用户协同过滤推荐系统实验报告

苏畅 124033910082

一、实验目的

本实验旨在实现一个基于用户协同过滤的推荐系统，通过用户历史评分数据构建用户相似度矩阵，利用相似用户的评分信息对目标用户的评分行为进行预测，并在验证集与测试集上进行性能评估。

二、实验方法

本实验采用UserCF算法进行评分预测。其核心思想是：如果用户 A 与用户 B 在历史行为上表现相似，则 A 在未来对某一物品的偏好很可能与 B 相似。

给定用户-物品评分矩阵，我们通过以下步骤完成预测任务：

1. 计算用户之间的相似度；
2. 对目标用户，寻找与其最相似的 K 个用户；
3. 基于这些相似用户对某个物品的评分，进行加权预测或简单平均预测；
4. 在验证集上评估预测准确性，计算 MAE，选出效果最好的参数K；
5. 在测试集上输出预测结果。

三、方法与实现细节

> **Note**
>
> 整体流程：
>
> 1. 读取 col_matrix.csv
> 2. 划分数据：
>   - 用户相似度部分：所有用户对前2700列的评分
>   - 验证集：随机1/10用户（不在测试区间），对[2700:]列的评分
>   - 测试集：col_matrix[4100:, 2700:]
> 3. 计算用户之间的相似度矩阵（带掩码 + 惩罚因子）
> 4. 在验证集/测试集上做评分预测：
>   - 对每个目标位置 (i,j)，选出对 j 打过分的 top-k 相似用户，取其评分加权均值
> 5. 生成 test_prediction.csv

1. 相似度函数：惩罚掩码余弦相似度

定义了一个自定义相似度函数 `penalized_masked_cosine_similarity(x, y)`，用于衡量两个用户向量之间的相似性：

- 仅在两用户同时评分的物品上计算余弦相似度；

  只在有共同信息的地方进行比较，能减少噪声；

  但如果交集太小，结果会变得不稳定，因此下面我们引入惩罚因子。

- 引入了一个“惩罚因子”，防止评分重合项太少而高估相似度；

  惩罚系数：交集维度数 / 并集维度数

  这有点类似 Jaccard 的思想，惩罚公式：

$$\text{penalty} = \frac{|\text{nonzero\_intersection}|}{|\text{nonzero\_union}|} \tag{1}$$

  然后把这个系数乘到余弦相似度上，形成最终的得分。

- 具体实现：

```
def penalized_masked_cosine_similarity(x, y, beta=1.0):
    x = np.array(x)
    y = np.array(y)
    mask_inter = (x != 0) & (y != 0)
    mask_union = (x != 0) | (y != 0)

    if np.sum(mask_inter) == 0:
        return 0.0

    x_mask = x[mask_inter]
    y_mask = y[mask_inter]
    sim = np.dot(x_mask, y_mask) / (np.linalg.norm(x_mask) * np.linalg.norm(y_mask))

    penalty = (np.sum(mask_inter) / np.sum(mask_union)) ** beta
    return sim * penalty
```

其中beta是一个可以调整的超参数，控制惩罚程度，值越大惩罚越大，默认值为1。

2. 验证集划分与约束处理

采用如下方式划分验证集：

```
val_users = np.random.choice(all_users, int(len(all_users) * VALIDATION_RATIO),
replace=False)
val_mask[val_users, 2700:] = True
```

- `VALIDATION_RATIO=0.1`，表示从训练用户中随机抽取 10% 作为验证集；
- 验证集的评分区域仅包含第 2700 列及之后的物品，这样能更好地跟测试集对齐。

特别地： 在验证集预测时，为了确保公平性，禁止某个验证用户参考其在验证区域的评分结果，即在预测中删除自身评分信息：

```
if i in val_users and j >= 2700:
    rated_users = rated_users[rated_users != i]
```

3. 用户相似度矩阵计算

对所有用户之间计算基于前 2700 列（训练区域）的相似度，得到对称矩阵 `similarity_matrix`。

```
for i in range(n_users):
    for j in range(i, n_users):
        sim = penalized_masked_cosine_similarity(user_vectors[i], user_vectors[j])
```

这一步是计算量最大的部分，时间复杂度为 `O(N*N*D)`，其中`D`是评分维度，在我们的设置里也就是 `2700`。

在我们的实验测试中计算出相似度矩阵需要`13m13s`。

```
100%|██████████| 6040/6040 [13:13<00:00,  7.62it/s]
```

