Tensors in Data Analysis

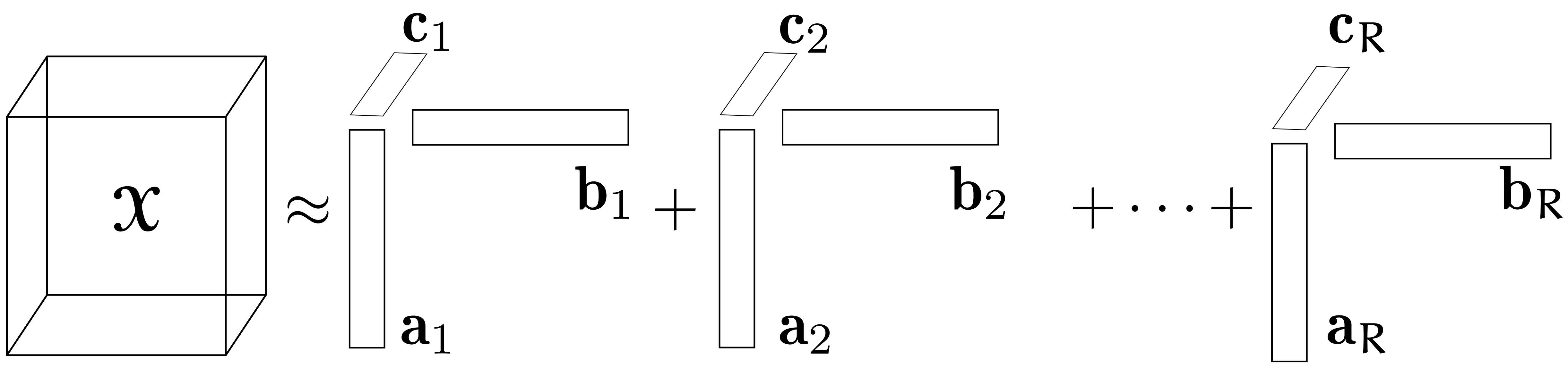
15 May 2014

max planck institut
informatik

Tensors in Data Analysis

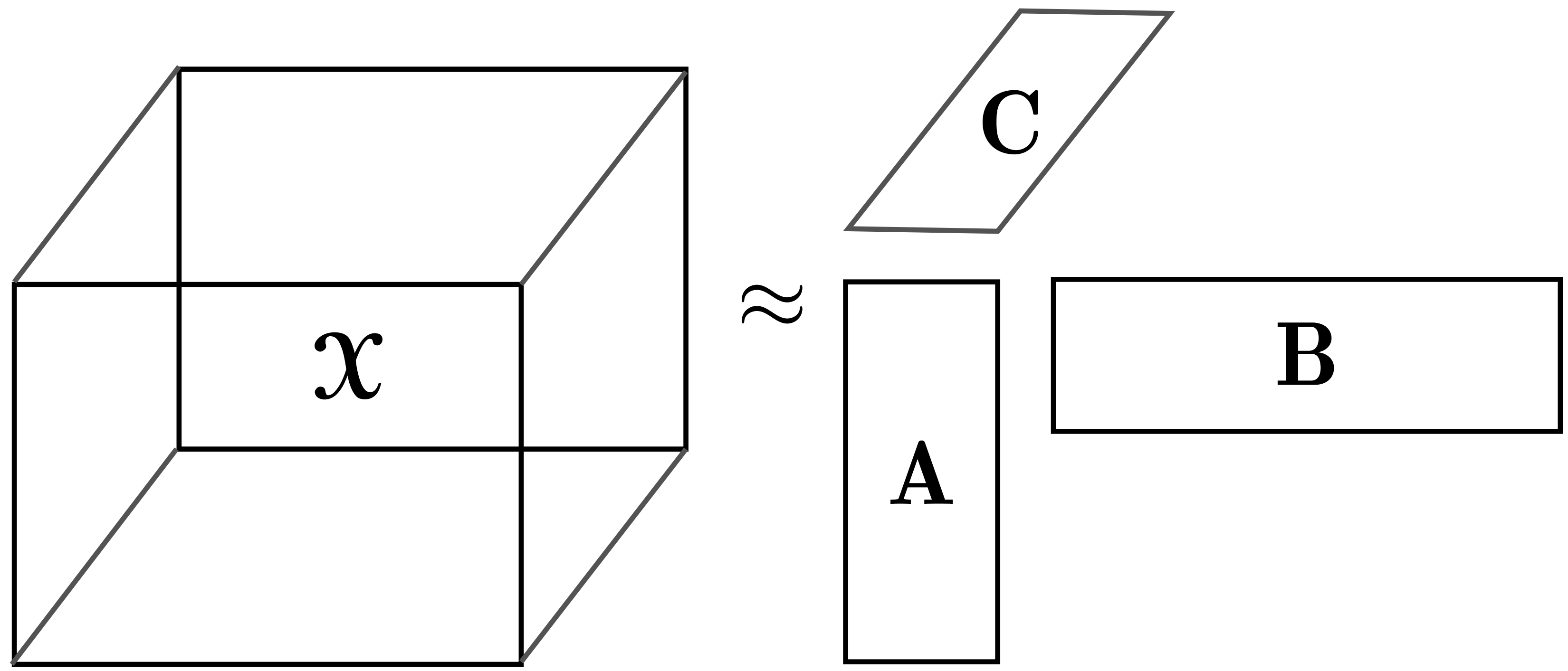
1. CP and INDSCAL and some applications
2. The Tucker tensor decompositions
3. HOSVD, RESCAL, and DEDICOM
4. The non-negative variants

CP Recap (Rank-1 View)



