CSE 392/CS 395T/M 397C: Matrix and Tensor Algorithms for Data

Instructor: Shashanka Ubaru

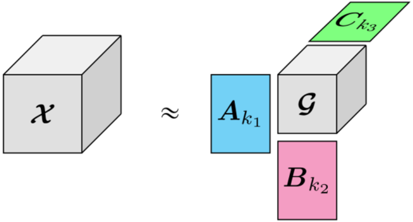
University of Texas, Austin
Spring 2025

Lecture 20: Randomized Tucker, TensorSketch

Outline

1. Randomized Tucker
2. TensorSketch
3. Tensor Train Decomposition

Recall: Tucker Decomposition



- 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 tensor 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 factor matrices.
- Elementwise:

$$x_{ij\ell} \approx \sum_{q=1}^{k_1} \sum_{r=1}^{k_2} \sum_{s=1}^{k_3} g_{qrs} d_{iq} b_{jr} c_{\ell s} \text{ for } i \in [m], j \in [n], \ell \in [p]$$

HOSVD Algorithm

Inputs: Tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, ranks $\{r_1, \ldots, r_d\} \in \mathbb{N}$.

1. **for** $\ell = 1, \ldots, d$ **do**
2. $\quad \boldsymbol{U}^{(\ell)} \leftarrow r_\ell$ leading left singular vectors of $\boldsymbol{A}_{(\ell)}$
3. **end for**
4. $\mathcal{G} = \mathcal{A} \times_1 \boldsymbol{U}^{(1)\top} \times_2 \boldsymbol{U}^{(2)\top} \cdots \times_d \boldsymbol{U}^{(d)\top}$
5. **return** $\mathcal{G}, \boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \cdots, \boldsymbol{U}^{(d)}$

HOOI Algorithm

Inputs: Tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, ranks $\{r_1, \ldots, r_d\} \in \mathbb{N}$.

1. Initialize $\boldsymbol{U}^{(\ell)} \in \mathbb{R}^{n_\ell \times r_\ell}$ for all $\ell \in [d]$
2. **repeat**
3. **for** $\ell = 1, \ldots, d$ **do**
4. $\mathcal{Y} = \mathcal{A} \times_1 \boldsymbol{U}^{(1)\top} \cdots \times_{\ell-1} \boldsymbol{U}^{(\ell-1)\top} \times_{\ell+1} \boldsymbol{U}^{(\ell+1)\top} \cdots \times_d \boldsymbol{U}^{(d)\top}$
5. $\boldsymbol{U}^{(\ell)} \leftarrow r_\ell$ leading left singular vectors of $\boldsymbol{Y}_{(\ell)}$
6. **end for**
7. **until** fit ceases to improve or maximum iterations exhausted
8. $\mathcal{G} = \mathcal{A} \times_1 \boldsymbol{U}^{(1)\top} \times_2 \boldsymbol{U}^{(2)\top} \cdots \times_d \boldsymbol{U}^{(d)\top}$
9. **return** $\mathcal{G}, \boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \cdots, \boldsymbol{U}^{(d)}$

Sequential Truncated HOSVD (ST-HOSVD)

- Choose an ordering in which to visit the modes
- Once left singular vectors for a mode are computed, immediately project. Then only operate on the projected core result
- Example (ordering 1,2,3 and truncation (k_1, k_2, k_3)):
  - Compute $\boldsymbol{U}^{(1)}$ from SVD of $\mathcal{A}_{(1)}$
  - Compute $\boldsymbol{U}^{(2)}$ from SVD of $\widehat{\mathcal{C}} := \mathcal{A} \times_1 (\boldsymbol{U}^{(1)})^\top$
  - Compute $\boldsymbol{U}^{(3)}$ from SVD of $\tilde{\mathcal{C}} := \widehat{\mathcal{C}} \times_2 (\boldsymbol{U}^{(2)})^\top$
  - $\mathcal{C} = \tilde{\mathcal{C}} \times_3 (\boldsymbol{U}^{(3)})^\top$
- Now let $\mathcal{A} \approx [\mathcal{C}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}]$. Worst case error bound is same as for tr-HOSVD.
- Computing on successively smaller objects, more efficient; often near-comparable, or better, behavior than tr-HOSVD!

