基于 spark 对 movielens 的数据分析及电影推荐系统

学号：16340141

姓名：林妙倩

目录

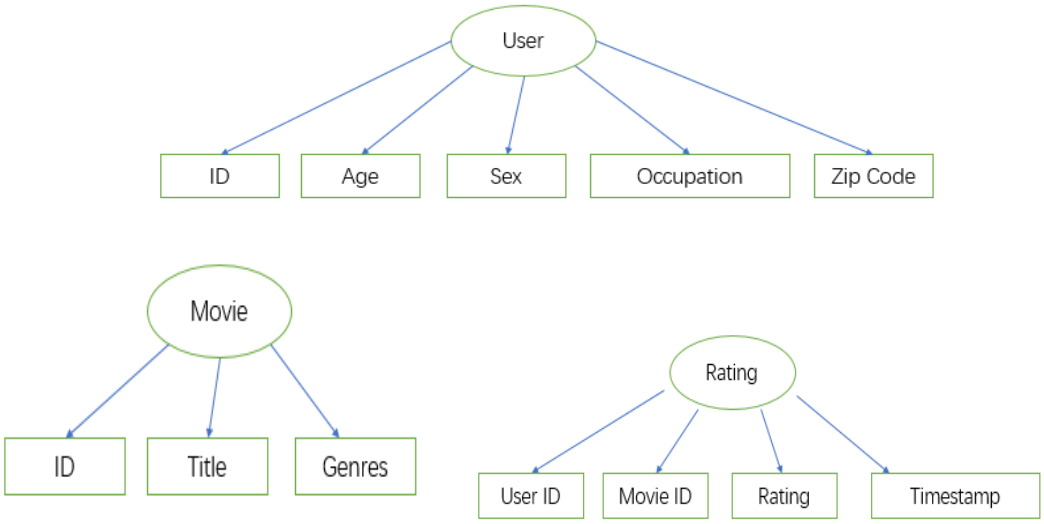
项目概述

本项目采用 spark 集群框架。利用 Map Reduce 思想对 movielens 数据集进行分析，得到用户职业分布，电影类型，评分等数据。并利用协同过滤 ALS 算法设计电影推荐系统，根据已有的评分数据每个用户推荐电影。

项目设计

数据分析

Movielens 数据集分为三部分：movies.dat、ratings.dat、users.dat。电影，评分，用户记录包含的属性如下：



统计每个职业的人数：

首先通过 map 将每条记录转化为（“职业名”，1）的元组，再通过 reduce 将职名相同的值相加，最后得到每个职业对应的人数。同理统计性别的人数，不同年龄段的人数。

统计每个年份电影的数量：

数据集中有些电影没有年份，所以需要进行数据清洗，过滤掉没有年份信息的电影，再通过 Map Reduce 得到结果。

分析 Rating 数据：

利用 mapReduce 思想统计分析 Rating,从而得出最大评分，最小评分，平均评分等数据。

推荐系统

常见的推荐的算法有三种：基于人口统计学的推荐、基于内容的推荐和协同过滤。其中协同过滤是最经典最常用的，协同过滤分为三类：

1. 基于用户的协同过滤算法
2. 基于物品的协同过滤算法
3. 基于模型的推荐算法 ALS

本项目采用协同过滤 ALS 算法

ALS 算法通过矩阵分解的方法来进行预测用户对电影的评分。同时考虑了 User 和 Item 两个方面，在此项目中 Item 即为 Movie. 根据数据集，建立 user* Movie 的 m*n 的矩阵，得到稀疏评分矩阵，将稀疏评分矩阵分解为用户特征向量矩阵和产品特征向量矩阵的乘积，交替使用最小二乘法逐步计算用户/产品特征向量，使得差平方和最小，通过用户/产品特征向量的矩阵来预测某个用户对某个产品的评分。

模型评估：

对整个 movielens 数据集进行评分预测计算 RMSE，来看 RMSE 的收敛情况来推断算法的训练效果。

项目实现

数据分析

首先创建 SparkContext 对象，调用 textFile 函数读取数据，用 map 对数据进行解析。比如用户的数据，通过 map 分别提取用户性别，职业等属性。然后使用 reduceByKey 函数统计数据。处理其他数据同理。使用 numpy 库进行一些数组操作。

画图可以使用 matplotlib 库，但是效果不太好，因此使用了 ECharts，这是一个使用 JavaScript 实现的开源可视化库。

推荐系统

Spark 的 ml 库集成了 ALS 算法，因此可以直接调用。将数据随机分为两部分，80%用来训练 ALS 模型，20%用来测试，设置最大迭代数为 5，正则化参数为 0.01。

