电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

学士学位论文

BACHELOR THESIS

论文题目 多元关系张量分解算法研究

专 业 信息安全

学 号 **2013060204011**

作 者 姓 名 郝立扬

指 导 教 师 王飞 教授

摘 要

对于不完全观测张量的恢复是一个有重要实践意义的问题。有限观测数目条件下的一般张量用有效的算法恢复是一个困难的问题，而成对互动张量由于其简单有效性质，今年来在多元关系数据建模中被广泛关注。

在本文中，研究成对互动张量这一特殊张量形式，并且设计了恢复算法，即使用求解最小化约束条件下的加权核范数的凸问题来解决张量恢复问题；参考奇异值阈值化的思想设计具体算法解决这一问题。并且设计并行算法模型在模拟数据和真实张量数据集上测试和分析，结果表明成对互动张量恢复并行算法，保持了和理论的一致性，并且有较好的恢复效果。

关键词：张量恢复，成对互动张量，奇异值阈值化，并行张量恢复算法

ABSTRACT

The restoration of imperfect observations tensor is an important practical problem. It's a difficult problem to design an effective algorithm to recover a general tensor , which has limited number of observations. Due to simple and effective nature, pairwise interaction tensor in the multi-relational data modeling is widely concerned this year.

In this paper, we design a tensor recovery algorithm for pairwise interaction tesnor, which is a special form of general tensor. The problem of tensor recovery is solved by minimizing the convex problem of weighted kernel norm under constrainted condition, and references to the singular value thresholding to design the algorithm. In addition, we design the parallel algorithm model and test it by simulated and real tensor data sets. The result show that pairwise interaction tensor parallel recovery algorithm has a better recovery effect with the consistency of the theory.

Keywords: tensor recovery, pairwise interaction tensor, singular value thresholding, parallel recovery algorithm

目 录

第一章 绪论

1.1 研究工作的背景和意义

随着科技的进步和社会的发展，以及大数据时代的到来，在学术理论研究和工程实践方面，以及人们日常生活和工作，数据无处不在。因此对于数据的处理和分析，以及如何从复杂，海量的数据中整理分析挖掘出关键的，有意义的信息始终是研究热点问题。随着计算能力和数据量的激增，大量的复杂结构数据和对于数据处理的需求全部浮现出来。例如：化学，医药和食品科学中常用的荧光激发发射数据以及荧光光谱所需要的具有模式“样本 x 激发 x 发射”的三路数据集；在医学和神经科学领域，多通道脑电图数据通常表示为一个 mxn 矩阵，m 是时间样本，n 为电极信号值，看似简单的结构但为了发掘隐藏的脑动力结构，需要考虑脑电信号的频率分量，也要考虑主题和条件这两个拟态。则有”通道 x 时间 x 频率 x 主题 x 条件”的五路数据集；再如社会网络分析以及网络挖掘中，目的是研究和发现社会网络中的隐藏结构，例如提取人与人之间或组织内的沟通模式；同样在人脸识别的多线性图像分析中，数据分析可将脸部建模为人和拍摄角度和光照和表情的形式[1]。上述的数据集形式都是两两之间作用，所以都可归结为多元关系的形式。总之，以上举例中数据均涉及到了高维度的数据以及对其的分析，正是现在数据的高维度以及复杂性为分析工作带来了巨大的挑战。

维数的增加给数据带来的首要问题就是“数据灾难”[1]，即随着维数的增加，学习以及训练的维数增加，学习和训练的所需样本数量呈现出指数级增长，这不仅使很多低维现有的高效算法不再适用，也使得计算的时间复杂度和空间复杂度都呈现出指数级增长。维数的激增不仅对现有理论提出了挑战，也对现有的计算能力和资源空间提出了考验。传统的数据方法多是单路或者双路数据分析，即只能先将高维数据降低到一维的向量或者二维的矩阵形式进行处理和分析，常用方法如主成分分析(principal component analysis, PCA)[1]等进行降维和进一步的处理。这些种分析方法存在不适用高维数据以及破坏数据结构潜在内部结构关系的严重缺点，从而达不到理想的分析效果。

在数据分析，数据挖掘和数据处理中，张量是向量，矩阵等低维数据形式在高维上的拓展所得到的一般形式，可以保持数据内在结构关系和信息，从而刻画蕴含复杂数据关系的事物。正是因为张量的这种特性使得其在高维数据的处理上有极大优势和潜力，张量分解是矩阵奇异值分解在高维度上的拓展，可以剖析高维度数据的同时，尽量保持数据内在构成和关系，是目前高维分析强而有力的工具。

近年来，张量分析的研究以及应用不再局限于最初的心理测量学，数学领域，而是广泛应用于计算机科学，物理学，医学成像，信息分类，图像视频处理，生物制药等各个领域[1]。张量分析为极多领域提供了理论框架，在此思想指导下，数据处理能力和效率大幅增加，提高。但探索的同时也有很多困境，例如高维度数据每一个维度的规模尺寸太大，导致了内存资源消耗巨大，空间复杂度巨大。在上述基础上，为大数据时代提供了计算能力保证的一个重要主流方法就是分布式并行的计算方法。这也是本文在设计多元关系张量恢复分析算法的同时为了提高效率节约资源的解决方法。

1.2 国内外研究现状

张量分解最初由 Hitchcock 在 1927 年提出，而在 1944 年 Cattell 提出了多路数据分析模型，由于当时人们对于数据处理计算的能力和资源太低，理论未广泛能应用在各个领域。直到 20 世纪 60 年代 Tucker 和 1970 年 Carroll, Chang, Hashman 的理论提出和工作才使得前面提到的理论被关注，但是仅仅应用于心理测量学中[1]。在 1981 年 Appellof 和 Davidson 把张量分解第一次用在了化学计量学中，从而引起了研究学者的关注和本领域流行。随着计算能力的提升，在后续的理论关注和发展中张量分解开始活跃于其他领域，例如计算机视觉，数值分析，数据挖掘，图形分析，神经科学，信号处理等等。

在张量分解中，高阶的分解形式有两种：一种侧重于保留核张量的对角形式，另一种侧重于生成的因子矩阵的正交性；从而产生了两种形式：CANDECOMP/PARAFAC(简称 CP 分解)以及 Tucker 分解。CP 分解保持了对角形式，应用于如:低秩近似，转置张量分解，张量分解的稀疏正则，随即压缩立方的并行分解等；而 Tucker 分解保持了因子矩阵的正交性，应用于:分层 Tucker 分解方法，稀疏 Tucker 分解进行低秩估计等[2]。这两种分解方法是张量分解方法的基本形式，目前基本成熟有很拓展但仍然存在局限性。

大数据时代的到来以及日益增长的大规模张量计算需求，但现实研究中仍然和需求有很大的差距[3]。目前对于张量的分解分析方法利用张量的稀疏性，低秩进行研究。例如在 GigaTensor 算法中解决了浮点操作的数目的最小化问题以及中间数据爆炸问题[4]。这是很多年才有研究者取得的突破进展之一。本文侧重的成对影响，或者叫成对互动张量(pairwise interaction tensor)这一概念由 Stenffen Randle 在 2010 的[2]提出，应用与推荐系统中。目前现有成熟理论仍然很少，国内研究者并不多，但是具有很高的理论价值，所以选择这一方面希望能有所成果。

另外，张量链(tensor train)[6]和张量网络(tensor network)[7]在近几年成了关注度

极高的研究热点，在神经网络和深度学习领域的研究热潮中，张量链和张量网络为网络结构提供了很好的数据处理能力，而这一思想的提出又是张量分析在物理学家的理论结合中产生，这也印证了上述的一点，张量分析理论应用变得广泛，研究具有价值和前途以及持续性。

1.3 本文的主要贡献和创新

本文提出了一种并行的，对多元关系张量恢复算法。成对互动张量(Pairwise Interaction Tensor)的分解和恢复算法(Pairwise Interaction Tensor Factorization, PITF)使用传统的贝叶斯概率张量分解(Bayesian Probabilistic Tensor Factorization, BPTF)方法[8]，而本文是而是结合了矩阵奇异值分解思想，使用了奇异值阈值分解(Singular Value Threshold, SVT)这一方法[9]，将 PITF 的思想和奇异值分解结合设计新的求解方法。而且实验结果表明，使用 PITF 的思想提高了效率和恢复精确度，而又解决了 SVT 不能很好处理分析的高维数据。

在大数据时代中，算法的可行性和系统上的具体实现同等重要，因此考虑执行效率将算法加入并行计算这一特性。使得效率和精度之间有着很好的平衡性。并且引入了高效率通信的信息传递接口(Message Protocol Interface, MPI)尝试设计了数据并行和模型并行两种通信泛型，均取得了较好的效果。

1.4 论文结构安排

本文第一章主要介绍了张量分解的背景和研究意义，以及国内外研究现状和趋势。另外介绍了本文的主要工作和贡献以及阐明了创新点和研究基本思想出发点。

第二章介绍基本概念，包括张量概念和张量分解的概念，以及张量分解的基本算法(CP 和 Tucker)两种方法；另外对数据分析领域最优化和凸问题的思想和求解进行阐述，对于矩阵理论中的奇异值分解(SVD)以及在高维上的拓展理论(SVT)进行介绍；针对算法中所要涉及的投影算子和伴随算子等数学空间操作进行具体化的简要的阐述。

第三章介绍文中算法实现并行时所需要的基本理论：通信泛型例如同步异步，并行计算架构参数服务器节点(Parameter Server)和计算节点(Worker)，以及通信时采用的 MPI 接口和 OpenMPI 以及 MPI for Python(mpi4py)的介绍。

第四章介绍张量恢复算法的分析和设计。首先对于恢复可行性进行分析，然后提出了本文的恢复模型和最优化问题设计思想，同时在分析同时改进一步一步得出最终模型。并且设计了无噪声精确恢复算法和有噪声稳定恢复算法，给出两

种算法算法具体流程和细节。

第五章介绍针对算法的设计实现和框架，以及在真实数据集上进行试验的结果。并且对于结果进行阐述和分析，验证理论正确性和结论可靠性。

第六章对全文进行总结和展望,并且介绍本文的后续工作以及自己未来的工作。

第二章 数学背景知识

2.1 张量和张量分解

2.1.1 张量的基本概念

数据沿以相同水平方向的排列称为一路阵列，数学上的张量专指多路阵列，矩阵用其元素和矩阵符号$[\cdot]$表示为$A=\left[a_{ij}\right]_{i,j=1}^{m,n}$，类似的 n 阶张量也可表示$\chi=\left[x_{i_1\ldots i_n}\right]_{i_1,\ldots,i_n}^{I_1,\ldots,I_n}$，其中是张量的$x_{i_1,\ldots,i_n}$第$(i_1,i_2,\ldots,i_n)$元素。所有$I_1\times I_2\times\ldots\times I_n$维张量的集合记为$\chi(I_1\times I_2\times\ldots\times I_n)$。通俗地讲，张量是一个多维数组。最常用的张量为三阶张量[1]。图 2-1-1 画出了三阶张量。

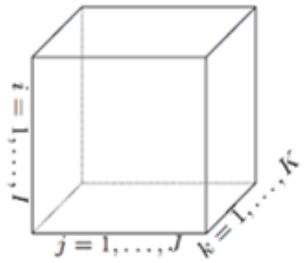


图 2-1-1 三阶张量$\chi^{I_1\times I_2\times I_3}$示意图

三阶张量的三路阵列不以行向量，列向量等相称，而改为张量纤维(tensor fiber)。纤维是保留一个下标可变，固定其他所有下标不变得到的一路阵列。分别称为水平纤维(row fiber, horizontal fiber)，竖直纤维(column fiber, vertical fiber)和纵深纤维(tube fiber, depth fiber)[2]。可分别用符号$x_{:jk}$，$x_{i:k}$，$x_{ij:}$表示。如图 2-1-2(a)~(c)：

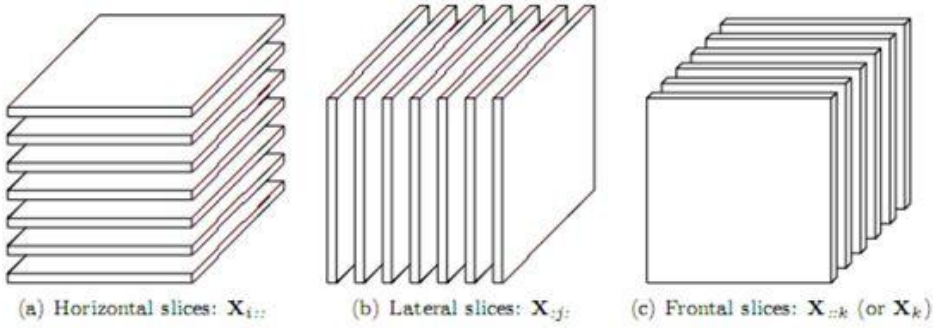


图 2-1-2 :三阶张量的三种纤维

定义 2.1.1 N阶张量$\chi\in K^{I_1\times I_2\times\ldots\times I_N}$的模式-n向量是一个以$i_n$为元素下标变量，而其他下标$\{i_1,\ldots,i_N\}\backslash i_n$全部被固定不变的$I_n$维向量，用符号记做$X_{i_1\ldots i_{n-1}:i_{n+1}\ldots i_N}$。

高阶张量也可以用矩阵的集合表示。这些矩阵形成了三阶张量的水平切片(horizontal slices)，侧向切片(lateral slices)和正面切片(frontal slices)，可用矩阵符号 $X_{i::}$， $X_{:j:}$ 和 $X_{::k}$ 表示[2]。如图 2-1-3(a)~(c)所示：

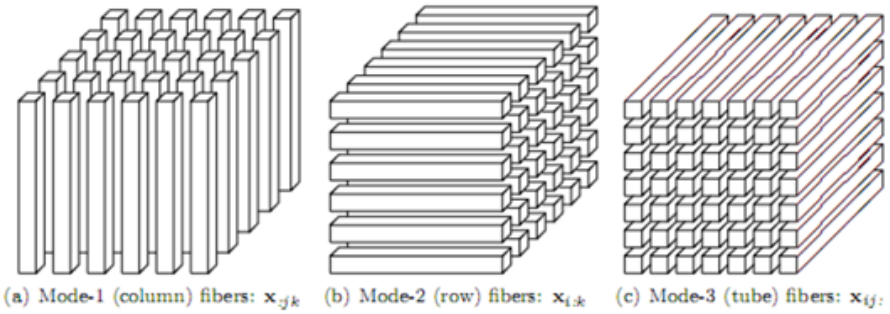


图 2-1-3：三阶张量的三种切片

张量的二范数和矩阵一样，也都可以用 F 范数表示。

$$\|x\|_2 = (|x_1|^2 + |x_2|^2 + ... + |x_m|^2)^{\frac{1}{2}} \tag{2-1}$$

Frobenius 范数在数学上对的定义如下公式所示：

$$\|\chi\|_F = \sqrt{\sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2}\cdots\sum_{i_N=1}^{I_N} x_{i_1 i_2 \ldots i_N}^2} \tag{2-2}$$

一个张量的 F 范数是其所有元素平方和的算术平方根，可以用如下公式表示：

因为矩阵是张量的二阶形式，所以高阶张量可以展开矩阵化。

三阶张量有 I 个水平切片矩阵，J 个侧向切片矩阵和 K 个正面切片矩阵。然而，在张量的分析和计算中，常用一个矩阵代表一个三阶张量。此时，需要一种运算，将三阶张量(三路阵列)经过重新组织和排列变成一个矩阵(二路阵列)[2]。

定义 2.1.2 将一个三路张量(三路阵列)重新组织成一个矩阵形式的变换称为张量的矩阵化(matricization)张量的矩阵化也称作张量的展开(unfolding)。对于某一维度的展开称为模式 n 展开(mode-n unfolding)，记为 $\chi_{(n)}$。例如，一个张量第 $(i_{1,}i_2...i_N)$ 个元素映射到矩阵的第 (i_n, j) 个元素，其中：

$$j = 1 + \sum_{\substack{k=1 \\ k\neq n}}^{N} (i_k - 1) J_k, J_k = \prod_{\substack{m=1 \\ m\neq n}}^{k-1} I_m \tag{2-3}$$

张量也可以用模式-n 的形式展开[2]。

虽然张量在符号和标记上比较复杂，但是可以和矩阵以及向量做乘法运算。一个张量 $\chi \in R^{I_1 \times I_2 \times ... \times I_N}$ 和一个矩阵 $U \in R^{J \times I_n}$ 的模式 n 乘积，记为 $\chi \times_n U$ ，规模为

$I_1 \times I_2 \times ... \times J \times I_{n+1} \times I_n$。对于每个元素，有$(\chi \times_n U)_{i_1 ... i_{n-1} j i_{n+1} ... i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 ... i_N} u_{j i_n}$。相当于用 U 乘以张量$\chi$的每个模式 n 纤维。这也可以用张量的模式 n 展开成矩阵形式表示：

$$y = \chi \times_n U \Leftrightarrow Y_{(n)} = UX_{(n)}$$

可以观察到，当有一系列乘法时，对于不同的模式，乘法的顺序是无影响的：$\chi \times_m A \times_n B = \chi \times_n A \times_m B (m \neq n)$，但是对于相同模式，则有$\chi \times_n A \times_n B = X \times_n (BA)$。

另外介绍张量的三种重要运算。

定义 2.1.5　Kronecker 积

矩阵$A \in R^{I \times J}$和$B \in R^{K \times L}$Kronecker 积，记为$A \otimes B$，它是一个的矩阵，定义如下$(IK) \times (JL)$的矩阵:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & ... & a_{1J}B \\ a_{12}B & a_{22}B & ... & a_{2J}B \\ ... & ... & ... & ... \\ a_{I1}B & a_{I2}B & ... & a_{IJ}B \end{bmatrix} \tag{2-4}$$

定义 2.1.6　Khatri-Rao 积

矩阵的 Khatri-Rao 积可看作是 Kronecker 积的列匹配。给定矩阵和矩阵，则它们的 Khatri-Rao 积，记为$A \odot B = \begin{bmatrix} a_1 \otimes b_1 & a_2 \otimes b_2 & \cdots & a_K \otimes b_K \end{bmatrix}$，是一个$(IJ) \times K$的矩阵。

定义 2.1.7 Hadamard 积：

矩阵的 Hadamard 积是矩阵的对应元素的乘积。对于两个都是$I \times J$的矩阵 A 和 B，它们的 Hadamard 积也是$I \times J$的，记为$A * B$。

$$A * B = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & ... & a_{1J}b_{1J} \\ a_{12}b_{12} & a_{22}b_{22} & ... & a_{2J}b_{2J} \\ ... & ... & ... & ... \\ a_{I1}b_{I1} & a_{I2}b_{I2} & ... & a_{IJ}b_{IJ} \end{bmatrix} \tag{2-5}$$

