CSE 392: Matrix and Tensor Algorithms for Data Spring 2024

Lecture 20 — 04/01/2024

Instructor: Shashanka Ubaru *Scribe: Shourya Pandey*

1 Randomized Tucker Decomposition

Last lecture, we saw the HOSVD and STHOSVD algorithms for the *tucker decomposition* of a tensor. The tucker decomposition of a three-mode tensor $\mathcal{X} \in \mathbb{R}^{m \times n \times p}$ is given by

$$\mathcal{X} \approx \mathcal{G} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C} =: [\![\mathcal{G}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!],$$

where $\mathcal{G} \in \mathbb{R}^{k_1 \times k_2 \times k_3}$ is called the *core* of the tensor $\mathcal{X}$ and $\boldsymbol{A} \in \mathbb{R}^{m \times k_1}$, $\boldsymbol{B} \in \mathbb{R}^{n \times k_2}$, and $\boldsymbol{C} \in \mathbb{R}^{p \times k_3}$ are called *factor* matrices. Typically, k_1, k_2, and k_3 are respectively much smaller than m, n, and p.

Both HOSVD and STHOSVD algorithms rely on matrix SVD algorithms. The first step towards a randomized tucker decomposition is to use randomized SVD instead of full SVD.

Algorithm 1: RandSVD [6]

Data: $\boldsymbol{X} \in \mathbb{R}^{m \times n}$, target rank $r \in \mathbb{N}$, oversampling parameter $p \geq 0$, $r + p \leq \min(m, n)$

Draw a random Gaussian matrix $\boldsymbol{\Omega} \in \mathbb{R}^{n \times (r+p)}$.

$\boldsymbol{Y} \leftarrow \boldsymbol{X}\boldsymbol{\Omega}$

QR-factorize $\boldsymbol{Y} = \boldsymbol{Q}\boldsymbol{R}$

$\boldsymbol{B} \leftarrow \boldsymbol{Q}^\top \boldsymbol{X}$

Calculate the thin-SVD $\boldsymbol{B} = \hat{\boldsymbol{U}}_B \hat{\boldsymbol{S}} \hat{\boldsymbol{V}}^\top$

$\hat{\boldsymbol{U}}, \hat{\boldsymbol{S}}, \hat{\boldsymbol{V}} \leftarrow \boldsymbol{Q}(\hat{\boldsymbol{U}}_B)_{:,1:r}, \hat{\boldsymbol{S}}_{1:r,1:r}, \hat{\boldsymbol{V}}_{:,1:r}$

return $[\hat{\boldsymbol{U}}, \hat{\boldsymbol{S}}, \hat{\boldsymbol{V}}]$

Algorithm 2: R-HOSVD [17]

Data: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, target rank vector $r \in \mathbb{N}^d$, oversampling parameter $p \geq 0$ such that $r_j + p \leq \min\left(n_j, \prod_{i \neq j} n_i\right)$ for all $j \in [d]$

for $j = 1, 2, \ldots, d$ **do**

  Draw a random Gaussian matrix $\boldsymbol{\Omega}_j \in \mathbb{R}^{\prod_{i \neq j} n_i \times (r+p)}$

  $[\hat{\boldsymbol{U}}, \hat{\boldsymbol{S}}, \hat{\boldsymbol{V}}] \leftarrow \mathsf{RandSVD}\left(\mathcal{A}_{(j)}, r_j, p, \boldsymbol{\Omega}_j\right)$

  $\boldsymbol{U}^{(j)} \leftarrow \hat{\boldsymbol{U}}$

$\mathcal{C} \leftarrow \mathcal{A} \times_{i=1}^{d} (\boldsymbol{U}^{(j)})^\top$

return $[\![\mathcal{C}; \boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(d)}]\!]$

The following theorem records the guarantee of R-HOSVD.

