Tensor Decompositions and Structured Randomization

Lecture 2 of 2: what replaces the SVD when there are more than two modes

Hussam Al Daas

Computational Mathematics Theme, STFC-RAL

What to do with multi-way data

Exploratory analysis of *matrix* data starts with the SVD, or PCA.

But many datasets have more than two modes:

- if an image is 2D, a video is 3D
- a single social network is 2D; several modalities makes it 3D or higher
- a parametrized PDE has one mode per parameter

What to do with multi-way data

Exploratory analysis of *matrix* data starts with the SVD, or PCA.

But many datasets have more than two modes:

- if an image is 2D, a video is 3D
- a single social network is 2D; several modalities makes it 3D or higher
- a parametrized PDE has one mode per parameter

Three options:

- **extract or average** into a matrix — losing information
- **reshape/unfold** into a matrix — losing relationships between modes
- **keep the multi-way structure** and use tensor methods

Warmup: the matrix SVD, and the four things we want from it

The **SVD** of an $M \times N$ matrix A:

$$\mathrm{A} = \mathrm{U}\Sigma\mathrm{V}^T = \sum_{i=1}^{N} \sigma_i \mathrm{u}_i \mathrm{v}_i^T \qquad \text{and the truncated SVD, a rank-}R\text{ approximation,}$$

$$\mathrm{A} \approx \mathrm{A}_R = \mathrm{U}_R \Sigma_R \mathrm{V}_R^T = \sum_{i=1}^{R} \sigma_i \mathrm{u}_i \mathrm{v}_i^T$$

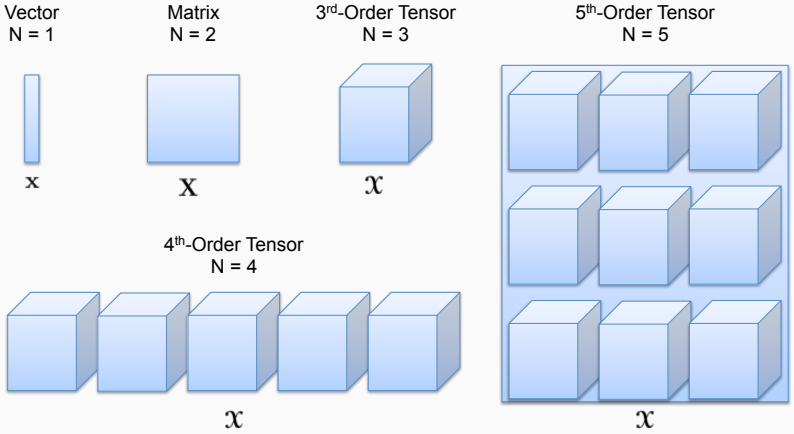
We are going to meet three tensor generalizations. Rather than list them, let us score each one on what the SVD gives us for free:

unique / interpretable factors	✔
solves the approximation problem	✔
compresses (storage linear in dimensions)	✔
stable, predictable algorithms	✔

No tensor format scores four out of four. Which boxes you need decides which format you use — that is the real content of the next forty minutes.

Notation

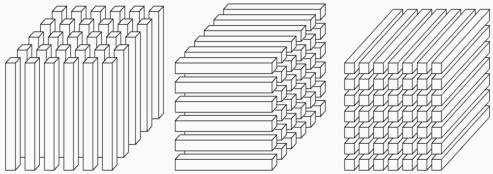
Basic tensor notation



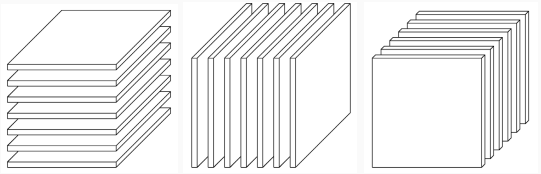
An N^{th}-order tensor has N *modes*

Notation convention: vector v, matrix M, tensor $\mathcal{T}$

Fibers and slices



fibers: fix all indices but one



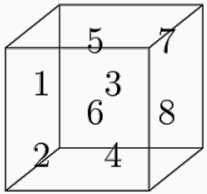
(a) Horizontal slices: $\mathbf{X}_{i,:,:}$ (b) Lateral slices: $\mathbf{X}_{:,j,:}$ (c) Frontal slices: $\mathbf{X}_{:,:,k}$ (or $\mathbf{X}_k$)

slices: fix one index

A tensor can be partitioned into the fibers, or the slices, of each mode.

Modal unfoldings / matricizations

$$\mathcal{X} =$$



$$\mathbf{X}_{(1)} = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix}$$

$$\mathbf{X}_{(2)} = \begin{bmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{bmatrix}$$

$$\mathbf{X}_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

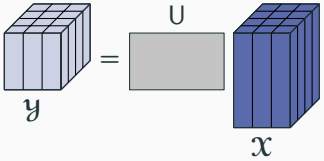
A tensor can be reshaped into matrices, called *unfoldings* or *matricizations*

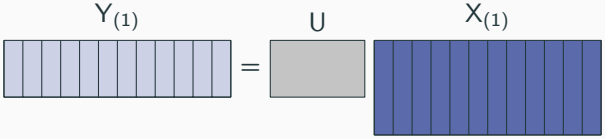
For *modal unfoldings*, fibers form columns and slices form rows

This is the bridge to everything you already know — but note the shape: $\mathbf{X}_{(n)}$ is $I_n \times (I_1 \cdots I_{n-1} I_{n+1} \cdots I_N)$, catastrophically wide.

Tensor times matrix (TTM)

Notation: $\mathcal{Y} = \mathcal{X} \times_1 \mathrm{U}$





- TTM is mode specific
- the input matrix multiplies all fibers in that mode
- unfolded, it is just matrix multiplication
- multiple TTMs compose, in any order

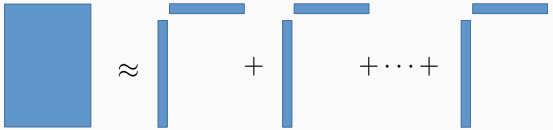
$$\mathcal{T} = \mathcal{G} \times_1 \mathrm{U} \times_2 \mathrm{V} \times_3 \mathrm{W}$$

This one operation is most of what we do all lecture. Remember it.

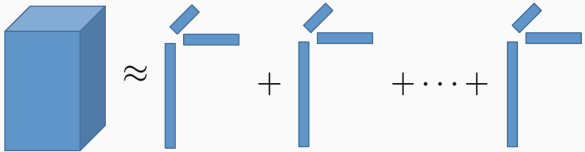
CP

CP: think of the SVD as a sum of outer products

Matrix:

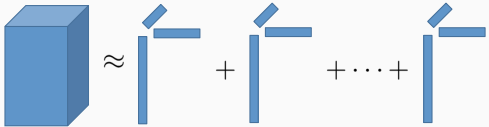


Tensor:



Canonical Polyadic, a.k.a. CANDECOMP/PARAFAC.

CP notation



$\mathcal{X} \approx \mathrm{a}_1 \circ \mathrm{b}_1 \circ \mathrm{c}_1 + \cdots + \mathrm{a}_r \circ \mathrm{b}_r \circ \mathrm{c}_r,$

$\mathcal{X} \in \mathbb{R}^{m \times n \times p}$

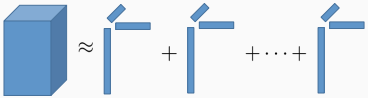
$\mathcal{X} \approx [\![\mathrm{A}, \mathrm{B}, \mathrm{C}]\!],$

$\mathrm{A} \in \mathbb{R}^{m \times r}, \mathrm{B} \in \mathbb{R}^{n \times r}, \mathrm{C} \in \mathbb{R}^{p \times r}$ are *factor matrices*

$x_{ijk} \approx \sum_{\ell=1}^{r} a_{i\ell} b_{j\ell} c_{k\ell},$

$1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq p$

The CP optimization problem



For fixed rank r, we want to solve

$$\min_{A,B,C} \left\| \mathcal{X} - \sum_{\ell=1}^{r} a_\ell \circ b_\ell \circ c_\ell \right\|$$

which is a nonlinear, nonconvex optimization problem

- in the matrix case, the SVD hands us the optimal solution
- in the tensor case, there is no such thing — no HOSVD-style shortcut, and the best rank-r approximation may not even exist

Alternating least squares — and the first structured matrix

Fix all but one factor matrix and the problem becomes *linear* least squares:

$$\min_{\mathsf{B}} \left\| \mathcal{X} - \sum_{\ell=1}^{r} \hat{\mathsf{a}}_\ell \circ \mathsf{b}_\ell \circ \hat{\mathsf{c}}_\ell \right\|$$

Alternating least squares — and the first structured matrix

Fix all but one factor matrix and the problem becomes *linear* least squares:

$$\min_{\mathsf{B}} \left\| \mathcal{X} - \sum_{\ell=1}^{r} \hat{\mathsf{a}}_\ell \circ \mathsf{b}_\ell \circ \hat{\mathsf{c}}_\ell \right\|$$

To see why, *matricize*:

$$\min_{\mathsf{B}} \| \underbrace{\mathsf{X}_{(2)}}_{n \times mp} - \underbrace{\mathsf{B}}_{n \times r} \underbrace{(\hat{\mathsf{C}} \odot \hat{\mathsf{A}})^{\mathsf{T}}}_{r \times mp} \|_F$$

where $\odot$ is the Khatri-Rao product, a column-wise Kronecker product:

$$\hat{\mathsf{C}} \odot \hat{\mathsf{A}} = \left[\hat{\mathsf{c}}_1 \otimes \hat{\mathsf{a}}_1 \cdots \hat{\mathsf{c}}_r \otimes \hat{\mathsf{a}}_r \right]$$

Note what just happened. The coefficient matrix is $mp \times r$ — enormous — but it is stored as two small factors. *We will never form it.*

The least squares subproblem, and where the time goes

The subproblem

$$\min_{B} \| \underbrace{X_{(2)}}_{n \times mp} - \underbrace{B}_{n \times r} \underbrace{(\hat{C} \odot \hat{A})^T}_{r \times mp} \|_F$$

has normal equations

$$\underbrace{(\hat{C} \odot \hat{A})^T (\hat{C} \odot \hat{A})}_{r \times r} \cdot \underbrace{B^T}_{r \times n} = (\underbrace{X_{(2)}}_{n \times mp} \underbrace{(\hat{C} \odot \hat{A})}_{mp \times r})^T$$

- $(\hat{C} \odot \hat{A})^T (\hat{C} \odot \hat{A})$ is cheap: it equals $\hat{C}^T \hat{C} * \hat{A}^T \hat{A}$, an $r \times r$ elementwise product — structure paying off immediately
- $X_{(2)}(\hat{C} \odot \hat{A})$ is the Matricized-Tensor Times Khatri-Rao Product (MTTKRP), and it touches every entry of $\mathcal{X}$

MTTKRP is bottleneck number one. $O(I^3 r)$ per mode per iteration.

