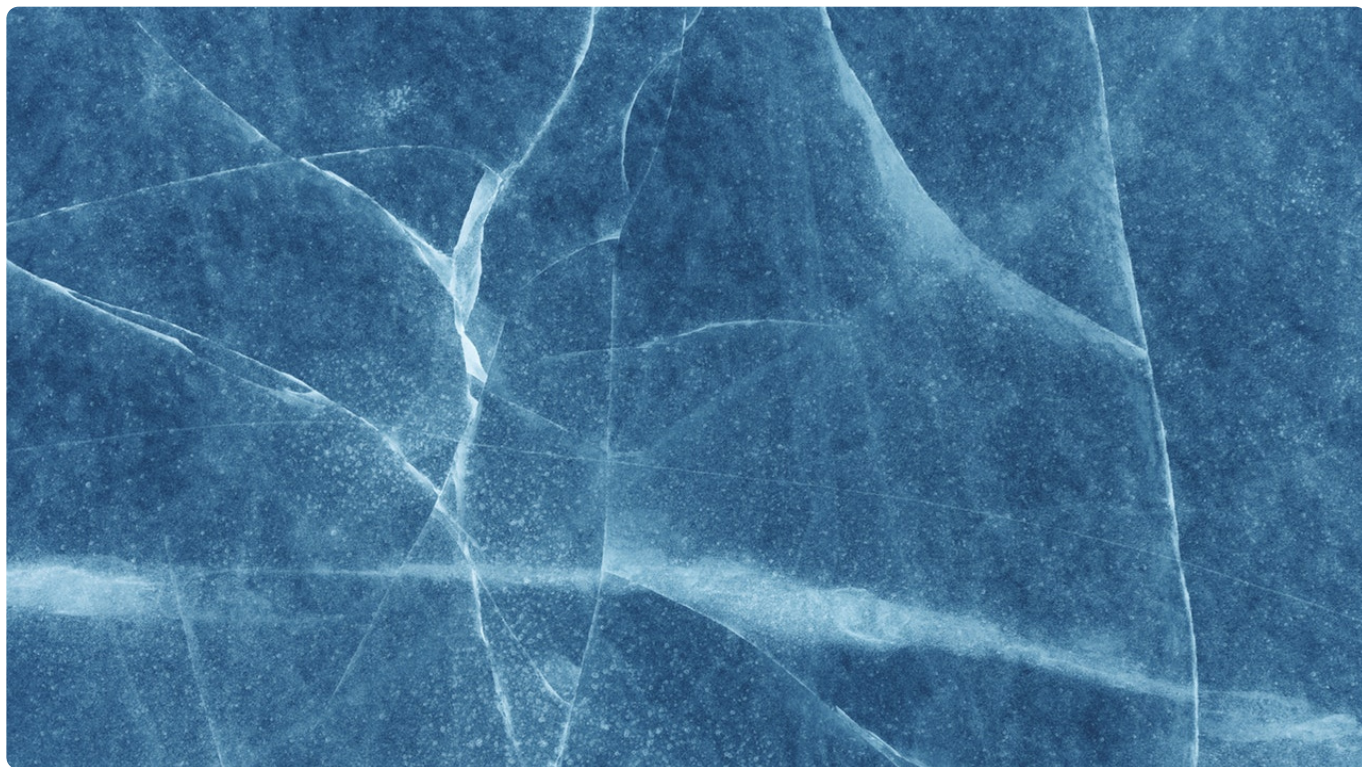


## 49 | 推荐系统（上）：如何实现基于相似度的协同过滤？

2019-04-08 黄申

程序员的数学基础课

[进入课程 >](#)



讲述：黄申

时长 08:02 大小 7.37M



你好，我是黄申。

个性化推荐这种技术在各大互联网站点已经普遍使用了，系统会根据用户的使用习惯，主动提出一些建议，帮助他们发现一些可能感兴趣的电影、书籍或者是商品等等。在这方面，最经典的案例应该是美国的亚马逊电子商务网站，它是全球最大的 B2C 电商网站之一。在公司创立之初，最为出名的就是其丰富的图书品类，以及相应的推荐技术。亚马逊的推荐销售占比可以达到整体销售的 30% 左右。可见，对于公司来说，推荐系统也是销售的绝好机会。因此，接下来的两节，我会使用一个经典的数据集，带你进行推荐系统核心模块的设计和实现。

