电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

硕士学位论文

MASTER THESIS

论文题目 基于集成学习的个性化推荐算法研究

学科专业 计算机软件与理论

学 号 201021060215

作者姓名 陈高洁

指导教师 傅 彦 教 授

分类号 _____ 密级 _____

UDC [注1] _____

学 位 论 文

基于集成学习的个性化推荐算法研究

（题名和副题名）

陈高洁

（作者姓名）

指导教师 傅 彦 教 授

电子科技大学 成 都

（姓名、职称、单位名称）

申请学位级别 硕士 学科专业 计算机软件与理论

提交论文日期 2013-03-15 论文答辩日期 2013-05-07

学位授予单位和日期 电子科技大学 2013 年 06 月 29 日

答辩委员会主席 _____

评阅人 _____

注 1：注明《国际十进分类法 UDC》的类号。

PERSONALIZED RECOMMENDATION ALGORITHM BASED ON ENSEMBLE LEARNING

A Master Thesis Submitted to

University of Electronic Science and Technology of China

Major: **Computer Software and Theory**

Author: **Chen Gaojie**

Advisor: **Fu Yan**

School : **School of Computer Science and Engineering**

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

作者签名：_____ 日期： 年 月 日

论文使用授权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将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、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

作者签名：_____ 导师签名：_____

日期： 年 月 日

摘　要

随着互联网以及电子商务的飞速发展，用户陷入了一种信息迷失的状态，面对海量的商品用户往往在找到感兴趣的商品之前已经非常沮丧、烦躁，因此很可能会放弃此次购物。解决该问题的有效途径一般认为是个性化推荐，协同过滤技术在推荐系统发展初期效果非常不错，但随着用户数目和商品种类的不断增加，打分矩阵变得越来越稀疏，传统的相似度计算方法会产生不准确的相似邻居，从而导致推荐质量下降。另一方面影响推荐质量的因素是很多的，而当前的个性化推荐算法不是单一算法，就是几个算法的简单组合，这些算法普遍准确性不高。

集成学习通过集成多个弱学习器解决同一个问题，具有很好的泛化能力和稳定性，目前集成学习已经广泛应用到了很多领域，在最近几年的 Netflix、Learning to Rank 比赛中，顶级团队大多数使用的也是集成学习技术。因此若能将集成学习技术与个性化推荐相结合，必定能提高推荐系统的准确率。在电子商务快速发展的时代，提高个性化推荐系统的准确率是人们永远的追求目标，针对这一目标，本文的主要研究工作是：

1、对协同过滤中的传统相似度算法进行了分析，根据它们存在的缺点，提出了一种改进的协同过滤方法 PCS 算法。该算法使用决策树自适应调整分割属性的特点，将决策树用于用户的分组上，因此就产生了一种寻找相似邻居的新方法。用户分组后只在组内进行个性化推荐，避免了计算所有用户间的相似度，提高了算法的在线可扩展性。另一方面，针对 Pearson 相似度存在的缺点，从流行度、用户共同打分商品数和用户打分差异度三方面对 Pearson 进行了改进，提高了推荐算法的准确性。

2、针对目前大多数的算法都是从某一方面考虑提高推荐的准确率，而影响推荐质量的因素是很多的，集成学习算法 SoftBoost 集成多个弱学习器解决同一个问题，且该算法是最大化软间隔算法，能够避免难分数据的影响，有很强的泛化能力，因此本文将其和个性化推荐相结合，使用 SoftBoost 算法作为框架的同时，本文使用 RankBoost 中基于对的思想重新定义了间隔和边界，提出了 SoftRankBoost 算法。

关键词：个性化推荐，稀疏性，集成学习

ABSTRACT

With the rapid development of the e-commerce, customers fall into a status of "information lost". Facing a flood of items, users feel depression and irritability before they find the items they are interested in, so they may give up this shopping. Personalized recommendation is considered to be the most effective approach to this problem, and Collaborative Filtering has a good quality in the early development of recommendation system. However, with the increasing of the number of users and the types of goods, scoring matrix becomes more and more sparse, so traditional similarity algorithms will produce inaccurate similarity neighbors, resulting in a bad recommendation. On the other hand, there are many factors affecting the recommendation quality, but present CF algorithm is either a single algorithm or a simple combination of several algorithms. These present CF algorithms generally have a low accuracy.

Ensemble Learning has a good generalization capability and stability by combine multiple learners to solve the same problem, so now it has been widely applied to many fields. In Netflix and Learning to Rank challenge, most of the winning teams use Ensemble Learning. If we can combine Ensemble Learning and Recommendation, the accuracy of recommendation will be significantly improved. To improve the accuracy, this article's contributions include:

(1) According to analyze the shortcomings of traditional CF algorithm, this article proposed a modified CF method PCS algorithm. With the advantage of adaptively adjusting the split attribute, we use decision tree to group users, thus gave rise to a new method to find similar neighbors. Then CF algorithms can operate on the group instead of the whole scoring matrix, this improves the online scalability of recommendation. In addition to, this article modified the Pearson similarity via popularity, number of common rated items and rating difference , this algorithm improves the accuracy of recommendation.

(2) The factor affecting the quality of recommendation is many, but most of the present CF algorithm improve the accuracy of the recommendation from one aspect,

SoftBoost algorithm combine multiple learners to solve the same problem, what's more it is a soft margin algorithm, it can avoid the over-fitting ,so it has a good generalization capability and stability. We proposed SoftRankBoost algorithm, using SoftBoost algorithm as a framework, on the other hand using RankBoost’s opinion to give the margin and edge a new definition.

Keywords: personalized recommendation , sparsity, ensemble learning

目 录

第一章 绪 论

1.1 选题背景与研究意义

进入 21 世纪以来，电子商务行业蓬勃发展，这极大的便利了用户，人们不用出门便能随时随地挑选上百万种商品，然而伴随着商品种类和数量呈指数增长，用户很难准确、高效的找到自己所需的商品。面对海量的商品用户往往把大量的时间花费在浏览无关的信息上面，这使用户陷入一种“信息迷失”的状态，继而会导致消费者的不断流失。电子商务零售商必须提供给用户满意的商品，以提高他们的满意度和忠诚度,才能留住原有的消费者并吸引新的消费者。为解决这一问题，电子商务个性化推荐系统应运而生。

个性化推荐系统[1]将用户放在主导地位，智能化的获取用户对商品的偏好和购买行为，然后利用特定的推荐算法向目标用户推荐他最可能感兴趣和购买的商品。目前推荐系统已被广泛应用到了电子商务、电影和视频、音乐、社交网络等领域[2]，如美国最大的电子商务亚马逊公司、美国最大的视频网站 YouTube、国际著名的个性化音乐网络电台 Pandora，最具代表性的社交网络 Facebook 和 Twitter 等,它们都成功应用了个性化推荐系统，尽管不同网站使用不同的个性化推荐技术。

推荐算法是个性化推荐系统的核心步骤，在众多的推荐算法中，协同过滤是个性化推荐系统应用最广泛和成功的技术。协同过滤的核心思想是根据用户的以往兴趣偏好[3]，找到与目标用户品味相同或兴趣相似的邻居，根据这些邻居对其他商品的喜好程度预测目标用户对这些商品的兴趣度。亚马逊的书籍推荐系统[4]，豆瓣网的音乐、图书推荐系统，Spotify 的网络电台推荐系统使用的都是协同过滤。但随着电子商务的不断发展，打分矩阵会越来越大，同时也会越来越稀疏，这种情况使得协同过滤技术在推荐准确性和可扩展性上都面临严峻挑战。一方面由于用户数目和商品数量的急剧增加，打分矩阵维数变得很大，计算相似度时的时间复杂度急剧升高，因此会产生算法的扩展性问题。另一方面，打分矩阵中 90%以上都是未知的，协同过滤算法很难正确的计算用户间的相似度，无法形成可靠的最近邻居，往往会产生不正确的推荐结果。另外，现有的推荐算法往往只考虑了影响推荐的某一方面因素，但实际中影响推荐的因素是很多的。

集成学习(Ensemble Learning)是机器学习的热点研究问题，它集成了多个弱学习器来解决同一个问题，它能显著提高学习系统的泛化能力[5-7],这好比你感觉身体

不舒服，要找医生诊断但为了准确起见，我们往往会找多个医生诊断然后综合每个医生的诊断结果（每个医生的权重可能相同或者不同）。集成学习把一系列的弱学习算法进行训练，生成多个弱假设，然后通过某种组合方式把这些弱假设组合起来成为一个强学习算法。Kearns 和 Valiant 指出，弱学习算法为该算法的准确率仅仅比随机猜测好；强学习算法为该算法的准确率很高，并能在多项式时间内收敛。找到一个准确率优于 0.5 的算法是一件很容易的事，同时将它们集成后又能大大提高算法的性能，因此集成学习成了热点问题。

目前集成学习已经成功应用到了文本分类、图像处理、疾病诊断[8]、排序问题、语音识别等领域。集成学习相对于其它算法的优势在于：性能比较稳定，不像其它算法在有些数据集上效果很好，而在另一些不同的数据集上效果就很差；泛化能力较强，组合多个从不同因素考虑的算法，提高算法的准确度；在一定程度上避免过拟合现象，因为它综合了多个算法的结果，即使某一弱学习算法有过拟合现象，通过其它算法的均衡也能将这一现象缓和，不会产生严重的过拟合。

由于集成学习具有很强的泛化能力，对未知样本进行分类时准确率很高，对噪声数据不会有过拟合现象，且在多个数据集上都有很好的性能，而如前所述目前的个性化推荐技术准确率普遍不高，且在这个数据集上性能好的算法在另一个数据集上可能非常差，故可以借助集成学习的优势，把单一的推荐算法作为弱学习器将两者结合，改善个性化推荐的不足。本文正是从这方面出发，将集成学习与个性化推荐相结合，进一步提高了个性化推荐的效果。

1.2 国内外现状

1.2.1 集成学习现状

集成学习成为研究热点源于 Schapire[9]给出了一个性能仅优于随机猜测的弱学习器与强学习器之间存在着等价关系的证明，这一理论让很多研究者看到了集成学习的优势，引起了他们浓厚的研究兴趣，由此开创了集成学习发展的新历程。

1992 年 Xu 和 Suen[10]等人考虑使用多个分类器合并的方法进行手写体识别，他们经过试验证明这种方法可以大大提高单一分类器的准确率，同时能提高分类系统的鲁棒性。1994 年 Battiti 和 Colla[11]由于他们的过失意外发现使用两到三层隐藏层的 BP 神经网络进行集成的性能比最好的单一神经网络算法的性能要好得多。

1998 年，Shimshoni 利用神经网络集成的方法判断地震波是天然地震引起的还

是人为爆炸引起的。文中作者使用了不同层次结构的神经网络和三种不同时间频率的分解地震波，利用集成学习的强泛化能力，使用神经网络作为弱学习算法，地震波作为输入样本进行集成，这一过程产生了三个强学习算法，再将这三个强学习算法进行加权组合[7,12]。2002 年 Ridgeway [13]将 Boosting 和 Bagging 应用到密度估计上，使用 Boosting 侧重于分错样本的特点提升算法准确度，同时利用 Bagging 算法对噪声数据不敏感的特点解决过拟合问题，两者结合后取得了不错的效果。

2009 年 Leistner [14]研究了 On-line GradientBoost 算法的鲁棒性，提出了用于 On-line GradientBoost 的新损失函数，验证了使用该损失函数后算法的性能有了很大提升。2010 年 Tsai 和 Chen[15]等人提出将集成学习应用到 Rank 上，采用将基于点、基于对、基于列表三种方法相结合，试验中一般情况下基于点的比基于对的和基于列表的性能要好，但将三者结合后性能高于三者中的任何一个。

近几年来，集成学习在国内也成为了热点研究问题，在理论研究上“选择性集成”概念[16]第一次被周志华提了出来，他们同时提出了相关的 GASEN 算法，这一理论研究翻开了集成学习的新篇章。“选择性集成”思想打破了人们以往的思维方式，人们普遍认为要想获得更好的性能就必须有更多的基学习器。但有些生成的弱学习器之间可能有很强的关联性，对最后的集成结果没有太大帮助，且使用它们集成也会加重算法的时间代价，“选择性集成”选择差异比较大的学习器。

在应用领域，周志华[17]等人在 2000 年提出了集成的神经网络方法，该方法能够识别一定角度范围内的任何视角的人脸，如从左 30 度视角非平面旋转到右 30 度视角。最后的实验结果表明了使用神经网络进行集成算法的性能有了很大提升，甚至优于固定角度下单一算法的最好情况。2002 年刁力力等人将 Stumps 作为弱分类器，使用 Boosting 集成框架进行文本分类[18]，实验证明该方法进行文本分类效果显著。2005 唐伟和周志华将集成学习应用到聚类上，由于聚类时训练样本没有对应的输出，因此与有监督学习下的集成相比，弱学习器的结合更加困难，他们对不同的聚类结果进行配准，同时采用基于互信息权进行个体学习器的选择，提出基于 Bagging 的选择性集成聚类算法，实验表明该算法能有效改善聚类质量。周水庚和袁时金等人研究了层次化中文文档分类，使用 Boosting 算法为框架，最近邻 KNN 为基分类器，结果显示了层次化分类的效果要远远优于平面分类[19]。

目前集成学习在文本分类、图像处理、生物特征识别、疾病诊断、语音识别等众多领域都得到了广泛应用并且很成功，但是在推荐方面还很少有人尝试，本文尝试将集成学习与个性化推荐相结合，提高推荐的适应性和准确率。

1.2.2 个性化推荐现状

个性化推荐的研究在国外开展的比较早，自 1992 年协同过滤的概念被 David Goldberg 第一次提出后，个性化推荐就成为了研究学者和各大网站关注的热点。

1995 年，卡耐基.梅隆大学的 Robert Armstrong[20]等人将个性化推荐应用在人工智能领域，在美国人工智能协会上提出个性化导航系统 Web Watcher; 同年 8 月，麻省理工学院的 Henry Lieberman 在 IJCAI 上提出了个性化智能导航体 Litizia[21]，当用户浏览网页时 Litizia 能主动跟踪用户的行为，并试图根据用户点击的连接和当前位置预测用户感兴趣的商品。

1992 年， Riedl 和 Resnick 着手创建新闻推荐系统 GroupLens，它搜集用户对以前新闻的评分，自动预测用户感兴趣的新闻。GroupLens 系统用于帮助用户在大量的新闻中找到用户感兴趣的新闻，它成为协同过滤推荐引擎创立的标志，它的核心思想是用户以前喜欢的新闻未来也会感兴趣，通过收集用户对以前的新闻的打分，自动预测用户感兴趣的新闻，并且为了保护用户的隐私，用户打分时可以匿名。1997 年，GroupLens 研究设计小组推出第二个推荐系统 MovieLens[23]，该系统基于协同过滤技术，通过用户对电影的以往喜好信息预测对未评分电影的喜好。当一个新用户进入该系统时，会被要求对 15 部电影评分，评分范围是 1-5，系统根据用户以往评分预测其感兴趣的电影。

Amazom 是美国的最大的一家电子商务公司，初期它只经营书籍销售业务。主要是通过基于商品的协同过滤推荐算法[4]，通过用户的以往购买记录和浏览记录，向用户推荐他们之前购买的或浏览过但是没购买的相似的商品，用户可以看到推荐理由，一般都是你以前购买过相似的或相关的商品，用户可以对推荐结果做出反应如“我已经买过了”或“不感兴趣”，那么下次登录时系统就不会再推荐这个商品。当你购买一件商品时，系统会给出其他用户通常一起购买的商品，供你选择是否购买组合一般一起购买价格会相对优惠。另外当你购买了一件商品后 Amazon 还会给你推荐两个列表：一个是购买过此商品的用户经常购买的商品，另一个是浏览过此商品的用户经常购买的商品。Amazon 的这种销售方式非常成功，据透露 Amazon 有 20%~30%的销售都来自推荐。

google 实验室发布 Google Reader, 用于 RSS 推荐服务。Google Reader 向用户提供热门阅读和推荐阅读功能，热门阅读为整个网络关注的文章，推荐阅读根据用户的过往阅读历史和浏览历史进行推荐。此外 Google Reader 根据用户的喜好将文章进行排序，而不是简单地按时间排序。用户的喜好准则根据用户以往的“喜

欢”和“加星”操作生成。

目前个性化推荐也成功应用到了网络广告上。2007 年，雅虎推出了 SmartAds 个性化广告方案。雅虎通过用户的注册信息收集用户的性别、年龄、收入水平、生活方式等信息，再通过收集用户近期的搜索、浏览行为给用户提供个性化的广告横幅。据报道 SmartAds 使广告的点击率上升了三倍多。2009 年，ChoiceStream 公司为美国网上零售商 Overstock 制作个性化广告方案，该方案为用户推荐用户之前在网站上浏览的感兴趣的商品相关的广告， Overstock 声称：“广告的点击率是以前的两倍，伴随而来的销售增长也高达 20%至 30%。

推荐系统带来的巨大效益使国内很多公司也相继推出自己的推荐系统，但与国外相比，国内在个性化推荐领域的研究及应用，相对落后于国外。但随着互联网的普及和发展，我国用户对个性化推荐的需求也越来越强烈，各大网站也都看到了个性化推荐与经济效益的联系，这将会大大推动国内个性化推荐的发展。

2009 年 7 月，国内首个个性化推荐系统科研团队北京百分点信息科技有限公司成立，百分点的产品包括百分点个性化推荐引擎（BRE）和百分点智能分析引擎（BAE）和百分点个性化邮件引擎，这些产品旨在通过分析电子商务网站用户的兴趣偏好，为它们提供量身定做的营销方案，为电子商务网站吸引新的消费者并保留老的消费者，提升消费者购买的品类数和多样性，从而提高电子商务网站的核心竞争力。2011 年 9 月，百度世界大会 2011 上，李彦宏将推荐引擎与云计算、搜索引擎并列为未来互联网重要战略规划以及发展方向。百度新首页将逐步实现个性化，智能地推荐出用户喜欢的网站和经常使用的 APP。

表 1-1 是国内使用推荐技术的知名互联网公司，虽然目前国内很多公司都在应用个性化推荐技术，但效果并不是很满意，与国外的互联网公司相比，国内推荐系统的主要不足表现在：

(1) 缺乏个性化：大多数国内网站推荐的产品都是最新的和最热的产品，每个用户推荐的产品大都一样，没有做到真正的“以用户为中心”，用户甚至感觉不到推荐的存在。

(2) 推荐的准确率不高：目前国内已经有很多网站应用了推荐技术，但是普遍准确率不高，并没有将用户参与进去，有时推荐的甚至是用户已经购买过的商品。像美国的 Amazon 就做的很好，推荐的商品系统给出了推荐理由，并且用户可以选择“已经购买过”或者“没兴趣”，这样用户在下次登录时系统就不会推荐这些商品了。

表 1-1 国内主要网站的推荐技术

推荐系统名称	应用领域	应用技术
豆瓣电台	图书，电影，音乐	协同过滤
淘宝	商品零售	信息过滤+信息检索
大众点评	美食	基于内容
土豆	电影	热门电影+最新资源
China-pub	书籍	协同过滤
八宝盒	音乐	基于内容+协同过滤

(3) 推荐还不够智能：目前国内网站大多需要用户输入关键字或根据系统的类别标签逐步搜索用户想要的商品，这种方式并没有分析用户的兴趣爱好，没有将用户作为主要参与者。

1.3 本文的主要研究内容

由于电子商务的迅速发展，用户数目和商品数量急剧增加，推荐系统的准确率和可扩展性面临严峻考验，并且目前的推荐系统一般都是单个算法，考虑的提升推荐质量的因素比较少，而现实中影响推荐质量的因素是很多的。提高系统的准确率是人们永远的追求目标，本论文的主要目标就是提高推荐的准确率。由于集成学习具有很好的泛化能力，本文试图将其应用到推荐上改善推荐的准确率。

本文主要开展了以下几方面的研究内容：

(1) 对集成学习技术和个性化推荐算法进行了深入研究，分析了它们的优缺点，以及目前的应用领域。

(2) 针对传统的相似度计算方法在打分矩阵稀疏时会产生不正确的推荐结果，本文提出了 PCS 算法。该算法先使用决策树将用户进行分组，然后在组内进行个性化推荐，提出了一种新的寻找相似邻居的方法，同时避免了计算所有用户间的相似度，提升了在线系统的可扩展性。组内推荐时针对 Pearson 的缺点，从流行度、共同打分商品数、打分差异度对 Pearson 进行了改进。

(3) 现实中影响推荐系统准确率的因素是很多的，单一的推荐算法往往使得推荐的准确率很低，针对这个问题，本文提出了 SoftRankBoost 算法。SoftBoost 算法集成多个弱学习器解决统一问题，此外由于该算法是最大化软间隔算法，能够有

效避免难分数据的影响，因此具有非常好的泛化能力。本文使用 RankBoost 基于 pairwise 的思想重新定义了间隔与边界，将 SoftBoost 与 RankBoost 思想结合，使用协同过滤算法作为弱排序器，形成了最终的 SoftRankBoost 算法。

1.4 本文的组织结构

基于上一节的研究内容，给出本文相应的组织结构：

第一章： 绪论,简要说明了本课题的选题背景与研究价值，并对集成学习与个性化推荐的国内外现状做了大致描述。接着描述了本课题研究的主要内容，本章最后给出了本文的组织结构。

第二章：介绍了集成学习的概念和优点，介绍了集成学习两大主要算法 Bagging 和 Boosting,比较详细的介绍了 Boosting 的有关算法，最后给出了集成学习在最近几年的应用。

