Low Rank Tucker Approximation of a Tensor from Streaming Data

Madeleine Udell

Operations Research and Information Engineering
Cornell University

Based on joint work with
Yiming Sun (Amazon), Yang Guo (UW Madison),
Charlene Luo (Cornell grad), and Joel Tropp (Caltech)

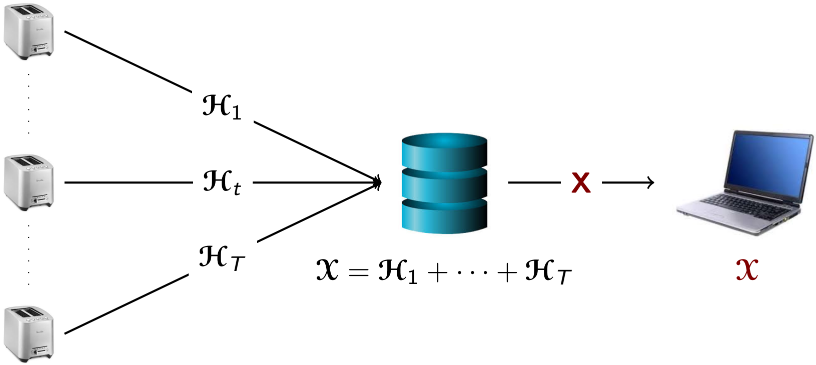
IPAM TMWS3, May 2021

Big data, small laptop

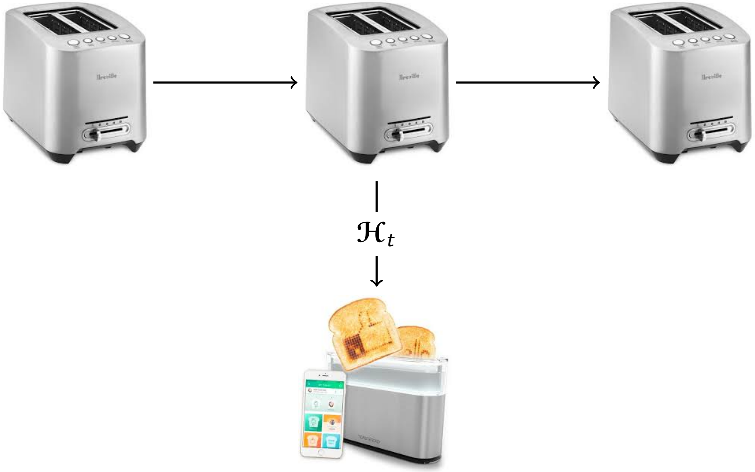
 $\xrightarrow{\quad \mathcal{H}_T \quad}$  $\xrightarrow{\quad \mathcal{X} \quad}$ 

$$\mathcal{X} = \mathcal{H}_1 + \cdots + \mathcal{H}_T \qquad\qquad \mathcal{X}$$

Distributed data



Streaming data



$$\mathcal{X}^{(t)} = \mathcal{H}_1 + \cdots + \mathcal{H}_t$$

Streaming multilinear algebra

turnstile model:

$$\mathcal{X} \;\; = \;\; \mathcal{H}_1 + \cdots + \mathcal{H}_T$$

- tensor $\mathcal{X}$ presented as sum of smaller, simpler tensors $\mathcal{H}_t$
- must discard $\mathcal{H}_t$ after it is processed
- **Goal:** without storing $\mathcal{X}$, approximate $\mathcal{X}$ after seeing all updates (with guaranteed accuracy)

Streaming multilinear algebra

turnstile model:

$$\mathcal{X} \;=\; \mathcal{H}_1 + \cdots + \mathcal{H}_T$$

- tensor $\mathcal{X}$ presented as sum of smaller, simpler tensors $\mathcal{H}_t$
- must discard $\mathcal{H}_t$ after it is processed
- **Goal:** without storing $\mathcal{X}$, approximate $\mathcal{X}$ after seeing all updates (with guaranteed accuracy)

applications:

- scientific simulation
- sensor measurements
- memory- or communication-limited computing
- low memory optimization

Linear sketch

$$\begin{aligned} \mathcal{X} &= \mathcal{H}_1 + \cdots + \mathcal{H}_T \\ \mathcal{L}(\mathcal{X}) &= \mathcal{L}(\mathcal{H}_1) + \cdots + \mathcal{L}(\mathcal{H}_T) \end{aligned}$$

Linear sketch

$$\begin{aligned} \mathcal{X} &= \mathcal{H}_1 + \cdots + \mathcal{H}_T \\ \mathcal{L}(\mathcal{X}) &= \mathcal{L}(\mathcal{H}_1) + \cdots + \mathcal{L}(\mathcal{H}_T) \end{aligned}$$

- select a linear map $\mathcal{L}$ independent of $\mathcal{X}$

Linear sketch

$$\begin{aligned} \mathcal{X} &= \mathcal{H}_1 + \cdots + \mathcal{H}_T \\ \mathcal{L}(\mathcal{X}) &= \mathcal{L}(\mathcal{H}_1) + \cdots + \mathcal{L}(\mathcal{H}_T) \end{aligned}$$

- select a linear map $\mathcal{L}$ independent of $\mathcal{X}$
- sketch $\mathcal{L}(\mathcal{X})$ is much smaller than input tensor $\mathcal{X}$

Linear sketch

$$\begin{aligned} \mathcal{X} &= \mathcal{H}_1 + \cdots + \mathcal{H}_T \\ \mathcal{L}(\mathcal{X}) &= \mathcal{L}(\mathcal{H}_1) + \cdots + \mathcal{L}(\mathcal{H}_T) \end{aligned}$$

- select a linear map $\mathcal{L}$ independent of $\mathcal{X}$
- sketch $\mathcal{L}(\mathcal{X})$ is much smaller than input tensor $\mathcal{X}$
- use randomness so sketch works for an arbitrary input