CP-ALS

Repeat

1. Solve $(B^T B * C^T C)A^T = (X_{(1)}(C \odot B))^T$ for A
2. Solve $(A^T A * C^T C)B^T = (X_{(2)}(C \odot A))^T$ for B
3. Solve $(A^T A * B^T B)C^T = (X_{(3)}(B \odot A))^T$ for C

CP-ALS

Repeat

1. Solve $(B^T B * C^T C)A^T = (X_{(1)}(C \odot B))^T$ for A
2. Solve $(A^T A * C^T C)B^T = (X_{(2)}(C \odot A))^T$ for B
3. Solve $(A^T A * B^T B)C^T = (X_{(3)}(B \odot A))^T$ for C

Other details

- initialization can be random, or more intelligent
- handling the scaling ambiguity within rank-one components
- overlapping computations across the three steps

CP-ALS

Repeat

1. Solve $(B^T B * C^T C)A^T = (X_{(1)}(C \odot B))^T$ for A
2. Solve $(A^T A * C^T C)B^T = (X_{(2)}(C \odot A))^T$ for B
3. Solve $(A^T A * B^T B)C^T = (X_{(3)}(B \odot A))^T$ for C

Other details

- initialization can be random, or more intelligent
- handling the scaling ambiguity within rank-one components
- overlapping computations across the three steps

Simple loop; the entire cost is three MTTKRPs per iteration.

Addition in CP format

Let $\mathcal{X} = [\![\mathrm{A}, \mathrm{B}, \mathrm{C}]\!]$ have rank r and $\mathcal{Y} = \left[\!\!\left[\tilde{\mathrm{A}}, \tilde{\mathrm{B}}, \tilde{\mathrm{C}} \right]\!\!\right]$ have rank $\tilde{r}$. The sum of two sums of rank-one terms is again a sum of rank-one terms, so we simply concatenate the factor matrices column-wise:

$$\mathcal{X} + \mathcal{Y} = \left[\!\!\left[\begin{bmatrix} \mathrm{A} & \tilde{\mathrm{A}} \end{bmatrix}, \begin{bmatrix} \mathrm{B} & \tilde{\mathrm{B}} \end{bmatrix}, \begin{bmatrix} \mathrm{C} & \tilde{\mathrm{C}} \end{bmatrix} \right]\!\!\right], \qquad \text{factors now } I \times (r+\tilde{r}), \text{ etc.}$$

Addition in CP format

Let $\mathcal{X} = [\![\mathsf{A}, \mathsf{B}, \mathsf{C}]\!]$ have rank r and $\mathcal{Y} = \left[\!\!\left[\tilde{\mathsf{A}}, \tilde{\mathsf{B}}, \tilde{\mathsf{C}}\right]\!\!\right]$ have rank $\tilde{r}$. The sum of two sums of rank-one terms is again a sum of rank-one terms, so we simply concatenate the factor matrices column-wise:

$$\mathcal{X} + \mathcal{Y} = \left[\!\!\left[\begin{bmatrix} \mathsf{A} & \tilde{\mathsf{A}} \end{bmatrix}, \begin{bmatrix} \mathsf{B} & \tilde{\mathsf{B}} \end{bmatrix}, \begin{bmatrix} \mathsf{C} & \tilde{\mathsf{C}} \end{bmatrix} \right]\!\!\right], \qquad \text{factors now } I \times (r+\tilde{r}), \text{ etc.}$$

- zero flops — the operation is a copy; storage is the sum of the two storages
- scalar multiples are just as cheap: $\alpha\mathcal{X} = [\![\alpha\mathsf{A}, \mathsf{B}, \mathsf{C}]\!]$, absorb α into any one factor
- the rank of the result is *at most* $r + \tilde{r}$, and generically equal to it

Addition in CP format

Let $\mathcal{X} = [\![\mathrm{A}, \mathrm{B}, \mathrm{C}]\!]$ have rank r and $\mathcal{Y} = [\![\tilde{\mathrm{A}}, \tilde{\mathrm{B}}, \tilde{\mathrm{C}}]\!]$ have rank $\tilde{r}$. The sum of two sums of rank-one terms is again a sum of rank-one terms, so we simply concatenate the factor matrices column-wise:

$$\mathcal{X} + \mathcal{Y} = \left[\!\!\left[\begin{bmatrix} \mathrm{A} & \tilde{\mathrm{A}} \end{bmatrix}, \begin{bmatrix} \mathrm{B} & \tilde{\mathrm{B}} \end{bmatrix}, \begin{bmatrix} \mathrm{C} & \tilde{\mathrm{C}} \end{bmatrix} \right]\!\!\right], \qquad \text{factors now } I \times (r+\tilde{r}), \text{ etc.}$$

- zero flops — the operation is a copy; storage is the sum of the two storages
- scalar multiples are just as cheap: $\alpha\mathcal{X} = [\![\alpha\mathrm{A}, \mathrm{B}, \mathrm{C}]\!]$, absorb α into any one factor
- the rank of the result is *at most* $r + \tilde{r}$, and generically equal to it

The catch: k additions give rank $O(kr)$, and CP has *no* cheap rounding — truncating back to rank r means solving another nonconvex problem with CP-ALS, at $O(I^3 r)$ per mode per iteration, with no guarantee of a good answer.

Inner products in CP format

Everything follows from one fact about rank-one tensors:

$$\langle \mathbf{a} \circ \mathbf{b} \circ \mathbf{c},\ \tilde{\mathbf{a}} \circ \tilde{\mathbf{b}} \circ \tilde{\mathbf{c}} \rangle = (\mathbf{a}^\mathsf{T}\tilde{\mathbf{a}})(\mathbf{b}^\mathsf{T}\tilde{\mathbf{b}})(\mathbf{c}^\mathsf{T}\tilde{\mathbf{c}})$$

Inner products in CP format

Everything follows from one fact about rank-one tensors:

$$\langle \mathsf{a} \circ \mathsf{b} \circ \mathsf{c},\ \tilde{\mathsf{a}} \circ \tilde{\mathsf{b}} \circ \tilde{\mathsf{c}} \rangle = (\mathsf{a}^{\mathsf{T}}\tilde{\mathsf{a}})(\mathsf{b}^{\mathsf{T}}\tilde{\mathsf{b}})(\mathsf{c}^{\mathsf{T}}\tilde{\mathsf{c}})$$

Expanding bilinearly over all $r\tilde{r}$ pairs of terms,

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{\ell=1}^{r} \sum_{\ell'=1}^{\tilde{r}} (\mathsf{a}_\ell^{\mathsf{T}}\tilde{\mathsf{a}}_{\ell'})(\mathsf{b}_\ell^{\mathsf{T}}\tilde{\mathsf{b}}_{\ell'})(\mathsf{c}_\ell^{\mathsf{T}}\tilde{\mathsf{c}}_{\ell'}) = 1^{\mathsf{T}} \left[(\mathsf{A}^{\mathsf{T}}\tilde{\mathsf{A}}) * (\mathsf{B}^{\mathsf{T}}\tilde{\mathsf{B}}) * (\mathsf{C}^{\mathsf{T}}\tilde{\mathsf{C}}) \right] 1$$

- $O((I + J + K)r\tilde{r})$: three thin Gram products, then an $r \times \tilde{r}$ elementwise product — the tensors are never formed
- the same $*$ identity behind the ALS normal equations: $(\mathsf{C} \odot \mathsf{B})^{\mathsf{T}}(\tilde{\mathsf{C}} \odot \tilde{\mathsf{B}}) = \mathsf{C}^{\mathsf{T}}\tilde{\mathsf{C}} * \mathsf{B}^{\mathsf{T}}\tilde{\mathsf{B}}$

Inner products in CP format

Everything follows from one fact about rank-one tensors:

$$\langle \mathsf{a} \circ \mathsf{b} \circ \mathsf{c},\ \tilde{\mathsf{a}} \circ \tilde{\mathsf{b}} \circ \tilde{\mathsf{c}} \rangle = (\mathsf{a}^{\mathsf{T}}\tilde{\mathsf{a}})(\mathsf{b}^{\mathsf{T}}\tilde{\mathsf{b}})(\mathsf{c}^{\mathsf{T}}\tilde{\mathsf{c}})$$

Expanding bilinearly over all $r\tilde{r}$ pairs of terms,

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{\ell=1}^{r} \sum_{\ell'=1}^{\tilde{r}} (\mathsf{a}_\ell^{\mathsf{T}}\tilde{\mathsf{a}}_{\ell'})(\mathsf{b}_\ell^{\mathsf{T}}\tilde{\mathsf{b}}_{\ell'})(\mathsf{c}_\ell^{\mathsf{T}}\tilde{\mathsf{c}}_{\ell'}) = 1^{\mathsf{T}} \left[(\mathsf{A}^{\mathsf{T}}\tilde{\mathsf{A}}) * (\mathsf{B}^{\mathsf{T}}\tilde{\mathsf{B}}) * (\mathsf{C}^{\mathsf{T}}\tilde{\mathsf{C}}) \right] 1$$

- $O((I + J + K)r\tilde{r})$: three thin Gram products, then an $r \times \tilde{r}$ elementwise product — the tensors are never formed
- the same $*$ identity behind the ALS normal equations: $(\mathsf{C} \odot \mathsf{B})^{\mathsf{T}}(\tilde{\mathsf{C}} \odot \tilde{\mathsf{B}}) = \mathsf{C}^{\mathsf{T}}\tilde{\mathsf{C}} * \mathsf{B}^{\mathsf{T}}\tilde{\mathsf{B}}$

Two consequences, letting ALS monitor its *fit* with no residual tensor:

$$\|\mathcal{X}\|^2 = 1^{\mathsf{T}}(\mathsf{A}^{\mathsf{T}}\mathsf{A} * \mathsf{B}^{\mathsf{T}}\mathsf{B} * \mathsf{C}^{\mathsf{T}}\mathsf{C})1, \qquad \|\mathcal{X} - \mathcal{Y}\|^2 = \|\mathcal{X}\|^2 - 2\langle \mathcal{X}, \mathcal{Y} \rangle + \|\mathcal{Y}\|^2$$

