中国科学技术大学

硕士学位论文

基于用户物品成对分块的协同过滤算法

作者姓名：吴建军

学科专业：计算机软件与理论

导师姓名：唐珂 教授

完成时间：二〇一四年四月

University of Science and Technology of China

A dissertation for master's degree

Collaborative Filtering Based on User-item Pairwise Blocking

Author :	Jianjun Wu
Speciality :	computer software and theory
Supervisor :	Prof. Ke Tang
Finished Time :	4, 2014

基于用户物品成对分块的协同过滤算法

二系　吴建军　中国科学技术大学

中国科学技术大学学位论文原创性声明

本人声明所呈交的学位论文, 是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

作者签名：_____ 签字日期：_____

中国科学技术大学学位论文授权使用声明

作为申请学位的条件之一，学位论文著作权拥有者授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定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，可以将学位论文编入《中国学位论文全文数据库》等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。本人提交的电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

☐ 公开 ☐ 保密 _____ 年

作者签名：_____ 导师签名：_____

签字日期：_____ 签字日期：_____

摘 要

据我们所知，几乎所有现存的协同过滤都需在某个特征空间上用某种相似度来发现所谓的物品之间或者用户之间的邻居关系。绝大多数协同过滤算法在寻找邻居的时候都隐含一个假设，那就是两个用户之间的相似或者邻近关系在面对不同的物品的时候是静止不变的。但是在现实中，却不是这样的，两个用户在一些物品上很相似，而在另一些物品上可能截然不同。在面对一个物品集时，根据对这个物品集的看法用户可以聚成多个类，然而在面对一个新的物品集时，先前的用户类结构可能就会坍塌，根据用户对新物品集的看法再次形成一个新的类结构。类似地，物品根据在某个用户集中的受欢迎程度形成的类结构在面对新的用户集的时候也有可能分裂重组。总而言之，用户类结构在面对不同物品时会不停地变化，反之，物品的类结构在面对不同的用户的时候也会不停地变化。在这篇论文中，我们试图去发掘隐藏在评分矩阵中的块结构。我们用一个块结构来建立用户类和物品类以及刻画他们之间的相互作用，每个块由许多用户和物品组成，在这样的每个块中，用户对这个块中的物品持有很相似的看法，而物品在这个块的用户集中也有着类似的受欢迎程度。每一个块都是一个富有深意的被用户和产品共享的结构，反映了在现实中用户兴趣和物品属性之间的切合程度以及用户类和物品类对彼此形成的影响。我们提出了用户物品成对分块算法的一般框架，这个框架包含用户和物品统一特征提取，用户和物品之间的对齐，物品或者用户基于对齐表示进行聚类并得到我们期望的块结构，最后我们利用已有的协同过滤算法进行全局和块内学习并给出未知评分的预测。我们给出一种这种框架的具体实现，并通过模拟实验表明这种方法可以有效地提升推荐算法的性能。

关键词： 协同过滤，动态相似，矩阵分块

ABSTRACT

To our knowledge, almost all existing collaborative filtering need to find elusive neighbouring relationship between users or between items based on some similarity measure in some space. However, a hypothesis behind most previous works is that neighbouring or similar relationship of users would be static across the whole set of items, which is not true in the reality. Two user who share similar taste on some items may have totally different opinion on others. Users maybe clustering into many groups in terms of their opinion on a set of items, these groups would collapse and new users'cluster structure would be built in terms of their opinion on a new set of items. Analogously, clusters of items formed according to their popularity among a group of users would be disintegrated when encounter a new group of users. In a nutshell, users'cluster structure would vary across the set of items, and vice versa, cluster'structure of items would vary across the set of users. In this paper, we strive to find block structure embedded in the ratings matrix. We use block structure to build the clusters of users and of items and characterize the interaction between users and items. Every block consists of many users and items, all users involved in some block share similar opinions on all of the items in the same block, on the other hand, in the same block, all items share similar popularity among the users. Every block is an meaningful colony resided by users and items, which reflecting the extent of coalescent of users'taste with the latent attributes of items and the interaction between the groups of users and of items. we suggest a general framework for collaborative filtering based on user-item pairwise blocking, which involves the unified feature extraction of users and items, the alignment of users to items, and clustering the item with the alignment representation, so to obtain the block structure we anticipate. At last, we adopt existing collaborative to learn the latent factor on the global level and on the block level. We develop a implementation of this framework and experimental evidences show that, with utilization of these taste blocks, performance of recommendation can be improved.

Keywords: collaborative filtering, dynamically similar, matrix blocking

目 录

表格索引

插图索引

算法索引

第一章　引言

1.1　推荐系统与协同过滤

随着互联网技术的发展，我们可以获得的信息量迅速增长，例如一个普通的电影网站的电影总数可能上万，一个新闻门户网站的新闻总数可达百万条。如何从这样的海量信息中得到我们想要的那部分信息成为一个关键问题。信息的海量涌现使得我们不可能将他们全部浏览一遍，导致无法真正获得我们感兴趣的那部分信息，信息的利用率反而下降，这就是所谓的信息过载问题。传统的搜索算法只能呈现给所有的用户一样的排序结果，无法针对不同用户的兴趣爱好提供相应的服务，而且在很多情况下用户其实并不明确自己的需要，或者他们的需求很难用简单的关键字来表述，这就导致了个性化推荐系统的出现。个性化推荐系统就是要根据历史已有的用户兴趣特点和购买行为方面的信息去评估那些他没有看过的物品，从而推断用户是否会对这些物品感兴趣。在个性化推荐系统的导航下，用户可以畅游海量信息，在其中发现自己未知的兴趣，享受为其量身定制的信息服务。个性化推荐技术可以可以用于推荐多种多样的物品，比如影视网站的电影推荐，购书网站的图书推荐，门户网站的新闻推荐，社交网络中的朋友推荐，旅游网站中线路推荐等等。个性化推荐系统已经成为众多商家争相发展的一种商业营销技术。一个好的推荐系统可以有效地吸引用户，加强用户的忠诚度，为商家带来丰厚的利润。目前，几乎所有的大型电子商务系统，都不同程度的使用了各种形式的推荐系统。成功的推荐系统会带来巨大的效益。

推荐系统中的核心 -推荐算法经过多年的发展，到目前大致可以分为两类，一类是基于内容的 (content-based)，另一类是协同过滤 (collaborative filtering)。基于内容的推荐是信息过滤技术的延续与发展，这类推荐算法首先根据领域知识去刻画物品的特征，比如文档的 TF-IDF 向量，然后根据历史信息为每个用户构建一个喜欢与否的二分类模型，比如决策树，每个节点就是物品的某个特征属性。新物品将根据这个分类模型计算自己的类别进而估计每个用户对自己的喜好。这种方法需要对物品做出详细的描述，但是在很多时候，物品是很难被恰当描述的，也很难用一种统一的格式描述性质差异很大的物品。在过去十多年的时间里，协同过滤已经成为推荐算法中最流行的一种技术。这种技术只根据用户与物品在过去的互动信息来预测用户可能感兴趣的物品，由于与领域知识无关，加之其有效性和易于部署实施的特点，协同过滤引起了极大的研究热情，并被广泛应用在实际的商业系统中。这种方法根据过去用户和物品之间的互动信息进行预测，常常输入一个存在大量缺失值的评分矩阵，其中记录了已有的每个用户对每个物品的评分，然后采用各种数据挖掘方法，最后给每个用户推荐一张他最有可能感兴趣的物品列表。

然而，协同过滤技术也面临一系列挑战，还有相当的改进空间。推荐中遇到的数据基本都是高维稀疏的，要从这样的数据中提取有用信息，主要困难有：

1 数据高维稀疏性。现在待处理的推荐系统规模越来越大，用户和物品数目动辄百千万计。用户可能只是对中几件物品进行了评分，一件冷门物品可能只被很少的用户评价过。评分表中存在大量缺失值。数据非常稀疏，往往稀疏度达 99%。

2 可扩展问题。因为用户和物品的数量都非常巨大，进行物品或者用户的初始表示时维数将很高。进行近邻搜索时，候选的相似对象集合非常巨大而真正相似的对象仅仅是极小的一部分，导致搜索时间代价很大从而效率很低，限制在线推荐速度。而且随着时间的推移和网站业务的扩大，算法的参数空间和模型的大小都将快速增长，模型更新过程将愈加耗时。

3 冷启动问题。在有些具体场景中，用户或者物品的更新非常频繁，比如新闻推荐中，新闻的时效性决定了新闻集合在不停地变化。新用户因为罕有可以利用的历史信息，很难给出精确的推荐。反过来，新商品由于被评价次数很少，也难以找到合适的办法推荐给用户。

4 随时间动态变化。用户的偏好可能会随着时间变化，商品的受欢迎程度也会随时间变化，而且这种变化有的是周期性，有的是短暂突变的，有着的长期渐变的。要准确及时捕捉每个用户和每个物品的这种变化也是一个难题。

此外，还存在结果的可解释性，系统的安全性，用户行为习惯的隐私性等等问题 [1]。

	item1	item2	item3	...	...	...	item...
user1	5	1	3			1	2
item2		2		...	...		5
...	1		3		...		5
item...				5	4	4	

图 1.1: 评分矩阵示意图

本文主要关注利用用户物品评分矩阵 (图1.1) 进行协同过滤推荐中的数据高维稀疏性问题。评分矩阵是一张二维表，表示用户对物品的显式评价，其中元素可以是取自某个实数区间，比如 $1-5$ 的评分，也可以是二元变量，表示曾经买过，或者表示浏览过。我们试图给出一种新颖的方法用于在稀疏的空间上提取有意义的用户物品共存的块结构来改进协同过滤算法的性能。

1.2　协同过滤的研究现状

协同过滤被首次提出到现在已有 22 年，先后出现了多次研究热潮 (2000 年左右，2007 年左右以及目前)，涌现了许多成果，直到现在，协同过滤依然是学术界和工业界的热点问题。本节就协同过滤中主要的工作做一些介绍。

协同过滤技术可以分为三种，基于内存的 (memory-based), 基于模型的 (model-based) 以及混合方法。

基于内存的的推荐思想最早由 [2] 在 1992 年提出。这种方法的核心思想是通过某种相似度量去寻找类似的多个用户或商品，把相似的用户或产品的评分进行加权平均作为未知评分的估计。它可以细分为基于用户 (user-based) 的和基于产品的 (item-based)。[3] 提出利用用户之间的相似关系来推荐给给目标用户一些流行的产品。找到兴趣相似的用户，通过将相似用户喜欢的物品推荐给目标用户。这个算法往往推荐出来的物品很多是大家都喜欢的物品。[4] 基于物品之间的相似度来做推荐，这样的算法对商品比较稳定的场合很有效。物品之间的相似性远不如用户之间的相似性那么容易变化，如果用户购买过与备荐物品相似的物品，那么他很有可能购买备荐物品。[5] 将基于用户的和基于产品的预测线性地组合起来作为最终的预测评分。相似性度量对于这类方法来说非常重要。两个用户之间的相似性常常通过计算评分矩阵相应两行之间的相似性来度量，比如 PCC(Pearson correlation coefficient)[6] 或者 VSS(vector cosine similarity)[7]。而两个物品之间的相似度则是通过比较评分矩阵相应的两列来得到。[4] 提出用修正的余弦相似度来计算两个物品之间的相似度，这种相似度考虑了不同用户的评分尺度对物品间相似性度量的干扰。而 [8] 采用条件概率，度量两个物品出现的概率依赖大小来计算两个物品之间的相似性。最近，[9] 提出两个用户之间的相似度在每个物品上应该不相同，作者通过为两个用户共同评价过的物品分配一个与备荐物品相关的权重来计算两个用户在备荐物品上的相似度。大量实验表明 PCC 相似度在很多情况下都是最好的，

基于内存的协同过滤直接基于已有评分度量每对用户或每对物品之间的相似度，还需要从所有物品或用户中选择相似度高的邻居，从而容易遭遇稀疏性和可扩展性问题。因为用户或者物品的数量可能巨大，物品对或者用户对的数目更是平方地增长，导致相似度的计算非常耗时，而相似用户或者相似物品的搜索需要在整个用户集和物品集中逐一比较，导致可扩展性很差。另外基于内存的协同过滤计算每两个用户在整个物品集上的相似度，因为用户往往只评价物品全集中的很小一部分，两个用户都评价过的物品集合往往更小，从而这种用在很小样本集上的相似度估计很大空间上的相似度的方法存在严重问题，于是便出现了基于模型的协同过滤。基于模型的方法通过采用机器学习的方法来建立一个模型，后续再用这个模型进行预测。聚类技术常常用在基于模型的方法中以便减少候选相似对象集合的大小，从而改善算法的可扩展性。为了更加

准确地度量用户之间或者物品之间的相似度，一些基于模型的方法采用降维和特征提取的方法重新表示用户和物品并基于新的特征表示再进行相似性度量。基于矩阵分解的模型则采用矩阵低似思想试图填补评分矩阵中的缺失值。

聚类技术常常被协同过滤用来应对可扩展性问题。把每个用户或者物品表示成评分矩阵中的相应的行向量或列向量，然后采用各种通用的聚类方法，比如 k-means 家族中的算法，ROCK 等等，对用户或物品进行聚类，[10] 对用户进行聚类划分，而 [11] 则对物品进行聚类划分。在完成聚类划分后，属于同一个用户类的用户或者属于同一个物品类的物品相互才是候选邻居，从而减少了邻居的搜索空间，最后预测在每个类中独立地进行。ClustKNN[12] 首先对用户进行二分 k-means 聚类，然后把每个类的均值作为类代表，预测时只考虑目标用户与每个用户类的类代表之间的相似度从而使相似度的计算量大为减少。[13] 则在创建用户类时也采用 k-measn 算法，不过其距离度量采用 PCC 相似度。在创建用户类后进行数据的光滑处理，也即根据类的信息对缺失值进行填充以减小稀疏性对后续相似判断的不利影响。不同于只对用户或者物品进行划分，[14] 采用 Bregman co-clustering[15] 同时对用户和物品进行划分，然后在每个用户类和物品类组成的小矩形中独立进行预测，这方法的可扩展性更好，而且能够快速响应新的评分，新的用户或新的物品。[16] 首先给出了新的预测公式用于多次导出更有利于增量更新的 co-clustering，然后采用演化计算的思想将不同的 co-clustering 进行交织混合产生新的 co-clustering 直到收敛形成最后的 co-clustering 结构。

因为直接将评分矩阵中的相应的行和列作为用户和物品的聚类表示容易受数据稀疏性的影响导致聚类结果不准确，一些降维技术被用来更好地表示用户和物品以便使基于新的特征表示的聚类可以达到更好的效果。Eigentaste[17] 首先要求每个用户对标准物品集中的每个物品进行评分，形成一个没有缺失值的评分矩阵，然后对这个矩阵进行 PCA(principal component analysis)，从而为每个用户获得一个二降维表示，然后采用在这些二维表示形成的平面上进行一个递归矩形聚类的过程。[18] 首先构建一张图，每个用户代表一个点，边的权值就是用户之间的相似度，然后采用 Normalised Cut 对这个图进行划分进而用图的 Laplacion 矩阵的特征向量导出每个用户的降维表示。不同于只提取用户或者物品的降维表示，[19] 通过把评分矩阵看做一个二部图，求解一个迹范数最小化问题将用户和物品映射到同一个特征空间中，然后基于新的表示采用模糊 c-means 进行聚类使得用户和产品同时被一起聚类。

为了缓解稀疏性问题，矩阵分解技术近年来被大量运用到协同过滤中。常规的 SVD(Singular Value Decomposition) 分解不能直接用来分解评分矩阵，因为存在大量缺失值。[20] 首先对稀疏的评分矩阵进行填充，然后对填充后的评分矩阵再进行 SVD。而填充过程可能并不恰当，甚至引入噪音，所以 [21] 提出了 RSVD(Regularized SVD)，这种方法从已有的评分中为每个用户和物品分别

学习一个特征，使得用户的特征矩阵和物品的特征矩阵的内积能够最好地逼近已有评分，并且这种方法的目标函数就是评价指标 RMSE，所以他在 RMSE 指标上往往能够取得高出其他方法很多的性能。[22] 随后将 RSVD 做了改进，使得 RSVD 成为协同过滤中当下最流行的方法。其实 RSVD 可以用一个漂亮的概率模型来解释 PMF(Probabilistic Matrix Factorization)[23]，在 PMF 中每个评分服从由对应用户和物品决定的正态分布，使用户和物品的特征值相对已有评分的后验概率最大便可以导出 RSVD 的目标函数。因为评分往往是非负的，所以非负矩阵分解 (NMF,Non-negative Matrix Factorization) 也可以运用到协同过滤中。同样由于存在缺失元素，[24] 提出了基于 EM 算法的 NMF 和 WNMF(Weighted NMF)。其实，在协同过滤中，有一些外部信息，如用户的人口统计信息和社交网络信息，物品的概貌信息等可以用来改进性能。[25] 首先根据外部信息为用户和产品构建一幅图，然后把已有的评分信息，用户图和物品图统一到非负矩阵分解的框架下，分解结果不仅仅要尽量逼近已有评分，也要使得图的邻接点之间尽可能地相似。[26] 把朋友关系用于推荐，使朋友之间的特征尽量相似。[27] 则要求朋友之间的特征相似需要由他们的之间的已有评分的相似大小进行加权，而非一味地靠近。Koren[28] 提出了一个综合模型。这个模型同时融合了用户评分尺度差异和物品评分尺度差异，用户的显式反馈和隐式反馈。他们用物品的隐式反馈特征和显式反馈特征一起建模用户喜好，并使得物品之前的隐式相似度和显式相似度在迭代中自动学习而非预先计算。这个综合模型将全局信息和局部信息，隐式信息和显式信息同时运用，最后取到了非常好的效果。

