ACdreamer
梦之河上，一叶扁舟！





# 协同过滤算法

2015-03-30 00:13 15725人阅读 评论(11) 收藏 举报

分类：
机器学习（34）



今天要讲的主要内容是协同过滤，即Collaborative Filtering，简称**CF**。

## Contents



## 1. 协同过滤的简介

关于协同过滤的一个最经典的例子就是看电影，有时候不知道哪一部电影是我们喜欢的或者评分比较高的，那
么通常的做法就是问问周围的朋友，看看最近有什么好的电影推荐。在i
都习惯于问跟自己口味差不
多的朋友，这就是协同过滤的核心思想。

协同过滤是在海量数据中挖掘出小部分与你品味类似的用户，在协同过滤中，这些用户成为邻居，然后根据他
们喜欢的东西组织成一个排序的目录推荐给你。所以就有如下两个核心问题

（1）如何确定一个用户是否与你有相似的品味？
（2）如何将邻居们的喜好组织成一个排序目录？

协同过滤算法的出现标志着推荐系统的产生，协同过滤算法包括基于用户和基于物品的协同过滤算法。

## 2. 协同过滤的核心

要实现协同过滤，需要进行如下几个步骤

（1）收集用户偏好
（2）找到相似的用户或者物品
（3）计算并推荐

收集用户偏好

从用户的行为和偏好中发现规律，并基于此进行推荐，所以如何收集用户的偏好信息成为系统推荐效果最基础

的决定因素。用户有很多种方式向系统提供自己的偏好信息，比如：评分，投票，转发，保存书签，购买，点

击流，页面停留时间等等。

以上的用户行为都是通用的，在实际推荐引擎设计中可以自己多添加一些特定的用户行为，并用它们表示用户

对物品的喜好程度。通常情况下，在一个推荐系统中，用户行为都会多于一种，那么如何组合这些不同的用户

行为呢 ？基本上有如下两种方式

（1）将不同的行为分组

一般可以分为查看和购买，然后基于不同的用户行为，计算不同用户或者物品的相似度。类似与当当网或者

亚马逊给出的"购买了该书的人还购买了"，"查看了该书的人还查看了"等等。

（2）不同行为产生的用户喜好对它们进行加权

对不同行为产生的用户喜好进行加权，然后求出用户对物品的总体喜好。

好了，当我们收集好用户的行为数据后，还要对数据进行预处理，最核心的工作就是减噪和归一化。

减噪： 因为用户数据在使用过程中可能存在大量噪音和误操作，所以需要过滤掉这些噪音。

归一化：不同行为数据的取值相差可能很好，例如用户的查看数据肯定比购买数据大得多。通过归一化，才能

使数据更加准确。

通过上述步骤的处理，就得到了一张二维表，其中一维是用户列表，另一维是商品列表，值是用户对商品的喜

好。还是以电影推荐为例，如下表

| | Titanic | Avatar |
|---|---|---|
| User1 | 0.5 | 0.3 |
| User2 | 0.4 | 0.3 |
| User3 | 0.4 | ? |

找到相似的用户或物品

对用户的行为分析得到用户的喜好后，可以根据用户的喜好计算相似用户和物品，然后可以基于相似用户或物

品进行推荐。这就是协同过滤中的两个分支了，基于用户的和基于物品的协同过滤。

关于相似度的计算有很多种方法，比如常用的余弦夹角，欧几里德距离度量，皮尔逊相关系数等等。而如果采

用欧几里德度量，那么可以用如下公式来表示相似度

$$sim(x,y) = \frac{1}{1+d(x,y)}$$

在计算用户之间的相似度时，是将一个用户对所有物品的偏好作为一个向量，而在计算物品之间的相似度时，

是将所有用户对某个物品的偏好作为一个向量。求出相似度后，接下来可以求相似邻居了。

计算并推荐

在上面，我们求出了相邻用户和相邻物品，接下来就应该进行推荐了。当然从这一步开始，分为两方面，分别

是基于用户的协同过滤和基于物品的协同过滤。我会分别介绍它们的原理

（1）基于用户的协同过滤算法

在上面求相似邻居的时候，通常是求出TOP K邻居，然后根据邻居的相似度权重以及它们对物品的偏好，

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进行推荐。

（2）基于物品的协同过滤算法

跟上述的基于用户的协同过滤算法类似，但它从物品本身，而不是用户角度。比如喜欢物品A的用户都喜

欢物品C，那么可以知道物品A与物品C的相似度很高，而用户C喜欢物品A，那么可以推断出用户C也可能

喜欢物品C。如下图

| 用户/物品 | 物品A | 物品B | 物品C |
|---|---|---|---|
| 用户A | √ | | √ |
| 用户B | √ | √ | √ |
| 用户C | √ | | 推荐 |

