



南開大學
Nankai University

南 開 大 學

計 算 機 學 院
大數據計算及應用實驗報告

推薦系統編程大作業

孟啟軒 2212452

趙熙 2210917

年級：2022 級

指導教師：楊征路

2025 年 6 月 8 日

目录

一、 实验概述 1

二、 实验要求 1

 (一) 任务目标 1

 (二) 数据集说明 1

 (三) 报告内容要求 1

三、 算法设计与实现 1

 (一) 算法原理 1

 (二) 数据集处理 2

 (三) 算法实现 3

 1. 初始化参数 3

 2. 随机梯度下降 (SGD) 3

 3. 训练过程 4

 4. 训练结果可视化 4

四、 实验结果 4

 (一) 数据集的基本统计信息 4

 (二) 训练、验证、预测过程 5

 (三) 推荐算法实验结果 5

 1. 评分预测结果 5

 2. RMSE 6

 3. 训练时间和空间消耗 7

五、 总结与感想 7

一、 实验概述

推荐系统是一种广泛应用于电子商务、社交媒体、视频流媒体等在线平台的技术，其核心目标是通过分析用户的兴趣、行为和偏好，为其提供个性化的内容、产品或服务推荐。推荐系统的基本原理是基于大量用户数据构建用户模型，并结合物品的特征与属性，通过匹配计算生成个性化推荐。在本实验中，我们采用 SVD（奇异值分解）算法来实现推荐系统。SVD 是一种常用的矩阵分解技术，它将一个矩阵 A 分解为 U 、 Σ 和 V^T 的乘积，其中 U 和 V^T 是正交矩阵， Σ 是对角矩阵。通过对用户-物品评分矩阵进行 SVD 分解，可以有效降低数据维度，提取用户和物品的潜在特征，并利用这些特征对缺失评分进行预测。本实验首先对数据集进行了全面分析，获取用户数量、商品数量、评分数量、最高评分、最低评分及评分均值等基础信息。随后，基于 SVD 算法构建模型并使用 SGD 算法对模型进行训练，并利用划分出的验证集计算 RMSE（均方根误差）以评估模型效果。最后，利用训练好的模型对测试集中的评分进行预测，并将结果保存至 Result.txt 文件中。

二、 实验要求

（一） 任务目标

根据提供的 train.txt 训练集，构建评分预测模型，对 test.txt 中的未知 (u, i) 对进行评分预测，并按照 ResultForm.txt 格式输出结果。

（二） 数据集说明

- train.txt：训练数据，包含用户-项目-评分三元组；
- test.txt：测试数据，仅包含用户-项目对，需预测评分；
- ResultForm.txt：结果格式模板；
- dataformatexplain.txt：数据格式说明文件。

（三） 报告内容要求

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

三、 算法设计与实现

（一） 算法原理

本次实验中，我们采用 SVD 推荐算法进行个性化推荐。SVD (Singular Value Decomposition, 奇异值分解) 是一种基于矩阵分解的技术，广泛应用于推荐系统中。其基本思想是通过将用户-物品评分矩阵分解为用户特征矩阵、物品特征矩阵和奇异值矩阵，来发现用户和物品之间潜在的关系。

具体来说，给定一个用户-物品评分矩阵 R ，SVD 会将其分解为三个矩阵的乘积：

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

其中： U 是一个 $m \times k$ 的矩阵，表示用户的潜在特征； Σ 是一个 $k \times k$ 的对角矩阵，包含了奇异值，表示各个潜在特征的重要性； V^T 是一个 $k \times n$ 的矩阵，表示物品的潜在特征。

