

# 数据挖掘期末项目

1352875 黄安娜  
1354361 刘林青

## 目录

- 数据挖掘期末项目 ..... 1
- 1 项目设计 ..... 2
  - 1.1 划分测试集、训练集 ..... 2
  - 1.2 设定数据结构 ..... 2
  - 1.3 计算相似值 ..... 2
  - 1.4 计算 baseline ..... 3
  - 1.5 找相似用户和相似电影 ..... 3
  - 1.6 计算 rating ..... 3
- 3 kMeans Clustering ..... 4
- 4 SVD Dimensionality Reduction ..... 8
  - 4.1 相关概念： ..... 8
  - 4.2 相关分析及算法描述： ..... 8
  - 4.3 SVD 降维函数（matlab）： ..... 10
  - 4.4 具体步骤及关键代码： ..... 11
- 5 Prediction Result ..... 16
- 6 小组分工 ..... 17

# 1 项目设计

## Recommendation Part[未优化]

### 1.1 划分测试集、训练集

### 1.2 设定数据结构

#### ● user \_ movie matrix

mID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
user																		
1		5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3	NaN	NaN
6		4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4	NaN
7	NaN	NaN	NaN	NaN	NaN	4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4	NaN	4	4	NaN
9		5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10		5	5	NaN	NaN	NaN	NaN	4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
12	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
15	NaN	NaN	NaN	NaN	NaN	4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
16	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
18		4	2	NaN	NaN	NaN	NaN	NaN	NaN	5	NaN	NaN	NaN	NaN	NaN	NaN	4	NaN
19		5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5	NaN	NaN	NaN	NaN	NaN	NaN	4	NaN
20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
21		3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
22	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3	NaN	NaN	NaN	NaN	NaN	4	NaN	NaN
23		4	2	NaN	NaN	NaN	3	NaN	NaN	NaN	2	NaN	NaN	NaN	NaN	4	NaN	NaN
24	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

方便之后的 user\_user 和 item\_item 调用数值  
采用 python pandas，生成任意维度的矩阵

#### ● movie\_bin directory

{movie\_name: {bin\_num1 :[[timestamp1,...],total\_rating]  
bin\_num2: [[timestamp2,...],total\_rating]

...}}

### 1.3 计算相似值

Pearson Similarity

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)(r_{ys} - \bar{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \bar{r}_y)^2}}$$

## 1.4 计算 baseline

$$b_{ui}(t) = \mu + b_u + \alpha_u \cdot \text{dev}_u(t) + b_{u,t} + b_i + b_{i,\text{Bin}(t)} \quad (11)$$

### ● user:

$$b_u + \alpha_u \cdot \text{dev}_u(t)$$

用户时间的偏移量：根据上述公式计算，其中：

alpha\_u: 取值为该用户时间均值和中值的比值

dev: beta = 0.4

### ● item:

$$b_i + b_{i,\text{Bin}(t)}$$

计算该电影对应时间在哪个 Bin 当中，每个 bin 都有偏移量

Bin: 统计该 bin 所有 rating 的均值 - 全部电影 rating 均值

## 1.5 找相似用户和相似电影

由 kMeans 得到目标用户或电影的 cluster，找到相似度最大的 20 个

## 1.6 计算 rating

$$r_{xi} = \underset{\text{user}}{b_{xi}} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

由以上公式，分别调用 user\_user CF 和 item\_item CF 来计算对应的 rating

### 3 kMeans Clustering

(i) 见附件 result\_20 目录

(ii) 使用 kMeans 对用户进行聚类，得到 k=5,k=10,k=20 和 k=100 的聚类结果。分别计算此时的

$$\phi = \sum_{x \in \mathcal{X}} \min_{c \in \mathcal{C}} \|x - c\|^2$$

得到结果如下：

K=5:



K=10:

```

命令行窗口
Replicate 1, 51 iterations, total sum of distances = 9.95686e+06.
Best total sum of distances = 9.95686e+06
the cost is 9956858.5209
fx >> |

```

探查器

文件(F) 编辑(E) 调试(B) 窗口(W) 帮助(H)

← → ↩ ↶ ↷

启动探查(P) 运行此代码(R):