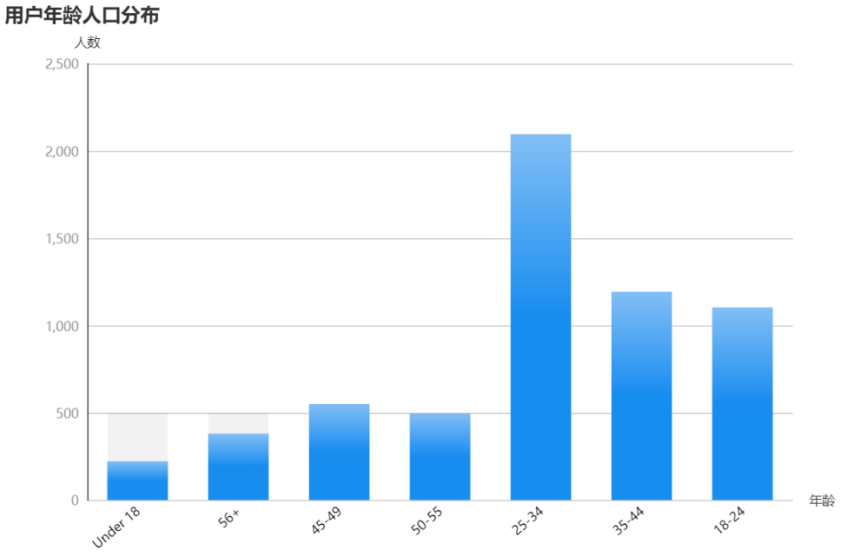
调用 fit 函数进行模型训练，设置冷启动策略为 drop 模式，得到模型，调用模型的 transform 方法可进行预测。

调用 recommendForAllUsers 方法为每个用户推荐电影，调用 recommendForAllItems 方法为每部电影预测可能喜欢的用户。设置 repartition 使得所有结果都输入到一个文件中。

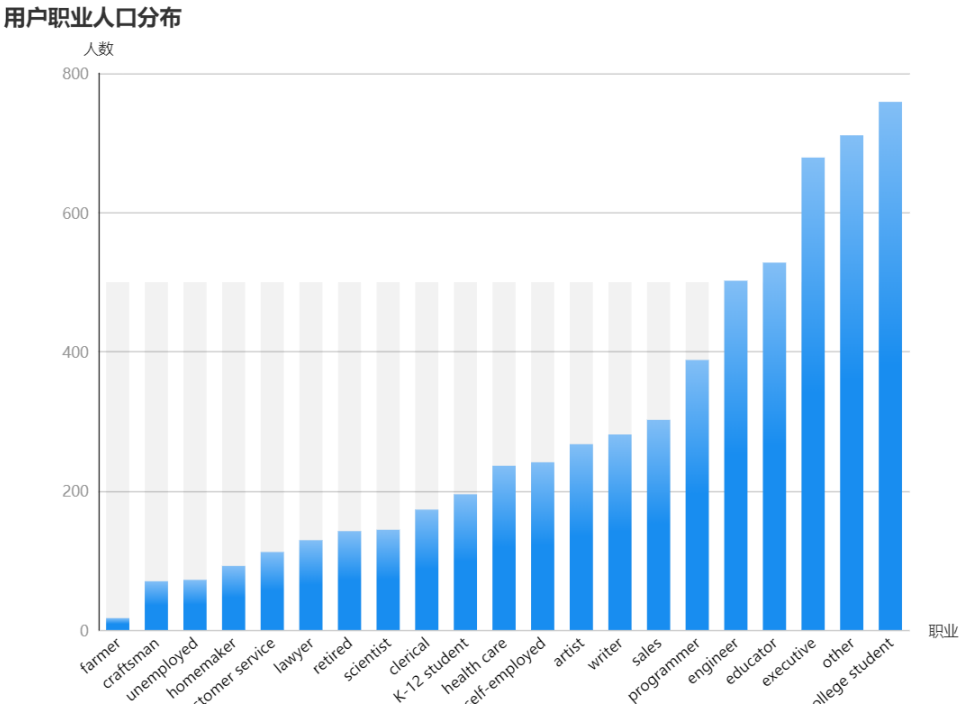
使用 spark 中的 RegressionEvaluator 计算得到模型的根方误差。

