EE613 - Machine Learning for Engineers

https://moodle.epfl.ch/course/view.php?id=16819

TENSOR FACTORIZATION

Sylvain Calinon
Robot Learning and Interaction Group
Idiap Research Institute
Nov 23, 2023

EE613 schedule

Thu. 21.09.2023	(C) 1. ML introduction
Thu. 28.09.2023	(C) 2. Bayesian 1 (C) 3. Bayesian 2
Thu. 12.10.2023	(C) 4. Hidden Markov Models
Thu. 19.10.2023	(C) 5. Dimensionality reduction
Thu. 26.10.2023	(C) 6. Decision trees
Thu. 02.11.2023	(C) 7. Linear regression
Thu. 09.11.2023	(C) 8. Nonlinear regression
Thu. 16.11.2023	(C) 9. Kernel Methods - SVM
Thu. 23.11.2023	(C) 10. Tensor factorization
Thu. 30.11.2023	(C) 11. Deep learning 1
Thu. 07.12.2023	(C) 12. Deep learning 2
Thu. 14.12.2023	(C) 13. Deep learning 3
Thu. 21.12.2023	(C) 14. Deep learning 4

Outline

Linear algebra:

- Products (Hadamard, Kronecker, Khatri-Rao)
- Separation of variables
- Singular value decomposition (SVD)

3 tensor decomposition models:

- Canonical polyadic (CP)
- Tucker
- Tensor train

Products (Hadamard, Kronecker, Khatri-Rao)

Hadamard (elementwise)

$$\boldsymbol{A} * \boldsymbol{B} = \begin{bmatrix} a_{1,1}b_{1,1} & a_{1,2}b_{1,2} & \cdots & a_{1,J}b_{1,J} \\ a_{2,1}b_{2,1} & a_{2,2}b_{2,2} & \cdots & a_{2,J}b_{2,J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I,1}b_{I,1} & a_{I,2}b_{I,2} & \cdots & a_{I,J}b_{I,J} \end{bmatrix}$$

$$\boldsymbol{A} \in \mathbb{R}^{I \times J}$$
$$\boldsymbol{B} \in \mathbb{R}^{I \times J}$$
$$\boldsymbol{A} * \boldsymbol{B} \in \mathbb{R}^{I \times J}$$

Kronecker

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{1,1}\boldsymbol{B} & a_{1,2}\boldsymbol{B} & \cdots & a_{1,J}\boldsymbol{B} \\ a_{2,1}\boldsymbol{B} & a_{2,2}\boldsymbol{B} & \cdots & a_{2,J}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I,1}\boldsymbol{B} & a_{I,2}\boldsymbol{B} & \cdots & a_{I,J}\boldsymbol{B} \end{bmatrix}$$

$$\boldsymbol{A} \in \mathbb{R}^{I \times J}$$
$$\boldsymbol{B} \in \mathbb{R}^{K \times L}$$
$$\boldsymbol{A} \otimes \boldsymbol{B} \in \mathbb{R}^{IK \times JL}$$

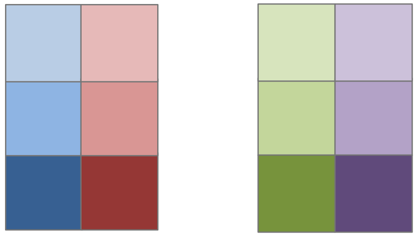
Khatri-Rao

$$\boldsymbol{A} \odot \boldsymbol{B} = \begin{bmatrix} a_{1,1}\boldsymbol{b}_1 & a_{1,2}\boldsymbol{b}_2 & \cdots & a_{1,K}\boldsymbol{b}_K \\ a_{2,1}\boldsymbol{b}_1 & a_{2,2}\boldsymbol{b}_2 & \cdots & a_{2,K}\boldsymbol{b}_K \\ \vdots & \vdots & \ddots & \vdots \\ a_{I,1}\boldsymbol{b}_1 & a_{I,2}\boldsymbol{b}_2 & \cdots & a_{I,K}\boldsymbol{b}_K \end{bmatrix}$$

$$\boldsymbol{A} \in \mathbb{R}^{I \times K}$$
$$\boldsymbol{B} \in \mathbb{R}^{J \times K}$$
$$\boldsymbol{A} \odot \boldsymbol{B} \in \mathbb{R}^{IJ \times K}$$

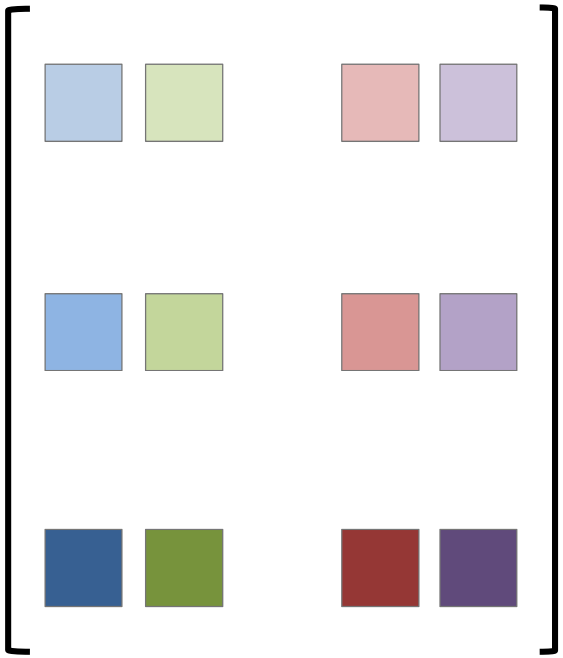
Hadamard (elementwise) product – Example

$$A \in \mathbb{R}^{3\times 2}$$
$$B \in \mathbb{R}^{3\times 2}$$
$$A * B \in \mathbb{R}^{3\times 2}$$

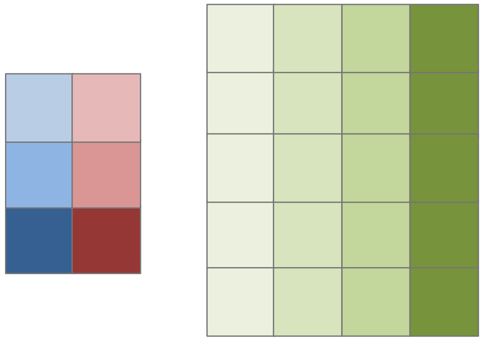


A B

$A * B =$

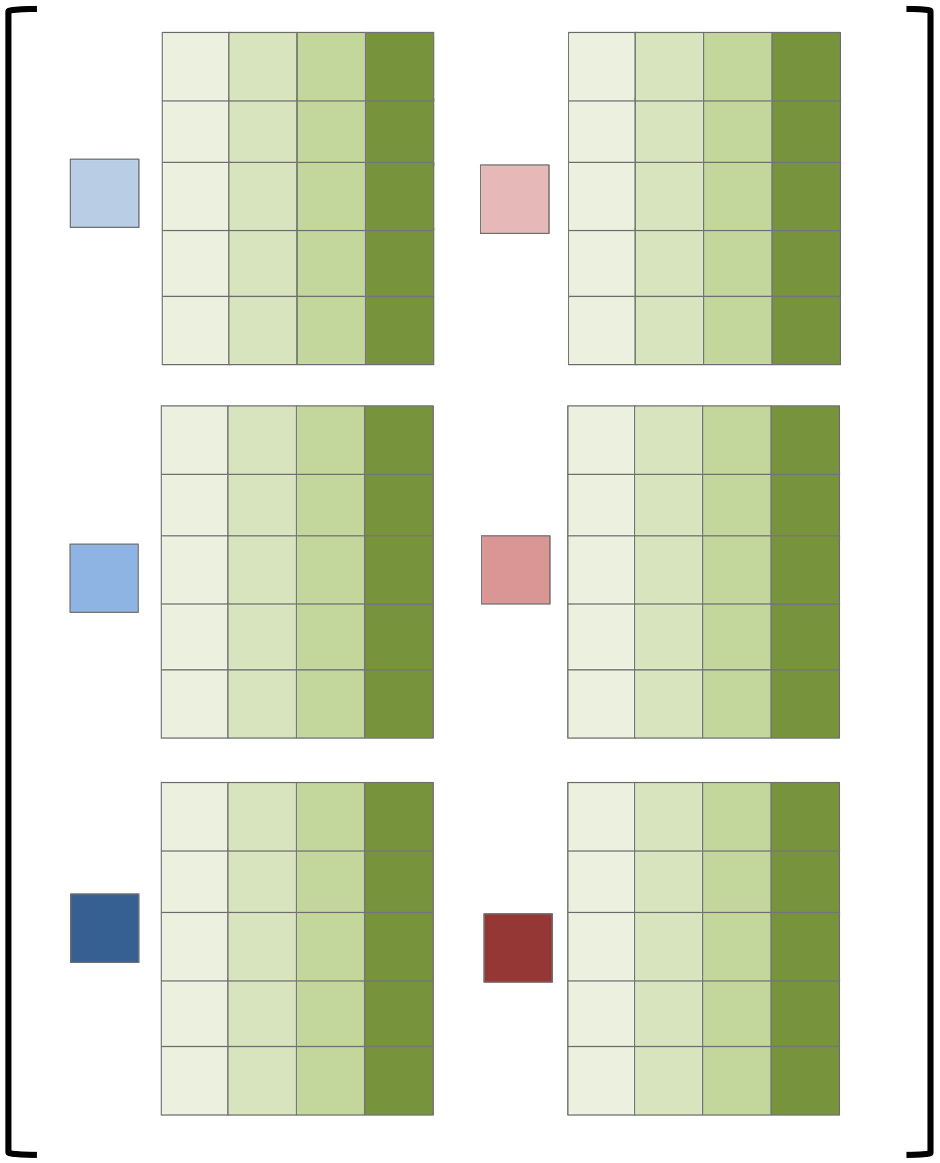


Kronecker product - Example



$\boldsymbol{A}$ $\boldsymbol{B}$

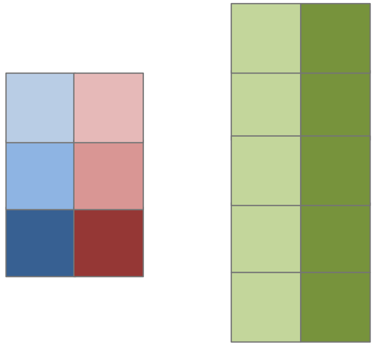
$$\boldsymbol{A} \otimes \boldsymbol{B} =$$



$$\boldsymbol{A} \in \mathbb{R}^{3\times 2}$$
$$\boldsymbol{B} \in \mathbb{R}^{5\times 4}$$
$$\boldsymbol{A} \otimes \boldsymbol{B} \in \mathbb{R}^{15\times 8}$$

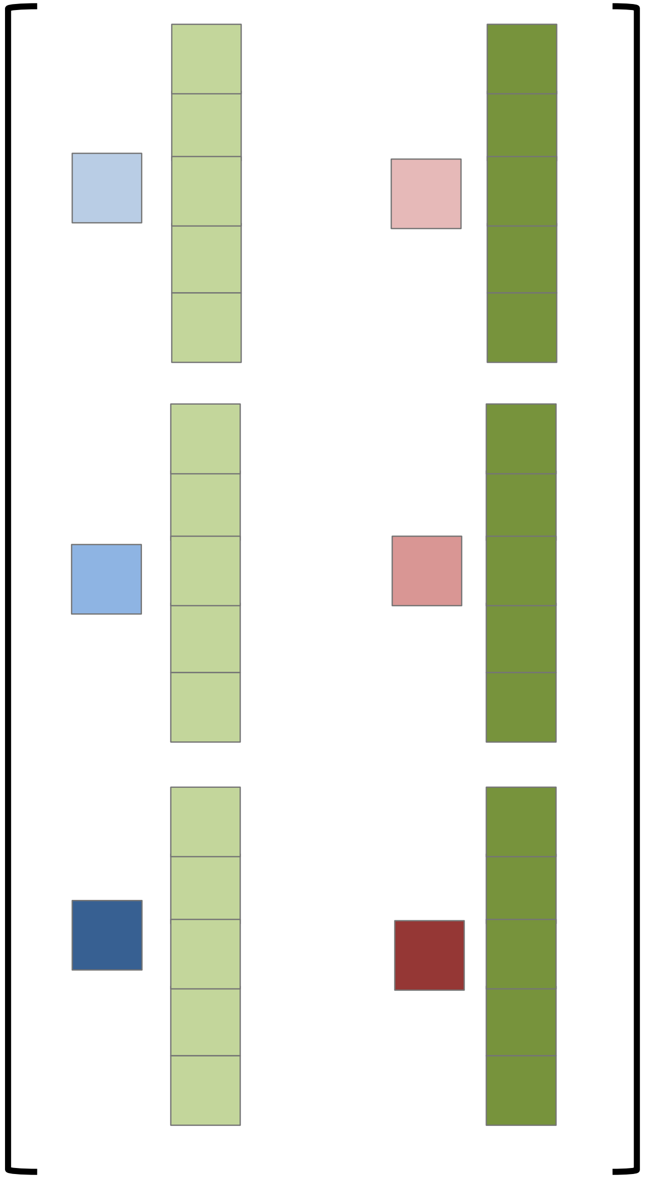
Khatri-Rao product - Example

$$\boldsymbol{A} \in \mathbb{R}^{3\times 2}$$
$$\boldsymbol{B} \in \mathbb{R}^{5\times 2}$$
$$\boldsymbol{A} \odot \boldsymbol{B} \in \mathbb{R}^{15\times 2}$$

