第 53 卷 增刊Ⅰ
2023 年 7 月

中 国 海 洋 大 学 学 报
PERIODICAL OF OCEAN UNIVERSITY OF CHINA

53(Sup.Ⅰ):190～198
July, 2023

张量 TTr1SVD 的随机算法*

丁明慧，解朋朋**

（中国海洋大学数学科学学院，山东 青岛 266100）

摘　要： 张量序列秩-1 奇异值分解（TTr1SVD）自然地将奇异值分解（SVD）推广到张量层面，将任意实张量分解为标准正交秩-1 外积的有限和。基于其具有已知数量上限的正交秩-1 外积项和易于截断误差量化的良好性质，本文首先给出了一种有利于张量分解和还原的表达形式，并提出了保持分解形式的截断 TTr1SVD 算法，在固定精度的同时大大降低了计算成本。受低秩矩阵逼近的随机算法的启发，本文还开发了针对固定精度问题的高效随机 TTr1SVD 算法。最后，给出的数值例子展现了所提算法在数据逼近和压缩方面的应用前景。

关键词： TTr1SVD；奇异值分解；截断；固定精度问题；随机算法

中图法分类号： O241.5　　**文献标志码：** A　　**文章编号：** 1672-5174(2023)07Ⅱ-190-09

DOI： 10.16441/j.cnki.hdxb.20220382

引用格式： 丁明慧，解朋朋. 张量 TTr1SVD 的随机算法[J]. 中国海洋大学学报(自然科学版)，2023，53(增Ⅰ)：190-198.

Ding Minghui，Xie Pengpeng. Randomized algorithm for tensor TTr1SVD[J]. Periodical of Ocean University of China，2023，53(Sup.Ⅰ)：190-198.

张量作为一种数据组织形式，已应用于包括心理测量学[1]、图像/视频和信号处理[2]和机器学习[3]在内不同的领域中。张量数据的压缩、排序、分析和许多其他处理都依赖于张量分解。

国内外很多学者已经研究了不同张量积下的张量分解，如 CANDECOMP/PARAFAC（CP）分解[1,4]、高阶奇异值分解（HOSVD）[4-5]、基于 t-product 的张量奇异值分解（T-SVD）[6]、张量序列（TT）[7]和张量序列秩-1 奇异值分解（TTr1SVD）[8]，并将线性代数方法扩展到多线性环境。其中，CP 分解将一个张量表示为有限个外积之和，因此张量的 CP 秩可以定义为分解中的最小项数。目前没有可行的算法来确定 CP 秩。HOSVD 可以有效地将一个张量压缩成一个核张量和几个列正交因子矩阵，也可以表示为有限个外积之和，其核张量中的每个元素都可以看作是一个外积的权重。在这种定义下，尽管所有的外积都是正交的，但核张量是稠密的，并不能很好的规范张量的秩。TTr1SVD 将高阶实张量分解为有限个实正交外积的和。与现有的张量分解相比，TTr1 分解具有许多有利的特性，如唯一性、易于量化近似误差，以及易于转换为具有稀疏核心张量的 Tucker 格式。为了进一步研究 TTr1SVD 在近似和数据压缩方面的作用，我们研究了在张量分解与还原过程更高效的计算方法，称该方法为具有固定精度的截断 TTr1SVD 算法。

近年来，随机矩阵方法已被用于有效地、准确地计算近似的低秩矩阵分解[9-11]。这些算法易于实现，并已推广到基于不同张量积的张量奇异值分解中[11-14]。TTr1SVD 需要计算张量重塑为矩阵的 SVD 和逐步计算产生的奇异向量重塑为矩阵后的 SVD，以产生已知项数上限的有限外积，对于大型张量这一过程无疑是耗时的。针对此问题，我们致力于研究一种与 TTr1SVD 相对应的新型随机化策略，希望该技术能够在保持精度的同时大大降低计算成本。

本文提出了具有固定精度的截断 TTr1SVD 算法和具有固定精度的随机 TTr1SVD 算法，并从张量的分解与还原过程中分析了 r 项逼近算法和提出的两个算法的计算成本，最后给出了数值算例，验证了算法在数据近似和压缩方面的性能。

1　基础知识

在本节，我们介绍本文中使用的定义和符号。用小写字母 $x,y,z\cdots$ 表示标量；用黑体小写字母 $\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}\cdots$ 表示向量，x_i 为向量 $\boldsymbol{x}$ 的第 i 个元素；用黑体大写字母 $\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}\cdots$ 表示矩阵，$\boldsymbol{x}_i$ 表示矩阵 $\boldsymbol{X}$ 的第 i 个列向

* 基金项目：国家自然科学基金项目(11801534)资助
Supported by the National Natural Science Foundation of China(11801534)
收稿日期：2022-09-01；修订日期：2022-10-18
作者简介：丁明慧(1998—　)，女，硕士生，研究方向为数值代数。E-mail：dingminghui@stu.ouc.edu.cn
** 通信作者：E-mail：xie@ouc.edu.cn

量；用 $\bar{\boldsymbol{X}},\bar{\boldsymbol{Y}},\bar{\boldsymbol{Z}}\cdots$ 表示张量，x_{ijk} 是张量 $\bar{\boldsymbol{X}}$ 的第(i,j,k)个元素。运算符 vec(·)，nnz(·)，diag(·)分别表示将矩阵或张量拉直成一个列向量，将向量、矩阵或张量的非零元素取出并按列向量排列，将向量元素作为对角元素变形为对角矩阵。

张量的纤维（Fiber）指除了某一维度外，固定其他所有维度的下标后抽取得到的向量。张量的切片(Slice)指除了某两个维度外，固定其他剩余维度后抽取出的二维子数组。对于三阶张量 $\bar{\boldsymbol{X}}$ 而言，它有三个模态的纤维，分别记为 $\bar{\boldsymbol{X}}_{i,:,k}$，$\bar{\boldsymbol{X}}_{:,jk}$，$\bar{\boldsymbol{X}}_{ij,:}$；有水平切片、侧面切片和正面切片三种切片，分别表示为 $\bar{\boldsymbol{X}}_{i,:,:}$，$\bar{\boldsymbol{X}}_{:,j,:}$，$\bar{\boldsymbol{X}}_{:,:k}$。为方便起见，我们有时使用 MATLAB 符号来表示矩阵或张量的子部分。例如，$\bar{\boldsymbol{X}}_{i,:,:}=\bar{\boldsymbol{X}}(i,:,:)$表示张量的第 i 个水平切片，$\boldsymbol{X}(i,:)$表示矩阵的第 i 行。

定义 1（张量矩阵化）[4] 张量 $\bar{\boldsymbol{X}}\in\mathbf{R}^{I_1\times I_2\times\cdots\times I_N}$ 的 n 模态矩阵化或称为 n 模态展开，是一种重新组织张量元素的操作：把张量的第 n 模态的纤维作为列向量按照顺序排列形成一个新矩阵，并记为 $\boldsymbol{X}_{(n)}$。

定义 2（内积）[4,8] 两个张量 $\bar{\boldsymbol{X}},\bar{\boldsymbol{Y}}\in\mathbf{R}^{I_1\times I_2\times\cdots\times I_N}$ 的内积定义为

$$\langle\bar{\boldsymbol{X}},\bar{\boldsymbol{Y}}\rangle=\sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2}\cdots\sum_{i_N=1}^{I_N}x_{i_1i_2\cdots i_N}y_{i_1i_2\cdots i_N}。$$

如果两个张量的内积为 0，我们称它们为正交的。张量 $\bar{\boldsymbol{X}}$ 的 Frobenius 范数为 $\|\bar{\boldsymbol{X}}\|_{\mathrm{F}}=\langle\bar{\boldsymbol{X}},\bar{\boldsymbol{X}}\rangle^{\frac{1}{2}}$，简称为 F 范数。

定义 3（秩-1 张量）[4] 一个张量 $\bar{\boldsymbol{X}}\in\mathbf{R}^{I_1\times I_2\times\cdots\times I_N}$ 被称为秩-1 张量，指的是它可以表示为 N 个向量的外积：

$$\bar{\boldsymbol{X}}=\boldsymbol{a}^{(1)}\circ\boldsymbol{a}^{(2)}\cdots\circ\boldsymbol{a}^{(N)},$$

其中每个元素为 $\bar{\boldsymbol{X}}_{i_1i_2\cdots i_N}=\boldsymbol{a}^{(1)}_{i_1}\boldsymbol{a}^{(2)}_{i_2}\cdots\boldsymbol{a}^{(N)}_{i_N}$。