Scorecard: CP

$$\mathcal{X} \approx \sum_{i=1}^{R} \mathbf{u}_i \circ \mathbf{v}_i \circ \mathbf{w}_i$$

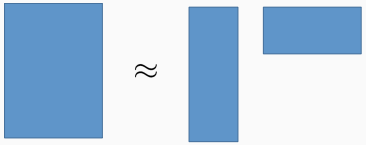
unique / interpretable factors	✔
solves the approximation problem	✘
compresses (storage linear in dimensions)	✔
stable, predictable algorithms	✘

- CP is almost always unique — so you can interpret the components. This is the property the SVD does *not* give you for tensors of order ≥ 3, and it is CP's whole reason to exist
- but there is no exact solve: you cannot reliably hit a prescribed ϵ
- storage is linear in the dimensions: $(I + J + K)R$
- algorithms are iterative and not predictable — run from several starting points and hope

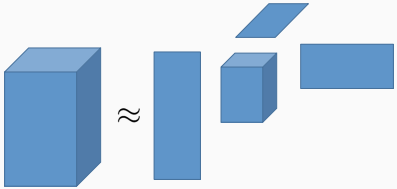
Tucker

Tucker: think of the SVD as identifying subspaces

Matrix:

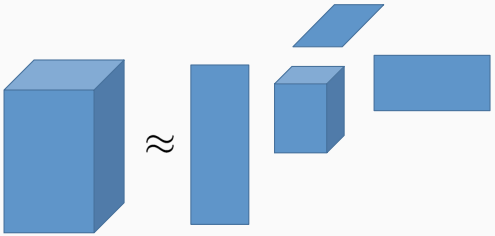


Tensor:



One subspace per mode, plus a small core that couples them.

Tucker notation



$\mathcal{X} \approx \mathcal{G} \times_1 \mathrm{U} \times_2 \mathrm{V} \times_3 \mathrm{W}$

$\mathcal{X} \approx [\![\mathcal{G}; \mathrm{U}, \mathrm{V}, \mathrm{W}]\!]$,

$\mathcal{X} \in \mathbb{R}^{I \times J \times K}, \mathcal{G} \in \mathbb{R}^{P \times Q \times R}$
is the *core tensor*
$\mathrm{U} \in \mathbb{R}^{I \times P}, \mathrm{V} \in \mathbb{R}^{J \times Q}, \mathrm{W} \in \mathbb{R}^{K \times R}$
are *factor matrices*

$$x_{ijk} \approx \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} u_{ip} v_{jq} w_{kr}$$

Note: this is exactly a chain of TTMs. Everything we do to it will be TTMs.

The Tucker optimization problem, and why it is easier

For fixed ranks P, Q, R:

$$\min_{\mathcal{G},\mathsf{U},\mathsf{V},\mathsf{W}} \|\mathcal{X} - \mathcal{G} \times_1 \mathsf{U} \times_2 \mathsf{V} \times_3 \mathsf{W}\|$$

— again nonlinear and nonconvex. But matricize it:

$$\min_{\mathcal{G},\mathsf{U},\mathsf{V},\mathsf{W}} \| \underbrace{\mathsf{X}_{(1)}}_{I \times JK} - \underbrace{\mathsf{U}}_{I \times P} \underbrace{\mathsf{G}_{(1)}(\mathsf{W} \otimes \mathsf{V})^{\mathsf{T}}}_{P \times JK} \|_F$$

The Tucker optimization problem, and why it is easier

For fixed ranks P, Q, R:

$$\min_{\mathcal{G},\mathrm{U},\mathrm{V},\mathrm{W}} \|\mathcal{X} - \mathcal{G} \times_1 \mathrm{U} \times_2 \mathrm{V} \times_3 \mathrm{W}\|$$

— again nonlinear and nonconvex. But matricize it:

$$\min_{\mathcal{G},\mathrm{U},\mathrm{V},\mathrm{W}} \| \underbrace{\mathrm{X}_{(1)}}_{I \times JK} - \underbrace{\mathrm{U}}_{I \times P} \underbrace{\mathrm{G}_{(1)}(\mathrm{W} \otimes \mathrm{V})^{\mathsf{T}}}_{P \times JK} \|_F$$

If we ignore the structure imposed on the $P \times JK$ factor and simply ask for the best rank-P approximation of $\mathrm{X}_{(1)}$, we would take U to be its leading left singular vectors.

That turns out to be good enough — and it decouples the modes completely.

Also note the Kronecker product $\mathrm{W} \otimes \mathrm{V}$ appearing on its own. Remember it too.

Truncated higher-order SVD (T-HOSVD)

1. Compute U = leading P left singular vectors of $\mathsf{X}_{(1)}$
2. Compute V = leading Q left singular vectors of $\mathsf{X}_{(2)}$
3. Compute W = leading R left singular vectors of $\mathsf{X}_{(3)}$
4. $\mathcal{G} = \mathcal{X} \times_1 \mathsf{U}^\mathsf{T} \times_2 \mathsf{V}^\mathsf{T} \times_3 \mathsf{W}^\mathsf{T}$

Quasi-optimality: the T-HOSVD error is within a factor of $\sqrt{N}$ of optimal for the given ranks. No iteration, no local minima, and a computable error bound.

Sequentially truncated variant (ST-HOSVD): truncate the tensor after each factor matrix is computed — cheaper, and equally accurate.

Truncated higher-order SVD (T-HOSVD)

1. Compute U = leading P left singular vectors of $\mathsf{X}_{(1)}$
2. Compute V = leading Q left singular vectors of $\mathsf{X}_{(2)}$
3. Compute W = leading R left singular vectors of $\mathsf{X}_{(3)}$
4. $\mathcal{G} = \mathcal{X} \times_1 \mathsf{U}^\mathsf{T} \times_2 \mathsf{V}^\mathsf{T} \times_3 \mathsf{W}^\mathsf{T}$

Quasi-optimality: the T-HOSVD error is within a factor of $\sqrt{N}$ of optimal for the given ranks. No iteration, no local minima, and a computable error bound.

Sequentially truncated variant (ST-HOSVD): truncate the tensor after each factor matrix is computed — cheaper, and equally accurate.

Bottleneck number two: those SVDs are of $I \times I^{N-1}$ unfoldings.

Addition in Tucker format

Let $\mathcal{X} = [\![\mathcal{G}; \mathrm{U}, \mathrm{V}, \mathrm{W}]\!]$ and $\mathcal{Y} = \left[\!\!\left[\tilde{\mathcal{G}}; \tilde{\mathrm{U}}, \tilde{\mathrm{V}}, \tilde{\mathrm{W}}\right]\!\!\right]$. Concatenate the factors and stack the cores block-diagonally:

$$\mathcal{X} + \mathcal{Y} = \left[\!\!\left[\begin{bmatrix} \mathcal{G} & \\ & \tilde{\mathcal{G}} \end{bmatrix}; \begin{bmatrix} \mathrm{U} & \tilde{\mathrm{U}} \end{bmatrix}, \begin{bmatrix} \mathrm{V} & \tilde{\mathrm{V}} \end{bmatrix}, \begin{bmatrix} \mathrm{W} & \tilde{\mathrm{W}} \end{bmatrix} \right]\!\!\right]$$

The new core is $(P+\tilde{P}) \times (Q+\tilde{Q}) \times (R+\tilde{R})$: $\mathcal{G}$ in the leading block, $\tilde{\mathcal{G}}$ in the trailing one, *zeros elsewhere* — keeping the two sets of subspaces from mixing.

Addition in Tucker format

Let $\mathcal{X} = [\![\mathcal{G}; \mathrm{U}, \mathrm{V}, \mathrm{W}]\!]$ and $\mathcal{Y} = \left[\!\!\left[\tilde{\mathcal{G}}; \tilde{\mathrm{U}}, \tilde{\mathrm{V}}, \tilde{\mathrm{W}}\right]\!\!\right]$. Concatenate the factors and stack the cores block-diagonally:

$$\mathcal{X} + \mathcal{Y} = \left[\!\!\left[\begin{bmatrix} \mathcal{G} & \\ & \tilde{\mathcal{G}} \end{bmatrix}; \begin{bmatrix} \mathrm{U} & \tilde{\mathrm{U}} \end{bmatrix}, \begin{bmatrix} \mathrm{V} & \tilde{\mathrm{V}} \end{bmatrix}, \begin{bmatrix} \mathrm{W} & \tilde{\mathrm{W}} \end{bmatrix} \right]\!\!\right]$$

The new core is $(P+\tilde{P}) \times (Q+\tilde{Q}) \times (R+\tilde{R})$: $\mathcal{G}$ in the leading block, $\tilde{\mathcal{G}}$ in the trailing one, *zeros elsewhere* — keeping the two sets of subspaces from mixing.

- again zero flops: ranks *add* in every mode, so the core *multiplies* — 2^N times bigger for two order-N tensors, mostly zeros
- so, unlike CP, a sum cannot be left unrounded for long

Addition in Tucker format

Let $\mathcal{X} = [\![\mathcal{G}; \mathrm{U}, \mathrm{V}, \mathrm{W}]\!]$ and $\mathcal{Y} = \left[\!\!\left[\tilde{\mathcal{G}}; \tilde{\mathrm{U}}, \tilde{\mathrm{V}}, \tilde{\mathrm{W}}\right]\!\!\right]$. Concatenate the factors and stack the cores block-diagonally:

$$\mathcal{X} + \mathcal{Y} = \left[\!\!\left[\begin{bmatrix} \mathcal{G} & \\ & \tilde{\mathcal{G}} \end{bmatrix}; \begin{bmatrix} \mathrm{U} & \tilde{\mathrm{U}} \end{bmatrix}, \begin{bmatrix} \mathrm{V} & \tilde{\mathrm{V}} \end{bmatrix}, \begin{bmatrix} \mathrm{W} & \tilde{\mathrm{W}} \end{bmatrix} \right]\!\!\right]$$

The new core is $(P+\tilde{P}) \times (Q+\tilde{Q}) \times (R+\tilde{R})$: $\mathcal{G}$ in the leading block, $\tilde{\mathcal{G}}$ in the trailing one, *zeros elsewhere* — keeping the two sets of subspaces from mixing.