Theorem 1 (R-HOSVD [10]). *Let $\hat{\mathcal{A}} = [\![\mathcal{C}; \boldsymbol{U}^{(1)}, \dots, \boldsymbol{U}^{(d)}]\!]$ be the output of Algorithm 2 with input ranks $r = (r_1, r_2, \dots, r_d)$ and oversampling parameter $p \geq 2$, such that $r_j + p \leq \min\left(n_j, \prod_{i \neq j} n_i\right)$ for all $j \in [d]$. Then,*

$$\begin{aligned}
\mathop{\mathbb{E}}_{\{\Omega_j\}_{j\in[d]}} \left[\|\hat{\mathcal{A}} - \mathcal{A}\|_F\right] &\leq \left(\sum_{j=1}^{d} \left(1 + \frac{r_j}{p-1}\right) \Delta_j^2(\mathcal{A})\right)^{1/2} \\
&\leq \left(1 + \frac{\sum_{j=1}^{d} r_j}{p-1}\right)^{1/2} \|\mathcal{A} - \hat{\mathcal{A}}_{opt}\|_F, \qquad (1)
\end{aligned}$$

where $\Delta_j^2(\mathcal{A}) := \sum_{i=r_j+1}^{n_j} \sigma_i^2(\mathcal{A}_{(j)})$. Moreover, if $r = \max_{j\in[d]} r_j$ and $p = r + 1$, then

$$\mathop{\mathbb{E}}_{\{\Omega_j\}_{j\in[d]}} \left[\|\hat{\mathcal{A}} - \mathcal{A}\|_F\right] \leq \sqrt{2}\|\mathcal{A} - \hat{\mathcal{A}}_{HOSVD}\|_F \leq \sqrt{2d}\|\mathcal{A} - \hat{\mathcal{A}}_{opt}\|_F,$$

and if $p = \lceil r/\epsilon \rceil + 1$ for some $\epsilon > 0$, then

$$\mathop{\mathbb{E}}_{\{\Omega_j\}_{j\in[d]}} \left[\|\hat{\mathcal{A}} - \mathcal{A}\|_F\right] \leq \sqrt{1+\varepsilon}\|\mathcal{A} - \hat{\mathcal{A}}_{HOSVD}\|_F \leq \sqrt{d(1+\varepsilon)}\|\mathcal{A} - \hat{\mathcal{A}}_{opt}\|_F.$$

Remark 1. Similar to randomized HOSVD, there exists a randomized STHOSVD algorithm [10]. When bounding the error in expectation for R-STHOSVD, at each intermediate step the truncated core tensor is a random tensor. It achieves the same error in expectation as R-HOSVD independent of the processing order. In practice, a data-driven choice of the processing order generally makes R-STHOSVD computationally and statistically more efficient than R-HOSVD.

1.1 Dynamic Randomized HOSVD and STHOSVD

The algorithms described in the previous section requires knowledge of the target rank r. Given a tensor $\mathcal{A}$ it is desirable to find a low rank tensor $\hat{\mathcal{A}}$ such that

$$\|\hat{\mathcal{A}} - \mathcal{A}\|_F \leq \varepsilon\|\mathcal{A}\|_F, \qquad (2)$$

but picking the correct rank is often a difficult task. Many adaptive randomized range finders have been suggested, see [6, 9, 16]. Given a matrix $\boldsymbol{X}$, the goal here is to compute an orthonormal matrix $\boldsymbol{Q}$ of low rank such that $\|\boldsymbol{X} - \boldsymbol{Q}\boldsymbol{Q}^\top \boldsymbol{X}\|_F \leq \varepsilon\|\boldsymbol{X}\|$. This subroutine, called $\mathsf{AdaptiveRangeFinder}(\boldsymbol{X}, \varepsilon, b)$ can be used for an adaptive R-HOSVD algorithm. Here, $\boldsymbol{X}$ is the matrix to be approximated, ε is the tolerance parameter, and b is a called *blocking integer* which is a "step size" for the rank.