通过保留最大的 k 个奇异值，我们可以得到对用户和物品之间潜在关系的低秩近似，从而有效减少数据的维度，提升计算效率并降低过拟合的风险。

在推荐系统中，SVD 通过对评分矩阵进行分解，能够捕捉到用户的潜在偏好和物品的特性，进而为用户推荐未评分的物品。具体操作包括：

- 通过训练集构建用户-物品评分矩阵；
- 使用 SVD 分解该矩阵，获得用户和物品的潜在特征；
- 对未评分物品进行预测，生成推荐列表。

SVD 方法的优势在于它能够捕捉到用户和物品之间的隐性关系，特别是当评分矩阵稀疏时，SVD 依然能够有效地从数据中提取有用的信息。

为了进一步提高推荐的准确性，SVD 可以与正则化方法相结合，如 SVD++，来考虑隐式反馈信息（例如用户对物品的浏览、点击行为）以及避免过拟合。

在本次实验中，我们使用了 SVD 算法来进行个性化推荐。我们从训练集构建了用户-物品评分矩阵，并采用 SVD 对该矩阵进行分解。接着，我们通过预测用户对未评分物品的评分来生成推荐列表，最终使用 RMSE（Root Mean Square Error，均方根误差）来评估推荐的效果。

具体来说，实验中使用了如下步骤：

- 数据预处理：加载并清洗训练数据；
- 矩阵分解：使用 SVD 对评分矩阵进行分解；
- 预测评分：根据分解后的用户和物品特征，预测用户未评分物品的评分；
- 结果评估：通过 RMSE 计算预测评分与实际评分之间的误差。

通过这些步骤，我们可以为用户推荐个性化的物品，并评估推荐系统的性能。

（二） 数据集处理

本次实验所使用的训练数据集存储于 `train.txt` 文件中，文件格式如下：

- 每个用户的数据记录以 ‘`|`’ 分隔，第一部分为用户 ID，第二部分为评分数量。
- 随后紧跟的若干行包含该用户对物品的评分，格式为 ‘物品 ID 评分’，每行一条评分记录。

设计了 `parse_train_file` 函数，获取数据集中的基本统计信息，其主要功能如下：

1. **初始化数据结构：**创建用户集合、物品集合，并定义变量 `score_num`、`score_sum`、`max_score` 和 `min_score` 分别用于统计评分数量、评分总和、最高评分和最低评分。
2. **文件解析：**按行读取数据集。对于每个用户数据块，首先获取用户 ID 和评分数量，然后依次读取该用户的评分数据，逐行解析并更新上述统计信息。
3. **统计数据更新：**

- `user_set.add(userid)` 用于统计用户数。
- `item_set.add(itemid)` 用于统计物品数。
- `score_sum += score` 用于累加评分。
- `max_score` 和 `min_score` 用于追踪评分最值。

4. **返回统计结果：**在文件读取结束后，将统计数据封装为字典格式进行返回。

(三) 算法实现

1. 初始化参数

在算法的初始化阶段，我们首先定义了模型的基本参数，包括用户偏置、物品偏置、用户和物品的特征矩阵等。这些矩阵将在训练过程中通过梯度下降算法不断优化。

- **用户偏置 (UserBias)：**初始化为零，表示用户的偏置项。
- **物品偏置 (ItemBias)：**初始化为零，表示物品的偏置项。
- **用户特征矩阵 (UserCharacteristicMatrix)：**通过正态分布随机初始化，矩阵大小为用户数目与特征维度的乘积。
- **物品特征矩阵 (ItemCharacteristicMatrix)：**同样通过正态分布随机初始化，矩阵大小为物品数目与特征维度的乘积。

2. 随机梯度下降 (SGD)

训练过程采用随机梯度下降算法优化用户和物品的潜在特征矩阵以及偏置项。在每个迭代周期内，系统会计算每个用户对物品的预测评分，并根据实际评分与预测评分之间的误差来更新模型参数。

具体过程如下：

- **预测评分计算：**对于每个用户-物品对 ('uid', 'iid')，通过计算用户和物品的特征矩阵的内积，结合用户偏置、物品偏置以及整体平均评分，得到预测评分。

```
1 multi = np.dot(self.ItemCharacteristicMatrix[iid], self.
    UserCharacteristicMatrix[uid]) # 计算内积
2 predict = mean_rating + self.UserBias[uid] + self.ItemBias[iid] + multi
    # 预测评分
3 predict = min(max(predict, 0), 100) # 限制预测评分在0和100之间
```