- again zero flops: ranks *add* in every mode, so the core *multiplies* — 2^N times bigger for two order-N tensors, mostly zeros
- so, unlike CP, a sum cannot be left unrounded for long

But rounding is cheap here, and touches nothing of size I^N:

1. $\begin{bmatrix} \mathrm{U} & \tilde{\mathrm{U}} \end{bmatrix} = \mathrm{Q}_1\mathrm{R}_1$ and likewise per mode — thin QR, $O(I(P+\tilde{P})^2)$
2. apply the triangular factors to the core, $\mathcal{G} \times_1 \mathrm{R}_1 \times_2 \mathrm{R}_2 \times_3 \mathrm{R}_3$ — TTMs on a small tensor
3. ST-HOSVD that small core and fold the result back into $\mathrm{Q}_1, \mathrm{Q}_2, \mathrm{Q}_3$

Inner products in Tucker format

Vectorize: $\mathrm{vec}(\mathcal{X}) = (\mathsf{W} \otimes \mathsf{V} \otimes \mathsf{U})\,\mathrm{vec}(\mathcal{G})$, then use the mixed-product rule

$$(\mathsf{W} \otimes \mathsf{V} \otimes \mathsf{U})^{\mathsf{T}}(\tilde{\mathsf{W}} \otimes \tilde{\mathsf{V}} \otimes \tilde{\mathsf{U}}) = (\mathsf{W}^{\mathsf{T}}\tilde{\mathsf{W}}) \otimes (\mathsf{V}^{\mathsf{T}}\tilde{\mathsf{V}}) \otimes (\mathsf{U}^{\mathsf{T}}\tilde{\mathsf{U}})$$

Inner products in Tucker format

Vectorize: $\mathrm{vec}(\mathcal{X}) = (W \otimes V \otimes U)\,\mathrm{vec}(\mathcal{G})$, then use the mixed-product rule

$$(W \otimes V \otimes U)^T(\tilde{W} \otimes \tilde{V} \otimes \tilde{U}) = (W^T\tilde{W}) \otimes (V^T\tilde{V}) \otimes (U^T\tilde{U})$$

which, back in tensor notation, says: push all six factor matrices onto one small core

$$\boxed{\langle \mathcal{X}, \mathcal{Y} \rangle = \langle \mathcal{G},\ \tilde{\mathcal{G}} \times_1 (U^T\tilde{U}) \times_2 (V^T\tilde{V}) \times_3 (W^T\tilde{W}) \rangle}$$

Inner products in Tucker format

Vectorize: $\mathrm{vec}(\mathcal{X}) = (\mathsf{W} \otimes \mathsf{V} \otimes \mathsf{U})\,\mathrm{vec}(\mathcal{G})$, then use the mixed-product rule

$$(\mathsf{W} \otimes \mathsf{V} \otimes \mathsf{U})^{\mathsf{T}}(\tilde{\mathsf{W}} \otimes \tilde{\mathsf{V}} \otimes \tilde{\mathsf{U}}) = (\mathsf{W}^{\mathsf{T}}\tilde{\mathsf{W}}) \otimes (\mathsf{V}^{\mathsf{T}}\tilde{\mathsf{V}}) \otimes (\mathsf{U}^{\mathsf{T}}\tilde{\mathsf{U}})$$

which, back in tensor notation, says: push all six factor matrices onto one small core

$$\boxed{\langle \mathcal{X}, \mathcal{Y} \rangle = \langle \mathcal{G},\ \tilde{\mathcal{G}} \times_1 (\mathsf{U}^{\mathsf{T}}\tilde{\mathsf{U}}) \times_2 (\mathsf{V}^{\mathsf{T}}\tilde{\mathsf{V}}) \times_3 (\mathsf{W}^{\mathsf{T}}\tilde{\mathsf{W}}) \rangle}$$

- three Gram products between the factor matrices: $O(IP\tilde{P} + JQ\tilde{Q} + KR\tilde{R})$
- a chain of TTMs on the small core, $O(\tilde{P}\tilde{Q}\tilde{R}P + P\tilde{Q}\tilde{R}Q + PQ\tilde{R}R)$, then a dot product
- the mode sizes I, J, K enter only linearly, through those Gram products

Inner products in Tucker format

Vectorize: $\mathrm{vec}(\mathcal{X}) = (\mathsf{W} \otimes \mathsf{V} \otimes \mathsf{U})\,\mathrm{vec}(\mathcal{G})$, then use the mixed-product rule

$$(\mathsf{W} \otimes \mathsf{V} \otimes \mathsf{U})^{\mathsf{T}}(\tilde{\mathsf{W}} \otimes \tilde{\mathsf{V}} \otimes \tilde{\mathsf{U}}) = (\mathsf{W}^{\mathsf{T}}\tilde{\mathsf{W}}) \otimes (\mathsf{V}^{\mathsf{T}}\tilde{\mathsf{V}}) \otimes (\mathsf{U}^{\mathsf{T}}\tilde{\mathsf{U}})$$

which, back in tensor notation, says: push all six factor matrices onto one small core

$$\boxed{\langle \mathcal{X}, \mathcal{Y} \rangle = \langle \mathcal{G},\ \tilde{\mathcal{G}} \times_1 (\mathsf{U}^{\mathsf{T}}\tilde{\mathsf{U}}) \times_2 (\mathsf{V}^{\mathsf{T}}\tilde{\mathsf{V}}) \times_3 (\mathsf{W}^{\mathsf{T}}\tilde{\mathsf{W}}) \rangle}$$

- three Gram products between the factor matrices: $O(IP\tilde{P} + JQ\tilde{Q} + KR\tilde{R})$
- a chain of TTMs on the small core, $O(\tilde{P}\tilde{Q}\tilde{R}P + P\tilde{Q}\tilde{R}Q + PQ\tilde{R}R)$, then a dot product
- the mode sizes I, J, K enter only linearly, through those Gram products

With orthonormal factors — as T-HOSVD always leaves them — $\mathsf{U}^{\mathsf{T}}\mathsf{U} = \mathsf{I}$ and this collapses to $\|\mathcal{X}\| = \|\mathcal{G}\|$: the norm of a Tucker tensor is the norm of its core.

Scorecard: Tucker

$$\mathcal{X} \approx \mathcal{G} \times_1 \mathsf{U} \times_2 \mathsf{V} \times_3 \mathsf{W}$$

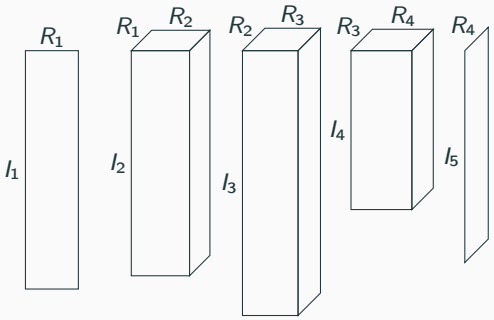
unique / interpretable factors	✗
solves the approximation problem	✓
compresses (storage linear in dimensions)	✓
stable, predictable algorithms	✓

- Tucker is *not* unique — the factors are subspaces, and you cannot interpret individual columns
- but it solves the approximation problem quasi-optimally, and you can hit a prescribed ϵ
- compression is real but not linear in the order: the core is $R_1 \cdots R_N$, so storage still grows exponentially in N
- algorithms are direct, stable and predictable — running time depends only on the dimensions

That third point is the catch, and it is why we need a third format.

Tensor Train

Tensor Train (TT) notation



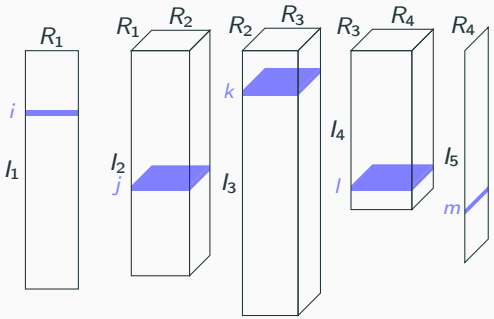
$\mathcal{X} \approx \{\mathcal{T}_{\mathcal{X},k}\}, \mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5}$

$\mathcal{T}_{\mathcal{X},n} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ are *TT cores*

$$x_{ijklm} \approx \sum_{\alpha=1}^{R_1} \sum_{\beta=1}^{R_2} \sum_{\gamma=1}^{R_3} \sum_{\delta=1}^{R_4} \mathcal{T}_{\mathcal{X},1}(i,\alpha)\mathcal{T}_{\mathcal{X},2}(\alpha,j,\beta)\mathcal{T}_{\mathcal{X},3}(\beta,k,\gamma)\mathcal{T}_{\mathcal{X},4}(\gamma,l,\delta)\mathcal{T}_{\mathcal{X},5}(\delta,m)$$

Storage $(I_1 + \cdots + I_N)R^2$: linear in the order. The format for the cookies problem.

Tensor Train (TT) notation



$\mathcal{X} \approx \{\mathcal{T}_{\mathcal{X},k}\}, \mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5}$

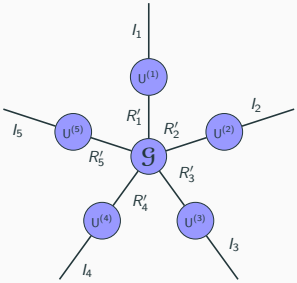
$\mathcal{T}_{\mathcal{X},n} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ are *TT cores*

$$x_{ijklm} \approx \sum_{\alpha=1}^{R_1} \sum_{\beta=1}^{R_2} \sum_{\gamma=1}^{R_3} \sum_{\delta=1}^{R_4} \mathcal{T}_{\mathcal{X},1}(i,\alpha)\mathcal{T}_{\mathcal{X},2}(\alpha,j,\beta)\mathcal{T}_{\mathcal{X},3}(\beta,k,\gamma)\mathcal{T}_{\mathcal{X},4}(\gamma,l,\delta)\mathcal{T}_{\mathcal{X},5}(\delta,m)$$

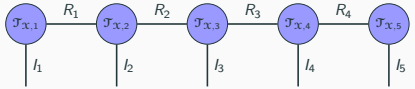
Storage $(I_1 + \cdots + I_N)R^2$: linear in the order. The format for the cookies problem.

Tensor network diagrams

Tucker



Tensor Train



Vertices are tensors, edges are modes, degree is dimension, connected edges are contractions

Tucker has one high-degree vertex (the core, R^N entries); TT has a chain of degree-3 vertices (NIR^2).

