推荐系统系列之一：近邻推荐

作者：周秀泽

邮箱：zhouxiuze@foxmail.com

程序地址：https://github.com/XiuzeZhou/Recommender-Systems

本文的 PDF 下载地址：https://github.com/XiuzeZhou/Recommender-Systems/tree/master/pdf

概述

协同：一些具有相似兴趣的人**共同给出**一个你可能感兴趣的列表；

过滤：从推荐的列表中**筛选**出你感兴趣的物品/服务。

核心思想：**人以群分 物以类聚**

人以群分 -> 基于用户的最近邻推荐

物以类聚 -> 基于物品的最近邻推荐

步骤：

1. 输入用户-商品的评分矩阵 $\mathbf{R}$；
2. 计算相似度，得到 K 个最近邻；
3. 预测未知项的评分；
4. 生成 top-n 推荐列表。

示例：

表 1：用户对电影的评分例子

						**用户均值**
小红	5	3	4	4	?	4.0
小橙	3	1	2	3	3	2.4
小黄	4	3	4	3	5	3.8
小绿	3	3	1	5	4	3.2
小青	1	5	5	2	1	2.8
电影均值	3.2	3.0	3.2	3.4	3.25	

1 基于用户的最近邻推荐

核心思想：你的喜好和**大部分人差不多**，总有几个品味特别一致的“好友”，他们喜欢的商品你也很可能会喜欢。

步骤：历史评分信息 —> **用户的相似度** —> K近邻用户 —> 近邻**共同评分**来计算评分

1) 非个性化方法

1). 非个性化方法

a. 所有参与过商品 i 评分的用户，即**商品的均值**

$$\hat{r}_{a,i} = \frac{1}{n_i}\sum_{b=1}^{n_i} r_{b,i}$$

其中，n_i 表示参加对商品 i 评分的人数。

最后，预测小红（a）给出的评分：$r_{a,5} = \frac{3+5+4+1}{4} = 3.25$

b. 考虑偏置项

原因：每个人的评分习惯不同：有些人喜欢给高评分，比如满意给5分，不满意给3分；有些人则比较鲜明，满意给5分，不满意给1分。所以，用每个人减去均值后的**偏差**来衡量喜欢程度。

$$\hat{r}_{a,i} = \bar{r}_a + \frac{1}{n_i}\sum_{b=1}^{n_i}(r_{b,i} - \bar{r}_b)$$

所以，预测小红给出的评分：$r_{a,5} = 4.0 + \frac{(3-2.4)+(5-3.8)+(4-3.2)+(1-2.8)}{4} = 4.2$

2) 个性化方法

相似度衡量方法——**Pearson 相关系数**：

$$s_{a,b}=\frac{\sum_{i\in I_{a,b}}\left(r_{a,i}-\overline{r}_a\right)\left(r_{b,i}-\overline{r}_b\right)}{\sqrt{\sum_{i\in I_{a,b}}\left(r_{a,i}-\overline{r}_a\right)^2}\sqrt{\sum_{i\in I_{a,b}}\left(r_{b,i}-\overline{r}_b\right)^2}}$$

a. 考虑用户间的相似度

$$\hat{r}_{a,i}=\frac{1}{\sum_{b\in N_a}s_{a,b}}\sum_{b\in N_a}s_{a,b}r_{b,i}$$

其中，N_a 表示用户 a 的近邻，$s_{a,u}$ 表示用户 a 和 u 之间的相似度。

小红（*a*）和小橙（*b*）之间的相似度：

$$s_{a,b}=\frac{\left(5-\overline{r}_a\right)\left(3-\overline{r}_b\right)+\left(3-\overline{r}_a\right)\left(1-\overline{r}_b\right)+\left(4-\overline{r}_a\right)\left(2-\overline{r}_b\right)+\left(4-\overline{r}_a\right)\left(3-\overline{r}_b\right)}{\sqrt{\left(5-\overline{r}_a\right)^2+\left(3-\overline{r}_a\right)^2+\left(4-\overline{r}_a\right)^2+\left(4-\overline{r}_a\right)^2}\sqrt{\left(3-\overline{r}_b\right)^2+\left(1-\overline{r}_b\right)^2+\left(2-\overline{r}_b\right)^2+\left(3-\overline{r}_b\right)^2}}=0.84$$

