Data Mining and Matrices

11 – Tensor Applications

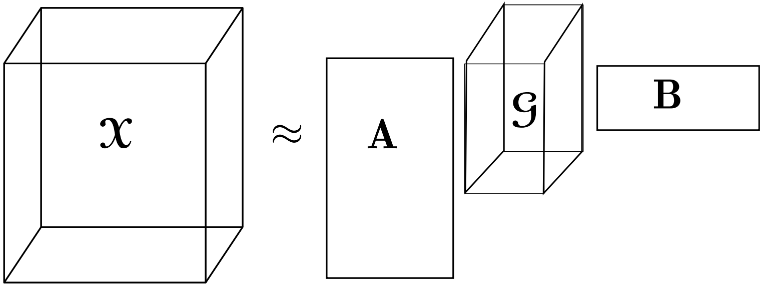
Rainer Gemulla, Pauli Miettinen

July 11, 2013

Outline

Tucker's many decompositions

- Recall: Tucker3 decomposition decomposes a 3-way tensor $\mathcal{X}$ into three factor matrices **A**, **B**, and **C**, and to smaller core tensor $\mathcal{G}$
- **Tucker2 decomposition** decomposes a 3-way tensor into core and **two** factor matrices
  - Equivalently, the third factor matrix is an identity matrix
  - If the original tensor was N-by-M-by-K, the core is I-by-J-by-K or I-by-M-by-J or N-by-I-by-J



Tucker2 sliced and matricezed

- Tucker2 can be presented slice-wise:

$$\mathbf{X}_k = \mathbf{A}\mathbf{G}_k\mathbf{B}^T \quad \text{for each } k$$

  - $\mathbf{X}_k$ is the kth (frontal) slice of $\mathcal{X}$
  - $\mathbf{G}_k$ is the kth (frontal) slice of the core $\mathcal{G}$
  - $\mathbf{A}$ and $\mathbf{B}$ are the factor matrices
- We can also use the normal matricized forms with $\mathbf{C}$ replaced with identity matrix $\mathbf{I}$

$$\mathbf{X}_{(1)} = \mathbf{A}\mathbf{G}_{(1)}(\mathbf{I} \otimes \mathbf{B})^T \quad \text{et cetera}$$

- To compute Tucker2:
  - update $\mathbf{A}$ and $\mathbf{B}$ using the matricized forms
  - update each frontal slice of $\mathcal{G}$ separately

Why Tucker2?

- Use Tucker2 if you don't want to factorize one of your modes
  - Too small dimension (e.g. 500-by-300-by-3)
  - Want to handle this mode separately
    - For example, if third mode is time, we might first do Tucker2 and then time series analysis on $\mathbf{G}_k$s
- Tucker2 is slightly simpler than Tucker3

The INDSCAL decomposition

- Recall: CP decomposition decomposes a 3-way tensor $\mathcal{X}$ into three factor matrices **A**, **B**, and **C**
  - Element-wise: $x_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}$
- The **INDSCAL decomposition** decomposes a 3-way tensor $\mathcal{X}$ into two factor matrices **A** and **C**
  - Element-wise: $x_{ijk} = \sum_{r=1}^{R} a_{ir} a_{jr} c_{kr}$
- $\mathcal{X}$ is expected to be symmetric on first two modes
  - Being symmetric is not absolutely necessary, but first and second mode **must** have the same dimensions
- Common way to compute INDSCAL is to compute normal CP and hope that **A** and **B** merge
  - Last step is to force **A** and **B** equal and to update **C** for the final INDSCAL decomposition

Why INDSCAL?

- If we know two modes are symmetric, INDSCAL won't destroy this structure
- INDSCAL stands for Individual Differences in Scaling
  - Assume K subjects ranked the similarity of N objects
  - Assume the same latent factors explain the similarity decisions by each subject, but different subjects weight different factors differently
  - INDSCAL tries to recover this kind of situation: $\mathbf{A}$ contains the factors explaining the similarities, $\mathbf{C}$ gives the individual scaling of the factors by subjects
  - More on this later...

