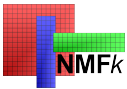
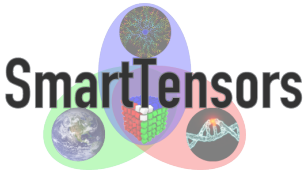
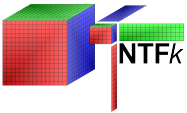
SmartTensors: Unsupervised Machine Learning

Velimir V. Vesselinov (monty) (vvv@lanl.gov, velimir.vesselinov@gmail.com)
Earth and Environmental Sciences Division, Los Alamos National Laboratory, NM, USA
http://tensors.lanl.gov
https://github.com/SmartTensors







Why unsupervised Machine Learning (ML)?

- **Supervised** ML: requires “labeling”, i.e., prior knowledge about the processed data

  Example: Recognizes images of cats and dogs after extensive training; but cannot recognize horses if not trained

  $\Rightarrow$ **Cannot discover something that we do not know already**

  $\Rightarrow$ **Accelerates data processing**

- **Unsupervised** ML: extracts hidden features (signals, signatures) in the processed data without any prior information

  Example: Identifies features that distinguish images of animals (e.g., cats, dogs, horses, etc.)

  $\Rightarrow$ **Allows data mining and discovery**

Why unsupervised Machine Learning (ML)?

- **Supervised** ML: requires “labeling”, i.e., prior knowledge about the processed data

  Example: Recognizes images of cats and dogs after extensive training; but cannot recognize horses if not trained

  ⇒ **Cannot discover something that we do not know already**

  ⇒ **Accelerates data processing**

- **Unsupervised** ML: extracts hidden features (signals, signatures) in the processed data without any prior information

  Example: Identifies features that distinguish images of animals (e.g., cats, dogs, horses, etc.)

  ⇒ **Allows data mining and discovery**

Why unsupervised Machine Learning (ML)?

- **Supervised** ML: requires “labeling”, i.e., prior knowledge about the processed data

  Example: Recognizes images of cats and dogs after extensive training; but cannot recognize horses if not trained

  ⇒ **Cannot discover something that we do not know already**

  ⇒ **Accelerates data processing**

- **Unsupervised** ML: extracts hidden features (signals, signatures) in the processed data without any prior information

  Example: Identifies features that distinguish images of animals (e.g., cats, dogs, horses, etc.)

  ⇒ **Allows data mining and discovery**

Why unsupervised Machine Learning (ML)?

- **Supervised** ML: requires “labeling”, i.e., prior knowledge about the processed data

  Example: Recognizes images of cats and dogs after extensive training; but cannot recognize horses if not trained

  ⇒ **Cannot discover something that we do not know already**
  ⇒ **Accelerates data processing**

- **Unsupervised** ML: extracts hidden features (signals, signatures) in the processed data without any prior information

  Example: Identifies features that distinguish images of animals (e.g., cats, dogs, horses, etc.)

  ⇒ **Allows data mining and discovery**

Why unsupervised Machine Learning (ML)?

- **Supervised** ML: requires “labeling”, i.e., prior knowledge about the processed data

  Example: Recognizes images of cats and dogs after extensive training; but cannot recognize horses if not trained

  ⇒ **Cannot discover something that we do not know already**
  ⇒ **Accelerates data processing**

- **Unsupervised** ML: extracts hidden features (signals, signatures) in the processed data without any prior information

  Example: Identifies features that distinguish images of animals (e.g., cats, dogs, horses, etc.)

  ⇒ **Allows data mining and discovery**

Why unsupervised Machine Learning (ML)?

- **Supervised** ML: requires “labeling”, i.e., prior knowledge about the processed data

  Example: Recognizes images of cats and dogs after extensive training; but cannot recognize horses if not trained

  ⇒ **Cannot discover something that we do not know already**
  ⇒ **Accelerates data processing**

- **Unsupervised** ML: extracts hidden features (signals, signatures) in the processed data without any prior information

  Example: Identifies features that distinguish images of animals (e.g., cats, dogs, horses, etc.)

  ⇒ **Allows data mining and discovery**

Why unsupervised Machine Learning (ML)?

- **Supervised** ML: requires “labeling”, i.e., prior knowledge about the processed data

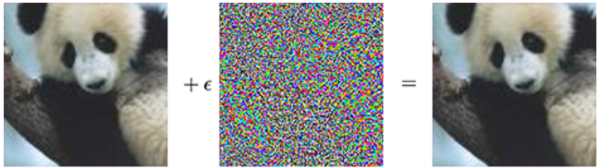
  Example: Recognizes images of cats and dogs after extensive training; but cannot recognize horses if not trained

  ⇒ **Cannot discover something that we do not know already**
  ⇒ **Accelerates data processing**

- **Unsupervised** ML: extracts hidden features (signals, signatures) in the processed data without any prior information

  Example: Identifies features that distinguish images of animals (e.g., cats, dogs, horses, etc.)
  ⇒ **Allows data mining and discovery**

- **Supervised** ML
  - introduces subjectivity (through the labeling process)
  - does not provide insights why horses are different from dogs / cats
  - requires huge training (labeled) datasets
  - we do not know why it works
  - is impacted by “adversarial examples”

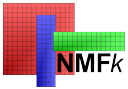


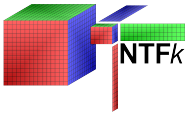
$\Rightarrow$ major limitations of the **supervised** ML methods for **science** applications

Unsupervised Machine Learning Applications: Data Analytics / Model Diagnostics

- Feature extraction (**FE**)
- Blind source separation (**BSS**)
- Detection of disruptions / anomalies
- Image recognition
- Separate physics processes
- Discover unknown dependencies and phenomena
- Develop reduced-order/surrogate models
- Identify dependencies between model inputs and outputs
- Guide development of physics models representing the data
- Make predictions
- Optimize data acquisition
- “Label” datasets for supervised ML analyses

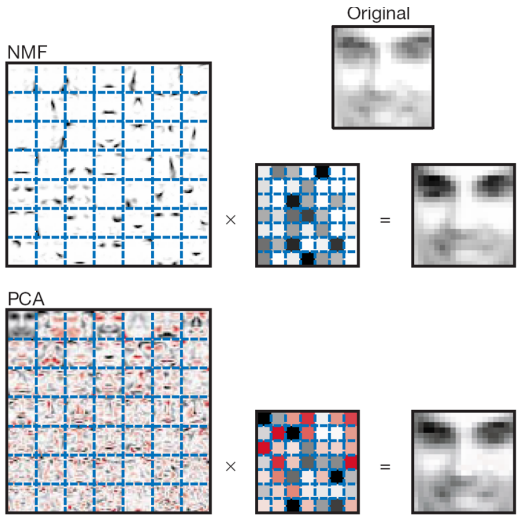
- Novel LANL-patented, open-source, unsupervised Machine Learning (ML) methods and computational techniques
- Based on matrix/tensor factorization coupled with custom k-means clustering and nonnegativity/sparsity/physics-informed constraints
- Developed in julia
  - `http://tensors.lanl.gov`
  - `https://github.com/SmartTensors`
- Capable to efficiently process large datasets (GB/TB's)





Why nonnegativity?

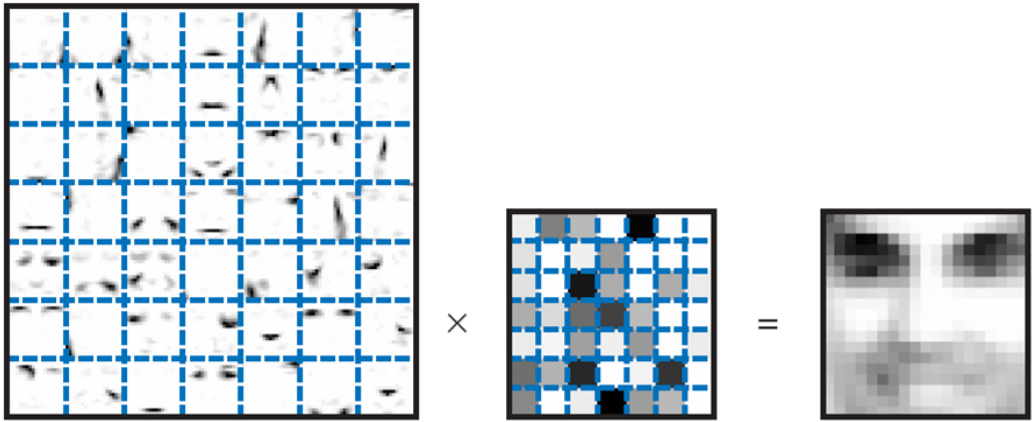
- NMF vs PCA (Lee & Seung, 1999)
- NMF: Nonnegative Matrix Factorization
- PCA: Principal Component Analysis