Towards Randomized Tucker

Both HOSVD and STHOSVD rely on matrix SVDs. The obvious first choice for randomization - use calls to matrix routine randSVD.

Matrix RandSVD: Given $\mathbf{X} \in \mathbb{R}^{m \times n}$, target rank r, oversampling parameter $p \geq 0$ such that $r + p \leq \min\{m, n\}$,

- Draw Gaussian random matrix $\Omega \in \mathbb{R}^{n \times (r+p)}$
- Multiply $\mathbf{Y} \leftarrow \mathbf{X}\Omega$
- Thin QR factorization $\mathbf{Y} = \mathbf{QR}$
- Form $\mathbf{B} \leftarrow \mathbf{Q}^\top \mathbf{X}$
- Calculate thin SVD $\mathbf{B} = \widehat{\mathbf{U}}_{\mathbf{B}} \widehat{\mathbf{S}} \widehat{\mathbf{V}}^\top$
- Form $\widehat{\mathbf{U}} \leftarrow \mathbf{Q}(\widehat{\mathbf{U}}_{\mathbf{B}})_{:,1:r}$
- Compress $\widehat{\mathbf{S}} \leftarrow \widehat{\mathbf{S}}_{1:r,1:r}$, and $\widehat{\mathbf{V}} \leftarrow \widehat{\mathbf{V}}_{:,1:r}$, so $\mathbf{X} = \widehat{\mathbf{U}}\widehat{\mathbf{S}}\widehat{\mathbf{V}}$

Halko, Martinsson, Tropp, SIREV, 2011

Towards Randomized Tucker

Both HOSVD and STHOSVD rely on matrix SVDs. The obvious first choice for randomization - use calls to matrix routine randSVD.

Given: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$; target rank vector $\mathbf{r} \in \mathbb{N}^d$; oversampling parameter $p \geq 0$

$j = 1 : d$

- Draw random Gaussian matrix $\mathbf{\Omega}_j \in \mathbb{R}^{\prod_{i \neq j} n_i \times (r_j + p)}$
- $[\widehat{\mathbf{U}}, \widehat{\mathbf{\Sigma}}, \widehat{\mathbf{V}}] = \text{RandSVD}(\mathcal{A}_{(j)}, r_j, p, \mathbf{\Omega}_j)$
- Set $\mathbf{U}^{(j)} \leftarrow \widehat{\mathbf{U}}$

Form $\mathcal{C} = \mathcal{A} \times_{j=1}^{d} (\mathbf{U}^{(j)})^\top$

G. Zhou, A. Cichocki, and S. Xie, Decomposition of Big Tensors with Low Multilinear Rank, arXiv 1412.1885, 2014

Result[1]

The output (with minor assumptions) satisfies

Theorem (Randomized HOSVD)

$$\mathbb{E}_{\{\mathbf{\Omega}_k\}_{k=1}^d} \|\mathcal{A} - \widehat{\mathcal{A}}\|_F \leq \left(d + \frac{\sum_{j=1}^d r_j}{p-1}\right)^{1/2} \|\mathcal{A} - \widehat{\mathcal{A}}_{opt}\|_F.$$

special cases: Let $r = \max_{1\leq j\leq d} r_j$. Then, if $p = r+1$,

$$\mathbb{E}_{\{\mathbf{\Omega}_k\}_{k=1}^d} \|\mathcal{A} - \widehat{\mathcal{A}}\|_F \leq \sqrt{2}\|\mathcal{A} - \widehat{\mathcal{A}}_{\text{HOSVD}}\|_F \leq \sqrt{2d}\|\mathcal{A} - \widehat{\mathcal{A}}_{opt}\|_F.$$