- **误差计算：**预测评分与真实评分之间的差异即为误差 ('eui')，误差被用于更新模型参数。

```
1 eui = rating - predict # 计算误差
2 sum_eui_squared += eui ** 2 # 计算误差平方和
```

- **参数更新：**使用误差 'eui' 更新用户偏置、物品偏置和特征矩阵。

```
1 self.UserBias[uid] += lr * (eui - regularization * self.UserBias[uid]) #
    更新用户偏置
2 self.ItemBias[iid] += lr * (eui - regularization * self.ItemBias[iid]) #
    更新物品偏置
```

```

3     for i in range(self.features):
4         temp = self.UserCharacteristicMatrix[uid, i]
5         self.UserCharacteristicMatrix[uid, i] += lr * (eui * self.
            ItemCharacteristicMatrix[iid, i] - regularization * temp) # 更新
            用户特征
6         self.ItemCharacteristicMatrix[iid, i] += lr * (eui * temp -
            regularization * self.ItemCharacteristicMatrix[iid, i]) # 更新物
            品特征

```

3. 训练过程

在每轮迭代中，我们会更新用户偏置、物品偏置和特征矩阵的元素。每次更新后，通过计算当前的 RMSE（均方根误差）来评估训练效果，逐步优化模型参数。

• 梯度更新公式：

– 用户偏置更新：

$$\text{UserBias}[\text{uid}] += \text{lr} \times (\text{eui} - \text{regularization} \times \text{UserBias}[\text{uid}])$$

– 物品偏置更新：

$$\text{ItemBias}[\text{iid}] += \text{lr} \times (\text{eui} - \text{regularization} \times \text{ItemBias}[\text{iid}])$$

– 用户特征矩阵更新：

$$\text{UserCharacteristicMatrix}[\text{uid}, \text{i}] += \text{lr} \times (\text{eui} \times \text{ItemCharacteristicMatrix}[\text{iid}, \text{i}] - \text{regularization} \times \text{temp})$$

– 物品特征矩阵更新：

$$\text{ItemCharacteristicMatrix}[\text{iid}, \text{i}] += \text{lr} \times (\text{eui} \times \text{temp} - \text{regularization} \times \text{ItemCharacteristicMatrix}[\text{iid}, \text{i}])$$

- **RMSE 计算：**每轮训练结束后，计算当前模型的 RMSE 值，评估训练效果，并绘制训练误差曲线。

4. 训练结果可视化

在训练完成后，我们绘制训练过程中每轮的 RMSE 曲线，以可视化训练的收敛情况。

- **训练误差曲线：**通过 matplotlib 绘制训练误差图，观察模型在训练过程中的表现。

训练过程通过多轮迭代，优化用户和物品的特征矩阵，最终生成一个能够对用户未评分物品进行预测的推荐模型。

四、 实验结果

(一) 数据集的基本统计信息

图 1 是数据集分析的统计结果：

```
用户数量: 598  
商品数量: 9077  
评分数量: 90854  
最高评分: 100.0  
最低评分: 10.0  
评分均值: 69.88211856384969
```

