分类技术---二分网络上的链路预测

实验环境：python3.7

**1、实验内容**

基于网络结构的链路预测算法被广泛的应用于信息推荐系统中。算法不考虑用户和产品的内容特征，把它们看成抽象的节点，利用用户对产品的选择关系构建二部图。为用户评估它从未关注过的产品，预测用户潜在的消费倾向。

具体步骤如下：

1.1，采用二分网络模型，对ml-1m文件夹中的“用户---电影”打分数据进行建模，考虑将用户信息、电影详细信息、以及打分分值作为该网络上的边、点的权重；

1.2，根据网络结构特征给出节点相似性度量指标；

1.3，基于相似性在二分网络上进行链路预测；

1.4，采用交叉验证的方法验证预测结果；

1.5，画出ROC曲线来度量预测方法的准确性。

**2、分析及设计**

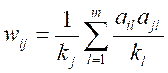
2.1、 采用二分网络模型，对ml-1m文件夹中的“用户---电影”打分数据进行建模；

用户对自己看过的电影打分1-5分，其中1分表示最不喜欢，5分表示最喜欢。假设分数大于3分的，表示用户喜欢这部电影，在二部图中构建一条从用户到该电影的连边。

考虑由m个用户n部电影构成的电影推荐系统。用户i对电影j打分超过3分，就在i和j之间连接一条边aji=1,否则aji=0。

2.2、 计算资源配额矩阵；

计算资源配额矩阵W中的元素wij表示产品j愿意分配给产品i的资源配额。假设一个用户选择过的商品j都有向该用户推荐其他产品i的能力。



kj表示产品j的度（被多少用户评价过），k*l*表示用户*l*的度（用户选择过多少产品）。

2.3、 对给定用户，按照其喜欢程度，对电影进行排名，进行电影推荐；

目标用户的资源分配矢量f。初始时，将他选择过的电影对应项资源设置为1，其他为0，得到初始n维0/1向量。则最终的资源分配矢量:

C:\Users\lenovo\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{37C9F79A-3F13-4332-8228-26778CC3B925}\{D8FC9997-A034-494B-98C4-304C4C0B1475}\ResourceMap\{3DD5853F-A346-422C-B174-D0CF27B386B4}

将用户所有没看过的电影按照C:\Users\lenovo\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{37C9F79A-3F13-4332-8228-26778CC3B925}\{D8FC9997-A034-494B-98C4-304C4C0B1475}\ResourceMap\{069ED0A6-9A3B-4D2F-BBE2-1CF72408A49E}中对应项的得分进行排序，推荐排序靠前的电影给该用户。

2.4、 算法预测准确性预测；

将二部图中的边随机分为两部分，期中90%归为训练集，10%归为测试集。

对给定用户i，假设其有Li个产品是未选择的，如果在测试集中用户i选择的电影j，而电影j依据向量C:\Users\lenovo\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{37C9F79A-3F13-4332-8228-26778CC3B925}\{D8FC9997-A034-494B-98C4-304C4C0B1475}\ResourceMap\{069ED0A6-9A3B-4D2F-BBE2-1CF72408A49E}被排在第Rij位，则计算其相对位置：



越精确的算法，给出的rij越小。对所有用户的rij求平均值C:\Users\lenovo\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{37C9F79A-3F13-4332-8228-26778CC3B925}\{D8FC9997-A034-494B-98C4-304C4C0B1475}\ResourceMap\{21D08B8B-EE2B-4FB6-BF98-E052C7FA71EE}来量化评价算法的精确度。

2.5、 画出ROC曲线来度量预测方法的准确性。

选取不同的算法阈值，计算相应的真阳性率（TP）以及假阳性率（FP），画出ROC曲线。

**3、详细实现**

3.1、数据导入，将data中的ratings.dat数据利用pandas的read\_table函数导入

rating\_names = ["user\_id","movie\_id","rating","timestamp"]

ratings = pd.read\_table("ml-1m/ratings.dat",sep="::",header=None,names=rating\_names)

3.2、按照为9:1分成测试集和训练集

mask = np.random.rand(len(ratings)) < 0.9

train\_ratings = ratings[mask]

test\_ratings = ratings[~mask]