### MovieLens 数据集

在开始之前，我们先来认识一个知名的数据集，MovieLens。你可以在它的[主页](#)查看详细的信息。这个数据集最核心的内容是多位用户对不同电影的评分，此外，它也包括了一些电影和用户的属性信息，便于我们研究推荐结果是不是合理。因此，这个数据集经常用来做推荐系统、或者其他机器学习算法的测试集。

时至今日，这个数据集已经延伸出几个不同的版本，有不同的数据规模和更新日期。我这里使用的是一个最新的小规模数据集，包含了 600 位用户对于 9000 部电影的约 10 万条评分，最后更新于 2018 年 9 月。你可以在这里下载：

<http://files.grouplens.org/datasets/movielens/ml-latest-small.zip>。

解压了这个 zip 压缩包之后，你会看到 readme 文件和四个 csv 文件（ratings、movies、links 和 tags）。其中最重要的是 ratings，它包含了 10 万条评分，每条记录有 4 个字段，包括 userId、movieId、rating、timestamp。userId 表示每位用户的 id，movieId 是每部电影的 ID，

rating 是这位用户对这部电影的评分，取值为 0-5 分。timestamp 是时间戳。而 movies 包含了电影的主要属性信息，title 和 genres 分别表示电影的标题和类型，一部电影可以属于多种类型。links 和 tags 则包含了电影的其他属性信息。我们的实验主要使用 ratings 和 movies 里的数据。

## 设计的整体思路

有了用于实验的数据，接下来就要开始考虑如何设计这个推荐系统。我在第 38 期讲解了什么是协同过滤推荐算法、基于用户的协同过滤和基于物品的协同过滤。这一节我们就以协同过滤为基础，分别实现基于用户和物品的过滤。

根据协同过滤算法的核心思想，整个系统可以分为三个大的步骤。

第一步，用户评分的标准化。因为有些用户的打分比较宽松，而有些用户打分则比较挑剔。所以，我们需要使用标准化或者归一化，让不同用户的打分具有可比性，这里我会使用 **z 分数** 标准化。

第二步，衡量和其他用户或者物品之间的相似度。我们这里的物品就是电影。在基于用户的过滤中，我们要找到相似的用户。在基于物品的过滤中，我们要找到相似的电影。我这里列出计算用户之间相似度  $us$  和物品之间相似度  $is$  的公式。之前我们讲过，这些都可以通过矩阵操作来实现。

$$us_{i1,i2} = \frac{X_{i1} \cdot X_{i2}}{\|X_{i1}\|_2 \times \|X_{i2}\|_2} = \frac{\sum_{j=1}^n x_{i1,j} \times x_{i2,j}}{\sqrt{\sum_{j=1}^n x_{i1,j}^2} \sqrt{\sum_{j=1}^n x_{i2,j}^2}}$$

$$is_{j1,j2} = \frac{X_{,j1} \cdot X_{,j2}}{\|X_{,j1}\|_2 \times \|X_{,j2}\|_2} = \frac{\sum_{i=1}^m x_{i,j1} \times x_{i,j2}}{\sqrt{\sum_{i=1}^m x_{i,j1}^2} \sqrt{\sum_{i=1}^m x_{i,j2}^2}}$$

我们以基于用户的过滤为例。假设我们使用夹角余弦来衡量相似度，那么我们就可以采用用户评分的矩阵点乘自身的转置来计算余弦夹角。用户评分的矩阵  $X$  中，每一行是某位用户的行向量，每个分量表示这位用户对某部电影的打分。而矩阵  $X'$  的每一列是某个用户的列向量，每个分量表示用户对某部电影的打分。

我们假设  $XX'$  的结果为矩阵  $Y$ ，那么  $y_{i,j}$  就表示用户  $i$  和用户  $j$  这两者喜好度向量的点乘结果，它就是夹角余弦公式中的分子。如果  $i$  等于  $j$ ，那么这个计算值也是夹角余弦公式分母的一部分。从矩阵的角度来看， $Y$  中任何一个元素都可能用于夹角余弦公式的分子，而对角线上的值会用于夹角余弦公式的分母。因此，我们可以利用  $Y$  来计算任何两个用户之间的相似度。