图 1: 数据集分析

(二) 训练、验证、预测过程

图 2 训练过程，验证过程以及预测过程：

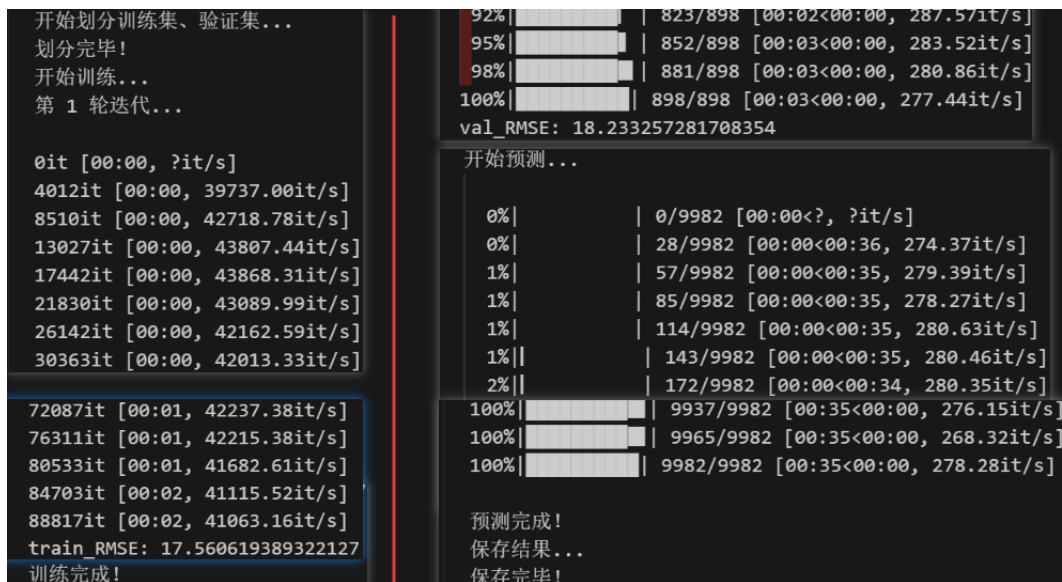
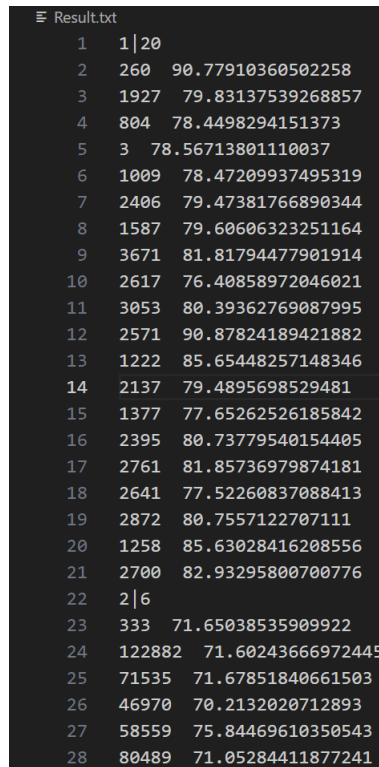


图 2: 训练验证预测过程

(三) 推荐算法实验结果

1. 评分预测结果

图 3 是得到的部分预测结果：



```

Result.txt
1 1|20
2 260 90.77910360502258
3 1927 79.83137539268857
4 804 78.4498294151373
5 3 78.56713801110037
6 1009 78.47209937495319
7 2406 79.47381766890344
8 1587 79.60606323251164
9 3671 81.81794477901914
10 2617 76.40858972046021
11 3053 80.39362769087995
12 2571 90.87824189421882
13 1222 85.65448257148346
14 2137 79.4895698529481
15 1377 77.65262526185842
16 2395 80.73779540154405
17 2761 81.85736979874181
18 2641 77.52260837088413
19 2872 80.7557122707111
20 1258 85.63028416208556
21 2700 82.93295800700776
22 2|6
23 333 71.65038535909922
24 122882 71.60243666972445
25 71535 71.67851840661503
26 46970 70.2132020712893
27 58559 75.84469610350543
28 80489 71.05284411877241

```

图 3: Result.txt 部分结果

2. RMSE

为了衡量模型预测结果的准确性，实验中引入了均方根误差（RMSE）作为主要评价指标，对训练集 `train.txt` 进行 RMSE 的计算。均方根误差用于度量预测评分与真实评分之间的偏差，其计算公式如下：

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (1)$$

其中：

- n ：样本数量；
- y_i ：真实评分；
- \hat{y}_i ：模型预测评分。

RMSE 的取值范围为 $[0, +\infty)$ 。值越小，表示模型预测结果与真实评分越接近；值越大，说明预测偏差较大。

RMSE 在模型训练过程中的变化趋势分析图 4 展示了模型在训练过程中 RMSE 的变化趋势。可以观察到：

- 在初始训练阶段（1-5 轮），RMSE 降低幅度较大，表明模型快速拟合训练数据，预测误差迅速减少。
- 在中期训练阶段（5-10 轮），RMSE 下降趋缓，模型逐步收敛，学习速率降低。
- 在后期训练阶段（10-20 轮），RMSE 基本保持稳定，说明模型已经达到最佳拟合状态，进一步训练对 RMSE 改进效果有限。

