

# 协同过滤与矩阵分解算法实验报告

## 1 项目环境配置

- 操作系统: windows 10
- 编辑软件: pycharm
- 协同过滤算法依赖包: numpy、pandas、sklearn
- 矩阵分解算法依赖包: numpy、pandas、sklearn、pytorch 1.0+

## 2 项目使用的数据集

项目中用到了  $ml-1m$  数据集中评分文件 `ratings.dat`。该文件包含 1000209 评分记录。实际使用对数据集进行了以下处理:

- 1) 去除时间戳列, 只保留 `user_id, item_id, rating` 列的数据。
- 2) 对数据集按照 8: 2 划分得到训练集 `train_df`, 测试集 `test_df`, 需要注意的是划分时是通过每个用户的评分记录进行 8: 2 划分, 确保训练集和测试集中每一位用户都有评分记录。如图1所示, 训练集与测试集中分别包含 800193 和 200016 条评分数据。

```
[8] print(train_df.head())
    print(train_df.shape)

user_id  item_id  rating
0         1      745      3
1         1     1193      5
2         1      150      5
3         1      608      4
4         1     3186      4
(800193, 3)
```

```
[9] print(test_df.head())
    print(test_df.shape)

user_id  item_id  rating
0         1      914      3
1         1     2355      5
2         1     2804      5
3         1      919      4
4         1      527      5
(200016, 3)
```

图 1: 训练集与测试集

## 3 项目包含的模型

### 3.1 用户协同过滤模型

User CF 用户  $u$  与用户  $v$  的相似度:

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

User CF 用户  $u$  对物品  $i$  的评分预测:

$$p(u, i) = \sum_{v \in S(u, K) \cap N(i)} w_{uv} r_{vi}$$

- $N(i)$ : 是喜欢物品  $i$  的用户的集合
- $S(u, K)$ : 是与用户  $u$  最为相似的  $K$  个用户
- $r_{v,i}$ : 用户  $v$  对物品  $i$  的兴趣 (可看作评分, 若是隐反馈数据, 有过行为则为 1)
- $w_{u,v}$ : 用户  $u$  与用户  $v$  的相似度

### 3.2 物品协同过滤模型

Item CF 物品  $i$  与物品  $j$  的相似度:

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

Item CF 用户  $u$  对物品  $j$  的评分预测:

$$p_{u,j} = \sum_{i \in N(u) \cap S(j, K)} w_{ji} r_{ui}$$

- $N(u)$  是用户喜欢物品的集合
- $S(j, K)$  与物品  $j$  最相似的物品集合
- $r_{u,i}$  用户  $u$  对物品  $i$  的兴趣 (可看作评分, 隐反馈数据, 有过行为则为 1)
- $w_{j,i}$ : 物品  $i$  与物品  $j$  的相似度

协同过滤算法相关公式出自项亮的《推荐系统实战》。

### 3.3 矩阵分解模型

物品评分预测模型;

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

- $\mu$ : global average(所有物品的均分)
- $b_i$ : item bias
- $b_u$ : user bias
- $q_i^T p_u$ : user-item interaction

优化函数:

$$\min_{p,q,b} \sum_{(u,i) \in \kappa} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2$$