尽管协同过滤方法有着诸多优点，但是他在解决冷启动问题上表现不佳，而基于内容的方法不需要用户和物品之间的历史互动信息，不存在冷启动问题，所以可以把协同过滤和基于内容的方法混合起来。Content-Boosted[29] 可以用来应对新物品，首先根据用户对其他物品的历史评分以及他评价过的物品与新物品在内容上的相似性建立一个 Bayesian 分类器，然后把分类器的输出填充到缺失位置，最后再进行协同过滤。不同于 Content-Boosted 的两阶段混合策略，电视节目推荐系统 Queveo.tv[30] 采用平行混合策略，他给出的推荐列表中一部分是基于内容的一部分是协同过滤的。

1.3　本文主要工作

据我们所知，几乎所有现存的协同过滤都需在某个特征空间上用某种相似度来发现所谓的物品之间或者用户之间的相似关系，我们发现相似关系在协同过滤中是一个基础性的问题。在我们看来，目前协同过滤中的相似关系可以分为独立相似和依赖相似两种。所谓独立相似就是指用户之间的相似度计算与物品之间的相似度计算没有相互作用。独立相似在早期的协同过滤的发展中起到重要作用，也是一个研究热点。然而我们觉得，用户相似应该针对某个物品类而言，因为几乎不会有两个用户始终在所有物品上的看法都是类似的。我们在

整个物品空间上计算用户之间的相似度会导致那些在局部物品类中高相似的用户关系在整个物品空间上被淹没。同样的，物品的相似关系也应该针对某个用户类而言。所以用户之间的相似和物品之间的相似应该是紧密联系且相互平等的，简单地对物品或者用户进行划分不能解决问题。所谓依赖相似则是用户之间的相似关系和物品之间的相似是相互依赖相互决定的。然而现有的依赖相似在寻找相似对象的时候往往都隐含一个假设，那就是两个用户之间的相似关系在面对不同的物品间相似关系时在是静止不变的。但是在现实中，两个用户在一些物品上很相似，而在另一些物品上可能截然不同。在面对一个物品集时，根据对这个物品集的看法用户可以聚成多个类，然而在面对一个新的物品集时，先前的用户类结构可能就会坍塌，根据用户对新物品集的看法再次形成一个新的类结构。类似地，物品根据在某个用户集中的受欢迎程度形成的类结构在面对新的用户集的时候也有可能分裂重组。总而言之，用户类结构在面对不同物品时会不停地变化，反之，物品的类结构在面对不同的用户的时候也会不停地变化。所以本文提出物品和用户之间的相似关系应是相互依赖，并是相互动态地依赖。在这篇论文中，我们用一个块结构来建立用户类和物品类以及刻画他们之间的相互依赖和动态变化。我们提出了用户物品成对分块算法的一般框架，这个框架包含用户和物品统一特征提取，用户和物品之间的对齐，物品或者用户基于对齐表示进行聚类并得到我们期望的块结构，最后我们利用已有的协同过滤算法进行全局和块内学习并给出未知评分的预测。我们给出一种这种框架的具体实现，并通过模拟实验表明这种方法可以有效地提升推荐算法的性能。

本文剩余的部分组织是，在第二章中，我们首先介绍了典型的独立相似和依赖相似的工作，指出他们的不足，并给出了我们方法的一般性框架。在第三章中，我们给出了一种框架的具体实现。在第四章，我们进行了模拟实验，最后在第五章中，我们总结了我们的工作。

第二章　用户物品成对分块协同过滤的一般框架

我们将二元数据表示中相似关系的研究分为相互独立的与相互依赖的。也就是在二元关系中，一方内部的相似关系是否依赖于另一方内部的相似关系。在协同过滤中，二元指的就是用户和物品，我们根据用户之间的相似关系与物品之间的相似关系是否独立来划分现有的算法。我们首先通过介绍几个著名的协同过滤算法，展示独立相似概念在协同过滤中的基础地位。然后介绍了已有的依赖相似方法。接着我们阐述了对依赖相似的新理解，并给出了根据新理解进行协同过滤算法的一般框架。

2.1　协同过滤形式化定义与符号说明

设有 m 个用户和 n 个物品，用户对物品的已有评分可以表示成一个存在大量缺失值的 $m \times n$ 矩阵 $R_{m\times n}$。如果 $r_{u,i} \neq 0$ 或者 $R_{ui} \neq 0$ 则表示这是用户 u 给物品 i 的已知的评分，否则表示用户 u 还没有显示地评价过物品 i。我们的目标便是给出矩阵缺失的元素。U 和 I 分别代表所有的用户和物品。U_i 则代表评价过物品 i 的用户集合。I_u 代表用户 u 评价过的物品集合。$\bar{r}_u$ 或 $\bar{R}_u$ 代表用户 u 已有评分的均值，$\bar{r}_i$ 或 $\bar{R}_i$ 代表物品 i 已获评分的均值。$\bar{r}$ 或 $\bar{R}$ 代表所有已知评分的均值。在后续中我们说目标用户和目标产品时，指的就是需要预测评分的用户产品对。$\mathbb{R}^k$ 表示 k 维实数空间。

2.2　典型的基于独立相似的协同过滤算法

在本节中，我们介绍一些典型的协同过滤算法。通过这些算法的介绍，我们可以一睹独立相似这一概念在协同过滤中的广泛存在和基础地位。

2.2.1　基于用户和基于物品的协同过滤

基于用户和基于物品的协同过滤首先度量每对用户或每对产品之间的相似度，然后根据相似度进行加权预测。PCC(Pearson correlation coefficient) 常常用来度量两个用户之间的相似度。用户 u 和用户 v 之间的 PCC 相似度为：

$$Corr(u,v) = \frac{\sum_{i\in I_u \bigcap I_v}(r_{u,i}-\bar{r}_u)(r_{v,i}-\bar{r}_v)}{\sqrt{\sum_{i\in I_u \bigcap I_v}(r_{u,i}-\bar{r}_u)^2}\sqrt{\sum_{i\in I_u \bigcap I_v}(r_{v,i}-\bar{r}_v)^2}}$$

基于用户的方法 [3] 按如下公式计算未知评分的估计：

$$\widehat{r}_{u,i} = \overline{r}_u + \frac{\sum_{v\in U_i^k} Corr(u,v)(r_{v,i}-\overline{r}_v)}{\sum_{v\in U_i^k}|Corr(u,v)|} \tag{2.1}$$

U_i^k 是与评价过物品 i 且与用户 u 最相似的 k 个用户的集合。如果评价过物品 i 的用户不足 k 个，那么 $U_i^k = U_i$.

[4] 提出了用修正的余弦相似度来度量两个物品 (i 和 j) 之间的相似度，

$$ACosine(i,j) = \frac{\sum_{u \in U_i \bigcap U_j}(r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U_i \bigcap U_j}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{u \in U_i \bigcap U_j}(r_{u,j} - \bar{r}_u)^2}} \tag{2.2}$$

注意，这种相似度的本质就是先做标准化，也就是每个评分减去用户的平均评分，最后计算评分矩阵中对应两列的余弦相似度。评分预测公式则改为如下：

$$\widehat{r}_{u,i} = \frac{\sum_{j \in I_u^k} ACosine(i,j) r_{u,j}}{\sum_{j \in I_u^k} |ACosine(i,j)|} \tag{2.3}$$

I_u^k 由用户 u 评价过的且与物品 i 最相似的 k 个物品组成，同样地，如果 $|I_u| < k$, 那么 $I_u^k = I_u$.

以上相似度存在虚假高相关问题，也就是两个用户共同评价过得物品很少 ($|I_u \bigcap I_v|$ 很小) 或者同时评价过两个物品的用户很少 ($|U_i \bigcap U_j|$ 很小) 元素很少，从而计算出来的相似度很高，我们采用显著性加权来避免这个问题，也即：

$$Sim(u,v) = \frac{min(\gamma, |I_u \bigcap I_v|)}{\gamma} Corr(u,v)$$

$$Sim(i,j) = \frac{min(\gamma, |U_i \bigcap U_j|)}{\gamma} ACosine(i,j)$$

2.2.2 相似度汇聚算法

Jun Wang 等 [31] 提出了相似度汇聚 (SF, Similarity Fusion) 算法。这种算法的核心思想就是每个已知评分都可以作为未知评分的估计。具体地，预测用户 u 对物品 i 的评分时，三种已知评分都可以认为未知评分等于自己。一是与用户 u 相似的用户对物品 i 的评分，这些评分构成集合 SUR，这里的用户间的相似由余弦相似度确定。二是用户 u 对与物品 i 相似的物品的评分，他们构成集合 SIR，这里的相似由修正的余弦相似度 (方程.2.2) 确定。三是与用户 u 相似的用户对与物品 i 相似的物品的评分，构成集合 SUIR，这里相似由用户间的余弦相似度和物品间的修正余弦相似度的几何均值决定。图2.1展示了三个集合，SUR，SIR 和 SUIR。最后的预测公式如下:

$$\widehat{r}_{u,i} = \sum_{v,j} W_{u,i}^{v,j}(r_{v,j} - (\bar{r}_v - \bar{r}_u) - (\bar{r}_j - \bar{r}_i)) \tag{2.4}$$

$W_{u,i}^{v,j}$ 是用评分 $r_{v,j}$ 预测未知评分 $\widehat{r}_{u,i}$ 时的权重，对于三种不同的评分，权值的计算公式如下:

$$W_{u,i}^{v,j} = \begin{cases} \frac{Cosine(u,v)\lambda(1-\delta)}{\sum_{r_{v,i}\in SUR} Cosine(v,u)} & r_{v,j} \in \text{SUR} \\ \frac{ACosine(i,j)(1-\lambda)(1-\delta)}{\sum_{r_{u,j}\in SIR} |ACosine(i,j)|} & r_{v,j} \in \text{SIR} \\ \frac{Sim(u,i;v,j)\delta}{\sum_{r_{v,j}\in SUIR} Sim(u,i;v,j)} & r_{v,j} \in \text{SUIR} \\ 0 & \text{otherwise} \end{cases}$$

其中，

$SUR = \{r_{v,i}|v \in U_i^N\}$，$SIR = \{r_{u,j}|j \in I_u^N\}$，$SUIR = \{r_{v,j}|v \in U_i^N, j \in I_u^N\}$，

$$Sim(u,i;v,j) = \frac{|ACosine(i,j)|Cosine(u,v)}{\sqrt{ACosine(i,j)^2 + Cosine(u,v)^2}}$$

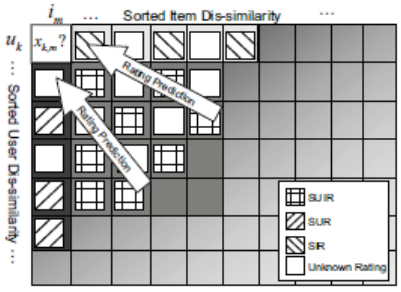


图 2.1: 相似度汇聚算法示意图

2.2.3　融合社会关系的协同过滤

最近，矩阵分解被大量运用在推荐算法中，RSVD(Regularized Singular Value Decomposition) 是最经常出现的分解方法。这种方法的核心思想就是用低维参数模型刻画用户与物品的互动。另一方面，如果近似标准是差矩阵的 Frobenius 范数，那么 SVD 是最好的矩阵低维逼近方法。RSVD 试图将评分矩阵做如下分解：

$$R \approx U^T V$$

分解的目标就是最小化如下函数：

$$\frac{1}{2}\sum_{u=1}^{m}\sum_{i=1}^{n} W_{ui}(r_{ui} - U_u^T V_i)^2 + \lambda(\|U\|_f^2 + \|V\|_f^2) \tag{2.5}$$

W 是一个 0-1 矩阵，如果 $r_{ui} \neq 0$，$W_{ui} = 1$, 否则等于 0。矩阵 $U \in \mathbb{R}^{k\times m}$ 的每列 U_u 代表一个用户 u 的特征。矩阵 $V \in \mathbb{R}^{k\times n}$ 的没列 V_i 代表一个物品 i 的特征。

标准的 RSVD 使求解的用户特征和物品特征的内积尽可能地逼近已有评分，而 [27] 提出用朋友关系来改进 RSVD。主要思想就是，一个人对某个物品的偏好是由他自己的属性，物品的属性以及用户的朋友圈子共同影响的。然而用户的朋友们的偏好可能差别很大，所以作者认为每个朋友对用户的影响是不一样

的，这种影响应该很据他们彼此之间的相似度来加权。从而提出最小化如下的目标函数：

$$\begin{aligned}&\frac{1}{2}\sum_{u=1}^{m}\sum_{i=1}^{n}W_{ui}(r_{ui}-U_u^TV_i)^2+\lambda_1\|U\|_f^2+\lambda_2\|V\|_f^2\\&+\frac{\beta}{2}\sum_{u=1}^{m}\sum_{v\in F(u)}Sim(u,v)\|U_u-U_v\|^2\end{aligned}\tag{2.6}$$

$F(u)$ 就是用户 u 的朋友集合。可以看出用户的特征将受每个朋友不同程度的拉扯。上面的优化问题可以通过随机梯度下降法解得。而至于怎么度量两个用户之间的相似度大小 $Sim(u,v)$ 的问题，作者在文中用 PCC 相似度，直接计算两个用户已有评分的行之间的相似程度。

2.2.4　迭代计算相似关系的协同过滤

前面几种方法都是根据已有评分数据直接计算出相似度，这个计算过程与预测本身并不存在紧密联系，这样计算出来的相似度与具体的预测公式以及性能评价方法是独立的。[32] 提出了一种也是基于独立相似的预测方法，不过其相似度计算与预测方法和评价指标紧密相连。预测公式为：

$$\widehat{r}_{u,i}=\sum_{j\in N(i;u)}w_{i,j}r_{u,j}\tag{2.7}$$

其中 $w_{i,j}$ 就是物品 i 和物品 j 的相似度。$N(i;u)$ 则是用户 u 评价过的与物品 i 最相似的 k 个物品集合。这里 $w_{i,j}$ 是最小化如下目标函数得到的：

$$\sum_{u,i|r_{u,i}\neq 0}\left(r_{u,i}-\sum_{j\in N(i;u)}w_{i,j}r_{u,j}\right)^2$$

为了得到所有的 $w_{i,j}$，先设

$$w=[w_{i,1},w_{i,2},\ldots,w_{i,n}]^T$$

然后，求解方程组：

$$\widehat{A}w=\widehat{b}$$

便得到了 $w_{i,1},w_{i,2},\ldots,w_{i,n}$，然后对于不同的 i 解一次上述的方程组便可以得到所有的 w_{ij}。其中的矩阵 $\widehat{A}$ 和向量 b 是根据已知评分构建的，具体的构建过程可以参考 [32]。我们可以发现这样计算出来的物品间相似关系仍然与用户之间的相似关系是独立的。另外，需要指出的是 $N(i;u)$ 的计算仍然是按照前述方法计算相似度然后进行选择得到的。

2.3　现有的基于依赖相似的协同过滤算法

我们已经看到了，上一节中介绍的方法有一个共同的特点，他们都需要用户 (物品) 之间的相似信息，而用户 (物品) 之间的相似关系与物品 (用户) 之间的相似关系是相互独立的。在这一节中，我们介绍我们目前所知的用户相似关系和物品相似关系相互影响相互依赖的工作，并指出他们的核心思想和不足。

2.3.1　基于用户物品同时聚类的推荐算法

有时候，我们通过聚类来建立对象之间的相似关系，属于同一个类的对象我们认为是相似的，所以在后续，我们说相似既可以指相似度的计算方法，也可以指物品或者用户的聚类方法。在协同过滤中有一些工作可以同时对用户和物品进行聚类。

[14] 提出将协同聚类 (co-clustering) 运用到协同过滤推荐算法中。主要思想就是对用户和物品同时划分，然后每个用户类和物品类一一结合，形成一个个小的评分矩阵，最后每个未知评分根据所在的小矩阵内的已知评分进行预测。注意我们说用户类和物品类结合就是指我们打算讨论用户类中的每个用户对物品类中每个物品的偏好。设最终用户的类标号函数和物品的类标号函数分别为:

$$\rho : \{1, 2, \ldots, m\} \mapsto \{1, 2, \ldots, k\}$$

$$\gamma : \{1, 2, \ldots, n\} \mapsto \{1, 2, \ldots, l\}$$

通过对评分矩阵的行和列进行交换，我们可以使同一个用户类的行排在一起，同属一个物品类的列排在一起，如图2.2。

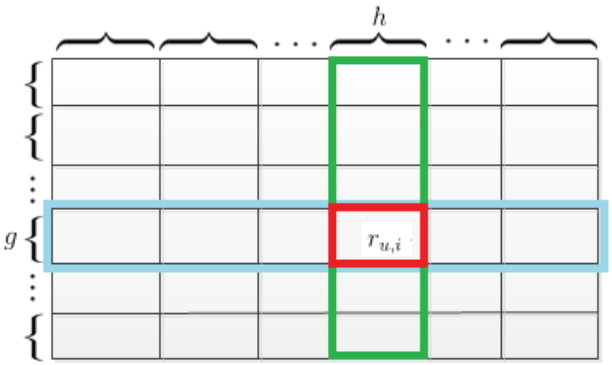


图 2.2: 协同聚类中的协同聚类示意图

那么最后的预测公式为:

$$\widehat{r}_{u,i} = r_{gh}^{COC} + (\overline{r}_u - r_g^{RC}) + (\overline{r}_i - r_h^{CC}) \tag{2.8}$$

其中 $g = \rho(u), h = \gamma(i)$，$r_{gh}^{COC}$ 就是图中红色方框内的已知评分的均值，r_g^{RC} 就是浅蓝色方框内已知评分的均值。r_h^{CC} 就是绿色方框内中已知评分的均值。最

