智能推荐系统第一次编程作业

——Memory-based CF for Rating Prediction

学号：10185102142

姓名：李泽浩

指导老师：张伟

项目名称：Memory-based CF for Rating Prediction

时间：2021年4月5日

目录

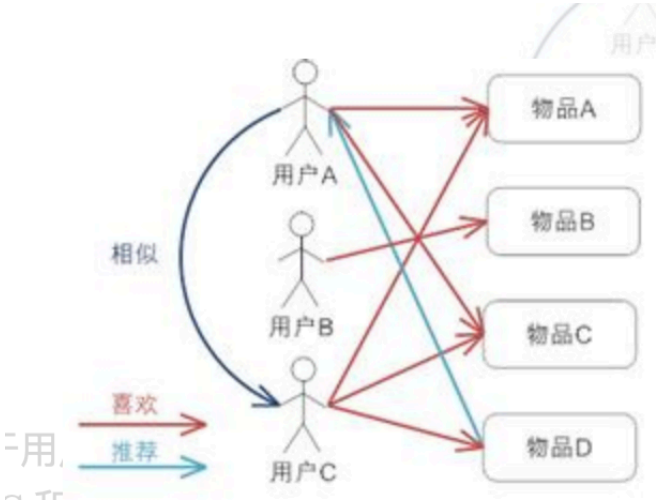
1、推荐算法简介

1.1基于协同过滤的推荐

基于协同过滤的推荐(Collaborative Filtering-based Recommendation)：它的原理就是根据用户对物品或者信息的偏好，发现物品或者内容本身的相关性，或者是发现用户的相关性，然后再基于这些关联性进行推荐，即利用集体的智慧进行推荐，比如你想看一部电影但不知道哪部，这时大部分人会问周围的朋友，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。基于协同过滤的推荐可以分为三个子类：基于用户的推荐（User-based Recommendation）、基于项目的推荐（Item-based Recommendation）和基于模型的推荐（Model-based Recommendation）。

1.2基于用户的协同过滤推荐(User-based Collaborative Filtering Recommendation)

基于用户的协同过滤推荐算法先使用统计技术寻找与目标用户有相同喜好的邻居，然后根据目标用户的邻居的喜好产生向目标用户的推荐。基本原理就是利用用户访问行为的相似性来互相推荐用户可能感兴趣的资源，如图所示：

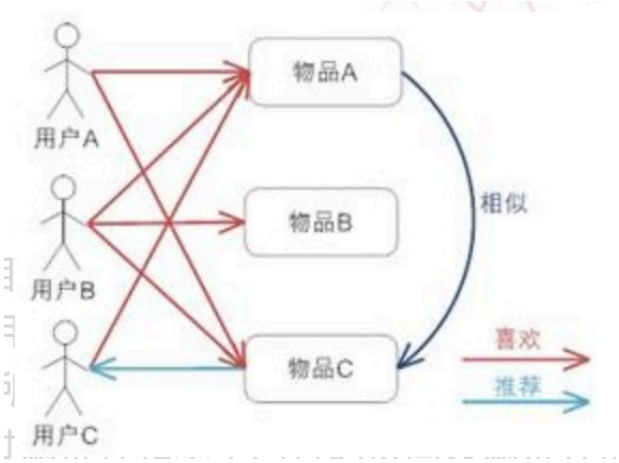


上图示意出基于用户的协同过滤推荐机制的基本原理，假设用户 A 喜欢物品 A、物品 C，用户 B 喜欢物品 B，用户 C 喜欢物品 A 、物品 C 和物品 D；从这些用户的历史喜好信息中，我们可以发现用户 A 和用户 C 的口味和偏好是比较类似的，同时用户 C 还喜欢物品 D，那么我们可以推断用户 A 可能也喜欢物品 D，因此可以将物品 D 推荐给用户 A。

基于用户的协同过滤推荐机制和基于人口统计学的推荐机制都是计算用户的相似度，并基于“邻居”用户群计算推荐，但它们所不同的是如何计算用户的相似度，基于人口统计学的机制只考虑用户本身的特征，而基于用户的协同过滤机制可是在用户的历史偏好的数据上计算用户的相似度，它的基本假设是，喜欢类似物品的用户可能有相同或者相似的口味和偏好。

