智能推荐系统: Assignment 1

10185102208@stu.ecnu.edu.cn

Edited on 2021 年 4 月 5 日

张伟 *2021 Spring*

华东师范大学
计算机科学与技术学院

李民选

10185102208

目录

1 算法简介

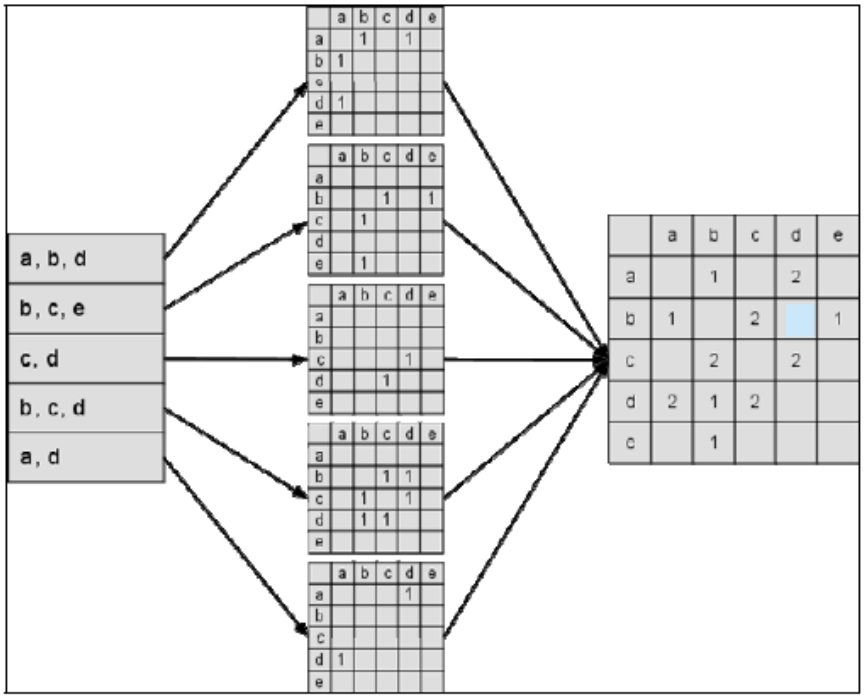
1.1 简介

在本次实验中，我用到的算法是基于 item 的协同过滤算法 (ICF)。基于物品的协同过滤算法是目前业界应用最多的算法。无论是 Amazon，还是 Netflix、Hulu、YouTube，其推荐算法的基础都是该算法。

基于物品的协同过滤算法给用户推荐那些和他们之前喜欢的物品相似物品。算法步骤主要分为两步：首先计算物品之间的相似度，然后根据物品的相似度和用户的历史行为给用户生成一个推荐列表，列表内容顺序按照相似度从大到小排列。

1.2 实现步骤

首先要计算物品相似度。和 UserCF 算法类似，ItemCF 算法计算物品相似度时也可以首先建立 User-Item 倒排表 (即对每个用户建立一个包含他喜欢的物品的列表)，然后对于每个用户，将他物品列表中的物品两两在共现矩阵中加 1。



上图是一个根据上面算法描述计算物品相似度的简单例子。图中最左边是输入的用户行为记录，每一行代表一个用户感兴趣的物品集合。然后对于每个物品集合，我们将里面的物品两两加一，得到一个矩阵。最终将这些矩阵相加得到上面的 C 矩阵。其中 C[i][j] 记录了同时喜欢物品 i 和物品 j 的用户数。最后将 C 矩阵归一化就可以得到物品之间的余弦相似度矩阵 W。

在得到物品之间的相似度之后，ItemCF 通过如下公式计算用户 u 对一个物品 j 的兴趣：

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

1.3 如何运用到本次实验

根据实验目的，本次实验最终是要对 test.csv 文件中的每一条记录产生一个预测评分，那么这个预测评分就要根据以上步骤中的推荐物品列表来计算、输出。

由于在此我使用的是 ICF 算法，那就不再将 user 字段作为关键字段。对于每一条记录，首先对 user 推荐 N=10 个物品列表，然后判断该条记录中的 item 是否正存在于这个推荐列表中，如果存在，那么将预测评分设置为列表中所有物品评分的平均值；如果不存在于该推荐物品列表中，那么根据该商品被其他用户评分的均值来给予一个初始评分值，然后这个初始评分值再与物品与推荐物品中最相似物品的相似度相乘，便能得到最后的预测评分。

