

Tensor Train Approximations

Riemannian Methods, Randomized Linear Algebra
and Applications to Machine Learning

Thesis defense of **Willem Hendrik (Rik) Voorhaar** for obtaining the title of Doctorat ès sciences
Thesis advisor: Prof. **Bart Vandereycken**

Jury: Dr. Gilles Vilmart, Prof. Lars Grasedyck, Prof. Sergey Dolgov



UNIVERSITÉ
DE GENÈVE

FACULTÉ DES SCIENCES
Section de mathématiques

Part 1 | Low-rank matrices



§1 | Low-rank matrices

A matrix is low-rank if it can be written as a product of two smaller matrices.

$$m \left\{ \begin{matrix} A \\ \underbrace{}_{n} \end{matrix} \right\} = m \left\{ \begin{matrix} X \\ \underbrace{}_r \end{matrix} \right\} \times \left\{ \begin{matrix} Y^\top \\ \underbrace{}_n \end{matrix} \right\}_r$$

A has rank $\leq r \Leftrightarrow A = XY^\top$ with X size $m \times r$ and Y size $n \times r$.

§1 | Low-rank matrices

A matrix is low-rank if it can be written as a product of two smaller matrices.

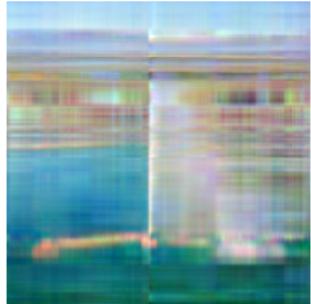
$$m \underbrace{\begin{Bmatrix} A \end{Bmatrix}}_{n} = m \underbrace{\begin{Bmatrix} X \end{Bmatrix}}_r \times \underbrace{\begin{Bmatrix} Y^\top \end{Bmatrix}}_n r$$

A has rank $\leq r \Leftrightarrow A = XY^\top$ with X size $m \times r$ and Y size $n \times r$.

Many matrices are not low-rank, but can be approximated by low-rank matrices.

Example: We can see an image as a matrix, and do a low-rank approximation

rank = 5



rank = 10



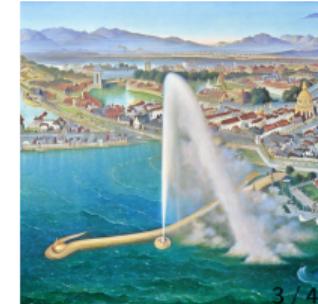
rank = 20



rank = 50

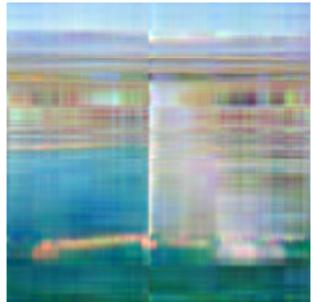


rank = 768



§1 | Low-rank matrices

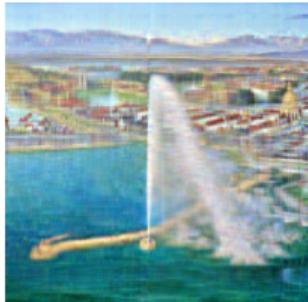
rank = 5



rank = 10



rank = 20



rank = 50



rank = 768



In general, a best low-rank approximation of A solves

$$\min_{\substack{B \\ \text{rank } \leq r}} \|A - B\|$$

Solution: truncated SVD $A = U\Sigma V^\top$, then $B = U_r \Sigma_r V_r^\top$ where:

Matrix	Size	Property
U_r	$m \times r$	$U_r^\top U_r = I_r$
V_r	$n \times r$	$V_r^\top V_r = I_r$
Σ_r	$r \times r$	Diagonal

§1 | Matrix completion

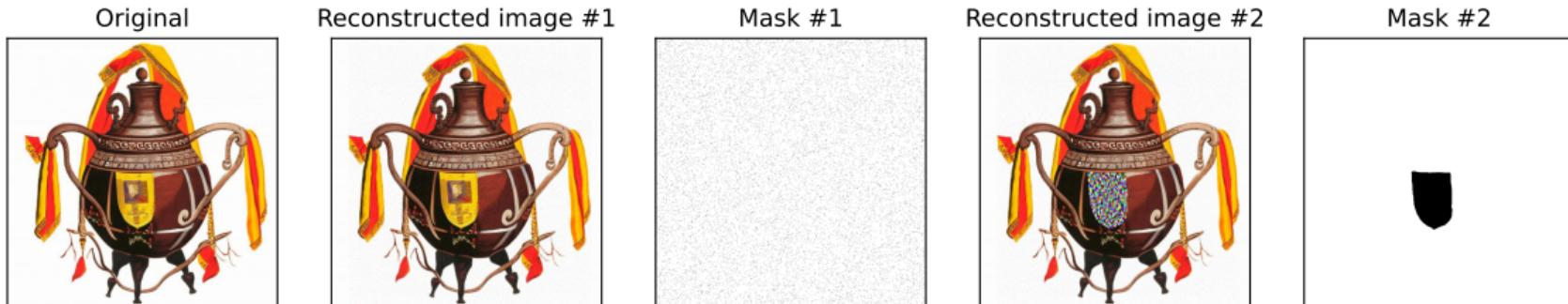
A rank $\leq r$ matrix uses $r(m + n) - r^2 < mn$ parameters. \Rightarrow we can find a rank $\leq r$ approximation of A without knowing all the entries.

Optimization problem:

$$\min_{B \text{ rank } \leq r} \|\mathcal{P}_\Omega A - \mathcal{P}_\Omega B\|,$$

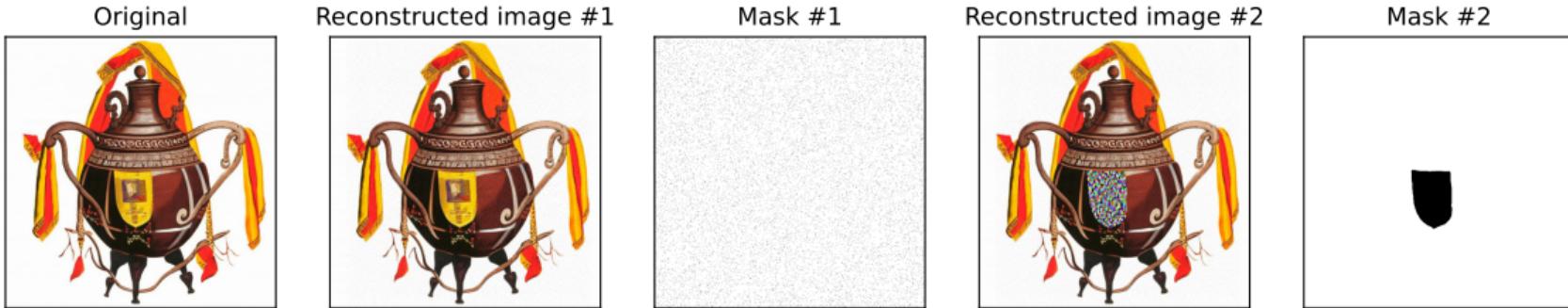
where \mathcal{P}_Ω projection onto known entries indexed by $\Omega \subset \{1, \dots, m\} \times \{1, \dots, n\}$.

Example: Image with 2.7% of points removed, restoration using rank 100.



§1 | Matrix completion

Example: Image with 2.7% of points removed, restoration using rank 100.



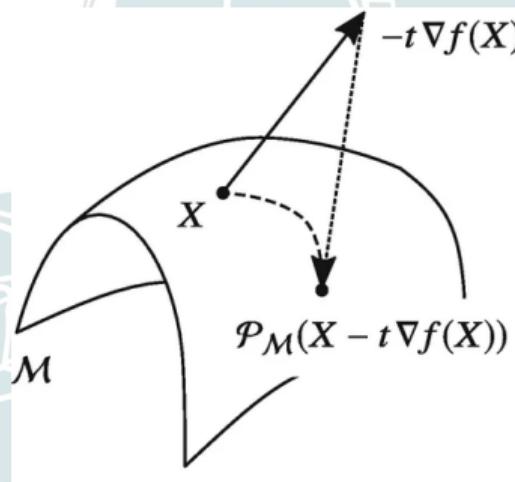
Techniques:

- ▶ Alternating least squares: Write $B = XY^T$ and alternatingly optimize for X and Y .
- ▶ Convex optimization: Remove constraints and replace $\|\cdot\|$ by nuclear norm.
- ▶ Riemannian gradient descent: Constrained optimization on the manifold of rank $\leq r$ matrices.

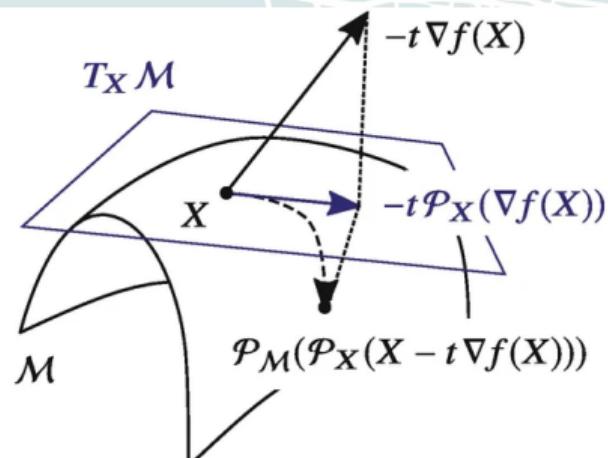
§1 | Riemannian gradient descent

Manifold-constrained optimization, two approaches:

- ▶ Perform step in gradient direction \rightarrow project down onto constrain set. Or:
- ▶ Project gradient onto tangent space \rightarrow perform step in projected direction \rightarrow project down onto constraint set.



