学校代码：10289
分 类 号：TP311.5
密　　级：公 开
学　　号：182070022

江苏科技大学
硕 士 学 位 论 文

基于隐语义模型推荐算法研究

孔 欢

研 究 生 姓 名 孔 欢　　导 师 姓 名 黄 树 成
申请学位类别 工 学 硕 士　　学位授予单位 江 苏 科 技 大 学
学 科 专 业 软 件 工 程　　论文提交日期 2020 年 12 月 15 日
研 究 方 向 推 荐 系 统　　论文答辩日期 2021 年 3 月 12 日
答辩委员会主席 韩 斌　　评 阅 人 盲 审
盲 审

2021 年 3 月 8 日

江苏科技大学

分类号：TP311.5

密　级：公开

学　号：182070022

基于隐语义模型推荐算法研究

学生姓名　孔欢

指导教师　黄树成 教授

江苏科技大学

二〇二一年 三月

Research on Recommender Algorithms Based on Latent Factor Model

Submitted by

Kong Huan

Supervised by

Professor ***Shucheng Huang***

Jiangsu University of Science and Technology

March, 2021

江苏科技大学学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

年　　月　　日

江苏科技大学学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权江苏科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于：

(1)保密□，在___年解密后适用本授权书。

(2)不保密□。

学位论文作者签名：　　指导教师签名：

年　月　日　　年　月　日

摘 要

随着互联网的迅猛发展，大数据时代已经来临，但是人们无法从众多数据中选出有价值的信息，这就出现了“信息过载”问题。然而推荐系统的出现能够很好的解决该问题，它能够根据用户的历史行为为用户做出合理的推荐，推荐算法是推荐系统的灵魂，隐语义模型是推荐算法中最常见的矩阵分解模型，自从 Netflix 推荐算法大赛诞生以来，该算法就备受学者热爱。但是数据的稀疏性对传统的隐语义模型性能大打折扣，除此之外，传统的隐语义模型在训练时需要遍历整个训练集，导致模型寻优时间也会十分漫长。

为了解决传统隐语义模型推荐算法带来的种种问题，本文从模型的两个维度进行优化。首先通过对算法本身进行优化，有效解决了数据稀疏性对算法精确度的影响，其次对算法的训练方式加以改良，大大缩短了模型的训练时间。本文主要研究内容如下：

（1）为了解决数据稀疏性对算法精确度带来的影响，本文从正负样本选取这一因素进行改善，通过聚类技术将具有相同兴趣的用户聚集在一起，对每个用户群体中的热门项目进行排序，选取自身未有过行为的项目作为负样本进行训练。最后通过实验验证了改进后的算法在较稀疏的数据集中的精确度具有优异的表现。

（2）通过对传统隐语义模型的训练方式进行分析，发现其每次迭代都需要遍历整个数据集，这极其影响模型寻优时间并且在大规模数据集中算法的精确度会大大下降，为了解决以上问题，本文提出了增量样本的训练方式，每次迭代仅仅需要遍历部分数据集，不仅能够缩短寻优时间，而且能够在大规模数据集中保持良好的推荐性能。但是该训练方式仍然存在着不足，接着融合以上两种训练方式的优点，添加动量因素其继承了上次迭代梯度的下降值，在不同数据规模的 MovieLens 数据集上进行实验，实验结果验证了改进后的训练方式不仅能够降低算法的寻优时间，而且能够提升算法的精确度。

（3）通过整合改进后的算法和改进后的训练方式，提出了一种混合模型，实验证明混合模型性能优于单一模型。

关键词：隐语义模型；矩阵分解；聚类技术；数据稀疏性；寻优时间

Abstract

With the rapid development of Internet, the era of big data has come. However, people can not choose valuable information from the public data, which leads to the problem of "information overload". However the emergence of the recommendation system can very good solve this problem, it can be based on user behavior make reasonable recommendations for the user, the history of recommendation algorithm is the soul of the recommendation system, the latent factor model is the most common recommendation algorithm model of the matrix factorization, since competition born Netflix recommendation algorithm, this algorithm have been scholars love. However, the sparse data greatly reduces the performance of the traditional cryptic meaning model. In addition, the traditional cryptic meaning model needs to traverse the entire training set during the training, resulting in a long optimization time for the model.

In order to solve the problems caused by the recommendation algorithm of traditional latent factor model, this paper optimizes the model from two dimensions. Firstly, by optimizing the algorithm itself, the influence of data sparsity on the accuracy of the algorithm is effectively solved. Secondly, the training mode of the algorithm is improved, which greatly reduces the training time of the model. The main research contents of this paper are as follows:

(1) In order to solve the data sparse on accuracy of the impact of this algorithm, this paper from the positive and negative samples we do to improve on this factor, through clustering technology to users with similar interests together and for each user group of the popular project to sort, select the project itself had a behavior as negative samples for training. Finally, experiments show that the improved algorithm has excellent accuracy in a sparse data set.

(2) Through the training mode of traditional latent factor model were analyzed, and found that the each iteration the need to traverse the entire data set, this optimization time and impact model and the accuracy of the algorithm in a large-scale data set will be greatly decreased, in order to solve the above problem, this paper puts forward the incremental sample way of training, only need to traverse the part of the data set, each iteration optimization can not only shorten the time, but also can recommend to keep good performance in large-scale data set. But the training way still exist insufficient, in this paper, the advantages of integration of above two kinds of training methods, adding momentum factor of its inherited the previous

iterative gradient values, in different size MovieLens data set on the experiment data, the experimental results verify the effectiveness of the improved training method can not only reduce the optimization time of algorithm, but also can improve the accuracy of the algorithm.

(3) By integrating the improved algorithm and the improved training method, a hybrid model is proposed, and the experiment proves that the hybrid model has better performance than the single model.

Keywords: latent factor model; matrix factorization; clustering technology; data sparsity; optimization time

目 录

Contents

第 1 章 绪论

1.1 课题研究背景及意义

近期中国互联网络中心（China Internet Network Information Center，CNNIC）在北京发布关于我国互联网络发展状况的统计报告，如图 1.1 所示。根据报告中统计的数据，截至 2018 年 6 月，我国网名规模已经突破 8 亿，到了 2020 年 6 月，我国网民人数达到 9.4 亿，较同年 3 月增长 3625 万，互联网普及率高达 67.0%，相比于同年 3 月提升 2.5 个百分点[1]。

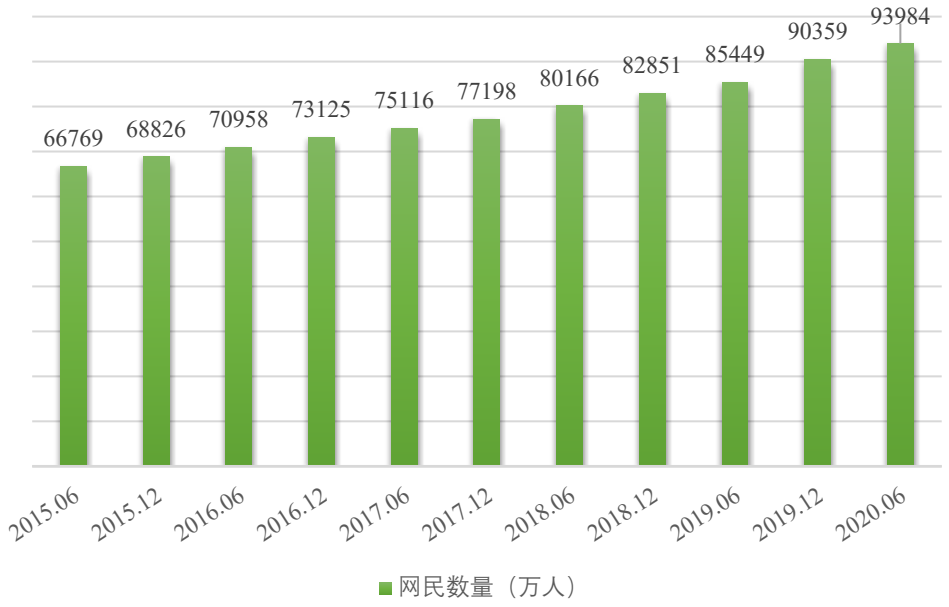

图 1.1 互联网发展统计图

Figure 1.1 Internet development statistics

随着互联网的快速发展，网络媒体也正以不可估计得速度进行更新，其中比较典型的是电影、视频以及音乐媒体。但是用户如何从众多的媒体中找到适合自己的并且自己感兴趣的已经成了一个新的难题，这同样对某些企业提出了新的挑战。像爱奇艺、优酷、腾讯视频这类 APP 面对海量的用户，就必须要根据用户经常浏览的资源为用户推荐可能让用户感兴趣的话题。因此设计一个推荐系统[2]的理念也就应运而生了。

随着电商系统的迅猛发展，各种电商网站提供给用户的选择越来越多，同时其结构也变得越来越复杂。这样导致人们从海量的数据中往往抓不住重点，电商平台也难以和用户面对面交流，这样就会导致电商对用户的信息理解错误，可能会阻碍该电商平台的发展。电子商务推荐系统不但能够帮助用户更迅速的从那些繁杂的海量的数据

中筛选出自己感兴趣的商品信息，而且能够帮助商家为用户推荐用户潜在地可能喜欢的商品。同样商家也可以根据大量用户对某种商品的喜爱程度，对这种商品进行打折促销活动。因此推荐系统的发展离不开电子商务的支持，电子商务的运营也离不开推荐系统的完善，两者是相辅相成的。

随着大数据时代的来临，人们已经很难从这些海量的数据中找到自己感兴趣的，这样就出现了“信息过载[3]（Information Overload）”问题，导致信息利用率降低。推荐系统的出现让人们从“大数据”中看到了“曙光”。然而对于个性化推荐系统的核心，推荐算法的研究一直是国内外的研究热点。2006 年 10 月, 美国 Netfilx 公司举办一场奖金约为 100 万美元的推荐算法竞赛, 竞赛的要求将 RMSE 降低到 0.8572 或更低。这极大的推动了推荐算法的发展。

目前基于协同过滤的推荐算法以及基于内容的推荐算法技术已经成熟，广泛应用在新闻、电商、视频等各个领域[4]。近期抖音备受各类群体的热爱，不同年龄段的人刷到的内容是完全不同的，这都要依赖于推荐算法的发展。但是目前推荐算法仍然存在瓶颈，提高算法的精确率是目前绝大多数论文研究的主要问题。当系统中注册了一个新用户，系统没有该用户的任何行为数据，这时候推荐系统不能给出较好的推荐，这就是典型的“冷启动问题[5]”。目前很多系统使用第三方登录，这样能够获得该用户在其他平台的行为数据，据此做出推荐。对于电影推荐系统来说，世界上电影的规模是非常庞大的，而用户也非常多，但是不可能一个用户去观看所有的电影，这样就导致了评分矩阵的“稀疏性[6-8]”，也会影响推荐性能，对此可能的解决方案有对矩阵进行分解降维。“冷启动问题”和“数据稀疏性问题”都会大大影响推荐算法的性能。

近年来由于深度学习的发展，不少学者将推荐算法与深度学习结合，效果极佳。为了给用户提供更优质的推荐，推荐算法也在不断发展。

1.2 国内外研究现状

1.2.1 国外研究现状

早在 20 世纪 90 年代，国外提出了推荐系统这一概念，经过近 30 年的不懈努力，已经在推荐系统领域取得了巨大成就，并且将理论知识付诸于实践，为商业领域谋得了巨大的价值。

1992 年，Goldberg[9]等首次提出协同过滤技术，并且将该技术应用到 Tapestry 邮件过滤系统，系统使用人员可以根据自己感兴趣的邮件进行订阅，而系统则会根据用户的订阅内容进行垃圾邮件过滤。随后 Resnick[10]等优化协同过滤技术将其应用到 Grouplens 的新闻推荐系统中，系统根据用户对新闻的评分数据进行推荐。2007 年，Mehta[11]等提出了 AdWords 系统，该系统根据用户搜索的关键字为用户推送一些相似

的广告信息。

一般推荐算法被分为三类：基于内容的推荐算法[12,13]、基于协同过滤的推荐算法[14]、混合推荐算法[15]。基于内容的推荐算法一般是通过用户或者物品属性信息构建的用户画像（User Profile）或者物品画像（Item Profile）进行个性化推荐。Debnath[16]等在基于内容的推荐算法中研究权重选取对算法性能的影响；Cramer[17]等研究了系统透明度对用户信任度的影响；Wu[18]等在 TF-IDF 中提出了一种新的概率检索模型；Martínez[19]等提出了一种多粒度语言的内容推荐模型，该模型不强迫用户使用某种特定的语言进行建模；Zenebe[20]等将模糊集理论应用到内容推荐模型中；Mooney[21]等描述了一个基于内容的图书推荐系统，该系统利用信息抽取和机器学习算法进行图书分类；与此同时 Cano[22]设计了一款音乐推荐系统。

与基于内容的推荐不同，协同过滤推荐算法主要是利用用户的历史行为数据进行推荐，其中最常见的协同过滤算法有基于用户的协同过滤、基于项目的协同过滤、基于模型的协同过滤。基于用户的协同过滤和基于项目的协同过滤都是利用相似度计算进行推荐的。2004 年，Sugiyama[23]等人将基于用户的协同过滤技术应用到搜索引擎中，通过分析用户一天的浏览历史，为每个用户生成不同的搜索结果；Sarwar[24]等研究协同过滤技术在电商网站的应用，且发现随着网站用户越来越多，基于用户的协同过滤算法精确度会明显降低，进而提出了基于项目的协同过滤算法，利用项目间的相似度进行推荐。此外，基于模型的协同过滤技术一般使用统计学、机器学习[25]、聚类分析、回归模型、矩阵分解等来进行数据挖掘。

矩阵分解是最常见的基于模型的协同过滤推荐算法。2000 年，Sarwar[26]等人首次将奇异值分解（Singular Value Decomposition, SVD）模型应用到推荐系统中。这种方式利用奇异值将评分矩阵分解为两个低维矩阵，在很大程度上解决了数据稀疏性问题，但是随着系统的不断壮大，SVD 分解效率明显降低；几年后 Funk[27]提出了基于正则化的奇异值分解（Regularized Singular Value Decomposition, RSVD）算法，即隐语义模型（Latent Factor Model, LFM），后来学者对此模型做了不同程度的优化。Koren[28]考虑用户评价历史的物品信息，提出 SVD++，后来又融入时间信息进行改进提出了 TimeSVD++；Rowe[29]提出了基于语义 SVD++模型，该模型利用推荐项目的语义类别作为给定用户的先验因素；Salakhutdinov[30]等人将概率模型应用到矩阵分解中，提出了基于概率矩阵因子分解模型（Probabilistic Matrix Factorization, PMF），次年提出了基于概率矩阵因子分解模型的全贝叶斯处理方法[31]（Bayesian Probabilistic Matrix Factorization， BPMF），该算法利用马尔科夫链蒙特卡洛方法训练贝叶斯 PMF 模型。

基于内容的推荐算法根据用户的标签信息生成对应的用户画像，因此该算法不存在冷启动问题，但是生成用户画像这一工作需要大量的时间和人力；然而基于协同过滤的推荐算法需要对用户的历史行为分析方能做出推荐，该算法可以通过离线分析，

但也存在着数据稀疏性、冷启动等问题。为此，Burke[32,33]提出了一种混合推荐算法，该算法结合了内容推荐和协同过滤技术的优点，在市场上应用广泛，并且绝大多数推荐系统都是使用多种推荐算法混合而成。算法混合策略一般可以分为切换[34]、加权[35]、交叉混合[36]、串联层叠[37]、元级[38]等方式。

1.2.2 国内研究现状

相比国外而言，国内的推荐系统起步相对较晚，但是随着互联网的迅猛发展以及亚马逊网站推荐系统的成功使用，以阿里巴巴为首的电子商务网站开始了对推荐系统的研究。阿里、腾讯、美团、字节跳动等公司在推荐系统领域均取得了巨大成就。2006年，当当网率先将音频、视频等信息推荐给用户，为用户提供个性化推荐，该系统备受广大用户的喜爱；2008 年 9 月，阿里上架了“淘宝”这一 APP，APP 可以根据用户的浏览记录做出推荐；几年后，阿里巴巴开创“天池”个性化推荐大赛，首次举办就吸引了来自全国乃至世界各地的推荐算法爱好者，这极大地促进了我国推荐算法的发展；2016 年，阿里旗下 YunOS 的推荐算法团队在 ACM RecSys 上荣获冠军；QQ 应用也使用了“猜你认识”和“附近同城”为用户搜寻可能认识的朋友；美团作为一个 O2O 项目，也推出了“猜你喜欢”等栏目，根据用户的订单记录为用户做出推荐；近年来，抖音的迅猛发展也离不开推荐系统，抖音首页设置了三个栏目分别是“同城”、“关注”、“推荐”，这三个栏目也无一不使用个性化推荐系统。

在学术方面，国内的学术爱好者也对推荐系统的发展做出了很大的贡献。Chen[39]等人提出一种简单高效的索引结构和启发式信息检索技术算法，并将其应用在电子商务系统中；Huang[40]等对比了不同的协同过滤推荐算法在电商中的精确度；Zhou[41]和Zhao[42]等研究了基于 Hadoop 的并行化相似度计算和协同过滤方法；近年来，Lee[43]等结合评分和评论信息提出了基于评分和评论的协同过滤算法；Zhao[44]等使用改进后的*k*-means 聚类算法实现了协同过滤推荐；Ding[45]等人结合用户评分相似度,兴趣倾向相似度和置信度提出了一种基于用户属性和评分的协同过滤算法，该算法一定程度上解决了冷启动问题；Zeng[46]等利用改进后的皮尔森系数提出了基于奇异值分解的HybridSVD 算法，实验证明该算法能够更好的解决数据稀疏性问题；Zhai[47]等结合用户的隐式数据与资源标签解决传统算法的缺陷，使系统的推荐更具有个性化。Wang[48]等人结合用户上下文信息以及隐马尔代夫模型设计了基于因子分解机的推荐算法，由于融入上下文信息可以对用户兴趣信息进行有效特征提取，所以改进后模型在预测精确度方面比传统推荐算法预测准确率要高；Zheng[49]等利用 PMF 技术提出了一种结合用户相似度的个性化推荐算法；Wei[50]等通过用户与物品的交互图构建物品相似性网络，提出了一种融合物品信息的社会化推荐算法。

