推荐系统第二次作业

Projects

1. See how well the Slope One recommender recommends movies for you. Rate 10 movies or so (ones that are in the MovieLens dataset). Does the recommender suggest movies you might like?

班级：电子商务 1402 班

姓名：李卫林、梁天华

1.

MovieLens data set

Let's try out the Slope One recommender on a different dataset. The MovieLens dataset—collected by the GroupLens Research Project at the University of Minnesota—contains user ratings of movies. The data set is available for download at www.grouplens.org. The data set is available in three sizes; for the demo here I am using the smallest one which contains 100,000 ratings (1-5) from 943 users on 1,682 movies. I wrote a short function that will import this data into the recommender class.

Let's give it a try.

Again, you can download the code to this chapter at www.guidetodatamining.com!

Projects

1. See how well the Slope One recommender recommends movies for you. Rate 10 movies or so (ones that are in the MovieLens dataset). Does the recommender suggest movies you might like?

Example：

```
>>> r.slopeOneRecommendations(r.data['25'])
[('Aiqing wansui (1994)', 5.674418604651163), ('Boys, Les (1997)',
5.523076923076923), ('Star Kid (1997)', 5.25), ('Santa with Muscles
(1996)',
```

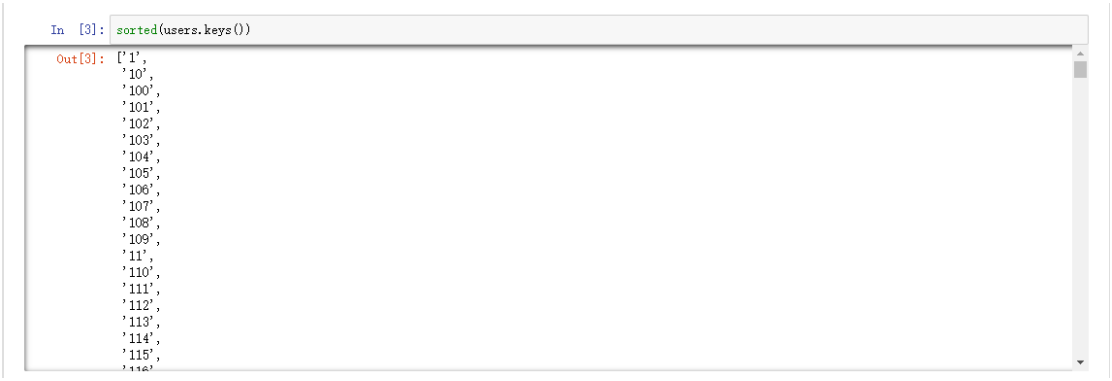
Save as (slopeone_homework.py)

```
from math import sqrt
from csv import *
users = {}
def trans_to_dict():
    '''读取 csv 转成 dictionary'''
    with open('ratings.csv', 'r')as csv_file:
        temp = '1'
        temp_dict = { }
        for i,line in enumerate(csv_file):
            row = line.split(',')
            if i != 0:
                if temp == row[0]:
                    temp_dict[row[1]] = float(row[2])
                else:
                    users[temp] = temp_dict
```

```
                    temp = row[0]
                    temp_dict = { }
                    temp_dict[row[1]] = float(row[2])
class recommender:
    def __init__(self, data, k=1):
        #以下变量将用于 Slope One 算法
        self.frequencies = {}
        self.deviations = {}
        self.k = k
        self.username2id = {}
        self.userid2name = {}
        self.productid2name = {}
        if type(data).__name__ == 'dict':
            self.data = data
    def convertProductID2name(self, id):
        if id in self.productid2name:
            return self.productid2name[id]
        else:
            return id
    def computeDeviations(self):
        # 遍历嵌套字典获取每位用户的评分数据
        for ratings in self.data.values():
        # 对于该用户的每个评分项
            for (item, rating) in ratings.items():
                self.frequencies.setdefault(item, {})
                self.deviations.setdefault(item, {})
                # 再次遍历该用户的每个评分项
                for (item2, rating2) in ratings.items():
                    if item != item2 :
                        self.frequencies[item].setdefault(item2, 0)
                        self.deviations[item].setdefault(item2, 0.0)
                        self.frequencies[item][item2] += 1
                        self.deviations[item][item2] += rating - rating2

        for (item, ratings) in self.deviations.items():
            for item2 in ratings:
                ratings[item2] /= self.frequencies[item][item2]
    def slopeOneRecommendations(self, userRatings):
        recommendations = {}
        frequencies = {}
        # 遍历目标用户的评分项
        for (userItem, userRating) in userRatings.items():
            # 对目标用户未评价的进行计算
            for (diffItem, diffRatings) in self.deviations.items():
```

```
                    if diffItem not in userRatings and userItem in self.deviations[diffItem]:
                        freq = self.frequencies[diffItem][userItem]
                        recommendations.setdefault(diffItem, 0.0)
                        frequencies.setdefault(diffItem, 0)
                        #计算分子
                        recommendations[diffItem] += (diffRatings[userItem] + userRating) *
freq
                        #计算分母
                        frequencies[diffItem] += freq
        recommendations =   [(self.convertProductID2name(k),
                              v / frequencies[k])
                              for (k, v) in recommendations.items()]
        # 将其排序并返回
        recommendations.sort(key=lambda artistTuple: artistTuple[1],reverse = True)
        return recommendations
#test1
trans_to_dict()
#print(users)
#print(users.keys())
#输出字典中所有的键
#r= recommender(users)
#r.computeDeviations()
#print(r.slopeOneRecommendations(users['3'])[0:5])
#输出推荐列表的前五个电影 ID。
```

```
In [2]: users
Out[2]: {'1': {'1029': 3.0,
               '1061': 3.0,
               '1129': 2.0,
               '1172': 4.0,
               '1263': 2.0,
               '1287': 2.0,
               '1293': 2.0,
               '1339': 3.5,
               '1343': 2.0,
               '1371': 2.5,
               '1405': 1.0,
               '1953': 4.0,
               '2105': 4.0,
               '2150': 3.0,
               '2193': 2.0,
               '2294': 2.0,
               '2455': 2.5,
               '2968': 1.0,
               '31': 2.5,
```



```
In [3]: sorted(users.keys())
Out[3]: ['1',
         '10',
         '100',
         '101',
         '102',
         '103',
         '104',
         '105',
         '106',
         '107',
         '108',
         '109',
         '11',
         '110',
         '111',
         '112',
         '113',
         '114',
         '115',
```