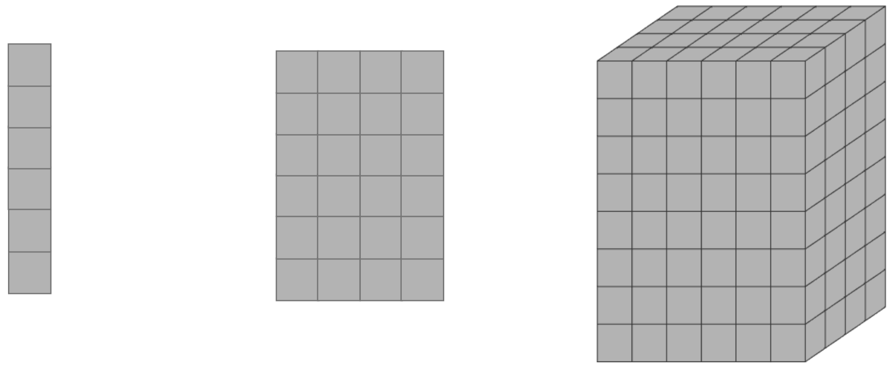


$\boldsymbol{A}$ $\boldsymbol{B}$

$$\boldsymbol{A} \odot \boldsymbol{B} =$$



Tensors



1st-order tensors

2nd-order tensors

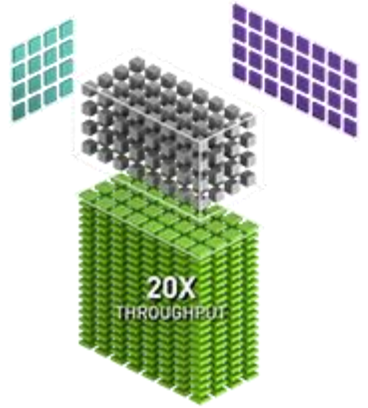
3rd-order tensors

…

TensorFlow

NVIDIA

TENSOR CORE



Images: 3D tensors
(width, height, color channels)

Videos: 4D tensors
(frame, width, height, color channels)

Tensors appear in various forms:

- Raw data
  (arrays of sensors, multidimensional channels)
- Data evolution over time window
  (sets of short sequences)
- Data in multiple coordinate systems
- Basis functions expansion

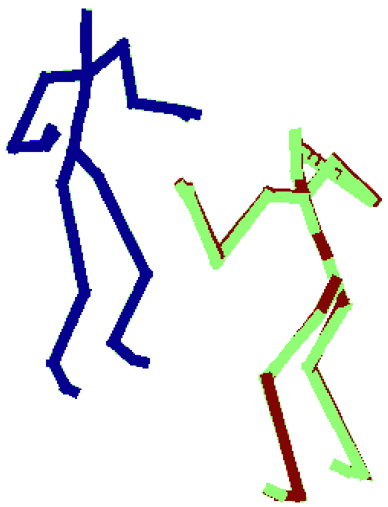
Tensor methods – Motivation

joint
agent
coordinate
sample
time step

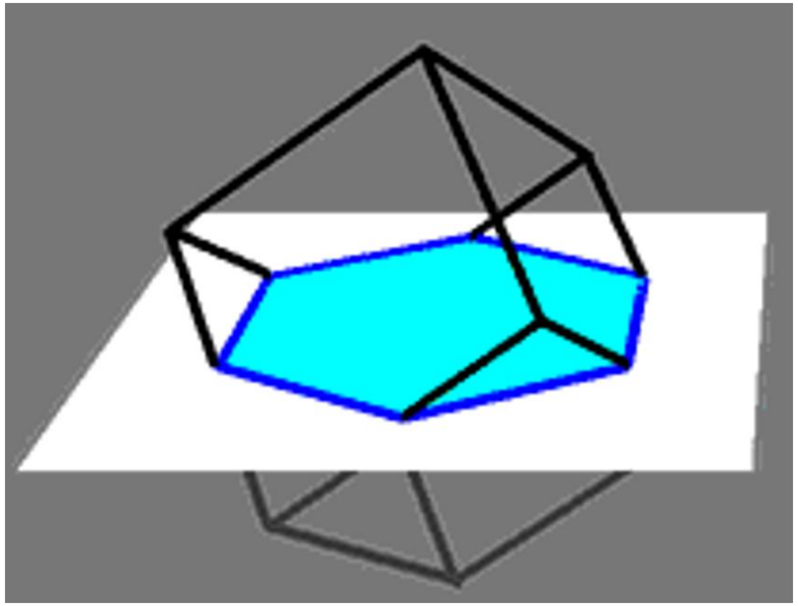
$\mathcal{X} \in \mathbb{R}^{10 \times 2 \times 31 \times 3 \times 100}$ $\boldsymbol{X} \in \mathbb{R}^{10 \times 18600}$

Tensor factorization

→ **Multiway analysis of the data**



Couldn't we simply vectorize/flatten our data before further processing?



Tensor data in robotics: Available processing tools

Neural networks

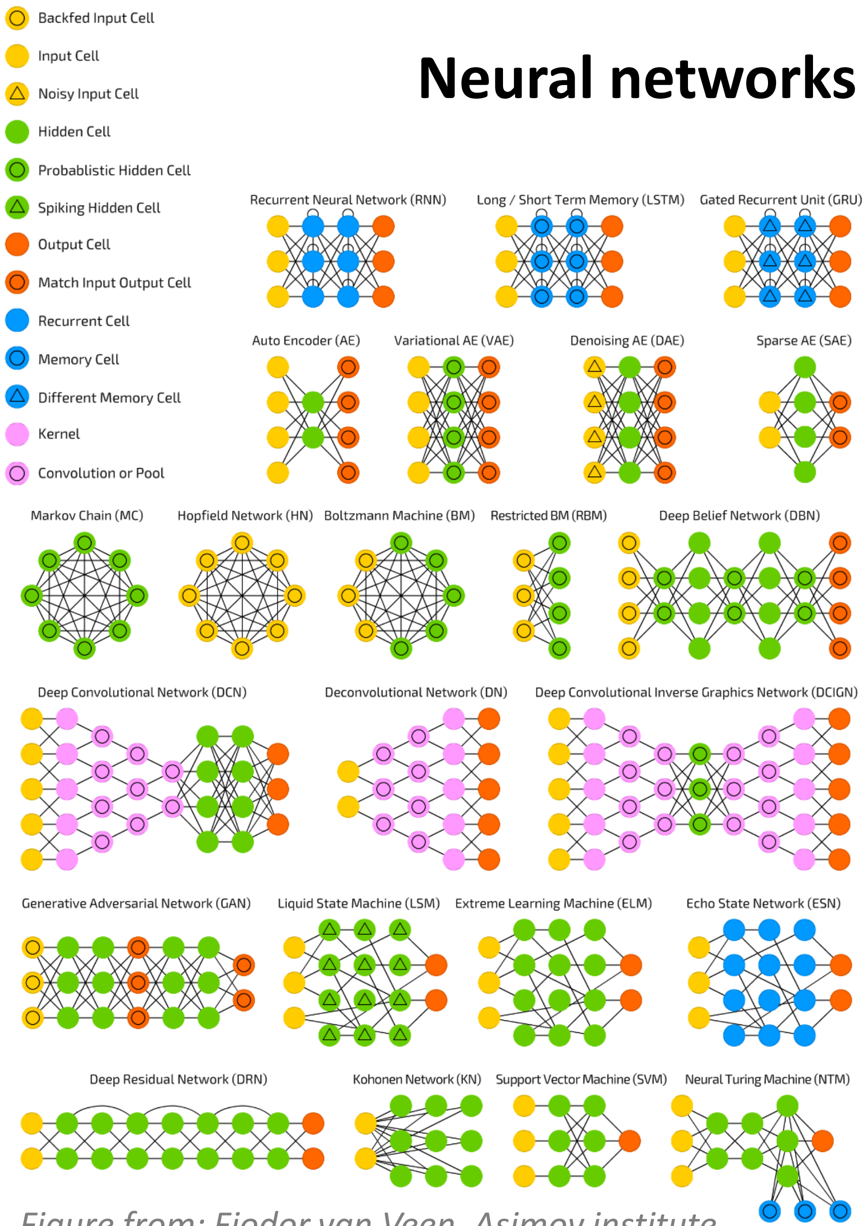
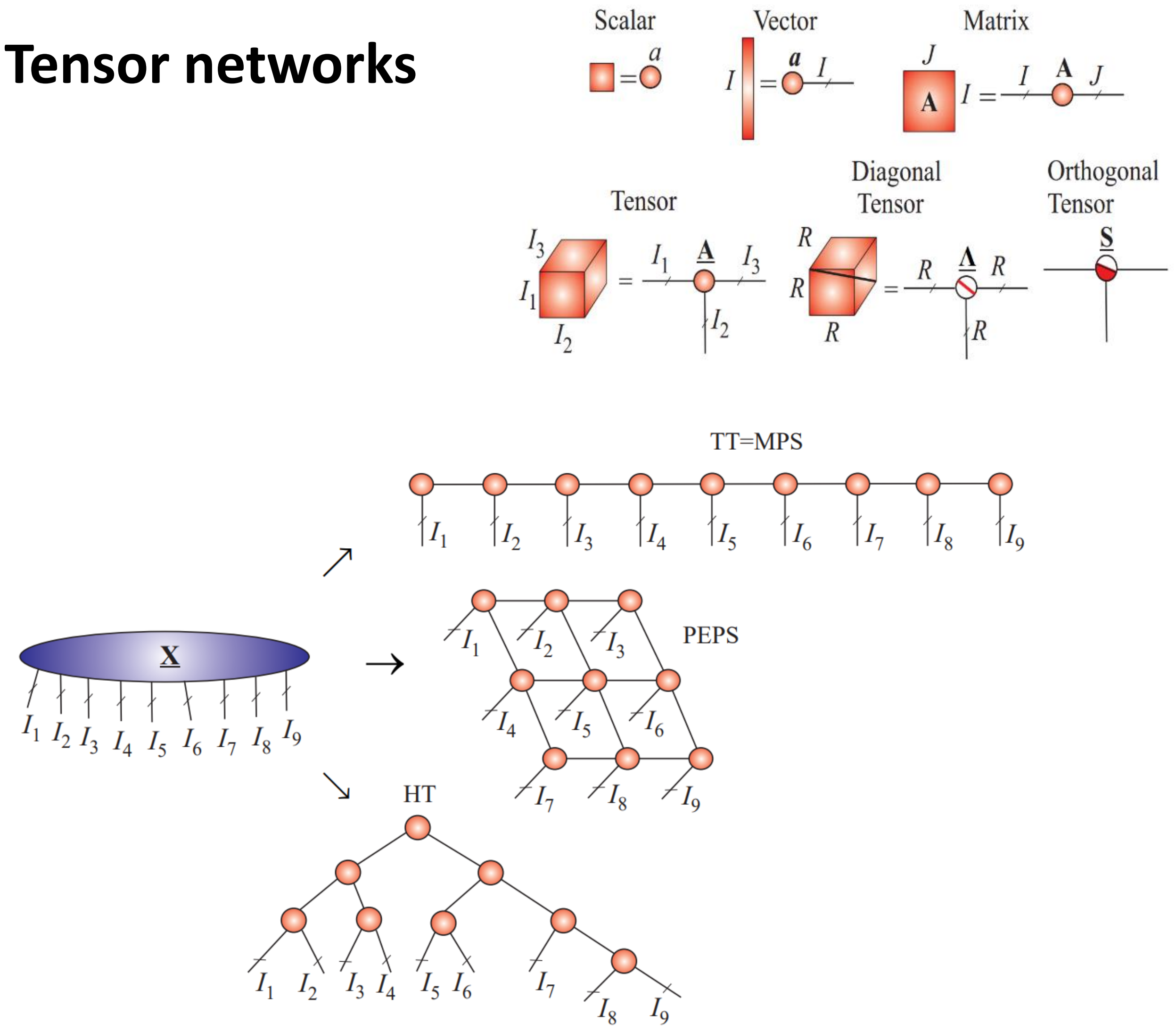