The RESCAL decomposition

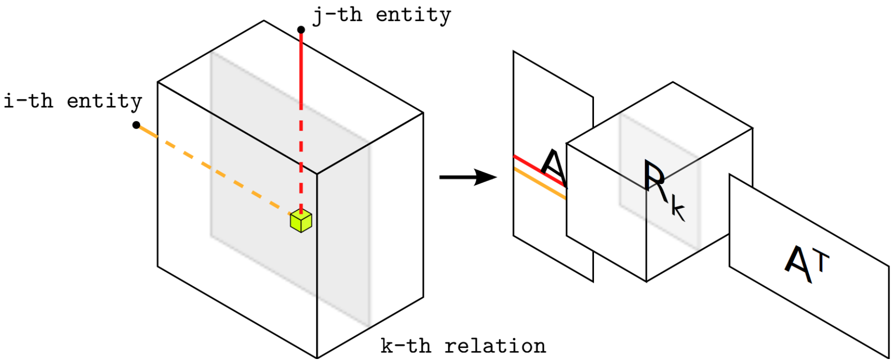
- The **RESCAL decomposition** merges Tucker2 and INDSCAL
- Given an N-by-N-by-K tensor $\mathcal{X}$ and rank R, find an N-by-R factor matrix $\mathbf{A}$ and R-by-R-by-K core tensor $\mathcal{R}$ such that they minimize

$$\sum_{k=1}^{K} \|\mathbf{X}_k - \mathbf{A}\mathbf{R}_k\mathbf{A}^T\|_F^2 \, .$$

- Tensor $\mathcal{X}$ does not have to be symmetric in first two modes
- We can also add regularization

$$\frac{1}{2}\sum_{k=1}^{K} \|\mathbf{X}_k - \mathbf{A}\mathbf{R}_k\mathbf{A}^T\|_F^2 + \frac{\lambda}{2}\left(\|\mathbf{A}\|_F^2 + \sum_{k=1}^{K}\|\mathbf{R}_k\|_F^2\right)$$

RESCAL in picture



Computing RESCAL (1)

- Recall that the mode-1 matricization for Tucker2 is $\mathbf{X}_{(1)} = \mathbf{A}\mathbf{R}_{(1)}(\mathbf{I} \otimes \mathbf{B})^T$
  - In RESCAL, this turns into $\mathbf{X}_{(1)} = \mathbf{A}\mathbf{R}_{(1)}(\mathbf{I} \otimes \mathbf{A}^T)$
  - This is a hard problem, because $\mathbf{A}$ appears on left and right-hand side
  - To simplify, we place slice pairs $(\mathbf{X}_k, \mathbf{X}_k^T)$ side-by-side and consider the right-hand side $\mathbf{A}$ fixed
    - ★ The $\mathbf{X}_k^T$s guide the updated $\mathbf{A}$ to fit well as the right-hand side
- For RESCAL, the minimization problem becomes $\|\mathbf{Y} - \mathbf{A}\mathbf{H}(\mathbf{I}_{2K} \otimes \mathbf{A}^T)\|$
  - $\mathbf{Y} = [\mathbf{X}_1 \quad \mathbf{X}_1^T \quad \cdots \quad \mathbf{X}_K \quad \mathbf{X}_K^T]$
  - $\mathbf{H} = [\mathbf{R}_1 \quad \mathbf{R}_1^T \quad \cdots \quad \mathbf{R}_K \quad \mathbf{R}_K^T]$
- Taking $\mathbf{A}^T$ fixed, the update rule for $\mathbf{A}$ is

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

  - $\mathbf{B}_k = \mathbf{R}_k\mathbf{A}^T\mathbf{A}\mathbf{R}_k^T$ and $\mathbf{C}_k = \mathbf{R}_k^T\mathbf{A}^T\mathbf{A}\mathbf{R}_k$

