中山大学数据科学与计算机学院
移动信息工程专业-数据挖掘
本科生实验报告

（2018-2019 学年春季学期）

课程名称：数据挖掘

[redacted]			
[redacted]			

一、 实验题目

推荐系统

二、 实验内容

1. 算法原理

（1）IBCF

算法原理：寻找与用户喜欢的商品最为相似的商品，通过计算用户已经给出的分数，对未评分的相似商品进行评分预测。

算法流程：①用皮尔逊相关系数来衡量商品的相似度：

$$sim(i,j)=\frac{\sum_{u\in U}(R_{u,i}-\overline{R_i})(R_{u,j}-\overline{R_j})}{\sqrt{\sum_{u\in U}(R_{u,i}-\overline{R_i})^2}\sqrt{\sum_{u\in U}(R_{u,j}-\overline{R_j})^2}}$$

② 预测评分： s_{ij}：商品 i 和商品 j 的相似度；r_{xj}：用户 x 对商品 j 的评分；N（i:x）:在 x 评分的商品中和商品 i 相似的商品集。

$$r_{xi}=\frac{\sum_{j\in N(i;x)}s_{ij}\cdot r_{xj}}{\sum_{j\in N(i;x)}s_{ij}}$$

（2）slopeone

算法原理：寻找用户点评过的商品之间的评分偏差，通过偏差推测出用户对未评分商品的评分。

算法流程：① 计算商品之间的评分偏差；

$$dev_{j,i}=\sum_{u\in U_{j,i}}\frac{R_{u,j}-R_{u,i}}{card(U_{j,i})}$$

U_{ji} 为共同点评过商品 i 和商品 j 的用户，card 为集合中元素数目。

② 根据物品间的评分偏差和用户历史评分，预测用户对未评分商品的评分。

$$R_{u,j}=\frac{1}{card(S(u)-\{j\})}\sum_{i\in S(u)-\{j\}}(dev_{j,i}+R_{u,i})$$

,s(u)为用户 u 评价过的商品集合。

（3）MF

算法原理：通过寻找用户因子矩阵和商品因子矩阵，从而得到一个评分矩阵的过程，即假设 user 和 item 有着共同现在的 K 个特征，组成矩阵 P 和 Q， 这两个矩阵是由原矩阵分解得到，由此可以得到原矩阵的估计。

$$R_{m\times n} \approx P_{m\times k} \times Q_{k\times n} = \hat{R}_{m\times n}$$

损失函数：使用原始的评分矩阵 Rmxn 与重新构建的评分矩阵$\widehat{R_{mxn}}$之间的误差的平方作为损失函数。即

$$e_{i,j}^2 = (r_{i,j} - \hat{r}_{i,j})^2 = \left(r_{i,j} - \sum_{k=1}^{K} p_{i,k}q_{k,j}\right)^2$$

目的：要求所有的评分选项的损失之和最小。

算法流程：① 随机初始化 P,Q；② 利用梯度下降法：《1》求解损失函数的负梯度：

$$\frac{\partial}{\partial p_{i,k}} e_{i,j}^2 = -2\left(r_{i,j} - \sum_{k=1}^{K} p_{i,k}q_{k,j}\right) q_{k,j} = -2e_{i,j}q_{k,j}$$

$$\frac{\partial}{\partial q_{k,j}} e_{i,j}^2 = -2\left(r_{i,j} - \sum_{k=1}^{K} p_{i,k}q_{k,j}\right) p_{i,k} = -2e_{i,j}p_{i,k}$$

《2》根据负梯度方向更新变量：

$$p_{i,k}' = p_{i,k} - \alpha\frac{\partial}{\partial p_{i,k}} e_{i,j}^2 = p_{i,k} + 2\alpha e_{i,j}q_{k,j}$$

$$q_{k,j}' = q_{k,j} - \alpha\frac{\partial}{\partial q_{k,j}} e_{i,j}^2 = q_{k,j} + 2\alpha e_{i,j}p_{i,k}$$