上面的相似度权重有时候需要加入惩罚因子，举个例子，在日常生活中，我们每个人购买卫生纸的的频率比

较高，但是不能说明这些用户的兴趣点相似，但是如果它们都买了照相机，那么就可以大致推出它们都是摄

影爱好者。所以像卫生纸这样的物品在计算时，相似度权重需要加上惩罚因子或者干脆直接去掉这类数据。

适用场景

对于一个在线网站，用户的数量往往超过物品的数量，同时物品数据相对稳定，因此计算物品的相似度不但

计算量小，同时不必频繁更新。但是这种情况只适用于电子商务类型的网站，像新闻类，博客等这类网站的

系统推荐，情况往往是相反的，物品数量是海量的，而且频繁更新。所以从算法复杂度角度来说，两种算法

各有优势。关于协同过滤的文章，可以参考这

里：**http://www.tuicool.com/articles/6vqyYfR**

### 3. 协同过滤的实现

上面已经介绍了协同过滤的核心思想，现在就来实战一下吧！ 采用数据集如下

链接：**http://grouplens.org/datasets/movielens/**

这个数据集是很多用户对各种电影的评分。接下来先采用**Python**实现基于用户的协同过滤算法。

首先，我们需要以表格形式读取数据，需要用到Texttable第三方包。安装包如下链接

链接：**https://pypi.python.org/pypi/texttable/**

大致用法如下

```
# -*-coding=utf-8 -*-

import sys
import math
from texttable import Texttable

if __name__ == '__main__':
    tb = Texttable()
    tb.set_deco(Texttable.HEADER | Texttable.VLINES
                | Texttable.HLINES | Texttable.BORDER
    tb.add_rows([
        ["Name", "Age", "Nickname"],
        ["Julia", 20, "Lua"],
        ["ACdreamer", 22, "Jack"]
    ])
    print tb.draw()
```

