电子科技大学

计算机科学与工程(网络空间安全)

学院

标 准 实 验 报 告

（实验）课程名称 信息检索

电子科技大学教务处制表

电 子 科 技 大 学

实　验　报　告

学生姓名：蔡与望

学　　号：2020010801024

指导教师：邵杰

一、实验环境

- Linux
- Python 3.11
- NumPy 1.26.2, Pandas 2.1.3

二、实验内容

实验要求使用 Python，基于矩阵分解算法实现评分预测；并需要在 MovieLens 数据集上验证算法，取得低于 1.5 的均方误差。

2.1 矩阵分解算法

评分预测是推荐系统中十分重要的一环。目前，主流的评分预测实现方法包括平均值算法、基于用户的邻域算法、基于物品的邻域算法、矩阵分解算法等。其中，基于 SVD 的一系列矩阵分解模型是运用最广泛的模型。

矩阵分解，顾名思义，就是把用户-物品的评分矩阵，用若干个矩阵的乘积进

行拟合，从而得到空缺评分的预测值。然而，传统的 SVD 分解有着计算复杂度高、需要补全稀疏矩阵、存储空间大、数据失真等问题。因此，后续研究提出了一系列基于 SVD 而改进的矩阵分解算法。

2.2 Funk-SVD

Funk-SVD 的基本思想是，将评分矩阵$R_{m\times n}$分解为用户特征矩阵$P_{m\times k}$和物品特征矩阵$Q_{n\times k}$的乘积，即$R = PQ^T$。用户特征矩阵可以理解为，m 位用户在 k 个特征方向上的喜好倾向；物品特征矩阵可以理解为，n 个物品在同样 k 个特征方向上的表现程度。

为了得到这样的 P 和 Q，Funk-SVD 可以采用损失函数，配合随机梯度下降法，逐步学习。具体来说，记(u,i)表示有记录的用户-物品评分对，那么要优化的目标函数是

$$\min_{p,q} \sum_{(u,i)} (r_{ui} - p_u q_i^T)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2)$$

随机梯度下降的更新规则是

$$p_u^* = p_u + \alpha(2e_{ui}q_i - 2\lambda p_u)$$

$$q_i^* = q_i + \alpha(2e_{ui}p_u - 2\lambda q_i)$$

$$e_{ui} = r_{ui} - p_u q_i^T$$

经过迭代，最终就可以得到能够较好地拟合已知评分的 P 和 Q。而它们的乘积，就可以用来预测缺失的评分。

2.3 BiasSVD

BiasSVD 在 Funk-SVD 的基础上，引入了偏置项的概念。

由于每个用户评分高低的习惯不同，所以每个用户在评分时，都有一些与物品无关的因素，也就是用户偏置；又由于每个物品本身质量对评分的影响也不同，所以每个物品收到的评分，也有一些与用户无关的因素，也就是物品偏置。而这两种偏置，是基于一个全局偏置来起作用的，也就是所有评分的平均值。所以，预测的评分可以用下式来表示。

$$r_{ui} = u + b_u + b_i + p_u q_i^T$$

其中，u是评分平均值，b_u是用户偏置，b_i是物品偏置。从而，优化的目标函数变为

$$\min_{p,q,b} \sum_{(u,i)} (r_{ui} - u - b_u - b_i - p_u q_i^T)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + \|b_u\|^2 + \|b_i\|^2)$$

更新规则随之变为

$$p_u^* = p_u + \alpha(2e_{ui}q_i - 2\lambda p_u)$$

$$q_i^* = q_i + \alpha(2e_{ui}p_u - 2\lambda q_i)$$

$$b_u^* = b_u + \alpha(2e_{ui} - 2\lambda b_u)$$

$$b_i^* = b_i + \alpha(2e_{ui} - 2\lambda b_i)$$

$$e_{ui} = r_{ui} - u - b_u - b_i - p_u q_i^T$$

三、实验过程

首先，读取 MovieLens（ml-latest-small.zip）的评分数据集和电影数据集。

```
ratings_data_set = pd.read_csv("data/ratings.csv")
movies_data_set = pd.read_csv("data/movies.csv")
```

将评分数据集划分为训练集、验证集和测试集，比例为 8:1:1。训练集用于计算评分预测矩阵，验证集用于调整超参数，测试集用于评估模型准确性。

```
train_set = ratings_data_set.sample(frac=0.8, random_state=42)
validation_and_test_set = ratings_data_set.drop(train_set.index)
validation_set = validation_and_test_set.sample(frac=0.5, random_state=42)
test_set = validation_and_test_set.drop(validation_set.index)
```