Algorithm 3: Adaptive R-HOSVD [10]

Data: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, tolerance $\varepsilon \in (0, 1)$, blocking integer $b \geq 1$

for $j = 1, 2, \dots, d$ **do**
  $\boldsymbol{U}^{(j)} \leftarrow \mathsf{AdaptiveRangeFinder}\left(\boldsymbol{X}_{(j)}, \varepsilon/\sqrt{d}, b\right)$
$\mathcal{C} \leftarrow \mathcal{A} \times_{i=1}^{d} (\boldsymbol{U}^{(j)})^\top$
return $[\![\mathcal{C}; \boldsymbol{U}^{(1)}, \dots, \boldsymbol{U}^{(d)}]\!]$

2 Preserving Tensor Structure

None of the above algorithms have guarantees on the structure of the tensor. If the tensor $\mathcal{A}$ has additional properties such as sparsity or non-negativity, can we compute decompositions of $\mathcal{A}$ such that the core preserves these structural properties? We are interested in computing a low-rank decomposition where the core $\mathcal{C}$ has entries taken from the original tensor $\mathcal{A}$, i.e. $\mathcal{A} \approx \mathcal{C} \times_{j=1}^{d} \boldsymbol{U}^{(j)}$, where $\boldsymbol{U}^{(j)}$ need not be orthonormal. We call such decompositions as *structure preserving* decompositions.

The idea is that instead of approximating $\boldsymbol{X} \approx \boldsymbol{Q}\boldsymbol{Q}^\top \boldsymbol{X}$ via adaptive range finding methods from [9, 16], we compute the strong rank-revealing QR-factorization of $\boldsymbol{Q}^\top$ as $\boldsymbol{Q}^\top \boldsymbol{S} = \boldsymbol{Z}\boldsymbol{N}$. Letting $\boldsymbol{P} := \boldsymbol{S}_{:,1:s}$, we define the oblique projector $\boldsymbol{Q}\left(\boldsymbol{P}^\top \boldsymbol{Q}\right)^{-1} \boldsymbol{P}^\top$ and apply it to $\boldsymbol{X}$:

$$\boldsymbol{X} \approx \boldsymbol{Q}\left(\boldsymbol{P}^\top \boldsymbol{Q}\right)^{-1} \underbrace{\boldsymbol{P}^\top \boldsymbol{X}}_{\hat{\boldsymbol{X}}}. \tag{3}$$

The matrix $\boldsymbol{Q}\left(\boldsymbol{P}^\top \boldsymbol{Q}\right)^{-1}$ doesn't necessarily have orthonormal columns but is well-conditioned, and that $\hat{\boldsymbol{X}}$ has rows from the matrix $\boldsymbol{X}$ determined by the operator $\boldsymbol{P}$ [10].

Algorithm 4: SP-STHOSVD [10]

Data: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, target rank vector $r \in \mathbb{N}^d$, oversampling parameter $p \geq 0$ such that $r_j + p \leq \min\left(n_j, \prod_{i \neq j} n_i\right)$ for all $j \in [d]$, processing order ρ

$\mathcal{C} \leftarrow \mathcal{A}$
for $j = 1, 2, \ldots, d$ **do**
  Draw a Gaussian matrix $\boldsymbol{\Omega}_{\rho_j} \in \mathbb{R}^{\prod_{\rho_i \neq \rho_j} n_{\rho_i} \times (r_{\rho_i} + p)}$
  $\boldsymbol{Y} \leftarrow \boldsymbol{C}_{\rho_j} \boldsymbol{\Omega}_{\rho_j}$
  Thin QR factorization $\boldsymbol{Y} \leftarrow \boldsymbol{Q}_{\rho_j} \boldsymbol{R}$
  Strong RRQR on $\boldsymbol{Q}_{\rho_j}^\top$ with parameter $\eta = 2$: $\boldsymbol{Q}_{\rho_j}^\top [\boldsymbol{S}_1 \ \ \boldsymbol{S}_2] = \boldsymbol{Z}[\boldsymbol{R}_{11} \ \ \boldsymbol{R}_{12}]$
  $\boldsymbol{P}_{\rho_j} = \boldsymbol{S}_1 \in \mathbb{R}^{n_{\rho_j} \times r_{\rho_j}}$ containg the columns from the identity matrix
  $\boldsymbol{U}^{\rho_j} \leftarrow \boldsymbol{Q}_{\rho_j} \left(\boldsymbol{P}_{\rho_j}^\top \boldsymbol{Q}\right)^{-1}$
  $\boldsymbol{C}_{\rho_j} \leftarrow \boldsymbol{P}_{\rho_j}^\top \boldsymbol{C}_{\rho_j}$
$\mathcal{C} \leftarrow \boldsymbol{C}_{\rho_d}$ in tensor format
return $[\![\mathcal{C}; \boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(d)}]\!]$

The SP-STHOSVD algorithm is computationally faster than the previous methods, especially for sparse tensors.