2.1.2 张量分解

2.1.2.1CANDECOMP/PARAFAC(CP 分解)

CP 分解全称 CANDECOMP/PARAFAC 分解，是一个张量分解成若干秩 1 张量的和的形式。例如：给定一个三阶张量$\chi \in R^{I \times J \times K}$，可写为下列形式：

$$\chi \approx \sum_{r=1}^{R} a_r \circ b_r \circ c_r \tag{2-6}$$

其中，R 是一个正整数，$a \in R^I$，$b \in R^J$，$c \in R^K$，$r = 1,...,R$。针对元素而言上式又可以写成 $x_{ijk} \approx \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}$，其中 $i = 1,...,I$，$j = 1,...,J$，$k = 1,...,K$

图解形式如图 2-1-6 所示：

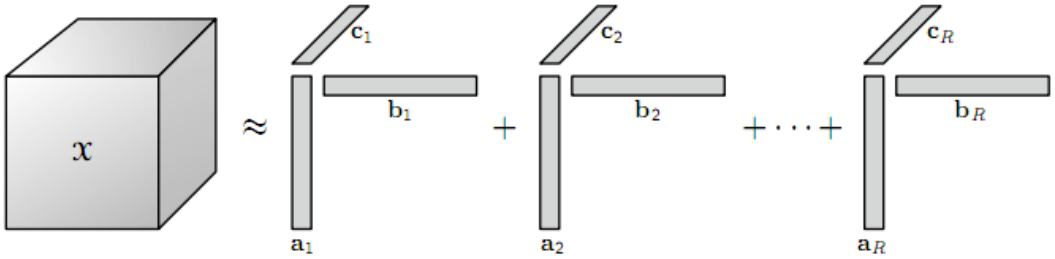


图 2-1-6 :三阶张量的 CP 分解

这些秩 1 张量中的对应向量组合起来构成了因子矩阵,所以 CP 分解可以用另一种形式表示:

$$\mathbf{X}_{(2)} \approx \mathbf{B}(\mathbf{C} \otimes \mathbf{A})^{\mathrm{T}}$$

也可以用正面切片的形式表现：

$$X_k \approx AD^{(k)}B^T$$

其中 $D^{(k)} \equiv diag(c_{k:})$，$k = 1,...,K$。同理也可以写成水平或侧面切片的形式。

通常情况下，三阶张量是应用最广泛最直观最有效的模型之一。对于一般的 N 阶张量 $\chi \in R^{I_1 \times I_2 \times ... \times I_N}$ 的 CP 分解形式：

$$\chi \approx \sum_{r=1}^{R} \alpha_r a_r^{(1)} \circ a_r^{(2)} \circ ... \circ a_r^{(N)} \tag{2-7}$$

其中 $a^{(n)} \in R^{I_n}$，$n = 1,...,N$。它的因子矩阵的形式为：

$$\boldsymbol{X}_{(n)} \approx \boldsymbol{A}^{(n)} \boldsymbol{\Lambda} (\boldsymbol{A}^{(N)} \odot \cdots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \cdots \odot \boldsymbol{A}^{(1)})^{\mathrm{T}}$$

其中 $\Lambda = diag(\alpha)$。

2.1.2.2Tucker 分解

Tucker 分解是高阶主成份分析法(HOPCA)的一种形式[1],它是将张量分解为一个核心张量 g 和沿 g 的每一个方向上的因子矩阵。因此，对于一个三阶张量 $\chi \in R^{I \times J \times K}$，我们有

$$\chi \approx g \times_1 A \times_2 B \times_3 C = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} a_p \circ b_q \circ c_r \tag{2-8}$$

这里 $A \in R^{I \times P}$，$B \in R^{J \times Q}$ 和 $C \in R^{K \times R}$ 是因子矩阵，通常是正交的，可认为是每个方向上的主成分。张量 $g \in R^{P \times Q \times R}$ 称为核心张量，它的数值表示了不同成分之间

相互关联的程度。

从元素角度考虑，上式可以写成

$$x_{ijk} \approx \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} g_{pqr} a_{ip} b_{jq} c_{kr}, i=1,...,I; j=1,...,J; k=1,...,K \tag{2-9}$$

其中 P，Q，R 分别是 A，B，C 的主成分的数目，如果 P，Q，R 比 I，J，K 小，则核张量 g 是 χ 的压缩版本，所以在有些时候，分解后的张量的存储量明显小于原张量。Tucker 分解的图解形式如图 2-1-7 所示。

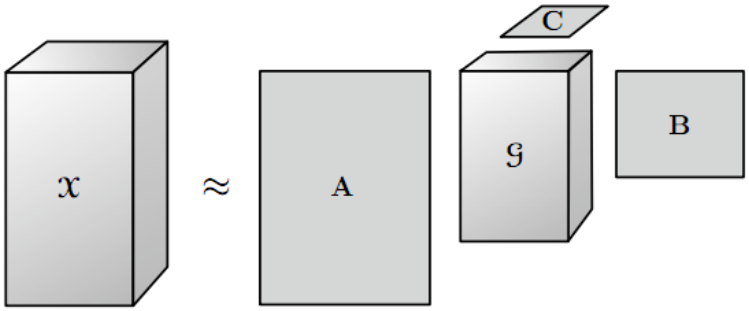


图 2-1-7:三阶张量的 Tucker 分解

我们很容易发现，CP 分解可以看作是 Tucker 分解的一种特殊形式，只要满足核张量是超对角的，并且 P=Q=R 即可。然是 Tucker 分解的大多数拟合算法都假设因子矩阵的列是正交且归一化的，而 CP 分解就不要求因子矩阵的列正交且归一化。Tucker 分解表示为因子矩阵形式为:

$$\begin{aligned} X_{(1)} &= AG_{(1)}(C \otimes B)^T \\ X_{(2)} &= AG_{(2)}(C \otimes A)^T \\ X_{(3)} &= AG_{(3)}(B \otimes A)^T \end{aligned} \tag{2-10}$$

Tucker 分解也可以推广到高阶张量的形式： $\chi = g \times_1 A^{(1)} \times_2 A^{(2)} \times ... \times_N A^{(N)}$ ， 也可写成元素的形式：

$$x_{ijk} \approx \sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}...\sum_{r_N=1}^{R_N} g_{r_1 r_2 ... r_N} a^{(1)}_{i_1 r_1} a^{(2)}_{i_2 r_2} ... a^{(N)}_{i_N r_N} \tag{2-11}$$

其中 $i_n = 1,...,I_n \quad n = 1,...,N$ 。其因子矩阵的形式为：

$$X_{(n)} = A^{(n)} G_{(n)} (A^{(N)} \otimes ... \otimes A^{(n+1)} \otimes A^{(n-1)} \otimes ... \otimes A^{(1)})^T$$

2.2 最优化问题

2.2.1 数学表述

定义 2. 2. 1 (最优化问题) 具有以下形式的问题称为最优化问题，给定一个函数 $f:A\to R^n$，寻找一个元素 $x^*\in A$，使得对于所有 A 中的 x，有：$f(x^*)\le f(x)$；或 $f(x^*)\ge f(x)$。符号化表示为[10]:

$$\arg\min_{x^*\in A} f(x) \tag{2-12}$$

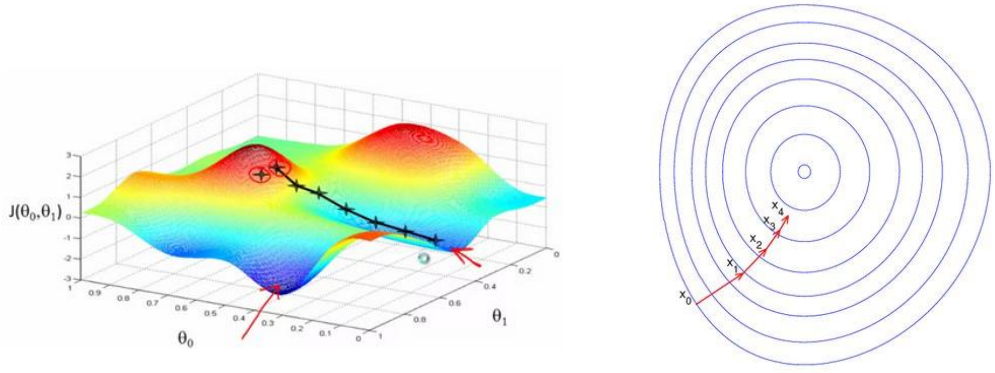
2.2.2 求解最优化问题的一般方法

求解最优化的方法有很多，其中最常用的方法有：1)经典的梯度下降(Gradient Descent)方法；2)对偶上升算法(Dual Ascent)。

2.2.2.1 梯度下降算法(Gradient Descent，GD)

梯度下降算法是最早最简单，也是最常用的最优化方法[11]，梯度下降方法实现简单，当目标函数是凸函数时，梯度下降的解是全局最优解;然而当目标函数是非凸函数时，只能解得一个局部最优解。

梯度下降法的优化思想是当前位置负梯度方向作为搜索方向，负梯度方向当然是最快的下降方向，而且越接近目标值，步长越小，梯度改变越小，我们经常使用迭代算法来实现 GD 算法，示意图如 2-2-2-1:(a)-(b)



(a) (b)

图 2-2-2-1:MATLAB 仿真示意图和梯度下降示意图

当训练数据较大时，可能导致迭代速度过慢而影响效率。为了解决这个问题引入了随即梯度下降方法。例如对于一个线性回归模型，假设 f(x)是拟合函数，$L(\theta)$是损失函数，m 是训练集样本个数，n 是特征个数。

对θ求偏导，按照每个θ的梯度负方向更新参数:

$$\theta_j^{`} = \theta_j + \frac{1}{m}\sum_{i=1}^{m}(y^i - f_\theta(x^i))x_j^i$$

$$f_\theta(x) = \sum_{j=0}^{n}\theta_j x_j, L(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(y^i - f_\theta(x^i))^2 \quad (2\text{-}7)$$

随机梯度下降每次只根据一个训练样本即可更新参数θ：

$$\theta_j^{`} = \theta_j + (y^i - f_\theta(x^i))x_j^i$$

2.2.2.2 对偶上升算法(Dual Ascent)

对偶上升发的核心思想是引入一个对偶变量。一个凸函数的对偶函数是原凸函数的一个下界，可以证明有一个较好的性质:在强对偶假设下，即最小化原始问题(primal problem)等价于最大化对偶问题(dual problem)，两者会同时达到最优[10]。对偶的转化可以将原来许多参数约束条件改变，可以把难以求解的问题转化为可以求解的问题。

原始问题如下：

$$\begin{aligned}&\min\ f(x)\\ &s.t.\ \text{Ax=b}\end{aligned} \quad (2\text{-}8)$$

对偶问题是：

$$g(y) = \inf_x L(x, y)$$

在强对偶假设下，原始和对偶问题的最优解是$x^* = \arg\min_x L(x, y^*)$

由此有迭代公式:

$$\begin{aligned}x^{k+1} &\leftarrow \arg\min_x(x, y^k)\\ y^{k+1} &\leftarrow y^k + \alpha^k \nabla g(y)\end{aligned} \quad (2\text{-}9)$$

其中$\nabla g(y) = Ax^{k+1} - b$。

上述对偶问题和原始问题的转化，在数学上要求函数严格凸，并且步长α的选择也要合适。

2.3 奇异值分析

2.3.1 奇异值分解(singular value decomposition, SVD)

奇异值分解是现代数值分析和数值计算最基本和最重要的工具之一，由Beltrami 于 1873 年对实正方矩阵提出来的[12]。Beltrami 从双线性函 $f(x, y) = x^T Ay$，$A \in R^{n\times n}$ 出发，通过引入线性变化 $x = U\xi$，$y = V\eta$，将双线性函数变为

$f(x,y)=\xi^T S\eta$，其中$S=U^T AV$。

如果约束 U 和 V 为正交矩阵，则它们的选择各存在n^2-n个自由度，利用这些自由度使矩阵 S 的对角线意外元素全为 0，即矩阵$S=diag(\sigma_1,\sigma_2,...,\sigma_n)$为对角矩阵。由 U 和 V 的正交性，左右分别乘$U$和$V^T$，可得到$A=U\Sigma V^T$。在 1874 年 Jordan 也独立提出了实方矩阵的奇异值分解；1902 年 Autonne 将奇异值分解推广到复正方矩阵，Eckart 与 Young 与 1939 年又进一步推广到了一般的复长方形矩阵[13]。

定理 2.3.1(矩阵的奇异值分解) 令$A\in R^{m\times n}$，则存在正交(或酉)矩阵$U\in R^{m\times m}$和$V\in R^{n\times n}$使得

$$A=U\Sigma V^T$$

式中：

$$\Sigma=\begin{bmatrix}\Sigma_1 & O\\ O & O\end{bmatrix} \tag{2-10}$$

且$\Sigma_1=diag(\sigma_1,\sigma_2,...,\sigma_r)$，其对角线元素按照顺序$\sigma_1\geq\sigma_2\geq...\geq\sigma_r>0$，$r=rank(A)$，数值$\sigma_1,\sigma_2,...,\sigma_r$连同$\sigma_{r+1}=\sigma_{r+2}=...=\sigma_n=0$叫做矩阵 A 的奇异值。

2.3.2 奇异值阈值化法(singular value thresholding ,SVT)

考虑低秩矩阵$Y\in R^{n_1\times n_2}$的截尾奇异值分解

$$Y=U\Sigma V^T,\Sigma=diag(\sigma_1,\sigma_2,...,\sigma_r)$$

式中$r=rank(Y)<<\min\{n_1,n_2\}$，$U\in R^{n_1\times r}$，$V\in R^{n_2\times r}$。

令$\tau\geq 0$，则$D_\tau(Y)=UD_\tau(\Sigma)V^T$称为矩阵 Y 的奇异值阈值化(singular value thresholding ,SVT),其中$D_\tau(\Sigma)=diag((\sigma_1-\tau)_+,...,(\sigma_r-\tau)_+)$为矩阵的软阈值化，并且

$$(\sigma_i-\tau)_+=\begin{cases}\sigma_i-\tau\\ 0\end{cases} \tag{2-11}$$

为软阈值运算。奇异值阈值化与奇异值分解的关系如下：若阈值为 0，则奇异值阈值化退化为截尾奇异值分解。所有奇异值以常数$\tau>0$进行阈值运算，并不改变左和右奇异值向量矩阵 U 和 V，只是改变了奇异值的大小。恰当的选择阈值τ，能够有效的将部分奇异值向零收缩，在这个意义上，又称奇异值阈值化这一变换为奇异值阈值收缩算子(singular value shrinkage operator)[13]。需要注意，如果阈值比τ比大多数奇异值大，则奇异值阈值化算子$D(Y)$的秩将比原矩阵 Y 的秩小很多。奇异值阈值化的关键是如何选择软阈值τ，下面给出定理。

定理 2.3.2 对于每一个软阈值$\tau\geq 0$和矩阵$Y\in R^{n_1\times n_2}$，奇异值收缩算子式服从

$$D_\tau(Y)=\arg\min_X\left\{\frac{1}{2}\|X-Y\|_F^2+\tau\|X\|_*\right\} \tag{2-12}$$

对于无约束问题 $\min_A \tau\|A\|_*+\frac{1}{2}\|A-D\|_F^2$ 由于目标函数 $\|A\|_*$ 和 $\frac{1}{2}\|A-D\|_F^2$ 分别是严格凸函数，所以上述无约束问题存在唯一最优解，并由已知数据矩阵 D 的奇异值阈值化直接给出 $\hat{A}=D_\tau(D)=UD_\tau(\Sigma)V^H$。由此对于无约束问题主要是考虑如何转变为上叙式的规范形式。下面举例说明奇异值阈值化如何的应用。

2.3.3 仿射约束的矩阵完备问题

对于仿射约束的矩阵完备问题，由于拉格朗日(Lagrange)函数有：

$$\begin{gathered}\min_X \tau\|X\|_*+\frac{1}{2}\|X\|_F^2\\ s.t.\quad A(X)=b\\ L(X,\lambda)=\tau\|X\|_*+\frac{1}{2}\|X\|_F^2+<\lambda,b-A(X)>\end{gathered} \tag{2-13}$$

所以迭代序列为：

$$\begin{cases}X_k=D_\tau(A^*(\lambda_{k-1}))\\ \lambda_k=\lambda_{k-1}+\mu(b-A(X_k))\end{cases} \tag{2-14}$$

其中 A^* 是满足 $A^*A=I$ 的仿射变换 A 的伴随算子。

Cai 针对奇异值阈值化的实现，提出了无需进行奇异值分解的快速奇异值阈值化方法[9]。数据矩阵 D 的奇异值分解 $D=U\Sigma V^T$ 可以分解为两部分之和：

$$D=U\begin{bmatrix}(\sigma_1-\tau)_+ & & 0\\ & \cdots & \\ 0 & & (\sigma_r-\tau)_+\end{bmatrix}V^T+U\begin{bmatrix}\min\{\sigma_1,\tau\} & & 0\\ & \cdots & \\ 0 & & \min\{\sigma_r,\tau\}\end{bmatrix}V^T \tag{2-15}$$

或者写作 $D=D_\tau(D)+P_\tau(D)$,式中:

$$P_\tau(D)=U\begin{bmatrix}\min\{\sigma_1,\tau\} & & 0\\ & \cdots & \\ 0 & & \min\{\sigma_r,\tau\}\end{bmatrix}V^T \tag{2-16}$$

表示数据矩阵 D 到 2-范数球的投影。

在本文设计的算法中，重要的核心步骤则是参考了 SVT 算法进行改进，在后续算法设计章节中会详细阐释。

2.4 投影算子和伴随算子

2.4.1 投影算子的基本概念

定义 2.4.1（投影算子）考虑向量空间的直和分解 $C^n = S \oplus H$ 内任意向量 $x \in C^n$。若 $x = x_1 + x_2$ 满足 $x_1 \in S$ 和 $x_2 \in H$，并且 x_1 和 x_2 是唯一确定的，则称映射 $Px = x_1$ 是向量沿着子空间H的方向,到了子空间S的投影,并称P是沿着H的方向，到 S 的投影算子，常简记为 $P_{S|H}$。[10]

定义 2.4.2 (正交投影算子) 映射 $P^{\perp} = I - P$ 称为 P 的正交投影算子(orthogonal projector)，若 P 不仅是幂等矩阵，而且还是埃尔米特(Hermitian)矩阵。

