Topics on High-Dimensional Data Analytics

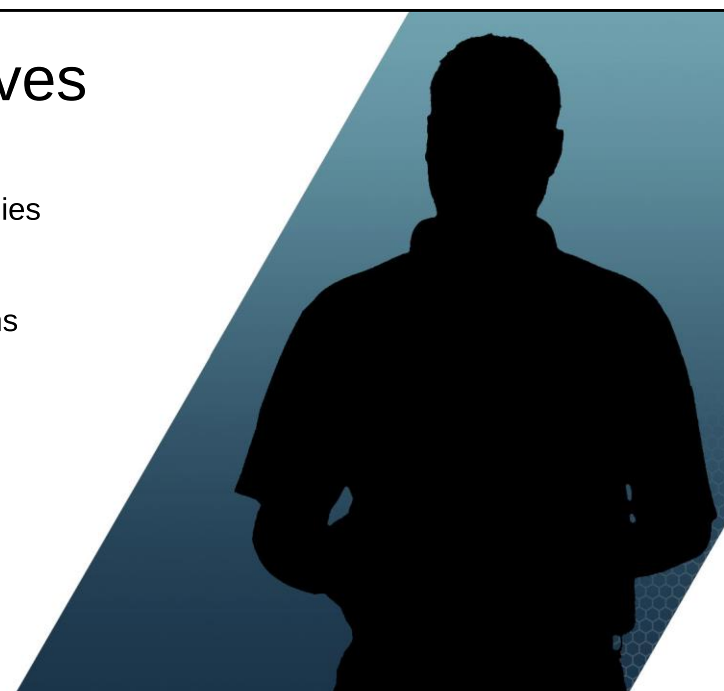
Tensor Data Analysis

Kamran Paynabar, Ph.D.
Associate Professor
School of Industrial & Systems Engineering

Multilinear Algebra
Preliminaries I

Learning Objectives

- Define a tensor
- Define some tensor terminologies
- Apply inner product and outer product
- Perform basic tensor operations

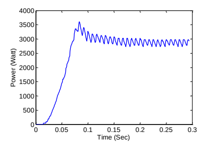


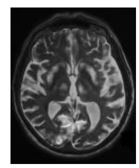
What is Tensor?

Tensor is a multidimensional array, examples include:

Scalars, i.e., 13

Vectors, i.e., $(13, 42, 2011)$

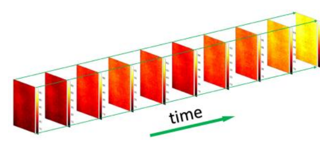




Matrices, i.e., $\begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$

Matrix sequences, i.e., $\begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix} \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix} \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$

…



Notations and Terminology

Letter notations

Scalars: Lowercase letters, e.g., x

Vectors: Boldface lowercase letters, i.e., $\boldsymbol{x}$

Matrices: Boldface capital letters, e.g., $\boldsymbol{X}$

Higher-order Tensors: Boldface Euler script letters, e.g., $\mathcal{X}$

Notations

The i^{th} entry of a vector $\boldsymbol{x}$: x_i $\quad (13, 42, 2011)$

The $(i, j)^{\text{th}}$ element of a matrix $\boldsymbol{X}$: x_{ij} $\quad \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$

The $(i, j, k)^{\text{th}}$ element of a 3-order tensor $\mathcal{X}$: x_{ijk} $\quad \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix} \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix} \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$

The n^{th} matrix in a sequence: $\boldsymbol{X}^{(n)}$

Subarrays are formed when a subset of the indices is fixed

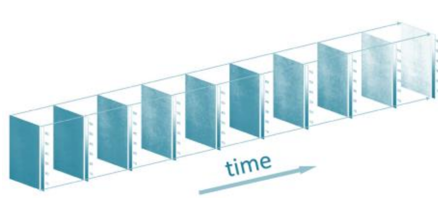
- The ith row of matrix $\boldsymbol{X}$: $\boldsymbol{x}_{i:}$
- The jth column of matrix $\boldsymbol{X}$: $\boldsymbol{x}_{:j}$

Terminology

Order: The number of dimensions of a tensor, also known as **ways** or **modes**

(a) 1-order tensor *(b) 2-order tensor* *(c) 3-order tensor*

e.g., an image stream is a 3-order tensor



Terminology

Fibers: A fiber, the higher order analogue of matrix row and column, is defined by fixing every index but one, e.g.,

- A matrix column is a mode-1 fiber and a matrix row is a mode-2 fiber
- Third-order tensors have column, row, and tube fibers
- Extracted fibers from a tensor are assumed to be oriented as column vectors.

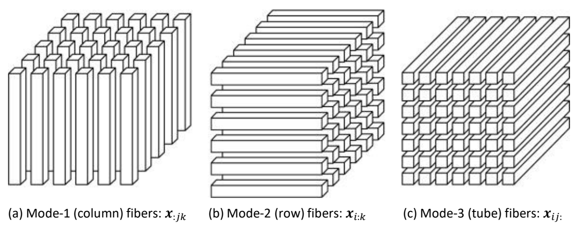


Fig. Fibers of a third-order tensor

Terminology

Slices: Two-dimensional sections of a tensor, defined by fixing all but two indices

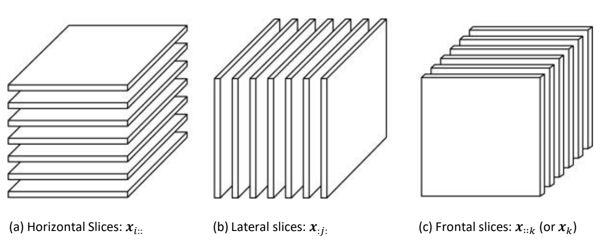


Fig. Slices of a third-order tensor

Terminology

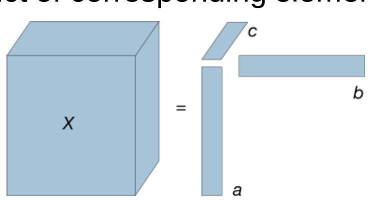
Norm: The norm of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the square root of the sum of the squares of all its elements

$$\| \mathcal{X} \| = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \cdots i_N}^2}.$$

This is analogous to the matrix Frobenius norm, which is denoted $\|\boldsymbol{A}\|_F$ for matrix $\boldsymbol{A}$

Basic Operations

Outer Product: A multi-way vector outer product is a tensor where each element is the product of corresponding elements in vectors



$X = \boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c}$

Inner product: Suppose $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ are same-sized tensors. Their inner product is defined by the sum of the products of their entries

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \cdots i_N} y_{i_1 i_2 \cdots i_N}.$$

Example

MATLAB code

```
X = tensor(rand(4,2,3))
Y = tensor(rand(4,2,3))
Z = innerprod(X,Y)
```

```
X is a tensor of size 4 x 2 x 3
    X(:,:,1) =
        0.8147    0.6324
        0.9058    0.0975
        0.1270    0.2785
        0.9134    0.5469
    X(:,:,2) =
        0.9575    0.9572
        0.9649    0.4854
        0.1576    0.8003
        0.9706    0.1419
    X(:,:,3) =
        0.4218    0.6557
        0.9157    0.0357
        0.7922    0.8491
        0.9595    0.9340
```

```
Y is a tensor of size 4 x 2 x 3
    Y(:,:,1) =
        0.6787    0.6555
        0.7577    0.1712
        0.7431    0.7060
        0.3922    0.0318
    Y(:,:,2) =
        0.2769    0.6948
        0.0462    0.3171
        0.0971    0.9502
        0.8235    0.0344
    Y(:,:,3) =
        0.4387    0.1869
        0.3816    0.4898
        0.7655    0.4456
        0.7952    0.6463
```

```
Z =

    8.0717
```

Basic Operations

- Tensor matricization**,** aka unfolding and flattening**,** unfolds an N-way tensor into a matrix
- **Mode-*n*** matricization arranges the mode-*n* fibers as columns of a matrix, which denoted by $\boldsymbol{X}_{(n)}$
- Vectorization of a tensor, denoted by $\mathrm{vec}(\mathcal{X})$, is transforming a tensor to a vector.

MATLAB code

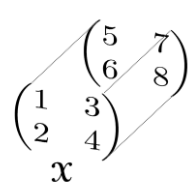
```
X1=[1 3;2 4];
X2=[5 7;6 8];
M = ones(2,2,2); %<-- A 2 x 2 x 2 array.
X = tensor(M) %<-- Convert to a tensor object.
X(:,:,1)=X1;
X(:,:,2)=X2;
A = tenmat(X,1) %<-- Mode-1 matricization.
B = tenmat(X,2) %<-- Mode-2 matricization.
C = tenmat(X,3) %<-- Mode-3 matricization.
```

$$\mathbf{X}_1 = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \qquad \mathbf{X}_2 = \begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix}$$

$$\mathbf{X}_{(1)} = \begin{pmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{pmatrix}$$

$$\mathbf{X}_{(2)} = \begin{pmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{pmatrix}$$

$$\mathbf{X}_{(3)} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$



Topics on High-Dimensional Data Analytics

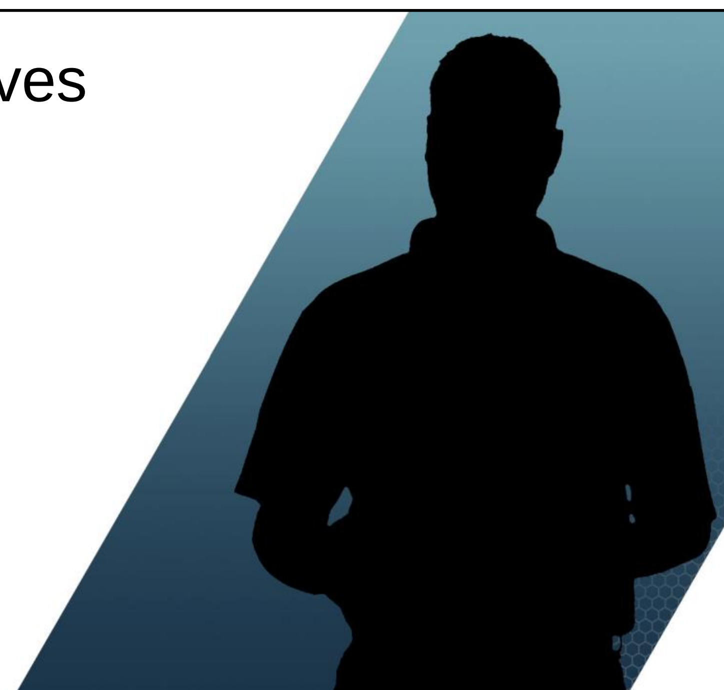
Tensor Data Analysis

Kamran Paynabar, Ph.D.
Associate Professor
School of Industrial & Systems Engineering