第三章：首先从整体上介绍了一下推荐系统，然后介绍了当前较为流行的个性化推荐技术，并分析了它们的优缺点。给出了评价推荐系统性能的评价指标。

第四章：针对协同过滤的数据稀疏问题和可扩展性问题，提出了一种新的协同过滤算法 PCS。该算法针对 Pearson 相似度存在的一些缺点，从流行度、共同打分商品数、打分差异度对 Pearson 进行了改进。此外由于计算所有用户间的相似度时间复杂度比较高，不利于在线系统的扩展，PCS 使用决策树将用户进行分组，然后在组内进行个性化推荐。

第五章：针对协同过滤技术考虑的提高推荐质量的因素比较单一，而现实中影响推荐的因素却比较多，集成学习技术使用多个弱学习器集成一个强学习器，具有很好的准确性和泛化性，将软间隔算法 SoftBoost 与个性化推荐结合，同时利用 RankBoost 基于对的思想，提出了 SoftRankBoost 算法，提升了推荐的准确率。

第六章：总结了本文的主要工作，并针对本文存在的不足对以后的改进方向提出了一些想法。

第二章 集成学习

本章简要介绍集成学习的基础知识，简述了集成学习的优点，并主要介绍了集成学习的 Boosting 算法，最后总结了一下集成学习在最近几年的一些应用。

2.1 集成学习概述

在机器学习中，提升分类准确率是人们一直追求的目标，因此研究者们对单一分类算法进行了不断的研究学习，并提出了很多改进方法，期望找到一个最有优的算法，但随着研究的一步步深入，算法的准确性已达到了一个瓶颈却仍不能达到满足，并且算法的准确性越高往往意味着算法越复杂越难以理解，时间复杂度越高。集成学习作为一种新的机器学习范式，使用多个弱学习器解决同一个问题，它具有很好的泛化能力[8-9]和稳定性，因此已经成功应用到了文本分类、图像处理、生物特征识别、疾病诊断[8]、Web 信息过滤、语音识别等领域。

自 20 世纪 90 年代，集成学习就引起了研究热潮，这主要归功于以下两个工作：第一个是 20 世纪 80 年代末 Hansen[24] 和 Salamon 的工作，他们使用多种方法训练神经网络以此来来提高分类的性能。在训练过程中，使用交叉验证调整神经网络的参数和结构，以此想获得最优的神经网络分类算法，他们进一步实验验证了将一组神经网络进行集成后的泛化误差远小于单一神经网络算法，集成后的性能甚至优于最好的单一神经网络；另一个归功于 Schapire 的理论研究，他通过构造性方法证明了弱学习算法与强学习算法之间存在着等价关系[9]，即一个准确率仅高于随机猜测的弱学习算法可以提升为一个强学习算法，并且这一理论证明有力的促成了 Boosting 算法的诞生，继而成为集成学习中最具影响力的算法之一。

1999 年，Schapire 提出了第一个可用的 Boosting 算法 AdaBoost，该算法不需要任何先验知识，并能自适应调整弱分类器，解决了 Boosting 算法早期的很多实际问题。Opitz 和 Maclin 将 Bagging 和 Boosting 算法进行了比较：实验发现在样本数据正常的情况下，Boosting 方法要优于 Bagging 方法，但是对于含有噪声数据的情况，Boosting 算法由于过分侧重于噪声数据而会产生过拟合反应，而 Bagging 算法对噪声数据不敏感，相对性能会更稳定一些。

Kuncheva 在 2002 年给出了几个简单的分类融合方法的分类误差：平均值，中

值，最大，最小，多数投票。实验发现在均匀分布下，最小化和最大化两种方法的效果最好，Kuncheva 进一步指出虽然分类器融合的方法没有分类器的多样性重要，但对于给定的一组分类器来说，一个好的融合方法才能获取尽可能多的信息。为了研究不同的分类器组合方式和多样性评价指标之间的关系，Ship 使用了 10 种不同的分类器组合方式和 10 种不同的多样性评价方法。但实验结果表明唯一的正相关性为多样性的 Double-Fault 评价和困难评价，这两种评价都显示了多数投票组合方式和 Naive-Bayes 组合方式之间的相关性。

Boosting 算法一般不会产生过拟合现象即使迭代次数很大的情况下，早先 Schapire 将这一现象归结为样本的间隔分布上，之后 Breiman 对这一说法提出了怀疑，认为 Boosting 算法不产生过拟合的原因是因为样本的间隔，而不是间隔分布，并提出了 arc-gv 算法，该算法虽然比 Adaboost 的间隔大，但泛化性能不如 AdaBoost。2006 年 Reyzin 和 Schapire 研究了 Breiman 的结果，发现 arc-gv 的泛化能力不强是因为 Breiman 用的弱学习器的复杂度比较高，他们认为，能够最大化间隔当然是好的，但是没有必要以其他因素（如弱分类器的复杂度）为代价。

2009 年 Leistner [14]研究了 On-line GradientBoost 算法的鲁棒性，提出了用于 On-line GradientBoost 的新损失函数，验证了使用该损失函数后算法的性能有了很大提升。2010 年 Tsai 和 Chen[20]等人提出将集成学习应用到 Rank 上，采用将基于点、基于对、基于列表三种方法相结合，试验中一般情况下基于点的比基于对的和基于列表的性能要好，但将三者结合后性能高于三者中的任何一个。

目前集成学习在文本分类、排序、图像处理、生物特征识别、疾病诊断、语音识别等很多领域都得到了应用并且很成功，这主要取决于集成学习本身的以下几个优点[25-26]：

(1) 泛化能力：泛化能力指机器学习算法对未知样本的适应能力，对应到试验中为机器学习算法应用于测试集中的样本时能将其正确归类的能力，提高泛化能力是机器学习永远的追求目标。集成学习并不是要寻找一个最优的算法而是使用多个准确率仅仅高于 0.5 的若学习器进行集成，实验表明其泛化能力远远优于单一的基学习器。

(2) 稳定性：对于单一的学习算法，它往往只能考虑某一方面的因素，而现实中影响性能的因素是很多的，因此单一算法往往会出现可能在这一数据集上准确率很高，在另一数据集上却很低。而集成学习相当于将同一个问题分解为多个子问题，使用多个不同的算法解决不同的子问题，然后将其结果进行集成提升算法的稳定性。

(3) 过拟合：过拟合是指获得的模型使训练样本的实际输出与模型输出基本一致，但测试样本的实际输出与模型输出却相差很远，即测试误差很高。造成这一结果的原因是往往太注重于将训练样本都分对，而训练样本中可能存在噪声数据，导致模型是错误的。集成学习将多个弱学习器的结果赋予权重组合在一起，避免了过拟合现象。实验表明经过多轮的迭代后 Boosting 不会产生过拟合现象，有时当训练误差为 0 时，其仍能继续降低测试误差。

2.2 集成学习两大算法

由于集成学习很好的泛化能力和稳定性，目前集成学习已经成功用到了很多领域，并且随着应用领域的不同以及相同领域不同的问题，研究者们已经提出了很多改进算法。但总体来说集成学习主要的算法为 Bagging 和 Boosting 两大算法族，本节简要介绍一下 Bagging 算法，重点介绍 Boosting 算法族。

2.2.1 Bagging 算法

1996 年 Breiman 提出了 Bagging[27]算法,Bagging 算法起源于 bootstrap 集成方法，使用 bootstrap 的有放回抽样方法获取多个样本集。由于使用有放回的抽样，因此一个样本可能会在样本集中出现多次，也可能一次都不出现。理论上来讲，每个样本集中大约包含 63.2%的原始样本集数据，因为每一个样本被抽到的概率为 $1-((N-1)/N)^N$，当 N 充分大时，该值接近接近于 1 – 1/e，大约为 0.632。Bagging 算法使用这些生成的样本集分别训练学习器，提高了模型的泛化能力，最终的输出结果是由训练得到的多个学习器的输出结果通过简单的平均或通过投票组合而成。Breiman 进一步指出理论和实验都表明对于不稳定的基学习算法(如决策树、神经网络等)，Bagging 能大大的提升预测准确度，而对于稳定的基学习算法，Bagging 算法反而会降低其准确度。上面的不稳定算法指的是如果训练集进行了稍微的改变，算法的输出结果就会有很大的不同，则为不稳定算法。同时由于 Bagging 采用有放回抽样，不侧重于任何特定的样本，因此各个样本被抽取的概率都一样，所以如果数据集中存在噪声数据，Bagging 也不太会产生过拟合现象。

Bagging 算法的伪代码如下图 2-1 所示：

$A\lg orithm: Bagging$

输入：初始训练样本集$S=\{(x_1,y_1),...,(x_n,y_n)\}$,迭代的最大次数$T$

输出：强分类模型

(1) $for\ t=1\ to\ T$

从原始样本集S中采用有放回随机抽样的方法获得样本集S'

使用基学习器在新样本集S'上训练，得到一个弱假设h_t

$endfor$

(2) 将各轮获得的弱假设$h_1,h_2,...,h_T$通过投票法集成，

生成强预测函数: $H(x)$

图 2-1 Bagging 伪代码

2.2.2 Boosting 算法

Boosting[9,28]：Boosting 集成算法的思想源于 Schapire 的理论研究，他通过构造性方法证明了弱学习算法与强学习算法之间有着等价关系[9](弱学习算法为准确率仅仅高于随机猜测的结果，强学习算法为准确率很高并能在多项式时间内收敛的算法)，这一理论证明不仅大大推动了集成学习的研究进展，也成为 Boosting 框架的雏形。起初 Boosting 算法主要用于分类问题，后来被成功扩展到回归问题上。Boosting 是一个框架算法，主要是通过对训练集的操作构造训练子集，然后使用弱学习算法训练得到一个预测函数，T 次迭代后得到T 个预测函数，最后将得到的这些函数系列加权组合作为最终的结果从而提高预测的准确率。

Boosting 与 Bagging 的不同之处是 Bagging 算法在对训练集的选择上各个训练集互不影响，因此训练机可以在模型训练前一起生成，然后并行生成一系列预测函数。而 Boosting 算法中各个训练集之间不是独立的，下一轮的训练集与前面各轮的训练结果有关，因此 Boosting 的各个预测函数不能并行生成，只能顺序产生。Boosting 算法比较侧重于分错的样本，因此在通常情况下其烦化能力很好，预测准确率很高，但当数据集中含有噪声样本时由于其过分侧重噪声样本，可能会导致过拟合现象。而 Bagging 是等概率抽样，比较稳定，相对来说不会产生过拟合。

2.2.2.1 AdaBoost 系列代表算法

Boosting 算法要求事先知道弱学习算法正确分类的下限，这在实际中很难做到。1995 年 Freund 和 Schapire 提出了 AdaBoost 算法，该算法自适应的调整弱分类器，并且不需要任何的先验知识，解决了早期 Boosting 算法的很多实际问题，

成为 Boosting 中的代表算法。它的主要思想是在样本集上维护一个样本权值分布，初始化时所有样本均匀分布，在每一轮迭代中，当使用一个基学习器得到弱假设后，计算该弱假设的错误率，使用得到的错误率更新权值分布，将分错的样本的权值增加，分对的样本的权值降低，侧重于训练上一轮分错的样本，经过T 次迭代，使用弱分类算法生成T 个学习器，最后将生成的多个学习器以加权组合的方式进行集成，其中每个分类器的权值由它的分类准确率决定，分类准确率高的分类器权值相对就高。具体的算法流程如图 2-2：

$Algorithm: AdaBoost$

输入：样本集$S=\{(x_1,y_1),(x_2,y_2),...,(x_n,y_n)\}$，迭代次数$T$

初始化：样本权值分布$D^1(i)=\frac{1}{n}, i=1,2,...,n$

(1) $for\ t=1\ to\ T$

使用基分类器在样本分布D^t上训练得到若假设h_t

基于D^t分布计算弱假设h_t的错误率ε_t

$if\ \varepsilon_t > 1/2$，则跳出循环

计算弱假设h_t的权重$\alpha_t=\frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$

更新权重$D^{t+1}(i)=\frac{D^t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

$endfor$

(2)输出强分类器：$H(x)=sign(\sum_t \alpha_t h_t(x))$

图 2-2 AdaBoost 算法伪代码

最初的 AdaBoost 算法只能解决二分类的问题，但现实生活中经常会碰到多分类的问题，Schapire 提出了 AdaBoost.M1 和 Adaboost.M2 解决多分类问题。其中 AdaBoost.M1 是 AdaBoost 最直接的扩展，因此理解起来非常简单，但该算法要求弱假设在经过迭代产生的任何困难样本分布上的准确率也要大于 1/2,如果达不到该准确率那么该算法将会失效，看起来这一要求并不难达到因为比随机猜测的准确率就为 1/2 了，但事实上是随机测测的准确率等于 1/2 的情况只是在二分类的情况下，而对于多分类比如 K 分类的情况下，随机猜测的准确率为$1/K$,并非 1/2，因此再多分类情况下要求弱学习器准确率高于 1/2 并不是简单的事。而 Adaboost.M2 算法将多分类问题看成多个二分类问题进行解决，如有 k 个类别标签{1,2…,k}，假设学习器 h 取值为{0,1}，对于一个样本 x_i 判断其属于哪个类别可采用如下方式解决问题：首先判断样本 x_i 是属于类别 1 还是不属于类别 1，若

$h(x_i,1)=1$，则认为该样本的类别为 1，若 $h(x_i,1)=0$ 则认为该样本属于除类别 1 以外的其他类别，然后继续以上过程。

Friedman 提出的 Gradient Boosting 用于解决回归和分类问题，该算法通过梯度下降算法（最速下降法）最小化损失函数来实现。损失函数 $L(y,F(x))$ 表示的是模型的不准确程度，损失函数值越小表示该模型性能越好，因此最终的目标是最小化损失函数。在每次建立新模型前，都要使用以往得到的模型计算样本的梯度，对于一个样本来说理想的梯度是 0，离 0 越远表示误差越大，该样本越应该得到关注，因此 Gradient Boosting 的思想是让损失函数沿着梯度的反方向移动，最快速度的减小损失函数的值。Gradient Boosting 具有很好的健壮性和可解释性，经常被用以解决 Learning to Rank 问题。图 2-3 给出该算法的伪代码：

Algorithm: *Gradient Boosting*

输入：训练集 $S=\{(x_1,y_1),...,(x_n,y_n)\}$, 损失函数 $L(y,F(x))$, 迭代次数 T

初始化：$F_0(x)=\arg\min_{\gamma}\sum_{i=1}^{n}L(y_i,\gamma)$

(1) *for* $t=1$ *to* T

计算反梯度 $r_{i,m}=-\left[\frac{\partial L(y_i,F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}, i=1,...,n$

得到一个与反梯度方向最接近的若假设 $h_t(x)$, 训练集是 $\{(x_i,r_{i,m})\}_{i=1}^{n}$

计算 $\gamma_t=\arg\min_{\gamma}\sum_{i=1}^{n}L(y_i,F_{m-1}(x_i))+\gamma h_t(x_i)$

更新模型 $F_t(x)=F_{t-1}(x)+\gamma_t(x_i)$

endfor

(2) 输出模型：$F_T(x)$

图 2-3 Gradient Boosting 伪代码

2.2.2.2 基于间隔的 Boosting 算法

FVeund 和 Schapire[28,29]使用训练误差,训练样本集的规模，弱假设的 VC 维和迭代次数给出了 AdaBoost 算法泛化误差的上界，该上界表明随着迭代次数变得很大时，将会出现过拟合现象。但有不少研究者[30-32]都通过实验发现即使在上千次的迭代后，AdaBoost 也不会出现过拟合现象，不仅如此他们还发现当训练误差变为 0 后，泛化误差有时候还会持续降低，这一发现有力的反驳了 FVeund 和 Schapire 提出的泛化误差上界。随后 Schapire[33]使用间隔(margins）假说对此给出了解释，并给出了泛化误差新的上界，该上界表明 AdaBoost 的上述现象与样本集的间隔分布有关，样本集的规模和弱假设的 VC 维有关，而与迭代次数没有任何关联。在样本数目和弱假设的 VC 维一定的情况下，拥有 margins 值较大的样本个数越多，

AdaBoost 的泛化能力就越好。

然而 Breiman[34]等人提出了对间隔分布假说的疑问，他们给出了泛化误差一个更加严格的界限，认为泛化误差的好坏并不是与样本集的间隔分布有关，而是与样本集的最小间隔有关（样本集的最小间隔为样本集中所有样本间隔的最小值，也称为样本集的间隔）。 与此同时 Breiman 提出了 Arc-GV 算法，该算法能得到近似最大化间隔。但该算法的泛化能力却还不如 AdaBoost 算法。Breiman 的最大化间隔理论大大的的刺激了最大化间隔 Boosting 算法的发展，如 LPBoost, AdaBoost*, TotalBoost 等都属于最大化间隔算法。

AdaBoost*是第一个能证明最大间隔化的算法，且该算法和 TotalBoost 能够在 $2\ln(N/\delta^2)$ 迭代后收敛到最大化间隔，这里 δ 是一个准确率参数。 LPBoost 算法使用组合数学中的最优化问题直接求最大化间隔，同时利用对偶性定理将求最大化间隔(margin)问题转变为最小边界(edge)。TotalBoost 算法与原先的 Boosting 算法更新样本权重的方法不同，原先的算法在每次获得弱假设后，使用本轮的弱假设更新样本权重，而 TotalBoost 在更新样本权重时，涉及到了训练得到的所有弱假设，TotalBoost 算法认为必须要把所有的弱假设都考虑进来，才能获得更准确的样本权重，该算法使用最小化相对熵的方法更新用于下一次迭代的样本权值，相对熵的定义如下：

$$\Delta(D, D^1) = D(n)\sum_n \ln \frac{D(n)}{D^1(n)} \tag{2-1}$$

上式中 D^1 是样本的初始权值分布，为均匀分布，使用初始分布计算相对熵是为了保证样本权值分布尽可能的接近初始分布。

对于不含噪声的正常数据集来说，LPBoost 算法的性能要优于 AdaBoost,甚至可以说 LPBoost 是完美的，因为 LPBoost 追求的是样本集的最大的间隔(margin), 而样本集的间隔越大就意味着该算法的泛化能力越强，如图 2-4 所示。但是当数据集中含有噪声数据或难分数据时，LPBoost 可能会出现过拟合现象，如图 2-5 所示，图中仅包含一个难分样本点，但却在很大程度上影响了最大间隔，原因是 LPBoost 力求将所有样本点都考虑进去，在含有噪声数据的情况下就会特别重视这些难分样本。而正因为 LPBoost 太过于重视这些样本点，反而会造成过拟合，影响其泛化能力。

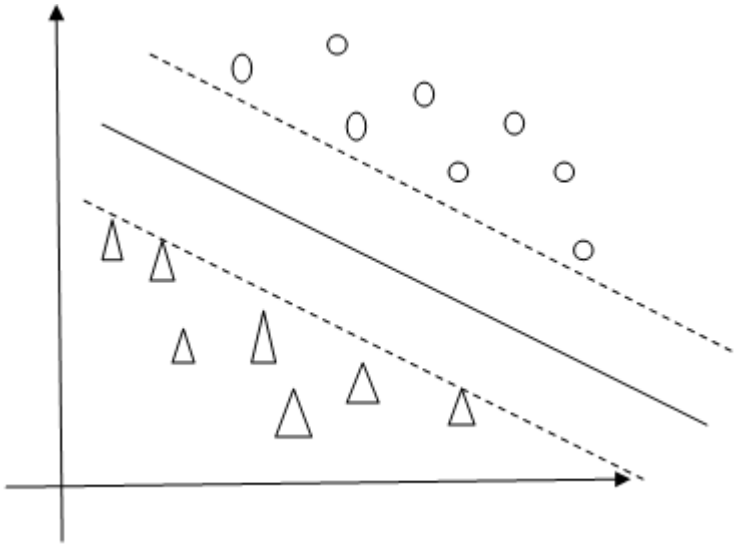

图 2-4 正常数据集

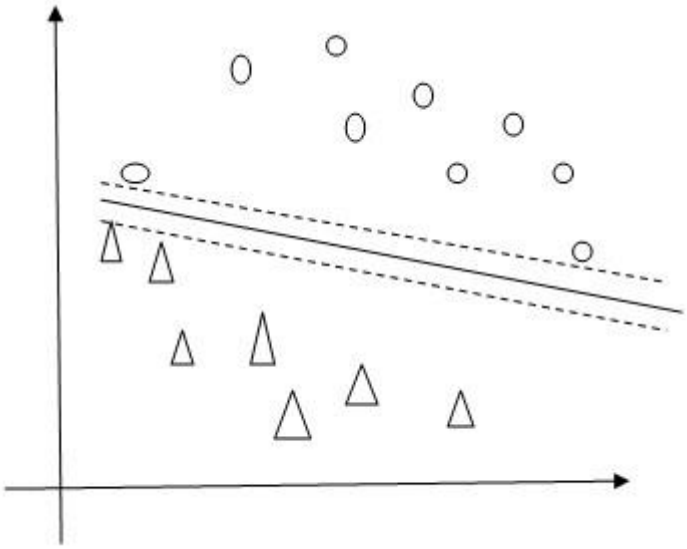

图 2-5 含有难分样本点

上述问题的存在，导致了一些列软间隔算法的产生如 SoftBoost, ERLPBoost SmoothBoost, MadaBoost。SmoothBoost 和 MadaBoost 也属于最大化软间隔算法，但是这两种算法何时能收敛到最大间隔却是未知的。此外 MadaBoost 算法要求弱假设的边界(edge)为单调递减，这在实际中很难满足。SoftBoost 是第一个能够在实际中应用的最大化软间隔算法，与最大化硬间隔 LPBoost 算法很相似，都是采用直接最大化样本集间隔的方法，不同的是 SoftBoost 引入了一个松弛变量 ξ_n 来放松对一些样本的要求，允许一些样本在间隔的另一面，即允许一些样本被分错。