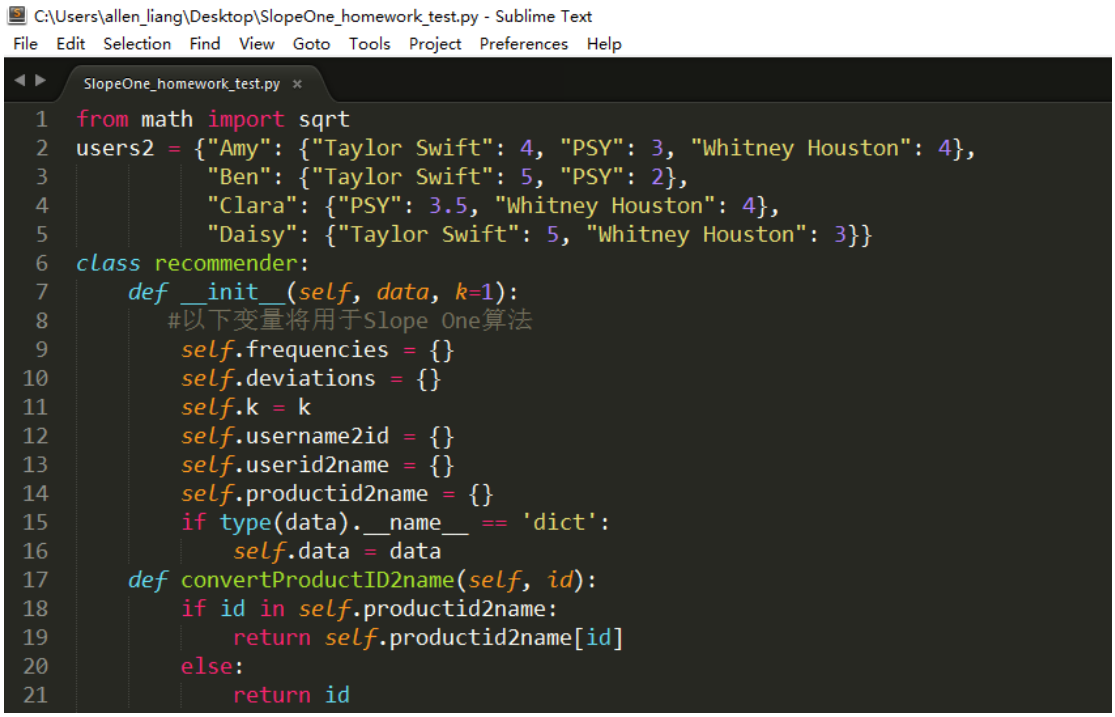
```
In [4]: r= recommender(users)
        r.computeDeviations()
        r.slopeOneRecommendations(users['3'])[0:5]
        #输出为用户ID为3的用户推荐电影ID列表的前五个。
Out[4]: [('107559', 6.846153846153846),
         ('109249', 6.5),
         ('4591', 6.166666666666667),
         ('4796', 6.166666666666667),
         ('1819', 6.166666666666667)]

In [ ]:
```

