大数据大作业 2　　个性化推荐

许稼轩　　2016211092　　GIX 全球创新学院

问题描述

分别用协同过滤法（Collaborative Filtering）与矩阵分解法（Matrix Decomposing）完成个性化推荐的功能。在这次作业中，有 Netflix 网站上 10000 名用户对 10000 个电影不完全打分的数据，需要在此基础上实现对特定用户的个性化电影推荐。

Part 1 数据预处理

任务要求：在 users.txt，netflix_train.txt，netflix_test.txt 这三个文件上基础上，建立训练矩阵 X_train 和测试矩阵 X_test。

完成情况：完成对全体数据的导入（10000 名用户对 10000 个电影的评价），构建了 10000×10000 大小的训练矩阵 X_train 和测试矩阵 X_test。

代码以及思路说明：

```
train_data=load('netflix_train.txt');  %导入训练集数据
test_data=load('netflix_test.txt'); %导入测试集数据
id_data=load('users.txt');  %导入用户编号数据
X_train=zeros(10000,10000);  %训练矩阵
X_test=zeros(10000,10000);  %测试矩阵

for i=1:6897746
    user_ID=find(id_data==train_data(i,1));   %首先对训练集合数据对每一个用户，找到其ID在users.txt中对应的位置
    X_train(user_ID,train_data(i,2))=train_data(i,3);  %编写训练矩阵
end

for i=1:1719466
    user_ID=find(id_data==test_data(i,1)); %对测试集合数据对每一个用户，找到其ID在users.txt中对应的位置
    X_test(user_ID,test_data(i,2))=test_data(i,3); %编写测试矩阵
end
```

对于训练矩阵 X_train 和测试矩阵 X_test，需要解决用户 ID 以及电影 ID 一致的问题，也就是两个矩阵的同一行对应同一个用户，同一列对应同一个电影。电影的顺序比较好解决，其列数直接采用其 ID 编号即可。而用户的 ID 编号是一串字符，无法直接作为行数。这里采用 Matlab 的 find 函数，找到用户 ID 在 users.txt 文件中出现的顺序，作为用户的行数。

Part 2 基于协同过滤（Collaborative Filtering）的推荐算法

任务要求：在训练矩阵 X_train 的基础上，完成基于用户的协同过滤推荐算法。对于用户 i，若要预测其对电影 j 的评价，需要了解其他用户对电影 j 的打分以及这些用户的电影品味和用户 i 的相似度。

完成情况：基于 10000×10000 大小的训练矩阵 X_train，完成了对全体数据的训练，可以得到任意用户 i 对电影 j 打分的预测结果，并且将预测结果与测试矩阵 X_test 的数据进行比较，得到 RMSE。经过实际运行，RMSE＝1.0184，运行时间

为 37 秒。

代码以及思路说明：

```
X_norm=X_train;  %X_norm是X_train的归一话矩阵，X_norm中每一行向量的模为1
X_predict=zeros(10000,10000);%X_predict是最后的预测矩阵
for i = 1:10000
    X_norm(i,:)=X_norm(i,:)/norm(X_norm(i,:)); %采用向量的2范数对X_train每行归一化，得到X_norm
end
con_X_train=X_norm*X_norm'; %将归一化之后的X_norm矩阵乘以其转置，得到协方差矩阵con_X_train，该矩阵的ij元素代表用户i与用户j的相似度
Sparse_X=X_train;
Sparse_X(Sparse_X~=0)=1; %Sparse_X是X_train的指示矩阵，其ij元素为1代表用户i对j有评价
score=0;

for i = 1:10000
    for j = 1:10000
        if(X_test(i,j)~=0)
            score=con_X_train(i,:)*X_train(:,j)/(con_X_train(i,:)*Sparse_X(:,j));% 只考虑训练集中对电影j有评价的用户
            X_predict(i,j)=score;
        end
    end
end

RMSE=norm(X_predict-X_test,'fro')/sqrt(1719466);% 通过X_predict与X_test之差的fro范数得到RMSE


```

采用向量的2范数对X_train每行归一化，得到X_norm， X_norm中每一行向量的模为1。将归一化之后的X_norm矩阵乘以其转置，得到协方差矩阵con_X_train，矩阵con_X_train的ij元素代表用户i与用户j的相似度。在预测电影j最终的评分时，为了只考虑训练集中对电影j有评价的用户，构造Sparse_X，它是X_train的指示矩阵，其ij元素为1代表用户i对j有评价。X_predict是最后的预测矩阵，通过X_predict与X_test之差的fro范数得到RMSE。

