协同过滤算法

1. 算法介绍

协同过滤简单来说是利用某兴趣相投、拥有共同经验之群体的喜好来推荐用户感兴趣的信息，个人通过合作的机制给予信息相当程度的回应（如评分）并记录下来以达到过滤的目的进而帮助别人筛选信息，回应不一定局限于特别感兴趣的，特别不感兴趣信息的纪录也相当重要。其核心思想是如果目标用户和某一用户再某些物品评分上很相似，那么目标用户对新物品的评分与该用户对新物品的评分也是类似的。

协同过滤方法可以大致分为两类：基于邻域的方法和基于模型的方法。

1. 基于邻域

在基于领域的协同过滤方法中，用户对物品的历史评分数据可以用来预测用户对新物品的评分。基于领域的方法包括两种：基于用户的推荐和基于物品的推荐。在基于用户的协同过滤方法中，目标用户对某一未接触物品的感兴趣程度，是由和用户具有相似评分模式的其他用户（近邻用户）对该物品的评分来估计的。基于物品的协同过滤方法，是根据某一用户对目标物品相似物品（近邻物品）的评分来预测用户对该目标物品的评分。

与基于邻域的方法不同的是，基于模型的方法使用评分信息来学习预测模型。主要思想是使用属性构建用户和物品之间的联系，这里的属性表示用户和物品的潜在特征，比如用户喜欢的类别和物品所属的类别。

基于邻域的协同过滤有以下优点：

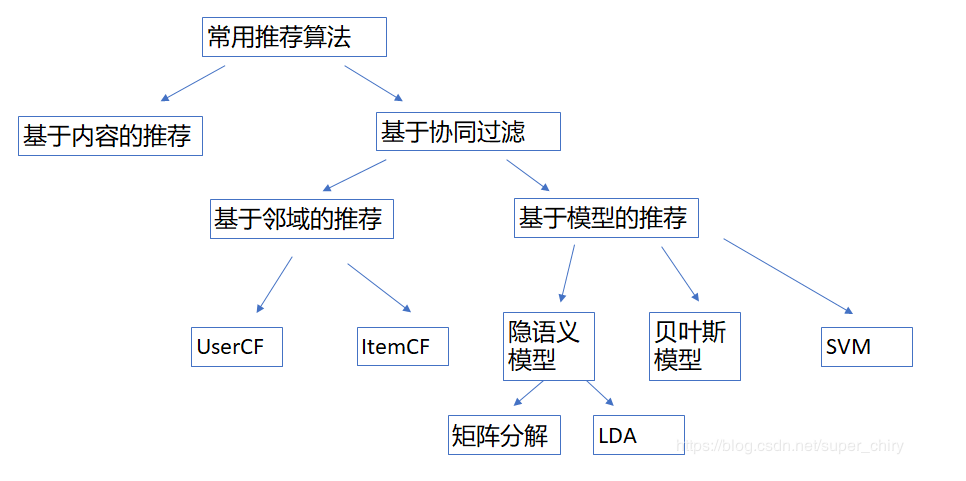
* 简单性：直接且容易实现
* 可解释性：推荐结果具有直观的解释性理由
* 高效：基于模型的协同过滤系统在训练阶段需要消耗大量资源
* 稳定：一旦计算完成，后续可进行增量计算。对于具有评分的新物品或新用户加入时，仅需要计算这些新对象和系统已有对象的相似度即可。

基于邻域的协同过滤的缺点：

* 覆盖受限：由于计算两个用户间的相似度是基于他们对相同物品的评分，而且只有对相同物品进行了评分的用户才可以作为近邻。这样仅仅被近邻用户评价过的物品才会被推荐，推荐方法的覆盖将受到限制
* 对稀疏数据的敏感：基于邻域的推荐方法的准确性会因为评分数目的缺少而受到影响。由于用户通常只是对一小部分物品进行了评分，导致评分数据具有稀疏性。一方面稀疏数据导致在推荐时只使用了非常有限的近邻，另外，相似度计算也只是依赖了少量的评分，导致相似度存在偏差，影响推荐结果

1. 基于模型

用于解决推荐任务的基于模型的推荐方法有很多，包括贝叶斯聚类、潜在语义分析、支持向量机以及奇异值分解等等。



这里介绍一下对用户-物品评分矩阵进行分解的推荐模型。在这种方法中，一个**U**x**I**的用户-物品评分矩阵R（矩阵秩为n）可近似表示为PQT，其中P是一个和基于邻域的协同过滤类似，对用户-物品评分矩阵分解，可看作在**U**x**k**用户因子矩阵，Q是一个**k**x**I**物品因子矩阵。这样矩阵P的第u行表示用户u在向量空间k上的映射向量。同样矩阵Q的第j行表示物品j在向量空间k上的映射向量。