Computing RESCAL (2)

- Each slice $\mathbf{R}_k$ can be updated separately when minimizing $\sum_{k=1}^{K} \|\mathbf{X}_k - \mathbf{A}\mathbf{R}_k\mathbf{A}^T\|_F^2$
- Writing $\mathbf{X}_k$ and $\mathbf{R}_k$ as vectors, we get

$$\min \|\mathrm{vec}(\mathbf{X}_k) - (\mathbf{A} \otimes \mathbf{A})\mathrm{vec}(\mathbf{R}_k)\|$$

  - Just linear regression, we can solve by setting $\mathrm{vec}(\mathbf{R}_k) = (\mathbf{A} \otimes \mathbf{A})^{\dagger}\mathrm{vec}(\mathbf{X}_k)$
    - But $(\mathbf{A} \otimes \mathbf{A})$ is N^2-by-R^2
- We can compute the skinny QR decomposition of $\mathbf{A}$, $\mathbf{A} = \mathbf{QU}$
  - $\mathbf{Q} \in \mathbb{R}^{N \times R}$ is column-orthogonal and $\mathbf{U} \in \mathbb{R}^{R \times R}$ is upper-triangular
- With QR decomposition, we can re-write the minimization for slice k to $\|\mathbf{X}_k - \mathbf{A}\mathbf{R}_k\mathbf{A}^T\|_F^2 = \|\mathbf{X}_k - \mathbf{QU}\mathbf{R}_k\mathbf{U}^T\mathbf{Q}^T\|_F^2 = \|\mathbf{Q}^T\mathbf{X}_k\mathbf{Q} - \mathbf{U}\mathbf{R}_k\mathbf{U}^T\|$
  - The update rule now has $(\mathbf{U} \otimes \mathbf{U})$ which is only R^2-by-R^2

Why RESCAL?

No factorization of third mode:

- Too small dimension
- Unsuitable for decomposition
- We want to handle that mode separately

Only one factor matrix:

- Models cases where the two modes correspond to same entities
  - sender–receiver–topic
  - subject–object–predicate
- "Information flow"
  - Elements that are similar in one mode are forced similar in the other

Both:

- One global factorization of the first two modes
- Each frontal slice has separate "mixing matrix" for the interactions between the factors

The DEDICOM decomposition

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

DEDICOM in picture

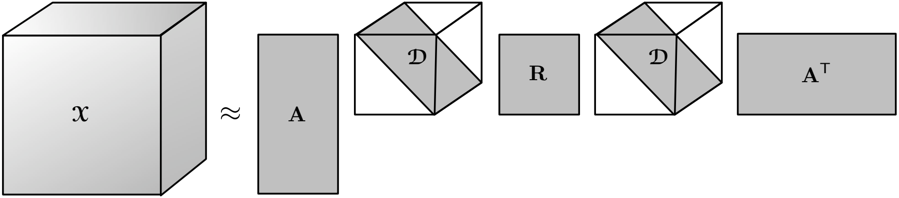


Fig. 5.2: Three-way DEDICOM model.

Computing DEDICOM: ASALSAN (1)