Part 3 基于梯度下降的矩阵分解

任务要求：将训练矩阵X分解为U，V两个矩阵的乘积，$X_{m*n} \approx U_{m*k} V_{n*k}^{T}$

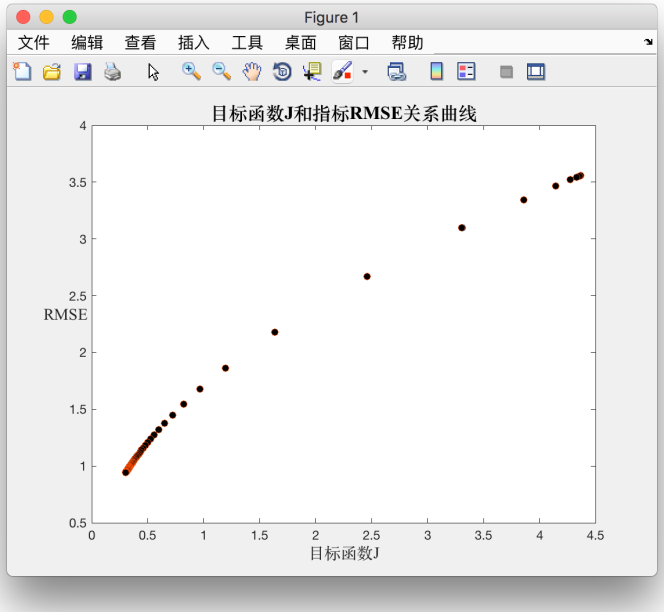
k为隐空间的维度，用梯度下降法求取U，V。

目标函数J为：$J = 0.5 \times \| A \circ (X - U V^{T}) \|_{F}^{2} + \lambda \| U \|_{F}^{2} + \lambda \| V \|_{F}^{2}$

完成情况：

a）k=50， λ = 0.01， a=0.0001的情况，画出迭代过程中目标函数值和测试集上RMSE的变化，给出最终的RMSE，并对结果简单分析。

目标函数值和测试集上RMSE的变化过程如图所示，最终得到的RMSE结果为0.9415，运行时间为131秒。



可见，与协调过滤算法相比，矩阵分解算法的计算精度更高，耗时更多。

b）调整 k 的值和 λ 的值，比较 RMSE 的效果。

这里调整 k 和 λ 的值，经过运行得到四组结果，详细的运行截图件文末附录。

k	λ	a	RMSE	运行时间	迭代步数
50	0.01	0.0001	0.9414	128 秒	74
20	0.01	0.0001	0.9413	112 秒	76
50	0.1	0.0001	0.9415	130 秒	74
50	0.001	0.0001	0.9415	134 秒	74

通过对比发现，不同的 λ 值对 RMSE 以及运行时间没有很大影响，k 值对 RMSE 也没有很大影响。不过 k 值越小，运行时间越小。所以此处 k＝20， λ ＝0.01 对效果最好。

代码以及思路说明：

```
k=50;
r=0.001;
a=0.0001;
U=0.01*rand(10000,k);
V=0.01*rand(10000,k);
X=X_train;
A=X_train;
A(A~=0)=1;
B=X_test;
B(B~=0)=1;


J=(norm(A.*(X-(U*V')),'fro')^2)/2+r*(norm(U,'fro')^2)+r*(norm(V,'fro')^2);
J=J/(10^7);
Jo=J+1;
sprintf('J值为:%d ',J)
steps=1;

while(Jo-J>0.0005 && steps<100 )
        Jo=J;

        J_U=(A.*(U*V'-X))*V+2*r*U;
        J_V=(A.*(U*V'-X))'*U+2*r*V;

        U=U-a*J_U;
        V=V-a*J_V;

        J=(norm(A.*(X-U*V'),'fro')^2)/2+r*(norm(U,'fro')^2)+r*(norm(V,'fro')^2);
        J=J/(10^7);
        %sprintf('J值为:%d ',J)
        RMSE=norm(B.*(U*V')-X_test,'fro')/sqrt(1719466);
        %srintf('RMSE值为:%d ',RMSE)
        x(steps)=J;
        y(steps)=RMSE;
        steps=steps+1;

end

        plot(x,y,'o','MarkerFaceColor','k')
        title('目标函数J和指标RMSE关系曲线','FontName','Times New Roman','FontWeight','Bold','FontSize',16)
        xlabel('目标函数J','FontName','Times New Roman','FontSize',14)
        ylabel('RMSE','FontName','Times New Roman','FontSize',14,'Rotation',0)
        hold on;