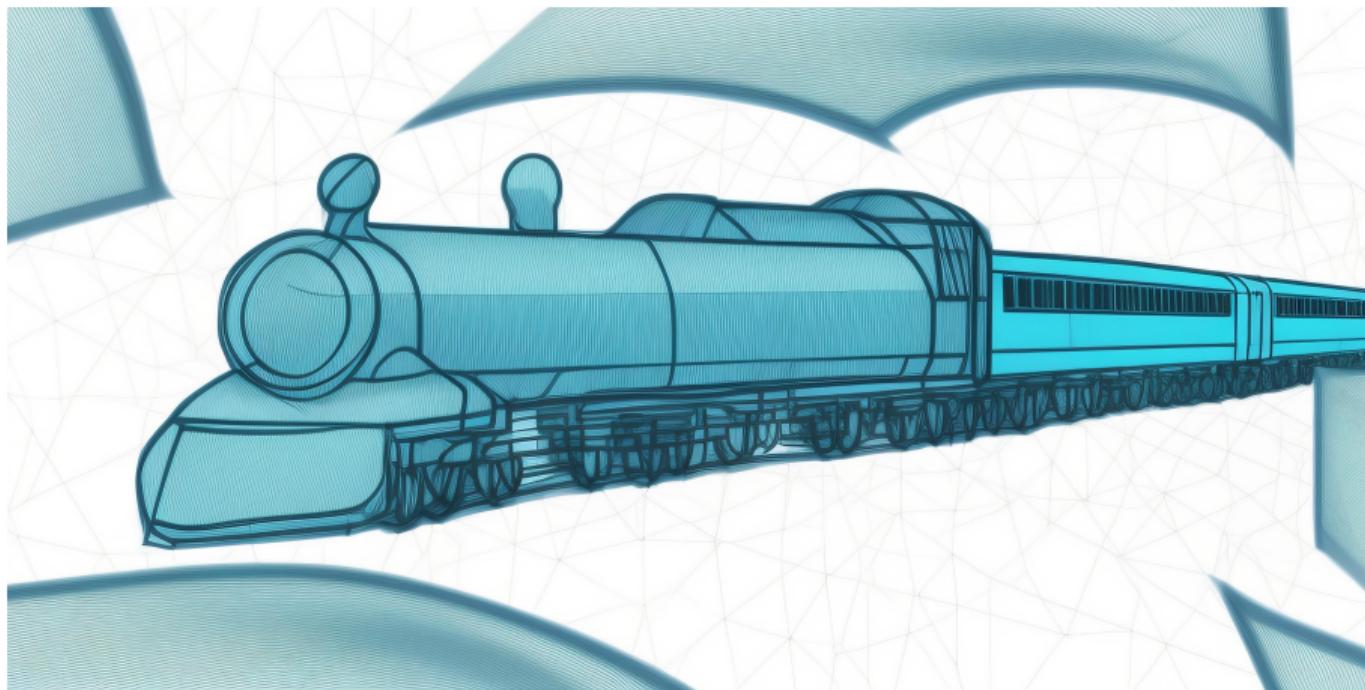
IHT



Riemannian SD

Credits: Vandereycken & Uschmajew

Part 2 | Tensors



§2 | Multilinear algebra / Tensors

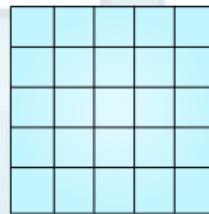
Tensors are high-dimensional arrays

$A[i]$



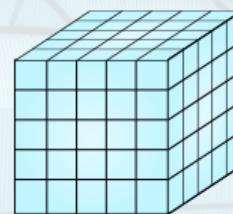
1D (vector)

$A[i, j]$



2D (matrix)

$A[i, j, k]$



3D

$A[i, j, k, l]$



4D

Examples

Audio
Stock prices

B&W images
Excel spreadsheet

Color images
B&W video
Minecraft map
MRI scan

Color video
fMRI scan

§2 | Low-rank tensors

Low-rank matrix \Leftrightarrow Product of two smaller matrices

$$A = XY^\top \rightarrow A[i, j] = \sum_{\ell=1}^r X[i, \ell]Y[j, \ell] \rightarrow \begin{array}{c} m \\ \text{---} \\ A \\ \text{---} \\ n \end{array} = \begin{array}{c} m \\ \text{---} \\ X \\ \text{---} \\ r \\ \text{---} \\ Y \\ \text{---} \\ n \end{array}$$

§2 | Low-rank tensors

Low-rank matrix \Leftrightarrow Product of two smaller matrices

$$A = XY^\top \rightarrow A[i, j] = \sum_{\ell=1}^r X[i, \ell]Y[j, \ell] \rightarrow \begin{array}{c} m \\ \text{---} \\ A \\ \text{---} \\ n \end{array} = \begin{array}{c} m \\ \text{---} \\ X \\ \text{---} \\ r \\ \text{---} \\ Y \\ \text{---} \\ n \end{array}$$

Low-rank tensor \Leftrightarrow Product of smaller tensors. **Example:**

$$A[i, j, k] = \sum_{\ell=1}^r X[i, \ell]Y[j, \ell]Z[k, \ell] \rightarrow \begin{array}{c} m \\ \text{---} \\ A \\ \text{---} \\ o \\ \mid \\ o \end{array} = \begin{array}{c} m \\ \text{---} \\ X \\ \text{---} \\ r \\ \text{---} \\ Y \\ \text{---} \\ n \\ \mid \\ Z \\ \mid \\ o \end{array}$$

§2 | Low-rank tensors

Low-rank matrix \Leftrightarrow Product of two smaller matrices

$$A = XY^\top \rightarrow A[i, j] = \sum_{\ell=1}^r X[i, \ell]Y[j, \ell] \rightarrow \begin{array}{c} m \\ \text{---} \\ A \\ \text{---} \\ n \end{array} = \begin{array}{c} m \\ \text{---} \\ X \\ \text{---} \\ r \\ \text{---} \\ Y \\ \text{---} \\ n \end{array}$$

Low-rank tensor \Leftrightarrow Product of smaller tensors. **Example:**

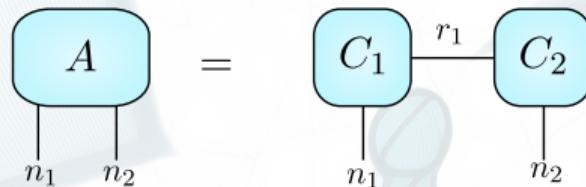
$$A[i, j, k] = \sum_{\ell=1}^r X[i, \ell]Y[j, \ell]Z[k, \ell] \rightarrow \begin{array}{c} m \\ \text{---} \\ A \\ \text{---} \\ o \\ \text{---} \\ n \end{array} = \begin{array}{c} m \\ \text{---} \\ X \\ \text{---} \\ r \\ \text{---} \\ Y \\ \text{---} \\ n \end{array} \quad \begin{array}{c} m \\ \text{---} \\ Z \\ \text{---} \\ o \\ \text{---} \\ n \end{array}$$

This is called a CP tensor. Equivalently we can write

$$A = \sum_{\ell=1}^r x_\ell \otimes y_\ell \otimes z_\ell,$$

i.e. a sum of elementary tensors.

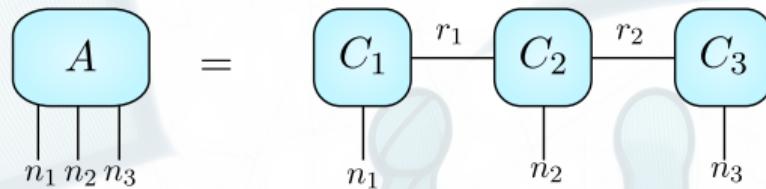
§2 | Tensor trains



$$\begin{aligned} A[i_1, i_2] &= \sum_{\ell_1=1}^{r_1} C_1[i_1, \ell_1] C_2[\ell_1, i_2] \\ &= C_1[i_1, :] C_2[:, i_2] \end{aligned}$$

The diagram illustrates the construction of a larger tensor from smaller ones. On the left is a large light blue grid divided into 16 smaller squares. One square in the second column from the left and second row from the top is highlighted in red. This is followed by an equals sign. To the right of the equals sign is a product expression: a smaller light blue grid with its second column highlighted in red, multiplied by a smaller light blue grid with its second row highlighted in red. The multiplication symbol is a standard times sign (\times).

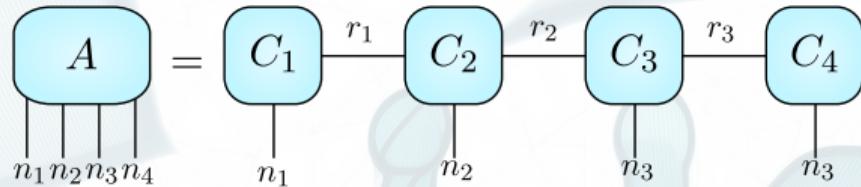
§2 | Tensor trains



$$\begin{aligned}A[i_1, i_2, i_3] &= C_1[i_1, :]C_2[:, i_2, :]C_3[:, i_3] \\&= \sum_{\ell_1=1}^{r_1} \sum_{\ell_2=1}^{r_2} C_1[i_1, \ell_1]C_2[\ell_1, i_2, \ell_2]C_3[\ell_2, i_3]\end{aligned}$$

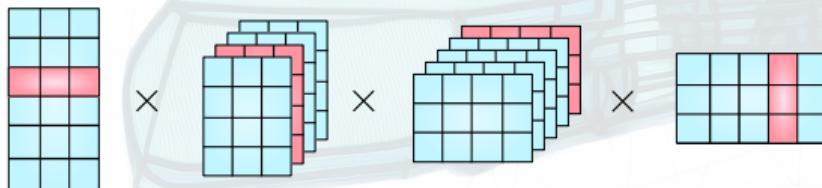
Below the equations, there is a diagram showing the element-wise multiplication of three tensors. On the left is a 4x4 grid with the bottom-left cell colored red. This is followed by a times sign (\times). To its right is a 3x3 grid where only the top row and left column are filled with blue cells, while the rest are red. Another times sign (\times) follows. To the right is a 4x4 grid where only the bottom row and right column are filled with blue cells, while the rest are red.

§2 | Tensor trains



$$A[i_1, i_2, i_3, i_4] = C_1[i_1, :] C_2[:, i_2, :] C_3[:, i_3, :] C_4[:, i_4]$$

$$= \sum_{\ell_1=1}^{r_1} \sum_{\ell_2=1}^{r_2} \sum_{\ell_3=1}^{r_3} C_1[i_1, \ell_1] C_2[\ell_1, i_2, \ell_2] C_3[\ell_2, i_3, \ell_3] C_4[\ell_3, i_4]$$



§2 | Tensor trains

Advantages

- ▶ Easy to optimize
- ▶ Easy to manipulate and compute entries, dot products, etc.
- ▶ Scales well (curse of dimensionality)

$$\mathcal{X} = \begin{matrix} C_1 & r_1 & C_2 & r_2 & \dots & r_{d-2} & C_{d-1} & r_{d-1} & C_d \\ | & & | & & & & | & & | \\ n_1 & & n_2 & & & & n_{d-1} & & n_d \end{matrix}$$

§2 | Tensor trains

Advantages

- ▶ Easy to optimize
- ▶ Easy to manipulate and compute entries, dot products, etc.
- ▶ Scales well (curse of dimensionality)

$$\mathcal{X} = C_1 \xrightarrow{r_1} C_2 \xrightarrow{r_2} \dots \xrightarrow{r_{d-2}} C_{d-1} \xrightarrow{r_{d-1}} C_d$$

$n_1 \quad \dots \quad n_d$

$n_1 \quad n_2 \quad \dots \quad n_{d-1} \quad n_d$



§2 | Tensor trains

Advantages

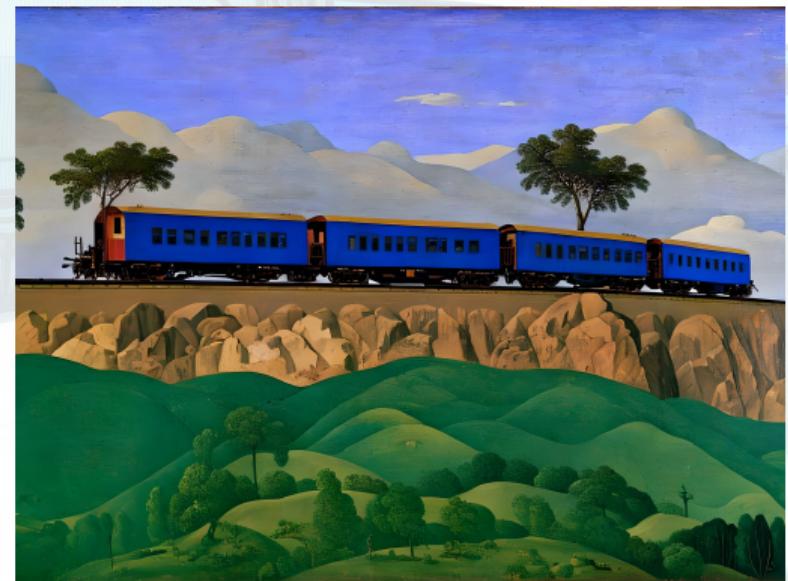
- ▶ Easy to optimize
- ▶ Easy to manipulate and compute entries, dot products, etc.
- ▶ Scales well (curse of dimensionality)

$$\mathcal{X} = \begin{matrix} C_1 & C_2 & \dots & C_{d-1} & C_d \end{matrix} \begin{matrix} r_1 \\ r_2 \\ \dots \\ r_{d-2} \\ r_{d-1} \end{matrix}$$