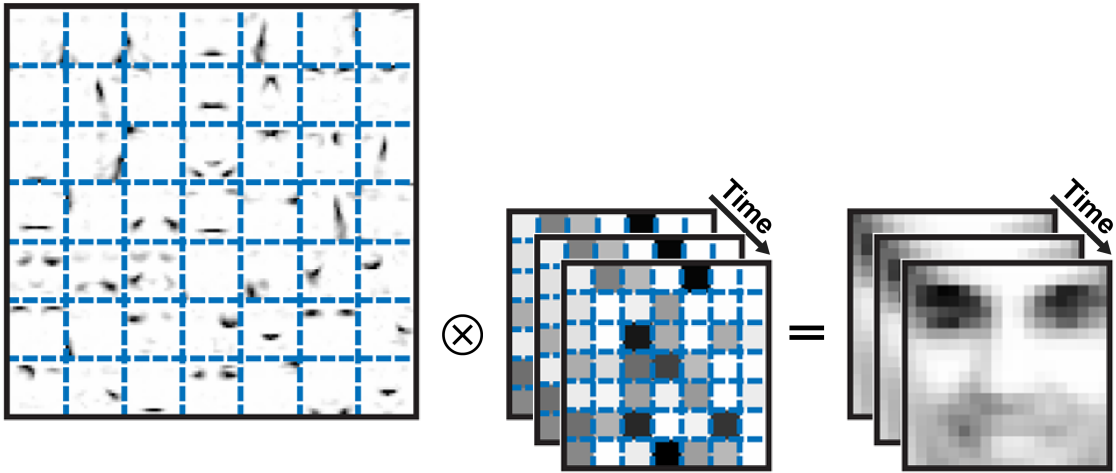
Nonnegativity constraints provide meaningful and interpretable results (+sparsity)

- **Tensors** (multi-dimensional/multi-modal/multi-way datasets) are everywhere:
  - observational data are typically a 5-D tensor (x, y, z, t, attributes)
  - model outputs are typically a 5-D tensor (x, y, z, t, attributes)
  - data dependency to N parameters will form a $(N + 5)$-D tensor

NMF: Nonnegative Matrix Factorization

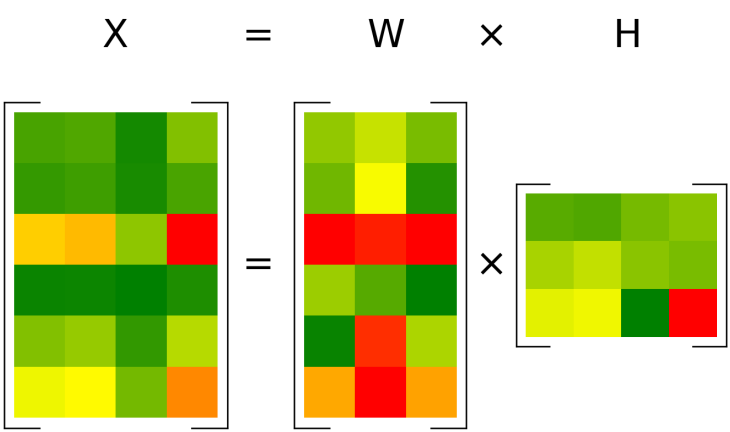


NTF: Nonnegative Tensor Factorization



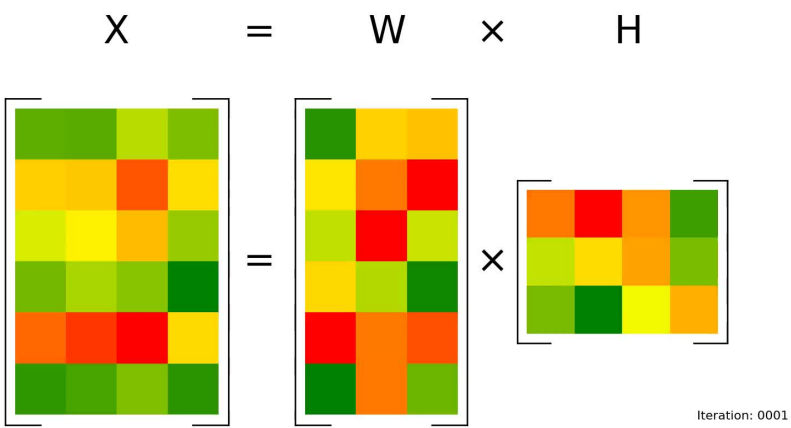
Why not SVD (PCA)?

- **SVD** cannot be applied on multi-dimensional (tensor) datasets
- **HOSVD** cannot tell us the number of features (signals)

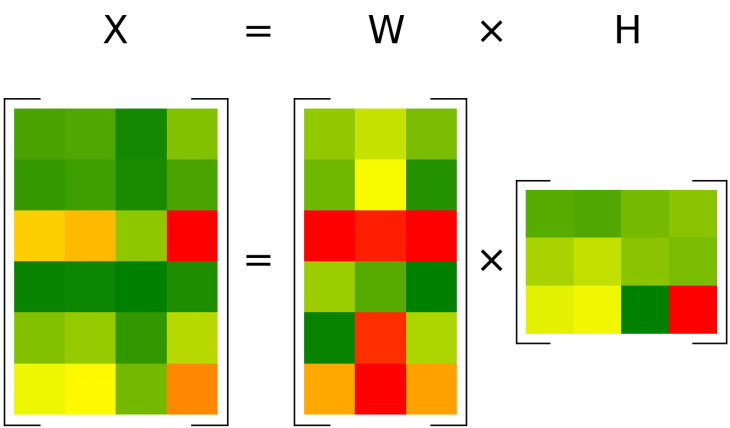


24 knowns $(6 \times 4) \rightarrow 30$ unknowns $(6 \times 3) + (3 \times 4)$

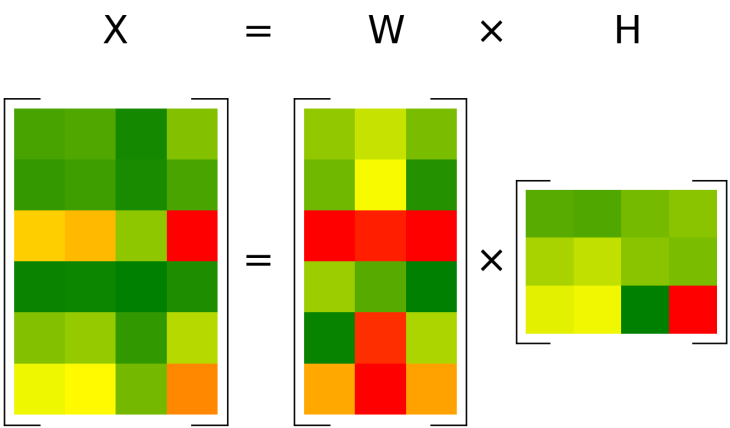
NMFk example: Factorization process



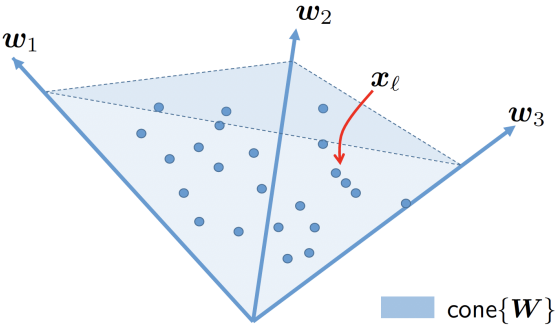
NMFk example: Estimate

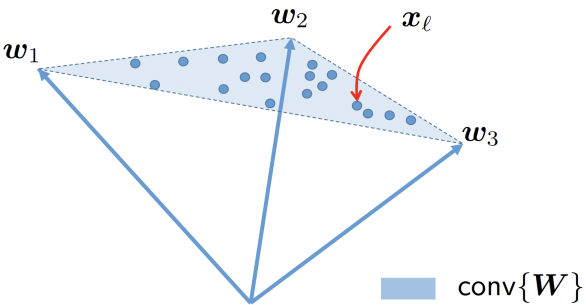


NMFk example: Truth



NMFk: Conic / Convex hull



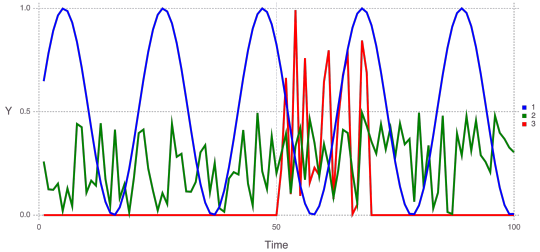


$$X = W \times H$$

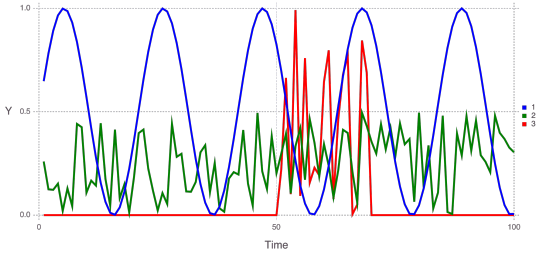
$X_{columns} \in cone\{W\}$

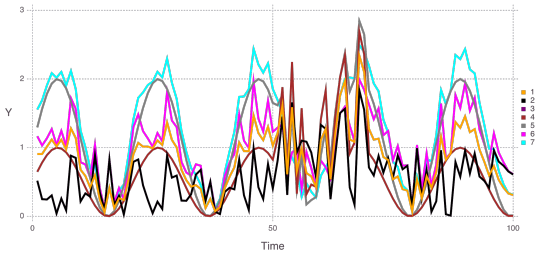
$X_{columns} \in conv\{W\}$
if H rows sum up to one
(“row stochastic”)

