**实验一 分类技术--二分网络上的链路预测**



班级：1403012

学号：14030120075

姓名：张国庆

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。MovieLens 是历史最悠久的推荐系统。它由美国 Minnesota 大学计算机科学与工程学院的 GroupLens 项目组创办，是一个非商业性质的、以研究为目的的实验性站点。MovieLens 主要使用 Collaborative Filtering 和 Association Rules 相结合的技术，向用户推荐他们感兴趣的电影。<https://grouplens.org/datasets/movielens/>

使用的数据集：ml-latest-small.zip中包括700个用户对9000部电影的100000条评价。

1. **实验分析及设计：**

对于链路预测实验，采用老师实验要求中的设计步骤来实现：

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。

1. 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）。

1. 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:



将用户所有没看过的电影按照中对应项的得分进行排序，推荐排序靠前的电影给该用户。

1. 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求平均值来量化评价算法的精确度。

1. 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。

1. **详细实现：**

对于上述所有实验步骤，都用c++语言进行了具体的实现，下面为实现步骤：

**3.1采用二分网络模型，对实验数据建模：**

3.1.1原始数据的读取

实验的原始数据存放在excel表格中，使用一些库可以方便的读取excel文件，不过excel文件中我们需要的数据文档只有两个，movie.csv和rating.csv。第一个是电影的列表，id，name，以及所属类别，不过我们需要的只有id和name。第二个是用户对电影的评分。包含用户id，电影id和评分。因为需要数据简单，所以无需调用库的复杂功能，我读取文件的方式是将csv文件转为txt文件，然后在程序中读取字符串。再进行相应的处理

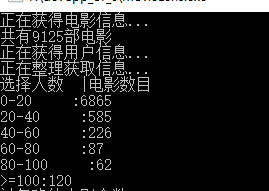
3.1.2二分图的组织形式

这个图不算稠密图，有9000部电影，100000个评分，700个用户而每个用户平均只评价了100多部电影，用图的邻接矩阵来表示，会造成空间的浪费，所以采用图的邻接表来实验对图的存储。具体实现中，采用两个map来存放。第一个key是用户id，value是放了用户选择的（评分3分及以上）电影的set，第二个key是电影id，value是放了选择该电影的用户的id。

**3.2计算资源配额矩阵**

3.2.1数据的筛选

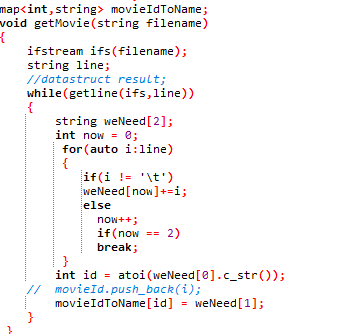
对于W矩阵的计算十分困难。我们观察W矩阵的计算办法，首先是wij的计算，需要m次，在本例中也就是700次。而电影共有9000个，所以计算次数达到9000\*9000\*700，也就是需要进行5亿次计算。加上每次计算一些别的处理，我的计算机需要耗时过久，无法完成实验。想了很久怎么去优化算法，也没有得到较好的结果。不过如果我们将选择次数打印出来，就可以很好的发现大部分数据并不是那么有用，如下图：



有6000多部电影，只有不足20个用户选择，而评价数大于100的电影只有120部。一个好的推荐算法应该推荐更好的电影。所以我对数据进行了筛选，好更快得到计算的结果。

综上，经过删选得到的图的代码如下：

读取数据用getMovie函数实现：



将数据变为图的代码用arrangeData函数实现：

void arrangeData(string filename,datastruct &movies,datastruct &users)

{

ifstream ifs(filename);

string uid,mid;

int num1=0,num2=0,num3=0,num4=0,num5=0,num6=6;

string score;

datastruct movie;

while(ifs>>uid>>mid)

{

ifs>>score;

if(score[0]>='3')

{

int u = atoi(uid.c\_str());

int m = atoi(mid.c\_str());

// users[u].insert(m);

movie[m].insert(u);

}

ifs>>score;

}

for(auto &i : movie)

{

int count = i.second.size();

if(count < 20)

++num1;

else if(count < 40)

++num2;

else if(count<60)

++num3;

else if(count<80)

++num4;

else if(count<100)

