Tensor Factorizations & Applications

Evrim Acar

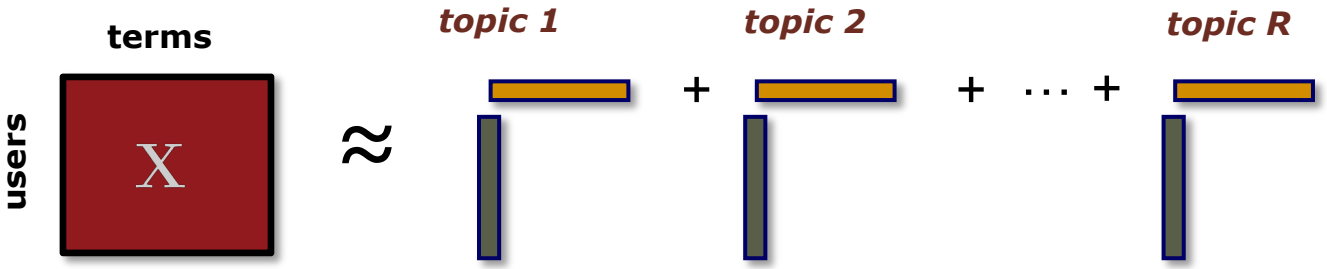
Simula Metropolitan Center for Digital Engineering

Oslo, Norway

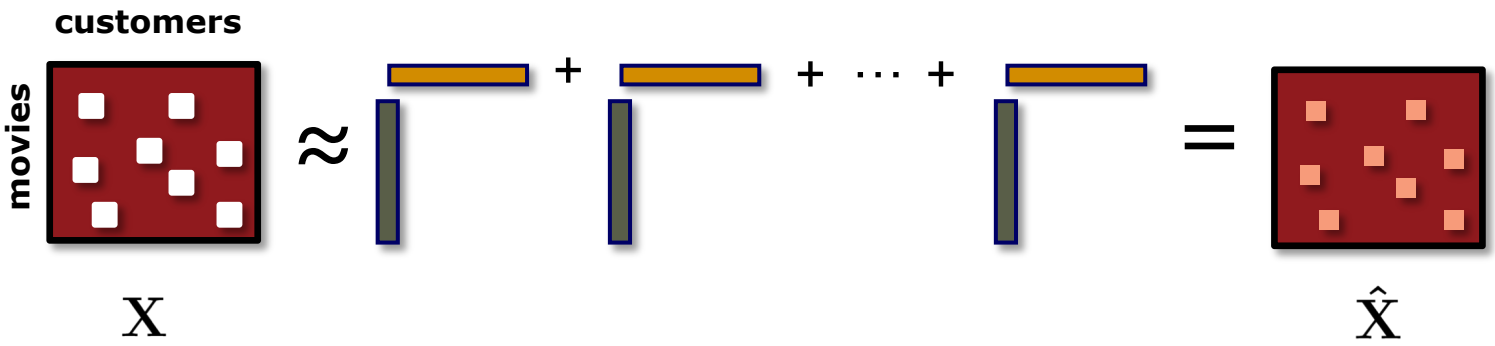
simula

Matrix Factorizations in Data Mining

Capturing underlying hidden factors/patterns



Incomplete Matrix Factorization/ Matrix Completion



simula

Data is often squeezed to be two-way!

2-way

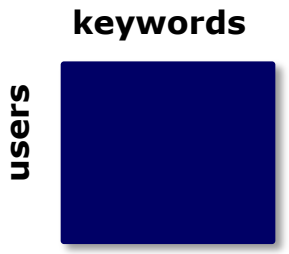
Social networks : **<users, keywords>**
<employee, employee>
<users, users>

Text mining: **< documents, terms>**

Recommender Systems: **<customers, items>**

Computer vision: **<people, pixels>**

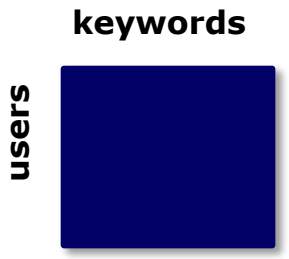
Neuroscience: **<electrodes, time>**

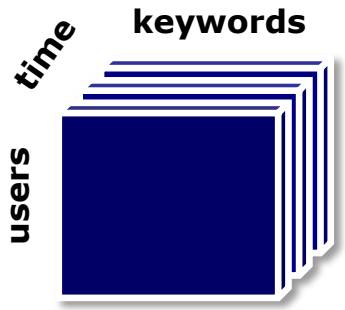


simula

Data is often squeezed to be two-way!

	2-way	Multi-way
Social networks :	**<users, keywords>** **<employee, employee>** **<users, users>**	**< users, keywords, time>** **< employee, employee, time>** **< users, users, communication type>**
Text mining:	**< documents, terms>**	**< documents, terms, terms>**
Recommender Systems:	**<customers, items>**	**<customers, items, tags>**
Computer vision:	**<people, pixels>**	**<people, pixels, viewpoints>**
Neuroscience:	**<electrodes, time>**	**<electrodes, time, frequency>** **<electrodes, time, subjects>**

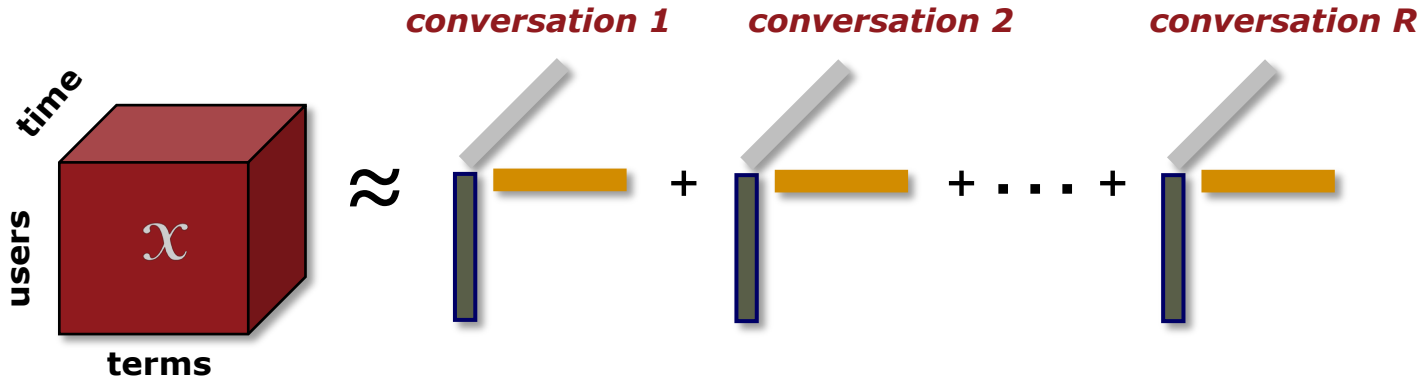




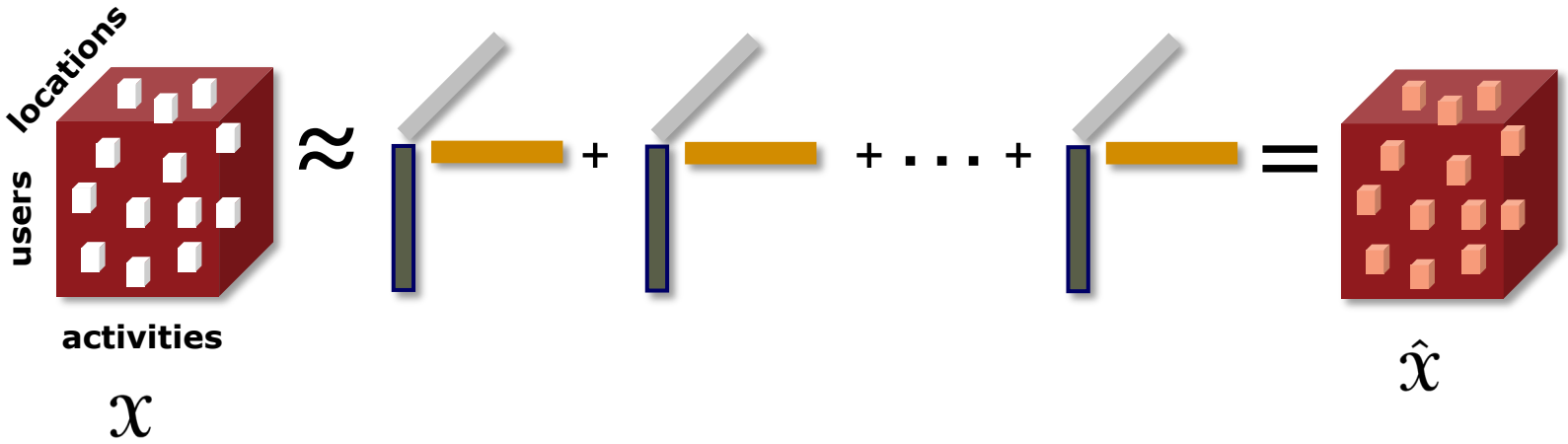
simula

Data is often multi-way!

Capturing underlying hidden factors/patterns



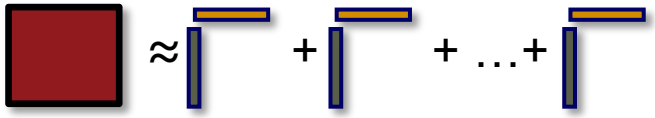
Incomplete Tensor Factorization/ Tensor Completion



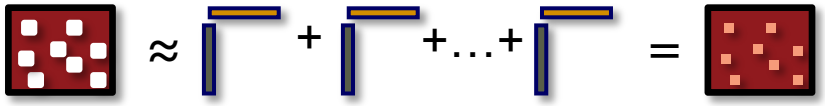
simula

Matrix Factorizations in Data Mining

Matrix Factorizations

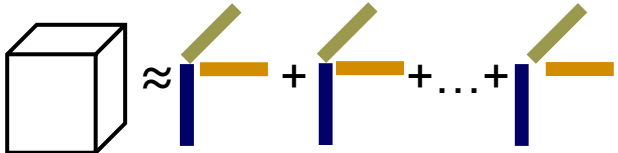


Matrix Completion

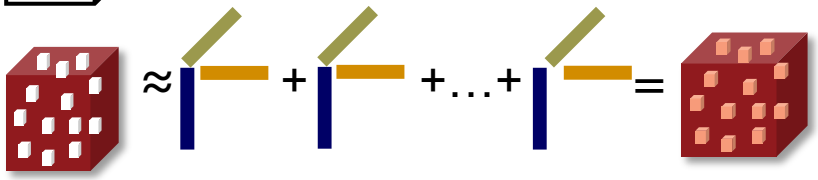


Data sets are often multi-way: Tensor Factorizations

Tensor Factorizations



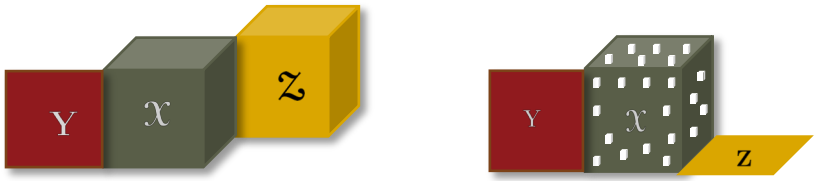
Tensor Completion



Data sets often come from multiple sources: Data Fusion

Coupled Matrix Factorizations

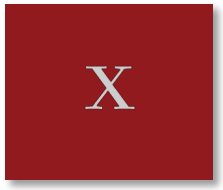
Coupled Tensor Factorizations



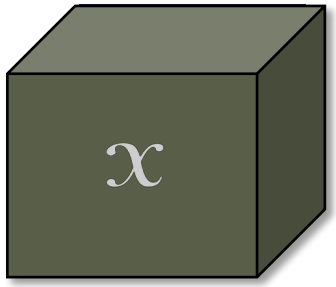
Tensors: Terminology

Order: Number of Modes or Ways

first-order tensor (vector)

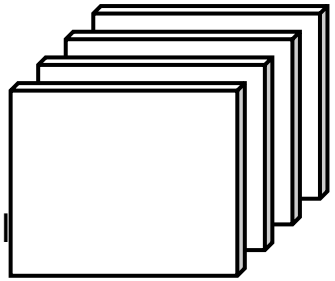


second-order tensor (matrix)

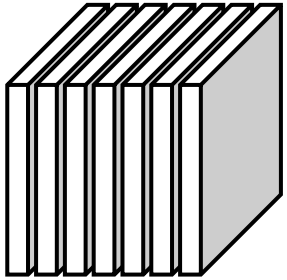


third-order tensor (three-way array)

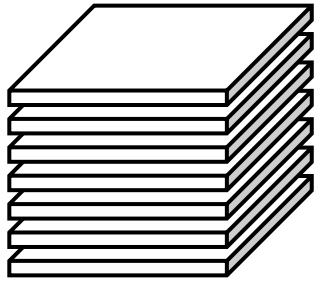
higher-order tensors (multi-way arrays)



Frontal slices

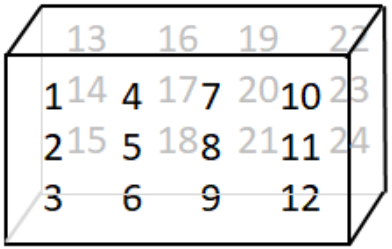


Vertical slices



Horizontal slices

$$\mathcal{X} =$$



Order: 3

Size: $3 \times 4 \times 2$

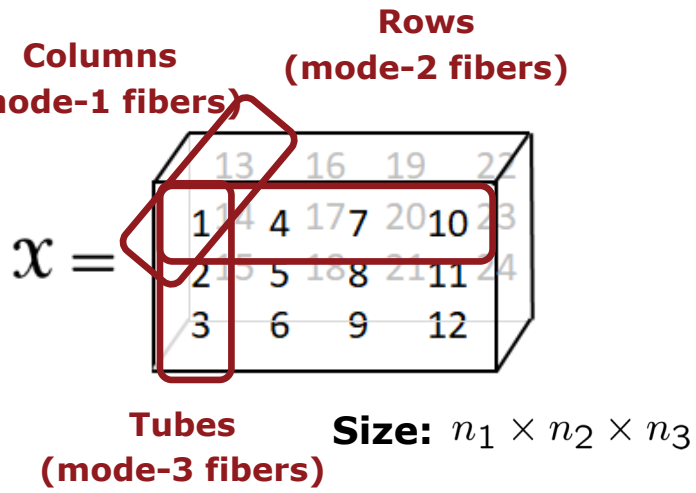
$$\mathbf{X}(:,:,1) = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}$$

Frontal slices

$$\mathbf{X}(:,:,2) = \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix}$$

Tensors: Basic Operations

Matricization (a.k.a. Unfolding or Flattening)



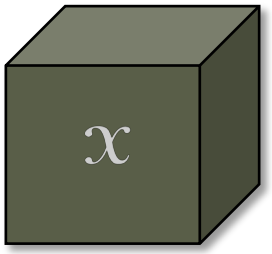
Size: $n_1 \times n_2 \times n_3$

$$\underset{n_1 \times n_2 n_3}{\mathbf{X}_{(1)}} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & \dots & 22 \\ 2 & 5 & 8 & 11 & 14 & \dots & 23 \\ 3 & 6 & 9 & 12 & 15 & \dots & 24 \end{bmatrix}$$

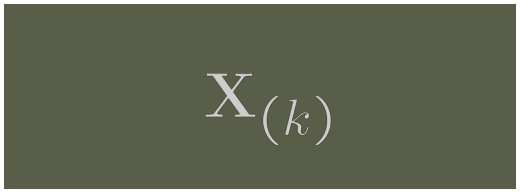
$$\underset{n_2 \times n_1 n_3}{\mathbf{X}_{(2)}} = \begin{bmatrix} 1 & 2 & 3 & 13 & \dots & 15 \\ 4 & 5 & 6 & 16 & \dots & 18 \\ 7 & 8 & 9 & 19 & \dots & 21 \\ 10 & 11 & 12 & 22 & \dots & 24 \end{bmatrix}$$

$$\underset{n_3 \times n_1 n_2}{\mathbf{X}_{(3)}} = \begin{bmatrix} 1 & 2 & 3 & 4 & \dots & 12 \\ 13 & 14 & 15 & 16 & \dots & 24 \end{bmatrix}$$

Unfolding a *d*-way array (mode-*k* unfolding)



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



$$n_k \times \prod_{i, i \neq k}^{d} n_i$$

simula

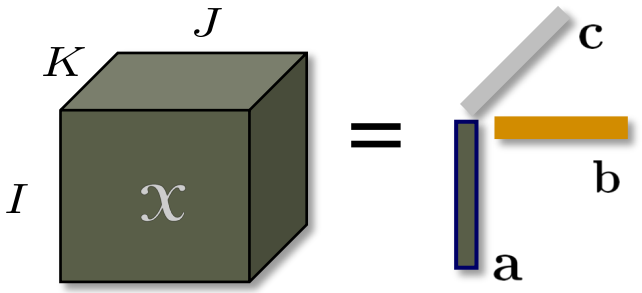
Tensors: Basic Operations (cont.)