确定评分矩阵的维度，即用户总数 m 和电影总数 n。

```
users_num = ratings_data_set.userId.unique().shape[0]
movies_num = movies_data_set.movieId.unique().shape[0]
```

build_R 函数根据给定的评分数据，创建评分矩阵；未评分用 NaN 来表示。

```
def build_R(ratings: pd.DataFrame):
    R = np.full((users_num, movies_num), np.nan)

    for rating in ratings.itertuples():
        row = rating.userId - 1
        column = movies_data_set[movies_data_set.movieId == rating.movieId].index[0]
        R[row, column] = rating.rating

    return R
```

matrix_factorization 函数使用 BiasSVD 算法来分解评分矩阵。

```
def matrix_factorization(R: np.ndarray, k=3, steps=3000, lr=0.0002, reg=0.01):
    m, n = R.shape

    P = np.random.rand(m, k)
    Q = np.random.rand(n, k)
    B = np.nanmean(R)
    BP = np.random.rand(m)
    BQ = np.random.rand(n)

    for step in range(steps):
        for i, row in enumerate(R):
            for j, value in enumerate(row):
                if np.isnan(value):
                    continue

                diff = value - B - BP[i] - BQ[j] - np.dot(P[i], Q[j])
                P[i] += lr * (2 * diff * Q[j] - 2 * reg * P[i])
                Q[j] += lr * (2 * diff * P[i] - 2 * reg * Q[j])
                BP[i] += lr * (2 * diff - 2 * reg * BP[i])
                BQ[j] += lr * (2 * diff - 2 * reg * BQ[j])

    return P, Q, B, BP, BQ
```

build_R_hat 函数构建评分预测矩阵。

```
def build_R_hat(P: np.ndarray, Q: np.ndarray, B: float, BP: np.ndarray, BQ:
np.ndarray):
    m = P.shape[0]
    n = Q.shape[0]

    return (
        P @ Q.T
        + B
        + BP.reshape((-1, 1)).repeat(n, axis=1)
        + BQ.reshape((1, -1)).repeat(m, axis=0)
    )
```

evaluate 函数在真实的评分数据集上，使用 MSE 来评估预测的准确性。

```
def evaluate(R_hat: np.ndarray, R_real: np.ndarray):
    m, n = R_real.shape

    values_num = np.count_nonzero(~np.isnan(R_real))
    square_error = 0.0
```

```
    for i in range(m):
        for j in range(n):
            if np.isnan(R_real[i, j]):
                continue

            square_error += (R_real[i, j] - R_hat[i, j]) ** 2

    return square_error / values_num
```

使用训练集计算评分预测矩阵。

```
R_train = build_R(train_set)
P, Q, B, BP, BQ = matrix_factorization(R_train)
R_hat = build_R_hat(P, Q, B, BP, BQ)
```

使用验证集调整超参数。

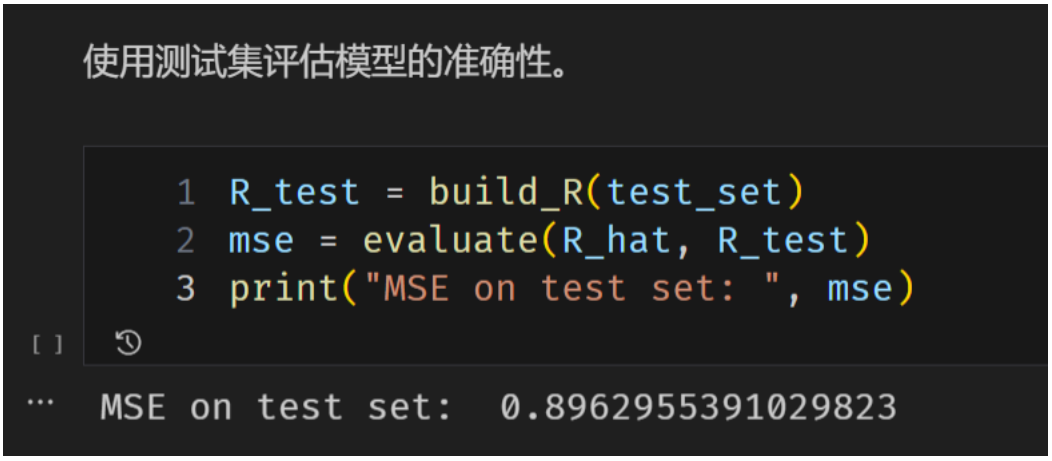
```
R_validation = build_R(validation_set)
mse = evaluate(R_hat, R_validation)
print("MSE on validation set: ", mse)
```

使用测试集评估模型的准确性。

```
R_test = build_R(test_set)
mse = evaluate(R_hat, R_test)
print("MSE on test set: ", mse)
```

四、实验结果

实验中编写的 BiasSVD 模型，可以在测试集上取得 0.896 的 MSE 值，满足“MSE 低于 1.5”的实验要求。



五、总结及体会

通过本次实验，我对推荐系统中 SVD 系列的矩阵分解算法有了深入的了解，包括传统 SVD 的缺点、Funk-SVD、BiasSVD 等；并且通过编写 BiasSVD 的代码，熟悉了它的原理和细节。

然而，该实验并没有考虑到用户喜好随时间的变化。如果能够结合评分数据集中的 timestamp 值，引入时间维度，模型应该会更准确而健壮。