之前我们使用了一个示例讲解过对于基于用户的协同过滤，如何计算矩阵  $Y$ ，以及如何使用  $Y$  来计算余弦夹角，我这里列出来给你参考。



$$X = \begin{bmatrix} 0.11 & 0.20 & 0.0 \\ 0.81 & 0.0 & 0.0 \\ 0.0 & 0.88 & 0.74 \\ 0.0 & 0.0 & 0.42 \end{bmatrix}$$

$$X' = \begin{bmatrix} 0.11 & 0.81 & 0.0 & 0.0 \\ 0.20 & 0.0 & 0.88 & 0.0 \\ 0.0 & 0.0 & 0.74 & 0.42 \end{bmatrix}$$

$$Y = X \bullet X' = \begin{bmatrix} 0.11 & 0.20 & 0.0 \\ 0.81 & 0.0 & 0.0 \\ 0.0 & 0.88 & 0.74 \\ 0.0 & 0.0 & 0.42 \end{bmatrix} \begin{bmatrix} 0.11 & 0.81 & 0.0 & 0.0 \\ 0.20 & 0.0 & 0.88 & 0.0 \\ 0.0 & 0.0 & 0.74 & 0.42 \end{bmatrix}$$

$$= \begin{bmatrix} 0.0521 & 0.0891 & 0.176 & 0 \\ 0.0891 & 0.6561 & 0 & 0 \\ 0.176 & 0 & 1.322 & 0.3108 \\ 0 & 0 & 0.3108 & 0.1764 \end{bmatrix}$$

$$\begin{aligned}
 & \begin{bmatrix} 0.0521 & 0.0891 & 0.176 & 0 \\ 0.0891 & 0.6561 & 0 & 0 \\ 0.176 & 0 & 1.322 & 0.3108 \\ 0 & 0 & 0.3108 & 0.1764 \end{bmatrix} \\
 & \begin{aligned}
 & \text{Red boxes highlight } 0.0521, 0.0891, 0.6561, \text{ and } 0.482. \\
 & \text{Red arrows show the calculation of } 0.482 \text{ from the boxed values.}
 \end{aligned} \\
 & \begin{aligned}
 uS_{1,2} &= \frac{\sum_{j=1}^n x_{1,j} \times x_{1,j}}{\sqrt{\sum_{j=1}^n x_{1,j}} \sqrt{\sum_{j=1}^n x_{1,j}}} = \frac{0.0891}{\sqrt{0.0521} \sqrt{0.6561}} \approx 0.482
 \end{aligned} \\
 & US = \begin{bmatrix} 1 & 0.482 & 0.671 & 0 \\ 0.482 & 1 & 0 & 0 \\ 0.671 & 0 & 1 & 0.644 \\ 0 & 0 & 0.644 & 1 \end{bmatrix}
 \end{aligned}$$

第三步，根据相似的用户或物品，给出预测的得分  $p$ 。

$$\begin{aligned}
 p_{i,j} &= \frac{\sum_{k=1}^m uS_{i,k} \times x_{k,j}}{\sum_{k=1}^m uS_{i,k}} \\
 p_{i,j} &= \frac{\sum_{k=1}^n x_{i,k} \times is_{k,j}}{\sum_{k=1}^m is_{k,j}}
 \end{aligned}$$

之前我们也解释过如何使用矩阵操作来实现这一步。还是以基于用户的过滤为例。假设通过第二步，我们已经得到用户相似度矩阵  $US$ ， $US$  和评分矩阵  $X$  的点乘结果为矩阵  $USP$ 。沿用前面的示例，结果就是下面这样。

$$\begin{aligned}
 USP = US \cdot X &= \begin{bmatrix} 1 & 0.482 & 0.671 & 0 \\ 0.482 & 1 & 0 & 0 \\ 0.671 & 0 & 1 & 0.644 \\ 0 & 0 & 0.644 & 1 \end{bmatrix} \begin{bmatrix} 0.11 & 0.20 & 0.0 \\ 0.81 & 0.0 & 0.0 \\ 0.0 & 0.88 & 0.74 \\ 0.0 & 0.0 & 0.42 \end{bmatrix} \\
 &= \begin{bmatrix} 0.500 & 0.790 & 0.496 \\ 0.863 & 0.096 & 0 \\ 0.074 & 1.014 & 1.010 \\ 0 & 0.566 & 0.896 \end{bmatrix}
 \end{aligned}$$