近年来，国内的推荐算法爱好者，也逐渐将深度学习相关知识融入到传统的推荐

算法进行改进，促进了推荐算法的进一步发展。

1.3 论文的主要内容

本文首先对基于协同过滤的推荐算法和基于内容的推荐算法进行介绍，包括介绍其定义、算法实现以及优缺点，并且对基于协同过滤的推荐算法进行扩展。对比协同过滤算法的缺点，引出本文主要研究的模型——隐语义模型，同时介绍了推荐算法的评测标准，最后对隐语义模型深入研究，并对其进行优化。在单一模型和混合模型均验证了算法的精确度。对单一模型提出了两个层面的优化，主要做的工作有如下三个方面：

（1）对传统的矩阵分解——奇异值分解进行研究，通过分析其不足引出隐语义模型。介绍 LFM 算法的实现，找到算法可能存在的优化方案。介绍 *k*-means 聚类算法的流程，并且将其应用在隐语义模型中。提出了一种带偏置的隐语义模型，该算法能极大的提升系统的覆盖率。

（2）介绍隐语义模型的两种样本训练方式：全量样本训练和增量样本训练。全量样本训练方式在数据集规模较小的情况下性能较好，但是训练时间过长，而增量样本训练方式在规模小的数据集上性能和全量样本训练方式相差不大，但是在大规模数据集下，性能要明显优于全量样本训练。通过结合全量样本训练和增量样本训练的优点提出了一种添加动量训练的算法，添加了用户动量 M、项目动量 N 以及其系数 β。其中 M、N 代表了上次迭代梯度的下降值，β 代表了接受上次迭代梯度下降的程度。

（3）将用户聚类的隐语义模型和添加动量训练模型方法融合，最后在 MovieLens 数据集上验证混合模型的准确率比单一模型高。

1.4 论文的结构安排

本论文的结构安排如下：

第一章 绪论。本章首先介绍了课题的研究背景，然后对课题的研究意义加以阐述，接着阐述了国内外对推荐算法的研究，最后对文章的主要内容和文章结构进行概括。

第二章 推荐算法及评估标准。本章首先介绍了基于协同过滤的推荐算法，其次介绍了基于内容的推荐算法，接着介绍了混合推荐算法的实现方式，引出隐语义模型。最后对推荐算法的评估标准进行阐述。

第三章 基于用户聚类隐语义模型的优化。首先介绍了矩阵分解的相关概念，包括奇异值分解和隐语义模型，然后对聚类算法进行详细的阐述重点介绍了 *k*-means 聚类算法，同时将聚类算法应用到隐语义模型中，最后通过证明算法的优越性。

第四章 结合动量训练优化后的隐语义模型。本章首先对隐语义模型中常见的两种

样本训练方式（全量样本训练和增量样本训练）加以阐述，并且分别分析了两种方法的优缺点，然后提取两种方法的优点，提出了一种结合动量训练的方式，接着将该方式分别整合到传统隐语义模型和用户聚类的隐语义模型中，最后对这些算法进行实验分析。

总结与展望。本章对论文进行总结概括，对结合动量训练用户聚类的隐语义模型不足进行阐述，以及对未来的研究方向进行展望。

第 2 章 推荐算法及评估标准

推荐系统就是利用用户的历史行为数据，来预测用户可能会感兴趣的物品。因为推荐算法是推荐系统的灵魂，所以设计一个好的推荐算法对于推荐系统来说是至关重要的。目前主流的推荐算法大致有基于协同过滤的推荐算法、基于内容的推荐算法、混合推荐算法等等。

基于协同过滤推荐算法是目前市场上使用最多的推荐算法，该算法利用用户或者项目之间的相似度进行推荐。该算法在推荐的精准度以及推荐的效果上都占据了明显的优势。但是该算法也存在着一些问题，比如系统冷启动以及稀疏的评分数据都一定程度上影响了算法的性能，从而给用户带来了不好的体验。

基于内容的推荐算法使用也很广泛。该算法利用用户和项目的特征进行推荐。该算法推荐的效果直观可见并且具有自适应好、推荐效果与历史行为数据成正比等优点。另一方面，该算法不存在协同过滤算法存在的数据稀疏性、冷启动等问题。然而，基于内容的推荐算法在面对一些较难提取特征的项目（提取音乐和视频的特征）就束手无策了。

2.1 基于协同过滤的推荐算法

Goldberg[9]等最早提出了“协同过滤（Collaborative Filtering，CF）”这一词，Tapestry是文献[9]中提出来的第一个基于协同过滤的推荐系统, 该系统是电子文档过滤系统，根据用户评价过的文档给其他用户推荐适合自己的文档。使用协同过滤的系统还有很多，比如 Amazon、Ringo 等系统。

基于协同过滤的推荐一般分为基于用户的协同过滤[51]（User-based Collaborative Filtering，UserCF）、基于项目的协同过滤[52]（Item-Based Collaborative Filtering，ItemCF）以及基于模型的协同过滤[53]（Model-Based Collaborative Filtering，ModelCF）。UserCF 算法将相同兴趣的人分为一组，把与 A 同一组的用户喜欢的项目都推荐给 A；ItemCF 算法将被相同用户喜欢的一部分项目分为一类，对于项目 A 和项目 B 都被共同用户喜欢，那么项目 A 和项目 B 被分为一类，所有喜欢项目 A 的用户也大概率地会喜欢项目 B；ModelCF 算法利用统计学及机器学习知识对用户行为数据建模分析进行推荐，比如聚类算法，决策树以及贝叶斯分类。

2.1.1 基于用户的协同过滤

基于用户的协同过滤旨在找到兴趣相似的用户，把他们归结成为一类群体。并且

以此为前提，分析得出目标用户的最近邻居 (最相似的若干用户) 对某个项目的评分进而预测出目标用户对该项目的评分[54]。该算法主要包括以下两个步骤：

（1）查找与目标用户有相似兴趣的用户集合；

（2）将用户集合中用户最感兴趣且目标用户未使用的项目推荐给目标用户。

UserCF 算法的核心是计算两个用户的爱好的相似度。这里，协同过滤算法通过分析用户的历史行为相似度来计算用户之间爱好的相似度。对于用户 u 和用户 v，令 $N(u)$ 表示用户 u 曾经有过正反馈的项目集合，$N(v)$为用户 v 曾经有过正反馈的项目集合。可以通过 Jaccard 公式简单地计算用户 u 和用户 v 的爱好的相似度，如式(2.1)所示：

$$w_{uv}=\frac{\left|N(u)\cap N(v)\right|}{\left|N(u)\cup N(v)\right|}. \tag{2.1}$$

计算用户的兴趣相似度后，UserCF 算法会给用户推荐和他兴趣最相似的 $\boldsymbol{k}$ 个用户喜欢的项目。由公式(2.2)可度量 UserCF 算法中用户 u 对项目 i 的感兴趣程度：

$$p(u,i)=\sum_{v\in S(u,\boldsymbol{k})\cap N(i)} w_{uv}r_{vi}. \tag{2.2}$$

其中，$S(u,k)$包含和用户 u 兴趣最接近的 $\boldsymbol{k}$ 个用户，$N(i)$是对项目 i 有过行为的用户的集合，w_{uv} 是用户 u 和用户 v 的兴趣相似度，r_{vi} 代表用户 v 对项目 i 的兴趣，因为使用的是单一行为的隐反馈数据，所以所有的 $r_{vi}=1$。UserCF 算法只有一个核心的参数 $\boldsymbol{k}$，表示的是对于每个用户选出 $\boldsymbol{k}$ 个和他兴趣最相似的用户，然后推荐 $\boldsymbol{k}$ 个用户感兴趣而自己没有行为的项目。

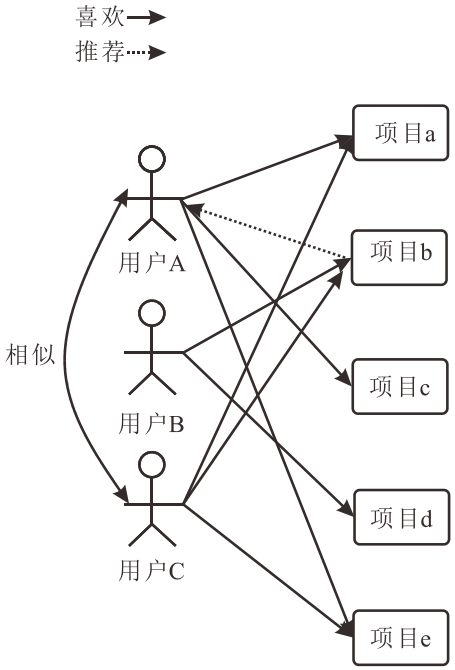

图 2.1 基于用户的协同过滤算法过程图

Figure 2.1 Process diagram of user-based collaborative filtering algorithm

基于用户的协同过滤推荐算法过程图如图 2.1 所示。用户 A、用户 C 都喜欢项目 a、项目 c、项目 e，除此之外用户 C 还比较喜欢项目 b，而用户 B 比较喜欢项目 b 和项目 d，通过相似度计算发现用户 A 和用户 C 是相似的。基于用户的协同过滤推荐算法会将相似的用户喜欢的项目推荐给目标用户，所以会将用户 C 喜欢而用户 A 没有过行为的项目推荐给用户 A。

2.1.2 基于项目的协同过滤

基于项目的协同过滤算法是业界应用最多的推荐算法。ItemCF 算法广泛应用在电商领域，比如亚马逊网、淘宝网、京东，其推荐算法的基础都是该算法。ItemCF 算法给目标用户推荐的是与当前用户喜欢的项目相似的项目集。对于项目 a 和项目 b，如果大部分用户都喜欢项目 a 和项目 b，那么项目 a 和项目 b 就是相似的。所以喜欢项目 a 的用户也很大可能性会喜欢项目 b。ItemCF 主要包括两个步骤：

（1）计算项目之间的相似度

（2）利用用户行为数据和项目相似度对目标用户进行推荐

由公式(2.3)定义项目的相似度：

$$w_{ij}=\frac{\left|N(i)\cap N(j)\right|}{\left|N(i)\right|}. \tag{2.3}$$

其中$|N(i)|$表示喜欢项目 i 的用户数，$\left|N(i)\cap N(j)\right|$表示既喜欢项目 i 又喜欢项目 j 的用户数，而 w_{ij} 则表示喜欢项目 i 的用户中喜欢项目 j 的用户所占比例。

如果项目 j 是一个热门项目时，也就意味着几乎所有的用户都喜欢项目 j。公式(2.3)中$\left|N(i)\cap N(j)\right|$的值等于喜欢项目 i 的用户数，那么 w_{ij} 将会无限接近 1。这样该项目与所有项目都相似，这对于一个推荐系统来说肯定是不好的。据此，对公式(2.3)进行优化：

$$w_{ij}=\frac{\left|N(i)\cap N(j)\right|}{\sqrt{\left|N(i)\right|\left|N(j)\right|}}. \tag{2.4}$$

通过公式(2.4)知当项目很热门时，$N(j)$会很大，这样 w_{ij} 会很小。也就意味着计算项目之间相似度时可以忽略热门项目，会使得推荐效果更好。

公式(2.5)计算了项目之间的相似度，ItemCF 算法通过公式(2.5)计算用户 u 对项目 j 的感兴趣程度：

$$p_{uj}=\sum_{i\in N(u)\cap s(j,\boldsymbol{k})} w_{ji}r_{ui}. \tag{2.5}$$

其中 $N(u)$表示的是用户 u 喜欢的项目的集合，$s(j,k)$表示与项目 j 最相似的 $\boldsymbol{k}$ 个项目的集合，w_{ji} 表示项目 i 和项目 j 的相似程度，r_{ui} 表示用户 u 对项目 i 的兴趣。对于一个

隐反馈数据集来说，若用户 u 对项目 i 有过正反馈，则 r_{ui}=1。ItemCF 算法会将与目标用户喜欢的项目相似最大的项目集推荐给该用户。

基于项目的协同过滤算法过程图如图 2.2 所示。图 2.2 中，用户 A、用户 B、用户 C 都喜欢项目 a 和项目 c，而用户 D 只喜欢项目 c。通过相似度计算得到项目 a 和项目 c 是相似的，根据基于项目得协同过滤算法，会将项目 a 推荐给用户 D。

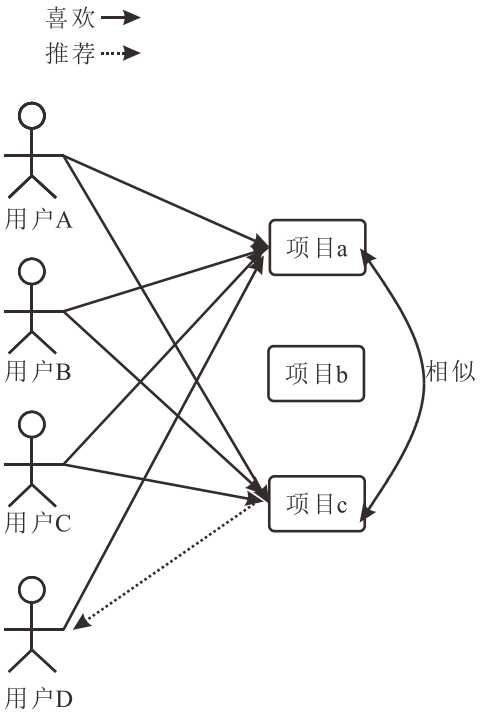

图 2.2 基于项目的协同过滤算法过程图

Figure 2.2 Process diagram of item-based collaborative filtering algorithm

2.1.3 基于模型的协同过滤

基于模型的协同过滤通常采用数据分析，机器学习等方式，比如聚类算法，决策树以及贝叶斯分类。它们都是对用户的行为数据进行建模训练，最终使用该模型为用户提供较为合理的推荐。Bereese 等在文献[55]提出了一个基于概率的算法，该算法从概率的角度看，基于模型的协同过滤算法视为计算用户评分的均值，对于比较活跃的用户，希望对未评分的项目进行评分预测，给定整数 $0 \le i \le m$，其公式如下：

$$R_{c,s} = E\left(R_{c,s}\right) = \sum_{i=0}^{m} pr\left(R_{c,s} = i \middle| R_{c,s'}, s' \in S_c\right) i \tag{2.6}$$

其中式(2.6)是在给定有过行为的项目评分的情况下将对项目 j 具有特定评分值的概率。

基于模型的协同过滤通过用户历史行为数据训练生成模型能够为用户提供实时推荐，并且也是很稳定的。但是，对于训练模型这一阶段耗时极长，当用户有了新的行为数据时，需要重新训练模型从而才能达到优质的推荐。所以，基于模型的协同过滤

不适用于用户极其活跃的推荐系统。

目前市场是流行的基于模型的协同过滤算法有关联规则、聚类算法、分类算法、矩阵分解等。

（1）关联规则

众所周知，大数据领域流传着一种尿布啤酒效应。在沃尔玛超市存在着一种奇怪的现象：尿布和啤酒放在一起售卖。在美国太太经常要求自己的丈夫在下班时买点尿布回家，丈夫在买尿布的同时也会买上自己爱喝的啤酒。商家通过一年来对用户消费数据的分析，发现有尿布的订单中也大多存在啤酒，于是商家将这两样东西放在了一起，神奇的是尿布和啤酒的销量都大幅上升。

上面的例子就是一种关联规则。能够通过找出用户购买的所有项目数据中频繁出现的项集活序列，来做频繁集挖掘，找到满足支持度阈值的关联物品的频繁 N 项集或者序列。如果用户购买了频繁 N 项集或者序列里的部分物品，那么可以将频繁项集或序列里的其他物品按一定的评分准则推荐给用户，这个评分准则可以包括支持度，置信度和提升度等。常见的关联规则算法有 Apriori，FP Tree 和 PrefixSpan 等。

（2）聚类算法

“物以类聚，人以群分”就是聚类算法的核心思想，通过将有相似行为的用户或者项目聚成一类，这种思想和基于用户的协同过滤和基于项目的协同过滤类似。基于用户聚类算法计算用户之间的相似度，将其进行排序，选择前 K 个用户的兴趣进行推荐。基于项目聚类算法计算项目之间的相似度，将其进行排序，选择前 K 个项目为用户进行推荐。常用的聚类推荐算法有 k-means，CURE，STING 和谱聚类等。

（3）分类算法

分类算法是基于概率模型实现的。为算法手动设置一个阈值，当评分大于等于此阈值时做出推荐，而当评分小于等于此阈值时不做推荐，进而将问题转换成了经典的二分类问题。最常用的分类算法是朴素贝叶斯。其思想是：对于给定的待分类项，计算所有该项目属于分类项的概率，概率最大的就是该项目的分类。

2.2 基于内容的推荐算法

与基于协同过滤算法不同，基于内容的推荐不需要获得用户的历史评分数据，这就在一定程度上解决了数据稀疏性问题，但是冷启动问题依然存在。基于内容的推荐算法首先获取项目的标签数据，然后通过标签信息计算项目间的相似度。这种算法一般用于文本处理，比如微博推荐、新闻推荐以及书籍推荐等。但是对于一些难以提取特征的对象，比如音乐、视频等显得非常束手无策。

基于内容的推荐，算法思想包括以下基本步骤：

（1）对用户和项目的信息进行一系列的提取，进行预处理，并对处理结果生成特征向量空间；

（2）对用户的与项目之间的历史交互行为进行处理，形成或者更新用户的历史交互信息的特征向量空间；

（3）通过选择合适的相似度度量方式来进行比较用户或者项目的特征向量空间，计算出个体之间的相似性；如果默认用户的兴趣可以通过他的历史行为数据集来直接得到，就可以直接找出与这个历史数据集相似度最大的项目，如果对推荐的项目的新鲜度和种类的多样性等有更多的要求可添加某些参数对模型进行优化或修正。

$$p_{u,i}=\mathrm{score}\left(User\left(u\right),\mathrm{Item}\left(i\right)\right). \tag{2.7}$$