本实验报告的矩阵分解模型是比较早期的模型 [1], 详情见参考文献。

## 4 项目文件描述

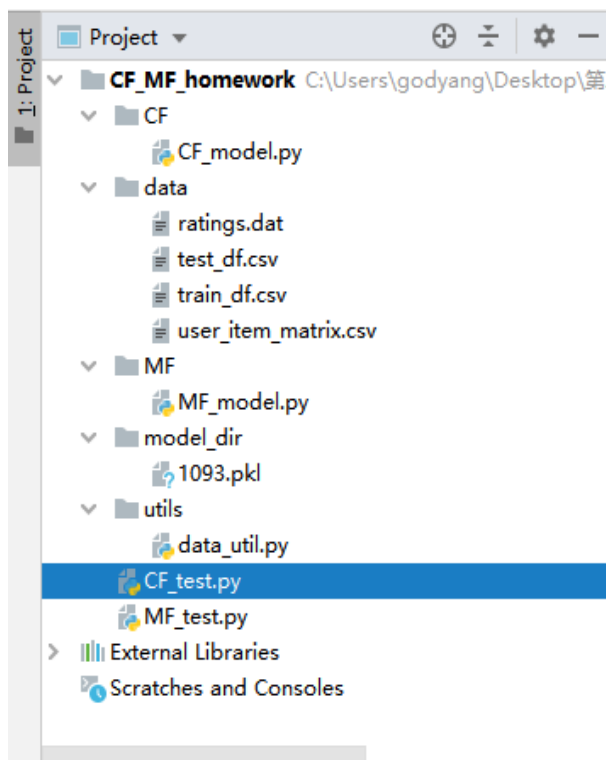


图 2: 项目文件结构

- *CF\_model.py*: 包含物品与用户协同过滤的相似度计算、评分预测函数
- *MF\_model.py*: 带有偏置的矩阵分解模型定义

- *data\_util.py*: 包含数据类型的转换工具、数据集的划分函数、用户评分矩阵的建立函数、用户、物品字典建立函数
- *CF\_test.py*: 用户协同过滤算法与物品协同过滤算法的测试文件
- *data\_util.py*: 矩阵分解算法的测试文件
- *data*: 原始评分文件、测试集、训练集、用户物品评分矩阵
- *model\_dir*: 训练好的 MF 模型参数

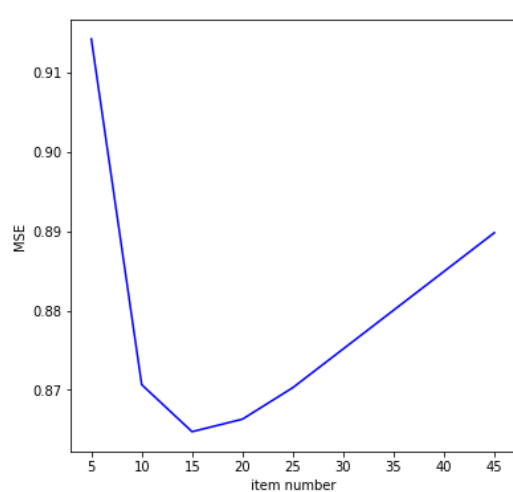
## 5 运行结果

### 5.1 协同过滤算法实验结果

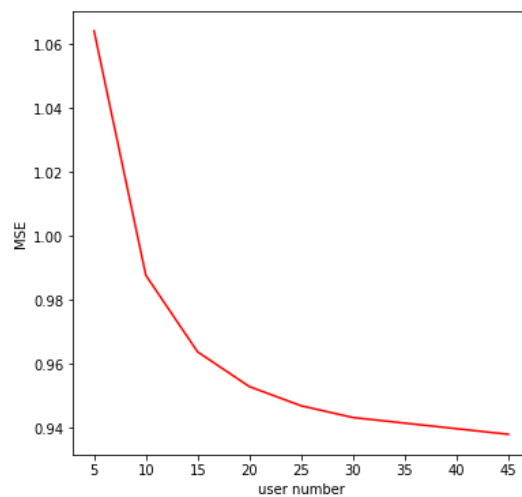
表 1: 不同相似的物品/用户数  $K$  下的 MSE 值

K	Use_CF	Item_CF
5	1.0640	0.9142
10	0.9877	0.8706
15	0.9637	0.8647
20	0.9529	0.8663
25	0.9469	0.8702
30	0.9432	0.8751
45	0.9380	0.8897

用户协同过滤算法与物品协同过滤算法在测试集上的 MSE 具体数值如表1和图4所示。其中  $K$  是指使用  $K$  个最为相似的用户/物品进行最后的评分预测。MSE 值随  $K$  的变化如图3, 总的来说, 协同过滤算法的 MSE 值都偏高, 此外, 实验可以看出, 物品协同过滤算法要比用户协同算法在结果上有一定优势。