表 2：用户之间的相似度

	小红	小橙	小黄	小绿	小青
小红	1.00	0.84	0.61	0.00	-0.77
小橙	0.84	1.00	0.47	0.49	-0.90
小黄	0.61	0.47	1.00	-0.16	-0.47
小绿	0.00	0.49	-0.16	1.00	-0.64
小青	-0.77	-0.90	-0.47	-0.64	1.00

最后，预测小红给出的评分：

I.当选择1个近邻的时候，即最近邻：$r_{a,5}=3$

II.当选择2个近邻的时候：$r_{a,5}=\frac{0.84\times3+0.61\times5}{0.84+0.61}=3.84$

b. 考虑偏置项

$$\hat{r}_{a,i}=\overline{r}_a+\frac{1}{\sum_{b\in N_a}s_{a,b}}\sum_{b\in N_a}s_{a,b}\left(r_{b,i}-\overline{r}_b\right)$$

最后，预测小红给出的评分：

I.当选择1个近邻的时候，即最近邻：$r_{a,5}=4+(3-2.4)=4.6$

II.当选择2个近邻的时候：$r_{a,5}=4+\frac{0.84\times(3-2.4)+0.61\times(5-3.8)}{0.84+0.61}=4.85$

III.当选择3个近邻的时候：$r_{a,5}=4+\frac{0.84\times(3-2.4)+0.61\times(5-3.8)+0\times(4-3.2)}{0.84+0.61+0}=4.85$

IV.当选择4个近邻的时候：$r_{a,5}=4+\frac{0.84\times(3-2.4)+0.61\times(5-3.8)+0\times(4-3.2)+(-0.77)\times(1-2.8)}{0.84+0.61+0+(-0.77)}=7.86$

==>**近邻的选择很重要！**

3) 存在的问题

a. 数据稀疏性问题

几百万、几千万甚至上亿的商品中，用户只有几百条评分记录，用户之间的重叠的商品较少，导致近邻用户的相似度过低。

b. 计算代价

千万、甚至上亿的用户之间两两计算相似度，代价大。

c. 用户的属性变化

比如一个高中生一段时间内，买的都是练习和考试用书。等他进入大学后，买的就是专业教材。

2 基于物品的最近邻推荐

原因：在基于用户的方法中，随着用户数量的不断增多，在大数量级的用户范围内进行"最近邻搜索"会成为整个算法的瓶颈。基于物品的方法通过计算商品之间的相似性来代替用户之间的相似性。对于商品来讲，它们之间的相似性要稳定很多。因此可以离线完成工作量最大的相似性计算步骤，从而大大降低了在线计算量，提高推荐效率。

出处：基于物品的推荐：《Amazon.com Recommendations Item-to-Item Collaborative Filtering》

核心思想：用户喜欢的东西总在**一定的范围内**，喜欢的商品总是与**以前**购买的**非常相关**。

步骤：历史评分信息 —> **物品的相关性** —> 未知商品的K个**有评分的**近邻 —> 计算评分

相似度衡量方法

a. 找出同时对 *a* 和 *b* 打过分的组合；

b. 对这些组合进行相似度计算。

1) 非个性化方法

a. 用户的所有评分的均值，即**用户的均值**

$$\hat{r}_{a,i} = \frac{1}{n_a}\sum_{j=1}^{n_a} r_{a,j}$$

其中，n_a 表示用户评分过的商品数目。

最后，预测小红（*a*）给出的评分：$r_{a,5} = \frac{5+3+4+4}{4} = 4.0$

b. 考虑偏置项

$$\hat{r}_{a,i} = \overline{r}_i + \frac{1}{n_a}\sum_{j=1}^{n_a}(r_{a,j} - \overline{r}_j)$$