NMFk: Extraction of features

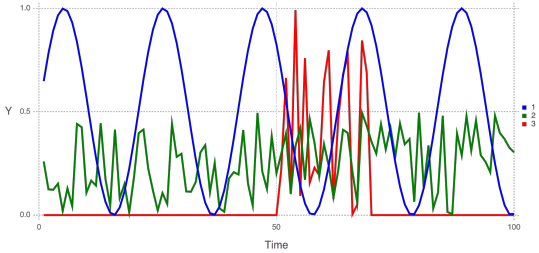


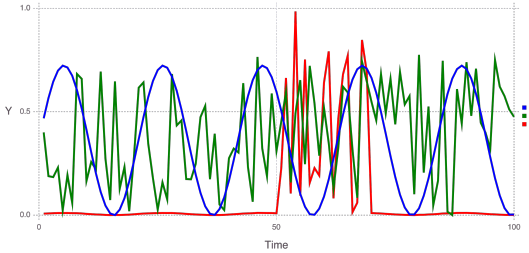
NMFk: Extraction of features

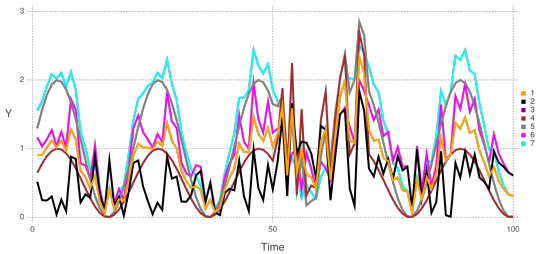




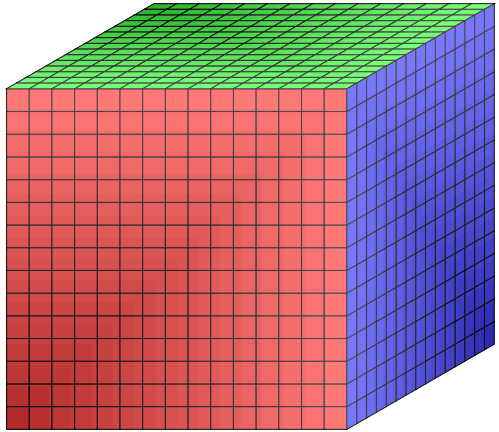
NMFk: Extraction of features





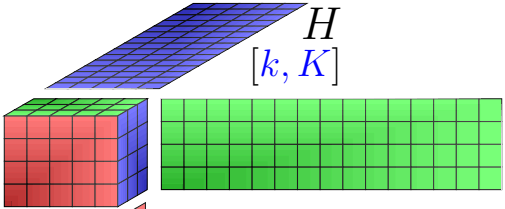


Tucker Tensor Decomposition (3D): Rank-60 Multirank-(3,4,5)



X
$[K, M, N]$

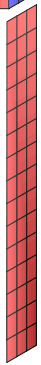
$\approx$



G
$[k, m, n]$

H
$[k, K]$

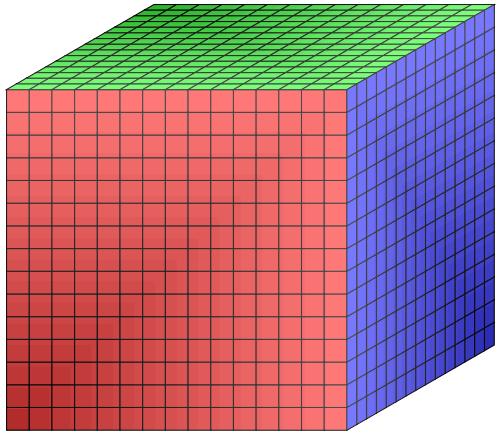
W
$[n, N]$



V
$[m, M]$

$$X \approx G \otimes H \otimes W \otimes V$$

Canonical Polyadic Tensor Decomposition (3D): Rank-1



X
$[K, M, N]$

$\approx$

H
$[1, K]$

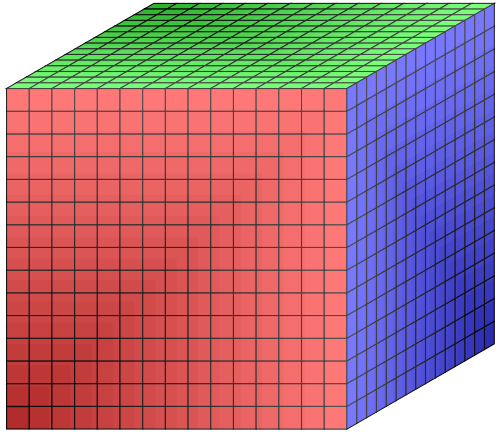
λ

W
$[1, N]$

V
$[1, M]$

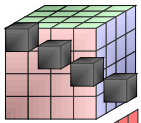
$$X \approx \lambda(H \otimes W \otimes V)$$

Canonical Polyadic Tensor Decomposition (3D): Rank-4

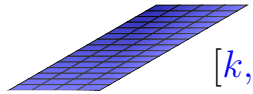


X
$[K, M, N]$

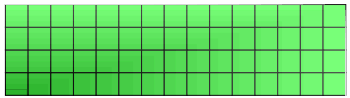
$\approx$



D
$[k, k, k]$



H
$[k, K]$

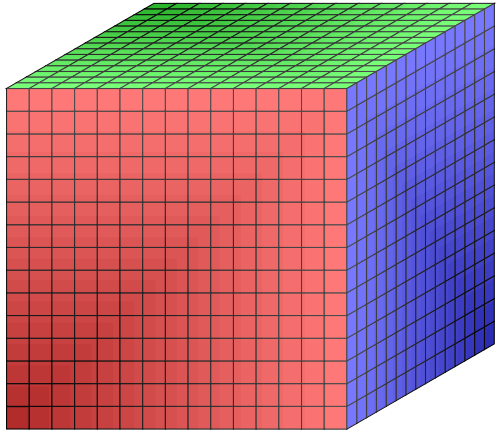


W
$[k, N]$

V
$[k, M]$

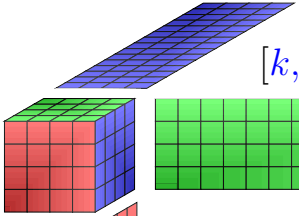
$$X \approx D \otimes H \otimes W \otimes V$$

Tucker Tensor Decomposition (3D): Rank-64 Multirank-(4,4,4)



X
$[K, M, N]$

$\approx$



G
$[k, k, k]$

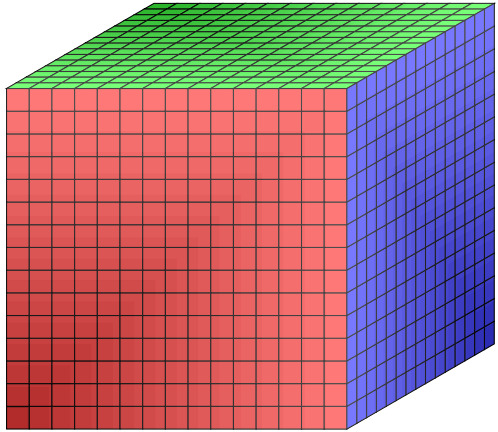
H
$[k, K]$

W
$[k, N]$

V
$[k, M]$

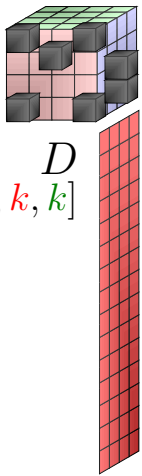
$$X \approx G \otimes H \otimes W \otimes V$$

Tucker Tensor Decomposition (3D): Rank-7 Multirank-(3,3,4)

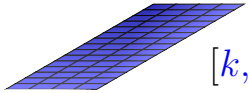


$$X$$
$$[K, M, N]$$

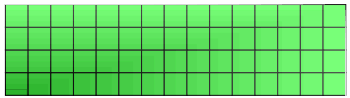
$\approx$



$$D$$
$$[k, k, k]$$



$$H$$
$$[k, K]$$



$$W$$
$$[k, N]$$

$$V$$
$$[k, M]$$

$$X \approx D \otimes H \otimes W \otimes V$$

Tensor Rank and Tensor Decomposition

- **Rank-1 Tensor**: a tensor product of a set of vectors in each dimension
- **Tensor Rank** r: smallest number r of Rank-1 tensors under Canonical Polyadic Decomposition
- **Tensor Multi-Rank** [k, m, n, ...]: smallest dimensions [k, m, n, ...] of core tensor G under Tucker Decomposition (it always exists)
- **Tensor Rank** is always equal to the rank of Tucker tensor core: $rank(X) = rank(G)$
- **Nonnegative Tensor Rank**: smallest number r of rank-1 tensors under Nonnegative Canonical Polyadic Decomposition
- **Nonnegative Tensor Multi-Rank**: smallest dimension of core tensor G under Nonnegative Tucker Decomposition (existence cannot be guaranteed)
- **Nonnegative Tensor Rank** is less than or equal to the rank of Nonnegative Tucker tensor core: $rank_+(X) \leq rank_+(G)$
- **Nonnegative Tensor Rank**: smallest number r of nonzero entries in tensor core under Nonnegative Tucker Decomposition