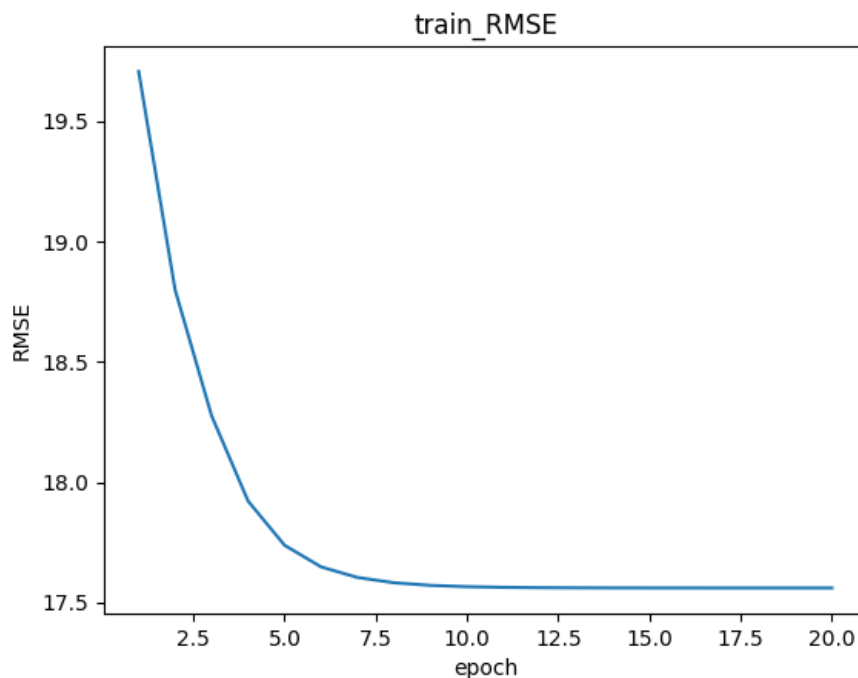


图 4: RMSE

通过 RMSE 的变化趋势，可以有效判断模型是否存在欠拟合或过拟合问题，RMSE 在训练早期急剧下降且在后期趋于平稳，说明模型已成功拟合数据。

3. 训练时间和空间消耗

共花费 94.55 秒
内存消耗：103.16 MB
内存峰值：114.30 MB

图 5: 训练时间和空间消耗

结合上述数据可以看出，模型训练阶段的内存占用相对稳定，训练时间控制在合理范围内，未出现显著的内存泄漏现象。

五、 总结与感想

在本次实验中，我们实现了基于奇异值分解 (SVD) 的推荐系统，并使用随机梯度下降 (SGD) 算法对模型进行训练和优化。通过本次实验，我深入理解了 SVD 在推荐系统中的应用，学习了如何通过矩阵分解捕捉用户和物品之间的潜在关系。通过初始化用户偏置、物品偏置以及用户和物品的特征矩阵，并通过多轮迭代优化这些参数，模型能够根据用户历史评分预测未评分物品的评分。在训练过程中，我们使用了均方根误差 (RMSE) 来衡量模型的预测效果，并通过可视化误差曲线观察了训练的收敛过程。尽管训练过程中模型的误差逐步减小，但仍面临着数据稀疏性和模型性能提升的挑战，未来可以通过引入隐式反馈数据或结合其他推荐算法来进一步优化模型。

此次实验让我们对 SVD 推荐算法有了更为深刻的理解，并且意识到其在面对大规模稀疏数据时的局限性。在未来的工作中，推荐系统的研究将更多地结合隐式反馈、模型融合及深度学习等新技术，以提升推荐系统的准确性和智能化水平。我们相信，随着技术的发展，推荐系统将在电商、社交平台和内容推荐等领域发挥越来越重要的作用，推动更多行业的创新与发展。

NIJU