推荐系统课程实验报告

Final

霍超凡

摘要

本课程实验探讨了面向评分预测的协同过滤算法和面向 CTR 任务的深度学习模型，在报告一介绍了本系列实验的计划安排；在报告二对数据集进行了分析，并探讨了数据集在内存中的组织形式及其访问效率；之后的报告三、四、五分别探讨了基于领域的协同过滤算法、基于 SVD 协同过滤算法和基于多层感知器的协同过滤算法。最后的报告六跳到另外一个话题—CTR 任务，这个报告探讨了神经网络模型在 Criteo 数据集上的效果。本系列实验涵盖到课程中的各个方面，诸如第 3 讲基于领域的协同推荐、第 4 讲的基于 SVD 协同推荐、第 9 讲因子分解机中的 DeepFM、第 11 讲推荐中的深度学习匹配表示学习。前四个实验均在 python 的开发环境开展，后面两个实验使用 Baidu 提供的 Paddle 学习框架训练网络。本报告对前面六个报告进行总结。

1 选题

推荐任务目标在于寻找物品用户之间的匹配关系，把用户感兴趣的物品推荐给用户，为了精确的推荐，我们的算法必须要充分了解用户的喜好和物品的性质，大多数情况下，用户的喜好是难以获得的，其中涉及到隐私等问题，历史活动记录成为我们推荐的主要依据。协同过滤算法是一种利用群体大数据的算法，它不聚焦单个用户，而是放眼全局，它一定程度上解决了数据稀疏的问题。协同过滤算法中最典型的是基于领域的协同过滤算法，基于领域的协同过滤算法又可分成基于用户的协同过滤和基于物品的协同过滤，针对这种协同过滤算法在二十世纪和二十一世纪之交研究比较火热，这些研究针对相似度计算、融合评分、引入聚类算法做出了不同程度的改进。基于领域的协同过滤算法最大的缺点在于十分消耗内存，基于 SVD 算法一定程度克服了这种缺陷，SVD 协同过滤算法把物品和用户向量化表示，这个向量是对物品和用户的描述，物品与用户之间的匹配程度需要在推荐过程中算出来，而不需要存储每一个用户和物品的匹配关系，由此减小的存储，但是另外一方面增加了冗余计算，当对系统内部的热门用户推荐时，需要多次重复计算该用户和其它物品的匹配关系，这可以使用缓存来解决重复计算的问题。基于多层感知器的推荐模型可以融合除了评分矩阵之外的信息，万物皆可 embedding，多层感知器重点在于对用户和物品的向量化表示方面。面向用户点击率的 CTR 任务是推荐系统的另外一个子领域，针对该任务提出的面向有 Deep & Cross Net、DIN、DeepFM 和 xDeepFM 等，实验中重现了 DeepFM 和 DNN 模型。

前面的协同过滤算法在 MovieLens 数据集上评比性能，后面的 CTR 模型在 Criteo 数据集上评比性能，本系列实验不涉及到开发平台或开源工具的安装。

2 工作流程

和实验配套的代码分成两部分，前面基于领域的协同过滤算法、基于 SVD 的协同过滤算法在 CPU 环境中，后面的 Multi-perception 模型和 CTR 模型在 GPU 环境下，下面依次介绍这些代码的组织形式。

2.1 CPU 环境下的推荐模型

在 cpu 文件夹下，有子文件夹 datasets、evaluate、models、utils、visualize，datasets 是各种数据集读取操作的 py 文件，models 主要是各种推荐模型，utils 文件夹中存放使用到的各种工具函数，evaluate 文件夹内部的文件计算这些模型的各种评价指标。

- 在 datasets 文件夹下，
  - dataset.py：评分矩阵的存储方式，有 RatingNDArray、RatingDataFrame、RatingDictionary 三个类，分别使用 ndarray、dataframe、dict 存储方式组织评分矩阵。
  - movieLens.py：movieLens 数据集的加载和操作，内部面对不同规模的 movieLens 数据集分别写了一个类。
  - book_crossing.py、netflix.py、personality.py：其它数据集的加载和操作，整个实验过程中没有使用到这些数据集。
- 在 models 文件夹下，
  - recommender.py：抽象类，定义了一个推荐算法的基本功能。
  - item_based.py：基于物品的推荐算法。
  - user_based.py：基于用户的推荐算法。
  - svd.py：基于 SVD 的推荐算法。
  - bias_svd.py：带有用户、物品偏好的 SVD 推荐算法。
  - svd_plus_user_based.py：融合 SVD 的基于用户的推荐算法，性能不好，没做进一步实验。
- 在 utils 文件夹下，
  - similarity.py：基于领域中的推荐算法计算相似度的方式，主要是 cosine 和 person 系数。
- 在 evaluate 文件夹下，
  - memory_access.py：评测不同存储组织结构的数据集访问效率
  - evaluate_userbased_itembased_model.py：评测基于领域的推荐算法，这是一开始编写的，实验大部分都使用下面两个文件评测。
  - evaluate_model.py：计算各种模型的评价指标。
  - evaluate_model_v2.py：修改版，在计算 precise 忽略了那些评分为 0 的数据。
- 在 visualize 文件夹下，
  - plot_curve_user_based.ipynb：可视化结果，需要在 Jupyter notebook 环境下运行。
  - plot_curve_svd.ipynb：同上。