Linear sketch

$$\begin{aligned} \mathcal{X} &= \mathcal{H}_1 + \cdots + \mathcal{H}_T \\ \mathcal{L}(\mathcal{X}) &= \mathcal{L}(\mathcal{H}_1) + \cdots + \mathcal{L}(\mathcal{H}_T) \end{aligned}$$

- select a linear map $\mathcal{L}$ independent of $\mathcal{X}$
- sketch $\mathcal{L}(\mathcal{X})$ is much smaller than input tensor $\mathcal{X}$
- use randomness so sketch works for an arbitrary input
- essentially the only way to handle the turnstile model [Li et al. 2014]

examples:

- $\mathcal{L}(\mathcal{X}) = \mathcal{X} \times_n \mathbf{\Omega}$ for some matrix $\mathbf{\Omega}$
- $\mathcal{L}(\mathcal{X}) = \{\mathcal{X} \times_n \mathbf{\Omega}_n\}_{n \in [N]}$ for some matrices $\{\mathbf{\Omega}_n\}_{n \in [N]}$

Main idea

sketch suffices for (Tucker) approximation:

- compute (randomized) linear sketch of tensor
- recover low rank (Tucker) approximation from sketch

Main idea

sketch suffices for (Tucker) approximation:

- compute (randomized) linear sketch of tensor
- recover low rank (Tucker) approximation from sketch
- (optional) improve approximation by revisiting data

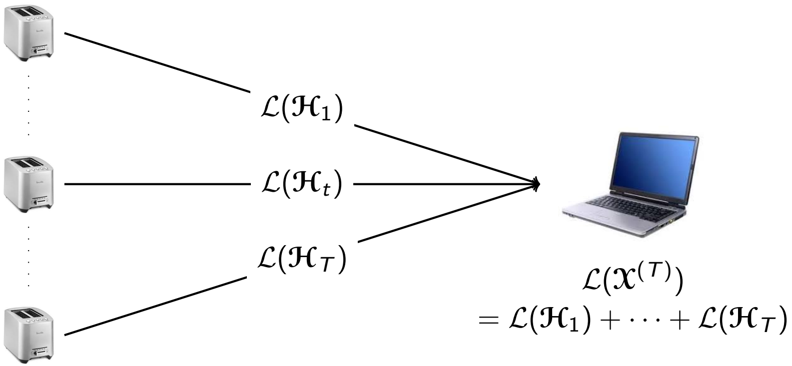
Big data, small laptop: sketch

 $\xrightarrow{\mathcal{H}_T}$  $\xrightarrow{\mathcal{L}(\mathcal{X})}$ 

$\mathcal{X} = \mathcal{H}_1 + \cdots + \mathcal{H}_T$ $\qquad$ $\mathcal{L}(\mathcal{X})$

- (+) reduced communication
- (+) sketch of data fits on laptop

Distributed data: sketch



- (+) reduced communication
- (+) no PITI (personally identifiable toast information)
- (+) sketch of data fits on laptop

Streaming data: sketch

 $\longrightarrow$  $\longrightarrow$ 

$$\Big| \quad \mathcal{L}(\mathcal{H}_t) \quad \downarrow$$

$$\mathcal{L}(\mathcal{X}^{(t)}) = \mathcal{L}(\mathcal{H}_1 + \cdots + \mathcal{H}_{t-1}) + \mathcal{L}(\mathcal{H}_t)$$

- $(+)$ even a toaster can form sketch

Notation

tensor to compress:

- tensor $\mathcal{X} \in \mathbf{R}^{I_1 \times \cdots \times I_N}$ with N modes
- sometimes assume $I_1 = \cdots = I_N = I$ for simplicity

Notation

tensor to compress:

- tensor $\mathcal{X} \in \mathbf{R}^{I_1 \times \cdots \times I_N}$ with N modes
- sometimes assume $I_1 = \cdots = I_N = I$ for simplicity

indexing:

- $[N] = 1, \ldots, N$
- $I_{(-n)} = I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times I_N$

Notation

tensor to compress:

- tensor $\mathcal{X} \in \mathbf{R}^{I_1 \times \cdots \times I_N}$ with N modes
- sometimes assume $I_1 = \cdots = I_N = I$ for simplicity

indexing:

- $[N] = 1, \ldots, N$
- $I_{(-n)} = I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times I_N$

tensor operations:

- mode n product: for $\mathcal{A} \in \mathbf{R}^{k \times I_n}$,
  $\mathcal{X} \times_n \mathbf{A} \in \mathbf{R}^{I_1 \times \cdots \times I_{n-1} \times k \times I_{n+1} \times \cdots \times I_N}$

Notation

tensor to compress:

- tensor $\mathcal{X} \in \mathbf{R}^{I_1 \times \cdots \times I_N}$ with N modes
- sometimes assume $I_1 = \cdots = I_N = I$ for simplicity

indexing:

- $[N] = 1, \ldots, N$
- $I_{(-n)} = I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times I_N$

tensor operations:

- mode n product: for $\mathcal{A} \in \mathbf{R}^{k \times I_n}$,
  $\mathcal{X} \times_n \mathbf{A} \in \mathbf{R}^{I_1 \times \cdots \times I_{n-1} \times k \times I_{n+1} \times \cdots \times I_N}$
- unfolding $\mathbf{X}^{(n)} \in \mathbf{R}^{I_n \times I_{(-n)}}$ stacks mode-n fibers of $\mathcal{X}$ as columns of matrix

Tucker factorization

rank $\mathbf{r} = (r_1, \ldots, r_N)$ **Tucker factorization** of $\mathcal{X} \in \mathbf{R}^{I_1 \times \cdots \times I_N}$:

$$\mathcal{X} \;\; = \;\; \mathcal{G} \times_1 \mathbf{U}_1 \cdots \times_N \mathbf{U}_N =: [\![\mathcal{G}; \mathbf{U}_1, \ldots, \mathbf{U}_N]\!]$$

where

- $\mathcal{G} \in \mathbf{R}^{r_1 \times \cdots \times r_N}$ is the **core matrix**
- $\mathbf{U}_n \in \mathbf{R}^{I_n \times r_n}$ is the **factor matrix** for each mode $n \in [N]$