其中 score 函数可以使用向量夹角的余弦值计算，如式(2.8)：

$$r_{u,i}=\cos\left(w_u,w_i\right)=\frac{w_u*w_i}{w_u*w_i}. \tag{2.8}$$

基于内容的推荐算法过程如图 2.3 所示，以歌曲为例，歌曲 a 和歌曲 c 都是许嵩的古风歌，而歌曲 b 是林俊杰的关于爱情的歌，通过基于内容的推荐计算歌曲之间的相似度，发现歌曲 a 和歌曲 c 相似度极高，而用户 A 喜欢听歌曲 a，这样用户也大概率地会喜欢歌曲 c，所以将歌曲 c 推荐给用户 A。

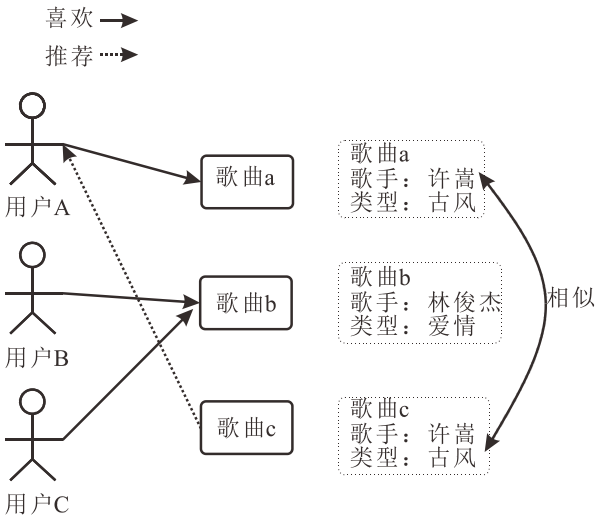

图 2.3 基于内容的推荐算法过程图

Figure 2.3 Process diagram of content-based recommendation algorithm

以上例子就是通过某种标签信息，计算项目之间的相似度然后将相似的项目推荐给该用户。基于内容的推荐算法和基于项目的协同过滤算法一个明显的区别就是后者是根据用户对项目评分来计算相似度而前者通过项目的固有属性，或者标签信息计算项目间的相似度。

2.3 隐语义模型

LFM 项目的隐藏权重进行分类，对于某个用户，通过用户的历史行为数据，首先得到他的所有潜在因子，并同时根据用户对每一类的潜在因子权重来确定用户感兴趣的项目。把一个问题分解成了两个子问题。基于潜在因子分类的方法需要解决三个问题：如何对项目进行分类并且确定用户对哪些类别的项目感兴趣，如何确定这些项目在一个类中的权重。选择哪些属于这个类的项目推荐给用户。为了解决这些问题，采用隐语义模型进行建模。潜在因子模型采用基于用户统计行为的自动聚类。

假设 $\mathbf{R}_{u\times i}$ 是用户 user 对物品 item 的评分矩阵，LFM 算法就是找到两个矩阵 $\mathbf{P}_{u\times k}$，$\mathbf{Q}_{k\times i}$ 使得 $\mathbf{P}*\mathbf{Q}\approx\mathbf{R}$。称 **P**，**Q** 分别为 user 和 item 的特征矩阵，其中 ***k*** 就是隐类的特征维度，最后计算出用户对每个物品 item 的预测评分 $\hat{r}_{ui}$，计算公式如下：

$$\hat{r}_{ui}=\sum_{k=1}^{K}\mathbf{P}_{uk}\mathbf{Q}_{ki}\,. \tag{2.9}$$

下面是 LFM 的目标函数，其中 r_{ui} 为用户 user 对物品 item 的真实评分：

$$min:Loss=\sum_{r_{ui}\neq 0}\left(r_{ui}-\hat{r}_{ui}\right)^2. \tag{2.10}$$

为了防止过拟合添加正则化项 $\sum\mathbf{P}_{uk}^2+\sum\mathbf{Q}_{ki}^2$，λ 为正则项系数，如式 2.11 所示：

$$min:\text{Loss}=\sum_{r_{ui}\neq 0}\left(r_{ui}-\hat{r}_{ui}\right)^2+\lambda\left(\sum\mathbf{P}_{uk}^2+\sum\mathbf{Q}_{ki}^2\right)=f\left(\mathbf{P},\mathbf{Q}\right). \tag{2.11}$$

求解上式最优化问题可以使用交替最小二乘法[56]（Alternating Least Square，ALS）和随机梯度下降法[57]（Stochastic Gradient Descent，SGD）。

（1）交替最小二乘法

ALS 算法思想：固定一个矩阵某一行或者某一列来求解另一个矩阵的某一行或者某一列，然后对另一个矩阵进行交替执行。

首先固定 **Q** 矩阵，对 P_{uk} 求偏导得：

$$\frac{\partial Loss}{\partial\mathbf{P}_{uk}}=-2\sum_{i}^{n}\mathbf{Q}_{ki}\left(r_{ui}-\hat{r}_{ui}\right)+2\lambda\mathbf{P}_u\,. \tag{2.12}$$

其中上式中 $\hat{r}_{ui}=\mathbf{P}_u^T\mathbf{Q}_i$：

$$\frac{\partial Loss}{\partial\mathbf{P}_{uk}}=-2\sum_{i}^{n}\mathbf{Q}_k\left(r_{ui}-\mathbf{P}_u^T\mathbf{Q}_i\right)+2\lambda\mathbf{P}_u,$$

$$=-2\mathbf{Q}r_{ui}+2\sum_{i}^{n}\mathbf{Q}_i\left(\mathbf{P}_u^T\mathbf{Q}_i\right)+2\lambda\mathbf{P}_u,$$

$$= -2\mathbf{Q}r_{ui} + 2\sum_{i}^{n}\mathbf{Q}_i\left(\mathbf{Q}_i^T\mathbf{P}_u\right) + 2\lambda\mathbf{P}_u,$$

$$= -2\mathbf{Q}r_{ui} + 2\lambda\mathbf{P}_u + 2\left(\sum_{i}^{n}\mathbf{Q}_i\mathbf{Q}_i^T\right)\mathbf{P}_u,$$

$$= -2\mathbf{Q}r_{ui} + 2\lambda\mathbf{P}_u + 2\left(\mathbf{Q}_0\mathbf{Q}_0^T + \mathbf{Q}_1\mathbf{Q}_1^T + \cdots + \mathbf{Q}_n\mathbf{Q}_n^T\right)\mathbf{P}_u,$$

$$= -2\mathbf{Q}r_{ui} + 2\lambda\mathbf{P}_u + 2\left[\mathbf{Q}_0,\mathbf{Q}_1,...,\mathbf{Q}_n\right]\begin{bmatrix}\mathbf{Q}_0^T\\ \mathbf{Q}_1^T\\ \cdots\\ \mathbf{Q}_n^T\end{bmatrix}\mathbf{P}_u,$$

$$= -2\mathbf{Q}r_{ui} + 2\lambda\mathbf{P}_u + 2\mathbf{Q}_{k\times n}\mathbf{Q}_{k\times n}^T\mathbf{P}_u = \mathbf{0}. \tag{2.13}$$

将上式(2.13)化简得：

$$\mathbf{P}_u = \left(\mathbf{Q}_{k\times n}\cdot\mathbf{Q}_{k\times n}^T + \lambda\mathbf{E}\right)^{-1}\cdot\mathbf{Q}_{k\times n}\cdot r_{ui}. \tag{2.14}$$

利用式(2.14)将 **P** 一行一行更新，更新完毕之后在用同样的方式更新 **Q**，最后解出 **P**，**Q**。

（2）随机梯度下降法

通过式(2.14)可知更新某一行或者某一列时需要计算矩阵的逆矩阵，然而计算逆矩阵的代价是很大的，所以推出了 SGD 算法求解式(2.11)最优化问题。该算法思路如下：

每次迭代单独更新参数 $\mathbf{P}_{uk}^{(t+1)} = \mathbf{P}_{uk}^{(t)} - \alpha\nabla\mathbf{P}_{uk}$ 和 $\mathbf{Q}_{ki}^{(t+1)} = \mathbf{Q}_{ki}^{(t)} - \alpha\nabla\mathbf{Q}_{ki}$ 其中 α 为学习率：

$$\nabla\mathbf{P}_{uk} = -2\left(r_{ui} - \hat{r}_{ui}\right)\mathbf{Q}_{ki}^{(t)} + 2\lambda\mathbf{P}_{uk}^{(t)}, \tag{2.15}$$

$$\nabla\mathbf{Q}_{ki} = -2\left(r_{ui} - \hat{r}_{ui}\right)\mathbf{P}_{uk}^{(t)} + 2\lambda\mathbf{Q}_{ki}^{(t)}. \tag{2.16}$$

式（2.15）（2.16）带入（2.11）得：

$$\mathbf{P}_{uk}^{(t+1)} = \mathbf{P}_{uk}^{(t)} + \alpha\left[\left(r_{ui} - \hat{r}_{ui}\right)\mathbf{Q}_{ki}^{(t)} - \lambda\mathbf{P}_{uk}^{(t)}\right], \tag{2.17}$$

同理：

$$\mathbf{Q}_{ki}^{(t+1)} = \mathbf{Q}_{ki}^{(t)} + \alpha\left[\left(r_{ui} - \hat{r}_{ui}\right)\mathbf{P}_{uk}^{(t)} - \lambda\mathbf{Q}_{ki}^{(t)}\right]. \tag{2.18}$$

通常的，学习率 α 影响着学习的快慢，若 α 过小则训练模型的时间将会很漫长，α 过大则会导致模型很难收敛，所以 α 的选择决定了训练的快慢。另一个影响的因素是 **P**，**Q** 初始值的选择，一般的 **P**，**Q** 初始为全 0 或 1 的矩阵和随机数矩阵。大量实验证明，**P**，**Q** 取随机值时模型推荐效果更佳。

2.4 混合推荐算法

前面所介绍的几种推荐算法在一定程度上实现了个性化的推荐。但由于这些算法都有存在缺点并在一定程度上受限于特定的应用场景；像协同过滤算法存在着冷启动和数据稀疏性的问题，当系统刚启动时，用户的历史数据与评分稀疏会导致推荐效果不佳；然而对于基于内容的推荐，要获取项目的本质或者内容属性，当遇到在音乐，电影等属性不好分析提取的项目上进行推荐时会比较困难；基于标签的推荐算法也会出现标签的多义性和同义分析问题等难题。同时，推荐系统中存在一些数据信息能够被一类推荐算法或多类算法所应用。一个推荐系统有可能不仅能获取到用户历史行为数据，还可以容易提取项目的内容属性。这样获取到大量信息的行为随着进入大数据时代是一个可行的方式。从上面两个方面考虑，混合推荐算法开始受到关注，混合推荐能够结合不同算法的优势，在一定程度上弥补不同算法的劣势。当前 Netflix 推荐算法大赛都是利用不同算法的优点进行混合推荐，才能喜提金牌。目前混合推荐尝试用的策略是协同过滤和基于内容的推荐算法进行混合，主要有如下两种方法[58]：

（1）协同过滤算法与基于内容的推荐算法思想互相融入。

（2）同时执行协同过滤和基于内容的推荐算法，对计算出的推荐列表进行加权整合。对于基于内容的推荐、基于协同过滤的推荐、基于标签的推荐算法等相互引荐，总结出更优地推荐算法。其中有以下不同混合方式：切换、加权、交叉混合、特征混合、串联层叠、特征补充、元级。

切换：使用多种算法在面对不同情况时进行相互切换。算法思想是充分利用各种算法的优点，将其应用在不同的情境之中。比如推荐系统上线初期，用户行为及项目数据集较稀疏，由于基于协同过滤的算法需要大量行为数据，所以会产生冷启动问题推荐准确率较低，然而基于内容的推荐算法只需考虑用户及项目属性，不会产生数据稀疏性问题，这时考虑使用基于内容的推荐算法。待系统运行一段时期后，用户行为数据集不再稀疏，这是考虑切换为基于协同过滤的推荐算法。切换行为面对的问题在于选择一个合适的指标来确认切换算法的时机。

加权：使用线性公式（linear formula）将几种不同的推荐算法按照一定权重组合起来。在测试集上使用不同的权重组合进行测试，选出使得推荐系统有最佳推荐效果的权重组合。加权的优点是对不同推荐算法的推荐能力进行线性组合。但是组合的推荐算法的擅长领域有冲突，可能会造成推荐效果不佳。例如，协同过滤算法在数据集稀疏的情况下推荐效果较差，此时若是采用基于内容的推荐能在此数据集上有着良好的性能表现，但线性组合起来反而造成推荐效果下降。

特征混合：从多个数据集中提取不同特征提供给某一个算法进行训练。与加权，

切换两种方式不同的是，采用特征组合的推荐系统中只存在一个算法模块。

交叉组合：这种方式直接将多种推荐算法计算得到的推荐结果合并为一个列表给推荐给用户。这种想思想非常简单，但是存在推荐列表喜好排序问题。

串联层叠：该方式一般使用两类推荐算法，包含一类主推荐算法和一类副推荐算法，主推荐算法得出推荐结果，而副推荐算法对推荐结果进行优化。

特征补充：从一个数据集中提取项目特征，将提取出来的特征提供给另一种推荐算法作为算法的输入。其优势是可以离线处理数据。

元级：对用户与项目之间的历史行为数据进行分析建模，将建立的模型提供给下一种算法作为算法的输入。这种方式和特征补充之间的区别在于给予下一种算法的输入的数据，元级将前一个算法建立的模型作为输入，而特征补充是将提取出来的特征作为下一种算法的输入。

混合推荐算法一般利用基于内容推荐，基于协同过滤推荐等多种算法选取其优势场景进行混合推荐，期望最大化地发挥各种算法的优势，尽量弥补算法的缺陷。例如，基于矩阵分解的协同过滤算法可以融入基于项目内容的推荐算法，可以使得推荐结果在多个评估指标上得到较好的结果。

2.5 推荐算法的评估标准

上述介绍了各种推荐算法及其优缺点，本节将要论述推荐算法的评估标准，通过评估标准去比较两个推荐算法孰好孰坏。

早期，几乎 99%与推荐算法相关的论文都在研究准确度这一指标。其中准确度经常使用平均绝对误差（Mean Absolute Error，MAE）和均方根误差（Root Mean Squared Error，RMSE）来计算。给定测试集中用户 u 与项目 i，r_{ui} 表示用户 u 对项目 i 的真实评分，$\hat{r}_{ui}$ 表示用户 u 对项目 i 的预测评分，故 MAE 计算公式为：

$$\mathrm{MAE}=\frac{\sum_{u,i\in T}\left|r_{ui}-\hat{r}_{ui}\right|}{\left|T\right|}, \tag{2.19}$$

同理 RMSE 使用均方根误差，其计算公式为：

$$\mathrm{RMSE}=\frac{\sqrt{\sum_{u,i\in T}\left(r_{ui}-\hat{r}_{ui}\right)^2}}{\left|T\right|}. \tag{2.20}$$

推荐系统为用户服务时往往是给用户一些个性化推荐列表，这就是 $top-N$ 推荐，而评估这种推荐的好坏是使用准确率（Precision），召回率（Recall）。下面给出 Precision，Recall 的计算公式：

$$\mathrm{Precision}=\frac{TP}{TP+FP}, \tag{2.21}$$

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{2.22}$$

其中上式的 *TP*（True Positive）表示预测为正，实际为正，即推荐的是用户喜欢的；*FP*（False Positive）表示预测为正，实际为负，即推荐的不是用户喜欢的；*FN*（False Negative）表示预测为负，实际为正，即用户喜欢的没有被推荐。通过定义可知，Precision 实际上是推荐命中占总推荐数的比重，其会随着推荐个数 N 值增大而减小；而 Recall 实际上指推荐命中占总样本得比重，其会随着推荐个数 N 值增大而增大。故而通过 Precision 和 Recall 作为评测指标实为不妥。因此使用 F1 值作为评测指标居多，其计算公式如下：

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{2.23}$$

但是对于一个好的推荐系统来说，需要做到的是“双赢”，即商家为用户做出推荐的时候也要使自己的利益最大化，推荐系统也要为用户带来便捷。对于电商来说，如果推荐系统推荐的商品该用户之前已经购买过了，这时 F1 值很高，但是这并不能让商家多卖一件这个商品，也不能让该用户感觉到多么便捷。这种推荐是很失败的。所以一个好的推荐系统应该为用户推荐那些看起来不流行实则用户很喜欢的商品。

从上述分析可知，覆盖率（Coverage）对一个推荐系统来说也很重要。其既可以提高商家得利益，也可以提高用户的体验。

2.6 本章小结

本章第一节介绍了基于协同过滤的推荐算法，并且将基于用户的协同过滤和基于项目的协同过滤进行对比，在基于模型的协同过滤中引出了本文的关键技术：隐语义模型，并且在第三节进行详细介绍。第二节说明了基于内容的推荐算法，将基于内容的推荐算法和基于协同过滤的推荐算法进行优缺点分析，在第四节提出了混合推荐算法。在第五节对推荐算法提出了一些评测指标。

第 3 章 基于用户聚类隐语义模型的优化

第二章中阐述了基于内容的推荐算法，该算法更多关注的是内容的属性和特征，对于文字类信息很容易提取其内容，但是像音乐、电影等媒体就很难提取其内容或者特征。而基于协同过滤的推荐算法更多的关注了用户的行为。基于模型的协同过滤中矩阵分解技术是隐语义模型的关键。

本章首先介绍矩阵分解相关技术，比如奇异值分解（Singular Value Decomposition，SVD），然后通过分析其不足进而提出了隐语义模型（LFM），基于用户兴趣提出了带偏置的 LFM（B-LFM）算法，采用具有相同兴趣的热门项目作为训练的负样本，提出了用户聚类的 LFM 算法（UC-LFM）。通过大量实验证明了，B-LFM 算法的 MAE、F1 值、Coverage 均优于 LFM 算法。并且 UC-LFM 算法也优于 B-LFM 算法。

3.1 矩阵分解

基于协同过滤的推荐算法都存在着数据稀疏性这一问题，对于一个评分矩阵来说很多的值都是空缺的。为了解决该问题，一般使用空值填充和矩阵分解技术。使用评分的平均值，众数等来填充空缺的值，但是每个人偏好是不同的，这样并不能保证推荐的准确性。另一方面，通过空值填充技术填充的矩阵将会是一个稠密矩阵，在进行矩阵计算时将会非常耗时。然而通过矩阵分解技术对原有矩阵进行降维，仅仅提取重要的信息，将一个高维低秩矩阵分解成两个低维矩阵进行计算。不但会提高计算效率，而且推荐准确度也会大幅增长。

