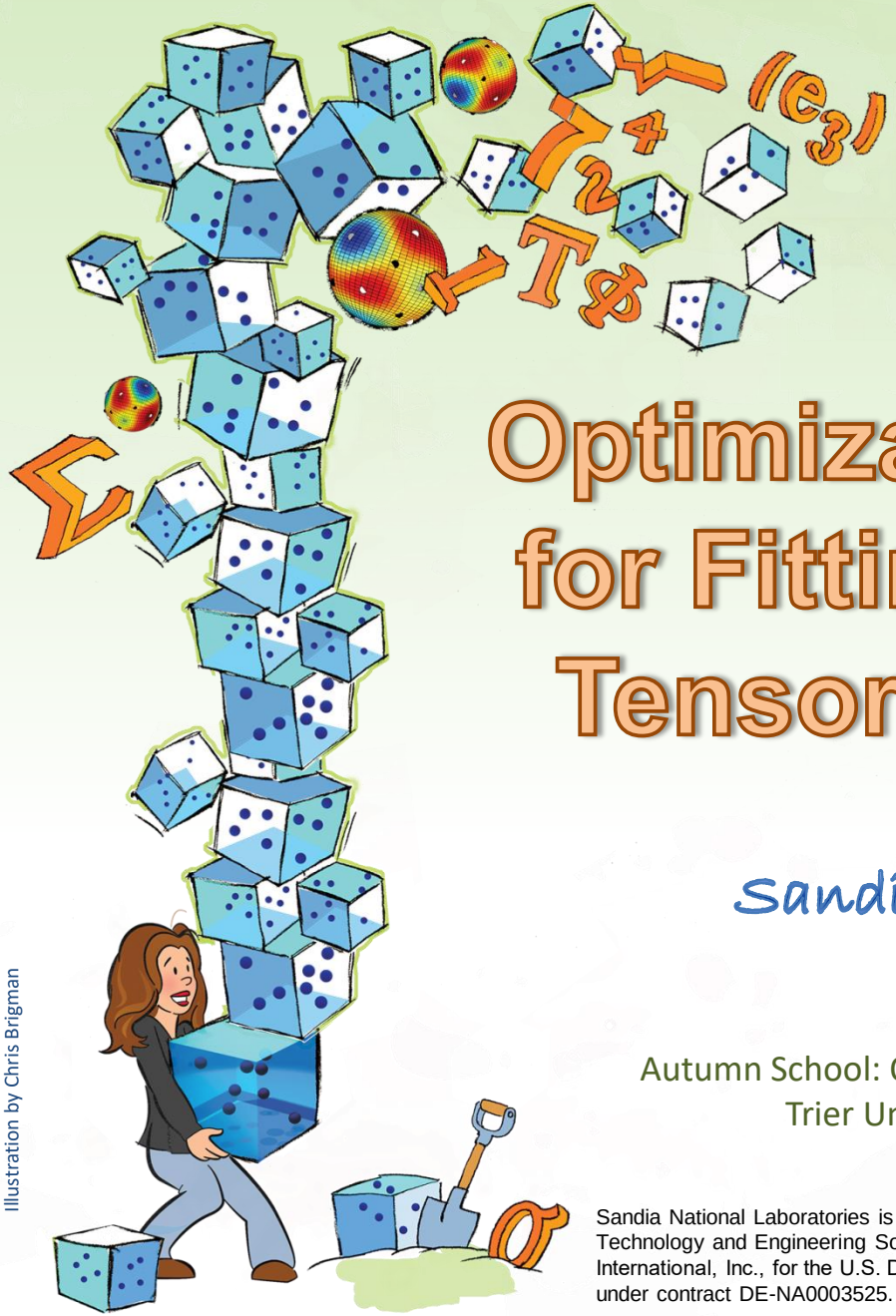


Illustration by Chris Brigman

Sandia National Laboratories

SAND2017-10099C

Optimization Approaches for Fitting the Canonical Tensor Decomposition

Tamara G. Kolda
Sandia National Laboratories
Livermore, CA

Autumn School: Optimization in Machine Learning and Data Science
Trier University, Trier, Germany, Aug. 28-31, 2017

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

U.S. DEPARTMENT OF ENERGY

Tensors and Their Decompositions

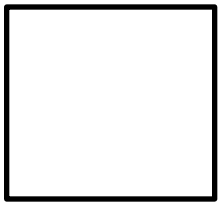
Kolda & Bader 2009

A Tensor is an d-Way Array

Vector
d = 1

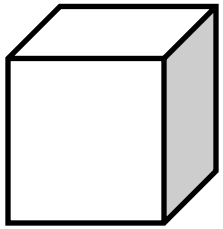
$\mathbf{x}$

Matrix
d = 2



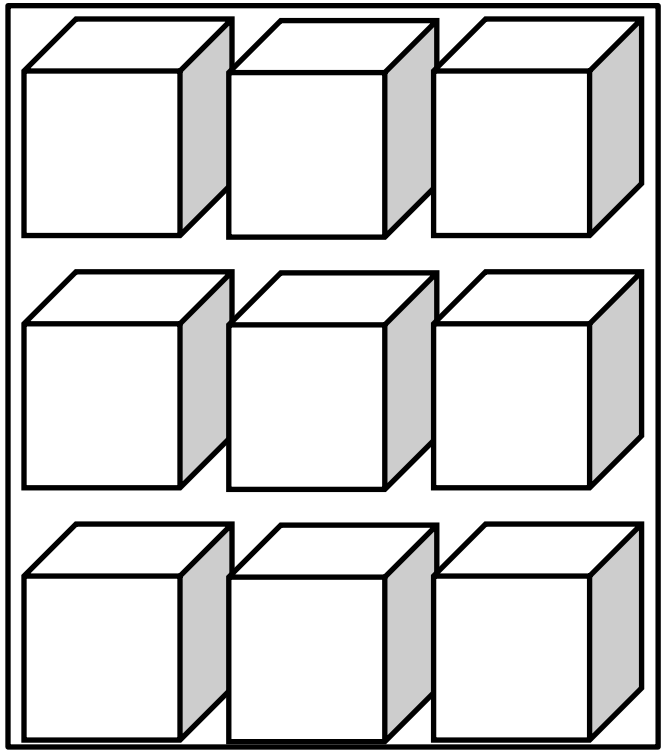
$\mathbf{X}$

3rd-Order Tensor
d = 3



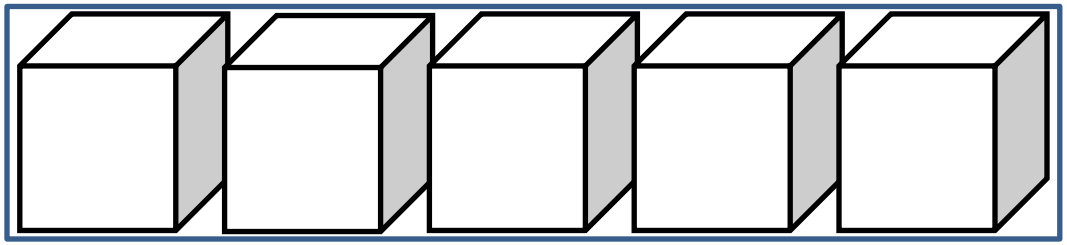
$\mathcal{X}$

5th-Order Tensor
d = 5



$\mathcal{X}$

4th-Order Tensor
d = 4

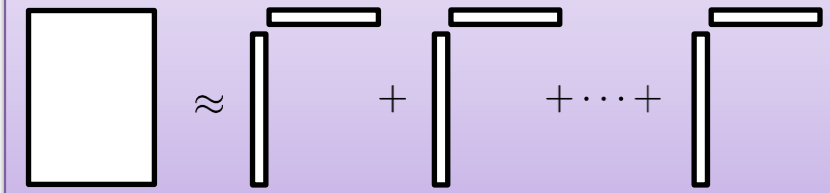


$\mathcal{X}$

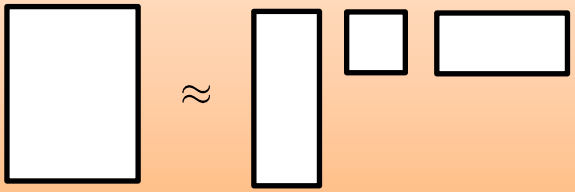
From Matrices to Tensors: Two Points of View

Singular value decomposition (SVD), eigendecomposition (EVD), nonnegative matrix factorization (NMF), sparse SVD, CUR, etc.

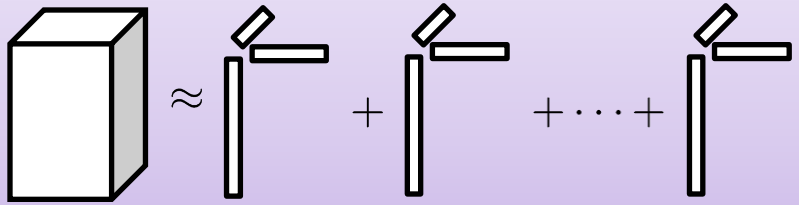
Viewpoint 1: Sum of outer products, useful for interpretation



Viewpoint 2: High-variance subspaces, useful for compression

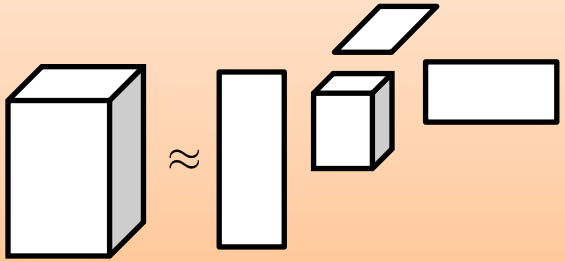


CP Model: Sum of d-way outer products, useful for interpretation



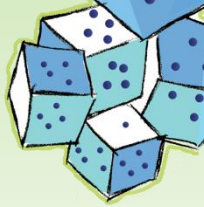
CANDECOMP, PARAFAC, Canonical Polyadic, CP

Tucker Model: Project onto high-variance subspaces to reduce dimensionality



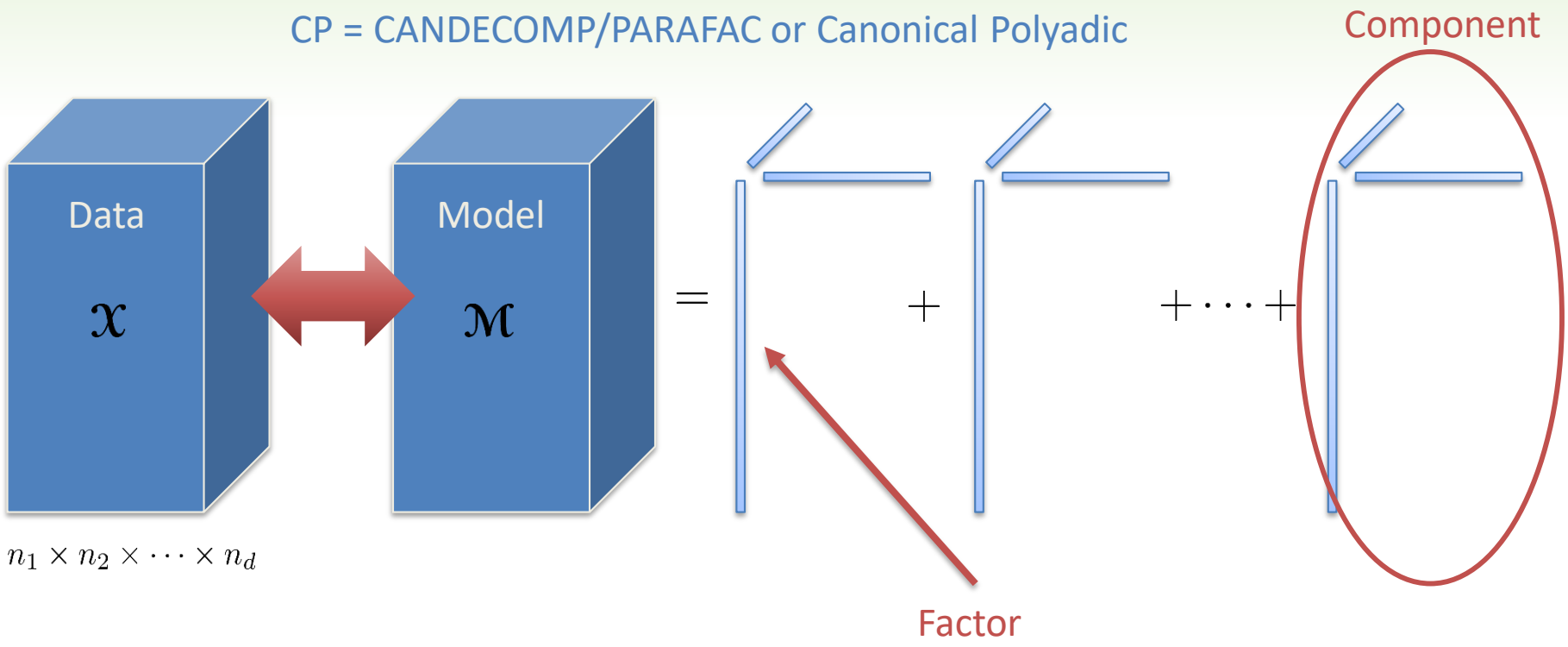
HO-SVD, Best Rank-$(R_1, R_2, \ldots, R_d)$ decomposition

Other models for compression include hierarchical Tucker and tensor train.



Introducing the CP Decomposition

CP Tensor Decomposition: Sum of Outer Products



Goal: $\min \sum_i (x_i - m_i)^2$ subject to $\mathcal{M}$ having "CP structure"

Hitchcock 1927, Harshman 1970, Carroll & Chang 1970

What's in a name?

- **Polyadic form of a tensor**, Hitchcock, 1927
- **CANDECOMP** or **CAND** (Canonical Decomposition), Carroll and Chang, 1970
- **PARAFAC** (Parallel Factors), Harshman, 1970
- **CP** = CANDECOMP/PARAFAC, proposed by Kiers 2000, or "canonical polyadic"
  - Reverse-engineering of the names, but recalls Hitchcock's original name
- **CPD** = CP decomposition

Richard Harshman 2000

Kolda & Bader 2009

Sandia National Laboratories

Motivation: CP for Mouse Neural Activity

Joint work with Alex Williams, Surya Ganguli, and others at Stanford University

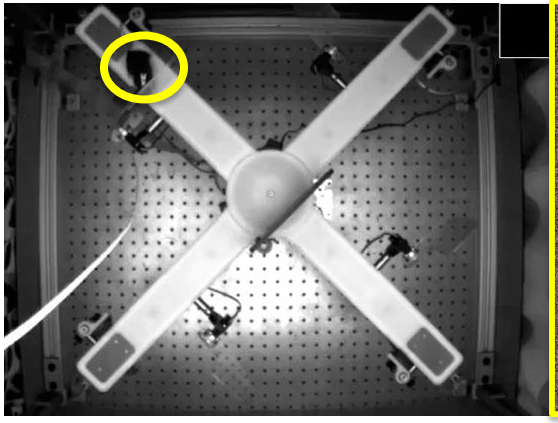
Alex Williams

Motivating Example: Neuron Activity in Learning

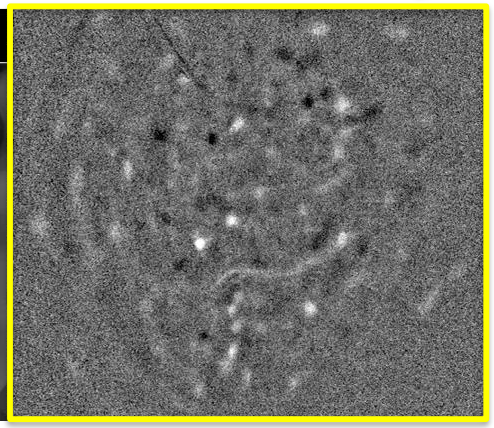
Thanks to Schnitzer Group @ Stanford
Mark Schnitzer, Fori Wang, Tony Kim

Microscope by Inscopix

mouse in "maze"



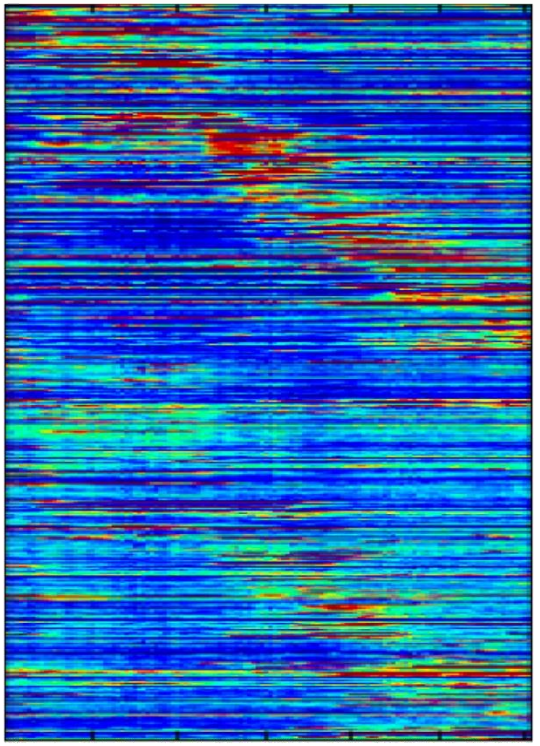
neural activity



One Column of Neuron x Time Matrix

One Trial

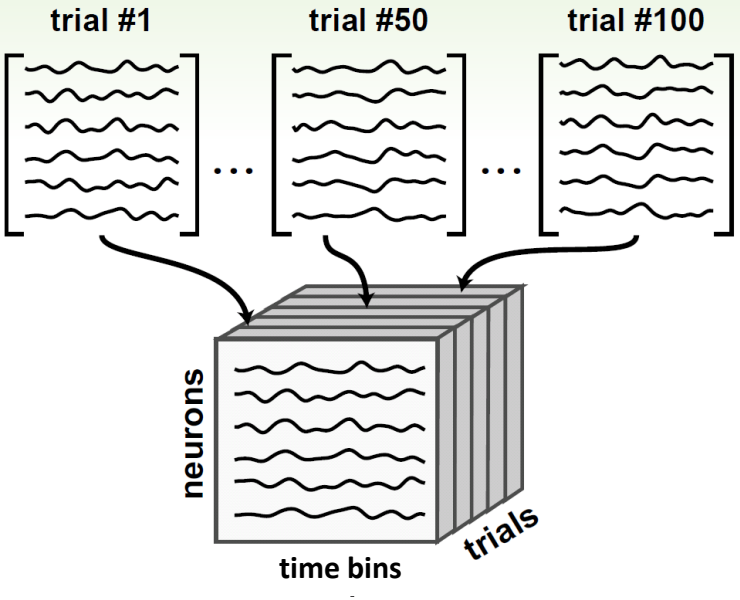
300 neurons × 120 time bins

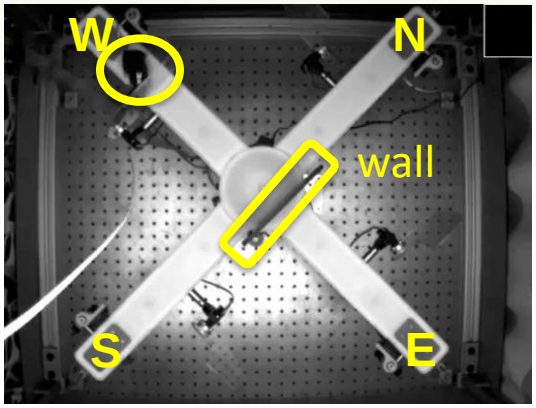


× 600 trials (over 5 days)

Williams, Ganguli, Kolda, et al. 2017

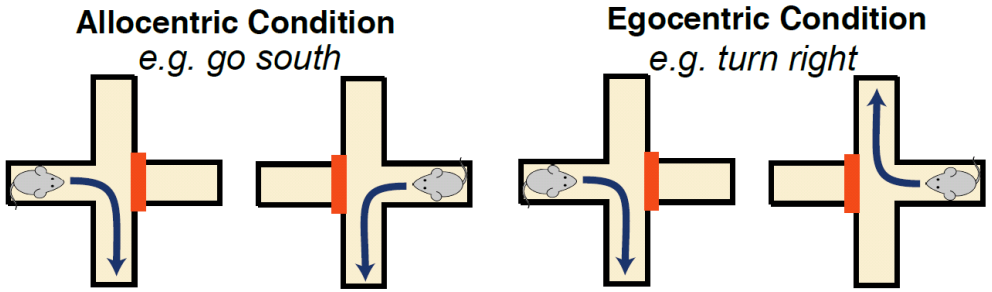
Trials Vary Start Position and Strategies





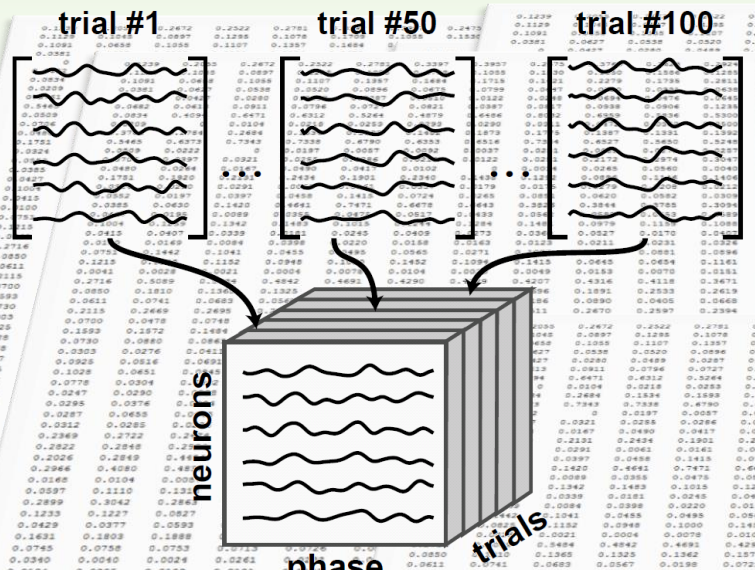
note different patterns on curtains

- 600 Trials over 5 Days
- Start West or East
- Conditions Swap Twice
  - ❖ Always Turn South
  - ❖ Always Turn Right
  - ❖ Always Turn South

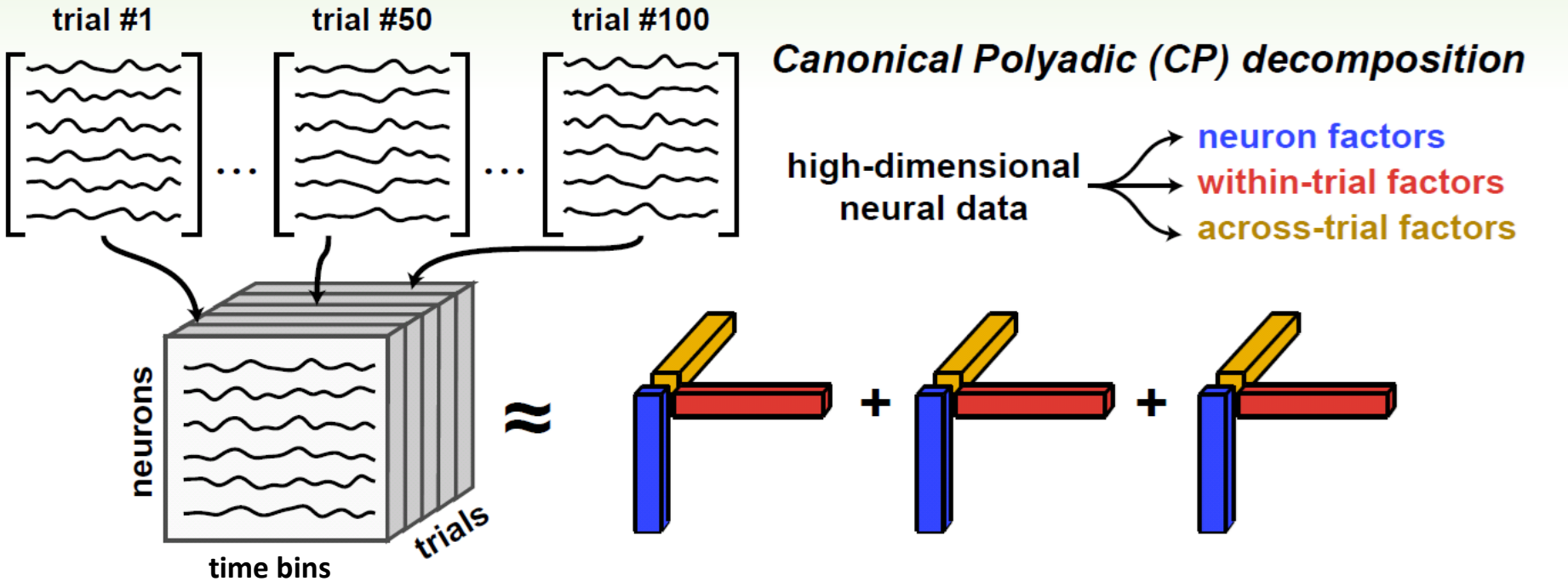


Williams, Ganguli, Kolda, et al. 2017

How to interpret so much raw data????



CP for Simultaneous Analysis of Neurons, Time, and Trial

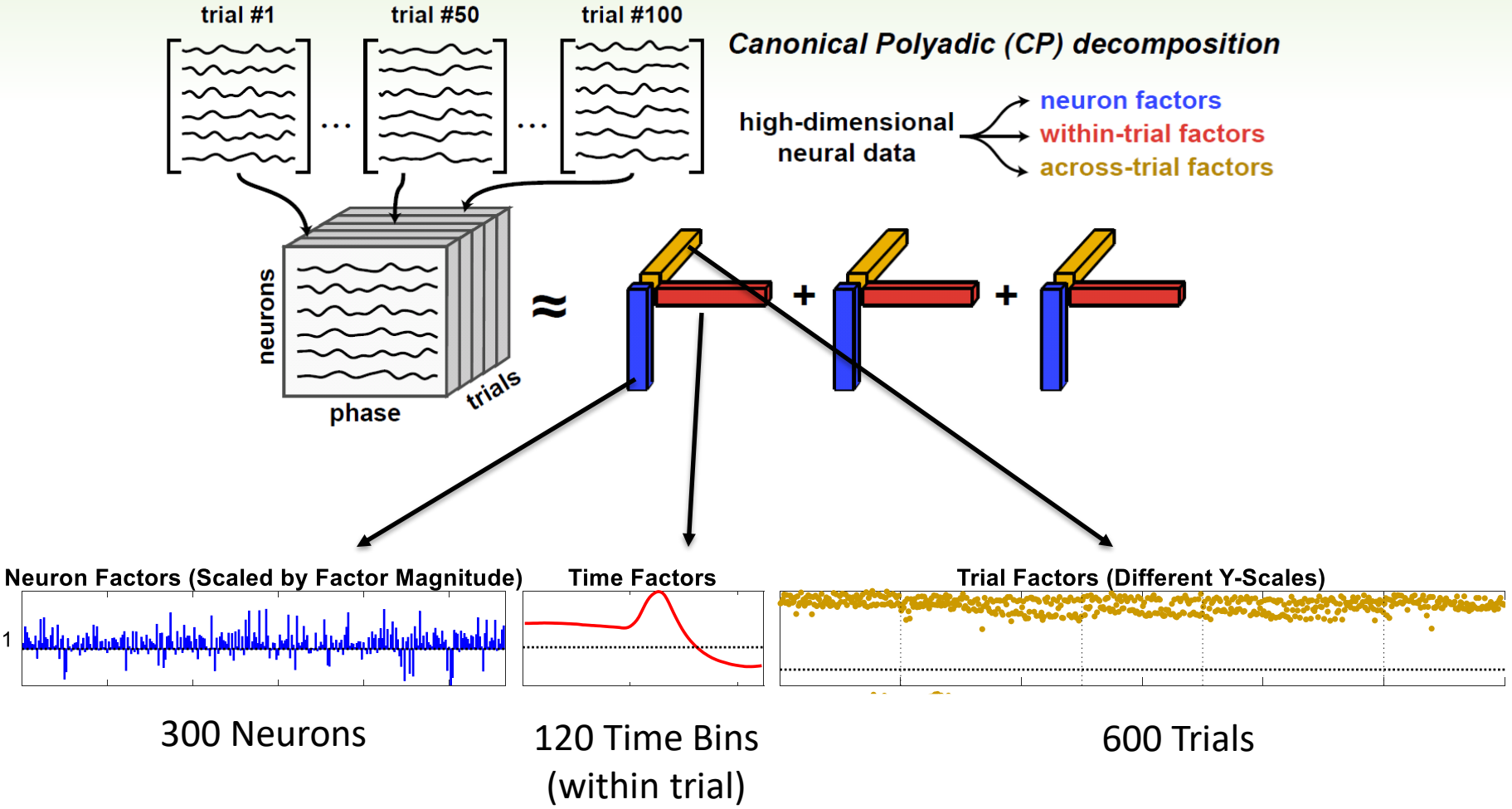


Past neuron-level work could only look at 2 factors at once: Time x Neuron or Trial x Neuron

Prior tensor work in neuroscience for fMRI and EEG: Andersen and Rayens (2004), Mørup et al. (2004), Acar et al. (2007), De Vos et al. (2007), and more

Williams, Ganguli, Kolda, et al. 2017

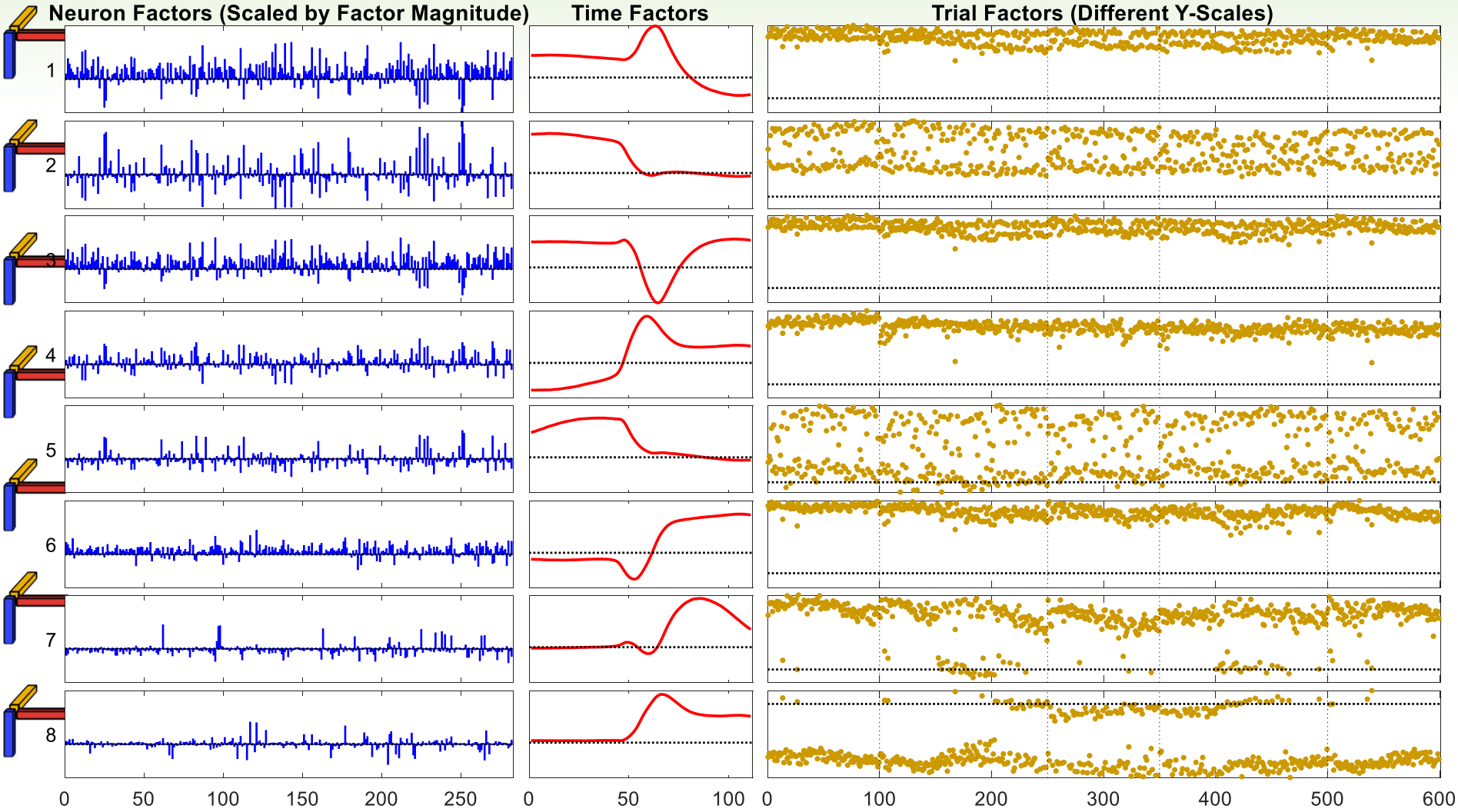
Results from 8-Component Model: Preview



8-Component CP Decomposition of Mouse Neuron Data

Sandia National Laboratories

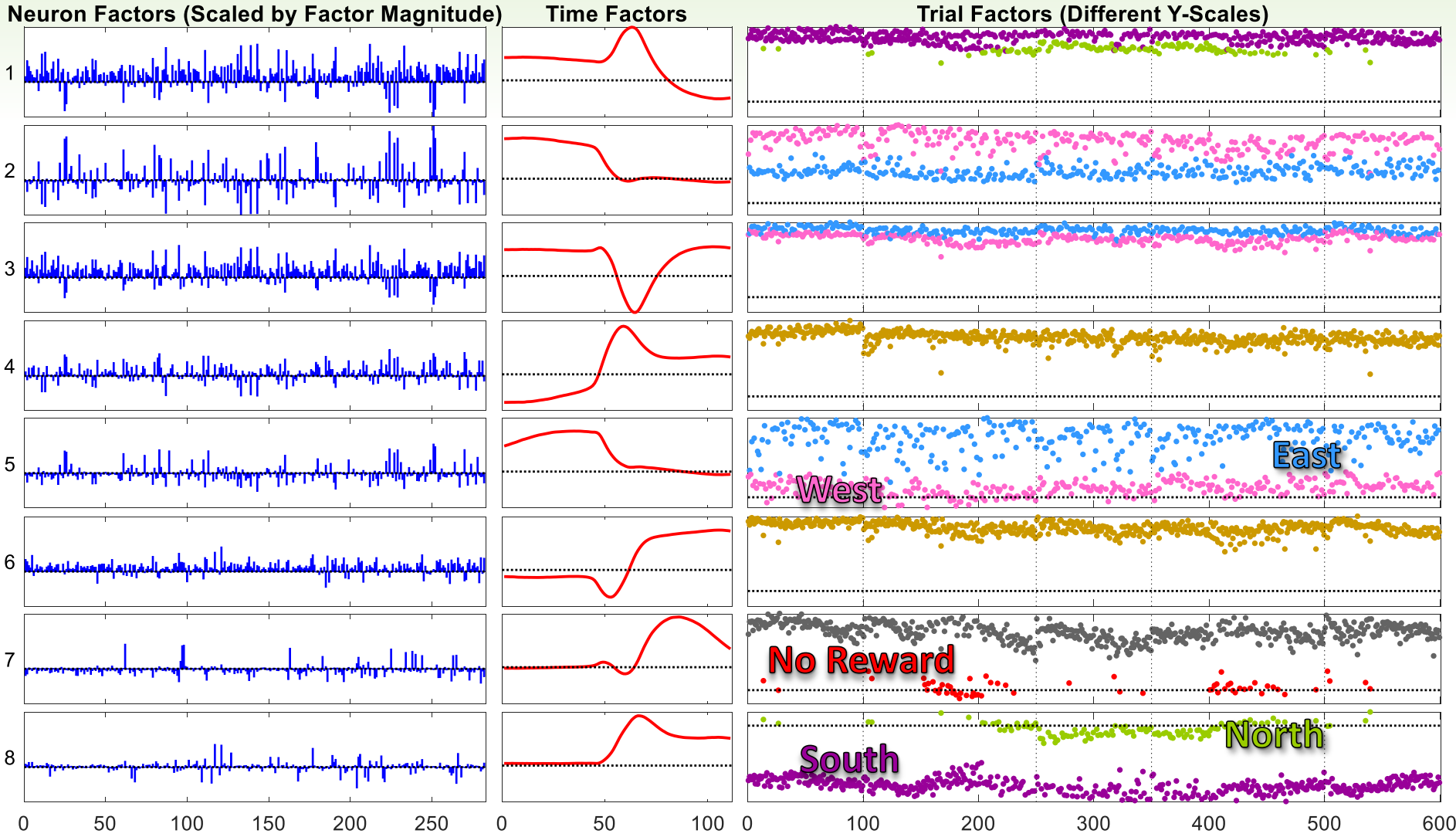
Williams, Ganguli, Kolda, et al. 2017

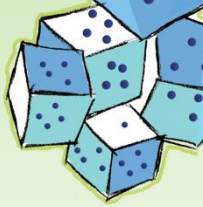


Interpretation of Mouse Neuron Data

Sandia National Laboratories

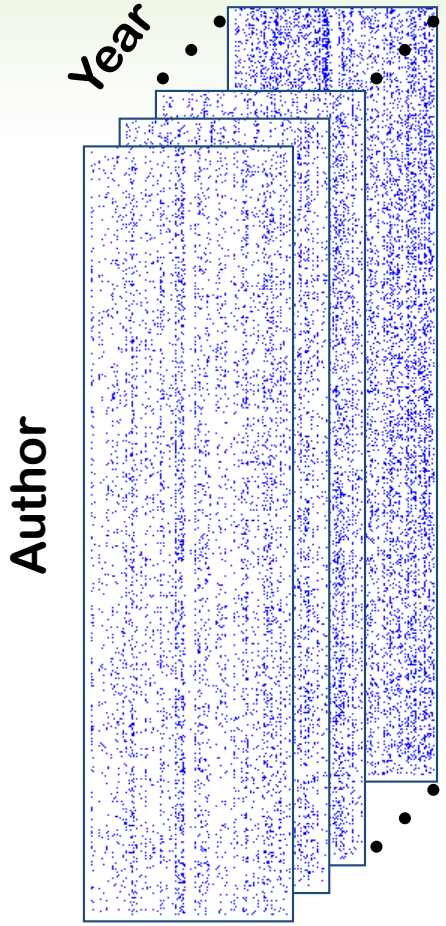
Williams, Ganguli, Kolda, et al. 2017





Sparse Example

Temporal Networks & Analysis



Tasks: Principal Components, Multidimensional Scaling, Clustering, Classification, Temporal Link Prediction

DBLP has data from 1936-2007
(used only "inproceedings" from 1991-2000)

Data	10 Years: 1991-2000
# Authors (min 10 papers)	7108
# Conferences	1103
Links	113k (0.14% dense)

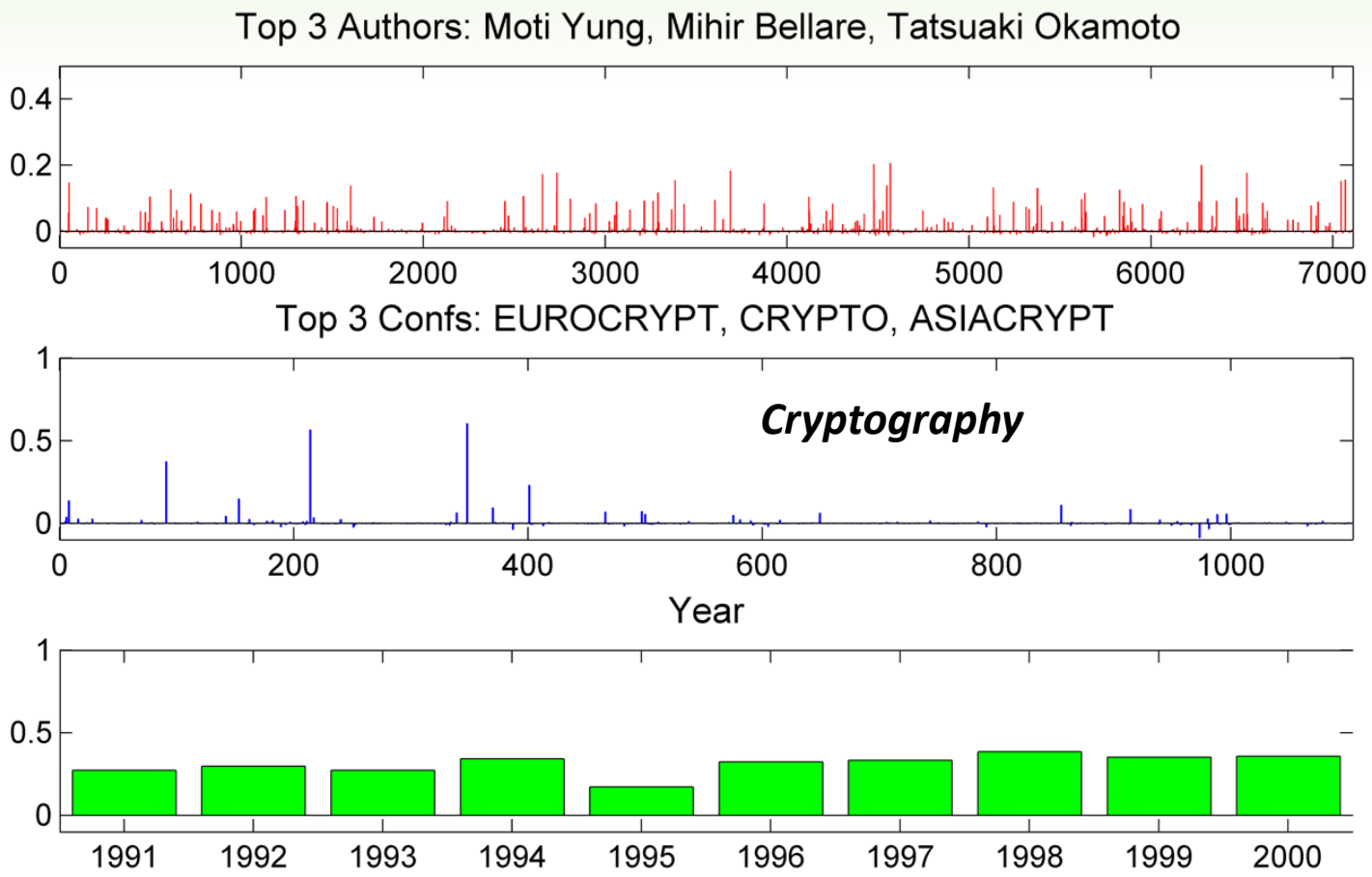
$c(i_1, i_2, i_3) =$ # papers by author i_1 at conference i_2 in year i_3

$$x_i = \begin{cases} \log(c_i) + 1 & \text{if } c_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

Let's look at some components sorted by size from a 50-component (R=50) factorization…