If $p = \lceil \frac{r}{\epsilon} \rceil + 1$ for some $\epsilon > 0$,

$$\mathbb{E}_{\{\mathbf{\Omega}_k\}_{k=1}^d} \|\mathcal{A} - \widehat{\mathcal{A}}\|_F \leq \sqrt{1+\epsilon}\|\mathcal{A} - \widehat{\mathcal{A}}_{\text{HOSVD}}\|_F \leq \sqrt{d(1+\epsilon)}\|\mathcal{A} - \widehat{\mathcal{A}}_{opt}\|_F.$$

[1] Minster, Saibaba, Kilmer, SIMODS, 2020

Randomized Sequentially Truncated-HOSVD (r-STHOSVD)

- Similar structure, except feed current intermediate (mode-wise unfolded) core tensor.
- Complication for the theoretical result: at each intermediate step, the partially truncated core tensor is a random tensor.
- The theoretical result gives the same upper bound (fix the processing order; but ultimately independent of this)!
- Again, we see that in the worst case, the randomized versions of either algorithm can have the same performance.
- The latter is cheaper, and in practice can perform better. Use processing order that makes it cheapest.

Dynamic Randomized HOSVD and STHOSVD [3]

If target rank is unknown, how to find $\widehat{\mathcal{A}}$ such that

$$\|\mathcal{A} - \widehat{\mathcal{A}}\|_F \leq \epsilon \|\mathcal{A}\|_F?$$

Utilize matrix adaptive randomized range finders[2] for interim calculations. Given a matrix $\mathbf{A}$ and a tolerance $\varepsilon > 0$, the goal is to find a matrix $\mathbf{Q}$ with orthonormal columns that satisfies

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^\top \mathbf{A}\| \leq \varepsilon \|\mathbf{A}\|.$$

No. columns in $\mathbf{Q}$ is assumed rank of the low rank approximation.
Example: equally apportioned per mode: choose the factor matrices $\mathbf{U}_j$ to satisfy

$$\|\mathcal{A}_{(j)} - \mathbf{U}^{(j)}(\mathbf{U}^{(j)})^\top \mathcal{A}_{(j)}\|_F = \|\mathcal{A} \times_j (\mathbf{I} - \mathbf{U}^{(j)}(\mathbf{U}^{(j)})^\top)\|_F \leq \frac{\varepsilon}{\sqrt{d}} \|\mathcal{A}\|_F.$$

[2] See W. Yu, Y. Gu, and Y. Li, SIMAX, 2018 and references therein.
[3] Minster, Saibaba, Kilmer, SIMODS, 2020

Preserving Structure?

We know that the (intermediate) core can be dense in any of the above mentioned processes.

What if $\mathcal{A}$ is sparse, or has other structure (e.g. non-negativity)?

Can we compute a factorization such that the core tensor inherits properties of $\mathcal{A}$?

Yes!

Process for a matrix $\mathbf{X}$

Instead of using $\mathbf{X} \approx \mathbf{Q}\mathbf{Q}^\top \boldsymbol{X}$ where $\mathbf{QR} = (\boldsymbol{X}\Omega_{r+p})$ do:

Compute strong RRQR of $\boldsymbol{Q}^\top$

$$\mathbf{Q}^\top \boldsymbol{S} = \boldsymbol{Z}\mathrm{N},$$

then $\mathbf{P} = \boldsymbol{S}_{:,1:s}$, so $\mathbf{P}^\top \boldsymbol{Q}$ well-conditioned rows of $\mathbf{Q}$.