(a) ItemCF



(b) UserCF

图 3: 不同 K 值情况下的 MSE 值

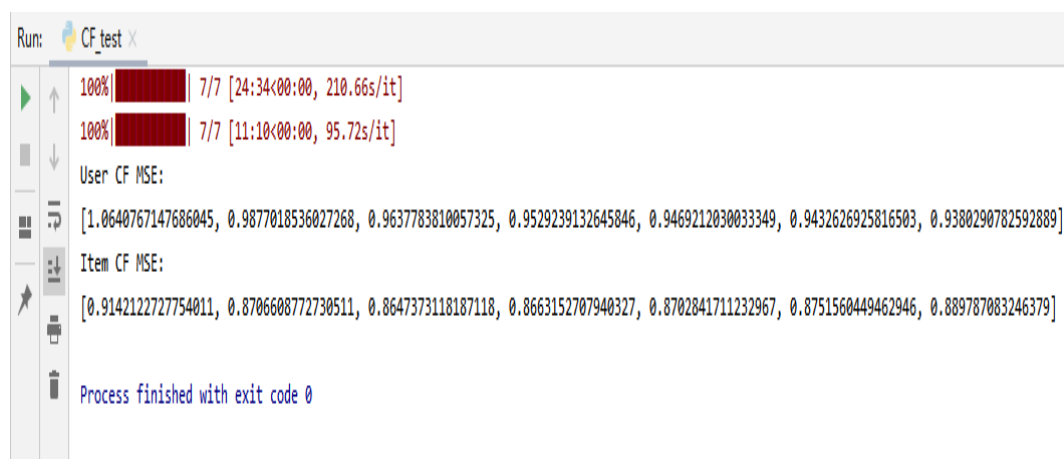
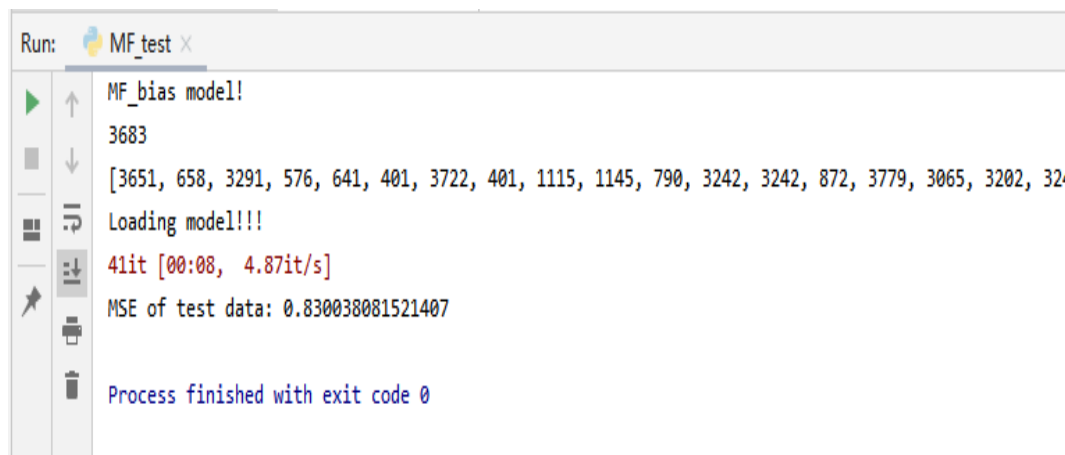


图 4: 协同过滤算法实验结果

## 5.2 矩阵分解算法实验结果



```
Run: MF_test x
MF_bias model!
3683
[3651, 658, 3291, 576, 641, 401, 3722, 401, 1115, 1145, 790, 3242, 3242, 872, 3779, 3065, 3202, 3242]
Loading model!!!
41it [00:08, 4.87it/s]
MSE of test data: 0.830038081521407
Process finished with exit code 0
```

图 5: 带有偏置的 MF 的实验结果

经过实验，如图5, 矩阵分解算法最终在测试集上的 MSE 值为 0.83 左右，该结果优于协同过滤算法。

## 6 总结

本次实验主要使用协同过滤算法与矩阵分解算法进行评分预测任务。从实验结果上看，针对实验数据集物品协同过滤算法效果比用户协同过滤好，矩阵分解算法效果比协同过滤算法要好。

就协同过滤算法而言，本次实验在相似度的计算上比较粗糙，如果要提升实验效果，这方面需要进行评估选取比较合适的相似度。另外一点就是在进行最终预测的时候，使用的最为相似的物品/用户数对实验结果也有一定影响。矩阵分解算法的种类比较多，这次实验选用的是早期的带有偏置的矩阵分解算法，实验结果实际上比目前的 **baseline** 要差一点。个人觉得模型的参数还是可以进一步优化的。

## 参考文献

- [1] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.