Theorem 2 (SP-STHOSVD [10])**.** *Let $\hat{\mathcal{A}} = [\![\mathcal{C}; \boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(d)}]\!]$ be the output of Algorithm 2 with input ranks $r = (r_1, r_2, \ldots, r_d)$ and oversampling parameter $p \geq 2$, such that $r_j + p \leq \min\left(n_j, \prod_{i \neq j} n_i\right)$ for all $j \in [d]$. For any processing order ρ, the expected approximation error satisfies*

$$\mathop{\mathbb{E}}_{\{\Omega_j\}_{j \in [d]}} \left[\|\hat{\mathcal{A}} - \mathcal{A}\|_F\right] \leq \sum_{j=1}^{d} \left(\prod_{k=1}^{j} \sqrt{1 + 4r_j(n_j - r_j)}\right) \left(1 + \frac{r_j}{p-1}\right)^{1/2} \|\mathcal{A} - \hat{\mathcal{A}}_{opt}\|_F. \tag{4}$$

2.1 Numerical Results on the FROSTT Database

The *Formidable Repository of Open Sparse Tensors and Tools* (FROSTT) [15] is a collection of publicly available sparse tensor datasets and tools. [10] consider two representative large and sparse tensor datasets. NELL-2 is a dataset built from the Web via an intelligent agent called Never-Ending Language Learner [3]. It is a three-dimensional dataset whose modes represent entity, relation, and entity respectively. Enron [14] contains word counts in emails released during an investigation by FERC. The modes represent sender, receiver, word, and date.

Original Tensor	Order	Size	Nonzeros
NELL-2	3	$12092 \times 9184 \times 28818$	$76,879,419$
Enron	4	$6066 \times 5699 \times 244268 \times 1176$	$54,202,099$

Condensed Tensor	Order	Size	Nonzeros
NELL-2	3	$807 \times 613 \times 1922$	$19,841$
Enron	3	$405 \times 380 \times 9771$	$6,131$

Table 2

Summary of sparse tensor examples from the FROSTT database—we include the details for both the full datasets and the condensed datasets used in our experiments.

NELL-2				
	Relative Error		Runtime in seconds	
Target Rank	SP-STHOSVD	R-STHOSVD	SP-STHOSVD	R-STHOSVD
30	0.2968	0.1319	0.5690	17.0642
60	0.2282	0.0914	0.9606	18.9203
90	0.1950	0.0699	1.5889	23.3303
120	0.1666	0.0573	2.0706	28.9399
150	0.1431	0.0478	2.1310	33.6867
180	0.1201	0.0417	2.3678	39.0644
210	0.1181	0.0367	3.0832	45.5227
240	0.1095	0.0326	3.7282	52.4856

Enron				
	Relative Error		Runtime in seconds	
Target Rank	SP-STHOSVD	R-STHOSVD	SP-STHOSVD	R-STHOSVD
20	0.6015	0.2081	0.4086	31.5615
45	0.3854	0.1259	0.7965	34.5802
70	0.3548	0.0870	1.3276	36.6431
95	0.2038	0.0632	2.3465	39.3095
120	0.1503	0.0458	2.8175	39.7169
145	0.0976	0.0332	3.5659	42.0969
170	0.0756	0.0239	6.2158	45.8429
195	0.0578	0.0180	6.8285	50.2907

Table 6

The relative error and runtime of both SP-STHOSVD and R-STHOSVD on both the condensed and sub-sampled Enron dataset and the condensed NELL-2 dataset as the target rank (r, r, r) increases. The processing order was $\rho = [3, 1, 2]$, and the oversampling parameter was $p = 5$. Note that the rank is the same for each mode for simplicity, and that the input rank for the R-STHOSVD was $(r+p, r+p, r+p)$ so the approximations have the same size.