```

本例中，经过反复试探，发现初始 U，V 取较小值，以及步长 a 较小时，收敛结果较好。

Part 4 将协调过滤和矩阵分解的优缺点

在本次作业中，协同过滤的耗时较少，**RMSE** 较大；而矩阵分解的耗时较大，**RMSE** 较小。可见，协同过滤效率更高，而矩阵分解更准确。

1） 协同过滤优缺点

优点：

效率高，速度快；

随着数据量增加，准确性越来越高；

便于基于社交媒体的好友推荐；

缺点：如果用户没有对商品评价，则无法推荐。

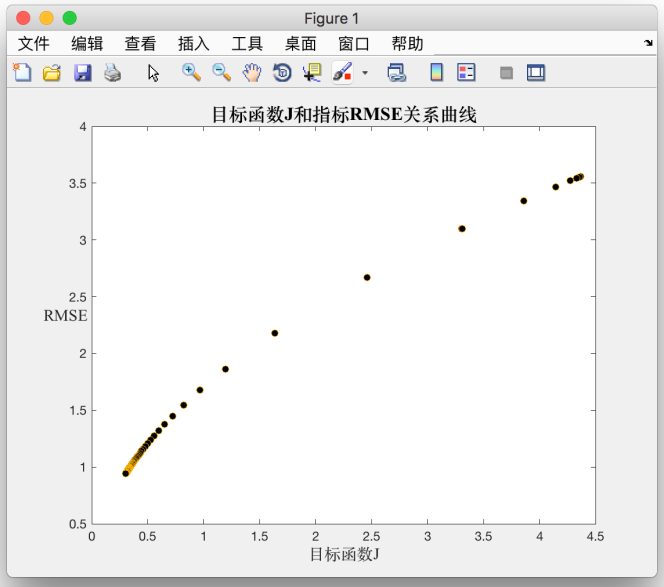
2） 矩阵分解优缺点

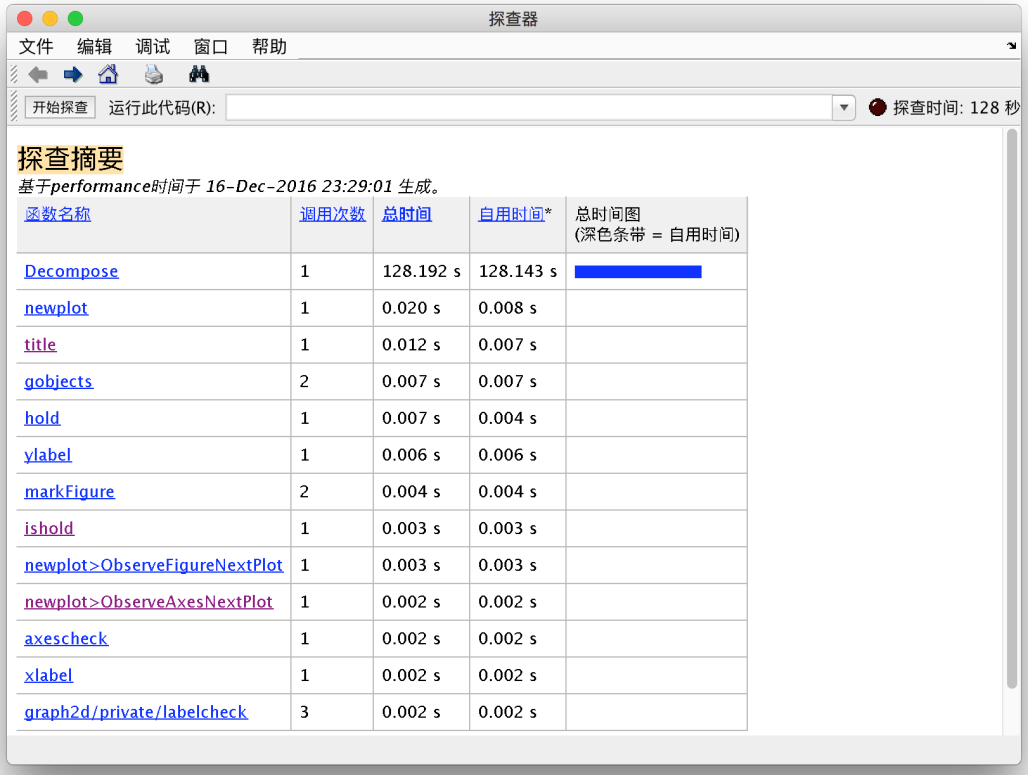
优点：精度高，扩展性好；

缺点：效率低，因子对意义不明确。

附录——矩阵分解在不同 k， λ 参数下的运行结果

k=50， λ = 0.01， a=0.0001

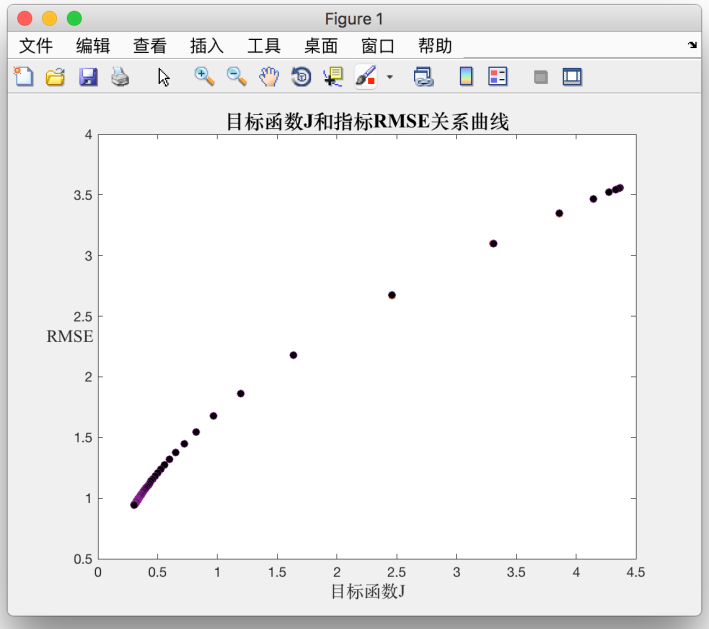




函数名称	调用次数	总时间	自用时间*	总时间图 (深色条带 = 自用时间)
Decompose	1	128.192 s	128.143 s	
newplot	1	0.020 s	0.008 s	
title	1	0.012 s	0.007 s	
gobjects	2	0.007 s	0.007 s	
hold	1	0.007 s	0.004 s	
ylabel	1	0.006 s	0.006 s	
markFigure	2	0.004 s	0.004 s	
ishold	1	0.003 s	0.003 s	
newplot>ObserveFigureNextPlot	1	0.003 s	0.003 s	
newplot>ObserveAxesNextPlot	1	0.002 s	0.002 s	
axescheck	1	0.002 s	0.002 s	
xlabel	1	0.002 s	0.002 s	
graph2d/private/labelcheck	3	0.002 s	0.002 s	

名称 ▲	值
a	1.0000e-04
A	*10000x10000 do...*
ans	'J值为:4.372735e...
B	*10000x10000 do...*
con_X_train	*10000x10000 do...*
i	7448
id_data	*10000x1 double*
j	9378
J	0.3037
J_U	*10000x50 double*
J_V	*10000x50 double*
Jo	0.3042
k	50
r	0.0100
RMSE	0.9414
score	3.8276
Sparse_X	*10000x10000 do...*
steps	74
test_data	*1719466x4 double*
train_data	*6897746x4 double*
U	*10000x50 double*
user_ID	9892
V	*10000x50 double*
x	*1x75 double*
X	*10000x10000 do...*
X_norm	*10000x10000 do...*
X_predict	*10000x10000 do...*
X_test	*10000x10000 do...*
X_train	*10000x10000 do...*
y	*1x75 double*

