本科生实验报告

（2017-2018 学年秋季学期）

一、 实验题目

推荐系统

二、 算法解析

1. 基于商品的协同过滤（IBCF）

IBCF（Item-Based Collaborative Filtering）算法是通过不同 item 的评分来评测 item 之间的相似性，基于 item 之间的相似性给用户推荐与其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。

算法步骤：

IBCF 算法可以分为构建用户评分矩阵、寻找相似用户集以及根据相似用户的兴趣来产生预测推荐这 3 个步骤。

（1）构建用户评分矩阵

假设推荐系统有 m 个用户和 n 个项目,则这个系统可以表示为一个 m×n 矩阵,评分矩阵 R 中的每一项表示用户 i 对项目 j 的评分。定义用户 i 对项目 j 的评分为 Ri,j,如果用户 i 没有对项目 j 进行评分,则可用 Ri,j= 0 或其他特殊字符来表示。协同过滤问题可以看成是预测 user-item 矩阵中的缺失值问题。

（2）寻找相似用户集

这个步骤是基于评分矩阵 R 来计算各个 item 的相似度，传统的相似度量方法主要有如下 3 种，在进行相似度度量时，对于两个 item i 和 j，仅考虑对这两个商品都进行评分的用户。

①余弦相似度

$$\mathrm{sim}(i,j)=\frac{\vec{i}\cdot\vec{j}}{\|i\|_2 * \|j\|_2}$$

②皮尔逊相关系数

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

③余弦适应性相似度

在余弦相似性度量方法中没有考虑不同用户的评分尺度问题,修正的余弦相似性度量方

法通过减去用户对项目的平均评分改善了该缺陷。

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

（3）预测推荐

用过对用户 u 已打分的物品的分数进行加权求和，权值为各个物品与物品 i 的相似度，然后对所有物品相似度的和求平均，计算得到用户 u 对物品 i 打分，公式如下：

$$\mathrm{r}_{ui}=\frac{\sum_{j\in N(i:u)}s_{ij}\cdot r_{uj}}{\sum_{j\in N(i:u)}s_{ij}}$$

其中 N(i;u)为用户 u 打分过的与商品 i 相似的集合，通过设置阈值或者取 topK 个获得相似邻居集合。

2. Slope One

Slope One 是一种协同过滤算法，本质为一元线性模型。Slope One 算法的原理是寻找用户点评过的商品之间的评分偏差，之后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

对于每个商品 i 和 j，根据同一用户对产品 i 和 j 的评分偏差，求平均值，得到商品的评分偏差。仅考虑对商品 i 和 j 都进行评分的用户为$U_{j,i}$，card 为集合元素数量。

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

（2）预测评分

根据商品之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：

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

3. 矩阵分解

用户-商品的评分矩阵非常稀疏，矩阵分解则能挖掘用户的潜在因子和项目的潜在因子。通过矩阵分解来得到用户因子矩阵和商品因子矩阵。用户因子矩阵和商品因子矩阵相乘能够得到近似评分矩阵，从而估计缺失值。

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

其中 P 为用户因子矩阵，Q 为商品因子矩阵。

使用原评分矩阵 $R_{m\times n}$ 与重新构建的评分矩阵 $\hat{R}_{m\times n}$ 之间的误差的平方作为损失函数：

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2$$

需要所有已知评分的损失最小，即：

$$\min loss = \sum_{r_{i,j}\neq -} e_{i,j}^2$$

算法步骤：

(1) 随机初始化用户矩阵 P 和商品矩阵 Q

(2) 采用梯度下降法更新矩阵 P 和 Q

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

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

仅对已知评分的 Rij 的位置进行计算更新，按照上式迭代更新 P，Q 直至 loss 收敛。

R'=P*Q;为预测的评分矩阵。

三、 代码实现

1. 基于商品的协同过滤（IBCF）

（1）相似度计算

①余弦相似度

```
%para=1表示余弦相似度；para=2表示皮尔逊相似度；para=1表示余弦适应性相似度；
[userNum,itemNum]=size(R);
sim=zeros(itemNum,itemNum);%item-item相似度矩阵
if para==1%余弦相似度
    for i=1:itemNum
        for j=1:itemNum
            tem_i=[];tem_j=[];tem_num=0;
            for k=1:userNum
                if (R(k,i)>0&&R(k,j)>0)
                    tem_i=[tem_i,R(k,i)];
                    tem_j=[tem_j,R(k,j)];
                    tem_num=tem_num+1;
                end
            end
            if tem_num~=0
                sim(i,j)=(tem_i*tem_j')/(norm(tem_i)*norm(tem_j));
            else
                sim(i,j)=0;
            end
        end
    end
```