$$x_{ijk} \approx \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}$$

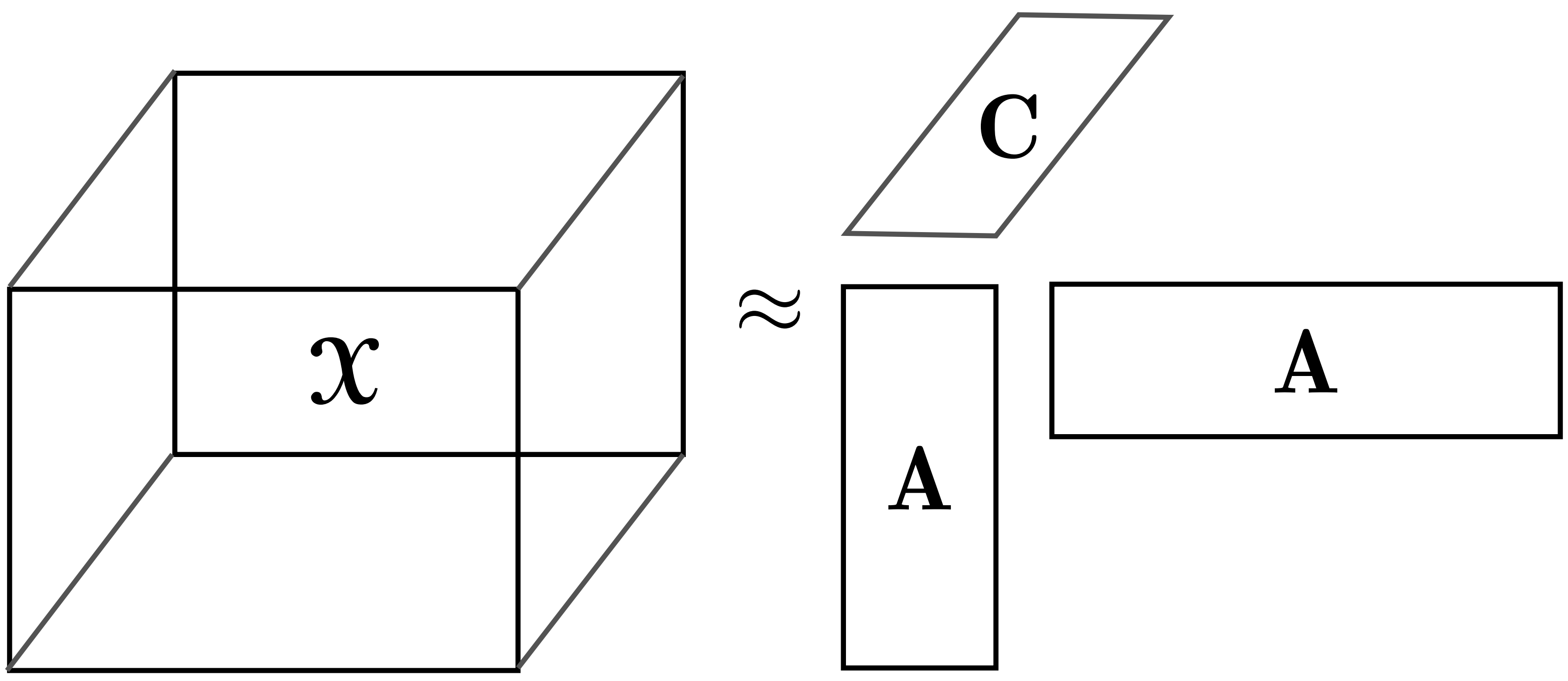
CP Recap (Matrix View)



$$x_{ijk} \approx \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}$$

The INDSCAL Decomposition

- The **INDSCAL decomposition** decomposes a 3-way tensor $\mathcal{X}$ into **two** factor matrices ***A*** and ***C***



$$x_{ijk} \approx \sum_{r=1}^{R} a_{ir} a_{jr} c_{kr}$$

More on INDSCAL

- First two modes of $\mathcal{X}$ are expect to be symmetric
  - Not mandatory, but must have same dimensions
- Commonly computed by solving CP and hoping ***A*** and ***B*** merge
  - End by forcing ***A*** and ***B*** the same and update ***C***

Why INDSCAL

- INDSCAL keeps the symmetry of the modes
- Stands for Individual Differences in Scaling
  - Assume K subjects ranked the similarity of N objects
  - Assume each subject is influenced by the same factors, but with different weights
  - $\boldsymbol{A}$ contains the factors, $\boldsymbol{C}$ gives the weights

INDSCAL Example

- Carroll and Chang (1970) proposed to use INDSCAL and CANDECOMP (CP) to analyse psychological data
  - PCA has long history in psychology
- Example: 20 subjects rate the similarity of countries
  - Multi-way data

Carroll, J.D. & Chan, J.-J., 1970. Analysis of individual differences in multidimensional scaling via an n-way generalization of ”Eckart–Young” decomposition. Psychometrika 35(3), pp. 283–319

Countries in Carroll & Chan (1970) [1]

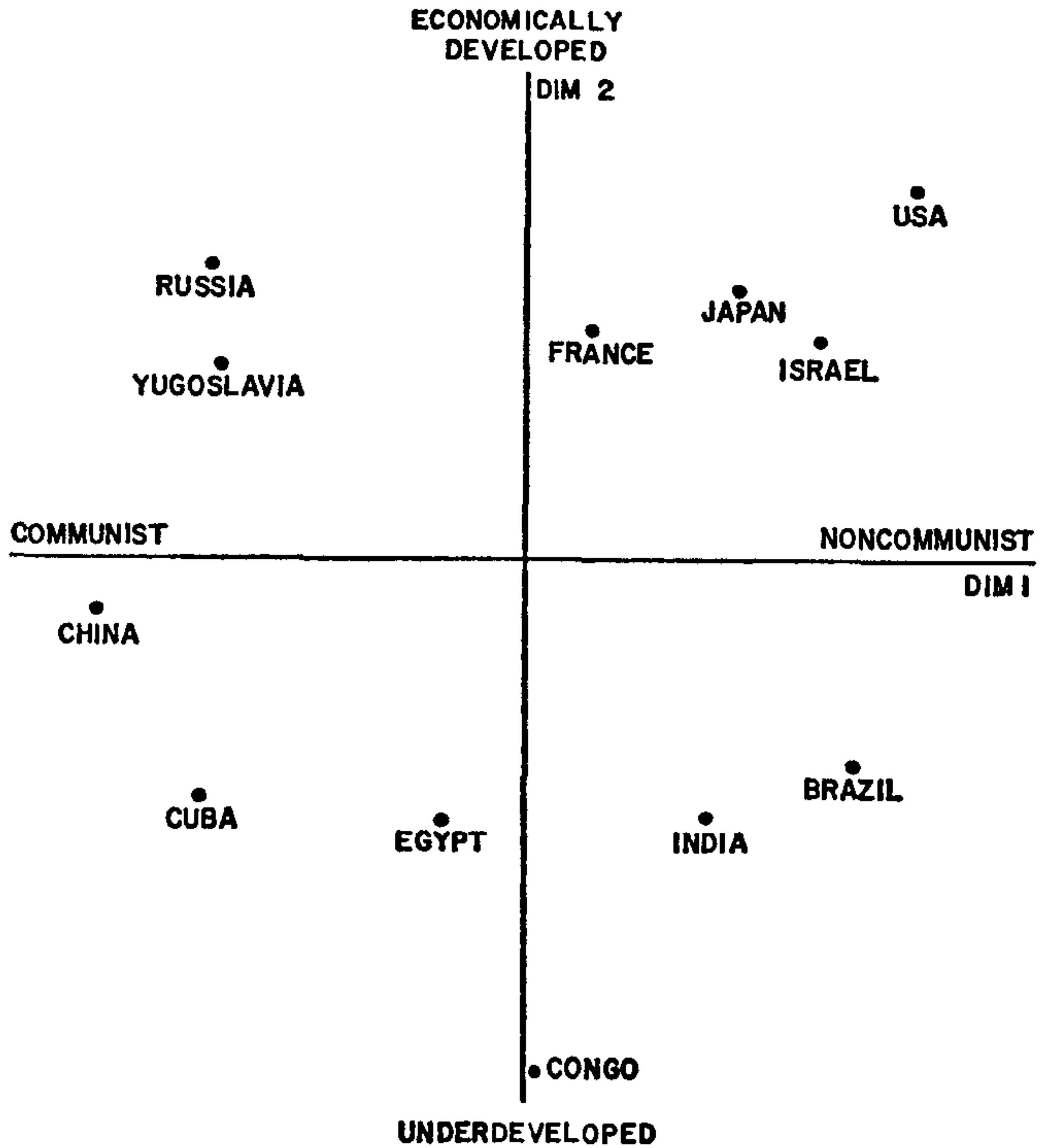


FIGURE 11

Countries in Carroll & Chan (1970) [2]