++num5;

else ++num6;

if(count>=50)

{

movies[i.first]=i.second;

movieId.push\_back(i.first);

}

}

cout<<"选择人数 |电影数目"<<endl;

cout<<"0-20 :"<<num1<<endl;

cout<<"20-40 :"<<num2<<endl;

cout<<"40-60 :"<<num3<<endl;

cout<<"60-80 :"<<num4<<endl;

cout<<"80-100 :"<<num5<<endl;

cout<<">=100:"<<num6<<endl;

cout<<"被忽略的电影个数:"<<movie.size() - movieId.size()<<endl;

int edgeCount = 0;

ifstream iff(filename);

while(iff>>uid>>mid)

{

iff>>score;

int u = atoi(uid.c\_str());

users[u];

if(score[0]>='3')

{

int m = atoi(mid.c\_str());

if(movies.find(m)!=movies.end())

{

users[u].insert(m);

++edgeCount;

}

}

iff>>score;

}

cout<<"共有"<<edgeCount<<"条边！"<<endl;

cout<<"计算矩阵需要的操作数:"<<movies.size()<<"\*"<<users.size()<<"\*"<<movies.size()<<endl;

}

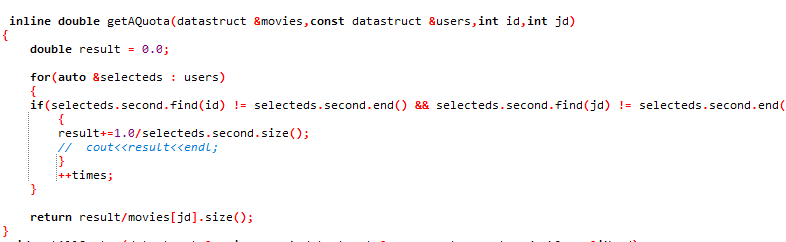
上述代码忽略选择数少于50的电影，并将信息打印出来：



* + 1. wij的计算

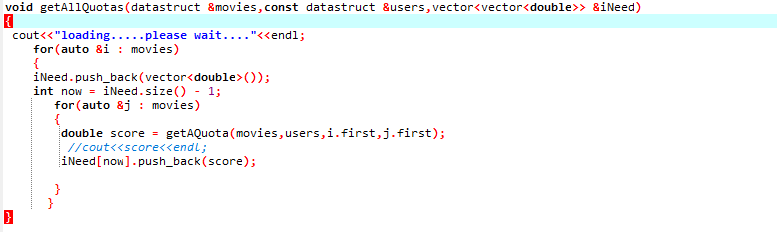
wij的计算较为简单，只有一点需要注意，就是我们这个函数并不会改变图的内容，所有使用引用（&）可以大大提高运行速度。