Tucker factorization

rank $\mathbf{r} = (r_1, \ldots, r_N)$ **Tucker factorization** of $\mathcal{X} \in \mathbf{R}^{I_1 \times \cdots \times I_N}$:

$$\mathcal{X} \quad = \quad \mathcal{G} \times_1 \mathbf{U}_1 \cdots \times_N \mathbf{U}_N =: [\![\mathcal{G}; \mathbf{U}_1, \ldots, \mathbf{U}_N]\!]$$

where

- $\mathcal{G} \in \mathbf{R}^{r_1 \times \cdots \times r_N}$ is the **core matrix**
- $\mathbf{U}_n \in \mathbf{R}^{I_n \times r_n}$ is the **factor matrix** for each mode $n \in [N]$

(sometimes assume $r_1 = \cdots = r_N = r$ for simplicity)

Tucker factorization

rank $\mathbf{r} = (r_1, \ldots, r_N)$ **Tucker factorization** of $\mathcal{X} \in \mathbf{R}^{I_1 \times \cdots \times I_N}$:

$$\mathcal{X} \;=\; \mathcal{G} \times_1 \mathbf{U}_1 \cdots \times_N \mathbf{U}_N =: [\![\mathcal{G}; \mathbf{U}_1, \ldots, \mathbf{U}_N]\!]$$

where

- $\mathcal{G} \in \mathbf{R}^{r_1 \times \cdots \times r_N}$ is the **core matrix**
- $\mathbf{U}_n \in \mathbf{R}^{I_n \times r_n}$ is the **factor matrix** for each mode $n \in [N]$

(sometimes assume $r_1 = \cdots = r_N = r$ for simplicity)

Tucker is useful for compression: when N is small,

- Tucker stores $O(rNI)$ numbers for rank r^3 approximation
- CP stores $O(rNI)$ numbers for rank r approximation

Computing Tucker: HOSVD

Algorithm Higher order singular value decomposition (HOSVD)
[De Lathauwer, De Moor & Vandewalle 2000, Tucker 1966]

Given: tensor $\mathcal{X}$, target rank $\mathbf{r} = (r_1, \ldots, r_N)$

1. *Factors.* Compute top r_n left singular vectors $\mathbf{U}_n$ of the unfolding $\mathbf{X}^{(n)}$ for each $n \in [N]$.
2. *Core.* Contract these with $\mathcal{X}$ to form the core

$$\mathcal{G} = \mathcal{X} \times_1 \mathbf{U}_1^T \cdots \times_N \mathbf{U}_N^T.$$

Return: Tucker approximation $\mathcal{X}_{\mathrm{HOSVD}} = [\![\mathcal{G}; \mathbf{U}_1, \ldots, \mathbf{U}_N]\!]$

Two pass HOSVD

HOSVD can be computed in two passes over the tensor:

Two pass HOSVD

HOSVD can be computed in two passes over the tensor:

- **Factors.** use randomized linear algebra

Two pass HOSVD

HOSVD can be computed in two passes over the tensor:

- **Factors.** use randomized linear algebra
  - need to find span of fibers of $\mathcal{X}$ along nth mode:

$$\mathbf{range}(\mathbf{U}_n) \approx \mathbf{range}(\mathbf{X}^{(n)})$$

Two pass HOSVD

HOSVD can be computed in two passes over the tensor:

- **Factors.** use randomized linear algebra
  - need to find span of fibers of $\mathcal{X}$ along nth mode:

$$\mathbf{range}(\mathbf{U}_n) \approx \mathbf{range}(\mathbf{X}^{(n)})$$

  - if $\mathbf{rank}(\boldsymbol{\Omega}) \geq \mathbf{rank}(\mathbf{X}^{(n)})$, then whp for random $\boldsymbol{\Omega}$,

$$\mathbf{range}(\mathbf{X}^{(n)}) = \mathbf{range}(\mathbf{X}^{(n)}\boldsymbol{\Omega})$$

Two pass HOSVD

HOSVD can be computed in two passes over the tensor:

- **Factors.** use randomized linear algebra
  - need to find span of fibers of $\mathcal{X}$ along nth mode:

$$\mathbf{range}(\mathbf{U}_n) \approx \mathbf{range}(\mathbf{X}^{(n)})$$

  - if $\mathbf{rank}(\mathbf{\Omega}) \geq \mathbf{rank}(\mathbf{X}^{(n)})$, then whp for random $\mathbf{\Omega}$,

$$\mathbf{range}(\mathbf{X}^{(n)}) = \mathbf{range}(\mathbf{X}^{(n)}\mathbf{\Omega})$$

algorithm:

1. compute sketch $\mathcal{L}(\mathcal{X}) = \{\mathbf{X}^{(n)}\mathbf{\Omega}_n\}_{n \in [N]}$
2. use QR on sketch to approximate $\mathbf{range}(\mathbf{X}^{(n)})$

Two pass HOSVD

HOSVD can be computed in two passes over the tensor:

- **Factors.** use randomized linear algebra
  - need to find span of fibers of $\mathcal{X}$ along nth mode:

$$\mathbf{range}(\mathbf{U}_n) \approx \mathbf{range}(\mathbf{X}^{(n)})$$

  - if $\mathbf{rank}(\boldsymbol{\Omega}) \geq \mathbf{rank}(\mathbf{X}^{(n)})$, then whp for random $\boldsymbol{\Omega}$,

$$\mathbf{range}(\mathbf{X}^{(n)}) = \mathbf{range}(\mathbf{X}^{(n)}\boldsymbol{\Omega})$$

  algorithm:
  1. compute sketch $\mathcal{L}(\mathcal{X}) = \{\mathbf{X}^{(n)}\boldsymbol{\Omega}_n\}_{n\in[N]}$
  2. use QR on sketch to approximate $\mathbf{range}(\mathbf{X}^{(n)})$
- **Core.** Computation is linear in $\mathcal{X}$:

$$\mathcal{G} = \mathcal{X} \times_1 \mathbf{U}_1^T \cdots \times_N \mathbf{U}_N^T.$$