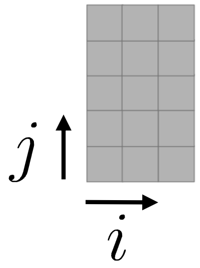
Figure from: Fjodor van Veen, Asimov institute

Tensor networks

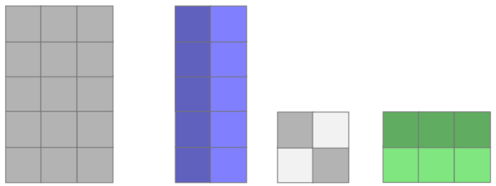


Figures from: Andrzej CICHOCKI (2014), Era of Big Data Processing: A New Approach via Tensor Networks and Tensor Decompositions

Separation of variables: a factorization problem



Matrix factorization with standard linear algebra:



$$\boldsymbol{X} = \boldsymbol{U} \; \boldsymbol{\Sigma} \; \boldsymbol{V}^\top$$

(singular value decomposition)

Rank-1 decomposition:

$$\boldsymbol{X}_{i,j} = \boldsymbol{U}_i \boldsymbol{V}_j$$ → Representation in a separable form

Rank-R decomposition:

$$\boldsymbol{X}_{i,j} = \sum_{r=1}^{R} \boldsymbol{U}_{i,r} \boldsymbol{V}_{j,r}$$

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^\top$$

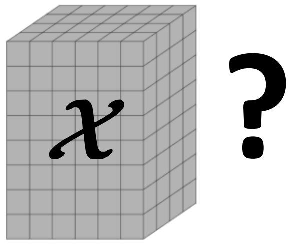
(in matrix form)

Extension to data with more indices (tensors):

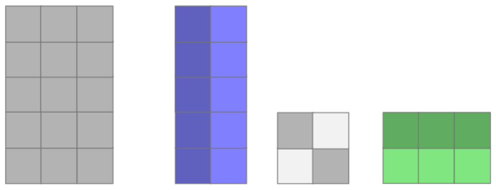
$$\boldsymbol{X}_{i,j,k,\ldots} = \sum_{r=1}^{R} \boldsymbol{U}_{i,r} \boldsymbol{V}_{j,r} \boldsymbol{W}_{k,r} \cdots$$

(CP decomposition)

Data structured as tensors

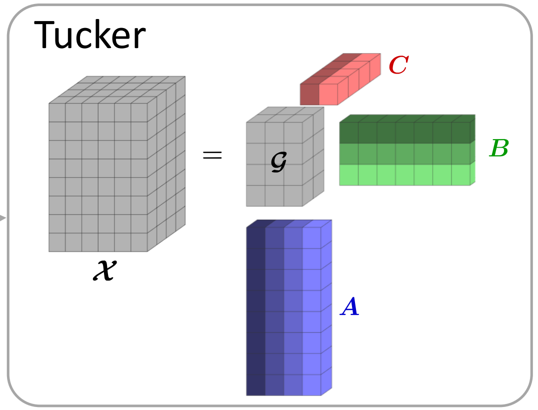


Matrix factorization with standard linear algebra:

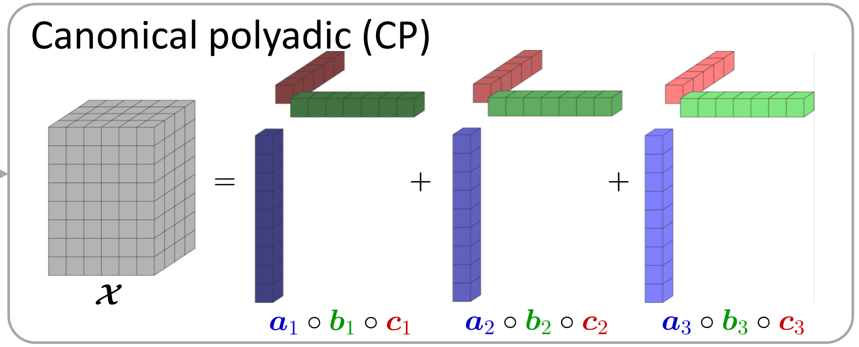


$$X = U \, \Sigma \, V^\top$$

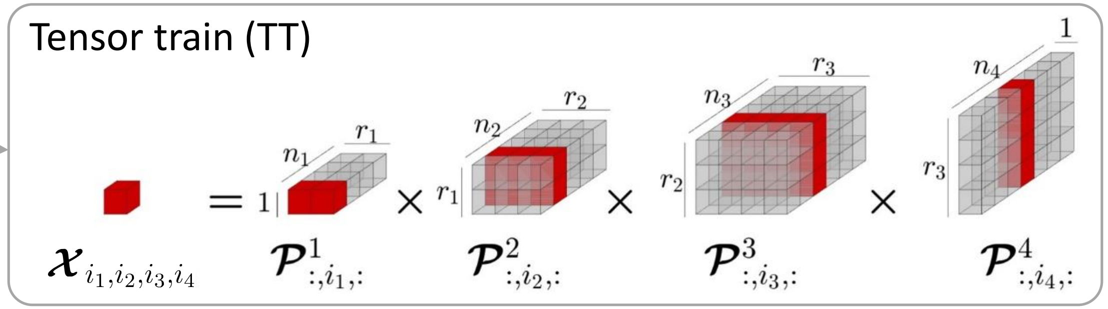
Tensor methods



Anima Anandkumar
(California Institute of Technology and NVIDIA)

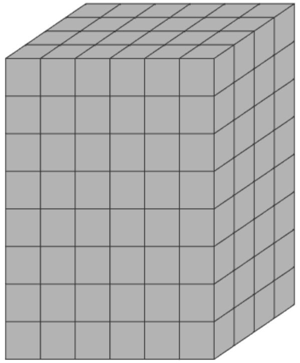


Lieven De Lathauwer
(KU Leuven)



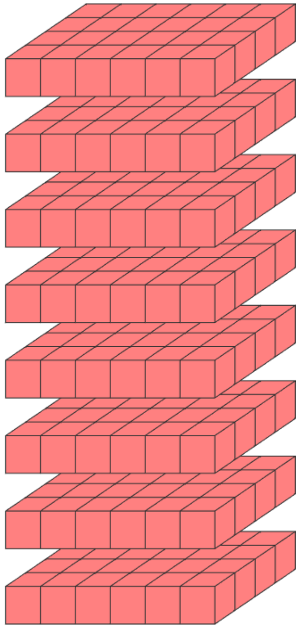
Ivan Oseledets
(Skolkovo Institute of Science and Technology)

Tensor indexing – Slices and fibers

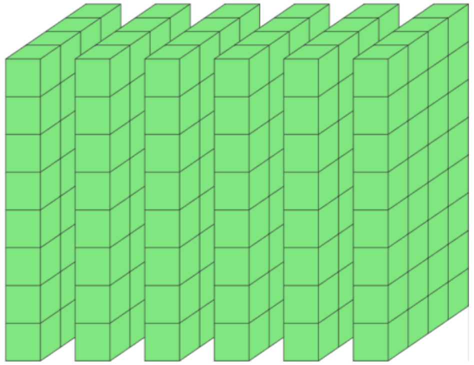


$\boldsymbol{\mathcal{X}} \in \mathbb{R}^{8\times 6\times 4}$

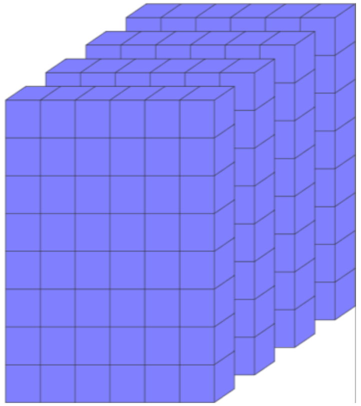
$\boldsymbol{X}_{i,:,:}$ (horizontal slice)

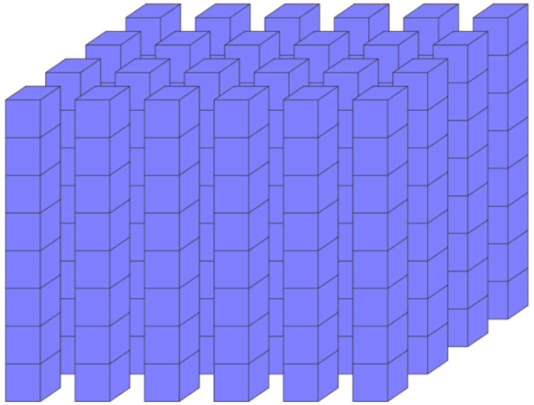


$\boldsymbol{X}_{:,j,:}$ (lateral slice)

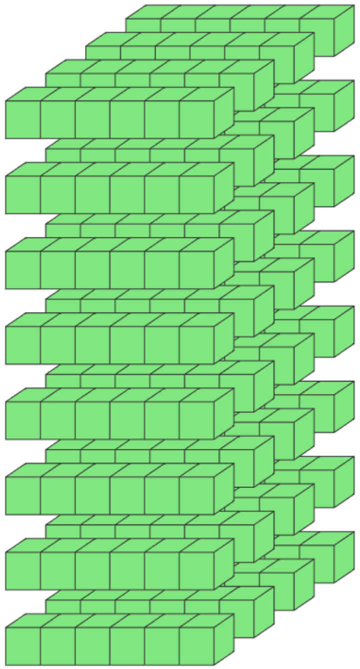


$\boldsymbol{X}_{:,:,k}$ (frontal slice)

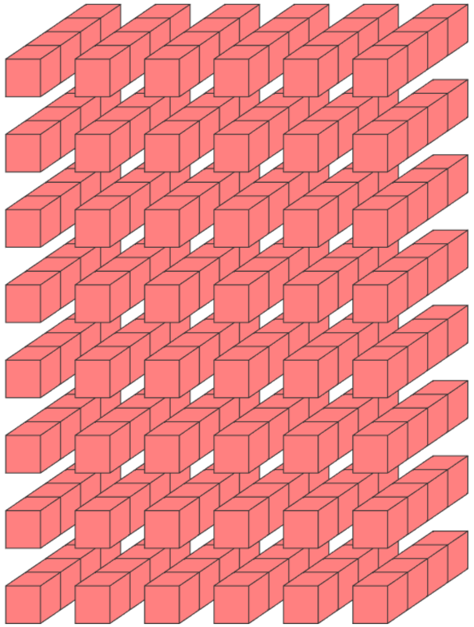




$\boldsymbol{x}_{:,j,k}$ (column fiber)



$\boldsymbol{x}_{i,:,k}$ (row fiber)

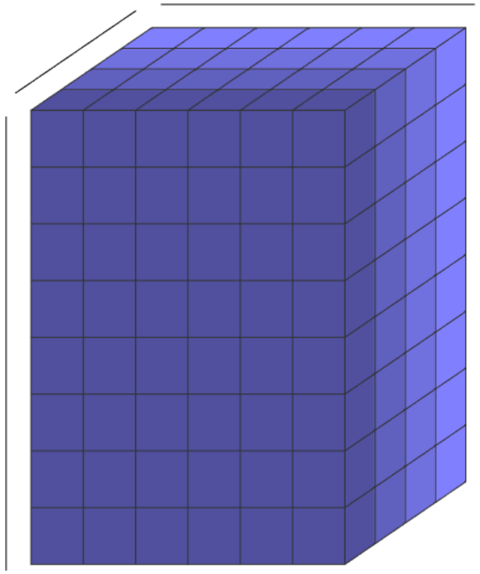


$\boldsymbol{x}_{i,j,:}$ (tube fiber)