Define the oblique projector:

$$\mathbf{Q}(\mathbf{P}^\top \boldsymbol{Q})^{-1} \boldsymbol{P}^\top$$

and apply this to $\boldsymbol{X}$.

$$\boldsymbol{X} \approx (\mathbf{Q}(\mathbf{P}^\top \mathbf{Q})^{-1}) \underbrace{\mathbf{P}^\top \boldsymbol{X}}_{\widehat{\boldsymbol{X}}}.$$

Note that $\widehat{\boldsymbol{X}}$ is subselected rows of $\boldsymbol{X}$.

Structure Preserving STHOSVD

Idea:

- pick an order to visit
- apply the previous idea to the current unfolded core
- update the current core (it will have subselected rows), and the resulting factor matrix is the left matrix product.

The final core will have multirank $(r_1 + p, r_2 + p, \ldots, r_d + p)$ and contain portions of the original tensor. The factor matrices are not orthogonal.

Randomized Variants that Handle Sparsity

Formidable Repository of Sparse Tensors and Tools database.

Tensor	Dimensions	Nonzeros
NELL-2	$12092 \times 9184 \times 28818$	76,879,419
Enron	$6066 \times 5699 \times 244268 \times 1176$	54,202,099

NELL-2: entity $\times$ relation $\times$ entity (NELL is a machine learning system that relates different categories)

Enron: sender $\times$ receiver $\times$ word $\times$ date (word counts in emails released during an investigation by the FERC)

Approximate truncated (r, r, r) HOSVD and ST-HOSVD

Results

	Relative Error		Runtime in seconds	
r	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
145	0.0976	0.0332	3.5659	42.0969
195	0.0578	0.0180	6.8285	50.2907

Table: Results, Subsampled Enron dataset.

Taking advantage of the sparsity structure allows for faster compression[4].

[4] R. Minster, A.K. Saibaba, and M. E. Kilmer, "Randomized Algorithms for low-rank Decompositions in the Tucker Format," SIMODS, 2020.

Recall: TUCKER-ALS

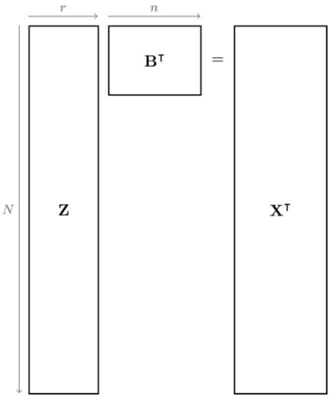
- Minimize the objective function:

$$F(\mathcal{G}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = \|\mathcal{X} - [\![\mathcal{G}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]\|_F^2$$

- The canonical TUCKER-ALS - repeatedly solve until convergence:
  - $\boldsymbol{A}_{t+1} = \arg\min_{\boldsymbol{A}} F\left(\mathcal{G}_t, \boldsymbol{A}, \boldsymbol{B}_t, \boldsymbol{C}_t\right) = \arg\min_{\boldsymbol{A}} \left\|\left(\boldsymbol{C}_t \otimes \boldsymbol{B}_t\right) \boldsymbol{G}_{(1),t}^\top \boldsymbol{A}^\top - \boldsymbol{X}_{(1)}^\top\right\|_F^2$
  - $\boldsymbol{B}_{t+1} = \arg\min_{\boldsymbol{B}} F\left(\mathcal{G}_t, \boldsymbol{A}_{t+1}, \boldsymbol{B}, \boldsymbol{C}_t\right) = \arg\min_{\boldsymbol{B}} \left\|\left(\boldsymbol{C}_t \otimes \boldsymbol{A}_{t+1}\right) \boldsymbol{G}_{(2),t}^\top \boldsymbol{B}^\top - \boldsymbol{X}_{(2)}^\top\right\|_F^2$
  - $\boldsymbol{C}_{t+1} = \arg\min_{\boldsymbol{C}} F\left(\mathcal{G}_t, \boldsymbol{A}_{t+1}, \boldsymbol{B}_{t+1}, \boldsymbol{C}\right) = \arg\min_{\boldsymbol{C}} \left\|\left(\boldsymbol{B}_{t+1} \otimes \boldsymbol{A}_{t+1}\right) \boldsymbol{G}_{(3),t}^\top \boldsymbol{C}^\top - \boldsymbol{X}_{(3)}^\top\right\|_F^2$
  - $\mathcal{G}_{t+1} = \arg\min_{\mathcal{G}} \left\|(\boldsymbol{C}_{t+1} \otimes \boldsymbol{B}_{t+1} \otimes \boldsymbol{A}_{t+1})\boldsymbol{g}_{(:)} - \boldsymbol{x}_{(:)}\right\|_2^2$