Source: [Halko et al. 2011, Zhou et al. 2014, Battaglino et al. 2019]

Computing Tucker: HOOI

Algorithm Higher order orthogonal iteration (HOOI) [De Lathauwer et al. 2000]

Given: tensor $\mathcal{X}$, target rank $\mathbf{r} = (r_1, \ldots, r_N)$
Initialize: compute $\mathcal{X} \approx [\![\mathcal{G}; \mathbf{U}_1, \ldots, \mathbf{U}_N]\!]$ using HOSVD
Repeat:

1. *Factors.* For $n \in [N]$,

$$\mathbf{U}_n \leftarrow \underset{\mathbf{U_n}}{\operatorname{argmin}} \|[\![\mathcal{G}; \mathbf{U}_1, \ldots, \mathbf{U}_N]\!] - \mathcal{X}\|_F^2 ,$$

2. *Core.*

$$\mathcal{G} \leftarrow \underset{\mathcal{G}}{\operatorname{argmin}} \|[\![\mathcal{G}; \mathbf{U}_1, \ldots, \mathbf{U}_N]\!] - \mathcal{X}\|_F^2 .$$

Return: Tucker approximation $\mathcal{X}_{\mathrm{HOOI}} = [\![\mathcal{G}; \mathbf{U}_1, \ldots, \mathbf{U}_N]\!]$

Computing Tucker: HOOI

Algorithm Higher order orthogonal iteration (HOOI) [De Lathauwer et al. 2000]

Given: tensor $\mathcal{X}$, target rank $\mathbf{r} = (r_1, \ldots, r_N)$
Initialize: compute $\mathcal{X} \approx [\![\mathcal{G}; \mathbf{U}_1, \ldots, \mathbf{U}_N]\!]$ using HOSVD
Repeat:

1. *Factors.* For $n \in [N]$,

$$\mathbf{U}_n \leftarrow \operatorname*{argmin}_{\mathbf{U_n}} \|[\![\mathcal{G}; \mathbf{U}_1, \ldots, \mathbf{U}_N]\!] - \mathcal{X}\|_F^2 \,,$$

2. *Core.*

$$\mathcal{G} \leftarrow \operatorname*{argmin}_{\mathcal{G}} \|[\![\mathcal{G}; \mathbf{U}_1, \ldots, \mathbf{U}_N]\!] - \mathcal{X}\|_F^2 \,.$$

Return: Tucker approximation $\mathcal{X}_{\mathrm{HOOI}} = [\![\mathcal{G}; \mathbf{U}_1, \ldots, \mathbf{U}_N]\!]$

- core update has closed form $\mathcal{G} \leftarrow \mathcal{X} \times_1 \mathbf{U}_1^\top \cdots \times_N \mathbf{U}_N^\top$

Previous work: one pass algorithm via HOOI

[Malik & Becker 2018]:

- (+) sketch design matrix to reduce size of HOOI subproblems
- (+) exploit Tucker structure of design matrix
- (-) expensive slow reconstruction (via iterative optimization)
- (-) no error guarantees for one pass algorithm

Background: randomized sketches

idea: random matrix $\mathbf{\Omega}$ is not orthogonal to range of interest (whp)

$$\textbf{range}(\mathbf{X}^{(n)}) = \textbf{range}(\mathbf{X}^{(n)}\mathbf{\Omega})$$

Background: randomized sketches

idea: random matrix $\mathbf{\Omega}$ is not orthogonal to range of interest (whp)

$$\mathbf{range}(\mathbf{X}^{(n)}) = \mathbf{range}(\mathbf{X}^{(n)}\mathbf{\Omega})$$

a **dimension reduction map** (DRM) (approximately) preserves range of its argument

Background: randomized sketches

idea: random matrix $\mathbf{\Omega}$ is not orthogonal to range of interest (whp)

$$\textbf{range}(\mathbf{X}^{(n)}) = \textbf{range}(\mathbf{X}^{(n)}\mathbf{\Omega})$$

a **dimension reduction map** (DRM) (approximately) preserves range of its argument

examples of DRMS: multiplication by random matrix $\mathbf{\Omega}$ that is

- gaussian
- sparse [Achlioptas 2003, Li et al. 2006]
- SSFRT [Woolfe et al. 2008]
- tensor random projection (TRP) [Sun et al. 2018]
- ...

The sketch

approximate factor matrices and core:

The sketch

approximate factor matrices and core:

- **Factor sketch (k)**. For each $n \in [N]$,
  fix random DRM $\boldsymbol{\Omega}_n \in \mathbb{R}^{I_{(-n)} \times k_n}$ and compute the sketch

$$\mathbf{V}_n = \mathbf{X}^{(n)} \boldsymbol{\Omega}_n \quad \in \mathbf{R}^{I_n \times k_n}.$$

The sketch

approximate factor matrices and core:

- **Factor sketch** ($\mathbf{k}$). For each $n \in [N]$, fix random DRM $\mathbf{\Omega}_n \in \mathbb{R}^{I_{(-n)} \times k_n}$ and compute the sketch

$$\mathbf{V}_n = \mathbf{X}^{(n)} \mathbf{\Omega}_n \quad \in \mathbf{R}^{I_n \times k_n}.$$

- **Core sketch** ($\mathbf{s}$). For each $n \in [N]$, fix random DRM $\mathbf{\Phi}_n \in \mathbf{R}^{I_n \times s_n}$. Compute the sketch

$$\mathcal{H} = \mathcal{X} \times_1 \mathbf{\Phi}_1^\top \cdots \times_N \mathbf{\Phi}_N^\top \quad \in \mathbf{R}^{s_1 \times \cdots \times s_N}.$$

The sketch

approximate factor matrices and core:

- **Factor sketch** ($\mathbf{k}$). For each $n \in [N]$, fix random DRM $\boldsymbol{\Omega}_n \in \mathbb{R}^{I_{(-n)} \times k_n}$ and compute the sketch

$$\mathbf{V}_n = \mathbf{X}^{(n)} \boldsymbol{\Omega}_n \quad \in \mathbf{R}^{I_n \times k_n}.$$

- **Core sketch** ($\mathbf{s}$). For each $n \in [N]$, fix random DRM $\boldsymbol{\Phi}_n \in \mathbf{R}^{I_n \times s_n}$. Compute the sketch

$$\mathcal{H} = \mathcal{X} \times_1 \boldsymbol{\Phi}_1^\top \cdots \times_N \boldsymbol{\Phi}_N^\top \quad \in \mathbf{R}^{s_1 \times \cdots \times s_N}.$$

- *Rule of thumb.* Pick $\mathbf{k}$ as big as you can afford, pick $\mathbf{s} = 2\mathbf{k}$.

The sketch

approximate factor matrices and core:

- **Factor sketch (k)**. For each $n \in [N]$,
  fix random DRM $\mathbf{\Omega}_n \in \mathbb{R}^{I_{(-n)} \times k_n}$ and compute the sketch

$$\mathbf{V}_n = \mathbf{X}^{(n)} \mathbf{\Omega}_n \quad \in \mathbf{R}^{I_n \times k_n}.$$

- **Core sketch (s)**. For each $n \in [N]$,
  fix random DRM $\mathbf{\Phi}_n \in \mathbf{R}^{I_n \times s_n}$. Compute the sketch

$$\mathcal{H} = \mathcal{X} \times_1 \mathbf{\Phi}_1^\top \cdots \times_N \mathbf{\Phi}_N^\top \quad \in \mathbf{R}^{s_1 \times \cdots \times s_N}.$$

- *Rule of thumb.* Pick $\mathbf{k}$ as big as you can afford, pick $\mathbf{s} = 2\mathbf{k}$.
- define $(\mathcal{H}, \mathbf{V}_1, \ldots, \mathbf{V}_N) = \textsc{Sketch}\left(\mathcal{X}; \{\mathbf{\Phi}_n, \mathbf{\Omega}_n\}_{n \in [N]}\right)$

Low memory DRMs

factor sketch DRMs are big!

- $I_{(-n)} \times k_n$ for each $n \in [N]$

Low memory DRMs

factor sketch DRMs are big!

- $I_{(-n)} \times k_n$ for each $n \in [N]$

how to store?

- don't store DRMS; instead, use pseudorandom number generator to generate (parts of) DRMs as needed.
- use structured DRM:
  - TRP generates DRM as Khatri-Rao product of simpler, smaller DRMs
  - behaves approximately like a Gaussian sketch

Source: [Sun et al. 2018, Rudelson 2012]

Recovery: factor matrices

- compute QR factorization of each factor sketch $\mathbf{V}_n$:

$$\mathbf{V}_n = \mathbf{Q}_n \mathbf{R}_n$$

where $\mathbf{Q}_n$ is orthonormal and $\mathbf{R}_n$ is triangular

Two pass algorithm

Algorithm Two Pass Sketch and Low Rank Recovery

Given: tensor $\mathcal{X}$, DRMs $\{\mathbf{\Phi}_n, \mathbf{\Omega}_n\}_{n\in[N]}$ with parameters $\mathbf{k}$ and $\mathbf{s} \geq \mathbf{k}$

1. *Sketch.* $(\mathcal{H}, \mathbf{V}_1, \dots, \mathbf{V}_N) = \textsc{Sketch}\left(\mathcal{X}; \{\mathbf{\Phi}_n, \mathbf{\Omega}_n\}_{n\in[N]}\right)$
2. *Recover factor matrices.* For $n \in [N]$,

$$(\mathbf{Q}_n, \sim) \leftarrow \mathrm{QR}(\mathbf{V}_\mathrm{n})$$

3. *Recover core.*

$$\mathcal{W} \leftarrow \mathcal{X} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N$$

Return: Tucker approximation $\tilde{\mathcal{X}} = [\![\mathcal{W}; \mathbf{Q}_1, \dots, \mathbf{Q}_N]\!]$ with rank $\leq \mathbf{k}$

Two pass algorithm

Algorithm Two Pass Sketch and Low Rank Recovery

Given: tensor $\mathcal{X}$, DRMs $\{\boldsymbol{\Phi}_n, \boldsymbol{\Omega}_n\}_{n\in[N]}$ with parameters $\mathbf{k}$ and $\mathbf{s} \geq \mathbf{k}$

1. *Sketch.* $(\mathcal{H}, \mathbf{V}_1, \ldots, \mathbf{V}_N) = \textsc{Sketch}\left(\mathcal{X}; \{\boldsymbol{\Phi}_n, \boldsymbol{\Omega}_n\}_{n\in[N]}\right)$
2. *Recover factor matrices.* For $n \in [N]$,

$$(\mathbf{Q}_n, \sim) \leftarrow \mathrm{QR}(\mathbf{V}_\mathrm{n})$$

3. *Recover core.*

$$\mathcal{W} \leftarrow \mathcal{X} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N$$

Return: Tucker approximation $\tilde{\mathcal{X}} = [\![\mathcal{W}; \mathbf{Q}_1, \ldots, \mathbf{Q}_N]\!]$ with rank $\leq \mathbf{k}$

accesses $\mathcal{X}$ twice: 1) to sketch 2) to recover core

Intuition: one pass core recovery

- we want to know $\mathcal{W}$:
  compression of $\mathcal{X}$ using factor range approximations $\mathbf{Q}_n$
- we observe $\mathcal{H}$:
  compression of $\mathcal{X}$ using random projections $\boldsymbol{\Phi}_n$

how to approximate $\mathcal{W}$?