3 Sketching in the Tensor World

To motivate this section, recall the CountSketch approach for sketching, introduced in [5] to estimate the frequency of items in a stream. The sampling matrix $\boldsymbol{S}$ is of the form

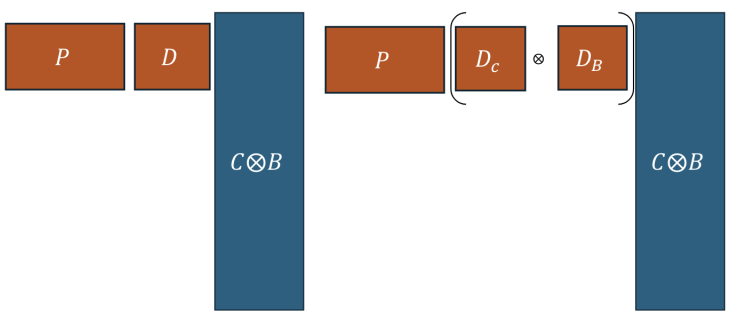
$$\boldsymbol{S} = \begin{bmatrix} 0 & -1 & 0 & 0 & \dots & 0 \\ +1 & 0 & 0 & +1 & \dots & 0 \\ 0 & 0 & -1 & 0 & \iddots & 0 \\ 0 & 0 & 0 & 0 & \dots & +1 \end{bmatrix},$$

where each column has exactly one non-zero entry (and these non-zero entries are iid Rademachers). Each ± 1 entry in the ith row of $\boldsymbol{S}$ contributes $\pm \boldsymbol{A}_{i*}$ to one of the rows of $\boldsymbol{SA}$. Suppose now that the matrix $\boldsymbol{A}$ is a Kronecker product or a Khatri-Rao product of two smaller matrices; this is common in many applications such as compressed matrix multiplication and efficient approximation of SVM polynomial kernels. To this end, efficient sketching methods have been proposed such as the TensorSketch [13] and GaussianSketch [2].

The CountSketch algorithm defines the sampling matrix $\boldsymbol{S} = \boldsymbol{PD} \in \mathbb{R}^{m \times N}$, where the columns of $\boldsymbol{P}$ are iid and of the m canonical basis vectors in $\mathbb{R}^m$ (uniformly at random) and $\boldsymbol{D}$ is a diagonal matrix with iid Rademacher entries. The TensorSketch algorithm, introduced in [13], uses a sketching matrix of the form

$$\boldsymbol{S} = \boldsymbol{P}\left(\boldsymbol{D}_C \otimes \boldsymbol{D}_B\right), \text{ where } \boldsymbol{D}_C \in \mathbb{R}^{n_3 \times n_3} \text{ and } \boldsymbol{D}_B \in \mathbb{R}^{n_2 \times n_2},$$

where $\boldsymbol{D}_B$ and $\boldsymbol{D}_C$ are diagonal matrices with iid Rademacher entries (but with a much smaller dimension).



If $\boldsymbol{C} \in \mathbb{R}^{n_3 \times m_3}$ and $\boldsymbol{B} \in \mathbb{R}^{n_2 \times m_2}$ then the following identities hold:

$$\boldsymbol{S}\left(\boldsymbol{C} \odot \boldsymbol{B}\right) = \mathsf{FFT}^{-1}\left(\mathsf{FFT}\left(\boldsymbol{S}_C \boldsymbol{C}\right) * \mathsf{FFT}\left(\boldsymbol{S}_B \boldsymbol{B}\right)\right) \tag{5}$$

$$\boldsymbol{S}\left(\boldsymbol{C} \otimes \boldsymbol{B}\right) = \mathsf{FFT}^{-1}\left(\left(\mathsf{FFT}\left(\boldsymbol{S}_C \boldsymbol{C}\right)^\top \odot \mathsf{FFT}\left(\boldsymbol{S}_B \boldsymbol{B}\right)^\top\right)^\top\right) \tag{6}$$