- To compute DEDICOM, we want to minimize $\sum_{k=1}^{K} \|\mathbf{X}_k - \mathbf{A}\mathbf{D}_k\mathbf{R}\mathbf{D}_k\mathbf{A}^T\|$
  - This is hard because $\mathbf{A}$ and $\mathbf{D}_k$ are in left and right-hand side
- The ASALSAN (Alternating Simultaneous Approximation, Least Squares, and Newton) is one way to solve DEDICOM
  - To update $\mathbf{A}$, ASALSAN stacks pairs $(\mathbf{X}_k, \mathbf{X}_k^T)$ next to each other to obtain $\mathbf{Y} = [\mathbf{X}_1\ \mathbf{X}_1^T\ \mathbf{X}_2\ \mathbf{X}_2^T\ \cdots\ \mathbf{X}_k\ \mathbf{X}_k^T]$
  - This gives $\|\mathbf{Y} - \mathbf{A}\mathbf{H}(\mathbf{I}_{2K} \otimes \mathbf{A}^T)\|_F^2$ with $\mathbf{H} = [\mathbf{D}_1\mathbf{R}\mathbf{D}_1\ \mathbf{D}_1\mathbf{R}^T\mathbf{D}_1\ \cdots\ \mathbf{D}_K\mathbf{R}\mathbf{D}_K\ \mathbf{D}_K\mathbf{R}^T\mathbf{D}_K]$
- To compute $\mathbf{A}$, ASALSAN considers left and right $\mathbf{A}$ different, fixes the right and updates the left
  - $\mathbf{A} \leftarrow \left(\sum_{k=1}^{K}(\mathbf{X}_k\mathbf{A}\mathbf{D}_k\mathbf{R}^T\mathbf{D}_k + \mathbf{X}_k^T\mathbf{A}\mathbf{D}_k\mathbf{R}\mathbf{D}_k)\right)\left(\sum_{k=1}^{K}(\mathbf{B}_k + \mathbf{C}_k)\right)^{-1}$
  - Here $\mathbf{B}_k = \mathbf{D}_k\mathbf{R}\mathbf{D}_k(\mathbf{A}^T\mathbf{A})\mathbf{D}_k\mathbf{R}^T\mathbf{D}_k$ and $\mathbf{C}_k = \mathbf{D}_k\mathbf{R}^T\mathbf{D}_k(\mathbf{A}^T\mathbf{A})\mathbf{D}_k\mathbf{R}\mathbf{D}_k$

Computing DEDICOM: ASALSAN (2)

- To update $\mathbf{R}$, we can cast the problem into vector setting

$$\min_{\mathbf{R}} \left\| \begin{pmatrix} \mathrm{Vec}(\mathbf{X}_1) \\ \vdots \\ \mathrm{Vec}(\mathbf{X}_K) \end{pmatrix} - \begin{pmatrix} \mathbf{AD}_1 \otimes \mathbf{AD}_1 \\ \vdots \\ \mathbf{AD}_K \otimes \mathbf{AD}_K \end{pmatrix} \mathrm{Vec}(\mathbf{R}) \right\|$$

  - This is standard regression
- To update $\mathcal{D}$, ASALSAN uses Newton's method for each slice $\mathbf{D}_k$

DEDICOM vs. RESCAL vs. INDSCAL vs. Tucker2

- RESCAL is a relaxed version of DEDICOM
  - The mixing matrix $\mathbf{R}$ is different for every slice
  - It is easier to compute as it doesn't have the tensor $\mathbf{D}$
    - The algorithm is similar to ASALSAN, just simpler
- RESCAL is the Tucker2 version of INDSCAL
  - Shares INDSCAL's equal factor
  - Uses Tucker2's core

Non-negative decompositions

- Simplest way to obtain non-negative CP is to replace the least-squares solver in the matricized equations with a non-negative least-squares solver
  - $\min_{\mathbf{A} \in \mathbb{R}_+^{N \times R}} \|\mathbf{X}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T\|$
- We can also use multiplicative updates as in NMF
  - $a_{ir} \leftarrow a_{ir} \frac{(\mathbf{X}_{(1)}\mathbf{Z})_{ir}}{(\mathbf{A}\mathbf{Z}^T\mathbf{Z})_{ir}}$ with $\mathbf{Z} = (\mathbf{C} \odot \mathbf{B})$
- Other method for non-negative CP exist
- Non-negative Tucker can be done using multiplicative update rules as well
- Non-negative variation of ASALSAN yields non-negative DEDICOM

