**大数据计算与应用实验报告**

**推荐系统编程报告**

**——**基于矩阵分解MF的推荐算法

**学院(系)：网络空间安全学院**

**年级专业：20级信安法**

**学 号：2010387**

**学生姓名：迟文韬**

**指导教师：杨征路**

## 关于MF算法

基于矩阵分解（Matrix Factorization, MF）的推荐算法是一种常用的协同过滤推荐算法。该算法通过将用户-物品评分矩阵分解为两个低维度的矩阵，来捕捉用户和物品之间的隐藏特征和关系，从而进行推荐。

## 设计思路

本次实验采取的是基于矩阵分解（Matrix Factorization，MF）的推荐算法实现推荐系统。

对于MF算法，其基本思路如下：

1. 用户-物品评分矩阵：
   * 假设有一个用户-物品评分矩阵，其中每一行代表一个用户，每一列代表一个物品，矩阵的元素是用户对物品的评分。
   * 由于评分矩阵通常是稀疏的，即大部分条目为空值，MF算法的目标是通过填充这些空值来预测用户对未评分物品的喜好程度。
2. 矩阵分解：
   * MF算法的核心思想是将用户-物品评分矩阵分解为两个低维矩阵的乘积。假设矩阵的行数为m，列数为n，因子数量为k，那么矩阵分解可表示为：R ≈ U × V^T
   * U是一个m×k的用户因子矩阵，每一行表示一个用户的隐含特征表示。
   * V是一个n×k的物品因子矩阵，每一行表示一个物品的隐含特征表示。
   * 通过学习U和V，MF算法能够捕捉到用户和物品之间的关联性和隐含特征。
3. 模型训练：
   * 训练MF模型的目标是通过最小化预测评分与真实评分之间的误差来学习U和V。
   * 常见的优化目标是最小化均方误差（Mean Square Error）或交叉熵损失（Cross-Entropy Loss）等。
   * 优化算法可以采用梯度下降、随机梯度下降或其他优化方法来更新U和V的值。
4. 预测评分：
   * 训练好的MF模型可以用于预测用户对未评分物品的评分。
   * 对于一个用户i和一个物品j，可以通过用户因子矩阵U的第i行和物品因子矩阵V的第j行的点积来预测评分：r̂\_ij = U\_i · V\_j^T
5. 推荐生成：
   * 利用预测评分，可以为每个用户生成Top-N推荐列表，给用户推荐未评分但可能感兴趣的物品。

## 具体设计

### 数据准备

读取训练数据(train.txt)：读取train.txt文件，其中每一行代表一个用户的评分数据，包括用户ID和评分项的数量，接着读取相应数量的评分项，存储在train\_data字典中。

*# 1. 读取训练数据 train.txt*

train\_data = {}

with open('train.txt', 'r') as f\_train:

    for line in f\_train:

        user\_id, num\_items = line.strip().split('|')

        num\_items = int(num\_items)

        item\_ratings = {}

        for \_ in range(num\_items):

            item\_id, score = map(int, f\_train.readline().strip().split())

            item\_ratings[item\_id] = score

        train\_data[user\_id] = item\_ratings

*# 2. 拆分训练集和验证集*

validation\_data = {}

validation\_ratio = 0.1  *# 验证集占比*

for user\_id, item\_ratings in train\_data.items():

    num\_items = len(item\_ratings)

    num\_validation = int(num\_items \* validation\_ratio)

    validation\_items = random.sample(item\_ratings.items(), num\_validation)

    train\_items = [(item\_id, score) for item\_id, score in item\_ratings.items() if (item\_id, score) not in validation\_items]

    train\_data[user\_id] = dict(train\_items)

    validation\_data[user\_id] = dict(validation\_items)

### 创建稀疏用户-物品矩阵

**create\_user\_item\_matrix**函数用于创建稀疏的用户-物品矩阵。首先，确定所有用户和物品的唯一标识（users和items）。

*# 创建用户-物品矩阵*

def create\_user\_item\_matrix(data):

    users = sorted(data.keys())

    items = set()

    for item\_ratings in data.values():

        items.update(item\_ratings.keys())

    items = sorted(items)

使用**scipy.sparse.dok\_matrix**创建一个稀疏的用户-物品矩阵(user\_item\_matrix)。

    user\_item\_matrix = np.zeros((len(users), len(items)))

    for i, user\_id in enumerate(users):

        item\_ratings = data[user\_id]

        for j, item\_id in enumerate(items):

            rating = item\_ratings.get(item\_id, 0)

            user\_item\_matrix[i, j] = rating

遍历训练数据(train\_data)，将用户的评分填充到稀疏矩阵中的对应位置。

    return user\_item\_matrix, users, items

### MF模型训练

**matrix\_factorization**函数用于训练MF模型。它使用奇异值分解（Singular Value Decomposition，SVD）对稀疏的用户-物品矩阵进行分解。

在分解过程中，选择了指定数量的因子（n\_factors）来表示用户和物品的隐含特征。

分解后，得到用户矩阵(user\_matrix)和物品矩阵(item\_matrix)。

*# 训练模型*

def matrix\_factorization(train\_matrix, n\_factors=10, n\_iterations=10, learning\_rate=0.01, reg\_param=0.01, random\_seed=42):

    np.random.seed(random\_seed)

    n\_users, n\_items = train\_matrix.shape

*# 初始化用户矩阵和物品矩阵*

    user\_matrix = np.random.normal(scale=1./n\_factors, size=(n\_users, n\_factors))