另一方面与 LPBoost 不同的是在更新样本的权值分布上，SoftBoost 采用与

TotalBoost 相似的方法更新样本权重，即也是采用最小化相对熵的方法。ERLPBoost 是对 LPBoost 算法的改进，也是采用引入松弛变量ξ_n放松对一些样本的要求，另外加入了一个正则因子η，改变 SoftBoost 在前面几次迭代过程中出现的泛化误差下降缓慢地问题。

2.3 集成学习的应用

由于集成学习能够集成多个弱学习器解决同一个问题，具有很好的泛化性和稳定性，因此已经成功应用到了很多领域如文本分类、图像处理、生物特征识别、疾病诊断[8]、Web 信息过滤、语音识别等领域。最近几年集成学习变得更加热门，经常被应用到排序问题中，在 KDD-CUP, Netflix，Learning to Rank 比赛中集成学习已经成为常用技术。

2003 年 F1reund 等人提出 RankBoost 算法，将 Boosting 算法应用到排序问题上，它是目前基于 pairwise 的最好的排序算法。该算法与其它 Boosting 算法相同，都是结合多个弱假设给出的排序结果，从而提高排序结果的准确度。该算法使用样本对进行输入，通过样本对间的相对排序确定最终的排序结果。

2010 年 3 月至 5 月 yahoo 公司发起了一场名为 Learning to Rank 的挑战赛，并公布了两个从未面世的真实数据集，其中第一个数据集包括 29921 个查询，744692 个文档，519 个特征。第二个数据集包括 6330 个查询，172870 个文档，596 个特征。比赛中发现获奖的顶级选手大多采用的都是集成方法，包括 Boosting, Bagging 和 Random Forests。如获得第一名的团队 Burges 等人使用了 12 个不同的排序模型进行线性组合，其中有 8 个使用基于 Boosted Tree 的 LambdaMART 的集成模型，有两个是基于神经网络的 LambdaRank 模型，另两个使用的是基于 logistic 回归损失的 Mart 模型。Pavlov 等人提出 BagBoo 算法，他们将 Bagging 算法和 Boosting 算法相结合，由于 Bagging 算法可以并行执行，因此算法的执行效率比较高，而且使用 bagging 算法可以避免过拟合现象。与此同时他们使用 Gradient Boosting 算法减小偏差，获得较好的准确率。Busa-Fekete 和 Szarva 等人主要使用 pointwise 方法，但在最后集成的时候使用 listwise 方法进行集成，他们的方法主要有 4 部分组成：对 querywise 做预处理，多分类算法 AdaBoost.MH，回归校准以及用于集成的指数加权。

由 ACM 的数据挖掘及知识发现专委会（SIGKDD）主办的历届 KDD-CUP 大赛中，也有不少团队使用的是集成学习算法。如 2009 年的 KDD-CUP（预测用户

关系）中，Jianjun Xi 和 Viktoria 等人将 Boosting 与 Bagging 算法相结合，使用基于梯度的 Boosting Tree 作为主要的分类器，为了进一步提升算法的健壮性和准确度，他们又使用了 Bagging 算法将一系列的 Boosted Tree 进行组合。Hung-Yi L 和 Kai-Wei Chang 等人将线性模型，异质 Boosting 算法和 Bayes 算法相结合。他们提出了一个能够处理多种属性和缺失值的异质基分类器，然后使用 AdaBoost 进行提升，最后使用 Selective Naive Bayes 分类器将种类特征和离散属性分类。

在线电影租赁商 Netflix 与 2006 年 10 月发起了一场 Netflix 比赛，Netflix 称任何组织或个人只要能够将它现有的推荐算法 Cinematch 的准确率提升 10%，就能得到一百万美元的奖金。终于 20009 年的 Netflix 大赛，前两名的团队都使用了集成学习技术。The Ensemble 团队由于在提交最终结果时比 Bellkor's Pragmatic Chaos 团队晚了二十分钟，而位居第二，与 Netflix 的百万大奖无缘。Bellkor's Pragmatic Chaos 算法集成了 BellKor，Pragmatic Theory 和 BigChaos 三个团队的算法，且每个团队的算法又集成了多个算法，如 BellKor 就是线性组合了 100 多个基算法的结果。The Ensemble 团队主要使用了 KNN 近邻，矩阵分解与降维，Boltzmann 机和 RBM，基于时间的模型作为基分类器，然后使用 Bagging 集成框架将这些算法进行融合。

2.4 本章小结

本章首先对集成学习整体进行了一个概述，然后讲述了集成学习中两大主要算法框架。简要介绍了一下 Bagging 算法，较详细的介绍了 Boosting 系列的相关算法，其中包括 AdaBoost 相关系列算法和基于最大化间隔（硬间隔和软间隔）算法，为第五章将集成学习应用到个性化领域做铺垫。最后讲述了集成学习技术在最近几年的应用。

第三章 个性化推荐

电子商务的迅猛发展使得个性化推荐变得越来越重要，对于各大电子商务网站来说，有一个好的推荐系统留住顾客和吸引顾客，是其生存的关键。本章简要介绍了推荐系统的思想，并介绍了主流的个性化推荐技术，分析了一下各自的优缺点。另外介绍了评价推荐系统性能的评价指标。

3.1 推荐系统概述

以往的电子商务网站服务的模式是“网站提供什么，用户就接受什么”，网站提供的信息对所有用户都是相同的，用户只能自己慢慢寻找感兴趣的商品，然而随着电子商务的迅速发展，商品数量和种类呈指数增加，用户面对海量的商品往往在找到感兴趣的商品之前已经非常沮丧、烦躁，对购买商品非常失望，继而就会放弃此次购物。个性化推荐彻底改变了这一传统购物模式，将用户从被动的浏览者变成了主动的参与者[34]，主动搜集用户的过往信息如浏览信息、购买信息等，继而分析用户的兴趣、品位及购买倾向，最终给目标用户提供其感兴趣的商品。

一般来说，一个好的个性化推荐系统应该有以下几方面作用：

(1) 提高用户的忠诚度：当用户在网站上购买商品时，推荐系统给用户提供其感兴趣的商品列表，用户能很轻松愉快的找到自己想要的商品，这无疑会使用户下次继续再次购物。

(2) 吸引新用户：用户第一次在某网站购物时，有可能只是偶然原因或者听别人介绍，好的推荐系统会让用户有一个满意的用户体验，满意的用户体验无疑会吸引一批新用户。

(3) 推荐潜在感兴趣商品，促进购买欲：用户有时只是随便浏览下商品，推荐系统能够提供给其潜在有兴趣的商品，促使用户购买商品，提升销售额。

(4) 搭配销售，提升销售额：当用户要购买某一商品时，推荐系统提供配套的商品进行组合销售，往往组合销售的话会有一定的优惠，这也会刺激用户一起进行购买。

(5) 帮助用户提高检索效率：现在网站上都设有购物导航服务，如果用户不知道具体输入什么进行搜索的时候，可以根据导航服务一步步查找要买的东西。

一个完整的推荐系统一般由三个部分组成[35]：收集用户过往行为信息模块、分析用户喜好品味模块及推荐算法模块。行为信息模块负责收集记录用户的信息行为，例如搜集用户对某一商品的打分、回答、购买、浏览、购物车等行为。其中打分和评价的信息属于显式反馈，相对来说很好获得，而购买、浏览等行为很难以某种显示形式呈现，只能通过 web 挖掘[36-37] (web mining)技术来获得,如使用关联规则、决策树等[36-37]数据挖掘方法从用户的点击、购买及购物车行为中挖掘用户偏好信息；爱好分析模块对上步的用户行为信息进行分析，建立合适的模型描述出用户的爱好、兴趣与个人品味；推荐算法模块是整个个性化推荐系统中的核心，推荐算法模块利用特定的推荐算法，从海量的商品集中筛选出用户有潜在购买兴趣的商品推荐给用户。

3.2 个性化推荐技术

推荐算法是整个推荐系统的核心部分，自 20 世纪 90 年代中期协同过滤概念第一次被提出以及最初的几篇协同过滤[22]文章被发表，关于个性化推荐的研究及成果便不断涌现。不少研究者从不同的角度对推荐系统进行了不同分类[38-40]，但最普遍的分类方式是根据推荐算法的不同分为：基于内容的推荐、协同过滤推荐以及混合推荐系统。

3.2.1 协同过滤推荐技术

基于协同过滤算法的推荐系统是目前应用最广泛和成功的推荐系统，它通过搜集其他用户的偏好或品味信息来自动预测某目标用户对特定商品的偏好程度。核心思想大致分为两部分：首先基于以往的用户过往历史行为信息计算用户两两之间的相似度，然后找出与目标用户相似度较高的前 N 个用户，利用这些邻居用户预测目标用户对特定商品的打分。该算法基于的假设是：若两个用户相似，则认为某一用户喜欢的商品另一个也会喜欢。

用户 u 对商品 i 的预测打分 $r_{u,i}$ 的函数形式如公式(3-1)：

$$
\begin{aligned}
&(a)\quad r_{u,i} = \frac{1}{N}\sum_{u'\in\hat{U}} r_{u',i} \\
&(b)\quad r_{u,i} = k\sum_{u'\in\hat{U}} sim(u,u')\times r_{u',i} \\
&(c)\quad r_{u,i} = \bar{r}_u + k\sum_{u'\in\hat{U}} sim(u,u')\times (r_{u',i} - \bar{r}_{u'})
\end{aligned}
\tag{3-1}
$$

其中$\hat{U}$表示与用户u购买了公共商品的用户集合中与其相似度最高的前N个用户的集合，$\overline{r}_u$与$\overline{r}_{u'}$分别表示用户u与用户u'的打分均值,K为标准化因子，通常记为$k=1/\sum_{u'\in\hat{U}} sim(u,u')$。在公式(3-1)中，方法(a)直接使用近似邻居对特定商品打分的平均值，是最简单的计算方法，方法(b)是常用和普遍的方法，使用用户间的相似度作为权重，若用户u与用户u'的相似度越高，则计算$r_{u,i}$时，用户u'所占的分量就越重。方法(c)进一步考虑了每个用户的打分习惯或打分尺度不太一样，克服了评判尺度不一致的缺点。

$sim(u,u')$为用户u与用户u'的相似度，协同过滤系统中用于计算两个用户间的相似度有很多种[41-43]。大多数的相似度计算方法是根据打分矩阵计算相似度，常用的相似度计算方法是余弦相似度和 Pearson 相似度，具体的计算公式如下：

(1) 余弦相似度：将用户对商品的打分看做向量，使用夹角余弦计算，其中$\boldsymbol{r}_u$与$\boldsymbol{r}_v$是用户u与用户v对商品的打分向量：

$$sim(u,v)=\cos(u,v)=\frac{\boldsymbol{r}_u \bullet \boldsymbol{r}_v}{\|\boldsymbol{r}_u\| * \|\boldsymbol{r}_v\|} \tag{3-2}$$

(2) Pearson 相似度：更加侧重于用户间共同评分过的商品，其中I_{uv}为用户u与用户v共同打分的商品集：

$$sim(u,v)=\frac{\sum_{i\in I_{uv}}(r_{u,i}-\overline{r}_u)(r_{v,i}-\overline{r}_v)}{\sqrt{\sum_{i\in I_{uv}}(r_{u,i}-\overline{r}_u)^2}\sqrt{\sum_{i\in I_{uv}}(r_{v,i}-\overline{r}_v)^2}} \tag{3-3}$$

(3) 修正的余弦相似度：上面的余弦相似度未考虑用户评分尺度问题，但是在基于评分的相似度计算中，消除用户对商品的评分倾向是很有必要的，否则相似度就变得意义不大了。根据最近的一篇报告显示在一些评分系统中，若能适当的将用户的评分倾向考虑进来，那么用户对未知商品预测评分的准确率将远远高于[44]基于评分的简单相似度计算方法。为解决上述问题，提出了修正的余弦相似度，其中I_{uv}为用户u与用户v共同打分的商品集，I_u和I_v表示用户u与用户v各自打分的商品集：

$$sim(u,v)=\frac{\sum_{i\in I_{uv}}(r_{u,i}-\overline{r}_u)(r_{v,i}-\overline{r}_v)}{\sqrt{\sum_{i\in I_u}(r_{u,i}-\overline{r}_u)^2}\sqrt{\sum_{i\in I_v}(r_{v,i}-\overline{r}_v)^2}} \tag{3-4}$$

由于用户的兴趣与爱好是会随着时间改变的（比如一个学生开始喜欢看科幻片、悬疑片，过一段时间她可能喜欢看韩剧、古装剧），并且为了避免在推荐时进行大量的计算影响系统的效率，普遍采用提前计算好两两用户之间的相似度，并且停一段时间进行一次更新以便适应用户的兴趣变化。

除了上面的基于用户相似度的协同过滤算法外， 2001 年 Sarwar[45]提出了基于商品的协同过滤。基于商品的协同过滤先计算已评分商品与待评分商品的相似度，然后把相似度作为权值，利用已评分商品与相似度权值预测用户对待评分商品的喜好程度。基于商品的协同过滤基于这样一种假设“能够引起用户感兴趣的商品，必定与其之前评分高的项目相似”。计算商品相似度的方法跟基于用户的相似度方法大体一致，如公式(3-5)所示为使用商品相似度计算为评分商品的预测评分。一般而言，用户的兴趣和爱好随着时间会逐渐变化，两者相比商品间的相似度比用户间的相似度会稳定的多，采用基于商品的相似度可以离线计算然后定期的进行更新，不仅提高了推荐效率而且不会影响推荐的质量；再者当数据量较小时比基于用户的相似度算法更准确。在实际的推荐系统中使用基于商品的 CF 算法可以做到当用户增加或更改购买过的商品信息时，相似度得到及时更新[46-47]。2004 年 Deshpande 和 Karypis[47]进一步推广了基于商品相似性的算法，他们提出在推荐时只取相似度最高的前 N 个商品，实验表明这种方法比传统的基于用户的方法快 1-2 个数量级，而且这种方法产生的推荐质量更好。

$$r_{u,i}=\frac{\sum_{j\in I_u} sim(i,j)r_{u,j}}{\sum_{j\in I_u} sim(i,j)} \tag{3-5}$$

基于用户和基于商品的协同过滤算法统称为基于记忆的协同过滤算法，它们可解释性强，且无需考虑被推荐商品的属性信息，但当打分矩阵极度稀疏时推荐质量会下降。以模型为基础的协同过滤(Model-based Collaborative Filtering)算法是先用历史信息得到一个模型，再用此模型给出推荐结果。以模型为基础的协同过滤广泛使用的技术包括有 Bayes 模型、基于聚类[48]的模型、矩阵分解等等，根据对样本的分析得到模型。基于模型的协同过滤与基于记忆的相比具有的优势是能处理稀疏矩阵，对于数据集很大并且稀疏的打分矩阵能够提高推荐系统的性能。它的缺点是模型的建立需要花费的代价很大，如需要大量的时间和空间。下面介绍一下几种矩阵分解算法：

(1) 奇异值分解 SVD[49]:SVD 是将一个 m×n 的实数矩阵或复数矩阵 M 分解

成如公式(3-6)所示的形式，分解后三个矩阵所需的存储空间远小于原始的矩阵 A，节省了存储空间，并且有助于做降维处理。

$$M \approx U\Sigma V^T \tag{3-6}$$

其中 U 和 V 矩阵分别是 m×m 和 n×n 实数或复数酉矩阵，Σ 是 m×n 的对角矩阵，其组成元素是 M 的奇异值，常见的做法是将奇异值由大到小排列，这样 Σ 便唯一确定了。SVD 因子分解方法被证明在基于打分的评价指标方面(如 RMSE 评价指标)比传统的基于相似邻居的方法更有效，但将其应用在 TopN 上时推荐效果并不好。这也表明不同的算法分别适用于不同的场合下，而不能以偏概全说哪个算法比哪个算法好，哪个算法不如哪个算法。

(2) RMF[50]算法：矩阵因子化方法假定每个用户的偏好特征可以用若干个比较少的因子进行描述，同样假定商品的特征属性可以用相同维数的因子描述，当某一用户的偏好特征与商品的属性特征相符合时，表明该商品是用户感兴趣的商品。使用 $m\times r$ 的矩阵 W 表示用户的特征矩阵，$r\times n$ 的矩阵 H 代表商品的属性矩阵，R 表示用户-商品打分矩阵，计算方式如公式(3-7)：

$$\mathrm{R}_{ij} \approx \sum_{l=1}^{r} W_{il} H_{lj} \tag{3-7}$$

为了降低模型的过度拟合，找到最接近 R 的 W、H 矩阵，可以通过最小化如下目标函数 $F(W,H)$ 获得：

$$min \quad F(W,H) = \frac{\lambda}{2}(\|\mathrm{W}\|_F^2 + \|H\|_F^2) + \sum_{ij\in S}(R_{ij} - W_i^T H_j)^2 \tag{3-8}$$

其中 $W \in R^{m\times r}$，$H \in R^{n\times r}$，λ 是正规化参数，其值大于 0，并且 $S = \{i, j \mid R_{i,j} > 0\}$。 $\|\bullet\|_F$ 代表 Frobenius 范数，如对于矩阵 W， $\|\mathrm{W}\|_F^2 = \sum_i \sum_j \mathrm{W}_{i,j}^2$ 。

上述的最优化问题可以通过最小二乘法或梯度下降法解决，但由于梯度下降法又叫最速下降法，可以在较短的时间内获得相对较高的准确率，因此选用梯度下降法最小化上述问题。关于公式(3-8)的一阶偏导数如下：

$$\frac{\partial F}{\partial w_{il}} = \lambda w_{il} - 2\sum_{j|ij\in S}(R_{ij} - W_i^T H_j)H_{jl} \tag{3-9}$$

$$\frac{\partial \mathrm{F}}{\partial H_{jl}} = \lambda H_{jl} - 2\sum_{i|ij\in S}(R_{ij} - W_i^T H_j)W_{il} \tag{3-10}$$

求解公式(3-8)时，需要初始化矩阵 W 和 H,如从均匀分布中随机抽取出一些值进行填充，然后使用梯度下降法迭代的更新矩阵 W 和 H，$W^{(t)}$ 表示矩阵 W 经过 t 次迭

代后的形式，相应的 $H^{(t)}$ 是矩阵 H 迭代 t 次的形式，τ 表示恒定步长。公式(3-11)和公式(3-12)是矩阵 W 和 H 的迭代更新过程：

$$W_{il}^{(t+1)} = W_{il}^{(t)} - \tau \frac{\partial F}{\partial W_{il}^{(t)}} \tag{3-11}$$

$$H_{jl}^{(t+1)} = H_{jl}^{(t)} - \tau \frac{\partial F}{\partial H_{jl}^{(t)}} \tag{3-12}$$

(3) NMF[50]算法：通常的矩阵因子化算法如 SVD、PCA 等都可能会把矩阵分解为含有负数的矩阵，但负值对于现实世界中的问题是没有什么意义的，如图像数据中像素点的值不可能为负、在电影推荐中负值往往也是无法解释的，NMF 是一种新的矩阵分解算法，与以往矩阵分解方法不同的是它对分解矩阵增加了约束条件即只能将矩阵分解为两个非负矩阵的乘积。

给定矩阵R,通过非负矩阵分解将R分解为两个非负矩阵W,H,其中R为 $m \times n$ 矩阵，W 为 $m \times r$ 矩阵，H 为 $r \times n$ 矩阵，其中 r 的取值一般满足 $(m+n)r < mn$，将原始的矩阵压缩为矩阵 W 和 H。计算方法如公式(3-13)：

$$R \approx WH \tag{3-13}$$

在得到两个非负矩阵 W 和 H 后，需要衡量一下 R 与 WH 的差异度，使用公式(3-14)进行衡量，该公式的最小值为 0，表示 R 与 WH 完全相同：

$$D(R \| WH) = \sum_{i=1}^{m} \sum_{j=1}^{n} (R_{ij} \log \frac{R_{ij}}{(WH)_{ij}} - R_{ij} + (WH)_{ij}) \tag{3-14}$$

为了获得好的效果，上述函数的目标是最小化 $D(R \| WH)$ 的值，同时满足约束条件 $W_{il}, H_{jl} \geq 0$，这里 $i = 1,...,m; j = 1,...,n; l = 1,...,r$。求解 W 和 H 时，先随机初始化这两个矩阵，然后使用公式(3-15)迭代更新 W 和 H 两个矩阵：

$$\begin{aligned} W_{ia} &\leftarrow W_{ia} \sum_{u} \frac{R_{i,u}}{(WH)_{i,u}} H_{a,u} \\ W_{i,a} &\leftarrow \frac{W_{i,a}}{\sum_{j} W_{j,a}} \\ H_{a,u} &\leftarrow H_{a\,u} \sum_{i} W_{i\,a} \frac{R_{i,u}}{(WH)_{i,u}} \end{aligned} \tag{3-15}$$

在本文的第五章实验部分使用了该算法作为对比算法，在实验中可以改变 r 的值，即改变矩阵 W 和 H 的秩，当 r 较小时能节省存储空间，当 r 越高时 WH 的结果越接近于 R。由于 NMF 的高准确率和快速性，目前 NMF 已经应用到了很多领域，