然后对  $USP$  按行求和，获得矩阵  $USR$ 。

$$USR = \begin{bmatrix} 1.786 & 1.786 & 1.786 \\ 0.959 & 0.959 & 0.959 \\ 2.098 & 2.098 & 2.098 \\ 1.462 & 1.462 & 1.462 \end{bmatrix}$$


最终，我们使用  $USP$  和  $USR$  的元素对应除法，就可以求得任意用户对任意电影的评分矩阵  $P$ 。

$$P = \begin{bmatrix} 0.500 & 0.790 & 0.496 \\ 0.863 & 0.096 & 0 \\ 0.074 & 1.014 & 1.010 \\ 0 & 0.566 & 0.896 \end{bmatrix} / \begin{bmatrix} 1.786 & 1.786 & 1.786 \\ 0.959 & 0.959 & 0.959 \\ 2.098 & 2.098 & 2.098 \\ 1.462 & 1.462 & 1.462 \end{bmatrix} = \begin{bmatrix} 0.280 & 0.442 & 0.278 \\ 0.900 & 0.100 & 0 \\ 0.035 & 0.483 & 0.482 \\ 0 & 0.387 & 0.613 \end{bmatrix}$$

有了这个设计的思路，下面我们就可以使用 Python 进行实践了。

## 核心 Python 代码

在实现上述设计的三个主要步骤之前，我们还需要把解压后的 csv 文件加载到数组，并转为矩阵。下面我列出了主要的步骤和注释。需要注意的是，由于这个数据集中的用户和电影 ID 都是从 1 开始而不是从 0 开始，所以需要减去 1，才能和 Python 数组中的索引一致。

 复制代码

```
1 import pandas as pd
2 from numpy import *
3
4
5 # 加载用户对电影的评分数据
6 df = pd.read_csv("/Users/shenhuang/Data/ml-latest-small/ratings.csv")
7
8
9 # 获取用户的数量和电影的数量
10 user_num = df["userId"].max()
11 movie_num = df["movieId"].max()
12
13
14 # 构造用户对电影的二元关系矩阵
15 user_rating = [[0.0] * movie_num for i in range(user_num)]
16
17
18 i = 0
19 for index, row in df.iterrows(): # 获取每行的 index、row
20
21
22     # 由于用户和电影的 ID 都是从 1 开始，为了和 Python 的索引一致，减去 1
23     userId = int(row["userId"]) - 1
24     movieId = int(row["movieId"]) - 1
25
26
27     # 设置用户对电影的评分
28     user_rating[userId][movieId] = row["rating"]
29
30
31     # 显示进度
32     i += 1
33     if i % 10000 == 0:
34         print(i)
35
36
37 # 把二维数组转化为矩阵
38 x = mat(user_rating)
39 print(x)
```

加载了数据之后，第一步就是对矩阵中的数据，以行为维度，进行标准化。

```

1 # 标准化每位用户的评分数据
2 from sklearn.preprocessing import scale
3
4
5 # 对每一行的数据，进行标准化
6 x_s = scale(x, with_mean=True, with_std=True, axis=1)
7 print(" 标准化后的矩阵: ", x_s)

```

第二步是计算表示用户之间相似度的矩阵 US。其中，y 变量保存了矩阵 X 左乘转置矩阵 X' 的结果。而利用 y 变量中的元素，我们很容易就可以得到不同向量之间的夹角余弦。

```

1 # 获取 XX'
2 y = x_s.dot(x_s.transpose())
3 print("XX'的结果是': ", y)
4
5
6 # 获得用户相似度矩阵 US
7 us = [[0.0] * user_num for i in range(user_num)]
8 for userId1 in range(user_num):
9     for userId2 in range(user_num):
10         # 通过矩阵 Y 中的元素，计算夹角余弦
11         us[userId1][userId2] = y[userId1][userId2] / sqrt((y[userId1][userId1] * y[user:
12