\vdots

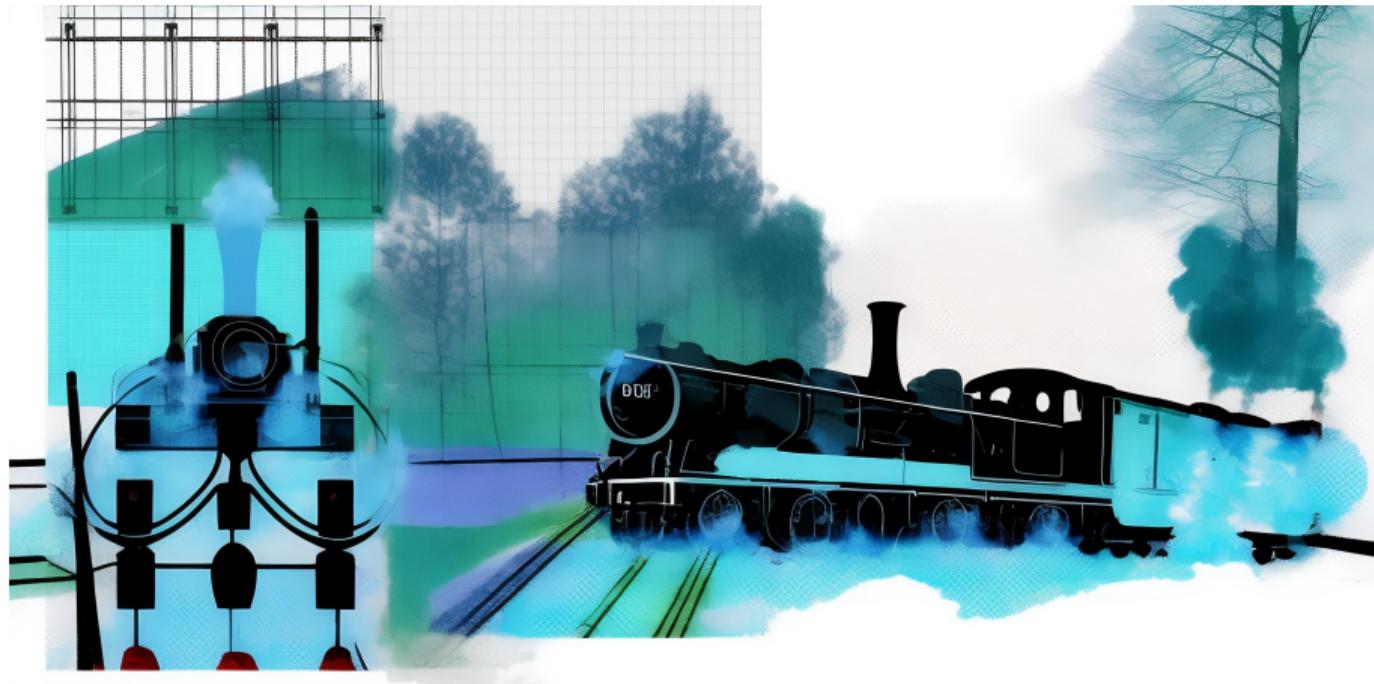
$n_1 \quad n_d$

$n_1 \quad n_2 \quad \dots \quad n_{d-1} \quad n_d$



Part 3 | TTML

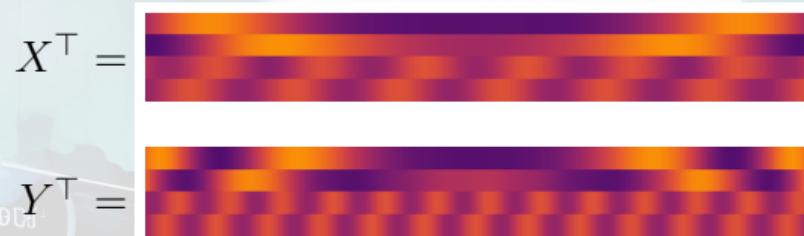
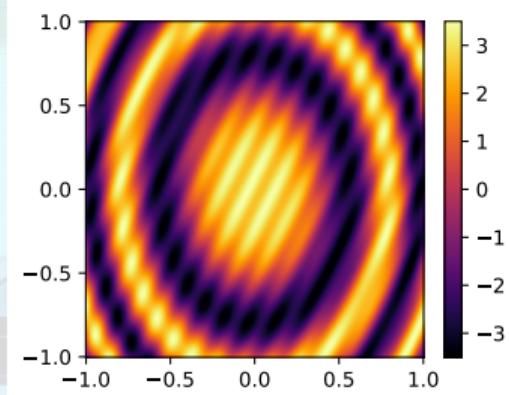
Joint work with Bart Vandereycken



§3 | Matrices as discretized functions

Many functions in 2D can be represented by low rank matrices.

$$f(x, y) = 3 \cos(10(x^2 + y^2/2)) - \sin(20(2x - y))/2$$

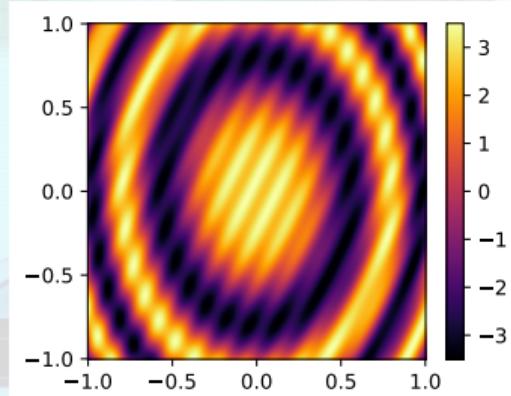


Matrix \Leftrightarrow function $I^2 \rightarrow \mathbb{R}$ discretized on a grid

§3 | Matrices as discretized functions

Many functions in 2D can be represented by low rank matrices.

$$f(x, y) = 3 \cos(10(x^2 + y^2/2)) - \sin(20(2x - y))/2$$



Matrix \Leftrightarrow function $I^2 \rightarrow \mathbb{R}$ discretized on a grid

Tensor \Leftrightarrow function $I^d \rightarrow \mathbb{R}$ discretized on a grid

- ▶ Low-rank tensors can represent complicated functions

§3 | Supervised learning

Idea: Use tensors to do machine learning

§3 | Supervised learning

Idea: Use tensors to do machine learning

Supervised learning. Learn a function $f: I^d \rightarrow \mathbb{R}$ based on a number of examples.

Example: Predict noise made by airplane wing in a wind tunnel.

Frequency (Hz)

Angle of attack (deg)

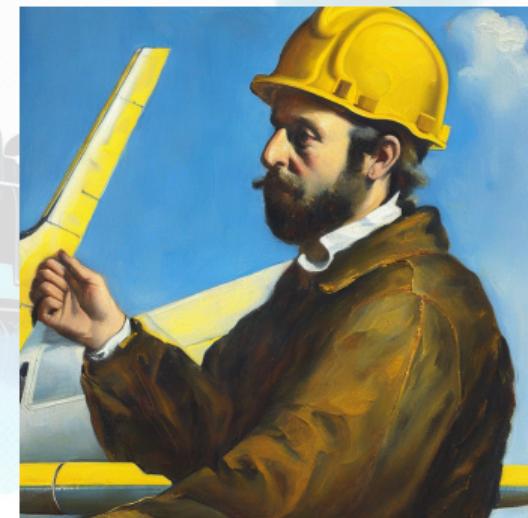
Chord length (m)

Wind tunnel airspeed (m/s)

Displacement thickness (m)



Sound pressure (dB)



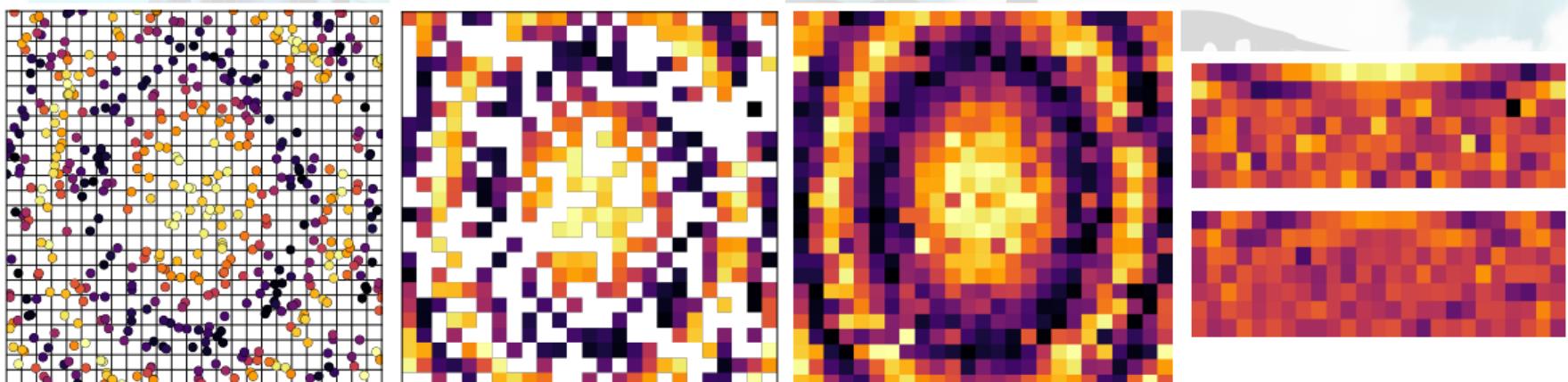
§3 | TTML: Matrix case

Matrix $A \Leftrightarrow$ function $\hat{f}: I^2 \rightarrow \mathbb{R}$ discretized on a grid

Supervised learning. Learn a function $f: I^d \rightarrow \mathbb{R}$ based on a number of examples

Each value $\hat{f}(x_1, x_2)$ corresponds to an entry A_{i_1, i_2} of a matrix.

- ▶ Learning $\hat{f} \Leftrightarrow$ matrix completion of the matrix A .