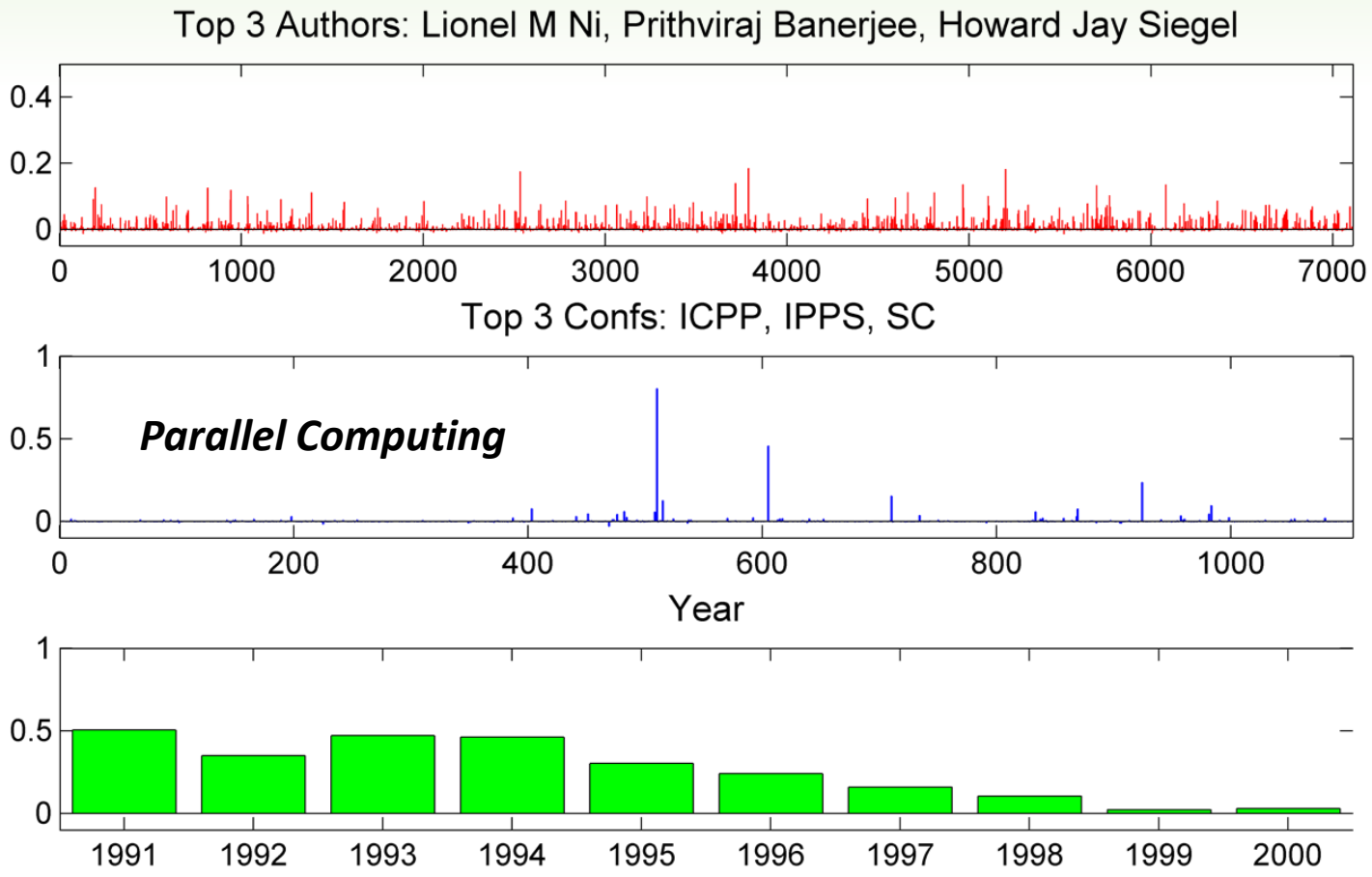
Acar, Dunlavy, & Kolda 2010

DBLP Component #30 (of 50)



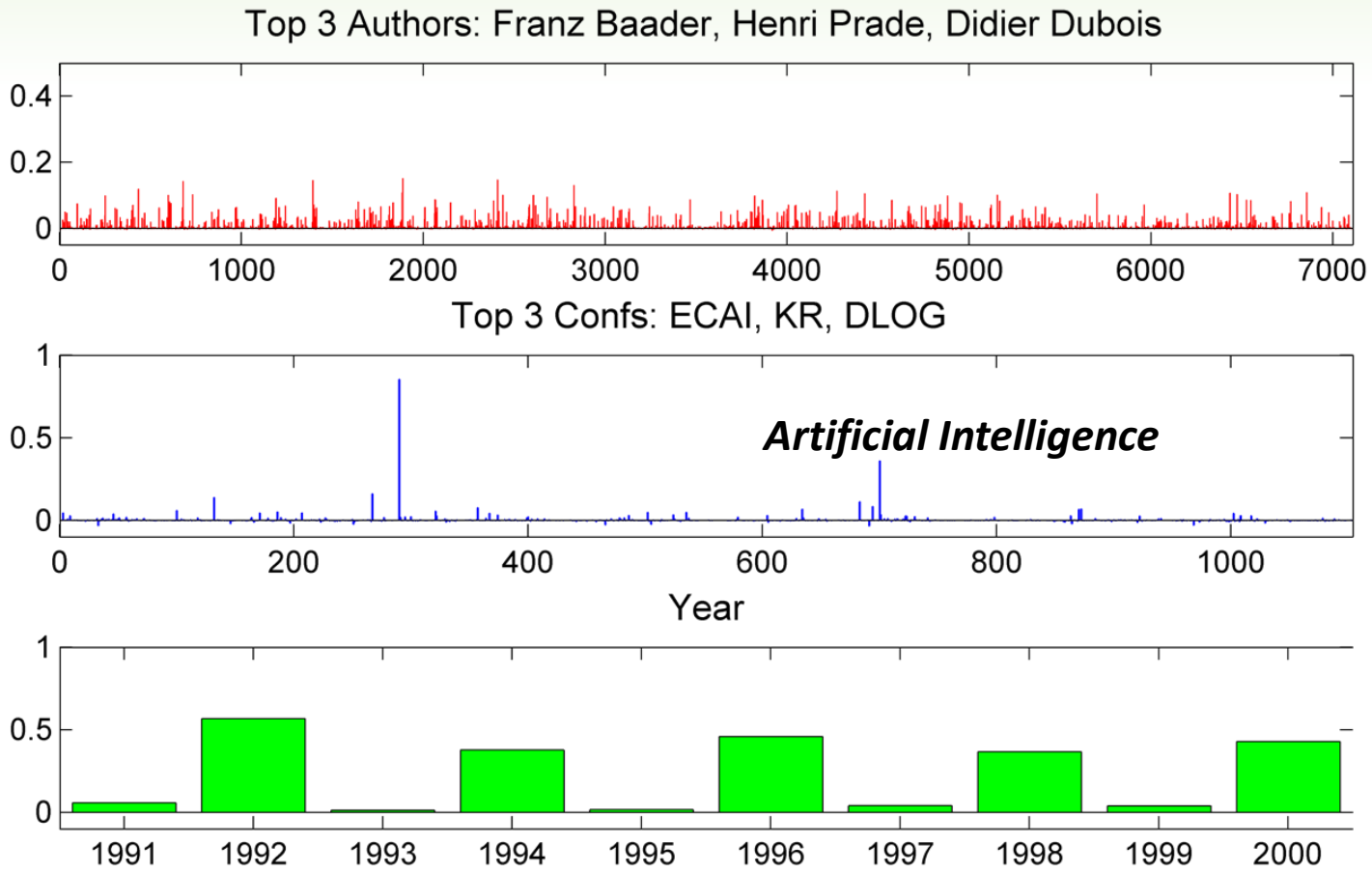
Acar, Dunlavy, & Kolda 2010

DBLP Component #19 (of 50)



Acar, Dunlavy, & Kolda 2010

DBLP Component #43 (of 50)

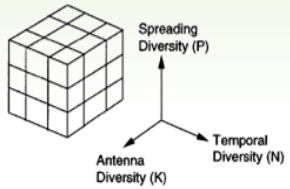


Acar, Dunlavy, & Kolda 2010

Tensor Factorizations have Numerous Applications

- Modeling fluorescence excitation-emission data (chemometrics)
- Signal processing
- Brain imaging (e.g., fMRI) data
- Network analysis and link prediction
- Image compression and classification; texture analysis
- Text analysis, e.g., multi-way LSI
- Approximating Newton potentials, stochastic PDEs, etc.
- Collaborative filtering
- Higher-order graph/image matching

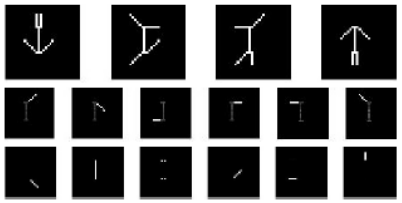
Furukawa, Kawasaki, Ikeuchi, and Sakauchi, *EGRW '02*



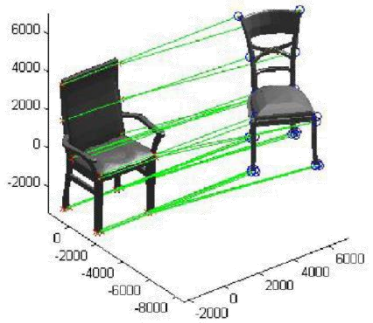
Sidiropoulos, Giannakis, Bro, *IEEE Trans. Signal Processing,* 2000

$$\mathcal{L}(x,t,\omega;u) = f(x,t,\omega) \quad (x,t) \in \mathcal{D} \times [0,T]$$
$$\mathcal{B}(x,t,\omega;u) = g(x,t) \quad (x,t) \in \partial\mathcal{D} \times [0,T]$$
$$\mathcal{I}(x,0,\omega;u) = h(x,\omega) \quad x \in \mathcal{D},$$

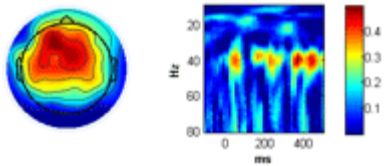
Doostan, Iaccarino, and Etemadi, *J. Computational Physics*, 2009



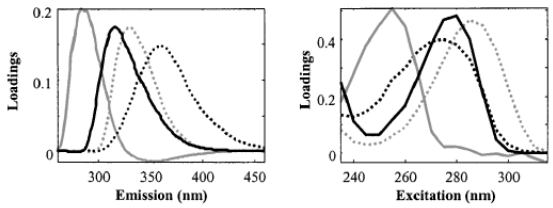
Hazan, Polak, and Shashua, *ICCV 2005*



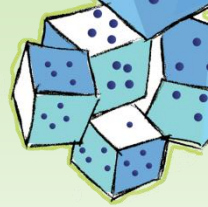
Duchenne, Bach, Kweon, Ponce, *TPAMI 2011*



ERPWAVELAB by Morten Mørup



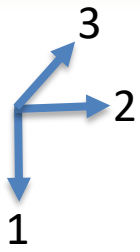
Andersen and Bro, *J. Chemometrics*, 2003



Background

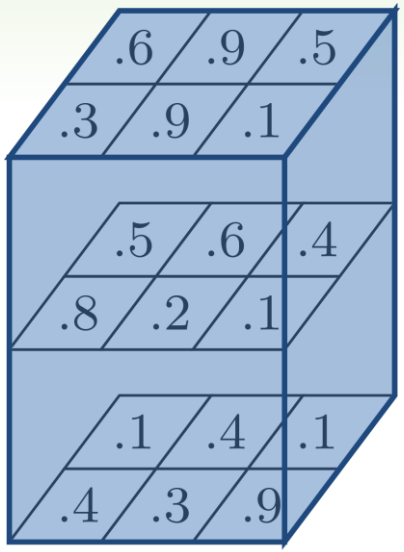
Working with Tensors

"Modes"



"Order" = Number of Modes or Ways

$$d = 3$$



$\mathcal{X}$

"Size"

$$n_1 \times n_2 \times n_3$$

$$3 \times 3 \times 2$$

"Frontal Slices"

$$\mathbf{X}(:,:,1) = \begin{bmatrix} 0.3 & 0.9 & 0.1 \\ 0.8 & 0.2 & 0.1 \\ 0.4 & 0.3 & 0.9 \end{bmatrix}$$

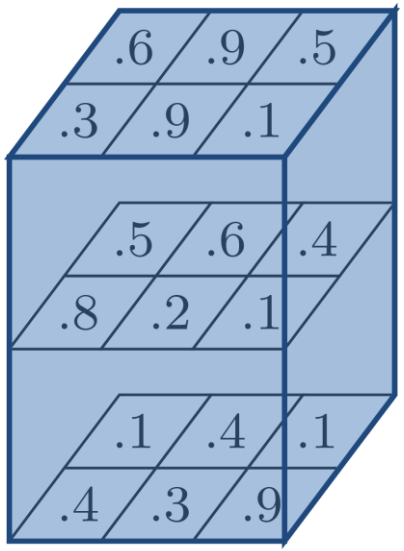
$$\mathbf{X}(:,:,2) = \begin{bmatrix} 0.6 & 0.9 & 0.5 \\ 0.5 & 0.6 & 0.4 \\ 0.1 & 0.4 & 0.1 \end{bmatrix}$$

This is how it's displayed in MATLAB.

The term "dimension" is overloaded and may be confusing.

Indexing a 3D Tensor

We use 1-based indexing throughout this talk to be consistent with MATLAB.

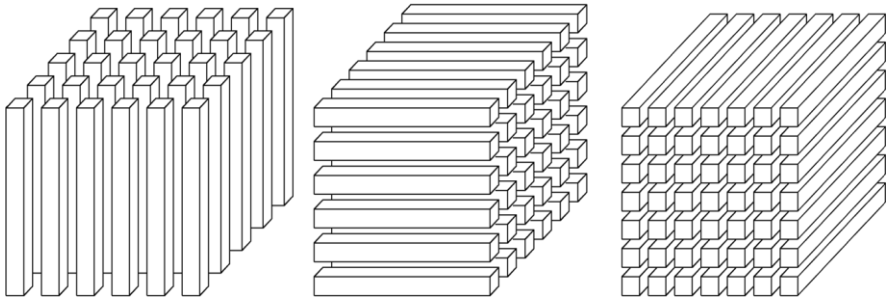


$\mathcal{X}$

Tensor Elements

$$x(1,1,1) = 0.3$$
$$x(2,1,1) = 0.8$$
$$x(1,1,2) = 0.6$$
$$x(3,2,1) = 0.3$$

Tensor Fibers



Mode-1 Fibers Mode-2 Fibers Mode-3 Fibers

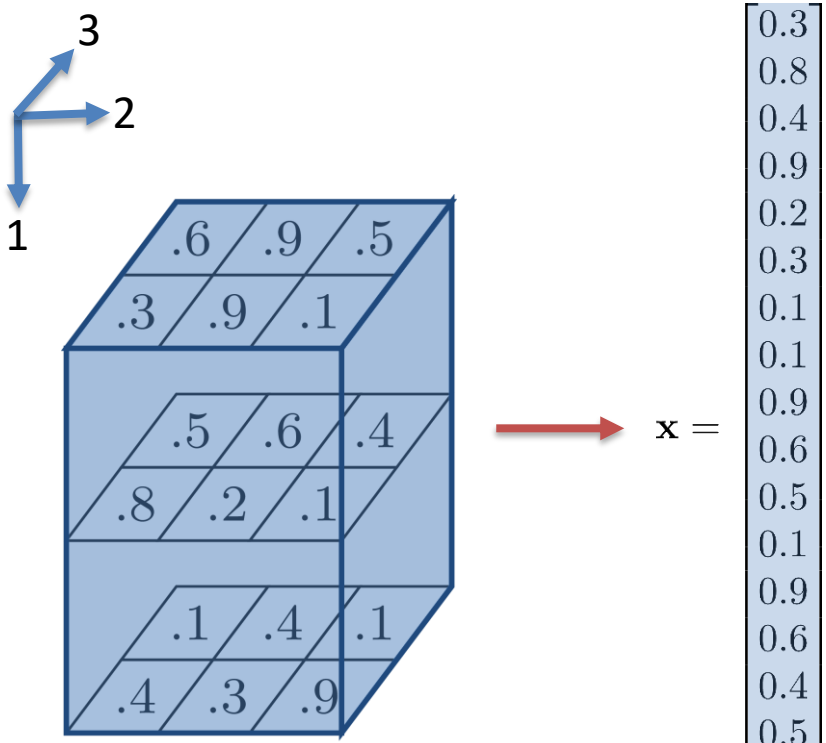
$$\mathbf{x}(:,1,1) = \begin{bmatrix} .3 \\ .8 \\ .4 \end{bmatrix} \qquad \mathbf{x}(1,1,:) = \begin{bmatrix} .3 \\ .6 \end{bmatrix}$$

$$\mathbf{x}(1,:,1) = \begin{bmatrix} .3 \\ .9 \\ .1 \end{bmatrix}$$

Tensor Hyperslices: Anything with at least two colons and one fixed index.
Ex: Frontal slices on previous slide.

Vectorizing a 3D Tensor (or How It's Stored in Memory!)

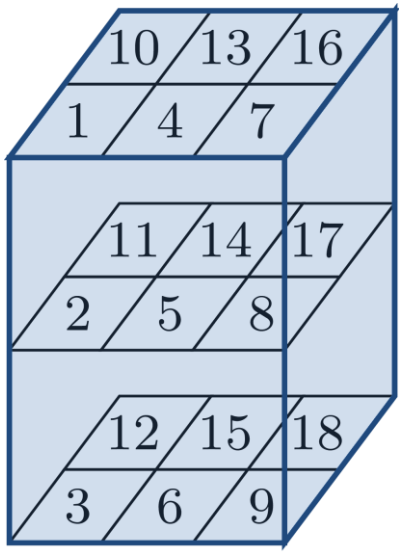
$$(i_1, i_2, i_3) \rightarrow i' = (i_3 - 1)n_2 n_1 + (i_2 - 1)n_1 + i_1$$



$\mathcal{X}$

$n_1 \times n_2 \times n_3$

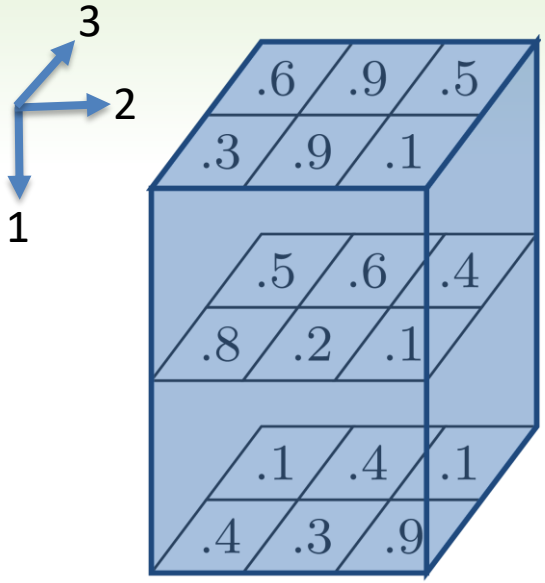
Ordering



$\mathcal{X}$

This is MATLAB's default order.
It may not be everyone's.

Unfolding a 3D Tensor



$\mathcal{X}$

$n_1 \times n_2 \times n_3$

$$(i_1, i_2, i_3) \rightarrow (i_i, i'_1), \quad i'_1 = (i_3 - 1)n_2 + i_2$$

$$\mathbf{X}_{(1)} = \begin{bmatrix} 0.3 & 0.9 & 0.1 & 0.6 & 0.9 & 0.5 \\ 0.8 & 0.2 & 0.1 & 0.5 & 0.6 & 0.4 \\ 0.4 & 0.3 & 0.9 & 0.1 & 0.4 & 0.1 \end{bmatrix} \quad n_1 \times n_2 n_3$$

$$(i_1, i_2, i_3) \rightarrow (i_2, i'_2), \quad i'_2 = (i_3 - 1)n_1 + i_1$$

$$\mathbf{X}_{(2)} = \begin{bmatrix} 0.3 & 0.8 & 0.4 & 0.6 & 0.5 & 0.1 \\ 0.9 & 0.2 & 0.3 & 0.9 & 0.6 & 0.4 \\ 0.1 & 0.1 & 0.9 & 0.5 & 0.4 & 0.1 \end{bmatrix} \quad n_2 \times n_1 n_3$$

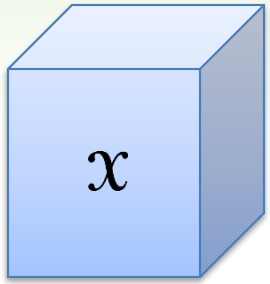
$$(i_1, i_2, i_3) \rightarrow (i_3, i'_3), \quad i'_3 = (i_2 - 1)n_1 + i_1$$

$$\mathbf{X}_{(3)} = \begin{bmatrix} 0.3 & 0.8 & 0.4 & 0.9 & 0.2 & 0.3 & 0.1 & 0.1 & 0.9 \\ 0.6 & 0.5 & 0.1 & 0.9 & 0.6 & 0.4 & 0.5 & 0.4 & 0.1 \end{bmatrix}$$

$n_3 \times n_1 n_2$

Unfolding also known as “matricization”.

Working with d-way Tensors



Size: $n_1 \times n_2 \times \cdots \times n_d$

Single Element: $x(i_1, i_2, \ldots, i_d)$ or x_i "multiindex"

Set of All Indices: $\mathcal{I} = \{ i \equiv (i_1, i_2, \ldots, i_d) \mid i_k \in \{ 1, \ldots, n_k \} \text{ for } k = 1, \ldots, d \}$

Geometric mean of the sizes: $n = \sqrt[d]{\prod_{k=1}^{d} n_k}$

Arithmetic mean of the sizes: $\bar{n} = \frac{1}{d} \sum_{k=1}^{d} n_k$

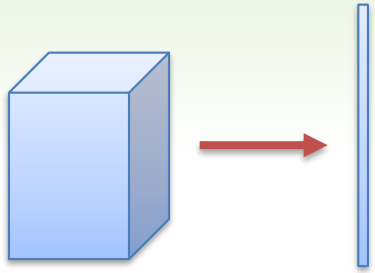
Total number of elements in the tensor:

$$|\mathcal{I}| = \prod_{k=1}^{d} n_k = n^d$$

Vectorizing a d-way Tensor

Order: d

Size: $n_1 \times n_2 \times \cdots \times n_d$



$$(i_1, i_2, \ldots, i_d) \in \mathcal{I} \rightarrow i' \in \{1, \ldots, n^d\}$$

$$i' = 1 + \sum_{k=1}^{d} (i_k - 1) n'_k$$

$$n'_k = \prod_{\ell=1}^{k-1} n_k$$

See MATLAB commands:
tt_sub2ind, tt_ind2sub

Tensor Size

$3 \times 4 \times 3 \times 2$

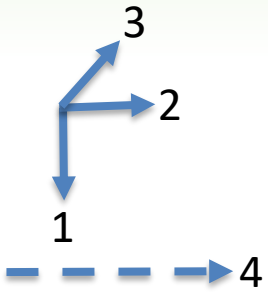
Example Mappings

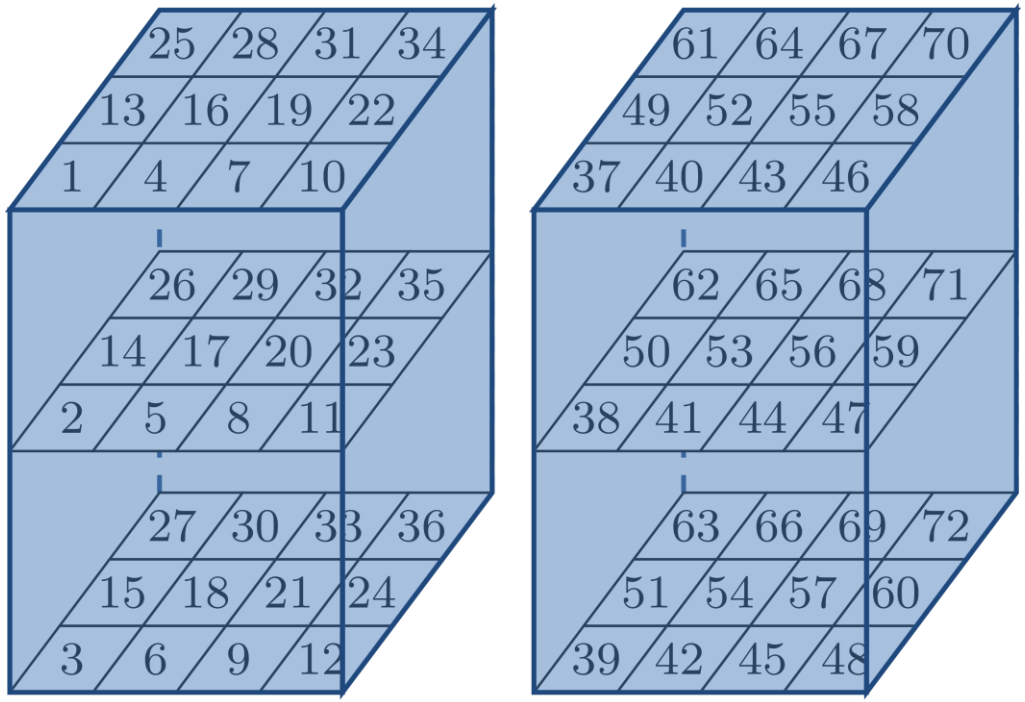
- (3,1,3,1) → 27
- (1,2,2,1) → 16
- (2,4,2,2) → 59
- (3,2,3,2) → 66
- (2,3,1,1) → 8

Example: Ordering of Elements in a 4D Tensor

Sandia National Laboratories

4^{th}-order tensor of size $3 \times 4 \times 3 \times 2$

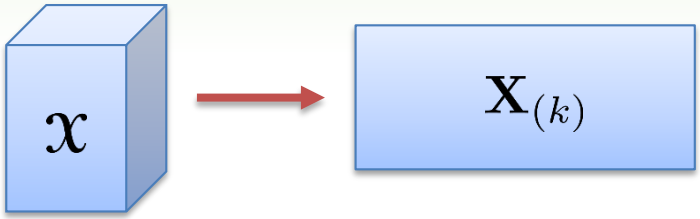




$\mathcal{X}(:,:,:,1)$ $\mathcal{X}(:,:,:,2)$

Unfolding a d-way Tensor

Matricization or Unfolding: Rearrange d-way array into 2-way array.



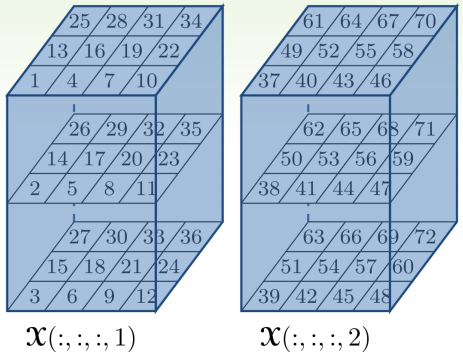
Size: $n_1 \times n_2 \times \cdots \times n_d$ Size: $n_k \times (n^d/n_k)$

Mode-k unfolding: $(i_1, i_2, \ldots, i_d) \rightarrow (i_k, i'_k)$

$$i'_k = 1 + \sum_{\ell=1}^{k-1} (i_\ell - 1) n'_\ell + \sum_{\ell=k+1}^{d} (i_\ell - 1)(n'_\ell / n_k) \qquad n'_k = \prod_{\ell=1}^{k-1} n_k$$

$$i'_k \in \{1, \ldots, n^d/n_k\}$$

Unfolding a 4D Tensor



$\mathcal{X}(:,:,:,1)$ $\mathcal{X}(:,:,:,2)$

Note the patterning!

block size = $n_k \times n'_k$

blocks = $n^d / n_k n'_k$

$$\mathbf{X}_{(1)} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & \cdots & 58 & 61 & 64 & 67 & 70 \\ 2 & 5 & 8 & 11 & 14 & 17 & \cdots & 59 & 62 & 65 & 68 & 71 \\ 3 & 6 & 9 & 12 & 15 & 18 & \cdots & 60 & 63 & 66 & 69 & 72 \end{bmatrix}$$

$$\mathbf{X}_{(2)} = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 & 25 & 26 & 27 & 37 & 38 & 39 & 49 & 50 & 51 & 61 & 62 & 63 \\ 4 & 5 & 6 & 16 & 17 & 18 & 28 & 29 & 30 & 40 & 41 & 42 & 52 & 53 & 54 & 64 & 65 & 66 \\ 7 & 8 & 9 & 19 & 20 & 21 & 31 & 32 & 33 & 43 & 44 & 45 & 55 & 56 & 57 & 67 & 68 & 69 \\ 10 & 11 & 12 & 22 & 23 & 24 & 34 & 35 & 36 & 46 & 47 & 48 & 58 & 59 & 60 & 70 & 71 & 72 \end{bmatrix}$$

$$\mathbf{X}_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 37 & \cdots & 48 \\ 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 49 & \cdots & 60 \\ 25 & 26 & 27 & 28 & 29 & 30 & 31 & 32 & 33 & 34 & 35 & 36 & 61 & \cdots & 72 \end{bmatrix}$$

$$\mathbf{X}_{(4)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & \cdots & 31 & 32 & 33 & 34 & 35 & 36 \\ 37 & 38 & 39 & 40 & 41 & 42 & \cdots & 67 & 68 & 69 & 70 & 71 & 72 \end{bmatrix}$$

Patterning: Battaglino, Perros, Sub, & Vuduc 2015; Austin, Ballard, & Kolda 2016

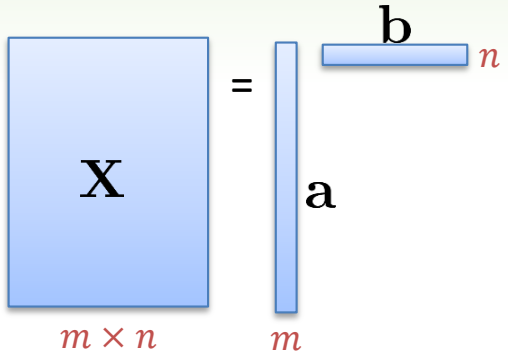
Vector Outer Product

Outer product of 2 vectors
⇒ rank-1 matrix

$$\mathbf{X} = \mathbf{a}\mathbf{b}'$$

$$\mathbf{X} = \mathbf{a} \circ \mathbf{b}$$

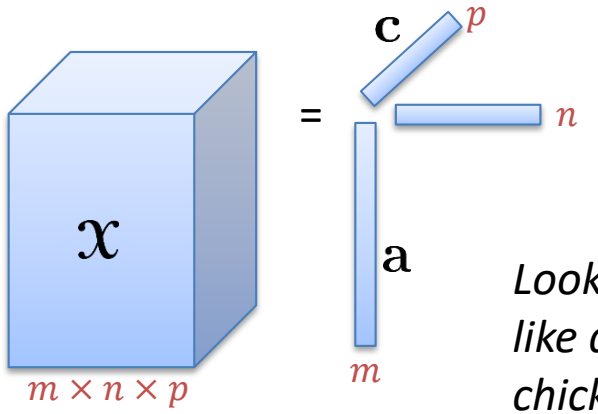
$$x(i_1, i_2) = a(i_1)\, b(i_2)$$



Outer product of 3 vectors
⇒ rank-1 tensor of order 3

$$\mathcal{X} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$$

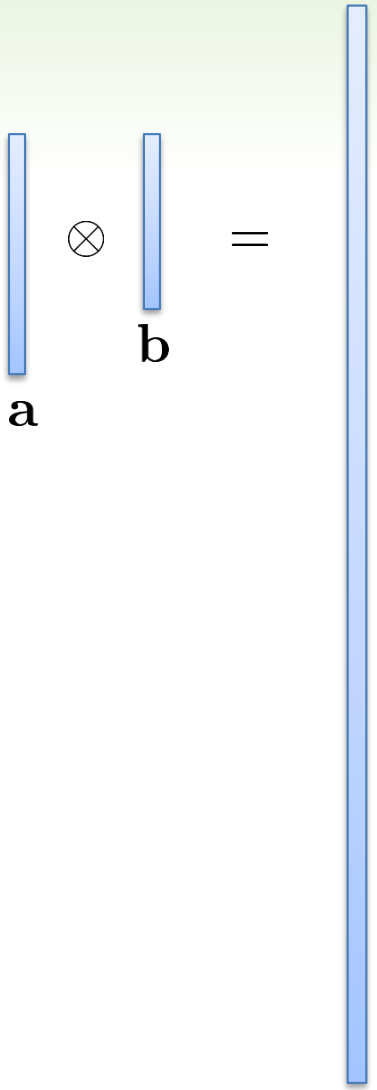
$$x(i_1, i_2, i_3) = a(i_1)\, b(i_2)\, c(i_3)$$



Looks like a chicken foot.

Vector Kronecker Product

$$\underset{m \quad n}{\mathbf{a} \otimes \mathbf{b}} = \underset{mn}{\begin{bmatrix} a(1)\,\mathbf{b} \\ a(2)\,\mathbf{b} \\ \vdots \\ a(m)\,\mathbf{b} \end{bmatrix}} = \begin{bmatrix} a(1)\,b(1) \\ a(1)\,b(2) \\ \vdots \\ a(1)\,b(n) \\ \vdots \\ a(m)\,b(1) \\ a(m)\,b(2) \\ \vdots \\ a(m)\,b(n) \end{bmatrix}$$



$$(\mathbf{a} \otimes \mathbf{b}) \otimes \mathbf{c} = \mathbf{a} \otimes (\mathbf{b} \otimes \mathbf{c})$$

Connection between Kronecker Product and Outer Products

Sandia National Laboratories

$$\mathbf{X} = \mathbf{a} \circ \mathbf{b} \quad \Leftrightarrow \quad \text{vec}(\mathbf{X}) = \mathbf{b} \otimes \mathbf{a}$$

$$\mathbf{X} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c} \quad \Leftrightarrow \quad \text{vec}(\mathbf{X}) = \mathbf{c} \otimes \mathbf{b} \otimes \mathbf{a}$$

$$\mathbf{X}_{(1)} = \mathbf{a}(\mathbf{c} \otimes \mathbf{b})'$$

$$\mathbf{X}_{(2)} = \mathbf{b}(\mathbf{c} \otimes \mathbf{a})'$$

$$\mathbf{X}_{(3)} = \mathbf{c}(\mathbf{b} \otimes \mathbf{a})'$$

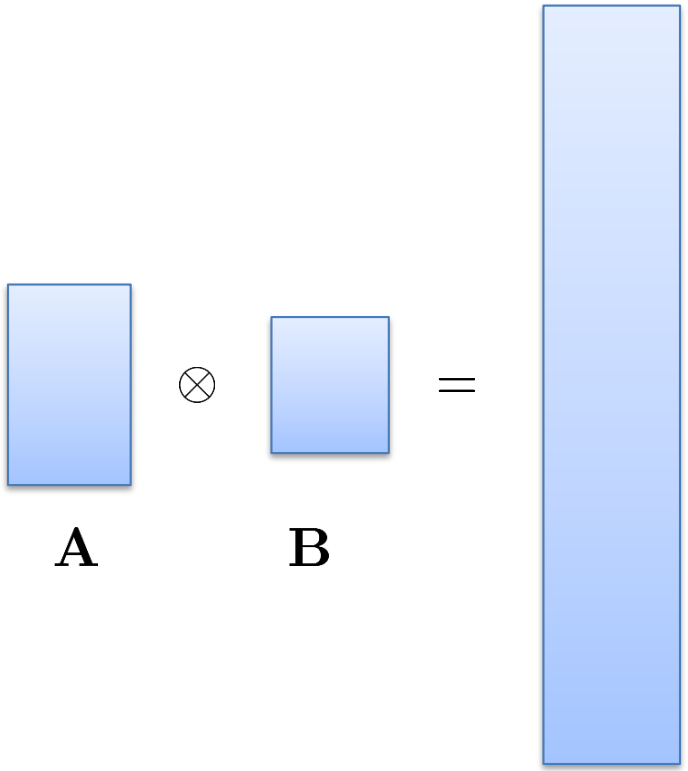
$$\mathcal{X} = \mathbf{a}_1 \circ \mathbf{a}_2 \circ \cdots \circ \mathbf{a}_d \quad \Leftrightarrow \quad \text{vec}(\mathbf{X}) = \mathbf{a}_d \otimes \cdots \otimes \mathbf{a}_1$$

$$\mathbf{X}_{(k)} = \mathbf{a}_k(\mathbf{a}_d \otimes \cdots \otimes \mathbf{a}_{k+1} \otimes \mathbf{a}_{k-1} \otimes \cdots \otimes \mathbf{a}_1)'$$

Matrix Khatri-Rao Product

Columnwise-Kronecker Product

$$\underset{m \times r}{\mathbf{A}} \odot \underset{n \times r}{\mathbf{B}} = \underset{mn \times r}{\left[\mathbf{a}(:,1) \otimes \mathbf{b}(:,1) \quad \mathbf{a}(:,2) \otimes \mathbf{b}(:,2) \quad \cdots \quad \mathbf{a}(:,r) \otimes \mathbf{b}(:,r)\right]}$$



Matrix Kronecker Product

$$\mathbf{A} \otimes \mathbf{B}$$

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a(1,1)\mathbf{B} & \cdots & a(1,n)\mathbf{B} \\ \vdots & \ddots & \vdots \\ a(m,1)\mathbf{B} & \cdots & a(m,n)\mathbf{B} \end{bmatrix}$$

Matrix Hadamard Product

Input two same-sized matrices

$$\mathbf{A} * \mathbf{B} = \begin{bmatrix} a(1,1)b(1,1) & a(1,2)b(1,2) & \cdots & a(1,n)b(1,n) \\ a(2,1)b(2,1) & a(2,2)b(2,2) & \cdots & a(2,n)b(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ a(m,1)b(m,1) & a(m,2)b(m,2) & \cdots & a(m,n)b(m,n) \end{bmatrix}$$

In MATLAB: A .* B

Pseudo-inverse

Let $\mathbf{A}$ be a matrix of size $n \times d$ where $n \gg d$ and its columns are linearly independent.

$$\mathbf{A}^{\dagger} = (\mathbf{A}'\mathbf{A})^{-1}\mathbf{A}'$$

This is useful for solving over-determined least squares problems.

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$$

$$\mathbf{A}'\mathbf{A}\mathbf{x} = \mathbf{A}'\mathbf{b}$$ *Normal Equations*

$$\mathbf{x}^* = \mathbf{A}^{\dagger}\mathbf{b}$$

MATLAB: A \ b

Properties

$$(\mathbf{A} \odot \mathbf{B}) \odot \mathbf{C} = \mathbf{A} \odot (\mathbf{B} \odot \mathbf{C})$$

$$(\mathbf{A} \odot \mathbf{B})'(\mathbf{A} \odot \mathbf{B}) = (\mathbf{A}'\mathbf{A} * \mathbf{B}'\mathbf{B})$$

$$\underset{mn \times r}{(\mathbf{A} \odot \mathbf{B})^{\dagger}} = \underset{r \times r}{((\mathbf{A}'\mathbf{A}) * (\mathbf{B}'\mathbf{B}))^{\dagger}} \underset{mn \times r}{(\mathbf{A} \odot \mathbf{B})'}$$

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \odot \mathbf{D}) = (\mathbf{AC} \odot \mathbf{BD})$$

Tensor Inner Product & Norm

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} x(i_1, i_2, \ldots, i_d) y(i_1, y_2, \ldots, y_d)$$

Shorthand: $\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i \in \mathcal{I}} x_i y_i$

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \text{vec}(\mathcal{X})' \text{vec}(\mathcal{Y})$$

$$\|\mathcal{X}\|^2 = \langle \mathcal{X}, \mathcal{X} \rangle = \sum_{i \in \mathcal{I}} x_i^2$$

MTTKRP: Matricized-Tensor Times Khatri-Rao Product

Tensor of size $n_1 \times n_2 \times \cdots \times n_d$: $\mathcal{X}$

Matrices of size $n_k \times r$ for $k = 1, \ldots, d$: $\mathbf{A}_1, \ldots, \mathbf{A}_d$

Matricized Tensor

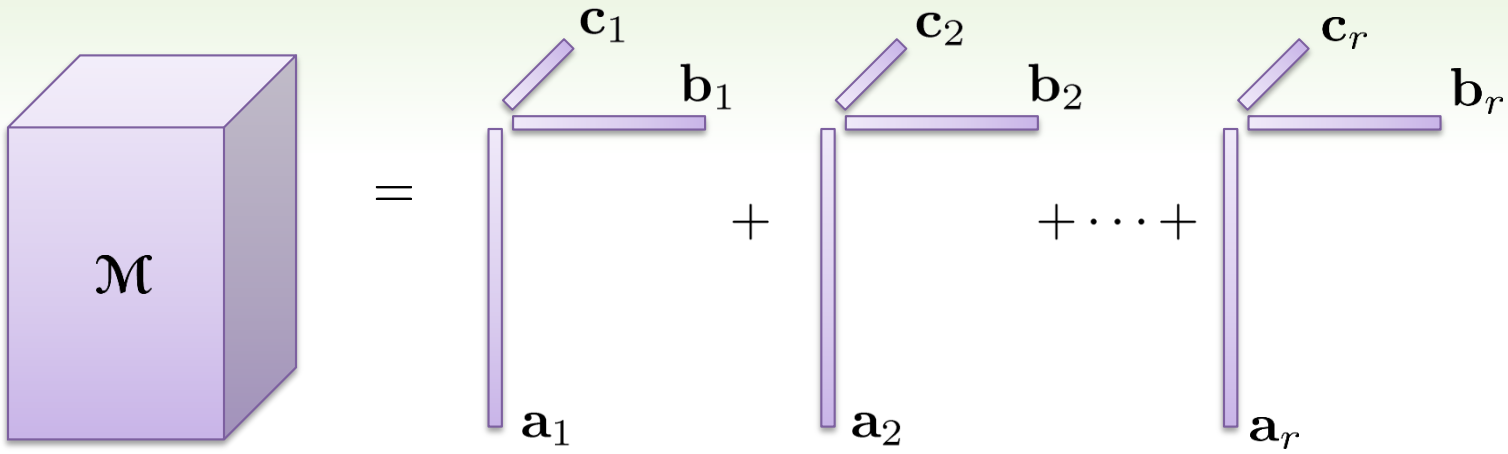
Khatro-Rao Product

$$\underbrace{\mathbf{B}}_{n_k \times r} = \underbrace{\mathbf{X}_{(k)}}_{n_k \times (n^d/n_k)} \underbrace{(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1)}_{(n^d/n_k) \times r}$$

$$b(i_k, j) = \sum_{i_1=1}^{n_1} \cdots \sum_{i_{k-1}=1}^{n_{k-1}} \sum_{i_{k+1}=1}^{n_{k+1}} \cdots \sum_{i_d=1}^{n_d} x(i_1, \ldots, i_d)\, a_1(i_1, j) \cdots a_{k-1}(i_{k-1}, j)\, a_{k+1}(i_{k+1}, j) \cdots a_d(i_d, j)$$

Bader & Kolda 2007

Ktensor: Sum of Outer Products

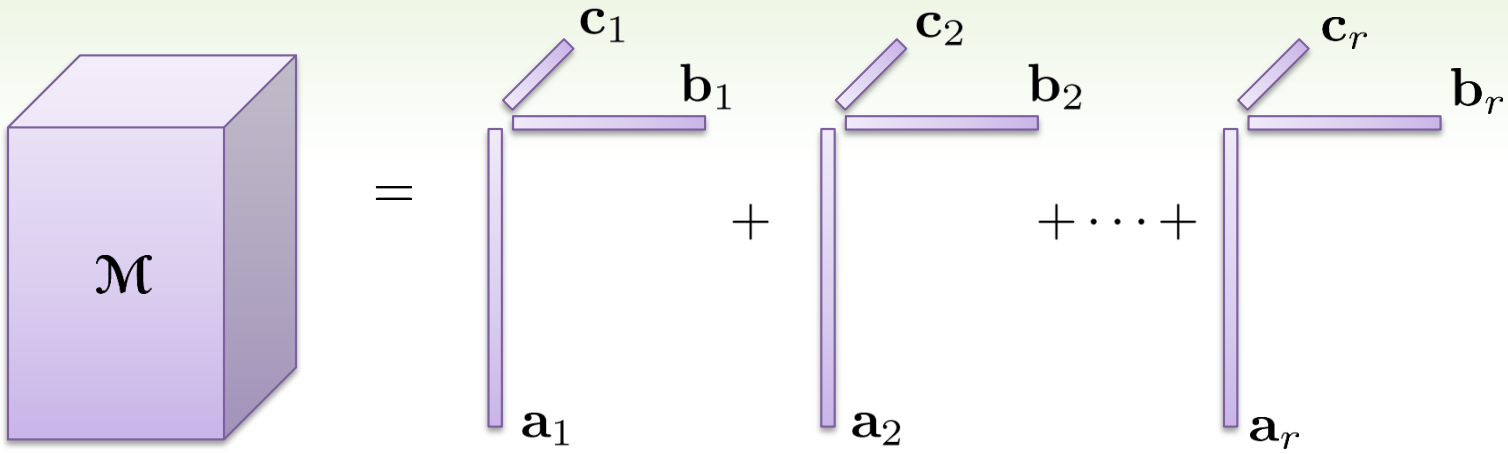


$$\mathcal{M} = \mathbf{a}_1 \circ \mathbf{b}_1 \circ \mathbf{c}_1 + \mathbf{a}_2 \circ \mathbf{b}_2 \circ \mathbf{c}_2 + \cdots + \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

$$m(i_1, i_2, i_3) = \sum_{j=1}^{r} a(i_1, j)\, b(i_2, j)\, c(i_3, j)$$

Bader & Kolda 2007

Ktensors and Khatri-Rao Products



$$\mathcal{M} = \mathbf{a}_1 \circ \mathbf{b}_1 \circ \mathbf{c}_1 + \mathbf{a}_2 \circ \mathbf{b}_2 \circ \mathbf{c}_2 + \cdots + \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