Tensor Rank and Tensor Decomposition

- **Rank-1 Tensor**: a tensor product of a set of vectors in each dimension
- **Tensor Rank** r: smallest number r of Rank-1 tensors under Canonical Polyadic Decomposition
- **Tensor Multi-Rank** $[k, m, n, ...]$: smallest dimensions $[k, m, n, ...]$ of core tensor G under Tucker Decomposition (it always exists)
- **Tensor Rank** is always equal to the rank of Tucker tensor core: $rank(X) = rank(G)$
- **Nonnegative Tensor Rank**: smallest number r of rank-1 tensors under Nonnegative Canonical Polyadic Decomposition
- **Nonnegative Tensor Multi-Rank**: smallest dimension of core tensor G under Nonnegative Tucker Decomposition (existence cannot be guaranteed)
- **Nonnegative Tensor Rank** is less than or equal to the rank of Nonnegative Tucker tensor core: $rank_+(X) \leq rank_+(G)$
- **Nonnegative Tensor Rank**: smallest number r of nonzero entries in tensor core under Nonnegative Tucker Decomposition

- **Rank-1 Tensor**: a tensor product of a set of vectors in each dimension
- **Tensor Rank** r: smallest number r of Rank-1 tensors under Canonical Polyadic Decomposition
- **Tensor Multi-Rank [**k**,** m**,** n**, ...]**: smallest dimensions [k, m, n, ...] of core tensor G under Tucker Decomposition (it always exists)
- **Tensor Rank** is always equal to the rank of Tucker tensor core: $rank(X) = rank(G)$
- **Nonnegative Tensor Rank**: smallest number r of rank-1 tensors under Nonnegative Canonical Polyadic Decomposition
- **Nonnegative Tensor Multi-Rank**: smallest dimension of core tensor G under Nonnegative Tucker Decomposition (existence cannot be guaranteed)
- **Nonnegative Tensor Rank** is less than or equal to the rank of Nonnegative Tucker tensor core: $rank_+(X) \leq rank_+(G)$
- **Nonnegative Tensor Rank**: smallest number r of nonzero entries in tensor core under Nonnegative Tucker Decomposition

Tensor Rank and Tensor Decomposition

- **Rank-1 Tensor**: a tensor product of a set of vectors in each dimension
- **Tensor Rank** r: smallest number r of Rank-1 tensors under Canonical Polyadic Decomposition
- **Tensor Multi-Rank [k, m, n, ...]**: smallest dimensions [k, m, n, ...] of core tensor G under Tucker Decomposition (it always exists)
- **Tensor Rank** is always equal to the rank of Tucker tensor core: $rank(X) = rank(G)$
- **Nonnegative Tensor Rank**: smallest number r of rank-1 tensors under Nonnegative Canonical Polyadic Decomposition
- **Nonnegative Tensor Multi-Rank**: smallest dimension of core tensor G under Nonnegative Tucker Decomposition (existence cannot be guaranteed)
- **Nonnegative Tensor Rank** is less than or equal to the rank of Nonnegative Tucker tensor core: $rank_+(X) \leq rank_+(G)$
- **Nonnegative Tensor Rank**: smallest number r of nonzero entries in tensor core under Nonnegative Tucker Decomposition

Tensor Rank and Tensor Decomposition

- **Rank-1 Tensor**: a tensor product of a set of vectors in each dimension
- **Tensor Rank** r: smallest number r of Rank-1 tensors under Canonical Polyadic Decomposition
- **Tensor Multi-Rank [k, m, n, ...]**: smallest dimensions [k, m, n, ...] of core tensor G under Tucker Decomposition (it always exists)
- **Tensor Rank** is always equal to the rank of Tucker tensor core: $rank(X) = rank(G)$
- **Nonnegative Tensor Rank**: smallest number r of rank-1 tensors under Nonnegative Canonical Polyadic Decomposition
- **Nonnegative Tensor Multi-Rank**: smallest dimension of core tensor G under Nonnegative Tucker Decomposition (existence cannot be guaranteed)
- **Nonnegative Tensor Rank** is less than or equal to the rank of Nonnegative Tucker tensor core: $rank_+(X) \leq rank_+(G)$
- **Nonnegative Tensor Rank**: smallest number r of nonzero entries in tensor core under Nonnegative Tucker Decomposition

Tensor Rank and Tensor Decomposition

- **Rank-1 Tensor**: a tensor product of a set of vectors in each dimension
- **Tensor Rank** r: smallest number r of Rank-1 tensors under Canonical Polyadic Decomposition
- **Tensor Multi-Rank [k, m, n, ...]**: smallest dimensions [k, m, n, ...] of core tensor G under Tucker Decomposition (it always exists)
- **Tensor Rank** is always equal to the rank of Tucker tensor core: $rank(X) = rank(G)$
- **Nonnegative Tensor Rank**: smallest number r of rank-1 tensors under Nonnegative Canonical Polyadic Decomposition
- **Nonnegative Tensor Multi-Rank**: smallest dimension of core tensor G under Nonnegative Tucker Decomposition (existence cannot be guaranteed)
- **Nonnegative Tensor Rank** is less than or equal to the rank of Nonnegative Tucker tensor core: $rank_+(X) \leq rank_+(G)$
- **Nonnegative Tensor Rank**: smallest number r of nonzero entries in tensor core under Nonnegative Tucker Decomposition

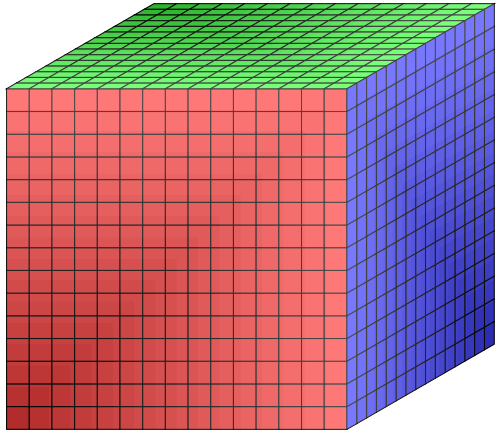
Tensor Rank and Tensor Decomposition

- **Rank-1 Tensor**: a tensor product of a set of vectors in each dimension
- **Tensor Rank** r: smallest number r of Rank-1 tensors under Canonical Polyadic Decomposition
- **Tensor Multi-Rank [k, m, n, ...]**: smallest dimensions [k, m, n, ...] of core tensor G under Tucker Decomposition (it always exists)
- **Tensor Rank** is always equal to the rank of Tucker tensor core: $rank(X) = rank(G)$
- **Nonnegative Tensor Rank**: smallest number r of rank-1 tensors under Nonnegative Canonical Polyadic Decomposition
- **Nonnegative Tensor Multi-Rank**: smallest dimension of core tensor G under Nonnegative Tucker Decomposition (existence cannot be guaranteed)
- **Nonnegative Tensor Rank** is less than or equal to the rank of Nonnegative Tucker tensor core: $rank_+(X) \leq rank_+(G)$
- **Nonnegative Tensor Rank**: smallest number r of nonzero entries in tensor core under Nonnegative Tucker Decomposition

Tensor Rank and Tensor Decomposition

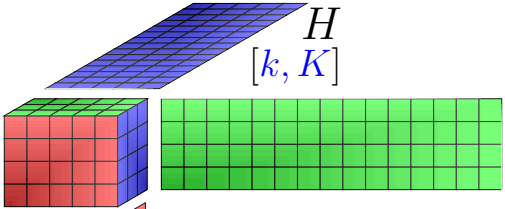
- **Rank-1 Tensor**: a tensor product of a set of vectors in each dimension
- **Tensor Rank** r: smallest number r of Rank-1 tensors under Canonical Polyadic Decomposition
- **Tensor Multi-Rank [k, m, n, ...]**: smallest dimensions [k, m, n, ...] of core tensor G under Tucker Decomposition (it always exists)
- **Tensor Rank** is always equal to the rank of Tucker tensor core: $rank(X) = rank(G)$
- **Nonnegative Tensor Rank**: smallest number r of rank-1 tensors under Nonnegative Canonical Polyadic Decomposition
- **Nonnegative Tensor Multi-Rank**: smallest dimension of core tensor G under Nonnegative Tucker Decomposition (existence cannot be guaranteed)
- **Nonnegative Tensor Rank** is less than or equal to the rank of Nonnegative Tucker tensor core: $rank_+(X) \leq rank_+(G)$
- **Nonnegative Tensor Rank**: smallest number r of nonzero entries in tensor core under Nonnegative Tucker Decomposition

Tucker Tensor Decomposition (3D): Rank-60 Multirank-(3,4,5)