wij计算交给getAQuato函数：



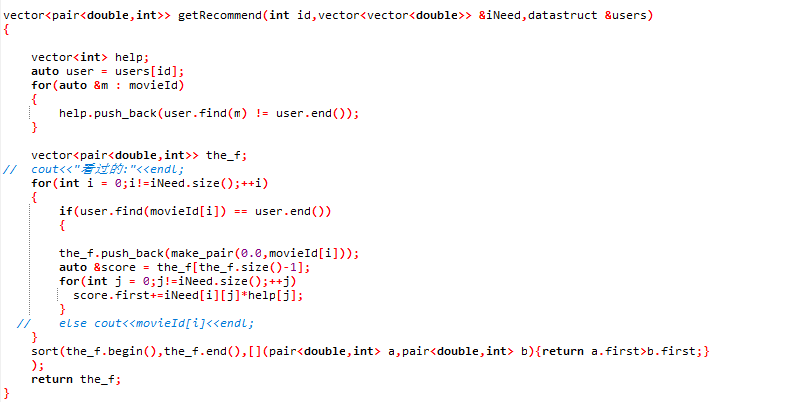
* + 1. W矩阵的计算

w矩阵就是wij矩阵的重复调用而已，也很简单，用getAllQuato函数实现：



* 1. **按照用户喜爱程度排名：**

f‘的计算较为简单，只是简单的矩阵相乘。只要告知用户id，就为其计算分数并排名。获取排名的功能交给getRecommend函数，算法如下：

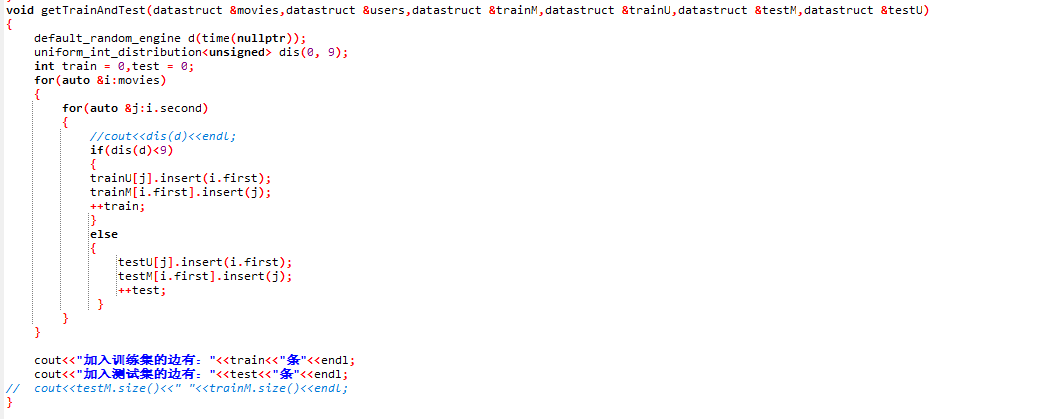


**3.4算法准确度预测：**

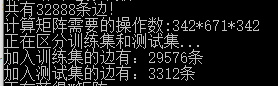
算法的准确度用测试集得分来衡量。

3.4.1 测试集和训练集的分离

对于题目要求，90%为训练集，10%分为测试集。使用c++11的random标准库来随机选择是属于训练集还是测试集。这个地方没什么棘手的，只需要注意一些细节。这个功能交给getTrainAndTest函数：