Vector outer product $\mathbf{a} \in \mathbb{R}^I, \mathbf{b} \in \mathbb{R}^J, \mathbf{c} \in \mathbb{R}^K$

$$\mathbf{X} = \mathbf{a}\mathbf{b}^\mathsf{T} = \mathbf{a} \circ \mathbf{b}$$

$$\mathbf{X} \in \mathbb{R}^{I \times J}, x_{ij} = a_i b_j$$

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



$$\mathcal{X} \in \mathbb{R}^{I \times J \times K}, x_{ijk} = a_i b_j c_k$$

Kronecker product

$\mathbf{a} \in \mathbb{R}^I, \mathbf{b} \in \mathbb{R}^J, \mathbf{A} \in \mathbb{R}^{I \times M}, \mathbf{B} \in \mathbb{R}^{J \times N}$

$$\mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} a_1\mathbf{b} \\ a_2\mathbf{b} \\ \vdots \\ a_I\mathbf{b} \end{bmatrix} \in \mathbb{R}^{IJ}$$

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \dots & a_{1M}\mathbf{B} \\ a_{21}\mathbf{B} & \dots & a_{2M}\mathbf{B} \\ \vdots & & \vdots \\ a_{I1}\mathbf{B} & \dots & a_{IM}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{IJ \times MN}$$

Khatri-Rao product (Columnwise Kronecker Product)

$\mathbf{A} \in \mathbb{R}^{I \times R}, \mathbf{B} \in \mathbb{R}^{J \times R}$

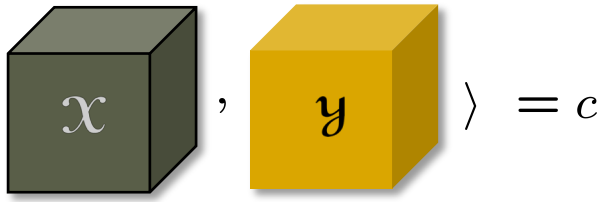
$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \dots & \mathbf{a}_R \otimes \mathbf{b}_R \end{bmatrix} \in \mathbb{R}^{IJ \times R}$$

simula

Tensors: Basic Operations (cont.)

Tensor Inner Product $\mathcal{X} \in \mathbb{R}^{I \times J \times K}, \mathcal{Y} \in \mathbb{R}^{I \times J \times K}$

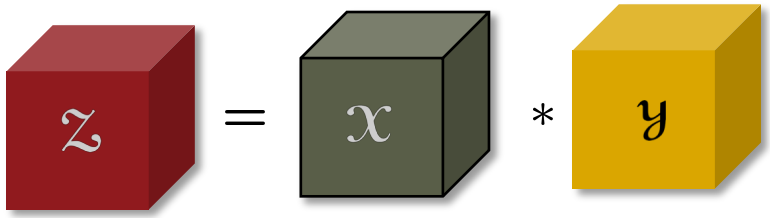
$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{k=1}^{K} \sum_{j=1}^{J} \sum_{i=1}^{I} x_{ijk} y_{ijk}$$



Frobenius Norm

$$\| \mathcal{X} \|^2 = \langle \mathcal{X}, \mathcal{X} \rangle = \sum_{k=1}^{K} \sum_{j=1}^{J} \sum_{i=1}^{I} x_{ijk}^2$$

Hadamard Product $\mathcal{X}, \mathcal{Y}, \mathcal{Z} \in \mathbb{R}^{I \times J \times K}$



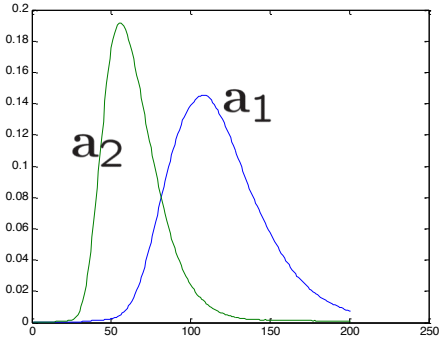
$$z_{ijk} = x_{ijk} y_{ijk}$$

simula

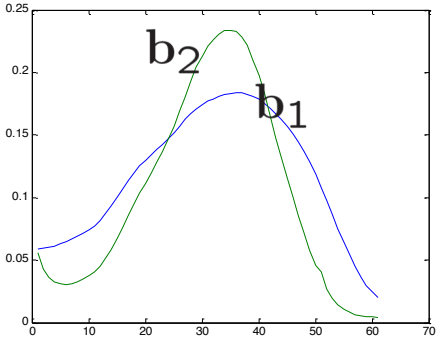
What if we have multiple matrices with the same underlying factors but in different proportions...

True factors

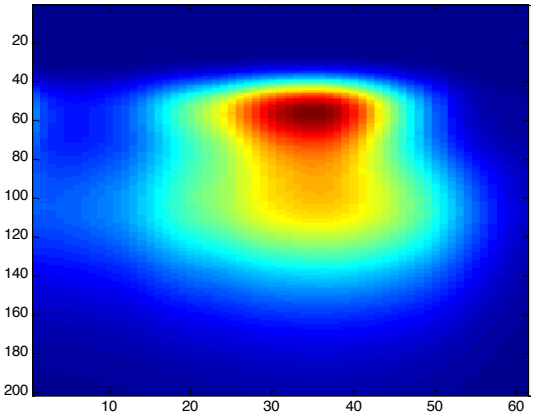
$$\mathbf{A} \in \mathbb{R}^{201 \times 2} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 \end{bmatrix}$$



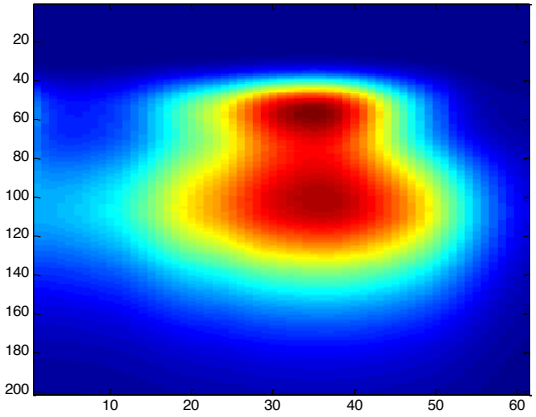
$$\mathbf{B} \in \mathbb{R}^{61 \times 2} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 \end{bmatrix}$$



$$\mathbf{X}_1 = \mathbf{A} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{B}^\mathsf{T}$$



$$\mathbf{X}_2 = \mathbf{A} \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \mathbf{B}^\mathsf{T}$$



We can recover the true factors uniquely up to trivial indeterminacies, i.e., scaling and permutation.

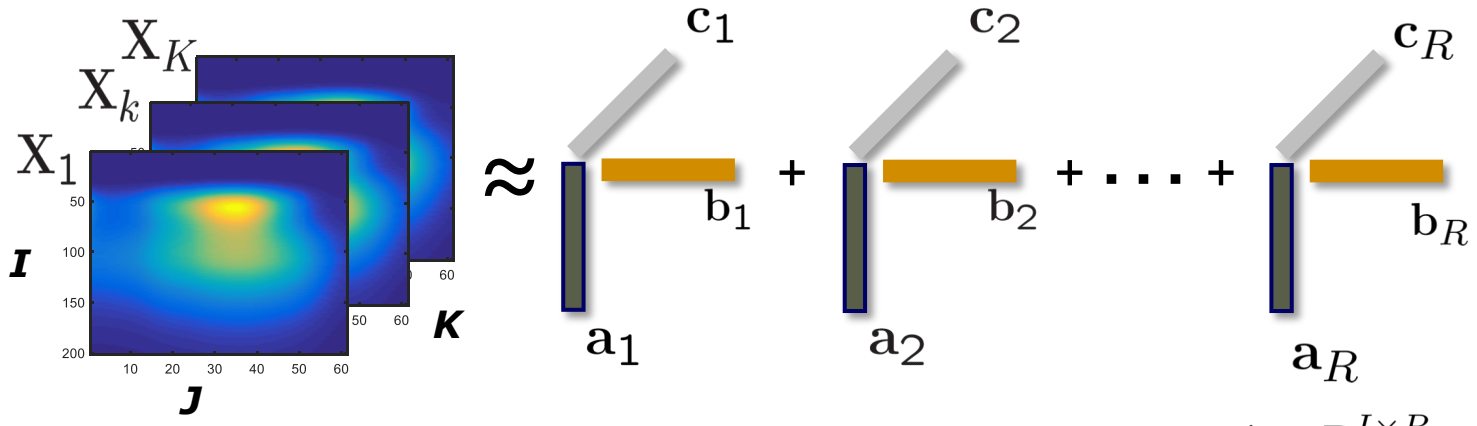
simula

A popular tensor factorization model: CP

Canonical Polyadic (CP)
CANDECOMP/PARAFAC (CP)

Hitchcock, 1927: Polyadic form of a tensor
Harshman, 1970: Parallel Factor Analysis (PARAFAC)
Carroll & Chang, 1970: Canonical Decomposition (CANDECOMP)

As an extension of matrix factorizations to higher-order tensors (multi-way arrays), tensor factorizations are used to extract the underlying factors in higher-order data sets. The CP model represents a tensor as a sum of rank-one tensors:



$$\mathbf{X}_k \approx \mathbf{A} \begin{bmatrix} c_{k1} & & \\ & \cdots & \\ & & c_{kR} \end{bmatrix} \mathbf{B}^\mathsf{T}$$

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \approx [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

$$\mathbf{A} \in \mathbb{R}^{I \times R} = \begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_R \end{bmatrix}$$

$$\mathbf{B} \in \mathbb{R}^{J \times R} = \begin{bmatrix} \mathbf{b}_1 & \cdots & \mathbf{b}_R \end{bmatrix}$$

$$\mathbf{C} \in \mathbb{R}^{K \times R} = \begin{bmatrix} \mathbf{c}_1 & \cdots & \mathbf{c}_R \end{bmatrix}$$

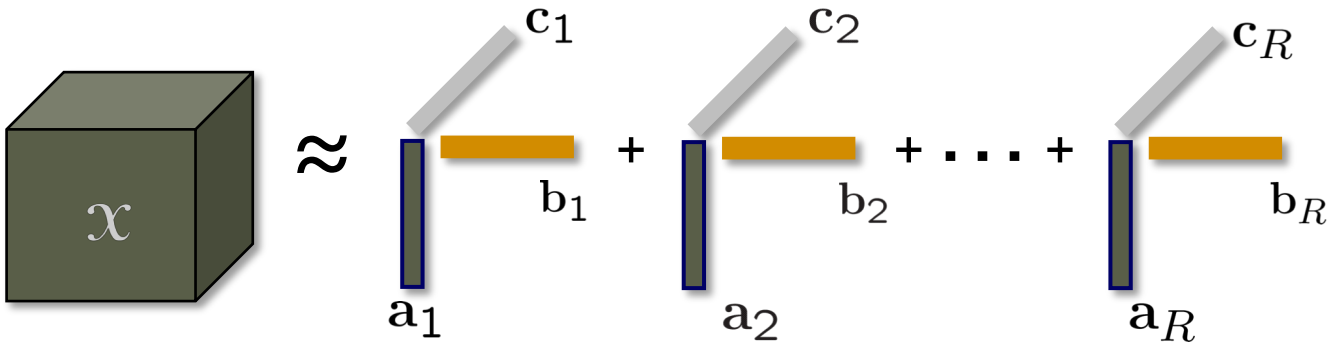
CP is unique up to scaling and permutation under the condition that

$$\text{krank}(\mathbf{A}) + \text{krank}(\mathbf{B}) + \text{krank}(\mathbf{C}) \geq 2R + 2$$

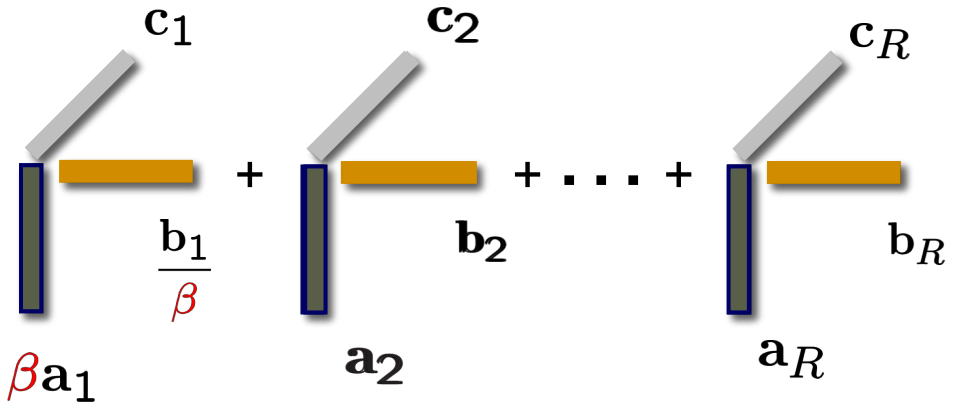
[Kruskal, 1977; Sidiropoulos and Bro, 2000]

where krank (A) = max. value of *k* such that any *k* columns of A is linearly independent.

CP is unique up to trivial indeterminacies



scaling



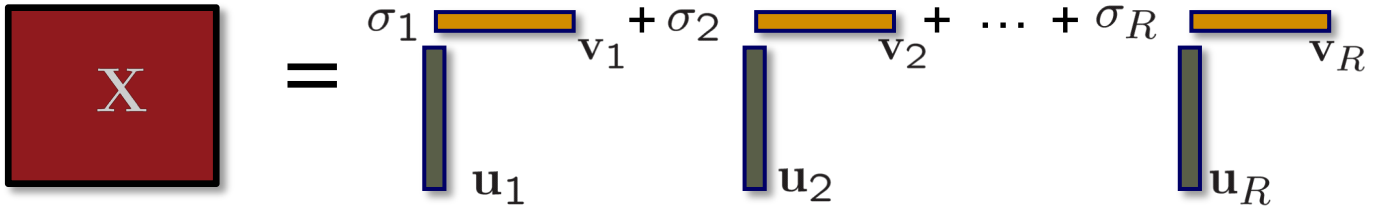
permutation

+ + . . . +

simula

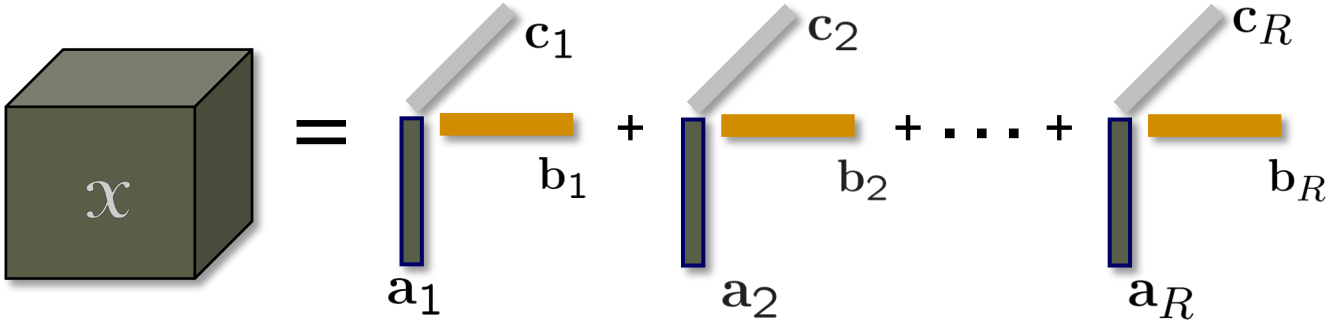
SVD vs. CP

SVD expresses a matrix as the sum of rank-one matrices.



$$\mathbf{X} = \sum_{r=1}^{R} \sigma_r \, \mathbf{u}_r \circ \mathbf{v}_r$$

CP expresses a higher-order tensor as the sum of rank-one tensors.



$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

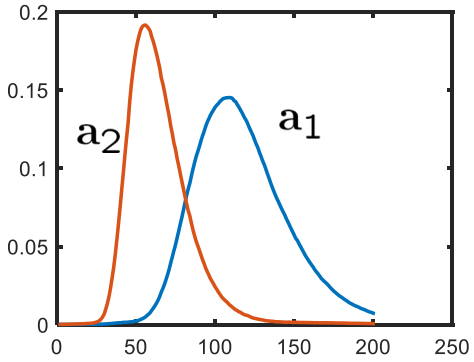
CP is an extension of SVD to higher-order tensors but it is different in many ways.