X
$[K, M, N]$

$\approx$



G
$[k, m, n]$

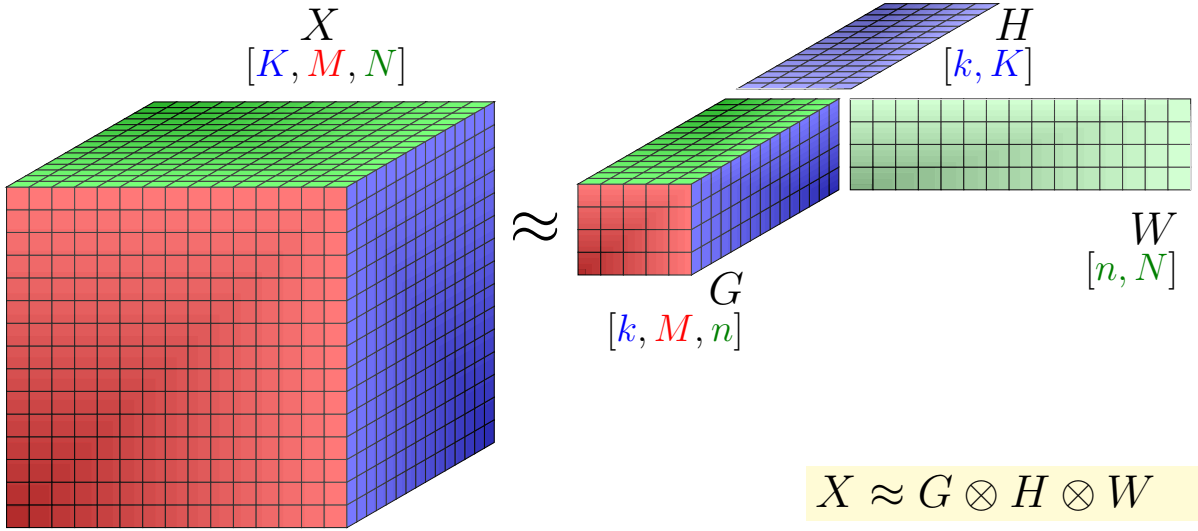
H
$[k, K]$

W
$[n, N]$

V
$[m, M]$

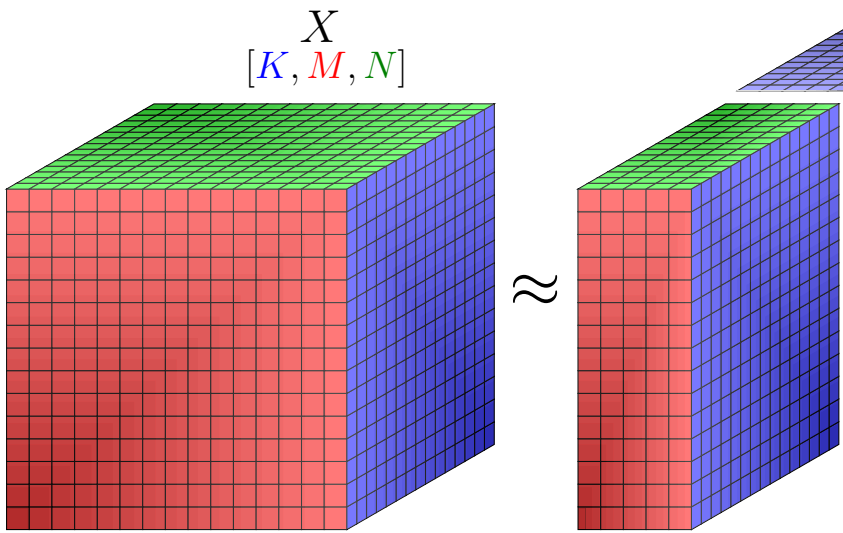
$$X \approx G \otimes H \otimes W \otimes V$$

Tucker Tensor Decomposition (3D): Tucker-2 (three possible alternatives)



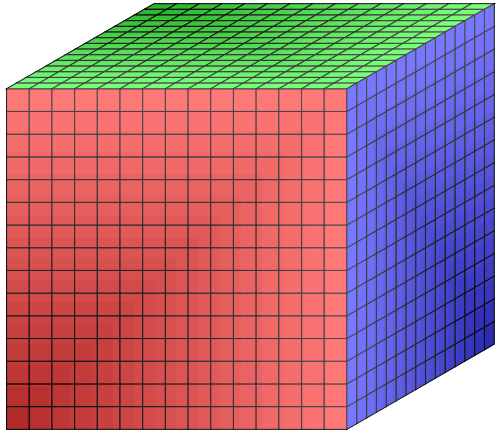
$$X \approx G \otimes H \otimes W$$

Tucker Tensor Decomposition (3D): Tucker-1 (three possible alternatives)



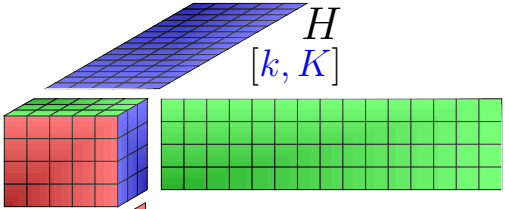
$$X \approx G \otimes H$$

Tucker Tensor Decomposition (3D): Rank-60 Multirank-(3,4,5)



X
$[K, M, N]$

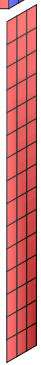
$\approx$



G
$[k, m, n]$

H
$[k, K]$

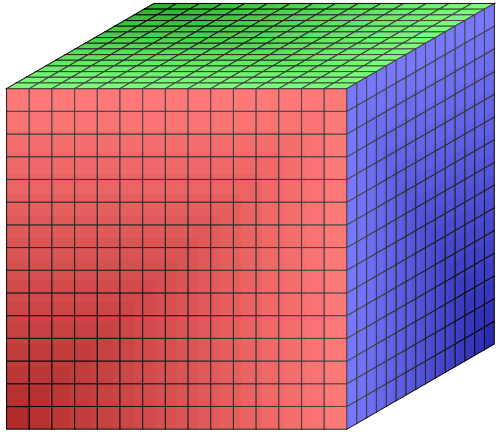
W
$[n, N]$



V
$[m, M]$

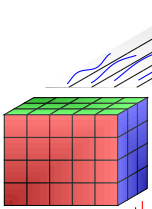
$$X \approx G \otimes H \otimes W \otimes V$$

Tucker Tensor Decomposition (3D): Feature extraction



X
$[K, M, N]$

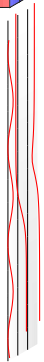
$\approx$



G
$[k, m, n]$

H
$[k, K]$

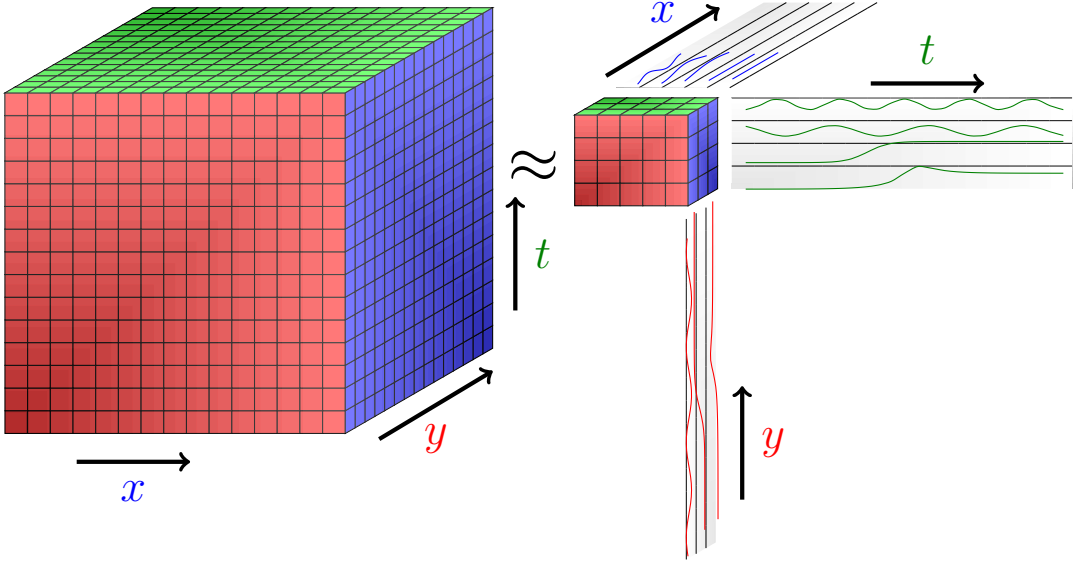
W
$[n, N]$



V
$[m, M]$

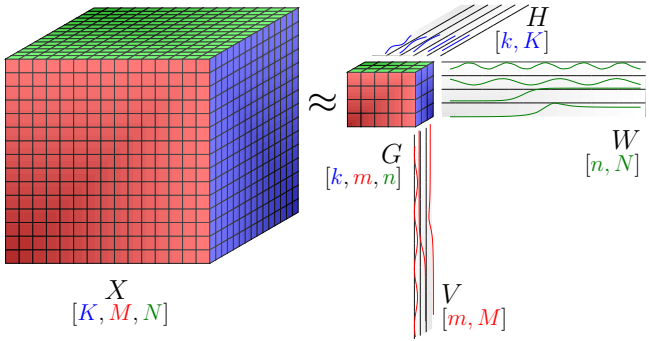
$$X \approx G \otimes H \otimes W \otimes V$$