3.3、将rating中的user\_id和movie\_id映射到对应的序号,即索引

users = ratings.user\_id.unique()

movies = ratings.movie\_id.unique()

uid2idx = {uid:k for k,uid in enumerate(users)}

mid2idx = {mid:k for k,mid in enumerate(movies)}

user\_size = len(users)

movie\_size = len(movies)

3.4、建立高分选择矩阵A和选择矩阵B，只要用户评价了某电影，就给矩阵B相应位置置1，只要用户给某电影打分不低于3分，就给A相应位置置1。

A = np.zeros((user\_size,movie\_size))

B = np.zeros((user\_size,movie\_size))

for \_,rating in train\_ratings.iterrows():

if(rating.rating>3): A[uid2idx[rating.user\_id],mid2idx[rating.movie\_id]] = 1

B[uid2idx[rating.user\_id],mid2idx[rating.movie\_id]] = 1

3.5、计算W矩阵,向量加速算法

k\_user = B.sum(axis=1)#表示某电影有多少人评价

k\_movie = B.sum(axis=0)#表示某人评价了多少部电影

W = np.zeros((movie\_size,movie\_size))

A1 = A/k\_user.reshape((-1,1))#除以用户的度

A1[np.isnan(A1)] = 0#太小的值发生了溢出，故置0

W = np.dot(A1.T,A)#向量乘法

W = W/k\_movie#除以产品的度

W[np.isnan(W)] = 0太小的值发生了溢出，故置0

3.6、利用W计算推荐分数矩阵F和推荐分数排名矩阵F\_sort，F[i][j]表示用户i下，电影j的推荐分数

F\_sort[i][j]表示用户i下，电影j的推荐分数在所有电影中的排名

选择A或B矩阵作为参数会得到不同的结果

F = np.dot(W,A.T).T

F\_sort\_index = np.argsort(F,axis=1)

F\_sort = np.zeros((user\_size, movie\_size))

for i in range(user\_size):

for j in range(movie\_size):

F\_sort[i,F\_sort\_index[i][j]] = movie\_size – j#此步将排名第movie\_size-j名的电影赋予他的排名

3.7、准备测试集矩阵，只有评分大于2时才认为用户喜欢该电影。

B\_test = np.zeros((user\_size,movie\_size))

for \_,rating in test\_ratings.iterrows():

if(rating.rating>3): B\_test[uid2idx[rating.user\_id], mid2idx[rating.movie\_id]] = 1

3.8、计算r的值及均值,评价算法的精确度。

L = movie\_size - k\_user# L:用户未选择的电影数

R = np.average(F\_sort\*B\_test,axis=1)/L

r\_aver = np.average(R)

print(r\_aver)

3.9、画ROC曲线，并计算积分的值AUC

TPR,FPR = [],[]

T\_ = np.sum(B\_test) #正样本

F\_ = np.sum(B\_test==False) #负样本

for threshold in np.arange(0,1,0.01):

F\_out = F\_sort < (movie\_size\*threshold)

F\_out = F\_out.astype(int)

TP = np.sum(B\_test \* F\_out)

FP = np.sum((1-B\_test) \* F\_out)

TPR.append(TP/T\_)

FPR.append(FP/F\_)

plt.plot(FPR,TPR)

plt.xlim(0,1)

plt.ylim(0,1)

plt.xlabel("FP")

plt.ylabel("TP")

plt.show()