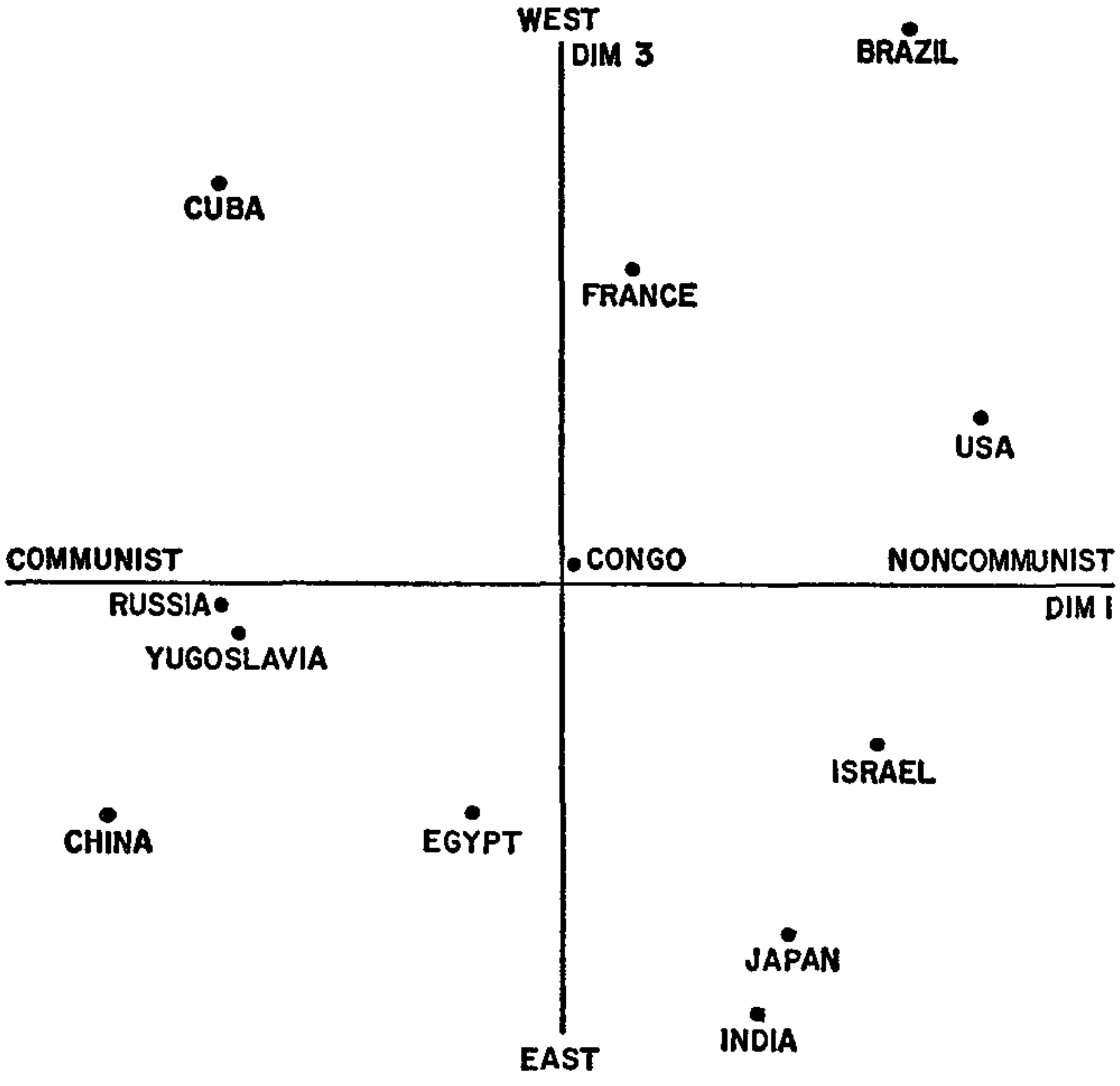


FIGURE 12

Countries in Carroll & Chan (1970) [3]