更多方法的使用需要参考Textdtable的源文件texttable.py。接下来可以实现协同过滤算法了。

代码：

```python
# -*-coding=utf-8 -*-

import sys
import math
from texttable import Texttable

#计算余弦距离
def getCosDist(user1, user2):
    sum_x = 0.0
    sum_y = 0.0
    sum_xy = 0.0
```

```
    for key1 in user1:
        for key2 in user2:
            if key1[0] == key2[0]:
                sum_x += key1[1] * key1[1]
                sum_y += key2[1] * key2[1]
                sum_xy += key1[1] * key2[1]
    if sum_xy == 0.0:
        return 0
    demo = math.sqrt(sum_x * sum_y)
    return sum_xy / demo

#读取文件，读取以行为单位，每一行是列表里的一个元素
def readFile(filename):
    contents = []
    f = open(filename, "r")
    contents = f.readlines()
    f.close()
    return contents

#数据格式化为二维数组
def getRatingInfo(ratings):
    rates = []
    for line in ratings:
        rate = line.split("\t")
        rates.append([int(rate[0]), int(rate[1]), int(rate[2])])
    return rates

#生成用户评分数据结构
def getUserScoreDataStructure(rates):

    #userDict[2]=[(1,5),(4,2)].... 表示用户2对电影1的评分是5，对电影4的评分是2
    userDict = {}
    itemUser = {}
    for k in rates:
        user_rank = (k[1], k[2])
        if k[0] in userDict:
            userDict[k[0]].append(user_rank)
        else:
            userDict[k[0]] = [user_rank]

        if k[1] in itemUser:
            itemUser[k[1]].append(k[0])
        else:
            itemUser[k[1]] = [k[0]]
    return userDict, itemUser

#计算与指定用户最相近的邻居
def getNearestNeighbor(userId, userDict, itemUser):
    neighbors = []
    for item in userDict[userId]:
        for neighbor in itemUser[item[0]]:
            if neighbor != userId and neighbor not in neighbors:
                neighbors.append(neighbor)
    neighbors_dist = []
    for neighbor in neighbors:
        dist = getCosDist(userDict[userId], userDict[neighbor])
        neighbors_dist.append([dist, neighbor])
    neighbors_dist.sort(reverse = True)
    return neighbors_dist

#使用UserFC进行推荐，输入：文件名,用户ID,邻居数量
def recommendByUserFC(filename, userId, k = 5):

    #读取文件
    contents = readFile(filename)

    #文件格式数据转化为二维数组
    rates = getRatingInfo(contents)

    #格式化成字典数据
    userDict, itemUser = getUserScoreDataStructure(rates)

    #找邻居
    neighbors = getNearestNeighbor(userId, userDict, itemUser)[:5]

    #建立推荐字典
    recommand_dict = {}
    for neighbor in neighbors:
        neighbor_user_id = neighbor[1]
```

```
        movies = userDict[neighbor_user_id]
        for movie in movies:
            if movie[0] not in recommand_dict:
                recommand_dict[movie[0]] = neighbor[0]
            else:
                recommand_dict[movie[0]] += neighbor[0]

    #建立推荐列表
    recommand_list = []
    for key in recommand_dict:
        recommand_list.append([recommand_dict[key], key])
    recommand_list.sort(reverse = True)
    user_movies = [k[0] for k in userDict[userId]]
    return [k[1] for k in recommand_list], user_movies, itemUser, neighbors

#获取电影的列表
def getMovieList(filename):
    contents = readFile(filename)
    movies_info = {}
    for movie in contents:
        single_info = movie.split("|")
        movies_info[int(single_info[0])] = single_info[1:]
    return movies_info

#从这里开始运行
if __name__ == '__main__':

    reload(sys)
    sys.setdefaultencoding('utf-8')

    #获取所有电影的列表
    movies = getMovieList("u.item")
    recommend_list, user_movie, items_movie, neighbors = recommendByUserFC("u.data", 50, 80)
    neighbors_id=[ i[1] for i in neighbors]
    table = Texttable()
    table.set_deco(Texttable.HEADER)
    table.set_cols_dtype(['t', 't', 't'])
    table.set_cols_align(["l", "l", "l"])
    rows=[]
    rows.append([u"movie name",u"release", u"from userid"])
    for movie_id in recommend_list[:20]:
        from_user=[]
        for user_id in items_movie[movie_id]:
            if user_id in neighbors_id:
                from_user.append(user_id)
        rows.append([movies[movie_id][0],movies[movie_id][1],""])
    table.add_rows(rows)
    print table.draw()
```

推荐结果如下

| movie name | release |
|---|---|
| Toy Story (1995) | 01-Jan-1995 |
| Father of the Bride Part II (1995) | 01-Jan-1995 |
| First Wives Club, The (1996) | 14-Sep-1996 |
| Heathers (1989) | 01-Jan-1989 |
| Mission: Impossible (1996) | 22-May-1996 |
| Leaving Las Vegas (1995) | 01-Jan-1995 |
| Jerry Maguire (1996) | 13-Dec-1996 |
| Back to the Future (1985) | 01-Jan-1985 |
| Raiders of the Lost Ark (1981) | 01-Jan-1981 |
| 2001: A Space Odyssey (1968) | 01-Jan-1968 |
| Independence Day (ID4) (1996) | 03-Jul-1996 |
| Fargo (1996) | 14-Feb-1997 |
| Blade Runner (1982) | 01-Jan-1982 |
| Star Wars (1977) | 01-Jan-1977 |
| Apollo 13 (1995) | 01-Jan-1995 |
| Dead Man Walking (1995) | 01-Jan-1995 |
| Six Degrees of Separation (1993) | 01-Jan-1993 |
| Romy and Michele's High School Reunion (1997) | 25-Apr-1997 |
| Mediterraneo (1991) | 01-Jan-1991 |
| Island of Dr. Moreau, The (1996) | 23-Aug-1996 |

接下来再来看一个题目，这个题目是2014年阿里的大数据竞赛题目，描述可以参考如下链接

题目：**http://102.alibaba.com/competition/addDiscovery/gameIntroduce.**

题意：根据用户在天猫的4个月的行为日志，建立用户的品牌偏好，并预测他们在接下来的一个月内对品牌商品的

购买行为。开放的字段类型如下

| 字段 | 字段说明 | 提取说明 |
|---|---|---|
| user_id | 用户标记 | 抽样&字段加密 |
| Time | 行为时间 | 精度到天级别&隐藏年份 |
| action_type | 用户对品牌的行为类型 | 包括点击、购买、加入购物车、收藏4种行为<br>(点击：0 购买：1 收藏：2 购物车：3) |
| brand_id | 品牌数字ID | 抽样&字段加密 |