Multilinear Algebra
Preliminaries II

Learning Objectives

- Compute tensor multiplication
- Utilize Kronecker product
- Define Khatri-Rao product
- Compute Hadamard product



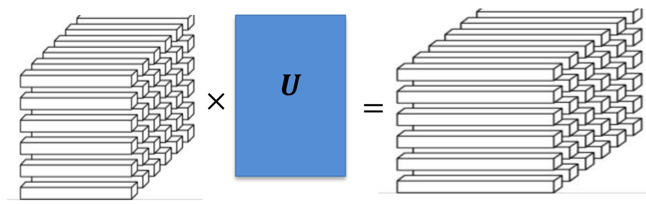
Tensor Multiplication

- The n-mode product is referred to as multiplying a tensor by a matrix (or a vector) in mode n.
- The n-mode (matrix) product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n \times \cdots \times I_N}$ with a matrix $\boldsymbol{U} \in \mathbb{R}^{J \times I_n}$ is denoted by $\mathcal{X} \times_{\mathrm{n}} \boldsymbol{U}$ and is of size $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$. We have

$$\mathcal{Y} = (\mathcal{X} \times_n \boldsymbol{U})_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_n \cdots i_N} u_{j i_n}$$

$$\boldsymbol{Y}_{(n)} = \boldsymbol{U}\, \boldsymbol{X}_{(n)}$$

- Example:
  Suppose $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\boldsymbol{U} \in \mathbb{R}^{J \times I_2}$.
  The n-mode product is obtained by multiplying each row fiber by Matrix $\boldsymbol{U}$.



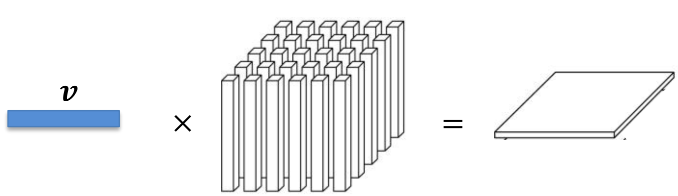
(b) Mode-2 (row) fibers: $\boldsymbol{x}_{i:k}$ (b) Mode-2 (row) fibers: $\boldsymbol{x}_{i:k}$

Tensor Multiplication

- The n-mode (vector) product of a tensor $\mathcal{X} \in \mathbb{R}^{\mathrm{I}_1 \times \mathrm{I}_2 \times \cdots \times \mathrm{I}_\mathrm{N}}$ with a vector $\mathrm{v} \in \mathbb{R}^{\mathrm{I}_\mathrm{n}}$ is denoted by $\mathcal{X}\, \bar{\times}_{\mathrm{n}}\, \boldsymbol{v}$. The result is of order $\mathrm{N}-1$, i.e., Elementwise,

$$\mathcal{Y} = (\mathcal{X}\, \bar{\times}_n\, \boldsymbol{v}\,)_{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} v_{i_n}$$

- Example: Suppose $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\boldsymbol{v} \in \mathbb{R}^{I_1}$. The n-mode product is obtained by inner product of $\boldsymbol{v}$ and each column fiber.



(a) Mode-1 (column) fibers: $\boldsymbol{x}_{:jk}$

Example: n-mode Matrix Product

```
X is a tensor of size 5 x 3 x 4
    X(:,:,1) =
        0.1788    0.6959    0.3196
        0.4229    0.6999    0.5309
        0.0942    0.6385    0.6544
        0.5985    0.0336    0.4076
        0.4709    0.0688    0.8200
    X(:,:,2) =
        0.7184    0.6110    0.1537
        0.9686    0.7788    0.2810
        0.5313    0.4235    0.4401
        0.3251    0.0908    0.5271
        0.1056    0.2665    0.4574
    X(:,:,3) =
        0.8754    0.2407    0.0680
        0.5181    0.6761    0.2548
        0.9436    0.2891    0.2240
        0.6377    0.6718    0.6678
        0.9577    0.6951    0.8444
    X(:,:,4) =
        0.3445    0.3868    0.4609
        0.7805    0.9160    0.7702
        0.6753    0.0012    0.3225
        0.0067    0.4624    0.7847
        0.6022    0.4243    0.4714
```

```
Y is a tensor of size 4 x 3 x 4
    Y(:,:,1) =
        0.8701    0.6687    1.3603
        0.6333    0.5696    1.0335
        0.9018    1.5605    1.7189
        0.4461    0.6679    0.8154
    Y(:,:,2) =
        0.8757    0.6765    1.0405
        0.7193    0.6752    0.6418
        1.7455    1.4742    1.0869
        0.7434    0.6743    0.4878
    Y(:,:,3) =
        1.8472    1.2401    1.2046
        1.3346    0.9450    0.8563
        2.5190    1.4227    1.0629
        1.2335    0.6785    0.5617
    Y(:,:,4) =
        0.9823    0.7531    1.2445
        0.9036    0.7580    0.8923
        1.6733    1.2015    1.5796
        0.7514    0.5819    0.7248
```

```
A =

    0.0358    0.1527    0.7384    0.7655    0.5762
    0.1759    0.3411    0.2428    0.1887    0.6834
    0.7218    0.6074    0.9174    0.2875    0.5466
    0.4735    0.1917    0.2691    0.0911    0.4257
```

MATLAB code

```
X = tenrand([5,3,4])
A = rand(4,5)
Y = ttm(X, A, 1) %<-- X times A in mode-1.
```

Example: n-mode Vector Product

```
X is a tensor of size 5 x 3 x 4
    X(:,:,1) =
        0.7212    0.7150    0.1978
        0.1068    0.9037    0.0305
        0.6538    0.8909    0.7441
        0.4942    0.3342    0.5000
        0.7791    0.6987    0.4799
    X(:,:,2) =
        0.9047    0.5767    0.4899
        0.6099    0.1829    0.1679
        0.6177    0.2399    0.9787
        0.8594    0.8865    0.7127
        0.8055    0.0287    0.5005
    X(:,:,3) =
        0.4711    0.5216    0.1499
        0.0596    0.0967    0.6596
        0.6820    0.8181    0.5186
        0.0424    0.8175    0.9730
        0.0714    0.7224    0.6490
    X(:,:,4) =
        0.8003    0.1332    0.0605
        0.4538    0.1734    0.3993
        0.4324    0.3909    0.5269
        0.8253    0.8314    0.4168
        0.0835    0.8034    0.6569
```

MATLAB code

```
X = tenrand([5,3,4])
A = rand(5,1)
Y = ttv(X, A, 1) %<-- X times A in mode 1.
```

```
A =

    0.6280
    0.2920
    0.4317
    0.0155
    0.9841
```

```
Y is a tensor of size 3 x 4
    Y(:,:) =
        1.5406    1.8188    0.6786    0.9167
        1.7902    0.5611    1.4326    1.1064
        0.9343    1.2827    1.1643    1.0348
```

Kronecker Product

The Kronecker product of matrices $\boldsymbol{A} \in \mathbb{R}^{I \times J}$ and $\boldsymbol{B} \in \mathbb{R}^{K \times L}$ is denoted by $\boldsymbol{A} \otimes \boldsymbol{B}$. The result is a matrix of size $(IK) \times (JL)$ and defined by

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots & a_{1J}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots & a_{2J}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\boldsymbol{B} & a_{I2}\boldsymbol{B} & \cdots & a_{IJ}\boldsymbol{B} \end{bmatrix}$$

$$= [\boldsymbol{a}_1 \otimes \boldsymbol{b}_1 \quad \boldsymbol{a}_1 \otimes \boldsymbol{b}_2 \quad \boldsymbol{a}_1 \otimes \boldsymbol{b}_3 \quad \cdots \quad \boldsymbol{a}_J \otimes \boldsymbol{b}_{L-1} \quad \boldsymbol{a}_J \otimes \boldsymbol{b}_L]$$

Example: Kornecker Product

MATLAB code
A=[1 2;3 4;5 6] *B=[1 2 3;4 5 6]* *C=kron(A,B)*

```
A =

     1     2
     3     4
     5     6

B =

     1     2     3
     4     5     6

C =

     1     2     3     2     4     6
     4     5     6     8    10    12
     3     6     9     4     8    12
    12    15    18    16    20    24
     5    10    15     6    12    18
    20    25    30    24    30    36
```

Khatri-Rao Product

The **Khatri-Rao product** is the "matching columnwise" **Kronecker product**
Given matrices $\boldsymbol{A} \in \mathbb{R}^{I \times K}$ and $\boldsymbol{B} \in \mathbb{R}^{J \times K}$, their **Khatri-Rao product**, denoted by $\boldsymbol{A} \odot \boldsymbol{B}$, is a matrix of size $(IJ) \times (K)$ and computed by

$$\boldsymbol{A} \odot \boldsymbol{B} = [\boldsymbol{a}_1 \otimes \boldsymbol{b}_1 \quad \boldsymbol{a}_2 \otimes \boldsymbol{b}_2 \quad \cdots \quad \boldsymbol{a}_K \otimes \boldsymbol{b}_K]$$

If $\boldsymbol{a}$ and $\boldsymbol{b}$ are vectors, then the Khatri-Rao and Kronecker products are identical, i.e.,

$$\boldsymbol{a} \otimes \boldsymbol{b} = \boldsymbol{a} \odot \boldsymbol{b}$$

Example: Khatri-Rao Product

MATLAB code
A=[1 2;3 4;5 6] *B=[1 2;3 4]* *C=khatrirao(A,B)*

$$\boldsymbol{A} = \begin{matrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{matrix}$$

$$\boldsymbol{B} = \begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}$$

$$\boldsymbol{C} = \boldsymbol{A} \odot \boldsymbol{B} = \begin{matrix} 1 & 4 \\ 3 & 8 \\ 3 & 8 \\ 9 & 16 \\ 5 & 12 \\ 15 & 24 \end{matrix}$$

Hadamard Product

The **Hadamard product**, of matrices $\boldsymbol{A} \in \mathbb{R}^{I \times J}$ and $\boldsymbol{B} \in \mathbb{R}^{I \times J}$, denoted by $\boldsymbol{A} * \boldsymbol{B}$, is defined by the elementwise matrix product, i.e., the resulting matrix $I \times J$ is computed by

$$\boldsymbol{A} * \boldsymbol{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{J1} & a_{I2}b_{J2} & \cdots & a_{IJ}b_{IJ} \end{bmatrix}$$