如对图像的分析和处理、文本挖掘、语音处理等。值得注意的是，上述的 NMF 算法不是收敛的算法，因此我们通过最优化方法求解最后得到的可能不是全局最优值。为减轻这一问题，通常的做法是使用多个不同的初始值运行这一算法，最终选择一个能最小化损失函数的值。

目前关于相似度的计算问题已经产生了许多改进算法并且已经被成功应用了，例如缺席投票方法是关于基于记忆的协同过滤算法的扩展与改进，如果用户评分的商品数比较少，那么两个用户共同打分的商品集就会很少，相似度计算就会不准确，推荐的质量就会严重下降。实验表明使用用户的打分均值或一个默认值[51]填充打分矩阵，可以缓解数据稀疏问题，提升推荐质量。

协同过滤技术需要解决的问题是：

(1) 数据的稀疏性和冷启动：目前商业性的推荐系统往往是基于大数据集的，但是用户对商品的评分往往很少，例如大家熟知的 MovieLens 数据集 95%都是未知的，而 Netflix 数据集 98%的打分是未知的，并且随着用户和商品的数目增多，打分矩阵变得非常大且越来越稀疏。这种情况导致的典型问题就是冷启动问题，因为协同过滤算法向用户推荐商品是基于用户的历史打分信息，用户必须给足够的商品打过分，系统才能正确的找到其相似邻居然后给出可信的推荐，但是新用户没有对任何一个商品打过分，不能找到它的相似邻居也就不能给出推荐；同理新商品也存在这个问题，当它们被添加到系统中时，必须被大量的用户评分后才能被系统推荐出去。数据的稀疏性问题和冷启动是相关的，当用户对商品的打分很少时，通过相似度得到的最近邻居是不准确的，因此系统也不可能给出好质量的推荐。

(2) 可扩展性：电子商务发展非常迅速，随着用户数目和商品数量的急剧增长，传统协同过滤算法的可扩展性面临严峻考验。例如对于上亿的用户和上百万的商品，即使对时间复杂度为线性($O(n)$)的协同过滤算法也是要花费很长时间的。因为商品的相似度比较稳定，因此有些推荐系统采用商品相似度推荐算法来节约时间，这种方法在商品相对稳定的系统中还是很有效的。此外很多系统都要求对用户的在线请求做出及时的响应并给出相应的建议，而不必考虑他们的购买和打分记录，这对可扩展性提出了更高的要求。

(3) 同义性：同义性问题指一些相似的商品甚至是相同的商品可能会有好几个不同的名称，如“计算机”和“电脑”本来是相同的东西，但是大多数的协同过滤算法并不能发现它们的潜在关系，所以便会区别对待它们，这就会影响协同过滤系统的推荐质量。

3.2.2 基于内容的推荐

基于内容的推荐技术是信息过滤和信息检索技术的延续与发展，它使用商品属性信息（内容信息）和用户偏好信息的相关性给用户推荐相应的商品。它不同于协同过滤的基于有相似偏好的相似邻居给目标用户推荐商品。基于内容的核心思想是：分别为每个用户和商品建立其对应的描述信息，通过分析用户购买或浏览的历史信息，构造用户的配置信息。然后提取出商品的属性信息，并通过计算与目标用户偏好的相似度向用户推荐与其配置信息最相似的商品。当用户偏好和商品属性的描述已知时，推荐问题就变成了信息检索问题。

在信息检索中经常使用向量空间模型[52](Vector Space Model, VSM)将文档以向量的形式表示。设文档总数为 N，文档集合为 $D=\{d_1,d_2,...,d_N\}$,所有文档中包含的关键词的集合为 $T=\{t_1,t_2,...,t_k\}$，K 为关键词的个数，每个关键词对应着向量的某一维，将每个文档用向量的形式进行表示记为 $d_j=(w_{1j},...,w_{kj})$，其中 w_{1j} 表示关键词 t_1 在文档 j 中的权值，若 t_1 在该文档中出现，则对应的值为非 0，否则为 0。计算这些权值的方法有很多，最常用的方法是使用 TF-IDF[53](term frequency–inverse document frequency, 词频-逆文档频率)衡量某个关键词对某个文档的重要程度。TF-IDF 的基本思想是：

(1) 一个关键词在一个文档中出现的频率越多，就越能代表这个文档(TF)；

(2) 一个关键词，包含它的文档越少它就越能用于区分开各个文档(IDF)；

TF(term frequency)代表一个词在某个文档中出现的次数。

$$TF(t_i,d_j)=\frac{f_{ij}}{\sum_{z=1}^{k} f_{zj}} \tag{3-16}$$

上式中 t_i 代表词 i， d_j 代表第 j 个文档， f_{ij} 代表关键词 i 在第 j 文档出现的次数，k 为关键词的总数。

关键词 t_i 的 IDF(inverse document frequency)表达形式为：

$$IDF(t_i)=\log\frac{N}{n_i} \tag{3-17}$$

上式中 N 代表文档总数， n_i 代表总的文档中含有词 t_i 的数目，IDF 公式用于衡量某个关键词对文档的区分度。结合 TF 和 IDF，给出词 t_i 在第 j 文档的权重公式：

$$w_{ij}=TF(t_i,d_j)\times IDF(t_i) \tag{3-18}$$

一个文档 d_j 的内容现在可以用一个向量来表示：

$$d_j = (w_{1j}, ..., w_{kj}) \tag{3-19}$$

基于内容的推荐算法给用户推荐那些与用户以前喜欢的商品或用户目前最感兴趣的商品相似的产品，即向目标用户推荐与用户的配置信息最相似的商品。设用户的配置信息用向量d_i表示，商品的特征信息用d_j表示，则用户偏好与商品特征间的相似度用余弦计算，具体公式为：

$$sim(\boldsymbol{d}_i, \boldsymbol{d}_j) = \frac{\boldsymbol{d}_i \cdot \boldsymbol{d}_j}{\|\boldsymbol{d}_i\|_2 \times \|\boldsymbol{d}_j\|_2} \tag{3-20}$$

基于内容的推荐系统能够将新商品和非热门的商品推荐出去，只要内容相似，再冷门的商品都有可能得到推荐；其次通过获取商品的特征信息，能更好的解释为什么向用户推荐这些商品，只需告诉用户这些商品有某某特征跟您的品味很符合，使用户有更好的购物体验。但其本身也存在一些缺点，如下所示：

1) 商品特征难以获得或被表示，如果商品是文档形式，则商品的特征很好获得。但现实生活中的商品往往是视频、图像和声音，这些商品的特征都不太好获得。另外我们得到的特征仅是商品的一部分特征，并不能完全代表其本身，如果特征抽取不合理，可能会是两个相差很远的 item 误认为是完全相同的。
2) 推荐内容太过单一，基于内容的推荐只会推荐那些与用户以往购买的商品相似的商品，挖掘不了用户潜在感兴趣的但却一直没购买的商品，如推荐系统发现用户喜欢看古装剧，就会经常向用户推荐古装剧，但也许用户对悬疑片、科幻片同样感兴趣，只是一直没去尝试看过。另外一直推荐与用户相似的商品久而久之用户也会觉得毫无新奇性，最终可能会觉得厌烦，不再使用该系统。
3) 无法解决新用户问题。新用户没有任何浏览或购买信息，系统没法获取其偏好，因此就无法向他推荐商品，除非该用户自己填写用户配置信息。

3.2.3 混合推荐技术

上述的协同过滤算法和基于内容的推荐算法在应用中都有各自的缺点，将两者结合往往可以获得更有效的推荐质量。混合推荐技术把多种不同的推荐算法相结合用来提高推荐系统的准确率。简单地可以将协同过滤和基于内容的推荐算法各自的推荐结果使用加权进行组合[54]，或在使用协同过滤算法时增加一些基于内容的功能都可以在一定程度上缓解推荐系统中的冷启动和数据稀疏问题。复杂点

的使用机器学习中的集成学习方法将各种类型的推荐算法结合，都能提高推荐系统的准确率。

2006 年 10 月 Netflix 公司举行了一项比赛，要求参赛者提交推荐算法预测 Netflix 的客户分别喜欢什么影片，若能把预测的效率提高 10%以上者将获得百万美元大奖。2009 年 BellKor’s Pragmatic Chaos[55]将 Netflix 的推荐准确率提高了 10.06% ，从而成为最终的获奖团队。该团队的算法组合了 BellKor，Pragmatic Theory 和 BigChaos 三个团队的算法。位居第二的 The Ensemble 团队使用 Bagging 将 KNN 近邻，矩阵分解与降维，Boltzmann 机和 RBM，基于时间的模型组合。

组合方法虽然能够提高推荐的准确率，但一般来说某算法包含的子算法越多该算法就越复杂，越难理解，同时所需的计算时间就越多。降低有大量算法组合的模型的时间复杂度是影响混合算法以及集成方法的关键。

3.3 评价指标

本节主要介绍关于评价推荐系统准确度的一些指标，主要包括预测评分准确度和基于推荐列表的准确度。

3.3.1 基于预测评分的准确度

当希望计算模型预测评分与真实评分间的差异时，可以使用基于预测打分的评价指标预测系统的准确度。

评估预测评分准确度的最简单的方法是直接计算模型预测评分与真实评分之间的平均绝对误差[56](Mean Absolute Error, MAE)，计算公式如下：

$$MAE = \frac{1}{c}\sum_{i=1}^{c} | r_{u,i} - r'_{u,i} | \tag{3-21}$$

其中 c 为测试集中用户 u 打分的商品数目，$r_{u,i}$ 是测试集中用户 u 的实际打分，$r'_{u,i}$ 为系统给出的预测打分。整个系统的准确度是所有用户准确度的均值。MAE 计算简单，通俗易懂因此得到了广泛的应用。但 MAE 方法也有很大的局限性，MAE 衡量的是整体的误差，但用户一般只关注商品推荐列表前面的一部分商品，对于一个系统即使它的 MAE 值很小，也不能说明它的推荐质量就很好，因为该算法很可能只是对于分值较低的商品预测的比较准确，以此降低了整体的 MAE 值。而有些系统 MAE 值很大，但它却能把应该位于推荐列表前面的商品预测准确，这种算法

的性能也是不错的。

除了平均绝对误差 MAE 方法，基于预测评分准确度的指标还有平均平方误差(Mean Squared Error，或 MSE)和均方根误差(Root Mean Squared Error),如公式(3-22)和公式(3-23)：

$$MSE=\frac{1}{c}\sum_{i=1}^{c}(r_{u,i}-r'_{u,i})^2 \tag{3-22}$$

$$RMSE=\sqrt{\frac{1}{c}\sum_{i=1}^{c}(r_{u,i}-r'_{u,i})^2} \tag{3-23}$$

平均平方误差 MSE 将用户实际评分与系统预测打分的差值进行平方，因此其对平均平方误差的影响比平均绝对误差大。RMSE 是 MSE 的开方，它使得最终结果与真实的打分在一个数量级上。

针对需要在不同的数据集上进行比较的情况，提出了标准平均绝对误差 NMAE[47](Normalized Mean Absolut Error)：

$$NMAE=\frac{MAE}{r_{\max}-r_{\min}} \tag{3-24}$$

其中 $r_{\max}$ 与 $r_{\min}$ 分别是测试集中用户对商品打分的最小值和最大值。

3.3.2 基于推荐列表的准确度

本小节介绍的评价指标适用于二分类喜好（喜欢、不喜欢）的协同过滤系统，或不关心用户具体打分而只关注推荐列表正确性的推荐系统。其主要的评价指标有：准确率(Precision)、召回率(Recall)、F 值、AUC、NDCG。

表 3-1 商品推荐的四种情况

	系统推荐商品数	系统未推荐商品数
用户喜欢的商品数	N_{rl}	N_{nl}
用户不喜欢的商品数	N_{rh}	N_{nh}

对于一个用户未曾选择过或打分的商品，结果有四种如表 3-1 所示。准确率 P 表示系统推荐的商品列表中用户实际喜欢的商品与被推荐的所有商品的比：

$$P=\frac{N_{rl}}{N_{rl}+N_{rh}} \tag{3-25}$$

召回率 R 表示推荐列表中用户喜欢的商品数占系统中用户实际喜欢的所有商品的比例，即用户喜欢的商品被推荐的比例：

$$R = \frac{N_{rl}}{N_{rl} + N_{nl}} \tag{3-26}$$

对一个推荐系统当准确率 P =1 时，表示推荐的商品都是用户喜欢的商品。召回率 R =1 时，表示用户喜欢的商品都得到了推荐。理想情况下是两者都很高，但通常它们存在负相关性，一个高另一个就会低，一个低另一个就会高。一般不会单独使用准确率或召回率来评价一个系统的好坏，Pazzani M 提出了 F-measure, 将两者融入一个式子，同时考察准确率和召回率,如公式(3-27)：

$$F = \frac{2P * R}{P + R} \tag{3-27}$$

除了上面的指标外，评价指标还有用于衡量分类准确率的 ROC(receiver operating characteristic)曲线和 AUC 指标。AUC 的值等于 ROC 曲线下面的面积，它的值介于 0.5-1.0 之间，表示一个推荐系统能够在多大程度上将用户喜欢与不喜欢的商品区分开，值为 0.5 时表示跟随机猜测的准确度相同，大于 0.5 表示该算法准确度优于随机猜测。一般采取如下方法近似的计算 AUC 的值：随机的从测试集中选取出一个商品 i (即用户喜欢的商品)，然后随机选择一个不在测试集中的商品 j，若商品 i 的预测评分高于商品 j 的评分，则 n' 的值增加 1，若商品 i 的预测评分等于商品 j 的评分则 n'' 的值增加。将此比较过程重复 n 次，那么 AUC 的计算公式如下所示：

$$AUC = \frac{n' + 0.5 * n''}{n} \tag{3-28}$$

由上式可以看出随机猜测时，AUC 的值为 0.5。

上述的评价指标比较侧重于系统的分类准确率，但对于比较看重推荐列表位置的系统来说，上述的指标显然是不太合适的。NDCG(Normalized Discounted Cumulative Gain)是 Learning to Rank for IR(信息检索)领域的评价指标，应用在推荐上来说它的主要思想就是：

1、用户喜欢的商品也有很多级别，如在 1-5 分制中，5 分的商品相对 2 分的商品就该赋予更高的权重。

2、用户喜欢的商品应该出现在推荐列表的前面，因为用户一般只对推荐列表前面的商品进行浏览，对于排在后面的商品用户由于精力或者耐心等因素可能根

本兴没趣查看，因此对于商品本身的位置也应该赋予一定权重。

这种评价方式使得推荐的准确度不仅与商品本身有关，而且与商品所处的位置也有关。如下为 NDCG 的公式：

$$NDCG_p = \frac{DCG_p}{IDCG_p}$$
$$DCG_p = \sum_{i=1}^{p} \frac{2^{r_i} - 1}{\log_2(i+1)} \tag{3-29}$$

其中 p 表示的是推荐列表的长度，i 代表模型预测的商品的排名，r_i 表示排名位置为 i 的商品对应的测试集中的打分值。$IDCG_p$ 是理想状态下的 DCG_p 应该达到的值。$NDCG_p$ 是标准化的 DCG_p。

3.4 本章小结

本章简要介绍了推荐系统中主流的几种推荐技术，分析了每种技术的优缺点，并简要介绍了不同推荐系统所需的几种评价指标。准确率是衡量推荐系统性能的一大指标，本文的第四章和第五章都是致力于提高推荐的准确度。

第四章 改进的协同过滤算法

4.1 问题提出及现有的解决方案

传统的相似度算法如在第三章提到的余弦相似度和Pearson相似度在推荐系统初期得到了广泛的应用，但是这种算法产生的一个缺陷是当评分矩阵比较稀疏时，这两种算法的准确率就会严重下降，因为传统的相似度算法是根据用户的以往打分情况给用户推荐商品，用户必须给足够的商品打过分，系统才能得到其正确的相似邻居，但是当用户的打分非常稀疏时使用上面的方法得到的相似度是不准确的。另一方面现在的电子商务系统用户数量和商品种类增长飞速，打分矩阵就会变来越来越庞大，计算相似度的时间复杂度就会急剧增加，不利于系统的扩展。

为解决上述的数据稀疏性问题,常见的方法有给打分矩阵中没有打分的商品一个默认的缺省值进行填充[58-59]或者使用用户的打分均值来填充，该方法在一定程度上可以缓解数据稀疏的问题，但用户对商品的喜好程度不可能是完全一样的，因此该方法并不能较有效的解决稀疏性问题;除了使用缺省默认值填充外，国内还有研究者提出将基于商品的协同过滤和基于用户的协同过滤相结合，首先计算商品之间的相似度，根据相似度大小得到商品的相似邻居，根据商品相似邻居获得未评分商品的预测值，填充稀疏打分矩阵，然后再使用基于用户的协同过滤方法进行推荐。

使用降维的技术如 SVD(奇异值分解)技术，将打分矩阵分解成为几个矩阵，然后用生成的矩阵相乘反过来预测用户对未知商品的打分。矩阵分解方法降低了输入矩阵的维度，该方法能有效缓解数据稀疏的问题，但该方法在模型的构建阶段花费的时间和空间代价都是很大的，另外它比较适用于基于预测评分的系统，当应用于推荐列表时推荐的质量会大大下降。此外还有将 SVD 和基于用户的协同过滤相结合的方法，使用 SVD 先估计出未打分商品的估计值，填充稀疏矩阵，然后在填充过的矩阵上使用用户最近邻进行推荐。

除上面的方法之外常用的还有基于数据挖掘聚类的协同过滤算法，该方法使用某种聚类算法（如 K-means）将用户或商品聚成簇。如对用户进行聚簇后，则具有相似品味的用户被分到了一个簇里，然后对于每一个簇使用簇内的其他用户对商品的打分来预测目标用户对未打分商品的打分值。基于聚类的协同过滤方法已

经得到了广泛的应用，如 1998 年 Unger 和 Foster[62]提出了一种基于 K-means 的 Gibbs 抽样方法，将购买过相似商品的用户聚成簇，然后将可能被同一用户喜欢的商品聚成簇。2002 年 Sarwar[62]根据用户间的相似性将用户进行聚簇，实验验证这一聚类方法可以与传统的 CF 方法相媲美，但值得一提的是该方法极大地提高了推荐系统的在线性能。基于聚类的协同过滤使得推荐的时候只需让特定用户与簇内的其他用户进行比较而不涉及到整个的用户集，该方法对于在线推荐来说有利于系统的扩展。

本文提出了一种改进的协同过滤算法 PCS，该算法针对传统相似度在数据稀疏时推荐质量严重下降的缺点，从流行度、共同打分商品数和打分差异度三个方面对 Pearson 相似度进行了改进。同时随着用户数目和商品种类的增多，计算所有用户间的相似度的时间复杂度太高，不利于在线系统的扩展，因此本章使用决策树算法用于用户的分组上，这就提出了一种新的选择相似邻居的方法，分组后只在组内进行个性化推荐，而不用计算每对用户间的相似度。

4.2 算法设计

传统协同过滤核心思想是利用用户以往打分信息计算用户间的相似度；然后根据相似用户的打分预测目标用户对未打分商品的喜好程度，以此给用户推荐商品。本章的 PCS 算法首先使用决策树将用户进行分组，然后仅在在组内进行个性化推荐，在组内用改进的相似度方法计算用户间的相似性，最后根据相似度进行推荐。

4.2.1 决策树技术

本节简要介绍一下决策树算法的思想。决策树是数据挖掘中用于解决分类问题的常用方法，它利用一个像树一样的决策流程图进行归纳学习，最后生成决策树模型。该模型是自上而下递归生成的，在每个节点处都含有一组样本，每个非叶节点都有一个分割属性，用于将样本集根据该分割属性的取值对应到不同的分支上，每个分支代表这个特征属性在某个值域上的输出，而每个叶节点代表从根节点走到叶节点所代表的这一类样本的类别。使用决策树模型进行决策的过程就是根据已经构成的决策树模型从根节点开始，测试未分类样本对应的特征属性，按照特征属性的值选择相应的输出分支，重复这一过程直到到达叶子节点，将叶子节点存放的类别作为未分类样本的决策结果。基于决策树的分类算法有很多，

比较经典的是较早时期提出的 ID3 和 C4.5 分类算法。

ID3 算法是 Quinlan 提出的较早的经典决策树算法，图 4-1 给出了 ID3 决策树算法的流程图：该算法的核心思想是：在选择最佳分割属性时使用信息增益(information gain)作为最佳分割属性的选择标准，目的是使用信息增益最大的属性作为分割节点能获得关于样本集最大的类别信息。其具体方法为：对当前树节点所包含的属性列表 attribute _ list 中任一未被选择为最佳分割点的属性，计算将它作

Algorithm: *ID*3

输入：样本集$D=\{(x_1,y_1),\ldots,(x_n,y_n)\}$，是一组训练样本和对应的类标号。

attribute _ list：候选分割属性的集合。

Generate_decision_tree(*D*,*attribute _ list*)

(1) 创建一个结点*N*;

(2) *if D*中的样本都属于同一类*C*, *then*

*return N*为叶子节点，*C*为所属类别

(3) *else if attribute _ list*为空，*then*

*return N*为叶子节点，*D*中比例最高的样本的类别为所属类别

(4) *else*使用信息增益找出最佳的分割属性*split _ best*

(5)用*split _ best*标记节点*N*

(6)*attribute _ list*=*attribute _ list*−*split _ best*

(7) *for split _ best*的每一个取值*j*

*D j*是*D*中*split _ best*属性值为*j*的样本集合

*if D j*为空集*then*

节点*N*为叶子节点，并标记为*D*中的多数类

else 调用算法*Generate_decision_tree*(*D j*,*attribute _ list*)

endfor

图 4-1 ID3 算法伪代码

为最佳分割点后对应的信息增益，将信息增益最大的属性作为最佳分割点，选择好最佳分割属性后该属性有多少个取值就对应多少分支，将样本根据属性对应的值分配到不同的分支。重复这一过程直到所有子集仅包含同一类别的数据为止。其中属性列表 attribute_list 初始的时候包含所有的样本属性。ID3 算法的优点是：算法引入了信息论，使用信息熵作为选择最佳分割属性的基础理论清晰，因此 ID3