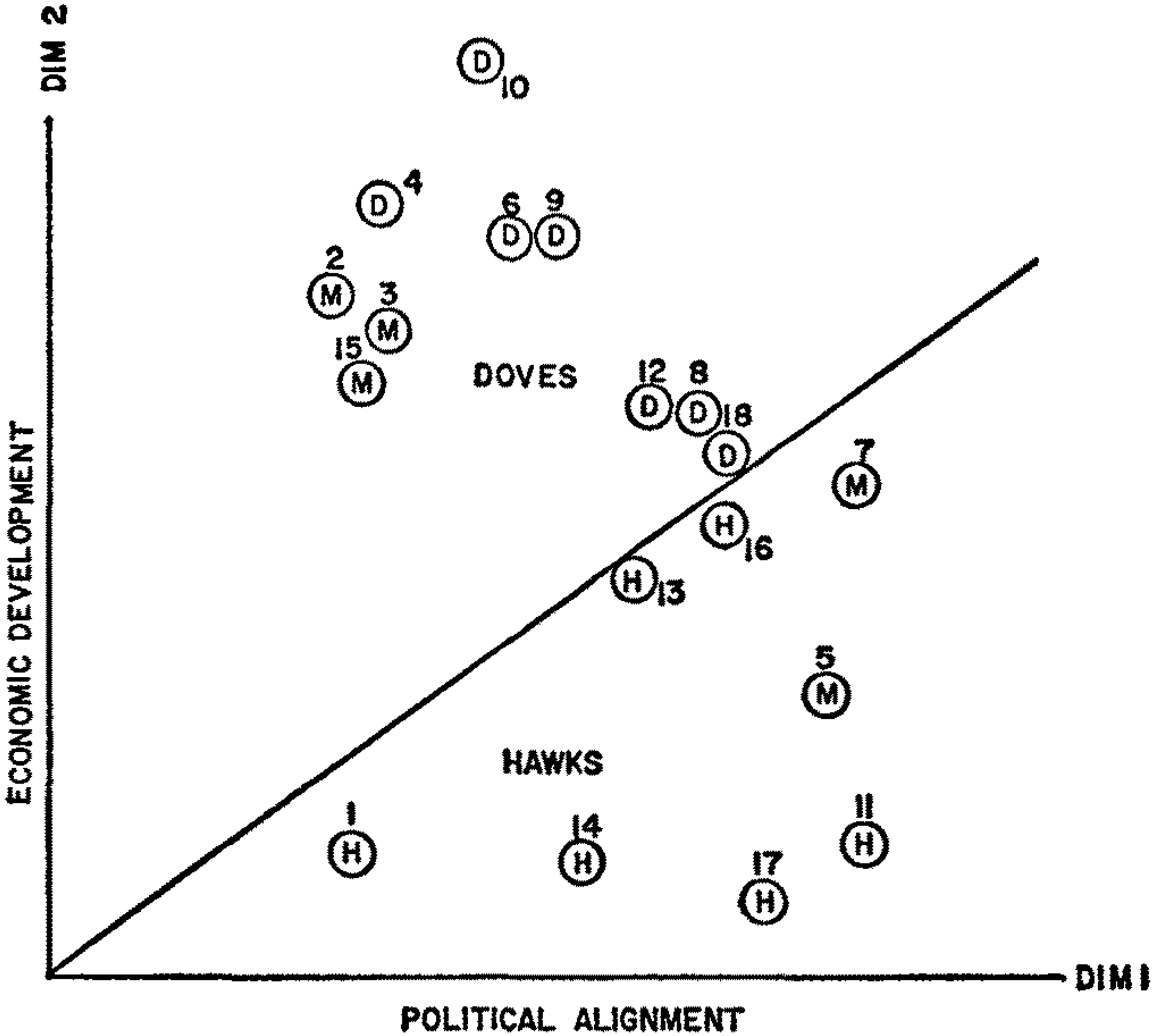


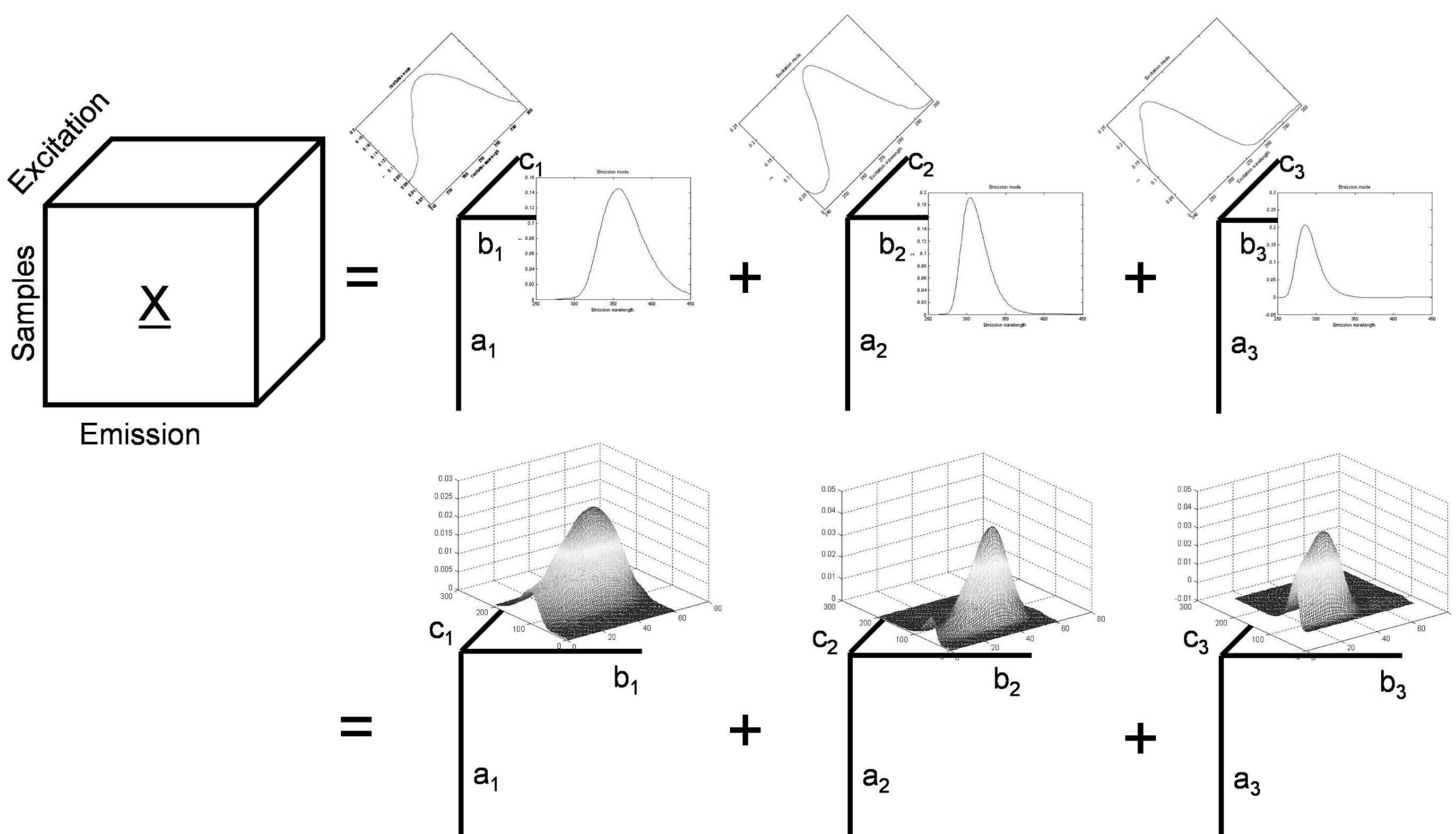
FIGURE 13

Fluorescence Excitation–Emission Analysis

- Fluorescence spectroscopy analyses (typically) organic compounds
  - A beam of (UV) light excites electrons in molecules
  - The excited electrons release a photon, which is measured
  - A **fluorescence landscape** of a compound is a rank-1 matrix that maps the exciter’s wavelength to the emitted photon’s wavelength
    - Lets us to identify the compounds

CP for Fluorescence Analysis

- Samples-by-excitation wavelengths-by-emission wavelengths tensor $\mathcal{X}$
  - Matrix $\boldsymbol{b}_i\boldsymbol{c}_i^T$ is the landscape for the *i*th component
  - Vector $\boldsymbol{a}_i$ gives the weights of landscapes in each sample



Acar, Evrim, and B Yener. 2009. “Unsupervised Multiway Data Analysis: a Literature Survey.” IEEE Transactions on Knowledge and Data Engineering 21(1). pp. 6–20.

TOPHITS for IR

- Three-way pages-by-pages-by-anchor text tensor $\mathcal{T}$
  - Element t_{ijk} = max{1+log(x_{ijk}), 0} where x_{ijk} is the number of times page *i* links to page *j* using term *k*
- The CP decomposition of $\mathcal{T}$ behaves akin to HITS
  - Each rank-1 component is one topic
  - ***A*** and ***B*** give the authority and hub scores, ***C*** gives the weights for terms

Kolda, Tamara G, Brett W Bader, and J P Kenny 2005. “Higher-Order Web Link Analysis Using Multilinear Algebra.” In ICDM '05, 242–49. IEEE. doi:10.1109/ICDM.2005.77.