Example: Hadamard Product

MATLAB code
A=[1 2;3 4;5 6] *B=[1 2;3 4;5 6]* *C=A.*B*

```
A =

     1     2
     3     4
     5     6


B =

     1     2
     3     4
     5     6
```

```
C =

     1     4
     9    16
    25    36
```

Topics on High-Dimensional Data Analytics

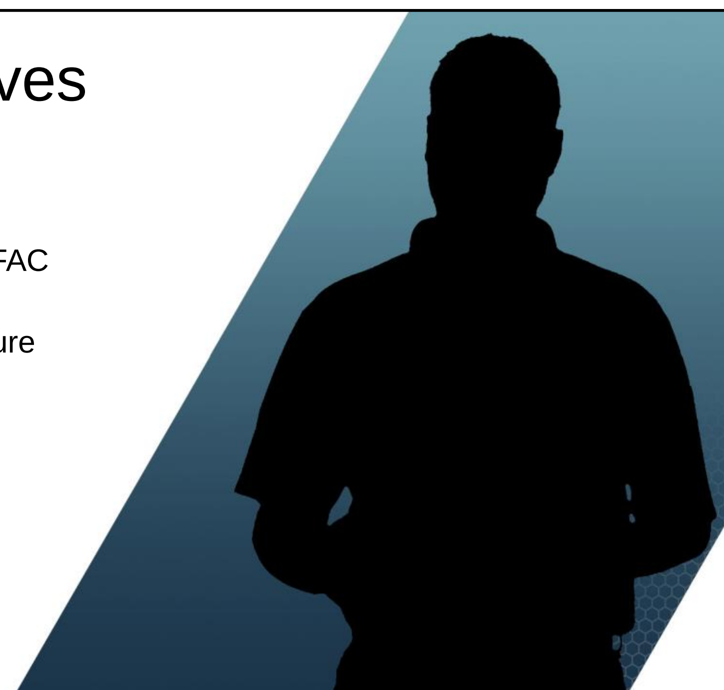
Tensor Data Analysis

Kamran Paynabar, Ph.D.
Associate Professor
School of Industrial & Systems Engineering

Tensor Decomposition Methods I
- CP Decomposition

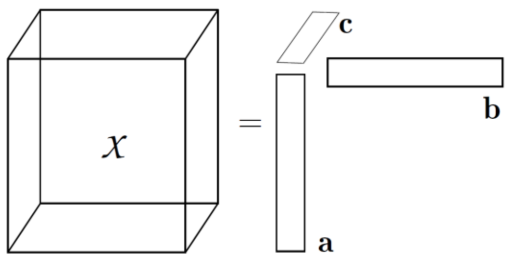
Learning Objectives

- Define tensor Ranks
- Define Rank-one tensors
- Perform CANDECOMP/PARAFAC (CP) Decomposition
- Identify low dimensional structure using CP



Rank-One Tensor

A **Rank-One Tensor** can be created by the outer product of multiple vectors, e.g., a 3-order rank-one tenor is obtained by



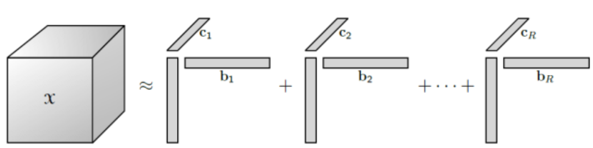
$$\mathcal{X} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$$

CANDECOMP/PARAFAC (CP) Decomposition

The **CP decomposition** factorizes a tensor into a **sum** of component **rank-one** tensors, e.g., given a third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, CP decomposition is given by

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

R is a positive integer, $\boldsymbol{a}_r \in \mathbb{R}^I, \boldsymbol{b}_r \in \mathbb{R}^J, \boldsymbol{c}_r \in \mathbb{R}^K$, for $r = 1, \ldots, R$



- If R is the rank of higher-tensor then the CP decomposition will be exact and unique.

Rank of Tensor

Rank of a tensor $\mathcal{X}$, denoted by rank($\mathcal{X}$), is the smallest number of rank-one tensors whose sum can generate $\mathcal{X}$. e.g.,

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

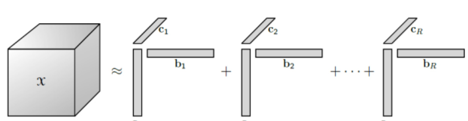
R is a positive integer, $\boldsymbol{a}_r \in \mathbb{R}^I, \boldsymbol{b}_r \in \mathbb{R}^J, \boldsymbol{c}_r \in \mathbb{R}^K$, for $r = 1, \dots, R$

- Determining the rank of a tensor is an NP-hard problem. Some weaker upper bounds, however, exist that helps restrict the rank space, e.g., for $\mathcal{X}^{I\times J\times K}$,

$$\text{rank}(\mathcal{X}) \leq \min\{IJ, JK, IK\}.$$

CANDECOMP/PARAFAC (CP) Decomposition

We can create factor matrices by concatenating the corresponding rank-one vectors from the rank-one tensors. For example, $\boldsymbol{A} = [\boldsymbol{a}_1 \quad \boldsymbol{a}_2 \quad \cdots \quad \boldsymbol{a}_R]$, $\boldsymbol{B} = [\boldsymbol{b}_1 \quad \boldsymbol{b}_2 \quad \cdots \quad \boldsymbol{b}_R]$, and $\boldsymbol{C} = [\boldsymbol{c}_1 \quad \boldsymbol{c}_2 \quad \cdots \quad \boldsymbol{c}_R]$.



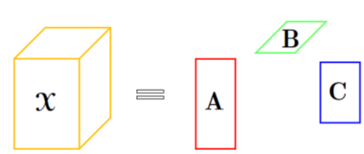
The CP decomposition can be rewritten by factor matrices in matrix form:

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

$$\boldsymbol{X}_{(1)} \approx \boldsymbol{A}(\boldsymbol{C} \odot \boldsymbol{B})^\top$$

$$\boldsymbol{X}_{(2)} \approx \boldsymbol{B}(\boldsymbol{C} \odot \boldsymbol{A})^\top$$

$$\boldsymbol{X}_{(3)} \approx \boldsymbol{C}(\boldsymbol{B} \odot \boldsymbol{A})^\top$$



CP Decomposition

If the column of factor matrices are normalized, CP can be rewritten by

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

$$\boldsymbol{X}_{(2)} \approx \boldsymbol{B}\boldsymbol{\Lambda}(\boldsymbol{C} \odot \boldsymbol{A})^{\top};$$

For an *n*-order tensor, CP is given by

$$\mathcal{X} \approx [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(n)}]\!] = \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(n)}$$

$$\boldsymbol{X}_{(k)} \approx \boldsymbol{A}^{(k)}\boldsymbol{\Lambda}\left(\boldsymbol{A}^{(n)} \odot \cdots \odot \boldsymbol{A}^{(k-1)} \odot \boldsymbol{A}^{(k+1)} \odot \cdots \odot \boldsymbol{A}^{(1)}\right)^{\top}.$$

CP Decomposition: Computation

CP decomposition can be obtained by solving the following optimization problem:

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{\lambda}} \|\mathcal{X} - [\![\boldsymbol{\lambda}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]\|^2 = \left\|\mathcal{X} - \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r\right\|^2$$

S.T. $\|\boldsymbol{a}_r\|$=1; $\|\boldsymbol{b}_r\|$=1; $\|\boldsymbol{c}_r\|$=1; for r = 1, 2, … , R

Alternative Least Square (ALS) can be used to solve the optimization problem. ALS iteratively solves the optimization model for one matrix given all other matrices. For example given $\boldsymbol{B}$ and $\boldsymbol{C}$, $\widetilde{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{\Lambda}$ is given by

$$\min_{\widetilde{\boldsymbol{A}}} \left\|\boldsymbol{X}_{(1)} - \widetilde{\boldsymbol{A}}(\boldsymbol{C} \odot \boldsymbol{B})^{\top}\right\|_{\mathrm{F}} \rightarrow \widetilde{\boldsymbol{A}} = \boldsymbol{X}_{(1)}(\boldsymbol{H}^{\top}\boldsymbol{H})^{-1}\boldsymbol{H}^{\top}; \text{ where } \boldsymbol{H} = (\boldsymbol{C} \odot \boldsymbol{B})^{\top}$$

After finding $\widetilde{\boldsymbol{A}}$, $\boldsymbol{A}$ is calculated by $\boldsymbol{a}_r = \frac{\tilde{a}_r}{\|\tilde{a}_r\|}$

CP Decomposition: Computation

The ALS algorithm for an n-mode tensor for a given *R*:

Initialize (e.g., random entries) $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n \times R}$

Repeat until convergence

for k=1,…, n do

$$\boldsymbol{V} = \boldsymbol{A}^{(1)^\top} \boldsymbol{A}^{(1)} * \cdots * \boldsymbol{A}^{(k-1)^\top} \boldsymbol{A}^{(k-1)} * \boldsymbol{A}^{(k+1)^\top} \boldsymbol{A}^{(k+1)} * \cdots * \boldsymbol{A}^{(n)^\top} \boldsymbol{A}^{(n)}$$

$$\boldsymbol{A}^{(k)} = \boldsymbol{X}_{(k)} \left(\boldsymbol{A}^{(n)} \odot \cdots \odot \boldsymbol{A}^{(k+1)} \odot \boldsymbol{A}^{(k-1)} \odot \cdots \odot \boldsymbol{A}^{(1)} \right) (\boldsymbol{V}^\top \boldsymbol{V})^{-1} \boldsymbol{V}^\top$$

Normalize columns of $\boldsymbol{A}^{(k)}$ and store its norm as $\boldsymbol{\lambda}$

end for

Return $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(n)}$ and $\boldsymbol{\lambda}$

Some convergence criteria include little or no change in the value of objective function, no or little change in the factor matrices, and reaching a predefined number of iterations.