Does it really work?

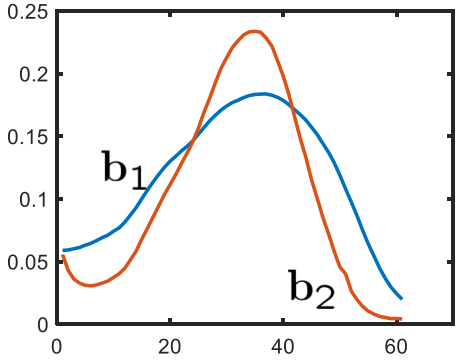
EXAMPLE:

True factors

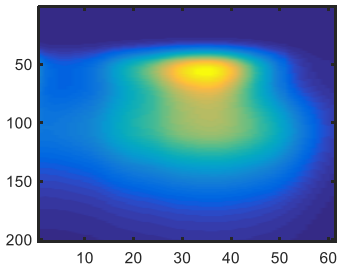
$$\mathbf{A} \in \mathbb{R}^{201\times 2} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 \end{bmatrix}$$

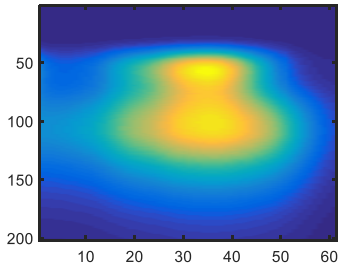


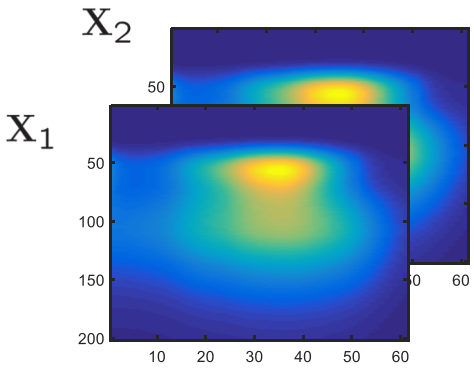
$$\mathbf{B} \in \mathbb{R}^{61\times 2} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 \end{bmatrix}$$



$$\mathbf{X}_1 = \mathbf{A}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\mathbf{B}^\mathsf{T} \qquad \mathbf{X}_2 = \mathbf{A}\begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}\mathbf{B}^\mathsf{T}$$



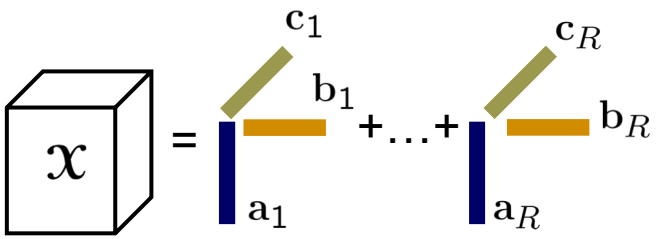




$$\mathcal{X} \approx [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

```
load toydata.mat
XX(:,:,1) =A*[1 0; 0 1]*B';
XX(:,:,2) =A*[3 0; 0 2]*B';
%set optimization parameters
options              = ncg('defaults');
options.DisplayIters = 50;
options.RelFuncTol   = 1e-8;
options.StopTol      = 1e-8;
% call cp_opt to fit the CP model
[Fac,FacInit,out]  = cp_opt(tensor(XX), 2,'init','nvecs','alg_options',options);
subplot(2,1,1);plot(Fac.U{1});
subplot(2,1,2);plot(Fac.U{2});
```

Algorithms for fitting the CP model



The goal is to find matrices **A**, **B**, **C** that solve the following optimization problem:

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

Traditional Approach:
Alternating Least Squares (ALS)
[Harshman, 1970; Carroll & Chang, 1970]

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

Matricized Tensor Times Khatri-Rao Product (MTTKRP)

while "not converged" **do**

Solve for **A** (with fixed **B**, **C**)

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

Solve for **B** (with fixed **A** and **C**)

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

Solve for **C** (with fixed **A** and **B**)

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

end while

However, better convergence properties and accuracy have been achieved using all-at-once optimization approaches:

First-order methods: [Paatero, 1999], CP-OPT [Acar, Dunlavy, Kolda, 2011]

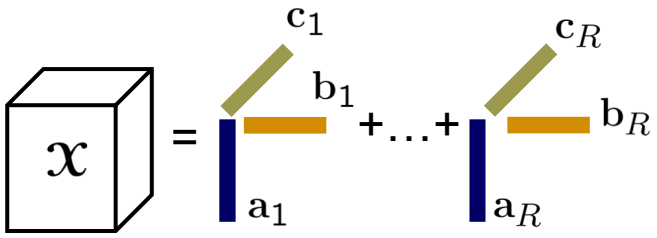
Second-order methods: [Paatero, 1997; Tomasi and Bro, 2006; Phan et al., 2013; Sorber et al., 2013]

CP-OPT: Gradient-based All-at-once Approach

[Acar, Dunlavy, and Kolda, 2011]

CP-OPT is a general gradient–based optimization approach for fitting a CP model.

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



Step1: Define the objective function

$$f(\mathbf{A},\mathbf{B},\mathbf{C}) = \frac{1}{2} \| \mathcal{X} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!] \|^2$$

Step2: Compute the gradient ***Remember the ALS steps!***

$$\frac{\partial f}{\partial \mathbf{A}} = -\mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B}) + \mathbf{A}(\mathbf{C}^\mathsf{T}\mathbf{C} * \mathbf{B}^\mathsf{T}\mathbf{B})$$

$$\frac{\partial f}{\partial \mathbf{B}} = -\mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A}) + \mathbf{B}(\mathbf{C}^\mathsf{T}\mathbf{C} * \mathbf{A}^\mathsf{T}\mathbf{A})$$

$$\frac{\partial f}{\partial \mathbf{C}} = -\mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A}) + \mathbf{C}(\mathbf{B}^\mathsf{T}\mathbf{B} * \mathbf{A}^\mathsf{T}\mathbf{A})$$

Vectorize and concatenate the partials

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial \mathbf{a}_1} \\ \vdots \\ \frac{\partial f}{\partial \mathbf{a}_R} \\ \frac{\partial f}{\partial \mathbf{b}_1} \\ \vdots \\ \frac{\partial f}{\partial \mathbf{b}_R} \\ \frac{\partial f}{\partial \mathbf{c}_1} \\ \vdots \\ \frac{\partial f}{\partial \mathbf{c}_R} \end{bmatrix}$$

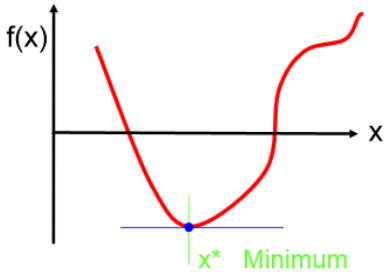
Step3: Pick a first-order optimization method

e.g., Nonlinear Conjugate Gradient (NCG) from the **Poblano Toolbox** [Dunlavy, Kolda and Acar, 2010]

Few words on optimization

Optimization problem

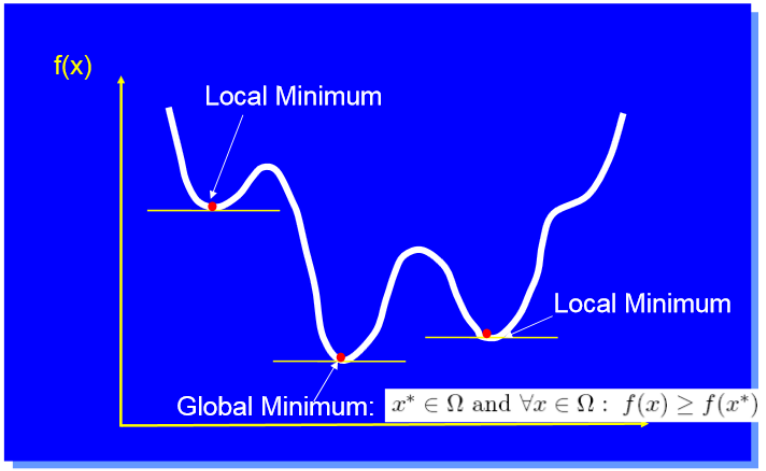
$$\min_x f(x)$$



Figures from Moritz Diehl's slides

Aim of most optimization algorithms

Find a local minimizer x*



Gradient

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

Hessian

$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}$$

First-order Necessary Conditions

If x* is a local minimizer and *f* is continuously differentiable in an open neighborhood of x*, then $\nabla f(\mathbf{x}^*) = 0$.

Second-order Necessary Conditions

If x* is a local minimizer of *f* and $\nabla^2 f$ exists and is continuous in an open neighborhood of x*, then $\nabla f(\mathbf{x}^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite.

Few words on optimization (cont.)

The general structure of the optimization algorithms in the Poblano Toolbox:

choose a starting value x_0
while not converged **do**
    find the search direction p_k
    determine the step length α
    new iterate $x_k = x_{k-1} + \alpha p_k$
end while

Different choices of search direction and step length lead to different optimization algorithms.

Taylor's approximation:

Suppose that *f* is twice continuously differentiable, then,

$$f(x_k + p) \approx f_k + p^{\mathsf{T}} \nabla f_k + \frac{1}{2} p^{\mathsf{T}} \nabla^2 f_k p$$

Steepest descent: $p_k = -\nabla f_k$

Newton's direction: $p_k = -(\nabla^2 f_k)^{-1} \nabla f_k$

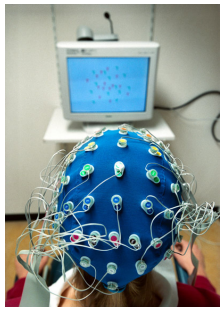
Quasi-Newton Methods (e.g., Limited-Memory BFGS): $p_k = -B_k^{-1} \nabla f_k$

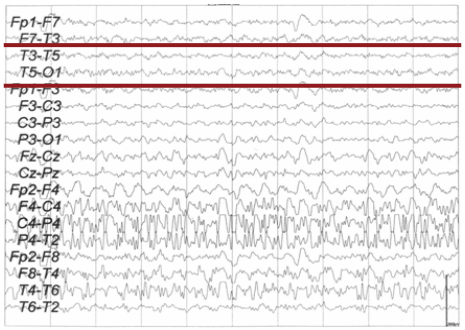
Nonlinear Conjugate Gradient Methods: $p_k = -\nabla f_k + \beta_k p_{k-1}$

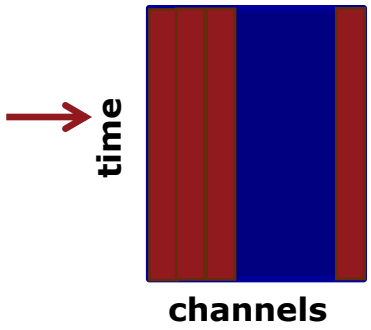
The ones using only the first-order info are implemented in the Poblano Toolbox.

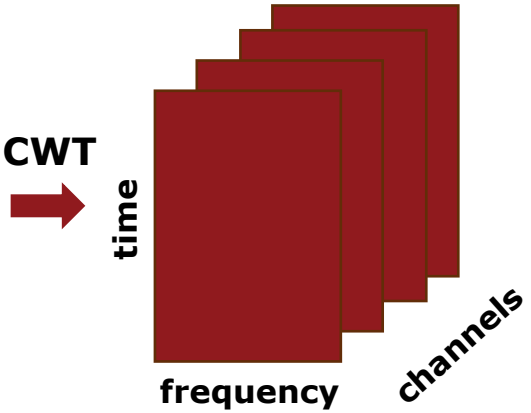
Application$_1$ (Neuroscience): CP has proved useful in epileptic seizure localization

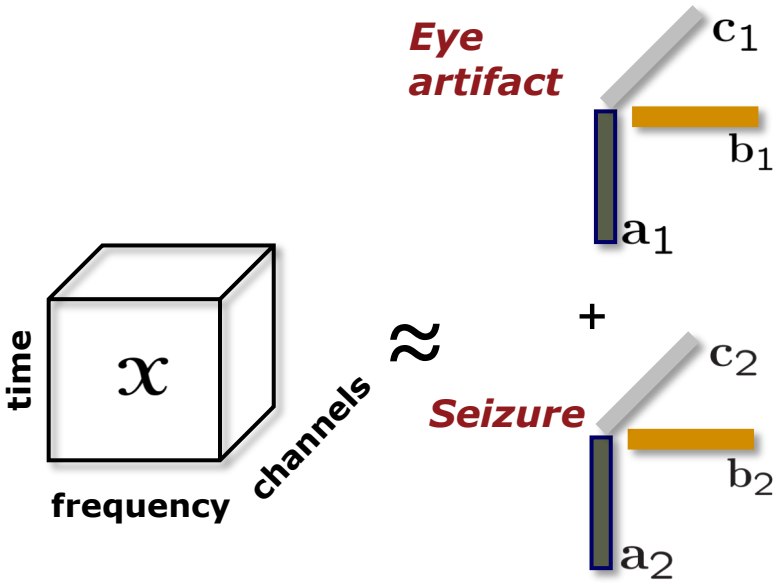
[Acar et al., 2007; De Vos et al., 2007]

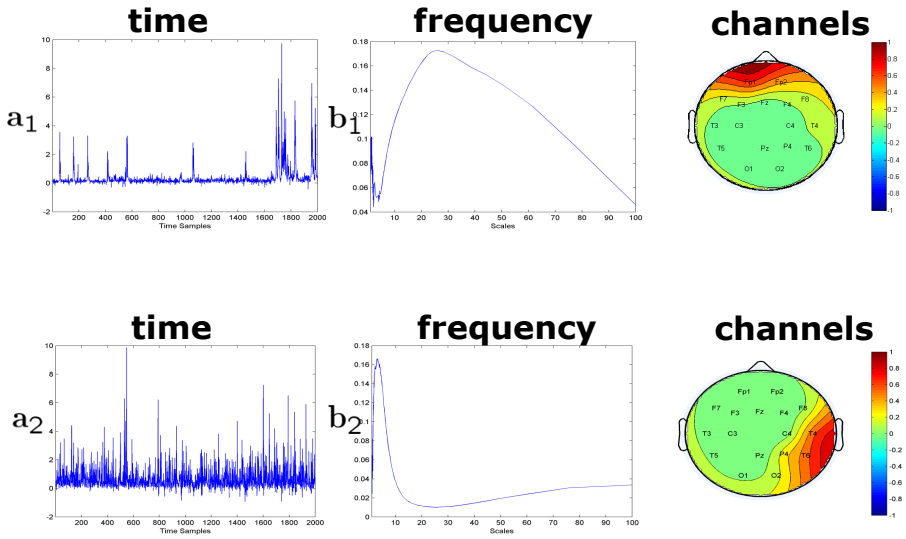










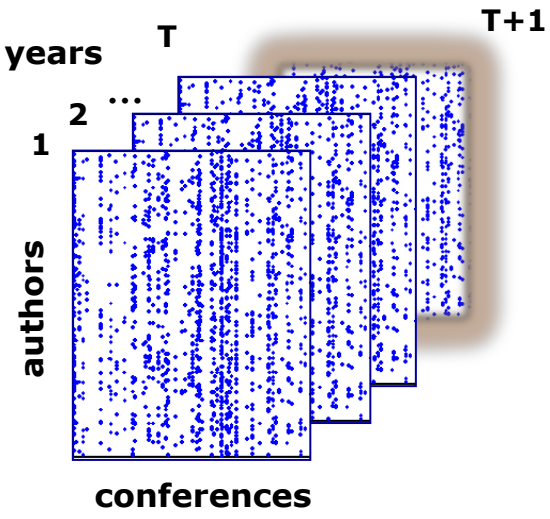


Application$_2$ (Recommender Systems): CP can capture temporal patterns useful for link prediction