所以，预测小红给出的评分：$r_{a,5} = 3.25 + \frac{(5-3.2)+(3-3.0)+(4-3.2)+(4-3.4)}{4} = 4.05$

2) 个性化方法

a. 考虑商品间的相似度

$$\hat{r}_{a,i} = \frac{1}{\sum_{j\in N_{a,i}} s_{i,j}}\sum_{j\in N_{a,i}} s_{i,j} r_{a,j}$$

Cosine 相似度 ：

$$s_{a,b}=\frac{\overrightarrow{a}\cdot\overrightarrow{b}}{|\overrightarrow{a}|\times|\overrightarrow{b}|}=\frac{\sum_{u\in U_{a,b}}r_{u,a}r_{u,b}}{\sqrt{\sum_{u\in U_{a,b}}r_{u,a}^2}\sqrt{\sum_{u\in U_{a,b}}r_{u,b}^2}}$$

其中，$N_{a,i}$ 表示用户 a 评分过的商品中 i 的近邻。

（ *a* ）和（ *b* ）之间的相似度：

$$s_{a,b}=\frac{[3,5,4,1]\times[3,4,3,1]}{\sqrt{3^2+5^2+4^2+1^2}\sqrt{3^2+4^2+3^2+1^2}}=0.99$$

表 3：商品之间的相似度

	拯救大兵瑞恩	千与千寻	机器人瓦里	泰坦尼克号	肖申克的救赎
拯救大兵瑞恩	1.00	0.78	0.82	0.94	0.99
千与千寻	0.78	1.00	0.94	0.85	0.74
机器人瓦里	0.82	0.94	1.00	0.78	0.72
泰坦尼克号	0.94	0.85	0.78	1.00	0.94
肖申克的救赎	0.99	0.74	0.72	0.94	1.00

最后，预测小红给出的评分：

I.当选择1个近邻的时候，即最近邻：$r_{a,5}=5$

II.当选择2个近邻的时候：$r_{a,5}=\frac{0.99\times5+0.94\times4}{0.99+0.94}=4.51$

b. 考虑偏置项

改进的 Cosine 相似度：

$$s_{a,b}=\frac{\sum_{u\in U_{a,b}}\left(r_{u,a}-\overline{r}_u\right)\left(r_{u,b}-\overline{r}_u\right)}{\sqrt{\sum_{u\in U_{a,b}}\left(r_{u,a}-\overline{r}_u\right)^2}\sqrt{\sum_{u\in U_{a,b}}\left(r_{u,b}-\overline{r}_u\right)^2}}$$

（ *a* ）和（ *b* ）之间的相似度：

表 4：商品之间的相似度

	拯救大兵瑞恩	千与千寻	机器人瓦里	泰坦尼克号	肖申克的救赎
拯救大兵瑞恩	1.00	-0.94	-0.55	0.27	0.84
千与千寻	-0.94	1.00	0.62	-0.36	-0.91
机器人瓦里	-0.55	0.62	1.00	-0.88	-0.76
泰坦尼克号	0.27	-0.36	-0.88	1.00	0.43
肖申克的救赎	0.84	-0.91	-0.76	0.43	1.00

最后，预测小红给出的评分：

I.当选择1个近邻的时候，即最近邻：$r_{a,5}=5$

II.当选择2个近邻的时候：$r_{a,5}=\frac{0.84\times5+0.43\times4}{0.84+0.43}=4.66$

III.当选择3个近邻的时候：$r_{a,5}=\frac{0.84\times5+0.43\times4-0.76\times4}{0.84+0.43-0.76}=5.65$

IV.当选择4个近邻的时候：$r_{a,5}=\frac{0.84\times5+0.43\times4-0.76\times4-0.91\times3}{0.84+0.43-0.76-0.91}=0.375$

==>**近邻的选择很重要！**

3) 优缺点

a. 优点

当用户比商品数多时，相似度相对更稳定；

可离线计算商品相似度，快速进行线上预测。

b. 缺点

计算量大；数据稀疏性问题。

3 代码示例

数据集：movielens100k

环境：Jupyter Notebook

下载地址：https://github.com/XiuzeZhou/Recommender-Systems/blob/master/KNN.ipynb

1) 基于用户的近邻推荐

```
import numpy as np
import pandas as pd
import math
from sklearn.model_selection import train_test_split
from data import *
from evaluation import *
%matplotlib inline
```