Tucker Tensor Decomposition (3D): Feature extraction

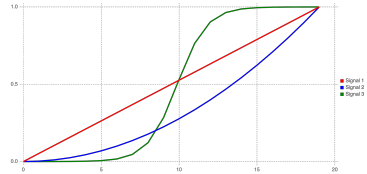


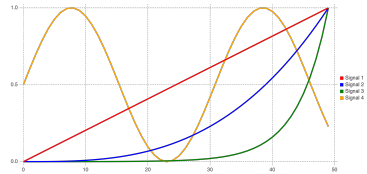
Tensor Decomposition

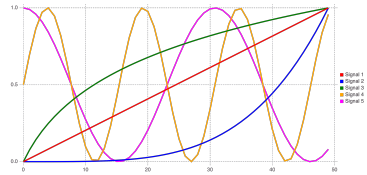
- Tucker/CPD decomposition is achieved through minimization
- Nonnegativity, sparsity, physics-informed constraints can be applied
- Optimal number of features $[k, m, n]$ (Tensor Multi-Rank) is estimated through k-means clustering of a series of minimization solutions with random initial guesses



Tucker Tensor Decomposition: Example



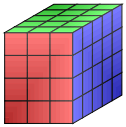




$$V = \begin{bmatrix} t \\ t^2 \\ tanh(t-10)+1 \end{bmatrix}$$

$$W = \begin{bmatrix} x \\ x^3 \\ e^x \\ sin(x)+1 \end{bmatrix}$$

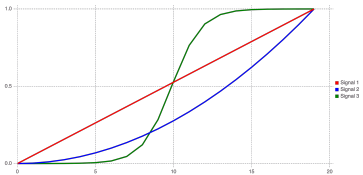
$$H = \begin{bmatrix} y \\ y^4 \\ ln(y) \\ sin(2y)+1 \\ cos(y)+1 \end{bmatrix}$$

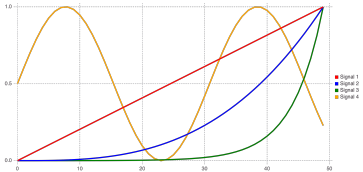


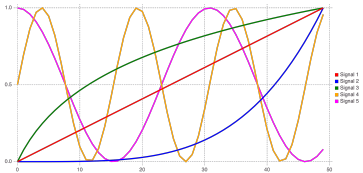
(12 elements)

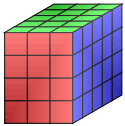
$$\begin{aligned} X &= G \otimes H \otimes W \otimes V \\ &= xyt + xy^4t + xtln(y) + \\ &xt(sin(2y)+1) + x^3yt + \\ &xt(cos(y)+1) + yte^x + \\ &yt(sin(x)+1) + \\ &xyt^2 + xy(1+tanh(t-10)) + \\ &t^2e^x(sin(2y)+1) + \\ &(1+tanh(t-10))(sin(x)+1) \\ &(cos(y)+1) \end{aligned}$$

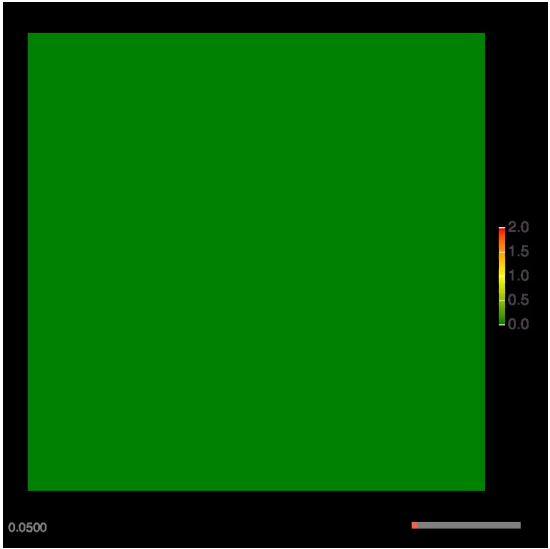
Tucker Tensor Decomposition: Example



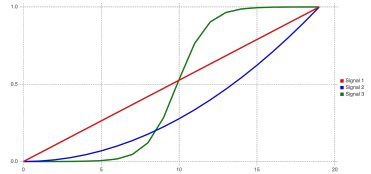


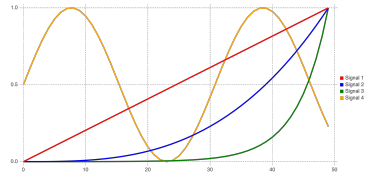


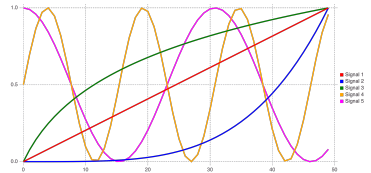




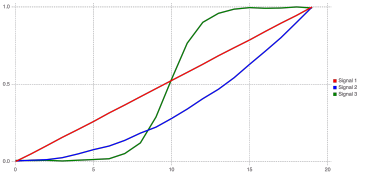
Tucker Tensor Decomposition: Example

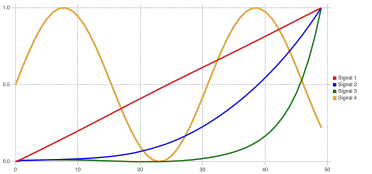


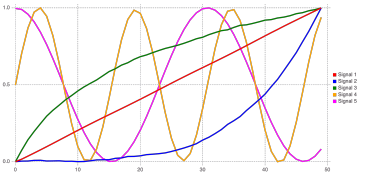




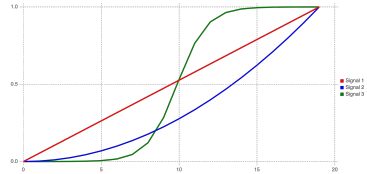
← **Truth** vs **Predictions** →



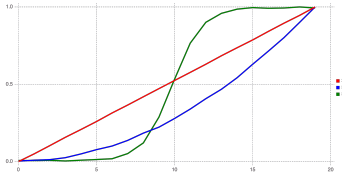




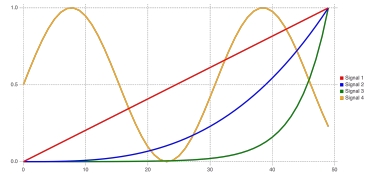
Tucker Tensor Decomposition: Example

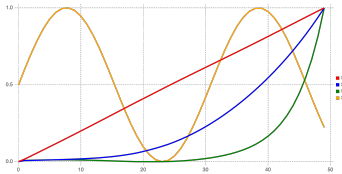


← Truth vs Predictions →

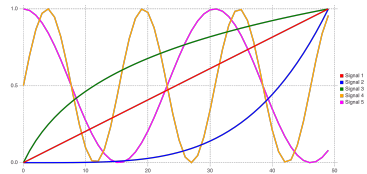


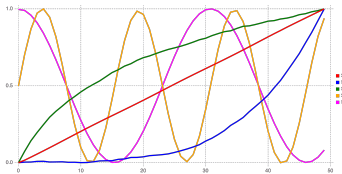
$$V = \begin{bmatrix} t \\ t^2 \\ tanh(t-10)+1 \end{bmatrix}$$





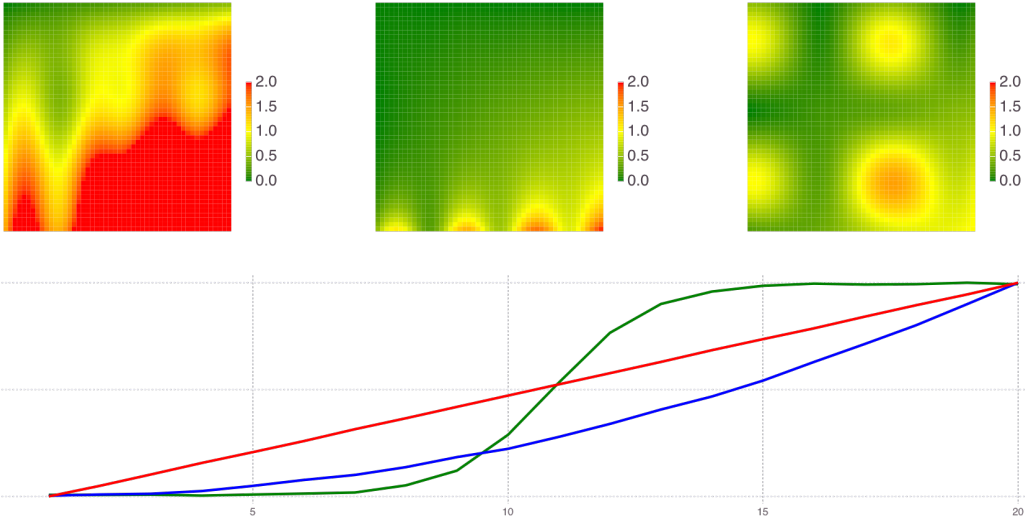
$$W = \begin{bmatrix} x \\ x^3 \\ e^x \\ sin(x)+1 \end{bmatrix}$$



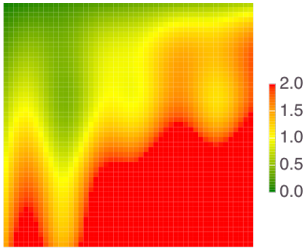


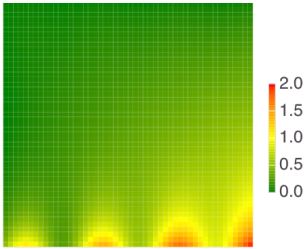
$$H = \begin{bmatrix} y \\ y^4 \\ ln(y) \\ sin(2y)+1 \\ cos(y)+1 \end{bmatrix}$$