The Tucker Decompositions

- The CP decomposition requires the factors to have the same number of columns
- In Tucker decompositions, different number of columns can be mixed using a **core tensor**
  - This enables very different looking decompositions

Tensor–Vector Multiplication

- Vectors can be multiplied with tensors along specific modes
  - For *n*-th mode multiplication, the tensor's dimensionality in mode *n* must agree with the vector's dimensions
- The *n*-mode vector product is denoted $\mathcal{X} \bar{\times}_n \boldsymbol{v}$
  - The result is of order *N*–1
  - $(\mathcal{X} \bar{\times}_n \boldsymbol{v})_{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} v_{i_n}$
    - Inner product between mode-*n* fibres and vector $\boldsymbol{v}$

Tensor-Vector Multiplication Example

Given tensor $\mathcal{T}$ and vector $\boldsymbol{v}$,

$$\boldsymbol{T}_1 = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \quad \boldsymbol{T}_2 = \begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix} \qquad \boldsymbol{v} = \begin{pmatrix} 2 & 1 \end{pmatrix}$$

Computing $\mathcal{Y} = \mathcal{T} \bar{\times}_3 \boldsymbol{v}$ gives

$$\mathcal{Y} = \begin{pmatrix} 7 & 13 \\ 10 & 16 \end{pmatrix}$$

Tensor–Matrix Multiplication

- Let $\mathcal{X}$ be an N-way tensor of size $I_1 \times I_2 \times \ldots \times I_N$, and let $\boldsymbol{U}$ be a matrix of size $J \times I_n$
  - The n-mode matrix product of $\mathcal{X}$ with $\boldsymbol{U}$, $\mathcal{X} \times_n \boldsymbol{U}$ is of size $I_1 \times I_2 \times \ldots \times I_{n-1} \times J \times I_{n+1} \times \ldots \times I_N$
  - $(\mathcal{X} \times_n \boldsymbol{U})_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} u_{j i_n}$
    - Each mode-n fibre is multiplied by the matrix $\boldsymbol{U}$
  - In terms of unfold tensors:

$$\mathcal{Y} = \mathcal{X} \times_n \boldsymbol{U} \iff \boldsymbol{Y}_{(n)} = \boldsymbol{U}\boldsymbol{X}_{(n)}$$

Tensor-Matrix Multiplication Example

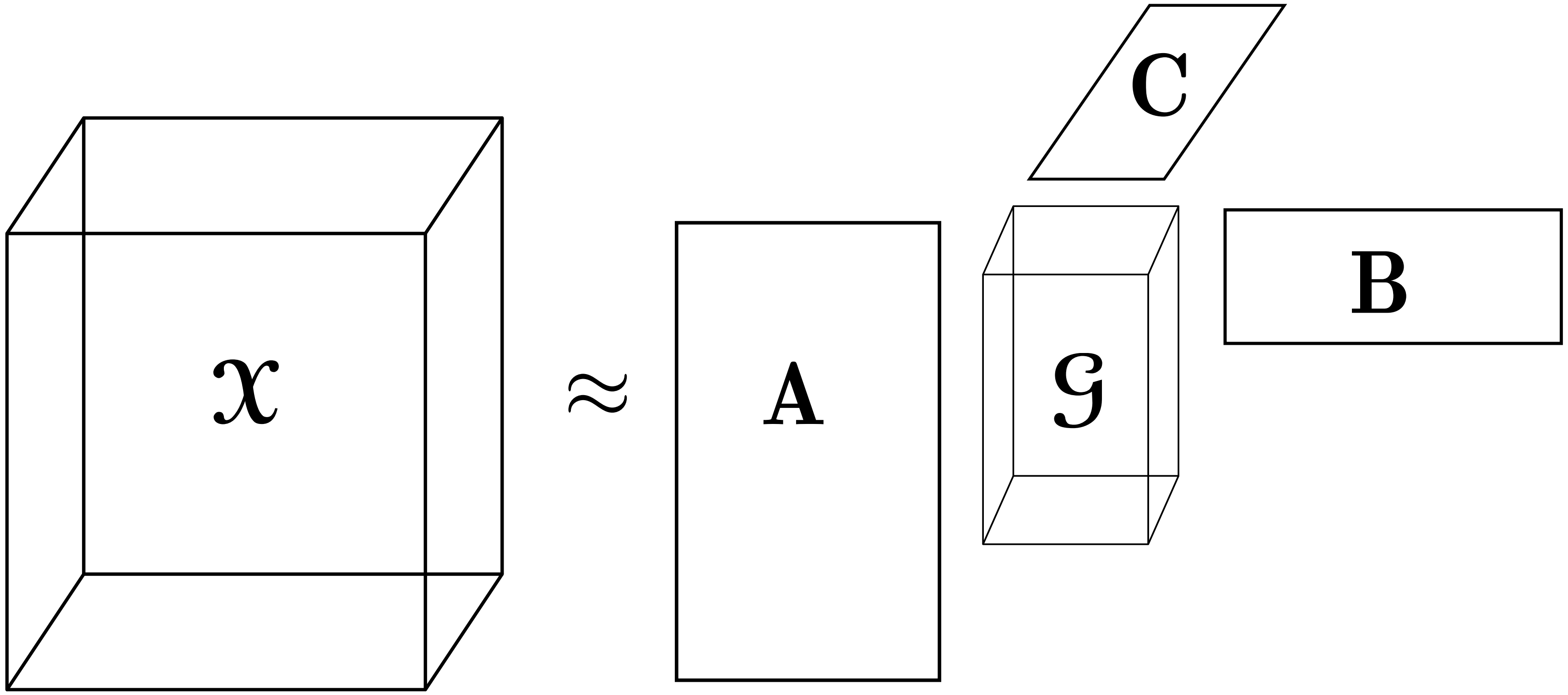
Given tensor $\mathcal{T}$ and matrix $\boldsymbol{M}$,

$$\boldsymbol{T}_1 = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \quad \boldsymbol{T}_2 = \begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix} \qquad \boldsymbol{M} = \begin{pmatrix} 10 & 0 \\ 0 & 100 \\ 1 & 1 \end{pmatrix}$$

Computing $\mathcal{Y} = \mathcal{T} \times_1 \boldsymbol{M}$ gives

$$\boldsymbol{Y}_1 = \begin{pmatrix} 10 & 30 \\ 200 & 400 \\ 3 & 7 \end{pmatrix} \qquad \boldsymbol{Y}_2 = \begin{pmatrix} 50 & 60 \\ 600 & 800 \\ 11 & 15 \end{pmatrix}$$

The Tucker3 Tensor Decomposition



$$x_{ijk} \approx \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} a_{ip} b_{jq} c_{kr}$$

Tucker3 Decomposition

- The **Tucker3 tensor decomposition** decomposes the tensor into three **factor matrices** $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$, and a **core tensor** $\mathcal{G}$
    - $\boldsymbol{A}$ has P, $\boldsymbol{B}$ has Q, and $\boldsymbol{C}$ has R columns and $\mathcal{G}$ is P-by-Q-by-R
- Many degrees of freedom: often $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are required to be orthogonal
- If $P=Q=R$ and core tensor $\mathcal{G}$ is **hyper-diagonal**, then Tucker3 decomposition reduces to CP decomposition

Solving Tucker3

- ALS-style methods are typically used
  - The matricized forms are

$$\boldsymbol{X}_{(1)} = \boldsymbol{A}\boldsymbol{G}_{(1)}(\boldsymbol{C} \otimes \boldsymbol{B})^T$$

$$\boldsymbol{X}_{(2)} = \boldsymbol{B}\boldsymbol{G}_{(2)}(\boldsymbol{C} \otimes \boldsymbol{A})^T$$

$$\boldsymbol{X}_{(3)} = \boldsymbol{C}\boldsymbol{G}_{(3)}(\boldsymbol{B} \otimes \boldsymbol{A})^T$$

- If factor matrices are orthogonal, we can get $\mathcal{G}$ as $\mathcal{G} = \mathcal{X} \times_1 \boldsymbol{A}^T \times_2 \boldsymbol{B}^T \times_3 \boldsymbol{C}^T$

HOSVD, Tucker2, RESCAL, and DEDICOM

- There are many tensor decompositions that are based on or similar to Tucker3
  - Or merge Tucker3 and CP
- Here are few, but the list is by no means exhaustive