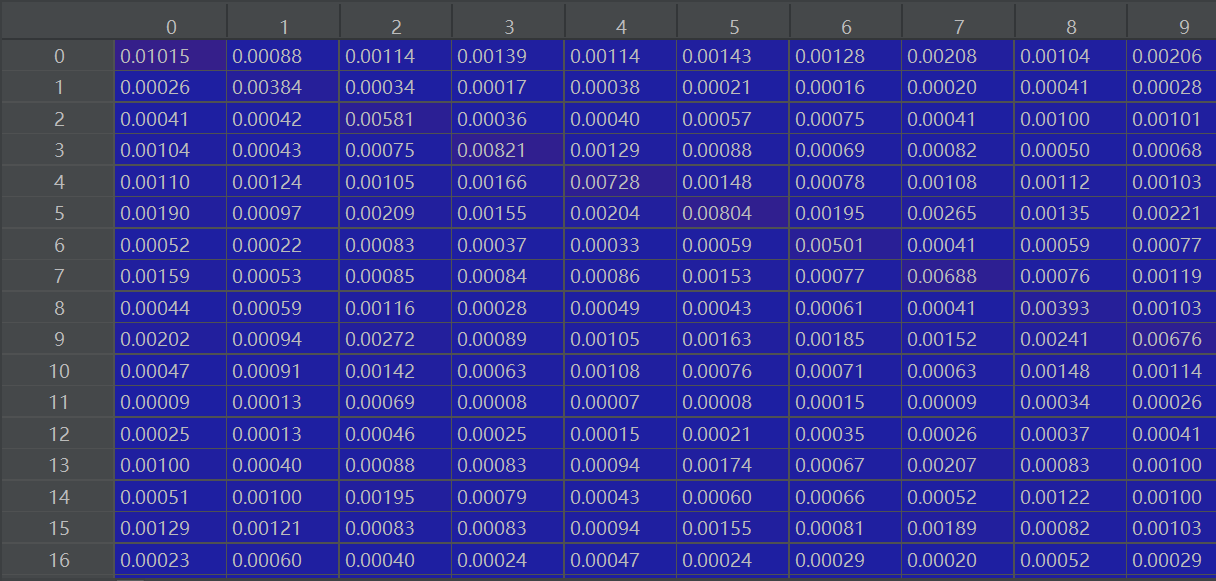
#ROC曲线的积分

AUC = np.sum([ 0.01\*tpr for tpr in TPR])

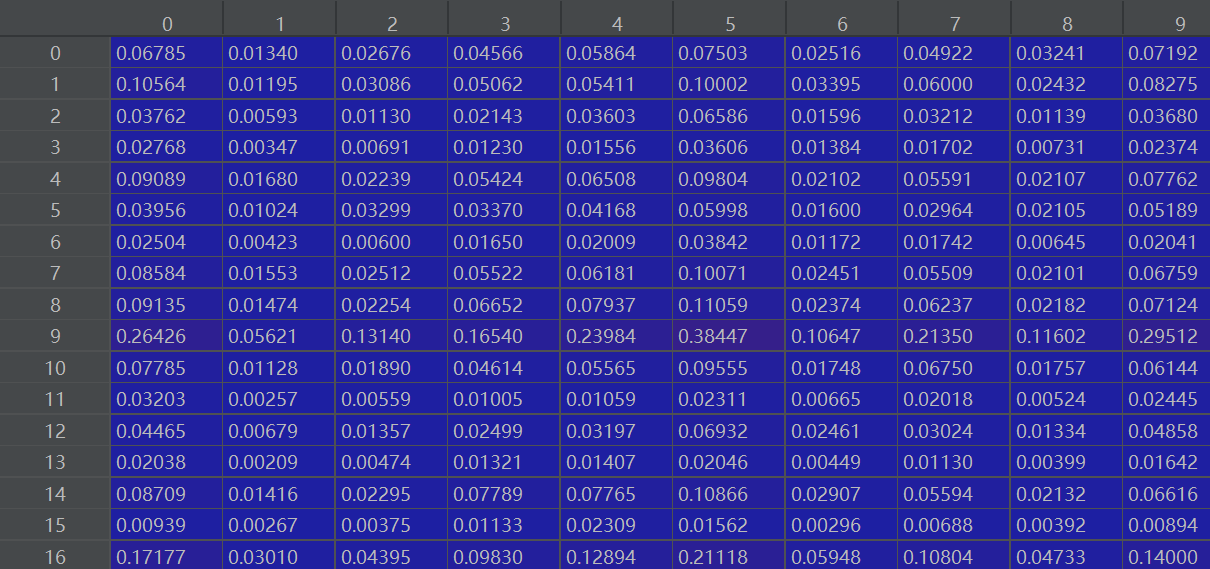
print(AUC)

