浅谈张量在 OI 的运用

魏忠浩

福建师范大学附属中学

2025 年 1 月 14 日

前言

张量作为一种数学工具，其在 20 世纪中后期在物理学、工程学、计算机科学学等方面得到广泛运用。并成为深度学习核心概念，可以优化 TCS 的一些结论，但在 OI 里传播较少。
本文将引入张量，特别是三阶张量的秩和分解，并讲述若干例子来介绍其在一类分治题目的用处。
本文略去了一些定理证明。

目录

符号约定

若无特殊规定，下标从 1 开始。

定义 (张量)

N 阶张量 $\mathcal{X}$ 是 N 维数组 $F^{I_1\times I_2\times\cdots\times I_N}$，$(i_1,i_2\cdots,i_N)$ 对应元素记为 $\mathcal{X}_{i_1i_2\cdots i_N}(i_j\le I_j)$，$\mathcal{X}$ 可以小写。下标间可用逗号隔开。

在表示三维数组的时候，一般采用 $\left[\ A^{(1)}\ \middle|\ \cdots\ \middle|\ A^{(n)}\ \right]$ 的形式，(i) 指第一层下标，即 $A^{(i)}_{j,k}=A_{i,j,k}$。
为表示方便，一些特殊的三维数组还可以表示为“运算表”形式：$\begin{pmatrix} f_{1,1} & \cdots & f_{1,m} \\ \vdots & \ddots & \vdots \\ f_{n,1} & \cdots & f_{n,m} \end{pmatrix}$，此时三维数组 $\mathcal{X}_{i,j,k}=[f_{i,j}+\mathbf{1}=k]$，
允许 $f_{i,j}$ 为空，此时对于任意 k 均为 0。

符号约定

对于 N 阶张量 $\mathcal{B}=\mathcal{A}^{(n)}$，则 $b_{i_2i_3\cdots i_N}=a_{ni_2i_3\cdots i_N}$，表示其包含的第 n 个 $N-1$ 阶张量。也可表示数组元素，$A_j^{(i)}=(A^{(i)})_j$。

定义 (内积)

张量内积 $\langle\mathcal{X},\mathcal{Y}\rangle=\sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2}\cdots\sum_{i_N=N}^{I_N}x_{i_1i_2\cdots i_N}y_{i_1i_2\cdots i_N}$。

范数 $\|\mathcal{X}\|=\sqrt{\langle X,X\rangle}$。

定义 (外积 / 直积)

记 $\mathcal{Z}=\mathcal{X}\circ\mathcal{Y}$，则 $z_{i_1i_2\cdots i_Nj_1j_2\cdots j_M}=x_{i_1i_2\cdots i_N}y_{j_1j_2\cdots j_M}$。亦可记作 $\mathcal{Z}=\mathcal{X}\otimes\mathcal{Y}$。

符号约定

定义 (秩一张量)

称 $\mathcal{X}$ 为秩一张量，若存在长为 N 的向量序列 y，y_i 长度为 I_i，且 $\mathcal{X} = y^{(1)} \circ y^{(2)} \circ \cdots \circ y^{(N)}$，即 $x_{i_1 i_2 \cdots i_N} = y^{(1)}_{i_1} y^{(2)}_{i_2} \cdots y^{(N)}_{i_N}$。

定义 (CANDECOMP/PARAFAC(CP) 分解)

CP 分解是将张量 $\mathcal{X}$ 分解成若干个秩一张量的和。

在三阶张量里，可以写作 $\mathcal{X} = \sum_{i=1}^{R} \mathbf{A}^{(i)} \circ \mathbf{B}^{(i)} \circ \mathbf{C}^{(i)}$。

定义 (张量秩)

对于一个张量 $\mathcal{X}$，$\mathrm{rank}(\mathcal{X})$ 表示最小的 R，使得 $\mathcal{X}$ 能被分解成 R 个一阶张量的和。