Outline

Psychology

- Carroll and Chang (1970) proposed the use of tensor decompositions to analyse psychological data
  - Using PCA to find the principal components of person-by-measurement data has long history in psychology
  - But PCA cannot model a matrix of stimuli
  - Example: tones are played to 20 people who rate their similarity, giving tone-by-tone-by-person tensor
  - Another example: country-by-country-by-person
    - Carroll and Chang presented the INDSCAL decomposition for this kind of data
    - In the same paper, they also proposed CANDECOMP
- Before Carroll and Chang, the proposed methods were rather more involved

Countries in Carroll & Chan

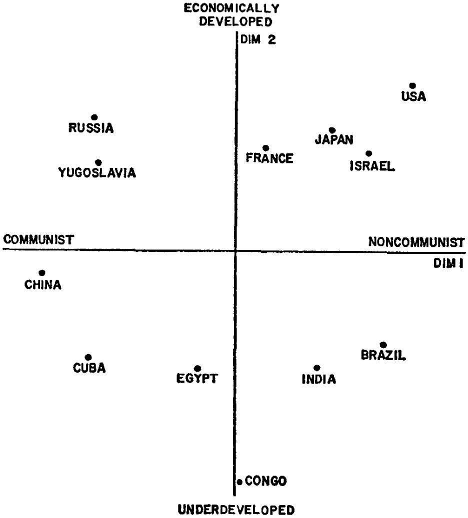


FIGURE 11

Countries in Carroll & Chan

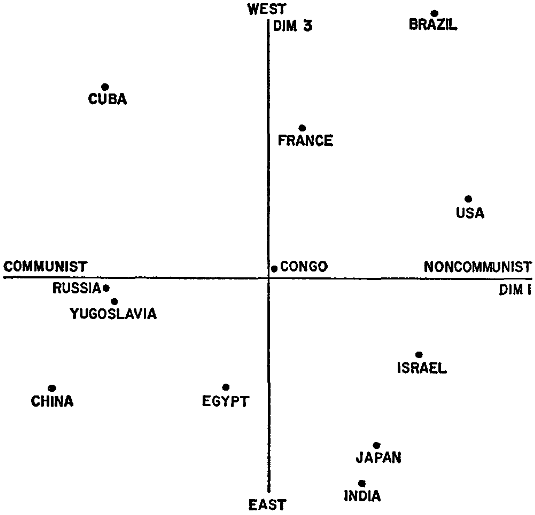


FIGURE 12

Countries in Carroll & Chan

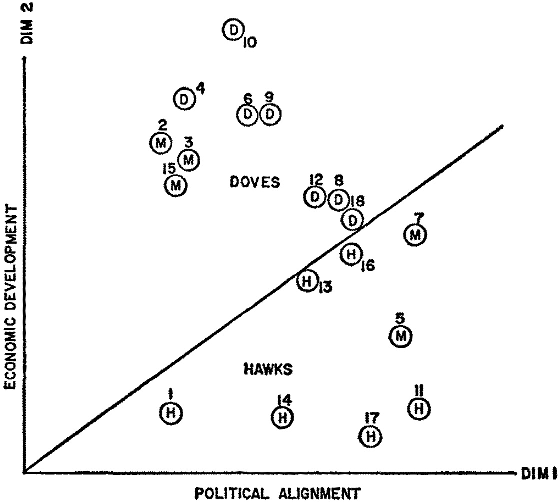
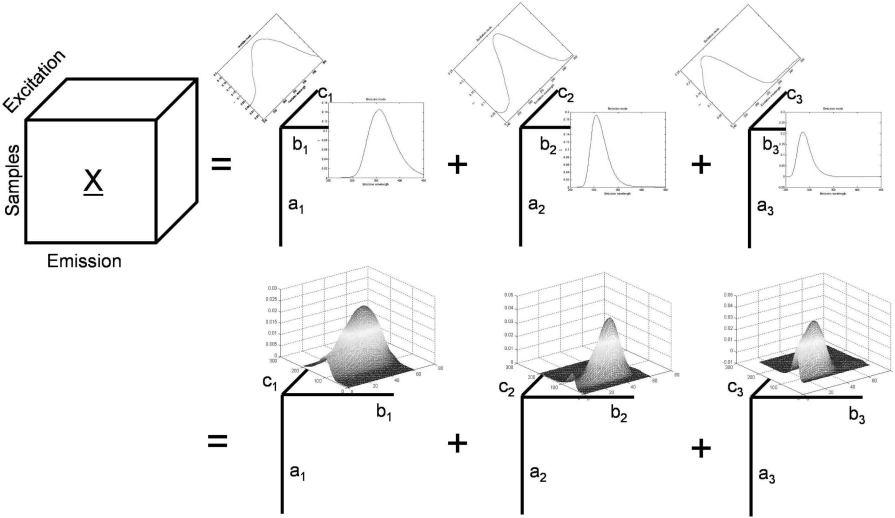