后的类标号函数是通过最小化下述目标函数获得的

$$min_{\rho,\gamma}\sum_{u=1}^{m}\sum_{i=1}^{n}W_{ui}(r_{ui}-\widehat{r}_{u,i})^2 \tag{2.9}$$

而根据 [15]，这个目标函数可以通过交替地优化用户类标号函数和物品类标号函数求解，也就是先随机给出一个用户的类划分，在此基础上，再重新分配每个物品的类标号，使得方程2.9到达最小，此时便得到了物品的类划分，接着固定物品的类划分，重新分配每个用户的类标号，也使得2.9到达最小，如此交替迭代直到彼此的类分配都不变为止。我们可以很容易得出结论，用户的类结构和物品的类结构是相互依赖的。

而 [33] 是利用对评分矩阵的分解同时得到用户类的物品类的。具体地求解如下问题，

$$\begin{aligned} min_{U,S,V} \quad & \|R-USV^T\|^2 \\ s.t \quad & U^TU=I \\ & V^TV=I \\ & U\succeq 0, S\succeq 0, V\succeq 0 \end{aligned} \tag{2.10}$$

其中 $\succeq$ 表示矩阵所有元素非负。得解后，$argmax_kU_{uk}$ 便是用户 u 的类标号，$argmax_kV_{ik}$ 便是物品 i 的类标号，$(SV^T)_{k\cdot}$ 便是用户类 k 的类心评分向量，$(US)_{\cdot l}$ 便是物品类 l 的类心得分向量。作者然后根据每个用户类类心与目标用户评分向量的 VSS 相似度选择一批相近类，然后再在相近类中再通过 VSS 相似度选择相似用户，最后采用基于用户的预测评分。类似的，也可以得到基于物品的预测评分，然后将二者线性组合作为最终的预测评分。同样地我们可以看出这种方法得到的用户类结构和物品类结构是相互依赖的。

上面这两种方法虽然使得用户的相似关系和物品的相似关系彼此依赖。但是这两种方法却有一个共同的缺点，那就是用户的相似关系在面对不同的物品时是静止的，即两个用户在面对任何物品时都展现出相近的偏好。另外还有另外一个缺点，在 [14] 中每个用户类和每个物品类的结合没有明确的理由和意义，我们不清楚为什么每个用户类要和每个物品类进行结合，尽管他们是交替依赖生成的。当然 [15] 没有遭遇用户类和物品类的结合问题。类结构其实只是为了减少寻找邻居时的比较次数，两种类结构同时出现也是为了将基于用户的和基于物品的预测进行线性地组合而已。

2.3.2　物品加权的用户相似度量

[9] 认为每两个用户在面对不同物品时的相似度都是不同的，所以提出用户 u 和用户 v 在物品 i 上的相似度应该为:

$$Sim^i(u,v)=\frac{\sum_{j\in I_u\bigcap I_v}lsim(i,j)(r_{u,j}-\overline{r}_u)lsim(i,j)(r_{v,j}-\overline{r}_v)}{\sqrt{\sum_{j\in I_u\bigcap I_v}(lsim(i,j)(r_{u,j}-\overline{r}_u))^2}\sqrt{\sum_{j\in I_u\bigcap I_v}(lsim(i,j)(r_{v,j}-\overline{r}_v))^2}}$$

两个物品之间的相似度 $lsim(i,j)$ 则是直接比较评分矩阵中相应两列的余弦得到的。这种方法有两个问题，其一，每次推荐时都要计算目标用户在目标物品上与哪些用户相似，然后再做出预测，那么毫无疑问计算时间将是非常巨大的，如果不直接计算而是预先存储所有必要的相似度那么需要的空间便是 $\mathcal{O}(m^2n+n^2)$。其二，这里虽然指出了用户之间的相似关系在面对不同物品时应该变化，但是这种变化却又是依赖于物品之间的相似是静止的假设，也即是 $lsim(i,j)$ 的计算跟用户的相似关系无关。

2.3.3　多类协同过滤算法

在意识到用户的相似性在面对不同物品类会变化的事实后，[19] 提出多类协同聚类算法 (MCoC,multiclass co-clustering)。这种方法首先根据已有评分，为每个用户和物品提取统一的特征表示，也就是最小化下面这个函数：

$$\sum_{u=1}^{m}\sum_{i=1}^{n}\|\frac{\mathbf{q}_u}{\sqrt{D_{uu}^{row}}}-\frac{\mathbf{p}_i}{\sqrt{D_{ii}^{col}}}\|_f^2 R_{ui} \tag{2.11}$$

其中 $\mathbf{q}_u\in\mathbb{R}^k$ 是用户 u 的特征表示，$\mathbf{p}_i\in\mathbb{R}^k$ 是物品 i 的特征表示。D_{uu}^{row} 是评分矩阵的第 u 行全部元素的和，D_{ii}^{col} 是评分矩阵第 i 列全部元素的和。

然后，将基于用户和物品的新表示对用户和物品一起聚类。也就是现在的被聚类的对象由用户和物品一起组成，但是在聚类中并不区分是用户还是物品，这样得到的每个类中既有物品也有用户。为了使用户物品可以属于多个类，作者采用模糊 c-means 聚类。最后每个类可以组成一个小的评分矩阵，然后再在每个小的评分矩阵中可以采用任意已有的协同过滤算法。需要指出的是这种方法采用模糊 c-means 聚类，所以一个用户物品对可能属于多个类，被多个小评分矩阵覆盖从而有多个预测值。对于这种情况，作者组合了来自多个类的预测。

这种方法在我们看来有四个地方有待改进。第一，特征提取的目标函数 (2.11) 设计的不合理。这个目标函数明显存在一个全局最小点，即 $\mathbf{q}_u$ 的每个分量都取值为 $\sqrt{D_{uu}^{row}}$，并且 $\mathbf{p}_i$ 的每个分量都取值为 $\sqrt{D_{ii}^{col}}$，此时式2.11取值为 0，达到全局最小。第二，作者是通过模糊 c-means 聚类使得电影和用户可以隶属多个类的，如果不做模糊化，那么我们得到的就是每个物品每个用户都只能属于一个类，我们交换评分矩阵的行列后，我们可以发现很重要的问题，其实这种做法只是对评分矩阵做了准对角化 (图2.3)，也就是用户和物品在准对角块中才能结合，那么我们就要问，难道每个用户只会对他所在准对角块中的物品进行评价吗？他与其他物品难道没有互动吗？当然作者采用模糊聚类使这个问题看似不存在，不过我们认为聚类的模糊化不足以从本质上解释用户和物品的结合。第三，作者得到了用户物品类以后，在每个小评分矩阵中用现存的协同过滤方法独立地预测，我们认为这种预测没有充分利用所得的用户物品类信息，而且作者忽略了全局信息。用户与物品的互动不仅仅受小矩阵中已有元素的影响，而且还受用户本身属性和物品本身属性的影响。也就是，用户对物品的偏好跟

他的本身的属性有关，比如他天生喜欢动作片，还跟他的局部属性有关，比如他看见周围用户喜欢某部电影，他也可能也会去观看并喜欢。同样地，电影也是由全局信息和局部信息一起刻画的。第四，这种方法会使小矩阵之间出现重叠，结果就是某个用户对某个物品的互动可能同时存在于多个小矩阵中。我们则认为某个用户对某个物品的互动应该只属于某一个类，而我们的目标就应该是将所有类似的用户物品对聚在同一个类中。

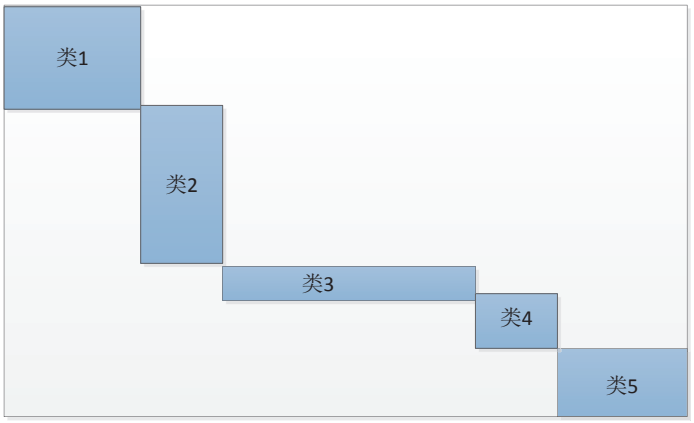


图 2.3: MCoC 非模糊聚类示意图

2.4　用户物品成对分块推荐算法的框架

在过去十多年的时间里，协同过滤已经成为推荐算法中最流行的一种技术。这种技术只根据用户与物品在过去的互动信息来预测用户可能感兴趣的物品，而不像基于内容的推荐算法那样需要对的产品做出细致的描述。正由于与领域知识无关，加之其有效性和易于部署实施的特点，协同过滤引起了很大的研究热情，并被广泛应用在实际的商业系统中。据我们所知，很多现有的协同过滤算法往往需在某个特征空间上用某种相似度来发现所谓的物品之间或者用户之间的邻居关系，在我们看来，相似关系或者邻居关系是协同过滤中的一个基础性问题。然而大多数协同过滤算法在寻找邻居的时候都隐含一个假设，那就是用户之间的相似性与物品之间的相似性是相互独立，比如 item-based 算法和 user-based 算法。尽管有些工作已经意识到用户之间的相似性和物品之间的相似性是相互影响相互决定的，但是这些工作往往又假设两个用户之间的相似关系在面对不同的物品的时候是静止不变的，两个物品之间的相似关系在面对不同用户时也是不变的，比如基于 co-clusteirng 的协同过滤。这一点可以从我们在2.2节中的相关工作介绍已经可以看出。但是在现实中，却不是这样的。

在面对一个物品集时，根据对这个物品集的看法用户可以聚成多个类，然而在面对一个新的物品集时，先前的用户类结构可能就会坍塌分裂，根据用户对新物品集的看法再次形成一个新的类结构。举个直观的例子，如图2.4所示，

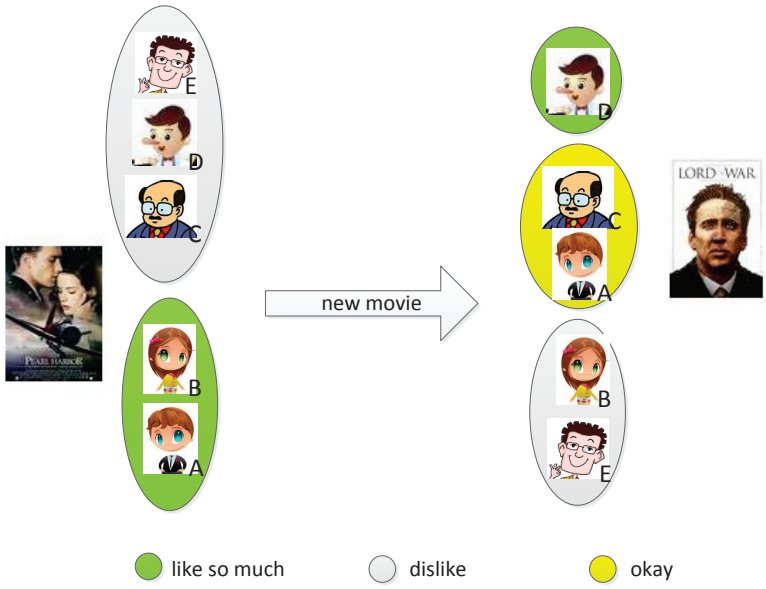


图 2.4: 用户类结构随着物品变化示意图

在面对珍珠港这部电影的时候，5 个用户可以根据他们喜欢与否分成喜欢 (A，B) 和不喜欢两个类 (C，D，E)。但是在面对战争之王这部电影时，5 个用户分成了 3 个类，喜欢 (D)，不喜欢 (B，E)，觉得还行 (C，A)。不仅用户之间的相似关系在面对不物品时是变化的，而且这种变化可能会导致类个数的增加或者减少。不同的物品在用户中有着不同的喜好区分度。类似地，物品根据在某个用户集中的受欢迎程度形成的类结构在面对新的用户集的时候也有可能分裂重组。

总而言之，用户类结构在面对不同物品时会不停地变化，反之，物品的类结构在面对不同的用户的时候也会不停地变化。用户间相似关系和物品间相似关系不仅仅相互依赖，而且依赖关系还会不断变化。这样的相似不是通过单独进行用户划分或者物品划分或者同时进行用户与者物品的划分就可以得到的。用户之间相似关系必须针对某个具体的物品集而言，然后可以凭借得到的相似关系形成一个个用户集。同样地，物品之间的相似关系也必须是相对于某个用户集而言的，然后据此才能形成一个个物品集。一对相似关系是一个四元组 (用户 u，用户 v，物品集 SI，相似度 Sim 或者类隶属关系)，表示在某个物品集上两个用户的相似度，或者为 (物品 i，物品 j，用户集 SU，相似度 Sim 或者类隶属关系)，表示在某个用户集上两个物品的相似度。

现在，我们可以总结我们期望的类结构具有如下特点：

1. 用户类结构和物品类结构相互依赖。因为我们希望用户类结构反映的是用户对某个物品类看法的相似性，即之所以这些用户在一个类里面是因为他们对某个物品类具有相似程度的喜好。同样的，一些物品之所以可以形成一个类，是因为他们在一个用户类中被类似地评价。

2. 用户类结构和物品类结构对彼此是相互变化的。也即是每两个用户几乎不

可能在所有的物品上都始终能持有相似的看法，每两个物品也几乎不可能在所有的用户中始终被类似地评价。随着所面临的物品的变化，用户之间的相似关系时弱时强。面对不同用户时，物品之间的相似关系也是会起伏变化。

3. 用户类和物品类的结合应当是有意义的。我们不必也不应该去讨论每个用户类对每个物品类的看法。我们希望之所以用户类和物品类能够结合是因为这些用户都对这个物品类中每个物品持有类似的看法，而物品类中每个物品也被用户类中所有用户类似评价。

4. 用户可以属于多个用户类。因为现实中，用户不会只有单一的兴趣偏好，而用户的兴趣偏好是由物品类结构中每个类的潜在语义解释的。同样地，每个物品可以属于多个物品类，因为每个物品在整个用户集合中不可能只有一种看法，必然有人喜欢，有人讨厌，有人很喜欢，有人很讨厌。但是每个用户物品对只能属于一个类。

在本文中，我们试图去发掘隐藏在评分矩阵中的块结构，每个块由许多用户和物品组成，一个块中的所有用户构成一个用户类，所有物品构成一个物品类。我们用一个块结构来建立用户类结构和物品类结构以及刻画他们之间的相互依赖，相互变化以及有意义的结合。在这样的每个块中，其中的用户对其中的物品持有很相似的看法，其中的物品被其中的用户类似地评价。每一个块都是一个富有意义的被用户和产品共享的结构，反映了在现实中用户兴趣和物品属性之间的切合程度以及用户类和物品类对彼此形成的影响。我们最终希望的块结构类似图2.5。每个块是整个评分矩阵的一个子矩阵，子矩阵的行对应着属于这个块的用户，列对应着属于这个块的物品，每个已知和未知的评分属于他所关联的用户和物品同时出现的块。与 co-clustering 得到的块不同，这里所有块的边缘不一定是对齐的，反映了用户和物品之间的类共现关系是变化的。

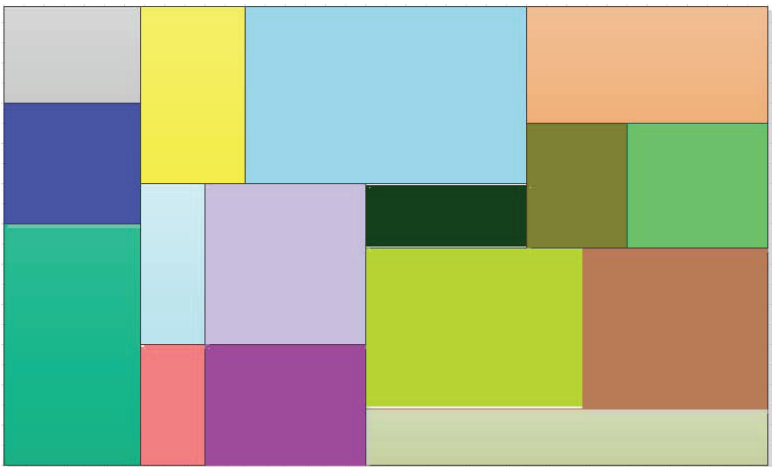


图 2.5: 期望的块结构

为了从评分矩阵中发掘想要的块结构，我们首先给出一个一般性的框架(图2.6)。我们的框架是一个分步骤顺序执行的算法，共分 6 步，每一步指明了我们当前的目标。至于每步的目标实现细节则取决于具体的组件。

图 2.6: 用户物品成对分块推荐算法的框架

第一步，我们首先提取用户和物品的统一低维表示。这是一个降维压缩的过程以便后续我们可以基于低维表示进行诸多处理而不必担心巨大的用户和物品数量带来的计算难题，也使得我们可以不必以物品表示用户或者以用户表示物品，而是有一个统一的表示，在必要情况下不必区分用户和物品，便于使用经典的聚类算法同时对用户和物品进行聚类。我们可以通过用户固有属性，用户的社交关系，物品的固有属性，物品之间的相互联系，用户对物品的显式反馈和隐式反馈等等已知信息中提取用户和物品的特征表示。我们可以设计各种各样的目标函数用于得到最优的特征，也可以用各种启发式算法。值得注意的是，我们需要用户和物品在新的特征空间中可以直接进行比较，例如度量距离。

第二步，在得到用户和物品的统一特征表示后，我们相对每个物品对用户进行划分操作。在这一步中，我们引进了用户对齐这个概念，意即相对每个物品，我们将所有用户进行划分，而且相对于不同物品的用户划分结果都用一种可比较的方式表示。我们之所以引进这个概念是为了比较不同物品的用户划分结果，并重新用一种规范的格式表示物品。我们将在后续章节详细讨论。当然可以相对于每个用户，对物品进行划分。