定义

这里给出更为简洁的定义。

高维“卷积”

给出若干二维运算表 op_i，定义 $f_i(a,b):U_i\times V_i\to W_i$，其中 $f_i(a,b)_t=\sum_{\mathsf{op}_{i,j,k}=t}a_jb_k$。
记 $f=f_1\times f_2\times\cdots\times f_m$，给定 a,b，求 $c=f(a,b)$。

一般的高维前缀和、异或“卷积”等运算均可以转化至此。注意到卷积其实也可以，但并没利用维度，平凡且无用。
这里默认假定 $\mathsf{op}_{i,j,k}\in W_i$。如果不假定的话子集卷积也可以纳入。

定义

对于上述概念进行推广：

拓展高维“卷积”：高维双线性计算问题

给出若干三阶张量 $\mathcal{X}^{(i)}$，定义 $f_i(a,b): U_i \times V_i \to W_i$，其中 $f_i(a,b)_t = \sum a_j b_k \mathcal{X}^{(i)}_{j,k,t}$。
记 $f = f_1 \times f_2 \times \cdots \times f_m$，给定 a, b，求 $c = f(a,b)$。

上述问题的拓展在于，贡献不只是系数为一，并且 $a_i b_j$ 可以贡献至所有 c_k。此范围较广，可以包含所有常见的位运算卷积（子集卷积等）。
普通的双线性计算问题是一维的版本，高维能降为一维：$X^{(1)} \otimes \cdots \otimes X^{(n)}$。所以高维相当于是一个额外的条件限制。可能能借此做到更优。

构造矩阵

在本文内出现的 r，大部分就是指对应的张量秩大小。
在异或、与运算“卷积”的流程中，通常需要先对原数组进行预处理，接着对结果相乘，接着对乘法后的结果“还原”。
预处理和还原采用的是线性变换。具体来说，一般操作至第 i 维，记 x_j 为第 i 维为 j 的所有元素“压平”（直接删去第 i 维作为新的下标）后的向量，构成矩阵 $\mathbf{X} \in \mathsf{F}^{2\times 2^{n-1}}$，$\mathbf{X}_{i,j} = x_{i,j}$。最后 $\mathbf{X}' = \mathbf{AX}$ 作为新的结果。按照原顺序放回对应的位置。也可以理解为固定除了 i 以外的维度，此时有一个长为 2 的向量，对其做线性变换 $\mathbf{A}$。
实际上 $\mathbf{A}$ 可以不是方阵。考虑对数组 a 的变换 $\mathbf{A}$，b 同理。对于还原至 c 的 $\mathbf{C}$，行列数和 a, b 相反。
要确保 $\mathbf{A}, \mathbf{B}, \mathbf{C}$ 的正确性，先考虑只有一维的情况（由于各位独立所以高维正确，多元运算亦可如下分析）：

构造矩阵

不拘泥于二进制运算和方阵的限制。考虑 K 进制运算 $\mathrm{op}_{i,j}$（这里出于下标一致规定定义域和值域为 $[1,K]$）。中间变量（预处理结果和还原输入）有 r 个。

于是 $c_k=\sum_{i=1}^{r}\mathbf{C}_{k,i}(\sum_{j=1}^{K}\mathbf{A}_{i,j}a_j)(\sum_{j=1}^{K}\mathbf{B}_{i,j}b_j)$。一般情况下该算法需要满足最后的双线性变换形式一致，也就是 $a_ib_j\to c_k$ 贡献和要求一致，即：$[\mathrm{op}_{i,j}=k]=\sum_{t=1}^{r}\mathbf{C}_{k,t}\mathbf{A}_{t,i}\mathbf{B}_{t,j}$。稍作变换就是三阶张量分解的形式（令 $\mathcal{X}_{i,j,k}=[\mathrm{op}_{i,j}=k]$，并考虑 $\mathbf{C}^t$）。