数据集读取 并展示部分数据

```
data_col = ['user_id','item_id','rating','timestamp']

item_col =
['movie_id','movie_title','release_date','video_release_date','IMDb_URL','unknow
n','Action','Adventure','Animation',"Children's",'Comedy','Crime','Documentary',
'Drama','Fantasy','Film-Noir','Horror','Musical','Mystery','Romance','Sci-
Fi','Thriller','War','Western']

#总数据包含了用户，物品，评分
data_dir = 'datasets/ml-100k/u.data'
data = pd.read_table(data_dir,header=None,names=data_col,parse_dates=
['timestamp'])

#物品详细数据
item_dir = 'datasets/ml-100k/u.item'
item = pd.read_table(item_dir,header=None,names=item_col,parse_dates=
['release_date','video_release_date'],encoding='ISO-8859-1',sep='|')
```

```
item.head()
```

	movie_id	movie_title	release_date	video_release_date	IMDb_URL	unknown	Action	Adventure
0	1	Toy Story (1995)	1995-01-01	NaT	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	0	0
1	2	GoldenEye (1995)	1995-01-01	NaT	http://us.imdb.com/M/title-exact?GoldenEye%20(...	0	1	1
2	3	Four Rooms (1995)	1995-01-01	NaT	http://us.imdb.com/M/title-exact?Four%20Rooms%...	0	0	0
3	4	Get Shorty (1995)	1995-01-01	NaT	http://us.imdb.com/M/title-exact?Get%20Shorty%...	0	1	0
4	5	Copycat (1995)	1995-01-01	NaT	http://us.imdb.com/M/title-exact?Copycat%20(1995)	0	0	0

5 rows × 24 columns

```
data.head()
```

	user_id	item_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

加载数据

将id标准化：新id从0开始按1递增，数据格式：(user, item, rating)

```
N, M, data_list,item_ids_dict = load_data(file_dir=data_dir)
print(' data length: %d \n user number: %d \n item number: %d' %
(len(data_list),N,M))
```

data length: 100000
user number: 943
item number: 1682

```
new_item_information = item.insert(0,'new_id',[item_ids_dict[old_idx] for
old_idx in item['movie_id']])
item.sort_values('new_id',inplace=False).set_index(['new_id']).reset_index().hea
d()
```

	new_id	movie_id	movie_title	release_date	video_release_date	IMDb_URL	unknown	Action
0	0	242	Kolya (1996)	1997-01-24	NaT	http://us.imdb.com/M/title-exact?Kolya%20(1996)	0	0
1	1	302	L.A. Confidential (1997)	1997-01-01	NaT	http://us.imdb.com/M/title-exact?L%2EA%2E+Conf...	0	0
2	2	377	Heavyweights (1994)	1994-01-01	NaT	http://us.imdb.com/M/title-exact?Heavyweights%...	0	0
3	3	51	Legends of the Fall (1994)	1994-01-01	NaT	http://us.imdb.com/M/title-exact?Legends%20of%...	0	0
4	4	346	Jackie Brown (1997)	1997-01-01	NaT	http://us.imdb.com/M/title-exact?imdb-title-11...	0	0

5 rows × 25 columns

分割数据集 (训练集:测试集=8:2)

```
train_list, test_list = train_test_split(data_list,test_size=0.2)
print ('train length: %d \n test length: %d' %(len(train_list),len(test_list)))
```

train length: 80000
test length: 20000

转化数据成矩阵格式

```
train_mat = sequence2mat(sequence = train_list, N = N, M = M)
test_mat = sequence2mat(sequence = test_list, N = N, M = M)
train_mat[0,:20]
```

array([0., 0., 0., 0., 0., 0., 0., 0., 0., 2., 0., 0., 0., 0., 0., 0., 0., 0.])