Recall: Kronecker FJLTs

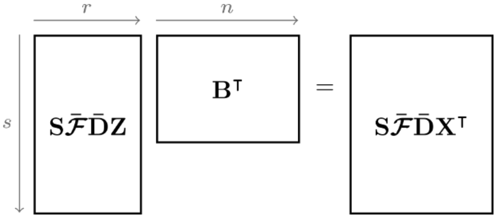
$$\min_{\boldsymbol{B}} \|\boldsymbol{Z}\boldsymbol{B}^\top - \boldsymbol{X}^\top\|_F^2$$



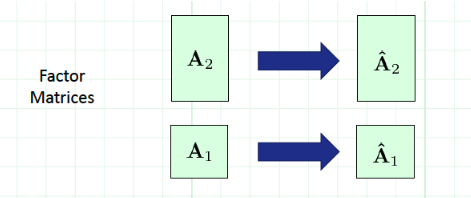
$$\boldsymbol{Z} = \boldsymbol{A}_d \odot \cdots \odot \boldsymbol{A}_{k+1} \odot \boldsymbol{A}_{k-1} \odot \cdots \odot \boldsymbol{A}_1$$

$$\min_{\boldsymbol{B}} \|\boldsymbol{S}\bar{\mathcal{F}}\bar{\boldsymbol{D}}\boldsymbol{Z}\boldsymbol{B}^\top - \boldsymbol{S}\bar{\mathcal{F}}\bar{\boldsymbol{D}}\boldsymbol{X}^\top\|_F^2$$

- $\boldsymbol{S}$ is $s \times N$ sampling matrix
- $\bar{\mathcal{F}} = \mathcal{F}_d \otimes \cdots \otimes \mathcal{F}_{k+1} \otimes \mathcal{F}_{k-1} \otimes \cdots \otimes \mathcal{F}_1$.
- $\bar{\boldsymbol{D}} = \boldsymbol{D}_d \otimes \cdots \otimes \boldsymbol{D}_{k+1} \otimes \boldsymbol{D}_{k-1} \otimes \cdots \otimes \boldsymbol{D}_1$.



Mixing Kronecker product Efficiently Using KFJLT



$$\begin{aligned}
\boldsymbol{S}\bar{\mathcal{F}}\bar{\boldsymbol{D}}\boldsymbol{Z} &= \boldsymbol{S}(\mathcal{F}_2 \otimes \mathcal{F}_1)(\boldsymbol{D}_2 \otimes \boldsymbol{D}_1)(\boldsymbol{A}_2 \odot \boldsymbol{A}_1) \\
&= \boldsymbol{S}\left((\mathcal{F}_2\boldsymbol{D}_2) \otimes (\mathcal{F}_1\boldsymbol{D}_1)\right)(\boldsymbol{A}_2 \odot \boldsymbol{A}_1) \\
&= \boldsymbol{S}\left((\mathcal{F}_2\boldsymbol{D}_2\boldsymbol{A}_2) \odot (\mathcal{F}_1\boldsymbol{D}_1\boldsymbol{A}_1)\right) \\
&= \boldsymbol{S}(\hat{\boldsymbol{A}}_2 \odot \hat{\boldsymbol{A}}_1)
\end{aligned}$$

