



南开大学  
Nankai University

南 开 大 学

计 算 机 学 院

大数据计算和应用

---

## 推荐系统实验报告

---

姓名：张丛

学号：2113662

专业：信息安全

指导教师：杨征路

2024 年 6 月 12 日

## 摘要

关键字：推荐系统；协同过滤；矩阵分解；SVD

## 目录

<b>一、 实验内容</b>	<b>1</b>
(一) 实验目的 . . . . .	1
(二) 实验要求 . . . . .	1
<b>二、 推荐系统概述</b>	<b>1</b>
(一) 什么是推荐系统 . . . . .	1
(二) 推荐系统分类 . . . . .	1
1. 基于内容的推荐系统 . . . . .	1
2. 协同过滤推荐系统 . . . . .	1
3. 混合推荐系统 . . . . .	1
(三) 推荐系统核心算法 . . . . .	2
1. 矩阵分解 . . . . .	2
2. 深度学习 . . . . .	2
3. 图神经网络 . . . . .	2
<b>三、 数据集说明</b>	<b>2</b>
(一) 数据集格式 . . . . .	2
(二) 数据集统计信息 . . . . .	3
<b>四、 实验过程</b>	<b>3</b>
(一) 实验思路：基于矩阵分解的协同过滤推荐算法 . . . . .	3
(二) 数据预处理 . . . . .	4
1. 索引的映射 . . . . .	4
2. 读取数据集数据 . . . . .	5
3. 获取均值与偏差 . . . . .	5
(三) SVD 算法流程 . . . . .	6
1. 初始化 . . . . .	6
2. 计算预测评分 . . . . .	7
3. 优化目标函数 . . . . .	8
4. 梯度下降更新参数 . . . . .	8
5. 重复训练 . . . . .	9
6. 测试集的预测评分 . . . . .	10
<b>五、 实验结果及分析</b>	<b>11</b>
(一) 预测评分 . . . . .	11
(二) 时间空间消耗 . . . . .	11
1. 时间消耗 . . . . .	11
2. 空间消耗 . . . . .	11

（三） 利用物品特征优化算法的可能 . . . . .	11
1. 特征提取 . . . . .	11
2. 特征整合 . . . . .	12
<b>六、 总结</b>	<b>12</b>

## 一、 实验内容

### (一) 实验目的

在本项目中，需要报告 Test.txt 文件中未知用户-物品对  $(u, i)$  的预测评分。

### (二) 实验要求

报告应包括但不限于以下内容：

- 数据集的基本统计信息（例如，用户数量、评分数量、物品数量等）；
- 算法细节；
- 推荐算法的实验结果（RMSE、训练时间、空间消耗）；
- 算法的理论分析或/和实验分析。

## 二、 推荐系统概述

### (一) 什么是推荐系统

推荐系统是一种基于用户行为和偏好，自动向用户推荐可能感兴趣的物品（如商品、电影、音乐等）的技术。它们在现代互联网应用中扮演着至关重要的角色，为用户提供个性化的体验，同时也帮助平台提升用户粘性和销售额。

### (二) 推荐系统分类

#### 1. 基于内容的推荐系统

基于内容的推荐系统通过分析物品的内容特征（如文本、图像、音频等）来进行推荐。

例如，新闻推荐系统会根据用户阅读过的文章内容，推荐相似主题的文章。

其优点是可以处理新物品的冷启动问题，但缺点是可能会导致推荐结果的多样性不足。

#### 2. 协同过滤推荐系统

- 基于用户的协同过滤：通过找到与当前用户兴趣相似的其他用户，推荐这些用户喜欢的物品。例如，A 用户和 C 用户有相似的兴趣，如果 C 用户喜欢某个物品，那么也会推荐给 A 用户。
- 基于物品的协同过滤：通过找到与当前物品相似的其他物品，推荐这些物品给用户。例如，如果用户喜欢某本书，则推荐与这本书相似的其他书籍。其优点是能够捕捉到用户和物品之间的复杂关系，但缺点是需要大量的用户行为数据，且数据稀疏性和冷启动问题较为突出。

#### 3. 混合推荐系统

混合推荐系统结合了多种推荐方法，以弥补单一方法的不足。

例如，Netflix 的推荐系统结合了基于内容和协同过滤的方法，以提高推荐的准确性和多样性。

### (三) 推荐系统核心算法

#### 1. 矩阵分解

矩阵分解是一种将用户-物品交互矩阵分解为低维潜在特征矩阵的方法。它通过捕捉用户和物品的隐含特征来进行推荐。

- 奇异值分解 (SVD)

