北京大学

博士研究生学位论文

题目：**Netflix Prize 中的协同过滤算法**

姓　　名：吴金龙

学　　号：10501829

院　　系：数学科学学院

专　　业：计算数学

研究方向：推荐系统中的协同过滤算法

导师姓名：鄂维南 教授，李铁军 副教授

二〇一〇 年 七 月

版权声明

Netflix Prize 中的协同过滤算法

吴金龙　计算数学
导师姓名：　鄂维南 教授, 李铁军 副教授

摘　要

伴随着存储技术的迅速发展，电子商务和零售企业正在积累着越来越多的用户交易数据。而产品或服务（统称为产品）的渐趋多样性使得用户在购买产品时往往需要花费大量的时间筛选产品。推荐系统被发展用来减轻用户的筛选负担，它为用户提供个性化的产品推荐。精确的推荐系统可以帮助用户更容易地找到他们所需的产品，也可以通过改进用户体验帮助企业提升用户忠诚度进而把更多的产品浏览者转换为产品购买者。

一般地推荐系统首先通过分析用户过去的购买行为建立合适的产品排序模型，然后使用获得的模型为用户产生个性化推荐。例如，一个在线商店可以根据用户对产品的评分，以及他们购买的、加入购物篮的和浏览的产品获得用户对产品的偏好，进而为用户推荐他们感兴趣的产品。

鉴于产生推荐的方式不同，推荐系统通常可以分为以下三类：（1）基于内容的过滤（content-based filtering）：基于内容的过滤（CBF）方法根据抽取出的用户和产品特征获得推荐。这类方法利用用户和产品的特征计算他们之间的匹配度，最终把匹配得最好的数个产品推荐给相应的用户。（2）协同过滤（collaborative filtering）：协同过滤（CF）方法首先分析已经收集到的用户-产品评分对中所呈现的用户与产品的相互作用，然后它们利用这些相互作用产生对用户的推荐。（3）CBF 与 CF 的混合过滤（hybrid filtering）：混合过滤方法组合 CBF 和 CF 方法以期在克服它们各自缺点的同时融合它们特有的优势。

本论文中我们研究及发展了应用于 Netlfix Prize 竞赛的各种 CF 算法。 Netflix Prize 竞赛由在线 DVD 租赁公司 Netflix 于 2006 年 10 月创建，旨在推进学术界和工业界对 CF 算法的研究。它所发布的数据集为目前最大的免费公开的 CF 数据集。在这几年的时间内众多参赛者提出了各种高效的 CF 算法，其中包括邻居模型（kNN）、矩阵分解（MF）、受限玻尔兹曼机（RBM）以及聚类模型等等。我们在本论文中提出了一些新的推荐模型，并详细介绍了 Netflix Prize 中各种具有代表性的模型。本论文的创新性主要体现在以下几个方面：

- 我们组合了因子模型 *matrix factorization (MF)* 和模糊聚类模型 *fuzzy c-means (FCM)* 的想法，提出了一种新的聚类模型——*modified fuzzy c-means*（*MFCM*）。相比于 MF，FCM 的解释更容易让人理解和接受。但对于 Netflix Prize 问题，MF 可以获得比 FCM 更精确得多的预测结果。新模型 MFCM 试图整合 FCM 更好的可解释性以及 MF 更加精确的结果预测性。我们也构造了两个新的算法来求解 MFCM，它们在 Netflix Prize 数据集上的实验表明 MFCM 获得了比 FCM 更加精确的推荐结果，其预测精度与 MF 相仿，且其算法结果比 MF 的结果更易解释。

- 对于离散 CF 问题，我们使用二项分布代替 MF 模型中评分假设的正态分布，从而获得了一种新的矩阵分解模型——*binomial matrix factorization*（*BMF*）。离散 CF 问题中评分只允许有数个离散值，所以此时 BMF 中的二项分布假设比 MF 中的正态分布假设更加合理。我们同时构造了两个新的算法来求解 BMF，这两个算法应用于 Netflix Prize 数据集时获得了比 MF 更好的预测精度。我们也把评分满足二项分布的这种思想扩展到了聚类模型，新产生的聚类模型在 Netflix Prize 问题上获得了很大的预测精度提升。

- 我们把 CF 问题发展到三维情形，并称之为立方填补（*cube completion*）问题。实际上个性化网页搜索和个性化广告投放就是典型的立方填补问题。关于立方填补的研究目前还很少，就我们所知，我们是首次把它们作为一类问题抽象出来，并对它们进行系统的研究。我们构造了一些求解立方填补问题的可行算法，如贝叶斯聚类、立方聚类和立方分解等等。我们在构造的虚拟

数据上对这些算法进行了一些检验，计算结果表明这些算法对于不太稀疏的数据集都可以获得很好的预测效果，但当数据稀疏程度加剧时，不同的模型之间将呈现出较大的预测效果差异。

关键词： 推荐系统, 协同过滤, 邻居模型, 矩阵分解, 受限玻尔兹曼机, 立方填补, Netflix Prize

Collaborative Filtering On the Netflix Prize Dataset

Jinlong Wu (Computational Mathematics)

Supervised by **Weinan E and Tiejun Li**

Abstract

With the rapid development of data storage technology, and further product or service (called *item* uniformly) diversification, many types of enterprises, such as e-commerce and retail enterprises, have been accumulating more and more transaction data from their customers who have been encountering too many different items. To help customers find out what they need more easily, and help enterprises improve loyalty of their customers by enhancing the user experience, then further convert more browsers into consumers, recommendation systems have been developed to offer personalized suggestions for customers.

Usually recommendation systems first analyze histories of customers' usage behaviors and achieve appropriate models for ranking, then use the resulting models to produce personalized recommendations for the customers. For example, an online store can learn customer preferences according to ratings they provided for products, and what products they bought, added to the cart, and browsed.

In the light of different ways to produce recommendations, recommendation systems are usually classified into three types: (1) content-based filtering (CBF): CBF methods provide recommendations based on features of users and items. Recommendations to one user are those items whose features best match those of the target user. (2) collaborative filtering (CF): CF methods first analyze interactions of

user-item pairs on the basis of known ratings, then uses the interactions to produce recommendations. (3) hybrid of CBF and CF: Hybrid filtering methods combine CBF and CF methods in order to overcome the drawbacks of both CBF and CF and benefit from their advantages.

In this thesis we focus on the development of CF algorithms for the *Netflix Prize* competition, which was founded by *Netflix* at October, 2006. So far many competitors have proposed variety of effective CF algorithms such as neighborhood models (kNN), matrix factorization (MF), restricted Boltzmann machines (RBM), clustering models and so forth. Typical individuals of them are introduced detailedly in the thesis. Our innovative contributions in the thesis mainly comprise of the following aspects:

- In comparion with factor models (e.g. MF), clustering models (e.g. fuzzy c-means, FCM for short) have better interpretability but usually produce worse predictive results for the Netflix Prize problem. Combining the ideas of MF and fuzzy c-means (FCM) models, we propose a new clustering model — *modified fuzzy c-means (MFCM)*. MFCM has better interpretability than MF, and better accuracy than FCM. MFCM also supplies a new perspective on MF models. Two new algorithms are also developed to solve this new model. They are applied to the Netflix Prize data set and acquire comparable accuracy with that of MF.

- MF models assume that the factor vectors of users and items are from normal distributions, and so are the ratings when the user and item factors are given. With more reasonable rating assumption that all ratings are from binomial distributions with different preference parameters for discrete CF problems, we suggest a new probabilistic perspective on MF for discrete CF problems. The new model is called *binomial matrix factorization (BMF)*, and two effective algorithms to learn BMF and finally make predictions are developed. They are applied to the Netflix Prize data set and acquire considerably better accuracy

than those of MF. The similar idea of binomial distributions can be used to obtain new clustering models.

- We generalize 2-dimensional CF problems to 3-dimensional cases, since mathematically 2-dimensional CF problems are matrix completion problems. 3-dimensional CF is called *cube completion (CC)*, and it is useful for personalized web search and personalized advertising. Several feasible algorithms are proposed and tested in our synthetic data sets.

Keywords: Recommendation System, Collaborative Filtering, Neighborhood Models, Matrix Factorization, Restricted Boltzmann Machines, Cube Completion, Netflix Prize

目　录

摘　要 i

Abstract v

目　录 ix

表　格 xiii

插　图 xv

第一章　绪　论 1

1.1　推荐系统介绍 1

1.2　本文结构 4

1.3　Netflix Prize 介绍 5

1.4　术语和记号 10

第二章　评分预处理方法 13

2.1　全局作用（Global Effects） 13

2.2　基准预测模型（Baseline Predictors） 17

第三章　邻居模型 25

3.1　传统的 kNN 模型 26

3.2　邻居关系（Neighborhood Relationships）模型 28

3.3　全局邻居（Global Neighborhood）模型 31

第四章 受限玻尔兹曼机 **35**

4.1 受限玻尔兹曼机（RBM） 35

4.2 条件受限玻尔兹曼机（Conditional RBM） 41

4.3 时间受限玻尔兹曼机（Time RBM） 44

4.4 因子化受限玻尔兹曼机（Factored RBM） 45

第五章 因子模型 **47**

5.1 矩阵分解（Matrix Factorization）模型 48

5.1.1 Basic MF 48

5.1.2 Biased MF 49

5.1.3 Probabilistic MF (PMF) 50

5.2 二项矩阵分解（Binomial MF）模型 55

5.2.1 BMF 56

5.2.2 Probabilistic BMF (PBMF) 57

5.3 修正 Fuzzy C-means（Modified FCM）模型 60

5.3.1 Fuzzy C-means 聚类模型 61

5.3.2 修正 Fuzzy C-means（Modified FCM）模型 63

5.4 其他的因子模型 66

5.5 实验结果 70

5.5.1 MF 模型与 BMF 模型 70

5.5.1.1 算法 MF 与算法 BMF 70

5.5.1.2 算法 PMF 与算法 PBMF 71

5.5.2 MF 模型与 MFCM 模型 73

第六章 模型组合方法 **77**

6.1 线性回归（Linear Regression）模型 78

6.2 神经网络（Neural Network）模型 79

6.3 后处理（Postprocessing）方法 80
6.4 实验结果 82

第七章 三维协同过滤：立方填补 87
7.1 贝叶斯聚类（Bayesian Clustering）模型 88
7.2 立方聚类（Cube Clustering）模型 93
7.3 立方分解（Cube Factorization）模型 95
7.4 邻居（Neighborhood）模型 97
7.5 实验结果 98
7.5.1 BC 模型和 CC 模型 99
7.5.2 CF 模型 101

第八章 总结与展望 105

附录 A Global Effects 中压缩的 Bayes 解释 109

附录 B 二项混合（Mixture of Binomials）模型 111

附录 C 超参数（hyper-parameter）学习 115
C.1 Nelder-Mead 算法 115
C.2 Automatic Parameter Tuners (APTs) 118

参考文献 121

致 谢 131

发表/待发表论文目录 135

个人简历 137

表　格

1.1 Netflix Prize 所颁发的各种奖项及对应的获奖团队 10

1.2 本文中使用的各种记号与术语 12

2.1 全局作用（GEs）以及由它们所产生的 Probe RMSE （表中数据来源于 [6, 88–90]） 15

5.1 学习率 η 取不同值时算法 MF 和 BMF 在 Netflix Prize 问题上所获得的 Probe RMSE。这些结果对应的因子数 $K = 40$，且算法 MF 和 BMF 所使用的惩罚系数 λ 分别为 0.025 和 0.0015。 71

5.2 算法 MF 与算法 MFCM1、MFCM2 在 Netflix Prize 问题上所获得的 Probe RMSE。所有算法所取的因子数 $K = 40$，而算法 MF 和 MFCM1 所使用的惩罚系数 $\lambda = 0.025$，算法 MFCM2 所使用的 $\lambda = 0.0002$，且其对应的动量 $\mu = 0.85$ [98]。 73

5.3 学习率 η 取不同值时算法 MFCM1 与 MFCM2 在 Netflix Prize 问题上所获得的 Probe RMSE。这些结果对应的因子数 $K = 40$，而算法 MFCM1 和 MFCM2 所使用的惩罚系数 λ 分别为 0.025 与 0.0002，且算法 MFCM2 所使用的动量 $\mu = 0.85$ [98]。对于算法 MFCM1，我们使用(5.83)逐渐降低学习率 η，其中参数 $\eta^{(0)} = 0.004$ 以及 $\epsilon_0 = 0.02$。对于算法 MFCM2，我们使用(5.84)逐渐降低学习率 η，其中参数 $\eta^{(0)} = 0.006$ 以及 $N = 80$ [98]。 . . . 74

7.1 生成训练和测试数据集时所使用的参数 99

插　图

1.1 Netflix Prize 数据集的生成过程与结构 7

1.2 左图为单个用户的评分数量直方图，从图中可见大部分用户只对很少的电影进行了评分；右图为单部电影被评分的数量直方图，从图中可见大部分电影只收到极少的用户评分。这些图都来自于 [9]。 9

1.3 左图显示了全局平均评分随着时间的变化情况；右上图显示了数据集中每天收到的评分数量随着时间的变化情况；右下图显示了每个用户每天的评分数量（frequency）的 log-直方图。左图来自于 [9]，右图来自于 [89]。 10

3.1 [6] 中给出的求解具有非负约束的二次最小化问题的优化方法 31

4.1 RBM 的图模型表示 36

4.2 CRBM 的图模型表示 42

5.1 PMF 的图模型表示 51

5.2 因子数 K 取不同值时算法 PMF 和 PBMF 在 Netflix Prize 数据集上所获得的 Probe RMSE 随着算法迭代的变化图。伴随着 K 的增大，PBMF 较之 PMF 呈现出越来越明显的优势。当 $K = 60$ 时，PBMF 在经历了 25 次迭代后获得了 0.9029 的 Probe RMSE，而 BMF 在 59 次迭代后获得了 0.9050 的预测误差[96]。 72

6.1 具有单个输出层单元的神经网络图形表示 80

6.2 最终组合中所使用的 93 个模型名称。组合结果在 Probe Set 上所获得的预测误差为 0.8717，而提交到 Netflix Prize 后在 Quiz Set 上所获得的预测误差为 0.8747。 84

6.3　（续图 6.2）最终组合中所使用的 93 个模型名称。组合结果在 Probe Set 上所获得的预测误差为 0.8717，而提交到 Netflix Prize 后在 Quiz Set 上所获得的预测误差为 0.8747。 85

7.1　贝叶斯聚类模型的图形表示 89

7.2　HOSVD 在三维时的图形表示 96

7.3　当稀疏度 $\delta_1 = 2 \times 10^{-4}$ 时某次抽取数据后训练数据集内 Z 方向两两对象之间共有评分数的直方图。 98

7.4　相对于不同的数据稀疏度 δ_1 （横坐标），模型 BC （左）和 CC （右）在测试集上所获得的预测误差（纵坐标）图。图中的黑点表示使用生成数据的模型参数值进行预测（即“Optimal”预测）所获得的预测误差。而模型名称后面的数字分别代表各个方向上聚类的类别数 L、 M 和 N，如“$10+10+5$”表示 $L=10$、 $M=10$ 和 $N=5$。 100

7.5　当训练数据集的稀疏度 $\delta_1 = 2 \times 10^{-4}$，且各个方向上聚类的实际类别数 $L=10$、 $M=10$ 和 $N=5$ 时，模型 BC （左）和 CC （右）选取不同的聚类类别数时在测试集上所获得的预测误差（纵坐标）图。图中的横坐标表示给定参数时训练数据集抽取的不同次数。图中的黑点表示使用生成数据的模型参数值进行预测（即“Optimal”预测）所获得的预测误差。标示内的数字分别代表各个方向上训练模型时所使用的聚类类别数，如“$10+10+5$”表示 $L=10$、 $M=10$ 和 $N=5$。 101

7.6　相对于不同的数据稀疏度 δ_1 （横坐标），模型 CF 在测试集上所获得的预测误差（纵坐标）图。图中的黑点表示使用生成数据的参数值进行预测（即“Optimal”预测）所获得的预测误差。而模型名称后面的数字分别代表各个方向上聚类的类别数 L、 M 和 N，如“$10+10+5$”表示 $L=10$、 $M=10$ 和 $N=5$。 102

B.1　二项混合模型的图形表示 112

C.1 $\mathbb{R}^2$ 和 $\mathbb{R}^3$ 中的单纯形（原图可见 [79]） 116

C.2 $\mathbb{R}^2$ 中各种变形的示例图（原图可见 [79]）。图中蓝色表示变形前的 $\mathcal{S}$，红色表示变形后的 $\mathcal{S}$。 117

第一章　绪　论

1.1　推荐系统介绍

从信息检索的方式来看互联网的发展大致可分为如下三个阶段：（1）门户网站阶段：这个时候互联网刚刚兴起，网络上出现的网页数量还较少且质量参差不齐。这种环境催生了引领互联网走入免费时代的门户网站 Yahoo[99]。Yahoo 当时的主要功能是为网络用户提供导航，让用户知道当时网络上最重要的信息来源于何处。（2）搜索引擎阶段：伴随着互联网飞速的发展，导航网站已经无法帮助用户定位网络上大量涌现的主要信息。这种现象促使搜索引擎行业的不断壮大，其中最具有代表性的搜索引擎莫过于 Google[31]。搜索引擎依据用户输入的关键词，返回给用户与关键词相关的网页。但与前一阶段导航网站不同的是，搜索引擎并不只是简单地返回相关的网页，它们还根据网页与关键词的相关性以及网页本身的重要性对网页进行排序，使得用户需要的网页尽量展示在结果的前面。但搜索引擎的缺点是对于任何用户，只要用户输入的是相同的关键词，最终获得的网页排序也将是相同的。也就是说，搜索引擎并不考虑搜索用户之间的需求差异。（3）推荐系统阶段：互联网资源的爆炸式增长使得搜索引擎的无用户差异搜索缺点越来越明显。网络上存在的与单个或少数几个关键词相关的重要性网页越来越多，而搜索用户在查看搜索结果时一般只查看排在最前面的几十个相关结果。用户需求的多样性使得他们在使用搜索引擎时不可能都获得很理想的排序结果。解决此缺点的一个方法是根据搜索用户的特点为不同的用户返回不同的排序结果，这也就是所谓的个性化网页搜索[81, 82]。这种观念其实已经在现在的搜索引擎中有所体现，如 Google 允许用户定制自己的网页重要性，百度[4]提出所谓的智能框搜索。

伴随着互联网的发展，另一个网络新兴产业是电子商务。电子商务把商品从现实的实体店搬到了网络上，使得用户不出家门便可购遍天下。而且，随着存储技术的不断发展，电子商店目前已经可以在很小代价下容纳几乎任意多的产品，这种优势是实体店无法比拟的。对于网络购物用户来说，最大的问题往往不是

可选择的产品太少，而是可选择的产品太多，让用户挑起来无从下手。解决这个问题的一种途径就是使用推荐系统为用户产生个性化的产品推荐，也即通过分析用户对产品的点击、浏览和购买等数据获得用户潜在感兴趣的产品，并把它们推荐给用户。精确的个性化推荐一方面可以缩短用户挑选产品的时间，提高用户的购买效率，另一方面也可以改进网站的用户体验，提升产品的交易率进而提高网站的利润。把推荐系统应用于商业领域最成功的公司应该是具有“推荐之王”美誉的在线购物企业 Amazon[3]。据报道[53, 68]， 2002 年 Amazon 产品总销售的 20% 源自它的推荐系统。目前推荐系统已经广泛应用于互联网的各个领域，如 Pandora[67]的音乐推荐、Netflix[60]的电影推荐、Google 的新闻推荐[32]以及 Facebook[30]的朋友推荐等等。

正如我们之前所说的，推荐系统首先通过分析用户的使用历史获得用户的兴趣偏向，然后使用得到的用户兴趣偏向获得用户潜在感兴趣的产品或服务（英文中产品或服务被统称为 *item*，本文中我们统称之为产品）。鉴于产生推荐的方式不同，推荐系统通常可以分为以下三类[1, 25]：

基于内容的过滤（content-based filtering） 基于内容的过滤（CBF）方法根据抽取出的用户和产品特征获得推荐。一般地一种产品的特征（如产品价格和种类）事先人为给定，而一个用户的特征可以根据他（她）消费的产品的特征而获得。然后 CBF 利用用户和产品的特征计算他们之间的匹配度（相似度），并最终把匹配得最好的数个产品推荐给相应的用户。CBF 方法具有两个主要的缺陷：（1）CBF 需要预处理产品以得到代表它们的特征，但这种预处理在实际问题中往往非常困难。（2）CBF 推荐给某个用户的产品往往和此用户已经消费过的产品很相似，它们无法发现用户并不熟悉但具有潜在兴趣的产品种类。

协同过滤（collaborative filtering） 协同过滤（CF）方法不需要事先获得产品或用户的特征，它们只依赖于用户过去的行为（如对产品的浏览或购买等）。通过用户过去的行为企业可以收集用户对产品的显式评分（如 Netflix）或隐式评分（如 Google 新闻）。通常 CF 方法首先分析已经收集到的用户-产品评分对中所呈现的用户与产品的相互作用，然后它们使用这些相互作用

为用户产生个性化产品推荐。基于用户的（*user-based*）CF 方法假设如果两个用户过去对产品有相似的喜好，那么他们现在对产品仍有相似的喜好。基于产品的（*item-based*）CF 方法假设如果某个用户过去喜欢某种产品，那么他（她）现在仍喜欢与此产品相似的产品。因为 CF 方法不需要预处理产品或用户的特征，所以它们不依赖于应用中的特有领域，并且能克服 CBF 方法内在的推荐缺陷。但 CF 方法也引入了新的困难：（1）冷启动：对于一个新用户，由于缺乏他（她）对产品的评分，CF 无法为他（她）提供可靠的产品推荐。对于一种新的产品，类似的困难同样存在，CF 无法确定该把它推荐给哪些用户。（2）可扩展性：CF 方法中可能涉及到数以百万计的用户为成千上万种产品提供的评分。传统的 CF 推荐算法（如 kNN）通常需要计算每对用户或产品之间的相似度，然后把这些相似度存放至电脑的主存中以便高效地产生推荐。当用户或产品的数量较大时，这类算法会被电脑主存大小所限制。

CBF 与 CF 的混合过滤（hybrid filtering） 混合过滤方法组合 CBF 和 CF 方法以期在克服它们各自缺点的同时融合它们特有的优势。通常组合的方式包括以下三种[1]：（1）加权（*weighted*）组合：首先分别独立应用 CBF 和 CF 方法获得对产品的预测评分，然后组合它们的预测评分以便获得混合过滤的预测评分，最后根据混合预测评分为用户产生推荐列表。（2）混合（*mixed*）组合：首先分别独立应用 CBF 和 CF 方法产生各自的推荐列表，然后组合这两组推荐列表以便获得最终的推荐列表。（3）序贯（*sequential*）组合：当可用评分较少时使用 CBF 方法获得用户的特征并进行推荐；而当可用评分积聚到一定程度时，使用 CF 方法代替原来的 CBF 方法获得最终的推荐列表。

本文中我们将主要研究及发展 CF 中的各类算法，一方面是因为 CBF 所使用的数据不易获得且难以处理，另一方面是因为 Netflix 在 2006 年发布了用于发展 CF 算法的 Netflix Prize[61]竞赛，并且公开了目前 CF 领域内最大的真实数据集（其详细介绍可见第 1.3 节）。

Breese 等 [16] 把 CF 算法分为两大类。第一类是所谓的基于内存的（*memory-*

based）的算法，这类算法利用用户或产品的邻居信息获得推荐。它们中的典型代表是 kNN 模型[35, 39, 53, 56, 71, 75, 78, 100]。第二类算法被称为基于模型的（*model-based*）算法，这类算法首先利用已知评分训练一个预测模型，然后利用预测模型获得预测评分进而产生最终的推荐列表。研究者已经发展了很多这类算法，如分类聚类模型[26, 45, 68, 91, 92]、贝叶斯图模型[15, 22, 24, 29, 55, 58, 59, 76, 77] 和因子模型[10, 49, 70, 72, 73, 96]等等。

1.2 本文结构

因为本文中所有的实验都是基于 Netflix Prize 竞赛中的 CF 数据集完成的，所以下一节我们将详细介绍与 Netflix Prize 相关的一些信息。本章的最后一节中我们介绍了本文中所使用的通用术语和记号。

在第二章中我们介绍了一些流行的 Netflix Prize 数据集预处理方法，这些方法在本文之后介绍的很多 CF 算法中都得以应用。接下来的第三章到第五章我们分别介绍了应用于 Netflix Prize 问题的三大类算法：邻居模型、受限玻尔兹曼机以及因子模型。这些模型中包括 CF 领域的经典推荐算法，也包括众多研究者最近提出的新式推荐算法。

对于离散 CF 问题，我们在第 5.2 节中使用二项分布代替传统因子模型（MF）评分假设中的正态分布，从而获得了一种更适合于离散 CF 问题的矩阵分解模型——*binomial matrix factorization*（*BMF*）。我们也构造了两个新的算法来求解 BMF，并在第 5.5 节中详细比较了它们与 MF 模型在 Netflix Prize 数据集上所获得的预测结果，这两个算法应用于 Netflix Prize 数据集时都获得了比 MF 更好的预测精度。

相比于因子模型（如 MF），聚类模型（如 fuzzy c-means，简记为 FCM）的解释通常更容易让人理解和接受。但对于 Netflix Prize 问题，因子模型往往可以获得比聚类模型更精确得多的预测结果。第 5.3.2 节中我们组合因子模型 MF 和模糊聚类模型 FCM 的想法，提出了一种新的聚类模型——*modified fuzzy c-means*（*MFCM*）。MFCM 较之 MF 有更好的可解释性，而较之 FCM 能产生更加精确的推荐结果。我们同时构造了两个新的算法来求解 MFCM，并在第 5.5 节中详细

比较了它们与 MF 模型在 Netflix Prize 数据集上所获得的预测结果。

根据参加 Netflix Prize 竞赛团队的提交经验，多个模型的有效组合往往可以很大程度地改进单个模型的预测精度。所以在第六章中我们介绍了一些能够有效应用于 Netflix Prize 数据集的组合多个预测模型的方法。第 6.4 节中我们使用这些组合方法组合第三章到第五章中介绍的各种模型，以便同时计入各种模型考虑的推荐因素。最终的组合结果（包括了 93 个模型）在 Probe Set 上获得了 0.8717 的预测误差，而我们提交给 Netflix Prize 的 Quiz Set 上的预测评分获得了 0.8747 的预测精度。

CF 问题从数学上来讲是矩阵填充问题。一个自然的推广是把二维的矩阵填充扩展到三维的立方填补上。目前关于立方填补的研究还很少（[81, 82] 中研究了个性化网页搜索问题），就我们所知，我们是首次把它们作为一类问题抽象出来，并对它们进行系统的研究。第七章中我们首先介绍了立方填补在网页搜索和网站广告投放上的应用，然后构造了一些求解立方填补问题的可行算法，如贝叶斯聚类、立方聚类和立方分解等等。因为缺乏可用的公开数据，我们无法在真实数据上检验这些模型的有效性，但我们利用生成的模拟数据对各个模型相对于训练数据集稀疏度和模型参数的敏感性进行了详细的比较。计算结果表明这些算法对于不太稀疏的数据集都可以获得很好的预测效果，但当数据稀疏程度加剧时，不同的模型之间将呈现出较大的预测效果差异。例如，相对于立方聚类，贝叶斯聚类对数据稀疏度呈现出更好的适应性，但它的代价是需要更多的训练时间；立方分解的预测误差随着数据稀疏程度的加剧缓慢增加，而贝叶斯聚类和立方聚类的预测误差则随着数据稀疏程度的加剧陡然增加。

在第八章中我们总结了全文的内容，并提出了一些 CF 领域中重要但尚未受到足够重视的问题。对这些问题的研究将是我们未来工作的主要方向。

1.3 Netflix Prize 介绍

Netflix Prize 背景简介

Netflix[60] 是美国一家提供在线电影租赁服务的公司。用户只要按月缴纳一定的租赁费用就可以向 Netflix 订阅大量的电影。在收到客户的订阅需求之后，

Netflix 会通过电子邮件（电子版）或投递（DVD碟片）的方式为客户提供其所需的电影。用户可以为 Netflix 上的电影提供评分（包括 $1, 2, \ldots, 5$），分数越高表示客户对相应电影的评价越高。从 1998 年 10 月到 2007 年 8 月，Netflix 已经收集了来自 $11,700,000$ 多个用户对 $85,000$ 多部电影的 $1,900,000,000$ 多个评分[14],并且评分数量仍在以每天 $2,000,000$ 的速度增长。Netflix 使用自己的推荐系统 Cinematch 分析这些积累的电影评分数据，学习用户的兴趣偏好，以便给他们推荐一些“潜在有趣”的电影。

2006 年 10 月，Netflix 对外发布了一个电影评分数据集，并建立了 Netflix Prize[61]竞赛。 Netflix Prize 的最终目标是发展高效的推荐系统，其推荐效果需在 Cinematch 的基础上改进10%（使用均方根预测误差来衡量）。任何人可以独立参赛或与他人组队参赛，但参赛的不同团队不允许人员完全一致。团队可以在线提交预测结果，在线系统会自动计算出预测误差返回给提交团队，并在必要时更新在线排名表[64]（根据预测误差大小进行的团队排名）。第一个达到要求推荐精度的参赛团队将被授予 Grand Prize，奖金为一百万美元。除了最终的 Grand Prize，Netflix Prize 每年还为获得当年最佳预测精度的团队（当年排名第一的团队）颁发 Progress Prize，奖金为五万美元。按照竞赛规则，任何奖项的获奖团队都必须公开发布其获得预测评分所使用的详细算法。这样其他团队就可以从中吸取经验，以便推动竞赛朝着最终目标前进。更详细的参赛规则请见 [65]。

Netflix Prize 数据集

数据集的产生与结构

正如我们前面介绍的，Netflix 所拥有的*全局评分数据库*（*Netflix's Global Rating Data Base*） 包括了从 1998 年 10 月到 2007 年 8 月的超过 $1,900,000,000$ 个评分，这些评分来自于 $11,700,000$ 多个用户对 $85,000$ 多部电影的评分（见图1.1的第一行）。 Netflix Prize 所使用的数据集 *Complete Netflix Prize Dataset (CNPD)*（见图1.1的第二行）是通过随机抽取全局评分数据库中的部分用户的评分数据而产生的。候选的用户必须在 1998 年 10 月至 2005 年 12 月期间至少对 20 部电影提供了评分（CNPD 中的所有数据都是在 1998 年 10 月到 2005 年 12 月期

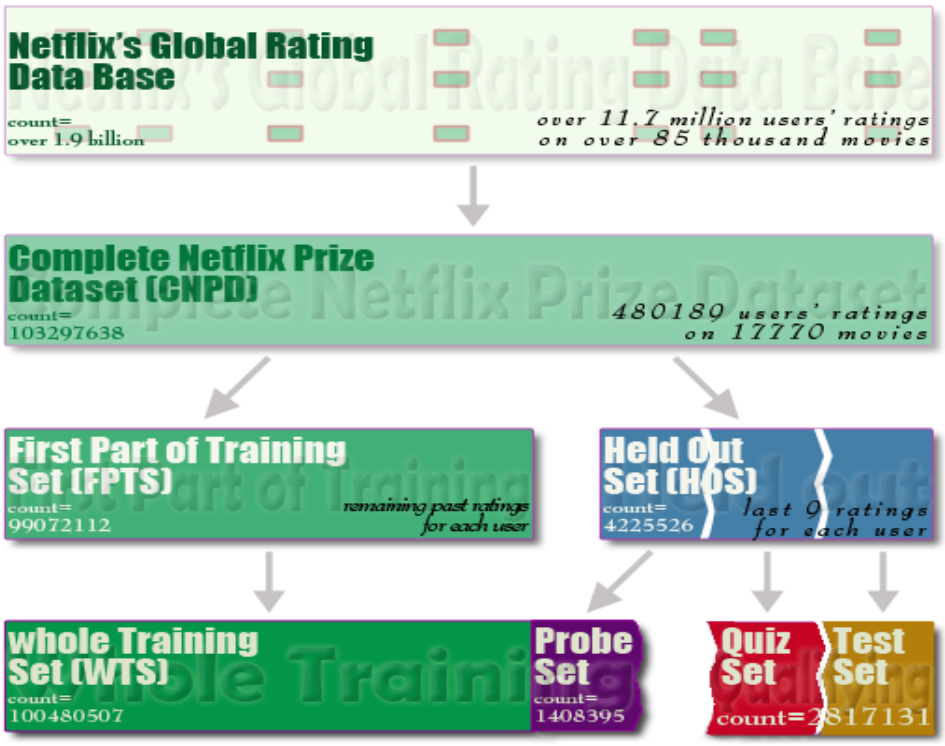

图 1.1: Netflix Prize 数据集的生成过程与结构

间收集到的)。为了保护客户的隐私，Netflix 对 CNPD 中的所有评分值做了不影响整体数据集统计性质的扰动[14]。最终形成的 CNPD 包括了 480,189 个用户在 1998 年 10 月到 2005 年 12 月期间对 17,770 部电影的 103,297,638 个评分。所有的评分值都是 1 到 5 中的整数值，其中分数越高表示客户对相应电影的评价越高(越喜欢)。

为了得到各种算法的预测效果比较，CNPD 被进一步分为两个数据集。CNPD 中每个用户的 9 个最近评分（由于扰动的原因，有些用户的评分总数可能不足 18，这时只选其中最近的一半评分）被独立挑选出来，形成了包括 4,225,526 个评分值的*Held Out Set (HOS)*[9]（见图1.1中第三行右边的蓝色部分)。CNPD 中的剩余部分（我们称之为*First Part of Training Set (FPTS)*）形成另一部分（见图1.1中第三行左边的绿色部分)。HOS 中的所有评分值被等概率地放入三个数据集——*Probe Set*，*Quiz Set* 和 *Test Set* 中的其中一个。最终 Probe Set

和 FPTS 一起形成了大小为 100,480,507 的整体训练数据集 *Whole Training Set (WTS)*（见图1.1中第四行的左部），而 Quiz Set 和 Test Set 一起形成了大小为 2,817,131 的 *Qualifying Set*（见图1.1中第四行的右部）。

参赛团队在线提交预测评分时必须提交整个 Qualifying Set 上的预测值，Netflix Prize 在线系统会自动计算出 Quiz Set 上的均方根预测误差（root mean squared error，简记为 RMSE），返回给提交团队，并在必要时更新在线排名表[64]（根据预测误差大小进行的团队排名）。Test Set 上的预测误差用于最终奖项的评测，故不返回给提交团队。

Netflix Prize 给出了训练数据集中的所有评分值及评分时间。由于 Qualifying Set 中的评分值用于竞赛评比，故不对外公开，但 Netflix Prize 公开了相应的评分时间。除了以上信息， Netflix Prize 也提供了 CNPD 中所有电影的名称及上映时间。

上面所说的用户直接给予产品的评分通常被称为显式信息（*explicit information*）或显式评分（*explicit ratings*），但在很多实际应用中往往还包括了一些所谓的隐式信息（*implicit information*）[46, 47]。例如电子商务公司（如Amazon，淘宝等等）还可以收集到用户的购买、收藏、浏览和点击信息等等。这些隐式信息往往可以被用来进一步改进推荐效果[40, 66]。 Netflix Prize 并没有给出特别的隐式信息，但给出了一种隐晦的隐式信息：用户对什么电影进行了评分。这种隐式信息的使用可以改进很多算法在 Netflix Prize 问题上的预测精度[46, 47, 73, 74]。

Netflix Prize 也公布了 Cinematch 在 Quiz Set 和 Test Set 上的预测误差，分别为 **0.9514** 和 **0.9525**，供参赛团队参考。最终的 Grand Prize 将被授予第一个在 Test Set 上预测精度相对于 Cinematch 改进了 10% 的参赛团队，也即预测误差达到 $0.9525\times 90\% = \mathbf{0.8572}$ 的团队[63]。

数据集的特点

通过分析 Netflix Prize 给出的训练数据集 WTS，我们不难发现一些显著的数据特征：

稀疏性 WTS 中包括了 480,189 个用户对 17,770 部电影的评分，而评分值只有

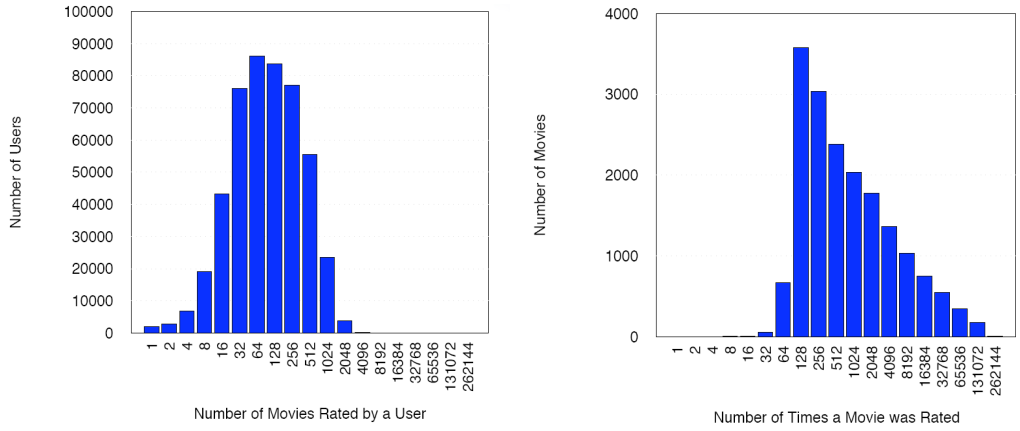

图 1.2: 左图为单个用户的评分数量直方图，从图中可见大部分用户只对很少的电影进行了评分；右图为单部电影被评分的数量直方图，从图中可见大部分电影只收到极少的用户评分。这些图都来自于 [9]。

100,480,507 个，也即近 99% 的评分值未知。

长尾性 大部分用户只对极少的电影进行了评分，如四分之一的用户只对少于 36 部电影进行了评分；大部分电影只收到极少的用户评分，如四分之一的电影只收到少于 190 个用户的评分[9]。图 1.2 给出了详细的分布图。从图 1.3 中右下部分我们也可以看出每个用户每天的评分数量（也即*frequency*）同样呈现出明显的长尾性，大部分用户每天只对极少的电影进行了评分[89]。更详细的讨论可见 [9, 89]。

时间性 图 1.3 显示了数据集中的评分随着时间变化的特征。图 1.3 中的左半部分表明全局平均评分在 2004 年初有明显的提升[9]，而从图 1.3 中的右上部分我们可以看出数据集中每天收到的评分数量随着时间在稳步增加[89]。更详细的讨论可见 [9, 89]。

一些更详细的数据特征分析可见 [23, 62]。

Netflix Prize 竞赛的里程碑

2009 年 6 月 26 日参赛团队 BellKor's Pragmatic Chaos (BPC)[13] 向 Netflix Prize 提交了他们的最新预测结果。此提交在 Quiz Set 上获得了 **0.8558** 的预测误差，

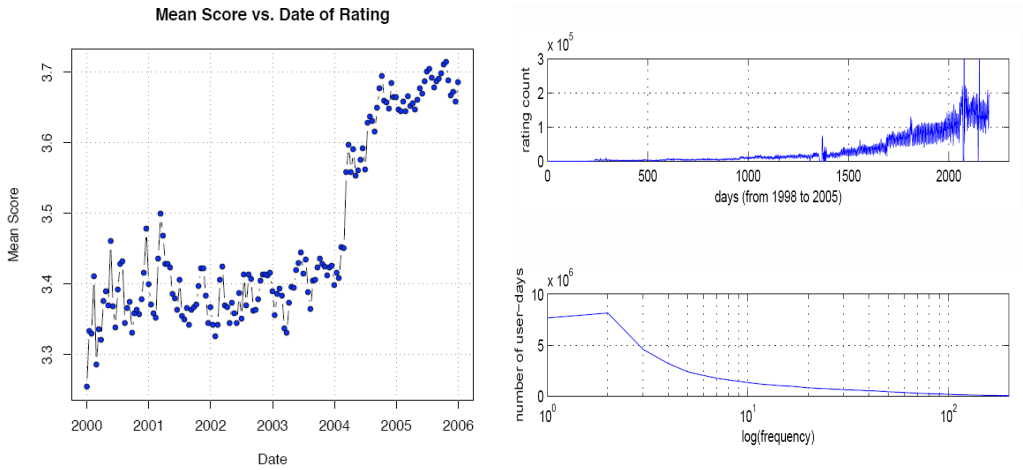

图 1.3: 左图显示了全局平均评分随着时间的变化情况；右上图显示了数据集中每天收到的评分数量随着时间的变化情况；右下图显示了每个用户每天的评分数量（frequency）的log-直方图。左图来自于 [9]，右图来自于 [89]。

相对于 Cinematch 首次获得了超过10%（10.05%）的改进， Netflix Prize 宣布进入竞赛的最后三十天角逐。

奖项	获奖团队	Test RMSE
Progress Prize 2007	KorBell	0.8723
Progress Prize 2008	BellKor in BigChaos	0.8627
Grand Prize	BellKor's Pragmatic Chaos	0.8567

表 1.1: Netflix Prize 所颁发的各种奖项及对应的获奖团队

2009 年 9 月 10 日， Netflix Prize 官方正式宣布 BPC 为竞赛的最终胜利者，获颁 Grand Prize ，整个竞赛正式结束。BPC 最终在 Quiz Set 和 Test Set 的预测误差分别为 **0.8554** 和 **0.8567** 。表 1.1 列出了 Netflix Prize 竞赛开始以来所颁发的各个奖项，以及获奖团队和相应的预测误差。

1.4 术语和记号

本论文的主要目的是发展和介绍应用于 Netflix Prize 问题的各种算法，这些算法基本涵盖了目前协同过滤（CF）里的各类算法。我们的侧重点在于各类算法

的独特性而不是它们真正应用于 Netflix Prize 竞赛所需要注意的算法细节（因为细节总是能在给出的相应参考文献中找到，而且它们往往不是算法的本质所在）。由于目前此竞赛已经结束，我们无法再提交预测评分数据从而获得 Quiz Set 上的预测误差用以验证具体算法的高效性。所以在本文中如果没有特别说明，我们并不使用第 1.3 节中介绍的 Qualifying Set ，而总是把整体训练数据集（WTS）分为两部分：FPTS （图1.1 中第四行左边的绿色部分）和 Probe Set（图1.1 中第四行左边的紫色部分）。默认情况下我们总是首先使用 FPTS 训练模型，得到模型相应的参数；然后把训练得到的模型应用于预测 Probe Set 中的评分值，得到模型在 Probe Set 上的均方根误差（称为 Probe RMSE ）。根据众多研究者在 Netflix Prize 问题上的模型实验，一个模型如果能获得更低的 Probe RMSE ，那么它也能在 Qualifying Set 上获得更低的 RMSE。所以上面这种使用数据集的方法在简化算法比较过程的同时也不会影响算法的比较结果。

综合上述原因，本文中如果没有特别说明，我们所说的训练数据集都是指 FPTS，而用于检验算法的测试数据集都是 Probe Set 。默认情况下我们所说的预测误差是指 Probe Set 上的 RMSE，也即 Probe RMSE 。

我们使用 U 和 M 分别表示数据集中包括的用户和电影[①]数量。对于 Netflix Prize 数据集来说，$U = 480,189$，而 $M = 17,770$。对应的小写字母 u 和 m 分别表示第 u 个用户和第 m 部电影。第 u 个用户对第 m 部电影的评分值记为 $r_{u,m}$ ，评分时间记为 $t_{u,m}$ 。模型对于 $r_{u,m}$ 的预测值通常记为 $\hat{r}_{u,m}$ 。

我们总是使用大写花体字母表示集合。例如使用 $\mathcal{S}$ 表示合法的评分值集合，对于 Netflix Prize 数据集而言 $\mathcal{S} = \{1,2,3,4,5\}$ ；使用 $\mathcal{P}$ 表示训练数据集（FPTS）中的用户-电影指标对，也即 $\mathcal{P}$ =$\{(u,m)|r_{u,m}$ 在训练集中已知$\}$（有时我们也会使用 $\mathcal{P}$ 直接代表训练数据集）；使用 $\mathcal{R}$ 代表 Probe Set ，也即模型的测试数据集。我们也使用 $\mathcal{P}_u$ 表示 $\mathcal{P}$ 中第 u 个用户评分的所有电影指标集合，即 $\mathcal{P}_u = \{m|(u,m) \in \mathcal{P}\}$ ；对称地，使用 $\mathcal{P}^m$ 表示 $\mathcal{P}$ 中对第 m 部电影提供了评分的所有用户集合，即 $\mathcal{P}^m = \{u|(u,m) \in \mathcal{P}\}$ 。