对于双线性运算，直接对题目所给的张量进行分解即可。

可以证明当一维情况正确，高维也正确（且时间复杂度约为 $\tilde{O}(\max\{n,r\}^m)$，m 是维数）。如果强行展开计算的话能够证明但过于繁琐。并且此算法在 r 比较大的时候无法复用空间。下文给出一种分治做法，可以更直接的证明和利用更少的空间。

分治算法

考虑目前需要解决 $f = f_1 \times \cdots \times f_m$ 的 $f(a, b)$ 计算问题。
考虑将其分解成若干个 $f_1 \times \cdots \times f_{m-1}$ 的子问题。
记 $\tilde{a}_i$ 表示最后一维为 i 的 a 压平之后的结果，$\tilde{b}_i$ 同理。
求 $d_i = (\sum_{j=1}^{|U_i|} \mathbf{A}_{i,j}\tilde{a}_j)(\sum_{j=1}^{|V_i|} \mathbf{B}_{i,j}\tilde{b}_j)$。这里用到了子问题的乘法，用了 r 次。
求出 $\tilde{c}_j = \sum_{i=1}^{r} \mathbf{C}_{j,i}d_i$。将 $\tilde{c}_j$ 按照定义还原至原位置即可。
正确性较为显然：将 f 分解成 $g \times f_m$，此时可以展开 f_m。相对于上文的暴力展开轻松很多。
通过空间复用，空间复杂度不会超过输入输出。（考虑每层分治专用若干个数组）
一般的时间复杂度分析需要使用主定理或者分治树。一个简单的答案是 $\tilde{O}(\prod r(i))$，$r(i)$ 是第 i 层分治的 r。
于是减小 r 成为了优化该问题的关键。

分治算法

假设每一维运算相同，$n=|U_i|$，$m=|V_i|$，$k=|W_i|$。借助常见卷积的推广，$r\neq\max\{n,m,k\}$ 复杂度为 $O(\max\{r,n,m,k\}^m)$，否则 $O(r^m m)$，这只是上界。例如一个特例：按位乘法，这里可以省去求 $\sum \mathbf{A}_{i,j}\tilde{a}_j$ 的步骤，改为直接使用指针指向。就可以省去 m。
如果用输入大小表示复杂度，指数 $c=\max\{\frac{\log\operatorname{rank}(\mathcal{X})}{\log\max\{n,m,k\}},1\}$。并存在可能的 $\log$ 因子。

张量秩的基础性质

张量秩的性质较为怪异。例如在复数域和实数域上，同一个三阶张量的张量秩不一样。而二阶张量（矩阵）则一样。例如

$$\left(\begin{array}{cc|cc} 1 & & & 1 \\ & -1 & 1 & \end{array}\right) = (1,i)\circ(1,i)\circ(\frac{1}{2},\frac{1}{2})+(1,-i)\circ(1,-i)\circ(\frac{1}{2},-\frac{1}{2})$$

同时，目前张量分解（求张量秩）问题在有限域 F 是 NPC，在一些常见的无限域 $F=\mathbb{R},\mathbb{C},\mathbb{Q}$ 上是 NP hard，所以不太可能存在求出精确解的优秀做法。但其至少有一些比较基础的性质：

定义

给定 $\mathcal{X}=F^{I_1\times I_2\times\cdots\times I_N}$，满足 $I_i=n$，给定排列 $\pi\in S_N$。定义 $\mathcal{Y}$ 满足 $y_{i_1i_2\cdots i_n}=x_{i_{\pi(1)}i_{\pi(2)}\cdots i_{\pi(N)}}$，则 $\operatorname{rank}(\mathcal{Y})=\operatorname{rank}(\mathcal{X})$。