2 核心代码

2.1 划分数据集

将 train.csv 文件划分出 20% 的数据作为测试集，用来计算 RMSE。

```
def readData_test(self):
        #将train.csv划分 80%为训练集，20%为测试集
        #train.csv文件一共大约有8000条数据字段，这里选取1600条为test集合
        self.rmse_user = []
        self.rmse_item = []
        self.rmse_star = []
        with open('train.csv','r') as csvfile:
                reader = csv.reader(csvfile)
                for i,rows in enumerate(reader):
                        if i==0: continue
                        row = rows
                        user = row[0] #读取当前数据的第一列
                        item = row[1] #读取当前数据的第二列
                        star = row[3] #读取当前数据的第四列
                        if(i<=1600):
                                self.rmse_user.append(user)
                                self.rmse_item.append(item)
                                self.rmse_star.append(int(float(star)))
```

2.2 计算 i-i 相似度矩阵

```
def ItemSimilarity(self):
        """
        计算物品之间的相似度
        """
        C = {} #items-items矩阵 行为次数的矩阵 共现矩阵
        N = {} #记录items被多少个不同用户购买
        #遍历训练数据，获取用户对有过行为的物品
```

```
        for user, items in self.train.items():
        #遍历该用户每件物品项
                for i in items.keys():
                #该物品被用户评分则计数加1
                        if i not in N.keys():
                                N.setdefault(i, 0)
                        N[i] += 1

                # 物品-物品共现矩阵数据加1
                        if i not in C.keys():
                                C.setdefault(i, {})
                for j in items.keys():
                        if i == j:
                                continue
                        if j not in C[i].keys():
                                C[i].setdefault(j, 0)
                        C[i][j] += 1
        #计算相似度矩阵， 计算物品-物品的相似度，余弦相似度
        self.W = {}
        for i, related_items in C.items():
                if i not in self.W.keys():
                        self.W.setdefault(i, {})
        for j, cij in related_items.items():
                self.W[i][j] = cij / (math.sqrt(N[i] * N[j]))
        return self.W
```

2.3 为 user 推荐物品

```
def Recommend(self, user, K=5, N=10):
        """
        给用户推荐物品，取相似度最大的 K 个物品，推荐排名靠前的 N 个物品
        """
        '''
                :param user: 用户(str)
                :param K: 相似度的前K个        W[item].items()
                :param N: 最后算出来的结果的前N个
                :return: 返回最后的前N个值 rank
        '''
        # 用户对物品的偏好值
        rank = {}
        # 用户产生过行为的物品项和评分
        action_item = self.train[user]
```

```
        for item, score in action_item.items():
        #遍历与item最相似的前K个物品，获得这些物品及分数
                for j, wj in sorted(self.W[item].items(), key=lambda x: x[1],
                    reverse=True)[0:K]:
                #若有该物品，跳过
                        if j in action_item.keys():
                                continue
                        if j not in rank.keys():
                                rank.setdefault(j, 0)
                        rank[j] += int(float(score)) * wj
        return sorted(rank.items(), key=lambda x: x[1], reverse=True)[0:N]
```

2.4 对 test 中数据进行预测评分

```
for k in range(len(L_user)):
        #遍历test.csv中的每一条记录
        ori_user = L_user[k]
        #首先对与用户推荐N=10个物品
        temp = cf.Recommend(ori_user)
        temp_item = []
        for t in temp: temp_item.append(t[0])
        item = L_item[k]
        #判断当前item是否在这个推荐物品的列表中
        if item in temp_item:
                score = 0.0
                num = 0
                for t in temp:
                        score += t[1]
                        num = num + 1
                score /= num

        #如果不在当前列表
        else:
        #如何设置初始评分 与相似度wj相乘？
        #一个想法是 计算当前物品产生所有评分的平均值 然后取整作为初始评分
                S = cf.train.items()
                score_final = 0.0
                counter = 0
                for user1,itemk in S:
                        S_temp = itemk.keys()
                        for i in S_temp:
                                if i==item:
```

```
                        score_final += float(itemk[i])
                        counter = counter + 1
            #设置当前初始评分，然后计算相似系数wj
            score = score_final / counter
            #遍历与item最相似的1个物品，获得这个物品及分数
            for j, wj in sorted(cf.W[item].items(), key=lambda x: x[1], reverse=
                True)[0:1]:
                    score = score * wj
            score_int = int(float(score))
            if score - score_int >= 0.5:
                    score_int = score_int + 1
            score_int += 1
            if score_int >5: score_int = 5
```