2.4.2 锥投影

如果对于任意 $x \in C$ 和 $\theta \geq 0$ 都有 $\theta x \in C$，我们称集合 C 是锥或者非负其次。如果集合 C 是锥，并且是凸的，则称 C 为凸锥，即对于任意的 $x_1, x_2 \in C$ 和 $\theta_1, \theta_2 \geq 0$ 都有：

$$\theta_1 x_1 + \theta_2 x_2 \in C$$

在几何上，具有此类形式的点构成了二维的扇形，这个扇形以 0 为顶点，边通过 x_1 和 x_2。

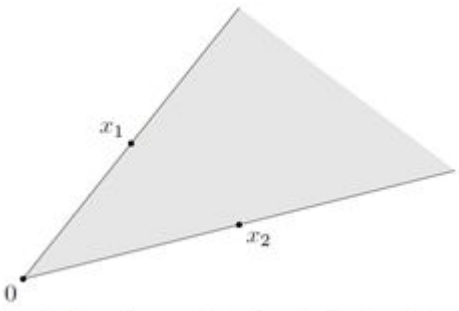


图 2-4-2：锥示意图，扇形显示了所有具有形式 $\theta_1 x_1 + \theta_2 x_2$ 的点

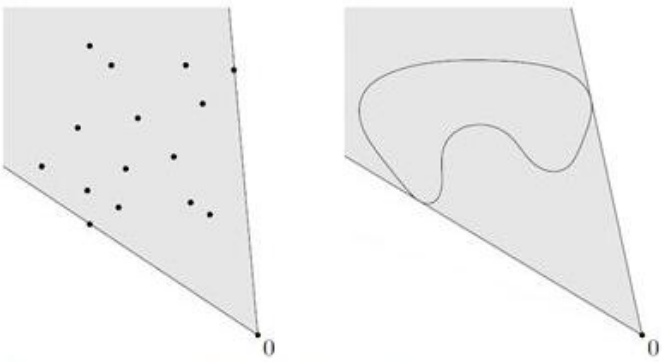


图 2-4-3: 两个集合的锥包示意图(阴影所示)

如图 2-4-2 所示。具有 $\theta_1 x_1 + \theta_2 x_2 + ... + \theta_k x_k$，$\theta_1, ..., \theta_k \geq 0$ 形式的点称为 $x_1, ..., x_k$ 的锥组合或非负线性组合。如果 x_i 均属于凸锥 C，那么 x_i 的每一个锥组合也在 C 中。另外，集合 C 是凸锥的充要条件是它包含其元素的所有锥组合[10]。如同凸组合一样，锥组合的概念也可以拓展。集合 C 的锥包是 C 中元素的所有锥组合的集合：

$$\{\theta_1 x_1 + \theta_2 x_2 + ... + \theta_k x_k \mid x_i \in C, \theta_i \geq 0, i = 1, ..., k\}$$

同时它也是包含 C 的最小的凸锥，如图 2-4-3 所示。

在多元关系张量分解算法设计中，针对有噪声张量恢复算法设计的时候，为了在算法迭代的时候同时更新采样向量和引入的噪声向量，并且要满足约束条件，故需要引入二阶锥的概念。另外，二阶锥是 Euclid 范数定义的范数锥[10]。

$$\begin{aligned} C &= \{(x,t) \in R^{n+1} \mid \|x\|_2 \leq t\} \\ &= \{\begin{bmatrix} x \\ t \end{bmatrix} \mid \begin{bmatrix} x \\ t \end{bmatrix}^T \begin{bmatrix} I & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ t \end{bmatrix} \leq 0, t \geq 0\} \end{aligned} \tag{2-17}$$

2.4.3 伴随算子(adjoint operator)

考虑在希尔伯特空间之间的一个线性算子 $A: H_1 \to H_2$，在不考虑任何细节的情况下，伴随运算符是（在大多数情况下唯一定义的）线性运算符:

$$A^*: H_2 \to H_1$$

且满足 $\langle Ah_1, h_2 \rangle_{H_2} = \langle h_1, A^* h_2 \rangle_{H1}$，这里 $\langle \cdot \rangle_{H_i}$ 表示希尔伯特空间 H_i 上的内积；注意两个希尔伯特空间相同的特殊情况，A 是希尔伯特空间上的运算符。可以定义运算符 $A: E \to F$ 的伴随，其中 E，F 是具有相应规范的的 Banach 空间，这里其伴随运算符被定义为：

$$\begin{aligned} (A^* f)(u) &= f(Au) \\ f &\in F^*, u \in E \end{aligned} \tag{2-18}$$

其中 $A^* f = (u \to f(Au))$。

Hilbert 空间设置中的伴随算子定义实际上只是 Banach 空间的一个应用特例，当它用双重(Dual)标识 Hilbert 空间时。那么我们也可以获得一个运算符 $A: H \to E$，H 是希尔伯特空间，E 是 Banach 空间，双重定义为 $\langle h_f, h \rangle_H = f(Au)$。

在多元关系张量分解算法设计中，在核心算法部分，数据采样和奇异值阈值收缩算子传入的参数时，需要将采样向量映射为矩阵，在此使用了伴随算子这一概念[15]。

2.4.4 仿射函数

函数 $f:R^n \to R^m$ 是仿射的，如果它是一个线性函数和一个常数的和，即具有 $f(x)=Ax+b$ 的形式，其中 $A \in R^{m\times n}$， $b \in R^m$ 。在几何上定义为两个向量空间之间的一个仿射变换或者仿射映射由一个非奇异的线性变换，运用一次函数进行的变换，接上一个平移变换组成[10]。举例如图 2-4-4 所示。

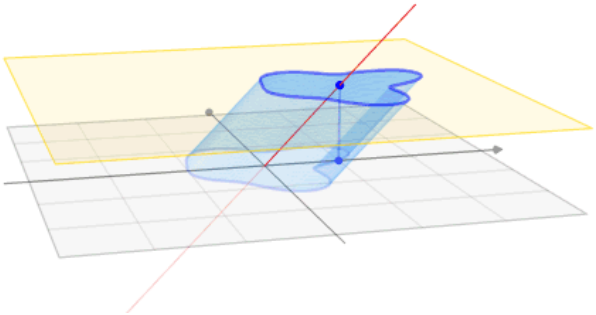


图 2-4-4：仿射变换在三维平移上的示意图

2.5 本章小结

本章主要是介绍数学上的基本概念，包括张量概念和张量分解的概念，以及张量分解的基本算法(CP 和 Tucker)两种方法；另外对数据分析领域最优化和凸问题的思想和求解进行阐述，对于矩阵理论中的奇异值分解(SVD)以及在高维上的拓展理论(SVT)进行介绍；针对算法中所要涉及的投影算子和伴随算子等数学空间操作进行具体化的简要的阐述。以及对于算法设计中如何应用基本概念做了简要说明，详细说明在第四章中具体阐述。

第三章 通信泛型和并行计算

3.1 通信泛型

3.1.1 基本概念

通信毫无疑问是分布式系统以及多任务多进程并行的核心之一，从操作系统的层面来看，通信主要是进程间通信(Inter-Process Communication, IPC)。不同的进程为了能够通信，它们之间必须有一套协商好的规则，用于指定如何通信、消息的格式和语义。这种规则就是我们常说的“协议”。例如，互联网使用最为广泛的协议是 TCP/IP 协议。同时，为了控制通信系统的复杂性，通信系统的构建以及对应的协议通常都是分层的。从另外一个角度看，通信需要有理论模型支持，常用的通信和计算泛型有同步泛型和异步泛型，以及部分同步泛型。

3.1.2 同步(synchronization)泛型

对于迭代计算任务，并发进程之间需要数据同步，则设计的通信模型叫做同步泛型。在严格的同步泛型中，并发程序每一轮迭代都需要在相互之间进行数据同步，如图 3-1-2 分割不同的程序在各个迭代阶段的竖线就是同步点。在 MPI 中使用了函数 Barrier()来控制同步。常用的模型有整体同步并行计算模型(Bulk Synchronous Parallel Computing Model, BSP)[16]。

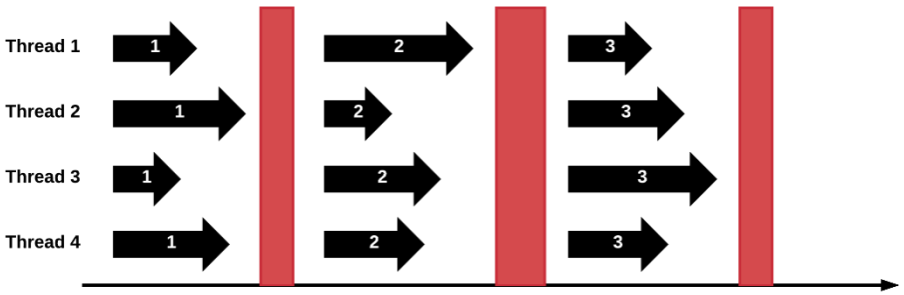


图 3-1-2: 同步泛型示意图(竖线代表同步的时间点)

3.1.3 异步(asynchronous)泛型

异步泛型则和同步泛型不同，在这种通信模型中不需要任何同步过程。任意时刻每个 Thread 都可以读取全局参数，也可以更新全局参数，这样可以使得资源利用效率高，整体的速度比同步泛型快(没有强制线程同步不用等待)。但是正是因为没有约束，如果 Thread 之间由于各种原因导致更新速度差异极大，不仅仅会导致最终结果不正确，而且可能因为某一 Thread 的迭代次数落后，而产生不收敛现象。如图 3-1-3 所示。

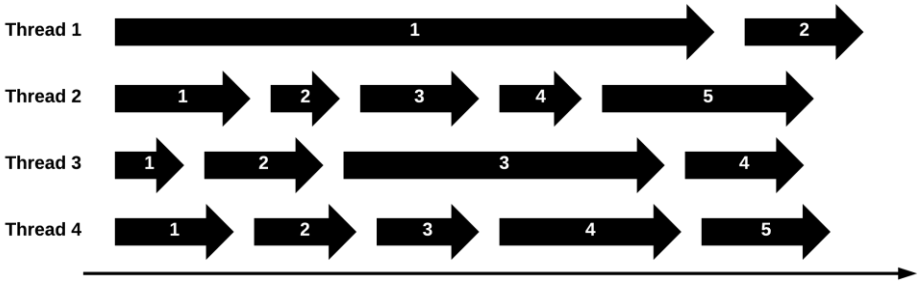


图 3-1-3: 异步泛型示意图

3.2 并行计算(Parallel Computing)架构和技术

3.2.1 并行计算架构和分布式计算架构

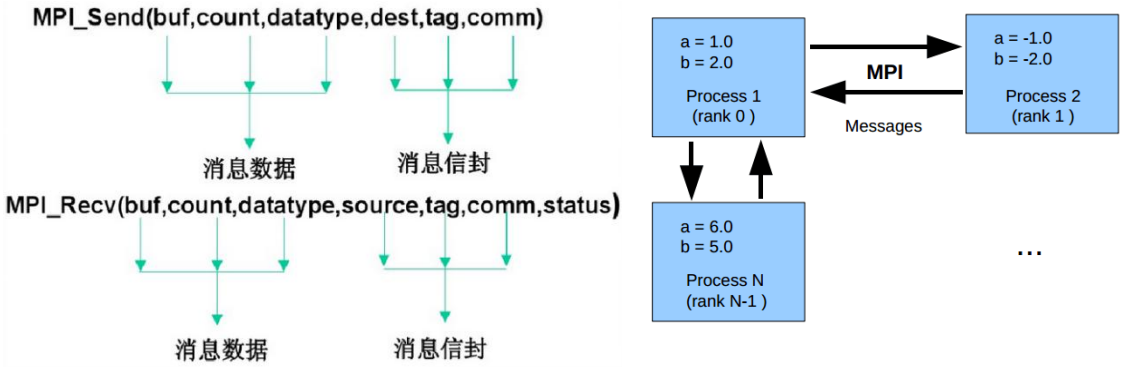
并行计算是指同时使用多种计算资源解决计算问题的过程，是提高计算机系统计算速度和处理能力的一种有效手段。它的基本思想是用多个处理器来协同求解同一问题，即将被求解的问题分解成若干个部分，各部分均由一个独立的处理机来并行计算。在一台机器上也可以实现并行计算，即使用多进程通信模拟多机通信.但是有一点需要特别指出，如果机器是单核处理器，那么只能称为”伪并行计算”，只是不同的进程或者线程在处理时来回切换，只有多核处理器下的并行是真正的并行。

分布式计算是一门计算机科学，它研究如何把一个需要非常巨大的计算能力才能解决的问题分成许多小的部分，然后把这些部分分配给许多计算机进行处理，最后把这些计算结果综合起来得到最终的结果。 并行程序并行处理的任务包之间有很大的联系，而且并行计算的每一个任务块都是必要的，没有浪费的分割的，就是每个任务包都要处理，而且计算结果相互影响，就要求每个的计算结果要绝对正确，而且在时间上要尽量做到同步，而分布式的很多任务块可以根本就不处理，有大量的无用数据块，所以说分布式计算的速度尽管很快，但是真正效率是不高的，存在很大的不确定性。而并行是在速度和确定性上都有保证。

3.2.2 信息传递接口(Message Passing Interface，MPI)

3.2.2.1MPI 的概念

信息传递接口(Message Passing Interface，MPI)是一个跨语言的通讯协议，用于编写并行计算机。支持点对点和广播。MPI 是一个信息传递应用程序接口，包括协议和和语义说明，指明其如何在各种实现中发挥其特性[17]。用 MPI 实现的程序是基于消息传递的并行程序，消息传递指的是并行执行的各个进程具有自己独立的堆栈和代码段，作为互不相关的多个程序独立执行，进程之间的信息交互完全通过显示地调用通信函数来完成。MPI 作为通信使用，追求效率而且需要在系统底层和上层有通信，故 MPI 多用于 C 和 C++语言进行实现，后来有人将 MPI 移植到了 python 上，这并不是在 python 上重构，只是为了开发者的方便，在 python 上提供借口，实际底层仍然调用 openmpi。如图 3-2-2(b)所示，这是 C 语言上 MPI 进行消息传递和通信的一个简单实例。



(a) (b)

图 3-2-2：MPI 简单通信示意图

上述图 3-2-2(a)是一个 MPI 通信的 API 层面示意，在这里简单说明。在整个 MPI 环境中有一套编程规则需要遵守，首先需要使用 MPI_Init 函数启动 MPI 环境，而且需要自己手动建立通信域(通信空间)MPI_COMM_WORLD，同时 MPI_Comm_size 和 MPI_Comm_rank 指定了进程数目和进程在通信空间中的逻辑编号；当通信调用结束则需要用 MPI_Finalize 表明结束通信。另外需要特别指出的是，MPI_Barrier 函数是 MPI 中非常重要的一个函数，用来使各个进程之间同步，封装了同步的具体实现，直接调用由 MPI 库函数实现。

3.2.2.2 OpenMPI 和 MPI for Python (mpi4py)

OpenMPI 是一种高性能消息传递库[18]。它是 MPI-2 标准的一个开源实现，由

一些科研机构和企业一起开发和维护。因此，OpenMPI 能够从高性能社区中获得专业技术、工业技术和资源支持，来创建最好的 MPI 库。OpenMPI 提供给系统和软件供应商、程序开发者和研究人员很多便利。

MPI for Python 是构建在 mpi 之上的 python 库，使得 python 的数据结构可以在进程（或者多个 cpu）之间进行传递[19]。MPI for Python 提供了一种基于标准 MPI-2 C++的面向对象的消息传递方式，提供的接口是将 C++的标准库绑定成 python 的重要方式，这样降低了编程的难度，使 Python 开发者也能使用 MPI。

3.2.2.3 参数服务器(Parameter Server, PS)

参数服务器(Parameter Server, PS)是近两年来最新提出的大规模分布式计算架构，其基于 SSP（Stale Synchronous Parallel 通信模型构建。[20]SSP（Stale Synchronous Parallel）模型是一种新的半同步模型，它的核心思想是通过设置一个同步阈值 s，使得最快的进程（或线程）与最慢的进程（或线程）的迭代次数相差不超过 s，从而限制了数据同步的开销。设计者号称能达到与异步算法相近的性能，且与严格同步模型相似的收敛精度，是近年来的研究热点。在多个领域都出现了应用，例如为了应对近年来大规模的分布式机器学习问题，由李沐等人提出了基于 SSP 通信模型的 Parameter Server 架构。

参数服务器的最大特点就是计算节点和参数服务节点分开，由参数服务节点负责存储,更新全局参数。系统的所有节点分为两类，Server 节点和 Worker(Client)节点，如图 3-2-2 所示。

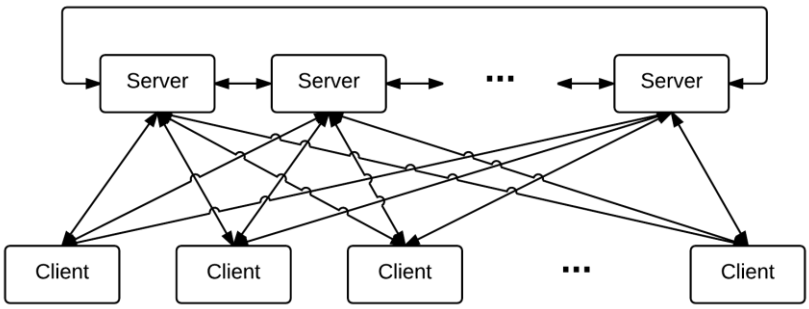


图 3-2-2：参数服务器架构:Server 和 Worker

由上图可以看到，Server 节点和很多 Worker 节点通信的同时也相互通信，但是 Worker 只于 Server 通信，之间不通信。因为全局参数的更新由 Worker 维护共享，而 Worker 只是将全局参数拉取(PULL)后处理运算然后推送(PUSH)给 Server。这也是 Server 向外部 Worker 提供的两个关键接口；另外 Server 还需要提供全局时

钟控制服务接口，这样才能保证上文提到的同步泛型的参数更新准确性。

3.3 本章小结

本章介绍文中算法实现并行时所需要的基本理论：通信泛型例如同步泛型，异步泛型；另外简单介绍了并行计算架构参数服务器节点(Parameter Server)和计算节点(Worker)，以及通信时采用的 MPI 和 OpenMPI 库以及在 python 环境下 MPI for Python(mpi4py)的介绍。在本文阐述的算法设计中为了提高运算迭代速度，设计了简单的 Parameter 架构，使用基于 openmpi 的 mpi4py 库调用 MPI，实现多进程并行算法，提高了算法效率。

第四章 张量恢复算法设计

4.1 成对互动张量分解/成对影响张量分解(Pairwise Interaction Tensor)

4.1.1 基本概念

成对互动张量(Pairwise Interaction Tensor)最近在一些推荐系统应用上有很不错的表现，例如标签推荐系统和序列数据分析。成对互动张量是通常意义张量的一个特例，是对不同变量间的成对关系进行直接建模。已电影推荐系统为例，对用户随着时间变化的电影评分去建模，成对互动张量假设每个评分由三个因子影响：用户本身故有的对电影的评价，电影流行的趋势以及用户随着时间变化的心情[5]。假设每个记录T_{ijk}都是大小为$n_1 \times n_2 \times n_3$张量T的一个元素。我们可以将整个记录写为：

$$T_{ijk} = \left\langle u_i^{(a)}, v_j^{(a)} \right\rangle + \left\langle u_j^{(b)}, v_k^{(b)} \right\rangle + \left\langle u_k^{(c)}, v_i^{(c)} \right\rangle \tag{4-1}$$