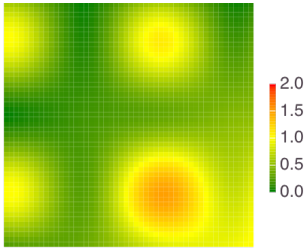
Tucker Tensor Decomposition: Example



Tucker Tensor Decomposition: Example







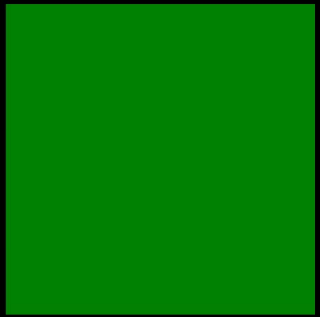
$$M = G \otimes H \otimes W$$

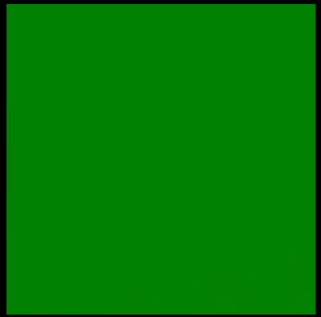
$$M_1 = xy + xy^4 + xlog(y+1) + x(sin(2y)+1) + ye^x + x(cos(y)+1) + x^3y + y(sin(x)+1)$$

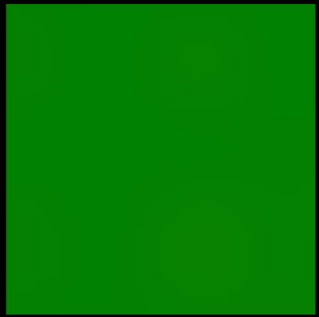
$$M_2 = xy + e^x(cos(y)+1)$$

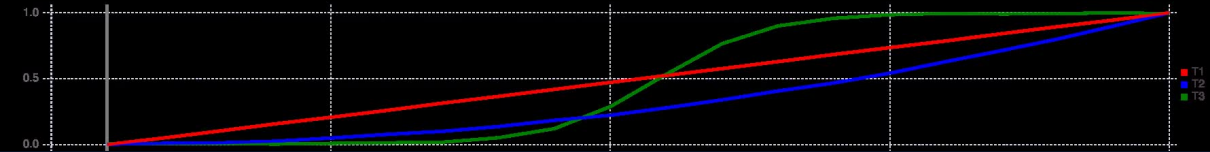
$$M_3 = xy + (sin(x)+1)(cos(y)+1)$$

Tucker Tensor Decomposition: Example



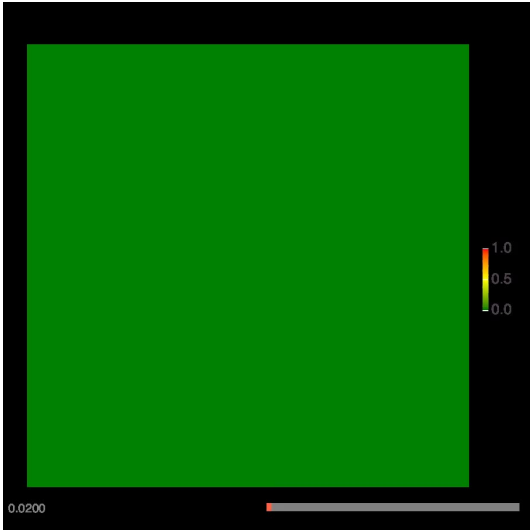




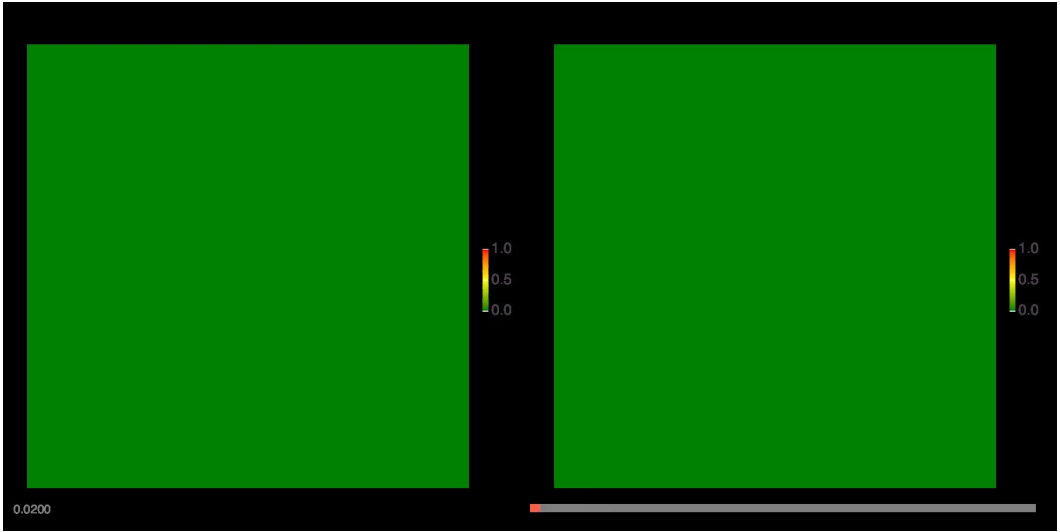


Tucker Tensor Decomposition: Example II

- $(50 \times 50 \times 50)$ tensor
- 50 columns in x
- 50 rows in y
- 50 time frames
- ‘ones‘ swimming in a sea of ‘zeros‘

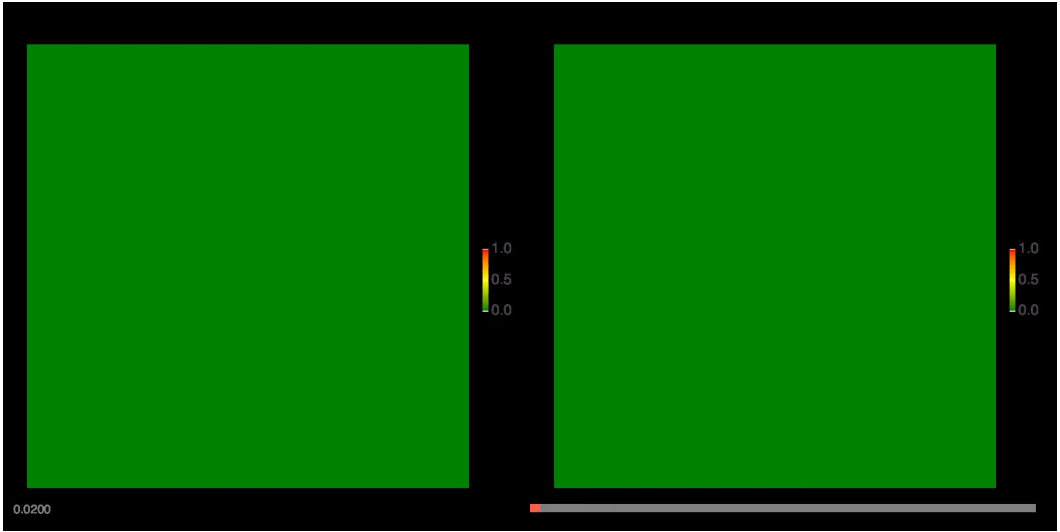


Tucker Tensor Decomposition: Example II



Factorizing all **3** dimensions $(50 \times 50 \times 50) \rightarrow (6 \times 44 \times 48)$

Tucker Tensor Decomposition: Example II



6 groups of swimmers (x); 44 lanes occupied (y); 48 time frames (first/last empty)

SmartTensors Challenges

- **Identifying the number of unknown features**:
  - resolved using custom k-means clustering and sparsity constraints on the core tensor
  - number of features identified based on the reconstruction quality (e.g., Frobenius norm) and cluster Silhouettes
- **Solving a non-unique optimization problem**:
  - addressed through multistarts, regularization and nonnegativity constraints
  - applying diverse optimization techniques (Multiplicative/Alternating Least Squares algorithms, NLopt, Ipopt, Gurobi, MOSEK, GLPK, Clp, Cbc, ...)
- **Processing Big Data**:
  - GPU's / TPU's / Distributed computing
  - Account for data sparsity and structure
  - Nonnegative Tensor Trains
- **Dealing with Noisy Data**:
  - Random noise impacts accuracy but it is accountable
  - Systematic noise is identified as separate signals

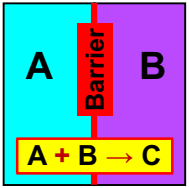
4GB Tensor ($1000 \times 1000 \times 1000$)

Framework	Execution time (seconds)
MATLAB	2634
NumPy	881
MXNet	644
PyTorch	121
TensorFlow	119
Julia	109

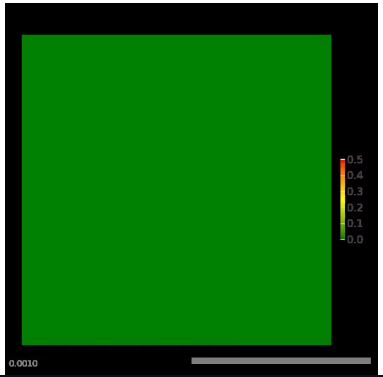
- **Field Data**:
  - Groundwater contamination
  - US Climate data
  - Geothermal data
  - Seismic data
- **Lab Data**:
  - X-ray Spectroscopy
  - UV Fluorescence Spectroscopy
  - Microbial population analyses
- **Operational Data**:
  - LANSCE: Los Alamos Neutron Accelerator
  - Oil/gas production
- **Model Outputs**:
  - Reactive mixing $A + B \rightarrow C$
  - Phase separation of co-polymers
  - Molecular Dynamics of proteins
  - EU Climate modeling