3 程序环境

3.1 package needed

1. csv
2. math

3.2 system environment

1. Windows 10 x64 Home Edition
2. Python 3.6.1 & IDLE

4 结果分析

4.1 RMSE 简介

RMSE，均方根误差，它是观测值与真值偏差的平方和观测次数 n 比值的平方根，在实际测量中，观测次数 n 总是有限的。真值只能用最可信赖 (最佳) 值来代替，方根误差对一组测量中的特大或特效误差反映非常敏感，所以，均方根误差能够很好地反映出测量的精密度。均方根误差，当对某一量进行甚多次测量时，取这一测量列真误差的均方根差称为标准偏差，以 σ 表示。σ 反映了测量数据偏离真实值的程度，σ 越小，表示测量精度越高。

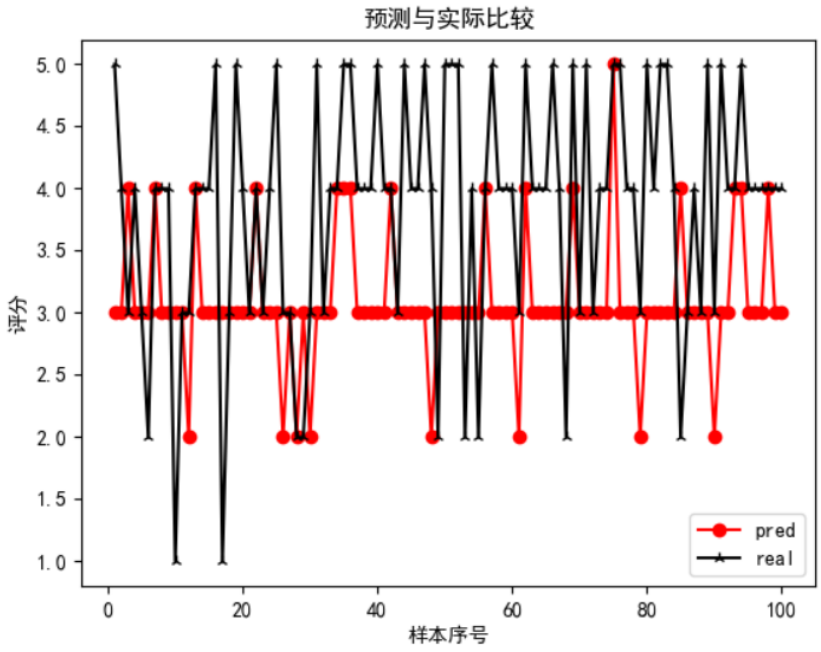
4.2 RMSE

```
>>>
========== RESTART: D:\华东师范大学\大三下\智能推荐系统\我的作业\第一次作业\test
_v2_rmse.py ==========
Program Launching!!!
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
Train dataset loading OK!!!
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
Test dataset loading OK!!!
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
Start to predict for test dataset!!!
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
RMSE: 1.294942083646987
>>>
```

4.3 预测评分与真实评分比较 (部分)

从 train.csv 中划分出 1600 条记录作为 test，这个数据量用 plot 做图库做出来就太过拥挤，因此这里选取了 100 条记录来作图，观测预测评分与真是评分的比较。

下图是 plot 绘制结果：



5 文件列表

5.1 predict.py

为 test.csv 中的每条记录给出一个预测评分。

5.2 rmse.py

计算该预测算法的 rmse(均方根误差)。

5.3 show_data.py

可视化预测数据与真实数据的比较。

5.4 prediction.csv

test.csv 的评分预测结果。

5.5 compare.csv

预测评分与真实评分的比较。