§3 | TTML: Matrix case

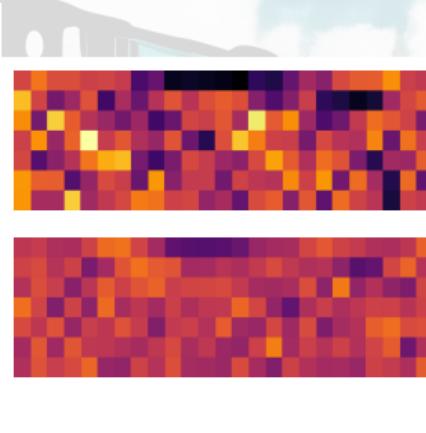
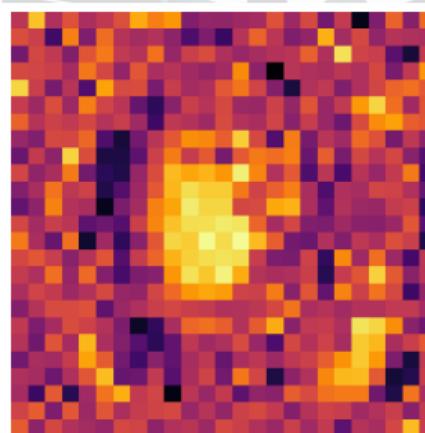
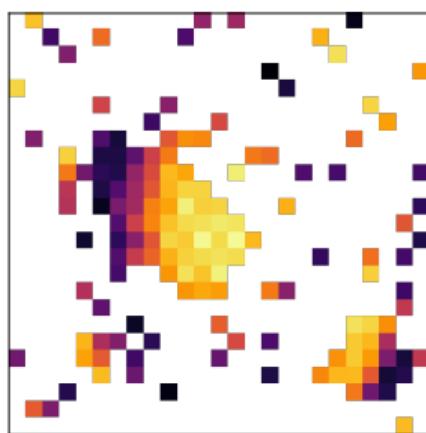
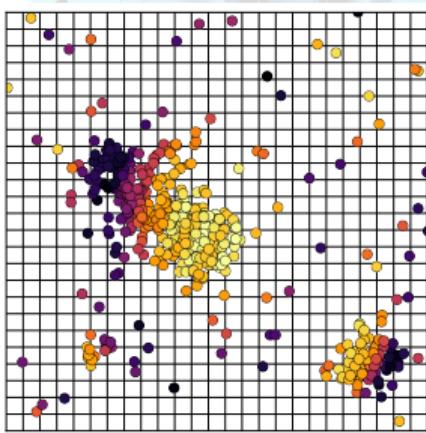
Matrix $A \Leftrightarrow$ function $\hat{f}: I^2 \rightarrow \mathbb{R}$ discretized on a grid

Supervised learning. Learn a function $f: I^d \rightarrow \mathbb{R}$ based on a number of examples

Each value $\hat{f}(x_1, x_2)$ corresponds to an entry A_{i_1, i_2} of a matrix.

- ▶ Learning $\hat{f} \Leftrightarrow$ matrix completion of the matrix A .

Problem: Data is rarely uniformly distributed



§3 | TTML: Matrix case

Matrix $A \Leftrightarrow$ function $\hat{f}: I^2 \rightarrow \mathbb{R}$ discretized on a grid

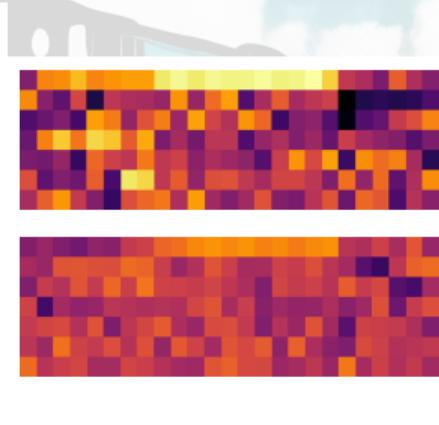
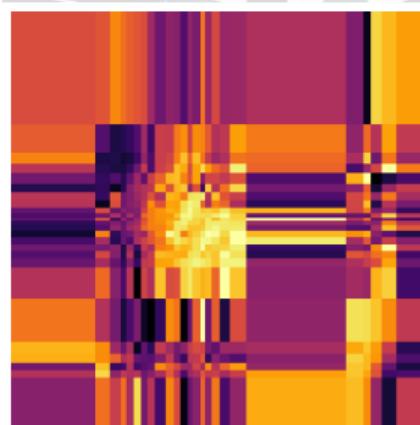
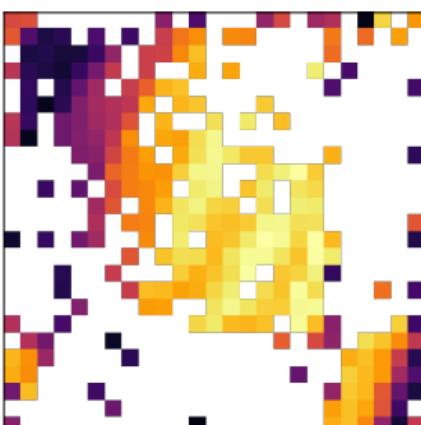
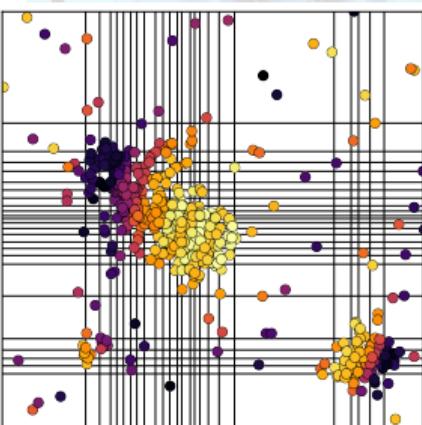
Supervised learning. Learn a function $f: I^d \rightarrow \mathbb{R}$ based on a number of examples

Each value $\hat{f}(x_1, x_2)$ corresponds to an entry A_{i_1, i_2} of a matrix.

- ▶ Learning $\hat{f} \Leftrightarrow$ matrix completion of the matrix A .

Problem: Data is rarely uniformly distributed

- ▶ Don't use a uniform grid



Tensor $\mathcal{T} \Leftrightarrow$ function $\hat{f}: I^d \rightarrow \mathbb{R}$ discretized on a grid

Supervised learning. Learn a function $f: I^d \rightarrow \mathbb{R}$ based on a number of examples

Each value $y_i = f(\mathbf{x}_i) = f(x_{i,1}, \dots, x_{i,d})$ corresponds to an entry $\mathcal{T}[i_1, \dots, i_d]$ of a tensor.

- ▶ Learning $\hat{f} \Leftrightarrow$ tensor completion of the tensor \mathcal{T} .

Tensor $\mathcal{T} \Leftrightarrow$ function $\hat{f}: I^d \rightarrow \mathbb{R}$ discretized on a grid

Supervised learning. Learn a function $f: I^d \rightarrow \mathbb{R}$ based on a number of examples

Each value $y_i = f(\mathbf{x}_i) = f(x_{i,1}, \dots, x_{i,d})$ corresponds to an entry $\mathcal{T}[i_1, \dots, i_d]$ of a tensor.

- ▶ Learning $\hat{f} \Leftrightarrow$ tensor completion of the tensor \mathcal{T} .

$$\min_{\mathcal{T} \in \mathcal{M}} \sum_{i=1}^N (\mathcal{T}[j_1(\mathbf{x}_i), \dots, j_d(\mathbf{x}_i)] - y_i)^2, \quad (*)$$

\mathcal{M} is set of all rank $\leq (r_1, \dots, r_{d-1})$ tensor trains.

- ▶ Solving $(*)$ using alternating least squares (ALS) leads to ill-conditioned subproblems for non-uniformly distributed data in practice.
- ▶ Riemannian gradient descent methods work well in practice

Tensor completion

$$\min_{\mathcal{T} \in \mathcal{M}} \sum_{i=1}^N (\mathcal{T}[j_1(\mathbf{x}_i), \dots, j_d(\mathbf{x}_i)] - y_i)^2, \quad (*)$$

Iterative methods need good initialization.

Trick:

- ▶ First train a different ML model $g: \mathbb{R}^d \rightarrow \mathbb{R}$ (e.g. random forest, neural network) on the same training data.
- ▶ Discretize $g \Rightarrow$ obtain tensor \mathcal{S} .
- ▶ Use initialisation \mathcal{T}_0 such that $\mathcal{T}_0 \approx \mathcal{S}$.
- ▶ This is possible because we can sample arbitrary entries of \mathcal{S} . We use TT-cross.
- ▶ Quality of model obtained depends strongly on ML model g used for initialization.

Example: Predict noise made by airplane wing in a wind tunnel.

Frequency (Hz)

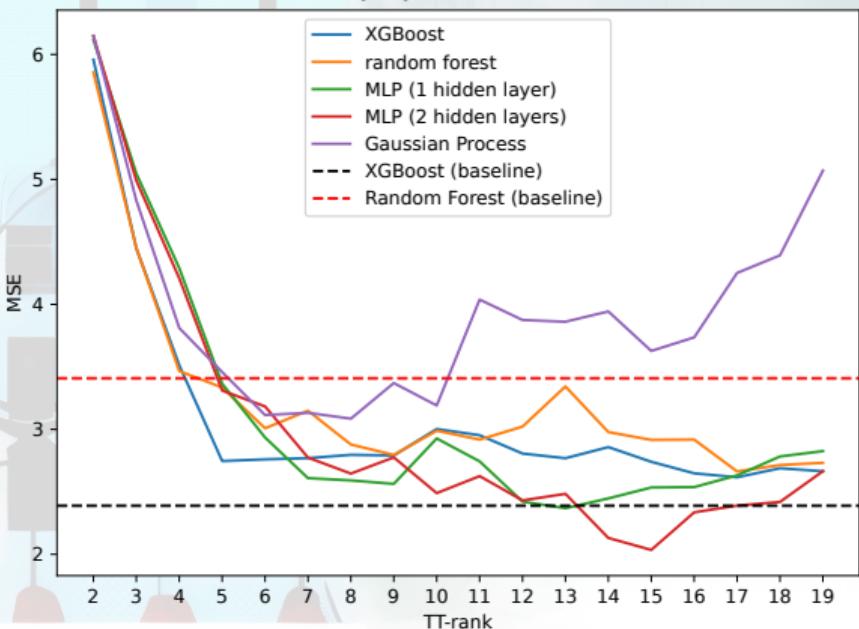
Angle of attack (deg)

Chord length (m)

→ Sound pressure (dB)

Wind tunnel airspeed (m/s)