Intuition for TT ranks

Tucker: the nth Tucker rank approximates the numerical rank of the nth modal unfolding

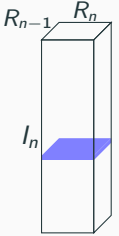
$$\mathsf{X}_{(n)} \quad \text{is} \quad I_n \times (I_1 \cdots I_{n-1} I_{n+1} \cdots I_N)$$

Tensor Train: the nth TT rank approximates the numerical rank of the unfolding that maps the *first* n modes to rows

$$\mathsf{X}_{(1:n)} \quad \text{is} \quad (I_1 \cdots I_n) \times (I_{n+1} \cdots I_N)$$

- so TT asks about a *sequence of splits* of the mode list, not about each mode against all the others
- the full-format tensor would be far too big to store, so $\mathsf{X}_{(1:n)}$ is only ever available implicitly, through the cores
- truncating the ranks of a tensor already in TT format is called TT rounding (Oseledets 2011)

Vertical and horizontal unfoldings of TT cores

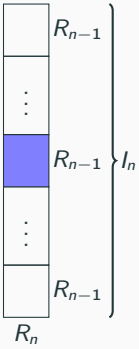


$\mathcal{T}_{\mathfrak{X},n} \in \mathbb{R}^{R_{n-\mathbf{1}} \times I_n \times R_n}$

is a *TT core*

$\mathcal{H}(\mathcal{T}_{\mathfrak{X},n}) \in \mathbb{R}^{R_{n-\mathbf{1}} \times I_n R_n}$

horizontal unfolding



$\mathcal{V}(\mathcal{T}_{\mathfrak{X},n}) \in \mathbb{R}^{R_{n-\mathbf{1}} I_n \times R_n}$

vertical unfolding

Example algorithm: TT rounding

function $\{\mathcal{T}_{\mathcal{Z},n}\} = \text{TT-ROUNDING}(\{\mathcal{T}_{\mathcal{X},n}\})$

▷ Orthogonalization Phase

for $n = N$ down to 2 **do**

$[\mathsf{Y}_n, \mathsf{R}_n] = \mathrm{QR}(\mathcal{H}(\mathcal{T}_{\mathcal{X},n})^T)$ ▷ (tall-skinny) QR factorization

$\mathcal{V}(\mathcal{T}_{\mathcal{X},n-\mathbf{1}}) = \mathcal{V}(\mathcal{T}_{\mathcal{X},n-\mathbf{1}}) \cdot \mathsf{R}^{\mathsf{T}}$ ▷ Apply R to previous core

▷ Truncation Phase

$\mathcal{Z} = \mathcal{X}$

for $n = 1$ to $N - 1$ **do**

$[\mathsf{Y}_n, \mathsf{R}_n] = \mathrm{QR}(\mathcal{V}(\mathcal{T}_{\mathcal{Z},n}))$ ▷ (tall-skinny) QR factorization

$[\hat{\mathsf{U}}_R, \hat{\Sigma}, \hat{\mathsf{V}}] \approx \mathrm{tSVD}(\mathsf{R}_n)$ ▷ Truncated SVD of R

$\mathcal{V}(\mathcal{T}_{\mathcal{Z},n}) = \text{APPLY-Q}(\mathsf{Y}_n, \hat{\mathsf{U}}_R)$

$\mathcal{H}(\mathcal{T}_{\mathcal{Z},n+\mathbf{1}})^T = \text{APPLY-Q}(\mathsf{Y}_{n+\mathbf{1}}, \hat{\mathsf{V}}\hat{\Sigma})$

Both core unfoldings are column major, so BLAS/LAPACK apply directly.

Bottleneck number three: that orthogonalization sweep, $O(NIR^3)$, dominates.

Addition in TT format

Each entry of a TT tensor is a product of matrices, one per mode. To add two of them, make those matrices block-diagonal, so the two chains run side by side without interacting:

$$\mathcal{T}_{\mathcal{Z},n}(:, i_n, :) = \begin{bmatrix} \mathcal{T}_{\mathcal{X},n}(:, i_n, :) & \\ & \mathcal{T}_{\mathcal{Y},n}(:, i_n, :) \end{bmatrix} \in \mathbb{R}^{(R_{n-1}+\tilde{R}_{n-1})\times(R_n+\tilde{R}_n)}, \quad 1 < n < N$$

closing the chain at both ends so the two products are *summed*, not kept apart:

$$\mathcal{T}_{\mathcal{Z},1}(i_1, :) = \begin{bmatrix} \mathcal{T}_{\mathcal{X},1}(i_1, :) & \mathcal{T}_{\mathcal{Y},1}(i_1, :) \end{bmatrix}, \qquad \mathcal{T}_{\mathcal{Z},N}(:, i_N) = \begin{bmatrix} \mathcal{T}_{\mathcal{X},N}(:, i_N) \\ \mathcal{T}_{\mathcal{Y},N}(:, i_N) \end{bmatrix}$$

Addition in TT format

Each entry of a TT tensor is a product of matrices, one per mode. To add two of them, make those matrices block-diagonal, so the two chains run side by side without interacting:

$$\mathcal{T}_{\mathcal{Z},n}(:, i_n, :) = \begin{bmatrix} \mathcal{T}_{\mathcal{X},n}(:, i_n, :) & \\ & \mathcal{T}_{\mathcal{Y},n}(:, i_n, :) \end{bmatrix} \in \mathbb{R}^{(R_{n-\mathbf{1}}+\tilde{R}_{n-\mathbf{1}})\times(R_n+\tilde{R}_n)}, \quad 1 < n < N$$

closing the chain at both ends so the two products are *summed*, not kept apart:

$$\mathcal{T}_{\mathcal{Z},1}(i_1, :) = \begin{bmatrix} \mathcal{T}_{\mathcal{X},1}(i_1, :) & \mathcal{T}_{\mathcal{Y},1}(i_1, :) \end{bmatrix}, \qquad \mathcal{T}_{\mathcal{Z},N}(:, i_N) = \begin{bmatrix} \mathcal{T}_{\mathcal{X},N}(:, i_N) \\ \mathcal{T}_{\mathcal{Y},N}(:, i_N) \end{bmatrix}$$

- multiplying out the block-diagonal chain gives $x_{i_{\mathbf{1}}\cdots i_N} + y_{i_{\mathbf{1}}\cdots i_N}$, as required
- zero flops, pure data movement — but every internal rank adds: $R_n + \tilde{R}_n$

Addition in TT format

Each entry of a TT tensor is a product of matrices, one per mode. To add two of them, make those matrices block-diagonal, so the two chains run side by side without interacting:

$$\mathcal{T}_{\mathcal{Z},n}(:, i_n, :) = \begin{bmatrix} \mathcal{T}_{\mathcal{X},n}(:, i_n, :) & \\ & \mathcal{T}_{\mathcal{Y},n}(:, i_n, :) \end{bmatrix} \in \mathbb{R}^{(R_{n-\mathbf{1}}+\tilde{R}_{n-\mathbf{1}})\times(R_n+\tilde{R}_n)}, \quad 1 < n < N$$

closing the chain at both ends so the two products are *summed*, not kept apart:

$$\mathcal{T}_{\mathcal{Z},1}(i_1, :) = \begin{bmatrix} \mathcal{T}_{\mathcal{X},1}(i_1, :) & \mathcal{T}_{\mathcal{Y},1}(i_1, :) \end{bmatrix}, \qquad \mathcal{T}_{\mathcal{Z},N}(:, i_N) = \begin{bmatrix} \mathcal{T}_{\mathcal{X},N}(:, i_N) \\ \mathcal{T}_{\mathcal{Y},N}(:, i_N) \end{bmatrix}$$

- multiplying out the block-diagonal chain gives $x_{i_{\mathbf{1}}\cdots i_N} + y_{i_{\mathbf{1}}\cdots i_N}$, as required
- zero flops, pure data movement — but every internal rank adds: $R_n + \tilde{R}_n$

This is why TT rounding is the workhorse kernel. Sums are everywhere — time stepping, Krylov methods, residuals — and each doubles the ranks. Rounding is the inner loop.

Inner products in TT format

Contract the two trains together one mode at a time, carrying a small $R_n \times \tilde{R}_n$ matrix M_n from left to right:

$$\mathrm{M}_0 = 1, \qquad \mathrm{M}_n = \sum_{i_n=1}^{I_n} \mathcal{T}_{\mathfrak{X},n}(:, i_n, :)^{\mathsf{T}}\, \mathrm{M}_{n-1}\, \mathcal{T}_{\mathfrak{Y},n}(:, i_n, :), \qquad \langle \mathfrak{X}, \mathfrak{Y} \rangle = \mathrm{M}_N$$

Inner products in TT format

Contract the two trains together one mode at a time, carrying a small $R_n \times \tilde{R}_n$ matrix M_n from left to right:

$$\mathrm{M}_0 = 1, \qquad \mathrm{M}_n = \sum_{i_n=1}^{I_n} \mathcal{T}_{\mathcal{X},n}(:, i_n, :)^{\mathsf{T}}\, \mathrm{M}_{n-1}\, \mathcal{T}_{\mathcal{Y},n}(:, i_n, :), \qquad \langle \mathcal{X}, \mathcal{Y} \rangle = \mathrm{M}_N$$

Each step is two dense matrix products on reshaped cores:

1. $\mathrm{M}_{n-1} \cdot \mathcal{V}(\mathcal{T}_{\mathcal{Y},n})$ costs $O(R_{n-1}\tilde{R}_{n-1} I_n \tilde{R}_n)$
2. contract the result against $\mathcal{T}_{\mathcal{X},n}$, costing $O(R_{n-1} I_n \tilde{R}_n R_n)$

for a total of $O(NIR\tilde{R}(R + \tilde{R}))$ — linear in the order and in the mode sizes, and BLAS-3 throughout.

Inner products in TT format

Contract the two trains together one mode at a time, carrying a small $R_n \times \tilde{R}_n$ matrix M_n from left to right:

$$\mathrm{M}_0 = 1, \qquad \mathrm{M}_n = \sum_{i_n=1}^{I_n} \mathcal{T}_{\mathcal{X},n}(:, i_n, :)^{\mathsf{T}}\, \mathrm{M}_{n-1}\, \mathcal{T}_{\mathcal{Y},n}(:, i_n, :), \qquad \langle \mathcal{X}, \mathcal{Y} \rangle = \mathrm{M}_N$$

Each step is two dense matrix products on reshaped cores:

1. $\mathrm{M}_{n-1} \cdot \mathcal{V}(\mathcal{T}_{\mathcal{Y},n})$ costs $O(R_{n-1}\tilde{R}_{n-1} I_n \tilde{R}_n)$
2. contract the result against $\mathcal{T}_{\mathcal{X},n}$, costing $O(R_{n-1} I_n \tilde{R}_n R_n)$

for a total of $O(NIR\tilde{R}(R + \tilde{R}))$ — linear in the order and in the mode sizes, and BLAS-3 throughout.