SVD 将用户-物品矩阵  $R$  分解为三个矩阵  $U$ ,  $V$ ,  $W$  的乘积。其中,  $U$  和  $V$  分别表示用户和物品的特征矩阵,  $W$  是对角矩阵, 包含奇异值。

- 交替最小二乘法 (ALS)

ALS 通过交替优化用户特征矩阵  $U$  和物品特征矩阵  $V$  来最小化预测误差。具体做法是固定一个矩阵, 优化另一个矩阵, 交替进行, 直到收敛。

#### 2. 深度学习

深度学习方法通过神经网络模型自动学习用户和物品的复杂特征, 能够捕捉非线性关系和高维数据中的隐含模式。

- 神经协同过滤: 适用于需要高精度推荐的场景, 如个性化商品推荐、个性化内容推荐等。
- 卷积神经网络: 电商平台可以通过 CNN 分析商品图片, 推荐相似风格的商品。
- 循环神经网络: 适用于时间序列推荐, 如根据用户的浏览历史推荐接下来的内容。

#### 3. 图神经网络

图神经网络通过图结构来建模用户和物品之间的复杂关系, 能够捕捉高阶连接信息。

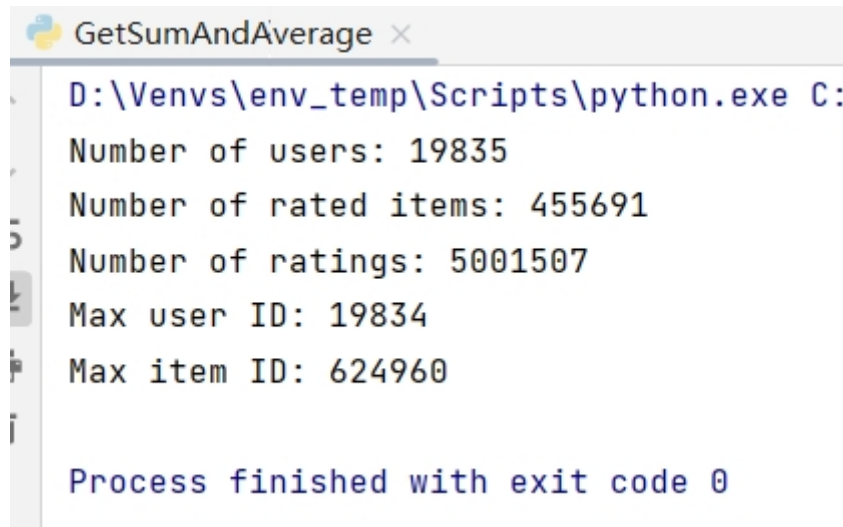
## 三、数据集说明

### (一) 数据集格式

- train.txt: 若干用户对不同物品的评分  
    <user id>|<numbers of rating items>  
    <item id> <score>
- test.txt: 若干用户索引以及每个用户对应的 6 个物品索引 (待评分)  
    <user id>|<numbers of rating items>  
    <item id>
- itemAttribute.txt: 每个物品的两个属性值 (存在 None)  
    <item id>|<attribute\_1>|<attribute\_2>

## (二) 数据集统计信息

编写程序统计用户数量、物品数量、最大用户索引、最大物品索引、总评分数。如下：



```
GetSumAndAverage x
D:\Venvs\env_temp\Scripts\python.exe C:
Number of users: 19835
Number of rated items: 455691
Number of ratings: 5001507
Max user ID: 19834
Max item ID: 624960

Process finished with exit code 0
```

图 1: 数据集统计信息

## 四、 实验过程

### (一) 实验思路：基于矩阵分解的协同过滤推荐算法

基于矩阵分解的推荐算法是协同过滤推荐系统中的一种重要方法，它主要用于处理用户-物品评分矩阵的稀疏性问题。此次实验使用 SVD 模型。

SVD（奇异值分解）将矩阵  $R$  分解为三个矩阵的乘积：

$$R = U\Sigma V^T$$

其中：

- $U$  是  $m \times k$  的左奇异矩阵。
- $\Sigma$  是  $k \times k$  的对角矩阵，对角线上的值为奇异值。
- $V$  是  $n \times k$  的右奇异矩阵。