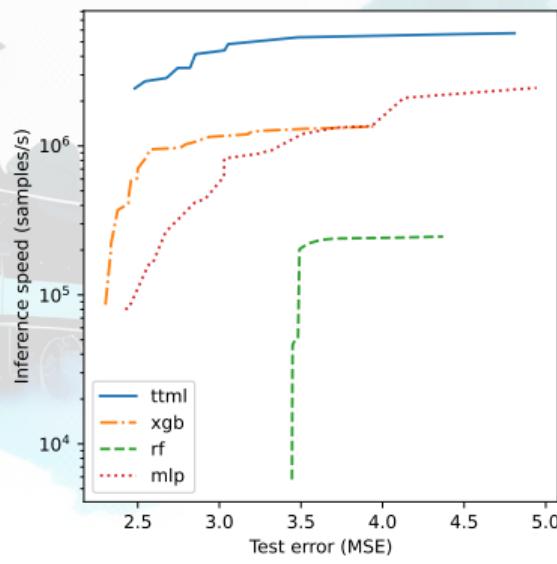
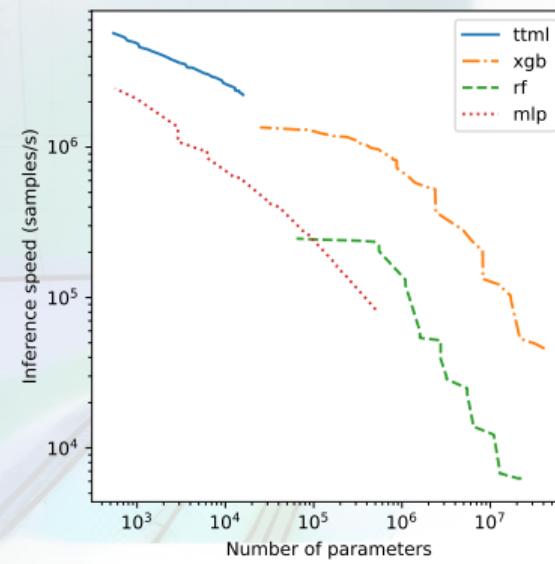
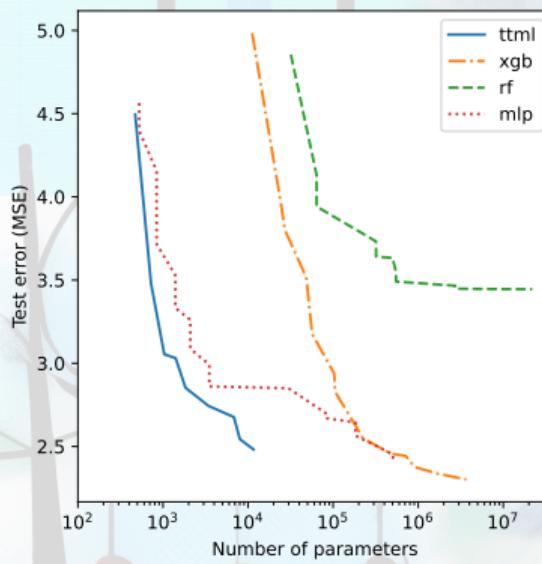
Displacement thickness (m)



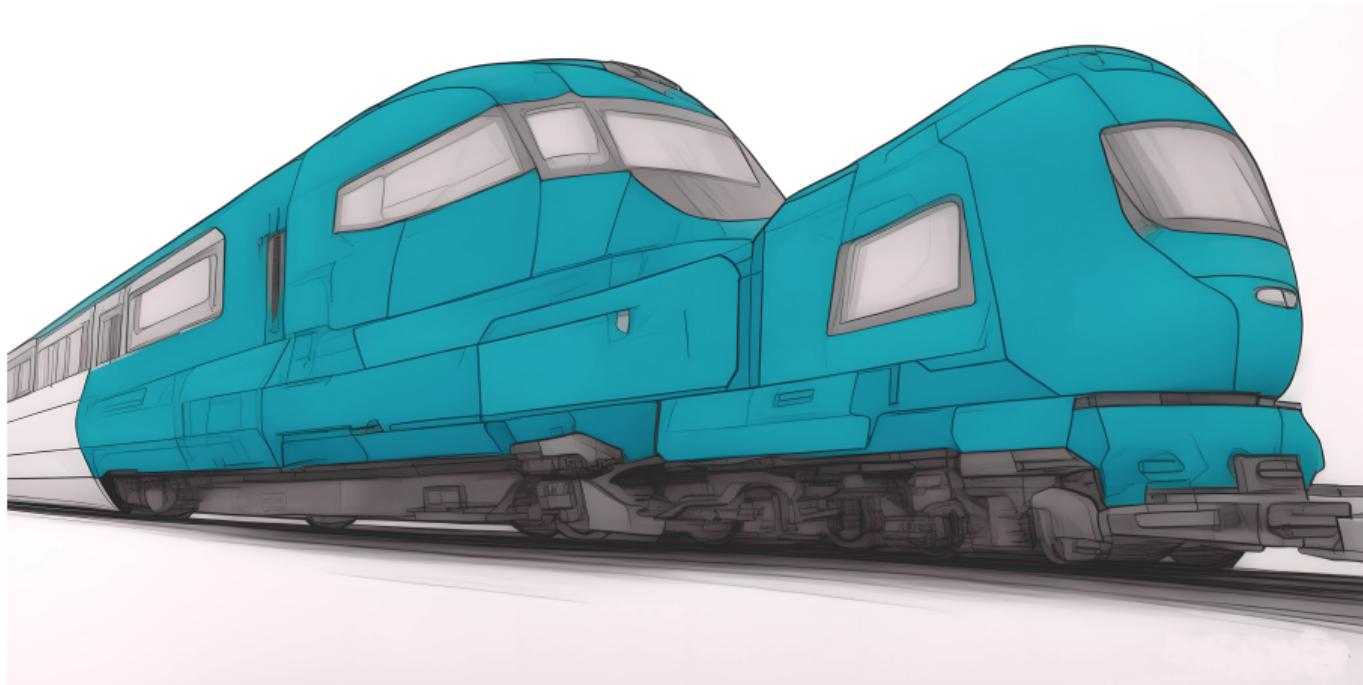
§3 | TTML: Experiments

For certain datasets, TTML is *faster*, uses fewer parameters/memory and gives better error than other methods.

- ▶ Memory and speed is usually excellent for TTML.
- ▶ Performance depends strongly on dataset used. TTML works best for .



Part 4 | Randomized linear algebra



§4 | Randomized linear algebra

Recall: Best rank r approximation of an $m \times n$ matrix is given by truncated SVD.

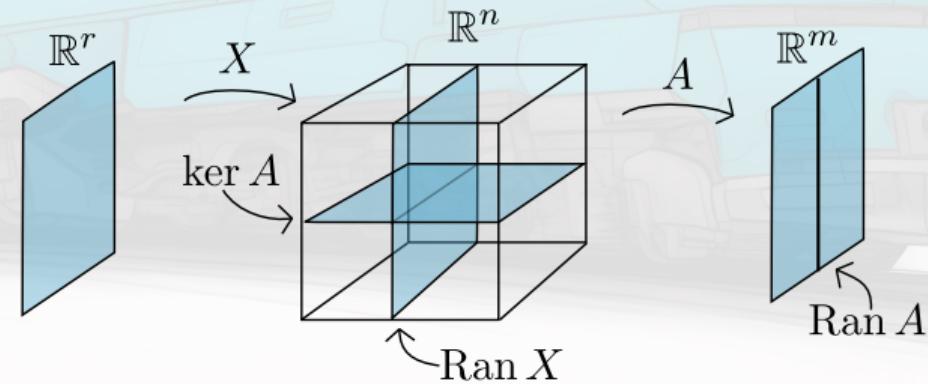
► **Problem:** SVD costs $O(mn^2)$ flops ($m > n$).

Randomized matrix decomposition: First multiply A by a random $n \times r$ matrix X , then try to obtain truncated SVD from AX . We call X ‘dimension reduction matrix’ (DRM).

Lemma

If A is rank $\hat{r} < r$ then AX has almost surely the same range as A .

Proof: Counting dimensions.

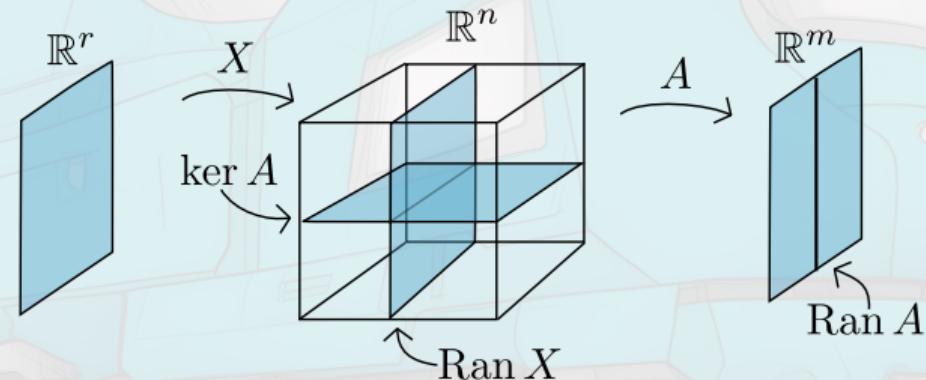


§4 | Randomized linear algebra

Lemma

If A is rank $\hat{r} < r$ then AX has almost surely the same range as A .

Proof: Counting dimensions.



Idea: If A can be well approximated by a rank- \hat{r} matrix $A_{\hat{r}}$, then restricting the range to $\text{Ran } AX$ keeps most of the information.

- ▶ That is, $\mathcal{P}_{AX}A \approx A$. More precisely $\|\mathcal{P}_{AX}A - A\| \leq C\|A_{\hat{r}} - A\|$.
- ▶ Observe $\mathcal{P}_{AX} = QQ^\top$ where Q is an $m \times r$ orthonormal basis of $\text{Ran } AX$.
- ▶ $Q(Q^\top A)$ is a rank r approximation.

§4 | Halko-Martinsson-Tropp method

Theorem (HMT)

Let X be $n \times r$ random normal with $r > \hat{r} + 1$, and let $Q = \text{Orth}(AX)$. Then

$$\mathbb{E} \|Q(Q^\top A) - A\|_F^2 \leq \left(1 + \frac{\hat{r}}{r - \hat{r} - 1}\right) \|A_{\hat{r}} - A\|_F^2$$

- ▶ Can compute good rank r approximation in $O(mnr) < O(mn^2)$ flops
- ▶ If A is a large structured (e.g. sparse) matrix, then computing $\text{Orth}(AX)$ is expensive and difficult to parallelize

§4 | Generalized Nyström method

Idea: If m, n are both big, AX is still a big matrix.

- ▶ Use two DRMs and compute sketch $Y^\top AX$ where X is $n \times r^R$ and Y is $m \times r^L$.
- ▶ Obtain rank r^R approximation $AX(Y^\top AX)^\dagger Y^\top A$, where \dagger denotes pseudoinverse.
- ▶ Can be written as *oblique* projection $\mathcal{P}_{AX,Y}A$.

Theorem (Tropp et al.)

Assume X, Y random normal and $r^L > r^R > \hat{r} - 1$. The generalized Nyström method $A \approx \mathcal{P}_{AX,Y}A = AX(Y^\top AX)^\dagger Y^\top A$ then satisfies

$$\mathbb{E} \|\mathcal{P}_{AX,Y}A - A\|_F^2 \leq \left(1 + \frac{r^R}{r^L - r^R - 1}\right) \left(1 + \frac{\hat{r}}{r^R - \hat{r} - 1}\right) \|A_{\hat{r}} - A\|_F^2$$

§4 | Generalized Nyström method

Theorem (Tropp et al.)