Example: CP Decomposition

```
>> X = sptenrand([5 4 3], 10)
P = parafac_als(X,2)
X is a sparse tensor of size 5 x 4 x 3 with 10 nonzeros
    (1,1,3)    0.5201
    (1,4,3)    0.3477
    (2,4,2)    0.1500
    (3,2,2)    0.5861
    (3,2,3)    0.2621
    (3,3,3)    0.0445
    (4,1,1)    0.7549
    (4,2,1)    0.2428
    (4,4,2)    0.4424
    (5,4,2)    0.6878

CP_ALS:
 Iter  1: f = 2.117500e-01 f-delta = 2.1e-01
 Iter  2: f = 2.367578e-01 f-delta = 2.5e-02
 Iter  3: f = 2.671289e-01 f-delta = 3.0e-02
 Iter  4: f = 3.041815e-01 f-delta = 3.7e-02
 Iter  5: f = 3.366513e-01 f-delta = 3.2e-02
 Iter  6: f = 3.632238e-01 f-delta = 2.7e-02
 Iter  7: f = 3.785624e-01 f-delta = 1.5e-02
 Iter  8: f = 3.832554e-01 f-delta = 4.7e-03
 Iter  9: f = 3.842076e-01 f-delta = 9.5e-04
 Iter 10: f = 3.843710e-01 f-delta = 1.6e-04
 Iter 11: f = 3.843977e-01 f-delta = 2.7e-05
 Final f = 3.843977e-01
```

```
P is a ktensor of size 5 x 4 x 3
    P.lambda = [ 0.83143       0.793 ]
    P.U{1} =
            -0.0000    0.0057
             0.1804    0.0000
             0.0013    0.0010
             0.5321    1.0000
             0.8272    0.0000
    P.U{2} =
            -0.0002    0.9520
             0.0009    0.3061
            -0.0000    0.0000
             1.0000    0.0006
    P.U{3} =
            -0.0003    1.0000
             1.0000    0.0002
            -0.0000    0.0037
```

MATLAB code
X = sptenrand([5 4 3], 10) *P = parafac_als(X,2)*

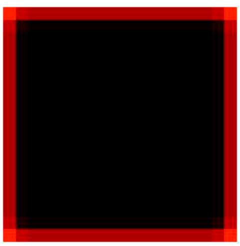
Example: Heat Transfer Data

An image is generated from the following heat transfer process:

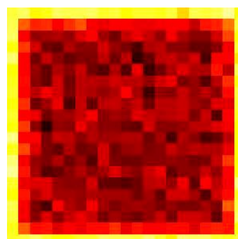
$$\frac{\partial \mathcal{S}_i}{\partial t} = \alpha_i\left(\frac{\partial^2 \mathcal{S}_i}{\partial x^2} + \frac{\partial^2 \mathcal{S}_i}{\partial y^2}\right)$$

Thermal diffusivity coefficient: $\alpha_i \sim unif\{0.5 \times 10^{-5}, 1.5 \times 10^{-5}\}$

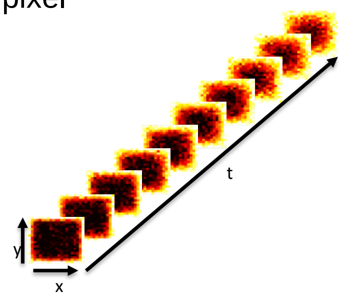
Noise: i.i.d. $\varepsilon \sim N(0, 0.01)$ are added to each pixel



(a) Without noise

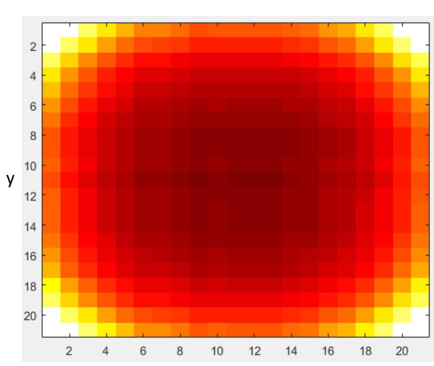


(b) With noise

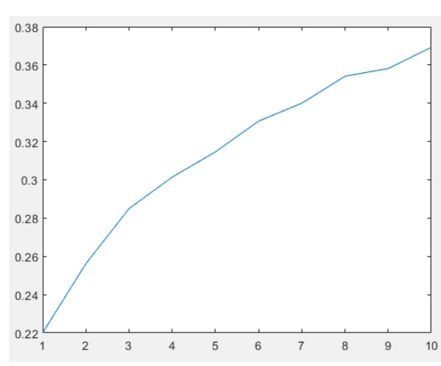


Example: Decoupling Spatial and Temporal

For R = 1, the columns $\boldsymbol{a}_1, \boldsymbol{b}_1,$ and $\boldsymbol{c}_r$ are computed.



$\boldsymbol{a}_1 \circ \boldsymbol{b}_1$ Spatial pattern

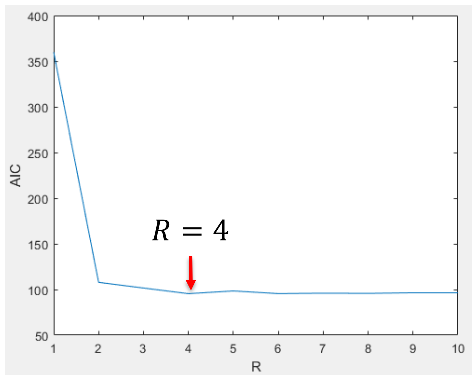


$\boldsymbol{c}_r$ Temporal pattern

Example: Rank Selection

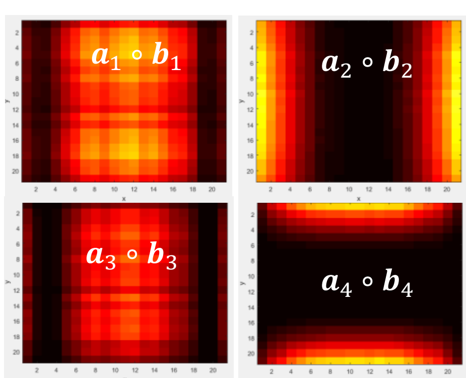
We can choose R using AIC $=2\left\|\mathcal{X}-\sum_{r=1}^{R}\lambda_r \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r\right\|^2 + 2k$

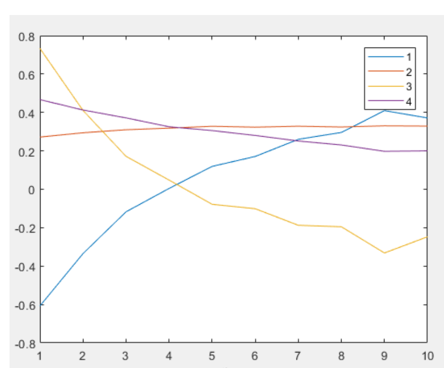
$$R \le \min(IJ, IK, JK) = 210$$



Example: Temporal and Spatial Patterns

For R = 4





Topics on High-Dimensional Data Analytics

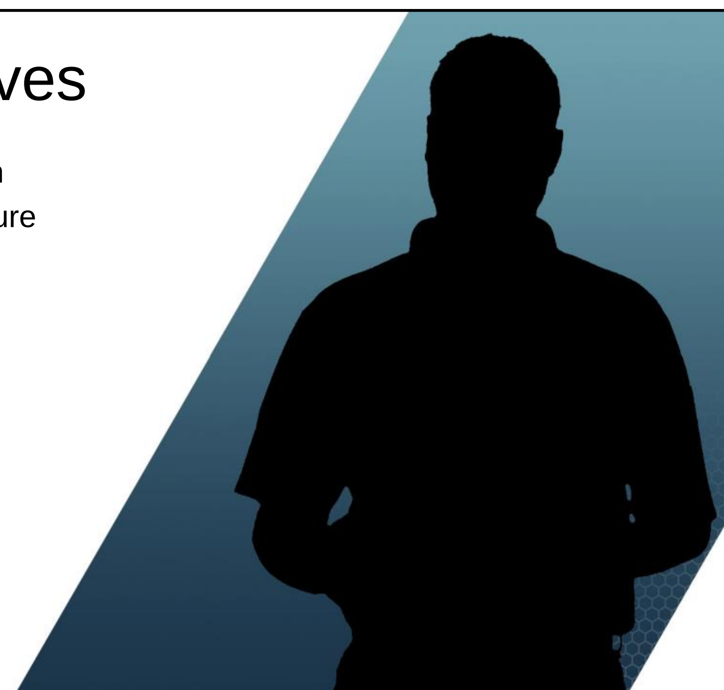
Tensor Data Analysis

Kamran Paynabar, Ph.D.
Associate Professor
School of Industrial & Systems Engineering

Tensor Decomposition Methods II – Tucker Decomposition

Learning Objectives

- Perform Tucker Decomposition
- Identify low dimensional structure using Tucker

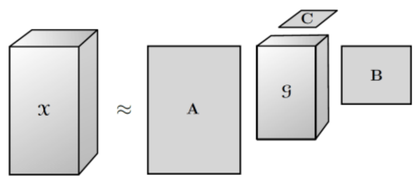


Tucker Decomposition

The **Tucker decomposition** decomposes a tensor into a core tensor multiplied (or transformed) by a set of factorizing matrix along each mode. For example, a 3-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$

$$\mathcal{X} \approx \mathcal{G} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C} = \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} g_{pqr}\, \boldsymbol{a}_p \circ \boldsymbol{b}_q \circ \boldsymbol{c}_r = [\![\mathcal{G}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$$

Where $\boldsymbol{A} \in \mathbb{R}^{I \times P}, \boldsymbol{B} \in \mathbb{R}^{J \times Q}$, $\boldsymbol{C} \in \mathbb{R}^{K \times R}, \mathcal{G} \in \mathbb{R}^{P \times Q \times R}$



The metricized from of the tucker can be written as

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

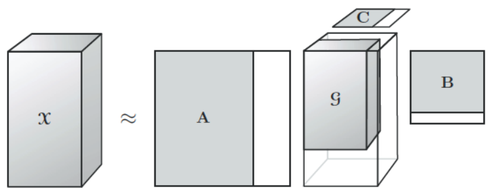
Tucker Decomposition

$$\mathcal{X} \approx \mathcal{G} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C} = \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} g_{pqr}\, \boldsymbol{a}_p \circ \boldsymbol{b}_q \circ \boldsymbol{c}_r = [\![\mathcal{G}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$$

- The core matrix captures the interaction among different modes and since often $P < I$, $Q < J$, $R < K$, the core tensor is considered as the compressed version of the original tensor.
- In most cases, it is assumed that factor matrices are column-wise orthogonal (not required though).
- CP decomposition can be seen as a special case of Tucker where the core matrix is super-diagonal and *P* = *Q* = *R*.

n-Rank of Tensor

- Consider an n-mode tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$. The n-rank of this tensor, denoted by $R_n = \mathrm{rank}_n(\mathcal{X})$, is defined by the column rank of the mode-n fibers, $\boldsymbol{X}_{(\boldsymbol{n})}$.
- The *n*-rank is different from rank of the tensor introduced before.
- Given the set of *n*-ranks, we can find the exact Tucker decomposition of $\mathcal{X}$ for this set.
- If the rank used for decomposition is smaller than the corresponding n-rank, truncated Tucker decomposition is obtained.