说明张量维度可以任意轮换，意味着两种输入和一种输出等价。

张量秩的基础性质

定理

$\operatorname{rank}(\mathcal{X} \oplus \mathcal{Y}) \le \operatorname{rank}(\mathcal{X}) + \operatorname{rank}(\mathcal{Y})$
$\operatorname{rank}(\mathcal{X} \circ \mathcal{Y}) \le \operatorname{rank}(\mathcal{X}) \operatorname{rank}(\mathcal{Y})$

以上三者的证明比较容易，直接任取一组分解进行排列或者组合就能证明。
其中直和的取等条件较为宽松，在许多小张量上等号成立，但存在非构造性证明说明大的复数张量存在反例。
相比直和，直积的反例容易取，考虑 $X = Y = \begin{pmatrix} 0 & 1 \\ 1 & \end{pmatrix}$，能够构造出 $r = 8$。（这里乘的越多，结果越偏离）

张量秩的基础性质

此外，还有一些简单的、容易发现的性质，比如对于张量的子张量等，秩不增；任意将某一维的 i 换成 j，j 换成 i 不影响秩。可以来解释一些事情：

例

求解 $c_i \sum_{i|j=k} a_j b_k$，注意求和条件。

考虑一维情况，依据贡献构造对应的张量 $\left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{array}\right]$，块、行、列坐标分别为 j, k, i。而交换行列坐标：$\left[\begin{array}{cc|cc} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{array}\right]$。是 OR 对应张量。一个比较好的理解是考虑 2^3 的正方体旋转。

张量秩的基础性质

根据算法流程已知：

$$\left[\begin{array}{cc|cc} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{array}\right] = (1,0)\circ(1,0)\circ(1,-1)+(1,1)\circ(1,1)\circ(0,1)$$

$$\left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{array}\right] = (1,0)\circ(1,-1)\circ(1,0)+(1,1)\circ(0,1)\circ(1,1)$$

所以有 $\mathbf{A}=\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}, \mathbf{B}=\begin{bmatrix}1 & -1\\ 0 & 1\end{bmatrix}, \mathbf{C}=\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}^t=\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}$。
可以理解为：$\mathrm{AND}(\mathrm{OR}(a)\cdot\mathrm{IAND}(b))=c$。
通过这个例子可以发现，如果直接构造，有时候过程抽象、不太直接（但好处是可以直接通过算法写出式子）。这是因为构造是基于三维的，符号表示和理解都略微麻烦，后面将进行转化，给出矩阵角度的理解和构造。

最大秩

对于形状相同的张量 $R^{I\times J\times K}$，其秩有一个上界，较为平凡且普通的上界是 $\min\{IJ, JK, IK\}$，没有找到更好的通用结果。
但对于一些小的实数张量和复数张量，有已发现的上确界（最大秩），例如根据一些研究结果，$3\times 3\times 3$ 的复数或实数张量最大秩为 5。（注意不是所有域，例如 $\mathbb{Z}_2$ 存在为 6 的张量）。

典型秩

除此之外，针对随机的张量也有秩的分析：

定义 (典型秩（Typical Rank）)

若一个非零测度的张量集合上秩均为 x，则 x 是典型秩。

可以简单理解为出现概率非 0。

定义 (通用秩（Generic Rank）)

若几乎所有张量的秩为 x，则称其为通用秩。

1 给出了一些特殊三阶张量的典型秩。2 给出了固定大小的张量的典型秩。其它资料（见集训队论文）也有说明。
实际运用如果对于近乎完全随机的张量，一般可以认为其秩属于典型秩，可以认为不能突破至低于典型秩的位置。
相对优秀的：能够从小推广，例如 K 进制与、或、异或等。

表格

Tensor Size	Typical Rank	Citation
$2 \times 2 \times 2$	$\{2,3\}$	[120]
$3 \times 3 \times 2$	$\{3,4\}$	[119, 173]
$5 \times 3 \times 3$	$\{5,6\}$	[175]
$I \times J \times 2$ with $I \geq 2J$ (very tall)	$2J$	[177]
$I \times J \times 2$ with $J < I < 2J$ (tall)	I	[177]
$I \times I \times 2$ (compact)	$\{I, I+1\}$	[173, 177]
$I \times J \times K$ with $I \geq JK$ (very tall)	JK	[174]
$I \times J \times K$ with $JK - J < I < JK$ (tall)	I	[174]
$I \times J \times K$ with $I = JK - J$ (compact)	$\{I, I+1\}$	[174]

Table 3. Typical rank over $\mathbb{R}$ for three-way tensors.