Assume X, Y random normal and $r^L > r^R > \hat{r} - 1$. The generalized Nyström method $A \approx \mathcal{P}_{AX,Y}A = AX(Y^\top AX)^\dagger Y^\top A$ then satisfies

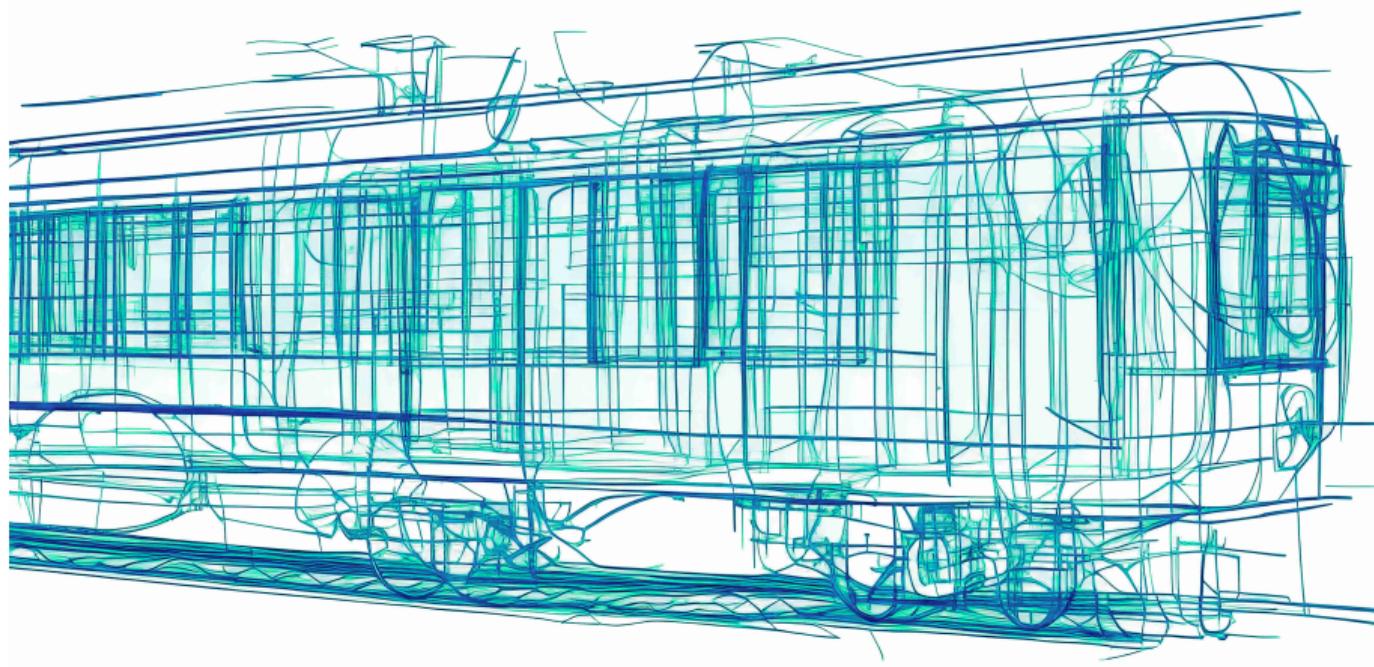
$$\mathbb{E} \|\mathcal{P}_{AX,Y}A - A\|_F^2 \leq \left(1 + \frac{r^R}{r^L - r^R - 1}\right) \left(1 + \frac{\hat{r}}{r^R - \hat{r} - 1}\right) \|A_{\hat{r}} - A\|_F^2$$

Advantages

- ▶ Cheap for very large structured matrices
- ▶ Streaming method; we can compute an approximation of $A + B$ both in parallel while seeing each of the summands only once.

Part 5 | Randomized TT Approximations

Joint work with Bart Vandereycken and Daniel Kressner



§5 | Streaming TT approximation (STTA)

Idea: Generalize the generalized Nyström method to tensor trains.

Step 1: Compute sketches – linear in the data and one-pass

$$\begin{array}{ccc} r_{\mu-1}^L & \xrightarrow{\Psi_\mu} & r_\mu^R \\ & n_\mu & \end{array} := \begin{array}{ccccc} r_{\mu-1}^L & Y_{\mu-1} & \xrightarrow{\vdots} & \mathcal{T} & \xrightarrow{\vdots} X_\mu & r_\mu^R \\ n_{\mu-1} & & n_1 & & n_{\mu+1} & \\ & & & & n_d & \\ & & & n_\mu & & \end{array}$$

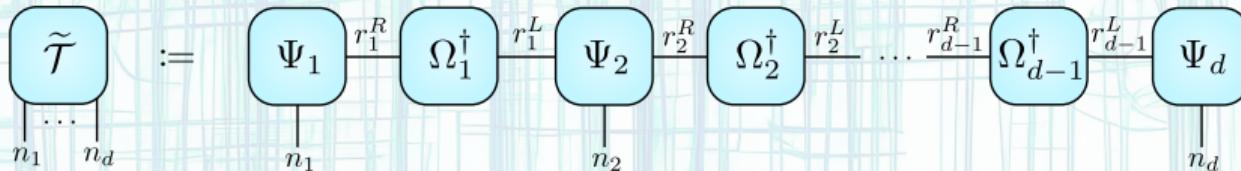
$$\begin{array}{ccc} r_\mu^L & \xrightarrow{\Omega_\mu} & r_\mu^R \\ & n_\mu & \end{array} := \begin{array}{ccccc} r_\mu^L & Y_\mu & \xrightarrow{\vdots} & \mathcal{T} & \xrightarrow{\vdots} X_\mu & r_\mu^R \\ n_\mu & & n_1 & & n_{\mu+1} & \\ & & & & n_d & \\ & & & n_\mu & & \end{array}$$

X_μ is a ‘right’ DRM of size $(n_{\mu+1} \cdots n_d) \times r_\mu^R$ for $1 \leq \mu \leq d-1$;

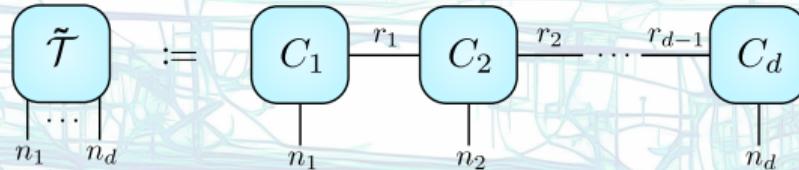
Y_μ is a ‘left’ DRM of size $(n_1 \cdots n_\mu) \times r_\mu^L$ for $1 \leq \mu \leq d-1$.

§5 | Streaming TT approximation (STTA)

Step 2: Assemble in a tensor network – zero cost

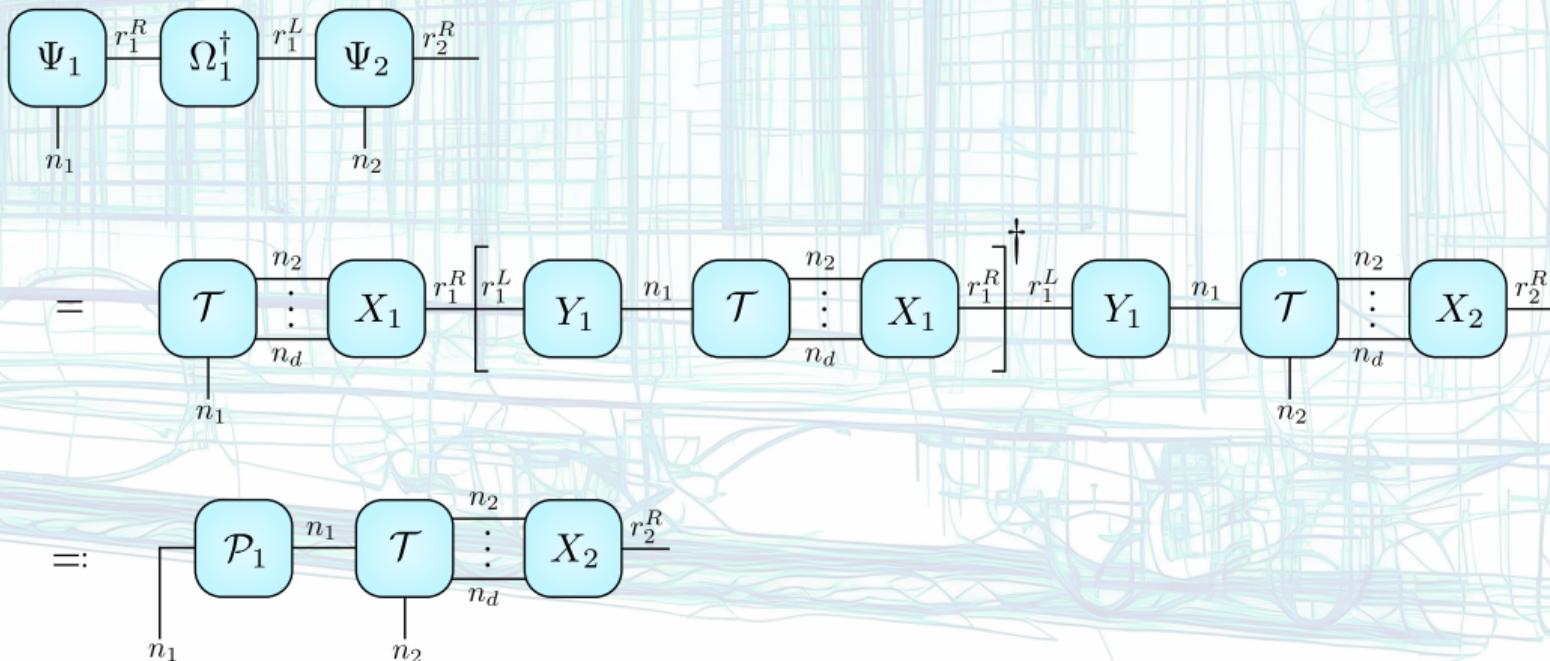


Step 3: Reduce to standard TT – optional and cheap



§5 | Why does STTA work?

► Write approximation as projector



§5 | Why does STTA work?

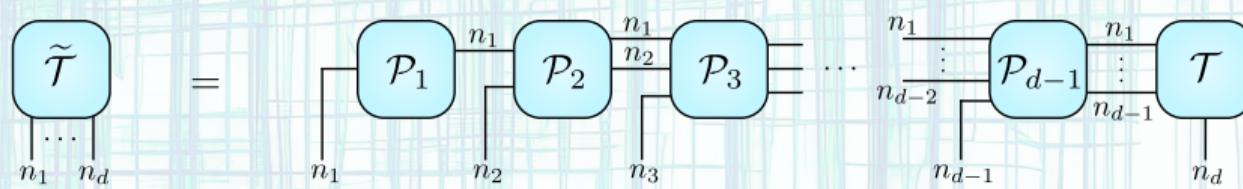
- ▶ Write approximation as projector

$$\tilde{\mathcal{T}} = \mathcal{P}_1 \xrightarrow{n_1} \mathcal{P}_2 \xrightarrow{n_2} \mathcal{P}_3 \xrightarrow{n_3} \dots \xrightarrow{n_{d-2}} \mathcal{P}_{d-1} \xrightarrow{n_{d-1}} \mathcal{T}$$

The diagram illustrates the decomposition of a projector $\tilde{\mathcal{T}}$ into a sequence of projectors $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \dots, \mathcal{P}_{d-1}, \mathcal{T}$. The input vector x (represented by a vertical line) passes through each projector sequentially. The dimension of the space at each stage is indicated by the number of lines: n_1 for the input, n_2 for the output of \mathcal{P}_1 , n_3 for the output of \mathcal{P}_2 , and so on, down to n_d for the final output of \mathcal{T} .