Tucker Decomposition: HOSVD

- Higher Order Singular Value Decomposition (HOSVD) is a simple method for performing Tucker decomposition.
- The idea is to find the factor matrices for each mode separately such that the maximum variation of the mode is captured.
- This can be done by performing SVD decomposition for each mode-*k* fiber of the tensor and keep the R_k leading left singular values of the matrix $\boldsymbol{X}_{(\boldsymbol{k})}$, denoted by $\boldsymbol{A}^{(\boldsymbol{k})}$.
- The core tensor is then obtained by

$$\mathcal{G} = \mathcal{X} \times_1 \boldsymbol{A}^{(\boldsymbol{1})\top} \times_2 \boldsymbol{A}^{(\boldsymbol{2})\top} \times_3 \ldots \times_n \boldsymbol{A}^{(\boldsymbol{n})\top}$$

- The truncated HOSVD is not optimal with respect to the least squared lack of fit measure.

Tucker Decomposition: ALS

- A natural approach for finding the Tucker decomposition components is to minimize the squared error between the original tensor and the reconstructed one. That is

$$\min_{\mathcal{G};\boldsymbol{A}^{(1)},\boldsymbol{A}^{(2)},\dots,\boldsymbol{A}^{(n)}} \left\| \mathcal{X} - [\![\mathcal{G};\boldsymbol{A}^{(1)},\boldsymbol{A}^{(2)},\dots,\boldsymbol{A}^{(n)}]\!] \right\|^2$$

S.T. $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$; $\boldsymbol{A}^{(k)} \in \mathbb{R}^{I_k \times R_k}$ *and* $\boldsymbol{A}^{(k)}$ *column-wise orthogonal for all k's*

- It can be shown that this problem can be reduced to

$$\min_{\mathcal{G};\boldsymbol{A}^{(1)},\boldsymbol{A}^{(2)},\dots,\boldsymbol{A}^{(n)}} \left\| \mathcal{X} - [\![\mathcal{G};\boldsymbol{A}^{(1)},\boldsymbol{A}^{(2)},\dots,\boldsymbol{A}^{(n)}]\!] \right\|^2 \equiv \max_{\boldsymbol{A}^{(1)},\dots,\boldsymbol{A}^{(n)}} \left\| \mathcal{X} \times_1 \boldsymbol{A}^{(1)\top} \times_2 \boldsymbol{A}^{(2)\top} \times_3 \dots \times_n \boldsymbol{A}^{(n)\top} \right\|^2$$

$$= \max_{\boldsymbol{A}^{(1)}\dots,\boldsymbol{A}^{(n)}} \left\| \boldsymbol{A}^{(k)\top} \boldsymbol{X}_{(k)} \left(\boldsymbol{A}^{(n)} \otimes \dots \otimes \boldsymbol{A}^{(k+1)} \otimes \boldsymbol{A}^{(k-1)} \otimes \dots \otimes \boldsymbol{A}^{(1)} \right) \right\|^2$$

S.T. $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$; $\boldsymbol{A}^{(k)} \in \mathbb{R}^{I_k \times R_k}$ *and* $\boldsymbol{A}^{(k)}$ *column-wise orthogonal for all k's*

Tucker Decomposition: ALS

- Using ALS, given all factor matrices but $\boldsymbol{A}^{(k)}$, the solution is determined by applying SVD decomposition on $\boldsymbol{X}_{(k)} \left(\boldsymbol{A}^{(n)} \otimes \dots \otimes \boldsymbol{A}^{(k+1)} \otimes \boldsymbol{A}^{(k-1)} \otimes \dots \otimes \boldsymbol{A}^{(1)} \right)$ and keeping the R_k leading left singular values

$$\max_{\boldsymbol{A}^{(k)}} \left\| \mathcal{X} \times_1 \boldsymbol{A}^{(1)\top} \times_2 \boldsymbol{A}^{(2)\top} \times_3 \dots \times_n \boldsymbol{A}^{(n)\top} \right\|^2 = \left\| \boldsymbol{A}^{(k)\top} \boldsymbol{X}_{(k)} \left(\boldsymbol{A}^{(n)} \otimes \dots \otimes \boldsymbol{A}^{(k+1)} \otimes \boldsymbol{A}^{(k-1)} \otimes \dots \otimes \boldsymbol{A}^{(1)} \right) \right\|^2$$

S.T. $\boldsymbol{A}^{(k)} \in \mathbb{R}^{I_k \times R_k}$ *and column-wise orthogonal for all k's*

Tucker Decomposition: Computation

The ALS algorithm for an n-mode tensor for a given $R_1, R_2, \dots, R_n$:

Initialize $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \dots, \boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ using HOSVD

Repeat until convergence

for k=1,…, n do

$$\boldsymbol{\mathcal{Y}} = \mathcal{X} \times_1 \boldsymbol{A}^{(1)\top} \times_2 \boldsymbol{A}^{(2)\top} \dots \times_{k-1} \boldsymbol{A}^{(k-1)} \times_{k+1} \boldsymbol{A}^{(k+1)} \dots \times_n \boldsymbol{A}^{(n)\top}$$

$\boldsymbol{A}^{(k)} \leftarrow$ the R_k leading left singular values of $\boldsymbol{Y}_{(\boldsymbol{k})}$

end for

$$\mathcal{G} = \mathcal{X} \times_1 \boldsymbol{A}^{(1)\top} \times_2 \boldsymbol{A}^{(2)\top} \times_3 \dots \times_n \boldsymbol{A}^{(n)\top}$$

Return $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \dots, \boldsymbol{A}^{(n)}$ and $\mathcal{G}$

Some convergence criteria include little or no change in the value of objective function, no or little change in the factor matrices, and reaching a predefined number of iterations.

Example: Tucker Decomposition

```
>> X = sptenrand([5 4 3], 10)
T = tucker_als(X,[2 2 1])  %<-- best rank(2,2,1) approximation
X is a sparse tensor of size 5 x 4 x 3 with 10 nonzeros
    (1,4,1)    0.9706
    (1,4,2)    0.8669
    (1,4,3)    0.0862
    (2,1,2)    0.3664
    (3,1,1)    0.3692
    (3,3,3)    0.6850
    (4,1,1)    0.5979
    (4,1,2)    0.7894
    (4,3,1)    0.3677
    (5,3,3)    0.2060

Tucker Alternating Least-Squares:
 Iter  1: fit = 5.288933e-01 fitdelta = 5.3e-01
 Iter  2: fit = 5.448539e-01 fitdelta = 1.6e-02
 Iter  3: fit = 5.450032e-01 fitdelta = 1.5e-04
 Iter  4: fit = 5.450044e-01 fitdelta = 1.1e-06
T is a ttensor of size 5 x 4 x 3
    T.core is a tensor of size 2 x 2 x 1
        T.core(:,:,1) =
        1.3028    0.0000
       -0.0000    1.0755
```

```
T.U{1} =
        1.0000   -0.0000
       -0.0000    0.2313
       -0.0000    0.2460
        0.0000    0.9413
        0.0000    0.0025
T.U{2} =
       -0.0000    0.9713
             0         0
        0.0000    0.2380
        1.0000   -0.0000
T.U{3} =
        0.7133
        0.6987
        0.0545
```

MATLAB code

```
X = sptenrand([5 4 3], 10)
T = tucker_als(X,[2 2 1])  %<-- best rank(2,2,1)
```

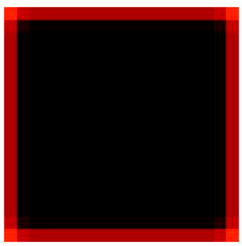
Example: Heat Transfer Data

An image is generated from the following heat transfer process:

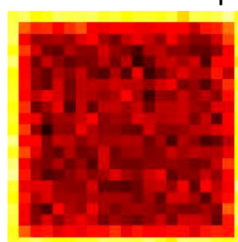
$$\frac{\partial \mathcal{S}_i}{\partial t} = \alpha_i\left(\frac{\partial^2 \mathcal{S}_i}{\partial x^2} + \frac{\partial^2 \mathcal{S}_i}{\partial y^2}\right)$$

Thermal diffusivity coefficient: $\alpha_i \sim unif\{0.5 \times 10^{-5}, 1.5 \times 10^{-5}\}$

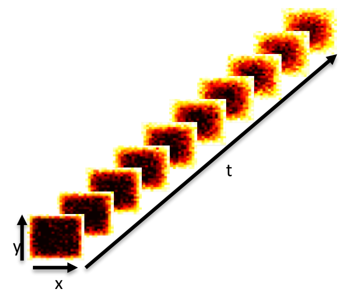
Noise: i.i.d. $\varepsilon \sim N(0, 0.01)$ are added to each pixel



(a) Without noise



(b) With noise



Example: Rank Selection

1-rank = 21; 2-rank = 21; 3-rank = 10

Try different combinations and use AIC to decide which one is optimal.

```
for i = 1:4
   for j = 1:4
      for k = 1:4
         T = tucker_als(X,[i,j,k]);
         T1 = tenmat(T,1);
         dif = tensor(X1-T1);
         err(i,j,k) = innerprod(dif,dif);
         AIC(i,j,k) = 2*err(i,j,k) + 2*(i+j+k);
      end
   end
end
```

val(:,:,1) =

363.9157	365.9159	367.9158	369.9161
365.9159	216.7511	218.7511	220.7511
367.9159	218.7511	219.2892	221.2446
369.9157	220.7511	221.3514	222.0706

val(:,:,2) =

365.9157	322.4362	324.4362	326.4362
329.8568	114.2685	111.6261	113.3280
331.8567	111.4599	108.5872	109.8128
333.8566	113.2581	110.0170	110.9051

val(:,:,3) =

367.9157	324.4362	323.9317	325.9412
331.8566	115.9026	110.7792	112.0531
331.3137	109.9953	$\boxed{104.6904}$	105.8061
333.3197	111.3266	105.[illegible]956	106.8353

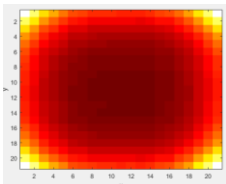
val(:,:,4) =

369.9158	326.4362	325.9331	327.2414
333.8566	117.8748	112.5831	113.3856
333.3140	111.7464	106.4217	106.9486
334.4661	112.7500	107.2565	107.9094