图: 1. 实数张量的典型秩

表格

Table 1
Typical ranks for 2-slice, 3-slice, and 4-slice unconstrained arrays. Values reported in bold correspond to smallest typical ranks computed numerically; values in plain font were known before. Values separated by commas are known typical ranks. In the complex field, the smallest value in a cell is generic.

	N_3	2				3			4	
N_1	N_2	2	3	4	5	3	4	5	4	5
2		2,3	3	4	4	3,4	4	5	4,5	5
3		3	3,4	4	5	5	5	5,6	**6**	**6**
4		4	4	4,5	5	5	**6**	**6**	**7**	**8**
5		4	5	5	5,6	5,6	**6**	**8**	**8**	**9**
6		4	6	6	6	6	**7**	**8**	**8**	**10**
7		4	6	7	7	7	**7**	**9**	**9**	**10**
8		4	6	8	8	8	8,9	**9**	**10**	**11**
9		4	6	8	9	9	9	**9**	**10**	**12**
10		4	6	8	10	9	10	10	**10**	**12**
11		4	6	8	10	9	11	11	**11**	**13**
12		4	6	8	10	9	12	12	12,13	**13**

图: 2. 实数张量的典型秩，若在复数域内，最小值为通用秩

有限域下的暴力做法

先考虑三阶张量的分解。
最直接的暴力做法是考虑枚举 $\mathbf{A}, \mathbf{B}, \mathbf{C}$ 然后判断是否合法。
注意到最后的形式是 $\mathcal{X}_{i,j,k} = \sum \mathbf{A}_{t,i}\mathbf{B}_{t,j}\mathbf{C}_{t,k}$，可以只枚举 $\mathbf{A}, \mathbf{B}$ 后高斯消元。
由此也能拓展出一种更适合人的构造方法：构造若干秩一矩阵，考虑能否线性组合成 $\mathcal{X}^{(i)}$（或是固定其他维度）。
对于能够表示为运算表形式的张量，一般考虑能否构造出 $\mathbf{D}_{i,j} = [\mathrm{op}_{i,j} = k]$，这里记为 $\mathbf{X}_k$。
只采用此方法，并加上一些启发式的结果可以做出一些较好的发现。

例子

例 (Tritwise mex)

记 $\mathsf{op}_{i,j} = \mathrm{mex}\{i, j\}$，求解上述高维“卷积”问题。维数 $n \le 12$。

例子

对于 Tritwise mex，从 $\mathrm{op}_{i,j} = \mathrm{mex}\{i,j\}$ 导出的张量，其表示成运算表形式为：

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

使用一个秩一矩阵表示 $\mathbf{X}_0$。接着可以使用四个角减去右下角，用两个秩一矩阵表示 $\mathbf{X}_1$。最后添加一个全局矩阵就可以容斥表示出 $\mathbf{X}_2$。
或者可以用两个秩一矩阵（单点）表示 2，最后容斥。
均使用 4 个秩一矩阵，这里 $\mathbf{A}, \mathbf{B}$ 确定了秩一矩阵，$\mathbf{C}$ 确定了这些秩一矩阵对 $\mathbf{X}_i$ 的转移贡献。

例子

对于通过 op$_{i,j}$ 做出来的张量（能够被运算表形式表示），可以考虑如下的一些方法：
优先找出一些子矩形使得包含元素一致；对这些矩形合并；使用整体减去局部的容斥。
一种非上述构造为异或运算，可以刻画黑白棋盘形状。更高进制可以刻画出类似的形状。但一般二进制异或运算的构造更常见。