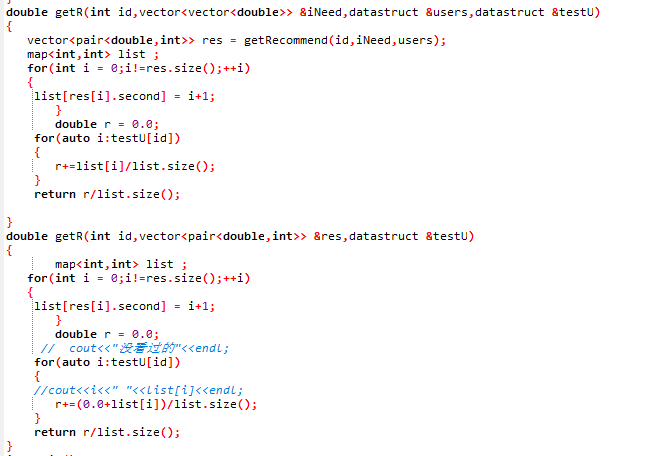


该函数会打印出加入训练集和测试集的边有那些：

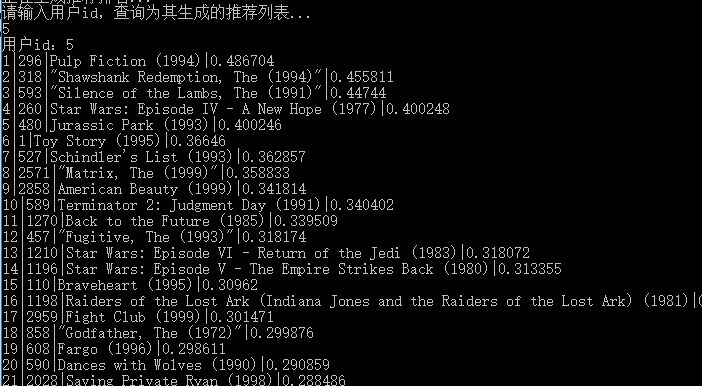


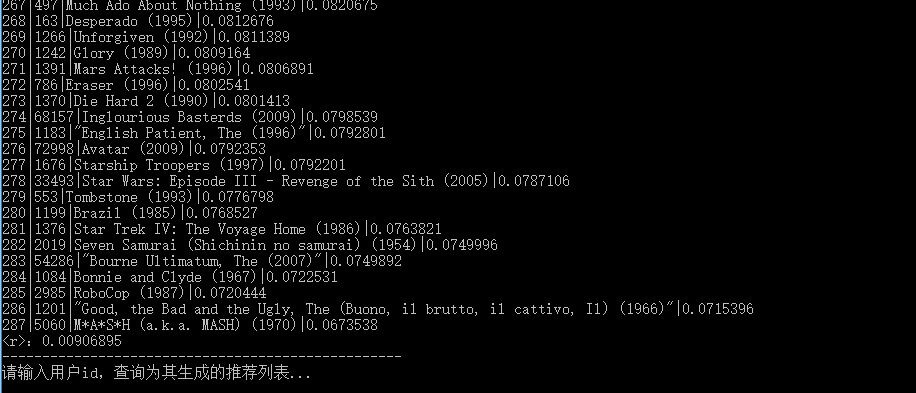
可以看到，比例近似9比1。

3.4.2 计算r。对于r的计算，我们需要快速得到测试集的电影在推荐列表中的排名。所以需要一个数据的重新组织，对推荐的电影建立map，方便迅速获取排名。计算R用了两个重载的函数，叫getR。第一个函数在排名时即时得到r值，第二个是后来计算，根据需求不同选择：

