基于协同过滤和 MLP 融合的电影推荐系统学习报告

23020007110 汤琦

一、项目背景

在信息爆炸的时代，用户面临海量的电影内容，如何快速有效地找到符合其偏好的电影成为一个重要问题。传统的推荐系统，如协同过滤，在处理数据稀疏性或新项目/用户（冷启动）时可能存在局限。内容推荐则依赖于项目本身的特征，可能缺乏个性化。本项目旨在构建一个电影推荐系统，结合经典的协同过滤算法和基于电影内容相似度的内容进行推荐，并最终使用深度学习的 MLP 模型对两者的预测结果进行融合，从而为用户提供更准确、个性化的电影推荐。通过融合不同推荐策略的优势，期望能够提高推荐的准确性。

二、 具体分工

在本项目中，我们认识到，无论是电影推荐还是其他信息服务，用户都常常淹没在海量且内容高度相似的信息之中，如何在如此庞杂的数据海洋中为用户精准筛选出个性化、有价值且非重复的内容，是推荐系统设计的根本所在。基于这一理解，小组成员根据各自的专长和职责进行了明确分工，从而合力完善了电影推荐系统。

作为核心算法的负责者，我主要致力于根据数据集中给出的电影类型和不同人的评分数据为用户提供相关电影推荐。其中，我负责了协同过滤（CF）与内容推荐（CB）两大基础推荐模块的实现，并主导了深度学习 MLP 融合模型的设计与训练。我的目标是通过这些技术手段，在一定程度上解决内容冗余问题。内容推荐模块着重于深入挖掘电影本身的类型特征，确保推荐电影内容题材符合用户偏好。二协同过滤模块使我们能够依据用户群体的历史行为和共同偏好，从电影库中识别出用户可能感兴趣但尚未发现的“佳片”，有效地避免了仅凭内容相似度可能导致的推荐结果过于同质化、缺乏惊喜感的问题。而 MLP 融合模型可以作为一个“元学习器”，学习如何权衡好协同过滤所产生的“基于相似群体的推荐”与内容推荐所体现的“基于个体偏好的推荐”。这种集成方式，

使得模型能够在复杂的电影内容冗余场景下，提供更为准确、更具多样性的推荐结果并尽量避免内容同质化带来的审美疲劳。

团队中的数据工程师负责了原始 MovieLens 1M 数据集的获取、清洗和预处理工作，确保了数据的完整性、一致性与可用性。后端开发工程师主要负责将我的算法模块进行封装，搭建 API 接口，使得推荐模型能够接收用户请求并高效地返回推荐结果，确保模型预测能够无缝地融入整个应用架构，为前端提供可靠的数据支持。前端开发工程师则负责将整个推荐系统的最终用户界面呈现出来。他们将后端提供的推荐结果直观地展示给用户，并设计了美观的用户交互界面，使得用户能够方便、享受地探索更多电影并提出反馈意见。

三、项目目标与核心流程

3.1 项目目标

- 熟悉 Surprise 库的基本使用。
- 理解基于用户-物品协同过滤算法（KNNBasic）的原理并实现。
- 掌握基于电影类别（genres）进行内容推荐的方法。
- 学习并设计、训练一个 MLP（多层感知机）模型，以融合不同推荐算法的预测结果，提升推荐质量。
- 尝试以公共数据集为基础，创建个性化的训练数据集来进行进一步模型训练。

3.2 核心流程

- **数据加载与预处理**：从 MovieLens 1M 数据集加载电影和用户评分数据，并基于数据特点进行预处理。
- **协同过滤（CF）推荐**：利用 Surprise 库实现 KNNBasic 算法，为用户生成基于相似用户的电影预测评分。
- **内容推荐（CB）生成**：对电影类型进行 TF-IDF 向量化，并通过计算用户高评分电影的类型偏好与未评分电影类型之间的余弦相似度，生成内容推荐分数。

- **MLP 融合模型训练**：设计并训练一个简单 MLP，以协同过滤预测和内容推荐相似度作为输入特征，预测最终的用户对电影的评分。
- **最终推荐列表生成**：根据 MLP 模型的预测结果，为指定用户输出推荐电影列表。

四、数据来源与预处理

4.1 数据来源

本项目使用了公开的 MovieLens 1M 数据集，该数据集广泛用于推荐系统研究，本次实践使用的数据集为：

- ratings.dat：存储了用户对电影的评分记录，包括 userId（用户 ID）、movieId（电影 ID）、rating（评分，1-5 星）和 timestamp（评分时间戳）。
- movies.dat：存储了电影的基本信息，包括 movieId（电影 ID）、title（电影标题）和 genres（电影类型，多个类型以 | 分隔，如：Animation|Children's|Comedy）。