方法简单易懂，它的缺点是：算法对噪声数据比较敏感，当训练数据集增大时，模型可能会随之改变；在选择分割属性时，比较偏向于多值属性；该算法只能处理离散属性值。C4.5 算法是 Quinlan 在 1993 年提出的，它是 ID3 算法的一种改进，它使用信息增益率代替 ID3 中的信息增益来选择分割属性，克服了 ID3 偏向于多值属性的缺点，并能处理连续属性（将其进行离散化）。

决策树算法广泛应用的原因取决于它的以下优点：

(1) 决策树算法理解起来简单，人们在听过解释后都能很快明白决策树算法所表达的意义，并且该算法很容易实现。

(2) 决策树的构造只需要很少的参数设置，不需要任何的先验知识，比较适合于探测式的发现某些关联。

(3) 可以较好的处理大规模的数据。

(4) 决策树分类器的预测效果一般都比较好，它可以被使用在绝大多数领域数据集的分类。

4.2.2 决策树进行分组

由于在构造过程中不需要任何的先验知识，只有很少的参数设置，生成的模型比较简单、且有比较好的分类精度因此决策树被广泛应用到分类和预测上。但将决策树用到推荐上的研究还很稀少，文献[63]使用 web 数据挖掘和决策树进行个性化推荐，首先使用决策树选择出那些可能会购买推荐商品的目标顾客，然后使用 web 挖掘技术通过点击流自动获取用户偏好和产品关联，最后通过用户偏好和产品间的关联给目标用户进行推荐。由于决策树具有的以上几个优点，且能根据上一次选择的分割属性自适应的调整下一次选择的分割属性，本章将决策树应用到个性化推荐上，使用决策树将用户进行分组。这种方法不必再计算全部用户间的相似度然后根据相似度确定相似邻居，而是建好决策树模型后位于叶子节点的所有用户就是相似用户，仅在组内进行推荐降低了相似度计算的时间复杂度，提高了系统的可扩展性。

本章将决策树应用到个性化推荐领域，提出了一种获得相似邻居的新方法。在此需要重新定义 4.2.1 节讲述的决策树中的样本和属性，将用户对应为其中的样本，将商品对应为样本分割属性。决策树的每一个节点都包含一组用户，每一个非叶子节点用来衡量用户对某一商品的喜好程度，位于同一叶子节点的用户属于同一组，表示他们的偏好相同，即为相似邻居。决策树的构造过程为一个用户在

头结点被问一个问题（如用户对某商品是否喜欢），根据用户对该问题的反应（如喜欢该商品、不喜欢该商品或不知道），会将用户对应到相应的分支上，另外由于决策树采用自适应的调整方法，则该用户下一次被问的问题与上一次的反应有关，用户从头节点根据其对提出的问题的反应沿着路径一直走到叶子节点，这一过程结束后就得到了用户的相似邻居，根据相似邻居对商品的打分就能获得推荐列表。在构造树的时候，由于本文实验是基于 MovieLens 数据集，而 MovieLens 数据集中用户的评分是 1 到 5 分，我们将用户的评分作如下处理：商品打分为 1-2 分视为用户“讨厌”该商品，商品打分为 3-5 分视为用户“喜欢”该商品，商品未打分视为用户对该商品的喜好“未知”。树的构造过程是一个自上而下的递归过程，树的内部节点代表的是选择的有代表性的商品，通过用户对商品的喜好将用户分组，在经典的 ID3，C4.5 算法中分别使用信息增益和信息增益率选出最佳分割属性，本文使用推荐系统中比较常用的 RMSE 指标选出最佳分割商品，根据用户对该商品的喜好将用户分到不同的分支。

例如给定一个树节点 t,候选商品 i,根据节点 t 的用户对商品 i 的喜好，将用户分为三组 $tl(i)$,$th(i)$,$tu(i)$，喜欢商品 i 的用户被分到 $tl(i)$ 组,讨厌商品 i 的用户被分到 $th(i)$ 组,未对该商品打分的用户分到 $tu(i)$ 组,则节点 t 关于商品 i 的均方根误差(RMSE)总和为公式(4-1)：

$$E_{st}(i) = e(tl(i)) + e(th(i)) + e(tu(i)) \tag{4-1}$$

对所有未被选择过的商品计算它的均方根误差(RMSE)和，选择均方根误差和最小的商品作为节点 t 的最佳分割节点：

$$split(t) = \arg\min_{i} E_{st}(i) \tag{4-2}$$

其中 $e(tl(i))$ 为在节点 t 的所有用户中对商品 i 的打分为 3-5 的所有用户的均方根误差，$e(th(i))$ 为在节点 t 的所有用户中对商品 i 的打分为 1-2 的所有用户的均方根误差，$e(tu(i))$ 为在节点 t 的所有用户中未对商品 i 打分的所有用户的均方根误差,为简单起见在此分别标记为 $e(tl)$，$e(th)$，$e(tu)$，其中 tl 为节点 t 中喜欢商品 i 的用户所在的节点，相应的 th,tu 分别为讨厌商品 i 的用户和未对商品 i 打分的用户所在的节点。若在节点 tl 关于商品 j 的均方根误差为公式(4-3)(其中 u 为属于 tl 且对商品 j 进行打分的用户，$R(j)$ 表示对商品 j 打分的用户集)：

$$e(tl)_j = \sqrt{\frac{1}{|tl \cap R(j)|} \sum_{u \in tl \cap R(j)} (r_{u,j} - m(tl)_j)^2} \tag{4-3}$$

公式(4-3)中 $m(tl)_j$ 为节点 tl 的所有用户对商品 j 打分的均值，其具体的计算方式为：

$$m(tl)_j = \frac{\sum u \in tl \bigcap R(j)^{r_{u,j}}}{|tl \bigcap R(j)|} \tag{4-4}$$

则在节点 tl 的均方根误差记为：$e(tl) = \sum_j e(tl)_j$ 。

上述是构造一个树节点的完整过程，对当前节点找到最佳分割商品后，根据用户对该商品的打分将用户对应到相应的分支，构造决策树的过程是自顶向下的，递归调用上述过程直到到达叶子节点，将用户分组。

一般来说，构造决策树最简单的终止条件是直接设定树的深度，如设定深度 depth 为 5 或 6，但是单独用这一终止条件，准确度不高必须与其它方法相结合；当商品分割属性用完时终止，但是这一条件在个性化推荐中可以忽略，因为个性化推荐中商品的种类是很多的，构造树时不可能用完；设定一个均方根误差的最小阀值 minE，当当前节点根据选出的最优分割商品进行分割后的均方根误差总和大于最小阀值时；另外为避免数据极端稀疏的情况，当当前节点的用户打分数目之和小于某个给定阀值 minN 时，也作为终止条件。

4.2.3 改进的相似度计算方法

将所有用户根据上述的决策树方法进行分组后，就确定了每个用户的相似邻居，然后仅在组内进行个性化推荐。简单情况下可以使用一组内的所有用户对某一商品的打分均值作为某一商品的预测打分，但该方法将所有用户的打分贡献都看做是相同的，没有考虑用户间的打分相似度，没有体现出个性化推荐；本章使用基于用户相似度的方法预测目标用户对特定商品的打分。但由于打分矩阵的稀疏性，传统的相似度计算方法并不准确，因此本节提出了一种改进的方法，提高推荐的质量。

在第三章 3.2.1 节介绍了协同过滤算法中常用的相似度计算方法，包括余弦相似度、Pearson 相似度和修正的余弦相似度。这三种相似度计算方法在个性化推荐系统中得到了广泛应用，但是它们同样存在很多问题：

(1) 若用户 a , b , c 的打分向量分别为 $<1,1>$, $<5,5>$ 与 $<5,4>$, 则使用基于 cosine 的相似度方法计算 a 与 b 的相似度为 1，忽略了用户的打分差异，且 b 和 c 的 cosine 相似度小于 1，但观察发现 b 和 c 的相似度理应高于 a 与 b 的相似度。

(2) 如用户 a , b , c 的打分向量分别为 $<4,0,0,2>$, $<4,3,2,2>$ 与 $<3,3,2,1>$, 则用

Pearson 得出的结果是用户 a 与 b 之间的相似度高于用户 b 和 c 的相似度，但通过观察会发现 b 和 c 的相似度理应高于用户 a 与 b 的相似度。这种情况在打分矩阵极为稀疏时更严重。

(3) 当用户 a,b 的打分分别为 $<5,1>,<3,1>$ 时，用 Pearson 相似度计算用户 a 与 b 的相似度为 1，完全忽略了用户的打分差异。

其中 Pearson 相似度侧重考虑用户间的共同打分商品，当以往的用户-商品打分信息较充足时准确度相对高一点，但是当打分矩阵稀疏时，该算法性能还不如 Cosine 相似度，因此本章以 Pearson 相似度为研究对象，对 Pearson 相似度从以下几个方面进行改进：

(1) 流行度(P)：假设用户 u 与用户 v 均购买了商品 i，当绝大多数用户也都购买了该商品时那么该商品对计算用户 u 与用户 v 之间的相似度价值就不大，因为该商品是流行商品、热门商品，两个用户同时购买了该商品并不能说明他们品位相同。相反如果该商品购买的人极少，而这两个用户却同时购买了该商品，则能充分说明两人有共同的兴趣爱好。在此引入流行度系数 P，计算方式如公式(4-5)(其中 $R(i)$ 为对商品 i 打过分的所有用户的集合)。由于 MovieLens 数据集中用户至少对 20 种商品打过分，本文以 20 作为分割点，当对商品 i 打过分的用户数不大于 20 时，流行度系数 P 为 1。具体的形式如公式(4-5)所示：

$$P=\begin{Bmatrix}\dfrac{1}{\sqrt{|R(i)|-20}} & if\ R(i)>20\\ 1 & otherwise\end{Bmatrix} \tag{4-5}$$

(2) 共同打分商品数(C)：Pearson 相似度侧重的是两个用户共同打分的商品，若两个用户共同打分的商品集很少，那么即使共同打分的商品的分值很相似甚至一样，也并不能说明两个用户的品位相同，但用 Pearson 相似度计算时，得出的相似度却很高。在此引入共同打分商品数系数 C，计算公式如下，当两个用户共同打分的商品数大于等于 L 时，C 为 1，其中 L 是一个常数，一般取值为 20-50，本章实验部分 L 取值为 50：

$$C=\begin{Bmatrix}\dfrac{|R(i)\bigcap R(j)|}{L} & if\ |R(i)\bigcap R(j)<L\\ 1 & otherwise\end{Bmatrix} \tag{4-6}$$

(3) 打分差异度(S)：MovieLens 数据集打分值为 1-5 分，将 1-2 分视为用户讨厌某商品，3-5 分视为用户喜欢某商品。当用户 a , b 的打分分别为 $<5,1>,<3,1>$ 时，用 Pearson 相似度计算用户 a 与 b 的相似度为 1,忽略了用户的打分差异；此外用户 u 与用户 v 打分 $<5,1>,<3,1>$，用户 u 与用户 v 打分分别为 $<4,1>,<2,1>$，这两种情况下 Pearson 相似度值是一样的，但对于前一种情况 3 与 5 分表示两个用户都喜欢该商品，而后一种情况 4 与 2 分表示一个用户喜欢该商品，一个用户讨厌该商品，因此前一种应比后一种情况的相似度权值应该高一点，因为两个用户对该商品的喜好度是一样的，引入打分差异度因子 S：

$$S(r_{u,i}, r_{v,i}) = \frac{1}{(1 + d(r_{u,i}, r_{v,i}))} \tag{4-7}$$

其中 $d(r_{u,i}, r_{v,i})$ 表示用户 u 与用户 v 对商品 i 的打分差值，当两个用户对商品的喜好不同时，需要进一步加重惩罚以减小用户相似度权值，具体计算方式如下：

$$d(r_{u,i}, r_{v,i}) = \begin{cases} |r_{u,i} - r_{v,i}| & if \text{ 两个用户对商品} i \text{的打分均为1-2或均为3-5} \\ 2*|r_{u,i} - r_{v,i}| & otherwise \end{cases} \tag{4-8}$$

则根据上面三个方面的改进，最终的用户相似度计算公式为：

$$sim(u,v) = \frac{C}{\sqrt{\sum_{i \in I_{uv}} (r_{u,i} - \overline{r_u})^2} \sqrt{\sum_{i \in I_{uv}} (r_{v,i} - \overline{r_v})^2}} \sum_{i \in I_{uv}} (r_{u,i} - \overline{r_u})(r_{v,i} - \overline{r_v}) * P * S \tag{4-9}$$

4.2.4 推荐产生

当使用决策树方法通过自适应的选择分割商品将用户进行分组后，在组内进行个性化推荐，其中目标用户的相似邻居不再需要通过计算所有用户间的相似度来获得，而是通过构造决策树获得，这样节省了时间复杂度，也有利于系统的扩展。在组内进行个性化推荐时，由于以往的相似度计算方法存在数据稀疏的问题，本章以 Pearson 相似度为研究对象，从流行度、共同打分商品数和打分差异度三方面对其进行改进，在组内计算用户间的相似度用来获得用户间的相似度权重，最后用户 u 对商品 i 的预测打分的具体计算公式为：

$$p_{u,i} = \bar{r}_u + \frac{1}{\sum_{u' \in \hat{U}} sim(u,u')} \sum_{u' \in \hat{U}} sim(u,u') \times (r_{u',i} - \bar{r}_{u'}) \tag{4-10}$$

式中，$\bar{r}_u$ 与 $\bar{r}_{u'}$ 分别表示用户 u 与用户 u' 的打分均值，$sim(u,u')$ 表示用户 u 与用户 u' 的相似度。

4.3 实验验证

本章采用 MovieLens 数据集对 PCS 算法进行验证，其中 MovieLens 是一个推荐系统网站，使用协同过滤技术根据用户的喜好向用户推荐电影。本章使用 MovieLens100K 数据集，其中用户数目 Users 为 943，商品数目 Items 为 1628，每个用户至少对 20 个商品打过分，总的投票数为 100K，数据稀疏度为 93.5%。采用五次交叉验证方法，随机将数据集分为五等份，每次取一份做测试集，其余四份做训练集。训练集与测试集的比例为 4:1。

由于本章是对相似度的一种改进，因此本章选择的对比算法是传统的相似度算法：Cosine 和 Pearson 相似度算法。而评价指标选用的是第三章 3.3.2 节介绍的准确(Precision)、召回率(Recall)、F-Measure 指标。由于给用户推荐商品时，用户往往只会对推荐列表前面的一部分商品比较关注，对比较靠后的商品一般都不会去浏览，因此试验中没有将推荐的所有商品都进行考虑，本章选取的推荐列表的长度为：5 ,10, 20, 40, 50 , 80 ,100 ,130, 150, 200。

从图 4-2 可以看出随着推荐列表的长度的增加，三种算法的准确率都会相应的下降。在推荐列表长度大于 150 时，可以看出 Cosine 和 Pearson 算法的准确度逐渐接近于 PCS 算法。但还是能明显看出本章提出的改进的相似度算法 PCS 在准确率上要优于基于 Cosine 和基于 Pearson 的相似度算法，特别是在推荐列表长度小于 50 的时候。

图 4-3 中 PCS 和 Cosine 算法在推荐列表小于 5 时，召回率几乎没多大差别，但是随着推荐列表长度的增加，两个算法的差别逐渐明显。整体可以看出 PCS 的效果最好，Cosine 其次，Pearson 效果最坏。图 4-4 中显示对于 F-measure 评价指标，PCS 的效果要优于 Cosine 和 Pearson，相比来说随着推荐列表增长，Cosine 会逐渐靠近 PCS，但 Pearson 算法的效果就差很多。对于 Pearson 相似度在准确率、召回率和 F 值效果都比较差的现象是 MovieLens 数据集比较稀疏造成的。

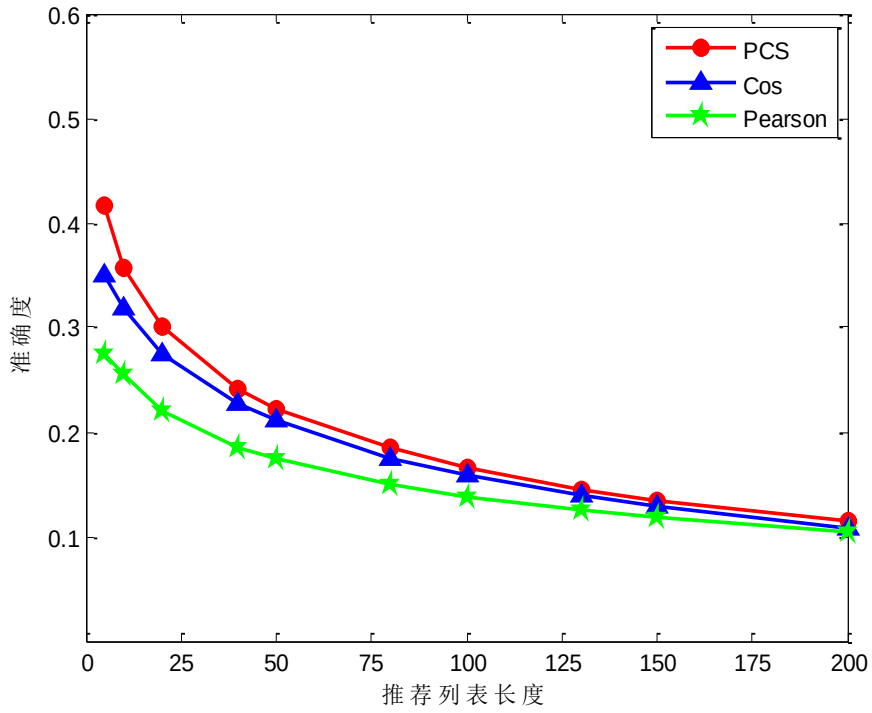

图 4-2 MovieLens 数据集上 PCS 与 Cosine 和 Pearson 的准确率对比

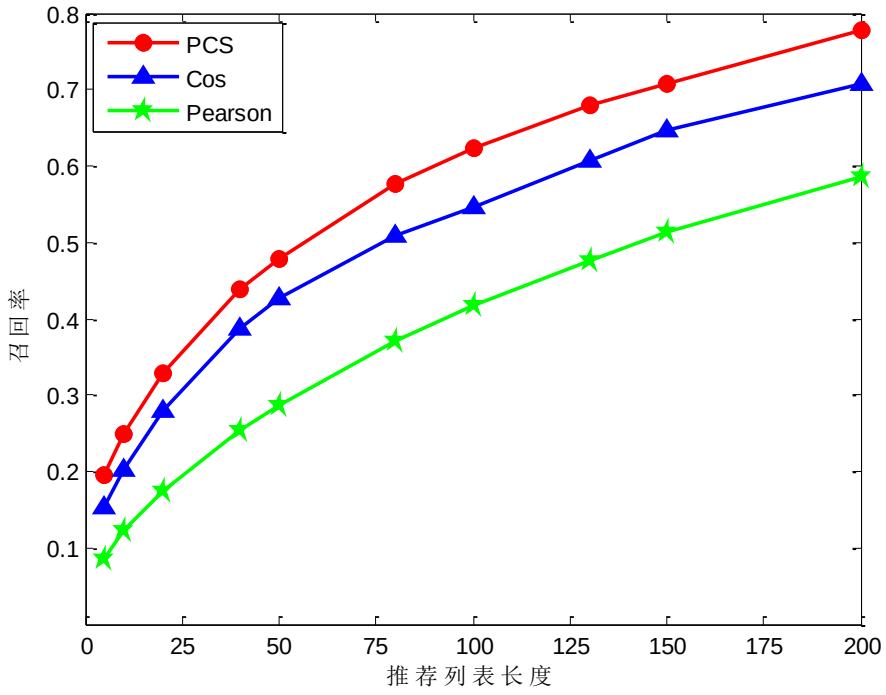

图 4-3 MovieLens 数据集上 PCS 与 Cosine 和 Pearson 的召回率对比

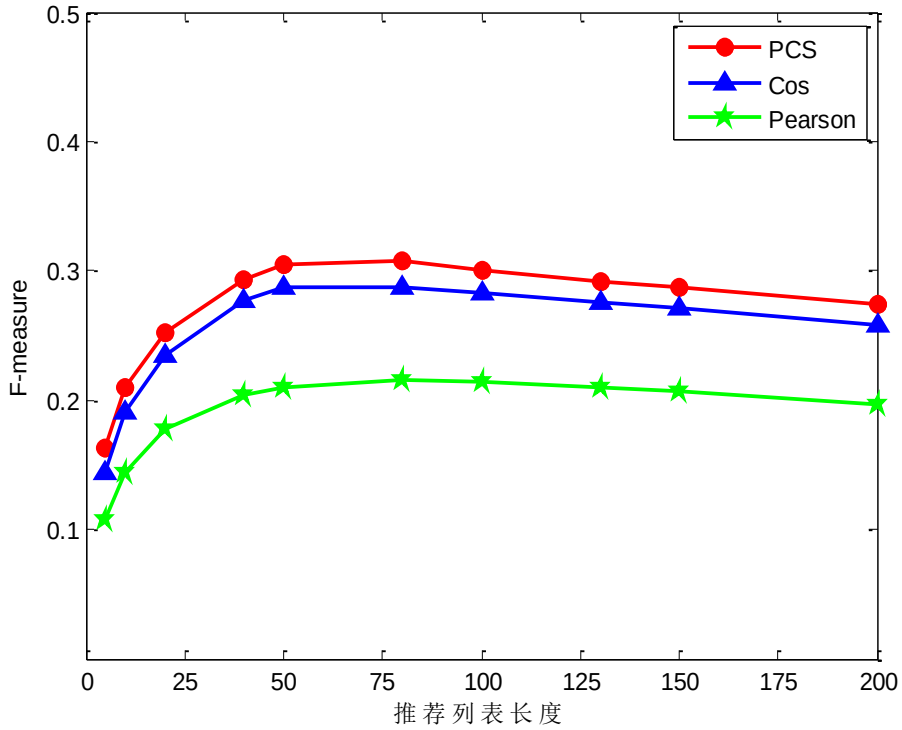

图 4-4 MovieLens 数据集上 PCS 与 Cosine 和 Pearson 的 F 值对比

4.4 本章小结

本章简要介绍了一下决策树技术，根据第三章介绍的传统的基于相似度的协同过滤的缺点提出了一种改进的协同过滤算法，首先使用决策树自适应选择分割属性的特点，根据用户对最佳分割商品的反应，将用户进行分组。然后在组内进行个性化推荐，由于传统的相似度计算方法存在数据稀疏性问题，本章从流行度、共同打分商品数和打分差异度三个方面对 Pearson 相似度算法进行了改进，提高了推荐的质量。