**4、实验结果**

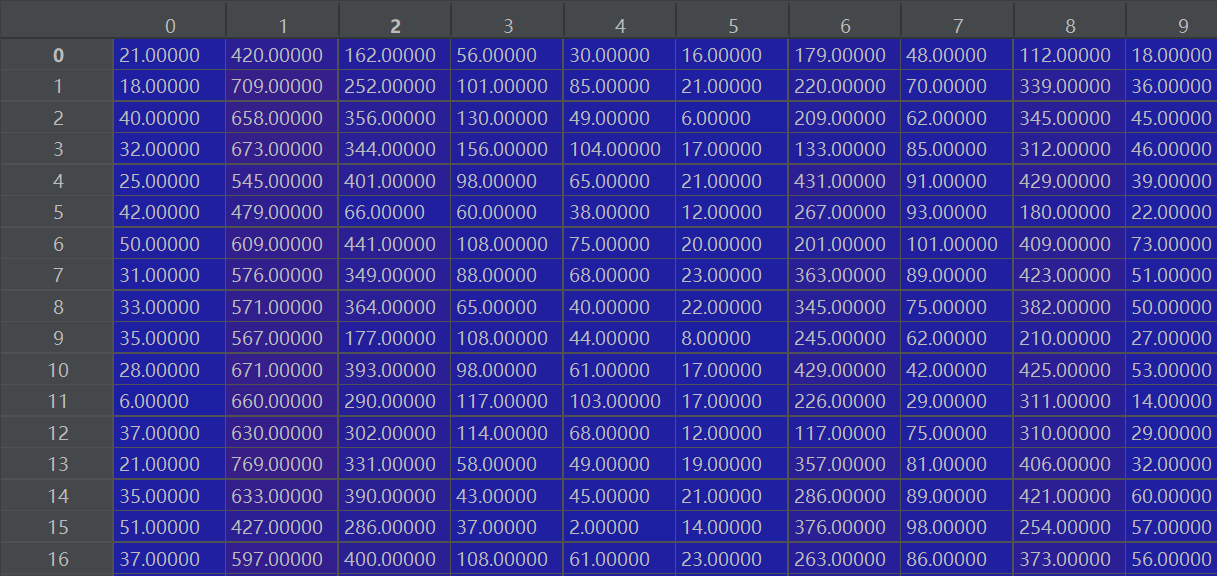
获得的权重矩阵W（部分值）：



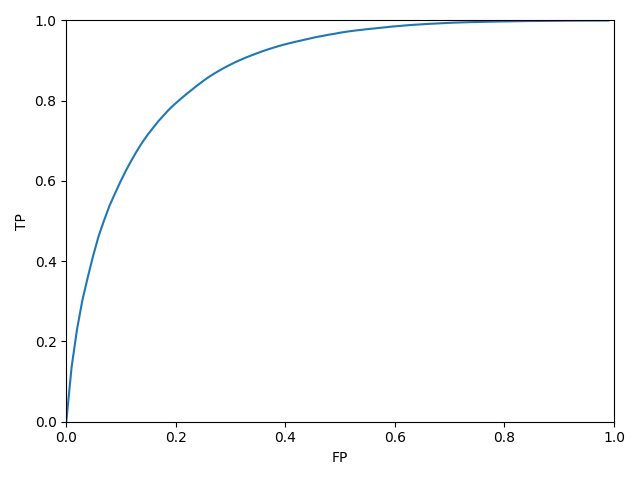
获得的推荐分数矩阵F（部分值）：



获得的推荐分数排名矩阵F\_sort（部分值）:



得到的ROC曲线如下图：



得到的AUC值为：0.8720229692755441（由于本实验的随机性，可能每次运行都稍微不同，但差别不会太大）

**5、心得**

本次实验我学习了数据挖掘的基础知识，提高了自己的动手能力，查阅资料与实践能力，更好的了解了数据挖掘知识的应用领域与方法，在实验过程中一开始没有考虑到本实验较大的数据量，直接用循环运算，结果花了很长时间都没出结果，后来改用矩阵乘法，成功得到结果。

做完之后考虑到用户选择了某电影不代表用户就喜欢某电影，所以我改用高分矩阵A训练权重矩阵，使得AUC值从0.8211提升到0.8719。

当然，本次实验只是最简单的推荐算法，学无止境，以后我会尝试学习更好的算法不断地提升自己。