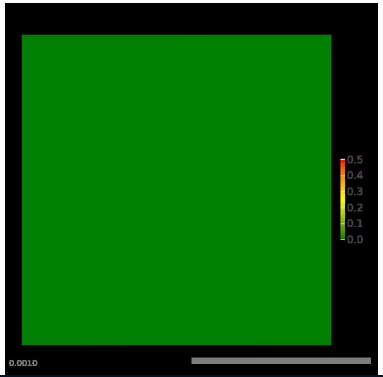
SmartTensors Publications

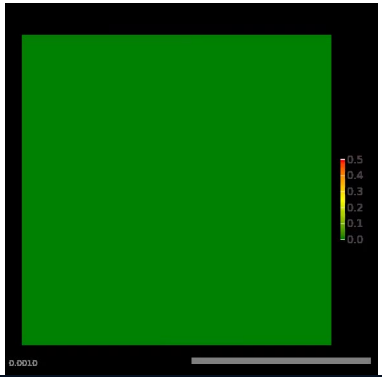
- Siler et al., Machine learning to identify geologic factors associated with production in geothermal fields: A case-study using 3D geologic data, Brady geothermal field, Nevada, **Geothermal Energy**, 10.1186/s40517-021-00199-8, 2021.
- Ahmmed et al., Machine Learning to Discover Mineral Trapping Signatures due to CO2 Injection, **Journal of Greenhouse Gas Control**, 10.1016/j.ijggc.2021.103382, 2021.
- Ahmmed et al., A comparative study of machine learning models for predicting the state of reactive mixing, **Journal of Computational Physics** 10.1016/j.jcp.2021.110147, 2021.
- Mehana et al., Machine-Learning Predictions of the Shale Wells? Performance, **Journal of Natural Gas Science and Engineering**, 10.1016/j.jngse.2021.103819, 2021.
- Vesselinov et al., Unsupervised Machine Learning Based on Non-Negative Tensor Factorization for Analyzing Reactive-Mixing, **Journal of Computational Physics**, Special issue: Machine Learning, 2019.
- Stanev et al., Unsupervised Phase Mapping of X-ray Diffraction Data by Nonnegative Matrix Factorization Integrated with Custom Clustering, **Nature Computational Materials**, 2018.
- Vesselinov et al., Nonnegative Tensor Factorization for Contaminant Source Identification, **Journal of Contaminant Hydrology**, 2018.
- O'Malley et al., Nonnegative/binary matrix factorization with a D-Wave quantum annealer, **PLOS ONE**, 2018.
- Vesselinov et al., Contaminant source identification using semi-supervised machine learning, **Journal of Contaminant Hydrology**, 2017.
- Alexandrov, Vesselinov, Blind source separation for groundwater level analysis based on nonnegative matrix factorization, **WRR**, 2014.



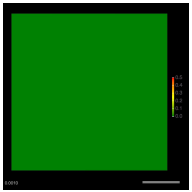
- > 2000 simulations of C concentrations in time/space with varying model inputs representing reactive mixing (5 input model parameters)
- **NTF**k identifies physics processes impacting C concentrations and their relationship to model inputs



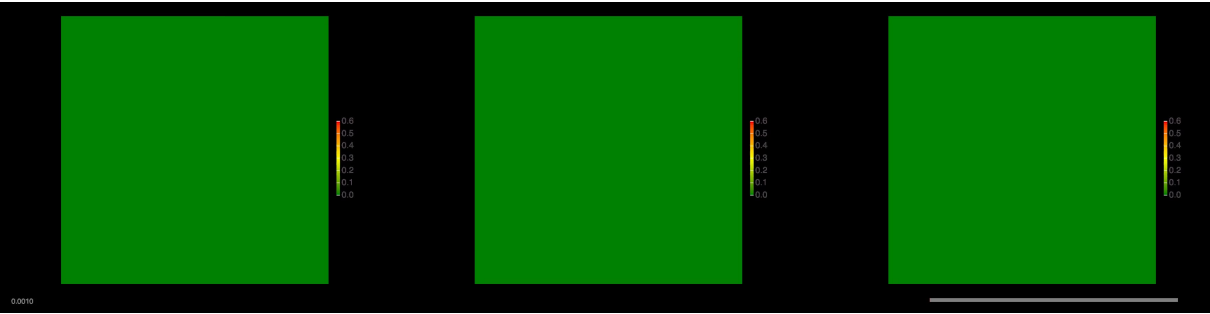




SmartTensors Example: Reactive mixing results



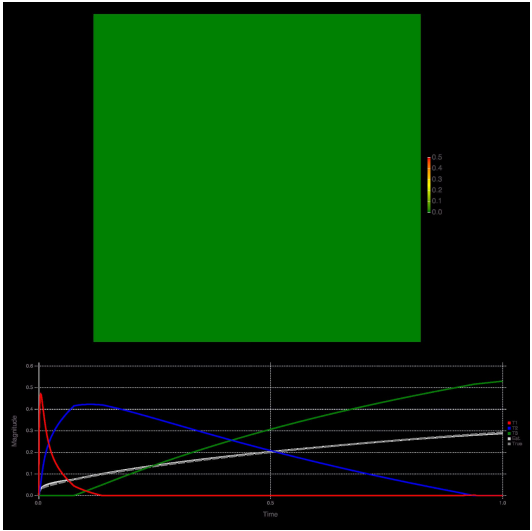
- **NTFk** extracts the dominant time/space features (**processes** / **vortices**) and compresses the model outputs
- Compression: > 200GB $\rightarrow \sim 70$MB (ratio ~ 3000
  Here, $(1000 \times 81 \times 81) \rightarrow (3 \times 12 \times 13)$ $(time \times rows \times columns)$



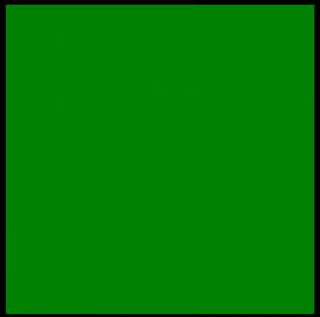
Advection **Dispersion** **Diffusion**

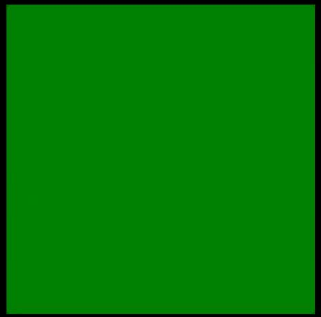
SmartTensors Example: Reactive mixing results

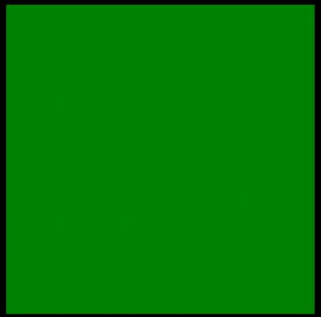
- T1: Advection
- T2: Dispersion
- T3: Diffusion

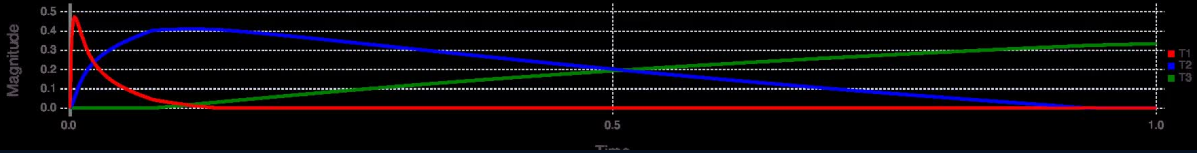


SmartTensors Example: Reactive mixing results

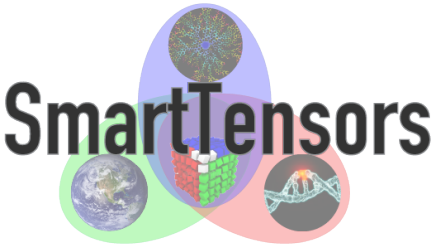






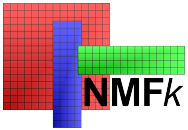


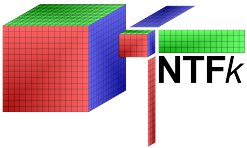
- Developed **novel** unsupervised ML methods and open-source computational framework called **SmartTensors** based on Nonnegative Factorization (Matrices/Tensors)
- **SmartTensors** has been used to solve various real-world problems
- **SmartTensors** commercialization and deployment as a service on JuliaHub is coming soon (DOE TCF funding obtained)



- Codes:

NMFk MADS NTFk





- Examples:

http://madsjulia.github.io/Mads.jl/Examples/blind_source_separation
http://tensors.lanl.gov
http://SmartTensors.github.io
https://github.com/SmartTensors
https://hub.docker.com/u/montyvesselinov