2　张量序列秩-1 奇异值分解

TTr1SVD 是一个类似于 CP 分解的秩-1 张量分解算法，但与 CP 分解寻求最小求和项来获得张量的 CP 秩不同，TTr1SVD 是基于矩阵奇异值分解可以迅速获得秩-1 张量数量上限的确定性计算过程。具体来说，TTr1SVD 算法连续计算从张量变形而成矩阵的 SVD。整个过程形成了一个树状结构，不同的分解顺序可以得到不同数量的秩-1 项。

以三阶张量为例，设 $\bar{\boldsymbol{X}}\in\mathbf{R}^{n_1\times n_2\times n_3}$，张量的阶数 $d=3$，它等于树的层数。假设分解顺序 $\rho=[1,2,3]$。在第一阶段，我们在第 0 层节点上使用运算符 reshape 将张量重塑为 $n_1\times n_2n_3$ 的矩阵 $\boldsymbol{X}_1$：

$$\boldsymbol{X}_1=\mathrm{reshape}(\bar{\boldsymbol{X}},[n_1,n_2n_3]),\tag{1}$$

这一步等价于将张量按照第一模态展开得到矩阵 $\boldsymbol{X}_{(1)}$，然后计算其经济型 SVD：

$$\boldsymbol{X}_1=\boldsymbol{U}_1\boldsymbol{S}_1\boldsymbol{V}_1^{\mathrm{T}}。\tag{2}$$

将第一次 SVD 得到的奇异值（即矩阵 $\boldsymbol{S}_1$ 的对角线元素）记为 $\sigma_1,\cdots,\sigma_{r_1}$，其中 $r_1=\min(n_1,n_2n_3)$，得到的奇异值数等于第 1 层的节点数。在第二阶段，将 $\boldsymbol{V}_1$ 的每个列向量 v_i 再次重塑为一个矩阵 $\tilde{\boldsymbol{V}}_i\in\mathbf{R}^{n_2\times n_3}$，

$$\tilde{\boldsymbol{V}}_i=\mathrm{reshape}(v_i,[n_2,n_3]),\tag{3}$$

并计算 $\tilde{\boldsymbol{V}}_i$ 的 SVD，如

$$\tilde{\boldsymbol{V}}_i=\boldsymbol{U}_{1i}\boldsymbol{S}_{1i}\boldsymbol{V}_{1i}^{\mathrm{T}},\tag{4}$$

重复这个操作，直到计算出第 2 层所有叶节点对应的奇异值，即 $\sigma_{11},\cdots,\sigma_{ij},\cdots,\sigma_{r_1r_2}$，其中 $r_2=\min(n_2,n_3)$，$i=1,\cdots r_1$，$j=1,\cdots r_2$。如图 1 所示。

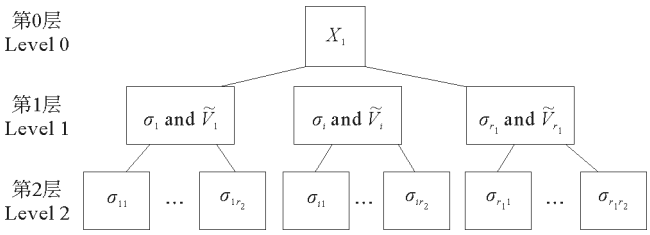


图 1　TTr1SVD 算法的树状表示

Fig. 1　Tree representation of TTr1SVD algorithm

任意矩阵进行 SVD 后可改写为秩-1 项的和的形式，$\bar{\boldsymbol{X}}$ 可表示为

$$\bar{\boldsymbol{X}}=\sum_{i=1}^{r_1}\sum_{j=1}^{r_2}\sigma_i\sigma_{ij}\circ\boldsymbol{u}_i\circ\boldsymbol{u}_{ij}\circ\boldsymbol{v}_{ij},\tag{5}$$

式中 $\boldsymbol{u}_i$、$\boldsymbol{u}_{ij}$、$\boldsymbol{v}_{ij}$ 分别是 $\boldsymbol{U}_1$、$\boldsymbol{U}_{1i}$、$\boldsymbol{V}_{1i}$ 的列向量。下面给出关于三阶张量 TTr1SVD 的定义

定义 4[9]　设 $\bar{\boldsymbol{X}}\in\mathbf{R}^{n_1\times n_2\times n_3}$，分解顺序为 $\rho=[1,2,3]$。则 $\bar{X}$ 的 TTr1SVD 是

$$\bar{\boldsymbol{X}}=\sum_{r=1}^{N}\tilde{\sigma}_r\boldsymbol{x}_r^{(1)}\circ\boldsymbol{x}_r^{(2)}\circ\boldsymbol{x}_r^{(3)},\tag{6}$$

其中 $\tilde{\sigma}_r$ 是分解结果中秩-1 张量的权重，因此可记为 $\bar{\boldsymbol{X}}$ 的奇异值；$\boldsymbol{x}_r^{(1)}\in\mathbf{R}^{n_1}$，$\boldsymbol{x}_r^{(2)}\in\mathbf{R}^{n_2}$，$x_r^{(3)}\in\mathbf{R}^{n_3}$ 是分解产生的列向量；N 是产生的秩-1 张量的数量，与张量的各个维度有关，具体表现为

$$N=\begin{cases}n_1\times\min(n_2,n_3), & n_1<n_2n_3\\ n_2n_3\times\min(n_2,n_3), & n_1\geqslant n_2n_3\end{cases}。$$

结合公式(5)、(6)和图 1 可知，$\tilde{\sigma}_r$ 是沿每个分支所有叶子结点奇异值的乘积，即 $\tilde{\sigma}_r=\sigma_i\sigma_{ij}$，$r=ij$；$\boldsymbol{x}_r^{(1)}$ 是第 0 层父节点进行 SVD 产生的左奇异向量 $\boldsymbol{u}_i$；$\boldsymbol{x}_r^{(2)}$ 和 $\boldsymbol{x}_r^{(3)}$ 分别是是第 1 层叶子节点上进行 SVD 产生的左奇异向量 $\boldsymbol{u}_{ij}$ 和右奇异向量 $\boldsymbol{v}_{ij}$。树的第 k 层的节点数量

$$r_k=\min\Big(n_k,\prod_{i=k+1}^{3}n_i\Big)\ (k=1,2),$$

分解得到的项的数目是所有的叶子节点的数目 $N=$

$\prod_{k=1}^{2} r_k$。

推论 1 设 $\bar{\boldsymbol{X}}\in\mathbf{R}^{n_1\times n_2\times n_3}$，(2)式是 $\bar{\boldsymbol{X}}$ 的 TTr1SVD，则

$$\|\bar{\boldsymbol{X}}\|_{\mathrm{F}}^2=\tilde{\sigma}_1^2+\cdots+\tilde{\sigma}_N^2=\sum_{i=1}^{r_1}\sum_{j=1}^{r_2}(\sigma_i\sigma_{ij})^2=\sum_{i=1}^{r_1}\sigma_i{}^2。$$

证明 由(6)式可得

$$\|\bar{\boldsymbol{X}}\|_{\mathrm{F}}^2=\Big\|\sum_{r=1}^{N}\tilde{\sigma}_r\boldsymbol{x}_r^{(1)}\circ\boldsymbol{x}_r^{(2)}\circ\boldsymbol{x}_r^{(3)}\Big\|_{\mathrm{F}}^2=\sum_{r=1}^{N}\tilde{\sigma}_r^2\|\boldsymbol{x}_r^{(1)}\circ\boldsymbol{x}_r^{(2)}\circ\boldsymbol{x}_r^{(3)}\|_{\mathrm{F}}^2。$$

由于分解产生的每个外积的 F 范数都为 1，所以第一个等号成立；由(5)式可知，$\tilde{\sigma}_r=\sigma_i\sigma_{ij}$，所以第二个等号成立，由(3)和(4)式可知 $\sigma_{ij}(j=1,\cdots,r_2)$ 是矩阵 $\tilde{\boldsymbol{V}}_i$ 的奇异值，则 $\sum_{j=1}^{r_2}\sigma_{ij}^2=\|\tilde{\boldsymbol{V}}_i\|_{\mathrm{F}}^2=\|v_i\|_{\mathrm{F}}^2$，因为单位向量 $\boldsymbol{v}_i$ 的 F 范数为 1，所以推论中第三个等号成立。证明毕。