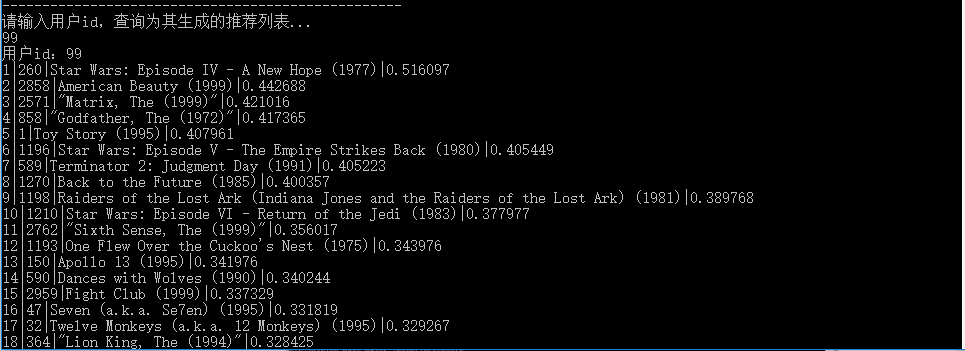


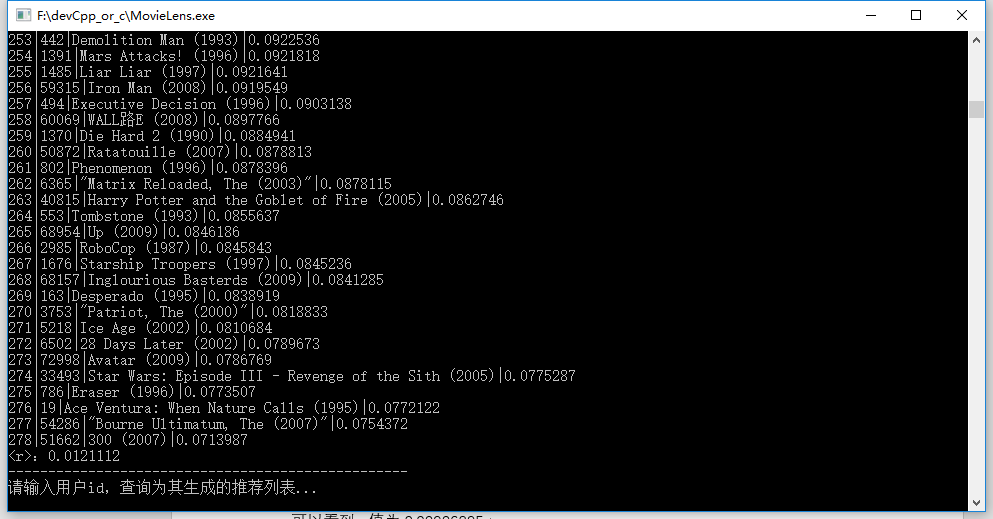
下面放上对几个用户的推荐电影表单：





可以看到r值为0.00906895

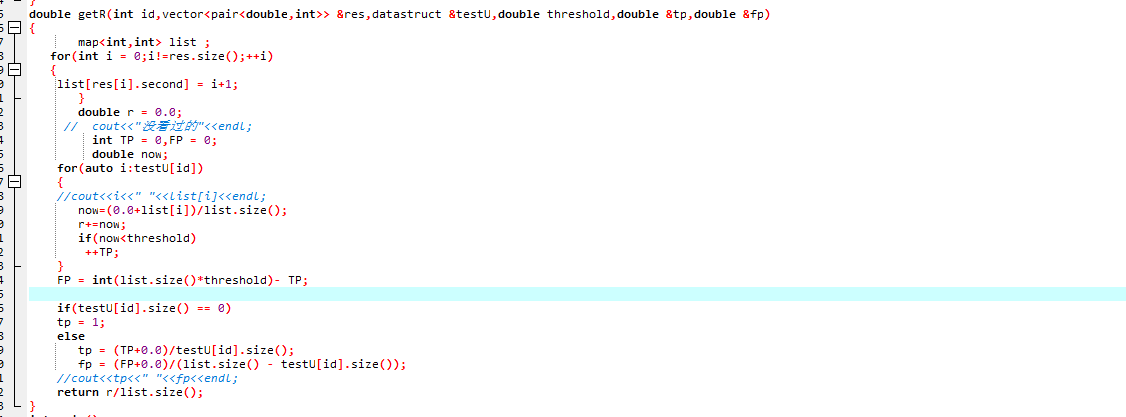




上述r值为0.0121112

**3.5画出ROC曲线来度量预测方法的准确性**