我们一般使用黑体字母表示张量。列向量一般使用小写黑体字母表示，而

[①]在协同过滤里产品通常被称为 *item*，对于 Netflix Prize 问题而言 item 就是电影。

记号	定义	取值（范围）
$\mathcal{S}$	合法的评分值集合	$\{1,2,3,4,5\}$
U	用户数量	$480,189$
M	电影（或产品）数量	$17,770$
u	第 u 个用户	$\{0,1,...,U\}$
m	第 m 部电影	$\{0,1,...,M\}$
$r_{u,m}$	用户 u 对电影 m 的评分值	$\mathcal{S}$
$t_{u,m}$	用户 u 对电影 m 的评分时间	非负整数
$\hat{r}_{u,m}$	模型对于 $r_{u,m}$ 的预测值	$[1.0,5.0]$
$\mathcal{P}$	训练数据集中的用户-电影指标对	$\{(u,m)\|r_{u,m}$ 在训练集中已知$\}$
$\mathcal{R}$	Probe Set 中的用户-电影指标对	$\{(u,m)\|r_{u,m}$ 在 Probe Set 中已知$\}$
$\mathcal{A}$	所有数据的用户-电影指标对	$\mathcal{A}=\mathcal{P}\cup\mathcal{R}$
$\mathcal{P}_u$	$\mathcal{P}$ 中用户 u 评分的所有电影指标集合	$\{m\|(u,m)\in\mathcal{P}\}$
$\mathcal{P}^m$	$\mathcal{P}$ 中为电影 m 提供了评分的所有用户集合	$\{u\|(u,m)\in\mathcal{P}\}$
RMSE	$\mathcal{R}$ 上的均方根误差	$\sqrt{\frac{1}{\|\mathcal{R}\|}\sum_{(u,m)\in\mathcal{R}}(\hat{r}_{u,m}-r_{u,m})^2}$

表 1.2: 本文中使用的各种记号与术语

矩阵一般使用大写黑体字母表示。例如在第五章中，$\mathbf{p}_u$ 一般用来表示用户 u 的因子列向量，而 $\mathbf{P}$ 一般用来表示由所有用户的因子向量形成的因子矩阵，即 $\mathbf{P}=(\mathbf{p}_1,\ldots,\mathbf{p}_U)^T$。对于随机变量，我们通常使用大写黑斜体字母表示。例如在第五章中，$\boldsymbol{P}$ 一般被用来表示用户因子矩阵（$\mathbf{P}$）对应的随机张量。对于超参数（*hyper-parameter/meta-parameter*），我们一般使用小写希腊字母表示。例如我们通常使用 η 表示梯度下降法中的学习率（*learning rate*），而 λ 表示惩罚（*regularization*）参数。更详细的说明请见表 1.2。

第二章　评分预处理方法

数据预处理（*pre-processing*）是数据分析中的重要步骤之一，它可以去除数据中掩盖真实信息的噪音，起到净化数据的作用。机器学习模型应用在预处理后的数据上才能更加真实地揭示数据中包含的有用信息。 Netflix Prize 数据集上的实验表明数据预处理方法对很多模型都至关重要。

本章我们将主要介绍预处理 Netflix Prize 评分的各种方法。通过这些方法，我们可以去除用户或电影单方面对评分的影响，从而为各种学习用户与电影之间相互作用的 CF 模型净化评分。我们在之后章节介绍的很多 CF 模型都使用这些方法预处理评分。

2.1　全局作用（Global Effects）

全局作用（*Global Effects*，简记为 GEs）描述的是用户对电影的评分数据本身所体现出来的各种趋势。例如有些用户对所有电影的评分都偏高，他（她）们可能以 4 分为基准，对不喜欢的电影给予 3 分的评分，而对喜欢的电影给予 5 分的评分；而另外一些用户可能对所有电影的评分都偏低，这些用户可能以 2 分为基准，对自己喜欢的电影只给予 3 分的评分。同样，有些电影较之其他电影偏向于获得较高的评分（可能是电影的剧情、画面、音乐、演员或上映时间等因素造成的）[6]。

除了上面所说的基本趋势，有些趋势可能依赖于更多的因素。比如一个用户对电影的评分可能会受此用户当时的心情、所处的环境以及其他电影等诸多因素的影响。同样一部电影所获得的评分也可能随着时间变化会有很大波动。

GEs 首先由 Bell & Koren[6]提出，而后 Töscher 等 [88–90] 对其进行了进一步的扩展。下面我们简单介绍学习各种 GE 的方法[6]。

GEs 学习的主要想法是每次只考虑一种 GE，然后依次得到所有的 GEs。在学习某个 GE 的过程中，我们总是使用前面已得到的所有 GEs 的预测残差

（*residual*）作为应变量（*dependent variables*）。所以在下面的方法中，除了计算第一个 GE（即 Overall mean，见表 2.1）所使用的 $r_{u,m}$ 为原始评分，计算其他 GEs 所使用的 $r_{u,m}$ 都为预测残差。

对于所有的 GEs（除 Overall mean 外[①]），我们的目标是为每个用户或每部电影估计一个特定参数。下面我们主要介绍如何学习基于用户（user-specific）的参数，因为基于电影的参数学习方法完全类似。记 $x_{u,m}$ 为与第 u 个用户和第 m 部电影相关的解释型变量（*explanatory variable*）。我们首先中心化（*center*）每个用户的所有 $x_{u,m}$，也即使第 u 个用户的所有 $x_{u,m}$ 的均值为 0。然后我们利用下面的模型计算第 u 个用户的参数 θ_u：

$$r_{u,m} = \theta_u x_{u,m} + \text{error} \quad 。 \tag{2.1}$$

如果第 u 个用户提供了充分多的评分，我们可以获得 θ_u 的无偏估计：

$$\tilde{\theta}_u = \frac{\sum_{m\in\mathcal{P}_u} r_{u,m}x_{u,m}}{\sum_{m\in\mathcal{P}_u} x_{u,m}^2} \quad 。 \tag{2.2}$$

但由于数据的稀疏性，一些用户可能只对很少的电影提供了评分，这些用户对应的 $\tilde{\theta}_u$ 不再是 θ_u 的可靠估计。为了避免过度拟合（*overfitting*），我们依据 $\mathcal{P}_u$ 的大小对获得的 $\tilde{\theta}_u$ 进行压缩：

$$\hat{\theta}_u = \frac{|\mathcal{P}_u|\tilde{\theta}_u}{|\mathcal{P}_u| + \alpha} \quad , \tag{2.3}$$

其中 α 为用户给定的压缩参数。对应于不同 GE 的多个 α 值可以通过交叉验证（*cross validation*）[6]、APT[88, 89]或其他超参数学习方法（见附录 C）逐个确定[6, 88]，也可以同时确定[89]。

类似(2.3)中的这种使通过已知数据计算出的参数值朝着一个公共值（(2.3)中为 0）压缩的方法在本文中随处可见。这种通过参与数据量的多少来决定压缩程度的方法首先由 Bell 等 [12] 引入到 Netflix Prize 问题中，并最终被证实可以很

[①] Overall mean 对应的参数只有一个——整个训练数据集的评分平均值 $\bar{r}$，即：

$$\bar{r} = \frac{1}{|\mathcal{P}|}\sum_{(u,m)\in\mathcal{P}} r_{u,m}^{\text{real}} \quad 。$$

No.	Global Effect	Probe RMSE	α
0	Overall mean	1.1296	0
1	Movie × 1	1.0526	22
2	User × 1	0.9840	7.5
3	User × Time(user)$^{1/2}$	0.9802	435
4	User × Time(movie)$^{1/2}$	0.9778	125
5	Movie × Time(movie)$^{1/2}$	0.9760	4100
6	Movie × Time(user)$^{1/2}$	0.9752	420
7	User × Average(movie)	0.9711	68
8	User × Support(movie)	0.9682	76
9	Movie × Average(user)	0.9671	140
10	Movie × Support(user)	0.9659	10^{10}
11	Movie × AvgMovieProductionYear(user)	0.9635	380
12	User × ProductionYear(movie)	0.9623	170
13	User × StandardDeviation(movie)	0.9611	130
14	Movie × StandardDeviation(user)	0.9604	3700
15	User × Average(movie) (on raw residuals/ratings)		
16	Movie × Average(user) (on raw residuals/ratings)		

表 2.1: 全局作用（GEs）以及由它们所产生的 Probe RMSE （表中数据来源于 [6, 88–90]）

大程度地降低诸多算法的预测误差。这种压缩方法的合理性可以通过统计中的 Bayes 方法得到证实，详见附录 A 或 [6]。

表 2.1 中列出了一些常用于 Netflix Prize 问题的 GEs，其中第 0 – 10 个 GEs 在 [6] 中被提出，第 11 – 14 个在 [90] 中被提出[②]，第 15 – 16 个在 [89] 中被提出[③]。表中第一列表示各个 GE 被考虑的顺序；第二列表示 GE 的名称；第三列表示使用所有已获得的 GEs （包括本 GE）作为预测模型在 Probe Set 上获得的 RMSE；第四列表示压缩系数的取值。其中第二列命名的意义为：在“×”之前

[②]根据 [89, 90] 中的说明，[88] 中对第 3 – 7 个 GEs 的描述应该存在打印错误。

[③]Töscher 等 [89] 对这两个 GEs 给出的解释是“ $x_{u,m}$ 是基于原始评分或残差得到的，而不是基于前面 GEs 得到的残差”。但这种解释似乎很让人费解，因为其他 GEs 对应的 $x_{u,m}$ 也不是“基于前面 GEs 得到的残差”而获得的。

的“User”（或“Movie”）代表本 GE 是基于用户（或基于电影）的；而在“×”之后的变量名称代表 $x_{u,m}$ 的取值含义。下面我们以第 3 个 GE 为例，详细说明上面的方法如何得到此 GE 对应的参数。

第 3 个 GE 的名称为“User × Time(user)$^{1/2}$”，也就是它是基于用户的（见(2.1)）；而“Time(user)”表示的是某用户自从第一次对某部电影给出评分后到给出此次评分中间所间隔的天数，所以这里的 $x_{u,m}$ 是

$$x_{u,m} = \sqrt{t_{u,m} - \min_{n \in \mathcal{P}_u} t_{u,n}} \quad 。$$

正如我们之前说的，在真正使用 $x_{u,m}$ 之前，我们首先应该对它进行中心化：

$$\hat{x}_{u,m} = x_{u,m} - \bar{x}_u \quad , \tag{2.4}$$

其中 $\bar{x}_u = \frac{1}{|\mathcal{P}_u|} \sum_{m \in \mathcal{P}_u} x_{u,m}$。而(2.2)中所使用的 $r_{u,m}$ 为前面 GEs 的预测残差：

$$\hat{r}_{u,m} = r^{\text{real}}_{u,m} - \hat{r}^{(2)}_{u,m} \quad , \tag{2.5}$$

其中 $r^{\text{real}}_{u,m}$ 为第 u 个用户对第 m 部电影的原始评分，而 $\hat{r}^{(2)}_{u,m}$ 表示利用第 $0-2$ 个 GEs 得到的预测评分。

把从(2.4)和(2.5)中获得的 $\hat{x}_{u,m}$ 和 $\hat{r}_{u,m}$ 代入(2.2) 我们可以得到 $\tilde{\theta}_u$，然后利用 α（这里 $\alpha = 435$）和(2.3) 我们最终获得第 3 个 GE 对应的参数 $\hat{\theta}_u$（$u = 1, \ldots, U$）。利用这些 $\hat{\theta}_u$ 和(2.4)中的 $\hat{x}_{u,m}$，我们可以获得利用第 $0-3$ 个 GEs 得到的预测评分：

$$\hat{r}^{(3)}_{u,m} = \hat{r}^{(2)}_{u,m} + \hat{\theta}_u \hat{x}_{u,m} \quad 。 \tag{2.6}$$

表 2.1 中第 4 个 GE 名称中的“Time(movie)”表示的是某部电影自从第一次被评分后到此次评分所间隔的天数；第 7, 9 个中的“Average”表示对应用户（或电影）的平均评分值；第 8, 10 个中的“Support”表示对应用户（或电影）的评分数量；第 11 个中的“AvgMovieProductionYear(user)”表示某个用户评分的所有电影的平均上映年份；其他 GEs 所代表的意义都可以容易地根据它对应的名称判断出来，详见 [6, 88–90]。

关于如何进一步改进 GEs 的预测效果，很多人都进行了更加深入和有效的探讨。例如在 Netflix Prize 的官方论坛[62]中网友 daviticus 就提出了比(2.3)更有效的压缩方法④。Töscher 等 [88, 89] 也进一步提出了所谓的全局时间作用（*Global Time Effects*，简记为 GTEs）。GTEs 加入了更多的时间因素，相比于(2.1)使用了更加灵活的模型。GTEs 最终可以获得 0.9509 的 Probe RMSE，预测精度已经超过了 Netflix 的 Cinematch 推荐系统。

2.2 基准预测模型（Baseline Predictors）

协同过滤（ CF ）中的模型一般都是通过学习用户与产品之间的相互作用进而获得最终的预测评分。但通过观察很多研究者发现 Netflix Prize 中的部分评分值还受到来自于用户或电影单方面的影响，而并非只受用户与电影之间的相互作用所影响[48]。这种非相互关系产生的评分影响的一个很好例子是我们在第 2.1 节中提到的一些用户（或电影）的平均评分较之其他用户（或电影）的评分明显偏高（或偏低）。

所谓的基准预测模型（*Baseline Predictors (BPs)*），其目标就是找出 CF 数据集中不能由相互作用所解释的那些规律（或趋势）。原始评分数据利用 BPs 消除掉非相互作用的评分因素后可以帮助其他模型更好地挖掘数据中隐藏的真正相互作用。下面我们介绍一些常用于 Netflix Prize 问题的 BPs 。

BP1

一个只简单计入评分中用户偏差（user biases）和电影偏差（movie biases）因素的 BP[46–48]（以下称之为 *BP1*）如下：

$$b_{u,m} = \bar{r} + b_u + b_m \quad , \tag{2.7}$$

其中 $\bar{r}$ 为全局评分平均值（overall mean），其计算方法见第 2.1 节；b_u 代表用户 u 的评分偏差，b_m 代表电影 m 获得的评分偏差。例如，我们想获得用户张三对电影 Titanic 的 BP1 预测评分。假设现在的全局平均评分 $\bar{r} = 3.6$，而 Titanic 是部

④详见网址：http://www.netflixprize.com//community/viewtopic.php?id=904 。

很好的电影，它获得的评分倾向于比平均评分高 0.6，但张三的评分倾向于比平均评分低 0.4，所以最终的 BP1 预测评分为 $3.6 - 0.4 + 0.6 = 3.8$ [46–48]。

对比第 2.1 节中的表 2.1，我们不难发现(2.7)中的 BP1 其实就是使用了第 $0-2$ 个 GEs 得到的预测模型。所以我们在第 2.1 节中使用的参数计算方法也适合于这里。首先，对于第 m 部电影（$m = 1, \ldots, M$），我们使用下式计算 b_m：

$$b_m = \frac{\sum_{u \in \mathcal{P}^m} (r_{u,m} - \bar{r})}{\lambda_1 + |\mathcal{P}^m|} \quad , \tag{2.8}$$

其次，对于第 u 个用户（$u = 1, \ldots, U$），我们使用下式计算 b_u：

$$b_u = \frac{\sum_{m \in \mathcal{P}_u} (r_{u,m} - \bar{r} - b_m)}{\lambda_2 + |\mathcal{P}_u|} \quad 。 \tag{2.9}$$

上面两式中的 λ_1 和 λ_2 为压缩系数，它们的值可以通过手动设置或使用附录 C 中介绍的方法确定。Koren [47, 48] 建议使用 $\lambda_1 = 25$ 和 $\lambda_2 = 10$。

我们也可以通过直接最小化 BP1 在训练数据集上的预测误差来获得更加精确的 b_u 和 b_m，也就是最小化目标函数：

$$\min_{b_*} \sum_{(u,m) \in \mathcal{P}} \left[(r_{u,m} - \bar{r} - b_u - b_m)^2 + \lambda_3 b_u^2 + \lambda_4 b_m^2 \right] \quad , \tag{2.10}$$

其中 b_* 代表所有的 b_u 和 b_m，而 λ_3 和 λ_4 为惩罚系数。(2.10)中的第一项 $\sum_{(u,m) \in \mathcal{P}} (r_{u,m} - \bar{r} - b_u - b_m)^2$ 表明我们希望获得 b_u 和 b_m 以便产生的 BP1 在训练数据集 $\mathcal{P}$ 上获得最低的 RMSE，第二项 $\sum_{(u,m) \in \mathcal{P}} [\lambda_3 b_u^2 + \lambda_4 b_m^2]$ 为惩罚项，其目的是减轻或避免过度拟合，它和*Ridge* 回归[34]中使用的约束具有同样的作用。Piotte & Chabbert [69] 也给出了更加灵活的惩罚项：

$$\sum_{(u,m) \in \mathcal{P}} \left\{ \left(\lambda_3 + \frac{\alpha_1}{|\mathcal{P}_u|} \right) b_u^2 + \left(\lambda_4 + \frac{\alpha_2}{|\mathcal{P}^m|} \right) b_m^2 \right\} \quad 。 \tag{2.11}$$

上面的惩罚项中对每个参数 b_* 的惩罚大小依赖于计算它所使用的数据量多少，使用的数据越多，惩罚越小；使用的数据越少，惩罚越大。

(2.10)中的最小二乘问题可以通过随机梯度下降（*stochastic gradient descent*）[48]或交替最小二乘回归（*alternate least squares regression*）[69]方法求解。相比于

前面(2.8)和(2.9)中的依次计算方法，求解(2.10)可以保证 b_u 和 b_m 被同等对待而没有先后之别。

我们在 [98] 中也建议使用另一种对称的 BP [⑤]（我们称之为*Average Predictor (AP)*）：

$$\hat{r}_{u,m} = \bar{r}_u + \bar{r}^m - \bar{r} \quad , \tag{2.12}$$

其中 $\bar{r}$ 定义如前，为全局评分平均值；$\bar{r}_u$ 为压缩后的第 u 个用户的平均评分：

$$\bar{r}_u = \frac{|\mathcal{P}_u|\tilde{r}_u + \lambda_5 \bar{r}}{|\mathcal{P}_u| + \lambda_5} = \bar{r} + \frac{|\mathcal{P}_u|}{|\mathcal{P}_u| + \lambda_5}(\tilde{r}_u - \bar{r}) \quad ; \tag{2.13}$$

类似地，$\bar{r}^m$ 为压缩后的第 m 部电影的平均评分：

$$\bar{r}^m = \frac{|\mathcal{P}^m|\tilde{r}^m + \lambda_6 \bar{r}}{|\mathcal{P}^m| + \lambda_6} = \bar{r} + \frac{|\mathcal{P}^m|}{|\mathcal{P}^m| + \lambda_6}(\tilde{r}^m - \bar{r}) \quad , \tag{2.14}$$

其中 $\tilde{r}_u$ 和 $\tilde{r}_m$ 分别为第 u 个用户和第 m 部电影的平均评分，而 λ_5 和 λ_6 为其对应的压缩系数。我们建议使用 $\lambda_5 = 50$ 和 $\lambda_6 = 100$ [98]。

使用(2.13)和(2.14)，我们可以重写(2.12)：

$$\begin{aligned} \hat{r}_{u,m} &= \bar{r}_u + \bar{r}^m - \bar{r} \\ &= \frac{|\mathcal{P}_u|\tilde{r}_u + \lambda_5 \bar{r}}{|\mathcal{P}_u| + \lambda_5} + \frac{|\mathcal{P}^m|\tilde{r}^m + \lambda_6 \bar{r}}{|\mathcal{P}^m| + \lambda_6} - \bar{r} \\ &= \frac{|\mathcal{P}_u|}{|\mathcal{P}_u| + \lambda_5}(\tilde{r}_u - \bar{r}) + \frac{|\mathcal{P}^m|}{|\mathcal{P}^m| + \lambda_6}(\tilde{r}^m - \bar{r}) + \bar{r} \quad 。 \end{aligned} \tag{2.15}$$

可见，如果我们把上式中的 $\frac{|\mathcal{P}_u|}{|\mathcal{P}_u|+\lambda_5}(\tilde{r}_u - \bar{r})$ 和 $\frac{|\mathcal{P}^m|}{|\mathcal{P}^m|+\lambda_6}(\tilde{r}^m - \bar{r})$ 分别看成(2.7)中的 b_u 和 b_m，则 AP 就成为一种特化的 BP1。

Piotte & Chabbert [69] 建议为 BP1 中的电影偏差 b_m 引入用户的评分尺度，从而得到以下的 BP 模型：

$$b_{u,m} = \bar{r} + b_u + b_m(1 + s_u) \quad 。 \tag{2.16}$$

以上模型中的参数可以通过最小化下面的目标函数获得：

$$\begin{aligned} \min_{b_*, s_*} \sum_{(u,m)\in\mathcal{P}} \Big[& (r_{u,m} - \bar{r} - b_u - b_m(1 + s_u))^2 \\ & + \left(\lambda_7 + \frac{\alpha_3}{|\mathcal{P}_u|}\right) b_u^2 + \left(\lambda_8 + \frac{\alpha_4}{|\mathcal{P}_u|}\right) s_u^2 + \left(\lambda_9 + \frac{\alpha_5}{|\mathcal{P}^m|}\right) b_m^2 \Big] \quad 。 \end{aligned} \tag{2.17}$$

[⑤] 这种方法在 [39] 中也被使用。

与最小化(2.10)一样，我们可以使用随机梯度下降或交替最小二乘方法来求解上面的最小化问题。超参数的确定方法也和之前介绍的一样，详见附录 C 。

BP2

BP1 假设用户或电影的评分偏差随着时间的变化是固定不变的，这种假设在真实环境中往往并不合理。

一个用户对某部电影的理解可能因为他（她）最近观看了其他电影而有所改变。例如一个用户为“Transformers”真人版的评分可能会因为他（她）最近是否观看了其卡通版而有所不同。而且，对于 Netflix Prize 数据集而言，一个用户往往代表的是一个家庭，而不是一个个人。可能家庭中的不同成员会在不同的时间观看电影并给出评分。

对于电影而言，一部电影可能会因为里面某个演员逐渐知名而被观众重新发掘。周星驰因为主演了“唐伯虎点秋香”等经典喜剧而家喻户晓，这些电影的卖座同时也让观众更加关注周星驰早期并不为人熟知的作品（现在大家连周星驰出道时跑龙套的电影都耳熟能详了）。另一个很好的例子是，一部新电影的上映可能会掀起同类电影的狂潮。例如 2009 年末灾难大片“2012”的上映在全世界掀起了一股“灾难”热潮，它把很多早期类似的电影再次带回了人们的视线之中，如 2005 年上映的经典灾难片“The Day After Tommorrow”。

考虑到上面所提的各种现实因素，一种对 BP1 的自然推广就是让用户和电影偏差随着时间而不断变化：

$$b_{u,m} = \bar{r} + b_u(t_{u,m}) + b_m(t_{u,m}) \quad , \tag{2.18}$$

其中 $b_u(t)$ 和 $b_m(t)$ 为时间的实值函数。我们称此模型为*BP2* 。

$b_u(\cdot)$ 和 $b_m(\cdot)$ 所取的函数形式应该尽量反映真实环境中的时间变化作用。一部电影的流行程度一般不会在短时间内（比如一两天内）变化很大，这种变化往往需要较长的时间才能体现出来。所以，对于电影偏差，我们可以通过对时间进行分段（*bin*）处理，然后在每段中使用相同的偏差值。划分的段数越多，对电影偏差刻画就越精细，但同时每段中估计偏差值所能使用的数据就越少，估计出的

值也就越不可靠。对于 Netflix Prize，Koren [48] 建议把整个时间段分成 30 段，每段的时间大约是连续的十周。所以最终电影偏差的形式为：

$$b_m(t) = b_m + b_{m,\mathrm{Bin}(t)} \quad , \tag{2.19}$$

其中函数 $\mathrm{Bin}(t)$ 把时间 t 映射为 1 至 30 的整数。

相对于电影而言，一个用户对电影的评分可能会在短期内变化很大。一个用户可能前一天心情很好，对所有的电影都给予偏高的评分。等他（她）睡完一觉后突然意识到自己还要为现在居住的“蜗居[⑥]”偿还二十年的债务时，他（她）可能会“痛恨”所有他（她）将要给予评分的电影。而且正如前面所说，在 Netflix Prize 数据中，一个用户可能并不是一个个人，所以这个用户在短时间内呈现出不同的评分趋势是极有可能的。

为简便起见，我们首先使用一个线性函数来获得用户偏差随着时间不断变化的作用。记第 u 个用户的平均评分时间为 $\bar{t}_u$，当此用户在时间 t 对一部电影给予了评分，那么这个评分的时间偏差（*time deviation*）定义为：

$$\mathrm{dev}_u(t) = \mathrm{sign}(t - t_u) \cdot |t - \bar{t}_u|^{\beta} \quad 。 \tag{2.20}$$

这里 $|t - \bar{t}_u|$ 表明时间 t 与 $\bar{t}_u$ 之间相差的天数，β 的值可以通过在 Probe Set 上使用交叉验证或其他超参数学习方法（见附录 C）确定，这里选定 $\beta = 0.4$。那么我们第一个随着时间变化的用户偏差定义如下：

$$b_u^{(1)}(t) = b_u + \alpha_u \cdot \mathrm{dev}_u(t) \quad , \tag{2.21}$$

其中的 b_u 和 α_u 为待学习的模型参数。

上面的线性函数捕捉的是用户偏差随着时间不断变化的作用，但正如我们之前所言，一个用户在非常短暂的时间内其偏差可能发生很大变化。从 Netflix Prize 数据中我们可以发现，一些用户在一天内的评分倾向于集中在某个值附近。我们第二个带有时间变化效应的用户偏差试着捕捉用户在单日内的评分趋势，其定义如下：

$$b_u^{(2)}(t) = b_u + b_{u,t} \quad , \tag{2.22}$$

[⑥]当本文写到这里时，“蜗居”是中国最受欢迎的反映现实生活的电视剧。

其中的 b_u 和 $b_{u,t}$ 为待学习的模型参数。

在 Netflix Prize 数据集中，平均来说每个用户提供的评分大概分散在 40 天当中，也就是说为了描述一个用户的评分偏差，大概需要 40 个模型参数。但(2.22)的局限性是只能捕捉用户的单日作用差异，却忽视了所有持续变化的作用。一个对其合理的加强是综合(2.21)和(2.22)中的偏差作用：

$$b_u^{(3)}(t) = b_u + \alpha_u \cdot \mathrm{dev}_u(t) + b_{u,t} \quad 。 \tag{2.23}$$

综合上面的电影偏差(2.19)和用户偏差(2.23)，我们得到 BP2 的一种具体形式：

$$b_{u,m} = \bar{r} + b_u + \alpha_u \cdot \mathrm{dev}_u(t_{u,m}) + b_{u,t_{u,m}} + b_m + b_{m,\mathrm{Bin}(t_{u,m})} \quad 。 \tag{2.24}$$

BP2 可以获得 0.9605 [48]的 Probe RMSE 。

上面所讨论的电影偏差与用户无关，但用户在不同的时间对电影偏差反应可能有所不同。对于某个用户，他（她）在心情较差时可能倾向于自己的判断，而在心情较好的时候可能会更加顾及其他用户对此电影的评价（电影偏差）。基于此想法，我们可以进一步为电影偏差引进与时间有关的用户尺度 $c_u(t)$ ，得到

$$b_{u,m} = \bar{r} + b_u + \alpha_u \cdot \mathrm{dev}_u(t_{u,m}) + b_{u,t_{u,m}} + (b_m + b_{m,\mathrm{Bin}(t_{u,m})}) \cdot c_u(t_{u,m}) \quad 。 \tag{2.25}$$

前面所讨论的用户偏差形式（见(2.21)-(2.23)）可以照搬至 $c_u(t)$ ，这里我们简单取 $c_u(t) = c_u + c_{u,t}$ 。 c_u 是其中的稳定部分，而 $c_{u,t}$ 则是每日变化的部分。

为了获得(2.25)中的模型参数，我们只要求解如下的最小化问题：

$$\begin{aligned} \min_{b_*, s_*} \sum_{(u,m)\in\mathcal{P}} \Big[& (r_{u,m} - \bar{r} - (b_u + \alpha_u \cdot \mathrm{dev}_u(t_{u,m}) + b_{u,t_{u,m}}) \\ & - (b_m + b_{m,\mathrm{Bin}(t_{u,m})}) \cdot (c_u + c_{u,t_{u,m}}))^2 + \lambda_1 \alpha_u^2 + \lambda_2 b_u^2 \\ & + \lambda_3 b_{u,t_{u,m}}^2 + \lambda_4 b_m^2 + \lambda_5 b_{m,\mathrm{Bin}(t_{u,m})}^2 + \lambda_6 (c_u - 1)^2 + \lambda_7 c_{u,t_{u,m}}^2 \Big] \quad 。 \end{aligned} \tag{2.26}$$

当使用随机梯度下降或交替最小二乘回归方法求解上面的最小化问题时，我们可以为 c_u 设定初始值 1 ，而为其他模型参数设定初始值 0 ，Koren [48] 也给出了超参数的详细取值。新引进的乘积因子 $c_u(t)$ 使得 BP2 的预测误差从 0.9605 降至 0.9555 。

BP3

对于 Netflix Prize 数据集而言，其中的评分可能是在不同背景下收集到的。当用户刚开始使用 Netflix 时，Netflix 可能邀请用户对已看过的电影进行评分以便学习用户的观看兴趣。而当用户使用 Netflix 一段时间以后，用户的评分可能就完全是其本身的自发行为。这些来源于不同途径的评分往往会有不同的特征[69]，所以 BPs 应该对它们区别对待。在 Netflix Prize 问题上我们无法精确判别评分的来源途径，但一个可以借助的有效信息是用户在同一天中评分的电影数量。有些用户在一天之内对成百上千部的电影进行了评分，这些电影不可能都是用户最近（更不用说当天）所观看的；而如果用户在当天只为一部电影提供了评分，那么很可能这部电影就是用户最近（甚至当天）观看的。下面我们借助评分数量信息进一步改进我们的 BP 预测效果。

记 $F_{u,m}$ 为用户 u 在时间 $t_{u,m}$ 内（ $t_{u,m}$ 当天）提供的评分数量， $f_{u,m}$ 为取整后的 $F_{u,m}$ 对数值，即 $f_{u,m}=\lfloor \log_a F_{u,m} \rfloor$ ，其中 a 为待定的超参数。

虽然 $f_{u,m}$ 由用户 u 所控制，但在 Netflix Prize 数据集上的实验结果表明它却主要对电影偏差有影响，而不是对用户 u 的偏差[48]。Koren [48] 认为一个用户在对很久以前观看的电影进行评分时他（她）所体现出来的依然是正常的喜好，而对于一部电影而言，它如果在被观看很久之后才被评分的话，对它进行评分的人要么特别喜欢它，要么对它深恶痛绝。这些原因导致了 $f_{u,m}$ 主要对电影偏差有影响。

基于上面的说明，我们扩展 BP2 为下面的*BP3*：

$$b_{u,m}=\bar{r}+b_u+\alpha_u\cdot \mathrm{dev}_u(t_{u,m})+b_{u,t_{u,m}}+(b_m+b_{m,\mathrm{Bin}(t_{u,m})})\cdot c_u(t_{u,m})+b_{m,f_{u,m}} \quad 。 \quad (2.27)$$

BP3 使得预测误差从 BP2 的 0.9555 降至 0.9278 （详见 [48]），可见其预测效果已经远优于 Netflix 的 Cinematch 系统（其预测误差为 0.9514 ）。

当使用上面介绍的 BP2 和 BP3 进行评分预测时，模型可能会碰到在预测某些评分时所需要的模型参数我们并不知道的情况。例如要预测的评分值是用户 u 在一个新的时间 t 内（用户 u 之前在这个时间内没有给出任何评分）给出的，那么，获得此预测值所需要的参数 $b_{u,t}$ 和 $c_u(t)$ 我们并不知道。一个简单的解决方法

就是直接使用参数默认值代替未知值[48]。例如，对于 BP3 (2.27)， $c_u(t_{u,m})$ 使用值 c_u ， $b_{u,t_{u,m}}$ 使用值 0 作为替代。

正如我们前面介绍的，BPs 可以获得很好的预测效果（如 BP3），但它们忽略了用户和电影之间所有的相互作用，所以我们不能直接使用它们来获得有意义的个性化推荐。BPs 的作用是清除数据中掩盖数据本质关系的噪音，使得清理后的数据更能反映出数据内部隐含的规律。所以 BPs 在 CF 问题中通常只被用来预处理（*preprocessing*）数据，而并不单独作为推荐模型 ⑦。

⑦ Netflix Prize 竞赛只要求对一些评分值进行预测，并不要求给出评分预测的解释。它更像是一个抽象的数学问题，而不是现实中的个性化推荐问题。所以很多团队（如 BellKor[48]）在最后提交的预测中其实也结合了直接使用 BPs 所获得的预测结果。

第三章　邻居模型

邻居模型通常也被称为 k-最近邻模型，或者简称为 kNN。KNN 模型可以获得精确的推荐结果并为结果给出合理的解释，它们是 CF 推荐系统中最早被使用也是直至目前最流行的一类模型。

初始的 kNN 主要是基于用户的（*user-based*）[35]，也就是说用户对产品的评分预测是基于与此用户相似的用户所给予的评分获得的。这类 kNN 模型假设如果两个用户过去有相似的兴趣，那么他们现在也同样保持相近的兴趣。之后，人们也对基于产品的（*item-based*）的 kNN 模型进行了广泛的研究[53, 75]。这类 kNN 模型假设一个用户现在保持着和过去相近的兴趣，也就是如果某个产品和用户过去喜欢的产品相似，那么此产品现在也很有可能被用户所喜欢。由于在很多实际问题中用户的数量远远多于产品的数量，相比于基于用户的 kNN，基于产品的 kNN 往往能获得更好的预测精度和可扩展性。例如，对于 Netflix Prize 问题，用户的数量大约为 $500,000$，而产品的数量才不到 $20,000$。基于产品的 kNN 的另一个优势是它具有更好的可解释性。用户熟悉自己过去喜欢的产品，但往往并不熟悉所谓的和自己相似的用户。实验表明，基于产品的 kNN 在 Netflix Prize 问题上较之基于用户的 kNN 能获得更低的预测误差[6, 75, 83]。由于此二类 kNN 相互对称，下面我们的讨论只局限于基于产品的 kNN，但其中的绝大部分方法都可以照搬至基于用户的 kNN。

本章中我们首先介绍了传统 kNN 模型的预测框架。虽然这类 kNN 模型既直观且容易实现，但它们也存在一些本质的缺陷，如强烈依赖相似度的计算方法和可能重复包含同一信息等等。之后我们介绍了 Bell & Koren [6] 和 Koren [46, 47] 提出的新 kNN 模型。这些新模型具有与传统模型近似的计算复杂度，但可以获得更加精确的预测结果。

3.1 传统的 kNN 模型

为了获得用户对产品的评分预测值，kNN 模型一般包括以下三步：

1. **计算相似度**

   这步中计算每对产品之间的相似度（*similarity*）。一些被广泛使用的相似度测度包括[16]：

   Pearson correlation

$$s^{\mathrm{p}}_{mn} = \frac{\sum_{v\in\mathcal{P}^{mn}}(r_{v,m}-\bar{r}^m)(r_{v,n}-\bar{r}^n)}{\sqrt{\sum_{v\in\mathcal{P}^{mn}}(r_{v,m}-\bar{r}^m)^2\sum_{v\in\mathcal{P}^{mn}}(r_{v,n}-\bar{r}^n)^2}} \quad , \tag{3.1}$$

   其中 $\bar{r}^m$ 和 $\bar{r}^n$ 分别表示电影 m 和 n 获得的评分平均值，而 $\mathcal{P}^{mn}$ 表示对电影 m 和 n 都提供了评分的用户集合，也即 $\mathcal{P}^{mn}=\mathcal{P}^m\cap\mathcal{P}^n$。

   Cosine

$$s^{\mathrm{c}}_{mn} = \frac{\sum_{v\in\mathcal{P}^{mn}} r_{v,m}r_{v,n}}{\sqrt{\sum_{v\in\mathcal{P}^{mn}} r^2_{v,m}\sum_{v\in\mathcal{P}^{mn}} r^2_{v,n}}} \quad 。 \tag{3.2}$$

   Adjusted Cosine

$$s^{\mathrm{ac}}_{mn} = \frac{\sum_{v\in\mathcal{P}^{mn}}(r_{v,m}-\bar{r}_v)(r_{v,n}-\bar{r}_v)}{\sqrt{\sum_{v\in\mathcal{P}^{mn}}(r_{v,m}-\bar{r}_v)^2\sum_{v\in\mathcal{P}^{mn}}(r_{v,n}-\bar{r}_v)^2}} \quad , \tag{3.3}$$

   其中 $\bar{r}_v$ 表示用户 v 的评分平均值。

   Squared Distance

$$s^{\mathrm{d}}_{mn} = \frac{\gamma_1}{\gamma_2+\sum_{v\in\mathcal{P}^{mn}}(r_{v,m}-r_{v,n})^2} \quad , \tag{3.4}$$

   其中 γ_1 和 γ_2 为超参数。

2. **选择邻居**

   为了预测用户 u 对电影 m 的评分值，我们首先从 $\mathcal{P}_u$ 中选取与电影 m 有最高相似度的特定数量的电影，这些电影形成 u–m 对的邻居（*neighborhood*），记为 $\mathcal{N}(m;u)$。

3. **产生预测值**

用户 u 对电影 m 的评分预测为上步获得的邻居 $\mathcal{N}(m;u)$ 中评分的加权平均值：

$$\hat{r}_{u,m} = b_{u,m} + \frac{\sum_{n\in\mathcal{N}(m;u)} s_{mn}(r_{u,n} - b_{u,n})}{\sum_{n\in\mathcal{N}(m;u)} s_{mn}} \quad , \tag{3.5}$$

其中 $b_{u,n}$ 为用户 u 对电影 n 的基准预测评分。这里的基准模型可以是任何可以产生预测评分的模型，如我们在第二章中所给出的预处理模型，或者任何其他更加精细的模型（如因子模型等）。

传统的 kNN 模型之间的主要差异在于不同的相似度计算方法以及预测公式(3.5)中的基准评分 $b_{u,n}$。由于基准评分可由其他模型得到，它并不是 kNN 的本质特征。在确定使用哪种基准评分后，改善 kNN 模型预测精度最关键的是如何更好地计算相似度矩阵。

通常共同对两部电影都给予评分的用户数量很少，也即 $\mathcal{P}^{mn}$ 通常包括很少的用户，故而只基于 $\mathcal{P}^{mn}$ 所得到的相似度往往并不可靠。Bell 等 [12] 建议使用压缩的方法来改进所获得相似度的可靠性，也即压缩原来的相似度 s_{mn} 为：

$$\tilde{s}_{mn} = \frac{|\mathcal{P}^{mn}|}{|\mathcal{P}^{mn}| + \gamma} \cdot s_{mn} \quad , \tag{3.6}$$

其中 γ 为压缩系数。 Netflix Prize 上的实验表明这种压缩相似度的方法可以很好地改进预测精度。

Töscher & Jahrer [88] 建议重新缩放（*rescale*）已获得的相似度以便进一步改进预测精度。例如我们可以按如下方式使用 *S*型（*sigmoid*）函数改进相似度的计算精度（更多的改进方式可见 [88]）：

- $$\hat{s}_{mn} = \sigma(\delta \cdot \tilde{s}_{mn} + \gamma) \quad , \tag{3.7}$$

其中 δ 和 γ 为超参数，而

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad 。 \tag{3.8}$$

- $$\hat{s}_{mn} = \sigma\left(\delta \cdot \tilde{s}_{mn} \cdot e^{-|\Delta t|/\beta} + \gamma\right) \quad , \tag{3.9}$$

  其中 δ 、 β 和 γ 为超参数，而 $|\Delta t| = |t_{u,m} - t_{u,n}|$ 为用户 u 对电影 m 和 n 给予评分所间隔的天数。

正如(3.9)中所建议的，评分时间的加入往往可以改进所获得的相似度精度，因为同一个用户的兴趣往往是随着时间在不断变化的。Piotte & Chabbert [69] 进一步把时间的作用加入到相似度的计算当中：

$$\hat{s}_{mn} = \frac{1}{1 + \gamma \Delta t} \cdot \tilde{s}_{mn} \quad , \tag{3.10}$$

其中 γ 为超参数，它的值可以通过交叉验证或附录 C 中介绍的方法确定。

虽然上面介绍的 kNN 模型既直观且容易编程实现，但同时它也存在一些本质缺陷[6, 11]。首先，(3.5)中并没有考虑到邻居 $\mathcal{N}(m;u)$ 中电影之间的相似性。如果 $\mathcal{N}(m;u)$ 中包含了三部极其相近的电影（如“指环王 I、II、III”），那么(3.5)中会重复三次地加入这三部电影中真正包含的信息。其次，(3.5)中所使用的加权权重总和被限制为 1 ，这种限制可能产生过度拟合。如果 $\mathcal{N}(m;u)$ 中所有的电影都与目标电影 m 很不相似（如相似度都小于某个阀值），此时我们应该忽略邻居中所包含的信息，而仅使用基准预测值。然而(3.5)中却并未考虑此种情形的特殊性，而仍是使用归一化的邻居信息。

为了克服传统 kNN 模型的以上缺点，Bell & Koren [12] 介绍了一种新的 kNN 模型。这种新模型中的相似度并不事先通过计算得到，而是在改进模型预测精度的过程中逐渐学习出来。它可以获得更好的预测精度，但同时需要更大得多的计算量。之后 Bell & Koren [6] 对这个新模型进行了改进，降低了这个模型的计算复杂度，并且保持了它的预测精度。下面我们介绍这种改进后的 kNN 模型。

3.2 邻居关系（Neighborhood Relationships）模型

这一节中使用的 $r_{u,m}$ 评分可以是原始评分，也可以是原始评分去除全局作用后的剩余评分。 $r_{u,m}$ 的不同取法并不会影响下面模型的建立过程。

假设我们现在要预测用户 u 对电影 m 的评分 $r_{u,m}$。首先利用电影之间的相似度（上面介绍的相似度测度都可以使用，如(3.1)）得到 $\mathcal{P}_u$ 中与电影 m 最相似的 K 部电影，其集合记为 $\mathcal{N}(m;u)$。K 一般可以取 20 到 50 中的整数值[6]。

确定 $\mathcal{N}(m;u)$ 后，我们使用下面的公式获得 $r_{u,m}$ 的预测值：

$$\hat{r}_{u,m} = \sum_{n\in\mathcal{N}(m;u)} w_{mn} r_{u,n} \quad , \tag{3.11}$$

其中插值权重（*interpolation weights*）$\{w_{mn}|n \in \mathcal{N}(m;u)\}$ 使用以下的方法获得。为了简化记号，我们不妨假设 $\mathcal{N}(m;u) = \{1,\ldots,K\}$。

既然我们希望利用适当的插值权重 $\{w_{mn}|n \in \mathcal{N}(m;u)\}$ 获得对 $r_{u,m}$ 的精确预测，我们有理由相信这些插值权重也可以被用来获得其他用户对电影 m 的评分预测。在进行下面的推导之前，我们首先做一个“不可能”假设：除用户 u 之外，所有其他用户都对第 m 部电影以及 $\mathcal{N}(m;u)$ 中的所有电影给予了评分。有了这个假设之后，我们可以通过求解以下的二次最小化问题获得插值权重值：

$$\min_{\mathbf{w}} \sum_{v\neq u} \left(r_{v,m} - \sum_{n\in\mathcal{N}(m;u)} w_{mn} r_{v,n} \right)^2 \quad 。 \tag{3.12}$$