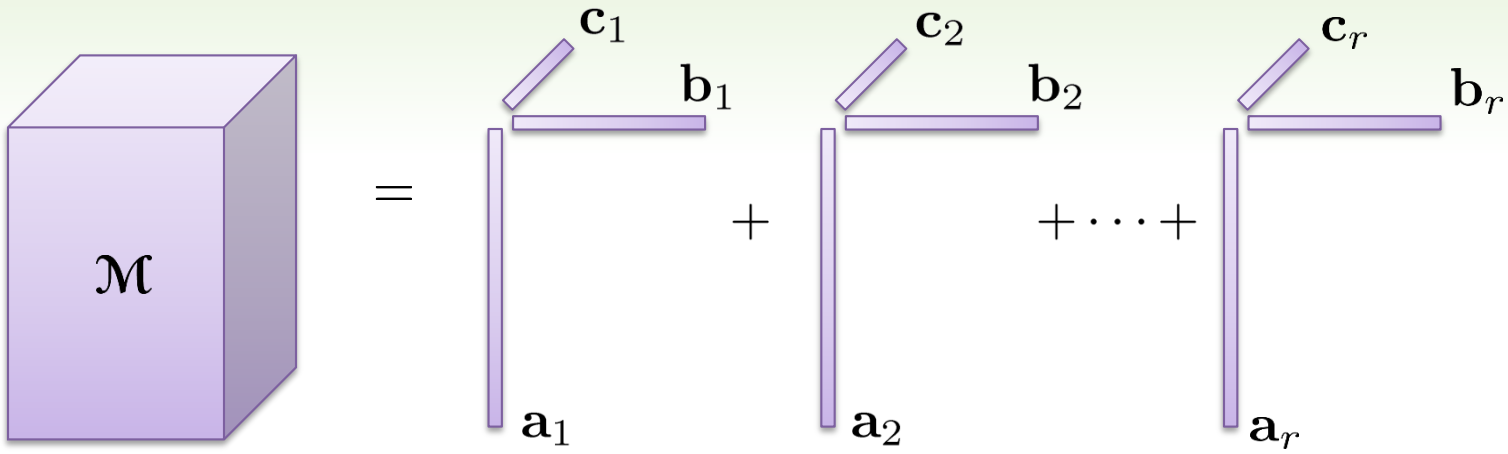
$$\mathbf{M}_{(1)} = \mathbf{a}_1(\mathbf{c}_1 \otimes \mathbf{b}_1)' + \mathbf{a}_2(\mathbf{c}_2 \otimes \mathbf{b}_2)' + \cdots + \mathbf{a}_r(\mathbf{c}_r \otimes \mathbf{b}_r)'$$

$$\mathbf{M}_{(1)} = \underbrace{\begin{bmatrix}\mathbf{a}_1 & \cdots & \mathbf{a}_r\end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix}\mathbf{c}_1 \otimes \mathbf{b}_1 & \cdots & \mathbf{c}_r \otimes \mathbf{b}_r\end{bmatrix}'}_{(\mathbf{C} \odot \mathbf{B})'}$$

$$\mathbf{M}_{(1)} = \mathbf{A}(\mathbf{C} \odot \mathbf{B})'$$

Bader & Kolda 2007

Ktensor Unfoldings



$$\mathcal{M} = \mathbf{a}_1 \circ \mathbf{b}_1 \circ \mathbf{c}_1 + \mathbf{a}_2 \circ \mathbf{b}_2 \circ \mathbf{c}_2 + \cdots + \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

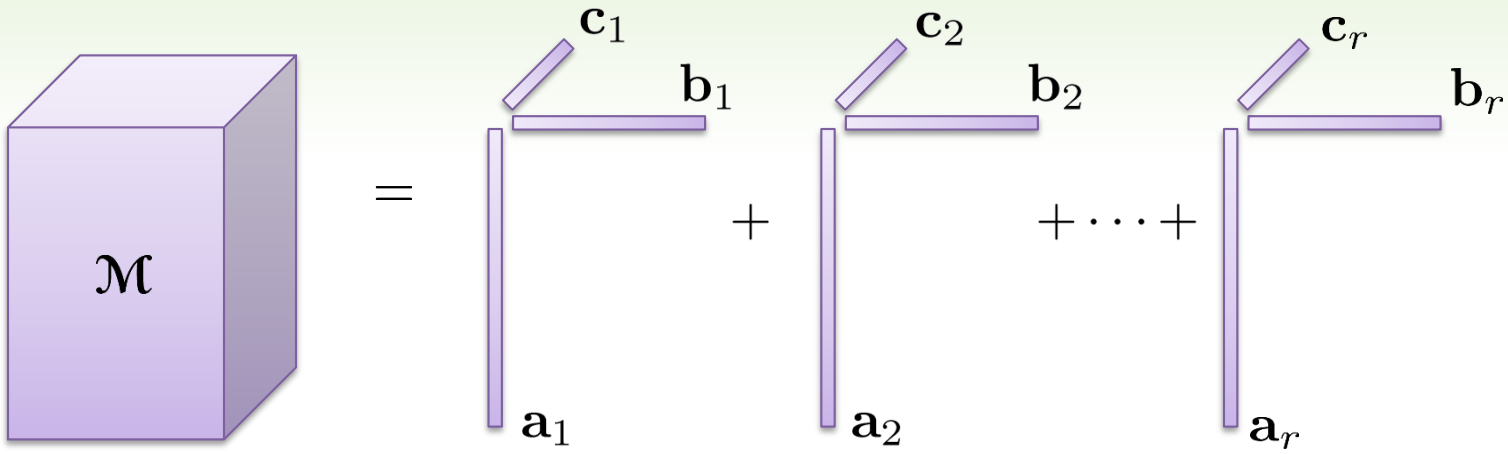
$$\mathbf{M}_{(1)} = \mathbf{A}(\mathbf{C} \odot \mathbf{B})'$$
$$\mathbf{M}_{(2)} = \mathbf{B}(\mathbf{C} \odot \mathbf{A})'$$
$$\mathbf{M}_{(3)} = \mathbf{C}(\mathbf{B} \odot \mathbf{A})'$$

Factor Matrices

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 \, \mathbf{a}_2 \cdots \mathbf{a}_r \end{bmatrix}$$
$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 \, \mathbf{b}_2 \cdots \mathbf{b}_r \end{bmatrix}$$
$$\mathbf{C} = \begin{bmatrix} \mathbf{c}_1 \, \mathbf{c}_2 \cdots \mathbf{c}_r \end{bmatrix}$$

Bader & Kolda 2007

Ktensor: Shorthand

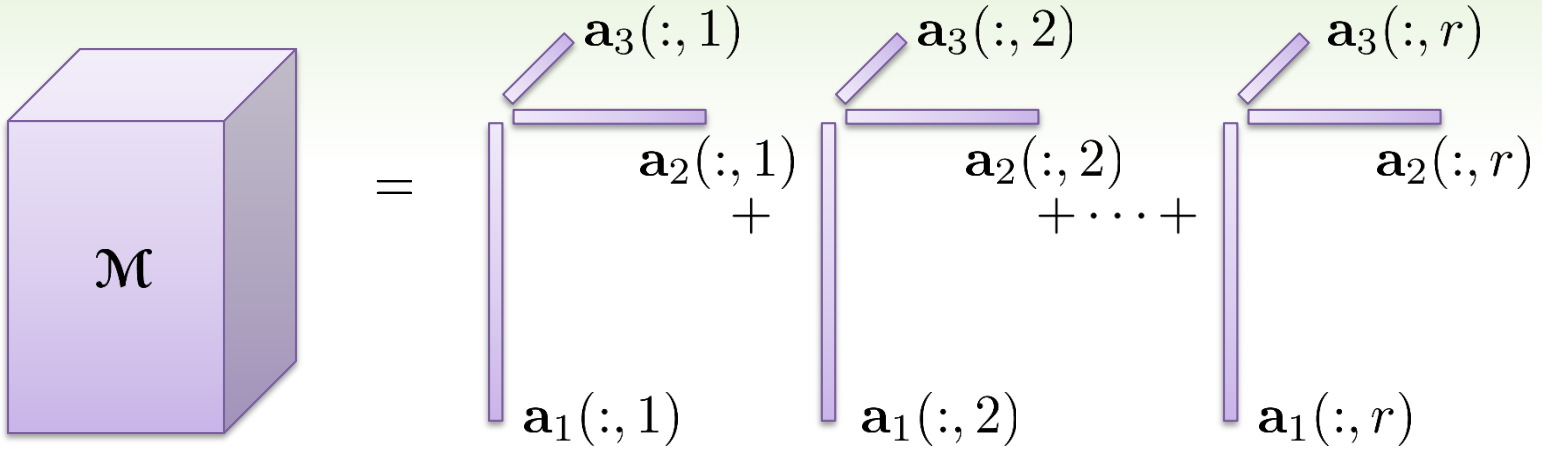


$$\mathcal{M} = \mathbf{a}_1 \circ \mathbf{b}_1 \circ \mathbf{c}_1 + \mathbf{a}_2 \circ \mathbf{b}_2 \circ \mathbf{c}_2 + \cdots + \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

$$\mathcal{M} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

Bader & Kolda 2007

Ktensor: Sum of Outer Products



$$\mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$$

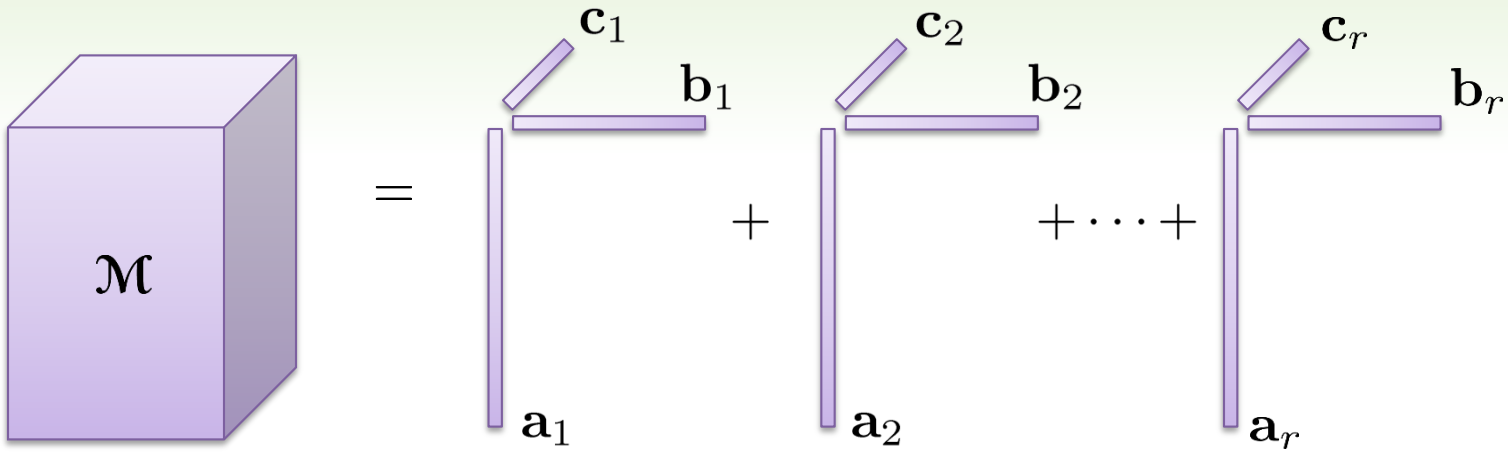
$$\mathcal{M} = \sum_{j=1}^{r} \mathbf{a}_1(:,j) \circ \mathbf{a}_2(:,j) \circ \cdots \circ \mathbf{a}_d(:,j)$$

$$m(i_1, i_2, \ldots, i_d) = \sum_{j=1}^{r} a_1(i_1, j)\, a_2(i_2, j) \cdots a_d(i_d, j)$$

$$\mathbf{M}_{(k)} = \mathbf{A}_k \left(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1\right)'$$

Bader & Kolda 2007

Scaling Ambiguity



$$\mathcal{M} = \mathbf{a}_1 \circ \mathbf{b}_1 \circ \mathbf{c}_1 + \mathbf{a}_2 \circ \mathbf{b}_2 \circ \mathbf{c}_2 + \cdots + \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

$$\mathcal{M} = \mathbf{a}_1 \circ 2\,\mathbf{b}_1 \circ {}^1\!/_2\,\mathbf{c}_1 + \mathbf{a}_2 \circ \mathbf{b}_2 \circ \mathbf{c}_2 + \cdots + \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

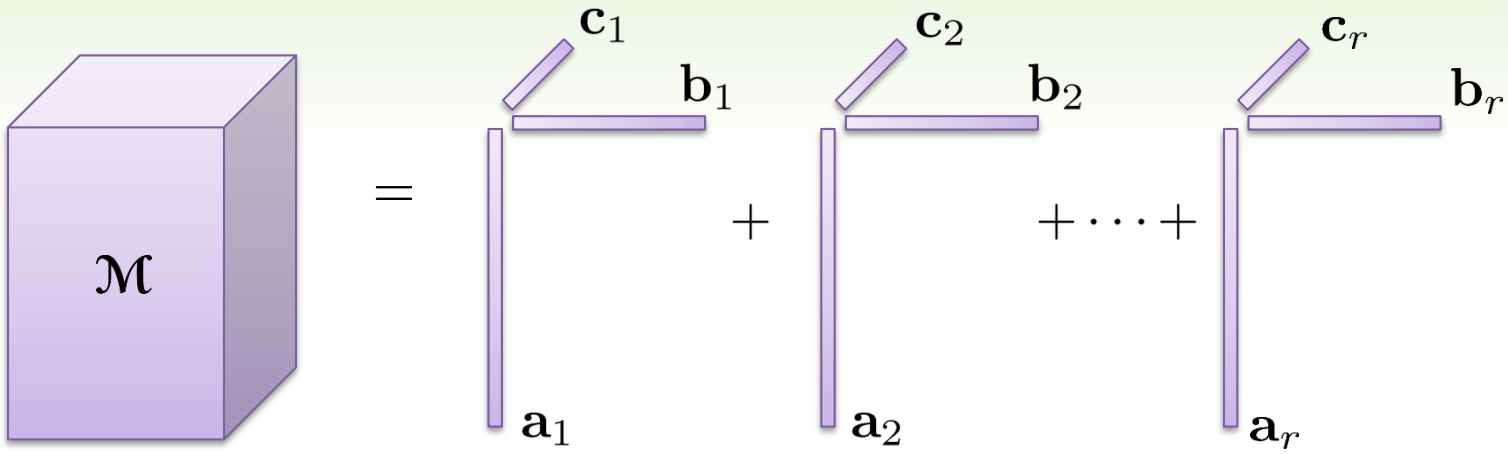
$$\mathcal{M} = \lambda_1\,\mathbf{a}_1 \circ \mathbf{b}_1 \circ \mathbf{c}_1 + \lambda_2\,\mathbf{a}_2 \circ \mathbf{b}_2 \circ \mathbf{c}_2 + \cdots + \lambda_r\,\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

Fixed by ktensor/normalize function.

$$\mathcal{M} = [\![\boldsymbol{\lambda}, \mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$$

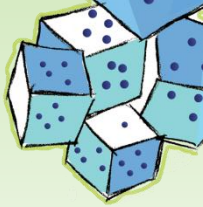
Bader & Kolda 2007

Permutation Ambiguity in Ktensor



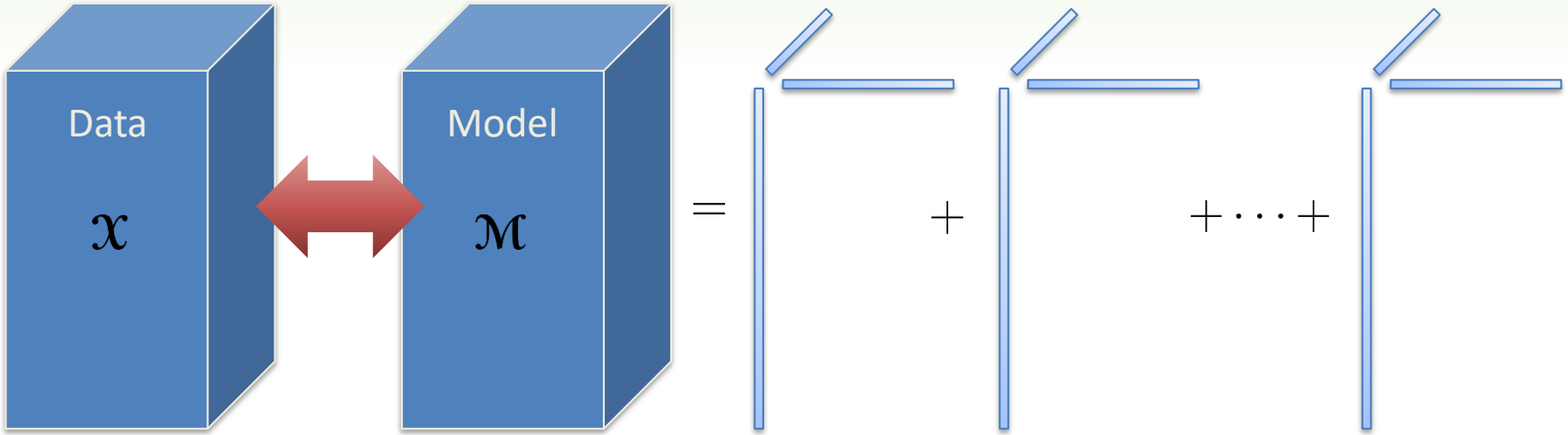
$$\mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!] = [\![\mathbf{A}_1\mathbf{\Pi}, \mathbf{A}_2\mathbf{\Pi}, \ldots, \mathbf{A}_d\mathbf{\Pi}]\!]$$

where $\mathbf{\Pi}$ is a $p \times p$ permutation matrix.



Computing CP

CP Objective Function

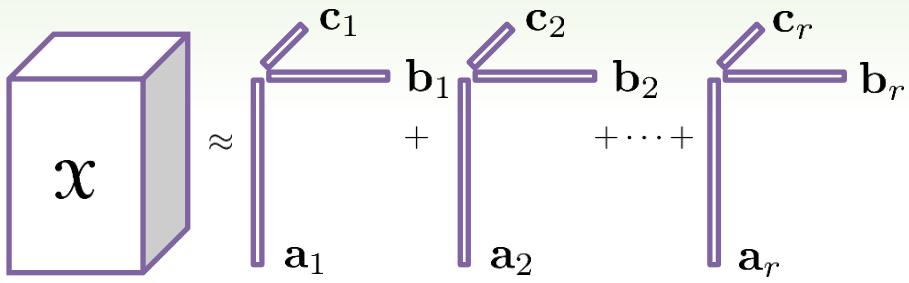


$$\min \sum_i (x_i - m_i)^2 \quad \text{subject to} \quad m_i = \sum_{j=1}^{r} a_1(i_1, j)\, a_2(i_2, j) \cdots a_d(i_d, j)$$

$$\min \|\mathcal{X} - \mathcal{M}\|^2 \quad \text{subject to} \quad \mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$$

Hitchcock 1927, Harshman 1970, Carroll & Chang 1970

CP Optimization Problem



Number of observations = n^d

Number of free parameters = $dr\bar{n}$

Typically, $dr\bar{n} \ll n^d$

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \|\mathcal{X} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|^2$$

Nonconvex

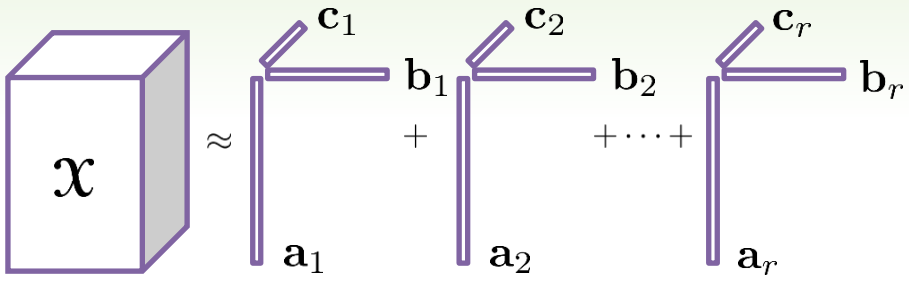
Permutation & Scaling Ambiguities

General Form

$$\min_{\{\mathbf{A}_k\}} \|\mathcal{X} - [\![\mathbf{A}_1,\mathbf{A}_2,\ldots,\mathbf{A}_d]\!]\|^2$$

How to pick r?

CP-ALS: Fitting CP Model via Alternating Least Squares



Convex (linear least squares) subproblems can be solved exactly

$$f(\mathbf{A},\mathbf{B},\mathbf{C}) = \|\mathcal{X} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|^2$$

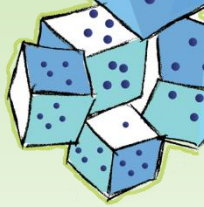
Repeat until convergence…

$$\min_{\mathbf{A}} \|\mathbf{X}_{(1)} - \mathbf{A}(\mathbf{C}\odot\mathbf{B})'\|^2$$

$$\min_{\mathbf{B}} \|\mathbf{X}_{(2)} - \mathbf{B}(\mathbf{C}\odot\mathbf{A})'\|^2$$

$$\min_{\mathbf{C}} \|\mathbf{X}_{(3)} - \mathbf{C}(\mathbf{B}\odot\mathbf{A})'\|^2$$

Special Structure of the Least Squares Problem



$$\min_{\mathbf{A}} \|\underbrace{\mathbf{X}_{(1)}}_{n_1 \times (n_2 n_3)} - \underbrace{\mathbf{A}}_{n_1 \times r} \underbrace{(\mathbf{C} \odot \mathbf{B})'}_{r \times (n_2 n_3)}\|^2$$

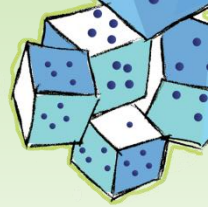
$$(\mathbf{C} \odot \mathbf{B})\mathbf{A}' = \mathbf{X}'_{(1)}$$

$$\mathbf{A}' = (\mathbf{C} \odot \mathbf{B})^{\dagger}\mathbf{X}'_{(1)}$$

$$\mathbf{A}' = (\mathbf{C}'\mathbf{C} * \mathbf{B}'\mathbf{B})^{\dagger}(\mathbf{C} \odot \mathbf{B})'\mathbf{X}'_{(1)}$$

$$\mathbf{A} = \underbrace{\mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})}_{\text{MTTKRP}}(\mathbf{C}'\mathbf{C} * \mathbf{B}'\mathbf{B})^{\dagger}$$

Special Structure of the Least Squares Problem: d-way



$$\min_{\mathbf{A}_k} \|\underbrace{\mathbf{X}_{(k)}}_{n_k \times (n^d/n_k)} - \underbrace{\mathbf{A}_k}_{n_k \times r} \underbrace{(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1)'}_{r \times (n^d/n_k)} \|^2$$

$$\mathbf{A}_k \leftarrow \mathbf{X}_{(k)}(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1)\mathbf{V}^\dagger$$

$$\mathbf{V} \leftarrow (\mathbf{A}_1'\mathbf{A}_1) * \cdots * (\mathbf{A}_{k-1}'\mathbf{A}_{k-1}) * (\mathbf{A}_{k+1}'\mathbf{A}_{k+1}) * \cdots * (\mathbf{A}_d'\mathbf{A}_d)$$

Exercise: Show that the cost is

$$rn^d + r(n^d/n_k) + r^2(\bar{n}+1) + r^3$$

CP-ALS Algorithm + Nuances

Sandia National Laboratories

1: $\text{fit} \leftarrow 1 - \left(\|\mathcal{X} - \mathcal{M}\| / \|\mathcal{X}\|\right)$

Roughly, the proportion of the data explained by the model.

2: **for** $\ell = 1, 2, \ldots$ **do**

3: $\quad \text{oldfit} \leftarrow \text{fit}$

4: $\quad$ **for** $k = 1, 2, \ldots, d$ **do**

5: $\quad\quad \mathbf{Z} \leftarrow \mathbf{X}_{(k)} \left(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1\right)$ {MTTKRP}

6: $\quad\quad \mathbf{V} \leftarrow (\mathbf{A}_1'\mathbf{A}_1) * \cdots * (\mathbf{A}_{k-1}'\mathbf{A}_{k-1}) * (\mathbf{A}_{k+1}'\mathbf{A}_{k+1}) * \cdots * (\mathbf{A}_d'\mathbf{A}_d)$

7: $\quad\quad \mathbf{A}_k \leftarrow \mathbf{Z}/\mathbf{V}$ {backsolve}

8: $\quad\quad \boldsymbol{\lambda} \leftarrow$ column norms of $\mathbf{A}_k$

9: $\quad\quad \mathbf{A}_k \leftarrow \mathbf{A}_k / \text{diag}(\boldsymbol{\lambda})$

Important to normalize!

$$\mathcal{M} = [\![\boldsymbol{\lambda}, \mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$$

10: $\quad$ **end for**

11: $\quad \text{fit} \leftarrow 1 - \left(\|\mathcal{X} - \mathcal{M}\| / \|\mathcal{X}\|\right)$

12: $\quad$ **if** $|\text{oldfit} - \text{fit}| < \tau$ **then**

13: $\quad\quad$ **break**

14: $\quad$ **end if**

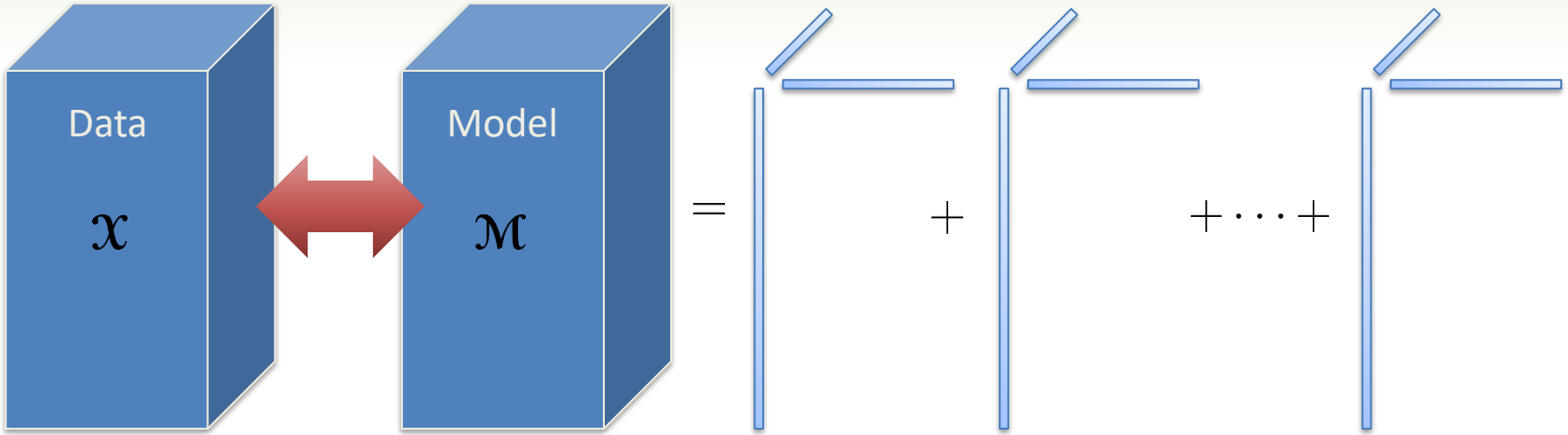
Stop when the fit stagnates.

15: **end for**

See MATLAB command: cp_als

Harshman 1970, Carroll & Chang 1970

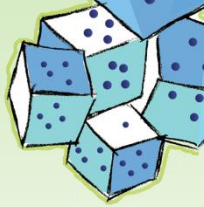
CP Objective Function



$$\min {}^1\!/_2 \|\mathcal{X} - \mathcal{M}\|^2 \text{ subject to } \mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$$

Hitchcock 1927, Harshman 1970, Carroll & Chang 1970

CP-OPT: Fitting CP Model via Direct Optimization



$$f = {}^1\!/_2 \|\mathcal{X} - \mathcal{M}\|^2 = {}^1\!/_2 \|\mathbf{X}_{(k)} - \mathbf{M}_{(k)}\|_F^2 \qquad \mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$$

$$\frac{\partial f}{\partial \mathbf{A}_k} = -\left(\mathbf{X}_{(k)} - \mathbf{M}_{(k)}\right) \frac{\partial \mathbf{M}_{(k)}}{\partial \mathbf{A}_k}$$

Note connect to ALS!

Recall: $\mathbf{M}_{(k)} = \mathbf{A}_k \left(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1\right)'$

$$\begin{aligned} \frac{\partial f}{\partial \mathbf{A}_k} &= -\left(\mathbf{X}_{(k)} - \mathbf{M}_{(k)}\right)\left(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1\right) \\ &= \underbrace{-\mathbf{X}_{(k)}\left(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1\right)}_{\text{MTTKRP}} + \mathbf{A}_k \mathbf{V}_k \end{aligned}$$

where $\mathbf{V}_k = (\mathbf{A}_1'\mathbf{A}_1) * \cdots * (\mathbf{A}_{k-1}'\mathbf{A}_{k-1}) * (\mathbf{A}_{k+1}'\mathbf{A}_{k+1}) * \cdots * (\mathbf{A}_d'\mathbf{A}_d)$

Acar, Dunlavy, Kolda 2011

CP-OPT

Calculating the function and gradient

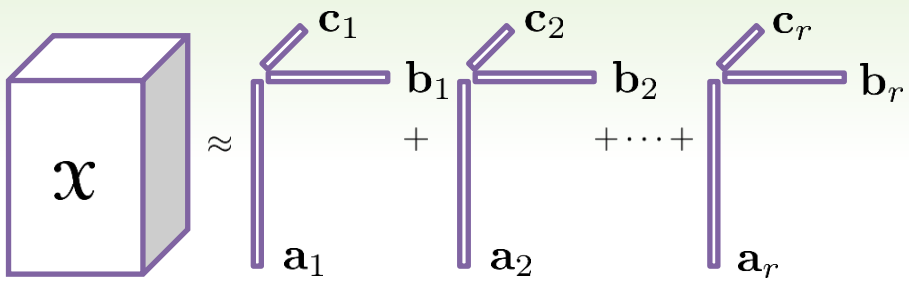
1: **for** $k = 1, 2, \ldots, d$ **do**
2: $\quad \mathbf{Z}_k \leftarrow \mathbf{X}_{(k)}\left(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1\right)$ {MTTKRP}
3: $\quad \mathbf{V}_k \leftarrow (\mathbf{A}_1'\mathbf{A}_1) * \cdots * (\mathbf{A}_{k-1}'\mathbf{A}_{k-1}) * (\mathbf{A}_{k+1}'\mathbf{A}_{k+1}) * \cdots * (\mathbf{A}_d'\mathbf{A}_d)$
4: $\quad \mathbf{\Delta}_k \leftarrow -\mathbf{Z}_k + \mathbf{A}_k\mathbf{V}_k$
5: **end for**
6: $f \leftarrow \|\mathcal{X}\|^2 + \mathbf{1}'\left[\mathbf{A}_d\mathbf{Z}_d\right]\mathbf{1} + \mathbf{1}'\left[\mathbf{A}_d'\mathbf{A}_d * \mathbf{V}_d\right]\mathbf{1}$

- Gradients not naturally in vector form.
- Need to convert before calling opitimization method.

Scaling ambiguities can cause problems – may want to regularize to ameliorate

Acar, Dunlavy, Kolda 2011

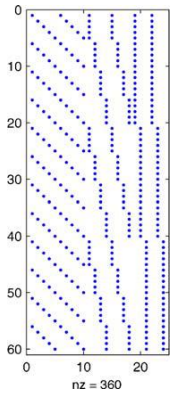
CP-OPT: Fitting CP via "All-at-once" Optimization



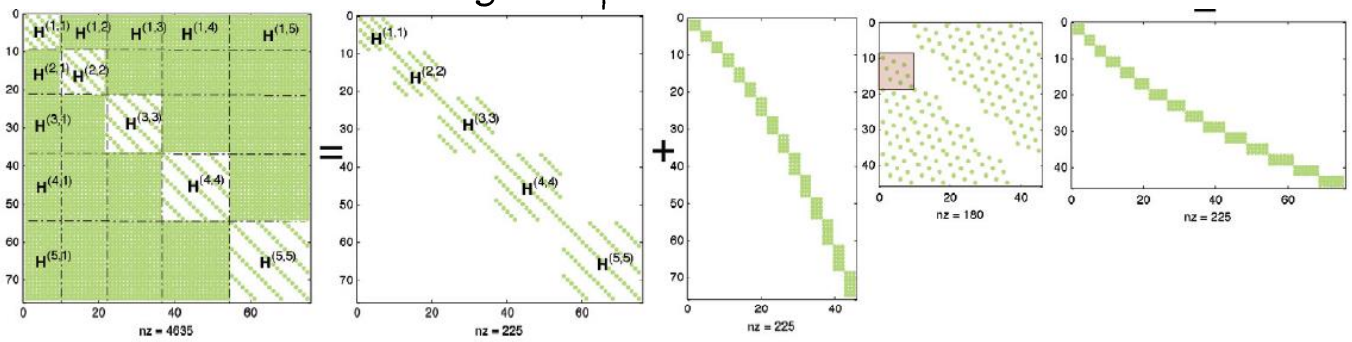
$$f(\mathbf{A},\mathbf{B},\mathbf{C}) = \|\mathcal{X} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|^2$$

- CP-OPT (Acar et al.): 1st-order method, better accuracy than ALS when r is "too big"
- CP-NLS (Paatero, Tomasi & Bro): Damped Gauss-Newton, accurate but slow
- CP-Newton (Phan et al.): Newton method, superior to CP-OPT for high order

Structured Jacobian

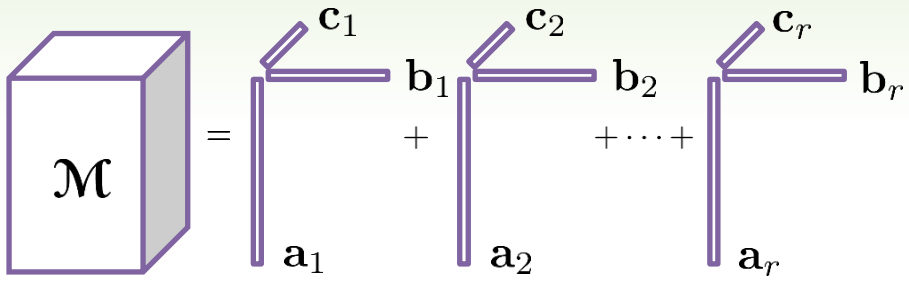


Structured Hessian can be written as block diagonal plus low-rank correction



Paatero 1997; Tomasi & Bro 2005, 2006; Acar, Dunlavy, & Kolda 2011; Phan, Tichavský, & Cichocki 2013

Uniqueness ⇒ Interpretability



k-rank($\mathbf{A}$) = maximum value k such that *any* k columns of $\mathbf{A}$ are linearly independent

Model is **essentially unique** (i.e., up to permutation and scaling) under the condition that the sum of the factor matrix k-rank values is $\geq 2r + d - 1$

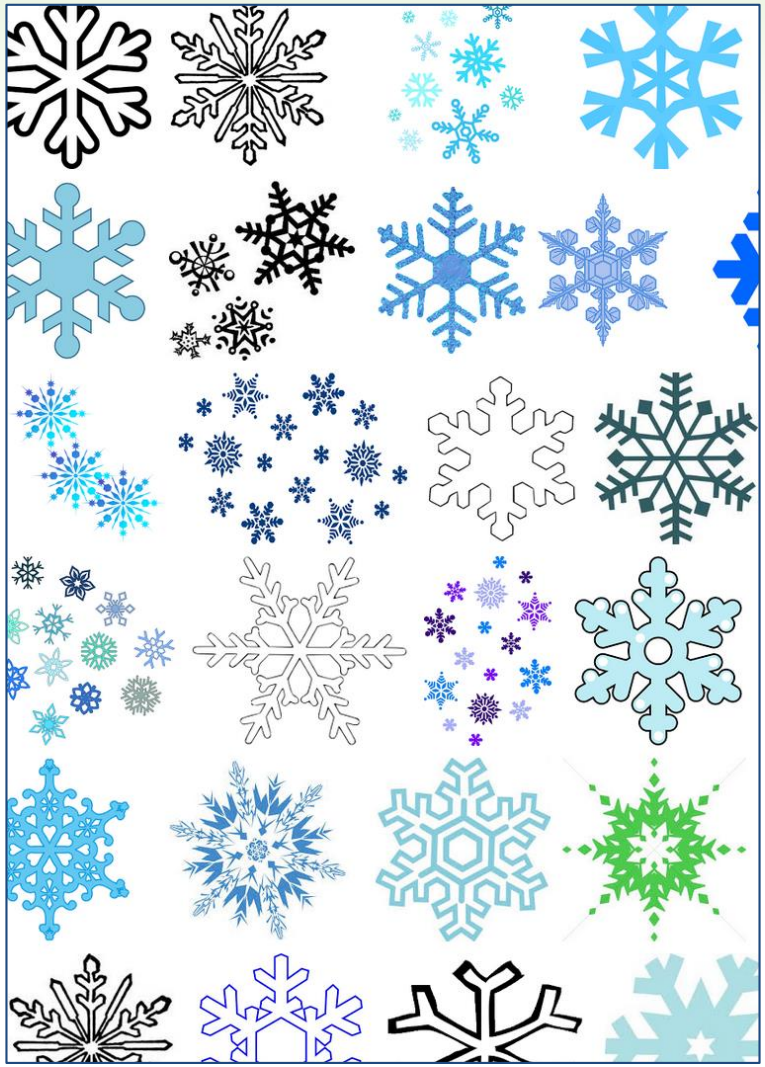
$$\text{k-rank}(\mathbf{A}) + \text{k-rank}(\mathbf{B}) + \text{k-rank}(\mathbf{C}) \geq 2r + 2$$

Matrix factorization does not share this property!

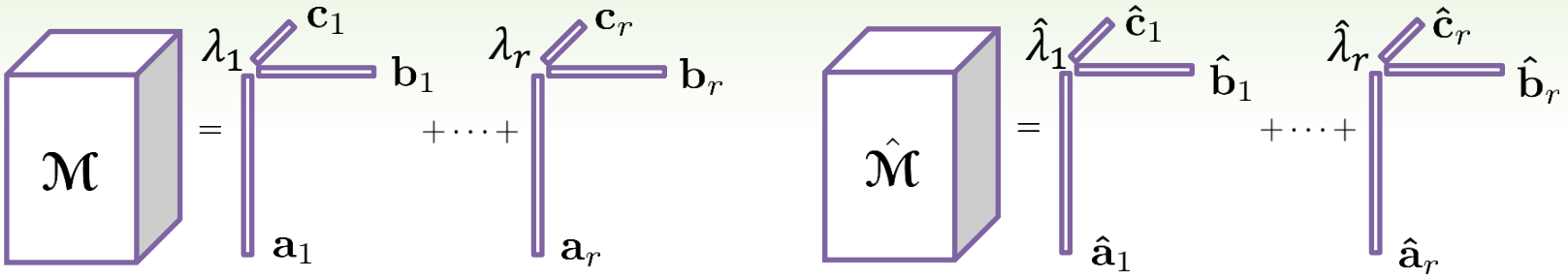
Kruskal 1977, Sidiropoulous & Bro 2000

Commentary on Uniqueness

- Each model may be unique (*just like a snowflake*)
- But there may be lots of models that are close fits
- Results on uniqueness only say that there is no other solution that gives the exact same model estimate
- It does not directly say there aren't other models with equally good or better solutions



Comparing CP Solutions



- Normalize all factors to 2-norm of 1, absorb weights into $\boldsymbol{\lambda}$
- Find the permutation π that maximizes the "score"

$$\text{score} = \frac{1}{r}\sum_{j=1}^{r} \rho_r(\mathbf{a}'_{\pi(j)}\hat{\mathbf{a}}_j)(\mathbf{b}'_{\pi(j)}\hat{\mathbf{b}}_j)(\mathbf{c}'_{\pi(j)}\hat{\mathbf{c}}_j) \quad \rho_j = 1 - \frac{|\lambda_{\pi(j)} - \hat{\lambda}_j|}{\max\{\lambda_{\pi(j)}, \hat{\lambda}_j\}}$$

Cosine of angle between vectors

- Okay for $\mathcal{M}$ to have more components than $\hat{\mathcal{M}}$
- Generally too expensive to try all possible permutations, so we do a greedy matching by default
- Score of 1 indicates perfect match

CP Rank

The (CP) **rank** of a tensor is the minimal value r such that the tensor can be expressed as the sum of exactly r rank-1 tensors and no fewer.

- The ranks of a tensor over $\mathbb{C}$ and $\mathbb{R}$ can be different
- The rank can be larger than the maximum dimension
- The typical ranks over $\mathbb{R}$ for $2 \times 2 \times 2$ are 2 and 3
  - Typical = Positive Probability

$$\mathbf{X}(:,:,1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{X}(:,:,2) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Rank over $\mathbb{R}$ is 3

$$\mathcal{X} = \left[\!\!\left[\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix} \right]\!\!\right]$$

Rank over $\mathbb{C}$ is 2

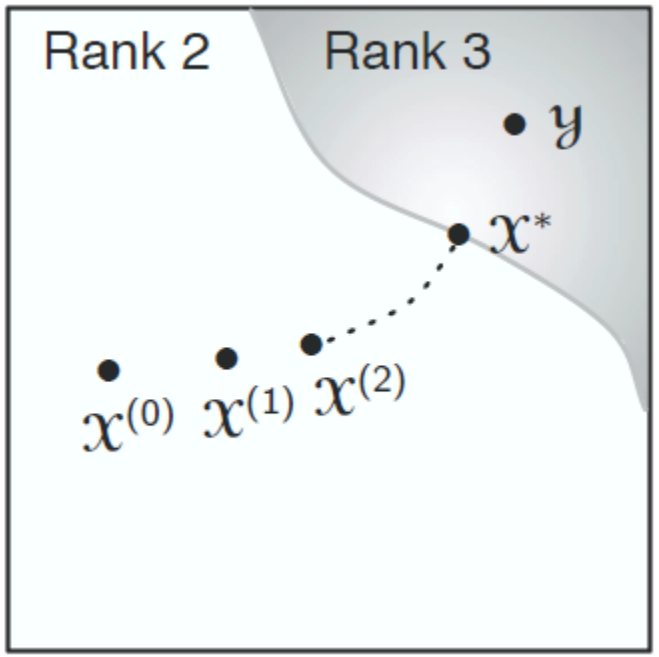
$$\mathcal{X} = \left[\!\!\left[\frac{1}{2}\begin{bmatrix} 1 & 1 \\ -i & i \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ i & -i \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ i & -i \end{bmatrix} \right]\!\!\right]$$

Kruskal 1977, Krusal 1983, Kruskal 1989, Bini et al. 1979; see also Kolda & Bader 2009

CP Rank is NP-Hard

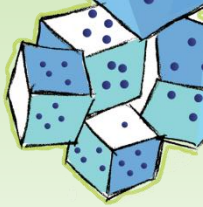
The (CP) **rank** of a tensor is the minimal value r such that the tensor can be expressed as the sum of exactly r rank-1 tensors and no fewer.