其中$(i, j, k) \in [n_1] \times [n_2] \times [n_3]$，$\{u_i^{(a)}\}_{i \in [n_1]}$，$\{v_j^{(a)}\}_{j \in [n_2]}$是$r_1$维的向量组；$\{u_j^{(b)}\}_{j \in [n_2]}$，$\{v_k^{(b)}\}_{k \in [n_3]}$是$r_2$维向量组；$\{u_k^{(c)}\}_{k \in [n_3]}$，$\{v_i^{(c)}\}_{i \in [n_1]}$是$r_3$维向量组。

一般形式的成对互动张量分解(PITF)模型形式如下：

$$\hat{y}_{x_1,\ldots,x_m}^{G-PITF} := \sum_{i=1}^{m} \sum_{j=i+1}^{m} \left\langle v_{x_i}^{i,j}, v_{x_j}^{j,i} \right\rangle = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \sum_{f=1}^{k_{i,j}} v_{x_i,f}^{i,j} \cdot v_{x_i,f}^{j,i} \tag{4-2}$$

其中模型的参数是$V^{i,j} \in R^{|X_i| \times k_{i,j}}$，$V^{j,i} \in R^{|X_j| \times k_{i,j}}$，$\forall i, j \in \{1,\ldots,m\}, i > j$，易得存在：

$$\frac{m(m-1)}{2}$$

因子分解对(factorization pairs)。

4.1.2PITF 和 PARAFAC 分解(CP)和 Tucker 分解(TD)的关系

成对互动张量分解(PITF)看似是复杂实际上属于PARAFAC分解的一种特殊形式。每个 PITF 可以用一个维度是：

$$k = \sum_{i=1}^{m} \sum_{j=i+1}^{m} k_{i,j}$$

的 PARAFAC 模型解释。假设给出了任意的 PITF 模型，将$k_{i,j}$作为成对因子维度。对应的 PARAFAC 模型的阶k可以通过将特征矩阵的部分设置为常数 1 而得到。设

集合 $Z_p := \{(i,j) \mid i,j \in \{1,..,m\}, i > j\}$ 为所有成对阶的集合[1]。显然

$$|Z_p| = \frac{m(m-1)}{2}$$

设 Z 是 PITF 所有有序索引对的集合：

$$Z := \{(i,j,f) \mid (i,j) \in Z_p, f \in \{1,..,k_{i,j}\}\}$$

Z 的基数是对应的 PARAFAC 模型的大小：

$$|Z| = k = \sum_{i=1}^{m} \sum_{j=i+1}^{m} k_{i,j}$$

令 ϕ 是 Z 到 $\{1,..,|Z|\}$ 的双射，PARAFAC 模型可以重写为：

$$\hat{y}^{PARAFAC}_{x_1,...,x_m} := \sum_{f=1}^{k} \prod_{i=1}^{m} v^{i}_{x_i,f} = \sum_{(i,j,f)\in Z} \prod_{l=1}^{m} v^{l}_{x_l,\phi(i,j,f)} \tag{4-2}$$

其中设定一些 PARAFAC 的一些参数作为常数：

$$v^{l}_{x_l,\phi(i,j,f)} = \begin{cases} u^{i,j}_{x_i,f} & \text{if } l = i \\ u^{j,i}_{x_j,f} & \text{if } l = \text{j} \\ 1 & \text{else} \end{cases}$$

综上可以推出：

$$\begin{aligned} \hat{y}^{PARAFAC}_{x_1,...,x_m} &= \sum_{(i,j,f)\in Z} \prod_{l=1}^{m} v^{l}_{x_l,\phi(i,j,f)} = \sum_{(i,j,k)\in Z} u^{i,j}_{x_i,f} u^{j,i}_{x_j,f} = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \sum_{f=1}^{k_{i,j}} u^{i,j}_{x_i,f} u^{j,i}_{x_j,f} \\ &= \sum_{i=1}^{m} \sum_{j=i+1}^{m} \left\langle u^{i,j}_{x_i}, u^{j,i}_{x_j} \right\rangle = \hat{y}^{G-PITF}_{x_1,...,x_m} \end{aligned} \tag{4-3}$$

验证了 PITF 模型是 PARAFAC 分解模型的一个特殊形式，但是当 m=2 的时候不能由 PARAFAC 模型推出 PITF 模型。对于双路的情况，PITF 和矩阵分解(matrix factorization, MF)相同:

$$\hat{y}^{PITF}_{x_1,x_2} = \left\langle v^{1,2}_{x_1}, v^{2,1}_{x_2} \right\rangle = \hat{y}^{MF}_{x_1,x_2}$$

4.2 算法设计思想

现有的一些成对互动张量的恢复算法使用局部优化方法，不能特别好的保证恢复性。在本文中，设计了一种有效的恢复成对互动张量的算法。在无噪声的情况，解决一个最小化加权矩阵核范数的凸优化问题可以很精确的恢复一个成对互动张量。将恢复问题重新定义为在特别的观察算子下的约束矩阵完备问题。先前，Gross 等人已经启发式的使用了核范数，发现可以用正交观测算子从足够数量的观测中恢复出低秩矩阵。其中正交性是非常重要的条件。而本文使用设计的算法思想依据于 Chen，Michael, King 已经证明的可以应用在非正交观测算子来处理其他

矩阵求解问题[5]。此外，算法拓展了奇异值阈值化方法来设计一个简单且可以拓展的方法来解决恢复问题。假设我们有成对互动张量的部分观测数据 $T = Pair(r, A, B$。我们设置 $\Omega \subseteq [n_1] \times [n_2] \times [n_3]$ 是 m 个观察项的索引组的集合。这里我们假设从所有的集合中均匀采样。我们的目标是恢复出矩阵 A,B,C，因此整个张量的观测值都可以恢复 $\{T_{ijk}\}_{(ijk)\in\Omega}$。在 4.1.1 中式子(4-1)可以用成对互动张量的矩阵形式来表示：

$$T_{ijk} = A_{ij} + B_{jk} + C_{ki}, \quad (i,j,k) \in [n_1] \times [n_2] \times [n_3] \tag{4-4}$$

其中 $A_{ij} = \langle u_i^{(a)}, v_j^{(a)} \rangle$， $B_{jk} = \langle u_j^{(b)}, v_k^{(b)} \rangle$， $C_{ki} = \langle u_k^{(c)}, v_i^{(c)} \rangle$，且 A，B，C 分别为秩为 r_1， r_2， r_3，的矩阵。

4.3 算法可行性分析

4.3.1 唯一性(uniqueness)分析

成对互动张量的原始恢复问题是不适定的问题。在经典的数学物理中，人们只研究适定问题。适定问题是指满足下列三个要求的问题：解是存在的（存在性）；解是惟一的（唯一性）；解连续依赖于初始值条件（稳定性）。这三个要求中，只要有一个不满足，则称之为不适定问题。我们可以构造很多不同的矩阵组合 $A`, B`, C`$，使得 $Pair(A, B, C) = Pair(A`, B`, C`)$。例如我们假定 $\delta \neq 0$， a 是一个任意大小为 n_1 非零向量

$$T_{ijk} = A_{ij} + B_{jk} + C_{ki} = (A_{ij} + \delta a_i) + B_{jk} + (C_{ki} + (1-\delta)a_i)$$

我们可以构建矩阵 $A`, B`, C`$，让 $A`_{ji} = A_{ji} + \delta_i$ ， $B`_{jk} = B_{jk}$ ， $C`_{ki} = C_{ki} + (1-\delta)a_i$ ，那么有 $T = Pair(A`, B`, C`)$。

这种不确定的模糊性质使得矩阵恢复受到困阻，即使张量整个全部被观测到但是有上述可知完全可能恢复成 $A`, B`, C`$ 而不是 A，B，C。为了避免这个问题，需要一组约束限制，使得给定任何的成对互动张量 $Pair(A, B, C)$ 存在这唯一的矩阵集 $Pair(A`, B`, C`)$ 有

$$Pair(A, B, C) = Pair(A`, B`, C`)$$

命题 4.3.1 对于任意的成对互动张量 $T = Pair(A, B, C)$，存在唯一的 $A` \in S_A$，$B` \in S_B$， $C` \in S_C$ 使得 $Pair(A, B, C) = Pair(A`, B`, C`)$

其中定义：

$$S_B = \{M \in R^{n_1 \times n_2} : 1^T M = 0^T\}, S_c = \{M \in R^{n_3 \times n_1} : 1^T M = 0^T\}, S_A = \{M \in R^{n_1 \times n_2} : 1^T M = (\frac{1}{n_2} 1^T M 1) 1^T\}$$

唯一性问题和偏好(bias)存在这某种自然的联系，取得了推荐系统领域的广泛

关注。

4.3.2 不相干性(incoherence)分析

很容由上述概念得出恢复一个成对张量$T = Pair(A,0,0)$等价于恢复其子项中的矩阵 A。因此恢复成对互动张量问题包括了矩阵完备问题。以前的研究已经证实了不相干性条件是保证矩阵成功恢复的必要基本条件。设$M = U\Sigma V^T$是对秩为 r 的奇异值分解，我们可以称矩阵 M 是(μ_0,μ_1)不相干当且仅当 M 满足：
(1)对于所有的$i \in [n_1]$和$j \in [n_2]$有

$$\frac{n_1}{r}\sum_{k\in[r]} U_{ik}^2 \le \mu_0$$

且

$$\frac{n_2}{r}\sum_{k\in[r]} V_{jk}^2 \le \mu_0 \tag{4-5}$$

(2)UV^T中的最大项界限为$\sqrt{r/n_1 n_2}$的绝对值。

众所周知，只有当矩阵满足了边界为u_0,u_1的(μ_0,μ_1)不相干条件(其中u_0,u_1是关于 n 的多对数)才有可能恢复。由于矩阵完备问题可以还原为成对互动张量的恢复问题，所以我们在理论上要在矩阵 A,B,C 上继承不相干性假设。

4.4 算法设计过程

4.4.1 凸优化算法表示恢复模型

首先考虑精确恢复情况，假设我们给出了 m 个观测值$\{T_{ijk}\}_{(ijk)\in\Omega}$，其中$\Omega$是在$[n_1]\times[n_2]\times[n_3]$上的随机均匀采样。我们假设恢复矩阵是 A,B,C，所以对$T = Pair(A,B,C)$使用一下的凸问题[5]：

$$\begin{aligned} &\min_{X\in S_A, Y\in S_B, Z\in S_C} imize \ \sqrt{n_3}\|X\|_* + \sqrt{n_1}\|Y\|_* + \sqrt{n_2}\|Z\|_* \\ &s.t.\ \ X_{ij} + Y_{jk} + Z_{ki} = T_{ijk} \ \ (i,j,k) \in \Omega \end{aligned} \tag{4-6}$$

其中$\|M\|_*$代表矩阵 M 的核范数，即矩阵 M 的奇异值之和，S_A，S_B，S_C在命题 4.3.1 中给出。

对于有噪声的情况，假设噪声是已经观测到的，则有

$$\hat{T}_{ijk} = T_{ijk} + \sigma_{ijk}$$

其中$(i,j,k) \in \Omega$，σ_{ijk}是噪声项，可以是确定的或者随机生成的。假设σ在Ω上有界且$\|P_\Omega(\sigma)\|_F \le \varepsilon_1$，部分$\varepsilon_1 > 0$。这里$P_\Omega(\cdot)$代表这在$\Omega$上的约束。在上述假设下，

我们可以得出误差界限遵循二次约束凸问题：

$$\min_{X\in S_A,Y\in S_B,Z\in S_C} imize \ \sqrt{n_3}\|X\|_* + \sqrt{n_1}\|Y\|_* + \sqrt{n_2}\|Z\|_* \tag{4-5}$$
$$s.t. \ \left\|P_\Omega(Pair(X,Y,Z)) - P_\Omega(\hat{T})\right\|_F \le \varepsilon_2$$

有几种方法可以考虑用来解决上述的两个凸优化问题。对于数据规模较小的问题，可以将优化问题定义为半正定问题，使用内点法解决。目前有很先进的内点求解方法可以得到最佳的解决方案以及很好的准确性。但是，传统的内点方法在数据(成对互动张量)规模变大的时候速度变得很缓慢。在 Chen，Michael，King 的文中提出了使用奇异值阈值化(SVT)方法，这一方法最早由 Cai 等人提出[9]。对于式子，将问题中每项核范数的系数改写在约束条件中，则可以另写为：

$$\min_{X\in S_A,Y\in S_B,Z\in S_C} imize \ \|X\|_* + \|Y\|_* + \|Z\|_*$$
$$s.t. \ \frac{X_{ij}}{\sqrt{n_3}} + \frac{Y_{jk}}{\sqrt{n_1}} + \frac{Z_{ki}}{\sqrt{n_2}} = T_{ijk} \ \ (i,j,k)\in\Omega \tag{4-6}$$

这样显然可以看到恢复张量形式变成了 $Pair(n_3^{-1/2}X, n_1^{-1/2}Y, n_2^{-1/2}Z)$，其中 X,Y,Z 是上式的最优解。(4-6)为了防止过拟合，引入了 L2 正则项(此处使用了 F 范数)，得：

$$\min_{X\in S_A,Y\in S_B,Z\in S_C} imize \ \tau(\|X\|_* + \|Y\|_* + \|Z\|_*) + \frac{1}{2}(\|X\|_F^2 + \|Y\|_F^2 + \|Z\|_F^2)$$
$$s.t. \ \frac{X_{ij}}{\sqrt{n_3}} + \frac{Y_{jk}}{\sqrt{n_1}} + \frac{Z_{ki}}{\sqrt{n_2}} = T_{ijk} \ \ (i,j,k)\in\Omega \tag{4-7}$$

当$\tau\to\infty$时上式和原式相同，所以在实际算法中，需要对参数τ选取一个很大的值来保证凸问题近似等价。

4.4.2 张量恢复算法中的约束采样，伴随算子以及收缩算子

设计的算法将迭代的求解最小化上述加入了正则化项的凸优化问题，产生一系列的矩阵$\{X^k,Y^k,Z^k\}$,使得最终收敛得到$\{X,Y,Z\}$做为上述凸优化问题的最优解。设计算法前需要定义如何采样以及具体的一些操作算符。

首先是对于观测空间的定义，即定义观测数据如何选择。设定$\Omega=\{a_i,b_i,c_i \mid i\in[m]\}$，a，b，c 分别是在三个不同维度上的采样位置序列，一共观测 m 个数据。对于所观测到的数据加上约束来表示观测在 X，Y，Z 矩阵上的影响。设定算子：$P_{\Omega A}: R^{n_1\times n_2}\to R^m$，$P_{\Omega B}: R^{n_2\times n_3}\to R^m$，$P_{\Omega c}: R^{n_3\times n_1}\to R^m$

可以看出三个算子分别是在各自采样空间中进行了空间变换，通俗的讲将二维矩阵压缩成了一维向量。三个算子的具体操作分别定义如下：

$$P_{\Omega_A}(X) = \frac{1}{\sqrt{n_3}}\sum_{i=1}^{m} X_{a_ib_i}\delta_i, P_{\Omega_B}(Y) = \frac{1}{\sqrt{n_1}}\sum_{i=1}^{m} Y_{b_ic_i}\delta_i, P_{\Omega_C}(Z) = \frac{1}{\sqrt{n_1}}\sum_{i=1}^{m} Z_{c_ia_i}\delta_i \tag{4-8}$$

其中 X，Y，Z 分别是采样对应的矩阵，δ_i 是第 i 个采样对应的采样向量。由上式子很容易证明有 $P_{\Omega_A}(X)+P_{\Omega_B}(Y)+P_{\Omega_C}(Z)=P_{\Omega}(Pair(n_3^{-1/2}X,n_1^{-1/2}Y,n_2^{-1/2}Z))$。同时，不仅对于矩阵仅有采样操作，在使用奇异值阈值化(SVT)方法的时候,因为是对矩阵操作，所以还需要将采样向量还原称为矩阵，因此还要设计采样操作 $P_{\Omega}(\cdot)$ 的伴随算子(adjoint operator)。这里用 P_{Ω}^{*} 表示。则有 $P_{\Omega_A}^{*}$，$P_{\Omega_B}^{*}$，$P_{\Omega_C}^{*}$ 分别表示 P_{Ω_A}，P_{Ω_B}，P_{Ω_C} 的伴随算子。

另外，对于一个大小为 $n_1\times n_2$ 的矩阵 X，定义:

$$center(X)=X-\frac{1}{n_1}11^{T}X \tag{4-9}$$

作为矩阵 X 的列中心化算子。可以看出 $center(X)$ 实际上是 n_2 上每一列减去了本列的平均值。并且很容易得到 $1^{T}center(X)=0^{T}$。还需要定义在奇异值阈值化操作中的收缩算子(shrinkage operator)。

收缩算子同样是由凸优化问题得到的，例如对于矩阵 X 上关于采样空间 Ω_A 的设定的收缩算子 $shrink_A$：

$$shrink_A(M,\tau):=\arg\min_{\tilde{M}\in S_A}\frac{1}{2}\left\|\tilde{M}-M\right\|_F^2+\tau\left\|\tilde{M}\right\|_* \tag{4-10}$$

收缩算子 $shrink_B$ 和 $shrink_C$ 也是用类似的方法进行定义，但各自都在其约束空间 S_B 和 S_C 内。在原始的奇异值阈值化方法中，对于矩阵是没有约束的，但是这里对矩阵进行了约束，另外带有约束的收缩算子可以使用奇异值分解(SVD)方法对列中心化的矩阵进行分解。定义列中心化矩阵的奇异值分解: $center(M)=U\Sigma V^{T}$，$\Sigma=diag(\{\sigma_i\})$。可以证明收缩算子可以以如下形式给出[9]:
$shrink_B(M,\tau)=Udiag(\{\sigma_i-\tau\}_+)V^{T}$ 其中，s_+ 是 s 的正部分，故 $s_+=\max\{0,s\}$。由于子空间的定义中 S_B 和 S_C 是相同形式的，所以在收缩算子的计算上很容易看出 $shrink_B$ 和 $shrink_C$ 相同计算形式，则有：$shrink_C(M,\tau)=Udiag(\{\sigma_i-\tau\}_+)V^{T}$。

但是对于 $shrink_A$ 的计算有一些不同和复杂，根据凸优化问题可以得出 $shrink_A$ 定义为：

$$shrink_A(M,\tau)=Udiag(\{\sigma_i-\tau\}_+)V^{T}+\frac{1}{\sqrt{n_1n_2}}(\{\delta-\tau\}_+ +\{\delta+\tau\}_-)11^{T} \tag{4-11}$$