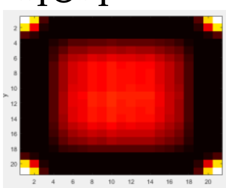
[3,3,3]

Example: Temporal and Spatial Patterns

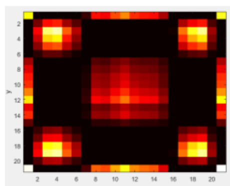
For rank = [3,3,3]



$\boldsymbol{a_1} \otimes \boldsymbol{b_1} \in \mathbb{R}^{21\times21}$



$\boldsymbol{a_2} \otimes \boldsymbol{b_2} \in \mathbb{R}^{21\times21}$



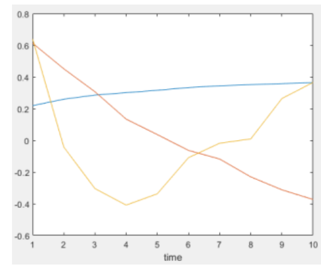
$\boldsymbol{a_3} \otimes \boldsymbol{b_3} \in \mathbb{R}^{21\times21}$

44.6120	0.5816	0.0773
0.6023	-8.6361	0.8254
0.0803	0.7911	-0.1804

-0.8042	4.6539	1.0258
4.2089	-3.6676	-0.7656
1.1440	-0.6997	0.1294

0.1118	-0.1497	0.9398
-0.1537	0.4542	-0.6264
1.0368	-0.7122	-0.2243

$\mathcal{G} \in \mathbb{R}^{3\times3\times3}$



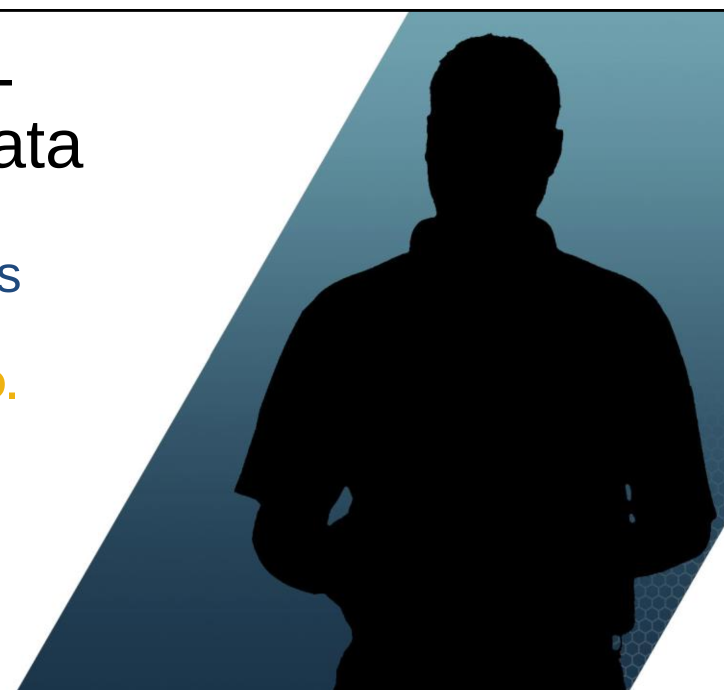
$\boldsymbol{C} \in \mathbb{R}^{10\times3}$

Topics on High-Dimensional Data Analytics

Tensor Data Analysis

Kamran Paynabar, Ph.D.
Associate Professor
School of Industrial & Systems Engineering

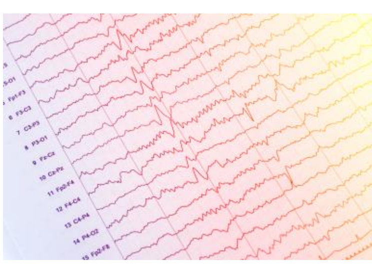
Tensor Analysis Applications I



Learning Objectives

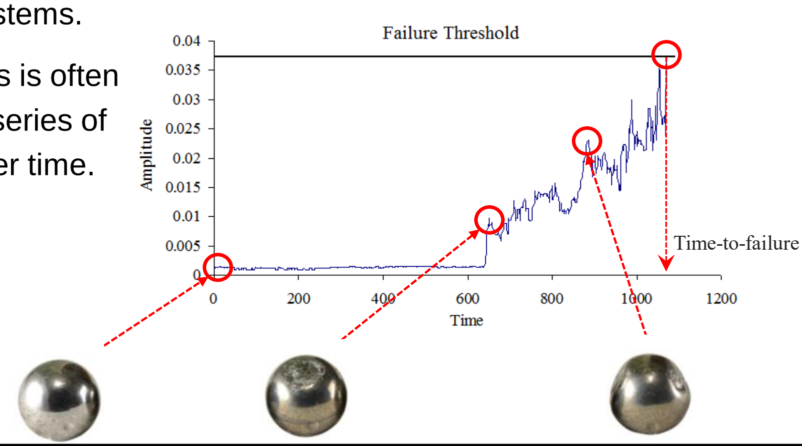
- Determine applications of CP and Tucker
- Use CP and Tucker in regression context.
- Perform regression analysis for Scalar Response and Tensor Predictors

Low Rank Representation



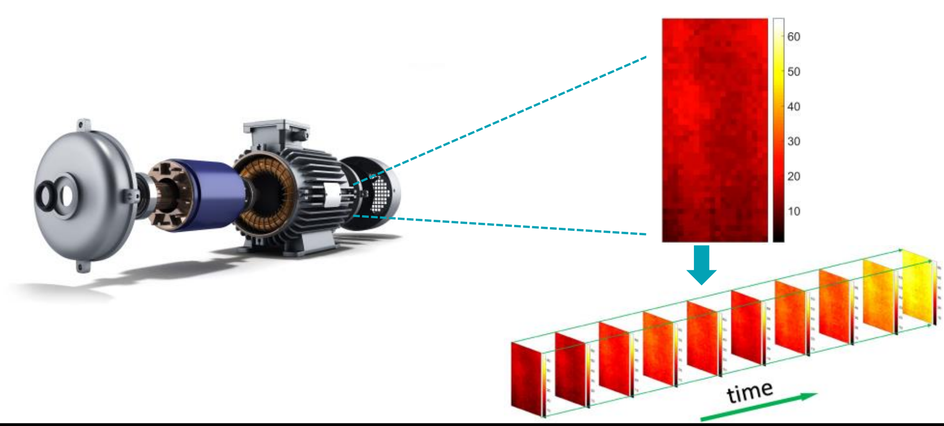
Degradation Modeling using Tensors

- **Degradation**: A gradual process of damage accumulation, which results in failure of engineering systems.
- The degradation process is often captured by a signal or series of data points collected over time.

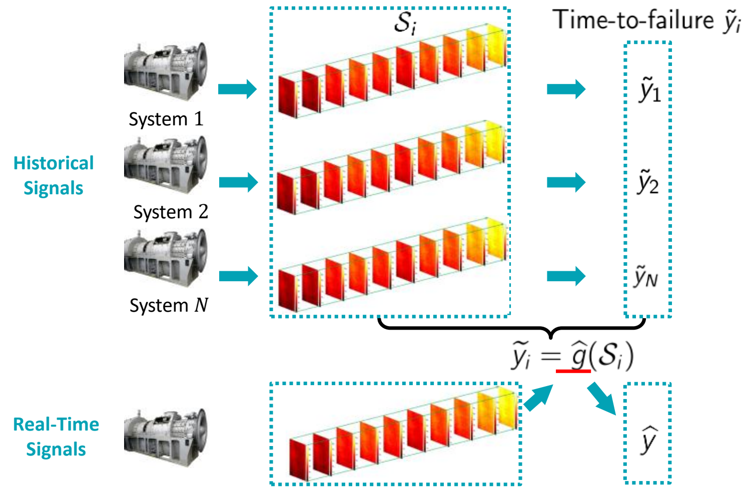


Degradation Analysis using Image Streams

- A degradation image stream that contains the degradation information of a system, can be used to predict the remaining lifetime of a system.



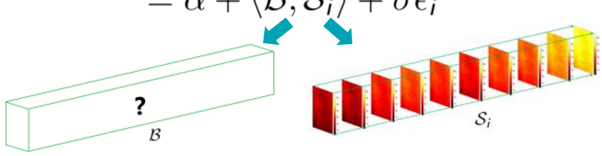
Prediction - Scalar Response and Tensor Predictors



Prediction Model

TTF ← $\boxed{y_i} \sim LLS(\mu_i, \sigma)$

$$y_i = \mu_i + \sigma\epsilon_i$$
$$= \alpha + \langle \mathcal{B}, \mathcal{S}_i \rangle + \sigma\epsilon_i$$



Challenge: High-dimensionality (too many parameters)

$\mathcal{B} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ → $2 + I_1 \times I_2 \times I_3$

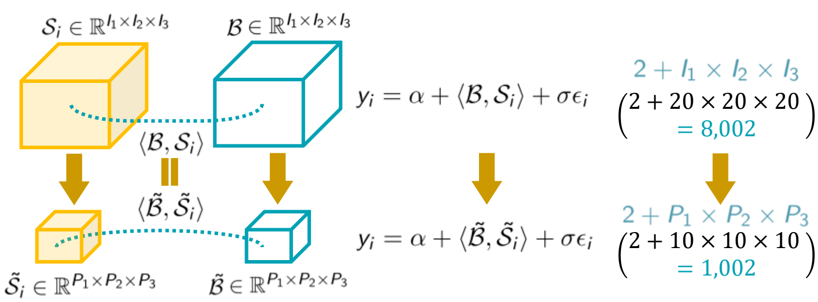
$20 \times 20 \times 20$ → $\left(2 + 20 \times 20 \times 20 = 8{,}002\right)$

Solution: Exploiting the inherent Low-dimensional structure of HD data

Dimension Reduction

Proposition 1

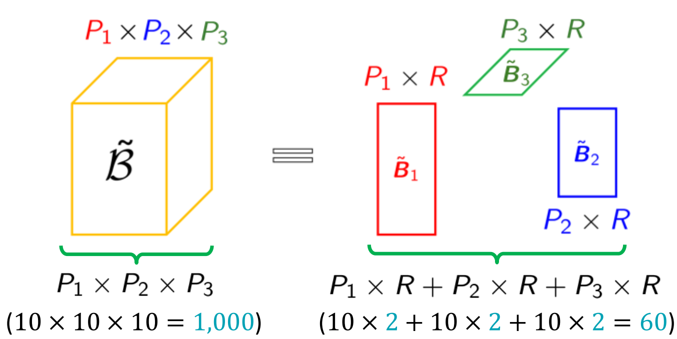
Suppose $\{\mathcal{S}_i\}_{i=1}^N$ can be expanded by $\mathcal{S}_i = \tilde{\mathcal{S}}_i \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times_3 \boldsymbol{U}_3$, where $\tilde{\mathcal{S}}_i \in \mathbb{R}^{P_1 \times P_2 \times P_3}$ is a low-dimensional tensor and matrices $\boldsymbol{U}_d \in \mathbb{R}^{P_d \times I_d}$, $\boldsymbol{U}_d^\top \boldsymbol{U}_d = \boldsymbol{I}_{I_d}$, $P_d < I_d$, $d = 1, 2, 3$. If the coefficient tensor, $\mathcal{B}$, is projected onto the tensor subspace spanned by $\{\boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{U}_3\}$, i.e., $\tilde{\mathcal{B}} = \mathcal{B} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top$, where $\tilde{\mathcal{B}}$ is the projected coefficient tensor, then $\langle \mathcal{B}, \mathcal{S}_i \rangle = \langle \tilde{\mathcal{B}}, \tilde{\mathcal{S}}_i \rangle$.