- Factorizations are **not nested** (Kolda 2001)
- The space of rank-r tensors is **not closed** except for $r = 1$ (Kruskal, Harshman, & Lundy 1989)
- The best rank-k approximation may **not exist** (de Silva & Lim 2006)
- Determining the rank of a tensor is **NP hard** (Håstad 1990, Hillar & Lim 2009)



Kolda & Bader 2009

Tensor of Unknown Rank



- Specific 9 x 9 x 9 tensor factorization problem
- Corresponds to being able to do fast matrix multiplication of two 3x3 matrices
- Rank is between 19 and 23 ⟶ at most 621 variables

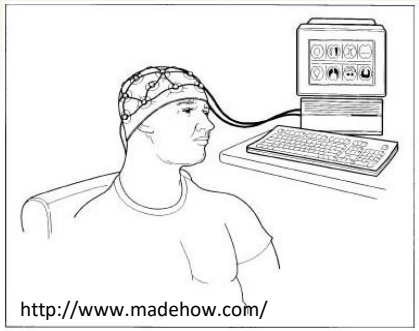
$$
\begin{aligned}
x_{1,1,1} &= 1 & x_{4,2,1} &= 1 & x_{7,3,1} &= 1 \\
x_{1,4,2} &= 1 & x_{4,5,2} &= 1 & x_{7,6,2} &= 1 \\
x_{1,7,3} &= 1 & x_{4,8,3} &= 1 & x_{7,9,3} &= 1 \\
x_{2,1,4} &= 1 & x_{5,2,4} &= 1 & x_{8,3,4} &= 1 \\
x_{2,4,5} &= 1 & x_{5,5,5} &= 1 & x_{8,6,5} &= 1 \\
x_{2,7,6} &= 1 & x_{5,8,6} &= 1 & x_{8,9,6} &= 1 \\
x_{3,1,7} &= 1 & x_{6,2,7} &= 1 & x_{9,3,7} &= 1 \\
x_{3,4,8} &= 1 & x_{6,5,8} &= 1 & x_{9,6,8} &= 1 \\
x_{3,7,9} &= 1 & x_{6,8,9} &= 1 & x_{9,9,9} &= 1
\end{aligned}
$$

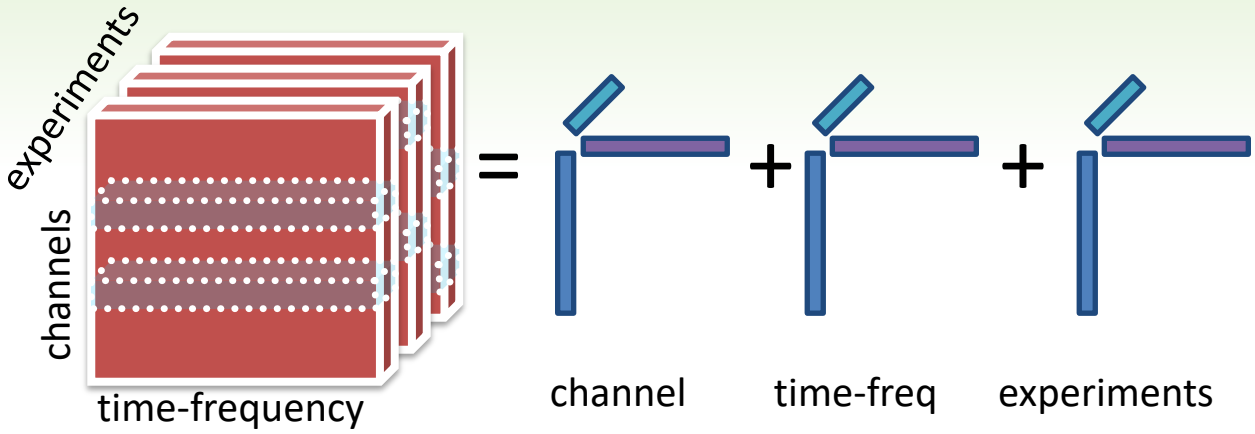
Laderman 1976; Bini et al. 1979; Bläser 2003; Benson & Ballard, PPoPP'15

Missing Data

Joint work with Evrim Acar, Morten Mørup, and Danny Dunlavy

Tensor Factorizations with Missing Data?





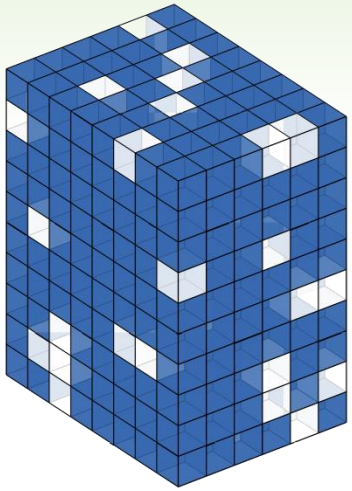
Biomedical signal processing

- EEG (electroencephalogram) signals can be recorded using electrodes placed on the scalp
- **Missing data problem** occurs when…
  - Electrodes get loose or disconnected, causing the signal to be unusable
  - Different experiments have over-lapping but not identical channels

Can we still do this calculation if data are missing?

Acar, Dunlavy, Kolda & Mørup 2010 & 2011

Three Approaches to Missing Data



Ω = *set of known entries (blue)*

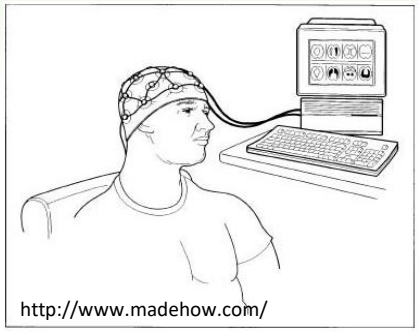
$$\min \sum_{i \in \Omega} (x_i - m_i)^2$$

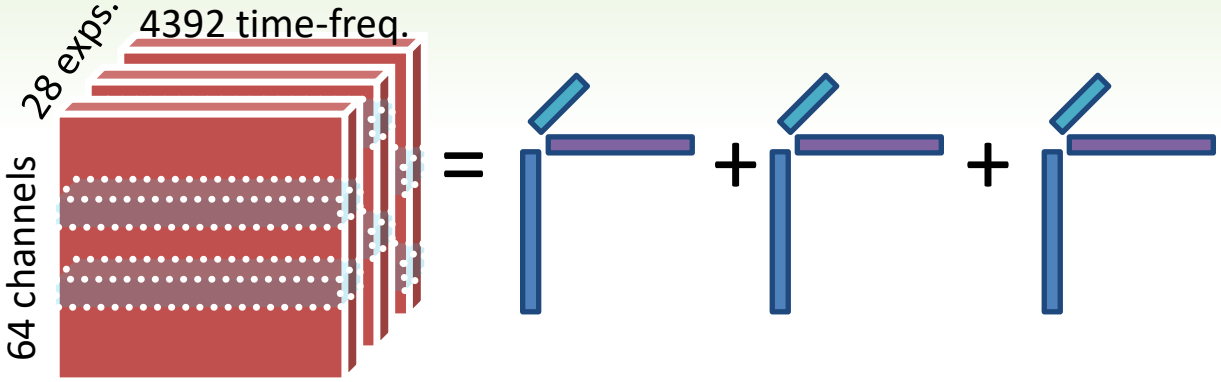
$$\text{s.t. } \mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$$

- Fill in with some mean or interpolated value
- Expectation Maximization
  1. Generate an initial model
  2. Estimate missing entries using current model
  3. Update model using update data tensor
  4. Go back to step #2
- Ignore missing data by reformulating objective function
  - See left
- Closely related to ideas in matrix completion…
- Except we haven't discussed about picking r

Acar, Dunlavy, Kolda & Mørup 2010 & 2011

Brain dynamics can be captured even extensive missing channels

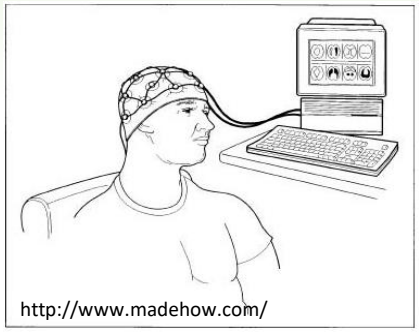


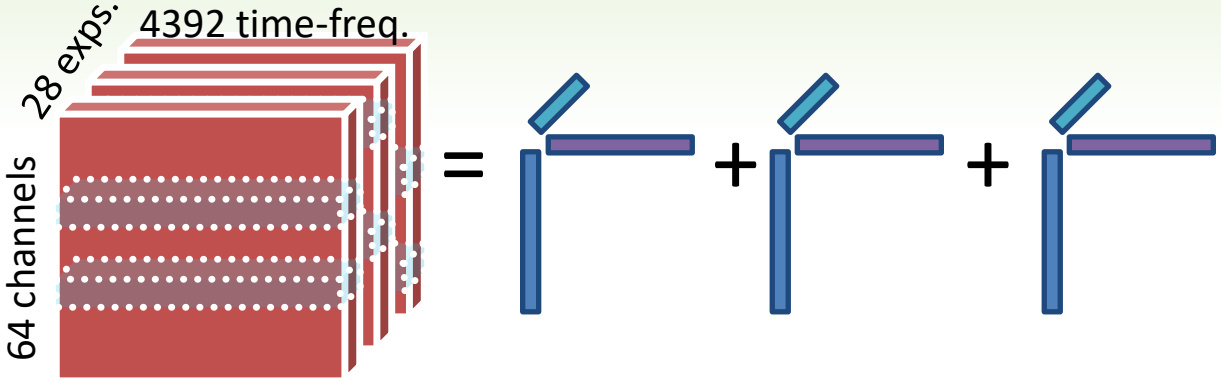


Number of Missing Channels	Replace Missing Entries with Mean
1	0.98
10	0.82
20	0.67
30	0.45
40	0.24

Acar, Dunlavy, Kolda & Mørup 2010 & 2011

Brain dynamics can be captured even extensive missing channels

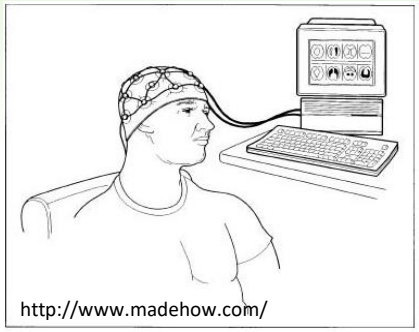


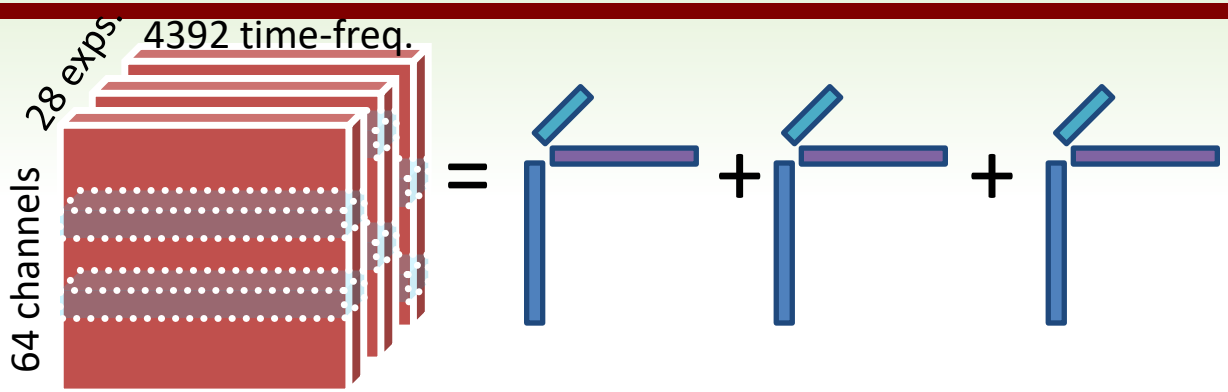


Number of Missing Channels	Replace Missing Entries with Mean	Ignore Missing Entries
1	0.98	1.00
10	0.82	0.98
20	0.67	0.95
30	0.45	0.89
40	0.24	0.65

Acar, Dunlavy, Kolda & Mørup 2010 & 2011

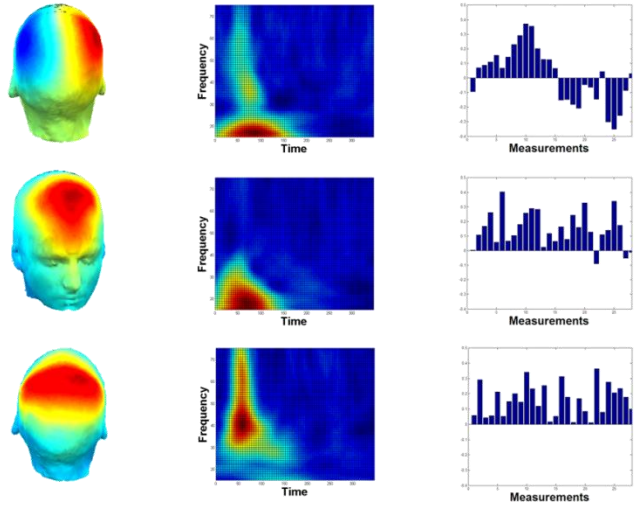
Brain dynamics can be captured even extensive missing channels





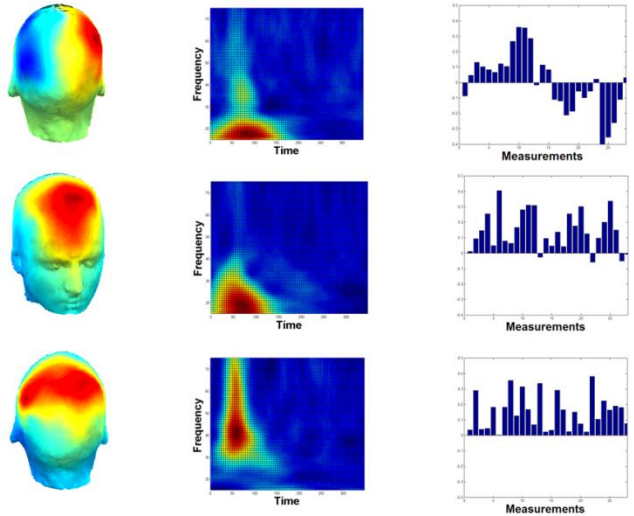
No Missing Data

channel | time-freq | experiments



30 Chan./Exp. Missing

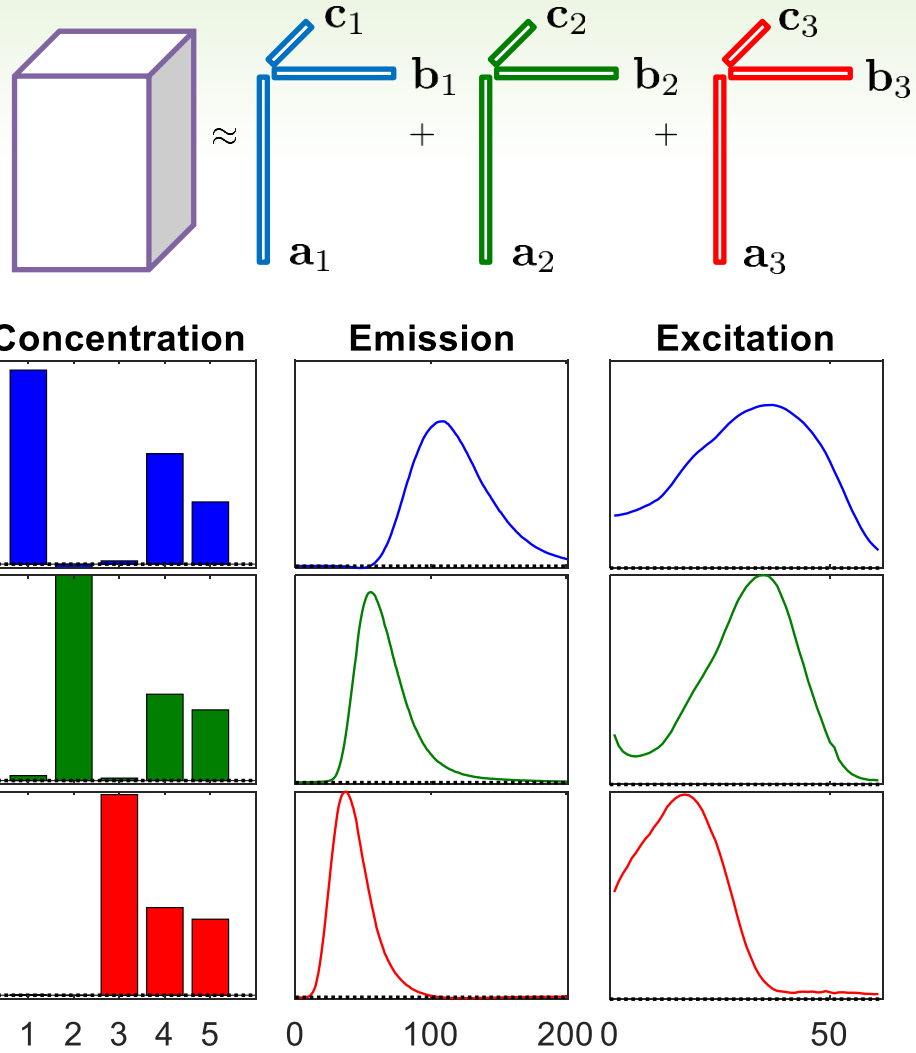
channel | time-freq | experiments



Acar, Dunlavy, Kolda & Mørup 2010 & 2011

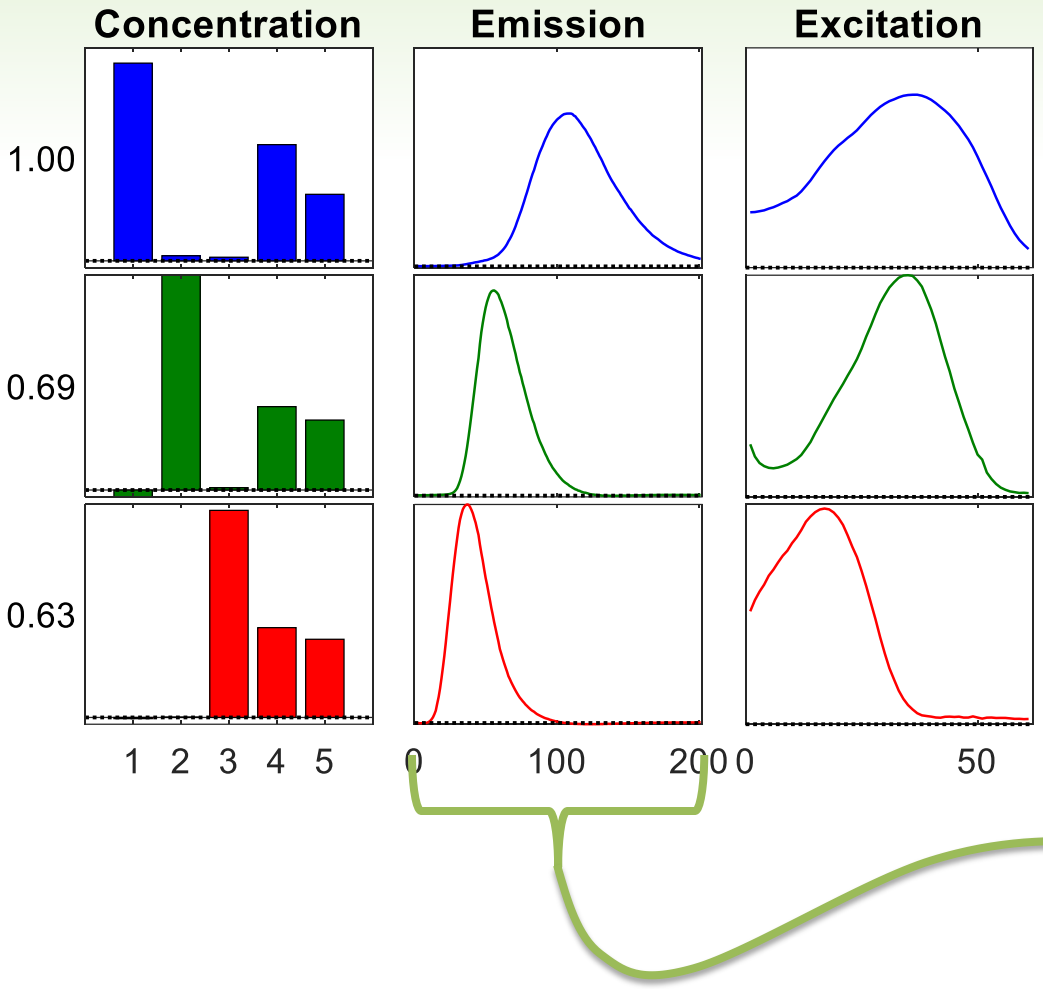
Chemometrics Example (Most-Used CP Example!)

- Fluorescence measurements of samples containing 3 amino acids
  - Tryptophan
  - Tyrosine
  - Phenylalanine
- Tensor of size 5 x 51 x 201
  - 5 samples
  - 51 excitations
  - 201 emissions
- Compute rank-3 CP
- Each amino acid corresponds to a rank-one components

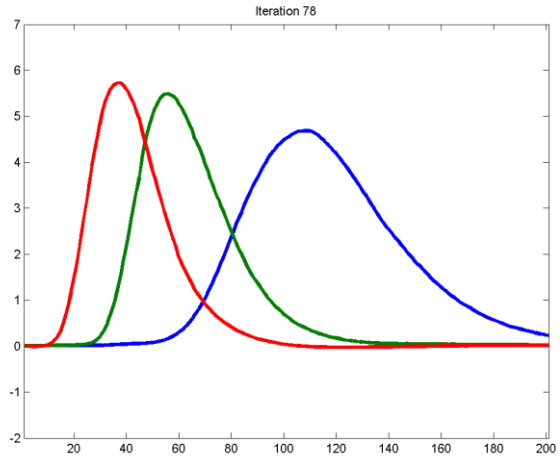


Anderson & Bro 2000

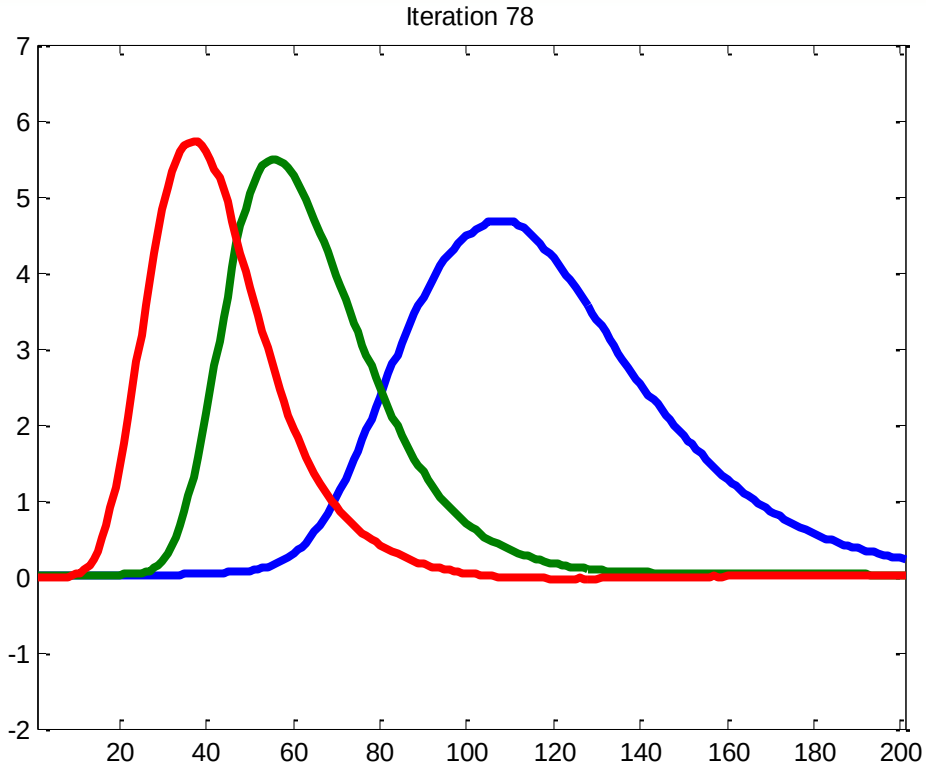
Amino Acid Results Viz



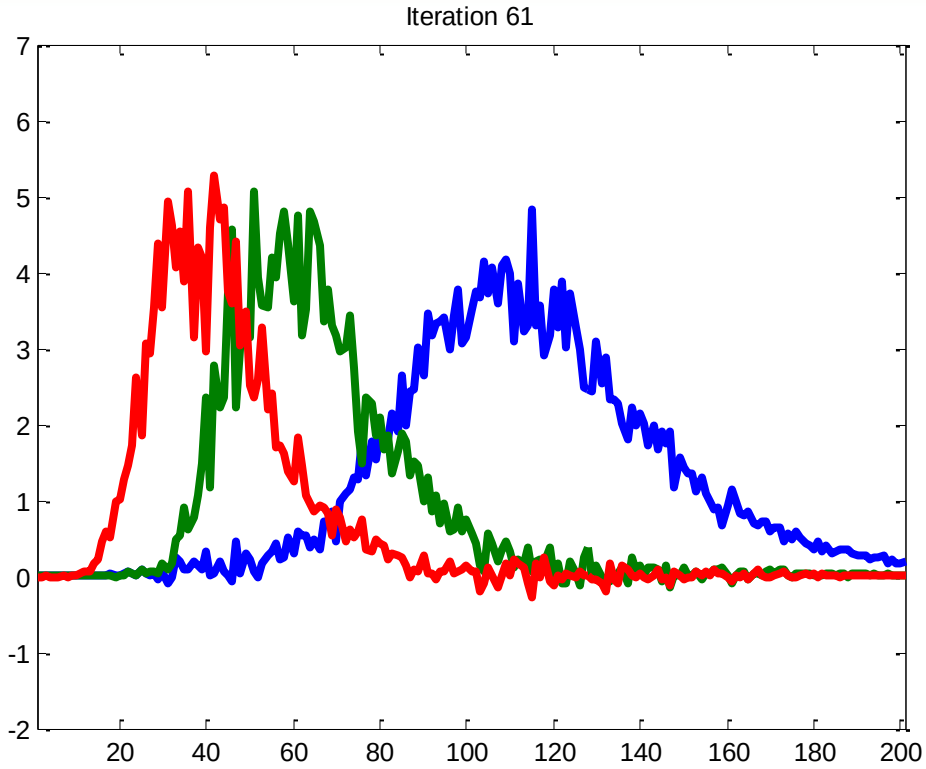
See ktensor/viz in MATLAB



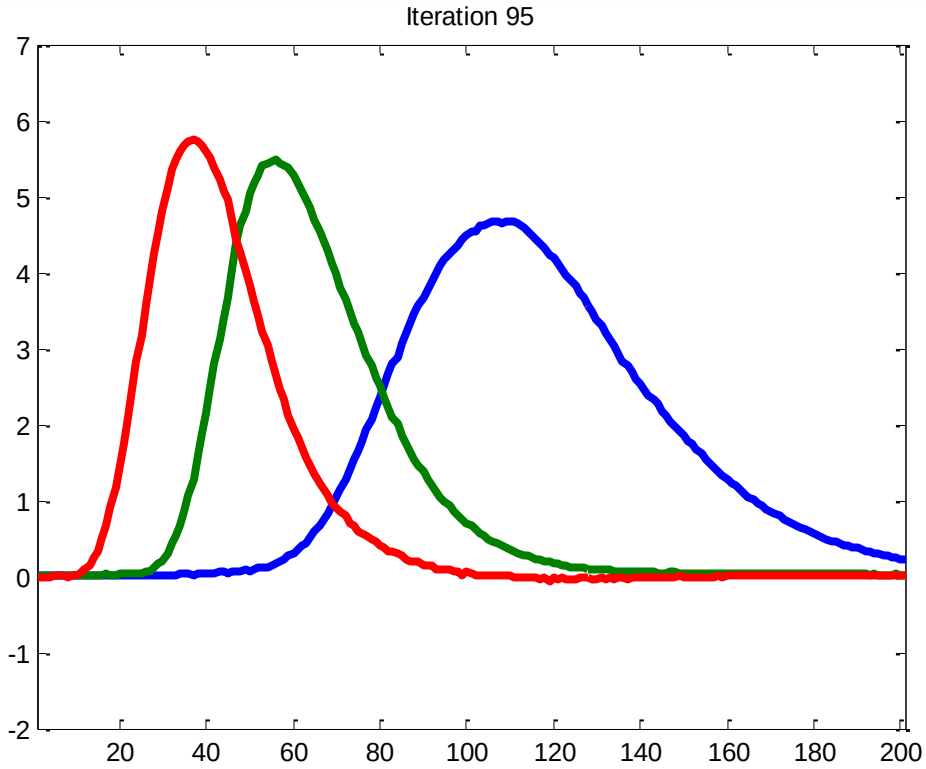
No Missing Data

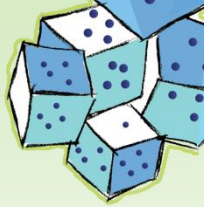


Replacing 50% Missing Values with Mean



75% Missing Data using Reformulation Approach

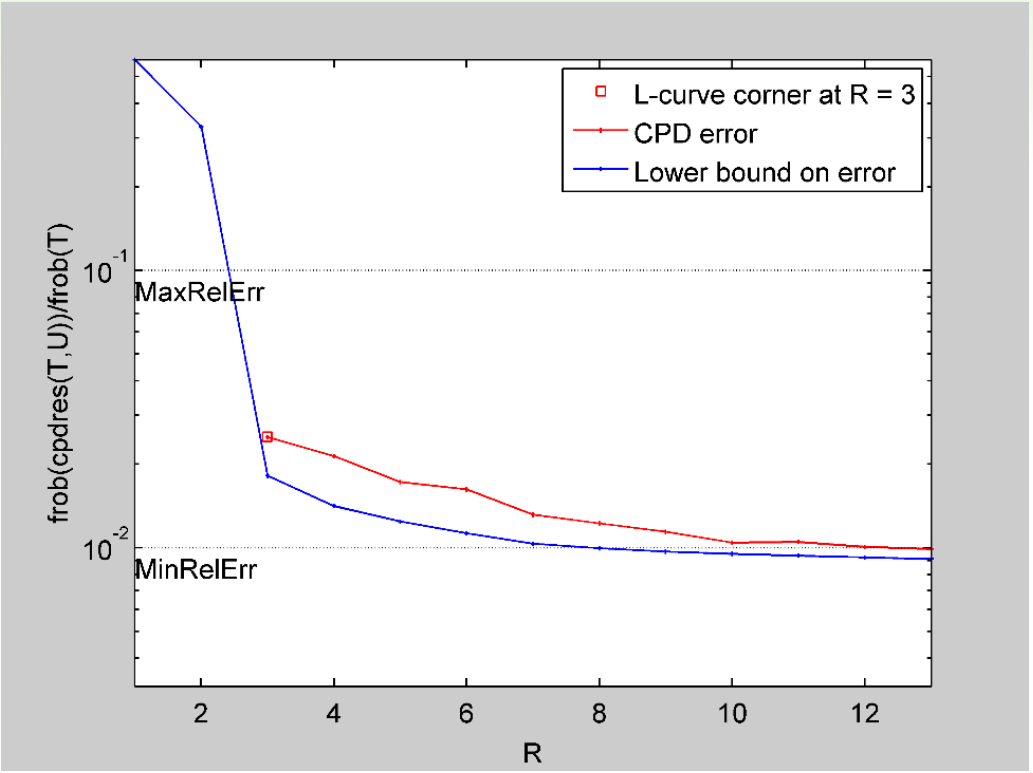




Choosing the number of components (r)

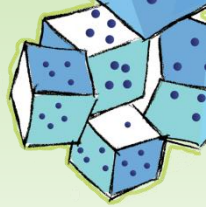
Choosing the number of components

- Fit improves monotonically as rank r increases
  - Because model becomes more complex
  - Assumes we can find global optimum
  - *But cannot always find global optimum*
  - Typically need to do multiple starting points
- Standard approach: Increase rank until fit ceases to improve
- Alternate: Core consistency metric (CORCONDIA)
  - Bro & Kiers 2003
- Cross-validation



http://www.tensorlab.net/doc/cpd.html#choosing-the-number-of-rank-one-terms

Sandia National Laboratories



Cross-Validation to Determine Number of Components

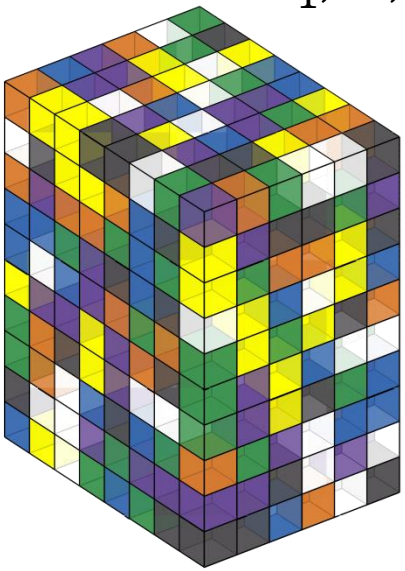
Joint work with Woody Austin (U. Texas, Austin)

Cross-Validation to Determine the Number of Components

Problem: Model error *always* reduces as rank increases, due to more parameters.
Solution: Hide some data from the model, for independent check.

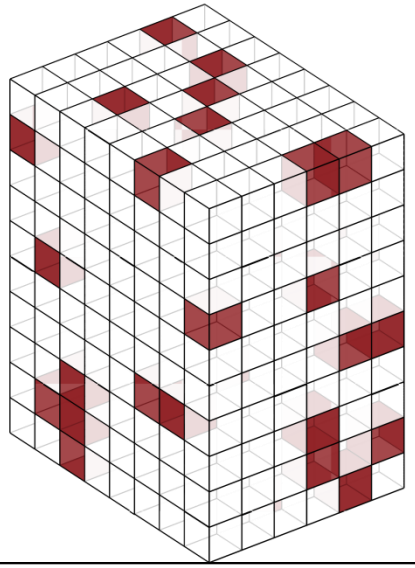
Create H holdout sets: $\Omega_1^c, \ldots, \Omega_H^c$. For each rank r and holdout set h…

Ω_h



Each color corresponds to a holdout set. White is no data.

Ω_h^c



Train model:

$$\mathcal{M}^{(hr)} = \underset{\text{rank}(\mathcal{M})=r}{\arg\min} \sum_{i \in \Omega_h} (x_i - m_i)^2$$

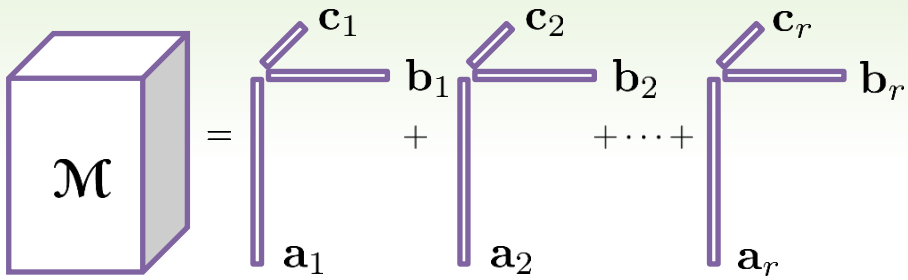
Evaluate model on holdout data:

$$e^{(hr)} = \sqrt{\frac{1}{|\Omega_h^c|} \sum_{i \in \Omega_h^c} (x_i - m_i^{(hr)})^2}$$

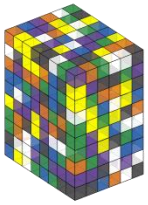
For each rank r, compute average holdout error *(or other statistics)*: $\bar{e}^{(r)} = \frac{1}{H} \sum_h e^{(hr)}$

Joint work with Woody Austin (U. Texas)

Cross-Validation to Determine the Number of Components

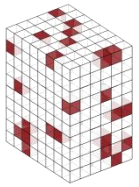


- Create H holdout sets: $\Omega_1^c, \ldots, \Omega_H^c$
- For $r = 1,2, \ldots$
  - Train model for $h = 1, \ldots, H$



$$\mathcal{M}^{(hr)} = \underset{\text{rank}(\mathcal{M})=r}{\arg\min} \sum_{i \in \Omega_h} (x_i - m_i)^2$$

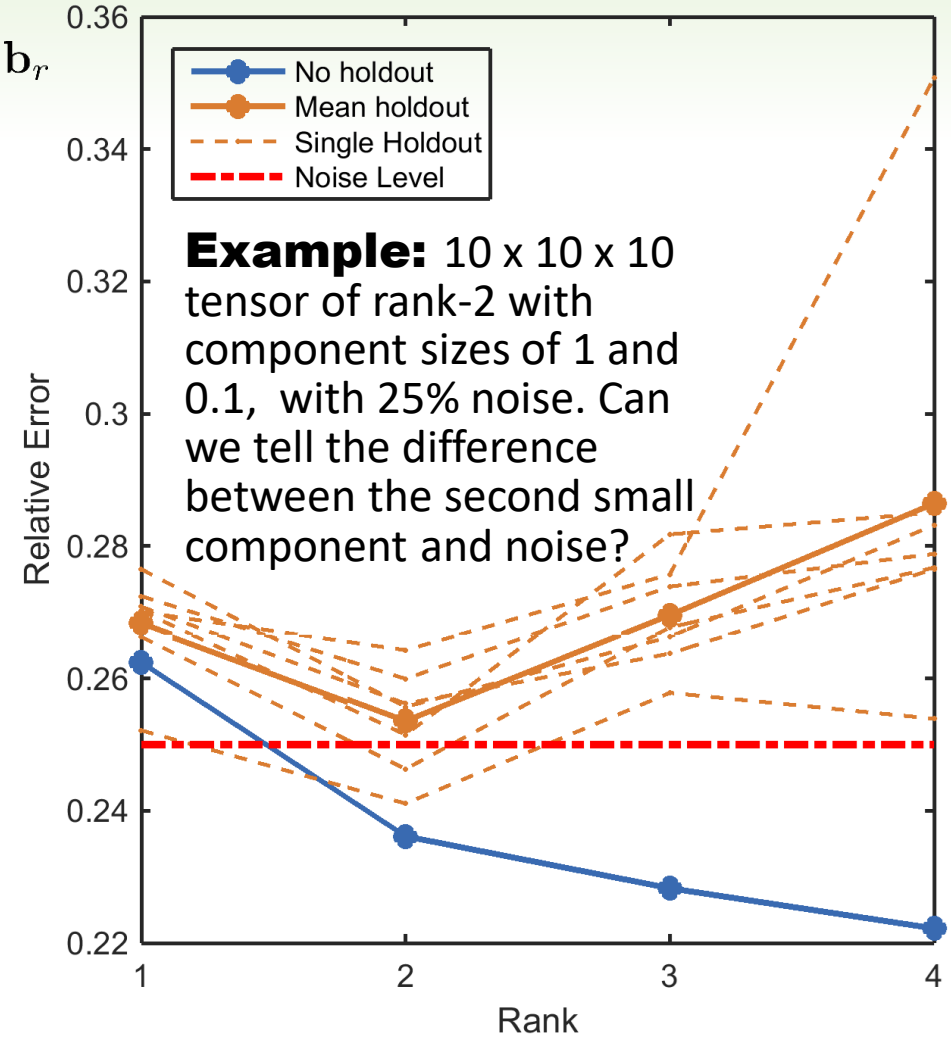
  - Compute error for h =1,…,H



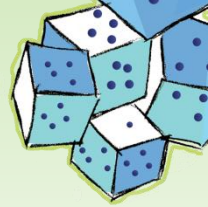
$$e^{(hr)} = \sqrt{\frac{1}{|\Omega_h^c|} \sum_{i \in \Omega_h^c} (x_i - m_i^{(hr)})^2}$$

  - Consider mean error

$$\bar{e}^{(r)} = \frac{1}{H} \sum_h e^{(hr)}$$



Joint work with Woody Austin (U. Texas)

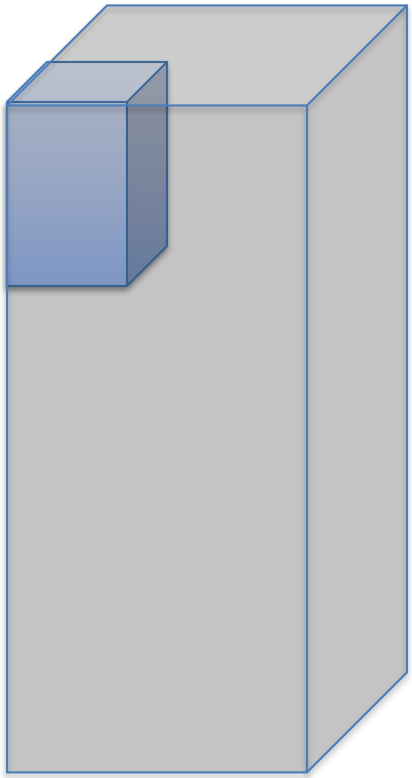


Large-Scale Data

CANDELINC: Project the tensor before computing CP

Sandia National Laboratories

Scenario: Very large tensor with low "effective" dimension in each mode.



$$m_k \equiv \text{rank}\left(\mathbf{X}_{(k)}\right) \ll n_k \text{ for } k = 1, \ldots, d$$

1. Set $\widehat{\mathbf{X}}_{(1)} = \mathbf{X}_{(1)}$
2. For $k = 1, \ldots, d$
   a) Let $\mathbf{U}_k =$ (approximate) orthogonal basis for column space of $\widehat{\mathbf{X}}_{(k)}$
   b) Compute $\widehat{\mathbf{X}}_{(k)} = \mathbf{U}_k' \widehat{\mathbf{X}}_{(k)}$
   c) Reshape result into a new tensor where the size of dimension k is changed from n_k to m_k
3. Compute CP on smaller tensor, $\widehat{\mathbf{X}} = [\![\widehat{\mathbf{A}}_1, \widehat{\mathbf{A}}_2, \ldots, \widehat{\mathbf{A}}_d]\!]$
4. Project resulting factor matrices to larger space, i.e., $\mathbf{A}_k = \widehat{\mathbf{A}}_k$ for $k = 1, \ldots, d$

CANDELINC = Linearly-constrained CANDECOMP

Carroll, Pruzansky, & Kruskal 1980

Sparse Tensors

$$\text{nnz}(\mathcal{X}) \ll n^d$$

- Dense version may be too big to fit in memory!
- Store *only* the nonzeros and their indices
  - $d + 1$ storage per nonzero

MTTKRP

$$\mathbf{B} = \mathbf{X}_{(k)} \left(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1 \right)$$

$$b(i_k, j) = \sum_{i \in \text{nz}(\mathcal{X})} x_i \, a_1(i_1, j) \cdots a_{k-1}(i_{k-1}, j) \, a_{k+1}(i_{k+1}, j) \cdots a_d(i_d, j)$$