第三步，基于每个物品的用户划分表示，我们对物品进行聚类。注意此时每个物品已经用一种规范的格式重新表示，可以直接基于这种表示进行聚类。这一步是标准的聚类问题，不过这将是一个在高维稀疏空间中寻找结构松散类

结构的问题，如果快速度量高维空间中两个对象的聚距离将是一个难题。另外尽管存在很多经典的聚类算法，但是也许直接使用这些聚类算法而不顾协同过滤这个具体的问题背景，可能得到的效果较差。在这一步完成后，我们就可以得到隐藏在评分矩阵中的块结构了。

第四步和第五步进行评分预测模型的学习。我们认为用户对物品的反馈受两大类因素影响。一个是他和物品各自本身固有的属性，这类属性比较稳定，不易变化，我们称之为全局特征。另一个则是受周围邻近的用户或物品的影响，这一类属性则容易变动。我们称之为局部特征。比如说，一个用户喜欢喜剧片，这是该用户的固有属性。当他发现很多人在看周星驰的电影后，他也去看，看完以后，他可能更喜欢喜剧片了，这里对喜剧片的喜好程度的就是受周围人的影响。所以在第四步和第五步中，我们分别学习这两类因素。

最后，我们就可以就运用学习到的预测特征进行评分预测了。下一章中，我们给出这种框架的一种具体的实现。

第三章　用户物品成对分块框架的具体实现

在上一章中，我们给出了一般性的框架，而在本章，我们则给出了这种框架中每一步的一种具体实现。具体的实现可以多种多样，取决于具体的环境，目的和可用数据。

3.1　统一特征提取与用户对齐

我们第一步就是从已有可用信息中提取用户和物品的统一低维特征。本文中，可用信息是已知的极少的历史评分。我们先根据假设导出特征提取问题的数学形式，这是一个标准的 Trace Ratio 优化问题。然后我们接着详细讨论了求解这个问题的数值算法。最后，我们实现框架的第二步，相对于每个物品，我们采用规范的方式对齐用户。结果使得每个物品表示成一个个用户集合。

3.1.1　统一特征提取的形式化

为了得到用户和物品统一低维表示，我们需要在同一个特征空间中重新表示用户和物品。我们认为，用户对物品的评分越高，那么该用户和该物品在特征空间中的距离就越近。反之，用户对物品的评分越低，那么他们在特征空间中距离就越远。这里的距离我们采用 Euclidean 距离。受 Laplacean Eigenmaps 和 Locality Preserving Projections (LPP) 的启发，我们提出最大化如下的目标函数：

$$\frac{\sum_{u,i} \|\mathbf{q}_u - \mathbf{p}_i\|_f^2 R_{ui} \delta(R_{ui} < r_\theta)}{\sum_{u,i} \|\mathbf{q}_u - \mathbf{p}_i\|_f^2 R_{ui} \delta(R_{ui} \geq r_\theta)} \tag{3.1}$$

其中 $\mathbf{q}_u \in \mathbb{R}^k$ 是用户 u 的低维表示向量, $\mathbf{p}_i \in \mathbb{R}^k$ 是物品 i 的低维表示向量。δ 是一个条件示性函数，如果条件为真，则取值为 1，否则为 0。为了决定用户和物品的靠近或远离，我们给出一个参数 r_θ，如果用户对物品的评分大于等于这个值，我们希望他们的特征尽可能地靠近，且靠近的距离与评分成反比，否则我们希望他们的特征尽可能地远离。且远离的距离与评分成反比。注意缺失值不会影响这个目标函数的求解，因为 $R_{ui} = 0$ 时上述目标函数的取值不变。不过上式可能存在泛化性能较差的问题，所以我们附加一个正则化因子，最后的目标函数如下：

$$\frac{\sum_{u,i} \|\mathbf{q}_u - \mathbf{p}_i\|_f^2 R_{ui} \delta(R_{ui} < r_\theta)}{\sum_{u,i} \|\mathbf{q}_u - \mathbf{p}_i\|_f^2 R_{ui} \delta(R_{ui} \geq r_\theta) + \lambda(\sum_u \|\mathbf{q}_u\|_f^2 + \sum_i \|\mathbf{p}_i\|_f^2)} \tag{3.2}$$

λ 是一个用于控制正则化程度的正数。

对式 (3.2) 做一些常规的化简，比如我们化简分子，有：

$$
\begin{aligned}
&\sum_{u,i} \|\mathbf{q}_u - \mathbf{p}_i\|_f^2 R_{ui}\delta(R_{ui} < r_\theta) \\
&= \sum_{u=1}^{m}\sum_{i=1}^{n} \|\mathbf{q}_u - \mathbf{p}_i\|_f^2 [R_2]_{ui} \qquad \text{(注意 } [R_2]_{ui} = R_{ui}\delta(R_{ui} < r_\theta)\text{)} \\
&= \sum_{u=1}^{m}\sum_{i=1}^{n} \mathbf{q}_u^T\mathbf{q}_u [R_2]_{ui} + \sum_{u=1}^{m}\sum_{i=1}^{n} \mathbf{p}_i^T\mathbf{p}_i [R_2]_{ui} - \sum_{u=1}^{m}\sum_{i=1}^{n} 2\mathbf{q}_u^T\mathbf{p}_i [R_2]_{ui} \\
&= \sum_{u=1}^{m} \mathbf{q}_u^T\mathbf{q}_u \sum_{i=1}^{n}[R_2]_{ui} + \sum_{i=1}^{n} \mathbf{p}_i^T\mathbf{p}_i \sum_{u=1}^{m}[R_2]_{ui} - 2\sum_{u=1}^{m}\sum_{i=1}^{n} \mathbf{q}_u^T\mathbf{p}_i [R_2]_{ui} \\
&= \sum_{u=1}^{m} \mathbf{q}_u^T\mathbf{q}_u [R_2^{row}]_{uu} + \sum_{i=1}^{n} \mathbf{p}_i^T\mathbf{p}_i [R_2^{col}]_{ii} - 2\sum_{u=1}^{m}\sum_{i=1}^{n} \mathbf{q}_u^T\mathbf{p}_i [R_2]_{ui} \\
&= tr(Q^T R_2^{row} Q) + tr(P^T R_2^{col} P) - 2tr(Q^T R_2 P) \\
&= tr([Q^T \quad P^T] \begin{bmatrix} R_2^{row} & -R_2 \\ -R_2^T & R_2^{col} \end{bmatrix} \begin{bmatrix} Q \\ P \end{bmatrix} \qquad \text{(注意 } tr(Q^T R_2 P) = tr(P^T R_2^T Q)\text{)}
\end{aligned}
$$

所以，我们可以得到最后化简的形式如下：

$$\frac{tr\left(X^T M_2 X\right)}{tr\left(X^T(M_1 + \lambda I)X\right)} \tag{3.3}$$

其中,

$$X = \begin{bmatrix} Q \\ P \end{bmatrix}, Q^T = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m], P^T = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n]$$

$$M_1 = \begin{bmatrix} R_1^{row} & -R_1 \\ -R_1^T & R_1^{col} \end{bmatrix}, M_2 = \begin{bmatrix} R_2^{row} & -R_2 \\ -R_2^T & R_2^{col} \end{bmatrix}$$

，$[R_1]_{ui} = R_{ui}\delta(R_{ui} \geq r_\theta)$，$R_1^{row}$ 是一个对角矩阵, 主对角线上第 u 个元素为 $\sum_{i=1}^{n}[R_1]_{ui}$, R_1^{col} 也是一个对角矩阵, 主对角线第 i 个元素为 $\sum_{u=1}^{m}[R_1]_{ui}$.。类似地，R_2^{row} 也是对角矩阵，每个对角元素为 R_2 相应行中元素之和。R_2^{col} 是对角矩阵，每个对角元素为 R_2 相应列中元素之和。I 是单位矩阵。$X \in \mathbb{R}^{(m+n)\times k}$，他的前 m 行是用户特征，后 n 行是物品特征。

如果，我们构建一个用户 -物品二部图 $G = (\mathcal{U}, \mathcal{W}, \mathcal{I}), \mathcal{U}$ 是用户顶点集，$\mathcal{I}$ 是物品顶点集，$\mathcal{W}$ 则是已知用户对物品的评分。当我们把该二部图 G 中边权小于 r_θ 的边去掉以后得到的新图的 Laplacian 矩阵便是 M_1。M_2 则是去掉图 G 中边权大于等于 r_θ 的边之后得到的新图的 Laplacian 矩阵。可以很容易地证明，图的 Laplacian 矩阵是半正定的。所以我们有：

$$M_2\text{半正定}, M_1 + \lambda I\text{正定}$$

注意，对于式3.3，最优解 X^* 的任意数乘都是最优解，另外 X^* 任意右乘一个正交矩阵也是最优解，也就是对 X 的列进行正交变换不改变解的最优性。但是 X^* 左乘正交矩阵将则不再是最优解，另外也为了防止 X 是全 0 矩阵，导致方程3.3分母为 0，我们对 X 施以列正交约束来得到唯一的解。所以我们的统一特征提取问题的形式化为：

$$\begin{cases} \max & \dfrac{tr\left(X^T M_2 X\right)}{tr\left(X^T (M_1 + \lambda I) X\right)} \\ \text{s.t} & X^T X = I \end{cases} \tag{3.4}$$

3.1.2　统一特征提取的求解算法

方程 (3.4) 是标准形式的 Trace Ratio 优化问题，这个问题经常出现在降维的文献中 [34, 35]。由于这个问题求解的困难性，之前的很多工作先将这个问题转换为 Ratio Trace 问题再求解，但是转换导致了解的扭曲。最近 [36] 提出了一种可以直接求解 Trace Ratio 本身的算法，该算法比起先转换再求解的方法更准确，而且还更高效，并保证收敛。这种算法的实质就先构造一个辅助方程，然后用 Newton 求解这个方程。比如一般化表示 Trace Ratio 问题如下：：

$$\begin{cases} \max & \dfrac{tr\left(V^T A V\right)}{tr\left(V^T B V\right)} \\ \text{s.t} & V^T V = I \end{cases}$$

只要 B 正定，A 对称，那么 Trace Ratio 问题就存在有限的最大值和最优解。然后构造如下辅助函数：

$$f(\rho) = \max tr[V^T (A - \rho B) V]$$

可以证明 $f(\rho)$ 是严格递减函数，而且方程 $f(\rho) = 0$ 的解就是 Trace Ratio 问题的最优值。而我们可以用 Newton 求解这个方程组：

$$\rho_{k+1} = \rho_k - \frac{f(\rho_k))}{f'(\rho_k)} = \frac{tr\left(V(\rho_k)^T A V(\rho_k)\right)}{tr\left(V(\rho_k)^T B V(\rho_k)\right)}$$

从而，可以得到3.4的解法 (3.1)：

在协同过滤中，$M_2 - \rho(M_1 + \lambda I)$ 是非常巨大和高度稀疏的，要高效地求解这样一个矩阵的最大几个特征值问题，我们采用 EIGIFP 算法 [37]。另外需要指出的是在算法 (3.1) 的初始化步骤中，我们首先随机产生一个矩阵，然后对这个矩阵的列向量进行 Gram － Schmidt 正交化，便得到一个列正交的初始矩阵。最后，我们采用相对误差小于阈值的终止标准。

1: select initial unitary matrix X, and compute
$\rho = \frac{tr\left(X^T M_2 X\right)}{tr(X^T(M_1+\lambda I)X)}$.
2: **while** not convergence **do**
3: compute k eigenvectors associated with k largest eigenvalues of matrix $M_2 - \rho(M_1 + \lambda I)$, $u_1, u_2, \ldots, u_k$, and let $X = [u_1, u_2, \ldots, u_k]$.
4: let $\rho = \frac{tr\left(X^T M_2 X\right)}{tr(X^T(M_1+\lambda I)X)}$.
5: **end while**

算法 3.1: algorithm for Trace Ratio maximization

3.1.3　用户相对于物品的对齐

在得到用户和物品的统一低维表示后，我们相对于每个物品，将用户进行对齐。当然也可以相对于每个用户，将物品进行对齐。我们认为，用户特征与物品特征之间的距离反映了用户对物品的喜好的近似表示。对于每个物品，我们对所有用户进行聚类，然后我们考察每个物品上的用户聚类结果的相似性，将聚类结果很相似的物品进行再次聚类。不同于常见的聚类过程，为了比较两个聚类结果的相似性，我们试图将聚类结果用有利于直接比较的规范的形式表示，这种表示我们暂且称之为标准表示。标准表示是由在一个个标准单位上的取值大小构成。我们认为这里所谓的标准表示可以有许多不同的定义，不过在本文中，标准单位是一个个实数区间。我们将区间 $[0,1]$ 等分成 n_b 个子区间，我们称之为桶。所有桶构成标准单位。我们用用户在桶上的分布来表示相对于每个物品的聚类结果。每个用户根据其自身的特征向量与物品特征向量的距离进入相应的桶：

$$bucket[u][i] = \lfloor f(\mathbf{q}_u, \mathbf{p}_i) n_b \rfloor$$

其中 $bucket[u][i]$ 代表了用户 u 相对于物品 i 的桶标号。因为用户和物品特征之间的距离可能分布在很大区间内，我们首先将距离进行标准化映射到区间 $[0,1]$。函数 f 正是我们采用的标准化函数，度量用户 u 和物品 i 之间的亲近程度。这个函数有很多可能的取法，不过在本文：

$$f(\mathbf{q}_u, \mathbf{p}_i) = e^{-\alpha(\|\mathbf{q}_u - \mathbf{p}_i\|)} \tag{3.5}$$

我们称这个函数为桶映射函数，其中参数 α 为桶分辨率参数。

继续用2.4节中的例子 (图2.4)。我们可以认为每个桶表示了用户对这部电影的喜欢程度，比如我们为每部电影设定，很喜欢，觉得还行和讨厌三种程度，分别对应三个桶，然后我们按照用户对电影的喜欢程度为每个用户分配不同的桶标号，结果如图3.1

至此，每个物品便可以以用户在桶上的分布来重新表示。

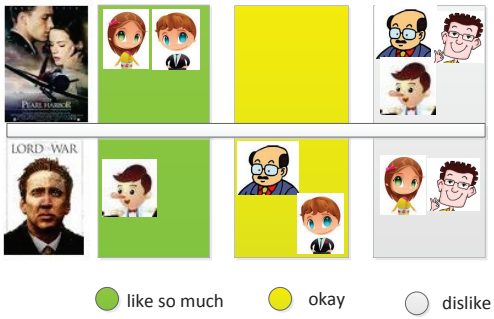


图 3.1: 用户相对于物品对齐示意图

3.2　物品聚类

现在，每个物品可以用用户聚类结果重新表示。物品 i 可以表示为：

$$I_i = \{B_{i1}, B_{i2}, \ldots, B_{in_b}\}$$

，其中 B_{il} 表示物品 i 的第 l 个桶中的用户集合。对于每个桶，我们考察每个物品在其上的用户集合，如果在桶 B_l 上物品 i 和物品 j 拥有很类似的用户集合，那么物品 i 和物品 j 就应该在同一个类中。对于集合之间的相似度度量，我们这里选择 Jaccard 相似度：

$$Jaccard(B_{il}, B_{jl}) = \frac{|B_{il} \bigcap B_{jl}|}{|B_{il} \bigcup B_{jl}|} \tag{3.6}$$

注意，当物品 i 和物品 j 相对于桶 l 属于同一个类，也即意味着物品 i，物品 j 以及用户集合 B_{il} 和 B_{jl} 中的用户属于同一个类。

3.2.1　通过 MinHashing 计算物品之间的相似度

由于每个桶中可能的用户是整个推荐系统的全体用户，这是一个很大的集合。如果我们采用逐一比较两个集合元素的方法去计算集合之间的真实 Jaccard 相似度，那将是非常耗时，需要的时间为 $O(n^2m^2n_b)$。我们不得不去寻找高效的计算方法。Google 新闻推荐中 [38]，将两个用户看过的新闻集合之间的 Jaccard 系数作为他们的相似度。由于新闻数量高达百万级别，不可能在线地一一比较集合元素来计算相似度，他们采用 Min-Hashing 来计算两个用户之间的 Jaccard 相似度。

在数据挖掘中，我们会遇到在高维空间中寻找相似结构并进行聚类的问题。这种问题中往往维数很高，但是仅仅只有少部分的对象之间的相似度比较高，其余的对象之间的相似度都非常低。要准确计算每两个对象之间的相似度需要在很高维的空间中逐维比较，比如对于集合对象之间的相似度计算需要的时间便是 $O(n^2)$，n 就是集合对象可能元素的全集。在很多应用问题中这都是一个非常巨大的集合。从而准确计算高维空间中每两个对象的准确相似度所耗费的时

间代价对于在线环境来说是不可接受的。所以，如果能找到一种算法，即使牺牲了一部分精度，但是如果能够将计算时间大幅度减少，这种算法还是可以接受的。而 Min-Hashing[39] 则先给每个对象产生一个签名，通常签名的产生速度与 n 成线性关系并且签名通常远远小于 n,，然后通过签名的相似度计算来估计真实的相似度。而签名之间的相似度计算很快，一则因为签名的维数远远小于 n，另外签名之间的比较是每一维一一对应的只需要线性的时间，而非原始集合之间的比较需要平方的时间。