Further reduce the number of parameters: Decompose $\tilde{\mathcal{B}}$ using
(1) CP decomposition; or
(2) Tucker decomposition

Dimension Reduction using CP

CANDECOMP/PARAFAC (CP) decomposition



Reformulated regression

$$y_i = \alpha + \langle \tilde{\mathcal{B}}, \tilde{\mathcal{S}}_i \rangle + \sigma\epsilon_i \Leftrightarrow y_i = \alpha + \left\langle (\tilde{\boldsymbol{B}}_3 \odot \tilde{\boldsymbol{B}}_2 \odot \tilde{\boldsymbol{B}}_1)\boldsymbol{1}_R, vec(\tilde{\mathcal{S}}_i) \right\rangle + \sigma\epsilon_i$$

Model Estimation using MLE

Maximum likelihood estimation (MLE)

$$\underset{\alpha,\sigma,\tilde{\boldsymbol{B}}_1,\tilde{\boldsymbol{B}}_2,\tilde{\boldsymbol{B}}_3}{\arg\max} \left\{ -N\log\sigma + \sum_{i=1}^{N} \log f\left(\frac{y_i - \alpha - \left\langle (\tilde{\boldsymbol{B}}_3 \odot \tilde{\boldsymbol{B}}_2 \odot \tilde{\boldsymbol{B}}_1)\mathbf{1}_R, vec(\tilde{\mathcal{S}}_i) \right\rangle}{\sigma} \right) \right\}$$

(1) Transfer to a multi-convex optimization problem

$$\tilde{\sigma} = 1/\sigma, \tilde{\alpha} = \alpha/\sigma$$

$$\underset{\boxed{\tilde{\alpha},\tilde{\sigma},\tilde{\boldsymbol{B}}_1,\tilde{\boldsymbol{B}}_2,\tilde{\boldsymbol{B}}_3}}{\arg\max} \left\{ N\log\tilde{\sigma} + \sum_{i=1}^{N} \log f\left(\tilde{\sigma} y_i - \tilde{\alpha} - \tilde{\sigma} \left\langle (\tilde{\boldsymbol{B}}_3 \odot \tilde{\boldsymbol{B}}_2 \odot \tilde{\boldsymbol{B}}_1)\mathbf{1}_R, vec(\tilde{\mathcal{S}}_i) \right\rangle \right) \right\}$$

(2) Block-wise optimization

Model Estimation using MLE

$$\underset{\boxed{\tilde{\alpha},\tilde{\sigma},\tilde{\boldsymbol{B}}_1,\tilde{\boldsymbol{B}}_2,\tilde{\boldsymbol{B}}_3}}{\arg\max} \left\{ N\log\tilde{\sigma} + \sum_{i=1}^{N} \log f\left(\tilde{\sigma} y_i - \tilde{\alpha} - \tilde{\sigma} \left\langle (\tilde{\boldsymbol{B}}_3 \odot \tilde{\boldsymbol{B}}_2 \odot \tilde{\boldsymbol{B}}_1)\mathbf{1}_R, vec(\tilde{\mathcal{S}}_i) \right\rangle \right) \right\}$$

- Iteratively optimize a block of variables given other blocks until convergence

$$\underset{\tilde{\alpha},\tilde{\sigma}}{\arg\max} \left\{ N\log\tilde{\sigma} + \sum_{i=1}^{N} \log f\left(\tilde{\sigma} y_i - \tilde{\alpha} - \tilde{\sigma} \left\langle (\tilde{\boldsymbol{B}}_3 \odot \tilde{\boldsymbol{B}}_2 \odot \tilde{\boldsymbol{B}}_1)\mathbf{1}_R, vec(\tilde{\mathcal{S}}_i) \right\rangle \right) \right\}$$

$$\underset{\tilde{\boldsymbol{B}}_1}{\arg\max} \left\{ N\log\tilde{\sigma} + \sum_{i=1}^{N} \log f\left(\tilde{\sigma} y_i - \tilde{\alpha} - \tilde{\sigma} \left\langle (\tilde{\boldsymbol{B}}_3 \odot \tilde{\boldsymbol{B}}_2 \odot \tilde{\boldsymbol{B}}_1)\mathbf{1}_R, vec(\tilde{\mathcal{S}}_i) \right\rangle \right) \right\}$$

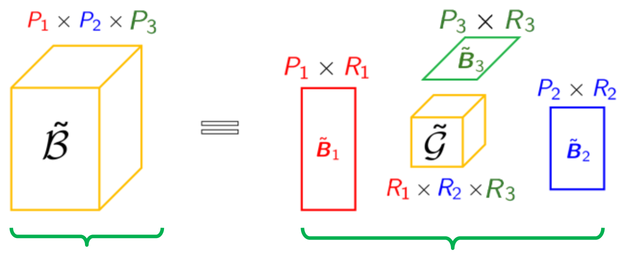
$$\underset{\tilde{\boldsymbol{B}}_2}{\arg\max} \left\{ N\log\tilde{\sigma} + \sum_{i=1}^{N} \log f\left(\tilde{\sigma} y_i - \tilde{\alpha} - \tilde{\sigma} \left\langle (\tilde{\boldsymbol{B}}_3 \odot \tilde{\boldsymbol{B}}_2 \odot \tilde{\boldsymbol{B}}_1)\mathbf{1}_R, vec(\tilde{\mathcal{S}}_i) \right\rangle \right) \right\}$$

$$\underset{\tilde{\boldsymbol{B}}_3}{\arg\max} \left\{ N\log\tilde{\sigma} + \sum_{i=1}^{N} \log f\left(\tilde{\sigma} y_i - \tilde{\alpha} - \tilde{\sigma} \left\langle (\tilde{\boldsymbol{B}}_3 \odot \tilde{\boldsymbol{B}}_2 \odot \tilde{\boldsymbol{B}}_1)\mathbf{1}_R, vec(\tilde{\mathcal{S}}_i) \right\rangle \right) \right\}$$

Rank selection: $BIC = -2\ell(\hat{\boldsymbol{\theta}}) + P\log(N)$

Dimension Reduction using CP

Tucker decomposition



$(10 \times 10 \times 10 = 1{,}000)$

$P_1 \times R_1 + P_2 \times R_2 + P_3 \times R_3 + R_1 \times R_2 \times R_3$

$(10 \times 10 \times 10 = 1{,}000)$ $(10 \times 2 + 10 \times 1 + 10 \times 2 + 2 \times 1 \times 2 = 54)$

Reformulated regression

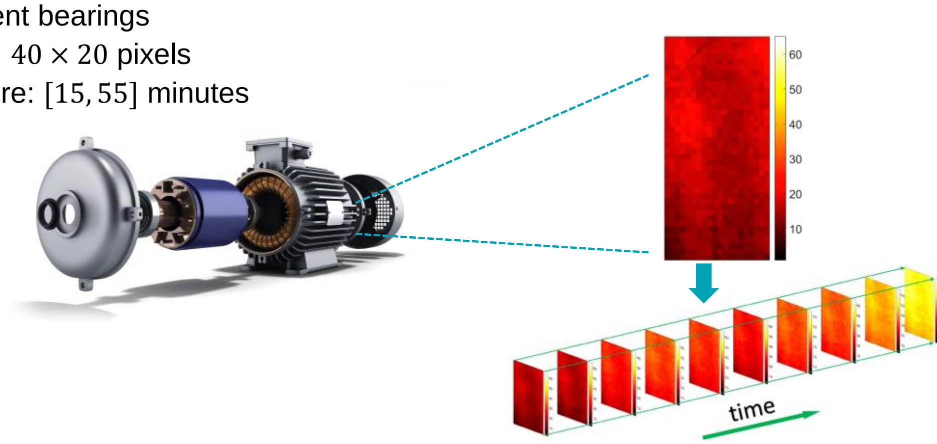
$$y_i = \alpha + \langle \tilde{\mathcal{B}}, \tilde{\mathcal{S}}_i \rangle + \sigma \epsilon_i \Leftrightarrow y_i = \alpha + \langle \tilde{\mathcal{G}} \times_1 \tilde{\boldsymbol{B}}_1 \times_2 \tilde{\boldsymbol{B}}_2 \times_3 \tilde{\boldsymbol{B}}_3, \tilde{\mathcal{S}}_i \rangle + \sigma \epsilon_i$$

Case Study

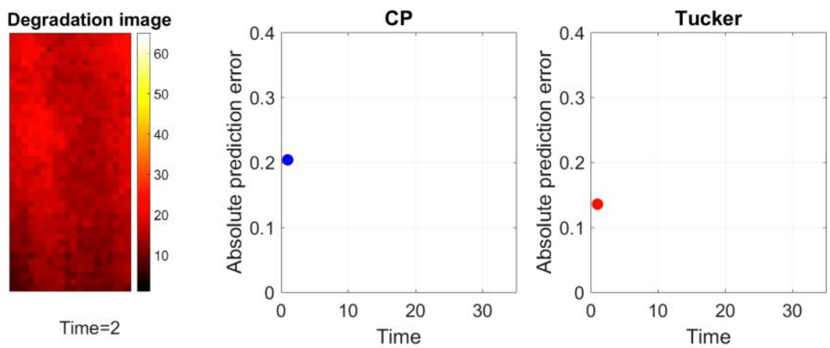
Experiments: degradation tests for rolling element bearings