Function Evaluation

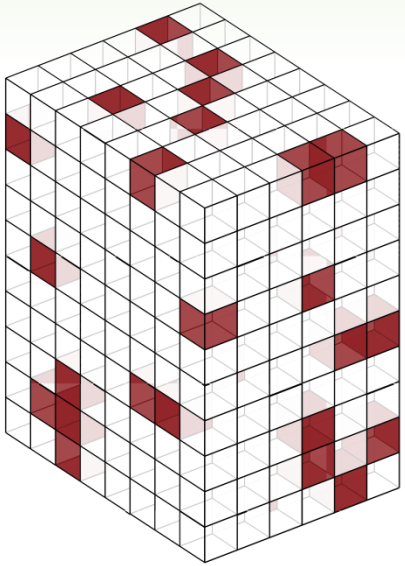
$$\|\mathcal{X} - \mathcal{M}\|^2 = \|\mathcal{X}\|^2 + \|\mathcal{M}\|^2 - 2\langle \mathcal{X}, \mathcal{M} \rangle$$

$$\|\mathcal{X}\|^2 = \sum_{i \in \text{nz}(\mathcal{X})} x_i^2 \qquad \langle \mathcal{X}, \mathcal{M} \rangle = \sum_{i \in \text{nz}(\mathcal{X})} x_i \, a_1(i_1, j) \cdots a_d(i_d, j)$$

$$\langle \mathcal{M}, \mathcal{M} \rangle = \mathbf{1}' \left((\mathbf{A}_1' \mathbf{A}_1) * \cdots * (\mathbf{A}_d' \mathbf{A}_d) \right) \mathbf{1}$$

Bader & Kolda 2007

Purposely Omit Data



- Recall that $n^d \gg dnr$
- In a sense, we have way more data than we need
- Similar to ideas in matrix completion
- Can just leave out a large fraction and still find the solution
  - Acar, Dunlavy, Kolda, & Mørup 2011
- Other ideas center around using only a subset of the data at each step in the iterative solution…

Sandia National Laboratories

Randomized Least Squares for CP Decomposition

Joint work with Casey Battaglino (GA Tech) and Grey Ballard (Wake Forest)

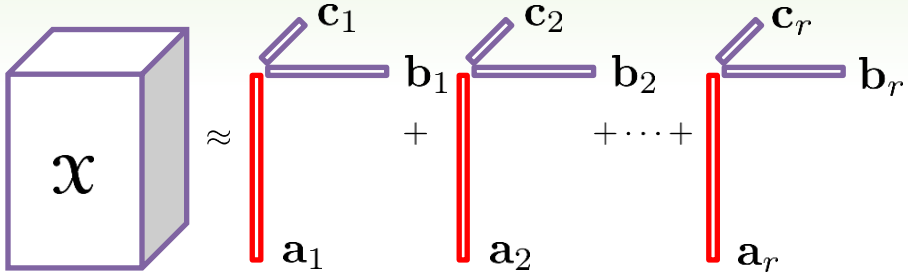
CP-ALS: Fitting CP Model via Alternating Least Squares

Repeat until fit stops changing…

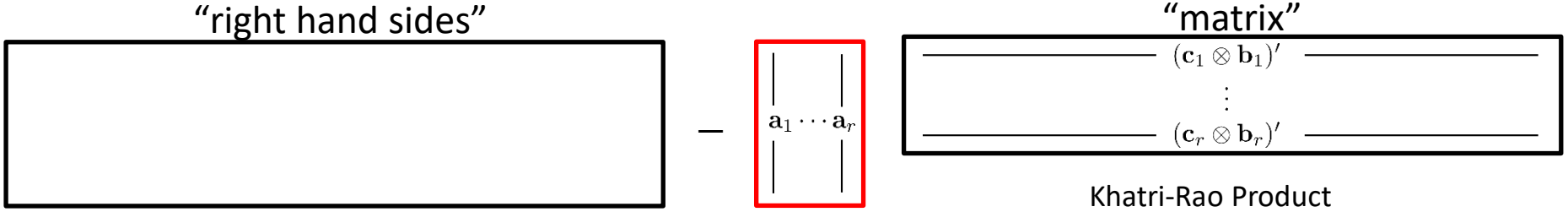
$$\min_{\mathbf{A}} \|\mathbf{X}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})'\|^2$$

$$\min_{\mathbf{B}} \|\mathbf{X}_{(2)} - \mathbf{B}(\mathbf{C} \odot \mathbf{A})'\|^2$$

$$\min_{\mathbf{C}} \|\mathbf{X}_{(3)} - \mathbf{C}(\mathbf{B} \odot \mathbf{A})'\|^2$$



$$\min_{\mathbf{A}} \|\mathbf{X}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})'\|^2$$



$$\|\mathbf{X}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})'\|_F^2$$

$n_k \times n^d/n_k$ $\quad$ $n_k \times r$ $\quad$ $r \times n^d/n_k$

Battaglino, Ballard, & Kolda 2017

Recall: Special Structure of the Least Squares Problem

$$\min_{\mathbf{A}} \|\underbrace{\mathbf{X}_{(1)}}_{n_1 \times (n_2 n_3)} - \underbrace{\mathbf{A}}_{n_1 \times r}\underbrace{(\mathbf{C} \odot \mathbf{B})'}_{r \times (n_2 n_3)}\|^2$$

$$(\mathbf{C} \odot \mathbf{B})\mathbf{A}' = \mathbf{X}'_{(1)}$$

$$\mathbf{A}' = (\mathbf{C} \odot \mathbf{B})^{\dagger}\mathbf{X}'_{(1)}$$

$$\mathbf{A}' = (\mathbf{C}'\mathbf{C} * \mathbf{B}'\mathbf{B})^{\dagger}(\mathbf{C} \odot \mathbf{B})'\mathbf{X}'_{(1)}$$

$$\mathbf{A} = \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{C}'\mathbf{C} * \mathbf{B}'\mathbf{B})^{\dagger}$$

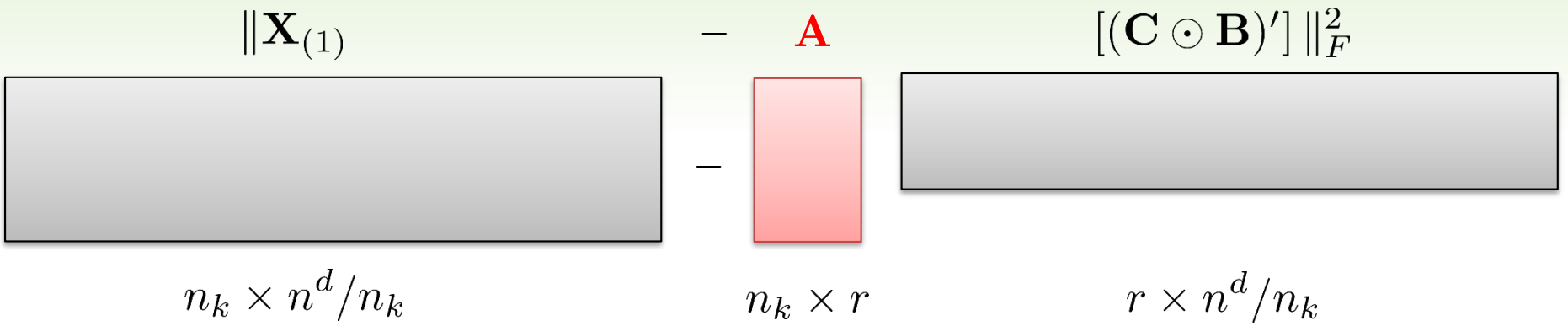
The most expensive step is *not* the backsolve.

Rather, it's the formation of the Khatri-Rao product!

So, how will randomized methods help?

Battaglino, Ballard, & Kolda 2017

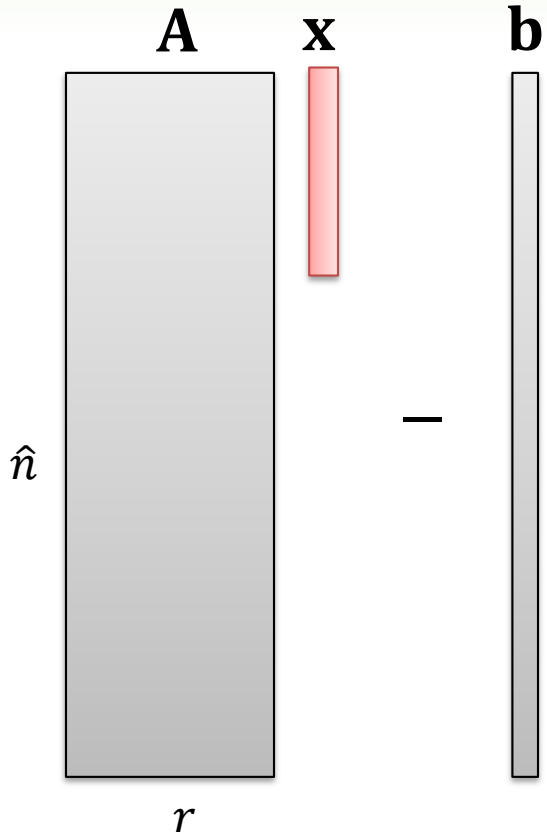
CP Least Squares Problem



How to randomize this?

Aside: Sketching for Standard Least Squares

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|$$



$$\mathcal{O}(\hat{n}r^2)$$

$$\mathbf{x} = \mathbf{A}^{\dagger}\mathbf{b}$$

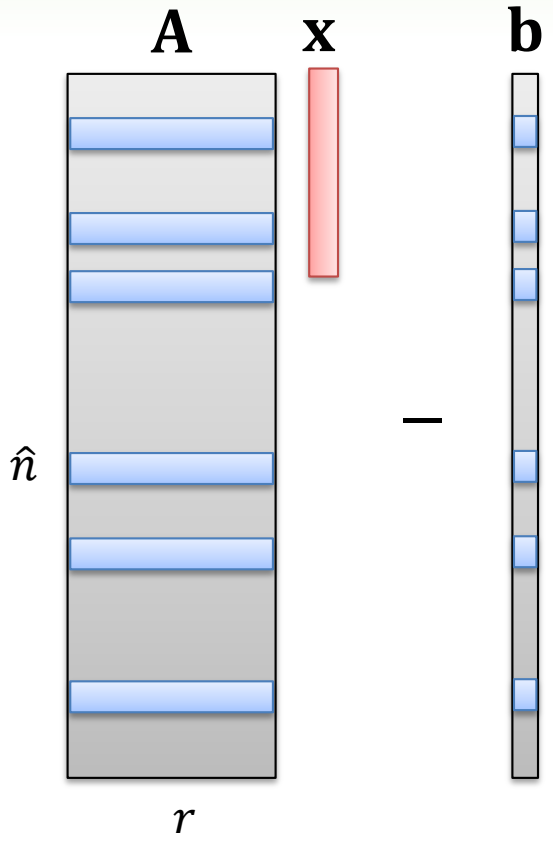
$$\mathbf{x} \leftarrow \mathbf{A} \backslash \mathbf{b}$$

Backslash causes MATLAB to automatically call the best solver (cholesky, qr, etc.)

Sarlós 2006, Woodruff 2014

Sampled Least Squares

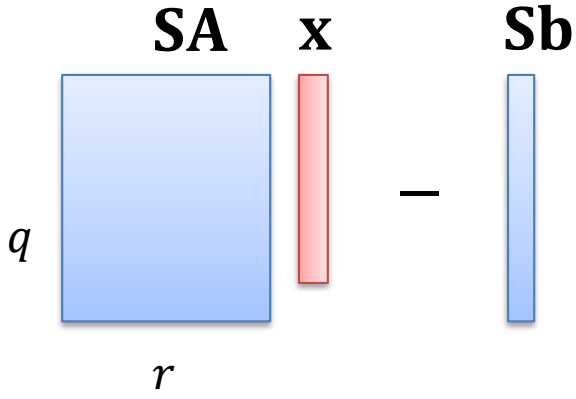
Choose q rows, uniformly at random



$\mathcal{O}(\hat{n}r^2)$

S

approximate

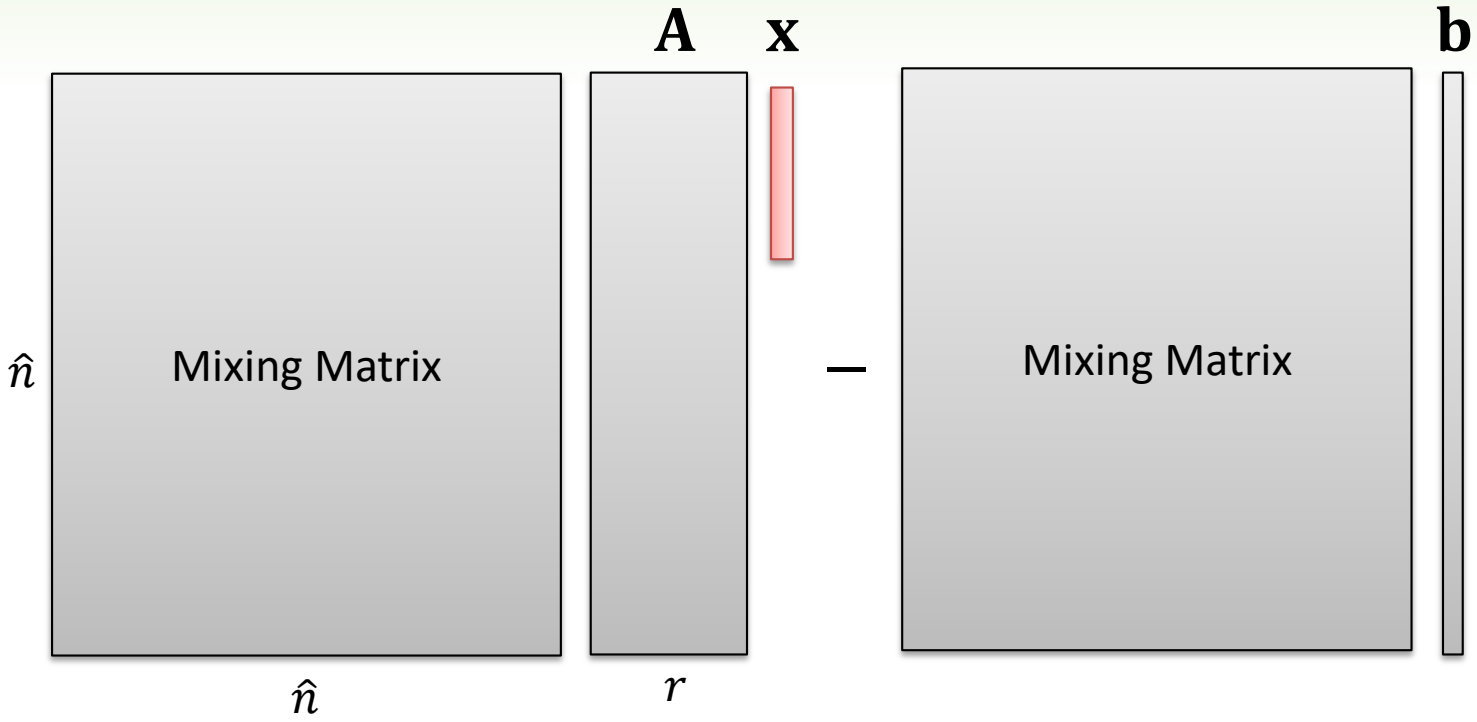


Sampling only guaranteed to "work" if the **A** *is incoherent.*

$\mathcal{O}(qr^2)$

Sarlós 2006, Woodruff 2014

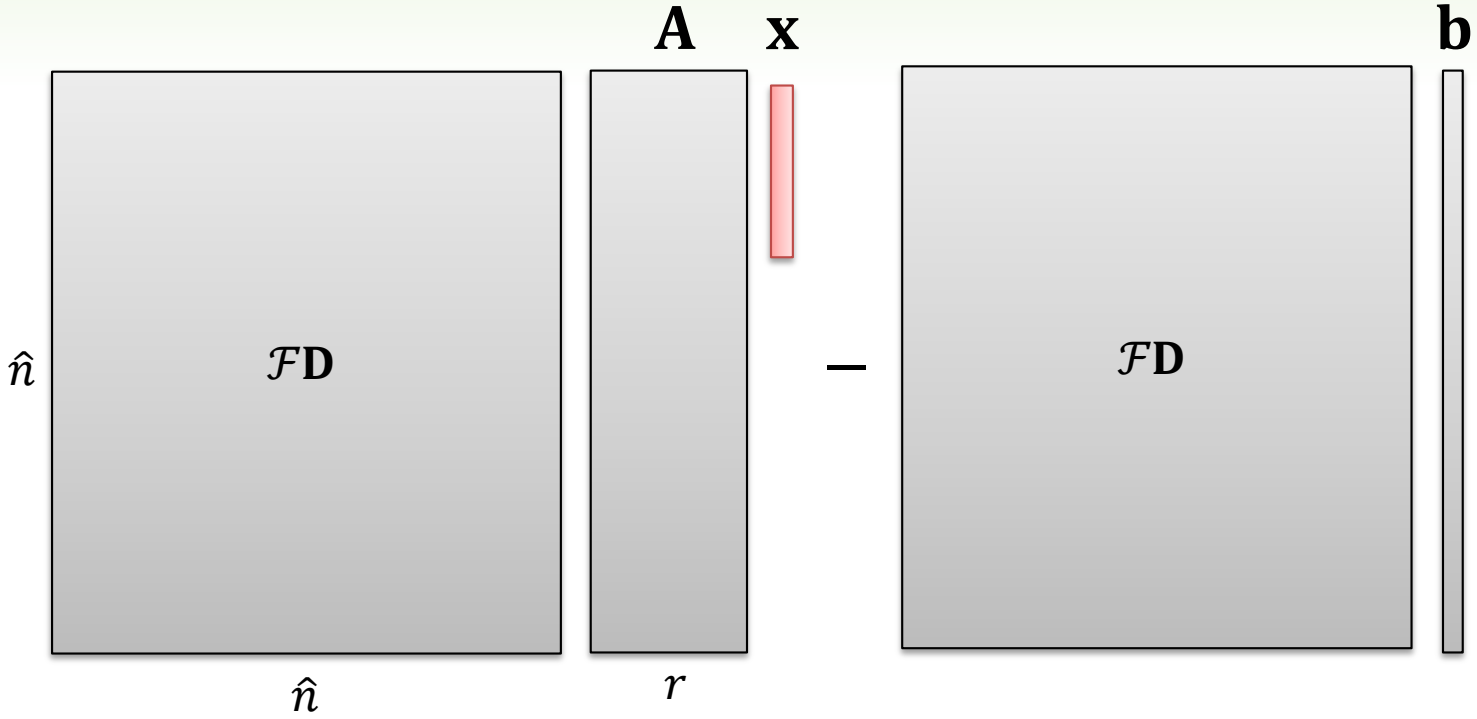
Enforcing Incoherence



- Many good choices of mixing matrix, such as a matrix with entries chosen from a uniform random distribution.
- But no reduction in cost!

Sarlós 2006, Woodruff 2014

Fast Johnson-Lindenstrauss Transform (FJLT)



- Instead, use FFT ($\mathcal{F}$) followed by random diagonal with +/- 1 entries ($\mathbf{D}$).
- Costs only $r \log \hat{n}$ to apply
- Practical application in Blendenpik, yielding ~4X speedup versus LAPACK

Sarlós 2006, Woodruff 2014, Ailon & Chazelle 2006, Avron, Maymounkov, & Toledo 2010

Sampled/Mixed Least Squares

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|$$

$$\min_{\mathbf{x}} \|\mathbf{SAx} - \mathbf{Sb}\|$$

Sampling only, No mixing.

$$\min_{\mathbf{x}} \|\mathbf{S}\mathcal{F}\mathbf{DAx} - \mathbf{S}\mathcal{F}\mathbf{Db}\|$$

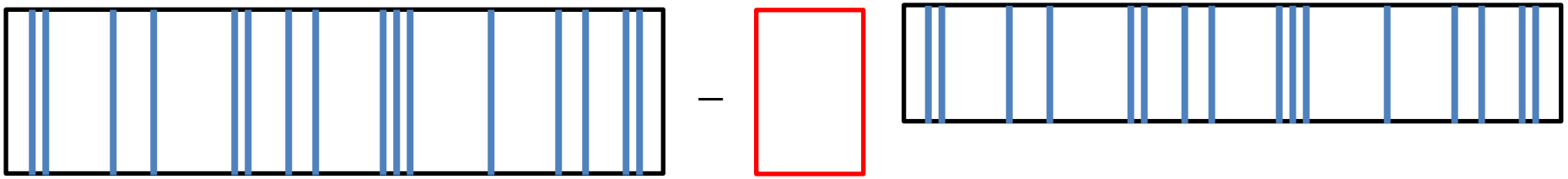
Sampling + Mixing

Ailon & Chazelle 2006; Avron, Maymounkov, & Toledo 2010

CP-ALS-RAND

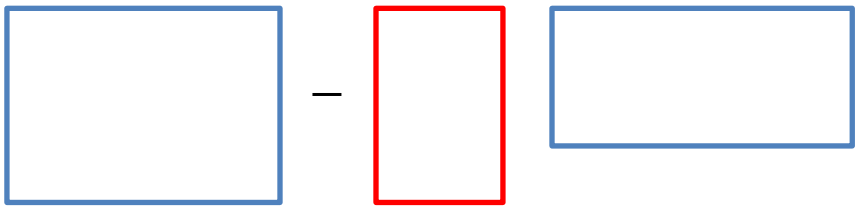
$$\min_{\mathbf{A}} \|\mathbf{X}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})'\|^2$$

$$\mathbf{A} \leftarrow \mathbf{X}_{(1)}/(\mathbf{C} \odot \mathbf{B})'$$



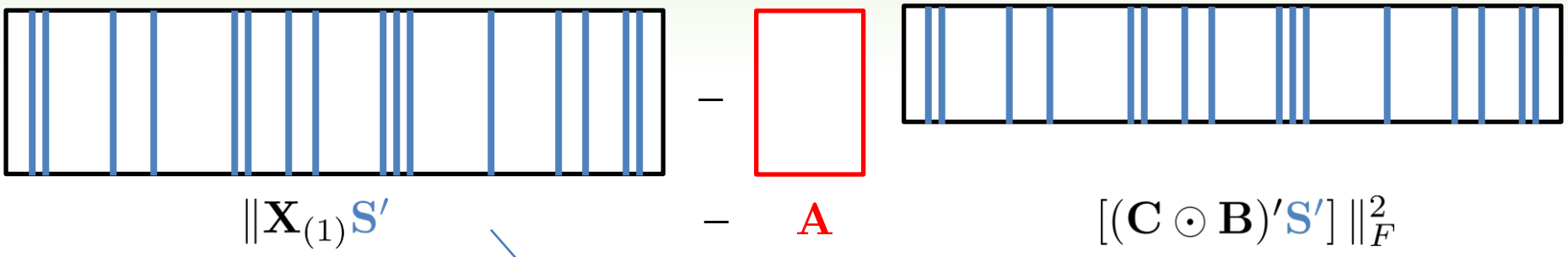
$$\|\mathbf{X}_{(1)}\mathbf{S}' - \mathbf{A}[(\mathbf{C} \odot \mathbf{B})'\mathbf{S}']\|_F^2$$

$$\mathbf{A} \leftarrow \mathbf{X}_{(1)}\mathbf{S}'/(\mathbf{C} \odot \mathbf{B})'\mathbf{S}'$$



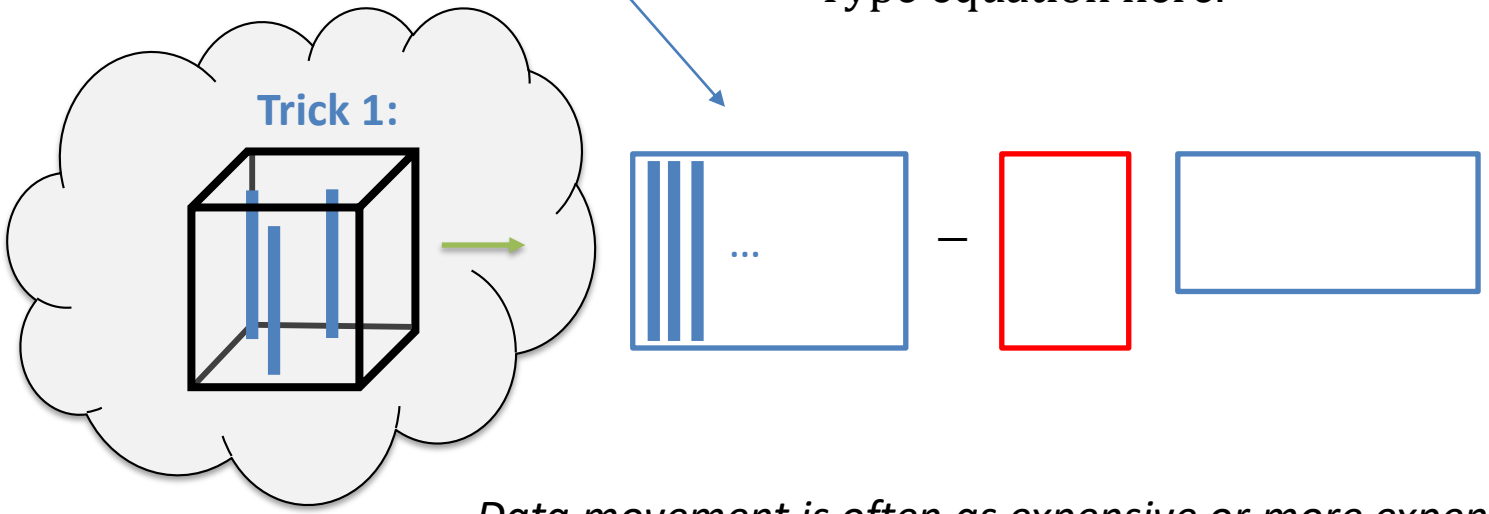
Battaglino, Ballard, & Kolda 2017

CP-ALS-RAND Trick #1: Avoid unfolding data tensor



$$\|\mathbf{X}_{(1)}\mathbf{S}' - \mathbf{A}[(\mathbf{C} \odot \mathbf{B})'\mathbf{S}']\|_F^2$$

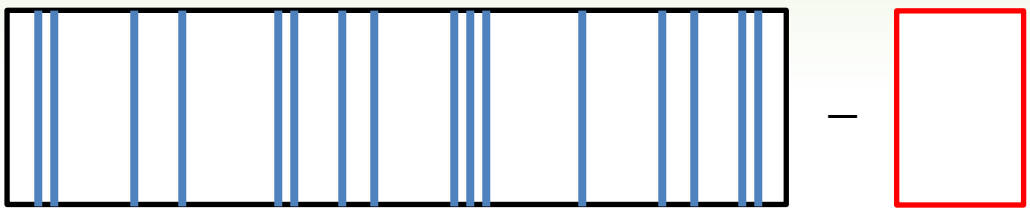
Type equation here.



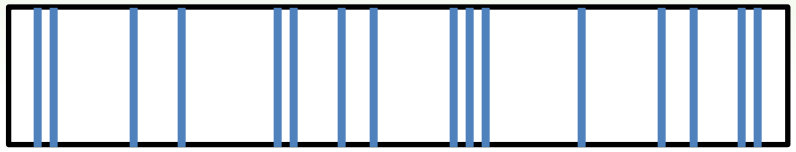
Data movement is often as expensive or more expensive than FLOPS. Just move the minimum and no more.

Battaglino, Ballard, & Kolda 2017

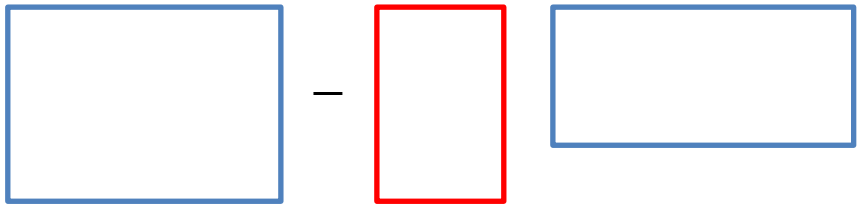
CP-ALS-RAND Trick #2: Don't form Khatri-Rao Product



$$\|\mathbf{X}_{(1)}\mathbf{S}' - \mathbf{A}$$



$$[(\mathbf{C} \odot \mathbf{B})'\mathbf{S}']\|_F^2$$



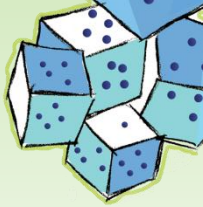
Trick 2:

Each column in the sample is of the form: (C(ℓ,:) .* B(k,:))'

The Khatri-Rao product is actually the most expensive part of CP-ALS. Skip this and save lotsa time.

Battaglino, Ballard, & Kolda 2017

CP-ALS-RAND (No Mixing)



1: **while** not converged **do**
2: $\quad \left[\mathbf{A} \leftarrow \mathbf{X}_{(1)}\mathbf{S}_1{}'\right] / \left[(\mathbf{C} \odot \mathbf{B})'\mathbf{S}_1{}'\right]$
3: $\quad \left[\mathbf{B} \leftarrow \mathbf{X}_{(2)}\mathbf{S}_2{}'\right] / \left[(\mathbf{C} \odot \mathbf{A})'\mathbf{S}_2{}'\right]$
4: $\quad \left[\mathbf{C} \leftarrow \mathbf{X}_{(3)}\mathbf{S}_3{}'\right] / \left[(\mathbf{B} \odot \mathbf{A})'\mathbf{S}_3{}'\right]$
5: **end while**

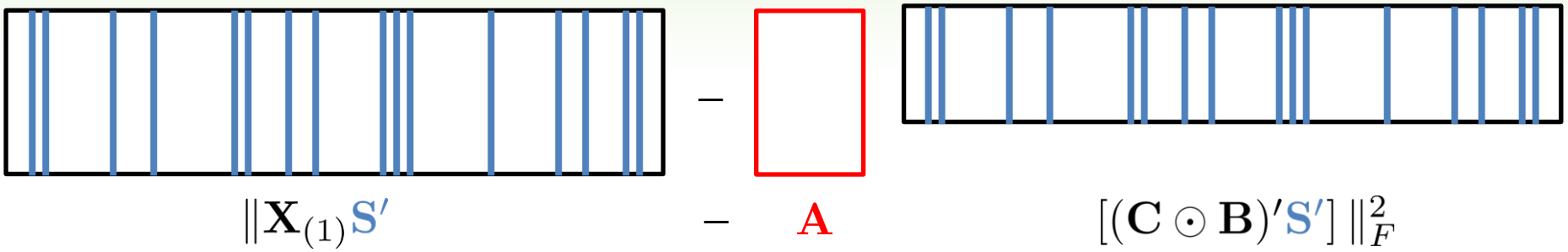
No mixing performed here, yet converges in many cases!

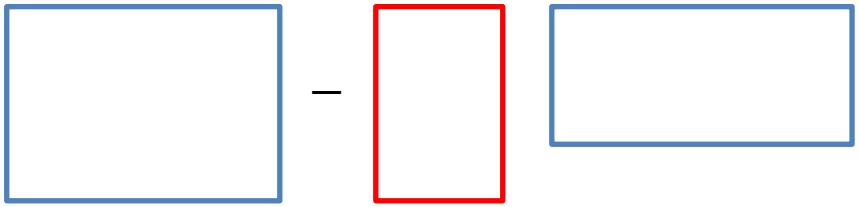
Theorem: Khatri-Rao product does some mixing (see paper).

Nevertheless, some problems still require mixing to converge.

Battaglino, Ballard, & Kolda 2017

CP-ALS-RAND Trick #3: Separately Mix Each Factor





Trick 3:

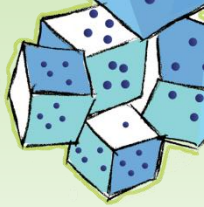
Instead of mixing $(\mathbf{C} \odot \mathbf{B})'$ after it's formed, mix each factor matrix separately:

$$\mathcal{F}_C\mathbf{D}_C\mathbf{C} \odot \mathcal{F}_B\mathbf{D}_B\mathbf{B}$$

Battaglino, Ballard, & Kolda 2017

More Details on Mixing

Sandia National Laboratories



Must mix both sides of the equation…

RHS:
$$\begin{aligned} & (\mathcal{F}_C\mathbf{D}_C\mathbf{C} \odot \mathcal{F}_B\mathbf{D}_B\mathbf{B})' \\ = \; & (\mathcal{F}_C\mathbf{D}_C \otimes \mathcal{F}_B\mathbf{D}_B)(\mathbf{C} \odot \mathbf{B})' \end{aligned}$$

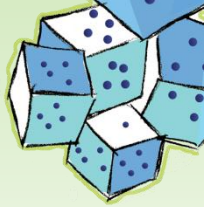
LHS:
$$\begin{aligned} & (\mathcal{F}_C\mathbf{D}_C \otimes \mathcal{F}_B\mathbf{D}_B)\mathbf{X}_{(1)} \\ = \; & \mathcal{X} \times_1 \mathcal{F}_C\mathbf{D}_C \times_2 \mathcal{F}_B\mathbf{D}_B \end{aligned}$$

Applies FFT to each mode unfolding via tensor-times-matrix operation:

$$\mathcal{X} \times_k \mathbf{U} = \mathbf{U}\mathbf{X}_{(k)}$$

Battaglino, Ballard, & Kolda 2017

CP-ALS-RAND Trick #4: Mix Original Tensor Only Once



Trick 4:
Mix original tensor only once

$$\hat{\mathcal{X}} = \mathcal{X} \times_3 \mathcal{F}_C \mathbf{D}_C \times_2 \mathcal{F}_B \mathbf{D}_B \times_1 \mathcal{F}_A \mathbf{D}_A$$

Unfold in mode 1

$$\hat{\mathbf{X}}_{(1)} = \mathbf{D}_A \mathcal{F}_A \mathbf{X}_{(1)} (\mathcal{F}_C \mathbf{D}_C \otimes \mathcal{F}_B \mathbf{D}_B)$$

Sampled on mixed matrix

$$\left[\mathbf{D}_A \mathcal{F}_A \mathbf{X}_{(1)} (\mathcal{F}_C \mathbf{D}_C \otimes \mathcal{F}_B \mathbf{D}_B)\right] \mathbf{S}'$$

Undo mode-1 mixing *after* sampling

$$\mathcal{F}_A^{-1} \mathbf{D}_A \left(\left[\mathbf{D}_A \mathcal{F}_A \mathbf{X}_{(1)} (\mathcal{F}_C \mathbf{D}_C \otimes \mathcal{F}_B \mathbf{D}_B)\right] \mathbf{S}' \right)$$

Battaglino, Ballard, & Kolda 2017

Putting It All Together

Trick 4:
Mix original tensor only once & invert only small result

Trick 3:
Mix factors rather than full product

$$\mathbf{A} \leftarrow \mathcal{F}_A \mathbf{D}_A \left(\left(\hat{\mathbf{X}}_{(1)} \mathbf{S}' \right) / \left((\mathcal{F}_C \mathbf{D}_C \mathbf{C} \odot \mathcal{F}_B \mathcal{D}_B \mathbf{B}) \mathbf{S}' \right) \right)$$

Trick 1:
Sample efficiently from tensor

Trick 2:
Sample without forming Khatri-Rao product

Plus, we have to be careful to make sure we get a real-valued result. See paper for details.

Battaglino, Ballard, & Kolda 2017

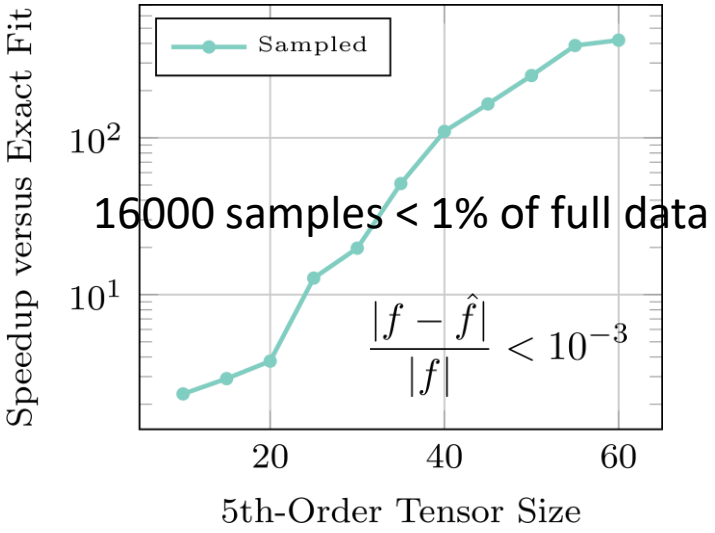
Randomizing the Convergence Check

Repeat until fit stops changing…

$$\min_{\mathbf{A}} \|\mathbf{X}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})'\|^2$$

$$\min_{\mathbf{B}} \|\mathbf{X}_{(2)} - \mathbf{B}(\mathbf{C} \odot \mathbf{A})'\|^2$$

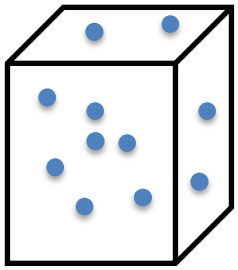
$$\min_{\mathbf{C}} \|\mathbf{X}_{(3)} - \mathbf{C}(\mathbf{B} \odot \mathbf{A})'\|^2$$



16000 samples < 1% of full data

$$f(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \sum_i (x_i - m_i)^2$$

$$\text{s.t. } \mathcal{M} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

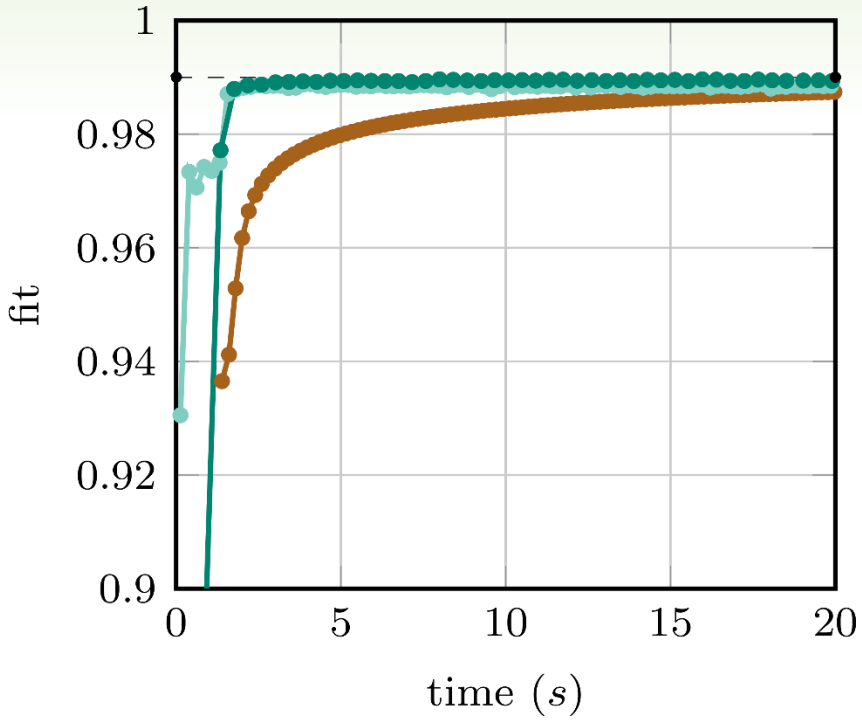


Estimate convergence of function values using small random subset of elements in function evaluation (use Chernoff-Hoeffding to bound accuracy)

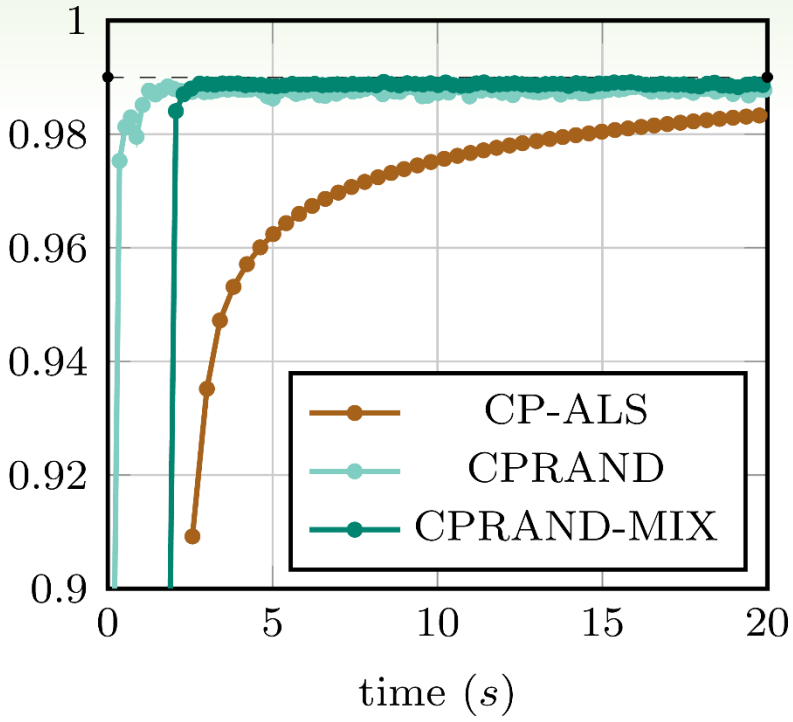
$$f(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \frac{n^d}{|\hat{\Omega}|} \sum_{i \in \hat{\Omega}} (x_i - m_i)^2$$

Battaglino, Ballard, & Kolda 2017

CPRAND faster than CP-ALS



300 x 300 x 300 Random Rank-5 Tensor with 1% Noise



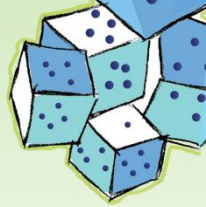
80 x 80 x 80 x 80 Random Rank-5 Tensor with 1% Noise

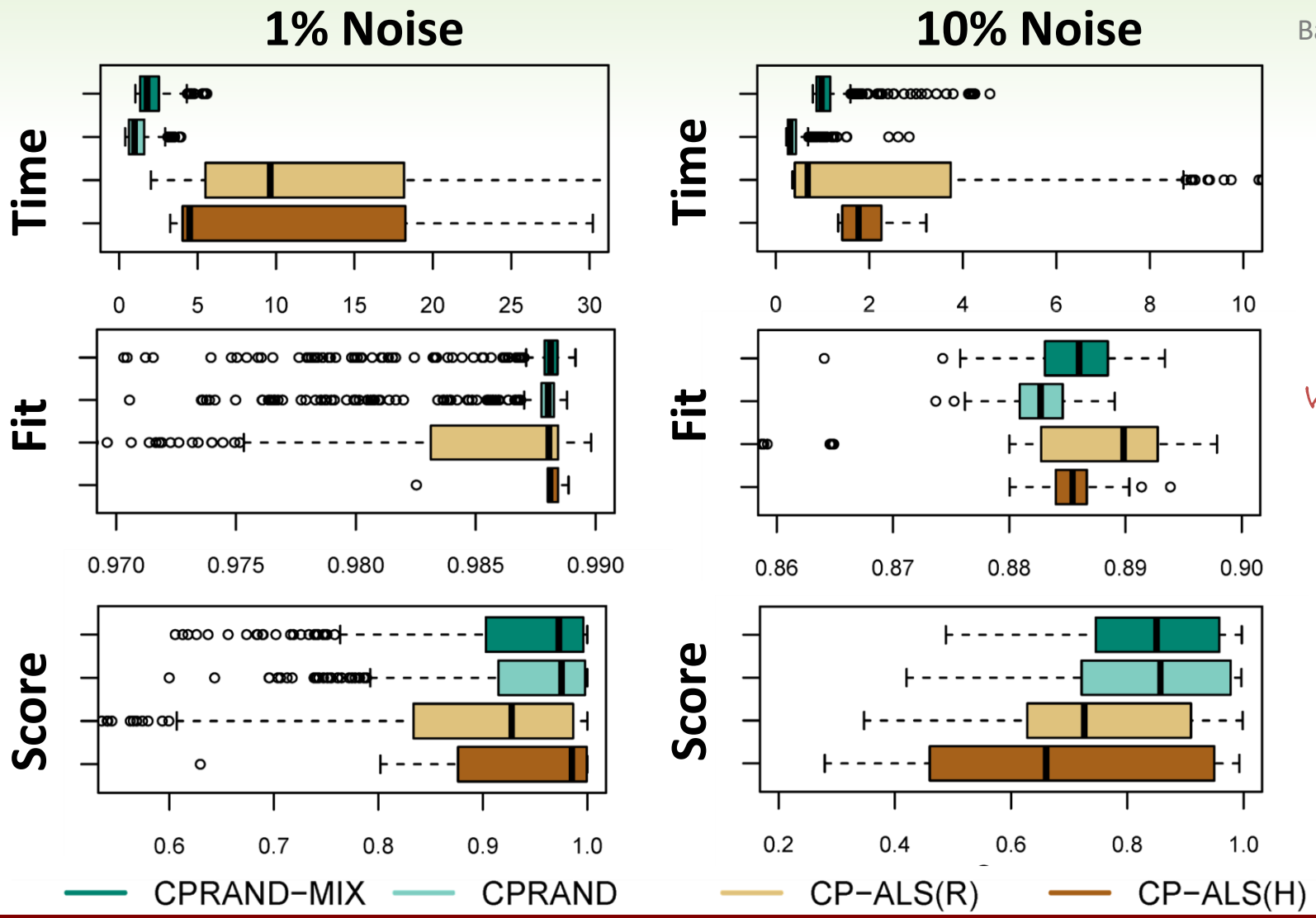
Battaglino, Ballard, & Kolda 2017

CPRAND Faster & More Robust than CP-ALS

200 Tensors: 400 x 400 x 400, r = 5 or 6

Sandia National Laboratories





Battaglino, Ballard, & Kolda 2017

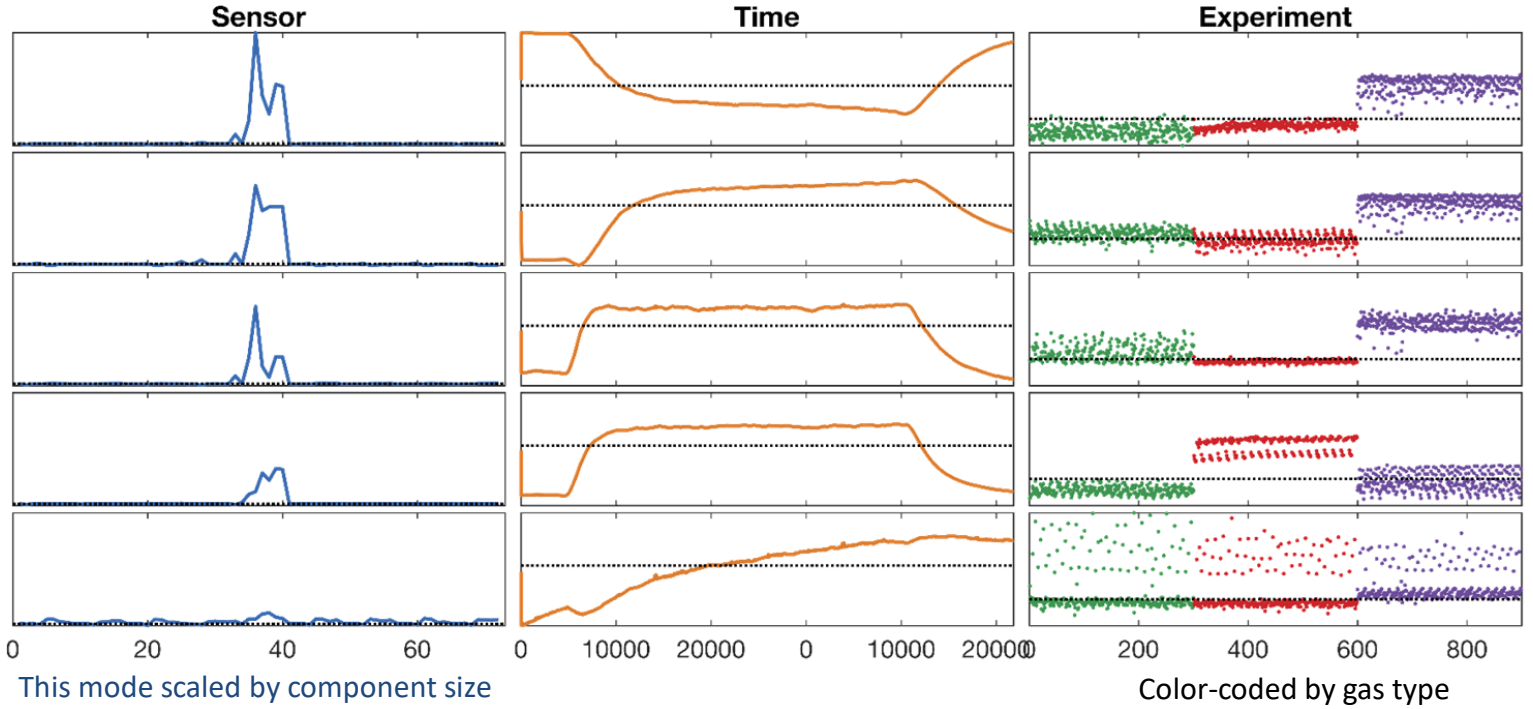
What we can measure

What we really want!