4.2 数据预处理流程

使用 Pandas 库读取 ratings.dat 和 movies.dat 文件，观察数据集文件发现其格式与之前的数据集不同——不同条目之间使用“::”作为分隔符，所以要自行设定分隔符参数，并且将 pandas 用于解析 csv 文件的 C 引擎更换为 Python 以适配双字符分隔符。在 movies.dat 文件中，存在着非 ASCII 字符，如：Misérables, Les (1995)中的 é 字符，查阅了一些资料使用了编码 ISO-8859-1，这种编码虽然对中文等双字节编码不适用，但是对于数据集

中出现的西欧字符适配性较好，然后将数据集中的时间戳属性从年-月-日格式改为用秒数表示，便于处理。具体代码见 Homework6.ipynb 的第一部分 – 加载数据集。

五、推荐模型设计与实现

5.1 协同过滤（Collaborative Filtering, CF）

我选用了基于物品的 KNNBasic 算法来完成协同过滤的部分。该算法工作流程如下：对于每一个电影，收集所有的评分，构成一个向量。然后通过计算不同向量的相似度来决定电影之间的相似度。这里我选择通过余弦相似度来计算向量的相似度，对于电影 i, j 的相似度具体定义如下（v_i, v_j分别表示电影 i, j 对应的特征向量）：

$$sim(i,j) = \frac{v_i \cdot v_j}{||v_i|| \cdot ||v_j||}$$

实现步骤：

1. 初始化 Reader 并加载 ratings 数据集。
2. 调用 data.build_full_trainset() 构建训练集 trainset。
3. 实例化 KNNBasic 模型，其相似度计算选项 sim_options 设置为 name='cosine'（余弦相似度）、计算方式 user_based=False（基于物品相似度来推荐）。
4. 使用 algo.fit(trainset) 方法在训练集上拟合模型。
5. 对于目标用户（例如 user_id = 1），遍历所有未评分的电影，通过 algo.predict(user_id, movie_id).est 获取该电影的协同过滤预测评分。
6. 将预测结果按评分降序排序，输出前十项。

具体代码见 Homework6.ipynb 中的第二部分。

5.2 内容推荐（Content-Based, CB）

首先将电影本身的属性（类型）向量化。然后，对于每一个用户，选择其曾给出过高评分(>4.0)的电影，将它们的属性向量加权平均，得到该用户的类型偏好向量。然后，通过计算每个电影的向量和用户的类型偏好向量之间的余弦相似度来按相似度降序排序。

实现步骤：

1. 利用 TF-IDF 向量化工具 TfidfVectorizer 将 movies 数据集中的 genres 列转换为数值表示的 tfidf_matrix。
2. 识别目标用户评分 ≥4.0 的电影的 movieId。
3. 从 tfidf_matrix 中提取这些高评分电影对应的类型向量，并计算它们的平均值，形成用户的内容偏好向量 profile。
4. 计算该 profile 向量与所有电影在 tfidf_matrix 中的类型向量的余弦相似度，得到 cb_scores。cb_scores 中的每个值代表一部电影与用户类型偏好的相似度。
5. 将 cb_scores 按降序排列，输出前 10 项，作为展示。

具体代码见 Homework6.ipynb 中的第三部分。

5.3 MLP 融合模型

构建一个 MLP（多层感知机）神经网络进行回归任务，网络结构如下：

1. 输入层：2 个神经元，分别接收协同过滤预测分数和内容推荐相似度分数。

2. 第一隐藏层：nn.Linear(2, 32)，使用 BatchNorm1d(32) 进行批归一化，ReLU 激活函数，以及 Dropout(0.3) 正则化。
3. 第二隐藏层：nn.Linear(32, 16)，同样使用 BatchNorm1d(16)，ReLU 激活函数和 Dropout(0.3) 正则化。
4. 输出层：nn.Linear(16, 1)，输出一个单一的预测评分。

模型设计思路大致如下：首先通过 Linear 层将输入（2 维向量）映射到更高维度的隐藏空间中来展现更复杂的变量之间的交互模式。然后通过查阅资料得知在深度神经网络中，随着层数的增加，前一层的参数变化可能会导致后一层输入的分布发生显著变化（即“内部协变量偏移”），所以需要加入批归一化层，批归一化通过将每一层的输入标准化，稳定了各层输入的分布，使得训练过程更加平滑，加快模型的收敛速度。接着添加 ReLU 激活函数，引入非线性，使模型在一定程度上具备学习变量之间复杂关系的能力，并添加 Dropout 层来提高模型的泛化能力。

将上述结构进行重复并逐步减小 Linear 层的输出规模，以达到精炼信息的目的，就有了这样的模型设计。

然后基于 MovieLens 1M 数据集，整合为用来训练 MLP 的数据集形式。将 (cf_pred, cb_pred) 对作为特征，真实评分 true_r 作为目标，构建 train_data 列表。定义 RecDataset 类，将 train_data 封装为 PyTorch 的 Dataset 对象，并使用 DataLoader 实现数据的批量加载和随机打乱，使用新建的数据集来训练 MLP 模型。