在推荐系统中，我们通常不直接使用 SVD 分解，而是采用一种变种形式，即：

$$R \approx PQ^T$$

其中：

- $P$  是  $m \times k$  的用户矩阵，对应于  $U\Sigma^{1/2}$ 。
- $Q$  是  $n \times k$  的物品矩阵，对应于  $V\Sigma^{1/2}$ 。

基于奇异值分解（SVD）的推荐算法的实现流程如下：

#### 1. 数据准备

- 数据收集，数据预处理

#### 2. 模型初始化

- 矩阵分解初始化：初始化用户特征矩阵和物品特征矩阵的维度。
- 随机初始化矩阵：用户特征矩阵和物品特征矩阵通常用小随机值初始化。

### 3. 模型训练

- 计算全局平均分：计算所有评分的平均值，这将用于预测评分时的偏置校正。
- 计算偏置：计算每个用户和每个物品的偏置（用户偏置和物品偏置）。
- 矩阵分解：使用梯度下降法或交替最小二乘法（ALS）来优化用户特征矩阵和物品特征矩阵，使得其乘积逼近原始评分矩阵。

### 4. 预测评分

- 评分预测公式：使用分解后的矩阵计算预测评分：

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

其中， $\mu$  是全局平均评分， $b_u$  是用户偏置， $b_i$  是物品偏置， $q_i$  是物品特征向量， $p_u$  是用户特征向量。

### 5. 模型评估

- 损失函数：计算训练损失和验证损失，通常使用均方误差（MSE）或均方根误差（RMSE）来评估模型的准确性。
- 正则化：为了防止过拟合，在损失函数中加入正则化项，对用户特征矩阵、物品特征矩阵、用户偏置和物品偏置进行惩罚。

### 6. 模型优化

- 梯度下降：通过反向传播更新用户和物品特征矩阵。

### 7. 结果生成

- 推荐物品：根据用户特征矩阵和物品特征矩阵的乘积，生成用户对物品的评分预测，并推荐评分最高的物品。

### 8. 模型保存与加载

## （二） 数据预处理

### 1. 索引的映射

在数据集基本信息统计可知，用户的总数是 19835，而最大用户索引是 19834，故而可知用户的索引是连续不间断的。

但是，物品的最大索引 624960 却大于物品数 455691，可知物品索引不连续。

为了提高计算效率、统一数据格式以及为了更方便的访问，我们将物品 ID 映射到一个更紧凑的整数索引。

在程序中，只需要逐行将物品放入集合，然后对集合内的物品进行索引的映射即可。

代码如下：

```
1 def allude_index(train_path):
2     item_set = set() # 物品的集合
3     with open(train_path, 'r') as f:
4         while True:
5             line = f.readline()
6             if not line:
7                 break
8             _, num = map(int, line.strip().split('|')) # 用户ID, 物品数量
9             for _ in range(num):
10                 line = f.readline() # 逐行读取
11                 item_id, _ = map(int, line.strip().split()) # 获得物品ID
12                 item_set.add(item_id) # 添加到集合
```

```
13
14 item_index = {node: idx for idx, node in enumerate(sorted(item_set))} #
    为每个物品ID分配一个唯一的索引
15 return item_index
```

## 2. 读取数据集数据

将预处理后的数据保存为.pkl 文件（字典形式），可以清晰地组织数据，同时在后续使用时直接加载，提升查询效率。

以读取训练集数据为例：

```
1 def get_train_data(train_path, item_index):
2     """
3     从训练数据文件中读取数据，返回以用户ID为键，值为[物品ID，评分]列表的字典
        和以物品ID为键，值为[用户ID，评分]列表的字典。
4
5     :param train_path: 训练数据文件路径
6     :param item_index: 物品ID映射索引的字典
7     :return: 以用户ID为键，值为[物品ID，评分]列表的字典和以物品ID为键，值为[
        用户ID，评分]列表的字典
8     """
9     data_user, data_item = defaultdict(list), defaultdict(list)
10    with open(train_path, 'r') as f:
11        while True:
12            line = f.readline()
13            if not line:
14                break
15            user_id, num = map(int, line.strip().split('|'))
16            # 逐行读取
17            for _ in range(num):
18                line = f.readline()
19                item_id, score = line.strip().split()
20                item_id, score = int(item_id), float(score)
21                score = score / 10
22                data_user[user_id].append([item_index[item_id], score])
23                data_item[item_index[item_id]].append([user_id, score])
24    return data_user, data_item
```

