优点：在数据集完善，内容丰富情况下，准确率较高，而且能够避开项目内容上的挖掘进行准确推荐，能够对项目关联性，用户偏好进行隐式透明的挖掘。

缺点：随着使用者数量的增多，计算的时间就会变长；新用户问题；冷启动问题；数据稀疏性问题会导致效率与伸缩性上均不足

1. 其次比较协同过滤与LFM

|  |  |  |
| --- | --- | --- |
|  | 协同过滤 | LFM |
| 时间复杂度 | 较小 | 较大 |
| 空间复杂度 | 较大 | 较小 |
| 准确率 | 较高 | 与参数有较大的相关性 |
| 应用场景 | 在线推荐系统 | / |

L

优缺点：

1. 协同过滤

优点：它不需要对物品或者用户进行严格的建模，而且不要求对物品特征的描述是机器可理解的，所以这种方法也是领域无关的。并且这种方法计算出来的推荐是开放的，可以公用他人的经验，很好地支持用户发现潜在的兴趣偏好

缺点：方法的核心是基于历史数据，所以对新物品和新用户都有“冷启动”问题；

推荐的效果依赖于用户历史偏好数据的多少和准确性；在大部分实现中，用户历史偏好是用稀疏矩阵进行存储的，而稀疏矩阵上的计算有些明显的问题，包括可能少部分人的错误偏好会对推荐的准确度有很大的影响等等。对于一些特殊品味的用户不能给予很好的推荐。

1. LFM

优点： LFM具有比较好的理论基础，它是一种学习方法，通过优化一个设定的指标建立最优的模型；LFM的空间复杂度较小，占用的内存相较于协同过滤要小很多。

缺点：一般来说，LFM的时间复杂度较高，因为它需要进行多次迭代；

1. 心得和体会

协同过滤和LFM各有优缺点。对于协同过滤来说，因为并没有学习的过程，所以不需要进行调参优化；但对于LFM来说，参数的设置对其是极为重要的，因此后续还需要学习如何进行调参，如利用交叉验证调整lamada参数的值等等。