探查时间: 644 秒

探查摘要

已使用 cpu 时间生成 07-Jan-2016 23:53:14。

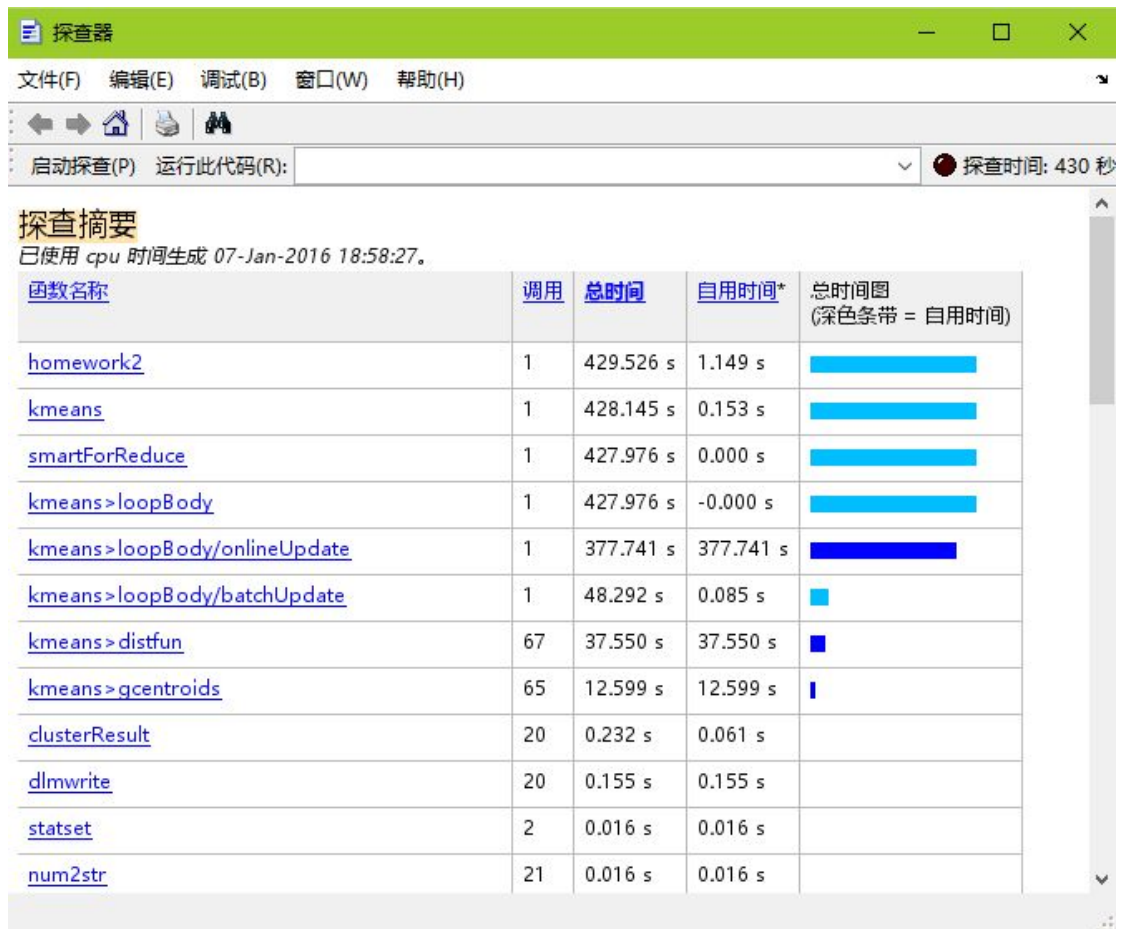
函数名称	调用	总时间	自用时间*	总时间图 (深色条带 = 自用时间)
<a href="#">homework2</a>	1	643.582 s	1.720 s	<div></div>
<a href="#">kmeans</a>	1	641.578 s	0.184 s	<div></div>
<a href="#">smartForReduce</a>	1	641.378 s	-0.000 s	<div></div>
<a href="#">kmeans&gt;loopBody</a>	1	641.363 s	0.000 s	<div></div>
<a href="#">kmeans&gt;loopBody/onlineUpdate</a>	1	616.548 s	616.548 s	<div></div>
<a href="#">kmeans&gt;loopBody/batchUpdate</a>	1	23.480 s	0.032 s	<div></div>
<a href="#">kmeans&gt;distfun</a>	32	17.484 s	17.484 s	<div></div>
<a href="#">kmeans&gt;qcentroids</a>	30	7.267 s	7.267 s	<div></div>
<a href="#">clusterResult</a>	10	0.269 s	0.030 s	<div></div>
<a href="#">dlmwrite</a>	10	0.239 s	0.239 s	<div></div>
<a href="#">unique</a>	29	0.032 s	0.016 s	<div></div>
<a href="#">statset</a>	3	0.016 s	0.016 s	<div></div>

K=20:

```

命令行窗口
Replicate 1, 76 iterations, total sum of distances = 9.62103e+06.
Best total sum of distances = 9.62103e+06
the cost is 9621030.5984

```



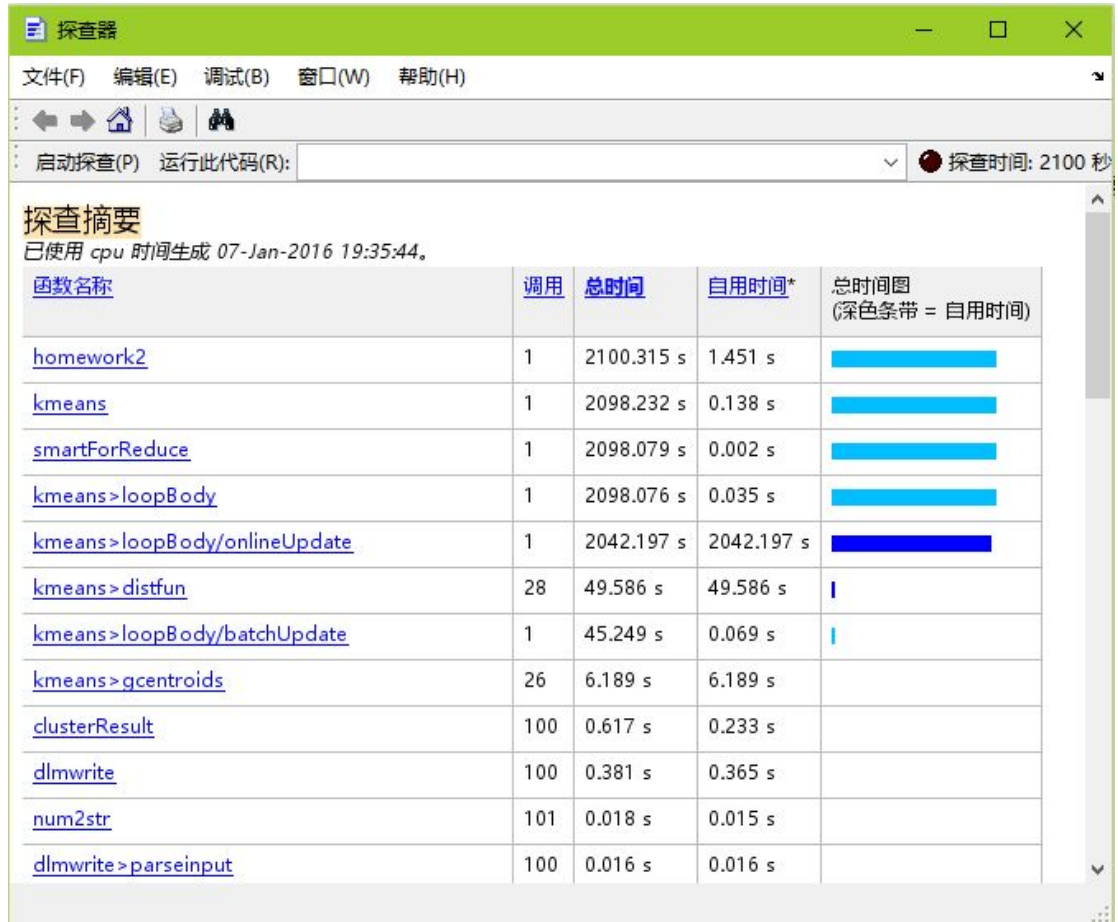
K=100:

命令行窗口

```

Replicate 1, 47 iterations, total sum of distances = 8.7452e+06.
Best total sum of distances = 8.7452e+06
the cost is 8745204.8687
fx >>

```



可以观察到在一定程度上，k 的值越大，聚类结果越好，但是时间也会有所增加。

为了后续进行 **based on item** 进行推荐，还要对电影进行聚类。

(iii) 将 kMeans 应用到推荐中：

以 k=20 为例，在第一步的推荐算法中，是利用训练集所对应的整个用户评分矩阵来计算用户之间、以及电影之间的相似度的，而进行了聚类之后，就将相似度的计算范围缩小了，只需在一个聚类中计算相似度，然后再取，因此可以极大的加快相似度的计算时间，速度约是原来的 6 倍，并且 RSME 的值基本不变（现在：1.0058 原：0.9673），预测效果基本不变。



## 4 SVD Dimensionality Reduction

### 4.1 相关概念：

#### SVD：

SVD (Singular value decomposition 奇异值分解) 是用来将一个大的矩阵以降低维数的方式进行有损地压缩。从信息论的角度来说，数据之间存在相关性，就具有可压缩性，因此 SVD 可以很好地用在对评分矩阵的压缩中。SVD 分解公式如下：

任意一个  $M \times N$  的矩阵  $A$  ( $M$  行  $\times$   $N$  列,  $M > N$ )，可以被写成三个矩阵的乘积：

1.  $U$ ： ( $M$  行  $M$  列的列正交矩阵)
  2.  $S$ ： ( $M \times N$  的对角线矩阵，矩阵元素非负)
  3.  $V$ ： ( $N \times N$  的正交矩阵的转置)
- 即  $A = U \times S \times V'$  (矩阵  $V$  需要转置)

### 4.2 相关分析及算法描述：

将矩阵  $A$  分解之后，可以看到矩阵  $S$  是一个对角矩阵，对角线元素为非负，且递减，这些对角线上的元素即为矩阵  $A$  的奇异值。当进行降维时，可以取矩阵  $S$  对角线上的前  $k$  个元素，相应的地  $U$  和  $V$  应该取前  $k$  列，这样就可以把矩阵  $A$  的维数从  $N$  降到了  $k$ 。此时  $A$  的列向量可以使用子分区  $V$  来特征性表示，而  $A$  的行向量可以使用子分区  $U$  来特征性表示。

使用具体的小规模数据进行分析计算如下：

如下图所示，该矩阵的列代表 用户、行代表

	Ben	Tom	John	Fred
Season 1	5	5	0	5
Season 2	5	0	3	4
Season 3	3	4	0	3
Season 4	0	0	5	3
Season 5	5	4	4	5
Season 6	5	4	5	5



A =

```
5    5    0    5
5    0    3    4
3    4    0    3
0    0    5    3
5    4    4    5
5    4    5    5
```

```
[U,S,Vttranspose]=svd(A)
```

U =

```
-0.4472  -0.5373  -0.0064  -0.5037  -0.3857  -0.3298
-0.3586   0.2461   0.8622  -0.1458   0.0780   0.2002
-0.2925  -0.4033  -0.2275  -0.1038   0.4360   0.7065
-0.2078   0.6700  -0.3951  -0.5888   0.0260   0.0667
-0.5099   0.0597  -0.1097   0.2869   0.5946  -0.5371
-0.5316   0.1887  -0.1914   0.5341  -0.5485   0.2429
```

S =

```
17.7139         0         0         0
         0   6.3917         0         0
         0         0   3.0980         0
         0         0         0   1.3290
         0         0         0         0
         0         0         0         0
```

Vttranspose =

```
-0.5710  -0.2228   0.6749   0.4109
-0.4275  -0.5172  -0.6929   0.2637
-0.3846   0.8246  -0.2532   0.3286
-0.5859   0.0532   0.0140  -0.8085
```

此时取 k=2 进行降维得到降维后的 U、S、V:

U		S		V.transpose	
-0.4472	0.5373	17.7139	0.0000	-0.5710	0.2228
-0.3586	-0.2461	0.0000	6.3917	-0.4275	0.5172
-0.2925	0.4033			-0.3846	-0.8246
-0.2078	-0.6700			-0.5859	-0.0532
-0.5099	-0.0597				
-0.5316	-0.1887				

此时，使用降维后的 U、S、V 来相乘得到 A2，

```
A2=U(1:6,1:2)*S(1:2,1:2)*(V(1:4,1:2))'
```

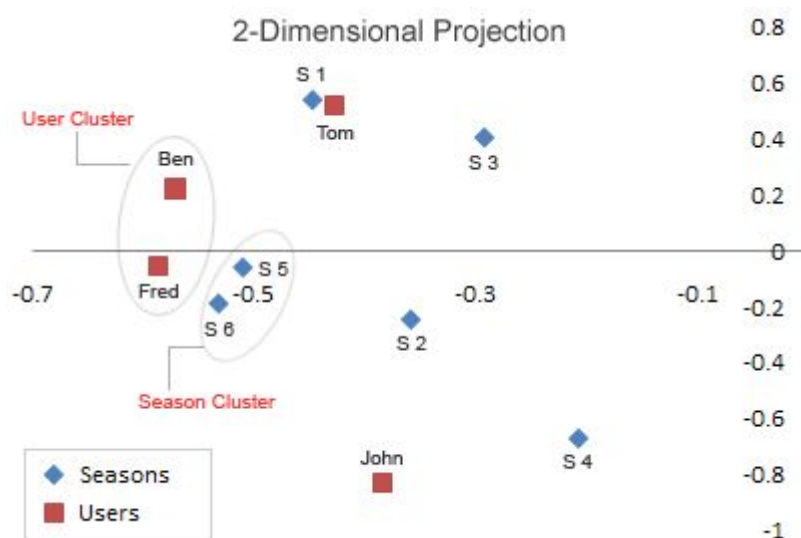
结果:

A2 =

5.2885	5.1627	0.2149	4.4591
3.2768	1.9021	3.7400	3.8058
3.5324	3.5479	-0.1332	2.8984
1.1475	-0.6417	4.9472	2.3846
5.0727	3.6640	3.7887	5.3130
5.1086	3.4019	4.6166	5.5822

对比 A 可以观察到, A2 与 A 很相近, 但是有些许误差, 这就是前面提到的数据的有损压缩。  
接下来开始分析矩阵中的数据相关性:

在进行降维之后, U 的每一行可以用一个二维向量表示, V 的每一行也可以用一个二维向量表示, 在降维到二维之后, 就可以可视化地观察数据了, 如下图所示:



可以观察到 Ben 与 Fred 的 distance 很小, 所以相似度很高; 同样 S5 和 S6 也是如此, 这样的结果与原始矩阵 A 是相同的, 所以可以使用降维后的矩阵 U 和 V 来近似代表 A, 也就是说将 A 压缩成了矩阵 U 和矩阵 V。

因此, 当有一个新用户 Bob 时, 可以使用如下公式, 对该用户进行降维, 然后采用某种相似度量标准来找该用户的相似用户 (实际应用在该项目中我们采用 pearson 相关系数)

$$Bob_{2D} = Bob^T \times U_2 \times S_k^{-1}$$

## 4.3 SVD 降维函数 (matlab) :

(1) 见附件 3.txt

(2) 分析如下:

$$[U,S,V] = \text{svd}(A)$$

## 4.4 具体步骤及关键代码：

首先利用 matlab 的 svd 公式，对训练集进行奇异值分解

```
[U,S,V]= svd(training_rating);
```

接着利用题目所给的条件

and the example <sup>2</sup> demonstrates the use of Jama's SVD. You need to ensure the dimensionality reduction incurs at most 10% energy loss, i.e.,  $\sum_{i'=1}^{k'} (\lambda_{i'})^2 \geq 90\% * \sum_{i=1}^k (\lambda_i)^2$ , where  $\lambda_{i'}$  is one of the number  $k'$  of retained singular values

确定 k 的最小取值

```
tempMatrix = S.^2;
sum_before = sum(tempMatrix(:));
for k = 1000:-1:1
    temp = tempMatrix(1:k,1:k);
    sum_after = sum(temp(:));
    if(sum_after/sum_before<0.9)
        break;
    end
end
disp(k);
%dimension
dim=k;
```

然后将原评论矩阵降维到维度为 k

%U\_new 表示用户特征子区

```
U_new = U(:,1:dim);
```

```
S_new = S(1:dim,1:dim);
```

%V\_new 表示电影特征子区

```
V_new = V(:,1:dim);
```

```
result = [num2str(dim),',',size(data,2)];
```

```
fid=fopen('3.txt','wt');
```

```
fprintf(fid,'%s\n',result);
```

```
fclose(fid);
```

```
save userTraining U_new;
```

```
save movieTraining V_new;
```

%movies 3952\*969 降维 降维后的数据集 based on movies

```
movies = rating_matrix'*U_new*S_new^-1;
```

```
save movieData movies;
```

%%

%users 6040\*969 降维 降维后的数据集 based on users

```
users = rating_matrix*V_new*S_new^-1;
```

```
save userData users;
```

此时就得到了降维后的数据集 movies 和 users 以及用训练集得到的用户及电影特征子区，接下来就是将其应用到推荐系统了。

以 based on user 为例，使用 users 及 U\_new 矩阵，针对每一个 user，求出与其相似度最高的前 n 个来进行相关的预测，后续步骤与 Problem1 中的方法相同。