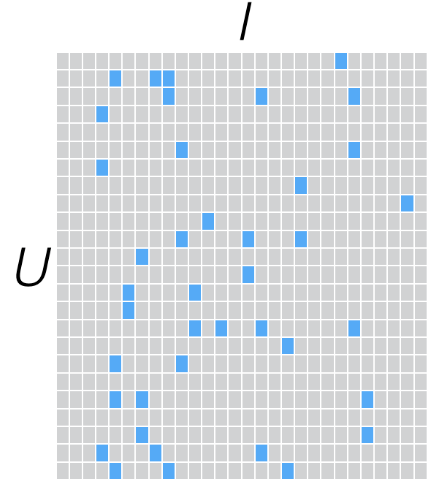
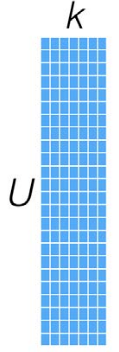
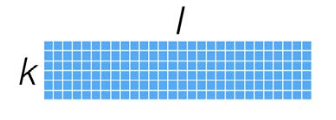
假设我们有一批用户数据，其中包含m个User和n个Item，则我们定义Rating矩阵，其中的元素表示第u个User对第i个Item的评分。

在实际使用中，由于n和m的数量都十分巨大，因此R矩阵的规模很容易就会突破1亿项。这时候，传统的矩阵分解方法对于这么大的数据量已经是很难处理了。

另一方面，一个用户也不可能给所有商品评分，因此，R矩阵注定是个稀疏矩阵。矩阵中所缺失的评分，又叫做missing item。

针对这样的特点，我们可以假设用户和商品之间存在若干关联维度（比如用户年龄、性别、受教育程度和商品的外观、价格等），我们只需要将R矩阵投射到这些维度上即可。实际上我们并不需要显式的定义这些关联维度，而只需要假定它们存在即可，因此这里的关联维度即上文中的潜在向量空间k，又被称为隐变量。k的典型取值一般是20～200。

用图来表示上述思想就是，用两个低维矩阵来尽可能的还原一个高维稀疏矩阵。

≈x

矩阵P和Q可以通过最小化以下函数来获取：

这和基于邻域的协同过滤类似，在潜在向量空间k上，可计算用户间的相似度和物品间的相似度，并根据结果进行推荐。评分矩阵分解属于User-Item CF，也叫做混合CF。它同时考虑了User和Item两个方面。这种方法可以解决冷启动和受限覆盖的问题。

上式中第二项为L2正则项，保证数值计算稳定性，防止过拟合。上式最小化问题的工程解法为交替最小二乘法ALS(Alternative Least Square)。

1. 交替最小二乘法ALS

ALS基本思想是对稀疏的用户-物品评分矩阵进行模型分解，评估出缺失项的值，以此来得到一个基本的训练模型。然后依照此模型可以针对新的用户和物品数据进行评估。ALS是采用交替的最小二乘法来算出缺失项的。交替的最小二乘法是在最小二乘法的基础上发展而来的。

上述公式直接优化是很困难的，因为P和Q的二元导数不容易计算，这是可以利用类似坐标下降法的算法，先固定变量P，对另一个变量Q进行优化，再反过来固定Q，再来优化P，持续迭代，直至收敛，即上述公式值的变化很小。因为这个迭代过程，交替优化P和Q，因此又被称作交替最小二乘算法（Alternating Least Squares，ALS）。

单变量的最小值优化问题就比较简单了，可采用梯度下降法。

Spark mllib库实现了矩阵分解的ALS算法，可进行直接使用。

1. 推荐系统架构

基于ALS协同过滤的视频推荐系统架构。



Prediction IO是以Spark mllib为基础，集成了数据、模型存储及模型在线服务功能的通用机器学习框架。

Prediction IO从系统MySql数据库读取用户对观看影视的评分记录，使用Spark ALS算法库实现评分矩阵分解，得到用户、视频的潜在特征向量，并将结果保存至Mysql数据库。Predition IO带有模型在线服务功能，使用restful的http服务对后台提供推荐接口，实现基于用户历史评分记录的视频推荐和视频详情页的相似视频推荐。

1. 推荐系统部署
2. 整个视频网站及推荐服务需要部署在linux环境下
3. 完成系统前后端部署
4. Prediction IO安装部署：

* 将附带的Prediction IO压缩包拷贝至自定义路径下，并解压缩
* 配置系统环境变量：

打开/etc/profile文件，并增加下面两行

export PIO\_HOME={上一步中的Prediction IO文件夹绝对路径}

PATH=$PATH:PIO\_HOME/bin

运行pio eventserver &启动precition io服务，执行pio status检查predition io是否成功运行

* 将项目movieRecommend压缩包上传至自定义路径下，并解压缩。

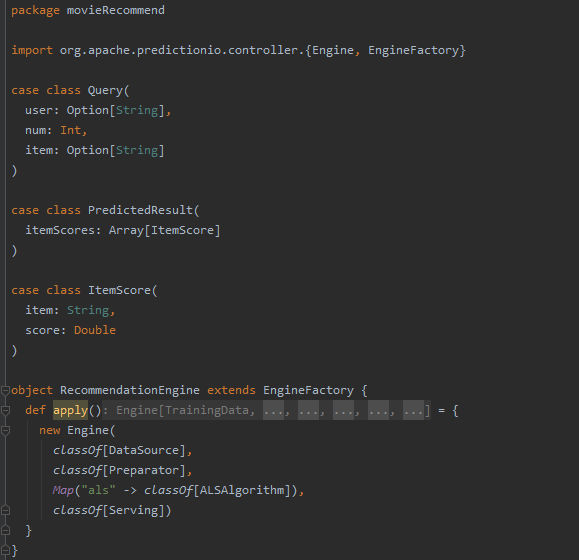
切换至movieRecommend目录下，执行run.sh，开始数据读取，模型训练以及打开在线推荐服务