解析：**http://www.tuicool.com/articles/AN7Rf2**

上面的是建立的简单的模型，实际上在天猫，有基于行为簇的用户偏好分析。

协同过滤资料

1. 推荐引擎算法
2. 开源推荐引擎框架
3. 协调过滤介绍
4. **Slop one**
5. 基于**Map Reduce**的协同过滤推荐算法的并行实现
6. 并行协同过滤推荐模型的研究

上一篇 L-BFGS算法
下一篇 关于欧拉工程的一道递推题

我的同类文章

机器学习（34）

- SlopOne推荐算法 2015-06-03 阅读 1398
- 径向基（RBF）神经网络 2015-06-02 阅读 3180
- L2正则化方法 2015-05-30 阅读 5950
- 多项式回归模型（Office P... 2015-05-28 阅读 1965
- 房价预测（HackerRank） 2015-05-23 阅读 1965
- 贝叶斯学习及共轭先验 2015-10-13 阅读 4657
- 蒙特卡洛算法 2015-04-12
- L-BFGS算法 2015-03-29

参考知识库

**Python**知识库
19147 关注 | 1324 收录

**Hadoop**知识库
5932 关注 | 554 收录

**Apache Spark**知识库
6309 关注 | 401 收录

算法与数据结构知识库
13219 关注 | 2320 收录

## 猜你在找

用redis 搭建大数据 热门排行榜，用户推荐系统
ArcGIS for javascript 项目实战（环境监测系统）
C语言系列之 数组与算法实战
C语言系列之 字符串相关算法
C语言系列之 快速排序与全排列算法
Mahout协同过滤算法源码分析1
利用Solr完成向量乘以矩阵功能以协同过滤算法
windows下使用mahoutTaste实现协同过滤算法
协同过滤算法python实现简单入门详细注释
利用协同过滤算法的皮尔逊系数计算歌曲相似度

查看评论

8楼 左手121 2016-09-28 14:43发表

有问题。
A 关注了100个电影
B关注了100个电影
其中A和B只有一个电影的交集，且他们评分都是4.

如果按照你的计算余弦距离函数，求出来 A和B相似度为1

7楼 kellywong 2016-09-22 16:23发表

谢谢楼主的分享，我看完有三个问题
1.你在生成推荐列表的时候是不是没考虑到要把目标用户已经评价过的电影去掉啊，recommand_list[]，里面好像要包括目标用户已经评价过的电影了。
2.主方法中，从for movie_id in recommend_list[:20]:这里开始在干嘛啊，求解。
3.最后一个问题，请问楼主在协同过滤中训练集和测试集怎么使用来求误差啊（也就是movielens数据集中给定的u1.base,u1.test怎么样）
谢谢

6楼 qq_23863607 2016-05-27 16:56发表

楼主，你好，我也正在学习协同过滤，请问能分享一下这个代码数据集吗，因为刚接触python，所以很多不懂，希望借助代码和数据集，理解的去学习

Re: ACdreamers 2016-05-31 20:28发表

回复qq_23863607：代码的数据集我不是给了链接了么？

5楼 Bamboozq 2016-05-08 15:40发表

大神，我有个问题，一个是那个余弦相似度为什么会有那么多邻居用户和目标用户都是1，还有那个电影的推荐分值是不是没有归一化处理，谢谢

4楼 bigshowxin 2016-04-01 12:55发表

def getUserScoreDataStructure(rates):
userDict是格式化的用户评分数据，而itermUser是什么？有什么用？
博主，能讲解一下def getNearestNeighbor(userId, userDict, itemUser): 这个计算最相邻的方法吗？

3楼 听风吹雨he 2016-02-27 15:39发表

def recommendByUserFC(filename, userId, k = 5):此处参数k=5的起什么作用？和下面测试参数取80有什么区别？

Re: 洗来洗麻衣 2016-03-21 18:40发表

回复听风吹雨he：经过测试，那个80没有用，原因是：neighbors = getNearestNeighbor(userId, userDict, itemUser)[:5] 这句话没有用k参数，而是用了固定的5.把这里改成k，下面的80就会有效

2楼 ThankGodIFail 2015-11-29 16:59发表

一直都喜欢看你的博客，真的很好

Re: ACdreamers 2015-11-30 16:25发表

回复ThankGodIFail：谢谢夸奖，能给你帮助感到很高兴！

1楼 yu836618672 2015-07-29 17:44发表

膜拜荀神

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker
OpenStack VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu
WAP jQuery BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora YMI
LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE Ca
CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App
SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP
HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap