《Python与数据挖掘》

第10章 智能推荐

讲师：武永亮

目录

智能推荐概述

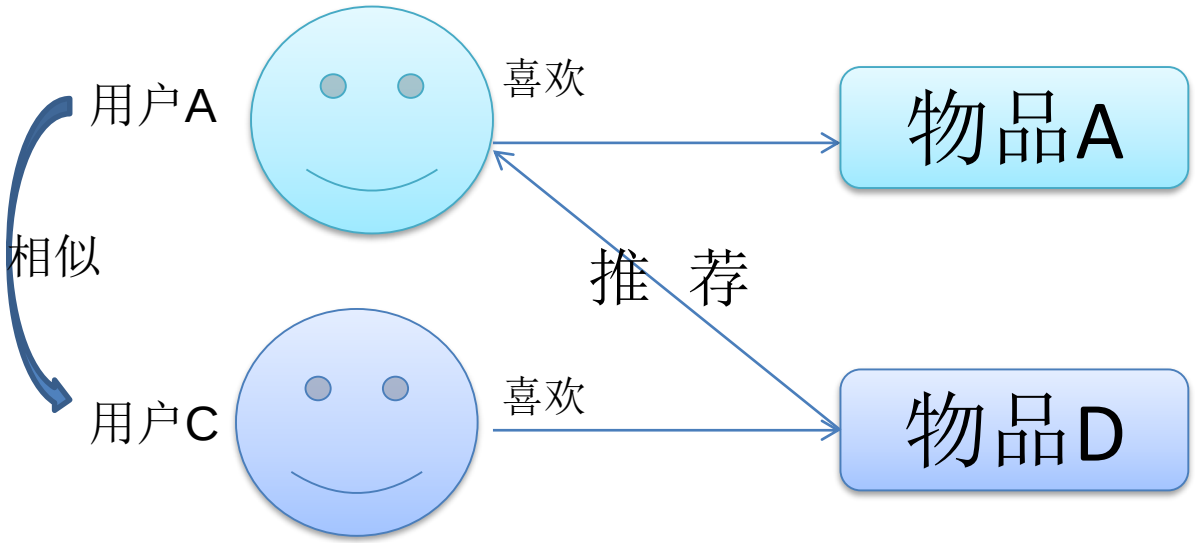
- 信息大爆炸时代来临，用户在面对大量的信息时无法从中迅速获得对自己真正有用的信息。
- 传统的搜索系统，需要用户提供明确需求，从用户提供的需求信息出发，继而给用户展现信息，无法针对不同用户的兴趣爱好提供相应地信息反馈服务。
- 推荐系统，相比于搜索系统，不需要用户提供明确需求，便可以为每一个用户实现个性化的推荐结果，让每个用户更便捷的获取信息。它是根据用户的兴趣特点和购买行为，向用户推荐用户感兴趣的信息和商品。

智能推荐的种类

- 智能推荐的方法有很多，常见的推荐技术主要分为：基于用户的协同过滤推荐和基于物品的协同过滤推荐。
- 基于用户的协同过滤的基本思想相当简单，基于用户对物品的偏好找到相邻邻居用户，然后将邻居用户喜欢的推荐给当前用户。
- 计算上，就是将一个用户对所有物品的偏好作为一个向量来计算用户之间的相似度，找到 K 邻居后，根据邻居的相似度权重以及他们对物品的偏好，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作为推荐。

基于用户的协同过滤

- 下图 给出了一个例子，对于用户 A，根据用户的历史偏好，这里只计算得到一个邻居 - 用户 C，然后将用户 C 喜欢的物品 D 推荐给用户 A。

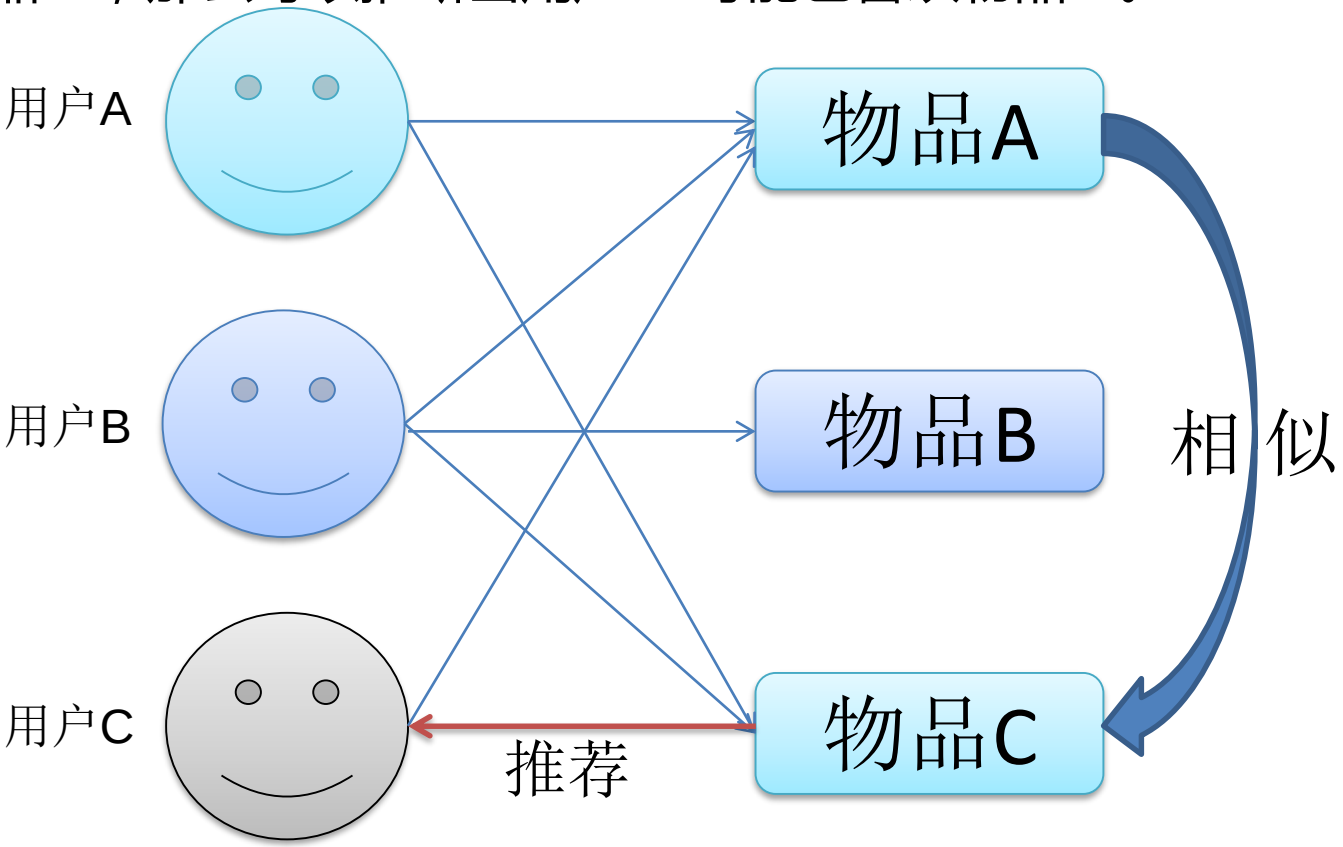


基于物品的协同过滤

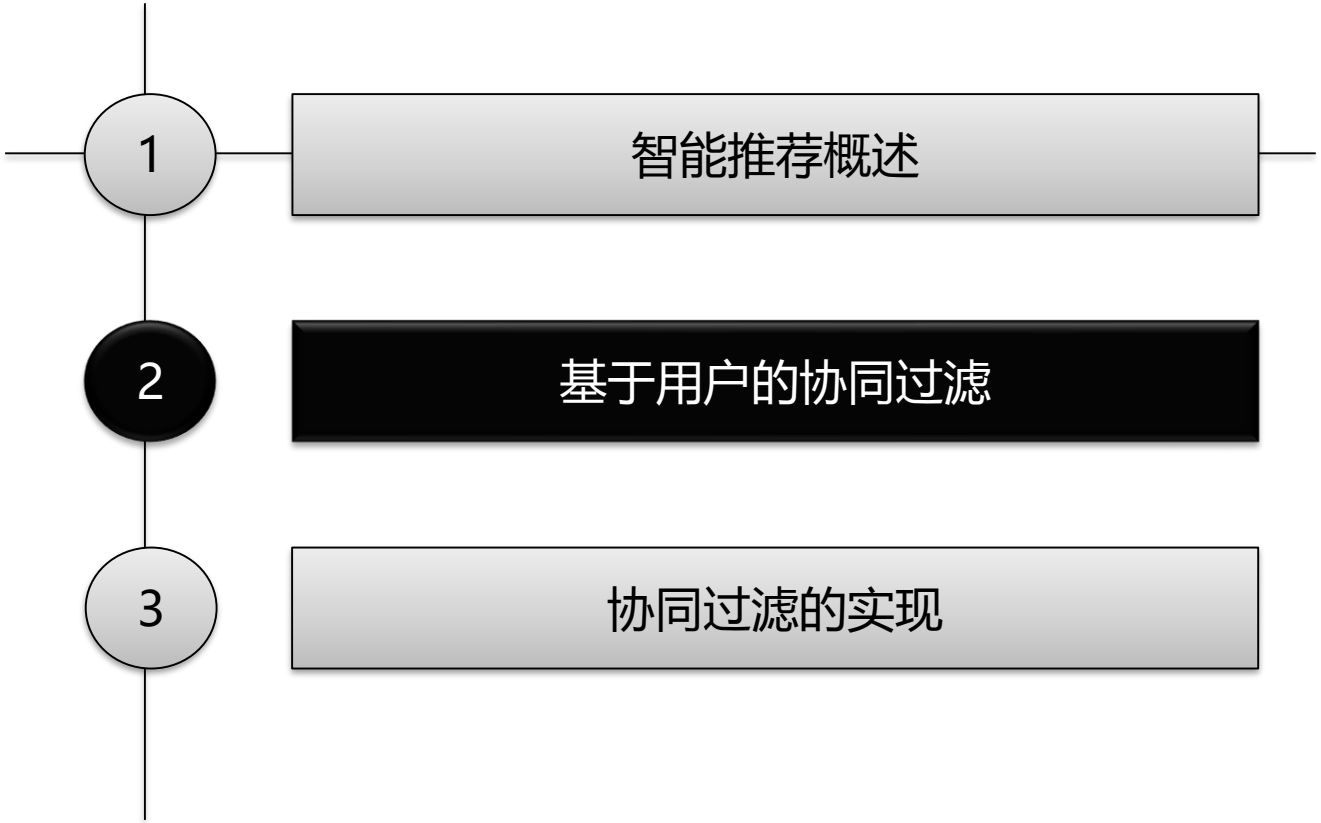
- 基于物品的协同过滤的原理和基于用户的协同过滤类似，只是在计算邻居时采用物品本身，而不是从用户的角度，即基于用户对物品的偏好找到相似的物品，然后根据用户的历史偏好，推荐相似的物品给他。
- 从计算的角度看，就是将所有用户对某个物品的偏好作为一个向量来计算物品之间的相似度，得到物品的相似物品后，根据用户历史的偏好预测当前用户还没有表示偏好的物品，计算得到一个排序的物品列表作为推荐。

基于物品的协同过滤

- 下图 给出了一个例子，对于物品 A，根据所有用户的历史偏好，喜欢物品 A 的用户都喜欢物品 C，得出物品 A 和物品 C 比较相似，而用户 C 喜欢物品 A，那么可以推断出用户 C 可能也喜欢物品 C。



目录



基于用户的协同过滤

- 以电影评分数据为例，实现基于用户的协同过滤算法第一个重要的步骤就是计算用户之间的相似度。而计算相似度，建立相关系数矩阵目前主要分为以下几种方法。
- 皮尔逊相关系数
- 皮尔逊相关系数一般用于计算两个定距变量间联系的紧密程度，它的取值在 [-1，+1] 之间。用数学公式表示，皮尔森相关系数等于两个变量的协方差除于两个变量的标准差。计算公式如下所示：

$$s(X,Y)=\frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

- 由于皮尔逊相关系数描述的是两组数据变化移动的趋势，所以在基于 用户的协同过滤系统中，经常使用。描述用户购买或评分变化的趋势，若趋势相近则皮尔逊系数趋近于1，也就是我们认为相似的用户。

基于用户的协同过滤

- 基于欧几里德距离的相似度余弦相似度
- 欧几里德距离计算相似度是所有相似度计算里面最简单、最易理解的方法
- 计算出来的欧几里德距离是一个大于0的数，为了使其更能体现用户之间的相似度，可以把它规约到(0, 1]之间，最终得到如下计算公式

$$s(X,Y)=\frac{1}{1+\sum\sqrt{(X_i-Y_i)^2}}$$

- 只要至少有一个共同评分项，就能用欧几里德距离计算相似度；如果没有共同评分项，那么欧几里德距离也就失去了作用。
- 其实照常理理解，如果没有共同评分项，那么意味着这两个用户或物品根本不相似。

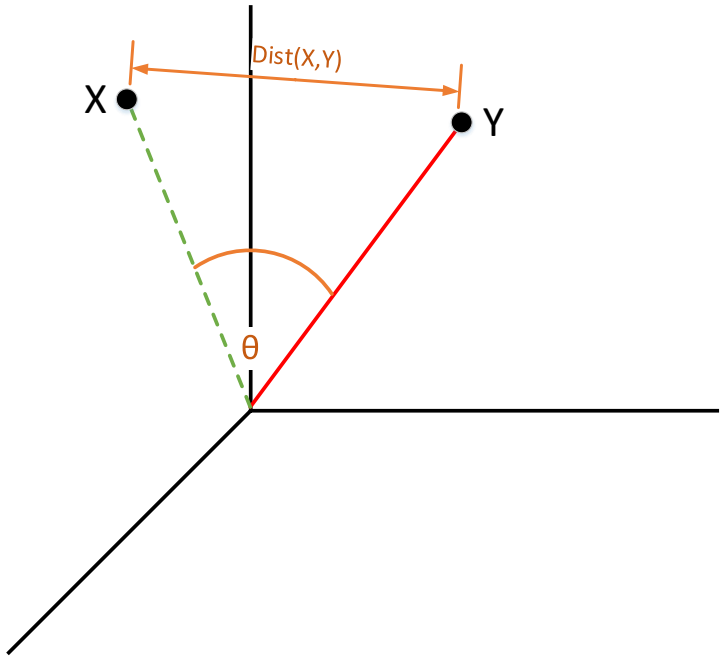
基于用户的协同过滤

- 余弦相似度
- 余弦相似度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。余弦相似度更加注重两个向量在方向上的差异，而非距离或长度上。计算公式如下所示：

$$s(X,Y)=\cos\theta=\frac{\vec{x}*\vec{y}}{\|x\|*\|y\|}$$

基于用户的协同过滤

- 从图上可以看出距离度量衡量的是空间各点间的绝对距离，跟各个点所在的位置坐标（即个体特征维度的数值）直接相关。
- 如果保持X点的位置不变，Y点朝原方向远离坐标轴原点，那么这个时候余弦相似度是保持不变的，因为夹角不变，而X、Y两点的距离显然在发生改变，这就是欧氏距离和余弦相似度的不同之处。



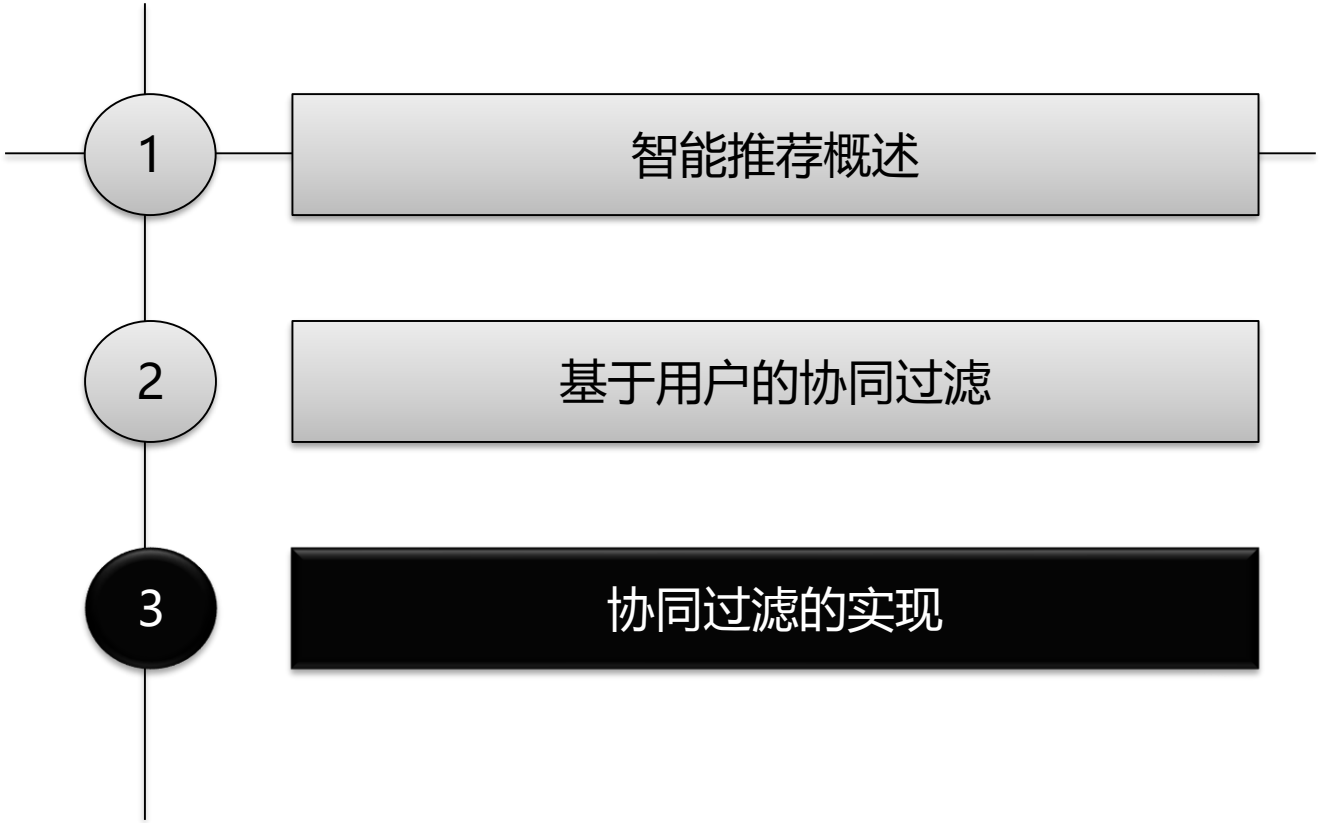
基于用户的协同过滤

- 基于用户的协同过滤算法，另一个重要的步骤就是计算用户 u对未评分商品的预测分值。
- 首先根据上一步中的相似度计算，寻找用户u 的邻居集N∈U, 其中N表示邻居集，U表示用户集。然后，结合用户评分数据集，预测用户 u对项 i的评分，计算公式如下所示：

$$p_{u,i} = \overline{r} + \frac{\sum_{u' \subset N} s(u-u')\left(r_{u',i} - \overline{r}_{u'}\right)}{\sqrt{\sum_{u' \subset N} \left| s(u-u') \right|}}$$

- 其中， s(u- u')表示用户 u和用户u' 的相似度。

目录



协同过滤的实现

- 下面通过个性化的电影推荐的例子演示基于用户的协同过滤算法在Python中的实现。
- 现在影视已经成为大众喜爱的休闲娱乐的方式之一，合理的个性化电影推荐一方面能够促进电影行业的发展，另一方面也可以让大众数量众多的电影中迅速得到自己想要的电影，从而做到两全齐美。甚至更近一步，可以明确市场走向，对后续电影的类型导向等起到重要作用。
- 现有的部分电影评分数据如下表 ：

用户ID	电影ID	电影评分	时间标签
1	1	5	874965758
1	2	3	876893171
1	3	4	878542960
1	4	3	876893119
1	5	3	889751712
1	6	4	875071561
1	7	1	875072484
…	…	…	…

协同过滤实现代码

- 在Python中实现基于用户的协同过滤推荐系统首先计算用户之间的相关系数。实现代码如下所示：

```
#使用基于UBCF算法对电影进行推荐
from __future__ import print_function
import pandas as pd
#############    主程序   ###############
if __name__ == "__main__":
    print("\n--------------使用基于UBCF算法对电影进行推荐 运行中... -----------\n")
    traindata = pd.read_csv('../data/u1.base',sep='\t', header=None,index_col=None)
    testdata = pd.read_csv('../data/u1.test',sep='\t', header=None,index_col=None)
    #删除时间标签列
    traindata.drop(3,axis=1, inplace=True)
    testdata.drop(3,axis=1, inplace=True)
    #行与列重新命名
    traindata.rename(columns={0:'userid',1:'movid',2:'rat'}, inplace=True)
    testdata.rename(columns={0:'userid',1:'movid',2:'rat'}, inplace=True)
    traindf=traindata.pivot(index='userid', columns='movid', values='rat')
    testdf=testdata.pivot(index='userid', columns='movid', values='rat')
    traindf.rename(index={i:'usr%d'%(i) for i in traindf.index} , inplace=True)
    traindf.rename(columns={i:'mov%d'%(i) for i in traindf.columns} , inplace=True)
    testdf.rename(index={i:'usr%d'%(i) for i in testdf.index} , inplace=True)
    testdf.rename(columns={i:'mov%d'%(i) for i in testdf.columns} , inplace=True)
    userdf=traindf.loc[testdf.index]
    #获取预测评分和推荐列表
    trainrats,trainrecomm=recomm(traindf,userdf)
```

Python输出结果

usr1([u'mov1290', u'mov1354', u'mov1678'], dtype='object', name=u'movid'),
usr2([u'mov1491', u'mov1354', u'mov1371'], dtype='object', name=u'movid'),
usr3([u'mov1304', u'mov1621', u'mov1678'], dtype='object', name=u'movid'),
usr4([u'mov1502', u'mov1659', u'mov1304'], dtype='object', name=u'movid'),
usr5([u'mov1304', u'mov1621', u'mov1472'], dtype='object', name=u'movid'),
usr6([u'mov1618', u'mov1671', u'mov1357'], dtype='object', name=u'movid'),
usr7([u'mov1472', u'mov1467', u'mov1374'], dtype='object', name=u'movid'),
usr8([u'mov1659', u'mov1316', u'mov1494'], dtype='object', name=u'movid'),
usr9([u'mov1621', u'mov1304', u'mov1491'], dtype='object', name=u'movid'),
usr10([u'mov1486', u'mov1494', u'mov437'], dtype='object', name=u'movid'),
usr11([u'mov1659', u'mov1654', u'mov1626'], dtype='object', name=u'movid'),
usr12([u'mov1659', u'mov1618', u'mov1661'], dtype='object', name=u'movid'),
usr13([u'mov1486', u'mov1494', u'mov1662'], dtype='object', name=u'movid'),
usr14([u'mov1661', u'mov1308', u'mov1671'], dtype='object', name=u'movid'),
usr15([u'mov1626', u'mov1671', u'mov1678'], dtype='object', name=u'movid'),
usr16([u'mov1618', u'mov1486', u'mov1494'], dtype='object', name=u'movid'),
usr17([u'mov1316', u'mov1621', u'mov1304'], dtype='object', name=u'movid'),
usr18([u'mov1618',u'mov1654',u'mov1626'], dtype='object', name=u'movid'),
usr19([u'mov1316', u'mov1661', u'mov1275'], dtype='object', name=u'movid'),
usr20([u'mov1659', u'mov1292', u'mov1304'], dtype='object', name=u'movid'),
.....
Total: 80000rows

结果分析

- 对输出结果进行解释：其中最前端格式为“usr+整数”字符串代表用户编号，“[]”内的字符串代表三部电影的编号，dtype为类型，name为字段名。
- 整体代表意思是，根据算法得出对用户usr1推荐他并未看过的三部电影，编号为：mov1290，mov1354，u'mov1678。

Thank You!