2.1 矩阵表示

记 $R_{\mathrm{tot}}=\sum_{i=1}^{R}L_i$，$L_i=\mathrm{rank}(\tilde{\boldsymbol{V}}_i)$，$R=\mathrm{rank}(\boldsymbol{X}_1)$。

根据(2)和(4)式的结果定义

$$\boldsymbol{X}_i^{(2)}=[\boldsymbol{u}_{i1},\cdots,\boldsymbol{u}_{iL_i}]\in\mathbf{R}^{n_2\times L_i},$$

$$\boldsymbol{X}^{(2)}=[\boldsymbol{X}_1^{(2)},\cdots,\boldsymbol{X}_R^{(2)}]\in\mathbf{R}^{n_2\times R_{\mathrm{tot}}},$$

$$\boldsymbol{X}_i^{(3)}=[\boldsymbol{v}_{i1},\cdots,\boldsymbol{v}_{iL_i}]\in\mathbf{R}^{n_3\times L_i},$$

$$\boldsymbol{X}^{(3)}=[\boldsymbol{X}_1^{(3)},\cdots,\boldsymbol{X}_R^{(3)}]\in\mathbf{R}^{n_3\times R_{\mathrm{tot}}},$$

$$\boldsymbol{x}_r^{(1)}=\boldsymbol{u}_r\in\mathbf{R}^{n_1},$$

并用一个 $r_2\times r_1$ 的矩阵 $\boldsymbol{S}$ 的列 $\boldsymbol{s}_i$ 存放 r_2 个 $\tilde{\sigma}_r$，满足 $s_{ji}=\sigma_i\sigma_{ij}$ 且 $\sigma_i\sigma_{i1}\geqslant\sigma_i\sigma_{i2}\geqslant\cdots\geqslant\sigma_i\sigma_{ir_2}$，则

$$\boldsymbol{X}_i^{(1)}=[\mathrm{nnz}(\boldsymbol{s}_i)^{\mathrm{T}}\otimes\boldsymbol{u}_i]\in\mathbf{R}^{n_1\times L_i},$$

$$\boldsymbol{X}^{(1)}=[\boldsymbol{X}_1^{(1)},\cdots,\boldsymbol{X}_R^{(1)}]\in\mathbf{R}^{n_1\times R_{\mathrm{tot}}}。$$

由以下的矩阵表示[3]

$$\boldsymbol{X}_1{}^{\mathrm{T}}=[\mathrm{vec}(\bar{\boldsymbol{X}}_{1::}),\cdots,\mathrm{vec}(\bar{\boldsymbol{X}}_{n_1::})]=(\boldsymbol{X}^{(3)}\odot\boldsymbol{X}^{(2)})\boldsymbol{X}^{(1)\mathrm{T}}\in\mathbf{R}^{n_2n_3\times n_1},$$

其中“⊙”表示矩阵的 Khatri-Rao 积[4]。由此，我们还可以利用这些矩阵单独地表示出张量的每个切片，如

$$\chi_{i::}=\boldsymbol{X}^{(2)}\mathrm{diag}(\boldsymbol{X}_{i:}{}^{(1)})\boldsymbol{X}^{(3)\mathrm{T}}。$$

进一步分析，还可以得出类似于$(L_r,L_r,1)$分解的表达形式[15]。

推论 2 设 $\bar{\boldsymbol{X}}\in\mathbf{R}^{n_1\times n_2\times n_3}$，分解顺序为 $\rho=[1,2,3]$。则 $\bar{\boldsymbol{X}}$ 的 TTr1SVD 是

$$\bar{\boldsymbol{X}}=\mathrm{permute}(\widetilde{\boldsymbol{X}},[3,1,2])\in\mathbf{R}^{n_1\times n_2\times n_3},$$

$$\widetilde{\boldsymbol{X}}=\sum_{i=1}^{R}H_i\circ\boldsymbol{x}_i^{(1)}\in\mathbf{R}^{n_2\times n_3\times n_1},$$

式中 $\boldsymbol{H}_i=\boldsymbol{X}_i^{(2)}\mathrm{diag}(\mathrm{nnz}(s_i))\boldsymbol{X}_i^{(3)\mathrm{T}}$，$L_i=\mathrm{rank}(\boldsymbol{H}_i)$，$\boldsymbol{X}_i^{(2)},\boldsymbol{X}_i^{(3)}$ 是列正交矩阵，$\boldsymbol{x}_i^{(1)}$ 是单位向量且与 $\boldsymbol{x}_j^{(1)}(i\neq j)$ 正交。$\mathrm{nnz}(s_i)$的使用保留了分解产生的非零奇异值，节省了存储奇异值所需的时间。

注 1 运算符 $\mathrm{permute}(\bar{\boldsymbol{X}},order)$定义为重新排列张量 $\bar{\boldsymbol{X}}$ 的维数，是它们按照向量 $order$ 的指定顺序排列。重排产生原因是外积元素相乘的顺序为 $\rho'=[2,3,1]$。若分解顺序为 $\rho=[3,1,2]$，则还原张量时无需重排。

2.2 固定精度的截断 TTr1SVD

Eckart-Young 定理[16]表明一个矩阵的最优秩-k近似可以用秩-k 截断的 SVD 来构造。类似地，一个张量的 r 项近似可以通过截断(6)式的前 r 项来计算。

引理 1(r 项逼近)[11] 令 $\bar{\boldsymbol{X}}\in\mathbf{R}^{n_1\times n_2\times n_3}$，(6)式是 $\bar{\boldsymbol{X}}$ 的 TTr1SVD 分解，设 $\bar{\boldsymbol{X}}_r$ 是 TTr1SVD 分解的 r 项逼近，则

$$\|\bar{\boldsymbol{X}}-\bar{\boldsymbol{X}}_r\|_{\mathrm{F}}=\sqrt{\tilde{\sigma}_{r+1}^2+\cdots+\tilde{\sigma}_N^2},$$

式中 $\tilde{\sigma}_1\geqslant\tilde{\sigma}_2\geqslant\cdots\geqslant\tilde{\sigma}_N$。

事实上如果我们确定了一个误差 ϵ，则项的最小数目 r 由以下要求来确定

$$\tilde{\sigma}_{r+1}^2+\cdots+\tilde{\sigma}_N^2\leqslant\leqslant\epsilon^2\text{或}\ \tilde{\sigma}_r^2+\cdots+\tilde{\sigma}_N^2>\epsilon^2 \tag{7}$$

这意味着在进行 r 项逼近时，需要将张量 $\bar{\boldsymbol{X}}$ 的奇异值重新排序，且逐项计算达到精度时所需要的项数，还原时还要找到每个奇异值所对应的外积，这无疑是费力且耗时巨大的，且不能用推论 2 进行矩阵表示。因此，我们试图寻找一种更简便的方法。

假设 $\bar{\boldsymbol{X}}_r$ 的奇异值分别取自 $\boldsymbol{S}$ 中每列 s_i 的 r_{2i} 个元素，若 $r_{2i}=0$，则该列的元素全部舍去，设有 l 列全部舍去，结合推论 2 进一步分析截断误差可得

$$\|\bar{\boldsymbol{X}}-\bar{\boldsymbol{X}}_r\|_{\mathrm{F}}^2=\|\bar{\boldsymbol{X}}\|_{\mathrm{F}}^2-\|\bar{\boldsymbol{X}}_r\|_{\mathrm{F}}^2=\sum_{i=1}^{r_1}\sigma_i{}^2\sum_{j=1}^{r_{2i}}\sigma_{ij}{}^2=\underbrace{\sigma_{\pi_1}{}^2+\cdots+\sigma_{\pi_l}{}^2}_{\epsilon_1^2}+\underbrace{\sum_{\pi_i=l+1}^{r_1}\sigma_{\pi_i}{}^2\sum_{j=r_{2\pi_i}+1}^{r_2}\sigma_{\pi_ij}{}^2}_{\epsilon_2^2}, \tag{8}$$

式中 $\pi_i(i=1,\cdots,r_1)$表示按照舍去项数从多到少对每列奇异值重新排序。根据(7)和(8)式，截断 TTr1SVD 的误差来自计算 $\boldsymbol{X}_1$ 的 SVD 时，选取 l 项 σ_i 进行舍弃产生的误差 ϵ_1 和计算 r_1-l 个 $\tilde{\boldsymbol{V}}_{\pi_i}$ 的 SVD 时，进行 $r_{2\pi_i}$ 项 σ_{ij} 截断时产生的误差和 ϵ_2。若 $\epsilon_1^2+\epsilon_2^2\leqslant\epsilon^2$，则 $\|\bar{\boldsymbol{X}}-\bar{\boldsymbol{X}}_r\|_{\mathrm{F}}^2\leqslant\epsilon^2$。