例子

例 (欧伊昔)

给定随机 3×3 的运算表 op，记每一维 $f_i(i,j)=\mathrm{op}_{i,j}$，给定 a,b 满足 $a_i,b_i\in[0,9]$。
需要解决高维卷积问题 $f=f_1\times\cdots\times f_n$，$n\le 11$。时限 3 秒。

这题需要考虑任意能够表示成运算表形式并无空元素的张量。
初看没有规律，只能使用容斥的方式从 $r=9$ 变为 $r=7$。
如果 $0,1,2$ 不是均出现 3 次，或者 op 有相同元素在同一行或者同一列可以优化至 $r=6$。否则是拉丁方可以有 $r=3$ 的复数构造（或者 $r=4$ 的有理数构造）。
此方法稍作优化（减去无用计算）足以通过。但不够优秀。
接下来对此题加强：运算表非随机，$n\le 12$。

例子

一个发现是考虑所有拉丁方于题给张量的不同点个数，可以推导或代码证明出最小值至多为 2，可以做到 $r=5$。
另一个发现是，对于二进制运算表形式（无空元素）的张量，均可以构造出 $\mathbf{A}_{i,j}, \mathbf{B}_{i,j} \in \{-1,0,1\}$ 的结果，枚举所有此类秩一矩阵进行高斯消元，使用程序穷举证明可以获得 $r \le 4$ 的结论。并且 mex 基本无法找到 $r=3$ 的构造，可以认为这已经是“最优秀”的结果。
笔者实现了一份代码，暴力找出所有秩一矩阵后穷举后高斯消元，未作优化构造过程就能达到 300ms 以内（如果存在构造的话），即使最后使用实数表示矩阵也能在本地跑进 3s，在 OJ 上会更快。一些优化方向是可能由于取模不使用实数表示，或者最后整体除分母规避小数。（一份不加任何优化的实现：QOJ 提交 707232）。

例子

另一种方式是使用接下来的近似算法，由于此题的答案较小可以快速达到精度。
因为近似所需的时间开销更少，所以比上述提交更快（原题数据），但可能不稳定。
并且曾经有一组运算表，近似构造的矩阵出现 10^{-100} 一类的数，导致之后的运算奇慢。但解决方法是容易的（保留一定小数位）。

交替最小二乘法（ALS）

一些近似算法可以在实际中用来更快给出一组好的解。通常这类算法需要钦定分解个数 r。
将原先的精确结果转为近似结果，可以设计一个指标，例如结果张量和原张量的差的范数。
如果不是枚举 $\mathbf{A}, \mathbf{B}$ 而是直接任意生成一组，那么很容易出现无解的情况。
但可以求出近似解 $\mathbf{C}$，使得指标尽可能小。
由于这里需要最小化范数，使用最小二乘法相关的结论可以获得一个解。
接下来不可能优化 $\mathbf{C}$ 了，可以改为优化 $\mathbf{A}, \mathbf{B}$，轮换优化。
该算法的好处是指标单调递减，但不一定趋于 0（即使存在解），但分解小张量的实际效果较好。

梯度下降法

此算法的核心思想是给出一个最优的“方向”，不断往“方向”移动。实现也较为简单。
需要求一些导数来直接计算结果，这对指标有一些限制（但比 ALS 宽松）。
具体来说，对于一个向量 v 和最优化函数 $f(v)$，每次令 $\nabla(x)_i = \frac{\partial f}{\partial x}$，然后 $v_{k+1} = v_k - \alpha_k \nabla(v_k)$。
在张量分解中，对最后的范数平方直接关于 $A_{i,k}$ 求导，就可以获得 $A_{i,k}$ 对应的偏导数，其余同理。
在要求较严格的情况下，α 要求比较严格，否则会出现震荡。但单次操作复杂度能低于 ALS。并非常容易实现。