Intuition: one pass core recovery

- we want to know $\mathcal{W}$:
  compression of $\mathcal{X}$ using factor range approximations $\mathbf{Q}_n$
- we observe $\mathcal{H}$:
  compression of $\mathcal{X}$ using random projections $\mathbf{\Phi}_n$

how to approximate $\mathcal{W}$?

$$\begin{aligned}
\mathcal{X} &\approx \mathcal{X} \times_1 \mathbf{Q}_1\mathbf{Q}_1^\top \times \cdots \times_N \mathbf{Q}_N\mathbf{Q}_N^\top \\
&= \left(\mathcal{X} \times_1 \mathbf{Q}_1^\top \times_N \cdots \times \mathbf{Q}_N^\top\right) \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N \\
&= \mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N \\
\underbrace{\mathcal{X} \times_1 \mathbf{\Phi}_1^\top \cdots \times_N \mathbf{\Phi}_N^\top}_{\mathcal{H}} &\approx \mathcal{W} \times_1 \mathbf{\Phi}_1^\top\mathbf{Q}_1 \times \cdots \times_N \mathbf{\Phi}_N^\top\mathbf{Q}_N
\end{aligned}$$

Intuition: one pass core recovery

- we want to know $\mathcal{W}$:
  compression of $\mathcal{X}$ using factor range approximations $\mathbf{Q}_n$
- we observe $\mathcal{H}$:
  compression of $\mathcal{X}$ using random projections $\mathbf{\Phi}_n$

how to approximate $\mathcal{W}$?

$$
\begin{aligned}
\mathcal{X} &\approx \mathcal{X} \times_1 \mathbf{Q}_1 \mathbf{Q}_1^\top \times \cdots \times_N \mathbf{Q}_N \mathbf{Q}_N^\top \\
&= \left(\mathcal{X} \times_1 \mathbf{Q}_1^\top \times_N \cdots \times \mathbf{Q}_N^\top \right) \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N \\
&= \mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N \\
\underbrace{\mathcal{X} \times_1 \mathbf{\Phi}_1^\top \cdots \times_N \mathbf{\Phi}_N^\top}_{\mathcal{H}} &\approx \mathcal{W} \times_1 \mathbf{\Phi}_1^\top \mathbf{Q}_1 \times \cdots \times_N \mathbf{\Phi}_N^\top \mathbf{Q}_N
\end{aligned}
$$

we can solve for $\mathcal{W}$: $\mathbf{s} > \mathbf{k}$, so each $\mathbf{\Phi}_n^\top \mathbf{Q}_n$ has a left inverse (whp):

$$\mathcal{W} \approx \mathcal{H} \times_1 (\mathbf{\Phi}_1^\top \mathbf{Q}_1)^\dagger \times \cdots \times_N (\mathbf{\Phi}_N^\top \mathbf{Q}_N)^\dagger$$

One pass algorithm

Algorithm One Pass Sketch and Low Rank Recovery

Given: tensor $\mathcal{X}$, rank $\mathbf{r} = (r_1, \ldots, r_N)$, DRMs $\{\mathbf{\Phi}_n, \mathbf{\Omega}_n\}_{n \in [N]}$

- *Sketch.* $(\mathcal{H}, \mathbf{V}_1, \ldots, \mathbf{V}_N) = \textsc{Sketch}\left(\mathcal{X}; \{\mathbf{\Phi}_n, \mathbf{\Omega}_n\}_{n \in [N]}\right)$
- *Recover factor matrices.* For $n \in [N]$,

$$(\mathbf{Q}_n, \sim) \leftarrow \mathrm{QR}(\mathbf{V}_\mathrm{n})$$

- *Recover core.*

$$\mathcal{W} \leftarrow \mathcal{H} \times_1 (\mathbf{\Phi}_1^\top \mathbf{Q}_1)^\dagger \times \cdots \times_N (\mathbf{\Phi}_N^\top \mathbf{Q}_N)^\dagger$$

Return: Tucker approximation $\hat{\mathcal{X}} = [\![\mathcal{W}; \mathbf{Q}_1, \ldots, \mathbf{Q}_N]\!]$

One pass algorithm

Algorithm One Pass Sketch and Low Rank Recovery

Given: tensor $\mathcal{X}$, rank $\mathbf{r} = (r_1, \ldots, r_N)$, DRMs $\{\mathbf{\Phi}_n, \mathbf{\Omega}_n\}_{n \in [N]}$

- *Sketch.* $(\mathcal{H}, \mathbf{V}_1, \ldots, \mathbf{V}_N) = \textsc{Sketch}\left(\mathcal{X}; \{\mathbf{\Phi}_n, \mathbf{\Omega}_n\}_{n \in [N]}\right)$
- *Recover factor matrices.* For $n \in [N]$,

$$(\mathbf{Q}_n, \sim) \leftarrow \mathrm{QR}(\mathbf{V}_\mathrm{n})$$

- *Recover core.*

$$\mathcal{W} \leftarrow \mathcal{H} \times_1 (\mathbf{\Phi}_1^\top \mathbf{Q}_1)^\dagger \times \cdots \times_N (\mathbf{\Phi}_N^\top \mathbf{Q}_N)^\dagger$$

Return: Tucker approximation $\hat{\mathcal{X}} = [\![\mathcal{W}; \mathbf{Q}_1, \ldots, \mathbf{Q}_N]\!]$

accesses $\mathcal{X}$ only once, to sketch

Source: [Sun et al. 2019]

Fixed rank approximation

to truncate reconstruction to rank $\mathbf{r}$, truncate core:

Lemma

For a tensor $\mathcal{W} \in \mathbb{R}^{k_1 \times \cdots \times k_N}$, orthogonal matrices $\mathbf{Q}_n \in \mathbb{R}^{k_n \times r_n}$,

$$[\![\mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N]\!]_{\mathbf{r}} = [\![\mathcal{W}]\!]_{\mathbf{r}} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N,$$

where $[\![\cdot]\!]$ denotes the best rank $\mathbf{r}$ Tucker approximation.