§5 | Why does STTA work?

- ▶ Write approximation as projector



- ▶ Each projector is **oblique**
- ▶ For TT-SVD and TT-HMT approximation has **orthogonal** projectors
- ▶ Effect of each projector on error can be analyzed independently

§5 | Error bound in expectation

Theorem

Let $\tilde{\mathcal{T}}$ be obtained using STTA, and let $\mathcal{T}_{\hat{r}}$ be a best rank $\hat{\mathbf{r}} = (\hat{r}_1, \dots, \hat{r}_{d-1})$ TT-approximation with $\hat{r}_\mu + 1 < r_\mu^R < r_\mu^L - 1$. Then

$$\mathbb{E} \|\tilde{\mathcal{T}} - \mathcal{T}\|_F \leq \left(\sum_{\mu=1}^{d-1} \left[\prod_{\alpha=1}^{\mu-1} c_\alpha \right] c'_\mu \right) \|\mathcal{T}_{\hat{r}} - \mathcal{T}\|_F,$$

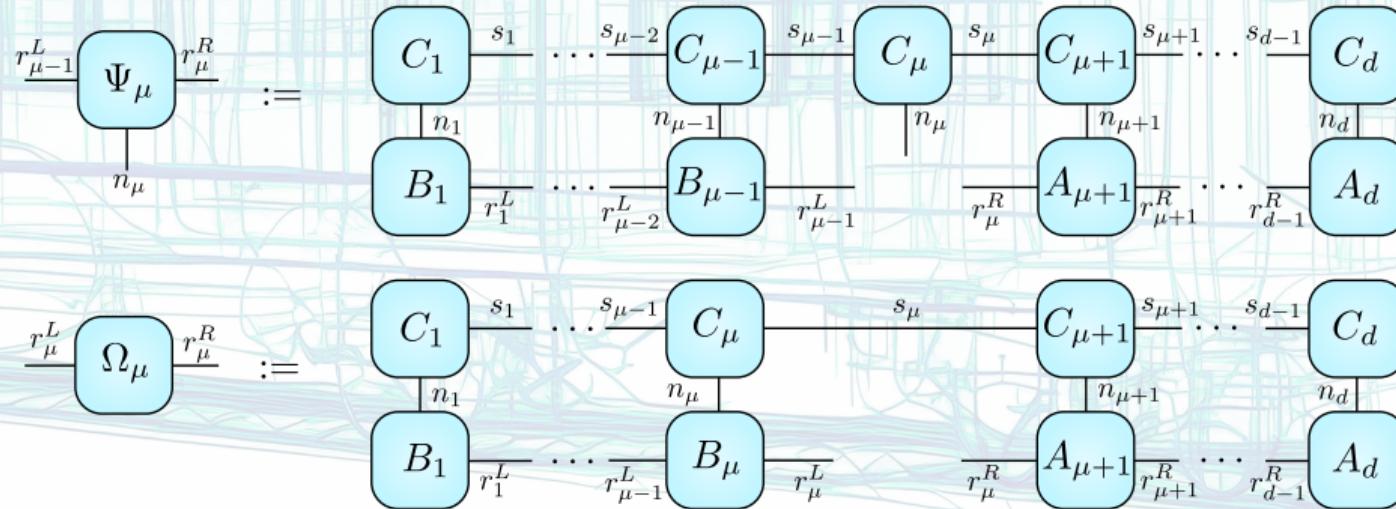
with

$$c_\mu = 1 + \sqrt{\frac{r_\mu^R}{r_\mu^L - r_\mu^R - 1}}, \quad c'_\mu = \sqrt{1 + \frac{r_\mu^R}{r_\mu^L - r_\mu^R - 1}} \cdot \sqrt{1 + \frac{\hat{r}_\mu}{r_\mu^R - \hat{r}_\mu - 1}}.$$

- ▶ Like GN and HMT, error is within a constant of optimal error.
- ▶ **Practical performance** is better than theoretical; no exponential dependence on d .
- ▶ Very parallelizable and streamable

§5 | Structured sketches

- ▶ Using TTs as DRMs we can do **fast structured sketches**.
- ▶ Efficient methods for CP, Tucker and sparse tensors, and any linear combination of these (e.g., TT + sparse).
- ▶ Example: diagram for computing Ψ_μ , Ω_μ when \mathcal{T} is a TT.



§5 | Orthogonalized Tensor Train Sketch (OTTS)

- ▶ We can modify STTA to improve accuracy at the cost of speed and streamability.
- ▶ Just replace left DRM in definition of Ψ_μ with previous cores, and do orthogonalization.

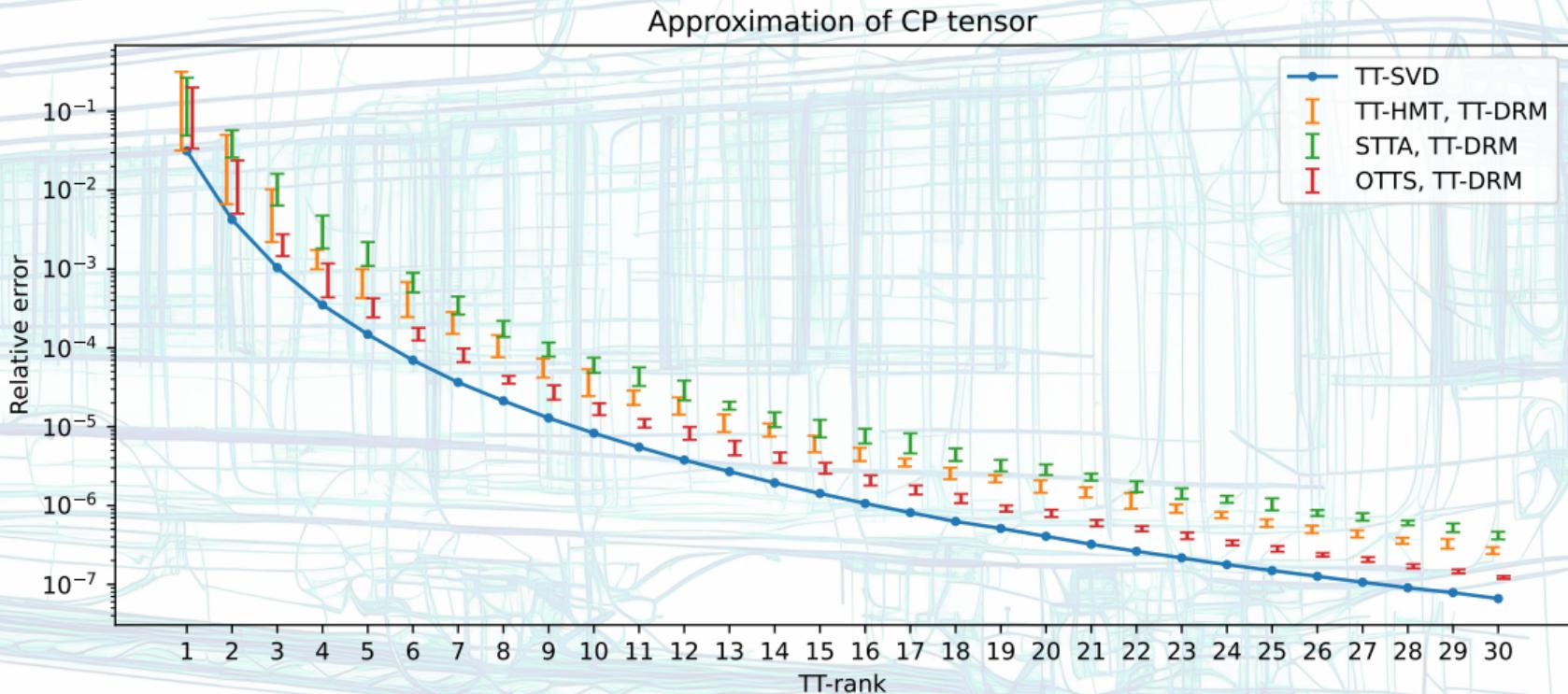
$$\begin{array}{c} r_{\mu-1}^L \quad \text{---} \quad \boxed{\Psi_\mu} \quad \text{---} \quad r_\mu^R \\ \text{---} \quad \Big| \quad \text{---} \\ n_\mu \end{array} := \begin{array}{c} r_{\mu-1}^L \quad \text{---} \quad \boxed{C_{\leq \mu-1}} \quad \text{---} \quad \begin{matrix} n_1 \\ \vdots \\ n_{\mu-1} \end{matrix} \quad \text{---} \quad \boxed{\mathcal{T}} \quad \text{---} \quad \begin{matrix} n_{\mu+1} \\ \vdots \\ n_d \end{matrix} \quad \text{---} \quad \boxed{X_\mu} \quad \text{---} \quad r_\mu^R \\ \text{---} \quad \Big| \quad \text{---} \\ n_\mu \end{array}$$

Theorem

Let $\tilde{\mathcal{T}}$ be obtained using OTTS, and let $\widehat{\mathcal{T}_r}$ be a best rank $\widehat{\mathbf{r}} = (\widehat{r}_1, \dots, \widehat{r}_{d-1})$ TT-approximation with $\widehat{r}_\mu + 1 < r_\mu^R < r_\mu^L - 1$. Then

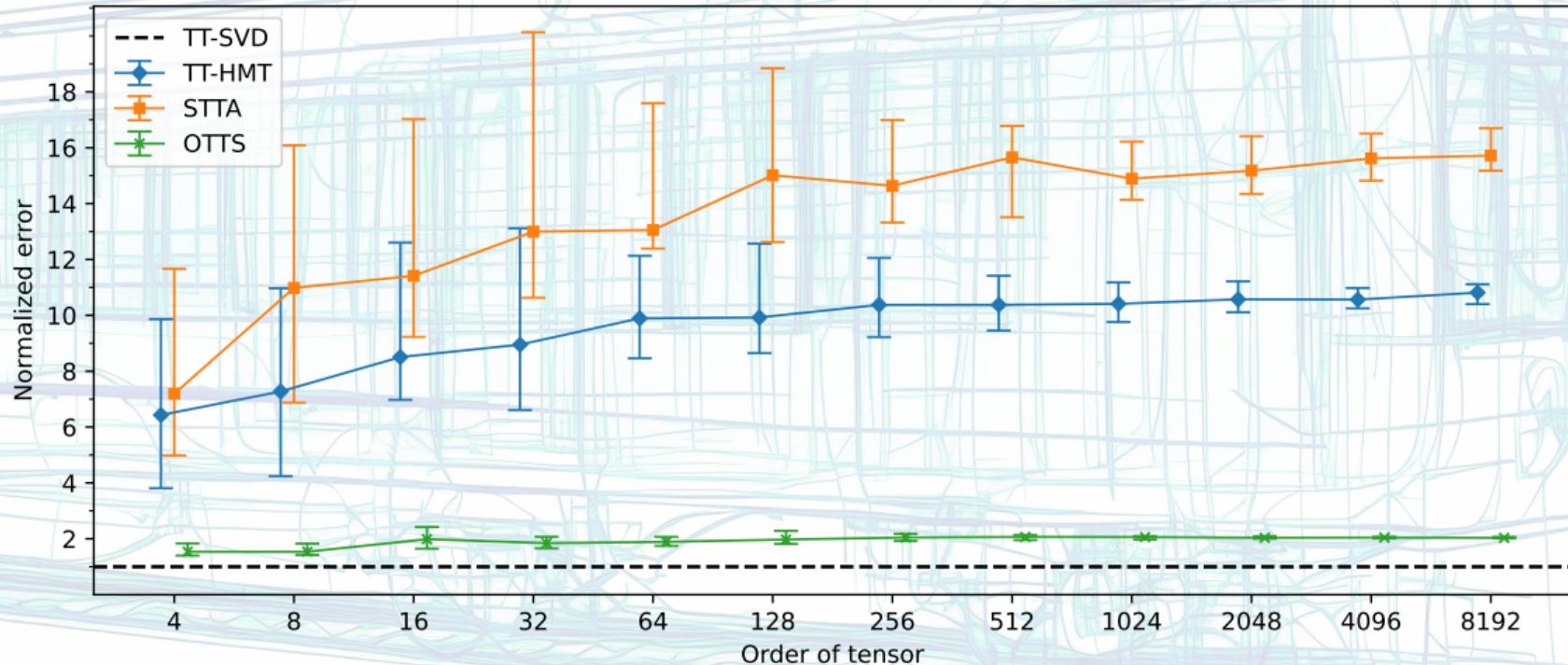
$$\mathbb{E} \|\tilde{\mathcal{T}} - \mathcal{T}\|_F^2 \left(\sum_{\mu=1}^{d-1} (c'_\mu)^2 \right) \|\widehat{\mathcal{T}_r} - \mathcal{T}\|_F^2,$$