一个启发

我们可以发现子集卷积难以构造出 $r=2$ 的对应结果。但确实存在 $\tilde{O}(2^n)$ 的算法。

使用 CP 分解，可以发现 $\begin{pmatrix} 0 & 1 \\ 1 & \end{pmatrix}$ 的时候只能达到 $r=3$，而其自身直积的结果不但能够做到 $r=8$，并且可以在 $r=4$ 的时候不断下降（尽管速度很慢），有收敛的趋势。

观察其结果可以重新整理至 $\begin{pmatrix} 0 & 1 \\ 1 & \end{pmatrix}$ 的一种近似解：

$$(1,\epsilon)\circ(1,\epsilon)\circ(1,\frac{1}{\epsilon})+(1,0)\circ(1,0)\circ(0,\epsilon)$$

不计运算开销确实可以取合适的 ϵ 做到 $O(2^n n)$，但这里的精度变为原来的 n 倍，结果为 $O(2^n n^2)$ 和之前一致。

边界秩

对于这一类张量，允许小量的存在会使其的张量分解变得更小。因此可以定义“边界秩”。一种定义是存在秩均为 R 的序列，极限为该张量。但不好直接推导和构造。这里给出另一种定义：

定义

记 $\operatorname{rank}_h(\mathcal{X}) = \{R | \exists \mathbf{A}, \mathbf{B}, \mathbf{C}, \sum_{i=1}^{R} \mathbf{A}^{(i)}\mathbf{B}^{(i)}\mathbf{C}^{(i)} = x^h\mathcal{X} + o(x^h)\}$，这里 $\mathbf{A}, \mathbf{B}, \mathbf{C}$ 的元素是 h 次多项式。

引理

$\operatorname{rank}_h(\mathcal{X})$ 单调不增，因此在 $h \to +\infty$ 收敛。

定义 (边界秩)

记 $\underline{\operatorname{rank}}(\mathcal{X}) = \lim_{h\to+\infty} \operatorname{rank}_h(\mathcal{X}) = \operatorname{rank}_w(\mathcal{X})$，$w$ 依据 $\mathcal{X}$ 定。

性质

虽然边界秩在二阶张量（矩阵）下和张量秩是一致的，但在高阶张量下不一致（例如上面的子集卷积）。
容易发现其有一些张量秩的性质：

定理

定义 $\mathcal{Y}$ 满足 $y_{i_1 i_2 \cdots i_n} = x_{i_{\pi(1)} i_{\pi(2)} \cdots i_{\pi(N)}}$，则 $\mathrm{rank}_h(\mathcal{Y}) = \mathrm{rank}_h(\mathcal{X})$。

定理

$\mathrm{rank}_h(\mathcal{X} \oplus \mathcal{Y}) \le \mathrm{rank}_h(\mathcal{X}) + \mathrm{rank}_h(\mathcal{Y})$

相比张量秩，边界秩的此条件容易简单粗暴的构造出一组不满足等式的例子。具体考虑使用一个 $r = IJ$ 的张量构造出一些边角料组合出另一个张量。

性质和运用

定理

$\operatorname{rank}_{h+l}(\mathcal{X} \circ \mathcal{Y}) \le \operatorname{rank}_h(\mathcal{X}) \operatorname{rank}_l(\mathcal{Y})$

同时边界秩可以转化至普通张量秩：

定理

$\operatorname{rank}(\mathcal{X}) \le \binom{h+2}{2} \operatorname{rank}_h(\mathcal{X})$。

可以发现：$\operatorname{rank}(\mathcal{X}^k) \le \binom{hk+2}{2} \operatorname{rank}_{hk}(\mathcal{X}^k) \le \binom{hk+2}{2} \operatorname{rank}_h(\mathcal{X})^k$，所以可以做到 $\tilde{O}(n^{w+\epsilon})$，其中 $w = \frac{\log \operatorname{rank}_h(\mathcal{X})}{\log \max\{n,m,k\}}$。能“当作”普通张量秩使用。