3.1.1 奇异值分解

评分矩阵是一个稀疏矩阵，对空值填充的精确程度将直接影响到推荐的准确率。本节的开始已经说明了空值填充的几种常见方式，也就意味着填充后的矩阵要近似等于原有矩阵。矩阵的特征值决定着矩阵的相似程度，对于矩阵奇异值分解恰巧可以满足。这样可以使空值填充的低准确度问题得以解决。

在使用 SVD 分解时，是不允许矩阵存在空值的，所以将空值使用评分的均值加以填充。给定 $\mathbf{R}$ 为用户的评分矩阵，其维度为 $m\times n$，对其进行空值填充后矩阵为 $\mathbf{R}^+$，对 $\mathbf{R}^+$ 进行 SVD 分解：

$$\mathbf{R}^+ = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T . \tag{3.1}$$

上式中，$\mathbf{U}$ 表示 $m\times m$ 维的正定矩阵（酉矩阵），$\mathbf{\Sigma}$ 表示 $m\times n$ 维对角矩阵，其值

为矩阵的奇异值，**V** 表示 $n\times n$ 维的酉矩阵。对矩阵 $\mathbf{R}^+$ 进行 SVD 分解的方法示意图如图 3.1 所示：

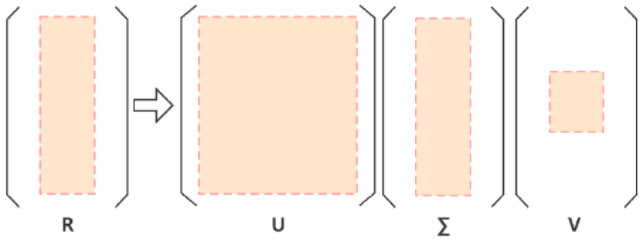

图 3.1 奇异值分解示意图

Figure 3.1 Singular value decomposition diagram

上面基于 SVD 的分解对评分矩阵原数据保留的基础上，分解出了秩为 $\boldsymbol{r}$ 的最佳相似矩阵，$\boldsymbol{\Sigma}$ 是奇异值从大到小排列的所以只有前 $\boldsymbol{k}$ 大的奇异值决定着矩阵的整体属性，而 $r-k$ 个奇异值或是影响较小，或是噪声可忽略不计，所以取前 $\boldsymbol{k}$ 个特征值得到低维的子矩阵。

一般地，$\boldsymbol{k}$ 的取值要远远小于 m、n，首先从原有的 $\boldsymbol{\Sigma}$ 矩阵中选取前 $\boldsymbol{k}$ 大的特征值组成另一个对角矩阵 $\boldsymbol{\Sigma}_k$，然后在酉矩阵 **U**、**V** 中取出 $\boldsymbol{k}$ 个奇异值对应的特征向量组成新的酉矩阵 $\mathbf{U}_k$、$\mathbf{V}_k$，最后根据构建出的 $\mathbf{U}_k$、$\boldsymbol{\Sigma}_k$、$\mathbf{V}_k$，利用公式(3.2)计算出预测评分矩阵 $\mathbf{R}_k$，使该矩阵无限相似于 $\mathbf{R}^+$：

$$\mathbf{R}^+ \approx \mathbf{R}_k = \mathbf{U}_k\boldsymbol{\Sigma}_k\mathbf{V}_k^T . \tag{3.2}$$

预测评分矩阵 SVD 分解示意图如图 3.2 所示：

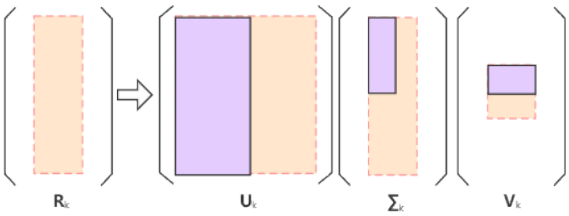

图 3.2 近似矩阵奇异值分解示意图

Figure 3.2 Singular value decomposition of approximate matrix

上图中，深色部分属于矩阵分解后保留的部分，其他是原数据部分可以舍弃。基于 SVD 矩阵分解的推荐算法步骤如下：

算法：SVD 矩阵分解推荐算法

输入：评分矩阵 **R**

输出：用户属性矩阵 **U**、项目属性矩阵 **V**

步骤：

（1）将评分矩阵 **R** 空值填充为 $\mathbf{R}^+$.

（2）求 $\mathbf{R}^{+}\mathbf{R}^{+T}$ 记为 $\mathbf{U}^{+}$; $\mathbf{R}^{+T}\mathbf{R}^{+}$ 记为 $\mathbf{V}^{+}$.

（3）分别对 $\mathbf{U}^{+}$， $\mathbf{V}^{+}$ 求其特征值 λ_i 及其对应的特征向量 ξ_i , ζ_i .

$$\mathbf{U}=\left(\xi_1,\ldots,\xi_r\right),\mathbf{V}=\left(\zeta_1,\ldots,\zeta_r\right).$$

（4）求出奇异值 $\sigma_i=\max\left\{sqrt\left(\lambda_i\right)\right\}$.

$$\mathbf{\Sigma}=diag\left(\sigma_1,\ldots,\sigma_r\right).$$

（5）选取前 ***k*** 个奇异值及其特征向量对应矩阵

$$\mathbf{U}_k=\left(\xi_1,\ldots,\xi_k\right),\mathbf{\Sigma}_k=diag\left(\sigma_1,\ldots,\sigma_k\right),\mathbf{V}_k=\left(\zeta_1,\ldots,\zeta_k\right).$$

（6）使用公式 3.2 计算 R_k，并根据评分大小排序进行推荐

3.1.2 隐语义模型

基于 SVD 的矩阵分解在面对用户少、项目少的系统时，能够很好的解决用户稀疏性问题。但是随着系统中用户的不断增长以及项目的不断丰富，使用 SVD 进行矩阵分解效率很差。第一，对用一个用户量大的稀疏矩阵，对其进行空值填充不能很好的表现出用户的真实行为；第二，SVD 矩阵分解本身效率就很差，在面对维度不高的矩阵影响较小，一旦矩阵维度 1000 进行矩阵分解非常耗时。

基于以上分析，SimonFunk 于 2006 年提出了一种基于 Funk SVD 的矩阵分解算法，该算法实际上是通过 SVD 分解进化而来。并且在一次 Netflix Prize 推荐算法大赛中拔得头筹，后来将该算法称之为隐语义模型（Latent Factor Model，LFM）。

LFM 算法在前面已经提到过，该算法利用降维的思想将用户评分矩阵进行补全。不同于 SVD 算法的是，该算法不需要通过奇异值将评分矩阵分解，而是直接分解为两个矩阵，分别是用户的潜在因子矩阵和项目的潜在因子矩阵。这种潜在因子是一种抽象的属性，比如在音乐推荐系统中《断桥残雪》和《红昭愿》看似毫无联系，但是它们都有一个潜在的抽象的属性**:** 古风。所以当用户喜欢《断桥残雪》时，使用 LFM 算法计算用户对《红昭愿》的评分也会很高。LFM 算法将评分矩阵 **R** 分解为两个低维潜在因子矩阵 **P**、**Q**，分解示意图如下：

	item1	item2	item3	item4
user1	r11	r12	r13	r14
user2	r21	r22	r23	r24
user3	r31	r32	r33	r34

R

	factor1	factor2	factor3
user1	p11	p12	p13
user2	p21	p22	p23
user3	p31	p32	p33

P

×

	item1	item2	item3	item4
factor1	q11	q21	q31	q41
factor2	q12	q22	q32	q42
factor3	q13	q23	q33	q43

Q^T

图 3.3 LFM 分解示意图

Figure 3.3 LFM decomposition diagram

基于 LFM 矩阵分解得：

$$\hat{\mathbf{R}} = \mathbf{P}\mathbf{Q}^T . \tag{3.3}$$

给定评分矩阵 **R**，其维度为 $m \times n$，则分解后矩阵 **P** 维度为 $m \times k$，矩阵 **Q** 维度 $n \times k$，其中 $\boldsymbol{k}$ 为潜在因子维度。利用公式(3.4)可以计算出用户 u 对项目 i 的评分：

$$perference = \hat{r}_{ui} = \sum_{k=1}^{K} \mathbf{P}\mathbf{Q}^T . \tag{3.4}$$

LFM 得目标函数为：

$$min : \text{Loss} = \sum_{r_{ui} \neq 0} \left(r_{ui} - \hat{r}_{ui} \right)^2 + \lambda \left(\sum \mathbf{P}_{uk}^2 + \sum \mathbf{Q}_{ki}^2 \right) = f\left(\mathbf{P}, \mathbf{Q} \right) . \tag{3.5}$$

其中 $\lambda \left(\sum \mathbf{P}_{uk}^2 + \sum \mathbf{Q}_{ki}^2 \right)$ 是为了防止过拟合加入的正则化项，λ 为正则化系数。求解该函数最优化问题已经在第二章中详细说明。

3.2 聚类算法相关知识

在机器学习领域有两种学习方式，分别是监督学习（Supervised Learning）和无监督学习（Unsupervised Learning）。监督学习中的样本包含特征和标签，在学习的过程中，系统可以判断预测的正确与否。这种学习方式最典型的代表是分类和回归；非监督学习中样本只包含特征而没有标签，系统将特征相同的通过学习聚合在一起，这种学习方式最典型的是聚类。

3.2.1 聚类概念

聚类就是把一些相似的食物通过某种规则聚合在一起，其在数据分析领域具有重要作用。在进行数据挖掘的时候，首先会将这些数据通过某种方式进行分组，然后对各个组类数据进行聚合分析。“物以类聚，人以群分”就是聚类的典型思想，下面通过一些歌曲的案例详细介绍聚类的概念。

对于以下歌曲：《断桥残雪》、《醉赤壁》、《半城烟沙》、《Something Just Like This》、《Play With Style》。将这些歌曲以不同的规则聚类得到的结果也是完全不同的。如果以歌名语种进行聚类，《断桥残雪》、《醉赤壁》、《半城烟沙》是中文歌名将其聚为一类，而《Something Just Like This》、《Play With Style》是英文歌名会聚在一起。如果以歌手进行聚类，《断桥残雪》、《半城烟沙》、《Play With Style》都是许嵩的歌曲会将其聚为一类，《醉赤壁》是林俊杰歌曲将其聚为一类，《Something Just Like This》是 The Chainsmokers 的歌曲，将此单独聚为一类。如果按照歌曲风格聚类，《醉赤壁》、《半城烟沙》、《断桥残雪》都是古风歌曲将其聚为一类，《Something Just Like This》、《Play

With Style》聚为一类。

从以上的例子可以看出，通过不同的规则聚类出来的结果也是完全不同的。聚类分析也被用于很多场合，上面说的数据挖掘领域，图像分割技术都应用了聚类分析。

设 X 是训练集，对于 $x_1,x_2,\ldots,x_N$，有：

$$\mathrm{X}=\{x_1,x_2,\ldots,x_N\},$$

定义 X 的 ***n*** 聚类，将 X 分成 ***n*** 个聚类，使其满足 $\{C_1,C_2\ldots C_n\}$，其约束如下：

$$\begin{cases} C_i\neq\varnothing,i=1,\ldots,n \\ \bigcup_{i=1}^{n}C_i=\mathrm{X} \\ C_i\cap C_j=\varnothing,i\neq j,i=1,\ldots,n,j=1,\ldots,n. \end{cases} \tag{3.6}$$

对于每一个聚类而言，都是真实存在的，而且所有聚类中的所有项目构成的集合等于 X，相同聚类的数据行为是极其相似的，不同聚类的数据行为是不同的。

传统的聚类分析方法主要有四种：基于划分的聚类[59]、基于层次的聚类[60]、基于密度的聚类、基于网格的聚类。

（1）基于划分的聚类

给定一个有 *N* 个元组的数据集，将其划分为 ***n*** 个分组，其中每个分组代表一个聚类，并且这 ***n*** 个分组必须满足两个条件：第一，每一个分组至少含有一项数据纪录；第二，每一项数据属于且仅属于某一个分组。对于每个分组 $\{C_1,C_2\ldots C_n\}$，算法会给出每个初始状态，之后通过反复迭代改变分组状态，使得同一分组中的数据越来越相似，而不同分组中的数据不同。使用这个基本思想的算法有：*k*-means 算法、CLARANS 算法等；

（2）基于层次的聚类

基于层次的聚类将数据集进行层次性分解，直到满足某种条件为止。分为“自底向上”和“自顶向下”两种分解方式。“自底向上”的层次分解又被称为凝聚（agglomerative），首先将每个对象定义为一个簇，然后不断扩大这个簇，直到满足某个终结条件或者这个簇不能再扩大为止；“自顶向下”的层次分解被称为分裂（divisive），首先将所有的对象都放到一个簇中，然后逐渐划分成越来越小的簇，直到满足某个终结条件或者每个对象成为一个簇为止。基于层次的聚类代表算法有：BIRCH 算法、AGNES 算法、DIANA 算法等；

（3）基于密度的聚类

基于密度的聚类和其它聚类不同的是：它是基于密度的。所以该方法不会出现基于距离的算法带来的 “类圆形”的聚类的缺点。基于密度的聚类思想：定义一个阈值，一旦某个区域的点密度大于这个阈值，将其加入到最近的聚类中。代表算法有：

DBSCAN 算法、OPTICS 算法、DENCLUE 算法等

（4）基于网格的聚类

基于网格的聚类中所有的处理都是以单元为基本单位，这样做的好处是处理速度很快。只和划分单元的个数有关。其中该类型的代表算法有：STING 算法、WAVE-CLUSTER 算法、CLIQUE 算法；

聚类算法中有一个常见的定义就是相似度计算。在聚类过程中，目的是将同类数据划分到一起，将相似度小的数据分开，聚类算法计算不同数据对象之间的相似度，并且以此作为依据划分成不同的簇。相似性度量通常采用距离和相似系数。距离越小，相似系数越高，两个样本间的相似度越高。

其中，计算距离有以下三个性质：

① 非负性：$\mathrm{d}(x,y)>0$；

② 对称性：$\mathrm{d}(x,y)=\mathrm{d}(y,x)$；

③ 不等性：$\mathrm{d}(x,y)\le \mathrm{d}(x,c)+\mathrm{d}(c,y)$；

常见的相似度计算公式主要有五种：欧氏距离、曼哈顿距离、切比雪夫距离、余弦相似度、皮尔森(Pearson)相关系数等。

（1）欧氏距离

欧氏距离是二维甚至多维坐标系中常见的距离计算方式，其描述的是两个点之间的直接距离。假设多维空间中有向量$\mathbf{a}\left(x_{i1},x_{i2},\ldots,x_{in}\right)$和向量$\mathbf{b}\left(x_{j1},x_{j2},\ldots,x_{jn}\right)$，则向量 **a** 和向量 **b** 的欧氏距离定义为：

$$\mathrm{d}(a,b)=\sqrt{\sum_{k=1}^{n}\left(x_{ik}-x_{jk}\right)^2}. \tag{3.7}$$

（2）曼哈顿距离

曼哈顿距离与欧氏距离类似，不过其计算的是各个坐标轴投影距离的和。比如计算直线 AB 的距离，对于欧氏距离，仅仅只是计算 AB 的真实距离，而对于曼哈顿距离来说，计算的是以 AB 为斜边的直角三角形中两个直角边的距离。

假设多维空间中有向量$\mathbf{a}\left(x_{i1},x_{i2},\ldots,x_{in}\right)$和向量$\mathbf{b}\left(x_{j1},x_{j2},\ldots,x_{jn}\right)$，则向量 **a** 和向量 **b** 的曼哈顿距离定义为：

$$\mathrm{d}(a,b)=\sum_{k=1}^{n}\left|x_{ik}-x_{jk}\right|. \tag{3.8}$$

（3）切比雪夫距离

切比雪夫距离是向量空间中的一种度量，从数学的观点来说，它是由一致范数衍生出的量，描述的是各坐标差绝对值的最大值。

假设多维空间中有向量$\mathbf{a}\left(x_{i1},x_{i2},\ldots,x_{in}\right)$和向量$\mathbf{b}\left(x_{j1},x_{j2},\ldots,x_{jn}\right)$，则向量 **a** 和向量 **b**

的切比雪夫距离定义为：

$$\mathrm{d}(a,b)=\sum_{k=1}^{\mathrm{n}} max\left|x_{ik}-x_{jk}\right|. \tag{3.9}$$

（4）余弦相似度

余弦相似度指的是两个向量夹角的余弦值。假设多维空间中有向量 $\mathbf{a}\left(x_{i1},x_{i2},\ldots,x_{in}\right)$ 和向量 $\mathbf{b}\left(x_{j1},x_{j2},\ldots,x_{jn}\right)$，则向量 **a** 和向量 **b** 的余弦相似度定义为：

$$\cos\langle a,b\rangle=\frac{\sum_{k=1}^{n}x_{1k}x_{2k}}{\sqrt{\sum_{k=1}^{n}x_{1k}^{2}}\sqrt{\sum_{k=1}^{n}x_{2k}^{2}}}. \tag{3.10}$$

（5）皮尔森(Pearson)相关系数

Pearson 相关系数用于描述两个变量 X，Y 之间的线性相关度，其定义为两个变量的协方差与标准差乘积的比值，如公式(3.11)所示：

$$\mathrm{R}=\frac{\mathrm{cov}(X,Y)}{\sigma_X\sigma_Y}=\frac{\sum_{i=1}^{\mathrm{n}}\left(X_{\mathrm{i}}-\bar{X}\right)\left(Y_{\mathrm{i}}-\bar{Y}\right)}{\sqrt{\sum_{\mathrm{i=1}}^{\mathrm{n}}\left(X_{\mathrm{i}}-\bar{X}^{2}\right)}\sqrt{\sum_{\mathrm{i=1}}^{\mathrm{n}}\left(Y_{\mathrm{i}}-\bar{Y}^{2}\right)}}. \tag{3.11}$$

3.2.2 *k*-means 聚类算法

k-means 聚类算法又称 *k* 均值聚类算法，该算法是基于距离的聚类算法，一般认为两个点之间的距离越小，这两个点越相似。*k*-means 算法实现简单，收敛速度快，在众多聚类算法中被广泛地使用。下面介绍 *k*-means 算法的具体流程。