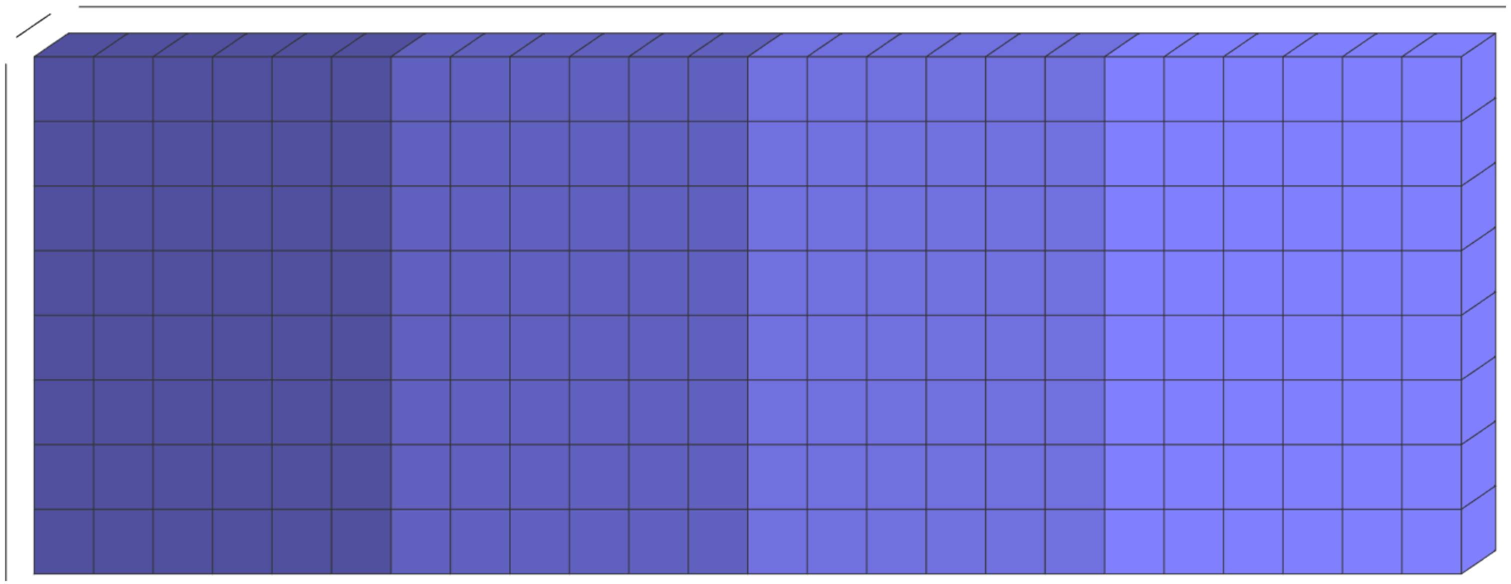
$\boldsymbol{\mathcal{X}}$	tensor
$\boldsymbol{X}$	matrix
$\boldsymbol{x}$	vector
x	scalar

Tensor matricization / unfolding

A matrix $\boldsymbol{X}_{(n)} \in \mathbb{R}^{I_n \times (I_1 \cdots I_{n-1} I_{n+1} \cdots I_N)}$ results from the mode-n matricization (unfolding) of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, which consists of turning the mode-n fibers of $\boldsymbol{\mathcal{X}}$ into the columns of a matrix $\boldsymbol{X}_{(n)}$.



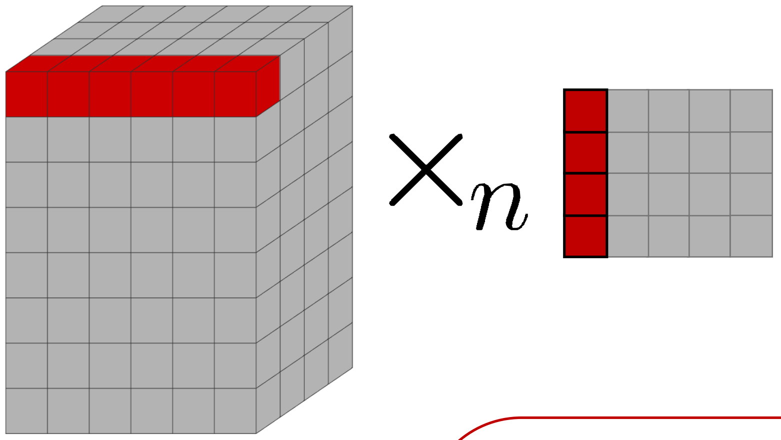
$\boldsymbol{\mathcal{X}} \in \mathbb{R}^{8 \times 6 \times 4}$



$\boldsymbol{X}_{(1)} \in \mathbb{R}^{8 \times 24}$

(mode-1 unfolding)

Mode-n product



$$\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$$

$$\boldsymbol{M} \in \mathbb{R}^{J \times I_n}$$

$$\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}$$

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \times_n \boldsymbol{M}$$

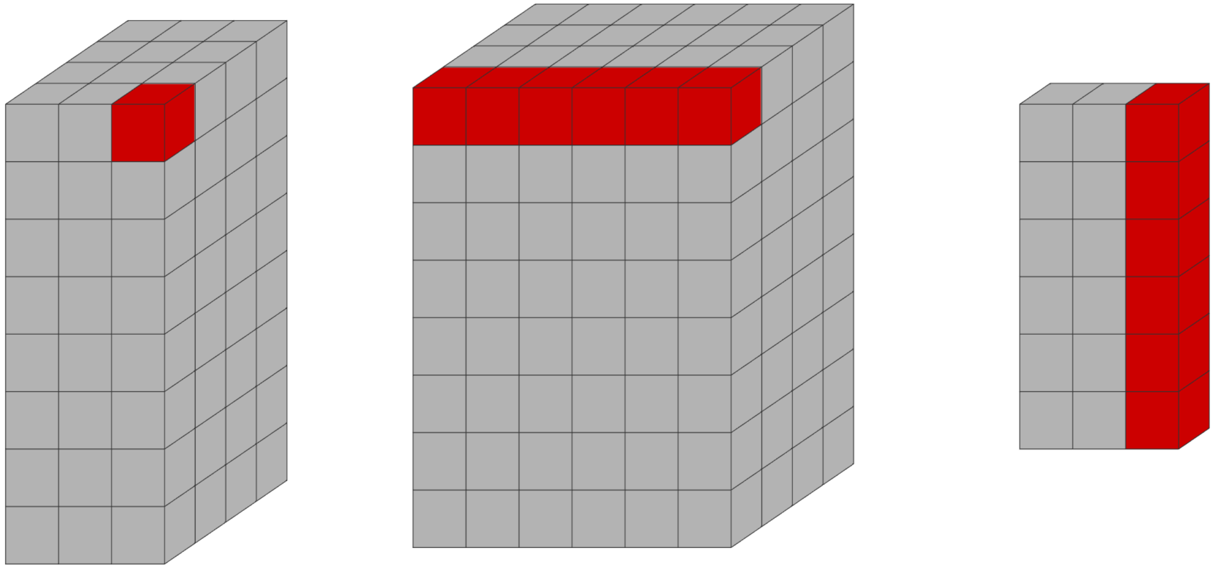
$$\boldsymbol{Y}_{(n)} = \boldsymbol{M}\boldsymbol{X}_{(n)} \quad \text{(matricized form)}$$

$$y_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_n} x_{i_1,\ldots,i_N}\, m_{j,i_n} \quad \text{(elementwise)}$$

Intuitively, the operation corresponds to multiplying each mode-n fiber of $\boldsymbol{\mathcal{X}}$ by the matrix $\boldsymbol{M}$.

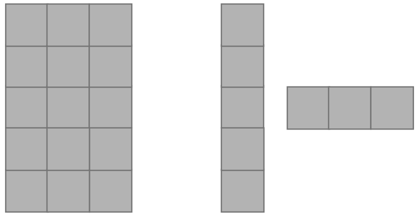
Mode-n product – Example

$\mathcal{X} \in \mathbb{R}^{8\times6\times4}$
$\boldsymbol{M} \in \mathbb{R}^{6\times3}$
$\mathcal{Y} \in \mathbb{R}^{8\times3\times4}$



$$\mathcal{Y} = \mathcal{X} \times_2 \boldsymbol{M}$$

Outer product and inner product

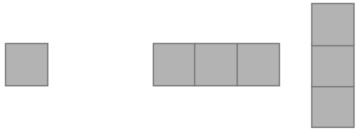


$$\boldsymbol{X} = \boldsymbol{a} \; \boldsymbol{b}^\top = \boldsymbol{a} \circ \boldsymbol{b}$$

(outer product)

The **outer product** of two vectors $\boldsymbol{a} \in \mathbb{R}^I$ and $\boldsymbol{b} \in \mathbb{R}^J$ results in a matrix $\boldsymbol{X} \in \mathbb{R}^{I \times J}$ denoted by $\boldsymbol{X} = \boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{a}\boldsymbol{b}^\top$.

The **outer product** of three (or more) vectors $\boldsymbol{a} \in \mathbb{R}^I$, $\boldsymbol{b} \in \mathbb{R}^J$ and $\boldsymbol{c} \in \mathbb{R}^K$ results in a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$ denoted by $\boldsymbol{\mathcal{X}} = \boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c}$ with elements $x_{i,j,k} = a_i \, b_j \, c_k$.



$$x = \boldsymbol{a}^\top \; \boldsymbol{b} = \langle \, \boldsymbol{a} \, , \, \boldsymbol{b} \, \rangle$$

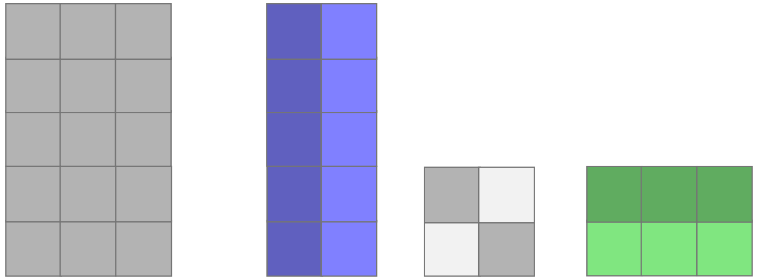
(inner product)

The **inner product** of two vectors $\boldsymbol{a} \in \mathbb{R}^I$ and $\boldsymbol{b} \in \mathbb{R}^I$ results in a scalar $x = \langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}^\top \boldsymbol{b} = \sum_{i=1}^{I} a_i b_i$.