②皮尔逊相关系数

```
elseif para==2%皮尔逊相关系数
    avgR_item=zeros(itemNum,1);
    for i=1:itemNum
        %求每个item的平均得分
        avgR_item(i)=sum(R(:,i))/sum(R(:,i)~=0);
    end
    for i=1:itemNum
        for j=1:itemNum
            tem1=0;tem2=0;tem3=0;
            for u=1:userNum
                if (R(u,i)>0&&R(u,j)>0)
                    tem1=tem1+(R(u,i)-avgR_item(i))*(R(u,j)-avgR_item(j));
                    tem2=tem2+(R(u,i)-avgR_item(i)).^2;
                    tem3=tem3+(R(u,j)-avgR_item(j)).^2;
                end
            end
            if tem2==0||tem3==0
                sim(i,j)=0;
            else
                sim(i,j)=tem1/(sqrt(tem2)*sqrt(tem3));
            end
        end
    end
```

（2）预测评分

```
for i=1:testNum
    user=test_M(i,1);
    item=test_M(i,2);
    tem_1=0;tem_2=0;
    for j=1:itemNum
        if sim(item,j)>0&&train_R(user,j)>0
            tem_1=tem_1+sim(item,j)*train_R(user,j);
            tem_2=tem_2+sim(item,j);
        end
    end
    if tem_2~=0
        predict(i)=round(tem_1/tem_2);
    else
        predict(i)=0;
    end
end
```

2. Slope One

（1）计算商品之间的评分偏差

```
dev=zeros(itemNum,itemNum);
for i=1:itemNum
    for j=1:itemNum
        tem_num=0;
        tem_sum=0;
        for k=1:userNum
            if (train_R(k,i)>0&&train_R(k,j)>0)
                tem_sum=tem_sum+train_R(k,i)-train_R(k,j);
                tem_num=tem_num+1;
            end
        end
        if tem_sum~=0
            dev(i,j)=tem_sum/tem_num;
        else
            dev(i,j)=0;
        end
    end
end
```

（2）预测评分

```
for i=1:testNum
    user=test_M(i,1);
    item=test_M(i,2);
    tem_sum=0;tem_num=0;
    for j=1:itemNum
        if j~=item
            if train_R(user,j)>0
                tem_sum=tem_sum+dev(item,j)+train_R(user,j);
                tem_num=tem_num+1;
            end
        end
    end
    if tem_num~=0
        predict(i)=round(tem_sum/tem_num);%取整
    else
        predict(i)=0;
    end
    if predict(i)>5 %设置上限
        predict(i)=5;
    end
end
```

3. 矩阵分解

（1）计算每次迭代的 loss

```
for t=1:maxiter%迭代更新
    R2=P*Q;
    E=train_R-R2;
    for i=1:userNum
        for j=1:itemNum
            if(train_R(i,j)>0)
                loss(t)=loss(t)+E(i,j).^2;%计算loss
            end
        end
    end
    if(loss(t)>err_last)%若loss开始增加则停止迭代
        break;
    else
        err_last=loss(t);
    end
    fprintf('loss\t%f\n',loss(t));
```

（2）梯度下降更新 P,Q

```
for i=1:userNum
    for j=1:itemNum
        if(train_R(i,j)>0)
            for k=1:K
                delP=-2*E(i,j)*Q(k,j);
                delQ=-2*E(i,j)*P(i,k);
                P(i,k)=P(i,k)-a*delP;
                Q(k,j)=Q(k,j)-a*delQ;
            end
        end
    end
end
```

四、 效果分析

1、实验设置

将数据集进行划分，80%训练集，20%测试集。使用训练集构建用户-商品的评分矩阵，使用 IBCF、SlopeOne、矩阵分解方法得到预测的评分矩阵，与测试集的结果进行比较。

实验采用 MovieLens 不同大小的数据集，评分范围为 1-5.