Analysis of Hazardous Gas Data

900 experiments (with three different gas types) x 72 sensors x 25,900 time steps (13 GB)

Method	Median Time (s)	Median Fit	Median Classification Error
CPRAND	53.6	0.715	0.61%
CP-ALS (H)	578.4	0.724	0.67%
CP-ALS (R)	204.7	0.724	0.67%

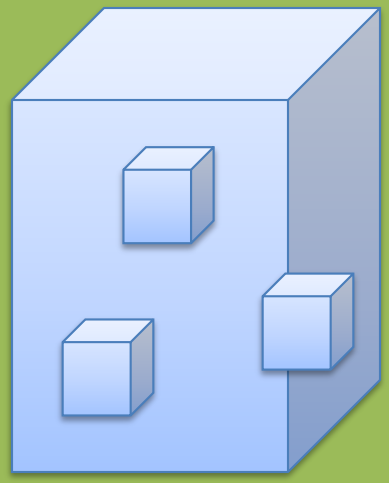


Data from Vergara et al. 2013; see also Vervliet and De Lathauwer (2016)

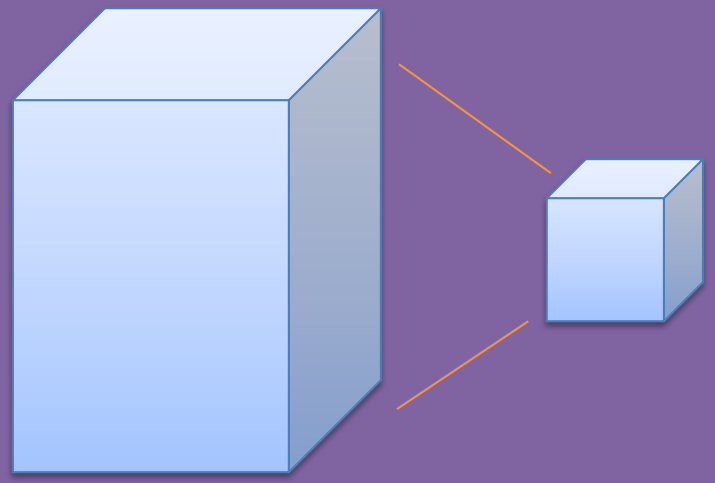
Battaglino, Ballard, & Kolda 2017

Other Randomized Methods

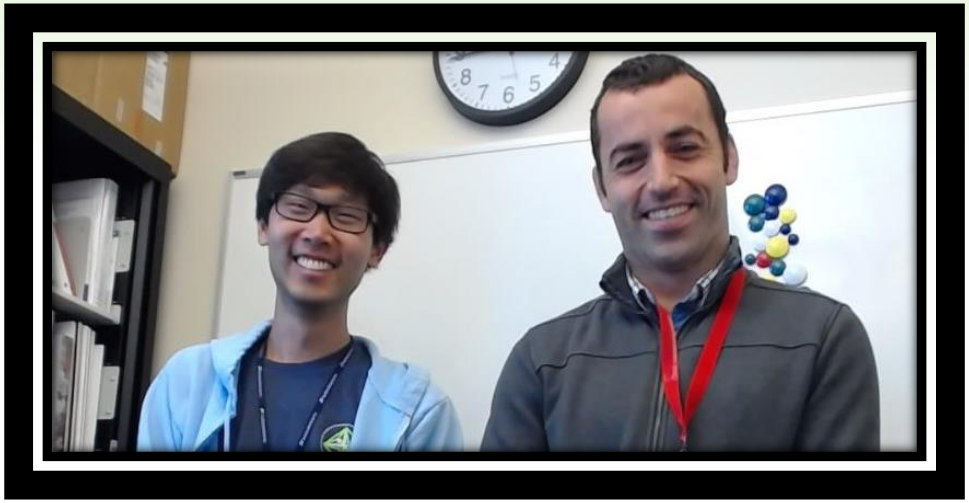
- Vervliet and De Lathauwer (2016) – Select a random d-way sub-block of data rather than random elements
  - Different sub-block at every iteration
  - Not contiguous as pictured



- Zhou, Cichocki, and Xie (2014) – Random projections (sketch) to smaller tensor
  - Project once and done



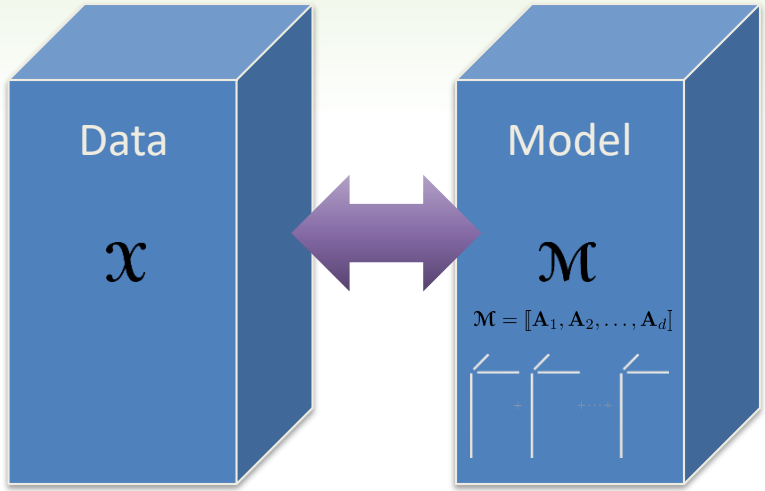
Sandia National Laboratories



Stochastic Gradient Descent for Generalized CP Decomposition

Joint work with David Hong (Michigan), Cliff Anderson-Bergman (Sandia)

"Standard" CP



Typically: Consider data to be low-rank plus "white noise"

$$x_i = m_i + \epsilon_i,\ \epsilon_i \sim \mathcal{N}(0, \sigma)$$

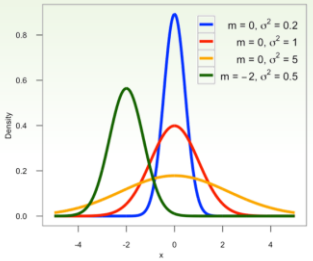
Equivalently, Gaussian with mean m_i

$$x_i \sim \mathcal{N}(m_i, \sigma)$$

$$\mathbb{E}(X_i) = m_i$$

Probability Density Function (PDF)

$$\frac{e^{-(x-m)^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}}$$



Want to maximize likelihood of model:

$$L(\mathcal{M}) = \prod_i \frac{e^{-(x_i-m_i)^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}}$$

Equivalent to minimizing negative log likelihood:

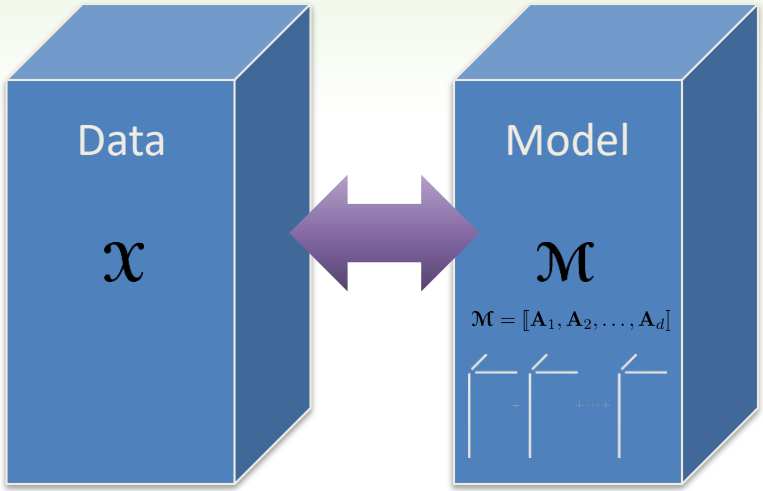
$$-\log(L(\mathcal{M})) = \sum_i \frac{(x_i - m_i)^2}{2\sigma^2} + {}^1\!/_2 \log 2\pi\sigma^2$$

Results in the "standard" objective:

$$\min F(\mathcal{X}, \mathcal{M}) = \sum_i \underbrace{(x_i - m_i)^2}_{f(x_i, m_i)}$$

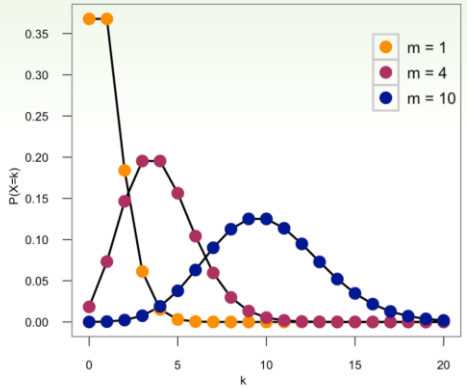
Anderson-Bergman, Hong, and Kolda 2017

Poisson CP: Identity Link



Probability Mass Function (PMF):

$$\frac{e^{-m} m^x}{x!}$$



Want to maximize likelihood of model:

$$L(\mathcal{M}) = \prod_i \frac{e^{-m_i} \, m_i^{x_i}}{x_i!}$$

Consider data to be Poisson distributed with parameter m_i

$$x_i \sim \text{Poisson}(m_i)$$

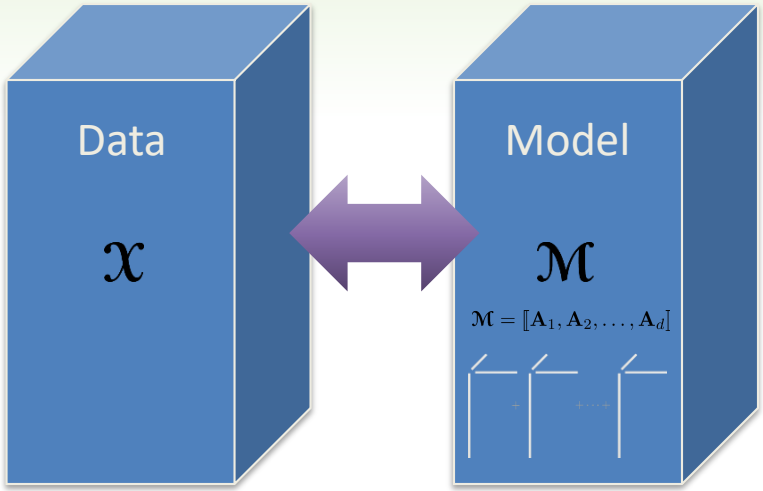
$$\mathbb{E}(X_i) = m_i$$

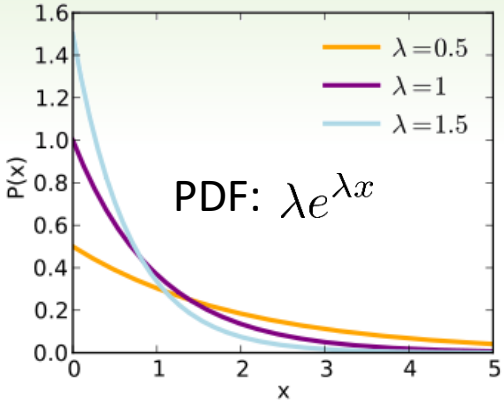
Equivalent to minimizing negative log likelihood:

$$\min F(\mathcal{X}, \mathcal{M}) = \sum_i \underbrace{m_i - x_i \log(m_i)}_{f(x_i, m_i)} + \log(x_i!)$$

Chi & Kolda 2012, Anderson-Bergman, Hong, and Kolda 2017

Exponential CP: Inverse Link





Typically: Consider data to be exponentially distributed with parameter m_i

$$x_i \sim \text{Exp}(m_i)$$

$$\mathbb{E}(X_i) = 1/m_i$$

Want to maximize likelihood of model:

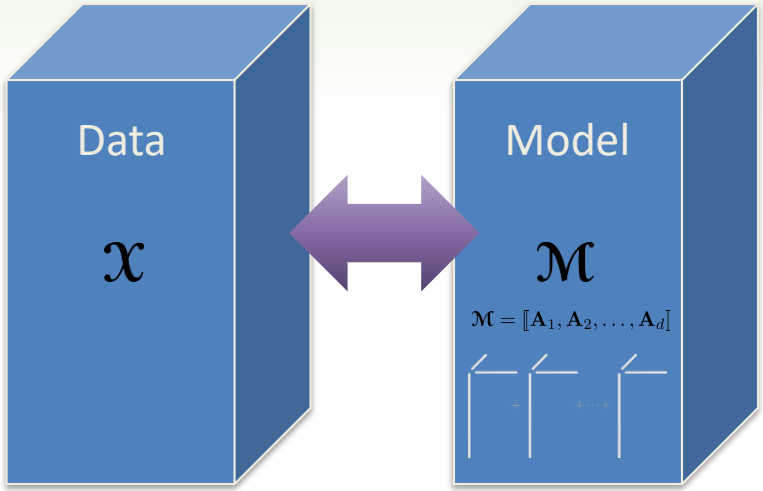
$$L(\mathcal{M}) = \prod_i m_i e^{-m_i x_i}$$

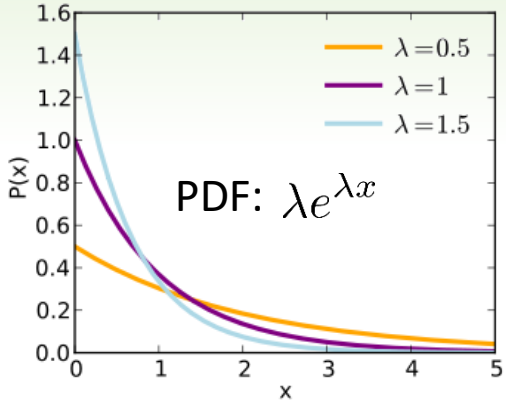
Equivalent to minimizing negative log likelihood:

$$\min F(\mathcal{X}, \mathcal{M}) = \sum_i \underbrace{-\log m_i + m_i x_i}_{f(x_i, m_i)}$$

Anderson-Bergman, Hong, and Kolda 2017

Exponential CP: Identity Link





Consider data to be exponentially distributed with parameter $1/m_i$

$$x_i \sim \text{Exp}(1/m_i)$$

$$\mathbb{E}(X_i) = m_i$$

Want to maximize likelihood of model:

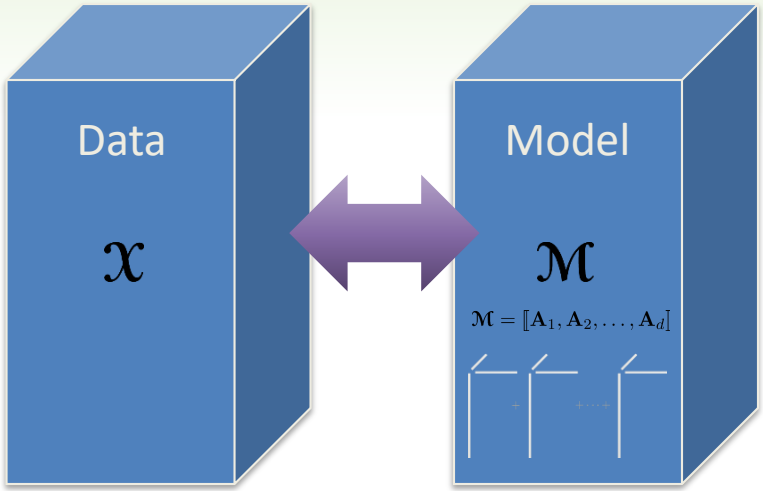
$$L(\mathcal{M}) = \prod_i (1/m_i) e^{-x_i/m_i}$$

Equivalent to minimizing negative log likelihood:

$$\min F(\mathcal{X}, \mathcal{M}) = \sum_i \underbrace{\log m_i + x_i/m_i}_{f(x_i, m_i)}$$

Anderson-Bergman, Hong, and Kolda 2017

Boolean CP: Odds Link



Consider data to be exponentially distributed with parameter m_i

$$x_i \sim \text{Bernoulli}(m_i/(1+m_i))$$

$$\mathbb{E}(X_i) = m_i/(1+m_i)$$

Probability Mass Function (PMF):

$$\left(\frac{m}{1+m}\right)^{x}\left(1-\frac{m}{1+m}\right)^{(1-x)}$$

Want to maximize likelihood of model:

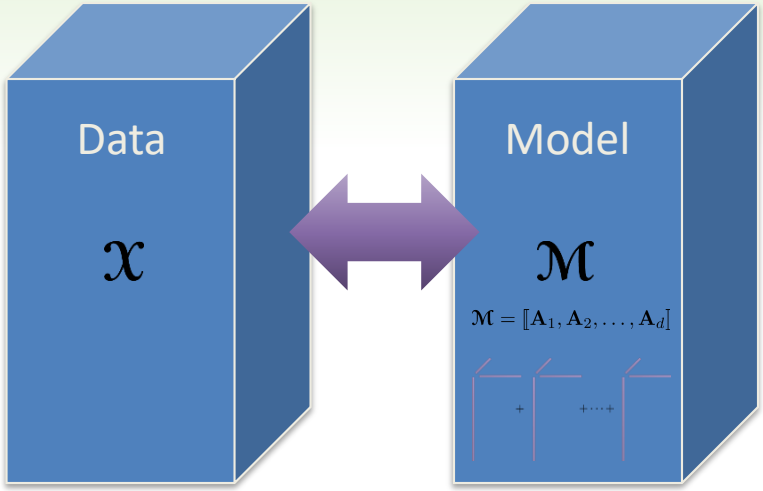
$$L(\mathcal{M}) = \prod_{\mathbf{i}\in\mathcal{I}} \left(\frac{m_{\mathbf{i}}}{1+m_{\mathbf{i}}}\right)^{x_{\mathbf{i}}}\left(1-\frac{m_{\mathbf{i}}}{1+m_{\mathbf{i}}}\right)^{(1-x_{\mathbf{i}})}$$

Equivalent to minimizing negative log likelihood:

$$F(\mathcal{X},\mathcal{M}) = \sum_{\mathbf{i}\in\mathcal{I}} \underbrace{\log(m_{\mathbf{i}}+1) - x_{\mathbf{i}}\log m_{\mathbf{i}}}_{f(x_i,m_i)}$$

Anderson-Bergman, Hong, and Kolda 2017

Generalized CP



$$\min \quad F(\mathcal{X}, \mathcal{M}) = \sum_i f(x_i, m_i)$$

$$\text{s.t.} \quad \mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$$

"Standard" CP uses:

$$f(x, m) = (x - m)^2$$

"Exponential" CP uses :

$$f(x, m) = xm - \log m \quad (x \geq 0, m \geq 0)$$

"Poisson" CP (Chi-Kolda 2012) uses:

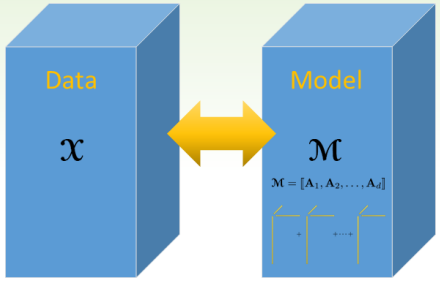
$$f(x, m) = m - x \log m \quad (x \in \mathbb{N}, m \geq 0)$$

"Boolean" CP uses:

$$f(x, m) = \log(m + 1) - x \log m \quad (x \in \{0, 1\}, m \geq 0)$$

Anderson-Bergman, Hong, and Kolda 2017

Generalized CP Gradient



$$\min \quad F(\mathcal{X}, \mathcal{M}) = \sum_i f(x_i, m_i)$$

$$\text{s.t.} \quad \mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$$

Define a tensor $\mathcal{G}$ such that $g(i_1, \ldots, i_d) = g_i = \dfrac{\partial f}{\partial m}(x_i, m_i)$

Recall $m_i = \sum_{j=1}^{r} a_1(i_1, j)\, a_2(i_2, j) \cdots a_d(i_d, j)$

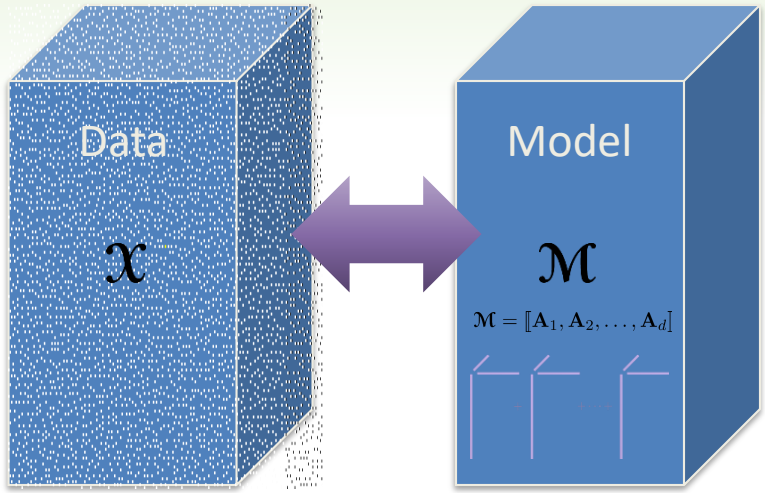
Then $$\frac{\partial F}{\partial a_k(i_k, j)} = \underbrace{g(i_1, \ldots, i_d)}_{\text{No dependency on model form}} \underbrace{a_1(i_1, j) \cdots a_{k-1}(i_{k-1}, j)\, a_{k+1}(i_{k+1}, j) \cdots a_d(i_d, j)}_{\text{No dependency of function}}$$

Matrix Version: $$\frac{\partial F}{\partial \mathbf{A}_k} = \mathbf{G}_{(k)}(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1)$$

MTTKRP!

Anderson-Bergman, Hong, and Kolda 2017

Generalized CP Gradient with Missing Data



Ω = Set of Known Entries in Data Tensor

$$\min \quad F(\mathcal{X}, \mathcal{M}) = \sum_{i \in \Omega} f(x_i, m_i)$$

$$\text{s.t.} \quad \mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$$

Define a tensor $\mathcal{G}$ such that

$$g_i = \begin{cases} \frac{\partial f}{\partial m}(x_i, m_i) & \text{if } i \in \Omega, \\ 0 & \text{if } i \notin \Omega. \end{cases}$$

Then (no change except G):

$$\frac{\partial F}{\partial \mathbf{A}_k} = \mathbf{G}_{(k)}(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1)$$

Anderson-Bergman, Hong, and Kolda 2017, Acar, Dunlavy, Kolda, Morup 2011

Fitting Generalized CP

Given data tensor $\mathcal{X}$ and initial guess $\mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$.

1: $F_0 \leftarrow \sum_i f(x_i, m_i)$
2: **for** $\ell = 1, 2, \ldots$ **do**
3: Compute $\mathcal{G}$ based on current $\mathcal{M}$
4: **for** $k = 1, \ldots, d$ **do**
5: $\Delta_k \leftarrow \mathbf{G}_{(k)}(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1)$
6: **end for**
7: Determine step length α
8: **for** $k = 1, \ldots, d$ **do**
9: $\mathbf{A}_k \leftarrow \mathbf{A}_k - \alpha\Delta_k$
10: **end for**
11: $F_\ell \leftarrow \sum_i f(x_i, m_i)$ using updated $\mathcal{M}$
12: Check convergence
13: **end for**

Compute Gradient

Compute optimization step and take it

Use either checks on function value or gradient

Acar, Dunlavy & Kolda, 2011

Observations on Generalized CP

- Can use any optimization method to solve the optimization problem

$$\min \quad F(\mathcal{X}, \mathcal{M}) = \sum_{i \in \Omega} f(x_i, m_i) \qquad \text{s.t.} \quad \mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$$

- Can easily add regularization

$$\min \quad F(\mathcal{X}, \mathcal{M}) = \sum_{i \in \Omega} f(x_i, m_i) + \sum_{k} \rho_k(\mathbf{A}_k) \quad \text{s.t.} \quad \mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$$

- If data tensor (X) is sparse, there is no guarantee that the gradient will be efficient
  - Standard CP is a special case that has exploitable structure
  - Scalability is potentially a major problem

- If known data is sparse (Ω), then gradient will be efficient
  - Doesn't require any sparsity in data tensor
  - Perhaps this can be exploited? Yes – stochastic algorithms.

Anderson-Bergman, Hong, and Kolda 2017

"Stochastic" Generalized CP

Given data tensor $\mathcal{X}$ and initial guess $\mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$.

1: $F_0 \leftarrow \sum_{i \in \hat{\Omega}} f(x_i, m_i)$ ← *Same sample every time in line 1 & 13*
2: **for** $\ell = 1, 2, \ldots$ **do**
3: **for** $t = 1, \ldots, T$ **do**
4: $\Omega \leftarrow N$ random entries in the data tensor ← *Different samples every time*
5: Compute $\mathcal{G}$ *only at entries in* Ω (everywhere else is zero)
6: **for** $k = 1, \ldots, d$ **do**
7: $\Delta_k \leftarrow \mathbf{G}_{(k)}(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1)$ ← *Sparse MTTKRP, super cheap!*
8: **end for**
9: **for** $k = 1, \ldots, d$ **do**
10: $\mathbf{A}_k \leftarrow \mathbf{A}_k - \alpha \Delta_k$
11: **end for**
12: **end for**
13: $F_\ell \leftarrow \sum_{i \in \hat{\Omega}} f(x_i, m_i)$ using updated $\mathcal{M}$
14: Check convergence
15: **end for**

Epoch

Compute Stochastic Gradient

Take tiny step (alpha is small)

Using *approximate* function values

Anderson-Bergman, Hong, and Kolda 2017

ADAM SGD for CP

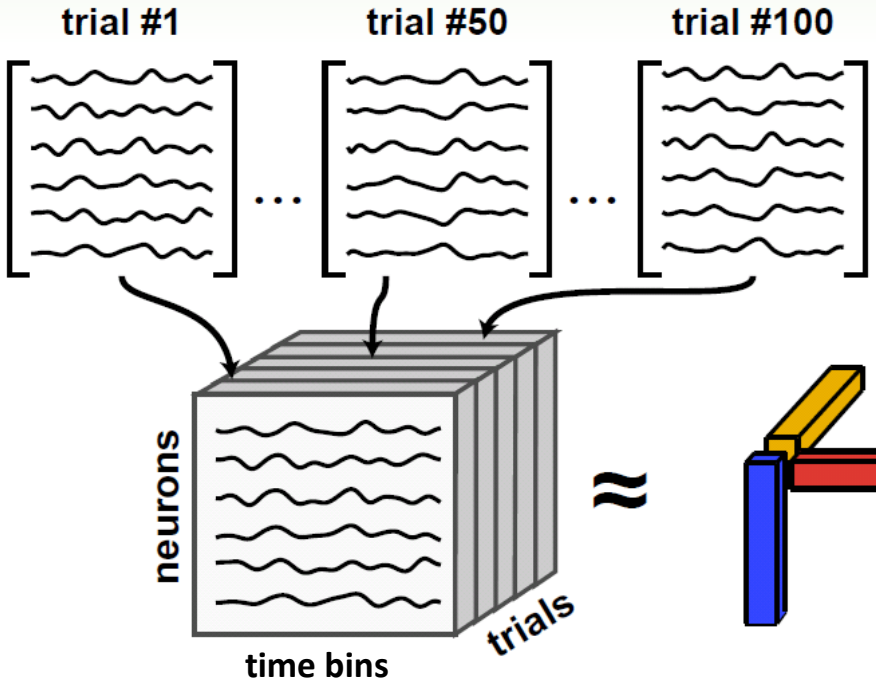
Given data tensor $\mathcal{X}$ and initial guess $\mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$.

for $k = 1, \ldots, d$ **do**
  $\Phi_k \leftarrow 0$, $\Psi_k \leftarrow 0$
end for
$F_0 \leftarrow \sum_{i \in \hat{\Omega}} f(x_i, m_i)$
for $\ell = 1, 2, \ldots$ **do**
  for $t = 1, \ldots, T$ **do**
    $\Omega \leftarrow N$ random entries in the data tensor
    Compute $\mathcal{G}$ *only at entries in* Ω (everywhere else is zero)
    for $k = 1, \ldots, d$ **do**
      $\Delta_k \leftarrow \mathbf{G}_{(k)}(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1)$
    end for
    for $k = 1, \ldots, d$ **do**
      $\Phi_k \leftarrow \beta_1 \Phi_k + (1 - \beta_1)\Delta_k$, $\hat{\Phi}_k \leftarrow \Phi_k / (1 - \beta_1^{(\ell-1)T+t})$
      $\Psi_k \leftarrow \beta_2 \Psi_k + (1 - \beta_2)\Delta_k^2$, $\hat{\Psi}_k \leftarrow \Psi_k / (1 - \beta_2^{(\ell-1)T+t})$
      $\mathbf{A}_k \leftarrow \mathbf{A}_k - \alpha \Phi_k / (\sqrt{\Psi_k} + \varepsilon)$
    end for
  end for
  $F_\ell \leftarrow \sum_{i \in \hat{\Omega}} f(x_i, m_i)$ using updated $\mathcal{M}$
  Check convergence
end for

Moment Correction Terms

Kingma & Ba 2015, Anderson-Bergman, Hong, and Kolda 2017

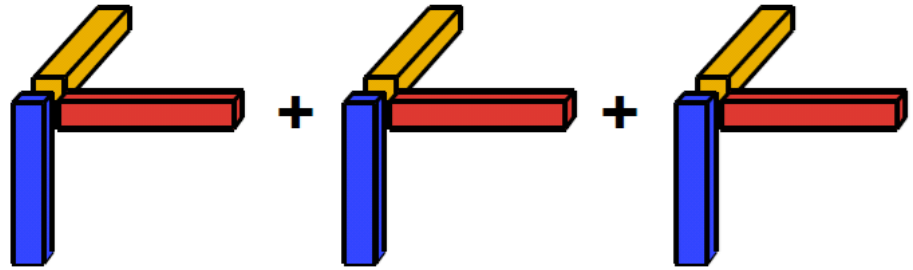
CP for Simultaneous Analysis of Neurons, Time, and Trial



Canonical Polyadic (CP) decomposition

high-dimensional neural data

- neuron factors
- within-trial factors
- across-trial factors



Past work could only look at 2 factors at once: Time x Neuron, Trial x Neuron, etc.

Let's do a nonnegative factorization using exponential assumption…

Algorithm Details for Mouse Data

Given data tensor $\mathcal{X}$ and initial guess $\mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$.

for $k = 1, \ldots, d$ **do**
  $\Phi_k \leftarrow 0$, $\Psi_k \leftarrow 0$
end for
$F_0 \leftarrow \sum_{i \in \hat{\Omega}} f(x_i, m_i)$
for $\ell = 1, 2, \ldots$ **do**
  for $t = 1, \ldots, T$ **do**
    $\Omega \leftarrow N$ random entries in the data tensor
    Compute $\mathcal{G}$ *only at entries in* Ω (everywhere else is zero)
    for $k = 1, \ldots, d$ **do**
      $\Delta_k \leftarrow \mathbf{G}_{(k)}(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1)$
    end for
    for $k = 1, \ldots, d$ **do**
      $\Phi_k \leftarrow \beta_1 \Phi_k + (1 - \beta_1)\Delta_k$, $\hat{\Phi}_k \leftarrow \Phi_k / (1 - \beta_1^{(\ell-1)T+t})$
      $\Psi_k \leftarrow \beta_2 \Psi_k + (1 - \beta_2)\Delta_k^2$, $\hat{\Psi}_k \leftarrow \Psi_k / (1 - \beta_2^{(\ell-1)T+t})$
      $\mathbf{A}_k \leftarrow \mathbf{A}_k - \alpha \hat{\Phi}_k / (\sqrt{\hat{\Psi}_k} + \varepsilon)$
    end for
  end for
  $F_\ell \leftarrow \sum_{i \in \hat{\Omega}} f(x_i, m_i)$ using updated $\mathcal{M}$
  Check convergence
end for

$$f(x, m) = -\log m + xm$$

$$|\mathcal{X}| \sim 10^7$$

$$|\hat{\Omega}| = 200,000$$

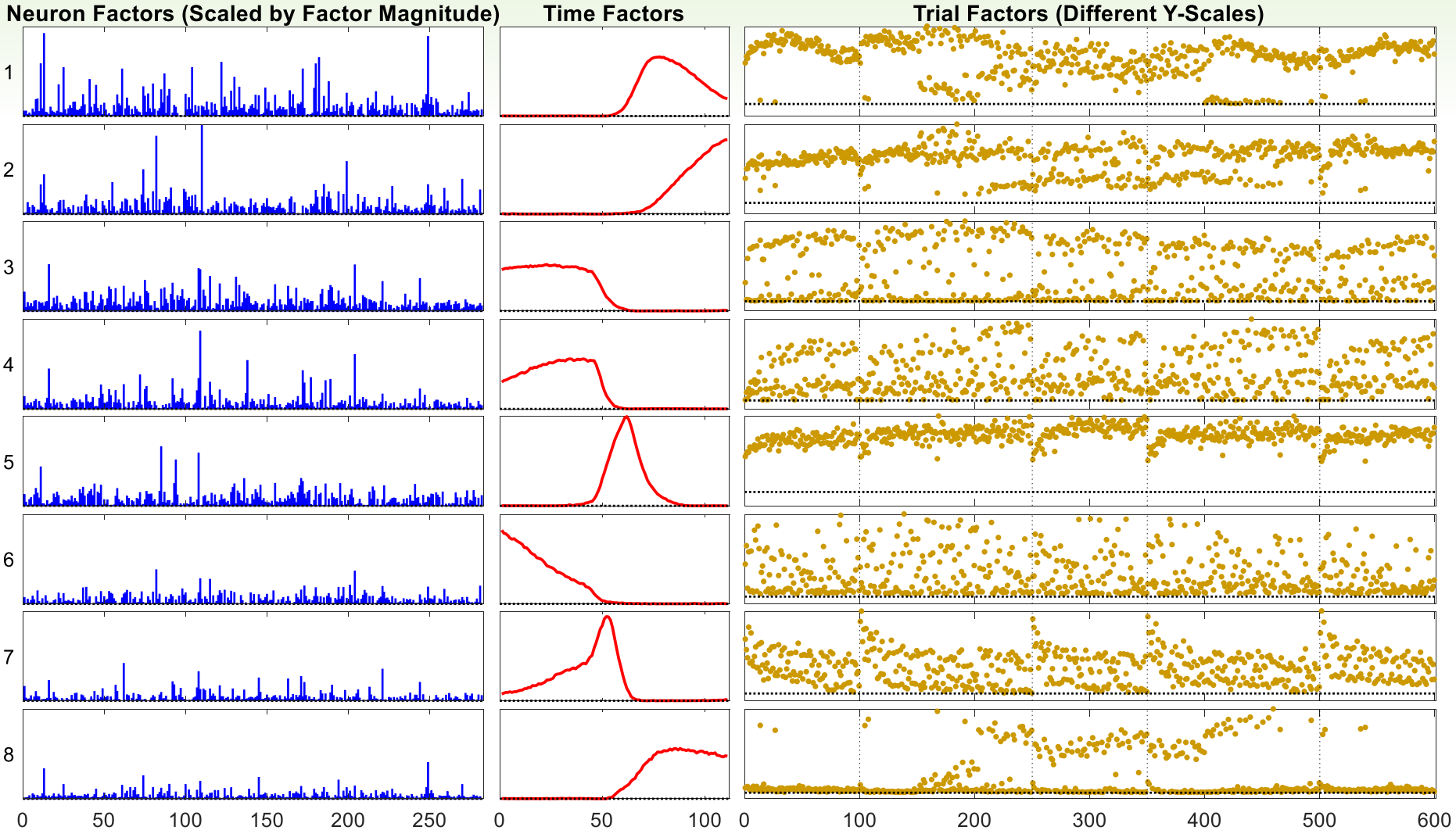
$$|\Omega| = 1,000$$

$$T = 1000$$

$$\alpha = .001$$
$$\beta_1 = 0.9$$
$$\beta_2 = 0.999$$
$$\epsilon = 10^{-8}$$

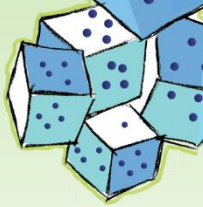
Anderson-Bergman, Hong, and Kolda 2017

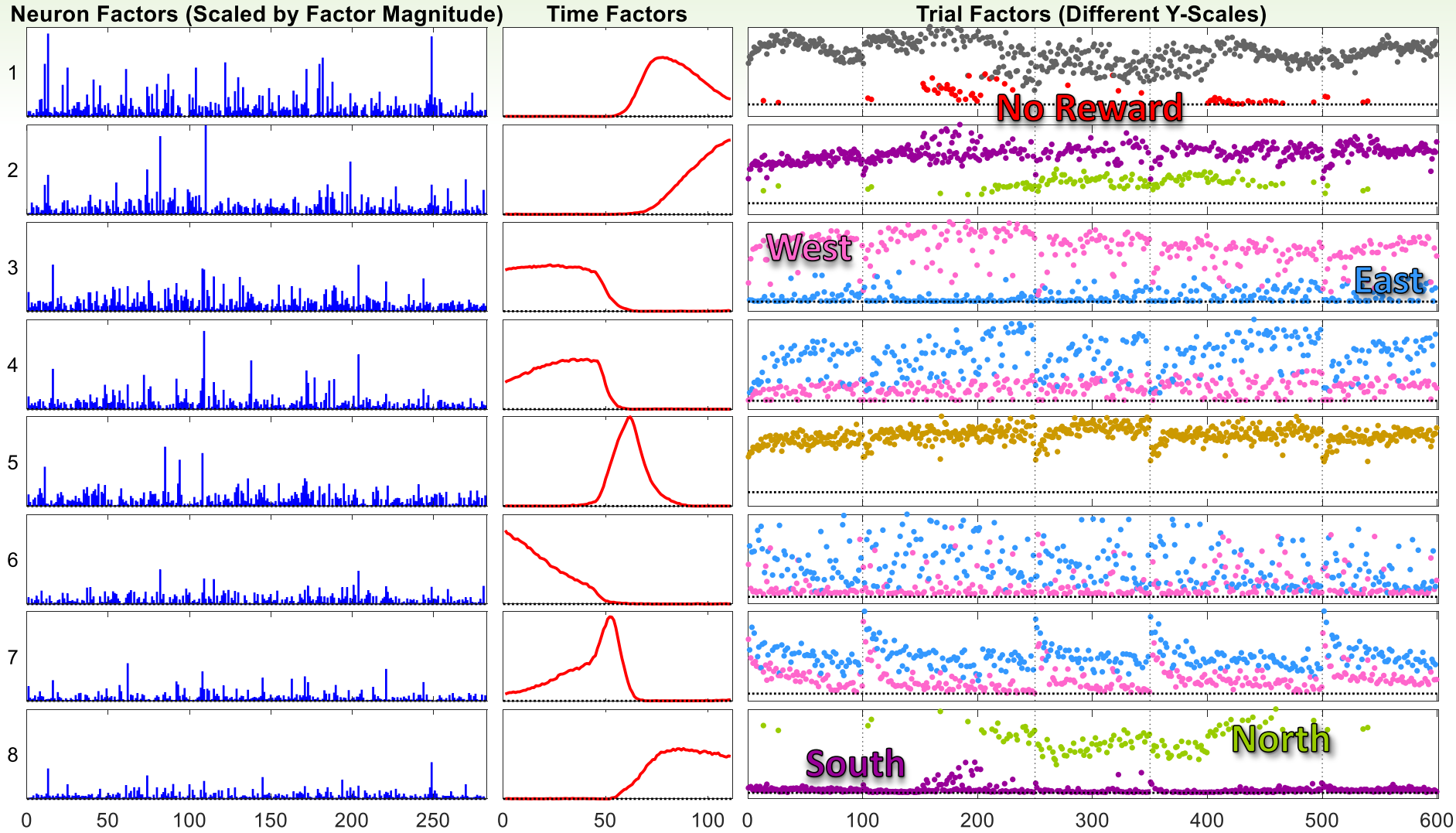
Revisiting Mouse Neuron Data Genten-Exponential CP (Nonneg)