The formulation can be extended to tensors $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ of the same size. We have

$$\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \langle \boldsymbol{A}_{(n)}, \boldsymbol{B}_{(n)} \rangle = \langle \mathrm{vec}(\boldsymbol{\mathcal{A}}), \mathrm{vec}(\boldsymbol{\mathcal{B}}) \rangle.$$

Singular value decomposition (SVD)



$$\boldsymbol{X} = \boldsymbol{U} \quad \boldsymbol{\Sigma} \quad \boldsymbol{V}^\top$$

$$= \sigma_1^2 \, \boldsymbol{u}_1 \, \boldsymbol{v}_1^\top + \sigma_2^2 \, \boldsymbol{u}_2 \, \boldsymbol{v}_2^\top$$

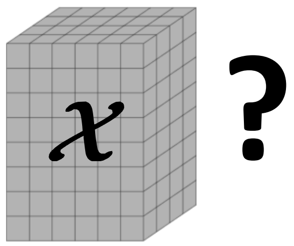
$$= \tilde{\boldsymbol{u}}_1 \, \tilde{\boldsymbol{v}}_1^\top + \tilde{\boldsymbol{u}}_2 \, \tilde{\boldsymbol{v}}_2^\top$$

$$= \tilde{\boldsymbol{u}}_1 \circ \tilde{\boldsymbol{v}}_1 + \tilde{\boldsymbol{u}}_2 \circ \tilde{\boldsymbol{v}}_2$$

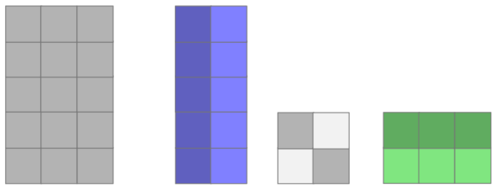
$$\tilde{\boldsymbol{u}}_i = \sigma_i \boldsymbol{u}_i$$

$$\tilde{\boldsymbol{v}}_i = \sigma_i \boldsymbol{v}_i$$

Data structured as tensors



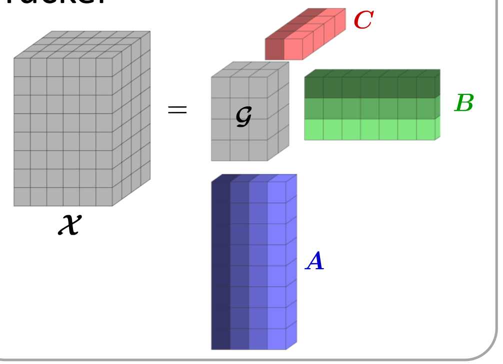
Matrix factorization with standard linear algebra:



$$X = U \; \Sigma \; V^{\top}$$

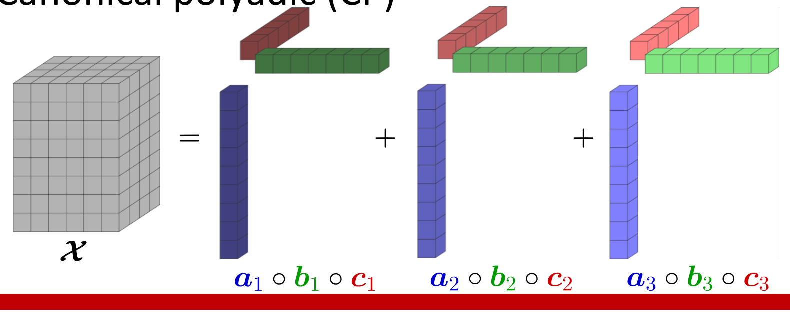
Tensor methods

Tucker



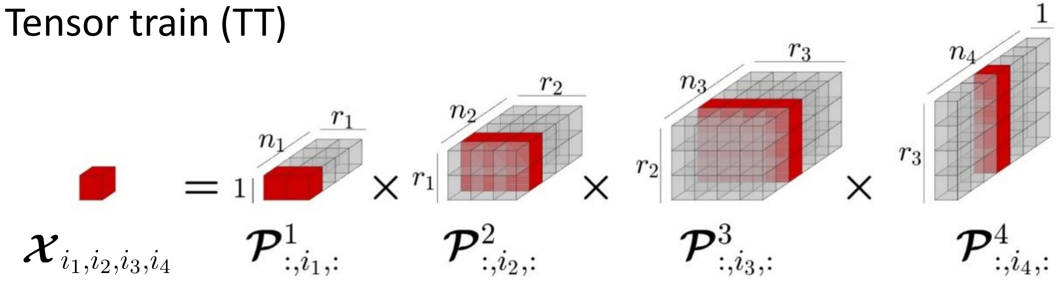
Anima Anandkumar
(California Institute of Technology and NVIDIA)

Canonical polyadic (CP)



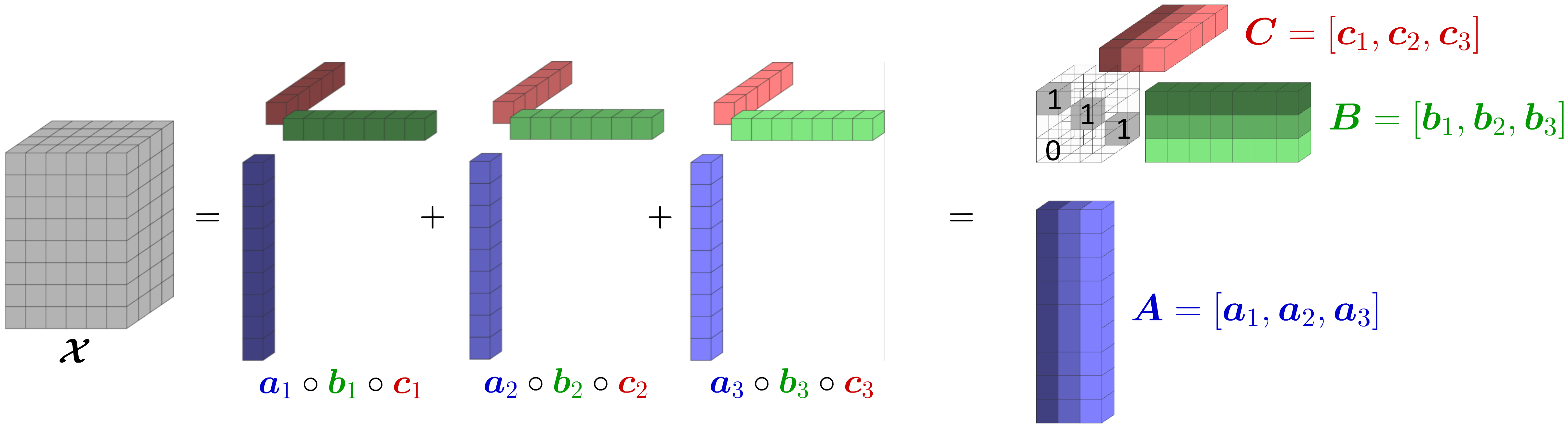
Lieven De Lathauwer
(KU Leuven)

Tensor train (TT)



Ivan Oseledets
(Skolkovo Institute of Science and Technology)

CP decomposition



CP decomposition

$$\begin{aligned}\boldsymbol{\mathcal{X}} &= \textstyle\sum_{j=1}^{r} \boldsymbol{a}_j \circ \boldsymbol{b}_j \circ \boldsymbol{c}_j \\ &= [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]\end{aligned}$$

Matricized form:

$$\begin{aligned}\boldsymbol{X}_{(1)} &= \boldsymbol{A}(\boldsymbol{C} \odot \boldsymbol{B})^\top \\ \boldsymbol{X}_{(2)} &= \boldsymbol{B}(\boldsymbol{C} \odot \boldsymbol{A})^\top \\ \boldsymbol{X}_{(3)} &= \boldsymbol{C}(\boldsymbol{B} \odot \boldsymbol{A})^\top\end{aligned}$$

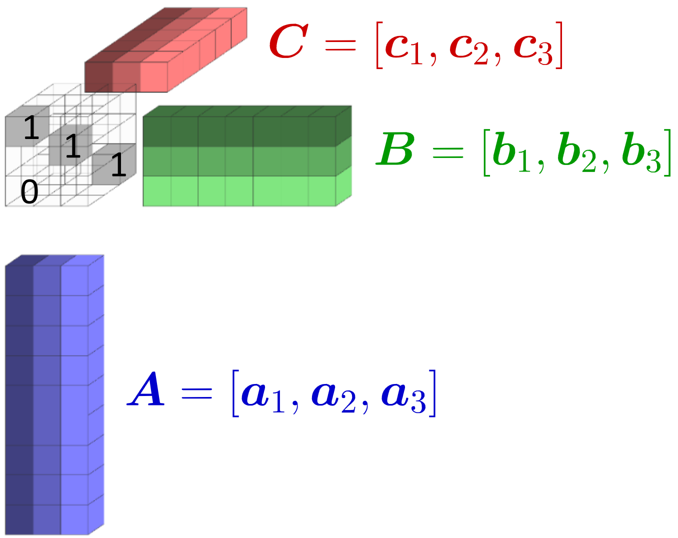
Vectorized form:

$$\mathrm{vec}(\boldsymbol{\mathcal{X}}) = (\boldsymbol{C} \odot \boldsymbol{B} \odot \boldsymbol{A})\, \mathbf{1}_R$$

Elementwise:

$$x_{i_1,i_2,i_3} = \textstyle\sum_{j=1}^{r} a_{i_1,j}\; b_{i_2,j}\; c_{i_3,j}$$

$$\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$$
$$\boldsymbol{A} \in \mathbb{R}^{n_1 \times r}$$
$$\boldsymbol{B} \in \mathbb{R}^{n_2 \times r}$$
$$\boldsymbol{C} \in \mathbb{R}^{n_3 \times r}$$



$\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_r]$ is called a factor matrix.

The **tensor rank** r corresponds to the smallest number of components required in the CP decomposition.

CP parameters estimation: Alternating least squares (ALS)

The CP decomposition can be solved by alternating least squares (ALS), by repeating

$$\boldsymbol{A} \leftarrow \arg\min_{\boldsymbol{A}} \left\|\boldsymbol{X}_{(1)} - \boldsymbol{A}(\boldsymbol{C} \odot \boldsymbol{B})^\top\right\|_F^2$$

$$\boldsymbol{B} \leftarrow \arg\min_{\boldsymbol{B}} \left\|\boldsymbol{X}_{(2)} - \boldsymbol{B}(\boldsymbol{C} \odot \boldsymbol{A})^\top\right\|_F^2$$

$$\boldsymbol{C} \leftarrow \arg\min_{\boldsymbol{C}} \left\|\boldsymbol{X}_{(3)} - \boldsymbol{C}(\boldsymbol{B} \odot \boldsymbol{A})^\top\right\|_F^2$$

until convergence, yielding the update rules

$$\boldsymbol{A} \leftarrow \boldsymbol{X}_{(1)}\left((\boldsymbol{C} \odot \boldsymbol{B})^\top\right)^\dagger$$

$$\boldsymbol{B} \leftarrow \boldsymbol{X}_{(2)}\left((\boldsymbol{C} \odot \boldsymbol{A})^\top\right)^\dagger$$

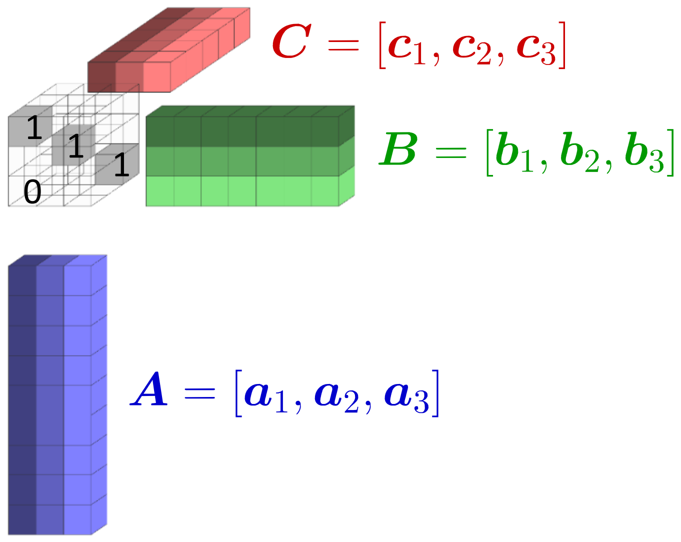
$$\boldsymbol{C} \leftarrow \boldsymbol{X}_{(3)}\left((\boldsymbol{B} \odot \boldsymbol{A})^\top\right)^\dagger$$

$\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$

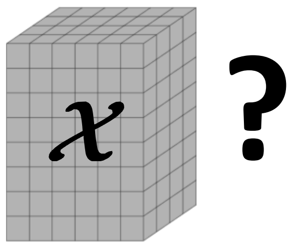
$\boldsymbol{A} \in \mathbb{R}^{n_1 \times r}$

$\boldsymbol{B} \in \mathbb{R}^{n_2 \times r}$

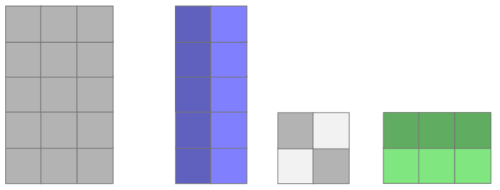
$\boldsymbol{C} \in \mathbb{R}^{n_3 \times r}$



Data structured as tensors

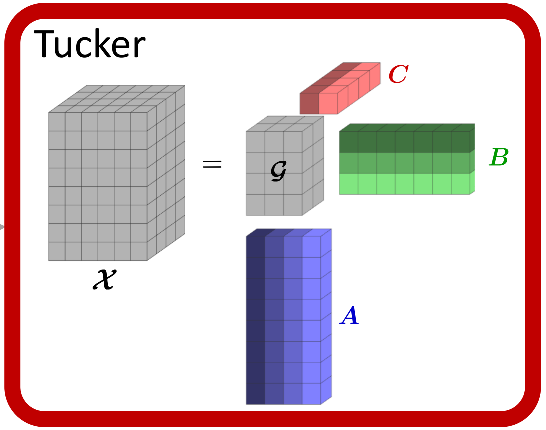


Matrix factorization with standard linear algebra:

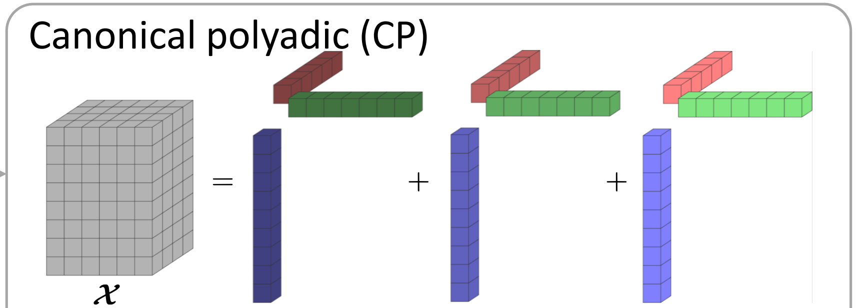


$$X = U\ \Sigma\ V^\top$$

Tensor methods

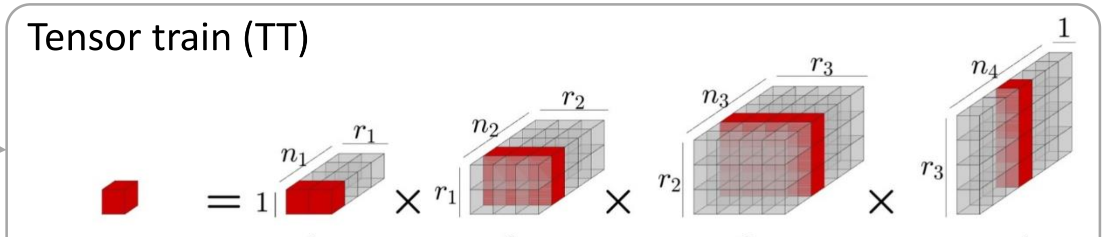


Anima Anandkumar
(California Institute of Technology and NVIDIA)



$a_1 \circ b_1 \circ c_1$ $a_2 \circ b_2 \circ c_2$ $a_3 \circ b_3 \circ c_3$

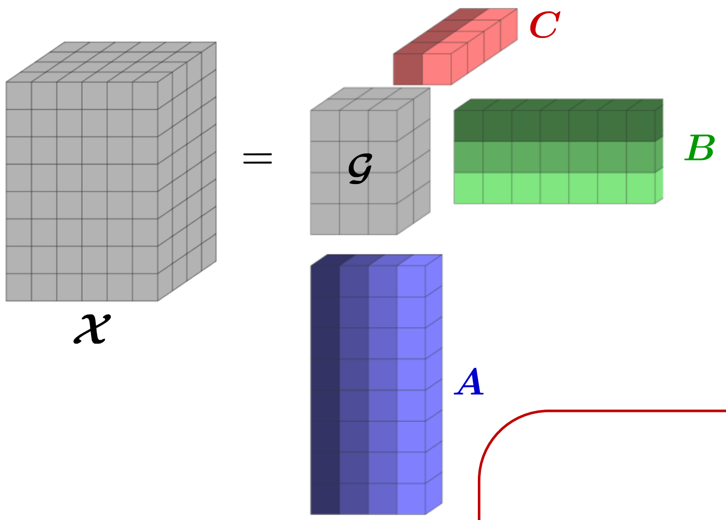
Lieven De Lathauwer
(KU Leuven)



$\mathcal{X}_{i_1,i_2,i_3,i_4} = \mathcal{P}^1_{:,i_1,:} \times \mathcal{P}^2_{:,i_2,:} \times \mathcal{P}^3_{:,i_3,:} \times \mathcal{P}^4_{:,i_4,:}$

Ivan Oseledets
(Skolkovo Institute of Science and Technology)

Tucker decomposition



Core tensor

$$\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$$
$$\boldsymbol{\mathcal{G}} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$$
$$\boldsymbol{A} \in \mathbb{R}^{n_1 \times r_1} \qquad \boldsymbol{A}^\top \boldsymbol{A} = \boldsymbol{I}_{r_1}$$
$$\boldsymbol{B} \in \mathbb{R}^{n_2 \times r_2} \qquad \boldsymbol{B}^\top \boldsymbol{B} = \boldsymbol{I}_{r_2}$$
$$\boldsymbol{C} \in \mathbb{R}^{n_3 \times r_3} \qquad \boldsymbol{C}^\top \boldsymbol{C} = \boldsymbol{I}_{r_3}$$

$$\begin{aligned}\boldsymbol{\mathcal{X}} &= \sum_{j_1=1}^{r_1} \sum_{j_2=1}^{r_2} \sum_{j_3=1}^{r_3} g_{j_1,j_2,j_3} \, \boldsymbol{a}_{j_1} \circ \boldsymbol{b}_{j_2} \circ \boldsymbol{c}_{j_3} \\ &= \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C} \\ &= [\![\boldsymbol{\mathcal{G}}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]\end{aligned}$$

Matricized form:

$$\begin{aligned}\boldsymbol{X}_{(1)} &= \boldsymbol{A}\boldsymbol{G}_{(1)}(\boldsymbol{C} \otimes \boldsymbol{B})^\top \\ \boldsymbol{X}_{(2)} &= \boldsymbol{B}\boldsymbol{G}_{(2)}(\boldsymbol{C} \otimes \boldsymbol{A})^\top \\ \boldsymbol{X}_{(3)} &= \boldsymbol{C}\boldsymbol{G}_{(3)}(\boldsymbol{B} \otimes \boldsymbol{A})^\top\end{aligned}$$

Elementwise:

$$x_{i_1,i_2,i_3} = \sum_{j_1=1}^{r_1} \sum_{j_2=1}^{r_2} \sum_{j_3=1}^{r_3} g_{j_1,j_2,j_3} \, a_{i_1,j_1} \, b_{i_2,j_2} \, c_{i_3,j_3}$$

Tucker parameters estimation: Higher-order SVD (HO-SVD)

$$\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$$
$$\boldsymbol{\mathcal{G}} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$$
$$\boldsymbol{A} \in \mathbb{R}^{n_1 \times r_1} \quad \boldsymbol{A}^\top \boldsymbol{A} = \boldsymbol{I}_{r_1}$$
$$\boldsymbol{B} \in \mathbb{R}^{n_2 \times r_2} \quad \boldsymbol{B}^\top \boldsymbol{B} = \boldsymbol{I}_{r_2}$$
$$\boldsymbol{C} \in \mathbb{R}^{n_3 \times r_3} \quad \boldsymbol{C}^\top \boldsymbol{C} = \boldsymbol{I}_{r_3}$$

The Tucker decomposition can be estimated by computing the truncated singular value decompositions (SVD)

$$\boldsymbol{X}_{(1)} = \boldsymbol{A}\boldsymbol{S}\boldsymbol{V}^\top$$
$$\boldsymbol{X}_{(2)} = \boldsymbol{B}\boldsymbol{S}\boldsymbol{V}^\top$$
$$\boldsymbol{X}_{(3)} = \boldsymbol{C}\boldsymbol{S}\boldsymbol{V}^\top$$

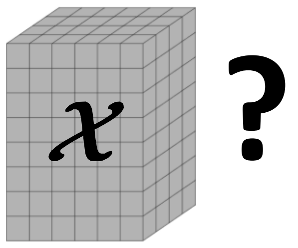
with $\boldsymbol{\mathcal{G}}$ finally evaluated as

$$\boldsymbol{\mathcal{G}} \leftarrow \boldsymbol{\mathcal{X}} \times_1 \boldsymbol{A}^\top \times_2 \boldsymbol{B}^\top \times_3 \boldsymbol{C}^\top$$

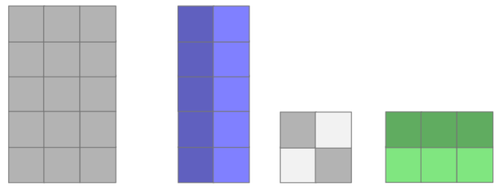
In contrast to CP, the Tucker decomposition is generally not unique

→ A, B and C constrained to be orthogonal matrices

Data structured as tensors



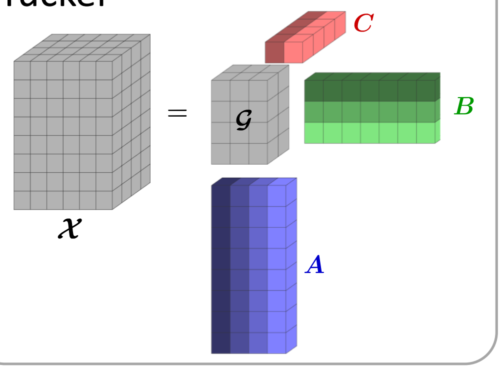
Matrix factorization with standard linear algebra:



$$X = U \; \Sigma \; V^{\top}$$

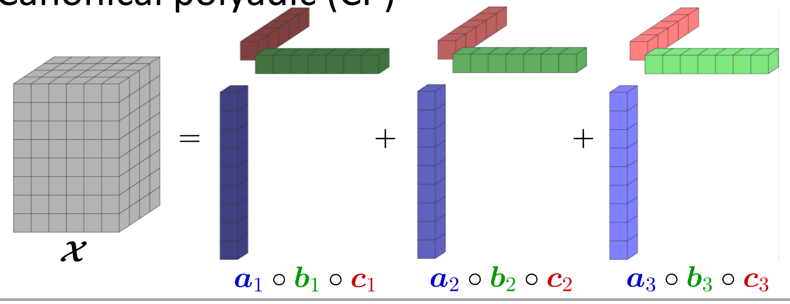
Tensor methods

Tucker



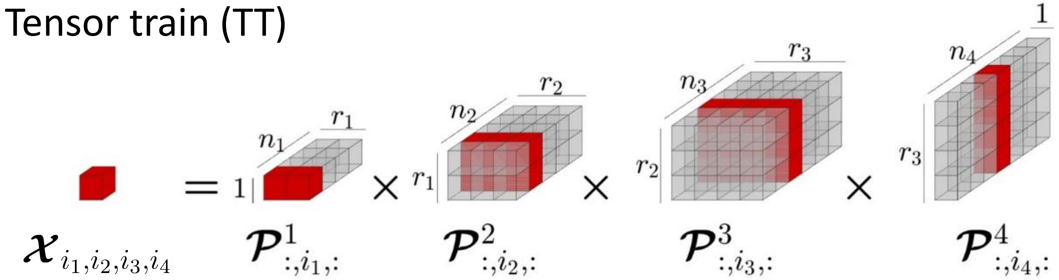
Anima Anandkumar
(California Institute of Technology and NVIDIA)

Canonical polyadic (CP)



Lieven De Lathauwer
(KU Leuven)

Tensor train (TT)

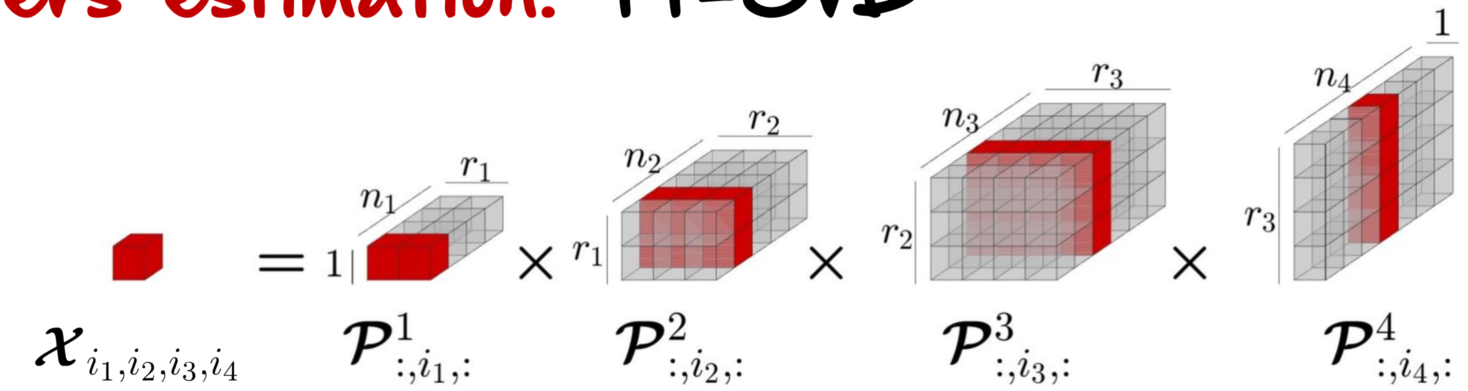


Ivan Oseledets
(Skolkovo Institute of Science and Technology)