实验的评测指标包括两种方式：基于评分的评测，使用 RMSE 和 MAE；使用阈值划分喜欢/不喜欢，使用 precision、recall 和 F1 进行评测。

（1）基于评分的评测

根据算法得到 1-5 的评分后，计算 RMSE 和 MAE，越小代表实验效果越好。

RMSE 是均方根误差，代表观测值与真值偏差的平方和与观测次数 m 比值的平方根。

$$RMSE=\sqrt{\frac{\sum_{i=1}^{N}(p_i-q_i)^2}{N}}$$

MAE 是平均绝对误差，是绝对误差的平均值。

$$MAE=\frac{\sum|p_i-q_i|}{N}$$

（2）使用阈值划分喜欢/不喜欢

在通过算法得到 1-5 的评分之后，以 3 为阈值，大于等于 3 代表喜欢，否则为不喜欢。计算 Precision、Recall 和 F1，越大实验效果越好。

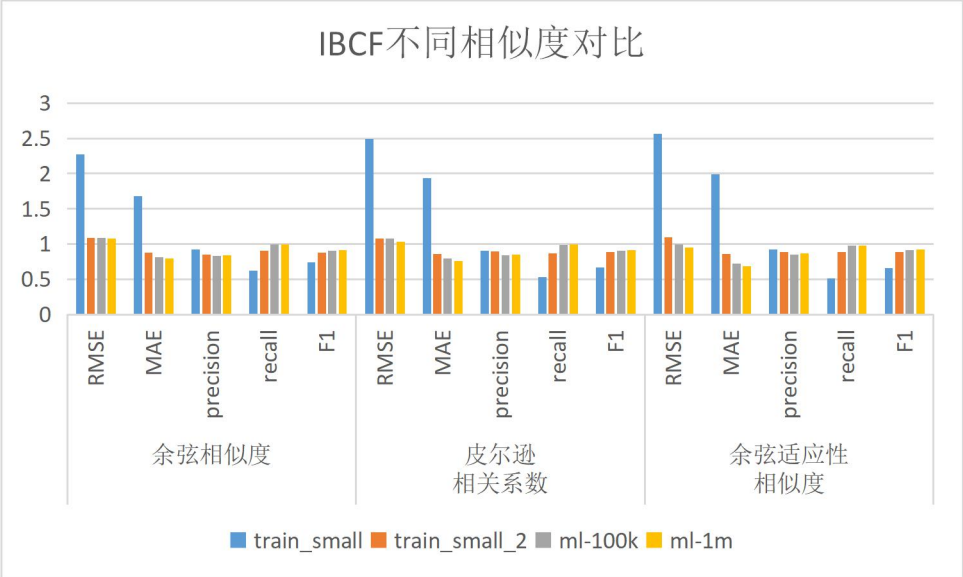
Precision = (预测为 1 且正确预测的样本数)/(所有预测为 1 的样本数)

Recall = (预测为 1 且正确预测的样本数)/(所有真实情况为 1 的样本数)

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

2、不同算法效果

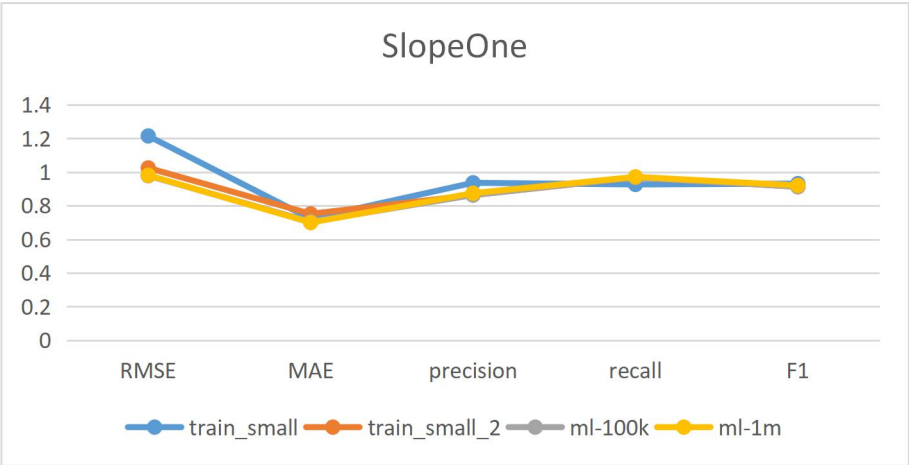
（1）基于商品的协同过滤（IBCF）



三种相似度表现结果相近，在数据集非常小的时候，三种相似度都表现一般。

（2）Slope One

评测指标	train_small	train_small_2	ml-100k	ml-1m
RMSE	1.214598	1.024531	0.978034	0.980918
MAE	0.722772	0.751678	0.70445	0.6997
precision	0.93617	0.863571	0.863338	0.87441
recall	0.926316	0.967974	0.968282	0.97132
F1	0.931217	0.912797	0.912803	0.920321