其中 $U\Sigma V^{T}$ 仍然是列中心矩阵 $center(M)$ 的奇异值分解,且

$$\delta = \frac{1}{\sqrt{n_1 n_2}} 1^T M 1 \tag{4-12}$$

是一个常数， $s_- = \min\{0, s\}$ 是 s 的负部分。整理上述的算子和定义，结合凸问题，使用迭代收敛的思想设计算法，核心步骤主要是一下几步：

(1)由成对互动张量生成数据采样序列 Ω 以及数据采样 P_Ω

(2)奇异值阈值化操作 $M^K = shrink(P_\Omega^*(y^{k-1}), \tau)$,其中 P_Ω^* 是伴随算子， y 是数据采样向量， τ 是阈值常数， $shrink$ 是收缩算子。

(3)收敛和误差判断

$$e^k = P_\Omega(T) - P_\Omega(Pair(n_3^{-1/2}X, n_1^{-1/2}Y, n_2^{-1/2}Z)) \tag{4-13}$$

$$y^k = y^{k-1} + \delta e^k$$

该算法在步骤(2)和(3)之间迭代并产生一系列 (X^k, Y^k, Z^k) 使得凸优化问题收敛得到最优解，迭代过程的终止条件是训练误差足够的小， $\|e^k\|_F \le \varepsilon$ 。关于结果的收敛性在 Cai 提出的奇异值阈值化算法(SVT algorithm)中有证明[9]。通过不断迭代

算法 4.4.2 收缩算子

1: 参数： $\hat{X}, \tau, r$

2：流程 1： $shrink_B(\hat{X}, \tau, r)$

3： $s \leftarrow r+1$

4： $repeat$

5： $[U, \Sigma, V] \leftarrow svd(center(\hat{X}), s)$

6： $s \leftarrow s+5$

7： $until\ \sigma_{s-5} \le \tau$

8: $r \leftarrow \max(j : \sigma_j > \tau)$

9: $X \leftarrow \sum(\sigma_i - \tau) u_j v_j^*$

10: $return$ [X,r]

11: 结束流程 1

12: 流程 2： $shrink_A(\hat{X}, \tau, r)$

13: $[X, r] \leftarrow shrink_B(\hat{X}, \tau, r)$

14: $\delta \leftarrow sum(\hat{X})$

15： $return$ [X+$\gamma 1 1^T$, r]

16：结束流程 2

更新，最终收敛达到设定误差范围或者最大迭代次数算法迭代终止。

收缩算子 $shrink_B(\hat{X},\tau,r)$ 实现如算法 4.2.2 所示，传入参数矩阵 $\hat{X}$，阈值 τ，以及秩 r。对中心化后的矩阵做奇异值分解，取大于阈值的奇异值并且确定了数目，然后截取这些奇异值恢复得到收缩后的矩阵 X。

4.4.3 无噪声精确恢复算法流程

基于本文前面提到的算法设计思想，可以写出算法 4.4.3 的整个流程。

传入参数：采样空间 Ω，观测采样序列 $P_\Omega(T)$，以及迭代更新步长 δ，阈值 τ，以及设定的错误率 ε。设定初始采样向量为全 0 向量，大小为观测数目 m。对于采样向量先进行采样运算的伴随算子操作后，恢复成一个矩阵然后使用矩阵奇异值阈值化方法，再进行收缩算子操作，对于每次迭代的每个收缩操作都会得到相应的矩阵，然后通过这些矩阵根据判断矩阵生成的采样张量和原来观测的样本的 F 范数的差值，来判断是否收敛。如果已经满足收敛
条件则跳出循环，若没有满足继续迭代，则通过在现有采样张量啥加上步长和误差张量的成绩构成新的采样张量后重复上述步骤，最后返回生成的三个带有系数处理的矩阵 X，Y，Z。

算法 4.4.3 无噪声成对互动作用张量的精确恢复算法

参数：$\Omega=\{a_ib_ic_i\}_{i\in[m]}$，$P_\Omega(T)=\{T_{a_ib_ic_i}\}_{i\in[m]}$，$\delta$，$\tau$，$\varepsilon$
流程：
    $y\leftarrow 0$
    for k=1,...k_{max} do
        $[X,r_A]\leftarrow shrink_A(P^*_{\Omega_A}(y),\tau,r_A)$
        $[Y,r_B]\leftarrow shrink_B(P^*_{\Omega_B}(y),\tau,r_B)$
        $[Z,r_C]\leftarrow shrink_B(P^*_{\Omega_C}(y),\tau,r_C)$
        $e\leftarrow P_\Omega(T)-P_\Omega(Pair(n_3^{-1/2}X,n_1^{-1/2}Y,n_2^{-1/2}Z))$
        if $\|e\|_F/\|P_\Omega(T)\|_F\le\varepsilon$ then
            break
        end if
        $y\leftarrow y+\delta e$
    end for
    return $[n_3^{-1/2}X,n_1^{-1/2}Y,n_2^{-1/2}Z]$

在每次迭代中，我们需要计算奇异值分解，并且执行几个基本的矩阵相加操

作，在每次迭代时，X 矩阵都会在采样空间$\Omega_A=\{a_ib_i\}$上收缩，并且是稀疏矩阵。相同的 Y，Z 同样也是稀疏矩阵。收缩算子是对列中心化矩阵进行了奇异值分解，其中矩阵是稀疏矩阵且是一个秩 1 矩阵，显然对于列中心化的矩阵向量乘法时间计算复杂度为$O(n+m)$。这样可用基于 SVD 的 Lanczos 方法方法实现，例如 PROPACK 和 SVDPACKC[12]，只需要计算矩阵向量积的部分流程。

在本文算法设计中使用了适合本文情况的用于计算收缩算子的 SVDPACK 版本。另外，为了对阈值τ有一个恰当的选择，$\{X^k,Y^k,Z^k\}$都要是低秩矩阵，与奇异值阈值化算法的要求一样。并且在迭代的过程中X^k，Y^k，Z^k可以保持低秩，因为计算时只需要执行部分 SVD 得到前 r 个奇异向量。使用[3]提出的类似方法可以使得估计的秩 r 随着迭代不断增长。

总而言之，恢复算法在每次迭代的总体复杂度为$O(r(n+m))$。

4.4.4 有噪声可靠恢复算法流程

带有噪声的优化问题可以用类似的算法流程 4.4.4 表示，但是在进行误差和收敛判断的时候需要修改，引入误差后需要将误差处理步骤修改来合并方程式的二次约束。修改如下：

$$\begin{aligned} e &\leftarrow P_\Omega(\hat{T})-P_\Omega(Pair(n_3^{-1/2}X,n_1^{-1/2}Y,n_2^{-1/2}Z)) \\ \begin{bmatrix} y^k \\ s^k \end{bmatrix} &= P_k\left(\begin{bmatrix} y^{k-1} \\ s^{k-1} \end{bmatrix}+\delta\begin{bmatrix} e^k \\ -\varepsilon \end{bmatrix}\right) \end{aligned} \tag{4-14}$$

其中$P_\Omega(\hat{T})$是噪声观测值，P_k是锥投影算子可由下式计算得到：

$$P_K:(x,t)\rightarrow\begin{cases} (x,t) & \text{if } \|x\| \le t \\ \dfrac{\|x\|+t}{2\|x\|}(x,\|x\|) & \text{if } -\|x\| \le t \le \|x\| \\ (0,0) & \text{if t } \le -\|x\| \end{cases} \tag{4-15}$$

当 x 的模长小于等于 t 时，原向量不变，保持在空间中原位置不变，当 t 大于负的 X 的模长且小于等于 X 的模长时，原向量被投影到了锥投影空间的边界上；当 t 小于等于负的 x 的模长时，原向量被投影到了空间中原点一个点上。

算法 4.4.4 有噪声成对互动作用张量的稳定恢复算法

1: 参数：

$\Omega=\{a_ib_ic_i\}_{i\in[m]}$， $P_\Omega(T)=\{T_{a_ib_ic_i}\}_{i\in[m]}$， $\delta,\tau,\varepsilon,\varepsilon_1$

2：流程：

3： $y\leftarrow 0$

4： *for* k=1,...k_{max} do

5： $[X,r_A]\leftarrow shrink_A(P^*_{\Omega_A}(y),\tau,r_A)$

6： $[Y,r_B]\leftarrow shrink_B(P^*_{\Omega_B}(y),\tau,r_B)$

7： $[Z,r_C]\leftarrow shrink_B(P^*_{\Omega_C}(y),\tau,r_C)$

8: $e\leftarrow P_\Omega(\hat{T})-P_\Omega(Pair(n_3^{-1/2}X,n_1^{-1/2}Y,n_2^{-1/2}Z))$

9: *if* $\|e\|_F/\|P_\Omega(\hat{T})\|_F\le\varepsilon$ then

10: *break*

11: *end* if

12: $y\leftarrow y+\delta e$

13: $s\leftarrow s-\delta\varepsilon_1$

14: $[v,s]\leftarrow P_r(v,s)$

15: *end* for

16: *return* $[n_3^{-1/2}X,n_1^{-1/2}Y,n_2^{-1/2}Z]$

4.5 算法的数据并行和模型并行设计

4.5.1 设计思想

并行计算模型通常指从并行算法的设计和分析出发，将各种并行计算机的基本特征抽象出来，形成一个抽象的计算模型。从更广的意义上说，并行计算模型为并行计算提供了硬件和软件界面，在该界面的约定下，并行系统硬件设计者和软件设计者可以开发对并行性的支持机制，从而提高系统的性能。如图4-5-1所示，展示了一个并行模型的例子。

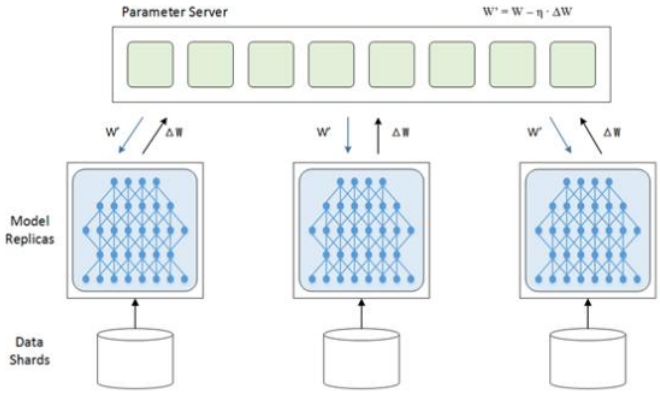


图 4-5-1:并行模型示意图

4.5.2 模型并行(Model Parallelism)

模型并行是将算法步骤分给多个 Worker，由 Server 进行协调控制，最后负责结果的整理。一般用于模型巨大，单机内存不足时将大模型分给不同机器。如图 4-5-2 所示。本文设计的算法模型没有特别复杂，而且模型上的开销并不大，但是为了提高计算效率可以设计基于模型并行的模型。设计的示意图，如下图 4-5-3 所示。

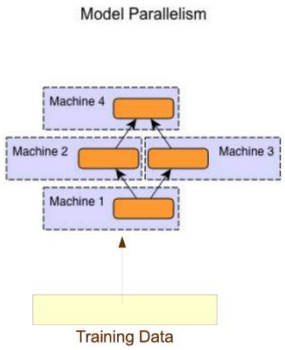


图 4-5-2：模型并行示意图

图中参数服务器有三条蓝色有向线指向了三台 Worker，经过采样空间生成出采样序列，三台 Worker 接收各种参数和采样向量后分别进行各种运算，例如收缩算子和伴随算子等。然后将结果(所得矩阵)由橙色有向线返回给参数服务器，由服务器进行误差计算，完成一次迭代，直到满足条件后停止否则重复上述过程。

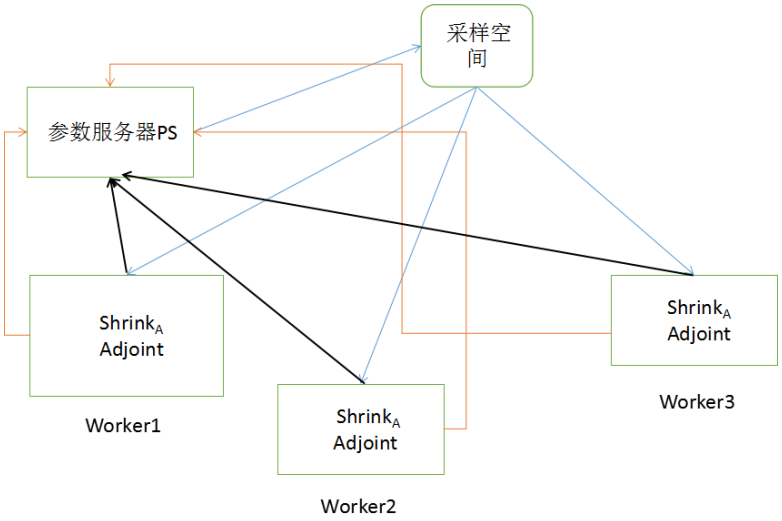


图 4-5-3：基于 PS 模型并行示意图

4.5.3 数据并行(Data Parallelism)

当数据巨大的时候，一次在单机上存储全部数据以及在网络通信上传递整个数据是不现实的，所以可以将数据分块，每块数据由 Worker 均可以独立处理，然后所得结果统一由 Server 进行处理，一般使用取平均值的方法或者增量更新的方式。如图4-5-4所示。另外一种设计方案可以将要处理的较大张量分为小张量处理，思想有对于大张量分割为小张量，也可以在采样空间进行分割成多个采样序列，使用类似批处理的方法处理。如图 4-5-5 所示。

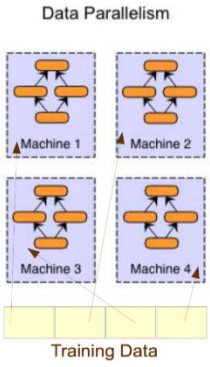


图 4-5-4：数据并行示意图

本文中采样空间的序列是三维形式，可以对采样向量的下标进行分割。一种简单的方法是每个 Worker 分的采样序列的一部分，分割如图蓝线所示。这样每个 Worker 相当与独立采样，独立操作。最后由 Server 统一处理计算。

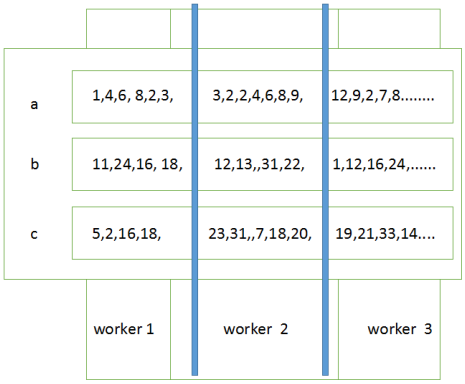


图 4-5-5:采样序列的数据并行形式示意图

4.6 本章小结

本章介绍张量恢复算法的分析和设计。首先对于恢复可行性进行分析，然后提出了本文的恢复模型和最优化问题设计思想，同时在分析同时改进提出的凸问题一步一步得出最终模型。并且设计了无噪声精确恢复算法和有噪声稳定恢复算法，给出两种算法算法具体流程。给出了算法的时间复杂度，并且详细解释了对于噪声使用了奇异值阈值化算法中的锥投影处理。

第五章 模型测试与实验结果分析

5.1 实验环境

表 5-1：实验设备配置表

类别	配置详情
CPU	Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHZ　6 核 12 线程
内存	64GB
操作系统	Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic x86_64)
编程语言	Python 3.5
Numpy	Numpy-1.11.3
OpenMPI	Openmpi-1.10.2
Mpi4py	Mpi4py-2.0.0
网络	万兆局域网

5.2 实验数据和参数设定

本次试验选取了两种数据，一个是由 numpy 的 numpy.random.rand 函数随机生成且均匀分布的数据；另外是 MovieLens Dataset,MovieLens 是历史最悠久的推荐系统。它由美国 Minnesota 大学计算机科学与工程学院的 GroupLens 项目组创办，是一个非商业性质的、以研究为目的的实验性站点。MovieLens 主要使用 Collaborative Filtering 和 Association Rules 相结合的技术，向用户推荐他们感兴趣的电影。

MovieLens 数据集的形式是四元组形式：

（用户 ID，电影 ID，时间戳，评分）

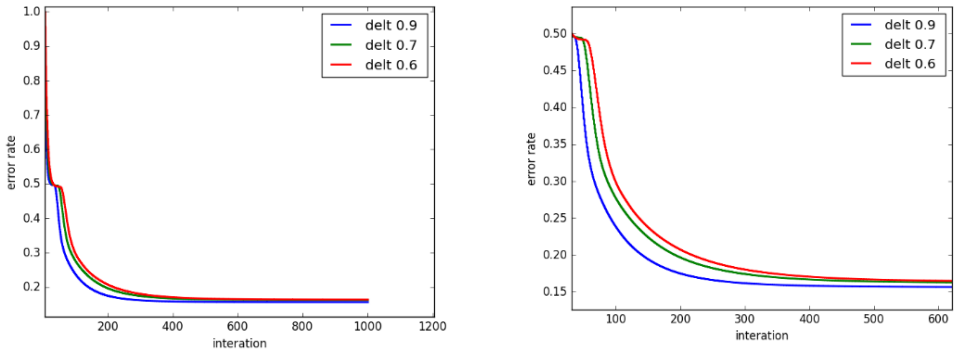
选取的 MovieLens 小数据集是从 2016 年 10 月 17 日更新的，其中用户有 671 个，电影有 164979 部，统计了从 1995 年 1 月 9 日到 2016 年 10 月 17 日这段时间十万个评分数据，大数据集约有两千万条数据，又因为时间戳使用的计算机计时的方法所以非常巨大，必须要进行处理。可以简单的将时间全部转为以月份为计时单位，即变为 12 个月的形式，极大降低了运算开销，便于计算。另外 MovieLens 是一个五星评分系统(5 star rating)，所以评分是在 0 到 5 之间离散值，且只会出现半星，即 0.5 这种小数评分。

数据处理后，还是得到一个很大的张量。用传统的张量分解算法会开销很大。所以选择部分数据作为训练集，然后用剩下数据作为测试集。另外，在原始数据中，是按照用户 ID 或者电影 ID 进行某一维度有序排列的，为了数据的随机性和结果的说服力，需要对数据进行打乱处理，然后再进行选择。

5.3 参数训练步长 δ 的选择

由算法 4.4.3 无噪声成对互动作用张量的精确恢复算法的整个流程可知，在一次迭代之后需要通过对采样向量的更新操作来重新完成下一次迭代。其中训练步长可以理解成每一次更新时选取了多大的更新量。根据数据的不同训练步长当然不同，如果步长过长，则会出现误差极快降低但并不收敛趋于稳定，而是大幅度波动或者出现极大的负增长；如果步长过小，则误差降低极慢，在最大迭代步数内不能完成收敛，浪费时间和资源。