计算相似度

```
# 计算 a 和 b 之间相似度
def calculate_similarity(a, b, model='pearson', minimum_common_items=5):
    assert a.shape==b.shape
    dim = len(a.shape) #向量维度
    common_items = a*b>0 # 共同评分的项
    common_size = np.sum(common_items,axis=dim-1)

    if model=='pearson':
```

```
        mean_a = np.sum(a,axis=dim-1)/np.sum(a>0,axis=dim-1)
        mean_b = np.sum(b,axis=dim-1)/np.sum(b>0,axis=dim-1)
        if dim ==1: #若是两个列向量
            aa = (a - mean_a)*common_items
            bb = (b - mean_b)*common_items
        else:
            aa = (a - np.reshape(mean_a, (-1,1)))*common_items
            bb = (b - np.reshape(mean_b, (-1,1)))*common_items
    else: #consine
        mean_u = np.sum(b,axis=0)/np.sum(b>0,axis=0)
        aa = (a - mean_u)*common_items
        bb = (b - mean_u)*common_items

    sim = np.sum(aa*bb,axis=dim-1)/(np.sqrt(np.sum(aa**2,axis=dim-
1))*np.sqrt(np.sum(bb**2, axis=dim-1))+1e-10)
    least_common_items = common_size>minimum_common_items # 共同评分的商品不少于
least_common_items
    return sim*least_common_items


# 计算用户之间的相似度
def similarity_matrix(mat, model='pearson', minimum_common_items=5):
    n,m = mat.shape
    sim_list=[]
    for u in range(n):
        a = np.tile(mat[u,:], (n,1))
        b = mat
        if model=='pearson':
            sim = calculate_similarity(a, b, model='pearson',
minimum_common_items=minimum_common_items)
        else: # consine
            sim = calculate_similarity(a, b, model='consine',
minimum_common_items=minimum_common_items)
        sim_list.append(sim)
        if u % 100 ==0:
            print(u)
    return np.array(sim_list)
```

```
sim_mat = similarity_matrix(mat=train_mat, model='pearson')
neighbors = np.argsort(-np.array(sim_mat)) # 获取近邻
sim_sort = -1*np.sort(-np.array(sim_mat)) # 获取对应近邻的相似度
```

查看用户 0 的信息

```
np.set_printoptions(precision=4, suppress=True)
print('user 0:')
print('neighbors:') # 用户0的近邻
print(neighbors[0,:10])
print('sim:\n') # 用户0 的近邻相似度
print(sim_sort[0,:10])
print('similarity_mat:') # 用户之间的相似度矩阵
print(sim_mat[:6,:6])
```

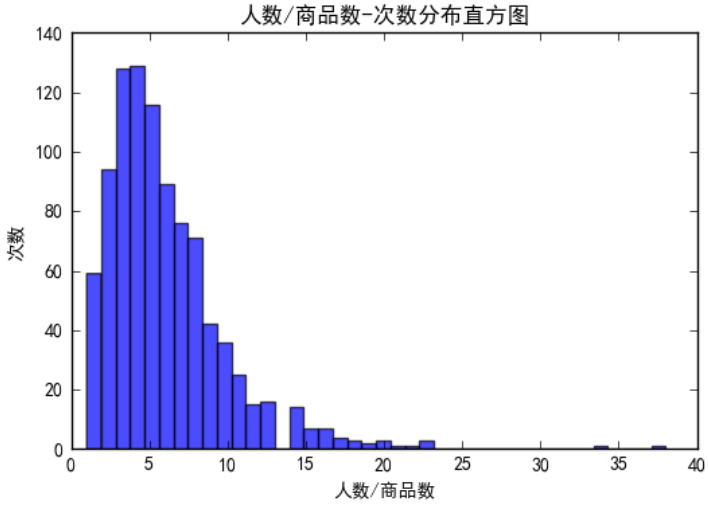
```
user 0:
neighbors:
[  0  20  87 887 123  25 660 928 876  54]
sim:
[ 1.     0.8112  0.7276  0.7175  0.7076  0.7043  0.6884  0.6744  0.654
  0.6533]
similarity_mat:
[[ 1.    -0.     0.3777 -0.1348 -0.     0.0008]
 [-0.     1.     0.427  -0.0791  0.     0.0339]
 [ 0.3777  0.427  1.     0.1021  0.     0.3127]
 [-0.1348 -0.0791  0.1021  1.     0.    -0.0071]
 [-0.     0.     0.     0.     1.     0.    ]
 [ 0.0008  0.0339  0.3127 -0.0071  0.     1.    ]]
```