Tensor train parameters estimation: TT-SVD

$\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$

$\boldsymbol{\mathcal{P}}^k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$



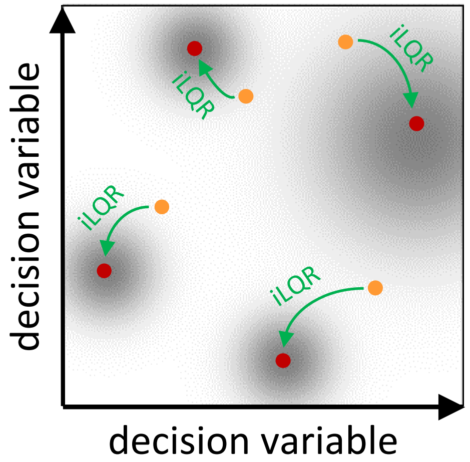
- $\boldsymbol{\mathcal{X}}$ is reshaped as a $n_1 \times n_2 n_3 n_4$ matrix $\boldsymbol{X}_1$

- $\boldsymbol{X}_1 \approx \boldsymbol{U}_1 \boldsymbol{S}_1 \boldsymbol{V}_1^\top$, where $\boldsymbol{U}_1$ is a $n_1 \times r_1$ matrix, reshaped as 1st core $\boldsymbol{\mathcal{P}}^1$
- $\boldsymbol{S}_1 \boldsymbol{V}_1^\top$ is a $r_1 \times n_2 n_3 n_4$ matrix reshaped into a $r_1 n_2 \times n_3 n_4$ matrix $\boldsymbol{X}_2$

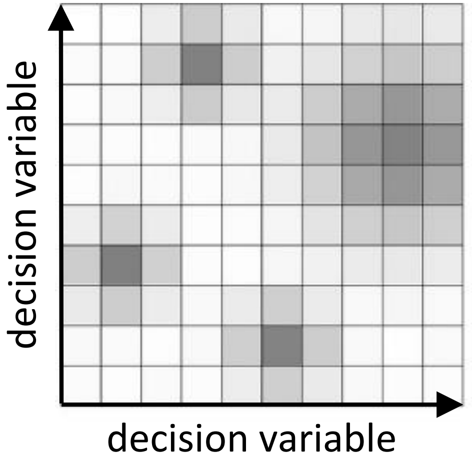
- $\boldsymbol{X}_2 \approx \boldsymbol{U}_2 \boldsymbol{S}_2 \boldsymbol{V}_2^\top$, where $\boldsymbol{U}_2$ is a $r_1 n_2 \times r_2$ matrix, reshaped as 2nd core $\boldsymbol{\mathcal{P}}^2$
- $\boldsymbol{S}_2 \boldsymbol{V}_2^\top$ is a $r_2 \times n_3 n_4$ matrix reshaped into a $r_2 n_3 \times n_4$ matrix $\boldsymbol{X}_3$

- $\boldsymbol{X}_3 \approx \boldsymbol{U}_3 \boldsymbol{S}_3 \boldsymbol{V}_3^\top$, where $\boldsymbol{U}_3$ is a $r_2 n_3 \times r_3$ matrix, reshaped as 3rd core $\boldsymbol{\mathcal{P}}^3$
- $\boldsymbol{S}_3 \boldsymbol{V}_3^\top$ is a $r_3 \times n_4$ matrix, reshaped as 4th core $\boldsymbol{\mathcal{P}}^4$

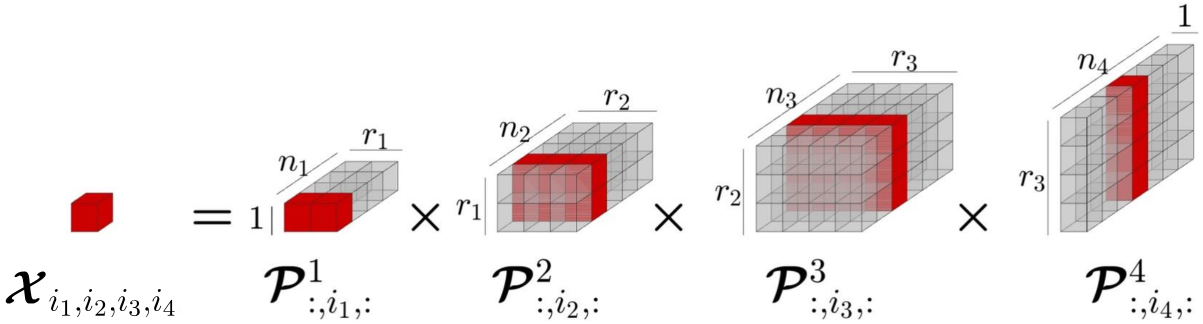
Example: Tensor train for global optimization



For 2D decision variable:



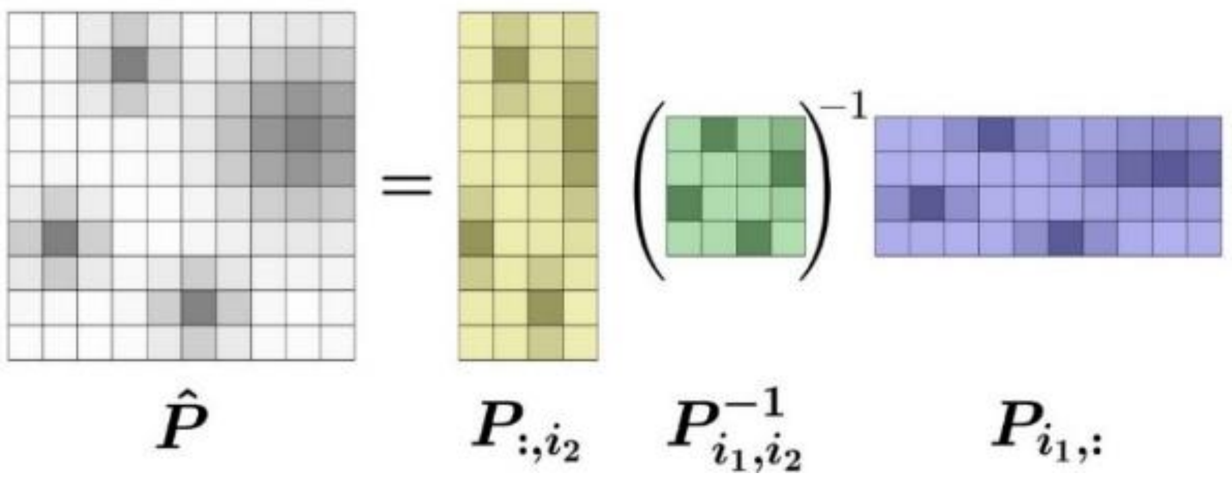
For nD decision variable:

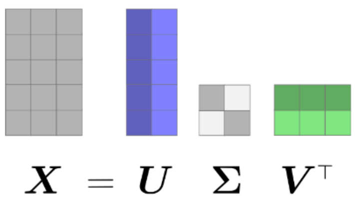


Tensor train (TT)

Example: Tensor train for global optimization

Cross approximation (skeleton decomposition) of a probability distribution:

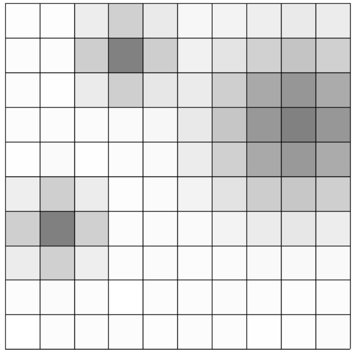




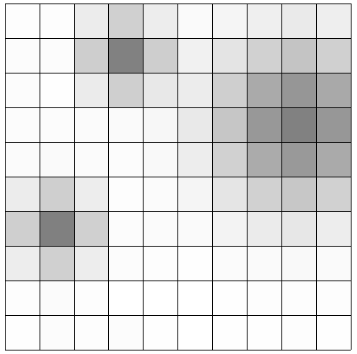
→ Can be used to approximate an unknown matrix **by querying rows and columns of the matrix** in an iterative manner, while estimating the rank of the matrix

Example: Tensor train for global optimization

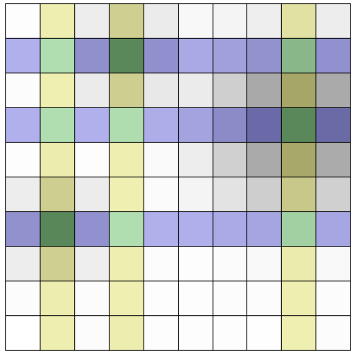
Cross approximation (skeleton decomposition) of a probability distribution:



Original distribution



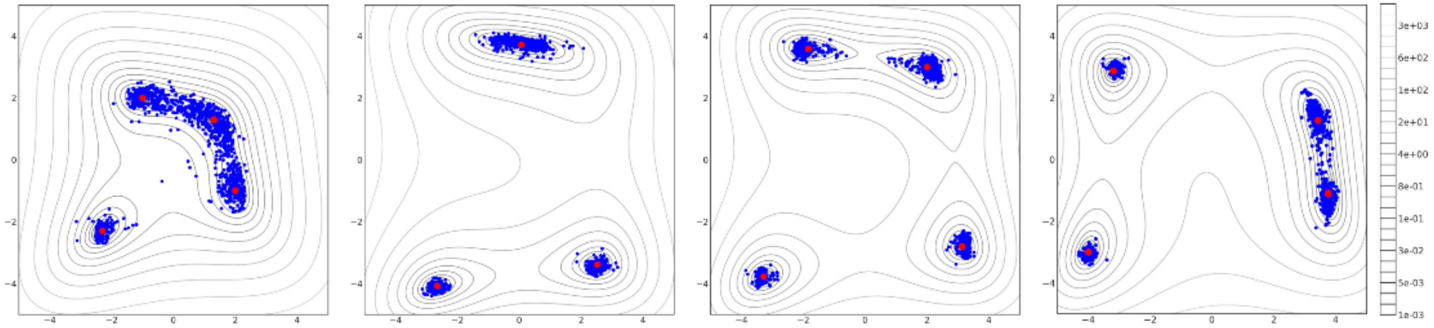
Reconstructed distribution
(rank 3 approximation)



→ Can be used to approximate an unknown matrix **by querying rows and columns of the matrix** in an iterative manner, while estimating the rank of the matrix

Example: Tensor train for global optimization

Optimization benchmarks with Himmelblau functions

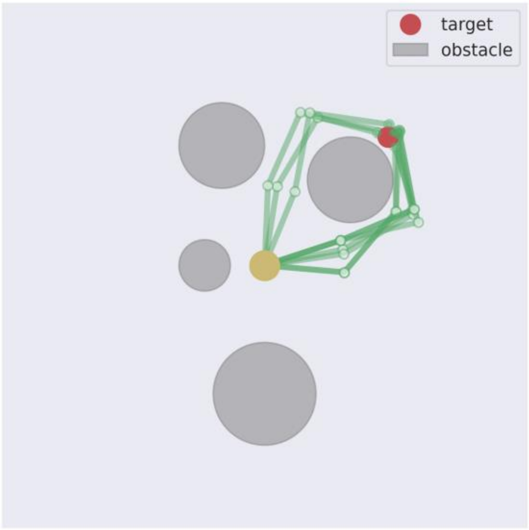


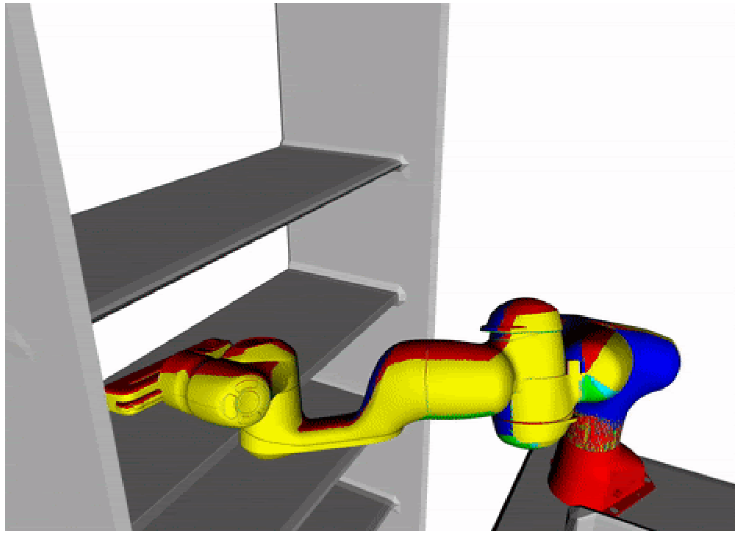
task param. (3,3) task param. (3,14) task param. (7,11) task param. (13,5)