我们现在来看看当相似度为 Jaccard 系数时 Min-Hashing 的原理。[40] 中为了剔除很相似的文档，需要计算两个文档之间的相似度。首先计算每个文档 D 的所有长度为 w 的子串集合 $S(D,w)$(注意这可以是一个多重集合)，文档 A 和 B 之间的相似度定义为：

$$r_w(A,B)=\frac{|S(A,w)\bigcap S(B,w)|}{|S(A,w)\bigcup S(B,w)|}$$

为了近似计算 $r_w(A,B)$，[40] 提出首先给整个语料库中全部长度为 w 的子串用 l 比特进行随机编码，设随机编码函数为 f，编码函数是从一个特殊的函数族中随机挑选的一个。然后近似的相似度计算如下：

$$r_{wf}(A,B)=\frac{|f(S(A,w))\bigcap f(S(B,w))|}{|f(S(A,w))\bigcup f(S(B,w))|}$$

也就是用子串编码的集合之间的相似度近似原来的相似度。作者还指出这种近似的误差有上界。具体地，下面的不等式以概率 99.9% 成立：

$$|r_{wf}(A,B)-r_w(A,B)|<\frac{|S(A,w)\bigcup S(B,w)|}{2^{l-11}}$$

其实 f 可以视作一个随机排列函数，他将整个语料库中全部长度为 w 的子串随机地排序，子串 s 的序号便是 $f(s)$，每个集合的签名便是 $f(S)$。那么排列后两个集合的元素的最小序号相等的概率就是他们 Jaccard 相似度的无偏估计，即：

$$Pr(min\{f(S(A,w))\}=min\{f(S(B,w))\})=r_w(A,B)$$

但是对于很大的集合，要产生随机排列仍然是很费时的任务。幸运的是，我们可以用 hash 函数来模拟随机排列。我们用 [41] 中的一个例子来说明具体的做法。在表3.1中，每行代表一个元素，共有 4 个集合 S_i，如果行代表的元素 j 在相应的集合 S_i 中，那么就将矩阵的第 j 行第 i 列的元素设置为 1，否则设置为 0。为了模拟集合元素的随机重排，我们采用两个 hash 函数 h_i，h_i 将原来序号为 x 的元素重新排列并且新的序号为 $h_i(x)$。

每个集合 S 的签名就是 $[h_{1,min}(S),h_{2,min}(S)]$。其中 $h_{i,min}(S)$ 就是重新排列后属于集合 S 的元素中的最小序号。从而我们可以计算出表3.1中每个集合的签名如表3.2：

表 3.1: Min-Hashing 示例 1

行	S_1	S_2	S_3	S_4	$h_1:(x+1)mod\ \ 5$	$h_2:(3x+1)mod\ \ 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

比如集合 S_1 和 S_3 的签名 $[1,0]$ 和 $[0,0]$，他们的第一个元素不同而第二元素相同，从而相似度为 0.5，而他们之间的真是相似度为 0.25。我们计算集合的最小 hash 签名之间的相似度并作为集合之间 Jaccard 相似度的估计值：

$$\widehat{J}accard(A,B) = Jaccard(Sig(A), Sig(B)) = \frac{\sum_{k=1}^{K_h} \delta(Sig_k(A) = Sig_k(B))}{K_h}$$

当我们选择足够多的 hash 函数，那么我们的近似结果将越来越好。

表 3.2: Min-Hashing 示例 2

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

在推荐系统中，因为用户的数量往往也是巨大的，而且相似度的计算需要快速进行。在这篇论文中，我们也采用 Min-Hashing 来快速计算两个物品在每个桶上的用户集合之间的 Jaccard 相似度。算法3.2给出了详细的过程。

1: 给全部用户进行编号，$\{1,2,3,\ldots\}$。令 $k=1$，
确定签名长度 K_h，输入用户个数 m。
2: **while** $k \leq K_h$ **do**
3: 随机产生一个区间 $[1,m]$ 内的整数 ξ。
4: 给每个物品 i 产生第 k 个签名 $h_l(i,k)$：
$h_l(i,k) = min_{u \in B_{il}}\{(No(u)+\xi)\%P\}$
5: k++.
6: **end while**
7: **for** each item i **do**
8: **for** each item j **do**
9: $\widehat{J}accard_l(i,j) = (\sum_k \delta(h_l(i,k) = h_l(j,k)))/K_h$.
10: **end for**
11: **end for**

算法 3.2: 通过 Min-Hashing 估计物品在桶 l 上的 Jaccard 相似度

其中 P 是一个大于 m 的最小质数。$No(u)$ 是用户 u 最初的编号。需要注意的是，我们可以发现物品之间的相似度往往很低，所以我们将相似度低于阈值的置为 0，最后的相似度矩阵是一个维数很高同时很稀疏的矩阵，我们将用稀疏格式存储相似度矩阵。

3.2.2　通过改进的谱聚类对物品进行聚类

在我们得到物品之间的相似度之后，我们需要对物品进行聚类。尽管存在很多优秀的聚类算法，不过在本文中，我们决定采用谱聚类。谱聚类将对象映射成无向图中的点，对象之间的相似度为点之间的边的权值，然后基于一些准则设计出合适的图划分算法。谱聚类在图像分割，电路设计，文档分析中都得到了广泛的应用。著名的谱聚类方法有：ratio-cut[42] 和 normalized-cut[43]。

Normalized-cut 算法力图将无向图的顶点集划分成多个不相交的类，使得类间的相似度很低，类内的相似度很高，同时类的大小要尽量平衡。将图 $G=(V,W)$(W为边权矩阵) 划分成两个部分 $\mathcal{V}_1,\mathcal{V}_2,\mathcal{V}_1\bigcup\mathcal{V}_2=V,\mathcal{V}_1\bigcap\mathcal{V}_2=\emptyset$，那么划分的代价为：

$$cut(\mathcal{V}_1,\mathcal{V}_2)=\sum_{u\in\mathcal{V}_1,v\in\mathcal{V}_2}w(u,v)$$

不过这样的目标函数将导致孤立的点自成一类，所以修改目标函数为：

$$Ncut(\mathcal{V}_1,\mathcal{V}_2)=\frac{cut(\mathcal{V}_1,\mathcal{V}_2)}{assoc(\mathcal{V}_1,V)}+\frac{cut(\mathcal{V}_1,\mathcal{V}_2)}{assoc(\mathcal{V}_2,V)}$$

其中 $assoc(\mathcal{V}_1,V)=\sum_{u\in\mathcal{V}_1,v\in V}w(u,v)$，我们最小化 $Ncut(\mathcal{V}_1,\mathcal{V}_2)$ 就是希望划分的 cut 最小同时划分的两个类大小比较均匀，这里类的大小用 $assoc(\mathcal{V}_1,V)$ 表示。$Ncut(\mathcal{V}_1,\mathcal{V}_2)$ 度量了类间距离距离，而类内距离定义为：

$$Nassoc(\mathcal{V}_1,\mathcal{V}_2)=\frac{assoc(\mathcal{V}_1,\mathcal{V}_1)}{assoc(\mathcal{V}_1,V)}+\frac{assoc(\mathcal{V}_2,\mathcal{V}_2)}{assoc(\mathcal{V}_2,V)}$$

因为有关系：

$$Ncut(\mathcal{V}_1,\mathcal{V}_2)=2-Nassoc(\mathcal{V}_1,\mathcal{V}_2)$$

所以最小化 $Ncut(\mathcal{V}_1,\mathcal{V}_2)$ 等价于最大化 $Nassoc(\mathcal{V}_1,\mathcal{V}_2)$，也就是最小化类间距离等价于同时最大化类内距离。而 Ratio-cut 算法的目标函数为：

$$\text{Ratio-cut}(\mathcal{V}_1,\mathcal{V}_1)=\frac{cut(\mathcal{V}_1,\mathcal{V}_2)}{|\mathcal{V}_1|}+\frac{cut(\mathcal{V}_1,\mathcal{V}_2)}{|\mathcal{V}_2|}$$

即类的大小等于类中点的个数。

可以看出 Ratio-cut 算法和 Normalized-cut 算法的不同在于类的大小的定义，进一步讲，我们觉得是每个顶点在聚类中的影响力的定义不同。Normalized-cut 算法中顶点关联边的权值和越大，这个点在聚类中的影响力就越大，而 Ratio-cut 算法中所有点的影响力是一样的。问题是，究竟怎么理解顶点的聚类影响力呢？在我们看来，所谓点在聚类过程中的影响力其实就是指这个顶点能够担任类领导角色的能力大小。Normalized-cut 算法中一个点关联边的权值和越大，那么他成为类领导的能力力也就越大，也就是他单独领导一个类的机会越大。当一个点的领导力越大时，那么周围的一些点就越容易进入该点所在的类。Normalized-cut 算法中两个关联边权值和都很大的顶点几乎不可能在同一个类

中，因为这样会导致该类的大小大大增加，从而破坏追求平衡类划分的目标。相反，两个关联边权值和都很大的顶点很容易在两个不同的类中。领导力很大顶点几乎很难再同一个类中，而是分布在不同的类中，从而在各自的类中充当领导角色。这样的理解对于我们针对具体应用问题的谱聚类具有重要的指导意义。

在 [44] 中，作者给每个顶点 v 赋一个正权重 $weight(v)$，并令

$$\mathbf{W_g} = \begin{bmatrix} weight(1) & 0 & \cdots & 0 \\ 0 & weight(2) & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & weight(|V|) \end{bmatrix}$$

然后最小化如下目标函数：

$$\mathcal{Q}(\mathcal{V}_1, \mathcal{V}_2) = \frac{cut(\mathcal{V}_1, \mathcal{V}_2)}{weight(\mathcal{V}_1)} + \frac{cut(\mathcal{V}_1, \mathcal{V}_2)}{weight(\mathcal{V}_2)} \tag{3.7}$$

其中 $weight(\mathcal{V}_i) = \sum_{v \in \mathcal{V}_i} weight(v)$，这其实也是一种类大小的定义方法。当我们用如下方式定义一个顶点划分时：

$$q_i = \begin{cases} +\sqrt{\frac{\eta_2}{\eta_1}}, i \in \mathcal{V}_1 \\ -\sqrt{\frac{\eta_1}{\eta_2}}, i \in \mathcal{V}_2 \end{cases} \tag{3.8}$$

其中，$\eta_i = weight(\mathcal{V}_i)$，那么我们就可以重新表示式3.7,

$$\mathcal{Q}(\mathcal{V}_1, \mathcal{V}_2) = \frac{\mathbf{q}^T \mathbf{L} \mathbf{q}}{\mathbf{q}^T \mathbf{W_g} \mathbf{q}}$$

其中 **L** 是图的 Laplacian 矩阵，$\mathbf{L} = \mathbf{D} - \mathbf{W}$，**D** 是一个对角阵，第 i 个主对角元素是 **W** 的第 i 行元素之和。我们的目的就是希望最小化 $\mathcal{Q}(\mathcal{V}_1, \mathcal{V}_2)$。当我们不再限定 **q** 的取值如3.8而是允许他在实数范围内取任意值时，这就变成了广义 Rayleigh 商，而广义 Rayleigh 商的最小化解就是广义特征值问题:

$$\mathbf{L}\mathbf{x} = \lambda \mathbf{W_g} \mathbf{x}$$

的最小特征值对应的特征向量。然而，我们可以发现 **e** 是最小特征值 $\lambda = 0$ 对应的特征向量，但是这是一个平凡的划分，没有意义。所以我们的问题变为：

$$\begin{cases} \min & \dfrac{\mathbf{q}^T \mathbf{L} \mathbf{q}}{\mathbf{q}^T \mathbf{W_g} \mathbf{q}} \\ \text{s.t} & \mathbf{q} \neq 0, \mathbf{q}^T \mathbf{W_g} \mathbf{e} = 0 \end{cases}$$

这个问题的解就是广义特征值问题:

$$\mathbf{L}\mathbf{x} = \lambda \mathbf{W_g} \mathbf{x} \tag{3.9}$$

的第二小特征值对应的特征向量。顺便指出的是，因为 $\mathbf{W_g}$ 正定，$\mathbf{L}$ 半正定，所以上述广义特征值问题的解存在且均为大于等于 0 的实数。由于广义特征值求解起来比标准特征值问题更加费时，所以我们可以将其转换为标准特征值问题：

$$\mathbf{W_g}^{-\frac{1}{2}}\mathbf{L}\mathbf{W_g}^{-\frac{1}{2}}\mathbf{x} = \lambda\mathbf{x}$$

这里注意因为 $\mathbf{W_g}$ 是对角矩阵，逆矩阵 $\mathbf{W_g}^{-\frac{1}{2}}$ 很容易求得，并且 $\mathbf{W_g}^{-\frac{1}{2}}$ 也是对角矩阵，从而在与其他矩阵的乘法中是可交换的，而这里交换矩阵乘法的顺序的目的在于使得 $\mathbf{W_g}^{-\frac{1}{2}}\mathbf{L}\mathbf{x}\mathbf{W_g}^{-\frac{1}{2}}$ 是对称矩阵，这将更有利于求解。

为了使谱聚类更适合协同过滤这一具体的环境，我们决定恰当地改进它。现在我们知道了，给图中的顶点赋更大的权重有利于使他们成为类的领导角色。而在协同过滤中，我们认为物品已有评分的个数越多，那么他们的影响力就越大，导致很多其他物品会参照他们进行聚类。所以我们决定最小化如下目标函数：

$$\frac{cut(\mathcal{V}_1,\mathcal{V}_2)}{(1-\rho)nassoc(\mathcal{V}_1,V)+\rho R(\mathcal{V}_1)}+\frac{cut(\mathcal{V}_1,\mathcal{V}_2)}{(1-\rho)nassoc(\mathcal{V}_2,V)+\rho R(\mathcal{V}_2)} \tag{3.10}$$

其中，$R(\mathcal{V}_1)=\sum_{i\in\mathcal{V}_1}T_i/T$。$T_i$ 是物品 i 现有评分的个数，T 是目前总共的评分个数。ρ 是一个调节参数。注意，这里我们对顶点的边权和进行了标准化以便每个顶点的边权和在 0 到 1 之间，这样的标准化会影响最优值，但是不会影响最优解。

$$nassoc(\mathcal{V}_i,V)=\frac{assoc(\mathcal{V}_i,V)}{assoc(V,V)}$$

换句话说，我们定义每个顶点的权重如下：

$$weight(i)=(1-\rho)\frac{\sum_{j\in V}W(i,j)}{\sum_{i,j\in V}W(i,j)}+\rho T_i/T$$

另外当预定的类的个数不是 2 时，而是 k_l，我们可以求方程3.9 的第 2 小，第 3 小，直到第 $\lceil log(k_l)\rceil$ 小的特征值对应的特征向量，然后再对得到的特征向量进行 k-means 聚类得到最终的类划分。

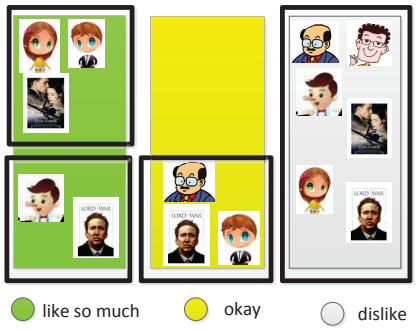


图 3.2: 物品聚类后的块结构示意图

继续用图2.4所示例子，在不喜欢这个桶上，珍珠港和战争之王两部电影的相似度是 0.25，在这个例子中是最高的相似度，所以在这个桶上，我们将这两部电影放在同一个类中，同时将不喜欢这两部电影的用户并集也放在这个类中。如此，形成一个个块结构 (图3.2)。

3.3　基于分块的推荐预测

我们认为用户对物品的反馈受两大类因素影响。一个是他和物品各自本身固有的属性，我们称之为全局特征。另一个则是受周围邻近的用户或物品的影响，我们称之为局部特征。我们将通过在整个评分矩阵中学习来获得全局特征，而在每个块中学习局部特征。

3.3.1　全局预测特征学习

矩阵分解方法已经充分证明了他在协同过滤中提取全局预测特征的能力，而且在我们看来，他还有另外一个优势在于他不使用任何相似度量。所以我们决定采用这种方法来学习用户和物品的全局特征。

类似于 RSVD，我们给每个用户 u 学习一个 k 维特征 $\mathbf{p}_u$，为每个物品 i 学习一个 k 维特征 $\mathbf{q}_i$。另外，因为现实中一些用户比较慷慨给物品的评分都比较高，除非他特别讨厌从而给一个很低的评分，而另外一些用户则很挑剔，除非特别喜欢某个物品从而给一个很高的评分，大多数情况下，他们都给出很低的评分，同样的，物品也存在类似的问题。所以我们采用 Improved Regularized SVD[22]，预测公式如下：

$$\widehat{r}_{u,i} = \mu + b_u + b_i + \mathbf{p}_u^T \mathbf{q}_i$$

其中 b_u 表示用户的评分尺度，b_i 表示物品的得分尺度。μ 则表示全部已知评分的均值。我们的目标函数如下：

$$\mathcal{L} = \sum_{(u,i)|r_{u,i}\text{已知}} \left((r_{u,i} - \mu - b_u - b_i - \mathbf{p}_u^T \mathbf{q}_i)^2 + \lambda(\|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2 + b_u^2 + b_i^2) \right) \quad (3.11)$$

我们采用随机梯度下降法来求解这个目标函数的最小值。对 $\mathcal{L}$ 中的未知变量求导有：

$$\frac{\partial \mathcal{L}}{\partial b_u} = \sum_{i|r_{u,i}\text{已知}} (-2(r_{u,i} - \widehat{r}_{u,i}) + 2\lambda b_u)$$

$$\frac{\partial \mathcal{L}}{\partial b_i} = \sum_{u|r_{u,i}\text{已知}} (-2(r_{u,i} - \widehat{r}_{u,i}) + 2\lambda b_i)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{p}_{uk}} = \sum_{i|r_{u,i}\text{已知}} (-2(r_{u,i} - \widehat{r}_{u,i})\mathbf{q}_{ik} + 2\lambda \mathbf{p}_{uk})$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{q}_{ik}} = \sum_{u|r_{u,i}\text{已知}} (-2(r_{u,i} - \widehat{r}_{u,i})\mathbf{p}_{uk} + 2\lambda \mathbf{q}_{ik})$$