FIGURE 13

Fluorescence excitation-emission analysis

- Fluorescence spectroscopy is a method to analyse (typically) organic compounds
  - A beam of (typically UV) light excites electrons in certain compounds' molecules
  - Later the exited electrons release a photon (light), which can be measured
  - A fluorescence landscape of a compound is a rank-1 matrix that maps the exciter's wavelength to the emitted photon's wavelength
  - The compounds can be identified by the shape of their fluorescence landscape
- We can build a tensor of samples–by–excitation wavelengths–by–emission wavelengths and compute the CP decomposition
  - Matrix $\mathbf{b}_i\mathbf{c}_i^T$ gives the fluorescence landscape for the ith component
  - Vector $\mathbf{a}_i$ explains how much this landscape appears in each sample

Example fluorescence spectoscopy data



RESCAL and subject–object–predicate data

- RESCAL decomposition can be applied to subject–object–predicate data that doesn't have too many predicates
  - The YAGO knowledge base has < 100 relations but millions of entities
  - Also DEDICOM could be applied, but it does not scale as well and the global $\mathbf{R}$'s interpretation is not necessarily obvious
- RESCAL's factor matrix can be used to find similar entities
  - To find entities similar to e in all relations, just order the rows of $\mathbf{A}$ based on their similarity to row e of $\mathbf{A}$
- RESCAL does not help to find similar relations; that would require different tensor decomposition

Mining the 'net: TOPHITS

- We can build a three-way tensor of web pages–by–web pages–by–anchor text to study the link structure and link topics of web pages
  - Build three-way tensor $\mathcal{C}$ such that c_{ijk} is the number of times page i links to page j using term k
  - The non-zero values in $\mathcal{C}$ are scaled to $1 + \log(c_{ijk})$
- The CP decomposition of this tensor behaves akin to HITS
  - In rank-1 CP, **a** gives the hub scores and **b** the authority scores for web pages, while **c** gives the weights for the terms
  - Rank-r CP divides the data in multiple topics, each with its own hubs, authorities, and terms
- Per-topic hubs and authorities can be used for more fine-grained answers

Detecting faces

- NMF and PCA (eigenfaces) are commonly used to decompose (and reconstruct) matrices that correspond to pictures of human faces
  - The PCA of the matrix can be used to classify new pictures as face/non-face by projecting it to the space spanned by the eigenvectors and computing the difference between the projected image and original image
- But matrix-based methods are not good at capturing more than one variation
  - But often we get variable lightning, expressions, poses, etc.
- If we have a complete set of pictures of people under different conditions, we can instead form a tensor and decompose it
  - TensorFaces does HOSVD on tensor that contains pictures of people under different conditions
  - The HOSVD decomposition captures the variation in the conditions better