因为步长和数据大小和观测数据量有关，而且步长取值范围在 0 到 1 之间，所以我们为了快速定位步长而不是每次都在取值范围内随意试取，所以设定每次选取步长 δ 的时候使用 $\delta = am(n_1n_2n_3)^{-1}$ 形式的定位，其中 m 是采样观测数目，n_1, n_2, n_3 分别是张量的三个维度，a 是一个常数。由于数据是稀疏的，所以 $m < n_1n_2n_3$ 是显然的因此 a 取值小于 1 肯定满足训练步长的要求。下面实验举例说明。



(a) (b)

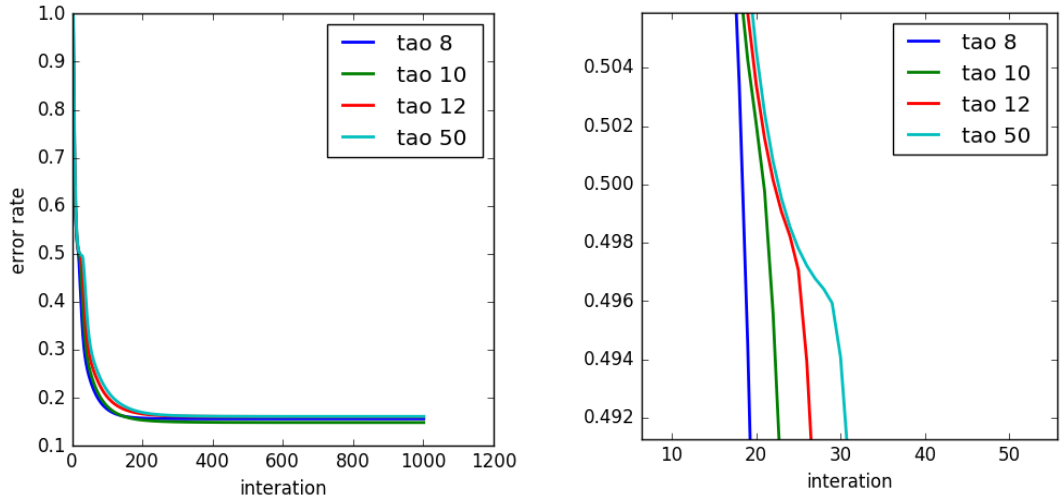
图 5-3-1：训练步长 δ 的选择和曲线下降趋势关系图

如图 5-3-1 所示，纵轴表示错误率，横轴表示迭代次数，固定阈值参数 τ，红色，蓝色，绿色分别表示相同形状数据不同时，训练步长中 a 选择不同值(0.9,0.7,0.6)的情况，并且为了证明训练步长本身不影响恢复准确率，使用同一个张量后的结果。另外，仅仅为了说明对于训练步长的选取并没有刻意调整其他参数，所以错误率(error rate)比实际的要高。但是，由图可以明显的看出，随着训练步长的减小，

曲线的下降趋势变缓。图中三条曲线所用数据是 numpy.random.rand 随机生成的在 0 到 10 之间的数据。左图(a)是整个迭代过程的图，右图(b)是左图放大后的图，可以清楚看到训练步长和曲线下趋势的关系。

5.4 参数奇异值阈值 τ 的选择

由文中 2.3 和 4.4 章节可知，在奇异值阈值化算法以及本文设计的算法之中，奇异值阈值是一个非常重要的值，这个参数控制着每一步 SVT 所保留的信息量。众所周知，奇异值分解中所得的奇异值对角矩阵中奇异值是从大到小排列，如果阈值选得过小，则收缩算子操作的收缩效果很差，如果阈值选得太大，会丢失很多信息。所以同 5.3 所说，为了快速定位奇异值阈值 τ 的选取，同样使用形式定位，即 $\tau = b\sqrt{n_1 n_2 n_3}$，其中 n_1, n_2, n_3 是张量的三个维度，b 是一个常数。因为在 SVT 中 τ 的选取值是一个比较大的参数，n_1, n_2, n_3 积的平方根随着张量维度增加也是一个较大的值，所以选择 b 的时候不必特别大。下面实验举例说明。



(a) (b)

图 5-4-1：奇异值阈值 τ 的选择和曲线下降趋势关系图

如图 5-4-1 所示，纵轴表示错误率，横轴表示迭代次数，固定阈值参数，蓝色，绿色，红色，青色分别表示相同形状数据不同时，奇异值阈值中选择不同值(8,10,12,50)的情况。可以看出对准确率没有过多的影响。但是在右图所示可以看出，τ 值的选择影响着迭代中很重要的一点：局部最小值问题(local minimum)。如果仅仅是基于梯度下降的简单的启发式学习方法，很有可能陷入局部最小值，即所谓的鞍点。通过我们的实验，可以说明：所提出的模型的凸问题是正确的，尽管 τ 值很大，

导致了曲线下降不是光滑的，但是最终可以避免陷入局部最优，达到全局最优。如图 5-4-2 所示，是正确的从局部最优值转向全局最优值的示意图。

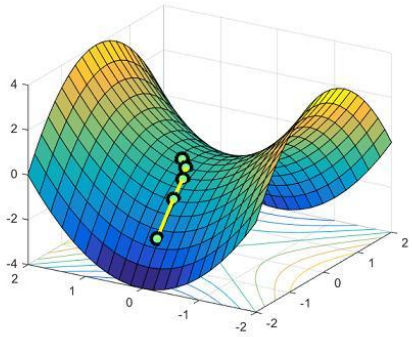


图 5-4-2：正确的梯度下降示意图

5.5 参数秩 rank 的选择

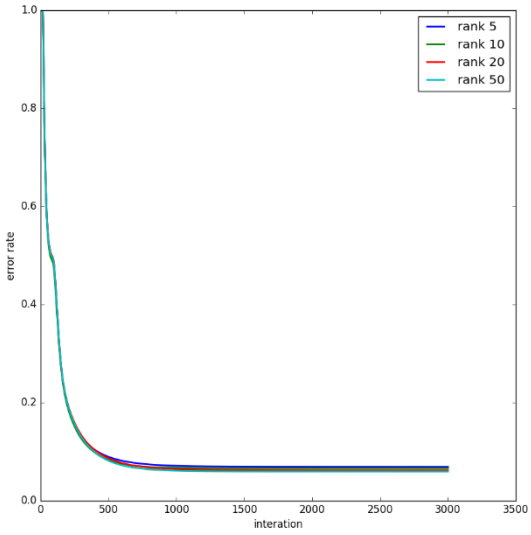


图 5-5-1：秩的影响

在线性代数中，一个矩阵 A 的列秩是 A 的线性独立的纵列的极大数。在本文算法中，秩也是一个需要指定的参数。张量可以用三个矩阵表示，如果随机生成每个矩阵，在每个子空间上进行约束后，然后生成随机矩阵时要用到秩。在算法设计中收缩算子需要传入秩作为参数。固定张量数据和形状，以及其它参数，改变秩的值。如图 5-5-1 所示。可以从图中清晰看出，范围内秩的变化对于成对互动张量恢复算法基本没有影响，rank 取值(5，10，20，50)四种分别由蓝色，绿色，红色，青色线表示。图中四条线基本重合，说明了秩只要取值符合算法逻辑并不

会影响结果。

5.6 实验在真实数据集上的分析

真实数据集上的实验使用了在 5.2 中提到的数据集 MovieLens，选取了大约十万行左右的数据，并且使用百分之七十作为训练集，剩下的百分之三十作为测试集。如表 5-6 所示，在不同的训练集合规模和参数选择上，选取 5 次训练实验结果，分别由小到大选择了张量形状，即整体数据规模；又选择了样本数目，即采样向量的长度。根据第四章中的算法收敛判断以及均方根误差(Root Mean Square Error,RMSE)作为实验结果的评判标准。

表 5-6 不同规格和参数的训练实验对比表

序号	张量形状	δ	τ	采样数目	错误率	均方根误差
1	(2000,150,36)	0.6	10	16800	0.06837903	0.25694
2	(3000,300,12)	0.9	13	25200	0.06618218	0.24538
3	(5000,300,36)	1.5	11	30000	0.07385616	0.39025
4	(7000,200,36)	4	20	50000	0.06156630	0.23068
5	(6000,300,36)	5	20	70000	0.07258203	0.27196

从上表可以看到，模型在调整合适参数训练时候可以获得很低的错误率，但是这只能说明算法可行，但是不能说明模型的泛化能力，而且非常容易存在过拟合现象。所以在下面的实验中主要使用测试集数据对训练所得模型进行验证。

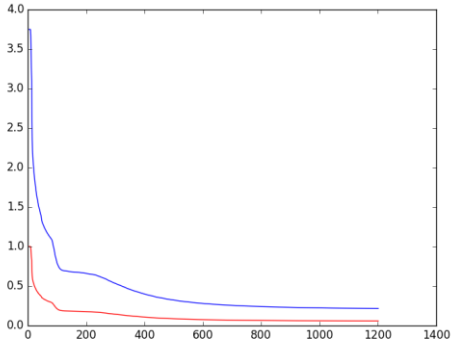


图 5-6-1:序号 5 实验完整迭代实验结果图

在图 5-6-1 中选取了一次训练实验，其中蓝色线表示错误率(Error rate)，红色线表示均方根误差(RMSE)。可以清楚的看出在近 1200 步迭代过程中，错误率和均

方根误差均在不断下降，在前 100 步迭代下降很快，从 200 步开始趋于缓和，基本在 600 步迭代的时候区域平缓，缓慢下降，直到收敛。

5.7 算法并行模型的伪代码

在本小节中，使用伪代码的形式将 4.5 中提到算法并行模型表示出来，通过代码框架的形式可以清晰的看出整个算法模型的架构。其中将 Server 和 Worker 的具体算法操作部分省略，通信部分用伪代码简单表示。

5.7 张量恢复的并行算法伪代码

```
生成 Server 进程;
生成 Worker 进程;

Server:
    数据预处理和采样;
    参数设置;
    广播发送参数字典到各个 Worker;
    循环:
        广播发送采样向量;
        接收计算的 X 矩阵;
        接收计算的 Y 矩阵;
        接收计算的 Z 矩阵;
        计算误差对比和迭代判断:
            满足迭代次数和误差:
                跳出循环;
        更新采样向量;

Worker:
    接收参数字典;
    循环:
    接受采样向量
    对于子进程 1:
        计算;
        发送结果
    对于子进程 2:
```

计算；

发送计算结果；

对于子进程 3：

计算；

发送计算结果；

5.8 本章小结

本章介绍针对算法的设计实现和框架，以及在模拟的随机数据和真实数据集上进行试验的结果。首先控制变量的方法研究了各个参数的设置选择，并且对出现的问题进行了研究和说明，并且给出了解决方法，最后进行了真实数据集上的实验并且对于结果进行阐述和分析，验证理论正确性和结论可靠性。

第六章 全文总结和展望

6.1 全文总结

本文针对成对互动张量的恢复问题，提出了一种高效的张量恢复算法，在对于恢复问题分析论证后，不仅利用了成对互动张量本身的概念和性质，并未使用传统方法，另外使用了奇异值阈值化的方法设计算法，并且为了提高效率设计了算法的并行结构，然后用真实的电影数据集测试获得了不错的效果。

本文首先介绍张量和张量分解的基本概念，以及张量分解的基本算法两种方法；另外对数据分析领域最优化和凸问题的思想和求解进行阐述，导出了算法的设计思想，对于算法中用到的奇异值阈值化进行介绍；针对算法中所要涉及的投影算子和伴随算子等数学空间操作进行具体化的简要的阐述。接着对于恢复可行性进行分析后，提出了本文的恢复模型和最优化问题设计思想，同时在分析同时改进一步一步得出最终模型。并且设计了并行的无噪声精确恢复算法和有噪声稳定恢复算法，给出两种算法算法具体流程和细节，然后在模拟生成的随机数据上和真实的电影评分数据上分别实验，和传统方法比得到了不错的效果。说明了算法的可行性，说明了对于现有概念，可以从不同的理论角度出发，使用新的方法解决问题，并且取得较好的效果。

6.2 后续工作展望

目前设计的成对互动张量的算法是对于高维张量中最普遍的三维形式进行设计，而且设计并行的模型也是一个较为简单的模型。后续研究工作希望能将算法拓展到更高维度上，或者拓展设计到大规模分布式系统上，可以应用到集群环境以及目前非常流行的 GPU 计算框架上，并且将拓展优化后的算法用于大规模分布式张量分解工具包中作为一个第三方库函数使用。

致 谢

首先感谢徐增林老师在本科毕设方面的方向指引，和百忙之中的理论指导；感谢父母，背井离乡异地求学这几年给予了物质和精神方面极大的支持；感谢实验室提供的科研条件和环境，以及给予各种帮助的师兄们，尤其是叶锦棉学长，对于一个编程水平一般的师弟的耐心和技术支持。

白驹过隙，俯仰之间，本科四年却做似水流年。回首之时，历历在目，恍若一梦。起起落落，跌宕起伏。宴散之时才发觉，在电子科大的这四年并不是象牙塔，是人的蜕变成长。在各种知识面前明白了人在浩瀚的客观世界中是多么的渺小，在各种为人处事中明白了人的主观是多么的复杂，思考了很多学到了很多，愿这些作为人生中踏入社会的一点资本。感谢时光的荏苒。

参考文献

[1] 张贤达.矩阵分析与应用[M].北京:清华大学出版社.2013.32-36,67-74,115-117, 209-263,285-323,563-619.

[2] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications[J]. SIAM review, 51(3):455–500, 2009.

[3] De Almeida A L F, Kibangou A Y. Distributed large-scale tensor decomposition[C].Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on. IEEE, 2014: 26-30

[4] Kang U, Papalexakis E, Harpale A, et al. Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries[C]//Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2012: 316-324.

[5] Rendle S. Context-aware ranking with factorization models[M]. Springer, 2011.

[6] Oseledets I V. Tensor-train decomposition[J]. SIAM Journal on Scientific Computing, 2011, 33(5): 2295-2317.

[7] Socher R, Chen D, Manning C D, et al. Reasoning with neural tensor networks for knowledge base completion[C]//Advances in neural information processing systems. 2013: 926-934.

[8] Rendle S, Schmidt-Thieme L. Pairwise interaction tensor factorization for personalized tag recommendation[C]//Proceedings of the third ACM international conference on Web search and data mining. ACM, 2010: 81-90.

[9] Cai J F, Candès E J, Shen Z. A singular value thresholding algorithm for matrix completion[J]. SIAM Journal on Optimization, 2010, 20(4): 1956-1982.

[10] Stephen Boyd ,Lieven Vandenberghe 著，王书宁，等译.凸优化[M]北京:清华大学出版社.2013.46-52,130-139,207-233.

[11] Mason L, Baxter J, Bartlett P L, et al. Boosting Algorithms as Gradient Descent[C]//NIPS. 1999: 512-518.

[12] Golub G H, Reinsch C. Singular value decomposition and least squares solutions[J]. Numerische mathematik, 1970, 14(5): 403-420.

[13] Cai J F, Candès E J, Shen Z. A singular value thresholding algorithm for matrix completion[J]. SIAM Journal on Optimization, 2010, 20(4): 1956-1982.

[14] Rasmus Munk Larsen. Propack-software for large and sparse svd calculations[C]. Available online., 2004.

[15] Kreyszig E. Introductory functional analysis with applications[M]. New York: wiley, 1989.

[16] Cheatham T, Fahmy A, Stefanescu D, et al. Bulk synchronous parallel computing—a paradigm for transportable software[M]//Tools and Environments for Parallel and Distributed Systems. Springer US, 1996: 61-76.

[17] Gropp W, Lusk E, Doss N, et al. A high-performance, portable implementation of the MPI message passing interface standard[J]. Parallel computing, 1996, 22(6): 789-828.

[18] Gabriel E, Fagg G E, Bosilca G, et al. Open MPI: Goals, concept, and design of a next generation MPI implementation[C]//European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting. Springer Berlin Heidelberg, 2004: 97-104.

[19] Daily J, Lewis R R. Using the global arrays toolkit to reimplement numpy for distributed computation[C]//Proceedings of the 10th Python in Science Conference. 2011.

[20] Li M, Andersen D G, Park J W, et al. Scaling Distributed Machine Learning with the Parameter Server[C]//OSDI. 2014, 14: 583-598.

外文文献翻译

外文文献原文

Pairwise Interaction Tensor Factorization For Personalized Tag Recommendation

Steffen Rendle
Department of Reasoning for Intelligence
The Institute of Scientific and Industrial Research
Osaka University, Japan
rendle@ar.sanken.osaka-u.ac.jp

Lars Schmidt-Thieme
Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany
schmidt-thieme@ismll.uni-hildesheim.de

ABSTRACT

Tagging plays an important role in many recent websites.Recommender systems can help to suggest a user the tags he might want to use for tagging a specific item. Factorization models based on the Tucker Decomposition (TD) model have been shown to provide high quality tag recommendations outperforming other approaches like PageRank, FolkRank, collaborative filtering, etc. The problem with TD models is the cubic core tensor resulting in a cubic runtime in the factorization dimension for prediction and learning.

In this paper, we present the factorization model PITF(Pairwise Interaction Tensor Factorization) which is a special case of the TD model with linear runtime both for learning and prediction. PITF explicitly models the pairwise interactions between users, items and tags. The model is learned with an adaption of the Bayesian personalized ranking (BPR) criterion which originally has been introduced for item recommendation. Empirically, we show on real world datasets that this model outperforms TD largely in

runtime and even can achieve better prediction quality. Besides our lab experiments, PITF has also won the ECML/PKDD Discovery Challenge 2009 for graph-based tag recommendation.

Keywords

Tag recommendation, Tensor factorization, Personalization,Recommender systems

INTRODUCTION

Tagging is an important feature of the Web 2.0. It allows the user to annotate items/ resources like songs, pictures,bookmarks, etc. with keywords. Tagging helps the user to organize his items and facilitate e.g. browsing and searching.Tag recommenders assist the tagging process of a user by suggesting him a set of tags that he is likely to use for an item. Personalized tag recommenders take the user’s tagging behaviour in the past into account when they recommend tags. That means each user is recommended a personalized list of tags – i.e. the suggested list of tags depends both on the user and the item. Personalization makes sense as people tend to use different tags for tagging the same item. This can be seen in systems like Last.fm that have a nonpersonalized tag recommender but still the people use different tags. In [18] an empirical example was shown where recent personalized tag recommenders outperform even the theoretical upper-bound for any non-personalized tag recommender.