运用

但注意到这里要求使用 $\mathcal{X}^k$ 作为分治对应的张量，隐含常数过于庞大，因此不实用。
可以像子集卷积那样，假设维数为 k，只需要维护 hk 次多项式即可。复杂度变为 $\tilde{O}((\prod r(i))(\sum h(i)))$，相比正常的张量秩只多了一个 $\sum h(i)$，渐近和维数同阶（所以也能吸收至 $\tilde{O}$，这里是为了突出变化）。对于普通的分治做法直接维护多项式即可。
对于复杂度分析，中间需要中间量次卷积，其余操作只需要加法和常数乘。考虑每一维大小为 d，若 $r \geq d$ 结果为 $O(r^k k^2)$。

运用

对于常用的子集卷积算法，此时维护的多项式一般叫做“占位多项式”，也可以先对于占位多项式的 x^i 系数集体做变换，然后卷积，然后逆变换还原。

这也意味着如果 $\mathbf{A}=\mathbf{B},\mathbf{AC}=\mathbf{I}$，且符合一些多项式技巧的先决条件，就可以套用。（对于普通秩亦是如此，只是变成了一个变量），例如使用小多项式快速幂完成快速幂操作。

推广

对于一些问题，可以构造出对应的分治乘法（分治过程保持不变，但输入输出和高维“卷积”不同）。此时的复杂度也只能具体分析，因为不一定能够使用高维“卷积”的结论。

例 (卷积)

求 $c_i = \sum_{j+k=i} a_j b_k$。

一种分治乘法是 Karatsuba 算法，分成三份长度减半，递归。可以考虑分成 m 段。则运算表为 $\mathrm{op}_{i,j} = i + j$。分解能通过 $2m-1$ 的循环卷积的张量得到，由于 $\frac{\log(2m-1)}{\log m} \to 1$，根据主定理，能够做到 $O(n^{1+o(1)})$。相当于暴力 DFT 卷积配合分治。

推广

另一种问题是矩阵乘法，不妨只考虑 $n \times n \times n$ 的乘法类型。这是因为对于任意 $N \times M \times K$ 对应的张量（大小 $NM \times MK \times NK$）可以做 $(NMK) \times (NMK) \times (NMK)$。
这里边界秩可以直接转张量秩套用。由于不同张量的直和的边界秩容易构造出不等关系，可以基于此构造更优秀的张量分解：

定理 (Schönhage's τ theorem)

若有 $\mathcal{X} = \underline{\text{rank}}(\oplus\langle n_i, m_i, k_i\rangle) \le r$，且 $\sum_{i=1}^{p}(n_i m_i k_i)^\tau = r$，则矩阵乘法最优指数 $\omega \le 3\tau$。

同边界秩转张量秩，自乘后取一决定项并对此分析可以得出。
这也对高维“卷积”能有一些启发，比如可以通过适当的分治方法利用上边界秩这种特性，但笔者尚未发现。

例题

例 (Jellyfish 改编)

给出若干关键点，求出 n 维 d 进制下距离为 i 的点对数。
$d \in \{2,3,4\}$，$d^n \le 500000$。

可以求每一维距离压位为 S_i 的点对数。这样 $d=4$ 时 $r=6$，$d=3$ 时 $r=4$。$d=2$ 时是二进制异或。
这题不是直接的高维卷积问题，考虑每两维的距离压在一起，由于是两维的距离直接相加，可能能做到比原先优秀，在迭代很多次后精度仍然保持收敛趋势。同时这题值域只有 d^{2n}，要求较低。但根据推断应该无显著优化。
$d=3$ 时求解迅速，压两维能够近似出 $r<16$；$d=4$ 时，压两维即使是 $r=36$ 也要跑非常久才能出满足精度的解，下降速率非常慢。

Thanks for listening !