TensorSketch sketches provide approximate matrix multiplication (AMM) and oblivious subspace embedding guarantees similar to CountSketch:

Theorem 3 (TensorSketch [1])**.** *Let $\boldsymbol{S} \in \mathbb{R}^{m \times n^q}$ be a TensorSketch matrix, and let $\varepsilon, \delta \in (0, 1)$ be parameters. Then, $\boldsymbol{S}$ satisfies the following:*

- *(AMM) Let* $\boldsymbol{A} \in \mathbb{R}^{n^q \times d}$ *and* $\boldsymbol{B} \in \mathbb{R}^{d \times n^q}$*. If* $m \geq \frac{2+3^q}{\varepsilon^2 \delta}$*, then with probability at least* $1 - \delta$*,*

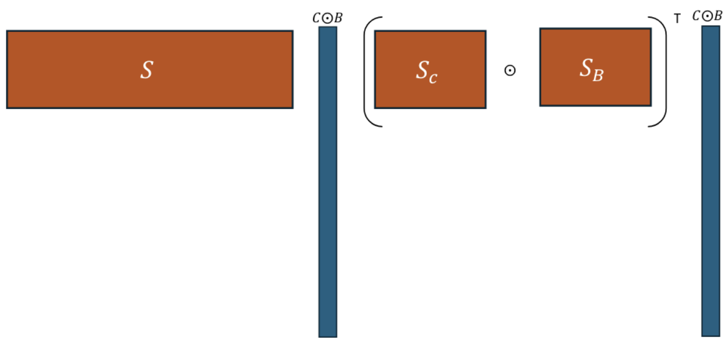
$$\|\boldsymbol{B}\boldsymbol{S}^\top \boldsymbol{S}\boldsymbol{A} - \boldsymbol{B}\boldsymbol{A}\|_F \leq \epsilon \|\boldsymbol{B}\boldsymbol{A}\|_F.$$

- *(Oblivious Subspace Embedding) Let* $\boldsymbol{U}$ *be some fixed* r*-dimensional subspace. If* $m \geq \frac{r^2(2+3^q)}{\varepsilon^2 \delta}$*, then with probability at least* $1 - \delta$*,*

$$\|\boldsymbol{U}^\top \boldsymbol{S}^\top \boldsymbol{S}\boldsymbol{U} - \boldsymbol{I}\| \leq \varepsilon.$$

Remark 2. There are other such extensions of matrix sketches to the tensor setting; one such sketch is the *structured Gaussian sketch.* If $\boldsymbol{C} \in \mathbb{R}^{n_3 \times r}$ and $\boldsymbol{B} \in \mathbb{R}^{n_2 \times r}$, then a the standard Gaussian sketching matrix $\boldsymbol{S}$ designed for $\boldsymbol{C} \odot \boldsymbol{B}$ has dimension $m \times n_2 n_3$. Instead, [2] suggest $\boldsymbol{S}_C \odot \boldsymbol{S}_B$, where $\boldsymbol{S}_C \in \mathbb{R}^{n_3 \times m}$ and $\boldsymbol{S}_B \in \mathbb{R}^{n_2 \times m}$ are Gaussian sketches. Note that

$$\boldsymbol{S}^\top (\boldsymbol{C} \odot \boldsymbol{B}) = (\boldsymbol{S}_C \odot \boldsymbol{S}_B)^\top (\boldsymbol{C} \odot \boldsymbol{B}) = \left(\boldsymbol{S}_C^\top \boldsymbol{C}\right) * \left(\boldsymbol{S}_B^\top \boldsymbol{B}\right).$$



Remark 3 (Tensor-TS)**.** In a seminal work on low-rank Tucker decompositions, [8] propose an algorithm that uses TensorSketch [12, 13].