§5 | Experiments

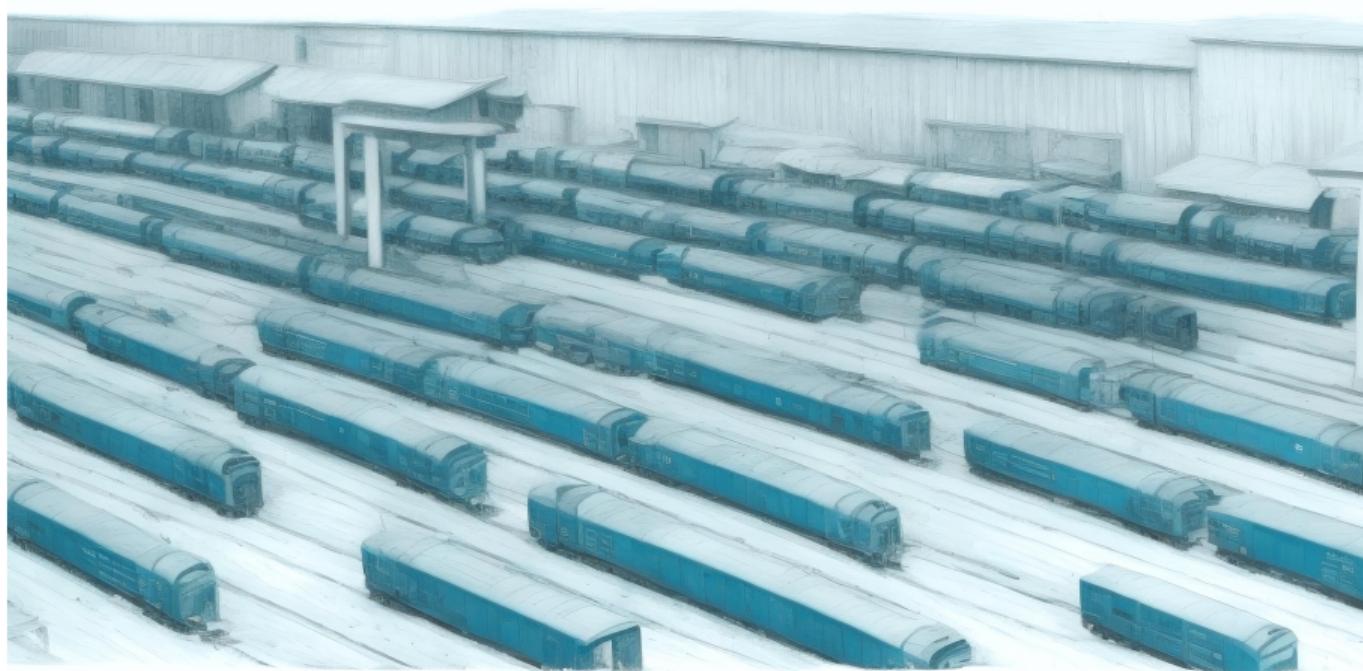


§5 | Experiments

Dependence of the error on the tensor order



Part 6 | Sketched low-rank solvers



§6 | Low-rank linear systems

Consider the generalized Sylvester equation:

$$\mathcal{L}(H) = AH + HB + \sum_{i=1}^{\ell} M_i H N_i = C$$

We want to find low-rank solutions.

- ▶ Need low-rank approximation of $\mathcal{L}(H)$
- ▶ Low-rank streaming methods like GN and STTA are excellent for this

§6 | Low-rank linear systems

Consider the generalized Sylvester equation:

$$\mathcal{L}(H) = AH + HB + \sum_{i=1}^{\ell} M_i H N_i = C$$

We want to find low-rank solutions.

- ▶ Need low-rank approximation of $\mathcal{L}(H)$
- ▶ Low-rank streaming methods like GN and STTA are excellent for this

Idea: Use properties of low-rank streaming methods to develop general approaches for solving low-rank linear systems.

§6 | Compatible inner products

Example: Generalized Nyström

We have an approximation $\mathcal{S}_{X,Y}(A) = AX(Y^\top AX)^\dagger Y^\top A$.

- ▶ Computing inner product $\langle A, B \rangle$ of two rank- r matrices costs $O(r^2(m + n))$ flops.
- ▶ Approximation $\langle A, B \rangle_{X,Y} = \langle Y^\top AX, Y^\top BX \rangle$ costs only $O(r^2)$.

§6 | Compatible inner products

Example: Generalized Nyström

We have an approximation $\mathcal{S}_{X,Y}(A) = AX(Y^\top AX)^\dagger Y^\top A$.

- ▶ Computing inner product $\langle A, B \rangle$ of two rank- r matrices costs $O(r^2(m+n))$ flops.
- ▶ Approximation $\langle A, B \rangle_{X,Y} = \langle Y^\top AX, Y^\top BX \rangle$ costs only $O(r^2)$.

Why does this work?

Compatibility with the sketched approximation $\mathcal{S}_{X,Y}$:

- ▶ $\langle A, B \rangle_{X,Y} = \langle A, \mathcal{S}_{X,Y}(B) \rangle_{X,Y} = \langle \mathcal{S}_{X,Y}(A), \mathcal{S}_{X,Y}(B) \rangle_{X,Y}$.
- ▶ $\langle A, A \rangle_{X,Y} = 0 \Leftrightarrow \mathcal{S}_{X,Y}(A) = 0$.

§6 | Sketched low-rank solvers

We formalize the notions of *linear low-rank streaming method* and with a compatible bilinear form to get:

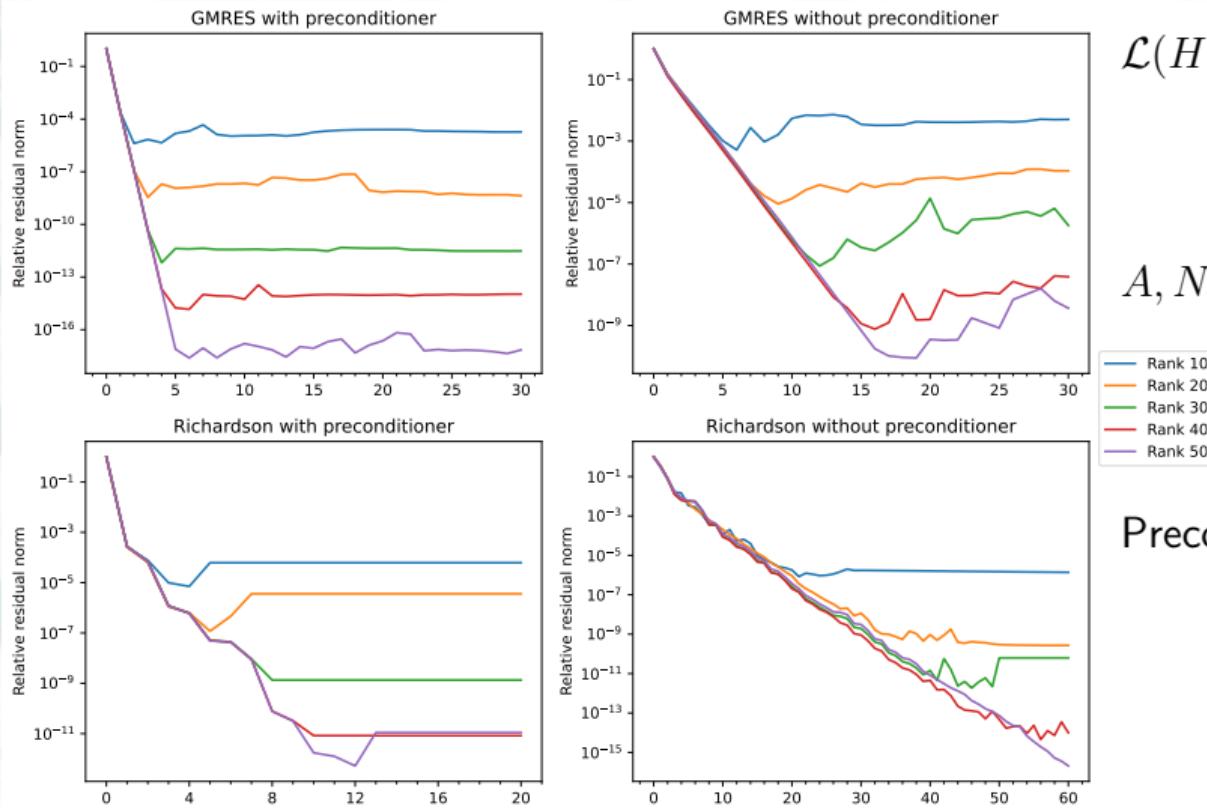
- ▶ Sketched version of Richardson iteration.
- ▶ Sketched version of Krylov methods (GMRES).

These methods are:

- ▶ generalist and apply to any linear system;
- ▶ compatible with preconditioning;
- ▶ linear in the number of terms in linear operator \mathcal{L} ;
- ▶ generalize existing sketched methods using e.g. $\mathcal{S}_{X,Y}(A) = \mathcal{P}_Y A \mathcal{P}_X$.

Does not work for STTA yet; no obvious compatible bilinear form.

§6 | Experiment



$$\begin{aligned}\mathcal{L}(H) &= AH + HA^\top \\ &\quad + \gamma^2 (N_1 H N_1^\top + N_2 H N_2^\top) \\ &= C,\end{aligned}$$

A, N_i tridiagonal, C rank 2.

Preconditioner: Lyapunov solver.

Part 7 | The part where you're supposed to ask questions