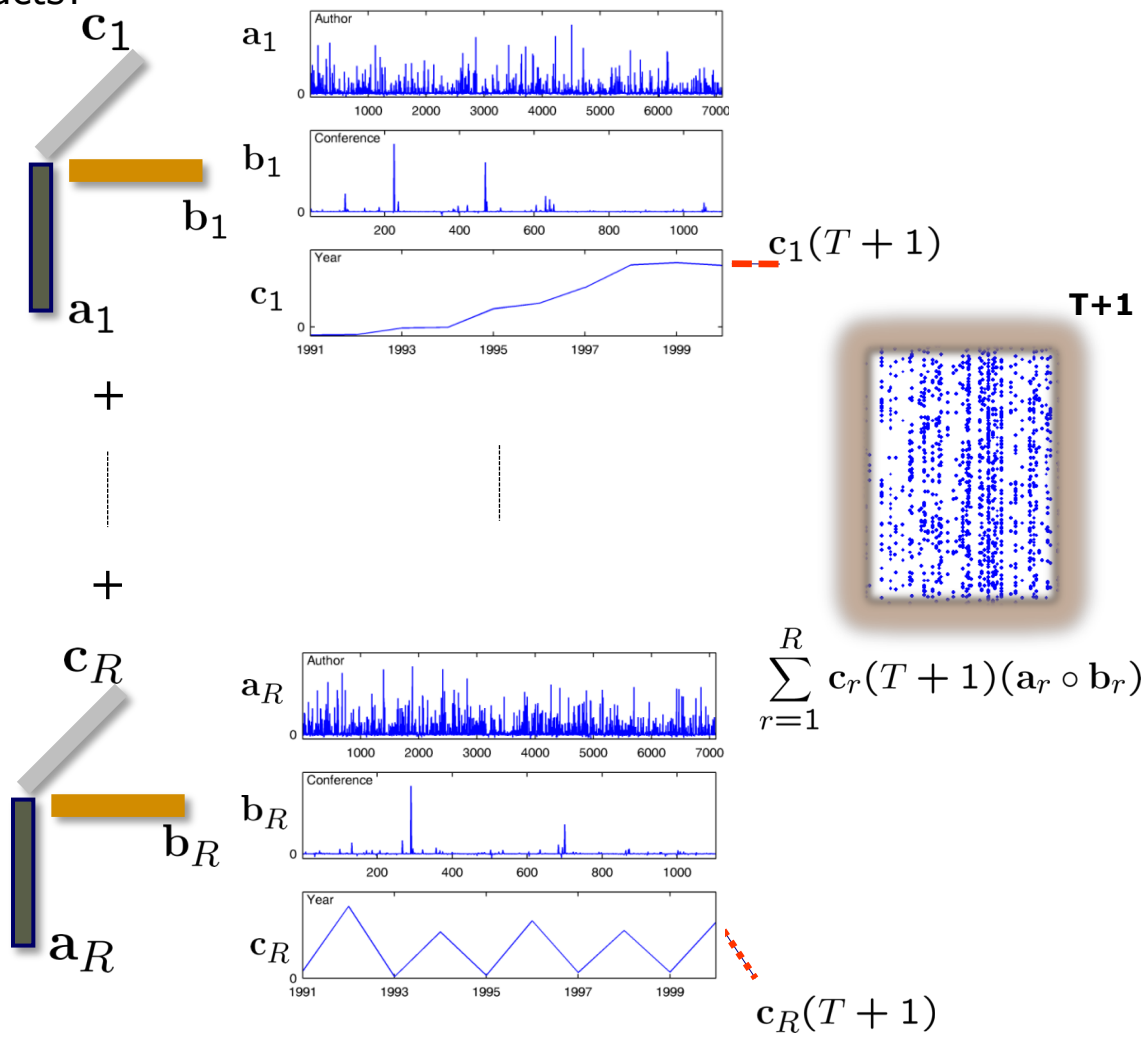
[Dunlavy, Kolda, Acar, 2011]

Temporal Link Prediction

Which customers will buy which products?
Which webpages will users visit?



$\approx$

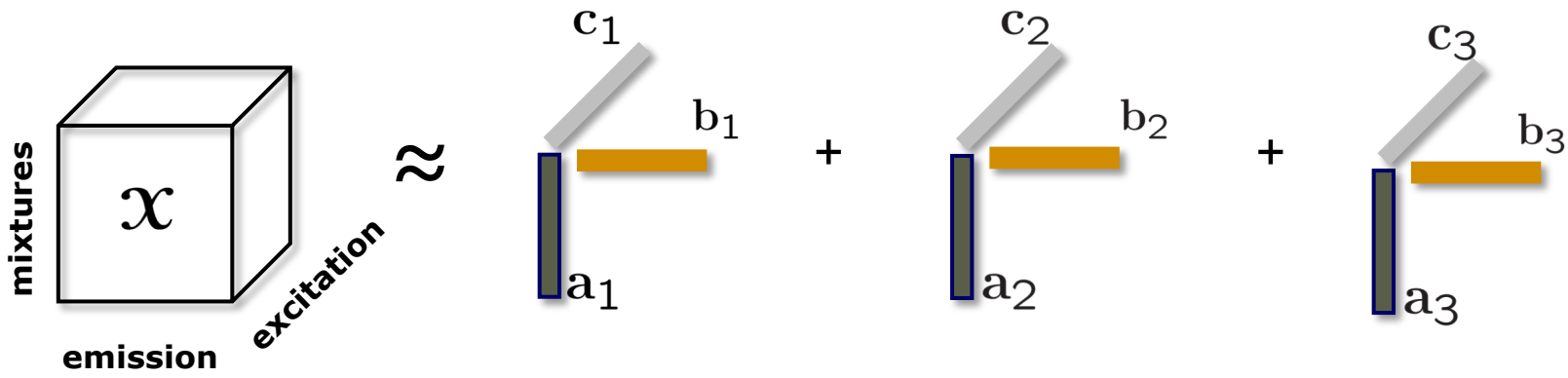


simula

Application$_3$ (Chemometrics): CP can separate the chemicals in mixtures

[Andersen and Bro, 2003]

A popular application of the CP model is the separation of individual chemicals from mixtures of chemicals measured using fluorescence spectroscopy.



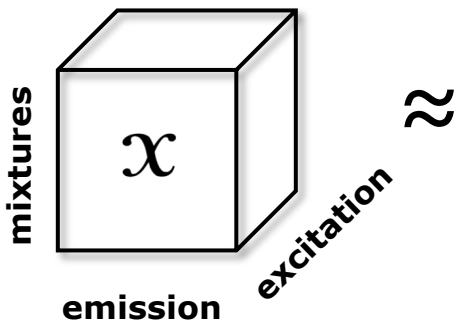
Amino Acid Data
http://www.models.life.ku.dk/

simula

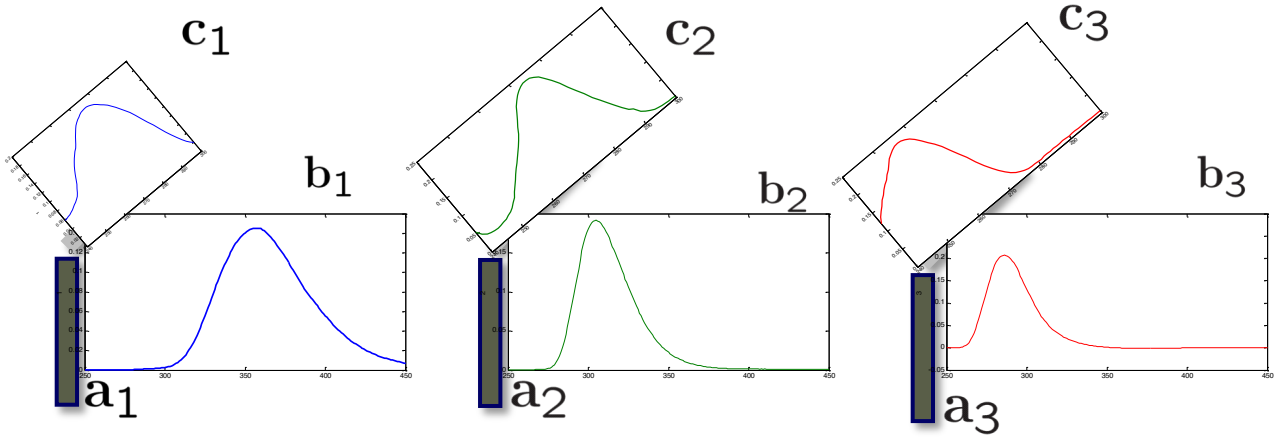
Application$_3$ (Chemometrics): CP can separate the chemicals in mixtures

[Andersen and Bro, 2003]

A popular application of the CP model is the separation of individual chemicals from mixtures of chemicals measured using fluorescence spectroscopy.

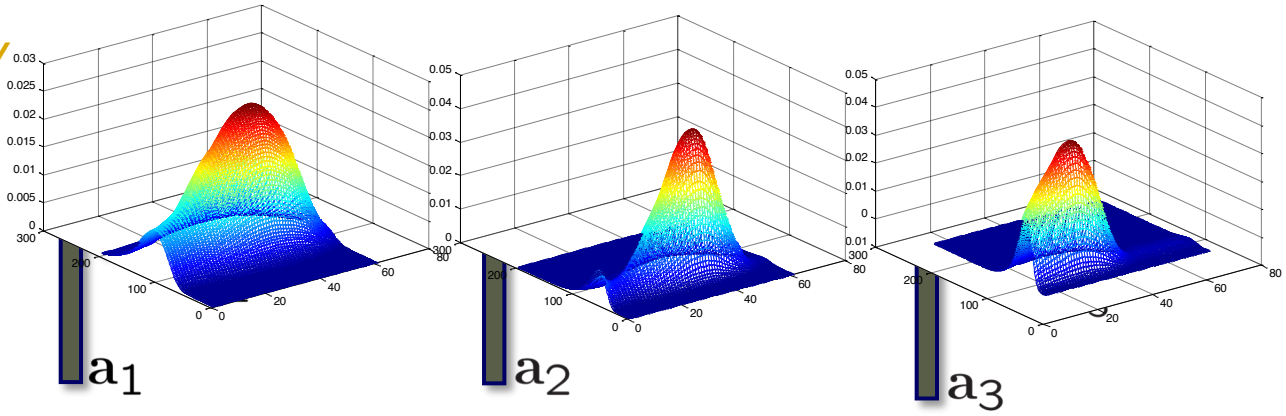


$\approx$



Amino Acid Data
http://www.models.life.ku.dk/

$\approx$



simula

Modeling the aminoacids data

```
% Load data
load aminoacids.mat
X = tensor(X.data);
X = X/norm(X);
% Set optimization parameters and fit a CP model using CP-OPT
options                = ncg('defaults');
options.DisplayIters   = 50;
options.RelFuncTol     = 1e-10;
options.StopTol        = 1e-10;
[Fac,FacInit,out]      = cp_opt(X, 3,'init','nvecs','alg','ncg','alg_options',options);
for i=1:3
    subplot(3,1,i)
    plot(Fac.U{i})
end
```

Model: CANDECOMP/PARAFAC (CP)

Algorithm: We can pick a gradient-based optimization method

- Nonlinear Conjugate Gradient (ncg)
- Limited–memory BFGS (lbfgs)
- ….

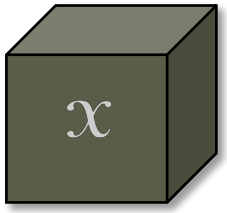
Initialization of factor matrices:

- svd-based
- entries randomly sampled from a standard normal distribution
- pre-defined cell array containing the factor matrices

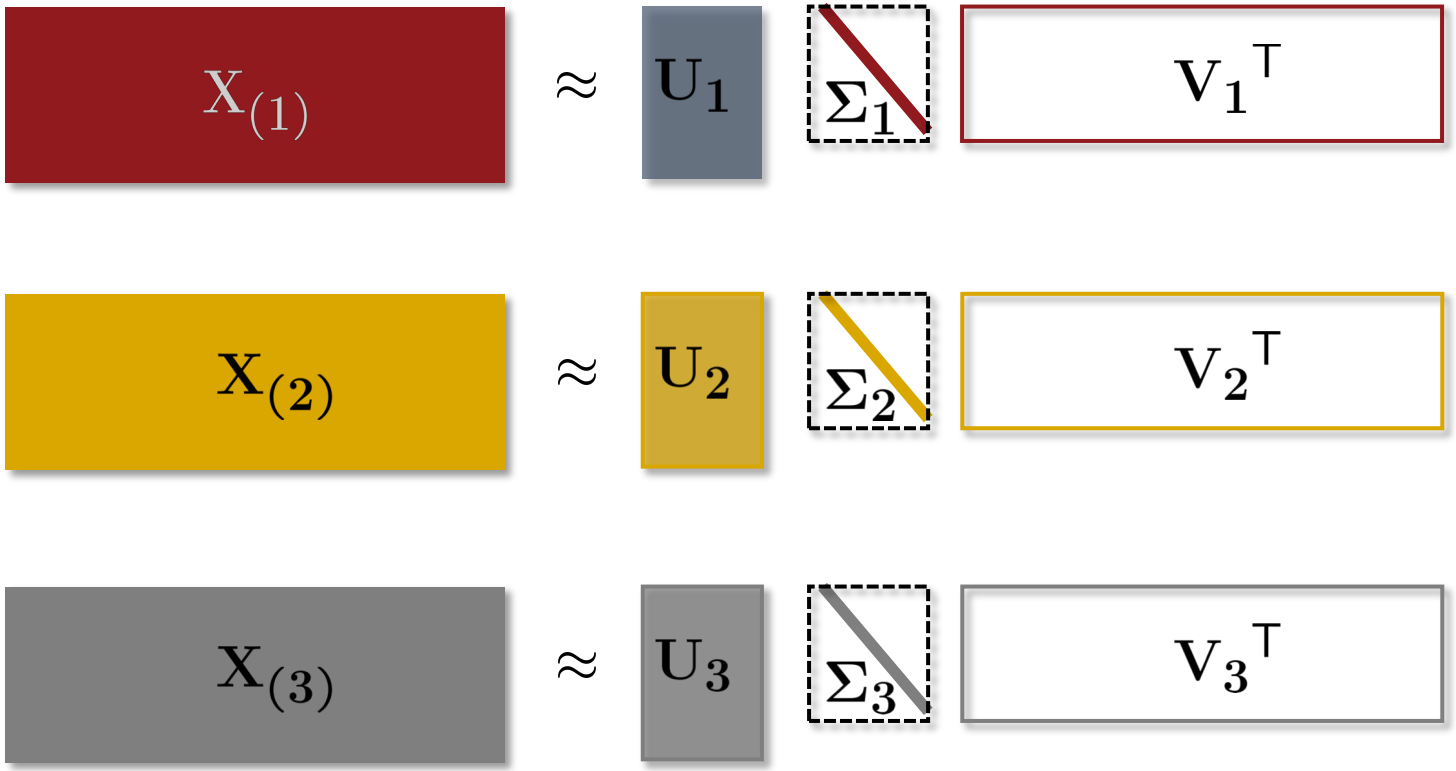
simula

SVD-based Initialization

Left singular vectors (i.e., $\mathbf{U_1}$, $\mathbf{U_2}$, $\mathbf{U_3}$) of matricized tensors in each mode can be used to initialize the factor matrices.