```

在最后一步中，我们就可以进行基于用户的协同过滤推荐了。需要注意的是，我们还需要使用元素对应的除法来实现归一化。

```

1 # 通过用户之间的相似度，计算 USP 矩阵
2 usp = mat(us).dot(x_s)
3
4
5 # 求用于归一化的分母
6 usr = [0.0] * user_num
7 for userId in range(user_num):
8     usr[userId] = sum(us[userId])
9
10
11 # 进行元素对应的除法，完成归一化

```



```
12 p = divide(usp, mat(usr).transpose())
```

我们可以来看一个展示推荐效果的例子。在原始的评分数据中，我们看到 ID 为 1 的用户并没有对 ID 为 2 的电影进行评分。而在最终的矩阵 P 中，我们可以看系统对用户 1 给电影 2 的评分做出了较高的预测，换句话说，系统认为用户 1 很可能会喜好电影 2。进一步研究电影的标题和类型，我们会发现用户 1 对《玩具总动员》（1995 年）这类冒险类和动作类的题材更感兴趣，所以推荐电影 2《勇敢者的游戏》（1995 年）也是合理的。

## 总结

在今天的 content 中，我通过一个常用的实验数据，设计并实现了最简单的基于用户的协同过滤。我们最关心的是这个数据中，用户对电影的评分。有了这种二元关系，我们就能构建矩阵，并通过矩阵的操作来发现用户或物品之间的相似度，并进行基于用户或者物品的协同过滤。对于最终的计算结果，你可以尝试分析针对不同用户的推荐，看看协同过滤推荐的效果是不是合理。

在你分析推荐结果的时候，可能会参考 movie.csv 这个文件中所描述的电影类型。这些电影类型都是一开始人工标注好的。那么，有没有可能在没有这种标注数据的情况下，在一定程度上自动分析哪些电影属于同一个或者近似的类型呢？如果可以，有没有可能在这种自动划分电影类型的基础之上，给出电影的推荐呢？下一节，我会通过 SVD 奇异值分解，来进行这个方向的尝试。

## 思考题

今天我使用 Python 代码实现了基于用户的协同过滤。类似地，我们也可以采用矩阵操作来实现基于物品的协同过滤，请使用你擅长的语言来实现试试。

欢迎留言和我分享，也欢迎你在留言区写下今天的学习笔记。你可以点击“请朋友读”，把今天的内容分享给你的好友，和他一起精进。

# 程序员的数学基础课

在实战中重新理解数学

黄申

LinkedIn 资深数据科学家



新版升级：点击「👤 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 48 | 搜索引擎（下）：如何通过查询的分类，让电商平台的搜索结果更相关？

下一篇 50 | 推荐系统（下）：如何通过SVD分析用户和物品的矩阵？

## 精选留言 (3)

写留言



qinggeouy...

2019-04-21

👍 2

# 优化了一下，运行时间减少一半以上

#

<https://github.com/qinggeouye/GeekTime/blob/master/MathematicProgrammer/4>

import numpy as np...

展开 ▾

作者回复：感谢代码的优化 🙏



CarreyWan...

2019-04-08

1

您好，对于上面的代码有三个问题

第一：

# 设置用户对电影的评分

`i += 1`

`if i % 10000 == 0:...`

展开

作者回复: 第1个只是为了显示加载进度，因为在我的机器上有点慢。

第2个是很好的点，可以让分母加一个很小的值避免分母为0

第3个确实漏了一个括号，回头补上

感谢看得这么仔细，帮我找出了2个笔误



丹~

2019-04-28

1

老师好，感觉按照公式来看，USR作为分母，第一行应该是US的第一行元素求和

(  $1 + 0.482 + 0.671 + 0 = 2.153$  )，而不是USP的第一行求和

(  $0.500 + 0.790 + 0.496 = 1.786$  )。否则会像文中一样，p矩阵每行求和为1也是因为USR这样计算导致的。代码部分`usr[userId] = sum(us[userId])`应该是对的。不知我理解得对不对？

展开

作者回复: 我回头仔细看一下，可能是个笔误