  若要重现这些实验工作，需要完成以下步骤：

  1. 在报告二所提供的各种数据集链接网页中下载相关的数据集。
  2. 在 models 文件夹中修改数据集的默认路径位置。
  3. 在 evaluate 文件夹下，修改 evaluate_model.py 中的数据集、模型参数设置，打开

命令行，输入 python evaluate_model.py 验证结果。

2.2 Paddle 环境下的推荐模型

在 paddle 文件夹下，有 datasets、models、utils、train、evaluate 子文件夹。

- datasets：Criteo 数据集的加载和操作，
  - movieLens.py：在 cpu/models/movieLens.py 基础上做出了修改。
  - criteo_dataset_generator.py：流水线加载数据集，融合到 Paddle 中的模型训练过程中，加快流水线训练。
  - criteo_dataset_reader.py、criteo_dataset_feeder.py：数据集的读取。
- models：各种神经网络模型
  - recommend.py：同 cpu/models/recommend.py。
  - multi_percetion.py：多层感知器推荐模型。
  - dnn.py、drnn.py、deepfm.py 等：各种网络结构。
- utils：工具文件
  - option.py：基本没用。
  - config.py：集中配置网络参数、训练参数。
- train：训练网络
  - 以 train_dygraph_2.py 为准，
- evaluate：测试模型

若要重现这些工作，需要注册一个 Aistudio 账号，在项目页面下，公开项目搜索 RecommendSystemExperiment，应该是可以重现当时的工作环境，当时数据集需要自己下载上传。

3 最终结果情况

3.1 系统截图

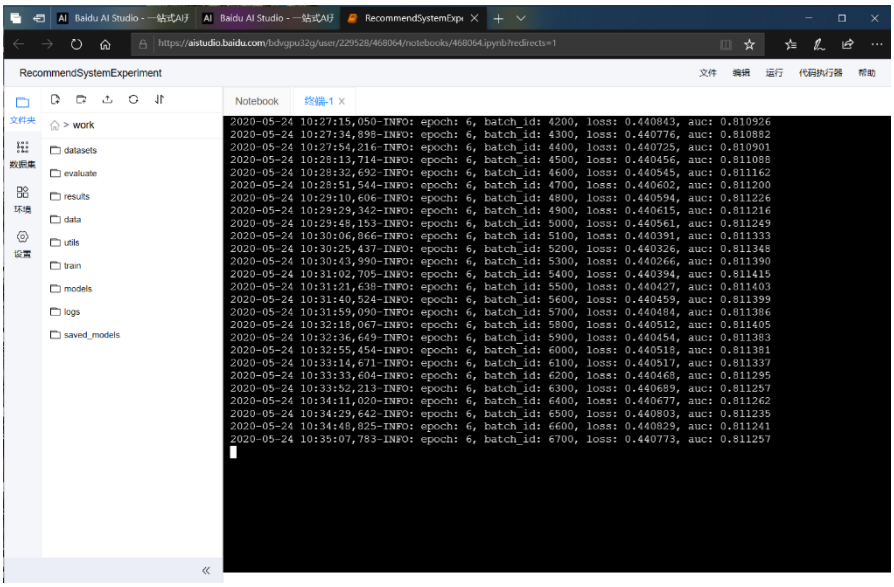

图 1：神经网络模型训练过程截图

3.2 数据量

协同过滤算法在 MovieLens 数据集上评测，数据量大小为 MovieLens100k、和 MovieLens1M，诸如 DNN、DeepFM 在数据集 Criteo 上评测，该数据集有 45M 条记录，每一条记录由 26 个离散特征、13 个整数特征和用户是否点击的 label 组成。

3.3 性能评价

各个模型的性能指标结果请参考之前的报告，这里只是把模型整合起来对比。在图 1 中对比了这些模型，注意横纵坐标做出了处理，横坐标 speed 表示算法每秒钟能给多少名用户推荐，纵坐标对 RMSE 取了分数，越往右上角的模型，性能越好，没有时间效率和预测准确度都好的模型，必须在时间效率和预测准确度之间做出权衡，基于多层感知器的协同过滤算法融合了内容信息和评分信息，具有最好的预测准确度，但是该模型在 CPU 上运行非常耗时，综合来看，SVD 一类算法综合时间效率和预测准确度来看，比其他算法要好，基于用户的协同过滤算法效果最差。