上面的最小化问题等价于求解以下的线性方程组：

$$\mathbf{A}\mathbf{w} = \mathbf{b} \quad , \tag{3.13}$$

其中矩阵 $\mathbf{A} \in \mathbb{R}^{K\times K}$ 的元素定义为

$$A_{kl} = \sum_{v\neq u} r_{v,k} r_{v,l} \quad , \tag{3.14}$$

而列向量 $\mathbf{b} \in \mathbb{R}^K$ 的元素定义为

$$b_k = \sum_{v\neq u} r_{v,k} r_{v,m} \quad 。 \tag{3.15}$$

上面计算 $\mathbf{A}$ 和 $\mathbf{b}$ 的方式依赖于我们之前的“不可能”假设，所以(3.14)和(3.15)并不具有可操作性。因此退而求其次，我们使用下面的公式获得相应的元素值：

$$\bar{A}_{kl} = \frac{1}{|\mathcal{P}^{kl}|} \sum_{v\in\mathcal{P}^{kl}} r_{v,k} r_{v,l} \quad , \tag{3.16}$$

$$\bar{b}_k = \frac{1}{|\mathcal{P}^{km}|} \sum_{v \in \mathcal{P}^{km}} r_{v,k} r_{v,m} \quad , \tag{3.17}$$

其中 $\mathcal{P}^{kl} = \mathcal{P}^k \cap \mathcal{P}^l$ ，而 $\mathcal{P}^{km}$ 的定义类似。

使用上面的公式计算时，不同元素依赖的评分值数量可能差别很大。当 $\mathcal{P}^{kl}$ 包含很少的元素时，使用(3.16)获得的相关元素值可能很不可靠。正如在相似度计算中使用压缩方法一样，我们这里也使用压缩方法来降低不可靠性。我们假设由(3.16)得到的矩阵 $\bar{\mathbf{A}}$ [①] 的对角线上的元素平均值记为 $\bar{a}_1$ ，非对角线上的元素平均值记为 $\bar{a}_2$ 。然后我们把 $\mathbf{A}$ 和 $\mathbf{b}$ 中的各个元素都朝着相应的平均值进行压缩：

$$\hat{A}_{kl} = \frac{|\mathcal{P}^{kl}| \cdot \bar{A}_{kl} + \beta \cdot \bar{a}}{|\mathcal{P}^{kl}| + \beta} \quad , \tag{3.18}$$

$$\hat{b}_k = \frac{|\mathcal{P}^{km}| \cdot \bar{b}_k + \beta \cdot \bar{a}}{|\mathcal{P}^{km}| + \beta} \quad , \tag{3.19}$$

其中 β 为压缩参数，[6] 建议 $\beta = 500$ 。当(3.18)和(3.19)中计算的是对角线元素值时， $\bar{a} = \bar{a}_1$ ；否则， $\bar{a} = \bar{a}_2$ 。

最后，我们使用上面计算得到的 $\hat{\mathbf{A}}$ 和 $\hat{\mathbf{b}}$ 来求解插值权重 $\mathbf{w}$ ：

$$\hat{\mathbf{A}}\mathbf{w} = \hat{\mathbf{b}} \quad , \tag{3.20}$$

然后将求解得到的 $\mathbf{w}$ 代入(3.11)最终获得 $r_{u,m}$ 的预测值。上面的(3.20)可以通过标准的线性方程求解器进行求解。对于 Netflix Prize 问题，实验表明如果约束插值权重 $\mathbf{w}$ 的所有元素非负我们可以获得更好的预测精度[6]。此时图 3.1 [②] 中给出的算法[6]可以用来求解 $\mathbf{w}$ 。

为了对 Probe Set 中的所有用户-电影对产生预测评分，我们需要求解 $|\mathcal{R}|$ 个线性方程组(3.20)。既然各个线性方程组中 $\hat{\mathbf{A}}$ 和 $\hat{\mathbf{b}}$ 所包含的元素有可能重叠，我们可以预计算（*precompute*）所有可能被使用的元素值，也即预计算所有的 $\hat{A}_{kl}$ （ $k = 1, \ldots, M;\ l = 1, \ldots, M$ ）。因此，为了获得某个用户-电影对的评分预测时，我们只需从已计算出的元素中找出相应的行和列组成所需的系数矩阵和右端项即可。

[①] 注：这里所说的 $\bar{\mathbf{A}}$ 其实表示的是整个大矩阵，其大小为 $\mathbb{R}^{M \times M}$ 。

[②] [6] 中给出的算法有一处笔误： $r \leftarrow Ax - b$ ，图 3.1 中我们已经更正为 $r \leftarrow b - Ax$ 。

NonNegativeQuadraticOpt ($A \in \mathbb{R}^{K\times K}, b \in \mathbb{R}^K$)
% Minimize $x^T Ax - 2b^T x$ *s.t.* $x \geqslant 0$

do
  $r \leftarrow b - Ax$ *% the residual, or "steepest gradient"*
  % find active variables - those that are pinned due to
  % nonnegativity constraints; set respective r_i*'s to zero*
  for $i = 1, \ldots, k$ **do**
    if $x_i = 0$ and $r_i < 0$ **then**
      $r_i \leftarrow 0$
    end if
  end for
  $\alpha \leftarrow \frac{r^T r}{r^T Ar}$ *% max step size*
  % adjust step size to prevent negative values:
  for $i = 1, \ldots, k$ **do**
    if $r_i < 0$ **then**
      $\alpha \leftarrow \min(\alpha, -x_i/r_i)$
    end if
  end for
  $x \leftarrow x + \alpha r$
while $\|r\| < \epsilon$ *% stop when residual is close to 0*
return x

图 3.1: [6] 中给出的求解具有非负约束的二次最小化问题的优化方法

如果我们使用去除第 $0-2$ 个GEs（见表 2.1）后的 $r_{u,m}$ 作为建立模型的评分，并且取邻居数量 $K = 20$，本节介绍的新 kNN 模型可以获得 0.9216 的预测误差，而使用第 3.1 节中的传统 kNN 模型只能获得 0.9431 的预测误差（相似度使用压缩后的 Pearson，邻居数量 $K = 20$），而且实验显示这两个模型所花费的计算时间相近（更多的实验结果可见 [6]）。可见新的 kNN 模型在获得了更加精确的预测评分的同时，还保持了传统 kNN 模型的计算量。

3.3 全局邻居（Global Neighborhood）模型

上节中我们介绍了一种通过模型学习而不是公式计算电影之间相关程度的 kNN 方法，这节中我们介绍另外一种获得电影之间相关性的方法。

记 w_{mn} 为从电影 n 到电影 m 的权重，它们的值将通过模型训练获得。一种

和原始 kNN 模型相近的评分预测模型如下：

$$\hat{r}_{u,m} = b_{u,m} + \sum_{n \in \mathcal{P}_u} (r_{u,n} - b_{u,n}) w_{mn} \quad , \tag{3.21}$$

其中 $b_{u,n}$ 为用户 u 对电影 n 的基准预测评分，这里我们简单取为第 2.2 节中介绍的 BP1：

$$b_{u,n} = \bar{r} + b_u + b_n \quad 。 \tag{3.22}$$

上面方程中的模型参数 b_u 和 b_n 可以使用计算 GEs 的方法获得，也可以通过模型训练获得，详见第 2.2 节。

一般来说 kNN 模型中的权值 w_{mn} 被解释为插值系数，而对应的 $r_{u,n} - b_{u,n}$ 被解释为相对于基准值的偏移（*offset*）。Koren [46, 47] 建议对(3.21)中的相关参数使用相反的解释，也即把 w_{mn} 解释为偏移量，表示 n 对 m 的可预测程度，而 $r_{u,n} - b_{u,n}$ 解释为相应系数。基于这种新的解释，我们可以利用隐式信息进一步加强(3.21)：

$$\hat{r}_{u,m} = b_{u,m} + \sum_{n \in \mathcal{P}_u} (r_{u,n} - b_{u,n}) w_{mn} + \sum_{n \in \mathcal{A}_u} c_{mn} \quad , \tag{3.23}$$

其中 c_{mn} 为模型参数，而 $\mathcal{A}_u$ 为用户 u 给予评分的所有电影集合，也即 $\mathcal{A}_u = \mathcal{P}_u \cup \mathcal{R}_u$。和 w_{mn} 类似，c_{mn} 可以解释为偏移量，其对应的系数为 1。

既然新的解释中 $r_{u,n} - b_{u,n}$ 为偏移对应的系数，我们可以对 $b_{u,m}$ 和 $b_{u,n}$ 使用不同的定义，也就是说可以推广(3.23)为

$$\hat{r}_{u,m} = \tilde{b}_{u,m} + \sum_{n \in \mathcal{P}_u} (r_{u,n} - b_{u,n}) w_{mn} + \sum_{n \in \mathcal{A}_u} c_{mn} \quad , \tag{3.24}$$

其中 $\tilde{b}_{u,m}$ 代表使用其他方法获得的预测值。这里我们使用下面的特殊形式：

$$\hat{r}_{u,m} = \bar{r} + b_u + b_m + \sum_{n \in \mathcal{P}_u} (r_{u,n} - b_{u,n}) w_{mn} + \sum_{n \in \mathcal{A}_u} c_{mn} \quad , \tag{3.25}$$

其中 $\bar{r}$ 为全局评分平均值，而 b_u 和 b_m 为模型参数。需要注意的是，上式中的 $b_{u,n}$ 是利用(3.22)提前计算出来的，在学习模型(3.25)中的参数之前，$b_{u,n}$ 已经是已知的常数，它的计算并不包括模型参数 b_u 或 b_m。

上面的模型(3.25)倾向于对提供评分更多的用户（$\mathcal{P}_u$ 或 $\mathcal{A}_u$ 包含更多元素）产生更大的偏差。所以我们修正它为下面的形式：

$$\hat{r}_{u,m} = \bar{r} + b_u + b_m + |\mathcal{P}_u|^{-\frac{1}{2}} \sum_{n \in \mathcal{P}_u} (r_{u,n} - b_{u,n}) w_{mn} + |\mathcal{A}_u|^{-\frac{1}{2}} \sum_{n \in \mathcal{A}_u} c_{mn} \quad 。 \tag{3.26}$$

记 $\mathcal{N}^K(m)$ 为与电影 m 最相似（相似度可由第 3.1 节中介绍的任何相似度测度获得）的 K 部电影组成的集合， $\mathcal{P}^K(m;u)$ 为 $\mathcal{P}_u \cap \mathcal{N}^K(m)$ ，以及 $\mathcal{A}^K(m;u)$ 为 $\mathcal{A}_u \cap \mathcal{N}^K(m)$ 。为了降低(3.26)的模型复杂度，我们修正它为：

$$\begin{aligned} \hat{r}_{u,m} =& \bar{r} + b_u + b_m + |\mathcal{P}^K(m;u)|^{-\frac{1}{2}} \sum_{n \in \mathcal{P}^K(m;u)} (r_{u,n} - b_{u,n}) w_{mn} \\ &+ |\mathcal{A}^K(m;u)|^{-\frac{1}{2}} \sum_{n \in \mathcal{A}^K(m;u)} c_{mn} \quad 。 \end{aligned} \tag{3.27}$$

如果取 $K = \infty$ （也即 $K \geq M$ ），则(3.27)就和(3.26)完全一致。

(3.27)是我们最终使用的预测模型，它的模型参数 b_* 、 w_* 和 c_* 可以通过求解下面的最小二乘优化问题而获得：

$$\begin{aligned} \min_{b_*, w_*, c_*} \sum_{(u,m) \in \mathcal{P}} \Bigg\{ \Big[r_{u,m} - \bar{r} - b_u - b_m - |\mathcal{P}^K(m;u)|^{-\frac{1}{2}} \sum_{n \in \mathcal{P}^K(m;u)} (r_{u,n} - b_{u,n}) w_{mn} \\ - |\mathcal{A}^K(m;u)|^{-\frac{1}{2}} \sum_{n \in \mathcal{A}^K(m;u)} c_{mn} \Big]^2 + \lambda \Big[b_u^2 + b_m^2 + \sum_{n \in \mathcal{P}^K(m;u)} w_{mn}^2 + \sum_{n \in \mathcal{A}^K(m;u)} c_{mn}^2 \Big] \Bigg\} \quad 。 \end{aligned} \tag{3.28}$$

这个最小化问题可以使用一般的标准线性代数软件包中的最小二乘求解器进行求解，但使用梯度下降法在这里可以获得更快的求解算法，具体更新步骤见算法 3.1 [46, 47]。

算法 3.1 (Global-kNN) 选择参数 K 、惩罚参数 λ 和学习率 η ；初始化模型参数 b_* 、 w_* 和 c_* （例如从均匀或正态分布中随机抽取出这些值，或简单地设定它们等于 0 ）。

1. 对每个用户-电影评分对 $(u,m) \in \mathcal{P}$ ：

(a) 计算评分残差 $e_{u,m} = r_{u,m} - \hat{r}_{u,m}$，其中 $\hat{r}_{u,m}$ 利用(3.27) 计算得到。

(b) 更新 b_u 和 b_m：

$$b_u \mathrel{+}= \eta \cdot (e_{u,m} - \lambda \cdot b_u) \quad ,$$
$$b_m \mathrel{+}= \eta \cdot (e_{u,m} - \lambda \cdot b_m) \quad 。$$

(c) $\forall\, n \in \mathcal{P}^K(m;u)$，更新 w_{mn}：

$$w_{mn} \mathrel{+}= \eta \cdot \left[\, |\mathcal{P}^K(m;u)|^{-\frac{1}{2}} \cdot e_{u,m} \cdot (r_{u,n} - b_{u,n}) - \lambda \cdot w_{mn} \,\right] \quad 。$$

(d) $\forall\, n \in \mathcal{A}^K(m;u)$，更新 c_{mn}：

$$c_{mn} \mathrel{+}= \eta \cdot \left[\, |\mathcal{A}^K(m;u)|^{-\frac{1}{2}} \cdot e_{u,m} - \lambda \cdot c_{mn} \,\right] \quad 。$$

2. 计算新的 Probe RMSE。如果新的 Probe RMSE 比之前的 Probe RMSE 小，则继续前面的更新步骤；否则终止算法。

Netflix Prize 数据集上的实验结果表明随着 K 的增加模型(3.27)的 Probe RMSE 单调下降。例如，当 $K = 250$ 时，模型误差为 0.9139；而当 $K = \infty$ 时，模型误差降为 0.9002。更具体的实验数据可参加 [46, 47]。

第四章　受限玻尔兹曼机

受限玻尔兹曼机（*Restricted Boltzmann Machines*，简记为 *RBM*）是一种两层的无向图模型[37, 80, 94, 95]。它包括一层可视化单元（*visible units*）和一层隐藏单元（*hidden units*），并且限制单元之间的连接只存在于不同层的单元之间，相同层的单元之间则不存在连接。Salakhutdinov 等 [74] 进一步把 RBM 引入 CF 问题，下面我们详细介绍如何把 RBM 应用于 Netflix Prize 数据集。

假设 RBM 中的隐藏层包括了 K 个隐藏单元 h_k （$k = 1, \ldots, K$），h_k 取值为 1 或 0 。如果我们把每个用户的所有评分组成的向量看成一个数据点，则训练数据集包括了 U 个数据点。不同的数据点之间共享可视化与隐藏单元之间的权重值与偏移，但它们拥有不同的隐藏单元值。所以如果两个用户对相同的电影进行了评分，那么他们在 RBM 中将使用相同的权重值，但他们对应的隐藏单元取值则可能大不相同。

在可视化与隐藏层之间使用不同的条件分布假设，我们可以获得不同的 RBM 模型。下面针对一些常用的条件假设我们具体介绍获得的各种 RBM 模型。

4.1　受限玻尔兹曼机（RBM）

Basic RBM

Basic RBM 假设在给定隐藏层的情况下可视化单元满足多项分布，而在给定可视化层时隐藏单元满足伯努利（*Bernoulli*）分布[74]，其数学表达式如下：

$$P(\boldsymbol{R}_{u,m} = s|\mathbf{H}) = \frac{\exp(b_{ms} + \sum_{k=1}^{K} h_{uk} w_{mks})}{\sum_{s'=1}^{S} \exp(b_{ms'} + \sum_{k=1}^{K} h_{uk} w_{mks'})} \quad , \tag{4.1}$$

$$P(\boldsymbol{H}_{uk} = 1|\mathbf{R}) = \sigma\left(b_k + \sum_{m \in \mathcal{P}_u} w_{mkr_{u,m}}\right) \quad , \tag{4.2}$$

其中 h_{uk} 表示用户 u 对应的第 k 个隐藏因子，$\mathbf{H} \triangleq (h_{uk}) \in \mathbb{R}^{U \times K}$；$\mathbf{W} \triangleq (w_{mks}) \in \mathbb{R}^{M \times K \times S}$ 为连接权重，而 b_{ms} 与 b_k 为偏移；$\sigma(\cdot)$ 为 sigmoid 函数(3.8)。Basic RBM

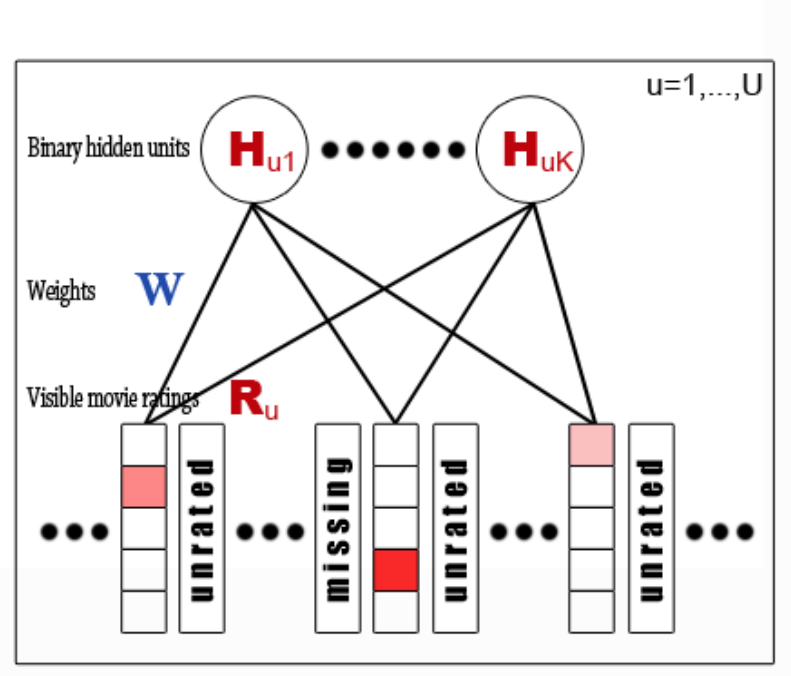

图 4.1: RBM 的图模型表示

的图模型表示见图 4.1。

在上面的条件假设下，我们可以获得 $\boldsymbol{R}$ 与 $\boldsymbol{H}$ 的联合分布：

$$P(\boldsymbol{R},\boldsymbol{H})=\prod_{u=1}^{U}P(\mathbf{r}_u,\mathbf{h}_u)=\prod_{u=1}^{U}\frac{\exp(-E(\mathbf{r}_u,\mathbf{h}_u))}{\sum_{\mathbf{r}'_u,\mathbf{h}'_u}\exp(-E(\mathbf{r}'_u,\mathbf{h}'_u))}\quad , \tag{4.3}$$

其中能量

$$E(\mathbf{r}_u,\mathbf{h}_u)=-\sum_{m\in\mathcal{P}_u}\sum_{k=1}^{K}w_{mkr_{u,m}}h_{uk}-\sum_{m\in\mathcal{P}_u}b_{mr_{u,m}}-\sum_{k=1}^{K}h_{uk}b_k\quad 。 \tag{4.4}$$

所以评分 $\mathbf{R}$ 的边缘分布为：

$$\begin{aligned}P(\mathbf{R})&=\sum_{\mathbf{H}}P(\mathbf{R},\mathbf{H})=\prod_{u=1}^{U}\sum_{\mathbf{h}_u}P(\mathbf{r}_u,\mathbf{h}_u)\\&=\prod_{u=1}^{U}\sum_{\mathbf{h}_u}\frac{\exp(-E(\mathbf{r}_u,\mathbf{h}_u))}{\sum_{\mathbf{r}'_u,\mathbf{h}'_u}\exp(-E(\mathbf{r}'_u,\mathbf{h}'_u))}\quad 。\end{aligned} \tag{4.5}$$

为了获得 Basic RBM 的模型权重 $\mathbf{W}$ 与偏移 b_*，我们使用梯度上升法获得 $\log P(\mathbf{R})=\sum_{u=1}^{U}\log P(\mathbf{r}_u)$ 的最大值。这里使用梯度法的关键步骤是计算

$\log P(\mathbf{R})$ 关于各个模型参数的偏导数，下面我们详细给出 $\log P(\mathbf{r}_u)$ 关于各个参数的偏导数。

$$\begin{aligned}
&\frac{\partial \log P(\mathbf{r}_u)}{w_{mks}} = \frac{1}{P(\mathbf{r}_u)} \frac{\partial P(\mathbf{r}_u)}{w_{mks}} \\
&= \frac{1}{P(\mathbf{r}_u)} \left(\sum_{\mathbf{h}_u} h_{uk} \delta(r_{u,m}, s) \cdot P(\mathbf{r}_u, \mathbf{h}_u) - P(\mathbf{r}_u) \sum_{\mathbf{r}'_u, \mathbf{h}'_u} h'_{uk} \delta(r'_{u,m}, s) \cdot P(\mathbf{r}'_u, \mathbf{h}'_u) \right) \\
&= \sum_{\mathbf{h}_u} h_{uk} \delta(r_{u,m}, s) \cdot P(\mathbf{h}_u | \mathbf{r}_u) - \sum_{\mathbf{r}'_u, \mathbf{h}'_u} h'_{uk} \delta(r'_{u,m}, s) \cdot P(\mathbf{r}'_u, \mathbf{h}'_u) \\
&= <h_{uk} \delta(r_{u,m}, s)>_{\text{data}} - <h_{uk} \delta(r_{u,m}, s)>_{\text{model}} \quad ,
\end{aligned} \tag{4.6}$$

其中函数

$$\delta(s_1, s_2) \triangleq \begin{cases} 1 \ , & \text{如果} s_1 = s_2 \quad ; \\ 0 \ , & \text{如果} s_1 \neq s_2 \quad ; \end{cases} \tag{4.7}$$

而

$$\begin{aligned}
<h_{uk} \delta(r_{u,m}, s)>_{\text{data}} &\triangleq \sum_{\mathbf{h}_u} h_{uk} \delta(r_{u,m}, s) \cdot P(\mathbf{h}_u | \mathbf{r}_u) \\
&= \delta(r_{u,m}, s) \cdot \sum_{\mathbf{h}_u} h_{uk} \cdot P(\mathbf{h}_u | \mathbf{r}_u) \\
&= \delta(r_{u,m}, s) \cdot \sigma \left(b_k + \sum_{m \in \mathcal{P}_u} w_{mk r_{u,m}} \right)
\end{aligned} \tag{4.8}$$

表示的是训练数据集包含的一种频率信息，它可以通过数据很容易地获得；

$$\begin{aligned}
<h_{uk} \delta(r_{u,m}, s)>_{\text{model}} &\triangleq \sum_{\mathbf{r}'_u, \mathbf{h}'_u} h'_{uk} \delta(r'_{u,m}, s) \cdot P(\mathbf{r}'_u, \mathbf{h}'_u) \\
&= \sum_{r'_{u,m}, h'_{uk}} h'_{uk} \delta(r'_{u,m}, s) \cdot P(r'_{u,m}, h'_{uk})
\end{aligned} \tag{4.9}$$

表示的是关于联合分布的期望，它的精确值很难获得。

为了避免计算 $<\cdot>_{\text{model}}$，Salakhutdinov 等 [74] 建议为(4.6)使用所谓的 *Constrastive Divergence (CD)*[36]近似：

$$\frac{\partial \log P(\mathbf{r}_u)}{w_{mks}} \simeq <h_{uk} \delta(r_{u,m}, s)>_{\text{data}} - <h_{uk} \delta(r_{u,m}, s)>_{\text{T}} \quad , \tag{4.10}$$

其中 $<\cdot>_{\mathrm{T}}$ 表示使用 *Gibbs* 抽样所获得的抽样分布下的期望，T 代表抽样的次数。我们通常在参数学习过程的开始阶段选取 $T=1$，伴随着学习过程的深入再不断增加 T 的值。只要 T 充分大，$<\cdot>_{\mathrm{T}}$ 与 $<\cdot>_{\mathrm{model}}$ 之间的差距可以任意小[21]，但对于真实问题我们往往不需要为 T 选取很大的值。

类似地我们可以获得 $\log P(\mathbf{r}_u)$ 关于参数 b_{ms} 与 b_k 的偏导数近似公式：

$$\frac{\partial \log P(\mathbf{r}_u)}{b_{ms}} \simeq <\delta(r_{u,m},s)>_{\mathrm{data}} - <\delta(r_{u,m},s)>_{\mathrm{T}} \quad , \tag{4.11}$$

$$\frac{\partial \log P(\mathbf{r}_u)}{b_k} \simeq <h_{uk}>_{\mathrm{data}} - <h_{uk}>_{\mathrm{T}} \quad 。 \tag{4.12}$$

在迭代开始前，我们可以为连接权重 $\mathbf{W}$ 和偏移 b_* 随机产生初始值，但实验表明 b_{ms} 选取如下初始值可以使得迭代过程更快收敛并最终获得更高的预测精度：

$$b_{ms} = \log\left(\frac{1+\sum_{u\in\mathcal{P}^m}\delta(r_{u,m},s)}{S+|\mathcal{P}^m|}\right) \quad 。 \tag{4.13}$$

算法 4.1 给出了使用非零动量梯度上升法更新模型参数的详细过程。

算法 4.1 (RBM) 选择隐藏单元数目 K、惩罚参数 λ_*、动量 μ 和学习率 η；随机初始化连接权重 $\mathbf{W}$ 和偏移 b_k，并使用(4.13)初始化偏移 b_{ms}；初始化临时变量 Δw_{mks}、Δb_{ms} 和 Δb_k 为 0。

1. 对每个用户 u：

   (a) 对每部电影 $m\in\mathcal{P}_u$，利用(4.10)和(4.11)计算相应偏导数的近似值，分别记为 δw_{mks} 和 δb_{ms}，然后使用下面的公式更新 w_{mks} 和 b_{ms}：

$$\Delta w_{mks} = \mu\cdot\Delta w_{mks} + \eta\cdot(\delta w_{mks} - \lambda_1\cdot w_{mks}) \quad , \tag{4.14}$$

$$w_{mks} \mathrel{+}= \Delta w_{mks} \quad ; \tag{4.15}$$

$$\Delta b_{ms} = \mu\cdot\Delta b_{ms} + \eta\cdot(\delta b_{ms} - \lambda_2\cdot b_{ms}) \quad , \tag{4.16}$$

$$b_{ms} \mathrel{+}= \Delta b_{ms} \quad 。 \tag{4.17}$$

   (b) 利用(4.12)计算 $\log(P(\mathbf{r}_u))$ 关于 b_k 的近似偏导数，记为 δb_k，然后使用下面的公式更新 b_k：

$$\Delta b_k = \mu\cdot\Delta b_k + \eta\cdot(\delta b_k - \lambda_2\cdot b_k) \quad , \tag{4.18}$$

$$b_k \mathrel{+}= \Delta b_k \quad 。 \tag{4.19}$$

2. 使用(4.20)计算评分预测值，并计算新的 Probe RMSE。如果新的 Probe RMSE 比之前的 Probe RMSE 小，则继续前面的更新步骤；否则终止算法。

在获得了模型参数后，我们可以使用下面的平均场（*mean field*）更新方法[①]获得用户 u 对电影 m 的预测评分[74]：

$$\hat{r}_{u,m} = \sum_{s=1}^{S} s \cdot P(\boldsymbol{R}_{u,m} = s|\hat{\mathbf{p}}_u) \quad ； \tag{4.20}$$

$$P(\boldsymbol{R}_{u,m} = s|\hat{\mathbf{p}}_u) = \frac{\exp(b_{ms} + \sum_{k=1}^{K} \hat{p}_{uk} w_{mks})}{\sum_{s'=1}^{S} \exp(b_{ms'} + \sum_{k=1}^{K} \hat{p}_{uk} w_{mks'})} \quad , \tag{4.21}$$

$$\hat{p}_{uk} = P(\boldsymbol{H}_{uk} = 1|\mathbf{R}) = \sigma\left(b_k + \sum_{m \in \mathcal{P}_u} w_{mkr_{u,m}}\right) \quad 。 \tag{4.22}$$

当隐藏单元数目 $K = 100$，学习率 $\eta = 0.001$，惩罚参数 $\lambda_1 = 0.001$ 及 $\lambda_2 = 0$，和动量 $\mu = 0$ 时，RBM 在 Netflix Prize 数据集上经过 14 次迭代后获得了 0.9142 的 Probe RMSE。

具有高斯隐藏单元的 RBM

具有高斯隐藏单元的 *RBM*（*RBM with Gaussian Hidden Units*，简记为 GHRBM）假设模型中的隐藏单元为高斯潜在变量，即在给定评分的条件下隐藏单元满足正态分布而不是伯努利分布(4.2)[74, 94]：

$$P(\boldsymbol{H}_{uk} = h|\mathbf{R}) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(h - b_k - \sum_{m \in \mathcal{P}_u} w_{mkr_{u,m}}\right)^2\right\} \quad 。 \tag{4.23}$$

GHRBM 同时使用了 Basic RBM 中的另一个多项分布假设(4.1)，它对应着 pLSA[38] 的无向图特化。

在上面新的假设下，GHRBM 对应的评分 $\mathbf{R}$ 的边缘分布仍然具有形式(4.5)，但其中的能量变为

$$E(\mathbf{r}_u, \mathbf{h}_u) = -\sum_{m \in \mathcal{P}_u} \sum_{k=1}^{K} w_{mkr_{u,m}} h_{uk} - \sum_{m \in \mathcal{P}_u} b_{mr_{u,m}} + \frac{1}{2} \sum_{k=1}^{K} (h_{uk} - b_k)^2 \quad 。 \tag{4.24}$$

[①] Salakhutdinov 等 [74] 也给出了另一种获得预测评分的方法。这种预测方法可以获得更加精确的预测值，但其所需的计算量也更大。

而且，GHRBM 中 $\log(P(\mathbf{r}_u))$ 关于模型参数的偏导数近似与 Basic RBM 中的一致，也即(4.10)、(4.11)和(4.12)，所以GHRBM 同样可以使用算法 4.1 学习模型中的参数。

在获得了模型参数后，类似地我们可以使用下面的平均场更新方法获得用户 u 对电影 m 的预测评分：

$$\hat{r}_{u,m} = \sum_{s=1}^{S} s \cdot P(\boldsymbol{R}_{u,m} = s|\hat{\mathbf{p}}_u) \quad ; \tag{4.25}$$

$$P(\boldsymbol{R}_{u,m} = s|\hat{\mathbf{p}}_u) = \frac{\exp(b_{ms} + \sum_{k=1}^{K} \hat{p}_{uk} w_{mks})}{\sum_{s'=1}^{S} \exp(b_{ms'} + \sum_{k=1}^{K} \hat{p}_{uk} w_{mks'})} \quad , \tag{4.26}$$

$$\hat{p}_{uk} = b_k + \sum_{m \in \mathcal{P}_u} w_{mk} r_{u,m} \quad 。 \tag{4.27}$$

当隐藏单元数目 $K = 100$，学习率 $\eta = 0.0001$，惩罚参数 $\lambda_1 = 0.0005$ 及 $\lambda_2 = 0$，和动量 $\mu = 0$ 时，GHRBM 在 Netflix Prize 数据集上经过 14 次迭代后获得了 0.9547 的 Probe RMSE。

具有高斯可视化单元的 RBM

具有高斯可视化单元的 *RBM*（*RBM with Gaussian Visible Units*，简记为 GVRBM）假设模型中的评分变量满足高斯分布，即在给定隐藏因子的条件下可视化单元满足正态分布而不是多项分布(4.1)[8, 88]，它对应的数学表达式为:

$$P(\boldsymbol{R}_{u,m} = s|\mathbf{H}) = \frac{1}{\sqrt{2\pi}} \exp\left\{ -\frac{1}{2} \left(s - b_m - \sum_{k=1}^{K} h_{uk} w_{mk} \right)^2 \right\} \quad , \tag{4.28}$$

$$P(\boldsymbol{H}_{uk} = 1|\mathbf{R}) = \sigma\left(b_k + \sum_{m \in \mathcal{P}_u} r_{u,m} w_{mk} \right) \quad 。 \tag{4.29}$$

在上面新的假设下，GVRBM 对应的评分 $\mathbf{R}$ 的边缘分布仍然具有形式(4.5)，但其中的能量变为

$$E(\mathbf{r}_u, \mathbf{h}_u) = -\sum_{m \in \mathcal{P}_u} \sum_{k=1}^{K} w_{mk} r_{u,m} h_{uk} + \frac{1}{2} \sum_{m \in \mathcal{P}_u} (r_{u,m} - b_m)^2 - \sum_{k=1}^{K} h_{uk} b_k \quad 。 \tag{4.30}$$

类似于 Basic RBM 我们可以导出 GVRBM 中 $\log(P(\mathbf{r}_u))$ 关于模型参数 w_{mk}、b_m 和 b_k 的偏导数近似为

$$\frac{\partial \log P(\mathbf{r}_u)}{w_{mk}} \simeq <h_{uk} r_{u,m}>_{\text{data}} - <h_{uk} r_{u,m}>_{\text{T}} \quad , \tag{4.31}$$

$$\frac{\partial \log P(\mathbf{r}_u)}{b_m} \simeq <r_{u,m}>_{\text{data}} - <r_{u,m}>_{\text{T}} \quad , \tag{4.32}$$

$$\frac{\partial \log P(\mathbf{r}_u)}{b_k} \simeq <h_{uk}>_{\text{data}} - <h_{uk}>_{\text{T}} \quad 。 \tag{4.33}$$

GVRBM 可以使用类似于算法 4.1 的方法学习这些模型参数。

在获得了模型参数后，类似地我们可以使用下面的平均场更新方法获得用户 u 对电影 m 的预测评分：

$$\hat{r}_{u,m} = b_m + \sum_{k=1}^{K} \hat{p}_{uk} w_{mk} \quad ; \tag{4.34}$$

$$\hat{p}_{uk} = P(\boldsymbol{H}_{uk} = 1|\mathbf{R}) = \sigma\left(b_k + \sum_{m \in \mathcal{P}_u} r_{u,m} w_{mk}\right) \quad 。 \tag{4.35}$$

相比于 Basic RBM 和 GHRBM，GVRBM 的一个优点是它可以直接应用于其他模型的残差评分上，从而与其他模型有效地结合使用。Netflix Prize 数据集上的实验表明如果把 GVRBM 应用于 MF 模型的残差评分，它可以很大程度地改进模型的预测精度。

当隐藏单元数目 $K = 100$，学习率 $\eta = 0.0001$，惩罚参数 $\lambda_1 = 0.0005$ 及 $\lambda_2 = 0$，和动量 $\mu = 0$ 时，GVRBM 作用于 Netflix Prize 数据集的AP模型残差评分时经过 6 次迭代后获得了 0.9448 的 Probe RMSE；而它作用于去除第 $0-10$ 个 GEs（见第 2.1 节）后的残差评分并取相同超参数值时经过 5 次迭代后获得了 0.9324 的 Probe RMSE。

4.2　条件受限玻尔兹曼机（Conditional RBM）

正如前面章节（如第 3.3 节）中介绍的很多模型，隐式信息的利用同样可以改进 RBM 的预测精度。加入隐式信息后的 RBM 被称为条件受限玻尔兹曼机（*Conditional RBM*，简记为CRBM）[74]，它的图表示见图 4.2，且模型假设如下：

$$P(\boldsymbol{R}_{u,m} = s|\mathbf{H}) = \frac{\exp(b_{ms} + \sum_{k=1}^{K} h_{uk} w_{mks})}{\sum_{s'=1}^{S} \exp(b_{ms'} + \sum_{k=1}^{K} h_{uk} w_{mks'})} \quad , \tag{4.36}$$

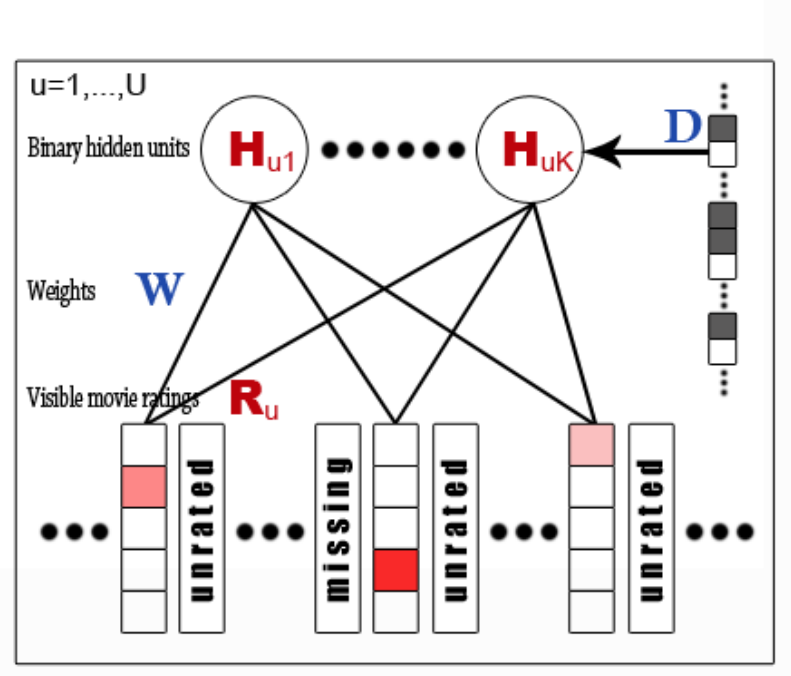

图 4.2: CRBM 的图模型表示

$$P(\boldsymbol{H}_{uk}=1|\mathbf{R})=\sigma\left(b_k+\sum_{m\in\mathcal{P}_u}w_{mkr_{u,m}}+\sum_{m\in\mathcal{A}_u}d_{mk}\right)\quad , \tag{4.37}$$

其中 $\mathbf{D}\triangleq(d_{mk})$ 为隐式信息对应的模型参数。

CRBM 对应的评分 $\mathbf{R}$ 的边缘分布仍然具有形式(4.5)，但其中的能量变为

$$E(\mathbf{r}_u,\mathbf{h}_u)=-\sum_{m\in\mathcal{P}_u}\sum_{k=1}^{K}w_{mkr_{u,m}}h_{uk}-\sum_{m\in\mathcal{P}_u}b_{mr_{u,m}}-\sum_{k=1}^{K}h_{uk}b_k-\sum_{k=1}^{K}\sum_{m\in\mathcal{A}_u}h_{uk}d_{mk}\quad 。 \tag{4.38}$$

类似于第 4.1 节中的推导，我们可以导出 CRBM 中 $\log(P(\mathbf{r}_u))$ 关于 d_{mk} 的偏导数近似为

$$\frac{\partial\log P(\mathbf{r}_u)}{d_{mk}}\simeq<h_{uk}>_{\text{data}}-<h_{uk}>_{\text{T}}\quad 。 \tag{4.39}$$

而 $\log(P(\mathbf{r}_u))$ 关于 w_{mk}、b_{ms} 和 b_k 的偏导数近似与 RBM 时有相同形式，即(4.10)、(4.11)和(4.12)。算法 4.2 给出了使用梯度上升法更新模型参数的详细过程。

算法 4.2 (CRBM) 选择隐藏单元数目 K、惩罚参数 λ、动量 μ 和学习率 η；随机初始化连接权重 $\mathbf{W}$、隐式信息参数 $\mathbf{D}$ 和偏移 b_k，并使用(4.13)初始化偏移 b_{ms}；

初始化临时变量 Δw_{mks}、 Δd_{mk}、 Δb_{ms} 和 Δb_k 为 0。

1. 对每个用户 u：

   (a) 对每部电影 $m \in \mathcal{P}_u$，利用(4.10)和(4.11) 计算相应偏导数的近似值，分别记为 δw_{mks} 和 δb_{ms}，然后使用下面的公式更新 w_{mks} 和 b_{ms}：

$$\Delta w_{mks} = \mu \cdot \Delta w_{mks} + \eta \cdot (\delta w_{mks} - \lambda \cdot w_{mks}) \quad , \tag{4.40}$$

$$w_{mks} \mathrel{+}= \Delta w_{mks} \quad ; \tag{4.41}$$

$$\Delta b_{ms} = \mu \cdot \Delta b_{ms} + \eta \cdot (\delta b_{ms} - \lambda \cdot b_{ms}) \quad , \tag{4.42}$$

$$b_{ms} \mathrel{+}= \Delta b_{ms} \quad 。 \tag{4.43}$$

   (b) 对每部电影 $m \in \mathcal{A}_u$，利用(4.39)计算相应偏导数的近似值，记为 δd_{mk}，然后使用下面的公式更新 d_{mk}：

$$\Delta d_{mk} = \mu \cdot \Delta d_{mk} + \eta \cdot (\delta d_{mk} - \lambda \cdot d_{mk}) \quad , \tag{4.44}$$

$$d_{mk} \mathrel{+}= \Delta d_{mk} \quad 。 \tag{4.45}$$

   (c) 利用(4.12)计算 $\log(P(\mathbf{r}_u))$ 关于 b_k 的近似偏导数，记为 δb_k，然后使用下面的公式更新 b_k：

$$\Delta b_k = \mu \cdot \Delta b_k + \eta \cdot (\delta b_k - \lambda \cdot b_k) \quad , \tag{4.46}$$

$$b_k \mathrel{+}= \Delta b_k \quad 。 \tag{4.47}$$

2. 使用(4.48)计算评分预测值，并计算新的 Probe RMSE。如果新的 Probe RMSE 比之前的 Probe RMSE 小，则继续前面的更新步骤；否则终止算法。

在获得了模型参数后，类似地我们可以使用下面的平均场更新方法获得用户 u 对电影 m 的预测评分：

$$\hat{r}_{u,m} = \sum_{s=1}^{S} s \cdot P(\boldsymbol{R}_{u,m} = s | \hat{\mathbf{p}}_u) \quad ; \tag{4.48}$$

$$P(\boldsymbol{R}_{u,m} = s | \hat{\mathbf{p}}_u) = \frac{\exp(b_{ms} + \sum_{k=1}^{K} \hat{p}_{uk} w_{mks})}{\sum_{s'=1}^{S} \exp(b_{ms'} + \sum_{k=1}^{K} \hat{p}_{uk} w_{mks'})} \quad , \tag{4.49}$$

$$\hat{p}_{uk} = P(\boldsymbol{H}_{uk} = 1|\mathbf{R}) = \sigma\left(b_k + \sum_{m \in \mathcal{P}_u} w_{mkr_{u,m}} + \sum_{m \in \mathcal{A}_u} d_{mk}\right) 。 \quad (4.50)$$

当隐藏单元数目 $K = 100$，学习率 $\eta = 0.002$，惩罚参数 $\lambda_1 = 0.0008$ 及 $\lambda_2 = 0$，和动量 $\mu = 0$ 时，CRBM 在 Netflix Prize 数据集上经过 9 次迭代后获得了 0.9132 的 Probe RMSE。

类似于 RBM，我们可以修改 CRBM 中的模型假设(4.37)为高斯分布而获得具有高斯隐藏单元的 CRBM（简记为 CGHRBM），或者修改 CRBM 中的模型假设(4.36)为高斯分布而获得具有高斯可视化单元的 CRBM（简记为 CGVRBM）。例如当隐藏单元数目 $K = 100$，学习率 $\eta = 0.0001$，惩罚参数 $\lambda_1 = 0.0005$ 及 $\lambda_2 = 0$，和动量 $\mu = 0$ 时，CGVRBM 作用于 Netflix Prize 数据集去除第 $0 - 10$ 个 GEs（见第 2.1 节）后的残差评分时经过 24 次迭代后获得了 0.9101 的 Probe RMSE。这些改变和 RBM 中的相应改变完全类似，我们这里不再赘述。

4.3 时间受限玻尔兹曼机（Time RBM）

对于 Netflix Prize 数据集来说，另一类可以用来改进模型预测精度的信息是评分的时间信息。很多研究者建议把时间信息加入到 RBM 中以进一步提高其预测精度[48, 69]。下面我们简单介绍其中的一些主要思想，并统称加入时间因素的 RBM 为时间受限玻尔兹曼机（*Time RBM*，简记为 TRBM）。