Never form the block-diagonal sum and then contract: that would square the ranks. Sweep instead.

Inner products in TT format

Contract the two trains together one mode at a time, carrying a small $R_n \times \tilde{R}_n$ matrix M_n from left to right:

$$\mathrm{M}_0 = 1, \qquad \mathrm{M}_n = \sum_{i_n=1}^{I_n} \mathcal{T}_{\mathcal{X},n}(:, i_n, :)^{\mathsf{T}}\, \mathrm{M}_{n-1}\, \mathcal{T}_{\mathcal{Y},n}(:, i_n, :), \qquad \langle \mathcal{X}, \mathcal{Y} \rangle = \mathrm{M}_N$$

Each step is two dense matrix products on reshaped cores:

1. $\mathrm{M}_{n-1} \cdot \mathcal{V}(\mathcal{T}_{\mathcal{Y},n})$ costs $O(R_{n-1}\tilde{R}_{n-1} I_n \tilde{R}_n)$
2. contract the result against $\mathcal{T}_{\mathcal{X},n}$, costing $O(R_{n-1} I_n \tilde{R}_n R_n)$

for a total of $O(NIR\tilde{R}(R + \tilde{R}))$ — linear in the order and in the mode sizes, and BLAS-3 throughout.

Never form the block-diagonal sum and then contract: that would square the ranks. Sweep instead.

Norms: orthogonalize the cores first (the sweep of QRs from the TT rounding algorithm); then all but one core cancels and $\|\mathcal{X}\|$ is the Frobenius norm of the single remaining core.

Scorecard: Tensor Train

$$x_{ijklm} \approx \mathcal{T}_{\mathcal{X},1}(i,:)\mathcal{T}_{\mathcal{X},2}(:,j,:)\mathcal{T}_{\mathcal{X},3}(:,k,:)\mathcal{T}_{\mathcal{X},4}(:,l,:)\mathcal{T}_{\mathcal{X},5}(:,m)$$

unique / interpretable factors	✗
solves the approximation problem	✓
compresses (storage linear in dimensions)	✓
stable, predictable algorithms	✓

- TT is not unique — you cannot interpret the cores
- quasi-optimal for given ranks, and you can hit a prescribed ϵ
- storage is linear in the order: $(I_1 + \cdots + I_N)R^2$ — this is the box Tucker could not tick, and the reason TT can handle an 11-way tensor
- stable, predictable algorithms

Scorecard: all four, together

	unique	optimal	compresses	stable
SVD	✓	✓	✓	✓
CP	✓	✗	✓	✗
Tucker	✗	✓	✓	✓
TT	✗	✓	✓	✓

- **want to interpret the factors?** CP is the only option, and you pay for it in stability
- **want a guaranteed error at a given cost?** Tucker or TT
- **order 3 or 4, and want the modes treated symmetrically?** Tucker
- **order 10 or more?** TT, because it is the only one whose storage does not blow up with the order

No format wins. Pick the column you cannot do without.

Three formats, three bottlenecks — and one shape

format	algorithm	bottleneck	cost
CP	ALS	MTTKRP / the LLS solve	$O(I^N r)$ per mode, per iteration
Tucker	ST-HOSVD	SVD of the modal unfoldings	$O(I^{N+1})$
TT	rounding	orthogonalization sweep	$O(NIR^3)$

Three formats, three bottlenecks — and one shape

format	algorithm	bottleneck	cost
CP	ALS	MTTKRP / the LLS solve	$O(I^N r)$ per mode, per iteration
Tucker	ST-HOSVD	SVD of the modal unfoldings	$O(I^{N+1})$
TT	rounding	orthogonalization sweep	$O(NIR^3)$

Every one of them has the same shape:

a huge, *implicitly defined* matrix, and we only want its range, or a least-squares solution against it

- CP: the coefficient matrix is a Khatri-Rao product, $I^{N-1} \times r$
- Tucker: the unfolding $\mathsf{X}_{(n)}$ is $I \times I^{N-1}$
- TT: the unfolding $\mathsf{X}_{(1:n)}$ exists only through the cores

That is precisely the situation randomization was invented for. After the break: why the obvious way to randomize does not work, and what does.

Break

10 minutes

Randomization, and why it needs structure

Randomized rank-k decomposition

Randomized range finder (Halko, Martinsson & Tropp 2011). To find an approximate basis for the range of $\mathsf{A} \in \mathbb{R}^{m \times n}$:

$$\mathsf{Y} = \mathsf{A}\Omega, \qquad \Omega \in \mathbb{R}^{n \times \ell} \text{ random}, \qquad \mathsf{Q} = \operatorname{orth}(\mathsf{Y})$$

with $\ell = r+$ a small oversampling parameter. Then $\mathsf{A} \approx \mathsf{Q}\mathsf{Q}^{\mathsf{T}}\mathsf{A}$.

Work out how you apply it to $A = XZ^T$ and $A = X_1 Z_1^T + X_2 Z_2^T$.

Why we cannot just use a Gaussian

Take an N-way tensor with all modes I. The mode-n unfolding is $I \times I^{N-1}$, so a dense Ω is $I^{N-1} \times \ell$.

	tensor I^N	dense Ω $I^{N-1}\ell$
I=100, N=3, ℓ=20	10^6	2×10^5
I=100, N=4, ℓ=20	10^8	2×10^7
I=100, N=5, ℓ=20	10^{10}	2×10^9
I=100, N=8, ℓ=20	10^{16}	2×10^{15}

The sketch is a constant fraction of the tensor — at every order.

And that is the *optimistic* reading. In the cases we actually care about:

- the tensor is never stored in full anyway — it lives in TT or Tucker form, and is far smaller than I^N
- so the random matrix would be orders of magnitude larger than the data it is compressing
- and forming $A\Omega$ would cost more than the deterministic algorithm we set out to replace

Why we cannot just use a Gaussian

Take an N-way tensor with all modes I. The mode-n unfolding is $I \times I^{N-1}$, so a dense Ω is $I^{N-1} \times \ell$.

	tensor I^N	dense Ω $I^{N-1}\ell$
I=100, N=3, ℓ=20	10^6	2×10^5
I=100, N=4, ℓ=20	10^8	2×10^7
I=100, N=5, ℓ=20	10^{10}	2×10^9
I=100, N=8, ℓ=20	10^{16}	2×10^{15}

The sketch is a constant fraction of the tensor — at every order.

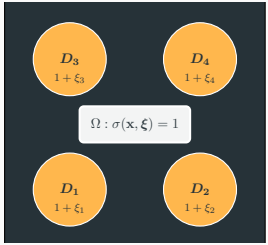
And that is the *optimistic* reading. In the cases we actually care about:

- the tensor is never stored in full anyway — it lives in TT or Tucker form, and is far smaller than I^N
- so the random matrix would be orders of magnitude larger than the data it is compressing
- and forming $A\Omega$ would cost more than the deterministic algorithm we set out to replace

We need an Ω that is never formed and never stored — only *applied* in a way exploiting the structure the data already has.

Three ways to do that, one per format.

Where rounding actually comes from



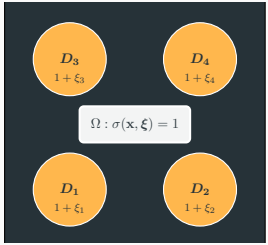
the cookies problem

Discretize the parameter-dependent PDE in space *and* in all p parameters at once and the linear system inherits the tensor structure:

$$\Big(\mathsf{A}_0 \otimes \mathsf{I} \otimes \cdots \otimes \mathsf{I} \;+\; \sum_{i=1}^{p} \mathsf{A}_i \otimes \mathsf{I} \otimes \cdots \otimes \underbrace{\mathsf{D}_i}_{\text{mode } i+1} \otimes \cdots \otimes \mathsf{I}\Big)\, \mathrm{vec}(\mathcal{U}) = \mathsf{f} \otimes 1 \otimes \cdots \otimes 1$$

with $\mathsf{D}_i = \mathrm{diag}(\rho_i^{(1)}, \ldots, \rho_i^{(I)})$, I samples per parameter: a rank-$(p{+}1)$ operator, a rank-one right-hand side, and an unknown $\mathcal{U}$ of order $N = p{+}1$ that is never formed.

Where rounding actually comes from



the cookies problem

Discretize the parameter-dependent PDE in space *and* in all p parameters at once and the linear system inherits the tensor structure:

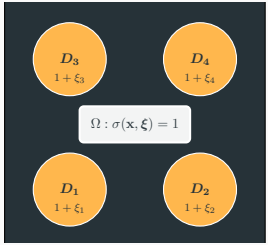
$$\Big(\mathsf{A}_0 \otimes \mathsf{I} \otimes \cdots \otimes \mathsf{I} + \sum_{i=1}^{p} \mathsf{A}_i \otimes \mathsf{I} \otimes \cdots \otimes \underbrace{\mathsf{D}_i}_{\text{mode } i+1} \otimes \cdots \otimes \mathsf{I}\Big) \operatorname{vec}(\mathcal{U}) = \mathsf{f} \otimes 1 \otimes \cdots \otimes 1$$

with $\mathsf{D}_i = \operatorname{diag}(\rho_i^{(1)}, \ldots, \rho_i^{(I)})$, I samples per parameter: a rank-(p+1) operator, a rank-one right-hand side, and an unknown $\mathcal{U}$ of order $N = p+1$ that is never formed.

Solve it with low-rank GMRES (TT-GMRES, Dolgov 2013): every Krylov vector is a low-rank tensor, and every step inflates the ranks:

- the matvec $\mathcal{W} = \mathcal{A}\mathcal{V}_j$ multiplies the ranks by p+1
- Arnoldi orthogonalization $\mathcal{W} \leftarrow \mathcal{W} - \sum_{i \le j} h_{ij}\mathcal{V}_i$ is a sum of j+1 low-rank tensors — ranks *add*; so is the update $\mathcal{X} = \sum_k y_k \mathcal{V}_k$

Where rounding actually comes from



the cookies problem

Discretize the parameter-dependent PDE in space *and* in all p parameters at once and the linear system inherits the tensor structure:

$$\Big(\mathsf{A}_0 \otimes \mathsf{I} \otimes \cdots \otimes \mathsf{I} + \sum_{i=1}^{p} \mathsf{A}_i \otimes \mathsf{I} \otimes \cdots \otimes \underbrace{\mathsf{D}_i}_{\text{mode } i+1} \otimes \cdots \otimes \mathsf{I}\Big)\, \mathrm{vec}(\mathcal{U}) = \mathsf{f} \otimes 1 \otimes \cdots \otimes 1$$

with $\mathsf{D}_i = \mathrm{diag}(\rho_i^{(1)}, \ldots, \rho_i^{(I)})$, I samples per parameter: a rank-(p+1) operator, a rank-one right-hand side, and an unknown $\mathcal{U}$ of order $N = p+1$ that is never formed.