运行结果

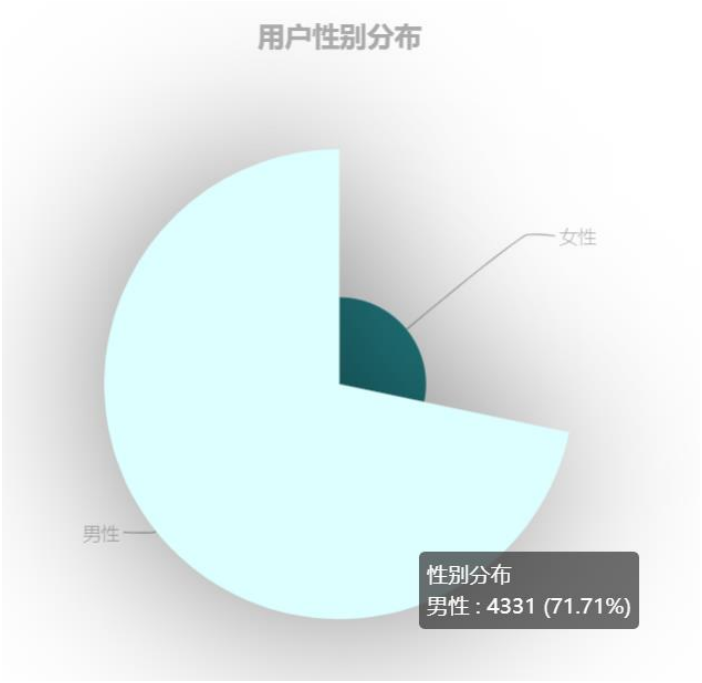
数据分析



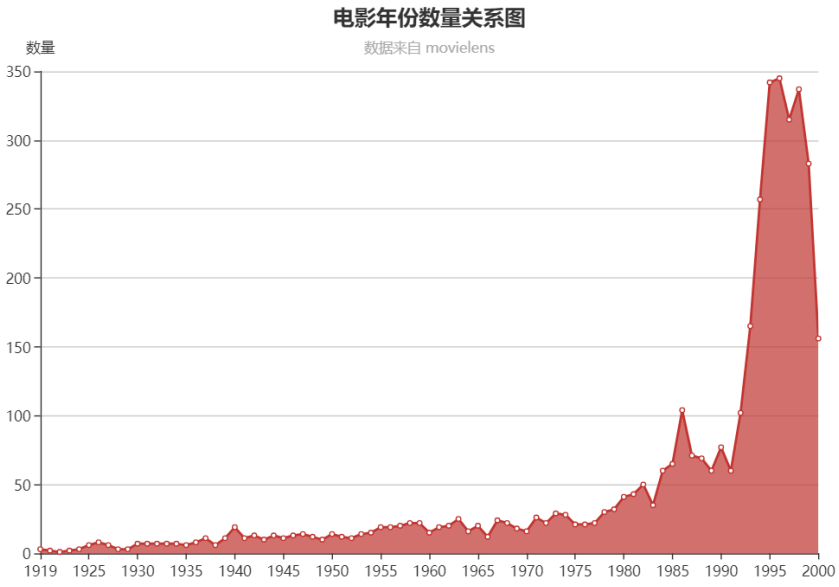
可以看到，用户年龄大多在 18-44 之间，其中 25-34 年龄段人数最多。



· 可以看到，用户中大学生、教育工作者、工程师等比较多，农民、工匠比较少。



可以看到，用户中男性占大多数，达到七成。

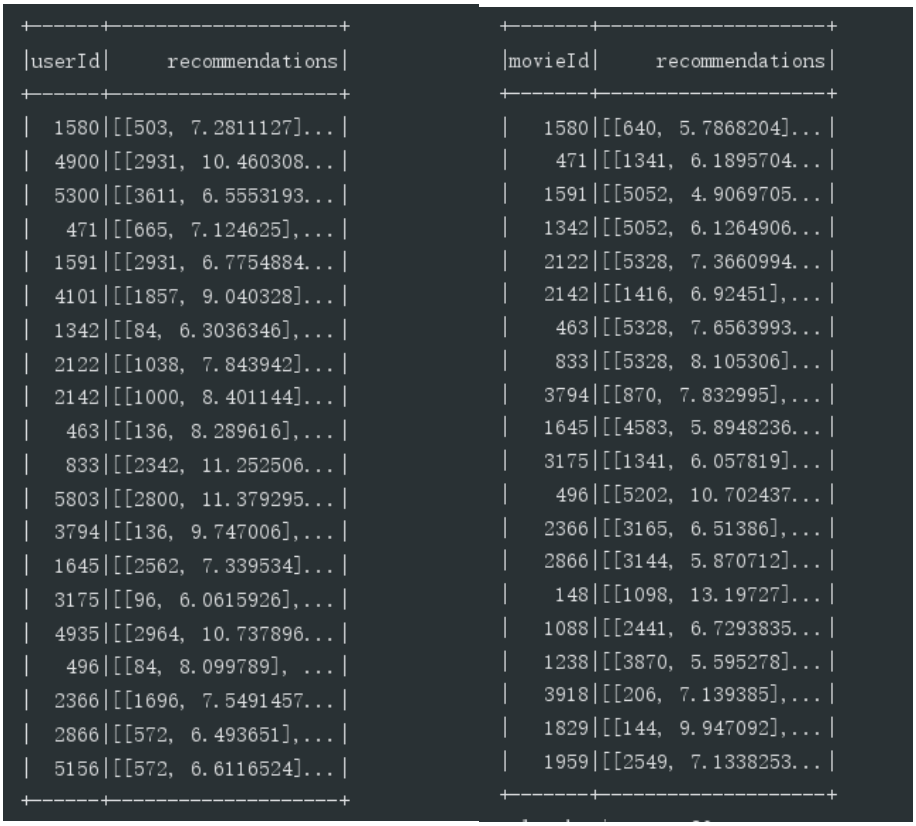


数据集中的电影发行时间大多数在 1995-2000 之间，没有 2000 以后的，也可以看出数据集比较老旧。

使用 Map Reduce 思想，计算得到评分的相关数据:

最低评分	1
最高评分	5
平均评分	3.58
中位评分	4

推荐系统



userId	recommendations
1580	[[503, 7.2811127]...
4900	[[2931, 10.460308...
5300	[[3611, 6.5553193...
471	[[665, 7.124625],...
1591	[[2931, 6.7754884...
4101	[[1857, 9.040328]...
1342	[[84, 6.3036346],...
2122	[[1038, 7.843942]...
2142	[[1000, 8.401144]...
463	[[136, 8.289616],...
833	[[2342, 11.252506...
5803	[[2800, 11.379295...
3794	[[136, 9.747006],...
1645	[[2562, 7.339534]...
3175	[[96, 6.0615926],...
4935	[[2964, 10.737896...
496	[[84, 8.099789], ...
2366	[[1696, 7.5491457...
2866	[[572, 6.493651],...
5156	[[572, 6.6116524]...

movieId	recommendations
1580	[[640, 5.7868204]...
471	[[1341, 6.1895704...
1591	[[5052, 4.9069705...
1342	[[5052, 6.1264906...
2122	[[5328, 7.3660994...
2142	[[1416, 6.92451],...
463	[[5328, 7.6563993...
833	[[5328, 8.105306]...
3794	[[870, 7.832995],...
1645	[[4583, 5.8948236...