从而按照梯度下降法，我们可以得到如下的迭代公式：

$$b_u = b_u - lrate * \frac{\partial \mathcal{L}}{\partial b_u}$$

$$b_i = b_i - lrate * \frac{\partial \mathcal{L}}{\partial b_i}$$

$$\mathbf{p}_{uk} = \mathbf{p}_{uk} - lrate * \frac{\partial \mathcal{L}}{\partial \mathbf{p}_{uk}}$$

$$\mathbf{q}_{ik} = \mathbf{q}_{ik} - lrate * \frac{\partial \mathcal{L}}{\partial \mathbf{q}_{ik}}$$

$lrate$ 是学习速率。当我们依次带入每个已知评分而不是一次性带入全部评分时且我们令 $r_{u,i} - \widehat{r}_{u,i} = e_{ui}$，我们便得随机梯度下降法的迭代公式，也即，每带入一个已知评分 $r_{u,i}$，则按如下方式更新各个未知变量。

$$\begin{cases} b_u = b_u + 2lrate * (e_{ui} - \lambda b_u) \\ b_i = b_i + 2lrate * (e_{ui} - \lambda b_i) \\ \mathbf{p}_{uk} = \mathbf{p}_{uk} + 2lrate * (e_{ui}\mathbf{q}_{ik} - \lambda \mathbf{p}_{uk}) \\ \mathbf{q}_{ik} = \mathbf{q}_{ik} + 2lrate * (e_{ui}\mathbf{p}_{uk} - \lambda \mathbf{q}_{ik}) \end{cases} \tag{3.12}$$

需要补充的是，我们先用全部评分训练第一维特征，然后再全部评分训练第二维特征，所以迭代中 k 是当前正在训练的特征中维的编号。

3.3.2　块内预测特征学习

为了提取局部预测特征，我们决定采用 RSVD，不过我们首先用上一节的方法做出全局预测，然后对残差评分进行学习，也即我们最终的预测公式是：

$$\widehat{r}_{u,i} = \overbrace{\mu + b_u + b_i + \mathbf{p}_u^T \mathbf{q}_i}^{global} + \overbrace{(\mathbf{p}_u^b)^T \mathbf{q}_i^b}^{block} \tag{3.13}$$

其中 $b = block(u,i)$，是用户 u 和物品 i 结合的块的编号，$\mathbf{p}_u^b \in \mathbb{R}^k, \mathbf{q}_i^b \in \mathbb{R}^k$ 是根据 (u,i) 所在块中的残余评分学习而得。我们首先对每个训练评分和测试评分根据全局特征做出全局预测，接着我们将每个训练评分和测试评分的真实值减去他的全局预测值，得到的差我们称之为残余评分，残余评分根据其关联的用户和物品位于某个块中，然后，我们将块当做相互独立的小矩阵，在每个块中对残余的评分使用 RSVD，学习到适用于块内预测的局部特征。

第四章　实验评价

这一章中，我们进行模拟实验，确定相关参数的最优设置，展示我们算法的性能并和其他算法比较。

4.1　评价方法

如何评价一个推荐算法是一个很关键的问题，这直接引领了推荐系统的研究潮流。评价用在实际电子商务网站中的推荐系统，最好的指标便是部署前后销售额的增长情况。然后，在离线实验中，很难通过这种方式去评价一个推荐算法。所以，研究者们提出了很多指标来度量一个算法的性能。不同的指标着重了算法不同的方面。

[45] 提出应该根据不用的用户任务来确定评价指标。主要的用户任务有两类。一是判断用户对每个物品的喜好，这种任务要求算法能更好地准确预测每个用户对每个物品的喜好程度。另一个是向用户推送一个他最有可能喜欢的物品列表，这是最常见的推荐任务。这种任务要求算法给出的物品列表中，尽可能都是用户真正喜欢的 (当然一般是用户还没有看过或者购买过的物品)。还有其他的用户任务，比如预测商品的购买序列等。现存的协同过滤评价指标基本都是评价算法的准确度，而不是算法的其他方面比如存储模型所需的存储空间或者算法的响应时间等等。[45] 把所有的准确度指标分为三类，预测精度指标，分类精度指标和排序精度指标。同一类中的不同指标在试验中显示了高度的关联性，，而不同类中的指标则几乎没有显示出关联性。

预测精度指标用来度量用户对物品喜好的预测值与真实值之间的差别。这类指标是迄今被使用最多的指标，代表性的有 MAE(Mean Absolute Error) 和 RMSE(root mean absolute error)。他们的定义分别如下：

$$MAE = \frac{1}{|test|} \sum_{(u,i) \in test} |r_{u,i} - \widehat{r}_{u,i}| \quad RMSE = \sqrt{\frac{1}{|test|} \sum_{(u,i) \in test} (r_{u,i} - \widehat{r}_{u,i})^2}$$

他们适合判断用户对每个物品喜好的任务，但是不适合进行列表推荐。因为一个很低的 MAE 可能是他很准确地预测了大量用户根本不感兴趣的物品的评分，但是对用户感兴趣的少量物品的预测值却很差，从而由预测值导出的推荐列表将是很不准确的。

分类精度指标适合二元数据或者可以二元化的数据，只考虑用户对物品的喜欢或者不喜欢两种情况，这种指标适合于为用户找出他喜欢的物品的任务。这类指标中最常用的便是从信息检索中改造的精度 (Precision) 和召回率 (Recall)。然而，信息检索中的这两个指标尽管很成熟，却不能直接用于协同过滤。当算法给用户一个物品列表，并认为用户喜欢这个列表中的全部物品，而不在这个

列表中的物品，算法便认为用户不喜欢，这样算法便给出了每个物品的预测类标号。因为数据的稀疏性，用户没有对很多物品进行评价，我们无法得知很多物品的真实类标号，故而精度和召回率无法计算。在协同过滤中，已经提出了好几种方法改造精度和召回率的计算方式，我们这里介绍 [46] 的做法。[46] 将已有数据划分成训练集和测试集，在训练集上学习并为每个用户给出一个大小为 N 的推荐物品列表，记为 $top(u)$，然后精度和召回率按如下计算：

$$recall = \frac{\bigcup_{u=1}^{m}(test(u)\bigcap top(u))}{\bigcup_{u=1}^{m} test(u)}$$

$$Precision = \frac{\bigcup_{u=1}^{m}(test(u)\bigcap top(u))}{mN}$$

其中 $test(u)$ 是用户 u 在测试集中喜欢的的物品集合。然而，这种方法仍然存在稀疏性的问题，如果用户组评价过的物品中他喜欢的很少，$test(u)$ 就很小，那么 Precision 就很小，即便算法给出了很多用户没评价过但是其实他会很喜欢的物品，这个算法的性能也会被认为很差。

当只注重用户对物品喜好的相对顺序时，排序精度指标的是合适的性能度量。这类指标度量算法给出的物品序和用户给出的物品序之间的距离。数学中，有一类被称作秩相关系数的量用来计算两个排序之间的距离，比如 Spearman 相关系数和 Kendall tau 距离。但是他们不能直接用于协同过滤，因为他们往往对排序中的每对物品给予相同的重要性，然而对于推荐算法来说，交换列表中第一项和第二项物品带来的用户体验差异和交换列表中第 1000 项和第 1001 项带来的用户体验差异显然是很不一样的，因为用户基本不会浏览列表中排在 100 以后的物品。还有一个不足之处，就是传统的秩相关系数要求不能两个物品有一样的序号。这里我们介绍一下 NDPM(normalized distance-based performance measure)，这个指标最早由 [47] 运用到协同过滤中。设用户 v 对物品的真是排序是 L_1，而算法给出的排序的是 L_2，那么 NDPM 计算如下：

$$NDPM(v) = \frac{2C^{-} + C^{u}}{2C}$$

C^{-} 是两个排序中冲突的物品对的数目，C^{u} 是两个排序中兼容的物品对数目，C 是 L_1 中有严格顺序的物品对数目。设 $\tau_k(i)$ 表示在 L_k 中物品 i 的序号，那么物品 i 和物品 j 排序冲突是指：

$$(\tau_1(i) > \tau_1(j) \wedge \tau_2(i) < \tau_2(j)) \vee (\tau_1(i) < \tau_1(j) \wedge \tau_2(i) > \tau_2(j))$$

物品 i 和物品 j 排序兼容是指：

$$(\tau_1(i) > \tau_1(j) \wedge \tau_2(i) = \tau_2(j)) \vee (\tau_1(i) < \tau_1(j) \wedge \tau_2(i) = \tau_2(j))$$

最后取全部用户 NDPM 的平均值最为整个系统的 NDPM。

除了上述三类准确度指标外，还有一些其他指标。其中比较有影响力的是覆盖率 (Coverage)，这个指标度量算法能够将整个物品集中多少物品推荐给合适的用户。

我们已经说过，对于预测精度指标来说，比如一个具有很低的 MAE 或者 RMSE 的算法可能只是准确预测了用户对大量他根本不感兴趣的物品的喜好程度，而对少部分用户很感兴趣的物品的喜好预测则很不准确。这种算法根据预测评分生成的推荐列表必然质量很差。这种算能很肯定告诉用户不喜欢那些物品，却不能告诉用户他喜欢的物品有哪些，必然容易引起用户不满。需要在向用户推送他可能喜欢的物品列表时，如果推荐列表中的预测评分很很差的话，那么推荐列表根本就是不可靠的。其实用户往往不在乎系统对他根本不会去观看或者购买的物品的预测值。而分类精度指标只是给出一张用户很喜欢的列表，却无法判断用户喜欢的程度，如果需要向用户推送推荐列表的同时需要展示系统的预测评分，这类指标将无法胜任。尽管可以考虑排序精度指标，然而这类指标以及分类精度指标存在一个共同的严重问题，那就是一些物品的真实类标号和序号无从得知，这时便需要采用近似方法，不过这却容易导致其他问题，比如过拟合。Cacheda 等人 [48] 将预测精度指标和分类精度指标进行结合，分别考虑用户真正喜欢的物品上的预测精度和系统认为用户喜欢的物品的预测精度，提出了两个新的指标 GIM(Good Item MAE) 和 GPIM(Good Predicted Item MAE)。GIM 计算那些用户真正喜欢的物品上的 MAE，GPIM 计算算法认为用户会喜欢的物品上的 MAE。具体的如下：

$$TP = \{(u,i)|r_{u,i} \geq R_g, \hat{r}_{u,i} \geq P_g, (u,i) \in test\}$$

$$FN = \{(u,i)|r_{u,i} \geq R_g, \hat{r}_{u,i} < P_g, (u,i) \in test\}$$

$$FP = \{(u,i)|r_{u,i} < R_g, \hat{r}_{u,i} \geq P_g, (u,i) \in test\}$$

当实际评分大于等于 R_g，我们就认为该用户确实喜欢该物品。当预测的评分大于等于 P_g 是，那么算法将认为该用户喜欢该物品。那么 GIM 和 GPIM 计算如下：

$$GIM = \frac{1}{|TP \bigcup FN|} \sum_{(u,i) \in TP \bigcup FN} |r_{u,i} - \widehat{r}_{u,i}|$$

$$GPIM = \frac{1}{|TP \bigcup FP|} \sum_{(u,i) \in TP \bigcup FP} |r_{u,i} - \widehat{r}_{u,i}|$$

类似地，我们可以提出 GIR 和 GPIR，如下

$$GIR = \sqrt{\frac{1}{|TP \bigcup FN|} \sum_{(u,i) \in TP \bigcup FN} (r_{u,i} - \widehat{r}_{u,i})^2}$$

$$GPIR = \sqrt{\frac{1}{|TP \bigcup FP|} \sum_{(u,i) \in TP \bigcup FP} (r_{u,i} - \widehat{r}_{u,i})^2}$$

在本文中，我们就采用 MAE，GIM，GPIM，RMSE，GIR，GPIR 一起来评价算法的性能。

4.2　实验设置

我们采用的数据集是 MovieLens ml-100k，这是协同过滤中很著名的一个数据集，它源自 Minnesota 大学的 GroupLens 项目。他包括 943 个用户对 1682 个电影的 100,000 个评分。评分的取值是 1-5 的整数值，每个用户至少已有 20 个评分。我们把随机地将已有评分划分成训练集和测试集，其中训练集包 80% 的已知评分，测试集包含 20% 的已知评分。我们在训练集上训练算法，然后根据算法在测试集上的性能来选择合适的参数并展示算法的最终性能。所有的实验均在一台普通 PC 上运行，2GB 的内存，2.6GHz 的单核 CPU。实验的软件环境为 Matlab 加 VS。

4.3　参数训练

我们首先需要提取用户和物品的统一特征，我们设置统一特征问题3.4中特征维数为 $k = 10, \lambda = 0.002, r_\theta = 3.5$，另外为了使得算法 3.1在适当的准则下停止迭代，我们判断当前迭代得到的迹比值 ρ_t 和上一轮迭代得到的迹比值 ρ_{t-1}，如果

$$\frac{\rho_t - \rho_{t-1}}{\rho_{t-1}} < 0.001$$

那么就算法停止。图4.1展示了的得到的物品特征和用户特征之间的欧几里得距离的分布。

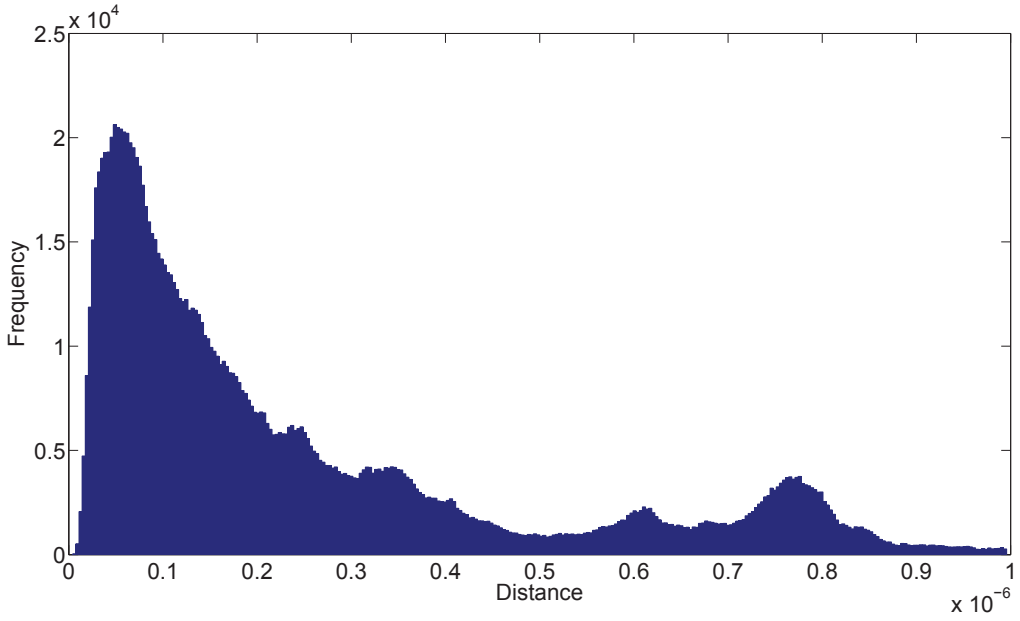


图 4.1: 用户特征与物品特征的欧式距离分布图

图4.1中横坐标是每个用户与每个物品之间欧式距离值，纵坐标是频数。可以看出距离基本都是在 10^{-6} 这样的数量级，而且大部分距离分布在靠近 0 的位置，也就是大部分用户和物品之间的距离靠近 0。为了根据用户和物品之间的欧几里得距离为每个用户分配合适的桶标号，我们采用一个桶映射函数 (方程3.5)，其中的桶分辨率参数 α 对我们块结构的形成是一个很关键的参数，图4.2 显示了一系列 α 取值下的桶映射函数图象，这可以使得我们可以确定 α 参数大概取值的范围。后续我们将选择这个范围内的不同的 α 取值并导出相应的块结构，然后比较这些块结构对性能的改进。

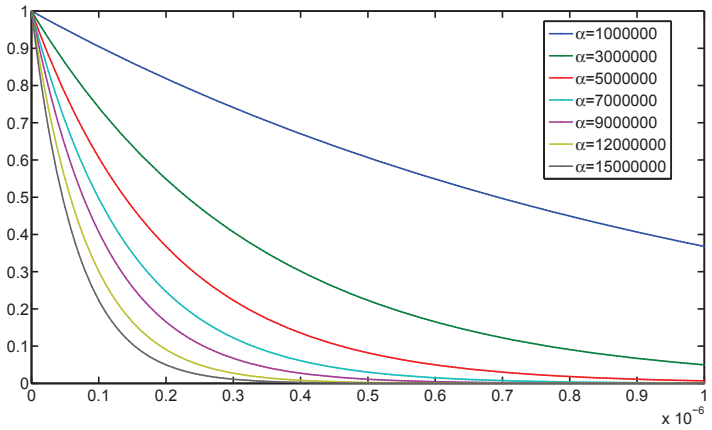


图 4.2: 不同 α 取值下的桶映射函数图象

我们利用 Min-Hashing 快速估计被表示成用户集合的物品之间的 Jaccard 相似度，我们需要首先利用 hash 函数计算每个集合的签名，然后基于集合的签名计算相似度。在本文中，我们的 hash 函数采用了最常用的取模函数，算法3.2中的参数 P 我们取为 1901，这是大于用户个数 1682 的最小的素数。签名的个数能够影响相似度的估计误差，为了取得误差和计算时间的折中，我们考察了不同的签名长度的 Min-hashing 进行相似度估计的误差 (图4.3)。