字典格式如下：

- data\_user: 用户 ID: [物品 ID, 评分]
- data\_item: 物品 ID: [用户 ID, 评分]
- data\_attribute: 物品 ID: [属性 1, 属性 2]

## 3. 获取均值与偏差

后续我们需要使用评分均值和偏差来改进评分预测，故而对获取的数据进行均值和偏差的计算。



代码如下：

```

1 def get_bias(train_data_user, train_data_item):
2     """
3     计算评分均值和用户、物品偏差。
4
5     :param train_data_user: 以用户ID为键，[物品ID, 评分]为值的字典
6     :param train_data_item: 以物品ID为键，[用户ID, 评分]为值的字典
7     :return: 全局评分均值，用户偏差，物品偏差
8     """
9     miu = 0.0
10    bx = np.zeros(user_num, dtype=np.float64) # 初始化
11    bi = np.zeros(item_num, dtype=np.float64)
12
13    # 计算用户偏差和全局评分均值
14    for user_id in train_data_user:
15        sum = 0.0
16        for item_id, score in train_data_user[user_id]:
17            miu += score # 累加全局评分
18            sum += score # 累加用户评分
19        bx[user_id] = sum / len(train_data_user[user_id]) # 计算当前用户的平均评分
20    miu /= ratings_num # 计算全局评分均值
21
22    # 计算物品偏差
23    for item_id in train_data_item:
24        sum = 0.0
25        for user_id, score in train_data_item[item_id]:
26            sum += score # 累加物品评分
27        bi[item_id] = sum / len(train_data_item[item_id]) # 计算当前物品的平均评分
28
29    bx -= miu # 减去全局评分均值，得到用户偏差
30    bi -= miu # 减去全局评分均值，得到物品偏差
31    return miu, bx, bi

```

### (三) SVD 算法流程

#### 1. 初始化

初始化一个 SVD 模型的各个参数和数据，包括加载已有的偏置和索引数据，计算全局平均评分，随机初始化用户和物品的潜在特征矩阵。

```

1 def __init__(self, model_path='./model',
2             data_path='./data/train_user.pkl',
3             lr=5e-3,
4             lamda1=1e-2,
5             lamda2=1e-2,
6             lamda3=1e-2,

```

```

7         lamda4=1e-2, factor=50):
8
9     #加载偏置
10    self.bx = load_pkl(bx_pkl) # 用户偏置
11    self.bi = load_pkl(bi_pkl) # 物品偏置
12
13    #设置模型参数
14    self.lr = lr # 学习率
15    self.lamda1 = lamda1 # 正则化系数
16    self.lamda2 = lamda2
17    self.lamda3 = lamda3
18    self.lamda4 = lamda4
19    self.factor = factor # 隐向量维度
20
21    #加载索引和数据
22    self.idx = load_pkl(idx_pkl)
23    self.train_user_data = load_pkl(data_path)
24    self.train_data, self.valid_data = split_data(self.train_user_data)
25    self.test_data = load_pkl(test_pkl)
26
27    #计算全局评分
28    self.globalmean = self.get_globalmean()
29
30    #初始化用户和物品矩阵
31    self.Q = np.random.normal(0, 0.1, (self.factor, len(self.bi))) # 矩阵Q(
        物品)
32    self.P = np.random.normal(0, 0.1, (self.factor, len(self.bx))) # 矩阵P(
        用户)
33
34    #模型路径
35    self.model_path = model_path

```

## 2. 计算预测评分

对于用户  $i$  和物品  $j$ , 预测评分公式为:

$$\hat{r}_{ij} = \mu + bx_i + bi_j + P_i^T Q_j$$

- 其中,  $P_i$  是用户  $i$  的隐向量表示,  $Q_j$  是物品  $j$  的隐向量表示。

```

1 def predict(self, user_id, item_id):
2     pre_score = self.globalmean + \
3         self.bx[user_id] + \
4         self.bi[item_id] + \
5         np.dot(self.P[:, user_id], self.Q[:, item_id])
6     return pre_score

```

### 3. 优化目标函数

目标是最小化预测评分与实际评分之间的均方误差 (MSE)，同时加上正则化项以防止过拟合：

$$\min \sum_{(i,j) \in R} (r_{ij} - \hat{r}_{ij})^2 + \lambda(\|P\|^2 + \|Q\|^2 + \|bx\|^2 + \|bi\|^2)$$