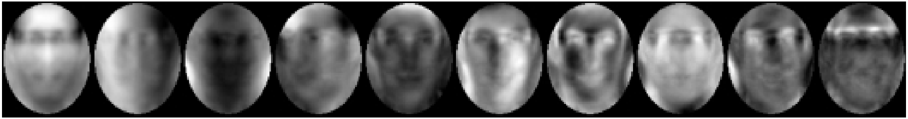
TensorFaces example

- Data: 7943-pixel B&W photographs of 28 people in 5 poses under 3 illumination setups performing 3 different expressions
  - $28 \times 5 \times 3 \times 3 \times 7943$ tensor (10M elements)

All images of one subject

TensorFaces example

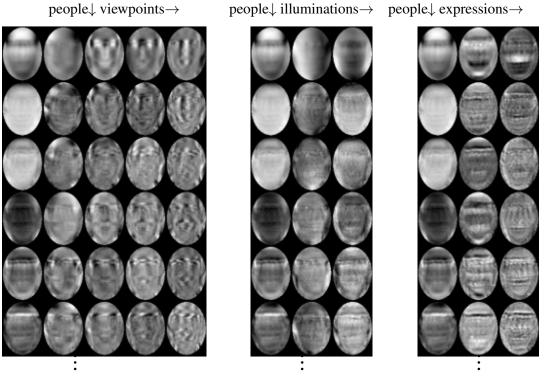
- Data: 7943-pixel B&W photographs of 28 people in 5 poses under 3 illumination setups performing 3 different expressions
  - $28 \times 5 \times 3 \times 3 \times 7943$ tensor (10M elements)



$\mathbf{U}_5$ contains the normal eigenfaces (as it is just the SVD of picture-by-pixels matrix)

TensorFaces example

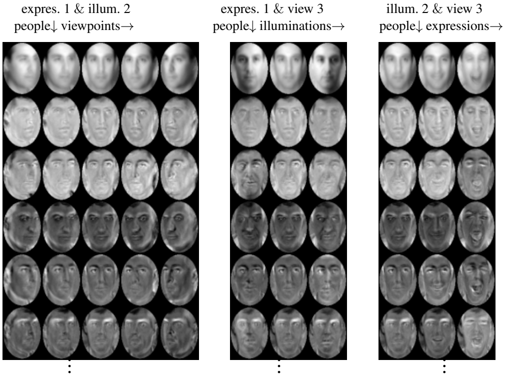
- Data: 7943-pixel B&W photographs of 28 people in 5 poses under 3 illumination setups performing 3 different expressions
  - $28 \times 5 \times 3 \times 3 \times 7943$ tensor (10M elements)



Some visualizations of $\mathcal{G} \times_5 \mathbf{U}_5$ showing the variability across the modes

TensorFaces example

- Data: 7943-pixel B&W photographs of 28 people in 5 poses under 3 illumination setups performing 3 different expressions
  - $28 \times 5 \times 3 \times 3 \times 7943$ tensor (10M elements)



Some visualizations of $\mathcal{G} \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3 \times_4 \mathbf{U}_4 \times_5 \mathbf{U}_5$. The rows are for different people and the columns are for different viewpoints, illuminations, and expressions (with other two modes fixed as indicated).

Outline

Lessons learned

- There are many, many tensor decompositions related to CP and Tucker
  $\rightarrow$ it's the user's responsibility to select the one that's best suited for the task at hand
  $\rightarrow$ consider also the complexity of computing the decomposition
- Tensor decompositions are used in many different fields of science
  $\rightarrow$ sometimes the wheel gets re-invented multiple times
- Most tensor problems are dense
  $\rightarrow$ much less algorithms for finding sparse decompositions of sparse tensors

Suggested reading

- Kolda & Bader *Tensor Decompositions and Applications*, SIAM Rew. 51(3), 2009
  - A great survey on tensor decompositions, includes many variations and applications
- Acar & Yener *Unsupervised Multiway Data Analysis: A Literature Survey*, IEEE Trans. Knowl. Data Eng. 21(1), 2009
  - Another survey, shorter and more focused on applications
- All the papers linked at the bottom parts of the slides