为用户 ID 为 3 的用户推荐电影 ID 列表的前五个结果如下：

```
[('107559', 6.846153846153846),
 ('109249', 6.5),
 ('4591', 6.166666666666667),
 ('4796', 6.166666666666667),
 ('1819', 6.166666666666667)]
```

2.save as SlopeOne_homework_test.py

重新认识理解 slopeone 算法




```python
from math import sqrt
users2 = {"Amy": {"Taylor Swift": 4, "PSY": 3, "Whitney Houston": 4},
          "Ben": {"Taylor Swift": 5, "PSY": 2},
          "Clara": {"PSY": 3.5, "Whitney Houston": 4},
          "Daisy": {"Taylor Swift": 5, "Whitney Houston": 3}}
class recommender:
    def __init__(self, data, k=1):
        #以下变量将用于Slope One算法
        self.frequencies = {}
        self.deviations = {}
        self.k = k
        self.username2id = {}
        self.userid2name = {}
        self.productid2name = {}
        if type(data).__name__ == 'dict':
            self.data = data
    def convertProductID2name(self, id):
        if id in self.productid2name:
            return self.productid2name[id]
        else:
            return id
```

```python
    def computeDeviations(self):
        # 遍历嵌套字典获取每位用户的评分数据
        for ratings in self.data.values():
        # 对于该用户的每个评分项（歌手、分数）
            for (item, rating) in ratings.items():
                self.frequencies.setdefault(item, {})
                self.deviations.setdefault(item, {})
                # 再次遍历该用户的每个评分项
                for (item2, rating2) in ratings.items():
                    if item != item2 :
                        self.frequencies[item].setdefault(item2, 0)
                        self.deviations[item].setdefault(item2, 0.0)
                        self.frequencies[item][item2] += 1
                        self.deviations[item][item2] += rating - rating2
                        print()
                        print('遍历1:ratings---->>>',ratings)
                        print('遍历2:frequencies---->>>',self.frequencies.items())
                        print('遍历3:deviations---->>>',self.deviations.items())
        for (item, ratings) in self.deviations.items():
            for item2 in ratings:
                ratings[item2] /= self.frequencies[item][item2]