k=20, λ = 0.01, a=0.0001



探查摘要

基于performance时间于 16-Dec-2016 21:22:27 生成。

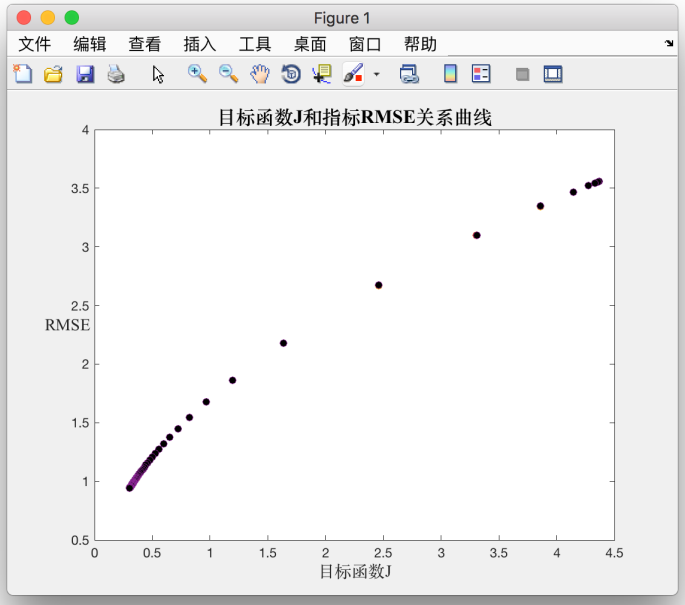
函数名称	调用次数	总时间	自用时间*	总时间图 (深色条带 = 自用时间)
Decompose	1	111.575 s	111.562 s	
newplot	1	0.007 s	0.002 s	
hold	1	0.004 s	0.003 s	
gobjects	2	0.003 s	0.003 s	
markFigure	1	0.001 s	0.001 s	
newplot>ObserveAxesNextPlot	1	0.001 s	0.001 s	
ishold	1	0.001 s	0.001 s	
newplot>ObserveFigureNextPlot	1	0.001 s	0.001 s	
axescheck	1	0.000 s	0.000 s	

自用时间是指函数耗费的时间，但不包括函数的子函数耗费的时间。自用时间还包括探查过程中产生的开销。

工作区

名称	值
a	1.0000e-04
A	*10000x10000 do...*
ans	'J值为:4.374486e...
B	*10000x10000 do...*
i	1719466
id_data	*10000x1 double*
J	0.3036
J_U	*10000x20 double*
J_V	*10000x20 double*
Jo	0.3041
k	20
r	0.0100
RMSE	0.9413
steps	76
test_data	*1719466x4 double*
train_data	*6897746x4 double*
U	*10000x20 double*
user_ID	9892
V	*10000x20 double*
x	*1x75 double*
X	*10000x10000 do...*
X_test	*10000x10000 do...*
X_train	*10000x10000 do...*
y	*1x75 double*