近邻K的取值

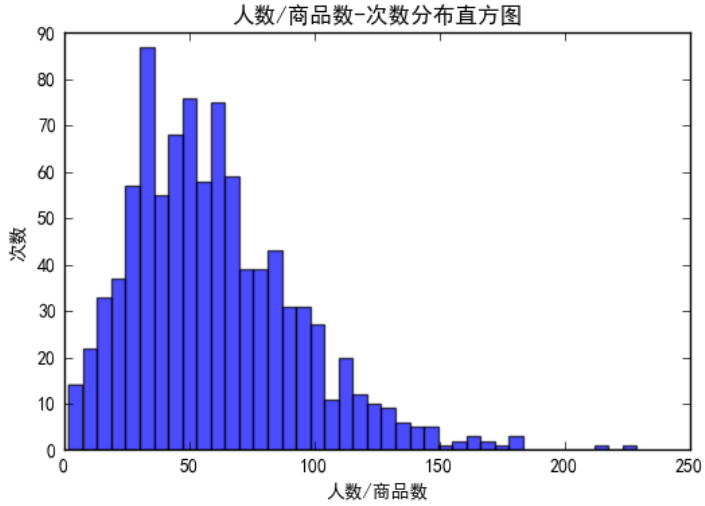
经验取值：一般取 10-50

```
def get_K(sim_mat, min_similarity=0.5):
    num = np.sum(sim_mat[:,1:]>min_similarity, axis=1) #统计用户大于min_similarity
的评分数

    # 画图
    plt.rcParams['font.sans-serif'] = [u'SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.hist(num, bins=40, normed=0, facecolor="blue", edgecolor="black",
alpha=0.7)
    plt.xlabel(u"人数/商品数")
    plt.ylabel(u"次数")
    plt.title(u"人数/商品数-次数分布直方图")
    plt.show()

    num_sort = np.sort(-num)
    line = int(0.8*len(sim_mat))
    K = -1*num_sort[line]
    return K
```

```
min_similarity=0.8
K = get_K(sim_mat, min_similarity=min_similarity)
print('min_similarity:',min_similarity,'K:',K)

min_similarity=0.5
K = get_K(sim_mat, min_similarity=min_similarity)
print('min_similarity:',min_similarity,'K:',K)
```



('min_similarity:', 0.8, 'K:', 3)



('min_similarity:', 0.5, 'K:', 32)

上图满足条件的用户明显偏少，下图的分布更近正态分布。故取 K=32。

预测评分

```
def prediction(train_mat, sim_mat, K=1, model='user_based'):
    assert len(train_mat.shape)>1
    n,m = train_mat.shape

    if  model=='user_based':
        sim_sort = -1*np.sort(-np.array(sim_mat))[:,1:K+1] # 除去最相似的自己
        neighbors = np.argsort(-np.array(sim_mat))[:,1:K+1]
        common_items = train_mat[neighbors]>0
        mean_user =
np.reshape(np.sum(train_mat,axis=1)/np.sum(train_mat>0,axis=1), (-1,1))
        mat_m = train_mat - mean_user
        aa =
np.sum(sim_sort[:,:,np.newaxis]*mat_m[neighbors]*common_items,axis=1)
        bb = np.sum(sim_sort[:,:,np.newaxis]*common_items,axis=1)+1e-10 #1e-10保
证分母不为0
        r_pred = mean_user + aa/bb
        return r_pred
    else: # 'item_based'
        r_pred=[]
        for u in range(n):
```

```
            u_mat = np.tile(train_mat[u],(m,1)) # m份用户u的记录,m*m
            rated_items_sim = (u_mat>0)*sim_mat # 保留有评分记录的相似度 m*m
            sim_sort = -1*np.sort(-np.array(rated_items_sim))[:,:K] # m*K
            neighbors = np.argsort(-np.array(rated_items_sim))[:,:K] # m*K
            neighbor_ratings = np.array([u_mat[i,neighbors[i]] for i in
range(m)])# m*K
            aa = np.sum(sim_sort*neighbor_ratings,axis=1) # m*1
            bb = np.sum(sim_sort,axis=1)+1e-10 # 1e-10保证分母不为0 m*1
            r_pred.append(aa/bb)

        return np.array(r_pred)
```

```
r_pred = prediction(train_mat=train_mat, sim_mat=sim_mat, K=K,
model='user_based')
```