```

Anaconda Prompt

```
(python35) C:\Users\allen_liang\Desktop>python SlopeOne_homework_test.py

遍历1:ratings---->>> {'Whitney Houston': 3, 'Taylor Swift': 5}
遍历2:frequencies---->>> dict_items([('Whitney Houston', {'Taylor Swift': 1})])
遍历3:deviations---->>> dict_items([('Whitney Houston', {'Taylor Swift': -2.0})])

遍历1:ratings---->>> {'Whitney Houston': 3, 'Taylor Swift': 5}
遍历2:frequencies---->>> dict_items([('Whitney Houston', {'Taylor Swift': 1}), ('Taylor Swift', {'Whitney Houston': 1})])
遍历3:deviations---->>> dict_items([('Whitney Houston', {'Taylor Swift': -2.0}), ('Taylor Swift', {'Whitney Houston': 2.0})])

遍历1:ratings---->>> {'PSY': 2, 'Taylor Swift': 5}
遍历2:frequencies---->>> dict_items([('PSY', {'Taylor Swift': 1}), ('Whitney Houston', {'Taylor Swift': 1}), ('Taylor Swift', {'Whitney Houston': 1})])
遍历3:deviations---->>> dict_items([('PSY', {'Taylor Swift': -3.0}), ('Whitney Houston', {'Taylor Swift': -2.0}), ('Taylor Swift', {'Whitney Houston': 2.0})])

遍历1:ratings---->>> {'PSY': 2, 'Taylor Swift': 5}
遍历2:frequencies---->>> dict_items([('PSY', {'Taylor Swift': 1}), ('Whitney Houston', {'Taylor Swift': 1}), ('Taylor Swift', {'PSY': 1, 'Whitney Houston': 1})])
遍历3:deviations---->>> dict_items([('PSY', {'Taylor Swift': -3.0}), ('Whitney Houston', {'Taylor Swift': -2.0}), ('Taylor Swift', {'PSY': 3.0, 'Whitney Houston': 2.0})
])

遍历1:ratings---->>> {'PSY': 3, 'Whitney Houston': 4, 'Taylor Swift': 4}
遍历2:frequencies---->>> dict_items([('PSY', {'Whitney Houston': 1, 'Taylor Swift': 1}), ('Whitney Houston', {'Taylor Swift': 1}), ('Taylor Swift', {'PSY': 1, 'Whitney
Houston': 1})])
遍历3:deviations---->>> dict_items([('PSY', {'Whitney Houston': -1.0, 'Taylor Swift': -3.0}), ('Whitney Houston', {'Taylor Swift': -2.0}), ('Taylor Swift', {'PSY': 3.0,
 'Whitney Houston': 2.0})])

遍历1:ratings---->>> {'PSY': 3, 'Whitney Houston': 4, 'Taylor Swift': 4}
遍历2:frequencies---->>> dict_items([('PSY', {'Whitney Houston': 1, 'Taylor Swift': 2}), ('Whitney Houston', {'Taylor Swift': 1}), ('Taylor Swift', {'PSY': 1, 'Whitney
Houston': 1})])
遍历3:deviations---->>> dict_items([('PSY', {'Whitney Houston': -1.0, 'Taylor Swift': -4.0}), ('Whitney Houston', {'Taylor Swift': -2.0}), ('Taylor Swift', {'PSY': 3.0,
 'Whitney Houston': 2.0})])

遍历1:ratings---->>> {'PSY': 3, 'Whitney Houston': 4, 'Taylor Swift': 4}
遍历2:frequencies---->>> dict_items([('PSY', {'Whitney Houston': 1, 'Taylor Swift': 2}), ('Whitney Houston', {'PSY': 1, 'Taylor Swift': 1}), ('Taylor Swift', {'PSY': 1,
 'Whitney Houston': 1})])
遍历3:deviations---->>> dict_items([('PSY', {'Whitney Houston': -1.0, 'Taylor Swift': -4.0}), ('Whitney Houston', {'PSY': 1.0, 'Taylor Swift': -2.0}), ('Taylor Swift',
{'PSY': 3.0, 'Whitney Houston': 2.0})])

遍历1:ratings---->>> {'PSY': 3, 'Whitney Houston': 4, 'Taylor Swift': 4}
遍历2:frequencies---->>> dict_items([('PSY', {'Whitney Houston': 1, 'Taylor Swift': 2}), ('Whitney Houston', {'PSY': 1, 'Taylor Swift': 2}), ('Taylor Swift', {'PSY': 1,
 'Whitney Houston': 1})])