Fixed rank approximation

to truncate reconstruction to rank **r**, truncate core:

Lemma

For a tensor $\mathcal{W} \in \mathbb{R}^{k_1 \times \cdots \times k_N}$, orthogonal matrices $\mathbf{Q}_n \in \mathbb{R}^{k_n \times r_n}$,

$$[\![\mathcal{W} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N]\!]_{\mathbf{r}} = [\![\mathcal{W}]\!]_{\mathbf{r}} \times_1 \mathbf{Q}_1 \cdots \times_N \mathbf{Q}_N,$$

where $[\![\cdot]\!]$ denotes the best rank **r** Tucker approximation.

$\implies$ compute fixed rank approximation using, e.g., HOOI on (small) core approximation $\mathcal{W}$

Tail energy

For each unfolding $\mathbf{X}^{(n)}$, define its ρ*th tail energy* as

$$(\tau_\rho^{(n)})^2 := \sum_{k>\rho}^{\min(I_n, I_{(-n)})} \sigma_k^2(\mathbf{X}^{(n)}),$$

where $\sigma_k(\mathbf{X}^{(n)})$ is the kth largest singular value of $\mathbf{X}^{(n)}$.

Guarantees (I)

Theorem (Recommended parameters [Sun et al. 2019])

Sketch $\mathcal{X}$ with Gaussian DRMs of parameters $\mathbf{k} = \mathbf{r} + 1$, $\mathbf{s} = 2\mathbf{k} + 1$. Form a rank $\mathbf{r}$ Tucker approximation $\hat{\mathcal{X}}$ using the one pass algorithm. Then

$$\mathbb{E}\|\mathcal{X} - \hat{\mathcal{X}}\|_F^2 \leq 4 \sum_{n=1}^{N} (\tau_{r_n}^{(n)})^2.$$

If $\mathcal{X}$ is truly rank $\mathbf{r}$, we obtain the true Tucker factorization!

Guarantees (II)

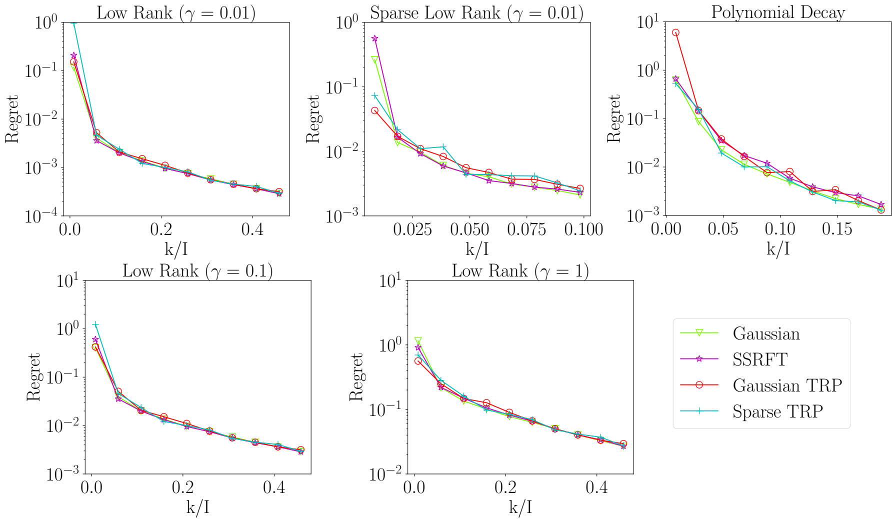
Theorem (Detailed guarantee [Sun et al. 2019])

Sketch $\mathfrak{X}$ with Gaussian DRMs of parameters $\mathbf{k}$, $\mathbf{s} \geq 2\mathbf{k} + 1$. Form a rank $\mathbf{r}$ Tucker approximation $\hat{\mathfrak{X}}$ using the one pass algorithm. Then

$$\mathbb{E}\|\mathfrak{X} - \hat{\mathfrak{X}}\|_F^2 \leq (1 + \Delta) \min_{1 \leq \rho_n < k_n - 1} \sum_{n=1}^{N} \left(1 + \frac{\rho_n}{k_n - \rho_n - 1}\right) (\tau_{\rho_n}^{(n)})^2$$

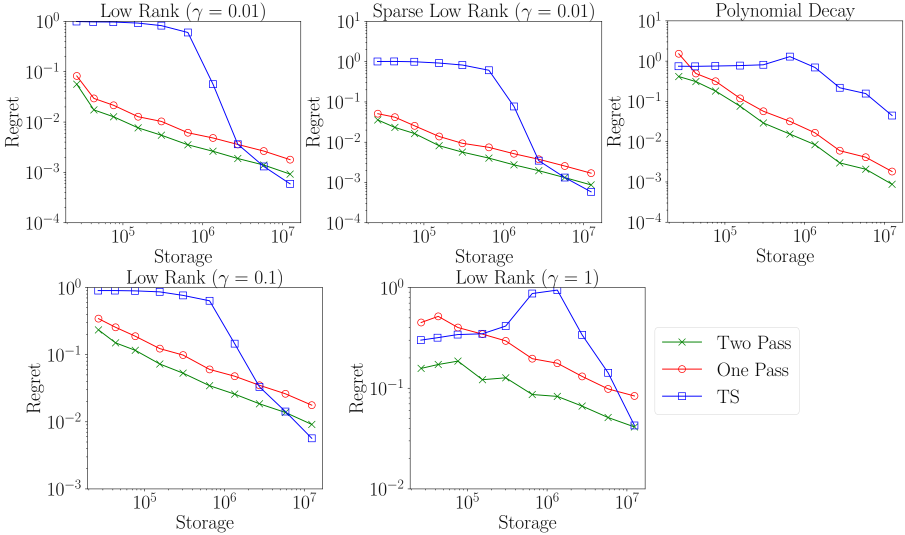
where $\Delta = \max_{n=1}^{N} k_n/(s_n - k_n - 1)$