Solve it with low-rank GMRES (TT-GMRES, Dolgov 2013): every Krylov vector is a low-rank tensor, and every step inflates the ranks:

- the matvec $\mathcal{W} = \mathcal{A}\mathcal{V}_j$ multiplies the ranks by p+1
- Arnoldi orthogonalization $\mathcal{W} \leftarrow \mathcal{W} - \sum_{i \le j} h_{ij}\mathcal{V}_i$ is a sum of j+1 low-rank tensors — ranks *add*; so is the update $\mathcal{X} = \sum_k y_k \mathcal{V}_k$

Additions were free; this is where the bill arrives. Each must be truncated back before the next step — and the kernel is always the same: round a sum of low-rank tensors.

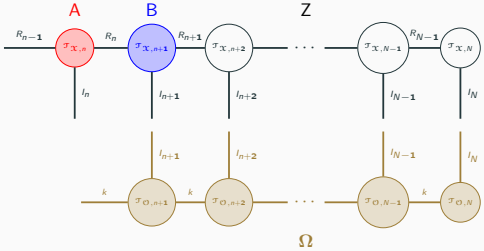
Family 1: tensor-train

Family 1: random matrices whose columns are tensor trains

For TT rounding we must sketch $\mathsf{X}_{(1:n)}$, which is $(I_1 \cdots I_n) \times (I_{n+1} \cdots I_N)$ and exists only implicitly. So give Ω the same structure the data has: build it from its own cores $\{\mathcal{T}_{\Omega,n+1}, \ldots, \mathcal{T}_{\Omega,N}\}$, each of rank k.

$$\Omega \in \mathbb{R}^{(I_{n+1} \cdots I_N) \times k}$$

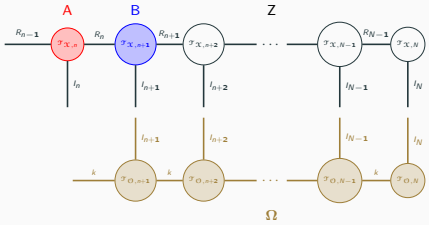
- stored in $O\left(\sum_j I_j k^2\right)$
- each *column* of Ω is itself a tensor in TT format



the data $\mathcal{X}$ (top) and the random Ω (bottom), as one network

Why this is cheap: the sketch is a contraction

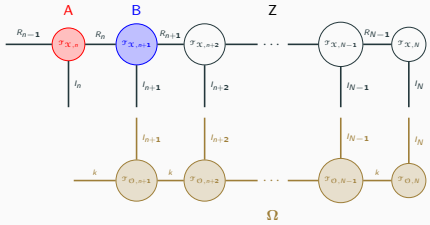
$X_{(1:n)}\Omega$ is not a matrix product we evaluate. It is a partial contraction of two tensor networks — read the picture from the right:



- contract $\mathcal{T}_{\mathcal{X},N}$ with $\mathcal{T}_{\mathcal{O},N}$ over mode I_N: a small $R_{N-1} \times k$ matrix
- absorb it into $\mathcal{T}_{\mathcal{X},N-1}$, contract again, and sweep leftwards
- total cost $O(NIR^2k)$ — linear in everything, and the unfolding is never formed

Why this is cheap: the sketch is a contraction

$X_{(1:n)}\Omega$ is not a matrix product we evaluate. It is a partial contraction of two tensor networks — read the picture from the right:



- contract $\mathcal{T}_{\mathcal{X},N}$ with $\mathcal{T}_{\mathcal{O},N}$ over mode I_N: a small $R_{N-1} \times k$ matrix
- absorb it into $\mathcal{T}_{\mathcal{X},N-1}$, contract again, and sweep leftwards
- total cost $O(NIR^2k)$ — linear in everything, and the unfolding is never formed

And the partial contractions are reused. The sweep that produces the sketch for mode n also produces the sketches for every other mode along the way: *sketch once, sweep once.*

Randomized TT rounding

TT rounding is bottlenecked by the orthogonalization sweep.

Matrix idea: randomized SVD (Halko, Martinsson & Tropp 2011)

- randomized range finder with an overestimated rank
- deterministic SVD on the projection to get the estimated rank

Tensor idea: TT-Rounding-RandOrth (Al Daas, Ballard, Saibaba et al. 2023)

- apply the randomized SVD to the tensor unfoldings
- use TT-structured random matrices, so each sketch is a contraction
- then deterministically round the TT with overestimated ranks, as in RandSVD

Replaces the $O(NIR^3)$ orthogonalization with $O(NIR^2k)$ contractions.

Randomized TT rounding

TT rounding is bottlenecked by the orthogonalization sweep.

Matrix idea: randomized SVD (Halko, Martinsson & Tropp 2011)

- randomized range finder with an overestimated rank
- deterministic SVD on the projection to get the estimated rank

Tensor idea: TT-Rounding-RandOrth (Al Daas, Ballard, Saibaba et al. 2023)

- apply the randomized SVD to the tensor unfoldings
- use TT-structured random matrices, so each sketch is a contraction
- then deterministically round the TT with overestimated ranks, as in RandSVD

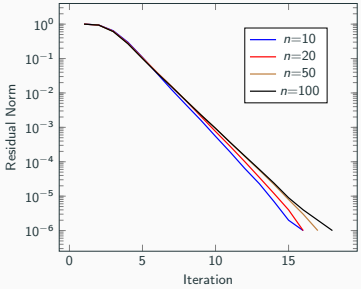
Replaces the $O(NIR^3)$ orthogonalization with $O(NIR^2k)$ contractions.

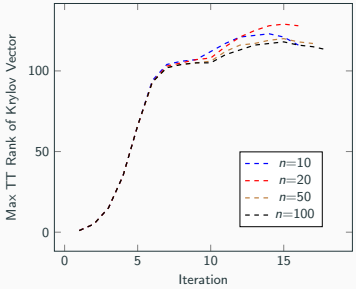
Same caveat as the Kronecker case, and more so: the entries of a TT-structured Ω are dependent, and the embedding constants degrade with both the sketch rank k and the order N. Oversample; the error is checkable afterwards.

The cookies problem

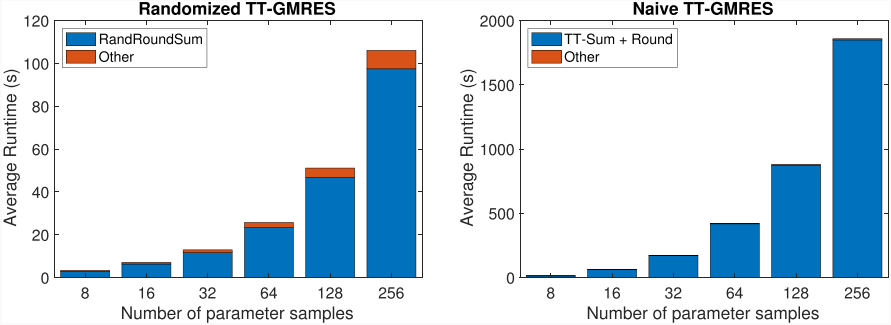
Solve the cookies problem with 4 cookies, on a tensor of dimension $1781 \times n \times n \times n \times n$, with TT-GMRES (Dolgov 2013).

- a Krylov method in which every vector is a TT tensor
- full tensors are never explicitly formed
- the bottleneck is *rounding* the TT tensors after each operation





Randomized TT rounding: timing on the cookies problem



randomized TT-rounding, for varying numbers of parameters

Family 2: Khatri-Rao

Family 2: Khatri-Rao structure

Low-rank GMRES keeps handing us $\mathcal{Y} = \sum_{k=1}^{K} \alpha_k \mathcal{X}_k$, each summand a TT tensor. Sketch it with a Khatri-Rao product of small Gaussians, $\Omega = \Omega_N \odot \cdots \odot \Omega_2$ with $\Omega_n \in \mathbb{R}^{I_n \times \ell}$: storage $\ell \sum_n I_n$, every column a random rank-one tensor.

Family 2: Khatri-Rao structure

Low-rank GMRES keeps handing us $\mathcal{Y} = \sum_{k=1}^{K} \alpha_k \mathcal{X}_k$, each summand a TT tensor. Sketch it with a Khatri-Rao product of small Gaussians, $\Omega = \Omega_N \odot \cdots \odot \Omega_2$ with $\Omega_n \in \mathbb{R}^{I_n \times \ell}$: storage $\ell \sum_n I_n$, every column a random rank-one tensor.

Applying it is a right-to-left sweep of partial contractions, one MTTKRP per core:

$$\mathrm{W}_N = \mathcal{H}(\mathcal{T}_{\mathcal{X},N})\,\Omega_N, \qquad \mathrm{W}_n = \mathcal{H}(\mathcal{T}_{\mathcal{X},n})\big[\mathrm{W}_{n+1} \odot \Omega_n\big], \quad n = N{-}1, \ldots, 2$$

- randomize-then-orthogonalize exactly as in Family 1: $\mathrm{S}_n = \mathcal{V}(\mathcal{T}_{\mathcal{Y},n})\mathrm{W}_{n+1}(:, 1{:}\ell_n)$, thin QR, new core; $O(NIR^2\ell)$ in total
- the sketch of the sum is the sum of the sketches — contract each $\mathcal{X}_k$ and accumulate, so the block-diagonal formal sum is never built

Family 2: Khatri-Rao structure

Low-rank GMRES keeps handing us $\mathcal{Y} = \sum_{k=1}^{K} \alpha_k \mathcal{X}_k$, each summand a TT tensor. Sketch it with a Khatri-Rao product of small Gaussians, $\Omega = \Omega_N \odot \cdots \odot \Omega_2$ with $\Omega_n \in \mathbb{R}^{I_n \times \ell}$: storage $\ell \sum_n I_n$, every column a random rank-one tensor.