然而，在不进行排序的情况下，精确选取需要舍去的 l 个奇异值是非常困难的。因此，我们设计了具有固定精度的截断 TTr1SVD 算法 (δ-TTr1SVD)，在不需要复杂排序的情况下实现了近似张量 $\bar{\boldsymbol{X}}_r$ 的快速计算，且仍然可以表示成推论 2 的形式。

对于给定误差 ϵ，首先对 $\boldsymbol{X}_1$ 进行 ϵ-截断 SVD，使得 $\boldsymbol{X}_1=\boldsymbol{U}'_1\boldsymbol{S}'_1\boldsymbol{V}'^{\mathrm{T}}_1+\boldsymbol{E}_1$，$\|E_1\|_{\mathrm{F}}=\epsilon'_1\leqslant\epsilon$，$r'_1=\mathrm{rank}_\epsilon(\boldsymbol{X}_1)$；

此时第 1 层叶子节点总数更新为 r'_1，$\boldsymbol{U}'_1$，$\boldsymbol{V}'_1$ 分别对应 $\boldsymbol{U}_1$，$\boldsymbol{V}_1$ 的前 r'_1 列，$\boldsymbol{S}'_1$ 的对角线元素为 $\sigma_1,\cdots,\sigma_{r'_1}$。

注意到对第 1 层每个节点需满足

$$\epsilon_{2i}^2=\sigma_i^2\sum_{j=r_{2i}+1}^{r_2}\sigma_{ij}^2,$$

将 ϵ_{2i} 更新为 $\epsilon_{2i}=\epsilon_{2i}/\sigma_i$。接下来对 $\widetilde{\boldsymbol{V}}_i$，$i=1,\cdots,r'_1$ 依次进行 ϵ_{2i}-截断 SVD，使得

$$\widetilde{\boldsymbol{V}}_i=\boldsymbol{U}'_{1i}\boldsymbol{S}'_{1i}\boldsymbol{V}'^{\mathrm{T}}_{1i}+\boldsymbol{E}_{1i},$$

$$\|\boldsymbol{E}_{1i}\|_{\mathrm{F}}=\epsilon'_{2i}\leqslant\epsilon_{2i},r'_{2i}=\mathrm{rank}_{\epsilon_{2i}}(\widetilde{\boldsymbol{V}}_i)。$$

此时，第 2 层叶子节点总数即分解产生的秩-1 项的数量更新为 $N_1=\sum_{i=1}^{r'_1}r'_{2i}$，$\boldsymbol{U}'_{1i}$，$\boldsymbol{V}'_{1i}$ 分别对应 $\boldsymbol{U}_{1i}$，$\boldsymbol{V}_{1i}$ 的前 r'_{2i} 列，$\boldsymbol{S}'_{1i}$ 的对角线元素为 $\sigma_{i1},\cdots,\sigma_{ir'_{2i}}$。

注 2　对于给定的矩阵 $\boldsymbol{X}$，其 ϵ-rank 定义为所有满足 $\|\boldsymbol{X}-\widetilde{\boldsymbol{X}}\|\leqslant\epsilon$ 的矩阵 $\widetilde{\boldsymbol{X}}$ 中 $\mathrm{rank}(\widetilde{\boldsymbol{X}})$ 的最小值，记为 $\mathrm{rank}_\epsilon(\boldsymbol{X})$；其 ϵ-截断 SVD（ϵ-truncated SVD），定义为对矩阵 $\boldsymbol{X}$ 的 SVD 进行 $\mathrm{rank}_\epsilon(\boldsymbol{X})$ 项截断操作。

若 $\epsilon_1'^2+\sum_{i=1}^{r'_1}\sigma_i^2\epsilon_{2i}'^2\leqslant\epsilon$，则 $\|\bar{\boldsymbol{X}}-\widehat{\bar{\boldsymbol{X}}}\|_{\mathrm{F}}^2\leqslant\epsilon^2$。这就产生了一个解决固定精度问题的框架，即算法 1。若想得到相对误差为 δ 的 $\widehat{\bar{\boldsymbol{X}}}$，则误差 $\epsilon=\delta\|\bar{\boldsymbol{X}}\|_{\mathrm{F}}$。下面证明算法 1 的正确性。

定理 1　设 $\widehat{\bar{\boldsymbol{X}}}$ 是由算法 1 得到的 $\bar{\boldsymbol{X}}$ 的具有固定精度 δ 的近似张量，则

$$\|\bar{\boldsymbol{X}}-\widehat{\bar{\boldsymbol{X}}}\|_{\mathrm{F}}^2\leqslant\delta^2\|\bar{\boldsymbol{X}}\|_{\mathrm{F}}^2。$$

证明　由推论 1 和推论 2 可得

$$\|\bar{\boldsymbol{X}}-\widehat{\bar{\boldsymbol{X}}}\|_{\mathrm{F}}^2=\|\bar{\boldsymbol{X}}\|_{\mathrm{F}}^2-\|\widehat{\bar{\boldsymbol{X}}}\|_{\mathrm{F}}^2=$$

$$\sum_{i=1}^{r_1}\sum_{j=1}^{r_2}(\sigma_i\sigma_{ij})^2-\sum_{i=1}^{r'_1}\sum_{j=1}^{r'_{2i}}(\sigma_i\sigma_{ij})^2=$$

$$(\sigma_{r'_1+1}^2+\cdots+\sigma_{r_1}^2)+\sum_{i=1}^{r'_1}\sum_{j=r'_{2i}+1}^{r_2}(\sigma_i\sigma_{ij})^2。$$

算法 1 的第 4 步和第 9 步将两部分误差表示为

$$\|\bar{\boldsymbol{X}}-\widehat{\bar{\boldsymbol{X}}}\|_{\mathrm{F}}^2=\epsilon_1^2+\sum_{i=1}^{r'_1}\sigma_i^2\epsilon_{2i}'^2\leqslant\epsilon_1^2+\sum_{i=1}^{r'_1}\sigma_i^2\epsilon_{2i}^2=\epsilon_1^2+\epsilon_2^2$$

由 $\epsilon=\delta\|\bar{\boldsymbol{X}}\|_{\mathrm{F}}$ 和算法第 5 步：$\epsilon_1^2+\epsilon_2^2=\epsilon^2$，可得到想要的结果。证明完毕。

由算法 1 的第 7 步可以看出对于第 1 层的每个节点采取了截断误差平均分配的方法。为了减少项数 R，还可以使用动态分配误差的方法，粗略的想法是在循环内完成误差的更新：$\epsilon_2^2=\epsilon^2-\epsilon_1^2-\epsilon_{2i}'^2$，并在第 7 步重新分配误差：$\epsilon_{2i}^2=\epsilon_2^2/[(r_1-i+1)\boldsymbol{S}_1^2(i,i)]$。

算法 1：δ－TTr1SVD

输入：$\bar{\boldsymbol{X}}\in\mathbf{R}^{n_1\times n_2\times n_3}$，固定精度 δ

输出：$\boldsymbol{X}^{(2)}\in\mathbf{R}^{n_2\times N_1}$，$X^{(3)}\in\mathbf{R}^{n_3\times N_1}$，$\boldsymbol{X}^{(1)}\in\mathbf{R}^{n_1\times r_1}$，$\boldsymbol{S}\in\mathbf{R}^{r_2\times r_1}$