RBM 中的条件可视化单元假设(4.1)只使用了与电影和评分值相关的偏移 b_{ms}。类似于第 2.2 节的做法，我们加入与用户和评分时间相关的偏移 b_{us} 和 b_{uts}[48]，所以新的可视化单元模型假设为：

$$P(\boldsymbol{R}_{u,m} = s|\mathbf{H}) = \frac{\exp(b_{ut_{u,m}s} + b_{us} + b_{ms} + \sum_{k=1}^{K} h_{uk} w_{mks})}{\sum_{s'=1}^{S} \exp(b_{ut_{u,m}s'} + b_{us'} + b_{ms'} + \sum_{k=1}^{K} h_{uk} w_{mks'})} \quad , \quad (4.51)$$

其中 $t_{u,m}$ 为用户 u 给予电影 m 评分的时间（天数）。

类似于前面的推导过程，我们可以导出 TRBM 中 $\log(P(\mathbf{r}_u))$ 关于新的偏移参数 b_{uts} 和 b_{us} 的偏导数近似为[48]：

$$\frac{\partial \log P(\mathbf{r}_u)}{b_{uts}} \simeq <\delta(r_{u,m}, s)\delta(t_{u,m}, t)>_{\text{data}} - <\delta(r_{u,m}, s)\delta(t_{u,m}, t)>_{\text{T}} \quad , \quad (4.52)$$

$$\frac{\partial \log P(\mathbf{r}_u)}{b_{us}} \simeq <\delta(r_{u,m},s)>_{\text{data}} - <\delta(r_{u,m},s)>_{\text{T}} \quad 。 \tag{4.53}$$

效仿第 2.2 节中的 BP3，我们可以进一步在(4.51)中加入与评分频率有关的偏移 b_{mfs}：

$$P(\boldsymbol{R}_{u,m}=s|\mathbf{H}) = \frac{\exp(b_{mf_{u,m}s} + b_{ut_{u,m}s} + b_{us} + b_{ms} + \sum_{k=1}^{K} h_{uk}w_{mks})}{\sum_{s'=1}^{S} \exp(b_{mf_{u,m}s'} + b_{ut_{u,m}s'} + b_{us'} + b_{ms'} + \sum_{k=1}^{K} h_{uk}w_{mks'})} \quad , \tag{4.54}$$

其中 $f_{u,m}$ 为用户 u 在时间 $t_{u,m}$ 内（第 $t_{u,m}$ 天内）提供的电影评分数量。TRBM 中 $\log(P(\mathbf{r}_u))$ 关于新的偏移参数 b_{mfs} 的偏导数近似为[48]：

$$\frac{\partial \log P(\mathbf{r}_u)}{b_{mfs}} \simeq <\delta(r_{u,m},s)\delta(f_{u,m},f)>_{\text{data}} - <\delta(r_{u,m},s)\delta(f_{u,m},f)>_{\text{T}} \quad 。 \tag{4.55}$$

另一种考虑评分频率的方法是把它加入到连接权重 $\mathbf{W}=(w_{mkfs})$ 中[48, 69]：

$$w_{mkfs} = p_{mks} \cdot (1+q_{kf}) \quad 。 \tag{4.56}$$

此时 $\log(P(\mathbf{r}_u))$ 关于新的模型参数 p_{mks} 和 q_{kf} 的偏导数近似分别为[69]：

$$\frac{\partial \log P(\mathbf{r}_u)}{p_{mks}} \simeq <\delta(r_{u,m},s)(1+q_{kf})h_{uk}>_{\text{data}} - <\delta(r_{u,m},s)(1+q_{kf})h_{uk}>_{\text{T}} \quad , \tag{4.57}$$

$$\frac{\partial \log P(\mathbf{r}_u)}{q_{kl}} \simeq <h_{uk}\sum_{m\in\mathcal{P}_u} p_{mkr_{u,m}}>_{\text{data}} - <h_{uk}\sum_{m\in\mathcal{P}_u} p_{mkr_{u,m}}>_{\text{T}} \quad 。 \tag{4.58}$$

当隐藏单元数目 $K=200$ 时，TRBM 使用(4.51)在 Netflix Prize 数据集上经过 52 次迭代后获得了 0.8951 的 Probe RMSE；而使用(4.54)经过 90 次迭代后获得了 0.8928 的 Probe RMSE [48]。

4.4　因子化受限玻尔兹曼机（Factored RBM）

RBM 的一个缺点是它拥有大量的模型参数（大约 MKS 个）。对于 Netflix Prize 数据集而言，如果我们选取隐藏单元数 $K=100$，则 RBM 中包含了大约九百万个模型参数。大量的参数通常会给模型带来严重的过度拟合问题从而影响模型的预测精度。这一节我们利用类似 MF （见第五章）的降维方法来降低模型的参数数量，也即因子化原始的 $\mathbf{W}$ 为：

$$\mathbf{W}_s = \mathbf{P}_s\mathbf{Q}^T \quad (s=1,\ldots,S) \quad , \tag{4.59}$$

其中 $\mathbf{W}_s \triangleq (w_{mks}) \in \mathbb{R}^{M\times K}$，而 $\mathbf{P}_s \triangleq (p_{mls}) \in \mathbb{R}^{M\times L}$ 和 $\mathbf{Q} \triangleq (q_{kl}) \in \mathbb{R}^{K\times L}$ 为 $\mathbf{W}_s$ 对应的因子矩阵。通常 $L \ll \min\{M, K\}$，所以模型参数的数量从原来的 $O(MKS)$ 降为 $O(MLS + KL)$。我们称这种通过因子化 $\mathbf{W}$ 而获得的新模型为因子化受限玻尔兹曼机（*Factored RBM*，简记为 FRBM）[74]。

类似于前面的推导过程，我们可以导出 FRBM 中 $\log(P(\mathbf{r}_u))$ 关于因子 p_{mls} 和 q_{kl} 的偏导数近似为

$$\frac{\partial \log P(\mathbf{r}_u)}{p_{mls}} \simeq <\delta(r_{u,m}, s)\sum_{k=1}^{K} q_{kl}h_{uk}>_{\text{data}} - <\delta(r_{u,m}, s)\sum_{k=1}^{K} q_{kl}h_{uk}>_{\text{T}} \ , \tag{4.60}$$

$$\frac{\partial \log P(\mathbf{r}_u)}{q_{kl}} \simeq <h_{uk}\sum_{m\in\mathcal{P}_u} p_{mlr_{u,m}}>_{\text{data}} - <h_{uk}\sum_{m\in\mathcal{P}_u} p_{mlr_{u,m}}>_{\text{T}} \ 。 \tag{4.61}$$

FRBM 中其他模型参数的更新公式与最终的预测公式都与 RBM 中的类似，我们这里不再赘述。FRBM 应用于 Netflix Prize 数据集的预测结果可见 [74]。

上面各节我们分别讨论了如何把隐式信息、评分时间信息或降低模型参数数量方法加入到 RBM 中。这些信息可以单独加入，也可以组合加入到 RBM 中。例如，我们可以把隐式信息和评分时间信息同时加入 RBM 中得到 *Conditional Time RBM (CTRBM)*，也可以把这些信息和因子化方法同时加入 RBM 中获得 *Conditional Time Factor RBM (CTFRBM)*。对于这些新的组合 RBM，我们同样可以使用 RBM 中改变模型假设的方法获得具有高斯隐藏单元或高斯可视化单元的组合 RBM 模型。例如，对于 FRBM，我们可以像 RBM 一样改变其中的伯努利分布为高斯分布而得到具有高斯隐藏单元的 FRBM（GHFRBM），或者改变其中的多项分布为高斯分布而得到具有高斯可视化单元的 FRBM（GVFRBM）。这些改变本质上与 RBM 中的相应改变相同，所以我们这里略去对它们的详细说明。

第五章　因子模型

因子（*Factor*）模型假设每个用户的特征都可以由若干个因子进行刻画，而每部电影的特征同样可以使用相同数量的因子加以表示。当一个用户的特征和一部电影的特征相吻合时，我们就认为这个用户会对这部电影给予高的评分，反之亦然。它的一般表达形式为[6, 10, 68, 73, 84, 98]：

$$\mathbf{R} \simeq \mathbf{P}\mathbf{Q}^T \quad , \tag{5.1}$$

其中 $\mathbf{P} \in \mathbb{R}^{U\times K}$ 和 $\mathbf{Q} \in \mathbb{R}^{M\times K}$ 分别为用户和电影的潜在因子矩阵， K 为刻画每个用户或每部电影特征的因子数。一般 K 的取值远小于 U 或 M 。上式(5.1)中的因子模型通常也被称为矩阵分解（*Matrix Factorization*，简记为 *MF*）或维数降低（*Dimensionality Reduction*）模型。而由于(5.1)是数学中奇异值分解（*Singular Value Decomposition*，简记为 *SVD*）

$$\mathbf{R} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \quad （其中 \mathbf{\Sigma} 为对角矩阵） \tag{5.2}$$

的一种简化形式，(5.1)有时也被称为 SVD 模型[68]。

Simon Funk [93] 首先使用(5.1)来解决 Netflix Prize 问题。由于 MF 模型在 Netflix Prize 问题上可以获得很好的预测结果，很多研究者也探讨了 MF 更深层的概率解释[52, 72, 73]。另外，很多研究者也建议了众多推广原始 MF 模型的方法[68, 84, 96, 98]。

本章我们将介绍原始的 MF 模型及其相应的概率解释。对于离散 CF 问题，我们使用二项分布代替原始 MF 模型中评分假设的正态分布，从而获得了一种新的矩阵分解模型——*Binomial Matrix Factorization*（*BMF*）。为了同时获得聚类模型 *Fuzzy C-means (FCM)* 的可解释性以及 MF 的精确预测性，我们组合了 FCM 和 MF 的想法，提出了一种新的聚类模型——*Modified Fuzzy C-means*（*MFCM*）。最后我们详细比较了这些新模型与原始 MF 模型在 Netflix Prize 数据集上所获得的实验结果。

5.1 矩阵分解（Matrix Factorization）模型

5.1.1 Basic MF

为了克服 CF 问题中已知评分的稀疏性，降低模型的过度拟合，我们通过最小化如下的目标函数获得最终的模型参数值 $\mathbf{P} = (\mathbf{p}_1, \ldots, \mathbf{p}_U)^T$ 和 $\mathbf{Q} = (\mathbf{q}_1, \ldots, \mathbf{q}_M)^T$：

$$F(\mathbf{P}, \mathbf{Q}) = \frac{1}{2} \sum_{(u,m)\in\mathcal{P}} \left\{ (r_{u,m} - \mathbf{p}_u^T \mathbf{q}_m)^2 + \lambda (\| \mathbf{p}_u \|_2^2 + \| \mathbf{q}_m \|_2^2) \right\} \quad , \tag{5.3}$$

其中 λ 为给定的惩罚参数。

通常梯度下降法[68, 84]或交替最小二乘法[6, 101] 可以被用来最小化(5.3)。对于 Netflix Prize 问题，梯度下降法可以在更短的训练时间内获得更高的预测精度，所以下面我们主要使用梯度下降法进行求解。但使用交替最小二乘法求解获得的预测结果可以在最终的模型组合中改进预测精度，所以很多参赛团队在最后的模型组合中还是包含了这种方法。

使用梯度下降法最小化(5.3)的具体步骤见算法 5.1。在获得了用户和电影因子矩阵 $\mathbf{P}$ 和 $\mathbf{Q}$ 后，我们使用下式获得用户 u 对电影 m 的预测评分：

$$\hat{r}_{u,m} = \mathbf{p}_u^T \mathbf{q}_m = \sum_{k=1}^{K} p_{uk} \cdot q_{mk} \quad 。 \tag{5.4}$$

算法 5.1 (MF) 选择潜在因子数 K、惩罚参数 λ 和学习率 η；初始化模型参数 $\mathbf{P}$ 和 $\mathbf{Q}$（例如从均匀或正态分布中随机抽取出这些值）。

1. 对每个用户-电影评分对 $(u, m) \in \mathcal{P}$：

   (a) 计算评分残差 $e_{u,m} = r_{u,m} - \hat{r}_{u,m}$，其中 $\hat{r}_{u,m}$ 利用(5.4)计算得到。

   (b) 更新用户 u 和电影 m 的因子向量 $\mathbf{p}_u$ 和 $\mathbf{q}_m$：

   $$p_{uk} \mathrel{+}= \eta \cdot (e_{u,m} \cdot q_{mk} - \lambda \cdot p_{uk}) \quad , \tag{5.5}$$

   $$q_{mk} \mathrel{+}= \eta \cdot (e_{u,m} \cdot p_{uk} - \lambda \cdot q_{mk}) \quad 。 \tag{5.6}$$

2. 计算新的 Probe RMSE。如果新的 Probe RMSE 比之前的 Probe RMSE 小，则继续前面的更新步骤；否则终止算法。

对于 Netflix Prize 问题，一般学习率 η 可以取为 0.001，惩罚参数 λ 可以取为 0.005。

5.1.2 Biased MF

为了改进 MF 模型的预测精度，Paterek [68] 建议为每个用户和每部电影分别引入一个评分偏差，那么原始的预测评分公式(5.4)就变为

$$\hat{r}_{u,m} = b_u + b_m + \mathbf{p}_u^T \mathbf{q}_m = b_u + b_m + \sum_{k=1}^{K} p_{uk} \cdot q_{mk} \quad 。 \tag{5.7}$$

我们称此加入偏差后的 MF 模型为 *BiasedMF*，其对应的最小化目标函数为

$$\begin{aligned} G(\mathbf{P}, \mathbf{Q}, b_*) = & \frac{1}{2} \sum_{(u,m)\in\mathcal{P}} \big\{ (r_{u,m} - b_u - b_m - \mathbf{p}_u^T \mathbf{q}_m)^2 \\ & + \lambda_1 (\| \mathbf{p}_u \|_2^2 + \| \mathbf{q}_m \|_2^2) + \lambda_2 (b_u + b_m - \bar{r})^2) \big\} \quad , \end{aligned} \tag{5.8}$$

其中 $\bar{r}$ 为全局评分平均值，而 λ_1 和 λ_2 为相应的惩罚参数。和(5.3)一样，我们也可以使用梯度下降法[68, 84]或交替最小二乘法[6, 101]最小化(5.8)，其使用梯度下降法求解的迭代过程见算法 5.2。

算法 5.2 (BiasedMF) 选择潜在因子数 K、惩罚参数 λ_j（$j = 1, 2$）和学习率 η；初始化模型参数 $\mathbf{P}$、$\mathbf{Q}$ 和 b_*（例如从均匀或正态分布中随机抽取出这些值）。

1. 对每个用户-电影评分对 $(u, m) \in \mathcal{P}$：

   (a) 计算评分残差 $e_{u,m} = r_{u,m} - \hat{r}_{u,m}$，其中 $\hat{r}_{u,m}$ 利用(5.7)计算得到。

   (b) 更新用户 u 和电影 m 的因子向量 $\mathbf{p}_u$ 和 $\mathbf{q}_m$：

   $$\begin{aligned} p_{uk} \ += \ & \eta \cdot (e_{u,m} \cdot q_{mk} - \lambda_1 \cdot p_{uk}) \quad , \\ q_{mk} \ += \ & \eta \cdot (e_{u,m} \cdot p_{uk} - \lambda_1 \cdot q_{mk}) \quad 。 \end{aligned}$$

   (c) 更新用户 u 和电影 m 的偏差：

   $$\begin{aligned} b_u \ += \ & \eta \cdot (e_{u,m} - \lambda_2 \cdot (b_u + b_m - \bar{r})) \quad , \\ b_m \ += \ & \eta \cdot (e_{u,m} - \lambda_2 \cdot (b_u + b_m - \bar{r})) \quad 。 \end{aligned}$$

2. 计算新的 Probe RMSE。如果新的 Probe RMSE 比之前的 Probe RMSE 小，则继续前面的更新步骤；否则终止算法。

Takács 等 [85, 86] 提出了另一种加入偏差的方法，他们称使用这种新方法加入偏差而获得的模型为 *BRISMF*。BRISMF 固定第一个电影因子的取值为 1，并固定第二个用户因子的取值为 1 。这样第一个用户因子就相当于用户偏差，而第二个电影因子就相当于电影偏差。BRISMF 的训练过程和原始 MF 的训练过程类似，只要使用算法 5.1 更新参数时不更新第一个电影和第二个用户因子即可。

5.1.3 Probabilistic MF (PMF)

PMF 是一种图模型，在特定的假设下我们可以推导出 MF 为 PMF 的简化特例[52, 73]。

PMF 假设在给定用户 u 和电影 m 的潜在因子随机向量（维数为 K ） $\boldsymbol{P}_u$ 和 $\boldsymbol{Q}_m$ 后，用户 u 对电影 m 的评分随机变量 $\boldsymbol{R}_{u,m}$ 满足均值为 $\boldsymbol{P}_u^T\boldsymbol{Q}_m$ ，方差为 η^2 的正态分布，其中 η^2 为超参数。PMF 也假设 $\boldsymbol{P}_u$ 和 $\boldsymbol{Q}_m$ 各自满足正态分布且相互独立。所以 PMF 的模型假设写成数学形式为：

$$P(\boldsymbol{R}_{u,m}|\boldsymbol{P}_u,\boldsymbol{Q}_m,\eta^2) = \mathrm{N}(\boldsymbol{R}_{u,m}|\boldsymbol{P}_u^T\boldsymbol{Q}_m,\eta^2) \quad , \tag{5.9}$$

$$P(\boldsymbol{P}_u|\mu,\sigma^2) = \mathrm{N}(\boldsymbol{P}_u|\mu,\sigma^2\mathbf{I}_d) \quad , \tag{5.10}$$

$$P(\boldsymbol{Q}_m|\theta,\gamma^2) = \mathrm{N}(\boldsymbol{Q}_m|\theta,\gamma^2\mathbf{I}_d) \quad , \tag{5.11}$$

其中 $\Theta \triangleq \{\eta^2,\mu,\sigma^2,\theta,\gamma^2\}$ 为模型的超参数，而 $\mathbf{I}_d$ 代表单位矩阵。

PMF 进一步假设以上的正态分布对于不同的 u 或 m 是相互独立的，因此整个概率模型可以表达为如下形式，其图形表示可见图 5.1 。

$$P(\boldsymbol{R}|\boldsymbol{P},\boldsymbol{Q}) = \prod_{(u,m)\in\mathcal{P}} P(\boldsymbol{R}_{u,m}|\boldsymbol{P}_u,\boldsymbol{Q}_m,\eta^2) = \prod_{(u,m)\in\mathcal{P}} \mathrm{N}(\boldsymbol{R}_{u,m}|\boldsymbol{P}_u^T\boldsymbol{Q}_m,\eta^2) \ , \tag{5.12}$$

$$P(\boldsymbol{P}|\mu,\sigma^2) = \prod_{u=1}^{U} P(\boldsymbol{P}_u|\mu,\sigma^2) = \prod_{u=1}^{U} \mathrm{N}(\boldsymbol{P}_u|\mu,\sigma^2\mathbf{I}_d) \quad , \tag{5.13}$$

$$P(\boldsymbol{Q}|\theta,\gamma^2) = \prod_{m=1}^{M} P(\boldsymbol{Q}_m|\theta,\gamma^2) = \prod_{m=1}^{M} \mathrm{N}(\boldsymbol{Q}_m|\theta,\gamma^2\mathbf{I}_d) \quad 。 \tag{5.14}$$

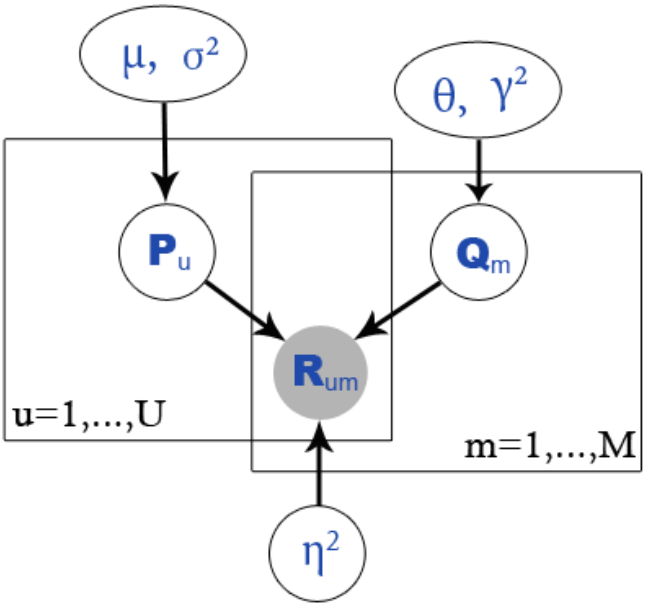

图 5.1: PMF 的图模型表示

图模型中 $\boldsymbol{P}$ 和 $\boldsymbol{Q}$ 通常被称为隐变量（*latent variables*）[5, 97]。

因此，$\mathbf{R}$，$\mathbf{P}$，$\mathbf{Q}$ 的联合分布为：

$$\begin{aligned} P(\mathbf{R},\mathbf{P},\mathbf{Q}|\Theta) &= P(\mathbf{R}|\mathbf{P},\mathbf{Q})P(\mathbf{P}|\mu,\sigma^2)P(\mathbf{Q}|\theta,\gamma^2) \\ &= \prod_{(u,m)\in\mathcal{P}} P(r_{u,m}|\mathbf{p}_u,\mathbf{q}_m,\eta^2)\cdot\prod_{u=1}^{U}P(\mathbf{p}_u|\mu,\sigma^2)\cdot\prod_{m=1}^{M}P(\mathbf{q}_m|\theta,\gamma^2) \quad 。 \end{aligned} \tag{5.15}$$

当我们假设参数 μ 和 θ 都为 0 时，最大化上面的联合分布等价于最小化 MF 的目标函数(5.3)[73]。

在给定 $\mathbf{R}$ 和 Θ 的条件下， $\mathbf{P}$ 和 $\mathbf{Q}$ 并不相互独立。它们的相互依赖性导致使用 EM 算法直接计算 $P(\mathbf{P},\mathbf{Q}|\mathbf{R},\Theta)$ 非常困难。此时一个合理的替代选择是使用变分 *EM*（VEM）算法进行计算。VEM 假设 $P(\mathbf{P},\mathbf{Q}|\mathbf{R},\Theta)$ 可以使用可完全分解的 q-分布进行近似，也即：

$$P(\mathbf{P},\mathbf{Q}|R,\Theta)\simeq Q(\mathbf{P},\mathbf{Q})=Q(\mathbf{P})Q(\mathbf{Q})=\prod_{u=1}^{U}Q(\mathbf{p}_u)\cdot\prod_{m=1}^{M}Q(\mathbf{q}_m) \quad 。 \tag{5.16}$$

所以 PMF 关于 q-分布的变分自由能可以写为：

$$\begin{aligned}
&\mathcal{F}(Q(\mathbf{P}), Q(\mathbf{Q}); \Theta) = \mathbb{E}_{\mathbf{P},\mathbf{Q}}[\log P(\mathbf{R}, \mathbf{P}, \mathbf{Q}|\Theta) - \log Q(\mathbf{P}, \mathbf{Q})] \\
&= -\frac{|\mathcal{P}|}{2}\log(2\pi\eta^2) - \mathbb{E}_{\mathbf{P},\mathbf{Q}}\Big[\sum_{(u,m)\in\mathcal{P}} \frac{(r_{u,m} - \mathbf{p}_u^T\mathbf{q}_m)^2}{2\eta^2}\Big] \\
&\quad - \frac{KU}{2}\log(2\pi\sigma^2) - \mathbb{E}_{\mathbf{P}}\Big[\sum_{u=1}^{U} \frac{(\mathbf{p}_u - \mu)^T(\mathbf{p}_u - \mu)}{2\sigma^2}\Big] \\
&\quad - \frac{KM}{2}\log(2\pi\gamma^2) - \mathbb{E}_{\mathbf{Q}}\Big[\sum_{m=1}^{M} \frac{(\mathbf{q}_m - \theta)^T(\mathbf{q}_m - \theta)}{2\gamma^2}\Big] \\
&\quad - \mathbb{E}_{\mathbf{P}}\log Q(\mathbf{P}) - \mathbb{E}_{\mathbf{Q}}\log Q(\mathbf{Q}) \quad ,
\end{aligned} \tag{5.17}$$

其中 K 为 $\mathbf{p}_u$ 的维数，而 $\mathbb{E}_{\mathbf{P}}[f(\mathbf{P})]$ 表示 $f(\mathbf{P})$ 关于 $Q(\mathbf{P})$ 的期望值（ $\mathbb{E}_{\mathbf{Q}}[f(\mathbf{Q})]$ 的意义类似）。

我们使用交替更新 $Q(\mathbf{P})$ 、 $Q(\mathbf{Q})$ 和 Θ 的方法最大化(5.17) 中的变分自由能。下面我们给出详细的更新推导过程。

为了更新分布 $Q(\mathbf{P})$ ，我们对(5.17)中的 $\mathcal{F}(Q(\mathbf{P}), Q(\mathbf{Q}); \Theta)$ 关于 $Q(\mathbf{P})$ 求导。设其导数为 0 ，并使用拉格朗日乘子法加入约束 $\int Q(\mathbf{P})\, d\mathbf{P} = 1$ ：

$$\begin{aligned}
\frac{\partial\mathcal{F}}{\partial Q(\mathbf{P})} &= -\mathbb{E}_{\mathbf{Q}}\left\{\sum_{(u,m)\in\mathcal{P}} \frac{(r_{u,m} - \mathbf{p}_u^T\mathbf{q}_m)^2}{2\eta^2}\right\} \\
&\quad - \sum_{u=1}^{U} \frac{(\mathbf{p}_u - \mu)^T(\mathbf{p}_u - \mu)}{2\sigma^2} - \log Q(\mathbf{P}) - 1 - \lambda \\
&= 0 \quad ,
\end{aligned} \tag{5.18}$$

其中 λ 为拉格朗日乘子。从上式中解出 $Q(\mathbf{P})$ ，得到：

$$Q(\mathbf{P}) \propto \prod_{u=1}^{U} \exp\left(-\frac{1}{2}(\mathbf{p}_u - \bar{\mathbf{p}}_u)^T\Phi_u^{-1}(\mathbf{p}_u - \bar{\mathbf{p}}_u)\right) \quad ; \tag{5.19}$$

$$\Phi_u = \left(\frac{1}{\eta^2}\sum_{m\in\mathcal{P}_u}(\Psi_m + \bar{\mathbf{q}}_m\bar{\mathbf{q}}_m^T) + \frac{1}{\sigma^2}\cdot\mathbf{I}_d\right)^{-1} \quad , \tag{5.20}$$

$$\bar{\mathbf{p}}_u = \Phi_u\left(\frac{1}{\eta^2}\sum_{m\in\mathcal{P}_u} r_{u,m}\bar{\mathbf{q}}_m + \frac{1}{\sigma^2}\mu\right) \quad 。 \tag{5.21}$$

类似地我们可以导出 $Q(\mathbf{Q})$ 的更新方程：

$$Q(\mathbf{Q}) \propto \prod_{m=1}^{M} \exp\left(-\frac{1}{2}(\mathbf{q}_m - \bar{\mathbf{q}}_m)^T \Psi_m^{-1}(\mathbf{q}_m - \bar{\mathbf{q}}_m)\right) \quad ; \tag{5.22}$$

$$\Psi_m = \left(\frac{1}{\eta^2}\sum_{u\in\mathcal{P}^m}(\Phi_u + \bar{\mathbf{p}}_u\bar{\mathbf{p}}_u^T) + \frac{1}{\gamma^2}\cdot\mathbf{I}_d\right)^{-1} \quad , \tag{5.23}$$

$$\bar{\mathbf{q}}_m = \Psi_m\left(\frac{1}{\eta^2}\sum_{u\in\mathcal{P}^m} r_{u,m}\bar{\mathbf{p}}_u + \frac{1}{\gamma^2}\theta\right) \quad 。 \tag{5.24}$$

下面我们导出参数 Θ 的更新方程。

$$\begin{aligned}
\frac{\partial\mathcal{F}}{\partial\mu} &= \frac{\partial}{\partial\mu}\left(-\mathbb{E}_{\mathbf{P}}\Big[\sum_{u=1}^{U}\frac{(\mathbf{p}_u-\mu)^T(\mathbf{p}_u-\mu)}{2\sigma^2}\Big]\right)\\
&= \frac{\partial}{\partial\mu}\left(-\sum_{u=1}^{U}\mathbb{E}_{\mathbf{P}}\Big[\frac{[(\mathbf{p}_u-\bar{\mathbf{p}}_u)+(\bar{\mathbf{p}}_u-\mu)]^T[(\mathbf{p}_u-\bar{\mathbf{p}}_u)+(\bar{\mathbf{p}}_u-\mu)]}{2\sigma^2}\Big]\right)\\
&= \frac{\partial}{\partial\mu}\left(-\sum_{u=1}^{U}\frac{(\bar{\mathbf{p}}_u-\mu)^T(\bar{\mathbf{p}}_u-\mu)+\mathrm{trace}(\Phi_u)}{2\sigma^2}\right)\\
&= \frac{\partial}{\partial\mu}\left(-\sum_{u=1}^{U}\frac{(\bar{\mathbf{p}}_u-\mu)^T(\bar{\mathbf{p}}_u-\mu)}{2\sigma^2}\right)\\
&= -\frac{1}{2\sigma^2}\sum_{u=1}^{U}(2\mu-2\bar{\mathbf{p}}_u)\\
&= 0 \quad 。
\end{aligned} \tag{5.25}$$

因此，

$$\mu \quad = \quad \frac{1}{U}\sum_{u=1}^{U}\bar{\mathbf{p}}_u \quad 。 \tag{5.26}$$

类似地我们可以获得 θ 的更新方程：

$$\theta \quad = \quad \frac{1}{M}\sum_{m=1}^{M}\bar{\mathbf{q}}_m \quad 。 \tag{5.27}$$

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial(\sigma^2)} &= -\frac{KU}{2}\frac{1}{\sigma^2} - \frac{\partial}{\partial(\sigma^2)}\left(\sum_{u=1}^{U}\mathbb{E}_{\mathbf{P}}\frac{[(\mathbf{p}_u-\bar{\mathbf{p}}_u)+(\bar{\mathbf{p}}_u-\mu)]^T[(\mathbf{p}_u-\bar{\mathbf{p}}_u)+(\bar{\mathbf{p}}_u-\mu)]}{2\sigma^2}\right)\\ &= -\frac{KU}{2}\frac{1}{\sigma^2} - \frac{\partial}{\partial(\sigma^2)}\left(\sum_{u=1}^{U}\frac{(\bar{\mathbf{p}}_u-\mu)^T(\bar{\mathbf{p}}_u-\mu)+\mathrm{trace}(\Phi_u)}{2\sigma^2}\right)\\ &= -\frac{KU}{2}\frac{1}{\sigma^2} - \frac{\partial}{\partial(\sigma^2)}\left(\sum_{u=1}^{U}\frac{(\bar{\mathbf{p}}_u-\mu)^T(\bar{\mathbf{p}}_u-\mu)+\mathrm{trace}(\Phi_u)}{2\sigma^2}\right)\\ &= -\frac{KU}{2}\frac{1}{\sigma^2} + \frac{1}{2\sigma^4}\sum_{u=1}^{U}[(\bar{\mathbf{p}}_u-\mu)^T(\bar{\mathbf{p}}_u-\mu)+\mathrm{trace}(\Phi_u)]\\ &= 0 \quad 。\end{aligned} \tag{5.28}$$

因此，

$$\sigma^2 = \frac{1}{KU}\sum_{u=1}^{U}[(\bar{\mathbf{p}}_u-\mu)^T(\bar{\mathbf{p}}_u-\mu)+\mathrm{trace}(\Phi_u)] \quad 。 \tag{5.29}$$

类似地我们可以获得 γ^2 的更新方程：

$$\gamma^2 = \frac{1}{KM}\sum_{m=1}^{M}[(\bar{\mathbf{q}}_m-\theta)^T(\bar{\mathbf{q}}_m-\theta)+\mathrm{trace}(\Psi_m)] \quad 。 \tag{5.30}$$

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial \eta^2} &= -\frac{|\mathcal{P}|}{2\eta^2} + \frac{1}{2\eta^4}\mathbb{E}_{\mathbf{P},\mathbf{Q}}\Big[\sum_{(u,m)\in\mathcal{P}}(r_{u,m}-\mathbf{p}_u^T\mathbf{q}_m)^2\Big]\\ &= -\frac{|\mathcal{P}|}{2\eta^2} + \frac{1}{2\eta^4}\sum_{(u,m)\in\mathcal{P}}\big[r_{u,m}^2 - 2r_{u,m}\bar{\mathbf{p}}_u^T\bar{\mathbf{q}}_m + \mathrm{trace}((\Phi_u+\bar{\mathbf{p}}_u\bar{\mathbf{p}}_u^T)(\Psi_m+\bar{\mathbf{q}}_m\bar{\mathbf{q}}_m^T))\big]\\ &= 0 \quad 。\end{aligned} \tag{5.31}$$

因此，

$$\eta^2 = \frac{1}{|\mathcal{P}|}\sum_{(u,m)\in\mathcal{P}}\big[r_{u,m}^2 - 2r_{u,m}\bar{\mathbf{p}}_u^T\bar{\mathbf{q}}_m + \mathrm{trace}((\Phi_u+\bar{\mathbf{p}}_u\bar{\mathbf{p}}_u^T)(\Psi_m+\bar{\mathbf{q}}_m\bar{\mathbf{q}}_m^T))\big] \quad 。 \tag{5.32}$$

在获得了 q-分布 $Q(\mathbf{P})$ 、 $Q(\mathbf{Q})$ 和参数 Θ 后，我们使用下面的公式获得用户 u 对电影 m 的预测评分：

$$\hat{r}_{u,m} = \bar{\mathbf{p}}_u^T \bar{\mathbf{q}}_m = \sum_{k=1}^{K} \bar{\mathbf{p}}_{uk} \cdot \bar{\mathbf{q}}_{mk} \quad 。 \tag{5.33}$$

5.2 二项矩阵分解（Binomial MF）模型

我们在第 5.1.3 节介绍的 PMF（其实也包括 MF）模型假设在给定用户和电影的潜在因子后，其对应的评分随机变量满足正态分布。但对于很多实际的 CF 问题，允许的评分往往只包括若干个整数值。例如在 Netflix Prize 问题中，真正的评分值只有 $\{1,2,3,4,5\}$ 。对于这种离散的 CF 问题，PMF 使用评分满足正态分布的假设显然不合理。Marlin [55, 56] 建议使用多项分布代替正态分布假设，例如 [56] 中所建议的多项混合模型（*MMM*）就使用了评分满足多项分布的假设。

但多项分布内的取值并不具有顺序性，不同取值之间互无关联，它的分布可能是多峰的（*multimodal*）。而对于评分数值来说，它们具有内在的有序性，它的分布应该只能是单峰的（*unimodal*）。所以，多项分布的多峰性使得它并不适合用来描述评分。比多项分布更加合理的选择是二项分布，我们在附录 B 中提出的二项混合模型（*MBM*）就是使用二项分布假设代替 MMM 中的多项分布假设。对于 Netflix Prize 问题，MBM 获得了比 MMM 更高得多的预测精度。

本节我们建议使用如下的二项分布假设代替 PMF 中的正态分布假设(5.9)，而保留 PMF 中的其他两个正态分布假设(5.10)和(5.11)[96]：

$$P(\boldsymbol{R}_{u,m}|\boldsymbol{P}_u, \boldsymbol{Q}_m) = \mathrm{B}(\boldsymbol{R}_{u,m} - 1|S-1, \beta_{um}) \quad , \tag{5.34}$$

其中 $\mathrm{B}(k|n,p)$ 为具有参数 n 和 p 的二项分布函数， S 为界定允许评分范围的定值（对于 Netflix Prize 问题， $S=5$ ），而 β_{um} 为用户 u 和电影 m 的潜在因子随机向量点积的某函数值，这里我们取

$$\beta_{um} = \frac{1}{1+e^{-\boldsymbol{P}_u^T \boldsymbol{Q}_m}} \quad 。$$

我们称这个新模型为 *Binomial MF*（*BMF*）模型。

BMF 中用户 u 给予电影 m 评分 r（ $r = 1, \ldots, S$ ）的概率为：

$$P(\boldsymbol{R}_{u,m} = r | \boldsymbol{P}_u = \mathbf{p}_u, \boldsymbol{Q}_m = \mathbf{q}_m) = \binom{S-1}{r-1} \beta_{um}^{r-1} (1-\beta_{um})^{S-r} \quad 。 \tag{5.35}$$

所以 BMF 中 $\mathbf{P}$ 和 $\mathbf{Q}$ 的 log-后验分布为：

$$\begin{aligned} &\log P(\mathbf{P}, \mathbf{Q} | \mathbf{R}, \Theta) \\ &= \sum_{(u,m)\in\mathcal{P}} \left\{ \log \binom{S-1}{r_{u,m}-1} - (S-1)\log(1+e^{-\mathbf{p}_u^T \mathbf{q}_m}) - (S-r_{u,m})\mathbf{p}_u^T \mathbf{q}_m \right\} \\ &\quad - \frac{1}{2\sigma^2} \sum_{u=1}^{U} (\mathbf{p}_u - \mu)^T (\mathbf{p}_u - \mu) - \frac{1}{2\gamma^2} \sum_{m=1}^{M} (\mathbf{q}_m - \theta)^T (\mathbf{q}_m - \theta) \\ &\quad - \frac{1}{2}[KU \log \sigma^2 + KM \log \gamma^2] + \text{Const} \quad 。 \end{aligned} \tag{5.36}$$

和 PMF 一样，BMF 也可以使用梯度下降法（对应于第 5.1.1 节中的 MF）或 VEM（对应于第 5.1.3 节中的 PMF）进行求解。下面我们逐一进行介绍。

5.2.1 BMF

如果我们让模型参数 μ 和 θ 都取值为 0，并且选取方差 σ^2 和 γ^2 为合适值[73]，则最大化(5.36)等价于最小化目标函数：

$$\begin{aligned} G(\mathbf{P}, \mathbf{Q}) =& \frac{1}{2} \sum_{(u,m)\in\mathcal{P}} \left\{ (S-1)\log(1+e^{-\mathbf{p}_u^T \mathbf{q}_m}) + (S-r_{u,m})\mathbf{p}_u^T \mathbf{q}_m \right\} \\ &+ \frac{\lambda}{2} \left\{ \sum_{u=1}^{U} \| \mathbf{p}_u \|_2^2 + \sum_{m=1}^{M} \| \mathbf{q}_m \|_2^2 \right\} \quad 。 \end{aligned} \tag{5.37}$$

类似于 MF 的求解，我们使用梯度下降法最小化 $G(\mathbf{P}, \mathbf{Q})$，其具体迭代步骤见算法 5.3 [①]。由于 BMF 和 MF 的相似性，我们可以很容易地把应用于 MF 的诸多改进（如第 5.1.2 节中介绍的加入偏差的方法）或变形（如第 5.4 节中将介绍的各种变形）方法移植到 BMF 上，本文中不再赘述这些方法。

[①] 实际上算法 5.3 最小化的并不是(5.37)中的 $G(\mathbf{P}, \mathbf{Q})$，而是

$$\frac{1}{2} \sum_{(u,m)\in\mathcal{P}} \left[(S-1)\log(1+e^{-\mathbf{p}_u^T \mathbf{q}_m}) + (S-r_{u,m})\mathbf{p}_u^T \mathbf{q}_m + \lambda(\| \mathbf{p}_u \|_2^2 + \| \mathbf{q}_m \|_2^2) \right] \quad 。$$

但是本文中我们认为这种区别是可以忽略不计的。

在获得了 $\mathbf{P}$ 和 $\mathbf{Q}$ 的值后，我们使用下面的公式获得用户 u 对电影 m 的预测评分：

$$\hat{r}_{u,m} = 1 + \frac{S-1}{1+e^{-\mathbf{p}_u^T \mathbf{q}_m}} \quad 。 \tag{5.38}$$

算法 5.3 (BMF) 选择潜在因子数 K、惩罚参数 λ 和学习率 η；初始化模型参数 $\mathbf{P}$ 和 $\mathbf{Q}$（例如从均匀或正态分布中随机抽取出这些值）。

1. 对每个用户-电影评分对 $(u,m) \in \mathcal{P}$：

   (a) 计算评分残差 $e_{u,m} = r_{u,m} - \hat{r}_{u,m}$，其中 $\hat{r}_{u,m}$ 利用(5.38) 计算得到。

   (b) 更新用户 u 和电影 m 的因子向量 $\mathbf{p}_u$ 和 $\mathbf{q}_m$：

$$p_{uk} \mathrel{+}= \eta \cdot (e_{u,m} \cdot q_{mk} - \lambda \cdot p_{uk}) \quad , \tag{5.39}$$

$$q_{mk} \mathrel{+}= \eta \cdot (e_{u,m} \cdot p_{uk} - \lambda \cdot q_{mk}) \quad 。 \tag{5.40}$$

2. 计算新的 Probe RMSE。如果新的 Probe RMSE 比之前的 Probe RMSE 小，则继续前面的更新步骤；否则终止算法。

5.2.2 Probabilistic BMF (PBMF)

如果把用户和电影因子随机变量 $\boldsymbol{P}$ 和 $\boldsymbol{Q}$ 看成隐变量，我们可以类似于第 5.1.3 节中假设 $P(\mathbf{P},\mathbf{Q}|\mathbf{R},\Theta)$ 可以使用可完全分解的 q-分布(5.16)进行近似，从而使用 VEM 方法来求解(5.36)。此时 PBMF 关于 q-分布的变分自由能为：

$$\begin{aligned}
&\mathcal{F}(Q(\mathbf{P}),Q(\mathbf{Q});\Theta) = \mathbb{E}_{\mathbf{P},\mathbf{Q}}[\log P(\mathbf{R},\mathbf{P},\mathbf{Q}|\Theta) - \log Q(\mathbf{P},\mathbf{Q})] \\
&= \mathbb{E}_{\mathbf{P},\mathbf{Q}} \sum_{(u,m)\in\mathcal{P}} \left\{ \log \binom{S-1}{r_{u,m}-1} - (S-1)\log(1+e^{-\mathbf{p}_u^T\mathbf{q}_m}) - (S-r_{u,m})\mathbf{p}_u^T\mathbf{q}_m \right\} \\
&\quad - \frac{KU}{2}\log(2\pi\sigma^2) - \mathbb{E}_{\mathbf{P}}\Big[\sum_{u=1}^{U} \frac{(\mathbf{p}_u-\mu)^T(\mathbf{p}_u-\mu)}{2\sigma^2}\Big] \\
&\quad - \frac{KM}{2}\log(2\pi\gamma^2) - \mathbb{E}_{\mathbf{Q}}\Big[\sum_{m=1}^{M} \frac{(\mathbf{q}_m-\theta)^T(\mathbf{q}_m-\theta)}{2\gamma^2}\Big] \\
&\quad - \mathbb{E}_{\mathbf{P}}\log Q(\mathbf{P}) - \mathbb{E}_{\mathbf{Q}}\log Q(\mathbf{Q}) \quad ,
\end{aligned} \tag{5.41}$$

其中 K 为 $\mathbf{p}_u$ 的维数，而 $\mathbb{E}_{\mathbf{P}}[f(\mathbf{P})]$ 表示 $f(\mathbf{P})$ 关于 $Q(\mathbf{P})$ 的期望值（ $\mathbb{E}_{\mathbf{Q}}[f(\mathbf{Q})]$ 的意义类似）。

一般 VEM 使用交替更新 $Q(\mathbf{P})$ 、 $Q(\mathbf{Q})$ 和 Θ 的方法最大化变分自由能。但(5.41)中的 logistic 函数 $\log(1+e^{-x})$ 使得直接更新所产生的 $Q(\mathbf{P})$ 和 $Q(\mathbf{Q})$ 具有极其复杂的表达形式，而不再像 PMF 中满足正态分布，因此我们不能直接使用 VEM 来最大化(5.41)。一种自然的想法是寻找一个形式更加简单的数学函数近似 $\log(1+e^{-x})$ ，且由这个新函数所产生的变分自由能必须是(5.41)中自由能的下界[41]。