This work builds on the recent personalized tag recommender models using factorization models. These models like Higher-Order-Singular-Value-Decomposition (HOSVD)[22] and Ranking Tensor Factorization (RTF) [18] are based on the Tucker Decomposition (TD) model. RTF has shown to result in very good prediction quality. The drawback of using full TD is that the model equation is cubic in the factorization dimension. That makes TD models using a high factorization dimension unfeasible for midsized and large datasets. In this paper, we present a new factorization model that explicitly models the pairwise interactions between users, items and tags. The advantage of this model is that the complexity of the model equation is linear in the number of factorization dimensions which makes it feasible for high dimensions. In statistics, another approach for tensor factorization with a model equation of linear complexity is the canonical decomposition (CD) [1] – aka parallel factor analysis (PARAFAC) [2]. We will show that our model is a special case of both CD and TD. Our experimental results also indicate that our pairwise interaction model clearly outperforms the CD

model in prediction quality and slightly in runtime. Furthermore for learning tag recommender models in general, we adapt the Bayessian Personalized Ranking optimization criterion (BPR-Opt) [17] from item recommendation to tag recommendation.

In all, our contributions are as follows:

1.We extend the Bayessian Personalized Ranking optimization criterion (BPR-Opt) [17] to the task of tag recommendation and provide a learning algorithm based on stochastic gradient descent with bootstrap sampling. This optimization criterion and learning algorithm is generic and not limited to factorization models like TD.

2.We provide the factorization model PITF with a linear prediction/ reconstruction runtime. We show the relationship to the general Tucker Decomposition (TD) model and the canonical decomposition (CD; aka PARAFAC).

3.Our experiments indicate that our method BPR-PITF outperfoms the best quality method RTF-TD largely in runtime as the runtime drops from O(k3) to O(k) —where k is the factorization dimension. Moreover, the quality of BPR-PITF is comparable to RTF-TD on the Bibsonomy dataset and even outperforms RTF-TD on

the larger Last.fm dataset.

...

FACTORIZATION MODELS

Factorization models are a very successful model class for recommender systems. E.g. many of the best performing models [10, 11] on the Netflix Challenge2 for rating prediction are based on matrix factorization. Also for the related task of item prediction, factorization models are known [20,6, 16, 17] to outperform models like k-nearest-neighbour collaborative filtering or the Bayesian models URP [15] and PLSA [4]. Also for tag recommendation recent results [18, 22] indicate that factorization models generate high quality predictions outperforming other approaches like Folkrank and adapted Pagerank [7]. In contrast to factorization models in two dimensions (matrix factorization), in tag recommendation there are many possibilities for factorizing the data. To the best of our knowledge, in tag recommendation only models based on Tucker decomposition have been analyzed yet [18, 22]. In the following, we describe three factorization models for tag recommendation: Tucker decomposition (TD), Canonical decomposition (DC) and our pairwise interaction tensor factorization model (PITF) (see figure 4). We will show for each model how it can be learned with BPR and

the relationships to the other models. All of our factorization models predict a scoring function which can be seen as a three-dimensional tensor Y where the value of entry (u, i, t) is the score ˆyu,i,t. That means for ranking within a post, we sort the tags with respect to ˆyu,i,t. And thus for applying BPR optimization,we set:

$$\hat{y}_{u,i,t_A,t_B} := \hat{y}_{u,i,t_A} - \hat{y}_{u,i,t_B}$$

Tucker Decomposition (TD) model

Tucker Decomposition [23] factorizes a higher-order cube into a core tensor and one factor matrix for each dimensions.

$$y^{TD}_{u,i,t} = \sum_{\tilde{u}} \sum_{\tilde{i}} \sum_{\tilde{t}} \hat{c}_{\tilde{u},\tilde{i},\tilde{t}} \cdot \hat{u}_{u,\tilde{u}} \cdot \hat{i}_{i,\tilde{i}} \cdot \hat{t}_{t,\tilde{t}}$$

or equivalently as tensor product :

$$Y^{TD} := \hat{C} \times_u \hat{U} \times_i \hat{I} \times_t \hat{T}$$

with model parameters:

$$\hat{C} \in R^{k_u \times k_i \times k_t}, \hat{U} \in R^{|U| \times k_u}$$
$$\hat{I} \in R^{|I| \times k_i}, \hat{T} \in R^{|T| \times k_t}$$

For learning such a TD model with BPR-Opt, the gradients

$$\frac{\partial \hat{Y}^{TD}}{\partial \hat{\theta}}$$

are:

$$\frac{\partial \hat{Y}^{TD}_{u,i,t}}{\partial \hat{c}_{\tilde{u},\tilde{i},\tilde{t}}} = \hat{u}_{u,\tilde{u}} \cdot \hat{i}_{i,\tilde{i}} \cdot \hat{t}_{t,\tilde{t}}$$
$$\frac{\partial \hat{Y}^{TD}_{u,i,t}}{\partial \hat{u}_{u,\tilde{u}}} = \sum_{\tilde{i}} \sum_{\tilde{t}} \hat{c}_{\tilde{u},\tilde{i},\tilde{t}} \cdot \hat{i}_{i,\tilde{i}} \cdot \hat{t}_{t,\tilde{t}}$$
$$\frac{\partial \hat{Y}^{TD}_{u,i,t}}{\partial \hat{i}_{i,\tilde{i}}} = \sum_{\tilde{u}} \sum_{\tilde{t}} \hat{c}_{\tilde{u},\tilde{i},\tilde{t}} \cdot \hat{u}_{u,\tilde{u}} \cdot \hat{t}_{t,\tilde{t}}$$
$$\frac{\partial \hat{Y}^{TD}_{u,i,t}}{\partial \hat{t}_{t,\tilde{t}}} = \sum_{\tilde{u}} \sum_{\tilde{i}} \hat{c}_{\tilde{u},\tilde{i},\tilde{t}} \cdot \hat{u}_{u,\tilde{u}} \cdot \hat{i}_{i,\tilde{i}}$$

An obvious drawback of TD is that the model equation is a nested sum of degree 3 – i.e. it is cubic in k := min(ku, ki, kt) and so the runtime complexity for predicting one triple (u, i, t) is O(k3). Thus learning a TD model is slow even for a small to mid-sized number of factorization dimensions.

Canonical Decomposition (CD) model

The CD model (Canonical Decomposition) is a special case of the general Tucker Decomposition model.

$$\hat{y}^{CD}_{u,i,t} = \sum_{f}^{k} \hat{u}_{u,f} \cdot \hat{i}_{i,f} \cdot \hat{t}_{t,f}$$

It can be derived from the Tucker Decomposition model by setting $\hat{C}$ to the diagonal tensor:

$$\hat{c}_{\tilde{u},\tilde{i},\tilde{t}} = \begin{cases} 1, \text{if} \quad \tilde{u}=\tilde{i}=\tilde{t} \\ 0, \qquad else \end{cases}$$

Obviously, only the first k := min{ku, ki, kt} features are used — i.e. if the dimensionality of the feature matrices differ, some features are not used, as the core will be 0 for these entries.

The gradients for this model are:

$$\frac{\partial \hat{Y}^{CD}_{u,i,t}}{\partial \hat{u}_{u,f}} = \hat{i}_{i,f} \cdot \hat{t}_{t,f}$$

$$\frac{\partial \hat{Y}^{CD}_{u,i,t}}{\partial \hat{i}_{i,f}} = \hat{u}_{u,f} \cdot \hat{t}_{t,f}$$

$$\frac{\partial \hat{Y}^{CD}_{u,i,t}}{\partial \hat{t}_{t,f}} = \hat{u}_{u,f} \cdot \hat{i}_{i,f}$$

Obviously, the CD model has a much better runtime complexity as the model equation contains no nested sums and thus is in O(k).

Pairwise Interaction Tensor Factorization(PITF) model

Our approach explicitly models the two-way interactions between users, tags and items by factorizing each of the three relationships:

$$\hat{y}_{u,i,t} = \sum_{f} \hat{u}^{T}_{u,f} \cdot \hat{t}^{U}_{t,f} + \sum_{f} \hat{i}^{T}_{i,f} \cdot \hat{t}^{I}_{t,f} + \sum_{f} \hat{u}^{I}_{u,f} \cdot \hat{i}^{U}_{i,f}$$

The user-item interaction vanishes for predicting rankings and for BPR optimization. The reason is that given a post (u, i), both the optimization criterion BPR and the ranking ignores any score on the user-item interaction. This results in our final model equation that we will refer to as the PITF (Pairwise Interaction Tensor Factorization) model:

$$\hat{y}_{u,i,t} = \sum_{f} \hat{u}_{u,f} \cdot \hat{t}^{U}_{t,f} + \sum_{f} \hat{i}_{i,f} \cdot \hat{t}^{I}_{t,f}$$

with model parameters:

$$\hat{C} \in R^{k_u \times k_i \times k_t}, \hat{U} \in R^{|U| \times k_u}$$
$$\hat{I} \in R^{|I| \times k_i}, \hat{T} \in R^{|T| \times k_t}$$

Again, the runtime for predicting a triple (u, i, t) is in O(k).

PITF is a special case of the CD model with dimensionality2 k where:

$$\hat{u}_{u,f}^{CD} = \begin{cases} \hat{u}_{u,f}, \text{if} \quad \text{f} \le \text{k} \\ 1, \qquad else \end{cases}$$

$$\hat{i}_{u,f}^{CD} = \begin{cases} 1, \qquad \text{if} \quad \text{f} \le \text{k} \\ \hat{i}_{i,f-k}, \qquad else \end{cases}$$

$$\hat{t}_{u,f}^{CD} = \begin{cases} \hat{t}_{t,f}^{U}, \text{if} \quad \text{f} \le \text{k} \\ \hat{t}_{t,f-k}^{I}, \qquad else \end{cases}$$

The gradients for the PITF model are:

$$\frac{\partial \hat{y}_{u,i,t}}{\partial \hat{u}_{u,f}} = \hat{t}_{t,f}^{U}, \frac{\partial \hat{y}_{u,i,t}}{\partial \hat{i}_{i,f}} = \hat{t}_{t,f}^{I}$$

$$\frac{\partial \hat{y}_{u,i,t}}{\partial \hat{t}_{t,f}^{U}} = \hat{u}_{u,f}, \frac{\partial \hat{y}_{u,i,t}}{\partial \hat{t}_{t,f}^{I}} = \hat{i}_{u,f}$$

外文文献截图

Pairwise Interaction Tensor Factorization for Personalized Tag Recommendation

Steffen Rendle*
Department of Reasoning for Intelligence
The Institute of Scientific and Industrial Research
Osaka University, Japan
rendle@ar.sanken.osaka-u.ac.jp

Lars Schmidt-Thieme
Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany
schmidt-thieme@ismll.uni-hildesheim.de

ABSTRACT

Tagging plays an important role in many recent websites. Recommender systems can help to suggest a user the tags he might want to use for tagging a specific item. Factorization models based on the Tucker Decomposition (TD) model have been shown to provide high quality tag recommendations outperforming other approaches like PageRank, FolkRank, collaborative filtering, etc. The problem with TD models is the cubic core tensor resulting in a cubic runtime in the factorization dimension for prediction and learning.

In this paper, we present the factorization model PITF (Pairwise Interaction Tensor Factorization) which is a special case of the TD model with linear runtime both for learning and prediction. PITF explicitly models the pairwise interactions between users, items and tags. The model is learned with an adaption of the Bayesian personalized ranking (BPR) criterion which originally has been introduced for item recommendation. Empirically, we show on real world datasets that this model outperforms TD largely in runtime and even can achieve better prediction quality. Besides our lab experiments, PITF has also won the ECML/PKDD Discovery Challenge 2009 for graph-based tag recommendation.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning*

General Terms

Algorithms, Experimentation, Measurement, Performance

Keywords

Tag recommendation, Tensor factorization, Personalization, Recommender systems

*Steffen Rendle is currently on leave from the Machine Learning Lab, University of Hildesheim, Germany.

WSDM'10, February 4–6, 2010, New York City, New York, USA.

1. INTRODUCTION

Tagging is an important feature of the Web 2.0. It allows the user to annotate items/ resources like songs, pictures, bookmarks, etc. with keywords. Tagging helps the user to organize his items and facilitate e.g. browsing and searching. Tag recommenders assist the tagging process of a user by suggesting him a set of tags that he is likely to use for an item. Personalized tag recommenders take the user's tagging behaviour in the past into account when they recommend tags. That means each user is recommended a personalized list of tags – i.e. the suggested list of tags depends both on the user and the item. Personalization makes sense as people tend to use different tags for tagging the same item. This can be seen in systems like Last.fm that have a non-personalized tag recommender but still the people use different tags. In [18] an empirical example was shown where recent personalized tag recommenders outperform even the theoretical upper-bound for any non-personalized tag recommender.

This work builds on the recent personalized tag recommender models using factorization models. These models like Higher-Order-Singular-Value-Decomposition (HOSVD) [22] and Ranking Tensor Factorization (RTF) [18] are based on the Tucker Decomposition (TD) model. RTF has shown to result in very good prediction quality. The drawback of using full TD is that the model equation is cubic in the factorization dimension. That makes TD models using a high factorization dimension unfeasible for midsized and large datasets. In this paper, we present a new factorization model that explicitly models the pairwise interactions between users, items and tags. The advantage of this model is that the complexity of the model equation is linear in the number of factorization dimensions which makes it feasible for high dimensions. In statistics, another approach for tensor factorization with a model equation of linear complexity is the canonical decomposition (CD) [1] – aka parallel factor analysis (PARAFAC) [2]. We will show that our model is a special case of both CD and TD. Our experimental results also indicate that our pairwise interaction model clearly outperforms the CD model in prediction quality and slightly in runtime. Furthermore for learning tag recommender models in general, we adapt the Bayessian Personalized Ranking optimization criterion (BPR-Opt) [17] from item recommendation to tag recommendation.

In all, our contributions are as follows:

1. We extend the Bayessian Personalized Ranking optimization criterion (BPR-Opt) [17] to the task of

That means, to apply LEARNBPR to a given model, only the gradient $\frac{\partial}{\partial \Theta}\hat{y}_{u,i,t_A,t_B}$ has to be computed. In the next section, we derive our factorization models and also show their gradients for optimization w.r.t. BPR-OPT with LEARNBPR.

1: **procedure** LEARNBPR(D_S, Θ)
2: initialize Θ
3: **repeat**
4: draw (u, i, t_A, t_B) uniformly from D_S
5: $\Theta \leftarrow \Theta + \alpha \frac{\partial}{\partial \Theta}\left(\ln \sigma(\hat{y}_{u,i,t_A,t_B}) - \lambda_\Theta \|\Theta\|_F^2\right)$
6: **until** convergence
7: **return** $\hat{\Theta}$
8: **end procedure**

Figure 3: Optimizing tag recommender models for BPR with bootstrapping based stochastic gradient descent. With learning rate α and regularization λ_Θ.

5. FACTORIZATION MODELS

Factorization models are a very successful model class for recommender systems. E.g. many of the best performing models [10, 11] on the Netflix Challenge[2] for rating prediction are based on matrix factorization. Also for the related task of item prediction, factorization models are known [20, 6, 16, 17] to outperform models like k-nearest-neighbour collaborative filtering or the Bayesian models URP [15] and PLSA [4]. Also for tag recommendation recent results [18, 22] indicate that factorization models generate high quality predictions outperforming other approaches like Folkrank and adapted Pagerank [7]. In contrast to factorization models in two dimensions (matrix factorization), in tag recommendation there are many possibilities for factorizing the data. To the best of our knowledge, in tag recommendation only models based on Tucker decomposition have been analyzed yet [18, 22].

In the following, we describe three factorization models for tag recommendation: Tucker decomposition (TD), Canonical decomposition (DC) and our pairwise interaction tensor factorization model (PITF) (see figure 4). We will show for each model how it can be learned with BPR and the relationships to the other models.

All of our factorization models predict a scoring function $\hat{Y} : U \times I \times T \to \mathbb{R}$ which can be seen as a three-dimensional tensor Y where the value of entry (u, i, t) is the score $\hat{y}_{u,i,t}$. That means for ranking within a post, we sort the tags with respect to $\hat{y}_{u,i,t}$. And thus for applying BPR optimization, we set:

$$\hat{y}_{u,i,t_A,t_B} := \hat{y}_{u,i,t_A} - \hat{y}_{u,i,t_B}$$

5.1 Tucker Decomposition (TD) model

Tucker Decomposition [23] factorizes a higher-order cube into a core tensor and one factor matrix for each dimensions.

$$\hat{y}^{\mathrm{TD}}_{u,i,t} := \sum_{\tilde{u}}\sum_{\tilde{i}}\sum_{\tilde{t}} \hat{c}_{\tilde{u},\tilde{i},\tilde{t}} \cdot \hat{u}_{u,\tilde{u}} \cdot \hat{i}_{i,\tilde{i}} \cdot \hat{t}_{t,\tilde{t}} \qquad (9)$$

or equivalently as tensor product (see figure 4):

$$\hat{Y}^{\mathrm{TD}} := \hat{C} \times_{\mathrm{U}} \hat{U} \times_{\mathrm{I}} \hat{I} \times_{\mathrm{T}} \hat{T} \qquad (10)$$

[2] http://www.netflixprize.com/

with model parameters:

$$\hat{C} \in \mathbb{R}^{k_u \times k_i \times k_t}, \quad \hat{U} \in \mathbb{R}^{|U| \times k_u}$$
$$\hat{I} \in \mathbb{R}^{|I| \times k_i}, \quad \hat{T} \in \mathbb{R}^{|T| \times k_t}$$

For learning such a TD model with BPR-OPT, the gradients $\frac{\partial \hat{Y}^{\mathrm{TD}}}{\partial \Theta}$ are:

$$\frac{\partial \hat{y}^{\mathrm{TD}}_{u,i,t}}{\partial \hat{c}_{\tilde{u},\tilde{i},\tilde{t}}} = \hat{u}_{u,\tilde{u}} \cdot \hat{i}_{i,\tilde{i}} \cdot \hat{t}_{t,\tilde{t}}$$
$$\frac{\partial \hat{y}^{\mathrm{TD}}_{u,i,t}}{\partial \hat{u}_{u,\tilde{u}}} = \sum_{\tilde{i}}\sum_{\tilde{t}} \hat{c}_{\tilde{u},\tilde{i},\tilde{t}} \cdot \hat{i}_{i,\tilde{i}} \cdot \hat{t}_{t,\tilde{t}}$$
$$\frac{\partial \hat{y}^{\mathrm{TD}}_{u,i,t}}{\partial \hat{i}_{i,\tilde{i}}} = \sum_{\tilde{u}}\sum_{\tilde{t}} \hat{c}_{\tilde{u},\tilde{i},\tilde{t}} \cdot \hat{u}_{u,\tilde{u}} \cdot \hat{t}_{t,\tilde{t}}$$
$$\frac{\partial \hat{y}^{\mathrm{TD}}_{u,i,t}}{\partial \hat{t}_{t,\tilde{t}}} = \sum_{\tilde{u}}\sum_{\tilde{i}} \hat{c}_{\tilde{u},\tilde{i},\tilde{t}} \cdot \hat{u}_{u,\tilde{u}} \cdot \hat{i}_{i,\tilde{i}}$$

An obvious drawback of TD is that the model equation is a nested sum of degree 3 – i.e. it is cubic in $k := \min(k_u, k_i, k_t)$ and so the runtime complexity for predicting one triple (u, i, t) is $O(k^3)$. Thus learning a TD model is slow even for a small to mid-sized number of factorization dimensions.