数据展示如下：

U:

U												
6040x6040 double												
	1	2	3	4	5	6	7	8	9	10	11	
1	-0.0044	-0.0057	-0.0016	0.0034	-0.0155	-0.0028	0.0084	0.0046	-0.0158	-0.0112	-0.0109	
2	-0.0089	-0.0040	3.4565e-04	0.0117	0.0073	-0.0238	0.0097	0.0028	-0.0016	0.0026	0.0067	-3.4
3	-0.0044	5.7262e-04	0.0029	0.0052	-0.0078	-0.0082	-0.0048	0.0064	0.0012	-0.0054	-0.0022	
4	-0.0025	-5.3840e-04	0.0075	0.0073	-0.0034	-0.0043	-0.0014	0.0030	-0.0064	-6.8504e-04	0.0014	
5	-0.0094	-0.0063	-0.0192	0.0216	0.0059	0.0153	-0.0011	-0.0036	0.0046	0.0041	-0.0171	
6	-0.0036	-0.0026	-0.0024	-0.0022	-0.0082	-0.0047	0.0130	0.0037	-0.0058	0.0037	-0.0061	
7	-0.0032	0.0050	0.0027	0.0117	5.4490e-04	-0.0090	0.0022	6.1556e-04	9.0497e-04	-0.0024	0.0028	
8	-0.0091	-0.0010	-0.0200	0.0200	0.0158	-0.0097	0.0096	-0.0155	-0.0187	0.0051	0.0018	3.7
9	-0.0082	-0.0049	-0.0186	0.0235	-0.0033	-0.0043	9.9499e-04	-0.0013	-0.0116	-0.0109	7.4857e-04	
10	-0.0255	0.0028	0.0127	-0.0245	-0.0518	5.6416e-04	0.0231	-0.0083	-0.0068	0.0027	0.0127	
11	-0.0100	-0.0055	-0.0245	0.0062	-0.0135	-0.0012	-0.0152	-6.0956e-04	1.4902e-04	-0.0071	0.0138	5.8
12	-0.0012	-0.0026	7.4789e-04	0.0019	7.9645e-04	-0.0014	2.1389e-04	0.0023	-3.1150e-05	-0.0049	0.0011	
13	-0.0081	0.0097	0.0126	0.0137	-0.0051	-0.0167	0.0084	-0.0067	-0.0042	-6.9015e-04	-0.0073	
14	-0.0018	-0.0010	-0.0042	0.0081	-0.0029	0.0057	-0.0011	0.0078	0.0016	0.0050	-0.0030	
15	-0.0118	0.0106	-0.0260	0.0220	0.0072	-0.0046	0.0060	0.0216	0.0033	0.0117	0.0146	

S:

U S												
6040x3952 double												
	1	2	3	4	5	6	7	8	9	10	11	
1	1.6693e+03	0	0	0	0	0	0	0	0	0	0	
2	0	585.6046	0	0	0	0	0	0	0	0	0	
3	0	0	521.3837	0	0	0	0	0	0	0	0	
4	0	0	0	467.0885	0	0	0	0	0	0	0	
5	0	0	0	0	401.9766	0	0	0	0	0	0	
6	0	0	0	0	0	369.5831	0	0	0	0	0	
7	0	0	0	0	0	0	349.5320	0	0	0	0	
8	0	0	0	0	0	0	0	320.7546	0	0	0	
9	0	0	0	0	0	0	0	0	303.3971	0	0	
10	0	0	0	0	0	0	0	0	0	293.7230	0	
11	0	0	0	0	0	0	0	0	0	0	267.045	
12	0	0	0	0	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	0	0	0	0	0	
14	0	0	0	0	0	0	0	0	0	0	0	
15	0	0	0	0	0	0	0	0	0	0	0	
16	0	0	0	0	0	0	0	0	0	0	0	

V:

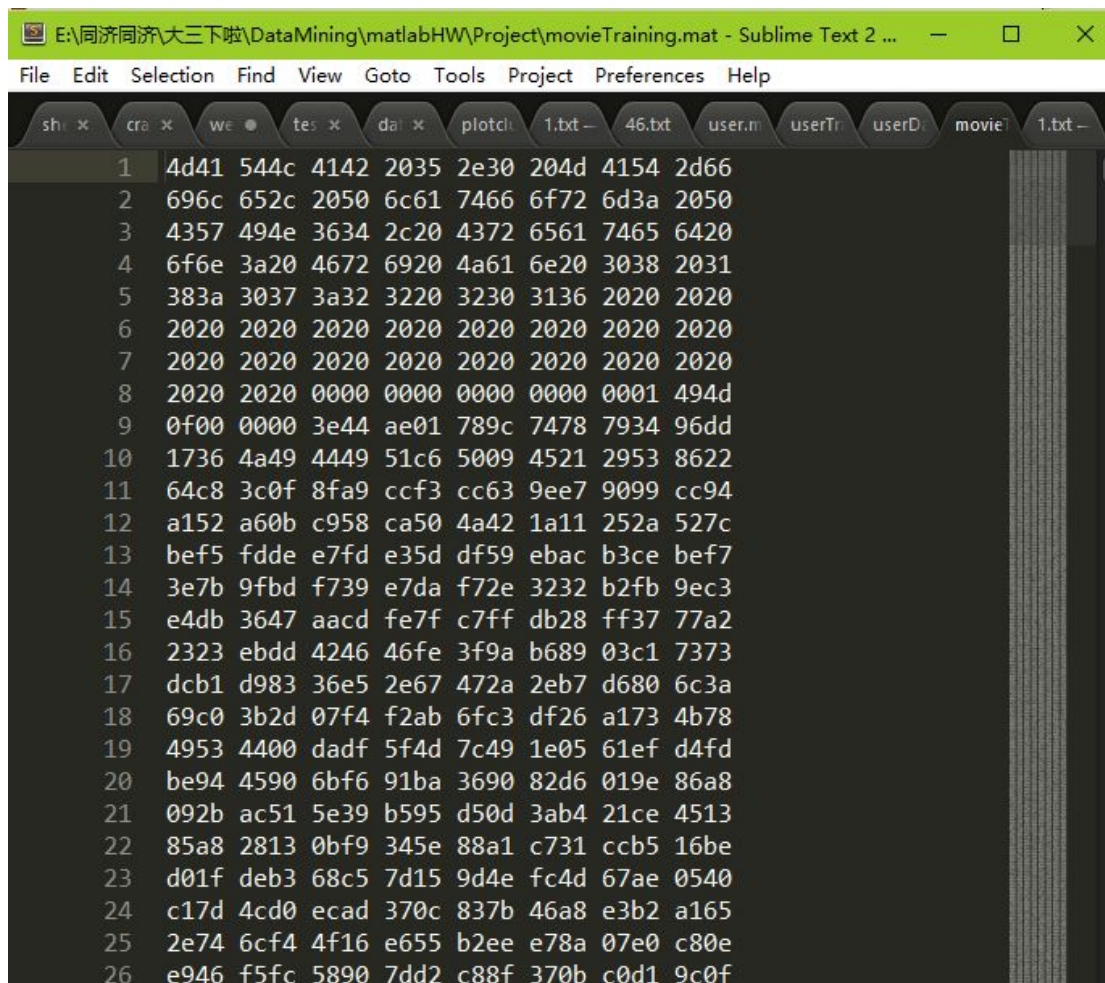


	1	2	3	4	5	6	7	8	9	10	11
1	-0.0654	0.0031	-0.0266	0.0344	-0.1189	-0.0161	0.0137	-1.4263e-04	0.0042	-0.1019	0.0
2	-0.0289	0.0349	0.0056	-0.0252	-0.0155	0.0159	0.0400	-0.0057	-0.0206	-0.0279	-0.0
3	-0.0175	0.0171	-0.0186	-0.0248	-0.0069	-0.0023	0.0173	0.0100	8.4808e-04	-0.0021	0.0
4	-0.0075	0.0033	-0.0185	-0.0159	0.0018	5.7824e-04	0.0092	-0.0080	-0.0045	0.0073	0.0
5	-0.0122	0.0155	-0.0203	-0.0275	-0.0088	-0.0071	0.0114	0.0102	0.0052	0.0014	0.0
6	-0.0372	0.0232	-0.0084	0.0396	0.0759	-0.0401	-0.0234	9.6929e-04	0.0339	-0.0413	-0.0
7	-0.0193	0.0060	-0.0327	-0.0339	-0.0194	-0.0237	0.0412	-0.0118	0.0024	0.0328	0.0
8	-0.0028	0.0038	-0.0020	-0.0037	-0.0052	0.0030	0.0097	0.0033	-0.0043	-0.0072	-0.0
9	-0.0049	0.0132	9.8802e-04	-0.0048	0.0105	-0.0027	0.0043	0.0052	0.0115	-0.0065	-0.0
10	-0.0369	0.0573	0.0103	0.0094	0.0230	-0.0491	0.0291	0.0087	0.0404	-0.0287	-0.0
11	-0.0368	6.8932e-04	-0.0569	-0.0272	-0.0229	-0.0451	0.0423	-0.0255	8.2318e-04	0.0370	0.0
12	-0.0067	0.0106	6.2193e-04	-0.0180	0.0063	0.0204	0.0038	0.0127	-0.0070	-0.0204	0.0
13	-0.0033	0.0032	6.1178e-04	-0.0063	-0.0075	0.0057	0.0140	0.0011	-0.0053	-0.0089	-0.0
14	-0.0074	-0.0034	-0.0105	-0.0081	0.0173	-3.5473e-04	0.0036	-0.0066	-0.0115	0.0039	-0.0
15	-0.0063	0.0120	6.4846e-04	-0.0102	0.0062	0.0021	0.0140	0.0034	0.0097	-0.0056	-0.0
16	-0.0279	9.5006e-04	-0.0360	0.0135	0.0552	-0.0094	-0.0193	-0.0143	-0.0232	-0.0144	0.0

UserTraing:

	1	2	3	4	5	6	7	8	9	10	11
1	4d41	544c	4142	2035	2e30	204d	4154	2d66			
2	696c	652c	2050	6c61	7466	6f72	6d3a	2050			
3	4357	494e	3634	2c20	4372	6561	7465	6420			
4	6f6e	3a20	4672	6920	4a61	6e20	3038	2031			
5	383a	3037	3a32	3020	3230	3136	2020	2020			
6	2020	2020	2020	2020	2020	2020	2020	2020			
7	2020	2020	2020	2020	2020	2020	2020	2020			
8	2020	2020	0000	0000	0000	0000	0001	494d			
9	0f00	0000	da87	ae02	789c	2c97	7738	95ff			
10	ffc7	5146	0a51	111a	da91	8615	6536	24a3			
11	42f6	3ee7	58e7	100e	8e3d	ceb1	f7de	7b45			
12	6908	0d32	9e66	4529	2bf9	1829	1489	4a65			
13	34e8	e77b	5dbf	fbba	ded7	ebba	efeb	bedf			
14	7fdc	efd7	f3f9	7a3e	b899	9898	965d	3b59			
15	d8d6	2ac7	dafa	5ffd	dfc5	faff	f719	024c			
16	4c1d	eb98	9898	ffff	99be	a593	8de7	5add			
17	b0b6	9ad6	be63	4a6d	715f	b7dd	152a	e2cc			
18	999a	9783	b09c	bfb7	eb15	bf2b	fcfa	0d59			
19	4bed	ad21	722e	d1d2	e96c	30be	e84f	8b16			
20	965d	4382	b9a1	81d6	2c19	df44	1eb3	8764			
21	0761	5da5	4279	cb7b	3a0e	6cfd	acae	b32f			
22	0bef	78b5	6d1e	0685	c0af	c237	f6dc	0f5d			
23	38f9	5324	0edf	a5c3	d2bc	fcee	24d5	1c12			
24	d55d	0450	b746	60d4	e550	ebde	6e3d	e856			

movieTraining:



```
1 4d41 544c 4142 2035 2e30 204d 4154 2d66
2 696c 652c 2050 6c61 7466 6f72 6d3a 2050
3 4357 494e 3634 2c20 4372 6561 7465 6420
4 6f6e 3a20 4672 6920 4a61 6e20 3038 2031
5 383a 3037 3a32 3220 3230 3136 2020 2020
6 2020 2020 2020 2020 2020 2020 2020 2020
7 2020 2020 2020 2020 2020 2020 2020 2020
8 2020 2020 0000 0000 0000 0000 0001 494d
9 0f00 0000 3e44 ae01 789c 7478 7934 96dd
10 1736 4a49 4449 51c6 5009 4521 2953 8622
11 64c8 3c0f 8fa9 ccf3 cc63 9ee7 9099 cc94
12 a152 a60b c958 ca50 4a42 1a11 252a 527c
13 bef5 fdde e7fd e35d df59 ebac b3ce bef7
14 3e7b 9fbd f739 e7da f72e 3232 b2fb 9ec3
15 e4db 3647 aacd fe7f c7ff db28 ff37 77a2
16 2323 ebdd 4246 46fe 3f9a b689 03c1 7373
17 dcb1 d983 36e5 2e67 472a 2eb7 d680 6c3a
18 69c0 3b2d 07f4 f2ab 6fc3 df26 a173 4b78
19 4953 4400 dadf 5f4d 7c49 1e05 61ef d4fd
20 be94 4590 6bf6 91ba 3690 82d6 019e 86a8
21 092b ac51 5e39 b595 d50d 3ab4 21ce 4513
22 85a8 2813 0bf9 345e 88a1 c731 ccb5 16be
23 d01f deb3 68c5 7d15 9d4e fc4d 67ae 0540
24 c17d 4cd0 ecad 370c 837b 46a8 e3b2 a165
25 2e74 6cf4 4f16 e655 b2ee e78a 07e0 c80e
26 e946 f5fc 5890 7dd2 c88f 370b c0d1 9c0f
```