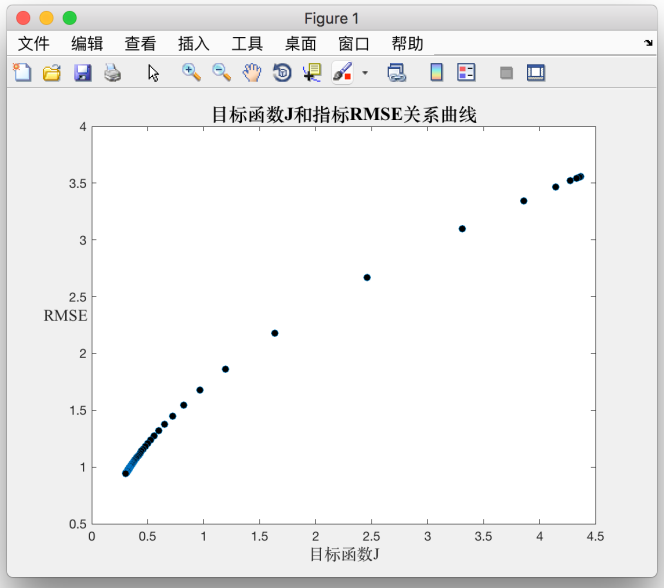
k=50, λ = 0.1, a=0.0001

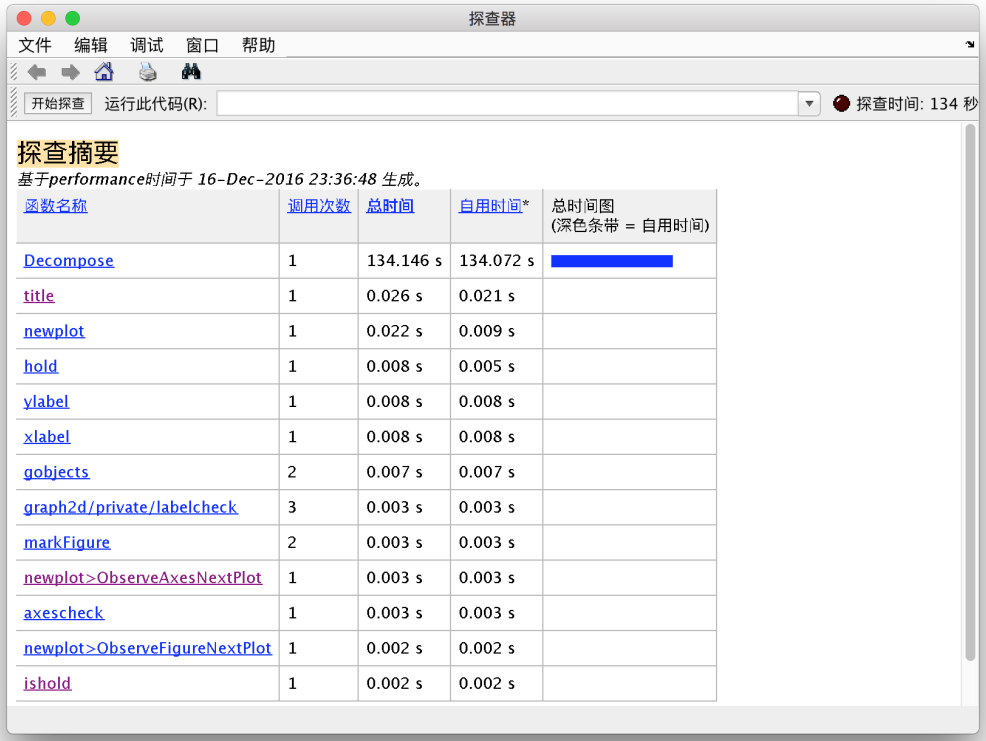


工作区

名称 ▲	值
a	1.0000e-04
A	*10000x10000 do...*
ans	'J值为:4.372743e...
B	*10000x10000 do...*
i	1719466
id_data	*10000x1 double*
J	0.3042
J_U	*10000x50 double*
J_V	*10000x50 double*
Jo	0.3047
k	50
r	0.1000
RMSE	0.9415
steps	74
test_data	*1719466x4 double*
train_data	*6897746x4 double*
U	*10000x50 double*
user_ID	9892
V	*10000x50 double*
x	*1x75 double*
X	*10000x10000 do...*
X_test	*10000x10000 do...*
X_train	*10000x10000 do...*
y	*1x75 double*

k=50, λ = 0.001, a=0.0001





探查摘要

基于performance时间于 16-Dec-2016 23:36:48 生成。

函数名称	调用次数	总时间	自用时间*	总时间图 (深色条带 = 自用时间)
Decompose	1	134.146 s	134.072 s	
title	1	0.026 s	0.021 s	
newplot	1	0.022 s	0.009 s	
hold	1	0.008 s	0.005 s	
ylabel	1	0.008 s	0.008 s	
xlabel	1	0.008 s	0.008 s	
gobjects	2	0.007 s	0.007 s	
graph2d/private/labelcheck	3	0.003 s	0.003 s	
markFigure	2	0.003 s	0.003 s	
newplot>ObserveAxesNextPlot	1	0.003 s	0.003 s	
axescheck	1	0.003 s	0.003 s	
newplot>ObserveFigureNextPlot	1	0.002 s	0.002 s	
ishold	1	0.002 s	0.002 s	

工作区

名称 ▲	值
a	1.0000e-04
A	*10000x10000 do...*
ans	'J值为:4.372742e...
B	*10000x10000 do...*
i	1719466
id_data	*10000x1 double*
J	0.3037
J_U	*10000x50 double*
J_V	*10000x50 double*
Jo	0.3042
k	50
r	1.0000e-03
RMSE	0.9415
steps	74
test_data	*1719466x4 double*
train_data	*6897746x4 double*
U	*10000x50 double*
user_ID	9892
V	*10000x50 double*
x	*1x75 double*
X	*10000x10000 do...*
X_test	*10000x10000 do...*
X_train	*10000x10000 do...*
y	*1x75 double*