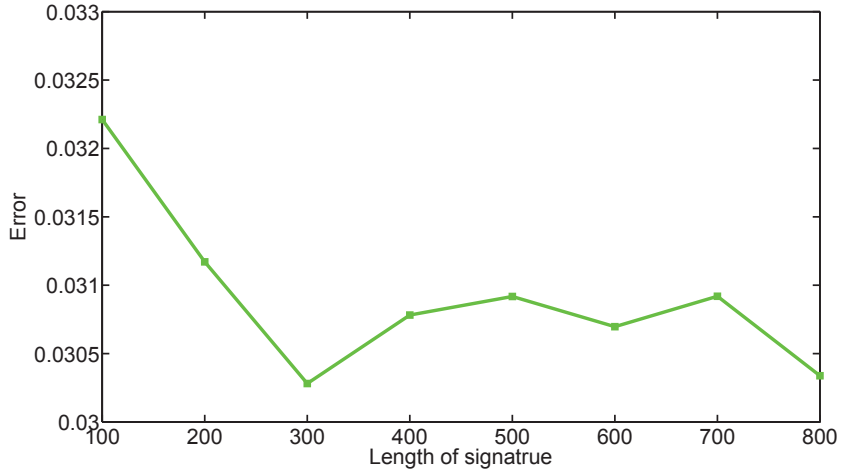


图 4.3: 用 Min-Hashing 估计相似度的误差

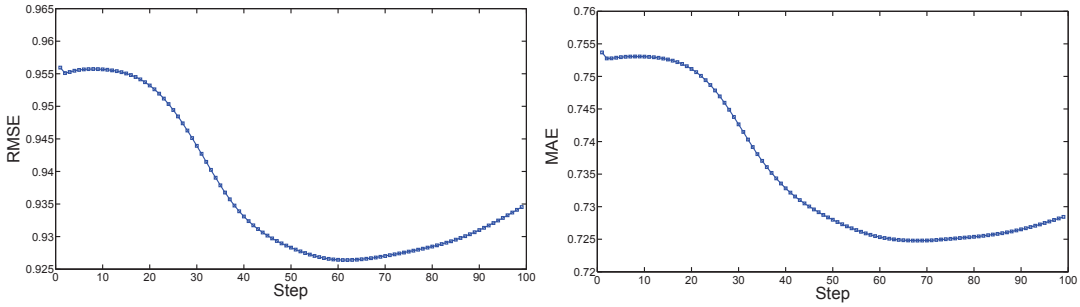
其中横坐标表示签名的长度，纵坐标是当我们设置 $\alpha = 5000000$，桶的个数取为 20，我们计算每个桶上物品之间的真实相似度和估计相似度之间差的绝对值，然后在每个桶上取平均而得到的。可以看出误差都很小，在 0.03 附近。因为我们的 hash 函数采用的随机种子，所以误差有小幅随机波动，不过我们多次试验发小，签名长度为 300 的时候取得了最小的误差，从而在后续的实验中签名的长度我们都设置为 300.

接下来，我们采用标准的 normalized-cut 对物品根据其相互的相似度形成的无向图进行划分。其实，我们利用 normalized-cut 为每个物品提取一个特征向量，然后采用 k-means 对特征向量进行聚类。试验中，为了兼顾计算速度和特征携带信息的能力，我们每次求解方程3.9时，我们都求前 10 个最小的特征值对应的特征向量，然后用第 2 到第 10 个最小特征向量中相应分量组成的向量作为每个物品的特征向量。另外，为了尽量减小 k-measns 中不良随机初始化的影响，我们重复 k-means 了 10 次，然后选择具有最小 SSE 的划分作为最后的结果。至此，我们可以开始提取块结构。

为了较好地组织块结构以便快速检索相关信息，我们首先为每个块分配与其关联的用户和物品并形成一个记录文件，块中的物品带有记号 1，块中的用户带有记号 2，文件中每个记录的格式为:

$$1 \quad itemNo \quad BlockNo, \text{或者}, 2 \quad userNo \quad BlockNo$$

并且我们建立了检索结构，记录每个块中用户列表起始和终止位置以及物品列表的起始和终止位置，使得可以快速找到每个块中的所有用户和物品，我们还建立了可以快速寻找一个用户和物品关联的所有块的索引文件。



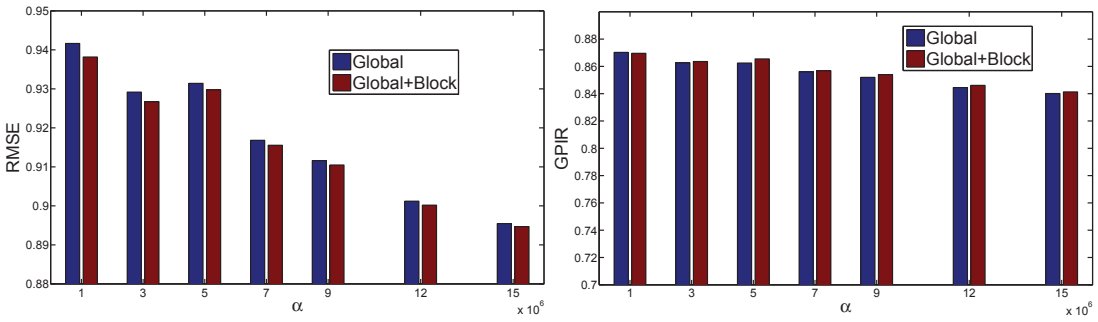
(a) RMSE　　(b) MAE

图 4.4: RSVD 随机梯度下降的迭代情况

我们采用 Improved Regularized SVD 进行全局预测特征学习。因为我们采用随机梯度下降法来求解问题3.11，为了考察这种算法的迭代特征点并确定停机准则，我们记录每一轮迭代后的 RMSE(图4.4a) 和 MAE(图4.4b)。可以看出迭代有一个局部极小点，我们应该停止在这里，然而因为最初的几轮迭代误差出现了轻微的上升，所以我们决定随机梯度下降法迭代 100 次并时刻记录已知最小误差对应的解，最后返回整个迭代中最小误差处的解。

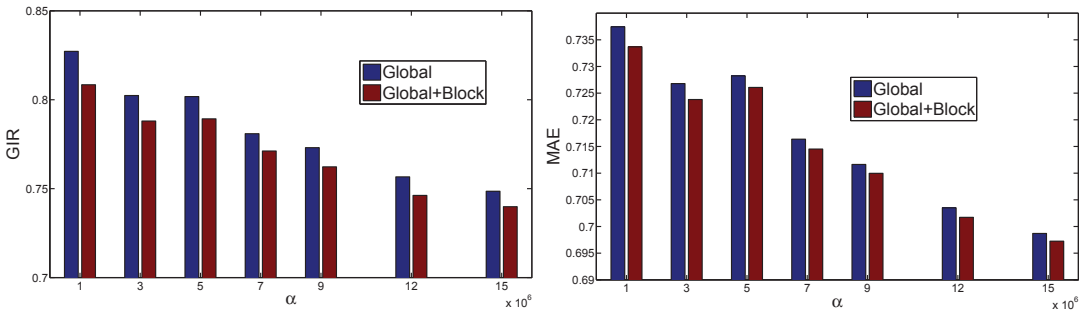
我们将每个已有评分的全局预测值附加在相应评分的末尾，然后根据已有块结构为每个块提取训练集和测试集。注意我们将一些只有很少用户或者物品的块在这一步去掉，因为如果一个块只有 1 个或者 0 个用户，那么他根本就不是一个我们期望意义的块。在实验中，我们规定如果快中用户个数少于 5 或者物品个数少于 5，我们就认为这些块结果缺乏意义并放弃在其内部进行局部预测。另外如果这个块已有的评分很少，我们则不进行块内学习。在实验中，我们规定块至少要有一个训练评分和一个测试评分，否则我们也放弃块内的学习。然后，我们对每个块中的用户和物品进行重新编号，从 1 开始，连续第为每个块中的用户或物品编号。最后，我们在每个块中独立地进行块内学习。我们令任何物品，任何用户在其关联的任何块中的局部特征的长度为 10，也就是方程3.13中，$\mathbf{p}_u^b, \mathbf{q}_i^b$ 的向量维数取为 10.

图4.5，4.6和图4.7展示了不同的桶分辨率参数 α 导出的块结构在各个指标上的改进。因为有些块我们没有进行块内学习，所以图中 Global 表示的是我们进行了块内学习的那些块中已有评分的全局预测误差，Global+Block 则表示我们将基于全部已知评分进行的全局预测和在块内基于块内残余评分进行的局部预测进行相加得到的最终预测的各个误差指标。注意，我们是将很多块中的预测一起计算相应的指标，而非每个取每个块相应指标的平均值。



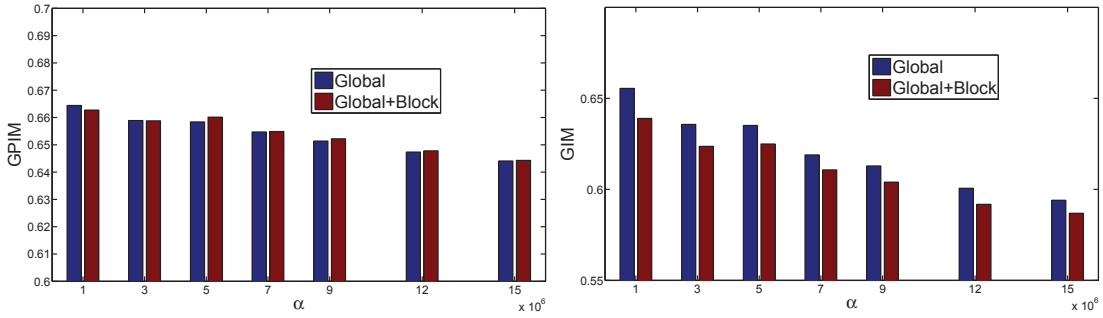
(a) RMSE　　(b) GPIR

图 4.5: 分块前后的性能改进 1



(a) GIR　　(b) MAE

图 4.6: 分块前后的性能改进 2

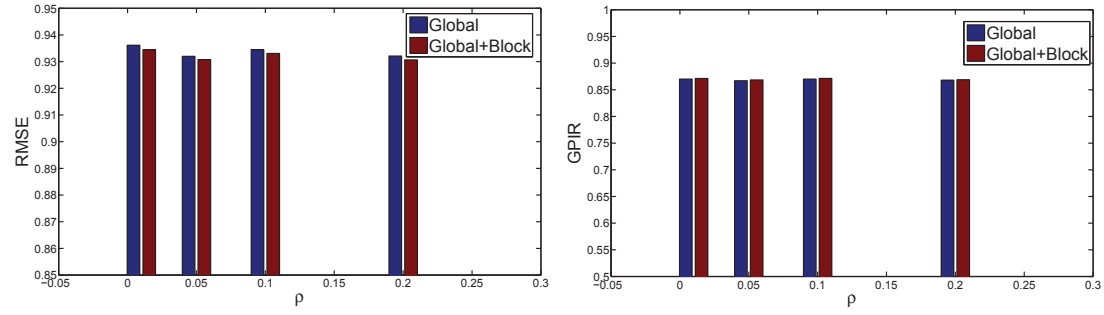


(a) GPIM　　(b) GIM

图 4.7: 分块前后的性能改进 3

在计算相关指标时，我们令 $R_g = 4.0, P_g = 3.6$ 是。我们可以看出在 RMSE 和 MAE 具有一定幅度的改进，而 GIR 和 GIM 则取得了大幅度的改进，在 GPIR 和 GPIM 则改进很小。也即是分块后，我们能够更加准确地预测用户对其真正喜欢物品的评分。我们还以发现 $\alpha = 3000000$ 时各个指标同时都取得了较好的改进，此时 GIR 由 0.8024 下降到 0.7880，GIM 由 0.6357 下降到 0.6236，改进比较明显，从而在接下来的实验中我们固定 $\alpha = 3000000$。

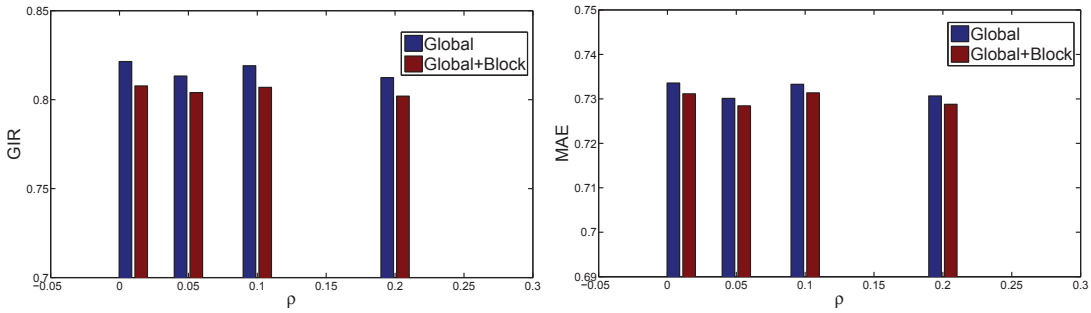
为了使谱聚类更适合协同过滤这一具体的应用环境，我们采用改进的谱聚类 (方程3.10）来对物品进行聚类。谱聚类中顶点被赋予更大的权重有利于使他们成为类的领导角色。在协同过滤中，我们认为物品已有评分的个数越多，那么他们的影响力就越大。我们通过参数 ρ 来控制物品在聚类时的影响力在其已有评分多少和他在相似图中的边权和之间的折中程度，他是一个取值在 0 到 1 之间的数，我们需要确定这个参数的最优设置。



(a) RMSE　　(b) GPIR

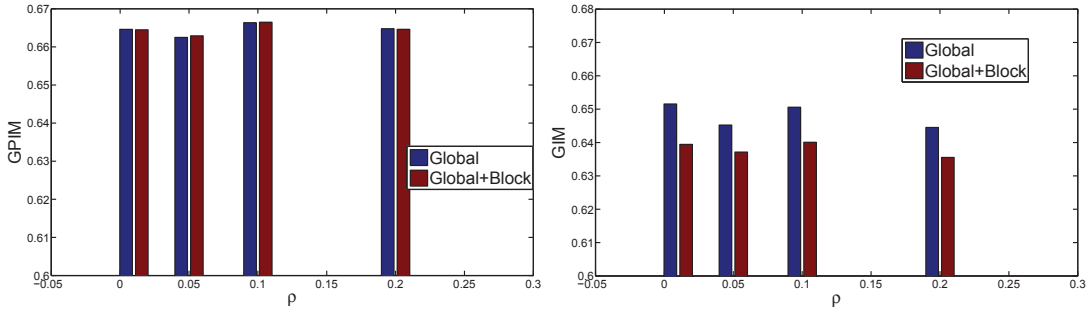
图 4.8: 采用改进的谱聚类进行分块前后的性能改进 1

保持其他参数不变，我们使 ρ 的取值变化，图4.8，4.9和图4.10展示了不同的 ρ 值导出的块结构在各个指标上的改进。我们可以看出在 GIR，MAE 和 GIM 都取得了较大的改进，而其他指标改进较小。这说明采用我们提出的改进的谱聚类方法对物品进行划分有助于更加准确地预测用户对其真正喜欢的物品的评分。当 $\rho = 0.01$ 时，各个指标都能去取得较好的改进，所以后续试验中，我们



(a) GIR　　(b) MAE

图 4.9: 采用改进的谱聚类进行分块前后的性能改进 2



(a) GPIM　　(b) GIM

图 4.10: 采用改进的谱聚类进行分块前后的性能改进 3

设置 $\rho = 0.01$。

现在，我们可以得出结论，分块以后，协同过滤的性能得到了改进，特别是，我们可以更加准确地预测用户对其真正喜欢物品的评分，从而可以改善用户的体验，提升用户对系统的满意度。另外，通过分块，我们还可以获得额外的好处，那就是我们可以更加有效地应对系统变化。随着时间的推移，用户会评价越来越多的物品，从而评分矩阵会不断地有新的评分出现，新评分出现对于我们更加准确地判断用户喜好以及他喜好的变迁有着重要作用。而很多协同过滤往往是在出现一大批新评分后才会对新评分一次性地处理，重新训练整个模型，这样对新评分的反应存在时延而且重新训练模型将花费大量时间。而我们的方法能有效适应数据变动。当新的评分到来时，我们只需要重新训练这个新评分所在块中的用户和物品的局部特征，并不需要全局特征，也不需要训练其他用户的任何特征。

接着，我们考察了所得块结构的大小。我们首先考察了每个块中多少物品和多少用户，如图4.11所示。横坐标表示每个块中用户的个数，纵坐标表示每个块中物品的个数。我们可以发现，很多块都位于图中左下角，说明很多块包含的用户个数和物品个数都较少。另外有一些块很靠近坐标轴，说明这些块中用户个数或者物品个数极少，这些块我们认为是没有意义的，所以我们放弃对这些块的局部学习。当我要求每个块的用户个数和物品个数不得少于 10 时，那么

块大小的分布如如4.12，此时块的数量从 634 个减少到 318 个。

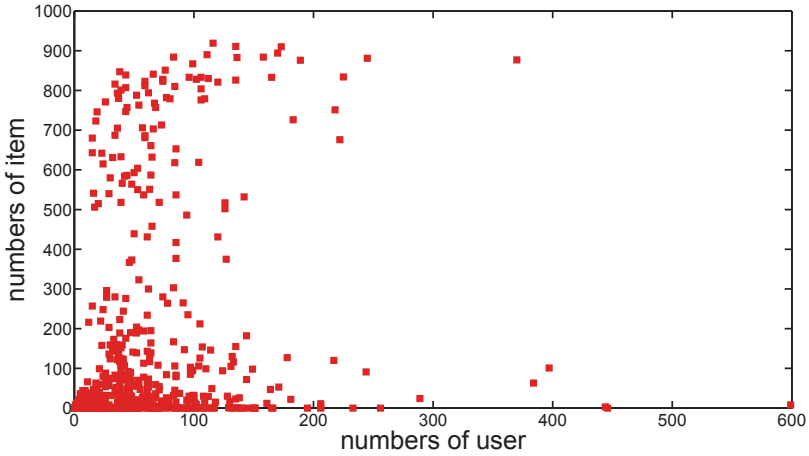


图 4.11: 块大小分布 (所有块)