1　确定误差 ϵ：$\|\bar{\boldsymbol{X}}-\widehat{\bar{\boldsymbol{X}}}\|_{\mathrm{F}}\leqslant\delta\|\bar{\boldsymbol{X}}\|_{\mathrm{F}}=\epsilon$；
2　初始化每层节点个数：$r_1=\min(n_1,n_2n_3)$，$r_2=\min(n_2,n_3)$；
3　$\boldsymbol{X}_1=\mathrm{reshape}(\bar{\boldsymbol{X}},[n_1,n_2n_3])$；
4　计算 ϵ-截断 SVD：$\boldsymbol{X}_1=\boldsymbol{U}_1\boldsymbol{S}_1\boldsymbol{V}_1^{\mathrm{T}}+\boldsymbol{E}_1$，$\|\boldsymbol{E}_1\|_{\mathrm{F}}=\epsilon_1\leqslant\epsilon,r'_1=\mathrm{rank}_\epsilon(\boldsymbol{X}_1)$；
5　更新 $r_1=r'_1$，ϵ_2：$\epsilon_2^2=\epsilon^2-\epsilon_1^2$；
6　For $i=1$ to r_1
7　　分配 ϵ_{2i}：$\epsilon_{2i}^2=\epsilon_2^2/(r_1\boldsymbol{S}_1^2(i,i))$；
8　　$\widetilde{\boldsymbol{V}}_i=\mathrm{reshape}(\boldsymbol{V}_1(:,i),[n_2,n_3])$；
9　　计算 ϵ_{2i}-截断 SVD：$\widetilde{\boldsymbol{V}}_i=\boldsymbol{U}_{1i}\boldsymbol{S}_{1i}\boldsymbol{V}_{1i}^{\mathrm{T}}+\boldsymbol{E}_{1i}$，$\|\boldsymbol{E}_{1i}\|_{\mathrm{F}}=\epsilon'_{2i}\leqslant\epsilon_{2i},r'_{2i}=\mathrm{rank}_{\epsilon_{2i}}(\widetilde{\boldsymbol{V}}_i)$；
10　　$\boldsymbol{X}_i^{(2)}=\boldsymbol{U}_{1i}$，$\boldsymbol{X}_i^{(3)}=\boldsymbol{V}_{1i}$，$\boldsymbol{x}_i^{(1)}=\boldsymbol{U}_1(:,i)$；
11　　For $i=1$ to r'_{2i}
12　　　$\boldsymbol{S}(j,i)=\boldsymbol{S}_1(i,i)\boldsymbol{S}_{1i}(j,j)$
13　　end
14　end

3　固定精度的随机 TTr1SVD

随机算法在规模较大矩阵的低秩逼近中起着关键作用。本节回顾低秩矩阵近似中的固定精度问题，提出固定精度的随机 TTr1SVD（δ-RTTr1SVD）算法。最后通过讨论张量的分解与还原过程得出算法的计算成本。

3.1 固定精度的随机 SVD

随机化技术的一个基本思想是使用随机投影来近似一个矩阵的主子空间。对于 $m\times n$（$m\geqslant n$）的矩阵 $\boldsymbol{A}$，$\boldsymbol{A}$ 乘以具有 $k+s$ 列的随机高斯矩阵 $\boldsymbol{\Omega}$ 得到 $\boldsymbol{Y}=\boldsymbol{A}\boldsymbol{\Omega}$，其中 s 为过采样参数，通常为较小的整数。对 $\boldsymbol{Y}$ 进行 QR 分解后得到正交矩阵 $\boldsymbol{Q}$，即为矩阵的近似子空间的正交基相乘的正交矩阵，则 $\boldsymbol{A}\approx\boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{A}=\boldsymbol{Q}\boldsymbol{B}$。此类算法最早在文献[10]中提出，我们称之为 randQB 算法。

低秩矩阵近似中固定精度问题是指寻找尽可能小的 $\boldsymbol{Q}$ 和 $\boldsymbol{B}$，使得 $\|\boldsymbol{A}-\boldsymbol{Q}\boldsymbol{B}\|_{\mathrm{F}}\leqslant\epsilon$，$\epsilon$ 为给定误差。文献[9]中提出 randQB 算法变体，适用于固定精度问题。随后在文献[10]的算法 2 中进行了改进，假设它可以被调用为 $[\boldsymbol{Q},\boldsymbol{B}]=\mathrm{randQB\_EI}(\boldsymbol{A},\delta,b,q)$，其中 $\boldsymbol{A}$ 是要近似的矩阵，δ 为给定相对误差。b 表示分块大小，用来确定选取随机高斯矩阵的列数，根据矩阵大小进行调整。q 为迭代参数，用来将算法中的投影步骤 $\boldsymbol{Y}=\boldsymbol{A}\boldsymbol{\Omega}$ 被替换为 $\boldsymbol{Y}=(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^q\boldsymbol{A}\boldsymbol{\Omega}$，在许多应用中，$q=1$ 时

算法已经达到了足够的精度。若继续对得到的低秩矩阵 $\boldsymbol{B}$ 进行 SVD，$\boldsymbol{B}=\boldsymbol{U}'\boldsymbol{S}\boldsymbol{V}^{\mathrm{T}}$，$\boldsymbol{A}\approx\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathrm{T}}$ 其中 $\boldsymbol{U}=\boldsymbol{Q}\boldsymbol{U}'$。上述过程我们称之为具有固定精度的随机 SVD 算法，记为 δ-RSVD。

3.2 固定精度的随机 TTr1SVD

现在我们将矩阵层面的随机算法推广到张量层面。最直接的方法就是将 δ-RSVD 应用到 TTr1SVD 算法的 r_1+1 次 SVD 中，这种应用在算法 1 中很容易实施，而推广的关键在于相对误差 δ 的分配。根据算法 1，若想得到相对误差为 δ 的 $\widehat{\bar{\boldsymbol{X}}}$，则误差 $\epsilon=\delta\|\bar{\boldsymbol{X}}\|_{\mathrm{F}}$。

第一次 δ_1-RSVD 是对 $\boldsymbol{X}_1$ 进行近似，由 (1)式得 $\|\bar{\boldsymbol{X}}\|_{\mathrm{F}}=\|\boldsymbol{X}_1\|_{\mathrm{F}}$，则 $\delta_1=\delta$，此时

$$\boldsymbol{X}_1\approx\widetilde{\boldsymbol{X}}_1=Q_1B_1=\widehat{\boldsymbol{U}}_1\widehat{\boldsymbol{S}}_1\widehat{\boldsymbol{V}}_1^{\mathrm{T}},\widehat{r}_1=\mathrm{rank}(\widehat{\boldsymbol{S}}_1);$$

第 1 层的叶子点数更新为 $\widehat{r}_1$。接下来对 $\widehat{\boldsymbol{V}}_i$ 的每列重塑为矩阵 $\widetilde{\boldsymbol{V}}_i$，依次进行 $\widehat{r}_1$ 次 δ_{2i}-RSVD，每次分配误差为 $\epsilon_{2i}^2=((\|\boldsymbol{X}_1\|_{\mathrm{F}}^2-\|\widehat{\boldsymbol{X}}_1\|_{\mathrm{F}}^2)/\widehat{r}_1)\widehat{\boldsymbol{S}}_{1i}^2$，由于 $\epsilon_{2i}^2=\delta_{2i}^2\|\widetilde{\boldsymbol{V}}_i\|_{\mathrm{F}}^2$，$\widetilde{\boldsymbol{V}}_i$ 是由单位向量重塑而来，则 $\epsilon_{2i}^2=\delta_{2i}^2$，此时

$$\widetilde{\boldsymbol{V}}_i\approx\widehat{\widetilde{\boldsymbol{V}}}_i=Q_{1i}B_{1i}=\widehat{\boldsymbol{U}}_{1i}\widehat{\boldsymbol{S}}_{1i}\widehat{\boldsymbol{V}}_{1i}^{\mathrm{T}},\widehat{r}_{2i}=\mathrm{rank}(\widehat{\boldsymbol{S}}_{1i});$$

第 2 层叶子节点总数即分解产生的秩-1 项的数量更新为 $N_2=\sum_{i=1}^{\widehat{r}_1}\widehat{r}_{2i}$。

随机算法依旧可以保持原分解的结构，只需将算法 1 的所有 ϵ-截断 SVD 全部替换为 δ-RSVD，即可得到 δ-RTTr1SVD 算法，记为算法 2。由推论 1、定理 1 和引理 1 分析算法 2 产生的近似张量的误差与张量奇异值之间的关系，可以得到定理 2。

定理 2 设 $\widehat{\bar{\boldsymbol{X}}}$ 是由算法 2 得到的 $\bar{\boldsymbol{X}}$ 的具有固定精度 δ 的近似张量，则

$$\|\bar{\boldsymbol{X}}-\widehat{\bar{\boldsymbol{X}}}\|_{\mathrm{F}}^2\leqslant\delta^2\sum_{i=1}^{N}\widetilde{\sigma}_i^2<\sum_{r}^{N}\widetilde{\sigma}_r^2。$$