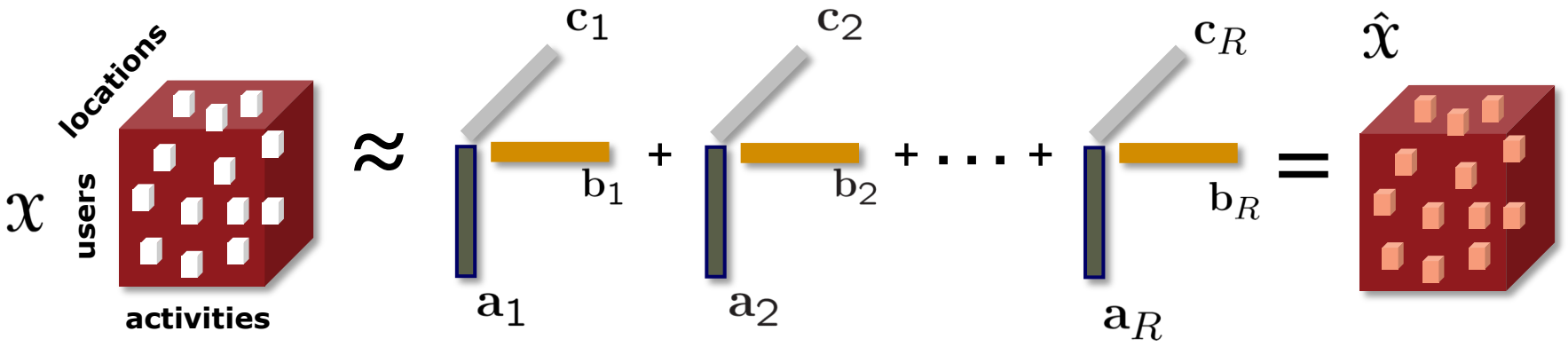


$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \| \mathcal{X} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \|^2$$



Tensor Factorizations: Missing Data Estimation (a.k.a. Tensor Completion)

Similar to the matrix case, in the case of missing data we can use low-rank tensor approximations to fill in missing entries [Tomasi and Bro, 2005]:



Finding low-rank approximation:

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

$$w_{ijk} = \begin{cases} 1 & \text{if } x_{ijk} \text{ is known,} \\ 0 & \text{if } x_{ijk} \text{ is missing.} \end{cases}$$

Data reconstruction:

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

simula

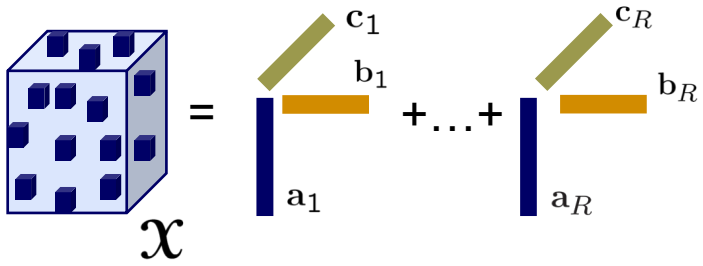
In the presence of missing data: CP-WOPT

[Acar, Dunlavy, Kolda, Mørup, 2011]

CP-WOPT (CP Weighted OPTimization) solves the following weighted optimization problem to fit the CP model to the known data entries in the tensor by ignoring the missing entries:

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

$$w_{ijk} = \begin{cases} 1 & \text{if } x_{ijk} \text{ is known,} \\ 0 & \text{if } x_{ijk} \text{ is missing.} \end{cases}$$



Step1: Define the objective function

$$f_{\mathcal{W}}(\mathbf{A},\mathbf{B},\mathbf{C}) = \| \mathcal{W} * (\mathcal{X} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]) \|^2 = \| \mathcal{Y} - \mathcal{Z} \|^2$$

$$\mathcal{Y} = \mathcal{W} * \mathcal{X}$$

$$\mathcal{Z} = \mathcal{W} * [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]$$

Step2: Compute the gradient

$$\frac{\partial f_{\mathcal{W}}}{\partial \mathbf{A}} = 2(\mathbf{Z}_{(1)} - \mathbf{Y}_{(1)})(\mathbf{C} \odot \mathbf{B})$$

$$\frac{\partial f_{\mathcal{W}}}{\partial \mathbf{B}} = 2(\mathbf{Z}_{(2)} - \mathbf{Y}_{(2)})(\mathbf{C} \odot \mathbf{A})$$

$$\frac{\partial f_{\mathcal{W}}}{\partial \mathbf{C}} = 2(\mathbf{Z}_{(3)} - \mathbf{Y}_{(3)})(\mathbf{B} \odot \mathbf{A})$$

Vectorize and concatenate the partials

$$\nabla f_{\mathcal{W}} = \begin{bmatrix} \frac{\partial f_{\mathcal{W}}}{\partial \mathbf{a}_1} \\ \vdots \\ \frac{\partial f_{\mathcal{W}}}{\partial \mathbf{a}_R} \\ \frac{\partial f_{\mathcal{W}}}{\partial \mathbf{b}_1} \\ \vdots \\ \frac{\partial f_{\mathcal{W}}}{\partial \mathbf{b}_R} \\ \frac{\partial f_{\mathcal{W}}}{\partial \mathbf{c}_1} \\ \vdots \\ \frac{\partial f_{\mathcal{W}}}{\partial \mathbf{c}_R} \end{bmatrix}$$

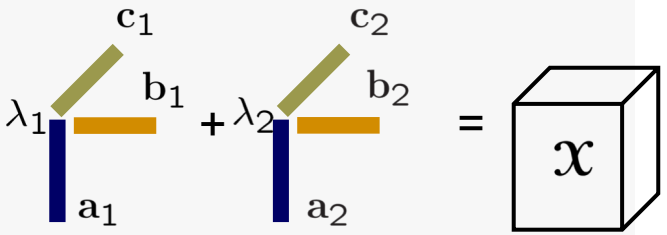
Step3: Pick a first-order optimization method

simula

Tensor Completion Example

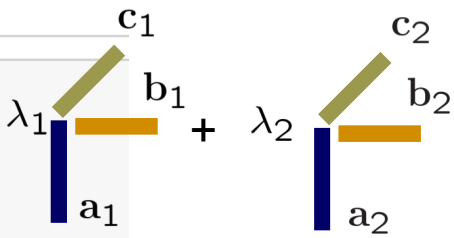
Construct a tensor with an underlying CP model

```
sz = [10 5 3];
R  = 2;
for i=1:length(sz)
    A{i}= randn(sz(i),R);
    for r=1:R
        A{i}(:,r)=A{i}(:,r)/norm(A{i}(:,r));
    end
end
lambda = [2 3]';
X = full(ktensor(lambda, A));
Xorig = X.data;
```



Compare the following with ktensor

```
Y=zeros(sz);
for r=1:R
    for k=1:size(Y,3)
        Y(:,:,k)= Y(:,:,k)+ lambda(r)*A{1}(:,r)*A{2}(:,r)'*A{3}(k,r);
    end
end
```



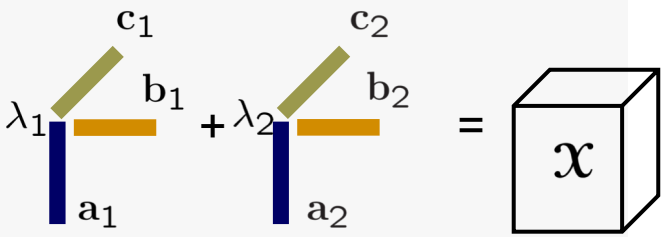
simula

Tensor Completion Example (cont.)

Construct a tensor with an underlying CP model

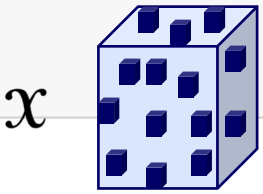
```
sz = [10 5 3];
R  = 2;
for i=1:length(sz)
    A{i}  = randn(sz(i),R);
    for r = 1:R
      A{i}(:,r)= A{i}(:,r)/norm(A{i}(:,r));
    end
end
lambda = [2 3]';

X     = full(ktensor(lambda,A));
Xorig = X;
```



Pick some random entries and set them to missing

```
M = round(rand(sz));
X(find(M==0))=NaN;
```



simula

Tensor Completion Example (cont.)

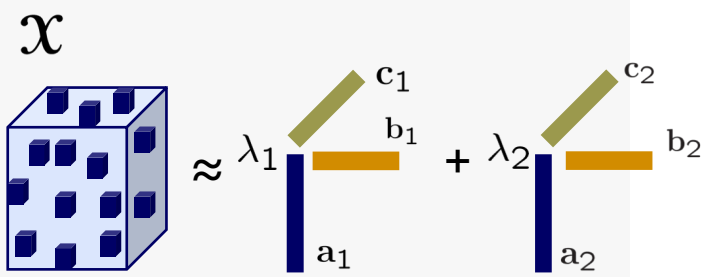
Suppose you are given the incomplete tensor X and you want to find the underlying CP components

```
% Record the places of the missing entries
W = tensor(~isnan(X.data));

% Set those missing entries in the tensor to 0
X(find(W==0))=0;

% parameters for the gradient-based optimization algorithm
options = ncg('defaults');
options.MaxFuncEvals = 10000;
options.MaxIters     = 10000; % change this line to change the max number of iterations
options.StopTol      = 1e-6;
options.RelFuncTol   = 1e-6;
options.DisplayIters = 50;
[Fac, FacInit, output] = cp_wopt(X,W,R,'alg', 'ncg','alg_options',options,'init','nvecs');

% Compare Fac(factors extracted by the model) with A (true factors)
for i=1:3
    corr(Fac.U{i},A{i})
end
```



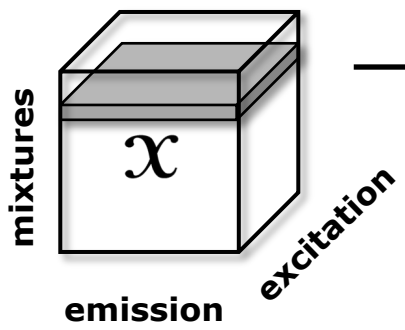
Reconstruct your data from the factor matrices

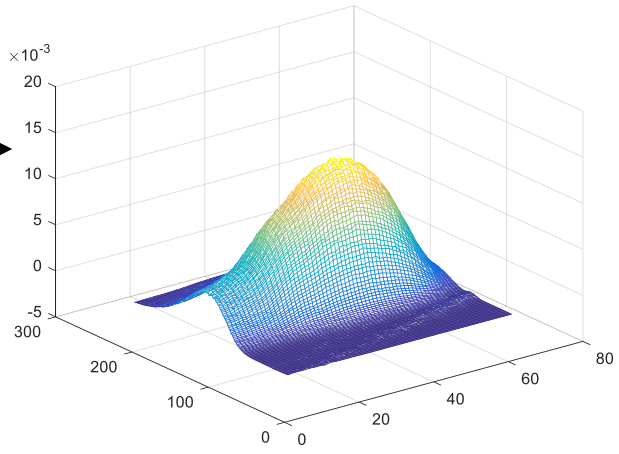
```
Xhat = full(Fac);
% Plot original vs. estimated values for the missing data
plot(Xorig(find(W==0)), Xhat(find(W==0)),'*');
xlabel('Original'); ylabel('Estimated')
```

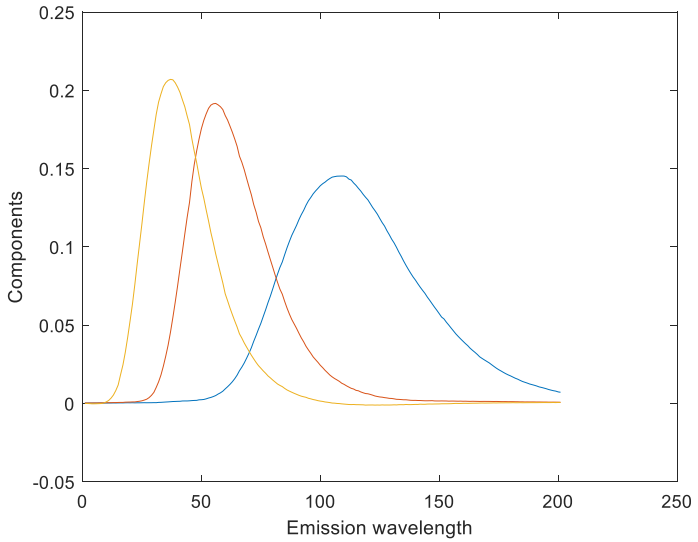
$$\hat{\mathcal{X}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

simula

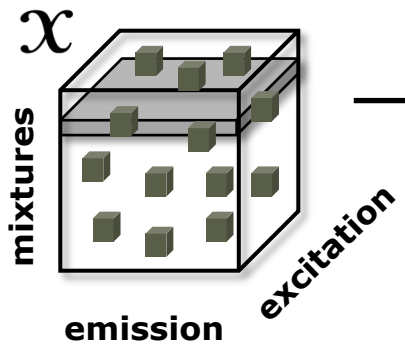
CP-WOPT can accurately capture the factors even with high amounts of missing data!

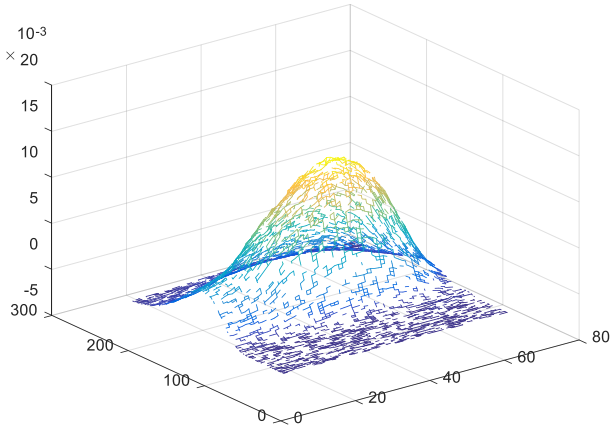


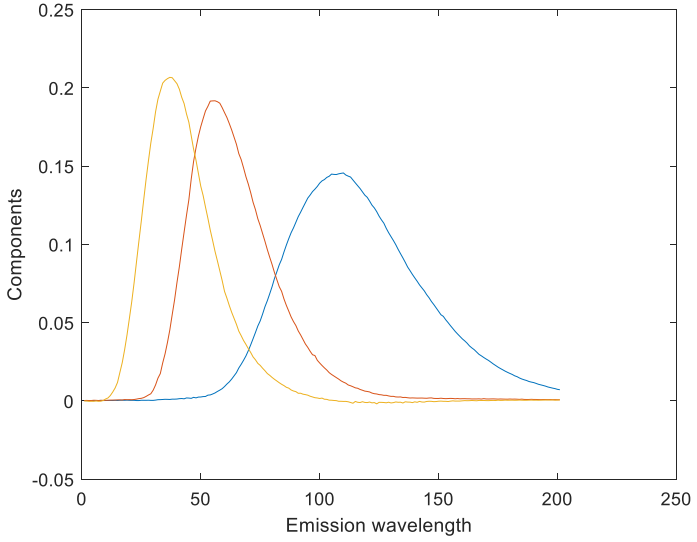




50% Missing Data



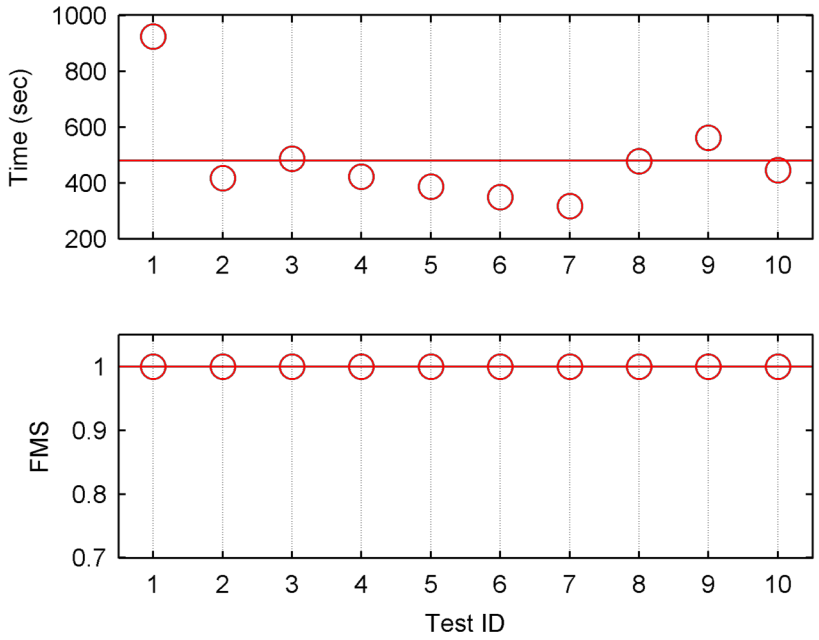




CP-WOPT scales to large problems!