幸运的是 Jaakkola & Jordan [41] 已经找到了满足上面条件的近似函数。他们证明了

$$-\log(1+e^{-x}) \geq -\log(1+e^{-\xi}) + \frac{1}{2}(x-\xi) - \lambda(\xi)(x^2-\xi^2) \quad , \tag{5.42}$$

其中 ξ 为固定常数，而

$$\lambda(\xi) \triangleq \frac{1}{4\xi}\tanh(\xi/2) = \frac{1}{4\xi}\frac{1-e^{-\xi}}{1+e^{-\xi}} \quad 。$$

当 $x=\xi$ 时，不等式(5.42)变成了等式。

把(5.42)代入(5.17)，我们获得了 $\mathcal{F}(Q(\mathbf{P}),Q(\mathbf{Q});\Theta)$ 的一个下界：

$$\begin{aligned}
&\tilde{\mathcal{F}}(Q(\mathbf{P}),Q(\mathbf{Q});\Theta,\Xi) \\
&= \mathbb{E}_{\mathbf{P},\mathbf{Q}} \sum_{(u,m)\in\mathcal{P}} \left\{ \log\binom{S-1}{r_{u,m}-1} - (S-1)\log(1+e^{-\xi_{um}}) + \left(r_{u,m} - \frac{S+1}{2}\right)\mathbf{p}_u^T\mathbf{q}_m \right. \\
&\quad \left. -(S-1)\left[\lambda(\xi_{um})(\mathbf{p}_u^T\mathbf{q}_m)^2 + \frac{1}{2}\xi_{um} - \lambda(\xi_{um})\xi_{um}^2\right] \right\} \\
&\quad - \frac{KU}{2}\log(2\pi\sigma^2) - \mathbb{E}_{\mathbf{P}}\left[\sum_{u=1}^{U}\frac{(\mathbf{p}_u-\mu)^T(\mathbf{p}_u-\mu)}{2\sigma^2}\right] \\
&\quad - \frac{KM}{2}\log(2\pi\gamma^2) - \mathbb{E}_{\mathbf{Q}}\left[\sum_{m=1}^{M}\frac{(\mathbf{q}_m-\theta)^T(\mathbf{q}_m-\theta)}{2\gamma^2}\right] \\
&\quad - \mathbb{E}_{\mathbf{P}}\log Q(\mathbf{P}) - \mathbb{E}_{\mathbf{Q}}\log Q(\mathbf{Q}) \quad ,
\end{aligned} \tag{5.43}$$

其中 Ξ 为所有 ξ_{um} 组成的集合。标准的 VEM 可以被用来最大化 $\tilde{\mathcal{F}}$。例如，为了更新 $Q(\mathbf{P})$，我们对 $\tilde{\mathcal{F}}$ 关于 $Q(\mathbf{P})$ 求导。设其导数为 0，并使用拉格朗日乘子法考虑归一化约束条件 $\int Q(\mathbf{P})d\mathbf{P}=1$，最终我们获得：

$$Q(\mathbf{P}) \propto \prod_{u=1}^{U} \exp\left(-\frac{1}{2}(\mathbf{p}_u-\bar{\mathbf{p}}_u)^T\Phi_u^{-1}(\mathbf{p}_u-\bar{bp}_u)\right) \quad ; \tag{5.44}$$

$$\Phi_u = \left(2(S-1)\sum_{m\in\mathcal{P}_u}\lambda(\xi_{um})(\Psi_m+\bar{\mathbf{q}}_m\bar{\mathbf{q}}_m^T)+\frac{1}{\sigma^2}\mathbf{I}_d\right)^{-1} \quad , \tag{5.45}$$

$$\bar{\mathbf{p}}_u = \Phi_u\left(\sum_{m\in\mathcal{P}_u}\left(r_{u,m}-\frac{S+1}{2}\right)\bar{\mathbf{q}}_m+\frac{1}{\sigma^2}\mu\right) \quad 。 \tag{5.46}$$

类似地我们可以获得 $Q(\mathbf{Q})$ 的更新方程：

$$Q(\mathbf{Q}) \propto \prod_{m=1}^{M} \exp\left(-\frac{1}{2}(\mathbf{q}_m-\bar{\mathbf{q}}_m)^T\Psi_m^{-1}(\mathbf{q}_m-\bar{\mathbf{q}}_m)\right) \quad ; \tag{5.47}$$

$$\Psi_m = \left(2(S-1)\sum_{u\in\mathcal{P}^m}\lambda(\xi_{um})(\Phi_u+\bar{\mathbf{p}}_u\bar{\mathbf{p}}_u^T)+\frac{1}{\gamma^2}\mathbf{I}_d\right)^{-1} \quad , \tag{5.48}$$

$$\bar{\mathbf{q}}_m = \Psi_m\left(\sum_{u\in\mathcal{P}^m}\left(r_{u,m}-\frac{S+1}{2}\right)\bar{\mathbf{p}}_u+\frac{1}{\gamma^2}\theta\right) \quad 。 \tag{5.49}$$

参数 Θ 和 Ξ 的更新公式可以类似得到，更详细的迭代更新步骤可见算法 5.4。

在获得了 q-分布 $Q(\mathbf{P})$、$Q(\mathbf{Q})$ 和模型参数 Θ、Ξ 后，用户 u 对电影 m 的期望预测评分为：

$$\hat{r}_{u,m} = \mathbb{E}_{\mathbf{P},\mathbf{Q}}(\boldsymbol{R}_{u,m}|\mathbf{R}) = 1+\int \frac{S-1}{1+e^{-\mathbf{p}_u^T\mathbf{q}_m}}Q(\mathbf{p}_u)Q(\mathbf{q}_m)d\mathbf{p}_u d\mathbf{q}_m \quad 。 \tag{5.50}$$

而另一种获得预测评分的简单方法是使用点近似：

$$\hat{r}_{u,m} = 1+\frac{S-1}{1+e^{-\bar{\mathbf{p}}_u^T\bar{\mathbf{q}}_m}} \quad 。 \tag{5.51}$$

由于(5.51)的简单性，我们下面都使用这种方法获得预测评分。

算法 5.4 (PBMF) 选择潜在因子数 K；初始化 Ψ_m、$\bar{\mathbf{q}}_m$（$m=1,\ldots,M$），以及模型参数 Θ，Ξ（例如从均匀或正态分布中随机抽取出这些值）。

1. 更新 q-分布 $Q(\mathbf{P})$ 和 $Q(\mathbf{Q})$：

   (a) 使用(5.45)和(5.46)更新 Φ_u 和 $\bar{\mathbf{p}}_u$（$u = 1, \ldots, U$）。

   (b) 使用(5.48)和(5.49)更新 Ψ_m 和 $\bar{\mathbf{q}}_m$（$m = 1, \ldots, M$）。

2. 更新参数 Θ 和 Ξ：

   (a) 更新 μ 和 θ：

$$\begin{cases} \mu = \dfrac{1}{U}\sum_{u=1}^{U}\bar{\mathbf{p}}_u & , \\ \theta = \dfrac{1}{M}\sum_{m=1}^{U}\bar{\mathbf{q}}_m & 。 \end{cases} \tag{5.52}$$

   (b) 更新 σ^2 和 γ^2：

$$\begin{cases} \sigma^2 = \dfrac{1}{KU}\sum_{u=1}^{U}[(\bar{\mathbf{p}}_u - \mu)^T(\bar{\mathbf{p}}_u - \mu) + \mathrm{trace}(\Phi_u)] & , \\ \gamma^2 = \dfrac{1}{KM}\sum_{m=1}^{M}[(\bar{\mathbf{q}}_m - \theta)^T(\bar{\mathbf{q}}_m - \theta) + \mathrm{trace}(\Psi_m)] & 。 \end{cases} \tag{5.53}$$

   (c) 对每个用户-电影对 $(u, m) \in \mathcal{P}$，更新 ξ_{um}：

$$\xi_{um} = \sqrt{\mathrm{trace}((\Phi_u + \bar{\mathbf{p}}_u\bar{\mathbf{p}}_u^T)(\Psi_m + \bar{\mathbf{q}}_m\bar{\mathbf{q}}_m^T))} \quad 。 \tag{5.54}$$

3. 使用(5.51)计算 Probe Set 中的预测评分，并计算新的 Probe RMSE。如果新的 Probe RMSE 比之前的 Probe RMSE 小，则继续前面的更新步骤；否则终止算法。

5.3 修正 Fuzzy C-means（Modified FCM）模型

既然本节中我们构造的新模型想法来源于 Fuzzy C-means（FCM）聚类模型，下面我们首先介绍 FCM。

5.3.1 Fuzzy C-means 聚类模型

虽然 k-means 对很多问题都很高效，但它也存在模型假设的本质缺陷。对于一个来自于现实的数据集，其数据点往往并不会确定性地属于某个类别，一个典型的例子是高斯混合（*Gaussian mixture*）模型[33, 34, 56]。高斯混合模型中的数据点一般并不确定性地属于任何类别，而是依概率属于各个类别。*Fuzzy C-means*（*FCM*）推广了确定性的 k-means 模型，它允许每个数据点以不同的概率属于不同的类别。下面我们详细介绍如何把 FCM 应用于 Netflix Prize 问题。

我们这里使用 FCM 聚类用户并最终获得预测值，所以一个用户的所有评分值形成的向量代表一个数据点。对称地 FCM 也可以被用来聚类电影然后获得预测值。假设我们把用户分为 K 类，其中 K 事先给定。记 z_{uk} 为用户 u 属于第 k 类的概率，所以 $z_{uk} \geq 0$ 且 $\sum_{k=1}^{K} z_{uk} = 1$；记 $\mathbf{c}_k$ 为第 k 类的中心向量，其长度为 M。那么 FCM 通过最小化下面的目标函数获得模型参数：

$$F(\mathbf{Z}, \mathbf{C}) = \sum_{u=1}^{U} \sum_{k=1}^{K} z_{uk}^{\alpha} \parallel \mathbf{r}_u - \mathbf{c}_k \parallel^2 \quad , \tag{5.55}$$

其中 $\mathbf{Z} \triangleq (\mathbf{z}_1, \ldots, \mathbf{z}_U)^T = (z_{uk}) \in \mathbb{R}^{U \times K}$， $\mathbf{C} \triangleq (\mathbf{c}_1, \ldots, \mathbf{c}_K) = (c_{mk}) \in \mathbb{R}^{M \times K}$，而 α 为取正值的超参数，其值可以事先给定或使用附录 C 中介绍的方法自动选取。$\parallel \cdot \parallel$ 为某种范数定义，一般取为 2-范数。对于 Netflix Prize 问题，我们取此范数为 2-范数的 2γ 次幂[52]，记为 $\parallel \cdot \parallel_\gamma$ [②]，也即(5.55) 中的范数为：

$$\parallel \mathbf{r}_u - \mathbf{c}_k \parallel_\gamma \triangleq \left(\sum_{m \in \mathcal{P}_u} (r_{u,m} - c_{mk})^2 \right)^{\gamma} \quad , \tag{5.56}$$

其中 γ 为取正值的超参数。

标准的 FCM 模型（非稀疏）可以通过先固定 $\mathbf{Z}$ 更新 $\mathbf{C}$，然后再固定 $\mathbf{C}$ 更新 $\mathbf{Z}$ 这种不断迭代的方式获得最终的模型参数 $\mathbf{Z}$ 和 $\mathbf{C}$。其具体的迭代公式如下：

1. 更新每个类的中心向量 $\mathbf{c}_k$ （$k = 1, \ldots, K$）：

$$\mathbf{c}_k = \frac{\sum_{u=1}^{U} {z_{uk}}^{\alpha} \mathbf{r}_u}{\sum_{u=1}^{U} {z_{uk}}^{\alpha}} \quad 。 \tag{5.57}$$

[②]请注意这里所说的范数 $\parallel \cdot \parallel_\gamma$ 其实已经不能满足真正范数所需满足的三角不等式，所以严格上说 $\parallel \cdot \parallel_\gamma$ 已经不再是范数，但这里我们只是取其形式，而不考虑其数学定义的严格性。

2. 更新每个用户所属各类的概率值 $\mathbf{z}_u$ （$u = 1, \ldots, U$）：

$$z_{uk} = \frac{1}{\sum_{l=1}^{K} \left(\frac{\|\mathbf{r}_u - c_k\|_\gamma}{\|r_u - c_l\|_\gamma} \right)^{\frac{2}{\alpha-1}}} , \quad (k = 1, \ldots, K) \quad 。 \tag{5.58}$$

但对于 Netflix Prize 问题而言，大部分 $\mathbf{r}_u$ 中的元素值都未给出，此时我们使用如下的公式代替(5.57)更新中心矩阵 $\mathbf{C}$：

$$c_{mk} = \frac{\sum_{u \in \mathcal{P}^m} {z_{uk}}^\alpha r_{u,m}}{\sum_{u \in \mathcal{P}^m} {z_{uk}}^\alpha} , \quad (m = 1, \ldots, M;\ k = 1, \ldots, K) \quad 。 \tag{5.59}$$

一个改进预测精度的方法是在更新 $\mathbf{Z}$ 和 $\mathbf{C}$ 的同时考虑它们更新前的取值，线性组合新计算出来的数值和之前的数值。FCM 应用于稀疏 Netflix Prize 问题的具体更新步骤见算法 5.5。

算法 5.5 (FCM) 选择用户类别数 K 及更新比例参数 δ；随机初始化概率矩阵 $\mathbf{Z}$（必须保证其概率意义）。

1. 利用(5.59)获得新的中心矩阵 $\mathbf{C}^{\text{new}}$，并使用下式获得最终更新后的 $\mathbf{C}$：

$$\mathbf{C} \leftarrow \delta \cdot \mathbf{C}^{\text{new}} + (1 - \delta) \cdot \mathbf{C} \quad 。 \tag{5.60}$$

2. 利用(5.58)获得新的概率矩阵 $\mathbf{Z}^{\text{new}}$，并使用下式获得最终更新后的 $\mathbf{Z}$：

$$\mathbf{Z} \leftarrow \delta \cdot \mathbf{Z}^{\text{new}} + (1 - \delta) \cdot \mathbf{Z} \quad 。 \tag{5.61}$$

3. 使用下面的预测公式获得 $r_{u,m}$ 的预测值

$$\hat{r}_{u,m} = \sum_{k=1}^{K} z_{uk} \cdot c_{mk} \quad , \tag{5.62}$$

并获得新的 Probe RMSE。如果新的 Probe RMSE 比之前的 Probe RMSE 小，则继续前面的更新步骤；否则终止算法。

当用户类别数 $K = 24$，超参数 $\alpha = 2$，$\gamma = 16$ 及 $\delta = 0.6$ 时，FCM 对于 Netflix Prize 问题获得了 0.9499 的 Probe RMSE。

5.3.2 修正 Fuzzy C-means（Modified FCM）模型

相比于因子模型所使用的因子解释，FCM 的模糊聚类解释往往更容易让人理解和接受，但 FCM 所使用的目标函数(5.55)似乎又让人迷惑不解。既然我们最终要使用(5.62)获得预测评分，为什么不能像因子模型那样直接最小化训练数据集上的预测误差呢？

所以，相比于目标函数(5.55)，一个更加直接且自然的目标函数是[98]：

$$\begin{aligned} H(\mathbf{Z},\mathbf{C}) =& \| \mathbf{R}-\mathbf{ZC} \|_{\mathrm{F}}^2 \\ =& \sum_{(u,m)\in\mathcal{P}} \left(r_{u,m} - \sum_{k=1}^{K} z_{uk} c_{km} \right)^2 = \sum_{(u,m)\in\mathcal{P}} \left(\sum_{k=1}^{K} z_{uk}(r_{u,m}-c_{km}) \right)^2 \quad , \end{aligned} \tag{5.63}$$

其中 $\mathbf{Z}$ 为概率矩阵，所以它满足条件：

$$\mathbf{Z1}=\mathbf{1} \quad 以及 \quad \mathbf{Z}\geq 0 \ [③] \quad 。 \tag{5.64}$$

我们称通过求解带约束优化问题

$$\min_{\mathbf{Z1}=\mathbf{1},\mathbf{Z}\geq 0} H(\mathbf{Z},\mathbf{C}) \tag{5.65}$$

而获得的新模型为 *Modified Fuzzy C-means*（*MFCM*）模型。

在获得了概率和中心矩阵 $\mathbf{Z}$、$\mathbf{C}$ 后，和 FCM 一样，我们使用

$$\hat{r}_{u,m} = \sum_{k=1}^{K} z_{uk} \cdot c_{mk} \tag{5.66}$$

获得用户 u 对电影 m 的预测评分。

值得注意的是，如果我们取(5.55)中的参数 $\alpha=2$，并使用 2-范数作为其中的范数，则(5.55)中的 $F(\mathbf{Z},\mathbf{C})$ 可以重写为

$$\sum_{(u,m)\in\mathcal{P}} \sum_{k=1}^{K} \left[z_{uk}(r_{u,m}-c_{km}) \right]^2 \quad ,$$

[③]本文中所说的一个矩阵 $\mathbf{A}\geq 0$，是指 $\mathbf{A}$ 的所有元素都大于等于 0。

其中的概率矩阵 $\mathbf{Z}$ 也要满足约束条件(5.64)。

虽然 MFCM 的优化问题(5.65)似乎和 FCM 中的优化问题相似，但实际上求解(5.65)要难得多。为了最小化 FCM 的目标函数(5.55)，我们只需要从方程

$$\frac{\partial F}{\partial z_{uk}} = 0 \ , \qquad \frac{\partial F}{\partial c_{mk}} = 0$$

中解出相应的 z_{uk} 和 c_{mk} 即可获得它们的更新公式。但这种求解方式并不适用于 MFCM 的优化问题(5.65)。

如果我们忽略(5.65)中的约束条件，则 MFCM 就和第 5.1.1 节介绍的 MF 没有区别。所以从某种角度来看我们也可以使用模糊聚类的观点来解释 MF 模型。既然 MF 可以使用梯度下降法高效求解，那么 MFCM 是否也可以呢？

最简单的处理约束的方法是惩罚法，也就是当概率矩阵 $\mathbf{Z}$ 不满足约束条件(5.64)时对其加以惩罚以使其朝着满足约束条件的方向发展。所以，为了获得(5.65)的最优解，我们最小化下面的目标函数：

$$H_1(\mathbf{Z},\mathbf{C}) = \frac{1}{2}\sum_{(u,m)\in\mathcal{P}}\Big[(r_{u,m}-\mathbf{z}_u^T\mathbf{c}_m)^2+\lambda \parallel \mathbf{c}_m \parallel_2^2 +\lambda(\parallel \mathbf{z}_u \parallel_2^2 +(\mathbf{z}_u^T\mathbf{1}-1)^2+ \parallel \mathbf{z}_{u\_} \parallel_2^2)\Big], \tag{5.67}$$

其中 $\mathbf{z}_{\_} = (z_{1\_},\ldots,z_{K\_})$，而 $z_{i\_} \triangleq \max\{0,-z_i\}$。算法 5.6 给出了最小化(5.67)的具体步骤，我们称此算法为 *MFCM1*。

算法 5.6 (MFCM1) 选择潜在因子数 K、惩罚参数 λ 和学习率 η；初始化模型参数 $\mathbf{Z}$ 和 $\mathbf{C}$（必须保证 $\mathbf{Z}$ 的概率意义）。

1. 对每个用户-电影评分对 $(u,m)\in\mathcal{P}$：

   (a) 计算评分残差 $e_{u,m} = r_{u,m} - \hat{r}_{u,m}$，其中 $\hat{r}_{u,m}$ 利用(5.66)计算得到。

   (b) 更新 $\mathbf{Z}$ 的第 u 行和 $\mathbf{C}$ 的第 m 行（$k = 1,\ldots,K$）：

$$z_{uk} \mathrel{+}= \eta\cdot\Big[e_{u,m}\cdot c_{mk} - \lambda\cdot\big[(\sum_{l=1}^{K} z_{ul} - 1) + z_{uk} - z_{uk\_}\big]\Big] \quad , \tag{5.68}$$

$$c_{mk} \mathrel{+}= \eta\cdot(e_{u,m}\cdot z_{uk} - \lambda\cdot c_{mk}) \quad 。 \tag{5.69}$$

2. 计算新的 Probe RMSE。如果新的 Probe RMSE 比之前的 Probe RMSE 小，则继续前面的更新步骤；否则终止算法。

MFCM1 在 Netflix Prize 问题上获得了优于 MF 的预测精度，但最终由 MFCM1 获得的概率矩阵 $\mathbf{Z}$ 一般不会严格满足约束条件(5.64)。另一种处理(5.65)中约束条件的方法是把约束条件融入目标函数之中：

$$\tilde{H}(\mathbf{Q},\mathbf{C}) = \sum_{(u,m)\in\mathcal{P}} \left(r_{u,m} - \frac{1}{\sum_{l=1}^{K} e^{q_{ul}}} \sum_{k=1}^{K} e^{q_{uk}} c_{mk} \right)^2 \quad , \tag{5.70}$$

此时用户 u 属于第 k 类的概率为

$$z_{uk} = \frac{e^{q_{uk}}}{\sum_{l=1}^{K} e^{q_{ul}}} \quad 。 \tag{5.71}$$

所以我们通过最小化目标函数

$$H_2(\mathbf{Q},\mathbf{C}) = \frac{1}{2} \sum_{(u,m)\in\mathcal{P}} \left(r_{u,m} - \frac{1}{\sum_{l=1}^{K} e^{q_{ul}}} \sum_{k=1}^{K} e^{q_{uk}} c_{mk} \right)^2 + \lambda(\| \mathbf{c}_m \|_2^2 + \| \mathbf{q}_u \|_2^2) \tag{5.72}$$

获得模型参数 $\mathbf{Q}$ 和 $\mathbf{C}$，然后使用(5.71)获得概率矩阵 $\mathbf{Z}$，最后使用(5.66)获得预测评分。算法 5.7 给出了使用非零动量的梯度下降法最小化(5.72) 的具体过程，我们称此算法为 *MFCM2*。

算法 5.7 (MFCM2) 选择用户类别数 K、惩罚参数 λ、动量 μ 和学习率 η；初始化模型参数 $\mathbf{Q}$ 和 $\mathbf{C}$（例如从均匀或正态分布中随机抽取出这些值）；初始化临时变量 $\Delta\mathbf{Q} = (\delta q_{uk}) \in \mathbb{R}^{U\times K}$ 和 $\Delta\mathbf{C} = (\delta c_{mk}) \in \mathbb{R}^{M\times K}$ 中的所有元素为 0。

1. 加入动量值:

   (a) 更新 $\mathbf{Q}$ 和 $\Delta\mathbf{Q}$:

$$\delta q_{uk} \times= \mu \quad ,$$

$$q_{uk} \mathrel{+}= \delta q_{uk} \quad 。$$

(b) 更新 $\mathbf{C}$ 和 $\Delta\mathbf{C}$：

$$\begin{aligned} \delta c_{mk} &\times= \mu \quad , \\ c_{mk} &+= \delta c_{mk} \quad 。 \end{aligned}$$

2. 对于每个用户 u：

(a) 利用(5.71)计算用户 u 属于第 k 类的概率 z_{uk}（$k = 1, \ldots, K$）。

(b) 对于每部电影 $m \in \mathcal{P}_u$：

i. 计算评分残差 $e_{u,m} = r_{u,m} - \hat{r}_{u,m}$，其中 $\hat{r}_{u,m}$ 利用(5.66)计算得到。

ii. 更新 $\mathbf{Q}$ 和 $\Delta\mathbf{Q}$ 的第 u 行，以及 $\mathbf{C}$ 和 $\Delta\mathbf{C}$ 的第 m 行：

$$\begin{aligned} \delta_1 &= \eta \cdot [e_{u,m} \cdot z_{uk} \cdot (c_{mk} + e_{u,m} - r_{u,m}) - \lambda \cdot q_{uk}] \quad , \\ q_{uk} &+= \delta_1 \quad , \\ \delta q_{uk} &+= \delta_1 \quad ; \\ \delta_2 &= \eta \cdot (e_{u,m} \cdot z_{uk} - \lambda \cdot c_{mk}) \quad , \\ c_{mk} &+= \delta_2 \quad , \\ \delta c_{mk} &+= \delta_2 \quad 。 \end{aligned}$$

3. 计算新的 Probe RMSE。如果新的 Probe RMSE 比之前的 Probe RMSE 小，则继续前面的更新步骤；否则终止算法。

MFCM2 在 Netflix Prize 问题上获得了略低于 MF 的预测精度，但最终由 MFCM2 获得的概率矩阵 $\mathbf{Z}$ 严格满足约束条件(5.64)。

5.4 其他的因子模型

NSVD

MF 模型假设每个用户和和每部电影都可以使用若干个因子加以描述，所以它的参数数量为 $O((U+M)K)$，其中 K 为描述一个用户或一部电影的因子数量。对于 Netflix Prize 问题，$U \approx 480,000$ 以及 $M \approx 20,000$，所以如果我们取因子

数 $K = 40$，则 MF 的模型参数数量为 $O(10^7)$。下面我们介绍两种减少 MF 模型参数数量的方法。

NSVD1

Paterek [68] 建议通过用户给予评分的电影的因子来获得用户本身的因子。具体地说，用户 u 的因子向量通过如下公式获得[86, 89]：

$$\mathbf{p}_u = \frac{1}{|\mathcal{A}_u|^{1/2}} \sum_{m \in \mathcal{A}_u} \mathbf{w}_m \quad ④ \quad , \tag{5.73}$$

其中 $\mathcal{A}_u$ 为用户 u 给予评分的所有电影集合（包括 Probe Set 中未知评分的电影），也即 $\mathcal{A}_u = \mathcal{P}_u \cup \mathcal{R}_u$。所以用户 u 对电影 m 的预测评分为：

$$\hat{r}_{u,m} = \left(\frac{1}{|\mathcal{A}_u|^{1/2}} \sum_{n \in \mathcal{A}_u} \mathbf{w}_n \right)^T \mathbf{q}_m \quad 。 \tag{5.74}$$

此模型被称为 *NSVD1*。

NSVD1 这种产生用户因子的方法使得模型参数的数量变为 $O(MK)$。对于 Netflix Prize 问题而言，如果因子数仍取 $K = 40$，此时 NSVD1 的模型参数数量为 $O(10^6)$，比 MF 中的参数数量降低了一个数量级。

算法 5.8 给出了使用梯度下降法求解 NSVD1 的详细步骤，Takács 等 [86] 也给出了加入偏差后的模型更新过程。

当因子数 $K = 500$ 时，NSVD1 应用于 Netflix Prize 问题的原始评分获得了 0.933 的 Probe RMSE [89]。

算法 5.8 (NSVD1) *选择潜在因子数 K、惩罚参数 λ 和学习率 η；初始化模型参数 $\mathbf{W}$ 和 $\mathbf{Q}$（例如从均匀或正态分布中随机抽取出这些值）。*

1. *对于每个用户 u（$u = 1, \ldots, U$）:*

   (a) *使用(5.73)计算用户 u 的因子向量 $\mathbf{p}_u$，并暂存为 $\mathbf{p}_u^{\text{old}}$。*

   (b) *对于每部电影 $m \in \mathcal{P}_u$：*

④ [68] 中给出的系数为 $1/(|\mathcal{A}_u| + 1)^{1/2}$，这里我们使用了 [86, 89] 中所使用的 $1/|\mathcal{A}_u|^{1/2}$。

i. 计算评分残差 $e_{u,m} = r_{u,m} - \hat{r}_{u,m}$，其中 $\hat{r}_{u,m} = \mathbf{p}_u^T \mathbf{q}_m$。

ii. 更新因子向量 $\mathbf{p}_u$ 和 $\mathbf{q}_m$：

$$p_{uk} \mathrel{+}= \eta \cdot (e_{u,m} \cdot q_{mk} - \lambda \cdot p_{uk}) \quad , \tag{5.75}$$

$$q_{mk} \mathrel{+}= \eta \cdot (e_{u,m} \cdot p_{uk} - \lambda \cdot q_{mk}) \quad 。 \tag{5.76}$$

(c) 对于每部电影 $m \in \mathcal{A}_u$，更新 w_{mk}：

$$w_{mk} \mathrel{+}= \frac{1}{|\mathcal{A}_u|^{1/2}} (\mathbf{p}_u - \mathbf{p}_u^{\text{old}}) \ ⑤ \quad 。 \tag{5.77}$$

2. 计算新的 Probe RMSE。如果新的 Probe RMSE 比之前的 Probe RMSE 小，则继续前面的更新步骤；否则终止算法。

NSVD2

Paterek [68] 也建议完全只使用一种电影因子，也即让(5.74)中的 $\mathbf{W} = \mathbf{Q}$。所以用户 u 对电影 m 的预测评分变为[68, 89]：

$$\hat{r}_{u,m} = \left(\frac{1}{|\mathcal{A}_u|^{1/2}} \sum_{n \in \mathcal{P}_u} \mathbf{q}_n \right)^T \mathbf{q}_m \ ⑥ \quad 。 \tag{5.78}$$

此模型被称为 *NSVD2*。我们可以使用类似于算法 5.8 的梯度下降法求解 NSVD2。

当因子数 $K = 500$ 时，NSVD2 应用于 Netflix Prize 问题的原始评分获得了 0.945 的 Probe RMSE [89]。

虽然 NSVD 模型的预测精度都远低于 MF 模型，但它们却可以在最终的模型组合中提高最终组合后的评分预测精度。

非对称（Asymmetric）因子模型

受 NSVD 模型的启发，Bell 等 [8] 建议使用如下公式获得用户 u 对电影 m 的预测评分：

$$\hat{r}_{u,m} = \sum_{k=1}^{K} \left\{ \sigma \left(b_k + \sum_{n \in \mathcal{A}_u} d_{r_{u,n}} \cdot w_{nk} \right) \cdot q_{mk} \right\} \quad , \tag{5.79}$$

⑤ $\frac{1}{|\mathcal{A}_u|^{1/2}} (\mathbf{p}_u - \mathbf{p}_u^{\text{old}}) = |\mathcal{A}_u|^{1/2} (\mathbf{p}_u - \mathbf{p}_u^{\text{old}}) / |\mathcal{A}_u|$。

⑥ [68] 中给出的形式没有系数 $1/|\mathcal{A}_u|^{1/2}$。

其中 $\sigma(\cdot)$ 为 sigmoid 函数，即

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad 。$$

此模型被称为非对称（*Asymmetric*）因子模型，它的模型参数包括因子 $\mathbf{Q} = (q_{mk})$ 及 $\mathbf{W} = (w_{mk})$、截距 $\mathbf{b} = (b_k)$ 和权重 $\mathbf{d} = (d_s)$。在考虑上式中的 $d_{r_{u,n}}$ 时，对于 $\mathcal{A}_u$ 中未给出评分的电影（也即 Probe Set 中的电影），我们设定 $r_{u,n} = 0$。所以权重 $\mathbf{d}$ 的长度为 $S+1$，对于 Netflix Prize 问题即为 6。

非对称因子模型可以使用标准的梯度下降法进行求解，关于它的更多细节请见 [8, 69]。读者也可以在这些文献中找到更多的非对称性因子模型，我们这里不再赘述。

MF 的其他变形

除了我们上面介绍的一些来源于 MF 的新模型，很多研究者也提出了众多原始 MF 的变形。其中很多修改后的模型并不会比原始 MF 获得更高的预测精度，但它们却可以在最终的模型组合中提高组合后的评分预测精度。所以对于 Netflix Prize 竞赛来说，它们都具有一定的作用。它们中的很多模型只是在原始 MF 的基础上做了微小的改动，原始 MF 所使用的算法 5.1 框架在稍作修改后对它们仍然适用。下面我们仅对它们稍作介绍[12, 85]。

Semipositive MF Semipositive MF 只是在原始 MF 的基础上限制用户因子 $\mathbf{P}$ 或电影因子 $\mathbf{Q}$ 中的其中一个为非负值。它同样可以使用算法 5.1 进行求解，但在使用(5.5)或(5.6)更新因子后需要考虑到对它们的非负性约束条件。例如，如果我们限制用户因子 $\mathbf{P}$ 非负，则在使用(5.5)更新后，强制把新的小于零的 $\mathbf{P}$ 元素值变为 0，这等价于使用如下公式更新 $\mathbf{P}$：

$$p_{uk} = \max\{0,\ p_{uk} + \eta \cdot (e_{u,m} \cdot q_{mk} - \lambda \cdot p_{uk})\} \quad 。 \tag{5.80}$$

Positive MF Positive MF 和 Semipositive MF 类似，只是它是在原始 MF 的基础上限制用户因子 $\mathbf{P}$ 和电影因子 $\mathbf{Q}$ 皆为非负值。类似地它也可以通过修改算法 5.1 中的(5.5)和(5.6)进行求解。

Momentum MF Momentum MF 只是在更新用户和电影因子时使用了带有非零动量的梯度下降法，而算法 5.1 中给出的更新公式(5.5)和(5.6) 是具有零动量的梯度下降法。所以新的更新公式为：

$$p_{uk} \mathrel{+}= \eta \cdot (e_{u,m} \cdot q_{mk} - \lambda \cdot p_{uk} + \delta \cdot \Delta p_{uk}) \quad , \tag{5.81}$$

$$q_{mk} \mathrel{+}= \eta \cdot (e_{u,m} \cdot p_{uk} - \lambda \cdot q_{mk} + \delta \cdot \Delta q_{mk}) \quad , \tag{5.82}$$

其中 Δp_{uk} 为上一次更新中 p_{uk} 的改变量（ Δq_{mk} 的意义类似），而超参数 δ 的取值为 $[0, 1]$ ，它表示动量的大小。

One-by-one MF 相比于 MF，One-by-one MF 只是使用了不同的学习因子的方法。它每次只学习一个因子参数，然后下一个因子参数通过已学习参数所获得的模型预测残差进一步获得[12]。这样每次学习一个因子参数，依次下去获得多个因子参数的取值。算法在计算预测残差时也使用了压缩技术，具体请见 [12]。

在 Netflix Prize 竞赛发布后的这三四年时间内，研究者们提出了各种各样的基于 MF 模型的新模型。虽然本章中我们介绍了其中的一些典型代表，但这只是众多相关模型中的很少一部分，更多的相关模型可见 [7, 8, 48, 69, 88, 89]。

5.5 实验结果

5.5.1 MF 模型与 BMF 模型

本节中我们将比较第 5.1 节中的 MF 模型与第 5.2 节中的 BMF 模型在 Netflix Prize 数据集上的实验结果[96]。这些实验结果都是相应算法应用于原始评分所得到的。

5.5.1.1 算法 MF 与算法 BMF

算法 MF（见算法 5.1）与 BMF（见算法 5.3）在 Netflix Prize 问题上获得的 Probe RMSE 见表 5.1。表 5.1 表明 BMF 较之 MF 获得了更低的预测误差。例如，当因子数 $K = 40$ 以及学习率 $\eta = 0.0005$ 时，BMF 经过 231 次迭代后获

算法	学习率 η	迭代步数	Probe RMSE
MF	0.004	47	0.923738
	0.002	91	0.916908
	0.001	182	0.913663
	0.0005	361	0.911919
BMF	0.004	27	0.918198
	0.002	56	0.913362
	0.001	115	0.910721
	0.0005	231	0.909483

表 5.1: 学习率 η 取不同值时算法 MF 和 BMF 在 Netflix Prize 问题上所获得的 Probe RMSE。这些结果对应的因子数 $K = 40$，且算法 MF 和 BMF 所使用的惩罚系数 λ 分别为 0.025 和 0.0015。

得了 0.9095 的 Probe RMSE，而 MF 经过 361 次迭代后获得了 0.9119 的 Probe RMSE。

正如 [84] 中所观察到的，对于算法 MF，更小的学习率通常可以产生更低的预测误差，但同时算法的收敛速度也变得更慢。表 5.1 显示这种趋势对于算法 BMF 也同样存在。解决预测精度与收敛速度这对矛盾的一种常用方法是在算法迭代开始时使用较大的学习率，之后随着迭代的继续逐渐降低学习率[34]。我们建议使用策略[96, 98]

$$\eta^{(n+1)} = \begin{cases} \eta^{(n)}/2, & \text{如果 } \delta_n/\eta^{(n)} \leq \epsilon_0 \\ \eta^{(n)}, & \text{其他} \end{cases} \tag{5.83}$$

逐渐降低学习率（其中 δ_n 表示第 n 步迭代中 Probe RMSE 的下降量）。对于 Netflix Prize 数据集，如果我们取 $\eta^{(0)} = 0.002$ 以及 $\epsilon_0 = 0.03$，则经过 77 次迭代后 BMF 的 Probe RMSE 降至 0.9098 [96]。

5.5.1.2 算法 PMF 与算法 PBMF

如果我们直接使用算法 5.4 求解 PBMF，我们需要存储所有的 Φ_u（$u = 1, \ldots, U$）和所有的 Ψ_m（$m = 1, \ldots, M$），这对于 Netflix Prize 这样大规模的数据

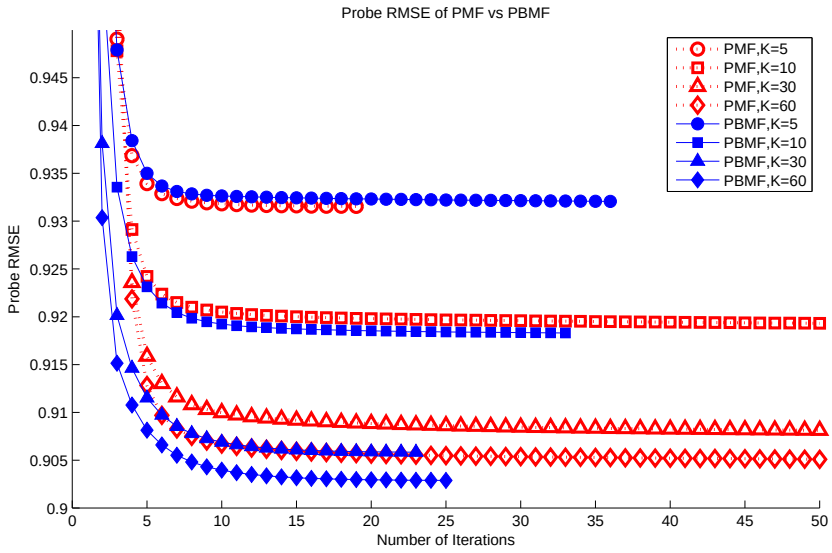

图 5.2: 因子数 K 取不同值时算法 PMF 和 PBMF 在 Netflix Prize 数据集上所获得的 Probe RMSE 随着算法迭代的变化图。伴随着 K 的增大，PBMF 较之 PMF 呈现出越来越明显的优势。当 $K = 60$ 时，PBMF 在经历了 25 次迭代后获得了 0.9029 的 Probe RMSE，而 BMF 在 59 次迭代后获得了 0.9050 的预测误差[96]。

集是不现实的。这里我们使用 [51] 中为 PMF 提出的解决方案，也即改变算法中的更新顺序。

对于 Φ_u，它只被用于更新 $\bar{\mathbf{p}}_u$、ξ_{um} 和 Ψ_m。一旦我们获得了新的 Φ_u，$\bar{\mathbf{p}}_u$ 和 ξ_{um} 就可以立即被更新。为了更新 Ψ_m，我们可以为每个 m 使用一个 $K \times K$ 的临时矩阵收集来自于不同 u 的更新结果（见(5.48)）。

使用上述方法我们可以把更新 Φ_u、$\bar{\mathbf{p}}_u$、Ψ_m 和 $\bar{\mathbf{q}}_m$ 的内存使用量降为 $O(MK^2)$，这个存储量对于 Netflix Prize 数据集而言是可以接受的。如果我们限制 Φ_u 和 Ψ_m 为特殊形式（如对角矩阵），算法 PBMF 的存储量可以进一步得到降低。但是这种简化是否可行需要我们进一步加以验证。

算法 PBMF 中另一个比较耗费内存的地方是需要存储所有的参数 Ξ，存储 Ξ 需要 $O(|\mathcal{P}|)$ 的内存量。同样地，我们可以通过限制 ξ_{um} 为特殊形式（如 $\xi_{um} = \xi_u + \xi_m$）来降低 Ξ 的存储量。

图 5.2 给出了当因子数 K 取不同值时算法 PMF 和算法 PBMF 在 Netflix Prize 数据集上所获得的预测误差。从图中可见，当 $K = 5$ 时 PMF 获得了略优于 PBMF 的预测精度，但伴随着 K 的增大，PBMF 较之 PMF 呈现出越来越明显的优势。当 $K = 30$ 时，PBMF 在经历了 23 次迭代后获得了 0.9058 的 Probe RMSE，而 BMF 在 56 次迭代后仅获得了 0.9081 的 Probe RMSE [96]。

5.5.2 MF 模型与 MFCM 模型

既然 MF 和 MFCM 模型在 Netflix Prize 数据集上获得了远高于 FCM 模型的预测精度，这节中我们只比较 MF 和 MFCM 模型在 Netflix Prize 数据集上的实验结果[98]。不同于前面实验的是，这些实验结果都是相应算法应用于 AP 模型（见第 2.2 节中的(2.12)）的残差评分所获得的预测结果。具体的实验结果可见表 5.2。表 5.2 中的结果表明算法 MFCM1 的预测精度略高于 MF，但 MFCM1 最终获得的概率矩阵并不严格满足约束条件(5.64)；而算法 MFCM2 的预测精度略低于 MF，但 MFCM2 最终获得的概率矩阵严格满足约束条件(5.64)。

算法	迭代步数	Probe RMSE
MF	37	0.920124
MFCM1	40	0.918029
MFCM2	112	0.922317

表 5.2: 算法 MF 与算法 MFCM1、MFCM2 在 Netflix Prize 问题上所获得的 Probe RMSE。所有算法所取的因子数 $K = 40$，而算法 MF 和 MFCM1 所使用的惩罚系数 $\lambda = 0.025$，算法 MFCM2 所使用的 $\lambda = 0.0002$，且其对应的动量 $\mu = 0.85$ [98]。

正如我们在第 5.5.1 节所说的，在算法 MF 和 BMF 中使用更小的学习率通常可以获得更低的预测误差，但同时算法的收敛速度也会变得更慢。表 5.3 表明这种趋势对于算法 MFCM1 和 MFCM2 也同样存在。当算法 MFCM1 中的学习率 η 从 0.004 降至 0.002，其对应的 Probe RMSE 从 0.9180 降至 0.9160，但相应的迭代步数从 40 增加至 85。为了解决预测精度与收敛速度间的这种矛盾，我们同样使用在迭代过程中逐步降低学习率的方法。

算法	学习率 η	迭代步数	Probe RMSE
MFCM1	0.004	40	0.918029
	0.002	85	0.916028
	0.001	176	0.915017
	使用(5.83)降低 η	55	0.915165
MFCM2	0.006	81	0.923233
	0.004	112	0.922317
	0.002	199	0.921644
	使用(5.84)降低 η	121	0.922183

表 5.3: 学习率 η 取不同值时算法 MFCM1 与 MFCM2 在 Netflix Prize 问题上所获得的 Probe RMSE。这些结果对应的因子数 $K = 40$，而算法 MFCM1 和 MFCM2 所使用的惩罚系数 λ 分别为 0.025 与 0.0002，且算法 MFCM2 所使用的动量 $\mu = 0.85$ [98]。对于算法 MFCM1，我们使用(5.83)逐渐降低学习率 η，其中参数 $\eta^{(0)} = 0.004$ 以及 $\epsilon_0 = 0.02$。对于算法 MFCM2，我们使用(5.84)逐渐降低学习率 η，其中参数 $\eta^{(0)} = 0.006$ 以及 $N = 80$ [98]。

如果我们使用(5.83)逐渐降低学习率 η，并取参数 $\eta^{(0)} = 0.004$ 以及 $\epsilon_0 = 0.02$，那么算法 MFCM1 在 55 步迭代后获得了 0.9152 的 Probe RMSE。如果我们使用

$$\eta^{(n+1)} = \frac{\eta^{(0)}}{1 + n/N} \tag{5.84}$$

逐渐降低学习率 η，其中参数 $\eta^{(0)} = 0.006$ 以及 $N = 80$，那么算法 MFCM2 在 121 步迭代后获得了 0.9222 的 Probe RMSE。