Higher-Order SVD (HOSVD)

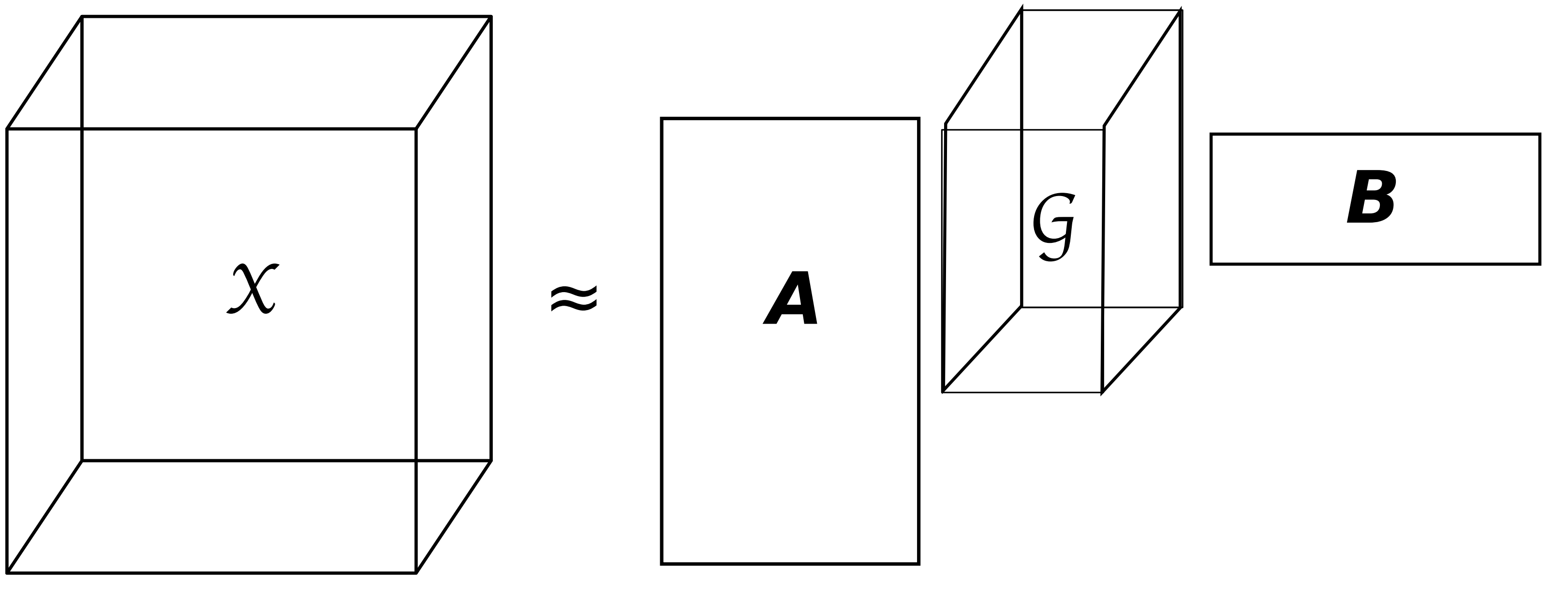
- One method to compute the Tucker3 decomposition
  - Set $\boldsymbol{A}$ as the leading P left singular vectors of $\boldsymbol{X}_{(1)}$
  - Set $\boldsymbol{B}$ as the leading Q left singular vectors of $\boldsymbol{X}_{(2)}$
  - Set $\boldsymbol{C}$ as the leading R left singular vectors of $\boldsymbol{X}_{(3)}$
  - Set tensor $\mathcal{G}$ as $\mathcal{X} \times_1 \boldsymbol{A}^T \times_2 \boldsymbol{B}^T \times_3 \boldsymbol{C}^T$

Why HOSVD?

- Can be used as is for data analysis
  - E.g. TensorFaces
- Can be used to initialize other Tucker3 algorithms
  - Instead of random $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$

Tucker2 Decomposition

- The Tucker2 decomposition decomposes a 3-way tensor into a core tensor and two factor matrices
  - Or, third factor matrix is forced to be an identity matrix
  - Core keeps that mode's dimensionality



Tucker2 Sliced and Matricized

- The slice-wise Tucker2: $\boldsymbol{X}_k = \boldsymbol{A}\boldsymbol{G}_k\boldsymbol{B}^T$ for each k
- Matricized forms replace $\boldsymbol{C}$ with identity matrix $\boldsymbol{I}$: $\boldsymbol{X}_{(1)} = \boldsymbol{A}\boldsymbol{G}_{(1)}(\boldsymbol{I} \otimes \boldsymbol{B})^T$ etc.
- To compute Tucker2:
  - Solve $\boldsymbol{A}$ and $\boldsymbol{B}$ using the matricized forms
  - Update each frontal slice of $\mathcal{G}$ separately

Why Tucker2?

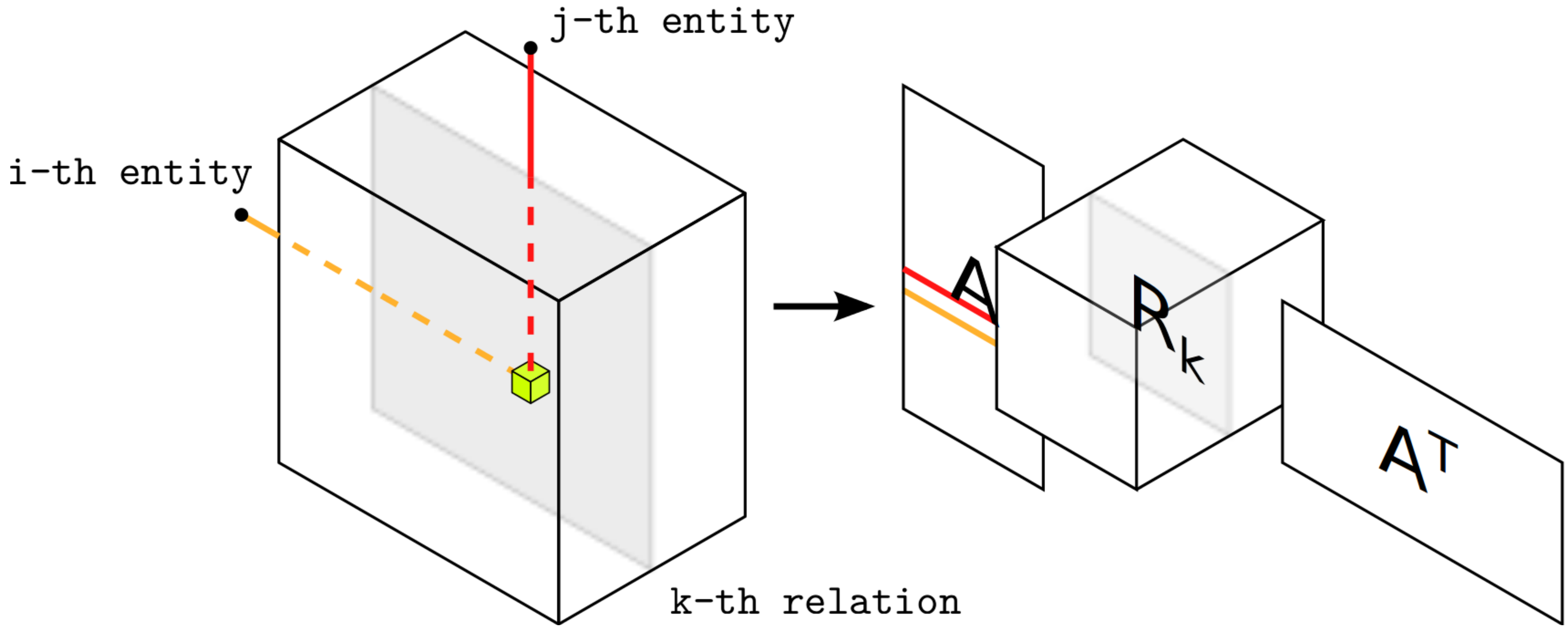
- Use Tucker2 if you don’t want to factorize one mode
  - Too small dimension (e.g. 500-by-300-by-3)
  - This mode requires separate handling
    - E.g. if third mode is time, first Tucker2 and then time-series analysis on third mode
- Tucker2 is slightly simpler than Tucker3