1.3基于项目的协同过滤推荐(Item-based Collaborative Filtering Recommendation)

根据所有用户对物品或者信息的评价，发现物品和物品之间的相似度，然后根据用户的历史偏好信息将类似的物品推荐给该用户，如图所示：



上图表明基于项目的协同过滤推荐的基本原理，用户A喜欢物品A和物品C，用户B喜欢物品A、物品B和物品C，用户C喜欢物品A，从这些用户的历史喜好中可以认为物品A与物品C比较类似，喜欢物品A的都喜欢物品C，基于这个判断用户C可能也喜欢物品C，所以推荐系统将物品C推荐给用户C。

基于项目的协同过滤推荐和基于内容的协同过滤推荐都是基于物品相似度预预测推荐，只是相似度度量的方法不一样，前者是从用户历史的偏好推断，而后者是基于物品本身的属性特征信息。

2、基于用户的协同过滤算法UCF

2.1相似度计算及应用场景

（1）欧几里得距离

欧几里得度量（euclidean metric）（也称欧氏距离）是一个通常采用的距离定义，指在m维空间中两个点之间的真实距离，或者向量的自然长度（即该点到原点的距离）。在二维和三维空间中的欧氏距离就是两点之间的实际距离。

欧几里得距离是数据上的直观体现，看似简单，但在处理一些受主观影响很大的评分数据时，效果则不太明显；比如，U1对Item1,Item2 分别给出了2分，4分的评价;U2 则给出了4分，8分的评分。通过分数可以大概看出，两位用户褒Item2 ,贬Item1，也许是性格问题，U1 打分更保守点，评分偏低，U2 则更粗放一点，分值略高。在逻辑上，是可以给出两用户兴趣相似度很高的结论。如果此时用欧式距离来处理，得到的结果却不尽如人意。即评价者的评价相对于平均水平偏离很大的时候欧几里德距离不能很好的揭示出真实的相似度。

（2）皮尔逊相关系数

Pearson 相关系数是用协方差除以两个变量的标准差得到的，虽然协方差能反映两个随机变量的相关程度（协方差大于0的时候表示两者正相关，小于0的时候表示两者负相关），但其数值上受量纲的影响很大，不能简单地从协方差的数值大小给出变量相关程度的判断。为了消除这种量纲的影响，于是就有了相关系数的概念。

当两个变量的方差都不为零时，相关系数才有意义，相关系数的取值范围为[-1,1]。

适用范围：适用于A的评价普遍高于B的评价

（3）余弦相似度

余弦距离，也称为余弦相似度，是用向量空间中两个向量夹角的余弦值作为衡量两个个体间差异的大小的度量。余弦值越接近1，就表明夹角越接近0度，也就是两个向量越相似，这就叫"余弦相似性"。

（4）Jaccard相似系数

定义：广义Jaccard相似度，元素的取值可以是实数。又叫作谷本系数

应用场景：比较文本相似度，用于文本查重与去重；计算对象间距离，用于数据聚类等。

2.2 实际应用

UserCF在目前的实际应用中使 用并不多。其中最著名的使用者是Digg，它在2008年对推荐系统进行了新的尝试1。Digg使用推 荐系统的原因也是信息过载，它的研究人员经过统计发现，每天大概会有15 000篇新的文章，而 每个用户的精力是有限的，而且兴趣差别很大。因此Digg觉得应该通过推荐系统帮用户从这么多 篇文章中找到真正令他们感兴趣的内容，同时使每篇文章都有机会被展示给用户。

Digg的推荐系统设计思路如下。用户在Digg中主要通过“顶”和“踩”两种行为表达自己对文章的看法。当用户顶了一篇 文章，Digg就认为该用户对这篇文章有兴趣，而且愿意把这篇文章推荐给其他用户。然后，Digg 找到所有在该用户顶文章之前也顶了这一篇文章的其他用户，然后给他推荐那些人最近顶的其他 文章。从这里的简单描述可以看到，Digg使用的是UserCF算法的简化版本。

Digg在博客中公布了使用推荐系统后的效果，主要指标如下所示。
 用户反馈增加:用户“顶”和“踩”的行为增加了40%。
 平均每个用户将从34个具相似兴趣的好友那儿获得200条推荐结果。 用户和好友的交互活跃度增加了24%。
 用户评论增加了11%。