Same approach holds for Kronecker products too:

$$\begin{aligned}
\boldsymbol{S}\bar{\mathcal{F}}\bar{\boldsymbol{D}}\boldsymbol{Z} &= \boldsymbol{S}(\mathcal{F}_2 \otimes \mathcal{F}_1)(\boldsymbol{D}_2 \otimes \boldsymbol{D}_1)(\boldsymbol{A}_2 \otimes \boldsymbol{A}_1) \\
&= \boldsymbol{S}\left((\mathcal{F}_2\boldsymbol{D}_2) \otimes (\mathcal{F}_1\boldsymbol{D}_1)\right)(\boldsymbol{A}_2 \otimes \boldsymbol{A}_1) \\
&= \boldsymbol{S}\left((\mathcal{F}_2\boldsymbol{D}_2\boldsymbol{A}_2) \otimes (\mathcal{F}_1\boldsymbol{D}_1\boldsymbol{A}_1)\right) \\
&= \boldsymbol{S}(\hat{\boldsymbol{A}}_2 \otimes \hat{\boldsymbol{A}}_1)
\end{aligned}$$

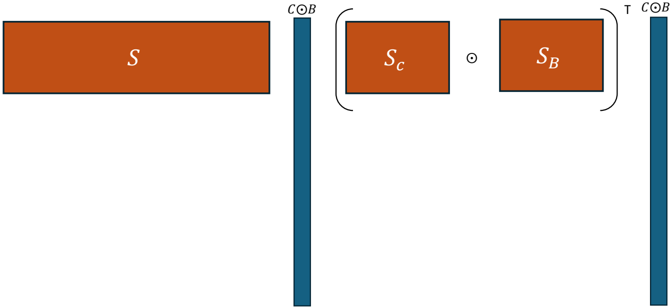
Structured Gaussian sketch

Standard Gaussian sketch: $\boldsymbol{S} \in \mathbb{R}^{m \times n_2 n_3}$
All entries are i.i.d Gaussian, $\boldsymbol{S}_{ij} \sim \mathbb{N}(0, 1)$.

Structured Gaussian sketch [BBB15]:
$\boldsymbol{S} = (\boldsymbol{S}_C \odot \boldsymbol{S}_B)^\top$, where $\boldsymbol{S}_C \in \mathbb{R}^{n_3 \times m}$ and $\boldsymbol{S}_B \in \mathbb{R}^{n_2 \times m}$. Then,

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

Efficient! Can prove JL-type results.



TensorSketch

- **TensorSketch:** Structured CountSketch.
- *CountSketch:* $\boldsymbol{S}$ is of the form:

$$\begin{bmatrix} 0 & -1 & 0 & 0 & \cdots & 0 \\ +1 & 0 & 0 & +1 & \cdots & 0 \\ 0 & 0 & -1 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \cdots & -1 \end{bmatrix}$$

  One random ± 1 per column. Row $\boldsymbol{A}_{i*}$ of $\boldsymbol{A}$ contributes $\pm \boldsymbol{A}_{i*}$ to one of the rows of $\boldsymbol{S}\boldsymbol{A}$.
- TensorSketch was first introduced in [Pham & Pagh, 2013] for compressed matrix multiplication and approximating SVM polynomial kernels efficiently.
- Avron et al. show that TensorSketch provides an oblivious subspace embedding.