The RESCAL Decomposition

- The **RESCAL decomposition** merges Tucker2 and INDSCAL
- Tensor $\mathcal{X}$ is factored into one factor matrix $\boldsymbol{A}$ and one core tensor $\boldsymbol{R}$
  - $\boldsymbol{X}_k = \boldsymbol{A}\boldsymbol{R}_k\boldsymbol{A}^T$
- Tensor $\mathcal{X}$ might not be symmetric on first two modes

Nickel, M., Tresp, V., & Kriegel, H.-P. (2011). A Three-Way Model for Collective Learning on Multi-Relational Data (pp. 809–816). Presented at the 28th International Conference on Machine Learning.

RESCAL in Picture



Nickel, M., Tresp, V., & Kriegel, H.-P. (2011). A Three-Way Model for Collective Learning on Multi-Relational Data (pp. 809–816). Presented at the 28th International Conference on Machine Learning.

Computing RESCAL (1)

- Mode-1 matricization of RESCAL is $\boldsymbol{X}_{(1)} = \boldsymbol{A}\boldsymbol{R}_{(1)}(\boldsymbol{I} \otimes \boldsymbol{A})^T$
  - This is hard as $\boldsymbol{A}$ is both left and right
  - Simplify: place pairs $(\boldsymbol{X}_k\ \boldsymbol{X}_k^T)$ side-by-side and consider the right $\boldsymbol{A}$ fixed
    - The $\boldsymbol{X}_k^T$ guide $\boldsymbol{A}$ to fit well also in RHS

Computing RESCAL (2)

- To minimize the error, we minimize $||\boldsymbol{Y} - \boldsymbol{AH}(\boldsymbol{I}_{2K} \otimes \boldsymbol{A}^T)||_F$
  - $\boldsymbol{Y} = [\boldsymbol{X}_1\ \boldsymbol{X}_1{}^T\ \boldsymbol{X}_2\ \boldsymbol{X}_2{}^T \ldots \boldsymbol{X}_K\ \boldsymbol{X}_K{}^T]$
  - $\boldsymbol{H} = [\boldsymbol{R}_1\ \boldsymbol{R}_1{}^T\ \boldsymbol{R}_2\ \boldsymbol{R}_2{}^T \ldots \boldsymbol{R}_K\ \boldsymbol{R}_K{}^T]$
- For fixed $\boldsymbol{A}^T$ and $\mathcal{R}$, the update rule for $\boldsymbol{A}$ is

$$\boldsymbol{A} = \left(\sum_{k=1}^{K}(\boldsymbol{X}_k\boldsymbol{R}_k^T + \boldsymbol{X}_k^T\boldsymbol{A}\boldsymbol{R}_k)\right)\left(\sum_{k=1}^{K}(\boldsymbol{B}_k + \boldsymbol{C}_k)\right)^{-1}$$

- Here, $\boldsymbol{B}_k = \boldsymbol{R}_k\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}_k{}^T$ and $\boldsymbol{C}_k = \boldsymbol{R}_k{}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{R}_k$

Computing RESCAL (3)

- Each slice $\boldsymbol{R}_k$ can be updated separately
  - Minimize $||\text{vec}(\boldsymbol{X}_k) - (\boldsymbol{A} \otimes \boldsymbol{A})\text{vec}(\boldsymbol{R}_k)||$
    - Linear regression, set $\text{vec}(\boldsymbol{R}_k) = (\boldsymbol{A} \otimes \boldsymbol{A})^{+}\text{vec}(\boldsymbol{X}_k)$
  - To avoid computing the pseudo-inverse of big $\boldsymbol{A} \otimes \boldsymbol{A}$, compute the skinny QR decomposition of $\boldsymbol{A}$
    - $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{U}$, $\boldsymbol{Q}$ column-orthogonal, $\boldsymbol{U}$ upper-triangular
    - Now: $||\boldsymbol{X}_k - \boldsymbol{A}\boldsymbol{R}_k\boldsymbol{A}^T|| = ||\boldsymbol{X}_k - \boldsymbol{Q}\boldsymbol{U}\boldsymbol{R}_k\boldsymbol{U}^T\boldsymbol{Q}^T||$ $= ||\boldsymbol{Q}^T\boldsymbol{X}_k\boldsymbol{Q} - \boldsymbol{U}\boldsymbol{R}_k\boldsymbol{U}^T||$ and update rule as $(\boldsymbol{U} \otimes \boldsymbol{U})$ which is only R^2-by-R^2

Why RESCAL

- No factorization of the third mode
  - Same as in Tucker2
- Only one factor matrix
  - We assume some kind of symmetry (INDSCAL)
    - E.g. subjects and objects
  - Provides ”information flow” between the modes
- Each frontal slice has a separate ”mixing matrix” for the interactions between factors

The DEDICOM Decomposition: Matrix Version

- The **DEDICOM decomposition** is a matrix decomposition for an asymmetric relation between entities
  - What is the value of export from country *i* to country *j*?
  - How many emails person *i* sent to person *j*?
- $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^T$
  - $\boldsymbol{A}$ factors the entities
  - $\boldsymbol{R}$ explains the asymmetric relation

The DEDICOM Decomposition: Tensor Version

- The **three-way DEDICOM** adds weights for each factor's participation in each position in the third mode
  - E.g. how much country factor r acts as a seller or buyer at time k?
- $\boldsymbol{X}_k = \boldsymbol{A}\boldsymbol{D}_k\boldsymbol{R}\boldsymbol{D}_k\boldsymbol{A}^T$
  - $\boldsymbol{A}$ and $\boldsymbol{R}$ as before, $\mathcal{D}$ is R-by-R-by-K tensor such that each frontal slice $\boldsymbol{D}_k$ is diagonal
  - $(\boldsymbol{D}_k)_{rr}$ is the weight for factor r at time k