Algorithm 5: TUCKER-TS [8]

Data: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, target rank vector $r \in \mathbb{N}^d$, sketch dimensions m_1, m_2

Initialize $\mathcal{C}, \boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \ldots, \boldsymbol{U}^{(d)}$

Define TENSORSKETCH operators $\boldsymbol{T}^{(i)} \in \mathbb{R}^{m_1 \times \prod_{j \neq i} n_j}$ for $i \in [d]$ and $\boldsymbol{T}^{(d+1)} \in \mathbb{R}^{m_2 \times \prod_j n_j}$

while *termination criteria is not met* **do**

    for $j = 1, 2, \ldots, d$ **do**

        $\boldsymbol{U}^{(j)} \leftarrow \arg\min_{\boldsymbol{U}} \|\left(\boldsymbol{T}^{(j)} \otimes_{j \neq i}^{1} \boldsymbol{U}^{(i)}\right) \boldsymbol{C}_i^\top \boldsymbol{U}^\top - \boldsymbol{T}^{(j)} \boldsymbol{A}_i^\top \|_F^2$

    $\mathcal{C} \leftarrow \arg\min_{\mathcal{Z}} \|\left(\boldsymbol{T}^{(d+1)} \otimes_{j=N}^{1} \boldsymbol{U}^{(j)}\right) \mathrm{vec}(\mathcal{Z}) - \boldsymbol{T}^{(d+1)} \mathrm{vec}(\mathcal{A})\|_F^2$

    Orthogonalize each $\boldsymbol{U}^{(j)}$ and update $\mathcal{C}$

return $[\![\mathcal{C}; \boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(d)}]\!]$

---

4 Tensor Train Decomposition

Recall that the canonical decomposition of a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is given by [4, 7]:

$$\mathcal{A}_{i_1,i_2,\ldots,i_d} = \sum_{\alpha=1}^{r} \boldsymbol{U}_1(i_1,\alpha)\boldsymbol{U}_2(i_2,\alpha)\ldots\boldsymbol{U}_d(i_d,\alpha).$$

The smallest r for which such a decomposition exists is called the rank of $\mathcal{A}$, and $\boldsymbol{U}_k$ are called the canonical factors. Unfortunately, computing r and these factors is NP-hard.

[11] suggests a different decomposition method. To demonstrate this, consider the unfolding of a 6-dimensional tensor:

$$\mathcal{A}(i_1 i_2; i_3 i_4 i_5 i_6) = \sum_{\alpha_2} \mathcal{U}(i_1, i_2; \alpha_2)\mathcal{V}(i_3, i_4, i_5, i_6; \alpha_2).$$

If we are provided with some a-priori knowledge about near-eparability of the variables, the dimension can be reduced (here, we have decomposed $\mathcal{A}$ into a sum of product of a 3-dimensional tensor and a 5-dimensional tensor). This process can be repeated for these tensors, leading to the *tensor train decomposition.*

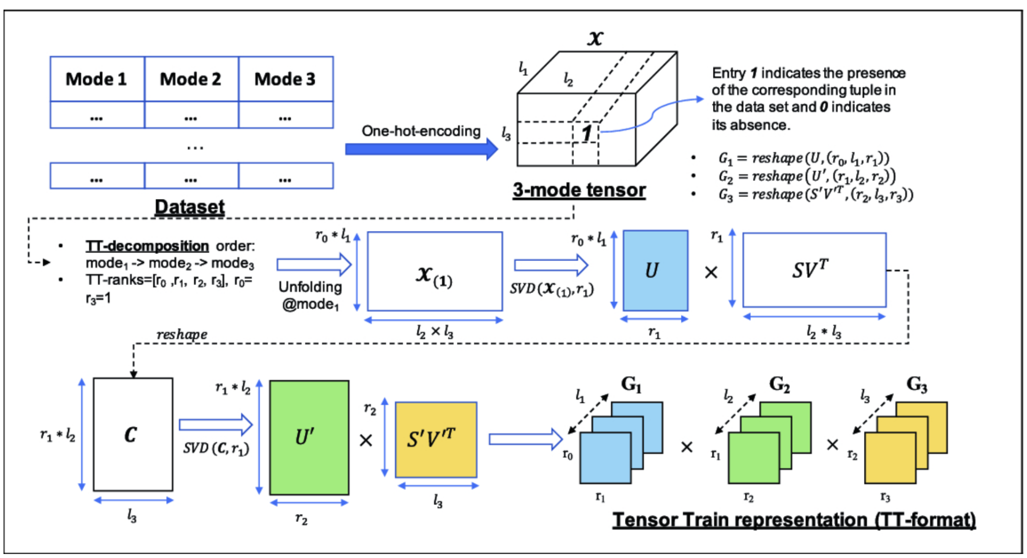
The TT-decomposition of a tensor $\mathcal{A}$ is of the form