需要注意的是真假阳性率的计算，在代码中，对真假阳性率的计算放在了getR函数中完成，因此对getR函数做了改动，添加了两个引用，如下：

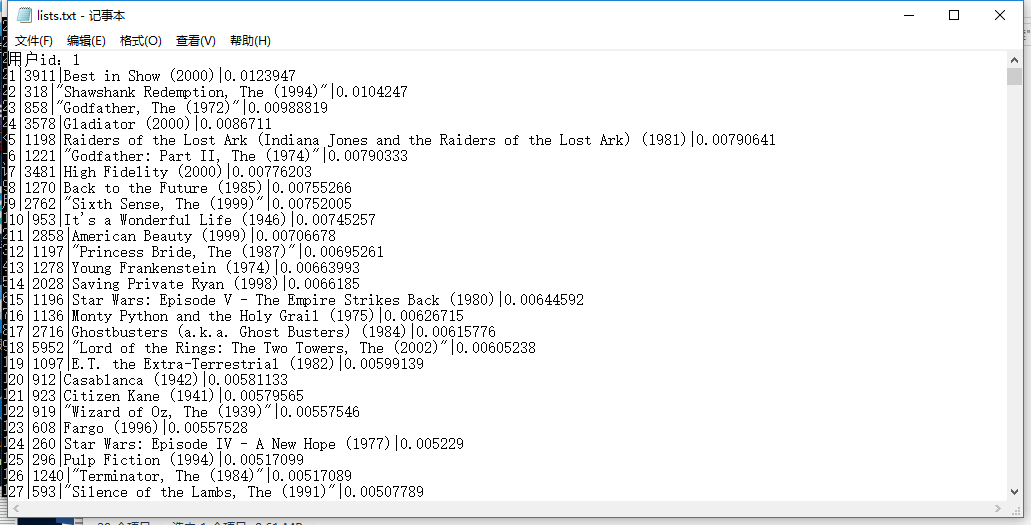


而ROC曲线用excel中的散点图完成。

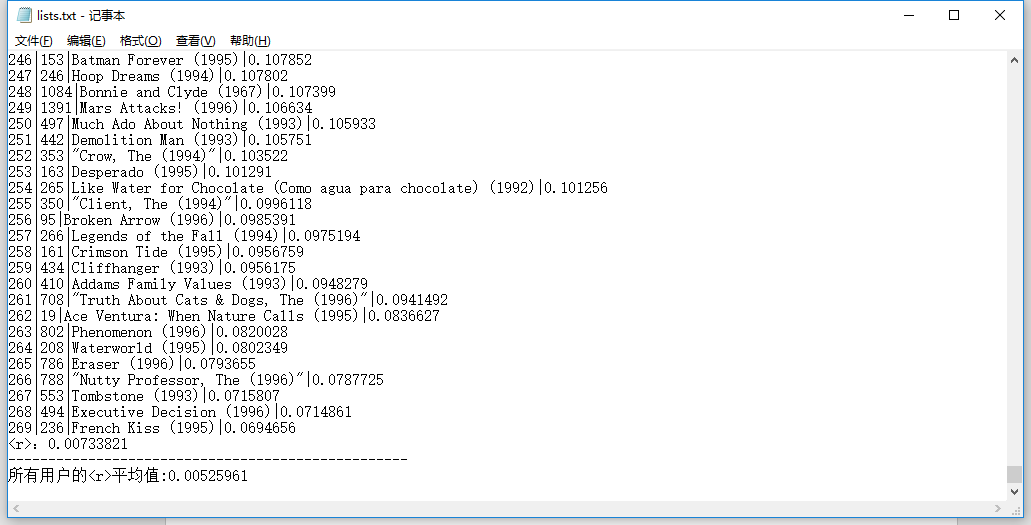
选取一个阈值r0作为分界点，比如r0=0.1，如果r小于0.1，则表明是真阳性。而这时候在前10%的不是用户选择的就是假阳性。用这个来定义真假阳性率。当r0逐渐增大时候，真假阳性率都趋向于100%。