5．4 生成最终得分，并进行推荐

待模型训练完成后，将模型切换到评估模式（model.eval()），此时，不需要生成计算图用于反向传播，故禁用梯度计算（torch.no_grad()）。遍历通过协同过滤得到的推荐电影列表 n_cf。对于每部电影，获取其协同过滤预测分数（cf_pred）和内容推荐相似度分数（cb_pred）。将这两个特征输入到训练好的 MLP 模型中，获取最终的融合预测评分，并存储在 nn_scores 字典中。最后，根据最终得分降序排列输出最终结果。

具体代码见 Homework6.ipynb 中的第四部分。

在代码实现中以用户 1 为例，最终推荐输出如下：

最终评分 Top10 推荐：
Loves of Carmen, The (1948) (MovieID: 3209), 预测融合评分: 4.75
Voyage to the Beginning of the World (1997) (MovieID: 1915), 预测融合评分: 4.75
Back Stage (2000) (MovieID: 3890), 预测融合评分: 4.58
Smoking/No Smoking (1993) (MovieID: 3530), 预测融合评分: 4.51
Chain of Fools (2000) (MovieID: 3323), 预测融合评分: 4.49
Silence of the Palace, The (Saimt el Qusur) (1994) (MovieID: 127), 预测融合评分: 4.44
Song of Freedom (1936) (MovieID: 3382), 预测融合评分: 4.44
Outside Ozona (1998) (MovieID: 2438), 预测融合评分: 4.36

Project Moon Base (1953) (MovieID: 3779), 预测融合评分: 4.33

Vampyros Lesbos (Las Vampiras) (1970) (MovieID: 3216), 预测融合评分: 4.32

五、模型局限性

当前 MLP 模型仅使用了协同过滤预测和内容相似度作为输入特征，并未充分利用其他潜在有价值的信息。同时，内容推荐部分考虑电影类型时，没有考虑哪一种类型为主类型，推荐的类型可能并不精准。

另外，对于新用户或新电影，由于缺乏足够的历史评分数据或 profile 信息，协同过滤和内容推荐可能效果不佳，这也会影响 MLP 融合的准确性。

六、收获与感悟

本次基于协同过滤与 MLP 融合的电影推荐系统构建实践，让我系统地学习和掌握了推荐系统的一些核心技术，提升了编程能力，也对实际项目中的机器学习算法应用、跨模型集成以及 AI 落地场景的复杂性和价值有了深刻的体会。

在整个项目实施过程中，团队成员通过持续而有效的沟通、共享代码库，确保了数据流的顺畅和各模块接口的兼容性，从而实现了跨职责的合作。在实践过程中，我也意识到了团队合作工作模式的价值。

首先，它极大地拓展了个人的技术视野。每个人确定好需要提供的接口的规范，利用计算机系统工程中的模块化和抽象来进行协作，使得每个人能专注于自己负责的模块，极大地提高了开发效率。在独立完成小作业时，我往往专注于特定算法的实现和优化，而团队协作迫使我跳出自己的舒适区，去理解并对接数据工程师提供的数据接口、后端工程师如何将模型封装为可用的 API 服务。这种跨领域的接触，让我从数据源头到最终用户界面对呈现对一个完整的 AI 应用的开发过程有了更具体的认知。

其次，团队协作显著提升了问题解决的效率和质量。在项目推进过程中，我们不可避免地遇到过数据格式不统一、模型接口冲突、训练过程中出现的性能瓶颈等问题。对于这些问题，团队成员通过集思广益，共同进行调试和优化最终很好的解决了问题。例如，数据预处理阶段遇到的效率问题，可能由数据工程师提出解决方案，算法工程书则需调整数据加载策略以配合；当我在模型融合阶段遇到性能瓶颈时，后端或部署同学可能会从硬件配置或分布式计算的角度提供建议。这种视角的碰撞，不仅加快了问题的解决速度，也往往能找到更优雅、更全面的解决方案。

另外，本次协作培养了我的沟通能力和系统化思维。在复杂的项目中，如何清晰地表达自己的需求、理解他人的限制、以及共同高效地协商解决方案至关重要。在本次实践中，我学会了如何在需求变更时与各方进行有效沟通，如何在具体实现之前通过明确的接口定义来尽量避免集成时的错误，以及如何考虑各方需求，从一个更宏观的角度去审视和优化自己的代码。

此外，本次项目也让我对 AI 落地场景的真实挑战有了更深刻的认识。电影推荐系统作为日常生活中 AI 技术的一个典型应用案例，在实现上还要考虑的现实的因素（如用户体验方面，应能够帮助用户在海量电影中快速有效地找到符合其偏好的内容）。在实践过程中，我在构建数据集时发现程序中的性能瓶颈并进行了简单的优化；在处理 MovieLens 1M 这样规模的数据集，以及在训练 MLP ，自行构建训练集的过程中，也让我意识到数据工程在大模型应用中的重要性。