对于算法 MFCM2 而言，实验中我们也观察到更小的动量 μ 可以使算法获得更低的 Probe RMSE，但同时也会降低算法的收敛速度。

既然上面的算法都是应用于 AP 模型的残差评分，它们最终的预测精度自然会依赖于 AP 模型的预测精度。一种改进 AP 预测精度的方法是寻找更好的超参数值，使得通过(2.13)和(2.14)可以获得更精确的 $\bar{r}_u$ 和 $\bar{r}_m$。

另一种方法是把 $\bar{r}_u$ 和 $\bar{r}_m$ 看做算法 MFCM1 和 MFCM2 中的模型变量，在算法迭代过程中不断修正它们的值。在这种情形下两种常见的方法可以被用来更新 $\bar{r}_u$ 和 $\bar{r}^m$。第一种方法是梯度下降法：

$$\bar{r}_u \mathrel{+}= \eta[e_{u,m} - \lambda_2(\bar{r}_u + \bar{r}^m - 2\bar{r})] \quad , \tag{5.85}$$

$$\bar{r}^m \mathrel{+}= \eta[e_{u,m} - \lambda_2(\bar{r}_u + \bar{r}^m - 2\bar{r})] \quad , \tag{5.86}$$

其中 λ_2 为对应的惩罚系数。另一种方法是使用

$$\frac{\partial H_i}{\partial \bar{r}_u} = 0 \quad \text{以及} \quad \frac{\partial H_i}{\partial \bar{r}^m} = 0 \qquad (i = 1 \text{ 或 } 2)$$

直接解出 $\bar{r}_u$ 和 $\bar{r}^m$ 的更新方程。例如，算法 MFCM2 中它们的更新方程为：

$$\bar{r}_u = \frac{1}{(1+\lambda_2)|\mathcal{P}_u|}\left\{\sum_m r_{u,m} - \mathbf{z}_u \sum_m \mathbf{c}_m - (1+\lambda_2)\sum_m \bar{r}^m + (1+2\lambda_2)|\mathcal{P}_u|\bar{r}\right\} , \tag{5.87}$$

$$\bar{r}^m = \frac{1}{(1+\lambda_2)|\mathcal{P}^m|}\left\{\sum_u r_{u,m} - \mathbf{c}_m \sum_u \mathbf{z}_u - (1+\lambda_2)\sum_u \bar{r}_u + (1+2\lambda_2)|\mathcal{P}^m|\bar{r}\right\} , \tag{5.88}$$

其中 $\mathbf{z}_u$ 的定义见(5.71)。

以上介绍的这两种更新 $\bar{r}_u$ 和 $\bar{r}^m$ 的方法都能容易地合并入算法 MFCM1 和 MFCM2。如果我们使用(5.85)和(5.86)更新 MFCM1 中的 $\bar{r}_u$ 和 $\bar{r}^m$，并取 $\lambda_2 = 0.05$，则 MFCM1 的 Probe RMSE 从 0.9152 降至 0.9110。如果我们使用(5.87)和(5.88)更新 MFCM2 中的 $\bar{r}_u$ 和 $\bar{r}^m$，并取 $\lambda_2 = 0.05$，则 MFCM2 的 Probe RMSE 从 0.9222 降至 0.9201。模型中其他超参数的取值见表 5.3。

第六章　模型组合方法

我们在前面章节中介绍的模型基本只考虑了可以产生推荐的因素中的某一方面，例如邻居模型只考虑评分数据中的局部邻居作用，而因子模型只考虑数据中的全局作用[46]等等。一些研究者也建议把少量几种（通常为两种或三种）预测因素组合进单个模型以便在模型预测时考虑多方面的因素[7, 46–48]。这些单个模型虽然可以组合两三种预测因素，但实际的预测因素可能有近百种，如何把它们全都组合在一起以提高预测精度呢？

这一章我们首先介绍组合多个模型预测结果的各种方法。为了验证这些组合方法的有效性，我们实现了前面章节介绍的各种模型，并最终选出其中的 93 个模型预测结果作为组合方法的输入数据。最终我们的组合结果在 Probe Set 上获得了 0.8717 的预测误差，而提交给 Netflix Prize 的 Quiz Set 上的预测评分获得了 0.8747 的预测精度。

正如第 1.3 节中介绍的， Netflix Prize 竞赛给出了 Probe Set 上的原始评分以便参赛团队自己判断模型的预测效果，而要求团队提交 Qualifying Set 上的预测评分以便验证模型的预测精确性。下面我们要介绍的组合多个模型预测结果的方法都是基于如下的组合框架：

训练

1. 从整体训练数据集（即 WTS）中去除 Probe Set ，然后剩余的训练数据集（即 FPTS）被用来分别训练每个模型。训练出来的每个模型分别被用来获得 Probe Set 上的预测评分。假设我们考虑组合 K 个模型的预测结果，记 Probe Set 上的原始评分为 $\mathbf{b} \in \mathbb{R}^{|\mathcal{R}|}$ ，而第 k 个模型对 Probe Set 上的评分预测值为 $\mathbf{x}_k \in \mathbb{R}^{|\mathcal{R}|}$ ，并记 $\mathbf{X} \triangleq (\mathbf{x}_1, \ldots, \mathbf{x}_K)$ 。
2. 把 Probe Set 放回 FPTS，使用 WTS 重新训练上一步对应的每个模型。训练每个模型时分别使用和上一步训练过程中完全相同的参数设置（如迭代步数、惩罚参数和学习率等等）。这样训练出来的对应模型分别被用来获得 Qualifying Set 上的预测评分。我们记第 k 个模型在

Qualifying Set 上获得的预测评分为 $\mathbf{y}_k \in \mathbb{R}^{|\mathcal{Q}|}$（其中 $\mathcal{Q}$ 表示 Qualifying Set），并记 $\mathbf{Y} \triangleq (\mathbf{y}_1, \ldots, \mathbf{y}_K)$。

组合

1. 使用前面步骤获得的各个模型在 Probe Set 上的预测评分 $\mathbf{X}$ 和 Probe Set 上的原始评分 $\mathbf{b}$ 训练模型组合方法 $f(\cdot)$。
2. 把获得的组合模型 $f(\cdot)$ 应用于组合各个模型在 Qualifying Set 上的预测评分 $\mathbf{Y}$，最终获得 Qualifying Set 上的组合预测评分 $f(\mathbf{Y})$。

以下各节中我们将分别介绍不同的组合方法 $f(\cdot)$，这些组合方法应用于 Netflix Prize 数据集都可以有效地降低单个模型的评分预测误差。

6.1 线性回归（Linear Regression）模型

简单线性回归

Bell 等 [8] 建议使用简单线性回归模型组合多个模型预测结果 $\mathbf{X}$，即 $f(\mathbf{X}) = \mathbf{X}\beta$。其中的 $\beta \in \mathbb{R}^K$ 为回归系数，它可以通过最小化下面的预测误差而获得：

$$\|\mathbf{X}\beta - \mathbf{b}\|_2^2 + \lambda\|\beta\|_2^2 \quad , \tag{6.1}$$

也即 $\beta = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_d)^{-1}\mathbf{X}^T\mathbf{b}$，其中 λ 为惩罚系数，通常取 $\lambda = 0$ 即可。

在获得了 β 后，我们可以获得 Qualifying Set 上的组合预测评分 $f(\mathbf{Y}) = \mathbf{Y}\beta$。

分片线性回归

简单线性回归的一个推广是分片（binned）线性回归：首先把训练数据 $\mathbf{X}$ 按照某种规则分为若干份，然后在每份上使用简单线性回归，获得相应的回归参数；然后对应的回归系数分别被用来获得 Qualifying Set 上的组合预测评分。定义用户 u 对电影 m 的评分支持度为

$$s_{u,m} \triangleq \min\{|\mathcal{P}_u|, |\mathcal{P}^m|\} \quad 。 \tag{6.2}$$

Bell 等 [8] 建议使用评分支持度对 $\mathbf{X}$ 进行分片，例如把 $\mathbf{X}$ 分为 15 份然后分别在每份上使用简单线性回归。

精细线性回归

我们前面介绍了使用线性组合的方法获得组合预测评分，也即：

$$\hat{r}_{u,m} = \sum_{k=1}^{K} \beta_k \cdot \hat{r}_{u,m}^{(k)} \quad , \tag{6.3}$$

其中 $\hat{r}_{u,m}^{(k)}$ 表示第 k 个模型对 $r_{u,m}$ 的预测评分，而 β_k 表示第 k 个模型对应的组合系数。其中的用户-电影指标对 (u,m) 要么属于 $\mathcal{R}$，要么属于 $\mathcal{Q}$。这种简单的线性组合方法为所有的预测指标对给定相同的权重，而并不考虑各个预测模型在不同的情况下可能具有不同的预测能力。

为了克服(6.3)存在的这种问题，Bell 等 [7] 建议使用如下更加精细的方法线性组合各个模型的预测评分：

$$\hat{r}_{u,m} = \sum_{k=1}^{K} \left(\beta_k + b_i^{(k)} + c_u^{(k)} \right) \cdot \hat{r}_{u,m}^{(k)} \quad , \tag{6.4}$$

其中 β_*、$b_*^{(k)}$ 和 $c_*^{(k)}$ 为组合模型参数，它们可以使用梯度下降法最小化如下目标函数而获得：

$$\begin{aligned} & F(\beta_*, b_*^{(k)}, c_*^{(k)}) \\ & = \sum_{(u,m)\in\mathcal{R}} \left\{ \left[r_{u,m} - \sum_{k=1}^{K} \left(\beta_k + b_i^{(k)} + c_u^{(k)} \right) \cdot \hat{r}_{u,m}^{(k)} \right]^2 + \lambda \cdot \left(\beta_k^2 + {b_i^{(k)}}^2 + {c_u^{(k)}}^2 \right) \right\} \text{。} \end{aligned} \tag{6.5}$$

Bell 等 [7] 建议使用学习率 $\eta = 2\times10^{-6}$ 和惩罚系数 $\lambda = 10^{-3}$。

在获得了组合模型的参数后，我们可以使用(6.4)组合各个模型在 Qualifying Set 上的预测评分，最终获得 Qualifying Set 上的组合预测评分。

鉴于 Probe Set 中评分的稀疏性，Bell 等 [7] 也建议使用特殊形式 $c^{(k)} \cdot \hat{a}_m + d^{(k)} \cdot \hat{a}_u$ 代替预测(6.4)中的 $c_u^{(k)}$，其中 $\hat{a}_u$ 和 $\hat{a}_m$ 分别为中心化后的 $\log|\mathcal{P}_u|$ 和 $\log|\mathcal{P}^m|$。

6.2 神经网络（Neural Network）模型

Töscher & Jahrer [88] 建议使用神经网络（*neural network*）模型[34]组合多个模型的预测结果。通常神经网络包括三层单元：输入层（*input layer*）、隐藏层

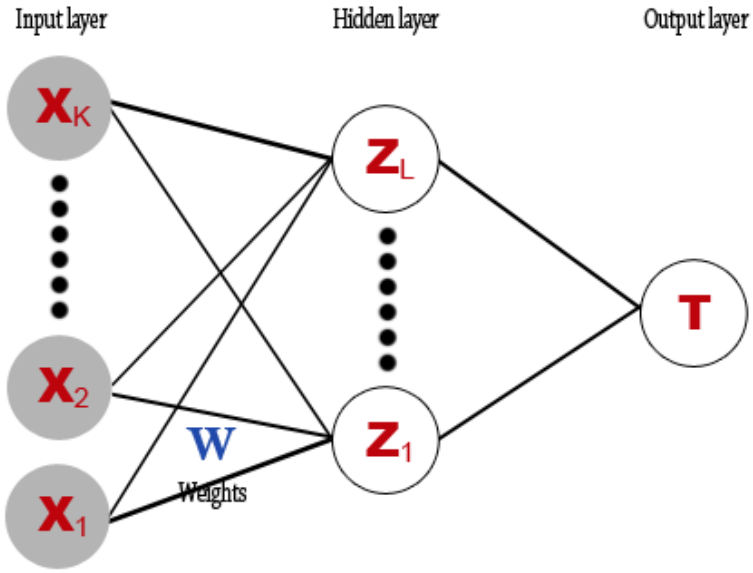

图 6.1: 具有单个输出层单元的神经网络图形表示

（*hidden layer*）和输出层（*output layer*）。当把它应用于组合多个模型的预测结果时，我们最终只需要获得组合后的预测评分，所以在输出层只需要一个神经单元，它的图形表示见图 6.1 。Töscher & Jahrer [88] 也建议使用单个隐藏层神经元，并使用 tanh(·) 作为输出函数。

神经网络的参数训练方法可见 [34]，它应用于 Netflix Prize 数据集时使用的训练参数可见 [88]。

以上两节中我们介绍了应用于 Netflix Prize 数据集的典型组合方法，一些其他的组合方法，如多项式回归（*polynomial regression*）和梯度提升决策树（*gradient boosted decision tree*）等等，在相关文献中都有详细说明[48, 69, 89]，本文不再一一赘述。

6.3 后处理（Postprocessing）方法

对于一些在训练时没有考虑时间因素的预测模型（如 BMF 模型），我们可以使用时间信息对预测评分进行调整以获得预测精度的提升[7]。下面我们简单介绍一些有效的方法。

记我们要调整的预测评分为用户-电影对 (u,m) 对应的预测评分 $\hat{r}_{u,m}$，并简记评分时间 $t_{u,m}$ 为 t。第一个调整方法是压缩 $\hat{r}_{u,m}$，使得它朝着用户 u 在第 t 天内的平均评分靠拢。为此，我们首先计算以下量：

- n_{ut} ——用户 u 在第 t 天内给予的评分数量；
- $\bar{r}_{ut}$ ——用户 u 在第 t 天内给予的评分均值；
- v_{ut} ——用户 u 在第 t 天内给予的评分方差。

在获得了上面的量之后，我们可以计算用户 u 在第 t 天内的置信系数（*confidence coefficent*）：

$$c_{ut} = n_{ut} \cdot \exp(-\alpha \cdot v_{ut}) \quad , \tag{6.6}$$

其中 α 为用户给定的超参数。之后我们使用置信系数获得预测评分 $\hat{r}_{u,m}$ 朝着评分均值压缩后的修正值：

$$\frac{\beta \cdot \hat{r}_{u,m} + c_{ut} \cdot \bar{r}_{ut}}{\beta + c_{ut}} \quad , \tag{6.7}$$

其中的超参数 β 和(6.6)中的 α 取值使用交叉验证方法加以确定，其最终获得的值为 $\alpha = 8$ 以及 $\beta = 11$ [7]。

相比于上面介绍的只计入当天评分的纠正方法，一种更强健的纠正方法是计入更长时间内的评分来获得评分纠正。记 s_{mn} 为电影 m 和 n 的相似度，这里我们取它为 Pearson 相关性（见(3.1)）的平方。相对于用户 u 而言，电影 m 和 n 的相似度定义为：

$$w^u_{mn} = s_{mn} \cdot \exp\left(-\theta |t_{u,m} - t_{u,n}|\right) \quad , \tag{6.8}$$

其中超参数 θ 取值为 0.2。和前面一样，我们首先计算以下量：

- $n_{um} = \sum_{n \in \mathcal{P}_u} w^u_{mn}$；
- $\bar{r}_{um} = n^{-1}_{um} \sum_{n \in \mathcal{P}_u} w^u_{mn} \cdot \hat{r}_{u,n}$；
- $v_{um} = n^{-1}_{um} \sum_{n \in \mathcal{P}_u} w^u_{mn} \cdot \hat{r}^2_{u,n} - \bar{r}^2_{um}$。

使用上面获得的各个值，我们可以计算相应的置信系数：

$$c_{um} = n_{um} \cdot \exp(-\alpha \cdot v_{um}) \quad , \tag{6.9}$$

其中的超参数 α 取为 5。最终我们使用置信系数获得预测评分 $\hat{r}_{u,m}$ 压缩后的修正值：

$$\frac{\hat{r}_{u,m} + c_{um} \cdot \bar{r}_{um}}{1 + c_{um}} \quad 。 \tag{6.10}$$

6.4 实验结果

使用本章开头所讲的组合框架，我们可以获得前面各章所介绍的各个模型对 Probe Set 和 Qualifying Set 中评分的预测值。我们最终选出图 6.2 和 6.3 中所列出的 93 个模型，使用第 6.1 节中介绍的简单线性回归方法组合它们的预测评分以获得最终的组合预测结果。组合结果在 Probe Set 上获得了 0.8717 的预测误差，而我们提交给 Netflix Prize 的 Quiz Set 上的预测评分获得了 0.8747 的预测误差。

对于邻居模型，我们尝试使用不同的相似度测度、邻居数目、压缩系数以及其他模型参数获得预测结果。邻居模型应用于不同基准模型的残差评分上以获得更加精确的预测评分。在进行重复性筛选后，我们在最终的组合中保留了其中的 12 个结果（见图 6.2），这些模型可以最大程度地改进最终组合的预测精度。

我们也组合了一些聚类模型的预测结果。虽然聚类模型在 Netflix Prize 问题上所获得的预测精度比邻居模型更低，但它们的加入对最终降低组合预测误差有一定的帮助。在我们的实验中，图 6.2 中给出的 7 个聚类模型结果可以使得组合预测误差降低约 0.001 。

最终组合中使用最多的模型结果来源于我们在第五章中介绍的各种因子模型。保留的 41 个模型结果来源于不同的因子模型以及同一模型具有不同的参数选取，它们包括了我们构造的 BMF 模型以及其他参赛者建议的其他因子模型。这些模型的有效性在 Netflix Prize 问题上已得到广泛认同。

受限玻尔兹曼机（RBM）模型是另一类被广泛应用于组合结果的模型。由于 RBM 模型刻画的数据关系与邻居模型和因子模型不同，它们往往可以较大程度地改进组合结果的预测精度。而且，我们在第 4.1 节中介绍的 GVRBM 可用于其

他模型的残差评分上，从而获得模型预测精度的提升。例如，我们在图 6.3 中给出的第 71 个模型就是 CGVRBM 应用于去除第 0 – 10 个 GEs 后的残差评分。

我们还把邻居模型应用于其他模型的残差评分以便综合多个模型的预测效果，我们在图 6.3 中称这样获得的模型为组合模型。例如，第 80 个模型是邻居模型应用于 NSVD1 的残差评分；第 86 个模型组合了邻居、RBM 和因子三类模型，它首先把 CGVRBM 应用于 BRISMF （见第 5.1.2 节）的残差评分，然后所得 CGVRBM 的残差评分又被应用于邻居模型。根据我们的实验，组合模型的使用往往对最终结果的预测精度影响很大。

邻居模型

1. Neighbor_numNeigh25_shrinkcoef150.000000_power2.500000_minsupp4_Pearson_MyBias2
2. Neighbor_numNeigh25_shrinkcoef150.000000_power2.000000_minsupp4_Pearson_MyBias2
3. Neighbor_numNeigh30_shrinkcoef350.000000_power2.000000_minsupp4_Pearson_Bias10
4. Neighbor_numNeigh55_shrinkcoef910.000000_power1.000000_minsupp4_PearsonV3_Raw
5. Neighbor_numNeigh55_shrinkcoef644.000000_power1.000000_minsupp4_PearsonV3_Bias10
6. Neighbor_numNeigh220_shrinkcoef450.000000_power1.000000_minsupp4_PearsonV3_Bias1
7. Neighbor_numNeigh55_shrinkcoef700.000000_power1.000000_minsupp4_PearsonV3_Bias1
8. Neighbor_numNeigh45_shrinkcoef350.000000_power2.500000_minsupp4_PearsonV3_Bias0
9. Neighbor_numNeigh45_shrinkcoef350.000000_power2.500000_minsupp4_PearsonV3_Bias2
10. Neighbor_numNeigh45_shrinkcoef350.000000_power2.500000_minsupp4_PearsonV3_Bias6
11. Neighbor_numNeigh45_shrinkcoef350.000000_power2.500000_minsupp4_PearsonV3_Bias10
12. Neighbor_numNeigh62_shrinkcoef600.000000_power2.500000_minsupp4_PearsonV3_Bias10

聚类模型

13. Clustering_FCM_K37_Alpha2.000000_Gamma16.000000_ratio0.600000
14. Clustering_FCMTrim_K37_Alpha2.000000_Gamma16.000000_ratio0.600000
15. Clustering_MMM_J20
16. Clustering_MBM_J20
17. Clustering_PMFVEM_K10
18. Clustering_PMFVEM2_K10
19. Clustering_binomialMFVEM_K20

因子模型

20. MatrixFactor_basicMF_K250_eta0.004_lambda0.025
21. MatrixFactor_basicMF_K300_eta0.005_lambda0.010_MyBias0
22. MatrixFactor_basicMF_K450_eta0.002000_lambda0.015000_beta0.030000_MyBias2_moved
23. MatrixFactor_binomialMF_K200_eta0.004000_lambda0.010000_retrainUM0
24. MatrixFactor_binomialMF_K200_eta0.004000_lambda0.010000_retrainUM1
25. MatrixFactor_MFNSVD1_K440_etau0.004000_lambdau0.010000_beta0.000000_MyBias2_moved
26. MatrixFactor_MFNSVD1_K440_etau0.024000_lambdau0.010000_beta0.000000_MyBias2_moved
27. MatrixFactor_MFNSVD1_K440_etau0.008000_lambdau0.010000_beta0.000000_MyBias2_moved
28. MatrixFactor_MFNSVD1_K440_etau0.016000_lambdau0.010000_beta0.000000_MyBias2_moved
29. MatrixFactor_NSVD1_K60_eta0.004000_lambda0.015000_MyBias0_retrainMU0
30. MatrixFactor_NSVD1_K60_eta0.004000_lambda0.015000_MyBias0_retrainMU0_S2
31. MatrixFactor_NSVD1_K60_eta0.004000_lambda0.010000_MyBias2_moved_retrainMU0
32. MatrixFactor_NSVD1_K60_eta0.004000_lambda0.010000_MyBias2_moved_retrainMU0_S2
33. MatrixFactor_NSVD1_K60_eta0.004000_lambda0.010000_MyBias2_moved_retrainMU1
34. MatrixFactor_NSVD1_K60_eta0.004000_lambda0.010000_MyBias2_moved_retrainMU1_S2
35. MatrixFactor_NSVD1_K80_eta0.005000_lambda0.010000_beta0.000000
36. MatrixFactor_NSVD1_K100_eta0.003000_lambda0.005000_beta0.005000_retrainMU0
37. MatrixFactor_NSVD1_K100_eta0.003000_lambda0.005000_beta0.005000_retrainMU0_S1
38. MatrixFactor_NSVD2_K80_eta0.000005_lambda0.010000_beta0.020000_retrainMU0_S2
39. MatrixFactor_alternateMF_K60_lambda0.055_beta0.110_MyBias2_moved
40. MatrixFactor_BRISMF_K800_etau0.000160_lambdau0.010000_retrainUM0_MyBias0
41. MatrixFactor_BRISMF_K800_etau0.016000_lambdau0.010000_retrainUM1_MyBias0
42. MatrixFactor_BRISMF_K800_etau0.000002_lambdau0.010000_retrainUM2_MyBias0
43. MatrixFactor_BRISMF_Mom_K250_etau0.010000_lambdau0.000050_mu0.300000_retrainUM0
44. MatrixFactor_BRISMF_Mom_K250_etau0.010000_lambdau0.000050_mu0.300000_retrainUM1
45. MatrixFactor_BRISMF_Mom_K250_etau0.010000_lambdau0.000050_mu0.300000_retrainUM2
46. MatrixFactor_BRISMF_Nonneg_K800_etau0.000160_lambdau0.010000_retrainUM0_MyBias0
47. MatrixFactor_BRISMF_Nonneg_K800_etau0.016000_lambdau0.010000_retrainUM1_MyBias0
48. MatrixFactor_BRISMF_Nonneg_K800_etau0.000002_lambdau0.010000_retrainUM2_MyBias0
49. MatrixFactor_BRISMF_K1000_etau0.008000_lambdau0.048000_retrainUM2_S1
50. MatrixFactor_BRISMF_Abs_TimeS_K150_etau0.005000_lambdau0.008000_retrainUM0_MyBias2
51. MatrixFactor_BRISMF_Abs_TimeS_K150_etau0.005000_lambdau0.008000_retrainUM0_MyBias2_S1
52. MatrixFactor_BRISMF_NonnegZ_TimeS_K100_etau0.005000_lambdau0.008000_retrainUM0_MyBias2
53. MatrixFactor_BRISMF_NonnegZ_TimeS_K100_etau0.005000_lambdau0.008000_retrainUM0_MyBias2_S1
54. MatrixFactor_BRISMF_TimeS_K150_etau0.008000_lambdau0.028000_retrainUM0

图 6.2: 最终组合中所使用的 93 个模型名称。组合结果在 Probe Set 上所获得的预测误差为 0.8717，而提交到 Netflix Prize 后在 Quiz Set 上所获得的预测误差为 0.8747。

因子模型（续）

55. MatrixFactor_BRISMF_TimeS_K150_etau0.008000_lambdau0.028000_retrainUM0_S2
56. MatrixFactor_LM_K1_eta0.001000_lambda0.000000_beta0.000000_retrainMU0_S2
57. MatrixFactor_AFM_K5_eta0.001000_lambda0.010000_Bias10
58. MatrixFactor_SVDXX_TimeS_K60_eta0.000200_lambda0.015000_MyBias2_S1
59. MatrixFactor_AsymmSVD_TimeS_K170_eta0.000200_lambda0.040000_MyBias2_S1
60. MatrixFactor_GlobalNgbr_TimeS_K170_eta0.005000_lambda0.002000_MyBias2

受限玻尔兹曼机模型

61. RBM_RBM_numFactor100_eta0.010000_lambda0.001000_mu0.000000
62. RBM_RBM_numFactor500_eta0.002000_lambda0.001000_mu0.000000
63. RBM_RBM_numFactor300_eta0.001000_lambda0.000800_mu0.000000
64. RBM_GVRBM_MyBias2_numFactor100_eta0.000100_lambda0.000500_mu0.000000
65. RBM_CRBM_v1h1_numFactor400_eta0.002000_lambda0.001000_mu0.000000
66. RBM_CRBM_numFactor300_eta0.002000_lambda0.001000_mu0.000000
67. RBM_GHRBM_numFactor500_eta0.000100_lambda0.000500_mu0.000000
68. RBM_CGVRBM_Bias2_numFactor100_eta0.000100_lambda0.000600_mu0.400000
69. RBM_CGVRBM_Bias6_numFactor100_eta0.000100_lambda0.000400_mu0.600000
70. RBM_CGVRBM_Bias6_numFactor300_eta0.000100_lambda0.000600_mu0.000000
71. RBM_CGVRBM_Bias10_numFactor500_eta0.000100_lambda0.000500_mu0.000000
72. RBM_CGVRBM_Bias10_numFactor600_eta0.000100_lambda0.000500_mu0.200000
73. RBM_CGVRBM_BiasLM_numFactor100_eta0.000100_lambda0.000600_mu0.600000
74. RBM_CGVRBM_BiasBRISTS_numFactor150_eta0.000100_lambda0.000400_mu0.600000
75. RBM_CGVRBM_BiasBRISNonnegZTS_numFactor100_eta0.000100_lambda0.000400_mu0.600000

组合模型

76. Neighbor_CDT_NSVD2_MyBias2_numNeigh25_aveshrnk11.000000
77. Neighbor_numNeigh80_shrinkcoef791.000000_power2.000000_minsupp4_Pearson_BiasBRISMF
78. Neighbor_numNeigh25_shrinkcoef150.000000_power2.500000_minsupp4_Pearson_MomBRIS
79. Neighbor_numNeigh25_shrinkcoef150.000000_power2.000000_minsupp4_Pearson_binomial
80. Neighbor_numNeigh25_shrinkcoef150.000000_power2.000000_minsupp4_Pearson_NSVD1
81. Neighbor_numNeigh25_shrinkcoef150.000000_power2.000000_minsupp4_Pearson_NSVD
82. Neighbor_numNeigh60_shrinkcoef650.000000_power2.500000_minsupp4_PearsonV3_Bias0NonnegZBRIS
83. Neighbor_numNeigh25_aveshrnk0.001000_shrinkcoef100.000000_power2.000000_minsupp4_Pearson_Bias10TSS_numFactor0
84. Neighbor_numNeigh25_aveshrnk0.100000_shrinkcoef100.000000_power2.000000_minsupp4_Pearson_RBMv1h1_numFactor50
85. Neighbor_numNeigh25_aveshrnk0.001000_shrinkcoef100.000000_power2.000000_minsupp4_Pearson_BiasCGVRBMmu6BiasBRISNonnegTS_numFactor100
86. Neighbor_numNeigh25_aveshrnk0.001000_shrinkcoef100.000000_power2.000000_minsupp4_Pearson_BiasCGVRBMmu6BiasBRISTS_numFactor150
87. Neighbor_numNeigh35_aveshrnk0.100000_shrinkcoef100.000000_power2.000000_minsupp4_Pearson_BiasCGVRBMmu6Bias6_numFactor100
88. Neighbor_numNeigh25_aveshrnk0.001000_shrinkcoef100.000000_power2.000000_minsupp4_Pearson_BiasCGVRBMmu6Bias2_numFactor100
89. Neighbor_numNeigh25_aveshrnk0.100000_shrinkcoef100.000000_power2.000000_minsupp4_Pearson_BiasRBM_numFactor500
90. Neighbor_numNeigh25_aveshrnk0.100000_shrinkcoef100.000000_power2.000000_minsupp4_Pearson_BiasRBM_numFactor300
91. Neighbor_numNeigh25_aveshrnk0.100000_shrinkcoef100.000000_power2.000000_minsupp4_Pearson_BiasCRBM_numFactor100
92. Neighbor_numNeigh25_aveshrnk0.100000_shrinkcoef100.000000_power2.000000_minsupp4_Pearson_BiasCRBM_numFactor300
93. SevComb_numBin5

图 6.3: （续图 6.2）最终组合中所使用的 93 个模型名称。组合结果在 Probe Set 上所获得的预测误差为 0.8717，而提交到 Netflix Prize 后在 Quiz Set 上所获得的预测误差为 0.8747。

第七章　三维协同过滤：立方填补

个性化推荐问题从数学上来讲是矩阵填补问题，也就是在给出矩阵少部分元素的情况下如何预测矩阵其他未知元素的值。通常矩阵的一行对应着一个用户，而一列则对应着一种产品或服务。一个自然的想法是我们能否把二维的矩阵填补问题扩展至三维的立方填补（*cube completion*）问题？从数学的角度来看这种扩展再自然不过，关键的问题是现实中存在立方填补的应用需求吗？

众所周知，搜索引擎（如 Google[31]、百度[4]和 Yahoo[99] 等）是用来帮助互联网用户快速查找需要的网络资源。用户通过在搜索框中输入关键词，搜索引擎依据网页与此关键词的相关程度以及网页本身的重要程度对所有网页进行排序，然后把排序后的相关网页呈现给用户。但随着互联网资源的爆炸式增长，我们现在几乎输入任何一个或几个关键词，都能找到成千上万的与之相关的网页。例如在 Google 上搜索“Peking”，我们可以获得 $10,500,000$ 个相关结果。用户在搜索“Peking”时，他（她）可能想找的是北京大学、北京景点、北京烤鸭、北京房价或者北京天气，往往不同的用户具有不同的需求。显然用户不可能把所有网页都浏览一遍以便确定自己最想要的，通常用户只会浏览搜索引擎返回的最前面的十几或几十个相关网页，以期找到自己想要的信息。用户需求的多样化与要求迅速定位所需信息之间构成了矛盾。一种解决这对矛盾的方法是针对不同的用户对相关网页给予不同的重要性，进而获得不同的搜索结果排序。这种个性化网页搜索问题中包括了三个维度：用户、查询的关键词和网页[81, 82]。我们可以根据用户过去的查询和点击等行为获得此用户在查询新关键词时的最佳网页排序。所以，个性化网页搜索问题其实就是一个立方填补问题。

除了搜索市场，互联网中另一个充满商机的地方是电子商务市场。网络的便捷性使得网民坐在家中利用电脑就可以浏览和购买各种各样的商品，既节约了用户的时间，又降低了商家的成本。对于很多的电子商务企业（如阿里巴巴[2] 旗下的电子商务企业淘宝[87]）而言，企业的大部分利润都来源于产品页面上投放的广

告。用户在点击进入某个产品介绍的页面后，一般在页面的周围都能发现企业投放的相关广告。当用户点击了某个广告，企业就可以向相应广告主收取一定的费用[①]。所以，对于电子商务企业而言，一个很重要的问题是如何提高产品页面上的广告点击率。这个广告投放问题中也包括了三个维度：用户、产品和广告。所以个性化广告投放其实也是一个立方填补问题。

综上我们可知立方填补在互联网应用中的重要性。相对于二维的协同过滤问题，三维立方填补问题更加突显了二维时的一些特点，如稀疏性和可扩展性。这也使得立方填补成为比矩阵填补更加困难的预测问题。本章中我们构造了一些求解立方填补问题的可行算法，并在构造的虚拟数据上对这些算法进行了一些检验。计算结果表明这些算法对于不太稀疏的数据集都可以获得很好的预测效果，但当数据稀疏程度加剧时，不同的模型之间将呈现出较大的预测效果差异。

本章中我们统称三个维度上的事物（如用户、产品或广告等）为对象（*object*）。记三个维度分别为 X、 Y 和 Z，与之对应的对象数量分别记为 I 、 J 和 K 。与 Netflix Prize 数据集中评分值为 $\{1,2,3,4,5\}$ 不同的是，本章我们考虑的评分数据中只包括二态值，即 0 或 1 。例如，在个性化网页搜索中，指标对 (i,j,k) 对应的值 $r_{ijk}=1$ 表示用户 i 搜索关键词 j 后点击了网页 k ， $r_{ijk}=0$ 则表示用户 i 搜索关键词 j 后没有点击网页 k 。

7.1 贝叶斯聚类（Bayesian Clustering）模型

贝叶斯聚类（*Bayesian Clustering*，简记为 BC）模型假设每个方向上的对象分别来源于不同的类别，而评分只依赖于相应对象所属的类别。它的具体模型假设如下：

$$P(\boldsymbol{R}_{ijk}|\boldsymbol{X}_i,\boldsymbol{Y}_j,\boldsymbol{Z}_k,\beta) = \mathrm{Ber}(\boldsymbol{R}_{ijk}|\beta_{\boldsymbol{X}_i\boldsymbol{Y}_j\boldsymbol{Z}_k}) \quad , \tag{7.1}$$

$$P(\boldsymbol{X}_i|\pi) = \mathrm{Mul}(\boldsymbol{X}_i|\pi) \quad , \tag{7.2}$$

$$P(\boldsymbol{Y}_j|\theta) = \mathrm{Mul}(\boldsymbol{Y}_j|\theta) \quad , \tag{7.3}$$

$$P(\boldsymbol{Z}_k|\mu) = \mathrm{Mul}(\boldsymbol{Z}_k|\mu) \quad , \tag{7.4}$$

[①]这种获利方式其实也是众多搜索引擎公司（如 Google，百度）最重要的盈利模式之一。

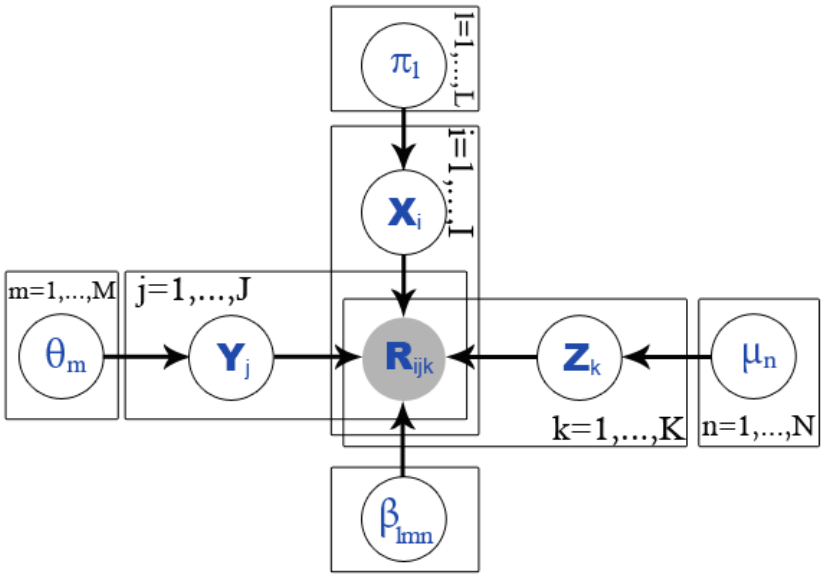

图 7.1: 贝叶斯聚类模型的图形表示

其中 $\mathrm{Ber}(\cdot|\xi)$ 和 $\mathrm{Mul}(\cdot|\zeta)$ 分别表示参数为 ξ 的伯努利分布和参数为 ζ 的多项分布，而 $\Theta \triangleq \{\beta, \pi, \theta, \mu\}$ 为模型的超参数。它的图表示见图 7.1。

BC 进一步假设上面的分布对于不同的对象是相互独立的。因此整个概率模型可以表达为如下形式，其中 $\boldsymbol{X}$ 、 $\boldsymbol{Y}$ 和 $\boldsymbol{Z}$ 通常被称为隐变量。

$$P(\boldsymbol{R}|\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z},\beta) = \prod_{(i,j,k)\in\mathcal{P}} P(\boldsymbol{R}_{ijk}|\boldsymbol{X}_i,\boldsymbol{Y}_j,\boldsymbol{Z}_k,\beta) = \prod_{(i,j,k)\in\mathcal{P}} \mathrm{Ber}(\boldsymbol{R}_{ijk}|\beta_{\boldsymbol{X}_i\boldsymbol{Y}_j\boldsymbol{Z}_k}), \quad (7.5)$$

$$P(\boldsymbol{X}|\pi) = \prod_{i=1}^{I} P(\boldsymbol{X}_i|\pi) = \prod_{i=1}^{I} \mathrm{Mul}(\boldsymbol{X}_i|\pi)\ , \quad (7.6)$$

$$P(\boldsymbol{Y}|\theta) = \prod_{j=1}^{J} P(\boldsymbol{Y}_j|\theta) = \prod_{j=1}^{J} \mathrm{Mul}(\boldsymbol{Y}_j|\theta)\ , \quad (7.7)$$

$$P(\boldsymbol{Z}|\mu) = \prod_{k=1}^{K} P(\boldsymbol{Z}_k|\mu) = \prod_{k=1}^{K} \mathrm{Mul}(\boldsymbol{Z}_k|\mu)\ 。 \quad (7.8)$$

因此，$\mathbf{R}$，$\mathbf{X}$，$\mathbf{Y}$，$\mathbf{Z}$ 的联合分布为：

$$\begin{aligned} P(\mathbf{R},\mathbf{X},\mathbf{Y},\mathbf{Z}|\Theta) &= P(\mathbf{R}|\mathbf{X},\mathbf{Y},\mathbf{Z},\beta)P(\mathbf{X}|\pi)P(\mathbf{Y}|\theta)P(\mathbf{Z}|\mu) \\ &= \prod_{(i,j,k)\in\mathcal{P}} P(\mathbf{R}_{ijk}|\mathbf{X}_i,\mathbf{Y}_j,\mathbf{Z}_k,\beta) \prod_{i=1}^{I} P(\mathbf{X}_i|\pi) \prod_{j=1}^{J} P(\mathbf{Y}_j|\theta) \prod_{k=1}^{K} P(\mathbf{Z}_k|\mu)\ 。 \end{aligned} \quad (7.9)$$

下面我们使用变分 EM（VEM）方法最大化数据似然 $P(\mathbf{R}|\Theta)$。VEM 假设 $P(\mathbf{X},\mathbf{Y},\mathbf{Z}|\mathbf{R},\Theta)$ 可以使用可完全分解的 q-分布进行近似，也即：

$$\begin{aligned} P(\mathbf{X},\mathbf{Y},\mathbf{Z}|\mathbf{R},\Theta) &\simeq Q(\mathbf{X},\mathbf{Y},\mathbf{Z}) \\ &= Q(\mathbf{X})Q(\mathbf{Y})Q(\mathbf{Z}) = \prod_{i=1}^{I} Q(\mathbf{X}_i) \prod_{j=1}^{J} Q(\mathbf{Y}_j) \prod_{k=1}^{K} Q(\mathbf{Z}_k) \quad 。 \end{aligned} \tag{7.10}$$

为简化符号，我们引入记号 $a_{il} \triangleq Q(X_i=l)$、$b_{jm} \triangleq Q(Y_j=m)$ 和 $c_{kn} \triangleq Q(Z_k=n)$。

BC 关于 q-分布(7.10)的变分自由能为：

$$\begin{aligned} \mathcal{F}(Q(\mathbf{X}),Q(\mathbf{Y}),Q(\mathbf{Z});\Theta) &= \mathbb{E}_{\mathbf{X},\mathbf{Y},\mathbf{Z}}[\log P(R,\mathbf{X},\mathbf{Y},\mathbf{Z}|\Theta) - \log Q(\mathbf{X},\mathbf{Y},\mathbf{Z})] \\ =&\mathbb{E}_{\mathbf{X},\mathbf{Y},\mathbf{Z}} \left\{ \sum_{(i,j,k)\in\mathcal{P}} \log\left[\beta_{\mathbf{X}_i\mathbf{Y}_j\mathbf{Z}_k}^{r_{ijk}}(1-\beta_{\mathbf{X}_i\mathbf{Y}_j\mathbf{Z}_k})^{1-r_{ijk}}\right] \right. \\ &\left. + \sum_{i=1}^{I} \log \pi_{\mathbf{X}_i} + \sum_{j=1}^{J} \log \theta_{\mathbf{Y}_j} + \sum_{k=1}^{K} \log \mu_{\mathbf{Z}_k} \right\} \\ &- \mathbb{E}_{\mathbf{X}} \log Q(\mathbf{X}) - \mathbb{E}_{\mathbf{Y}} \log Q(\mathbf{Y}) - \mathbb{E}_{\mathbf{Z}} \log Q(\mathbf{Z}) \ , \end{aligned} \tag{7.11}$$

其中 $\mathbb{E}_{\mathbf{X}}[f(\mathbf{X})]$ 表示 $f(\mathbf{X})$ 关于 $Q(\mathbf{X})$ 的期望值（$\mathbb{E}_{\mathbf{Y}}[f(\mathbf{Y})]$ 和 $\mathbb{E}_{\mathbf{Z}}[f(\mathbf{Z})]$ 的意义类似）。

我们使用交替更新分布 $Q(\cdot)$ 和超参数 Θ 的方法最大化(7.11) 中的变分自由能。下面我们给出详细的更新推导过程。

为了更新 $Q(\mathbf{X})$，我们对(7.11)中的 $\mathcal{F}(Q(\mathbf{X}),Q(\mathbf{Y}),Q(\mathbf{Z});\Theta)$ 关于 a_{il} 求导。设其导数为 0，并使用拉格朗日乘子法加入约束 $\sum_{l=1}^{L} a_{il} = 1$：

$$\begin{aligned} \frac{\partial \mathcal{F}}{\partial a_{il}} =& \frac{\partial}{\partial a_{il}} \left(\mathbb{E}_{X,Y,Z} \left\{ \sum_{(i,j,k)\in\mathcal{P}} \log\left[\beta_{X_iY_jZ_k}^{r_{ijk}}(1-\beta_{X_iY_jZ_k})^{1-r_{ijk}}\right] \right\} + \sum_{i=1}^{I} \log \pi_{X_i} \right) \\ &- \log a_{il} - 1 - \lambda \\ =& \sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{(j,k)\in\mathcal{P}_i} b_{jm}c_{kn}\Big[r_{ijk}\log\beta_{lmn} + (1-r_{ijk})\log(1-\beta_{lmn})\Big] + \log\pi_l \\ &- \log a_{il} - 1 - \lambda \\ =& \sum_{(j,k)\in\mathcal{P}_i} \Big[r_{ijk} <\log\beta_{lmn}>_{m,n|j,k} + (1-r_{ijk}) <\log(1-\beta_{lmn})>_{m,n|j,k}\Big] + \log\pi_l \end{aligned}$$

$$
\begin{aligned}
&- \log a_{il} - 1 - \lambda \\
&= 0 \quad ,
\end{aligned}
$$

其中 $<\log \beta_{lmn}>_{m,n|j,k} \triangleq \sum_{m=1}^{M} \sum_{n=1}^{N} b_{jm} c_{kn} \log \beta_{lmn}$，而 λ 为拉格朗日乘子。因此，

$$
a_{il} \propto \pi_l \cdot \exp\left(\sum_{(j,k)\in\mathcal{P}_i} \left[r_{ijk} <\log \beta_{lmn}>_{m,n|j,k} + (1 - r_{ijk}) <\log(1 - \beta_{lmn})>_{m,n|j,k} \right] \right) 。 \tag{7.12}
$$