- 其中， $\lambda$  是正则化参数。

```

1 def loss(self, is_valid=False):
2     """
3     计算模型的损失函数。
4     """
5     loss, count = 0.0, 0
6     data = self.valid_data if is_valid else self.train_data
7
8     #误差的平方和
9     for user_id, items in data.items():
10         for item_id, score in items:
11             loss += (score - self.predict(user_id, item_id)) ** 2
12             count += 1
13
14     #添加正则化项
15     if not is_valid:
16         loss += self.lamda1 * np.sum(self.P ** 2)
17         loss += self.lamda2 * np.sum(self.Q ** 2)
18         loss += self.lamda3 * np.sum(self.bx ** 2)
19         loss += self.lamda4 * np.sum(self.bi ** 2)
20
21     #计算平均损失
22     loss /= count
23     return loss

```

### 4. 梯度下降更新参数

在迭代过程中，通过计算误差来得到损失函数，然后根据损失函数对每个参数（用户偏置、物品偏置、用户隐向量、物品隐向量）的梯度来更新参数。

- 对于每一个评分  $r_{ij}$ ，计算预测误差：

$$e_{ij} = r_{ij} - \hat{r}_{ij}$$

- 更新用户和物品的偏置，以及矩阵  $P$  和  $Q$ ：

$$bx_i \leftarrow bx_i + \gamma(e_{ij} - \lambda_3 bx_i)$$

$$bi_j \leftarrow bi_j + \gamma(e_{ij} - \lambda_4 bi_j)$$

$$P_i \leftarrow P_i + \gamma(e_{ij} Q_j - \lambda_1 P_i)$$

$$Q_j \leftarrow Q_j + \gamma(e_{ij} P_i - \lambda_2 Q_j)$$

- 其中， $\gamma$  是学习率。

```

1 def train(self, epochs=10, save=False, load=False):
2     """
3     训练模型。
4     """
5     if load:
6         self.load_weight()
7         print('Start training...')
8
9     for epoch in range(epochs):
10        for user_id, items in tqdm(self.train_data.items(), desc=f'Epoch {
11            epoch + 1}'):
12            for item_id, score in items:
13                error = score - self.predict(user_id, item_id)
14
15                # 更新用户和物品的偏置
16                self.bx[user_id] += self.lr * (error - self.lamda3 * self.bx[
17                    user_id])
18                self.bi[item_id] += self.lr * (error - self.lamda4 * self.bi[
19                    item_id])
20
21                # 更新用户和物品的隐向量
22                self.P[:, user_id] += self.lr * (error * self.Q[:, item_id] -
23                    self.lamda1 * self.P[:, user_id])
24                self.Q[:, item_id] += self.lr * (error * self.P[:, user_id] -
25                    self.lamda2 * self.Q[:, item_id])
26
27            # 每个epoch的训练和验证集损失
28            print(f'Epoch {epoch + 1} train loss: {self.loss():.6f} valid loss: {
29                self.loss(is_valid=True):.6f}')
30
31        print('Training finished.')
32
33    if save:
34        self.save_weight()

```

## 5. 重复训练

重复上述步骤，直到达到预定的训练轮数或损失函数收敛。

最后一轮训练完成后，计算 rmse 值衡量模型在训练数据上的拟合程度。代码如下：

```

1 def rmse(self):
2     """
3     计算均方根误差。
4     """
5     rmse, count = 0.0, 0
6     for user_id, items in self.train_user_data.items():
7         for item_id, score in items:
8             rmse += (score - self.predict(user_id, item_id)) ** 2

```

```

9         count += 1
10    rmse /= count
11    rmse = np.sqrt(rmse)
12    return rmse