定义一个聚类 R，表示数据集 X 的 ***n*** 聚类，L 为该聚类的损失函数。给定的输入 X，*k*-means 算法任务是找到聚类 $C=\left\{C_1,C_2\ldots C_n\right\}$，使得损失函数 L(*X*,*C*)最小，有上面约束可知，最终训练集将被分到不同的簇中，且 $C_i\cap C_j=\varnothing$，对于每个簇 C_i 都有一个中心点 μ_i，若 X 被嵌入到更广的空间 X′，即 $\mathrm{X}\subseteq\mathrm{X}'$，$\mu\in\mathrm{X}'$，L 测量 X 中各个点到各个簇中心点距离的平方，即 $\mathrm{d}(x,\mu)^2$。C_i 中心点定义为：

$$\mu_i\left(C_i\right)=\arg\min_{\mu\in\mathrm{X}}\sum_{x\in C_i}\mathrm{d}(x,\mu)^2. \tag{3.12}$$

所以 *k*-means 算法的目标就是：

$$\mathrm{L}_{k-\mathrm{means}}\left(X,C\right)=\sum_{i=1}^{n}\sum_{x\in C_i}\mathrm{d}\left(x,\mu_i\left(C_i\right)\right)^2. \tag{3.13}$$

该算法的流程图如图 3.4 所示：

（1）从数据集 X 中选取 ***n*** 个中心点 $\mu_1,\ldots,\mu_n$ 作为初始的聚类中心；

（2）使用公式(3.7)计算数据集中每个点 X_i 到各个中心点 μ_j 的距离，记为 $\|\mathrm{X}_i\|_j$，对每个 X_i 选出距离最短的 $\|\mathrm{X}_i\|_j$，记 X_i 的类别为 $\tau(\mathrm{X}_i)$；

（3）对每个聚类中的点均值化 $\kappa_1,\ldots,\kappa_n$，更新 $\mu_1,\ldots,\mu_n$ 为 $\kappa_1,\ldots,\kappa_n$；

（4）重复(2)(3)直到聚类中心点不再发生变化，目标收敛，算法结束。

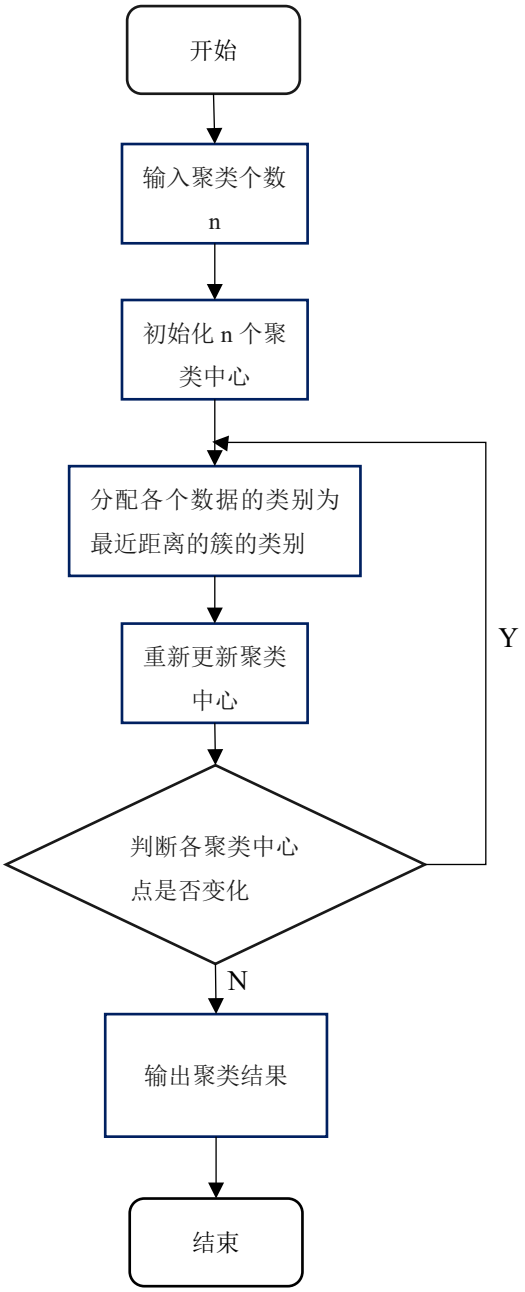

图 3.4 *k*-means 算法流程图

Figure 3.4 K-means algorithm flow chart

k-means 算法迭代过程如图 3.5 所示，其中图(a)表示输入的原始数据，图(b)初始化六个质心，意味着将数据分为六个簇。将每个点分别与各个簇计算距离，将该数据点划分到距离最近的质心中。然后计算每个簇的样本中心，将质心更新为该样本中心，循环以上过程直到簇中数据不再变化。如图 3.5 中的图(c)是第二次迭代的结果，这时有些点已经分类完毕，图(d)中仅仅是某些比较近的点存在着误分类，而图(e)和图(f)所

有点均分类完毕，并且最终样本中心不再变化。

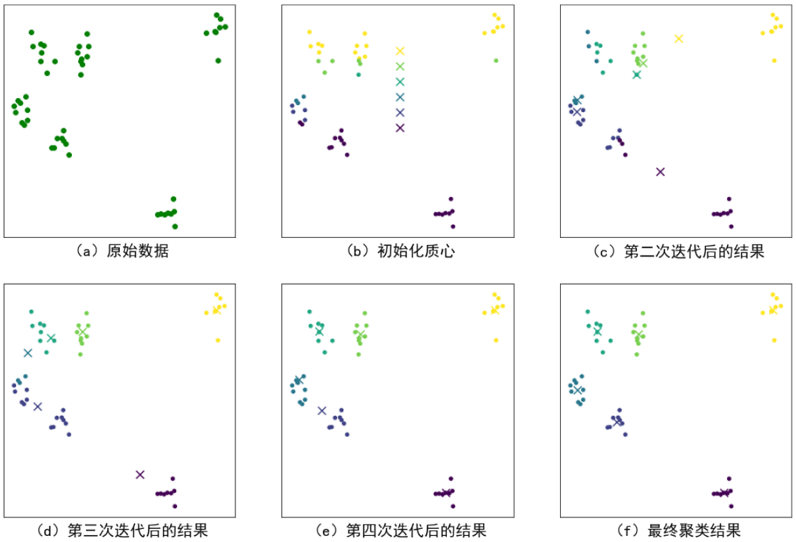

图 3.5 *k*-means 算法迭代过程图

Figure 3.5 K-means algorithm iteration process diagram

3.3 基于用户聚类隐语义模型的优化

上面的 LFM 算法没有考虑到用户的偏好和项目的热门程度。假设用户 A 偏向于打分很低，即使自己很喜欢的物品也只会给个中等分数，而用户 B 偏向于打分很高，每个物品都给了较高的分数，LFM 对于这种用户的推荐的准确率就会降低。同样的，假定物品 X 属于热门产品，大多数用户都对该物品有过行为，而物品 Y 属于冷门产品，为极少数用户所知。可能该物品某个用户非常喜欢，但是没有为该用户进行推荐，导致覆盖率很低。针对以上问题，下面提出了带偏置的 LFM 算法。大量实验对比了 LFM 是 *Top-N* 中推荐的性能得出影响 LFM 算法性能的重要参数是：隐类的特征维度 $\boldsymbol{k}$、学习率 α、正则化系数 λ、正负样本比例。其中正负样本选取对算法性能影响最大，据此在 LFM 中引入了用户聚类算法。

3.3.1 带偏置的 LFM 算法

之前提到的 LFM 算法预测公式为：

$$perference = \hat{r}_{ui} = \sum_{k=1}^{K} \mathbf{P}\mathbf{Q}^{T} . \tag{3.14}$$

这个公式通过隐特征将用户和项目联系到了一起。实际上，有些评分数据的固有属性与用户和项目都无关，同样对于用户一些属性也与项目无关，项目一些属性也与用户无关。据此提出一种带偏置的 LFM 算法，本文将其定义为 B-LFM 算法。其预测

公式如下：

$$\hat{r}_{ui} = \sum_{k=1}^{K} \mathbf{P}\mathbf{Q}^{T} + \mu + \mathbf{b}_u + \mathbf{b}_i . \tag{3.15}$$

式中 μ 代表系统的全局平均分，对于不同的系统而言，有些系统评分普遍很低，而有些系统评分很高，μ 表示系统本身对评分的影响；$\mathbf{b}_u$ 表示用户偏置，也就是用户的打分习惯，有些用户打分比较严格，即使很喜欢的项目也不会打很高的分，有些用户打分随缘，即使不喜欢的项目，可能也会打高分，$\mathbf{b}_u$ 表示了用户自身喜好对评分的影响；$\mathbf{b}_i$ 表示项目偏置，也就是项目的热门程度，对于一些电影来说，电影本身就很精彩，所以获得的评分也很高，而对于某些烂片，几乎很少人喜欢，得到的评分自然很低，$\mathbf{b}_i$ 表示项目热门程度对评分的影响。

对公式 3.14 给出目标函数：

$$min: \mathrm{Loss} = \sum_{r_{ui} \neq 0} \left(r_{ui} - \hat{r}_{ui} \right)^2 + \lambda \left(\sum \mathbf{P}_{uk}^2 + \sum \mathbf{Q}_{ki}^2 + \sum \mathbf{b}_u^2 + \sum \mathbf{b}_i^2 \right). \tag{3.16}$$

其中 $\sum \mathbf{P}_{uk}^2 + \sum \mathbf{Q}_{ki}^2 + \sum \mathbf{b}_u^2 + \sum \mathbf{b}_i^2$ 是为了防止过拟合加入的正则化项，λ 表示正则化系数。

求解该目标函数最优化问题与求解 LFM 算法的最优化问题类似，更新：$\mathbf{b}_u$、$\mathbf{b}_i$、$\mathbf{P}$、$\mathbf{Q}$ 如下所示：

更新 $\mathbf{b}_u$：

$$\mathbf{b}_u^{(t+1)} = \mathbf{b}_u^{(t)} + \alpha * \left(r_{ui} - \hat{r}_{ui} - \lambda * \mathbf{b}_u^{(t)} \right), \tag{3.17}$$

更新 $\mathbf{b}_i$：

$$\mathbf{b}_i^{(t+1)} = \mathbf{b}_i^{(t)} + \alpha * \left(r_{ui} - \hat{r}_{ui} - \lambda * \mathbf{b}_i^{(t)} \right), \tag{3.18}$$

更新 P：

$$\mathbf{P}_{uk}^{(t+1)} = \mathbf{P}_{uk}^{(t)} + \alpha \left[\left(r_{ui} - \hat{r}_{ui} \right) \mathbf{Q}_{ki}^{(t)} - \lambda \mathbf{P}_{uk}^{(t)} \right], \tag{3.19}$$

更新 Q：

$$\mathbf{Q}_{ki}^{(t+1)} = \mathbf{Q}_{ki}^{(t)} + \alpha \left[\left(r_{ui} - \hat{r}_{ui} \right) \mathbf{P}_{uk}^{(t)} - \lambda \mathbf{Q}_{ki}^{(t)} \right]. \tag{3.20}$$

反复利用公式(3.17)(3.18)(3.19)(3.20)迭代更新 $\mathbf{b}_u$、$\mathbf{b}_i$、$\mathbf{P}$、$\mathbf{Q}$ 直到收敛。

3.3.2 用户聚类算法在 LFM 算法中的应用

目前绝大多数推荐系统商家都是基于点击通过率（Click-Through-Rate，CTR）来做的，而正负样本的选择则是 CTR 中重要的组成部分。对于不同的业务场景来说，关键考核指标（Key Performance Indicator，KPI）也是不同的，从而导致模型的训练目标

也大相径庭。比如使用点击率当作 KPI，说明只要用户点击了某项目就表示该用户喜欢，此时的训练目标就是推荐的准确性，提升用户的准确率；如果使用交易额当作 KPI，这时系统考虑的是用户有没有可能购买该商品，这时候训练目标是用户的下单率。

然而有些 KPI 指标并不能量化为模型训练，比如用户的活跃度、用户的平均逗留时长等等。这时可以通过分析业务指标，进而量化成模型训练的目标。比如在《王者荣耀》游戏中，很多玩家可能出现“短暂”退游情况，这时候可以给用户推送一些返场福利，该用户很可能就变成了活跃用户。

推荐算法中正负样本的选取一直是学术论文中研究的热点话题之一。对于正样本的选取，不能单单凭借用户产生行为的就是正样本，这里存在着误点击。比如在某新闻系统，总是存在着一些“标题党”，其标题和内容完全不符。但是用户会被标题所吸引，点进去发现并不是自己喜欢的，这就不能作为正样本。对于一些视频推荐网站比如 YouTube 会有“订阅”、“分享”、“顶一下”、“踩一下”，当用户订阅了某个视频，意味着该用户极大可能喜欢这个视频，而对“分享”、“顶一下”行为并不能表示用户喜欢该视频，相反如果用户“踩一下”某个视频，那么该用户必定不会喜欢该视频。

在大多数推荐系统中往往不会只使用一种推荐算法，通常是对各推荐算法交替使用各司其职。据此，选取相同兴趣的用户群体喜欢的热门项目作为负样本。这样能更大程度上保证用户看过该项目，但是没有点击进去，充分说明了用户不喜欢该项目，能够更好的满足负样本的要求。

正负样本不均衡可能导致比例大的样本过拟合，而比例小的样本欠拟合，这样会极大的降低模型的泛化能力。所以要保证正负样本的均衡。

大量实验证明，LFM 算法中正负样本的选取对算法性能影响最大。对于负样本的选择要遵循以下规则：第一，对于每个正负样本比例要适中，并且负样本数目不小于正样本数目；第二，对于每个用户选取热门的并且用户没有过行为的项目作为负样本，用户对一个热门项目的反馈是负面的更能体现推荐系统的准确性；第三，选取相同兴趣的用户喜欢的项目。基于以上原则，下面提出了一种基于用户聚类的负样本选取算法。

给定训练集 X，其中 X 包含每个用户及其有过行为的项目。将其转化为一个评分矩阵 **R** 与用户集合 U。首先初始化 n 个质心 μ_1， μ_2，…， μ_n，然后计算每个用户与该质心的相似度矩阵，对每个相似度矩阵进行从大到小排序，对每个用户选取与自身最相似的簇，作为自身的类别，接着对于每个簇中的用户计算中心点，将质心更新为当前中心，重复迭代直到所有簇的用户不再发生改变为止。其中算法的关键是找到用户之间的相似度，其中相似度可以通过 Pearson、欧几里德距离等去计算，由于欧几里得距离计算相似度很有局限性，通常采用 Pearson 进行计算。用户 i 和 j 的相似度记为

$sim(i,j)$，R_{ik}、R_{jk}为用户i、j对项目k的真实评分，而$\overline{R_i}$、$\overline{R_j}$是用户i、j对所有物品评分的平均值。其计算公式为：

$$sim(i,j)=\frac{\sum_{k\in n}\left(R_{ik}-\overline{R_i}\right)\left(R_{jk}-\overline{R_j}\right)}{\sqrt{\sum_{k\in n}\left(R_{ik}-\overline{R_i}\right)^2}\sqrt{\sum_{k\in n}\left(R_{jk}-\overline{R_j}\right)^2}}. \tag{3.21}$$

从上文分析知，基于用户聚类的 LFM 算法流程如下：

算法 3.1 基于用户聚类的 LFM 算法

输入：评分矩阵 **R**，用户集合U，聚类个数 n，隐特征个数 $\boldsymbol{k}$

输出：预测评分矩阵 **P**，**Q**

初始化：随机初始化用户中心 $\boldsymbol{\mu}_1$，$\boldsymbol{\mu}_2$，…，$\boldsymbol{\mu}_n$

1. 计算出样本集中用户总数 N
2. *for* $\boldsymbol{i}=1$ to N
3. *for* $j=1$ to N
4. 公式(3.21)计算U_i到各个用户中心$\boldsymbol{\mu}_j$的相似度$sim(U_i,\boldsymbol{\mu}_j)$
5. 对所有$sim(U_i,\boldsymbol{\mu}_j)$从大到小排序
6. end *for*
7. 迭代所有$sim(U_i,\boldsymbol{\mu}_s)=\max\{sim(U_i,\boldsymbol{\mu}_1),sim(U_i,\boldsymbol{\mu}_2),\ldots,sim(U_i,\boldsymbol{\mu}_n)\}$
8. 更新用户中心$\boldsymbol{\mu}_i$
9. end *for*
10. *for* $\boldsymbol{i}=1$ to n
11. 提取每个聚类用户中的项目$\boldsymbol{i_pool_i}$，按照评分进行排序
12. end *for*
13. 随机初始化$\mathbf{P}_u$，$\mathbf{Q}_i$
14. *for* $\boldsymbol{Iter}=1$ to $\boldsymbol{max\_iter}$
15. *for* $\boldsymbol{i}=1$ to N
16. 从$\boldsymbol{i_pool_i}$中选取正负样本
17. *while* $\boldsymbol{j}\in\boldsymbol{i\_pool_i}$
18. *for* $m=1$ to $\boldsymbol{k}$
19. 根据公式(2.17)更新$\mathbf{P}_u$
20. 根据公式(2.18)更新$\mathbf{Q}_i$
21. end *for*
22. end *while*
23. end *for*

24. end *for*

3.4 实验与分析

3.4.1 实验数据及评测标准

本章实验使用了 MovieLens 数据集，它是由 GroupLens 项目组创办的一个包含大、中、小规模的推荐系统数据集。其中小规模数据 *100k* 包含了 1000 个用户对 1700 部电影的 10000 条评分记录；中规模数据 *1M* 包含了 6000 个用户对 4000 部电影的 1000000 个评分；大规模数据 *10M* 包含了 72000 个用户对 10000 部电影的 10000000 个评分和 100000 个标签。本章实验使用了中、小规模两个数据集对实验数据按照 8：2 划分训练集（train data）与测试集（test data）与此同时保证了每个用户都至少有 20 个评分数据。