类似地，我们可以导出 b_{jm} 和 c_{kn} 的更新方程：

$$
b_{jm} \propto \theta_m \cdot \exp\left(\sum_{(k,i)\in\mathcal{P}_j} \left[r_{ijk} <\log \beta_{lmn}>_{n,l|k,i} + (1 - r_{ijk}) <\log(1 - \beta_{lmn})>_{n,l|k,i} \right] \right) , \tag{7.13}
$$

$$
c_{kn} \propto \mu_n \cdot \exp\left(\sum_{(i,j)\in\mathcal{P}_k} \left[r_{ijk} <\log \beta_{lmn}>_{l,m|i,j} + (1 - r_{ijk}) <\log(1 - \beta_{lmn})>_{l,m|i,j} \right] \right) 。 \tag{7.14}
$$

下面我们导出参数 Θ 的更新方程。

$$
\begin{aligned}
\frac{\partial \mathcal{F}}{\partial \beta_{lmn}} &= \frac{\partial}{\partial \beta_{lmn}} \left\{ \mathbb{E}_{X,Y,Z} \left(\sum_{(i,j,k)\in\mathcal{P}} \log \left[\beta_{X_i Y_j Z_k}^{r_{ijk}} (1 - \beta_{X_i Y_j Z_k})^{1 - r_{ijk}} \right] \right) \right\} \\
&= \sum_{(i,j,k)\in\mathcal{P}} a_{il} b_{jm} c_{kn} \left(r_{ijk} \frac{1}{\beta_{lmn}} + (1 - r_{ijk}) \frac{1}{\beta_{lmn} - 1} \right) \\
&= 0 \quad 。
\end{aligned}
$$

因此，

$$
\beta_{lmn} = \frac{\sum_{(i,j,k)\in\mathcal{P}^1} a_{il} b_{jm} c_{kn}}{\sum_{(i,j,k)\in\mathcal{P}} a_{il} b_{jm} c_{kn}} \quad , \tag{7.15}
$$

其中 $\mathcal{P}^1 \triangleq \{(i,j,k) | (i,j,k) \in \mathcal{P},$ 以及 $r_{ijk} = 1\}$。

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial \pi_l} &= \frac{\partial}{\partial \pi_l}\left(\mathbb{E}_X \sum_{i=1}^{I} \log \pi_{X_i}\right) - \lambda \\ &= \frac{1}{\pi_l}\sum_{i=1}^{I} a_{il} - \lambda \\ &= 0 \quad 。\end{aligned}$$

因此，

$$\pi_l = \frac{\sum_{i=1}^{I} a_{il}}{\sum_{l=1}^{L}\sum_{i=1}^{I} a_{il}} \quad , \qquad l = 1, \ldots, L \quad 。 \tag{7.16}$$

类似地，我们可以获得 θ_m 和 μ_n 的更新方程：

$$\theta_m = \frac{\sum_{j=1}^{J} b_{jm}}{\sum_{m=1}^{M}\sum_{j=1}^{J} b_{jm}} \quad , \qquad m = 1, \ldots, M \quad 。 \tag{7.17}$$

$$\mu_n = \frac{\sum_{k=1}^{K} c_{kn}}{\sum_{n=1}^{N}\sum_{k=1}^{K} c_{kn}} \quad , \qquad n = 1, \ldots, N \quad 。 \tag{7.18}$$

在更新过程中，我们也使用了第 5.3.1 节中介绍的线性组合新旧参数值的方法。例如，在更新 a_{il} 时，我们首先利用(7.12)计算出新的值 a_{il}^{new}，然后使用下式获得更新后的 a_{il} ：

$$a_{il} \leftarrow \delta \cdot a_{il}^{\text{new}} + (1-\delta) \cdot a_{il} \quad ,$$

其中 δ 为组合系数，这里我们取 $\delta = 0.6$ 。

更新过程的主要计算量来自于对 a_{il} 、 b_{jm} 和 c_{kn} 的更新，它们对应的计算量为 $O(|\mathcal{P}|LMN)$, 其中 $|\mathcal{P}|$ 为训练数据集 $\mathcal{P}$ 的大小。存储 Θ 所需的空间为 $O(IL + JM + KN + LMN)$ 。

当上面的迭代过程收敛后，我们使用以下公式获得测试数据集上的预测评分：

$$\hat{r}_{ijk} = \mathbb{E}_{\mathbf{X},\mathbf{Y},\mathbf{Z}}(\boldsymbol{R}_{ijk}|\mathbf{R}, \Theta) = \sum_{l=1}^{L}\sum_{m=1}^{M}\sum_{n=1}^{N} a_{il} b_{jm} c_{kn} \beta_{lmn} \quad 。 \tag{7.19}$$

7.2　立方聚类（Cube Clustering）模型

Dhillon [27] 提出了一种新的聚类模型——联合聚类（*Co-clustering*）模型。不同于传统聚类高维数据的方法，联合聚类模型同时聚类不同维度上的对象。它首先获得新的对象类别标识以便最小化交互信息（*mutual information*）的损失，然后更新各个类别对应的分布。这个过程迭代地进行，直到给定的停止准则成立。

对于现在的立方填补问题，借鉴联合聚类的想法我们可以迭代更新对象的类别标识和不同维度之间的聚类参数。我们使用最小化下面的均方误差（mean squared error，简记为MSE）以获得各个方向上对象的类别标识以及模型参数：

$$\mathcal{F}(C_X(\cdot), C_Y(\cdot), C_Z(\cdot); \beta) = \sum_{(i,j,k)\in\mathcal{P}} \left(r_{ijk} - \beta_{C_X(i)C_Y(j)C_Z(k)}\right)^2 \quad , \tag{7.20}$$

其中 $C_X(i)$ 表示 X 方向上的第 i 个对象所属的类别（$C_Y(j)$ 和 $C_Z(k)$ 的意义类似）。我们称此模型为立方聚类（*Cube Clustering*，简记为 CC）模型。

正如 [27] 中的联合聚类模型，CC 模型中 X 方向上第 i 个对象所对应的类别标识 $C_X(i)$ 可使用如下方式进行更新：

$$\begin{aligned} C_X(i) &= \arg\min_l \sum_{(j,k)\in\mathcal{P}_i} \left(r_{ijk} - \beta_{lC_Y(j)C_Z(k)}\right)^2 \\ &= \arg\min_l \sum_{(j,k)\in\mathcal{P}_i} \left(\beta^2_{lC_Y(j)C_Z(k)} - 2r_{ijk}\beta_{lC_Y(j)C_Z(k)}\right) \\ &= \arg\min_l \sum_{(j,k)\in\mathcal{P}_i} \left(\beta_{lC_Y(j)C_Z(k)} \left[\beta_{lC_Y(j)C_Z(k)} - 2r_{ijk}\right]\right) \quad 。 \end{aligned} \tag{7.21}$$

类似地我们可以获得 $C_Y(j)$ 和 $C_Z(k)$ 的更新方式。

在更新对象的类别标识后，我们使用下式更新模型参数 β：

$$\beta_{lmn} = \frac{|\mathcal{P}^1_{lmn}|}{|\mathcal{P}_{lmn}|} \quad , \tag{7.22}$$

其中

$$\mathcal{P}_{lmn} \triangleq \{(i,j,k) \mid (i,j,k)\in\mathcal{P}, \text{和 } C_X(i)=l, C_Y(j)=m, C_Z(k)=n\} \ ,$$

以及

$$\mathcal{P}^1_{lmn} \triangleq \{(i,j,k) \mid (i,j,k) \in \mathcal{P}_{lmn}, \text{和 } r_{ijk} = 1\} \text{。}$$

既然训练数据集极度稀疏， $\mathcal{P}_{lmn}$ 可能包括很少的元素，这使得(7.22) 中的更新往往并不可靠。降低这种不可靠性的有效方法是朝着它的平均值

$$\bar{r} \triangleq \frac{|\mathcal{P}^1|}{|\mathcal{P}|} \quad \text{（其中 } \mathcal{P}^1 \triangleq \{(i,j,k) \mid (i,j,k) \in \mathcal{P}, \text{以及 } r_{ijk} = 1\} \text{）}$$

压缩(7.22)中获得的值。因此，参数 β 的更新公式变为：

$$\beta_{lmn} = \frac{|\mathcal{P}^1_{lmn}| + \kappa\bar{r}}{|\mathcal{P}_{lmn}| + \kappa} \quad , \tag{7.23}$$

其中 κ 为压缩系数，实验中我们取 $\kappa = 500$ 。

更新过程的主要计算量来自于对 $C_X(i)$ 、 $C_Y(j)$ 和 $C_Z(k)$ 的更新，它们对应的计算量为 $O(|\mathcal{P}|(L+M+N))$ 。所以相对于第 7.1 节中的 BC 模型，CC 模型的计算量要小得多。迭代过程中 CC 模型所需的存储空间为 $O(I+J+K+LMN)$。

CC 模型训练的具体过程见算法 7.1。在获得了模型参数 β 和各个方向上对象的类别标识后，我们使用下式获得预测评分：

$$\hat{r}_{ijk} = \beta_{C_X(i)C_Y(j)C_Z(k)} \quad \text{。} \tag{7.24}$$

算法 7.1 (CC) 选择 X、Y 和 Z 方向上的聚类类别数 L 、 M 和 N ，以及压缩系数 κ 。初始化各个方向上对象所属的类别 $C_X(i)$ 、 $C_Y(j)$ 和 $C_Z(k)$ （例如随机抽取出这些值）。

1. 使用(7.23)更新参数 β ，然后使用下式更新 X 方向上对象的所属类别:

$$C_X(i) = \arg\min_l \sum_{(j,k)\in\mathcal{P}_i} \left(\beta_{lC_Y(j)C_Z(k)} \left[\beta_{lC_Y(j)C_Z(k)} - 2r_{ijk}\right]\right), \quad (i = 1,\ldots,I) \text{。}$$

2. 使用(7.23)更新参数 β ，然后使用下式更新 Y 方向上对象的所属类别:

$$C_Y(j) = \arg\min_m \sum_{(k,i)\in\mathcal{P}_j} \left(\beta_{C_X(i)mC_Z(k)} \left[\beta_{C_X(i)mC_Z(k)} - 2r_{ijk}\right]\right), \quad (j = 1,\ldots,J) \text{。}$$

3. 使用(7.23)更新参数 β，然后使用下式更新 Z 方向上对象的所属类别：

$$C_Z(k) = \arg\min_n \sum_{(i,j)\in\mathcal{P}_k} \left(\beta_{C_X(i)C_Y(j)n}\left[\beta_{C_X(i)C_Y(j)n} - 2r_{ijk}\right]\right), \quad (k = 1, \ldots, K)。$$

4. 使用(7.24)获得测试数据集上的预测评分，然后计算新的测试误差。如果新的测试误差比之前的测试误差小，则继续前面的更新步骤；否则终止算法。

7.3　立方分解（Cube Factorization）模型

Lathauwer 等 [50] 把二维矩阵的奇异值分解方法推广到高维张量，得到所谓的高阶奇异值分解（*High-Order SVD*，简记为 HOSVD），即任意一个维数为 $I_1 \times I_2 \times \ldots \times I_N$ 的张量 $\mathbf{R}$ 都可以分解为：

$$\mathbf{R} = \mathbf{C} \times_1 \mathbf{Y}_1 \times_2 \mathbf{Y}_2 \cdots \times_N \mathbf{Y}_N \quad , \tag{7.25}$$

其中 $\mathbf{C}$ 为满足一定条件的小尺寸 N 阶张量（通常被称为核张量（*core tensor*）），$\mathbf{Y}_n$ 为二阶酉矩阵（*unitary matrix*），而 $\times_n$ 被称为 n-式乘积（*n-mode product*）[50]（$n = 1, \ldots, N$）。与二阶 SVD 不同的是，$\mathbf{C}$ 并不是“对角张量”，它只满足全正交（*all-orthogonality*）性质[50]。更多的细节可见 [50]。

例如在三维情况下（$N = 3$），一个三阶张量 $\mathbf{R}$ 可以分解为 $\mathbf{R} = \mathbf{C} \times_1 \mathbf{X} \times_2 \mathbf{Y} \times_3 \mathbf{Z}$，即 $\mathbf{R}$ 的第 (i, j, k) 个元素为

$$r_{ijk} = \sum_{l=1}^{L} x_{il} \sum_{m=1}^{M} y_{jm} \sum_{n=1}^{N} z_{kn} c_{lmn} \quad , \tag{7.26}$$

其中 $\mathbf{X}$、$\mathbf{Y}$ 和 $\mathbf{Z}$ 为酉矩阵，而 $\mathbf{C}$ 为满足全正交条件的小尺寸三阶张量[50]。此分解的图形表示可见图 7.2。

既然矩阵分解模型可以高效地应用于求解二维协同过滤问题，我们认为三阶 SVD 对应的因子化模型也应该可以用于求解三维立方填补问题。具体地说，我们使用下式获得预测评分：

$$\hat{r}_{ijk} = \sum_{l=1}^{L} x_{il} \sum_{m=1}^{M} y_{jm} \sum_{n=1}^{N} z_{kn} c_{lmn} \quad , \tag{7.27}$$

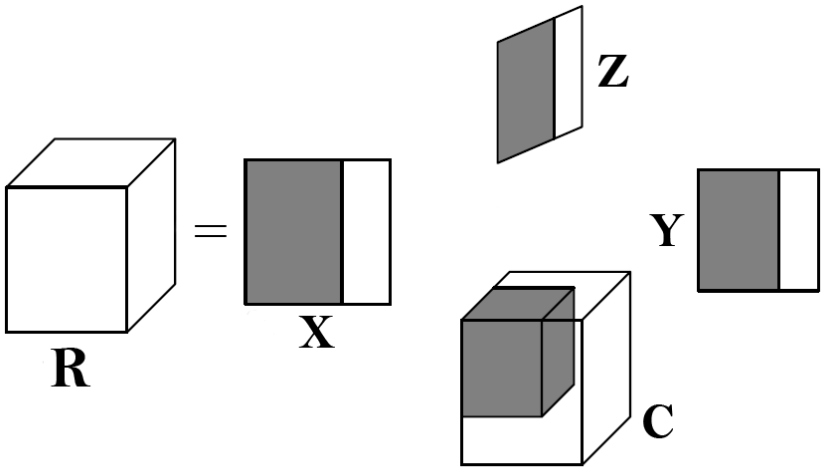

图 7.2: HOSVD 在三维时的图形表示

其中 $\mathbf{X}$ 、 $\mathbf{Y}$ 、 $\mathbf{Z}$ 和 $\mathbf{C}$ 为模型参数。类似地我们使用梯度下降法最小化下面的目标函数以获得其中的模型参数：

$$\begin{aligned}\mathcal{F}(\mathbf{X},\mathbf{Y},\mathbf{Z},\mathbf{C}) =& \frac{1}{2}\sum_{(i,j,k)\in\mathcal{P}}\Bigg[\left(r_{ijk}-\sum_{l=1}^{L}x_{il}\sum_{m=1}^{M}y_{jm}\sum_{n=1}^{N}z_{kn}c_{lmn}\right)^2 \\ &+\lambda\left(\|\mathbf{x}_i\|^2+\|\mathbf{y}_j\|^2+\|\mathbf{z}_k\|^2+\|\mathbf{C}\|^2\right)\Bigg] \quad ,\end{aligned} \tag{7.28}$$

其中 λ 为惩罚系数。我们称此模型为立方分解（*Cube Factorization*，简记为 CF）模型，其模型训练的具体过程见算法 7.2 。

算法 7.2 (CF) 选择潜在因子数 L 、 M 和 N ，以及惩罚参数 λ 和学习率 η ；初始化模型参数 $\mathbf{X}$ 、 $\mathbf{Y}$ 、 $\mathbf{Z}$ 和 $\mathbf{C}$ （例如从均匀或正态分布中随机抽取出这些值）。

1. 对每个指标对 $(i,j,k)\in\mathcal{P}$:

   (a) 计算评分残差 $e_{ijk}=r_{ijk}-\hat{r}_{ijk}$ ，其中 $\hat{r}_{ijk}$ 利用(7.27)计算获得。

   (b) 更新 $\mathbf{X}$ 的第 i 行、 $\mathbf{Y}$ 的第 j 行、 $\mathbf{Z}$ 的第 k 行，以及核立方阵 $\mathbf{C}$:

$$x_{il} \mathrel{+}= \eta\cdot\left(e_{ijk}\sum_{m=1}^{M}y_{jm}\sum_{n=1}^{N}z_{kn}c_{lmn}-\lambda x_{il}\right),\ (l=1,\ldots,L)\ ;$$

$$y_{jm} \mathrel{+}= \eta\cdot\left(e_{ijk}\sum_{l=1}^{L}x_{il}\sum_{n=1}^{N}z_{kn}c_{lmn}-\lambda y_{jm}\right),\ (m=1,\ldots,M)\ ;$$

$$z_{kn} \mathrel{+}= \eta \cdot \left(e_{ijk} \sum_{l=1}^{L} x_{il} \sum_{m=1}^{M} y_{jm} c_{lmn} - \lambda z_{kn} \right), \quad (n = 1, \ldots, N) ;$$

$$c_{lmn} \mathrel{+}= \eta \cdot \left(e_{ijk} x_{il} y_{jm} z_{kn} - \lambda c_{lmn} \right),$$

$$(l = 1, \ldots, L;\ m = 1, \ldots, M;\ n = 1, \ldots, N) 。$$

2. 使用(7.27)获得测试数据集上的预测评分，然后计算新的测试误差。如果新的测试误差比之前的测试误差小，则继续前面的更新步骤；否则终止算法。

算法 7.2 的每部迭代中计算 e_{ijk} 以及更新 x_{il}、 y_{jm} 和 z_{kn} 的时间复杂度都是 $O(|\mathcal{P}|LMN)$，而迭代过程中存储模型变量所需的空间为 $O(IL + JM + KN + LMN)$。

7.4　邻居（Neighborhood）模型

正如我们在第 3.1 节中所介绍的，邻居模型（kNN）由于预测精度高、容易解释以及容易实现等优点而被广泛应用于求解二维协同过滤问题。这一节我们介绍如何把 kNN 应用于求解三维的立方填补问题。

和二维情形类似，我们可以计算同一方向上两个对象之间的相似度。但由于三维情形更加显著的稀疏性问题，两个对象之间已知的共有评分数量很少，这使得相似度的计算很不可靠。图 7.3 给出了稀疏度[②] $\delta_1 = 2 \times 10^{-4}$ 时某次抽取数据（见第 7.5.1 节）后训练数据集内 Z 方向两两对象之间共有评分数的直方图。

从图中可见不同对象之间共有的评分数量主要集中在 3 以下，这使得利用 Cosine 或 Pearson 相关性获得的对象之间的相似度变得非常不可靠。在我们的实验中我们使用下面的方法计算相似度：

$$s_{kk'} = \log(\alpha_1 n_1 - \alpha_2 n_2 + 1) \quad , \tag{7.29}$$

其中 n_1 和 n_2 分别表示在对象 k 和 k' 的公共评分中评分值相同和不同的数量，而 α_1 和 α_2 为相应的非负系数。

[②]稀疏度为已知评分占总评分数量的比例，也即 $|\mathcal{P}|/(IJK)$。

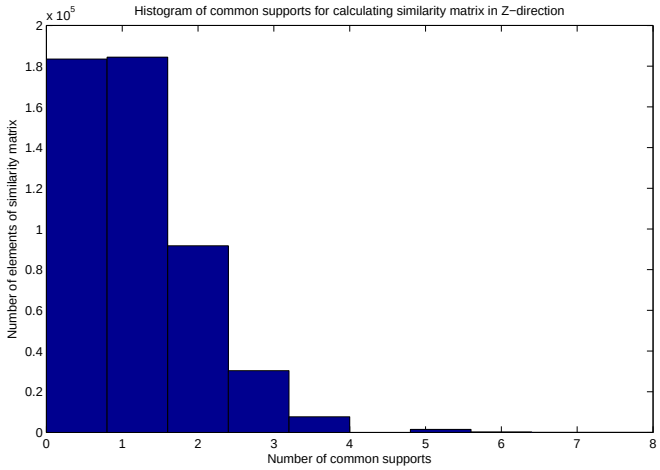

图 7.3: 当稀疏度 $\delta_1 = 2 \times 10^{-4}$ 时某次抽取数据后训练数据集内 Z 方向两两对象之间共有评分数的直方图。

我们的实验表明，虽然较之相似度 Cosine 和 Pearson，相似度(7.29)的使用可以获得更加精确的预测，但它的预测精度相比之前介绍的其他模型仍然很差。对于与图 7.3 中所使用数据同时抽取出的测试数据集，它的预测误差仅为 0.4982（其中的邻居数量取为 20，而 $\alpha_1 = 3$ 以及 $\alpha_2 = 12$），而此时最优的预测误差（详细说明见第 7.5 节）为 0.3962。

7.5 实验结果

就我们所知，目前还没有企业公开过可用于立方填补研究的免费数据，因此我们无法在实际数据上验证各个模型的预测效果。既然相对于二维的协同过滤问题，三维立方填补问题的最大困难源于它更加显著的稀疏性，下面的实验我们主要集中在验证各个模型相对于训练数据集稀疏度的强壮性。对于各个模型，我们的检验框架为：（1）对给定的不同训练数据集稀疏度（ δ_1 ），使用此模型产生相应的训练和测试数据集；（2）使用训练数据集和模型的训练方法学习模型参数，然后获得此模型在测试数据集上的均方根预测误差（RMSE）。为了降低训练和测试数据集抽取的随机性对最终预测误差的影响，对于每个给定的稀疏度 δ_1，我们都会重复抽取训练和测试数据集几次（实验中以“numSample”表示重复抽取次数）以获得在此稀疏度下模型的平均预测误差。

7.5.1 BC 模型和 CC 模型

对于 BC 和 CC 模型，我们使用下面的方法生成实验数据，而生成数据时所选取的参数值见表 7.1：

1. 从均匀分布 $\mathrm{U}(0,1)$ 中随机抽取出各个方向上的归一化的多项分布参数，分别记为 $\pi \in \mathbb{R}^L$ 、 $\theta \in \mathbb{R}^M$ 和 $\mu \in \mathbb{R}^N$ 。

2. 分别从多项分布 $\mathrm{Mul}(L,\pi)$ 、 $\mathrm{Mul}(M,\theta)$ 和 $\mathrm{Mul}(N,\mu)$ 中为对应方向上的所有对象抽取类别标识 $C(\cdot)$ 。

3. 从均匀分布 $\mathrm{U}(0,1)$ 中抽取三个方向上各个类别之间的喜好参数 β_{lmn} 。

4. 选定训练数据集（ $\mathcal{P}$ ）占整个立方体数据的比例 δ_1 ，即训练集包含的评分数为 $\delta_1 IJK$ ；随机抽取出训练集中包括的 $\delta_1 IJK$ 个指标对 (i,j,k) ，然后从伯努利分布 $\mathrm{Ber}(\beta_{lmn})$ 中抽取出评分值 r_{ijk} ，其中 l 、 m 和 n 分别为三个方向上对象 i 、 j 和 k 所属的类别指标。

5. 选定测试数据集（ $\mathcal{R}$ ）与训练数据集的比例 δ_2 ，即测试集包含的评分数为 $\delta_2\delta_1 IJK$；和上一步类似地抽取出测试集，但同时保证测试与训练数据集不相互重叠。

参数含义	参数说明
X 方向上的对象数 I	$5,000$
Y 方向上的对象数 J	$5,000$
Z 方向上的对象数 K	$1,000$
X 方向上的类别数 L	10
Y 方向上的类别数 M	10
Z 方向上的类别数 N	5
训练数据集所占的比例 δ_1	2×10^{-4} ～ $1/2^6\times10^{-4}$
测试集与训练集的比例 δ_2	20%

表 7.1: 生成训练和测试数据集时所使用的参数

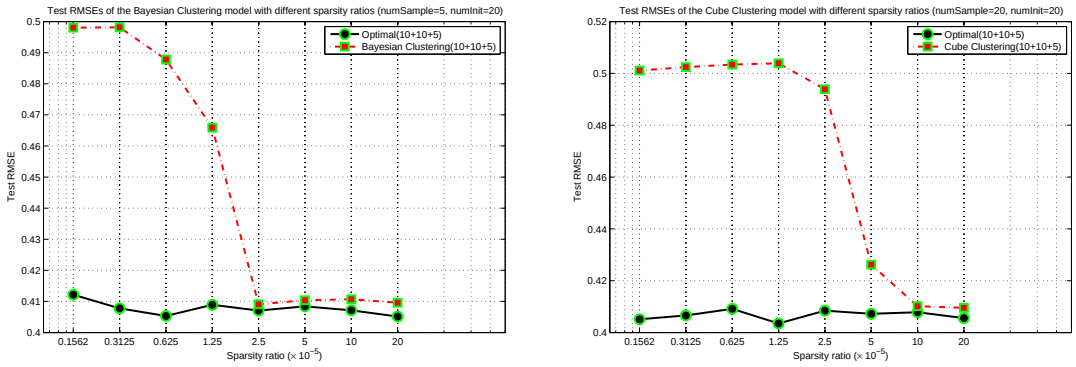

图 7.4: 相对于不同的数据稀疏度 δ_1 （横坐标），模型 BC （左）和 CC （右）在测试集上所获得的预测误差（纵坐标）图。图中的黑点表示使用生成数据的模型参数值进行预测（即“Optimal”预测）所获得的预测误差。而模型名称后面的数字分别代表各个方向上聚类的类别数 L 、 M 和 N ，如“$10+10+5$”表示 $L=10$ 、 $M=10$ 和 $N=5$ 。

图 7.4 给出了模型 BC 和 CC 应用于稀疏度不同的训练数据集时产生的预测模型在测试集上所获得的预测误差。为了与模型的预测结果进行比较，我们在图中也给出了使用生成数据的模型参数值进行预测（相应的模型被称为最优预测模型（*Optimal Prediction*，简记为 OP））所获得的预测误差。图中标示内模型名称后面的数字分别代表各个方向上聚类的类别数 L 、 M 和 N ，如“$10+10+5$”表示 $L=10$ 、 $M=10$ 和 $N=5$。

为了降低模型参数初始值对最终预测误差的影响，对于给定的训练数据集我们随机初始化模型参数多次（图 7.4 中的“numInit”值），然后选取最低预测误差作为模型此次训练的最终误差。正如我们在本节开头所说的，为了降低训练和测试数据集抽取的随机性对最终预测误差的影响，对于每个给定的稀疏度 δ_1 ，我们都会重复抽取数据集若干次（图 7.4 中的“numSample”表示重复抽取次数）以获得在此稀疏度下模型的平均预测误差。

从图 7.4 中我们可以看出，在稀疏度 δ_1 取值较大的情况下，BC 和 CC 模型都获得了与最优预测模型相近的预测误差。但随着 δ_1 的降低，BC 相对于 CC 呈现出更好的强壮性。例如，当 $\delta_1 = 2.5\times10^{-5}$ 时，BC 的预测误差为 0.4091 ，而 CC 的预测误差为 0.4939 。但当各个方向上的聚类类别数较大时，CC 所需的训练

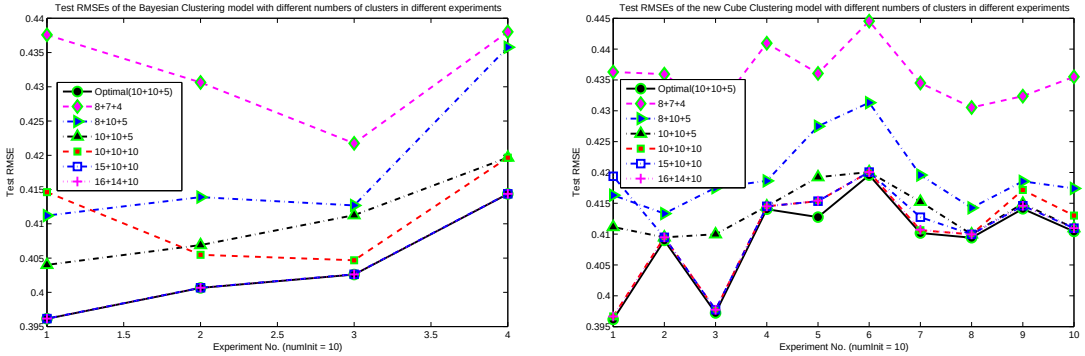

图 7.5: 当训练数据集的稀疏度 $\delta_1 = 2 \times 10^{-4}$，且各个方向上聚类的实际类别数 $L = 10$、$M = 10$ 和 $N = 5$ 时，模型 BC （左）和 CC （右）选取不同的聚类类别数时在测试集上所获得的预测误差（纵坐标）图。图中的横坐标表示给定参数时训练数据集抽取的不同次数。图中的黑点表示使用生成数据的模型参数值进行预测（即“Optimal”预测）所获得的预测误差。标示内的数字分别代表各个方向上训练模型时所使用的聚类类别数，如“$10 + 10 + 5$”表示 $L = 10$、$M = 10$ 和 $N = 5$。

时间要比 BC 少得多。

除了训练数据集的稀疏度对两个模型预测精度的影响外，我们也希望知道聚类类别数的选取对它们最终的预测精度会产生什么样的影响。图 7.5 给出了在固定训练数据集稀疏度 $\delta_1 = 2 \times 10^{-4}$ 的情况下，选取不同的聚类类别数时产生的预测模型在测试数据集上所获得的预测误差。为了降低数据集抽取的随机性对最终预测误差的影响，我们重复抽取若干次数据集（对于 BC 模型，重复抽取了 4 次；对于 CC 模型，重复抽取了 10 次），并在这些数据集上选取不同的聚类类别数，最终获得相应模型在测试集上的预测误差。图 7.5 中的结果表明这两个模型对类别数的选取都不敏感。当选取比实际类别数更大的类别数时，模型不会产生过度拟合问题，它们仍能很好地拟合真实数据。

7.5.2　CF 模型

在生成 CF 模型所使用的数据集时，除了训练数据集稀疏度 δ_1 的范围稍微有所不同外，我们仍然使用表 7.1 中给出的参数值。生成数据集的具体方法如下：

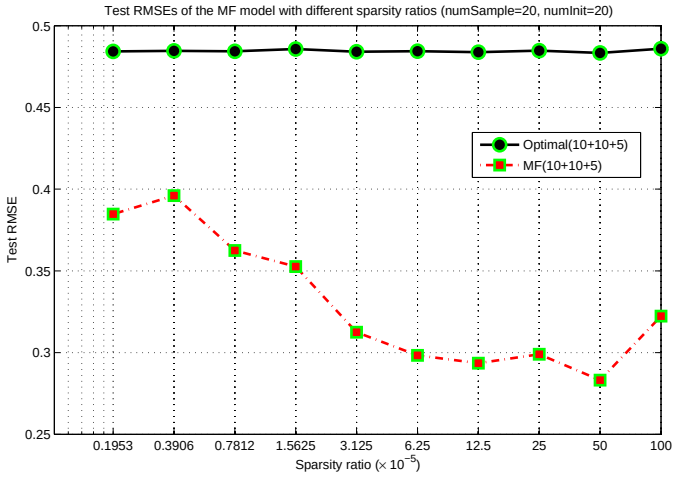

图 7.6: 相对于不同的数据稀疏度 δ_1 （横坐标），模型 CF 在测试集上所获得的预测误差（纵坐标）图。图中的黑点表示使用生成数据的参数值进行预测（即“Optimal”预测）所获得的预测误差。而模型名称后面的数字分别代表各个方向上聚类的类别数 L、 M 和 N，如“$10+10+5$”表示 $L=10$、 $M=10$ 和 $N=5$。

1. 从均匀分布 $\mathrm{U}(0,0.5)$ 中随机抽取出模型参数 $\mathbf{X}$、 $\mathbf{Y}$、 $\mathbf{Z}$ 和 $\mathbf{C}$。

2. 选定训练数据集（ $\mathcal{P}$ ）占整个立方体数据的比例 δ_1，即训练集包含的评分数为 $\delta_1 IJK$；随机抽取出训练集中包括的 $\delta_1 IJK$ 个指标对 (i,j,k)，然后使用(7.26)计算得到非整数评分 r_{ijk}^{float}；如果 $r_{ijk}^{\mathrm{float}} \geq 0.5$，则最终选取 $r_{ijk}=1$，否则选取 $r_{ijk}=0$。

3. 选定测试数据集（ $\mathcal{R}$ ）与训练数据集的比例 δ_2，即测试集包含的评分数为 $\delta_2\delta_1 IJK$；和上一步类似地抽取出测试集，但同时保证测试与训练数据集不相互重叠。

图 7.6 给出了模型 CF 应用于稀疏度不同的训练数据集时产生的预测模型在测试集上所获得的预测误差。和前面类似，我们在图中也给出了使用生成数据的模型参数值进行预测（图中的“Optimal”）所获得的预测误差。图中标示内模型名称后面的数字分别代表各个方向上聚类的类别数 L、 M 和 N，如“$10+10+5$”表示 $L=10$、 $M=10$ 和 $N=5$。

与图 7.4 中 BC 和 CC 模型结果不同的是，图 7.6 中使用原始模型参数产生的最优预测模型的预测效果反而比训练得到的 CF 模型的预测效果更差。产生这个现象的原因是在生成 CF 模型所使用的训练和测试数据集时，我们对非整数评分加入了截断，这种截断导致了最优预测模型的预测效果反而比 CF 模型更差。图 7.6 也表明 CF 模型的预测误差随着数据稀疏程度的加剧缓慢增加，这与图 7.4 中 BC 和 CC 模型的预测误差随着数据稀疏程度的加剧陡然增加完全不同。这说明 CF 模型对数据稀疏度表现出了更强的适应性。

第八章　总结与展望

Netflix 于 2006 年发布的 Netflix Prize 竞赛极大地推动了推荐系统中协同过滤（CF）算法的研究。经过上万参赛团队近三年的不懈努力，Quiz Set 上预测误差的降幅终于超过了 10%，由 Cinematch 的 0.9525 降至团队 BPC 的 0.8567。本文中我们系统地介绍了各个参赛团队近三年来发展的应用于 Netflix Prize 数据集的典型 CF 算法，其中包括我们组合 FCM 和 MF 模型所提出的新型聚类模型 MFCM，以及对于离散 CF 问题比 MF 模型更加合适的 BMF 模型。

在本文的第七章中我们把二维的 CF 推广到三维情形。我们介绍了三维情形在互联网中的实际应用，提出了一些可用于求解三维情形的有效模型，并在构造的虚拟数据中对这些模型进行了相应的检验。

虽然近几年 CF 推荐算法在众多研究者的努力下取得了很大的发展，但大部分出版的研究都集中在如何发展新的推荐算法上，而 CF 中一些其他方面的问题却并没有得到应有的重视。下面我们列举了其中一些具有代表性的问题，我们相信对这些问题的深入探讨一定可以进一步改进 CF 算法的推荐精度。如何更好地解决这些问题也将是我们未来工作的主要方向。

算法理论依据的缺乏 到目前为止，大部分关于 CF 的研究都是集中在发展实用的推荐算法，而这些高效算法背后的理论却很少有人问津。正如我们在第七章开头所讲的， CF 从数学上来讲是矩阵填充问题。关于矩阵填充，一些研究者已经在一些特殊假设下获得了一些理论结果[18–20]，并依据获得的理论结果设计了一些填充算法[17, 42–44, 54]。但从我们的实验结果来看，这些算法目前只能处理比较小的矩阵填充问题，而且往往只适用于利用相应理论构造出来的数据。对于真实的 CF 数据集，这些算法甚至无法获得收敛的结果。要把这些来源于理论的算法应用于真实的大规模 CF 数据集（如 Netflix Prize 数据集），研究者需要做的工作还有很多。

评分的非随机缺失性 对于一般的数据缺失问题，一个通常使用的默认假设是数据的缺失是完全随机的。但对于 CF 问题而言，用户往往只对自己喜欢或讨厌的产品给予评分。用户的这种行为必然导致 CF 问题中的评分缺失并不满足随机缺失性。这个结论也在 Marlin 的研究 [57] 中得到证明。通过考虑评分的非随机缺失性质，Marlin [57] 可以在原始 RBM 模型的基础上获得进一步的预测精度提升。

数据的稀疏性 正如我们之前多次强调的， CF 推荐算法面临的一个很大问题是可用评分数据的稀疏性。中国有一句俚语：“巧妇难为无米之炊”，如果没有足够的数据信息，再强大的算法也不可能获得好的推荐效果。这也是为什么众多电子商务公司都愿意花费巨大的人力财力来收集更多更精确的用户使用数据。很多人认为之所以 Amazon 能够为用户提供精准的产品推荐，就是因为 Amazon 在十几年内积累了用户大量的使用数据，而并不是因为它所使用的推荐算法比其他公司的推荐算法更加高效[①]。

实际情况中，如果可利用的显式评分较少，很多研究者建议使用所谓的隐式信息（*implicit information*）来提高推荐精度。例如，对于网上商店而言，除了用户对产品的显式评分，我们还可以使用用户查看过的产品、添加进购物篮的产品和购买了的产品等信息获得更加精确的个性化推荐[1, 25]。关于隐式信息如何被用来改进推荐效果，一些研究者已经做过一些探讨[40, 66]，但目前这方面公开发表的文献还相对较少。

算法的可扩展性 很多 CF 模型虽然可以获得很好的预测精度，但由于它们训练的计算复杂度较高而难以应用于真实的大规模 CF 问题。一种降低模型训练时间的方法是对它们进行并行化处理，也即把一个大的计算问题划分为若干个小的计算问题然后使用多台计算机同时处理这些小的计算问题。但对于很多的 CF 算法，如何把它们并行化却又是一大难题。发展高效的并行化 CF 算法在现实应用中已经变得越来越重要。

[①] 一些详细的讨论见 http://answers.google.com/answers/threadview/id/29373.html 。

与 CBF 推荐算法的组合 虽然目前研究者已经发展了很多组合 CBF 和 CF 算法的方法（见第 1.1 节），但这些方法大部分还只是停留在“线性组合”的阶段。如何更加高效地整合所有可以利用的信息仍然是推荐系统中最重要的问题之一。

附录 A　Global Effects 中压缩的 Bayes 解释

假设真实的 θ_u 是从如下正态分布中独立抽取出来的[6]：

$$\theta_u \sim \mathrm{N}(\mu, \tau^2) \text{（其中 } \mu \text{ 和 } \tau \text{ 为已知的参数）} \quad , \tag{A.1}$$

且

$$\hat{\theta}_u|\theta_u \sim \mathrm{N}(\theta_u, \sigma_u^2) \text{（其中 } \sigma_u^2 \text{ 为已知的参数）} \quad , \tag{A.2}$$

那么利用

$$P(\theta_u|\hat{\theta}_u) \propto P(\hat{\theta}_u|\theta_u)P(\theta_u) \quad , \tag{A.3}$$

我们可以得到

$$\mathbb{E}(\theta_u|\hat{\theta}_u) = \int \theta_u P(\theta_u|\hat{\theta}_u)d\theta_u = \frac{\tau^2\hat{\theta}_u + \sigma_u^2\mu}{\tau^2 + \sigma_u^2} \quad 。 \tag{A.4}$$

μ 和 σ_u^2 能够从真实的 θ_u 中计算出来，而经验 Bayes[28]也给出了 τ^2 的最大似然估计：

$$\tau^2 = \frac{\sum_u[(\hat{\theta}_u - \mu)^2 - \sigma_u^2]/(\tau^2 + \sigma_u^2)^2}{\sum_u 1/(\tau^2 + \sigma_u^2)^2} \quad 。 \tag{A.5}$$

如果简单地选取 $\mu = 0$ 和 $\sigma_u^2 \propto \frac{1}{|\mathcal{P}_u|}$，我们就获得了(2.3)中的压缩公式[6]。

附录 B　二项混合（Mixture of Binomials）模型

多项混合模型（*MMM*）[56]假设每类用户对每部电影的评分满足多项分布。正如我们在第 5.2 节中所解释的，多项分布的多峰性使得它并不适合用来描述评分。比多项分布更加合理的模型假设是二项分布。本章我们提出了一种新的混合模型——二项混合模型（*Mixture of Binomials Model*，简记为 MBM）。

和 MMM 类似，它假设所有用户可以分为 K 类，用户对所有电影的评分只依赖于他（她）所属的类别。但与 MMM 不同的是，MBM 假设每类用户对每部电影的评分满足二项分布而不是多项分布。MBM 的模型假设可以表达为如下的数学形式，其图形表示可见图 B.1：

$$P(\boldsymbol{R}_{u,m}|\boldsymbol{Z}_u) = \mathrm{B}(\boldsymbol{R}_{u,m}|S-1, \beta_{m\boldsymbol{Z}_u}) \quad , \tag{B.1}$$

$$P(\boldsymbol{Z}_u) = \theta_{\boldsymbol{Z}_u} \quad , \tag{B.2}$$

其中 β_{mz} 表示的是第 z 类用户对第 m 部电影的喜好程度，而 θ_z 为每个用户属于第 z 类的概率。

对于 Netflix Prize 问题而言，用户可以为每部电影提供一个评分，也即在给定的五颗星内给这部电影多少颗星。既然给出的评分至少为 1 分，我们可以认为用户会根据自己对某部电影的喜好把 4 颗星中的每颗放入两个篮子（分别代表“喜欢”和“不喜欢”）中的一个，这样用户对电影的评分即满足二项分布。对评分的这种解释同样应用于其他模型（如第 5.2 节的 BMF）的假设当中。

使用二项分布假设的另一个好处是它可以减少模型参数数量。二项分布的参数值只有一个，而多项分布的参数值有 S 个。

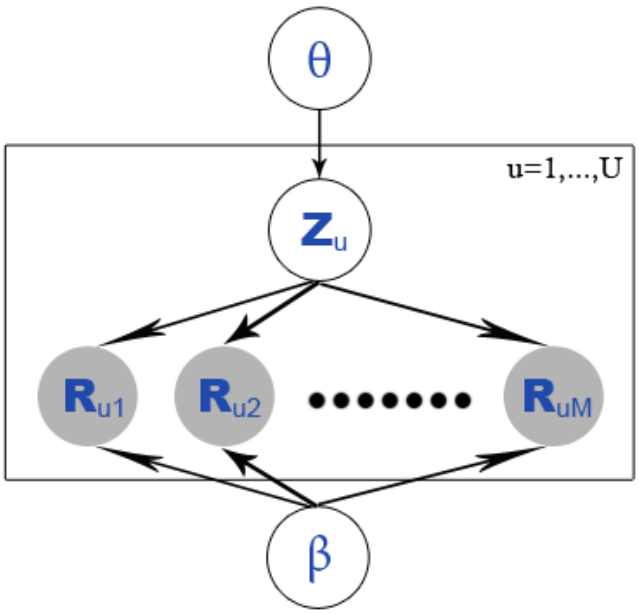

图 B.1: 二项混合模型的图形表示

MBM 中整个数据集的似然如下：

$$\begin{aligned} P(\mathbf{R},\mathbf{Z}|\Theta) &= P(\mathbf{R}|\mathbf{Z},\beta)\cdot P(\mathbf{Z}|\theta) \\ &= \prod_{u=1}^{U} P(z_u|\theta)\cdot \prod_{(u,m)\in\mathcal{P}} P(r_{u,m}|z_u,\beta) \\ &= \prod_{u=1}^{U}\left\{\theta_{z_u}\cdot\prod_{m\in\mathcal{P}_u}\binom{S-1}{r_{u,m}-1}\beta_{mz_u}^{r_{u,m}-1}(1-\beta_{mz_u})^{S-r_{u,m}}\right\} \quad , \end{aligned} \tag{B.3}$$

其中 β_{mz} 为第 z 类用户对第 m 部电影评分所满足的二项分布的参数，而 $\Theta \triangleq \{\beta,\theta\}$。

类似于 [56] 中所介绍的，我们这里使用 VEM 求解 MBM，也即使用可完全分解的 q-分布来近似 $P(\mathbf{Z}|\mathbf{R},\Theta)$ [5, 97]：

$$P(\mathbf{Z}|\mathbf{R},\Theta)\simeq Q(\mathbf{Z})\triangleq\prod_{u=1}^{U}Q(z_u)=\prod_{u=1}^{U}\psi_{uz_u} \quad 。 \tag{B.4}$$