在个性化推荐领域，准确率是人们永远的追求目标，下一章将把集成学习应用到个性化推荐上,进一步提升推荐的效果。

第五章 基于集成学习的个性化推荐

无论是在 2009 年结束的 Netflix 大赛，还是 2010 年 yahoo 公司举办的 Learning to Rank 比赛，亦或是 ACM 举办的 KDD-CUP 大赛，顶级获奖团队大都采用的是集成学习技术，这主要得益于集成学习较强的泛化能力和稳定性，并且它只要求弱学习器的准确率优于随机猜测，这相对来说很容易满足。本章介绍了最大化软间隔算法 SoftBoost，将集成学习与协同过滤相结合，利用 SoftBoost 的强泛化能力和有效避免过拟合的优点，提升推荐系统的推荐质量。

5.1 问题的提出

为提高推荐系统的准确率，研究者们提出了很多改进算法，但大多都是单一算法或是几个算法的简单加权组合，虽然能在一定程度上提高推荐的准确率,但高的准确率就意味着算法的复杂度，难以理解度，另外使用单一算法的准确率也会达到一个瓶颈，究其原因主要是现实中影响推荐质量的因素很多，而单一的推荐算很难将多个影响因素考虑进去。

而集成学习的主要思想是通过合并一系列的预测函数解决同一个问题，其中每一个预测函数都可以独立的解决该问题，最后将多个预测函数的结果进行投票组合或者使用权值组合得到最终的结果。集成学习的方法就好比多个决策者共同决策一件事情，如同一个人觉得他生病了，要去医院诊断病情，一般情况下为了保险起见，人们会选择让多个医生进行诊断，而最终的结果可以是多个医生的投票结果或者有些医生的名望高他占的权重就大些，有些权重相对较小，将这些结果进行加权组合，用这种方法得到的结果就比较可信。

由于集成学习能够提升系统的泛化能力，因此集成学习已经成功应用到了很多领域，最近几年的 Netflix，Learning to Rank，KDD-CUP 大赛中，集成学习更是成为获奖团队们应用的主要技术：如 2010 年 3 月至 5 月 yahoo 公司发起的 Learning to Rank 挑战赛，比赛中发现获奖的顶级选手大多采用的都是集成方法，包括 Boosting, Bagging 和 Random Forests。如获得第一名的团队 Burges 等人使用了 12 个不同的排序模型进行线性组合，其中有 8 个使用基于 Boosted Tree 的 LambdaMART 的集成模型，有两个是基于神经网络的 LambdaRank 模型，另两个

使用的是基于 logistic 回归损失的 Mart 模型。Pavlov 等人提出 BagBoo 算法，他们将 bagging 算法和 Boosting 算法相结合，由于 Bagging 算法可以并行执行，因此算法的执行效率比较高，而且使用 bagging 算法可以避免过拟合现象。与此同时他们使用 Gradient Boosting 算法减小偏差，获得较好的准确率。2006 年 10 月发起的 Netflix 推荐大赛，Netflix 称任何组织或个人只要能够将它现有的推荐算法 Cinematch 的准确率提升 10%，就能得到一百万美元的奖金。该比赛于 2009 年结束，其中前两名的团队都使用了集成学习技术。The Ensemble 团队由于在提交最终结果时比 Bellkor's Pragmatic Chaos 团队晚了二十分钟，而位居第二，与 Netflix 的百万大奖无缘。Bellkor's Pragmatic Chaos 算法集成了 BellKor，Pragmatic Theory 和 BigChaos 三个团队的算法，且每个团队的算法又集成了多个算法，如 BellKor 就是线性组合了 100 多个基算法的结果。The Ensemble 团队主要使用了 KNN 近邻，矩阵分解与降维，Boltzmann 机和 RBM，基于时间的模型作为基分类器，然后使用 Bagging 集成框架将这些算法进行融合。

2003 年 Freund 等人提出了应用于排序问题的 Boosting 算法称为 RankBoost 算法，它是目前基于 pairwise 的最好的排序算法。该算法与其它 Boosting 算法相同，都是结合多个弱假设给出的排序结果，从而提高排序结果的准确度。该算法使用样本对进行输入，通过样本对间的相对排序确定最终的排序结果。本文使用 SoftBoost 算法框架，结合 RankBoost 基于对的思想，将协同过滤算法作为弱排序器，提出了 SoftRankBoost 算法，将集成学习应用到个性化推荐上。

5.2 算法设计

5.2.1 最大化软间隔算法 SoftBoost

FVeund 和 Schapire[28,40]使用训练误差,训练样本集的规模，弱假设的 VC 维和迭代次数给出了 AdaBoost 算法泛化误差的上界，该上界表明随着迭代次数变得很大时，将会出现过拟合现象。但早期的研究者通过实验却发现即使在经过上千次的迭代后，AdaBoost 也不会出现过拟合现象，不仅如此他们还发现当训练误差变为 0 后，泛化误差有时候还会持续降低。图 5-1 给出了以 C4.5 为弱分类器的 Boosting 算法的训练误差和测试误差，该实验使用的是 letter 数据集，图中上面的那条线是测试误差，下面的那条线是训练误差。随后 Schapire[33]使用间隔(margins)假说对此给出了解释，并给出了泛化误差新的上界，该上界表明 AdaBoost 的上述现象与样

本集的间隔分布有关，样本集的规模和弱假设的 VC 维有关，而与迭代次数没有任何关联。在样本数目和弱假设的 VC 一定的情况下，拥有 margins 值较大的样本个数越多，AdaBoost 的泛化能力就越好。图 5-2 显示了在迭代次数分别为 5,100,1000 次之后数据集 letter 上的训练样本的间隔分布，对应的三条曲线分别为短虚线、长虚线以及实曲线。图中可以看出在训练误差降为 0 后，AdaBoost 会继续提升样本的间隔，相应的测试误差就会继续下降。

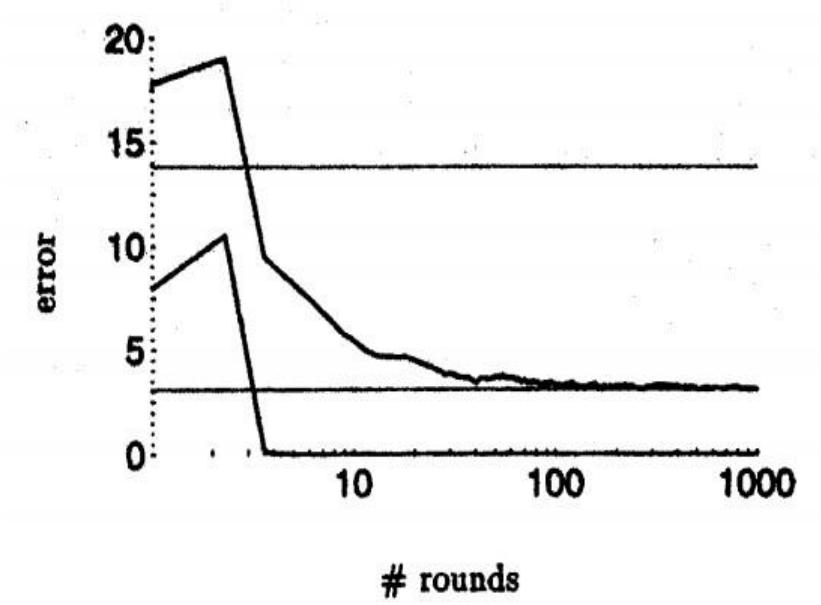

图 5-1 AdaBoost 的训练误差和测试误差[33]

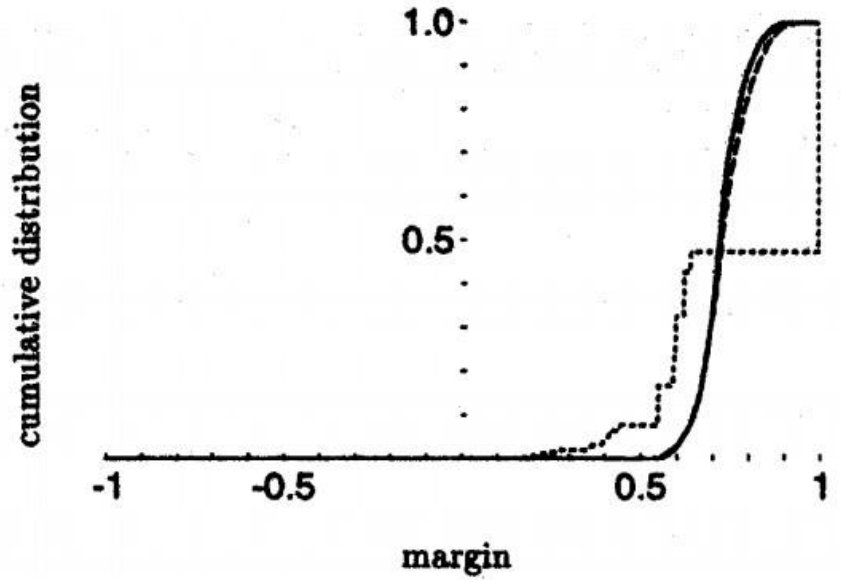

图 5-2 样本集间隔分布[33]

然而 Breiman[34]等人提出了对间隔分布假说的疑问，他们给出了泛化误差一个更加严格的界限，认为泛化误差的好坏并不是与样本集的间隔分布有关，而是与样本集的最小间隔有关（样本集的最小间隔为样本集中所有样本间隔的最小值，称为样本集的间隔），经典的 AdaBoost 算法并没有最大化间隔。与此同时 Breiman 提出了 Arc- GV 算法，该算法能得到近似最大化间隔。Breiman 的这一原理有利的刺激了最大化间隔 Boosting 算法的发展，如 LPBoost, AdaBoost*,TotalBoost 等都是最大化间隔算法。

对于正常的数据集来说，最大化样本集间隔是提高泛化误差的关键因素，那么上述的 LPBoost 看起来是完美的。但是 LPBoost 算法属于最大化硬间隔算法，它考虑所有的样本点，特别是在存在噪声数据或者难分数据的数据集中，容易受少数点的影响，由于 LPBoost 特别关注这些样本，很可能会产生过拟合现象。为此提出了软间隔算法，即放松对一些噪声样本或难分样本的关注，允许这些样本在间隔的另一面，即允许将这些样本分错，但需要加入一些松弛变量对这些样本进行惩罚。由此看来，在存在噪声数据时，基于软间隔的算法避免了 AdaBoost 算法中出现的过拟合现象，进一步提高了泛化能力。软间隔的代表算法是 SoftBoost 算法，由于软间隔只是在硬间隔上增加了松弛变量，因此本节先介绍下硬间隔算法的思想，然后引出软间隔 SoftBoost 算法：

给定 N 个带标签样本集为 (x_n, y_n)，$n=1,2...,N$，x_n 为用向量表示的样本，y_n 为样本所属的标签。Boosting 算法在样本上维持一个样本权值分布 D，表示样本所占的权重，由于 Boosting 算法偏向于难分样本，因此对于这些样本所占的权重会大一些。在每次的迭代过程中，Boosting 算法在样本权值分布 D 上使用弱分类器进行训练后得到一个弱假设 $h: x->[-1,+1]$，为简便起见在此 h 的范围看成是二分类{+1，-1}。

弱假设 h 的错误率 $\varepsilon_h = P_D[h(x_n) \neq y_n] = \sum_{n=1}^{N} D(n)[h(x_n) \neq y_n]$，其中 π 表示 π 为真时该值为 1，π 为假时该值为 0。得到若假设 h 后，选择界限 γ_h 测量弱假设 h 在样本权值分布 D 上的重要程度 $\gamma_h = \sum_{n=1}^{N} D(n) y_n h(x_n)$。理想状况下当弱假设 h 将所有样本都分对的话，γ_h=1，都分错的话 γ_h=-1。当样本的权重都一样的时候，可以看出随机猜测的情况下 γ_h=0，从上式可以看出 γ_h 的值越大，弱假设 h 在训练样本上的分类效果越好。

经过 T 次迭代后，得到最终的强分类器 $H(x) = \sum_{t=1}^{T} w_t h_t(x)$，这里 $h_t(x)$ 表示在第 t 次迭代得到的弱假设，w_t 代表该弱假设对应的权重。在此引入间隔(margin)的定义，根据文献[33]得知，样本 (x_n, y_n) 的间隔为 $\rho_n = y_n H(x_n) = y_n \sum_{t=1}^{T} w_t h_t(x_n)$，$\rho_n$ 相当于样

本 x_n 到强假设 $H(x)$ 的距离， ρ_n 值的大小可以理解为置信度的高低，当值为负时，表示分类器将该样本分错了。如图 5-3 样本 x_1， x_2 到分类器的间隔分别为 ρ_1， ρ_2。而整个训练集的间隔 ρ 就等于所有样本的间隔值中最小的值即 $\rho=\min_{n=1}^{N}\rho_n$，最大化间隔 Boosting 算法就是要找到最大的最小间隔，即满足上述条件的情况下间隔的最大取值，以此来提高算法的泛化能力。

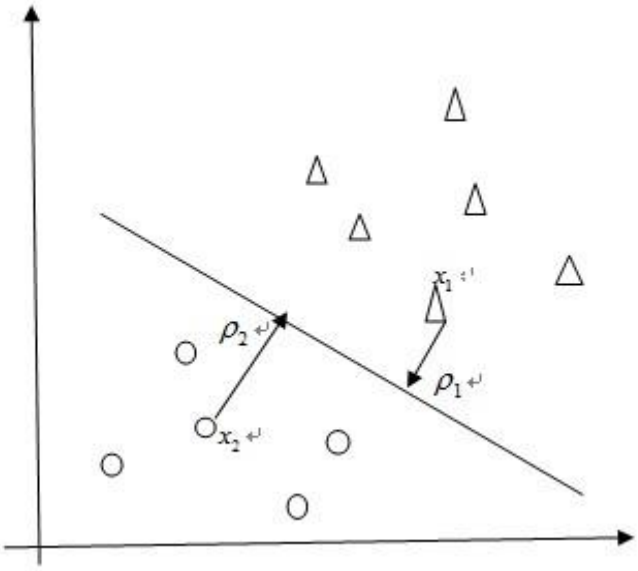

图 5-3 样本间隔

引入一个矩阵 U ,将弱分类器 $h_m(x_n)$ 与样本的标签 y_n 联系起来 $U_{n,m}=y_n h_m(x_n)$，$m=1,...,t$。将 U_n 记为矩阵 U 中的行向量，U_m 记为矩阵 U 中的列向量。引入向量 U_n 后，样本 (x_n,y_n) 的间隔改写为 $\rho_n=U_nW$，其中 $W=(w_1,w_2,...,w_t)$ 代表分类器的权重向量，则样本集的间隔为 $\rho=\min_{n=1}^{N}U_nW$。相应的关于弱假设 h_m 的界限为 $\gamma_m=DU_m$。

为了提升算法的泛化误差，即让泛化误差最大，则需要求出最大间隔，这可用组合数学中的最优化问题求解。最大化间隔 Boosting 是在满足约束条件 $w_m\geq 0,\sum_{m=1}^{t}w_m=1$ 的情况下，使得间隔 ρ 最大。该问题用最优化问题表示为如下形式（其中 T 表示的是迭代次数）：

$$\begin{aligned}&\max\rho\\ &s.t\sum_{m=1}^{t}U_{n,m}w_m\geq\rho,\ \ for\,n=1,...,N\\ &w_m\geq 0,\sum_m w_m=1\end{aligned}\tag{5-1}$$

使用组合数学中的对偶性定理，上式可以转化求最小界限γ的对偶性问题，如公式(5-2)：

$$\begin{aligned} &min\ \gamma \\ &s.t\ \sum_{n=1}^{N} U_{n,m} D(n) \le \gamma,\ \ for\ m = 1,...,t \\ &D(n) \ge 0, \sum_{n} D(n) = 1 \end{aligned} \tag{5-2}$$

上述问题是最大化硬间隔问题，即追求将所有样本都考虑进去，在样本分布正常的情况下，这种方法无疑是最好的，能够最大化间隔泛化能力很强，但是当数据集中的样本存在难分样本或者噪声样本时，如果过分关注这些样本点，会使样本集间隔ρ的值变的很小甚至会小于 0，这就使得到的强分类器会产生过拟合现象，泛化能力严重下降。为解决此问题，产生了软间隔算法 SoftBoost，软间隔算法的思想是，放松对最大化间隔的要求，允许在一定的范围内降低对难分样本或者错误样本的关注度，即允许这些样本处于间隔的另一面，以换得更强的泛化误差，但同时要加入一组松弛变量ξ_n对其进行惩罚。对应的解决公式为(其中$1/\nu$相当于一个惩罚因子，相当于计算加上松弛变量后目标函数的损失)：

$$\begin{aligned} &\max(\rho - \frac{1}{\nu} \sum_{n=1}^{N} \xi_n) \\ &s.t \sum_{m=1}^{t} U_{n,m} w_m \ge \rho - \xi_n, for\ n = 1,..N \\ &w_m \ge 0, \sum_{m} w_m = 1 \\ &\xi_n \ge 0 \end{aligned} \tag{5-3}$$

使用组合数学中的对偶性定理，上式可以转化求最小界限γ的对偶性问题，如公式(5-4)：

$$\begin{aligned} &min\ \gamma \\ &s.t \sum_{n=1}^{N} U_{n,m} D(n) \le \gamma,\ \ for\ m = 1,t \\ &0 \le D(n) \le \frac{1}{\nu} \sum_{n} D(n) = 1 \end{aligned} \tag{5-4}$$

SoftBoost 初始样本权重D^1为均匀分布，在获得本次迭代的弱假设之后，将样

本分布从D^t更新到D^{t+1}，用于下一轮迭代。一般的 Boosting 更新样本权重时只与当前的弱假设有关，而 SoftBoost 采用与 TotalBoost 算法相同的样本权重更新方法，即 SoftBoost 使用最小化相对熵的办法更新权值，公式(5-5)给出了相对熵$\Delta(D,D^1)$的定义，该方法不仅与当前得到的弱学习器有关，还与前面获得的弱学习器均有关，更新权重 D^{t+1}时应该满足的约束条件如公式(5-6)：

$$\Delta(D,D^1)=D(n)\sum_n \ln\frac{D(n)}{D^1(n)} \tag{5-5}$$

$$D^{t+1}=\arg\min_D \Delta(D,D^1)$$

$$s.t\sum_{n=1}^{N}U_{n,m}D(n)\le\gamma,\ \ for\ m=1,...,t \tag{5-6}$$

$$0\le D(n)\le\frac{1}{\nu},\sum_n D(n)=1$$

下图 5-4 中给出了 SoftBoost 算法详细的流程图。

5.2.2 将 SoftBoost 用于个性化推荐

目前大多数的协同过滤算法都是单一算法，只考虑从某一方面改善系统的推荐质量，因此适用的场合也不同。如基于 Pearson 的相似度考虑了用户的打分尺度问题，侧重于用户间共同评分的商品，这种方法一般比基于 Cosine 的相似度方法准确度高，但是当打分矩阵极度稀疏时，基于 Pearson 的方法却不如基于 Cosine 的方法；在本文的第四章，提出了一种改进的相似度算法，从流行度、共同打分商品数以及用户打分差异度三方面对 Pearson 方法进行了改进，虽然能够在一定程度上提高推荐的准确度，但该方法也只是考虑了打分矩阵的局部作用，仅从相似用户入手，通过相似用户得到预测的评分；而矩阵分解模型如 SVD，NMF 等只从打分矩阵的全局考虑，将一个打分矩阵分解为用户特征矩阵和商品特征矩阵，使用这两个矩阵预测用户对商品的打分，却没有考虑用户之间或商品之间的关系。而集成学习算法使用多个弱学习器解决同一个问题，根据最大化软间隔原理我们知道在上一小节描述的 SoftBoost 算法在存在噪声数据或难分数据时，能够降低对这些样本的关注，避免过拟合现象的发生，具有极强的泛化能力，因此使用该算法作为框架能够保证集成结果的准确性，因此本节用该算法作为框架，将一些简单的协同过滤算法作为弱学习器，将集成学习应用在个性化推荐上。