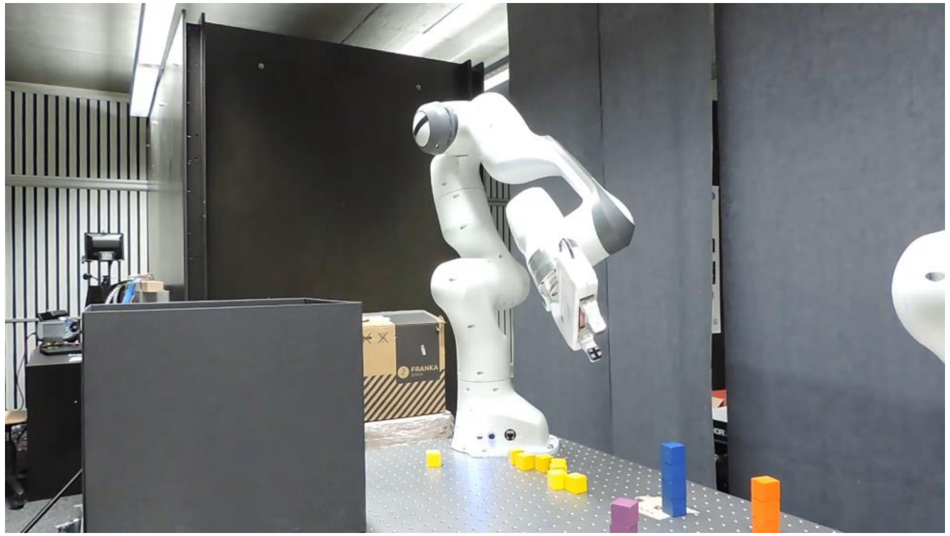
Inverse kinematics (success rate)	Number of samples 1	10	100	1000
TTGO	94.00%	98.00%	98.00%	99.00%
Uniform	37.75%	45.50%	59.25%	75.00%

Target reaching (success rate)	Number of samples 1	10	100	1000
TTGO	62.00%	86.00%	86.00%	88.00%
Uniform	19.25%	28.75%	41.00%	53.50%

Pick-and-place (success rate)	Number of samples 1	10	100	1000
TTGO	70.00%	81.00%	79.00%	89.00%
Uniform	23.75%	30.25%	39.5%	44.25%



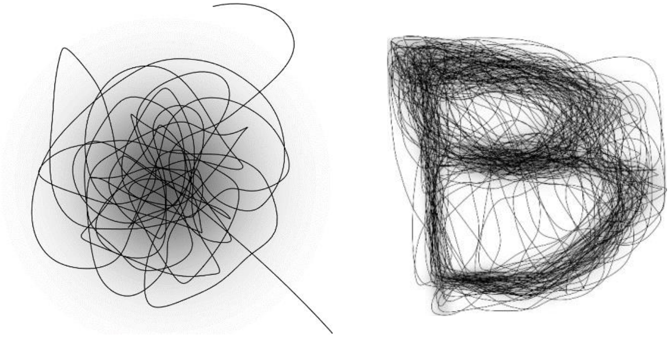


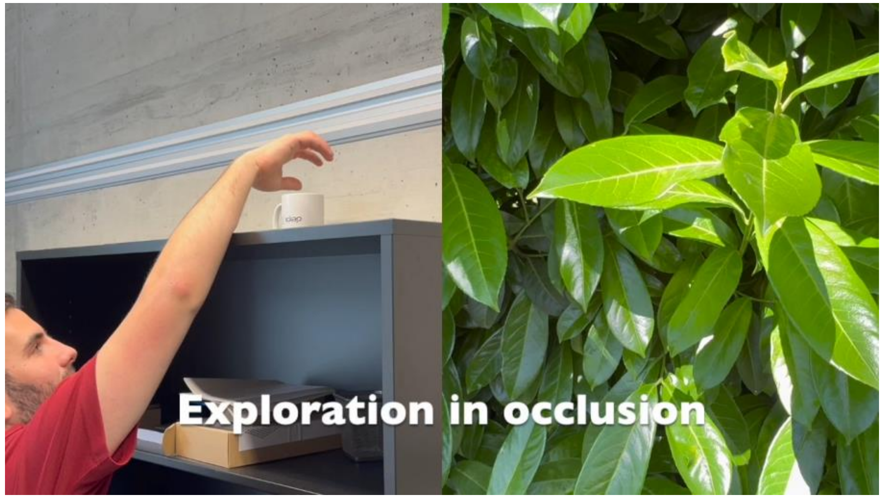


Ergodic control: Spectral multiscale coverage problem

Exploring
=
Tracking in the frequency domain

Input: Spatial distribution
Output: Control commands

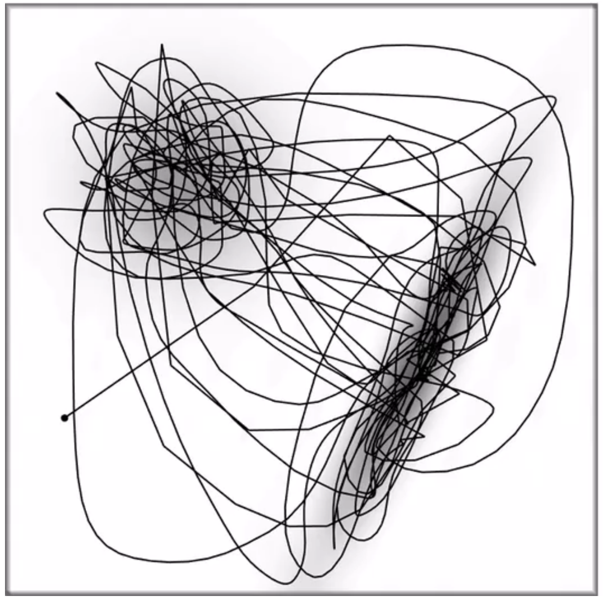


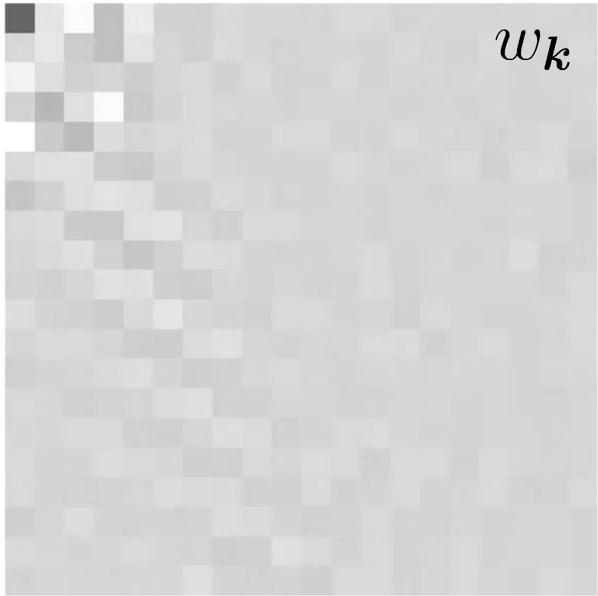


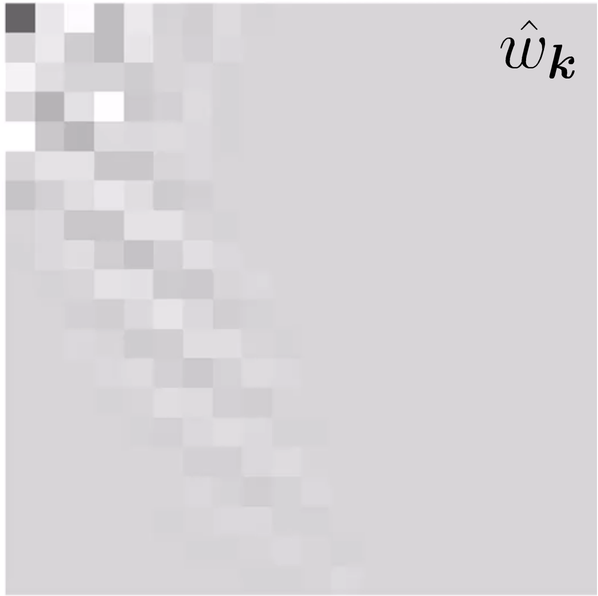
fixed weights

$$\min_{u(t)} \quad \sum_{k\in\mathcal{K}} \Lambda_k (w_k - \hat{w}_k)^2$$

Aim: Matching Fourier series coefficients

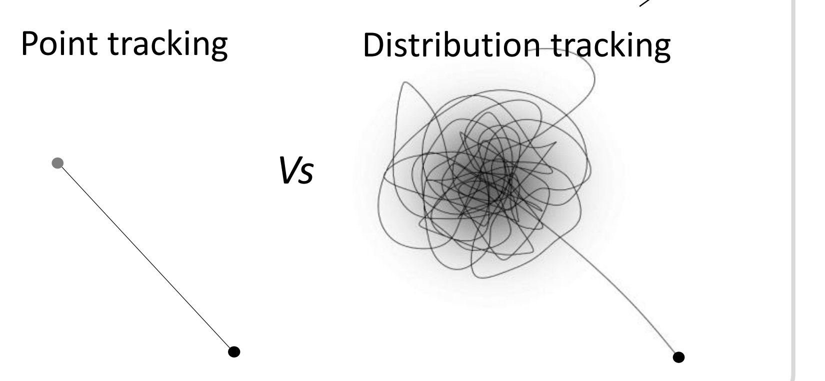






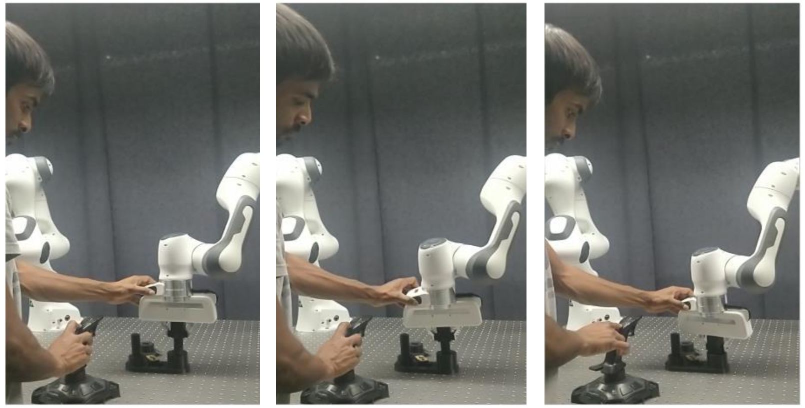
Ergodic control for insertion tasks

Ergodic control as search behavior



The **Fourier basis functions** expansion does not scale well for more than 3 dimensions:
→ **low-rank tensor factorization is required**

Insertion task (Siemens gears benchmark)



Demonstration of insertion pose variations to provide a spatial reference distribution

We evaluate the proposed approach using two different peg grasps:

Grasp #1

Grasp #2

References

Tensor methods

Kolda T, Bader B (2009) Tensor decompositions and applications. SIAM Review 51(3):455-500

Rabanser S, Shchur O, Günnemann S (2017) Introduction to tensor decompositions and their applications in machine learning. arXiv:171110781 pp 1-13

Shetty, S., Lembono, T., Löw, T. and Calinon, S. (2023). Tensor Train for Global Optimization Problems in Robotics. International Journal of Robotics Research (IJRR).

Shetty, S., Silvério, J. and Calinon, S. (2022). Ergodic Exploration using Tensor Train: Applications in Insertion Tasks. IEEE Trans. on Robotics (T-RO), 38:2, 906-921.

Tensor methods - Softwares

https://tensornetwork.org
http://tensorly.org (Python)
https://www.tensorlab.net (Matlab)