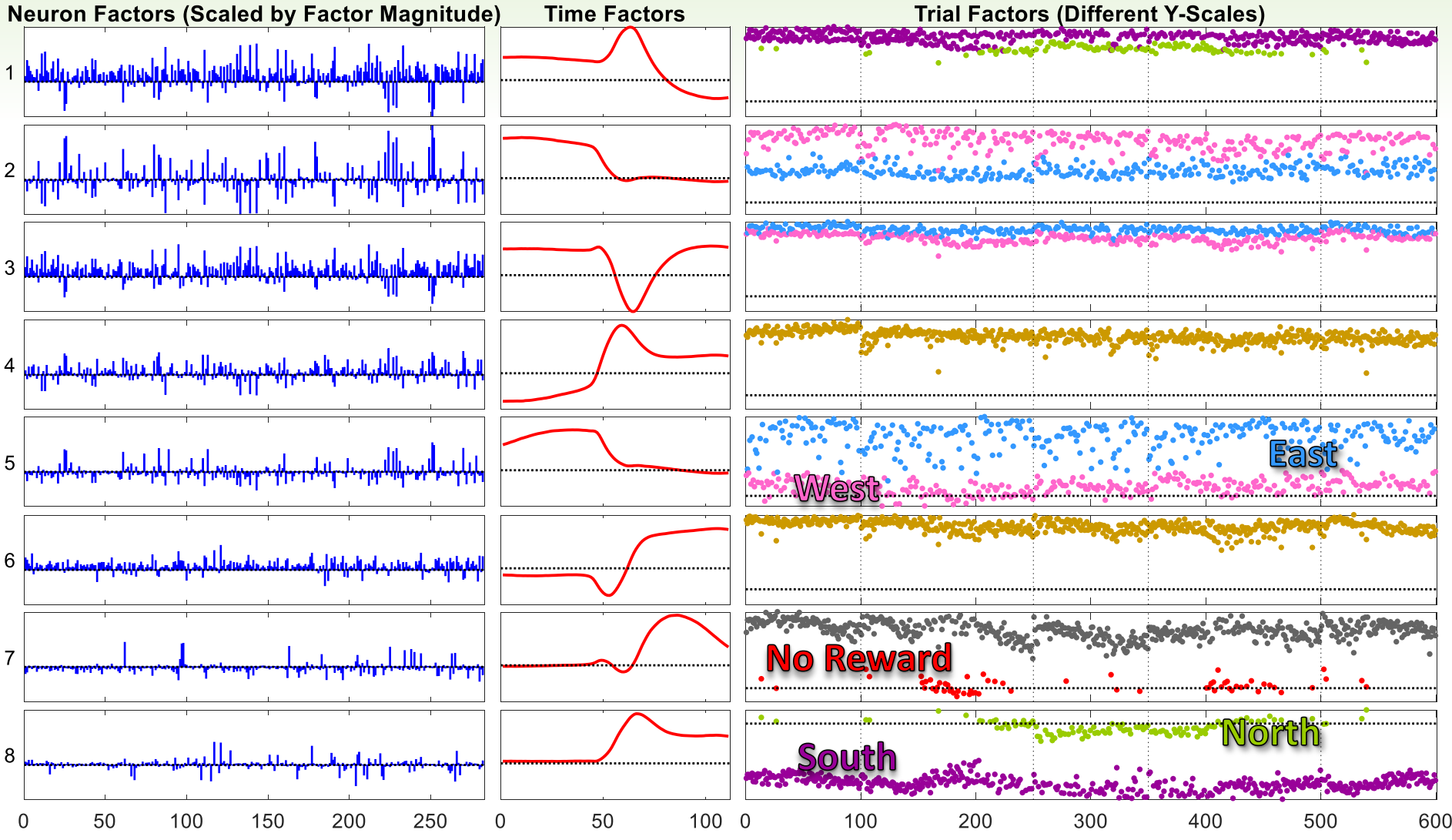
Revisiting Mouse Neuron Data Genten-Exponential CP (Nonneg)

Sandia National Laboratories





Gaussian is harder to interpret due to cancellation effects…



Example Results: Senate Voting Data

- Data on U.S. Senate: 1989-2016
- 271 Senators
  - Two senators per state (100 total)
  - Six-year term, can serve multiple terms
  - Two main parties: Republicans (red) & Democrats (blue)
- 9044 Items
  - Roll calls from 1989 to 2016
- 3 possible Votes
  - Yea, Nay, No Vote (NA)
- Tensor is 3-way & binary
  - *63% missing*
  - *Stochastic method only samples from <u>known</u> entries in data tensor*

Anderson-Bergman, Hong, and Kolda 2017

Algorithm Details for Senate Voting

Given data tensor $\mathcal{X}$ and initial guess $\mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$.

for $k = 1, \ldots, d$ **do**
  $\Phi_k \leftarrow 0$, $\Psi_k \leftarrow 0$
end for
$F_0 \leftarrow \sum_{i \in \hat{\Omega}} f(x_i, m_i)$
for $\ell = 1, 2, \ldots$ **do**
  for $t = 1, \ldots, T$ **do**
    $\Omega \leftarrow N$ random entries in the data tensor
    Compute $\mathcal{G}$ *only at entries in* Ω (everywhere else is zero)
    for $k = 1, \ldots, d$ **do**
      $\Delta_k \leftarrow \mathbf{G}_{(k)}(\mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1)$
    end for
    for $k = 1, \ldots, d$ **do**
      $\Phi_k \leftarrow \beta_1 \Phi_k + (1 - \beta_1)\Delta_k$, $\hat{\Phi}_k \leftarrow \Phi_k / (1 - \beta_1^{(\ell-1)T+t})$
      $\Psi_k \leftarrow \beta_2 \Psi_k + (1 - \beta_2)\Delta_k^2$, $\hat{\Psi}_k \leftarrow \Psi_k / (1 - \beta_2^{(\ell-1)T+t})$
      $\mathbf{A}_k \leftarrow \mathbf{A}_k - \alpha \hat{\Phi}_k / (\sqrt{\hat{\Psi}_k} + \varepsilon)$
    end for
  end for
  $F_\ell \leftarrow \sum_{i \in \hat{\Omega}} f(x_i, m_i)$ using updated $\mathcal{M}$
  Check convergence
end for

$$f(x, m) = \log(m + 1) - x \log m$$

$$|\mathcal{X}| \sim 10^7 \quad |\mathrm{nnz}(\mathcal{X})| \sim 10^7$$

$$|\hat{\Omega}| = 200,000$$

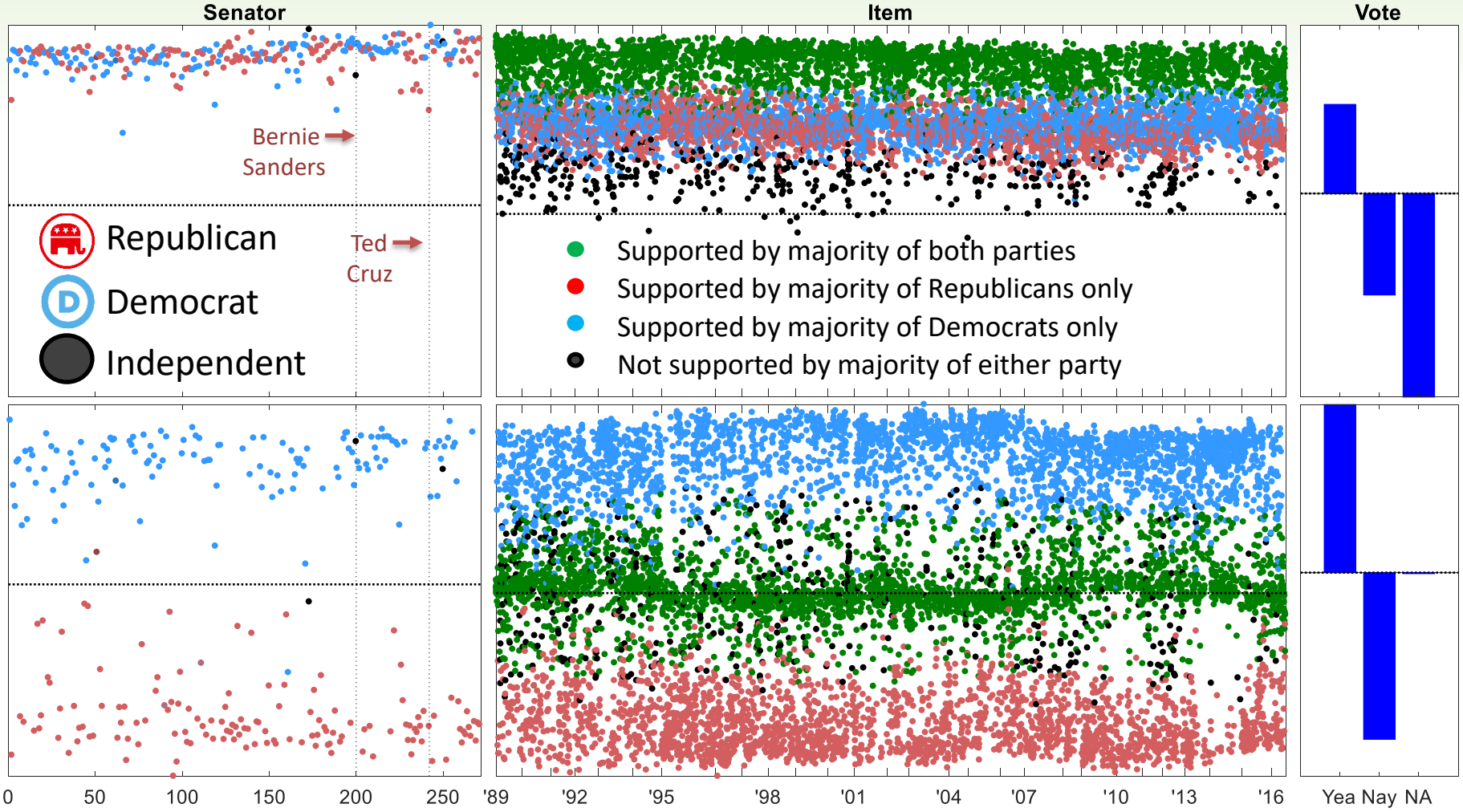
$$|\Omega| = 10,000$$

$$T = 2000$$

$$\alpha = .001$$
$$\beta_1 = 0.9$$
$$\beta_2 = 0.999$$
$$\epsilon = 10^{-8}$$

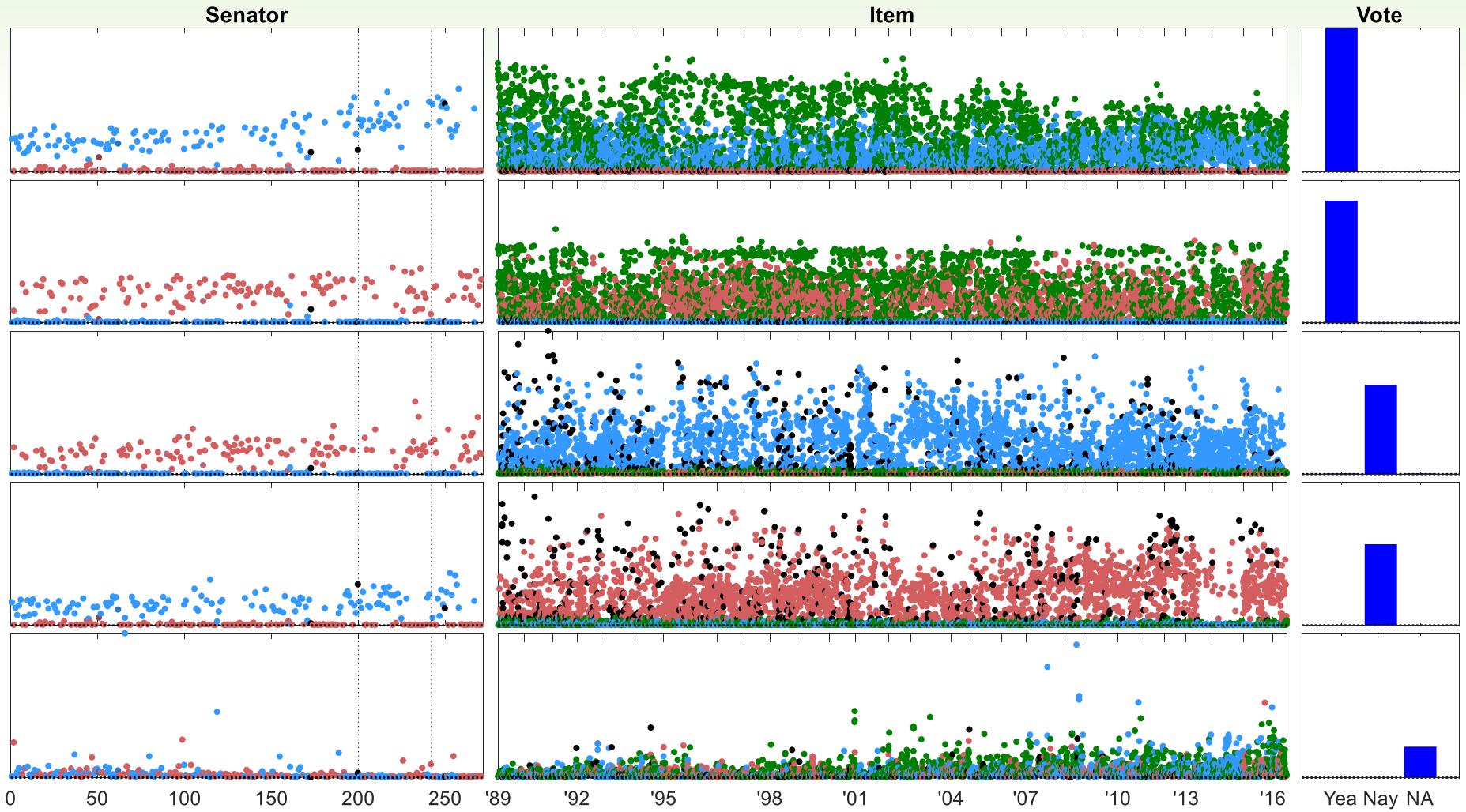
Anderson-Bergman, Hong, and Kolda 2017

Gaussian (+/- 1 Data)



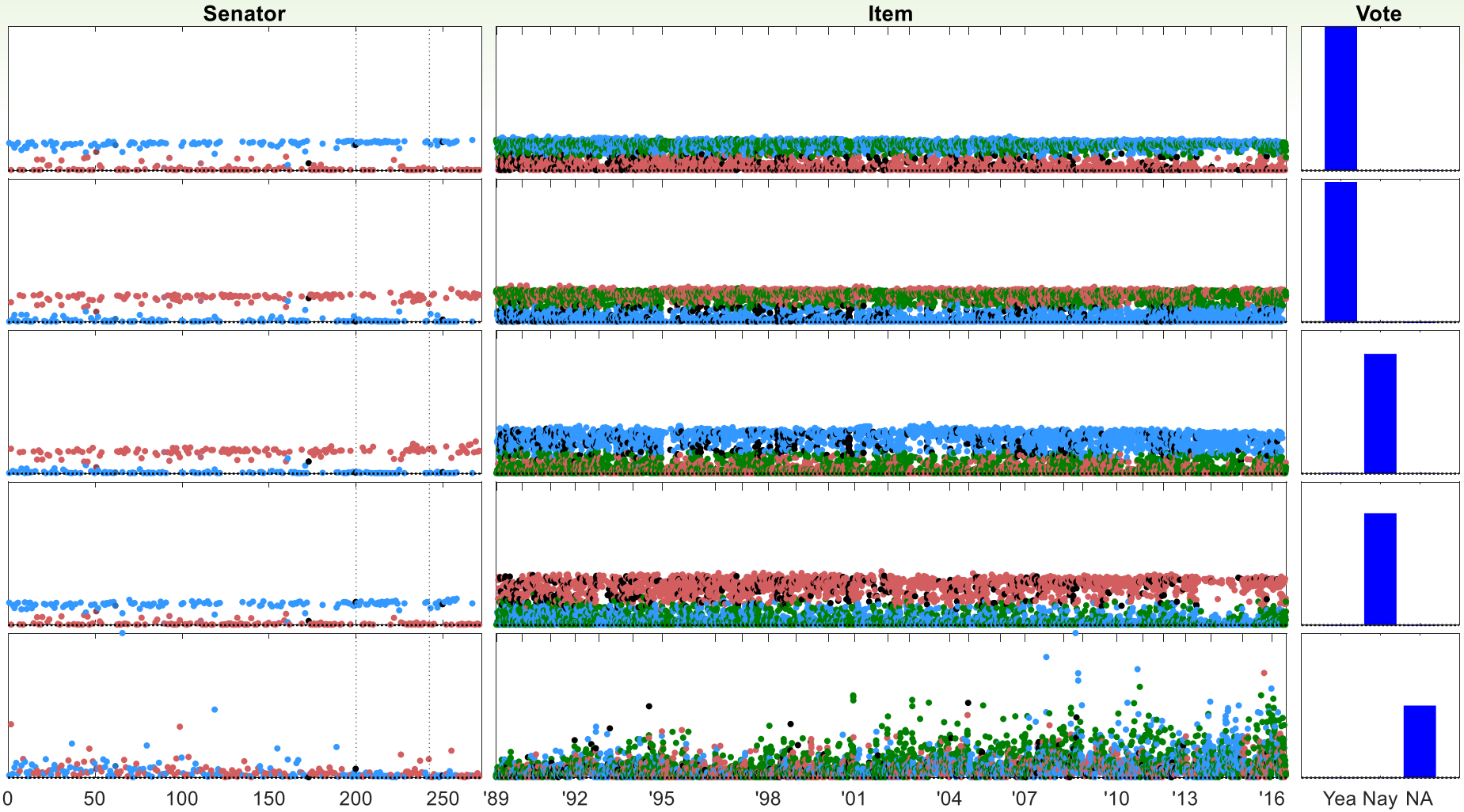
Anderson-Bergman, Hong, and Kolda 2017

Bernoulli



Anderson-Bergman, Hong, and Kolda 2017

Poisson



Anderson-Bergman, Hong, and Kolda 2017

Poisson Tensor Factorization

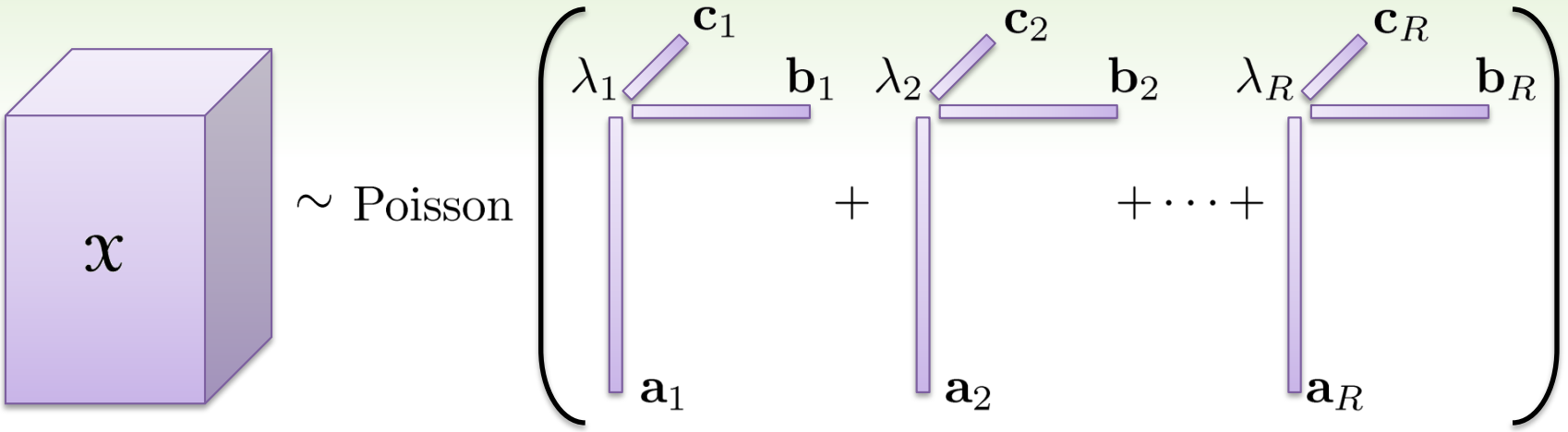
Joint work with Eric Chi (NC State), Todd Plantenga (FireEye), Sammy Hansen (Spotify)

Sparse Tensor Computations

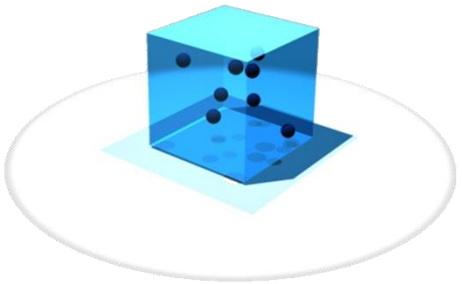
- Many real-world data analysis problems are naturally expressed as in terms of a *sparse* tensor
  - Computer traffic analysis
  - Term-document analysis
  - Email analysis
  - Link prediction
  - Web page analysis

- But where do sparse tensors come from?
  - How are the entries generated?
  - How can we use that information in model fitting?

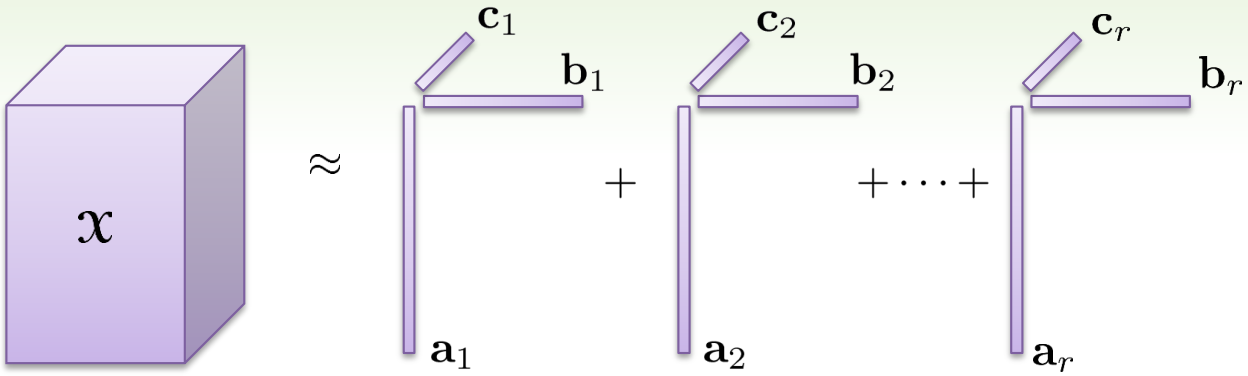
Generating Sparse Test Data



- Each "occurrence" generated as follows
- Choose component j proportional to λ_j
- Given component j:
  - Choose index i_1 proportional to $\mathbf{a}_j$
  - Choose index i_2 proportional to $\mathbf{b}_j$
  - Choose index i_3 proportional to $\mathbf{c}_j$
- Increment $x(i_1, i_2, i_3)$ by one
- Repeat



Solving the Poisson Regression Problem



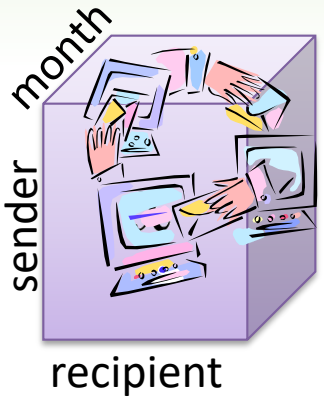
$$\min_{\mathcal{M}} \sum_i m_i - x_i \log m_i \text{ subject to } \mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$$

- Highly nonconvex problem!
  - Assume r is given
- Alternating Poisson regression
  - Assume $(d-1)$ factor matrices are known and solve for the remaining one
  - Multiplicative updates like Lee & Seung (2000) for NMF, but improved
  - Typically assume data tensor $\mathcal{X}$ is sparse and have special methods for this
  - Newton or Quasi-Newton method

Chi & Kolda 2012; Hansen, Plantenga, & Kolda 2015

Motivating Example: Enron Email

- Emails from Enron FERC investigation
  - 8540 Messages
  - 28 Months (from Dec 1999 to Mar 2002)
  - 105 People (sent and received at least one email every month)
  - $x(i_1, i_2, i_3)$ = # emails from sender i_1 to recipient i_2 in month i_3
  - 105 x 105 x 28 = 308,700 possible entries
  - 8,500 nonzero counts
  - **3% dense**

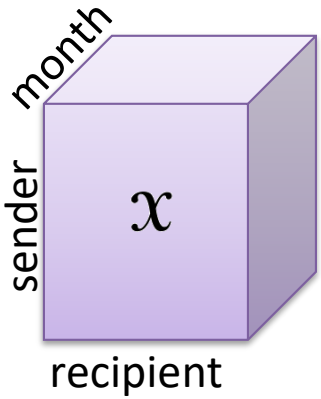


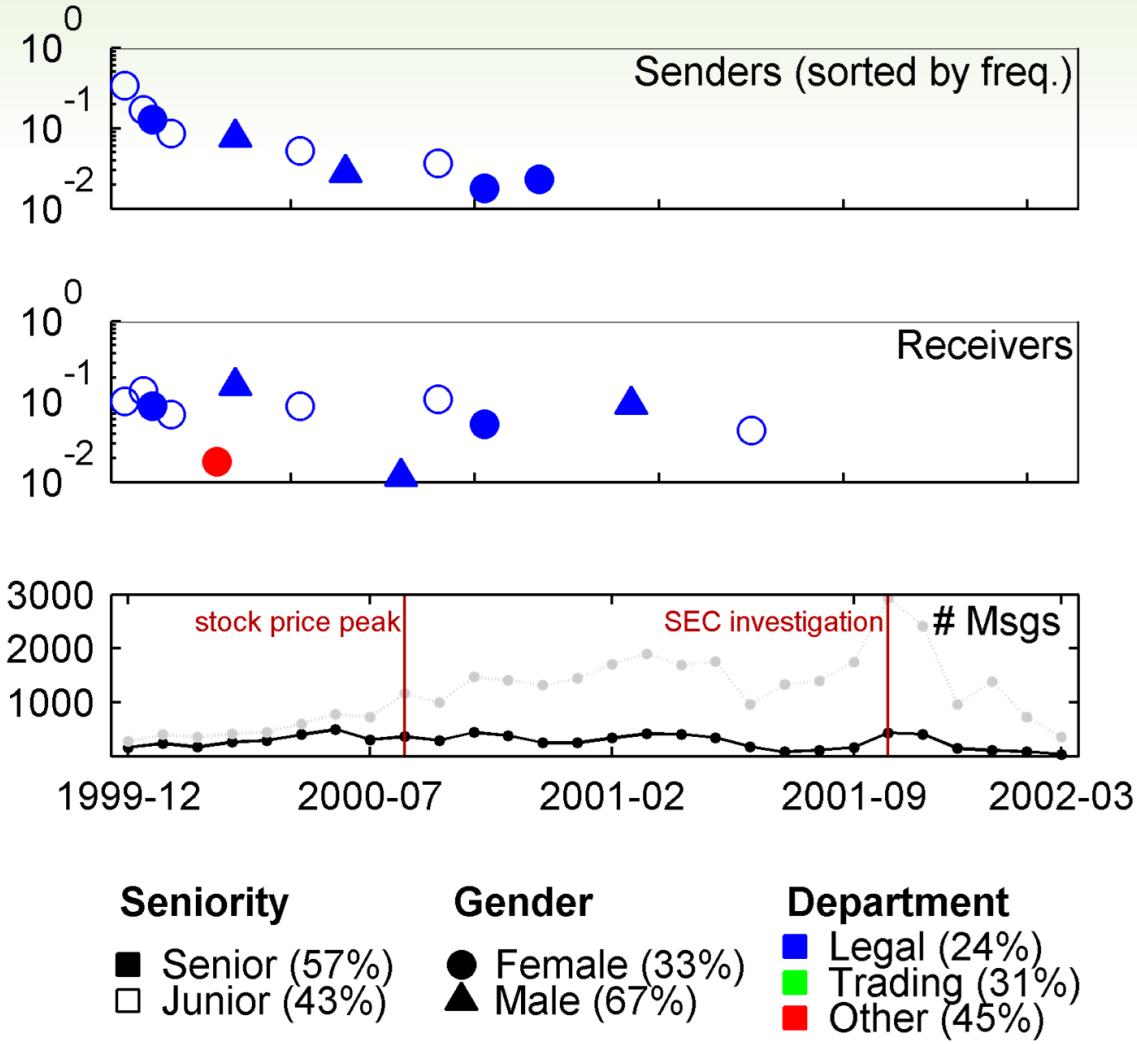
- Questions: What can we learn about this data?
  - Each person labeled by Zhou et al. (2007); see also Owen and Perry (2010)
    - Seniority: 57% senior, 43% junior
    - Gender: 67% male, 33% female
    - Department: 24% legal, 31% trading, 45% other

This information is not part of the tensor factorization

Enron Email Data (Component 1)

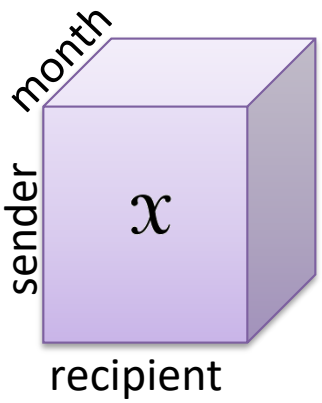
Legal Dept;
Mostly Female

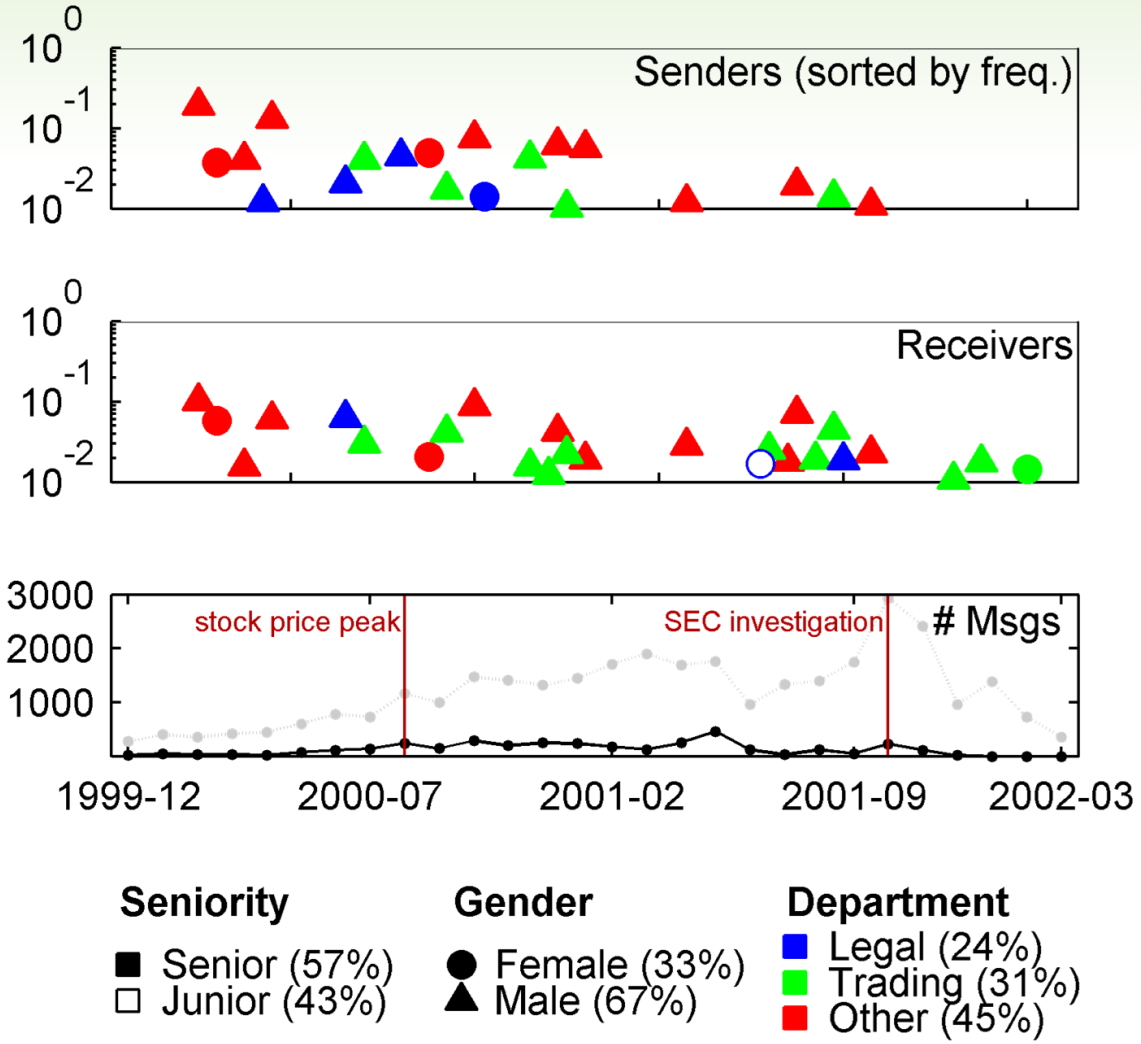




Enron Email Data (Component 3)

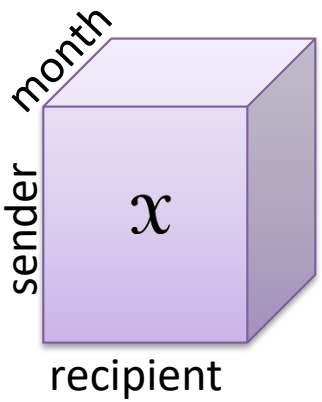
Senior;
Mostly Male

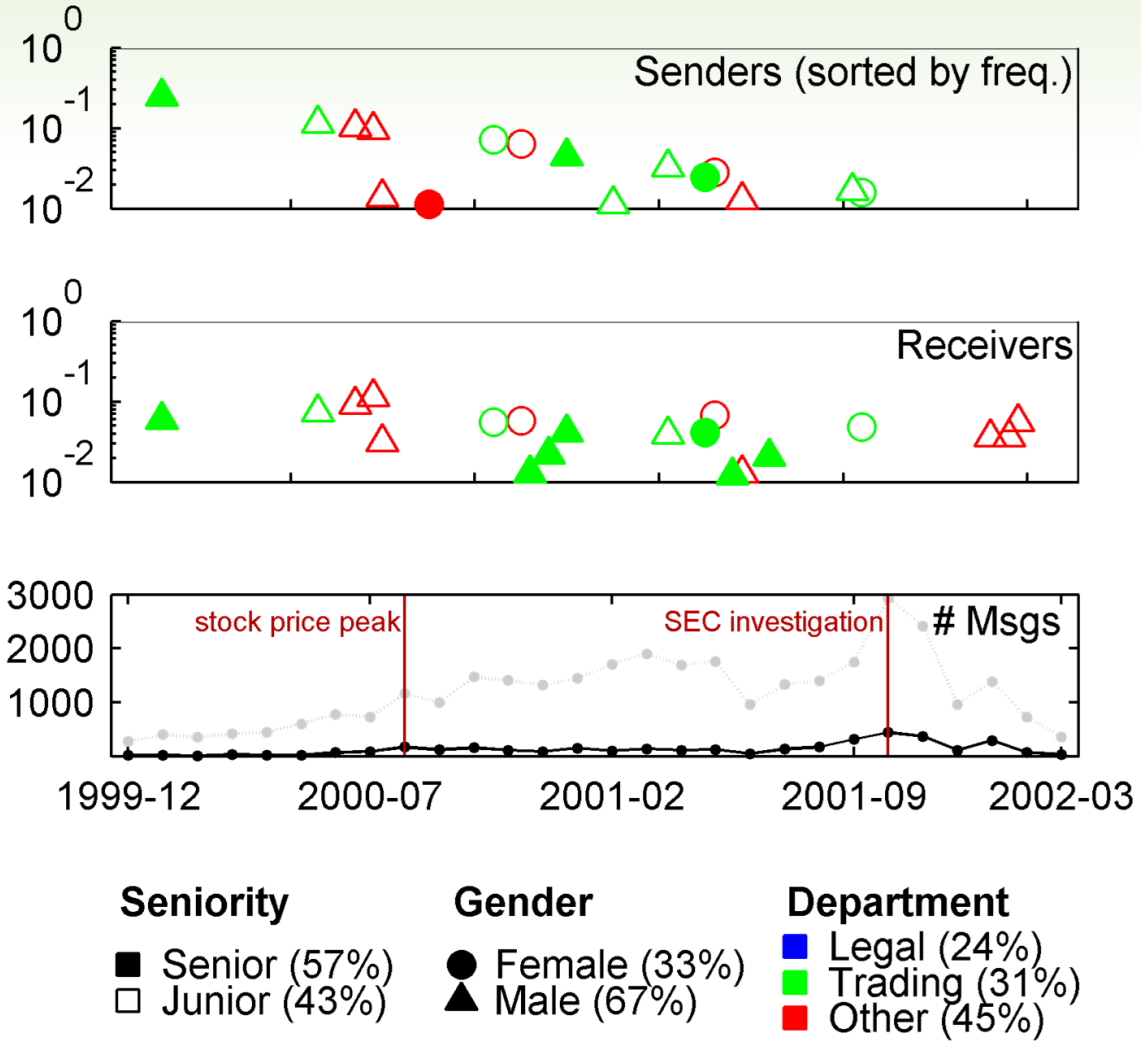




Enron Email Data (Component 4)

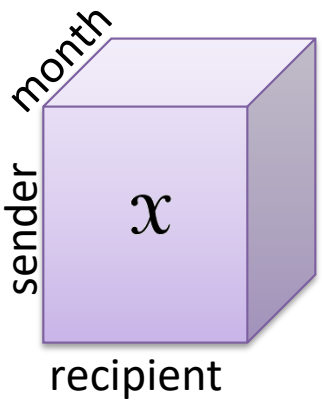
Not Legal

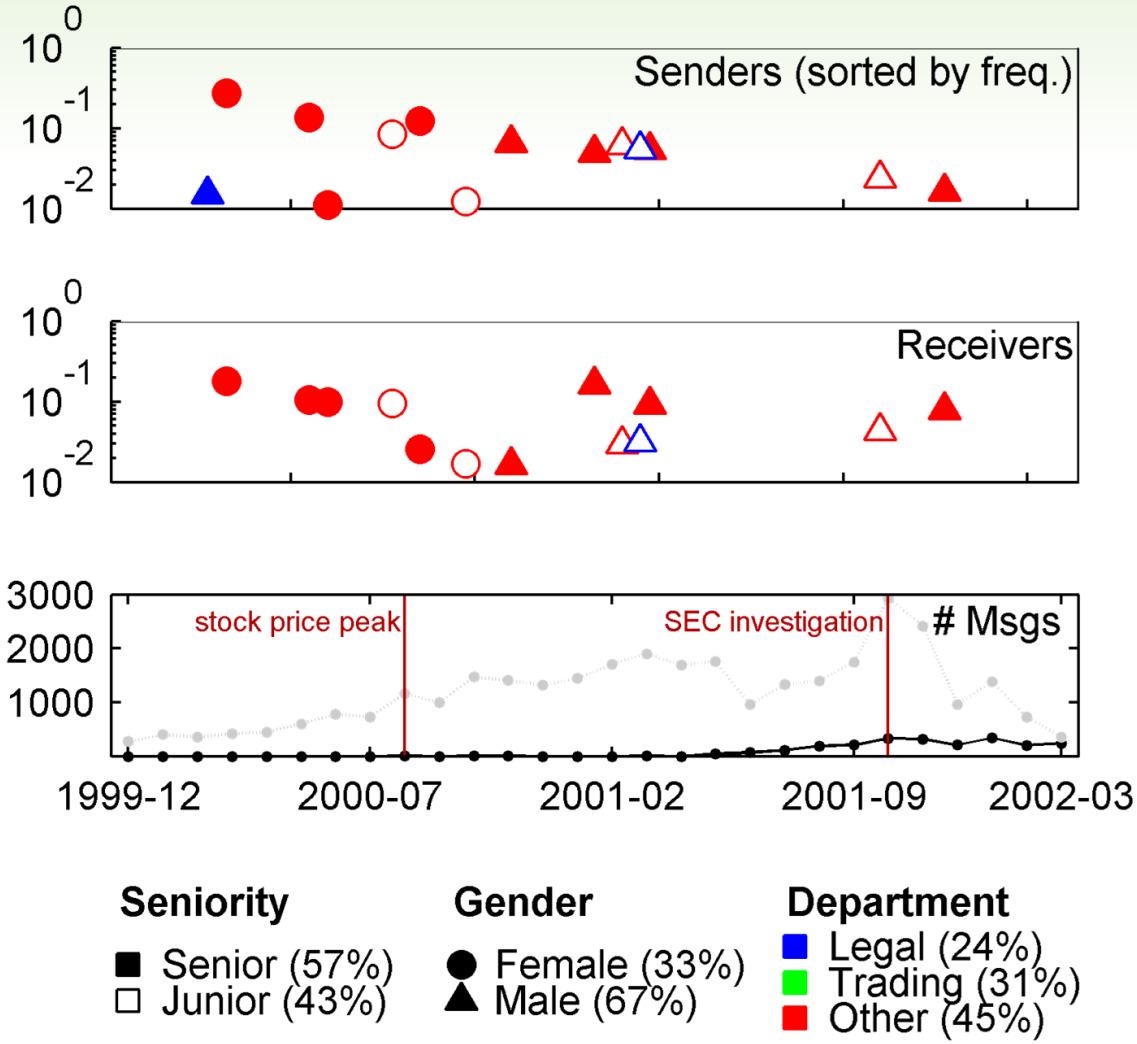




Enron Email Data (Component 5)

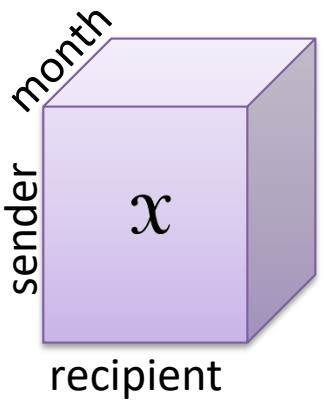
Other;
Mostly Female

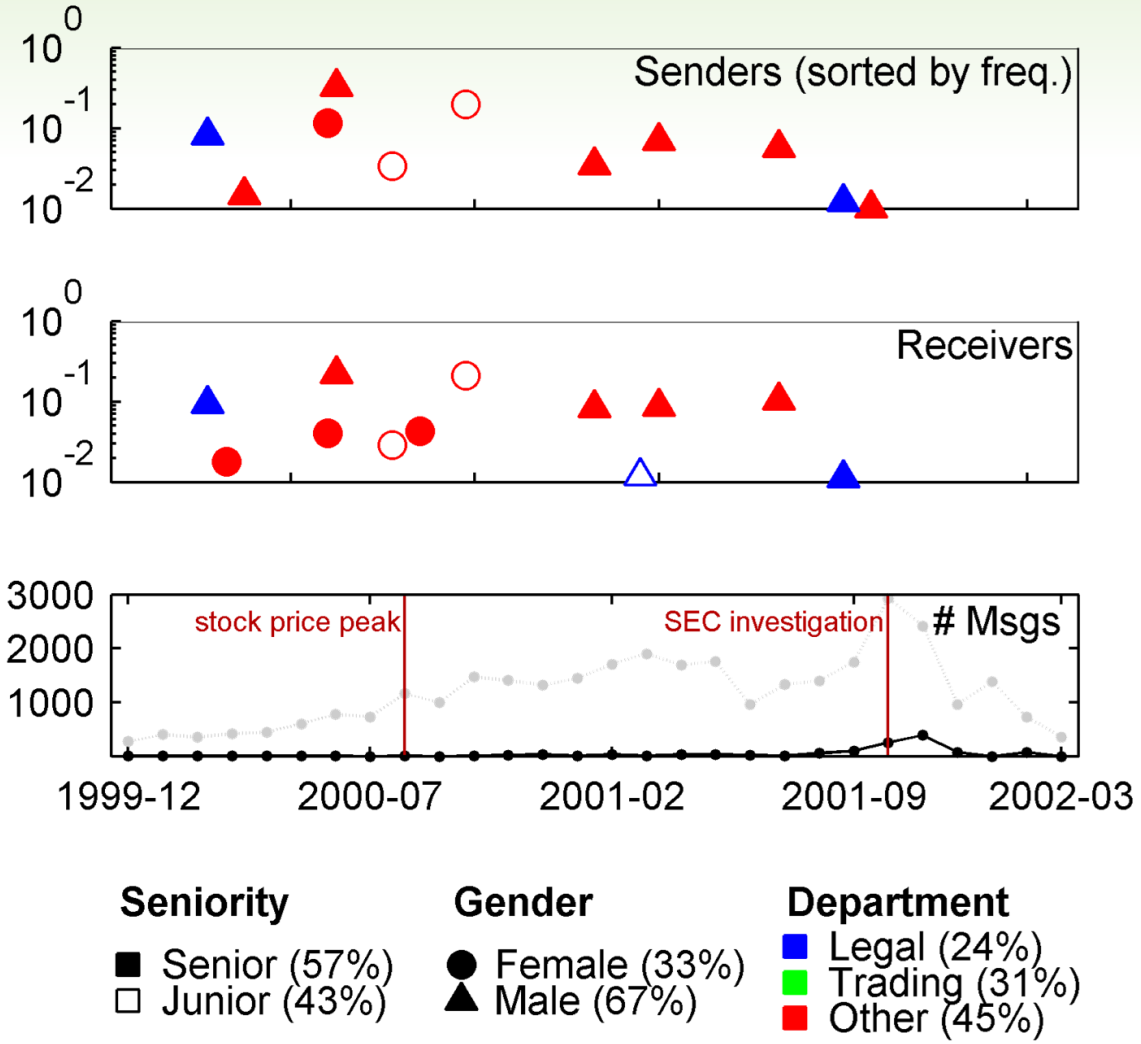


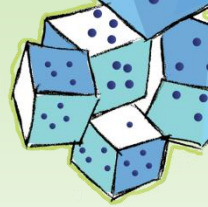


Enron Email Data (Component 10)

Mostly Other







Wrapping up Lectures

Illustration by Chris Brigman

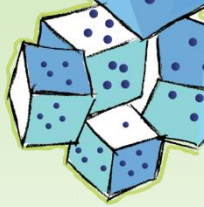
Acknowledgements

Tensor Toolbox for MATLAB:
https://gitlab.com/tensors/tensor_toolbox
Bader, Kolda, Acar, Dunlavy, and others

Current Collaborators

- Cliff Anderson-Bergman (Sandia)
- Gavin Baker (Sandia)
- Grey Ballard (Sandia/Wake Forest)
- Casey Battaglino (Sandia/GA Tech)
- Karen Devine (Sandia)
- Jed Duersch (Sandia)
- Danny Dunlavy (Sandia)
- Chris Forster (Sandia)
- Alex Gorodetsky (Sandia/Michigan)
- David Hong (Sandia/Michigan)
- Alicia Klinvex (Sandia)
- Hemanth Kolla (Sandia)
- Jiajia Li (GA Tech)
- Eric Phipps (Sandia)
- Prashant Rai (Sandia)
- Keita Teranishi
- Rich Vuduc (GA Tech)
- Alex Williams (Sandia/Stanford)
- Kina Winoto (Sandia)
- Jeff Young (GA Tech)
- Plus many more previous collaborators!