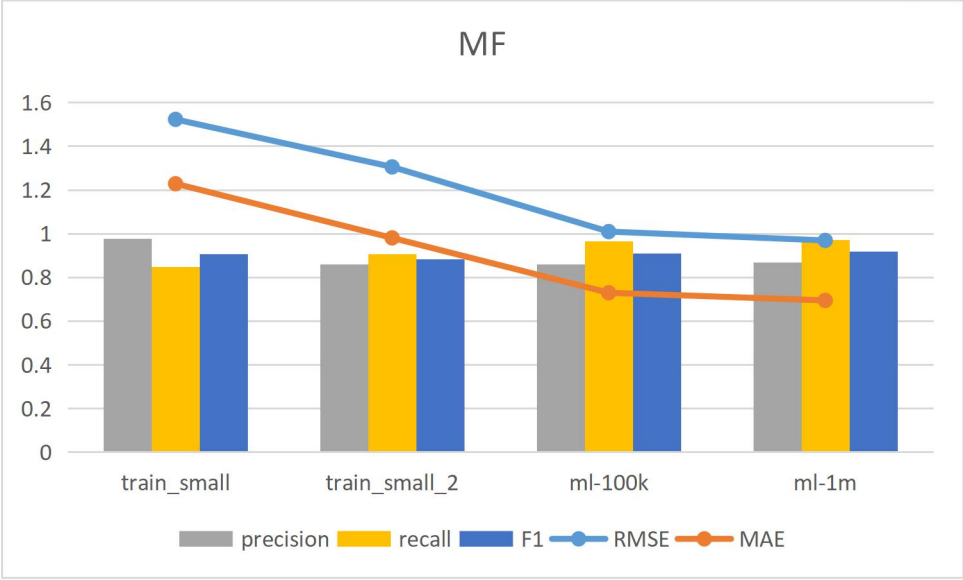
对于不同的数据集，Slope One 的效果都非常稳定。

（3）矩阵分解

参数选择：

矩阵分解算法包括两个参数需要调整，用户、商品因子矩阵的维数 K，和梯度下降的步长 a。K 越大，算法的运行时间越长；步长 a 决定了能够能够较好的梯度下降。经过实验，对于不同数据集采用如下参数能够得到较好的效果。

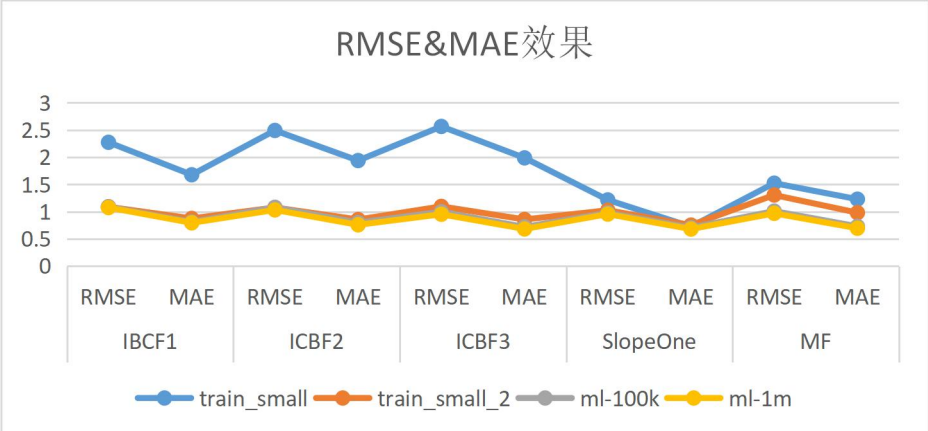
参数选择	train_small	train_small_2	ml-100k	ml-1m
K	100	50	10	10
a	0.001	0.001	0.0005	0.00005



可以看出，随着数据集逐渐增大，矩阵分解预测的效果越好。

（4）不同算法效果对比

a. 不同算法、数据集的 RMSE 和 MAE 效果如下图所示：



可以发现：

train_small数据集在IBCF和MF都在一定程度上效果不好，train_small数据集太小，已知

的评分不够，导致预测效果比较差。而Slope One在所有数据集上表现相对稳定。

随着数据集的增大，所有算法的效果都变好，并且非常接近。

总的来说，Slope One表现最好。