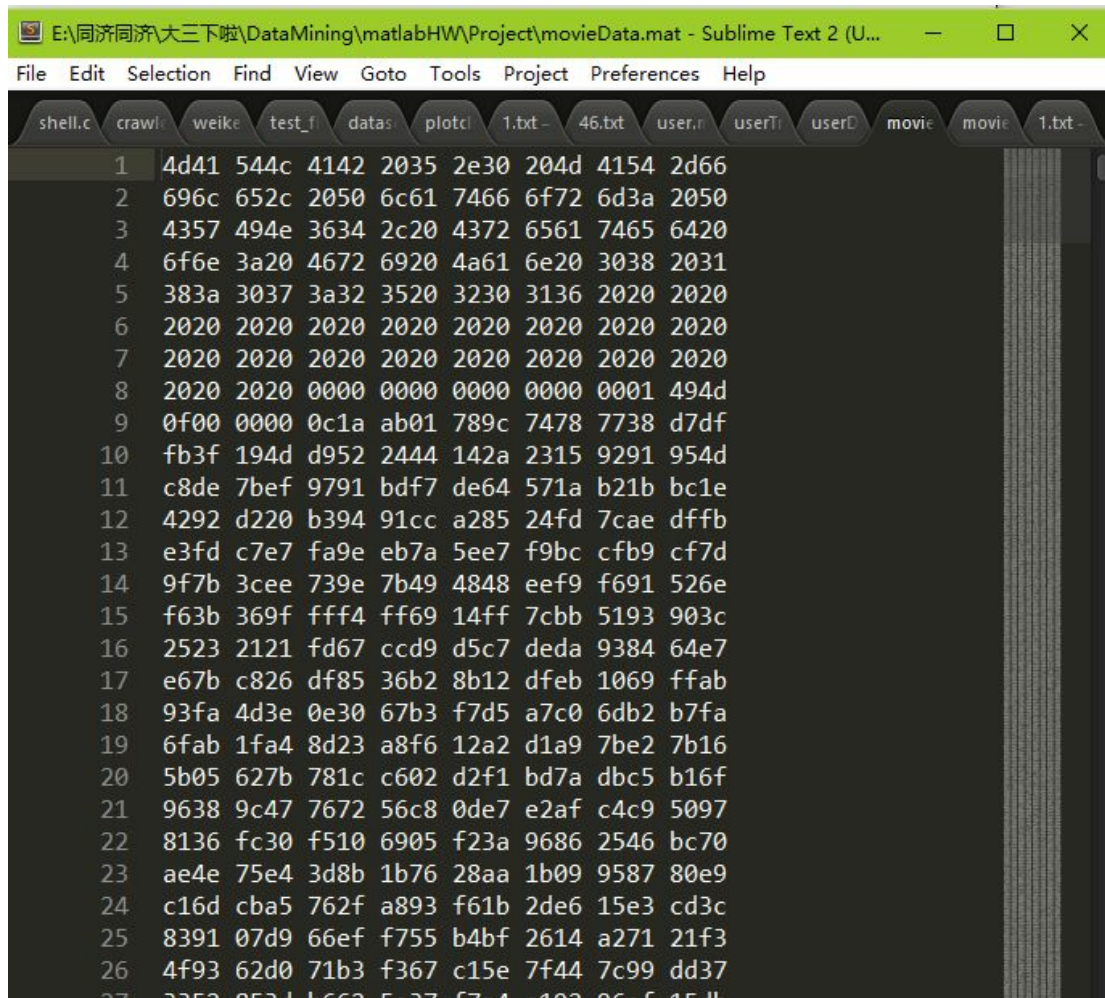
userData:



```
1 4d41 544c 4142 2035 2e30 204d 4154 2d66
2 696c 652c 2050 6c61 7466 6f72 6d3a 2050
3 4357 494e 3634 2c20 4372 6561 7465 6420
4 6f6e 3a20 4672 6920 4a61 6e20 3038 2031
5 383a 3037 3a32 3820 3230 3136 2020 2020
6 2020 2020 2020 2020 2020 2020 2020 2020
7 2020 2020 2020 2020 2020 2020 2020 2020
8 2020 2020 0000 0000 0000 0000 0001 494d
9 0f00 0000 477e ae02 789c 2c97 0934 55df
10 fbc6 cd53 4529 4525 890a 99a2 9108 45a1
11 4124 73a6 6b9e b9d7 704d 17d7 35cf f33c
12 87a2 bee6 221e 1952 144a a850 2a44 4a69
13 a40c fffb 5beb 7fd6 da6b af73 ce3a 67bf
14 7bed e77d dfcf c3cd c0c0 b0e4 d9cb c446
15 9f39 e8e3 7ff3 ff2e d6ff bfcf dcc1 c0f0
16 9899 8181 f1ff 9f91 bc6c 3cbd e833 277d
17 b4d3 bfeb b49c 1e22 2911 e1a3 5699 736e
18 7b38 7849 cd35 c68e de88 66ab b1d8 3d63
19 8bbe 3aef 1a2f 1e2a 7cee b5ae 4e2a 7b42
20 9183 bf52 31db 032d 3663 af97 34a8 68bd
21 336b ff86 2d0c each dddb ee5d c8c7 8d16
22 b387 8b6d a1f8 bdf3 fa32 c5d0 02dc ba21
23 c6a6 ec61 381f 6078 68e8 ab39 a29d 6be4
24 7ddd 62e0 70ef f0d4 716d 4354 dfba c1ef
```

movieData:





## 5 Prediction Results

RSME:

优化前:

$RMSE(result\_1\_1) = 1.027915812345421(\text{baseline})$

$RMSE(result\_1\_movie) = 0.9336212(\text{item-item})$

$RMSE(result\_1\_user) = 1.0282677(\text{user-user})$

$RMSE(result\_1\_3) = 0.89902134(\text{Incorporating Temporal Dynamics})$

优化后:

$RMSE(result\_kMeans) = 0.9345669189747243(\text{based on user})$

$RMSE(result\_SVD) = 0.934418131$

结果分析:

kMeans: 经过  $k=10, k=20, k=100$  的检验,  $k$  取 20 时效果最好; 可能原因是当  $k$  取 10 时由于聚类个数较少, 因此同一聚类中的元素相似度可能并没有很高; 而当  $k$  取 100 时, 进行聚类所花的时间较大, 但产生的效果不明显, 因此得不偿失。综合来说  $k$  取 20 的时候效果较好, 时间较快而且 RSME 的值也较高。

kMeans 的结果:

kMeans 的结果比先前的差, 因为之前是直接对所有的用户或电影按相似度排名, cluster 之后, 虽然速度提升了很多, 但是由于 cluster 的中间误差, 导致相似度排名的结果也产生偏差。

这里 item-based 的结果比 user-based 的结果要好。因为 user 实际上会涉及 sparsity 和 scalability 的问题, 并且一般的系统也更倾向于选用 item-based. 另外, 在本次项目中没有涉及到数据量的变化。经过小范围的数据量测试变化表明: 当用户量很少的时候, user-based CF 效率很高, 而当数据量增加之后, 它的表现会下降; 而对于 item-based CF 而言, 虽然刚开始用户量少的时候表现较差, 但是当数据量增多之后会有良好的表现。

SVD 的结果:

由于用户对很多电影是没有评分的, 所以在评分矩阵中, 好多项都是 0, 因此可以采用 SVD 进行降维压缩矩阵可以删除不重要的或噪音用户和项目, 降低用户-项目评分矩阵的维度这样在维度小的多的数据情况下能够快速得到结果, 并且由于降维后的矩阵可以以很小的误差来代表原始矩阵, 所以使用 SVD 降维后的矩阵进行预测效果也很好。

## 6 小组分工

刘林青: 负责 Recommendation (2) 部分整体架构的代码及该部分文档撰写

黄安娜: 负责 kMeans (3) 和 SVD (4) 部分的优化设计及该部分文档撰写