```

计算结果如下：

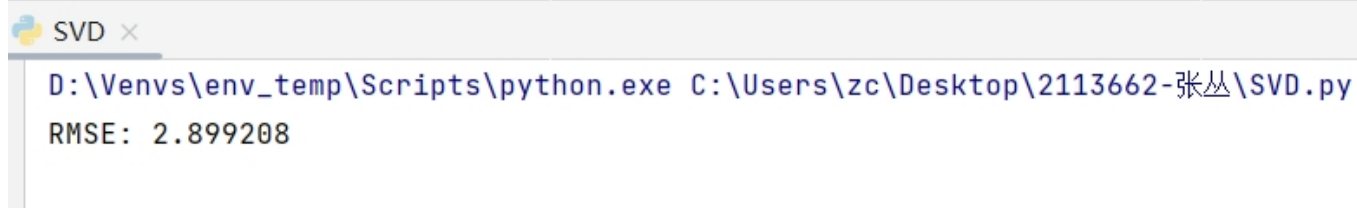


图 2: rmse 值

可见拟合较好。

## 6. 测试集的预测评分

对测试数据中的每个用户和物品对进行评分预测，并将结果存储在字典中，最终返回所有预测评分结果并保存在.txt 文件中。

```

1 def test(self, write_path='./result/result.txt', load=True):
2     # 存储预测评分结果
3     predict_score = defaultdict(list)
4
5     # 遍历测试数据中的每个用户和对应的物品列表
6     for user_id, item_list in self.test_data.items():
7         for item_id in item_list:
8             if item_id not in self.idx: # 如果物品不在索引中,用平均评分作为
                默认值
9                 pre_score = self.globalmean * 10
10            else:
11                new_id = self.idx[item_id] # 获取物品的新索引
12                pre_score = self.predict(user_id, new_id) * 10 # 预测评分
13
14            # 将评分限制在0到100之间
15            if pre_score > 100.0:
16                pre_score = 100.0
17            elif pre_score < 0.0:
18                pre_score = 0.0
19
20            # 将预测结果加入字典
21            predict_score[user_id].append((item_id, pre_score))
22
23    print('Testing finished.')
24    return predict_score

```

## 五、实验结果及分析

### (一) 预测评分

用户 ID	物品 ID	评分
0	127640	70.76322402424361
0	192496	80.89995910199812
0	147073	73.76623674579224
0	70896	94.39158373829105
0	578821	83.61021151947662
0	522229	28.440788941816166
1	507010	92.53585190030523
1	55629	92.23167505688558
1	453396	89.86433845645563
1	137915	90.51832322025412
1	261386	89.59107717060621
1	34525	90.91811610929489

### (二) 时间空间消耗

#### 1. 时间消耗

对于实验中矩阵分解模型 (SVD)，每轮训练实现大约一分钟左右。

```
Epoch 10 train loss: 1.270021 valid loss:
Training finished.
saving weight...
done.
Training time: 726.8152215480804 seconds
```

图 3: 训练时间

#### 2. 空间消耗

整个项目大约占据磁盘空间 550MB。其中，原始数据和中间数据占一半以上。

### (三) 利用物品特征优化算法的可能

此次实验中提供的 ItemAttribute.txt 给出了物品的两个特征（属性）值。

考虑到物品的特征，特征提取和特征整合是优化推荐算法的关键步骤。

#### 1. 特征提取

- 独热编码：将属性信息转换为独热编码。对于每个属性，创建一个新的二进制特征，表示物品是否具有该属性。

- 词袋模型：将属性信息看作是一组词汇，使用词袋模型将属性信息转换为特征向量。每个属性可以被表示为一个向量，向量的每个元素对应一个词汇，表示属性是否包含该词汇。
- 词嵌入：使用词嵌入技术将属性信息转换为连续的特征向量。这种方法可以捕捉到属性之间的语义关系，通常在深度学习模型中使用。

## 2. 特征整合

- 与原始特征合并：将提取的特征与原始的用户-物品评分矩阵进行合并。可以将提取的特征与用户特征或物品特征进行拼接，形成新的特征矩阵。
- 特征相乘：将提取的特征与用户或物品的原始特征进行相乘。例如，将用户的特征向量与物品的属性特征向量相乘，得到用户对物品的预测评分。
- 特征组合：将不同的特征进行组合，生成新的特征。例如，可以将用户的属性特征与物品的属性特征进行组合，生成新的特征向量。

# 六、 总结

此次实验，主要实现了对 Test.txt 文件中未知用户-物品对 (u, i) 的预测评分，运用的理论知识是推荐系统，更具体的说是基于梯度下降法来训练 SVD 模型。

在课堂上，杨老师对推荐系统的讲解由浅入深，层层递进，让我一度以为我对推荐系统有了一定的习得。然而真正的开始做项目时，却又觉得自己好像什么都差一点。从如何处理数据，到如何计算所需数据，到如何训练模型，都经历了很多困难，让我深刻认识到了自己的不足。

原因是多方面的。比如，python 和深度学习相关的知识储备不够，这与平时的积累有关。又比如是理论知识和实践没有很好的结合，在课上学到的很好很妙的算法，自己却实现不出来，这是很悲伤的事情。

总之，此次实验，或者说大数据计算和应用这门课程，让我受益匪浅。

在今后的学习中，我也定然不忘老师的教导，努力向前。