本实验分别使用平均绝对误差（MAE）、F1 值、覆盖率（Coverage）作为实验的评测指标。MAE 是常用的评测指标能够准确描述预测矩阵和评分矩阵的差值，其值越小说明算法性能越好；F1 值是对准确率（Precision）和召回率（Recall）的均衡，F1 值越高，算法性能越好；Coverage 则是描述推荐的覆盖广度，是一种商业规则，一般认为 Coverage 越大越好。给定测试集中用户 u 与项目 i，r_{ui} 表示用户 u 对项目 i 的真实评分，$\hat{r}_{ui}$ 表示用户 u 对项目 i 的预测评分，故 MAE 计算公式为：

$$\mathrm{MAE}=\frac{\sum_{u,i\in T}\left|r_{ui}-\hat{r}_{ui}\right|}{\left|T\right|}. \tag{3.22}$$

F1 值的计算公式为：

$$F1=\frac{2\times\mathrm{Precision}\times\mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}}. \tag{3.23}$$

其中 Precision 指的是推荐列表中真正命中的比例，Recall 指的是推荐列表中真正命中的在总测试集的比重，对于用户 u 推荐的 N 个物品组成的集合记作 $R(u)$，用户 u 在测试集上喜欢的物品组成的集合记作 $T(u)$。

$$\mathrm{Precision}=\frac{\sum_u|R(u)\cap T(u)|}{\sum_u|R(u)|}, \tag{3.24}$$

$$\mathrm{Recall}=\frac{\sum_u|R(u)\cap T(u)|}{\sum_u|T(u)|}. \tag{3.25}$$

Coverage 的计算公式为：

$$\text{Coverage} = \frac{\left|\bigcup_{u \in U} R(u)\right|}{|I|} . \tag{3.26}$$

3.4.2 实验结果及分析

实验开始前首先确认隐类个数 ***k***，本次实验 ***k*** 值分别选取 10、50、100、200、400 进行试验，实验所比较的算法是传统 LFM 以及使用用户聚类的 LFM（以下叫做 UC-LFM）算法，实验的评测的标准使用 MAE 和 F1 值。对于图 3.6 来说，10<***k***<200 时 LFM 和 UC-LFM 算法的 MAE 都是呈现下降趋势，但是当 ***k***>200 时，MAE 明显上升，并且随着 ***k*** 值的增大 MAE 也随之增大；对于图 3.7 来说，同样 10<***k***<200 时这两个算法的 F1 值都在不断增长，尤其是 ***k***=200 时增幅最大，由此说明 ***k***=200 时算法效率最佳，所以一下实验均使用 ***k***=200。同样从图上可以看出 UC-LFM 算法在 MAE 和 F1 值上均由于传统 LFM 算法。由此说明正负样本选取确实对 LFM 算法性能有极大的影响。

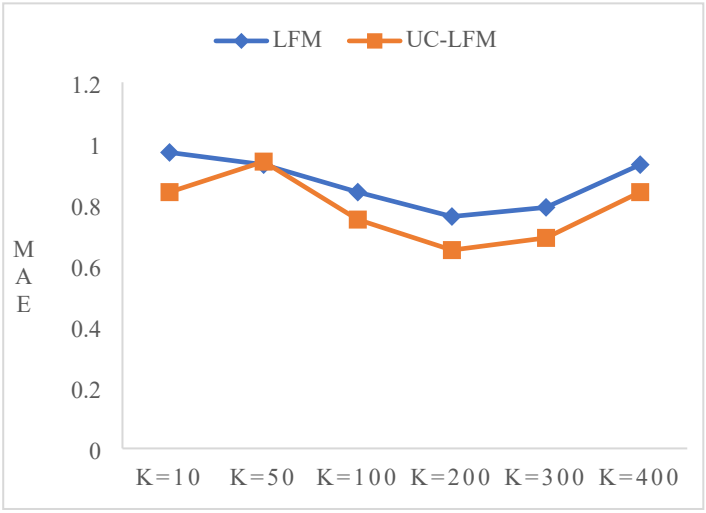

图 3.6 不同 k 值下算法的 MAE

Figure 3.6 MAE of the algorithm under different K values

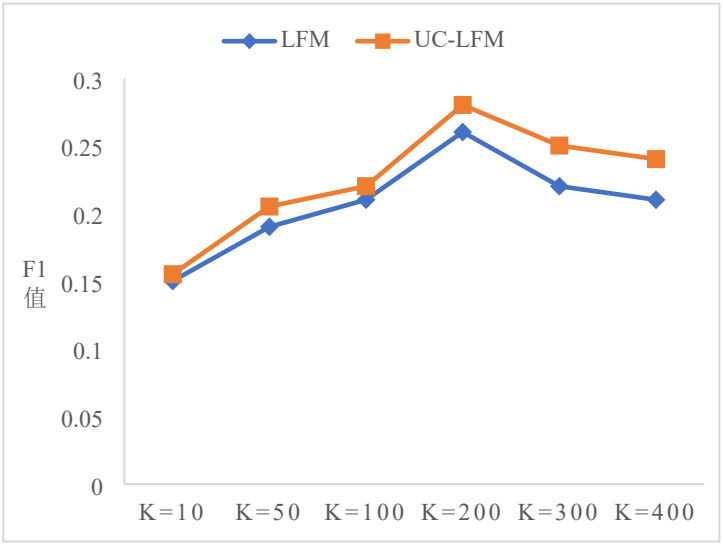

图 3.7 不同 ***k*** 值下算法的 F1 值

Figure 3.7 F1 values of the algorithm under different K values

得到了最优的 ***k*** 值，接着研究最佳聚类个数 ***n*** 的取值。本次实验算法为 UC-LFM 算法，使用那个评测指标为 MAE，***n*** 的取值为 5、10、15、20、50。实验结果如图 3.8，图中，***n***=5 时，算法的 MAE 和 LFM 算法相近，说明此时聚类效果不明显，当 ***n***=10 时，MAE 降至最低，同样 ***n***=15、20 时 MAE 变化不大。直到 ***n***=50 时算法的 MAE 出现了暴涨，这是由于当聚类个数很大时，每个聚类的用户就会很少，同样项目池中的项目也会很少，这样正样本会比负样本多得多，从而导致算法无法收敛，MAE 持续上升。

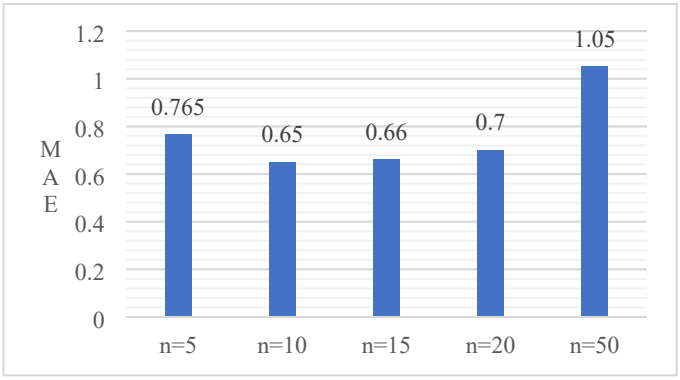

图 3.8 不同 ***n*** 值 UC-LFM 算法的 MAE

Figure 3.8 MAE of UC-LFM algorithm with different N values

得到了最优的 ***n*** 值，下面对比 LFM、B-LFM、UC-LFM 的 MAE、F1 值以及覆盖率，实验中所有的超参数均使用最优值，本次采用的数据集时小规模数据集，从图 3.9 中可以的结果来看，首先将算法按照精确率排序为 UC-LFM>B-LFM>LFM，由此可得带偏置的 LFM 和用户聚类的 LFM 算法均优于传统 LFM 算法，并且 UC-LFM 算法优于 B-LFM 算法。然而以覆盖率而言，B-LFM 算法遥遥领先于另外两种算法，而另外两个算法没有明显差异，由此说明添加偏置确实能够提高系统的覆盖率。

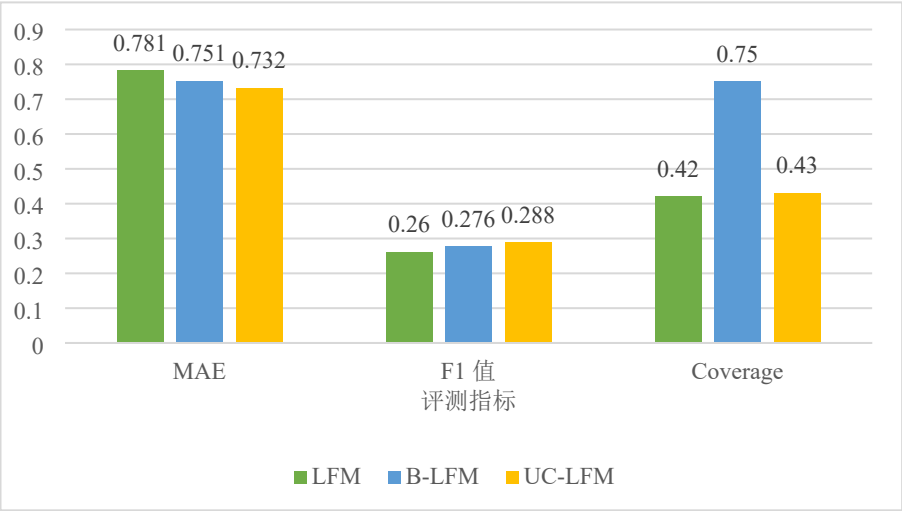

图 3.9 不同算法性能对比

Figure 3.9 Performance comparison of different algorithms

3.5 本章小结

本章开始介绍了矩阵分解的相关知识，提出了基于 SVD 的矩阵分解算法，通过分析其不足优化出 LFM 算法，通过综合考虑提出加偏置的 B-LFM 算法，并且通过实验证明了 B-LFM 算法整体上要优于 LFM 算法。通过分析正负样本选取原则，提出了一种基于用户聚类选取负样本的算法，并将其整合到 LFM 算法之中，通过实验验证其 MAE、F1 值均优于 LFM 算法。

第 4 章 结合动量训练优化后的隐语义模型

上一章提出了一种基于用户兴趣的隐语义模型，并且在此模型中添加了用户聚类算法，通过大量实验证明，改进后的 B-LFM 算法优于传统的 LFM 算法，UC-LFM 在 MAE、F1 值上也优于 B-LFM 算法，证明了负样本的选取对 LFM 算法影响最大。

本章继续对 LFM 算法进行优化，提出全量样本训练和增量样本训练的方法，并且通过分析各自的优缺点将其混合使用提出添加动量的训练方法（MO-LFM）。通过实验证明了 MO-LFM 算法性能远超过 LFM 算法。最后将结合第三章提出的算法，将所有算法进行对比。

4.1 全量样本训练

4.1.1 全量样本训练概念

LFM 算法具有以下若干变量：用户特征矩阵 **P**、项目特征矩阵 **Q**、学习率 α 以及正则化参数 λ。算法的思想是迭代更新出 **P**、**Q**，下面介绍全量样本训练算法主要步骤

算法 4.1 全量样本训练 LFM 算法

输入：评分矩阵 **R**，用户集 U

输出：迭代后的 **P**，**Q**

初始化：最大迭代次数 ***max_iter***，学习率 $\boldsymbol{\alpha}$，隐特征个数 k

1. 随机初始化 $\mathbf{P}_u$， $\mathbf{Q}_i$，生成 ***i _ pool***
2. *for* ***Iter*** $=1$ to ***max_iter***
3. *while* ***user*** $\in$ U
4. 从 ***i _ pool*** 中选取正负样本
5. *while* ***j*** $\in$ ***i _ pool***
6. *for* ***m*** $=1$ to ***k***
7. 根据公式(2.17)更新 $\mathbf{P}_u$
8. 根据公式(2.18)更新 $\mathbf{Q}_i$
9. end *for*
10. end *while*
11. end *while*
12. end *for*

通过算法步骤可知，学习率 α 会决定训练时间的快慢，α 过大可能会导致过拟合，

从而算法无法收敛。而 α 过小虽然会让算法性能得以提升，但是带来的问题是算法的训练时间将会漫长。这对于一个推荐系统来说也是绝不允许的。

另一个影响算法性能的因素是 **P**、**Q** 初始值的选取，一般来说 **P**、**Q** 初始值有三种选取方式：全 0、全 1、随机值。其中选取全 0 或者全 1 矩阵算法效率不佳，故常使用随机值将 **P**、**Q** 矩阵进行填充。然而随机值的选取如果过于随意，可能会导致算法非常不稳定。所以随机值选取一般符合范围内的高斯分布，由于预测矩阵为 **P** 和 **Q** 的乘积，通过实验证明，让评分矩阵与隐类 k 成反比会提高算法的性能，所以 **P**、**Q** 取值遵从一下两个原则：服从高斯分布、与 $\sqrt{K}$ 成反比。

4.1.2 优缺点分析

全量样本训练算法训练过程简单，参数选取合理的情况下，在小规模数据集中具有良好的性能。

但是，在推荐系统中，用户对项目的评分是极其少的，所以矩阵的稀疏性是无法避免的。因此全量样本训练算法中梯度值会有非常大的震荡幅度。除此之外，全量样本训练算法每次更新迭代 **P**、**Q** 均需要遍历整个训练集，计算量非常大。为了保证算法收敛，又必须设定比较小的学习率，因此对大规模训练集，全量样本训练算法的性能不佳。

4.2 增量样本训练

4.2.1 增量样本训练概念

针对全量样本训练算法的缺点提出了一种增量样本训练算法，与全量样本训练算法不同的是，该算法仅仅扫描了部分训练集数据，迭代出的 **P**、**Q** 也只是部分更新。该算法的思想是在考虑单个用户 u，用户的变化仅仅和以下两部分有关：用户特征向量 $\mathbf{P}_u$、用户 u 给过评分的项目。

对于单个用户 u 来说其损失函数定义如下：

$$Loss_u = \sum_{i=1}^{n} \mathbf{I}_{ui}\left(r_{ui} - \hat{r}_{ui}\right)^2 + \lambda \mathbf{P}_u{}^2 + \lambda \sum_{i=1}^{n} \mathbf{Q}_i{}^2 . \tag{4.1}$$

公式(4.1)对 P_u 和 Q_i 求偏导得：

$$\frac{\partial Loss_u}{\partial \mathbf{P}_u} = -2\sum_{i=1}^{n} \mathbf{I}_{ui}\left(\left(r_{ui} - \hat{r}_{ui}\right)\mathbf{Q}_i\right) + 2\lambda \mathbf{P}_u , \tag{4.2}$$

$$\frac{\partial Loss_u}{\partial \mathbf{Q}_i} = -2\mathbf{I}_{ui}\left(\left(r_{ui} - \hat{r}_{ui}\right)\mathbf{P}_u\right) + 2\lambda \mathbf{I}_{ui}\mathbf{Q}_i . \tag{4.3}$$

公式中 $\mathbf{I}_{ui}$ 有以下含义：

$$\mathbf{I}_{ui}=\begin{cases}0,\text{用户}u\text{对项目}i\text{未评分}\\1,\text{用户}u\text{对项目}i\text{已评分}\end{cases}.$$

对于公式(4.3)，如果用户 u 对项目 i 没有过评分，即 $\mathbf{I}_{ui}=0$ 时，$\mathbf{Q}_i$ 的梯度为 0，因此不需要对用户 u 没有评分的特征向量进行更新。

增量样本训练算法与全量样本训练算法不同之处在于，更新项目特征向量只需要扫描部分评分用户，对于单个评分就可更新一次。该算法的损失函数定义为：

$$Loss=\sum_{u=1}^{m}\sum_{i=1}^{n}\mathbf{I}_{ui}\left(r_{ui}-\hat{r}_{ui}\right)^2+\lambda\sum_{u=1}^{m}\mathbf{P}_u^{\,2}+\lambda\sum_{u=1}^{m}\sum_{i=1}^{n}\mathbf{I}_{ui}\mathbf{Q}_i^{\,2}. \tag{4.4}$$

上式中对每个项目的特征向量都添加了相应的系数 $\sum_{i=1}^{n}\mathbf{I}_{ui}$，其代表项目 i 存在评分的数量。所以评分越多的项目，系数越大。下面介绍增量样本训练算法主要步骤：

算法 4.2：增量样本训练 LFM 算法

输入：评分矩阵 **R**，用户集 U

输出：迭代后的 **P**，**Q**

初始化：最大迭代次数 $\boldsymbol{max\_iter}$，学习率 $\boldsymbol{\alpha}$，隐特征个数 $\boldsymbol{k}$

1. 随机初始化 $\mathbf{P}_u$，$\mathbf{Q}_i$，生成 $\boldsymbol{i\_pool}$
2. *for* $\boldsymbol{Iter}=1$ to $\boldsymbol{max\_iter}$
3. 从 $\boldsymbol{i\_pool}$ 中选取正负样本 $\boldsymbol{i\_pool_{user}}$
4. *while* $\boldsymbol{user}\in$ U
5. 根据公式(4.2)计算 $\nabla\mathbf{P}_{uk}$，更新 $\mathbf{P}_u\leftarrow\mathbf{P}_u-\boldsymbol{\alpha}\nabla\mathbf{P}_{uk}$
6. *while* $\boldsymbol{j}\in\boldsymbol{i\_pool}$
7. *for* $\boldsymbol{m}=1$ to $\boldsymbol{k}$
8. 根据公式(4.3)计算 $\nabla\mathbf{Q}_{ik}$，更新 $\mathbf{Q}_i\leftarrow\mathbf{Q}_i-\boldsymbol{\alpha}\nabla\mathbf{Q}_{ik}$
9. end *for*
10. end *while*
11. end *while*
12. end *for*

4.2.2 优缺点分析

增量样本训练算法不需要遍历整个训练集，并且每个评分值只更新一次，对学习率 α 设置很小的情况下，也能保证较快的训练速度，对于大规模样本，该算法表现十分优异。但是，该算法模型较为复杂，实现较为困难，此外对于小规模的数据集来说，

该算法可能会退化为全量样本训练算法，不能保证其精确度。

4.3 结合动量训练隐语义模型

前两节分析了全量样本训练算法和增量样本训练算法的优缺点。对于全量样本训练算法来说学习率α设置过小会导致算法学习时间过长，而过大又不能保证其收敛。增量样本训练算法虽然能改善以上的缺点，但是面对小规模数据集该算法表现并不是十分优异。由此结合两者的优点提出了结合动量训练隐语义模型的算法（MO-LFM）。