1. **实验结果：**

我将所有用户的推荐表单已经输出到名叫lists.txt文件中：

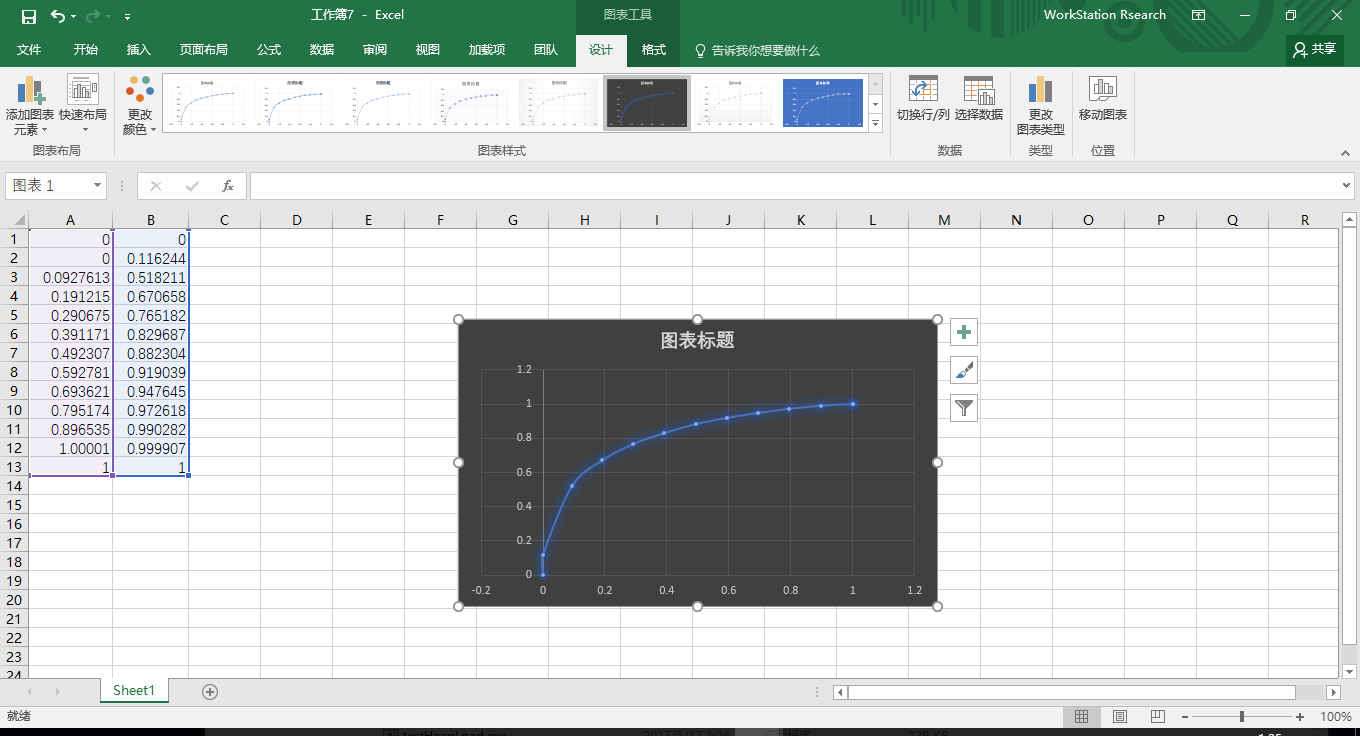


文件大小有8.6m，以附件的形式发送。r的平均值也在文件尾部可以找到：

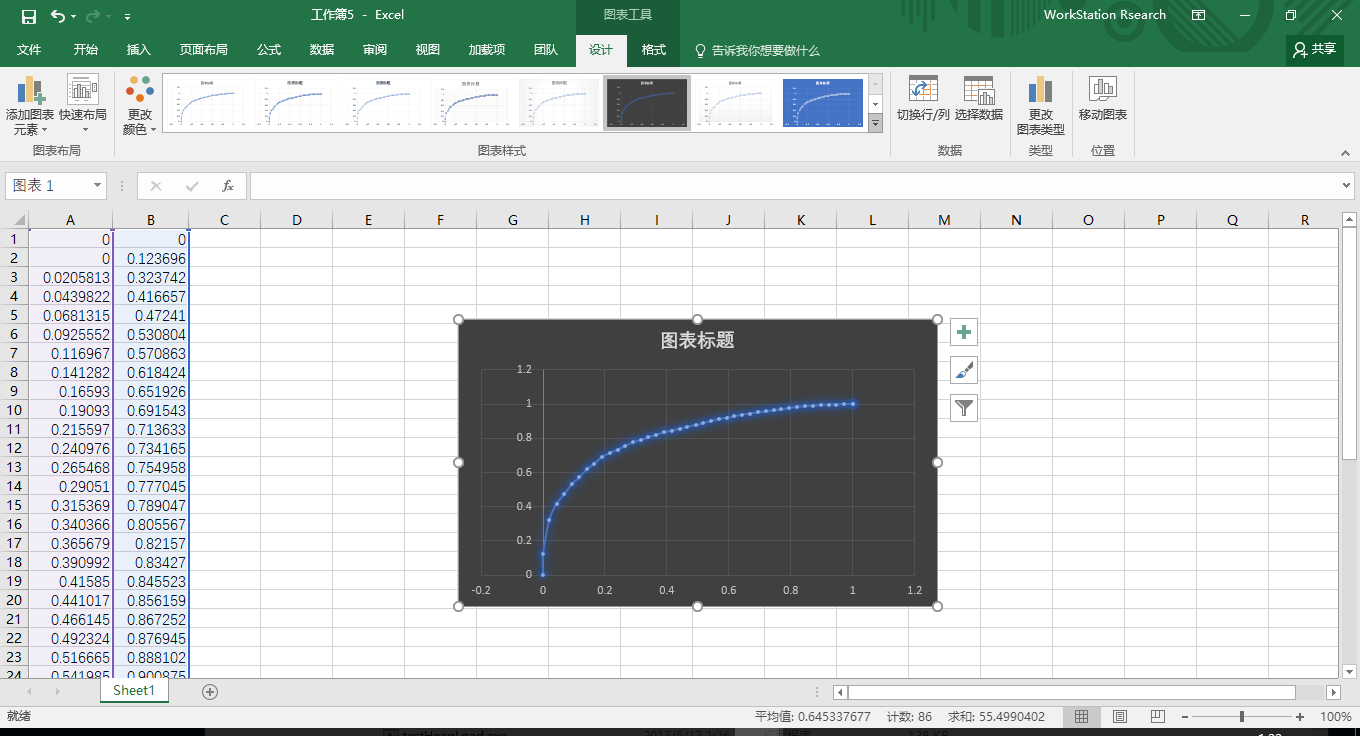


可以看到所有用户的r平均值为：0.00525961。

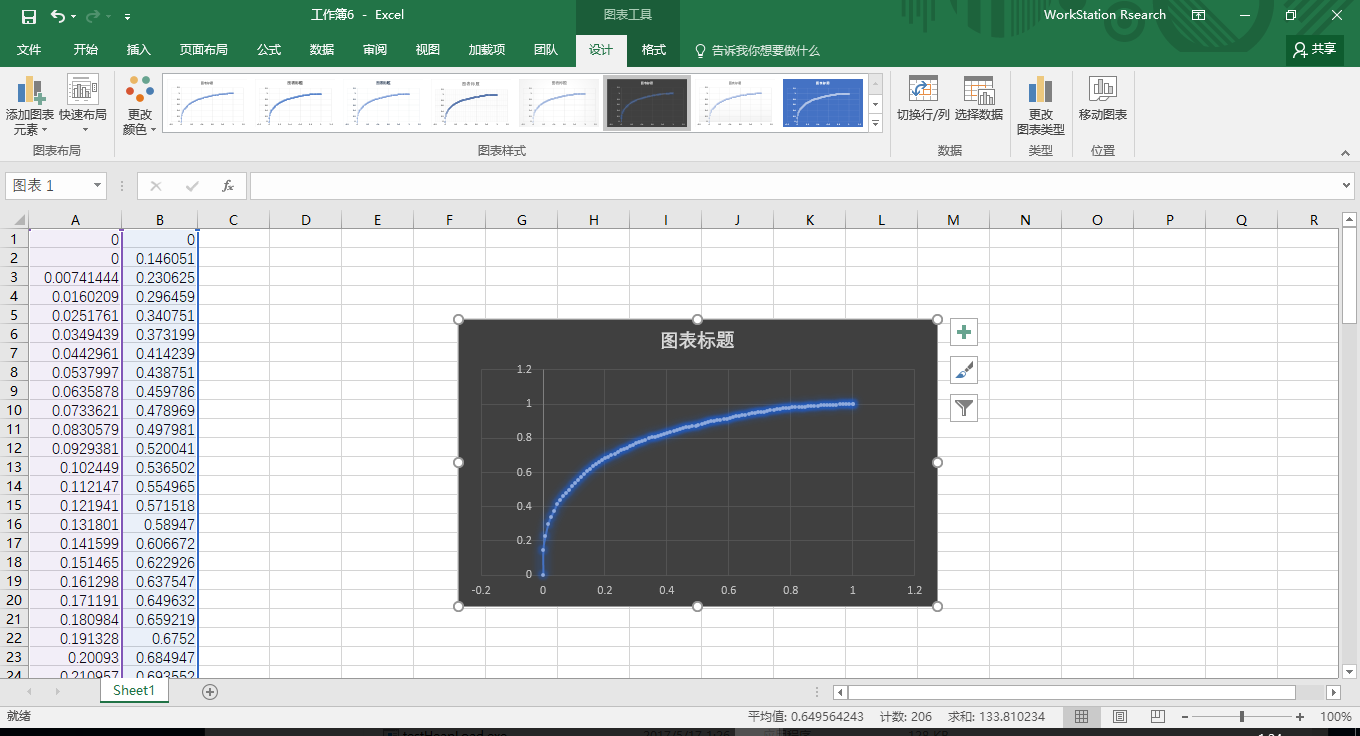
ROC曲线绘制时候，当阈值每次增幅为0.1时，也就是做10个点，roc曲线如下（TP为纵坐标，FP为横坐标，其中TP，FP为所有用户的平均值）：



这时候这张图还不是很直观，当增幅为0.025时候，图上有40个点：



当增幅为0.01时候，也就是图上会有100个点，ROC曲线如下：



可以这是的roc曲线已经非常明显，而真阳性率增幅速度更快，说明算法具有较好的准确率。

所有获得的数据打包到压缩包中，其中lists是所有用户的推荐表单以及r值，r0\_0.1为增幅为0.1时候的表格，TP，FP值以及图像。相应的有r0\_0.025与r0\_0.01。

1. **心得体会：**

这次数据挖掘实验让我对链路预测算法有了基本的了解。我明白了数学在数据挖掘中的重要性，并且深深认识到了自己知识的浅薄。这次实验同样激起了我对数据挖掘的兴趣。我明白了一些推荐算法的基本原理以及好的推荐算法的重要性。以后的日子里会更加努力学习，而不会因为课程的结束而停止。