Different DRMs perform similarly



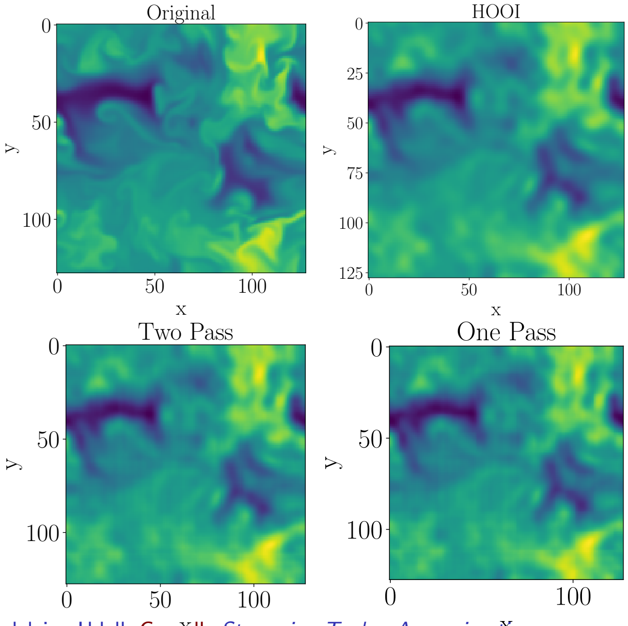
Comments: Synthetic data, $I = 600$ and $\mathbf{r} = (5, 5, 5)$. $k/I = .4 \implies 20\times$ compression.

Sensible reconstruction at practical compression level



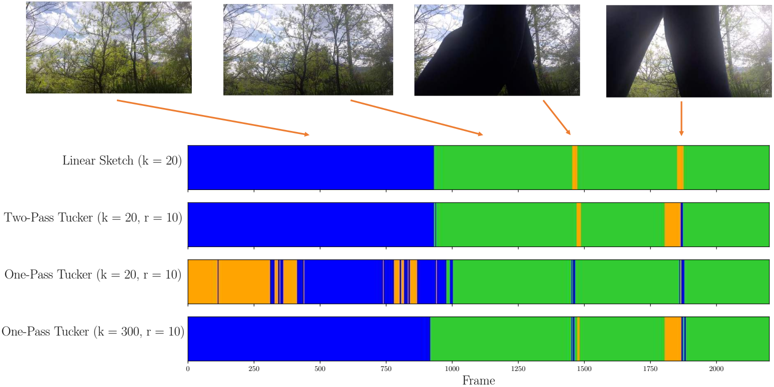
Comments: Error of fixed-rank approximation relative to HOOI for $r = 10$, $I = 300$ using TRP. Total memory use is $((2k+1)^N + kIN)$ and $(Kr^{2N} + K * r^{2N-2})$. Low-rank data uses $\gamma = 0.01, 0.1, 1$.

Combustion simulation



Comments:
$1408 \times 128 \times 128$ simulated combustion data from [Lapointe, Savard & Blanquart 2015].

Video scene classification



Comments: Video data $2200 \times 1080 \times 1980$. Classify scenes using k-means on: 1) linear sketch along the time dimension $k = 20$ (Row 1); 2) The Tucker factor along the time dimension, computed via our two pass (Row 2) and one pass (Row 3) sketching algorithm $(r, k, s) = (10, 20, 41)$. 3) The Tucker factor along the time dimension, computed via our one pass (Row 4) sketching algorithm $(r, k, s) = (10, 300, 601)$.

Summary

Streaming Tucker approximation compresses tensor without storing it.

useful for:

- streaming data
- distributed data
- low memory compute

key ideas:

- form linear sketch of tensor and recover from sketch
- random projection of tensor preserves dominant information

References

- Sun, Y., Guo, Y., Tropp, J. A., and Udell, M. Tensor random projection for low memory dimension reduction. In NeurIPS Workshop on Relational Representation Learning, 2018.
- Sun, Y., Guo, Y., Luo, C., Tropp, J. A., and Udell, M. Low rank Tucker approximation of a tensor from streaming data. SIMODS 2020.
- Tropp, J. A., Yurtsever, A., Udell, M., and Cevher, V. Streaming low-rank matrix approximation with an application to scientific simulation. SISC 2019.

Achlioptas, D. (2003). Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, *66*(4), 671–687.

Battaglino, C., Ballard, G., & Kolda, T. G. (2019). Faster parallel tucker tensor decomposition using randomization.

De Lathauwer, L., De Moor, B., & Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, *21*(4), 1253–1278.

Halko, N., Martinsson, P.-G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, *53*(2), 217–288.

Lapointe, S., Savard, B., & Blanquart, G. (2015). Differential diffusion effects, distributed burning, and local extinctions in high karlovitz premixed flames. *Combustion and flame*, *162*(9), 3341–3355.

Li, P., Hastie, T. J., & Church, K. W. (2006). Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (pp. 287–296). ACM.

Li, Y., Nguyen, H. L., & Woodruff, D. P. (2014). Turnstile streaming algorithms might as well be linear sketches. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, (pp. 174–183). ACM.

Malik, O. A. & Becker, S. (2018). Low-rank tucker decomposition of large tensors using tensorsketch. In *Advances in Neural Information Processing Systems*, (pp. 10116–10126).

Rudelson, M. (2012). Row products of random matrices. *Advances in Mathematics*, *231*(6), 3199–3231.

Sun, Y., Guo, Y., Luo, C., Tropp, J. A., & Udell, M. (2020). Low rank tucker approximation of a tensor from streaming data. *SIMODS*.

Sun, Y., Guo, Y., Tropp, J. A., & Udell, M. (2018). Tensor random projection for low memory dimension reduction. In *NeurIPS Workshop on Relational Representation Learning*.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, *31*(3), 279–311.

Woolfe, F., Liberty, E., Rokhlin, V., & Tygert, M. (2008). A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, *25*(3), 335–366.

Zhou, G., Cichocki, A., & Xie, S. (2014). Decomposition of big tensors with low multilinear rank. *arXiv preprint arXiv:1412.1885*.