2.3 用户打分预测公式

在一个在线个性化推荐系统中，当一个用户A需要个性化推荐 时，可以先找到和他有相似兴趣的其他用户，然后把那些用户喜欢的、而用户A没有听说过的物品推荐给A。这种方法称为基于用户的协同过滤算法。 从上面的描述中可以看到，基于用户的协同过滤算法主要包括两个步骤。
(1) 找到和目标用户兴趣相似的用户集合。
(2) 找到这个集合中的用户喜欢的，且目标用户没有听说过的物品推荐给目标用户。 步骤(1)的关键就是计算两个用户的兴趣相似度。这里，协同过滤算法主要利用行为的相似度计算兴趣的相似度。给定用户u和用户v，令N(u)表示用户u曾经有过正反馈的物品集合，令N(v) 为用户v曾经有过正反馈的物品集合。

基于此用户a对商品p打分对预测公式为：

$$\boldsymbol{pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}}$$

3、代码详解

3.1导入必要的函数库

```
#导入必要的函数库
import numpy
import random
import pandas as pd
import math
from sklearn.model_selection import train_test_split
from sklearn import datasets
from collections import defaultdict
```

3.2数据预处理

```
#对train按照用户ID排序，便于查看
import pandas as pd
filename = "train.csv"
data = pd.read_csv(filename)
data = data.sort_values(by='user_id', ascending=False)
data.to_csv("train.csv", index=False)
```

```
#对test按照商品ID排序
import pandas as pd
filename = "test.csv"
data = pd.read_csv(filename)
data = data.sort_values(by='user_id', ascending=False)
data.to_csv("train.csv", index=False)
```

3.3读取训练集数据

```
#读取测试集数据
filename = "train.csv"
f = open(filename, "rt", encoding="utf-8")
header = f.readline()
header = header.strip().split(',')#列名
print(header)
```

3.4将每一行数据存入dataset空列表中

```
dataset = []#存储每一行用户ID、商品ID、打分时间、打分星级

for line in f:
    fields = line.strip().split(',')
    d = dict(zip(header, fields))
    d["stars"] = float(d["stars"])
    dataset.append(d)
```

3.5数据分析

```
UserPerItem = defaultdict(set)  #记录对某商品评价过的用户
ItemPerUser = defaultdict(set)  #记录某用户评价过的商品
#把训练集数据由dataset读入上述字典中
for d in dataset:
    user,item = d['user_id'], d['business_id']
    UserPerItem[item].add(user)#评价过某商品的用户列表，key值为business_id
    ItemPerUser[user].add(item)#某用户评价过的商品列表，key值为用户名user_id
```

3.6相似度函数

```
#余弦相似度
def cos(s1,s2):
    demon = 0.0
    number = len(s1.intersection(s2))#并集
    l1 = len(s1) #s1集合中元素个数
    l2 = len(s2) #s2集合中元素个数
    demon += math.sqrt(l1 * l2)
    if demon == 0:
        return 0
    return number / demon
```

3.7预测函数

```
reviewsPerUser = defaultdict(list)
reviewsPerItem = defaultdict(list)

for d in dataset:
    user,item = d['user_id'], d['business_id']
    reviewsPerItem[item].append(d)
    reviewsPerUser[user].append(d)

def prdictRating(user,item):
    ratings = []
    similarities = []
    for d in reviewsPerUser[user]:
        i2 = d['business_id']
        if i2 == item:continue
```

```
            ratings.append(d['stars'])
            similarities.append(cos(UserPerItem[item],UserPerItem[i2]))
        if(sum(similarities)>0):
            weightedRatings = [(x*y) for x,y in zip(ratings,similarities)]
            return sum(weightedRatings) / sum(similarities)
        else:
            return ratingMean
```

4、结果分析及准确率

4.1读取测试集数据预测并写回

```
#读取测试集数据
filename2 = "test.csv"
f2 = open(filename2, "rt", encoding="utf-8")
header2 = f2.readline()
header2 = header2.strip().split(',')#列名
print(header2)
header2.append('pre_stars')
print(header2)

predata = []

for line in f2:
    fields = line.strip().split(',')
    d = dict(zip(header2, fields))
    d["pre_stars"] = 0
    predata.append(d)

for d in predata:
    u, i = d['user_id'], d['business_id']
    s = round(prdictRating(u,i))
    d['pre_stars'] = s

df = pd.DataFrame(predata)
df.to_csv(filename2)
```