What We Covered

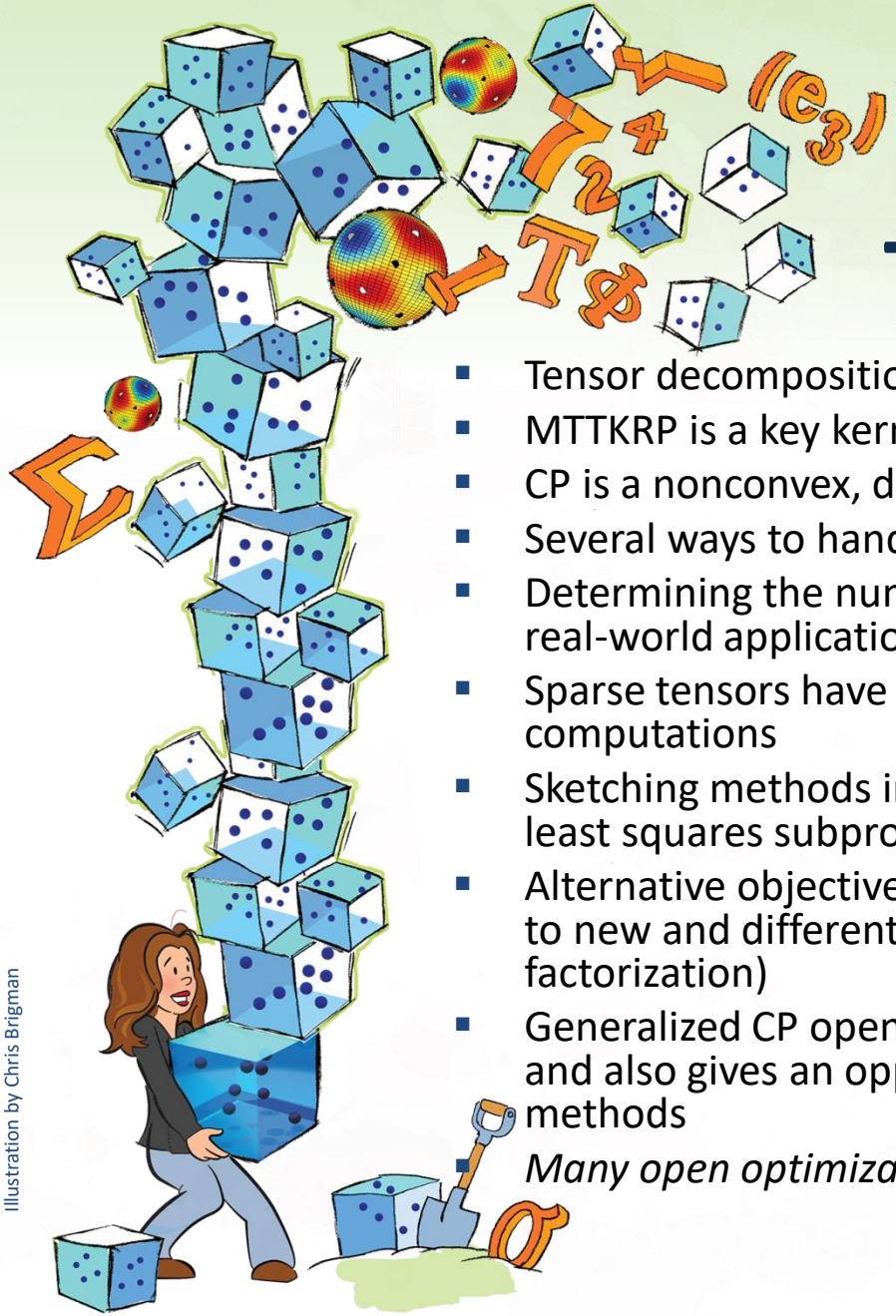


What we covered

- Tensors & Their Decompositions
- Background
- Intro to CP Decomposition
- Motivations for CP
- Computing CP
- Missing data
- Choosing the number of components
- Large-scale data
- Randomized CP-ALS
- Generalized CP
- Poisson tensor factorization

Some of what we didn't cover

- *From my life*
  - Coupled factorizations
  - Tensor eigenproblems
  - Symmetric tensor decomposition
  - Functional (i.e., continuous) tensor decomposition
  - Tucker tensor decomposition for data compression
- From elsewhere
  - Tensor train and hierarchical tensor decomposition
  - Tensor completion & tensor nuclear norm



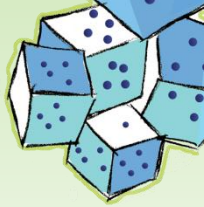
Takeaways

- Tensor decompositions have many applications
- MTTKRP is a key kernel
- CP is a nonconvex, difficult optimization problem
- Several ways to handle missing data; recommend *ignoring* it
- Determining the number of components is NP-hard but critical for real-world applications
- Sparse tensors have special structure that can be exploited in computations
- Sketching methods in linear algebra can be used to speed up the least squares subproblems in CP
- Alternative objective functions have merit for applications and lead to new and different optimization challenges (see Poisson tensor factorization)
- Generalized CP opens the door to many different objective functions and also gives an opportunity to use stochastic optimization methods
- *Many open optimization problems in this area!*

References

- **Overall review:** T. G. Kolda and B. W. Bader, ***Tensor Decompositions and Applications***, SIAM Review, 2009, doi:10.1137/07070111X
- **Original MATLAB paper:** B. W. Bader and T. G. Kolda, ***Algorithm 862: MATLAB Tensor Classes for Fast Algorithm Prototyping***, ACM Transactions on Mathematical Software, 2006, doi:10.1145/1186785.1186794
- **Special MATLAB classes:** B. W. Bader and T. G. Kolda, ***Efficient MATLAB Computations with Sparse and Factored Tensors***, SIAM Journal on Scientific Computing, 2007, doi:10.1137/060676489
- **Application to Mouse Data:** A. Williams, T. G. Kolda, S. Ganguli, et al., ***Unsupervised discovery of low-dimensional neural dynamics both within and across single trials through tensor analysis***, 2017 (coming soon)
- **Application to Link Prediction:** D. M. Dunlavy, T. G. Kolda and E. Acar, ***Temporal Link Prediction using Matrix and Tensor Factorizations***, ACM Transactions on Knowledge Discovery from Data, 2011, doi:10.1145/1921632.1921636
- **All-at-once Optimization Approach:** E. Acar, D. M. Dunlavy and T. G. Kolda, ***A Scalable Optimization Approach for Fitting Canonical Tensor Decompositions***, Journal of Chemometrics, 2011, doi:10.1002/cem.1335
- **Missing data:** E. Acar, D. M. Dunlavy, T. G. Kolda, M. Mø and rup, ***Scalable Tensor Factorizations for Incomplete Data***, Chemometrics and Intelligent Laboratory Systems, 2011, doi:10.1016/j.chemolab.2010.08.004
- **More missing data:** E. Acar, D. M. Dunlavy, T. G. Kolda, M. Mø and rup, ***Scalable Tensor Factorizations with Missing Data***, SDM10: Proceedings of the 2010 SIAM International Conference on Data Mining, 2010, doi:10.1137/1.9781611972801.61
- **Randomized CP-ALS:** C. Battaglino, G. Ballard and T. G. Kolda, ***A Practical Randomized CP Tensor Decomposition***, January 2017, arXiv:1701.06600
- **Poisson tensor factorization:** E. C. Chi and T. G. Kolda, ***On Tensors, Sparsity, and Nonnegative Factorizations***, SIAM Journal on Matrix Analysis and Applications, 2012, doi:10.1137/110859063
- **More Poisson tensor factorization:** S. Hansen, T. Plantenga and T. G. Kolda, ***Newton-Based Optimization for Kullback-Leibler Nonnegative Tensor Factorizations***, Optimization Methods and Software, 2015, doi:10.1080/10556788.2015.1009977
- **Generalized CP:** Cliff Anderson-Bergman, D. Hong, T. G. Kolda, ***Generalized Canonical Polyadic Tensor Decomposition***, 2017 (coming soon)

Contact: Tammy Kolda, tgkolda@sandia.gov



Bonus Materials

Time Permitting

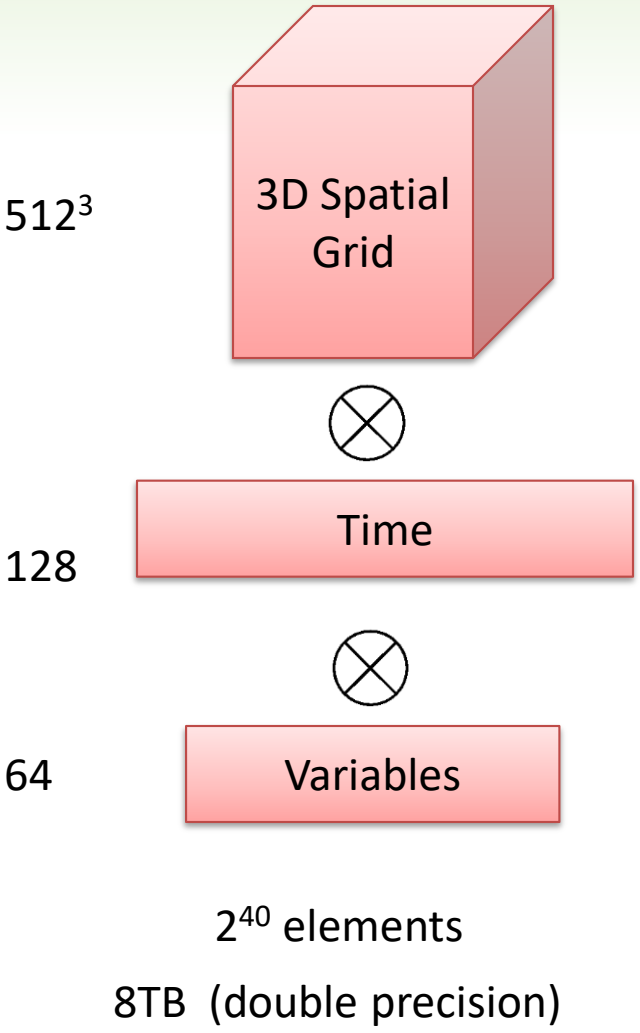
Sandia National Laboratories




Parallel Tucker Decomposition for Data Compression

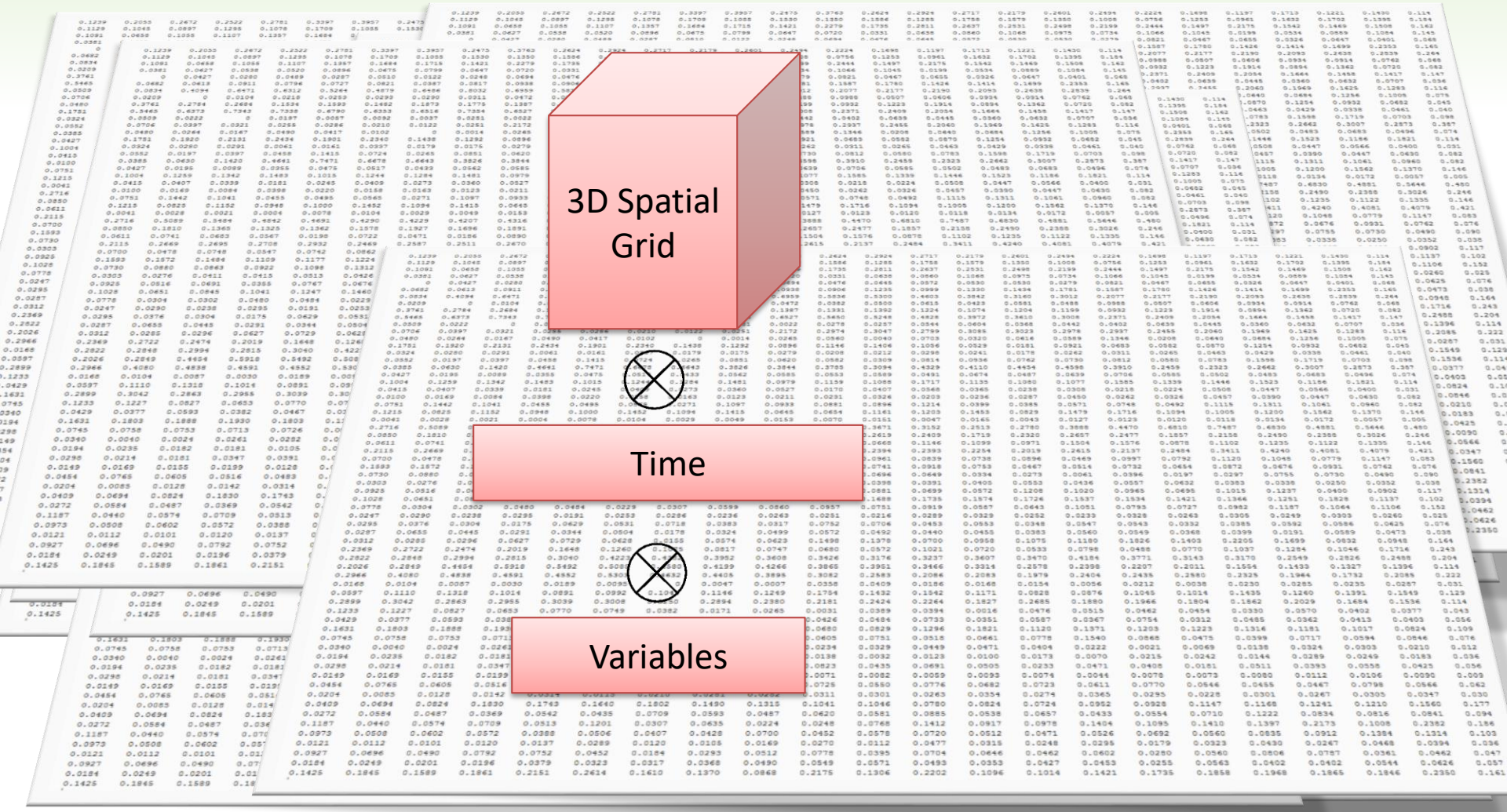
Featuring work of Woody Austin (Univ. Texas, Austin) , Grey Ballard (Wake Forest), Alicia Klinvex (Sandia), Hemanth Kolla (Sandia)

DOE Advanced Simulations and Experiments Deluged by Data

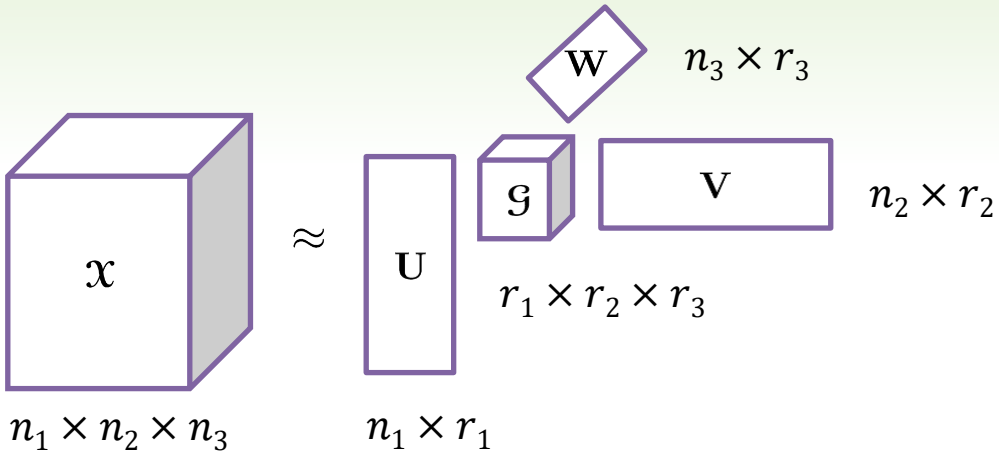


- Combustion simulations
  - S3D used direct numerical simulation
  - Gold standard for comparisons, but…
  - Single experiment produces terabytes of data
  - Storage limits spatial, temporal resolutions
  - Difficult to analyze or transfer data
- Other applications
  - Electron Microscopy Experiments
  - Telemetry Experiments
  - Cosmology Simulations
  - Climate Modeling

How to compress so much raw data but preserve accuracy????



Tucker Compression (3-way)



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

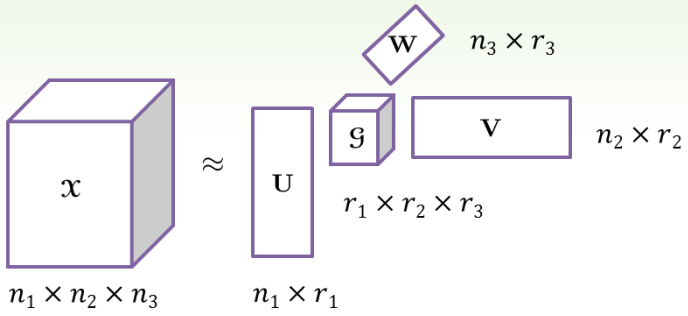
$\mathcal{G}$ = "Core Tensor" = Reduced representation, determined by factor matrices

$\mathbf{U}, \mathbf{V}, \mathbf{W}$ = "Factor Matrices" = Orthogonal matrices spanning high-variance subspaces

$\times_k$ = Tensor-times-matrix in mode k

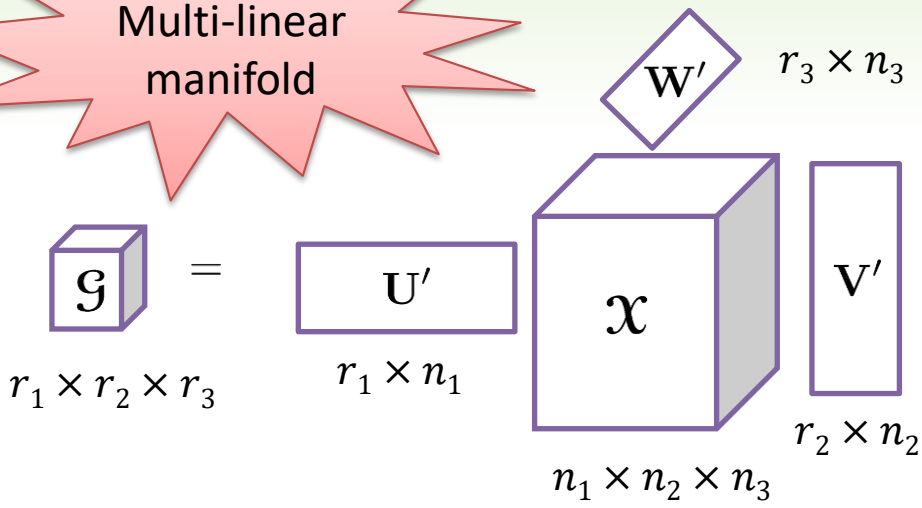
- Hitchcock (1927) – Mathematical definition
- Tucker (1966) – Algorithms and applications for 3-way data
- Kapteyn, Neudecker, Wansbeek (1986) – Algorithms for N-way data
- Vannieuwenhoven, Vandebril, and Meerbergen (2012) – Sequentially-truncated algorithm

Choosing Tucker Ranks to Retain Accuracy



Find orthogonal matrices **U**, **V**, **W** that reduce the size of tensor but retain its "mass"

Multi-linear manifold



For a given relative error ϵ choose projection ranks r_1, r_2, and r_3 such that:

$$\|\mathcal{X} - (\mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W})\| \leq \epsilon \|\mathcal{X}\|$$

Core tensor satisfies: $\mathcal{G} = \mathcal{X} \times_1 \mathbf{U}' \times_2 \mathbf{V}' \times_3 \mathbf{W}'$

$$\|\mathcal{X}\|^2 - \|\mathcal{G}\|^2 \leq \epsilon^2 \|\mathcal{X}\|^2$$

Tucker Compression (d-way)

$$\underbrace{\mathcal{X}}_{n_1 \times n_2 \cdots \times n_d} \approx \underbrace{\mathcal{G}}_{r_1 \times r_2 \cdots \times r_d} \times_1 \underbrace{\mathbf{U}_1}_{n_1 \times r_1} \times_2 \underbrace{\mathbf{U}_2}_{n_2 \times r_2} \cdots \times_d \underbrace{\mathbf{U}_d}_{n_d \times r_d}$$

$\mathcal{G}$ = "Core Tensor" = Reduced representation, determined by factor matrices

$\mathbf{U}^{(k)}$ = kth "Factor Matrix" = Orthogonal matrix spanning high-variance subspaces

$\times_k$ = Tensor-times-matrix in mode k

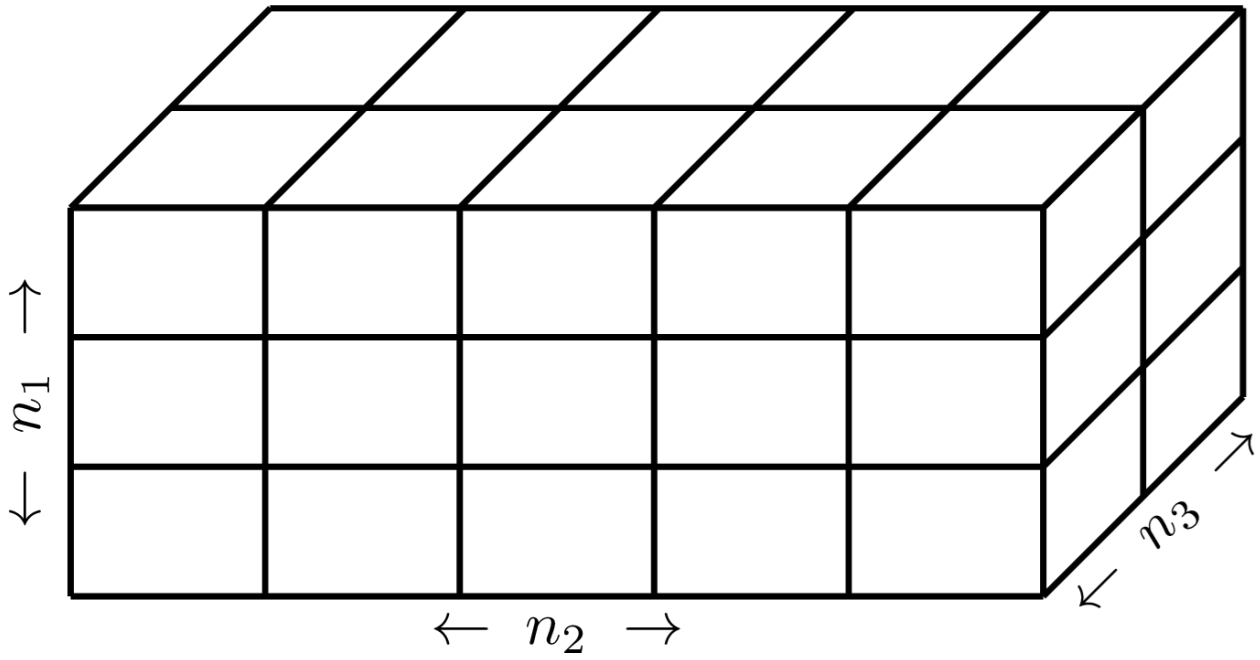
Compression Ratio: $C \approx \prod_{k=1}^{d} \frac{n_k}{r_k}$

New Contribution: Parallel Tucker Decomposition

Sandia National Laboratories

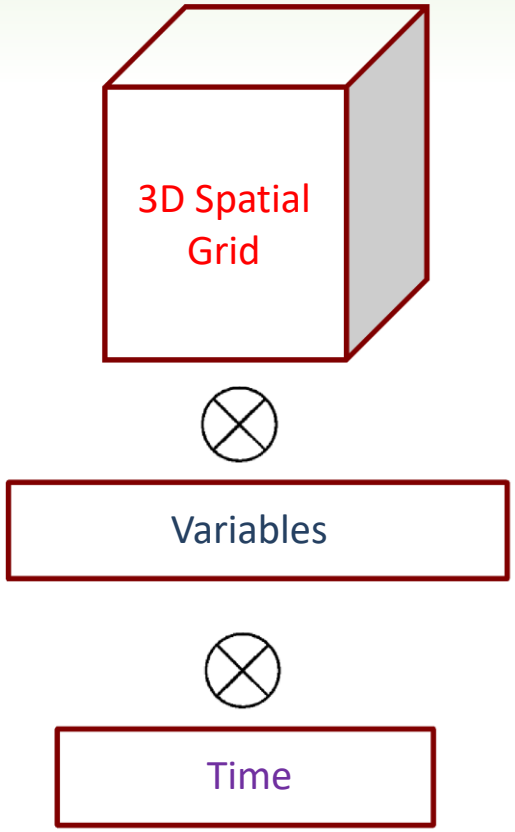
For n-way tensor, Cartesian block distribution on n-way processor grid

$$p_1 \times p_2 \times p_3 = 3 \times 5 \times 2$$



Local block size: $\frac{n_1}{p_1} \times \frac{n_2}{p_2} \times \frac{n_3}{p_3}$

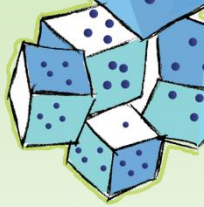
Up to 200,000X compression

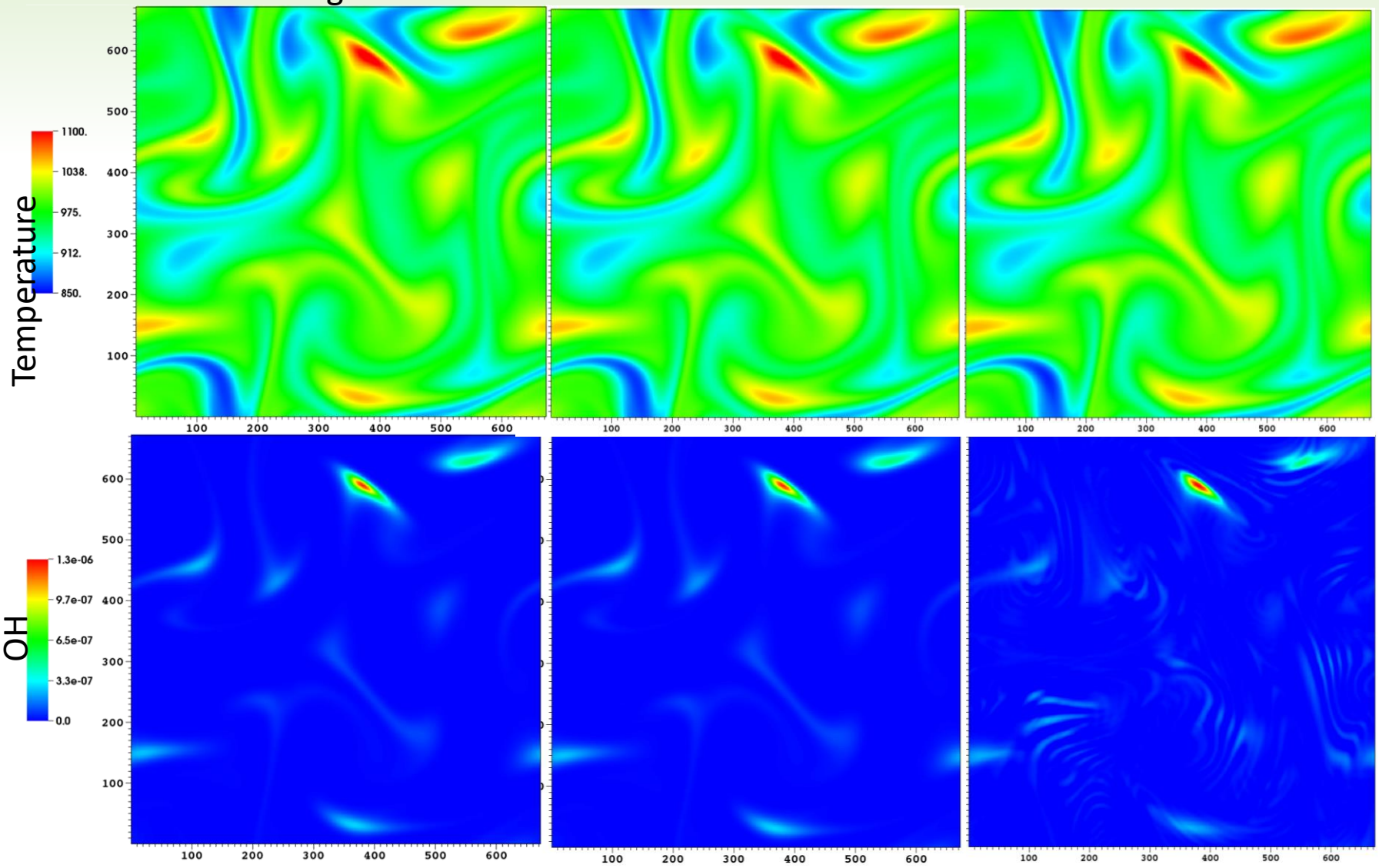


	Original	$\epsilon = 10^{-4}$	$\epsilon = 10^{-2}$
HCCI	672 x 672 x 32 x 626 **67 GB**	330 x 310 x 31 x 199 **(14 X)**	111 x 105 x 22 x 46 **(760 X)**
SP	500 x 500 x 500 x 11 x 400 **4 TB**	95 x 129 x 125 x 7 x 125 **(410 X)**	30 x 38 x 35 x 6 x 11 **(200,000 X)**
JICF	1500 x 2080 x 1500 x 18 x 10 **6 TB**	424 x 387 x 261 x 18 x 10 **(110 X)**	90 x 61 x 48 x 13 x 6 **(40,000 X)**

HCCI 67 GB Combustion Simulation : Temperature & OH

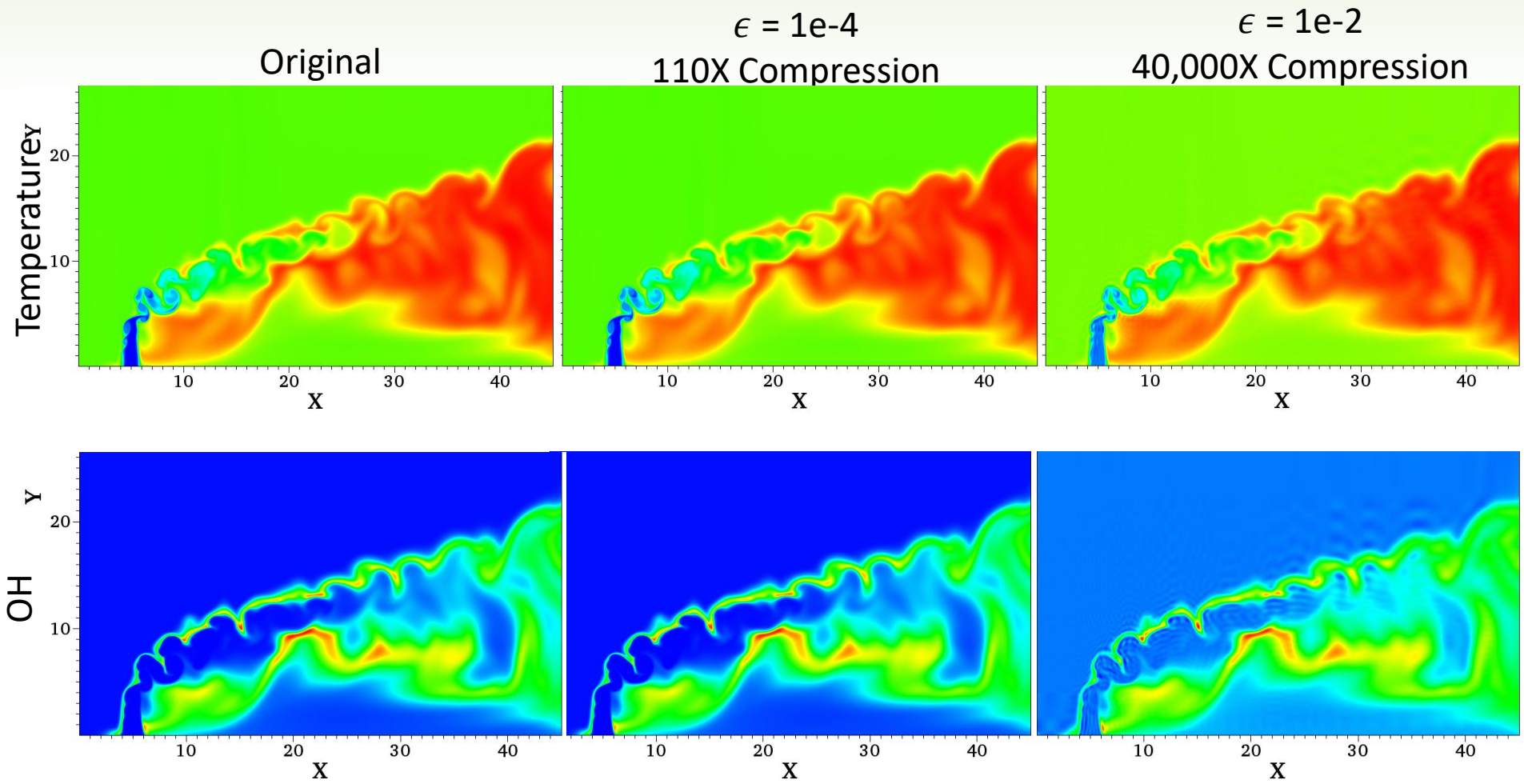
Sandia National Laboratories





6 TB Combustion Simulation Temperature & OH

Sandia National Laboratories



Single timestep; fixed Z coordinate

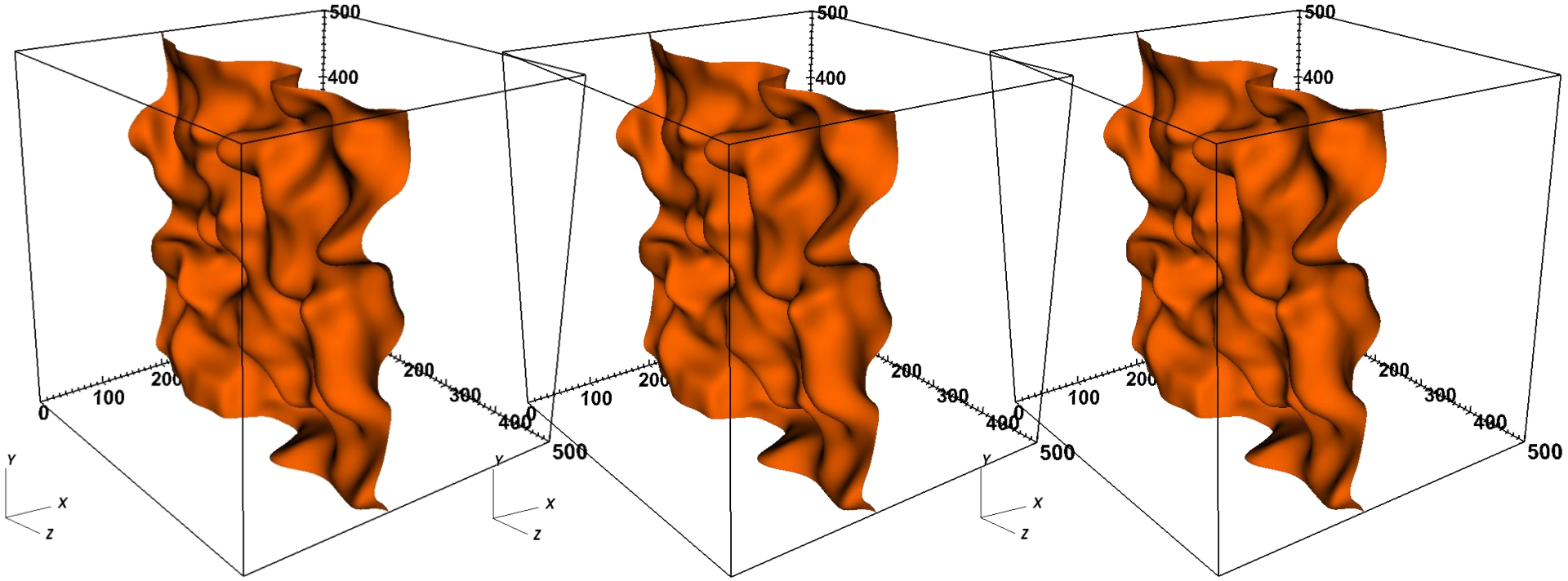
4 TB Combustion Simulation: Temperature Iso-surface

Sandia National Laboratories

Original

ϵ^2 = 1e-4

ϵ^2 = 1e-2



410X Compression

200,000X Compression

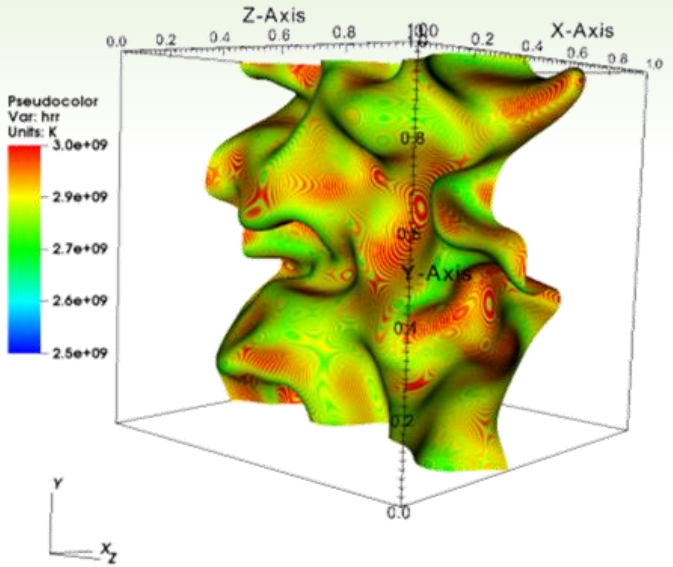
Flame surface at single time step. Using temperature variable (iso-value is 2/3 of max).

Key Feature: Need Only Do Partial Reconstruction on Laptops, etc.

Sandia National Laboratories

Reconstruction requires as much space as the original data!

$$\underset{n_1 \times n_2 \times n_3 \times n_4 \times n_5}{\hat{\mathcal{X}}} = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3 \times_4 \mathbf{U}_4 \times_5 \mathbf{U}_5$$



But we can just reconstruct the portion that we need at the moment:

$$\underset{n_1 \times n_2 \times \frac{n_3}{2} \times 1 \times 1}{\bar{\mathcal{X}}} = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{C}_3\mathbf{U}_3 \times_4 \mathbf{C}_4\mathbf{U}_4 \times_5 \mathbf{C}_5\mathbf{U}_5$$

$$\mathbf{C}_3 \begin{bmatrix} 1/2 & 0 & \cdots & 0 \\ 1/2 & 0 & \cdots & 0 \\ 0 & 1/2 & \cdots & 0 \\ 0 & 1/2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix}$$

Downsample

$$\mathbf{C}_4 = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{bmatrix}$$

Pick single variable

$$\mathbf{C}_5 = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{bmatrix}$$

Pick single time step

TuckerMPI Parallel Performance

`git@gitlab.com:tensors/TuckerMPI.git`

Alicia Klinvex, Woody Austin, Grey Ballard, Hemanth Kolla, Tammy Kolda

Compression Time

- 4.4 TB → 10G GB (410X)
- 20 sec. on 300 nodes x 16 cores/node = 4800 cores (Sandia Sky Bridge Cluster w/ 2.6 GHz Intel Sandy Bridge)
- For comparison, read time for the 4.4 TB onto 300 nodes is 257 sec.

Reconstruction Time

- Reconstruct 3D mesh for single time and variable (500 x 500 x 500 x 1 x 1 = 1GB)
- 4 sec. on *single* node (no parallelism) with 64 GB RAM
- For comparison, read time of 10GB compressed data onto one node is 16 sec.

Full paper with Alicia Klinvex and Grey Ballard coming soon.

Current work (with Casey Battaglino, Grey Ballard): Randomized version of Tucker!