Applying it is a right-to-left sweep of partial contractions, one MTTKRP per core:

$$W_N = \mathcal{H}(\mathcal{T}_{\mathcal{X},N})\,\Omega_N, \qquad W_n = \mathcal{H}(\mathcal{T}_{\mathcal{X},n})\big[W_{n+1} \odot \Omega_n\big], \quad n = N{-}1, \ldots, 2$$

- randomize-then-orthogonalize exactly as in Family 1: $S_n = \mathcal{V}(\mathcal{T}_{\mathcal{Y},n})W_{n+1}(:, 1{:}\ell_n)$, thin QR, new core; $O(NIR^2\ell)$ in total
- the sketch of the sum is the sum of the sketches — contract each $\mathcal{X}_k$ and accumulate, so the block-diagonal formal sum is never built

Same leading order as the TT-structured sketch, with $O(NI\ell)$ random numbers instead of $O(NI\ell^2)$ — and this sketch can be widened after the fact.

Paying for a weaker embedding: adapt the sketch size

A KRP of N Gaussians is not a Gaussian: $\frac{1}{\ell}\|A\Omega\|_F^2$ is still unbiased for $\|A\|_F^2$, but the guarantee costs more:

$$\ell \gtrsim \max\left\{\varepsilon^{-1} C''_\alpha \ln^N(2/\delta),\ \varepsilon^{-2} C'_\alpha \ln^2(2/\delta)\right\} \quad \text{with probability } 1-\delta$$

That $\ln^N$ is the price of the structure — *prescribing ℓ up front is guesswork.*

Paying for a weaker embedding: adapt the sketch size

A KRP of N Gaussians is not a Gaussian: $\frac{1}{\ell}\|A\Omega\|_F^2$ is still unbiased for $\|A\|_F^2$, but the guarantee costs more:

$$\ell \gtrsim \max\left\{\varepsilon^{-1} C''_\alpha \ln^N(2/\delta),\ \varepsilon^{-2} C'_\alpha \ln^2(2/\delta)\right\} \quad \text{with probability } 1-\delta$$

That $\ln^N$ is the price of the structure — *prescribing ℓ up front is guesswork.*

Grow it instead. Per mode, start from b_{init} columns and enrich in blocks

- the stopping test is that same estimator: use the enriching block to estimate the residual $\|X - QQ^T X\|_F$
- enriching is cheap *because* the sketch is a KRP: draw new Ω_n and append $W_n \leftarrow [W_n \;\; W_n^{\text{new}}]$, keeping every contraction already done

Paying for a weaker embedding: adapt the sketch size

A KRP of N Gaussians is not a Gaussian: $\frac{1}{\ell}\|A\Omega\|_F^2$ is still unbiased for $\|A\|_F^2$, but the guarantee costs more:

$$\ell \gtrsim \max\left\{\varepsilon^{-1} C''_\alpha \ln^N(2/\delta),\ \varepsilon^{-2} C'_\alpha \ln^2(2/\delta)\right\} \quad \text{with probability } 1-\delta$$

That $\ln^N$ is the price of the structure — *prescribing ℓ up front is guesswork.*

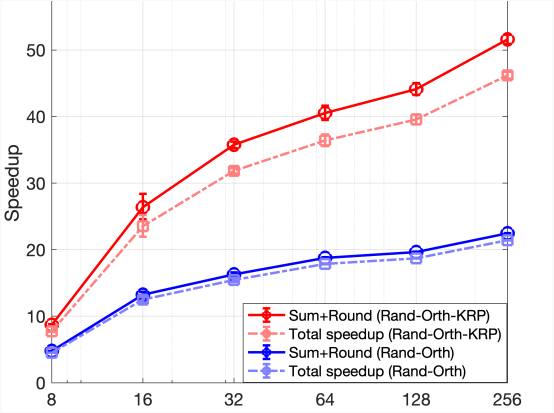
Grow it instead. Per mode, start from b_{init} columns and enrich in blocks

- the stopping test is that same estimator: use the enriching block to estimate the residual $\|X - QQ^T X\|_F$
- enriching is cheap *because* the sketch is a KRP: draw new Ω_n and append $W_n \leftarrow [W_n \ \ W_n^{\text{new}}]$, keeping every contraction already done

Adaptivity is what the Khatri-Rao structure buys: very cheap to append columns compared to a TT-format.

Adaptive KRP rounding: back to the cookies problem

TT-GMRES on 4 cookies, with *sum-then-round* done three ways:



red: KRP, blue: TT sketch; axis: mode size I

- $51\times$ on the sum-and-round kernel and $46\times$ on the whole solve
- $2\times$ faster than the Family 1 TT-structured sketch — at the same TT ranks per iteration
- Using 7 cookies, $N = 8$: the fixed-rank randomized solver stalls at residual 10^{-2}; the adaptive one reaches 10^{-7} against a 10^{-8} target, $25\times$ faster than naive — which runs out of memory at $I = 128$

Al Daas, Ballard, Grigori, Martinez Aguilar, Saibaba & Verma, *Adaptive randomized tensor train rounding using Khatri-Rao products*.

Family 3: Kronecker

Family 3: Kronecker-structured random matrices

Build Ω from one small random matrix per mode, $\Omega_j \in \mathbb{R}^{\ell_j \times I_j}$:

$$\Omega = (\Omega_N \otimes \cdots \otimes \Omega_{j+1} \otimes \Omega_{j-1} \otimes \cdots \otimes \Omega_1)$$

Storage

$$\sum_{i \neq j} I_i \ell_i \quad \text{instead of} \quad \prod_{i \neq j} I_i \ell_i$$

For $I = 100$, $N = 5$, $\ell = 20$: 8×10^3 instead of 2×10^9.

Application — the key identity

$$Y_j = X_{(j)} \, \Omega^{\mathsf{T}}$$

is exactly a multi-TTM:

$$\mathcal{Y} = \mathcal{X} \times_1 \Omega_1 \cdots \times_{j-1} \Omega_{j-1} \times_{j+1} \Omega_{j+1} \cdots \times_N \Omega_N$$

Family 3: Kronecker-structured random matrices

Build Ω from one small random matrix per mode, $\Omega_j \in \mathbb{R}^{\ell_j \times I_j}$:

$$\Omega = (\Omega_N \otimes \cdots \otimes \Omega_{j+1} \otimes \Omega_{j-1} \otimes \cdots \otimes \Omega_1)$$

Storage

$$\sum_{i \neq j} I_i \ell_i \quad \text{instead of} \quad \prod_{i \neq j} I_i \ell_i$$

For $I = 100$, $N = 5$, $\ell = 20$: 8×10^3 instead of 2×10^9.

Application — the key identity

$$Y_j = X_{(j)} \Omega^T$$

is exactly a multi-TTM:

$$\mathcal{Y} = \mathcal{X} \times_1 \Omega_1 \cdots \times_{j-1} \Omega_{j-1} \times_{j+1} \Omega_{j+1} \cdots \times_N \Omega_N$$

So sketching costs TTM flops, using the operation we discussed earlier — and we never form Ω, never form $X_{(j)}$.

Better still: the partial products $\mathcal{X} \times_1 \Omega_1 \times_2 \Omega_2 \cdots$ are *shared* between the sketches for different modes j.

Randomized T-HOSVD

T-HOSVD is bottlenecked by the SVDs of the unfoldings.

Matrix idea: randomized range finder (Halko, Martinsson & Tropp 2011)

- post-multiply by a random matrix to approximate the range
- need an estimate of the rank, plus oversampling

Tensor idea: RandTucker (Minster, Li & Ballard 2022; Minster, Saibaba & Kilmer 2020)

1. Compute U = orthonormalization of $\mathcal{X} \times_2 \Omega_2 \times_3 \Omega_3$
2. Compute V = orthonormalization of $\mathcal{X} \times_1 \Omega_1 \times_3 \Omega_3$
3. Compute W = orthonormalization of $\mathcal{X} \times_1 \Omega_1 \times_2 \Omega_2$
4. $\mathcal{G} = \mathcal{X} \times_1 \mathrm{U}^\mathsf{T} \times_2 \mathrm{V}^\mathsf{T} \times_3 \mathrm{W}^\mathsf{T}$

Every line is a TTM. Parallelizable, cache-friendly, and the sequentially truncated variant randomizes the same way.

The catch: a Kronecker sketch is a weaker embedding

A Kronecker Ω has $\prod_i l_i \ell_i$ entries but only $\sum_i l_i \ell_i$ *degrees of freedom*. Its entries are heavily dependent, and the embedding guarantee degrades accordingly.

What changes

- for a Gaussian, $\ell = O(r/\epsilon^2)$
- for a Kronecker product of N factors, the required ℓ picks up a dependence on N — the known bounds degrade roughly exponentially in the number of factors
- the tail/concentration behaviour is heavier: rare bad draws are less rare

What to do about it

- oversample per mode, not globally: take $\ell_j = R_j + p$ with p a genuine margin, not $p = 5$
- the bounds are pessimistic — in practice these sketches behave far better than the theory promises
- and you can always *verify*: the Tucker error is computable after the fact

This is the honest trade. We buy a factor of 10^5 in storage and pay for it in sketch size and in theory. It is nearly always worth it — but know that you are paying.

Wrapping up

The three families, side by side

structure of Ω	where it arises	storage	cost to apply	embedding
dense Gaussian	— (the ideal)	$I^{N-\mathbf{1}}\ell$	$O(I^N\ell)$	$\ell = O(r/\epsilon^{\mathbf{2}})$
TT columns $\{\mathcal{T}_{\mathfrak{O},n}\}$, rank k	TT rounding: sketching $\mathsf{X}_{(\mathbf{1}:n)}$	$O(\sum_j I_j k^{\mathbf{2}})$	$O(NIR^{\mathbf{2}}k)$ contraction; one sweep serves all n	degrades with k and N; k is the dial
Khatri-Rao $\Omega_N \odot \cdots \odot \Omega_{\mathbf{2}}$	rounding a sum of TT tensors, to a *tolerance*	$\ell \sum_n I_n$	$O(NIR^{\mathbf{2}}\ell)$ partial contractions; $O(NI\ell)$ RNGs	weakest of the three ($\ln^N$), but the sketch can grow
Kronecker $\Omega_N \otimes \cdots \otimes \Omega_{\mathbf{1}}$	Tucker: sketching modal unfoldings $\mathsf{X}_{(j)}$	$\sum_i I_i\ell_i$	multi-TTM; partial products shared across modes	degrades with the number of factors

One idea throughout: give the random matrix the same structure as the data, so that "multiply by Ω" becomes an operation the format already supports.