遍历3:deviations---->>> dict_items([('PSY', {'Whitney Houston': -1.0, 'Taylor Swift': -4.0}), ('Whitney Houston', {'PSY': 1.0, 'Taylor Swift': -2.0}), ('Taylor Swift',
{'PSY': 3.0, 'Whitney Houston': 2.0})])
```

```python
    def slopeOneRecommendations(self, userRatings):
        recommendations = {}
        frequencies = {}
        # 遍历目标用户的评分项（歌手、分数）
        for (userItem, userRating) in userRatings.items():
            # 对目标用户未评价的歌手进行计算
            for (diffItem, diffRatings) in self.deviations.items():
                print()
                print('遍历1:deviations.items---->>>',self.deviations.items())
                if diffItem not in userRatings and userItem in self.deviations[diffItem]:
                    freq = self.frequencies[diffItem][userItem]
                    recommendations.setdefault(diffItem, 0.0)
                    frequencies.setdefault(diffItem, 0)
                    #计算分子
                    recommendations[diffItem] += (diffRatings[userItem] + userRating) * freq
                    #计算分母
                    frequencies[diffItem] += freq
        recommendations =  [(self.convertProductID2name(k),
                             v / frequencies[k])
                            for (k, v) in recommendations.items()]
        # 将其排序并返回
        recommendations.sort(key=lambda artistTuple: artistTuple[1],reverse = True)
        return recommendations
#test1
r= recommender(users2)
r.computeDeviations()
#test1
#print(r.deviations)
#test2
print(r.slopeOneRecommendations(users2['Ben']))
```

```
(python35) C:\Users\allen_liang\Desktop>python SlopeOne_homework_test.py

遍历1:deviations.items---->>> dict_items([('PSY', {'Taylor Swift': -2.0, 'Whitney Houston': -0.75}), ('Taylor Swift', {'PSY': 2.0, 'Whitney Houston': 1.0}), ('Whitney H
ouston', {'PSY': 0.75, 'Taylor Swift': -1.0})])

遍历1:deviations.items---->>> dict_items([('PSY', {'Taylor Swift': -2.0, 'Whitney Houston': -0.75}), ('Taylor Swift', {'PSY': 2.0, 'Whitney Houston': 1.0}), ('Whitney H
ouston', {'PSY': 0.75, 'Taylor Swift': -1.0})])

遍历1:deviations.items---->>> dict_items([('PSY', {'Taylor Swift': -2.0, 'Whitney Houston': -0.75}), ('Taylor Swift', {'PSY': 2.0, 'Whitney Houston': 1.0}), ('Whitney H
ouston', {'PSY': 0.75, 'Taylor Swift': -1.0})])

遍历1:deviations.items---->>> dict_items([('PSY', {'Taylor Swift': -2.0, 'Whitney Houston': -0.75}), ('Taylor Swift', {'PSY': 2.0, 'Whitney Houston': 1.0}), ('Whitney H
ouston', {'PSY': 0.75, 'Taylor Swift': -1.0})])

遍历1:deviations.items---->>> dict_items([('PSY', {'Taylor Swift': -2.0, 'Whitney Houston': -0.75}), ('Taylor Swift', {'PSY': 2.0, 'Whitney Houston': 1.0}), ('Whitney H
ouston', {'PSY': 0.75, 'Taylor Swift': -1.0})])

遍历1:deviations.items---->>> dict_items([('PSY', {'Taylor Swift': -2.0, 'Whitney Houston': -0.75}), ('Taylor Swift', {'PSY': 2.0, 'Whitney Houston': 1.0}), ('Whitney H
ouston', {'PSY': 0.75, 'Taylor Swift': -1.0})])
[('Whitney Houston', 3.375)]

(python35) C:\Users\allen_liang\Desktop>
```