5.2 Canonical Decomposition (CD) model

The CD model (Canonical Decomposition) is a special case of the general Tucker Decomposition model.

$$\hat{y}^{\mathrm{CD}}_{u,i,t} := \sum_{f}^{k} \hat{u}_{u,f} \cdot \hat{i}_{i,f} \cdot \hat{t}_{t,f} \qquad (11)$$

It can be derived from the Tucker Decomposition model by setting $\hat{C}$ to the diagonal tensor:

$$\hat{c}_{\tilde{u},\tilde{i},\tilde{t}} = \begin{cases} 1, & \text{if } \tilde{u} = \tilde{i} = \tilde{t} \\ 0, & \text{else} \end{cases}$$

Obviously, only the first $k := \min\{k_u, k_i, k_t\}$ features are used — i.e. if the dimensionality of the feature matrices differ, some features are not used, as the core will be 0 for these entries.

The gradients for this model are:

$$\frac{\partial \hat{y}^{\mathrm{CD}}_{u,i,t}}{\partial \hat{u}_{u,f}} = \hat{i}_{i,f} \cdot \hat{t}_{t,f}$$
$$\frac{\partial \hat{y}^{\mathrm{CD}}_{u,i,t}}{\partial \hat{i}_{i,f}} = \hat{u}_{u,f} \cdot \hat{t}_{t,f}$$
$$\frac{\partial \hat{y}^{\mathrm{CD}}_{u,i,t}}{\partial \hat{t}_{t,f}} = \hat{u}_{u,f} \cdot \hat{i}_{i,f}$$

Obviously, the CD model has a much better runtime complexity as the model equation contains no nested sums and thus is in $O(k)$.

5.3 Pairwise Interaction Tensor Factorization (PITF) model

Our approach explicitly models the two-way interactions between users, tags and items by factorizing each of the three relationships:

$$\hat{y}_{u,i,t} = \sum_{f} \hat{u}^{T}_{u,f} \cdot \hat{t}^{U}_{t,f} + \sum_{f} \hat{i}^{T}_{i,f} \cdot \hat{t}^{I}_{t,f} + \sum_{f} \hat{u}^{I}_{u,f} \cdot \hat{i}^{U}_{i,f} \qquad (12)$$

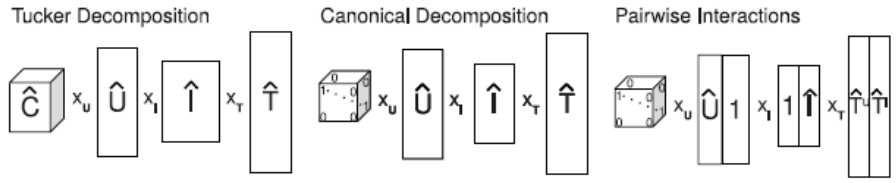


Figure 4: Tensor Factorization models: $\hat{C}$, $\hat{U}$, $\hat{I}$ and $\hat{T}$ are the model parameters (one tensor, three matrices). In Tucker Decomposition the core $\hat{C}$ is variable and the factorization dimensions can differ. For Canonical Decomposition and Pairwise Interactions the core is a fixed diagonal tensor. In Pairwise Interaction parts of the feature matrices are fixed which corresponds modelling pairwise interactions.

The user-item interaction vanishes for predicting rankings and for BPR optimization. The reason is that given a post (u, i), both the optimization criterion BPR and the ranking ignores any score on the user-item interaction. This results in our final model equation that we will refer to as the PITF (Pairwise Interaction Tensor Factorization) model:

$$\hat{y}_{u,i,t} = \sum_f \hat{u}_{u,f} \cdot \hat{t}^U_{t,f} + \sum_f \hat{i}_{i,f} \cdot \hat{t}^I_{t,f} \qquad (13)$$

with model parameters:

$$\hat{U} \in \mathbb{R}^{|U| \times k}, \quad \hat{I} \in \mathbb{R}^{|I| \times k},$$
$$\hat{T}^U \in \mathbb{R}^{|T| \times k}, \quad \hat{T}^I \in \mathbb{R}^{|T| \times k}$$

Again, the runtime for predicting a triple (u, i, t) is in $O(k)$.

PITF is a special case of the CD model with dimensionality $2 \cdot k$ where:

$$\hat{u}^{\text{CD}}_{u,f} = \begin{cases} \hat{u}_{u,f}, & \text{if } f \leq k \\ 1, & \text{else} \end{cases}$$

$$\hat{i}^{\text{CD}}_{i,f} = \begin{cases} 1, & \text{if } f \leq k \\ \hat{i}_{i,f-k}, & \text{else} \end{cases}$$

$$\hat{t}^{\text{CD}}_{u,f} = \begin{cases} \hat{t}^U_{t,f}, & \text{if } f \leq k \\ \hat{t}^I_{t,f-k}, & \text{else} \end{cases}$$

The gradients for the PITF model are:

$$\frac{\partial \hat{y}_{u,i,t}}{\partial \hat{u}_{u,f}} = \hat{t}^U_{t,f}, \quad \frac{\partial \hat{y}_{u,i,t}}{\partial \hat{i}_{i,f}} = \hat{t}^I_{t,f},$$
$$\frac{\partial \hat{y}_{u,i,t}}{\partial \hat{t}^U_{t,f}} = \hat{u}_{u,f}, \quad \frac{\partial \hat{y}_{u,i,t}}{\partial \hat{t}^I_{t,f}} = \hat{i}_{u,f}$$

The complete BPR learning algorithm for PITF can be found in figure 5.

5.4 Relation between TD, CD and PITF

We have shown the relationships of our proposed PITF model to both the CD and TD model class. Obviously, the expressiveness of the model classes is:

$$\mathcal{M}^{\text{TD}} \supset \mathcal{M}^{\text{CD}} \supset \mathcal{M}^{\text{PITF}}$$

At first glance, one might think that reducing the expressiveness leads to worse prediction quality — i.e. that quality is traded in for e.g. runtime. But actually, our evaluation

1: **procedure** LEARNBPR-PITF($P_S, \hat{U}, \hat{I}, \hat{T}^U, \hat{T}^I$)
2: draw $\hat{U}, \hat{I}, \hat{T}^U, \hat{T}^I$ from $N(\mu, \sigma^2)$
3: **repeat**
4: draw (u, i, t_A, t_B) uniformly from D_S
5: $\hat{y}_{u,i,t_A,t_B} \leftarrow \hat{y}_{u,i,t_A} - \hat{y}_{u,i,t_B}$
6: $\delta \leftarrow (1 - \sigma(\hat{y}_{u,i,t_A,t_B}))$
7: **for** $f \in 1, \ldots, k$ **do**
8: $\hat{u}_{u,f} \leftarrow \hat{u}_{u,f} + \alpha \left(\delta \cdot (\hat{t}^U_{t_A,f} - \hat{t}^U_{t_B,f}) - \lambda \cdot \hat{u}_{u,f}\right)$
9: $\hat{i}_{i,f} \leftarrow \hat{i}_{i,f} + \alpha \left(\delta \cdot (\hat{t}^I_{t_A,f} - \hat{t}^I_{t_B,f}) - \lambda \cdot \hat{i}_{i,f}\right)$
10: $\hat{t}^U_{t_A,f} \leftarrow \hat{t}^U_{t_A,f} + \alpha \left(\delta \cdot \hat{u}_{u,f} - \lambda \cdot \hat{t}^U_{t_A,f}\right)$
11: $\hat{t}^U_{t_B,f} \leftarrow \hat{t}^U_{t_B,f} + \alpha \left(-\delta \cdot \hat{u}_{u,f} - \lambda \cdot \hat{t}^U_{t_B,f}\right)$
12: $\hat{t}^I_{t_A,f} \leftarrow \hat{t}^I_{t_A,f} + \alpha \left(\delta \cdot \hat{i}_{i,f} - \lambda \cdot \hat{t}^I_{t_A,f}\right)$
13: $\hat{t}^I_{t_B,f} \leftarrow \hat{t}^I_{t_B,f} + \alpha \left(-\delta \cdot \hat{i}_{i,f} - \lambda \cdot \hat{t}^I_{t_B,f}\right)$
14: **end for**
15: **until** convergence
16: **return** $\hat{U}, \hat{I}, \hat{T}^U, \hat{T}^I$
17: **end procedure**

Figure 5: Optimizing the PITF model with LearnBPR.

shows that this is not always the case. The reason is that our PI approach explicitly models a structure that might be hard to find for the TD and CD approach. Especially, regularization approaches like ridge regression which usually assume that the model parameters are normally distributed with mean zero $\Theta \sim N(0, \sigma^2_\Theta I)$ might fail to learn the structure modeled explicitly. Thus, if a model structure is known a priori, it might be better to model it explicitly than trying to learn it.

6. EVALUATION

In our evaluation, we investigate the learning runtime and prediction quality of our proposed PITF model. For the runtime, we want to justify the results of the theoretical complexity analysis (TD is in $O(k^3)$, CD/PITF in O(k)) by an empirical comparison of the TD model to the CD/PARAFAC model and our PITF model. With respect to prediction quality, we investigate empirically whether the speedup of CD/ PITF is paid with quality – i.e. if there is a trade-off between quality and runtime between the model classes.

个人化标签推荐系统的成对互动张量因子分解

外文中文翻译

摘要

最近标签在许多网站中起着重要作用。推荐系统可以帮助用户建议他可能偏向标记特定项目的标签。基于 Tucker 分解(TD)模型的因子分解模型已经被证明可以提供优于其他方法(例如 PageRank，FolkRank，协同过滤等)的高质量标签建议。TD 模型的主要是三阶核心张量形式做因子分解用于预测和学习。

在本文中，我们提出了分解模型 PITF(成对互动张量因子分解)，这是 TD 模型的特殊形式，且在线性运行时间中用于学习和预测。PITF 明确地模拟用户，项目，标签之间的成对交互。该模型通过适应最初作为项目推荐引入的贝叶斯个性化排名标准而被学习。实际上，我们在现实世界的数据集上显示，这种模型在运行时大部分优于 TD 模型，甚至可以实现更好的预测质量。除了在我们实验室试验后，PITF 还赢得了 2009 年基于图形推荐标签推荐的 ECML/PKDD 挑战比赛。

关键词 标签推荐，张量分解，个性化，推荐系统

介绍

标签是 Web 2.0 的一个重要功能。它允许用户使用关键字来注释诸如歌曲，图片，书签等的项目/资源。标签帮助用户组织他的物品，并促进例如浏览和搜索.标签推荐系统通过向他建议一组他可能用于某个项目的标签来协助用户的标记过程。个性化标签推荐者在推荐标签时，会将用户的标记行为纳入考虑范围。这意味着每个用户都建议使用个性化的标签列表，即建议的标签列表取决于用户和项目。个性化是有意义的，因为人们倾向于使用不同的标签来标记相同的项目。这可以在像 Last.fm 这样的系统中看到，这些系统有一个非个性化的标签推荐器，但是人们仍然使用不同的标签。在[18]中，显示了一个经验示例，其中最近的个性化标签推荐者比任何非个性化标签推荐者的理论上限都优于大部分。

这项工作建立在使用因式分解模型的最近的个性化标签推荐模型上。这些模型，如高阶奇异值分解（HOSVD）[22]和排名张量因子分解（RTF）[18]基于塔克分解（TD）模型。 RTF 显示出非常好的预测质量。使用全 TD 的缺点是模型方程在因式分解维度上是立方的。这使得使用高分解尺度的 TD 模型对中型和大型数据集是不可行的。在本文中，我们提出一个新的因式分解模型，明确地模拟用户，项目和标签之间的成对互动。该模型的优点在于，模型方程的复杂度在分解维数

方面是线性的，这使得其可以用于高维度。统计学中，用线性复杂度模型方程的张量因式分解的另一种方法是规范分解（CD）[1] - 又称并行因子分析（PARAFAC）[2]。我们将展示我们的模式是 CD 和 TD 的一个特例。我们的实验结果也表明，我们的成对交互模型明显优于 CD 模型的预测质量，稍微在运行时。此外，对于一般的学习标签推荐模型，我们将 Bayesian 个性化排名优化标准（BPR-Opt）[17]从项目推荐与标签推荐相结合。

总而言之，我们的贡献如下：

1.我们将 Bayessian 个性化排名优化标准（BPR-Opt）[17]扩展到标签推荐任务，并提供基于随机梯度下降和自举抽样的学习算法。 该优化标准和学习算法是通用的，不限于因式分解像 TD 这样的模型。

2.我们用线性预测/重建运行时间提供分解模型 PITF。我们显示与普通塔克分解（TD）模型和规范分解（CD;又名 PARAFAC）的关系。

3.我们的实验表明，随着运行时间从 O（k3）下降到 O（k），我们的方法 BPR-PITF 在运行时间内大大超出了质量最好的方法 RTF-TD，其中 k 是分解维度。 此外，BPR-PITF 的质量与 Bibsonomy 数据集上的 RTF-TD 相当，甚至优于 RTF-TD 较大的 Last.fm 数据集。

。。。

分解模型

因式分解模型是推荐系统非常成功的模型。例如。 Netflix Challenge2 评估预测中的许多最佳表现模型[10,11]都是基于矩阵分解。同样对于项目预测的相关任务，分解模型已知[20,6,16,17]优于 k-最近邻协同过滤或贝叶斯模型 URP [15]和 PLSA [4]等模型。此外，对于标签推荐，最近的结果[18,22]表明，分解模型产生的高质量预测优于其他方法，如 Folkrank 和适应的 Pagerank [7]。与二维分解模型（矩阵分解）相反，在标签推荐中，有许多将数据分解的可能性。据我们所知，在标签推荐中，仅分析了基于 Tucker 分解的模型[18,22]。在下文中，我们描述了标签推荐的三个因式分解模型：塔克分解（TD），规范分解（DC）和我们的成对互动张量因子分解模型（PITF）（见图 4）。我们将为每个模型展示如何使用 BPR 学习以及与其他模型的关系。我们所有的因式分解模型预测了一个评分函数，可以看作三维张量 Y，其中入口值（u，i，t）是分数 yu，i，t。这意味着在帖子中排名，我们对 yu，i，t 分类标签。

Tucker 分解（TD）模型

Tucker 分解[23]将高阶多维数据因子分解为核心张量和每个维度的一个因子

矩阵。或等价于张量乘积： 模型参数为:用 BPR-Opt 学习的 TD 模型，梯度

是：TD 的一个明显的缺点是模型方程是 3 级的嵌套和，即它是 k：= min（ku，ki，kt）中的立方，因此预测一个三元组（u，i，t）的运行时复杂度 是 O（k3）。因此，即使对于小到中等数量的因式分解维度，学习 TD 模型也很慢。

Canonical Decomposition (CD)模型

CD 模型（Canonical Decomposition）是一般塔克分解模型的特例。

$$\hat{y}_{u,i,t} = \sum_f \hat{u}_{u,f} \cdot \hat{t}^{\,U}_{t,f} + \sum_f \hat{i}_{i,f} \cdot \hat{t}^{I}_{t,f}$$

它可以通过设置到对角张量从 Tucker 分解模型得出：

显然，仅使用第一个 k：= min {ku，ki，kt}特征 即如果特征矩阵的维度不同，则不使用某些特征，因为这些条目的核心将为 0。

这个模型的梯度为：

$$\frac{\partial \hat{Y}^{CD}_{u,i,t}}{\partial \hat{u}_{u,f}} = \hat{i}_{i,f} \cdot \hat{t}_{t,f}$$

$$\frac{\partial \hat{Y}^{CD}_{u,i,t}}{\partial \hat{i}_{i,f}} = \hat{u}_{u,f} \cdot \hat{t}_{t,f}$$

$$\frac{\partial \hat{Y}^{CD}_{u,i,t}}{\partial \hat{t}_{t,f}} = \hat{u}_{u,f} \cdot \hat{i}_{i,f}$$

显然，CD 模型具有更好的运行时复杂度，因为模型方程不包含嵌套和，因此在 O（k）中。

成对互动张量因子分解（PITF）模型

我们的方法通过对三种关系中的每一种进行分解来明确地模拟用户，标签和项目之间的双向交互：

用户项目交互消失用于预测排名和 BPR 优化。原因是，给定一个帖子（u，i），优化标准 BPR 和排名都忽略用户项目交互的任何分数。这导致我们的最终模型方

$$\hat{y}_{u,i,t} = \sum_f \hat{u}^{T}_{u,f} \cdot \hat{t}^{\,U}_{t,f} + \sum_f \hat{i}^{T}_{i,f} \cdot \hat{t}^{I}_{t,f} + \sum_f \hat{u}^{I}_{u,f} \cdot \hat{i}^{U}_{i,f}$$

程，我们将被称为 PITF（成对互动张量因子分解）模型:

模型参数：

$$\hat{C} \in R^{k_u \times k_i \times k_t}, \hat{U} \in R^{|U| \times k_u}$$
$$\hat{I} \in R^{|I| \times k_i}, \hat{T} \in R^{|T| \times k_t}$$

再次，用于预测三元组（u，i，t）的运行时间在 O（k）中。PITF 是具有维度 2 k 的 CD 模型的特殊情况 PITF 模型的导数为：

$$\hat{u}_{u,f}^{CD} = \begin{cases} \hat{u}_{u,f}, \text{if} \quad \text{f} \le \text{k} \\ 1, \qquad else \end{cases}$$
$$\hat{i}_{u,f}^{CD} = \begin{cases} 1, \qquad \text{if} \quad \text{f} \le \text{k} \\ \hat{i}_{i,f-k}, \qquad else \end{cases}$$
$$\hat{t}_{u,f}^{CD} = \begin{cases} \hat{t}_{t,f}^{U}, \text{if} \quad \text{f} \le \text{k} \\ \hat{t}_{t,f-k}^{I}, \qquad else \end{cases}$$
$$\frac{\partial \hat{y}_{u,i,t}}{\partial \hat{u}_{u,f}} = \hat{t}_{t,f}^{U}, \frac{\partial \hat{y}_{u,i,t}}{\partial \hat{i}_{i,f}} = \hat{t}_{t,f}^{I}$$
$$\frac{\partial \hat{y}_{u,i,t}}{\partial \hat{t}_{t,f}^{U}} = \hat{u}_{u,f}, \frac{\partial \hat{y}_{u,i,t}}{\partial \hat{t}_{t,f}^{I}} = \hat{i}_{u,f}$$