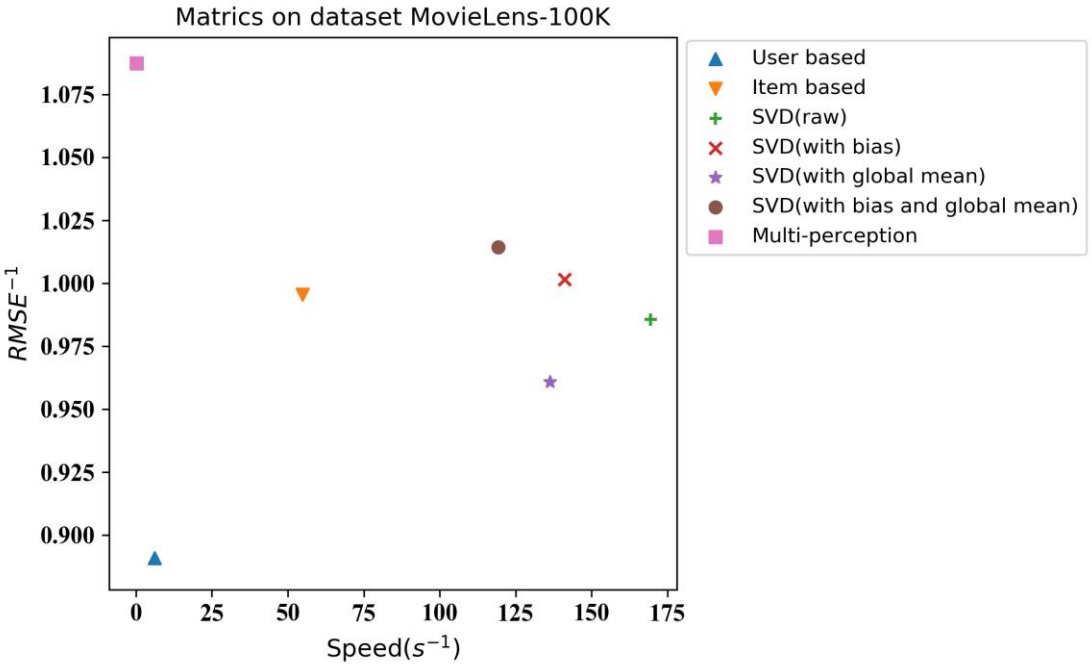

图 2：各种模型在数据集 MovieLens-100K 上的指标

图 3 给出了，DRNN 在数据集 Criteo 上的性能，这种图中，可以发现深层 DRNN 要好于前层 DRNN，我们必须意识到到深度 DRNN 比浅层 DRNN 的预测时间长，如果只是为了追求一点点的性能提高，而使模型时间效率减半，在实际应用中要仔细权衡。

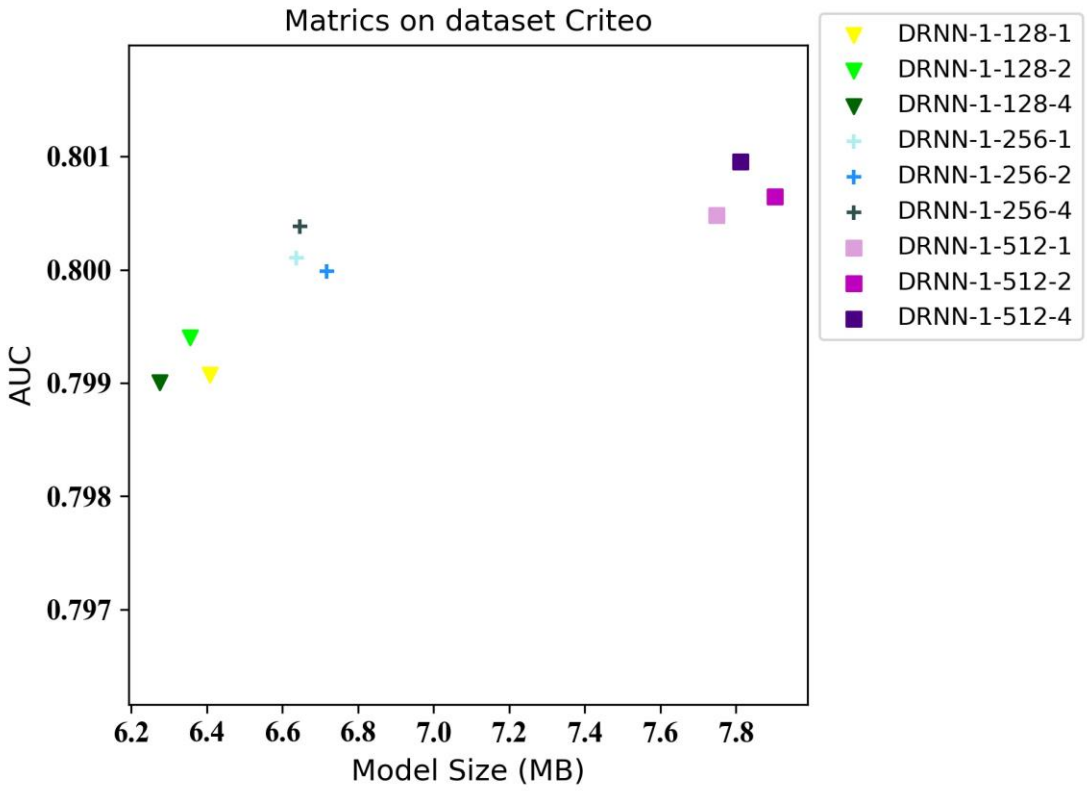

图 3：DRNN 在 Criteo 数据集上的指标