由于 SoftBoost 主要是用于解决分类的算法，前面的间隔(margin)和边界(edge)都是针对分类样本定义的，而个性化推荐是给用户提供一个商品排序列表，因此

如将 SoftBoost 算法用于个性化推荐，需要重新定义一下间隔(margin)和边界(edge)的定义。

Algorithm: *SoftBoost*

输入：$S=((x_1,y_1),(x_2,y_2),...,(x_N,y_N))$，准确率参数$\delta$，惩罚因子$\nu\in[1,N]$

初始化：D^1为均匀分布，γ=1,$t=1$

(1) *while*(t)

使用权值分布D^t，调用基分类器获得弱假设h_t

$$\gamma_t=\sum_{n=1}^{N}U_{n,t}D^t(n)=\sum_{n=1}^{N}y_nh_t(x_n)D^t(n),\ \gamma=\min(\gamma,\gamma_t)$$

更新样本分布D^{t+1}:

{

$$D^{t+1}=\arg\min_{D}\Delta(D,D^1)$$

$$s.t\sum_{n=1}^{N}U_{n,m}D(n)\le\gamma\text{-}\delta,\ for\ m=1,...,t$$

$$0\le D(n)\le\frac{1}{\nu},\sum_{n}D(n)=1$$

}

如果上述问题变为不可解或者D^t中包含有为0的元素，则$T=t-1$, *break*

$t=t+1$

endwhile

(2)输出：$H(x)=\sum_{t=1}^{T}w_th_t(x)$,其中$w_t$为弱假设的权重，使用LP中的最大化间隔法求解

图 5-4 SoftBoost 算法伪代码

针对推荐的目的是给每个用户提供一个商品推荐列表，本文将用户对应于 SoftBoost 中的样本，在一次迭代过程中，当得到一个排序器 h 后，该排序器 h 的正确度可以用基于用户权重分布 D 的边界参数 γ_h 来衡量 $\gamma_h=\sum_{u=1}^{N}D(u)C(r_u,h^u)$，其中 N 表示的是用户的数目，$D(u)$ 表示用户 u 所占的权重，r_u 为用户 u 关于商品的实际排序列表，h^u 表示弱排序器 h 给用户 u 预测的商品排序列表。$C(r_u,h^u)$ 用来衡量实际商品列表与预测商品列表的相近程度，在此使用 RankBoost 的中的基于对的排序方法衡量弱排序器 h 的性能，下面简要介绍下 RankBoost 算法。

在 RankBoost 算法中，并不关心每个文档的具体打分是多少，因为对于排序问题最重要的是给出一个正确的排序列表，只要文档间的相对位置正确，文档的具体打分值是多少则并不重要，RankBoost 是基于对的排序，该算法的思想是只要知道了两两文档间的相对关系，那么整体的排序关系就得到了。如文档对 x_i 与 x_j，

若实际情况下文档 x_i 比文档 x_j 的相关度高，文档 x_i 排在文档 x_j 的前面，那么相应的我们希望排序器给出的结果是 $h(x_i) >= h(x_j)$。RankBoost 中通过一个排序损失函数 $rlossD$ 来衡量排序器的性能，具体方式如下：

$$rlossD(H) = \sum_{x_0, x_1} D(x_0, x_1) \ H(x_1) \le H(x_0) \ = \Pr_{(x_0, x_1) \square D}\left[H(x_1) \le H(x_0)\right] \tag{5-7}$$

其中 π 表示 π 为真时该值为 1，π 为假时该值为 0，x_0 为非相关文档，x_1 为相关文档，理论上希望 x_1 的预测得分高于 x_0 即 $H(x_1) > H(x_0)$，$rlossD$ 衡量的是在样本对分布 D 上模型预测的所有关键对排序错误的程度，最终的目标是使 $rlossD$ 的值尽可能小。结合 RankBoost 基于 pairwise 的思想，本文给出应用于个性化推荐时 RankBoost 算法衡量实际商品列表与预测排序列表相近度的 $C(r_u, h^u)$ 表达式：

$$C(r_u, h^u) = K \sum_{i,j} \left[(h^u(x_i) - h^u(x_j))(r_{u,i} - r_{u,j}) >= 0 \right] \tag{5-8}$$

与 RankBoost 不同的是这里 i, j 表示商品间的任意组合，且 $i \neq j$。$h^u(x_i)$ 表示弱排序器预测的用户 u 对商品 i 的打分，相应的 $h^u(x_j)$ 表示弱排序器预测的用户 u 对商品 j 的打分，$r_{u,i}$ 表示的是打分矩阵给出的用户对商品 i 的实际打分，相应的 $r_{u,j}$ 表示的是打分矩阵给出的用户对商品 j 的实际打分。当用户 u 对商品 i 的实际打分 $r_{u,i}$ 高于对商品 j 的实际打分 $r_{u,j}$ 时，与 RankBoost 相似我们希望弱排序器的打分 $h^u(x_i) > h^u(x_j)$，仅当 $(h^u(x_i) - h^u(x_j))$ 与 $(r_{u,i} - r_{u,j})$ 使的值符号相同时，表示预测正确。用上式的目的是衡量实际推荐列表与预测推荐列表中商品的相对排序一致程度，该值越大越好。K 是为了归一化 $K = (1/2)m(m-1)$，m 表示的是商品的个数，由于一般情况下商品的数量比较庞大，而人们通常只会关注推荐列表前面的一部分，m 可以取一个 $10-50$ 之间的值。

与原先的 SoftBoost 算法相似，经过 T 次迭代后，得到最终的强预测函数 $H(x) = \sum_{t=1}^{T} w_t h_t(x)$。在此引入应用到推荐商品排序的间隔(margin)定义：用户 u 的间隔为 $\rho_u = r_u H^u = C(r_u, H^u)$，$r_u$ 为用户 u 对商品的实际排序列表，H^u 表示排序器 H 给用户 u 预测的商品排序列表，用户 u 的间隔实际上代表的是实际排序列表与预测排序列表之间的距离，整个用户集上的间隔 $\rho = \min_{u=1}^{N} \rho_u = \min_{u=1}^{N} C(r_u, H^u)$，SoftBoost 算法是要找到极大软间隔，即找到在满足一定条件时 ρ 的最大值，最大化算法的泛化能力。

为了使泛化误差最大，则需要求出最大间隔，这可用组合数学中的最优化问题求解。引入向量 U_u，将弱排序器 h_m 给出的预测排序列表与实际列表联系起来 $U_{u,m} = C(r_u, h_m^u)$。使用上式后，样本 (x_u, r_u) 的间隔改写为 $\rho_u = U_u W$，其中

$W=(w_1,w_2,...,w_t)$ 代表分类器的权重向量。则整个用户集的间隔为 $\rho=\min_{u=1}^{N} U_u W$，其中 N 表示的是用户的数目。将上述的整个思想应用到SoftBoost框架，对应的求用户集到排序器的最大化间隔的公式为（其中 ξ_n 是松弛变量，$1/\nu$ 相当于一个惩罚因子）：

$$\begin{aligned} &\max(\rho-\frac{1}{\nu}\sum_{n=1}^{N}\xi_n) \\ &s.t\ \sum_{m=1}^{t}C(r_u,h_m^u)w_m \ge \rho\text{-}\xi_n,\ \ for\,u=1,...,N \\ &w_m\ge 0,\sum_m w_m=1 \\ &\xi_n\ge 0 \end{aligned} \tag{5-9}$$

根据对偶定理，将求最大化间隔问题转换为求最小边界问题，其相应的对偶公式为：

$$\begin{aligned} &min\ \gamma \\ &s.t\ \sum_{n=1}^{N}C(r_u,h_m^u)D(u)\le\gamma,\ \ for\,m=1,...,t \\ &0\le D(u)\le\frac{1}{\nu},\sum_u D(u)=1 \end{aligned} \tag{5-10}$$

5.2.3 弱学习器的生成

SoftRankBoost 使用基于用户的相似度、基于商品的相似度、以及非负矩阵分解 NMF 协同过滤算法作为其弱排序器。其中基于用户的相似度使用 Cosine 相似度和 Pearson 相似度，近邻数目 K 取值为：5,10,20,30,40,50。基于商品的相似度与基于用户的取法相同。非负矩阵因子分解 NMF 中分解矩阵的秩的取值分别为 20，50，80,120,150,200。

另外本章使用如下公式作为使用相似度协同过滤时用户 u 对商品 i 的预测打分 $r_{u,i}$，其中 $\hat{U}$ 表示与用户 u 购买了公共商品的用户集合中与其相似度最高的前 N 个用户的集合，$\bar{r}_u$ 与 $\bar{r}_{u'}$ 分别表示用户 u 与用户 u' 的打分均值，K 为标准化因子，通常记为 $k=1/\sum_{u'\in\hat{U}} sim(u,u')$。该方法考虑了每个用户的打分习惯或打分尺度不太一样，克服了评判尺度不一致的缺点：

$$r_{u,i}=\bar{r}_u+k\sum_{u'\in\hat{U}} sim(u,u')\times(r_{u',i}-\bar{r}_{u'}) \tag{5-11}$$

此外由于使用的是 SoftBoost 集成学习算法框架，包含了多个弱排序器的生成过程，因此该算法所需的运行时间相对单一推荐算法来说较久，但在此我们主要关注推荐系统的准确度指标，不再讨论该算法的时间复杂度。

5.2.4 具体算法描述

在重新定义了应用于推荐系统的间隔和边界后，下面给出 SoftRankBoost 算法的详细伪代码。

$Algorithm: SoftRankBoost$

输入：$S=(U,I,R)$，准确率参数δ，惩罚因子$\nu \in [1,N]$

初始化：$D^1=1/|U|$为均匀分布，$\gamma=1, t=1$

(1)*while*(t)

使用权值分布D^t，获得弱假设h_t

$$\gamma_t = \sum_{u=1}^{N} U_{u,t} D^t(u) = \sum_{u=1}^{N} C(r_u, h_t^u) D^t(u), \gamma = \min(\gamma, \gamma_t)$$

更新样本分布D^{t+1}:

{

$$D^{t+1} = \arg\min_{D} \Delta(D, D^1)$$

$$s.t \sum_{u=1}^{N} C(r_u, h_m^u) D(u) \le \gamma - \delta, \quad for\ m = 1,...,t$$

$$0 \le D(n) \le \frac{1}{\nu}, \sum_n D(n) = 1$$

}

如果上述问题变为不可解或者D^t中包含有为0的元素，则$T=t-1$, *break*

$t = t+1$

endwhile

(2)输出：$H(x) = \sum_{t=1}^{T} w_t h_t(x)$,其中$w_t$为弱假设的权重，使用LP中的最大化间隔法求解

图 5-5 SoftRankBoost 伪代码

5.3 实验结果与分析

本章采用 MovieLens 数据集和 Netflix 数据集对 SoftRankBoost 算法的推荐质量进行验证，其中 MovieLens 是一个推荐系统网站，使用协同过滤技术根据用户的喜好向用户推荐电影，本章使用 MovieLens100K 数据集，其中用户数目 Users 为 943，商品数目 Items 为 1628，总的投票数为 100K，数据稀疏度为 93.5%。Netflix

是一家在线影片租赁公司, Netflix 数据集是举办 Netflix 大赛时公开的，该数据集的用户数目 Users 为 480189，商品数 Items 为 17770，大约包含了 100M 个评分信息，评分值与 MovieLens 一样为 1-5 分。本章试验时随机抽取了一部分 Netflix 数据集，用户数目 Users 为 2000，商品数目 Items 为 3000，总的投票数为 139471，数据稀疏度为 97.7%。采用五次交叉验证方法，分别将这两个数据集随机的分成不相交的五等份，每次选取一个作为测试集，其余四份为训练集，最终的结果取五次实验结果的平均值。

本章选择的对比算法为下面四个：

1) 第四章提出的改进协同过滤算法 PCS

2) 非负矩阵因子化 NMF[50]

3) 基于用户的余弦相似度 Cosine

4) 物质扩散算法 Prob-S[64]

5.3.1 基于 TopN 的结果分析

本节是关于 SoftRankBoost 算法与上面的其它四种算法在基于 TopN 推荐列表的评价，与第四章评价 PCS 算法一样，评价时并没有将所有推荐的商品都考虑进去，而是根据用户浏览推荐商品的喜好，只选取了位于推荐列表前面的商品，选取的推荐列表的长度分别为：5 ,10, 20, 40, 50 , 80 ,100 ,130, 150, 200。

图 5-6，图 5-7 和图 5-8 是在 MovieLens 数据集上测试的五种算法的准确度、召回率和 F-measure 指标，可以看出对准确率来说，随着推荐列表长度的增加，五种算法的准确率都会下降。但可以看出 SoftRankBoost 的准确率最高，Prob-S 和 PCS 算法相差不大，其次是 Cosine，而非负矩阵因子化算法 NMF 效果最差。一般来说 NMF 的推荐效果至少应该优于 Cosine 算法，但是每种算法适用的推荐系统是不同的，NMF 算法在 RMSE 评价指标上效果比传统的相似度算法要好，但在 TopN 评价指标上就差很多。对于召回率来说，在推荐列表长度接近 5 的情况下，SoftRankBoost，Prob-S，PCS 和 Cosine 相差都不大，但随着推荐列表的增长，距离在逐渐增大，SoftRankBoost 明显优于其它算法。在 F-measure 指标上，SoftRankBoost 算法也是明显优于其它算法。综合准确率、召回率和 F-measure 三个指标可以看出，五种算法的性能好坏为：SoftRankBoost 最好，其次是 Prob-S 和 PCS，然后是 Cosine，NMF 效果最差。