DEDICOM in Picture

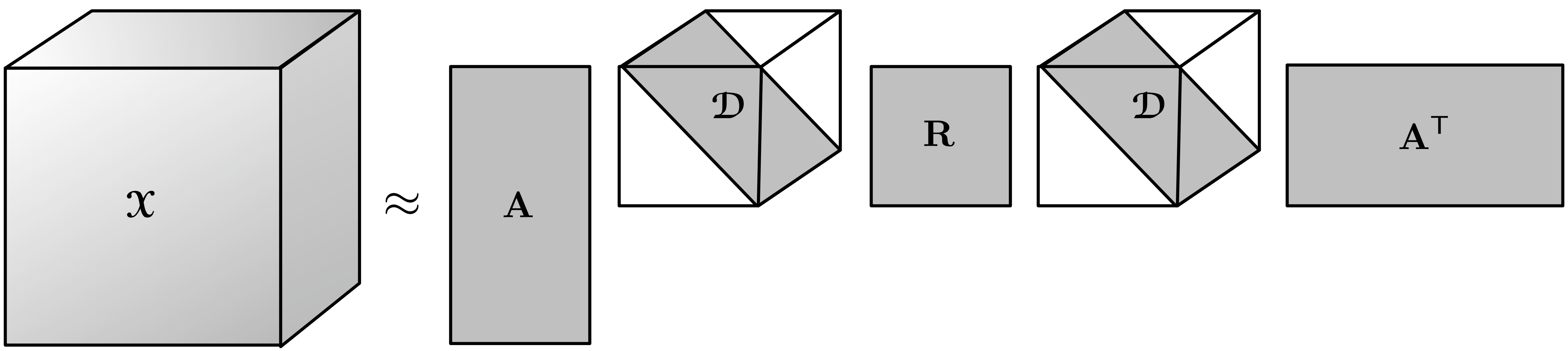


Fig. 5.2: Three-way DEDICOM model.

Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. SIAM Review, 51(3), 455–500.

Computing DEDICOM: ASALSAN (1)

- We want to minimize $\sum_k ||\boldsymbol{X}_k - \boldsymbol{A}\boldsymbol{D}_k\boldsymbol{R}\boldsymbol{D}_k\boldsymbol{A}^T||$
- ASALSAN (Alternating Simultaneous Approximation, Least Squares, and Newton) is one way
  - Stack pairs ($\boldsymbol{X}_k$ $\boldsymbol{X}_k^T$): $\boldsymbol{Y}$=[$\boldsymbol{X}_1$ $\boldsymbol{X}_1^T$ … $\boldsymbol{X}_K\boldsymbol{X}_K^T$]
  - We get $||\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{H}(\boldsymbol{I}_{2K} \otimes \boldsymbol{A}^T)||$ with $\boldsymbol{H} = [\boldsymbol{D}_1\boldsymbol{R}\boldsymbol{D}_1\ \boldsymbol{D}_1\boldsymbol{R}^T\boldsymbol{D}_1 \ldots \boldsymbol{D}_K\boldsymbol{R}\boldsymbol{D}_K\ \boldsymbol{D}_K\boldsymbol{R}^T\boldsymbol{D}_K]$

Bader, B. W., Harshman, R. A., & Kolda, T. G. (2007). Temporal Analysis of Semantic Graphs Using ASALSAN (pp. 33–42). Presented at the 7th IEEE International Conference on Data Mining, IEEE. doi:10.1109/ICDM.2007.54

Computing DEDICOM: ASALSAN (2)

- To update $\boldsymbol{A}$, fix right $\boldsymbol{A}$ and update the left
  $\boldsymbol{A} = \left(\sum_{k=1}^{K} (\boldsymbol{X}_k \boldsymbol{A} \boldsymbol{D}_k \boldsymbol{R}^T \boldsymbol{D}_k + \boldsymbol{X}_k^T \boldsymbol{A} \boldsymbol{D}_k \boldsymbol{R} \boldsymbol{D}_k)\right) \left(\sum_{k=1}^{K} (\boldsymbol{B}_k + \boldsymbol{C}_k)\right)^{-1}$
  - $\boldsymbol{B}_k = \boldsymbol{D}_k \boldsymbol{R} \boldsymbol{D}_k (\boldsymbol{A}^T \boldsymbol{A}) \boldsymbol{D}_k \boldsymbol{R}^T \boldsymbol{D}_k$ and $\boldsymbol{C}_k = \boldsymbol{D}_k \boldsymbol{R}^T \boldsymbol{D}_k (\boldsymbol{A}^T \boldsymbol{A}) \boldsymbol{D}_k \boldsymbol{R} \boldsymbol{D}_k$
- To update R, we use vectors:

$$\min_{\boldsymbol{R}} \left\| \begin{pmatrix} \mathrm{vec}(\boldsymbol{X}_1) \\ \vdots \\ \mathrm{vec}(\boldsymbol{X}_K) \end{pmatrix} - \begin{pmatrix} \boldsymbol{A}\boldsymbol{D}_1 \otimes \boldsymbol{A}\boldsymbol{D}_1 \\ \vdots \\ \boldsymbol{A}\boldsymbol{D}_K \otimes \boldsymbol{A}\boldsymbol{D}_K \end{pmatrix} \mathrm{vec}(\boldsymbol{R}) \right\|$$

- To update $\mathcal{D}$, use Newton's method for each slice $\boldsymbol{D}_k$

DEDICOM vs. RESCAL vs. INDSCAL vs. Tucker2

- RESCAL is a relaxed version of DEDICOM
  - Mixing matrix $\boldsymbol{R}$ is different for each slice
  - Easier to compute as there's no tensor $\mathcal{D}$
    - Algorithm similar to ASALSAN, but simpler
- RESCAL is to Tucker2 what INDSCAL is to CP
  - Share's INDSCAL's equal factor matrix
  - Uses Tucker2's core

The Non-Negative Variants

- Sometimes having non-negative factors is beneficial for data analysis
  - Improved interpretability
    - E.g. physical measurements
  - Sparsity
- All of the discussed methods can be cast into non-negative variants

Non-Negative CP

- The simplest way to compute non-negative CP is to use non-negative least-squares solver with the matricized equations
  - $\min_{\boldsymbol{A} \in \mathbb{R}_{+}^{N \times R}} \left\| \boldsymbol{X}_{(1)} - \boldsymbol{A}(\boldsymbol{C} \odot \boldsymbol{B})^T \right\|$
- Also multiplicative updates are possible
  - $a_{ir} = a_{ir} \frac{(\boldsymbol{X}_{(1)}\boldsymbol{Z})_{ir}}{(\boldsymbol{A}\boldsymbol{Z}^T\boldsymbol{Z})_{ir}}$ with $\boldsymbol{Z} = (\boldsymbol{C} \odot \boldsymbol{B})$
- Also other methods exist

Non-Negative Others

- Also non-negative Tucker[2|3] can be solved using multiplicative update rules
- Non-negative ASALSAN yields non-negative DEDICOM
  - Similar algorithm will work for RESCAL

Summary

- Many, many different tensor decomposition
  - User's responsibility to choose the correct one
    - How do the results look like?
    - What's the time complexity?
- Most algorithms are geared towards dense data
  - But many data analysis data are sparse

Suggested Reading

- In addition to those from last time:
- Acar, E., & Yener, B. (2009). Unsupervised Multiway Data Analysis: A Literature Survey. *IEEE Transactions on Knowledge and Data Engineering*, 21(1), 6–20. doi:10.1109/TKDE.2008.112
    - Shorter and more focused on applications than Kolda & Bader (2009)