其中 $\widetilde{\sigma}_i$ 为 $\bar{\boldsymbol{X}}$ 按照降序排列的第 i 个奇异值，r 为 $\bar{\boldsymbol{X}}$ 进行 r 项逼近算法时得到的具有固定精度 δ 的秩-1 张量项数。

3.3 算法的计算成本和存储成本

TTr1SVD 的分解过程主要是由分解需要的 SVD 的次数决定的。而随机算法大大减少了分解所需要的时间。由上文可知 δ-TTr1SVD 进行 r'_1+1 次 SVD，产生的秩-1 项为 N_1，δ-RTTr1SVD 进行 $\widehat{r}+1$ 次 RSVD，产生的秩-1 项为 N_2。张量还原的过程是 TTr1SVD 的逆过程，我们有两种方法可以得到还原张量：一种将定义 4 的 TTr1SVD 表达形式中的各列向量依次进行克罗内克（Kronecker）积[5]之后，将得到的列向量重塑为原始维数的张量并逐项相加；另一种为利用推论 2 的表达形式，进行矩阵与向量外积运算并逐步相加，分情况进行置换运算。r 项逼近算法无法表示出推论 2 的形式，所以采用前者进行还原，假设 r 项逼近的秩-1 项为 N，则要进行 N 次张量加法；δ-TTr1SVD 和 δ-RTTr1SVD 用后者进行还原，需要进行的张量加法缩减为 r'_1、$\widehat{r}$ 次。

表 1 给出了 δ-TTr1SVD、δ-RTTr1SVD 以及 r 项逼近方法的计算成本和存储成本。而张量的计算成本又包括了分解成本和还原成本。为了更直观的展示，我们只记录主要计算成本，假设张量的三个维度相等，即 $n_1=n_2=n_3=n$。

表 1 三种算法计算成本和存储成本的比较

Table 1 Comparison of the computational and storage costs of the three algorithms

算法 Algorithm	分解成本 Decomposition costs	还原成本 Reduction costs	存储成本 Storage costs
r 项逼近 r term approximation	$n^4+r_1n^3$	$2Nn^3$	$N(3n+1)$
δ-TTr1SVD	$n^4+r'_1n^3$	$2r'_1n^3+N_1n^2$	$N_1(2n+1)+r'_1n$
δ-RTTr1SVD	$\widehat{r}_1n^3+N_2n^2$	$2\widehat{r}_1n^3+N_2n^2$	$N_2(2n+1)+\widehat{r}_1n$

4 数值算例

在本节我们通过数值算例对提出的 δ-TTr1SVD、δ-RTTr1SVD 算法与 TTr1SVD 算法 r 项逼近算法（r-TTr1SVD）进行比较，充分验证两种算法在数值近似和数据压缩方面的性能和优势。此外，为评估所提出的随机算法的优势，我们还与文献[14]中提到自适应随机高阶奇异值分解算法（Adaptive R-HOSVD）和自适应随机顺序截断高阶奇异值分解算法（Adaptive R-STHOSVD）进行了比较。所有的计算基于 Matlab Tensor Toolbox[17]。所有的实验进行三次且取平均值作为实验结果。

由算法 1 和算法 2 可知，固定精度的张量分解所得秩-1 项的数目（N）不仅依赖于阶数的排序，还与奇异值的分布有关。“累积奇异能量”表示奇异值与所有奇异值总和的累计和，其分布是描述奇异值与所有奇异值总和的曲线，通常称为奇异能量的分布[18]。压缩率（Ratio）定义为原张量元素数与分解后所需元素数之比。

例 1 我们利用高光谱图像 Salinas① 探究分解顺序与

① https://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes#Salinas

奇异值的分布对算法的影响。该图像是由 512×217 像素和 224 个光谱波段组成，记为 $\bar{\boldsymbol{X}}\in\mathbf{R}^{512\times217\times224}$。三阶张量的后两个阶数的顺序不会影响 TTr1SVD，所以表 2 只展示了对分解顺序为 $\rho_1=[1,2,3]$，$\rho_2=[2,1,3]$，$\rho_3=[3,1,2]$进行分解时间 t_1、还原时间 t_2、秩-1 项数量 N、压缩率 ratio 在固定精度 $\delta=0.05$ 下的比较。δ-RTTr1SVD 用到的参数为 $[b_1,b_2]=[40,30]$，$q=1$。图 2 展示了 $\bar{\boldsymbol{X}}$ 分别沿三个分解顺序展开第一次形成矩阵的奇异能量分布。由图 2 可知，张量在分解顺序为 ρ_3 的情况下，$\boldsymbol{X}_3$ 的奇异值能量累计到 1 的速度最快，$\boldsymbol{X}_2$ 次之。

表 2　三种算法的结果比较

Table 2　Comparison of the results of the three algorithms

ρ	算法 Algorithm	分解时间 Decomposition time t_1/s	还原时间 Reduction time t_2/s	秩-1 项数 Number of rank-1 terms	压缩率 Ratio
ρ_1	r-TTr1SVD	10.51	13.76	180	144.92
	δ-TTr1SVD	4.09	0.55	387	120.29
	δ-RTTr1SVD	1.19	0.94	760	66.39
ρ_2	r-TTr1SVD	6.49	8.81	116	224.89
	δ-TTr1SVD	3.31	0.83	352	92.37
	δ-RTTr1SVD	1.17	1.03	620	40.56
ρ_3	r-TTr1SVD	6.88	9.14	111	235.02
	δ-TTr1SVD	2.75	0.27	126	248.99
	δ-RTTr1SVD	1.02	0.31	160	211.85

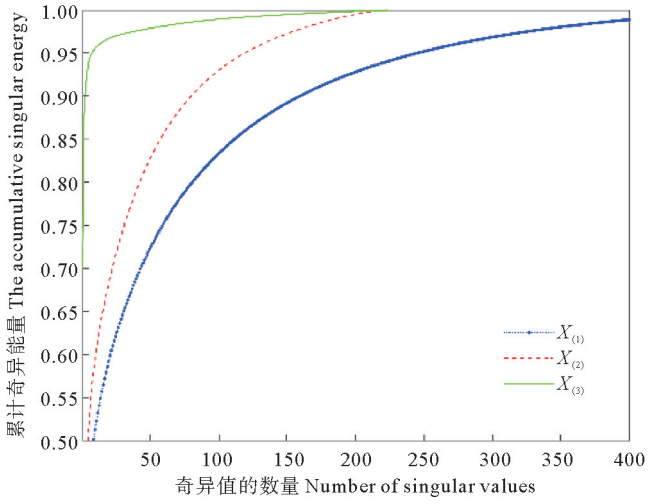


图 2　张量 $\bar{\boldsymbol{X}}$ 沿三个方向展开的累计奇异能量的分布

Fig. 2　The distributions of accumulative singular energy of the tensor $\bar{\boldsymbol{X}}$ unfolded along the three directions

由表 2 可得，三种算法在三种分解顺序下产生的秩-1 项数量依次增加，在 ρ_1、ρ_2 分解顺序下，δ-TTr1SVD 和 δ-RTTr1SVD 虽然产生的秩-1 项数较多，但仍然比 r 项逼近的时间少得多，且压缩率不随项数的增加成比例下降，在一定程度上说明提出的算法的有效性；在 ρ_3 分解顺序下产生的秩-1 项数与 r 项逼近产生的秩-1 项数最接近，与其他分解顺序相比，产生的秩-1 项数最少，在保持良好压缩率的同时，大大缩短了分解时间和还原时间，其中分解时间的加速比分别为 2.5 和 6.7 倍，还原时间的加速比分别为 33.8 和 29.5 倍。