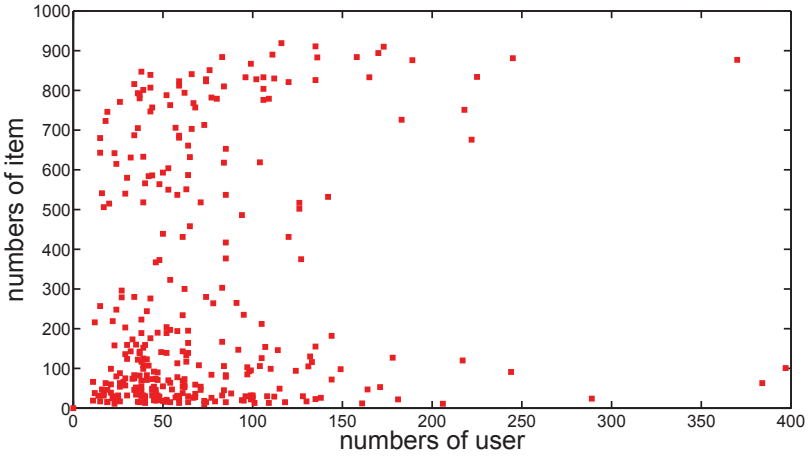


图 4.12: 块大小分布 (去掉无意义的块后)

4.4　实验对比

下面，我们将本文提出的方法和协同过滤中很有代表性的两个方法，RSVD[21] 和 IRSVD[22]，进行实验对比。我们将数据集 MovieLens ml-100k 做了两次随机划分并分别进行进行对比实验，以降低随机划分对实验结果的影响，每次将 80% 的已有评分作为训练集，20% 的已有评分作为测试集，然后我们展示了各个方法在测试集上每个指标下的性能，如表4.1 和表4.2 所示。

表4.1和表4.2中 PBCF-nc 代表采用标准的 normalized cut 算法对物品进行聚类的用户物品成对分块协同过滤算法 (PBCF，Collaborative Filtering Based on User-item Pairwise Blocking)，PBCF-inc 则表示采用改进的 normalized cut 算法对物品进行聚类的 PBCF。RSVD 方法和 IRSVD 方法的参数设置都为 $K = 30, lrate = 0.005, \lambda = 0.02$。对于 PBCF-nc 和 PBCF-inc，除了一些参数设置已经

表 4.1: 结果对比 (数据集划分 1)

	RMSE	GPIR	GIR	MAE	GPIM	GIM
RSVD	0.9419	0.8723	0.8440	0.7374	0.6679	0.6678
IRSVD	0.9282	0.8605	0.8039	0.7259	0.6572	0.6367
PBCF-nc	0.9258	0.8626	0.7880	0.7229	0.6575	0.6232
PBCF-inc	0.9337	0.8719	0.8010	0.7229	0.6638	0.6333

表 4.2: 结果对比 (数据集划分 2)

	RMSE	GPIR	GIR	MAE	GPIM	GIM
RSVD	0.9319	0.8523	0.8460	0.7290	0.6539	0.6678
IRSVD	0.9282	0.8554	0.8256	0.7258	0.6573	0.6513
PBCF-nc	0.9254	0.8560	0.8076	0.7223	0.6565	0.6360
PBCF-inc	0.9265	0.8590	0.8092	0.7237	0.6586	0.6381

在前文所述外，两个重要的参数设置分别为，在 PBCF-nc 中 $\alpha = 3000000, \rho = 0$，在 PBCF-inc 中 $\alpha = 3000000, \rho = 0.01$。

我们可以发现 PBCF-inc 的性能得到了较大改进，在两次划分中，在其他 4 个指标 (RMSE，MAE，GPIR，GPIM) 上，PBCF-inc 相对于 RSVD 和 IRSVD 有小幅改进或者持平，不过在 GIR 和 GIM 这两个指标上改进明显。其中 GIR 相对于 RSVD，分别改进了约 0.06(划分 1)，0.04(划分 2), 而 GIM 分别改进了 0.04(划分 1)0，0.03(划分 2)。GIR 相对于 IRSVD，分别改进了 0.02(划分 1)，0.04(划分 2), 而 GIM 分别改进了 0.01(划分 1)，0.01(划分 2)。而 GIR 和 GIM 度量的是对于用户真正喜欢的那些物品的预测评分的准确度，也即，我们的方法能够更加准确地预测用户对其真正喜欢的物品的评分，推荐出的物品是用户真正会喜欢的，从而带给用户更好的使用体验，赢得用户对系统的信任。

第五章　结论与展望

在这篇论文中，我们认为，用户类结构在面对不同物品时会不停地变化，而且，物品的类结构在面对不同的用户的时候也会不停地变化。我们提出物品和用户之间的相似关系应是相互依赖，并是相互动态地依赖。我们试图去发掘隐藏在评分矩阵中的块结构。我们用一个块结构来建立用户类和物品类以及刻画他们之间的相互作用，每个块由许多用户和物品组成，在这样的每个块中，用户对这个块中的物品持有很相似的看法，而物品在这个块的用户集中也有着类似的受欢迎程度。每一个块都是一个富有深意的被用户和产品共享的结构，反映了在现实中用户兴趣和物品属性之间的切合程度以及用户类和物品类对彼此形成的影响。

我们提出了用户物品成对分块算法的一般框架，这个框架包含用户和物品统一特征提取，用户和物品之间的对齐，物品或者用户基于对齐表示进行聚类并得到我们期望的块结构，最后我们利用已有的协同过滤算法进行全局和块内学习并给出未知评分的预测。我们给出一种这种框架的具体实现，将统一特征提取形式化为标准的 Trace Ratio 优化问题，通过 MinHashing 快速计算物品之间的相似度，并根据协同过滤这个具体的问题背景，我们通过改进的谱聚类对物品进行聚类，从而得到块结构。在得到块结构后，我们采用 Improved Regularized SVD 和 RSVD 进行全局和块内特征学习。

通过模拟实验表明我们的方法可以有效地提升推荐算法性能。分块以后，协同过滤的性能得到了改进，特别是，我们可以更加准确地预测用户对其真正喜欢物品的评分，从而可以改善用户的体验，提升用户对系统的满意度。另外，通过分块，我们可以有效地应对数据变动。当新的评分到来时，我们只需要重新训练这个新评分所在块中的用户和物品的块内特征，而不需要训练他们的全局特征。

下一步，我们可以进一步探讨框架中各个组件的其他具体实现。比如目前的特征提取方法可以采用新兴的深度学习 (deep learning) 框架 [49]，这种方法可以提取更高层次的特征，已经在多个应用问题取得了很好的效果。物品和用户之间的对齐可以考虑其他更合适的方法，比如可以参考人工神经网络中的各种 Sigmoid 函数，选择新的桶映射函数。很多聚类技术都可以运用到本文提出的框架中，比如经典聚类算法的核函数版本。特别地，针对协同过滤这个具体的应用问题，恰当地改进已有的聚类算法可以得到更好的效果。另外，我们认为，通过分块可以有效应对数据变动。当新评分到来时，我们只需要重新训练新评分所在的块中的用户和物品的块内特征，进行增量更新。不过目前还无法形式化地表示这种更新方法的合理性，后续可以推导出具体的更新公式以便同时证明更新策略的合理性，并通过实验验证增量更新策略导出的演进模型相对于进行

整体重新学习得到的模型在预测性能上的差异。最后，本文给出一种对二元表示的大型矩阵数据进行分块以便应对高维问题的思路，我们同时学习全局和块内特征以便用较小的计算代价捕获全局信息和局部信息。这种思路可以运用到很多其他具体问题中，比如生物信息学中的基因阵列数据分析。可以在进一步完善这种方法后将之运用到其他问题中。[50]

参考文献

[1] Su X, Khoshgoftaar T M. A survey of collaborative filtering techniques. Advances in artificial intelligence, 2009, 2009:4.

[2] Goldberg D, Nichols D, Oki B M, et al. Using collaborative filtering to weave an information tapestry. Communications of the ACM, 1992, 35(12):61–70.

[3] Herlocker J L, Konstan J A, Borchers A, et al. An algorithmic framework for performing collaborative filtering. Proceedings of Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1999. 230–237.

[4] Sarwar B, Karypis G, Konstan J, et al. Item-based collaborative filtering recommendation algorithms. Proceedings of Proceedings of the 10th international conference on World Wide Web. ACM, 2001. 285–295.

[5] Ma H, King I, Lyu M R. Effective missing data prediction for collaborative filtering. Proceedings of Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2007. 39–46.

[6] Resnick P, Iacovou N, Suchak M, et al. GroupLens: an open architecture for collaborative filtering of netnews. Proceedings of Proceedings of the 1994 ACM conference on Computer supported cooperative work. ACM, 1994. 175–186.

[7] Breese J S, Heckerman D, Kadie C. Empirical analysis of predictive algorithms for collaborative filtering. Proceedings of Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1998. 43–52.

[8] Karypis G. Evaluation of item-based top-n recommendation algorithms. Proceedings of Proceedings of the tenth international conference on Information and knowledge management. ACM, 2001. 247–254.

[9] Choi K, Suh Y. A new similarity function for selecting neighbors for each target item in collaborative filtering. Knowledge-Based Systems, 2013, 37:146–153.

[10] Sarwar B M, Karypis G, Konstan J, et al. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. Proceedings of Proceedings of the fifth international conference on computer and information technology, volume 1. Citeseer, 2002.

[11] O’Connor M, Herlocker J. Clustering items for collaborative filtering. Proceedings of Proceedings of the ACM SIGIR workshop on recommender systems, volume 128. Citeseer, 1999.

[12] Al Mamunur Rashid S K L, Karypis G, Riedl J. ClustKNN: a highly scalable hybrid model-& memory-based CF algorithm. Proceeding of WebKDD, 2006..

[13] Xue G R, Lin C, Yang Q, et al. Scalable collaborative filtering using cluster-based smoothing. Proceedings of Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2005. 114–121.

[14] George T, Merugu S. A scalable collaborative filtering framework based on co-clustering. Proceedings of Data Mining, Fifth IEEE International Conference on. IEEE, 2005. 4–pp.

[15] Banerjee A, Dhillon I, Ghosh J, et al. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. Proceedings of Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2004. 509–514.

[16] Khoshneshin M, Street W N. Incremental collaborative filtering via evolutionary co-clustering. Proceedings of Proceedings of the fourth ACM conference on Recommender systems. ACM, 2010. 325–328.

[17] Goldberg K, Roeder T, Gupta D, et al. Eigentaste: A constant time collaborative filtering algorithm. Information Retrieval, 2001, 4(2):133–151.

[18] Bellogín A, Parapar J. Using graph partitioning techniques for neighbour selection in user-based collaborative filtering. Proceedings of Proceedings of the sixth ACM conference on Recommender systems. ACM, 2012. 213–216.

[19] Xu B, Bu J, Chen C, et al. An exploration of improving collaborative recommender systems via user-item subgroups. Proceedings of Proceedings of the 21st international conference on World Wide Web. ACM, 2012. 21–30.

[20] Sarwar B, Karypis G, Konstan J, et al. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[21] Funk S. Netflix Update: Try This at Home. http://sifter.org/ simon/journal/20061211.html, 2006.

[22] Paterek A. Improving regularized singular value decomposition for collaborative filtering. Proceedings of Proceedings of KDD cup and workshop, volume 2007, 2007. 5–8.

[23] Salakhutdinov R, Mnih A. Probabilistic Matrix Factorization. Proceedings of NIPS, volume 1, 2007. 2–1.

[24] Zhang S, Wang W, Ford J, et al. Learning from Incomplete Ratings Using Non-negative Matrix Factorization. Proceedings of SDM. SIAM, 2006.

[25] Gu Q, Zhou J, Ding C H. Collaborative Filtering: Weighted Nonnegative Matrix Factorization Incorporating User and Item Graphs. Proceedings of SDM. SIAM, 2010. 199–210.

[26] De Meo P, Ferrara E, Fiumara G, et al. Improving recommendation quality by merging collaborative filtering and social relationships. Proceedings of Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on. IEEE, 2011. 587–592.

[27] Ma H, Zhou D, Liu C, et al. Recommender systems with social regularization. Proceedings of Proceedings of the fourth ACM international conference on Web search and data mining. ACM, 2011. 287–296.

[28] Koren Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. Proceedings of Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008. 426–434.

[29] Pazzani M J, Billsus D. Content-based recommendation systems. Proceedings of The adaptive web. Springer, 2007: 325–341.

[30] Barragáns-Martínez A B, Costa-Montenegro E, Burguillo J C, et al. A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. Information Sciences, 2010, 180(22):4290–4311.

[31] Wang J, De Vries A P, Reinders M J. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. Proceedings of Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2006. 501–508.

[32] Bell R M, Koren Y. Improved neighborhood-based collaborative filtering. Proceedings of KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. sn, 2007.

[33] Chen G, Wang F, Zhang C. Collaborative filtering using orthogonal nonnegative matrix tri-factorization. Information Processing & Management, 2009, 45(3):368–379.

[34] Yan S, Xu D, Zhang B, et al. Graph embedding: A general framework for dimensionality reduction. Proceedings of Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 2. IEEE, 2005. 830–837.

[35] Wang H, Yan S, Xu D, et al. Trace ratio vs. ratio trace for dimensionality reduction. Proceedings of Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on. IEEE, 2007. 1–8.

[36] Ngo T T, Bellalij M, Saad Y. The trace ratio optimization problem. SIAM Review, 2012, 54(3):545–569.

[37] Money J H, Ye Q. Algorithm 845: Eigifp: a matlab program for solving large symmetric generalized eigenvalue problems. ACM Transactions on Mathematical Software (TOMS), 2005, 31(2):270–279.

[38] Das A S, Datar M, Garg A, et al. Google news personalization: scalable online collaborative filtering. Proceedings of Proceedings of the 16th international conference on World Wide Web. ACM, 2007. 271–280.

[39] Broder A Z, Charikar M, Frieze A M, et al. Min-wise independent permutations. Proceedings of Proceedings of the thirtieth annual ACM symposium on Theory of computing. ACM, 1998. 327–336.

[40] Broder A Z. On the resemblance and containment of documents. Proceedings of Compression and Complexity of Sequences 1997. Proceedings. IEEE, 1997. 21–29.

[41] Rajaraman A, Ullman J D. Mining of massive datasets. Cambridge University Press, 2012.

[42] Hagen L, Kahng A B. New spectral methods for ratio cut partitioning and clustering. Computer-aided design of integrated circuits and systems, ieee transactions on, 1992, 11(9):1074–1085.

[43] Shi J, Malik J. Normalized cuts and image segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2000, 22(8):888–905.

[44] Dhillon I S. Co-clustering documents and words using bipartite spectral graph partitioning. Proceedings of Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2001. 269–274.

[45] Herlocker J L, Konstan J A, Terveen L G, et al. Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS), 2004, 22(1):5–53.

[46] Sarwar B, Karypis G, Konstan J, et al. Analysis of recommendation algorithms for e-commerce. Proceedings of Proceedings of the 2nd ACM conference on Electronic commerce. ACM, 2000. 158–167.

[47] Balabanović M, Shoham Y. Fab: content-based, collaborative recommendation. Communications of the ACM, 1997, 40(3):66–72.

[48] Cacheda F, Carneiro V, Fernández D, et al. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. ACM Transactions on the Web (TWEB), 2011, 5(1):2.

[49] Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets. Neural computation, 2006, 18(7):1527–1554.

[50] Wu J, Miao Z. Regression-Based Fusion Prediction for Collaborative Filtering. Proceedings of Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on, 2013. 312–319.

致 谢

在中国科学技术大学攻读硕士学位的三年里，我所从事的学习和研究工作，都是在导师以及实验室同学的指导和帮助下进行的。在完成论文之际，请容许我对他们表达诚挚的谢意。

首先感谢导师唐珂教授的指导和教诲，是他把我带到了学术研究这片让我兴趣盎然的沃土。本研究及学位论文是在唐老师的亲切关怀和悉心指导下完成的。唐老师严谨的研究态度，认真细致的治学态度及宽广的胸怀，使我受益终身。在研究生期间，无论是在学习和生活上，唐老师给予了我莫大的关怀和帮助，再次向唐珂老师致以诚挚的谢意!

其次，我要感谢实验室的各位同学：陆晓芬，李丙栋、孙宇，刘金鹏，周俊阳、李泠汐，万伦军、缪志高、庄黎黎、蒋雪、曹倩，钟锦红等。他们的研究和见地拓展了我的学术视野，从他们身上我学到很多知识。感谢一起生活、学习和工作过的同学和朋友，虽然没能一一提及你们的名字，但非常珍视你们的一路同行。感谢有你们的陪伴让我的研究生生活更加丰富多彩，祝愿大家前程似锦!

最后我要感谢我的家人。他们总在背后默默地关心我，支持我，给我前进的动力！正是有他们无私的付出，才有我今天的成长。

吴建军

2014 年 5 月 29 日

在读期间发表的学术论文与取得的研究成果

已发表论文：

1. Jianjun Wu and Zhigao Miao. Regression-based fusion prediction for collaborative filtering. In Proceedings of 2013 International Conference on Cloud Computing and Big Data(CloudCom-Asia), pages 312–319. IEEE, 2013.

待发表论文：

1. Jianjun Wu, Fengjuan Zhang and Yuepeng Wang.Privacy-Preserving Regression Modeling and Attack Analysis in Sensor Network.