    item\_matrix = np.random.normal(scale=1./n\_factors, size=(n\_items, n\_factors))

    for \_ in range(n\_iterations):

        for i in range(n\_users):

            for j in range(n\_items):

                if train\_matrix[i, j] > 0:

                    error = train\_matrix[i, j] - np.dot(user\_matrix[i, :], item\_matrix[j, :].T)

                    user\_matrix[i, :] += learning\_rate \* (2 \* error \* item\_matrix[j, :] - reg\_param \* user\_matrix[i, :])

                    item\_matrix[j, :] += learning\_rate \* (2 \* error \* user\_matrix[i, :] - reg\_param \* item\_matrix[j, :])

    return user\_matrix, item\_matrix

*# 创建训练集和验证集的用户-物品矩阵*

train\_matrix, train\_users, train\_items = create\_user\_item\_matrix(train\_data)

validation\_matrix, \_, \_ = create\_user\_item\_matrix(validation\_data)

*# 训练MF模型*

n\_factors = 10  *# 因子数量*

n\_iterations = 10  *# 迭代次数*

learning\_rate = 0.01  *# 学习率*

reg\_param = 0.01  *# 正则化参数*

user\_matrix, item\_matrix = matrix\_factorization(train\_matrix, n\_factors, n\_iterations, learning\_rate, reg\_param)

### 验证集评价指标计算

**calculate\_rmse**函数用于计算RMSE（Root Mean Square Error）作为评价指标。它比较验证集中真实评分和预测评分的差异来衡量模型的准确性。

首先，获取验证集的真实评分矩阵(validation\_matrix)和MF模型的预测评分矩阵(validation\_predicted)。

然后，通过调用**scikit-learn**库中的**mean\_squared\_error**函数计算RMSE。

*# 4. 计算验证集的 RMSE 评价指标*

def calculate\_rmse(predicted\_matrix, true\_matrix):

    non\_zero\_indices = true\_matrix.nonzero()

    true\_values = true\_matrix[non\_zero\_indices]

    predicted\_values = predicted\_matrix[non\_zero\_indices]

    rmse = np.sqrt(mean\_squared\_error(true\_values, predicted\_values))

    return rmse

validation\_predicted = np.dot(user\_matrix, item\_matrix.T)

validation\_rmse = calculate\_rmse(validation\_predicted, validation\_matrix)

print(f"RMSE (Validation): {validation\_rmse}")

### 测试集预测和输出

使用训练好的MF模型对测试集数据进行预测，并将预测结果输出到**output.txt**文件中。

遍历测试数据(test.txt)，读取用户ID和评分项的数量。

对于每个评分项，如果该物品在训练数据中存在，则在用户矩阵和物品矩阵中找到对应的索引，并计算预测评分。

将预测结果写入**output.txt**文件中。

*# 5. 进行测试集预测，并输出结果到 output.txt*

with open('test.txt', 'r') as f\_test, open('output2.txt', 'w') as f\_output:

    for line in f\_test:

        user\_id, num\_items = line.strip().split('|')

        num\_items = int(num\_items)

        f\_output.write(f"{user\_id}|{num\_items}\n")

        for \_ in range(num\_items):

            item\_id = int(f\_test.readline().strip())

            user\_index = train\_users.index(user\_id)

            item\_index = train\_items.index(item\_id)

            predicted\_score = np.dot(user\_matrix[user\_index, :], item\_matrix[item\_index, :].T)

            f\_output.write(f"{item\_id}\t{predicted\_score}\n")

## 运行结果

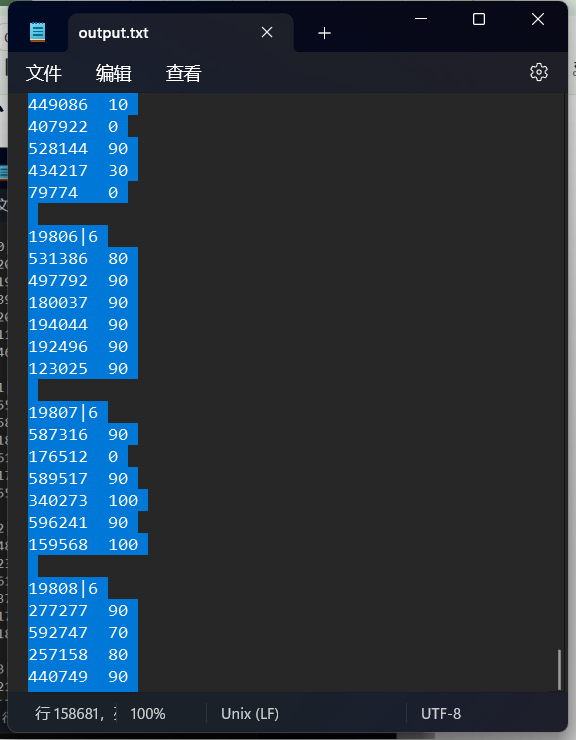
Timecost：14.5h

RMSE：29.877

number of users:19835

number of ratings:5001507

number of items:455705



## 反思总结

　　 MF算法通过矩阵分解，将用户-物品评分矩阵分解为两个低维矩阵的乘积，从而学习用户和物品的隐含特征表示。通过训练和预测评分，MF算法能够为用户生成个性化推荐。MF算法的优点是简单而且可扩展，但也存在一些挑战，如处理稀疏数据、处理冷启动问题和推荐解释性的限制等。

由于实验训练时间较长，故租用服务器进行测试