作业二：矩阵分解

刘昱杉 2024214103

2024 年 11 月 23 日

一、数据准备

本次作业中，我们使用 Netflix 数据集，数据集被分成训练集 `netflix_train.txt` 和测试集 `netflix_test.txt`。每一行的格式为 `user_id`, `movie_id`, `rating`，表示用户 `user_id` 对电影 `movie_id` 的评分为 `rating`。数据准备主要包括：

- **数据加载**：使用 `pandas` 库加载数据集。
- **索引映射**: 为用户 ID 和电影 ID 创建唯一映射，并将 ID 映射为索引，便于后续矩阵运算。
- **稀疏矩阵创建**: 将评分数据转换为稀疏矩阵 `coo_matrix`，其中行为用户，列为电影，值为评分。

这里使用了**完整的数据集**，即训练集和测试集的数据都被加载，以便后续的矩阵分解模型训练和评估。

二、协同过滤与评价指标

本作业中的协同过滤算法通过计算**用户相似度矩阵**实现，通过余弦相似度计算。具体来说，首先对用户评分矩阵归一化，之后计算用户之间的点积。评分预测通过对**最相似的 k 个用户**的评分进行平均来实现。

评价指标选用均方根误差 (RMSE) 结果，本作业中通过对测试集上计算协同过滤方法的 RMSE，最终得到值为 **1.1021**。整个流程包括：计算用户相似度、预测评分、评估 RMSE，对整个测试集的计算耗时约 **0.85s**。计算过程通过 PyTorch 进行了优化，以确保计算效率。

关于公式到矩阵计算的推导如下：

假设有用户-电影评分矩阵 $R \in \mathbb{R}^{m \times n}$，其中 m 为用户数量，n 为物品数量。用户 i 对物品 j 的评分为 r_{ij}。协同过滤通过计算用户之间的相似度矩阵 $S \in \mathbb{R}^{m \times m}$，使用如下的余弦相似度公式：

$$s_{ij} = \frac{R_i \cdot R_j}{\|R_i\| \cdot \|R_j\|}, \tag{1}$$

其中，R_i 和 R_j 分别表示用户 i 和用户 j 的评分向量。归一化后，用户评分矩阵可以表示为：

$$\hat{R}_i = \frac{R_i}{\|R_i\|}, \tag{2}$$

相似度矩阵 S 则可以通过矩阵乘法表示为：

$$S = \hat{R}\hat{R}^T, \tag{3}$$

其中 $\hat{R}$ 为归一化后的用户评分矩阵。通过这个相似度矩阵，我们可以计算用户 u 对电影 j 的预测评分 $\hat{r}_{uj}$，方法是对最相似的 k 个用户的评分取加权平均：

$$\hat{r}_{uj} = \frac{\sum_{i \in N_k(u)} s_{ui} r_{ij}}{\sum_{i \in N_k(u)} s_{ui}}, \tag{4}$$

其中 $N_k(u)$ 表示与用户 u 最相似的 k 个用户集合，s_{ui} 为用户 u 和用户 i 之间的相似度。

三、矩阵分解

矩阵分解模型同样基于 `Pytorch` 实现，主要包括 **User-Item Embedding** 和 **USer- Item Bias**，可以更好地捕捉特定特征；同时防止过拟合添加了 Dropout 项。训练模型使用 **AdamW** 优化器，学习率设置为 0.01，空间维度设置为 50，正则化参数 λ 设置为 0.01，训练轮数为 50。训练过程中分别记录每 5 次 epoch 的 loss 和测试集上的 RMSE, 基于此参数下的 RMSE 为 **0.9187**。分别在训练集和测试集上进行 RMSE 计算并进行曲线绘制如下。

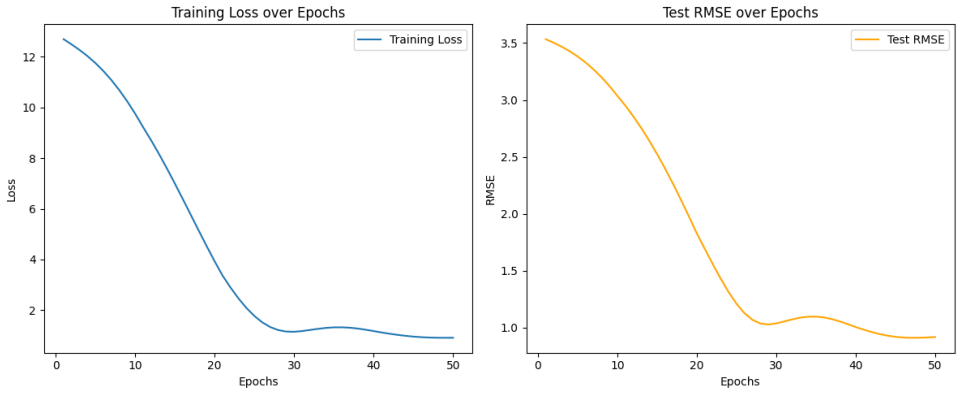

图 1: 训练集和测试集上的 RMSE 曲线

参数调优

分别使用不同的空间维度和正则化参数进行调优，并分别比较其在测试集上的 RMSE 值，结果如下。由原理可知，更高的 k 值可以捕捉更多特征，前期下降的更快，但到一定程度后效果会趋于平稳。增加 λ 有助于防止过拟合，但过高的值会阻碍模型学习。其中最佳参数组合为：空间维度 $k = 50$，正则化参数 $\lambda = 0.001$，测试集 RMSE 为 0.9241。

空间维度 k	正则化参数 λ	测试集 RMSE
20	0.001	0.9804
20	0.01	0.9806
20	0.1	0.9857
50	0.001	0.9241
50	0.01	0.9314
50	0.1	0.9319

表 1: 不同参数下的测试集 RMSE

四、比较与讨论

通过对协同过滤和矩阵分解模型的比较，可以发现矩阵分解模型在测试集上的 RMSE 值更低，更好地捕捉潜在的交互关系。同时，由于协同过滤模型的计算复杂度较高，因此在计算效率上矩阵分解模型更具优势。

协同过滤和矩阵分解各有优劣，其各自的优缺点分别如下：

协同过滤

- 优点：简单易实现，不需要额外的特征工程，可以捕捉用户之间的相似度。
- 缺点：协同过滤对于稀疏数据不够鲁棒，且当用户数量庞大时，计算相似度矩阵的开销很大，影响效率。

矩阵分解

- 优点：矩阵分解模型可以更好地捕捉用户和物品之间的交互关系，同时可以通过添加特征更好地解释模型。通过降低矩阵维度，矩阵分解适用于大型数据集，具有良好的扩展性。
- 缺点：矩阵分解模型需要额外的特征工程，且其性能对超参数（如嵌入维度、正则化系数等）较为敏感。

参考文献

[1] Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.

[2] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8), 30-37.

[3] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32.

[4] ChatGPT, OpenAI. (2024). Implementation and comparison of collaborative filtering and matrix factorization in recommendation systems.