4.3.1 传统 LFM 算法整合

根据上一节结论可知，全量样本训练算法要想精确率比较高，这时学习率α要设置很小，这时带来的问题是训练时间过长，而且在大批量样本中性能也不佳。对此本节提出了在原有算法中添加用户动量 **M** 和项目动量 **N**，来提高算法效率，并且降低学习时间。其中用户动量 **M** 表示上一次迭代 $\nabla \mathbf{P}_u$ 的下降值，项目动量 **N** 表示上一次迭代 $\nabla \mathbf{Q}_i$ 的下降值，为了防止过拟合添加了 β 作为其能接受上次迭代梯度下降的程度。动量 **M** 和动量 **N** 的添加可以加快 **P**、**Q** 的寻优时间，下面给出算法的具体过程：

算法 4.3 添加动量训练 LFM 算法

输入：评分矩阵 **R**，用户集 U

输出：迭代后的 **P**，**Q**

初始化：迭代次数 $\boldsymbol{max\_iter}$，学习率 $\boldsymbol{\alpha}$，隐特征个数 $\boldsymbol{k}$，动量 M，N，系数 β

1. 随机初始化 $\mathbf{P}_u$，$\mathbf{Q}_i$，生成 $\boldsymbol{i\_pool}$
2. *for* $\boldsymbol{Iter} = 1$ to $\boldsymbol{max\_iter}$
3. 从 $\boldsymbol{i\_pool}$ 中选取正负样本 $\boldsymbol{i\_pool_{user}}$
4. *while* $\boldsymbol{user} \in$ U
5. *while* $\boldsymbol{j} \in \boldsymbol{i\_pool}$
6. *for* $\boldsymbol{m} = 1$ to $\boldsymbol{k}$
7. 令 $\mathbf{M}_u \leftarrow \beta \mathbf{M}_u$，$\mathbf{N}_i \leftarrow \beta \mathbf{N}_i$
8. 根据公式(4.2)(4.3)计算 $\Delta \mathbf{P}_{uk}, \Delta \mathbf{Q}_{ik}$，更新 $\mathbf{M}_u, \mathbf{N}_i$
9. $\mathbf{M}_u \leftarrow \mathbf{M}_u - \alpha \Delta \mathbf{P}_{uk}, \mathbf{N}_i \leftarrow \mathbf{N}_i - \alpha \Delta \mathbf{Q}_{ik}$
10. 更新 $\mathbf{P}_u \leftarrow \mathbf{P}_u + \mathbf{M}_u, \mathbf{Q}_i \leftarrow \mathbf{Q}_i + \mathbf{N}_i$
11. end *for*
12. end *while*
13. end *while*
14. end *for*

4.3.2 用户聚类 LFM 算法整合

第三章提到了使用用户聚类提取正负样本，并且在后面通过实验证明了算法的可行性，下面使用本章提出的动量训练算法与其整合得到基于用户聚类以及动量训练的隐语义模型（以下称为 UC-MO-LFM 算法），下面是算法的整体流程图：

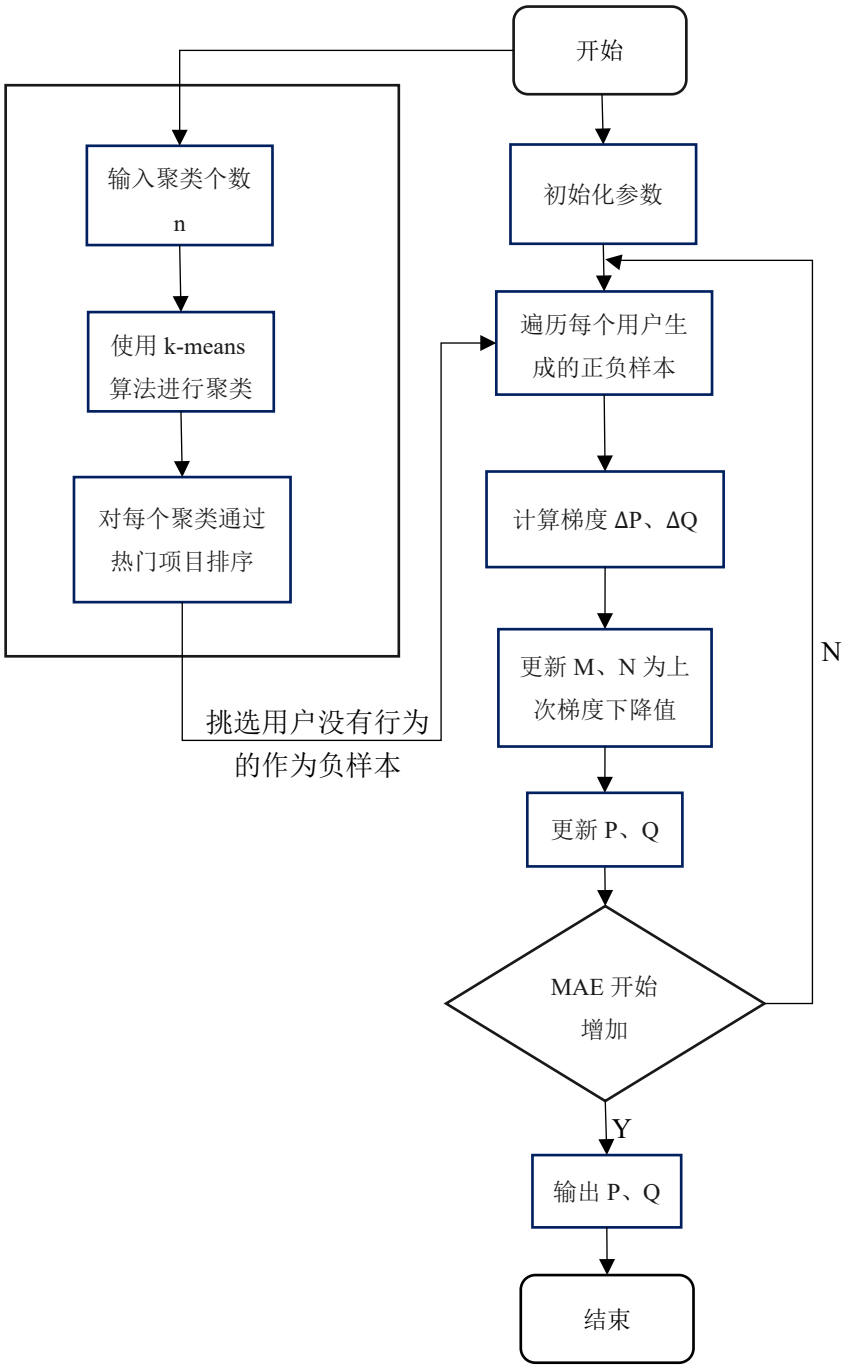

图 4.1 UC-MO-LFM 算法流程图

Figure 4.1 Flow chart of UC-MO-LFM algorithm

下面总结该算法的整体过程：

（1）预处理阶段

该阶段主要是对输入的数据进行处理，比如去噪，二值化，分析样本，生成评分矩阵。接着对所有用户进行聚类，这里使用的聚类方法是 *k*-means 聚类算法，最后对每个聚类簇中的用户有过行为的项目进行热度排序。

（2）初始化阶段

该阶段对模型中的参数进行初始化，参数主要包含用户特征矩阵 **P**、项目特征矩阵 **Q**、隐特征维数 ***k***、学习率 α、正则化系数 λ 以及动量 **M**、**N** 及其相关系数。

对于 **P**、**Q** 的初始化一般采用随机值的方法，并且该随机值满足与 $\sqrt{K}$ 成反比的高斯分布；大量实验证明，***k***=200 时算法最优；学习率 α 设置的过大可能导致算法过拟合无法收敛，而设置的过小可能导致训练时间过于漫长；正则化系数要在 0 到 1 之间；由 **M**、**N** 含义知初始化时上一阶段的梯度下降值为 0，故 **M**、**N** 均使用 **0** 矩阵进行初始化。

（3）训练阶段

该阶段是算法最重要的阶段之一，对于每个用户遍历其正负样本，首先更新 **P**、**Q** 上一阶段的梯度下降值，从而更新出 **P**、**Q**、**M**、**N**。

4.4 实验与分析

4.4.1 实验数据及评测标准

本次实验依然采用使用了 MovieLens 数据集，它是由 GroupLens 项目组创办的一个包含大、中、小规模的推荐系统数据集。其中小规模数据 *100k* 包含了 1000 个用户对 1700 部电影的 10000 条评分记录；中规模数据 *1M* 包含了 6000 个用户对 4000 部电影的 1000000 个评分；大规模数据 *10M* 包含了 72000 个用户对 10000 部电影的 10000000 个评分和 100000 个标签。各常见数据集的稀疏度如表 4.1 所示。本次实验使用了大、中、小规模三个数据集进行对比试验，实验数据按照 8：2 划分训练集（train data）与测试集（test data）并且保证了每个用户都至少有 20 个评分数据。

表 4.1 常见数据集的稀疏度

Table 4.1 Sparsity of Common Data sets

序列号	数据集	稀疏度(%)
1	MovieLens	4.5
2	Netflix Prize	1.2
3	Bibsonomy	0.35
4	Delicious	0.046

本章实验评测标准依然使用 MAE（见式(3.22)）和 F1 值（见式(3.23)），F1 值越大说明算法性能越好，反之，推荐性能越差；MAE 越小说明，预测准确度越高，反之，

准确度越低。

本章实验环境见表 4.2：

表 4.2 实验环境

Table 4.2 Experimental Environment

类别	规格
操作系统(OS)	Win 10 专业版
处理器(CPU)	Intel i7-9750H
内存(Memory)	24 G
开发环境(IDE)	Pycharm 2020 Community
使用语言及版本(Language&Version)	Python 3.8
数据开发包(SDK)	Numpy, Pandas, Sklearn
实验数据集(DataSource)	MovieLens 大中小规模

4.4.2 实验结果及分析

本次实验中对比的算法主要有：全量样本训练的隐语义模型（LFM(FS)）、增量样本训练的隐语义模型（LFM(IS)）、带偏置的隐语义模型（B-LFM）结合动量训练的隐语义模型（MO-LFM）、结合动量训练用户聚类的隐语义模型（UC-MO-LFM）。其中 LFM(FS)、LFM(IS)以及 MO-LFM 分别采用大、中、小三中规模的数据集对比，其余的算法均使用中等规模的数据集对比。

常见实验方式有控制变量法、比较法、类比法等，对于 LFM(FS)、LFM(IS)以及 MO-LFM 算法采用比较法，这种方法能够比较直观的看出算法的优劣性。而对于其余算法一般使用控制变量法进行试验。控制变量法是研究中使用的最多的方法，该方法每次实验固定某一个或多个量不变，进而研究其他量对实验结果的影响。

实验分为四个部分：

（1）最优 α 值

本轮实验目的是找出各个算法的最佳 α 值，实验选取了小规模的数据集，实验中隐类维数 k=200。实验结果如图 4.2 所示。图中 LFM(FS)算法和 LFM(IS)算法近乎重合，这就验证了，小规模数据集下全量样本训练算法和增量样本训练算法效率相当，在 α 值过大时，增量样本训练算法性能可能还不如全量样本训练算法。相比之下，融合了两者的优点提出的 MO-LFM 算法的性能要远远高于以上两个算法。从图中知，α=0.02 时，LFM(FS)算法、LFM(IS)算法以及 MO-LFM 算法的 MAE 均为最小，故下面实验选用 α=0.02。

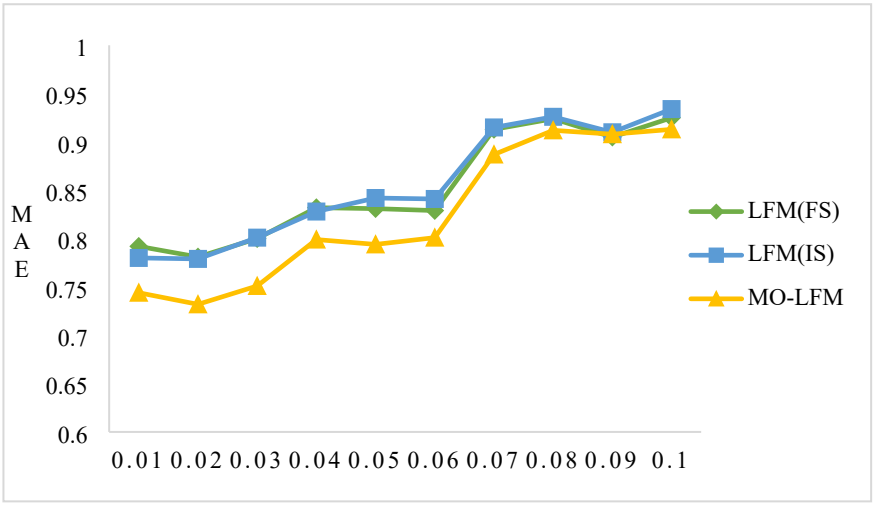

图 4.2 不同 α 值算法的 MAE

Figure 4.2 MAE for different α value algorithms

（2）不同规模数据集比较

本轮实验将分别使用大、中、小三中规模的数据集对以上三个算法的 MAE 进行比较，其中 α=0.02、k=200。实验结果如图 4.3 所示。图中很明显可以看出 LFM(FS)算法在数据集规模越大的情况下性能越差，而 LFM(IS)算法和 MO-LFM 算法受数据集规模影响较小。由于现实中的推荐系统用户量很大，项目也会很多，可能会比 MovieLens 大规模数据集还要大，故整体来说 LFM(IS)算法要优于 LFM(FS)算法，而 MO-LFM 算法优于 LFM(IS)算法。

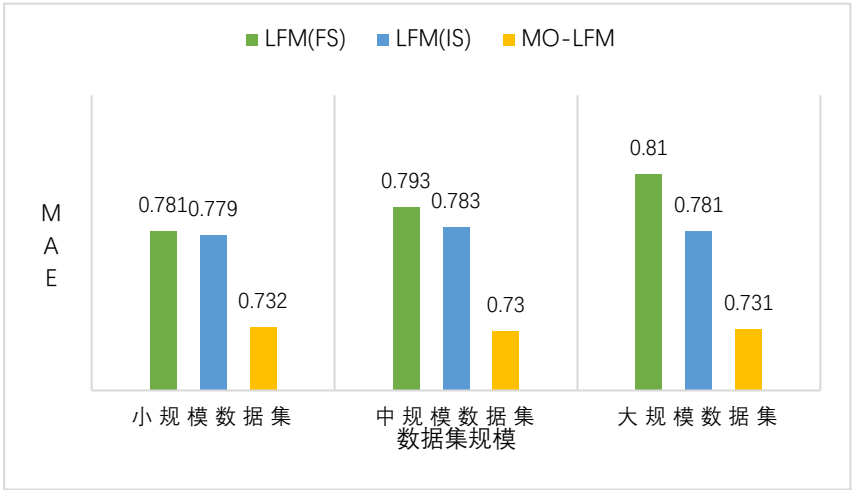

图 4.3 不同规模数据集算法的 MAE

Figure 4.3 MAE of different size data set algorithms

（3）最优 β 值

在进行下一项实验前需要找到最优的 β 值，本轮实验仅使用小规模数据集进行实验，同时 α=0.02、k=200，使用 MO-LFM 算法进行试验，找到最优的 β 值。实验结果

如图 4.4 所示。图中可以看出，β 值取得过大或者过小 MAE 都会很高，所以要均衡梯度的下降值，才能让 MO-LFM 算法充分体现。β=0.6 时 MAE 最小，故以下实验均使用 β=0.6。

（4）对比多个算法的 MAE 及 F1 值

本轮实验将对基于用户的协同过滤算法（UserCF）、基于项目的协同过滤（ItemCF）、带偏置的隐语义模型（B-LFM）、结合动量训练用户聚类的隐语义模型（UC-MO-LFM）的 MAE 和 F1 值进行对比，实验结果如图 4.5 所示。首先，UserCF 算法在算法精确度上要明显优于 ItemCF 算法，其次本次实验使用的优化后的 LFM 算法，证明 LFM 算法的性能要高于 UserCF 和 ItemCF 算法，最后本文提出的 UC-MO-LFM 算法性能更好。

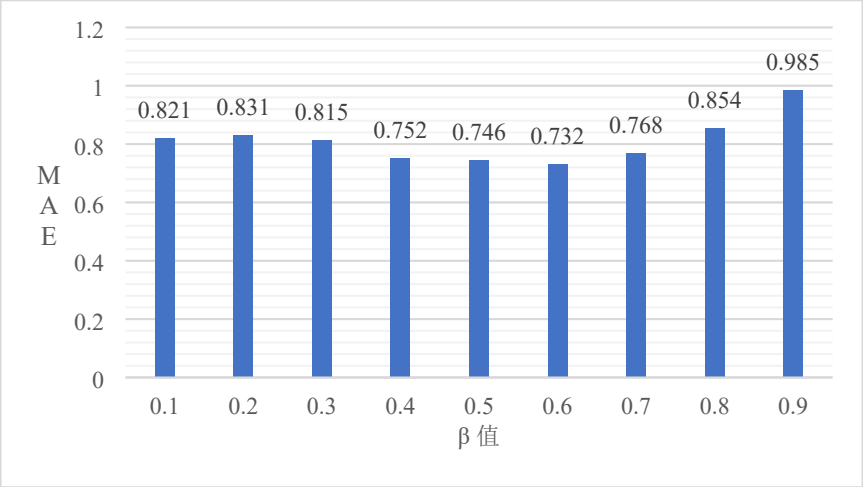

图 4.4 不同 β 值 MO-LFM 算法的 MAE

Figure 4.4 MAE with different β values of MO-LFM algorithm

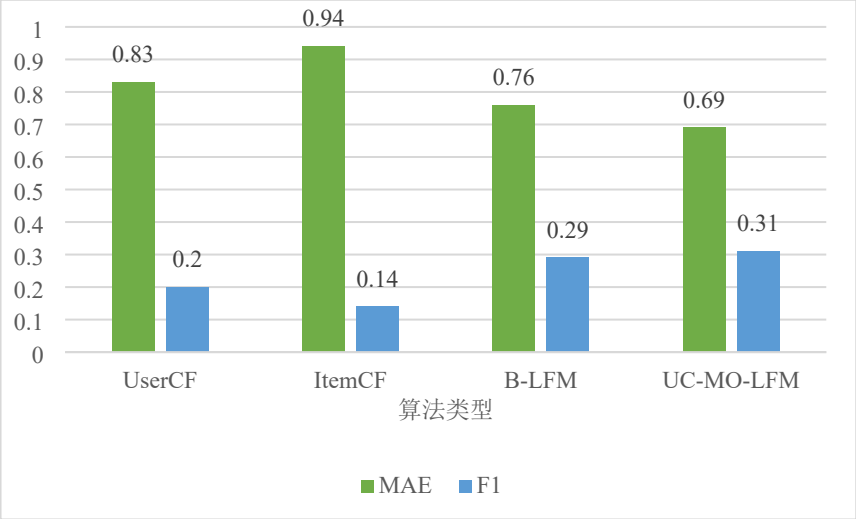

图 4.5 不同算法 MAE 和 F1 值比较

Figure 4.5 Comparison of MAE and F1 values of different algorithms

4.5 本章小结

本章首先介绍了隐语义模型两种样本训练的方式：全量样本训练和增量样本训练。全量样本训练在面临大规模数据集的时候，性能表现不佳，而在面临小规模数据集时，学习率 α 设置过大，可能导致算法无法收敛，而过小的学习率算法性能固然不错，但是寻优时间将会漫长；增量样本训练虽然能够解决以上问题，算法却也不是最优的，由此，提出结合动量训练的隐语义模型，然后将其与第三章提出的用户聚类算法进行整合，得出算法 UC-MO-LFM。最后通过大量实验证明了算法的优越性。