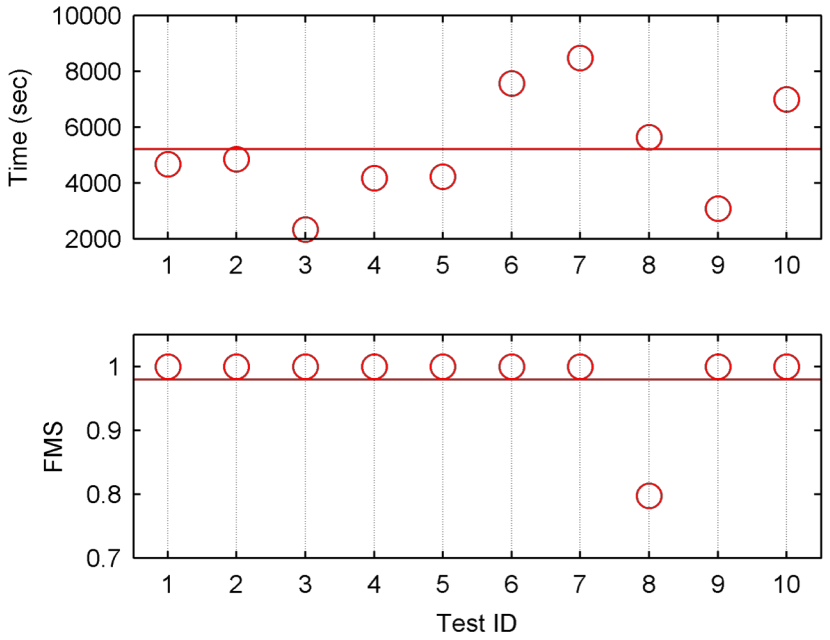
500 x 500 x 500 with 99% missing entries (i.e., 1.25 million known entries)

Dense storage = 1GB
Sparse storage = 40MB



1000 x 1000 x 1000 with 99.5% missing entries (i.e., 5 million known entries)

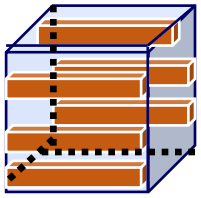
Dense storage = 8GB
Sparse storage = 160MB



Factor Match Score

$$\text{FMS} = \max_{\sigma \in \Pi(R,\bar{R})} \frac{1}{\bar{R}} \sum_{r=1}^{R} \left(1 - \frac{\left| \lambda_r - \bar{\lambda}_{\sigma(r)} \right|}{\max\left\{ \lambda_r, \bar{\lambda}_{\sigma(r)} \right\}} \right) \prod_{n=1}^{N} \left| a_r^{(n)\mathrm{T}} \bar{a}_{\sigma(r)}^{(n)} \right|$$

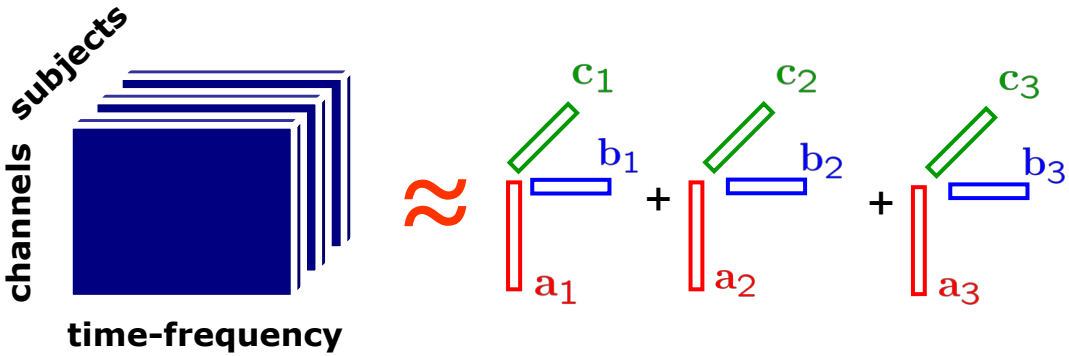
On a Linux workstation with 2 Quad-Core Intel Xeon 3.0 GHz processors and 32 GB RAM

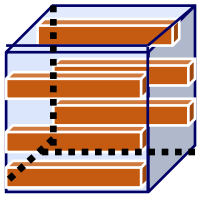


Application$_4$ (Neuroscience): We can capture underlying factors accurately from EEG data with missing channels

[Acar, Dunlavy, Kolda, Mørup, 2011]

Goal: To differentiate between left and right hand stimulation

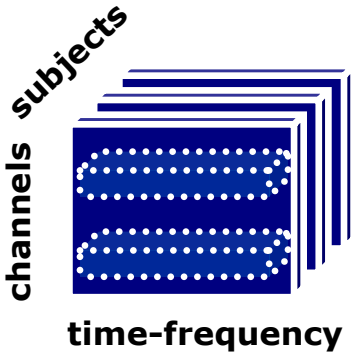


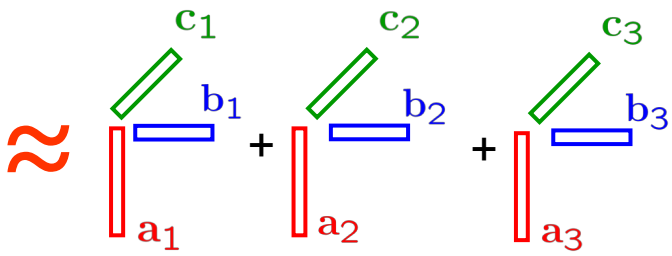


Application$_4$ (Neuroscience): We can capture underlying factors accurately from EEG data with missing channels

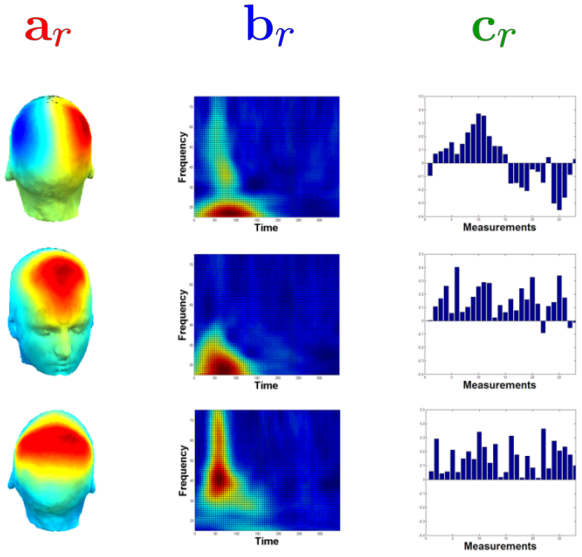
[Acar, Dunlavy, Kolda, Mørup, 2011]

Goal: To differentiate between left and right hand stimulation

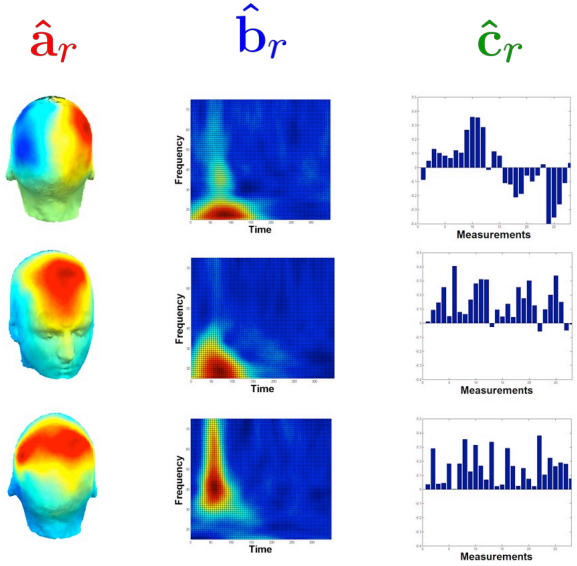


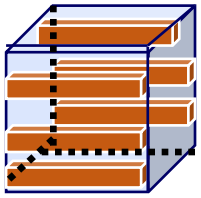


No Missing Data



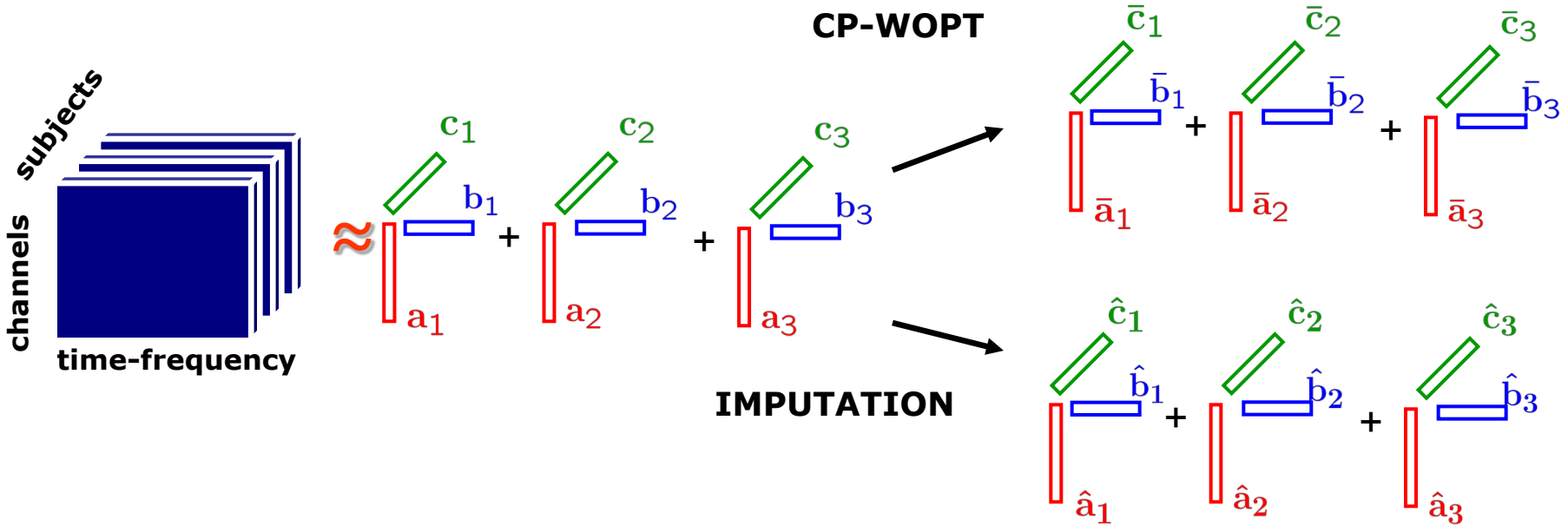
30 Channels/slice Missing

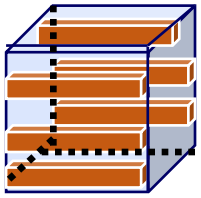




Ignoring missing entries vs. Imputation

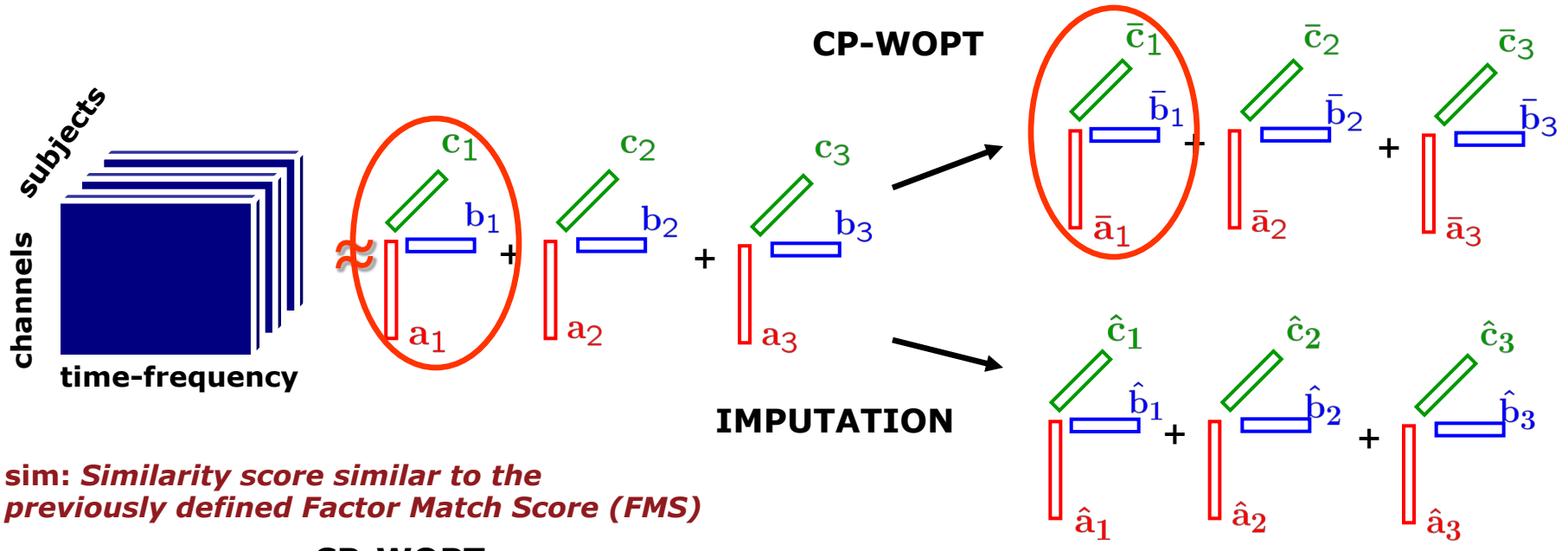
Ignoring missing entries, in particular, when there is large amount of missing data is better than alternatives based on imputation, e.g., imputing missing entries with mean.





Ignoring missing entries vs. Imputation

Ignoring missing entries, in particular, when there is large amount of missing data is better than alternatives based on imputation, e.g., imputing missing entries with mean.



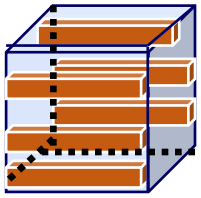
sim: ***Similarity score similar to the previously defined Factor Match Score (FMS)***

CP-WOPT

Missing Channels	sim($r=1$)	sim($r=2$)	sim($r=3$)
1	0.9989	0.9995	0.9991
10	0.9869	0.9936	0.9894
20	0.9560	0.9826	0.9697
30	0.9046	0.9604	0.9312
40	0.6192	0.8673	0.7546

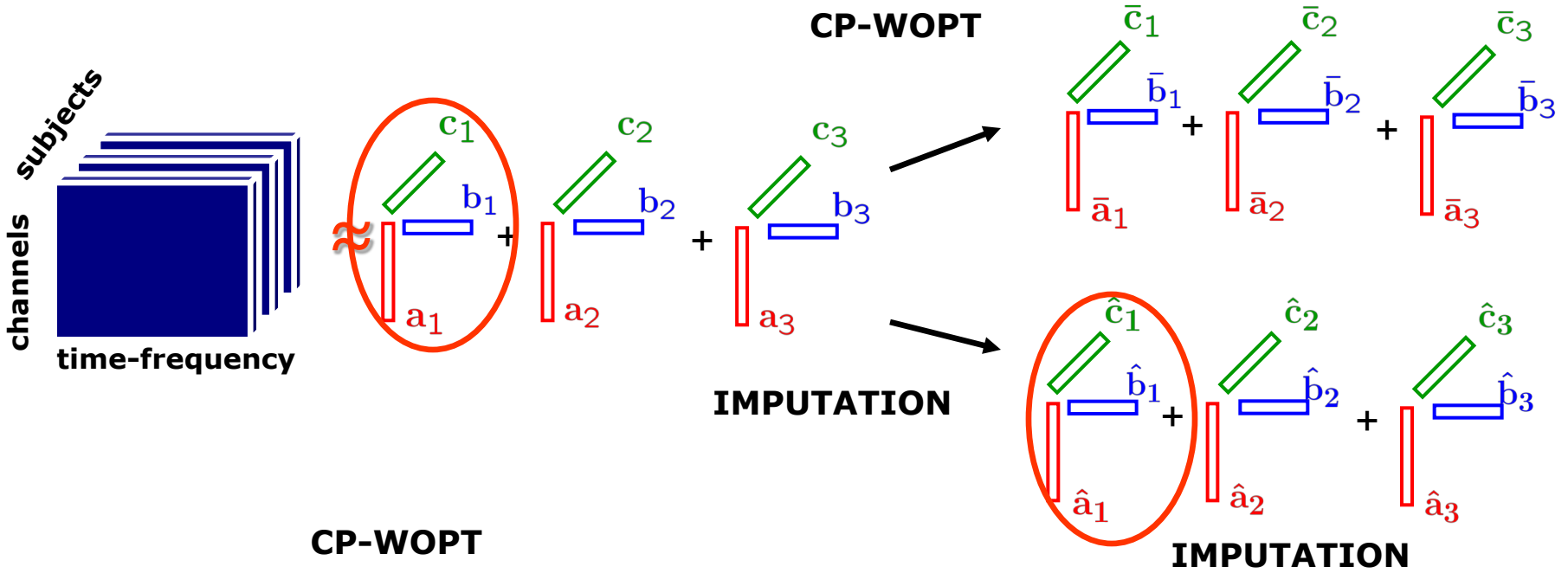
IMPUTATION

Missing Channels	sim($r=1$)	sim($r=2$)	sim($r=3$)
1	0.9970	0.9984	0.9982
10	0.9481	0.9729	0.9752
20	0.9002	0.9475	0.9371
30	0.6435	0.8719	0.8100
40	0.3045	0.7126	0.5699



Ignoring missing entries vs. Imputation

Ignoring missing entries, in particular, when there is large amount of missing data is better than alternatives based on imputation, e.g., imputing missing entries with mean.