3175	[[1341, 6.057819]...
496	[[5202, 10.702437...
2366	[[3165, 6.51386],...
2866	[[3144, 5.870712]...
148	[[1098, 13.19727]...
1088	[[2441, 6.7293835...
1238	[[3870, 5.595278]...
3918	[[206, 7.139385],...
1829	[[144, 9.947092],...
1959	[[2549, 7.1338253...

对每个用户推荐十部电影的数据保存在 users.txt 文件中。
对每部电影预测十名可能喜欢该电影的用户保存在 movies.txt 文件中。

Movielens 中的评分数据为 0-5，而训练出的评分会大于 5，因为 ALS 并没有设置最大值，可以通过归一化将数值限定在 0-5 之间，但这不是必要的，因为推荐系统看的不是具体的数值，只要用户对不同电影的预测值有区分就可以。

模型最后得到 rmse 的值为 0.8940475736136243。

项目评价

优点

1. 运算速度快，Spark 基于内存计算，不需要每步都写入磁盘。

缺点

1.推荐系统是一个离线算法，无法进行模型的增量训练。
2.推荐模型基于历史数据，所以对于新用户和新电影都有“冷启动”问题。

改进

如果有时间可以做这些方面的改进，使项目更加人性化和实用

1. 使用图形化界面，用户可以查看统计结果并且系统根据给定的用户 ID 给出推荐。
2. 提供在线模型训练，用户可以上传的自己数据，设置模型参数，即在线建模。

项目总结

课程设计从一开始就纠结选题，有些题目找不到合适的数据于是放弃了，通过不断的摸索，最后确定了该选题。

该项目对 movielens 数据进行了多方面的分析，并且利用 Spark 自带 ml 库中的 ALS 进行推荐电影，但是数据集比较老旧，实用性不高，但是是个很好的练习项目。通过项目，可以熟悉 hadoop、spark 的相关配置及使用，了解 Hadoop 和 spark 的差别，spark 的不同运行模式，熟悉 spark 中的数据结构以及相关的操作。深入理解了 Map Reduce 的思想。

在项目遇到了许多问题，主要都是配置相关的，大部分问题都可以通过谷歌解决，还有一些问题需要自己不停地尝试，总结需要注意的点：spark 不要搭配 Java9，应该使用 java8，环境变量配置中不要包括空格，用 Maven 构建项目打包 jar 运行找不到主类时可能是 pom.xml 文件编写问题。

第一次使用 hadoop、spark 框架，一切从零开始，因此项目还有很多不足的地方，希望今后不断学习，设计出更加出色、更加实用的项目。