总结与展望

本文围绕经典的推荐模型——隐语义模型展开研究，主要通过两个维度对算法进行改进。

第一，对传统的 LFM 算法进行改进，将聚类算法应用于隐语义模型中，对具有相同兴趣的用户群体进行聚类，对每个群体挑选热门的自身没有行为的项目作为负样本，从实验来看，该方法能够大大提高算法的精确率；第二，对传统的 LFM 算法的训练过程进行改进，提出了结合动量训练隐语义模型，以实验验证了算法的优越性。最后将两种算法进行混合，以实验对比单一模型和混合模型，结果混合模型效果更佳。本文主要做的工作有以下四个方面：

（1）通过查阅各种文献，了解到影响 LFM 算法性能的因素主要有三个：隐特征维度、学习率、正负样本选取。大量实验证明，前两个对算法性能的影响较小，而正负样本选取合适与否会大大影响算法的精确度。负样本选取遵循的原则是用户最不可能喜欢的项目。通过学习聚类算法，提出了用户聚类为相似兴趣用户群体的思想，选择具有相同兴趣的用户喜欢的热门项目的负样本，用户极大可能不会喜欢，接着通过实验说明改进后算法 MAE 和 F1 值都优于传统 LFM 算法。

（2）介绍隐语义模型的两种样本训练方式：全量样本训练和增量样本训练。全量样本训练每次训练都要遍历所有的数据集，这样如果学习率设置的过小，虽然性能很好，但是带来的问题是，训练的时间过于漫长，反之，如果学习率设置的过大，可能导致算法无法收敛，精确率极低；增量样本训练仅仅只扫描了部分训练集，但是在样本规模比较小的时候可能也会扫描所有训练集，这样导致算法在小规模样本中提升不大。反之，在样本规模较大时，由于算法只扫描部分训练集，所以即使在较小的学习率下也能有不错的表现。

（3）吸取全量样本训练（LFM(FS)）和增量样本训练（LFM(IS)）的优点，提出结合动量训练隐语义模型（MO-LFM），添加用户动量 **M** 和项目动量 **N**，其代表了上次迭代梯度的下降值，将三个算法的 MAE 值分别在三个不同规模的数据集进行比较实验，在三个数据集中，MO-LFM 算法均是最优，同时数据规模越大 LFM(IS)算法比 LFM(FS)算法越好。

（4）将所有的工作融合提出了 UC-MO-LFM 算法，将该算法与 UserCF、ItemCF、LFM 算法的 MAE 和 F1 值进行实验比较，UC-MO-LFM 算法的性能最好，其次是 LFM 算法，UserCF 算法要优于 ItemCF 算法。

本文提出的算法仍然有以下几点需要改进的地方：

（1）本文对用户聚类时使用了常见的聚类算法：*k*-means 聚类算法，但是 *k*-means 聚类算法存在很多问题，该算法无法分辨噪声，对于某一个噪点数据，会将其视为特征数据加以使用，这样会影响聚类的效果，从而影响整体算法性能。对此可以使用更优的聚类算法。

（2）在使用相似度计算模型时使用较单一。可以使用一些改进后的相似度计算模型，比如：加入时间或者流行度的相似训练模型。也可以使用多个相似度计算模型进行比较，选择最合适的模型。

（3）本文的训练数据集均使用了 MovieLens 数据集，没有在稀疏度极小的数据集进行验证实验，也没有在稀疏性极大的数据集中验证结果。

参考文献

[1] 中国互联网信息中心. 第 46 次中国互联网发展状况统计报告[R]. 北京: 中国互联网信息中心, 2018.

[2] P. Resnick, H.R. Varian. Recommender systems[J]. Communications of the ACM, 1997, 40(3): 56-58.

[3] A. Borchers, J. Herlocker, et al. Ganging Up on Information Overload[J]. Computer, 1998, 31(4): 106-108.

[4] K. Srinivasa, G.M. Siddesh, H. Srinidhi. Network Data Analytics[M]. 2018: 303-318.

[5] Y. Shao, Y.H. Xie. Research on Cold-Start Problem of Collaborative Filtering Algorithm[C]. In: 2019 International Conference on Big Data Research, 2019, pp. 67-71.

[6] N. Fadhil, S.H. Hashim. Reducing Data Sparsity in Recommender Systems[J]. Journal of Al-Nahrain University Science, 2018, 21(2): 138-147.

[7] N Sharma, V. Suryawanshi. Association Rule in Recommendation to Reduce Scalability and Sparsity[J]. International Journal of Computer Applications, 2018, 182(8): 37-40.

[8] 李远博, 曹菡. 基于 PCA 降维的协同过滤推荐算法[J]. 计算机技术与发展, 2016, 26(2): 26-30.

[9] D. Goldberg, D.A. Nichols, B.M. Oki, et al. Using Collaborative Filtering to Weave an Information Tapestry[J]. Communications of the ACM, 1992, 35(12): 61-70.

[10] P. Resnick, N. Iacovou, M. Suchak, et al. GroupLens: an Open Architecture for Collaborative Filtering of Netnews[J]. Working Paper Series, 1994: 75-186.

[11] A. Mehta, A. Saberi, U.V. Vazirani, et al. Awords and Generalized Online Matching[J]. Journal of the ACM, 2007, 54(5): 22.

[12] 王光，张杰民，董帅含，等. 基于内容的加权粒度序列推荐算法[J]. 计算机工程与科学, 2018, 40(3): 564-570.

[13] N. Saat, S.A.M. Noah, M. Mohd. Towards Serendipity for Content–Based Recommender Systems[J]. International Journal on Advanced Science, Engineering and Information Technology, 2018, 8(4): 1762-1769.

[14] Z.Y. Cheng, Y. Ding, L. Zhu, et al. Aspect-Aware Latent Factor Model: Rating Prediction with Ratings and Reviews[C]. In: 2018 International Conference of World Wide Web, 2018, pp. 639-648.

[15] R. Burke, H. Abdollahpouri, E.C. Malthouse, et al. Recommendation in Multistakeholder Environments[C]. In: 2019 ACM Conference on Recommender Systems, 2019, pp. 566-567.

[16] S. Debnath, N. Ganguly, P. Mitra. Feature Weighting in Content Based Recommendation System

Using Social Network Analysis[C]. In: 2008 International Conference of World Wide Web, 2008, pp. 1041-1042.

[17] H. Cramer, V. Evers, S. Ramlal, et al. The Effects of Transparency on Trust and Acceptance in Interaction with a Content-Based Art Recommender [J]. User Modeling and User-Adapted Interaction, 2008, 18(5): 455-496.

[18] H.C. Wu, R.W.P. Luk, K.F. Wong, et al. Interpreting TF-IDF Term Weights as Making Relevance Decisions[J]. ACM Transactions on Information Systems, 2008, 26(3): 55-59.

[19] L. Martínez, L.G. Pérez, M.J. Barranco. A Multigranular Linguistic Content-Based Recommendation Model[J]. International Journal of Intelligent Systems, 2007, 22(5): 419-434.

[20] A. Zenebe, A.F. Norcio. Representation, Similarity Measures and Aggregation Methods Using Fuzzy Sets for Content-Based Recommender Systems[J]. Fuzzy Sets and Systems, 2009, 160(1): 76-94.

[21] R.J. Mooney, L. Roy. Content-Based Book Recommending Using Learning for Text Categorization[C]. In: 2000 ACM Conference on Digital Libraries, 2000, pp. 195-204.

[22] P. Cano, M. Koppenberger, N. Wack. Content-Based Music Audio Recommendation[C]. In: 2005 ACM International Conference 2005, pp. 211-212.

[23] K. Sugiyama, K. Hatano, M. Yoshikawa. Adaptive Web Search Based on User Profile Constructed without any Effort From Users[C]. In: 2004 International Conference of World Wide Web, 2004, pp. 675-684.

[24] B. Sarwar, G. Karypis, J. Konstan, et al. Analysis of Recommendation Algorithms for E-Commerce[C]. In: 2000 ACM Conference on Electronic Commerce, 2000, pp. 158-167.

[25] 周志华. 机器学习[M]. 北京: 清华大学出版社, 2016: 1-12.

[26] B Sarwar, G. Karypis, J. Konstan, et al. Application of Dimensionality Reduction in Recommender System: A Case Study[C]. In: 2000 Web Mining for E-Commerce Workshop, 2000, pp. 612-625.

[27] X. Luo, M.C. Zhou, Y.N. Xia, et al. Generating Highly Accurate Predictions for Missing QoS Data Via Aggregating Nonnegative Latent Factor Models[J]. Institute of Electrical and Electronics Engineers Transactions on Neural Networks and Learning Systems, 2015, 27(3): 524-537.

[28] Y. Koren, R. Bell. Advances in Collaborative Filtering[M]. Recommender Systems Handbook, 2015: 77-118.

[29] M. Rowe. SemanticSVD++: Incorporating Semantic Taste Evolution for Predicting Ratings[C]. In: 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence. 2014, pp. 213-220.

[30] R. Salakhutdinov, A. Mnih. Probabilistic matrix factorization[C]. In: 2007 International Conference on Neural Information Processing Systems, 2007, pp. 1257-1264.

[31] R Salakhutdinov, A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo[C]. In: 2008 International Conference on Machine Learning, 2008, pp. 880-887.

[32] R. Burke. Hybrid Recommender Systems: Survey and Experiments[J]. User Modeling and User-Adapted Interaction, 2002, 12(4): 331-370.

[33] R.D. Burke. Hybrid Systems for Personalized Recommendations[C]. In: 2003 International Joint Conference on Artificial Intelligence, 2003, pp. 133-152.

[34] D.R. Liu, Y.Y. Shih. Hybrid Approaches to Product Recommendation Based on Customer Lifetime Value and Purchase Preferences[J]. Journal of Systems and Software, 2005, 77(2): 181-191.

[35] J. Gemmell, T. Schimoler, B. Mobasher, et al. Resource Recommendation in Social Annotation Systems: A Linear-Weighted Hybrid Approach[J]. Journal of Computer and System Sciences, 2012, 78(4): 1160-1174.

[36] Y.Y. Shih, D.R. Liu. Hybrid Recommendation Approaches: Collaborative Filtering via Valuable Content Information[C]. In: 2005 Hawaii International Conference on System Sciences, 2005, pp. 217b.

[37] A. Albadvi, M. Shahbazi. A Hybrid Recommendation Technique Based on Product Category Attributes[J]. Expert Systems with Applications, 2009, 36(9): 11480-11488.

[38] D.R. Liu, C.H. Lai, W.J. Lee. A Hybrid of Sequential Rules and Collaborative Filtering for Product Recommendation[J]. Information Sciences, 2007, 179(20): 3505-3519.

[39] J. Chen, Y. Jian, H. Jin. Automatic Content-Based Recommendation in E-commerce[C]. In: 2005 Institute of Electrical and Electronics Engineering International Conference, 2005, pp. 748-753.

[40] Z. Huang, D.D. Zeng, H.C. Chen. A Comparison of Collaborative-Filtering Recommendation Algorithms for E-commerce[C]. In: 2007 Institute of Electrical and Electronics Engineering Intelligent Systems, 2007, pp. 68-78.

[41] Y.H. Zhou, D.M. Wilkinson, R. Schreiber, et al. Large-Scale Parallel Collaborative Filtering for the Netflix Prize[C]. In: 2008 International Conference on Algorithmic Applications in Management, 2008, pp. 337-348.

[42] Z.D. Zhao, M.S. Shang. User-Based Collaborative-Filtering Recommendation Algorithms on Hadoop[C]. In: 2010 International Conference on Knowledge Discovery and Data Mining, 2010, pp. 378-481.

[43] 李伟霖, 王成良, 文俊浩. 基于评论与评分的协同过滤算法[J]. 计算机应用研究, 2017, 34(2): 361-364, 412.

[44] 赵伟, 林楠, 韩英, 等. 一种改进的 K-means 聚类的协同过滤算法[J]. 安徽大学学报(自科版), 2016, 40(2): 32-36.

[45] 丁少衡, 姬东鸿, 王路路. 基于用户属性和评分的协同过滤推荐算法[J]. 计算机工程与设计, 2015, 36(2): 487-491, 497.

[46] 曾小波, 魏祖宽, 金在弘. 协同过滤系统的矩阵稀疏性问题的研究[J]. 计算机应用, 2010, 30(4):

1079-1082.

[47] 翟航天, 汪学明. 基于隐式反馈 LDA 模型的协同推荐算法研究[J]. 计算机技术与发展, 2019, 29(6): 7-12.

[48] 王晓耘, 李贤, 袁媛. 基于因子分解机和隐马尔可夫的推荐算法[J]. 计算机技术与发展, 2019, 29(6): 85-89.

[49] 郑英丽, 王新, 马倩, 等. 一种结合用户相似度的社会化推荐算法[J]. 云南民族大学学报(自然科学版), 2019, 28(1): 93-99.

[50] 卫鼎峰, 李梁, 柴晶. 融合物品信息的社会化推荐算法[J]. 计算机工程与应用, 2020, 41(11): 3047-3052.

[51] M.S. Fu, H. Qu, D. Moges, et al. Attention Based Collaborative Filtering[J]. Neurocomputing, 2018, 311(15): 88-98.

[52] X. Feng, X.N. He, X. Wang, et al. Deep Item-based Collaborative Filtering for Top-N Recommendation[J]. ACM Transactions on Information Systems, 2019, 37(3): 1-25.

[53] B. Loepp, J. Ziegler. Towards Interactive Recommending in Model-based Collaborative Filtering Systems[C]. In: 2019 ACM Conference on Recommender Systems, 2019, pp. 546-547.

[54] 陈希, 李玲娟. 基于降维和聚类的协同过滤推荐算法[J]. 计算机技术与发展, 2020, 30(2): 138-142.

[55] J.S. Breese, D. Heckerman, C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering[J]. Uncertainty in Artificial Intelligence, 2013, 98(7): 43-52.

[56] 李改, 李磊. 基于矩阵分解的协同过滤算法[J]. 计算机工程与应用, 2011, 47(30): 4-7.

[57] N. Ketkar. Deep Learning with Python[M]. 2017: 111-130.

[58] 陈彪. 基于 LFM 的音乐推荐系统的研究与实现[D]. 湖南大学硕士学位论文. 2017: 7-15

[59] H.H. Nguyen. Privacy-Preserving Mechanisms for K-Modes Clustering[J]. Computers and Security, 2018, 78: 60-75.

[60] K. Nakagawa, M. Imamura, K. Yoshida. Stock Price Prediction using K-Medoids Clustering with Indexing Dynamic Time Warping[J]. Institute of Electrical Engineers of Japan Transactions on Electrical and Electronic Engineering, 2018, 138(8): 986-991.

攻读学位期间的科研成果

[1] 第一作者, 计算机与数字工程, 已录用.

致谢

时光荏苒如弹指一挥，两年半的研究生生涯也即将结束，将要面临的是更大的挑战。回顾过去的两年多时间里，我学习了很多做人的道理，也结识了很多优秀的小伙伴，从他们身上我学习到了很多我之前没有的品质，也让我的研究生圆满结束。在此，我将向那些给与我帮助、爱护、鼓励的人们表达我最诚挚的谢意。

最应该感谢的是我的导师黄树成教授，他是一个学术修养非常高的老师，他常告诫我们，每个人身上都有我们不具备的品质，我们都应该虚心学习。在我论文选题、写作期间，黄树成老师给了我很多意见，使我能够更好的完成论文。除此之外，黄树成老师还将我推荐到上汽集团实习，实习期间，幸得冯海清师傅的指导，我才能完成我的任务，丰富自身的项目经验。同时我也要感谢我的同事们，他们丰富了我的业务经验。正因为有这么优秀的实习经验，我才能在秋招时找到一份满意的工作。

其次我要谢谢朱霞老师，这两年多时间里，她在学习上以及生活上都给我谆谆教诲，从她身上我也学到了很多优秀的品质。

接着我要感谢计算机学院的领导和老师们对我的帮助、关爱，在与领导的交流中我的沟通能力得到很大的提升，老师们教会了我很多计算机相关的专业知识，这也是我秋招能够顺利的一个关键因素。

我还要感谢我的师兄师姐，在刚入学的那段时间里，他们教育我如何和老师沟通，也给我传授了一些有用的学习经验。我要感谢同门的小伙伴们，他们陪我一起走过了这两年半的春夏秋冬，我们在学习工作上相互帮助，共同进步。感谢我研究生期间遇到的同学们，是你们让我能在温暖的集体中不断学习成长。

借此，我还要感谢我的父母，他们含辛茹苦的把我培养成为一个对社会有用的人，为我的学业提供物质、精神方面的保障，他们是我最可靠的依靠。

虽然即将离开校园，结束学生时代，但是学习一直在路上，成长一直在路上，我将怀着一颗感恩的心，以前辈为榜样，不忘初心，一步一个脚印地走好我未来的人生道路。

最后我要感谢各位参与评审的专家们能够在百忙之中抽出宝贵的时间对本论文进行评阅。在此我将由衷地感谢各位评审专家们。