所以 MBM 关于 q-分布的变分自由能可以写为：

$$\begin{aligned}
&\mathcal{F}(Q(\mathbf{Z});\Theta) = \mathbb{E}_{\mathbf{Z}}[\log P(\mathbf{R},\mathbf{Z}|\Theta) - \log Q(\mathbf{Z})] \\
=&\mathbb{E}_{\mathbf{Z}}\left\{\sum_{(u,m)\in\mathcal{P}} \log\binom{S-1}{r_{u,m}-1} + (r_{u,m}-1)\log\beta_{mz_u} + (S-r_{u,m})\log(1-\beta_{mz_u})\right\} \\
&+ \mathbb{E}_{\mathbf{Z}}\left\{\sum_{u=1}^{U}\log\theta_{z_u}\right\} - \mathbb{E}_{\mathbf{Z}}Q(\mathbf{Z}) \quad ,
\end{aligned} \tag{B.5}$$

其中 $\mathbb{E}_{\mathbf{Z}}(\cdot)$ 表示关于 $Q(\mathbf{Z})$ 的期望值。

对(B.5)中的 $\mathcal{F}$ 关于 ψ 、 θ 和 β 求导，可以得到下面的更新公式。

$$\begin{aligned}
\frac{\partial\mathcal{F}}{\partial\psi_{uz}} &= \sum_{m\in\mathcal{P}_u}\left[\log\binom{S-1}{r_{u,m}-1} + (r_{u,m}-1)\log\beta_{mz} + (S-r_{u,m})\log(1-\beta_{mz})\right] \\
&\quad + \log\theta_z - \log\psi_{uz} - 1 - \lambda \\
&= 0 \quad ,
\end{aligned}$$

因此，

$$\psi_{uz} \propto \exp\left\{\sum_{m\in\mathcal{P}_u}\left[(r_{u,m}-1)\log\beta_{mz} + (S-r_{u,m})\log(1-\beta_{mz})\right] + \log\theta_z\right\} \quad 。 \tag{B.6}$$

$\mathcal{F}$ 关于 β_{mz} 求导，

$$\begin{aligned}
\frac{\partial\mathcal{F}}{\partial\beta_{mz}} &= \frac{\partial}{\partial\beta_{mz}}\left\{\sum_{u\in\mathcal{P}^m}\psi_{mz}\left[(r_{u,m}-1)\log\beta_{mz} + (S-r_{u,m})\log(1-\beta_{mz})\right]\right\} \\
&= \sum_{u\in\mathcal{P}^m}\psi_{uz}\left(\frac{r_{u,m}-1}{\beta_{mz}} + \frac{S-r_{u,m}}{1-\beta_{mz}}\right) \\
&= 0 \quad ,
\end{aligned}$$

因此，

$$\beta_{mz} = \frac{\sum_{u\in\mathcal{P}^m}\psi_{uz}(r_{u,m}-1)}{(S-1)\sum_{u\in\mathcal{P}^m}\psi_{uz}} \quad 。 \tag{B.7}$$

最后，

$$\frac{\partial\mathcal{F}}{\partial\theta_z} = \sum_{u=1}^{U}\frac{\psi_{uz}}{\theta_z} - \lambda = 0 \quad ,$$

因此，

$$\theta_z = \frac{1}{U}\sum_{u=1}^{U}\psi_{uz} \quad 。 \tag{B.8}$$

在获得了模型参数 ψ 、 θ 和 β 后，我们使用下面的公式获得用户 u 对电影 m 评分所满足的二项分布的参数：

$$\hat{\beta}_{um} = \sum_{z=1}^{K}\beta_{mz}\cdot Q(Z_u = z) = \sum_{z=1}^{K}\beta_{mz}\cdot\psi_{uz} \quad 。 \tag{B.9}$$

而最终的预测评分为评分二项分布的期望值：

$$\hat{r}_{u,m} = 1 + (S-1)\hat{\beta}_{um} \quad 。 \tag{B.10}$$

在更新过程中，我们也使用了第 5.3.1 节介绍的线性组合新旧参数值的方法。但不同于(5.60)和(5.61)的是，我们使用了前两次的旧参数值与新的参数值进行组合。例如，在第 l 次迭代过程中，我们使用下面的公式更新参数 β ：

$$\beta^{(l)} \leftarrow \delta_1\cdot\beta^{(l-1)} + (1-\delta_1)\cdot(\delta_2\cdot\beta^{(l-2)} + (1-\delta_2)\cdot\beta^{(l-3)}) \quad , \tag{B.11}$$

其中 δ_1 和 δ_2 为组合系数。

对于 Netflix Prize 问题，当用户类别数 $K = 20$ 以及组合系数 $\delta_1 = \delta_2 = 0.7$ 时，MBM 获得了 0.9542 的 Probe RMSE ，而 MMM 仅获得了 0.9694 的 Probe RMSE 。

附录 C　超参数（hyper-parameter）学习

C.1　Nelder-Mead 算法

Nelder-Mead (NM) 是求解无约束非线性优化问题的一种算法。NM 属于单纯形直接搜索（*simplex-based direct search*）方法，它只需要函数值，而并不需要去近似函数的导数值。

一个 K 维空间（$\mathbb{R}^K$）中的单纯形（*simplex*）$\mathcal{S}$ 定义为由 $K+1$ 个顶点 $x_0, x_1, \ldots, x_K \in \mathbb{R}^K$ 所构成的凸包（*convex hull*）。例如 $\mathbb{R}^2$ 上的单纯形是三角形，而 $\mathbb{R}^3$ 上的单纯形是四面体，见图 C.1。

相比于大部分直接搜索算法在每部迭代中需要计算 K 个或更多点处的函数值，NM 在每部迭代中一般只需要计算一或二个点处的函数值，所以它被广泛应用于函数值计算代价较高的高维情形。

算法 C.1 给出了 NM 算法的基本框架，下面我们以最小化函数 $f(x)$（$x \in \mathbb{R}^K$）为例详细介绍算法 C.1 中的每一步[79]。

算法 C.1 **Nelder-Mead**

1. 建立初始单纯形 $\mathcal{S}^{(0)}$，令 $\mathcal{S} = \mathcal{S}^{(0)}$。

2. 重复以下步骤，直到循环终止条件满足：

   (a) 计算循环终止条件，如果条件成立则跳至步骤 3；

   (b) 根据规则更新单纯形 $\mathcal{S}$。

3. 返回当前单纯形 $\mathcal{S}$ 中获得最优值的顶点及其对应的函数值。

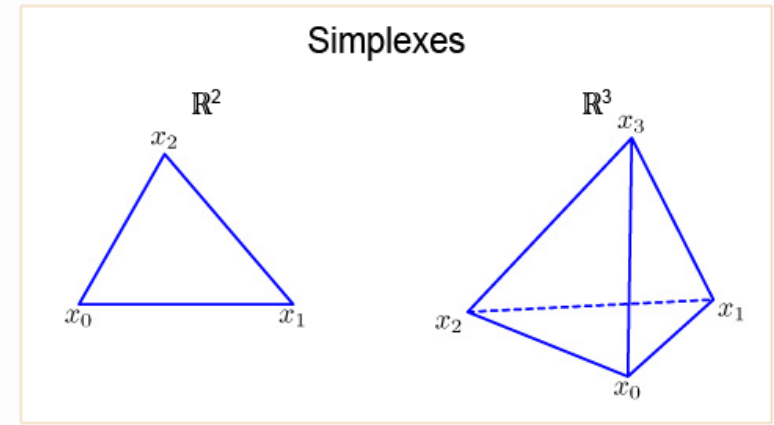

图 C.1: $\mathbb{R}^2$ 和 $\mathbb{R}^3$ 中的单纯形（原图可见 [79]）

初始的单纯形 $\mathcal{S}^{(0)}$ 的 $K+1$ 个顶点 $x_0, x_1, \ldots, x_K$ 通常围绕着某个点 $x^{(0)}$。通常我们可以取 $x_0 = x^{(0)}$，然后按如下两种方法中的一种选取其他 K 个点以便形成 $\mathcal{S}^{(0)}$：

- 选取 $x_k = x_0 + h_k e_k$（$k = 1, \ldots, K$），其中 e_k 为第 k 维坐标的单位坐标基，h_k 为对应的步长。
- 选取 x_k（$k = 1, \ldots, K$）使得 $\mathcal{S}^{(0)}$ 的边都有给定的某长度值。

算法 C.1 的核心是如何更新 $\mathcal{S}$，而每次迭代更新 $\mathcal{S}$ 包括如下三步：

1. **排序（ordering）**：找出当前 $\mathcal{S}$ 中具有最大、次大和最小函数值对应的顶点：

$$f_h = \max_k f_k \quad , \quad f_s = \max_{k \neq h} f_k \quad , \quad f_l = \min_{k \neq h} f_k \quad , \tag{C.1}$$

其中 $f_k = f(x_k)$（$k = 0, \ldots, K$）。

2. **找中心（centroid）**：计算除具有最大函数值的顶点外的其他顶点的平均值：

$$c = \frac{1}{K} \sum_{k \neq h} x_k \quad 。 \tag{C.2}$$

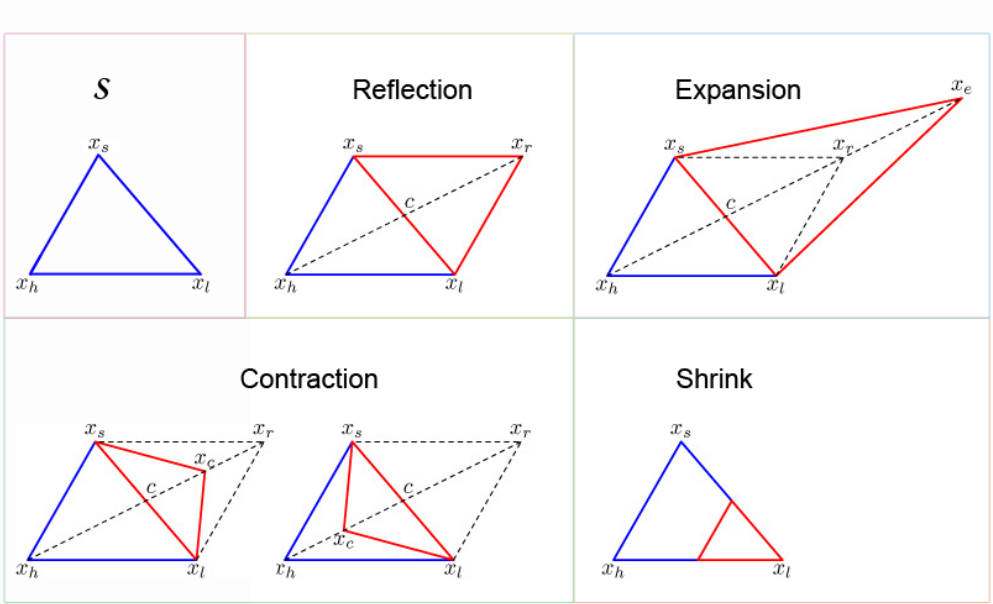

图 C.2: $\mathbb{R}^2$ 中各种变形的示例图（原图可见 [79]）。图中蓝色表示变形前的 $\mathcal{S}$，红色表示变形后的 $\mathcal{S}$。

3. **变形（transformation）**：变形当前的单纯形 $\mathcal{S}$。首先，尝试使用反射（*reflection*）、扩展（*expansion*）或紧缩（*contraction*）获得一个比 x_h 更好（函数值更小）的顶点作为 x_h 的替代，其中的候选顶点都位于由 x_h 和 c 决定的直线上，而且最多需要计算两个候选点。如果成功获得新顶点，则变形完成，此步迭代结束；否则朝着最好的顶点 x_l 压缩（*shrink*）$\mathcal{S}$。具体做法如下（图 C.2 中给出了下面操作在 $\mathbb{R}^2$ 上的示例）：

   反射 计算反射点 $x_r = c + \alpha(c - x_h)$ 及其函数值 $f_r = f(x_r)$，其中 $\alpha > 0$，通常取为 1。如果 $f_l \le f_r < f_s$，接受 x_r 并终止此次迭代。

   扩展 如果 $f_r < f_l$，则计算扩展点 $x_e = c + \gamma(x_r - c)$ 及其函数值 $f_e = f(x_e)$，其中 $\gamma > 1$，通常取为 2。如果 $f_e < f_r$，接受 x_e 并终止此次迭代；如果 $f_e \ge f_r$，接受 x_r 并终止此次迭代。

   紧缩 如果 $f_r \ge f_s$，则使用 x_h 和 x_r 中函数值更低的点计算紧缩点 x_c：

      Outside 如果 $f_s \le f_r < f_h$，计算 $x_c = c + \beta(x_r - c)$ 及其函数值 $f_c = f(x_c)$，其中 $0 < \beta < 1$，通常取为 $\frac{1}{2}$。如果 $f_c \le f_r$，接受 x_c

并终止此次迭代；否则执行压缩变形。

Inside 如果 $f_r \geq f_h$，计算 $x_c = c + \beta(x_h - c)$ 及其函数值 $f_c = f(x_c)$，其中 β 取法如前。如果 $f_c < f_h$，接受 x_c 并终止此次迭代；否则执行压缩变形。

压缩 计算 K 个新顶点 $x_k := x_l + \delta(x_k - x_l)$ 及其函数值 $f_k = f(x_k)$，其中 $0 < \delta < 1$，通常取为 $\frac{1}{2}$； $k = 0, \ldots, K$，且 $k \neq l$。根据以往的经验，实际中需要使用压缩变形的时候非常少见[79]。

循环终止的条件可以借鉴其他优化方法所使用的条件，例如当 $\mathcal{S}$ 的体积变得充分小时，或者 $K+1$ 个顶点上的函数值充分接近时，或者迭代步数达到设定的最大值时，循环终止。

NM 算法其他方面的详细讨论可见 [79] 及其参考文献。

C.2 Automatic Parameter Tuners (APTs)

Töscher[88] 建议了两种简单的自动调节参数方法：APT1 和 APT2。假设模型包含了 K 个参数 $\mathbf{p} = (p_1, \ldots, p_K)$，模型参数取值为 p 时所得到的模型误差记为 $f(p)$（对于 Netflix Prize 问题而言， $f(\cdot)$ 即为 Probe RMSE）。

APT1

算法 C.2 **APT1** 设定算法所需的各种值：最大迭代步数 N_{iter}、允许的最小方差 $\sigma^2_{\min}$ 和误差降低阀值 ϵ；

1. 为参数设定初始值： $p = p^{(0)}$， $f = +\infty$。
2. 重复下面的参数更新过程 N_{iter} 次：
   (a) 从 $\{1, \ldots, K\}$ 中随机选出一个参数指标 k；
   (b) 从正态分布 $\mathrm{N}(\mu, \sigma^2)$ 中抽取出参数 p_k 的更新值 p_k^{new}，其中 $\mu = p_k$，$\sigma^2 = \max\{0.1 \cdot |p_k|, \quad \sigma^2_{\min}\}$；

(c) 如果上面产生的新参数值 p_k^{new} 对应的模型误差 $f^{\text{new}} < f - \epsilon$，则接受此新参数值；否则拒绝此新参数值。

3. 返回最终获得的参数值 p 及其相应的模型误差 f。

APT1 可能会改变参数的符号，所以对于某些有非负约束的参数（如方差），我们可以在算法 C.2 的过程中加入截断。

APT2

算法 C.3 APT2　设定算法所需的各种值：最大迭代步数 N_{iter}、内部循环的次数 N_{inner}、初始搜索指数 $e^{(0)}$（$0 < e^{(0)} < 1$）、压缩系数 ρ（$0 < \rho < 1$）和误差降低阀值 ϵ；

1. 初始化所有模型参数对应的搜索指数 $e_k = e^{(0)}$（$k = 1, \ldots, K$）及初始误差 $f = +\infty$。

2. 初始化 $k = 1$，重复下面的参数更新过程 N_{iter} 次：

   (a) 从 $\{1, 2, \ldots, K, 1, 2, \ldots\}$ 中选出下一个参数指标 $k := k + 1$；

   (b) 重复尝试 N_{inner} 次更新 p_k 以获得更低的误差：

      i. 计算新参数值 $p_k^{\text{new}} := p_k \cdot e_k$，并得到相应的误差 f^{new}；如果 $f^{\text{new}} < f - \epsilon$，则接受此新参数值，此次尝试结束；否则，

      ii. 计算新参数值 $p_k^{\text{new}} := p_k \cdot e_k^{-1}$，并得到相应的误差 f^{new}；如果 $f^{\text{new}} < f - \epsilon$，则接受此新参数值，此次尝试结束；否则，

      iii. 更新相应的搜索指数 $e_k^{\text{new}} = e_i^{\rho}$，此次尝试结束。

3. 返回最终获得的参数值 p 及其相应的模型误差 f。

参考文献

[1] A. Albadvi, M. Shahbazi, A hybrid recommendation technique based on product category attributes, Expert Systems with Applications.

[2] Alibaba.
URL http://www.alibaba.cn

[3] Amazon.
URL http://www.amazon.com

[4] Baidu.
URL http://www.baidu.com

[5] M. Beal, Variational algorithms for approximate bayesian inference, Ph.D. thesis (2003).

[6] R. Bell, Y. Koren, Scalable collaborative filtering with jointly derived neighborhood interpolation weights, in: Proceedings of IEEE International Conference on Data Mining (ICDM'07), 2007.

[7] R. Bell, Y. Koren, C. Volinsky, The bellkor 2008 solution to the netflix prize, Tech. rep. (2007).

[8] R. Bell, Y. Koren, C. Volinsky, The bellkor solution to the netflix prize, Tech. rep. (2007).

[9] R. Bell, Y. Koren, C. Volinsky, Chasing $1,000,000: How we won the netflix progress prize, vol. 18, NO. 2, 2007.

[10] R. M. Bell, Y. Koren, Lessons from the netflix prize challenge, SIGKDD Explorations 9 (2) (2007) 75–79.

[11] R. M. Bell, Y. Koren, Improved neighborhood-based collaborative filtering, in: KDDCup'07, San Jose, California, USA, August 12, 2007.

[12] R. M. Bell, Y. Koren, C. Volinsky, Modeling relationships at multiple scales to improve accuracy of large recommender systems, in: P. Berkhin, R. Caruana, X. Wu (eds.), KDD, ACM, 2007.

[13] BellKor's-Pragmatic-Chaos.
URL http://www2.research.att.com/~volinsky/netflix/bpc.html

[14] J. Bennett, S. Lanning, The netflix prize, in: Proceedings of KDD Cup and Workshop, 2007.

[15] D. Billsus, M. J. Pazzani, Learning collaborative information filters, in: ICML, 1998.

[16] J. S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, 1998.

[17] J.-F. Cai, E. J. Candès, Z. Shen, A singular value thresholding algorithm for matrix completion, SIAM J. on Optimization 20(4).

[18] E. J. Candès, Y. Plan, Matrix completion with noise, in: Proceedings of the IEEE, 2009.

[19] E. J. Candès, B. Recht, Exact matrix completion via convex optimization, Found. of Comput. Math. 9 (2008) 717–772.

[20] E. J. Candès, T. Tao, The power of convex relaxation: Near-optimal matrix completion, CoRR abs/0903.1476.

[21] M. A. Carreira-Perpinan, G. E. Hinton, On contrastive divergence learning, in: 10th international workshop on Artificial Intelligence and Statistics (AISTATS'2005), 2005.

[22] Y. H. Chen, E. I. George, A bayesian model for collaborative filtering, in: Proceeding of 7th-Workshop Artificial Intelligence and Statistics, 1999.

[23] J. Cheng, V. Chu, Y. Wang, Cs229 final report statistical analysis and application of ensemble method on the netflix challenge (2006).

[24] D. M. Chickering, D. Heckerman, C. Meek, A bayesian approach to learning bayesian networks with local structure, in: UAI, 1997.

[25] Y. H. Cho, K. Kim, Application of web usage mining and product taxonomy to collaborative recommendations in e-commerce, Expert Systems with Applications 26 (2004) 233–246.

[26] M. Connor, J. Herlocker, Clustering items for collaborative filtering (2001).

[27] I. S. Dhillon, S. Mallela, D. S. Modha, Information-theoretic co-clustering, in: KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, New York, NY, USA, 2003.

[28] B. Efron, C. Morris, Data analysis using stein's estimator and its generalization, Journal American Statistical Association 70 (1975) 311–319.

[29] C. Elkan, Naive bayesian learning, Tech. rep., No. CS97-557 (1997).

[30] Facebook.
URL http://www.facebook.com

[31] Google.
URL http://www.google.com

[32] Google-News.
URL http://news.google.com

[33] J. Han, M. Kamber, Data Mining: Concepts and Techniques (2nd edition), Morgan Kaufmann Publishers, 2006.

[34] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer Series in Statistics, Springer New York Inc., New York, NY, USA, 2001.

[35] J. Herlocker, J. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in: Proceedings of the 1999 Conference on Researsh and Development in Information Retrieval (SIGIR-99), 1999.

[36] G. E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Comput. 14 (8) (2002) 1771–1800.

[37] G. E. Hinton, Scholarpedia, Boltzmann machine (2007).
URL http://www.scholarpedia.org/article/Boltzmann_machine

[38] T. Hofmann, Probabilistic latent semantic analysis, in: Proceedings of Uncertainty in Artificial Intelligence, UAI'99, 1999.

[39] T. Hong, D. Tsamis, Use of knn for the netflix prize.
URL http://www.stanford.edu/class/cs229/proj2006/HongTsamis-KNNForNetflix.pdf

[40] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, IEEE International Conference on Data Mining 0 (2008) 263–272.

[41] T. Jaakkola, M. Jordan, Bayesian parameter estimation via variational methods, in: Statistics and Computing, 2000.

[42] R. H. Keshavan, A. Montanari, S. Oh, Low-rank matrix completion with noisy observations: a quantitative comparison, CoRR abs/0910.0921.

[43] R. H. Keshavan, A. Montanari, S. Oh, Matrix completion from noisy entries, CoRR abs/0906.2027.

[44] R. H. Keshavan, S. Oh, A. Montanari, Matrix completion from a few entries, CoRR abs/0901.3150.

[45] A. Kohrs, B. Merialdo, Clustering for collaborative filtering applications, in: Computational Intelligence for Modelling, Control & Automation. IOS, Press, 1999.

[46] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Y. Li, B. Liu, S. Sarawagi (eds.), KDD, ACM, 2008.

[47] Y. Koren, Factor in the neighbors: Scalable and accurate collaborative filtering, in: Transactions on Knowledge Discovery from Data (TKDD), 2009.

[48] Y. Koren, The bellkor solution to the netflix grand prize, Tech. rep. (August, 2009).

[49] Y. Koren, R. M. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, IEEE Computer 42 (8) (2009) 30–37.

[50] L. D. Lathauwer, B. D. Moor, J. Vandewalle, A multilinear singular value decomposition, SIAM J. Matrix Anal. Appl. 21 (4) (2000) 1253–1278.

[51] Y. J. Lim, Y. W. Teh, Variational bayesian approach to movie rating prediction, in: Proceedings of KDD Cup and Workshop, 2007.

[52] D. Lin, L. Mackey, Collaborative filtering on a large scale recommender system data set (May 2007).

[53] G. Linden, B. Smith, J. York, Amazon.com recommendations: Item-to-item collaborative filtering, in: IEEE Internet Computing, 2003.

[54] S. Ma, D. Goldfarb, L. Chen, Fixed point and bregman iterative methods for matrix rank minimization, CoRR abs/0905.1643.

[55] B. Marlin, Modeling user rating profiles for collaborative filtering, in: Proceedings of the 17-th Annual Conference on Neural Information Proceeding Systems (NIPS-2003), 2003.

[56] B. Marlin, Collaborative filtering: A machine learning perspective, Master's thesis (2004).

[57] B. Marlin, Missing data problems in machine learning, Ph.D. thesis (2008).

[58] B. J. Mirza, B. J. Keller, N. Ramakrishnan, Studying recommendation algorithms by graph analysis, J. Intell. Inf. Syst. 20 (2) (2003) 131–160.

[59] K. Miyahara, M. J. Pazzani, Collaborative filtering with the simple bayesian classifier, in: PRICAI, 2000.

[60] Netflix.
URL http://www.netflix.com

[61] Netflix, Netflix prize.
URL http://www.netflixprize.com

[62] Netflix, Netflix prize forum.
URL http://www.netflixprize.com//community

[63] Netflix, Netflix prize grandprize.
URL http://www.netflixprize.com//prize?id=1

[64] Netflix, Netflix prize leaderboard.
URL http://www.netflixprize.com//leaderboard

[65] Netflix, Netflix prize rules.
URL http://www.netflixprize.com/rules#quiz

[66] D. Oard, J. Kim, Implicit feedback for recommender systems, in: Proceedings of the AAAI Workshop on Recommender Systems, 1998.

[67] Pandora.
URL http://www.pandora.com

[68] A. Paterek, Improving regularized singular value decomposition for collaborative filtering, in: KDD-Cup and Workshop, ACM press, 2007.

[69] M. Piotte, M. Chabbert, The pragmatic theory solution to the netflix grand prize, Tech. rep., Pragmatic Theory Inc., Canada (August, 2009).

[70] T. Raiko, A. Ilin, J. Karhunen, Principal component analysis for large scale problems with lots of missing values., in: J. N. Kok, J. Koronacki, R. L. de Mántaras, S. Matwin, D. Mladenic, A. Skowron (eds.), ECML, vol. 4701 of Lecture Notes in Computer Science, Springer, 2007.
URL http://dblp.uni-trier.de/db/conf/ecml/ecml2007.html#RaikoIK07

[71] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, J. Riedl, Grouplens: An open architecture for collaborative filtering of networks, in: Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work, ACM, Chapel Hill, North Carolina, 1994.

[72] R. Salakhutdinov, A. Mnih, Bayesian probabilistic matrix factorization using markov chain monte carlo, in: ICML, 2008.

[73] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in: Advances in Neural Information Proceeding Systems, vol. 20, 2008.

[74] R. Salakhutdinov, A. Mnih, G. Hinton, Restricted boltzmann machines for collaborative filtering, in: Proceedings of 24th Annual International Conference on Machine Learning, 2007.

[75] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, Proceedings of 10th World Wide Web (WWW 10) conference, Hong Kong, 2001.

[76] E. Savia, K. Puolamäki, S. Kaski, Latent grouping models for user preference prediction, Mach. Learn. 74 (1) (2009) 75–109.

[77] E. Savia, K. Puolamäki, J. Sinkkonen, S. Kaski, Two-way latent grouping model for user preference prediction, in: Proceedings of the UAI'05, AUAI Press, 2005.

[78] U. Shardanand, P. Maes, Social information filtering: Algorithms for automating "word of mouth", in: CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems, vol. 1, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1995.
URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.3561

[79] S. Singer, J. Nelder, Nelder-mead algorithm (2009).
URL http://www.scholarpedia.org/article/Nelder-Mead_algorithm

[80] P. Smolensky, Information processing in dynamical systems: foundations of harmony theory, in: D. E. Rumelhart, J. L. McClelland (eds.), Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations, MIT Press, Cambridge, MA, USA, 1986.

[81] J.-T. Sun, X. Wang, D. Shen, H.-J. Zeng, Z. Chen, Mining clickthrough data for collaborative web search, in: WWW '06: Proceedings of the 15th international conference on World Wide Web, ACM, New York, NY, USA, 2006.

[82] J.-T. Sun, H.-J. Zeng, H. Liu, Y. Lu, Z. Chen, Cubesvd: a novel approach to personalized web search, in: WWW '05: Proceedings of the 14th international conference on World Wide Web, ACM, New York, NY, USA, 2005.

[83] G. Takács, I. Pilászy, B. Németh, D. Tikk, Major components of the gravity recommendation system, SIGKDD Explor. Newsl. 9 (2) (2007) 80–83.

[84] G. Takács, I. Pilászy, B. Németh, D. Tikk, On the gravity recommendation system, in: Proceedings of KDD Cup Workshop at SIGKDD'07, 13th ACM

International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, 2007.

[85] G. Takács, I. Pilászy, B. Németh, D. Tikk, Investigation of various matrix factorization methods for large recommender systems, in: 2nd Netflix-KDD Workshop, Las Vegas, NV, USA, 2008.

[86] G. Takács, I. Pilászy, B. Németh, D. Tikk, A unified approach of factor models and neighbor based methods for large recommender systems, in: 1th IEEE Workshop on Recommender Systems and Personalized Retrieval, Ostrava, Czech Republic, August 4, 2008.
URL http://www.dirf.org/diwt2008/workshop2.asp

[87] Taobao.
URL http://www.taobao.com

[88] A. Töscher, M. Jahrer, The bigchaos solution to the netflix prize 2008, Tech. rep., Neuer Weg 23, A-8580 Köflach, Austria (November 25, 2008).

[89] A. Töscher, M. Jahrer, R. M. Bell, The bigchaos solution to the netflix grand prize, Tech. rep. (September 5, 2009).

[90] A. Töscher, M. Jahrer, R. Legenstein, Improved neighborhood-based algorithms for large-scale recommender systems, in: 2nd Netflix-KDD Workshop, Las Vegas, NV, USA, 2008.

[91] L. H. Ungar, D. P. Foster, Clustering methods for collaborative filtering, Tech. rep., WS-98-08 (1998).

[92] S. Vucetic, Z. Obradovic, A regression-based approach for scaling-up personalized recommender systems in e-commerce, in: ACM WebKDD 2000 Web, 2000.

[93] B. Webb, Netflix update: Try this at home (2006).
URL http://sifter.org/~simon/journal/20061211.html

[94] M. Welling, M. Rosen-Zvi, G. Hinton, Exponential family harmoniums with an application to information retrieval, in: Advances in Neural Information Processing Systems 17, MIT Press, Cambridge, MA, 2005.

[95] Wikipedia, Boltzmann machine.
URL http://en.wikipedia.org/wiki/Boltzmann_machine

[96] J. Wu, Binomial matrix factorization for discrete collaborative filtering, in: W. Wang, H. Kargupta, S. Ranka, P. S. Yu, X. Wu (eds.), ICDM, IEEE Computer Society, 2009.

[97] J. Wu, T. Li, Graphical models (2008).
URL http://dsec.pku.edu.cn/~jinlong/learningnotes/GM.pdf

[98] J. Wu, T. Li, A modified fuzzy c-means algorithm for collaborative filtering, in: 2nd Netflix-KDD Workshop, Las Vegas, NV, USA, 2008.

[99] Yahoo.
URL http://www.yahoo.com

[100] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, H.-P. Kriegel, Probabilistic memory-based collaborative filtering, Trans. on Knowledge and Data Engineering 16 (1) (2004) 56–69.

[101] Y. Zhou, D. Wilkinson, R. Schreiber, R. Pan, Large-scale parallel collaborative filtering for the netflix prize, Tech. rep., HP Labs, 1501 Page Mill Rd, Palo Alto, CA, 94304.

致　谢

牛顿在 1676 年给友人的信中写道：*“如果说我看的比别人更远，那是因为我站在巨人的肩膀上。”* 如果把这整句话看成训练数据点*“牛顿”*对应的结果，那么理想的机器学习模型对于数据点*“吴金龙”*的预测结果应该是：*“如果说我走的比以前更快更远，那是因为我在北大的海洋里追逐着浪花。”*

百花齐放的春天，绿虫垂挂的夏天，黄叶铺地的秋天，未名凝固的冬天，这是我心中北大校园的写照。可是，北大于我而言，更多的是孜孜不倦的同学，激情四射的老师，以及对知识永恒的渴望和探寻。在这里，我快活得像大海中的一条小鱼，虽然微不足道却可自由游弋。谢谢北大，谢谢你的舞台以及不求回报的奉献。

在北大五年的学习生涯中，我有幸结识了很多的好朋友。你们或许来自于宿舍楼，或许来自于数学院 05 级博士班，或许来自于数学院……。谢谢田瑜、叶盛和中锋，你们是我大学时光的见证者，是我郁闷时候的出气筒，更是亲如兄弟的好友。有你们的地方，我就有家。谢谢李磊，你总是在我烦恼的时候给我安慰，在我觅食的时候免费提供食物。谢谢继伟，你在我们同居的日子里一直维护着宿舍卫生，并总是在我生病的时候为我翻寻药品。谢谢小玉，你总是被我拉着出去转溜，还时不时地被我掠夺口粮。谢谢霄哥，你总是像大哥那样在我没主意的时候为我出谋划策，并亲力亲为。谢谢张硕，你总会跟我讲 BBS 上的各种趣事，在我求助的时候放下自己的事情耐心给我帮助，并最终在你毕业时心甘情愿地被我打劫了近 20 本好书。谢谢新未，你总是让我发觉自己思维中的盲点，并努力为我提供接触不一样世界的机会。谢谢王端，你总是像大姐一样照顾着我，抽空陪我看自己不感兴趣的电影，并在我嘴馋的时候请我吃美食。谢谢煜成，你的微笑总是让我在惆怅时找到欣慰，而你桌上的食物从来都是对我的肚子 open 。谢谢刘健，你的豪爽总是巾帼不让须眉，与你的讨论总会让我收获快乐，这大概就是美女的魅力吧。谢谢丁炎、陶大江、黄晶、朱忍胜、闵斌、李筱光和刘烨，你们

的激情时常使我们的讨论班热火朝天，并且因为你们我经常可以吃到免费而丰盛的晚餐。谢谢澎涛、亚龙、蒋维、玉昭、宋鹏、云华、保滨、高升、严巍、陈刚、大力、伟波、李琨、王涵、林虎、江源、晓峰以及美女张诚、侯琳、罗丹、刘秋霞、朱梅、薛燕和阮志玲，你们总是在我需要帮助的时候伸出援手，陪伴着我度过一个又一个难关。

“传道授业解惑”，我觉得只是北大老师作用的一个真子集。你们还偶尔帮我们改善伙食，操心我们的感情生活，并为我们以后的前途出谋划策。谢谢张平文老师，您严谨的治学态度，对学术始终如一的热情以及宽容待人的胸怀一直感染着我，也为我在前进的道路上竖立起了指向标。谢谢鄂维南老师，您广博的学识使我可以自由选择自己感兴趣的论文题目，而您对学术和应用前沿的精准定位又保证了论文内容的实际应用价值。谢谢刘勇老师、汤华中老师、李若老师、李治平老师、高立老师等北大数学院的老师，你们的教授让我掌握了各种基础知识，同时又了解到了国际上对各种问题的最新研究进展。

第一次见到李铁军老师的时候，是在 2005 年的 summer school 课堂上，那时候您剃了个光头。我当时的感觉是：哦，原来老师也可以理这种发型啊。这个老师还真是“另类”！后来张老师让我找您，我才知道，原来那个“另类”老师竟然以后五年内都将是我的导师，那时一句话在我的头脑里一闪而过：我惨了！

事实证明我真的惨了，只不过是错惨了。进了北大之后，与您的接触变得越来越多。从选课与您的沟通、到上您的“流体力学”，再到和您讨论科研问题等等，逐渐地我就觉得这个特年轻还长得挺帅的老师不仅脾气特好，连课都讲得很棒。板书写的那叫一个工整，估计两百度近似不戴眼镜抄笔记绝对没问题。那时候我背地里一般叫您“小李”，因为怎么看您都不老，叫“老李”实在不合适，可是又没有“中李”之说，所以就剩“小李”了。叫您“小李”还有另一个原因，那就是感觉“小李”叫的很亲切，像亲人一样。

当然，是老虎总还是有“发威”的时候的。我印象中您对我说话最严厉的一次是我没有及时给您回复邮件。您当时在 Princeton 访问，所以那时我们唯一的交流方式就是电邮。不过我当时查信的频率比较低，所以您给我发邮件后经常很久都得不到回复。后来您终于“发威”了，质问我为什么那么久都不查看邮件。

后来我就战战兢兢地装了个 Foxmail，并把收取邮件的时间间隔设置得很短。

等您从 Princeton 访问回来，我的“科研生涯”也就正式开始了。还记得那时候我经常上午去找您讨论，然后中午我们就一起去燕南吃饭，吃完饭后继续讨论。讨论问题时有时候我会坐着看您在黑板上写思路，那时常常想：为什么“小李”懂那么多，他也没比我大多少啊。我到他这个年龄有希望懂这么多吗？那段时间几乎每天都会和您讨论很久的学术，现在想来，估计这就是我当时一直单身的主要原因。后来逐渐觉得叫您“小李”越来越不太合适了，所以就改叫您“老李”了。这倒不是因为您已经老到够得上“老李”的称呼了，而是觉得以您渊博的知识叫您“小李”，忒委屈您了。

说实话，在这五年中我真是做了挺多对不住您的事。比如关于我毕业后的去向问题，您花了很多心思栽培我，希望我以后能成为一个出色的数学家，可是我却执拗地告诉您我以后要成为工程师。您开始一直很不理解，为什么想成为工程师的我要读博士。北大这个林子很大，据说有上百种鸟呢。后来在我逐渐的“开导”下，您也就逐渐理解了我这种想不开的想法了。再比如，因为科研的需要，我得用并行机算程序，您当时特别给我在计算中心开了个户。可是没想到后来那个账户被我折腾得欠了很多钱。刚知道欠了那么多钱的时候我握着鼠标的手都不听使唤，心里净想着：完了，这次估计得把自己卖了。后来我心惊胆颤地拨通您的电话，把自己捅的娄子告诉您。可是您当时连一句责备我的话都没有，只是详细询问了一下具体情况，然后告诉我您会解决这事。类似的荒唐事我还做了很多很多，可是每次您都能包容我的过错，并尽力帮助我改正错误。“老李”，谢谢您一直以来对我的宽容与扶助，真的谢谢。

去年过年回家的时候，我在去火车站的公交上跟珍说我毕业时想送“老李”一幅对联。上联我都想好了，是“亦师亦兄亦友”，可下联我一直不知道该怎么对。后来我们两人讨论了很久，下联也只对出了八成，是“时严时慈时□”，最后那个字我们一直没想到合适的。这个任务我一定要在毕业前完成。我要感谢的，还包括师母以及小家伙李白天。师母不仅当过我的司机，还偶尔兼职我们的保姆，为我们盛饭倒酒，顺便买单。小白天不仅给我们表演相声，还教会了我该怎么保存香蕉。谢谢你们给我带来的快乐。

风筝可以飞得很高很高，可是让它能成为一道风景的关键在于牵着风筝的线。对于我来说，线就是我的家，我的亲人。没有你们，我所做的一切都将毫无意义。还记得我小时候爷爷用篓筐担着我去县城，一筐放我，一筐放石头。到了城里，爷爷总是给我买各种吃的，自己却从来不舍得吃一口。奶奶，您从我出生的那一刻起就一直在为我操心，操心我的身体，操心我的学业，操心我的工作，然后还要操心我娶妻生子。每次您跪着向菩萨祈福时总会说一句话：“保佑我的龙平平安安，健健康康，事事顺利!”您在我生病的时候，总是不顾日夜的守在我的身边。在我考上大学的时候，您和爷爷挑着整担的鞭炮蜡烛，走几十里颠簸的山路，只是为了跟菩萨说一声：谢谢。不论冬天有多冷，周日晚上只要一听到家里的电话响，您总会急切地从被子里爬起来接听电话，目的只是为了确认一下我是不是什么都好。而每次当我在电话里问您和爷爷身体是不是都好时，您总是说都很好，即便有几次您连说话的力气都没有。奶奶，在我心中，您就是菩萨。爸爸，一直以来您都以自己的勤奋和坚忍支撑着整个家。您从来没有对我抱怨过为什么上学要那么多钱，只是在我需要钱的时候默默地为我准备好。您虽然不善言辞，却有一颗爱着我们的深邃的心。妈妈，不论您在果园里做完农活有多累，在您回到家时总是会为我们准备好热乎的饭菜。您在为我们盛好新鲜米饭后，却常常为自己单独盛一碗没吃完的剩饭。妈妈，您为这个家倾注了自己的全部心血。弟弟，对你我更多的是歉意。一直以来我都没有尽过自己作为哥哥的责任，你小的时候我不但不保护你，还常常带头欺负你。长大了后我又基本不在家，家里的爷爷奶奶都由你帮我照顾着。弟弟，谢谢你一直以来为我默默地付出着。姐姐，你是我的英语启蒙老师，更是我的思想启蒙老师。你教我要努力追寻自己的梦想，并用你的身体力行不断证明着这句话。这十几中只要我遇到困难，首先想到的总是你，因为在我心里，你就是信仰。珍，去年年末我有幸把你拉入了自己的生活。从那以后我们共同经历了很多很多，从凌晨五点去买票，到 798 的未来邮局，再到观看*“I see you”*、去派出所温习我们的悲剧……。你带给我的不仅仅是快乐，更重要的是生活。*“生死契阔，与子相悦，执子之手，与子偕老”*。

最后，谢谢所有看到这篇论文的人。正是因为你们，它才具有存在的价值。祝愿所有的你们在以后的生活中一帆风顺，并始终 follow your dreams 。

发表/待发表论文目录

[1] **Jinlong Wu**, Tiejun Li, Image segmentation by using the localized subspace iteration algorithm, Science in China Series A: Mathematics, Vol. 51, No. 8, 2008.

[2] **Jinlong Wu**, Tiejun Li, A Modified Fuzzy C-Means Algorithm For Collaborative Filtering, 2nd Netflix-KDD Workshop, Las Vegas, NV, USA, August 24, 2008.

[3] **Jinlong Wu**, Binomial Matrix Factorization for Discrete Collaborative Filtering, IEEE-ICDM, Miami, Florida, USA, December 06-09, 2009.

[4] Tiejun Li, **Jinlong Wu**, Xiang Peng, Hong Guo, Statistical method for resolving the photon-photoelectron-counting inversion problem, submitted to J. Comp. Phys..

个人简历

基本情况

吴金龙，男，1984 年生于江西省南丰县

电子邮箱：jinlon.wu@gmail.com

教育状况

2001.9-2005.6 本科南开大学数学科学学院，科学与工程计算

2005.9-2010.7 博士研究生北京大学数学科学学院，科学与工程计算
研究课题：应用于推荐系统的基于机器学习的高效协同过滤算法

工作及实习经历

教学实习（北京大学）—— 2007 至 2009 年

- 担任物理科学学院或信息科学学院《高等数学》课程的助教

社团及社会工作—— 2007 至 2008 年

- 2007 年担任北京大学数学科学学院研究生会宣传部副部长
- 2008 年参与了北京奥运会的志愿者工作
- 作为负责人，组队参加了北京大学2008 年的“河合创业基金”评选，和团队成员一起完成创业计划书的书写及最后产品的研发；团队最终以排名第一的成绩获得了此基金的资助（目前此项目在线网站：www.isee007.com）

实习——阿里巴巴，2008 年 11 月至 2009 年 4 月

- 汇总当前有效应用于 Netflix Prize 竞赛中的各种协同过滤推荐算法（主要包括矩阵分解、基于邻居、受限玻尔兹曼机和聚类等几大类），编程实现这些算法并对它们的结果进行灵活组合，最终向 Netflix 提交近百个模型的组合结果以验证其有效性
- 参与其拼音输入法项目，并负责调研其他公司如搜狗、腾讯、谷歌及微软等在拼音输入法领域内所提交的专利，对所有专利涉及的知识与技术点进行归纳总结

所获荣誉

南开大学新电元一等奖学金，2003年

南开大学校级三好学生，2003年

南开大学奖学金，2001-2004年

北京大学光华奖学金（博士），2008年

北京大学河合创业基金，2008年

北京大学学位论文原创性声明和使用授权说明

原创性声明

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

论文作者签名：　　日期：　　年　　月　　日

学位论文使用授权说明

（必须装订在提交学校图书馆的印刷本）

本人完全了解北京大学关于收集、保存、使用学位论文的规定，即：

- 按照学校要求提交学位论文的印刷本和电子版本；
- 学校有权保存学位论文的印刷本和电子版，并提供目录检索与阅览服务，在校园网上提供服务；
- 学校可以采用影印、缩印、数字化或其它复制手段保存论文；
- 因某种特殊原因需要延迟发布学位论文电子版，授权学校□一年/□两年/□三年以后，在校园网上全文发布。

（保密论文在解密后遵守此规定）

论文作者签名：　　导师签名：

日期：　　年　　月　　日