例 2　本例从腾讯视频上截取了一段关于萝卜生长的延时拍摄视频①。每帧构成视频张量的 RGB 图片又可以视为 234×423×3 的张量，加上视频本身 150 帧，从而视频所有数据整体视为 4 维张量$\widetilde{\boldsymbol{Z}}\in\mathbf{R}^{234\times432\times3\times150}$。为简化计算，将张量$\widetilde{\boldsymbol{Z}}$的第 3 和第 4 维度进行合并，即$\bar{\boldsymbol{Z}}=\mathrm{reshape}(\widetilde{\boldsymbol{Z}},[n_1,n_2,n_3n_4])$，从而得到 234×432×450 的三阶张量 $\bar{\boldsymbol{Z}}$。通过设置不同精度对 $\bar{\boldsymbol{Z}}$ 进行分解，并且将分解得到的不同数量的秩-1 张量进行还原，得到还原张量$\hat{\bar{\boldsymbol{Z}}}$，即可得到不同清晰度的视频。将运行总时间记为 t_{tot}，用峰值信噪比(Peak-signal-to-noise ratio，PSNR)来衡量视频图像帧在进行上述处理前后的差别。PSNR 的单位是 dB，数值越大表示失真越小。一般来说，如果 PSNR 的值高于 40 dB，则说明还原出来的帧图像的效果十分接近原始帧图像，30～40 dB，则表示还原图像的失真虽然可以被察觉，但仍然可以接受；若在 20～30 dB 则说明图像质量差；当 PSNR 低于 20 dB 图像不可接受。PSNR 的公式：

$$\mathrm{PSNR}=20\log_{10}\left(\frac{\max(\bar{\boldsymbol{Z}})}{\sqrt{\mathrm{MSE}}}\right),$$

其中 MSE 为张量的均方差，$\mathrm{MSE}=\dfrac{\|\bar{\boldsymbol{Z}}-\hat{\bar{\boldsymbol{Z}}}\|_{\mathrm{F}}^2}{n_1\times n_2\times n_3}$。由图 3 的曲线斜率可知 $\boldsymbol{Z}_3$ 的奇异值能量累计速度最快，所以设定分解顺序为 $\rho_3=[3,1,2]$。δ-RTTr1SVD 用到的参数为 $[b_1,b_2]=[40,30]$，$q=1$。

表 3 展示了三种算法在不同精度下的运行总时间、PSNR、秩-1 项的数目以及压缩率。δ-TTr1SVD 和 δ-RTTr1SVD 的运行时间远远小于 r 项逼近，主要体现在固定精度为：在 $\delta=0.05$ 的加速比分别为 7.0 和 20.1 倍，在 $\delta=0.03$ 的加速比分别为 6.4 和 15.0 倍。在同一精度三种分解产生的秩-1 项虽然差距很大，但并没有导致压缩比率大幅下降，说明算法的有效性。值得注意的是，在 $\delta=0.03$ 时 δ-RTTr1SVD 产生的秩-

① https://v.qq.com/x/page/v3237ztwzs7.html

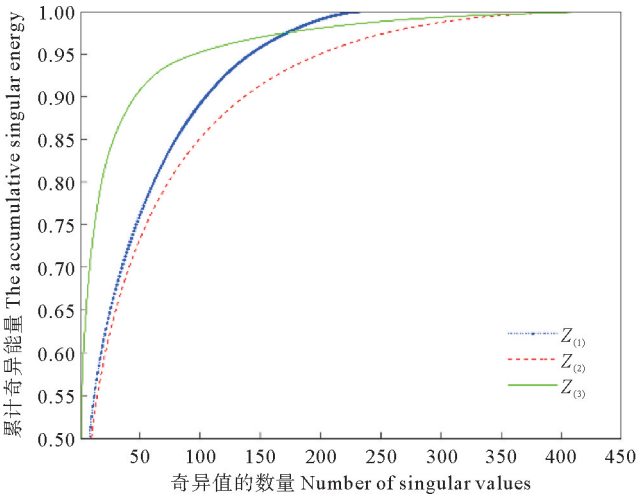


图 3 张量 $\bar{\boldsymbol{Z}}$ 沿三个方向展开的累计奇异能量的分布

Fig. 3 The distributions of accumulative singular energy of the tensor $\bar{\boldsymbol{Z}}$ unfolded along the three directions

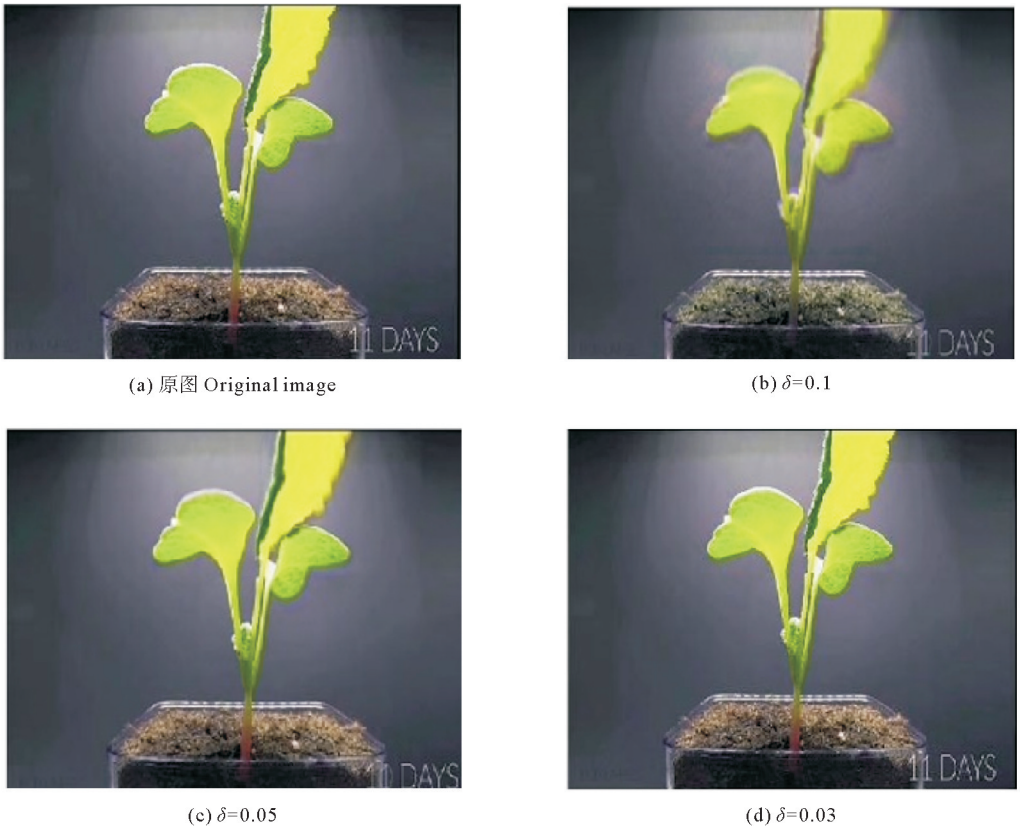
1 项少于 δ-TTr1SVD，其原因在于算法在进行第一次随机 SVD 时，产生的右奇异向量在进行第二次随机 SVD 时产生的奇异值能量分布情况比算法 1 要好。这从侧面反映出随机算法的稳定性。

图 4 将视频原图像与在不同精度下的还原图像进行对比，以 δ-RTTr1SVD 为例，$\delta=0.1$ 时，图像出现重影质量很差；$\delta=0.05$ 时，图像中的叶子边缘较模糊，但在可接受的范围内；$\delta=0.03$ 时，图像与原图差别不大，说明图像的还原效果很好。综上所述，基于 δ-RTTr1SVD 给出的视频压缩效果最好，δ-TTr1SVD 优于 r 项逼近。

表 3 三种算法的结果比较

Table 3 Comparison of the results of the three algorithms

δ	算法 Algorithm	总时间 Total time t_{tot}/s	峰值信噪比 PSNR	秩-1 项数 Number of rank-1 terms	压缩率 Ratio
0.1	r-TTr1SVD	35.67	28.81	116	351.07
	δ-TTr1SVD	7.03	29.63	294	228.82
	δ-RTTr1SVD	2.64	30.56	360	186.99
0.05	r-TTr1SVD	104.56	34.43	631	64.54
	δ-TTr1SVD	14.97	34.65	1 531	44.21
	δ-RTTr1SVD	5.20	35.52	1 840	36.80
0.03	r-TTr1SVD	173.58	39.51	1 693	24.05
	δ-TTr1SVD	26.96	39.60	5 366	12.67
	δ-RTTr1SVD	11.61	40.53	4 140	16.38



(a) 原图 Original image (b) δ=0.1

(c) δ=0.05 (d) δ=0.03

图 4 不同固定精度下的压缩图像与原图的比较

Fig. 4 Comparison of compressed image with original image under different fixed precision

例 3　本例利用例 1 的张量 $\bar{\boldsymbol{X}}\in\mathbf{R}^{512\times217\times224}$ 和例 2 的张量 $\bar{\boldsymbol{Z}}\in\mathbf{R}^{234\times432\times450}$ 进行了在相同处理顺序 $\rho_3=[3,1,2]$ 和精度 δ 的情况下，设置不同参数时 δ-RTTr1SVD，Adaptive R-HOSVD 和 Adaptive R-STHOSVD 算法的比较。为了方便比较，我们设 $b_1=b_2=b$，$q_1=q_2=q$。