运行check.sh，若返回数据则推荐服务正常启动

1. MovieRecommend代码讲解

本项目以Prediction IO框架为基础，使用Scala语言编写，Prediction IO框架实现了数据获取、处理、模型训练与部署功能流水线，故我们只需要关系各部分功能实现就可以了。

1. Engine.scala



这个文件定义了本项目RecommendationEngine的结构：

DataSource🡪Preparator🡪ALSAlgorithm🡪Serving。

还定义了对外服务接口的传入参数格式Query，接口输出结果格式PredictedResult。

1. DataSource.scala

这个文件实现了从数据源读取数据，并将数据处理成定义的格式TrainingData。

1. Preparator.scala

这个文件是对DataSource生成的TrainingData进行处理，在传入ALSAlgorithm的train函数进行算法训练。这里不对数据做任何处理，直接传入train。

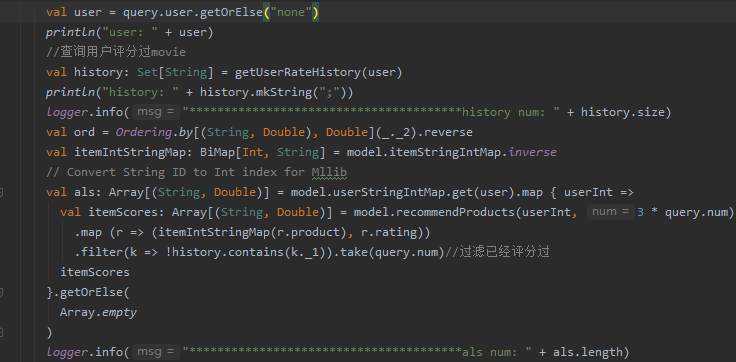
1. Serving

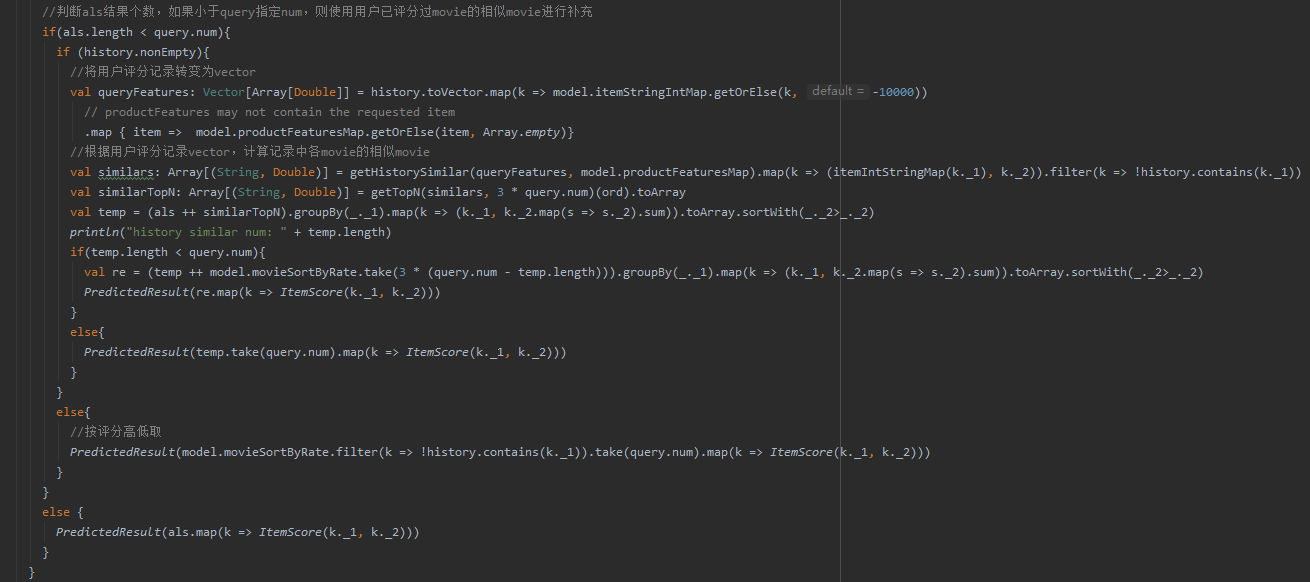
这里定义了对外推荐服务的 查询参数Query和返回的结果PredictedResult。

1. ALSAlgorithm

这里实现了两个函数，一个是train，对传入的数据采用ALS算法进行训练，生成模型；

另一个是predict函数，基于train的模型，根据Serving传入的查询参数，返回响应的推荐结果。





Predict的推荐分为两部分，一部分是为基于用户的历史行为进行视频推荐，一部分是针对某个视频推荐相似视频。