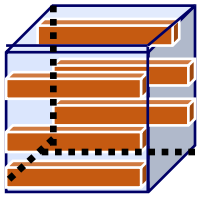


CP-WOPT

Missing Channels	sim($r=1$)	sim($r=2$)	sim($r=3$)
1	0.9989	0.9995	0.9991
10	0.9869	0.9936	0.9894
20	0.9560	0.9826	0.9697
30	0.9046	0.9604	0.9312
40	0.6192	0.8673	0.7546

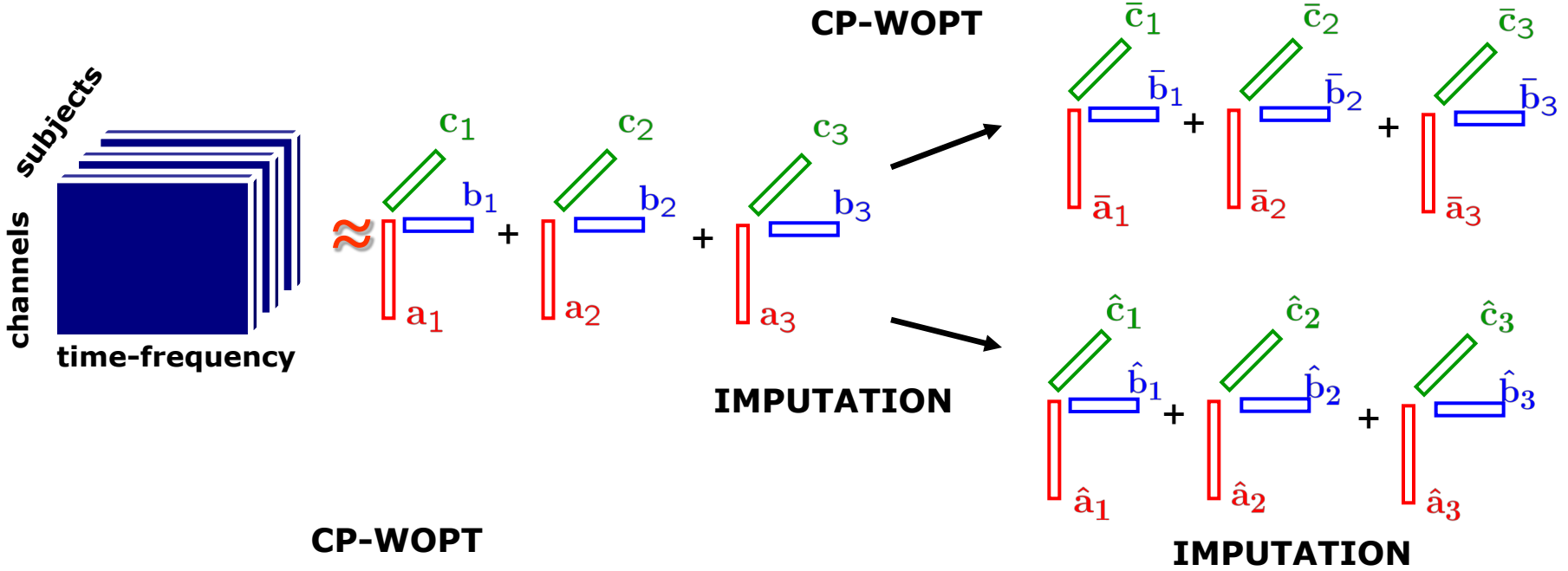
IMPUTATION

Missing Channels	sim($r=1$)	sim($r=2$)	sim($r=3$)
1	0.9970	0.9984	0.9982
10	0.9481	0.9729	0.9752
20	0.9002	0.9475	0.9371
30	0.6435	0.8719	0.8100
40	0.3045	0.7126	0.5699



Ignoring missing entries vs. Imputation

Ignoring missing entries, in particular, when there is large amount of missing data is better than alternatives based on imputation, e.g., imputing missing entries with mean.



CP-WOPT

Missing Channels	sim($r=1$)	sim($r=2$)	sim($r=3$)
1	0.9989	0.9995	0.9991
10	0.9869	0.9936	0.9894
20	0.9560	0.9826	0.9697
30	0.9046	0.9604	0.9312
40	0.6192	0.8673	0.7546

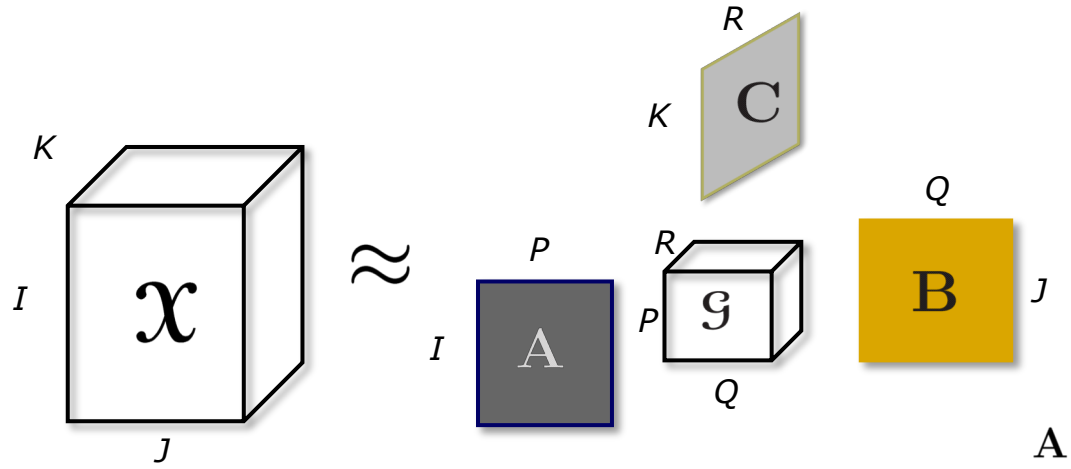
IMPUTATION

Missing Channels	sim($r=1$)	sim($r=2$)	sim($r=3$)
1	0.9970	0.9984	0.9982
10	0.9481	0.9729	0.9752
20	0.9002	0.9475	0.9371
30	0.6435	0.8719	0.8100
40	0.3045	0.7126	0.5699

Another popular tensor factorization model: Tucker

[Tucker, 1963 & 1964 & 1966]

A more flexible tensor factorization model compared to the CP model is the Tucker3 model, which models a third-order tensor using three factor matrices corresponding to each mode as well as a core array.



$$\mathcal{X} \approx \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} \mathbf{a}_p \circ \mathbf{b}_q \circ \mathbf{c}_r$$

$$\mathcal{X} \approx [\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

$$\mathbf{A} \in \mathbb{R}^{I \times P} = \begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_P \end{bmatrix}$$

$$\mathbf{B} \in \mathbb{R}^{J \times Q} = \begin{bmatrix} \mathbf{b}_1 & \cdots & \mathbf{b}_Q \end{bmatrix}$$

$$\mathbf{C} \in \mathbb{R}^{K \times R} = \begin{bmatrix} \mathbf{c}_1 & \cdots & \mathbf{c}_R \end{bmatrix}$$

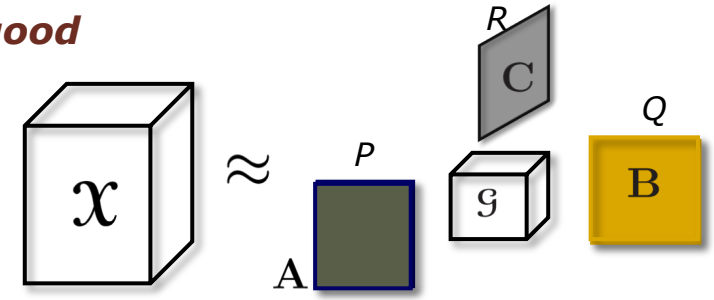
$\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$: core tensor

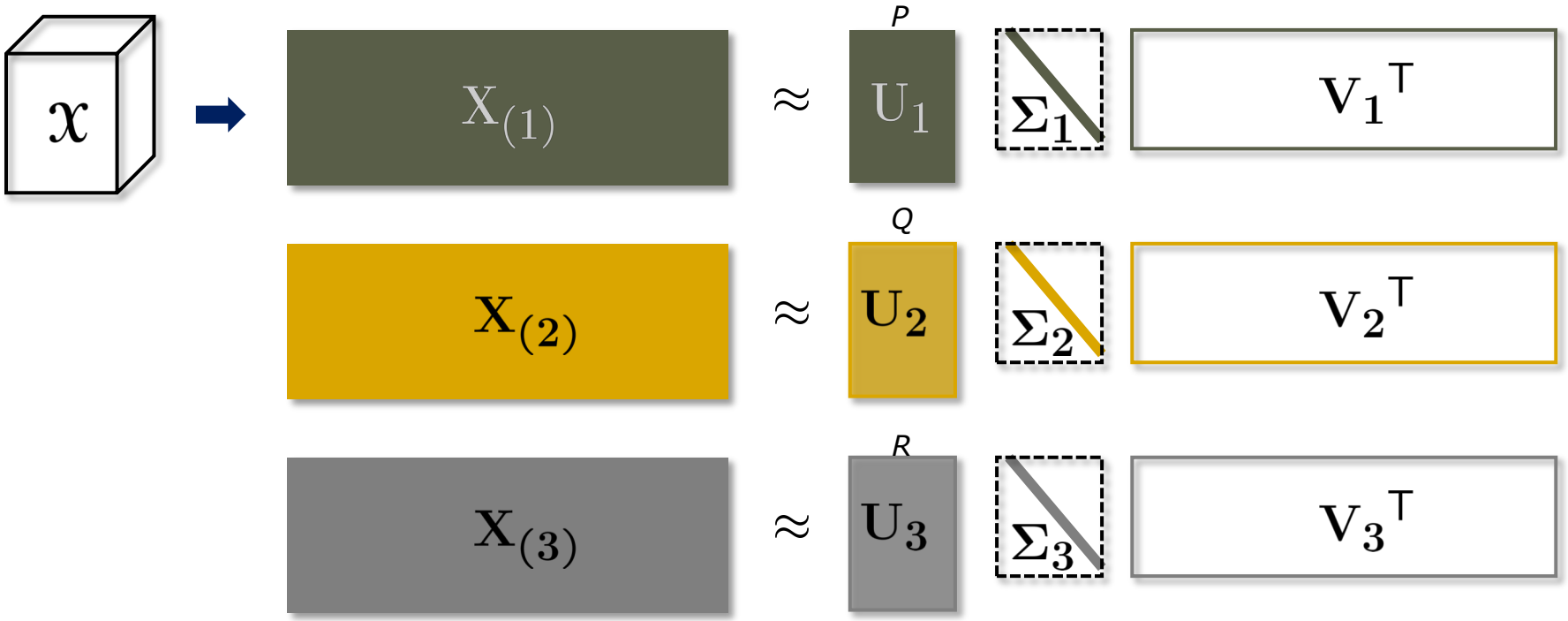
Tucker with orthogonality constraints on A, B, and C corresponds to the best rank-(P,Q,R) approximation of the tensor.

Tucker is considered to be another generalization of SVD to higher-order tensors: Higher-order SVD (HOSVD)

[Tucker, 1966; De Lathauwer, De Moor, Vandewalle, 2000]

Truncation of HOSVD gives a good rank-(P,Q,R) approximation

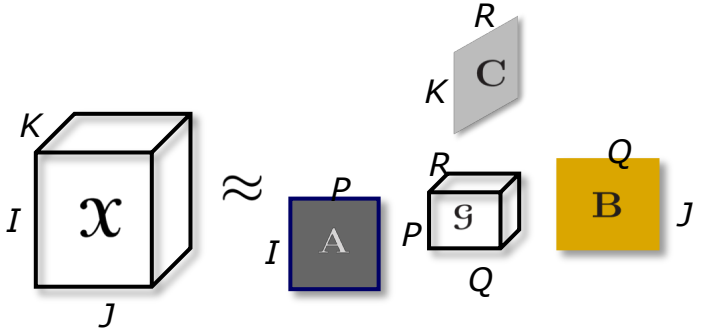




$$\mathcal{G} = \mathcal{X} \times_1 \mathbf{U}_1^{\mathsf{T}} \times_2 \mathbf{U}_2^{\mathsf{T}} \times_3 \mathbf{U}_3^{\mathsf{T}}$$

Tucker vs. CP

Tucker



$$\mathcal{X} \approx \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} \mathbf{a}_p \circ \mathbf{b}_q \circ \mathbf{c}_r$$

$$\mathcal{X} \approx [\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

More flexible than CP

Different number of components in each mode

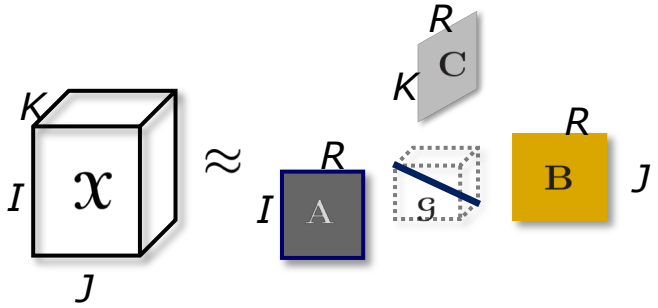
The core tensor models the interaction between factors in different modes

Not-unique

$$[\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = [\![\mathcal{G} \times_1 \mathbf{U}; \mathbf{A}\mathbf{U}^{-1}, \mathbf{B}, \mathbf{C}]\!]$$

EXPLORATORY DATA ANALYSIS & COMPRESSION!

CP



$$\mathcal{X} \approx \sum_{r=1}^{R} g_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

Restricted model with a specific structure

Same number of components in each mode

Super-diagonal core tensor

Nice uniqueness properties

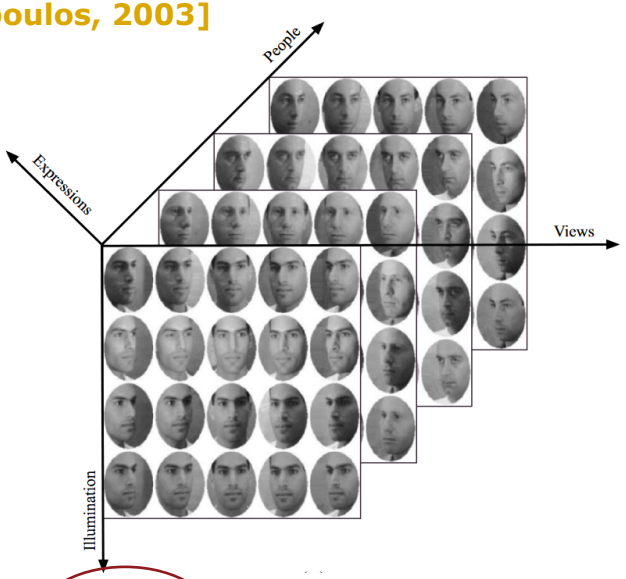
INTERPRETABILITY!!!

Application$_5$ (Face Recognition): TensorFaces

[Vasilescu and Terzopoulos, 2003]

Facial image data tensor with modes (people x viewpoints x illuminations x expressions x pixels) is constructed by arranging the images from 28 people in 5 viewpoints, 4 illuminations and 3 expressions.

$$\mathcal{D} : 28 \times 5 \times 4 \times 3 \times 7943$$



HOSVD gives the following:

$$\mathcal{D} = \mathcal{Z} \times_1 \mathbf{U}_{\text{people}} \times_2 \mathbf{U}_{\text{views}} \times_3 \mathbf{U}_{\text{illums}} \times_4 \mathbf{U}_{\text{expres}} \times_5 \mathbf{U}_{\text{pixels}}$$

eigenfaces

$$\mathcal{Z} \times_5 \mathbf{U}_{\text{pixels}}$$

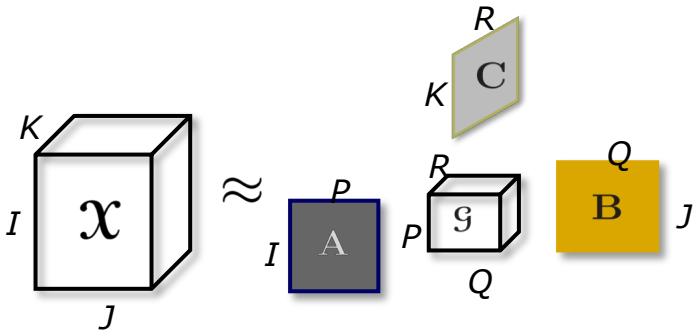
tensorfaces

Better face recognition performance using tensorfaces compared to eigenfaces

Truncation in the illumination mode:

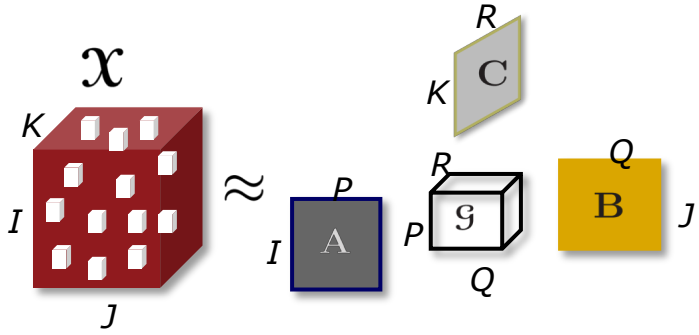
simula

Similar to CP, Tucker model can be fit to incomplete data using weighted optimization!



$$\mathcal{X} \approx [\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

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



$$\min_{\mathcal{G},\mathbf{A},\mathbf{B},\mathbf{C}} \| \mathcal{W} * (\mathcal{X} - [\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]) \|^2$$

$$w_{ijk} = \begin{cases} 1 & \text{if } x_{ijk} \text{ is known,} \\ 0 & \text{if } x_{ijk} \text{ is missing.} \end{cases}$$

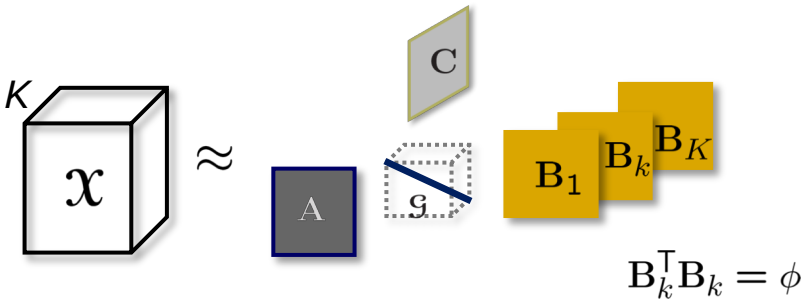
$$\hat{\mathcal{X}} = [\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

simula

There are other tensor factorization models

Mainly for applications with uniqueness & interpretability concerns

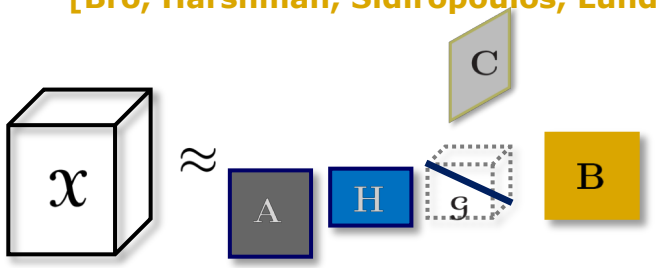
PARAFAC2 [Harshman, 1972]



$$\mathbf{B}_k^\top \mathbf{B}_k = \phi$$

PARALIND (Parallel profiles with Linear Dependences)

[Bro, Harshman, Sidiropoulos, Lundy, 2009]



DEDICOM (Decomposition into directional components [Harshman, 1978]

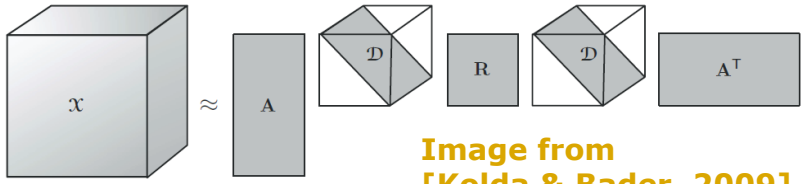


Image from [Kolda & Bader, 2009]

Handles differences between factors across different slices. Similar in that sense, there are also Shifted PARAFAC and Convolutive PARAFAC.