表 4 展示了三种张量随机算法的分解时间、秩-1 项数和压缩率。从表 4 可以看出，在 δ 相同时改变参数 b 和 q，对 Adaptive R-HOSVD 和 Adaptive R-STHOSVD 两种算法的影响较大，主要体现在 b 增大时，虽然时间会减少，但秩-1 项数会大幅度增加，压缩率减小，q 增大时，时间会增加，秩-1 项数改变不大，而 δ-RTTr1SVD 算法依然能保持较短的分解时间、较少的秩-1 项数和较高的压缩率，这证明了我们所提算法的稳定性。对于张量 $\bar{\boldsymbol{X}}$，δ-RTTr1SVD 分解时间与 Adaptive R-STHOSVD 相差不大，但压缩率最高为后者的 2.65 倍。对于张量 $\bar{\boldsymbol{Z}}$，我们发现 δ-RTTr1SVD 在 $\delta=0.1$ 时，压缩率比另外两种算法高。在 $\delta=0.05$ 时，在压缩率相差不大的情况下，δ-RTTr1SVD 在时间和秩-1 项数上仍然比 Adaptive R-HOSVD 具有优势。对于两个张量的实验结论不同，是 $\bar{\boldsymbol{X}}$ 与 $\bar{\boldsymbol{Z}}$ 在第三个维度展开为矩阵时的奇异值能量累计速度不同导致的，具体原因留给以后研究。综上所述，在同一精度下，δ-RTTr1SVD 具有一定稳定性，可以更好地规范张量秩。

表 4　三种随机算法的结果比较

Table 4　Comparison of the results of the three randomized algorithms

张量 Tensor	δ	参数 Parameters	δ-RTTr1SVD			Adaptive R-HOSVD			Adaptive R-STHOSVD		
			分解时间 Decomposition time t_1/s	秩-1 项数 Number of rank-1 terms	压缩率 Ratio	分解时间 Decomposition time t_1/s	秩-1 项数 Number of rank-1 terms	压缩率 Ratio	分解时间 Decomposition time t_1/s	秩-1 项数 Number of rank-1 terms	压缩率 Ratio
$\bar{\boldsymbol{X}}$	0.02	$b=5,q=1$	0.98	495	68.66	13.02	369 750	46.60	0.91	325 000	51.47
		$b=20,q=1$	1.19	480	70.80	6.98	832 000	24.78	0.84	728 000	27.77
		$b=5,q=2$	0.97	410	81.84	19.06	350 000	48.73	1.25	306 250	53.97
		$b=20,q=2$	1.27	460	73.86	9.21	728 000	27.77	1.24	728 000	27.77
$\bar{\boldsymbol{Z}}$	0.1	$b=40,q=1$	1.62	360	186.99	4.27	256 000	138.99	1.12	256 000	138.99
	0.05	$b=40,q=1$	1.87	1 520	44.50	7.85	672 000	58.42	1.48	576 000	67.50

5　结语

在本文中，我们给出了 TTr1SVD 的不同表达形式，并改进了 r 项逼近算法，提出了一种高效的具有固定精度的截断 TTr1SVD 算法，验证了其准确性。在此基础上，设计了具有固定精度的随机 TTr1SVD 算法，将最新的随机化策略推广到张量分解上。同时，还比较了提出的算法与原始的近似方法的计算成本和存储成本。此外，我们还用数值算例验证了算法在图像与视频压缩方面的有效性。

针对如何有效减少算法产生的正交秩-1 项的问题，我们可以使用稀疏字典表示方法[19]表示张量分解，然而我们需要克服表示方法不唯一的问题。未来，我们将继续探索 TTr1SVD 在其他领域的应用。

参考文献：

[1] Carroll J D, Chang J J. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition[J]. Psychometrika, 1970, 35(3): 283-319.

[2] Sidiropoulos N D, De Lathauwer L, Fu X, et al. Tensor decomposition for signal processing and machine learning[J]. IEEE Transactions on Signal Processing, 2017, 65(13): 3551-3582.

[3] Signoretto M, Tran Dinh Q, De Lathauwer L, et al. Learning with tensors: A framework based on convex optimization and spectral regularization[J]. Machine Learning, 2014, 94(3): 303-351.

[4] Kolda T G, Bader B W. Tensor decompositions and applications [J]. SIAM Review, 2009, 51(3): 455-500.

[5] Tucker L R. Some mathematical notes on three-mode factor analysis[J]. Psychometrika, 1966, 31(3): 279-311.

[6] Kilmer M E, Martin C D. Factorization strategies for third-order tensors[J]. Linear Algebra and its Applications, 2011, 435(3): 641-658.

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

[8] Batselier K, Liu H, Wong N. A constructive algorithm for decomposing a tensor into a finite sum of orthonormal rank-1 terms[J]. SIAM Journal on Matrix Analysis and Applications, 2015, 36(3): 1315-1337.

[9] Halko N, Martinsson P G, Tropp J A. Finding structure with ran-

domness: Probabilistic algorithms for constructing approximate matrix decompositions[J]. SIAM Review, 2011, 53(2): 217-288.

[10] Yu W, Gu Y, Li Y. Efficient randomized algorithms for the fixed-precision low-rank matrix approximation[J]. SIAM Journal on Matrix Analysis and Applications, 2018, 39(3): 1339-1359.

[11] Hallman E. A block bidiagonalization method for fixed-accuracy low-rank matrix approximation[J]. SIAM Journal on Matrix Analysis and Applications, 2022, 43(2): 661-680.

[12] Zhang J, Saibaba A K, Kilmer M E, et al. A randomized tensor singular value decomposition based on the t-product[J]. Numerical Linear Algebra with Applications, 2018, 25(5): e2179.

[13] Che M, Wei Y. Randomized algorithms for the approximations of Tucker and the tensor train decompositions[J]. Advances in Computational Mathematics, 2019, 45(1): 395-428.

[14] Minster R, Saibaba A K, Kilmer M E. Randomized algorithms for low-rank tensor decompositions in the Tucker format[J]. SIAM Journal on Mathematics of Data Science, 2020, 2(1): 189-215.

[15] Sorber L, Van Barel M, De Lathauwer L. Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-(L_r, L_r, 1) terms, and a new generalization[J]. SIAM Journal on Optimization, 2013, 23(2): 695-720.

[16] Eckart C, Young G. The approximation of one matrix by another of lower rank[J]. Psychometrika, 1936, 1(3): 211-218.

[17] Kolda T G, Bader B W. MATLAB tensor toolbox, Version 3.0[CP/OL]. (2007-03-07)[2022-09-01]. https://www.tensortoolbox.org. 2017.

[18] Li R, Pan Z, Wang Y, et al. The correlation-based Tucker decomposition for hyperspectral image compression[J]. Neurocomputing, 2021, 419: 357-370.

[19] Govindarajan N, Epperly E N, De Lathauwer L. (Lr, Lr, 1)-decompositions, sparse component analysis, and the blind separation of sums of exponentials[J]. SIAM Journal on Matrix Analysis and Applications, 2022, 43(2): 912-938.

Randomized Algorithms for Tensor TTr1SVD

Ding Minghui, Xie Pengpeng

(School of Mathematical Sciences, Ocean University of China, Qingdao 266100, China)

Abstract: The tensor-train rank-1 SVD (TTr1SVD) naturally generalizes the singular value decomposition (SVD) to the tensor scenario, which decomposes an arbitrary real tensor into a finite sum of orthonormal rank-1 outer products. Based on its excellent properties such as orthogonal rank-1 outer product terms with known upper limits of number and easy quantification of truncation errors, this paper first gives an expression that is conducive to tensor decomposition and reduction, and proposes a truncated TTr1SVD algorithm that maintains the decomposition form, which greatly reduces the computational cost while fixing the accuracy. Motivated by the development of randomized methods of low-rank matrix approximations, this paper also develops an efficient randomized TTr1SVD algorithm for fixed precision problems. Finally, numerical examples are given to demonstrate the promise of the proposed algorithm in data approximation and compression.

Key words: TTr1SVD; singular value decomposition; truncation; fixed precision problem; randomized algorithm

AMS Subject Classifications: 15A69; 65F30

责任编辑 朱宝象