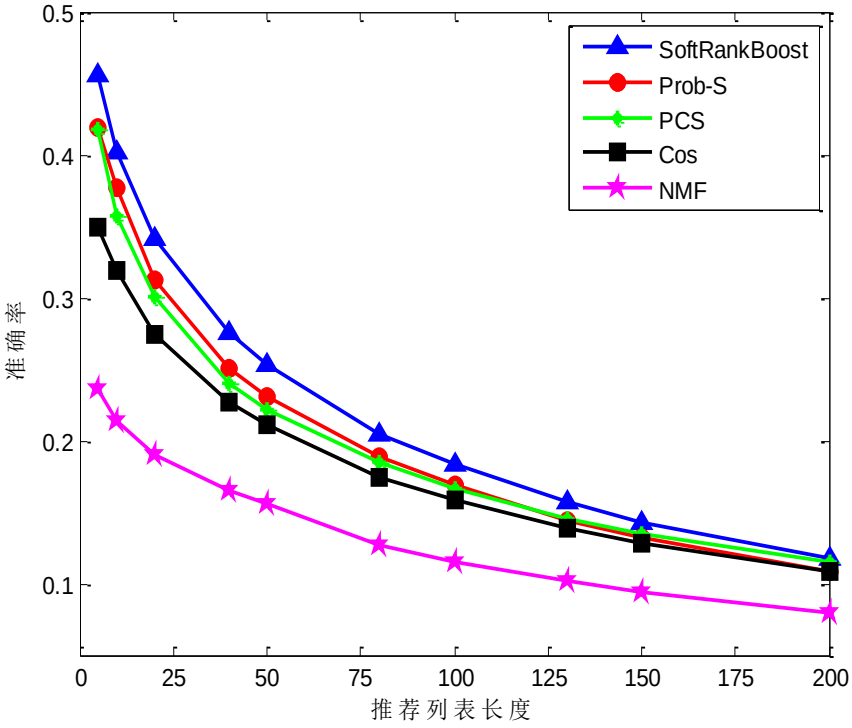

图 5-6 五种算法在 MovieLens 数据集上的准确率比较

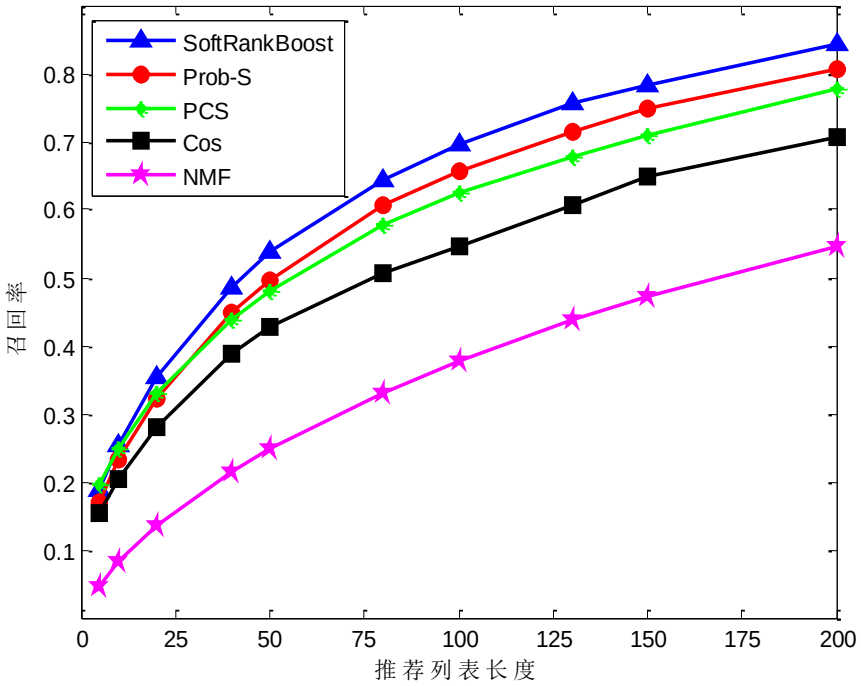

图 5-7 五种算法在 MovieLens 数据集上的召回率比较

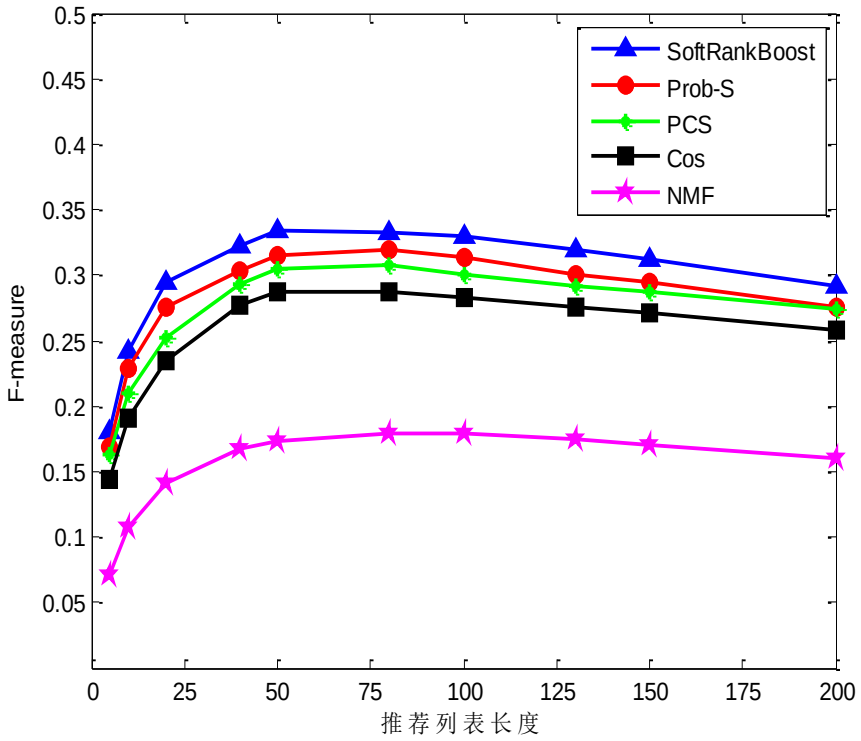

图 5-8 五种算法在 MovieLens 数据集上的 F 值比较

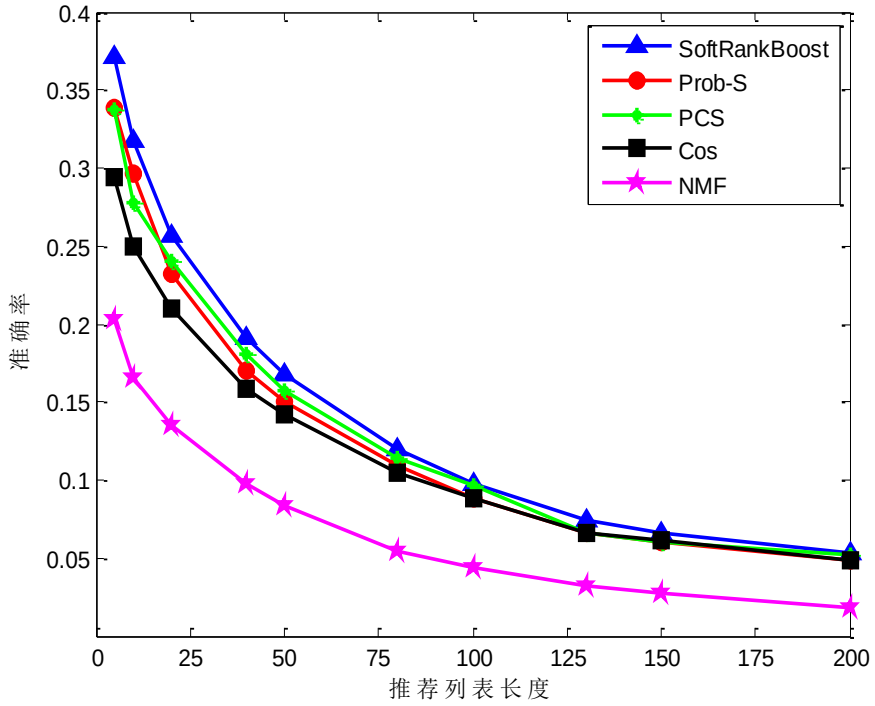

图 5-9 五种算法在 Netflix 数据集上的准确率比较

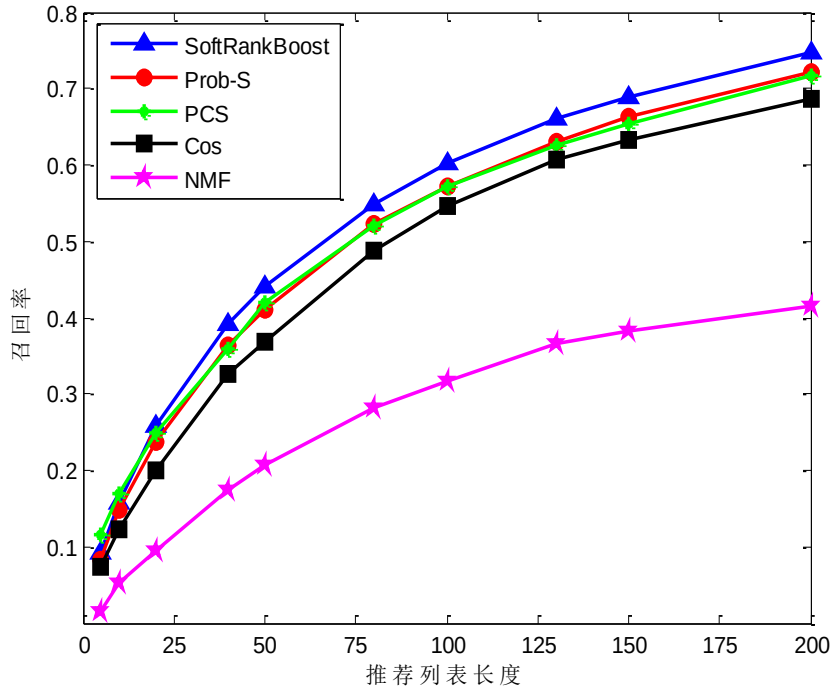

图 5-10 五种算法在 Netflix 数据集上的召回率比较

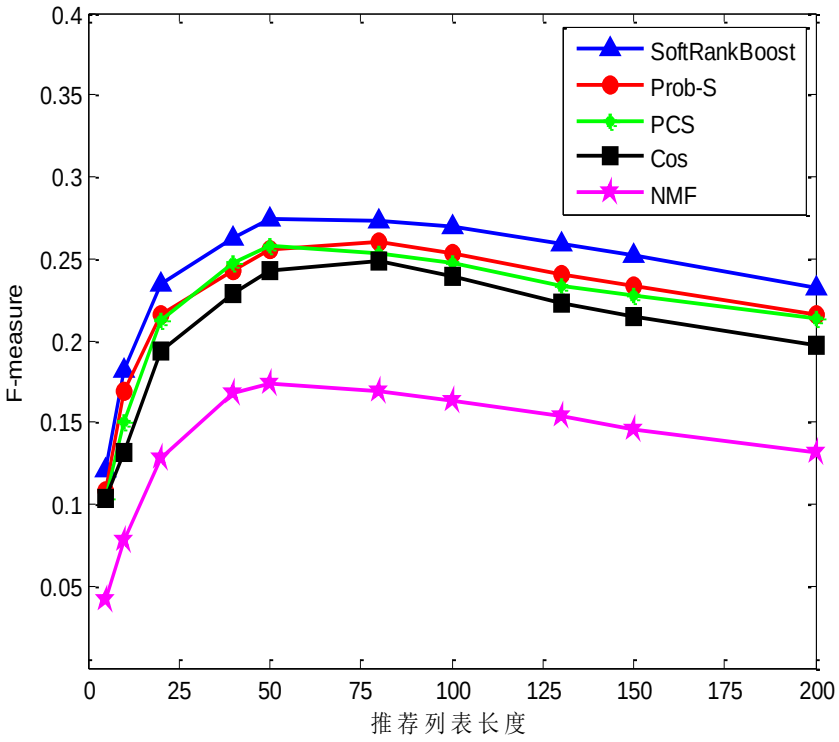

图 5-11 五种算法在 Netflix 数据集上的 F 值比较

图 5-9，图 5-10 和图 5-11 是在 Netflix 数据集上测试的五种算法的准确率、召回率和 F-measure。实验结果跟 MovieLens 上大体一致，效果最好的仍然是 SoftRankBoost 算法，但可以看出五种算法在三个指标上的整体效果都不如 MovieLens 好，原因可能是 Netflix 比 MovieLens 还要稀疏造成的。此外可以看出在准确率上，PCS 稍稍高于 Prob-S，这是因为 PCS 从流行度、共同打分商品数、打分差异度三方面对 Pearson 进行了改进，使得矩阵稀疏性对它的影响减小，相比于其它算法，准确率影响小一点。

5.3.2 基于 NDCG 评价指标的结果分析

由于 SoftRankBoost 采用了排序的思想，给出的是商品的推荐列表，本节根据第三章 3.3.2 节介绍的信息检索中的 NDCG 评价指标进行评价。$NDCG_p$ 中的 p 的取值为 5,10,20,30。表 5-1 是 MovieLens 数据集上的实验结果，表 5-2 是在 Netflix 数据集上的结果，可以看出两个数据集中 SoftRankBoost 算法的效果都是最好的。依次是 Prob-S ，PCS ，Cos ，NMF。NMF 效果最差在上一节已经分析过，是因为 NMF 算法在基于预测评分的评价指标上的效果比较好，而在基于列表的评价指标上效果很差。

表 5-1 五种算法在 MovieLens 上的 NDCG 指标比较

	SoftRankBoost	Prob-S	PCS	Cos	NMF
$NDCG_5$	0.355	0.326	0.326	0.326	0.168
$NDCG_{10}$	0.338	0.329	0.321	0.315	0.16
$NDCG_{20}$	0.356	0.341	0.332	0.324	0.163
$NDCG_{30}$	0.376	0.369	0.351	0.341	0.169

表 5-2 五种算法在 Netflix 上的 NDCG 指标比较

	SoftRankBoost	Prob-S	PCS	Cos	NMF
$NDCG_5$	.0.22	0.20	0.20	0.197	0.15
$NDCG_{10}$	0.245	0.218	0.215	0.214	0..16
$NDCG_{20}$	0.287	0.24	0.238	0.235	0.176
$NDCG_{30}$	0.32	0.275	0.273	0.261	0.19

5.4 本章小结

本章介绍了集成学习最大化软间隔算法 SoftBoost,利用它的强泛化能力和有效避免过拟合的优点，将其用于个性化推荐领域，并结合 RankBoost 基于对的思想，重新定义了 SoftBoost 中的间隔和边界，提出了最终的 SoftRankBoost 算法。该算法使用基于相似度的 CF 算法和非负矩阵因子分解算法 NMF 为弱排序器，并在本章的 5.3 节从 TopN 和 NDCG 两个评价指标验证了该算法能有效提高推荐的准确率。

第六章 总结和展望

6.1 本文工作总结

互联网的飞速发展，导致了信息爆炸，各大电子商务网站要想保持竞争力站稳脚步，必须能够吸引和保持自己的消费群体，由此个性化推荐的性能就非常重要。早期的基于用户和基于商品的协同过滤技术获得了广泛应用，但随着电子商务网站规模的扩大，用户数量和商品种类曾指数增长趋势，打分矩阵非常稀疏，这就使得推荐的质量严重下降。

集成学习技术集成多个弱学习器解决同一个问题，有很好的泛化能力，将其与协同过滤技术相结合，应用在个性化推荐上，提高了个性化推荐的质量

本文的主要研究工作包括以下几个方面：

(1) 介绍了集成学习和个性化推荐近几年国内外的研究现状，分析了集成学习的优点及其在信息检索和个性化推荐中的应用。指出了当前个性化推荐技术的优点和不足之处，给出了用于评价个性化推荐系统质量的主要评价指标。

(2) 介绍了当前协同过滤技术中常用的几种相似度计算方法，如余弦相似度、Pearson 相似度以及修正的余弦相似度，分析了这三种相似度计算方式的缺点，然后从商品流行度，用户间共同评分的商品数目，以及评分间的差异三个方面对现有相似度进行了改进， 提出了一种改进的相似度计算方式。此外由于相似度的计算时间复杂度比较大，为了提高在线系统的可扩展性，将决策树用于用户的分组上，使用决策树的自适应性根据用户对商品的反馈一步步将其分组。

(3) 详细介绍了最大化软间隔算法 SoftBoost,由于 SoftBoost 引入了松弛变量降低了对难分样本点的关注，避免过拟合现象，具有非常好的泛化性能。因此本文将协同过滤算法作为弱学习算法，与 SoftBoost 相结合，将其用于个性化推荐，同时本文将 RankBoost 基于对的思想应用进来，提出了 SoftRankBoost 算法。

6.2 工作展望

本文提出了一种改进的相似度计算方法，提高了推荐的准确率。另外本文将集成学习最大化软间隔算法 SoftBoost 与个性化推荐技术相结合进一步提高了推荐

的准确率。但本文仍有待改进：

(1) 衡量个性化推荐的指标有很多，虽然个性化推荐的准确率很重要，但追求个性化推荐的多样性、新奇性也很重要，一个好的推荐系统应该平衡各方面指标。

(2) 本文只是从打分矩阵的角度入手，如能将基于内容的推荐算法和协同过滤相结合，则能大大缓解数据稀疏性问题和新用户问题。

(3) 用户的兴趣和爱好随着时间是会改变的，但是本文并没有把时间因素也考虑进去，下一步可以从时间因素对协同过滤算法进行改进。

致 谢

三年的研究生生活转眼就要结束了，在这里我向研究生期间给予我帮助和关心的各位老师和同学表示深深的感谢。

最要感谢的是我的导师傅彦教授，研究生期间傅老师对我的成长给予了很多机会和很大鼓励。她经常教导我们做任何事都要认真谨慎、踏实苦干，每次的实验室活动中，更让我感受到了团队的重要性，同时让我明白只有充满激情、努力奋斗才能把事情做好。

特别要感谢周俊临老师和董强老师，他们为我的毕业论文选题和研究过程都提了很多宝贵的建议，工作上他们认真负责，生活中他们充满风趣，和我们像朋友一样玩耍。感谢高辉老师、陈端兵老师、刘震老师，他们严谨的治学态度以及对生活乐观向上的态度深深影响了我。感谢方育柯博士对我论文的指导和在生活当中对我的极大帮助。

感谢那些陪伴我一起成长，一起吃喝玩乐的同学和朋友们，没有你们的陪伴生活将是多么单调，遇到困难时没有你们的鼓励和开导，将是多么的无助和痛苦。最后感谢我的父亲和母亲，你们抚养我长大，供我读书上学，没有你们的支持就没有今天的我。

参考文献

[1] F Ricci , R Lior ,S Bracha.Introduction to Recommender Systems Handbook,Recommender Systems Handbook[M]. Springer, 2011,1-35

[2] 项亮.推荐系统实战[M].人民邮电出版社,2012

[3] 吴颜,沈洁,顾天竺,等.协同过滤推荐系统中数据稀疏问题的解决[J].计算机应用研究, 2007

[4] G. Linden, B. Smith, J. York. Amazon. com recommendations: item-to-item collaborative filtering[J].IEEE Internet Computing, 2003, 7(1):76-80

[5] T G Dietterich.Machine Learning Research:Four Current Directions[J].AI Magzine,1997, 18(4):97-136

[6] M Sewell.Ensemble Learning[M]. UCL Department of Computer Science ,2011,1-10

[7] 方育柯.集成学习理论研究及其在个性化推荐中的应用[D].成都:电子科技大学,2011

[8] R Polikar, A Topalis, D Parikh, et al.An ensemble based data fusion approach for early diagnosis of Alzheimer's disease[J].Information Fusion,2008,9(1):83-95

[9] R E Schapire.The strength of weak learnability[J].Machine Learning,1990,5(2):197-227

[10] L Xu, A Krzyzak, C Y Suen.Methods of combining multiple classifiers and their applications to handwriting recognition[J].IEEE Transaction on systems, Man and Cybernetics,22(3),418-435

[11] R Battiti, A. M Colla. Democracy in neural nets: Voting schemes for classification[J],Neural Networks,1994, 7(4):691–707

[12] Y Shinmshoni, N Intrator. Classification of seismic signals by integrating ensembles of neural networks[J].IEEE Transactions Signal Processing, 1998,46(5): 1194-1201

[13]G. Ridgeway. Looking for lumps: boosting and bagging for density estimation[J]. Computational Statistics and Data Analysis, 2002, 38(4):379-392

[14] C Leistner, A Saffari, P M Roth, et al.On robustness of on-line boosting - a competitive study[C], Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on.IEEE,2009,1362–1369

[15] M F Tsai, S T Chen, C S Ferng ,et al. An ensemble ranking solution to the yahoo! learning to rank challenge[J].2010, National Taiwan University, Taipei 106

[16] Z H Zhou, J X Wu, W Tang ,et.al.Combining regression estimators:GA-based selective neural network ensemble[J].Internationnal Journal of Computational Intelligence and Applications,2001,1(4):341-356

[17] Z H Zhou, F H Huang.Pose invariant face recognition[J]. The fourth IEEE International Conference on Automatic Face and Gesture Recogition,Grenoble, Fance.2000,245-250

[18] 刁力力, 胡可云. 用 Boosting 方法组合增强 stumps 进行文本分类[N]. 软件学报, 2002, 13(8):1391-1367

[19] 袁时金 ,李荣陆 ,周水庚.层次化中文文档分类[J]. 通信学报, 2004, 11 期

[20] R Armstrong , D Freitag , T Joachims , et. al .WebWatcher: A Learning Apprentice for the World Wide Web[J]. School of Computer Scienc,1995

[21] H Lieberman.Letizia: An Agent That Assists Web Browsing[C]. International Joint Conference on Artificial Intelligence - IJCAI , 1995,924-929

[22] P Resnick, N Iacovou, M Suchak,et.al. "GroupLens: an open architecture for collaborative filtering of netnews"[C].Proceedings of the 1994 International ACM Conference on Computer Supported Cooperative Work. Computer Supported Cooperative Work. ACM Press. 1994,175–186.

[23] V Jesse, S Shilad, J Riedl. "Tagsplanations: Explaining Recommendations using Tags" [C]. Proceedings of the 13th international conference on Intelligent user interfaces. International Conference on Intelligent User Interfaces. ACM Press. 2009,47–53

[24] L.K Hansen, P Salamon.Neural network ensembles[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence,1990,12(10):993–1001

[25] Z H Zhou.Ensemble Learning[C]. In: S. Z. Lied.Encyclopedia of Biometrics, Berlin: Springer, 2009,270-273

[26] T G. Dietterich.Ensemble Methods in Machine Learning[C].International Workshop on Multiple Classifier Systems. 2000

[27] L Breiman. Bagging predictors[J].Machine Learning,1996,24(2): 123–140

[28] R E. Schapire . A Brief Introduction to Boosting[C]. Proceeding IJCAI'99 Proceedings of the 16th international joint conference on Artificial intelligence . 1999,2: 1401-1406

[29] Y Preund , R E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting[J]. Journal of Computer and System Sciences, 1997,55(1): 119-139

[30] L Breiman. Arcing classifiers[J]. The Annals of Statistics.1998,26(3):801-849

[31] H Drucker, C Cortes. Boosting decision trees[J]. In Advances in Neural Information Processing Systems , 1996,8: 479-485

[32] J. R. Quinlan. Bagging, boosting, and C4.5[C]. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, 1996, 725-730

[33] R E. Schapire, Y Freund, P Bartlett,et.al. Boosting the margin: A new explanation for the effectiveness of voting methods. The Annals of Statistics, 1998, 26(5):1651-1686

[34] 梅田望夫.网络剧变元年——你必须参加的大未来[M].先觉出版社，2006

[35] 刘建国，周涛，汪秉宏.个性化推荐系统的研究进展[J].自然科学进展，2009,19(1)：1-15

[36] Y H Cho, J K Kim, S H Kim .A personalized recommender system based on web usage mining and decision tree induction[J]. Expert Systems with Applications. 2002 , 23:329–342

[37] K Raymond, H Blockeel "Web Mining Research: A Survey".SIGKDD Explorations, 2000,2(1):1-10

[38] J B Schafer, J Konstan, J Riedl. Recommender systems in e-commerce[C]. The 1st ACM Conference on Electronic Commerce . New York, ACM Press, 1999,158− 166

[39]B Sarwar, G Karypis,J Konstan,et.al.Application of Dimensionality Reduction in Recommender System -A Case Study[R].GroupLens Research Group/Army HPC Research Center ,2000

[40] M Balabanovic, Y Shoham.Fab: Content-Based, collaborative recommendation Communica-. tions of the ACM[J]. 1997,40(3):66 − 72

[41] G. Adomavicius, A Tuzhilin.Toward the Next Generation of Recommender Systems:A Survey of the State-of-the-Art and Possible Extensions[J]. IEEE Transactions onKnowledge and Data Engineering ,2005,17: 734-749

[42] J.L. Herlocker, J.A. Konstan, L.G. Terveen,et.al. Evaluating collaborative filtering recommender systems[J]. ACM Transactions on Information Systems,2004,22:5-53

[43] J.L. Herlocker, J.A. Konstan, A. Borchers,et.al. An algorithmic framework for performing collaborative filtering[C]. The 22nd annual international ACM SIGIR conference on Research and development in information retrieval, ACM Press, New York, 1999, 230-237

[44] F. Cacheda, V. Carneiro, D. Fern. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable,high-performance recommender systems[R]. ACM Transactions on Web ,2011

[45] B Sarwar,G Karypis,J Konstan，et a1.Item based collaborative filtering recommendation algorithms[C].The 10th Int’1 WWW Conf,Hong Kong,2001,1-5

[46] J Davidson, B Liebald, J Liu, et al. The YouTube video recommendation system[C]. In Proceedings of the fourth ACM conference on Recommender systems , ACM, New York, NY, 2010, 293-296

[47] M Deshpande,G Karypis.Item based topN recommendation algorithms[J].ACM Trans Information Systems,2004,22(1):143- 177

[48] L H Ungar,D P Foster.Clustering methods for collaborative filtering[C].Recommender Systems,Papers from 1998 Workshop,Technical Report WS-98-08,Menlo Park,1 998:84- 88

[49] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Application of Dimensionality Reduction in Recommender System-A Case Study[J].Computer and Information Science,2000, 1625(1):264-8

[50]Ming-Rui Wu.Collaborative Filtering via Ensemble of Matrix Factorizations[C]. KDD Cup and Workshop at the 13th ACM SIGKDD Conference.2007

[51] TQ Lee, Y Park, Y T Park.A time-based approach to effective recommender systems using implicit feedback[J].Expert Systems with applications,2008,34(4):3055-3062

[52] G Salton ,A Wong ,C S Yang.A vector space model for automatic indexing[J]. Communications of the ACM, 1975,18(11): 613-620

[53] H C Wu, R W P Luk, K F Wong, et.al. "Interpreting tf–idf term weights as making relevance decisions"[C]. ACM Transactions on Information Systems ,2008,26 (3): 1–37

[54] G. Adomavicius, A Tuzhilin. "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions"[C]. IEEE Transactions on Knowledge and Data Engineering. 2005,17 (6): 734–749

[55] "BellKor's Pragmatic Chaos". 2009

[56] J L HERLOCKER, J A KONSTAN, A BORCHERS, et a1. An algorithmic framework for performing collaborative filtering[C]. Proceedings of the 22nd International Conference on Research and Development in InformationRetrieval (SIGIR'99). New York:ACM Press, 1999: 230-237

[57] M BALABANOVIC, Y SHOHAM .Fab: content-based collaborative recommendation[J]. Comm ACM , 1997, 40(3): 66-72

[58] TQ Lee，Y Park，YT Park．A time—based approach to effective recommender systems using implicit feedback[J].Expert Systems with Applications,2008,34(4):3055-306

[59] J S Breese，D Heckerman，C Kadie．Empirical analysis of predictive algorithms for collaborative filtering[C].The 14th Conf Uncertainty in Artificial Intelligence Madison,1998,43-52

[60] S Badrual ,K George,K Joseph,et.al.Analysis of Recommendation Algorithms for E-Commerce[C].In Proceeding of ACMEC'00 Conference.2000,158-167

[61] L H Ungar,D P Foster.Clustering methods for collaborative filtering[C].Recommender Systems,Papers from 1998 Workshop,Technical Report WS-98-08,Menlo Park,1 998,84-88

[62] B Sarwar, G Karypis, J Konstan, et.al. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering[C]. The fifth International Conference on computer and Information Technology,2002

[63] H C Yoon, K K Jae, H K Soung. A personalized recommender system based on web usage mining and decision tree induction[J]. Expert Systems with Applications ,2002,23: 329–342

[64] T. Zhou, Z. Kuscsik, J.-G. Liu et al. Solving the apparent diversity-accuracy dilemma of recommender systems. Proceedings of the National Academy of Sciences, 2010,107(10):4511–4515

攻硕期间取得的研究成果

[1] 基于决策树的协同过滤改进算法.电子科技大学计算机科学与工程学院硕士生学术论坛

[2] 智能短信分类. 横向项目

[3] 办公系统. 横向项目