$$\mathcal{A}(i_1, i_2, \ldots, i_d) = \sum_{\alpha_0,\alpha_1,\ldots,\alpha_d} \mathcal{C}_1(\alpha_0, i_1, \alpha_1)\mathcal{C}_2(\alpha_1, i_2, \alpha_2)\ldots\mathcal{C}_d(\alpha_{d-1}, i_d, \alpha_d),$$

which can be represented compactly as a matrix product

$$\mathcal{A}(i_1, i_2, \ldots, i_d) = \underbrace{\mathcal{C}_1[i_1]}_{1\times r_1}\underbrace{\mathcal{C}_2[i_2]}_{r_1\times r_2}\ldots\underbrace{\mathcal{C}_d[i_d]}_{r_d\times 1}.$$

The tensors $\mathcal{C}_i$ are called the *TT-cores*, and the ranks r_i are called *TT-ranks*. If $r := \max_i r_i$ is the maximum TT-rank, then TT uses $O\left(ndr^2\right)$ memory to store the $O(nd)$ elements. Therefore, it is efficient if the ranks are small.



References

[1] Haim Avron, Huy Nguyen, and David Woodruff. Subspace embeddings for the polynomial kernel. *Advances in neural information processing systems*, 27, 2014.

[2] David J Biagioni, Daniel Beylkin, and Gregory Beylkin. Randomized interpolative decomposition of separated representations. *Journal of Computational Physics*, 281:116–134, 2015.

[3] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam Hruschka, and Tom Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, pages 1306–1313, 2010.

[4] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. *Psychometrika*, 35(3):283–319, 1970.

[5] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 693–703. Springer, 2002.

[6] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[7] Richard A Harshman et al. Foundations of the parafac procedure: Models and conditions for an explanatory multi-modal factor analysis. *UCLA working papers in phonetics*, 16(1):84, 1970.

[8] Osman Asif Malik and Stephen Becker. Low-rank tucker decomposition of large tensors using tensorsketch. *Advances in neural information processing systems*, 31, 2018.

[9] Per-Gunnar Martinsson and Sergey Voronin. A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices. *SIAM Journal on Scientific Computing*, 38(5):S485–S507, 2016.

[10] Rachel Minster, Arvind K Saibaba, and Misha E Kilmer. Randomized algorithms for low-rank tensor decompositions in the tucker format. *SIAM journal on mathematics of data science*, 2(1):189–215, 2020.

[11] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[12] Rasmus Pagh. Compressed matrix multiplication. *ACM Transactions on Computation Theory (TOCT)*, 5(3):1–17, 2013.

[13] Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–247, 2013.

[14] Jitesh Shetty and Jafar Adibi. The enron email dataset database schema and brief statistical report (2004). *Available on< http://www. isi. edu/˜ adibi/Enron/Enron_Dataset_Report. pdf.*

[15] Shaden Smith, Jee W. Choi, Jiajia Li, Richard Vuduc, Jongsoo Park, Xing Liu, and George Karypis. FROSTT: The formidable repository of open sparse tensors and tools, 2017. URL: http://frostt.io/.

[16] Wenjian Yu, Yu Gu, and Yaohang Li. Efficient randomized algorithms for the fixed-precision low-rank matrix approximation. *SIAM Journal on Matrix Analysis and Applications*, 39(3):1339–1359, 2018.

[17] Guoxu Zhou, Andrzej Cichocki, and Shengli Xie. Decomposition of big tensors with low multilinear rank. *arXiv preprint arXiv:1412.1885*, 2014.