Each image: 40×20 pixels

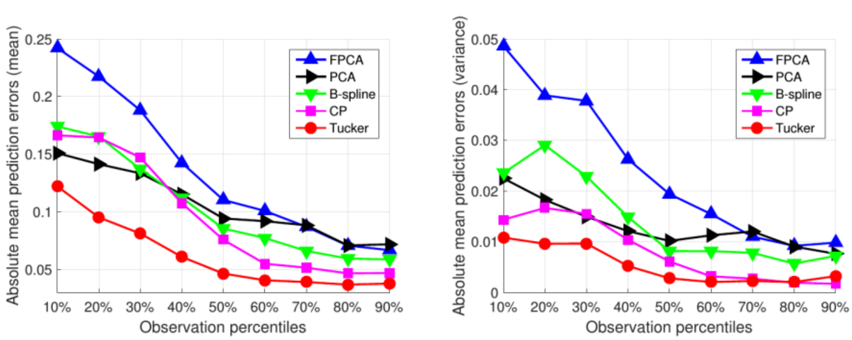
Time-to-failure: $[15, 55]$ minutes



Case Study



Case Study



(a) mean (b) variance

- Both CP-based and Tucker-based models achieved smaller prediction errors than non-tensor-based models
- Tucker-based model performed better than CP-based model

Topics on High-Dimensional Data Analytics

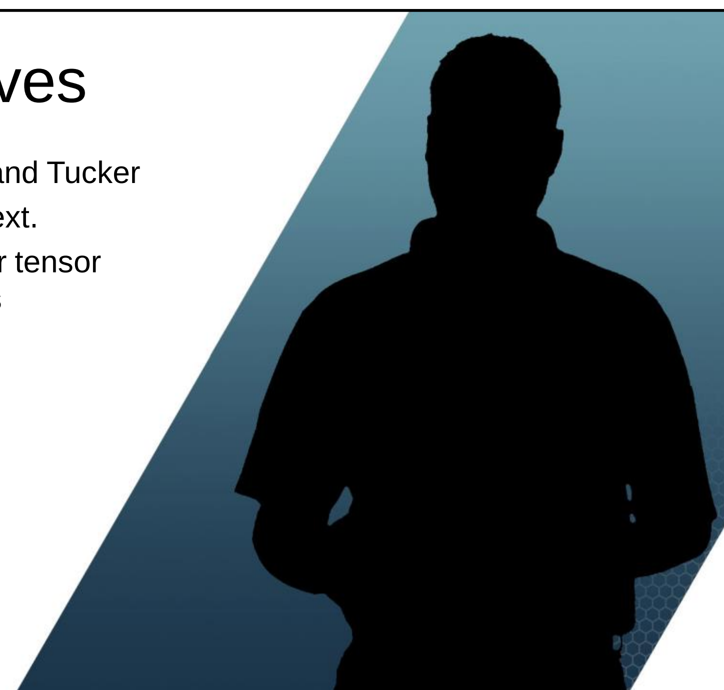
Tensor Data Analysis

Kamran Paynabar, Ph.D.
Associate Professor
School of Industrial & Systems Engineering

Tensor Analysis Applications II

Learning Objectives

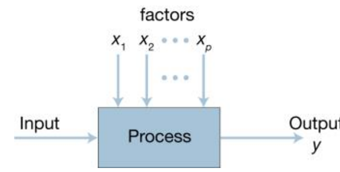
- Determine applications of CP and Tucker
- Use Tucker in regression context.
- Perform regression analysis for tensor response and scalar predictors



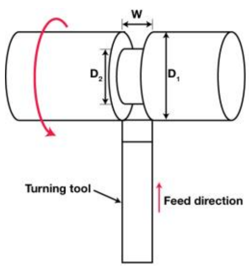
Prediction and Optimization

Scalar Response and Tensor Predictors

Surface shape depends on cutting depth and speed



$$y = f(x_1, x_2, \dots, x_p)$$

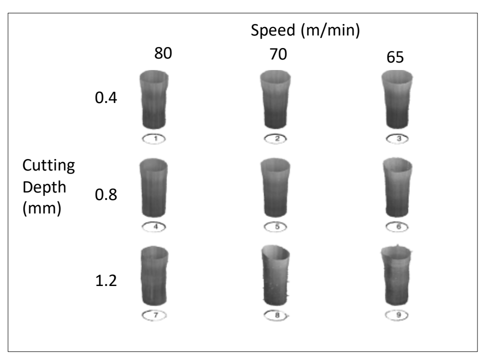


Turning Process   3D CMM machine   *y*: Point Cloud

Prediction and Optimization

	Second Cutting Step	
Ex. No	**depth** (mm)	**speed** (m/min)
1	0.4	80
2	0.4	70
3	0.4	65
4	0.8	80
5	0.8	70
6	0.8	65
7	1.2	80
8	1.2	70
9	1.2	65

→

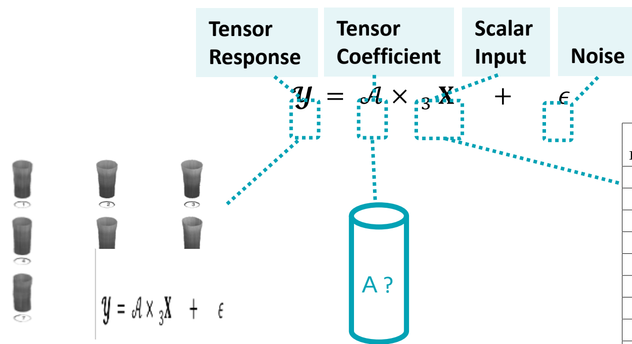


Objective: To build a prediction model for a structured point cloud as a function of controllable factors for the purpose of process optimization.

$$y = f(x_1, x_2)$$

Scalar Response and Tensor Predictors

Tensor response $\mathbf{Y} \in \boldsymbol{R}^{210\times 64\times 90}$ with scalar input $\boldsymbol{X} \in \boldsymbol{R}^{90\times 2}$



Ex. No	Second Cutting Step depth (mm)	speed (m/min)
1	0.4	80
2	0.4	70
3	0.4	65
4	0.8	80
5	0.8	70
6	0.8	65
7	1.2	80
8	1.2	70
9	1.2	65

High-dimensionality challenge

- Number of parameters: A is of dimension $210 \times 64 \times 2 = 26{,}880$.

Tucker Decomposition of Tensor Coefficient

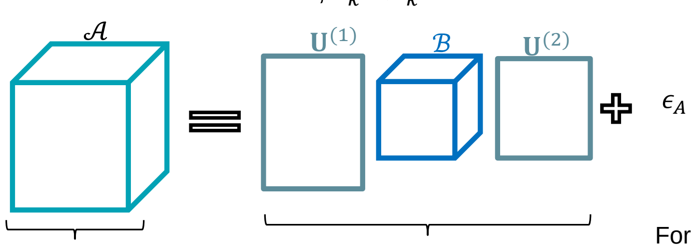
Reduce dimension of $\mathcal{A} \in R^{I_1\times I_2\times p}$ by Tucker Decomposition

Basis expansion:

$$\mathcal{A} = \mathcal{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} + \epsilon_A$$

$\mathbf{U}^{(k)} \in R^{I_k\times P_k}$ is orthogonal factor matrices

$\mathcal{B} \in R^{P_1\times P_2\times p}$ is the core tensor, $P_k < I_k$



$I_1 I_2 p = 210 \times 64 \times 2 = 26{,}880$ $\quad$ $P_1 I_1 + P_2 I_2 + P_1 P_2 p = 556$

For $P_1 = P_2 = 2$

$I_1 = 210, I_2 = 64, p = 2$

Tucker Decomposition and Regression

Tucker Decomposition

$$\mathbf{A} \approx \mathbf{B} \times_1 \mathbf{U}^{(1)} \times \mathbf{U}^{(2)}$$

Tensor Regression

$$\mathbf{Y} = \mathbf{A} \times_3 \mathbf{X} + \epsilon$$

$$\hat{\mathcal{B}} = \underset{\mathcal{B}}{\operatorname{argmin}} \|\mathcal{Y} - \mathcal{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{X}\|_F^2$$

Least Square solution given the factor matrices:

$$\hat{\mathcal{B}} = \mathcal{Y} \times_1 (\mathbf{U}^{(1)^T}\mathbf{U}^{(1)})^{-1}\mathbf{U}^{(1)^T} \times_2 (\mathbf{U}^{(2)^T}\mathbf{U}^{(2)})^{-1}\mathbf{U}^{(2)^T} \times_3 (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$$

Tucker Decomposition and Regression

A two-step approach can be used for estimation of model parameters:

1) Find the core matrix using Tucker

$$\{\hat{\mathcal{S}}, \hat{\mathbf{U}}^{(1)}, \hat{\mathbf{U}}^{(2)}\} = \underset{\mathcal{S},\mathbf{U}^{(1)},\mathbf{U}^{(2)}}{\operatorname{argmin}} \|\mathcal{Y} - \mathcal{S} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)}\|_F^2$$

2) Regress the core tensor on X

$$\hat{\mathcal{B}} = \hat{\mathcal{S}} \times_1 (\hat{\mathbf{U}}^{(1)^T}\hat{\mathbf{U}}^{(1)})^{-1} \times_2 (\hat{\mathbf{U}}^{(2)^T}\hat{\mathbf{U}}^{(2)})^{-1} \times_3 (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$$

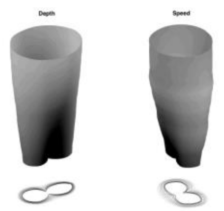
$$\boxed{\hat{\mathcal{B}} = \hat{\mathcal{S}} \times_3 (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T}$$

Case Study

Find optimal basis and coefficient obtained by Regularized Tucker Decomposition

Tensor Response | Tensor Coefficient | Scalar Input | Noise

$$\mathcal{Y} = \mathcal{A} \times_3 \mathbf{x} + \epsilon$$



(a) Tensor regression coefficient $\mathcal{A}$

Case Study – Process Optimization

Objective function: sum of squared differences of the produced mean shape and the uniform cylinder with radius r_t.
Surface roughness $\sigma \leq \sigma_0$

$$\min_{\mathbf{x}} \|\bar{\mathbf{Y}} + \hat{\mathcal{A}} \times_3 \mathbf{x} - r_t\|_F^2 \quad s.t. \sigma \leq \sigma_0, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$$

Optimal settings:
- speed: 80m/min
- cutting depth: 0.8250mm

Reduce shape variation by 65%

Simulated Surfaces with noise under the optimal settings