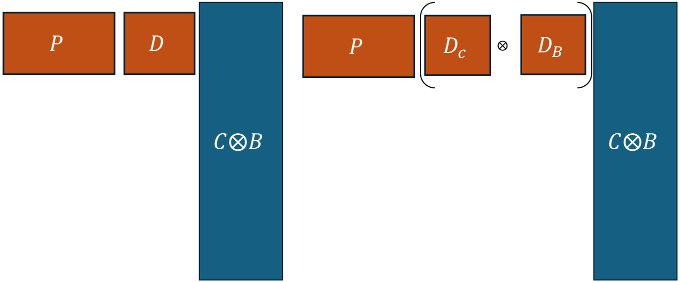
TensorSketch: Structured CountSketch

Countsketch: $\boldsymbol{S} = \boldsymbol{P}\boldsymbol{D} \in \mathbb{R}^{m \times N}$ where $\boldsymbol{P}$ has 1 nonzero per column $\boldsymbol{P}_{:,j} \sim Unif(\boldsymbol{e}_1, \dots, \boldsymbol{e}_N)$.
and $\boldsymbol{D}$ is diagonal with i.i.d ± 1 entries.

TensorSketch:
$\boldsymbol{S} = \boldsymbol{P}(\boldsymbol{D}_C \otimes \boldsymbol{D}_B)$, where $\boldsymbol{D}_C \in \mathbb{R}^{n_3 \times n_3}$ and $\boldsymbol{D}_B \in \mathbb{R}^{n_2 \times n_2}$. Then,

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

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



TuckerTS Algorithm

Algorithm 2: TUCKER-TS (proposal)

input : $\mathcal{Y}$, target rank $(R_1, R_2, \ldots, R_N)$, sketch dimensions (J_1, J_2)

output : Rank-$(R_1, R_2, \ldots, R_N)$ Tucker decomposition $[\![\mathcal{G}; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!]$ of $\mathcal{Y}$

Initialize $\mathcal{G}$, $\mathbf{A}^{(2)}, \mathbf{A}^{(3)}, \ldots, \mathbf{A}^{(N)}$

Define TensorSketch operators $\mathbf{T}^{(n)} \in \mathbb{R}^{J_1 \times \prod_{i \neq n} I_i}$, for $n \in [N]$, and $\mathbf{T}^{(N+1)} \in \mathbb{R}^{J_2 \times \prod_i I_i}$

repeat

    for $n = 1, \ldots, N$ **do**

        $\mathbf{A}^{(n)} = \arg\min_{\mathbf{A}} \left\| \left(\mathbf{T}^{(n)} \bigotimes_{i=N, i \neq n}^{1} \mathbf{A}^{(i)} \right) \mathbf{G}_{(n)}^{\top} \mathbf{A}^{\top} - \mathbf{T}^{(n)} \mathbf{Y}_{(n)}^{\top} \right\|_F^2$

    end

    $\mathcal{G} = \arg\min_{\mathcal{Z}} \left\| \left(\mathbf{T}^{(N+1)} \bigotimes_{i=N}^{1} \mathbf{A}^{(i)} \right) \mathbf{z}_{(:)} - \mathbf{T}^{(N+1)} \mathbf{y}_{(:)} \right\|_2^2$

    Orthogonalize each $\mathbf{A}^{(i)}$ and update $\mathcal{G}$

until *termination criteria met*

return $\mathcal{G}$, $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$

$$\mathbf{TA} = \mathbf{T} \bigotimes_{i=N}^{1} \mathbf{A}^{(i)} = \mathrm{FFT}^{-1} \left(\left(\bigodot_{i=N}^{1} \left(\mathrm{FFT} \left(\mathbf{S}^{(i)} \mathbf{A}^{(i)} \right) \right)^{\top} \right)^{\top} \right).$$

TensorSketch: Results

[Avron et al, 2014]

(AMM) Given $\boldsymbol{A} \in \mathbb{R}^{n^q \times d}$; $\boldsymbol{B} \in \mathbb{R}^{d' \times n^q}$; $\epsilon, \delta > 0$.
Let $\boldsymbol{S} \in \mathbb{R}^{m \times n^q}$ be a TensorSketch matrix, and if $m \geq \frac{(2+3^q)}{\epsilon^2 \delta}$, then with probability at least $1 - \delta$:

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

(Subspace embedding) For any fixed r-dimensional subspace $\boldsymbol{U}$, if $m \geq \frac{r^2(2+3^q)}{\epsilon^2 \delta}$, then with probability at least $1 - \delta$:

$$\|\boldsymbol{U}^\top \boldsymbol{S}^\top \boldsymbol{S}\boldsymbol{U} - \boldsymbol{I}\|_2 \leq \epsilon$$

TensorSketch: Results

[Malik & Becker, 2018]

Assume $\boldsymbol{T}^{(q+1)}$ be a TensorSketch matrix as in TuckerTS algorithm, and let $\boldsymbol{M} := \left(\boldsymbol{T}^{(q+1)} \otimes_{i=q}^{1} \boldsymbol{A}^{(i)}\right)^{\top} \left(\boldsymbol{T}^{(q+1)} \otimes_{i=q}^{1} \boldsymbol{A}^{(i)}\right)$, where all $\boldsymbol{A}^{(i)}$ have orthonormal columns. If $m \geq \frac{(\prod_i r_i)^2(2+3^q)}{\epsilon^2\delta}$, then with probability at least $1-\delta$:

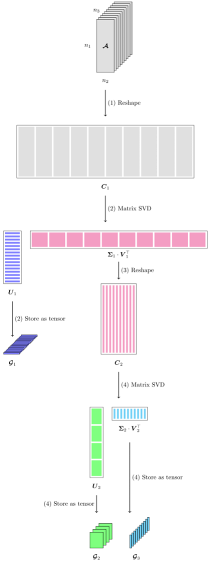
$$\|\boldsymbol{M} - \boldsymbol{I}\|_2 \leq \epsilon$$

Tensor Train Decomposition

- Based on the notion of variable splitting, consider an unfolding $\boldsymbol{A}$ of a tensor $\mathcal{A}$

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

- Provided a-priori knowledge as for separability of variables, the dimension has reduced (e.g. 6-dimensional tensor is decomposed into a product of 3- and 5-dimensional tensors)
- The process can obviously be repeated recursively leading to the Tensor Train decomposition



Tensor Train Decomposition

TT format of a tensor $\mathcal{A}$

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

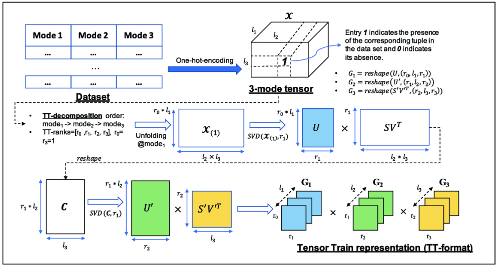
Can be represented compactly as a matrix product:

$$\mathcal{A}(i_1, \ldots, i_d) = \underbrace{\mathcal{G}_1[i_1]}_{1 \times r_1} \underbrace{\mathcal{G}_2[i_2]}_{r_1 \times r_2} \ldots \underbrace{\mathcal{G}_d[i_d]}_{r_{d-1} \times 1}$$

- $\mathcal{G}_i$: TT-cores (collections of matrices)
- r_i: TT-ranks
- $r = \max r_i$: the maximal TT-rank

TT uses $\mathcal{O}(dnr^2)$ memory to store $\mathcal{O}(nd)$ elements

Efficient only if the ranks are small



Oseledets, Tensor-train decomposition, 2011

References:

- Biagioni, David J., Daniel Beylkin, and Gregory Beylkin. "Randomized interpolative decomposition of separated representations." Journal of Computational Physics 281 (2015): 116-134.
- Pham, Ninh, and Rasmus Pagh. "Fast and scalable polynomial kernels via explicit feature maps" Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. 2013.
- Avron, Haim, Huy Nguyen, and David Woodruff. "Subspace embeddings for the polynomial kernel." Advances in neural information processing systems 27 (2014).
- Malik, Osman Asif, and Stephen Becker. "Low-rank tucker decomposition of large tensors using tensorsketch." Advances in neural information processing systems 31 (2018).

Questions?