对商品进行排序　获取top-n列表

```
n = 10
topn = get_topn(r_pred=r_pred, train_mat=train_mat, n=n)
print('user 0:')
print('top-n list:\n',topn[0])
```

user 0:
top-n list:
[533, 785, 1432, 984, 572, 1281, 1279, 410, 782, 806]

评估模型

$$MAE = \frac{1}{|R_{test}|} \sum_{(u,i) \in R_{test}} |r_{u,i} - \hat{r}_{u,i}|$$

$$RMSE = \sqrt{\frac{1}{|R_{test}|} \sum_{(u,i) \in R_{test}} (r_{u,i} - \hat{r}_{u,i})^2}$$

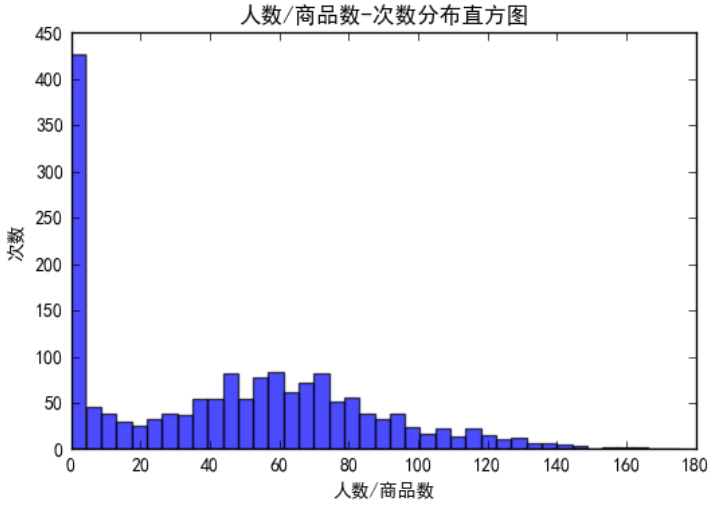
```
mae, rmse = mae_rmse(r_pred=r_pred, test_mat=test_mat)
print('mae:%.4f; rmse:%.4f'%(mae,rmse))
```

mae:0.8407; rmse:1.0796

2) 基于商品的近邻推荐

计算相似度

```
sim_mat = similarity_matrix(mat=train_mat.T, model='consine',
minimum_common_items=3)
neighbors = np.argsort(-np.array(sim_mat)) # 获取近邻
sim_sort = -1*np.sort(-np.array(sim_mat)) # 获取对应近邻的相似度

# 获取近邻数K
min_similarity = 0.5
K = get_K(sim_mat, min_similarity=min_similarity)
print('min_similarity:',min_similarity,'K:',K)
```



min_similarity: 0.5, K: 1

K 过小，所以手动设置，K=5

预测评分

```
r_pred = prediction(train_mat=train_mat, sim_mat=sim_mat, K=5,
model='item_based')
```

评估算法

```
n = 10
topn = get_topn(r_pred=r_pred, train_mat=train_mat, n=n)

# 评估算法
mae, rmse = mae_rmse(r_pred=r_pred, test_mat=test_mat)
print('mae:%.4f; rmse:%.4f'%(mae,rmse))
recall, precision = recall_precision(topn=topn, test_mat=test_mat)
print('recall:%.4f; precision:%.4f'%(recall,precision))
```

mae:0.8277; rmse:1.0722
recall:0.0092; precision:0.0194

3) 总结

	MAE	RMSE
基于用户的近邻推荐	0.8407	1.0796
基于商品的近邻推荐	0.8277	1.0722

从实验结果看，基于商品的近邻推荐和比基于用户的近邻推荐差不多，但前者比后者要好一丢丢。还有，它们在排序指标上均不理想。