Mainly for compression & reconstruction applications and higher-order tensors with many modes, e.g., 10th-order tensor

Tensor Train (TT) ***And hierarchical tucker (H-Tucker)***

[Oseledets, 2011]

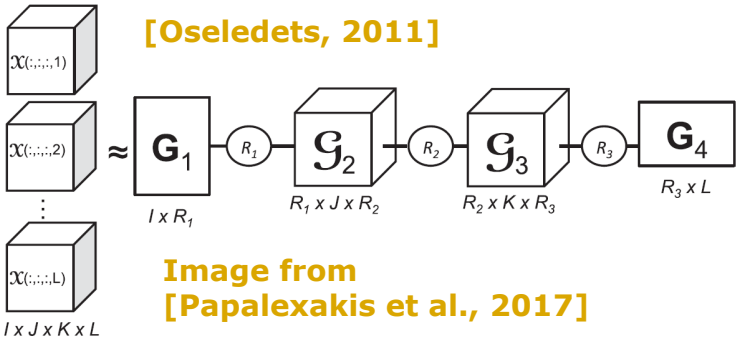


Image from [Papalexakis et al., 2017]

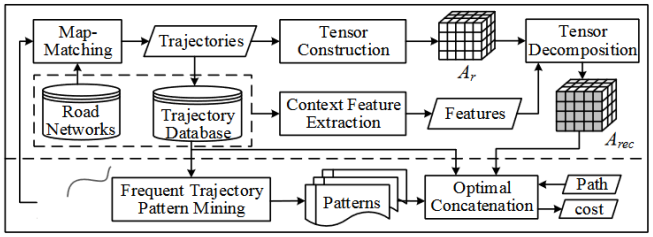
Sequences of SVDs on various unfoldings:

$$\text{SVD of reshape}(\mathcal{X}, [I, JKL]) : \mathbf{G}_1, \mathbf{Z}_1$$
$$\text{SVD of reshape}(\mathbf{Z}_1, [R_1 J, KL]) : \mathcal{G}_2, \mathbf{Z}_2$$
$$\text{SVD of reshape}(\mathbf{Z}_2, [R_2 K, L]) : \mathcal{G}_3, \mathbf{G}_4$$

There are many other applications of tensor factorizations

Urban Computing

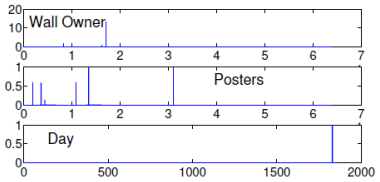
Travel Time Estimation [Wang et al., *KDD,* 2014]



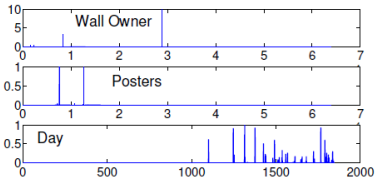
road segments x drivers x time slots

Social Network Analysis

[Papalexakis et al., *PKDD,* 2012]



(a) FACEBOOK anomaly (Wall owner's birthday)



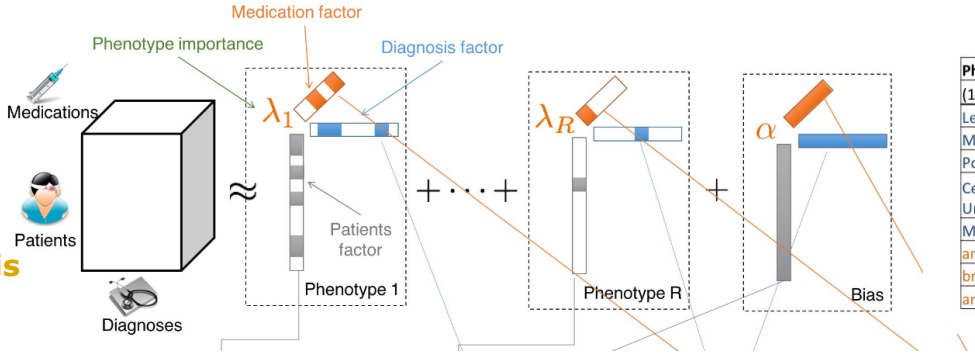
(b) FACEBOOK normal activity

wall owner x poster x day

Electronic Healthcare Records

EHR-based Phenotyping [Henderson et al., *ICHI,* 2017]

patients x medications x diagnosis



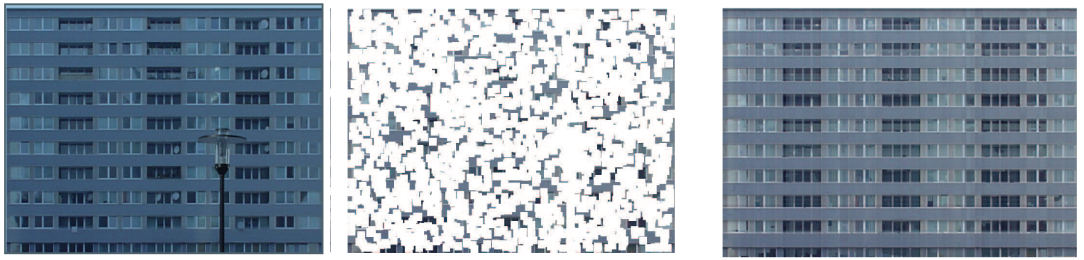
Phenotype 1
(15.43% of Patients)
Legally Blind
Major Symptoms, Abnormalities (1,2)
Polyneuropathy
Cerebrovascular Disease Late Effects, Unspecified
Multiple Sclerosis
anticonvulsants
bronchodilators
anxiolytics, sedatives, and hypnotics

Computer Vision

Image Reconstruction: [Liu et al., *PAMI,* 2013]

Tensor: original color image with color channels

original ***observed*** ***reconstruction***



Active Research Areas

Manifesto from Dagstuhl Perspectives Workshop 16152

Tensor Computing for Internet of Things

SCHLOSS DAGSTUHL
Leibniz-Zentrum für Informatik

[Acar, Anandkumar, Mullin, Rusitschka, Tresp, 2018]

Scalability

Parallel and distributed computations [Smith and Karypis, 2016], [Kaya and Ucar, 2015 & 2016]

Portability to emerging architectures [Phipps et al., 2017]

Sampling [Phan and Cichocki, 2011; Papalexakis et al., 2012; Vervliet and De Lathauwer, 2016; Battaglino et al., 2018]

Different Loss Functions

[Yilmaz, Cemgil, Simsekli, 2011; Ermis, Acar, Cemgil, 2015] [Kolda et al., *Presented at TRICAP*, 2018] [Vandecappelle et al., *Presented at TRICAP*, 2018]

Deep Learning and Tensor Factorizations

Tensor Train Factorization of a Feed Forward Layer [Novikov, Podoprikhin, Osokin, Vetrov, 2015]

TT-Recurrent Neural Networks [Yang, Krompass, Tresp, 2017]

Software

MATLAB: Nway Toolbox, Tensor Toolbox, TT Toolbox, TensorLab

R: multiway

Sparse: *http://frostt.io/*

Data Sets

Dense: *www.models.life.ku.dk*

Public data sets for multivariate data analysis

IMPORTANT: all downloadable material listed on these pages - appended by specifics mentioned under the individual headers/chapters - is available for public use. Please note that while great care has been taken, the software, code and data are provided "as is" and that Q&T, LIFE, KU does not accept any responsibility or liability.

Name (click for details)	Description	Located at	Format
NIR	NIR Sugarcane data	University of Copenhagen	Matlab
Tongue data	Three-way data from the work Richard Harshman	University of Copenhagen	Text
JODA data set **NEW**	NMR, LC-MS and EEM prototypical experimental coupled data sets for JODA	University of Copenhagen	Matlab
RAMAN pork fat **NEW**	The samples for this study were 16 pork carcasses	University of Copenhagen	Matlab
NIR soil **NEW**	Soil samples from long-term field experiment in Abisko, northern Sweden	University of Copenhagen	Matlab
AutoChrome **NEW**	Automatically find PARAFAC2 components of hyphenated chromatographic data	University of Copenhagen	Matlab
Metabolomic data fusion **NEW**	Biomarker, fluorescence and 1H-NMR data from case/control study on colorectal cancer	University of Copenhagen	Matlab
Olive oil	Oil samples analyzed by HPLC with charged aerosol detector	University of Copenhagen	Matlab
Example tensors	Some example tensors with known problems such as degeneracy, swamps and local minima	University of Copenhagen	Matlab

All tensors:

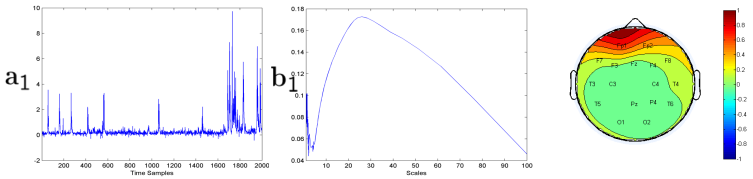
Name	Non-zeros	Order
Amazon Reviews	1,741,809,018	3
Chicago Crime	5,330,673	4
Delicious	140,126,181	4
Enron Emails	54,202,099	4
Flickr	112,890,310	4
LBNL-Network	1,698,825	5
Matrix Multiplication	M*K*N	3
NELL-1	143,599,552	3
NELL-2	76,879,419	3
NIPS Publications	3,101,609	4
Patents	3,596,640,708	3
Reddit-2015	4,687,474,081	3
Uber Pickups	3,309,490	4
VAST 2015 Mini-Challenge 1	26,021,945	5

Summary

Matrix Factorizations: How to use matrix factorizations to (i) find the underlying factors, (ii) predict missing entries

Matrix factorizations without constraints are not unique.

Constraints need to make sense in terms of the application; otherwise, the model will not reveal what we are looking for.



Tensor Factorizations

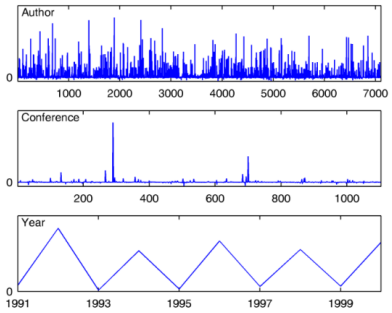
Tensor factorization methods such as CP have better uniqueness properties compared to matrix factorizations.

How to use tensor factorizations, in particular, CP, to find the underlying factors, e.g.,

Chemometrics

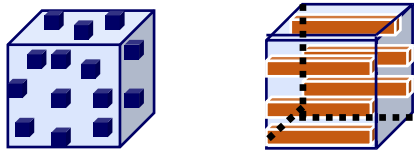
Neuroscience

Temporal Link Prediction



How to fit a CP model to a higher-order tensor with missing entries

Algorithms: ALS, Gradient-based optimization approaches



MATLAB functions

cp-opt: Fitting a CP model to a tensor **(from Tensor Toolbox)**

[Fac, FacInit, output] = cp_opt(X, R, 'init', 'random', 'alg','ncg', 'alg_options', options);

cp-wopt: Fitting a CP model to a tensor with missing entries **(from Tensor Toolbox)**

[Fac, FacInit, output] = cp_wopt(X, W, R, 'init', 'random', 'alg','ncg', 'alg_options', options);

Model: Optimization problem

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \| \mathcal{X} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!] \|^2 \qquad \min_{\mathbf{A},\mathbf{B},\mathbf{C}} \| \mathcal{W} * (\mathcal{X} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]) \|^2$$

Tensor Toolbox
Computation of the function value and the gradient

$$f(\mathbf{A},\mathbf{B},\mathbf{C}) = \frac{1}{2} \| \mathcal{X} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!] \|^2$$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial \mathbf{a}_1} \\ \vdots \\ \frac{\partial f}{\partial \mathbf{a}_R} \\ \frac{\partial f}{\partial \mathbf{b}_1} \\ \vdots \\ \frac{\partial f}{\partial \mathbf{b}_R} \\ \frac{\partial f}{\partial \mathbf{c}_1} \\ \vdots \\ \frac{\partial f}{\partial \mathbf{c}_R} \end{bmatrix} \quad \begin{aligned} \frac{\partial f}{\partial \mathbf{A}} &= -\mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B}) + \mathbf{A}(\mathbf{C}^\mathsf{T}\mathbf{C} * \mathbf{B}^\mathsf{T}\mathbf{B}) \\ \frac{\partial f}{\partial \mathbf{B}} &= -\mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A}) + \mathbf{B}(\mathbf{C}^\mathsf{T}\mathbf{C} * \mathbf{A}^\mathsf{T}\mathbf{A}) \\ \frac{\partial f}{\partial \mathbf{C}} &= -\mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A}) + \mathbf{C}(\mathbf{B}^\mathsf{T}\mathbf{B} * \mathbf{A}^\mathsf{T}\mathbf{A}) \end{aligned}$$

cp_opt

$$f_{\mathcal{W}}(\mathbf{A},\mathbf{B},\mathbf{C}) = \| \mathcal{W} * (\mathcal{X} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]) \|^2 = \| \mathcal{Y} - \mathcal{Z} \|^2$$

$$\nabla f_{\mathcal{W}} = \begin{bmatrix} \frac{\partial f_{\mathcal{W}}}{\partial \mathbf{a}_1} \\ \vdots \\ \frac{\partial f_{\mathcal{W}}}{\partial \mathbf{a}_R} \\ \frac{\partial f_{\mathcal{W}}}{\partial \mathbf{b}_1} \\ \vdots \\ \frac{\partial f_{\mathcal{W}}}{\partial \mathbf{b}_R} \\ \frac{\partial f_{\mathcal{W}}}{\partial \mathbf{c}_1} \\ \vdots \\ \frac{\partial f_{\mathcal{W}}}{\partial \mathbf{c}_R} \end{bmatrix} \quad \begin{aligned} \frac{\partial f_{\mathcal{W}}}{\partial \mathbf{A}} &= 2(\mathbf{Z}_{(1)} - \mathbf{Y}_{(1)})(\mathbf{C} \odot \mathbf{B}) \\ \frac{\partial f_{\mathcal{W}}}{\partial \mathbf{B}} &= 2(\mathbf{Z}_{(2)} - \mathbf{Y}_{(2)})(\mathbf{C} \odot \mathbf{A}) \\ \frac{\partial f_{\mathcal{W}}}{\partial \mathbf{C}} &= 2(\mathbf{Z}_{(3)} - \mathbf{Y}_{(3)})(\mathbf{B} \odot \mathbf{A}) \end{aligned}$$

cp_wopt

Poblano Toolbox
Gradient-based optimization algorithms

ncg, lbfgs, …