4.2对训练集进行预测并查看准确率

```
s = len(dataset)
count = 0
for d in dataset:
    user,item,star = d['user_id'], d['business_id'], d['stars']
    star = float(star)
    p = round(prdictRating(user,item))
    if p - star <= 0.5:
        count += 1
print(count/s)
```

结果如下：

```
s = len(dataset)
count = 0
for d in dataset:
    user,item,star = d['user_id'], d['business_id'], d['stars']
    star = float(star)
    p = round(prdictRating(user,item))
    if p - star <= 0.5:
        count += 1
print(count/s)
```

executed in 177ms, finished 22:40:10 2021-03-31

```
0.7294503277861826
```

4.3计算均方误差

```
#均方误差函数
def MSE(predictions,labels):
    differences = [(x-y)**2 for x,y in zip(predictions,labels)]
    return sum(differences) / len(differences)
# 实际打分平均值
alwaysPredictMean = [ratingMean for d in dataset]
#预测打分
cfPredicitions = [prdictRating(d['user_id'], d['business_id']) for d in
dataset]

labels = [d['stars'] for d in dataset]

MSE(alwaysPredictMean, labels)

MSE(cfPredicitions, labels)
```

结果如下：

```
MSE(alwaysPredictMean, labels)
```

executed in 5ms, finished 22:44:52 2021-03-31

```
1.0033436473005906
```

```
MSE(cfPredicitions, labels)
```

executed in 6ms, finished 22:44:52 2021-03-31

```
1.0792256987993392
```

4.4预测某商品时通过考虑商品热度进行衰减

```
def prdictRating(user,item):
    c = len(UserPerItem[item])#计算有多少个用户评价过该商品
    c = 1 / (1 + math.log(c,10))
    ratings = []
    similarities = []
    for d in reviewsPerUser[user]:
        i2 = d['business_id']
        if i2 == item:continue
        ratings.append(d['stars'])
        similarities.append(cos(UserPerItem[item],UserPerItem[i2]) * c)
    if(sum(similarities)>0):
        weightedRatings = [(x*y) for x,y in zip(ratings,similarities)]
        return sum(weightedRatings) / sum(similarities)
    else:
        return ratingMean
```

```
#导入新测试集数据并预测
#读取测试集数据
filename2 = "test2.csv"
f2 = open(filename2, "rt", encoding="utf-8")
header2 = f2.readline()
header2 = header2.strip().split(',')#列名
#print(header2)
header2.append('pre_stars')
#print(header2)

predata = []

for line in f2:
    fields = line.strip().split(',')
    d = dict(zip(header2, fields))
    d["pre_stars"] = 0
    predata.append(d)

for d in predata:
    u, i = d['user_id'], d['business_id']
    s = round(prdictRating(u,i))
    d['pre_stars'] = s

df = pd.DataFrame(predata)
df.to_csv(filename2)
```

4.5对比test.csv和test2.csv

```
filename_1 = "test.csv"
filename_2 = "test2.csv"

file1 = open(filename_1, "rt", encoding="utf-8")
file2 = open(filename_2, "rt", encoding="utf-8")

headers1 = file1.readline()
headers1 = headers1.strip().split(',')#列名
headers2 = file2.readline()
headers2 = headers2.strip().split(',')#列名

data1 = []
data2 = []

for line in file1:
    fields = line.strip().split(',')
    d = dict(zip(headers1, fields))
    data1.append(d)

for line in file2:
    fields = line.strip().split(',')
    d = dict(zip(headers2, fields))
    data2.append(d)

s = len(data1)#总数据量
count = 0#统计评分相等的个数
for i in range(len(data1)):
    s1 = float(data1[i]['pre_stars'])
    s2 = float(data2[i]['pre_stars'])
    if(s1 == s2):
        count += 1
print(count / s)
```

executed in 20ms, finished 11:15:39 2021-04-03

```
0.9989806320081549
```

5、提交文件列表

实验报告lab1-10185102142-李泽浩

源代码source_code.ipynb

预测结果后的test.csv与test2.csv