4. 预测函数：`predict_scores`

核心预测函数 `predict_scores(target_users, target_items)`，对每一个目标用户和目标物品对，执行以下操作：

- 找出对该物品有评分的所有用户 `rated_users`；
- 若该用户是验证用户，且该物品属于验证区域，则剔除自身评分；
- 根据相似度矩阵查找与目标用户最相似的 K 个评分者；
- 采用相似度的加权平均：

  ```
  pred = np.dot(top_k_ratings, top_k_sims) / np.sum(top_k_sims)
  ```

  根据相似度进行加权的做法可以使我们的算法对于k值更加鲁棒，不易因k值产生较大抖动。
- 若没有参考评分，则默认预测为中立值 3。

最后，将预测值进行四舍五入并裁剪至评分范围 [1, 5]：

```
prediction_matrix = np.clip(np.round(prediction_matrix), 1, 5)
```

5. 验证集评估指标

使用 Mean Absolute Error (MAE) 衡量预测误差：

```
mae = np.mean(np.abs((val_truth[val_truth != 0] - val_pred[val_truth != 0])))
```

`0` 表示**缺失值**，不是实际的评分；如果把预测结果和原始矩阵中的 `0` 比较，是在对“根本不知道真实值”的位置做评估，是不合理的。因此实验中只统计非零评分项（即实际有打分的数据）。

对验证集上的410位用户测试时间为`51s`。

```
100%|██████████| 410/410 [00:51<00:00,  8.00it/s]
```

6. 测试集预测

对测试集用户（4100 以后的用户）进行预测：

- 测试用户可以参考训练用户和验证用户的评分信息；
- 对所有 item（2700 后）进行预测；
- 保存为 `test_prediction.csv` 文件，格式为整数评分；
- 经代码检查，我们的结果符合格式要求。

对验证集上的1940位用户测试时间为`3m56s`。

```
100%|██████████| 1940/1940 [03:56<00:00,  8.20it/s]
```

> **Important**
>
> 总结来说，本实验采用的是基于用户的协同过滤，并结合掩码余弦相似度与惩罚因子进行改进，以适应数据中评分高度稀疏的特点。
>
> 以下是我们的实现与传统UserCF算法的比较。

	本实现	传统 UserCF
相似度	掩码余弦相似度 + 惩罚因子	标准余弦或皮尔逊相关系数
特征向量来源	所有用户对前 2700 个电影的评分	通常是全量用户历史
处理稀疏	明确考虑了评分缺失（0为缺失）	一般默认忽略
相似度惩罚	如果两个用户交集少，则相似度被削弱	一般没有惩罚因子

四、实验结果

我们进行了一系列的实验，测试了不同k值下我们的方法在验证集的MAE值，如下表所示：

	k=5	k=10	k=20	k= 30	k=40	k=50	k=100
MAE	0.7637	0.7407	0.7278	0.7216	0.7215	0.7223	0.7245

可以看到，由于我们在预测评分是根据相似度对分数进行加权平均，所以对k值比较鲁棒。

当`k=40`时候性能最好，因此我们在测试集上使用`k=40`对分数进行预测，结果保存在`col_matrix.csv`文件里。

五、总结与分析

本实验实现了基于用户协同过滤的评分预测方法，核心思想是通过计算用户之间的兴趣相似度，利用相似用户的历史评分对目标用户的未知评分进行预测。整个流程包括数据加载、验证集划分、相似度计算、评分预测和性能评估等步骤。

在相似度计算部分，采用了带惩罚因子的掩码余弦相似度，能够有效处理评分矩阵中存在大量缺失值的情况，同时引入交集与并集比例作为惩罚项，增强了对评分重合度的考量，提升了相似度的可信度。

为了确保验证评估的公平性，在对验证用户进行评分预测时明确排除了该用户对验证区间内条目的实际评分，避免了信息泄露，确保了模型评估的客观性。同时，在测试阶段，允许测试用户参考所有非测试用户的评分信息，包括验证集用户，以充分利用已有数据进行预测。

预测部分使用了加权平均的方式对评分进行估计，其中相似度作为权重，选取相似度前 K 大的用户进行加权汇总。最终预测结果经四舍五入并限制在评分区间 [1, 5] 内，保证了结果的合理性。

综上，本实验顺利实现了用户对特定影片的评分预测，通过基于相似用户的加权评分估计方法，成功完成了对用户兴趣的建模与推理，并在验证集上取得了较为合理的预测结果，验证了协同过滤方法在处理稀疏评分数据上的有效性和可行性。