③ 重复②直到算法收敛。

2. 关键代码截图（带注释）

（1）IBCF

① 读取数据集，获取用户个数和商品个数：

```
data = load('u.data');
data = data(:,1:3);
```

```
[dif1,r1] = unique(data(:,1),'rows');
user = size(dif1,1);
[dif2,r2] = unique(data(:,2),'rows');
item = size(dif2,1);
```

② 构建用户-商品矩阵：

```
score = zeros(user,item);
for i=1:user
    for j=1:item
        score(i,j)=-1;
    end
end

for i=1:size(data,1)
    score(data(i,1),data(i,2)) = data(i,3);
end
```

③ 抽出一部分已知评分的商品作为验证集：

```
%抽出一部分已知评分的商品作为验证集
test = zeros(user,item);
for i=1:user
    for j=1:item
        test(i,j)=-1;
    end
end
```

```
for i=1:user
    nor = find(score(i,:)~=-1);
    sn = size(nor,2);
    if sn<4
        break;
    else
        sn = floor(sn/4);
    end
    for j=1:sn
        test(i,nor(j)) = score(i,nor(j));
        score(i,nor(j)) = -1;
    end
end
```

④ 用 pearson 相关系数求商品相似性：

《1》计算均值：

```
%用pearson相关系数求商品相似性
%首先计算均值
average = zeros(item,1);
for i=1:item
    r = find(score(:,i)~=-1);
    sum = 0;
    for j=1:size(r,1)
        sum = sum + score(r(j),i);
    end
    average(i,1) = sum/size(r,1);
end
```

《2》 计算相似性矩阵：

```
for i=1:item-1
    %给item i评分了的用户
    ri = find(score(:,i)~=-1);

    for j=i+1:item
        %给item j评分了的用户
        rj = find(score(:,j)~=-1);

        %找到给i,j都评分了的用户
        u = intersect(ri,rj);
        num = size(u);
        if num(1)==0 || num(2)==0
            sim(i,j)=0;
        else
            up=0;
            down1=0;
            down2=0;
            for k=1:num(1)
                up = up + (score(u(k),i)-average(i))*(score(u(k),j)-average(j));
                down1 = down1 + (score(u(k),i)-average(i))*(score(u(k),i)-average(i));
                down2 = down2 + (score(u(k),j)-average(j))*(score(u(k),j)-average(j));
            end
            sim(i,j)= up/(sqrt(down1)*sqrt(down2));
            sim(j,i) = sim(i,j);
        end
    end
end
```

⑤ 根据用户已经评分的商品的分数来预测未评分的相似商品的分数：

```
for i=1:user
    pos = find(score(i,:)~=-1);
    for j=1:item
        if ~any(pos==j)
            u=0;
            d=0;
            for k=1:size(pos,2)
                u = u + sim(pos(k),j)*score(i,pos(k));
                d = d + sim(pos(k),j);
            end
            score(i,j) = u/d;
        end
    end
end
```

⑥ 根据验证集来计算 RMSE：

```
RMSE = calRMSE(score,test,user,item);
disp(['RMSE = ',num2str(RMSE)]);
```

```
function RMSE = calRMSE(train,test,user,item)
num=0;
err = 0;
for i=1:user
    r = find(test(i,:)~=-1);
    nr = size(r,2);
    num = num + nr;
    for j=1:nr
        err = err + (test(i,r(j)-train(i,r(j))))*(test(i,r(j)-train(i,r(j))));
    end
end
RMSE = sqrt(err/num);
end
```

根据验证集来计算准确率：（评分分差的平方小于 1 则认为评分准确）

```
function acc = Accurancy(train,test,user)
num=0;
right = 0;
for i=1:user
    r = find(test(i,:)~=-1);
    nr = size(r,2);
    num = num + nr;
    for j=1:nr
        err = (test(i,r(j))-train(i,r(j)))*(test(i,r(j))-train(i,r(j)));
        if err < 1
            right = right+1;
        end
    end
end
acc = right/num;
end
```

（2）slopeone

① 计算商品之间的评分偏差：

```
for i=1:item-1
    %给item i评分了的用户
    ri = find(score(:,i)~=-1);

    for j=i+1:item
        %给item j评分了的用户
        rj = find(score(:,j)~=-1);

        %找到给i,j都评分了的用户
        u = intersect(ri,rj);
        num = size(u);
        if num(1)==0 || num(2)==0
            dev(i,j)=0;
        else
            up=0;
            down = num(1);
            for k=1:num(1)
                up = up + score(u(k),i)-score(u(k),j);
            end
            dev(i,j) = up/down;
            dev(j,i) = dev(i,j);
        end
    end
end
```

② 根据物品间的评分偏差和用户历史评分，预测用户对未评分商品的评分：

```
for i=1:user
    pos = find(score(i,:)~=-1);
    pre = find(score(i,:)==-1);
    for j=1:size(pre,2)
        an = 0;
        for k=1:size(pos,2)
            an = an + dev(pre(j),pos(k))+ score(i,pos(k));
        end
        score(i,pre(j)) = an/size(pos,2);
    end
end
```

（3）MF

① 随机初始化 P,Q：

```
k = 10;
%随机初始化P,Q
P = abs(rand(user,k));
Q = abs(rand(k,item));
iterator = 20; %迭代次数
itnum = 1;
a = 0.01;
```

② 根据损失函数更新 P,Q：

```
before = 100;
while itnum <= iterator
    %计算损失函数
    [e,alle] = calE(score,P,Q,user,item,k);
    if alle < before
        before = alle;
    end
    %更新P,Q
    [P,Q]= updatePQ(P,Q,a,e,user,item,k);
    itnum = itnum+1
end
```

《1》 计算损失函数：

```
function [E,allE] = calE(R,P,Q,n,m,nk)
    allE = 0;
    E = zeros(n,m);
    for i=1:n
        for j=1:m
            sumk=0;
            for k=1:nk
                sumk = sumk+P(i,k)*Q(k,j);
            end
            if R(i,j)~=-1
                E(i,j) = (R(i,j) - sumk)*(R(i,j) - sumk);
                allE = allE + E(i,j);
            else
                E(i,j) = 0;
            end
        end
```

《2》更新 P,Q：

```
function [P,Q] = updatePQ(P,Q,a,E,n,m,nk)
    for i=1:n
        for j=1:m
            for k=1:nk
                p = P(i,k);
                q = Q(k,j);
                P(i,k) = P(i,k)+2*a*E(i,j)*q;
                Q(k,j) = Q(k,j)+2*a*E(i,j)*p;
            end
        end
    end
end
```

③ 计算 RMSE 和准确率：

```
train = cross(P,Q);
RMSE = calRMSE(train,test,user,item);
disp(['RMSE = ',num2str(RMSE)]);

acc = Accurancy(train,test,user);
disp(['准确率 = ',num2str(acc)]);
```

三、 实验结果及分析

（1）IBCF

ml-100k:

```
RMSE = 82.362
准确率 = 0.56932
```

ml-1m:

```
RMSE = 61.231
准确率 = 0.4
```

（2）Slopeone

ml-100k:

```
RMSE = 1.0181
准确率 = 0.66499
```

ml-1m:

```
RMSE = 1.101
准确率 = 0.6
```

（3）MF

MF 的准确率与随机初始化的 P、Q 矩阵，以及迭代次数有关。

ml-100k:

```
RMSE = 1.1693
准确率 = 0.6
```

ml-1m:

```
RMSE = 1.2894
准确率 = 0.8
```

```
RMSE = 1.1309
准确率 = 0.6
```